Universidad del Valle de Guatemala

Facultad de Ingeniería

Sistema de Aprendizaje Remoto para el Robot R17

Guatemala

Diciembre 2007

Trabajo de Investigación presentado para optar al grado académico de

Licenciado en Ingeniería Electrónica

Pedro Pablo Alvarado Saenz

John Andrew Carmichael

Pablo Samuel Galindo Martínez

Oscar Enrique García González

Aldo Ivan Garcia Cruz

Julio Fernando Marcucci Fagiani

Julio Cesar Ruiz Herrera

William Andrés Villela Guerra

Trabajo de Investigación presentado para optar al grado académico de

Licenciado en Ingeniería en Ciencias de la Computación

Henry Martinez

Fernando Matzdorf Colina

Sistema de aprendizaje remoto para el robot R17

Universidad del Valle de Guatemala

Facultad de Ingeniería

Sistema de Aprendizaje Remoto para el Robot R17

Guatemala

Diciembre 2007

Vo.Bo.:

(f)_____

MSc. Carlos Tercero Villagrán

Tribunal:

(f)_____

MSc. Carlos Tercero Villagrán

(f)_____

Ing. Marie André Destarac

(f)_____

Ing. Rolando Rodriguez

Fecha de Aprobación: _____

PREFACIO

Durante los últimos diecisiete meses hemos trabajado en el desarrollo del proyecto de aprendizaje remoto para el robot R17. El mismo fue desarrollado en la modalidad de Megaproyecto, ya que permite que varios estudiantes, incluso de distintas carreras, desarrollen una investigación con alto impacto para la sociedad; en este caso, la investigación estuvo dirigida al sector industrial.

Inicialmente no se tenía una aplicación clara, pero durante los primeros cinco meses de trabajo, destinados a la investigación, determinamos que el objetivo más interesante era el de crear un sistema de aprendizaje para un robot. Conscientes de que en Guatemala la robótica aun se encuentra en una etapa de desarrollo primaria, los miembros del proyecto decidimos que el mismo debería tener como objetivo principal apoyar la inserción de la robótica en los procesos industriales que se realizan en nuestro país. A partir de esta decisión, iniciamos la investigación para determinar la manera como podríamos insertar el robot R17 en algún proceso industrial. Por la importancia de los procesos en nuestro medio, tuvo mayor énfasis la investigación de los procesos de soldadura y pintura. Durante esa etapa del proyecto también se creía que el objetivo más adecuado era la inserción del robot R17 en un proceso específico y por lo mismo hubo una etapa de búsqueda de empresas que pudieran interesarse en nuestro desarrollo; finalmente se decidió que el proyecto debía ser una investigación académica que luego debería poder optimizarse, fácilmente, para su implementación comercial y que no fuera, meramente, un desarrollo comercial específico.

Luego de haber concretado los objetivos y alcances que el proyecto tendría, inició la etapa de diseño, la cual tuvo una duración de 4 meses. Ésa etapa fue sumamente

importante porque teniendo mayor información de la robótica en procesos industriales y sabiendo que el proyecto sería una investigación sin fines comerciales, se determinaron los módulos de trabajo para distribuir, adecuadamente, las tareas necesarias.

RESUMEN

El sistema que se ha desarrollado fue dividido en tres módulos generales para una mejor organización del mismo. Cada uno de estos módulos fue subdividido en submódulos para focalizar las tareas en las que cada grupo de trabajo debía concentrarse. Para dividir las tareas y distribuirlas adecuadamente fue necesario crear las fases de trabajo, en donde cada fase es desarrollada por una sola persona. Con la organización anterior fue posible monitorear, en todo momento, el estado del proyecto y las actividades a las que cada miembro se dedicó durante la etapa de desarrollo.

Para este proyecto de investigación y desarrollo fueron propuestos tres módulos de trabajo y se les asignaron los siguientes nombres: Silla Operativa, Comunicaciones y Brazo robótico.

El módulo de Comunicaciones se dividió en tres áreas: Procesamiento de señales, Comunicación Local y Comunicación Remota. Este trabajo de graduación describe el desarrollo de una parte del submódulo de Comunicación Local, fase de recepción, dejando la tarea de transmisión para la fase complementar del submódulo,.

I. INTRODUCCIÓN

Los robots han adquirido una gran importancia en la actualidad; tienen la finalidad de ayudar en diferentes actividades como: trabajos repetitivos, manejo de materiales peligrosos y actividades que superan las capacidades naturales del ser humano. Estas son máquinas reprogramables y tienen limitaciones y no saben operar sin ayuda externa.

Actualmente en nuestro país todos los robots son escasos, en comparación con otros países, ya que no existe una formación que valide su utilización y presentan un alto grado de dificultad al programar su función requerida. El presente trabajo presenta una solución, orientada al seguimiento de los movimientos del programador.

Este sistema consta de seis partes principales: Silla Operativa, los sensores de movimiento, red local, red externa.

Por una parte la silla operativa, al igual que el robot, posee seis grados de libertad, cada sensor da la información del desplazamiento de cada uno ellos. El sistema debe ser capaz de seguir el movimiento del usuario de la silla a fin de obtener la información necesaria para reproducirlos. Por esto, se busca que las actividades de procesamiento se desarrollen suficientemente rápido, es decir, con respuesta inmediata. Así también se busca que la respuesta que proporcione el procesamiento se ajuste a las contingencias del medio, como son las perturbaciones. Respecto a la parte de comunicación con el robot, la acción motora debe ser capaz de interpretar las órdenes enviadas por la información procesada de los sensores del movimiento del objeto.

En la red local se utilizó el protocolo de comunicación CAN. Éste posibilitó interconectar los sensores internos a la silla disminuyendo en gran cantidad el cableado y proveyendo una tasa de transferencia de alrededor de 200kbps. Éste es un protocolo

orientado a mensajes, cada mensaje tiene un identificador único, por lo que garantizó que la información llegara al destinatario correcto.

La red externa utilizó el protocolo WiFi. Esta proveyó la posibilidad de que el procesamiento de comunicación con el robot pudiera obtener la información de los sensores en tiempo real sin necesidad de estar físicamente conectados a un determinado lugar.

En este sistema se utiliza: tres tipos diferentes de sensores, un sistema motor y equipo de cómputo empleado para el procesamiento de la información. Para su diseño se aplican disciplinas como son: análisis de cinemática, dinámica, teoría de control y cómputo en tiempo real.

II. ANTECEDENTES

La robótica ha evolucionado constantemente desde la década de 1930 y cada vez ha conseguido facilitar un poco más nuestras vidas; la variedad de aplicaciones en donde se utiliza crece cada día más y puede ir desde la operación de una mascota electrónica hasta la implementación de los operadores de una industria completa.

Probablemente el área de la robótica que tiene mayor auge en nuestros días es la robótica industrial. Esto ocurre porque, cada vez más, las industrias desean automatizarse para hacer posible al mismo tiempo la eliminación de tareas repetitivas, agobiantes e incluso peligrosas para sus empleados. Sus aplicaciones pueden ser simples, como mover la materia prima de un lugar a otro, o bien pueden ser muy complejas como realizar soldadura de arco o ensamblar componentes electrónicos.

Es importante notar que la robótica actual es más eficiente que los operarios humanos en cuanto a la eficiencia al realizar trabajos repetitivos porque puede trabajar durante más horas y con mayor precisión, además de repetir con más rapidez una misma operación sin agotarse, lo cual tiene como efecto secundario un gran ahorro de materiales y tiempo. Los robots también pueden ser utilizados en lugares peligrosos, reduciendo el riesgo de las vidas de los operarios como ocurre en las centrales de generación de energía nuclear, en los centros de limpieza de desechos tóxicos.

A. Robots

Un robot es un dispositivo electromecánico que puede realizar tareas autónomas o preprogramadas. Puede actuar bajo el control directo de un humano o autónomamente bajo el control de una computadora. Los robots son utilizados para automatizar tareas repetitivas que pueden ser realizadas con mayor precisión por un robot que por un humano.

Robot también puede usarse para describir un dispositivo mecánico inteligente con forma humana. A esta forma de robot se le conoce comúnmente como androide. Estos robots no son comunes en la realidad, debido a las dificultades y los costos de lograr que una máquina bípeda pueda mantener el equilibrio y moverse de una manera parecida a la humana.

La palabra robot también se utiliza en el sentido general para referirse a cualquier máquina que mimetiza las acciones de un humano, en el sentido físico o mental. Proviene de la palabra checa "robota", que quiere decir "trabajo tedioso". Fue acuñada por el checo Karen Capek y popularizada por el escritor Isaac Asimov.

B. Robótica

La robótica se define como la ciencia y tecnología de los robots, su diseño, manufactura y aplicación; esta ciencia requiere un alto conocimiento de electrónica, mecánica e informática. Aunque la apariencia y capacidad de los robots varía mucho, todos los robots comparten las características de una estructura mecánica y móvil bajo alguna clase de control. La estructura de un robot es usualmente mecánica y puede llamarse cadena cinemática; su funcionalidad es afín a la del esqueleto humano. La cadena se forma por eslabones, afines a los huesos; actuadores, afines a los músculos; y articulaciones que permiten uno o más grados de libertad.

La mayoría de los robots contemporáneos utilizan cadenas seriales abiertas en las cuales cada eslabón enlaza los eslabones anterior y posterior a él. Estos robots se llaman robots seriales y son similares al brazo humano. Otras estructuras, como las cadenas cinemáticas paralelas o las que mimetizan la estructura mecánica de los seres vivos, son comparativamente escasas. Sin embargo, el desarrollo de tales estructuras en los robots es un área activa de investigación llamada biomecánica. Los robots manipuladores tienen, por lo general, un manipulador montado en el último eslabón,

que puede ser cualquier herramienta desde un dispositivo de soldadura hasta una mano mecánica.

La estructura mecánica de un robot debe ser controlada para realizar tareas y el control tiene tres fases: percepción, procesamiento y acción. Los sensores proveen información del ambiente o del robot mismo, como la posición de sus articulaciones o de la herramienta en el último eslabón. Ésta información se procesa para calcular las señales apropiadas que se enviarán a los actuadores o motores que mueven la estructura mecánica. Las estrategias de control de un robot son variables, pueden considerar desde la planeación de rutas y reconocimiento de patrones hasta operaciones tan complejas como la inteligencia artificial.

Cualquier tarea involucra el movimiento del robot por lo que el estudio del movimiento se divide en cinemática y dinámica. La cinemática directa se refiere al cálculo de la posición de la herramienta, orientación, velocidad y aceleración cuando se conocen las posiciones de las articulaciones. Cinemática inversa se refiere al caso opuesto, donde se calculan estas posiciones con base en la posición de la herramienta. Una vez que todas las posiciones, velocidades y aceleraciones relevantes se conocen, la dinámica se usa para estudiar los efectos de las fuerzas en estos movimientos. Dinámica directa se refiere al cálculo de aceleraciones una vez las fuerzas aplicadas se conocen y se emplea en la simulación por computadora de los robots. La dinámica inversa se refiere al cálculo de la fuerza necesaria para crear cierta aceleración en la herramienta de trabajo y con esta información se pueden mejorar los algoritmos de control.

Los esfuerzos de la investigación a menudo se enfocan en optimizar o mejorar las estrategias existentes para la interacción entre todas estas áreas. Para ello, es necesario desarrollar criterios para el desempeño óptimo del diseño, estructura y control del robot y luego implementar estos criterios como estrategias.

C. Clases de Robots

a) Robots Industriales

Los robots industriales son aquellos que se emplean para tareas vinculadas con la producción de bienes de origen manufacturado en fábricas o industrias. Dado que es en este grupo que se encuentra el robot R17, la próxima sección está dedicada, por completo, a la explicación de esta clase de robots.

b) Robots Autónomos

La robótica autónoma es el área de la robótica que desarrolla robots capaces de desplazarse y actuar sin intervención humana. El robot debe ser capaz de percibir su entorno y actuar de forma adecuada a él, además de poder completar sus tareas.

Dado que la mayoría de las aplicaciones robóticas dependen de un entorno conocido, el controlador tradicionalmente depende de un mapa detallado para realizar la programación del robot. Sin embargo, mientras decrece la información del entorno, el robot se ve obligado a realizar tareas más complejas, reconociendo su entorno y reaccionando de forma adecuada.

En entornos no estructurados, la solución a través de mapas no es viable por lo que no se utiliza la inteligencia artificial clásica con un controlador centralizado, sino la inteligencia artificial basada en multiagentes, o planteamientos conexionistas usando redes neuronales. Cuando se utilizan algoritmos genéticos en las redes neuronales este proceso se conoce como Robótica Evolutiva.

c) Robots Manipuladores

Estos robots están diseñados para secuencias repetitivas ya que son precisos, rápidos y puede decirse que en general son de alta velocidad; aunque presentan la desventaja de tener una percepción limitada.

El *Robot Institute of America* define que un robot industrial es un manipulador programable y multifuncional, diseñado para mover materiales, piezas, herramientas o dispositivos especiales mediante movimientos variados, programados para la ejecución de distintas tareas.

La morfología de estos robots se puede separar en cuatro áreas: el sistema mecánico, formado por las articulaciones; los actuadores, es decir los motores; los sensores que permiten comunicación, percepción, etc.; y por último, los sistemas de control, es decir los sistemas que generan las trayectorias y planificación para la ejecución de las tareas.

d) Robots de Servicio

Los robots de servicio son los que permitieron llevar la revolución industrial a los hogares, colegios y hospitales; es a este grupo de robots a los que se les conoce como *robots de la tercera revolución industrial*.

Este grupo de robots incrementa su autonomía porque poseen sistemas de navegación automática y por lo tanto pueden planificar, percibir y controlarse en tiempo real; de esa forma, la intervención humana es prácticamente nula.

e) Robots de plataforma móvil y manipulador

Estos robots poseen gran autonomía y un acople de la plataforma al manipulador. Entre estos, se pueden mencionar las grúas, algunos medios de transporte y de asistencia en la minería.

f) Telerobots

Estas maquinas no son realmente robots ya que no son programados sino operados. Al decir que son teleoperados quiere decirse que, es el hombre, a distancia, el responsable de la percepción, planificación y manipulación del robot. Un ejemplo de estos son los brazos utilizados en vehículos de exploración submarina.

D. Robots Industriales

Los robots industriales son definidos por la Organización Internacional de Estandarización (ISO, según la abreviación aceptada internacionalmente) como *controlados automáticamente, re-programables y manipuladores programables en tres o más ejes.* El campo de los robots industriales puede ser definido como el estudio, diseño y uso de sistemas robóticos para la manufactura. (RRG; 2007)

Las aplicaciones típicas de los robots industriales incluyen las tareas de soldadura, pintura, ensamblaje, “pick and place”, inspección de productos y pruebas, las cuales son realizadas con alta precisión y velocidad.

Retomando los avances de manera cronológica, en 1954 George Devol recibió las primeras patentes de robots industriales; junto a Joseph F. Engelberger fundó en 1956 la primera empresa que produjo robots industriales, Unimation, la cual basaba sus modelos en los planos originales de Devol. Los robots de Unimation también eran conocidos como máquinas de transferencia programables porque su función era la de

transferir objetos de un punto a otro, a menos de 5 metros de distancia. Estos robots utilizaban actuadores hidráulicos y se programaban con las coordenadas de las uniones porque se grababan los ángulos de las uniones durante la fase de enseñanza y después eran simplemente reproducidos al ponerlos en operación. La única competencia de Unimation era Cincinnati Milacron Inc., pero esto cambió radicalmente en la década de 1970 cuando varias empresas japonesas comenzaron a producir robots industriales similares. Unimation tenía las patentes para EE.UU., pero no las tenía en Japón y estos se rehusaron a cumplir las normas internacionales de propiedad intelectual y entonces los diseños de Unimation fueron copiados.

En 1969 Víctor Scheinman, en la Universidad de Stanford, inventó el *Stanford Arm*, completamente eléctrico, formado de 6 ejes articulados y diseñado para permitir una solución de brazo. Una solución de brazo es el procedimiento matemático necesario para calcular los ángulos y deslizamientos requeridos por los brazos robóticos para llegar a una posición deseada. Esta solución permitía que el robot siguiera exactamente un camino en el espacio y amplió el potencial de uso del robot en aplicaciones más sofisticadas como ensamblaje o soldadura de arco. Scheinnman también diseñó un brazo para el laboratorio de Inteligencia Artificial de MIT, llamado el *Brazo MIT*. Scheinman vendió sus diseños a Unimation, quienes los desarrollaron con ayuda de General Motors y después los vendieron como una "Máquina Programable Universal para Ensamblaje" (PUMA, por sus siglas en inglés.) En 1973, KUKA Robotics construyó el primer robot industrial, conocido como FAMULUS, siendo éste el primer robot articulado en tener seis ejes controlados electromecánicamente.

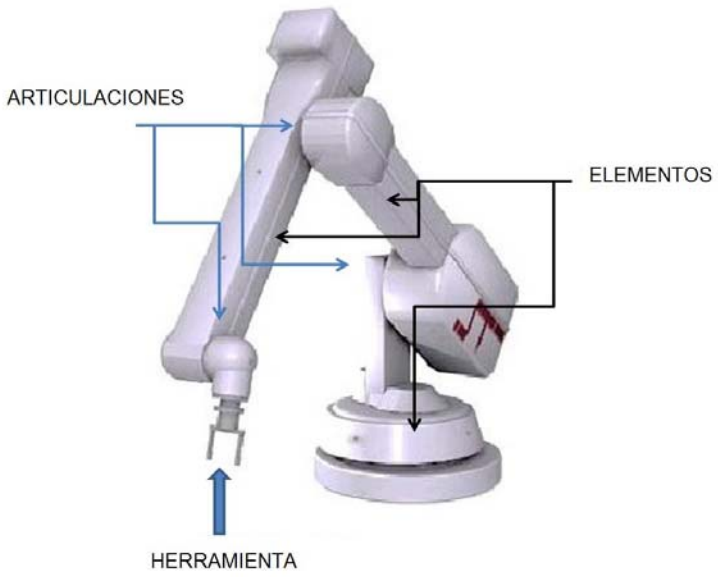
Durante la década de 1970, el interés hacia los robots industriales creció rápidamente, y muchas empresas, incluyendo corporaciones como General Electric y General Motors, entraron al mercado con sus propios productos. Cuando la expansión de la industria de los robots industriales llegó a su punto máximo en 1984, Unimation

fue comprada por Westinghouse Electric Corporation, y después, en 1988, fue revendida a una empresa francesa. Ésta empresa que adquirió Unimation continua haciendo robots industriales para aplicaciones de cuartos limpios y la industria en general e incluso compró la división de robótica de Bosch en 2004.

Hoy en día las empresas japonesas de robótica dominan el mercado, siendo pocas las empresas de otros países que logran imponer una competencia seria.

1) Componentes de un robot

Cabe destacar que la característica antropomórfica más común en nuestros días es la de un brazo mecánico para realizar tareas industriales. El componente principal lo constituye el manipulador y consta de varias articulaciones y sus elementos.



Gráfica 1Componentes de un Robot (ST Robotics, 2006)

Las partes que conforman el manipulador reciben los nombres de: cuerpo, brazo, muñeca y herramienta final, siendo esta, generalmente, un sujetador.

Además del manipulador, los otros elementos que forman parte del robot son un controlador, los mecanismos de entrada y salida de datos y dispositivos especiales.

El controlador del robot, como su nombre lo indica, es el que controla cada uno de los movimientos del manipulador y guarda sus posiciones. El controlador recibe y envía señales a otras máquinas-herramientas y es capaz de almacenar programas para las rutinas que el robot ejecutará.

Los mecanismos de entrada y salida más comunes son: teclado, monitor y caja de comandos llamada "teach pendant"; éste es un dispositivo portátil que se puede utilizar como herramienta de enseñanza para el robot, haciendo mas fácil su programación. Usualmente implementa todos los movimientos posibles del robot y el controlador graba los movimientos que el operario indica.

2) Parámetros técnicos

Existen varios parámetros técnicos que describen a un robot. En la siguiente lista se describen algunos de estos:

1. Número de ejes: dos ejes son necesarios para mover el robot en un plano; para que el robot se mueva en un espacio se necesitan tres ejes y para controlar la orientación de la muñeca del brazo son necesarios tres ejes más. Dependiendo de la aplicación, los robots son diseñados con más o menos ejes y algunos diseños, por ejemplo en robots SCARA, intercambian limitaciones de movimiento por reducción de costos, incremento de velocidad y exactitud.
2. Cinemática: es el arreglo de las partes rígidas y las uniones entre ellas; esta puede ser cartesiana, articulada, paralela y SCARA.
3. Volumen de trabajo: es la región del espacio que el robot puede alcanzar.
4. Capacidad de carga: es la cantidad de peso que el robot puede cargar.

5. Velocidad: indica qué tan rápido se puede posicionar el extremo del brazo robótico.
6. Exactitud: indica qué tan cerca se puede posicionar el robot del punto deseado. La exactitud puede variar con la velocidad y la posición en el espacio de trabajo.
7. Control de movimiento: en ciertas aplicaciones, como "*pick and place*", el robot tiene que moverse entre una posición de origen y la posición final pero, en aplicaciones como la soldadura de arco, el movimiento debe ser monitoreado continuamente para seguir un camino en el espacio, con orientación y velocidad controladas.
8. Fuente de potencia: algunos robots utilizan motores eléctricos, otros utilizan acusadores hidráulicos dependiendo de las tareas que deban ejecutar.
9. Transmisión: Algunos robots utilizan engranajes o bandas para conectarse a las uniones mientras que otros utilizan conexiones directas (transmisión directa, o *Direct Drive*).

3) Grado de complejidad de los robots

a) Robots de primera generación

Estos son dispositivos que actúan como esclavos mecánicos del hombre, quien debe intervenir directamente para controlar los dispositivos de movimiento. Esta intervención ocurre gracias a los mecanismos operados con las extremidades superiores del humano. Entre las aplicaciones más comunes se encuentra la manipulación de materiales radiactivos y la obtención de muestras submarinas.

b) Robots de segunda generación

El dispositivo actúa automáticamente sin intervención humana frente a posiciones fijas en las que el trabajo ha sido preparado y ubicado adecuadamente para que el robot ejecute movimientos repetitivos en el tiempo, obedeciendo lógicas combinatorias o secuenciales, o bien respondiendo a las instrucciones de programadores paso a paso o controladores lógico-programables (PLC). Un aspecto muy importante de estos robots es la facilidad de rápida re-programación, lo que los convierte en unidades versátiles, cuyo campo de aplicación no sólo se encuentra en la manipulación de materiales sino en todos los procesos de manufactura. Entre estos se puede mencionar el estampado en frío y en caliente, la asistencia a las máquinas y herramientas para la carga y descarga de piezas; también se encuentran en la inyección de termoplásticos y metales no ferrosos, en los procesos de soldadura continua y por puntos, en tareas de pintura y de reemplazo de algunas máquinas convencionales.

c) Robots de tercera generación

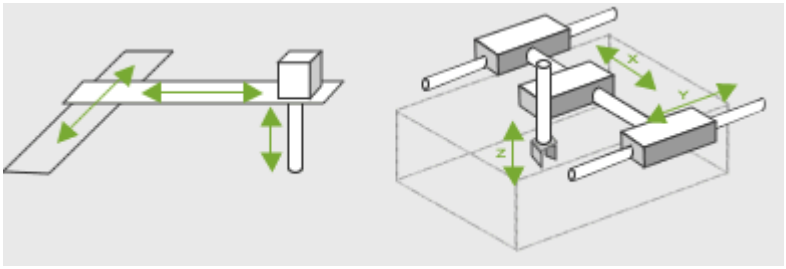
Son dispositivos que habiendo sido construidos para alcanzar determinados objetivos, serán capaces de elegir la mejor forma de hacerlo teniendo en cuenta el ambiente que los rodea. Para obtener estos resultados es necesario que el robot posea algunas condiciones que posibiliten su interacción con el ambiente y los objetos. Las mínimas aptitudes requeridas son: capacidad de reconocer un elemento determinado en el espacio y la capacidad de adoptar propias trayectorias para conseguir el objetivo deseado. Los métodos de identificación empleados hacen referencia a la utilización de imágenes para la observación de objetos, sin embargo, no puede asegurarse que la que solución natural para el hombre sea la mejor solución para el robot.

4) Configuración de los robots

Cuando se habla de la configuración de un robot, se habla de la forma física que se le ha dado al brazo del robot. El brazo del manipulador puede presentar cuatro configuraciones clásicas: la cartesiana, cilíndrica, polar y angular.

a) Configuración cartesiana

Posee tres movimientos lineales, es decir, tiene tres grados de libertad, los cuales corresponden a los movimientos localizados en los ejes X, Y y Z. Los movimientos que realiza este robot entre un punto y otro se basan en interpolaciones lineales. Interpolación, en este caso, se refiere al tipo de trayectoria que realizará el manipulador cuando se desplace entre los puntos de origen y destino. A la trayectoria realizada en línea recta se le conoce como interpolación lineal y a la trayectoria hecha de acuerdo con el tipo de movimientos que tienen sus articulaciones se le llama interpolación por articulación. (Loop Technology Limited, 2007)

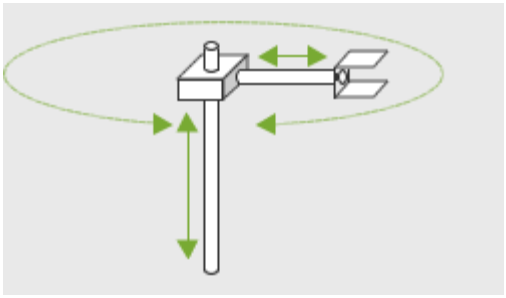


Gráfica 2 Robot con configuración Cartesiana (Loop Technology Limited, 2007)

b) Configuración cilíndrica

Puede realizar dos movimientos lineales y uno rotacional, o sea que presenta tres grados de libertad. El robot de configuración cilíndrica está diseñado para ejecutar los movimientos conocidos como interpolación lineal e interpolación

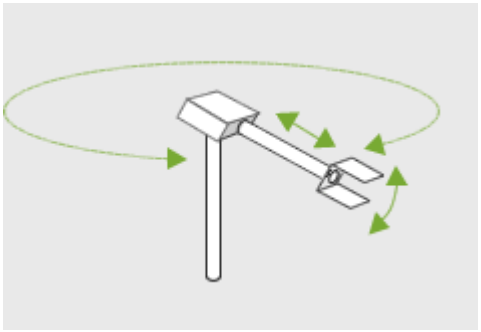
por articulación; la última se lleva a cabo por medio de la primera articulación ya que ésta puede realizar un movimiento rotacional. (Loop Technology Limited, 2007)



Gráfica 3 Robot con configuración Cilíndrica (Loop Technology Limited, 2007)

c) Configuración Polar

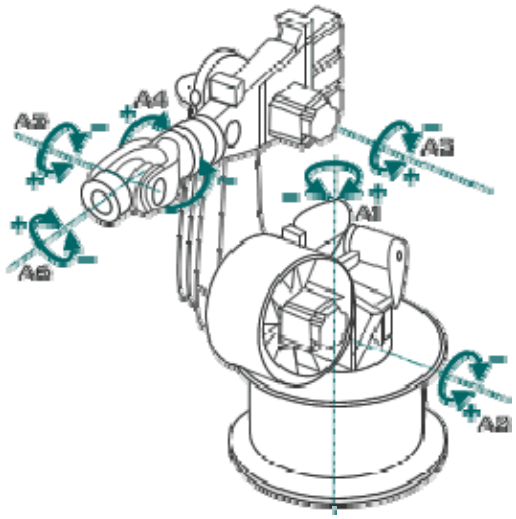
Tiene varias articulaciones, cada una de ellas puede realizar un movimiento distinto: rotacional, angular y lineal. Este robot utiliza la interpolación por articulación para moverse en sus dos primeras articulaciones y la interpolación lineal para la extensión y retracción. (Loop Technology Limited, 2007)



Gráfica 4 Robot con configuración Polar (Loop Technology Limited, 2007)

d) Configuración angular

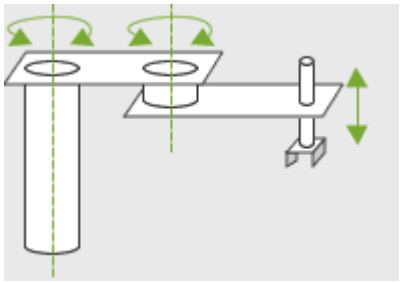
Presenta una articulación con movimiento rotacional y al menos dos movimientos angulares. Aunque el brazo articulado puede realizar el movimiento de interpolación lineal, para lo cual requiere mover simultáneamente dos o tres de sus articulaciones, el movimiento natural es el de interpolación por articulación, tanto rotacional como angular. (Loop Technology Limited, 2007)



Gráfica 5 Robot con configuración angular (Loop Technology Limited, 2007)

e) Robot SCARA

Además de las cuatro configuraciones clásicas mencionadas, existen otras configuraciones llamadas no clásicas. El ejemplo más común de una configuración no clásica lo representa el robot tipo SCARA, este brazo puede realizar movimientos horizontales de mayor alcance debido a sus dos articulaciones rotacionales. El robot de configuración SCARA también puede hacer un movimiento lineal mediante su tercera articulación. (Loop Technology Limited, 2007)



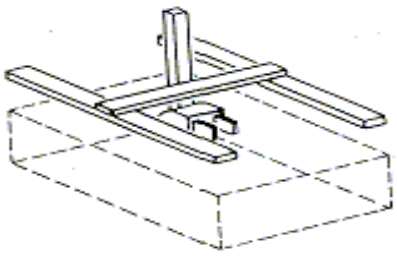
Gráfica 6 Robot SCARA (Loop Technology Limited, 2007)

5) Espacio de trabajo de los robots

Entre las características que identifican a un robot se encuentran su volumen de trabajo y ciertos parámetros como el control de resolución, la exactitud y la repetitividad.

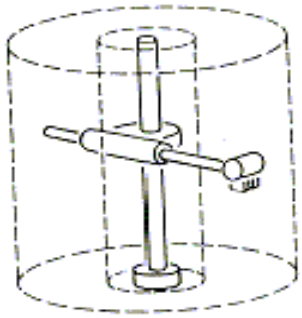
El volumen de trabajo de un robot se refiere únicamente al espacio dentro del cual puede desplazarse el extremo de su muñeca. Para determinar el volumen de trabajo no se toma en cuenta la herramienta final ya que es común que esta pueda ser intercambiada. (García, Álvarez, Cava; 2003)

El robot cartesiano y el robot cilíndrico presentan volúmenes de trabajo regulares. El robot cartesiano genera una figura cúbica.



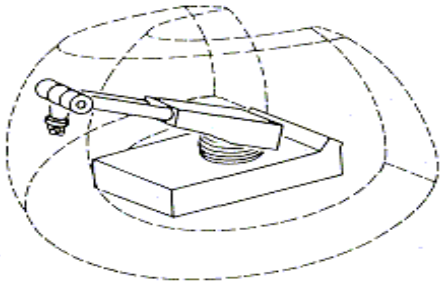
Gráfica 7 Espacio de trabajo configuración cartesiana (García, Álvarez, Cava; 2003)

El robot de configuración cilíndrica presenta un volumen de trabajo parecido a un cilindro (normalmente este robot no tiene una rotación de 360°)



Gráfica 8 Espacio de trabajo configuración cilíndrica (García, Álvarez, Cava; 2003)

Por su parte, los robots que poseen una configuración polar, los de brazo articulado y los modelos SCARA presentan un volumen de trabajo irregular.



Gráfica 9 Espacio de trabajo configuración polar (García, Álvarez, Cava; 2003)

6) Programación de los robots

La etapa de programación de secuencias de movimientos para un robot industrial usualmente se desarrolla conectando el controlador del robot a un ordenador portátil. El ordenador es conectado con el software correspondiente. Este proceso de enseñanza también puede realizarse por medio de un "*Teach Pendant*", el cual es un dispositivo portátil para la programación y control del robot. Cuando

el robot se termina de programar, se desconecta el “Teach Pendant”, el ordenador, o ambos y se procede a ejecutar el programa cargado al controlador.

Aparte de los dispositivos de programación, el operario usualmente tiene a su alcance dispositivos de interacción humana, los cuales permiten modificar programas o incluso comunicarse con otros dispositivos periféricos que ayuden al robot a realizar su trabajo. Estos periféricos incluyen interruptores de emergencia, sistemas de visión de maquinas, impresoras de códigos de barras y otros que permitan que el acceso del operador.

7) Calibración de los robots

El proceso de calibración de los robots industriales es muy importante ya que incrementa la exactitud del robot al ejecutar las tareas programadas. La calibración es el proceso de definir los parámetros cinéticos de la estructura del robot como las posiciones relativas de las uniones del mismo. Un robot calibrado posee una exactitud absoluta mayor a la de un robot no calibrado, es decir que, la posición final del robot es más cercana a la posición calculada por el modelo matemático del robot. Además de la calibración del robot, la calibración de las herramientas que utiliza, así como de los materiales con los que trabaja, minimiza la ocurrencia de errores y aumenta la seguridad del proceso.

La exactitud obtenida después de la calibración del robot puede variar dependiendo de la edad del robot y el ambiente de trabajo; puede oscilar entre décimas de milímetro a varios milímetros, pero en condiciones usuales se pueden obtener exactitudes de posicionamiento de cerca de 0.2 mm a 0.3 mm e incluso se puede llegar a tener exactitud de 0.1 mm.

La calibración es importante en varios sectores de la industria donde la mano de obra y las máquinas están siendo reemplazadas por robots y en muchos de estos

casos la exactitud al operar es muy importante. Aplicaciones como la maquila de componentes electrónicos, ensamblado de chasis de carros, remachado y fresado en industrias aeroespaciales y recientemente en aplicaciones médicas, requieren que la calibración sea muy exacta ya que un pequeño error puede tener un costo muy elevado e incluso poner en riesgo la vida de alguna persona.

8) Aplicaciones de los robots

Los robots poseen una gran gama de aplicaciones y a continuación se presentan algunas de las más comunes:

a) Soldadura

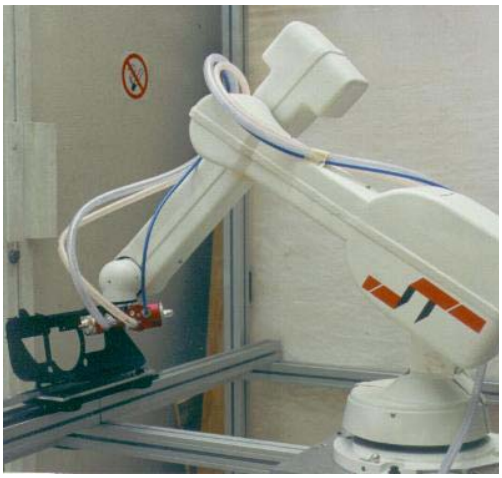
Es una de las aplicaciones más populares de la industria debido a su alta repetitividad, calidad uniforme y velocidad. Existen dos tipos principales de soldaduras: la soldadura de punto y soldadura en arco (Gráfica 10), aunque también se realiza la soldadura láser. Cerca de un 25% de las aplicaciones robóticas ejecutan la tarea de soldadura.

La industria automotriz utiliza en gran cantidad los robots de soldadura de punto. En 1985 las plantas de Chrysler Motor Corporation contaban con 900 robots, de los cuales 670 se utilizaban para soldadura de punto. Para el año 1990, esta compañía contaba con 2,350 robots. En la soldadura de arco dos partes adyacentes se unen por fusión, creando una unión hermética; los robots que ejecutan la soldadura de arco suelen tener la apariencia de un brazo articulado.

Gráfica 10 Robots para soldadura de arco (Berge, 2005)

b) Pintura con spray

La consistencia y repetitividad de los movimientos robóticos han permitido una calidad casi perfecta al mismo tiempo en que se reduce la cantidad de pintura utilizada. La pintura con spray personifica la aplicación adecuada de la robótica, evitando que el operador humano realice tareas muy precisas y peligrosas mientras que incrementa la calidad del trabajo, uniformidad y reduce costos. En las gráficas 11 y 12 se observan los robots en aplicación de pintura.



Gráfica 11 Robot de pintura con spray
(ST Robotics; 2006)

Gráfica 12 Robot de pintura con spray
(ST Robotics; 2006)

c) Operaciones de Ensamblaje

Los robots se prestan para realizar tareas de alta repetitividad; a partir de éste tipo de solicitudes se han desarrollado nuevas tecnologías para la maquila de componentes electrónicos, por ejemplo. Cerca de un 33% de las aplicaciones robóticas ejecutan tareas de ensamblaje.

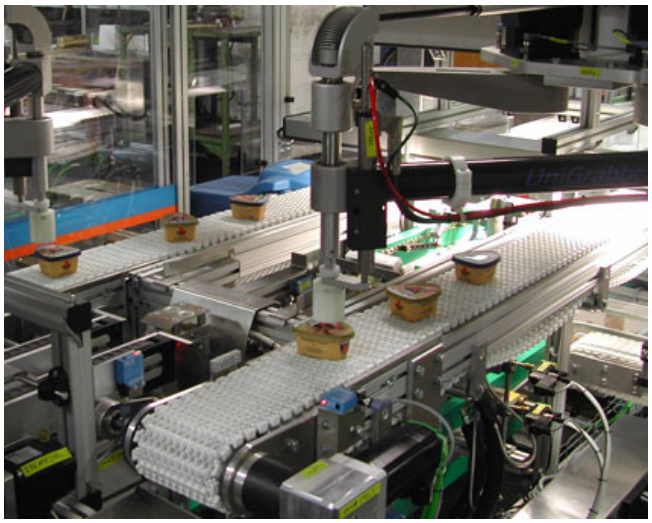
En la industria de semiconductores hay varios procesos que se llevan a cabo en un ambiente limpio. Esto requiere que tanto el personal como los robots no introduzcan polvo, suciedad o aceite en el área. Debido a que los robots no respiran o estornudan, son ideales para trabajar en ambientes limpios, como el que demanda la industria de semiconductores, como se puede observar en la Gráfica 13.

Gráfica 13 Robot para el ensamblaje de piezas electrónicas (Etneo, 2004)

d) Entarimado y manejo de materiales

Los robots para entarimar se encargan de ubicar productos, materiales y objetos sobre una tarima. Estos sistemas automáticos son de diversas naturalezas y aspectos. Algunos de estos robots se encuentran distribuidos por todo el recinto de almacenaje y tienen la forma de cintas transportadoras y numerosas carretillas que permiten situar las tarimas a diferentes alturas. Otros robots presentan la forma de un eje de dos dimensiones para situar las tarimas en repisas a una sola altura (RRG; 2007)

Por otro lado, algunas industrias, como la alimenticia, están empezando a utilizar los robots para el manejo de materiales ya que al evitar el contacto humano se obtiene proceso más ágil e higiénico (Gráfica 14). Actualmente esta es una aplicación poco solicitada, 2.8%, pero se espera que incremente cuando el manejo de los robots se simplifique.



Gráfica 14 Manejo de productos alimenticios (ELAU, 2006)

Gráfica 15 Entarimado de paquetes (Webster Griffing Ltd.; 2006)

E. El sistema de aprendizaje remoto en el ámbito académico

Conociendo un poco más de robots y de robótica en general, es posible proceder a explicar la posición del proyecto de aprendizaje remoto para el robot R17 en el ámbito académico. En muchas universidades se desarrollan proyectos que incluyen robots: en algunos casos, aunque sean pocos, son aplicados directamente en la industria; otras veces, es bastante común encontrar proyectos orientados a la investigación biomédica; y en otros casos, los proyectos tienen como fin el estudio de nuevas soluciones para la robótica en general. La gran mayoría de proyectos con fines industriales han sido desarrollados en los países responsables de la creación y producción de nuevas tecnologías, es decir que son países con una gran cantidad de recursos económicos destinados para la investigación de nuevas soluciones de robótica aplicada.

Después de una breve investigación, se determinó que en Latinoamérica la mayoría de las universidades que trabajan en el campo de la robótica no lo hacen en conjunto con la industria sino que el objetivo, muchas veces, consiste en desarrollar robots para competencias estudiantiles y por lo tanto resultan perfeccionando técnicas de movilidad y velocidad en pequeños robots exploradores. En algunos casos, cuando ya se tiene un robot, el objetivo se convierte en la mejoría de los procesos para hacer que el robot sea más autónomo. Por otro lado, para muchos trabajos de graduación, existen investigaciones que tienen como objetivo perfeccionar algún sistema de reconocimiento o interacción de robots, pero estos siguen siendo no industriales y se puede decir que las técnicas son para beneficio de la robótica general, únicamente.

Probablemente, hablando de la región latinoamericana, existan investigaciones privadas que se concentran en el desarrollo de la robótica industrial pero con el objetivo de proteger los resultados, no existen documentos publicados que permitan hacer referencias concretas.

La realidad de los países desarrollados tecnológicamente es distinta; en éstos, la robótica industrial ha tomado una gran relevancia en los últimos años porque ha sido la industria la que ha recurrido a las universidades para que éstas, como ente investigador, desarrollen nuevas soluciones. Esta tendencia también es nueva porque para el año 2000, la industria afirmaba que las universidades hacían la investigación que consideraban importante de acuerdo a sus intereses, mientras que en muchas ocasiones, esas investigaciones ya habían sido realizadas en laboratorios de empresas privadas y se había concluido que para el medio industrial no traían ningún beneficio porque el retorno económico no compensaba la inversión inicial. (Akeel y Holland; 2000)

Como parte de la investigación realizada, se ha encontrado que en universidades, principalmente europeas y estadounidenses, se están estudiando soluciones para la reducción de costos de nuevas tecnologías; esto sucede porque a pesar de que la industria de esos países posee grandes recursos económicos, el costo de algunas soluciones aun es elevado.

Por otro lado, se han impuesto nuevas tendencias para la tecnología y producción de robots industriales; como lo dicen Hadi A. Akeel (FANUC America Corporation) y Steve W. Holland (General Motors), la nueva tendencia es cumplir los requerimientos de aplicaciones especificas para lograr un retorno económico mayor y por lo anterior, los avances tecnológicos deben darse más en el control del robot que en su estructura mecánica, propiamente. (Akeel y Holland; 2000)

Algunas de las tendencias que se han tomado a nivel industrial son:

- La evolución de nuevas geometrías.
- La expansión de la utilización de sensores.
- La integración de sistemas para aplicaciones especificas.
- La utilización de sensores y actuadores inteligentes

Industrialmente, se considera que es más importante el desarrollo de nuevos sensores inteligentes y de bajo costo porque ya existe una amplia investigación en la reducción de tamaño físico de las estructuras, por ejemplo; y para que las nuevas tecnologías sean exitosas comercialmente primero deben reducirse los costos de producción. (Akeel y Holland; 2000)

La investigación de los proyectos realizados en otras universidades ha permitido observar que, en países desarrollados, los objetivos son la optimización de tareas porque ya fue superada la etapa de inserción de la robótica en la industria. Por lo mismo, las investigaciones son muy puntuales ya que se concentran en algún dispositivo del robot, que ya está implantado en alguna aplicación específica.

Por todo lo anterior, la comparación de este trabajo (sistema de aprendizaje remoto para el robot R17), con las realizadas en otras universidades es difícil porque, cuando las necesidades son distintas, los objetivos también lo son; y el mayor problema es que en países con necesidades similares a las de Guatemala, no existen tantas publicaciones.

Finalmente, es importante destacar que la robótica, en Guatemala, aun se encuentra en una etapa de desarrollo primario y por lo tanto antes de desarrollar técnicas especializadas de recopilación de datos y de control robótico, es necesario crear soluciones de bajo costo para que la industria las implemente y luego, podrá perfeccionarse la ejecución de esas tareas.

III. OBJETIVOS

A. Generales

1. Aplicar una gran parte de los conocimientos adquiridos a lo largo de la carrera en un proyecto multidisciplinario.

2. Impulsar la robótica industrial en Guatemala, permitiendo que cualquier persona pueda programar los movimientos requeridos para un robot y que consiga realizar tareas en tiempo real.

3. Desarrollar una aplicación para el robot R17 que pueda ser útil a la industria, programando tareas fuera del modo habitual para reducir el tiempo de programación y de colocación del robot en el proceso industrial.

B. Específicos

1) Silla Operativa

a) Estructura Mecánica

- Diseñar y construir interfaz humana que replique las partes y articulaciones del brazo robótico R17 para que el operario pueda identificar en la estructura el elemento a mover del brazo robótico.
- a. Diseñar y construir una silla que aloje al operario, los circuitos y la estructura de la interfaz humana.
- b. Diseñar, constuir y acoplar un brazo que el operario pueda maniobrar, identificando cada una de las partes del robot, y que aloje los sensores para detección de movimientos

- c. Proporcionar a los sensores el cableado de la potencia y de las líneas de datos.

b) Sensores

- Proveer un sistema que permita realizar lecturas de movimiento que sean transmitidas al eje denominado "cintura" del robot R17.
- Dar solución a la limitación, de la estructura mecánica, de un barrido de 360 grados sobre el eje paralelo a la superficie.
- Dar movilidad al sexto grado de libertad del Robot R17 desde la estructura mecánica.
- Seleccionar una familia de sensores capaz de obtener información de los movimientos dinámicos eficazmente y permita el uso de ésta en tiempo real.
- Diseñar y construir un sistema completamente automático que calcule la posición y orientación del eje de rotación de la silla operativa.
- Validar el uso de acelerómetros en aplicaciones de baja aceleración, tal como la detección de inclinación, y estudiar sus ventajas sobre otras familias de sensores.
- Instalar de la manera más precisa, estable y adecuada el sensor rotacional de la muñeca y la mano en la estructura mecánica para controlar el robot R17.
- Calibrar el sensor rotacional y ampliar el rango de la señal obtenida para mapear el dato capturado, considerando el error de fabrica del instrumento y el retraso en la señal de mando vrs. la respuesta del sistema.

c) Monitoreo

- Facilitar el uso del brazo robot al usuario, con una interfaz electrónica, que contenga información necesaria para el control del mismo.
- Permitir que un usuario pueda controlar el sistema desde la silla operativa mediante la interfaz electrónica.

2) Comunicaciones

a) Red Local

- Crear una red de comunicaciones robusta y adaptable, que satisfaga tanto las necesidades de comunicación entre nodos como las necesidades físicas del sistema de aprendizaje remoto para el robot R17.
- Transmitir detectar, validar y filtrar los mensajes recibidos para cada nodo de la silla operativa, respetando la jerarquía de prioridades para la utilización del bus de datos.

b) Red Wifi

- Transmitir sin errores todos los datos desde el módulo de silla operativa al módulo de brazo robótico.
- Transmitir los datos de forma inalámbrica bajo el estándar 802.11b/g
- Asegurar el envío correcto de todos los datos, así como detección y corrección de errores
- Recibir datos de forma inalámbrica por medio del estándar 802.11b/g.
- Asegurar el funcionamiento apropiado del enlace inalámbrico, así como la recepción correcta de la información.

3) Brazo robótico

- Crear una interfaz con el robot R17 que se opere de manera intuitiva y no requiera de conocimientos de programación.
- Establecer la comunicación entre la silla operativa y el robot R17.
- Permitir al usuario la creación de rutinas simples y repetitivas para su ejecución automática por medio del robot.
- Transformar los movimientos del usuario en información e instrucciones en el lenguaje del robot sin necesidad de programación o edición manual de código.
- Optimizar tanto las sentencias de programa como los movimientos realizados por el robot al realizar los movimientos deseados.

IV. MARCO TEÓRICO

A. Señales Eléctricas

Una señal eléctrica es un medio que sirve de soporte a la transmisión de alguna información. Las señales son representaciones de medidas de fenómenos físicos. Generalmente, las señales dependen de una variable, casi siempre describiendo el tiempo, o de varias variables, que frecuentemente tienen carácter espacial (Campus de Amaquique, 2001).

Todos los modelos matemáticos habitualmente utilizados para describir señales tienen sus correspondientes versiones continuas y discretas, calificativos que hacen referencia a la variable independiente del esquema funcional elegido. Específicamente, hablamos de señales analógicas, digitales, continuas o discretas (Campus de Amaquique, 2001).

En el proceso de cuantificación de una señal, los valores que puede tomar, están limitados a un conjunto discreto previamente seleccionado, normalmente relacionado con la precisión deseada; es decir con el número determinado de bits que representarán dicha señal. La señal digital o discreta, es el resultado del muestreo y consecuente cuantificación de una señal continua en un intervalo de tiempo específico.

En la mayoría de sistemas de comunicación se encuentran diferentes clases de señales. Las señales que registran el fenómeno son analógicas; después se convierten en digitales para facilitar su transmisión; finalmente se transforman de nuevo en analógicas para su recepción (Campus de Amaquique, 2001).

B. Digital versus Analógico

- **Ruido en señales digitales.** Cuando los datos son transmitidos usando métodos analógicos, una cierta cantidad de ruido entra a la señal. Esto puede tener diferentes causas: datos transmitidos por radio pueden tener una mala recepción, sufrir interferencias de otras fuentes de radio, etc. Pulsos eléctricos que son enviados por cableados pueden ser atenuados por la resistencia de los mismos, y dispersados por su capacitancia, y variaciones de temperatura pueden acrecentar o disminuir estos efectos. Cualquier variación puede proveer una gran cantidad de distorsión en una señal analógica (Lynn y Fuerst, 1999).
- En el caso de las señales digitales, aún las pequeñas variaciones en la señal pueden ser ignoradas de forma segura. En una señal digital, estas variaciones, se pueden sobreponer, pues, cualquier señal cercana a un valor particular será interpretada como ese valor (Lynn y Fuerst, 1999).
- **Display Analógico versus digital.** En la lectura de información, tanto los métodos digitales como los analógicos resultan gran utilidad. Si lo que se requiere es una impresión instantánea de resultados, los medidores analógicos usualmente ofrecen la información de una manera rápida, cuando lo que se requiere es exactitud los digitales son los preferidos (Lynn y Fuerst, 1999).
- **Perdida sistemática de datos.** Cuando se desea convertir una señal analógica a una digital, para ser procesada por otros sistemas digitales, algunos datos pueden perderse. El conversor analógico-digital sólo tiene cierta resolución, que depende de la cantidad de bits que se están utilizando para cuantificar una señal muestreada.

C. Procesadores digitales.

Un procesador digital de señales es un microprocesador especializado y diseñado específicamente para procesar señales digitales en tiempo real. Este provee secuencias de instrucciones de alta velocidad (Floyd, T.L., 2000).

La mayoría de los procesadores digitales son de punto fijo, debido a que comúnmente no se requiere de mucha precisión. Sin embargo existen los procesadores de punto flotante, que son comúnmente utilizados en campos científicos, donde existe la necesidad de alta precisión.

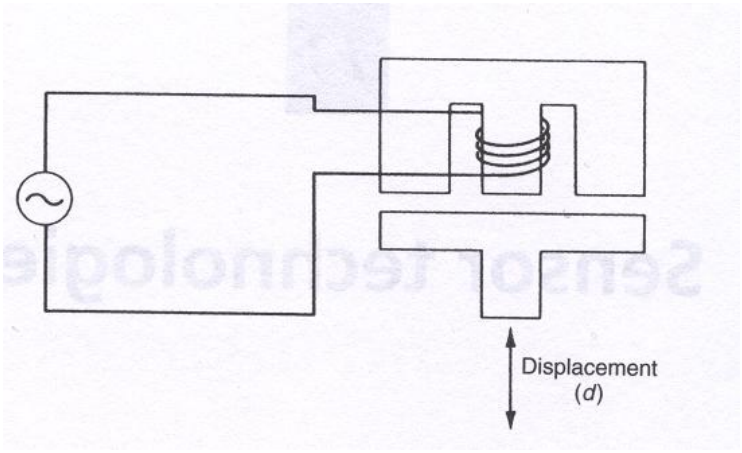
En su núcleo, un procesador es altamente numérico y repetitivo. A la vez que cada dato llega, éste debe ser manipulado según se requiera. Lo que permite realizar todo ello es la velocidad del dispositivo. Los sistemas basados en procesadores digitales deben trabajar en tiempo real, capturando y procesando información a la vez que ocurre. Los conversores deben adquirir la información lo suficientemente rápido como para captar todas las fluctuaciones relevantes de las señales. Si éste es muy lento, se perderá información. El procesador también debe trabajar rápido para no perder información que le llega desde el conversor, y además cumplir con el adecuado procesamiento de las señales (Floyd, T.L., 2000).

D. Sensores Magnéticos

Los sensores magnéticos utilizan los fenómenos de inductancia, resistencia y la corriente de Eddy para indicar el valor de la cantidad medida, que es usualmente una forma de desplazamiento.

Los sensores inductivos traducen el movimiento a un cambio en la inductancia mutua entre dos partes magnéticas. Un ejemplo de este tipo des sensor es el presentado en la figura 1. En este tipo de sensor, el embobinado en la parte central del cuerpo

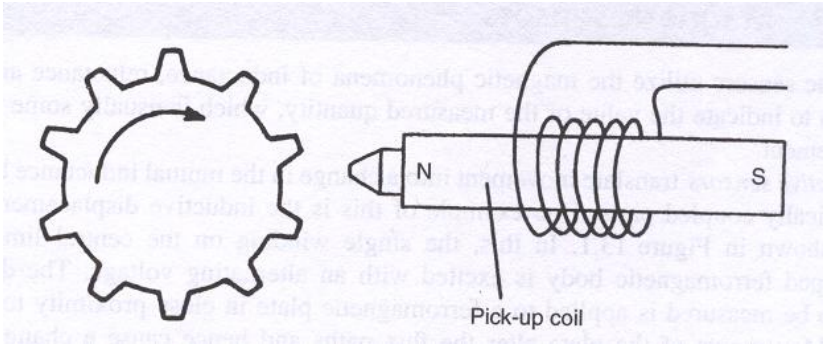
ferromagnético en forma de “E” es excitado con una corriente alterna. El desplazamiento a ser medido es aplicado a una placa ferromagnética que se encuentra pegada a la pieza en forma de “E”. Los movimientos en la placa alteran la trayectoria del flujo magnético, lo que causa un cambio en la corriente del embobinado. Por la ley de Ohm, la corriente que fluye en el embobinado esta dada por I=V/wL. Para valores fijos de w y V, la ecuación se convierte en I=1/KL, donde K es una constante. La relación entre L y el desplazamiento, d, aplicado a la placa de metal es no-linear, por lo que la relación “corriente-salida/desplazamiento” debe ser calibrada, antes de cualquier medición.



Gráfica 16 Sensor de Inductancia

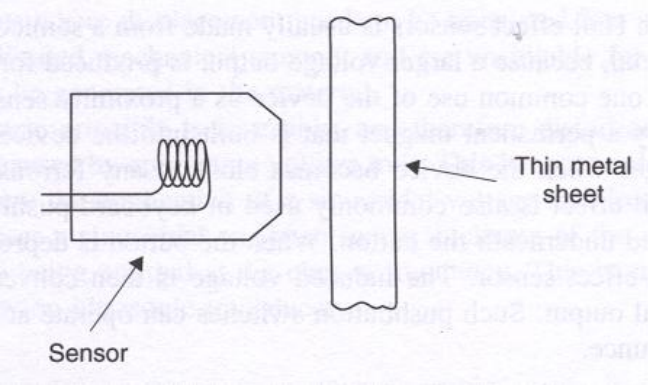
En los sensores de resistencia magnética variable, un núcleo esta sujetado a un magneto, a diferencia de los sensores de inductancia en los cuales una pieza de metal ferromagnético es la que está sujeta al magneto. Estos dispositivos son utilizados comúnmente para medir velocidades angulares. La figura 2 muestra un típico instrumento en el cual se coloca un engranaje ferromagnético cerca del sensor. A medida que cada diente del engranaje se acerca y se aleja de la punta de medición, el cambiante flujo magnético del núcleo causa que un voltaje sea inducido en el mismo núcleo, dicho voltaje es proporcional al rango de cambio del flujo. Por lo tanto, la

salida es una secuencia de pulsos negativos y positivos cuya frecuencia es proporcional a la velocidad angular del engranaje (Morris, 2001).



Gráfica 17 Sensor de resistencia magnética variable

Los sensores de corriente Eddy están compuestos por una punta con un núcleo, como el que se muestra en la figura 3, que es excitado a una alta frecuencia, típicamente a 1MHz. Este es utilizado para medir el desplazamiento de la punta de medición con relación a una pieza de metal. Debido a las altas frecuencias de excitación, las corrientes Eddy son inducidas únicamente en la superficie de la pieza, y la magnitud de la corriente se reduce a prácticamente cero cuando se esta muy cerca del objeto. Esto permite al sensor detectar objetos muy delgados. La corriente Eddy altera la inductancia del núcleo de la punta de medición, y este cambio puede ser traducido a un voltaje DC que será proporcional a la distancia entre la punta y el objeto. Este tipo de sensor puede alcanzar una resolución de hasta 0.1um. Puede trabajar con objetos no conductores si se les adhiere una envoltura o pieza de conductora, comúnmente aluminio.



Gráfica 18 Sensor de corriente Foucault

E. Protocolo I2C

El bus I2C está formado por dos líneas y una conexión a tierra. Las líneas activas, llamadas SDA y SCL, son bidireccionales. SDA es la "línea de data serial" y SCL es "línea de reloj serial", por sus siglas en inglés.

Cada dispositivo conectado al bus I2C tiene su propia dirección única, sin importar que tipo de dispositivos sea, un MCU, LCD, memoria, etc. Cada uno de estos dispositivos puede funcionar como transmisor y/o receptor, dependiendo de la tarea requerida.

Adicionalmente, cada dispositivo puede trabajar como maestro o como esclavo de la red en un determinado periodo de tiempo, según sea el requerimiento en ese momento.

Como primer paso, para transmitir algo a través del bus, el maestro envía una señal de "Start", la cual funciona como una señal de "Atención" para todos los dispositivos funcionando como esclavos. A partir de ese momento, todos los esclavos conectados al bus "escucharán" al mismo, esperando la información transmitida. Luego el maestro

envía la dirección del dispositivo al cual desea acceder, junto con una indicación del tipo de acceso, ya sea lectura o escritura. Una vez recibida la dirección, todos los esclavos la comparan con su propia dirección. Si la dirección no coincide con la propia, el dispositivo espera a que el bus sea liberado por la instrucción "stop". Si la dirección concuerda, el dispositivo produce una señal de respuesta conocida como "Acknowledge. Una vez enviada la señal de respuesta el esclavo está listo para empezar a transmitir o recibir información.

Cuando la transmisión con un esclavo termina, el maestro envía una señal de "stop", liberando así el bus para una nueva transmisión.

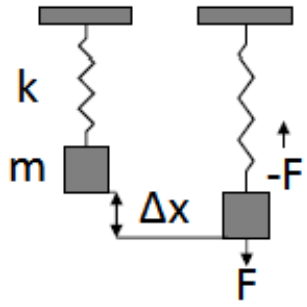
Las líneas SDA y SCL se conectan a la alimentación positiva por medio de un par de resistencias "pull-up". Cuando el bus está libre, ambas líneas están en nivel alto. La transmisión bidireccional serial consiste de ocho bits. Dicha transmisión de datos puede realizarse a 100Kbits/s en el modo estándar o 400 Kbits/s en el modo rápido.

Dentro de las ventajas del protocolo I2C se puede nombrar:

- Requiere solamente dos líneas, la de datos (SDA) y la de reloj (SCL).
- Cada dispositivo conectado al bus tiene un código de dirección seleccionable mediante software, habiendo permanentemente una relación maestro/ esclavo entre el microcontrolador y los dispositivos conectados.
- El bus permite la conexión de varios maestros, ya que incluye un detector de colisiones.
- El protocolo de transferencia de datos y direcciones posibilita diseñar sistemas completamente definidos por software.
- Los datos y direcciones se transmiten con palabras de 8 bits.

F. Acelerómetro

Se denomina acelerómetro a un instrumento destinado a medir aceleraciones. Generalmente, los acelerómetros consisten en una masa sísmica suspendida a un marco fijo por un resorte.



Gráfica 19 Modelo Acelerometro

La inercia de la masa suspendida se utiliza para detectar la aceleración. El principio de funcionamiento de un acelerómetro se basa en la segunda ley de Newton del movimiento, que se puede expresar de la siguiente forma, si se asume que una masa constante:

$$\vec{F} = m\vec{a}$$

La fuerza ejercida en la masa por la aceleración causa un desplazamiento, este desplazamiento corresponde al estiramiento del resorte:

$$\Delta x = \frac{-F}{k} = \frac{-ma}{k}$$

Así, el desplazamiento de la masa es una medida para la aceleración que actúa sobre ella. Además, el resorte presenta una tensión debido a su cambio en longitud.

Por lo tanto, la aceleración puede también ser determinada midiendo la tensión en el resorte. Ambos métodos se utilizan para detectar la aceleración que ocurre.

Al medir la magnitud de la aceleración, estática, provocada por la gravedad, se puede medir la inclinación del dispositivo con respecto a la superficie de la tierra. Y al medir la magnitud de aceleración dinámica, se puede analizar la manera en la que el dispositivo se ha movido.

Hay diversas maneras de hacer un acelerómetro, algunos acelerómetros utilizan el efecto piezoeléctrico, contienen estructuras cristalinas microscópicas que al ser tensionadas por las fuerzas de la aceleración generan un voltaje. Otra manera de hacerlo está detectando cambios en capacitancia. Si existen dos microestructuras, una al lado de la otra, tienen cierta capacitancia entre ellas. Si una fuerza aceleración mueve una de las estructuras, la capacitancia cambiará.

TIPOS COMUNES DE ACELERÓMETROS	
Tipo de Sensor	**Descripción**
Capacitivo	Unas vigas pequeñas o micromaquinadas de metal producen capacitancia entre ellas. Ésta capacitancia cambia con respecto de la aceleración.
Piezo-Eléctrico	Un dispositivo piezoeléctrico se monta en una masa. Éste al sufrir las fuerzas de aceleración, genera un voltaje proporcional a esta.
Piezo-Resistivos	Se compone de elementos micromaquinados, que al sufrir las fuerzas de la aceleración, cambian su resistencia.
Efecto Hall	Pueden sensar el movimiento debido al cambio de los campos magnéticos.
Magneto-Resistivo	Sus elementos estan hechos de materiales que cambian su resistencia en presencia de campos magnéticos
Transferencia de Calor	El movimiento de una masa a cierta temperatura se puede seguir midiendo el calor.

Tabla 1 Tipos de Acelerometro

1) Acelerómetro Capacitivo

El funcionamiento del acelerómetro capacitivo se fundamenta en una masa suspendida mediante un soporte elástico a un marco fijo, y sometida a los efectos de la aceleración que se desea medir. Midiendo los desplazamientos de esta masa, es posible determinar la aceleración a que está sometida. Esta medida puede hacerse directa o indirectamente. Este sistema tiene una frecuencia de resonancia no amortiguada ω_o dada por:

$$\omega_o = \sqrt{\frac{k}{m}}$$

Puesto que para conocer la aceleración se tiene un sistema dinámico, en el que el desplazamiento de la masa sigue a la fuerza (o aceleración) exterior sin retrasarse, se debe asegurar en el diseño del sensor que la frecuencia máxima esperada para la aceleración a medir sea mucho menor que la frecuencia natural del sistema ω_0.

El desplazamiento puede obtenerse a partir de la ley de Hooke:

$$X = \frac{F}{k}$$

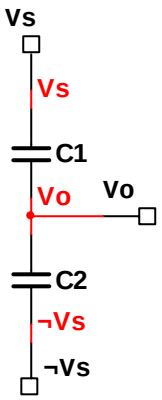
Si se sustituye:

$$F = ma \qquad k = \omega_0^2 m$$

$$X = \frac{a}{\omega_0^2}$$

A partir de ésta ecuación se deduce que un acelerómetro capaz de responder rápidamente a cambios en la aceleración (ω_o grande), tiene una amplitud de señal de salida pequeña.

La posición del acelerómetro se obtiene midiendo la variación de la capacidad entre ella y un electrodo fijo. La configuración más utilizada es la de capacitancia diferencial, en la que existen dos electrodos fijos y uno móvil. La capacitancia entre los electrodos fijos es siempre constante, mientras que la capacidad existente entre el electrodo móvil y cualquiera de los fijos varía con la posición.



Gráfica 20 Modelo acelerometro capacitivo

Supongamos que la separación entre placas de C1 es G1 y la de C2 es G2, y que se aplica una tensión V_S al electrodo superior de la figura y al inferior $-V_S$. Entonces la tensión a la salida Vo es:

$$I_{C1} = I_{C2}$$
$$(V_s - V_o)\, j\omega C_1 = (V_s + V_o)\, j\omega C_2$$
$$V_s C_1 - V_o C_1 = V_s C_2 + V_o C_2$$
$$V_o (C_1 + C_2) = V_s (C_1 - C_2)$$
$$V_o = \frac{C_1 - C_2}{C_1 + C_2} V_s$$

Si el área de los capacitores es igual para ambos entonces la ecuación anterior se transforma a:

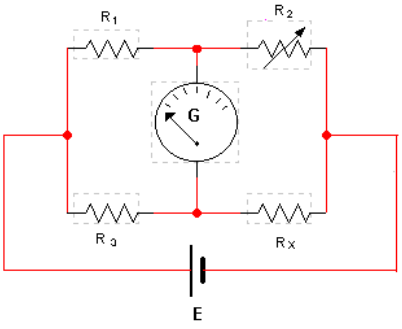
$$V_o = \frac{G_1 - G_2}{G_1 + G_2} V_s$$

Con esta ecuación se pude concluir que si la distancia es la misma para los dos capacitores entonces el voltaje de salida será nulo. En otro caso el voltaje de salida variará linealmente con respecto de la distancia de separación.

Una de las mayores desventajas del principio de detección capacitivo son su alta impedancia de la salida y la presencia posible de los resistores de la salida. Las ventajas principales del principio son su bajo consumo de energía, alta sensibilidad, dependencia baja de la temperatura y alta estabilidad.

2) Acelerómetro Piezoeléctrico

La mayoría de los sensores de estado sólido para las señales mecánicas se basan en el efecto piezo-resistivo que es muy alto en silicio. El cambio en la conductividad específica debido a una tensión aplicada se llama piezo-resistencia. Básicamente, los piezoresistores se incluyen en una configuración del puente del Wheatstone en la posición de los valores extremos de la tensión.

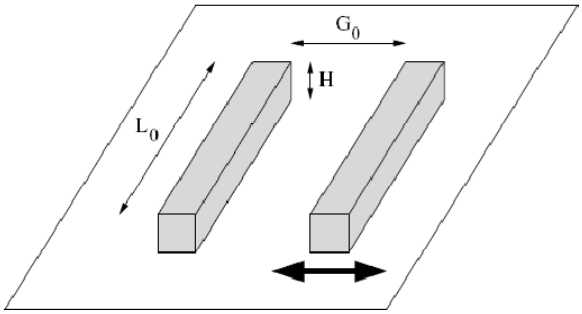


Gráfica 21 Puente Wheatstone

Puesto que el puente de Wheatstone se equilibra en la situación cero de la tensión, el voltaje de la salida debido al desbalance del puente es una medida directa de la tensión actual en el substrato del silicio. Esta tensión se liga directamente, vía la ley de Hooke, al desplazamiento de la masa suspendida y por lo tanto a la aceleración aplicada. Dos ventajas del principio de los piezo-resistores son que una respuesta real de la Corriente Directa puede ser medida y no se necesita circuitería adicional necesaria para la detección del cambio del voltaje. Las desventajas principales de este principio es la dependencia fuerte de la temperatura de los coeficientes piezo-resistivos, la deriva y consumo de alta energía (una corriente directa permanente se requiere para realizar medidas continuas). La alineación exacta de los piezo-resistores respecto al marco puede también causar problemas. Sin embargo, el principio piezo-resistivo se utiliza en la mayoría de los acelerómetros comercialmente disponibles.

3) Acelerómetro Micro mecanizado

El acelerómetro micro mecanizado se puede modelar de la siguiente manera:



Gráfica 22 Modelo del acelerometro micromaquinado

Si se supone que éste está formado por placas planas, la capacitancia entre ellas está dada por la siguiente fórmula:

$$C = \frac{\varepsilon_0 H L_0}{G_0 \pm y}$$

donde H es el espesor de la capa del material, L_0 es la longitud enfrentada de los electrodos, G_0 es la separación de los electrodos inicial y x es la variación de esa separación.

En el caso de una viga de sección rectangular con una fuerza aplicada en su centro, la constante elástica es:

$$k = \frac{F}{y} = \left(\frac{\pi^4}{6}\right)\left(\frac{EWH^3}{L_0^3}\right)$$

donde F es la fuerza aplicada, *y* el desplazamiento del centro de la viga, E el módulo de Young del material de la Viga, W el ancho de la Viga y H el espesor.

4) Acelerómetro de frecuencia Resonante

El principio del la obtención de la aceleración por medio de la frecuencia resonante se basa en el hecho de que la frecuencia resonante de un micro puente cambia cuando está sometido a la tensión extensible o compresiva. Colocando estos puentes resonantes en suspensión, la frecuencia cambia con el desplazamiento de la masa y por consecuencia con la aceleración. La frecuencia resonante se puede determinar de esta forma piezo-resistivamente, capacitivamente, u ópticamente. El

coeficiente de la temperatura del dispositivo de la lectura (piezo-resistores, capacitores o diodos) no es importante, porque los cambios de temperatura afectan simplemente la amplitud de la señal de salida y no de la frecuencia resonante del acelerómetro. El comportamiento termal restante del dispositivo es determinado por las características mecánicas y materiales del sensor, tales como módulo de Young y tensión termal inducida en el paquete. Así, la estabilidad termal sigue siendo uno de los problemas. Algunas de las desventajas d son la necesidad de circuitería relativamente compleja, incluyendo un oscilador.

5) Acelerómetro Inductivo

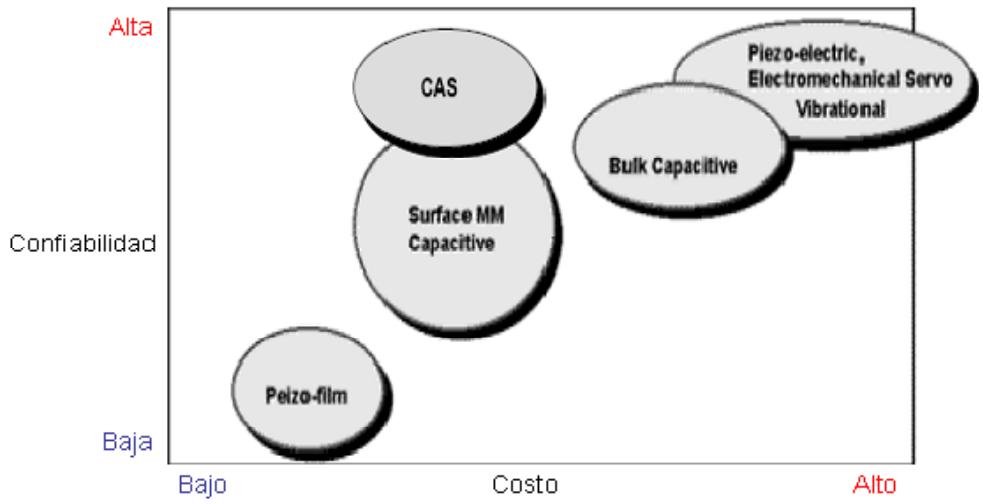
El sensor inductivo de la aceleración consiste en dos bobinas planares, una en la masa móvil y otra fija. Una de las bobinas se utiliza para generar un campo magnético que se alterna. Consecuentemente, un voltaje inducido se genera en la otra bobina, con una amplitud proporcional a la distancia entre las dos bobinas. De esta manera, el desplazamiento total y la aceleración pueden ser determinadas. Una desventaja de este método es que la fabricación de bobinas mMicro maquinadas es un procedimiento muy complicado, por lo tanto lleva a un alto costo de los sensores.

6) Acelerómetros Ópticos

Los acelerómetros ópticos no son muy comunes, pero los que se han construido últimamente se basan en cambios en intensidad de luz por la masa móvil que actúa como el obturador de una cámara, o por un cambio en longitud de onda reflejada usando una rejilla de Bragg en una guía de onda planar. El principio óptico es muy exacto, pero requiere un difícil calibración de los dispositivos ópticos junto con la estructura de detección mecánica.

7) Acelerómetros Térmicos

En el principio termal, la posición de la masa afecta la cantidad de flujo del calor debido a la conducción de éste a través del gas entre la masa y la placa fija. Como resultado del flujo del calor variable, una diferencia de temperatura entre la fuente y el disipador de calor, depende de la posición de la masa y así de la aceleración. La masa puede actuar como una fuente de calor o disipador de calor, y al mismo tiempo la encapsulación sirve como el disipador de calor o la fuente de calor, respectivamente. Se mide la diferencia de la temperatura usando termopilas. La desventaja principal de este método es su consumo de alta energía debido a la disipación constante de la energía en la fuente de calor.

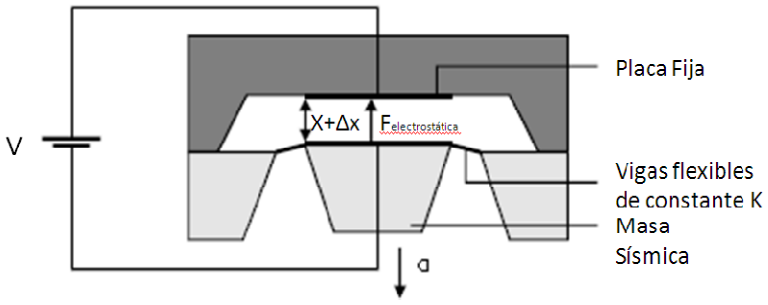


Gráfica No. 1.Costo Vs. Confiabilidad Acelerómetros

G. Funcionamiento del Acelerómetro Tipo Capacitivo

1) Generación de Fuerza Electrostática

Para generar una fuerza electrostática, se aplica un voltaje a través de la placa fija del capacitor y en la placa de la masa sísmica movible, causando una fuerza electrostática que actúa en la masa. Debido a la naturaleza atractiva de esta fuerza, la masa se desvía hacia la placa fija. Cuando dos placas se colocan en lados opuestos de la masa sísmica, la masa puede moverse en cualquier dirección. Una desventaja en la generación de fuerza electrostática es el voltaje relativamente grande requerido para obtener una fuerza razonable, pero este puede ser reducido drásticamente acortando la distancia entre las placas del capacitor.



Gráfica No. 2.Fuerzas dentro de un acelerómetro

En una situación estática, la suma de todas las fuerzas que actúan sobre la masa sísmica es cero.

$$0 = ma - F_{electrostatica} - k\Delta x$$

$$F_{electrostatica} = \frac{\varepsilon A V^2}{2(X + \Delta x)}$$

donde V es el voltaje aplicado entre las placas del capacitor, A es el área de las placas y ε es la constante dieléctrica del medio entre la masa sísmica y la placa contraria del capacitor.

Cuando no se aplica ningún voltaje a través de las placas del condensador, la fuerza electrostática es cero y el desplazamiento de la masa sísmica ΔX, es proporcional a la aceleración aplicada. Si un voltaje V se aplica a través de las placas del condensador, una fuerza electrostática ocurre y la relación entre ΔX y la aceleración aplicada llega a ser no linear. Para los desplazamientos pequeños de la masa sísmica, de $\frac{\Delta x}{X} < 5\%$; se puede llegar a despreciar la aceleración. En este caso, al aplicar un voltaje a través de las placas del capacitor corresponde a sujetar la masa a una aceleración determinada por el voltaje, en esto se basa el principio de autoprueba (Self-Test) de los acelerómetros.

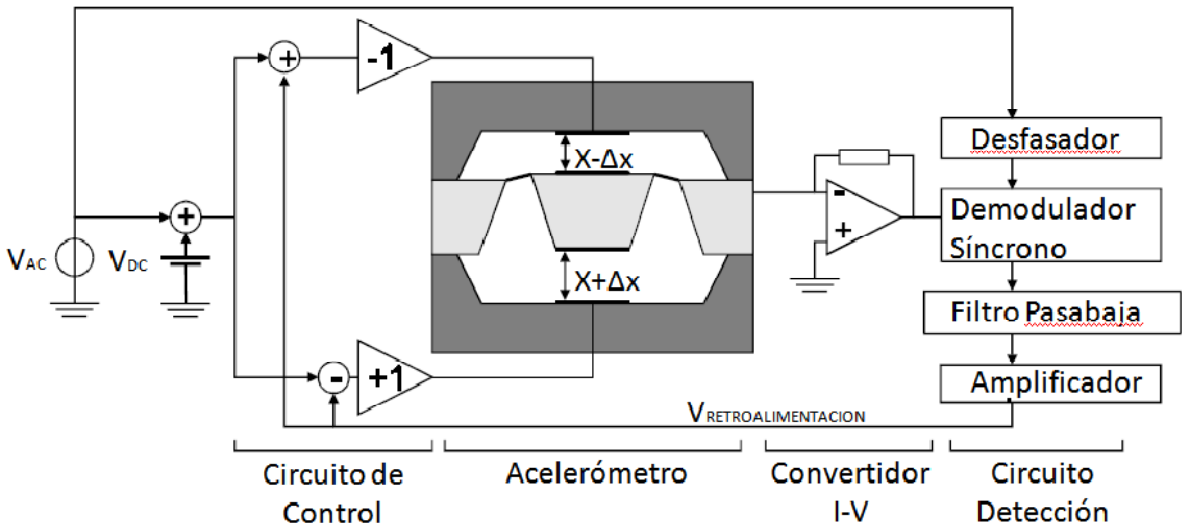
2) Balanceo de Fuerzas dentro un acelerómetro

El método para balancear fuerzas dentro de un acelerómetro consiste en aplicar una fuerza electrostática a la masa sísmica para contrarrestar la fuerza causada por la aceleración y así lograr que el sistema vuelva a su posición inicial. El valor del voltaje de balanciamiento debe ser proporcional a la aceleración que ocurre en la masa.

La ventaja más grande de éste método es que linealiza la respuesta de los sensores esencialmente los no lineares. Además, el rango de respuesta dinámica puede ser ampliado. La desventaja principal es el consumo de energía depende del estado actual del acelerómetro y de su modo para sensar.

El principio de funcionamiento del balanceamiento de fuerzas es el siguiente. Se aplican dos voltajes a las placas fijas del capacitor: un voltaje AC para sensar la capacitancia entre la masa sísmica y las placas fijas del acelerómetro, y un voltaje DC para controlar la posición de la masa. El voltaje AC se encuentra desfasado 90º para poder extraer la información de la aceleración actual desde la señal de salida, mediante demodulación síncrona.

El voltaje D.C. consiste en un voltaje inicial y un voltaje de retroalimentación El voltaje inicial se utiliza para definir el punto de trabajo del dispositivo y linealiza su característica de salida.



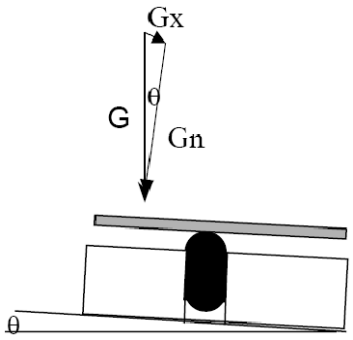
Gráfica No. 3.Modelo Acelerómetro Balanceo de Fuerzas

El voltaje AC se aplica a las dos placas simétricas fijas del capacitor. Puesto que la masa sísmica es desplazada por una aceleración, las capacitancias entre la masa

sísmica y las placas fijas del capacitor cambian. La corriente de salida, debido a la diferencia en las capacitancias pasa a través de la masa sísmica, y es convertida en un voltaje por un convertidor I-V. El voltaje de salida del convertidor se aplica al demodulador síncrono que funciona como un amplificador *lock-in*.

La señal desfasada se aplica al demodulador síncrono como señal de referencia. Como resultado el demodulador síncrono da un voltaje proporcional a la diferencia entre las capacitancias. Este voltaje entonces se amplifica para obtener el voltaje necesario de retroalimentación para mantener la masa sísmica en su posición inicial. El valor del voltaje de retroalimentación es, por lo tanto, una medida de la aceleración que aplicada.

3) Detección de Inclinaciones

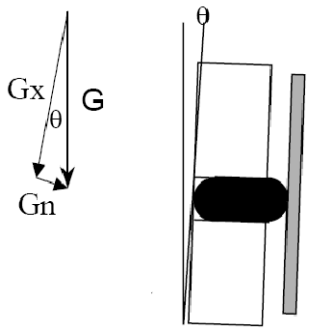


Gráfica No. 4.Modelo Fisico Eje Y Acelerómetro

La única aceleración que actúa sobre uno de los ejes del acelerómetro es la gravedad.La única fuerza efectiva en el sensor es G_n. Y esta puede escribirse de la siguiente manera.

$$G_n = G \cdot \cos(\theta)$$

Cómo se demostró anteriormente, la salida del acelerómetro varía linealmente conforme varía la fuerza G. Por lo que la salida de éste también variará linealmente con respecto de la posición.



Gráfica No. 5.Modelo Fisico Eje Y Acelerómetro

De igual forma se demuestra que para el otro eje del acelerómetro la salida variará linealmente ya que:

$$G_n = G \cdot \sin(\theta)$$

H. Diseño de Acelerómetros Tipo Capacitivo

Hay diferentes tipos de acelerometros disponibles de estado sólido en silicio, tal como micromaquinados y la técnica de LIGA.

1) Micromaquinados

Las masas y las suspensiones del resorte son grabadas a través de la placa completa. Por lo tanto, las masas disponibles para la conversión de la aceleración a fuerza son relativamente grandes. Esto permite la fabricación de dispositivos muy sensibles. Un sistema hecho de esta manera necesita ser amortiguado debido a su alto factor de la calidad. Por lo tanto, para evitar un comportamiento oscilatorio de la masa, el dispositivo se encapsula. El gas (o a veces líquido) entre la masa y la encapsulación proporciona un amortiguador.

Por otra parte, protege la masa contra colisiones, proporcionando una protección del extra. Debido al gran tamaño de la masa, relativamente, las señales de salida son grandes, de tal modo que este tipo de acelerómetro permite el uso de interacción con otros circuitos sin necesidad de un circuito de acoplamiento.

2) Micromaquinado Superficial

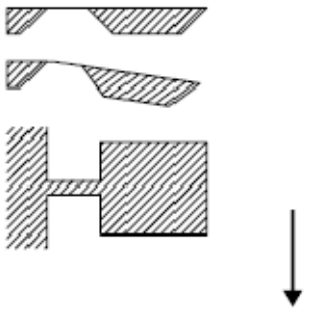
Las masas y la suspensión del resorte son formadas por medio de un grabado selectivo de capas, que tienen un grueso, típicamente, de varios micrones. Consecuentemente, las masas son pequeñas. Esta técnica requiere técnicas de proceso más robustas, pues la señal de salida del sensor es pequeña y los efectos del ruido sobre éste son relativamente grandes. La ventaja principal de esta tecnica es el bajo costo de fabricación.

I. Configuración Acelerómetros

1) Micromaquinado Masa-Suspensión

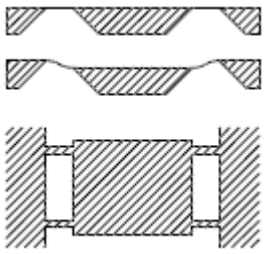
La suspensión tipo *cantiléver* masa-viga fue la primera en ser utilizada para la construcción del acelerómetro micromaquinado de silicio. La masa sísmica se suspende a partir de uno de sus lados con una o más vigas. Esta se mueve en la

dirección señalada cuando se le aplica una aceleración. Debido a que la masa se suspende simplemente por uno de sus lados, esta configuración ofrece alta sensibilidad en la dirección para la que está diseñado, pero desafortunadamente también tiene una alta sensibilidad a aceleraciones laterales en dirección diferente al eje.



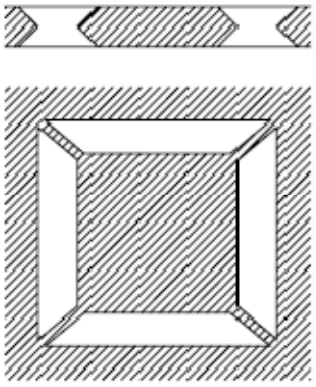
Gráfica No. 6.Configuración Tipo Cantiléver

En la configuración masa-viga tipo puente la masa sísmica se suspende por medio de vigas en por lo menos dos de sus lados opuestos. Solamente dispositivos con propiedades piezoresistivas se han desarrollado para este tipo de configuración. Si se acoplan los piezoresistores en un puente de Wheatstone, se puede llegar a obtener una compensación a la sensibilidad para aceleraciones laterales.



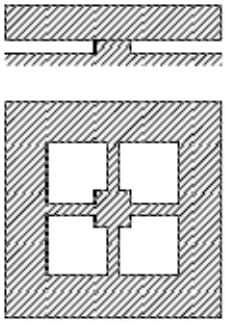
Gráfica No. 7.Configuración Tipo Puente

En la configuración masa- viga simétrica, se suspende simétricamente la masa por sus lados superiores e inferiores. Debido al alto número de vigas, la desviación de éstas es relativamente pequeña (al igual que las tensiones mecánicas), por lo que la detección piezoresistiva no es efectiva. Para este tipo de configuración se utiliza detección por medio de capacitores. Un de las mayores ventajas es que tipo de configuración muestra una sensibilidad despreciable a movimientos fuera de su eje de medición.



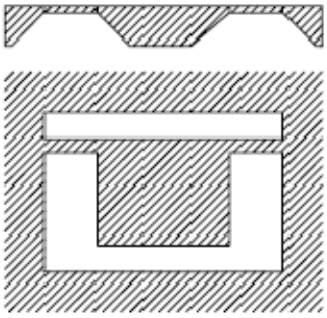
Gráfica No. 8.Configuración Simétrica

El tipo masa circundante se compone de una masa sísmica, la cuál se suspende con las vigas a un marco en el interior de la masa. En comparación a los otros tipos de configuración, la masa es mucho más grande. Por lo tanto, este tipo ofrece una sensibilidad muy alta.

C. 

Gráfica No. 9.Configuración Masa Circundante

En el caso de la configuración tipo barras de torsión, la aceleración da lugar a una rotación de la masa alrededor de las vigas suspendidas. Al igual que el tipo Cantiléver, este tipo de configuración ofrece una alta sensibilidad. Para este tipo de configuración sólo se han desarrollado sensores del tipo capacitivo.

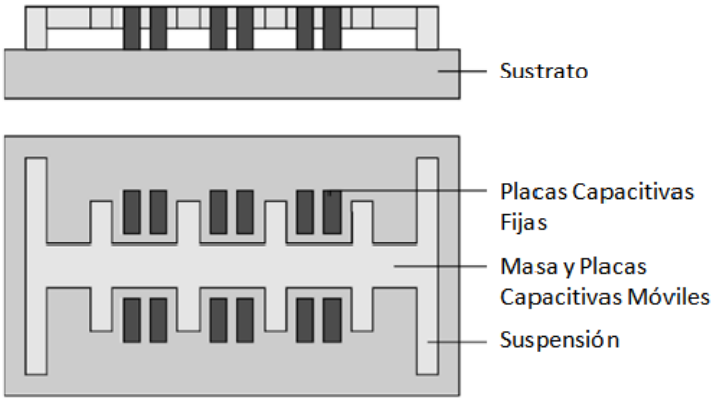
D. 

Gráfica No. 10.Configuración barras de Torsión

2) Micromaquinado Superficial Masa-Suspensión

En la configuración tipo peine, la masa sísmica se asemeja a una letra "H". Los brazos de la masa, anclan la viga de detección al substrato. Así, la masa sísmica

está libre moverse en un plano perpendicular al de sus brazos. Las articulaciones se originan de la masa central, cada una actúa como una placa de un condensador de placas paralelas variables. Este sensor sólo puede utilizar sensores de la familia capacitiva.



Gráfica No. 11. Configuración Peine

Para sensores micromaquinados también existen configuraciones tipo Cantiléver, Puente y Barras de torsión.

J. Terminología

+1g: La salida del sensor con el conector base orientado hacia arriba.

0g: La salida del sensor con el conector base horizontal.

-1g: La salida del sensor con el conector base orientado hacia abajo.

Descripción	Nivel g
Gravedad Terrestre	1g
Pasajero de un carro en una curva	2g
Túmulos	2g
Carro Indy en una curva	3g
Limite de Conciencia Humano	7g
Despegue Cohete	10g

Tabla No. 1. Aceleraciones de Referencia

K. Características que definen a un Acelerómetro

Un acelerómetro se caracteriza por las siguientes especificaciones:

- ***Sensibilidad:*** es el cambio que se presenta en la salida del sensor con respecto al cambio en la aceleración de la entrada. Este se mide en

$$\frac{Volts}{g}.$$

Usualmente la fórmula para calcularla es:

$$Sensibilidad = \frac{\Delta V}{\Delta g} = \frac{V_{out,+1g} - V_{out,-1g}}{2g}$$

- ***Rango:*** se define por la máxima aceleración que se puede medir. Esta es limitada por margen de movimiento de los electrodos.

- **Ancho de Banda:** el acelerómetro está limitado por la condición enunciada anteriormente en la cual la frecuencia máxima de la aceleración debe ser mucho menor que ω_0. Usualmente se define cómo:

$$10 \cdot f_{\max} \leq \frac{\omega_0}{2\pi}$$

- **Linealidad:** es la desviación máxima del sensor con respecto de su curva de calibración.

$$Linearidad = V_{out,0g} - \frac{1}{2}\left(V_{out,+1g} + V_{out,-1g}\right)$$

- **%Vcc:** las lecturas se representan a menudo como un porcentaje del voltaje fuente. Esto permite la corrección debido a las variaciones del voltaje fuente entre las lecturas.

L. Uso de los Acelerómetros Capacitivos en aplicaciones de baja Aceleración

Los acelerómetros se pueden utilizar en una amplia variedad de aplicaciones de baja aceleración, tales como inclinación y orientación, análisis de vibración, detección de movimiento, etc. Es extremadamente importante al momento de medir aceleraciones bajas el nivel de introducción de datos perceptibles, es decir, el rango del acelerómetro a gama completa de aceleración.

La resolución tiene como limitante el ruido *"floor"* de la medida. Este incluye el ruido ambiente y el ruido del mismo acelerómetro.

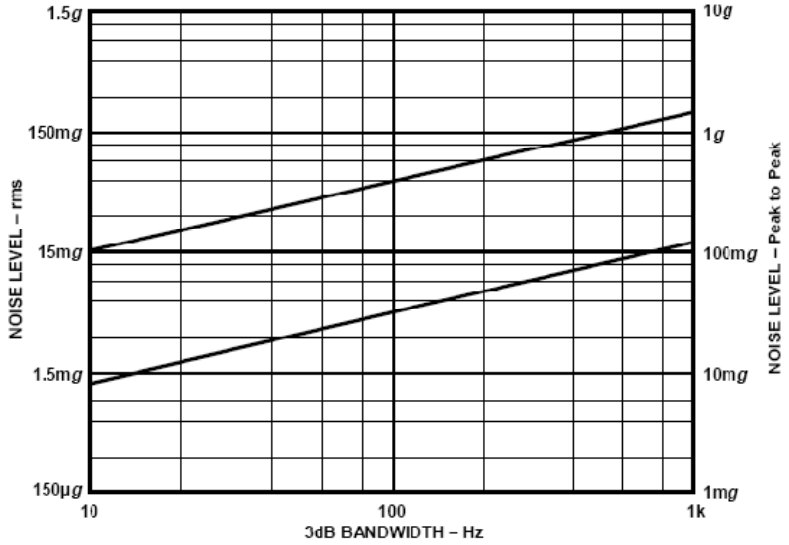
El nivel del ruido *"floor"* varía directamente con el ancho de banda de la medida. Si se reduce el ancho de banda de la medida, el ruido *"floor"* se reducirá, mejorando el cociente señal-ruido de la medida.

El ruido de salida de los acelerómetros varía linealmente con la raíz cuadrada del ancho de banda de la medida. La amplitud máxima del ruido, valor pico a pico, define aproximadamente la resolución en el peor de los casos. El ruido pico a pico es aproximadamente igual a 6.6 veces su valor del rms (lo que define una incertidumbre media de 0.1%).

$$N \propto \sqrt{BW}$$

$$N_{p-p} \cong 6.6 V_{Orms}$$

El ancho de banda del acelerómetro puede ser reducido fácilmente agregando un filtro pasa baja o pasa banda.



Gráfica No. 12. Ancho de Banda Vs. Ruido

Como se muestra en la Gráfica No.16, el ruido del dispositivo se minimiza si se reduce el ancho de banda de funcionamiento. Por ejemplo, cuando está funcionado en un ancho de banda de 1 kHz, se tiene típicamente un nivel de ruido pico a pico de 130mg. Con aceleraciones de ±18 aplicadas al acelerómetro, este límite de la

resolución es muy satisfactorio; pero para una aceleración más pequeña, el ruido es un porcentaje mucho mayor al de la señal. Como se muestra en la Gráfica No.16, cuando el ancho de banda del dispositivo disminuye a 100 Hz, el nivel de ruido de pico a pico es reducido aproximadamente a 40mg, y en 10 Hz está por debajo de los 10mg.

Alternativamente, el cociente señal-ruido puede ser mejorado considerablemente usando un microprocesador para realizar medidas múltiples y después computar el nivel medio de la señal. Al usar esta técnica, el nivel de la señal aumenta directamente con el número de medidas mientras que el ruido aumentará solamente en su raíz cuadrada. Por ejemplo, con 100 medidas, el cociente de señal-ruido será aumentado en un factor de 10 (20dB).

M. Técnicas para reducir el ruido en la medida

Además de reducir ruido del circuito, cualquier interferencia electromágnetica necesita ser considerada.

El cable blindado debe ser utilizado para conectar el acelerómetro con cualquier equipo o circuito que se encuentre relativamente lejos.

Un problema muy común es el de la inducción de 60 Hz de la línea voltaje alterna. Esto puede ser reducido físicamente moviendo el dispositivo lejos de las líneas de alimentación, o utilizando técnicas de blindaje y puestas a tierra apropiadas. En la mayoría de los casos, es recomendable solamente conectar a tierra el protector de cable en un extremo y conectar referencias separadas entre los circuitos; esto ayudará a prevenir los lazos de tierra. También, si el acelerómetro se encuentra en el interior o cerca de un recinto del metal, este se debe poner a tierra.

N. Codificador Absoluto

Los codificadores de posición absoluta ofrecen en su salida una señal codificada correspondiente a la posición de un elemento móvil, regla o disco, con respecto a una referencia interna. Para ello, el elemento móvil dispone de zonas con una propiedad que las distingue, y a las que se asigna un valor binario ¨0¨ ó ¨1¨. Cuenta con varias pistas con zonas diferenciadas y están agrupadas de tal forma que el sistema de lectura obtiene directamente, en cada pocisión del elemento móvil, el número codificado que da su posición .

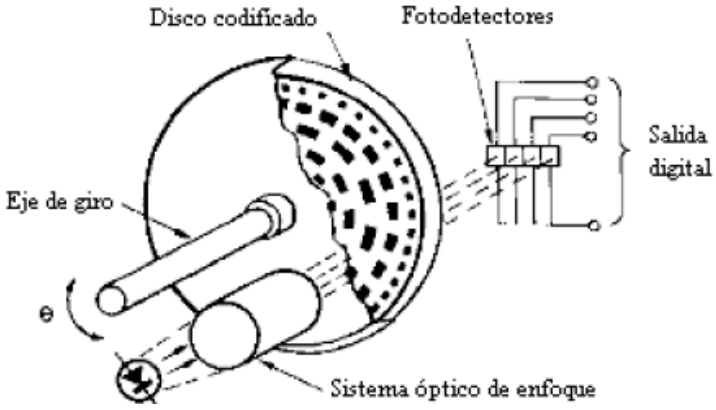


Gráfico 3 Principio de funcionamiento de un *codificador*

Cada pista representa un bit de salida, siendo la pista más interior la correspondiente al bit de mayor peso. Los tipos de sensores más empleados en este caso son los ópticos, con zonas opacas y transparentes. Hay conjuntos de foto-sensores integrados que facilitan la realización del codificador.

Estos codificadores tienen inmunidad intrínseca frente a las interrupciones e interferencias electromagnéticas, con la ayuda de cabezales de lectura más complejos. Se debe a que hay tantos elementos de lectura como pistas y a la necesidad de que

todos ellos estén bien alineados, de forma contraria, el código ofrecido a la salida puede estar formado por bits correspondientes a dos posiciones contiguas (en particular cuando se produzca la transición de una a otra). El código continuo más empleado en codificaciones es el Gray.

Su inconveniente es que si la información hay que mandarla a un ordenador, conviene convertir la salida a salida de código BCD. Estas conversiones se obvian en el caso de tener el disco codificado directamente en código de utilización final. Si la información se va transmitir a un ambiente ruidoso, el código de Gray no permite la detección de errores de transmisión. Otro método, consiste en disponer un doble juego de cabezales de lectura desplazados entre sí una distancia determinada, empleando luego una regla de decisión para aceptar la lectura de uno u otro sensor para cada pista.

También se puede disponer una marca en el centro de cada sector, aceptando entonces la lectura del cabezal sólo cuando hay garantía de estar en una zona que no es de transición entre las posiciones. Una memoria almacena la última lectura obtenida y se actualiza cuando hay un cambio válido.

La resolución que se obtiene con estos codificadores es de 6 a 21 bits en código Gary, siendo los rango de 8 a 12 bits los habituales. Para aumentar la resolución, la opción inmediata es aumentar el número de pistas codificadas, pero el inevitable aumento de diámetro e inercia limita esta solución. El empleo de un engranaje y otro codificador es una alternativa, aunque la resolución final siempre queda limitada por la obtenible por el primer disco. Para aumentar la resolución también se puede se añadir una pista adicional, dispuesta en dirección radial, en la parte más externa del disco.

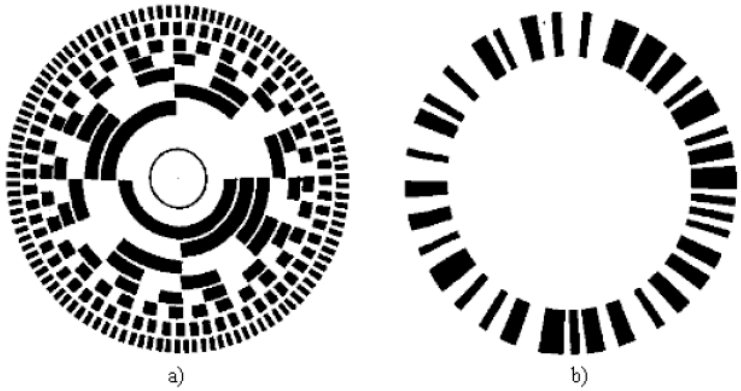


Figura 5.10 a)Discos de un codificador digital que añade una pista externa para aumentar la resolución mediante un sistema de rejillas fijas. b)Codificador absoluto basado en un código pseudoaleatorio.

Gráfico 4 Técnica para el aumento de escala

La aplicación de los codificadores de posición es relativa a la medida y control de posición lineal y angular con alta resolución. Se emplea así en robótica, grúas, válvulas hidráulicas, mesas de dibujos automáticas (plotters), máquinas herramientas, posicionamiento de cabezales de lectura en discos magnéticos y de fuentes de radiación en radioterapia, radar, orientación de telescopios, etc. También se pueden aplicar a la medida de magnitudes que se pueden convertir en un desplazamiento por medio de un sensor primario adecuado.

O. Tipos de pantalla para sistemas inmersos

1) Pantallas LCD

La pantalla LCD es uno de los periféricos más empleados para la presentación de mensajes, variables y casi cualquier información proveniente de un microcontrolador. Gracias a su flexibilidad, buena visibilidad y precio reducido, se ha convertido en el estándar de visualización más utilizado con los microcontroladores.

Gráfica No. 13 Pantalla LCD Alfanumerica

(Jameco, 2007)

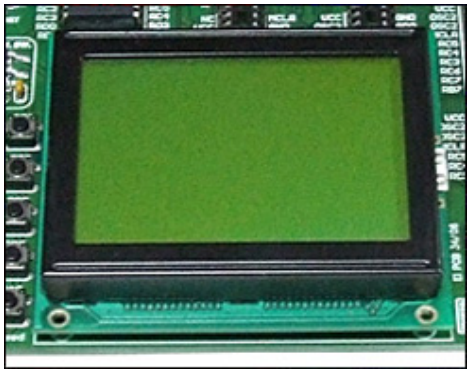
2) Algunos tipos de pantalla LCD

- LCD Alfanumérica: pantalla en la cual se pueden presentar caracteres y símbolos especiales en las líneas predefinidas del LCD. Su especificación viene dada como cantidad de caracteres por columna y número de filas. Por ejemplo: 2 x 16, 4 x 20.
- LCD Gráfica: pantalla en la cual se pueden presentar caracteres, símbolos especiales y gráficos. Su especificación viene dada en píxeles, por ejemplo 128 x 64.

3) Comunicación:

- LCD Paralela: los datos y comandos son enviados a través de un bus de datos paralelo, ya sea en modo de 4 ó 8 bits.
- LCD Serial: es capaz de recibir la información serial asincrónica utilizando un bus de sólo 2 líneas. Toda pantalla paralela puede ser convertida a serial mediante un circuito especializado, como por ejemplo el controlador serial para LCD modelo SLCD-IC.

Gráfica No. 14 Pantalla Utilizada, GLCD 128x64



(Mikroe, 2007)

Las pantallas de LCD poseen un microcontrolador integrado dentro de su propio módulo, el cual, se encarga de gestionar el control de sus terminales para la presentación de los caracteres. Muchos de estos almacenan un conjunto de letras y caracteres predefinidos en una memoria no volátil. El controlador más utilizado para las pantallas alfanuméricas es el HITACHI 44780.

Los controladores de estas pantallas suelen tener una memoria interna, que puede ser:

- CGRAM (Character Generator RAM) Memoria volátil de 64 bytes que permite almacenar hasta 8 caracteres personalizados para ser mostrados en la pantalla.
- SCRRAM (Display Data RAM) memoria volátil de 80 bytes, en la cual se almacenan los caracteres que se van a mostrar en la pantalla.

4) Pantalla Oled

Existe un grupo de pantallas con alta nitidez de imagen, brillantes y de poco consumo para los nuevos dispositivos pequeños. El principal beneficio de las nuevas pantallas de diodo orgánico emisor de luz (OLED) son las imágenes claras

y brillantes que producen. Las pantallas tipo OLED se refrescan con más rapidez, por lo cual son mejores para mostrar video. Los mejores modelos de matriz activa pueden mostrar casi cuatro veces más colores que los LCD equivalentes.

A diferencia de los LCD, los OLED emiten su propia luz electroluminiscente. Como resultado, las pantallas OLED son más brillantes y proporcionan una mayor nitidez que las LCD, además de su imagen no se ve afectada por el ángulo en que se observen. Y al no necesitar la luz de fondo que requiere un LCD permite el ahorro de energía en baterías.

Los OLED no suplantarán a los LCD de carrera. Son difíciles de fabricar, por lo cual aumentan el costo de cualquier dispositivo que use una pantalla OLED. Para el 2010, pronostica el analista de tecnologías de pantallas Paul Semenza, de ISuppli, las fábricas producirán anualmente 289 millones de pantallas OLED de matriz activa. Según estima ese analista, el 88 por ciento de las pantallas terminarán formando parte de teléfonos móviles.

Las pantallas OLED se han utilizado en otros medio como: en ciertos automóviles lujosos del año 2005 (como el Aston Martin DB9), el tablero de instrumentos ofrece una pantalla OLED de matriz activa para mostrar información. Incluso algunos estéreos para autos ya están disponibles con pantallas OLED monocromas. (J. Perenson, Melissa 2005)

Gráfica No. 15 Comparación pantalla Oled y LCD

(Yashenko, 2003)

Cuadro No. 1 Tabla de Comparación entre pantallas Oled y LCD

Pantalla LCD	Pantalla Oled
o No emite su propia luz, requiere de una luz de fondo o *backlight.*	o El material orgánico es electrofosforecente, es decir genera su propia luz.
o La luz trasera consume energía	o Requiere muy poca energía para su funcionamiento
o El ángulo de visión es menor por la forma de presentar la imagen.	o El ángulo de visión es casi 180 grados respecto a la imagen.
o Es más económico y más común.	o Debido a ser una tecnología relativamente nueva y poco desarrollada su precio es muy superior a una LCD de dimensiones similares.

La diferencia entre las pantallas se observa con la comparación de una imagen en cada una de las mismas, donde la pantalla oled se destaca por su luminosidad, que aparenta generar una mayor nitidez en la imagen en comparación de una LCD la cual es más opaca.

P. Pulsadores, rebotes y sistemas inmersos.

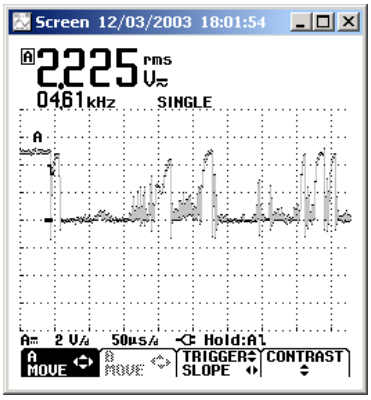
1) Rebotes debido a pulsadores o "push-buttons"

Un problema que se encuentra en cualquier circuito electrónico son los rebotes que generan principalmente los interruptores y pulsadores al cambiar de un estado a otro.

Los interruptores no producen un cambio tipo escalón cuando se accionan, como se espera teóricamente, sino después de cada pulsación o cambio de estado se producen una serie de oscilaciones rápidas, denominadas rebotes, que pueden afectar al correcto funcionamiento del circuito. Esto se acentúa en circuitos donde los rebotes afectan a las entradas de un microcontrolador y que en ocasiones puede afectar el funcionamiento del mismo de manera crítica.

Los rebotes están asociados a las características físicas de los propios interruptores/pulsadores, y aparecen en el momento en que cambiamos de estado y antes de adoptar el nuevo. En la gráfica no. 1 se puede apreciar como antes de que la señal se estabilice definitivamente, se produce un caso de rebotes múltiples al conmutar de 5V a 0V. (MicroLadder 2005)

Gráfica No. 16 - Vista de Salida de un Pulsador con Rebotes (400uS)

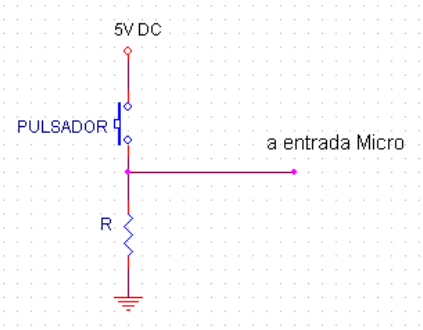


(MicroLadder 2005)

2) Protección anti-rebotes

El circuito de la gráfica no. 5 presenta un error común para la conexión de un pulsador a la entrada de un microcontrolador. En la configuración se busca que al presionar el pulsador, el potencial de la entrada del microcontrolador cambie a VDD.

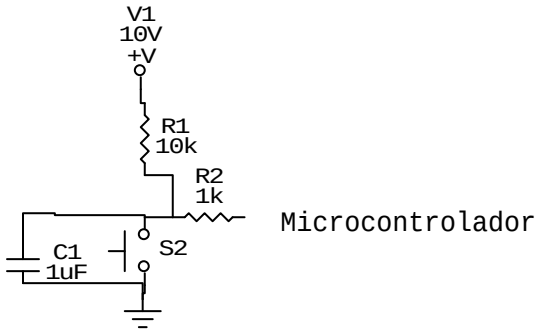
Gráfica No. 17 - Conexión Común de un Pulsador



(MicroLadder 2004)

Este pequeño circuito no posee protección anti-rebotes. Como se vio anteriormente, los rebotes pueden ser especialmente perjudiciales cuando se producen a la entrada de microcontroladores, por ello es aconsejable eliminarlos a través de un circuito antirebotes como el de la gráfica no. 3.

Gráfica No. 18 - Protección Antirebotes



(Microchip 2004)

Como se puede apreciar en la gráfica no. 3, añadiéndole una resistencia y un condensador se obtiene un filtro paso bajo, que elimina los rebotes de la entrada del microcontrolador. Las variaciones rápidas de voltaje (rebotes) son filtradas por el condensador a tierra, atenuando así el efecto del rebote y dejando a la entrada del microcontrolador solo los valores de VDD y 0V, que son los deseados. (MicroLadder 2004)

Q. Medios de Transmisión

Existen varios medios para la transmisión de datos. En la actualidad es muy común hablar de transmisión a través de cables, fibra óptica e incluso utilizando el espacio libre. Las señales transmitidas son afectadas por las características que el medio de transmisión impone, algunas de estas son el ancho de banda, la atenuación y el ruido e interferencia. (*TCM laboratory,1999)*

Entre los medios más utilizados se debe mencionar las líneas de transmisión más comunes:

- Cables paralelos: se recomienda cuando las distancias no superan los 50 m y las tasas de transmisión son menores que 20 Kbps. Esta es bastante utilizada por su simplicidad y bajo costo, sin embargo, es bastante susceptible al ruido.
- Par trenzado: se recomienda utilizar el par trenzado cuando se desea transmitir hasta 1 Mbps a 100 m de distancia y se puede incrementar la distancia de transmisión mientras se reduzca la tasa de transferencia. Este tipo de línea presenta mayor inmunidad al ruido debido a sus características físicas.
- Cable coaxial: esta línea soporta tasas de transferencia mayores a distancias mayores. Por ejemplo pueden transmitirse a 10 Mbps a varios cientos de metros de distancia. Por lo anterior esta medio se utiliza para conexiones punto-a-punto y multipunto.
- la fibra óptica: esta línea utiliza señales ópticas en lugar de señales eléctricas; posee la ventaja de que las señales viajan mas rápido pero a cambio de esto, su implementación es mas costosa.

Además de las líneas de transmisión, otros medios de transmisión importantes son:

- la transmisión satelital
- la transmisión por radio.

Las propiedades principales que se deben observar en un medio de transmisión son:

- Ancho de banda: es el rango de frecuencias que no presenta pérdidas significativas para un circuito.
- Atenuación: a medida que una señal es transmitida, su amplitud decrece debido a los efectos de propagación.

- Ruido: existen varias fuentes de ruido en un medio de transmisión; son comunes el ruido de fondo de la línea, el ruido por impulsos y el cruce de señales.
- Distorsión: ocurre distorsión ya que las componentes frecuenciales de la señal transmitida se propagan a distintas velocidades y con distintas atenuaciones.

R. Señales

Las señales se pueden dividir en dos grandes grupos según su naturaleza, analógicas o digitales. Cuando la señal que se desea transmitir es analógica, el ancho de banda de la señal debe coincidir con el ancho de banda aceptado por el canal mientras que cuando la señal es digital, la tasa de transferencia de bits de la señal debe estar de acuerdo a la tasa de transferencia del canal. (*TCM laboratory,1999)*

Para transmitir una señal ocasionalmente se necesita modularla para que se adapte correctamente y sea soportada por la línea de transmisión. En ocasiones la señal también es modulada para transformar la señal a una frecuencia óptima, determinada para un medio específico. Para realizar esta modulación, distintas variables de la onda electromagnética portadora pueden ser modificadas, por ejemplo, la amplitud, la frecuencia o la fase. (*TCM laboratory,1999)*

S. Señales Digitales

Estrictamente hablando, una señal digital se define como una función del tiempo que únicamente puede tomar valores discretos de amplitud. Si la señal digital es una señal binaria, solamente dos valores serán permitidos. *(Leon W. Couch, 2001)*

Algunas ventajas de la utilización de señales de comunicación digitales son:

- Los circuitos digitales que se utilizan para la comunicación son relativamente baratos.
- La privacidad se mantiene al permitir que las señales sean encriptadas.
- Es posible multiplexar señales de voz, video y datos para transmitirlas por un sistema de transmisión digital, con una sola línea de transmisión.
- A pesar de que muchas líneas de transmisión pueden presentar ruido, los errores en la información recibida serán menores; además de poder ser detectados y corregidos.

Así como existen ventajas de la utilización de sistemas de señales digitales, también existen desventajas y se puede mencionar entre estas la necesidad de aumentar el ancho de banda en relación a los sistemas analógicos y la necesidad de que los sistemas estén sincronizados para que la transmisión sea posible. *(Samson, 2007)*

Para transmitir señales digitales se puede utilizar dos métodos, la transmisión en paralelo, y la transmisión en serie.

- Con una transmisión en paralelo todos los bits de una unidad de información son transmitidos al mismo tiempo por un conjunto de líneas. Este tipo de transmisión es recomendado para distancias cortas, ya que los costos son elevados por requerir varias líneas de transmisión.
- Cuando la transmisión es en serie, se envían todos los bits de la unidad de información por una misma línea. Como resultado se observa que esta transmisión es más lenta, sin embargo es la solución ideal para transmisiones a distancias mayores debido a la reducción de costos.

Debido a los costos, como se mencionó anteriormente, la gran mayoría de señales digitales son transmitidas en serie. Este tipo de transmisiones tiene como característica principal la tasa de transferencia de bits (*bitrate*) que indica, como lo explica su

nombre, la cantidad de bits que se puede transmitir por el canal en cada segundo. *(Samson, 2007)*

T. Redes

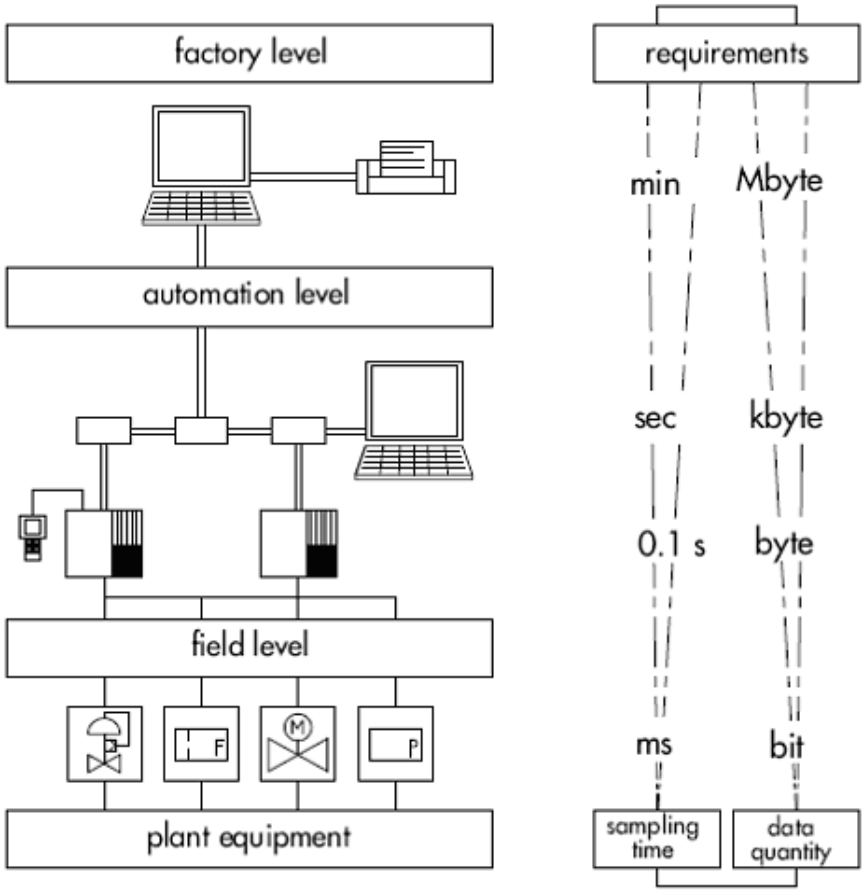
Desde hace varios años se ha tenido la necesidad de intercambiar información entre equipos electrónicos y computadoras, por ejemplo; para realizar dicho proceso se han utilizado señales digitales transmitidas en serie, como plataforma de los sistemas que actualmente se implementan. *(Giles, 2007)*

Existe una gran variedad de propiedades para caracterizar una red. Sin embargo, se puede decir que la más general es la distancia entre los nodos que se comunicaran, pero luego dependiendo de la aplicación de la red, se debe considerar la tasa de transferencia de la información, la cantidad de información enviada en cada ciclo de comunicación, la disponibilidad de utilizar la red en tiempo real, etc.

Una clasificación orientada a la práctica indica que las redes se pueden dividir cuatro niveles o jerarquías, como se muestra a continuación

El nivel mas bajo de la pirámide representa la red que debe tener el mejor gerenciamiento posible; esto ocurre ya que el tiempo de respuesta no es tan importante como la seguridad de transmitir toda la información, incluyendo mensajes largos. Debido a que este tipo de red representa la base de la pirámide se puede considerar que es la columna de la red. Algunos protocolos utilizados en este nivel son:

- MMS – Management Message Specification
- PROFIBUS-FMS
- TCP/IP – Transmission Control Protocol / Internet Protocol



Gráfica 23 Clasificación de Redes *(Giles, 2007)*

En el siguiente nivel, que es donde los nodos generalmente son sensores o actuadores, los mensajes, que son cortos, deben ser transmitidos a altas velocidades. Debido a la variedad de requerimientos que se han planteado, existe una diversidad de redes:

- ASI (Actuator-Sensor Interface),
- CAN (Controller Area Network),

- FF (FOUNDATION Fieldbus),
- InterBus-S,
- PROFIBUS-DP y -PA

En los dos niveles superiores de la pirámide mostrada, los mensajes transmitidos tienen un tamaño determinado y debido a los requerimientos se puede utilizar las siguientes redes:

- Bitbus,
- FF (FOUNDATION Fieldbus),
- FIP (Factory Information Protocol),
- LON (Local Operating Network),
- Modbus
- PROFIBUS-FMS/DP, dependiendo de la aplicación

Las estructuras, así como el software y hardware de las redes antes mencionadas, son bastante parecidos, ya que el objetivo de todas, sin importar el nivel en el que se encuentren es el mismo, transmitir los mensajes entre los nodos conectados sin problema alguno.

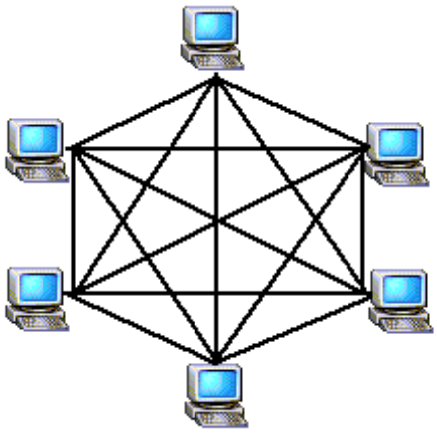
U. Topología de redes

Por topología de la red se entiende el arreglo físico y lógico que debe tener cada nodo, en relación a los otros nodos conectados a la red. A continuación se presentan las topologías de red básicas para las comunicaciones que son de tipo serial. *(Giles, 2007)*

1) Topología de malla

Este es el método mas complejo ya que se necesita conectar a cada uno de los nodos, con todos los otros nodos de la red, usando diferentes líneas de transmisión.

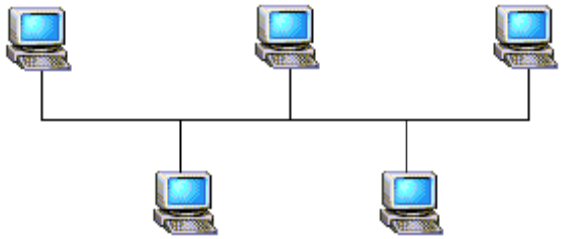
Cuando la red se expande, la flexibilidad de esta topología es inversamente proporcional a la cantidad de hardware que se requiere, por lo que esta raramente se utiliza en la práctica.



Gráfica 24 Topología de malla *(Giles, 2007)*

2) Topología de bus

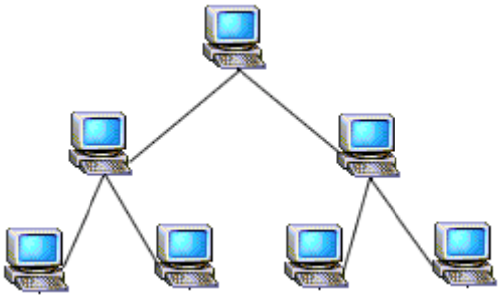
Una estructura mucho más simple es la topología de línea o bus. En este caso todos los nodos se comunican a través de una única línea de transmisión y están conectados de manera que cada nodo apenas requiere una interfaz para el bus. Desde que un mensaje transmitido puede ser "visto" por todos los participantes, los servicios de multicasting (envío de un mensaje dirigido a un grupo de nodos) y broadcasting (envío de un mensaje dirigido a todos los nodos) son fácilmente implementados. El medio de transmisión es controlado con los permisos de acceso al bus. Debido a las características mencionadas, el hecho de que algún nodo tenga problemas con su comunicación al bus, no implica que la red deje de funcionar.



Gráfica 25 Topología de bus *(Giles, 2007)*

3) Topología de árbol

Esta es similar, en algunos aspectos, a la topología de bus, excepto por el hecho de que la estructura permite que se conecten ramas o sub-buses, de la misma manera que se conectan los nodos. La ventaja de esta topología es que permite la conexión de redes a distancias mayores. De la misma manera que la topología de bus, existe una distancia máxima a la que la red puede trabajar, así como se limita el número máximo de nodos que se puede conectar.

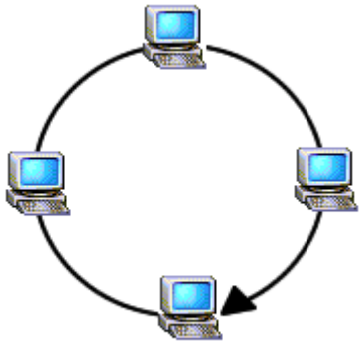


Gráfica 26 Topología de árbol *(Giles, 2007)*

4) Topología de anillo

Esta permite realizar varias conexiones de dos puntos creando un anillo físicamente. Para transmitir un mensaje cada nodo debe enviarlo al siguiente nodo hasta llegar a su destino. La ventaja de esta topología es que puede ocurrir

amplificación en cada nodo y por lo tanto las distancias que se pueden cubrir son mayores. Por otro lado, el riesgo que presenta es que si por algún motivo un nodo falla, la red entera puede dejar de funcionar.



Gráfica 27. Topología de anillo *(Giles, 2007)*

5) Topología de estrella

En esta topología todos los nodos se conectan a una estación central con líneas de dos puntos. Esta estación central puede ser activa, actuando como maestro en el control de la red, o puede ser pasivo, trabajando como acoplador de la estrella únicamente. En ambos casos, el mal funcionamiento del nodo central provoca que la red falle totalmente.

Gráfica 28 Topología de estrella *(Giles, 2007)*

Conociendo las redes básicas se puede conseguir estructuras más complejas con la combinación de estas, como por ejemplo una red de subredes. Cada subred podría tener una topología distinta, pero de cualquier forma cada nodo debe ser capaz de distinguir a los otros nodos, así como debe poder direccionarlos directamente.

V. Protocolos de transmisión – Modelo ISO/OSI

Los protocolos de transmisión definen las reglas para el envío de bloques de información de un nodo a otro dentro de una red. El modelo estándar para los protocolos de comunicación de redes es el *International Standard Organization's Open System Interconnect* (ISO/OSI). El objetivo de este modelo es determinar una arquitectura que defina las tareas de comunicación. (*Cisco Systems, 2007)*

La idea básica de ISO/OSI es que la comunicación entre nodos para alguna tarea determinada es demasiado compleja para considerarla como una sola unidad.

El modelo ISO/OSI define siete capas de la red, no teniendo importancia, para este modelo, el protocolo que se utilice en cada capa, sino que la importancia recae sobre el funcionamiento general de la capa y su interfaz para las capas tanto superiores como inferiores. *(Burden's, 2007)*

A continuación se hace una breve descripción de cada una de las capas del modelo:

1. Capa física: Esta capa define el medio de transmisión en sí. Todos los medios son funcionalmente equivalentes siendo la mayor diferencia los costos de instalación y mantenimiento. Cabe mencionar que los convertidores de un medio de transmisión a otro operan en este nivel.

2. Capa de enlace de datos: En esta capa se define el formato de la información en la red. La trama de datos de la red incluye el chequeo de la

transmisión, las direcciones de fuente y destino, así como la información que se esta transmitiendo. En esta capa se define la unidad máxima de transmisión (Maximun Transmisión Unit) que es el paquete de información mas grande que puede ser transmitido por la red. Esta capa maneja las conexiones físicas y lógicas para que el paquete pueda ser entregado a la interfaz de destino.

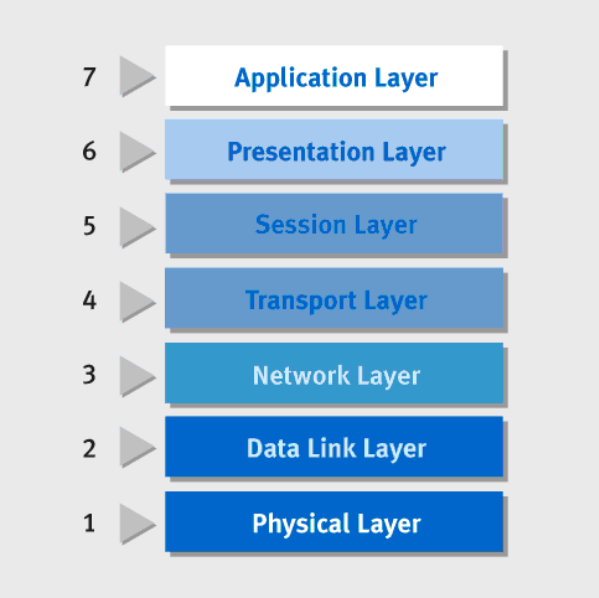
3. Capa de red: Esta capa define como se enrutan los mensajes para que puedan ser entregados. Este direccionamiento esta separado de la parte de hardware que implementa las conexiones de red.

4. Capa de transporte: Esta es la capa que provee la interfaz para las capas de alto nivel. Con esta capa se logra enmascarar la red de hardware y la topología, de la vista de las aplicaciones. Los protocolos que se utilizan en esta capa son altamente complejos para lograr los resultados deseados.

5. Capa de sesión: Estos protocolos especifican como se establece la sesión de comunicación con un sistema remoto. En esta capa también se especifican los detalles de seguridad tales como la utilización de claves de acceso.

6. Capa de presentación: En esta capa se indica como se presenta la información a cada nodo. Estos protocolos son necesarios porque se permite que cada nodo utilice la representación interna más conveniente para su funcionamiento.

7. Capa de aplicación: En esta capa es donde se encuentra la aplicación que requiere la utilización de la red. Una de las aplicaciones más comunes es la transferencia de archivos y el correo electrónico, por ejemplo.



Gráfica 29. Capas definidas en el modelo ISO/OSI *(CAN in Automation (CiA), 2007)*

W. Protocolo CAN

CAN (*Controller Area Network*) es un sistema avanzado de comunicaciones en serie a través de un bus que soporta de forma eficiente distintos sistemas de control. Este protocolo especifica principalmente la segunda capa del modelo ISO/OSI, determinando la forma como se enlazan los datos, pero además determina parte de la capa física de la red. Internacionalmente está estandarizado como ISO 11898-1; también esta avalado por SAE (*Society of Automotive Engineers*) y presenta las siguientes características:

- CAN es un bus multi-maestro con una estructura lineal y abierta con un bus y nodos conectados al mismo. El número de nodos no esta limitado por el protocolo.

- Los nodos conectados al bus no poseen una dirección específica; por el contrario, las direcciones están contenidas en los identificadores de los mensajes transmitidos, indicando el contenido y prioridad del mensaje.

- El número de nodos puede ser modificado dinámicamente sin distorsionar la comunicación de otros nodos.

- CAN soporta los servicios de *Multicasting* y *Broadcasting*.

- CAN permite una detección de errores sofisticada y provee mecanismos para el manejo de estos. Además, CAN posee una alta inmunidad a la interferencia electromagnética.

- CAN permite el reenvío de mensajes que hayan sido entregados con algún error y los errores temporales pueden ser recuperados.

- CAN utiliza la codificación NRZ (*Non-Return-To-Zero*).

El acceso al bus de la red se maneja con protocolos de comunicación serial avanzada para detectar la utilización del bus, así como la detección de colisiones con un arbitraje no destructivo; lo anterior indica que las colisiones se evitan sin perdida de tiempo.

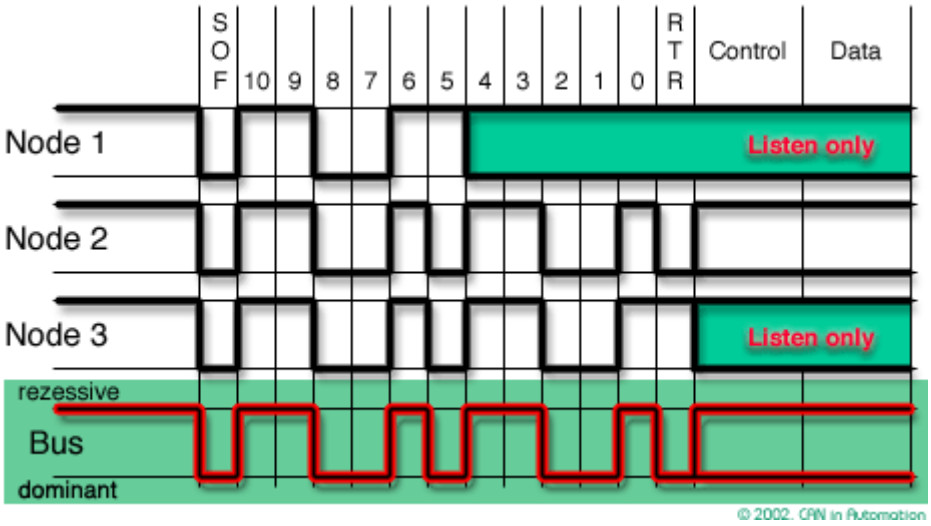
1) Conceptos básicos

El bus puede tomar dos estados, conocidos como dominante y recesivo. Para detectar el estado del bus, este utiliza un mecanismo lógico que se asemeja a una compuerta AND, en la cual es frecuente que un bit cero-lógico pueda ser dominante ante un bit uno-lógico (recesivo). Con la información anterior se puede determinar que el bus estará en estado recesivo únicamente cuando todos los nodos transmitan

mensajes recesivos ya que si un único nodo transmite mensajes dominantes, consecuentemente el bus entrará en estado dominante. *(CiA, 2007)*

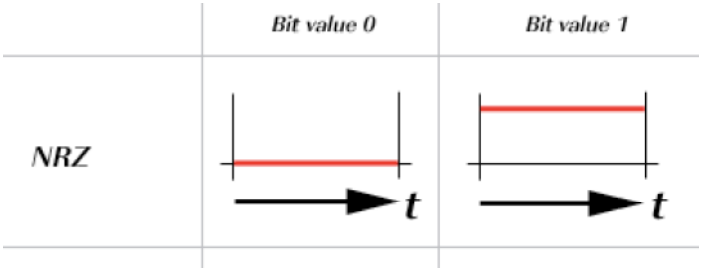
El protocolo CAN controla el acceso al bus con el concepto de acceso múltiple con detección de ocupación del bus (CSMA, por sus siglas en ingles *Carrier Sense Multiple Access*), además del arbitraje de la prioridad de los mensajes.

Este arbitraje evita las colisiones de los mensajes que iniciaron la transmisión simultáneamente y asegura que el mensaje con mayor importancia sea entregado lo mas rápido posible, sin perdidas de tiempo. Cada nodo envía los bits de identificación y monitorea el nivel del bus. Como ya se ha mencionado anteriormente, cuando un bit dominante es enviado, el estado del bus será dominante también; sin embargo, si se envía un bit recesivo, el estado del bus dependerá de que otros nodos estén transmitiendo al mismo tiempo. El estado recesivo del bus indica que no ha habido colisiones mientras que el estado dominante indica que por lo menos un nodo esta enviando un mensaje dominante. Cuando un nodo recibe un bit dominante mientras está enviando un bit recesivo, automáticamente pierde el arbitraje y retira su transmisión. Los nodos que pierden el arbitraje, automáticamente tratarán de repetir la transmisión tan pronto como el bus se encuentre libre.



Gráfica 30. Proceso de arbitraje para el acceso de los nodos al bus *(CiA, 2007)*

Como ya se mencionó anteriormente, el protocolo CAN utiliza la codificación NRZ, lo que significa que una señal es constante para el tiempo total de transmisión de un bit y únicamente un segmento de tiempo se necesita para representar cada bit.



Gráfica 31 Codificación NRZ *(CiA, 2007)*

Dado que durante largas transmisiones con una misma polaridad (1 o 0) no ocurren cambios automáticos en la línea, es necesario que los nodos puedan ser resincronizados; la manera mas adecuada para solucionar este problema es que cuando ocurra la transmisión de 5 bits con la misma polaridad se debe insertar un

bit adicional de la polaridad opuesta antes de continuar la transmisión. El nodo receptor, a diferencia del nodo transmisor, reconocerá que un bit fue insertado, por lo que lo removerá para obtener el mensaje original.

2) Tramas de la señal transmitida

Existen varias tramas posibles para el envío de señales, estas son:

a) Trama de datos

Este es el tipo de trama más importante. Esta se genera cuando el nodo CAN desea transmitir alguna información. La trama tiene los siguientes componentes:

1. *Inicio de la trama* (Start of Frame, SOF), es el bit que permite la sincronización de todos los nodos. *(CiA, 2007)*

2. *Campo de arbitraje*, este campo consiste de 11 o 29 bits, según la versión de protocolo CAN que se utilice. En este campo se determina el contenido y la prioridad del mensaje transmitido. *(CiA, 2007)*

3. *Campo de control*, es formado por 6 bits siendo el primero el bit de identificación de la extensión del mensaje; el siguiente bit esta reservado por el protocolo; los 4 bits restantes indican la cantidad de bytes de información que contiene el mensaje (0-8). *(CiA, 2007)*

4. *Campo de información*, es el campo donde se transmite la información propia del mensaje. *(CiA, 2007)*

5. *Campo de chequeo de información* (CRC), este campo se utiliza para detectar posibles errores de la transmisión del mensaje. Este campo contiene 15 bits, mas un bit delimitador. *(CiA, 2007)*

6. *Campo de confirmación* (ACK); este campo de dos bits sirve para que con el CRC, los otros nodos de la red confirmen, modificando el estado del bus, que el mensaje fue entregado correctamente y en caso de que ningún nodo confirme que el mensaje se transmitió correctamente, el nodo transmisor considerara que ocurrió algún error. *(CiA, 2007)*

7. *Fin de la transmisión*, (End Of Transmission, EOF), se envían 7 bits para confirmar el final de la transmisión. *(CiA, 2007)*

Gráfica 32 Representación de la Trama de datos *(CiA, 2007)*

b) Trama de solicitud de información

CAN permite que un nodo solicite información a otro nodo. Esta trama incluye el identificador del mensaje que se está solicitando, por lo que el nodo con la información requerida transmitirá la respuesta. Esta trama no tiene campo de información.

c) Trama de errores

La trama de errores puede ser generada por cualquier nodo que detecta un error en el bus. Esta trama consiste de dos campos, una bandera de error, seguida de un campo delimitador del error. El campo delimitador posee 8 bits que permiten que las comunicaciones sean reiniciadas luego del error.

d) Trama de sobrecarga

Esta trama tiene el mismo formato que una trama de error; la diferencia que existe entre las dos tramas es el momento en el que se originan ya que las tramas de error ocurren durante la transmisión de un mensaje, mientras que la trama de sobrecarga ocurre únicamente en el espacio entre las tramas. Generalmente esta trama se genera cuando un nodo no esta preparado para recibir la información en el bus. Para que la transmisión del siguiente mensaje sea retardada es necesario que ocurran dos tramas de sobrecarga.

El espacio entre las tramas se encarga de separar una trama de cualquier tipo de una trama de información o de solicitud de información. Este espacio se compone de por lo menos 3 bits recesivos.

3) Manejo de errores

Como ya se ha mencionado, el protocolo CAN tiene un sistema de detección de errores sofisticado. Todos los errores son públicos y se envían a todos los nodos con las tramas de error. La transmisión de mensajes erróneos se interrumpe tan pronto como se detecta el error y el nodo transmisor intentará de repetir la transmisión tan pronto como le sea posible.

- El primer mecanismo de detección de errores es el chequeo cíclico de redundancia (CRC), en el cual el nodo transmisor calcula la suma de la secuencia de bits desde el inicio de la trama hasta el final del campo de información. Esta secuencia se transmite en el campo CRC de la trama de información y el nodo receptor calculará de manera similar una secuencia CRC y la comparará con la secuencia recibida para verificar que ambas sean iguales. En caso de que las secuencias CRC sean distintas, habrá ocurrido un

error y el nodo receptor transmitirá una trama de error, solicitando la retransmisión del mensaje erróneo.

- El segundo sistema de detección de errores es el chequeo de confirmación del campo ACK de la trama de datos, con el cual el transmisor coloca dos bits recesivos en el bus, y si algún nodo receptor modifica el estado del bus, se confirmará que el mensaje fue transmitido correctamente. En caso contrario, el mensaje será retransmitido, aunque no se genere una trama de error.
- El tercer mecanismo es el chequeo de tramas; si el nodo transmisor detecta un bit dominante en el segmento de CRC, ACK, EOF o en el espacio entre tramas, entonces habrá ocurrido un error y se generará una trama de error solicitando el reenvío del mensaje.
- El monitoreo de bits es otro mecanismo y ocurre en todos los nodos de la red. Un error de bit ocurre si un nodo transmisor envía un bit dominante pero detecta un bit recesivo, o viceversa. Este proceso no ocurre durante el proceso de arbitraje ya que se permite que el estado del bus sea modificado.
- Si por algún motivo se transmiten seis bits de la misma polaridad se generara una trama de error y se solicitara la retransmisión del mensaje, ya que como se mencionó anteriormente, se tolera un máximo de cinco de bits de la misma polaridad.

4) Sincronización del bus

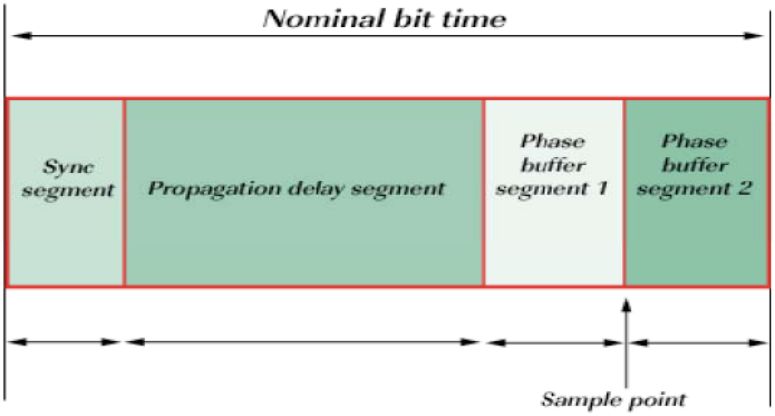
CAN maneja la transmisión de mensajes síncronos; es decir que todos los nodos son sincronizados al inicio de cada mensaje con el primer flanco descendente de la trama que pertenece al bit SOF. Para garantizar que la señal se detecta correctamente hasta el final de cada mensaje, se hace una re-sincronización del bus en cada cambio de estado del bus de recesivo a dominante.

5) Tiempo de cada bit

El tiempo de un bit CAN es especificado como cuatro segmentos no traslapados. Un segmento se construye a partir de un múltiplo entero de la menor resolución discreta de tiempo utilizada por el nodo CAN (TQ, Time Quantum). La duración del bit se genera a partir de la frecuencia del oscilador. Por lo anterior, la tasa de trasferencia de bits se selecciona con la programación de la longitud de TQ y el número de TQ en los segmentos. Normalmente el tiempo de un bit varía entre 8 y 25 TQ.

Los cuatro segmentos del tiempo de un bit son:

- Segmento de sincronización. Este segmento siempre tiene duración de un TQ. Si existe algún cambio de estado entre el bit anterior y el actual, los nodos receptores esperan que el cambio ocurra en este segmento.
- Segmento de propagación. Este segmento sirve para compensar el retardo de la propagación de la señal a lo largo de la red. Se considera que este segmento puede durar entre uno y ocho TQ.
- Adicionalmente existen dos segmentos que se conocen como buffer de fase 1 y 2, los cuales se utilizan para compensar los errores de fase del flanco.



Gráfica 33. Representación del tiempo de un bit *(CiA, 2007)*

6) Versiones del protocolo CAN

Las especificaciones originales del protocolo presentaban un identificador de 11 bits para cada mensaje; actualmente estas especificaciones se conocen como *Standard CAN*. Luego, para aumentar la cantidad de identificadores existentes, se aumentó el tamaño del identificador a 29 bits, y esta nueva definición se reconoce como CAN extendido, permitiendo la administración de 536 millones de identificadores posibles. *(CiA, 2007)*

Actualmente se utilizan tres tipos de módulos CAN:

2.0A – maneja identificadores de 11 bits, considerando los identificadores de 29 bits, como errores.

2.0B Pasivo – maneja identificadores de 11 bits, ignorando los mensajes con identificador de 29 bits.

2.0B Activo – maneja ambos identificadores.

X. Estándar IEEE802.11

El estándar IEEE802.11, también conocido como Wi-Fi, establece un grupo de normas para regular las redes de Wireless LAN, o WLAN. Estas redes son las redes inalámbricas utilizadas por todo el mundo hoy en día, y el estándar incluye las famosas redes 802.11a, 802.11b, y 802.11g. Más recientemente se esta finalizando el estándar para presentar la 802.11n, que presenta mayor alcance y mayor velocidad del transmisión. Para los fines de este trabajo nos centraremos en los estándares 802.11b y 802.11g por ser el estándar que se implementará en el desarrollo del sistema.

1) Historia

El estándar IEEE802.11 fue desarrollado en 1997 y presentaba una velocidad de transmisión máxima de 2 Mbps. El estándar estaba desarrollado para ser implementado a través de puertos IR, o a través de sistemas RF de salto de frecuencias. Opera a una frecuencia de 2.4 GHz, y ofrecía "Carrier Sense Multiple Access with Collision Avoidance" CSMA/CA como medio para obtener acceso a la red.

La idea detrás de CSMA/CA no es detectar las colisiones, sino prevenirlas cuando es más probable que ocurran: en el momento que se libera el bus. En el momento que el bus es liberado, todos los equipos esperan un tiempo aleatorio antes de ver si se esta utilizando el bus. Si el bus está siendo utilizado, los equipos duplican el tiempo de espera, y cuando se vuelve a vencer, intentan a enviar de nuevo. Antes de cada transmisión se espera un DIFS (Distributed Coordination Inter Frame Space), entre cada acuse de recibo (ACK) se espera un SIFS (Short Inter Frame Space). Esto reduce significativamente la cantidad de colisiones, ya que es poco probable que dos equipos comiencen a transmitir simultáneamente. (Vos)

Una de las más grandes desventajas de este estándar original es que ofrecía demasiadas opciones de implementación, por lo cual interoperabilidad entre varias marcas era muy difícil. El estándar 802.11 original no tuvo mayor auge, y fue aceptado comercialmente cuando salió la revisión al estándar 802.11b. (Nygaard, 2002)

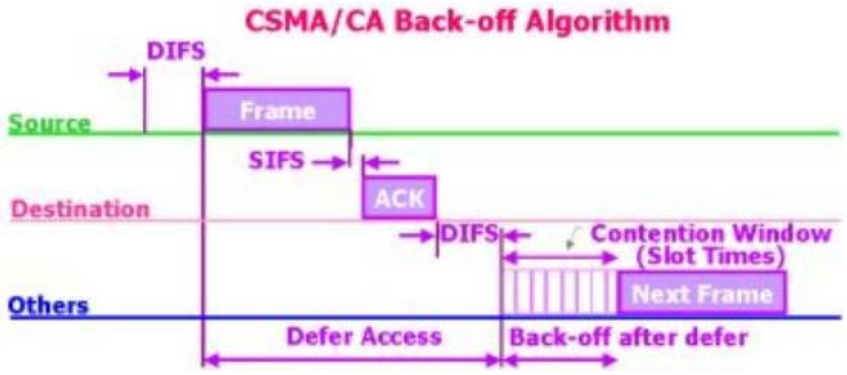


Figura 1. Algoritmo CSMA/CA (Nygaard, 2002)

2) 802.11b

La revisión al estándar 802.11 que siguió fue la de 802.11b. Esta revisión incluía velocidades de transmisión de hasta 11 Mbps, y seguía funcionando a la misma frecuencia, 2.4 GHz. 802.11b utiliza el mismo sistema CSMA/CA para acceder a la red, por lo cual la velocidad real de transmisión es de solamente 5.9 Mbps en TCP y 7.1 Mbps en UDP. El alcance que lograban estas redes era significativamente mayor al del estándar 802.11 original, y llegaba hasta 90 m en exteriores. (Flickenger, 802.11b Tips, Tricks, and Facts, 2001)

802.11b usa DSSS (Direct-Sequence Spread Spectrum), a velocidades bajas. El DSSS funciona multiplicando la señal a ser transmitida por una señal de "ruido". Esta señal de ruido es una secuencia al azar de "1" y "-1", a una frecuencia mucho

mas alta que la de la señal. Esta señal tiende a parecerse al ruido blanco de las grabaciones de audio, pero la diferencia es que este "ruido" puede ser filtrado en el lado receptor para recuperar la señal transmitida. Esta tecnología tienes las ventajas de ser resistente a interferencia intencional y no intencional, se puede utilizar un mismo canal para varios usuarios, y se puede determinar el tiempo relativo entre transmisor y receptor. (Maxim, 2003)

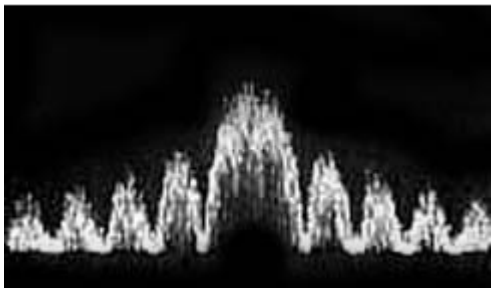


Figura 2. Espectro de frecuencia de una señal DSSS (Maxim, 2003)

Cuando se transmite a altas velocidades se utiliza Complementary Code Keying (CCK), que es una variación de CDMA, por lo cual era fácil desarrollar chips nuevos, ya que solo eran actualizaciones de otros ya disponibles. Complementary Code Keying funciona produciendo 64 palabras de ocho bits para codificar información. Estas palabras codificadas tienen propiedades matemáticas únicas, las cuales permiten que se distinga una de la otra, incluso en la presencia de interferencia. Funciona en conjunto con DSSS, aplicando rutinas matemáticas complejas a los datos, permitiendo que más volumen de datos pueda ser representado en un mismo ciclo de reloj. Gracias a CCK y el incremento dramático de velocidad, aumento de distancia, interoperabilidad, y llegada rápida al mercado, la revisión 802.11b rápidamente se convirtió en la tecnología de WLAN de preferencia. (Zyren, 2001)

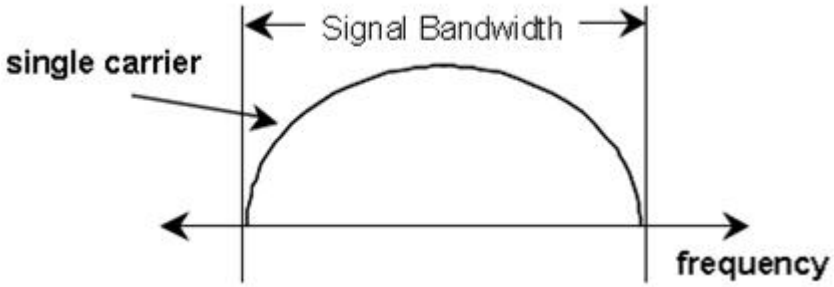


Figura 3. Señal CCK modulada en una frecuencia portadora (Zyren, 2001)

Las redes 802.11b pueden utilizarse en varias configuraciones. La más común es Punto – Multi Punto, en la cual por medio de una antena omni – direccional varios equipos se conectan a un mismo punto de acceso. La distancia típica de transmisión es 30 m en interiores, y 90 m en exteriores, pero a velocidades de 1 Mbps. (Flickenger, 802.11b Tips, Tricks, and Facts, 2001)

Con antenas de alta ganancia, se pueden obtener transmisiones de hasta 5 km fácilmente, y han sido reportadas distancias de hasta 20 km con equipos especiales y manteniendo línea de vista. Esto resulta muy útil cuando no se desea contratar una línea dedicada, y cuando no se desea utilizar equipo microondas muy sofisticado. (Flickenger, A Wireless Long Shot, 2001)

La revisión 802.11b tiene "Adaptive Rate Selection", lo que significa que puede transmitir a 11 Mbps, pero esto puede ser bajado a 5.5, 2 y hasta 1 Mbps cuando la calidad de la señal sea muy mala. Dado que las velocidades más bajas de transmisión utilizar métodos menos complejos y más redundantes para codificar los datos, estos resultan menos susceptibles a corrupción y perdidas. Existen extensiones hechas a 802.11b que permiten transmisión de datos a 22, 33 y 44 Mbps, pero estas son propietarias de varias empresas, y no son apoyadas ni aprobadas por la IEEE. (Flickenger, 802.11b Tips, Tricks, and Facts, 2001)

3) 802.11g

En Junio 2003 se ratifico otra revisión al estándar 802.11. Esta era llamada 802.11g, la cual daba mayor velocidad, mayor alcance y seguía funcionando a 2.4 GHz. La velocidad máxima de este estándar es 54 Mbps, y tiene un alcance máximo de unos 100 m.

802.11g utiliza "Orthogonal Frequency Division Multiplexing" (OFDM) como tecnología de modulación. OFDM funciona utilizando señales sub – portadoras (señales ya moduladas, que después se vuelven a modular a una frecuencia más alta de la original). Estas señales son producidas utilizando el algoritmo de la Transformada Rápida de Fourier. Dado que las sub – portadoras son ortogonales entre si, se evita la implementación de filtros para cada canal de transmisión. (Zyren, 2001)

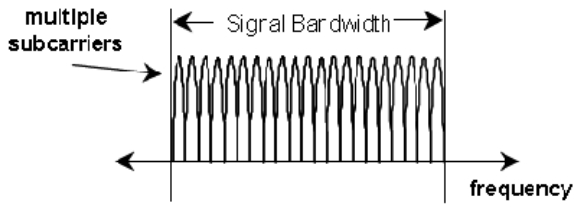


Figura 4. Señal modulada con OFDM (Zyren, 2001)

La tecnología OFDM permite que se transmita a velocidades máximas de 54 Mbps, y se puede ir revertiendo hasta 1 Mbps, en caso de pérdida de calidad. Una de las grandes ventajas de la revisión 802.11g es que permite compatibilidad con 802.11b, lo que significa que redes o equipos que utilizaran 802.11b podían fácilmente integrarse a redes nuevas, y también equipos nuevos con 802.11g pueden integrarse fácilmente con redes más antiguas. (Zyren, 2001)

IEEE WLAN Standard	Over-the-Air (OTA) Estimates	MAC Estimates
802.11b	11 Mbps	5 Mbps
802.11g	54 Mbps	25 Mbps (11b no presente)
802.11a	54 Mbps	25 Mbps
802.11n	**200+ Mbps**	**100 Mbps**

Tabla 2. Velocidades de Transmisión acorde al estándar IEEE 802.11 (Intel)

4) Funcionamiento

El 802.11b tanto como el 802.11g trabajan a 2.4 GHz, y para operar dividen esta banda en 14 canales solapados, con frecuencia central a 5 MHz de distancia. El estándar no especifica el ancho de cada canal, sino más bien especifica la frecuencia central. Es requisito del estándar que cada canal sea atenuado 30 dB de su energía máxima a 11 MHz de la frecuencia central, y 50 dB de su energía máxima a 22 MHz de la frecuencia central. (Zyren, 2001)

Figura 5. Distribución de canales en el estándar 802.11 (Nygaard, 2002)

Dado que la especificación de energía solo se describe hasta 22 MHz de la frecuencia central, se puede pensar que más allá de eso no habrá interferencia, pero si una señal es demasiado fuerte en comparación con otra, puede ocurrir interferencia hasta entre el canal 1 y el 11. (Cisco, 2004)

5) Desventajas

Una de las principales desventajas de 802.11b y 802.11g es precisamente que operan a 2.4 GHz. Esta frecuencia esta ya demasiado poblada por varios dispositivos, como teléfonos inalámbricos y hornos microondas. Otra desventaja de interferencia es que muchos dispositivos tienen como predeterminado un mismo canal, lo cual produce una gran congestión en ese canal, causando deficiencia de velocidad, así como de calidad. (Flickenger, 802.11b Tips, Tricks, and Facts, 2001)

Aunque el estándar incluye seguridad para la red, denominada WEP (Wired Equivalent Privacy), esta ha sido probada fácil de quebrar, lo cual puede comprometer redes con información sensible. Otro problema es que muchos dispositivos comprado tienen como predeterminado no ofrecer seguridad, entonces usuarios principiantes podrían ofrecer acceso a su red sin saberlo. (Geier, 2002)

Una desventaja de ámbito internacional es que las limitaciones y leyes de operación no son definidas mundialmente, lo cual causa problemas de compatibilidad, ya que hay países como EE.UU. que usan canales del 1 – 11, mientras Europa usa 1 – 13, Japón del 1 – 14, y en otros países eliminan algunos canales. (IT-Expert On Call, 2007)

Y. Protocolo TCP/IP

El TCP/IP fue desarrollado por el Departamento de Defensa de EE.UU. para unir redes de diferentes empresas en una "red de redes", o lo que se conoce hoy como Internet. Tuvo auge inicialmente gracias a que proveía de varios servicios básicos (transferencia de archivos, correo electrónico y logon remoto) a una gran variedad de usuarios y servidores. Un departamento puede utilizar TCP para comunicación en una LAN, y la porción IP provee direcciones para conectarse a la red empresarial, la red regional, y ultimadamente a Internet. Dado que el protocolo TCP/IP fue diseñado para uso en campos de batalla, era de esperarse que algún nodo o línea de transmisión sufriera daños, por lo cual el Departamento de Defensa construyó el protocolo muy robusto y con recuperación automática de daños, lo cual permite redes grandes con poca administración central.

1) TCP

Transmision Control Protocol, o TCP es uno de los protocolos básicos del grupo de protocolos de Internet. Utilizando TCP, dispositivos en redes pueden crear

conexiones entre unos y otros, con las cuales pueden intercambiar datos. El protocolo también garantiza la entrega confiable y en orden, así como también distingue datos de varias conexiones por diferentes aplicaciones, por ejemplo, email, servidores Web, http, que se ejecutan en el mismo dispositivo. El protocolo TCP opera en la capa 4 del modelo OSI, la cual corresponde a la capa de Transporte.

2) Funcionamiento

El protocolo TCP es intermediario entre el Protocolo de Internet (IP), que se encuentra en la capa 3, y la sesión que se encuentra en la capa 5. Las aplicaciones envían series de datos (bytes) los cuales el TCP empaqueta en segmentos de tamaño apropiado (usualmente delimitados por el "Maximum Transmission Unit" MTU). El TCP después envía los segmentos al IP, el cual se encarga de que sean transmitidos por la red hasta un receptor TCP en el otro lado. (TCP/IP Guide, 2005)

El TCP se asegura que no se haya perdido ningún paquete, asignándole un número de secuencia a cada uno, asegurando también que sean recibidos en el orden correcto. Cuando el receptor recibe todos los paquetes, envía una respuesta. El TCP también implementa un contador, el cual si expira antes de recibir la respuesta, asume que el paquete fue perdido, y procede a reenviar. También implementa un "checksum", el cual es generado por el transmisor y corroborado por el receptor, para asegurar que los paquetes que se recibieron contienen la información correcta. (Cisco, 1996)

El TCP presenta varias ventajas, principalmente la entrega ordenada y correcta de toda la información. Esto resulta muy importante en aplicaciones donde la integridad de la información transmitida es muy importante, y donde no se puede perder datos, ni recibir en el orden equivocado. Para transmisiones donde no se requiere la entrega ordenada, y donde no importa si se pierden algunos paquetes,

suele usarse UDP o “User Datagram Protocol”, pero este tiende a tener muchas perdidas de información, por lo cual no es adecuado para muchas aplicaciones. (Cisco, 1996)

3) IP

El Protocolo de Internet (IP) es un protocolo orientado a datos utilizado para comunicar datos a través de una red en los cuales paquetes son ruteados por diferentes nodos, hasta llegar a su destino final. El IP funciona en la capa de Red, y esta encapsulado en la capa de Enlace de Datos, según el modelo OSI. Siendo este un protocolo de capas menores, ofrece direcciones únicas entre computadores en redes. (TCP/IP Guide, 2005)

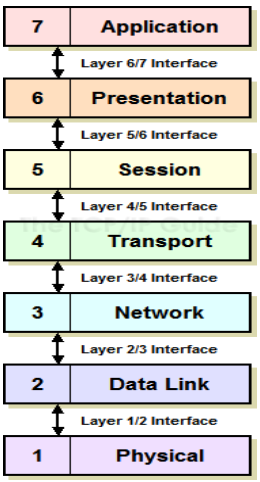


Figura 6. Diagrama del modelo OSI de redes (TCP/IP Guide.com, 2005)

4) Funcionamiento

El IP recibe paquetes de datos de capas superiores del modelo OSI, y las prepara para ser enviadas. El IP es un protocolo que no necesita conexiones, y esta solamente interesado en el destinatario final del paquete. Dado en encapsulamiento provisto por el modelo OSI, el protocolo IP puede ser implementad en redes

conteniendo varias tecnologías (Ethernet, ATM, Wi-Fi) sin consecuencias para capas superiores. (Gilbert, 1995)

El ruteo de paquetes IP es un ruteo dinámico, o sea, cuando el paquete va a ser enviado, el equipo de ruteo calcula la ruta a tomar en ese momento, proveyendo que el ruteo pueda adaptarse a condiciones cambiantes de la red, así asignando al paquete la ruta más adecuada. Al principio de las rutas no se sabe exactamente por donde se irá el paquete, sino que en cada parada, el nodo decide cual es la siguiente parada comparando la dirección de destino con una tabla de direcciones que tiene almacenada. El papel de cada nodo es solo de reenviar el paquete basado en información interna, no de presentar informes de errores o anomalías al transmitir. (Gilbert, 1995)

El protocolo IP provee una transmisión de datos no confiable. Esto significa que la red no garantiza la entrega de paquetes, y que los paquetes pueden sufrir corrupción de datos, perdida del orden, duplicados y paquetes perdidos. La única función de prevención de errores es la revisión del encabezado del paquete con el "checksum", y si se encuentra algún error, el paquete es descartado inmediatamente, sin aviso al transmisor. La revisión de errores de transmisión corresponde a capas superiores, y debe ser manejado por estas. (Gilbert, 1995)

Cuando los paquetes recibidos de capas superiores son más grandes que el MTU especificado, el IP se ve forzado a fragmentar los datos y enviarlos en paquetes más pequeños. El IP sí ofrece el servicio de reordenar estos fragmentos, dejándolos en el estado original como fueron enviados. (Gilbert, 1995)

La razón por la cual el IP no es tan confiable es para prevenir que los routers se vuelvan muy complejos. Esto permite que los routers hagan lo que quieran con los paquetes, así proveyéndole al usuario una mejor experiencia al utilizar el protocolo. (Gilbert, 1995)

5) Formato Paquetes TCP/IP

Los paquetes TCP/IP contienen datos que son trasladados desde capas superiores hasta llegar a la capa de Enlace de Datos, la más baja. En la figura a la derecha se muestra el camino que toman los datos desde niveles superiores hasta la capa más baja. Como se puede ver, el nivel TCP recibe información de una aplicación que contiene sus propios encabezados y datos. El TCP toma este dato y le agrega su propio encabezado TCP, encapsulando el dato. Cuando el TCP ya encapsuló el dato, lo envía a la capa donde el IP le agrega sus propios encabezados, y así encapsula el dato obtenido de TCP. Esto continúa hasta llegar a la capa Física, donde ya es enviado. Al ser recibido el dato en la capa física este es subido hasta la capa 3, donde el protocolo IP mira la dirección del destinatario, y si es él, quita el encapsulado IP y lo envía a la capa de Transporte donde TCP quita sus encabezados, y envía el dato recibido a la aplicación correspondiente. (TCP/IP Guide, 2005)

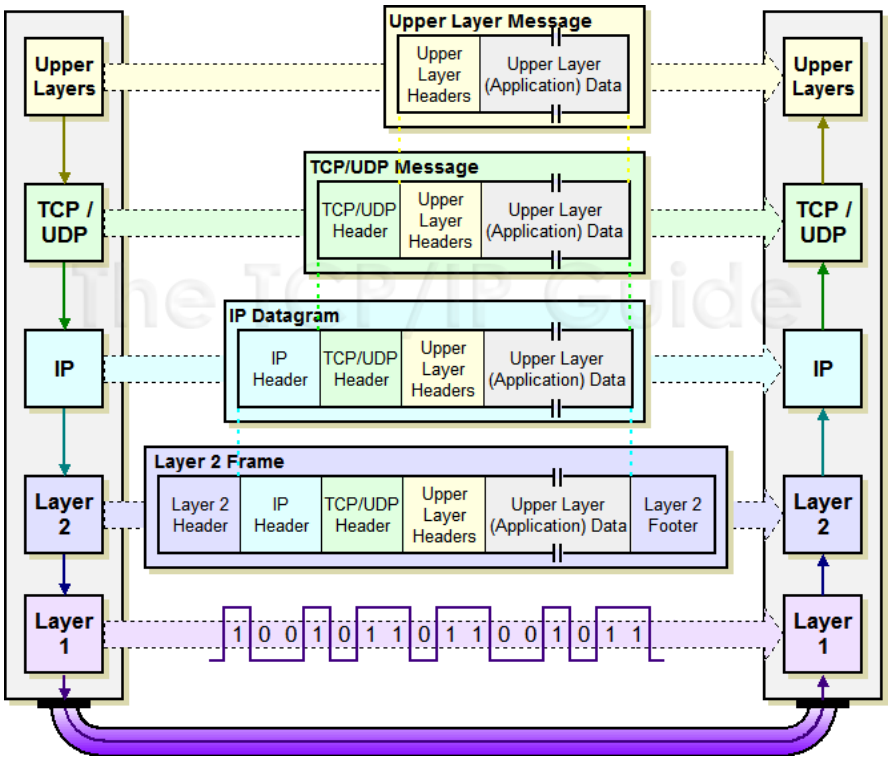


Figura 7. Diagrama de mensaje TCP/IP (TCP/IP Guide, 2005)

Los encabezados de IP contienen la siguiente información:

- Versión: Especifica la versión de IP usada para generar el paquete.
- Internet Header Length: Especifica el tamaño del encabezado IP en palabras de 32 bits.
- Type of Service (ToS): Utilizado para proveer calidad de servicio.
- Total Length (TL): Tamaño total del paquete IP.
- Identificación: Contiene un valor de 16 bits común a todos los fragmentos de un mismo paquete. Se utiliza para que el receptor pueda reconstruir el dato sin mezclar datos de diferentes fragmentos. Se llena incluso cuando el mensaje no ha sido fragmentado, por si un router tiene que hacerlo más adelante.

- Banderas: Tres banderas de control, dos de las cuales se utilizan para manejar la fragmentación.
- Fragment Offset: Valor que especifica donde en el mensaje completo va ese fragmento en específico.
- Time To Live (TTL): Especifica cuantos nodos puede cruzar el paquete y ser considerado "vivo". Cada vez que atraviesa un nodo se decrementa este numero, y si llega a 0, se considera que tomo demasiado tiempo en ser transmitido y se descarta.
- Protocolo: Especifica que protocolo de alto nivel esta encapsulado en el dato.
- Header Checksum: Checksum para verificar que el encabezado esta correcto.
- Source Address: Dirección en 32 bits de la fuente.
- Destination Address: Dirección de 32 bits del receptor del paquete.
- Opciones y Padding: Son bits variables que definen que opciones se desean utilizar y el padding agrega 0 suficientes para hacer los bits de opciones un múltiplo de 32.
- Data: Los datos a ser transmitidos. Puede contener un mensaje entero o un fragmento de uno.(TCP/IP Guide, 2005)

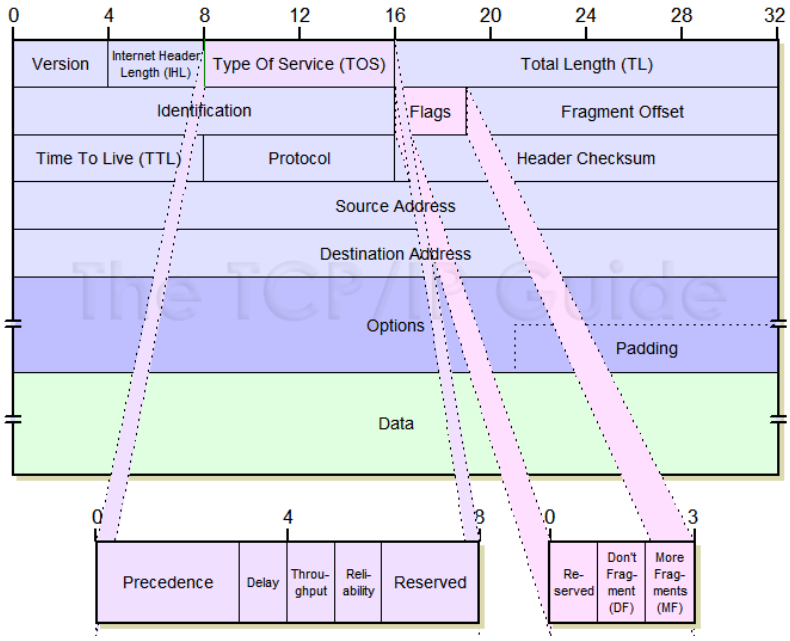


Figura 8. Datagrama IP (TCP/IP Guide, 2005)

6) Desventajas

Una de las desventajas de las redes TCP/IP es la recuperación automática de errores. Aunque esta provea muchas ventajas, tiene la desventaja que los fallos de equipo y conexiones pueden pasar mucho tiempo sin ser detectadas, ya que la red se adapta y continúa funcionando, causando un desbalance o exceso de tráfico por un nodo en especial. (Gilbert, 1995)

También tiene la desventaja que tiende a consumir mucho ancho de banda en ciertas aplicaciones, ya que los paquetes pueden llegar a ser bastante extensos, y causar lentitud o errores continuos al intentar transmitir. (Gilbert, 1995)

Z. Protocolo Serial RS232

El protocolo RS232 fue desarrollado por la Electronic Industries Association (EIA) en la década de 1960. Fue desarrollado como una interfaz estándar para aparatos de comunicaciones de datos, y diseñado para asegurar comunicación confiable entre dos aparatos, así como para asegurar interoperabilidad entre dispositivos de diferentes empresas. El protocolo especifica voltajes de las señales, sincronización, función de las diferentes señales, un protocolo para intercambio de información, e incluso los conectores mecánicos. (Strangio)

1) Funcionamiento

El protocolo RS232 envía bits en series temporizadas, los cuales son muestreados a intervalos determinados. El protocolo soporta transmisión de datos síncronos y asíncronos. El estándar define circuitos para control de flujo, y siendo el circuito de transmisión independiente del circuito de recepción, el protocolo se dice que es "full – dúplex", o que soporta transmisión en ambos sentidos simultáneamente. (Strangio)

2) Voltajes

El protocolo RS232 utiliza lógica negada bi-polar, o sea que un voltaje negativo representa un "1" lógico, y un voltaje positivo representa un "0" lógico. El protocolo también define una tierra común, que es la referencia con la cual se mide cualquier voltaje que se transmita, ya sean datos, señal de sincronía y señales de control. Además de esto, la tierra común se utiliza también para amarrar el blindaje electromagnético a tierra, y así disminuir interferencia. (Strangio)

Voltajes de -3V a -25V, con respecto a la tierra común se consideran como un "1" lógico, mientras voltajes de +3V a +25V se consideran un "0" lógico. El

espacio que resta entre +3V y -3V se considera espacio muerto, y el estado de la señal no esta asignado allí. En la práctica los niveles de voltaje más comunes son ±5V, ±10V, ±12V y ±15V. El voltaje que algún sistema utilice termina siendo dependiente de la fuente que el dispositivo contenga. Dado que los niveles de voltaje del estándar son más altos que los típicamente utilizados en circuitos lógicos, es necesario utilizar circuitos de interfaz, que cambian los niveles de voltaje entre los dos circuitos. (Strangio)

El protocolo RS232 es un protocolo no balanceado, lo que significa que sus voltajes son medidos contra una referencia general, mientras los protocolos balanceados son aquellos que el voltaje se determina de la diferencia de voltaje entre dos líneas, o sea no utiliza una tierra común. (Strangio)

3) Conectores Mecánicos y Cables

El estándar RS232 original llama al equipo de comunicación un DCE (Data Circuit – termintaing Equipment, usualmente un modem) y el equipo terminal DTE (Data Terminal Equipment). Ambos equipos tienen terminales DB25, la terminal DTE con uno macho, y la terminal DCE con uno hembra. Estos conectores utilizan 22 de los 25 pines disponibles para señales y tierra. El cable utilizado es de 25 conductores, y esta conectado en forma recta, sin cruces ni conexiones internas. A continuación se muestra una figura de la conexión entre un equipo DTE y uno DCE. (Adrio Communications Ltd, Radio-Electronics.Com)

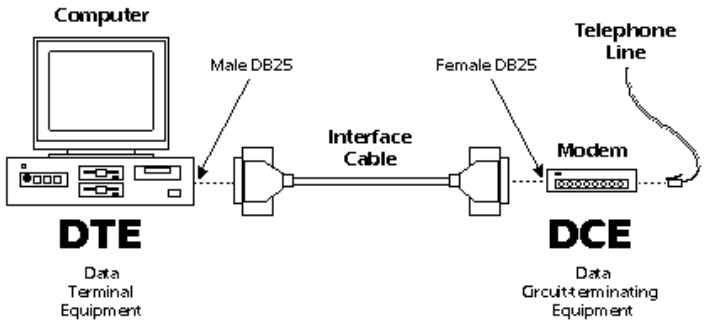


Figura 9. Diagrama de conexión entre equipo DTE y DCE (Strangio)

En la actualidad se han utilizado diferentes conectores para RS232, a parte del DB25. El más común de hoy en día es el DB9, que contiene 9 pines. Este se comenzó a utilizar con la introducción de las computadoras personales, ya que la miniaturización y la falta de necesidad de canales secundarios hicieron que algunos pines ya no fueran necesarios, y que se pudieran eliminar. (Adrio Communications Ltd, Radio-Electronics.Com)

Los cables utilizados para conexiones con RS232 se recomienda que sean de menos de 15 m, ya que después de esa distancia, la interferencia comienza a ser problema mayor. Los cables para RS232 no deben ser trenzados, ya que el protocolo no es balanceado. (Adrio Communications Ltd, Radio-Electronics.Com)

4) Funciones de Pines

El protocolo RS232 define las funciones de pines en los conectores DB25. Las funciones de los pines para conectores DB9 también fueron estandarizadas, para prevenir problemas de compatibilidad en el futuro (estándar TIA – 579). En esta sección se presentarán las funciones de los pines en el conector DB9, que es el conector de interés en el proyecto.

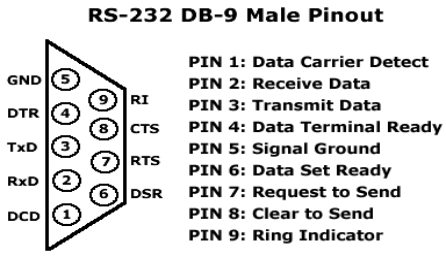


Figura 10. Diagrama de Pines de conector DB-9 (Strangio)

- Pin 1: Data Carrier Detect. Este pin tiene la función de alertar al equipo de cuando ya se estableció una conexión con el equipo remoto. Pertenece a las señales de control.
- Pin 2: Receive Data. Este pin tiene la función de recibir datos que el dispositivo DCE le envíe al dispositivo DTE. Pertenece a las señales del canal de comunicaciones primario.
- Pin 3: Transmit Data. Este pin se encuentra activo cuando se esta enviando datos desde la terminal DTE a la terminal DCE. Pertenece al canal de comunicaciones primario.
- Pin 4: Data Terminal Ready. Este pin se utiliza por la terminal DTE para indicarle a la terminal DCE que desea transmitir. Esta señal pertenece a las de control y estatus.
- Pin 5: Common Ground. Este pin es la tierra común del cable. Sirve para atar las tierras de los equipos, así tienen la misma referencia. También se une con el blindaje del cable para enviar a tierra cualquier interferencia.
- Pin 6: Data Set Ready. Se activa cuando sucede una de las siguientes:
- El DCE esta conectado a una línea telefónica “descolgada”.
- El DCE esta en modo de transmisión de datos; y
- El DCE ha terminado de marcar.
- Pertenece a las señales de control.

- Pin 7: Request To Send. Se activa cuando para que el DCE se prepare para recibir datos del DTE. Pertenece al canal de comunicaciones primario.
- Pin 8. Clear To Send. Se active cuando el DCE esta listo para recibir datos del DTE. Pertenece al canal de comunicaciones primario. El Request To Send y Clear To Send se utilizan para administrar el flujo de datos, y para que no se pierdan estos, o haya sobre envío.
- Pin 9: Ring Indicator: Cuando el DCE es un modem, indica que esta recibiendo una señal de timbre en la línea telefónica. Pertenece a las señales de control. (Strangio)

Existen otros pines que se utilizan en el protocolo RS232. Los aquí descritos son los necesarios para una transmisión asíncrona con control de flujo en hardware. Cuando se desea utilizar comunicación síncrona, o el canal secundario, es necesario usar el conector DB25 que tienen otras funciones en los pines. Dado que el enfoque de esta investigación es utilizar el protocolo asíncrono con conexión por DB9, esa parte ha sido obviada. (Strangio)

5) Velocidad de Transmisión

El estándar RS232 esta definido para velocidades de hasta 20,000 bits por segundo. Cuando el estándar fue escrito, se definió este como el límite, ya que después de esas velocidades el ruido comienza a ser un problema. En la teoría se puede alcanzar casi cualquier velocidad, pero se debe intentar mantener las distancias de cables cortas, así como asegurarse que el hardware que se utilice sea capaz de manejar transmisiones a esas velocidades. (Strangio)

6) Método de Transmisión

Los bits son transmitidos uno detrás de otro, por esto el nombre común de "puerto serial". Esto permite que solo un conductor sea necesario para transmitir y

otro solo para recibir, lo que habilita el full dúplex. El receptor y transmisor deben estar configurados de igual manera, estableciendo la velocidad de transmisión, cantidad de bits a transmitir, cantidad de stop bits, y si se incluye algún bit de paridad. (Strangio)

Las opciones de cantidad de bits a transmitir usualmente se limitan a 7 ó 8 bits. Los stop bits pueden ser comúnmente 1, 1.5 o 2. El bit de paridad se utiliza como un método de chequear la integridad de la información enviada. La paridad "par" dice que hay una cantidad par de bits en estado lógico 1, y la paridad "impar" nos dice que hay una cantidad impar de bits en estado lógico 1 en el dato a ser transmitido. Esta paridad se representa por 1 bit, y el tipo de paridad debe ser establecido antes en ambas terminales. (Strangio)

La transmisión en sí se inicia con un "start bit". Cuando la línea esta en reposo, su estado es un lógico 1. El start bit siempre es un lógico 0. Cuando el receptor recibe esto, se sincroniza y comienza a muestrear la línea para recibir la información enviada. Después de haber enviado el dato, se envía el bit de paridad, que es opcional, y al final se envían los stop bits. Los stop bits indican el fin de un byte en particular, y pueden ser ajustados desde 1 hasta 2. (Strangio)

El muestreo de los bits ocurre a la mitad de cada uno, y para que esto sea posible, tanto el transmisor como el receptor deben tener la misma velocidad configurada. Cierto grado de tolerancia es permitida en éste ámbito, pero este error de cualquier manera es mejor si se mantiene bajo, así se evita la perdida de información por errores de transmisión. (Strangio)

7) Desventajas de RS232

Dado que desde el desarrollo de RS232, la electrónica y las computadoras han avanzado a velocidad increíble, el estándar RS232 se ha ido adaptando a las

necesidades actuales. Hoy en día las interfaces seriales son parecidas a RS232 en sus voltajes, protocolos y conectores, siendo estas versiones simplificadas del estándar original. Dado que no se ha estandarizado la simplificación del protocolo, muchas empresas generan sus propias versiones ligeramente diferentes. Aunque estas tienen el mismo conector o mismos voltajes se presentan dificultades como la ausencia o escasez de líneas de control de flujo, uso incorrecto de funciones de ciertas líneas, causando conexiones equivocadas de líneas de transmisión y recepción, e incluso género del conector incorrecto, o uso de pines incorrectos que causan errores de conexión o transmisión. (Adrio Communications Ltd, Radio-Electronics.Com)

Otras de las limitaciones que se encuentra con este protocolo son:

- La necesidad de fuentes de alimentación bi-polares y con cambios de voltajes grandes, causando alto consumo de potencia así como limitaciones de velocidad causadas por la inhabilidad de producir los cambios de voltaje rápidamente.
- Transmisión de datos referenciada a tierra limita su inmunidad a ruidos y su distancia máxima de transmisión.
- No se ha definido la operación de RS232 en un ambiente que conecte más de dos dispositivos.
- Dado que funciones de cada lado de la transmisión esta asignado asimétricamente (funciones DTE vs funciones DCE), resulta difícil establecer las funciones de un equipo desarrollado recientemente.
- El flujo de control esta originalmente diseñado para establecer conexiones con circuitos de comunicaciones por teléfono, entonces resulta difícil utilizar estas líneas para establecer control de flujo en otros dispositivos.
- Aunque el estándar recomienda un conector y configuración de pines, el conector es de gran tamaño para estándares actuales.

- Pese a sus varias desventajas, el puerto serial sigue siendo utilizado por su facilidad de implementación, su comunicación eficaz en condiciones normales, y velocidades aceptables. Para comunicaciones de mayores velocidades (más de 1 Mbps), si se debería considerar algún otro protocolo para la comunicación serial.

V. DELIMITACIÓN DEL TEMA

A. Silla Operativa

1) Estructura Mecanica

El módulo incluye el diseñó y construcción de un sistema mecánico que acomoda al usuario, la interfaz humana que el usuario opera, la fuente de poder y el conjunto de circuitos para adquisición, procesamiento y transmisión de datos. La estructura del brazo operado por el usuario incluye la implementación de las piezas que sostienen los sensores en las correspondientes partes del brazo sin afectar el movimiento del mismo. La estructura incluye un lugar para el almacenamiento de los circuitos y contempla un espacio para el módulo de monitoreo.

Forma parte de este módulo soportar al usuario con una silla que le permita maniobrar sin dificultades de espacio y/o posiciones incómodas la operación del brazo. Además este módulo incluye el cableado que lleva la información de los sensores a los circuitos de procesamiento.

2) Brujula y joystick

Para el desarrollo del Megaproyecto “Sistema de Aprendizaje Remoto para el Robot R17”, se utilizaron sensores para determinar con precisión cual fue el movimiento realizado en la estructura mecánica y traducir este movimiento a coordenadas entendidas por el robot R17. Dichas señales sensadas están dadas en forma analógica, y la tarea del sub-módulo “Procesamiento de señales” consiste en digitalizar dichas señales.

El presente trabajo trata sobre la digitalización e implementación de uno de los sensores utilizados, la brújula electrónica. Dicho sensor dio las lecturas del movimiento rotacional sobre el plano XY realizado en la estructura mecánica.

Las señales enviadas por la brújula digital debieron ser convertidas del protocolo I2C a serial, y de serial a protocolo CAN, para lograr la transmisión hasta la computadora que tuvo la tarea de procesar las coordenadas y transmitirlas al R17.

También se implementó un Joystick que permitió superar la limitación física de la estructura mecánica, dando una movilidad de 360 grados a la unión conocida como "cintura". De la misma manera, el Joystick dio movimiento al sexto eje de libertad, la banda en la que desplaza el robot.

Dicha digitalización y conexión del joystick fue realizada por medio del microcontrolador 18F458, debido a su robusta arquitectura y soporte, tanto del protocolo I2C como del protocolo CAN.

3) Acelerometros

En el módulo de Silla Operativa Submódulo Sensores se colocó en la estructura mecánica sensores de aceleración que permiten obtener información necesaria para que el robot pueda reaccionar ante un estímulo del usuario.

Se seleccionó un sensor que pudiera cumplir con los requisitos impuestos por el brazo mecánico hecho por el submódulo de estructura, tales como resolución, tiempo de respuesta, etc., y que al mismo tiempo cumpliera con las recomendaciones de uso y aplicación del fabricante.

Por a su fácil instalación, bajo costo y eficacia se escogió como sensor de posición para las articulaciones de hombro y brazo al acelerómetro. Estos se

calibraron de tal manera que su información fuera una representación precisa de los movimientos de la silla.

Ya que estos son sensores con salidas analógicas, estas fueron muestreadas, con un reloj de conversión de 3Khz, por medio de un microcontrolador 18F458. Dentro del microprocesador se compara el nuevo dato muestreado con el último para determinar si hubo un cambio significativo en el estado del sensor; de ser así este dato se enviara de forma binaria a través de la red CAN, hecha por el módulo de Comunicaciones submódulo Red CAN.

Debido a que la información es enviada a través de ésta red sin ningún orden lógico (solo se enviaran los datos de los sensores que están cambiando) es necesario que cada trama que contenga los datos posea un identificador, este identificador se armó en conjunto con los Modulos de Brazo Robótico, Submódulo Procesamiento de Coordenadas y con el Módulo de Silla Operativa, Submódulo Monitoreo.

La señal digital tiene un tamaño de 2bytes por sensor; ya que los microcontroladores tienen una resolución de diez en el convertidor de analógico-digital, se puede llegar a tener una resolución de 11 bits por encoder, 10 bits de ángulo y uno de dirección, y se utilizan 3 bits de identificador. Esto da como resultado la inutilización de 2 bits, los cuales son usados por el submódulo de Monitoreo para funciones específicas de él.

4) Codificadores

El problema a solucionar es el siguiente; se requiere de un sensor rotacional que detecte el movimiento forzado por el usuario sobre las junturas de la mano y la muñeca en la estructura de mando. Como ya se ha mencionado en el resumen y la introducción anteriormente, este trabajo de graduación se limita a la instalación,

funcionamiento adecuado y procesamiento de las señales propagadas por los codificadores. Éstos están instalados en las junturas de la estructura que replican los movimientos del usuario. Los codificadores capturan el movimiento deseados para mover la mano o la muñeca de robot R17.

5) Monitoreo

El submódulo de monitoreo se limita a realizar una interfaz que permita el control del sistema desde la silla operativa, retroalimentando al usuario con información necesaria para saber acerca del correcto funcionamiento tanto de la comunicación hacia el robot, como de los límites físicos en los que el robot pueda encontrarse. Además, simplifica la tarea para que un único usuario pueda realizar la operación del sistema.

El desarrollo del submódulo se limita al control de una pantalla y del correcto despliegue de los menús en ella. Esto conlleva a realizar un diseño que permita el manejo apropiado de la pantalla por el usuario. Así también, la implementación de funciones de comunicación con el módulo CAN, con el que se realiza el intercambio de información que existe entre el módulo del Wiport, el programa de la PC y el módulo de monitoreo. La información recibida por el módulo es acomodada y desplegada en la pantalla cuando es solicitada.

Además, el submódulo monitoreo modifica el estado de mímica, grabación (programación) o alto al movimiento reproducido en el robot R17 por medio de la silla operativa. También, mediante el uso de un joystick se maneja la articulación del riel y la cintura, para el movimiento respecto a nuevos parámetros de referencia en la silla operativa, esto para las articulaciones del track y el waist particularmente.

Por último, se participa en el diseño de la silla y el acople del submódulo de monitoreo en la estructura física de la silla operativa, buscando dejar protegida la pantalla y el circuito de esta, acoplandola de una manera ergonómica para el usuario.

B. Modulo de Comunicaciones

1) Submodulo de red local

e) Transmision

La silla operativa para el robot R17 posee 8 nodos, los cuales deben poder comunicarse entre sí. Estos nodos no solo se comunican unos con otros sino también poseen cierto grado de prioridad en el momento de la transmisión de mensajes dentro de la red. Para satisfacer las necesidades del proyecto, se implementará un protocolo CAN. Por medio de la creación de una librería se creará una función con la cual los nodos podrán transmitir a través de la red CAN de manera rápida y sencilla. Dentro de la función de transmisión se resolverá el problema de las prioridades de los mensajes, de esta forma los nodos no tendrán que enfrentarse con problemas de prioridades ni identificadores. La red CAN implementada será creada para 8 nodos, sin embargo gracias a la flexibilidad de la librearía y de la función de transmisión, podrá ser fácilmente modificada en caso se requiera alterar el numero de nodos.

f) Recepcion

Es responsabilidad de esta fase filtrar los mensajes para que el nodo que así lo requiera, únicamente procese los mensajes que le fueron enviados directamente, ignorando el contenido de los mensajes que tienen como destinatario a otros nodos.

Esta fase, en conjunto con la fase de transmisión, busca lograr que todas las comunicaciones sean exitosas y que los clientes de la red únicamente utilicen una función sin tener que implementar alguna operación directa del protocolo CAN. Por lo anterior, este desarrollo se concentra en la creación de una librería de funciones independiente de cualquier programa, con lo que se hace aun más robusta y permite que en un futuro sea reutilizada en otras implementaciones.

En relación con el objetivo principal, este desarrollo no busca crear ningún protocolo de comunicación nuevo con especificaciones propias para el sistema de aprendizaje, sino que busca aprovechar el protocolo de las capas física y de enlace de datos del modelo ISO/OSI, para implementar las siguientes tres capas (red, transporte y sesión) y con eso permitir que los clientes únicamente tengan que trabajar en las capas de presentación y aplicación. Con lo anterior, este desarrollo permite que un protocolo robusto, CAN, sea implementado de manera sencilla en un proyecto global.

2) Submódulo de red WiFi

g) Transmision de datos

Este módulo se encarga de leer de la red CAN la información enviada por los módulos de sensores y procesamiento. El módulo contempla la formación de paquetes de los datos enviados por los módulos de procesamiento de sensores. Incluye la transmisión de toda la información por Wi-Fi hasta una PC. El módulo no incluye el procesamiento posterior luego de que la PC recibió la información, ni la recepción de información de la PC.

h) Recepción de datos

Este trabajo abarca el diseño de un circuito electrónico para poder utilizar el WiPort b/g, así como el desarrollo de software para que se pueda comunicar con demás módulos.

El trabajo inició con el desarrollo del circuito impreso, y su construcción en un protoboard. Después de esta fase se siguió con pruebas de funcionamiento y determinación de la velocidad más adecuada para que el sistema funcionara adecuadamente. Cuando se tenía identificada la velocidad adecuada de transmisión se comenzó el desarrollo de software necesario para que las comunicaciones entre módulos fueran adecuadas.

Durante el desarrollo se encontró que la mejor velocidad de transmisión es 57,600 bps.

C. Brazo Robótico

1) Control del robot

El presente trabajo abarca distintas áreas, entre ellas se encuentran

- El detalle del protocolo DeucaTalk. Para este trabajo se desarrollo un protocolo de comunicación entre la silla operativa y la aplicación, la explicación del mismo, y el lugar que le corresponde. Además es posible dar una con esto se da la explicación del manejo de la posición actual del brazo robótico.
- El manejo de límites de las distintas uniones del brazo robótico. En este aspecto el trabajo aborda la calibración de los límites, de manera que si en algún momento el brazo robótico esta llegando aun límite, la aplicación enviará un conjunto de acercamientos a límites con los identificadores de las

uniones, en el caso de que el brazo este en un límite este devolverá un conjunto de las uniones que ya no se seguirán moviendo.

- El presente trabajo se ocupa del área de comunicación con el robot desde una computadora, tomando en cuenta el protocolo que se definió con la silla operativa. Además de controlar el brazo robótico, es en esta parte del trabajo en la que se le da seguimiento a la posición actual del robot en cada una de las uniones del mismo, lo que permite poder llevar un control de los límites, suspendiendo la acción de un movimiento cuando esté fuera del área de funcionamiento, o bien pudiendo dar una retroalimentación a la silla operativa, específicamente a la sección de monitoreo.
- La tabla de instrucciones usadas para comunicarse con el brazo robótico.
- Explicación de algunos algoritmos importantes, entre ellos:
- La conversión de una medición a una coordenada.
- El proceso de carga de la información, así como el flujo del mismo.
- El proceso de guardar una rutina, así como la carga de la misma.
- Las modalidades de manual y automática.

2) Comunicación desde la PC

El módulo denominado "Brazo Robótico" se limita a la conexión entre la silla operativa y el robot R17 mediante una aplicación de escritorio que puede ser utilizada para enseñar las tareas a realizar al robot. Este módulo comprende tanto la comunicación entre la aplicación y el robot propiamente dicho como la comunicación entre la silla operativa y la aplicación y viceversa con el área de monitoreo.

El Submódulo de "Comunicación con Robot", derivado del módulo anterior, comprende y se limita a la transmisión de instrucciones codificadas en el lenguaje ROBOFORTH hacia el robot R17 por medio del puerto serial para su inmediata

ejecución, así como la recepción de los mensajes de diagnóstico enviados por el robot en cualquier momento de la fase de enseñanza o de ejecución. También cumple con la función de transmitir un bloque de código hacia la memoria interna del robot, para su ejecución independiente del software de manera repetida y según la programación recibida.

El submódulo de "Comunicación con Monitoreo", también derivado del mismo módulo, comprende el otro extremo de la comunicación, y se limita a la transmisión y recepción de mensajes entre la aplicación y la silla operativa. Dichos mensajes contienen la información de posición específica de la silla operativa, la cual deberá ser traducida por otro submódulo; de igual manera contienen información de diagnóstico que se obtiene del mismo submódulo que procesa los datos y que informa a la sección de monitoreo de la silla operativa sobre varios estados del robot en cualquier momento dado.

Esta parte de la investigación se limita al desarrollo de dos partes de la aplicación de escritorio: la capa de comunicación que se encarga de la transmisión de información entre la aplicación y la Silla Operativa, y la capa de comunicación que realiza en envío y recepción de datos con el robot R17. Es una aplicación que funciona específicamente bajo el sistema operativo Windows XP, se comunica mediante el protocolo TCP con la Silla Operativa, y lo hace exclusivamente a través de un puerto serial con el robot R17. El desarrollo se limita al análisis, diseño e implementación adecuada de dos clases, las cuales realizan todo lo referente a la comunicación de la aplicación, tanto con la Silla Operativa como con el robot R17.

VI. METODOLOGÍA

A. Silla Operativa

1) Estructura Mecanica

1. Al inicio se tomaron mediciones de los brazos de usuarios elegidos al azar para determinar las medidas de la estructura del brazo.

2. Luego se tomaron medidas del largo de las piernas de usuarios elegidos al azar para determinar la altura de la silla y la altura del brazo.
3. Se determinaron el resto de las medidas de la silla como ancho de la silla, altura y largo de la caja que almacena los circuitos, altura y largo del soporte que se usa para descansar el brazo izquierdo y posicionar el módulo de monitoreo.
4. Se determinaron las posiciones de los sensores en la estructura del brazo y se definieron los materiales a trabajar: lámina negra para la estructura de la silla, y aluminio para la estructura del brazo.
5. Se cortaron los tubos que forman el esqueleto de la silla y soporte del brazo izquierdo y las planchas para las caras de la caja que contiene los circuitos, y luego se soldaron todas las partes.
6. Se extrajo la parte ergonómica de una silla de oficina, se abrió un agujero a la caja que contiene los circuitos, se soldó un tubo al agujero y se acopló sobre la estructura una silla de oficina.
7. Se adquirieron 4 articulaciones de aluminio que permiten una rotación con fricción para asegurar que la estructura del brazo mantuviera rigidez al realizar movimientos, pero que fuera lo suficientemente flexible para rotar.
8. Se cortaron 2 tubos de aluminio para unir las articulaciones de la muñeca y mano, y las articulaciones de la mano y el codo. Se perforaron tanto los tubos como las articulaciones y se insertaron tornillos.

9. Se cortó un tubo de lámina negra y se le adaptó la articulación de la cintura. Al otro extremo de la articulación, se atornilló el resto de las articulaciones.
10. Se colocó la brújula sobre una pieza en forma de 'L', la cual se atornilló a la articulación de la cintura. Se perforaron 2 agujeros en la articulación del codo y de la cintura para colocar los acelerómetros con tornillos. Se performaron 2 agujeros en la mano y en la muñeca, sobre los cuales se atornilló una pieza en forma de 'L' para colocar los codificadores.

Herramientas

Programas

- AutoCAD: para diseño en 2D y 3D

Equipo

- Equipo de soldar: electrodo, fuente de corriente y lentes de protección.
- Manejo de metales: dobladora y cortadora de planchas, pulidora, sierra, lápiz metálico y metro.
- Equipo para acoplar sensores: dremel, barreno, brocas de 2 mm diámetro, tornillos de 3mm diámetro, desarmador.
- Engranajes: máquina de manufactura flexible (FMS), tornillos de cabeza hexagonal y llaves Allen.
- Protección contra corrosión: pintura anticorrosiva y brocha.

2) Brujula y joystick

1. Investigación sobre los tipos de sensores disponibles, para determinar el más adecuado para este proyecto; así como una investigación sobre el protocolo I2C.

2. Creación de rutinas de calibración del sensor y de lectura por medio del protocolo I2C, las cuales permitieron la reducción de errores y obtención de los datos a ser transmitidos, correspondientemente

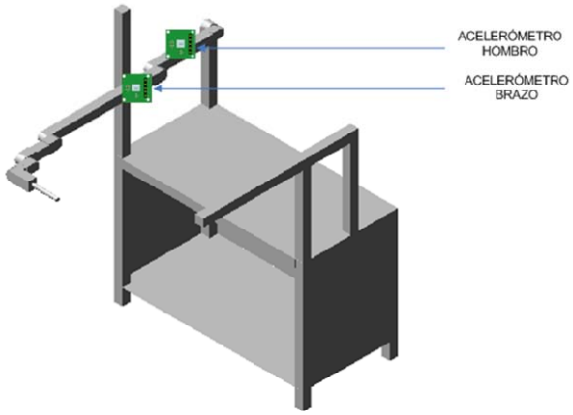
3. Caracterización del sensor en términos de exactitud y precisión, lo que permitió cuantificar el error en las lecturas realizadas.
4. Creación e implementación de una ecuación de regresión que permitió disminuir el error en las lecturas.
5. Creación de tramas de transmisión con 2 bytes de resolución.
6. Creación de lectura y de las tramas de transmisión del joystick.
7. Incorporación de la rutina CAN para la transmisión de los datos recolectados por el sensor y el joystick.
8. Optimización del sistema una vez incorporado a la estructura mecánica.

Herramientas

- Multisim 10: Software desarrollado por "National Instruments" que permite la simulación de circuitos electrónicos. Con este software se realizó un diagrama del sistema implementado.
- TraxMaker: Software desarrollado por "Protel International", fue utilizado para realizar la plantilla del circuito impreso del sistema.
- MikroC: Software desarrollado por la empresa serbia "Mikroelektronika", este software comprende un compilador en C para uso con microcontroladores. Este software permitió la implementación de las rutinas necesarias para cumplir con los objetivos del presente trabajo.

3) Acelerometros

Instalación de Sensores



Gráfica No. 19. Instalación de Sensores en Silla

La instalación del acelerómetro y la preparación superficial pueden afectar la respuesta amplitud-frecuencia de la medida, particularmente en los casos de alta frecuencia. Se debe tener mucho cuidado para asegurar un acoplamiento correcto con una superficie lisa y plana.

Pruebas de Funcionamiento

Se hicieron pruebas de funcionamiento tratando de simular lo mas cercanamente posible las condiciones de funcionamiento del acelerómetro en el brazo. Se midieron los voltajes de salida y rapidez de reacción.

Captura de Datos Sensores

El dispositivo analógico ADXL321 es un acelerómetro de 2 ejes bastante simple que puede conectarse fácilmente a microcontroladores. Este dispositivo

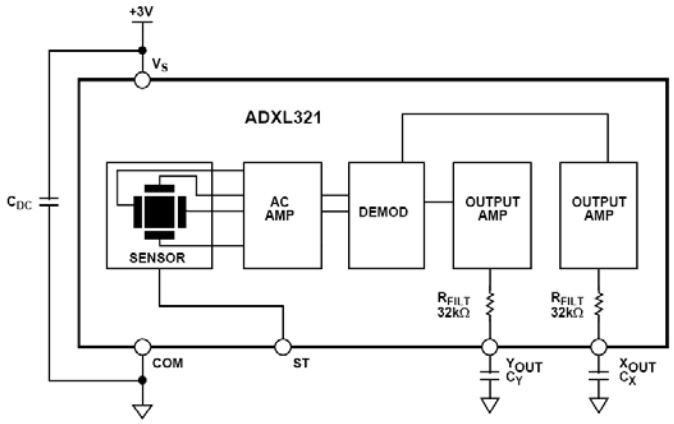
es capaz de leer aceleraciones estáticas y dinámicas con un rango de medidas que oscilan entre ±18 *g*. La resolución depende del ancho de banda, y es de 60mm/s2 (5 m*g*) a 60 Hz(Evaluation Board).

Creación de Paquetes para enviar en la Red

Se crearon los paquetes de información que serán interpretados por el Módulo de brazo robótico para poder reproducir los movimientos del usuario

Herramientas de Trabajo

ADXL321



Gráfica No. 20.Diagrama de Bloques ADXL321

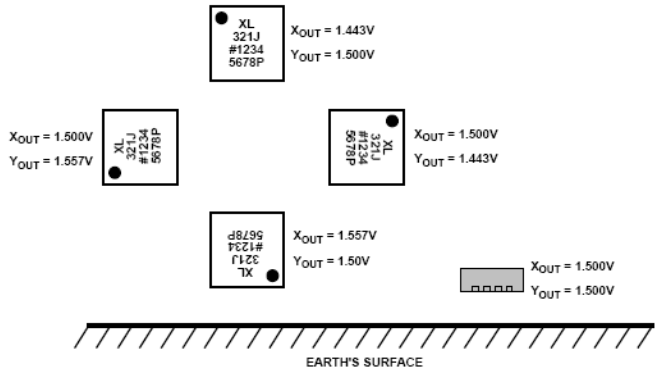
El ADXL321 es un sistema completo de medida de la aceleración. Este tiene una gama de medida de ±18 g. El acelerómetro mide fuerzas estáticas de la aceleración, tales como gravedad, que permite que sea utilizado como sensor de inclinación. Su salida son voltajes analógicos que son proporcionales a la aceleración a la que es sometido.

El sensor es una estructura con superficie-micromaquinada de polysilicon construida encima de una placa de silicio. Los resortes de Polysilicon suspenden la estructura sobre la superficie de la placa y proporcionan una resistencia contra fuerzas de la aceleración. La desviación de la estructura se mide usando un condensador diferenciado que consiste de placas fijas independientes y placas unidas a la masa móvil. Las placas fijas son alimentadas con ondas cuadradas desfasadas 180°. La aceleración desvía la viga y desequilibra el condensador diferenciado, dando por resultado una onda cuadrada de amplitud proporcional a la aceleración.

	Parámetro	Condiciones	Min	Típico	Max	Unidad
Entrada Sensor	Rango	Cada eje		±18		g
	No Linealidad			±0.2%		%
	Error de Alineación de Paquete			±1		Grados
	Error de Alineación	Sensor X a Y		±0.1		Grados
	Sensibilidad de Ejes			±2		%
Sensibilidad	Sensibilidad a X_{OUT},Y_{OUT}		51	57	63	mV/g
	Sensibilidad a Temperatura			0.01		%/°C
Respuesta 0g	Voltaje X_{OUT},Y_{OUT} a 0g		1.4	1.5	1.6	V
	0g Offset vs. Temperatura			±2		mg/°C
	Ruido			320		µg/vHz
	Cx,Cy Rango		0.002		10	µF
	Tolerancia R_{FILT}			32±15%		kΩ
	Frecuencia de Resonancia			5.5		kHz
Respuesta Self-Test	Entrada Low			0.6		V
	Entrada High			2.4		V
	Resistencia Self-Test a Tierra			50		kΩ
	Cambio X_{OUT},Y_{OUT}			18		mV/g
	Rango Temperatura		-20		70	°C

Especificaciones ADXL321

La salida del ADXL 321, cuando este esta alimentado por 3V, esta dada por el siguiente diagrama:



Gráfica No. 21. Salidas ADXL321

4) Codificadores

Investigación

Se realizó una investigación de parte del módulo de sensores y de procesamiento de datos previa a la adquisición de los materiales a utilizar. En esta fase se determinaron las marcas y los modelos que estaban dentro de nuestro alance económico y que poseían las características suficientes y necesarias para la detección precisa de un movimiento generado en la silla. Se analizaron las frecuencias de muestreo de dichos sensores y su respectiva resolución. Se analizaron los codificadores rotacionales absolutos como la primera opción para representar los movimientos de la mano y la muñeca, respectivamente.

Herramientas de Trabajo

Para llevar a cabo la instalación de los codificadores sobre la estructura fue necesario adquirir diferentes tipos de materiales, entre ellos podemos listar: plástico Vekton®, fresas HSS y de Tugnsteno de Titanio, tornillos y ¨L´s¨

metálicas de aluminio de 4 mm de largo, alto y profundidad. Las fresas y el plástico que se emplearon para la fabricación de engranajes, y las ¨L´s¨ para la instalación de los codificadores. Adicionalmente se evaluó la posibilidad de utilizar el sistema FMS del departamento de mecánica de la UVG, para evitar costo de fabricación de los engranajes y poder monitorear más de cerca el proceso de su fabricación.

Para el procesamiento de datos se utilizó el micro controlador de Microchip 18F458. Se implementó éste micro controlador ya que posee un módulo de transmisión y recepción del protocolo CAN, el cual se conecta en los pines. Dicho protocolo se implementará para la red de comunicaciones local. Adicionalmente, se implementó el módulo convertidor analógico-digital y el módulo comparador de tensión.

Procesamiento de Datos

Se implementó la programación de un microcontrolador 18F458 para la lectura de los codificadores y el procesamiento adecuado de la trama que se desea enviar. Se analizó que se debe de emplear un microcontrolador para el procesamiento de cada codificador, y un oscilador de alta velocidad de 40 MHZ.

5) Monitoreo

Con la disponibilidad del robot Deucalion R17, en el cual se basa el desarrollo del sistema de aprendizaje del megaproyecto, se da la necesidad de buscar una interfaz para tener una forma fácil de controlar el robot y de conocer el estado actualizado del sistema cuando este se utilice.

Por lo tanto, la primera parte consistió en realizar una investigación acerca de las distintas interfaces que se pueden implementar para el sistema y que éstas

sean adecuadas para que el operador tenga una visualización clara, y que sean económicamente viables.

Con la pantalla seleccionada, se buscó un microcontrolador adecuado para manejar las tareas del submódulo. Se utilizó un compilador en el lenguaje C para la programación, y se buscó la forma de controlar la interfaz, logrando desplegar información de monitoreo del sistema, de una forma fácil de entender.

Se buscó la forma de colocarlo en la silla operativa de modo que el submódulo permita una forma fácil de interactuar, dado que la investigación de la interfaz adecuada también incluyó la mejor forma de navegar e ingresar los datos que se quieren desplegar.

Mediante la comunicación implementada por los otros módulos, se solicita la información que sea necesaria para el usuario, con lo cual se logra informarle del correcto funcionamiento del sistema y permitir mandar parámetros del control de grabación para simplificar el sistema, y lograr que un solo operario maneje el sistema.

Programación del módulo

Habiendo realizado una investigación que incluyó cotizaciones y presentando varias opciones para la implementación del módulo, se definieron las siguientes actividades generales para ir desarrollando el submódulo:

1. Primero, la adquisición de materiales, consistiendo en la selección mediante el consenso de todo el grupo, de la pantalla y la forma de ingreso (botones, teclado, etc.) basado en el presupuesto.
2. Luego de tener seleccionada la pantalla, se define la forma en que se desea visualizar la información y las diferentes acciones que se desea

que el usuario pueda realizar con la pantalla, tomando en cuenta las restricciones que se tendrán del hardware.

3. Teniendo esto, se hace un mapeo de las salidas del microcontrolador a todo el equipo especificado, con lo que se generan las librerías para el control tanto de la pantalla y los botones de esta.
4. En esta etapa, ya se puede proceder a la implementación de los menús previamente definidos y su optimización basado en una visualización real en la pantalla. Se finaliza con la implementación del módulo completo en la silla operativa y su acople buscando proteger el circuito y que tenga una posición ergonómica para el usuario.

En cada tarea del submódulo, se busca que la programación sea de manera modular, de tal manera que el cambio de cualquier componente de hardware, tenga bajo impacto en el resto del código del programa.

Diseño del circuito eléctrico

Mapeo de puertos del microcontrolador

La parte del circuito eléctrico comienza por el mapeo de los puertos del PIC. Primero se empieza respetando los pines de comunicación CAN. Con estos pines se realizará la comunicación de la silla operativa hasta el Wiport.

Acople a la silla operativa

Con el circuito soldado y probado, se procede a su acoplamiento a la silla operativa mediante una caja plástica, en la cual solo quedará visible la pantalla para su protección. Para manejarla, quedarán expuestos solo los pulsadores y se sellarán las aberturas dejando todo el circuito dentro de la caja, para protegerlo de polvo o cualquier suciedad.

Herramientas de Trabajo

Las herramientas a utilizar para el desarrollo de este submódulo son:

- Microcontrolador (Microchip PIC 18F)
- Mikroelektronika EasyPic3 (Quemador PIC)
- Software CCS C compiler (Programador Lenguaje C del Pic)
- Software MikroC compiler (Extractor de Bitmap – Bytes)
- Display (Menú y status de la silla)
- Integrados Varios para control Display
- Microcontrolador, Familia PIC 18F

Se utilizará el Microchip PIC de la familia 18F, en específico el PIC 18F458. Es un microcontrolador de arquitectura RISC de un máximo de 10 MIPS. Posee módulos internos con 4 distintos temporizadores, módulo de comunicación serial asíncrona y síncrona y un módulo de comunicación CAN. Posee una memoria ROM, tipo flash de 32,768 instrucciones, memoria SRAM de 3328 bytes y una EEPROM de 1024 bytes. En caso de que la memoria ROM y RAM no sean suficientes, se cuenta con el PIC 18F4680 con las mismas características, pero el doble de memoria. Ambos son microcontroladores fáciles de controlar con un compilador en código en C++ y cumplen con ser un microcontrolador con suficientes entradas y salidas para el manejo del display paralelo de 14 pines, el los botones. Además de que incluyen un módulo de comunicación comunicación CAN requerida.

CCS C compiler es una herramienta necesaria para poder programar en el código C, el cual fue aprendido en los inicios de la carrera, y así poder desarrollar funciones más poderosas en menos tiempo. Además, se usarán y modificarán las librerías predefinidas que trae para el manejo de pantallas paralelas. Se utilizará para la creación de las librerías de la parte de monitoreo

en sí, como para la exportación de las librerías de otros módulos, como lo es la parte de comunicación por CAN.

En el anexo 2 se podrá revisar la librería de la pantalla gráfica de LCD (GLCD) de interfaz paralela, que está basada en la librería abierta del compilador CCS C Compiler para controladores de pantalla KS0180 y las modificaciones a los puertos con los que el PIC se comunica con ésta, así como también está disponible la librería para la pantalla uOled con interfaz serial que no fue implementada, pero es funcional para proyectos futuros.

EasyPic3, es el programador que graba el código programado en la computadora en el microcontrolador que se va a utilizar. Este, además de soportar una variedad de microcontroladores, permite realizar pruebas en el mismo módulo en que se inserta el microcontrolador para programarlo.

MikroC compiler, se utilizará la versión gratis que permite exportar bitmaps monocromos, un archivo en C de arreglos de bytes para la ayuda de creación de algunos menús o inserción de logos como el de la UVG.

B. Modulo de Comunicaciones

1) Red Local

La red desarrollada por el submódulo es la que comunica todos los nodos involucrados en la silla operativa; para conseguir una comunicación correcta se realizaron las siguientes tareas:

1. Estudio de la interacción del submódulo dentro del sistema de aprendizaje remoto.

2. Estudio del protocolo de comunicación

3. Diseño de la arquitectura de la red.
4. Diseño del funcionamiento de un nodo genérico.
5. Adquisición de los materiales para la implementación de la red.
6. Utilización básica del modulo CAN de los microcontroladores.
7. Implementación de una red de 3 nodos.
8. Creación de rutina para recepción.
9. Implementación del chequeo de identificadores basados en la prioridad de los mensajes.
10. Creación de la librería CAN4R17 y pruebas de chequeo de identificadores
11. Implementación de la librería en los nodos de la red.

Herramientas de trabajo

A continuación se describen las herramientas utilizadas para el desarrollo del presente trabajo de investigación.

1. Microcontroladores: Se seleccionó el microcontrolador PIC18F458 de Microchip ya que incluye un modulo CAN en su arquitectura.

2. Transceivers: Se seleccionó el adaptador MCU2551 de Microchip, al igual que los microcontroladores, para adaptar las señales de comunicación al bus de datos.
3. Compilador: Para programar los microcontroladores se utilizó el software PIC-C de Custom Computer Services ya que el departamento de Ingeniería Electrónica ofrece la licencia del mismo para sus estudiantes.

2) Comunicación WiFi

Transmision de datos

1. Se alimentó el WiPort y se reconoció a través de la tarjeta de red inalámbrica de una PC.

2. Se hicieron pruebas de transmisión de datos al WiPort haciendo eco con a los datos recibidos desde la PC a 9600 bits por segundo. El eco consistió en unir la línea de transmisión y de recepción del WiPort.
3. Se diseñó en Multisim el circuito que conectaba el WiPort con los PIC, y se contempló indicadores de estado, los osciladores y el regulador de 3.3V adicional.
4. Luego del diseño, se construyó el circuito en protoboard y se conectó el PIC18F458. Se programó el PIC para hacer eco a los datos que recibía por serial. Se hicieron 3 pruebas con velocidades de transmisión de: 38400 bits/s, 57600 bits/s y 115,200 bits/s.
5. Por último, se programó el PIC en modo de transmisión únicamente. En este caso el WiPort no recibía datos de la PC sino solamente envió datos que recibió del PIC. Se utilizó un programa que leía los datos de la tarjeta de red inalámbrica de la PC para corroborar los datos enviados por el PIC.

Herramientas

- Hyperterminal de Windows XP version 5.1
- Compilador PIC en lenguaje C: MikroC version 7.0 y PIC C versión 3.0
- Programa monitoreo TCP/IP: Hércules versión 3.0.2
- Diseño circuitos: Multisim versión 10.0
- Programador de PIC: PicFlash versión 2.0
- Circuito: protoboard, multímetro, fuente de poder y osciloscopio.

Recepcion de datos

La metodología que se siguio para desarrollar el trabajo fue la siguiente:

- Diseñar circuito para conexión del WiPort al micro-controlador PIC.
- Realizar pruebas de conexión y transmisión del WiPort hacia una computadora.
- Aplicar todas las configuraciones necesarias para el funcionamiento correcto.
- Desarrollar el software necesario para la comunicación entre el PIC y el WiPort.
- Realizar pruebas para asegurar la comunicación básica entre el PIC y el WiPort.
- Incorporar la librería CAN para poder conectar otros nodos, y hacer pruebas de transmisión y recepción.

Herramientas

- Para la construcción del circuito y el desarrollo del software se utilizarán las siguientes herramientas:
- National Instruments Multisim 10: este es un software desarrollado por National Instruments que se utiliza para la captura y simulación de circuitos electrónicos. En este programa se desarrollará el circuito, para después proceder a enrutarlo en una placa de cobre y fresarlo.
- CCS Inc. PCH: este es un compilador en C para micro controladores de la familia PIC18. Este nos permite escribir el código en ANSI C, y después el compilador se encarga de convertirlo en lenguaje de máquina, que se utiliza para programar el micro controlador.
- Programador PICFlash 2.0: programador de PIC's de la empresa mikroElektronika.
- HyperTerminal: programa que permite establecer conexiones TCP/IP y seriales.

C. Brazo Robotico

Control del robot

1. Para realizar la comunicación con el Robot Deucalion R 17, se desarrolló una fase de investigación en la que se realizaron distintas pruebas con el lenguaje de programación Roboforth, mediante el programa RobWin2 versión 6.3. En de las pruebas que se realizaron, se identificó la funcionalidad de las instrucciones para el manejo del robot, entre estas se encuentran las de movimiento de las articulaciones, y la calibración del robot.

2. Se investigó además, las instrucciones, y se estudió cómo podían utilizarse sin la necesidad del uso de Robowin2. Esta investigación se llevo a cabo filtrando el puerto serial mientras se hacían las pruebas de la funcionalidad. Se llegó a determinar por medio de esta investigación que Robowin2 envía al controlador instrucciones completas que el controlador interpreta y lo convierte a lenguaje que el brazo robótico puede utilizar como instrucciones.
3. En la siguiente etapa, se estudió la forma de poder enviarle instrucciones al robot sin necesidad del uso de RobWin2, es decir, desde una aplicación diseñada y creada para resolver el problema. Se escogió un lenguaje de programación para realizar esta tarea siendo este C#.
4. Adaptación de estas instrucciones, a una aplicación para el manejo del robot, que pueda tener una entrada, y en general, que se pueda editar una rutina deseada sin tener necesidad de reprogramar la rutina, es decir, sin tocar el código generado.
5. Dentro de las características que ésta aplicación tiene contemplado, esta el manejo del robot tomando en cuenta las instrucciones enviadas a él por medio de WiFi, y recopiladas por los distintos codificadores en la silla operativa.

6. Se realizaron pruebas con la estructura propuesta a fin de establecer la correcta comunicación de los componentes del sistema.
7. Se estudió la necesidad de una retroalimentación en la cual la aplicación proporciona información al operador que se encuentre en la silla operativa, como por ejemplo si existe riesgo de llegar a algún límite, o que la última operación esta siendo ejecutada, etc.
8. Se desarrolló la forma de comunicación para ambos sentidos, es decir, se propuso y aprobó el protocolo "DeucaTalk", entre el módulo de la silla operativa, específicamente entre el sub módulo de monitoreo, y el sub módulo de comunicación, específicamente con la comunicación por medio de CAN.
9. Como parte de esta aplicación, se pretende crear una modalidad en la cual se pueda ir grabando los movimientos realizados para que posteriormente se puedan repetir las secuencias. También está contemplada la implementación con la cual se pueda observar claramente todo lo que está ejecutando la aplicación y su respectiva retroalimentación.

Comunicación desde la PC

1. Los requerimientos necesarios para el desarrollo de la capa de comunicación con el robot R17, fueron definidos luego de un análisis del sistema a construir y una observación de las características esenciales para esta capa.

2. Se diseñó la estructura básica de la capa de comunicación tomando como base la especificación de requerimientos definida anteriormente con el propósito de determinar una arquitectura eficiente para la comunicación.
3. Fue desarrollado el diseño anterior de manera completa codificando la arquitectura de manera óptima y eficiente, utilizando como herramienta la plataforma .NET de Microsoft y dejando como resultado la primera versión de la aplicación de escritorio.

4. Se realizaron las pruebas con la aplicación en su primera versión y el robot R17, tomando nota de las fallas y corrigiendo los errores menores que surgieron durante las pruebas para evitar los mismos problemas y su corrección en las versiones siguientes.
5. El diseño fue evaluado nuevamente y se realizaron los cambios necesarios para corregir los fallos anotados en la fase previa, para luego repetir las pruebas y tomar nota de nuevos fallos. Este proceso se repitió hasta que la aplicación resultante cumpliera su función de manera satisfactoria.
6. De igual manera que en la capa de comunicación con el robot, para la comunicación con monitoreo fue necesario realizar la obtención formal de requerimientos a partir de un análisis del sistema a construir y la observación detenida de características deseables para esta capa específica de comunicación.
7. Fue diseñada la capa de comunicación con monitoreo según los requerimientos tomados con anterioridad y observando la arquitectura más eficiente, en este caso utilizando tecnologías inalámbricas o WiFi.
8. Con base en el diseño anterior, se desarrolló la parte de la aplicación correspondiente a la capa de comunicación con monitoreo utilizando la plataforma .NET como herramienta de desarrollo, dejando como resultado la segunda versión preliminar de la aplicación de escritorio.
9. Se implementó la aplicación y se realizaron las pruebas correspondientes para corregir los errores presentes en la arquitectura y el desarrollo.
10. El diseño fue evaluado nuevamente tomando como base los resultados de las pruebas anteriores, y el proceso se repitió hasta lograr un funcionamiento adecuado de esta parte de la aplicación.

Herramientas de Trabajo

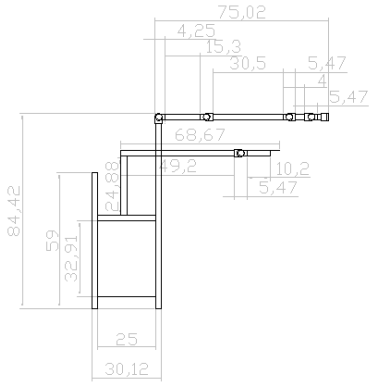
- La principal herramienta de trabajo es el robot R17, utilizado para realizar todas las pruebas y para descubrir la forma correcta de comunicación con el Submódulo de Comunicación con Robot.
- Ambiente de desarrollo integrado Microsoft Visual Studio 2005 Express Edition, sobre el sistema operativo Windows XP Professional, utilizado para desarrollar la aplicación de escritorio en su totalidad.
- Lenguaje de programación C#, con el cual se desarrolló toda la aplicación.

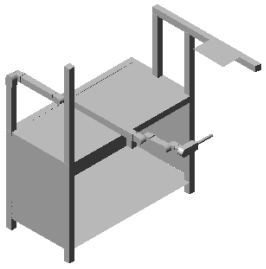
VII. DESARROLLO

A. Silla operativa

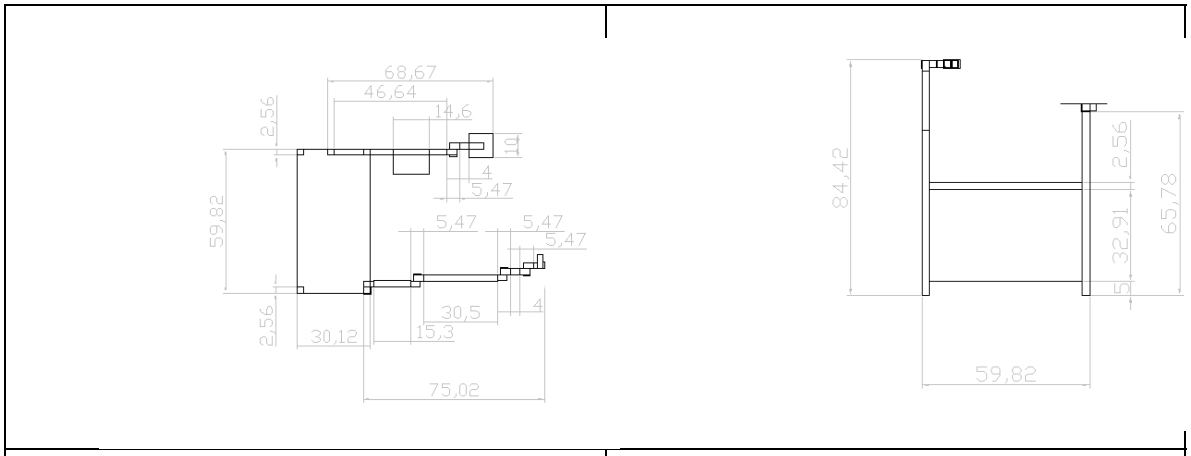
3) Estructura Mecanica

La construcción de la estructura mecánica requirió en una primera fase de diseño. Se definieron las medidas de la silla, las medidas y posición del brazo, y la ubicación del usuario en la silla. Se tomaron mediciones de varias extremidades humanas para que el usuario final se acomodara a la estructura. Luego se dibujaron los planos de las vistas de perfil, frontal y vista desde arriba que se muestran en la figura 19. Estas vistas se dibujaron en AutoCAD.





Vista perfil	Vista 3D



Vista planta | Vista frontal

Gráfico 1. Vistas en segunda y tercera dimensión de la silla operativa

Así mismo, se diseñaron 2 engranajes de diferentes diámetros con una extensión del programa AutoCAD que pertenece al centro de manufactura flexible (FMS). Esta extensión de AutoCAD contenía medidas predeterminadas para fabricar engranajes, que son los que ayudan a los codificadores a medir la rotación de las articulaciones.

Para obtener una idea general de la vista final de la estructura, se diseñó la silla en 3D usando AutoCAD. Esto permitió tener una vista que se asemeja a la forma final de la silla, lo cual ayudó a optimizar el diseño previo a la construcción de la misma. Las gráficas 20 y 21 muestran 2 de las 4 vistas isométricas que se obtuvieron del diseño en tercera dimensión. Las demás vistas isométricas pueden consultarse en el apéndice.

Se definieron los materiales con los que se construyó cada parte y se adquirieron los que aparecen en la gráfica 22. Se decidió construir la silla que acomoda al operario con lámina negra porque su costo es menor. Se compraron planchas, tubos cuadrados y redondos de lámina negra para las diferentes partes de la silla. Para la

parte de la interfaz humana, es decir, el brazo, se utilizó aluminio dado que es menos denso y el operario puede levantarlo con menos dificultad que otros metales como la lámina negra. Para unir las diferentes partes del brazo, se adquirieron las articulaciones que se muestran en la gráfica 23. Estas articulaciones presentan fricción al rotar, de manera que al dejar el brazo inmóvil, las articulaciones mantienen cierta fijación.

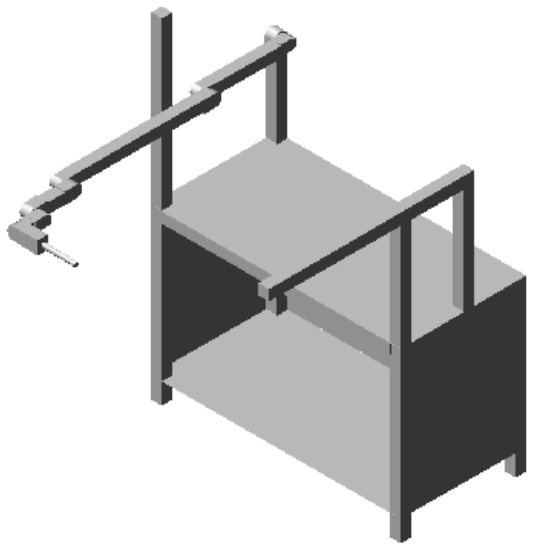


Gráfico 2. Vista Isométrica Sur Este

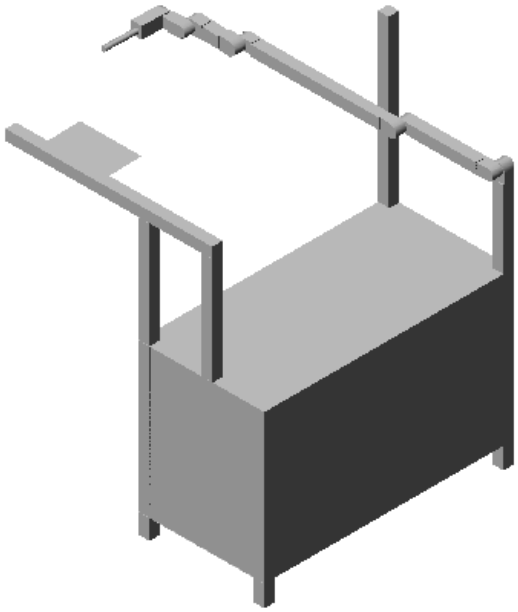


Gráfico 3. Vista Isométrica Norte Este

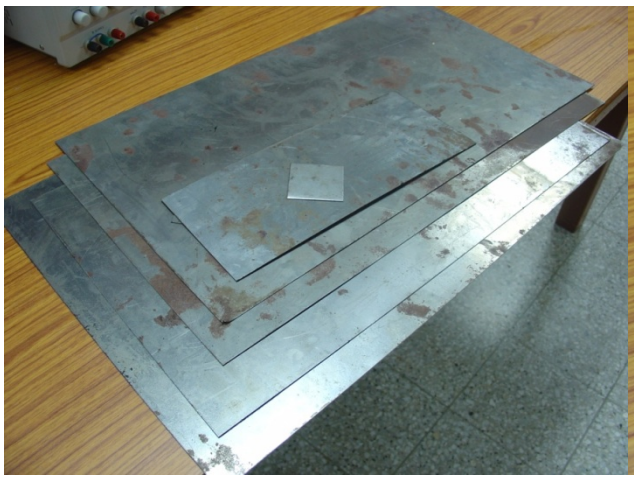


Gráfico 4. Materiales usados

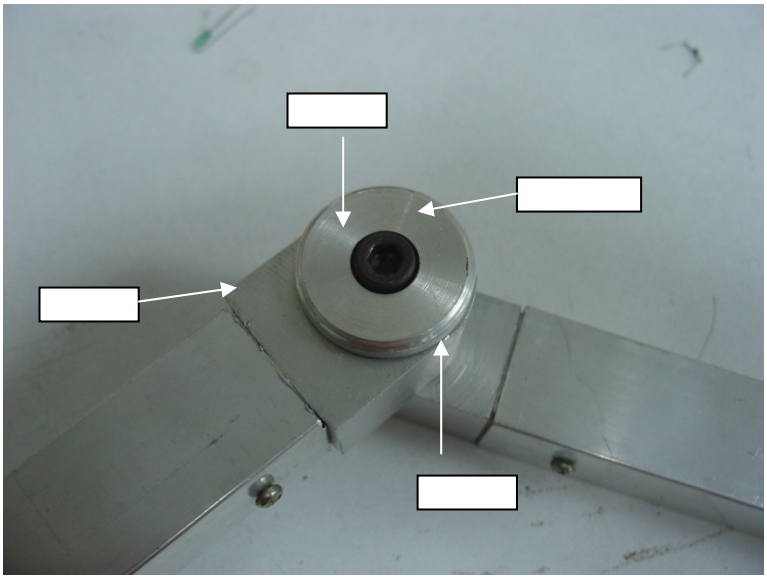


Gráfico 5. Articulaciones de aluminio

El mecanismo de la gráfica 23 consta de 2 piezas que rotan respecto del eje central definido por el tornillo. Dentro de cada una de las pieza 1 hay un espacio en donde se coloca un resorte, el cual tiene una de las caras transversales haciendo presión sobre la tapadera y la otra cara transversal ejerce presión sobre la pieza 2. De esta forma se mantiene cierta rigidez al rotar las piezas.

La segunda fase consistió en llevar a cabo el diseño. Primero se cortaron las planchas de metal, así como los tubos, de acuerdo a las medidas del diseño. Luego se llevaron a un taller de soldadura donde se unieron las piezas para formar la estructura de la silla. Primero se soldaron los tubos que conforman el esqueleto de la silla, luego se soldaron las planchas para formar el cajón inferior donde se

guardan los circuitos, y por último se soldaron las piezas que sirven de apoyo para el brazo izquierdo. Se abrió un agujero del lado superior del cajón que guarda los circuitos para soldarle un tubo metálico que sostiene la silla ergonómica de oficina.

Luego de construir la silla, se armó la interfaz humana o brazo de la silla operativa. Para construir la interfaz, se compró un tubo cuadrado de 1 pulg. de aluminio. Se cortaron 4 piezas que se unieron a través de las articulaciones. Se barrenaron todas las intersecciones entre el tubo de aluminio y las articulaciones, usando una broca de 2.5mm. Luego se insertaron los tornillos para asegurar la estructura, usando tornillo de 3 mm. Luego se perforó las partes inferiores del brazo y antebrazo para atornillar los acelerómetros que ya contenían agujeros para el ensamblaje. Para colocar los codificadores y la brújula, se acopló en los respectivos lugares unas piezas metálicas en forma de 'L'. Sobre estas piezas se montaron los sensores. Las piezas se atornillaron al brazo.

Los engranajes diseñados en AutoCAD se fabricaron con la máquina FMS. Estos dos engranajes se acoplaron al codificador y al centro de la articulación. En la gráfica 24 se observan ambos engranajes. El engranaje de diámetro mayor rota respecto de la pieza 1 dado que está sujeto por el centro, a través del tornillo, con esta pieza. El engranaje de menor diámetro está fijo al codificador, el cual a su vez está fijo a la pieza 2. Por lo tanto, al rotar la articulación, rotan los engranajes uno respecto del otro permitiendo medir el movimiento.

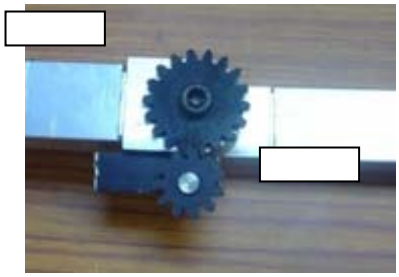


Gráfico 6 - Engranajes fabricados con FMS

Como resultado se obtuvo la estructura de la silla que se observa en la gráfica 25. Se aprecia la estructura de la silla, la silla ergonómica y la estructura del brazo. En el apéndice se encuentra el resto de las vistas de la silla operativa.

Gráfico 7 - Silla Operativa

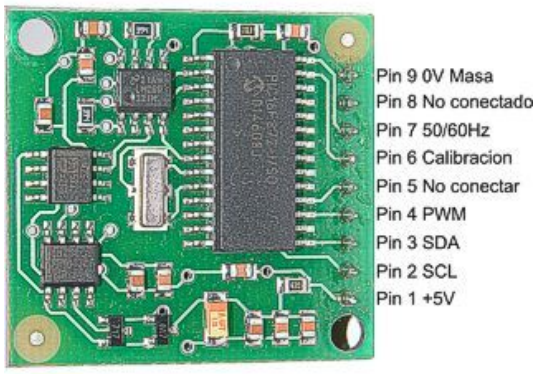
Brujula

Tras investigar los tipos de sensores para medir movimientos circulares, se determinó que la mejor opción se encontraba dentro de la gama de sensores magnéticos, ya que estos permitían una más fácil implementación a la estructura mecánica que los codificadores, y daban la libertad de mover la pieza rotatoria en el eje Z, haciendo el brazo ajustable.

Dentro de los sensores magnéticos la brújula digital CMPS03 S320160 proporciona la relación costo-beneficio adecuada para este proyecto, ya que su implementación y adaptación a distintas tareas es sencilla, cuentan con una alta resolución y su costo es bajo.

La brújula digital CMPS03 es un sensor de campos magnéticos al calibrarse ofrece una precisión de 3-4 grados y una resolución de décimas. Cuenta dos interfaces de salida. Una PWM que varía su ciclo de trabajo de acuerdo al ángulo medido, y una interfaz de bus I2C, lo que facilita su comunicación con los microcontroladores. La brújula esta basada en los sensores KMZ51 de Phillips que son lo suficientemente sensibles para captar el campo magnético de la tierra. Usando dos de estos sensores colocados en ángulo de 90 grados, permite al microprocesador calcular la dirección de la componente horizontal del campo magnético natural.

Figura 11: Brújula Digital CMPS03 y pines.



La siguiente tarea fue comprender el protocolo de comunicación I2C y este es implementado en para lecturas en el CMPS03. Ya que este era el método de lectura con mayor precisión y que mejor se acoplaba a las necesidades del sistema.

Creación de rutina de calibración y lectura.

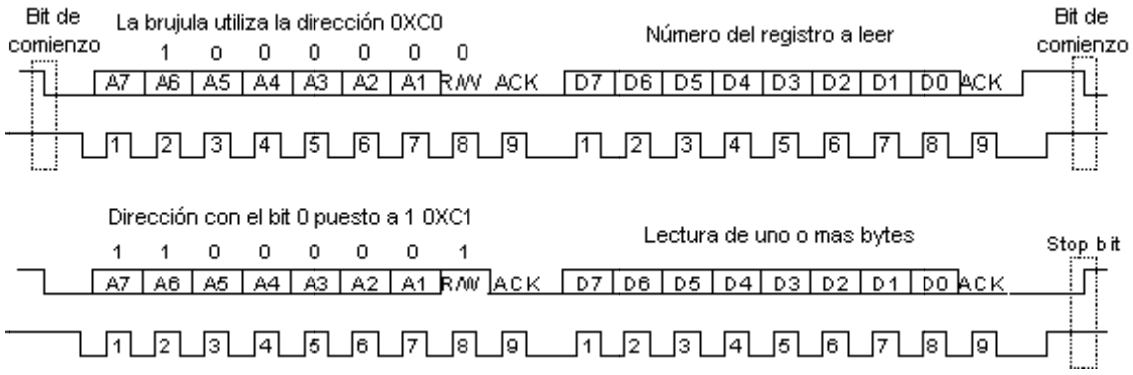
La calibración solo fue necesaria hacerla una vez, ya que los datos de este proceso fueron almacenados en la memoria EEPROM del PIC 16F872 (PIC interno del sensor). Para realizar la calibración el sensor requirió que se escribiera un 255 al registro 15 (registro exclusivo de calibración) y que se rotara la brújula 360° muy lentamente. Durante el periodo de calibración, el registro 14 dio una lectura de 0 y regresó a 255 cuando los cuatro puntos de calibración (Norte, Sur, Este, Oeste) fueron registrados. Finalmente se escribió un 0 al registro 15 para guardar estos datos a la memoria EEPROM.

Para realizar las lecturas, se implementó un programa en MikroC que acceso los registros de lectura del sensor por medio del protocolo de comunicación I2C.

La interfaz I2C está implementada en los pines 2 y 3 de la brújula, que permite una lectura directa del valor en grados de la dirección. Tal y como el

protocolo lo indica, primero se envió un bit de "start", el cual alerta al dispositivo que se enviarán datos por el bus. Luego se envió la dirección del dispositivo (0XC0) con el bit de lectura a cero. Una vez establecida la comunicación con la brújula se le indicó que registro se deseaba accesar. Después se volvió a mandar el bit de comienzo y la dirección del modulo con el bit de lectura a uno (0XC1) indicándole que se deseaba leer el registro anteriormente especificado. De esta manera se leyeron los registros de ocho bits que sostenían la posición del sensor en el plano XY.

Figura 12: Forma de las ondas I2C



Caracterización del sensor.

La caracterización del instrumento fue hecha en términos de precisión y exactitud, que eran los dos factores que importaban para la aplicación. Para tal caracterización se dibujaron líneas cada 10 grados, con un transportador, entre 0 y 350 grados. Luego, se alineó la orilla de la brújula dando una lectura de 90 grados con la línea correspondiente, hecha con el transportador. Se alineó la brújula en cada una de las líneas y se tomaron los datos. Este proceso se repitió cinco veces para poder analizar la precisión.

De la tabla 1 podemos observar que el sensor guarda una exactitud, que se encuentra entre las especificaciones de fábrica, entre los 20 y 150 grados. A partir de los 160°, se percibe un desfase que llega hasta los diez grados.

En cuanto a precisión, se observa en la tabla 1 que el máximo error en todas las medidas hechas fue de cuatro grados, lo cual cae dentro de las especificaciones de fábrica del sensor. Es este mismo factor el que nos permite realizar una recta de regresión para eliminar el error.

Ángulo Rea	Medida 1	Medida 2	Medida 3	Medida 4	Medida 5
0	4.9	0	0	1.5	0
10	14.2	15.7	15.2	14.2	15.7
20	23.5	23.6	23.5	23.5	23.7
30	32.5	33.9	32.7	32.9	33.7
40	42.5	43.5	42.8	43.1	43.3
50	51.1	52.1	52.1	51.6	52
60	61.1	60.8	60.8	60.9	61
70	69.4	70	69.8	70	69.5
80	79.3	79.6	79.4	79.3	79.4
90	89.9	89.2	89.9	89.5	89.7
100	101.2	102.2	101.7	101.9	102
110	109.8	110.4	110.4	109.8	110.1
120	122.5	121.1	121.5	122.1	121.7
130	134.6	131.9	131.9	133.5	132.4
140	144.3	142.6	142.6	143.5	142.9
150	153.8	154.4	154.2	154.1	154
160	165.8	166.4	166.2	166.2	166.1
170	177.4	177.4	177.4	177.4	177.4
180	188.8	187.6	188.8	187.6	187.9
190	200	198.5	199.8	200	199.4
200	211.1	209.3	210.6	209.1	209.9
210	221.2	221.5	221.5	221.2	221.4
220	231.2	231.7	231.5	231.7	231.5
230	240.9	241.5	241.3	241.2	241
240	250.4	250.5	250.5	250.5	250.4
250	259.4	260.7	259.9	259.9	260.5
260	269.7	270	270	270	270
270	278.6	279	278.6	279	278.7
280	288.7	289.7	288.7	288.9	289.4
290	299	300.1	300	299.9	300.1
300	309.5	309.7	309.6	309.4	309.7
310	319.6	317.3	318.5	317.6	317.4
320	328.6	326.7	327.5	326.7	327.6
330	338.2	337.3	337.4	337.3	337.5
340	347.5	345.5	345.9	347.1	345.8
350	355.5	355.5	355.5	355.5	355.5

Gráfica 34 Lecturas del sensor

Ecuación de regresión para corregir errores.

Para obtener la recta de regresión, se promediaron los datos de la tabla 1, de manera que se tuviera una única tabla de valores medidos contra valores reales. Dicho resultado puede observarse en la tabla 2.

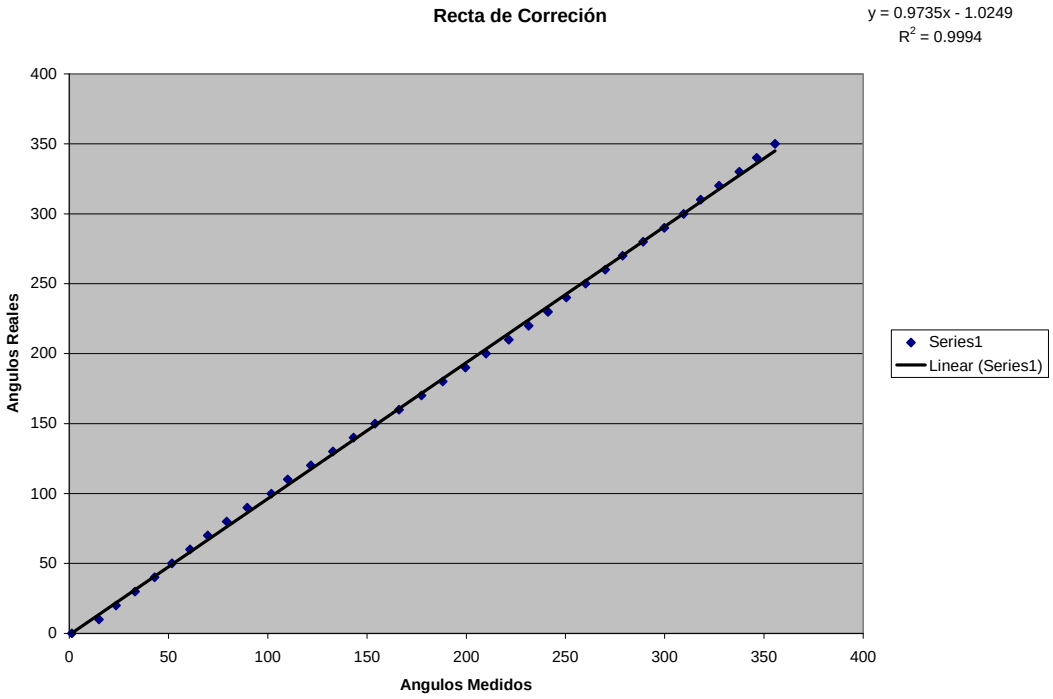
Ángulo Real	Medidas Promediadas
0	**1.28**
10	**15**
20	**23.56**
30	**33.14**
40	**43.04**
50	**51.78**
60	**60.92**
70	**69.74**
80	**79.4**
90	**89.64**
100	**101.8**
110	**110.1**
120	**121.78**
130	**132.86**
140	**143.18**
150	**154.1**
160	**166.14**
170	**177.4**
180	**188.14**
190	**199.54**
200	**210**
210	**221.36**
220	**231.52**
230	**241.18**
240	**250.46**
250	**260.08**
260	**269.94**
270	**278.78**
280	**289.08**
290	**299.82**
300	**309.58**
310	**318.08**
320	**327.42**
330	**337.54**
340	**346.36**
350	**355.5**

Gráfica 35 Valores promedio leidos

Los datos promediados fueron graficados utilizando estos como la variable independiente y los valores reales esperados como la variable dependiente. Este proceso se realizó para estudiar la tendencia y determinar qué recta se acoplaba de mejor manera a esta.

En la figura 6 puede observarse que la tendencia es lineal, por lo que se utilizó una regresión lineal, dando como resultados la ecuación de línea recta: **y=0.9735x-1.0249.**

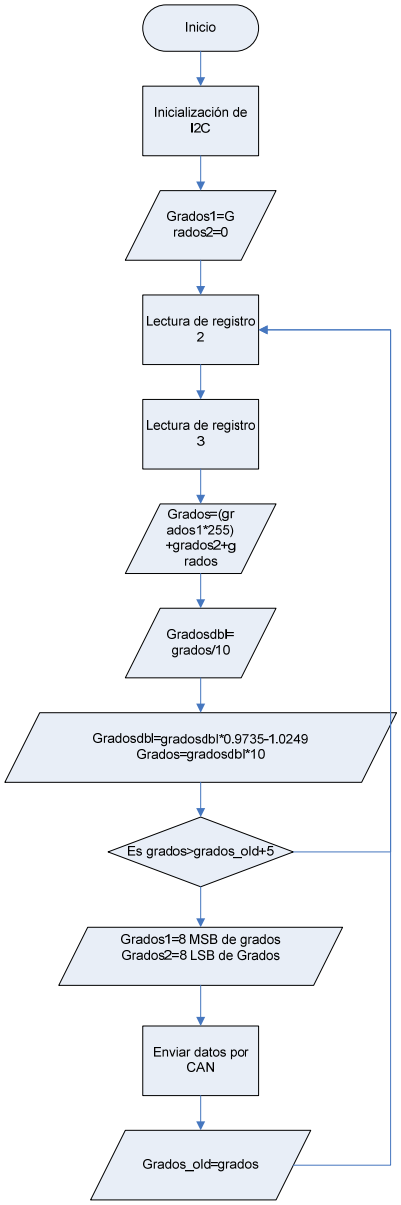
Figura 13: Datos Medidos vrs Datos Reales y ecuación de regresión lineal



Finalmente, el programa lee ambos registros de un byte del sensor y los almacena en variables distintas, “grados1” y “grados2”. Ya que “grados1” contiene los 8 bits más significativos del ángulo medido y “grados2” los menos

significativos, fue necesario obtener el dato entero real y almacenarlos en la variable “grados”.

Posteriormente se introduce la lectura en la ecuación y se obtiene el dato corregido. Con el dato corregido, se analiza si es mayor o menor al dato anteriormente leído, si cumple con esta condición el dato esta listo para ser enviado. De lo contrario, se termina la rutina de lectura.

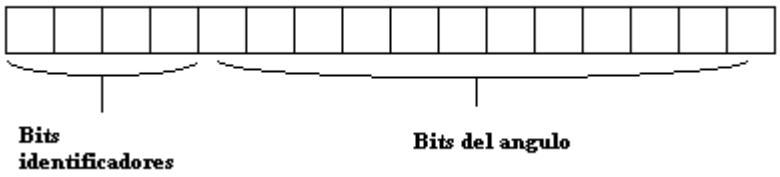


Gráfica 36 Diagrama de flujo del programa

Creación de trama de transmisión

La forma de la trama a ser enviada fue acordada con los submódulos de monitoreo y procesamiento de coordenadas. La forma de la trama que es envíada tiene la estructura que se muestra en la figura 8.

Figura 15: Forma de la trama.



Los bits del identificador le indican al módulo de "Procesamiento de Coordenadas" qué sensor es el que le está enviando la información, de acuerdo a la tabla 3.

Tabla 3: Bits de identificación de la trama.

Waist	0
Shoulder	1
Elbow	2
Wrist	3
Hand	4
Track	5

Grip	6
Monitor	7
Wiport	8

La brújula, en su modo de dos bytes de lectura, da números de 0 hasta 3599, que representaría 360°. De esta representación podemos ver que el número máximo de bits que dará el sensor son doce, los cuales se envían en los últimos doce bits de la trama.

De lo anterior, la trama que se enviaría a través de la red CAN para el número 360 sería: 0000111000001111.

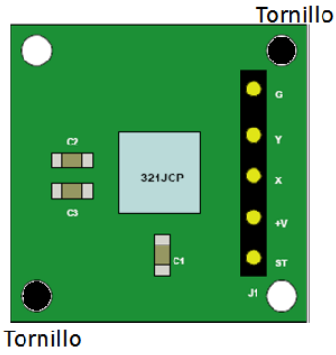
Para crear esta trama, se tomaron los dos bytes que se tenían de los registros 2 y 3 de la brújula (grados1 y grados2) y se forzó a cero los cuatro bits más significativos de la variable grados1, por medio de una rutina en lenguaje ensamblador.

Juri

A. Instalación de Sensores

Hay tres métodos que usados típicamente para el montaje de los acelerómetros: montaje con tornillos, montaje con goma y montaje con imanes.

Se utilizó el método de montaje con tornillo, pues de los tres es el mejor método para montajes permanentes. Este método permite que el transductor mida la aceleración según las especificaciones del fabricante. Se utilizaron tornillos de 4 x 3/8” para asegurarlos al brazo. Los tornillos están puestos en las esquinas opuestas del acelerómetro como se muestra en la figura. Entre el acelerómetro y el brazo se pusieron arandelas las cuales aseguran de una mejor forma los acelerómetros con el brazo, esto se hizo para evitar vibraciones en las mediciones.



Gráfica No. 22. Evaluation Board ADXL321

Para la localización del montaje del acelerómetro se tuvo cuidado que estuviera limpia, sin pintura y que no le ocasionara problemas de movilidad al usuario.

El acelerómetro se puede alinear de dos maneras para porder detectar el ángulo de inclinación de éste con respecto de la tierra. Una de las formas es

alinear el eje sensible paralelamente con la superficie de la tierra, la respuesta a un ángulo de inclinación θ, es $V_o(\emptyset) = V_0 + S_1(1 - \cos(\emptyset))g$, donde V_0 es la salida del acelerómetro a 0g (un valor distinto a cero para las fuentes de alimentación unipolares) y S_1 es la sensibilidad del acelerómetro en volts/g. El principio de funcionamiento del acelerómetro se basa en ángulos pequeños, por lo que el $\cos(\emptyset) \cong 1$, y se debería de detectar variaciones de voltaje muy pequeñas la cuales pueden ser afectadas drásticamente por el ruido.

La otra alternativa es alinear el acelerómetro ortogonalmente con la superficie de la tierra. En este caso, la respuesta a un ángulo de inclinación es $V_o(\emptyset) = V_0 + S_1 \sin(\emptyset)g$ Si se considera para ángulos pequeños como se hizo anteriormente se obtiene que $V_o(\emptyset) = V_0 + S_1 \emptyset g$, en este caso se puede observar que la salida dependerá entonces únicamente de la sensibilidad del acelerómetro y de su grado de inclunación.

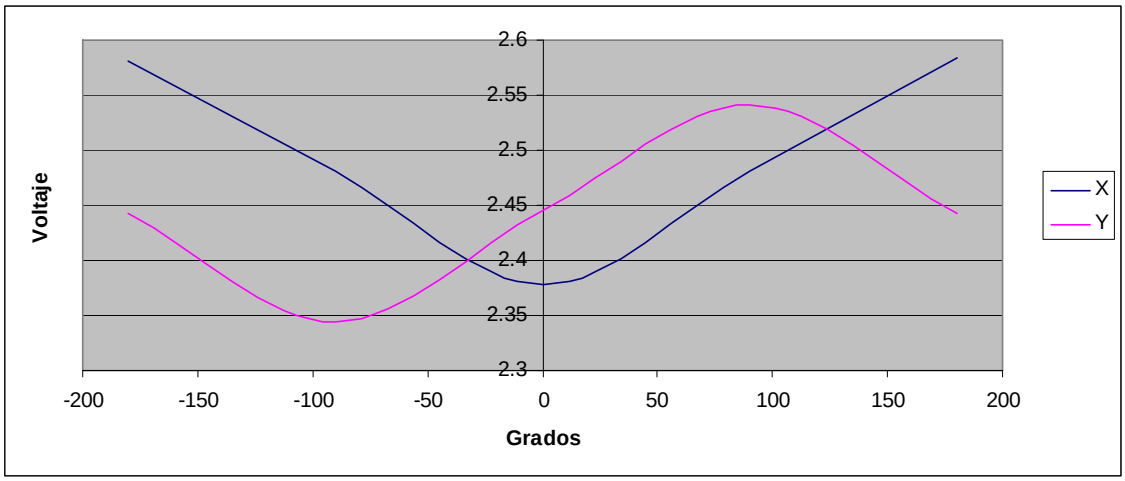
Para evitar que lazos de tierra crearan ruido en el sistema, el común del circuito y la tierra se conectaron con la tierra en solo un extremo. El procedimiento que se tomó fue de conectar los comunes y la tierra del lado del PIC monitor.

B. Pruebas de Funcionamiento

Se hicieron pruebas de funcionamiento con el acelerómetro ADXL321 y se obtuvieron los siguientes datos, alimentando el acelerómetro con 5V:

Grados	X	Y
-180	2.581	2.442
-90	2.481	2.344
0	2.378	2.445
90	2.481	2.541
180	2.584	2.442

Tabla No. 2. Funcionamiento ADXL321 5V



Gráfica No. 23..Funcionamiento ADXL321 5V

Estas son pruebas de funcionamiento de rotación vertical, donde 0º es el acelerómetro orientado con el pin #1 del acelerómetro hacia arriba, y 180º orientado con el pin #1 hacia abajo. El signo negativo denota el movimiento a favor de las agujas del reloj.

Como se puede ver en la Gráfica No. 38, la salida típica de un acelerómetro asemeja a la función del seno. El cambio de la inclinación corresponde directamente a un cambio en la aceleración debido a la gravedad que actúa en el acelerómetro. La pendiente de la curva es la sensibilidad del dispositivo.

Como se puede ver en la Tabla No. 5, el máximo valor de salida del acelerómetro es 2.584V y el mínimo es 2.344V. Lo que deja un rango de trabajo de 0.24V.

Para poder trabajar con mayor resolución en este rango, se referenció el PIC 18F458 con dos voltajes para su conversión analógica digital. El voltaje de referencia menor es de 2.3 V y el mayor es de 2.6 V, el cual deja el acelerómetro en un área de trabajo de aproximadamente 80%.

Con la ayuda de la herramienta Microsoft Excel se pudo hacer una función que se apegara a la función X

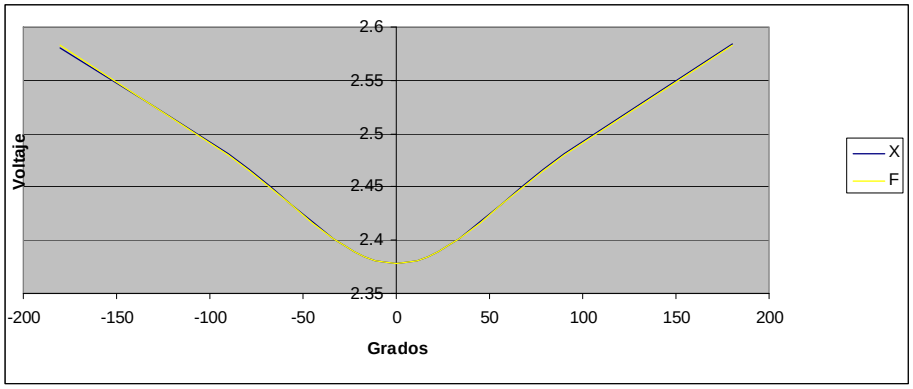
Grados	X	Fx	X-Fx
-180	2.581	2.582697	0.001697
-90	2.481	2.480349	0.000651
0	2.378	2.378	9.64E-09
90	2.481	2.480349	0.000651
180	2.584	2.582697	0.001303

Tabla No. 3. Aproximación X con MSExcel

Esta función esta descrita por la siguiente fórmula:

$$Fx(\phi) = -0.102\cos(\phi) + 2.48$$

Como se puede observar en la Tabla No. 7, el error es menor que el 0.1%, pero esta función presenta un consumo de recursos para el PIC, ya que el coseno no es una función lineal, lo que hace que el procesamiento del ángulo sea lento.



Gráfica No. 24. Modelo de MSExcel Salida Eje X

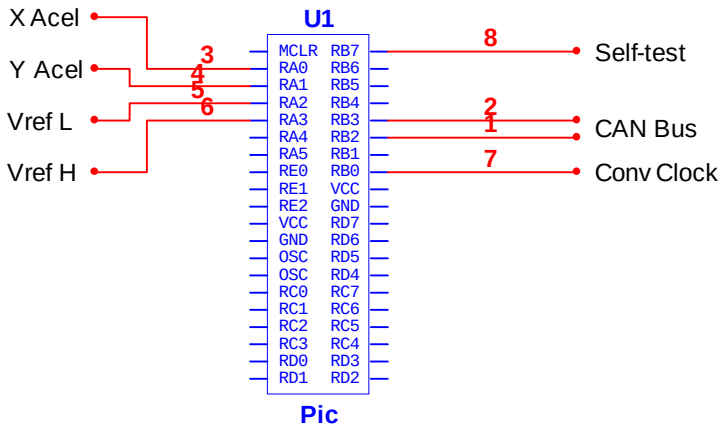
Como se puede ver en la gráfica, la respuesta de salida X del acelerómetro es par, por lo que se puede obtener el valor absoluto del ángulo simplemente con la parte positiva de la función. Si se observa solo la parte positiva de la función, también se puede concluir que presenta un comportamiento lineal. La ecuación de éste comportamiento es:

$$Fx(\phi) = 0.0011x + 2.378$$
$$R^2 = 0.99998$$

La cual es una solución más sencilla de implementar en el PIC, ya que consume menos cantidad de recursos. Por consecuencia el tiempo de procesamiento de la señal es menor.

El signo del ángulo se obtiene con la otra señal de salida del acelerómetro. Como se puede observar en la Gráfica No. 38, el voltaje de salida para ángulos negativos es menor a 2.445 y para ángulos positivos es mayor.

C. Captura de Datos Sensores



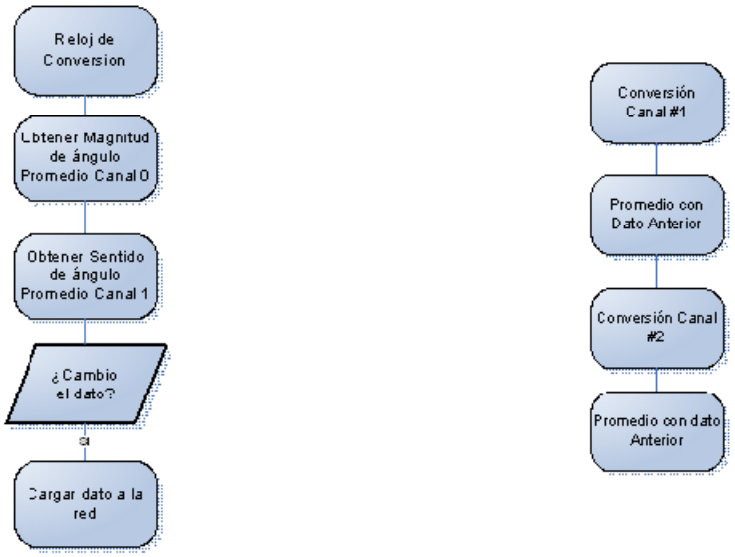
Gráfica No. 25. Conexión Acelerómetro PIC

Como se puede observar en la gráfica 18, la conversión analógica digital del acelerómetro se hace por medio de dos canales del PIC Los otros dos canales analógicos que se utilizan del PIC son canales de referencia, ya que como se mostrará posteriormente, el acelerómetro varía 0.3 V en su rango de trabajo.

También se posee un reloj, 3kHz, el cual dará el tiempo de conversión, aunque la conversión del PIC tendrá su propio reloj. Mientras el reloj de conversión no de la señal de conversión, los datos serán acumulados en el PIC, cuando el reloj de la señal de conversión, estos datos acumulados son promediados, lo cual como se explicó anteriormente, aumenta drásticamente la eficacia y confiabilidad de la medida.

El sensor genera una salida analógica proporcional a la aceleración para cada uno de los dos canales de salida. El componente de aceleración de cada eje se deriva fácilmente al convertir la señal a digital. Esta señal se procesa para

determinarse si ha ocurrido un cambio de severa importancia para que el robot reaccione.



Gráfica No. 26.Programa de Captura de Datos

Se hicieron pruebas de funcionamiento con el programa de captura de datos, con este se pudo observar que se presentaban voltajes aleatorios que rompían con la linealidad del sistema. Estos no se presentaban cuando el acelerómetro se encontraba en reposo, solamente cuando se encontraba en movimiento. Esto motivó a cambiar la rutina de captura de datos para minimizar el error que se producían en las mediciones.

Una solución que se encontró a este problema fue utilizar una filtración digital (hacer un promedio de los canales) para reducir ruido. A pesar de las ventajas que trae, hace más lento el sistema, por lo que reduce su ancho de

banda. En la aplicación en que se tiene, el acelerómetro no se necesita un ancho de banda grande, pues los movimientos del usuario de la silla son relativamente lentos.

Las pruebas anteriormente mencionadas se hicieron con voltajes de referencia para la conversión, los cuales fueron 2.3V y 2.6V. Este rango es en el que el acelerómetro funciona típicamente, pero al hacer pruebas con el programa y comprobar que existía un comportamiento no lineal, se tomó como voltaje de referencia 0V y 5V. De esta manera se observó que las magnitudes de estos errores no eran constantes.

Otro problema que se presenta es que en los ángulos límites, ángulos muy cercanos a 0º ó 360º, la linealidad se pierde, no en lo absoluto, pero no sigue el modelo presentado anteriormente. La sensibilidad de volts/grado del sensor disminuye en este sector.

D. Creación de Paquetes para enviar en la Red

A la data obtenida anteriormente se le agrega en sus cuatro bits más significativos, el identificador según corresponda. En este caso al acelerómetro 1 se la agregará en sus 4 bits más significativos el identificador 1, y al acelerómetro 2 se le agregará el identificador 2.

	ID	**DATA**	**DATA2**
	4	**4**	**8**
		MSB	**LSB**
Waist	0		

Shoulder	1		
Elbow	2		
Wrist	3		
Hand	4		
Track	5		
Grip	6		
Monitor	7		
Wiport	8		

Tabla No. 4. Identificadores Red

A. Investigación

Después de una previa investigación, se tomó la decisión de adquirir los sensores rotacionales, absolutos magnéticos MA3 (US-DIGITAL). La ventaja de éste sensor respecto uno óptico, radica en el costo, y satisface con los parámetros deseados de tiempo de muestreo y error en la medición, como se ha comentado con anterioridad en la delimitación del tema.

El rango de operación de los sensores rotacionales está entre 0 y 360 grados, y mapea una señal de 0 a 5 Volts lineal y proporcional al giro. Al lograrse 360 grados rotados, el sensor se refresca y la medición siguiente corresponde a 0 grados nuevamente. Es decir, que su escala de medición está delimitada al girar una vuelta completa la perilla del sensor, ya que éste se *reiniciará* después de haber

completado los 360 grados. Con esto se define un modelo lineal para el sensor. El error de fábrica del codificador es de 0.9 grados ó 20 mVolts.

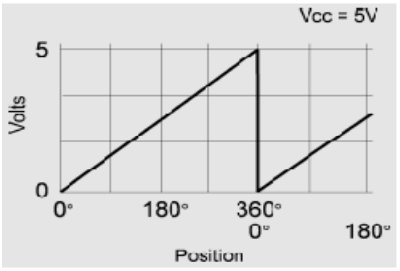


Gráfico 5 Describe el Comportamiento Lineal del Sensor

B. Herramientas de Trabajo

Para poder fabricar los engranajes fue necesario implementar dos diseños previos de los mismos, los cuales se elaboraron en AUTOCAD. Actualmente el departamento de mecánica de la Universidad del Valle de Guatemala posee un sistema FMS, para la elaboración de piezas mecánicas.

Para elaborar los engranajes fue necesario comprar una fresa HSS de 2mm y posteriormente una fresa de Tungsteno, también de 2mm. A su vez fue necesario adquirir dos tipos de plásticos distintos, dos polímeros (Vekton®), cuya diferencia radica en el espectro de resistencia y durabilidad contra fuerzas de fricción (especial para aplicaciones industriales), y en el precio. La fabricación de los engranajes la llevó a cabo

el operario Dany Ortiz, por lo que él se encargó de la programación del sistema FMS, para la elaboración adecuada de las piezas. La adaptación del sensor sobre la estructura se hizo con una ¨L¨ de aluminio, y para ello se perforó la estructura para atornillarla. Se le hizo una ranura rectangular de longitud igual a la del diámetro de la rosca del codificador. La altura de esta ¨L¨ varía según la relación entre los engranajes, de manera que casen los dientes del engranaje en el sensor con e atornillado a la extremidad o juntura de la estructura. El sensor requiere de un poco de pegamento especial para permanecer fijo con el engranaje.

Gráfico 4 engranajes con relación 1:1

Gráfico 4 engranajes con relación 1:2

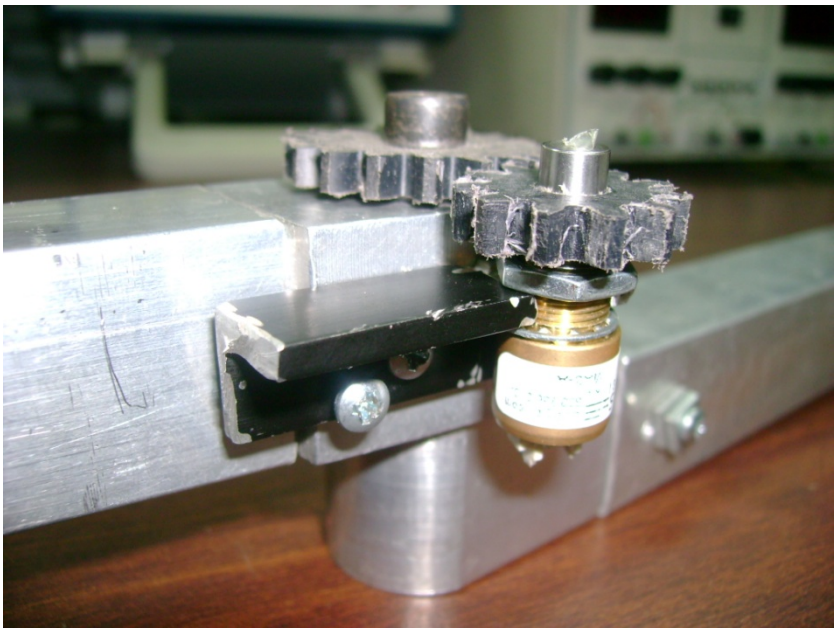


Gráfico 7 Instalación del codificador sobre la estructura

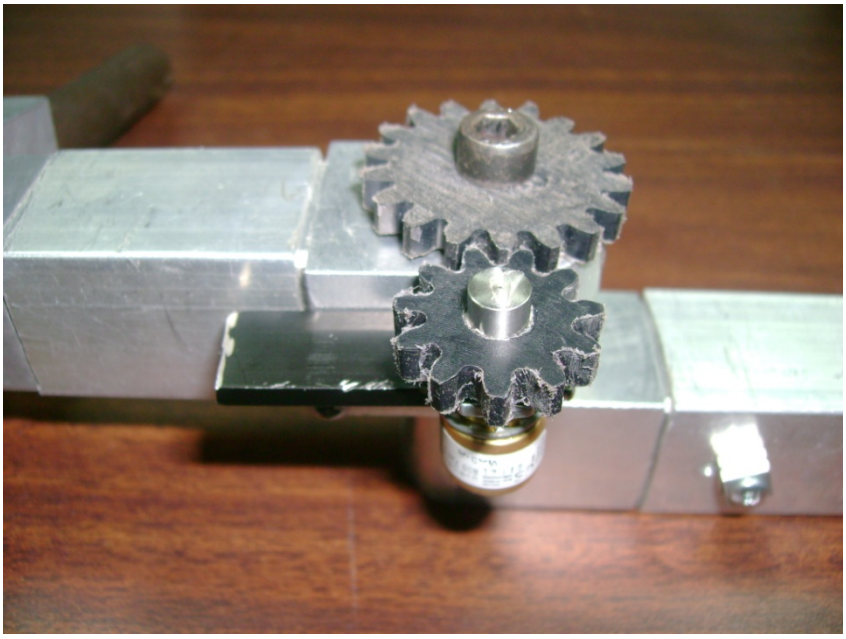


Gráfico 8 Instalación del codificador sobre la estructura

C. Procesamiento de Datos

La última fase require de la programación de un microcontrolador 18F458 para realizar la lectura y el procesamiento necesario para armar una trama, y transmitirla por medio de la red CAN (incluir foto de encoder, materiales y engranajes, más plano en cad).

Para instalar el sensor que detecta los movimientos de la mano en la estructura, se requirió de la fabricación de un juego de engranajes. La relación de los engranajes corresponde a una relación 1:1, por lo que no fue necesario implementar un ajuste en la programación del controlador que procesa las señales adquiridas por los codificadores. La señal se procede a *binarizarse* después de haberse procesado en el ADC interno del controlador. De ésta manera se obtiene un rango de números decimales con un resolución de 10 bits, es decir de 0 a 980.

Para instalar el sensor que detecta los movimientos de la muñeca en la estructura, se requirió de la fabricación de un juego de engranajes. La relación entre los engranajes es de 1:2. Para detectar adecuadamente los movimientos capturados fuera del rango nominal del sensor fue indispensable programar el controlador para que detectara el *refrescado* de

los sensores y se procesara la señal con la escala ampliada, es decir un rango decimal de 0 a 1860.

Inicialmente, al momento de definir el megaproyecto, elegimos que por convención las tramas de los sensores fueran de 2 bytes para cada uno. En la cual se está considerando, del bit menos significativo al más significativo, 10 bits de datos (señal de posición), 1 bit para ampliar la escala, y tres bits de encabezado que determinan el identificador definido para cada sensor. Los codificadores que determinan el movimiento deseado por el usuario, para la muñeca y la mano, tiene el identificador decimal 3 y 4, respectivamente.

Trama (16 bits)	MSB				ID	OV						
MANO	0	0	0	1	1	X	X	X	X	X	X	X
MUÑECA	0	0	1	0	0	X	X	X	X	X	X	X

Tabla 1 Trama de 2 Bytes orientada a bit.

Trama Decimal	**MAX**	**MIN**
MANO	8191	6144
MUÑECA	10239	8192

Tabla 2 Trama de 2 Bytes orientada a números decimales.

Después de armar la trama final que se desea enviarla. Se procederá a utilizar el protocolo CAN y la librería diseñada por el módulo de CAN, y es por ello

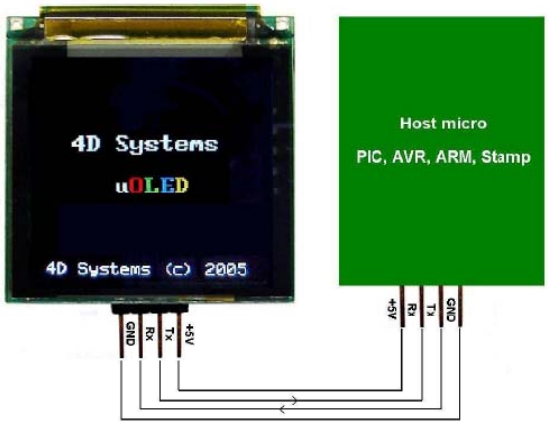
que se implementó un microcontrolador 18F458 de microchip, ya que ésta familia trae dicho módulo implementado en su pines.

Monitoreo

Adquisición de Materiales

Tras una investigación en internet, buscando la tecnología más moderna en pantallas electrónicas, se procedió a adquirir una pantalla Oled. En particular se encontró una pantalla con interfaz serial, la cual puede ser manejada por un PIC, la uOled con 128x128 píxeles y 65,000 colores de la empresa australiana 4dsystems. La gráfica no. 4 ejemplifica su interfaz con un PIC.

Gráfica No. 27 - Pantalla uOled



Al mismo tiempo, se buscó el PIC con mayor memoria RAM y ROM que poseyera interfaz CAN. Se adquirió el PIC 18F4680 para la implementación final y el PIC 18F458. El primero solo serviría de apoyo

por si el 458 no tenía la suficiente ROM ó RAM para contener todo el programa, dado que el 4680 tiene el doble de capacidad de ambas memorias.

Sin embargo, como se describe después, la pantalla uOled presentó problemas para la aplicación que se desea desarrollar y tuvo que ser reemplazada por la pantalla, mundialmente probada, LCD monocróma gráfica o (GLCD). En específico, se utilizó la pantalla ITM-12864A LCM, que tiene dos controladores KS0108, que son los que proporcionan la interfaz paralela para su control.

La pantalla uOled fue escogida por la alta resolución de su pequeña pantalla y los colores con los que permite encender cada uno de sus píxeles, con lo cual da una idea de todo el trabajo interno que conlleva el sistema. Sin embarg r el tipo de tecnología, requiere de mucha protección en lo que respecta a una energía limpia y en específico, al cuidado que debe de tener al momento de apagarse. Por su naturaleza, el estar mucho tiempo encendida puede llevar a quemar los pixeles en la pantalla como sucede en ciertos dispositivos como las *PDA,* con tecnología LCD multicolor. Sin embargo, con la pantalla uOled se experimentó que esta situación era un poco más delicada. El sistema de apagado se debía de realizar primero por software, con lo cual se desenergizan transistores de potencia que alimentan la pantalla para evitar el problema del *burn-in* o quema de píxeles en la pantalla.

La pantalla se utilizó luego de que la librería de control estuvo completa, buscando evitar la posibilidad del burn-in, dado que la pantalla requiere apagarse por software antes de que sea removida la energía. Sin embargo, la pantalla sufrió de una pérdida de luminosidad la cual,

eventualmente fue degradando la pantalla y solo su controlador respondía a comandos, que no se reflejaba en la pantalla. El problema fue transmitido a la empresa de 4dsystems, la cual respondió inmediatamente y mandó un reemplazo por garantía. Sin embargo, a partir de esto se decidió hacer el cambio por una pantalla más grande y más robusta, aunque menos llamativa, que fue la GLCD previamente descrita. Esta pantalla tiene la característica de poder mantener encendido por más tiempo un pixel sin que este se queme en la pantalla, es decir, sin que genere el efecto del *burn-in*. Esta ha funcionado sin ningún problema.

Gráfica No. 28 Pantalla Inicial

Modo de Despliegue

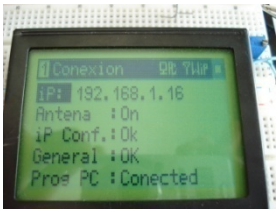
Con la limitación de la resolución de ambas pantallas, se concluyó que el sistema podrá ser manejado a través de los siguientes menús y submenús:

Menús:

Gráfica No. 29 Menú Monitoreo

- Monitoreo: contiene el diagrama del robot, con el cual se busca que el usuario identifique fácilmente el R17 y vea qué articulación es la que está presentando un posible problema o se encuentra deshabilitada para su protección. Para llamar la atención del usuario, se implementó una alarma intermitente.

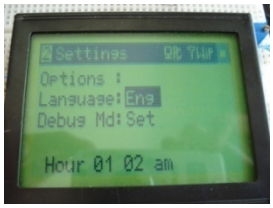
Gráfica No. 30 Menú Conexión



- Conexión: contiene datos como la IP de la silla operativa, para poder ser modificada, el despliegue de errores con la red Wifi o problemas con el

Wiport. Además, permite revisar que exista conexión con el programa de la PC.

Gráfica No. 31 Menú Configuración



- Configuración: busca poder mover parámetros del despliegue como cambió idioma, entrar en modo de edición, ver el ajuste de la hora para comprobar comunicación con la PC, o cualquier otra función necesaria de optimización.

Submenús:

Gráfica No. 32 Barra de Estado, Barra lateral y Selección

- Barra de Estado: estará en la parte superior de los menús anteriores para despliegue de la correcta comunicación con el programa de la PC, el estatus del Wiport y conocer el modo de grabación en que se encuentra la silla con el R17.

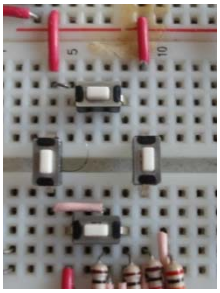
- Menú para el modo de selección de grabación, reproducción, pausa o paro, el cual llevará también el tiempo de grabación.

- Menú de selección, para poder navegar entre cada uno de los menús principales

Entradas y Salidas del Submódulo

Para el manejo de la pantalla, se optó por no utilizar el teclado matricial, dado que está reservado en su mayoría para aplicaciones de seguridad y porque se desea que la interfaz sea sencilla para que el usuario pueda operar todo el sistema desde la silla operativa.

Gráfica No. 33 Botones de Navegación



Se optó por un sistema en cruz de 4 botones y un interruptor. Con éstos cuatro botones se tiene la opción de reproducir o parar, los movimientos que se generen con la silla operativa, y de grabar o pausar el movimiento que se desee programar. Con estos mismos cuatro botones en la otra modalidad del interruptor de selección, se tienen las funciones de arriba, abajo, validar o cancelar, dentro del menú de despliegue de la pantalla, con lo cual se puede realizar cualquier operación basado en unos menús sencillos y fáciles de comprender y navegar.

Los botones se implementaron mediante los interruptos de cambio de flanco del puerto B del PIC, con lo cual se asegura que el pulsado del botón va ser recibido por el PIC. Al momento de ser validada una acción de interrupto, mediante una bandera se forza un ciclo completo de despliegue antes de que permita aceptar otro comando, refrescando la pantalla con la acción requerida.

En la tabla No. 2 se resumen los puertos utilizados en el PIC:

Cuadro No. 2 - Puertos del Microcontrolador PIC

Descripción	Puerto
Estado del PIC	A0
Comunicación CAN	B2-B7
Pulsadores	B4-B7
Pines de Control del GLCD	C0-C5
Comunicación Serial	C6-C7
Pines de Datos del GLCD	D0-D7

Como se puede ver, el puerto B contiene la salida de varios módulos. La parte de control de la pantalla se realiza con los primeros seis bits del puerto C, dejando libre los pines del módulo de comunicación serial del PIC, por lo que se podría implementar con el sistema, algún tipo de gestión local con salida de tipo RS-232 o una pantalla extra controlada por medio de una interfaz serial. La tabla No. 3 resume la comunicación que se da entre el PIC y el GLCD.

Cuadro No. 3 - Pines de control del GLCD

Pin del PIC 18F	Pin del GLCD	Descripción
C1	CS1	Seleccionador del Chip de control del GLCD de los primeros 64x64 pixeles
C0	CS2	Seleccionador del Chip de control del GLCD de los 64x64 pixeles izquierdos.
C2	DI	Especificación para dato o instrucción
C3	RW	Lectura o Escritura
C4	E	Habilitar (*Enable*)
C5	RST	Reset de GLCD
Puerto D	D0-D7	Datos (Comunicación Paralela)

D. Librerías y funciones dentro del submódulo

1. Programación de la pantalla GLCD y sus botones

Al especificar los pines descritos en el Cuadro No. 2 se programaron en base a las librerías abiertas de PIC C CCS, donde se modificó la librería de una "*Hantronix HDM64GS12 with a KS0108 display controller*", que se acopla perfecto para la pantalla GLCD que se seleccionó. Esto simplificó el manejo de la pantalla y redujo el tiempo perdido en el desarrollo de la librería para la pantalla uOled que no se utilizó.

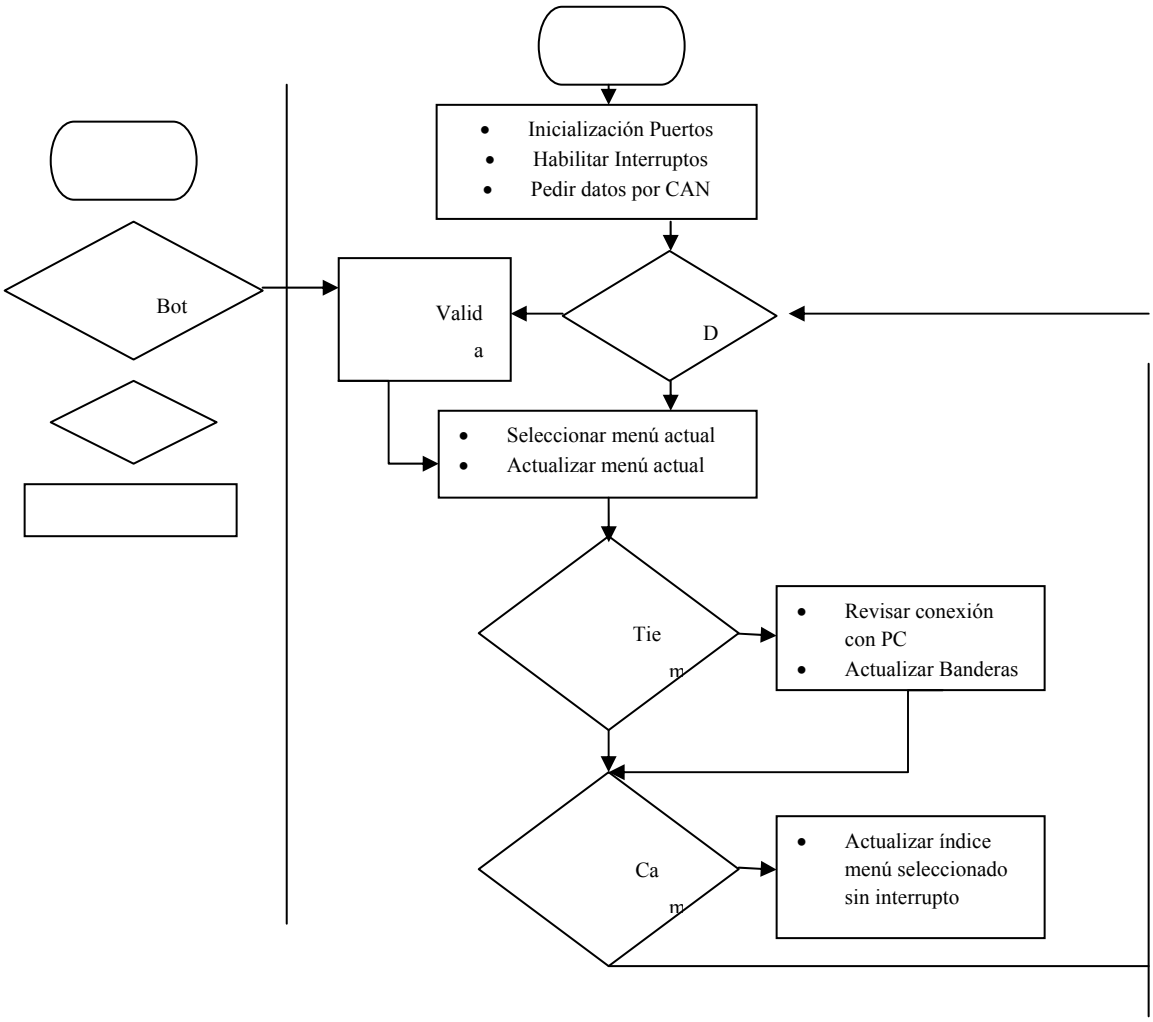
Para la parte del manejo de los pulsadores, se creó el archivo de <Input_Debounce.c>, con el cual se manejan los interruptos del puerto B analizando y separando el filtrado del botón o *debounce,* entregando solo un carácter a la función de validación de la acción de los botones. Con esto se puede lograr una sustitución de los botones pulsadores por un teclado matricial 4x4, utilizando los mismos interruptos del puerto B y conectando los retornos, del teclado matricial, a pines del puerto A disponibles, entregando caracteres a la función de validación, la cual está en un archivo aparte.

Esto se logra mediante 2 variables globales, una que indica qué menú es el actual y otra en la que se indica la posición en ese menú. Haciendo que la función "Boton_switch(xx)" del código reconozca en qué menú está y qué tipo de validación se debe de efectuar, permitiendo anidar submenús sin mayor complicación que la programación del nuevo menú.

La gráfica No. 14 resume el ciclo que realiza el código del programa para el submódulo. Se observa que el programa de validación de los botones corre por aparte, en la rutina de servicio de los

interruptos, actualizando la información en la pantalla un ciclo después de ser modificada.

Gráfica No. 34 – Diagrama de Flujo del Programa



Al tener las variables globales para saber que menú es el actual y en qué posición del menú se encuentra, se especificó una función de validación y una función para el despliegue en pantalla para cada menú.

2. Implementación de un reloj con los módulos de temporizadores del PIC

Se utilizó un oscilador de 20MHz para alimentar el microcontrolador, con lo que tarda 0.2us por instrucción. Para la librería del reloj, se utilizaron los temporizadores descritos en el cuadro No.4. Se tiene un Timer0 con un registro que cuenta de 0 a FFh, en hexadecimal, el cual genera una interrupción en el paso de FFh a 00h. El Timer1 de 16 bits que genera su interrupción en el paso de FFFFh a 00h. Este se forza en el evento de interrupción, a que su registro empiece a contar en 19531. Por último el Timer2, que genera su interrupción al llegar al límite del registro especificado en 210.

Cuadro No. 4 - Temporizadores del PIC

Reloj	Timer	Prescaler	Registro Timer	Postcaler
20 MHz	0	256	0	1
IPS	1	16	210	16
0.2uS	2	8	19531	1

En base al cuadro No.5 se obtienen los siguientes resultados mostrados.

Cuadro No. 5 - Error por Temporizador

Timer	Interrputo (s)	Multiplicador	Tiempo (s)	% Error	Min. / Hr. (Retrasados)
Timer0	0.0131	76	0.996147	0.3853	13.870
Timer1	0.0312	32	0.999987	0.0013	0.046
Timer2	0.0108	93	0.999936	0.0064	0.230

Como se puede observar en el cuadro No. 5, el temporizador con menor error es el Timer1 con un atraso de 0.05 minutos cada hora, al cual le sigue el Timer2 con un atraso de 0.230 min/h y por último el timer0 con 13 min/h. Para la aplicación que solo requiere llevar el tiempo de grabación, el Timer1 o 2 cumplen su función dado que se considera que el máximo de grabación que tendrá será de ciertas de horas por sesión y el reloj de la pantalla se actualiza cada vez que entra en el menú de configuración.

Pueden haber otras configuraciones con los pre-escaladores de los temporizadores y con los registros de los temporizadores, pero los descritos anteriormente son los que generan el mayor tiempo entre interrupción, dejando el tiempo de ejecución del ciclo normal del programa mas fluido. De tener solo el Timer0 disponible, se puede forzar al igual que el Timer1, a que en su interrupción se reescriba el registro del temporizador para que empiece en 46 reduciéndolo a un tamaño de 210, solo que a diferencia del Timer0 el Timer2, no tiene esa necesidad. Hay que tomar en cuenta en los datos a calibrar que el interrupto ocurre al terminar de contar el FFh y no al llegar al FFh.

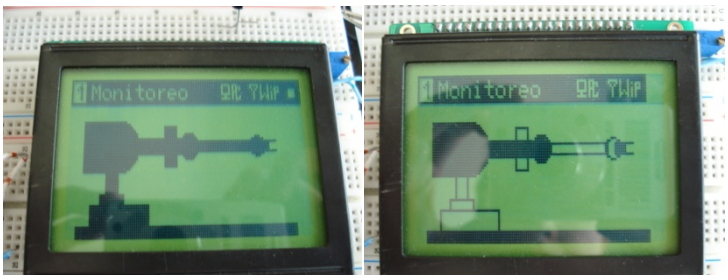
3. Desarrollo de los menús de despliegue

Para la barra superior que siempre esta a la vista, se imprime el nombre del menú actual con base a la variable global *msel*, menú seleccionado, que es la variable que determina en qué posición del menú se encuentra la pantalla actualmente. Además, se imprime el valor de *mpos*, posición en menú, la variable global de la posición actual del menú, con lo cual se facilita la explicación en un manual de usuario o a la hora de hacer alguna referencia a una opción, a base de seguimiento de números.

Se tienen a la vista alarmas globales del Wiport, de la comunicación con el programa de la PC y el estado de grabación. Las primeras dos, en caso de haber problemas, se hacen titilar mediante un múltiplo de la velocidad del ciclo del programa, que alterna una variable habilitando la impresión intermitente de los parámetros que no están en el modo normal de uso.

Para el menú de monitoreo, se consideró que la forma más fácil de anunciar una falla o una alarma sobre alguna de las articulaciones del R17 era mediante el trazado de su esquema en la pantalla, modificando cada una de sus articulaciones por separado para el reconocimiento del problema de una forma inmediata por el usuario. Éstas se hacen encender y apagar a la misma velocidad que las variables que puedan estar en error en la barra superior, usando la misma técnica.

Gráfica No. 35 Articulaciones de cintura, codo y mano deshabilitadas



Esto se diseñó de tal manera que cuando el robot está llegando a su límite en alguna de las articulaciones, se muestre la articulación en problemas y el usuario sepa que está trabajando sobre los límites de tal articulación del robot y que si el movimiento continua, para evitar daños en el robot, los límites predefinidos en la PC evitarán que las coordenadas lleguen hasta el controlador del R17, deshabilitándolo en ese sentido, evitando que sufra algún daño y el usuario logrará saber mediante el aviso de una imagen fija de error, que no ha habido falla de comunicación, sino el movimiento ha sido deshabilitado en ese sentido de esa articulación en esa dirección.

El sistema se especificó de tal manera que, cuando alguna articulación entra en estado de precaución, la computadora manda el

parámetro de todas las articulaciones que están dentro del límite de precaución e igualmente manda la información cuando desactiva articulaciones, desplegando el valor de cada una individualmente hasta que ocurre un cambio y regresan reenviando el estado del parámetro en su estado normal.

El robot se dibujó basado en los diagramas de Strobotics del R17. En base a esto, las medidas en centimetros se trasladaron a píxeles y se procedió a hacer el dibujo mediante la función de drawbot, que contiene las funciones de cada articulación por separado para ir variando el despliegue de estas según el estado en que se encuentre.

Para el menú de configuración, se deteminó que una de las funciones más importantes es la implementación del cambio de idioma, lo que se considera una de las funciones básicas para la personalización y comodidad del despliegue de información en general. Esto se implementó mediante el uso de una bandera global, la cual se toma en cuenta a lo largo del código de despliegue seleccionando la fuente en el idioma deseado. Par el PIC 18F458 el programa final consumió el 80% del total de su memoria RAM y 53% de la ROM, por lo que la implementación de esta función que consume recursos, no fue problema.

También se agregó la función de Debug, o Modo de Prueba, con la cual se pueden escribir a las variables que se ven modificadas solo desde la comunicación CAN, útil tanto para cuando la pantalla está en desarrollo, como para explicar y dar a entender funciones en la pantalla, sin que ésta se encuentre conectada.

Para el menú de conexión, se centralizaron todas las alarmas referentes a la comunicación, tanto de la PC con la silla operativa, como del funcionamiento del Wiport. En este menú se decidió desplegar la IP actualizada que el Wiport posee, facilitando así su conexión en la red de Ethernet, o su cambio mediante su servidor de página web. Esto facilita mucho su proceso de conexión, dado que en muchos sistemas inmersos con conexión a Ethernet, la IP del equipo no es fácilmente visible o su IP default se encuentra en un manual no disponible en el campo y teniendo que recurrir a otros métodos menos sencillos para que el usuario pueda reconocer su IP.

Se solicita la IP al Wiport, cada vez que se entra al menú, confirmando el dato que se despliega y puede ser recibida en cualquier momento en que haya cambio, desplegando la IP más reciente que se tiene en memoria. También se posee información de errores en el Wiport (WiP en pantalla), donde se especifica si la IP fue duplicada, si la antena esta apagada o si hay un error general.

Además de esto en el menú principal, como se observa en la gráfica No. 5, se realizan funciones periódicas. Entre ellas verificar el estado de comunicación con el programa en la PC, de no estar conectado informa en la barra superior y pone en *stop* el modo de grabación del brazo, hasta que se habilite la conexión con el programa nuevamente. Esto informa al usuario que el problema no esta en la silla operativa sino en el programa de la PC.

4. Comunicación CAN

En el cuadro No. 6 se resume la forma en que se realiza la comunicación en los dos bytes que se transmiten a través de CAN para el submódulo de monitoreo.

Cuadro No. 6 - Recepción del Submódulo de Monitoreo, datos de PC

Byte	MSB	LSB
Límite Precaución	10	00 + 6 bits de Articulaciones
Límite Deshabilitado	11	00 + 6 bits de Articulaciones
Unidad Minutos	20	Dato en decimal
Decena Minutos	21	Dato en decimal
Unidad Hora	22	Dato en decimal
Decena Hora	23	Dato en decimal
AM/PM	24	1 PM 0 Am
Acknowlege PC en línea	25	--
Handshake con PC	26	--
Estado de Grabación	27	1-Play, 2-Stop, 3-Rec, 4-Pause

		(en Decimal)

Las articulaciones descritas en el cuadro no. 6, contando los bits de 1 a 8, del menos al más significativo, el bit 1 es la cintura, 2 el hombro, y en orden sucesivo, codo, mano, muñeca y el riel es el bit 6.

Cuadro No. 7 - Recepción del Submódulo de Monitoreo, datos de Wiport

Byte	MSB	LSB
Status del Wiport	101	00000 + Error General + IP Duplicada + Antena On/Off (0/1)
IP del Wiport	102	IP MSB byte (decimal)
IP del Wiport	103	IP 2ndo byte (decimal)
IP del Wiport	104	IP 3er byte (decimal)
IP del Wiport	105	IP LSB byte (decimal)

Al igual, en el cuadro no. 7 la antena es el bit 1, la Ip duplicada o error de IP el bit 2 y el bit 3 un error general de conexión del Wiport.

Cuadro No. 8 - Transmisión del Submódulo de Monitoreo a PC

Byte	MSB	LSB
Handshake con Monitoreo	112	1
Request PC en línea	112	2
Request Hora	112	3
Modo de Grabación	112	4-Play, 5-Stop, 7-Rec, 6-Pause (en Decimal)
Izquierda Waist	112	8-Mueve el punto de referencia de la brújula hacia la izquierda una posición
Derecha Waist	112	9-Mueve el punto de referencia de la brújula hacia la derecha una posición

Las últimas cuatro funciones del cuadro no. 7 son parte del submódulo de monitoreo, pero son implementadas por el PIC del

submódulo de procesamiento de señales que maneja la brújula digital, mediante un joystick de pulsadores.

Cuadro No. 9 - Transmisión del Submódulo de Monitoreo a Wiport

Byte	MSB	LSB
Request IP	241	--
Request Estado de Wiport	242	--

Para el cuadro No. 9 los bytes menos significativos no tienen uso.

La comunicación CAN que se realiza es una implementación de una función entregada por el módulo de CAN, tanto para la transmisión como para la recepción. Se realiza usando 2 bytes, dado que es la cantidad necesaria para la correcta transmisión de los encoders dentro de la silla operativa. En el primer byte, los primeros 4 bits dan al programa de la PC el identificador del módulo transmisor del paquete. En el caso del submódulo de monitoreo, su ID es 7. Los otros 12 bits de información se usan para dar los datos.

La comunicación que se tiene para el estado de grabación en que el usuario desea poner al sistema, es una comunicación bidireccional. Se manda a la PC el estado en que se desea poner el sistema y el usuario es notificado hasta el momento real en que la PC ha entrado en el modo especificado, consiguiendo una seguridad en los datos desplegados y la comunicación con el programa de la PC.

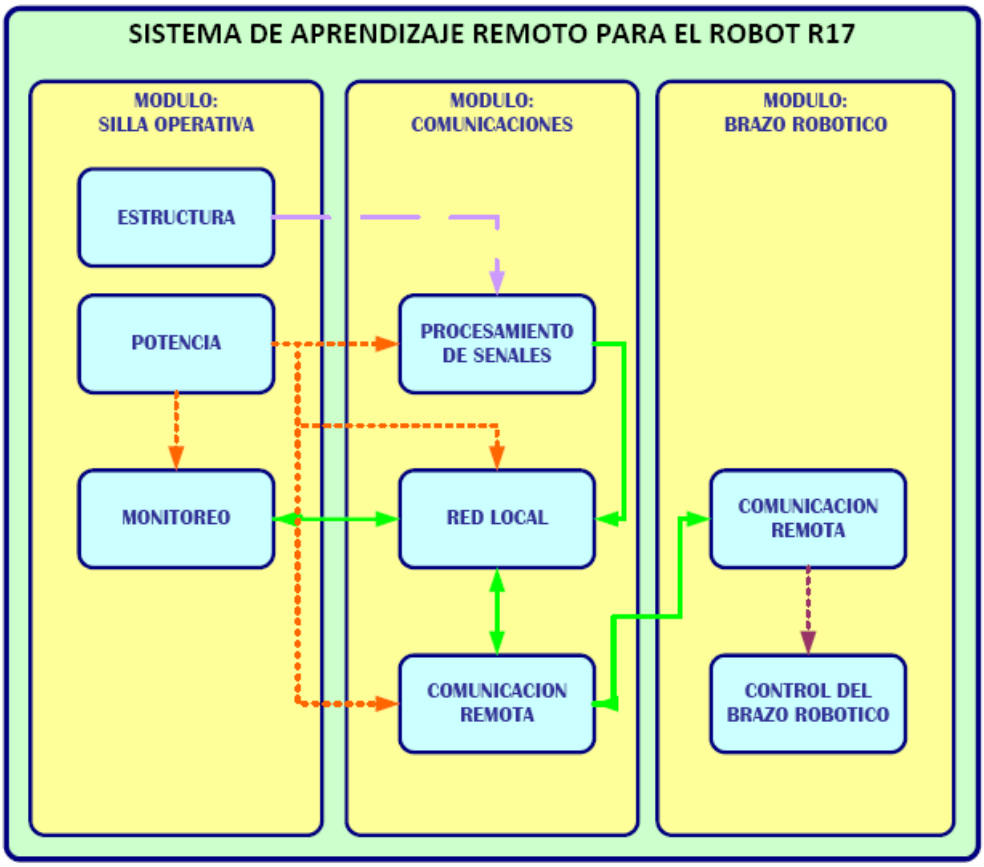
Comunicaciones

Red local

Recepción

A. Estudio de la interacción del submódulo dentro del sistema de aprendizaje remoto

El primer paso para el desarrollo del submódulo de red local fue el estudio de la interacción de este desarrollo con los demás submódulos del proyecto. En la Gráfica 27 se puede observar la distribución de módulos y submódulos dentro del proyecto y la relación que cada uno tiene con la red local.



Gráfica 37 Interacción del submódulo de red local

Con la observación global de la interacción de los submódulos, es posible describir con mayor detalle algunas dependencias del submódulo de red local.

1) Dependencia dentro del Submódulo de Red local

Dentro del Submódulo hay una dependencia total con la fase de transmisión ya que es la responsable del envío de mensajes; y con el mismo nivel de dependencia, la fase de transmisión necesita de la fase de recepción ya que es esta la responsable del filtrado para la entrega de los mensajes al cliente de la red. Esta interdependencia puede observarse en la Grafica 28.

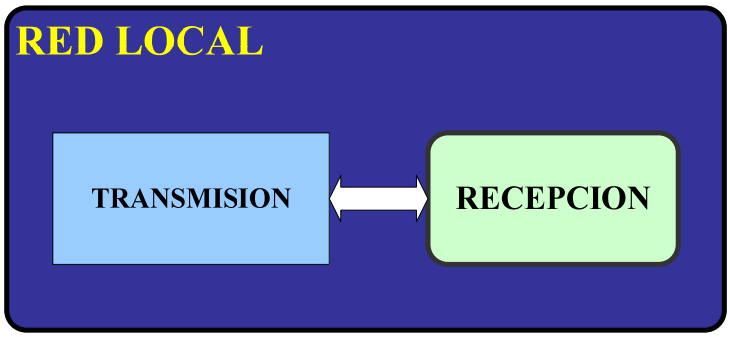
2) Dependencia dentro del modulo de Comunicación

El modulo de procesamiento de señales depende de la red local para poder comunicarse con el modulo de transmisión. Todas las señales procesadas serán enviadas a través de la red local hacia el submódulo de comunicación remota. (Gráfica 29)

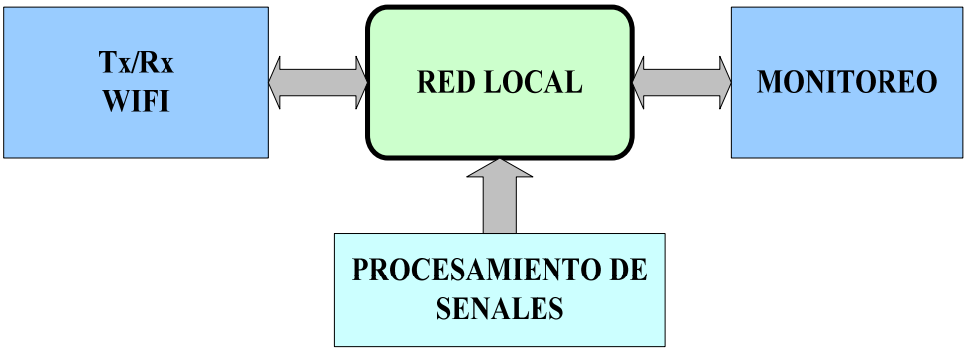
3) Dependencia dentro del Sistema de aprendizaje

Los submódulos de monitoreo y brazo robótico dependen de la red local ya que ésta es la encargada de permitir su comunicación bidireccional. (Gráfica 30)

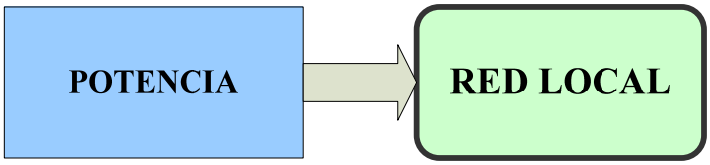
Por otro lado, la red local depende del módulo Silla Operativa ya que éste es el encargado de suministrar la potencia de alimentación. (Gráfica 27)



Gráfica 38 Dependencia dentro de la red local



Gráfica 39 Dependencias del Submódulo Red Local



Gráfica 40 Dependencia del módulo Silla Operativa

E. Estudio del protocolo de comunicación

Conociendo los detalles del tipo de comunicaciones necesarias para el proyecto, se estudiaron varios protocolos de comunicación que actualmente se utilizan en el campo industrial, entre los que cabe mencionar RS-232, I2C y CAN.

Por las ventajas ofrecidas, se escogió CAN; la principal de éstas es que dicho protocolo fue diseñado inicialmente por la industria automotriz para crear un protocolo apto para trabajar en ambientes ruidosos, es decir con mucha influencia de campos magnéticos, y por lo tanto CAN es el protocolo mas adecuado para hacer una contribución a la industria guatemalteca.

Con el protocolo establecido, se procedió a estudiar, en detalle, la especificación del protocolo; dicho estudio se realizó con materiales provistos por los fabricantes de componentes para el protocolo, principalmente Microchip.

F. Diseño de la arquitectura de la red

Al diseñar la red, se estudió cuidadosamente las prioridades que debían utilizarse ya que se requiere que los mensajes urgentes sean entregados sin demora; se consideran mensajes urgentes aquellos que indiquen cualquier falla detectada por el submódulo de monitoreo en la silla operativa, así como también son urgentes las alarmas emitidas por el submódulo de brazo robótico.

G. Diseño del funcionamiento de un nodo genérico.

Para iniciar la implementación se estudiaron las tareas que un nodo genérico debía poder realizar; es decir que sin tratar algún nodo específico se estudiaron todas las necesidades posibles que un cliente de la red podría tener. (Gráfica 31)

H. Adquisición de los materiales para la implementación de la red

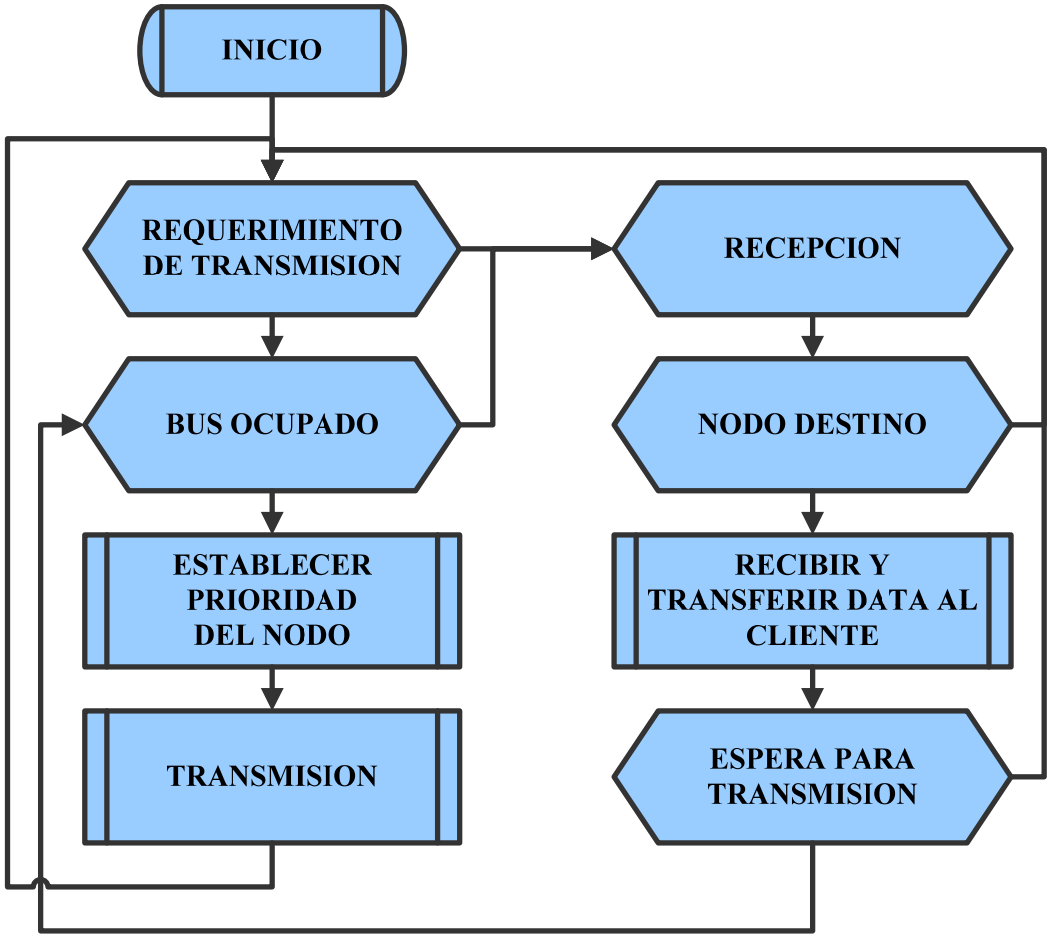
Con la arquitectura de red diseñada y con el estudio del funcionamiento del nodo genérico se buscó un microcontrolador que permitiera implementar el protocolo CAN y se encontró que el mas adecuado era el PIC18F458 de Microchip; de la misma manera, se encontró que el adaptador adecuado para el bus de comunicación

(transceiver) era el circuito integrado MCU2551, de Microchip también, por lo que se procedió a solicitar los materiales; los microcontroladores fueron comprados en el extranjero por no estar disponibles en Guatemala y se solicitaron algunos transceivers al fabricante como muestras.

I. Utilización básica del modulo CAN de los microcontroladores.

La primera implementación realizada se hizo con dos nodos únicamente para verificar el funcionamiento de los microcontroladores, transceivers y el compilador.

A partir de esta etapa del desarrollo, la fase de recepción quedó separada de la fase de transmisión porque ya que estaban definidas las especificaciones de materiales y funcionamiento.



Gráfica 41 Diagrama de flujo para un nodo genérico

J. Implementación de una red de 3 nodos.

Con los nodos operando correctamente se inició la implementación de la verificación de los identificadores y para que fuera posible clasificar los mensajes se agregó un nodo más a la red. De esa forma, se solicitó a la fase de transmisión que creara un nodo que transmitiera dos mensajes con identificadores distintos y se logró

separar los mensajes observándolos como salidas visuales (LED's) en un puerto del microcontrolador.

K. Creación de rutina para recepción

Teniendo la implementación de la recepción con separación según los identificadores, se procedió a encapsular el programa de la etapa anterior en una función de fácil acceso.

La función creada tiene el nombre canRx y utiliza tres parámetros:

1. canCounter: es un contador para la verificación de que el nodo no quede aislado de la red por falta de prioridad.
2. canByteH: registro donde se desea almacenar el byte más significativo del mensaje recibido.
3. canByteL: registro donde se desea almacenar el byte menos significativo del mensaje recibido.

L. Implementación del chequeo de identificadores basados en la prioridad de los mensajes.

Considerando los problemas que podrían ocurrir para la sincronización de prioridades y el conocimiento de dichos identificadores en la fase de recepción, se propuso el método que se describe a continuación:

- Se asignará una prioridad fija a cada uno de los nodos, en relación a las necesidades de transmisión del sistema completo.

- Se hará una estimación del mayor tiempo aceptable que un nodo pueda esperar a que le sea otorgado el derecho sobre el bus.

- Considerando que no todos los nodos desearán transmitir el cien por ciento del tiempo de operación y conociendo el tiempo máximo mencionado anteriormente, todos los nodos evaluarán la cantidad de mensajes que sean transportados por el bus sin importar el nodo al que vayan destinados.

- En caso de que el tiempo haya sido rebasado, se hará una modificación en las prioridades con lo que los nodos tomarán prioridades más altas momentáneamente para hacer la transmisión, y luego regresarán a la prioridad asignada inicialmente.

Las prioridades determinadas para el proceso anterior son las siguientes:

NODO	PRIORIDAD (HEX)	PRIORIDAD (DEC)	PRIORIDAD (BIN)
DISPLAY	0X60	96	0 1 1 0 0 0 0 0
WI-FI	0X40	64	0 1 0 0 0 0 0 0
TRACK	0X20	32	0 0 1 0 0 0 0 0
WAIST	0X10	16	0 0 0 1 0 0 0 0
SHOULDER	0X08	08	0 0 0 0 1 0 0 0
ELBOW	0X04	04	0 0 0 0 0 1 0 0
HAND	0X03	03	0 0 0 0 0 0 1 1
WRIST	0X02	02	0 0 0 0 0 0 1 0

Tabla 3 Prioridades para los nodos

Como se puede observar de Tabla 1 es posible darle prioridad máxima a un nodo con sumarle 128 (0X80 = 0B10000000) al registro de prioridad; con lo anterior, el nodo adquirirá derecho sobre el bus de comunicación inmediatamente, porque se espera que las prioridades del resto de nodos sean inferiores a 128.

Al determinar la identificación de prioridades para los nodos, se actualizó la función de recepción creada anteriormente. La función desarrollada se llama **canRx**, devuelve 1 (verdadero) en caso de que el mensaje tenga al nodo como destino, o 0 (falso) cuando no. La función tiene cuatro parámetros:

1. canNode – Identificador del nodo que hace la recepción.

2. canCounter – Contador propio de cada nodo para verificar que el nodo no sea excluido del bus por falta de prioridad para adquirir derecho sobre el bus.
3. canByte – Puntero para indicar la posición de memoria en el que se almacenarán los bytes recibidos en el mensaje.

M. Creación de la librería CAN4R17 y pruebas de chequeo de identificadores

Al finalizar la implementación de la función de recepción, con chequeo de los identificadores, se procedió a integrar la función de recepción con la de transmisión, creada en la otra fase del submódulo. Estas fueron integradas en una librería única, la cual fue probada en su totalidad en una red de cuatro nodos; esta librería es la entrega final del submódulo de red local (ver Anexos) y para su utilización, los clientes deben incluirla dentro del código de programa.

N. Implementación de la librería en los nodos de la red

Luego de pasar la etapa de verificación, se realizó la primera implementación de la red local con clientes reales; se colocaron dos nodos del submódulo de procesamiento de señales, los cuales transmitían los datos de posición de dos sensores distintos y la información era entregada a un cliente receptor que, al mismo tiempo, transmitía los valores recibidos a una computadora utilizando el protocolo serial RS-232.

A. Manejo de errores

Con respecto a los errores, el modulo CAN del PIC 18F458 los maneja con los métodos de detección de errores CRC, Acknowledge error, Form error, Bit error y Stuff bit error. Esto lo hace de forma automática. Los errores detectados son hechos públicos gracias a los bloques de errores generados. La transmisión de mensajes erróneos es abortada y el bloque se retransmite lo más rápido

posible. De hecho, cada nodo puede estar en diferentes estados dependiendo de los contadores de errores. Existen dos contadores de errores, uno de transmisión y el otro de recepción. La figura No.2 ilustra los estados de errores posibles para los nodos. Como vemos, existen tres modos, el error active es el modo natural. En el Bus-off el nodo no puede recibir ni transmitir hasta regresar al modo error active, para esto se debe de cumplir lo indicado en la figura No.2. Por otro lado, el buffer de recepción permite varios tipos de recepción, incluyendo la de todos los mensajes o bien la recepción de solo mensajes validos, es decir mensajes sin error.

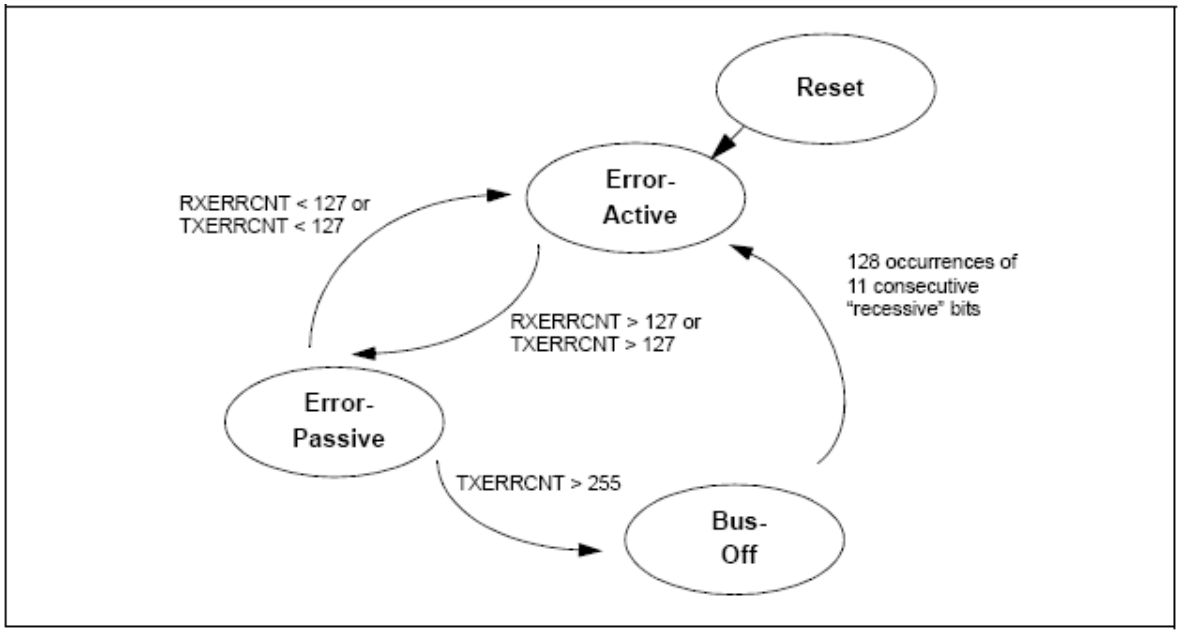


Figura 11. Diagrama de estado de modo de errores

Para esto, en la rutina de inicialización se incluyó el código siguiente: *RXB0CON.rxm=CAN_RX_VALID;*

RXB1CON.rxm=CAN_RX_VALID;

el cual inicializa los buffers para que reciban solamente mensajes sin errores.

De esta fase de desarrollo, recepción en la red local, se obtuvo como resultado la correcta recepción de los mensajes; es decir que se consiguió detectar, validar y filtrar los mensajes que fueran destinados para cada uno de los nodos.

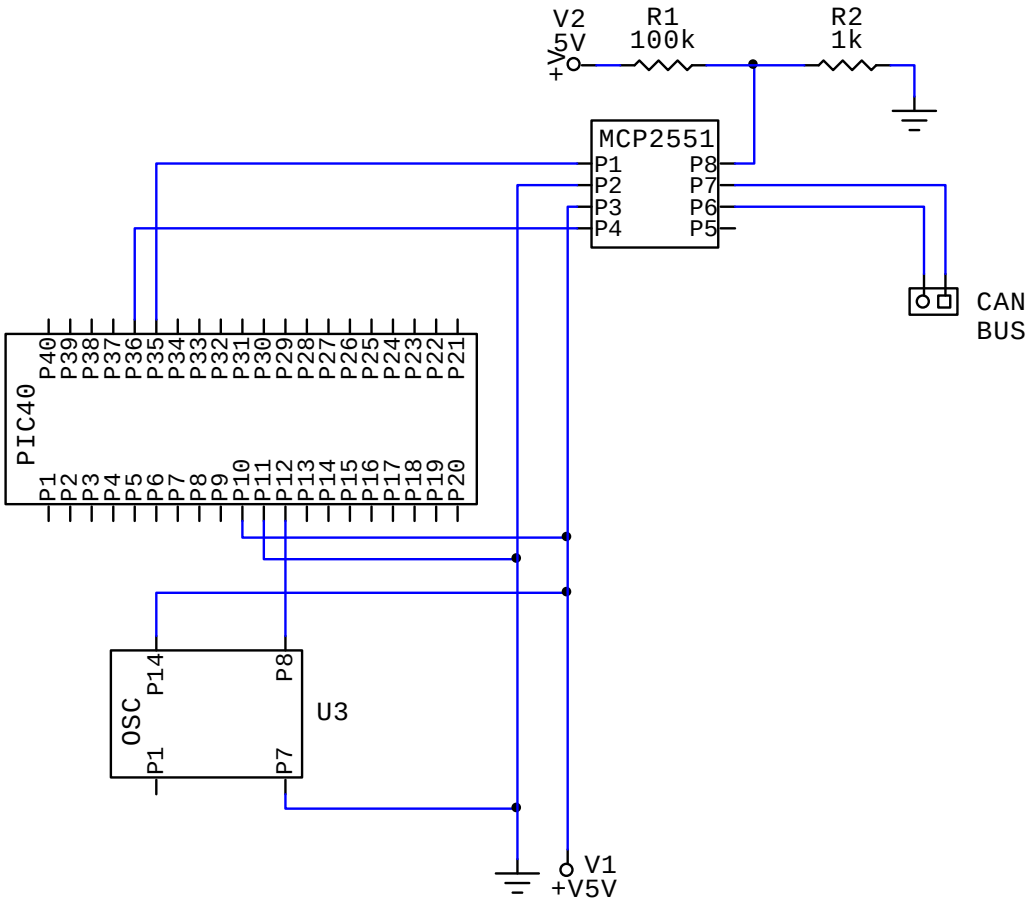
Esta implementación se hizo en una librería de funciones, llamada CAN4R17, para que pueda ser entregada de manera sencilla a los clientes de la red y evita que esos clientes tengan que verificar directamente cada uno de los mensajes que circulan por el bus de datos.

Finalmente, el submódulo hace una entrega única. Se entrega una sola librería a los clientes porque desde el punto de vista de facilidad de utilización, es necesario que las dos fases que conforman el submódulo trabajen en conjunto

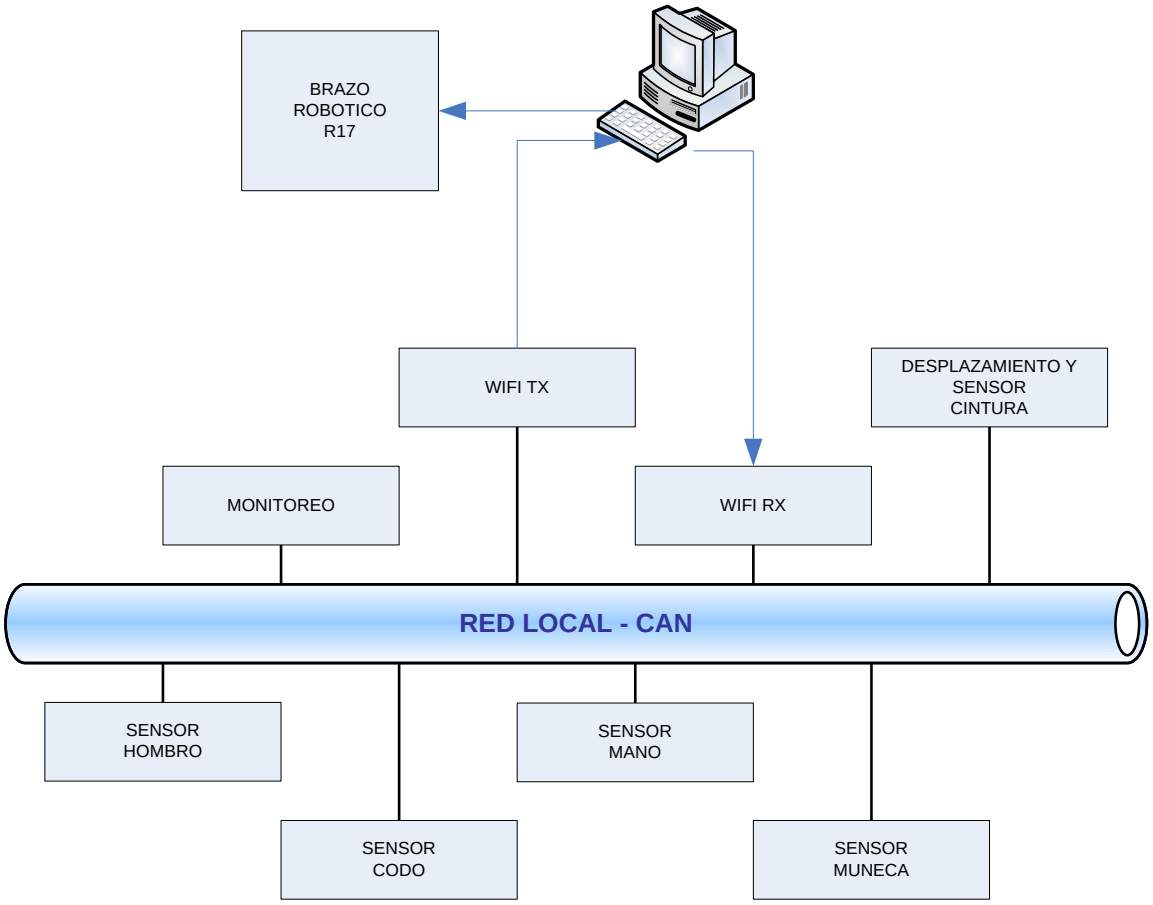
Para la utilización de la librería CAN4R17 se sugiere utilizar el circuito mostrado en la Gráfica 32.

Aunque inicialmente se pensó en la creación de una red de servicio para el sistema de aprendizaje remoto para el robot R17, la librería permite que la arquitectura del sistema sea modificada fácilmente; en la Gráfica 33 se presenta un diagrama de la arquitectura sugerida para la red local.

En los anexos del presente trabajo se encuentra el código de programación de la librería entregada.



Gráfica 42 Circuito para utilizar la librería CAN4R17



Gráfica 43 Arquitectura sugerida para la red local

4) red WiFi

Recepcion

Diseño del Circuito

El diseño del circuito se inició identificando los componentes que serían necesarios para que el WiPort funcione adecuadamente. También se tuvieron que incluir elementos necesarios para la red CAN, así como elementos para el monitoreo del WiPort.

El WiPort b/g es un dispositivo inalámbrico que permite convertir una señal serial RS-232 a una trama TCP, y enviarla a través de una red inalámbrica 802.11b/g. Esta conversión a una trama enviada por 802.11b/g facilita la integración de la silla de mando, ya que nos permite cierta autonomía y no dependemos de cuan largo es el cable de conexión. Siendo 802.11b/g un estándar muy utilizado a nivel internacional, el WiPort facilitará la integración a redes ya existentes, y permite mayor flexibilidad, ya que se pueden enrutar los paquetes que envía el WiPort a través de Internet, y llegar a casi cualquier punto del mundo.

Figura 12. Vista del WiPort b/g (Lantronix)

El WiPort funciona a 3.3V, y la alimentación de la silla operativa es de 5V. Por esta razón se procedió a buscar un regulador de voltaje que proveyera 3.3V de una forma constante, y que no variara al aplicarle más carga. Para esto,

se seleccionó el regulador de voltaje LM2576. Este es un regulador ajustable que provee de 3A a 3.3V constantes, y sin variación en el voltaje de salida al aplicarle más carga. Este regulador también tiene la ventaja que requiere pocos componentes externos para funcionar, haciéndolo simple y sencillo de construir.

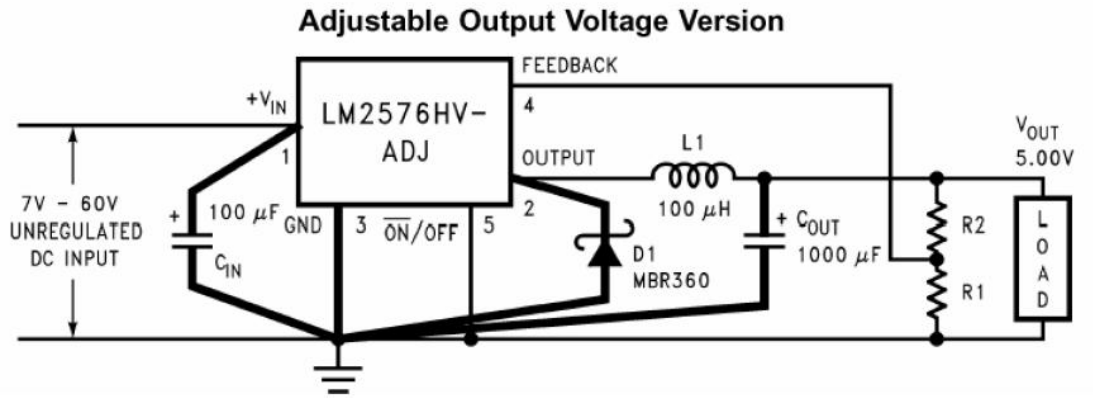


Figura 13. Circuito para regulador LM2576-ADJ

Otro de los componentes que eran necesarios para el WiPort eran los dos micro controladores de Microchip PIC. Uno de ellos se emplea para la recepción de datos desde la computadora a través del WiPort. Para esto se escogió el PIC 18F458, por ser de alta velocidad, y también por poseer un puerto Transmisor Receptor Serial Universal Síncrono/Asíncrono (USART, por sus siglas en inglés), y un transmisor receptor CAN incorporados. Estos dos elementos son esenciales, ya que el USART es necesario para comunicación serial asíncrona con el WiPort, y CAN es necesario para la comunicación con el resto de componentes de la silla operativa, a través del bus.

El PIC necesita pocas conexiones externas. Las esenciales son 5V y tierra, así como su reloj a base de cristal resonador a 40 MHz. Se escogió esta frecuencia, ya que daría al PIC una alta velocidad de ejecución de instrucciones y de procesamiento de datos. Las conexiones entre el PIC y el WiPort fueron solamente las del puerto USART, y las de las señales de error. Entre el puerto USART del PIC y el serial del WiPort no se necesitó ningún ajuste de nivel, ya que el WiPort es tolerante de niveles de 5V en sus entradas, y el PIC también detecta 3.3V como un 1 lógico, dado que su puerto es de niveles TTL. Las

conexiones a los pines de monitoreo de estado fueron 2. Estas conexiones se hicieron al puerto B del PIC, específicamente al bit 0 y 1, para poder manejar los errores a base de interruptos, y así poder contar la frecuencia de las señales, ya que estas son generadas por el WiPort, variando la frecuencia con la que ocurre cada pulso dependiendo del error.

Adicionalmente a la conexión entre el PIC y el WiPort, también fue necesario agregar una conexión entre el PIC y un transmisor/receptor CAN. Este transmisor/receptor se necesita para poder convertir los niveles TTL que utiliza el PIC a niveles de voltaje correspondientes al estándar de la red CAN. La red CAN es utilizada para comunicar todos los nodos de los sensores y el módulo de monitoreo entre si, y así lograr una comunicación rápida y efectiva.

Componente	Cantidad
WiPort	1
PIC 18F458	2
Transmisor/Receptor CAN	2
Cristal Resonador 40MHz	1
Regulador de Voltaje LM2576	1
Push Buttons	4
Resistencias Variadas	11
Indicadores LED	4

Capacitores Variados	2
Diodos	1

Tabla 4. Listado de materiales del circuito impreso

Construcción del Circuito Diseñado

El circuito diseñado en MultiSim fue construido en un protoboard, por las ventajas que da este a la hora de realizar pruebas. Las ventajas principales que se aprovecharon son la facilidad de cambiar el circuito armado en caso de que se necesitara una modificación al diseño, y la versatilidad a la hora de construir los diseños, ya que se puede implementar casi cualquier circuito en el.

En el circuito se puso el regulador de voltaje, los dos PICs con su cristal resonador a 40 MHz correspondiente, y el WiPort. El WiPort, dado que tiene una conexión de 40 pines “surface mount”, se debió soldar un adaptador especial. Este adaptador fue unido a unos alambres calibre 24, los cuales facilitaron la conexión al protoboard.

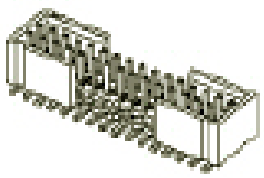


Figura 14. Conector de 40 pines WiPort

Cuando se finalizó de armar el circuito en el protoboard, se pudo proceder a comenzar las pruebas de transmisión.

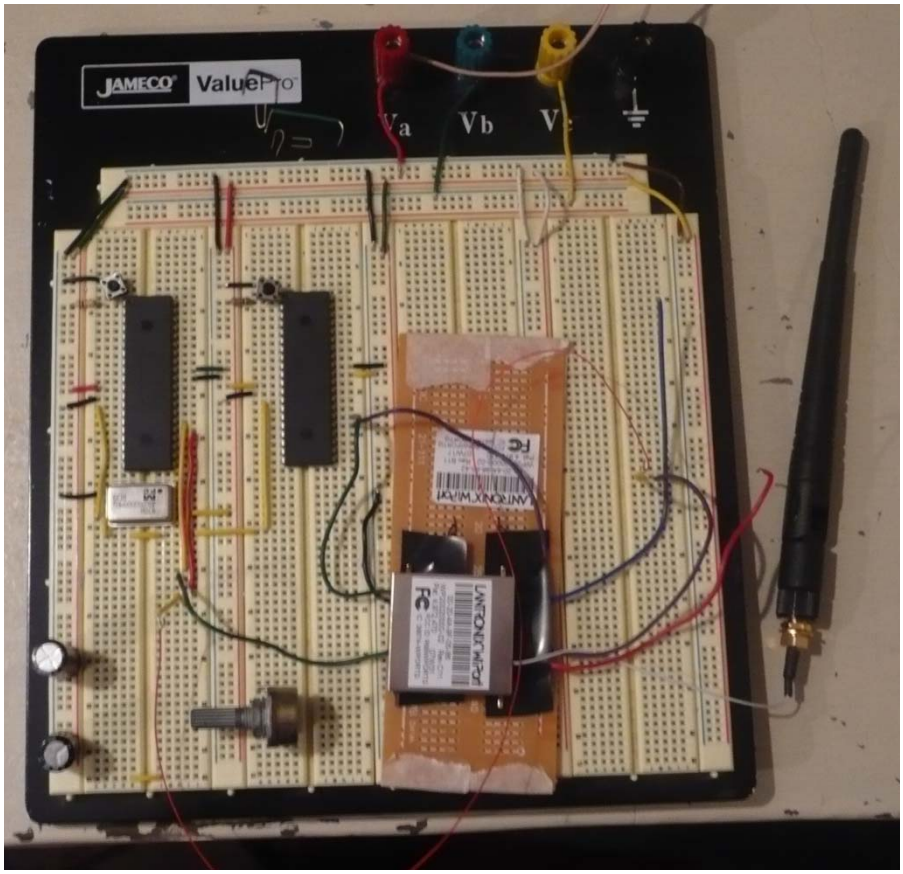


Figura 15. Circuito WiPort armado en protoboard

Pruebas del WiPort

La conexión se hizo por medio de una conexión Ad-Hoc, es decir, directamente entre los dos equipos. La dirección IP del WiPort es 169.254.10.1, pero esta puede ser cambiada para que tenga cualquier valor, así como puede ser conectado el dispositivo a una red con infraestructura (con un Access Point) para poder ser accedida desde cualquier equipo conectado a la red.

Figura 16. Pantalla principal de configuración del WiPort

Desde la pantalla principal se puede hacer cualquier cambio a la configuración del dispositivo. Dado que las configuraciones que eran necesarias cambiar eran las de los puertos seriales, se procedió a la página para cambiar estos.

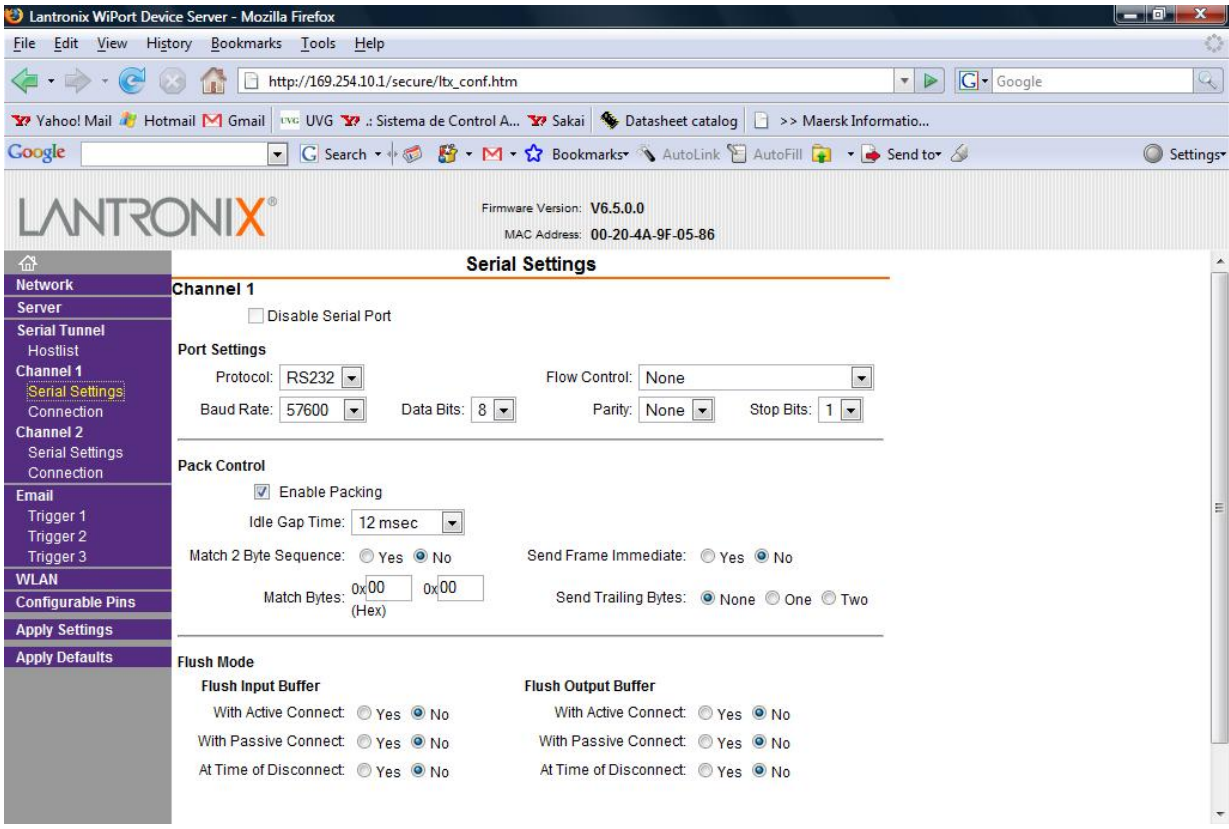


Figura 17. Pantalla de configuración Serial

En esta pantalla se pueden cambiar todas las opciones del puerto serial. En esta ocasión, se establecieron estos parámetros en RS-232, a una velocidad de 57600 bits/segundo, 8 bits de datos por palabra, ninguna paridad, ningún control de flujo, y 1 bit de parada. Con estas configuraciones se procedió a probar el puerto serial del WiPort.

Pruebas del Puerto Serial

El puerto serial del WiPort se procedió a probar primero con una conexión del PIC a la computadora, por medio de un adaptador de serial a USB. Esto se realizó para poder verificar que el software de prueba desarrollado funcionara adecuadamente. Cuando se verificó esto, se pudo proceder a realizar la prueba de interfaz con el WiPort.

Las primeras pruebas se hicieron en el puerto serial 0 del WiPort, a 9,600 bits/s. Estas pruebas fueron a una baja velocidad para asegurar que la conexión entre ambas fuera correcta. Para todas las pruebas de transmisión se hizo un simple "ECHO" de los datos enviados, es decir, se recibían en el PIC, y después se reenviaban de regreso al origen. Para esta prueba se enviaron únicamente caracteres ASCII que se escribían en la HyperTerminal, utilizándola con conexiones TCP/IP.

Para la conexión entre el WiPort y la PC se decidió utilizar el protocolo TCP/IP, ya que es un protocolo orientado a conexiones y que asegura que los datos enviados sean recibidos correctamente. Esto es necesario ya que los datos que están siendo enviados a través de esta conexión deben ser entregados de forma correcta, ya que son datos de posiciones que deben ser muy precisas. La orientación a conexiones nos permite que el servidor sepa que esta conectado a la silla, y que debe esperar datos de ella, así como enviar palabras de estado al módulo de comunicaciones.

Cuando se verificó el funcionamiento adecuado básico del WiPort se procedió a incrementar la velocidad del serial. Dado que la transmisión de los datos debe ser lo más rápida posible, se decidió intentar una transmisión a 115,200 bits/s. Para iniciar esta prueba se comenzaron a enviar caracteres ASCII individuales, escritos desde la HyperTerminal. En este caso funcionaba adecuadamente, haciendo el "ECHO" sin errores. A continuación, se procedió a enviar un

archivo de texto, el cual contenía texto variado escrito. Al enviar esto nos dimos cuenta que muchos de los caracteres devueltos no eran correctos, o incluso que algunos simplemente nunca llegaban. Por este hecho se pudo deducir que la velocidad de transmisión era demasiado alta para el PIC, y que este no había terminado de procesar algunos datos, y ya le estaban llegando más, lo cual causaba que información se perdiera. (Ver Archivos de Texto en Anexo)

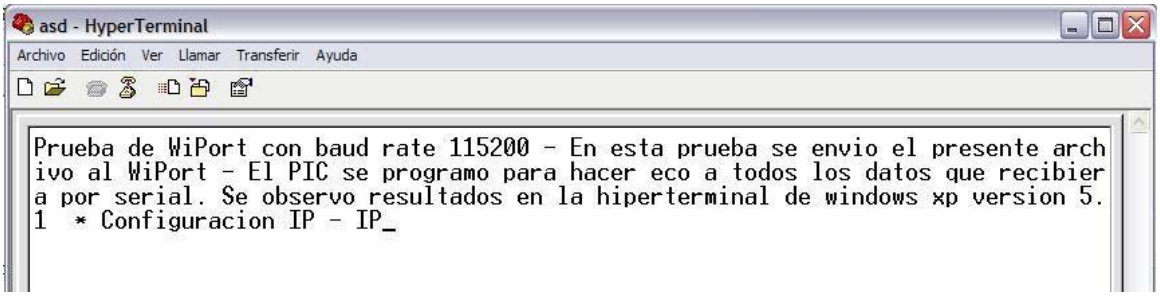


Figura 18. Resultado de transmisión de prueba a 115,200 bits/s

Como se pudo apreciar que no llegó toda la información que se requería de regreso a la PC, se procedió a bajar la velocidad de transmisión. La siguiente velocidad más baja a la que el WiPort puede funcionar es de 57,600 bits/s. Para esto se reprogramó el PIC para funcionar a una velocidad de 57,600 bps, y se conectó. La prueba se desarrollo de la misma manera, conectando el WiPort por TCP/IP a la PC, y enviando un archivo de texto. En esta ocasión se recibió de forma adecuada el archivo, ya que todos los caracteres que se habían enviado se recibieron de regreso.

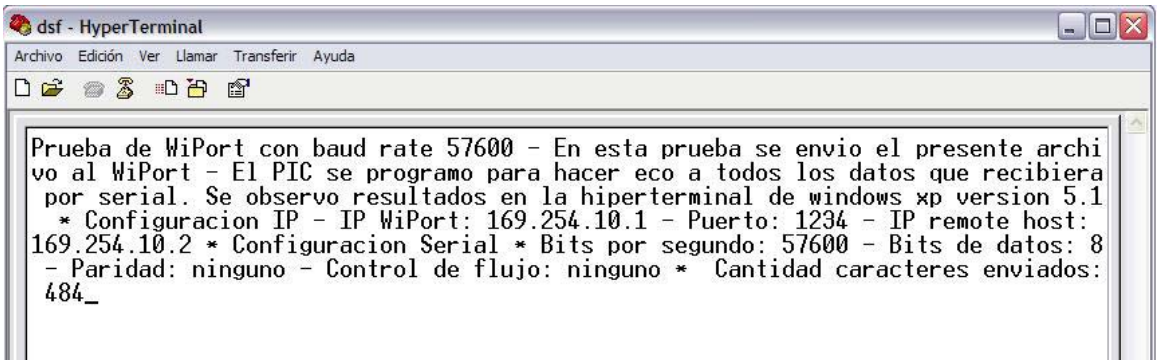


Figura 19. Resultado de transmisión de prueba a 57,600 bits/s

Para fines de asegurar que el WiPort estuviera funcionando de forma adecuada, se hizo una última prueba a 38,400 bits/s. Como en las dos pruebas anteriores se procedió a enviar un archivo de texto desde la HyperTerminal hasta el WiPort, y ver que este fuera replicado adecuadamente.

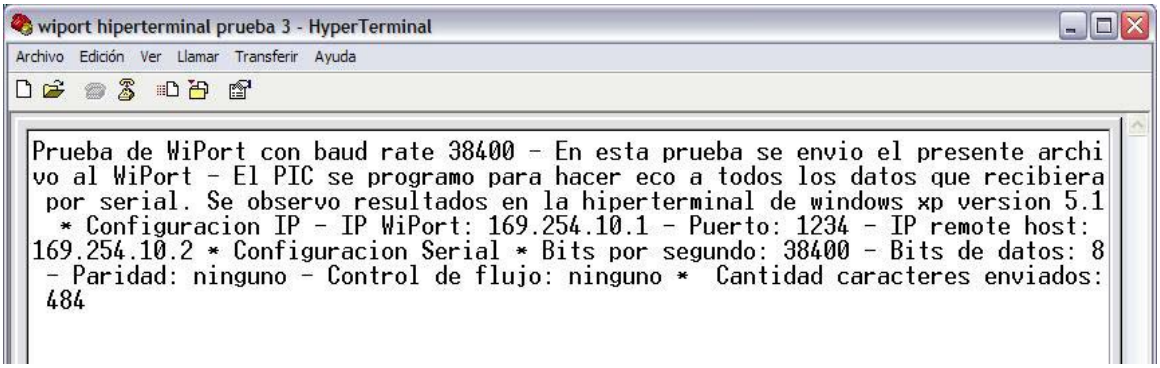


Figura 20. Resultado de transmisión de prueba a 38,400 bits/s

Lo más importante a tener en mente en este desarrollo es la transmisión confiable de datos, a la velocidad más alta posible. Como se puede ver en los resultados de las transmisiones, la velocidad que cumple estos requisitos es la de 57.6 kbps. Esto es porque a esta velocidad se logran enviar y recibir la información a una

velocidad que es lo suficientemente alta para que no hayan retrasos, así como lo suficientemente baja para que el PIC la pudiera procesar toda de manera adecuada. Por este motivo se decidió dejar 57.6 kbps como la velocidad de transmisión para el módulo de recepción.

Comunicación de Mensajes de Monitoreo

Una vez comprobado el desempeño del WiPort, y habiendo fijado su velocidad de transmisión a 57,600 bps, se procedió a incorporar el WiPort con el módulo de Monitoreo. Para eso se incorporó la librería de CAN, la cual se encargaba de transmitir y recibir datos del bus CAN. Una vez incluida esta librería, se pudo comenzar a desarrollar el software necesario para la transmisión y recepción de datos hacia el módulo de Monitoreo.

La transmisión consiste en recibir datos del WiPort, empaquetarlos en grupos de 2 bytes, y enviarlos a través del bus CAN. La recepción consiste en recibir datos a través del bus CAN, procesarlos y después enviarlos a la PC, por medio del WiPort, si es necesario.

El procesamiento que se le da a los datos en la recepción de los mismos es de analizar si es un dato para la PC o un dato para el micro controlador. Cuando son datos para la PC se procede simplemente con enviarlo a través del WiPort. Ahora bien, cuando es un dato para el micro controlador, se procede a decodificarlo, y responder adecuadamente. Existen dos posibles comandos que pueden ser enviados por el módulo de monitoreo hacia el micro controlador. Estos son solicitud de estado, y solicitud de dirección IP.

Comando	Código
Solicitud de IP	133

Solicitud de Estado	134

Tabla 5. Comandos recibidos de Módulo de Monitoreo

Comando	Código (Dato 0)	Código (Dato 1)
Estado del WiPort	106	00000+Error General + IP Duplicada + Antena ON/OFF
IP Byte 1	107	Byte más significativo (Decimal)
IP Byte 2	108	Byte 2 (Decimal)
IP Byte 3	109	Byte 3 (Decimal)
IP Byte 4	110	Byte menos significativo (Decimal)

Tabla 6. Comandos de respuesta para Módulo de Monitoreo

Cuando se recibe un comando de solicitud de IP, este es procesado por el micro controlador y se envía una respuesta. La respuesta consiste en 4 paquetes de 2 datos cada uno, conteniendo cada uno un identificador (Dato 0) y datos de respuesta (Dato 1), como se observa en la tabla 4. La IP del sistema es guardada en la memoria EEPROM del micro controlador, ya que esta es no volátil, dando la ventaja que no se pierda la información. Cuando se va a transmitir la IP, se obtiene la misma de la memoria EEPROM, y se empaqueta junto con su identificador en el Dato 0. Se envía en el orden de el Byte más significativo primer, y después el siguiente más significativo, y así sucesivamente los 4 datos.

Cuando se recibe un comando de solicitud de error, se regresa al módulo de Monitoreo el dato que contiene la información de errores, como se observa en la Tabla 4. Este dato nos indica en caso que haya un error en alguno de los sistemas del WiPort, para que sea desplegado por Monitoreo.

Cualquier otro dato recibido por el micro controlador del Módulo de Monitoreo es enviado a la PC, para que este sea procesado allí. El código se intenta que sea lo

más rápido posible, para evitar problemas de pérdida de datos por respuesta lenta del micro controlador.

Brazo Robotico

5) . INVESTIGACIÓN DE ROBOFORTH

Este estudio del lenguaje Roboforth se realizó utilizando el programa Rovowin2 que el fabricante ST Robotics sugiere para programar el brazo robótico.

Se realizaron una serie de pruebas en las que se pretendía conocer el lenguaje de programación Roboforth. Lo que se pretendía con dichas pruebas era aprender a programar con dicho lenguaje. Como resultado de estas pruebas reconocieron algunas de las instrucciones más importantes dentro del mismo.

Se identificaron las instrucciones que podrían utilizarse para el movimiento del brazo robótico tal y como se necesitaba para este Mega Proyecto.

Se tomó de base para la aplicación un conjunto de instrucciones que se que presentan en el cuadro No. 1.

- Nombre
- Descripción

• START	• Establece la comunicación entre la computadora, el controlador, y el brazo robótico.
• TELL (join) Dato MOVE	• Le indica a la unión cual posición moverse en posiciones absolutas, mueve solamente un unión.
• CALIBRATE	• Calibra los ejes, y las uniones en sus posiciones absolutas en 0.
• (n) SPEED	• Le indica al brazo robótico en que velocidad trabajar.
• DECIMAL Track Mano Muñeca Codo Hombro Cintura JMA	• Mueve a la posición absoluta de cada uno de las uniones.
• ENERGIZE	• Proporciona energía al brazo robótico
• DE-ENERGIZE	• Desenergiza al brazo robótico
•	•

Tabla No. 1. Cuadro de instrucciones utilizadas de Roboforth

En esta fase además se estudió la posibilidad de programar desde Robowin la aplicación final para este Mega Proyecto, paralelo a esto se estudió la posibilidad de crear la aplicación en otro lenguaje de programación.

Los beneficios de crear la aplicación final en Robowin son:

- Robowin contaba con una interfaz directa para crear programas para el brazo robótico.
- Robowin permitía crear aplicaciones, guardarlas y ejecutarlas posteriormente.
- Robowin posee un manejo directo con el puerto serial que es entendido por el controlador del brazo robótico.

Las desventajas de crear la aplicación final en Robowin son:

- Robowin no permite tener una interfaz directa entre un entrada externa al la aplicación.
- Robowin no permite crear un proyecto al mismo tiempo que esta se está ejecutando.

Se evaluaron las opciones y se llegó a la conclusión que desde un lenguaje de programación como por ejemplo C# es posible tener una interfaz directa entre una entrada externa, y ademas es posible crear un proyecto al mismo tiempo que este se ejecuta. Además las ventajas que provee Robowin se pueden implementar en C# por ejemplo. En cambio en Robowin no es posible cumplir estos requerimientos.

6) B. CREACIÓN DE PROTOCOLO DE COMUNICACIÓN SILLA OPERATIVA - COMPUTADOR

En la siguiente parte del trabajo se realizó un protocolo con el cual se pueda obtener una correcta comunicación entre la silla. Para esto se trabajó de manera conjunta con el módulo de monitoreo, y de CAN.

Lo que se discutió principalmente fue una forma en que se pudiera comunicar la silla operativa con la aplicación. El resultado es protocolo DeucaTalk.

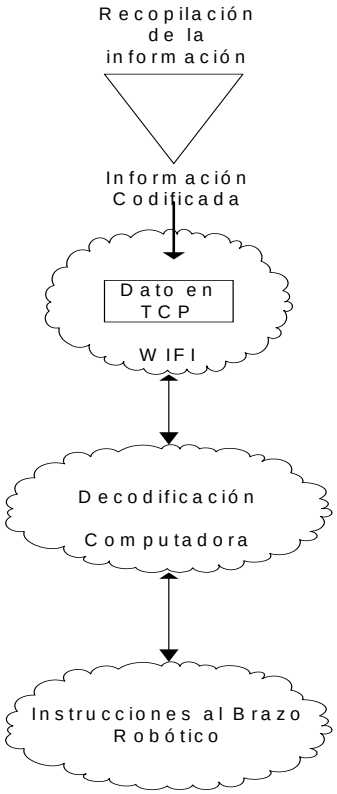


Grafico No. 1 Flujo de la Información

La figura No. 1 nos muestra el flujo de la información desde que esta es capturada por las uniones, pasando por una fase de codificación atravez del protocolo DeucaTalk, y por último, se convierten estas en instrucciones para Deucalion R17.

El resultado esta recopilado en las tablas 2 a 6:

MONITOREO, ENCODERS		a	Wiport,PC
	ID	DATA	DATA2
	4	4	8
		Bit más significativo	Bit menos significativo
Waist	0	Información de la absoluta	Coordenada, posición
Shoulder	1	Información de la absoluta	Coordenada, posición
Elbow	2	Información de la absoluta	Coordenada, posición

Wrist	3	Información de la Coordenada, posición absoluta
Hand	4	Información de la Coordenada, posición absoluta
Track	5	Información de la Coordenada, posición absoluta
Grip	6	Información de la Coordenada, posición absoluta
Monitor	7	Información de la Coordenada, posición absoluta
Wiport	8	Información de la Coordenada, posición absoluta

Tabla No. 2. Manejo de instrucciones

La tabla No. 2 nos proporciona la codificación utilizada para la obtención de la información por parte del módulo brazo robótico. Lo que se hizo básicamente es utilizar 4 bits del primer byte para indicar a cuál codificador, brújula, o acelerómetro se va a referir la información que contendrá en los siguientes 12 bits.

Esta estructura de los siguientes 12 bits sirve básicamente para estos dispositivos, que recogen las posiciones absolutas de la silla operativa, en el caso de monitor, y de Wiport, (7,8), existen aún más especificaciones que son las que se presentan en el cuadro No. 3

PC	ID (4bits)	Dato (12bits)	Información sobre la operaciones de monitoreo
Handshake	7	1	Sirve para iniciar la comunicación.
Request Online	7	2	Hacer una petición, para ver si se esta en línea.
Request Hour	7	3	Hace una petición para enviar la hora.
Play	7	4	Envía una señal de inicialización del modo play.
Stop	7	5	Envía una señal de stop, detiene la aplicación y no guarda las últimas coordenadas.
Pause	7	6	Envía una señal de pause,

			detiene la aplicación, guardando las últimas coordenadas.
Rec	7	7	Envía una señal de record, inicializa la aplicación y comienza a guarda coordenadas.
Izq Waist	7	8	Envía una señal para que gire la cintura 90 grados hacia la izquierda.
Der Waist	7	9	Envía una señal para que gire la cintura 90 grados hacia la derecha.

Tabla No. 3. Codificación de información para monitoreo

El Sub -Modulo de Monitoreo no necesita enviar posiciones, ya que este se encarga de transmitir instrucciones a realizar, por eso cuenta con una especificación distinta a la de los otros. La tabla No. 3 recopila la forma en que se transmitirá la información al computador.

Byte	MSB	LSB
Límite Precaución	10	00 + 6 bits de Articulaciones
Límite Deshabilitado	11	00 + 6 bits de Articulaciones
Unidad Minutos	20	Dato en decimal
Decena Minutos	21	Dato en decimal
Unidad Hora	22	Dato en decimal
Decena Hora	23	Dato en decimal
AM/PM	24	1 PM 0 Am
Acknowlege PC en línea	25	--
Handshake con PC	26	--
Estado de Grabación	27	1-Play, 2-Stop, 3-Rec, 4-Pause (en Decimal)

Tabla No. 4. Retroalimentación PC – Monitoreo Codificación de mensajes

El cuadro No. 4, nos presenta la forma en que el computador regresará la información al módulo de monitoreo. Contando de 1 a 8 del bit menos a al más significativo, la posición 1 es la cintura, 2 el hombro, y en orden siguiente codo, mano, muñeca, y el riel es el último o bit 6.

En otra fase del desarrollo, se calibraron los límites tomando en cuenta la cantidad de pasos que puede dar cada una de las uniones, cada una de las uniones se muestran en la figura No. 2. De este análisis se realizó el cuadro No. 5.

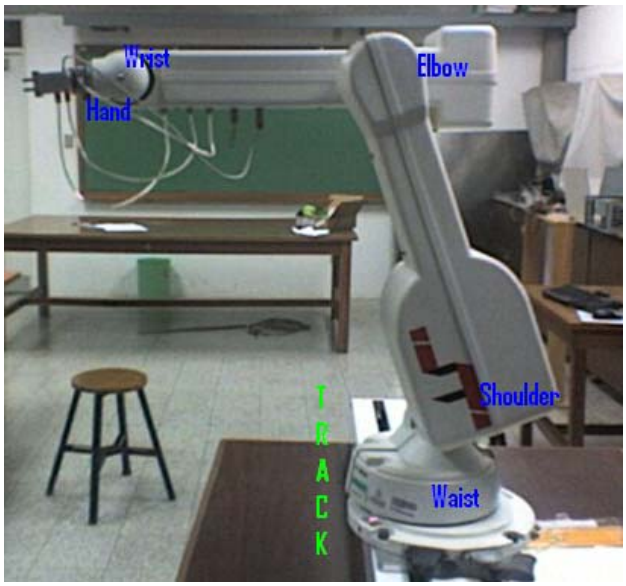


Gráfico No. 2 Diagrama de R17 con articulaciones y sus nombres.

Track	25000
Waist Izq	19600
Waist Der	19600
Shoulder Adelante	22500

Shoulder Atrás	22500
Elbow Delante	15000
Elbow Atrás	15000
Wrist Derecha	11000
Wrist Izquierda	11000
Hand Derecha	3500
Hand Izquierda	3700

Tabla No. 5. Límites de las uniones

7) APLICACIÓN

La aplicación fue desarrollada en el lenguaje de programación C#. Se escogió este lenguaje, ya que existe una buena documentación de otros procesos que se necesitaron para la creación de la actual aplicación, por ejemplo del uso del puerto serial, o bien del uso de WiFi.

La aplicación se creó como un modelo incremental, primero se desarrolló el manejo de cada una de las uniones, es decir como se iban a ejecutar los movimientos desde cada una de las uniones, esto se creó para el lanzamiento del Mega Proyecto. Seguido de este punto se desarrolló una modalidad controlado completamente desde la aplicación, las partes de esta aplicación fueron la calibración del robot, y el manejo de las uniones desde la aplicación.

En la siguiente etapa se creó una sección que mostrara la capacidad del brazo, a través de una rutina guardada previamente. Esto se hizo guardando las posiciones en donde se quería que el brazo robótico estuviera. Esta información se guardo como secuencia de posiciones.

Se creó una sección en la cual el usuario podía crear una secuencia, y además podía guardar dicha secuencia para ser ejecutada posteriormente. Para esto se utilizó una pila de posiciones que el usuario definia con la interfaz de cambio de posición.

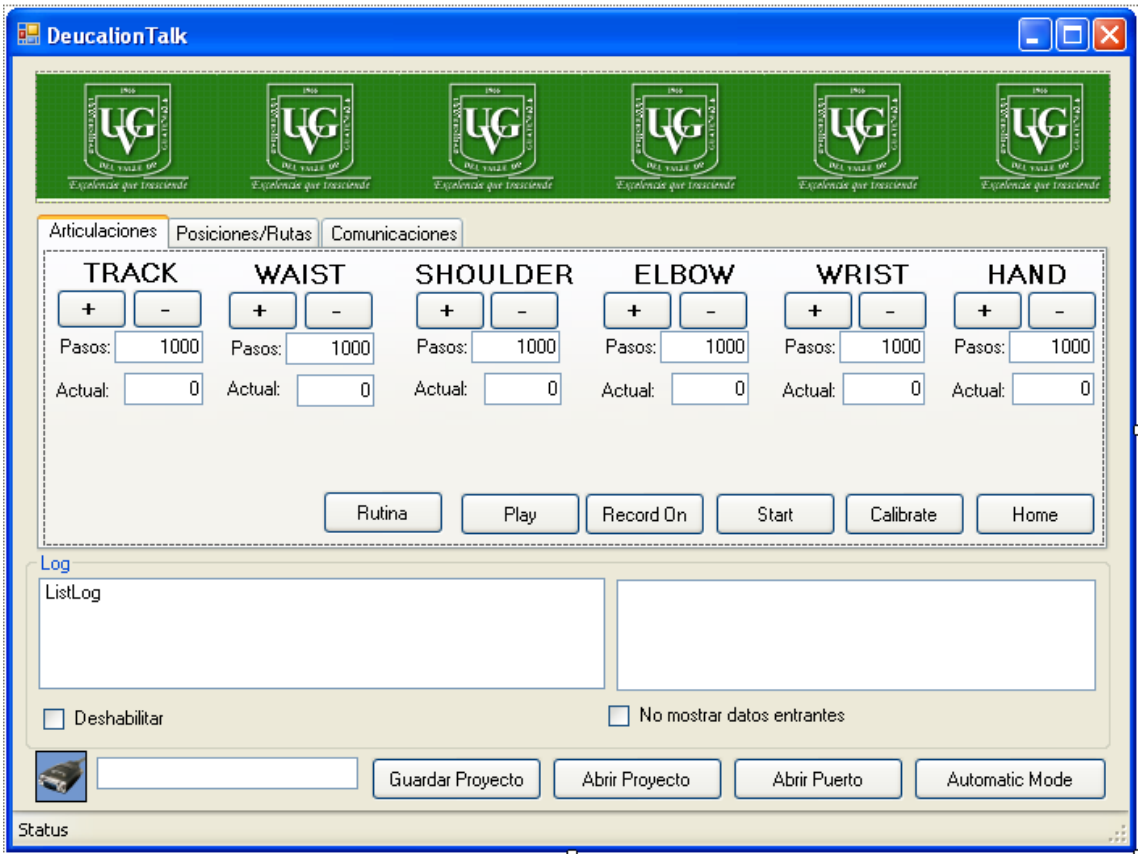


Gráfico No. 3 Fotografía de la aplicación, modo edición.

Luego se creó una sección con la que es posible guardar las rutinas hechas, esto con el fin de cargarlas posteriormente y ejecutarlas. Además se creo la modalidad automática con la que es posible recibir de una lista de instrucciones, enviadas, desde la silla operativa, y recibidas por medio de WiFi. Con dichas instrucciones se cuenta con el reconocimiento de las mismas basadas en el protocolo DeucaTalk expuesto en las tablas de la 2 – 6. Cuando se han computado las coordenadas enviadas se genera una posición, la cual es enviada al controlador, y si algún eje esta cerca de su límite envía una bandera de

proximidad, para dicha unión y si esta fuera de rango, no se mueve sobre ese eje, y envía una bandera de fuera de rango para dicha unión.

La aplicación que se tiene es capaz de almacenar una secuencia de instrucciones previamente grabadas, así como ejecutarlas, se escogió guardar el proyecto en xml, por la facilidad de uso. Este es el paso preliminar a la obtención de dichas instrucciones directas de los codificadores.

Teniendo esta información, el computador será el encargado de convertirla en instrucciones y coordenadas que el robot pueda manejar. Una vez estas instrucciones se realicen, se podrá contar con retroalimentación de las actividades hechas por la aplicación.

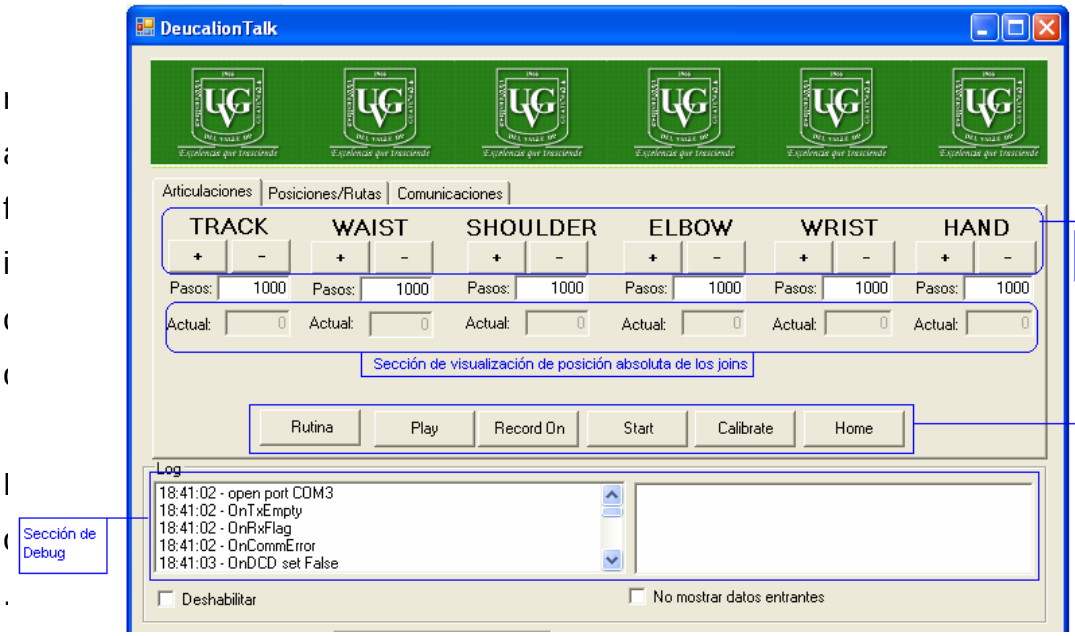
En la gráfica No 4, se encuentran unas etiquetas, las cuales se utilizan para representar secciones. Una de ellas es la sección de cambio de posición de la unión. La idea es que por medio de esta sección se pueda cambiar la posición de cualquiera de las uniones alguna cantidad de pasos.

La sección de visualización de posiciones absolutas de las uniones sirve para saber en qué posición se encuentra el brazo robótico, además de cumplir con un doble propósito, pues son estas las que determinan si una unión está cerca de un límite. La sección de visualización de la información sirve para saber si ha ocurrido algún error, almacenando esto también en una cola de mensajes, el cual hará las veces de una bitácora de errores, pero también dirá si todo marcha bien. Existe también la sección de manipulación de acciones, que es la encargada de indicarle al brazo robótico qué acción realizará.

8) D. SECCIONES DE LA APLICACIÓN

Cada una de estas secciones tiene una relación con la aplicación que esté conectada a la silla operativa, por ejemplo, en el caso de la sección de movimiento, se puede mover unión por unión. Sin embargo, para el manejo de la silla operativa, se tiene contemplado la utilización de posiciones por lo que en esta modalidad esta sección será deshabilitada. Esto además es por que es mucho más rápido mover a una posición que por separado cada unión.

G



4 Fotografía de la aplicación, documentando sus secciones.

L

a sección de visualización, tiene como finalidad saber en qué posición se encuentra el robot y de esta forma se pueda movilizar el brazo robótico a dicha posición.

La sección de manipulación de acciones es una de las secciones que puede ser cubierta por comandos enviados desde la silla operativa, por ejemplo el botón RecordOn de la gráfica No.4 llama a un procedimiento, que es el encargado de guardar en una lista las instrucciones que se van realizando, de modo que cuando ésta se quiera volver a realizar, sólo se deba leer de dicha lista, estas instrucciones son las misma que presentan en la figura No. 5.

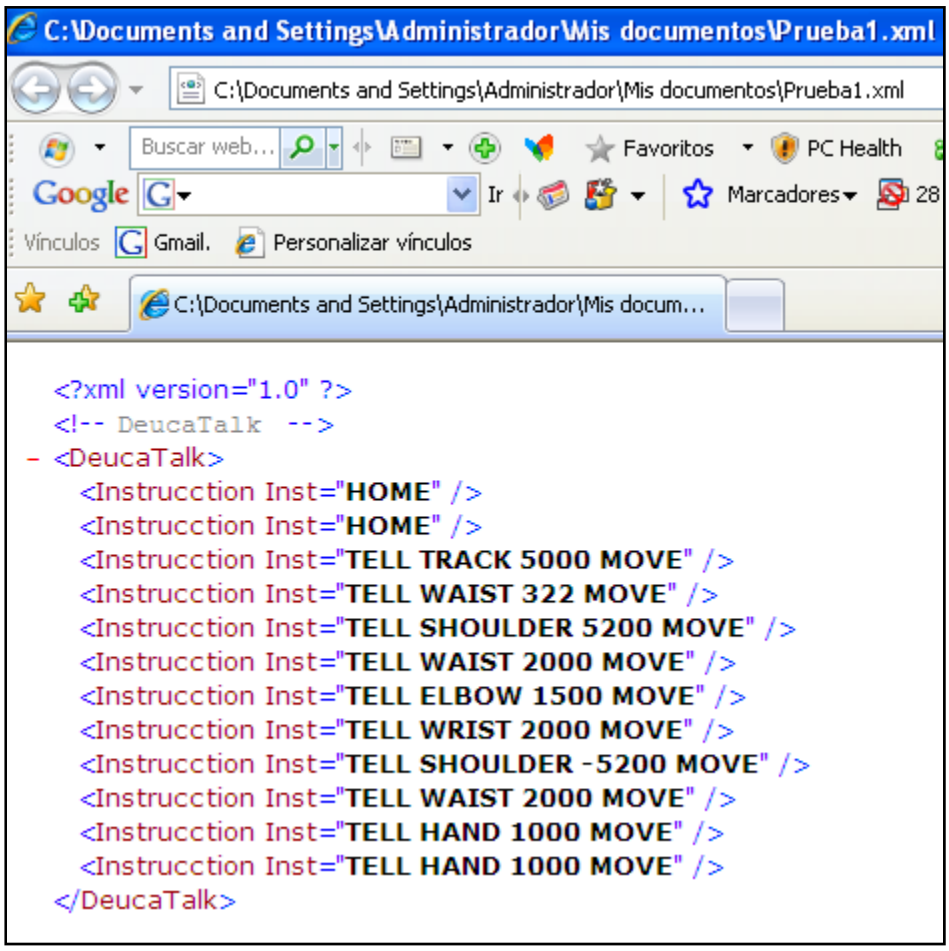


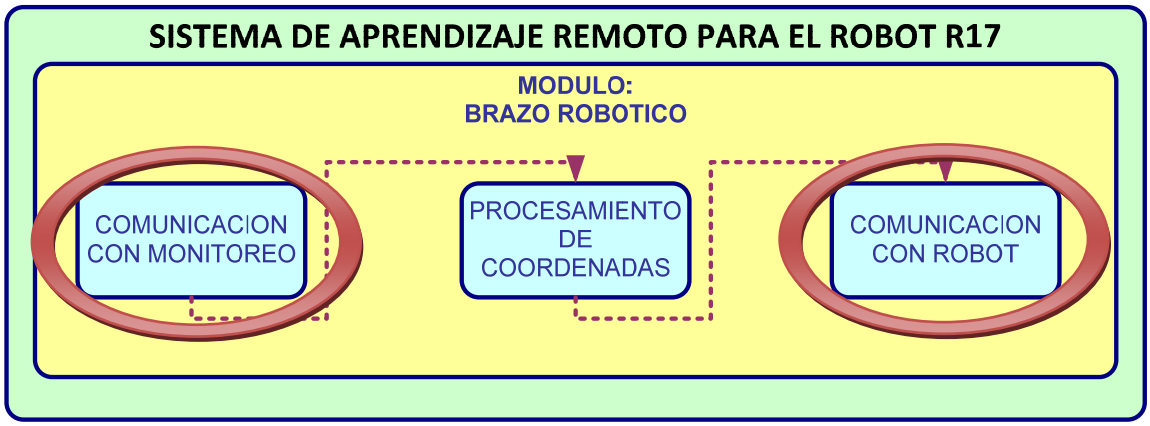
Gráfico No. 5 Fotografía de instrucciones guardadas en un archivo.

La sección de manipulación es la encargada de abrir el puerto serial y hacer la conexión con el controlador de ST Robotics, Así mismo es el encargado de cambiar los modos de automático a manual, con esto es el encargado además de comenzar la comunicación con Wifi. Cumple también la función de guardar un proyecto previamente realizado, en la gráfica No. 5 se visualiza un proyecto guardado, también se puede abrir un proyecto previamente guardado.

La sección de visualización es importante, ya que es la que permite darle una retroalimentación al módulo de monitoreo en la silla operativa. Este refleja por ejemplo, si alguna instrucción que se intentó realizar no tuvo éxito. Esta información enviada desde el brazo robótico será codificada para ser enviada en forma de mensaje a monitoreo. Existe un conjunto de mensajes que no son parte de la información enviada por el brazo robótico, por ejemplo aquellas que tienen que ver con los límites de las uniones, ya que esto lo realiza la aplicación.

La subdivisión del módulo de Brazo Robótico se realizó de la siguiente manera:

- Procesamiento de Coordenadas
- Comunicación con Monitoreo
- Comunicación con Robot



Gráfica 20: diagrama de subdivisión del módulo de Brazo Robótico

El primer Submódulo, Procesamiento de Coordenadas, se encargará de transformar los valores recibidos de parte de la silla operativa para cada articulación en coordenadas e instrucciones inteligibles para el robot. Este Submódulo actúa de enlace entre los dos otro submódulos.

El Submódulo de Comunicación con Monitoreo se ocupa de la capa de comunicación entre la aplicación de escritorio y la silla operativa. Los mensajes que contienen información de cada articulación son recibidos por esta capa, mientras que los mensajes que contienen información sobre el estado actual del robot son enviados a través de esta capa.

El tercer Submódulo, Comunicación con Robot, está a cargo de la capa opuesta de comunicación entre el robot propiamente dicho y la aplicación de escritorio. Esta capa se encarga de enviar las instrucciones necesarias al robot y de recibir las confirmaciones y mensajes de estado de parte del robot para comunicarlos a la sección de monitoreo de la silla operativa.

Los submódulos de interés para este documento son los de Comunicación con Robot y Comunicación con Monitoreo; estos dos submódulos constituyen la capa de Data Link del módulo de Brazo Robótico ya que proveen la

infraestructura necesaria para comunicar la aplicación de escritorio tanto con el robot R17 como con la silla operativa. Se tomó la decisión de utilizar el lenguaje de programación C# en conjunto con la plataforma .NET de Microsoft, utilizando el ambiente de desarrollo Visual Studio 2005 en su edición gratuita (Express Edition), debido a la gran cantidad de librerías que pone a disposición del usuario.

B. Observación e Investigación

Luego de haber definido y atacado el problema por medio de la división de tareas, se procedió a la fase de observación e investigación. Antes de comenzar a investigar los detalles técnicos del robot y los módulos a desarrollar, fue necesario pensar sobre lo que el robot debería ser capaz de hacer luego de la finalización del Megaproyecto. La investigación realizada tomó como objetivo identificar las necesidades de la industria en Guatemala y definir las posibles aplicaciones del robot que pudieran satisfacer estas necesidades. Esta investigación sirvió para darle un enfoque concreto al Megaproyecto.

Continuando con la fase de observación, fue necesario enfocarse en el aspecto computacional del robot. El robot R17 cuenta con un controlador que hace las veces de ordenador para el robot, ya que contiene memoria y un procesador capaz de tomar instrucciones de la memoria y ejecutarlas sin ayuda de un dispositivo externo. Esta memoria contiene programas definidos en el lenguaje de programación ROBOFORTH, el cual es un lenguaje de programación de cuarto nivel utilizado para programar los movimientos del robot, y diseñado para la programación de robots de este tipo en general.

Para poder trasladar los programas a la memoria, el controlador se conecta mediante un puerto serial a una computadora que contiene el software especializado ROBWIN, el cual es utilizado para evaluar instrucciones y programar el robot interactuando con él mediante el empleo de ROBOFORTH.

Es precisamente este software el que se pretende reemplazar con la interfaz intuitiva de entrenamiento para el robot. Parte de la investigación fue realizada utilizando la documentación incluida con el robot, por ejemplo el manual del programador para ROBOFORTH, y algunos documentos que ejemplifican el uso correcto de ROBWIN.

Debido a la naturaleza de la tecnología con la que se trabajó, la observación jugó un papel clave en la obtención de información necesaria para proceder con la siguiente fase. Básicamente se consultó toda la información disponible en los manuales de referencia del robot R17 y luego se observó el comportamiento del robot en varias circunstancias que se detallan a continuación.

Para el Submódulo de Comunicación con Robot, fue necesario recurrir a la observación, debido a que no existe una documentación del protocolo que utiliza el software ROBWIN para comunicarse con el robot R17. Por ello fue necesario utilizar un software especializado para realizar un rastreo del puerto serial mientras ocurriera la comunicación con el robot, para poder descifrar el protocolo utilizado entre la aplicación ROBWIN y el robot R17.

Durante la observación del uso de ROBWIN para enviar instrucciones sencillas de movimiento al robot, se descubrió que las instrucciones se transmiten como texto plano en el lenguaje de programación ROBOFORTH, enviando las instrucciones carácter por carácter al robot y recibiendo el mismo carácter enviado como confirmación, y al final un carácter de ENTER. Las instrucciones requieren de una confirmación por parte del robot para asegurar su ejecución correcta. Con esta información, se determinó la secuencia de las instrucciones a enviarse para inicialización del robot y la manera como el robot envía sus confirmaciones para cada instrucción.

Asimismo, en este Submódulo se recurrió a una etapa de investigación para determinar la forma correcta de programar la capa de comunicación con el

robot por medio del puerto serial. Como resultado de esta investigación se encontró una clase específica, llamada SerialPort del namespace System.IO.Ports, dentro del marco de trabajo utilizado (.NET Framework 2.0 de Microsoft, para ser utilizado mediante el ambiente de desarrollo Visual Studio 2005) que maneja toda la comunicación a través del puerto serial, tanto de emisión como de recepción.

Anteriormente, los integrantes del Submódulo de comunicaciones de la silla operativa tomaron la decisión de utilizar tecnologías inalámbricas para la comunicación entre la silla y la aplicación de escritorio. Por ello, el siguiente paso en la investigación consistió de investigar, por el lado de la aplicación, todas las formar posibles de aprovechar la infraestructura WiFi desde un punto de vista de programación, y luego de observar su compatibilidad con el dispositivo elegido por los integrantes de los demás submódulos, en este caso una tarjeta especial llamada WiPort que toma una comunicación por puerto serial y la transmite por WiFi.

Se procedió de esta manera con el Submódulo de Comunicación con Monitoreo, donde fue necesaria la observación del comportamiento de la tarjeta WiPort utilizada dentro de la silla operativa en el módulo respectivo, así como la investigación de la manera adecuada de encapsular paquetes a utilizar con el protocolo TCP para el envío y recepción de datos, aprovechando cualquier punto de acceso inalámbrico. De esta forma se separó la transmisión y recepción de datos y se logró aislar el Submódulo de Procesamiento de Coordenadas para su desarrollo independiente.

La conclusión inevitable fue la del uso necesario de la clase llamada Socket del namespace System.Net.Sockets, parte del .NET Framework 2.0 de Microsoft a utilizarse con Visual Studio 2005. La investigación en este sentido se simplificó habiendo definido con anterioridad el ambiente de desarrollo a

utilizar, por lo cual las opciones se vieron reducidas en número y fue mucho más fácil hacer una elección.

Habiendo realizado toda la parte de investigación, surgió la necesidad de desarrollar rápidamente una aplicación que controlara el robot de una manera muy básica desde la computadora, con el propósito de respaldar una presentación formal preliminar del Megaproyecto con una aplicación tangible que el público pudiera utilizar y de esta manera crear confianza en el Megaproyecto.

La solución temporal más factible fue conseguir una versión gratuita de un componente para .NET que se encargara de toda la capa de comunicación con el puerto serial. Con esta base, la programación se simplificó de gran manera y se desarrolló una pequeña versión de la aplicación de escritorio que realizara movimientos básicos en cada una de las articulaciones del robot. Esto permitió no sólo que fuera posible para cualquier persona controlar el robot desde una computadora, sino que se experimentara la manera correcta de controlar el robot para el desarrollo futuro de la aplicación.

C. Submódulo de Comunicación con Robot

El desarrollo de los dos submódulos se llevó a cabo siguiendo el modelo básico de la ingeniería de software. Primero, se realizó un análisis del sistema obteniendo los requerimientos y estableciéndolos claramente. Luego, con base en dichos requerimientos se realizó el diseño básico de la aplicación, lo cual incluye la arquitectura básica y las clases a utilizar con un esbozo de sus métodos. Al tener el diseño, se procedió a desarrollar la aplicación en sí utilizando las herramientas necesarias para ello, para luego proceder a la fase de implementación y pruebas. Se realizaron estos pasos en un ciclo para mejorar el software incrementalmente, hasta llegar a un resultado adecuado para el propósito que se deseaba cumplir.

El primer Submódulo desarrollado fue el de Comunicación con Robot. La tarea consistió en desarrollar una interfaz programable para la comunicación con el robot R17 con el propósito de ser flexible y contemplar los posibles cambios en el protocolo. En este caso, el protocolo involucra un puerto serial mediante el cual se mandan y reciben datos en forma de texto ASCII. Los requerimientos de este Submódulo son simples, y se detallan a continuación:

- La aplicación debe utilizar el puerto serial para realizar las comunicaciones con el robot R17.
- Los parámetros de conexión con el puerto serial deben ser configurables para escalabilidad en el futuro y compatibilidad adecuada.
- El protocolo a utilizar es el de texto plano, enviando un carácter a la vez y esperando la confirmación de cada uno.
- El robot envía una confirmación luego de cada instrucción, así que la aplicación debe manejar esto antes de enviar la siguiente instrucción.
- La aplicación debe mostrar toda la información necesaria para depurar el programa e identificar los errores que puedan ocurrir de manera clara.

Estos requerimientos fueron el resultado de la etapa anterior de observación e investigación, y representan una descripción de las funciones básicas de esta parte de la aplicación de escritorio.

El diseño de la aplicación corresponde a una arquitectura por capas, debido a la naturaleza de los submódulos implicados. De esta manera, se abstrae la capa de procesamiento de datos y se coloca por encima de la capa de comunicaciones, a su vez colocada por encima de la capa física ya separada desde las librerías del .NET. Por ello, los dos submódulos forman parte de la capa de comunicaciones, comúnmente denominada capa DataLink, y el Submódulo de Procesamiento de Coordenadas forma parte de la capa de aplicación.

De acuerdo con lo anterior, se diseñó el Submódulo de Comunicación con Robot para cumplir con la función de abstraer toda la lógica de los datos y limitarse a la transmisión y recepción eficiente. Para ello, el diseño consistió de una clase básica encapsulando la funcionalidad del puerto serial e implementando los métodos básicos de envío y recepción, además de un buffer intermedio donde son colocados los datos que se reciben y otro buffer de donde se leen los datos que se envían.

El desarrollo consistió en programar una clase con funciones para la entrada y salida de datos a través del puerto serial, para facilitar la comunicación con el robot. Esta clase se compiló como un DLL para facilitar su uso. La clase contiene métodos que envían y reciben datos, así como manejadores para colocar los datos en un buffer intermedio que cumple la función de un pipeline que comunica la capa de comunicaciones con la capa de aplicación.

Debido a que los parámetros del puerto serial recibidos por el controlador del robot R17 deben permanecer estáticos, la aplicación no permite alterarlos para asegurar la permanencia de la compatibilidad. Los parámetros y sus valores se establecen al inicializar la aplicación y están definidos como sigue:

- Puerto: COM1.
- Control de flujo (Flow Control): Xon/Xoff.
- Baudrate: 19,200.
- Databits: 8 databits.
- Paridad: Sin paridad.
- Stop bits: 1 bit.

Las pruebas fueron realizadas con la misma versión preliminar de la aplicación que fue utilizada para la presentación oficial del Megaproyecto, y

con el robot R17. Se reemplazó la clase gratuita por el DLL desarrollado y se verificó que todas las características ya desarrolladas para la aplicación funcionaran de la misma manera que lo hacían anteriormente. No ocurrió mayor problema durante la implementación de esta clase, y no hubo necesidad de regresar para revisar el diseño o las especificaciones para modificar la arquitectura básica.

De esta manera quedó implementado el Submódulo en su totalidad, y quedó establecida la forma de comunicación entre la aplicación de escritorio y el robot R17. Debe mencionarse que el desarrollo de este Submódulo fue crucial para poder iniciar con el desarrollo del Submódulo de Procesamiento de Coordenadas, por lo cual se utilizó la clase gratuita de puerto serial desde el inicio para facilitar el desarrollo paralelo de los submódulos involucrados en el módulo de Brazo Robótico.

D. Submódulo de Comunicación con Monitoreo

El desarrollo de este Submódulo consistió básicamente de los mismos pasos que el Submódulo anterior, aplicado al extremo opuesto de la aplicación de escritorio. Este extremo utiliza el protocolo TCP en lugar del puerto serial para enviar y recibir datos por medio de una red inalámbrica especial para este propósito. Siguiendo con el modelo de desarrollo de software utilizado, lo primero fue la especificación de requerimientos para esta parte de la aplicación. Los requerimientos obtenidos se detallan a continuación:

- La aplicación debe utilizar el protocolo TCP para comunicarse con la silla operativa, de acuerdo con el módulo de comunicaciones que se encarga de la transmisión de datos.
- La aplicación debe recibir los datos relativos a la información de cada articulación de la silla operativa y la información enviada por la interfaz de monitoreo de la silla operativa.

- La aplicación debe enviar los datos de estado del robot y cualquier otra información pertinente requerida por la interfaz de monitoreo.
- Los parámetros de conexión, como la dirección IP y el puerto a la cual se debe conectar la aplicación, deben permanecer completamente configurables para asegurar compatibilidad y escalabilidad.
- La aplicación debe mostrar toda la información necesaria para depurar e identificar cualquier error que pudiera ocurrir en el proceso de comunicación con la silla operativa.

Al igual que ocurrió con el Submódulo anterior, los requerimientos son el resultado de la fase de investigación y observación. Debe mencionarse que la utilización de tecnologías inalámbricas no afecta el desarrollo ni la arquitectura de la aplicación, pues es algo que concierne a la capa física y al hardware, y está separado incluso de las clases que manejan el protocolo base de comunicación. De esta manera, la aplicación funciona independientemente de la tecnología, aunque la implementación de una interfaz cableada del lado de la silla operativa es algo que permanece fuera del alcance de este Megaproyecto.

Continuando con la arquitectura de capas empleada en el Submódulo anterior, la capa de DataLink se completó diseñando la parte de comunicación con la silla operativa bajo el mismo modelo. Aprovechando la abstracción que proveen las clases de .NET con las cuales se trabajó, el diseño consistió de una clase sencilla que encapsula la funcionalidad del protocolo TCP y provee métodos para enviar y recibir datos mediante un buffer intermedio. Este diseño es congruente con el diseño del Submódulo anterior y provee todo lo necesario para que la capa de aplicación pueda desarrollarse independientemente del protocolo implementado.

El desarrollo del Submódulo de Comunicación con Monitoreo consistió nuevamente en programar la clase diseñada en la etapa anterior, implementando los métodos utilizando la clase Socket mencionada anteriormente. Los métodos

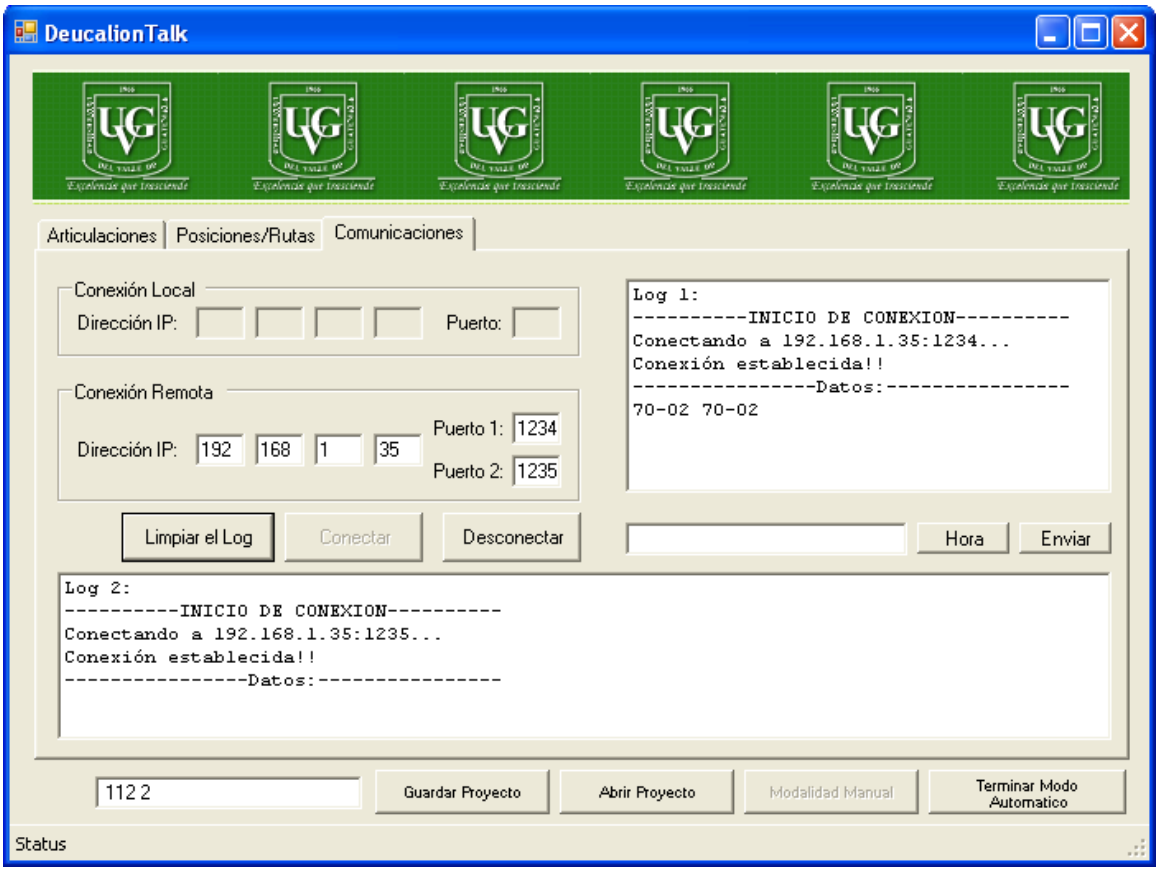
desarrollados son los de enviar y recibir datos, utilizando para ello dos buffers intermedios. Del primero se leen los datos colocados para enviarse a la silla operativa, y en el segundo se colocan según van llegando por medio del puerto TCP configurado.

Para optimizar la comunicación entre la aplicación, la Silla Operativa y Monitoreo, fue necesario utilizar threads (hilos) concurrentes. Fue necesario crear dos threads independientes de la aplicación principal, uno para manejar la conexión TCP que recibe los datos de la Silla Operativa y los coloca dentro del buffer, y el otro para enviar y recibir instrucciones con la parte de Monitoreo. Cada uno de estos threads se conecta a un puerto distinto de la dirección IP del WiPort. Para las pruebas se colocaron los puertos 1234 y 1235 para conexiones con Monitoreo y con la Silla Operativa, respectivamente. Sin embargo, tanto la dirección IP del WiPort como los puertos son configurables mediante la aplicación.

Los threads se manejaron desde métodos delegados similares pero independientes entre sí. Estos métodos consisten básicamente de un ciclo infinito que revisa constantemente el socket y recibir datos enviados desde la Silla Operativa o desde Monitoreo. Dentro de este método se maneja la lógica específica de la aplicación, tomando los datos recibidos y colocándolos dentro de un buffer, y manejando tanto los errores como los datos recibidos en un despliegue de texto.

Al igual que con el Submódulo anterior, las pruebas fueron realizadas con la aplicación de escritorio y una aplicación diseñada específicamente para probar el funcionamiento de estos aspectos, lo cual sirvió para identificar errores en la lógica y el diseño del programa. Luego de identificarlos se revisó el diseño y se realizaron cambios a la implementación de acuerdo con las fallas observadas, para luego realizar nuevas pruebas. Este proceso se repitió varias

veces hasta lograr corregir todos los errores mayores de la aplicación y lograr un funcionamiento satisfactorio de la clase de comunicación con la silla operativa.



Gráfica 21: Captura de pantalla de la aplicación de escritorio.

Así se concluyó el desarrollo de la capa de DataLink de la aplicación de escritorio. Según la arquitectura por capas, una vez funcionando la implementación de esta capa, todas las demás deben ser capaces de interactuar sin problemas entre sí, y con la capa DataLink específicamente, encapsulando sus datos y agregando los niveles de abstracción necesarios en cada capa.

La integración con el Submódulo de Procesamiento de Coordenadas, una vez desarrollado, consistió básicamente de sincronizar el uso de los buffers compartidos como pipelines de comunicación en ambas direcciones. Una vez integrado este Submódulo con los dos anteriores, se dio por concluido el desarrollo de la aplicación de escritorio y consecuentemente el Módulo de Brazo Robótico fue completado exitosamente en su desarrollo.

VIII. CONCLUSIONES Y RECOMENDACIONES

1. El proyecto de investigación realizado permitió que diez estudiantes, de ingeniería electrónica y de ingeniería en ciencias de la computación, implementaran los conocimientos adquiridos a lo largo de la carrera y lograran consolidarlos mientras solucionaban un problema real que involucra a la industria.

2. Por medio del proyecto, la Universidad del Valle de Guatemala, a través de sus estudiantes, hace una contribución directa a la industria guatemalteca. Por lo anterior, participa activamente en el desarrollo tecnológico del país ya que el sistema de aprendizaje remoto permite que cualquier usuario sin conocimientos de robótica logre "enseñarle" al robot R17 una rutina especializada.

3. El proyecto tiene la versatilidad de adaptarse a cualquier proceso industrial ya que se pueden crear estructuras mecánicas de acuerdo a las especificaciones de los robots que las industrias posean, para luego instalar los dispositivos y hacer los ajustes de programación que las nuevas estructuras requieran.

Silla Operativa

Estructura Mecanica

A. Se construyó la estructura mecánica que permite al operario maniobrar la interfaz humana que replica las partes del brazo robótico R17. La estructura contempla un espacio para colocar los circuitos que procesan los datos de los sensores, así como el acople de los sensores a la estructura misma. .

Brujula

- Se logró la implementación de la brújula digital, la cual proveyó la requerida lectura del movimiento circular en el eje XY.

- Se logró la conversión de protocolo I2C a CAN para la transmisión de datos hacia la PC que se comunicaba con el robot.

- La brújula digital presentó una inexactitud de 10 grados, que fue corregida con la implementación de una recta de regresión y = 0.9735x-1.0249. Una vez aplicada la ecuación se corrigieron los problemas de inexactitud y se obtuvieron los valores esperados para cada movimiento.

- Un análisis profundo de las ventajas y desventajas de utilizar un encoder en lugar de una brújula serian beneficiosos para futuras implementaciones.

- Realizar una rutina de calibración automática eliminaría errores en las lecturas, ya que las condiciones cardinales cambian según el lugar de trabajo.

Acelerometros

Como en toda aplicación, el tipo de sensor de movimiento depende tanto de los requisitos del sistema, como del fenómeno medido.

Al estudiar los componentes principales que representan e influencian el movimiento de un cuerpo, fue demostrado un método capaz de determinar las coordenadas de los ejes del cuerpo por medio de acelerómetros.

Algunos acelerómetros tienen mejor rendimiento al medir vibraciones, los de tecnología piezoeléctrica. Otros, tales como los micromaquinados, tienen un mayor rendimiento al medir la inclinación. La aplicación depende de la tecnología del diseño del acelerómetro.

La detección de la inclinación es un ejemplo, en las aplicaciones de los acelerómetros, que exige una ancho de banda que se extienda hasta 0 hertz.

Los acelerómetros son sensibles a los procesos de acoplamiento a una estructura. Para obtener un desfasamiento mínimo de voltaje de salida, mostrado como la fuente de error más significativa, después del acoplamiento, se debe tener mucho cuidado al escoger la superficie sobre la que ira montado. Esta debe ser lisa con la menor cantidad de imperfecciones, tal que minimice las vibraciones entre la estructura en movimiento y el acelerómetro.

El desfasamiento de los voltajes de salida puede también ser influenciado por los cambios de la temperatura ambiente. Aunque se tienen que realizar experimentos adicionales, con una gama más amplia de temperaturas de trabajo, para probarlo.

Codificadores

Emplear un juego de engranajes con relación 1:1 en la mano de la estructura, facilita la programación del procesamiento para el sistema de control, y principalmente evita que el conector del sensor colisione entre la piezas de la estructura, cuando el ángulo entre la juntura es muy pequeño.

Emplear un juego de engranajes con la relación 1:2 en la muñeca de la estructura, requiere de la detección del refrescado del sensor, cuando éste gira una vuelta completa. La ventaja de este diseño permite una ampliación del rango de salida nominal proporcionado por el codificador. Se ampliará la escala decimal del sensor de 0-1860.

El error de fábrica típico del codificador, produce una oscilación entre los valores anterior y siguiente, de la lectura, el cual corresponde a un ruido analógico típico del sensor de 20 MVolts pico a pico.

Monitoreo

- Se determinó que la pantalla GLCD es más robusta y más adecuada para su operación en la silla operativa, debido a que permite un mayor tiempo encendida sin daños, y su relación tamaño costo.

- Se determinó que es necesario, para el manejo del sistema desde la silla operativa, la implementación de los menús de monitoreo, conexión, grabación y configuración, debido a la resolución de la pantalla y a que la información necesaria en las tres áreas más importantes fuera fácilmente de comprender.

- Se lograron desarrollar funciones de despliegue que permiten una fácil implementación para otro tipo de menús, o pantallas como la barra de lado.

- Si la pantalla experimenta un uso prolongado, se debe de implementar una guarda pantalla, *screensaver*, o un apagado por medio de alguno de los temporizadores del PIC o del reloj virtual ya implementado, para proteger la pantalla de un quemado de píxeles.

- Se recomienda la pantalla uOled como una buena implementación, si se sustituye la silla operativa del sistema mediante un guante. Implementando los acelerómetros en este al igual que la pantalla. Esta se vería beneficiada por su tamaño y ángulo de visión. Esto implica el que la estructura de la silla operativa se ve reemplazada por un guante y se implemente una estructura externa para guardar la fuente de poder y el Wiport.

Comunicaciones

Red Local

1. La red de comunicación local utiliza el protocolo CAN y funciona adecuadamente para los requerimientos del proyecto; permite que los nodos se comuniquen satisfactoriamente, respetando las prioridades de cada uno y no permite que queden aislados por falta de prioridad para transmisión de mensajes.

2. A pesar de que se puede dar el caso que dos nodos posean un mismo identificador de mensaje al momento de la transmisión, no existe colisión en el bus ni se pierden los datos, debido al protocolo de recepción de CAN.

4. La fase de recepción funciona adecuadamente ya que implementa en una sola función, general para todos los nodos, la detección, validación y filtrado de los mensajes que circulan en el bus de comunicaciones.

Red WiFi

Transmision de datos

A. Se determinó que la velocidad de comunicación serial óptima entre WiPort y PIC18F458 fue de 57,600 bits/s dado que fue la velocidad más alta de transmisión de un archivo de texto sobre la cual no ocurrieron errores.

Recepcion de datos

Se determinó que la transmisión Wi-Fi del PIC18F458, encargado de armar paquetes de 12 bytes y dedicado a enviar estos paquetes por serial asíncrono a una velocidad de 57,600 bits/s, hacia la PC a través del WiPort, transmitió todos los datos sin errores.

1. Se logró establecer una conexión inalámbrica por 802.11b/g entre el WiPort y una PC. La conexión se inició en modo Ad-Hoc (conexión directa sin infraestructura) y se configuró posteriormente para que fuera una transmisión con infraestructura. Esto permite versatilidad a la hora de instalar el robot en ambientes variados.
2. Se encontró que la velocidad más adecuada para la transmisión de datos es 57,600 bps. Esta velocidad de transmisión de datos fue escogida por ser la velocidad más alta a la que se podía transmitir sin perder datos. Las perdidas de datos se debían a limitaciones de hardware de los micros controladores.
3. Se logró la comunicación entre el WiPort y un PIC a través de puerto serial. Esta comunicación fue obtenida utilizando la velocidad descrita anteriormente, y utilizando el puerto serial del PIC.
4. Se logró la comunicación efectiva de datos entre el WiPort y el Módulo de Monitoreo. Esta comunicación consiste en transmisión y recepción de datos recibidos del Módulo de Monitoreo y su envío a la PC a través de WiPort, así

como el procesamiento de comandos enviados por el Módulo de Monitoreo, y su respuesta correspondiente.

5. Se recomienda instalar un sistema de enfriamiento al WiPort, ya que este puede sobrecalentarse al utilizarse por períodos prolongados, que a largo plazo pueden significarle un daño permanente. Una alternativa viable a esto es disminuir la velocidad interna del WiPort, o disminuir la potencia de transmisión de la antena, ya que estos dos pasos disminuyen el consumo de potencia y ayudan a disminuir el sobrecalentamiento, y así cualquier daño que puede sufrir el WiPort.

Brazo Robotico

- La creación de la aplicación permite establecer una comunicación entre la silla operativa y el brazo robótico, por medio del cual es posible replicar, guardar y cargar movimientos generados por la silla operativo.
- La combinación del conjunto de instrucciones de roboforth con el protocolo DeucaTalk, permiten crear una comunicación fluida entre la silla operativa y el brazo robótico.
- La visualización del flujo de la información así como la posición actual de cada una de las uniones permiten tener un panorama del funcionamiento del sistema del tal forma que es posible entender que procesos se están ejecutando sin tener conocimiento de programación del robot.
- Se recomienda para una futura implementación hacer pruebas conjuntas con los módulos antes de ponerlo en producción.
- Se recomienda impartir una inducción antes de que el usuario final utilice el proyecto.
- Se recomienda leer la documentación técnica antes de utilizar el proyecto.

A partir del trabajo realizado y de la experiencia y conocimiento acumulados a lo largo del desarrollo del Megaproyecto, es posible extraer una serie de conclusiones que resumen el propósito general y destacan el cumplimiento de los objetivos establecidos al principio de este documento. Dichas conclusiones se presentan a continuación, seguidas de sus respectivas recomendaciones.

Como conclusión principal del trabajo realizado, con base en los resultados obtenidos durante el desarrollo del Megaproyecto, y observando que el robot R17 requiere de un nivel muy alto de conocimiento para programarse, se puede afirmar que es posible crear una interfaz intuitiva para programar el robot R17 sin necesidad de utilizar un lenguaje de programación como tal ni de tener conocimientos técnicos. La aplicación de escritorio creada, está diseñada específicamente para aquellas personas que no poseen este tipo de conocimiento y tendrán que enfrentarse a diario con tecnología robótica. Una interfaz intuitiva que utiliza los movimientos del cuerpo para enseñarle al robot es una manera eficiente de esquivar la dificultad de programar un robot con un lenguaje de programación, y por encima de todo es algo que cualquier persona está en capacidad de utilizar.

Se recomienda, con base en esta conclusión, el desarrollo de una interfaz lo menos técnica posible para la programación adecuada del robot R17, ya que este sistema posee una complejidad que requiere de una gran cantidad de conocimiento técnico por parte del usuario final. Esto impactará la facilidad y rapidez con la que los usuarios pueden dar uso al sistema sin la necesidad de un entrenamiento riguroso, gracias a la cualidad intuitiva que posee una interfaz poco técnica.

Es posible concluir también, de manera más general, que mediante la abstracción es posible encapsular los detalles técnicos de un proceso que involucra componentes tecnológicamente avanzados y crear un adaptador que permita utilizarlos sin necesidad de conocer lo que hay dentro, lo cual es

especialmente útil para personas con poca o nula educación y/o experiencia con respecto al uso de la tecnología, ya sea un computador o un robot. En este caso, el adaptador consiste en la aplicación de escritorio desarrollada que enlaza la Silla Operativa con el robot R17.

Consecuentemente, es recomendable aplicar los conceptos de abstracción y encapsulamiento a todos los aspectos técnicos de cualquier producto cuyos usuarios finales no poseen conocimientos técnicos. El beneficio será para ambas partes, ya que el usuario se beneficia con una interfaz sencilla de utilizar y la industria lo hace con una mayor productividad por parte del usuario final y un tiempo más reducido, tanto en el aprendizaje como en el uso diario de la tecnología.

Lo anterior lleva a la siguiente conclusión, la cual indica que la facilidad de uso por parte de personal sin conocimientos técnicos *per se* implica un alto nivel de adaptabilidad para las industrias existentes en el país, lo cual a su vez se traduce en niveles de productividad más altos y a la larga en costos reducidos. Esto contribuye de gran manera al desarrollo de la industria en el país, y lo lleva un paso más cerca del nivel al que están los países más industrializados.

El Megaproyecto como tal deja como conclusión que la incorporación de varias disciplinas aparentemente incompatibles dentro de un solo proyecto no sólo es posible, sino que es necesaria para lograr resultados que tengan alguna trascendencia. Si bien el efecto inmediato del Megaproyecto no es el levantamiento de la industria y el progreso del país, este es un paso pequeño que ha sido dado para colocar al país más cerca de una posición competitiva en el mercado.

Como última recomendación debe aconsejarse que en cualquier proyecto que involucre disciplinas distintas, la mejor opción es que personas expertas en cada tema trabajen en equipo para solucionar el problema desde un principio, ya que no hay nadie mejor para entender cada parte individual que un experto en el

área, y la integración debe ser planificada desde el inicio para asegurar que los problemas de incompatibilidad se minimicen. La división del problema multidisciplinario en problemas más pequeños que involucren una sola disciplina es el punto clave para asegurar que el proyecto pueda resolverse de esta manera, y en la menor cantidad de tiempo.

Se recomienda que, tanto el sector académico como industrial, procuren trabajar en conjunto para solucionar problemas relacionados con las nuevas tecnologías. Se ha observado que muchas veces las industrias no poseen los recursos necesarios para hacer las investigaciones pertinentes mientras que las universidades realizan investigaciones sin considerar los problemas que acontecen en la industria y en un país como Guatemala, donde no existen tantos recursos económicos para hacer investigación porque aun hay necesidades sociales insatisfechas, es importante aprovechar cada oportunidad al máximo sin desperdiciar la capacidad y el potencial de las personas.

IX. BIBLIOGRAFÍA

- CAN in Automation (CiA); Controller Area Network (CAN) – Protocol [Documento en línea]; 2007-01-02; Disponible en Web: http://www.can-cia.org/can/protocol/ [26 de agosto de 2007].

- Cisco Systems, Inc; Open System Interconnection Protocols [[Documento en línea]; URL: http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/osi_prot.htm [08 de febrero de 2007].

- Depto. de Arquitectura y Tecnología de Computadores; Robótica Industrial; E. U. Politécnica de la Universidad de Sevilla; [Documento en línea] URL: http://www.atc.us.es/index.php?op=descargas&pag=4 [30 de octubre de 2007]

- Dr. Lee Giles; Networking and Telecommunications [Documento en línea]; IST 220-002; School of Information Sciences and Technology; The Pennsylvania State University; URL: http://www.ist.psu.edu/faculty_pages/giles/IST220/ [08 de febrero de 2007]

- ELAU; Dans l'emballage, les robots ont le vent en poupe; Schneider Electric; 2006 [Documento en linea]; URL: http://www.elau.de/Rahmen.asp?Knoten=1375 [30 de octubre de 2007]

- Etneo; Soldering Robot; Italia; 2004 [Documento en línea]; URL: http://www.etneo.com/lcaten.htm [30 de octubre de 2007]

- Hadi A. Akeel, Steve W. Holland; Product and Technology Trends for Industrial Robots; Industrial Robots Symposium; 2000; [Archivo PDF]

- Industrial Manipulators; TrueForce [Documento en línea]; URL: http://trueforce.com/ [30 de octubre de 2007]
- International Federation of Robotics (IFR); Robot Applications; [Documento en linea] URL: http://www.ifr.org/index.asp; [30 de octubre de 2007]
- IT Technical Institute; Media and Topologies [Documento en línea]; URL: http://www.studynotes.net/net1.htm [08 de febrero de 2007].
- Jim Berge; Welding robots and lean manufacturing learn to play together;; octubre 2005; [Documento en linea]; URL: http://www.thefabricator.com [30 de octubre de 2007]
- Leon W. Couch, II; Digital and Analog Communication systems [Texto impreso]; Sexta edición; Prentice Hall, Inc.; 2001; 758 Págs.
- Loop Technology Limited; Types of Robots; Inglaterra, 2007; [Documento en línea]; URL: http://www.looptechnology.com/robotic-robot-types.asp [30 de octubre de 2007]
- Marcos Garcia Bartolomé, Jose Mª Álvarez Ontivero, Daniel Cava Jimenez; Robotronica, Aplicaciones de la Robotica; Universitat Politecnica de Catalunya; junio 2003 [Archivo PDF]
- Microchip. Microchip PIC18FXX8 Data Sheet [Archivo PDF]. Revisión D [U.S.A]: septiembre 2004. 19.0 CAN Module; pág.: 202 / 402. URL: http://www.microchip.com/stellent/idcplgidcplg?IdcService=SS_GET_PAGE&nodeId=1335&dDocName=en010301 [26 de agosto de 2007]

- National Aeronautics and Space Administration (NASA); ROVer Ranch; JSC Learning Technologies; enero 2007 [Documento en linea]; URL: http://prime.jsc.nasa.gov/ [30 de octubre de 2007]

- Peter Burden's; The ISO Open Systems Interconnection Reference Model [Documento en línea]; School of Computing and I.T. (SCIT), University of Wolverhampton, UK; URL: http://www.scit.wlv.ac.uk/~jphb/comms/std.7layer.html [08 de febrero de 2007].

- Richard Martin, *Director of marketing at Broadcom Corporation*; Understanding Layer 3 and 4 Classification to Enhance Gigabit Ethernet Deployment [Documento en línea]; Marzo 2002; URL: http://www.dell.com/content/topics/global.aspx/power/en/ps1q02_broadcom?c=us&cs=555&l=en&s=biz [08 de febrero de 2007].

- Robotics Research Group (RRG); Industrial Robots; The University of Texas at Austin; J.J. Pickle Research Campus; [Documento en línea]; URL: http://www.robotics.utexas.edu/rrg/learn_more/low_ed/types/industrial.html; [30 de octubre de 2007]

- SAMSON Technical Information; Digital Signals [Documento en línea]; URL: http://www.samson.de/pdf_en/l150en.pdf [08 de febrero de 2007].

- SAMSON Technical Information; Serial Data Transmisión [Documento en línea]; URL: http://www.samson.de/pdf_en/l153en.pdf [08 de febrero de 2007].

- ST Robotics; R17 Robot Manual; Deucalion; diciembre 2006 [Archivo PDF]

- TCM Laboratory; Electrical Basics of data Transmisión [Documento en línea]; 24.2.1999; URL: http://www.tml.tkk.fi/Opinnot/Tik-110.250/1999/Kalvot/vemppa195/ [08 de febrero de 2007].

- Webster Griffin Ltd; Robot Palletising Systems; 2006 [Documento en línea] URL: http://www.webstergriffin.com/?pgid=42 [30 de octubre de 2007]

- Campus de Amaquique. *¿Qué es una señal?*

- Comunidad Electrónicos. *El bus I2C.* http://www.comunidadelectronicos.com/articulos/i2c.htm

- Dans l'emballage, les robots ont le vent en poupe; ELAU, Schneider Electric; 2006 [Documento en linea]; URL: http://www.elau.de/Rahmen.asp?Knoten=1375 [30 de octubre de 2007]
- Floyd, T.L. (2000) *Fundamentos Digitales.* Ed. Prentice Hall

http://campus-llamaquique.uniovi.es/virtual/docencia/websignal/modelos/proces.htm

- Industrial Manipulators; TrueForce [Documento en línea]; URL: http://trueforce.com/ [30 de octubre de 2007]
- Industrial Robots; Robotics Research Group (RRG); The University of Texas at Austin; J.J. Pickle Research Campus; [Documento en línea]; URL: http://www.robotics.utexas.edu/rrg/learn_more/low_ed/types/industrial.html; [30 de octubre de 2007]
- Jim Berge; Welding robots and lean manufacturing learn to play together;; octubre 2005; [Documento en linea]; URL: http://www.thefabricator.com [30 de octubre de 2007]
- Lynn, Paul A. (1992) ***Introductory Digital Signal Processing with Computer Applications.*** 2a edición

- Morris, Alan S. (2001). *Measurement & Instrumentation Principles.* Massachusetts: Butterworth Heinemann.
- *Muestreo y Digitalizacion de Señales*. Universidad de Sevilla, Departamento de Tecnología Electrónica.

- Product and Technology Trends for Industrial Robots; Hadi A. Akeel, Steve W. Holland; Industrial Robots Symposium; 2000; [Archivo PDF]
- R17 Robot Manual; ST Robotics; Deucalion; diciembre 2006 [Archivo PDF]
- Robot Applications; International Federation of Robotics (IFR); [Documento en linea] URL: http://www.ifr.org/index.asp; [30 de octubre de 2007]
- Robot Palletising Systems; Webster Griffin Ltd; 2006 [Documento en línea] URL: http://www.webstergriffin.com/?pgid=42 [30 de octubre de 2007]
- Robótica Industrial; E. U. Politécnica de la Universidad de Sevilla; Depto. de Arquitectura y Tecnología de Computadores [Documento en línea] URL: http://www.atc.us.es/index.php?op=descargas&pag=4 [30 de octubre de 2007]
- Robotronica, Aplicaciones de la Robotica; Marcos Garcia Bartolomé, Jose Mª Álvarez Ontivero, Daniel Cava Jimenez;Universitat Politecnica de Catalunya; junio 2003 [Archivo PDF]
- ROVer Ranch; National Aeronautics and Space Administration (NASA); JSC Learning Technologies; enero 2007 [Documento en linea]; URL: http://prime.jsc.nasa.gov/ [30 de octubre de 2007]
- Soldering Robot; Etneo; Italia; 2004 [Documento en línea]; URL: http://www.etneo.com/lcaten.htm [30 de octubre de 2007]
- Types of Robots; Loop Technology Limited; Inglaterra, 2007; [Documento en línea]; URL: http://www.looptechnology.com/robotic-robot-types.asp [30 de octubre de 2007]

- ADXL05:, E. A. (s.f.). *EL ACELERÓMETRO MONOLÍTICO ADXL05.* Recuperado el 30 de Octubre de 2007, de FCEIA: http://www.fceia.unr.edu.ar/enica3/adxl05.pdf
- Campus, J. P. (s.f.). *Industrial Robots.* Recuperado el 30 de Octubre de 2007, de The University of Texas at Austin: http://www.robotics.utexas.edu/rrg/learn_more/low_ed/types/industrial.html
- Computadores, D. d. (s.f.). *Robótica Industrial.* Recuperado el 30 de Octubre de 2007, de E. U. Politécnica de la Universidad de Sevilla: http://www.atc.us.es/index.php?op=descargas&pag=4

- Devices, A. (s.f.). *ADXL05 - 1g to 5g Single Chip Accelerometer with Signal Conditioning.* Recuperado el 2007 de October de 30, de Analog Devices: http://www.analog.com

- E. Abbaspour-Sani, R. H. (1994). *A linear electromagnetic accelerometer.*

- Estepa, A. L. (s.f.). *Diseño de un acelerómetro basado en tecnología MEMS.* Recuperado el 25 de October de 2007, de Grupo de Tecnologia Electronica: http://www.gte.us.es/ASIGN/SEA/MEMS_PRACT1.pdf

- Gardner, J. (1994). *Microsensors: principles and applications.* JohnWiley & Sons.

- Hadi A. Akeel, S. W. (2000). *Product and Technology Trends for Industrial Robots.* Industrial Robots Symposium.

- Kitchin, C. (s.f.). *Using Accelerometers in Low g Applications.* Recuperado el 20 de September de 2007, de Analog Devices: www.analog.com

- Lemkin, M. B. (1996). *A micromachined fully differential lateral accelerometer.* Dig. Tech. Papers.

- Lu, C. L. (1995). *A monolithic surface micromachined accelerometer with digital output.* IEEE Journal of Solid State Circuits.

- Madou, M. J. (1997). *Fundamentals of Microfabrication.* CRC Press.

- Mike Shuster, B. B. (s.f.). *Mounting Considerations for ADXL Series Accelerometers.* Recuperado el 2007 de October de 20, de Analog Devices: http://www.analog.com

- Morris, J. (1973). *Accelerometry.* Biomech.

- *PIC18FXX8 Datasheet.* (s.f.). Recuperado el 20 de October de 2007, de Microchip: http://www.microchip.com

- *Robot Applications.* (s.f.). Recuperado el 30 de Octubre de 2007, de International Federation of Robotics (IFR): http://www.ifr.org/index.asp

- Rudolf, F. (1983). *A micromechanical capacitive accelerometer.*
- Senturia, S. D. (2001). *Microsystem design.* Kluwer Academic.

- Senturia, S. D. (2001). *Microsystem Design.* Kluwer Academic.

- Yazdi, N. (1998). *Micromachined Inertial Sensors.* Proc. IEEE, vol.86.

Barros, S., (2003). **Sensores para Medir Desplazamiento**, extraído en Octubre de 2006, de: http://www.infoplc.net/Documentacion/Docu_Instrumentacion/infoPLC_net_Medida_Desplazamiento.pdf

Harris, D. M. J. (1993). **Robótica: Una introducción**, Mc Cloy, 1ª Edición, Editorial Limusa, México 1993.

Gonzales, ¨et als¨. Universidad Nacional de Córdoba (2002). **Sensores Digitales**, extraído en Octubre de 2006, de:

http://www.investigacion.frc.utn.edu.ar/sensores/Tutorial/TECNO5.pdf

Mártinez, ¨et als¨. Universidad de Alicante Departamento de Ciencias de la Computación e Inteligencia Artificial (2000). **Sensores internos,** extraído en Octubre de 2006, de:

http://www.dccia.ua.es/dccia/inf/asignaturas/ROB/optativos/Sensores/internos.html

Santos, J.H, (1990) . **Técnicas de Automatización Industrial,** , 2ª Edición, Editorial Limusa, Mexico 1990

Addlink Software Científico Multisim. Ingeniería- Electrónica. [Documento en línea] 2007. Disponible en URL: http://www.ingenieria-electronica.com/secciones/educacion [2007]

Alegsa. Que es la dirección IP. Noviembre 2006. [Documento en línea]. Disponible en URL: http://www.alegsa.com.ar/Notas/83.php [2007]

CANOBD2. ¿Qué es CAN?. 2006. [Documento en línea]. Disponible en Web: http://www.canobd2.com/knowledge/what_is_can-esp.asp [2007]

Dans l'emballage, les robots ont le vent en poupe; ELAU, Schneider Electric; 2006 [Documento en linea]; URL: http://www.elau.de/Rahmen.asp?Knoten=1375 [30 de octubre de 2007]

Jim Berge; Welding robots and lean manufacturing learn to play together;; octubre 2005; [Documento en linea]; URL: http://www.thefabricator.com [30 de octubre de 2007]

Industrial Manipulators; TrueForce [Documento en línea]; URL: http://trueforce.com/ [30 de octubre de 2007]

Lantronix. WiPort. [Documento en PDF]; 2007. Disponible en URL: http://www.co-nss.co.jp/download/datasheet/WiPort_IG.pdf [2007]

Melissa J. Perenson, OLED: La nueva estrella de la pantalla pequeña,. PCWorld en Español. 2005. [Documento en linea] Disponible en URL: http://www.pcwla.com/pcwla2.nsf/0/419421C1455A976B00256FD60048B915 [2007]

MicroLadder. Circuito Anti-Rebote. 2004. [Documento en linea] Disponible en URL: http://www.microladder.com/page.php?n=an2_rebound_effect_es [2007]

Nebojsa Matic. The PIC Microcontroller Book 1. [Documento en línea]; Junio 2004. Disponible en URL: http://www.mikroe.com/en/books/picbook/0_Uvod.htm [2007]

Parallax. “¿Qué es un Microcontrolador?” Guía del Estudiante Version 1.1. Versión en Castellano 1.1 2006. [Documento en PDF] Disponible en URL: http://parallax.com/dl/docs/books/edu/wamv1_1spanish.pdf [2007]

Pjmicro. ¿Que es un microcontrolador?.Noviembre 2006. [Documento en línea] Disponible en URL: http://pjmicrocontroladores.wordpress.com/2006/11/06/%C2%BFque-es-un-microcontrolador/ [2007]

Product and Technology Trends for Industrial Robots; Hadi A. Akeel, Steve W. Holland; Industrial Robots Symposium; 2000; [Archivo PDF]

Rivera, Cuathemoc. 2000 [Documento en línea] Disponible en URL: http://www.fismat.umich.mx/~crivera/tesis/tesis.html [2007]

Roso Control. Interfaces. 2004 [Documento en linea] Disponible en URL: http://www.roso-control.com/Espanol/iBOARD/10_iBOARD_Interfaces/20_LCD/LCD.htm#ConceptosBasicos [2007]

Robótica Industrial; E. U. Politécnica de la Universidad de Sevilla; Depto. de Arquitectura y Tecnología de Computadores [Documento en línea] URL: http://www.atc.us.es/index.php?op=descargas&pag=4 [30 de octubre de 2007]

Industrial Robots; Robotics Research Group (RRG); The University of Texas at Austin; J.J. Pickle Research Campus; [Documento en línea]; URL: http://www.robotics.utexas.edu/rrg/learn_more/low_ed/types/industrial.html; [30 de octubre de 2007]

Robot Applications; International Federation of Robotics (IFR); [Documento en linea] URL: http://www.ifr.org/index.asp; [30 de octubre de 2007]

R17 Robot Manual; ST Robotics; Deucalion; diciembre 2006 [Archivo PDF]

Robotronica, Aplicaciones de la Robotica; Marcos Garcia Bartolomé, Jose Mª Álvarez Ontivero, Daniel Cava Jimenez;Universitat Politecnica de Catalunya; junio 2003 [Archivo PDF] ROVer Ranch; National Aeronautics and Space Administration (NASA); JSC Learning Technologies; enero 2007 [Documento en linea]; URL: http://prime.jsc.nasa.gov/ [30 de octubre de 2007]

Robot Palletising Systems; Webster Griffin Ltd; 2006 [Documento en línea] URL: http://www.webstergriffin.com/?pgid=42 [30 de octubre de 2007]

Soldering Robot; Etneo; Italia; 2004 [Documento en línea]; URL: http://www.etneo.com/lcaten.htm [30 de octubre de 2007]

Types of Robots; Loop Technology Limited; Inglaterra, 2007; [Documento en línea]; URL: http://www.looptechnology.com/robotic-robot-types.asp [30 de octubre de 2007]

Robótica Industrial; E. U. Politécnica de la Universidad de Sevilla; Depto. de Arquitectura y Tecnología de Computadores [Documento en línea] URL: http://www.atc.us.es/index.php?op=descargas&pag=4 [30 de octubre de 2007]

Industrial Robots; Robotics Research Group (RRG); The University of Texas at Austin; J.J. Pickle Research Campus; [Documento en línea]; URL: http://www.robotics.utexas.edu/rrg/learn_more/low_ed/types/industrial.html; [30 de octubre de 2007]

Robot Applications; International Federation of Robotics (IFR); [Documento en linea] URL: http://www.ifr.org/index.asp; [30 de octubre de 2007]

Product and Technology Trends for Industrial Robots; Hadi A. Akeel, Steve W. Holland; Industrial Robots Symposium; 2000; [Archivo PDF]

Industrial Manipulators; TrueForce [Documento en línea]; URL: http://trueforce.com/ [30 de octubre de 2007]

R17 Robot Manual; ST Robotics; Deucalion; diciembre 2006 [Archivo PDF]

Robotronica, Aplicaciones de la Robotica; Marcos Garcia Bartolomé, Jose Mª Álvarez Ontivero, Daniel Cava Jimenez;Universitat Politecnica de Catalunya; junio 2003 [Archivo PDF]

ROVer Ranch; National Aeronautics and Space Administration (NASA); JSC Learning Technologies; enero 2007 [Documento en linea]; URL: http://prime.jsc.nasa.gov/ [30 de octubre de 2007]

Types of Robots; Loop Technology Limited; Inglaterra, 2007; [Documento en línea]; URL: http://www.looptechnology.com/robotic-robot-types.asp [30 de octubre de 2007]

Jim Berge; Welding robots and lean manufacturing learn to play together;; octubre 2005; [Documento en linea]; URL: http://www.thefabricator.com [30 de octubre de 2007]

Soldering Robot; Etneo; Italia; 2004 [Documento en línea]; URL: http://www.etneo.com/lcaten.htm [30 de octubre de 2007]

Dans l'emballage, les robots ont le vent en poupe; ELAU, Schneider Electric; 2006 [Documento en linea]; URL: http://www.elau.de/Rahmen.asp?Knoten=1375 [30 de octubre de 2007]

ROBOT PALLETISING SYSTEMS; WEBSTER GRIFFIN LTD; 2006 [DOCUMENTO EN LÍNEA] URL: HTTP://WWW.WEBSTERGRIFFIN.COM/?PGID=42 [30 DE OCTUBRE DE 2007]

Robótica Industrial; E. U. Politécnica de la Universidad de Sevilla; Depto. de Arquitectura y Tecnología de Computadores [Documento en línea]
URL: http://www.atc.us.es/index.php?op=descargas&pag=4
[30 de octubre de 2007]

Industrial Robots; Robotics Research Group (RRG); The University of Texas at Austin; J.J. Pickle Research Campus; [Documento en línea];
URL: http://www.robotics.utexas.edu/rrg/learn_more/low_ed/types/industrial.html;
[30 de octubre de 2007]

Robot Applications; International Federation of Robotics (IFR); [Documento en linea]
URL: http://www.ifr.org/index.asp;
[30 de octubre de 2007]

Product and Technology Trends for Industrial Robots; Hadi A. Akeel, Steve W. Holland; Industrial Robots Symposium; 2000; [Archivo PDF]

Industrial Manipulators; TrueForce [Documento en línea];
URL: http://trueforce.com/
[30 de octubre de 2007]

R17 Robot Manual; ST Robotics; Deucalion; diciembre 2006 [Archivo PDF]

Robotronica, Aplicaciones de la Robotica; Marcos Garcia Bartolomé, Jose Mª Álvarez Ontivero, Daniel Cava Jimenez;Universitat Politecnica de Catalunya; junio 2003 [Archivo PDF]

ROVer Ranch; National Aeronautics and Space Administration (NASA); JSC Learning Technologies; enero 2007 [Documento en linea];
URL: http://prime.jsc.nasa.gov/
[30 de octubre de 2007]

Types of Robots; Loop Technology Limited; Inglaterra, 2007; [Documento en línea];
URL: http://www.looptechnology.com/robotic-robot-types.asp
[30 de octubre de 2007]

Jim Berge; Welding robots and lean manufacturing learn to play together;; octubre 2005; [Documento en linea];
URL: http://www.thefabricator.com
[30 de octubre de 2007]

Soldering Robot; Etneo; Italia; 2004 [Documento en línea];
URL: http://www.etneo.com/lcaten.htm
[30 de octubre de 2007]

Dans l'emballage, les robots ont le vent en poupe; ELAU, Schneider Electric; 2006 [Documento en linea];
URL: http://www.elau.de/Rahmen.asp?Knoten=1375
[30 de octubre de 2007]

Robot Palletising Systems; Webster Griffin Ltd; 2006 [Documento en línea] URL: http://www.webstergriffin.com/?pgid=42 [30 de octubre de 2007]

Adrio Communications Ltd. (Radio-Electronics.Com). *RS-232 Tutorial*. Recuperado el 7 de Febrero de 2007, de http://www.radio-electronics.com/info/telecommunications_networks/networking/rs232/rs232.php

Akeel, H. A., & Holland, S. W. (2000). *Product and Technology Tends for Industrial Robots.* Industrial Robots Symposium.

Bartolomé, M. G., Alvarez, J. M., & Cava, D. (2003). *Robotrónica, Aplicaciones de la Robótica.* Universitat Politecnica de Catalunya.

Berge, J. (Octubre de 2005). *Welding robots and lean manufacturing learn to play together*. Recuperado el 30 de Octubre de 2007 , de http://www.thefabricator.com

Cisco. (2004). *Channel Deployment Issues for 2.4-GHz 802.11 WLANs*. Recuperado el 11 de Noviembre de 2007, de Cisco Systems: http://www.cisco.com/en/US/docs/wireless/technology/channel/deployment/guide/Channel.html

Cisco. (6 de February de 1996). *TCP/IP*. Recuperado el 6 de Febrero de 2007, de Cisco - TCP/IP: http://www.cisco.com/warp/public/535/4.html

Dpto. de Arquitectura y Tecnología de Computadores. (s.f.). *Robótica Industrial*. Recuperado el 20 de Octubre de 2007, de E. U. Politécnica de la Universidad de Sevilla: http://www.atc.us.es/index.php?op=descargas&pag=4

Etneo. (2004). *Soldering Robot*. Recuperado el 30 de Octubre de 2007, de http://www.etneo.com/lcaten.htm

Flickenger, R. (3 de February de 2001). *802.11b Tips, Tricks, and Facts*. Recuperado el 11 de November de 2007, de O'Reilly Network: http://www.oreillynet.com/pub/a/wireless/2001/03/02/802.11b_facts.html

Flickenger, R. (5 de Marzo de 2001). *A Wireless Long Shot*. Recuperado el 10 de Noviembre de 2007, de O'Reilly Network: http://www.oreillynet.com/pub/a/wireless/2001/05/03/longshot.html

Geier, J. (20 de Junio de 2002). *802.11 WEP: Concepts and Vulnerability*. Recuperado el 11 de Noviembre de 2007, de Wi-Fi Planet: http://www.wi-fiplanet.com/tutorials/article.php/1368661

Gilbert, H. (2 de February de 1995). *Introduction to TCP/IP*. Recuperado el 5 de Febrero de 2007, de http://www.yale.edu/pclt/COMM/TCPIP.HTM

Intel. (s.f.). *IEEE 802.11 standard for WLAN - Intel in Standards*. Recuperado el 21 de Octubre de 2007, de http://www.intel.com/standards/case/case_802_11.htm

International Federation of Robotics. (s.f.). *Robot Applications*. Recuperado el 30 de Octubre de 2007, de http://www.ifr.org/index.asp

IT-Expert On Call. (25 de Octubre de 2007). *Channels and Interference*. Recuperado el 11 de Noviembre de 2007, de IT - Expert On Call: http://www.itexpertoncall.com/mimo/247.html

Kuka Industrial Robots. (29 de January de 2007). *KUKA Robotics showcases new wireless robot control concept at ATX EXPO*. Recuperado el 29 de Octubre de 2007, de Kuka Industrial Robots: http://www.kuka.com/usa/en/pressevents/news/NN_070129_New_Wireless_Robot_Control.htm

Lantronix. (s.f.). *Embedded Wireless Networking*. Recuperado el 11 de Febrero de 2007, de http://www.lantronix.com/device-networking/embedded-device-servers/wiport.html

Lantronix, Inc. (March de 2006). *WiPort Data Sheet.* Recuperado el 28 de Octubre de 2007, de Lantronix Documentation: http://www.lantronix.com/pdf/WiPort_DS.pdf

Lantronix, Inc. (September de 2006). *WiPort User Guide.* Recuperado el 28 de Octubre de 2007, de Lantronix Documentation: http://www.lantronix.com/pdf/WiPort_UG.pdf

Loop Technology Limited. (2007). *Types of Robots*. Recuperado el 30 de Octubre de 2007, de http://www.looptechnology.com/robotic-robot-types.asp

Maxim. (18 de February de 2003). *An Introduction to Direct-Sequence Spread-Spectrum Communications*. Recuperado el 10 de Noviembre de 2007, de Maxim/Dallas: http://www.maxim-ic.com/appnotes.cfm/appnote_number/1890/

Mikroelektronica. (s.f.). *mikroC*. Recuperado el 12 de Febrero de 2007, de http://www.mikroe.com/en/compilers/mikroc/pic/

Mitchell, B. (s.f.). *Wireless Standards - 802.11b 802.11a 802.11g and 802.11n*. Recuperado el 21 de Octubre de 2007, de http://compnetworking.about.com/cs/wireless80211/a/aa80211standard.htm

National Aeronautics and Space Administration (NASA). (enero de 2007). *ROVer Ranch*. Recuperado el 30 de Octubre de 2007, de JSC Learning Technologies: http://prime.jsc.nasa.gov

Nygaard, B. (21 de August de 2002). *802.11standard*. Recuperado el 21 de Octubre de 2007, de http://wlan.nat.sdu.dk/802_11standard.htm

Robotics Research Group (RRG). (s.f.). *Industrial Robotics*. Recuperado el 30 de Octubre de 2007, de The University of Texas at Austin: http://www.robotics.utexas.edu/rrg/learn_more/low_ed/types/industrial.html

Schneider Electric. (2006). *Dans l'emballage, les robots ont le vent en poupe*. Recuperado el 30 de Octubre de 2007 , de http://www.elau.de/Rahmen.asp?Knoten=1375

ST Robotics. (2006). *R17 Robot Manual.* Deucalion.

Strangio, C. E. (s.f.). *The RS232 Standard*. Recuperado el 6 de Febrero de 2007, de CAMI Research Inc.: http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html

TCP/IP Guide. (20 de September de 2005). *IP Datagram General Format*. Recuperado el 6 de Febrero de 2007, de TCP/IP Guide: http://www.tcpipguide.com/free/t_IPDatagramGeneralFormat.htm

TCP/IP Guide.com. (20 de Septiembre de 2005). *Interfaces: Vertical (Adjacent Layer) Communication* . Recuperado el 11 de Noviembre de 2007, de TCP/IP Guide.com: http://www.tcpipguide.com/free/t_InterfacesVerticalAdjacentLayerCommunication.htm

TCP/IPGuide.com. (20 de September de 2005). *IP Datagram Encapsulation*. Recuperado el 6 de Febrero de 2007, de TCP/IP Guide: http://www.tcpipguide.com/free/t_IPDatagramEncapsulation.htm

The Industrial Ethernet Book. (s.f.). *Case Study: Robot cell uses wireless I/O*. Recuperado el 29 de Octubre de 2007, de The Industrial Ethernet Book: http://ethernet.industrial-networking.com/articles/articledisplay.asp?id=1871

The Industrial Wireless Book. (s.f.). *Ethernet radios create safer environment for workers*. Recuperado el 29 de Octubre de 2007, de The Industrial Wireless Book: http://wireless.industrial-networking.com/articles/articledisplay.asp?id=1972

TrueForce. (s.f.). *Industrial Manipulators*. Recuperado el 30 de Octubre de 2007, de TrueForce: http://trueforce.com

Vos, M. (s.f.). *Intro - CSMA/CA*. Recuperado el 10 de Noviembre de 2007, de Wireless LAN - Testing the 802.11 MAC Protocol: http://www.science.uva.nl/research/air/projects/old_projects/wlan/simulations/Intro_-_WLAN/Intro_-_CSMA_CA/intro_-_csma_ca.html

Webster Griffin Ltd. (s.f.). *Robot Palletising Systems*. Recuperado el 30 de Octubre de 2007, de http://www.webstergriffin.com/?pgid=42

ZYREN, J. (6 DE DECEMBER DE 2001). *IEEE 802.11G EXPLAINED*. RECUPERADO EL 10 DE NOVIEMBRE DE 2007, DE INTERSIL CORPORATION: HTTP://FORSKNINGSNETT.UNINETT.NO/WLAN/DOWNLOAD/WP_IEEE802GEXPLA_12_06.PDF

X. APÉNDICES

A. Historia

A continuación se presentan algunos hechos históricos de importancia para la robótica.

- 350 A.C.: El matemático griego Arquitas de Tarento construye un pájaro mecánico propulsado con vapor al cual llama "la Paloma". Constituye uno de los estudios más antiguos del vuelo, además de ser probablemente el primer aeroplano a escala en la historia.
- 322 A.C.: El filósofo griego Aristóteles escribe: "Si cada herramienta, al ser ordenada, o de su propia voluntad, pudiera hacer el trabajo que le toca… no habría necesidad ni de aprendices para los maestros ni de esclavos para los dueños." Lo cual explica la utilidad de la existencia de los robots.
- 200 A.C.: El inventor y físico griego Tesibio de Alejandría diseña relojes de agua llamados clepsidras que contienen figuras móviles. Los relojes de agua son un gran avance en la medición del tiempo. Hasta entonces, los griegos utilizaban relojes de arena que debían voltearse una vez la arena había pasado completamente a uno de los lados. Esta invención cambió esta noción ya que medía el tiempo como el resultado de la fuerza del agua cayendo constantemente a través del reloj con un flujo constante. En general, los griegos estaban fascinados con autómatas de todos tipos, usándolos frecuentemente para producciones teatrales y ceremonias religiosas.
- 1495: Leonardo Da Vinci diseña un aparato mecánico que parece un caballero con armadura. Los mecanismos dentro del "Robot de Leonardo" están diseñados para mover al caballero como si hubiera una persona de verdad dentro de la armadura. Los inventores de la Edad Media a menudo construían máquinas de este tipo para divertir a la realeza.
- 1738: Jacques de Vaucanson comienza a construir autómatas en Grenoble, Francia; construye tres en total. El primero fue un flautista que tocaba doce canciones. El segundo

vino rápidamente y además de la flauta tocaba un tambor o pandereta; el tercero fue el más famoso de todos, el pato fue un ejemplo del intento de Vaucanson de lo que él mismo llamó "anatomía motriz", es decir que modelaba la anatomía de humanos y animales con la mecánica. El pato se movía, emitía sonidos, batía sus alas y asombrosamente comía y digería alimentos.

- 1770: Los relojeros suizos e inventores del reloj de pulsera moderno, Pierre Jaquet-Droz y luego su hijo Henri-Louis Jaquet-Droz, comienzan a construir autómatas para la realeza europea. Crean tres muñecos en total, cada uno con una función única: el primero podía escribir, otro interpretaba piezas musicales y el tercero dibujaba.
- 1801: Joseph Jacquard construye un telar automático que se controla con tarjetas perforadas. Las tarjetas perforadas son usadas mucho después como método de entrada para las computadoras más primitivas del siglo XX.
- 1822: Charles Babbage demuestra un prototipo de su "Motor Diferencial" a la Real Sociedad Astronómica. Continúa su trabajo al diseñar un proyecto aún más ambicioso llamado "Motor Analítico" que al parecer utilizaba tarjetas perforadas, inspirado por el invento de Jacquard. Durante su vida nunca produjo una versión funcional de ninguna de las dos máquinas. A pesar de esto es frecuentemente considerado como el padre de la computadora y su trabajo sobrevive como la base del sistema de numeración binario que a su vez es la base de las computadoras modernas.
- 1847: George Boole representa la lógica en forma matemática con su álgebra booleana.
- 1898: Nikola Tesla construye y demuestra un barco operado por control remoto en el Madison Square Garden en Nueva Cork.
- 1921: El escritor checo Karel Capek introduce la palabra "robot" en su obra teatral "R.U.R." (Rossuum's Universal Robots). La palabra, en checo, viene de la palabra "robota" que quiere decir "trabajo tedioso". Capek propone un paraíso donde las máquinas inicialmente traen muchos beneficios pero al final producen mucho mal en la forma de desempleo e inestabilidad social. Estos robots, curiosamente, no eran de naturaleza mecánica sino química.

- 1936: Alan Turing introduce el concepto de una computadora teórica llamada la Máquina de Turing, la cual es un avance fundamental en la lógica de las computadoras y también da lugar a una nueva escuela dentro de las matemáticas.
- 1938: El primer mecanismo para un sistema de pintura programable es diseñado por Willard Pollard y Harold Roselund para la compañía DeVilbiss.
- 1940: Isaac Asimov generalmente se lleva el crédito de la popularización del término "robótica", el cual se mencionó por primera vez en una de sus historias, pero probablemente su contribución mas grande fue la creación de las Tres Leyes de la Robótica:
    - Un robot no debe causar daño a un ser humano o, a través de la inacción, permitir que el daño llegue a un ser humano.
    - Un robot debe obedecer las órdenes que le son dadas por los seres humanos, excepto cuando tales órdenes estén en conflicto con la Primera Ley.
    - Un robot debe proteger su propia existencia siempre y cuando esta protección no entre en conflicto con la Primera o Segunda Ley.
    - Asimov luego añadió una "ley cero" a la lista: Un robot no debe dañar a la humanidad o, a través de la inacción, permitir que el daño llegue a la humanidad.
- 1946: George Devol obtiene la patente de un dispositivo de control para máquinas usando grabación magnética. J. Presper Eckert y John Mauchly constuyen ENIAC, la primera computadora electrónica, en la universidad de Pennsylvania. En MIT, Whirlwind, la primera computadora de propósito general, resuelve su primer problema.
- 1948: Norbert Wiener, profesor de MIT, publica "Cybernetics or Control and Communication in the Animal", donde describe el concepto de comunicación y control en sistemas mecánicos, electrónicos y biológicos.
- 1950: Alan Turing publica "Computing Machinery and Intelligence" en el cual propone una prueba para determinar si una máquina ha adquirido el poder de pensar por sí misma y a esta se le conoce como la "prueba Turing".

- 1954: George Devol diseña el primer robot programable y acuña el término “universal automation”, inspiración para el nombre de su futura compañía Unimation.
- 1956: Alan Newell y Herbert Simon crean el Teórico Lógico, el primer “sistema experto”. Es utilizado para ayudar en la resolución de problemas matemáticos muy difíciles. Con la ayuda de una concesión monetaria de la Fundación Rockefeller, John McCarthy, Marvin Minsky, Nat Rochester y Claude Shannon organizan el proyecto de investigación de verano de Dartmouth sobre inteligencia artificial, en la universidad de Dartmouth. Como resultado de esta conferencia se acuña el término “inteligencia artificial”.
- 1959: John McCarthy y Marvin Minsky fundan el laboratorio de inteligencia artificial en el Massachussets Institute of Technology (MIT).
- 1961: Heinrich Ernst desarrolla el MH-1, una mano mecánica operada por computadora, en MIT.
- 1962: El primer brazo robótico industrial, llamado Unimate, es promocionado en el mercado. Está diseñado para completar tareas repetitivas o peligrosas en la línea de ensamble de General Motors.
- 1963: John McCarthy deja MIT para fundar el laboratorio de inteligencia artificial en la universidad de Stanford.
- 1965: Las transformaciones homogéneas se aplican a la cinemática de los robots. Esta sigue siendo la base de la teoría de robots actualmente.
- 1966: El Instituto de Investigación de Stanford (conocido luego como SRI Technology) desarrolla a Shakey, el primer robot móvil que conoce y reacciona a sus propias acciones. Entre otros logros SRI es el instituto que ayudó a traer el detergente moderno al desarrollar Tide. Un programa de inteligencia artificial llamado ELIZA es creado en MIT por Joseph Weizembaum. ELIZA funciona como un psicólogo computacional que manipula las sentencias de sus usuarios para formar preguntas. Weizembaum se perturba al ver qué tan rápido las personas desarrollan fe en su pequeño programa.

- 1967: Richard Greenblatt escribe MacHack, un programa que juega al ajedrez, como respuesta a un artículo reciente escrito por Hubert Dreyfus donde sugería, como crítica a los esfuerzos en la inteligencia artificial, que un programa de computadora jamás podría vencerlo en un juego de ajedrez. Cuando el programa es finalizado y se invita a Dreyfus para jugar contra la computadora, lleva la delantera durante casi todo el juego pero al final pierde luego de un juego cerrado. El programa de Greenblatt se convirtió en la base para muchos programas de ajedrez, culminando en el programa de la computadora Big Blue, que derrota al gran maestro de ajedrez Gary Kasparov.
- 1968: Kawasaki licencia un diseño de robots hidráulicos y comienza su producción en Japón.
- 1969: Victor Scheinman, un estudiante de ingeniería mecánica que trabaja en el laboratorio de inteligencia artificial de Stanford, crea el Brazo Stanford. El diseño del brazo se convierte en un estándar y aun tiene influencia sobre el diseño de brazos robóticos hoy en día.
- 1970: La universidad de Stanford produce el Stanford Cart. Está diseñado para ser un seguidor de líneas pero también puede ser controlado por una computadora a través de un enlace de radio.
- 1974: Victor Scheinman forma su propia compañía y comienza a mercadear el Silver Arm, el cual es capaz de ensamblar partes pequeñas usando sensores de tacto.
- 1976: Shigeo Hirose diseña el Sofá Gripper en el instituto de tecnología de Tokio. Está diseñado para envolver un objeto como una serpiente. Se utilizan brazos robóticos en las sondas espaciales Viking 1 y 2.
- 1977: Los exploradores del espacio Voyager 1 y 2 son lanzados desde el Centro Espacial Kennedy.
- 1979: Se establece el Instituto de Robótica en la universidad Carnegie Mellon. El Stanford Cart es reconstruido por Hans Moravec. Añade un sistema de visión más robusto, lo cual permite una autonomía mayor. Estos son los primeros experimentos con el mapeo de un ambiente en tres dimensiones.

- 1980: Seymour Papera publica “Mindstorms: Niños, Computadoras e Ideas Poderosas” donde apoya el construccionismo, o aprender a través de hacer.
- 1981: Takeo Kanade construye el brazo de manejo directo. Es el primero en tener motores instalados directamente en las articulaciones del brazo. Este cambio lo hace más rápido y mucho más preciso que los brazos robóticos anteriores.
- 1986: LEGO y el MIT Media Lab colaboran para realizar los primeros productos educacionales basados en LEGO, llegando a miles de escuelas primarias. Honda comienza un programa de investigación robótica con la premisa de que “el robot debe coexistir y cooperar con los seres humanos, haciendo lo que una persona no puede hacer y cultivando una nueva dimensión en movilidad para beneficiar a la sociedad”.
- 1989: Un robot que camina llamado Genghis es mostrado por el Mobile Robots Group de MIT y se vuelve famoso por su manera peculiar de caminar.
- 1992: En un intento por construir una aspiradora controlada por radio, Marc Thorpe tiene la idea de comenzar un evento de combate de robots. El doctor John Adler desarrolla el concepto del CyberKnife, un robot que toma imágenes de rayos X de un paciente y busca tumores, dando una dosis planeada de radiación al tumor cuando lo encuentra.
- 1993: Dante, un robot caminante de 8 patas desarrollado en la universidad de Carnegie Mellon, se interna en el monte Erebrus en la Antártica; su misión era recolectar datos sobre un ambiente hostil, como sería el de otro planeta, pero la misión falla cuando, luego de una caída corta de 20 pies, la cuerda que sostiene a Dante se rompe y cae dentro del cráter.
- 1994: Dante II, una versión más robusta de su predecesor, entra en el cráter del volcán Spurr, en Alaska y esta misión se considera un éxito. Marc Thorpe inicia Robot Wars en el centro Fort Mason de San Francisco, California.
- 1995: El segundo evento anual Robot Wars toma lugar en el centro Fort Mason.
- 1996: RoboTuna es diseñado y construido por David Barrett para su tesis doctoral en MIT. Se utiliza para estudiar la manera en la que los peces nadan. Luego de un accidente, Chris Campbell y Stuart Wilkinson toman una nueva inspiración y de su trabajo resulta

Gastrobot, un robot que digiere materia orgánica para producir dióxido de carbono que luego se usa para propulsión. En ese mismo año, Honda revela el P3, fruto del esfuerzo de una década para construir un robot humanoide.

- 1997: El primer nodo de la Estación Espacial Internacional se coloca en órbita. Durante los próximos años se adjuntan más componentes, incluyendo un brazo robótico diseñado por la compañía canadiense MD Robotics. La misión Pathfinder llega a Marte y su explorador robótico Sojourner baja de una rampa a suelo marciano a principios de julio y continúa enviando datos de la superficie marciana hasta septiembre.
- 1998: La industria de la robótica se infiltra en el mercado de juguetes ya que Tiger Electronics introduce Furby y rápidamente se convierte en el juguete más popular de la temporada porque utilizando varios sensores, la mascota puede reaccionar a su ambiente y comunicarse usando más de 800 frases en inglés y en su propio lenguaje, Furbish. LEGO también lanza su primer Robotics Invention System 1.0, al cual le llaman "Mindstorms" en honor al trabajo de Seymour Papert en 1980.
- 1999: LEGO lanza el Robotics Discovery Set, Droid Developer Kit y el Robotics Invention System 1.5. SONY lanza la mascota robótica AIBO.
- 2000: Honda lanza su nuevo robot humanoide, ASIMO. El evento Battlebots, con mayor importancia, toma lugar en Las Vegas, Nevada. LEGO lanza el nuevo sistema Mindstorms Robotics Invention System 2.0.
- 2001: LEGO lanza el sistema Mindstorms Ultimate Builder's Set. En agosto de ese mismo año, el CyberKnife, un robot para el campo biomédico, es aprobado para tratar tumores en cualquier parte del cuerpo.
- 2002: El robot ASIMO de Honda suena la campana inicial en la bolsa de Nueva York.
- 2003: NASA lanza las misiones MER-A y MER-B con destino a Marte. SONY lanza su mascota robótica de tercera generación, AIBO ERS-7.
- 2004: Las dos misiones de la NASA llegan a Marte.
- 2005: Honda estrena el nuevo robot ASIMO.

- 2006: Honda y ATR desarrollan satisfactoriamente la interfaz para el control de robots desde el cerebro humano.
- 2007: KASPAR, un robot humanoide del tamaño de un niño, permite que un grupo de investigadores británicos estudie el uso de sistemas robóticos como herramientas para la educación y tratamiento de las habilidades sociales de niños autistas.

O.Imágenes de diseño de la silla operativa

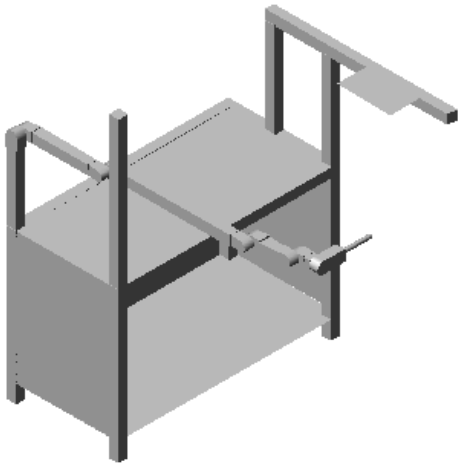


Gráfico 8 - Vista Isométrica Sur Oeste

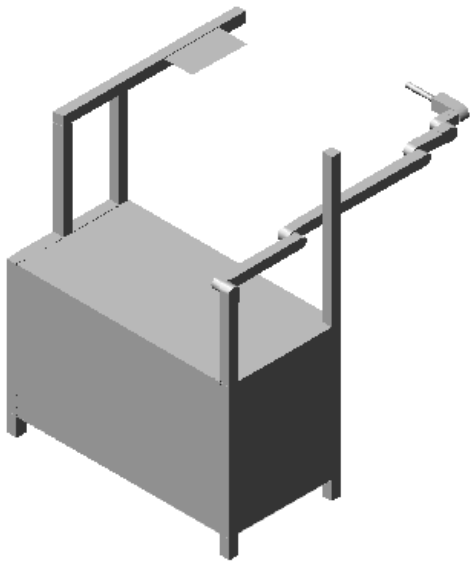


Gráfico 9 - Vista Isométrica Norte Oeste

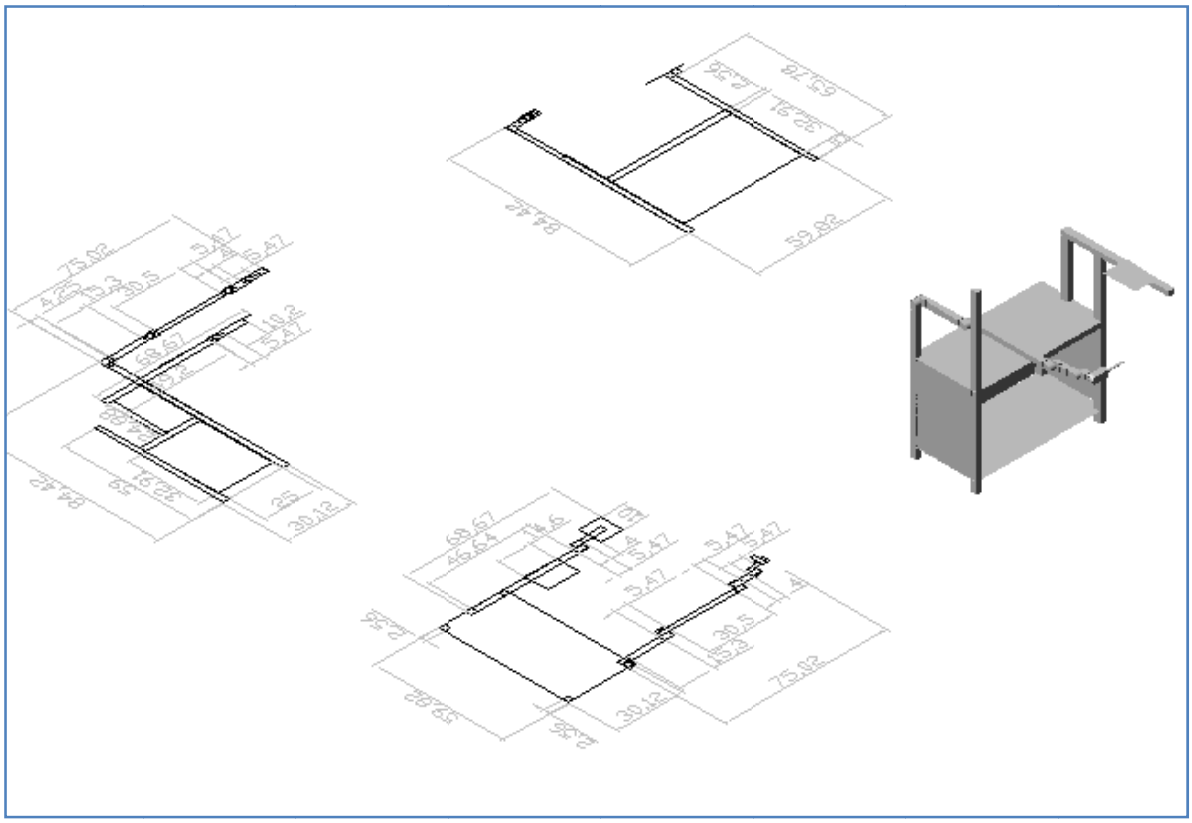


Gráfico 10 - Todas las vistas silla operativa

Imágenes de las partes de la interfaz humana de la silla operativa

Gráfico 11. Brazo y cintura

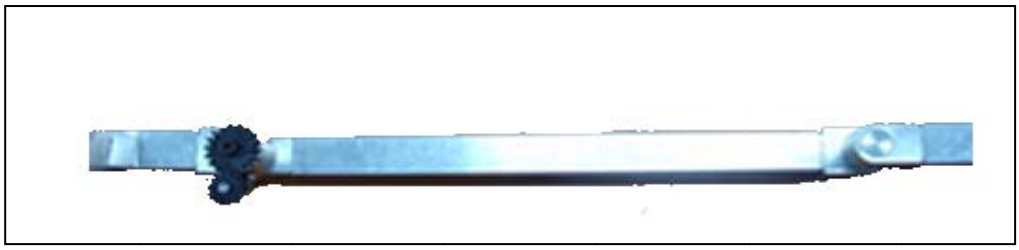


Gráfico 12. Antebrazo

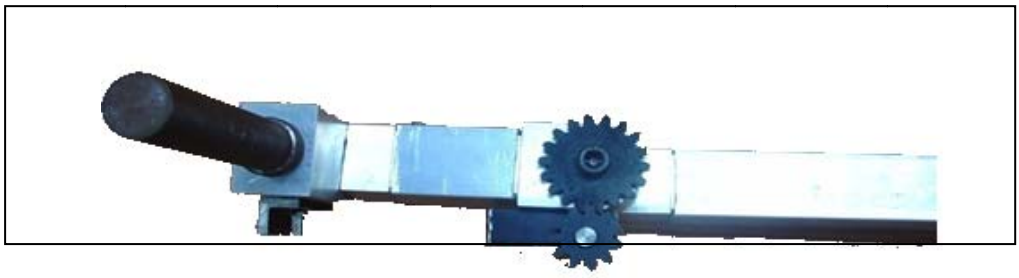


Gráfico 13. Mano y muñeca

Imágenes de la silla operativa

Gráfico 14. Vista Isométrica Sur Oeste

Gráfico 15 - Vista Isométrica Norte Este

Gráfico 16. Vista Isométrica Norte Oeste

P. Sensores

```
//Sistema de aprendizaje remoto para robot R17
 //Procesamiento - Brujula

 void main() {
   TRISD=0;
   I2C_Init(100000);
   I2C_Start();
   I2C_Wr(0xC0);
   delay_us(50);
   I2C_Wr(15);
   I2C_Repeated_Start();
   I2C_Wr(0xC1);
   I2C_Wr(0xFF);
   PORTD=1;

   I2C_Repeated_Start();
   I2C_Wr(0xC0);
   delay_us(50);
   I2C_Wr(14);
   I2C_Repeated_Start();
   I2C_Wr(0xC1);
   while(I2C_Rd(0)==0);
   PORTD=0;
```

```
   I2C_Repeated_Start();
   I2C_Wr(0xC0);
   delay_us(50);
   I2C_Wr(15);
   I2C_Repeated_Start();
   I2C_Wr(0xC1);
   I2C_Wr(0);

   I2C_Stop();
}//~!
```

B. Código de lectura y creación de trama

```
//Sistema de aprendizaje remoto para robot R17
 //Procesamiento - Brujula
```

```
 char grados_str[4];
 int grados1,grados2,grados_old;
 double gradosdbl;

 void lectura() {
   long grados;
   I2C_Start();
   I2C_Wr(0xC0);
   delay_us(50);
   I2C_Wr(2);
   I2C_Repeated_Start();
   I2C_Wr(0xC1);
   //PORTD = I2C_Rd(0);
   grados1=I2C_Rd(0); //Lectura de 8 MSB
   I2C_Repeated_Start();
   I2C_Wr(0xC0);
   delay_us(50);
   I2C_Wr(3);
   I2C_Repeated_Start();
   I2C_Wr(0xC1);
   grados2=I2C_Rd(0); //Lectura de 8 LSB
   I2C_Stop();
   //PORTD=grados;
   grados=grados1*256;
   grados+=grados2;   //paso de los 2bytes a un entero
   //Si la lectura esta fuera del rango de la antigua
   //lectura se transmite.
   gradosdbl=grados/10;
   gradosdbl=gradosdbl*0.9735-1.0249;
   grados=gradosdbl*10;
   if (grados>grados_old+5) || (grados<grados_old-5) {
     InttoStr(grados,grados_str);
     grados1=grados >> 8; //8 bits mas significativos
     grados2=grados << 8; //8 bits menos significativos
     //enviar datos
     Lcd_Out(2,1,grados_str);
   }
   grados_old=grados;
 }

char *text = "Lectura:";

void main() {
  TRISD=0;
  I2C_Init(100000);
  Lcd_Init(&PORTB);
  Lcd_Cmd(LCD_CLEAR);
  Lcd_Cmd(LCD_CURSOR_OFF);
  Lcd_Out(1,1, text);
  grados1=grados2=0;

  do {
     lectura();
     delay_ms(100);
  }while (1);
}//~!
```

A. Programa de Captura de Datos

```
#include <18F458.h>
#device adc=10                //Convertidor ADC 10bits
#include <can4r17.c>        //Libreria CAN
```

```
#FUSES NOWDT                 //No Watch Dog Timer
#FUSES WDT128                //Watch Dog Timer usa postscale 1:128
#FUSES HS                     //High speed Osc (> 4mhz)
#FUSES NOPROTECT          //Codigo No protegido
#FUSES NOOSCSEN          //Oscilador Primario
#FUSES BROWNOUT               //Reset when brownout detected
#FUSES BORV20                 //Brownout reset at 2.0V
#FUSES PUT                    //Power Up Timer
#FUSES NOCPD                  //No existe proteccion de EE
#FUSES STVREN                 //reset en Stack full/underflow
#FUSES NODEBUG                //No Debug mode for ICD
#FUSES NOLVP                  //No  low voltage prgming,  B3(PIC16)  or  B5(PIC18)
   used for I/O
#FUSES NOWRT                  //Program memory not write protected
#FUSES NOWRTD                 //Data EEPROM not write protected
#FUSES NOWRTB                 //Boot block not write protected
#FUSES NOCPB                  //No Boot Block code protection
#FUSES NOWRTC                 //configuration not registers write protected
#FUSES NOEBTR                 //Memory not protected from table reads
#FUSES NOEBTRB                //Boot block not protected from table reads

#use delay(clock=40000000)    //Reloj Primario
//************************CREACION DE VARIABLES*******************************
EXTERN  INT16  CONVX,CONVY,DATA;        //Variables  usadas  para  la  conversión  y
   envío de datos
EXTERN  INT  COUNT,ENV[2],CONTADOR;    //counter  de  promedio,  variables  de  envio
   CAN y contador CAN
EXTERN INT1 FLAG;                    //Flag que indica el canal de conversion
EXTERN FLOAT ACUMX,ACUMY;            //Acumuladores para el promedio
//***********************FIN CREACION DE VARIABLES*****************************


//*****************+**********INTERRUPTOS************************************
#int_EXT
EXT_isr()                            //RELOJ DE CONVERSIÓN
{
   ACUMX/=COUNT;                     //Promedio de los Valores leídos en el canal
   X
   ACUMY/=COUNT;                     //Promedio de los Valores leídos en el canal
   Y
   IF (ACUMY>512)                      //Si el canal Y es mayor que la mitad del
   voltaje de conversión
      ACUMX=2047-ACUMX;               //Entonces se mide la otra mitad del canal
   de conversión X
   DATA=(INT16)ACUMX;                //Casting de Float a INT 16
   ENV[1]=DATA>>8;                   // 8 bits mas significativos a ENV1
   bit_set(ENV[1],5);                //Header de Sensor segun su ubicación
   DATA<<=8;
   ENV[0]=DATA>>8;                   // 8 Bits menos significativos a ENV0
   ACUMX=0;                          //Reiniciacion de variables
   ACUMY=0;
   COUNT=0;
   CAN_ENVIAR(&CONTADOR,wifiTX,ENV);//Envio a través de la red CAN
}

#int_AD
AD_isr()
{
   IF(FLAG)                          //Canal 0 Acumulador X
   {
      ACUMX+=CONVX;
   }
   ELSE
   {
      ACUMY+=CONVY;                  //Canal 1 Acumulador Y
   }
}
```

```
//*************************FIN INTERRUPTOS************************************

void main()
{
//*******************INIALIZACION DE PIC E INTERRUPTOS************************
   setup_adc_ports(AN0_AN1_VREF_VREF);
   setup_adc(ADC_CLOCK_DIV_2);
   setup_psp(PSP_DISABLED);
   setup_spi(FALSE);
   setup_wdt(WDT_OFF);
   setup_timer_0(RTCC_INTERNAL);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);
   enable_interrupts(INT_AD);
   enable_interrupts(INT_EXT);
   enable_interrupts(GLOBAL);
   setup_low_volt_detect(FALSE);
   setup_oscillator(False);
 //*****************************************************************************
   CAN_INIT();                                 //Inializacion de red CAN
   COUNT=0;                                    //Inializacion de Variables
   ACUMX=0;
   ACUMY=0;
   CONTADOR=0;
//*************************PROGRAMA PRINCIPAL*********************************
   DO
   {
      SET_ADC_CHANNEL(0);
      DELAY_US(10);
      FLAG=0;
      CONVX=READ_ADC();                   //Se inicia la conversion en el eje X
      SET_ADC_CHANNEL(1);
      DELAY_US(10);
      FLAG=1;
      CONVY=READ_ADC();                   //Se inicia la conversion en el eje Y
      COUNT++;                            //Se cuenta el numero de conversiones
   }                                      //realizadas
   WHILE(TRUE);
 //*************************FIN PROGRAMA PRINCIPAL*****************************

}
```

Q.ALDO

A. Código fuente para la programación de la muñeca.

```
#include "C:\Program Files\PICC\uvg\prueba18F458.h"
#include <stdio.h>
int16 T_sample;           // variable para periodo de muestreo
int16 wrist;
int16 temp1;
int16 temp2;
int1 bandera1;

void main() {
int16 ID_wrist;
int16 ajuste;
```

```
int16 aux_wrist;

//CONFIG/////////////////////////
   setup_adc_ports(ALL_ANALOG);
   setup_adc(ADC_CLOCK_INTERNAL);
   setup_psp(PSP_DISABLED);
   setup_spi(FALSE);
   setup_wdt(WDT_OFF);
   setup_timer_0(RTCC_INTERNAL);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
   setup_comparator(D1_VR_D3_VR);
   setup_vref(FALSE);
   enable_interrupts(INT_COMP);
   enable_interrupts(GLOBAL);
   setup_low_volt_detect(FALSE);
   setup_oscillator(False);
////////////////////////////////////
//VAR INIT//////////////////////////
   T_sample = 1;
   ID_wrist = 12288;                                  //011   0    0    0000000000
   //ID   SIG ADJ DAT
   ajuste = 981;
   temp1 = 0;
   temp2 = 0;
   bandera1=FALSE;
//START: IF NOT OVERFLOW//////////////
      while(TRUE){
        if (bandera1==FALSE){
            do{
//         medicion_wrist:
             set_adc_channel(2);
             delay_us(8);
             MA3_1 = read_adc();                    // variable para encoder 1
            temp1 = MA3_1;                          // guardo estado anterior
            wrist = temp1 + ID_wrist;                //ajusta valor de la trama
   para enviarlo
//SHOW VARS HYPERTERMINAL//////////////
            printf("MA3%LU temp1%LU wrist%LU\r\n\f", MA3_1,temp1,wrist);
//        printf("k%LU temp1%LU hand%LU\r\n\f", k1,temp1,hand);
           delay_us(T_sample);
           }while ((MA3_1>=3) && (MA3_1<=979));
//        delay_us(100);
            }//end if
}  //end program
```

B. Código fuente para la programación de la mano.

```
#include "C:\Program Files\PICC\uvg\prueba318F458.h"
#include <stdio.h>
#int_COMP

//VARIABLES GLOBALES
int16 k1;
int16 k2;              // variable para primer encoder
int16 T_sample;           // variable para periodo de muestreo

int16 temp1;
int16 temp2;
int1 bandera;

COMP_isr() {
if (bandera==FALSE){
   bandera == TRUE;
   }
bandera == FALSE;
}

void main() {

//VARIABLES LOCALES
int16 hand;
int16 ID_hand;
int16 ajuste;
//int16 aux_hand;

//CONFIGURACION GENERAL
   setup_adc_ports(ALL_ANALOG);
   setup_adc(ADC_CLOCK_INTERNAL);
   setup_psp(PSP_DISABLED);
   setup_spi(FALSE);
   setup_wdt(WDT_OFF);
   setup_timer_0(RTCC_INTERNAL);
   setup_timer_1(T1_INTERNAL|T1_DIV_BY_1);
   setup_timer_2(T2_DISABLED,0,1);
   setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
   setup_comparator(D1_VR_D3_VR);
   setup_vref(FALSE);
   enable_interrupts(INT_COMP);
   enable_interrupts(GLOBAL);
   setup_low_volt_detect(FALSE);
   setup_oscillator(False);

//VAR INIT////////////////////////////
   T_sample = 1;
   ID_hand = 12288;                                  //011   0    0    0000000000
   //ID   SIG ADJ DAT
   ajuste = 971;
   temp1 = 0;
   temp2 = 0;
   bandera = FALSE;

while(TRUE){
//CHEQUEO DE OVERFLOW
      do{
      set_adc_channel(2);                //  D0_D3_D1_D2 // CANAL A2
      delay_us(8);
      k1 = read_adc();                   // variable para encoder 1
```

```
      temp1 = k1;                          // guardo estado anterior
      hand = temp1 + ID_hand;                 //ajusta valor de la trama para
   enviarlo
      printf("k1%LU temp1%LU hand%LU\r\n\f", k1,temp1,hand);
      delay_ms(T_sample);
      }while(bandera == FALSE);
      delay_ms(T_sample);
      do{
      set_adc_channel(2);
      delay_us(8);
      k2 = read_adc();                     // variable para encoder 1
      temp2 = k2;                          // guardo estado anterior
      hand = temp2 + ID_hand + ajuste;              //ajusta valor de la trama
   para enviarlo
      printf("k2%LU temp2%LU hand%LU\r\n\f", k2,temp2,hand);
      delay_ms(T_sample);
      }while(bandera == TRUE);
}

}
```

C. Código fuente para la programación de la mano y la muneca

```
#include "C:\Program Files\PICC\uvg\final18F458.h"
#include <can4r17.c>
#include <stdio.h>

//GLOBALES////////////////////////////////
int16 k1;
int16 temp1;
int16 k2;
int16 temp2;

//PRINCIPAL//////////////////////////////
void main() {

//LOCALES////////////////////////////////
int16 mano;
int16 ID_hand;
int16 muneca;
int16 ID_wrist;
int8 contador;
int8 menos1, menos2, mas1, mas2;
int8 data1[2];
int8 data2[2];

//PIC - CONFIG////////////////////////////
   setup_adc_ports(ALL_ANALOG);
   setup_adc(ADC_CLOCK_INTERNAL);
   setup_psp(PSP_DISABLED);
   setup_spi(FALSE);
   setup_wdt(WDT_OFF);
   setup_timer_0(RTCC_INTERNAL);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);
```

```
   setup_low_volt_detect(FALSE);
   setup_oscillator(False);
   can_start(hand);

//VAR-INIT//////////////////////////
   ID_hand = 12288;                         //011  0   0   0000000000
   ID_wrist = 16384;                        //100  0   0   0000000000
   temp1 = 0;
   temp2 = 0;

while(TRUE){
      set_adc_channel(2);                 //  D0_D3_D1_D2 // CANAL A2
      delay_us(8);
      k1 = read_adc();
      temp1 = k1 * 4;
      mano = temp1 + ID_hand;
      data1[1] = mano;
      menos1 = data1[1];
      data1[0] = mano >> 8;
      mas1 = data1[0];
      delay_ms(5);
      printf("k1=%LU menos1=%U mas1=%U\n", k1,menos1,mas1);
      can_enviar(&contador,wifiTx,data1);
      delay_ms(10);
      set_adc_channel(3);
      delay_us(8);
      k2 = read_adc();
      temp2 = k2 * 4;
      muneca = temp2 + ID_wrist;
      data2[1] = muneca;
      menos2 = data2[1];
      data2[0] = muneca >> 8;
      mas2 = data2[0];
      delay_ms(5);
      printf("k2=%LU menos2=%U mas2=%U\r\n\f", k2,menos2,mas2);
      can_enviar(&contador,wifiTx,data2);
      delay_ms(10);
      }
}
```

Monitoreo

```
////////////////////////////////////////////////////////////////////////
////
////                        Main_SubModulo_Monitoreo.c
////  DESCRPICION: Archivo principal del Submódulo de Monitoreo
////  PIC 18F458 - CCS C COMPILER
////
////  NOMBRE: JULIO CESAR RUIZ HERRERA
////   CARNE: 03038
////   FECHA: NOVIEMBRE 2007
////
////////////////////////////////////////////////////////////////////////


#include <18F458.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#define FAST_GLCD


//////////////////////////////DEFINE//////////////////////////
```

```
#define Punt_Ent 20
#define Last_mp 3

#define Initx_mp 24
#define Inity_mp 21
#define Initx_cs 20
#define Inity_cs 12

#define Initx_cc  1
#define Initx_cct 55
#define Inity_cc  18
#define Space_cc  10

#define Initx_ez  1
#define Initx_ezt 55
#define Inity_ez  13
#define Space_ez  10

#define Initx_vh  5
#define Initx_vht 55
#define Inity_vh  13
#define Space_vh  11

#define eeclave  85
#define eelang   142
#define eemodule 143

//#define lcd10 58


#define Spacey_mp 11

#define Inity_bar 10
#define Space_bar 13 //  =(63-10-2)/4

#define Leftbar TRUE

#define skip_tresh 5
#define clave_limit 6


////////////////////////////USES//////////////////////////

#USE FAST_IO(B)
#use fast_io(D)
#use fast_io(C)

#USE RS232(BAUD=2400, XMIT=PIN_C6, RCV=PIN_C7)


////////////////////////////INCLUDES//////////////////////////

//INCLUDES PREDEFINIDOS
#include <ctype.h>
#include <string.h>
#include <stdlib.h>

//INCLUDES DESARROLLADOS


//Definiciones Globales:
#include <Definiciones.c>

//GLCD:
#include <GLCD_Hardware.c>
#include <GLCD_Software.c>
#include <math.h>
```

```
//Input:
#include <Botones_Debounce.c> // Botones
#include <Clockit.c>
//keypad //#include <Control_Key.c>

//Varias:
#include <Funciones_Generales.c>

//Menus Display:
#include <Disp_Menus_mod.c>

#include <Drabo.c>
#include <Menu_Main.c>
#include <Menu_Mon.c>
#include <Menu_Conf.c>
#include <Menu_Conex.c>
#include <Menu_Recording.c>


//#include <LosMenus.c>
#include <can4r17.c>
#include <CANopy.c>
//#rom int 0xf00055 = {'9','1','1',' ',' ',' ','\0'}

#use RS232(xmit = pin_C6, rcv = pin_C7, baud=9600)

////////////////////////////// MAIN ////////////////////////////
void main() {

   //VARS
   //char  Titulo[] = "by Julio Ruiz";
   long  tilter_c = 0;

   short CAN_flag=0;

////////////////////////////// INIT ////////////////////////////

   //INIT
   set_tris_d(0);
   set_tris_c(128);

   output_c(0);

   //INPUT INIT
      //kbd_init();
      set_tris_a(0x30);
      set_tris_b(0xF0);
      output_b(0x00);

   //GLCD
      glcd_init(ON);

   //Timer (Clock)
   int_tmr2_count = TMR2_PORSEC;
   setup_timer_2(T2_DIV_BY_16, 209, 16);
   set_timer2(0);

//////////////////////////////INTERRUPTS////////////////////////////


   enable_interrupts(int_rb);
   enable_interrupts(int_timer2);
   enable_interrupts(global);

//////////////////////////////Inicializacion Programa ///////////////

   //IP default
   ipt[0] = 192;
```

```
   ipt[1] = 168;
   ipt[2] = 1;
   ipt[3] = 16;

   get_eeprom_vars();

                //Initial Requests por CAN_tx

   //HACIA PC
   Handshake();

   Rq_Hour();

   //HACIA WIP
   Wip_IP();



////////////////////////////Main Bucle ////////////////////////////
   for(;;) {

int Rjoint_r =0;  // Articulaciones Deshabilitadas
int Rjoint_y =0;  // Articulaciones Problemas

      //IF CAN RX
      Can_Flag = CanRx(Display, &contador, Can_RX);
      if (Can_flag){
         Procesar_Can();
      }


      //Actualizar Pantalla

      if (!printed){

         printed = true;
         Menu_Switch();
         Refresh = 0;
      }

      output_toggle(pin_a1); //Salida de Frecuencia del Main


      ////////////////////TILT Operations:
      if (tilter_c++ == 5){


         tilter_c=0;

         if (tilt) tilt = 0;
         else tilt =1;

         printed = false;

         //Operationes chequeo CAN_tx Periodicas
         if ((!Debug)&&(!PCt)) PC_Online =0;
         PCt =0;

         Rq_Online();


         output_toggle(pin_a2); //Frecuencia Main Divida para Tilt
      }

      //Chequeo Menu Grabacion
      Recorder();
```

```
      //Modo de Refrescado de Pantalla
      //Fast_GLCD consume 1kb memoria.
      #ifdef FAST_GLCD
         glcd_update();
      #else
         delay_ms(100);
      #endif
   }
}
```

C. Definiciones Generales

```
////////////////////////////////////////////////////////////////////////
////
////                        GLCD_Hardware.c
////  DESCRPICION: Driver para la pantalla GLCD, con pequeñas modi-
////  ficaciones de la librería HDM64GS12.c de PIC CSS C COMPILER
////
////  NOMBRE: JULIO CESAR RUIZ HERRERA
////   CARNE: 03038
////   FECHA: NOVIEMBRE 2007
////
////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////
////                           HDM64GS12.c                          ////
////                                                                ////
//// This file contains drivers for using a Hantronix HDM64GS12 with ////
//// a KS0108 display controller. The HDM64GS12 is 128 by 64 pixels. ////
//// The driver treats the upper left pixel as (0,0).               ////
////                                                                ////
//// Use #define FAST_GLCD if the target chip has at least 1k of RAM ////
//// to decrease the time it takes to update the display.           ////
//// glcd_update() must then be called to update the display after   ////
//// changing the pixel information.                                ////
//// See ex_glcd.c for suggested usage.                             ////
//// See KS0108.c for controlling a single 64 by 64 display         ////
////////////////////////////////////////////////////////////////////////
////                                                                ////
//// LCD Pin connections:                                           ////
//// (These can be changed as needed in the following defines).     ////
////  * 1: VSS is connected to GND                                  ////
////  * 2: VDD is connected to +5V                                  ////
////  * 3: V0  - LCD operating voltage (Constrast adjustment)       ////
////  * 4: D/I - Data or Instruction is connected to B2             ////
////  * 5: R/W - Read or Write is connected to B4                   ////
////  * 6: Enable is connected to B5                               ////
////  *7-14: Data Bus 0 to 7 is connected to port d                 ////
////  *15: Chip Select 1 is connected to B0                        ////
////  *16: Chip Select 2 is connected to B1                        ////
////  *17: Reset is connected to C0                                ////
////  *18: Negative voltage is also connected to the 20k Ohm POT    ////
////  *19: Positive voltage for LED backlight is connected to +5V   ////
////  *20: Negavtive voltage for LED backlight is connected to GND  ////
////                                                                ////
////////////////////////////////////////////////////////////////////////
////                                                                ////
////  glcd_init(mode)                                               ////
////     * Must be called before any other function.                ////
////       - mode can be ON or OFF to turn the LCD on or off        ////
////                                                                ////
```

```
////  glcd_pixel(x,y,color)                                          ////
////     * Sets the pixel to the given color.                        ////
////       - color can be ON or OFF                                  ////
////                                                                 ////
////  glcd_fillScreen(color)                                         ////
////     * Fills the entire LCD with the given color.                ////
////       - color can be ON or OFF                                  ////
////                                                                 ////
////  glcd_update()                                                  ////
////     * Write the display data stored in RAM to the LCD           ////
////     * Only available if FAST_GLCD is defined                    ////
////                                                                 ////
/////////////////////////////////////////////////////////////////////////
////        (C) Copyright 1996, 2004 Custom Computer Services        ////
//// This source code may only be used by licensed users of the CCS  ////
//// C compiler.  This source code may only be distributed to other  ////
//// licensed users of the CCS C compiler.  No other use,            ////
//// reproduction or distribution is permitted without written       ////
//// permission.  Derivative programs created using this software    ////
//// in object code form are not restricted in any way.              ////
/////////////////////////////////////////////////////////////////////////


#ifndef HDM64GS12
#define HDM64GS12

#ifndef GLCD_WIDTH
#define GLCD_WIDTH   128
#endif

#ifndef GLCD_CS1
#define GLCD_CS1     PIN_C1   // Chip Selection 1
#endif

#ifndef GLCD_CS2
#define GLCD_CS2     PIN_C0   // Chip Selection 2
#endif

#ifndef GLCD_DI
#define GLCD_DI      PIN_C2   // Data or Instruction input
#endif

#ifndef GLCD_RW
#define GLCD_RW      PIN_C3   // Read/Write
#endif

#ifndef GLCD_E
#define GLCD_E       PIN_C4   // Enable
#endif

#ifndef GLCD_RST
#define GLCD_RST     PIN_C5   // Reset
#endif

#define GLCD_LEFT    0
#define GLCD_RIGHT   1

#ifndef ON
#define ON           1
#endif

#ifndef OFF
#define OFF          0
#endif

//////////////////////////////////////////////////////////////////////////
// Function Prototypes
```

```
////////////////////////////////////////////////////////////////////////
void glcd_init(int1 mode);
void glcd_pixel(int8 x, int8 y, int1 color);
void glcd_fillScreen(int1 color);
void glcd_writeByte(int1 side, BYTE data);
BYTE glcd_readByte(int1 side);
void glcd_update();
////////////////////////////////////////////////////////////////////////

#ifdef FAST_GLCD
struct
{
   int8 left[512];
   int8 right[512];
} displayData;
#endif


// Purpose:       Initialize the LCD.
//                Call before using any other LCD function.
// Inputs:        OFF - Turns the LCD off
//                ON  - Turns the LCD on
void glcd_init(int1 mode)
{
   // Initialze some pins
   output_high(GLCD_RST);
   output_low(GLCD_E);
   output_low(GLCD_CS1);
   output_low(GLCD_CS2);

   output_low(GLCD_DI);                // Set for instruction
   glcd_writeByte(GLCD_LEFT,  0xC0);   // Specify first RAM line at the top
   glcd_writeByte(GLCD_RIGHT, 0xC0);   //   of the screen
   glcd_writeByte(GLCD_LEFT,  0x40);   // Set the column address to 0
   glcd_writeByte(GLCD_RIGHT, 0x40);
   glcd_writeByte(GLCD_LEFT,  0xB8);   // Set the page address to 0
   glcd_writeByte(GLCD_RIGHT, 0xB8);

   if(mode == ON)
   {
      glcd_writeByte(GLCD_LEFT,  0x3F); // Turn the display on
      glcd_writeByte(GLCD_RIGHT, 0x3F);
   }
   else
   {
      glcd_writeByte(GLCD_LEFT,  0x3E); // Turn the display off
      glcd_writeByte(GLCD_RIGHT, 0x3E);
   }

   glcd_fillScreen(OFF);               // Clear the display

   #ifdef FAST_GLCD
   glcd_update();
   #endif
}


// Purpose:    Update the LCD with data from the display arrays
#ifdef FAST_GLCD
void glcd_update()
{
   int8 i, j;
   int8 *p1, *p2;

   p1 = displayData.left;
   p2 = displayData.right;

   // Loop through the vertical pages
```

```
   for(i = 0; i < 8; ++i)
   {
      output_low(GLCD_DI);                      // Set for instruction
      glcd_writeByte(GLCD_LEFT, 0x40);          // Set horizontal address to 0
      glcd_writeByte(GLCD_RIGHT, 0x40);
      glcd_writeByte(GLCD_LEFT, i | 0xB8);      // Set page address
      glcd_writeByte(GLCD_RIGHT, i | 0xB8);
      output_high(GLCD_DI);                     // Set for data

      // Loop through the horizontal sections
      for(j = 0; j < 64; ++j)
      {
         glcd_writeByte(GLCD_LEFT, *p1++);      // Turn pixels on or off
         glcd_writeByte(GLCD_RIGHT, *p2++);     // Turn pixels on or off
      }
   }
}
#endif


// Purpose:    Turn a pixel on a graphic LCD on or off
// Inputs:     1) x - the x coordinate of the pixel
//             2) y - the y coordinate of the pixel
//             3) color - ON or OFF
void glcd_pixel(int8 x, int8 y, int1 color)
#ifdef FAST_GLCD
{
   int8* p;
   int16 temp;
   temp =  y/8;
   temp *= 64;
   temp += x;

   if(x > 63)
   {
      p = displayData.right + temp - 64;
   }
   else
   {
      p = displayData.left + temp;
   }

   if(color)
   {
      bit_set(*p, y%8);
   }
   else
   {
      bit_clear(*p, y%8);
   }
}
#else
{
   BYTE data;
   int1 side = GLCD_LEFT;  // Stores which chip to use on the LCD

   if(x > 63)              // Check for first or second display area
   {
      x -= 64;
      side = GLCD_RIGHT;
   }

   output_low(GLCD_DI);                         // Set for instruction
   bit_clear(x,7);                                 // Clear the MSB. Part of an
   instruction code
   bit_set(x,6);                                  // Set bit 6. Also part of an
   instruction code
   glcd_writeByte(side, x);                     // Set the horizontal address
```

```
   glcd_writeByte(side, (y/8 & 0xBF) | 0xB8);   // Set the vertical page address
   output_high(GLCD_DI);                        // Set for data
   glcd_readByte(side);                         // Need two reads to get data
   data = glcd_readByte(side);                  //  at new address

   if(color == ON)
      bit_set(data, y%8);        // Turn the pixel on
   else                          // or
      bit_clear(data, y%8);      // turn the pixel off

   output_low(GLCD_DI);          // Set for instruction
   glcd_writeByte(side, x);      // Set the horizontal address
   output_high(GLCD_DI);         // Set for data
   glcd_writeByte(side, data);   // Write the pixel data
}
#endif


// Purpose:    Fill the LCD screen with the passed in color
// Inputs:     ON  - turn all the pixels on
//             OFF - turn all the pixels off
void glcd_fillScreen(int1 color)
#ifdef FAST_GLCD
{
   int8  data;
   int8  *p1, *p2;
   int16 i;

   p1 = displayData.left;
   p2 = displayData.right;
   data = 0xFF * color;

   for(i=0; i<512; ++i)
   {
     *p1++ = data;
     *p2++ = data;
   }
}
#else
{
   int8 i, j;

   // Loop through the vertical pages
   for(i = 0; i < 8; ++i)
   {
      output_low(GLCD_DI);                      // Set for instruction
      glcd_writeByte(GLCD_LEFT, 0b01000000);    // Set horizontal address to 0
      glcd_writeByte(GLCD_RIGHT, 0b01000000);
      glcd_writeByte(GLCD_LEFT, i | 0b10111000);// Set page address
      glcd_writeByte(GLCD_RIGHT, i | 0b10111000);
      output_high(GLCD_DI);                     // Set for data

      // Loop through the horizontal sections
      for(j = 0; j < 64; ++j)
      {
         glcd_writeByte(GLCD_LEFT, 0xFF*color);  // Turn pixels on or off
         glcd_writeByte(GLCD_RIGHT, 0xFF*color); // Turn pixels on or off
      }
   }
}
#endif


// Purpose:    Write a byte of data to the specified chip
// Inputs:     1) chipSelect - which chip to write the data to
//             2) data - the byte of data to write
void glcd_writeByte(int1 side, BYTE data)
{
```

```
   if(side)                   // Choose which side to write to
      output_high(GLCD_CS2);
   else
      output_high(GLCD_CS1);

   output_low(GLCD_RW);       // Set for writing
   output_d(data);            // Put the data on the port
   delay_cycles(1);
   output_high(GLCD_E);       // Pulse the enable pin
   delay_cycles(5);
   output_low(GLCD_E);

   output_low(GLCD_CS1);      // Reset the chip select lines
   output_low(GLCD_CS2);
}


// Purpose:    Reads a byte of data from the specified chip
// Ouputs:     A byte of data read from the chip
BYTE glcd_readByte(int1 side)
{
   BYTE data;                 // Stores the data read from the LCD

   set_tris_d(0xFF);          // Set port d to input
   output_high(GLCD_RW);      // Set for reading

   if(side)                   // Choose which side to write to
      output_high(GLCD_CS2);
   else
      output_high(GLCD_CS1);

   delay_cycles(1);
   output_high(GLCD_E);       // Pulse the enable pin
   delay_cycles(4);
   data = input_d();          // Get the data from the display's output register
   output_low(GLCD_E);

   output_low(GLCD_CS1);      // Reset the chip select lines
   output_low(GLCD_CS2);
   return data;               // Return the read data
}

#endif

////////////////////////////////////////////////////////////////////////
////
////                        GLCD_Software.c
////  DESCRPICION: Funciones para dibujo en la pantalla GLCD,
////  con pequeñas modificaciones de la librería graphics.c
////  de PIC CSS C COMPILER
////
////  NOMBRE: JULIO CESAR RUIZ HERRERA
////   CARNE: 03038
////   FECHA: NOVIEMBRE 2007
////
////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////
////                          graphics.c                            ////
////                                                                ////
////   This file contains functions to draw lines, rectangles, bars, ////
////   circles and text to a display. A function which draws a      ////
////   single pixel must be defined before calling the functions in  ////
////   this file. Call it glcd_pixel(x, y, color) where x is the     ////
////   horizontal coordinate, y is the vertical coordinate, and      ////
////   color is 1 bit to turn the pixel on or off.                   ////
////                                                                 ////
////   * Note: (0, 0) is treated as the upper left corner            ////
```

```
////                                                                    ////
////////////////////////////////////////////////////////////////////////////
////                                                                    ////
////  glcd_line(x1, y1, x2, y2, color)                                  ////
////     * Draws a line from the first point to the second point        ////
////       with the given color                                         ////
////       - color can be ON or OFF                                     ////
////                                                                    ////
////  glcd_rect(x1, y1, x2, y2, fill, color)                            ////
////     * Draws a rectangle with one corner at point (x1,y1) and       ////
////       the other corner at point (x2,y2)                            ////
////       - fill can be YES or NO                                      ////
////       - color can be ON or OFF                                     ////
////                                                                    ////
////  glcd_bar(x1, y1, x2, y2, width, color)                            ////
////     * Draws a bar (wide line) from the first point to the          ////
////       second point                                                 ////
////       - width is the number of pixels wide                         ////
////       - color is ON or OFF                                         ////
////                                                                    ////
////  glcd_circle(x, y, radius, fill, color)                            ////
////     * Draws a circle with center at (x,y)                          ////
////       - fill can be YES or NO                                      ////
////       - color can be ON or OFF                                     ////
////                                                                    ////
////  glcd_text57(x, y, textptr, size, color)                           ////
////     * Write the null terminated text pointed to by textptr with    ////
////       the upper left coordinate of the first character at (x,y)    ////
////       Characters are 5 pixels wide and 7 pixels tall               ////
////       - size is an integer that scales the size of the text        ////
////       - color is ON or OFF                                         ////
////     * Note - This function wraps characters to the next line       ////
////              use #define GLCD_WIDTH to specify a display width     ////
////                                                                    ////
////////////////////////////////////////////////////////////////////////////
////        (C) Copyright 1996, 2004 Custom Computer Services           ////
//// This source code may only be used by licensed users of the CCS     ////
//// C compiler.  This source code may only be distributed to other     ////
//// licensed users of the CCS C compiler.  No other use,               ////
//// reproduction or distribution is permitted without written          ////
//// permission.  Derivative programs created using this software       ////
//// in object code form are not restricted in any way.                 ////
////////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////////
#ifndef GRAPHICS_DRAWING_FUNCTIONS
#define GRAPHICS_DRAWING_FUNCTIONS
////////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////////
#ifndef ON
#define ON  1
#endif

#ifndef OFF
#define OFF 0
#endif

#ifndef YES
#define YES 1
#endif

#ifndef NO
#define NO  0
#endif
////////////////////////////////////////////////////////////////////////////
```

```
/*

# Play
$ Stop
% Rec
& Pause

{ Antena On
} Antena Off
[ PC on
] PC off
' PC
^ iP

*/
//////////////////////////////////////////////////////////////////////
//// Defines a 5x7 font
//////////////////////////////////////////////////////////////////////
const int8 FONT[51][5] ={0x00, 0x00, 0x00, 0x00, 0x00, // SPACE
                         0x00, 0x00, 0x5F, 0x00, 0x00, // !
                         0x00, 0x03, 0x00, 0x03, 0x00, // "
                        //0x14, 0x3E, 0x14, 0x3E, 0x14, // #
                         0x00, 0x7F, 0x3E, 0x1C, 0x08, // Play
                        //0x24, 0x2A, 0x7F, 0x2A, 0x12, // $
                         0x00, 0x3C, 0x3C, 0x3C, 0x3C, // Stop
                        //0x43, 0x33, 0x08, 0x66, 0x61, // %
                         0x38, 0x7C, 0x7C, 0x7C, 0x38, // Rec
                        //0x36, 0x49, 0x55, 0x22, 0x50, // &
                         0x7E, 0x7E, 0x00, 0x7E, 0x7E, // Pause
                        //0x00, 0x05, 0x03, 0x00, 0x00, // '
                         0x7F, 0x09, 0x7F, 0x42, 0x66, // PC
                         0x00, 0x1C, 0x22, 0x41, 0x00, // (
                         0x00, 0x41, 0x22, 0x1C, 0x00, // )
                         0x14, 0x08, 0x3E, 0x08, 0x14, // *
                         0x08, 0x08, 0x3E, 0x08, 0x08, // +
                         0x00, 0x50, 0x30, 0x00, 0x00, // ,
                         0x08, 0x08, 0x08, 0x08, 0x08, // -
                         0x00, 0x60, 0x60, 0x00, 0x00, // .
                         0x20, 0x10, 0x08, 0x04, 0x02, // /
                         0x3E, 0x51, 0x49, 0x45, 0x3E, // 0
                         0x00, 0x04, 0x02, 0x7F, 0x00, // 1
                         0x42, 0x61, 0x51, 0x49, 0x46, // 2
                         0x22, 0x41, 0x49, 0x49, 0x36, // 3
                         0x18, 0x14, 0x12, 0x7F, 0x10, // 4
                         0x27, 0x45, 0x45, 0x45, 0x39, // 5
                         0x3E, 0x49, 0x49, 0x49, 0x32, // 6
                         0x01, 0x01, 0x71, 0x09, 0x07, // 7
                         0x36, 0x49, 0x49, 0x49, 0x36, // 8
                         0x26, 0x49, 0x49, 0x49, 0x3E, // 9
                         0x00, 0x36, 0x36, 0x00, 0x00, // :
                         0x00, 0x56, 0x36, 0x00, 0x00, // ;
                         0x08, 0x14, 0x22, 0x41, 0x00, // <
                         0x14, 0x14, 0x14, 0x14, 0x14, // =
                         0x00, 0x41, 0x22, 0x14, 0x08, // >
                         0x02, 0x01, 0x51, 0x09, 0x06, // ?
                         0x3E, 0x41, 0x59, 0x55, 0x5E, // @
                         0x7E, 0x09, 0x09, 0x09, 0x7E, // A
                         0x7F, 0x49, 0x49, 0x49, 0x36, // B
                         0x3E, 0x41, 0x41, 0x41, 0x22, // C
                         0x7F, 0x41, 0x41, 0x41, 0x3E, // D
                         0x7F, 0x49, 0x49, 0x49, 0x41, // E
                         0x7F, 0x09, 0x09, 0x09, 0x01, // F
                         0x3E, 0x41, 0x41, 0x49, 0x3A, // G
                         0x7F, 0x08, 0x08, 0x08, 0x7F, // H
                         0x00, 0x41, 0x7F, 0x41, 0x00, // I
                         0x30, 0x40, 0x40, 0x40, 0x3F, // J
                         0x7F, 0x08, 0x14, 0x22, 0x41, // K
                         0x7F, 0x40, 0x40, 0x40, 0x40, // L
```

```
                          0x7F, 0x02, 0x0C, 0x02, 0x7F, // M
                          0x7F, 0x02, 0x04, 0x08, 0x7F, // N
                          0x3E, 0x41, 0x41, 0x41, 0x3E, // O
                          0x7F, 0x09, 0x09, 0x09, 0x06, // P
                          0x1E, 0x21, 0x21, 0x21, 0x5E, // Q
                          0x7F, 0x09, 0x09, 0x09, 0x76};// R

const int8 FONT2[44][5]={0x26, 0x49, 0x49, 0x49, 0x32, // S
                          0x01, 0x01, 0x7F, 0x01, 0x01, // T
                          0x3F, 0x40, 0x40, 0x40, 0x3F, // U
                          0x1F, 0x20, 0x40, 0x20, 0x1F, // V
                          0x7F, 0x20, 0x10, 0x20, 0x7F, // W
                          0x41, 0x22, 0x1C, 0x22, 0x41, // X
                          0x07, 0x08, 0x70, 0x08, 0x07, // Y
                          0x61, 0x51, 0x49, 0x45, 0x43, // Z
                        //0x00, 0x7F, 0x41, 0x00, 0x00, // [
                          0xDE, 0xDF, 0xFF, 0xDF, 0x00, // PC ON
                          0x02, 0x04, 0x08, 0x10, 0x20, // \
                        //0x00, 0x00, 0x41, 0x7F, 0x00, // ]
                          0xDF, 0xD1, 0xF1, 0xD1, 0xDF, // PC OFF
                        //0x04, 0x02, 0x01, 0x02, 0x04, // ^
                          0x7A, 0x00, 0x7E, 0x0A, 0x0E, //iP
                          0x40, 0x40, 0x40, 0x40, 0x40, // _
                          0x00, 0x01, 0x02, 0x04, 0x00, // `
                          0x20, 0x54, 0x54, 0x54, 0x78, // a
                          0x7F, 0x44, 0x44, 0x44, 0x38, // b
                          0x38, 0x44, 0x44, 0x44, 0x44, // c
                          0x38, 0x44, 0x44, 0x44, 0x7F, // d
                          0x38, 0x54, 0x54, 0x54, 0x18, // e
                          0x04, 0x04, 0x7E, 0x05, 0x05, // f
                          0x08, 0x54, 0x54, 0x54, 0x3C, // g
                          0x7F, 0x08, 0x04, 0x04, 0x78, // h
                          0x00, 0x44, 0x7D, 0x40, 0x00, // i
                          0x20, 0x40, 0x44, 0x3D, 0x00, // j
                          0x7F, 0x10, 0x28, 0x44, 0x00, // k
                          0x00, 0x41, 0x7F, 0x40, 0x00, // l
                          0x7C, 0x04, 0x78, 0x04, 0x78, // m
                          0x7C, 0x08, 0x04, 0x04, 0x78, // n
                          0x38, 0x44, 0x44, 0x44, 0x38, // o
                          0x7C, 0x14, 0x14, 0x14, 0x08, // p
                          0x08, 0x14, 0x14, 0x14, 0x7C, // q
                          0x00, 0x7C, 0x08, 0x04, 0x04, // r
                          0x48, 0x54, 0x54, 0x54, 0x20, // s
                          0x04, 0x04, 0x3F, 0x44, 0x44, // t
                          0x3C, 0x40, 0x40, 0x20, 0x7C, // u
                          0x1C, 0x20, 0x40, 0x20, 0x1C, // v
                          0x3C, 0x40, 0x30, 0x40, 0x3C, // w
                          0x44, 0x28, 0x10, 0x28, 0x44, // x
                          0x0C, 0x50, 0x50, 0x50, 0x3C, // y
                          0x44, 0x64, 0x54, 0x4C, 0x44, // z
                        //0x00, 0x08, 0x36, 0x41, 0x41, // {
                          0x03, 0x07, 0xFF, 0x07, 0x03, //Atena ON
                          0x00, 0x00, 0x7F, 0x00, 0x00, // |
                        //0x41, 0x41, 0x36, 0x08, 0x00, // }
                          0x03, 0x05, 0xF9, 0x05, 0x03, //Antena OFF

                          0x02, 0x01, 0x02, 0x04, 0x02};// ~
//////////////////////////////////////////////////////////////////////////


//////////////////////////////////////////////////////////////////////////
// Purpose:       Draw a line on a graphic LCD using Bresenham's
//                line drawing algorithm
// Inputs:        (x1, y1) - the start coordinate
//                (x2, y2) - the end coordinate
//                color - ON or OFF
// Dependencies:  glcd_pixel()
//////////////////////////////////////////////////////////////////////////
```

```
#ifdef LARGE_LCD
void glcd_line(int16 x1, int16 y1, int16 x2, int16 y2, int1 color)
#else
void glcd_line(int8 x1, int8 y1, int8 x2, int8 y2, int1 color)
#endif
{
   int16        dy, dx;
   signed int8  addx=1, addy=1;
   signed int16 P, diff;

   #ifdef LARGE_LCD
   int16 i=0;
   dx = abs((signed int16)(x2 - x1));
   dy = abs((signed int16)(y2 - y1));
   #else
   int8 i=0;
   dx = abs((signed int8)(x2 - x1));
   dy = abs((signed int8)(y2 - y1));
   #endif

   if(x1 > x2)
      addx = -1;
   if(y1 > y2)
      addy = -1;

   if(dx >= dy)
   {
      dy *= 2;
      P = dy - dx;
      diff = P - dx;

      for(; i<=dx; ++i)
      {
         glcd_pixel(x1, y1, color);

         if(P < 0)
         {
            P  += dy;
            x1 += addx;
         }
         else
         {
            P  += diff;
            x1 += addx;
            y1 += addy;
         }
      }
   }
   else
   {
      dx *= 2;
      P = dx - dy;
      diff = P - dy;

      for(; i<=dy; ++i)
      {
         glcd_pixel(x1, y1, color);

         if(P < 0)
         {
            P  += dx;
            y1 += addy;
         }
         else
         {
            P  += diff;
            x1 += addx;
            y1 += addy;
```

```
         }
      }
   }
}


////////////////////////////////////////////////////////////////////////
// Purpose:       Draw a rectangle on a graphic LCD
// Inputs:        (x1, y1) - the start coordinate
//                (x2, y2) - the end coordinate
//                fill  - YES or NO
//                color - ON or OFF
// Dependencies:  glcd_pixel(), glcd_line()
////////////////////////////////////////////////////////////////////////
#ifdef LARGE_LCD
void glcd_rect(int16 x1, int16 y1, int16 x2, int16 y2, int1 fill, int1 color)
#else
void glcd_rect(int8 x1, int8 y1, int8 x2, int8 y2, int1 fill, int1 color)
#endif
{
   if(fill)
   {
      #ifdef LARGE_LCD
      int16 i, xmin, xmax, ymin, ymax;
      #else
      int8  i, xmin, xmax, ymin, ymax;
      #endif

      if(x1 < x2)                           //  Find x min and max
      {
         xmin = x1;
         xmax = x2;
      }
      else
      {
         xmin = x2;
         xmax = x1;
      }

      if(y1 < y2)                           // Find the y min and max
      {
         ymin = y1;
         ymax = y2;
      }
      else
      {
         ymin = y2;
         ymax = y1;
      }

      for(; xmin <= xmax; ++xmin)
      {
         for(i=ymin; i<=ymax; ++i)
         {
            glcd_pixel(xmin, i, color);
         }
      }
   }
   else
   {
      glcd_line(x1, y1, x2, y1, color);      // Draw the 4 sides
      glcd_line(x1, y2, x2, y2, color);
      glcd_line(x1, y1, x1, y2, color);
      glcd_line(x2, y1, x2, y2, color);
   }
}

////////////////////////////////////////////////////////////////////////
```

```
// Purpose:       Draw a bar (wide line) on a graphic LCD
// Inputs:        (x1, y1) - the start coordinate
//                (x2, y2) - the end coordinate
//                width  - The number of pixels wide
//                color - ON or OFF
////////////////////////////////////////////////////////////////////////
#ifdef LARGE_LCD
void glcd_bar(int16 x1, int16 y1, int16 x2, int16 y2, int8 width, int1 color)
#else
void glcd_bar(int8 x1, int8 y1, int8 x2, int8 y2, int8 width, int1 color)
#endif
{
   int8         half_width;
   signed int16 dy, dx;
   signed int8  addx=1, addy=1, j;
   signed int16 P, diff, c1, c2;

   #ifdef LARGE_LCD
   int16 i=0;
   dx = abs((signed int16)(x2 - x1));
   dy = abs((signed int16)(y2 - y1));
   #else
   int8 i=0;
   dx = abs((signed int8)(x2 - x1));
   dy = abs((signed int8)(y2 - y1));
   #endif

   half_width = width/2;
   c1 = -(dx*x1 + dy*y1);
   c2 = -(dx*x2 + dy*y2);

   if(x1 > x2)
   {
      signed int16 temp;
      temp = c1;
      c1 = c2;
      c2 = temp;
      addx = -1;
   }
   if(y1 > y2)
   {
      signed int16 temp;
      temp = c1;
      c1 = c2;
      c2 = temp;
      addy = -1;
   }

   if(dx >= dy)
   {
      P = 2*dy - dx;
      diff = P - dx;

      for(i=0; i<=dx; ++i)
      {
         for(j=-half_width; j<half_width+width%2; ++j)
         {
            #ifdef LARGE_LCD
            int16 temp;
            #else
            int8 temp;
            #endif

            temp = dx*x1+dy*(y1+j);    // Use more RAM to increase speed
            if(temp+c1 >= 0 && temp+c2 <=0)
               glcd_pixel(x1, y1+j, color);
         }
         if(P < 0)
```

```
         {
            P  += 2*dy;
            x1 += addx;
         }
         else
         {
            P  += diff;
            x1 += addx;
            y1 += addy;
         }
      }
   }
   else
   {
      P = 2*dx - dy;
      diff = P - dy;

      for(i=0; i<=dy; ++i)
      {
         if(P < 0)
         {
            P  += 2*dx;
            y1 += addy;
         }
         else
         {
            P  += diff;
            x1 += addx;
            y1 += addy;
         }
         for(j=-half_width; j<half_width+width%2; ++j)
         {
            #ifdef LARGE_LCD
            int16 temp;
            #else
            int8 temp;
            #endif

            temp = dx*x1+dy*(y1+j);    // Use more RAM to increase speed
            if(temp+c1 >= 0 && temp+c2 <=0)
               glcd_pixel(x1+j, y1, color);
         }
      }
   }
}


////////////////////////////////////////////////////////////////////////
// Purpose:       Draw a circle on a graphic LCD
// Inputs:        (x,y) - the center of the circle
//                radius - the radius of the circle
//                fill - YES or NO
//                color - ON or OFF
////////////////////////////////////////////////////////////////////////
#ifdef LARGE_LCD
void glcd_circle(int16 x, int16 y, int16 radius, int1 fill, int1 color)
#else
void glcd_circle(int8 x, int8 y, int8 radius, int1 fill, int1 color)
#endif
{
   #ifdef LARGE_LCD
   signed int16 a, b, P;
   #else
   signed int8  a, b, P;
   #endif

   a = 0;
   b = radius;
```

```
   P = 1 - radius;

   do
   {
      if(fill)
      {
         glcd_line(x-a, y+b, x+a, y+b, color);
         glcd_line(x-a, y-b, x+a, y-b, color);
         glcd_line(x-b, y+a, x+b, y+a, color);
         glcd_line(x-b, y-a, x+b, y-a, color);
      }
      else
      {
         glcd_pixel(a+x, b+y, color);
         glcd_pixel(b+x, a+y, color);
         glcd_pixel(x-a, b+y, color);
         glcd_pixel(x-b, a+y, color);
         glcd_pixel(b+x, y-a, color);
         glcd_pixel(a+x, y-b, color);
         glcd_pixel(x-a, y-b, color);
         glcd_pixel(x-b, y-a, color);
      }

      if(P < 0)
         P += 3 + 2 * a++;
      else
         P += 5 + 2 * (a++ - b--);
    } while(a <= b);
}


////////////////////////////////////////////////////////////////////////
// Purpose:       Write text on a graphic LCD
// Inputs:        (x,y) - The upper left coordinate of the first letter
//                textptr - A pointer to an array of text to display
//                size - The size of the text: 1 = 5x7, 2 = 10x14, ...
//                color - ON or OFF
////////////////////////////////////////////////////////////////////////
#ifdef LARGE_LCD
void glcd_text57(int16 x, int16 y, char* textptr, int8 size, int1 color)
#else
void glcd_text57(int8 x, int8 y, char* textptr, int8 size, int1 color)
#endif
{
   int8 j, k, l, m;                      // Loop counters
   int8 pixelData[5];                    // Stores character data

   for(; *textptr != '\0'; ++textptr, ++x)// Loop through the passed string
   {
      if(*textptr < 'S') // Checks if the letter is in the first font array
         memcpy(pixelData, FONT[*textptr - ' '], 5);
      else if(*textptr <= '~') // Check if the letter is in the second font
   array
         memcpy(pixelData, FONT2[*textptr - 'S'], 5);
      else
         memcpy(pixelData, FONT[0], 5);   // Default to space

      // Handles newline and carriage returns
      switch(*textptr)
      {
         case '\n':
            y += 7*size + 1;
            continue;
         case '\r':
            x = 0;
            continue;
      }
```

```
      if(x+5*size >= GLCD_WIDTH)          // Performs character wrapping
      {
         x = 0;                           // Set x at far left position
         y += 7*size + 1;                 // Set y at next position down
      }
      for(j=0; j<5; ++j, x+=size)         // Loop through character byte data
      {
         for(k=0; k < 7; ++k)             // Loop through the vertical pixels
         {
            if(bit_test(pixelData[j], k)) // Check if the pixel should be set
            {
               for(l=0; l < size; ++l)    // These two loops change the
               {                          // character's size
                  for(m=0; m < size; ++m)
                  {
                     glcd_pixel(x+m, y+k*size+l, color); // Draws the pixel
                  }
               }
            }
         }
      }
   }
}

#endif

////////////////////////////////////////////////////////////////////////
////
////                        GLCD_Software.c
////  DESCRPICION: Funciones para dibujo en la pantalla GLCD,
////  con pequeñas modificaciones de la librería graphics.c
////  de PIC CSS C COMPILER
////
////  NOMBRE: JULIO CESAR RUIZ HERRERA
////   CARNE: 03038
////   FECHA: NOVIEMBRE 2007
////
////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////
////                          graphics.c                              ////
////                                                                  ////
////   This file contains functions to draw lines, rectangles, bars, ////
////   circles and text to a display. A function which draws a       ////
////   single pixel must be defined before calling the functions in  ////
////   this file. Call it glcd_pixel(x, y, color) where x is the     ////
////   horizontal coordinate, y is the vertical coordinate, and      ////
////   color is 1 bit to turn the pixel on or off.                   ////
////                                                                 ////
////   * Note: (0, 0) is treated as the upper left corner            ////
////                                                                 ////
////////////////////////////////////////////////////////////////////////
////                                                                 ////
////  glcd_line(x1, y1, x2, y2, color)                               ////
////     * Draws a line from the first point to the second point     ////
////       with the given color                                      ////
////       - color can be ON or OFF                                  ////
////                                                                 ////
////  glcd_rect(x1, y1, x2, y2, fill, color)                         ////
////     * Draws a rectangle with one corner at point (x1,y1) and    ////
////       the other corner at point (x2,y2)                         ////
////       - fill can be YES or NO                                   ////
////       - color can be ON or OFF                                  ////
////                                                                 ////
////  glcd_bar(x1, y1, x2, y2, width, color)                         ////
////     * Draws a bar (wide line) from the first point to the       ////
////       second point                                              ////
////       - width is the number of pixels wide                      ////
```

```
////       - color is ON or OFF                                        ////
////                                                                   ////
////  glcd_circle(x, y, radius, fill, color)                           ////
////     * Draws a circle with center at (x,y)                         ////
////       - fill can be YES or NO                                     ////
////       - color can be ON or OFF                                    ////
////                                                                   ////
////  glcd_text57(x, y, textptr, size, color)                          ////
////     * Write the null terminated text pointed to by textptr with ////
////       the upper left coordinate of the first character at (x,y) ////
////       Characters are 5 pixels wide and 7 pixels tall              ////
////       - size is an integer that scales the size of the text       ////
////       - color is ON or OFF                                        ////
////     * Note - This function wraps characters to the next line      ////
////              use #define GLCD_WIDTH to specify a display width   ////
////                                                                   ////
/////////////////////////////////////////////////////////////////////////
////        (C) Copyright 1996, 2004 Custom Computer Services          ////
//// This source code may only be used by licensed users of the CCS   ////
//// C compiler.  This source code may only be distributed to other   ////
//// licensed users of the CCS C compiler.  No other use,             ////
//// reproduction or distribution is permitted without written        ////
//// permission.  Derivative programs created using this software     ////
//// in object code form are not restricted in any way.               ////
/////////////////////////////////////////////////////////////////////////


/////////////////////////////////////////////////////////////////////////
#ifndef GRAPHICS_DRAWING_FUNCTIONS
#define GRAPHICS_DRAWING_FUNCTIONS
/////////////////////////////////////////////////////////////////////////


/////////////////////////////////////////////////////////////////////////
#ifndef ON
#define ON  1
#endif

#ifndef OFF
#define OFF 0
#endif

#ifndef YES
#define YES 1
#endif

#ifndef NO
#define NO  0
#endif
/////////////////////////////////////////////////////////////////////////
/*

# Play
$ Stop
% Rec
& Pause

{ Antena On
} Antena Off
[ PC on
] PC off
' PC
^ iP

*/
/////////////////////////////////////////////////////////////////////////
//// Defines a 5x7 font
/////////////////////////////////////////////////////////////////////////
```

```
const int8 FONT[51][5] ={0x00, 0x00, 0x00, 0x00, 0x00, // SPACE
                           0x00, 0x00, 0x5F, 0x00, 0x00, // !
                           0x00, 0x03, 0x00, 0x03, 0x00, // "
                         //0x14, 0x3E, 0x14, 0x3E, 0x14, // #
                           0x00, 0x7F, 0x3E, 0x1C, 0x08, // Play
                         //0x24, 0x2A, 0x7F, 0x2A, 0x12, // $
                           0x00, 0x3C, 0x3C, 0x3C, 0x3C, // Stop
                         //0x43, 0x33, 0x08, 0x66, 0x61, // %
                           0x38, 0x7C, 0x7C, 0x7C, 0x38, // Rec
                         //0x36, 0x49, 0x55, 0x22, 0x50, // &
                           0x7E, 0x7E, 0x00, 0x7E, 0x7E, // Pause
                         //0x00, 0x05, 0x03, 0x00, 0x00, // '
                           0x7F, 0x09, 0x7F, 0x42, 0x66, // PC
                           0x00, 0x1C, 0x22, 0x41, 0x00, // (
                           0x00, 0x41, 0x22, 0x1C, 0x00, // )
                           0x14, 0x08, 0x3E, 0x08, 0x14, // *
                           0x08, 0x08, 0x3E, 0x08, 0x08, // +
                           0x00, 0x50, 0x30, 0x00, 0x00, // ,
                           0x08, 0x08, 0x08, 0x08, 0x08, // -
                           0x00, 0x60, 0x60, 0x00, 0x00, // .
                           0x20, 0x10, 0x08, 0x04, 0x02, // /
                           0x3E, 0x51, 0x49, 0x45, 0x3E, // 0
                           0x00, 0x04, 0x02, 0x7F, 0x00, // 1
                           0x42, 0x61, 0x51, 0x49, 0x46, // 2
                           0x22, 0x41, 0x49, 0x49, 0x36, // 3
                           0x18, 0x14, 0x12, 0x7F, 0x10, // 4
                           0x27, 0x45, 0x45, 0x45, 0x39, // 5
                           0x3E, 0x49, 0x49, 0x49, 0x32, // 6
                           0x01, 0x01, 0x71, 0x09, 0x07, // 7
                           0x36, 0x49, 0x49, 0x49, 0x36, // 8
                           0x26, 0x49, 0x49, 0x49, 0x3E, // 9
                           0x00, 0x36, 0x36, 0x00, 0x00, // :
                           0x00, 0x56, 0x36, 0x00, 0x00, // ;
                           0x08, 0x14, 0x22, 0x41, 0x00, // <
                           0x14, 0x14, 0x14, 0x14, 0x14, // =
                           0x00, 0x41, 0x22, 0x14, 0x08, // >
                           0x02, 0x01, 0x51, 0x09, 0x06, // ?
                           0x3E, 0x41, 0x59, 0x55, 0x5E, // @
                           0x7E, 0x09, 0x09, 0x09, 0x7E, // A
                           0x7F, 0x49, 0x49, 0x49, 0x36, // B
                           0x3E, 0x41, 0x41, 0x41, 0x22, // C
                           0x7F, 0x41, 0x41, 0x41, 0x3E, // D
                           0x7F, 0x49, 0x49, 0x49, 0x41, // E
                           0x7F, 0x09, 0x09, 0x09, 0x01, // F
                           0x3E, 0x41, 0x41, 0x49, 0x3A, // G
                           0x7F, 0x08, 0x08, 0x08, 0x7F, // H
                           0x00, 0x41, 0x7F, 0x41, 0x00, // I
                           0x30, 0x40, 0x40, 0x40, 0x3F, // J
                           0x7F, 0x08, 0x14, 0x22, 0x41, // K
                           0x7F, 0x40, 0x40, 0x40, 0x40, // L
                           0x7F, 0x02, 0x0C, 0x02, 0x7F, // M
                           0x7F, 0x02, 0x04, 0x08, 0x7F, // N
                           0x3E, 0x41, 0x41, 0x41, 0x3E, // O
                           0x7F, 0x09, 0x09, 0x09, 0x06, // P
                           0x1E, 0x21, 0x21, 0x21, 0x5E, // Q
                           0x7F, 0x09, 0x09, 0x09, 0x76};// R

const int8 FONT2[44][5]={0x26, 0x49, 0x49, 0x49, 0x32, // S
                           0x01, 0x01, 0x7F, 0x01, 0x01, // T
                           0x3F, 0x40, 0x40, 0x40, 0x3F, // U
                           0x1F, 0x20, 0x40, 0x20, 0x1F, // V
                           0x7F, 0x20, 0x10, 0x20, 0x7F, // W
                           0x41, 0x22, 0x1C, 0x22, 0x41, // X
                           0x07, 0x08, 0x70, 0x08, 0x07, // Y
                           0x61, 0x51, 0x49, 0x45, 0x43, // Z
                         //0x00, 0x7F, 0x41, 0x00, 0x00, // [
                           0xDE, 0xDF, 0xFF, 0xDF, 0x00, // PC ON
                           0x02, 0x04, 0x08, 0x10, 0x20, // \
```

```
                        //0x00, 0x00, 0x41, 0x7F, 0x00, // ]
                          0xDF, 0xD1, 0xF1, 0xD1, 0xDF, // PC OFF
                        //0x04, 0x02, 0x01, 0x02, 0x04, // ^
                          0x7A, 0x00, 0x7E, 0x0A, 0x0E, //iP
                          0x40, 0x40, 0x40, 0x40, 0x40, // _
                          0x00, 0x01, 0x02, 0x04, 0x00, // `
                          0x20, 0x54, 0x54, 0x54, 0x78, // a
                          0x7F, 0x44, 0x44, 0x44, 0x38, // b
                          0x38, 0x44, 0x44, 0x44, 0x44, // c
                          0x38, 0x44, 0x44, 0x44, 0x7F, // d
                          0x38, 0x54, 0x54, 0x54, 0x18, // e
                          0x04, 0x04, 0x7E, 0x05, 0x05, // f
                          0x08, 0x54, 0x54, 0x54, 0x3C, // g
                          0x7F, 0x08, 0x04, 0x04, 0x78, // h
                          0x00, 0x44, 0x7D, 0x40, 0x00, // i
                          0x20, 0x40, 0x44, 0x3D, 0x00, // j
                          0x7F, 0x10, 0x28, 0x44, 0x00, // k
                          0x00, 0x41, 0x7F, 0x40, 0x00, // l
                          0x7C, 0x04, 0x78, 0x04, 0x78, // m
                          0x7C, 0x08, 0x04, 0x04, 0x78, // n
                          0x38, 0x44, 0x44, 0x44, 0x38, // o
                          0x7C, 0x14, 0x14, 0x14, 0x08, // p
                          0x08, 0x14, 0x14, 0x14, 0x7C, // q
                          0x00, 0x7C, 0x08, 0x04, 0x04, // r
                          0x48, 0x54, 0x54, 0x54, 0x20, // s
                          0x04, 0x04, 0x3F, 0x44, 0x44, // t
                          0x3C, 0x40, 0x40, 0x20, 0x7C, // u
                          0x1C, 0x20, 0x40, 0x20, 0x1C, // v
                          0x3C, 0x40, 0x30, 0x40, 0x3C, // w
                          0x44, 0x28, 0x10, 0x28, 0x44, // x
                          0x0C, 0x50, 0x50, 0x50, 0x3C, // y
                          0x44, 0x64, 0x54, 0x4C, 0x44, // z
                        //0x00, 0x08, 0x36, 0x41, 0x41, // {
                          0x03, 0x07, 0xFF, 0x07, 0x03, //Atena ON
                          0x00, 0x00, 0x7F, 0x00, 0x00, // |
                        //0x41, 0x41, 0x36, 0x08, 0x00, // }
                          0x03, 0x05, 0xF9, 0x05, 0x03, //Antena OFF

                          0x02, 0x01, 0x02, 0x04, 0x02};// ~
////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////
// Purpose:       Draw a line on a graphic LCD using Bresenham's
//                line drawing algorithm
// Inputs:        (x1, y1) - the start coordinate
//                (x2, y2) - the end coordinate
//                color - ON or OFF
// Dependencies:  glcd_pixel()
////////////////////////////////////////////////////////////////////////
#ifdef LARGE_LCD
void glcd_line(int16 x1, int16 y1, int16 x2, int16 y2, int1 color)
#else
void glcd_line(int8 x1, int8 y1, int8 x2, int8 y2, int1 color)
#endif
{
   int16        dy, dx;
   signed int8  addx=1, addy=1;
   signed int16 P, diff;

   #ifdef LARGE_LCD
   int16 i=0;
   dx = abs((signed int16)(x2 - x1));
   dy = abs((signed int16)(y2 - y1));
   #else
   int8 i=0;
   dx = abs((signed int8)(x2 - x1));
   dy = abs((signed int8)(y2 - y1));
```

```
   #endif

   if(x1 > x2)
      addx = -1;
   if(y1 > y2)
      addy = -1;

   if(dx >= dy)
   {
      dy *= 2;
      P = dy - dx;
      diff = P - dx;

      for(; i<=dx; ++i)
      {
         glcd_pixel(x1, y1, color);

         if(P < 0)
         {
            P  += dy;
            x1 += addx;
         }
         else
         {
            P  += diff;
            x1 += addx;
            y1 += addy;
         }
      }
   }
   else
   {
      dx *= 2;
      P = dx - dy;
      diff = P - dy;

      for(; i<=dy; ++i)
      {
         glcd_pixel(x1, y1, color);

         if(P < 0)
         {
            P  += dx;
            y1 += addy;
         }
         else
         {
            P  += diff;
            x1 += addx;
            y1 += addy;
         }
      }
   }
}


////////////////////////////////////////////////////////////////////////
// Purpose:       Draw a rectangle on a graphic LCD
// Inputs:        (x1, y1) - the start coordinate
//                (x2, y2) - the end coordinate
//                fill  - YES or NO
//                color - ON or OFF
// Dependencies:  glcd_pixel(), glcd_line()
////////////////////////////////////////////////////////////////////////
#ifdef LARGE_LCD
void glcd_rect(int16 x1, int16 y1, int16 x2, int16 y2, int1 fill, int1 color)
#else
void glcd_rect(int8 x1, int8 y1, int8 x2, int8 y2, int1 fill, int1 color)
```

```
#endif
{
   if(fill)
   {
      #ifdef LARGE_LCD
      int16 i, xmin, xmax, ymin, ymax;
      #else
      int8  i, xmin, xmax, ymin, ymax;
      #endif

      if(x1 < x2)                          //  Find x min and max
      {
         xmin = x1;
         xmax = x2;
      }
      else
      {
         xmin = x2;
         xmax = x1;
      }

      if(y1 < y2)                          // Find the y min and max
      {
         ymin = y1;
         ymax = y2;
      }
      else
      {
         ymin = y2;
         ymax = y1;
      }

      for(; xmin <= xmax; ++xmin)
      {
         for(i=ymin; i<=ymax; ++i)
         {
            glcd_pixel(xmin, i, color);
         }
      }
   }
   else
   {
      glcd_line(x1, y1, x2, y1, color);      // Draw the 4 sides
      glcd_line(x1, y2, x2, y2, color);
      glcd_line(x1, y1, x1, y2, color);
      glcd_line(x2, y1, x2, y2, color);
   }
}

////////////////////////////////////////////////////////////////////////
// Purpose:       Draw a bar (wide line) on a graphic LCD
// Inputs:        (x1, y1) - the start coordinate
//                (x2, y2) - the end coordinate
//                width  - The number of pixels wide
//                color - ON or OFF
////////////////////////////////////////////////////////////////////////
#ifdef LARGE_LCD
void glcd_bar(int16 x1, int16 y1, int16 x2, int16 y2, int8 width, int1 color)
#else
void glcd_bar(int8 x1, int8 y1, int8 x2, int8 y2, int8 width, int1 color)
#endif
{
   int8         half_width;
   signed int16 dy, dx;
   signed int8  addx=1, addy=1, j;
   signed int16 P, diff, c1, c2;

   #ifdef LARGE_LCD
```

```
int16 i=0;
dx = abs((signed int16)(x2 - x1));
dy = abs((signed int16)(y2 - y1));
#else
int8 i=0;
dx = abs((signed int8)(x2 - x1));
dy = abs((signed int8)(y2 - y1));
#endif

half_width = width/2;
c1 = -(dx*x1 + dy*y1);
c2 = -(dx*x2 + dy*y2);

if(x1 > x2)
{
   signed int16 temp;
   temp = c1;
   c1 = c2;
   c2 = temp;
   addx = -1;
}
if(y1 > y2)
{
   signed int16 temp;
   temp = c1;
   c1 = c2;
   c2 = temp;
   addy = -1;
}

if(dx >= dy)
{
   P = 2*dy - dx;
   diff = P - dx;

   for(i=0; i<=dx; ++i)
   {
      for(j=-half_width; j<half_width+width%2; ++j)
      {
         #ifdef LARGE_LCD
         int16 temp;
         #else
         int8 temp;
         #endif

         temp = dx*x1+dy*(y1+j);    // Use more RAM to increase speed
         if(temp+c1 >= 0 && temp+c2 <=0)
            glcd_pixel(x1, y1+j, color);
      }
      if(P < 0)
      {
         P  += 2*dy;
         x1 += addx;
      }
      else
      {
         P  += diff;
         x1 += addx;
         y1 += addy;
      }
   }
}
else
{
   P = 2*dx - dy;
   diff = P - dy;

   for(i=0; i<=dy; ++i)
```

```
      {
         if(P < 0)
         {
            P  += 2*dx;
            y1 += addy;
         }
         else
         {
            P  += diff;
            x1 += addx;
            y1 += addy;
         }
         for(j=-half_width; j<half_width+width%2; ++j)
         {
            #ifdef LARGE_LCD
            int16 temp;
            #else
            int8 temp;
            #endif

            temp = dx*x1+dy*(y1+j);    // Use more RAM to increase speed
            if(temp+c1 >= 0 && temp+c2 <=0)
               glcd_pixel(x1+j, y1, color);
         }
      }
   }
}


////////////////////////////////////////////////////////////////////////
// Purpose:       Draw a circle on a graphic LCD
// Inputs:        (x,y) - the center of the circle
//                radius - the radius of the circle
//                fill - YES or NO
//                color - ON or OFF
////////////////////////////////////////////////////////////////////////
#ifdef LARGE_LCD
void glcd_circle(int16 x, int16 y, int16 radius, int1 fill, int1 color)
#else
void glcd_circle(int8 x, int8 y, int8 radius, int1 fill, int1 color)
#endif
{
   #ifdef LARGE_LCD
   signed int16 a, b, P;
   #else
   signed int8  a, b, P;
   #endif

   a = 0;
   b = radius;
   P = 1 - radius;

   do
   {
      if(fill)
      {
         glcd_line(x-a, y+b, x+a, y+b, color);
         glcd_line(x-a, y-b, x+a, y-b, color);
         glcd_line(x-b, y+a, x+b, y+a, color);
         glcd_line(x-b, y-a, x+b, y-a, color);
      }
      else
      {
         glcd_pixel(a+x, b+y, color);
         glcd_pixel(b+x, a+y, color);
         glcd_pixel(x-a, b+y, color);
         glcd_pixel(x-b, a+y, color);
         glcd_pixel(b+x, y-a, color);
```

```
         glcd_pixel(a+x, y-b, color);
         glcd_pixel(x-a, y-b, color);
         glcd_pixel(x-b, y-a, color);
      }

      if(P < 0)
         P += 3 + 2 * a++;
      else
         P += 5 + 2 * (a++ - b--);
    } while(a <= b);
}


////////////////////////////////////////////////////////////////////////
// Purpose:       Write text on a graphic LCD
// Inputs:        (x,y) - The upper left coordinate of the first letter
//                textptr - A pointer to an array of text to display
//                size - The size of the text: 1 = 5x7, 2 = 10x14, ...
//                color - ON or OFF
////////////////////////////////////////////////////////////////////////
#ifdef LARGE_LCD
void glcd_text57(int16 x, int16 y, char* textptr, int8 size, int1 color)
#else
void glcd_text57(int8 x, int8 y, char* textptr, int8 size, int1 color)
#endif
{
   int8 j, k, l, m;                     // Loop counters
   int8 pixelData[5];                   // Stores character data

   for(; *textptr != '\0'; ++textptr, ++x)// Loop through the passed string
   {
      if(*textptr < 'S') // Checks if the letter is in the first font array
         memcpy(pixelData, FONT[*textptr - ' '], 5);
      else if(*textptr <= '~') // Check if the letter is in the second font
   array
         memcpy(pixelData, FONT2[*textptr - 'S'], 5);
      else
         memcpy(pixelData, FONT[0], 5);   // Default to space

      // Handles newline and carriage returns
      switch(*textptr)
      {
         case '\n':
            y += 7*size + 1;
            continue;
         case '\r':
            x = 0;
            continue;
      }

      if(x+5*size >= GLCD_WIDTH)          // Performs character wrapping
      {
         x = 0;                           // Set x at far left position
         y += 7*size + 1;                 // Set y at next position down
      }
      for(j=0; j<5; ++j, x+=size)         // Loop through character byte data
      {
         for(k=0; k < 7; ++k)             // Loop through the vertical pixels
         {
            if(bit_test(pixelData[j], k)) // Check if the pixel should be set
            {
               for(l=0; l < size; ++l)    // These two loops change the
               {                          // character's size
                  for(m=0; m < size; ++m)
                  {
                     glcd_pixel(x+m, y+k*size+l, color); // Draws the pixel
                  }
               }
```

```
            }
         }
      }
   }
}

#endif

//////////////////////////////////////////////////////////////////////
////
////                        GLCD_Software.c
////  DESCRPICION: Funciones para dibujo en la pantalla GLCD,
////  con pequeñas modificaciones de la librería graphics.c
////  de PIC CSS C COMPILER
////
////  NOMBRE: JULIO CESAR RUIZ HERRERA
////   CARNE: 03038
////   FECHA: NOVIEMBRE 2007
////
//////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////
////                          graphics.c                          ////
////                                                              ////
////   This file contains functions to draw lines, rectangles, bars, ////
////   circles and text to a display. A function which draws a       ////
////   single pixel must be defined before calling the functions in  ////
////   this file. Call it glcd_pixel(x, y, color) where x is the     ////
////   horizontal coordinate, y is the vertical coordinate, and      ////
////   color is 1 bit to turn the pixel on or off.                   ////
////                                                                 ////
////   * Note: (0, 0) is treated as the upper left corner            ////
////                                                                 ////
//////////////////////////////////////////////////////////////////////
////                                                                 ////
////  glcd_line(x1, y1, x2, y2, color)                               ////
////     * Draws a line from the first point to the second point     ////
////       with the given color                                      ////
////       - color can be ON or OFF                                  ////
////                                                                 ////
////  glcd_rect(x1, y1, x2, y2, fill, color)                         ////
////     * Draws a rectangle with one corner at point (x1,y1) and    ////
////       the other corner at point (x2,y2)                         ////
////       - fill can be YES or NO                                   ////
////       - color can be ON or OFF                                  ////
////                                                                 ////
////  glcd_bar(x1, y1, x2, y2, width, color)                         ////
////     * Draws a bar (wide line) from the first point to the       ////
////       second point                                              ////
////       - width is the number of pixels wide                      ////
////       - color is ON or OFF                                      ////
////                                                                 ////
////  glcd_circle(x, y, radius, fill, color)                         ////
////     * Draws a circle with center at (x,y)                       ////
////       - fill can be YES or NO                                   ////
////       - color can be ON or OFF                                  ////
////                                                                 ////
////  glcd_text57(x, y, textptr, size, color)                        ////
////     * Write the null terminated text pointed to by textptr with ////
////       the upper left coordinate of the first character at (x,y) ////
////       Characters are 5 pixels wide and 7 pixels tall            ////
////       - size is an integer that scales the size of the text     ////
////       - color is ON or OFF                                      ////
////     * Note - This function wraps characters to the next line    ////
////              use #define GLCD_WIDTH to specify a display width  ////
////                                                                 ////
//////////////////////////////////////////////////////////////////////
////        (C) Copyright 1996, 2004 Custom Computer Services        ////
```

```
//// This source code may only be used by licensed users of the CCS  ////
//// C compiler.  This source code may only be distributed to other  ////
//// licensed users of the CCS C compiler.  No other use,            ////
//// reproduction or distribution is permitted without written       ////
//// permission.  Derivative programs created using this software    ////
//// in object code form are not restricted in any way.              ////
//////////////////////////////////////////////////////////////////////////


//////////////////////////////////////////////////////////////////////////
#ifndef GRAPHICS_DRAWING_FUNCTIONS
#define GRAPHICS_DRAWING_FUNCTIONS
//////////////////////////////////////////////////////////////////////////


//////////////////////////////////////////////////////////////////////////
#ifndef ON
#define ON  1
#endif

#ifndef OFF
#define OFF 0
#endif

#ifndef YES
#define YES 1
#endif

#ifndef NO
#define NO  0
#endif
//////////////////////////////////////////////////////////////////////////
/*

# Play
$ Stop
% Rec
& Pause

{ Antena On
} Antena Off
[ PC on
] PC off
' PC
^ iP

*/
//////////////////////////////////////////////////////////////////////////
//// Defines a 5x7 font
//////////////////////////////////////////////////////////////////////////
const int8 FONT[51][5] ={0x00, 0x00, 0x00, 0x00, 0x00, // SPACE
                         0x00, 0x00, 0x5F, 0x00, 0x00, // !
                         0x00, 0x03, 0x00, 0x03, 0x00, // "
                        //0x14, 0x3E, 0x14, 0x3E, 0x14, // #
                         0x00, 0x7F, 0x3E, 0x1C, 0x08, // Play
                        //0x24, 0x2A, 0x7F, 0x2A, 0x12, // $
                         0x00, 0x3C, 0x3C, 0x3C, 0x3C, // Stop
                        //0x43, 0x33, 0x08, 0x66, 0x61, // %
                         0x38, 0x7C, 0x7C, 0x7C, 0x38, // Rec
                        //0x36, 0x49, 0x55, 0x22, 0x50, // &
                         0x7E, 0x7E, 0x00, 0x7E, 0x7E, // Pause
                        //0x00, 0x05, 0x03, 0x00, 0x00, // '
                         0x7F, 0x09, 0x7F, 0x42, 0x66, // PC
                         0x00, 0x1C, 0x22, 0x41, 0x00, // (
                         0x00, 0x41, 0x22, 0x1C, 0x00, // )
                         0x14, 0x08, 0x3E, 0x08, 0x14, // *
                         0x08, 0x08, 0x3E, 0x08, 0x08, // +
                         0x00, 0x50, 0x30, 0x00, 0x00, // ,
```

```
                            0x08, 0x08, 0x08, 0x08, 0x08, // -
                            0x00, 0x60, 0x60, 0x00, 0x00, // .
                            0x20, 0x10, 0x08, 0x04, 0x02, // /
                            0x3E, 0x51, 0x49, 0x45, 0x3E, // 0
                            0x00, 0x04, 0x02, 0x7F, 0x00, // 1
                            0x42, 0x61, 0x51, 0x49, 0x46, // 2
                            0x22, 0x41, 0x49, 0x49, 0x36, // 3
                            0x18, 0x14, 0x12, 0x7F, 0x10, // 4
                            0x27, 0x45, 0x45, 0x45, 0x39, // 5
                            0x3E, 0x49, 0x49, 0x49, 0x32, // 6
                            0x01, 0x01, 0x71, 0x09, 0x07, // 7
                            0x36, 0x49, 0x49, 0x49, 0x36, // 8
                            0x26, 0x49, 0x49, 0x49, 0x3E, // 9
                            0x00, 0x36, 0x36, 0x00, 0x00, // :
                            0x00, 0x56, 0x36, 0x00, 0x00, // ;
                            0x08, 0x14, 0x22, 0x41, 0x00, // <
                            0x14, 0x14, 0x14, 0x14, 0x14, // =
                            0x00, 0x41, 0x22, 0x14, 0x08, // >
                            0x02, 0x01, 0x51, 0x09, 0x06, // ?
                            0x3E, 0x41, 0x59, 0x55, 0x5E, // @
                            0x7E, 0x09, 0x09, 0x09, 0x7E, // A
                            0x7F, 0x49, 0x49, 0x49, 0x36, // B
                            0x3E, 0x41, 0x41, 0x41, 0x22, // C
                            0x7F, 0x41, 0x41, 0x41, 0x3E, // D
                            0x7F, 0x49, 0x49, 0x49, 0x41, // E
                            0x7F, 0x09, 0x09, 0x09, 0x01, // F
                            0x3E, 0x41, 0x41, 0x49, 0x3A, // G
                            0x7F, 0x08, 0x08, 0x08, 0x7F, // H
                            0x00, 0x41, 0x7F, 0x41, 0x00, // I
                            0x30, 0x40, 0x40, 0x40, 0x3F, // J
                            0x7F, 0x08, 0x14, 0x22, 0x41, // K
                            0x7F, 0x40, 0x40, 0x40, 0x40, // L
                            0x7F, 0x02, 0x0C, 0x02, 0x7F, // M
                            0x7F, 0x02, 0x04, 0x08, 0x7F, // N
                            0x3E, 0x41, 0x41, 0x41, 0x3E, // O
                            0x7F, 0x09, 0x09, 0x09, 0x06, // P
                            0x1E, 0x21, 0x21, 0x21, 0x5E, // Q
                            0x7F, 0x09, 0x09, 0x09, 0x76};// R

const int8 FONT2[44][5]={0x26, 0x49, 0x49, 0x49, 0x32, // S
                            0x01, 0x01, 0x7F, 0x01, 0x01, // T
                            0x3F, 0x40, 0x40, 0x40, 0x3F, // U
                            0x1F, 0x20, 0x40, 0x20, 0x1F, // V
                            0x7F, 0x20, 0x10, 0x20, 0x7F, // W
                            0x41, 0x22, 0x1C, 0x22, 0x41, // X
                            0x07, 0x08, 0x70, 0x08, 0x07, // Y
                            0x61, 0x51, 0x49, 0x45, 0x43, // Z
                           //0x00, 0x7F, 0x41, 0x00, 0x00, // [
                            0xDE, 0xDF, 0xFF, 0xDF, 0x00, // PC ON
                            0x02, 0x04, 0x08, 0x10, 0x20, // \
                           //0x00, 0x00, 0x41, 0x7F, 0x00, // ]
                            0xDF, 0xD1, 0xF1, 0xD1, 0xDF, // PC OFF
                           //0x04, 0x02, 0x01, 0x02, 0x04, // ^
                            0x7A, 0x00, 0x7E, 0x0A, 0x0E, //iP
                            0x40, 0x40, 0x40, 0x40, 0x40, // _
                            0x00, 0x01, 0x02, 0x04, 0x00, // `
                            0x20, 0x54, 0x54, 0x54, 0x78, // a
                            0x7F, 0x44, 0x44, 0x44, 0x38, // b
                            0x38, 0x44, 0x44, 0x44, 0x44, // c
                            0x38, 0x44, 0x44, 0x44, 0x7F, // d
                            0x38, 0x54, 0x54, 0x54, 0x18, // e
                            0x04, 0x04, 0x7E, 0x05, 0x05, // f
                            0x08, 0x54, 0x54, 0x54, 0x3C, // g
                            0x7F, 0x08, 0x04, 0x04, 0x78, // h
                            0x00, 0x44, 0x7D, 0x40, 0x00, // i
                            0x20, 0x40, 0x44, 0x3D, 0x00, // j
                            0x7F, 0x10, 0x28, 0x44, 0x00, // k
                            0x00, 0x41, 0x7F, 0x40, 0x00, // l
```

```
                        0x7C, 0x04, 0x78, 0x04, 0x78, // m
                        0x7C, 0x08, 0x04, 0x04, 0x78, // n
                        0x38, 0x44, 0x44, 0x44, 0x38, // o
                        0x7C, 0x14, 0x14, 0x14, 0x08, // p
                        0x08, 0x14, 0x14, 0x14, 0x7C, // q
                        0x00, 0x7C, 0x08, 0x04, 0x04, // r
                        0x48, 0x54, 0x54, 0x54, 0x20, // s
                        0x04, 0x04, 0x3F, 0x44, 0x44, // t
                        0x3C, 0x40, 0x40, 0x20, 0x7C, // u
                        0x1C, 0x20, 0x40, 0x20, 0x1C, // v
                        0x3C, 0x40, 0x30, 0x40, 0x3C, // w
                        0x44, 0x28, 0x10, 0x28, 0x44, // x
                        0x0C, 0x50, 0x50, 0x50, 0x3C, // y
                        0x44, 0x64, 0x54, 0x4C, 0x44, // z
                      //0x00, 0x08, 0x36, 0x41, 0x41, // {
                        0x03, 0x07, 0xFF, 0x07, 0x03, //Atena ON
                        0x00, 0x00, 0x7F, 0x00, 0x00, // |
                      //0x41, 0x41, 0x36, 0x08, 0x00, // }
                        0x03, 0x05, 0xF9, 0x05, 0x03, //Antena OFF

                        0x02, 0x01, 0x02, 0x04, 0x02};// ~
//////////////////////////////////////////////////////////////////////////


//////////////////////////////////////////////////////////////////////////
// Purpose:       Draw a line on a graphic LCD using Bresenham's
//                line drawing algorithm
// Inputs:        (x1, y1) - the start coordinate
//                (x2, y2) - the end coordinate
//                color - ON or OFF
// Dependencies:  glcd_pixel()
//////////////////////////////////////////////////////////////////////////
#ifdef LARGE_LCD
void glcd_line(int16 x1, int16 y1, int16 x2, int16 y2, int1 color)
#else
void glcd_line(int8 x1, int8 y1, int8 x2, int8 y2, int1 color)
#endif
{
   int16        dy, dx;
   signed int8  addx=1, addy=1;
   signed int16 P, diff;

   #ifdef LARGE_LCD
   int16 i=0;
   dx = abs((signed int16)(x2 - x1));
   dy = abs((signed int16)(y2 - y1));
   #else
   int8 i=0;
   dx = abs((signed int8)(x2 - x1));
   dy = abs((signed int8)(y2 - y1));
   #endif

   if(x1 > x2)
      addx = -1;
   if(y1 > y2)
      addy = -1;

   if(dx >= dy)
   {
      dy *= 2;
      P = dy - dx;
      diff = P - dx;

      for(; i<=dx; ++i)
      {
         glcd_pixel(x1, y1, color);

         if(P < 0)
```

```
         {
            P  += dy;
            x1 += addx;
         }
         else
         {
            P  += diff;
            x1 += addx;
            y1 += addy;
         }
      }
   }
   else
   {
      dx *= 2;
      P = dx - dy;
      diff = P - dy;

      for(; i<=dy; ++i)
      {
         glcd_pixel(x1, y1, color);

         if(P < 0)
         {
            P  += dx;
            y1 += addy;
         }
         else
         {
            P  += diff;
            x1 += addx;
            y1 += addy;
         }
      }
   }
}


////////////////////////////////////////////////////////////////////////
// Purpose:       Draw a rectangle on a graphic LCD
// Inputs:        (x1, y1) - the start coordinate
//                (x2, y2) - the end coordinate
//                fill  - YES or NO
//                color - ON or OFF
// Dependencies:  glcd_pixel(), glcd_line()
////////////////////////////////////////////////////////////////////////
#ifdef LARGE_LCD
void glcd_rect(int16 x1, int16 y1, int16 x2, int16 y2, int1 fill, int1 color)
#else
void glcd_rect(int8 x1, int8 y1, int8 x2, int8 y2, int1 fill, int1 color)
#endif
{
   if(fill)
   {
      #ifdef LARGE_LCD
      int16 i, xmin, xmax, ymin, ymax;
      #else
      int8  i, xmin, xmax, ymin, ymax;
      #endif

      if(x1 < x2)                          //  Find x min and max
      {
         xmin = x1;
         xmax = x2;
      }
      else
      {
         xmin = x2;
```

```
         xmax = x1;
      }

      if(y1 < y2)                              // Find the y min and max
      {
         ymin = y1;
         ymax = y2;
      }
      else
      {
         ymin = y2;
         ymax = y1;
      }

      for(; xmin <= xmax; ++xmin)
      {
         for(i=ymin; i<=ymax; ++i)
         {
            glcd_pixel(xmin, i, color);
         }
      }
   }
   else
   {
      glcd_line(x1, y1, x2, y1, color);      // Draw the 4 sides
      glcd_line(x1, y2, x2, y2, color);
      glcd_line(x1, y1, x1, y2, color);
      glcd_line(x2, y1, x2, y2, color);
   }
}

//////////////////////////////////////////////////////////////////////
// Purpose:       Draw a bar (wide line) on a graphic LCD
// Inputs:        (x1, y1) - the start coordinate
//                (x2, y2) - the end coordinate
//                width  - The number of pixels wide
//                color - ON or OFF
//////////////////////////////////////////////////////////////////////
#ifdef LARGE_LCD
void glcd_bar(int16 x1, int16 y1, int16 x2, int16 y2, int8 width, int1 color)
#else
void glcd_bar(int8 x1, int8 y1, int8 x2, int8 y2, int8 width, int1 color)
#endif
{
   int8         half_width;
   signed int16 dy, dx;
   signed int8  addx=1, addy=1, j;
   signed int16 P, diff, c1, c2;

   #ifdef LARGE_LCD
   int16 i=0;
   dx = abs((signed int16)(x2 - x1));
   dy = abs((signed int16)(y2 - y1));
   #else
   int8 i=0;
   dx = abs((signed int8)(x2 - x1));
   dy = abs((signed int8)(y2 - y1));
   #endif

   half_width = width/2;
   c1 = -(dx*x1 + dy*y1);
   c2 = -(dx*x2 + dy*y2);

   if(x1 > x2)
   {
      signed int16 temp;
      temp = c1;
      c1 = c2;
```

```
   c2 = temp;
   addx = -1;
}
if(y1 > y2)
{
   signed int16 temp;
   temp = c1;
   c1 = c2;
   c2 = temp;
   addy = -1;
}

if(dx >= dy)
{
   P = 2*dy - dx;
   diff = P - dx;

   for(i=0; i<=dx; ++i)
   {
      for(j=-half_width; j<half_width+width%2; ++j)
      {
         #ifdef LARGE_LCD
         int16 temp;
         #else
         int8 temp;
         #endif

         temp = dx*x1+dy*(y1+j);    // Use more RAM to increase speed
         if(temp+c1 >= 0 && temp+c2 <=0)
            glcd_pixel(x1, y1+j, color);
      }
      if(P < 0)
      {
         P  += 2*dy;
         x1 += addx;
      }
      else
      {
         P  += diff;
         x1 += addx;
         y1 += addy;
      }
   }
}
else
{
   P = 2*dx - dy;
   diff = P - dy;

   for(i=0; i<=dy; ++i)
   {
      if(P < 0)
      {
         P  += 2*dx;
         y1 += addy;
      }
      else
      {
         P  += diff;
         x1 += addx;
         y1 += addy;
      }
      for(j=-half_width; j<half_width+width%2; ++j)
      {
         #ifdef LARGE_LCD
         int16 temp;
         #else
         int8 temp;
```

```
            #endif

            temp = dx*x1+dy*(y1+j);    // Use more RAM to increase speed
            if(temp+c1 >= 0 && temp+c2 <=0)
               glcd_pixel(x1+j, y1, color);
         }
      }
   }
}


////////////////////////////////////////////////////////////////////////
// Purpose:       Draw a circle on a graphic LCD
// Inputs:        (x,y) - the center of the circle
//                radius - the radius of the circle
//                fill - YES or NO
//                color - ON or OFF
////////////////////////////////////////////////////////////////////////
#ifdef LARGE_LCD
void glcd_circle(int16 x, int16 y, int16 radius, int1 fill, int1 color)
#else
void glcd_circle(int8 x, int8 y, int8 radius, int1 fill, int1 color)
#endif
{
   #ifdef LARGE_LCD
   signed int16 a, b, P;
   #else
   signed int8  a, b, P;
   #endif

   a = 0;
   b = radius;
   P = 1 - radius;

   do
   {
      if(fill)
      {
         glcd_line(x-a, y+b, x+a, y+b, color);
         glcd_line(x-a, y-b, x+a, y-b, color);
         glcd_line(x-b, y+a, x+b, y+a, color);
         glcd_line(x-b, y-a, x+b, y-a, color);
      }
      else
      {
         glcd_pixel(a+x, b+y, color);
         glcd_pixel(b+x, a+y, color);
         glcd_pixel(x-a, b+y, color);
         glcd_pixel(x-b, a+y, color);
         glcd_pixel(b+x, y-a, color);
         glcd_pixel(a+x, y-b, color);
         glcd_pixel(x-a, y-b, color);
         glcd_pixel(x-b, y-a, color);
      }

      if(P < 0)
         P += 3 + 2 * a++;
      else
         P += 5 + 2 * (a++ - b--);
    } while(a <= b);
}


////////////////////////////////////////////////////////////////////////
// Purpose:       Write text on a graphic LCD
// Inputs:        (x,y) - The upper left coordinate of the first letter
//                textptr - A pointer to an array of text to display
//                size - The size of the text: 1 = 5x7, 2 = 10x14, ...
```

```
//               color - ON or OFF
////////////////////////////////////////////////////////////////////////
#ifdef LARGE_LCD
void glcd_text57(int16 x, int16 y, char* textptr, int8 size, int1 color)
#else
void glcd_text57(int8 x, int8 y, char* textptr, int8 size, int1 color)
#endif
{
   int8 j, k, l, m;                      // Loop counters
   int8 pixelData[5];                    // Stores character data

   for(; *textptr != '\0'; ++textptr, ++x)// Loop through the passed string
   {
      if(*textptr < 'S') // Checks if the letter is in the first font array
         memcpy(pixelData, FONT[*textptr - ' '], 5);
      else if(*textptr <= '~') // Check if the letter is in the second font
   array
         memcpy(pixelData, FONT2[*textptr - 'S'], 5);
      else
         memcpy(pixelData, FONT[0], 5);   // Default to space

      // Handles newline and carriage returns
      switch(*textptr)
      {
         case '\n':
            y += 7*size + 1;
            continue;
         case '\r':
            x = 0;
            continue;
      }

      if(x+5*size >= GLCD_WIDTH)          // Performs character wrapping
      {
         x = 0;                           // Set x at far left position
         y += 7*size + 1;                 // Set y at next position down
      }
      for(j=0; j<5; ++j, x+=size)         // Loop through character byte data
      {
         for(k=0; k < 7; ++k)             // Loop through the vertical pixels
         {
            if(bit_test(pixelData[j], k)) // Check if the pixel should be set
            {
               for(l=0; l < size; ++l)    // These two loops change the
               {                          // character's size
                  for(m=0; m < size; ++m)
                  {
                     glcd_pixel(x+m, y+k*size+l, color); // Draws the pixel
                  }
               }
            }
         }
      }
   }
}

#endif

////////////////////////////////////////////////////////////////////////
////
////                        GLCD_Software.c
////  DESCRPICION: Funciones para dibujo en la pantalla GLCD,
////  con pequeñas modificaciones de la librería graphics.c
////  de PIC CSS C COMPILER
////
////  NOMBRE: JULIO CESAR RUIZ HERRERA
////   CARNE: 03038
////   FECHA: NOVIEMBRE 2007
```

```
////
////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////
////                        graphics.c                                ////
////                                                                  ////
////   This file contains functions to draw lines, rectangles, bars,  ////
////   circles and text to a display. A function which draws a        ////
////   single pixel must be defined before calling the functions in   ////
////   this file. Call it glcd_pixel(x, y, color) where x is the      ////
////   horizontal coordinate, y is the vertical coordinate, and       ////
////   color is 1 bit to turn the pixel on or off.                    ////
////                                                                  ////
////   * Note: (0, 0) is treated as the upper left corner             ////
////                                                                  ////
////////////////////////////////////////////////////////////////////////
////                                                                  ////
////  glcd_line(x1, y1, x2, y2, color)                                ////
////     * Draws a line from the first point to the second point      ////
////       with the given color                                       ////
////       - color can be ON or OFF                                   ////
////                                                                  ////
////  glcd_rect(x1, y1, x2, y2, fill, color)                          ////
////     * Draws a rectangle with one corner at point (x1,y1) and     ////
////       the other corner at point (x2,y2)                          ////
////       - fill can be YES or NO                                    ////
////       - color can be ON or OFF                                   ////
////                                                                  ////
////  glcd_bar(x1, y1, x2, y2, width, color)                          ////
////     * Draws a bar (wide line) from the first point to the        ////
////       second point                                               ////
////       - width is the number of pixels wide                       ////
////       - color is ON or OFF                                       ////
////                                                                  ////
////  glcd_circle(x, y, radius, fill, color)                          ////
////     * Draws a circle with center at (x,y)                        ////
////       - fill can be YES or NO                                    ////
////       - color can be ON or OFF                                   ////
////                                                                  ////
////  glcd_text57(x, y, textptr, size, color)                         ////
////     * Write the null terminated text pointed to by textptr with  ////
////       the upper left coordinate of the first character at (x,y)  ////
////       Characters are 5 pixels wide and 7 pixels tall             ////
////       - size is an integer that scales the size of the text      ////
////       - color is ON or OFF                                       ////
////     * Note - This function wraps characters to the next line     ////
////              use #define GLCD_WIDTH to specify a display width  ////
////                                                                  ////
////////////////////////////////////////////////////////////////////////
////        (C) Copyright 1996, 2004 Custom Computer Services         ////
//// This source code may only be used by licensed users of the CCS   ////
//// C compiler.  This source code may only be distributed to other   ////
//// licensed users of the CCS C compiler.  No other use,             ////
//// reproduction or distribution is permitted without written        ////
//// permission.  Derivative programs created using this software     ////
//// in object code form are not restricted in any way.               ////
////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////
#ifndef GRAPHICS_DRAWING_FUNCTIONS
#define GRAPHICS_DRAWING_FUNCTIONS
////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////
#ifndef ON
#define ON  1
```

```
#endif

#ifndef OFF
#define OFF 0
#endif

#ifndef YES
#define YES 1
#endif

#ifndef NO
#define NO  0
#endif
////////////////////////////////////////////////////////////////////////
/*

# Play
$ Stop
% Rec
& Pause

{ Antena On
} Antena Off
[ PC on
] PC off
' PC
^ iP

*/
////////////////////////////////////////////////////////////////////////
//// Defines a 5x7 font
////////////////////////////////////////////////////////////////////////
const int8 FONT[51][5] ={0x00, 0x00, 0x00, 0x00, 0x00, // SPACE
                         0x00, 0x00, 0x5F, 0x00, 0x00, // !
                         0x00, 0x03, 0x00, 0x03, 0x00, // "
                        //0x14, 0x3E, 0x14, 0x3E, 0x14, // #
                         0x00, 0x7F, 0x3E, 0x1C, 0x08, // Play
                        //0x24, 0x2A, 0x7F, 0x2A, 0x12, // $
                         0x00, 0x3C, 0x3C, 0x3C, 0x3C, // Stop
                        //0x43, 0x33, 0x08, 0x66, 0x61, // %
                         0x38, 0x7C, 0x7C, 0x7C, 0x38, // Rec
                        //0x36, 0x49, 0x55, 0x22, 0x50, // &
                         0x7E, 0x7E, 0x00, 0x7E, 0x7E, // Pause
                        //0x00, 0x05, 0x03, 0x00, 0x00, // '
                         0x7F, 0x09, 0x7F, 0x42, 0x66, // PC
                         0x00, 0x1C, 0x22, 0x41, 0x00, // (
                         0x00, 0x41, 0x22, 0x1C, 0x00, // )
                         0x14, 0x08, 0x3E, 0x08, 0x14, // *
                         0x08, 0x08, 0x3E, 0x08, 0x08, // +
                         0x00, 0x50, 0x30, 0x00, 0x00, // ,
                         0x08, 0x08, 0x08, 0x08, 0x08, // -
                         0x00, 0x60, 0x60, 0x00, 0x00, // .
                         0x20, 0x10, 0x08, 0x04, 0x02, // /
                         0x3E, 0x51, 0x49, 0x45, 0x3E, // 0
                         0x00, 0x04, 0x02, 0x7F, 0x00, // 1
                         0x42, 0x61, 0x51, 0x49, 0x46, // 2
                         0x22, 0x41, 0x49, 0x49, 0x36, // 3
                         0x18, 0x14, 0x12, 0x7F, 0x10, // 4
                         0x27, 0x45, 0x45, 0x45, 0x39, // 5
                         0x3E, 0x49, 0x49, 0x49, 0x32, // 6
                         0x01, 0x01, 0x71, 0x09, 0x07, // 7
                         0x36, 0x49, 0x49, 0x49, 0x36, // 8
                         0x26, 0x49, 0x49, 0x49, 0x3E, // 9
                         0x00, 0x36, 0x36, 0x00, 0x00, // :
                         0x00, 0x56, 0x36, 0x00, 0x00, // ;
                         0x08, 0x14, 0x22, 0x41, 0x00, // <
                         0x14, 0x14, 0x14, 0x14, 0x14, // =
                         0x00, 0x41, 0x22, 0x14, 0x08, // >
```

```
                            0x02, 0x01, 0x51, 0x09, 0x06, // ?
                            0x3E, 0x41, 0x59, 0x55, 0x5E, // @
                            0x7E, 0x09, 0x09, 0x09, 0x7E, // A
                            0x7F, 0x49, 0x49, 0x49, 0x36, // B
                            0x3E, 0x41, 0x41, 0x41, 0x22, // C
                            0x7F, 0x41, 0x41, 0x41, 0x3E, // D
                            0x7F, 0x49, 0x49, 0x49, 0x41, // E
                            0x7F, 0x09, 0x09, 0x09, 0x01, // F
                            0x3E, 0x41, 0x41, 0x49, 0x3A, // G
                            0x7F, 0x08, 0x08, 0x08, 0x7F, // H
                            0x00, 0x41, 0x7F, 0x41, 0x00, // I
                            0x30, 0x40, 0x40, 0x40, 0x3F, // J
                            0x7F, 0x08, 0x14, 0x22, 0x41, // K
                            0x7F, 0x40, 0x40, 0x40, 0x40, // L
                            0x7F, 0x02, 0x0C, 0x02, 0x7F, // M
                            0x7F, 0x02, 0x04, 0x08, 0x7F, // N
                            0x3E, 0x41, 0x41, 0x41, 0x3E, // O
                            0x7F, 0x09, 0x09, 0x09, 0x06, // P
                            0x1E, 0x21, 0x21, 0x21, 0x5E, // Q
                            0x7F, 0x09, 0x09, 0x09, 0x76};// R

const int8 FONT2[44][5]={0x26, 0x49, 0x49, 0x49, 0x32, // S
                            0x01, 0x01, 0x7F, 0x01, 0x01, // T
                            0x3F, 0x40, 0x40, 0x40, 0x3F, // U
                            0x1F, 0x20, 0x40, 0x20, 0x1F, // V
                            0x7F, 0x20, 0x10, 0x20, 0x7F, // W
                            0x41, 0x22, 0x1C, 0x22, 0x41, // X
                            0x07, 0x08, 0x70, 0x08, 0x07, // Y
                            0x61, 0x51, 0x49, 0x45, 0x43, // Z
                           //0x00, 0x7F, 0x41, 0x00, 0x00, // [
                            0xDE, 0xDF, 0xFF, 0xDF, 0x00, // PC ON
                            0x02, 0x04, 0x08, 0x10, 0x20, // \
                           //0x00, 0x00, 0x41, 0x7F, 0x00, // ]
                            0xDF, 0xD1, 0xF1, 0xD1, 0xDF, // PC OFF
                           //0x04, 0x02, 0x01, 0x02, 0x04, // ^
                            0x7A, 0x00, 0x7E, 0x0A, 0x0E, //iP
                            0x40, 0x40, 0x40, 0x40, 0x40, // _
                            0x00, 0x01, 0x02, 0x04, 0x00, // `
                            0x20, 0x54, 0x54, 0x54, 0x78, // a
                            0x7F, 0x44, 0x44, 0x44, 0x38, // b
                            0x38, 0x44, 0x44, 0x44, 0x44, // c
                            0x38, 0x44, 0x44, 0x44, 0x7F, // d
                            0x38, 0x54, 0x54, 0x54, 0x18, // e
                            0x04, 0x04, 0x7E, 0x05, 0x05, // f
                            0x08, 0x54, 0x54, 0x54, 0x3C, // g
                            0x7F, 0x08, 0x04, 0x04, 0x78, // h
                            0x00, 0x44, 0x7D, 0x40, 0x00, // i
                            0x20, 0x40, 0x44, 0x3D, 0x00, // j
                            0x7F, 0x10, 0x28, 0x44, 0x00, // k
                            0x00, 0x41, 0x7F, 0x40, 0x00, // l
                            0x7C, 0x04, 0x78, 0x04, 0x78, // m
                            0x7C, 0x08, 0x04, 0x04, 0x78, // n
                            0x38, 0x44, 0x44, 0x44, 0x38, // o
                            0x7C, 0x14, 0x14, 0x14, 0x08, // p
                            0x08, 0x14, 0x14, 0x14, 0x7C, // q
                            0x00, 0x7C, 0x08, 0x04, 0x04, // r
                            0x48, 0x54, 0x54, 0x54, 0x20, // s
                            0x04, 0x04, 0x3F, 0x44, 0x44, // t
                            0x3C, 0x40, 0x40, 0x20, 0x7C, // u
                            0x1C, 0x20, 0x40, 0x20, 0x1C, // v
                            0x3C, 0x40, 0x30, 0x40, 0x3C, // w
                            0x44, 0x28, 0x10, 0x28, 0x44, // x
                            0x0C, 0x50, 0x50, 0x50, 0x3C, // y
                            0x44, 0x64, 0x54, 0x4C, 0x44, // z
                           //0x00, 0x08, 0x36, 0x41, 0x41, // {
                            0x03, 0x07, 0xFF, 0x07, 0x03, //Atena ON
                            0x00, 0x00, 0x7F, 0x00, 0x00, // |
                           //0x41, 0x41, 0x36, 0x08, 0x00, // }
```

```
                        0x03, 0x05, 0xF9, 0x05, 0x03, //Antena OFF

                        0x02, 0x01, 0x02, 0x04, 0x02};// ~
////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////
// Purpose:       Draw a line on a graphic LCD using Bresenham's
//                line drawing algorithm
// Inputs:        (x1, y1) - the start coordinate
//                (x2, y2) - the end coordinate
//                color - ON or OFF
// Dependencies:  glcd_pixel()
////////////////////////////////////////////////////////////////////////
#ifdef LARGE_LCD
void glcd_line(int16 x1, int16 y1, int16 x2, int16 y2, int1 color)
#else
void glcd_line(int8 x1, int8 y1, int8 x2, int8 y2, int1 color)
#endif
{
   int16        dy, dx;
   signed int8  addx=1, addy=1;
   signed int16 P, diff;

   #ifdef LARGE_LCD
   int16 i=0;
   dx = abs((signed int16)(x2 - x1));
   dy = abs((signed int16)(y2 - y1));
   #else
   int8 i=0;
   dx = abs((signed int8)(x2 - x1));
   dy = abs((signed int8)(y2 - y1));
   #endif

   if(x1 > x2)
      addx = -1;
   if(y1 > y2)
      addy = -1;

   if(dx >= dy)
   {
      dy *= 2;
      P = dy - dx;
      diff = P - dx;

      for(; i<=dx; ++i)
      {
         glcd_pixel(x1, y1, color);

         if(P < 0)
         {
            P  += dy;
            x1 += addx;
         }
         else
         {
            P  += diff;
            x1 += addx;
            y1 += addy;
         }
      }
   }
   else
   {
      dx *= 2;
      P = dx - dy;
      diff = P - dy;
```

```
      for(; i<=dy; ++i)
      {
         glcd_pixel(x1, y1, color);

         if(P < 0)
         {
            P  += dx;
            y1 += addy;
         }
         else
         {
            P  += diff;
            x1 += addx;
            y1 += addy;
         }
      }
   }
}


////////////////////////////////////////////////////////////////////////
// Purpose:       Draw a rectangle on a graphic LCD
// Inputs:        (x1, y1) - the start coordinate
//                (x2, y2) - the end coordinate
//                fill  - YES or NO
//                color - ON or OFF
// Dependencies:  glcd_pixel(), glcd_line()
////////////////////////////////////////////////////////////////////////
#ifdef LARGE_LCD
void glcd_rect(int16 x1, int16 y1, int16 x2, int16 y2, int1 fill, int1 color)
#else
void glcd_rect(int8 x1, int8 y1, int8 x2, int8 y2, int1 fill, int1 color)
#endif
{
   if(fill)
   {
      #ifdef LARGE_LCD
      int16 i, xmin, xmax, ymin, ymax;
      #else
      int8  i, xmin, xmax, ymin, ymax;
      #endif

      if(x1 < x2)                           //  Find x min and max
      {
         xmin = x1;
         xmax = x2;
      }
      else
      {
         xmin = x2;
         xmax = x1;
      }

      if(y1 < y2)                           // Find the y min and max
      {
         ymin = y1;
         ymax = y2;
      }
      else
      {
         ymin = y2;
         ymax = y1;
      }

      for(; xmin <= xmax; ++xmin)
      {
         for(i=ymin; i<=ymax; ++i)
         {
```

```
            glcd_pixel(xmin, i, color);
         }
      }
   }
   else
   {
      glcd_line(x1, y1, x2, y1, color);      // Draw the 4 sides
      glcd_line(x1, y2, x2, y2, color);
      glcd_line(x1, y1, x1, y2, color);
      glcd_line(x2, y1, x2, y2, color);
   }
}

////////////////////////////////////////////////////////////////////////
// Purpose:       Draw a bar (wide line) on a graphic LCD
// Inputs:        (x1, y1) - the start coordinate
//                (x2, y2) - the end coordinate
//                width  - The number of pixels wide
//                color - ON or OFF
////////////////////////////////////////////////////////////////////////
#ifdef LARGE_LCD
void glcd_bar(int16 x1, int16 y1, int16 x2, int16 y2, int8 width, int1 color)
#else
void glcd_bar(int8 x1, int8 y1, int8 x2, int8 y2, int8 width, int1 color)
#endif
{
   int8         half_width;
   signed int16 dy, dx;
   signed int8  addx=1, addy=1, j;
   signed int16 P, diff, c1, c2;

   #ifdef LARGE_LCD
   int16 i=0;
   dx = abs((signed int16)(x2 - x1));
   dy = abs((signed int16)(y2 - y1));
   #else
   int8 i=0;
   dx = abs((signed int8)(x2 - x1));
   dy = abs((signed int8)(y2 - y1));
   #endif

   half_width = width/2;
   c1 = -(dx*x1 + dy*y1);
   c2 = -(dx*x2 + dy*y2);

   if(x1 > x2)
   {
      signed int16 temp;
      temp = c1;
      c1 = c2;
      c2 = temp;
      addx = -1;
   }
   if(y1 > y2)
   {
      signed int16 temp;
      temp = c1;
      c1 = c2;
      c2 = temp;
      addy = -1;
   }

   if(dx >= dy)
   {
      P = 2*dy - dx;
      diff = P - dx;

      for(i=0; i<=dx; ++i)
```

```
      {
         for(j=-half_width; j<half_width+width%2; ++j)
         {
            #ifdef LARGE_LCD
            int16 temp;
            #else
            int8 temp;
            #endif

            temp = dx*x1+dy*(y1+j);    // Use more RAM to increase speed
            if(temp+c1 >= 0 && temp+c2 <=0)
               glcd_pixel(x1, y1+j, color);
         }
         if(P < 0)
         {
            P  += 2*dy;
            x1 += addx;
         }
         else
         {
            P  += diff;
            x1 += addx;
            y1 += addy;
         }
      }
   }
   else
   {
      P = 2*dx - dy;
      diff = P - dy;

      for(i=0; i<=dy; ++i)
      {
         if(P < 0)
         {
            P  += 2*dx;
            y1 += addy;
         }
         else
         {
            P  += diff;
            x1 += addx;
            y1 += addy;
         }
         for(j=-half_width; j<half_width+width%2; ++j)
         {
            #ifdef LARGE_LCD
            int16 temp;
            #else
            int8 temp;
            #endif

            temp = dx*x1+dy*(y1+j);    // Use more RAM to increase speed
            if(temp+c1 >= 0 && temp+c2 <=0)
               glcd_pixel(x1+j, y1, color);
         }
      }
   }
}


////////////////////////////////////////////////////////////////////////
// Purpose:       Draw a circle on a graphic LCD
// Inputs:        (x,y) - the center of the circle
//                radius - the radius of the circle
//                fill - YES or NO
//                color - ON or OFF
////////////////////////////////////////////////////////////////////////
```

```
#ifdef LARGE_LCD
void glcd_circle(int16 x, int16 y, int16 radius, int1 fill, int1 color)
#else
void glcd_circle(int8 x, int8 y, int8 radius, int1 fill, int1 color)
#endif
{
   #ifdef LARGE_LCD
   signed int16 a, b, P;
   #else
   signed int8  a, b, P;
   #endif

   a = 0;
   b = radius;
   P = 1 - radius;

   do
   {
      if(fill)
      {
         glcd_line(x-a, y+b, x+a, y+b, color);
         glcd_line(x-a, y-b, x+a, y-b, color);
         glcd_line(x-b, y+a, x+b, y+a, color);
         glcd_line(x-b, y-a, x+b, y-a, color);
      }
      else
      {
         glcd_pixel(a+x, b+y, color);
         glcd_pixel(b+x, a+y, color);
         glcd_pixel(x-a, b+y, color);
         glcd_pixel(x-b, a+y, color);
         glcd_pixel(b+x, y-a, color);
         glcd_pixel(a+x, y-b, color);
         glcd_pixel(x-a, y-b, color);
         glcd_pixel(x-b, y-a, color);
      }

      if(P < 0)
         P += 3 + 2 * a++;
      else
         P += 5 + 2 * (a++ - b--);
    } while(a <= b);
}


////////////////////////////////////////////////////////////////////////
// Purpose:       Write text on a graphic LCD
// Inputs:        (x,y) - The upper left coordinate of the first letter
//                textptr - A pointer to an array of text to display
//                size - The size of the text: 1 = 5x7, 2 = 10x14, ...
//                color - ON or OFF
////////////////////////////////////////////////////////////////////////
#ifdef LARGE_LCD
void glcd_text57(int16 x, int16 y, char* textptr, int8 size, int1 color)
#else
void glcd_text57(int8 x, int8 y, char* textptr, int8 size, int1 color)
#endif
{
   int8 j, k, l, m;                      // Loop counters
   int8 pixelData[5];                    // Stores character data

   for(; *textptr != '\0'; ++textptr, ++x)// Loop through the passed string
   {
      if(*textptr < 'S') // Checks if the letter is in the first font array
         memcpy(pixelData, FONT[*textptr - ' '], 5);
      else if(*textptr <= '~') // Check if the letter is in the second font
   array
         memcpy(pixelData, FONT2[*textptr - 'S'], 5);
```

```
      else
         memcpy(pixelData, FONT[0], 5);   // Default to space

      // Handles newline and carriage returns
      switch(*textptr)
      {
         case '\n':
            y += 7*size + 1;
            continue;
         case '\r':
            x = 0;
            continue;
      }

      if(x+5*size >= GLCD_WIDTH)          // Performs character wrapping
      {
         x = 0;                           // Set x at far left position
         y += 7*size + 1;                 // Set y at next position down
      }
      for(j=0; j<5; ++j, x+=size)         // Loop through character byte data
      {
         for(k=0; k < 7; ++k)             // Loop through the vertical pixels
         {
            if(bit_test(pixelData[j], k)) // Check if the pixel should be set
            {
               for(l=0; l < size; ++l)    // These two loops change the
               {                          // character's size
                  for(m=0; m < size; ++m)
                  {
                     glcd_pixel(x+m, y+k*size+l, color); // Draws the pixel
                  }
               }
            }
         }
      }
   }
}

#endif
```

D. Código Menus

```
////////////////////////////////////////////////////////////////////////
////
////                        Menu_IntroMain.c
////  DESCRPICION: Funciones para dibujo en la pantalla GLCD,
////  con pequeñas modificaciones de la librería graphics.c
////  de PIC CSS C COMPILER
////
////  NOMBRE: JULIO CESAR RUIZ HERRERA
////   CARNE: 03038
////   FECHA: NOVIEMBRE 2007
////
////////////////////////////////////////////////////////////////////////
////
////  SB_Dis(int div) - Genera una barra lateral, del menu principal,
////  div es el tamaño que desea el tamaño del cursor de la barra.
////
////  TopBar_Dis(short Mode) - Genera la barra superior con el nombre
////  del menu y su numero. Ademas de los iconos de alarma.
////
////
////
////////////////////////////////////////////////////////////////////////

//Función de dibujo del menú de introducción.
void Dis_Intro(){

   glcd_fillscreen(on);

   glcd_rect(1, Punt_Ent-2, 126, Punt_Ent+14+4, NO, OFF);


   strcpy(Menu,"R17 TeachIN");
   glcd_text57(4, Punt_Ent, Menu, 2, off);

                strcpy(Menu,"Presione  Enter");
   if (lang ==0) strcpy(Menu,"Press Enter (E)");

   glcd_text57(23, 50, Menu, 1, off);
}

void Boton_Intro(char xx){
   switch(xx){
      case 'E' :
            msel =1;
            mpos =1;
      break;

      case 'C' :
            msel =1;
            mpos =1;
      break;
   }
}



void SB_Dis(int div){
   //BARRA lateral
   glcd_rect(0, Inity_bar,  2, 63 ,no,on);
   glcd_line(1,  Inity_bar+1  +    (div  +1)*mpos  ,       1,  Inity_bar+1  +
   (div+1)*(mpos+1),    ON );

}
```

```
void TopBar_Dis(short mode){

   char  Mposc[2] = "0";
   char temp;

   //Number
   glcd_rect(0,0,8,10, no,mode);

   if (!mode)  glcd_rect(1,1,7,9, yes, !mode);
   Mposc[0] = (mpos+49);
   glcd_text57(2,2,Mposc, 1, mode);

   glcd_rect(9,0,127,10, mode, mode);
   glcd_text57(11,2,Menu, 1, !mode);


   strcpy(Menu, " ");
   if (PC_Online)     strcpy(Menu, "]'");
   else if (tilt)     strcpy(Menu, "['");

   glcd_text57(83, 2, Menu,1,!mode);

   strcpy(Menu, "   ");
   if (Wip_stat==0)    strcpy(Menu, "}W^");
   else if (tilt)      strcpy(Menu, "{W^");

   glcd_text57(99, 2, Menu,1,!mode);

   Mposc[0] = '#' + recording_status;
   glcd_text57(120, 2, Mposc,1, !mode);

}



void Boton_Intro(char xx){
   switch(xx){
      case 'E' :
            msel =1;
            mpos =1;
      break;

      case 'C' :
            msel =1;
            mpos =1;
      break;
   }
}
////////////////////////////////////////////////////////////////////////
////
////                        Menu_IntroMain.c
////  DESCRPICION: Funciones para dibujo en la pantalla GLCD,
////  con pequeñas modificaciones de la librería graphics.c
////  de PIC CSS C COMPILER
////
////  NOMBRE: JULIO CESAR RUIZ HERRERA
////   CARNE: 03038
////   FECHA: NOVIEMBRE 2007
////
////////////////////////////////////////////////////////////////////////
////
////  SB_Dis(int div) - Genera una barra lateral, del menu principal,
////  div es el tamaño que desea el tamaño del cursor de la barra.
////
////  TopBar_Dis(short Mode) - Genera la barra superior con el nombre
////  del menu y su numero. Ademas de los iconos de alarma.
////
////
```

```
////
//////////////////////////////////////////////////////////////////////////

//Función de dibujo del menú de introducción.
void Dis_Intro(){

   glcd_fillscreen(on);

   glcd_rect(1, Punt_Ent-2, 126, Punt_Ent+14+4, NO, OFF);


   strcpy(Menu,"R17 TeachIN");
   glcd_text57(4, Punt_Ent, Menu, 2, off);

                strcpy(Menu,"Presione  Enter");
   if (lang ==0) strcpy(Menu,"Press Enter (E)");

   glcd_text57(23, 50, Menu, 1, off);
}

void Boton_Intro(char xx){
   switch(xx){
      case 'E' :
            msel =1;
            mpos =1;
      break;

      case 'C' :
            msel =1;
            mpos =1;
      break;
   }
}



void SB_Dis(int div){
   //BARRA lateral
   glcd_rect(0, Inity_bar,  2, 63 ,no,on);
   glcd_line(1,  Inity_bar+1  +     (div  +1)*mpos  ,       1,  Inity_bar+1  +
   (div+1)*(mpos+1),    ON );

}


void TopBar_Dis(short mode){

   char  Mposc[2] = "0";
   char temp;

   //Number
   glcd_rect(0,0,8,10, no,mode);

   if (!mode)  glcd_rect(1,1,7,9, yes, !mode);
   Mposc[0] = (mpos+49);
   glcd_text57(2,2,Mposc, 1, mode);

   glcd_rect(9,0,127,10, mode, mode);
   glcd_text57(11,2,Menu, 1, !mode);


   strcpy(Menu, " ");
   if (PC_Online)     strcpy(Menu, "]'");
   else if (tilt)     strcpy(Menu, "['");

   glcd_text57(83, 2, Menu,1,!mode);

   strcpy(Menu, "   ");
```

```
   if (Wip_stat==0)    strcpy(Menu, "}W^");
   else if (tilt)      strcpy(Menu, "{W^");

   glcd_text57(99, 2, Menu,1,!mode);

   Mposc[0] = '#' + recording_status;
   glcd_text57(120, 2, Mposc,1, !mode);

}



void Boton_Intro(char xx){
   switch(xx){
      case 'E' :
            msel =1;
            mpos =1;
      break;

      case 'C' :
            msel =1;
            mpos =1;
      break;
   }
}


void Dis_Main(){

         glcd_fillscreen(off);

         #ifdef LEFTBAR
           //Menubarra
            SB_dis(12);

           //Barsize = 51/4;
            strcpy(Menu,"Menu Inico");

            if (lang ==0) strcpy(Menu,"Main Menu");

            TopBar_Dis(on);

         #else
            glcd_rect(127-8,0,127,10,no,on);
            Mposc[0] = (mpos+49);
            glcd_text57(127-2,2,Mposc, 1,on);

            //BARRA
            glcd_rect(127-2,  Inity_mp-7,   127,  63,    NO,    ON); //Inity_mp +
   Spacey_mp*3 +12
            glcd_line(127-1, Inity_mp-6 + Spacey_mp*mpos ,     127-1, Inity_mp +
   Spacey_mp*mpos + Spacey_mp, ON );

         #endif

        //SUPERIOR
            if ( mpos == 0) {
              mp_set(Last_mp);
              //glcd_text57(Initx_mp, Inity_mp, Menu, 1, on);
              c_skipU++;
            }
            else{
              mp_set(mpos-1);
              glcd_text57(Initx_mp, Inity_mp, Menu, 1, on);
              c_skipU=0;
            }

         //MIDLE
```

```
            glcd_rect(Initx_mp, Inity_mp + Spacey_mp*1, Initx_mp + 85 , Inity_mp
   + Spacey_mp*2 ,1,on);
            mp_set(mpos);
            glcd_text57(Initx_mp+3, Inity_mp + Spacey_mp*1 + 2, Menu,1,off);


         //INFERIOR
            if ( mpos == Last_mp) {
              mp_set(0);
              //glcd_text57(Initx_mp, Inity_mp + Spacey_mp*2 + 3, Menu, 1, on);
              c_skipD++;
            }
            else{
              mp_set(mpos+1);
              glcd_text57(Initx_mp, Inity_mp + Spacey_mp*2 + 3, Menu, 1, on);
              c_skipD=0;
            }
}


void Boton_Main(char xx){
   switch(xx){

      case '1':
               mpos = 0;
      break;

      case '2':
               mpos = 1;
      break;

      case '3':
               mpos = 2;
      break;

      case '4':
               mpos = 3;
      break;

      case 'E':
               msel = 2+mpos;
               mpos = 0;
               //inCl = 0; //"012345"
               //Clear_Mob();
      break;

      case 'C':
               msel = 0;
      break;
      case 'U':
         if (mpos == 0) {
            if (c_skipU >=skip_tresh)  mpos = Last_mp;
            else                       mpos = mpos;
         }

         else  mpos--;
      break;
      case 'D':
          if (mpos == Last_mp){
             if (c_skipD >=skip_tresh)    mpos = 0;
             else                         mpos = mpos;
          }
          else mpos++;
      break;
   }
}
```

A. Función de Transmisión

```
////////////////////////////////////////////////////////////////////////
//    can_enviar(int contador , int id, int *data)
//
//    FUNCION DE TRANSMISION A TRAVES DEL BUS CAN
//    LOS PARAMETROS SON
//       contador -  CONTADOR PROPIO DE CADA NODO PARA VERIFICAR
//                   QUE ESTE NO QUEDE ABANDONADO POR FALTA DE
//                   PRIORIDAD
//       id       -  EL IDENTIFICADOR DEL NODO QUE QUIERE ENVIAR
//       data[0]    -  BYTE CON DATOS MENOS SIGNIFICATIVOS DEL MENSAJE
//       data[1]    -  BYTE CON DATOS MAS SIGNIFICATIVOS DEL MENSAJE
//
int1 can_enviar(int *contador ,int id, int *data) {
   int i;
   int *txd0;
   int *prueba;
   int identificador;
   int temp;

   identificador=id;
   txd0=&TXRXBaD0;


   //se busca un transmisor libre
   //mapea las direcciones del access bank addresses al transmisor vacio
   if (!TXB0CON.txreq) {
      CANCON.win=CAN_WIN_TX0; //CAN_WIN-TX0 equivale a 100
   }
   else if (!TXB1CON.txreq) {
      CANCON.win=CAN_WIN_TX1;  //CAN_WIN-TX1 equivale a 011
   }
   else if (!TXB2CON.txreq) {
      CANCON.win=CAN_WIN_TX2;  //CAN_WIN-TX2 equivale a 010
   }
   else {

      return(0); //no hay transmisor libre
   }

   //setea prioridad del buffer, no se usa.
   TXBaCON.txpri=3;

   if (identificador == display){
         identificador = 10;         // setea el identificador de display
   }else if (identificador == wifiRx){

         identificador = 15;        // setea el identificador de wifiRx
   }else{
            temp = *contador;
            temp ++;
```

```
   switch (temp){  // setea el identificador de sensores dependiendo     del
contador

         case 1: identificador=20;
                 break;
         case 2: identificador=21;
                 break;
         case 3: identificador=22;
```

```
                    *contador=0;
                    break;
             case 4: identificador=23;
                    break;
             case 5: identificador=24;
                    break;
             case 6: identificador=25;
                    break;
             case 7: identificador=26;
                    break;
             case 8: identificador=27;
                    break;
             case 9: identificador=28;
                    break;
            case 10: identificador=29;
                    *contador=0;
                    break;
            default: identificador=30;
                    *contador=0;
                    break;
      }
   }

   //setea el id
   can_set_id(TXRXBaID, identificador, 1);

   //setea tx data count
   TXBaDLC=2;
   TXBaDLC.rtr=0;

    for (i=0; i<2; i++) {
      *txd0=*data;
      txd0++;
      data++;
    }

   //habilita transmission
   TXBaCON.txreq=1;

   CANCON.win=CAN_WIN_RX0;

   return(1);
}
```

R.Funcion de recepción con verificacion de los identificadores

```
int1 canRx(int canNode, int * canCounter, int * canByte){
   struct rx_stat stats;   // Estructura para datos de informacion de la recepcion
   int32 MsgID;            // Identificador del mensaje
   int MsgData[2];         // Buffer que almacena los datos recibidos
   int MsgLength;          // Longitud del mensaje
   int i;                  // Contador general
   int * ptr;              // Puntero para el buffer de datos
   int temp;

// Verificar estado del buffer, saber si esta lleno
   if (RXB0CON.rxful) {
       CANCON.win=CAN_WIN_RX0;
       stats.buffer=0;
       CAN_INT_RXB0IF=0;
       stats.err_ovfl=COMSTAT.rx0ovfl;
       COMSTAT.rx0ovfl=0;
       if (RXB0CON.rxb0dben) {
        stats.filthit=RXB0CON.filthit0;
```

```
        }
    }
    else if ( RXB1CON.rxful ){
        CANCON.win=CAN_WIN_RX1;
        stats.buffer=1;
        CAN_INT_RXB1IF=0;
        stats.err_ovfl=COMSTAT.rx1ovfl;
        COMSTAT.rx1ovfl=0;
        stats.filthit=RXB1CON.filthit;
    }
    else{
          return 0;
    }
    // especificacion del tamano del mensaje
    MsgLength = 2;    // La longitud es fija, 2 bytes
    stats.rtr=RXBaDLC.rtr;
    stats.ext=TXRXBaSIDL.ext;
    MsgId=can_get_id(TXRXBaID,stats.ext);
    ptr = &TXRXBaD0;

           if (canNode == display){
                for ( i = 0; i < 2; i++ ) {
                   *canByte = *ptr;
                   canByte++;
                   ptr++;
                }
           }
           else if (canNode == wifiRx){
                for ( i = 0; i < 2; i++ ) {
                   *canByte = *ptr;
                   canByte++;
                   ptr++;
                }
           }
           else if (canNode == wifiTx){
                for ( i = 0; i < 2; i++ ) {
                   *canByte = *ptr;
                   canByte++;
                   ptr++;
                }
           }else{

             temp = *ptr;
             ptr++;
             temp = *ptr;
           }

    CANCON.win=CAN_WIN_RX0;
    stats.inv=CAN_INT_IRXIF;
    CAN_INT_IRXIF = 0;

    if (stats.buffer) {
      RXB1CON.rxful=0;
    }
    else {
      RXB0CON.rxful=0;
    }



    // Si el buffer recibe mensajes para otros nodos, debe contar
    //    el numero de veces que esto ocurre, para alterar la
    //    prioridad y hacer que el nodo no quede abandonado.


     return 1;
}
```

S. Librería de funciones completa

CAN4R17.H

```
#include <can-18xxx8.h>

#ifndef __CCS_CAN4R17_LIB_DEFINES__
#define __CCS_CAN4R17_LIB_DEFINES__


void can_start(int node);
int1 canRx(int canNode, int * canCounter, int * canByte);
int1 can_enviar(int * contador ,int id, int * data);
void can_init(void);
void can_set_baud(void);
void can_set_mode(CAN_OP_MODE mode);
void can_set_id(int* addr, int32 id, int1 ext);
int32 can_get_id(int * addr, int1 ext);

#endif
```

CAN4R17.C

```
////////////////////////////////////////////////////////////////////////
////
////                        CAN4R17.c
////  Libreria con rutinas para el Microchip PIC 18F458.
////
////  Esta libreria provee las funciones necesarias para la
////  comunicacion entre los nodos de la silla del controlador.
////
////  La comunicacion se realiza utilizando el protocolo CAN y
////  esta libreria esta hecha para ser utilizada con el compilador
////  PIC-C de CCS.
////
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
////  A continuacion se presentan las funciones que se encuentran
////  disponibles de la libreria de Custom Computer Services
////    can_init - Configures the PIC18xxx8 CAN peripheral
////    can_set_baud - Sets the baud rate control registers
////    can_set_mode - Sets the CAN module into a specific mode
////    can_set_id - Sets the standard and extended ID
////    can_get_id - Gets the standard and extended ID
////    can_putd - Sends a message/request with specified ID
////    can_getd - Returns specifid message/request and ID
////    can_kbhit - Returns true if there is data in one of the
////                receive buffers
////    can_tbe - Returns true if the transmit buffer is ready to
////              send more data
////    can_abort - Aborts all pending transmissions
////////////////////////////////////////////////////////////////////////
////
////  Para la transmision se utiliza el pin CANTX (35) y para
////  la recepcion se utiliza el pin CANRX (36). Ambos del
////  microcontrolador Microchip PIC 18F458.
```

```
////
////  Para la conexion al bus de datos es NECESARIO utilizar
////  el transceiver MCP2551 (transmisor/receptor) de Microchip.
////  Con el anterior se cumplen los requisitos determinados
////  para el funcionamiento de la capa fisica del protocolo.
////
//////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////
//    OTRAS LIBRERIAS NECESARIAS
#include <CAN4R17.h>
//////////////////////////////////////////////////////////////////////////
//    MACROS
#define can_kbhit()                (RXB0CON.rxful || RXB1CON.rxful)
#define  can_tbe()                          (!TXB0CON.txreq  ||  !TXB1CON.txreq  ||
   !TXB2CON.txreq)
#define can_abort()                (CANCON.abat=1)


//////////////////////////////////////////////////////////////////////////
//    CONSTANTES DE OPERACION PROPIAS DE LA COMUNICACION LOCAL
#define display   10
#define wifiTx    83
#define wifiRx    15
#define track     85
#define waist     86
#define shoulder  87
#define elbow     88
#define hand      89
#define wrist     90
#define nodeNumber 10
//////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////
//    CAN_START()
//    FUNCION PARA INICIAR EL FUNCIONAMIENTO DEL NODO CAN,
//    SEGUN LOS REQUERIMIENTOS DEL MEGAPROYECTO
//
//    ES NECESARIO QUE CADA NODO, AL INICIAR, INDIQUE QUE FUNCION
//    REALIZARA. POR EJEMPLO, LA BRUJULA DEBERA INDICAR QUE MEDIRA
//    LA POSICION PARA "waist" DE LA SIGUIENTE MANERA
//                  can_start(waist)
//
void can_start(int node){
   can_init();
   //agregar primer envio con prioridad = (normal + 128)
   //can_enviar((node+128),0,0);
}
//////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////
//    canRx(int canNode, int canCounter, int *canByte)
//
//    FUNCION DE RECEPCION A TRAVES DEL BUS CAN
//    LOS PARAMETROS SON
//       canNode     -  NODO QUE HACE LA RECEPCION
//       canCounter  -  CONTADOR PROPIO DE CADA NODO PARA VERIFICAR
//                      QUE ESTE NO QUEDE ABANDONADO POR FALTA DE
//                      PRIORIDAD
//       canByte[1]    -  BYTE CON DATOS MAS SIGNIFICATIVOS DEL MENSAJE
//       canByte[0]    -  BYTE CON DATOS MENOS SIGNIFICATIVOS DEL MENSAJE
//
int1 canRx(int canNode, int * canCounter, int * canByte){
   struct rx_stat stats;   // Estructura para datos de informacion de la recepcion
```

```
   int32 MsgID;              // Identificador del mensaje
   int MsgData[2];         // Buffer que almacena los datos recibidos
   int MsgLength;          // Longitud del mensaje
   int i;                  // Contador general
   int * ptr;              // Puntero para el buffer de datos
   int temp;
   int1 bandera=0;

// Verificar estado del buffer, saber si esta lleno
   if (RXB0CON.rxful) {
       CANCON.win=CAN_WIN_RX0;
       stats.buffer=0;
       CAN_INT_RXB0IF=0;
       stats.err_ovfl=COMSTAT.rx0ovfl;
       COMSTAT.rx0ovfl=0;
       if (RXB0CON.rxb0dben) {
        stats.filthit=RXB0CON.filthit0;
       }
   }
   else if ( RXB1CON.rxful ){
       CANCON.win=CAN_WIN_RX1;
       stats.buffer=1;
       CAN_INT_RXB1IF=0;
       stats.err_ovfl=COMSTAT.rx1ovfl;
       COMSTAT.rx1ovfl=0;
       stats.filthit=RXB1CON.filthit;
   }
   else{
         return 0;
   }
   // especificacion del tamano del mensaje
   MsgLength = 2;    // La longitud es fija, 2 bytes
   stats.rtr=RXBaDLC.rtr;
   stats.ext=TXRXBaSIDL.ext;
   MsgId=can_get_id(TXRXBaID,stats.ext);
   ptr = &TXRXBaD0;


          if (canNode == MsgId){
               for ( i = 0; i < 2; i++ ) {
                  *canByte = *ptr;
                  canByte++;
                  ptr++;
                  bandera=1;
               }
          }else if (canNode == wifiTx && MsgId != display && MsgId != wifiRx){
                  for ( i = 0; i < 2; i++ ) {
                  *canByte = *ptr;
                  canByte++;
                  ptr++;
                  bandera=1;
               }
          }
          else{
            temp = *ptr;
            ptr++;
            temp = *ptr;
          }

   CANCON.win=CAN_WIN_RX0;
   stats.inv=CAN_INT_IRXIF;
   CAN_INT_IRXIF = 0;

   if (stats.buffer) {
```

```
      RXB1CON.rxful=0;
   }
   else {
      RXB0CON.rxful=0;
   }

         if (bandera==1){
            return 1;
         }
         else{
            return 0;
         }

   // Si el buffer recibe mensajes para otros nodos, debe contar
   //    el numero de veces que esto ocurre, para alterar la
   //    prioridad y hacer que el nodo no quede abandonado.

}

////////////////////////////////////////////////////////////////////////
//    can_enviar(int contador , int id, int *data)
//
//    FUNCION DE TRANSMISION A TRAVES DEL BUS CAN
//    LOS PARAMETROS SON
//       contador -  CONTADOR PROPIO DE CADA NODO PARA VERIFICAR
//                   QUE ESTE NO QUEDE ABANDONADO POR FALTA DE
//                   PRIORIDAD
//       id       -  EL IDENTIFICADOR DEL NODO QUE QUIERE ENVIAR
//       data[0]    -  BYTE CON DATOS MENOS SIGNIFICATIVOS DEL MENSAJE
//       data[1]    -  BYTE CON DATOS MAS SIGNIFICATIVOS DEL MENSAJE
//
int1 can_enviar(int *contador ,int id, int *data) {
   int i;
   int *txd0;
   int *prueba;
   int identificador;
   int temp;

   identificador=id;
   txd0=&TXRXBaD0;


   //se busca un transmisor libre
   //mapea las direcciones del access bank addresses al transmisor vacio
   if (!TXB0CON.txreq) {
      CANCON.win=CAN_WIN_TX0; //CAN_WIN-TX0 equivale a 100
   }
   else if (!TXB1CON.txreq) {
      CANCON.win=CAN_WIN_TX1;  //CAN_WIN-TX1 equivale a 011
   }
   else if (!TXB2CON.txreq) {
      CANCON.win=CAN_WIN_TX2;  //CAN_WIN-TX2 equivale a 010
   }
   else {

      return(0); //no hay transmisor libre
   }

   //setea prioridad del buffer, no se usa.
   TXBaCON.txpri=3;

  if(identificador == wifiTx){

            temp = *contador;
```

```
            temp++;
            *contador = temp;

      switch (temp){                          // setea el identificador de sensores
   dependiendo del contador

            case 1: identificador=20;
                    break;
            case 2: identificador=21;
                    break;
            case 3: identificador=22;
                    break;
            case 4: identificador=23;
                    break;
            case 5: identificador=24;
                    break;
            case 6: identificador=25;
                    break;
            case 7: identificador=26;
                    break;
            case 8: identificador=27;
                    break;
            case 9: identificador=28;
                    break;
           case 10: identificador=29;
                    *contador=0;
                    break;
           default: identificador=30;
                    *contador=0;
                    break;
      }
   }

   //setea el id
   can_set_id(TXRXBaID, identificador, 1);

   //setea tx data count
   TXBaDLC=2;
   TXBaDLC.rtr=0;

    for (i=0; i<2; i++) {
      *txd0=*data;
      txd0++;
      data++;
    }

   //habilita transmission
   TXBaCON.txreq=1;

   CANCON.win=CAN_WIN_RX0;

   return(1);
}

////////////////////////////////////////////////////////////////////////
//    A continuacion se presentan algunas funciones basicas
//       obtenidas de la libreria de PICC (CCS)
////////////////////////////////////////////////////////////////////////
//
// can_init()
//
// Initializes PIC18xxx8 CAN peripheral.  Sets the RX filter and masks so the
// CAN peripheral will receive all incoming IDs.  Configures both RX buffers
// to only accept valid valid messages (as opposed to all messages, or all
```

```
// extended message, or all standard messages).  Also sets the tri-state
// setting of B2 to output, and B3 to input (apparently the CAN peripheral
// doesn't keep track of this)
//
// The constants (CAN_USE_RX_DOUBLE_BUFFER, CAN_ENABLE_DRIVE_HIGH,
// CAN_ENABLE_CAN_CAPTURE) are given a default define in the can-18xxx8.h file.
// These default values can be overwritten in the main code, but most
// applications will be fine with these defaults.
//
////////////////////////////////////////////////////////////////////////////
void can_init(void) {
   can_set_mode(CAN_OP_CONFIG);   //must be in config mode before params can be set
   can_set_baud();

   RXB0CON=0;
   RXB0CON.rxm=CAN_RX_VALID;
   RXB0CON.rxb0dben=CAN_USE_RX_DOUBLE_BUFFER;
   RXB1CON=RXB0CON;

   CIOCON.endrhi=CAN_ENABLE_DRIVE_HIGH;
   CIOCON.cancap=CAN_ENABLE_CAN_CAPTURE;

   can_set_id(RX0MASK, CAN_MASK_ACCEPT_ALL, CAN_USE_EXTENDED_ID);  //set mask 0
   can_set_id(RX0FILTER0, 0, CAN_USE_EXTENDED_ID);  //set filter 0 of mask 0
   can_set_id(RX0FILTER1, 0, CAN_USE_EXTENDED_ID);  //set filter 1 of mask 0

   can_set_id(RX1MASK, CAN_MASK_ACCEPT_ALL, CAN_USE_EXTENDED_ID);  //set mask 1
   can_set_id(RX1FILTER2, 0, CAN_USE_EXTENDED_ID);  //set filter 0 of mask 1
   can_set_id(RX1FILTER3, 0, CAN_USE_EXTENDED_ID);  //set filter 1 of mask 1
   can_set_id(RX1FILTER4, 0, CAN_USE_EXTENDED_ID);  //set filter 2 of mask 1
   can_set_id(RX1FILTER5, 0, CAN_USE_EXTENDED_ID);  //set filter 3 of mask 1

   set_tris_b((*0xF93 & 0xFB ) | 0x08);   //b3 is out, b2 is in

   can_set_mode(CAN_OP_NORMAL);
}

////////////////////////////////////////////////////////////////////////////
//
// can_set_baud()
//
// Configures the baud rate control registers.  All the defines here
// are defaulted in the can-18xxx8.h file.  These defaults can, and
// probably should, be overwritten in the main code.
//
// Current defaults are set to work with Microchip's MCP250xxx CAN
// Developers Kit if this PIC is running at 20Mhz.
//
////////////////////////////////////////////////////////////////////////////
void can_set_baud(void) {
   BRGCON1.brp=CAN_BRG_PRESCALAR;
   BRGCON1.sjw=CAN_BRG_SYNCH_JUMP_WIDTH;

   BRGCON2.prseg=CAN_BRG_PROPAGATION_TIME;
   BRGCON2.seg1ph=CAN_BRG_PHASE_SEGMENT_1;
   BRGCON2.sam=CAN_BRG_SAM;
   BRGCON2.seg2phts=CAN_BRG_SEG_2_PHASE_TS;

   BRGCON3.seg2ph=CAN_BRG_PHASE_SEGMENT_2;
   BRGCON3.wakfil=CAN_BRG_WAKE_FILTER;
}

void can_set_mode(CAN_OP_MODE mode) {
   CANCON.reqop=mode;
```

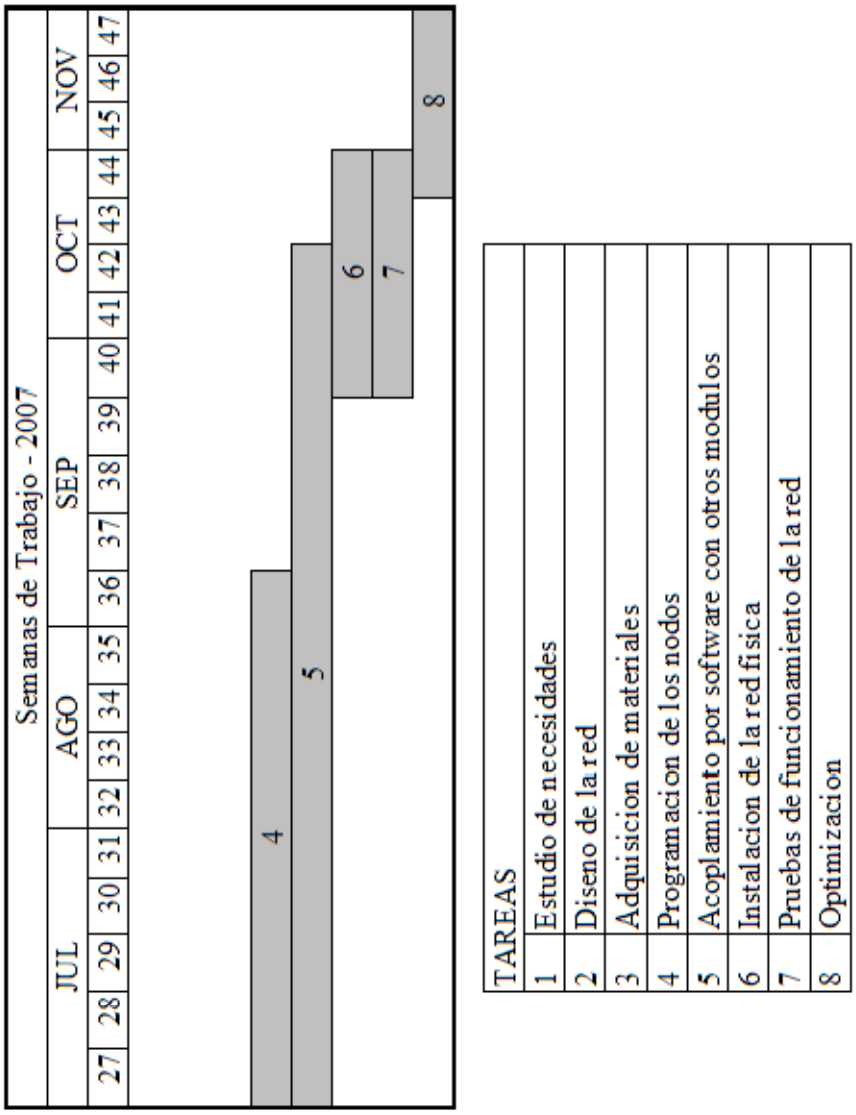
```
   while( (CANSTAT.opmode) != mode );
}


//////////////////////////////////////////////////////////////////////
//
// can_set_id()
//
// Configures the xxxxEIDL, xxxxEIDH, xxxxSIDL and xxxxSIDH registers to
// configure the defined buffer to use the specified ID
//
//   Paramaters:
//     addr - pointer to first byte of ID register, starting with xxxxEIDL.
//            For example, a pointer to RXM1EIDL
//     id - ID to set buffer to
//     ext - Set to TRUE if this buffer uses an extended ID, FALSE if not
//
//////////////////////////////////////////////////////////////////////
void can_set_id(int* addr, int32 id, int1 ext) {
   int *ptr;

   ptr=addr;

   if (ext) {  //extended
      //eidl
      *ptr=make8(id,0); //0:7

      //eidh
      ptr--;
      *ptr=make8(id,1); //8:15

      //sidl
      ptr--;
      *ptr=make8(id,2) & 0x03;   //16:17
      *ptr|=(make8(id,2) << 3) & 0xE0; //18:20
      *ptr|=0x08;


      //sidh
      ptr--;
      *ptr=((make8(id,2) >> 5) & 0x07 ); //21:23
      *ptr|=((make8(id,3) << 3) & 0xF8);//24:28
   }
   else {   //standard
      //eidl
      *ptr=0;

      //eidh
      ptr--;
      *ptr=0;

      //sidl
      ptr--;
      *ptr=(make8(id,0) << 5) & 0xE0;

      //sidh
      ptr--;
      *ptr=(make8(id,0) >> 3) & 0x1F;
      *ptr|=(make8(id,1) << 5) & 0xE0;
   }
}

//////////////////////////////////////////////////////////////////////
//
```

```
// can_get_id()
//
// Returns the ID of the specified buffer.  (The opposite of can_set_id())
// This is used after receiving a message, to see which ID sent the message.
//
//   Paramaters:
//     addr - pointer to first byte of ID register, starting with xxxxEIDL.
//            For example, a pointer to RXM1EIDL
//     ext - Set to TRUE if this buffer uses an extended ID, FALSE if not
//
//   Returns:
//     The ID of the buffer
//
////////////////////////////////////////////////////////////////////////
int32 can_get_id(int * addr, int1 ext) {
   int32 ret;
   int * ptr;

   ret=0;
   ptr=addr;

   if (ext) {
      ret=*ptr;  //eidl

      ptr--;     //eidh
      ret|=((int32)*ptr << 8);

      ptr--;     //sidl
      ret|=((int32)*ptr & 0x03) << 16;
      ret|=((int32)*ptr & 0xE0) << 13;

      ptr--;     //sidh
      ret|=((int32)*ptr << 21);

   }
   else {
      ptr-=2;    //sidl
      ret=((int32)*ptr & 0xE0) >> 5;

      ptr--;     //sidh
      ret|=((int32)*ptr << 3);
   }

   return(ret);
}
```

T. Cronograma de trabajo

A continuación se presenta el cronograma de trabajo, utilizado para el desarrollo de este trabajo de investigación.

Gráfica 44 Cronograma de trabajo (enero – junio)



Gráfica 45 Cronograma de trabajo (julio – noviembre)

U.Red WiFi

Transmisión de datos

Compilador PicC

```
#include  "C:\Documents  and  Settings\Billy  Astroman\Mis  documentos\MEGAPROY\07  -
   picc\tranmision1.h"
  #include <stdio.h>
  #include <stdlib.h>
  #use rs232(baud=57600,xmit=PIN_C6, rcv=PIN_C7)


  int data_transmit[12];      //Arreglo de 12 variables para transmision datos
  int i = 0;                  //Variable de contador



void config(){
  for (i = 0; i < 12; i++)
      data_transmit[i] = 0;   //Inicializar arreglo

}


void main(){

   config();

   while(true){

         //Bucle que leerá de CAN y arma paquetes
         for (i = 0; i < 12; i++)
             data_transmit[i] = 65 + i;

         //Bucle que transmite el paquete de 12 datos
         for (i = 0; i < 12; i++)
             putc(data_transmit[i]);

         delay_ms(1000);




   }

}
```

Circuito para Implementación del WiPort

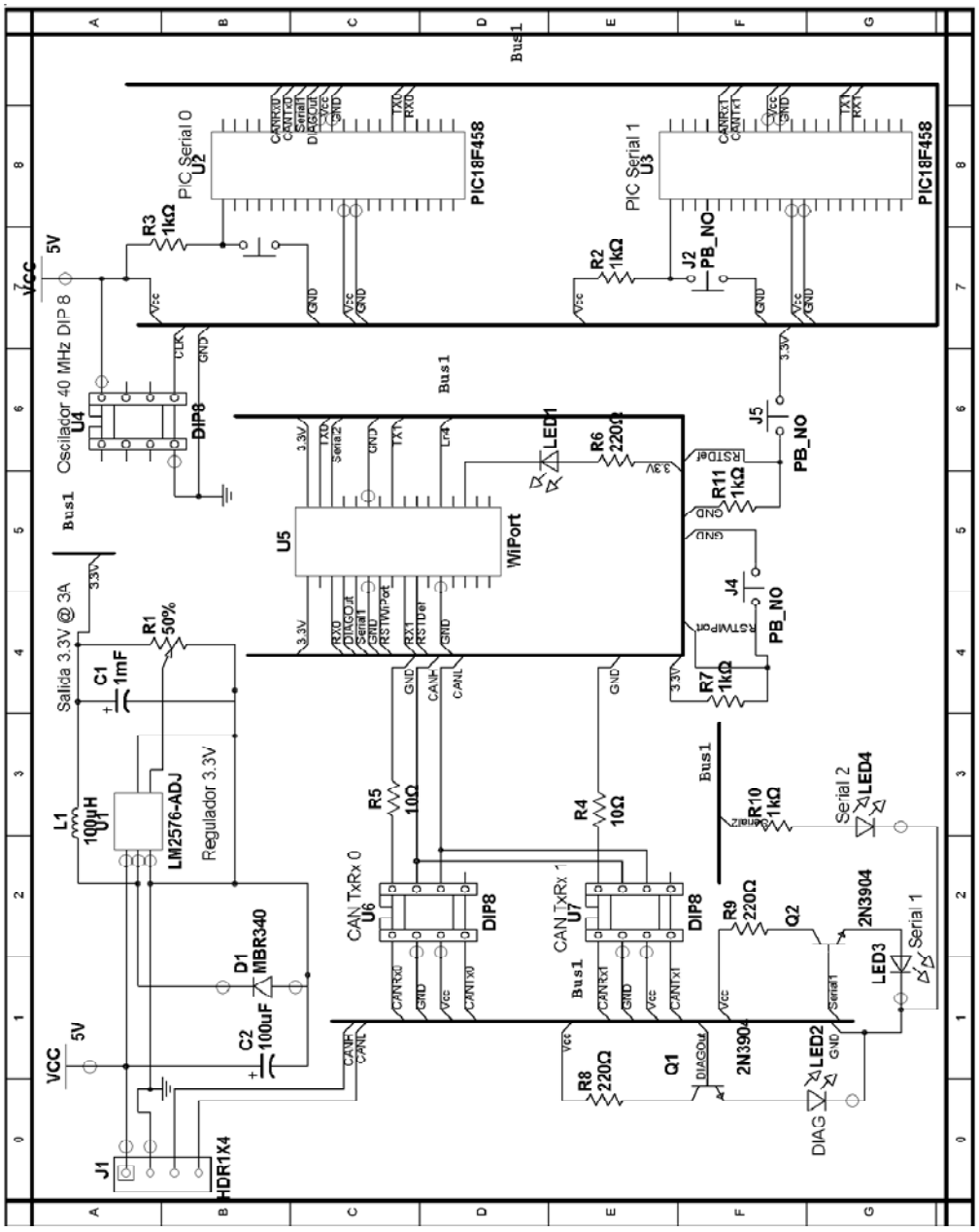


Figura 21. Circuito diseñado para la implementación del WiPort

Archivos de Texto para Pruebas

115,200 bps

Prueba de WiPort con baud rate 115200 - En esta prueba se envio el presente archivo al WiPort - El PIC se programo para hacer eco a todos los datos que recibiera por serial. Se observo resultados en la hiperterminal de windows xp version 5.1 * Configuracion IP - IP WiPort: 169.254.10.1 - Puerto: 1234 - IP remote host: 169.254.10.2 * Configuracion Serial * Bits por segundo: 115200 - Bits de datos: 8 - Paridad: ninguno - Control de flujo: ninguno * Cantidad caracteres enviados: 486

57,600 bps

Prueba de WiPort con baud rate 57600 - En esta prueba se envio el presente archivo al WiPort - El PIC se programo para hacer eco a todos los datos que recibiera por serial. Se observo resultados en la hiperterminal de windows xp version 5.1 * Configuracion IP - IP WiPort: 169.254.10.1 - Puerto: 1234 - IP remote host: 169.254.10.2 * Configuracion Serial * Bits por segundo: 57600 - Bits de datos: 8 - Paridad: ninguno - Control de flujo: ninguno * Cantidad caracteres enviados: 484

38,400 bps

Prueba de WiPort con baud rate 38400 - En esta prueba se envio el presente archivo al WiPort - El PIC se programo para hacer eco a todos los datos que recibiera por serial. Se observo resultados en la hiperterminal de windows xp version 5.1 * Configuracion IP - IP WiPort: 169.254.10.1 - Puerto: 1234 - IP remote host: 169.254.10.2 * Configuracion Serial * Bits por segundo: 38400 - Bits de datos: 8 - Paridad: ninguno - Control de flujo: ninguno * Cantidad caracteres enviados: 484

Código Fuente PIC

```
 #include     "C:\Users\John\Documents\Megaproyecto\WiPort\Software     PICC\PIC
probando IP\WiPort Get IP.h"
#include <stdio.h>
#include <stdlib.h>
#include <Can4R17.c> //Librería de CAN
#define A1 PIN_A1
#define A0 PIN_A0
#define loc_inicial 0

//El pin A1 define si se solicita IP o no



int8 recep;     //Variable que recibirá los datos
int8 contador;  //Variable externa necesaria para CAN
int8 numDatos=0,numDatostmp=0;
int8 datos[2],datostmp[2], recibido[2];
int i;
int8 conf=0,iprec=0,contIP=0;
int8 ip[15];
int8 ipadd[4];


void get_IP();

#int_RDA
void isr_rda(){   //Rutina de interruptos para la recepción de Serial

   recep=getc();     //Obtiene datos recibidos
   if(conf==3){      //Si no esta en modo de monitoreo
      if(numDatos==0){  //Si es primer dato recibido, lo guarda en pos 0
         datos[0]=recep;
         numDatos++;
      }
      else if(numDatos==1){
         datos[1]=recep;   //Si es el segundo lo guarda en pos 1
         numDatos++;
```

```
      }
      output_toggle(A0);
   }
   else{    //Si esta en modo monitoreo
     if(conf==1){
         if(iprec!=3){
            if(recep==0x49){     //Cuando recibe el indentificador IP
               iprec=1;
               //output_high(A0);
               //putc(recep);
            }
            else if(recep==0x50){
               iprec=2;
               //putc(recep);
            }
            else if(recep==0x20){
               iprec=3;
               //putc(recep);
            }
            if(contIP==15){         //Cuando ya termino de recibir la IP,
               if(recep==0x3E){     //Espera el fin de TX
                  conf=3;           //DIce que ya termino la TX
                  //output_high(A0);
                  //printf("\nFIN\n");
               }
            }
         }
         else if(iprec==3){         //IF para recibir la IP
            ip[contIP]=recep;
            //putc(ip[contIP]);
            contIP++;               //Recibe la IP
            if(contIP==15){
               iprec=4;             //Cuando termina
               //printf("\nFin IP\n");
            }
         }

     }  //FIN IF conf ==1
     else{     //Maneja la espera de info del WiPort
         if(recep==0x3E){
            conf=1;
         }
    }



   }

   //putc(recep);      //Los reenvía, haciendo el ECO, proposito de prueba
   clear_interrupt(int_RDA);  //Limpia la bandera de interrupción Serial

}

void init(){                    //Función para configurar el dispositivo
   enable_interrupts(int_rda);   //Habilita interrupciones del Serial
   enable_interrupts(GLOBAL);    //Habilita las interrupciones globales
   can_start(wifiRx);
   write_eeprom(loc_inicial,192);   //Guarda IP en EEPROM
   write_eeprom(loc_inicial+1,168);
   write_eeprom(loc_inicial+2,1);
   write_eeprom(loc_inicial+3,35);


}
```

```
void main(){

   init();                        //Llama a la función de inicialización

   if(input_state(A1)){
      delay_ms(900);
      get_IP();
   }

   datostmp[0]=106;
   datostmp[1]=0;
   can_enviar(&contador,display,datostmp); //Envia "No erroes al iniciar"


   while(true){                   //Bucle infinito para ejecución

      //output_toggle(A0);

      conf=3;

      if((datos[0]==254)||(datos[1]==254)){ //Secuencia para limpiar buffer
         numDatos=0;                        //de transmisión
         datos[0]=0;
         datos[1]=0;
         output_toggle(A0);
      }

      if(numDatos==2){

         //disable_interrupts(global);  //Apaga interruptos para enviar

         can_enviar(&contador,display,datos); //Envia datos a CAN

         //enable_interrupts(global);    //Reinicia los interruptos

         datos[0]=0;
         datos[1]=0;
         numDatos=0;
         output_toggle(A0);
      }
      if(canRx(wifiRx,&contador,recibido)){ //Procesa recepcion CAN
         //133 es req. IP
         //134 es estado
         if(recibido[0]==134){   //Determina si es comando de Estado
           datostmp[0]=106;
           datostmp[1]=0;
           can_enviar(&contador,display,datostmp);
           datostmp[0]=datostmp[1]=0;
         }
         else if(recibido[0]==133){    //Determina si es comando de IP
            output_high(A0);
            for(i=0;i<4;i++){
               datostmp[0]=107+i;
               datostmp[1]=read_eeprom(loc_inicial+i);
               can_enviar(&contador,display,datostmp);   //Envía IP
               delay_ms(50);
            }
            datostmp[0]=0;
            datostmp[1]=0;
         }
         else{
            for(i=0; i<2;i++){
```

```
               putc(recibido[i]);   //Si es dato normal, los envía al
            }                       //WiPort
         }

      }

   }//FIN WHILE

} //Fin de programa


void get_IP(){
   int k;

   set_uart_speed(9600); //Establece el USART a 9600 para modo Prueba

   conf=2;

   for(k=0;k<3;k++){
      putc(0x7a);       //Envia entrada a modo Conf
   }
   //delay_ms(10);
   //putc(0x0D);          //Fin de linea

   delay_ms(10);


   while(conf==2){      //Espera la recepcion del OK
      delay_ms(400);
      output_toggle(A0);
   }

   putc(0x4E);          //Envía instruccion de NC
   putc(0x43);
   putc(0x0D);

   while(conf==1){      //Espera le recepción de la IP
      output_toggle(A0);
      delay_ms(100);
   }
   for(k=0;k<15;k++){
      ip[k]-=0x30;
   }

   //Se crea el formato de la IP
   ipadd[0]=((ip[0])*100)+(ip[1]*10)+((ip[2])*1);
   ipadd[1]=((ip[4])*100)+(ip[5]*10)+((ip[6])*1);
   ipadd[2]=((ip[8])*100)+(ip[9]*10)+((ip[10])*1);
   ipadd[3]=((ip[12])*100)+(ip[13]*10)+((ip[14])*1);

   disable_interrupts(GLOBAL);
   for(k=0;k<4;k++){
      datos[0]=107+k;
      datos[1]=ipadd[k];
      write_eeprom(loc_inicial+k,ipadd[k]);
      can_enviar(&contador,display,datos);    //Transmite la IP
   }
   delay_ms(5);

   putc(0x51);    //Q
   putc(0x55);    //U
   putc(0x0D);    //Enter

   enable_interrupts(GLOBAL);
```

```
   //printf("Ya");
   output_high(A0);


   conf=3;
   set_uart_speed(57600);  //Setea el USART a 57,600

}
```

A. CÓDIGO FUENTE

a. Clase que se encarga del manejo de guardar y cargar un proyecto

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Xml;

namespace Example
{
    class FileManager
    {
        private bool grabado;
        public FileManager()
        {
            grabado = false;
        }
        public bool SaveProyect(System.Windows.Forms.ListBox toSave, string name)
        {
         grabado = false;
         try
         {
             XmlTextWriter textWriter = new XmlTextWriter(name, null);
             textWriter.WriteStartDocument();
             textWriter.WriteComment("DeucaTalk");
             textWriter.WriteStartElement("DeucaTalk");

             for (int i = 0; i < toSave.Items.Count; i++)
             {
                 textWriter.WriteStartElement("Instrucction");
                 textWriter.WriteAttributeString("Inst",
   toSave.Items[i].ToString());
                 textWriter.WriteEndElement();
             }

             textWriter.WriteEndElement();
             textWriter.Close();
             grabado = true;
         }
         catch { grabado = false; }
         return grabado;
        }
        public bool getGrabado()
        {
```

```
            return grabado;
        }

        public System.Windows.Forms.ListBox toLoad(string name)
        {
            System.Windows.Forms.ListBox           _toLoad           =           new
    System.Windows.Forms.ListBox();
            bool xch = false;
            grabado = false;
            try
            {
                XmlTextReader textReader = new XmlTextReader(name);
                xch = false;

                while (textReader.Read())
                {
                    XmlNodeType nType = textReader.NodeType;
                    if ((nType == XmlNodeType.Element))
                    {
                        if (textReader.Name.ToString() == "DeucaTalk")
                            xch = true;
                        if (textReader.Name.ToString() == "Instrucction")
                        {

    _toLoad.Items.Add(textReader.GetAttribute(0).ToString());
                            xch = true;
                        }
                        if (xch == false)
                        {
                            return null;
                        }
                    }
                    if ((nType == XmlNodeType.EndElement))
                    {
                        if (textReader.Name.ToString() == "DeucaTalk")
                        {
                            xch = true;
                        }
                    }
                }
                if (_toLoad.Items.Count < 1)
                    _toLoad.Items.Add("");
                grabado = true;
            }
            catch
            {
                grabado = false;
                _toLoad.Items.Add("");
            }

            return _toLoad;
        }


    }
}
```

Clase que se encarga del manejo del modo automático

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Collections;
using System.Xml;
```

```
namespace Generador
{
    public class Error_Warning
    {
        private int Error, Warning;
        private string Message;
        public Error_Warning()
        {
            Error = 0;
            Warning = 0;
            Message ="";
        }
        public void init()
        {
            Error = 0;
            Warning = 0;
            Message ="";
        }
        public int getError()
        {
            return Error;
        }
        public int getWarning()
        {
            return Warning;
        }
        public string getMessege()
        {
            return Message;
        }
        public void setError_Warning(int _error, int _warning)
        {
            Error = _error;
            Warning = _warning;
        }
        public void setError(int _error)
        {
            Error = _error;
        }
        public void setWarning(int _warning)
        {
            Warning = _warning;
        }
        public void setMessege(string _messege)
        {
           Message = _messege;
        }
    }


    public class AutomaticMode
    {
        //private ArrayList

        int  pasos_Track,  pasos_Waist,  pasos_Shoulder,  pasos_Elbow,  pasos_Wrist,
   pasos_Hand;
        Error_Warning Watch;


        public AutomaticMode()
        {
```

```
        pasos_Track =0;
        pasos_Waist=0;
        pasos_Shoulder=0;
        pasos_Elbow=0;
        pasos_Wrist=0;
        pasos_Hand = 0;
        Watch = new Error_Warning();
        }

        public  Error_Warning  Convertir(int  _pasos_Track,  int  _pasos_Waist,  int
_pasos_Shoulder, int _pasos_Elbow, int _pasos_Wrist, int _pasos_Hand)
        {
            int _error=0, _warning=0;
            string Msg=" ";
            // Track
            if ((_pasos_Track > 24900)||(_pasos_Track<0))
             _error=32+_error;
            else
            {
              if ((_pasos_Track > 24000)||(_pasos_Track<1000))
                  _warning=32+_warning;
              pasos_Track = _pasos_Track;
            }

            //Wrist
            if ((_pasos_Wrist > 11000) || (_pasos_Wrist < -11000))
             _error=16+_error;
            else
            {
                if ((_pasos_Wrist > 10000) || (_pasos_Wrist < -10000))
                  _warning=16+_warning;
              pasos_Wrist = _pasos_Wrist;
            }

            //Hand
            if ((_pasos_Hand > 3500) || (_pasos_Hand < -3500))
             _error=8+_error;
            else
            {
                if ((_pasos_Hand > 3000) || (_pasos_Hand < -3000))
                  _warning=8+_warning;
              pasos_Hand = _pasos_Hand;
            }

            //Elbow
            if ((_pasos_Elbow > 15000) || (_pasos_Elbow < -15000))
             _error=4+_error;
            else
            {
                if ((_pasos_Elbow > 14000) || (_pasos_Elbow < -14000))
                  _warning=4+_warning;
              pasos_Elbow = _pasos_Elbow;
            }

            //Shoulder
            if ((_pasos_Shoulder > 22500) || (_pasos_Shoulder < -22500))
             _error=2+_error;
            else
            {
                if ((_pasos_Shoulder > 22000) || (_pasos_Shoulder < -22000))
                  _warning=2+_warning;
              pasos_Shoulder = _pasos_Shoulder;

            }
```

```
            //Waist
            if ((_pasos_Waist > 19600) || (_pasos_Waist < -19600))
             _error=1+_error;
            else
            {
                if ((_pasos_Waist > 19000) || (_pasos_Waist < -19000))
                  _warning=1+_warning;
              pasos_Waist = _pasos_Waist;
            }

            Msg=Msg+Convert.ToString(pasos_Track)+" ";
            Msg = Msg + Convert.ToString(pasos_Hand) + " ";
            Msg=Msg+Convert.ToString(pasos_Wrist)+" ";
            Msg=Msg+Convert.ToString(pasos_Elbow)+" ";
            Msg=Msg+Convert.ToString(pasos_Shoulder)+" ";
            Msg=Msg+Convert.ToString(pasos_Waist)+" ";

            Watch.setError_Warning(_error,_warning);
            Watch.setMessege(Msg);
            return Watch;
        }


    }
}
```

Clase que se encarga del manejo del modo automático

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Threading;


namespace Example
{
   /// <summary>
   /// Summary description for Form1.
   /// </summary>
   public class Form1 : System.Windows.Forms.Form
    {
        private bool vRec;
        private bool vTeach;
        private int lines;
        private bool isok;
        private bool isplay;
        private int cont;
        private bool isWaiting;
        private string lastText;
        private Label label1;
        private Label label2;
        private ComboBox ComboDatabits;
        private Label label3;
        private GroupBox groupBox1;
        private ComboBox ComboPort;
        private ComboBox ComboFC;
        private Label label8;
        private ComboBox ComboBaudrate;
        private Label label5;
        private ComboBox ComboParity;
        private Label label4;
```

```
private ComboBox ComboStopbits;
private ListBox ListLog;
private CheckBox checkLog;
private CheckBox checkTerminal;
public ListBox Re;
private TextBox textBox2;
private GroupBox Log;
private PictureBox pictureBox1;
private AxSPortLib.AxSPortAx axSPortAx1;
private Button btnOpen;
private TabControl tcPrincipal;
private TabPage tabPage1;
private Panel cPanel;
private Button btnPlay;
private Button button1;
private Label label17;
private TextBox vWrist;
private TextBox txWrist;
private Label label16;
private TextBox vElbow;
private TextBox txElbow;
private Label label15;
private TextBox vShoulder;
private TextBox txShoulder;
private TextBox vWaist;
private Label label14;
private TextBox txWaist;
private TextBox vTrack;
private Label label13;
private TextBox txTrack;
private Label label12;
private Button btnStart;
private Button btnCalibrate;
private Button btnHome;
private Button btnWristDown;
private Button btnWristUp;
private Button btnElbowDown;
private Button btnElbowUp;
private Button btnShoulderDown;
private Button btnShoulderUp;
private Button btnWaistDown;
private Button btnWaistUp;
private Button btnTrackDown;
private Button btnTrackUp;
private Label label11;
private Label label10;
private Label label9;
private Label label7;
private Label label6;
private TabPage tabPage2;
private TextBox txPosName;
private Button btnGrabarPos;
private Label label21;
private Label label20;
private Label label19;
private Label label18;
private Label label22;
private TextBox vHand;
private TextBox txHand;
private Label label23;
private Button btnHandDown;
private Button btnHandUp;
private Label label24;
private ListBox lbPosiciones;
```

```
    private TextBox txLastText;
    private TextBox Rec;
    private Button button3;
    private TabPage tabPage3;
    private GroupBox gbRemoto;
    private GroupBox gbLocal;
    private Label label26;
    private TextBox txIPRemoto4;
    private TextBox txIPRemoto3;
    private TextBox txIPRemoto1;
    private TextBox txIPRemoto2;
    private Label label25;
    private TextBox txIPLocal4;
    private TextBox txIPLocal3;
    private TextBox txIPLocal2;
    private TextBox txIPLocal1;
    private Label label27;
    private TextBox txPuertoRemoto;
    private Label label28;
    private TextBox txPuertoLocal;
    private IContainer components;
    private Thread WifiThread;
    private Button btnConectarWifi;
    private Button Teach;
    private Button btnGuardar;
    private Button btnAbrir;
    private StatusStrip statusStrip;
    private ToolStripStatusLabel toolStripStatusLabel;
    private System.Windows.Forms.Timer TimertoSend;
    private WiFiConnection WiPort;
    private ArrayList Datos;
    private Generador.AutomaticMode Automat;
    private Generador.Error_Warning Watch ;
    private int track, wrist, shoulder, waist, elbow, hand;

public Form1()
{
  InitializeComponent();
        WifiThread = new Thread(new ThreadStart(WifiLoop));
        WiPort = new WiFiConnection();
}

protected override void Dispose( bool disposing )
{
  if( disposing )
  {
    if (components != null)
    {
      components.Dispose();
                WifiThread.Abort();
    }
  }
  base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
        this.components = new System.ComponentModel.Container();
```

```
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(Form1));
            this.label1 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.ComboDatabits = new System.Windows.Forms.ComboBox();
            this.label3 = new System.Windows.Forms.Label();
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.ComboPort = new System.Windows.Forms.ComboBox();
            this.ComboFC = new System.Windows.Forms.ComboBox();
            this.label8 = new System.Windows.Forms.Label();
            this.ComboBaudrate = new System.Windows.Forms.ComboBox();
            this.label5 = new System.Windows.Forms.Label();
            this.ComboParity = new System.Windows.Forms.ComboBox();
            this.label4 = new System.Windows.Forms.Label();
            this.ComboStopbits = new System.Windows.Forms.ComboBox();
            this.ListLog = new System.Windows.Forms.ListBox();
            this.checkLog = new System.Windows.Forms.CheckBox();
            this.checkTerminal = new System.Windows.Forms.CheckBox();
            this.Re = new System.Windows.Forms.ListBox();
            this.textBox2 = new System.Windows.Forms.TextBox();
            this.Log = new System.Windows.Forms.GroupBox();
            this.Rec = new System.Windows.Forms.TextBox();
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.btnOpen = new System.Windows.Forms.Button();
            this.tcPrincipal = new System.Windows.Forms.TabControl();
            this.tabPage1 = new System.Windows.Forms.TabPage();
            this.cPanel = new System.Windows.Forms.Panel();
            this.button3 = new System.Windows.Forms.Button();
            this.label22 = new System.Windows.Forms.Label();
            this.vHand = new System.Windows.Forms.TextBox();
            this.txHand = new System.Windows.Forms.TextBox();
            this.label23 = new System.Windows.Forms.Label();
            this.btnHandDown = new System.Windows.Forms.Button();
            this.btnHandUp = new System.Windows.Forms.Button();
            this.label24 = new System.Windows.Forms.Label();
            this.label21 = new System.Windows.Forms.Label();
            this.label20 = new System.Windows.Forms.Label();
            this.label19 = new System.Windows.Forms.Label();
            this.label18 = new System.Windows.Forms.Label();
            this.btnPlay = new System.Windows.Forms.Button();
            this.button1 = new System.Windows.Forms.Button();
            this.label17 = new System.Windows.Forms.Label();
            this.vWrist = new System.Windows.Forms.TextBox();
            this.txWrist = new System.Windows.Forms.TextBox();
            this.label16 = new System.Windows.Forms.Label();
            this.vElbow = new System.Windows.Forms.TextBox();
            this.txElbow = new System.Windows.Forms.TextBox();
            this.label15 = new System.Windows.Forms.Label();
            this.vShoulder = new System.Windows.Forms.TextBox();
            this.txShoulder = new System.Windows.Forms.TextBox();
            this.vWaist = new System.Windows.Forms.TextBox();
            this.label14 = new System.Windows.Forms.Label();
            this.txWaist = new System.Windows.Forms.TextBox();
            this.vTrack = new System.Windows.Forms.TextBox();
            this.label13 = new System.Windows.Forms.Label();
            this.txTrack = new System.Windows.Forms.TextBox();
            this.label12 = new System.Windows.Forms.Label();
            this.btnStart = new System.Windows.Forms.Button();
            this.btnCalibrate = new System.Windows.Forms.Button();
            this.btnHome = new System.Windows.Forms.Button();
            this.btnWristDown = new System.Windows.Forms.Button();
            this.btnWristUp = new System.Windows.Forms.Button();
            this.btnElbowDown = new System.Windows.Forms.Button();
            this.btnElbowUp = new System.Windows.Forms.Button();
```

```
            this.btnShoulderDown = new System.Windows.Forms.Button();
            this.btnShoulderUp = new System.Windows.Forms.Button();
            this.btnWaistDown = new System.Windows.Forms.Button();
            this.btnWaistUp = new System.Windows.Forms.Button();
            this.btnTrackDown = new System.Windows.Forms.Button();
            this.btnTrackUp = new System.Windows.Forms.Button();
            this.label11 = new System.Windows.Forms.Label();
            this.label10 = new System.Windows.Forms.Label();
            this.label9 = new System.Windows.Forms.Label();
            this.label7 = new System.Windows.Forms.Label();
            this.label6 = new System.Windows.Forms.Label();
            this.tabPage2 = new System.Windows.Forms.TabPage();
            this.lbPosiciones = new System.Windows.Forms.ListBox();
            this.txPosName = new System.Windows.Forms.TextBox();
            this.btnGrabarPos = new System.Windows.Forms.Button();
            this.tabPage3 = new System.Windows.Forms.TabPage();
            this.btnConectarWifi = new System.Windows.Forms.Button();
            this.gbRemoto = new System.Windows.Forms.GroupBox();
            this.label27 = new System.Windows.Forms.Label();
            this.txPuertoRemoto = new System.Windows.Forms.TextBox();
            this.label26 = new System.Windows.Forms.Label();
            this.txIPRemoto4 = new System.Windows.Forms.TextBox();
            this.txIPRemoto3 = new System.Windows.Forms.TextBox();
            this.txIPRemoto1 = new System.Windows.Forms.TextBox();
            this.txIPRemoto2 = new System.Windows.Forms.TextBox();
            this.gbLocal = new System.Windows.Forms.GroupBox();
            this.label28 = new System.Windows.Forms.Label();
            this.txPuertoLocal = new System.Windows.Forms.TextBox();
            this.label25 = new System.Windows.Forms.Label();
            this.txIPLocal4 = new System.Windows.Forms.TextBox();
            this.txIPLocal3 = new System.Windows.Forms.TextBox();
            this.txIPLocal2 = new System.Windows.Forms.TextBox();
            this.txIPLocal1 = new System.Windows.Forms.TextBox();
            this.txLastText = new System.Windows.Forms.TextBox();
            this.axSPortAx1 = new AxSPortLib.AxSPortAx();
            this.Teach = new System.Windows.Forms.Button();
            this.btnGuardar = new System.Windows.Forms.Button();
            this.btnAbrir = new System.Windows.Forms.Button();
            this.statusStrip = new System.Windows.Forms.StatusStrip();
            this.toolStripStatusLabel                    =                    new
System.Windows.Forms.ToolStripStatusLabel();
            this.TimertoSend = new System.Windows.Forms.Timer(this.components);
            this.groupBox1.SuspendLayout();
            this.Log.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
            this.tcPrincipal.SuspendLayout();
            this.tabPage1.SuspendLayout();
            this.cPanel.SuspendLayout();
            this.tabPage2.SuspendLayout();
            this.tabPage3.SuspendLayout();
            this.gbRemoto.SuspendLayout();
            this.gbLocal.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.axSPortAx1)).BeginInit();
            this.statusStrip.SuspendLayout();
            this.SuspendLayout();
            //
            // label1
            //
            this.label1.Location = new System.Drawing.Point(27, 21);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(48, 16);
            this.label1.TabIndex = 0;
```

```
            this.label1.Text = "Port:";
            //
            // label2
            //
            this.label2.Location = new System.Drawing.Point(9, 51);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(64, 16);
            this.label2.TabIndex = 2;
            this.label2.Text = "Databits:";
            //
            // ComboDatabits
            //
            this.ComboDatabits.DropDownStyle                                        =
System.Windows.Forms.ComboBoxStyle.DropDownList;
            this.ComboDatabits.Items.AddRange(new object[] {
            "5",
            "6",
            "7",
            "8"});
            this.ComboDatabits.Location = new System.Drawing.Point(58, 48);
            this.ComboDatabits.Name = "ComboDatabits";
            this.ComboDatabits.Size = new System.Drawing.Size(96, 21);
            this.ComboDatabits.TabIndex = 3;
            this.ComboDatabits.SelectedIndexChanged                 +=              new
System.EventHandler(this.ComboDatabits_SelectedIndexChanged);
            //
            // label3
            //
            this.label3.Location = new System.Drawing.Point(167, 51);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(72, 16);
            this.label3.TabIndex = 4;
            this.label3.Text = "StopBits:";
            //
            // groupBox1
            //
            this.groupBox1.Controls.Add(this.ComboPort);
            this.groupBox1.Controls.Add(this.ComboFC);
            this.groupBox1.Controls.Add(this.label8);
            this.groupBox1.Controls.Add(this.ComboBaudrate);
            this.groupBox1.Controls.Add(this.label5);
            this.groupBox1.Controls.Add(this.ComboParity);
            this.groupBox1.Controls.Add(this.label4);
            this.groupBox1.Controls.Add(this.ComboStopbits);
            this.groupBox1.Controls.Add(this.label3);
            this.groupBox1.Controls.Add(this.ComboDatabits);
            this.groupBox1.Controls.Add(this.label2);
            this.groupBox1.Controls.Add(this.label1);
            this.groupBox1.Location = new System.Drawing.Point(20, 16);
            this.groupBox1.Name = "groupBox1";
            this.groupBox1.Size = new System.Drawing.Size(556, 71);
            this.groupBox1.TabIndex = 0;
            this.groupBox1.TabStop = false;
            this.groupBox1.Text = "Options";
            //
            // ComboPort
            //
            this.ComboPort.DropDownStyle                                            =
System.Windows.Forms.ComboBoxStyle.DropDownList;
            this.ComboPort.Location = new System.Drawing.Point(58, 16);
            this.ComboPort.Name = "ComboPort";
            this.ComboPort.Size = new System.Drawing.Size(96, 21);
            this.ComboPort.TabIndex = 1;
            //
```

```
            // ComboFC
            //
            this.ComboFC.DropDownStyle                                          =
System.Windows.Forms.ComboBoxStyle.DropDownList;
            this.ComboFC.Items.AddRange(new object[] {
            "Xon\\Xoff",
            "Hardware",
            "None"});
            this.ComboFC.Location = new System.Drawing.Point(394, 48);
            this.ComboFC.Name = "ComboFC";
            this.ComboFC.Size = new System.Drawing.Size(96, 21);
            this.ComboFC.TabIndex = 12;
            this.ComboFC.SelectedIndexChanged                  +=                  new
System.EventHandler(this.ComboFC_SelectedIndexChanged);
            //
            // label8
            //
            this.label8.Location = new System.Drawing.Point(323, 50);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(80, 16);
            this.label8.TabIndex = 11;
            this.label8.Text = "Flow control :";
            //
            // ComboBaudrate
            //
            this.ComboBaudrate.DropDownStyle                                    =
System.Windows.Forms.ComboBoxStyle.DropDownList;
            this.ComboBaudrate.Items.AddRange(new object[] {
            "110",
            "300",
            "600",
            "1200",
            "2400",
            "4800",
            "9600",
            "14400",
            "19200",
            "38400",
            "56000",
            "57600",
            "115200",
            "128000",
            "256000"});
            this.ComboBaudrate.Location = new System.Drawing.Point(223, 16);
            this.ComboBaudrate.Name = "ComboBaudrate";
            this.ComboBaudrate.Size = new System.Drawing.Size(96, 21);
            this.ComboBaudrate.TabIndex = 9;
            this.ComboBaudrate.SelectedIndexChanged              +=              new
System.EventHandler(this.ComboBaudrate_SelectedIndexChanged);
            //
            // label5
            //
            this.label5.Location = new System.Drawing.Point(167, 21);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(72, 16);
            this.label5.TabIndex = 8;
            this.label5.Text = "Baudrate:";
            //
            // ComboParity
            //
            this.ComboParity.DropDownStyle                                      =
System.Windows.Forms.ComboBoxStyle.DropDownList;
            this.ComboParity.Items.AddRange(new object[] {
            "No parity",
```

```
            "Odd",
            "Even",
            "Mark",
            "Space"});
            this.ComboParity.Location = new System.Drawing.Point(394, 16);
            this.ComboParity.Name = "ComboParity";
            this.ComboParity.Size = new System.Drawing.Size(96, 21);
            this.ComboParity.TabIndex = 7;
            this.ComboParity.SelectedIndexChanged               +=               new
System.EventHandler(this.ComboParity_SelectedIndexChanged);
            //
            // label4
            //
            this.label4.Location = new System.Drawing.Point(354, 20);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(48, 16);
            this.label4.TabIndex = 6;
            this.label4.Text = "Parity:";
            //
            // ComboStopbits
            //
            this.ComboStopbits.DropDownStyle                                        =
System.Windows.Forms.ComboBoxStyle.DropDownList;
            this.ComboStopbits.Items.AddRange(new object[] {
            "1 stop",
            "1,5 stop",
            "2 stop"});
            this.ComboStopbits.Location = new System.Drawing.Point(222, 48);
            this.ComboStopbits.Name = "ComboStopbits";
            this.ComboStopbits.Size = new System.Drawing.Size(96, 21);
            this.ComboStopbits.TabIndex = 5;
            this.ComboStopbits.SelectedIndexChanged              +=              new
System.EventHandler(this.ComboStopbits_SelectedIndexChanged);
            //
            // ListLog
            //
            this.ListLog.Location = new System.Drawing.Point(8, 16);
            this.ListLog.Name = "ListLog";
            this.ListLog.Size = new System.Drawing.Size(351, 69);
            this.ListLog.TabIndex = 1;
            //
            // checkLog
            //
            this.checkLog.Location = new System.Drawing.Point(11, 92);
            this.checkLog.Name = "checkLog";
            this.checkLog.Size = new System.Drawing.Size(104, 24);
            this.checkLog.TabIndex = 5;
            this.checkLog.Text = "Deshabilitar";
            //
            // checkTerminal
            //
            this.checkTerminal.Location = new System.Drawing.Point(361, 90);
            this.checkTerminal.Name = "checkTerminal";
            this.checkTerminal.Size = new System.Drawing.Size(240, 24);
            this.checkTerminal.TabIndex = 6;
            this.checkTerminal.Text = "No mostrar datos entrantes";
            //
            // Re
            //
            this.Re.Enabled = false;
            this.Re.Location = new System.Drawing.Point(611, 17);
            this.Re.Name = "Re";
            this.Re.Size = new System.Drawing.Size(51, 69);
            this.Re.TabIndex = 2;
```

```
            this.Re.Visible = false;
            this.Re.SelectedIndexChanged                    +=                    new
System.EventHandler(this.Re_SelectedIndexChanged);
            //
            // textBox2
            //
            this.textBox2.Enabled = false;
            this.textBox2.Location = new System.Drawing.Point(366, 17);
            this.textBox2.Multiline = true;
            this.textBox2.Name = "textBox2";
            this.textBox2.Size = new System.Drawing.Size(297, 69);
            this.textBox2.TabIndex = 2;
            this.textBox2.KeyPress                      +=                    new
System.Windows.Forms.KeyPressEventHandler(this.textBox2_KeyPress);
            //
            // Log
            //
            this.Log.Controls.Add(this.textBox2);
            this.Log.Controls.Add(this.Rec);
            this.Log.Controls.Add(this.Re);
            this.Log.Controls.Add(this.checkTerminal);
            this.Log.Controls.Add(this.checkLog);
            this.Log.Controls.Add(this.ListLog);
            this.Log.Location = new System.Drawing.Point(8, 307);
            this.Log.Name = "Log";
            this.Log.Size = new System.Drawing.Size(672, 122);
            this.Log.TabIndex = 1;
            this.Log.TabStop = false;
            this.Log.Text = "Log";
            this.Log.Enter += new System.EventHandler(this.Log_Enter);
            //
            // Rec
            //
            this.Rec.Enabled = false;
            this.Rec.Location = new System.Drawing.Point(550, 17);
            this.Rec.Multiline = true;
            this.Rec.Name = "Rec";
            this.Rec.Size = new System.Drawing.Size(55, 69);
            this.Rec.TabIndex = 7;
            this.Rec.WordWrap = false;
            //
            // pictureBox1
            //
            this.pictureBox1.BackgroundImage                                                =
((System.Drawing.Image)(resources.GetObject("pictureBox1.BackgroundImage")));
            this.pictureBox1.Location = new System.Drawing.Point(14, 9);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(666, 82);
            this.pictureBox1.TabIndex = 2;
            this.pictureBox1.TabStop = false;
            //
            // btnOpen
            //
            this.btnOpen.Location = new System.Drawing.Point(446, 434);
            this.btnOpen.Name = "btnOpen";
            this.btnOpen.Size = new System.Drawing.Size(106, 27);
            this.btnOpen.TabIndex = 11;
            this.btnOpen.Text = "Abrir Puerto";
            this.btnOpen.Click += new System.EventHandler(this.btnOpen_Click);
            //
            // tcPrincipal
            //
            this.tcPrincipal.Controls.Add(this.tabPage1);
            this.tcPrincipal.Controls.Add(this.tabPage2);
```

```
this.tcPrincipal.Controls.Add(this.tabPage3);
this.tcPrincipal.HotTrack = true;
this.tcPrincipal.Location = new System.Drawing.Point(15, 98);
this.tcPrincipal.Name = "tcPrincipal";
this.tcPrincipal.SelectedIndex = 0;
this.tcPrincipal.Size = new System.Drawing.Size(665, 209);
this.tcPrincipal.TabIndex = 12;
//
// tabPage1
//
this.tabPage1.Controls.Add(this.cPanel);
this.tabPage1.Location = new System.Drawing.Point(4, 22);
this.tabPage1.Name = "tabPage1";
this.tabPage1.Padding = new System.Windows.Forms.Padding(3);
this.tabPage1.Size = new System.Drawing.Size(657, 183);
this.tabPage1.TabIndex = 0;
this.tabPage1.Text = "Articulaciones";
this.tabPage1.UseVisualStyleBackColor = true;
//
// cPanel
//
this.cPanel.Controls.Add(this.button3);
this.cPanel.Controls.Add(this.label22);
this.cPanel.Controls.Add(this.vHand);
this.cPanel.Controls.Add(this.txHand);
this.cPanel.Controls.Add(this.label23);
this.cPanel.Controls.Add(this.btnHandDown);
this.cPanel.Controls.Add(this.btnHandUp);
this.cPanel.Controls.Add(this.label24);
this.cPanel.Controls.Add(this.label21);
this.cPanel.Controls.Add(this.label20);
this.cPanel.Controls.Add(this.label19);
this.cPanel.Controls.Add(this.label18);
this.cPanel.Controls.Add(this.btnPlay);
this.cPanel.Controls.Add(this.button1);
this.cPanel.Controls.Add(this.label17);
this.cPanel.Controls.Add(this.vWrist);
this.cPanel.Controls.Add(this.txWrist);
this.cPanel.Controls.Add(this.label16);
this.cPanel.Controls.Add(this.vElbow);
this.cPanel.Controls.Add(this.txElbow);
this.cPanel.Controls.Add(this.label15);
this.cPanel.Controls.Add(this.vShoulder);
this.cPanel.Controls.Add(this.txShoulder);
this.cPanel.Controls.Add(this.vWaist);
this.cPanel.Controls.Add(this.label14);
this.cPanel.Controls.Add(this.txWaist);
this.cPanel.Controls.Add(this.vTrack);
this.cPanel.Controls.Add(this.label13);
this.cPanel.Controls.Add(this.txTrack);
this.cPanel.Controls.Add(this.label12);
this.cPanel.Controls.Add(this.btnStart);
this.cPanel.Controls.Add(this.btnCalibrate);
this.cPanel.Controls.Add(this.btnHome);
this.cPanel.Controls.Add(this.btnWristDown);
this.cPanel.Controls.Add(this.btnWristUp);
this.cPanel.Controls.Add(this.btnElbowDown);
this.cPanel.Controls.Add(this.btnElbowUp);
this.cPanel.Controls.Add(this.btnShoulderDown);
this.cPanel.Controls.Add(this.btnShoulderUp);
this.cPanel.Controls.Add(this.btnWaistDown);
this.cPanel.Controls.Add(this.btnWaistUp);
this.cPanel.Controls.Add(this.btnTrackDown);
this.cPanel.Controls.Add(this.btnTrackUp);
```

```
            this.cPanel.Controls.Add(this.label11);
            this.cPanel.Controls.Add(this.label10);
            this.cPanel.Controls.Add(this.label9);
            this.cPanel.Controls.Add(this.label7);
            this.cPanel.Controls.Add(this.label6);
            this.cPanel.Location = new System.Drawing.Point(-12, -11);
            this.cPanel.Margin = new System.Windows.Forms.Padding(0);
            this.cPanel.Name = "cPanel";
            this.cPanel.Size = new System.Drawing.Size(673, 205);
            this.cPanel.TabIndex = 18;
            //
            // button3
            //
            this.button3.Location = new System.Drawing.Point(185, 160);
            this.button3.Name = "button3";
            this.button3.Size = new System.Drawing.Size(75, 27);
            this.button3.TabIndex = 59;
            this.button3.Text = "Rutina";
            this.button3.UseVisualStyleBackColor = true;
            this.button3.Click += new System.EventHandler(this.button3_Click);
            //
            // label22
            //
            this.label22.AutoSize = true;
            this.label22.Location = new System.Drawing.Point(572, 91);
            this.label22.Name = "label22";
            this.label22.Size = new System.Drawing.Size(40, 13);
            this.label22.TabIndex = 58;
            this.label22.Text = "Actual:";
            //
            // vHand
            //
            this.vHand.Enabled = false;
            this.vHand.Location = new System.Drawing.Point(612, 87);
            this.vHand.Name = "vHand";
            this.vHand.Size = new System.Drawing.Size(51, 20);
            this.vHand.TabIndex = 55;
            this.vHand.Text = "0";
            this.vHand.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            //
            // txHand
            //
            this.txHand.Location = new System.Drawing.Point(610, 61);
            this.txHand.Name = "txHand";
            this.txHand.Size = new System.Drawing.Size(53, 20);
            this.txHand.TabIndex = 56;
            this.txHand.Text = "1000";
            this.txHand.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            //
            // label23
            //
            this.label23.AutoSize = true;
            this.label23.Location = new System.Drawing.Point(573, 64);
            this.label23.Name = "label23";
            this.label23.Size = new System.Drawing.Size(39, 13);
            this.label23.TabIndex = 57;
            this.label23.Text = "Pasos:";
            //
            // btnHandDown
            //
            this.btnHandDown.Cursor = System.Windows.Forms.Cursors.Hand;
            this.btnHandDown.Font = new System.Drawing.Font("Microsoft Sans Serif",
10F,       System.Drawing.FontStyle.Bold,       System.Drawing.GraphicsUnit.Point,
((byte)(0)));
```

```
            this.btnHandDown.Location = new System.Drawing.Point(621, 36);
            this.btnHandDown.Margin = new System.Windows.Forms.Padding(0);
            this.btnHandDown.Name = "btnHandDown";
            this.btnHandDown.Size = new System.Drawing.Size(44, 25);
            this.btnHandDown.TabIndex = 54;
            this.btnHandDown.Text = "-";
            this.btnHandDown.UseVisualStyleBackColor = true;
            this.btnHandDown.Click                      +=                      new
System.EventHandler(this.btnHandDown_Click);
            //
            // btnHandUp
            //
            this.btnHandUp.Cursor = System.Windows.Forms.Cursors.Hand;
            this.btnHandUp.Font  =  new  System.Drawing.Font("Microsoft  Sans  Serif",
10F,       System.Drawing.FontStyle.Bold,       System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.btnHandUp.Location = new System.Drawing.Point(575, 36);
            this.btnHandUp.Margin = new System.Windows.Forms.Padding(0);
            this.btnHandUp.Name = "btnHandUp";
            this.btnHandUp.Size = new System.Drawing.Size(44, 25);
            this.btnHandUp.TabIndex = 53;
            this.btnHandUp.Text = "+";
            this.btnHandUp.UseVisualStyleBackColor = true;
            this.btnHandUp.Click += new System.EventHandler(this.btnHandUp_Click);
            //
            // label24
            //
            this.label24.AutoSize = true;
            this.label24.Font  =  new  System.Drawing.Font("Microsoft  Sans  Serif",
12F,       System.Drawing.FontStyle.Bold,       System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.label24.Location = new System.Drawing.Point(592, 16);
            this.label24.Name = "label24";
            this.label24.Size = new System.Drawing.Size(59, 20);
            this.label24.TabIndex = 52;
            this.label24.Text = "HAND";
            //
            // label21
            //
            this.label21.AutoSize = true;
            this.label21.Location = new System.Drawing.Point(467, 91);
            this.label21.Name = "label21";
            this.label21.Size = new System.Drawing.Size(40, 13);
            this.label21.TabIndex = 51;
            this.label21.Text = "Actual:";
            //
            // label20
            //
            this.label20.AutoSize = true;
            this.label20.Location = new System.Drawing.Point(355, 91);
            this.label20.Name = "label20";
            this.label20.Size = new System.Drawing.Size(40, 13);
            this.label20.TabIndex = 50;
            this.label20.Text = "Actual:";
            //
            // label19
            //
            this.label19.AutoSize = true;
            this.label19.Location = new System.Drawing.Point(239, 90);
            this.label19.Name = "label19";
            this.label19.Size = new System.Drawing.Size(40, 13);
            this.label19.TabIndex = 49;
            this.label19.Text = "Actual:";
            //
```

```
            // label18
            //
            this.label18.AutoSize = true;
            this.label18.Location = new System.Drawing.Point(123, 90);
            this.label18.Name = "label18";
            this.label18.Size = new System.Drawing.Size(40, 13);
            this.label18.TabIndex = 48;
            this.label18.Text = "Actual:";
            //
            // btnPlay
            //
            this.btnPlay.Location = new System.Drawing.Point(270, 161);
            this.btnPlay.Name = "btnPlay";
            this.btnPlay.Size = new System.Drawing.Size(75, 27);
            this.btnPlay.TabIndex = 47;
            this.btnPlay.Text = "Play";
            this.btnPlay.UseVisualStyleBackColor = true;
            this.btnPlay.Click += new System.EventHandler(this.button2_Click);
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(347, 161);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(75, 27);
            this.button1.TabIndex = 46;
            this.button1.Text = "Record On";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // label17
            //
            this.label17.AutoSize = true;
            this.label17.Location = new System.Drawing.Point(17, 91);
            this.label17.Name = "label17";
            this.label17.Size = new System.Drawing.Size(40, 13);
            this.label17.TabIndex = 45;
            this.label17.Text = "Actual:";
            //
            // vWrist
            //
            this.vWrist.Enabled = false;
            this.vWrist.Location = new System.Drawing.Point(507, 87);
            this.vWrist.Name = "vWrist";
            this.vWrist.Size = new System.Drawing.Size(51, 20);
            this.vWrist.TabIndex = 43;
            this.vWrist.Text = "0";
            this.vWrist.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            //
            // txWrist
            //
            this.txWrist.Location = new System.Drawing.Point(505, 61);
            this.txWrist.Name = "txWrist";
            this.txWrist.Size = new System.Drawing.Size(53, 20);
            this.txWrist.TabIndex = 43;
            this.txWrist.Text = "1000";
            this.txWrist.TextAlign                                                  =
System.Windows.Forms.HorizontalAlignment.Right;
            //
            // label16
            //
            this.label16.AutoSize = true;
            this.label16.Location = new System.Drawing.Point(468, 64);
            this.label16.Name = "label16";
            this.label16.Size = new System.Drawing.Size(39, 13);
```

```
            this.label16.TabIndex = 44;
            this.label16.Text = "Pasos:";
            //
            // vElbow
            //
            this.vElbow.Enabled = false;
            this.vElbow.Location = new System.Drawing.Point(401, 87);
            this.vElbow.Name = "vElbow";
            this.vElbow.Size = new System.Drawing.Size(48, 20);
            this.vElbow.TabIndex = 41;
            this.vElbow.Text = "0";
            this.vElbow.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            //
            // txElbow
            //
            this.txElbow.Location = new System.Drawing.Point(394, 61);
            this.txElbow.Name = "txElbow";
            this.txElbow.Size = new System.Drawing.Size(55, 20);
            this.txElbow.TabIndex = 41;
            this.txElbow.Text = "1000";
            this.txElbow.TextAlign                                                  =
System.Windows.Forms.HorizontalAlignment.Right;
            //
            // label15
            //
            this.label15.AutoSize = true;
            this.label15.Location = new System.Drawing.Point(357, 64);
            this.label15.Name = "label15";
            this.label15.Size = new System.Drawing.Size(39, 13);
            this.label15.TabIndex = 42;
            this.label15.Text = "Pasos:";
            //
            // vShoulder
            //
            this.vShoulder.Enabled = false;
            this.vShoulder.Location = new System.Drawing.Point(285, 87);
            this.vShoulder.Name = "vShoulder";
            this.vShoulder.Size = new System.Drawing.Size(48, 20);
            this.vShoulder.TabIndex = 39;
            this.vShoulder.Text = "0";
            this.vShoulder.TextAlign                                                =
System.Windows.Forms.HorizontalAlignment.Right;
            //
            // txShoulder
            //
            this.txShoulder.Location = new System.Drawing.Point(277, 61);
            this.txShoulder.Name = "txShoulder";
            this.txShoulder.Size = new System.Drawing.Size(55, 20);
            this.txShoulder.TabIndex = 39;
            this.txShoulder.Text = "1000";
            this.txShoulder.TextAlign                                               =
System.Windows.Forms.HorizontalAlignment.Right;
            //
            // vWaist
            //
            this.vWaist.Enabled = false;
            this.vWaist.Location = new System.Drawing.Point(169, 88);
            this.vWaist.Name = "vWaist";
            this.vWaist.Size = new System.Drawing.Size(48, 20);
            this.vWaist.TabIndex = 37;
            this.vWaist.Text = "0";
            this.vWaist.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            //
            // label14
```

```
            //
            this.label14.AutoSize = true;
            this.label14.Location = new System.Drawing.Point(240, 64);
            this.label14.Name = "label14";
            this.label14.Size = new System.Drawing.Size(39, 13);
            this.label14.TabIndex = 40;
            this.label14.Text = "Pasos:";
            //
            // txWaist
            //
            this.txWaist.Location = new System.Drawing.Point(162, 62);
            this.txWaist.Name = "txWaist";
            this.txWaist.Size = new System.Drawing.Size(55, 20);
            this.txWaist.TabIndex = 37;
            this.txWaist.Text = "1000";
            this.txWaist.TextAlign                                                  =
System.Windows.Forms.HorizontalAlignment.Right;
            //
            // vTrack
            //
            this.vTrack.Enabled = false;
            this.vTrack.Location = new System.Drawing.Point(62, 87);
            this.vTrack.Name = "vTrack";
            this.vTrack.Size = new System.Drawing.Size(49, 20);
            this.vTrack.TabIndex = 35;
            this.vTrack.Text = "0";
            this.vTrack.TextAlign = System.Windows.Forms.HorizontalAlignment.Right;
            //
            // label13
            //
            this.label13.AutoSize = true;
            this.label13.Location = new System.Drawing.Point(125, 65);
            this.label13.Name = "label13";
            this.label13.Size = new System.Drawing.Size(39, 13);
            this.label13.TabIndex = 38;
            this.label13.Text = "Pasos:";
            //
            // txTrack
            //
            this.txTrack.Location = new System.Drawing.Point(56, 61);
            this.txTrack.Name = "txTrack";
            this.txTrack.Size = new System.Drawing.Size(55, 20);
            this.txTrack.TabIndex = 35;
            this.txTrack.Text = "1000";
            this.txTrack.TextAlign                                                  =
System.Windows.Forms.HorizontalAlignment.Right;
            //
            // label12
            //
            this.label12.AutoSize = true;
            this.label12.Location = new System.Drawing.Point(19, 64);
            this.label12.Name = "label12";
            this.label12.Size = new System.Drawing.Size(39, 13);
            this.label12.TabIndex = 36;
            this.label12.Text = "Pasos:";
            //
            // btnStart
            //
            this.btnStart.Location = new System.Drawing.Point(428, 161);
            this.btnStart.Name = "btnStart";
            this.btnStart.Size = new System.Drawing.Size(75, 27);
            this.btnStart.TabIndex = 34;
            this.btnStart.Text = "Start";
            this.btnStart.UseVisualStyleBackColor = true;
```

```
            this.btnStart.Click += new System.EventHandler(this.btnStart_Click);
            //
            // btnCalibrate
            //
            this.btnCalibrate.Location = new System.Drawing.Point(508, 161);
            this.btnCalibrate.Name = "btnCalibrate";
            this.btnCalibrate.Size = new System.Drawing.Size(75, 27);
            this.btnCalibrate.TabIndex = 33;
            this.btnCalibrate.Text = "Calibrate";
            this.btnCalibrate.UseVisualStyleBackColor = true;
            this.btnCalibrate.Click                      +=                      new
System.EventHandler(this.btnCalibrate_Click);
            //
            // btnHome
            //
            this.btnHome.Location = new System.Drawing.Point(589, 161);
            this.btnHome.Name = "btnHome";
            this.btnHome.Size = new System.Drawing.Size(75, 27);
            this.btnHome.TabIndex = 32;
            this.btnHome.Text = "Home";
            this.btnHome.UseVisualStyleBackColor = true;
            this.btnHome.Click += new System.EventHandler(this.btnHome_Click);
            //
            // btnWristDown
            //
            this.btnWristDown.Cursor = System.Windows.Forms.Cursors.Hand;
            this.btnWristDown.Font   =   new   System.Drawing.Font("Microsoft   Sans
Serif",  10F,  System.Drawing.FontStyle.Bold,  System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.btnWristDown.Location = new System.Drawing.Point(516, 36);
            this.btnWristDown.Margin = new System.Windows.Forms.Padding(0);
            this.btnWristDown.Name = "btnWristDown";
            this.btnWristDown.Size = new System.Drawing.Size(44, 25);
            this.btnWristDown.TabIndex = 31;
            this.btnWristDown.Text = "-";
            this.btnWristDown.UseVisualStyleBackColor = true;
            this.btnWristDown.Click                      +=                      new
System.EventHandler(this.btnWristDown_Click);
            //
            // btnWristUp
            //
            this.btnWristUp.Cursor = System.Windows.Forms.Cursors.Hand;
            this.btnWristUp.Font  =  new  System.Drawing.Font("Microsoft  Sans  Serif",
10F,      System.Drawing.FontStyle.Bold,      System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.btnWristUp.Location = new System.Drawing.Point(470, 36);
            this.btnWristUp.Margin = new System.Windows.Forms.Padding(0);
            this.btnWristUp.Name = "btnWristUp";
            this.btnWristUp.Size = new System.Drawing.Size(44, 25);
            this.btnWristUp.TabIndex = 30;
            this.btnWristUp.Text = "+";
            this.btnWristUp.UseVisualStyleBackColor = true;
            this.btnWristUp.Click                      +=                      new
System.EventHandler(this.btnWristUp_Click);
            //
            // btnElbowDown
            //
            this.btnElbowDown.Cursor = System.Windows.Forms.Cursors.Hand;
            this.btnElbowDown.Font   =   new   System.Drawing.Font("Microsoft   Sans
Serif",  10F,  System.Drawing.FontStyle.Bold,  System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.btnElbowDown.Location = new System.Drawing.Point(405, 36);
            this.btnElbowDown.Margin = new System.Windows.Forms.Padding(0);
            this.btnElbowDown.Name = "btnElbowDown";
```

```
            this.btnElbowDown.Size = new System.Drawing.Size(44, 25);
            this.btnElbowDown.TabIndex = 29;
            this.btnElbowDown.Text = "-";
            this.btnElbowDown.UseVisualStyleBackColor = true;
            this.btnElbowDown.Click                     +=                     new
System.EventHandler(this.btnElbowDown_Click);
            //
            // btnElbowUp
            //
            this.btnElbowUp.Cursor = System.Windows.Forms.Cursors.Hand;
            this.btnElbowUp.Font  =  new  System.Drawing.Font("Microsoft  Sans  Serif",
10F,       System.Drawing.FontStyle.Bold,       System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.btnElbowUp.Location = new System.Drawing.Point(358, 36);
            this.btnElbowUp.Margin = new System.Windows.Forms.Padding(0);
            this.btnElbowUp.Name = "btnElbowUp";
            this.btnElbowUp.Size = new System.Drawing.Size(44, 25);
            this.btnElbowUp.TabIndex = 28;
            this.btnElbowUp.Text = "+";
            this.btnElbowUp.UseVisualStyleBackColor = true;
            this.btnElbowUp.Click                     +=                     new
System.EventHandler(this.btnElbowUp_Click);
            //
            // btnShoulderDown
            //
            this.btnShoulderDown.Cursor = System.Windows.Forms.Cursors.Hand;
            this.btnShoulderDown.Font  =  new  System.Drawing.Font("Microsoft  Sans
Serif",  10F,  System.Drawing.FontStyle.Bold,  System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.btnShoulderDown.Location = new System.Drawing.Point(288, 36);
            this.btnShoulderDown.Margin = new System.Windows.Forms.Padding(0);
            this.btnShoulderDown.Name = "btnShoulderDown";
            this.btnShoulderDown.Size = new System.Drawing.Size(44, 25);
            this.btnShoulderDown.TabIndex = 27;
            this.btnShoulderDown.Text = "-";
            this.btnShoulderDown.UseVisualStyleBackColor = true;
            this.btnShoulderDown.Click                     +=                     new
System.EventHandler(this.btnShoulderDown_Click);
            //
            // btnShoulderUp
            //
            this.btnShoulderUp.Cursor = System.Windows.Forms.Cursors.Hand;
            this.btnShoulderUp.Font   =   new   System.Drawing.Font("Microsoft   Sans
Serif",  10F,  System.Drawing.FontStyle.Bold,  System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.btnShoulderUp.Location = new System.Drawing.Point(242, 36);
            this.btnShoulderUp.Margin = new System.Windows.Forms.Padding(0);
            this.btnShoulderUp.Name = "btnShoulderUp";
            this.btnShoulderUp.Size = new System.Drawing.Size(44, 25);
            this.btnShoulderUp.TabIndex = 26;
            this.btnShoulderUp.Text = "+";
            this.btnShoulderUp.UseVisualStyleBackColor = true;
            this.btnShoulderUp.Click                     +=                     new
System.EventHandler(this.btnShoulderUp_Click);
            //
            // btnWaistDown
            //
            this.btnWaistDown.Cursor = System.Windows.Forms.Cursors.Hand;
            this.btnWaistDown.Font   =   new   System.Drawing.Font("Microsoft   Sans
Serif",  10F,  System.Drawing.FontStyle.Bold,  System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.btnWaistDown.Location = new System.Drawing.Point(173, 36);
            this.btnWaistDown.Margin = new System.Windows.Forms.Padding(0);
            this.btnWaistDown.Name = "btnWaistDown";
```

```
            this.btnWaistDown.Size = new System.Drawing.Size(44, 25);
            this.btnWaistDown.TabIndex = 25;
            this.btnWaistDown.Text = "-";
            this.btnWaistDown.UseVisualStyleBackColor = true;
            this.btnWaistDown.Click                     +=                     new
System.EventHandler(this.btnWaistDown_Click);
            //
            // btnWaistUp
            //
            this.btnWaistUp.Cursor = System.Windows.Forms.Cursors.Hand;
            this.btnWaistUp.Font  =  new  System.Drawing.Font("Microsoft  Sans  Serif",
10F,       System.Drawing.FontStyle.Bold,       System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.btnWaistUp.Location = new System.Drawing.Point(126, 36);
            this.btnWaistUp.Margin = new System.Windows.Forms.Padding(0);
            this.btnWaistUp.Name = "btnWaistUp";
            this.btnWaistUp.Size = new System.Drawing.Size(44, 25);
            this.btnWaistUp.TabIndex = 24;
            this.btnWaistUp.Text = "+";
            this.btnWaistUp.UseVisualStyleBackColor = true;
            this.btnWaistUp.Click                     +=                     new
System.EventHandler(this.btnWaistUp_Click);
            //
            // btnTrackDown
            //
            this.btnTrackDown.Cursor = System.Windows.Forms.Cursors.Hand;
            this.btnTrackDown.Font   =   new   System.Drawing.Font("Microsoft   Sans
Serif",  10F,  System.Drawing.FontStyle.Bold,  System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.btnTrackDown.Location = new System.Drawing.Point(67, 35);
            this.btnTrackDown.Margin = new System.Windows.Forms.Padding(0);
            this.btnTrackDown.Name = "btnTrackDown";
            this.btnTrackDown.Size = new System.Drawing.Size(44, 25);
            this.btnTrackDown.TabIndex = 23;
            this.btnTrackDown.Text = "-";
            this.btnTrackDown.UseVisualStyleBackColor = true;
            this.btnTrackDown.Click                     +=                     new
System.EventHandler(this.btnTrackDown_Click);
            //
            // btnTrackUp
            //
            this.btnTrackUp.Cursor = System.Windows.Forms.Cursors.Hand;
            this.btnTrackUp.Font  =  new  System.Drawing.Font("Microsoft  Sans  Serif",
10F,       System.Drawing.FontStyle.Bold,       System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.btnTrackUp.Location = new System.Drawing.Point(20, 35);
            this.btnTrackUp.Margin = new System.Windows.Forms.Padding(0);
            this.btnTrackUp.Name = "btnTrackUp";
            this.btnTrackUp.Size = new System.Drawing.Size(44, 25);
            this.btnTrackUp.TabIndex = 22;
            this.btnTrackUp.Text = "+";
            this.btnTrackUp.UseVisualStyleBackColor = true;
            this.btnTrackUp.Click                     +=                     new
System.EventHandler(this.btnTrackUp_Click);
            //
            // label11
            //
            this.label11.AutoSize = true;
            this.label11.Font  =  new  System.Drawing.Font("Microsoft  Sans  Serif",
12F,       System.Drawing.FontStyle.Bold,       System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.label11.Location = new System.Drawing.Point(487, 16);
            this.label11.Name = "label11";
            this.label11.Size = new System.Drawing.Size(66, 20);
```

```
            this.label11.TabIndex = 21;
            this.label11.Text = "WRIST";
            //
            // label10
            //
            this.label10.AutoSize = true;
            this.label10.Font  =  new  System.Drawing.Font("Microsoft  Sans  Serif",
12F,       System.Drawing.FontStyle.Bold,      System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.label10.Location = new System.Drawing.Point(373, 16);
            this.label10.Name = "label10";
            this.label10.Size = new System.Drawing.Size(72, 20);
            this.label10.TabIndex = 20;
            this.label10.Text = "ELBOW";
            //
            // label9
            //
            this.label9.AutoSize = true;
            this.label9.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label9.Location = new System.Drawing.Point(238, 16);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(108, 20);
            this.label9.TabIndex = 19;
            this.label9.Text = "SHOULDER";
            //
            // label7
            //
            this.label7.AutoSize = true;
            this.label7.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label7.Location = new System.Drawing.Point(139, 16);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(65, 20);
            this.label7.TabIndex = 18;
            this.label7.Text = "WAIST";
            //
            // label6
            //
            this.label6.AutoSize = true;
            this.label6.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label6.Location = new System.Drawing.Point(33, 15);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(67, 20);
            this.label6.TabIndex = 17;
            this.label6.Text = "TRACK";
            //
            // tabPage2
            //
            this.tabPage2.Controls.Add(this.lbPosiciones);
            this.tabPage2.Controls.Add(this.txPosName);
            this.tabPage2.Controls.Add(this.btnGrabarPos);
            this.tabPage2.Location = new System.Drawing.Point(4, 22);
            this.tabPage2.Name = "tabPage2";
            this.tabPage2.Padding = new System.Windows.Forms.Padding(3);
            this.tabPage2.Size = new System.Drawing.Size(657, 183);
            this.tabPage2.TabIndex = 1;
            this.tabPage2.Text = "Posiciones/Rutas";
            this.tabPage2.UseVisualStyleBackColor = true;
            //
            // lbPosiciones
            //
            this.lbPosiciones.FormattingEnabled = true;
```

```
            this.lbPosiciones.Items.AddRange(new object[] {
            "HOME"});
            this.lbPosiciones.Location = new System.Drawing.Point(13, 39);
            this.lbPosiciones.Name = "lbPosiciones";
            this.lbPosiciones.Size = new System.Drawing.Size(154, 134);
            this.lbPosiciones.TabIndex = 2;
            //
            // txPosName
            //
            this.txPosName.Location = new System.Drawing.Point(13, 10);
            this.txPosName.Name = "txPosName";
            this.txPosName.Size = new System.Drawing.Size(154, 20);
            this.txPosName.TabIndex = 1;
            //
            // btnGrabarPos
            //
            this.btnGrabarPos.Location = new System.Drawing.Point(173, 10);
            this.btnGrabarPos.Name = "btnGrabarPos";
            this.btnGrabarPos.Size = new System.Drawing.Size(93, 20);
            this.btnGrabarPos.TabIndex = 0;
            this.btnGrabarPos.Text = "Grabar Posición";
            this.btnGrabarPos.UseVisualStyleBackColor = true;
            this.btnGrabarPos.Click                      +=                      new
System.EventHandler(this.btnGrabarPos_Click);
            //
            // tabPage3
            //
            this.tabPage3.Controls.Add(this.btnConectarWifi);
            this.tabPage3.Controls.Add(this.gbRemoto);
            this.tabPage3.Controls.Add(this.gbLocal);
            this.tabPage3.Location = new System.Drawing.Point(4, 22);
            this.tabPage3.Name = "tabPage3";
            this.tabPage3.Size = new System.Drawing.Size(657, 183);
            this.tabPage3.TabIndex = 2;
            this.tabPage3.Text = "Comunicaciones";
            this.tabPage3.UseVisualStyleBackColor = true;
            //
            // btnConectarWifi
            //
            this.btnConectarWifi.Location = new System.Drawing.Point(243, 142);
            this.btnConectarWifi.Name = "btnConectarWifi";
            this.btnConectarWifi.Size = new System.Drawing.Size(84, 30);
            this.btnConectarWifi.TabIndex = 2;
            this.btnConectarWifi.Text = "Conectar";
            this.btnConectarWifi.UseVisualStyleBackColor = true;
            this.btnConectarWifi.Click                      +=                      new
System.EventHandler(this.btnConectarWifi_Click);
            //
            // gbRemoto
            //
            this.gbRemoto.Controls.Add(this.label27);
            this.gbRemoto.Controls.Add(this.txPuertoRemoto);
            this.gbRemoto.Controls.Add(this.label26);
            this.gbRemoto.Controls.Add(this.txIPRemoto4);
            this.gbRemoto.Controls.Add(this.txIPRemoto3);
            this.gbRemoto.Controls.Add(this.txIPRemoto1);
            this.gbRemoto.Controls.Add(this.txIPRemoto2);
            this.gbRemoto.Location = new System.Drawing.Point(10, 78);
            this.gbRemoto.Name = "gbRemoto";
            this.gbRemoto.Size = new System.Drawing.Size(318, 50);
            this.gbRemoto.TabIndex = 1;
            this.gbRemoto.TabStop = false;
            this.gbRemoto.Text = "Conexión Remota";
            //
```

```
// label27
//
this.label27.AutoSize = true;
this.label27.Location = new System.Drawing.Point(234, 22);
this.label27.Name = "label27";
this.label27.Size = new System.Drawing.Size(41, 13);
this.label27.TabIndex = 16;
this.label27.Text = "Puerto:";
//
// txPuertoRemoto
//
this.txPuertoRemoto.Location = new System.Drawing.Point(276, 19);
this.txPuertoRemoto.MaxLength = 3;
this.txPuertoRemoto.Name = "txPuertoRemoto";
this.txPuertoRemoto.Size = new System.Drawing.Size(30, 20);
this.txPuertoRemoto.TabIndex = 14;
//
// label26
//
this.label26.AutoSize = true;
this.label26.Location = new System.Drawing.Point(10, 22);
this.label26.Name = "label26";
this.label26.Size = new System.Drawing.Size(68, 13);
this.label26.TabIndex = 13;
this.label26.Text = "Dirección IP:";
//
// txIPRemoto4
//
this.txIPRemoto4.Location = new System.Drawing.Point(192, 19);
this.txIPRemoto4.MaxLength = 3;
this.txIPRemoto4.Name = "txIPRemoto4";
this.txIPRemoto4.Size = new System.Drawing.Size(30, 20);
this.txIPRemoto4.TabIndex = 15;
//
// txIPRemoto3
//
this.txIPRemoto3.Location = new System.Drawing.Point(156, 19);
this.txIPRemoto3.MaxLength = 3;
this.txIPRemoto3.Name = "txIPRemoto3";
this.txIPRemoto3.Size = new System.Drawing.Size(30, 20);
this.txIPRemoto3.TabIndex = 14;
//
// txIPRemoto1
//
this.txIPRemoto1.Location = new System.Drawing.Point(84, 19);
this.txIPRemoto1.MaxLength = 3;
this.txIPRemoto1.Name = "txIPRemoto1";
this.txIPRemoto1.Size = new System.Drawing.Size(30, 20);
this.txIPRemoto1.TabIndex = 12;
//
// txIPRemoto2
//
this.txIPRemoto2.Location = new System.Drawing.Point(120, 19);
this.txIPRemoto2.MaxLength = 3;
this.txIPRemoto2.Name = "txIPRemoto2";
this.txIPRemoto2.Size = new System.Drawing.Size(30, 20);
this.txIPRemoto2.TabIndex = 13;
//
// gbLocal
//
this.gbLocal.Controls.Add(this.label28);
this.gbLocal.Controls.Add(this.txPuertoLocal);
this.gbLocal.Controls.Add(this.label25);
this.gbLocal.Controls.Add(this.txIPLocal4);
```

```
this.gbLocal.Controls.Add(this.txIPLocal3);
this.gbLocal.Controls.Add(this.txIPLocal2);
this.gbLocal.Controls.Add(this.txIPLocal1);
this.gbLocal.Location = new System.Drawing.Point(10, 16);
this.gbLocal.Name = "gbLocal";
this.gbLocal.Size = new System.Drawing.Size(318, 48);
this.gbLocal.TabIndex = 0;
this.gbLocal.TabStop = false;
this.gbLocal.Text = "Conexión Local";
//
// label28
//
this.label28.AutoSize = true;
this.label28.Location = new System.Drawing.Point(234, 20);
this.label28.Name = "label28";
this.label28.Size = new System.Drawing.Size(41, 13);
this.label28.TabIndex = 18;
this.label28.Text = "Puerto:";
//
// txPuertoLocal
//
this.txPuertoLocal.Location = new System.Drawing.Point(276, 17);
this.txPuertoLocal.MaxLength = 3;
this.txPuertoLocal.Name = "txPuertoLocal";
this.txPuertoLocal.Size = new System.Drawing.Size(30, 20);
this.txPuertoLocal.TabIndex = 17;
//
// label25
//
this.label25.AutoSize = true;
this.label25.Location = new System.Drawing.Point(10, 20);
this.label25.Name = "label25";
this.label25.Size = new System.Drawing.Size(68, 13);
this.label25.TabIndex = 12;
this.label25.Text = "Dirección IP:";
//
// txIPLocal4
//
this.txIPLocal4.Location = new System.Drawing.Point(192, 17);
this.txIPLocal4.MaxLength = 3;
this.txIPLocal4.Name = "txIPLocal4";
this.txIPLocal4.Size = new System.Drawing.Size(30, 20);
this.txIPLocal4.TabIndex = 11;
//
// txIPLocal3
//
this.txIPLocal3.Location = new System.Drawing.Point(156, 17);
this.txIPLocal3.MaxLength = 3;
this.txIPLocal3.Name = "txIPLocal3";
this.txIPLocal3.Size = new System.Drawing.Size(30, 20);
this.txIPLocal3.TabIndex = 10;
//
// txIPLocal2
//
this.txIPLocal2.Location = new System.Drawing.Point(120, 17);
this.txIPLocal2.MaxLength = 3;
this.txIPLocal2.Name = "txIPLocal2";
this.txIPLocal2.Size = new System.Drawing.Size(30, 20);
this.txIPLocal2.TabIndex = 9;
//
// txIPLocal1
//
this.txIPLocal1.Location = new System.Drawing.Point(84, 17);
this.txIPLocal1.MaxLength = 3;
```

```
            this.txIPLocal1.Name = "txIPLocal1";
            this.txIPLocal1.Size = new System.Drawing.Size(30, 20);
            this.txIPLocal1.TabIndex = 8;
            //
            // txLastText
            //
            this.txLastText.Location = new System.Drawing.Point(52, 434);
            this.txLastText.Name = "txLastText";
            this.txLastText.Size = new System.Drawing.Size(162, 20);
            this.txLastText.TabIndex = 13;
            //
            // axSPortAx1
            //
            this.axSPortAx1.Enabled = true;
            this.axSPortAx1.Location = new System.Drawing.Point(14, 430);
            this.axSPortAx1.Name = "axSPortAx1";
            this.axSPortAx1.OcxState                                                    =
((System.Windows.Forms.AxHost.State)(resources.GetObject("axSPortAx1.OcxState"))
);
            this.axSPortAx1.Size = new System.Drawing.Size(32, 32);
            this.axSPortAx1.TabIndex = 3;
            this.axSPortAx1.OnDCD                          +=                       new
AxSPortLib._ISPortAxEvents_OnDCDEventHandler(this.axSPortAx1_OnDCD);
            this.axSPortAx1.OnCommError                      +=                     new
AxSPortLib._ISPortAxEvents_OnCommErrorEventHandler(this.axSPortAx1_OnCommError);
            this.axSPortAx1.OnRxFlag                       +=                       new
System.EventHandler(this.axSPortAx1_OnRxFlag);
            this.axSPortAx1.OnBreak                       +=                        new
System.EventHandler(this.axSPortAx1_OnBreak);
            this.axSPortAx1.OnTxEmpty                      +=                       new
System.EventHandler(this.axSPortAx1_OnTxEmpty);
            this.axSPortAx1.OnDSR                          +=                       new
AxSPortLib._ISPortAxEvents_OnDSREventHandler(this.axSPortAx1_OnDSR);
            this.axSPortAx1.OnRing                         +=                       new
System.EventHandler(this.axSPortAx1_OnRing);
            this.axSPortAx1.OnCTS                          +=                       new
AxSPortLib._ISPortAxEvents_OnCTSEventHandler(this.axSPortAx1_OnCTS);
            this.axSPortAx1.OnRxChar                       +=                       new
AxSPortLib._ISPortAxEvents_OnRxCharEventHandler(this.axSPortAx1_OnRxChar);
            //
            // Teach
            //
            this.Teach.Enabled = false;
            this.Teach.Location = new System.Drawing.Point(558, 434);
            this.Teach.Name = "Teach";
            this.Teach.Size = new System.Drawing.Size(120, 27);
            this.Teach.TabIndex = 60;
            this.Teach.Text = "Automatic Mode";
            this.Teach.UseVisualStyleBackColor = true;
            this.Teach.Click += new System.EventHandler(this.Teach_Click);
            //
            // btnGuardar
            //
            this.btnGuardar.Enabled = false;
            this.btnGuardar.Location = new System.Drawing.Point(222, 434);
            this.btnGuardar.Name = "btnGuardar";
            this.btnGuardar.Size = new System.Drawing.Size(106, 27);
            this.btnGuardar.TabIndex = 11;
            this.btnGuardar.Text = "Guardar Proyecto";
            this.btnGuardar.Click                          +=                       new
System.EventHandler(this.btnGuardar_Click);
            //
            // btnAbrir
            //
```

```
            this.btnAbrir.Enabled = false;
            this.btnAbrir.Location = new System.Drawing.Point(334, 434);
            this.btnAbrir.Name = "btnAbrir";
            this.btnAbrir.Size = new System.Drawing.Size(106, 27);
            this.btnAbrir.TabIndex = 11;
            this.btnAbrir.Text = "Abrir Proyecto";
            this.btnAbrir.Click += new System.EventHandler(this.btnAbrir_Click);
            //
            // statusStrip
            //
            this.statusStrip.Items.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.toolStripStatusLabel});
            this.statusStrip.Location = new System.Drawing.Point(0, 467);
            this.statusStrip.Name = "statusStrip";
            this.statusStrip.Size = new System.Drawing.Size(692, 22);
            this.statusStrip.TabIndex = 61;
            this.statusStrip.Text = "StatusStrip";
            //
            // toolStripStatusLabel
            //
            this.toolStripStatusLabel.DisplayStyle                                    =
System.Windows.Forms.ToolStripItemDisplayStyle.Text;
            this.toolStripStatusLabel.Name = "toolStripStatusLabel";
            this.toolStripStatusLabel.Size = new System.Drawing.Size(38, 17);
            this.toolStripStatusLabel.Text = "Status";
            //
            // TimertoSend
            //
            this.TimertoSend.Tick                        +=                        new
System.EventHandler(this.TimertoSend_Tick);
            //
            // Form1
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.ClientSize = new System.Drawing.Size(692, 489);
            this.Controls.Add(this.statusStrip);
            this.Controls.Add(this.Teach);
            this.Controls.Add(this.txLastText);
            this.Controls.Add(this.tcPrincipal);
            this.Controls.Add(this.btnAbrir);
            this.Controls.Add(this.btnGuardar);
            this.Controls.Add(this.btnOpen);
            this.Controls.Add(this.axSPortAx1);
            this.Controls.Add(this.pictureBox1);
            this.Controls.Add(this.Log);
            this.Controls.Add(this.groupBox1);
            this.MaximumSize = new System.Drawing.Size(700, 600);
            this.MinimumSize = new System.Drawing.Size(700, 500);
            this.Name = "Form1";
            this.SizeGripStyle = System.Windows.Forms.SizeGripStyle.Show;
            this.Text = "DeucalionTalk";
            this.Leave += new System.EventHandler(this.Form1_Leave);
            this.Load += new System.EventHandler(this.Form1_Load);
            this.groupBox1.ResumeLayout(false);
            this.Log.ResumeLayout(false);
            this.Log.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
            this.tcPrincipal.ResumeLayout(false);
            this.tabPage1.ResumeLayout(false);
            this.cPanel.ResumeLayout(false);
            this.cPanel.PerformLayout();
            this.tabPage2.ResumeLayout(false);
```

```
            this.tabPage2.PerformLayout();
            this.tabPage3.ResumeLayout(false);
            this.gbRemoto.ResumeLayout(false);
            this.gbRemoto.PerformLayout();
            this.gbLocal.ResumeLayout(false);
            this.gbLocal.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.axSPortAx1)).EndInit();
            this.statusStrip.ResumeLayout(false);
            this.statusStrip.PerformLayout();
            this.ResumeLayout(false);
            this.PerformLayout();

        }
        #endregion

        [STAThread]
        static void Main()
        {
          Application.Run(new Form1());
        }

        private void WifiLoop()
        {
            Datos = new ArrayList();

            this.WiPort.Connect(this.WiPort.RemoteIPAddress,
this.WiPort.RemotePort);

            while (true)
            {
                Datos.Add(this.WiPort.GetReceivedData());
                Thread.Sleep(1000);
            }
        }

        private void sendData(string datos) {

            if (vRec == true)
            {
                Re.Items.Add(datos);
            }

            byte key;

            for (int i = 0; i < datos.Length; i++) {
                key = Convert.ToByte(datos[i]);
                axSPortAx1.Write(ref key, 1);
            }
        }

        private void doEvent()
        {
            if (((isok == true) || (cont == 0))&&(Re!=null))
            {
                sendData(Re.Items[cont].ToString());
                cont = cont + 1;
                isok = false;
            }
        }


        private void Form1_Load(object sender, System.EventArgs e)
        {
```

```
        vRec = false;
        vTeach = true;
        isok = false;
        cont = 0;
        isWaiting = false;
        //SerialPortAx
  for ( int i=0; i<axSPortAx1.CountPorts; i++ )
  {
    ComboPort.Items.Add( axSPortAx1.GetPortName( i ) );
  }
  ComboPort.SelectedIndex = 0;

  ComboBaudrate.SelectedIndex = 0;
  ComboDatabits.SelectedIndex = 2;
  ComboFC.SelectedIndex = 2;
  ComboParity.SelectedIndex = SerialPortAx.Constants.SP_NOPARITY;
  ComboStopbits.SelectedIndex = SerialPortAx.Constants.SP_ONESTOPBIT;
        tcPrincipal.Enabled = false;
        Automat = new Generador.AutomaticMode();
        Watch = new Generador.Error_Warning();
        track=0;
        wrist=0;
        shoulder=0;
        waist=0;
        elbow = 0;
        hand=0;
}

    private void btnOpen_Click(object sender, EventArgs e)
    {
        sendData("DE-ENERGIZE\r");


        if (axSPortAx1.IsOpened)
  {

    if (axSPortAx1.Close())
    {
      ComboPort.Enabled = true;
      textBox2.Enabled = false;

      btnOpen.Text = "Abrir";
                tcPrincipal.Enabled = false;
                Teach.Enabled = false;
                btnAbrir.Enabled = false;
                btnGuardar.Enabled = false;

      if ( !checkLog.Checked )
        ListLog.Items.Add( DateTime.Now.ToString ("T") +
                " - close port " + ComboPort.Text );

    }
  }
  else
  {
              /*ComboPort.Text*/
    if (axSPortAx1.Open("COM1"))
    {
                String config;
                ComboPort.Enabled = false;
      textBox2.Enabled = true;
                Re.Enabled = true;
```

```
                axSPortAx1.HandShake                                        =
SerialPortAx.Constants.SERIAL_DTR_CONTROL;
                axSPortAx1.FlowReplace                                      =
SerialPortAx.Constants.SERIAL_RTS_CONTROL
                                        |
SerialPortAx.Constants.SERIAL_AUTO_TRANSMIT
                                        |
SerialPortAx.Constants.SERIAL_AUTO_RECEIVE;

                config = "19200,N,8,2,x";

                axSPortAx1.InitString( config );

      btnOpen.Text = "Cerrar";
                tcPrincipal.Enabled = true;
                Teach.Enabled = true;
                btnAbrir.Enabled = true;
                btnGuardar.Enabled = true;

      if ( !checkLog.Checked )
       ListLog.Items.Add(DateTime.Now.ToString ("T")+
             " - open port " + ComboPort.Text );
                sendData("ENERGIZE\r");
    }
   }
  }

  private            void            textBox2_KeyPress(object            sender,
System.Windows.Forms.KeyPressEventArgs e)
  {
   byte key = Convert.ToByte(e.KeyChar);
   axSPortAx1.Write(ref key, 1);

   if ( !checkLog.Checked )
    ListLog.Items.Add(DateTime.Now.ToString ("T") + " - 1 - byte sent");
  }

  private void ComboBaudrate_SelectedIndexChanged(object sender, System.EventArgs
e)
  {
   axSPortAx1.BaudRate = Convert.ToInt32 (ComboBaudrate.Text);
  }

  private  void  ComboParity_SelectedIndexChanged(object  sender,  System.EventArgs
e)
  {
   axSPortAx1.Parity = Convert.ToByte (ComboParity.SelectedIndex);
  }

  private void ComboDatabits_SelectedIndexChanged(object sender, System.EventArgs
e)
  {
   axSPortAx1.Databits = Convert.ToByte (ComboDatabits.SelectedIndex + 5);
  }

  private void ComboStopbits_SelectedIndexChanged(object sender, System.EventArgs
e)
  {
   axSPortAx1.StopBits = Convert.ToByte (ComboStopbits.SelectedIndex);
  }

  private void ComboFC_SelectedIndexChanged(object sender, System.EventArgs e)
  {
      switch (  ComboFC.SelectedIndex )
```

```
      {
       case 0:  // Xon\Xoff
       axSPortAx1.HandShake = SerialPortAx.Constants.SERIAL_DTR_CONTROL;
       axSPortAx1.FlowReplace = SerialPortAx.Constants.SERIAL_RTS_CONTROL
                | SerialPortAx.Constants.SERIAL_AUTO_TRANSMIT
                | SerialPortAx.Constants.SERIAL_AUTO_RECEIVE;
      break;
       case 1: // Hardware
       axSPortAx1.HandShake = SerialPortAx.Constants.SERIAL_DTR_CONTROL
                 | SerialPortAx.Constants.SERIAL_DTR_HANDSHAKE;
       axSPortAx1.FlowReplace = SerialPortAx.Constants.SERIAL_RTS_CONTROL
                | SerialPortAx.Constants.SERIAL_RTS_HANDSHAKE;
      break;
       case 2: // None
                    axSPortAx1.HandShake                                          =
SerialPortAx.Constants.SERIAL_DTR_HANDSHAKE |
               SerialPortAx.Constants.SERIAL_CTS_HANDSHAKE |
               SerialPortAx.Constants.SERIAL_DSR_HANDSHAKE;
       axSPortAx1.FlowReplace = SerialPortAx.Constants.SERIAL_RTS_HANDSHAKE;
        break;
   }
  }

  private           void           axSPortAx1_OnRxChar(object           sender,
AxSPortLib._ISPortAxEvents_OnRxCharEvent e)
  {
   if ( !checkLog.Checked )
    ListLog.Items.Add(DateTime.Now.ToString ("T")  + " - OnRxChar "
                + Convert.ToString(e.lCount));

   byte [] Buff  = new byte [e.lCount];
   axSPortAx1.Read (out Buff [0], out e.lCount);
         lastText = System.Text.ASCIIEncoding.ASCII.GetString(Buff);
         textBox2.AppendText(lastText);
   if ( !checkTerminal.Checked )
    //textBox2.Text=lastText;

            if (isplay == true)
            {
                if (lastText.Contains("\r\n"))
                {
                    isWaiting = false;
                    isok = true;
                    doEvent();
                }

            }
  }

  private void axSPortAx1_OnBreak(object sender, System.EventArgs e)
  {
   if ( !checkLog.Checked )
    ListLog.Items.Add(DateTime.Now.ToString ("T") + " - OnBreak ");
  }

  private          void          axSPortAx1_OnCommError(object          sender,
AxSPortLib._ISPortAxEvents_OnCommErrorEvent e)
  {
   if ( !checkLog.Checked )
    ListLog.Items.Add(DateTime.Now.ToString ("T") + " - OnCommError ");
  }

  private            void            axSPortAx1_OnCTS(object            sender,
AxSPortLib._ISPortAxEvents_OnCTSEvent e)
```

```
  {
    if ( !checkLog.Checked )
      ListLog.Items.Add(DateTime.Now.ToString  ("T")  +  "  -  OnCTS  set  "  +
e.bCts.ToString ());
  }

  private            void            axSPortAx1_OnDCD(object            sender,
AxSPortLib._ISPortAxEvents_OnDCDEvent e)
  {
    if ( !checkLog.Checked )
      ListLog.Items.Add(DateTime.Now.ToString  ("T")  +  "  -  OnDCD  set  "  +
e.bDcd.ToString ());
  }

  private            void            axSPortAx1_OnDSR(object            sender,
AxSPortLib._ISPortAxEvents_OnDSREvent e)
  {
    if ( !checkLog.Checked )
      ListLog.Items.Add(DateTime.Now.ToString  ("T")  +  "  -  OnDSR  set  "  +
e.bDsr.ToString ());
  }

  private void axSPortAx1_OnRing(object sender, System.EventArgs e)
  {
    if ( !checkLog.Checked )
      ListLog.Items.Add(DateTime.Now.ToString ("T") + " - OnRing");
  }

  private void axSPortAx1_OnRxFlag(object sender, System.EventArgs e)
  {
    if ( !checkLog.Checked )
      ListLog.Items.Add(DateTime.Now.ToString ("T") + " - OnRxFlag");
  }

  private void axSPortAx1_OnTxEmpty(object sender, System.EventArgs e)
  {
    if ( !checkLog.Checked )
      ListLog.Items.Add(DateTime.Now.ToString ("T") + " - OnTxEmpty");
  }

      private void btnTrackUp_Click(object sender, EventArgs e)
      {
          if  ((Convert.ToInt32(vTrack.Text)  +  Convert.ToInt32(txTrack.Text))  <
25000)
          {
              sendData("TELL TRACK " + txTrack.Text + " MOVE\r");
              vTrack.Text    =    Convert.ToString(Convert.ToInt32(vTrack.Text)    +
Convert.ToInt32(txTrack.Text));
          }
      }

      private void btnTrackDown_Click(object sender, EventArgs e)
      {
          if ((Convert.ToInt32(vTrack.Text) - Convert.ToInt32(txTrack.Text)) > 0)
          {
              sendData("TELL TRACK -" + txTrack.Text + " MOVE\r");
              vTrack.Text    =    Convert.ToString(Convert.ToInt32(vTrack.Text)    -
Convert.ToInt32(txTrack.Text));
          }
      }

      private void btnWaistUp_Click(object sender, EventArgs e)
      {
```

```
        if  (Convert.ToInt32(vWaist.Text)  +  Convert.ToInt32(txWaist.Text)  <
19600)
        {
            sendData("TELL WAIST " + txWaist.Text + " MOVE\r");
            vWaist.Text   =   Convert.ToString(Convert.ToInt32(vWaist.Text)   +
Convert.ToInt32(txWaist.Text));
        }
    }

    private void btnWaistDown_Click(object sender, EventArgs e)
    {
        if  (Convert.ToInt32(vWaist.Text)  -  Convert.ToInt32(txWaist.Text)  >  -
19600)
        {
            sendData("TELL WAIST -" + txWaist.Text + " MOVE\r");
            vWaist.Text   =   Convert.ToString(Convert.ToInt32(vWaist.Text)   -
Convert.ToInt32(txWaist.Text));
        }
    }

    private void btnShoulderUp_Click(object sender, EventArgs e)
    {

        if  (Convert.ToInt32(vShoulder.Text)  +  Convert.ToInt32(txShoulder.Text)
< 22500)
        {
            sendData("TELL SHOULDER " + txShoulder.Text + " MOVE\r");
            vShoulder.Text = Convert.ToString(Convert.ToInt32(vShoulder.Text) +
Convert.ToInt32(txShoulder.Text));
        }

    }

    private void btnShoulderDown_Click(object sender, EventArgs e)
    {
        if  (Convert.ToInt32(vShoulder.Text)  -  Convert.ToInt32(txShoulder.Text)
> -22500)
        {
            sendData("TELL SHOULDER -" + txShoulder.Text + " MOVE\r");
            vShoulder.Text = Convert.ToString(Convert.ToInt32(vShoulder.Text) -
Convert.ToInt32(txShoulder.Text));
        }
    }

    private void btnElbowUp_Click(object sender, EventArgs e)
    {
        if  (Convert.ToInt32(vElbow.Text)  +  Convert.ToInt32(txElbow.Text)  <
15000)
        {
            sendData("TELL ELBOW " + txElbow.Text + " MOVE\r");
            vElbow.Text   =   Convert.ToString(Convert.ToInt32(vElbow.Text)   +
Convert.ToInt32(txElbow.Text));
        }
    }

    private void btnElbowDown_Click(object sender, EventArgs e)
    {
        if  (Convert.ToInt32(vElbow.Text)  -  Convert.ToInt32(txElbow.Text)  >  -
15000)
        {
            sendData("TELL ELBOW -" + txElbow.Text + " MOVE\r");
            vElbow.Text   =   Convert.ToString(Convert.ToInt32(vElbow.Text)   -
Convert.ToInt32(txElbow.Text));
        }
```

```
    }

    private void btnWristUp_Click(object sender, EventArgs e)
    {
        if  (Convert.ToInt32(vWrist.Text)  +  Convert.ToInt32(txWrist.Text)  <
11000)
        {
            sendData("TELL WRIST " + txWrist.Text + " MOVE\r");
            vWrist.Text   =   Convert.ToString(Convert.ToInt32(vWrist.Text)   +
Convert.ToInt32(txWrist.Text));
        }
    }

    private void btnWristDown_Click(object sender, EventArgs e)
    {
        if  (Convert.ToInt32(vWrist.Text)  -  Convert.ToInt32(txWrist.Text)  >  -
11000)
        {
            sendData("TELL WRIST -" + txWrist.Text + " MOVE\r");
            vWrist.Text   =   Convert.ToString(Convert.ToInt32(vWrist.Text)   -
Convert.ToInt32(txWrist.Text));
        }
    }

    private void btnHome_Click(object sender, EventArgs e)
    {
        sendData("HOME\r");
        vWaist.Text = "0";
        vWrist.Text = "0";
        vElbow.Text = "0";
        vTrack.Text = "0";
        vShoulder.Text = "0";
        vHand.Text = "0";
    }

    private void btnCalibrate_Click(object sender, EventArgs e)
    {
        sendData("CALIBRATE\r");
    }

    private void btnStart_Click(object sender, EventArgs e)
    {
        sendData("START\r");
    }

    private void button2_Click(object sender, EventArgs e)
    {
        cont = 0;
        isplay = true;
        Rec.Text = Re.Text;
        doEvent();
        Rec.Text = Re.Text;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        if (vRec == false)
        {
            vRec = true;
            button1.Text="Record Off";
            lines = 0;
            sendData("HOME\r");
            vWaist.Text = "0";
            vWrist.Text = "0";
```

```
            vElbow.Text = "0";
            vTrack.Text = "0";
            vShoulder.Text = "0";
            vHand.Text = "0";
        }
        else
        {
            vRec = false;
            button1.Text = "Record On";
            Re.Text = " ";
        }
    }

    private void btnGrabarPos_Click(object sender, EventArgs e)
    {
        if (txPosName.Text != "")
        {
            lastText = "";
            isWaiting = true;
            sendData("PLACE " + txPosName.Text + "\r");
            lbPosiciones.Items.Add(txPosName.Text);
        }
        else
            MessageBox.Show("Ingrese un nombre para la posición!","Error");
    }

    private void btnHandUp_Click(object sender, EventArgs e)
    {
        if (Convert.ToInt32(vHand.Text) + Convert.ToInt32(txHand.Text)<3500)
        {
         sendData("TELL HAND " + txHand.Text + " MOVE\r");
         vHand.Text     =     Convert.ToString(Convert.ToInt32(vHand.Text)     +
Convert.ToInt32(txHand.Text));
        }
    }

    private void btnHandDown_Click(object sender, EventArgs e)
    {
        if (Convert.ToInt32(vHand.Text) - Convert.ToInt32(txHand.Text) > -3500)
        {
            sendData("TELL HAND -" + txHand.Text + " MOVE\r");
            vHand.Text    =    Convert.ToString(Convert.ToInt32(vHand.Text)    -
Convert.ToInt32(txHand.Text));
        }
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        sendData(Re.Items[cont].ToString());
        if (lastText.Contains("OK"))
        {
         cont = cont + 1;
        }
    }

    private void Form1_Leave(object sender, EventArgs e)
    {
        sendData("HOME\r");
        sendData("DE-ENERGIZE\r");
    }

    private void button3_Click(object sender, EventArgs e)
    {
        isok = true;
```

```
            isplay = true;
            Re.Text = "";
            Re.Items.Add("HOME\r");
            Re.Items.Add("DECIMAL 0 0 0 -7000 -17000 9000 JMA\r");
            Re.Items.Add("DECIMAL 0 0 0 0 0 9000 JMA\r");
            Re.Items.Add("DECIMAL 9000 1000 0 7000 17000 9000 JMA\r");
            Re.Items.Add("HAND -1000 MOVE\r");
            Re.Items.Add("WRIST 1000 MOVE\r");
            Re.Items.Add("HAND 1000 MOVE\r");
            Re.Items.Add("HAND -1000 MOVE\r");
            Re.Items.Add("HAND 1000 MOVE\r");
            Re.Items.Add("DECIMAL 15000 0 0 0 0 9000 JMA\r");
            Re.Items.Add("DECIMAL 17000 0 0 -7000 -17000 9000 JMA\r");
            Re.Items.Add("HAND -1000 MOVE\r");
            Re.Items.Add("WRIST 1000 MOVE\r");
            Re.Items.Add("HAND 1000 MOVE\r");
            Re.Items.Add("DECIMAL 17000 0 0 0 0 9000 JMA\r");
            Re.Items.Add("DECIMAL 19000 1000 0 7000 17000 9000 JMA\r");
            Re.Items.Add("DECIMAL 21000 1000 0 7000 17000 0 JMA\r");
            Re.Items.Add("HOME\r");
            doEvent();
        }

        private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
        {

        }

        private void Re_SelectedIndexChanged(object sender, EventArgs e)
        {

        }

        private void Log_Enter(object sender, EventArgs e)
        {

        }

        private void btnConectarWifi_Click(object sender, EventArgs e)
        {
            if ((this.txIPLocal1.Text == "") || (this.txIPLocal2.Text == "") ||
(this.txIPLocal3.Text    ==    "")    ||    (this.txIPLocal4.Text    ==    "")    ||
(this.txPuertoLocal.Text == ""))
            {
                MessageBox.Show("Debe ingresar una dirección IP local y su puerto
para escuchar.");
            }
            else if ((this.txIPRemoto1.Text == "") || (this.txIPRemoto2.Text == "")
||   (this.txIPRemoto3.Text   ==   "")   ||   (this.txIPRemoto4.Text   ==   "")   ||
(this.txPuertoRemoto.Text == ""))
            {
                MessageBox.Show("Debe ingresar una dirección IP remota y su puerto
para enviar datos.");
            }
            else{
                try
                {
                    string     DireccionRemota     =     this.txIPRemoto1.Text     +
this.txIPRemoto2.Text + this.txIPRemoto3.Text + this.txIPRemoto4.Text;
                    WiPort.SetAddress(DireccionRemota,
Convert.ToInt32(this.txPuertoRemoto.Text));
                    WifiThread.Start();
                }
                catch
```

```
        {
            return;
        }
    }
}

private void revision_Datos(string _Datos)
{
    string Dat = "";
    string os = "";
    Dat = Dat + _Datos[0];
    os = os + _Datos[1];


    switch (_Datos)
    {
        case "a": track = Convert.ToInt16(os);
            break;
        case "b": waist = Convert.ToInt16(os);
            break;
        case "c": elbow = Convert.ToInt16(os);
            break;
        case "d": shoulder = Convert.ToInt16(os);
            break;
        case "e": wrist = Convert.ToInt16(os);
            break;
        case "f": hand = Convert.ToInt16(os);
            break;

    }

}

private void Teach_Click(object sender, EventArgs e)
{
    if (vTeach == false)
    {
        tcPrincipal.Enabled = true;
        Teach.Enabled = true;
        vTeach = true;
        Teach.Text = "Automatic Mode";
        lines = 0;
        TimertoSend.Enabled = false;
    }
    else
    {
        vTeach = false;
        Teach.Text = "Teach Mode";
        tcPrincipal.Enabled = false;
        Re.Text = " ";
        TimertoSend.Enabled = true;
    }


}

private void Teach_Click_1(object sender, EventArgs e)
{
}

private void btnGuardar_Click(object sender, EventArgs e)
{
    bool grabado= false;
    FileManager toSave = new FileManager();
```

```
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.InitialDirectory                                    =
Environment.GetFolderPath(Environment.SpecialFolder.Personal);
            saveFileDialog.Filter = "DeucaTalk Proyect (*.xml)|*.xml";
            if (saveFileDialog.ShowDialog(this) == DialogResult.OK)
            {
                string FileName = saveFileDialog.FileName;
                grabado = toSave.SaveProyect(Re, FileName);

                if (grabado)
                {
                    statusStrip.Text = "Project " + FileName + " saved";
                }
                else
                {
                    statusStrip.Text = "Existe algun error al escribir el archivo";
                }
            }

        }

        private void btnAbrir_Click(object sender, EventArgs e)
        {
            FileManager toLoad = new FileManager();


            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.InitialDirectory                                    =
Environment.GetFolderPath(Environment.SpecialFolder.Personal);
            openFileDialog.Filter = "DeucaTalk Proyect (*.xml)|*.xml";
            if (openFileDialog.ShowDialog(this) == DialogResult.OK)
            {
                string FileName = openFileDialog.FileName;
                Re = toLoad .toLoad(FileName);

                if (toLoad.getGrabado())
                {
                    statusStrip.Text  =  "Proyecto  "  +  FileName  +  "  Cargado  con
exito";
                    toolStripStatusLabel.Text  =  "Presione  Play  para  correr  la
rutina cargada";
                }
                else
                {
                    statusStrip.Text = "Error al Cargar el proyecto  " + FileName ;
                    Re.Items.Clear();
                }
            }

        }

        private void TimertoSend_Tick(object sender, EventArgs e)
        {
            for (int i=0; i< Datos.Count;i++)
            {
                revision_Datos(Datos[i].ToString());
            }

            Watch = Automat.Convertir(track, waist, shoulder, elbow, wrist, hand);
            WiPort.SendData(Convert.ToString(Convert.ToChar(Watch.getError())));
            WiPort.SendData(Convert.ToString(Convert.ToChar(Watch.getWarning())));
            Re.Items.Add("DECIMAL "+  Watch.getMessege()+" JMA\r");
        }
```

```
    }
}
```

V. Glosario

- *AutoCAD:* programa especializado en dibujo técnico en dos dimensiones y tres dimensiones.

- *Flexible Manufacturing System (FMS):* sistema de modelado de piezas a través de fresado y torneado que permite la fabricación de diversas piezas las cuales son identificadas por el sistema a través de algún programa de coordenadas.

- *Hyperterminal:* es un programa de computadoras para comunicarse con otros equipos a través de protocolos establecidos como serial, TCP, telnet y otros.

- *Multisim:* programa especializado en el diseño y/o simulación de circuitos electrónicos con la ventaja que posee modelos genéricos y por fabricante para algunos de los componentes.

- *Open System Interconnection (OSI):* modelo que estandariza la comunicación de aplicaciones a través de una red mediante la implementación 7 capas: física, enlace, red, transporte, sesión, presentación y aplicación.

- *Organizationally Unique Identifiers (OUI):* una de las organizaciones de IEEE que regula la administración de identificadores a cada empresa u organización de manera que se mantenga la unicidad en los códigos proporcionados. Entre los códigos que brindan se encuentra la dirección MAC, direcciones de grupos, direcciones de acceso de subredes, etc.

- Peripheral Interface Controller (PIC): es una computadora que contiene ROM, RAM y puertos de entrada y salida que almacena y ejecuta un conjunto de instrucciones.

- *Printed Circuit Board (PCB):* es un circuito impreso que básicamente es una tarjeta y sus componentes soldados a la misma de manera que la placa hace todas las conexiones necesarias de acuerdo al diseño del circuito.

- *Ultiboard:* programa de la familia Multisim especializado en el diseño de las conexiones para la fabricación de la tarjeta de un circuito impreso.

- Bit error: un Bit error ocurre si el transmisor transmite un bit dominante y detecta un bit recesivo, o si transmite un bit recesivo y detecta un bit dominante, cuando monitorea el nivel del bus actual y lo compara con el bit que acaba de ser enviado.

- Buffer: es una ubicación de la memoria en el PIC reservada para el almacenamiento temporal de información digital, mientras que está esperando ser procesada

- CRC: Códigos de Redundancia Cíclica o códigos polinómicos.

- Oscilador: es aquel oscilador que incluye en su realimentación un resonador piezoeléctrico.

- PIC: Familia de microcontroladores del fabricante Microchip

- Stuff bit error: si, dentro del bloque de comienzo y el delimitador CRC, seis bits consecutivos son detectados con la misma polaridad, se viola la regla del bit stuffing. Por lo que ocurre un Stuff bit error y se genera un bloque de error para que el mensaje sea repetido.

- *Transceiver*: es un transmisor y receptor de señales; se deriva de las palabras en ingles *transmitter* y *receiver*

- Corriente Eddy: Es causada cuando un campo magnético variable intersecta un conductor, o viceversa. El movimiento relativo causa una circulación de electrones, o corriente dentro del conductor. Estas corrientes circulares de Eddy crean electroimanes con campos magnéticos que se oponen al efecto del campo magnético aplicado (ver Ley de Lenz). Mientras más fuerte sea el campo magnético aplicado, o mayor la conductividad del conductor, o mayor la velocidad relativa de movimiento; mayores son las corrientes de Eddy y los campos opositores generados.
- Magnetismo: el magnetismo es un fenómeno por el que los materiales ejercen fuerzas de atracción o repulsión a otros materiales.
- Microcontrolador: es un circuito integrado o chip que incluye en su interior las tres unidades funcionales de una computadora: CPU, Memoria y Unidades de E/S.
- Protocolo: es el conjunto de reglas que especifican el intercambio de datos u órdenes durante la comunicación entre las entidades que forman parte de una red.
- *Aceleración:* La aceleración es la magnitud física que mide la tasa de variación de la velocidad respecto del tiempo. Es una magnitud vectorial con dimensiones de longitud/tiempo² (en unidades del sistema internacional se usa generalmente [m/s²]).
-
- *Electrostática.* es la rama de la física que estudia los fenómenos eléctricos producidos por distribuciones de cargas estáticas, esto es, el campo electrostático de un cuerpo cargado.
-
- *Frecuencia de Resonancia:* La frecuencia de resonancia de un cuerpo rígido o un sistema es aquella a la cual la respuesta del mismo, dado un estímulo, es la máxima posible.
-

- *Masa:* La masa es la medida de la inercia de un cuerpo. Aunque es frecuente que se defina como la cantidad de materia contenida en un cuerpo, esta última definición es incompleta.
-
- *PIC:* es una familia de los microcontroladores de la arquitectura de Harvard hechos por Microchip Technology, derivados del PIC1650 desarrollado originalmente por General Instrument's Microelectronics Division.
- El nombre PIC era originalmente un acrónimo "Programmable Intelligent Computer".
-
- *Piezoeléctrico:* La piezoelectricidad es un fenómeno presentado por determinados cristales que al ser sometidos a tensiones mecánicas adquieren una polarización eléctrica en su masa, apareciendo una diferencia de potencial y cargas eléctricas en su superficie. Este fenómeno también se presenta a la inversa, esto es, se deforman bajo la acción de fuerzas internas al ser sometidos

- a un campo eléctrico. El efecto piezoeléctrico es normalmente reversible: al dejar de someter los cristales a un voltaje exterior o campo eléctrico, recuperan su forma.
- **CAN:** CAN, o CAN Bus, es la forma abreviada de Controller Area Network. Es un bus de comunicaciones serial para aplicaciones de control en tiempo real, con una velocidad de comunicación de hasta 1 Mbit por segundo, y tiene excelente capacidad de detección y aislamiento de errores. (CanOBD2, 2006)
- **IP:** La dirección IP es una serie de números asociadas a un dispositivo (generalmente una computadora), con la cual es posible identificarlo dentro de una red configurada específicamente para utilizar este tipo de direcciones (una red configurada con el protocolo I.P. - Internet Protocol). Internet es un ejemplo de una red basada en protocolo IP version 4. (Alegsa, 2006)
- **Microcontrolador:** Un microcontrolador es un circuito integrado que nos ofrece las posibilidades de un pequeño computador. En su interior encontramos un procesador, memoria, y varios periféricos. El secreto de los microcontroladores lo encontramos en su tamaño, su precio y su diversidad. Su valor medio de seis euros, y su tamaño se reduce a unos pocos centímetros cuadrados. (Parallax, 2006)
- **Multisim:** NI Multisim es una herramienta para el diseño y simulación de circuitos eléctricos y electrónicos. Esta herramienta proporciona avanzadas características que permiten ir desde la fase de diseño a la de producción utilizando una misma herramienta. Ofrece entradas esquemáticas, una amplia base de datos, simulación SPICE, entradas y simulación VHDL o Verilog, puede manejar circuitos de radiofrecuencia, realiza postprocesado y es capaz de generar la placa PCB. (Addlink, 2007)

- **PIC:** Los 'PIC' son una familia de microcontroladores tipo RISC fabricados por Microchip Technology Inc. y derivados del PIC1650. Por PIC se entiende Peripheral Interface Controller o un Controlador de Interfaz Periférico. (Neboisa, 2005)
- **WiPort:** El WiPort dispositivo inmerso inalámbrico, es una solución de red basado en la norma IEEE 802.11b/g del estándar inalámbrico, con capacidad WPA y WEP. El WiPort permite agregar conectividad inalámbrica a productos mediante su incorporación al un circuito con mínimas complicaciones. (Lantronix, 2007)

  WIFI: Logo de Wi-Fi Alliance,que certifica equipos que cumplen con la regal IEEE 802.1, del estándar de Ethernet inalámbrico. (ZDNet Dictionary, 2007).
- **Abstracción**: el proceso de separar las cualidades de un objeto para considerarlas aisladamente o para considerar el mismo objeto en su pura esencia o noción.
- **Aplicación**: en la computación, se refiere a la descripción, software y documentación que definen la integración de la computadora en una tarea.
- **Arquitectura**: la estructura bajo la cual se desarrolla una aplicación de computación.
- **Articulación**: traducción de *joint*, es el enlace de dos piezas de un robot que representa un grado de libertad en su movimiento.
- **ASCII**: siglas de *American Standard Code for Information Interchange*, código normalizado que utiliza en total 8 bits y representa un caracter de 1 byte.
- **Bit**: contracción de *binary digit*, que designa la unidad mínima de información que puede representarse físicamente, representada por un 1 ó un 0.
- **Buffer**: un área de datos compartida por dispositivos de hardware o procesos de programa que operan a velocidades distintas o con un nivel distinto de

prioridades, que permite a cada dispositivo o proceso operar sin retrasarse por esperar a otro.

- **Byte**: cadena fija de 8 bits que se emplea para codificar un caracter.
- **Capa**: abstracción del software que permite el desarrollo por niveles extendidos uno sobre otro, intercomunicados según su posición relativa.
- **Caracter**: representación en la computación de un símbolo, particularmente de las letras de un abecedario, que ocupa 1 byte de espacio.
- **Clase**: en la programación orientada a objetos, una clase es una construcción que agrupa propiedades y métodos relacionados a un tipo de objeto específico, que manipulan los datos del objeto y realizan tareas.
- **Computación**: conjunto de disciplinas y técnicas desarrolladas para el tratamiento automático de la información mediante el uso de computadoras.
- **Datagrama**: en el protocolo TCP, representa un paquete contenido en sí mismo e independiente de los demás paquetes, que contiene información suficiente para el enrutamiento de una terminal a otra sin depender de intercambios anteriores entre el equipo y la red.
- **Data link:** es la capa 2 del modelo OSI (Open Systems Interconnection) para desarrollo de software, y corresponde a la que se comunica con la capa física y la capa de red.
- **Dispositivo periférico**: cualquier dispositivo de entrada, salida o almacenamiento que está conectado de manera externa o interna al CPU, tal como un monitor, una impresora, un disco duro, o un ratón.
- **DLL**: siglas de *Dynamically Linked Library*, es un archivo ejecutable que permite a los programas compartir código y otros recursos necesarios para realizar tareas específicas.
- **Encapsulamiento**: en la programación orientada a objetos, se refiere a ocultar el funcionamiento de un objeto o un método de otros para lograr una perspectiva de "caja negra" mediante la cual los demás objetos utilizan las funciones sin saber cómo están implementadas.

- **Framework**: es un espacio definido para la programación que contiene funciones y una implementación específica, en el cual se desarrollan las aplicaciones (literalmente, marco de trabajo).
- **Hardware**: el equipo físico que forma una computadora y sus dispositivos periféricos.
- **Interfaz**: para una aplicación, la interfaz de usuario es una manera gráfica, de preferencia amigable, de comunicar al usuario con la aplicación o el sistema operativo.
- **Librería**: una colección predefinida de clases, objetos y funciones que puede ser utilizada por un programa y evita el desarrollo de funciones ya existentes.
- **Megaproyecto**: un proyecto universitario de gran extensión que involucra varias disciplinas a la vez y permite la interacción de los estudiantes de cada una de ellas durante su desarrollo.
- **Memoria**: un lugar de almacenamiento temporal de datos al cual el procesador de una computadora puede acceder rápidamente, y que usualmente contiene las partes principales del sistema operativo, y algunos o todos los programas que están siendo utilizados junto con sus datos.
- **Namespace**: una agrupación lógica de los nombres utilizados dentro de un programa. En el marco de .NET, es una librería de clases.
- **Paquete**: la unidad básica de datos que se transmite entre un origen y un destino en cualquier red que utilice enrutamiento de paquetes, tal como Internet.
- **Pipeline**: un canal directo a través del cual se transmite información privadamente.
- **Plataforma**: el sistema operativo utilizado en una computadora dada.
- **Protocolo**: en la tecnología de la información, un protocolo es una serie de reglas especiales que los puntos finales en una conexión utilizan para comunicarse.

- **Puerto serial**: una interfaz de comunicación basada en el estándar RS-232, en la cual sólo un bit se transmite a la vez (serialmente).
- **Puerto TCP**: denominación de los extremos de una conexión lógica en el protocolo TCP o UDP, identificado con un número designado entre 1 y 65535 y diferente del los que están siendo utilizados por otras aplicaciones.
- **Requerimiento**: una característica específica de un sistema, que se utiliza durante la fase de análisis de sistemas en el desarrollo de software, para identificar los puntos clave que se deben tomar en cuenta para el diseño del sistema a desarrollar.
- **ROBOFORTH**: un lenguaje de programación de 4ta generación utilizado para programar los movimientos de un robot móvil como el R17. Se basa en los movimientos posicionales de cada eje o articulación, aunque también soporta coordenadas cartesianas.
- **Robótica**: la disciplina compuesta por el diseño, manufactura y aplicación de robots.
- **Robot**: un agente artificial mecánico o virtual, con la característica de ser programable y poder interactuar con el ambiente que lo rodea.
- **ROBWIN**: una aplicación para programar el robot R17 específicamente.
- **Rutina**: una secuencia de instrucciones ejecutadas por el robot R17, que realiza una tarea específica y se puede ejecutar periódicamente.
- **Sentencia o instrucción**: un comando enviado al robot que realiza un movimiento básico específico. En un programa, una instrucción es una línea de código.
- **Sistema**: una organización de recursos y procedimientos, regulada por reglas de interacción e interdependencia, cuyo propósito es lograr un conjunto de funciones específicas.
- **Socket**: representación de una conexión única entre dos aplicaciones que se comunican a través de una red. Los extremos son los puertos TCP utilizados en cada computadora.

- **Software**: un término general que representa los diferentes tipos de programas utilizados para operar una computadora y sus dispositivos relacionados.
- **WiFi**: se refiere a cualquier tipo de red del estándar 802.11, relacionado con el uso de tecnologías inalámbricas en la topología de una red de comunicaciones.
- **Acknowledge error:** en el campo acknowledge del mensaje, el transmisor chequea si el rubro acknowledge (el cual fue enviado como un bit recesivo) contenga un bit dominante. Si este no es el caso, ningún otro nodo recibió el bloque correcto. Por lo que ocurre un Acknowledge Error, se genera un bloque de error y el mensaje deberá de ser repetido.
-
- **Bit error:** un Bit error ocurre si el transmisor transmite un bit dominante y detecta un bit recesivo, o si transmite un bit recesivo y detecta un bit dominante, cuando monitorea el nivel del bus actual y lo compara con el bit que acaba de ser enviado.
- **Buffer:** es una ubicación de la memoria en el PIC reservada para el almacenamiento temporal de información digital, mientras que está esperando ser procesada
- **CRC:** los **códigos cíclicos** también se llaman CRC (**Códigos de Redundancia Cíclica**) o códigos polinómicos. Su uso está muy extendido porque pueden implementarse en hardware con mucha facilidad y son muy potentes.
- Estos códigos se basan en el uso de un polinomio generador **G(X)** de grado **r**, y en el principio de que **n** bits de datos binarios se pueden considerar como los coeficientes de un polinomio de orden **n-1**.
- Por ejemplo, los datos 10111 pueden tratarse como el polinomio $x^4 + x^2 + x^1 + x^0$

- A estos bits de datos se le añaden **r** bits de redundancia de forma que el polinomio resultante sea divisible por el polinomio generador. El receptor verificará si el polinomio recibido es divisible por **G(X)**. Si no lo es, habrá un error en la transmisión.

- Los bits de datos se dividen en bloques (llamados frames en inglés), y a cada bloque se le calcula **r**, que se denomina secuencia de comprobación de bloque (Frame Check Sequence, FCS, en inglés)

-

- **Form error:** si un nodo detecta un bit dominante en uno de los cuatro segmentos, incluyendo el fin de bloque, espacio interframe, delimitador Acknowledge o delimitador CRC, entonces ocurrió un Form error y se genera un bloque de error. El mensaje es repetido.

-

- **Oscilador (cristal):** es aquel oscilador que incluye en su realimentación un resonador piezoeléctrico.

-

- **PIC:** es un micro controlador.
-
- **Stuff bit error:** si, dentro del bloque de comienzo y el delimitador CRC, seis bits consecutivos son detectados con la misma polaridad, se viola la regla del bit stuffing. Por lo que ocurre un Stuff bit error y se genera un bloque de error. El mensaje es repetido.
- **Transceiver:** es un transmisor/receptor de señales.

- Assembler: Compilador que traduce instrucciones de lenguaje ensamblador a instrucciones en lenguaje de máquina.
- bps, kbps: Bits por Segundo, o Kilo Bits por Segundo: se refiere a la cantidad de bits que pueden ser transmitidas en 1 segundo.
- CANbus: Significa "Controller Area Network". Es una red para microcontroladores desarrollada por Bosch para el uso en automóviles. Es muy robusto y fue originalmente diseñado para uso en ambientes con mucho ruido electromagnético.
- Ethernet: Estándar desarrollado para el intercambio de mensajes entre computadoras de una red de área local (LAN) utilizando cable coaxial, fibra óptica o cables de par trenzado.
- Microcontrolador: Dispositivo que integra unidades de memoria de programa, memoria de datos, periféricos, y una unidad central de procesamiento en un solo chip.
- PIC: Microcontrolador fabricado por la empresa Microchip. Se puede encontrar en versiones desde 8 bits hasta 16. Significa "Peripheral Interface Controller"
- Robot: Máquina que puede realizar tareas, usualmente realizadas por humanos, automáticamente. Principalmente utilizados para realizar tareas repetitivas en líneas de producción.
- Serial Tunnel: Dispositivo que encapsula datos recibidos por el puerto serial en datagramas TCP/IP. Estos datagramas pueden ser enviados ya sea por Ethernet, o 802.11b/g.
- Teach-In: Método de programación de robots industriales por el cual manualmente se mueve el robot al lugar que se requiere, y después se guardan sus coordenadas.
- WLAN: Se refiere a una red de área local inalámbrica (Wireless Local Area Network).

- EEPROM: Memoria que puede ser borrada por impulsos eléctricos. Tiene la ventaja de guardar información incluso si se le quita la alimentación.

B. PIC18F458

Este es un microcontrolador de la familia de 16 bits. Tiene las siguientes especificaciones.:

RISC CPU de alto desempeño:

- Velocidad de funcionamiento: 40 MHz, hasta 10 MIPS
- Voltaje de funcionamiento: 4.2-5.5V
- Gama de temperaturas industrial (-40° a +85°C)
- Direccionamiento linear de memoria de programa, hasta 2 MBYTES
- Direccionamiento linear de memoria de datos, hata 4 kilobytes
- 4-10 MHz oscilador/Reloj con PLL activo
- Instrucciones amplias 16-bit, ancho de datos 8-bit
- Capacidad de la interrupción (21 fuentes de la interrupción) con los niveles de la prioridad

Funciones especiales:

- Memoria Flash: 32 Kbytes (16,384 words)
- Datos SRAM: 1536 bytes
- Datos EEPROM: 256 bytes
- Opciones configurables de oscilador, incluyendo:
    - 4× Phase Lock Loop de oscilador Primario
    - Oscilador secundario (32 kHz)

Funciones Analógicas:

- Convertidor Analógico-Digital 10-bit
    - Se hace Conversión durante Modo Sleep
    - Hasta 8 canales disponibles
- Módulo de Comparador Analógico
    - Multiplexación Programable de Entrada Salida

Modulo de Voltaje de Referencia Comparador