UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Reingeniería de Megaproyectos Fase I

Tomo II

Trabajo de graduación en modalidad de Megaproyecto presentado por los estudiantes Erick Iván Hernández Woc, Johnny Omar del Cid Guzmán, José Javier Mérida Rodríguez para optar al grado académico de Licenciados en Ingeniería Mecatrónica; y Otto Guillermo Wantland Conde, Vidal Roberto Villegas Zabala, William Abel Orozco Cifuentes, Rony Andrés Ajtún Bulux para optar al grado académico de Licenciados en Ingeniería Electrónica.

Guatemala,
2017

Reingeniería de Megaproyectos Fase I

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

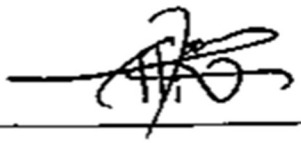
Reingeniería de Megaproyectos Fase I

Tomo II

Trabajo de graduación en modalidad de Megaproyecto presentado por los estudiantes Erick Iván Hernández Woc, Johnny Omar del Cid Guzmán, José Javier Mérida Rodríguez para optar al grado académico de Licenciados en Ingeniería Mecatrónica; y Otto Guillermo Wantland Conde, Vidal Roberto Villegas Zabala, William Abel Orozco Cifuentes, Rony Andrés Ajtún Bulux para optar al grado académico de Licenciados en Ingeniería Electrónica.

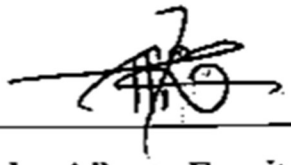
Guatemala,
2017

Vo. Bo.:

(f) 

Ing. Carlos Alberto Esquit Hernández

Director

(f) 

Ing. Carlos Alberto Esquit Hernández

Fecha de aprobación:

Guatemala, 27 de noviembre de 2017

PREFACIO

Este trabajo se llevó a cabo en la modalidad de megaproyecto con tal de aplicar los conocimientos multidisciplinarios adquiridos durante la carrera para resolver problemas de la vida real y darle un uso práctico para la proposición de alternativas distintas a métodos convencionales. Por medio de estos proyectos se puede observar y experimentar la cercana relación que tienen distintos ámbitos científicos y tecnológicos para el desarrollo de soluciones eficientes.

Ésta iniciativa comenzó por parte del Departamento de Ingeniería Electrónica y Mecatrónica con la idea de retomar proyectos realizados en otros años, los cuales han sido descuidados, y restaurar su funcionamiento, así mismo, optimizarlos para mejorar su uso en presentaciones públicas y la difusión de procesos de ingeniería para la resolución de problemas.

Se agradece a la Universidad del Valle de Guatemala por permitirnos ser parte de esa fase; a nuestros profesores, por enseñarnos las herramientas necesarias para obtener el éxito dentro y fuera de la Universidad; a nuestros asesores, quienes con paciencia y arduo trabajo nos orientaron en el desarrollo de este trabajo.

ÍNDICE

Página

LISTA DE TABLAS

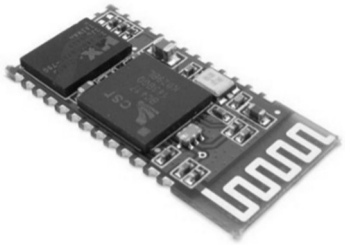
LISTA DE FIGURAS

VIII. SWARM ROBOTICS

A. DESARROLLO DEL MÓDULO DE COMUNICACIÓN ELECTRÓNICA

1. Selección del módulo de comunicación. El primer paso para el diseño del sistema de comunicación utilizado por los robots fue la selección de un módulo de comunicación inalámbrica que cumpliera con los requisitos generales del proyecto. Primordialmente se buscaba un equipo que fuera rápido, estable, capaz de manejar equipos tanto individuales como miembros de un enjambre y de costo reducido para poder comprarse en grandes cantidades. Las dos opciones principales que se tomaron en cuenta fueron el módulo HC-05 de comunicación Bluetooth y el módulo ESP-8266 de comunicación Wi-Fi.

Figura 201. Módulo HC-05 de comunicación inalámbrica con protocolo Bluetooth.



[52]

Figura 202. Módulo ESP-8266 de comunicación inalámbrica con protocolo Wi-Fi.

[118]

Estos dos módulos sobresalieron de los demás debido a su bajo costo, así como la variedad de recursos que existen para su aplicación en internet. El primer tema por evaluar fue el de la velocidad de transmisión de los equipos. Dado que se espera utilizar los robots como sistemas de robótica capaces de

ejecutar algoritmos de control de múltiples complejidades se requiere que el sistema de transmisión cuente con la taza de transferencia de datos más alta posible, asegurando así un control fluido y efectivo. En esta primera área se investigaron ambos protocolos y se encontró que el protocolo Wi-Fi es capaz de transmitir información a una taza ideal de 54mbps mientras que el protocolo Bluetooth solamente es capaz de alcanzar tazas de transmisión de 1mbps. [61]

Tabla XLII. Comparación de los protocolos de Bluetooth, UWB, ZigBee, Wi-Fi.

Standard	Bluetooth	UWB	ZigBee	Wi-Fi
IEEE spec.	802.15.1	802.15.3a *	802.15.4	802.11a/b/g
Frequency band	2.4 GHz	3.1-10.6 GHz	868/915 MHz; 2.4 GHz	2.4 GHz; 5 GHz
Max signal rate	1 Mb/s	110 Mb/s	250 Kb/s	54 Mb/s
Nominal range	10 m	10 m	10 - 100 m	100 m
Nominal TX power	0 - 10 dBm	-41.3 dBm/MHz	(-25) - 0 dBm	15 - 20 dBm
Number of RF channels	79	(1-15)	1/10; 16	14 (2.4 GHz)
Channel bandwidth	1 MHz	500 MHz - 7.5 GHz	0.3/0.6 MHz; 2 MHz	22 MHz
Modulation type	GFSK	BPSK, QPSK	BPSK (+ ASK), O-QPSK	BPSK, QPSK COFDM, CCK, M-QAM
Spreading	FHSS	DS-UWB, MB-OFDM	DSSS	DSSS, CCK, OFDM
Coexistence mechanism	Adaptive freq. hopping	Adaptive freq. hopping	Dynamic freq. selection	Dynamic freq. selection, transmit power control (802.11h)
Basic cell	Piconet	Piconet	Star	BSS
Extension of the basic cell	Scatternet	Peer-to-peer	Cluster tree, Mesh	ESS
Max number of cell nodes	8	8	> 65000	2007
Encryption	E0 stream cipher	AES block cipher (CTR, counter mode)	AES block cipher (CTR, counter mode)	RC4 stream cipher (WEP), AES block cipher
Authentication	Shared secret	CBC-MAC (CCM)	CBC-MAC (ext. of CCM)	WPA2 (802.11i)
Data protection	16-bit CRC	32-bit CRC	16-bit CRC	32-bit CRC

* Unapproved draft.
• Acronyms: ASK (amplitude shift keying), GFSK (Gaussian frequency SK), BPSK/QPSK (binary/quardrature phase SK), O-QPSK (offset-QPSK), OFDM (orthogonal frequency division multiplexing), COFDM (coded OFDM), MB-OFDM (multiband OFDM), M-QAM (M-ary quadrature amplitude modulation), CCK (complementary code keying), FHSS/DSSS (frequency hopping/direct sequence spread spectrum), BSS/ESS (basic/extended service set), AES (advanced encryption standard), WEP (wired equivalent privacy), WPA (Wi-Fi protected access), CBC-MAC (cipher block chaining message authentication code), CCM (CTR with CBC-MAC), CRC (cyclic redundancy check).

[61]

Esto se vio reflejado en la información obtenida para cada módulo, con el módulo ESP-8266 ofreciendo un Baud Rate máximo de 115200bps sin problemas mientras que el módulo HC-05 solamente ofrecía un Baud Rate de 9600bps de manera estándar . El segundo punto clave fue la seguridad de envío de información; esto debido a que el sistema de control depende de una multitud de pequeñas instrucciones, de las cuales la pérdida de una o más puede tener efectos severos sobre la implementación de un algoritmo. En esta área nuevamente se veía favorecido el módulo ESP-8266 gracias en parte al hecho de que la conexión Wi-Fi ofrece sistemas de revisión de errores y protección de datos de 32 bits mientras que los sistemas Bluetooth solo son capaces de manejar 16 bits como se puede ver en la tabla XXIX. Asimismo, el módulo ESP-8266 ofrece la flexibilidad de trabajar con cualquiera de dos protocolos, TCP o UDP, proveyendo mayor flexibilidad para el usuario. Finalmente, las últimas áreas a evaluar para ambos módulos fueron sus capacidades para manejar enjambres y su precio. Tanto el módulo HC-05 como el módulo ESP-8266 ofrecen

la capacidad de conectarse a un servidor central y funcionar como miembros de un grupo colectivo por lo que no sobresalió ninguno de los dos. Por otro lado, en cuanto a precio el módulo HC-05 se encontraba a la venta por $5 en la mayoría de vendedores y distribuidores en línea mientras que el ESP-8266 estaba valorado en tan solo $3. Gracias a esta discrepancia en su precio, así como los otros factores favorables al módulo ESP-8266 este fue elegido como el sistema de comunicación para los robots del proyecto. [35] [45]

2. Selección del protocolo de comunicación. Con el módulo de comunicación seleccionado se prosiguió a trabajar en la selección del protocolo de comunicación indicado para el proyecto. Los dos protocolos trabajables con el módulo ESP-8266 son el protocolo TCP y el protocolo UDP. Se comenzó realizando una investigación acerca de los beneficios y desventajas de cada uno de estos módulos y como se vería afectado el funcionamiento de los sistemas al trabajarse con cada uno. Como se describe en el marco teórico de este trabajo ambos protocolos se encargan del envío y recibimiento de datos a través de aplicaciones. La mayor diferencia entre ambos protocolos radica en la función a la que le asignan una mayor prioridad con el protocolo TCP eligiendo estabilidad y seguridad de transmisión sobre velocidad mientras que UDP se enfoca en una transmisión veloz pero sujeta a una mayor pérdida de paquetes. Esto se ve aún más notablemente en transmisiones de multimedia donde la característica de UDP de no retransmitir paquetes perdidos causa una notable pérdida en la calidad de la imagen mientras que TCP maneja una imagen de alta calidad, aunque su taza de transmisión es bastante reducida. [140]

Tabla XLIII. Comparación de protocolos TCP y UDP

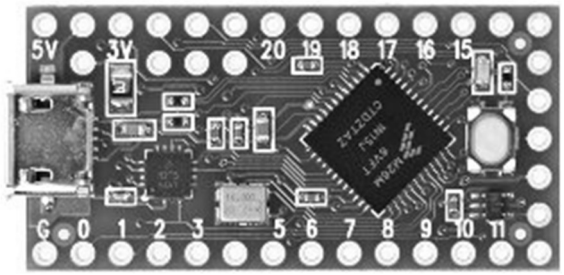
Característica	TCP	UDP
1	Confiable	No confiable
2	Provee control de errores y control de flujo.	No cuenta con control de errores ni control de flujo.
3	Conexión segura con apretón de manos de tres vías.	No ofrece ninguna seguridad de conexión.
4	Segmentos enviados en secuencia ordenada.	Segmentos enviados sin ningún orden.
5	Retransmite paquetes perdidos.	No retransmite ningún paquete.
6	Velocidad reducida de transmisión.	Alta velocidad de transmisión de paquetes.

[Elaboración propia]

Conociendo la necesidad de que el sistema sea tanto veloz como confiable se decidió utilizar el protocolo TCP para poder asegurar que el envío de instrucciones fuera recibido sin problemas por el equipo, aún si no se logra alcanzar la velocidad máxima posible con el módulo. Esta decisión se debe a que, ya que el equipo será utilizado como una herramienta didáctica, se consideró que era necesario establecer un método de comunicación seguro y confiable que evitara la pérdida de paquetes al máximo para que en un futuro estudiantes que utilicen la herramienta se sientan confiados en que cualquier error que pueda presentarse se puede atribuir a su diseño y no a la pérdida de información por parte del módulo de comunicación.

3. Diseño y prueba preliminar. Una vez se eligió el protocolo de comunicación indicado para el funcionamiento del equipo se comenzó a trabajar en el código encargado de manejar la conexión entre el módulo y Matlab. Se comenzó diseñando las funciones necesarias para poder establecer una conexión entre el microcontrolador utilizado en el robot y la sesión de Matlab. El robot comenzó trabajándose con el microcontrolador Teensy LC, elegido gracias a su bajo costo y su alta eficiencia.

Figura 203. Microcontrolador Teensy LC.



[95]

Este microcontrolador se eligió debido a que proveía una velocidad de operación alta, a 48MHz, además de una variedad de puertos para múltiples tipos de conexiones. Asimismo, el Teensy LC ofrecía la posibilidad de programarse en Arduino, permitiendo reducir la complejidad del código utilizado para controlarlo. Además de esto el Teensy LC también contaba con un costo reducido para un microcontrolador de este tipo, con cada componente listado en $12 en línea. La siguiente tabla muestra cómo se compara este microcontrolador con el siguiente modelo de la línea, el Teensy 3.2.

Tabla XLIV. Comparación entre microcontroladores tipo Teensy

Feature	Teensy LC	Teensy 3.2	Units
Price	$11.65	$19.80	US Dollars
Processor Core Rated Speed Overclockable	MKL26Z64VFT4 Cortex-M0+ 48 -	MK20DX256VLH7 Cortex-M4 72 96	 MHz MHz
Flash Memory Bandwidth Cache	62 96 64	256 192 256	kbytes Mbytes/sec Bytes
RAM	8	64	kbytes
EEPROM	1/8 (emu)	2	kbytes
Direct Memory Access	4	16	Channels
Digital I/O Voltage Output Current Output Voltage Input	27 3.3V + one 5V 5mA + four 20mA 3.3V Only	34 3.3V 10mA 5V Tolerant	Pins Volts milliAmps Volts
Analog Input Converters Resolution Usable Prog Gain Amp Touch Sensing Comparators	13 1 16 12 0 11 1	21 2 16 13 2 12 3	Pins Bits Bits Pins
Analog Output DAC Resolution	1 12	1 12	Pins Bits

[96]

Se investigó la posibilidad de utilizar una librería ya existente para el manejo desde el microcontrolador, sin embargo, esto posaba dos problemas importantes. El primero fue que el uso de una librería ajena podía limitar la aplicación del módulo debido a que dictaría de forma específica las acciones que se pueden realizar con dicho módulo, problema que no existe si se realizan las acciones utilizando código diseñado personalmente. Asimismo, la librería contaba con un diseño a muy alto nivel lo cual podría reducir la velocidad de operación de nuestro microcontrolador, así como el desempeño de nuestro módulo ESP-8266. Gracias a esto se comenzó trabajando en un set de operaciones básicas para la comunicación con el módulo.

e. Establecimiento de la conexión con la red. La primera de las operaciones básicas que debían ser implementadas para el funcionamiento del módulo fue la del establecimiento de una conexión con el enrutador principal y la asignación de una dirección IP. Para poder realizar esto se comenzó investigando las instrucciones básicas necesarias para controlar el módulo ESP8266. Este trabaja a base de instrucciones AT transmitidas a través de una conexión serial. Para poder enviar estas instrucciones desde nuestro microcontrolador sin problemas se estableció una función para el microcontrolador que permitiría establecer un tiempo de espera de respuesta determinado para la conexión. Esta función, la primera de la Tabla XXXIII, permite manejar más directamente la velocidad de transmisión de los datos del equipo y previene la posibilidad de la comunicación cayendo en un lazo infinito debido a una mala transmisión. Esta trabaja revisando la información siendo procesada por el buffer del microcontrolador. La función analiza cada dato recibido durante un período de tiempo determinado y revisa si se obtuvo la cadena de "Okrn" la cual indica que el envío de información fue exitoso. Si la cadena fue recibida el programa continúa trabajando sin problemas, de lo contrario la función devuelve un 0, indicando un intento fallido.

Tabla XLV. Comandos AT reconocidos por el módulo ESP-8266

Función	Comando	Descripción	Parámetros y su descripción	
1	AT+RST	Reinicia el módulo.	N.A.	N.A.
2	AT+CWMODE = mode	Selecciona el modo de conexión.	mode	1 = Estático
				2 = AP
				3 = Ambos
3	AT+CWJAP = ssid, pwd	Conecta el módulo a la red seleccionada.	ssid	Nombre de la red.
			pwd	Contraseñar de la red
4	AT+CWLAP	Muestra un listado de las redes disponibles	N.A.	N.A.
5	AT+CIPMUX = mode	Establece el número de conexiones que acepta el servidor.	mode	0 = conexión única.
				1 = múltiples conexiones.
6	AT+CIPSERVER = mode, port	Establece al equipo como un servidor TCP	mode	0 = Cerrar servidor.
				1 = abrir servidor.
			port	número de puerto para la conexión.

Continuación tabla XLV. Comandos AT reconocidos por el módulo ESP-8266

Función	Comando	Descripción	Parámetros y su descripción	
7	AT+CIPSEND = id, length	Envía una cadena de caracteres.	id	Para puertos muiltiplexados. Identifica a qué conexión enviar los datos
			length	Largo de la cadena de caracteres que se desea enviar.
8	AT+CIPCLOSE = id	Cierra la conexión TCP.	id	Identifica a qué conexión cerrar.

[Elaboración propia]

Con la función de tiempo de espera ya integrada se pudo comenzar a trabajar en la comunicación directamente con el módulo ESP8266. La siguiente función que se trabajó fue la de establecimiento de una conexión con el enrutador. Esta función, número 2 en la Tabla XXXIII, trabaja con instrucciones AT para dirigir el funcionamiento del módulo. En primer lugar, se reinicia el módulo con la instrucción 1 de la Tabla XXXII para poder prevenir cualquier retención de información por parte del mismo debido a usos pasados. Con el reinicio terminado se prosiguió a listar las redes disponibles para el equipo y unirse a la red Wi-Fi deseada utilizando las instrucciones 4 y 3 de la Tabla XXXII respectivamente. Los valores de los parámetros de ambas funciones son definidos al comienzo del código por el encargado del robot.

Tabla XLVI. Funciones para el manejo de comunicación en el microcontrolador

Función	Comando	Descripción	Parámetros y su descripción	
1	Timeout(timeout)	Establece un tiempo de espera para las respuestas del ESP-8266	int timeout	Tiempo (en ms) que se desea esperar la respuesta.
2	setupWiFi()	Establece la conexión Wi-Fi	N.A.	N.A.

Continuación Tabla XLVI. Funciones para el manejo de comunicación en el microcontrolador

Función	Comando	Descripción	Parámetros y su descripción	
3	setupTCP()	Establece al robot como un servidor TCP para la comunicación con el ordenador	N.A.	N.A.
4	DataSend(data_data_send, largo_data_send)	Envía datos del microcontrolador hacia el ordenador.	int data_data_send	Conjunto de datos numéricos que se desean transmitir.
			int largo_data_send	Largo del listado de datos.
5	recepcionSerial()	Recibe e identifica las órdenes del ordenador	N.A.	N.A.

[Elaboración propia]

Una vez se logra realizar la conexión con el enrutador este le asigna un valor de dirección IP al módulo ESP8266, dirección necesaria para comunicarse con otros elementos dentro de la red como el ordenador que desea controlar al equipo. Para poder lograr esta conexión es necesario identificar el protocolo de comunicación a utilizar. Para el caso del robot en modo uno a uno este trabajará como un servidor TCP, recibiendo instrucciones desde el ordenador. La función encargada de este trabajo se encuentra definida en la Tabla XXXIII. Esta función comienza con el envío de la instrucción 2 de la Tabla XXXII para la selección del modo de Wi-Fi, asignando un valor de 1 al parámetro de modo para indicarle al módulo que debe trabajar con una dirección IP estática y dedicada para funcionar como servidor.

La siguiente instrucción que se envía es la de selección de multiplexación, número 5 en la tabla, con un parámetro de 0. Esta se encarga de que el módulo sepa que trabajará con una única conexión, lo cual evita que otros equipos se conecten al servidor y puedan interrumpir su funcionamiento. Finalmente, la función termina con la instrucción 6 de la Tabla XXXII para el establecimiento del módulo como un servidor TCP con un puerto abierto indicado por el valor del parámetro puerto para una conexión con el equipo.

Para el funcionamiento del equipo se eligió el puerto 8080. Esto se debe a que se necesitaba un puerto para la conexión que permitiera un acceso rápido y seguro. Se comenzó trabajando con el puerto 80 debido a su popularidad, sin embargo, este causaba conflictos al momento de recibir información debido a que es el puerto utilizado para recibir mensajes de tipo http. Al encontrar este problema se prosiguió a elegir un puerto libre que fuera exclusivo de la comunicación del robot.

Una vez el equipo funcionó como un servidor dedicado se finalizó este segmento con la implementación de una instrucción de envío de datos. Esta función, la número 4 en la Tabla XXXIII, recorre una lista de valores, los cuales concatena entre sí para crear un mensaje de tipo cadena de caracteres que puede ser transmitido por el módulo ESP8266. Para poder realizar este envío se debe obtener el largo de la cadena de caracteres que se creó, este valor numérico se utiliza como parámetro de entrada a la función 7 de la Tabla XXXII para indicarle cuántos caracteres debe transmitir. Para finalizar la transmisión la función es seguida de la cadena creada por la función 4 anteriormente.

f. Diseño preliminar de la librería de Matlab. Con el servidor establecido y listo para enviar y recibir datos se pudo proseguir con el diseño de la librería de Matlab encargada de manejar al equipo. El objetivo de esta librería es que sea un conjunto de funciones básicas esenciales para la implementación de sistemas de control de distintos niveles. Gracias a esto se buscó que la librería fuera amigable al usuario y lograra acomodar tanto a principiantes como a estudiantes que desean integrar sistemas de control más complejos. Para lograrlo se definieron un set de instrucciones esenciales que la librería debe ofrecer a los usuarios. Las dos primeras instrucciones esenciales fueron la encargada de la transmisión y el recibimiento de datos.

Tabla XLVII. Funciones integradas a la librería de Matlab

Función	Comando	Descripción	Parámetros	
1	TCPTX(message,IP,TO,port)	Envía una cadena de caracteres hacia el servidor TCP.	message	Cadena de caracteres que se desea enviar.
			IP	Dirección IP del servidor.
			TO	Tiempo de espera para la respuesta.
			port	Número de puerto para la conexión.

Continuación Tabla XLVII. Funciones integradas a la librería de Matlab

Función	Comando	Descripción	Parámetros	
2	TCPRX(IP,TO,port)	Recibe y procesa información obtenida desde el servidor TCP.	IP	Dirección IP del servidor.
			TO	Tiempo de espera para la respuesta.
			port	Número de puerto para la conexión.
3	SensorCheck(IP,TO,port)	Ordena al robot que recolecte información de sus sensores y los envíe al ordenador.	IP	Dirección IP del servidor.
			TO	Tiempo de espera para la respuesta.
			port	Número de puerto para la conexión.
4	MotorControl(DirL, VelL, DirR, VelR)	Controla el movimiento del robot a través de la velocidad y dirección de cada rueda.	DirL y DirR	1 = Movimiento hacia el frente.
				0 = Movimiento en retroceso.
			VelL y VelR	Valores de velocidad. De 0 (máxima velocidad) a 255 (paro completo).
5	RoboIdentify(Name)	Define los valores de IP, TO y puerto para las funciones de acuerdo al robot con el que se desea trabajar.	Name	Nombre del equipo con el que se desea conectar.

[Elaboración propia]

Para manejar la conexión al servidor y el envío de datos se utilizó la librería de Matlab “tcpclient”, la cual permite fácilmente conectarse a otros dispositivos utilizando el protocolo TCP. Para el envío de datos se programó la primera función de la Tabla XXXIV. Esta función recibe la dirección IP y el puerto del servidor al que deseamos conectarnos, así como los datos que se desea enviar y el tiempo de espera deseado para cada respuesta. Estos datos son utilizados en conjunto con la función de escritura de la librería para comunicación TCP para transmitir la información deseada. La segunda operación básica necesaria para la comunicación fue la de recibimiento de datos. Para encargarse de esto se diseñó la función 2 de la Tabla XXXIV la cual, de forma similar a la función de envío de datos, recibe los valores de la dirección IP y puerto del servidor, así como el tiempo de espera que se desea. Utilizando estos datos y la función de escritura de la librería de comunicación de Matlab se crea una conexión con el servidor TCP. Con la conexión establecida la función analiza si existe algún valor en el buffer de transmisión, de ser así toma este valor y lo convierte en un valor numérico que Matlab es capaz de trabajar. Por otro lado, si la función no encuentra un valor en el buffer dentro del tiempo establecido cierra la conexión y devuelve un valor de 99999 para indicar una falla en el recibo.

Una vez se terminaron las operaciones de transmisión y recepción de datos se continuó con la implementación de funciones útiles más complejas. La primera de estas fue la función 3 de la Tabla XXXIV. Esta función se encarga de tomar la información de cada uno de los seis sensores ultrasónicos que componen la falda de sensores del robot. Además de esto la función trabaja creando una lista vacía de valores para albergar los datos que se le pedirán al robot. Después repite seis veces la toma de datos de nuestro sistema utilizando la función 2. Los datos obtenidos a través de esta operación pueden ser entregados al usuario convertidos a un formato procesable por Matlab, la forma estándar de la función, o pueden obtenerse sin convertir permitiendo al usuario el manejo de datos en bruto para aplicaciones que los requieran.

Durante esta etapa se realizó la primera prueba de velocidad del sistema buscando encontrar que tan rápida era la taza de transmisión máxima entre la falda de sensores y el ordenador. Para esto se realizó un programa compuesto por un lazo cerrado de repetición que utilizaría la función 3 de la Tabla XXXIV para pedir datos de la falda de sensores de forma constante. La velocidad de operación se calculó en base a los valores de los retrasos necesarios entre cada repetición de la instrucción. Esta primera prueba demostró que el equipo era capaz de transmitir la información de la falda de sensores con una mínima cantidad de errores con un retraso de 10ms entre cada repetición, equivalente a una frecuencia de 100Hz. Esta velocidad superó las expectativas del sistema hasta este momento y se planteó como el valor ideal de funcionamiento. Sin embargo, la adición de sistemas extras al microcontrolador afectó considerablemente este valor.

La segunda función necesaria para comenzar a realizar pruebas con el diseño preliminar del robot fue una función de control de los motores. Esta es la cuarta función de la Tabla XXXIV y se encarga de dictar la velocidad y dirección de las ruedas que manejan al robot. Esta función enviaba las ordenes a ambas ruedas al mismo tiempo y trabajaba a base de números específicos para indicar la dirección en la que

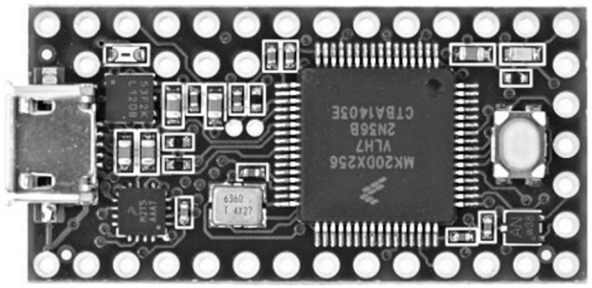
se deseaba mover al robot. La función podía recibir los números 1, 2, 3 y 4 asignado a las direcciones de adelante, atrás, izquierda y derecha respectivamente. Este sistema probó ser efectivo para las pruebas preliminares del control del equipo.

g. Comunicación entre el equipo y Matlab. Con ambas funciones de control y de toma de datos preparadas se buscó finalizar esta etapa con el diseño de una función capaz de recibir los mensajes enviados por el usuario en Matlab y actuar acorde a las órdenes recibidas. Para esto se definió la función 5 de la Tabla XXXIII. La función de recepción serial se utiliza cada vez que el microcontrolador detecta una señal proveniente de la conexión serial, indicando que el módulo ESP8266 está comunicándose con él. La primera dificultad de esta etapa fue la de lograr discernir entre la acción que se desea realizar. Para indicar que se desea trabajar con los motores se envía un mensaje desde Matlab que comienza con la palabra "motores" seguida de los datos relevantes para el funcionamiento de los mismos. Esto permite fácilmente discernir este tipo de orden, sin embargo, la instrucción de toma de datos de sensores fue más difícil debido al hecho de que la función de toma de datos no porta normalmente una palabra clave.

Se intentó resolver esto transmitiendo primero un mensaje con una palabra clave, seguido de un mensaje con la orden de abrir el canal de comunicación y enviar los datos esperados. Sin embargo, esta técnica conlleva múltiples problemas. En primer lugar, el envío de este mensaje preliminar retrasa al sistema debido a que el servidor debe cerrar su conexión antes de procesar cualquier mensaje por lo que necesitaría cerrar su conexión, procesar el mensaje recibido y volver a entablar la misma conexión para poder enviar los datos. Esto probó ser un serio problema para la efectividad del módulo por lo que se desechó. En vez de eso se analizó la señal enviada por el Módulo ESP8266 naturalmente cada vez que se le pedía que enviara datos hacia el cliente. Este análisis reveló que toda conexión realizada por el módulo a través de la conexión serial venía acotado por la palabra "CONNECT". Gracias a esto se integró un código de análisis de texto a la función para analizar el texto recibido; todo aquel que no contuviera la palabra "Motores", pero sí tuviera "CONNECT" se podía catalogar como una orden de pedida de información. Con esto se logró manejar ambas ordenes fácilmente sin reducir la velocidad de transmisión del sistema.

Con la función de manejo de ordenes implementada se realizaron las primeras pruebas con el equipo. Para las pruebas preliminares se utilizó el robot como un carro de control remoto, con las instrucciones para el movimiento siendo impartidas una por una por el controlador desde Matlab. Esta prueba causó problemas con el equipo ya que el microcontrolador elegido para el robot hasta este momento no fue capaz de correr el código completo y comunicarse con Matlab. Durante esta sesión de pruebas se descubrió que la memoria del Teensy LC no contaba con la capacidad necesaria para procesar el código del manejo del equipo y se quedaba atorado en un ciclo sin fin. Esto reveló la necesidad de un microcontrolador mejor equipado para manejar las necesidades de procesamiento del código, por lo que se decidió utilizar el siguiente modelo de la línea, el Teensy 3.2.

Figura 204. Teensy 3.2



[96]

Esta primera prueba demostró que el equipo era capaz de funcionar sin problemas, procesando las ordenes y reaccionando a ellas de forma rápida y precisa. Sin embargo, para poder medir de mejor manera el tiempo de operación del equipo se diseñó un algoritmo de control simple. Este consistió en analizar los datos de los sensores ultrasónicos para detectar un obstáculo frente al robot. De detectar dicho obstáculo el algoritmo le indicaría al robot que debe detenerse y esperar a que el obstáculo sea removido. Este código era simple pero efectivo en probar cada parte del funcionamiento del robot de manera rápida. Para determinar la velocidad de operación se tomaron en cuenta los tiempos de los retrasos utilizados entre cada operación para obtener un funcionamiento constante del equipo.

Esta prueba reveló un serio problema con el funcionamiento del equipo en ese momento. En primer lugar, este requería de retrasos de 1 segundo entre cada instrucción enviada dentro del algoritmo para poder trabajar sin problemas por lo que la toma de decisiones era lenta e impráctica. Retrasos de tiempo más pequeños ocasionaban una pérdida de paquetes de casi 90%, volviendo inútil al sistema. Estos retrasos se atribuyeron en gran parte al sistema de control de los motores del equipo, dado que la falda de sensores era capaz de trabajar con retrasos de solamente 0.1 segundos entre instrucciones. Por otro lado, los motores tendían a actuar de formas imprevistas al recibir múltiples ordenes dentro de un rango de tiempo reducido y, si bien eran capaces de trabajar correctamente con la función de control actual era necesario independizar el control de cada rueda para permitir un control más adecuado del equipo.

4. Reducción de retrasos y pruebas finales. Tomando en cuenta lo aprendido sobre el funcionamiento del equipo en la fase anterior se finalizó el trabajo buscando mejorar el sistema en cuanto a su velocidad de reacción y procesamiento. Para esto se buscó la manera de reducir la cantidad de código procesado por el microcontrolador en sí, buscando relegar el trabajo de procesamiento de datos directamente a Matlab. En primer lugar, se comenzó removiendo las funciones que se encontraban dentro de interrupciones del sistema y reemplazándolas a través de la utilización de banderas en el código. Dichas banderas serían activadas de acuerdo con ciertas interrupciones del microcontrolador, como la detección de una señal en su puerto serial. Este cambio ayudó a reducir el retraso en las acciones del microcontrolador.

Asimismo, el código de manejo de sensores y motores se simplificó, eliminando todo procesamiento de datos que se realizaba anteriormente. En la primera entrega ambas funciones tomaban los datos obtenidos por sus sensores y realizaban operaciones aritméticas para convertirlos en valores procesados y comprensibles que eran enviados a Matlab. Estas acciones ocasionaban retrasos debido a que ocupaban recursos de procesamiento para realizar acciones complejas como divisiones y multiplicaciones de números reales. Al eliminar estos procesos el equipo es capaz de dedicarse completamente a tomar datos y enviarlos cuando se le piden, aumentando la velocidad de transmisión de información del equipo.

La función de motores también se actualizó, con la decisión de dirección y velocidad separada para cada motor. Esto se debe a que independizar cada rueda permite al usuario tener un mayor control sobre el funcionamiento del equipo, dictando con mayor precisión el movimiento que se espera que realice el robot. Sin embargo, este cambio ocasionó problemas en el funcionamiento del robot, debido a que se comenzó implementando un sistema en el que la función de control de motores enviaría los datos de un motor seguido por los datos del segundo motor. Esta manera de transmisión y control ocasionó una discrepancia entre ambos motores, con el derecho actuando antes que el izquierdo y causando problemas. Durante las pruebas el equipo tendía a chocar o moverse de manera extraña cuando se le ordenaba que parara gracias a que una rueda se detenía antes que la siguiente. Este problema fue fácil de arreglar ya que solamente se revisó la función de control de motores en el microcontrolador para que fuera capaz de recibir los datos para ambos motores al mismo tiempo. Al recibir ambos datos la función asignaría instrucciones a cada motor al mismo tiempo, eliminando la discrepancia que se tenía.

Con las actualizaciones realizadas al código se decidió tomar nuevamente mediciones de velocidad del sistema. Nuevamente se trabajó con un algoritmo básico de control para poner a prueba el funcionamiento completo del equipo. Este consistió en utilizar la información provista por la falda de sensores para identificar un obstáculo frente al robot y detener al mismo antes de que sufra una colisión.

Las pruebas fueron exitosas, con el robot respondiendo de manera más rápida y fluida a la presencia de distintos obstáculos en su camino y siendo capaz de tomar decisiones con más velocidad que antes. De nuevo se utilizaron los tiempos de retraso entre cada instrucción como medida del tiempo total de operación del equipo. Para el funcionamiento efectivo y constante del robot el mínimo tiempo de retraso posible entre

instrucción fue de 0.3 segundos, como se indica en la Tabla XXXV. Esto indica una frecuencia de operación de 3.33Hz por instrucción para la primera iteración del proyecto. Si bien es posible reducir estos tiempos de retrasos aún más en el futuro este valor ya es suficientemente rápido para comenzar a aplicar sistemas de control y robótica de bajo nivel.

Tabla XLVIII. Resultados obtenidos en las pruebas realizadas

Prueba	Descripción	Tiempo de retrasos (s)	Instrucciones recibidas (%)	Instrucciones perdidas (%)
1	Lazo de repetición de pedida de datos a falda de sensores.	1	100	0
		0.1	95	5
		0.01	80	20
		0.001	65	35
2	Algoritmo básico de control. Detección de obstáculo y control de motores para prevenir colisión.	1	90	10
		0.1	60	40
		0.01	35	65
		0.001	20	80
3	Algoritmo básico de control. Detección de obstáculo y control de motores para prevenir colisión con funciones revisadas.	1	100	0
		0.3	90	10
		0.01	60	40
		0.001	40	60

[Elaboración propia]

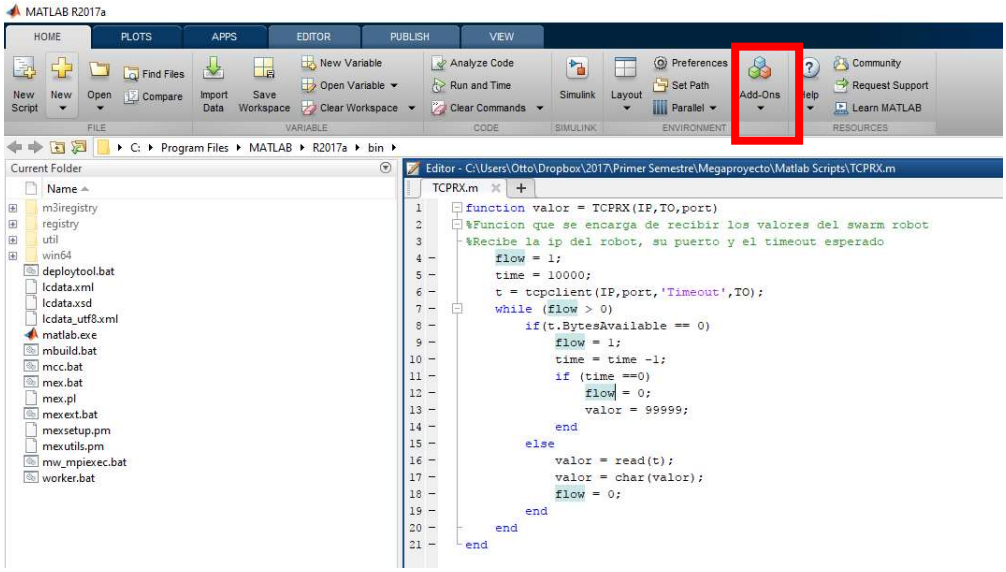
Asimismo, la seguridad de la transmisión fue otro de los objetivos principales del módulo, buscando proveer un sistema capaz de perder una mínima cantidad de instrucciones. El equipo aún presentó algunos problemas en este sentido. Durante las pruebas finales se notó que el equipo cuenta con un rango de tiempo durante el cual la pérdida de paquetes de instrucciones aumenta a casi 80%, con la mayoría de instrucciones para motores siendo ignoradas por el equipo. Este rango de tiempo dura un promedio de 2 minutos desde que se enciende el equipo y se establece como servidor TCP. Sin embargo, al pasar esta etapa la cantidad de paquetes perdidos baja a alrededor de 10%, y aún con estos el equipo es capaz de continuar trabajando con la última información recibida.

Para finalizar se trabajó en la creación de una caja de herramientas de Matlab, lo cual permitirá que los códigos utilizados en las pruebas y demás puedan instalarse y utilizarse sin problemas en cualquier computador. El uso de una caja de herramientas es la mejor manera de compartir y trabajar con librerías de

funciones en Matlab debido a que se puede compartir fácilmente el instalador creado para obtener todas las funciones y los archivos adjuntos a estas sin preocuparse por los requerimientos de cada programa separado.

Para crear esta caja de herramientas se realizaron los siguientes pasos en Matlab. Primero se ubicó el área de “Environment” en la pestaña de “Home” como se muestra en la siguiente figura.

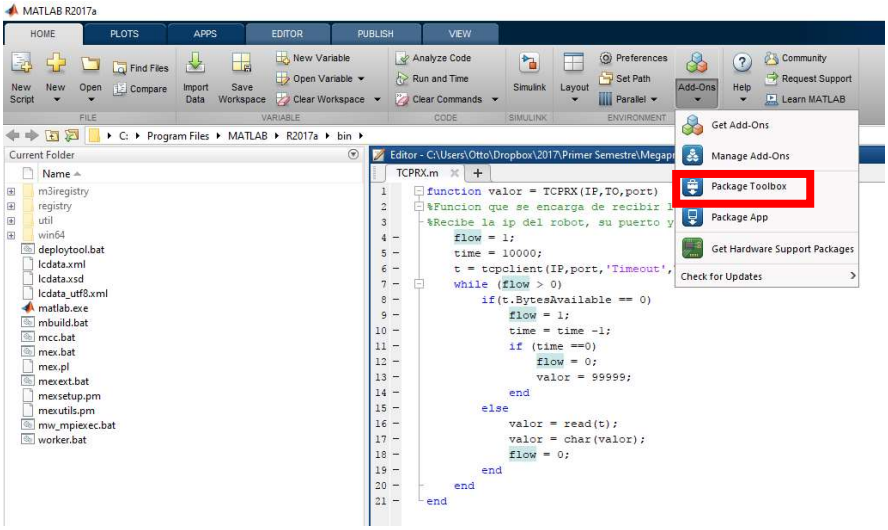
Figura 205. Primer paso para la creación de la caja de herramientas



[Elaboración propia]

En esta área se selecciona el botón de “Add-Ons” y se busca la opción de “Package Toolbox” como se muestra a continuación.

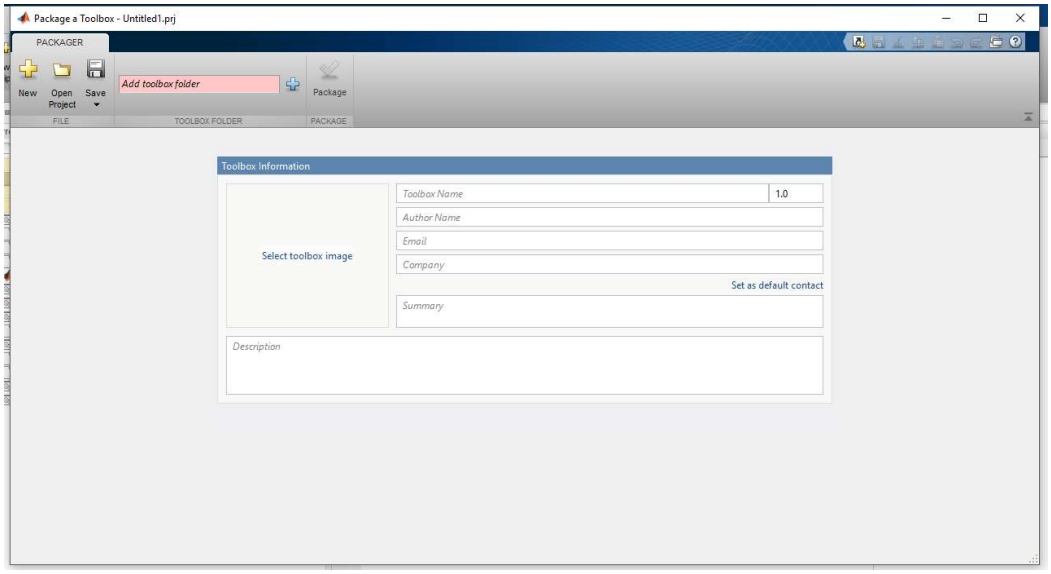
Figura 206. Ubicación de la opción de crear la caja de herramientas en Matlab



[Elaboración propia]

Seleccionando el botón indicado se pasará a un área para definir las características de la caja de herramientas, incluyendo el nombre de esta y la información de contacto de los desarrolladores. Además de esto se puede colocar una imagen y una corta descripción de lo que los sets de funciones realizan. La información detallada de cada función por separado se incluyó dentro de los comentarios iniciales de cada uno para facilitar su uso. Con la información colocada se prosigue a presionar el botón de "+" el cual permite seleccionar los archivos que se desea incluir en la caja de herramientas. Se aconseja colocar todos los archivos en un solo folder dentro de su ordenador para seleccionar este y acelerar el proceso.

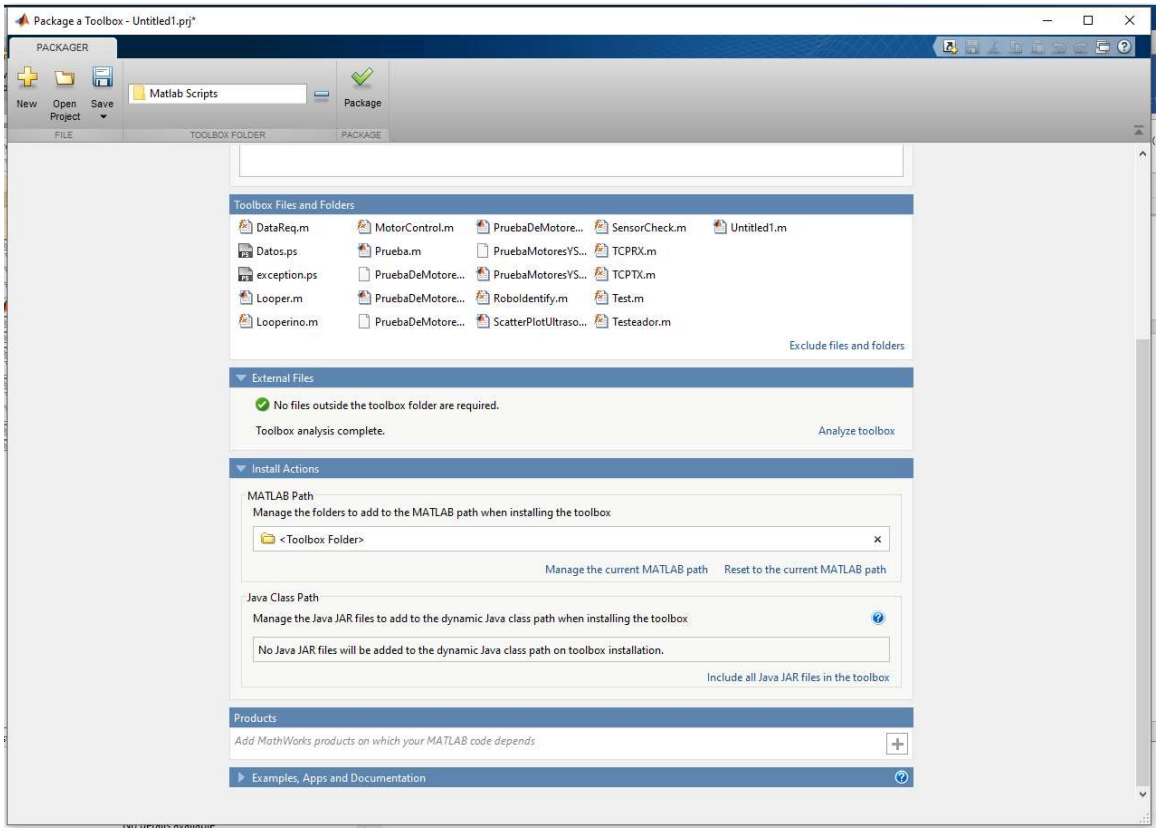
Figura 207. Pantalla para definir las características de la caja de herramientas



[Elaboración propia]

Una vez se ha seleccionado la carpeta deseada Matlab analizará los archivos para indicar si existe alguna librería externa adicional que requiera ser instalada para utilizar la caja de herramientas. En el caso de las funciones definidas en el proyecto no se requirió la instalación de ninguna librería adicional por lo que se prosiguió con el paso final. Para terminar, se puede definir el folder en el que Matlab guardará los archivos en la computadora del usuario que instale la caja de herramientas. Para este proyecto se utilizó la dirección estándar provista por Matlab y se finalizó la creación de la caja de herramientas presionando el botón de "Package".

Figura 208. Pantalla de análisis de archivos de la caja de herramientas y características finales



[Elaboración propia]

Una vez creada la caja de herramientas se obtendrá un archivo con el nombre de la caja de herramientas y con extensión ".mltbx". Este archivo se puede distribuir como se desee y cualquier usuario de Matlab será capaz de instalarlo y utilizar sus contenidos sin problemas.

B. DESARROLLO DEL MÓDULO DE DISEÑO DE ESTRUCTURA E IMPLEMENTACIÓN DE POTENCIA ELÉCTRICA PARA UN ROBOT DESTINADO A TRABAJAR EN ENJAMBRE.

1. Estructura. Existe muchas opciones para realizar una estructura para un robot destinado a trabajar en enjambre, antes de comenzar el diseño se debió tomar en cuenta los siguientes aspectos para que los cumpliera nuestro diseño.

- Fabricable con la maquinaria disponible en la universidad, refiriéndonos a la impresora 3D y la cortadora laser de MDF.
- Materiales económicos.
- Fácil de reparar.
- No utilizar ensambles que requieran pegar piezas.
- Fácil de construir.
- Debe albergar todos los componentes necesarios para el funcionamiento autónomo del robot.

a. Pieza para ultrasónicos. Para poder cumplir los requisitos anteriores se discutieron varias soluciones para la estructura y se llegó a la conclusión que la estructura seria de tipo modular *stackable* a base de PLA y MDF con 4 tornillos infinitos que serían su columna vertebral, esta solución cumple con todos los requisitos necesarios. El ensamble constara de varios módulos donde se albergarían todos los componentes necesarios. Se hablará de ellos en el orden que se desarrolló, en esta sección no se incluirá los módulos de placas electrónicas. Se decidió distribuir los componentes en los módulos de la siguiente manera.

Figura 209.Primer bosquejo de la estructura del robot.

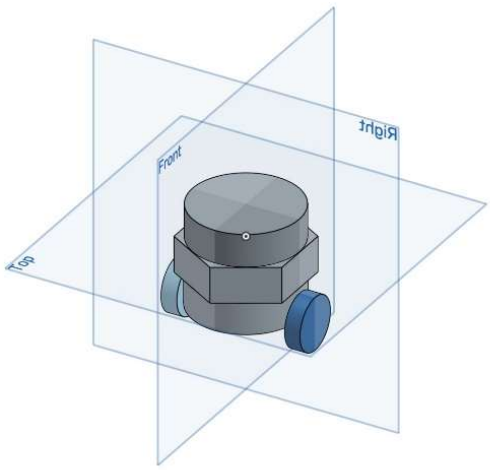
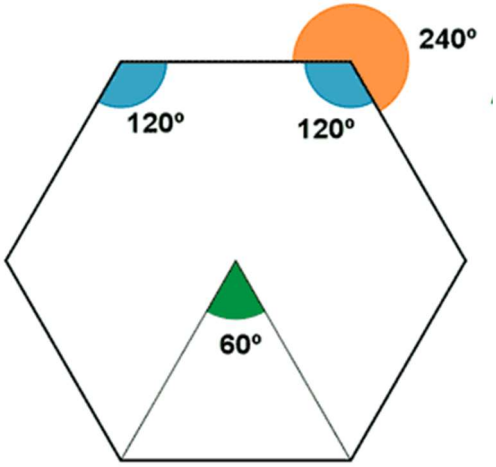


Tabla XLIX. Distribución de componentes del robot

Componentes	Módulo
6 sensores ultrasónicos	Hexágono
Switch, fusible	Porta Switch
2 motores con sus encoders ,2 baterías LiPo y Ball caster.	Placa base de MDF

Primero se desarrolló el módulo estructural que albergaría los seis sensores ultrasónicos, al ser el primero este definió el *footprint* del robot. Se tubo que encontrar una forma geométrica donde se pudiera poner los seis sensores ultrasónicos y que cada uno cubriera un intervalo de grados diferente. Con esto en mente se encontró que la mejor solución sería una forma de hexágono, de esta manera cada sensor ultrasónico cubriría 60 grados del espacio donde se encuentra el robot.

Figura 210. Hexágono con sus ángulos exteriores e interiores.



Se empezó a desarrollar una estructura en forma de hexágono que pudiera albergar los seis sensores ultrasónicos en la herramienta Autodesk Inventor. Podemos observar el proceso del diseño. Primero se tomaron medidas de los sensores para que estos encajaran en la estructura, luego se diseñó un conjunto de círculos donde estos pudieran encajar. Después se decidió de que tamaño iban a ser los lados de este hexágono, se decidió que el largo de los sensores más 4 milímetros iba a ser suficiente. Por último, se agregó en medio de la estructura los 4 soportes donde van a pasar los tornillos infinitos tamaño M3.

Figura 211. Proceso de diseño de pieza para ultrasónicos

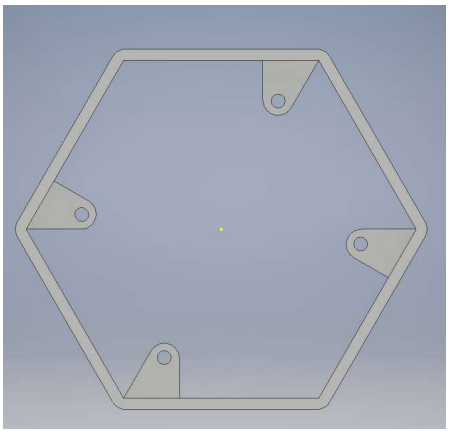


Figura 212. Proceso de diseño de pieza para ultrasónicos.

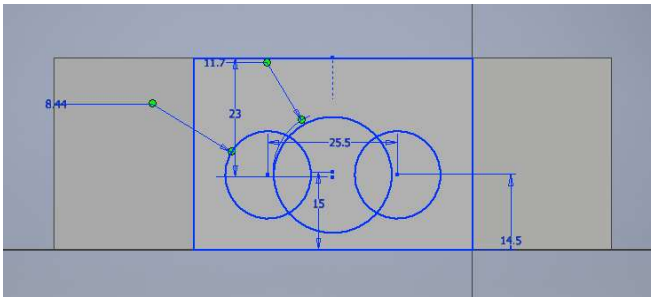


Figura 213. Proceso de diseño de pieza para ultrasónicos.

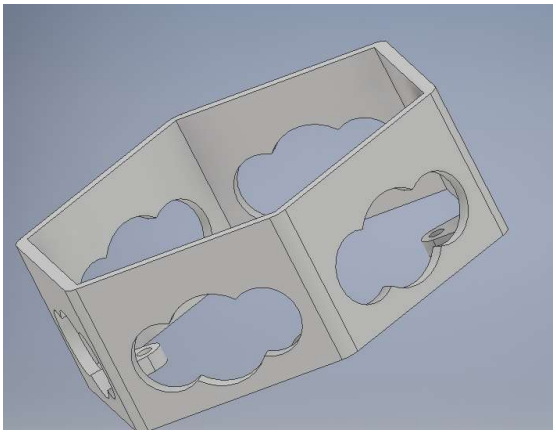
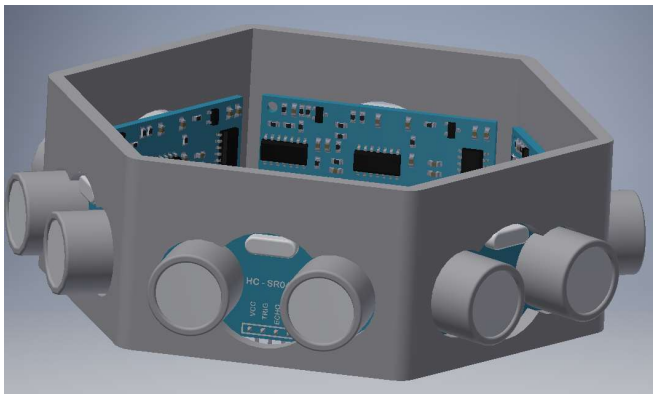
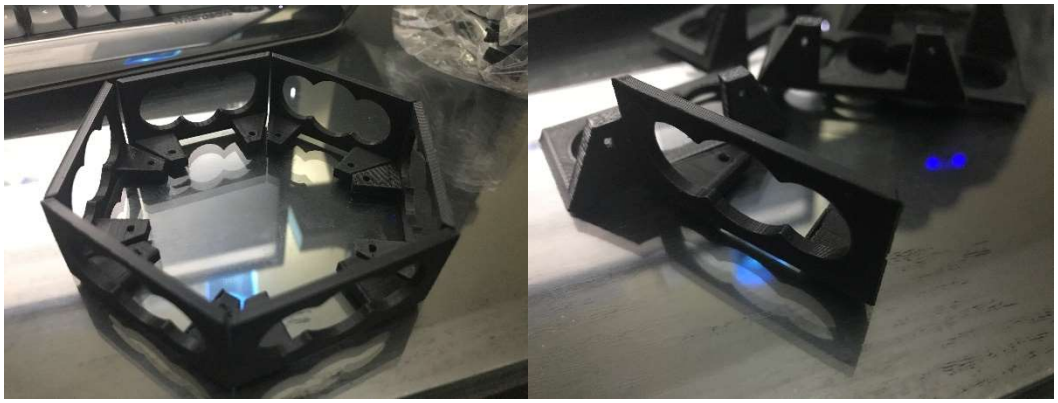


Figura 214. Proceso de diseño de pieza para ultrasónicos.



Para el módulo de hexágono se decidió fabricarlo con la impresora 3D. Debido a la precisión y las limitaciones de este método de fabricación se debió ajustar el diseño para poder ser fabricado. Se agrandaron 0.4 mm los agujeros donde iban a ir los ultrasónicos y los tornillos infinitos. Luego se probaron varias maneras de fabricar la estructura ya que por la forma de la pieza era difícil que sus agujeros salieran bien. La primera manera de ellas fue seccionando la pieza en seis y se obtuvo el resultado desplegado en la imagen inferior. Como se puede observar debido a la falta de precisión de la impresión las piezas no encajan lo suficiente bien y por lo tanto esta solución no fue viable.

Figura 215. Primer intento de fabricación de módulo hexágono.

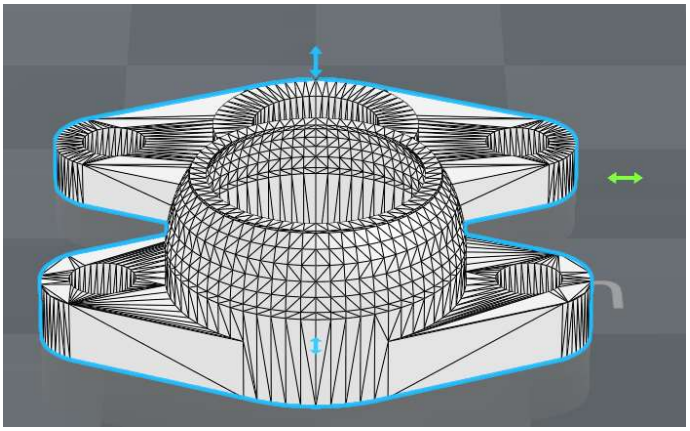


Luego se probó agregar paredes de refuerzo de 0.4 mm en los agujeros del diseño, esto sí dio un resultado positivo al imprimirlo. Ya al tener la pieza impresa se retiraron cuidadosamente con una navaja las paredes de apoyo que se le habían agregado anteriormente.

Figura 216. Segundo intento de fabricación de módulo hexágono.

b. Ball Caster. Después se desarrolló una parte específica, el Ball caster. Esta pieza suplirá la función de la rueda de en medio del robot. La pieza consta de una esfera vacía que adentro llevara una canica y una tapa que permitiría el rodamiento de ella. Nos dará la estabilidad necesaria para que no que el robot no se desbalancee. Esta parte fue sacada de la página www.tingyverse.com, solo se escalaron sus dimensiones para que esta pudiera ser colocada con tornillos M3. Esta modificación fue realizada sobre el archivo stl en el programa blender.

Figura 217. Ball Caster utilizado.



c. Porta Switch y fusible. Luego se desarrolló el módulo que portara el Switch y el fusible, este fue algo sencillo ya que se utilizó como anclaje dos de los tornillos que atraviesan la estructura por medio. Lo único que se tuvo que tomar en cuenta es que el Switch tenía que estar en cierta posición para que este módulo luego encajara al ras con la placa de potencia que se desarrollaría después. Otro detalle que también se tuvo que tomar en cuenta es que la altura de este módulo tuvo que ser la del Switch más 2 milímetros extras para que el módulo que va arriba de este encaje perfectamente.

Figura 218. Diseño de porta Switch y fusible

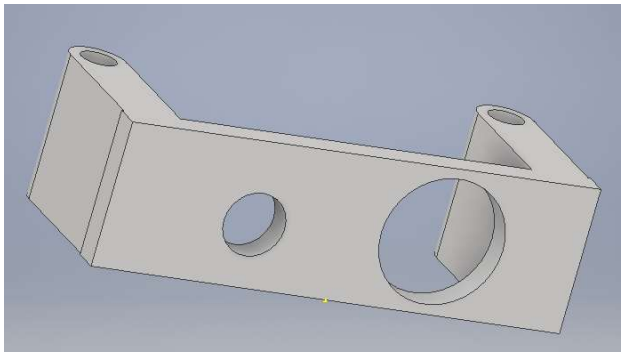
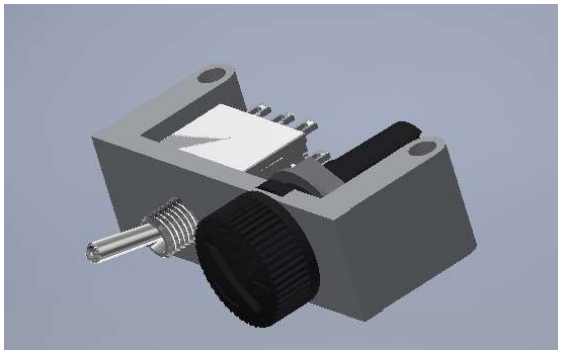
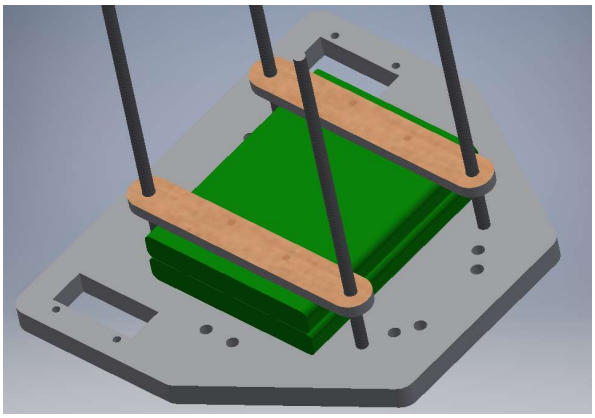


Figura 219. Porta Switch y fusible ya implementado en ensamble.



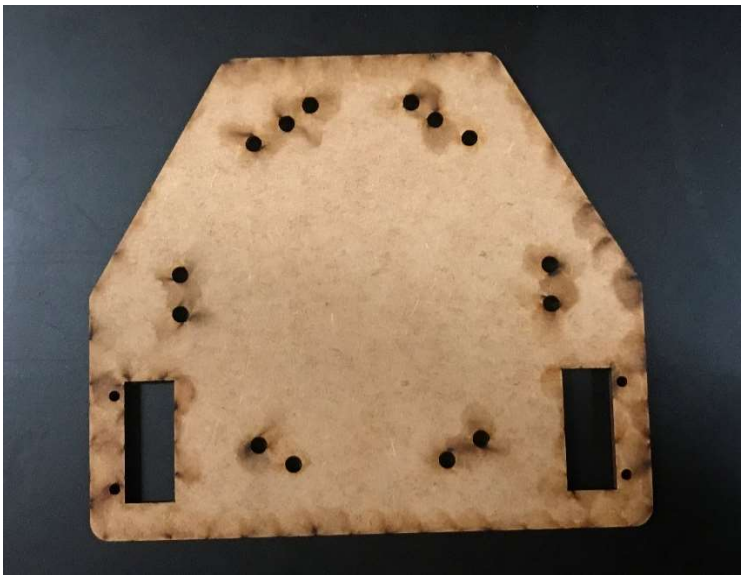
d. Base de robot de MDF y sostenedores para baterías LiPo. Por último, se desarrolló la placa de MDF que soportaría los motores con sus encoders, Ball caster y baterías LiPo. De primero se pensó dónde colocar los dos motores y el Ball caster para tener la mejor estabilidad posible para el robot. Se tomó la decisión de seguir la forma de hexágono para esta estructura así también como colocar los motores en un lado del hexágono y el Ball caster del otro lado para una mayor estabilidad. Para colocar los motores se utilizaron los dos tornillos que tenían estos módulos para agregarlos a la placa base de MDF, se debió cortar unos rectángulos para que estos encajaran adecuadamente. Así también se debió cortar seis agujeros, dos para implementar el Ball caster y otros 4 para los cuatro tornillos infinitos que atravesaran toda la estructura. Para las baterías se encontró que el espacio que existe entre los tornillos sería suficiente para acomodarlas y con dos piezas de MDF se podía las asegurar las baterías a la pieza.

Figura 220. Diseño de placa base para motores con piezas de MDF que sostiene las baterías tipo LiPo (en verde).



Después se fabricó la pieza base de MDF que habíamos discutido anteriormente. Esta solo se debió exportar el diseño 3D a uno 2D de formato DWG que puede entender la cortadora láser. Se corto la pieza y no se tuvo ningún problema debido a que esta tiene una precisión de xxx por lo que todas las piezas encajaron en el primer intento.

Figura 221. Pieza de MDF impresa.



2. Placa de potencia eléctrica. Desde el principio se buscó utilizar módulos que ya tenían circuitos implementados para las funciones que deseábamos en los casos que se podían. La razón de esto fue para disminuir las posibilidades de fallo del robot ya que estos módulos son desarrollados por empresas que tienen mucha más experiencia y conocimiento que un alumno de grado de licenciatura, por ello sabemos que estos módulos tendrán un mejor desempeño que algún circuito que se pudiera implementar, además de que los objetivos de estos módulos no son el desarrollo de circuitos de carga de baterías o de manejar de motores sino la de implementar estas funciones a través de los componentes necesarios.

Cabe aclarar que para el momento en que se empezó a desarrollar este módulo ya se había seleccionado y obtenido el driver de los motores y los motores con su encoders antes que todo lo demás, es por esta razón que ya podemos calcular cual será el consumo total de los motores con todas sus partes incluidas. Para saber los consumos de los motores se probó conectarlos al driver y manejarlos a máxima rapidez de manera que pudimos ver cuanto era la corriente máxima que estos podían necesitar. Además, también se conectaron los encoders y se grabó cuanta corriente estos consumían.

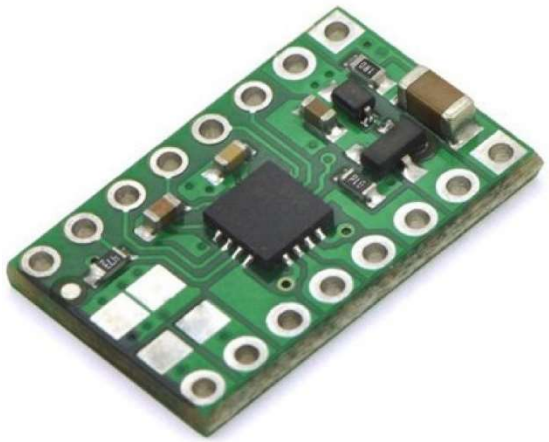
Para el driver de los motores se seleccionó el módulo DRV8833, este cuenta con un circuito integrado que tiene la capacidad de manejar dos motores DC a 1.2 amperios de manera continua y 2 amperios como pico. Ésta fue una muy buena selección ya que sabíamos que nuestros motores consumirían típicamente una máxima carga de 222 mA por motor. Para cada motor se cuenta un puente H doble que nos permite manejarlos en ambas direcciones, los motores pueden ser manejados con una señal PWM de control para

poder controlar la velocidad por medio de un ciclo de trabajo. A demás de todo esto cuenta con protecciones de sobre corriente y sobre temperatura. En el siguiente cuadro podemos ver más características de este módulo. Como podemos ver este módulo tiene parámetro como VM, que es voltaje que se le va a suplir al motor, muy flexibles. Esto nos permitirá tener un diseño más fácil y practico de la PCB.

Tabla L. Parámetros máximos del módulo DRV8833.

		VALUE	UNIT
VM	Power supply voltage range	–0.3 to 11.8	V
	Digital input pin voltage range	–0.5 to 7	V
	xISEN pin voltage	–0.3 to 0.5	V
	Peak motor drive output current	Internally limited	A
T_J	Operating junction temperature range	–40 to 150	°C
T_{stg}	Storage temperature range	–60 to 150	°C

Figura 222. Driver doble puente H DRV8833



El modo de control PWM específicamente en este módulo viene en dos modos: el de rápido decaimiento y el de lento decaimiento. Lo que significan estos dos modos es que tan rápido va a decaer la velocidad respecto a la diminución del ciclo de trabajo. En la Tabla LI podemos ver como se implementan las señales para ambos modos. Los dos modos fueron probados y se decidió utilizar el de lento decaimiento ya que esta respuesta es menos agresiva y nos ayudara más a la hora de implementar control.

Tabla LI Funciones de control PWM para el driver DRV8833.

xIN1	xIN2	FUNCTION
PWM	0	Forward PWM, fast decay
1	PWM	Forward PWM, slow decay
0	PWM	Reverse PWM, fast decay
PWM	1	Reverse PWM, slow decay

Para el desarrollo de una placa que pudiera suplir de energía a todos los componentes internos del robot primero debíamos de saber varias cosas como: cuánta corriente debía entregar, que voltajes debíamos de suplir y que funciones específicas debía llevar a cabo.

Tabla LII. Corrientes requeridas por el robot.

Amperaje por componente				
Componente	Amperaje (mA)	Componentes	Total, alimentación de 5V (mA)	Total, alimentación de 3V (mA)
Motores DC	222	2	444	0
Encoders	15	2	0	30
Dual DC Motor Driver	1.8	1	1.8	0
Módulo de ultrasónicos y control	166	1	100	66
LED 3V	30	1	0	20
LED 5V	30	1	20	0
Amperaje necesario (mA) total			681.8	
En línea de 5V (mA)			565.8	
En línea de 3V (mA)			116	

Funciones que debe de cumplir la placa:

- Alimentar dos motores DC a 5V 444 mA.
- Alimentar placas de ultrasónicos y control con los voltajes y corrientes siguientes: 5V 100 mA y 3V 66 mA por medio de una conexión no permanente.

- Albergar el driver de los motores y proveer sus conexiones necesarias para su funcionamiento correcto.
- Cargar 2 baterías LiPo.
- Circuitos necesarios para convertir energía de estas baterías en los voltajes y corrientes necesarios.
- Capacitores para eliminar ruido agregado por los motores.
- Disipar parte del calor generado por la transformación de energía.
- Proveer una conexión modulable entre las placas de ultrasónicos y control y los encoders de los motores.
- Proveer una conexión con las demás placas que permita un desensamble fácil del robot.
- Protección de corto circuito entre líneas y tierra.

En la placa de potencia se eligieron los componentes necesarios para cumplir con las tareas que requiere este módulo. Componentes como los reguladores, cargadores de baterías y drivers fueron buscados para lograr seleccionar los más óptimos para la tarea necesaria. Se investigo y analizo componente por componente para que todos trabajen en condiciones que no violen sus limitaciones de potencia y si en caso se pudo se sobredimensionan para una mayor seguridad de diseño. Así también se analizó el ruido eléctrico que agregan los dos motores DC y se implementaron en esta placa los capacitores necesarios para filtrar estas perturbaciones eléctricas no deseadas.

a. Selección de componentes. Lo primero que se analizo fue la selección de dos reguladores con las baterías. Sabemos que requerimos de una batería que nos pueda brindar al menos 1 amperio de descarga segura. Cabe aclarar que se sobredimensiono la corriente para tener un margen de seguridad grande y que los componentes trabajen en una región segura. A demás de la seguridad esto nos brindara la posibilidad de que en el futuro esta placa pueda alimentar más componentes de los que se han tomado en cuenta esta vez. Para los reguladores se buscó que la línea de 5 V pudiera otorgar fácilmente 0.681 amperios requeridos y un regulador de 3V que nos diera también fácilmente los 0.117 amperios requeridos. Se seleccionaron los siguientes modelos detallados en el siguiente tabla.

Tabla LIII. Reguladores seleccionados y sus características.

Regulador	Tipo	Voltaje de salida	Voltaje de entrada mínimo (típico)	Voltaje de entrada mínimo	Temperatura máxima	Corriente de salida máxima	Protección
NCP1117 LP	LDO	3 V	4.8 V	18 V	125 °C	1 A	Térmica y de sobre corriente.
L7805	LDO	5V	7 V	24 V	150 °C	1.5 A	Térmica y de sobre corriente.

Ya definido esto podemos empezar a dimensionar la batería, debido a que cada celda de una batería aporta entre 4.2 V (en su carga máxima) y 2.4 V (en su carga mínima) aproximadamente se supo que para alimentar los reguladores se debían conectar dos celdas tipo LiPo en serie para generar el voltaje requerido. A demás de esto el robot requiere al menos 5 horas de autonomía en operación máxima. Para saber la capacidad necesaria de la batería que nos lograra la autonomía necesaria podemos utilizar la siguiente formula. El factor 0.7 es para tomar en cuenta factores externos que pueden disminuir el tiempo de vida de la batería.

$$Tiempo\ de\ vida\ = \frac{Capacidad\ (mAh)}{Corriente\ de\ la\ carga\ (mA)} \times 0.7$$

(digikey, 2017)

Esto para nuestro caso nos da lo siguiente formula despejada y con valores ingresados.

$$\frac{5\ H\ \times 681\ A}{0.7} = 4{,}864\ mAh$$

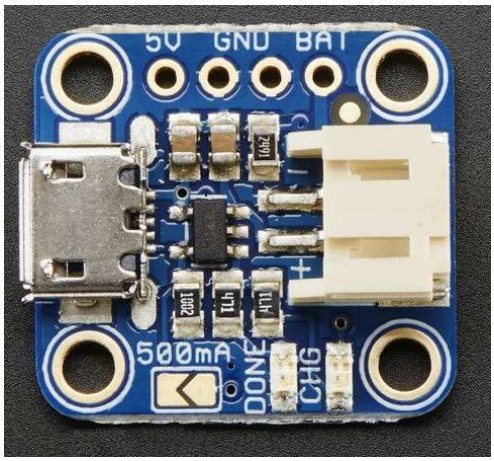
El cálculo anterior nos genera un tamaño muy grande de batería, para lograr esto necesitaríamos 4 celdas. Esto no es posible debido a limitaciones de espacio para las baterías y los costos. Por lo anterior se decidió utilizar un cálculo más conservador donde se colocó la misma fórmula con una hora menos y sin el factor de sobredimensionamiento.

$$\frac{4H\ \times 681\ \mathrm{m}A}{1} = 2{,}724\ mAh$$

Con este cálculo más conservador podemos utilizar dos celdas de 2,500 mAh en serie, que es lo más cercano a la capacidad calculada y que a la vez no resultara muy caro implementar. Se eligieron dos celdas tipo LiPo modelo 785060 de 2500 mAh. Estas celdas nos brindaran un voltaje máximo de 4.2 V y un voltaje mínimo de 2.75 V, una carga menor 1C A (2.5 amperios) y una descarga de 0.5C A ósea de 1.25 A, lo cual es más que suficiente para nuestra aplicación. A demás de todo esto estas traen protección contra corto circuito, sobre carga y sobre descarga. Como hemos discutido anteriormente estas celdas requieren un cargador específico para ellas. Para ello se buscó el cargador adecuado y se seleccionó el cargador LiPo microusb de Adafruit. Este módulo es capaz de cargar una celda tipo LiPo de 3.7V/4.2V con un conector tipo

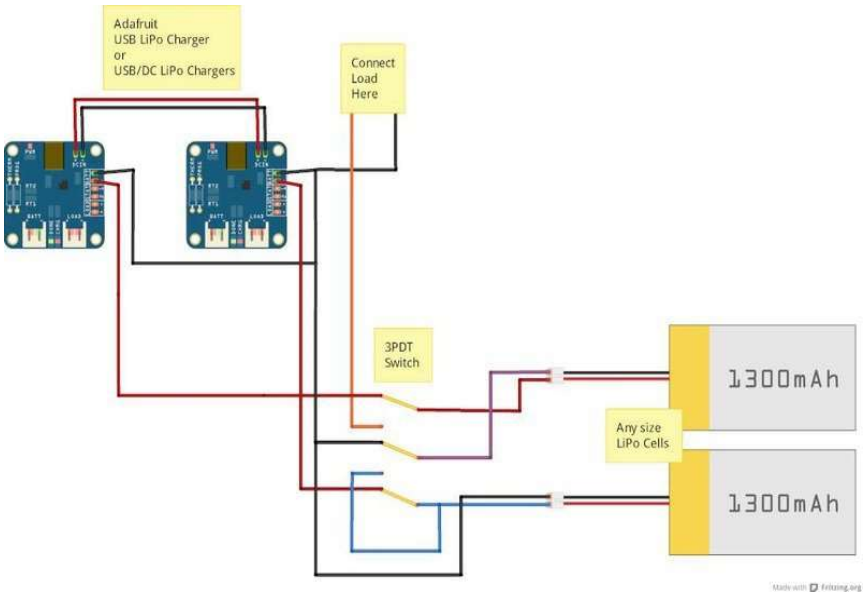
JST a una corriente de 100 mA ajustable a 500mA con un punto de soldadura. Todo esto es alimentado por un puerto microusb. Se realizó el punto de soldadura para lograr cargar más rápido la batería ya que la celda si aguanta esa corriente de carga.

Figura 223. Módulo de carga LiPo.



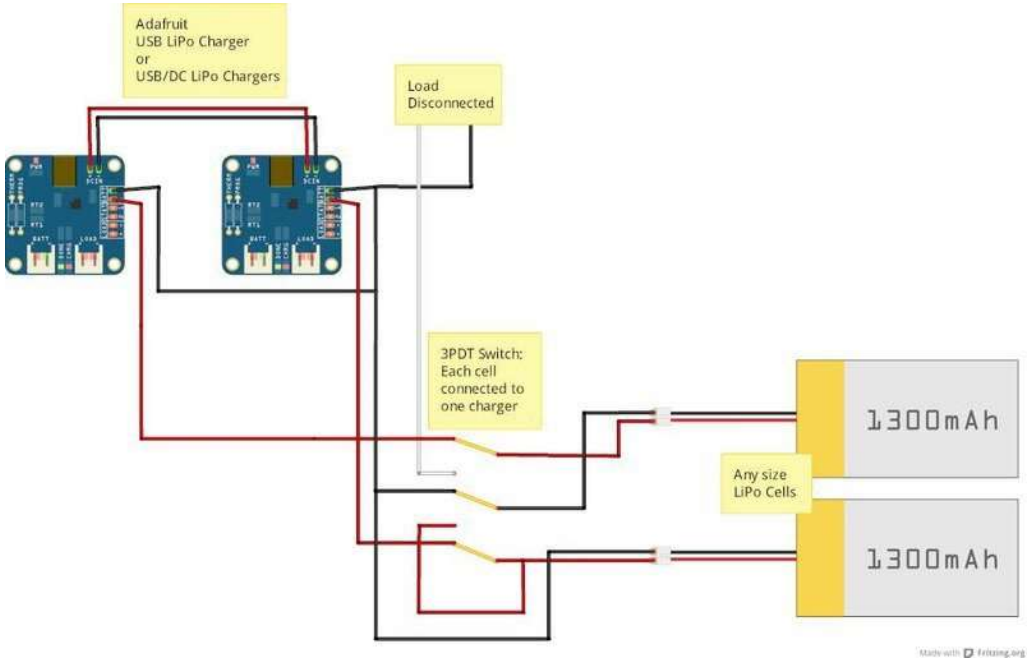
Como se habló anteriormente, para lograr el voltaje y capacidad necesario del sistema se decidió utilizar dos celdas en serie. El problema de esto es que al momento de cargarse cada celda debe de estar con su cargador en paralelo y no conectadas en series entre ellas. Para solucionar esto se encontró un circuito que cambia entre estos dos estados, lo único que requirió este circuito fue un Switch de tres polos dobles. En las siguientes figuras se podrá ver la conexión de las baterías, módulos de carga y carga.

Figura 224. Cableado de baterías y cargadores.



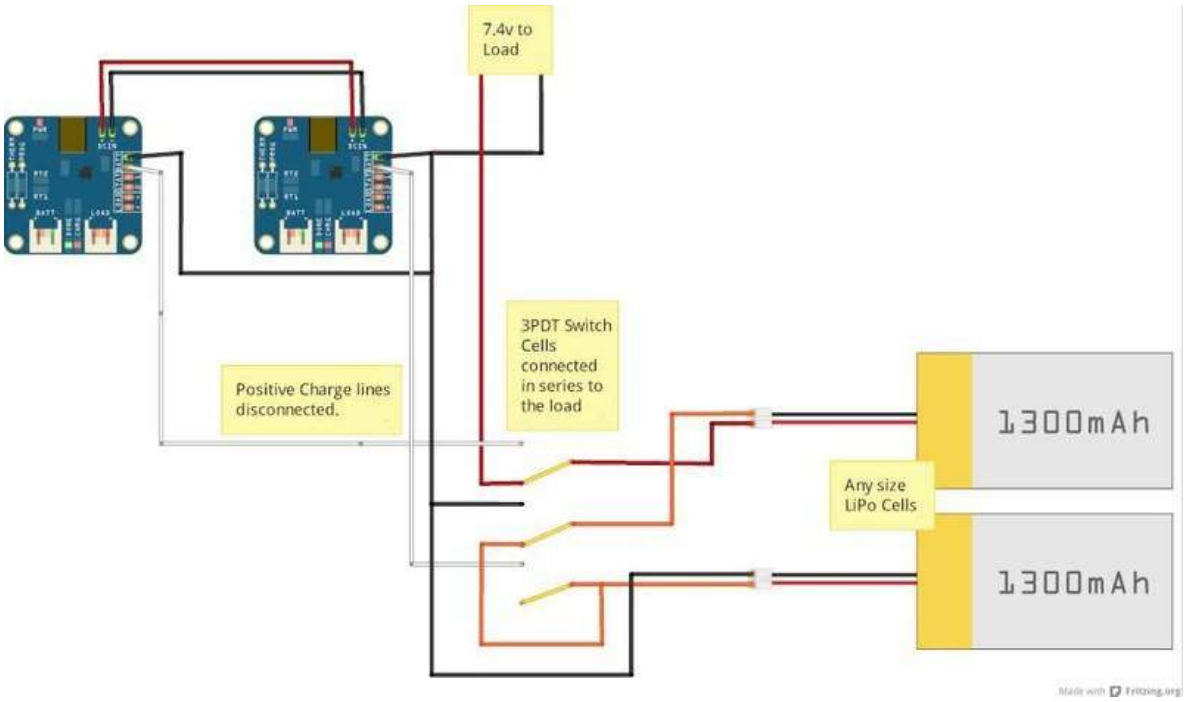
[32]

Figura 225. Cableado de baterías y cargadores en modo carga.



[32]

Figura 226. Cableado de baterías y cargadores en modo descarga.

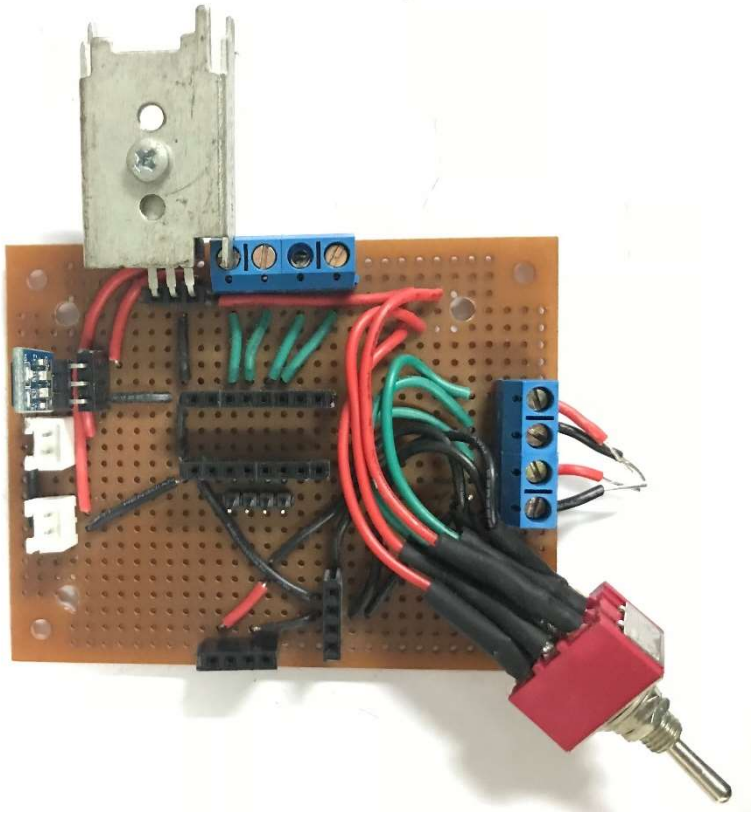


[32]

En las imágenes anteriores se puede ver cómo cada batería está conectada a su cargador en paralelo cuando el circuito está en modo carga y luego cuando se cambia a modo descarga las baterías se desconectan de los cargadores y se interconectan en serie entre ellas, logrando así también conectarse a la carga que se desea alimentar. La ventaja de utilizar esta configuración es que al tener dos cargadores se tendrá una carga más rápida de las celdas, la única desventaja que tiene esto es que las baterías no se balancearan.

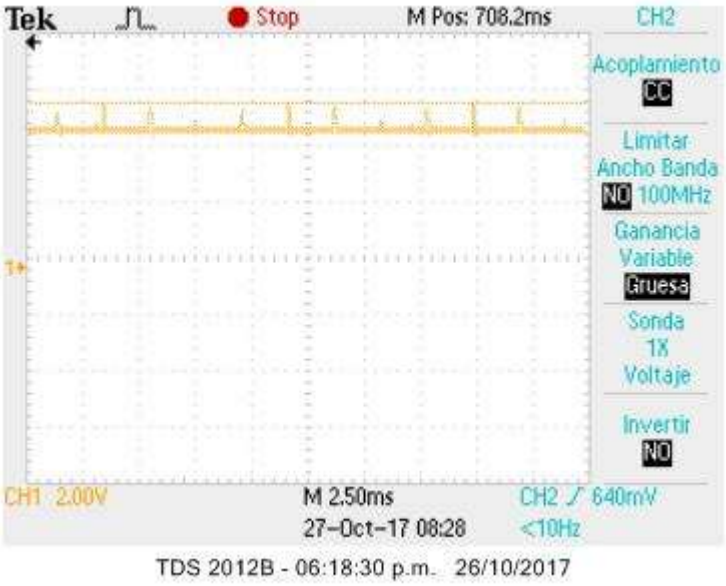
b. Fabricación de circuito de prueba. Debido a que se necesitaba desarrollar un prototipo para seguir realizando pruebas en los demás módulos del robot tipo Swarm se debió fabricar una placa de potencia provisional. Esta se hizo en una placa perforada para poder armarla sin necesidad de tener un diseño a computadora ya terminado. Ya que se hizo en una placa perforada se seleccionaron las versiones *through hole*, estas tenían la ventaja que se podían conseguir en Guatemala y nos iban a permitir probar si la selección de los componentes había sido la adecuada. Una desventaja de este tipo de empaquetado de los componentes es que ocupan más altura y esto hizo que la placa fuera más alta de lo necesario.

Figura 227. Circuito de prueba.



c. Ruido. En pruebas con la placa prototipo se encontró una cantidad de ruido significativo en la línea de 5 V, justamente la línea donde se encuentran los motores con sus drivers, ultrasónicos y microcontrolador. Como podemos ver en la Figura 229 el ruido es significativo, tienen picos máximos de 1.04 V y mínimos de 0.08V además su frecuencia es de 476Hz.

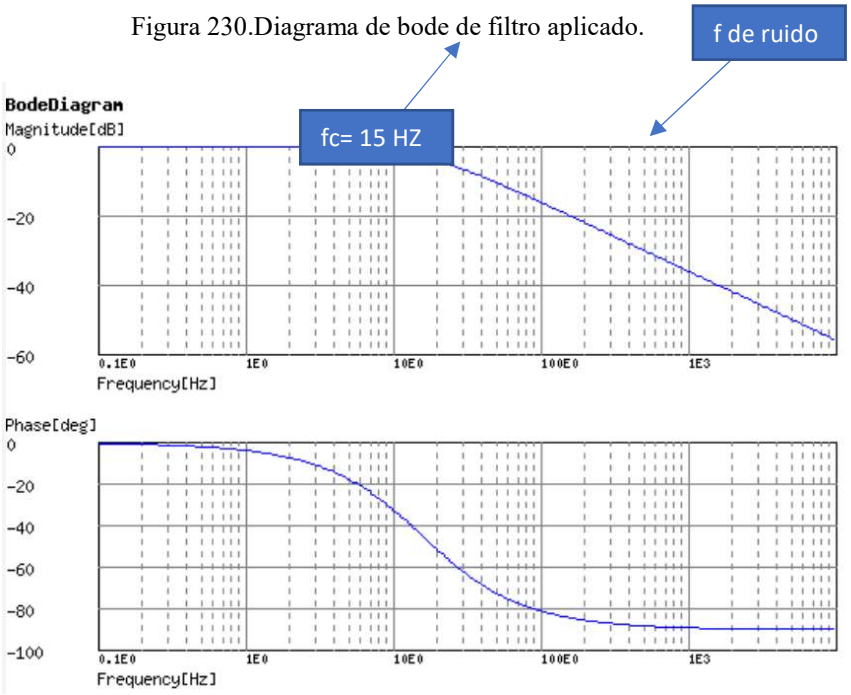
Figura 228. Ruido en línea de 5V.



Este se eliminó agregando un capacitor de bypass (filtro pasa bajas) de 1,000 micro faradios. Ya se tenía capacitor de bypass de 0.1 micro faradios antes en la línea de 5 V requerido por el regulador LDO. Así que al agregar este capacitor se obtuvo una capacitancia de 1,000.1 micro faradios debido a que estos ahora se encuentran en paralelo. El valor de 1,000 micro faradios se probó en el circuito porque es un valor grande, lo que indica una gran reserva de energía para compensar todos los picos de voltaje que se dan. Resulto ser exitosa como podemos ver en la Figura 230. Luego se confirmó que este capacitor era el adecuado utilizando la Ecuación. 8 para calcular el capacitor como un filtro pasa bajar con una carga de 10 ohms, a pesar de que nuestro circuito no es una carga de solo resistencias sino también reactancias e inductancias se decidió tomar 10 ohms como referencia de la carga ya que esto era el resultado de aplicar la ley de ohm a la carga en la línea 5V del circuito. Utilizando la ecuación mencionada anteriormente con los datos de 10 ohms y 1,000 micro faradios se logra generar una frecuencia de corte de 15.9235668790 Hz lo cual esta lo suficiente lejos para eliminar definitivamente una señal de 476 Hz, esto lo podemos ver en el diagrama de la Figura 231.

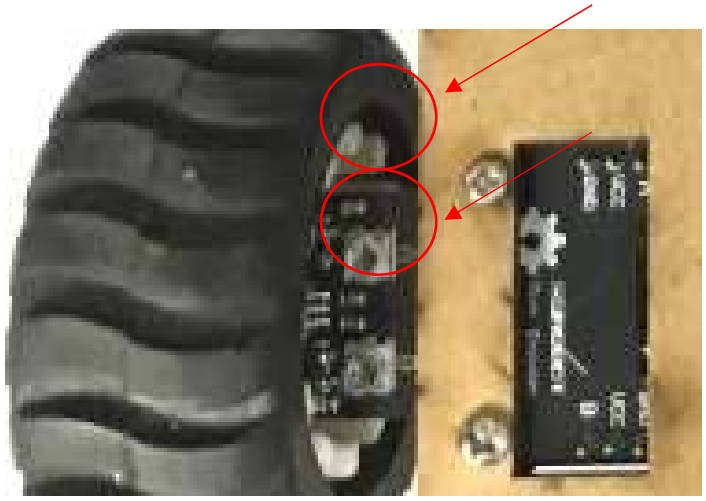
Figura 229. Línea de 5 voltios luego de ser filtrada.

Figura 230.Diagrama de bode de filtro aplicado.



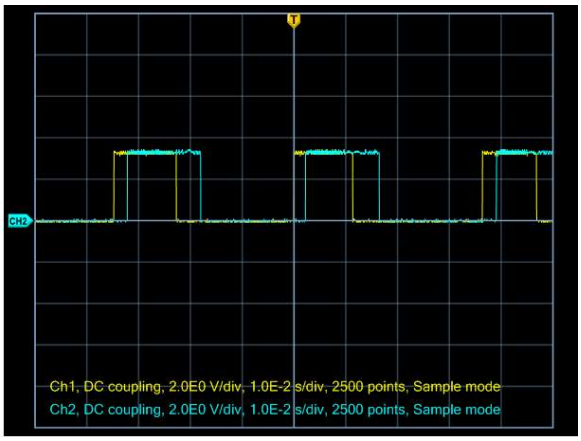
3. Implementación de encoders y de motores. Las señales que nos entregan los encoders utilizados son dos pulsos cuadrados desfasados. Lo primero que se debió hacer fue calibrarlos para el voltaje con el que van a ser utilizados, en este caso fue un voltaje de 3.3 V. Para lograr esto se movieron dos potenciómetros de precisión ubicados en la parte de abajo de la placa de los motores, cada uno de estos corresponde a una señal y tuvo que ser movido hasta lograr ver una señal cuadrada limpia en cada pulso. Se repitió este proceso para las dos ruedas.

Figura 231. Potenciómetros de precisión de los encoders.



Para procesar esta información que se nos fue entregada por los encoders se escribieron unas funciones en Arduino. Antes de escribir el programa se tomó en cuenta que nuestro control necesita solo saber la cantidad de pulsos que llevan los encoders. En la Figura 232 podemos ver como se miran las señales generadas por los encoders al girar la rueda, estas son señales cuadradas con un desfase que nos dirá la dirección en la que está girando el motor.

Figura 232. Señales de encoders en reversa.

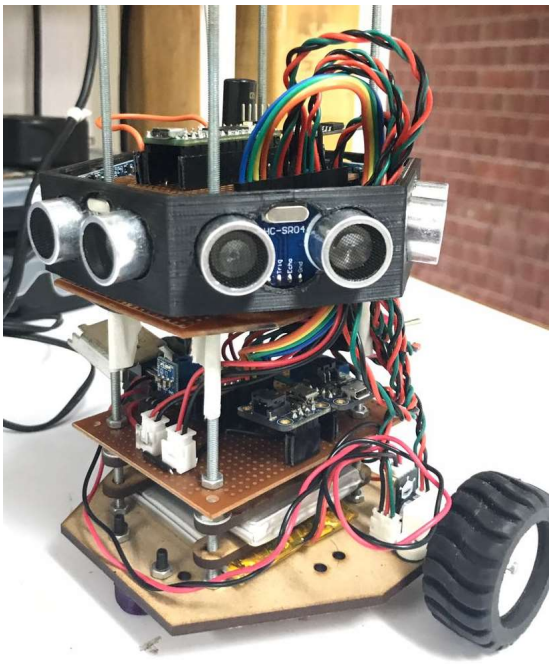


Nuestro código básicamente requiere entregarnos la dirección en la que está girando la rueda y el conteo de cuantos pulsos lleva. Para lograr esto se modificó el código que nos proveía el fabricante para que el programa nos diera la información de los dos encoders. El programa funciona detectando flancos positivos de la señal A de cada encoder, luego mediante lógica calcula cuál de las dos señales entro antes para saber la dirección la que está girando la rueda. Teniendo la dirección el algoritmo decide si debe restar o sumar los pulsos detectado y así poder modificar la velocidad.

Para los motores se requería controlar el dual h bridge driver discutido anteriormente, se programaron funciones para que al ser llamadas estas se encargaran de ingresar las señales necesarias para mover los motores a la velocidad y dirección necesaria. Tales señales fueron las que se vieron en la Figura 233. Se implemento un código que pudiera cambiar entre los dos tipos de modo de manejo de motores (Fast decay y Slow decay) para que el usuario pudiera decidir como quería la respuesta de sus motores.

4. Armado de prototipo. Por motivos de pruebas se tuvo que armar una versión prototipo del robot. Este se armó con las piezas provisionales que teníamos en ese momento. Como se puede ver en la Figura 234 la estructura del robot quedo muy alta debido a que las placas de potencia y de control eran perforadas. Así también hicieron falta piezas como: espaciadores, fusible, pieza que soporta el Switch y el fusible y los headers que transportan las señales de los motores y los encoders hacia la placa de control. Estos detalles hacían que el robot no fuera fácil de armar y desarmar.

Figura 233. Robot prototipo.

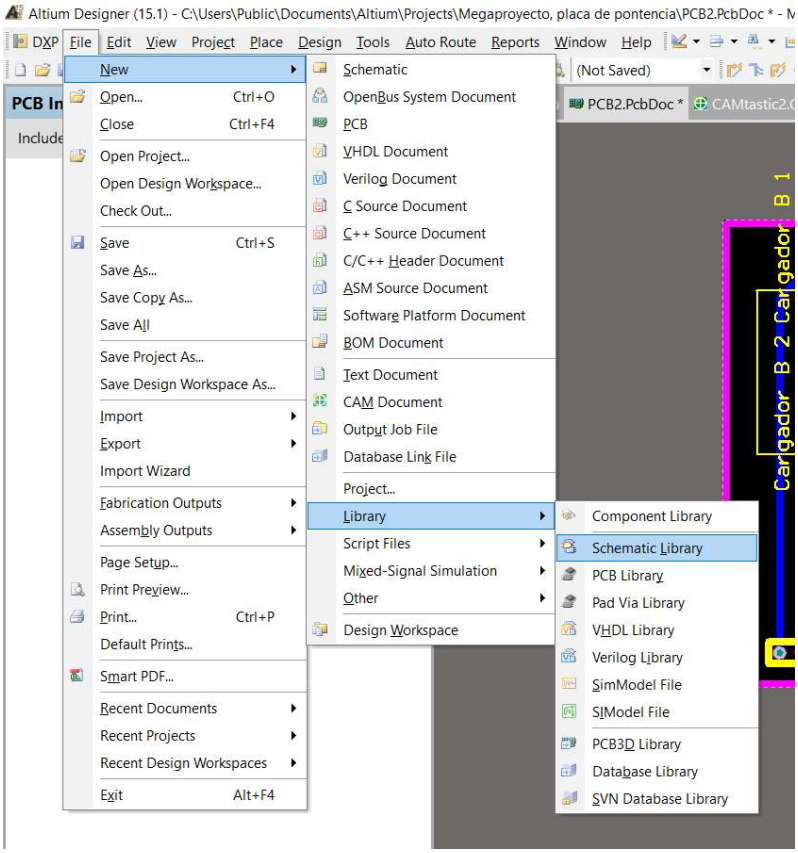


5. Fabricación y diseño de placa de potencia final. Para el diseño de la PCB se decidió utilizar el programa Altium 2015 por la calidad y flexibilidad que este nos brinda para diseñar circuitos. Ya sabiendo que nuestros componentes que habíamos elegido si cumplían con lo que esperábamos ya se pudo empezar a desarrollar el diseño de la placa. Un cambio importante que se hizo al pasar de la placa prototipo a la placa final fue cambiar los componentes que no eran módulos de la placa anterior a componentes SMD (componentes de superficie), para ello se seleccionaron los mismos componentes solo que en sus versiones empaquetados SMD.

El primer paso que se tomo fue crear las librerías necesarias para los componentes que se van a utilizar. En este caso se debió hacer librerías para los dos reguladores, resistencias, capacitores y Leds SMD. Se detallará el proceso que se debe llevar para realizar esto en la siguiente lista.

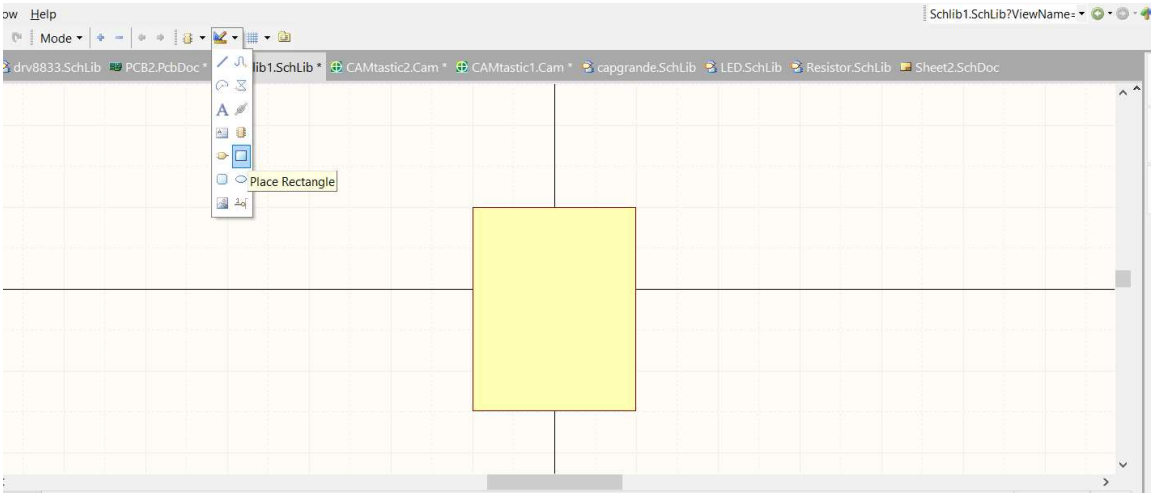
1. Crear librería de esquemático.

Figura 234. Creación de librería de esquemático



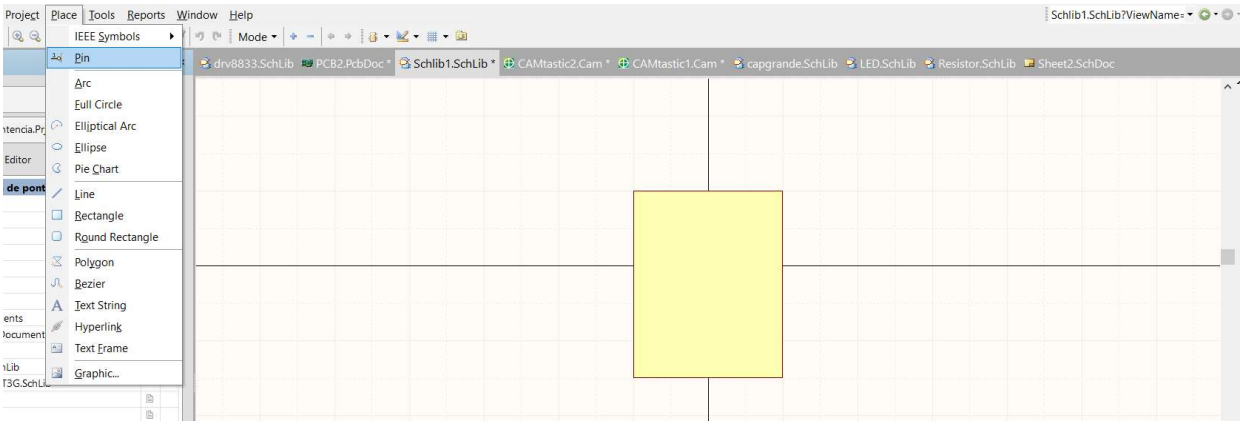
2. Dibujar símbolo de componente.

Figura 235.Dibujando símbolo de componente.



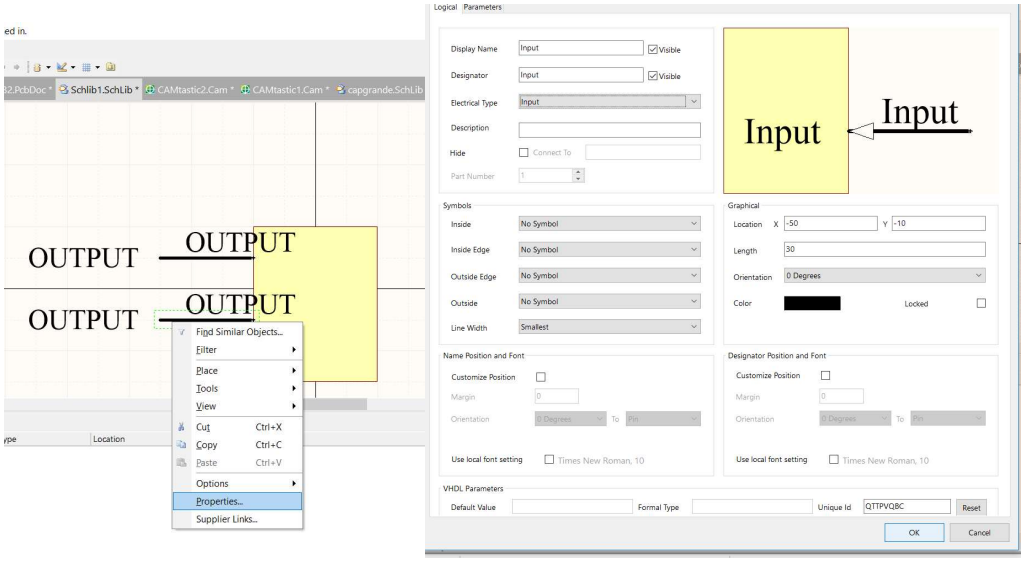
3. Colocar pines de entradas o salidas.

Figura 236.Colocación de pines.



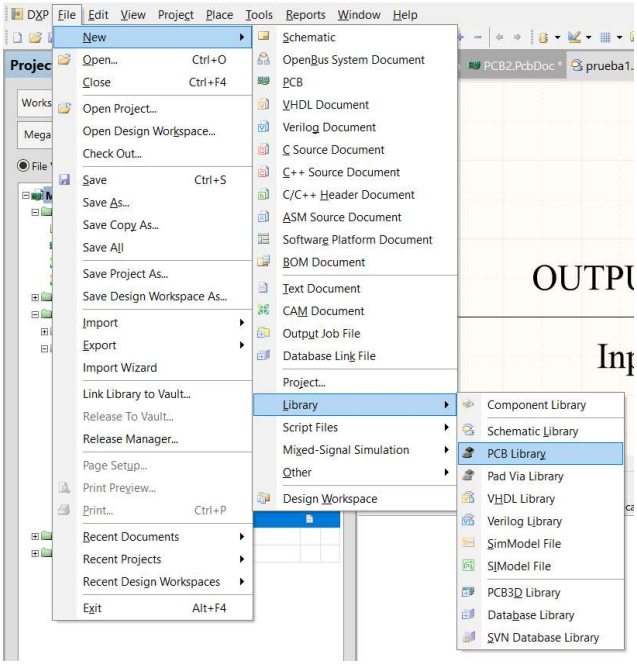
4. Editar pin.

Figura 237.Editando pin.



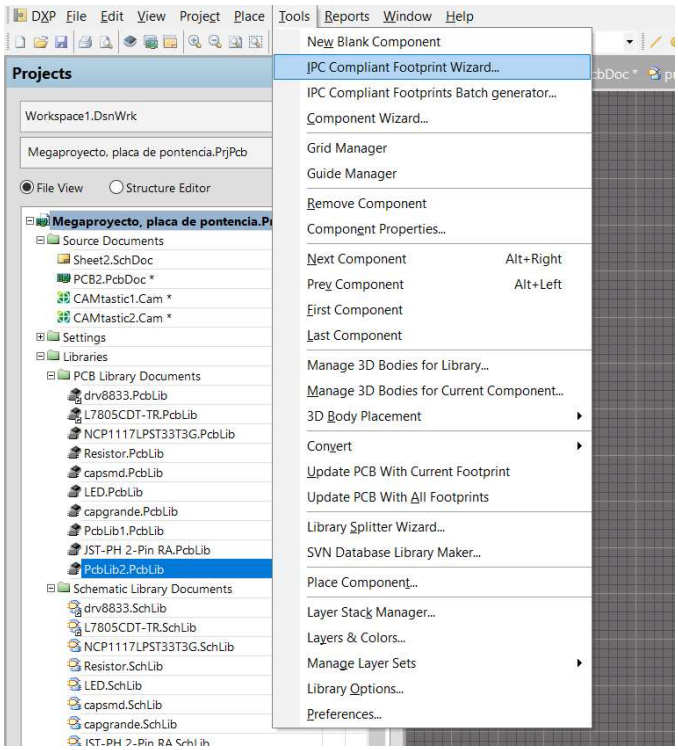
5. Habiendo guardado lo anterior se crea una librería de PCB.

Figura 238. Creación de librería PCB.



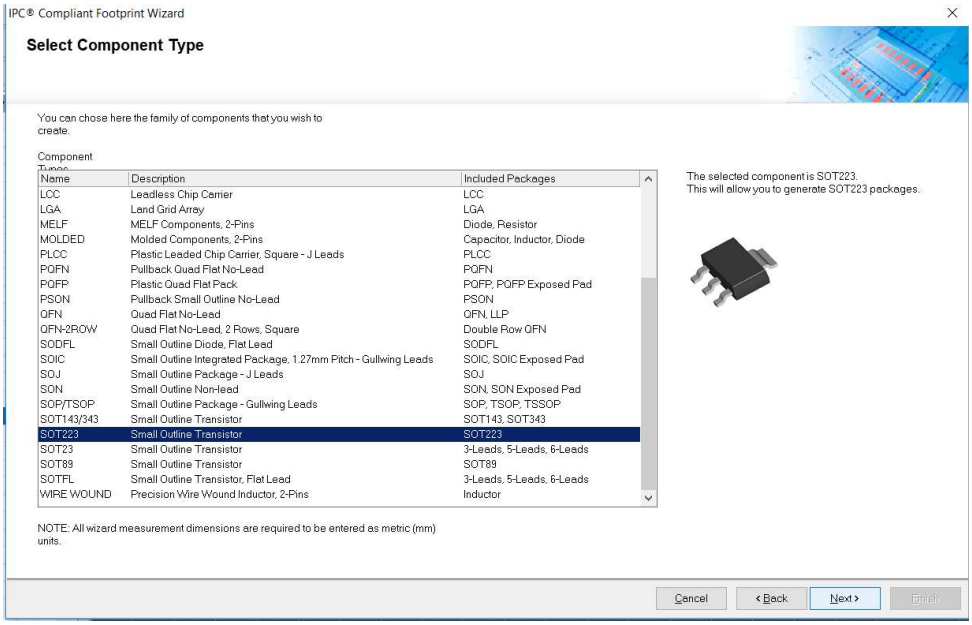
6. Selección IPC Compliant Footprint Wizard.

Figura 239. IPC Compliant Footprint Creator.



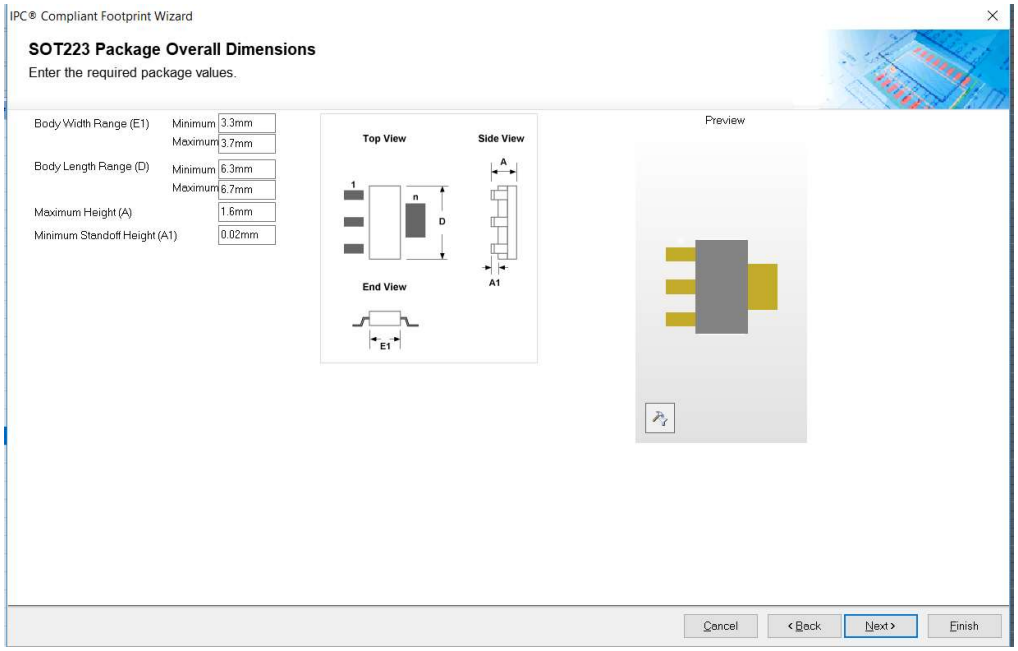
7. Seleccionar empaquetado.

Figura 240.Selección de empaquetado.



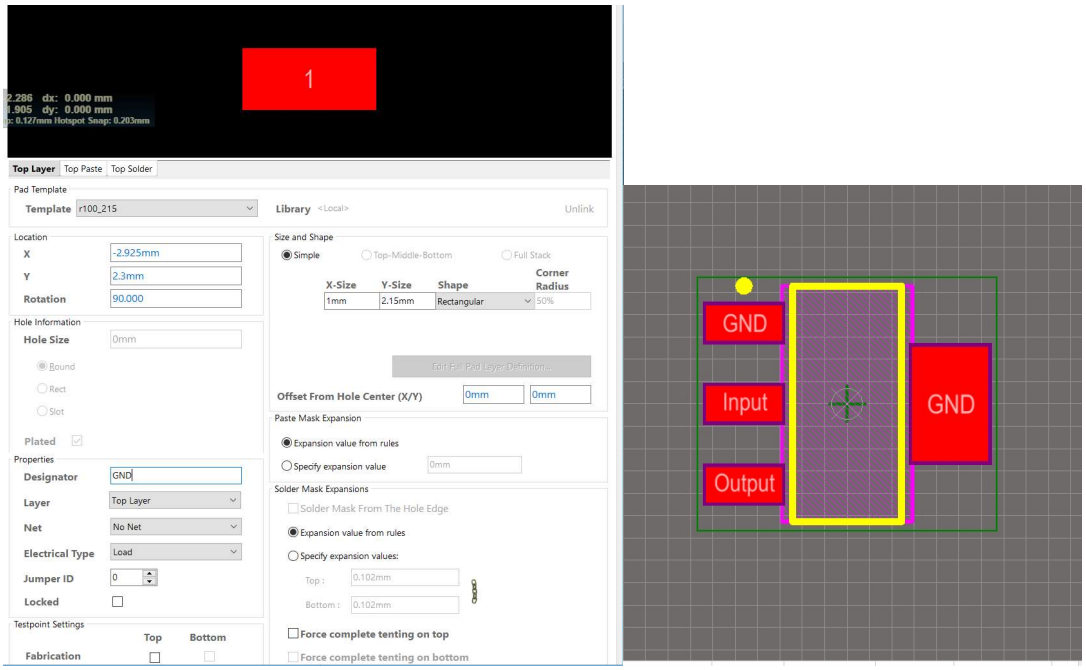
8. Ingresar medidas y apachar luego *finish*.

Figura 241.Ingresar medidas.



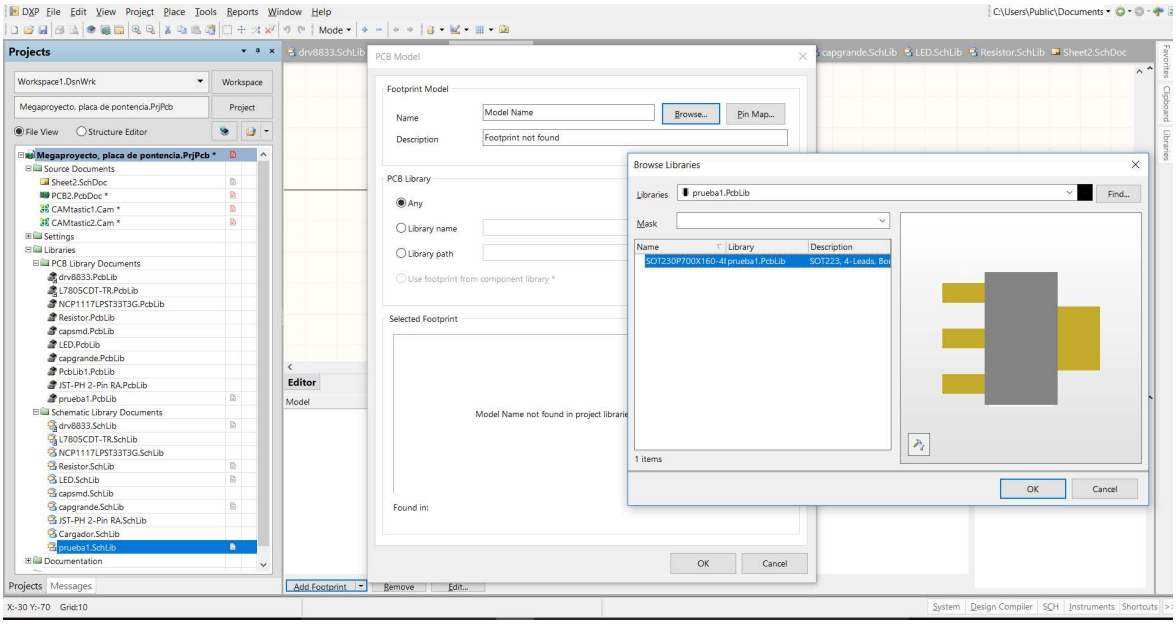
9. Modificar Pads, deben de llevar el mismo designador que llevan los pines en la librería de esquemático creada anteriormente.

Figura 242.Modificación de pad.



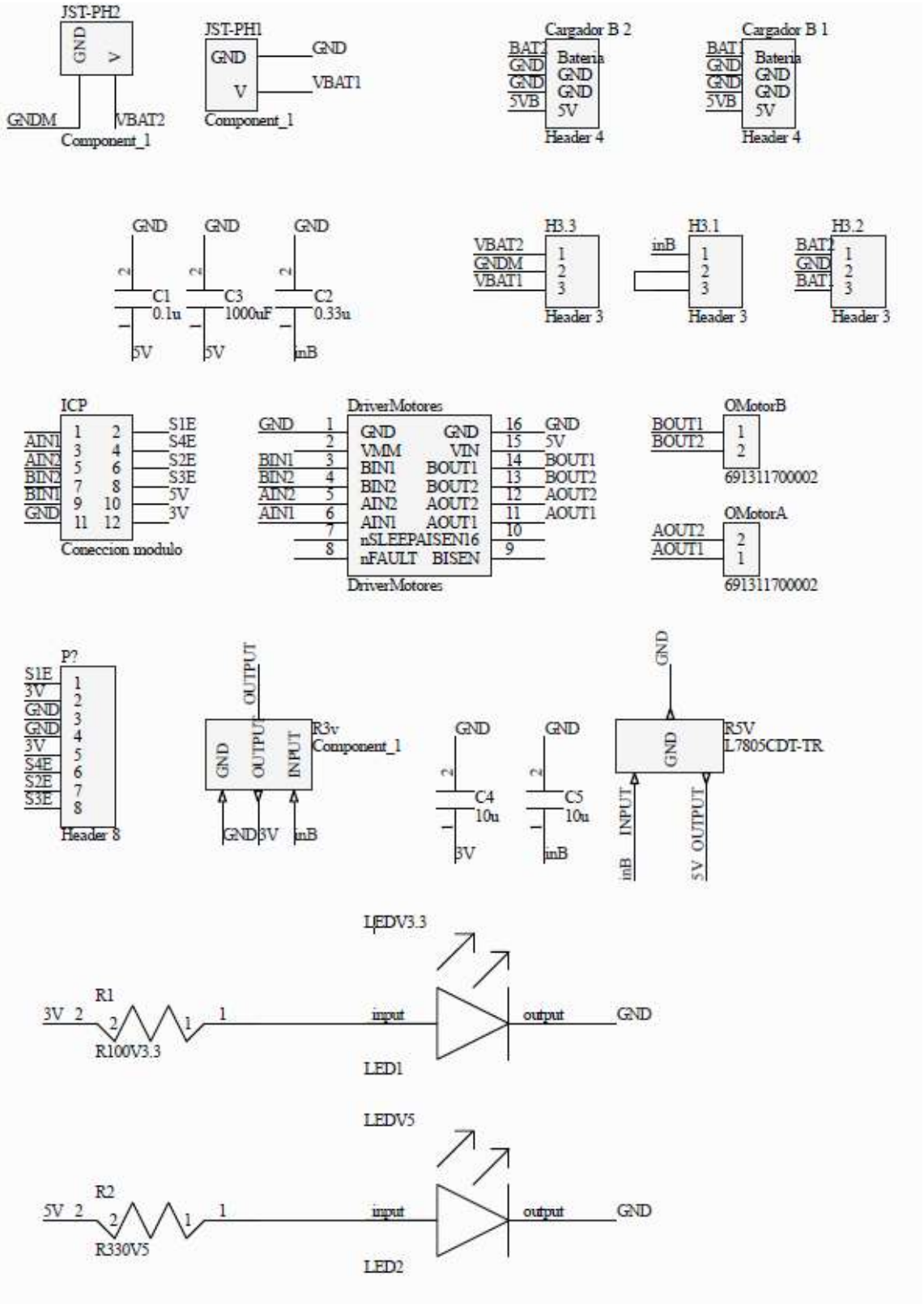
10. Agregar *Footprint* a librería de esquemático.

Figura 243.Agregando footprint al esquemático.



El proceso anterior se repitió para todos los componentes SMD debido a que no se tenía una librería de ellos. Luego de esto ya podíamos desarrollar el circuito, para ello se creó un esquemático nuevo donde se colocaron los símbolos de cada componente y se interconectaron entre ellos de manera que se pudieran formar los circuitos necesarios para el funcionamiento correcto de tales componentes. Como se puede ver en la figura inferior para los módulos de carga y el Switch de nuevo polos basto reutilizar los *headers* que ya contaba Altium en su librería para armar el circuito.

Figura 244. Esquemático de circuito.



Ya teniendo el circuito armado y sabiendo su funcionamiento correcto debido a las pruebas anteriores en la placa prototipo, se pudo proceder a la realización de la PCB. Lo primero que se debió hacer fue calcular el ancho de los *tracks* de la placa para evitar problemas de calentamiento. Como medida de seguridad y para asegurar el correcto funcionamiento de la placa se tomó la corriente máxima que podía pasar por algún nodo de la placa, a esta corriente de 681 mA que es la máxima teórica. Luego se le sobredimensión el 15 % más y se obtuvo una corriente de 783 mA, ya sabiendo la corriente y que la placa de cobre utilizada tendrá 0.5 onzas de cobre por pie cuadrado se pudo ingresar los datos a la herramienta a ANSI PCB TRACE WIDTH CALCULATOR y se tomó el tamaño que generaba el cálculo de dimensión con esta corriente para todos los *tracks* de la placa. En la figura inferior podemos ver que la herramienta nos generó un ancho de track de aproximadamente 35.2 mil para tracks internos y un ancho de 13.53 mil para tracks externos. Como medida de seguridad, y para tener una menor caida de voltaje y una soldada mas facil se tomo como regla de diseño para esta placa los valores generados para los tracks internos y no los generados para los externos. Asi tambien se tomo el clearence requerido señalado en la parte inferior derecha de la Figura 243 de 24 mil.

Figura 245.Herramienta para cálculo de ancho de tracks.

ANSI PCB TRACE WIDTH CALCULATOR

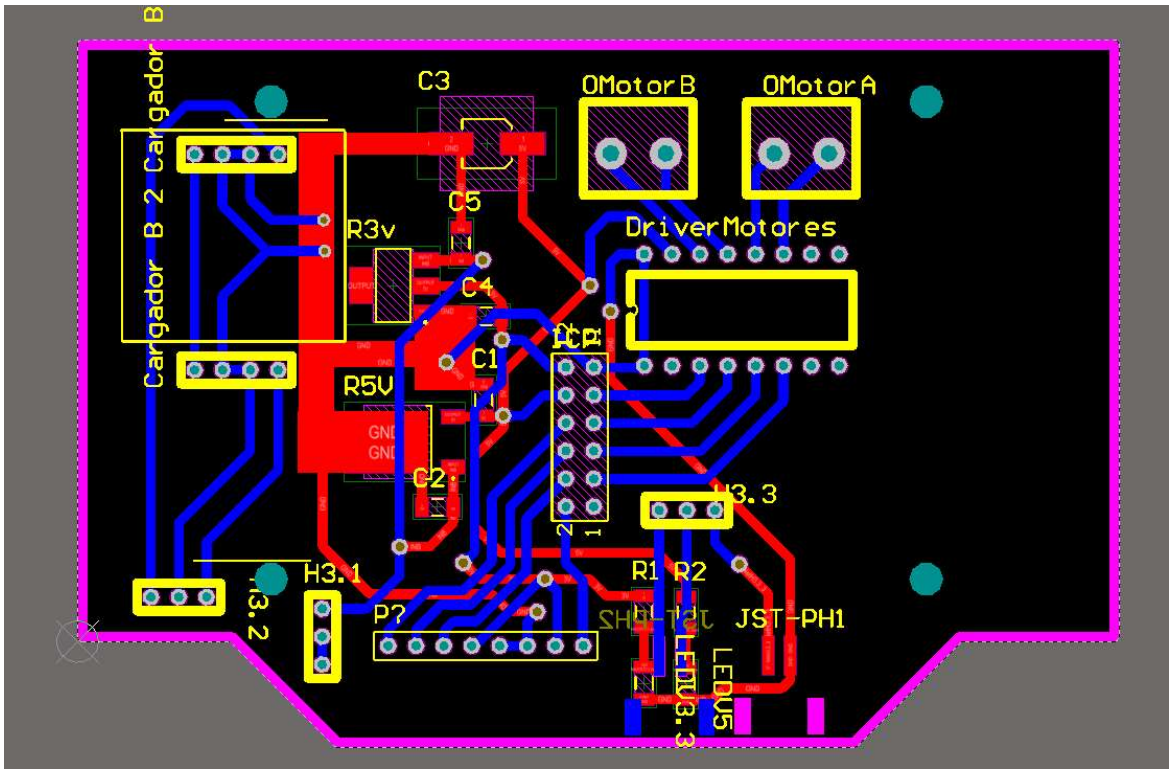
Input Data			Results Data				
				Internal Traces		External Traces	
Field	Value	Units	Trace Data	Value	Units	Value	Units
Current (max. 35A)	783	mA	Required Trace Width	35.2	mil	13.53	mil
Temperature Rise (max. 100°C)	15	°C	Cross-section Area	23.65	mil²	9.09	mil²
Cu thickness	0.5	oz/ft²	Resistance	0.03	Ω Ohms	0.08	Ω Ohms
Ambient Temperature	25	°C	Voltage Drop	0.02	Volts	0.06	Volts
Conductor Length	1	inches	Loss	0.02	Watts	0.05	Watts
Peak Voltage	5	Volts	Required Track Clearance	24	mil		

[114]

Ya con esto se podía establecer las reglas de diseño para poder comenzar con el desarrollo de la PCB en Altium, para esta placa se decidió realizarla de dos lados esto no fue por la limitación de espacio, sino que se necesitaba utilizar componentes de agujero y de superficie. Como se puede ver en la figura inferior se dejó un pad para la tierra de los reguladores, esto nos servirá como disipador de calor y nos ayudará con el ruido. Además, se tuvo que tomar las indicaciones de diseño que nos indicaba el fabricante de los

reguladores en el datasheet. La especificación de diseño nos dice que debemos de colocar en cada regulador dos capacitores de *bypass*. Se colocaron los respectivos capacitores uno cerca de la entrada y otro cerca de la salida como indica el fabricante, estos fueron C1,C2,C4 y C5. C3 es un capacitor utilizado para la mitigación del ruido como se discutió anteriormente.

Figura 246. Diseño de PCB.



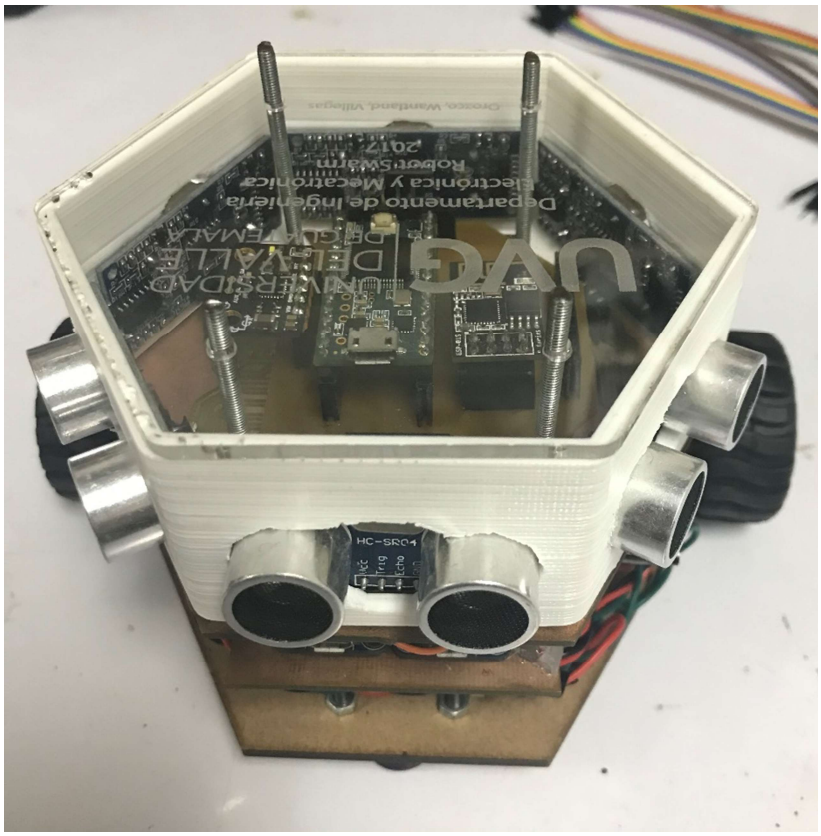
6. Protecciones eléctricas. Como vimos anteriormente la mayoría de componentes electrónicas de la placa tienen protecciones contra sobre corriente. Los reguladores, drivers y baterías todas tienen protecciones dentro de ellas por ello no se decidió implementar algún circuito de protección, lo único que si se decidió implementar fue un fusible de 1 Amperio, el fusible ha sido sobredimensionado para que haya un margen de error a la corriente que hemos calculado y a demás ya que sabemos que el circuito puede proveer sin problema más corriente este módulo podría alimentar otros módulos que se le agreguen en el futuro en caso de ser necesario. La razón de este fusible es que puede existir algún pico de corriente causada por lo motores o algún otro componente y esto puede llegar a dañar los módulos de control del robot.

7. Ensamblaje final. Se tuvieron que hacer unas modificaciones a la estructura para el diseño final, así como implementar las piezas que no se habían puesto en el prototipo. A la base del robot se le agregó un diseño para identificar al robot. La tapa fue implementada y también se le agregó ese diseño.

Figura 247. Tapa del robot.

Figura 248. Base del robot con identificación.

Figura 249. Armado final de robot.



8. Prueba de eficiencia de reguladores y puntos de operación. Para probar la eficiencia de los reguladores se colocaron los reguladores con cargas variables, esto para ver qué tan eficientes eran al convertir voltajes. Se midió cuanto voltaje y corriente entraba al regulador y luego cuanto de estos mismos parámetros salía. De primero se midió la corriente que entraba y salía con varias resistencias de 5 watts. La corriente de entrada se midió con la fuente de voltaje y para la medición de corriente de salida se conectó un multímetro en serie a la resistencia. Luego con estos resultados se calculó la eficiencia para cada voltaje de entrada con la siguiente ecuación. Cabe resaltar que se utilizaron los empaquetados SMD para hacer estas pruebas esto para que nuestros resultados se apeguen lo más posible a la aplicación que se les está dando.

Ecuación. 11. Eficiencia de reguladores.

$$Eficiencia\ \% = \frac{Potencia\ de\ salida}{Potencia\ de\ entrada} \times 100 = \frac{V_{salida} \times I_{salida}}{V_{entrada} \times I_{entrada}}$$

Figura 250. Circuito de prueba para regulador de 5V.

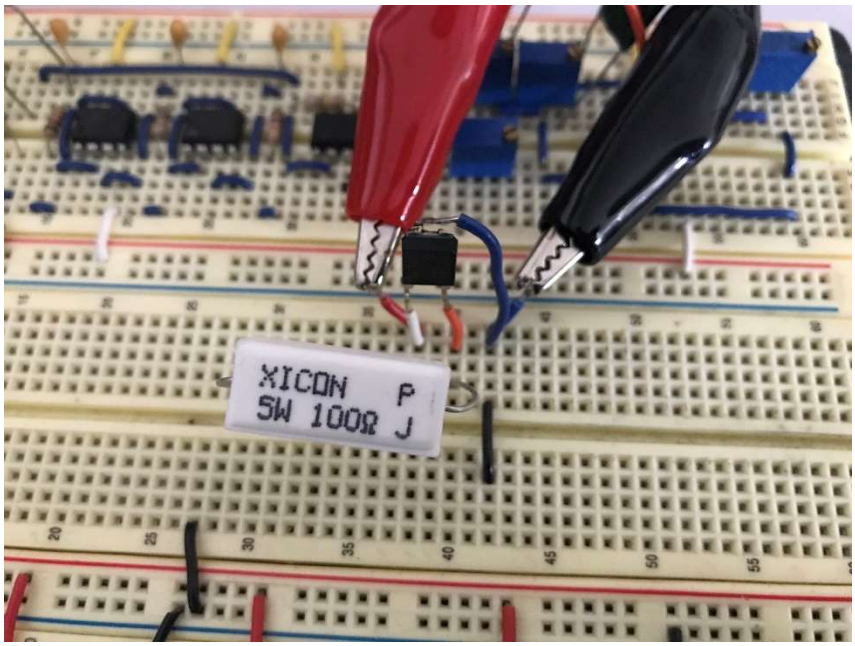
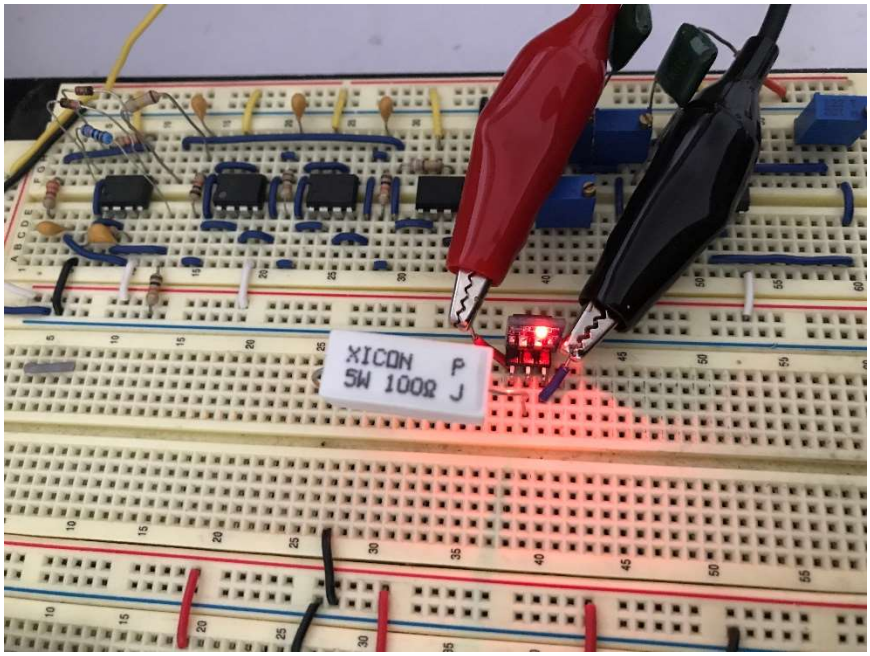


Figura 251. Circuito de prueba para regulador de 3V.



Para sacar los puntos de operación del robot se descargó y cargo varias veces. Para la descarga se activaron todos los módulos en varios intervalos de tiempo. De manera que por momentos se medían todos

los sensores, se enviaba todas estas mediciones mediante el módulo de wifi y activaban los motores a máxima potencia para generar el peor caso. Se trato de recrear el uso que se le dará normalmente a este robot.

9. Pruebas térmicas. Se decidió medir la disipación de potencia en los reguladores. Para ello se utilizó una cámara térmica Flir C2. En la Figura 253 podemos ver como se desarmó el robot y coloco cables jumpers para realizar la conexión entre módulos. De esta manera el robot podía seguir funcionando normalmente y la cámara podía ver lo que estaba pasando en la placa de potencia eléctrica. Las mediciones se tomaron poniendo los motores del robot y demás módulos a su máximo consumo. Se realizaron dos mediciones una sin disipador de calor y otro con disipador de calor.

Figura 252. Robot desarmado para prueba térmica.

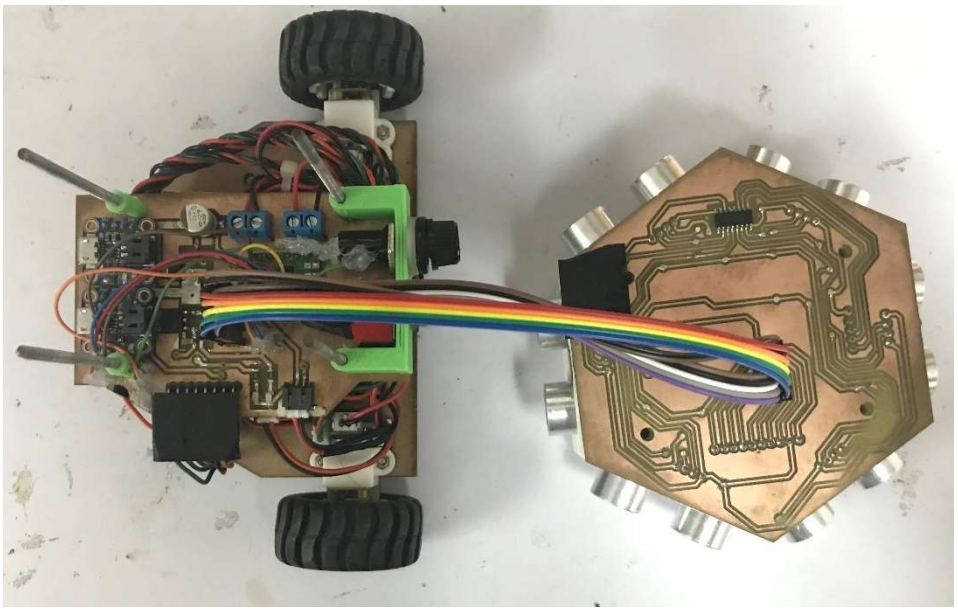
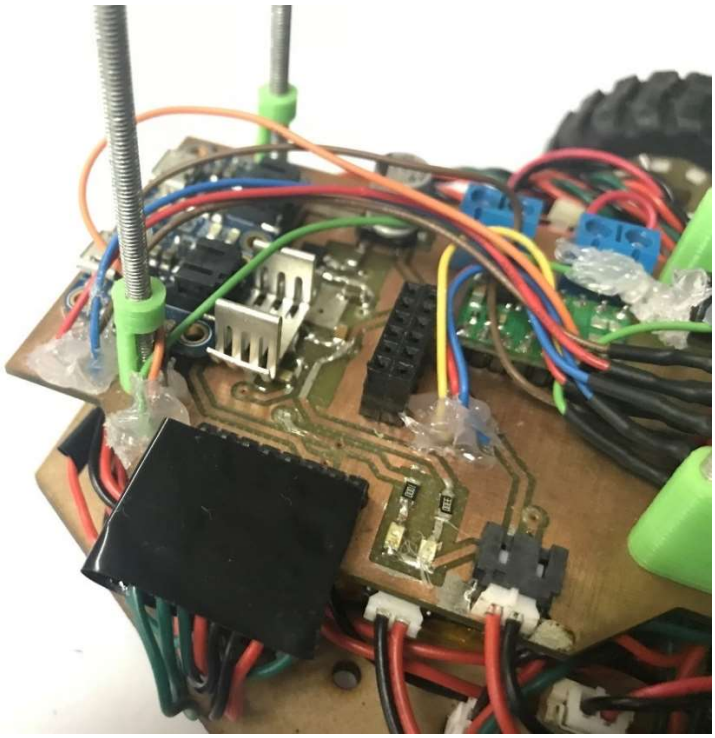
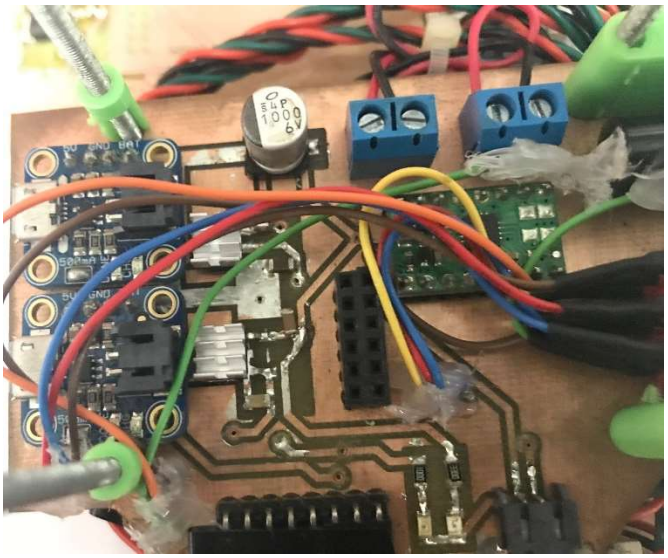


Figura 253. Disipador colocado sobre regulador de 5V.



Luego de ver los resultados térmicos de los experimentos anteriores se llegó a la conclusión de que se requerían disipadores para ambos reguladores tipo LDO, no se podían utilizar dos de los anteriores debido que eran muy grandes para haber dos en el espacio requerido. Para ello se adquirieron disipadores más pequeños (6.35 mm x 6.35mm x 3.18mm), los cuales si cabían dos de ellos como podemos ver en la Figura 254.

Figura 254. Disipadores nuevos colocados.



10. Resultados

a. Eficiencia de los reguladores y puntos de operación.

Tabla LIV. Resultados de mediciones para regulador de 5V.

Resistencia (ohm)	Corriente de entrada (A)	Corriente de salida (A)	Voltaje entrada (V)	Voltaje de salida (V)
5	0.98	0.98	7.4	5.05
100	0.052	0.05	7.4	5.05
10	0.48	0.48	7.4	5.05
5	0.98	0.98	6.45	5.05
100	0.052	0.05	6.45	5.05
10	0.48	0.48	6.45	5.05

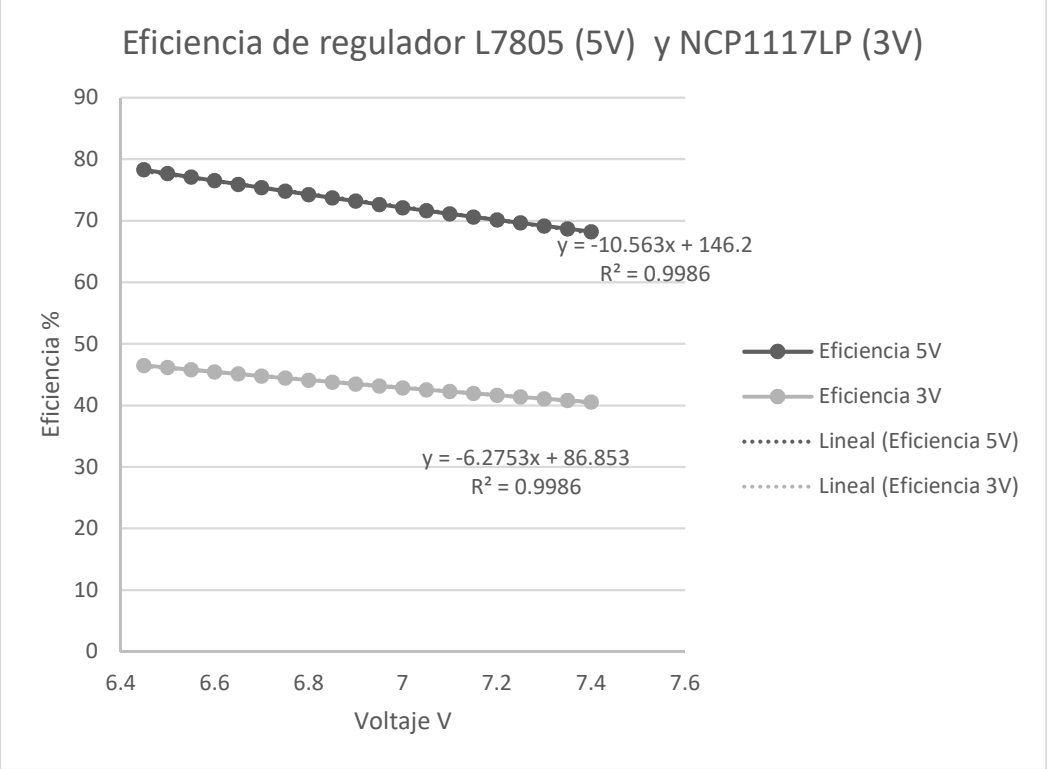
Tabla LV. Resultados de mediciones para regulador de 3V.

Resistencia (ohm)	Corriente de entrada (A)	Corriente de salida (A)	Voltaje entrada (V)	Voltaje de salida (V)
5	0.6	0.6	7.4	3
100	0.03	0.03	7.4	3
10	0.31	0.031	7.4	3
5	0.6	0.6	6.45	3
100	0.03	0.03	6.45	3
10	0.3	0.3	6.45	3

Tabla LVI. Puntos de operación del robot.

Puntos de operación			
	Mínima	Máxima	Típico
Corriente Máxima	0.440 mA	0.600 mA	0.510 mA
Voltaje de entrada reguladores	7.815 V	8.4 V	8.10 V
Autonomía (duración de batería)	4 h	4.5 h	4.7 h
Voltaje de carga de baterías	4.2 V	4.2 V	4.2 V
Tiempo de carga de baterías	4.8 h	5.2 h	5 h
Diferencia de voltaje entre celdas	0.1 V	0.25 V	0.2 V

Figura 255. Eficiencia de regulador de 5V y de 3V.



b. Pruebas térmicas.

Figura 256. Temperatura en operación baja.

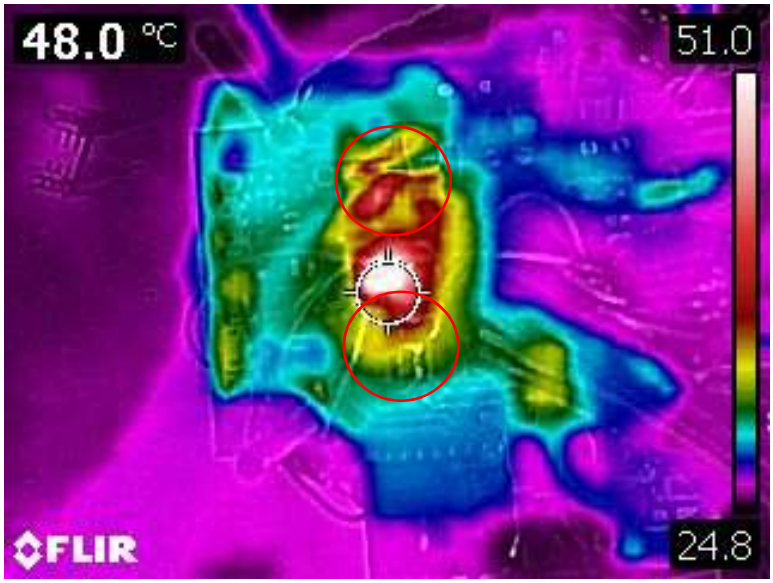


Figura 257. Temperatura en operación alta.

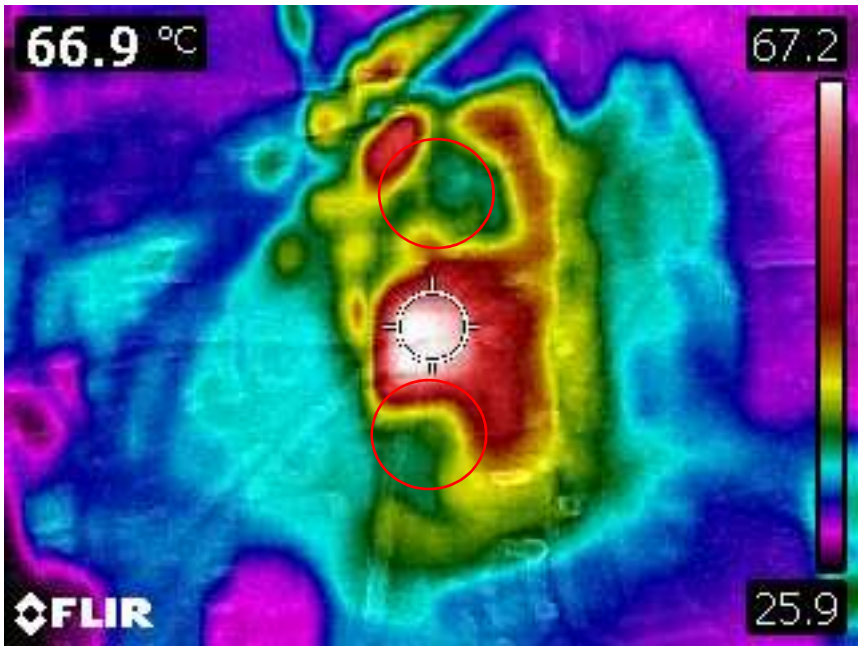


Figura 258. Temperatura con disipador en operación alta.

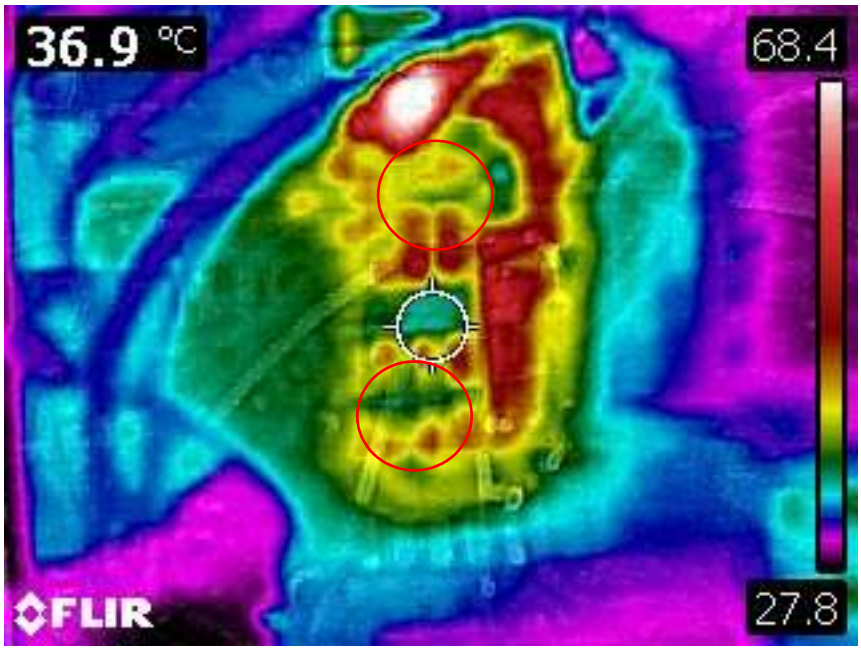


Figura 259. Disipadores nuevos a carga baja.

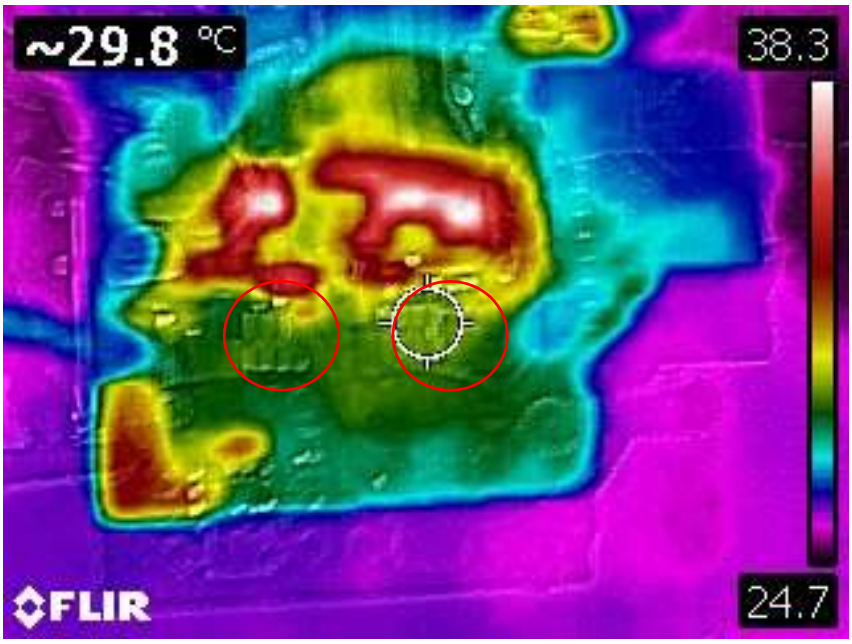


Figura 260.Disipadores nuevos con carga alta.

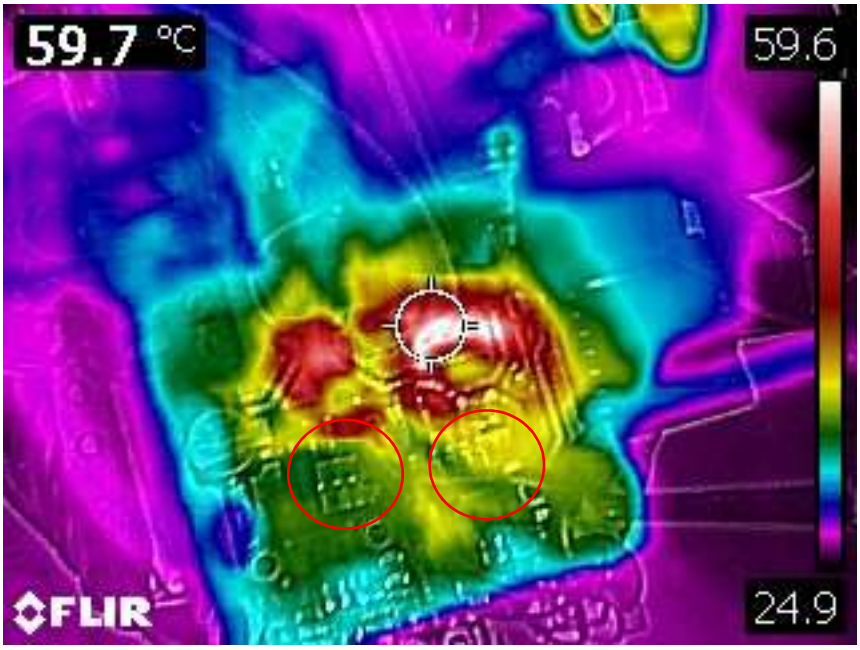
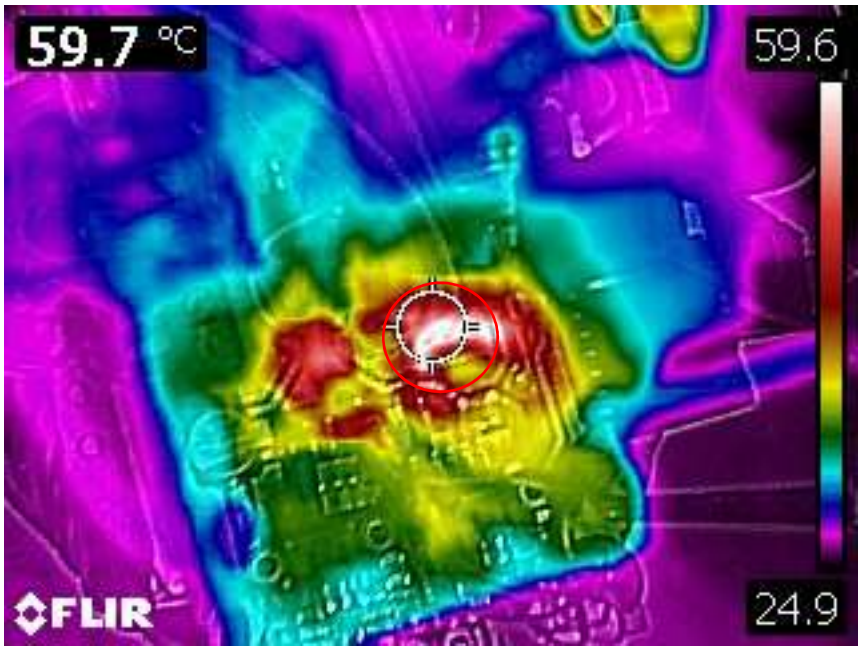
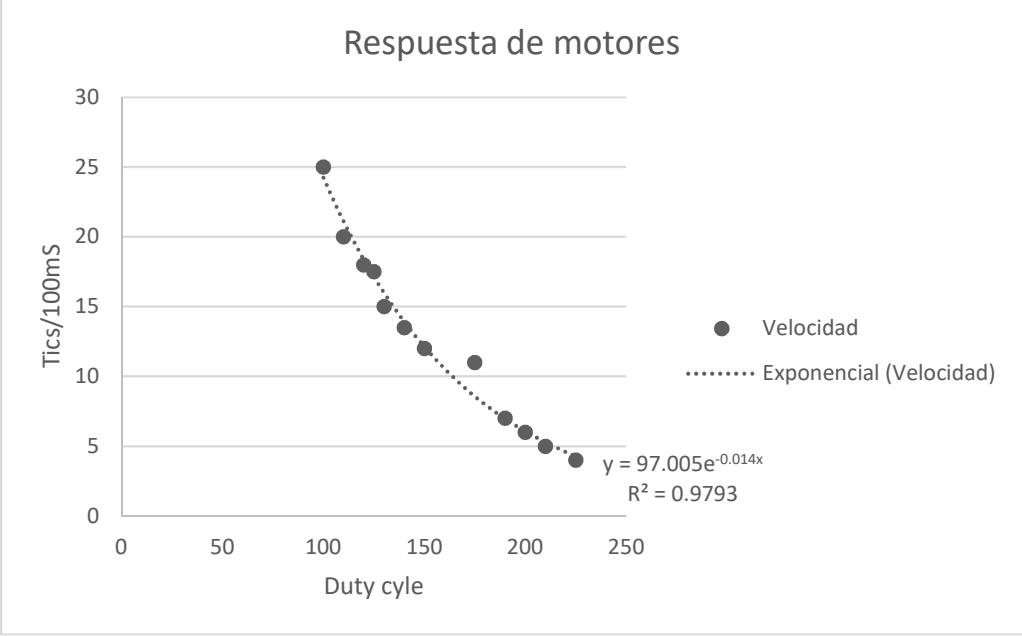


Figura 261.Disipación de calor por medio de la placa.



c. Respuesta de los motores. En la siguiente figura se mostrará la velocidad de los motores en ticks por segundo contra el ciclo de trabajo de la señal PWM de control que ha sido ingresada al driver doble puente H DRV8833 de los motores DC.

Figura 262. Respuesta de motores.



11. Análisis de resultados

a. Eficiencia de reguladores y puntos de operación. La manera en que podemos medir la eficiencia de un sistema es que tanta potencia disipa o pierde al convertir su energía. En la Figura 263 podemos ver la eficiencia de ambos reguladores, se puede ver que la eficiencia no es constante en nuestro sistema debido a que el voltaje que se ingresa a el regulador tampoco lo será. Este voltaje no será constante porque viene de unas baterías colocadas en serie que al descargarse van disminuyendo su voltaje, en este caso un voltaje que va desde 8.40 V hasta 6.45 V. La eficiencia va aumentando mientras el voltaje de entrada va disminuyendo, esto es de esperarse ya que estos reguladores lo que hacen es disipar el voltaje extra que se les ingresa. Esto lo podemos ver claramente en la Ecuación. 3 y la Ecuación. 11. Esta gran pérdida de potencia se debe a la necesaria caída de voltaje que tienen los transistores tipo FET que se utilizan dentro de un regulador tipo LDO. Otro detalle importante que sale a raíz de esto y es evidente en la Figura 263 es que entre más bajo sea el voltaje al que el regulador debe de llegar mayor será la perdida de potencia. Otro aspecto negativo relacionado con la selección de reguladores y su voltaje de caída es que estos requieren al menos 1.45 V arriba del voltaje que van a regular, lo que significa esto es que las baterías no se van a descargar por completo y se desperdiciara tiempo de autonomía. Específicamente este caso las celdas colocadas en serie tienen un límite inferior de voltaje de 5.5 V (2.75 V cada una) y el regulador de 5V deja de funcionar cuando las estas llegan 6.45 V. Esto significa que se está desperdiciando al menos 0.9 V de descarga o aproximadamente el 32.7 % de autonomía. Por lo discutido anteriormente se recomienda no utilizar un regulador de tipo LDO para una aplicación como esta.

Los puntos de operación de este sistema fueron algo muy interesante ya que tales dependieron de muchos aspectos de diseño del robot. Uno de estos datos fue la autonomía del robot, en ella se puede ver como a pesar de tener unos reguladores tan poco eficientes se logró la autonomía calculada teóricamente. Esto sucedió porque a pesar de haber utilizado una batería de menor capacidad a la calculada se encontró una corriente menor a la teórica, esto compenso la capacidad menor a la calculada. Tal corriente típica fue un 20% menor a la que habíamos calculado teóricamente. Esto se pudo haber dado porque las corrientes teóricas de todos los componentes eran en los peores casos, lo que probablemente nosotros no logramos en todos los módulos. Ya con esto se disminuye el consumo y aumenta la autonomía por mucho. La otra razón es que el tiempo de vida que nos dio incluye intervalos de tiempo donde el robot se encontraba en espera y por lo tanto estaba consumiendo menos potencia en esos momentos. Definitivamente este parámetro disminuirá en el futuro por dos razones, la primera es que las baterías sufrirán un desgaste natural y les costara más retener una carga, la segunda es que cuando este robot sea utilizado en una clase varias personas lo utilizaran al día por lo tanto tendrá más intervalos operando a su máximo.

Por último, se puede ver que las baterías no sufren un desbalanceo muy alto. Lo que significa que no es necesario agregar un circuito que las balancee en esta aplicación. Agregarlo significaría disminuir la autonomía del robot porque un balanceador lo que hace es descargar la celda que tiene más voltaje para que ambas terminen con el mismo nivel de voltaje.

b. Pruebas térmicas. Un gran factor que afecta la confiablidad y duración de un circuito implementado en PCB es el calor que este sufre. En la Figura 257, Figura 258, Figura 259, Figura 260, Figura 261 y Figura 262 se puede ver las diferentes situaciones en las que se examinó el calor de la placa. Como era de esperarse las partes más calientes fueron los dos reguladores, como se ha discutido anteriormente estos disipan la gran pérdida de potencia a través de calor. Cabe recalcar que el regulador de 3 V se calentó más que el de 5V en todos los casos estudiados, esto lo podemos respaldar con las curvas de eficiencia de la Figura 263 que nos muestra que la eficiencia del regulador de 3 V es inferior y que por lo tanto debe de disipar más temperatura a través de su empaquetado.

La temperatura máxima en toda la placa que se encontró fue de 68.4 grados Celsius en el área de los reguladores LDOs. A pesar de ser una temperatura alta de operación no existe algún riesgo hacia los componentes, esto debido a que las temperaturas encontradas se encuentran muy por debajo de la temperatura máxima de operación de ellos, para ser especifico entre 75 y 50 grados Celsius menor a la máxima como podemos ver en la Figura 263. Luego el driver de los motores se encontraba a temperatura ambiente, este no sufrió algún calentamiento porque se encentraba trabajando a el 18.5 % de su capacidad nominal de corriente por motor. Otra parte de la PCB que no sufrió algún calentamiento y que vale mencionar fueron los tracks. Esta afirmación nos indica que si se tuvo éxito al sobredimensionar el ancho de ellos y que si se recomienda siempre tomar este paso.

Por último, podemos ver la diferencia de calentamiento en el regulador de 5 V al comparar la Figura 257 y Figura 258. Claramente el colocar un disipador de calor disminuyó la temperatura significativamente. Hubo una diferencia de 30 grados Celsius al colocar el disipador. Esto resultados tienen mucho sentido al ver el área superficial que agrego el disipador al regulador LDO, ya que el al ser un componente SMD este no tenía mucha área en donde dispar toda la potencia perdida. Cabe mencionar que se decidió colocar un disipador en el regulador de 5V ya que este era el más crítico para el funcionamiento del robot, al bajarle la temperatura del mejoramos su actuación y por lo tanto la eficiencia del sistema.

Al tener el análisis anterior se decidió que era mejor que ambos reguladores tuvieran un disipador, ya que anteriormente solo el de 5V tenía a pesar de que el de 3V se calentaba lo mismo o más. En la Figura 259 podemos ver estos nuevos disipadores trabajando en una carga alta. Al ver la temperatura es evidente

que estos disipadores no son tan efectivos como los anteriores (los anteriores bajaban 30 grados Celsius y estos solo 10 grados Celsius), esto se debe a que los nuevos tienen una menor capacidad para disipar calor que el anterior porque tienen una menor área superficial. A pesar de esto son una mejor solución ya que con ellos si caben dos disipadores en vez de solo uno, por ende, podremos proteger dos reguladores y no solo uno. Otra cosa interesante que se observó en estas nuevas pruebas térmicas fue en la Figura 260 como existió una mayor temperatura en la PCB que en los mismos reguladores, específicamente en los planos metálicos cercanos a los reguladores.

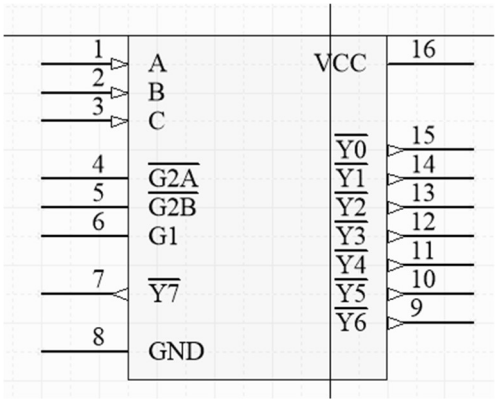
c. Respuesta de los motores. Parte esencial del éxito del sistema de control que se le va a aplicar a esta planta es saber cómo se comportan los actuadores de ella. En este caso se analizó la actuación de los motores, su velocidad en tics por cada 100 mili segundos, respecto a el PWM que se les enviaba. Cabe mencionar que para realizar esta prueba se modificó el código para que contara cuantos pulsos entregaban los encoders por cada 100 mili segundos y que el driver de los motores está siendo controlado en el modo slow decay. Al ver la Figura 263 se puede ver que los motores no tienen un comportamiento lineal ante la variación del ciclo de trabajo de la señal PWM, sino un comportamiento exponencial. Esto es válido decir ya que la regresión exponencial que se le aplico logra explicar el 97 % de las medidas que se tomaron. El comportamiento de estos motores puede tomar una tendencia exponencial más rápida al cambiarse el modo de manejo PWM a fast decay. La selección del modo dependerá de la aplicación que se le quiera dar y que convenga mejor para el control que se utilizara. Al ver los datos se puede notar que en la lectura existen datos que no siguen la tendencia, esto se debe a que las ruedas no son mecanismos perfectos y que la medición no es perfecta en todas las ocasiones.

C. DESARROLLO DEL MÓDULO DE INSTRUMENTACIÓN ELECTRÓNICA

1. Diseño y pruebas del circuito de control de los sensores ultrasónicos. El conjunto de sensores ultrasónicos se utilizará en robots destinados a funcionar en enjambre. Por ello es importante verificar que los sensores no interfieran entre sí, o de lo contrario las mediciones serán inútiles. Se decidió utilizar lógica combinacional externa para utilizar 4 puertos de entrada y salida digitales, en lugar de sacrificar 12 puertos digitales de entrada y salida para usar los sensores directamente con el microcontrolador. Se observó que los sensores ultrasónicos HC-SR04 funcionan con un voltaje de alimentación de 5V. Se realizaron pruebas con los sensores ultrasónicos operando a 3.3V, pero el comportamiento de las mediciones fue errático, parecía aleatorio, midiendo distancias constantes. Por ello se optó por usar 5V como manda la hoja de datos, para estos sensores y su lógica combinacional, para alimentar el microcontrolador, la IMU, y los motores; y 3.3V para el resto de componentes.

En el diagrama de *timing* (Figura 263) se aprecia que el sensor ultrasónico HCSR04 devuelve información de proximidad con un pulso en el pin *Echo* de duración proporcional a la distancia a la que se encuentra el objeto. Esto ocurre cuando el pin de *Trigger* está en un cero lógico. Si el *Trigger* está en un uno lógico, el pin *Echo* no se activa. Aprovechando esta característica del módulo, para garantizar que solo un sensor ultrasónico esté midiendo a la vez, se utiliza un decodificador de 3 a 8 (Figura 264). Se mencionó en el marco teórico que dependiendo de la combinación de bits en A, B, C, una de las salidas $\overline{Yx}$ será diferente al resto, manteniéndose en un nivel lógico bajo, y las demás en alto. Tres pines digitales de salida del microcontrolador se conectan a las entradas A, B, C. Las señales de activación de los pines 4, 5 y 6 se conectan a tierra, tierra y voltaje, respectivamente. Los pines $\overline{Y0}$ a $\overline{Y5}$ se conectan a las señales de *Trigger* de cada módulo.

Figura 263. Decodificador de 3 a 8.



Las seis señales de Echo de los sensores ultrasónicos se simplifican con ayuda de lógica combinacional, para usar una sola señal que se conecta a un pin de entrada digital del microcontrolador. Realizando la conexión de la Figura 264 se garantiza que el pin de Echo de cada ultrasónico solo va a proporcionar información cuando el *Trigger* del ultrasónico esté en cero lógico. Todos los demás cambios de Echo cuando *Trigger* esté en uno lógico no tendrán efecto en la salida. Esta señal va de 0 a 5V. El Teensy

3.2 es capaz de tolerar estos voltajes, pero el Teensy LC no. Por eso se convierten los voltajes de la señal de Echo a los valores 0 y 3.3V, implementando dos compuertas NOT en cascada con transitores BJT NPN. De esta forma se protege al Teensy LC de daños ocasionados por voltajes superiores a los que es capaz de tolerar. Las señales que terminan en asterisco (*) representan señales complementadas (lógica invertida).

Figura 264. Lógica combinacional para simplificar los pines de Echo de los sensores ultrasónicos

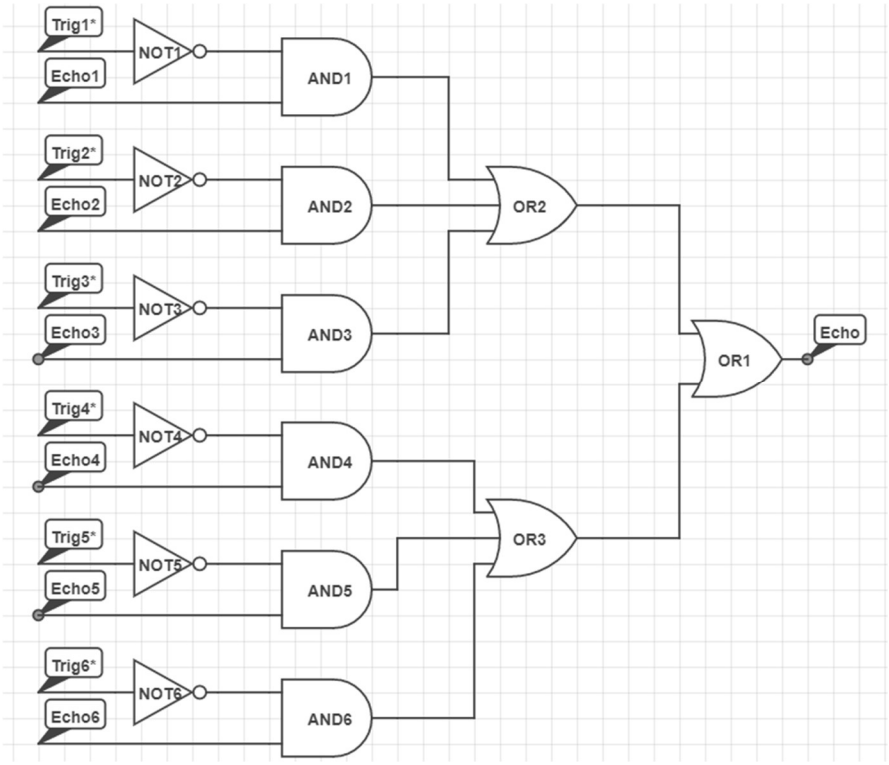


Figura 265. Circuito para probar el funcionamiento de la falda de sensores ultrasónicos

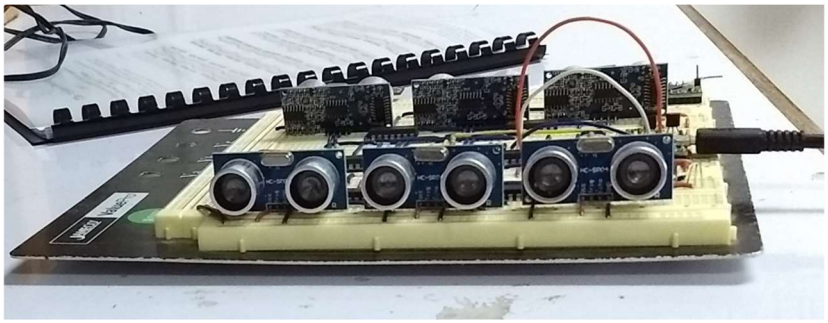


Figura 266. Placas para el prototipo de la falda de sensores.

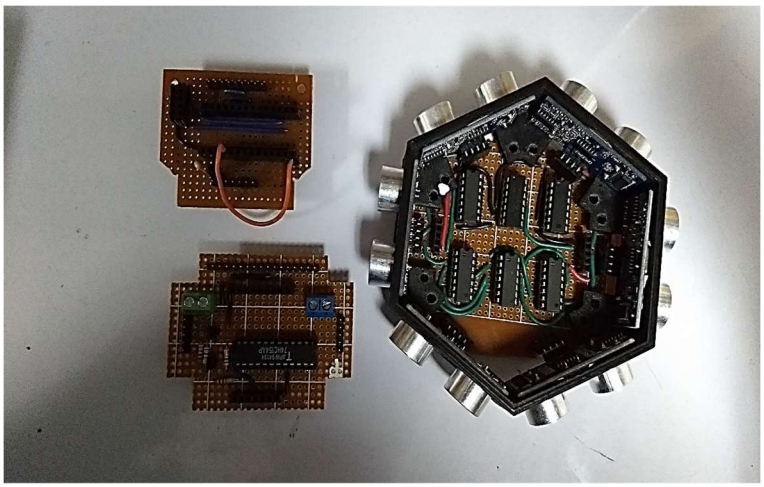


Figura 267. Prototipo de la falda de sensores.

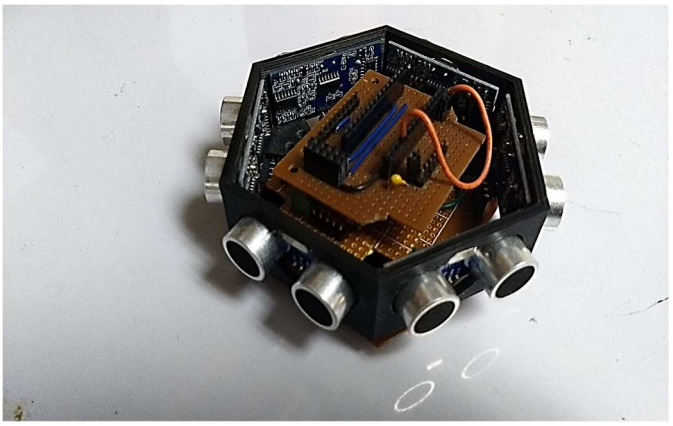
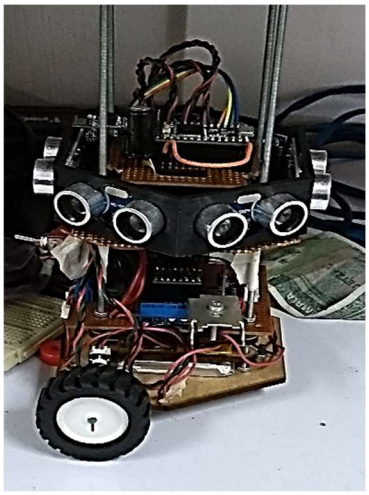


Figura 268. Prototipo ensamblado.



2. Diseño y fabricación de PCB del circuito de control de sensores. Se utilizó el software Altium Designer 14 para diseñar y fabricar el PCB de sensores ultrasónicos y el PCB de módulos. Es importante conocer los accesos directos con el teclado y el mouse de Altium, para facilitar la elaboración del esquemático y el ruteo del PCB. El *grid* puede tener diferente resolución y dos unidades de medida, mil y mm.

a. Creación de componentes. Se seleccionan los componentes en diversos proveedores como tiendas locales, adafruit.com y mouser.com, y luego se crean los esquemáticos y *footprints* para cada uno. Los componentes que se compraron en mouser se muestran en el Anexo h de Swarm Robotics. Primero se crea una librería de esquemáticos de componentes, y en ella se añaden los pines que indica la hoja de datos de cada componente. Con las herramientas de dibujo se crea la forma que aparecerá en el esquemático. Para circuitos integrados, módulos y *headers* se utilizó un rectángulo, para capacitores y resistencias se usaron símbolos conocidos.

Figura 269. Ejemplo de la librería esquemáticos de componentes, para un circuito integrado.

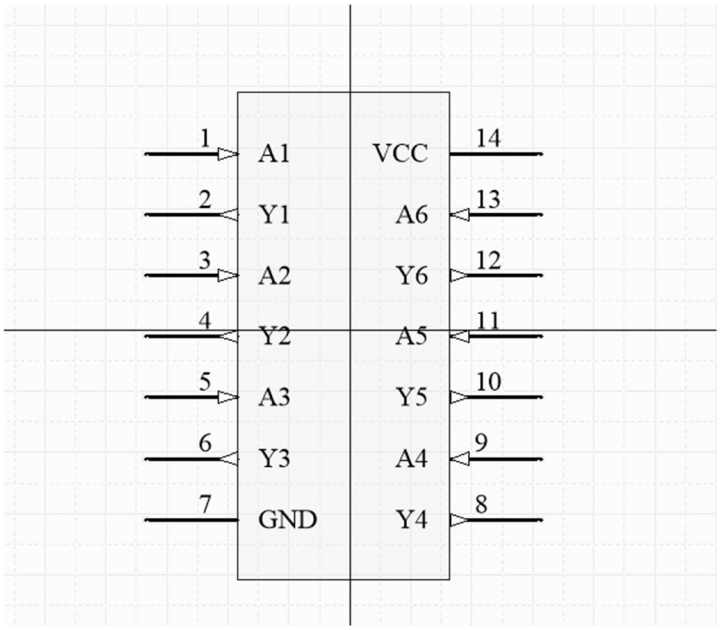


Figura 270. Ejemplo de la librería esquemáticos de componentes, para un transistor npn.

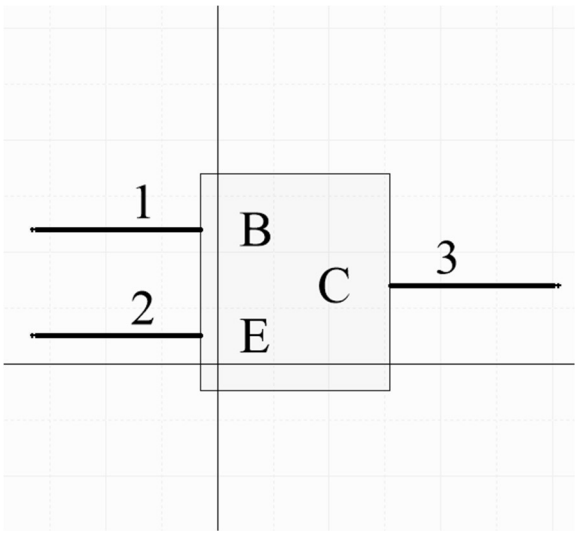


Figura 271. Ejemplo de la librería esquemáticos de componentes, para un capacitor.

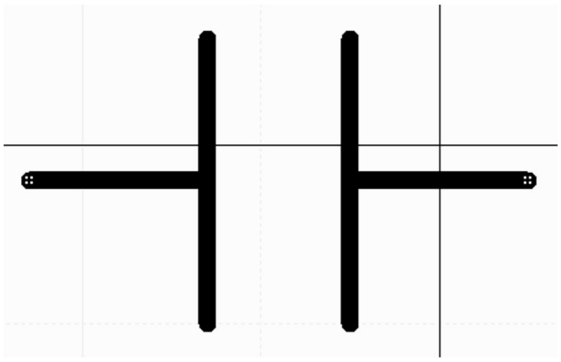
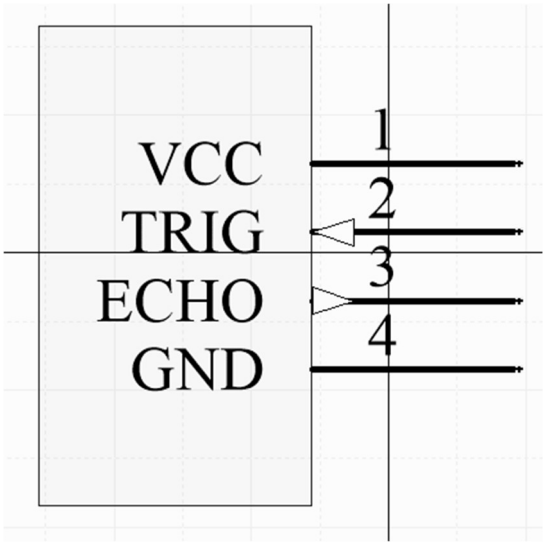


Figura 272. Esquemático del sensor ultrasónico HCSR04.



Después de la elaboración de la librería de esquemáticos, se crean las librerías de *footprints*, para luego añadirlas al esquemático correspondiente en la librería anterior. En cada *footprint* se añade un objeto 3D como archivo step embebido, que se puede obtener en fuentes como grabcad.com, 3dcontentcentral.com, o formas básicas que se pueden realizar en Altium. De esta manera se logra una apreciación tridimensional del componente y su *footprint*, como se vería en realidad. Se debe tener cuidado de que el designador de cada pin tenga el número correspondiente en el número de pin del esquemático, ya que Altium mapea los pines del *footprint* a los pines del esquemático, que tengan el mismo número.

Altium Designer posee un asistente para elaboración de footprints estándar, que incluye empaquetados SMD, DIP, BGA, entre otros. Es útil porque permite introducir dimensiones y tolerancias, información que se encuentra en las hojas de datos de los componentes.

Figura 273. Ejemplo de la librería de *footprints*, para un circuito integrado.

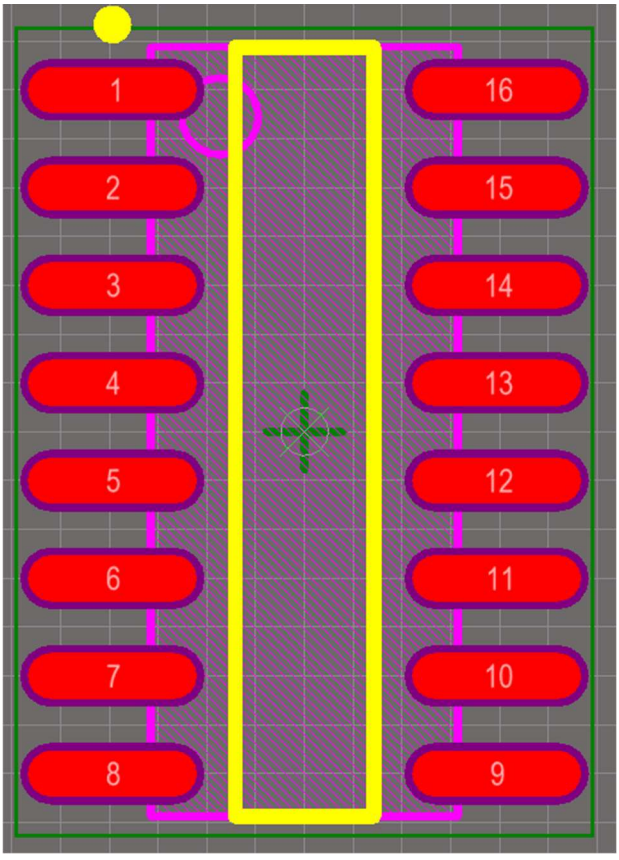


Figura 274. Ejemplo de la vista 3D del componente en la librería de *footprints*.

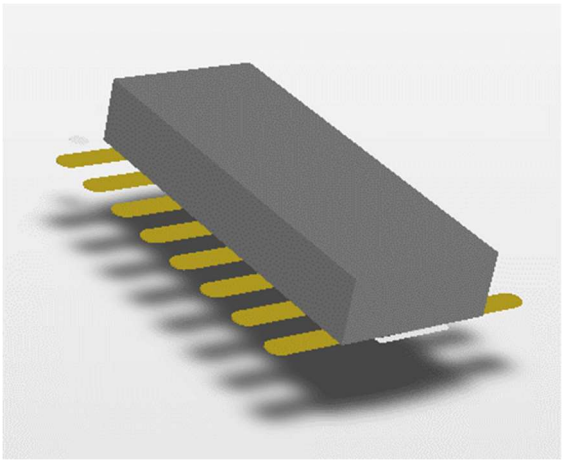
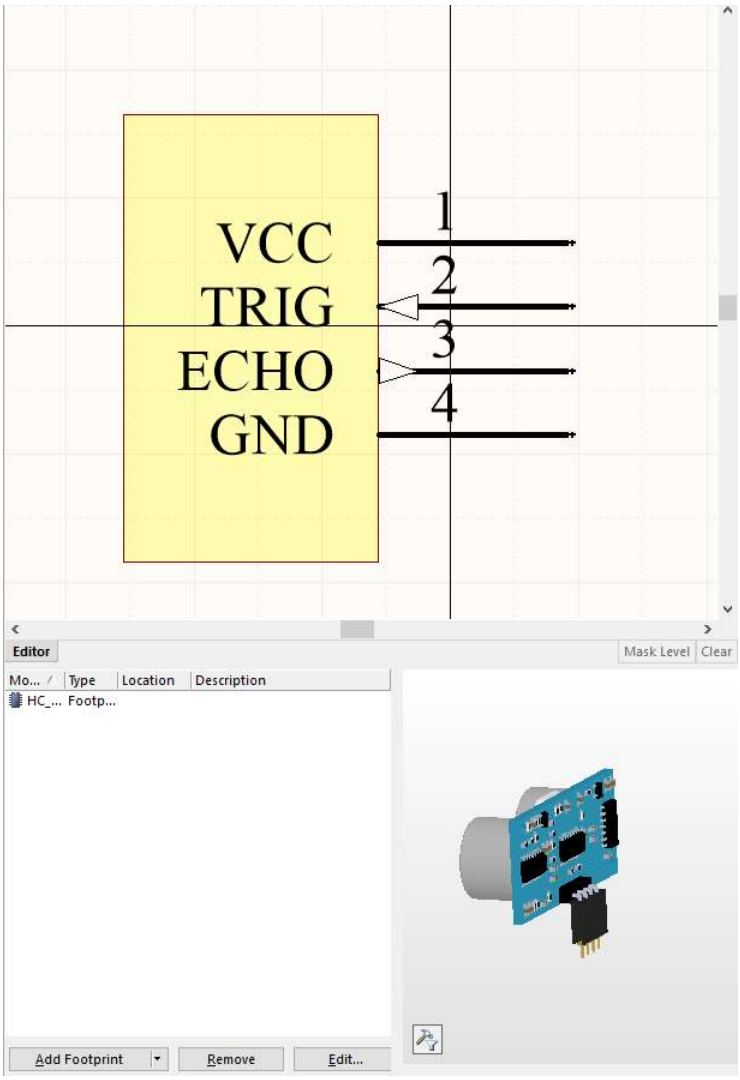


Figura 275. *Footprint* añadido en la librería de esquemáticos, para el sensor ultrasónico HCSR04.



b. Elaboración del esquemático. Usando los componentes de la librería creada, se añaden a un nuevo archivo de esquemático de Altium. Luego se interconectan de acuerdo con el cableado del prototipo de la falda de sensores. Se recomienda trabajar con las Nets de Altium en lugar de hacerlo con wires, ya que el aspecto final es más ordenado y claro. Para el presente módulo se realizaron dos placas que van montadas una sobre otra, interconectadas con *headers*. El esquemático de la placa de módulos, que es la que lleva el Teensy, la IMU, y el módulo ESP8266, se adjunta en el Anexo b de Swarm Robotics.. El esquemático de la placa de la falda de sensores ultrasónicos, que además de ellos incluye su lógica combinacional, se adjunta en el Anexo c de Swarm Robotics.

c. Diseño del PCB. Se crea un nuevo archivo de documento PCB en Altium. Se verifica primero que en el esquemático las conexiones estén hechas de acuerdo con el diseño, y se validan. Luego se transfieren las conexiones al archivo PCB. Se establece el número de capas de cobre para cada PCB. La placa de módulos es de una cara, y la placa de la falda de sensores ultrasónicos es de dos caras. Utilizando el diseño de la pieza que da soporte a los sensores ultrasónicos, se traza el contorno de la placa de la falda de sensores. El contorno de la otra placa se realiza de manera que sea fácil extraerla. Luego se ubican los componentes dentro del contorno, ordenándolos para aprovechar el espacio disponible, sin que exista un traslape con algún agujero por donde atraviesan los tornillos que dan sostén a toda la estructura. Después se rutean las conexiones entre los componentes.

Figura 276. Diseño del PCB de la placa de la falda de sensores ultrasónicos.

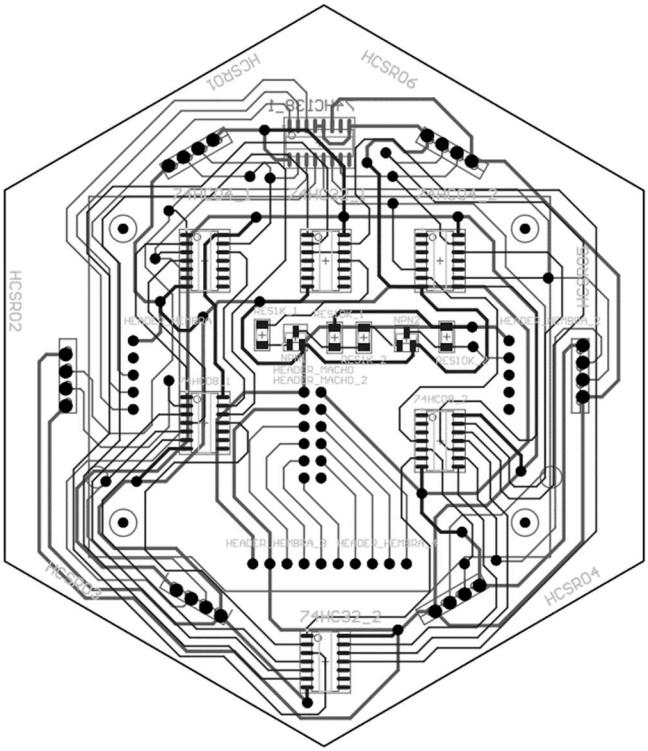


Figura 277. Vista 3D del PCB de la falda de sensores ultrasónicos.

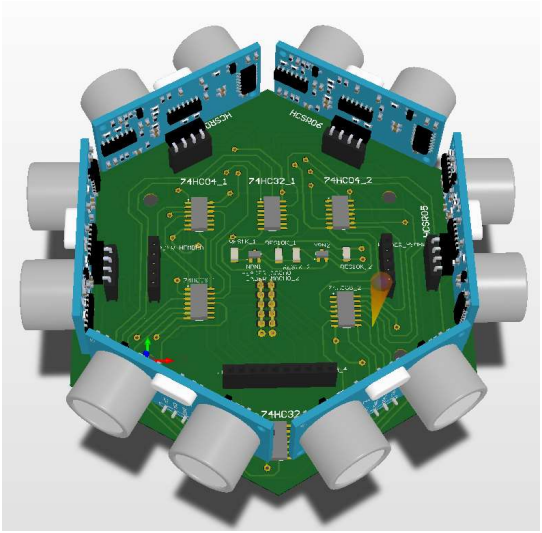


Figura 278. Diseño del PCB de la placa de módulos.

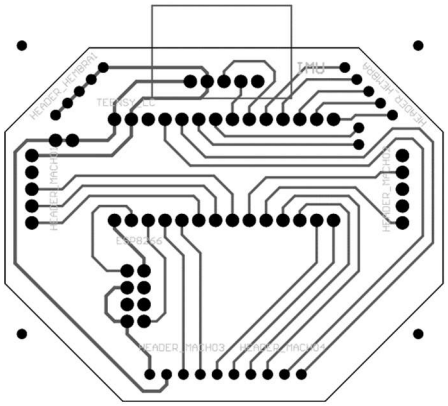
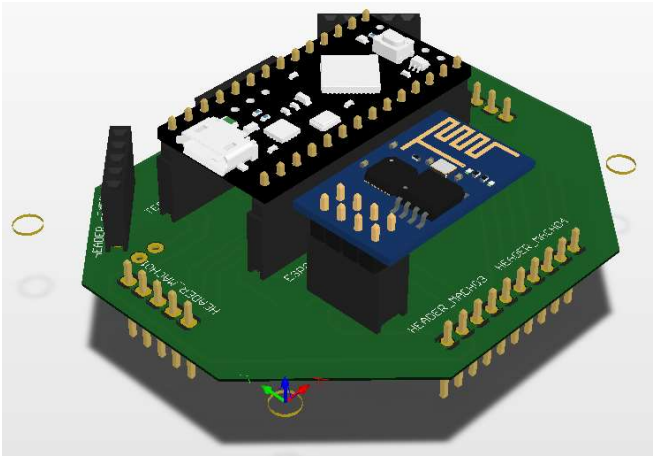


Figura 279. Vista 3D del PCB de módulos.



Finalmente se hacen los cálculos de ancho de pista para las conexiones del PCB. La longitud de las pistas ruteadas se mide con Altium, que tiene la opción de seleccionar todos los segmentos que están en la misma pista, y medir la longitud total. El ancho de pista varía dependiendo de la corriente que debe pasar por ellas, de su longitud y de la caída de voltaje permitida. Para esto se utilizó la herramienta en línea ANSI PCB *Trace Width Calculator*. A continuación, se muestra el parámetro de cobre de las placas que se utilizaron, que son las que el Departamento de Ingeniería Electrónica y Mecatrónica tiene disponibles para el fresado de PCB. También se muestran otros parámetros necesarios para el cálculo del ancho de pista. Las señales que terminan en asterisco (*) representan señales complementadas (lógica invertida). [27]

Grosor de la placa: 0.5 oz/ft2.

Incremento de temperatura: 5°C.

Temperatura ambiente: 25°C.

Voltaje pico: 5V.

Tabla LVII. Resumen del cálculo de ancho de pistas para el PCB de la falda de sensores.

Net	Longitud (mm)	Corriente medida/corriente de diseño (mA)	Ancho (cálculo en página) (mil)	Ancho en diseño (en PCB) (mil)
GND	626.615	80 / 400	10.44	20
5V	583.632	80 / 400	10.44	20
TRIG1*	34.851	15/75	1.04	15
ECHO1	62.167	15/75	1.04	15
TRIG2*	87.628	15/75	1.04	15
ECHO2	35.675	15/75	1.04	15
TRIG3*	120.811	15/75	1.04	15
ECHO3	47.652	15/75	1.04	15
TRIG4*	117.165	15/75	1.04	15
ECHO4	55.467	15/75	1.04	15
TRIG5*	85.031	15/75	1.04	15
ECHO5	50.576	15/75	1.04	15
TRIG6*	36.823	15/75	1.04	15
ECHO6	87.996	15/75	1.04	15
Otras señales	Variable	15/75	1.04	15

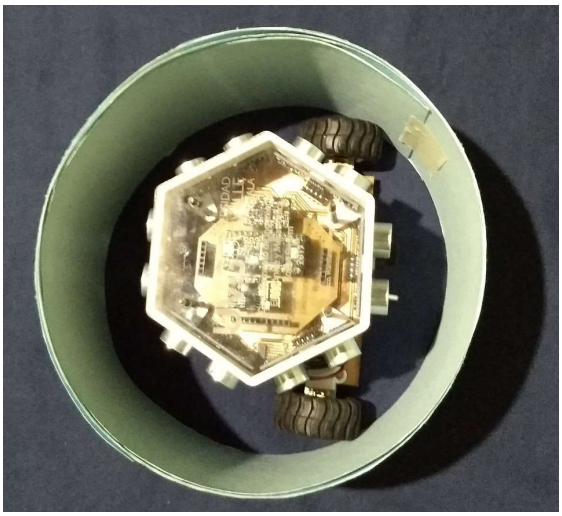
Se seleccionaron 2 anchos de track: 15 y 20 mil. Dependiendo del cálculo dado por la página web, se ubica el ancho del *track* en uno de esos valores. Esto se realizó debido a que anchos menores de pista pueden ocasionar que la fresadora desgaste por completo alguna sección de cobre, y deje un grosor no deseado, o incluso que no haya continuidad en una pista. Por eso es mejor sobredimensionar el ancho, y dejarle un margen a la fresadora por cualquier inconveniente. Para la placa de módulos se dejan los mismos anchos de pista, 20 mil para los voltajes de alimentación y 15 mil para las señales de control.

3. Programación del microcontrolador para el manejo de sensores. Para la toma de datos de la IMU MPU6050 se utilizó la librería MPU6050_raw. Recopila los datos de giro y acelerómetro, como valores sin procesar (*raw*) de 32 bits con signo. Números negativos representan aceleración en una dirección, y los positivos van en la otra dirección. Es una librería probada en Teensy 3.2 y Teensy 3.6, diseñada para la tercera generación de Teensy, incluyendo al LC. Básicamente accede a los registros de la IMU, les asigna valores, e inicia una interfaz de comunicación I2C con el Teensy. La librería para la configuración de registros se puede obtener del repositorio de Github MPU6050de Simon Levy (https://github.com/simondlevy/MPU6050). Para visualizar los datos, se hizo el respectivo procesamiento para obtener los ángulos *roll, pitch*. Con la información de ángulos y aceleración se dibujó en Matlab un cubo en 3D, para agilizar el procesamiento de gráficos, que fue la representación de la IMU. El código empleado para generar el cubo se adjunta en los Anexos e y f de Swarm Robotics. Se guarda un video con los *frames* generados por Matlab. [62]

La falda de sensores es controlada por un decodificador de 3 a 8. Por ello se programaron solo 3 pines digitales de salida en el Teensy para los *Trigger*, y con la lógica combinacional implementada se logra tener un solo pin de Echo como entrada digital al Teensy. Se agrega prioridad al puerto D, que es donde se ubica el pin 7 del Teensy, que se usó como Echo. Con ello se logra que la interrupción se realice en el momento adecuado, y que no pase mucho tiempo hasta que el microcontrolador calcule la duración del pulso. El código para obtener datos de sensores ultrasónicos y de la IMU se adjunta en el Anexo d de Swarm Robotics. Para las pruebas con la IMU se obtuvo información a través del puerto USB del Teensy.

4. Calibración del conjunto de sensores ultrasónicos. Se colocaron muestras radialmente simétricas, círculos de papel. El radio se pudo variar, para apreciar el funcionamiento a diferentes distancias. Se recopilaron datos vía WiFi.

Figura 280. Diseño experimental para la calibración de la falda de sensores.



5. Pruebas con sistemas de control. El diseño experimental para la prueba del sistema fue una superficie cubierta con papel, y referencias intercambiables. A continuación, se muestra la superficie sobre la que se movió el robot durante las pruebas del algoritmo de control. Se muestra un metro, con una longitud desplegada de 1m como referencia de distancia.

Figura 281. Superficie para las pruebas del algoritmo de control.

En el Anexo g de Swarm Robotics se encuentra el código de Matlab que se programó para realizar las pruebas con el algoritmo de control que implementa el modelo uniciclo y un sistema de control proporcional para el error en la velocidad angular. Para ello se usó información de *encoders*, uno por cada llanta. La información se recolectó vía WiFi, a través del módulo ESP8266, y por la misma vía se enviaron órdenes a los motores después de procesar la información de los *encoders*.

6. Resultados

a. PCB falda de sensores

Figura 282. PCB de la falda de sensores ultrasónicos, después de ser fresados.

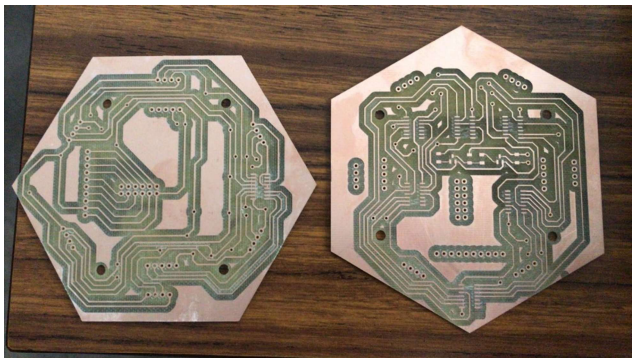


Figura 283. PCB de módulos, después de ser fresado.

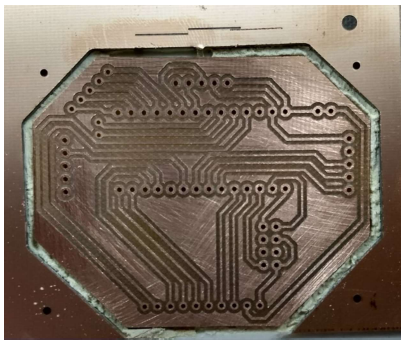


Figura 284. Primera etapa de soldadura del PCB de sensores ultrasónicos.

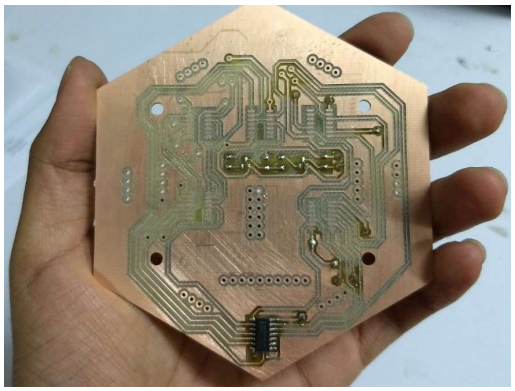


Figura 285. Sensores ultrasónicos y pieza de soporte montados sobre el PCB.

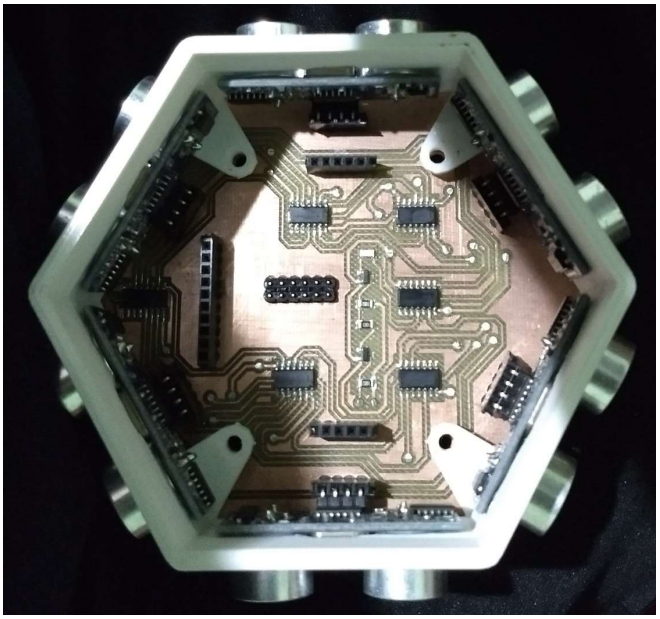


Figura 286. Parte inferior del PCB de sensores ultrasónicos.

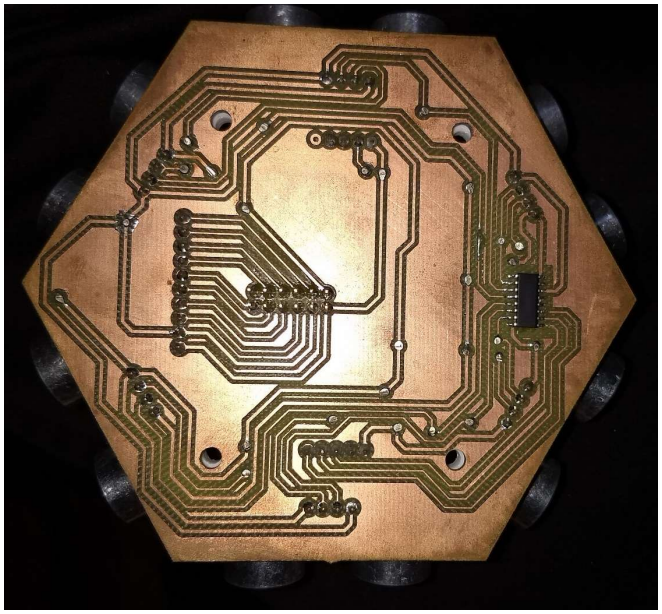


Figura 287. PCB de módulos, con el Teensy, la IMU y el ESP8266.

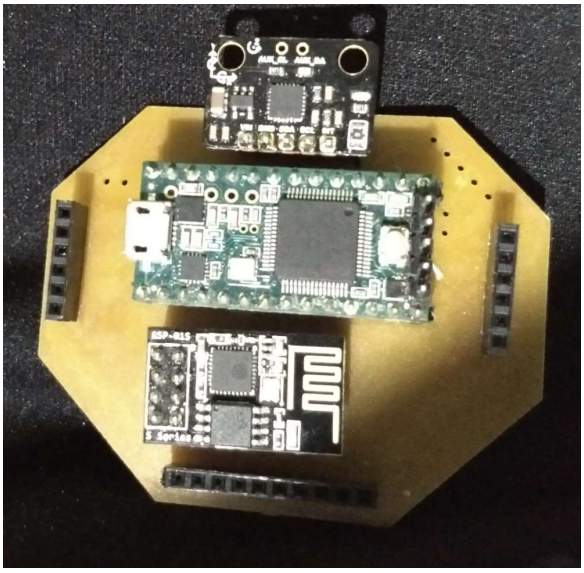


Figura 288. Parte inferior del PCB de módulos.

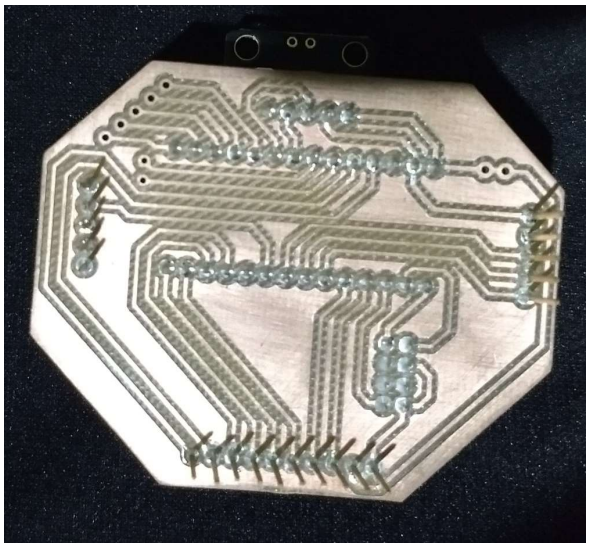


Figura 289. PCB de módulos montado sobre PCB de sensores ultrasónicos.

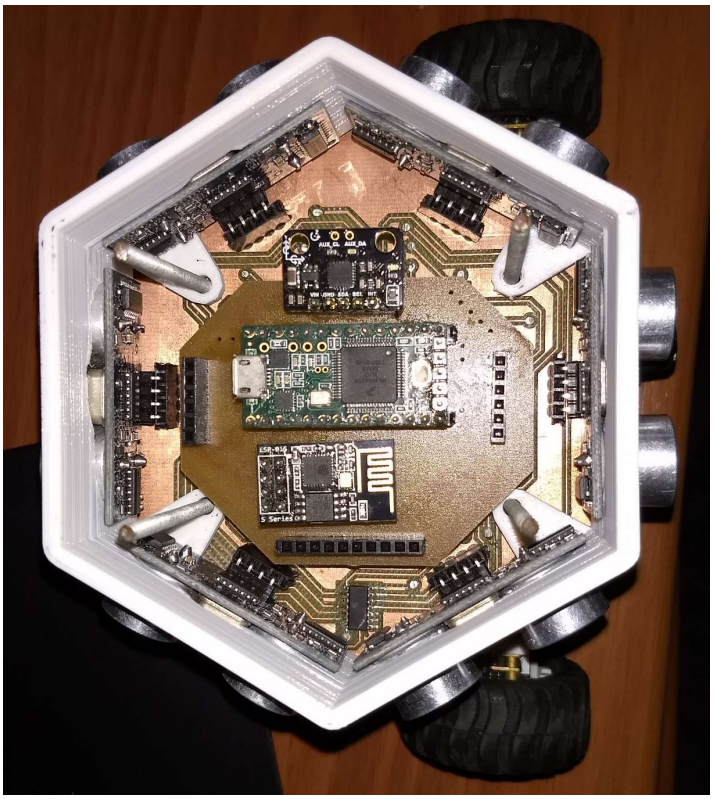
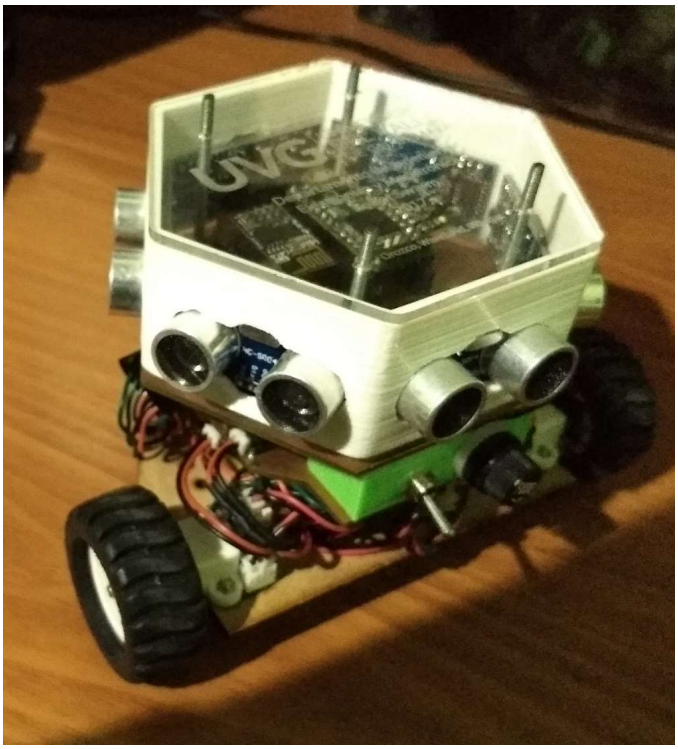


Figura 290. Vista final del robot, con los PCB montados sobre la estructura.



En la Figura 282 y Figura 283 se observan los PCB después de ser fresados. En las primeras placas no hubo mayor problema. Sin embargo, al replicar el diseño en otras dos placas más, la fresadora tuvo inconvenientes, y ahí se nota la importancia de dejar un mayor ancho de pista que el calculado, para dejarle margen a que desgaste más cobre sin correr el riesgo de perder continuidad en las pistas. Para interconectar los PCB de módulos y de la falda de sensores se utilizaron *headers* hembra de patas largas. Esto se hizo con el fin de dejar puntos de prueba, para medir señales y comprobar que los voltajes de alimentación llegaran hasta la placa superior. La conexión con la placa de potencia se hizo mediante *headers* macho de patas largas.

El reto al soldar componentes de superficie es que debe ser una soldadura hecha con más cuidado para no quemar el componente, y para que se coloque el estaño necesario. De lo contrario, el exceso de calor puede ser un problema serio para cualquier circuito integrado SMD, y se arruina. Las vías se realizaron con cable soldado de ambos lados, y funcionaron correctamente. Los PCB cumplieron sus funciones. El de la falda de sensores controló a los HCSR04, y el de módulos sirvió para conectar el Teensy, la IMU y el WiFi, centros de comunicación y de manejo de toda la estructura.

El diseño del contorno del PCB de módulos fue útil para extraer esa placa. Esto es necesario para cargar nuevos programas al Teensy, cuando se esté depurando el firmware para corregir errores u optimizarlo. Dejar espacio para los dedos es una buena práctica al momento de diseñar PCB modulares de baja complejidad, como en este caso.

b. Mediciones de falda de sensores e IMU. Las mediciones de la IMU se obtuvieron como valores de 32 bits, que al escalarlos se pueden representar como una fracción de la aceleración de la gravedad terrestre (g). Para la falda de sensores se realizaron 100 mediciones para cada distancia, y un muestreo de 0.3s. Los datos se obtuvieron en Matlab via WiFi y se analizaron en Excel con la herramienta Histograma, de Análisis de Datos.

Tabla LVIII. Frecuencias para distancias medidas con una referencia a 4 cm.

Distancia medida	Frecuencia					
	Sensor1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6
Entre 3 cm y 4 cm	92	83	82	86	35	88
Entre 4 cm y 5 cm	0	7	0	4	46	0
85 cm	8	10	18	10	19	12

Figura 291. Tiempos medidos por los ultrasónicos, referencia a 4 cm.

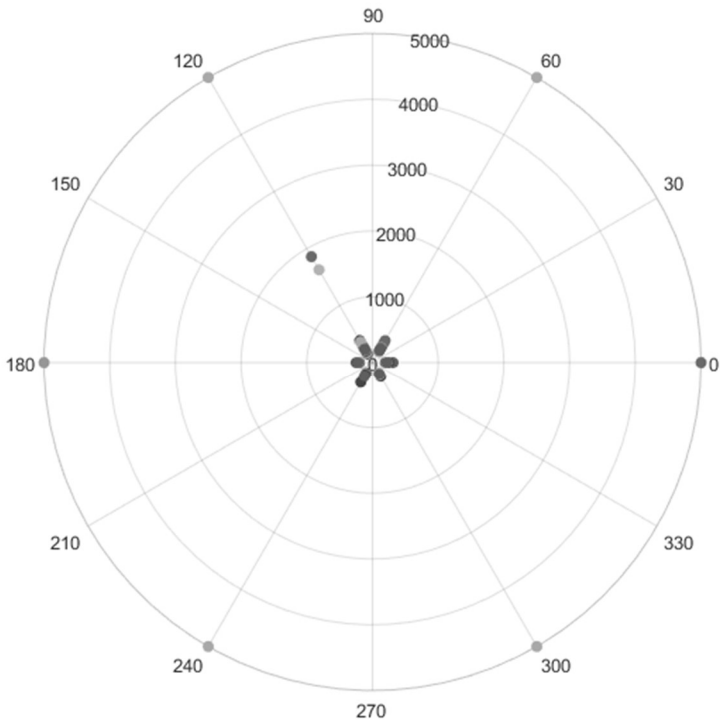


Tabla LIX. Frecuencias para distancias medidas con una referencia a 10 cm.

Distancia medida	Frecuencia					
	Sensor1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6
Entre 9 cm y 10 cm	80	0	2	6	8	87
Entre 10 cm y 11 cm	7	91	85	70	75	4
85 cm	13	9	13	24	17	9

Figura 292. Tiempos medidos por los ultrasónicos, referencia a 10 cm.

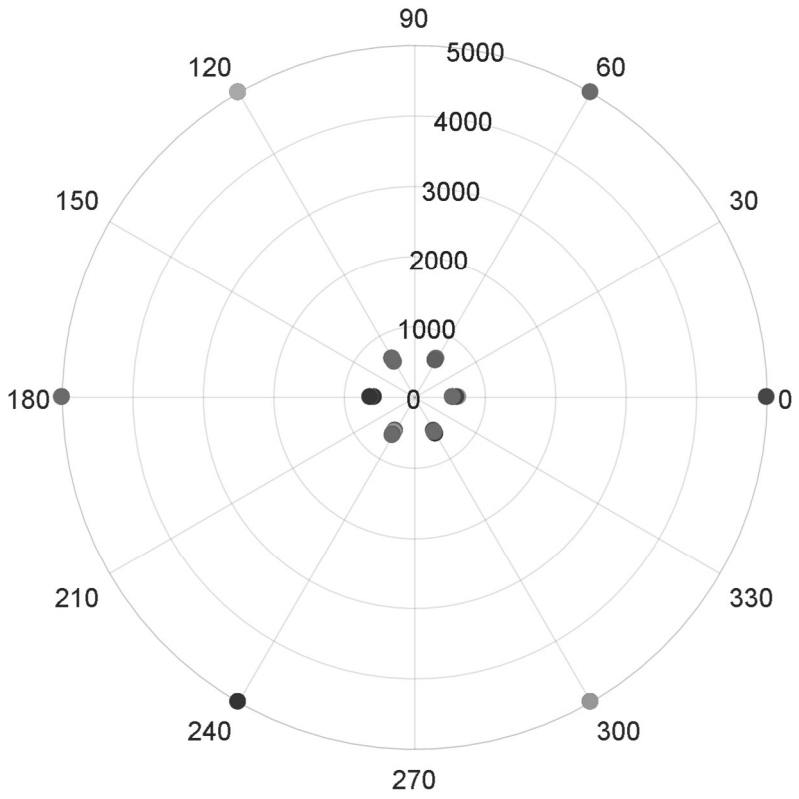


Tabla LX. Frecuencias para distancias medidas con una referencia a 24 cm.

Distancia medida	Frecuencia					
	Sensor1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6
Entre 22 cm y 23 cm	7	7	71	7	3	35
Entre 23 cm y 24 cm	75	79	12	71	83	37
85 cm y otros	18	14	17	22	14	28

Figura 293. Tiempos medidos por los ultrasónicos, referencia a 24 cm.

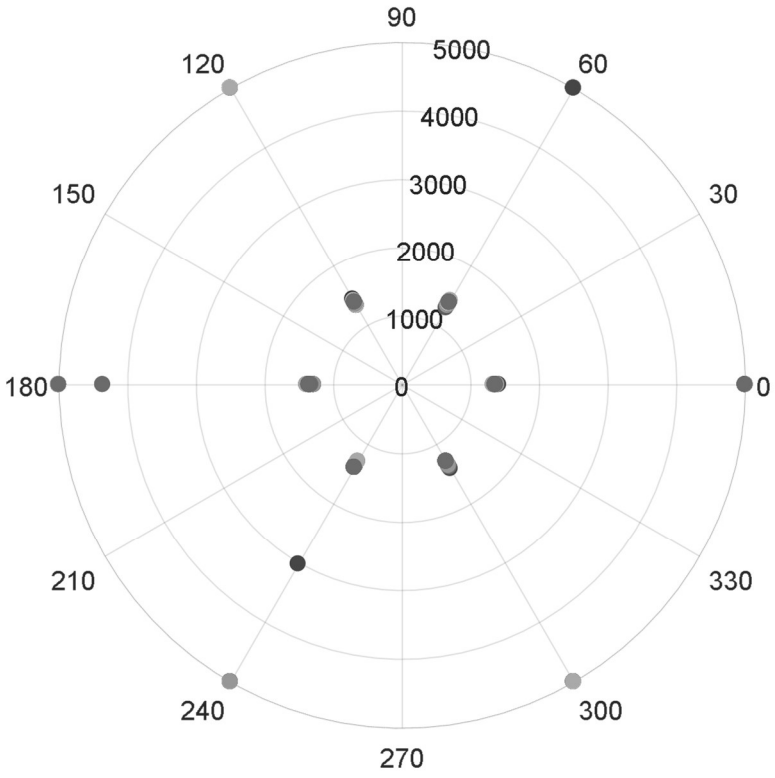
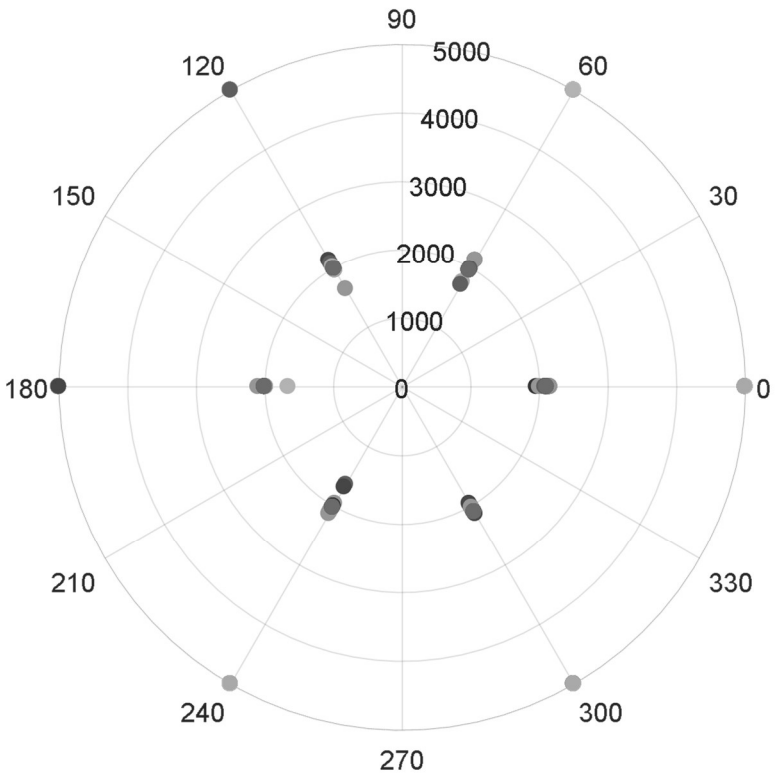


Tabla LXI. Frecuencias para distancias medidas con una referencia a 34 cm.

Distancia medida	Frecuencia					
	Sensor1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6
Entre 34 cm y 36 cm	53	85	84	84	86	7
Entre 36 cm y 38 cm	31	6	9	1	4	81
85 cm y otros	16	9	7	15	10	12

Figura 294. Tiempos medidos por los ultrasónicos, referencia a 34 cm.



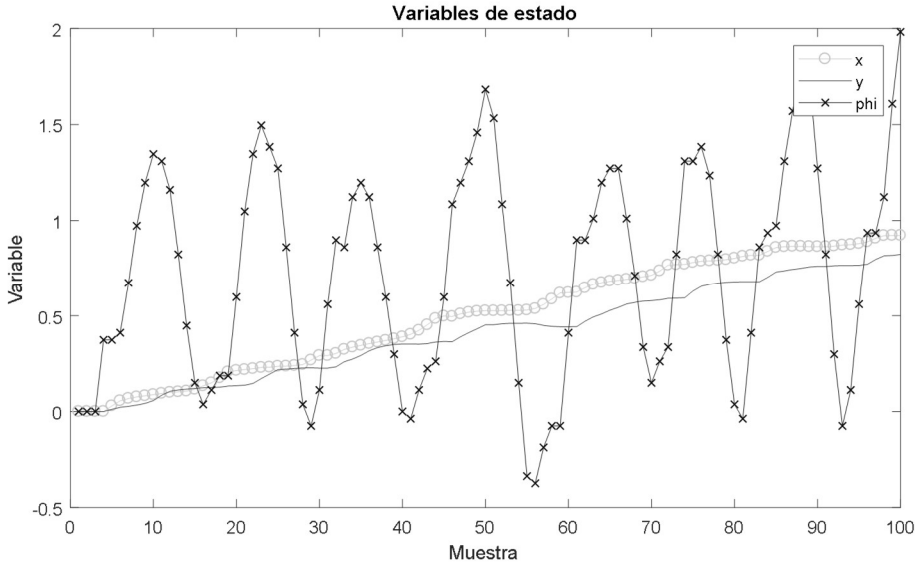
La IMU devuelve datos de 32 bits, lo que representa un alto grado de sensibilidad. Dependiendo del rango configurado con los registros, al escalar los datos se obtiene un resultado relativo a la gravedad terrestre (g). La comunicación i2C necesita un tiempo de pausa entre cada solicitud de información, por lo que es necesario tomarlo en cuenta para no alterar la toma de datos.

De la Tabla LVIII a la Tabla LXI se observa que, en cada 100 muestras de toma de datos de los sensores ultrasónicos, siempre aparecerá un valor atípico de proximidad. Este es el *timeout* programado en

el Teensy, que es un tiempo de espera máximo que el microcontrolador le da a la medición del Echo. Si durante ese tiempo no recibe un pulso de retorno, el Teensy ignora el retorno del pulso y devuelve un ancho de pulso con el valor de ese *timeout*. En el presente megaproyecto se espera como máximo 5000 uS, que equivalen a 85 cm. Los puntos que aparecen en la orilla de las gráficas polares de la Figura 291, Figura 292, Figura 293 y Figura 294 ilustran el tiempo máximo de espera. Sin embargo, no siempre que haya un *timeout* significa que no hay un obstáculo enfrente. Estas mediciones se obtuvieron a través del módulo WiFi. Cuando se usa un puerto serial USB para obtener los datos, no aparece el *timeout*, las mediciones son más consistentes. Con una distancia de 4cm hacia todos los sensores, hay un error máximo de 19%, como se observa en la Tabla LVIII. Conforme la distancia crece, no hay una tendencia clara, pero se aprecia que el error crece. A 10 cm, hay máximo 24 muestras erróneas, como dice la Tabla LIX. En la Tabla LX se observa que hay 28 muestras erróneas, en el peor caso, a 24 cm de distancia. Y en la Tabla LXI se observan 16 mediciones incorrectas a 34 cm de distancia. En estas variaciones pudo influir la geometría del papel. A pesar de que se intentó que con tiras de papel se formara un cilindro alrededor del robot, no es un material que se mantiene en su misma forma por un lapso de tiempo. Ligeras inclinaciones en el papel provocan que las ondas ultrasónicas reboten de forma extraña, y por tanto se obtienen tiempos de rebote distintos a los esperados. Es recomendable revisar la geometría del objeto a colocar alrededor de los ultrasónicos al momento de realizar la calibración de los sensores ultrasónicos.

c. Controlador proporcional

Figura 295. Prueba 1 controlador proporcional.



Para la prueba 1 del controlador proporcional se utilizó kP_omega = 50; 100 muestras; T = 0.3; punto de partida: (0,0); ángulo inicial: 0 rad; meta: (1, 1).

Figura 296. Secuencia de imágenes de la trayectoria seguida durante la prueba 1 de control proporcional

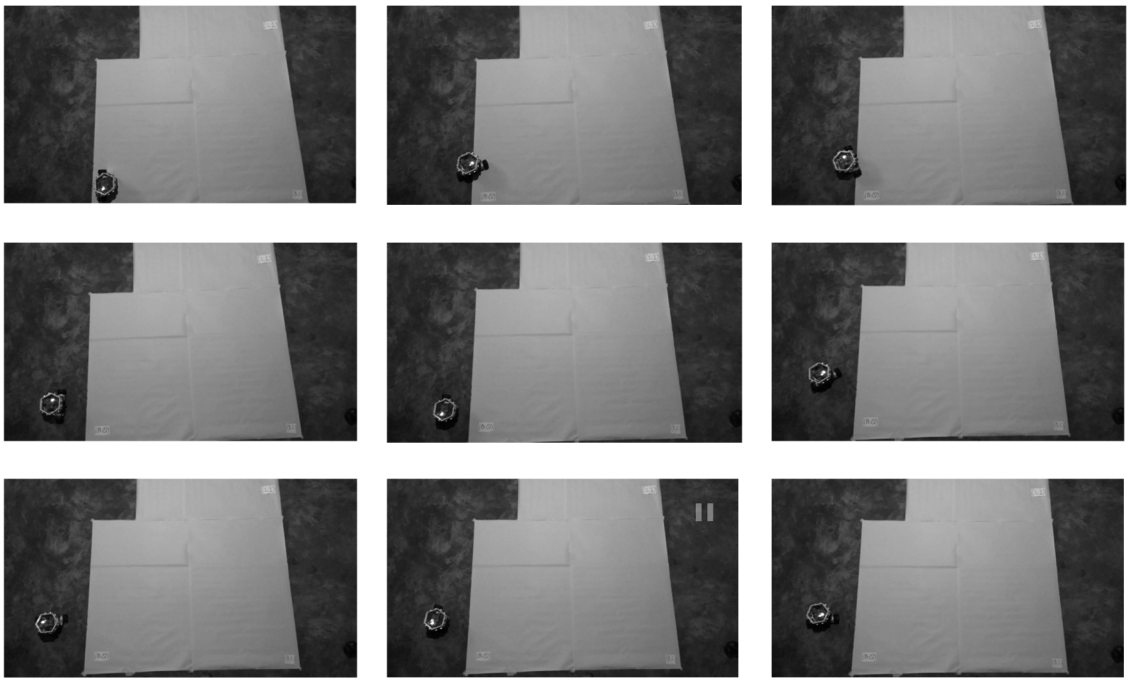
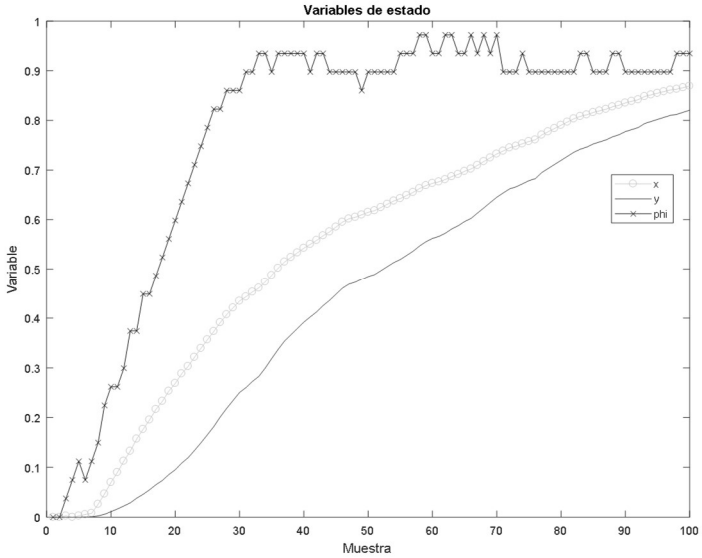


Figura 297. Prueba 2 controlador proporcional.



Para la prueba 2 del controlador proporcional se utilizó kP_omega = 10; 100 muestras; T = 0.3; punto de partida: (0,0); ángulo inicial: 0 rad; meta: (1, 1).

Figura 298. Secuencia de imágenes de la trayectoria seguida durante la prueba 2 de control proporcional

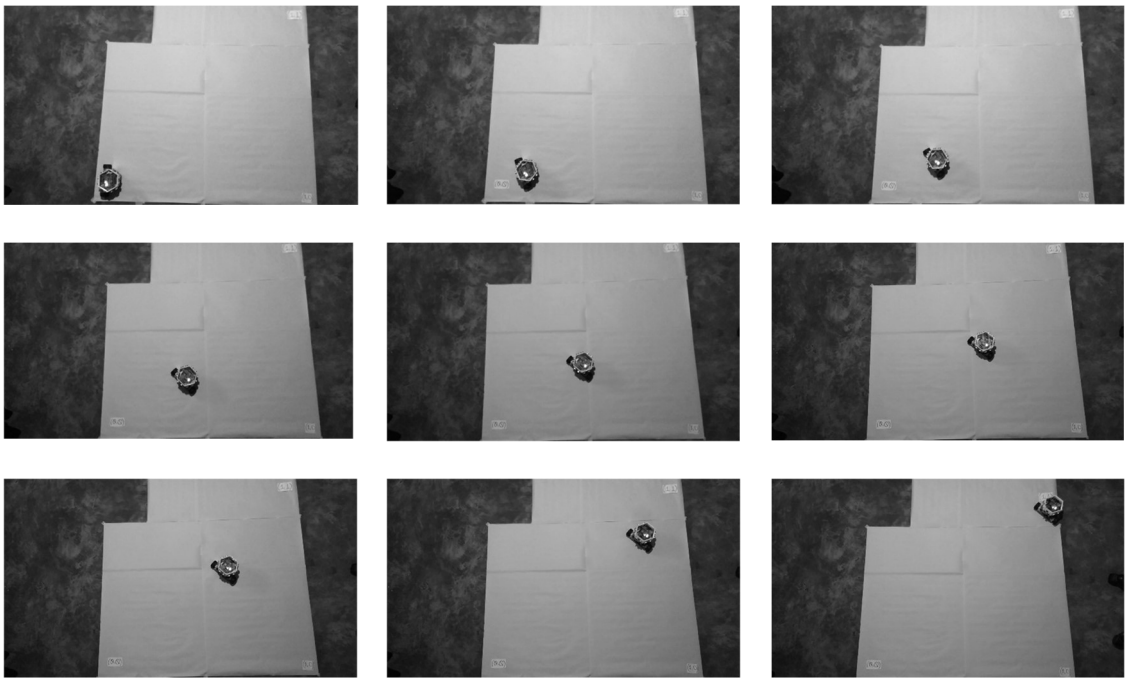
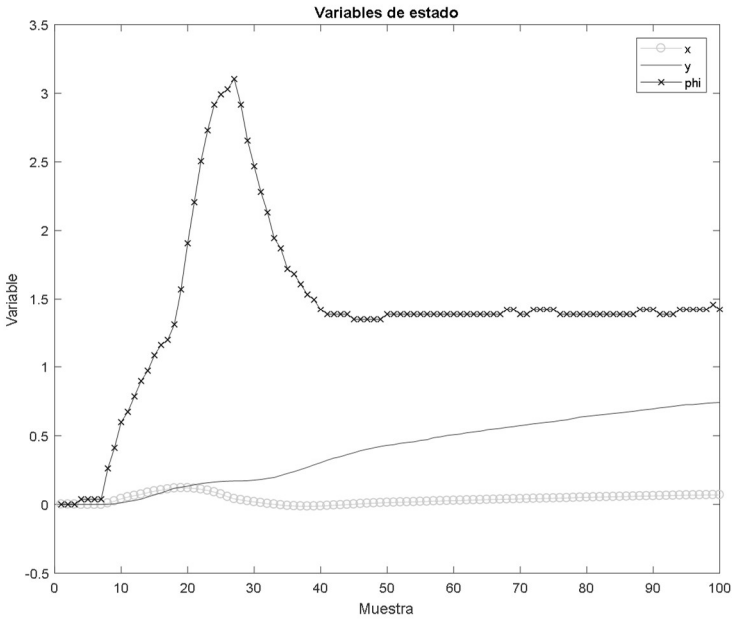


Figura 299. Prueba 3 controlador proporcional.



Para la prueba 3 del controlador proporcional se utilizó kP_omega = 10; 100 muestras; T = 0.3; punto de partida: (0,0); ángulo inicial: 0 rad; meta: (0.1, 1).

Figura 300. Secuencia de imágenes de la trayectoria seguida durante la prueba 3 de control proporcional

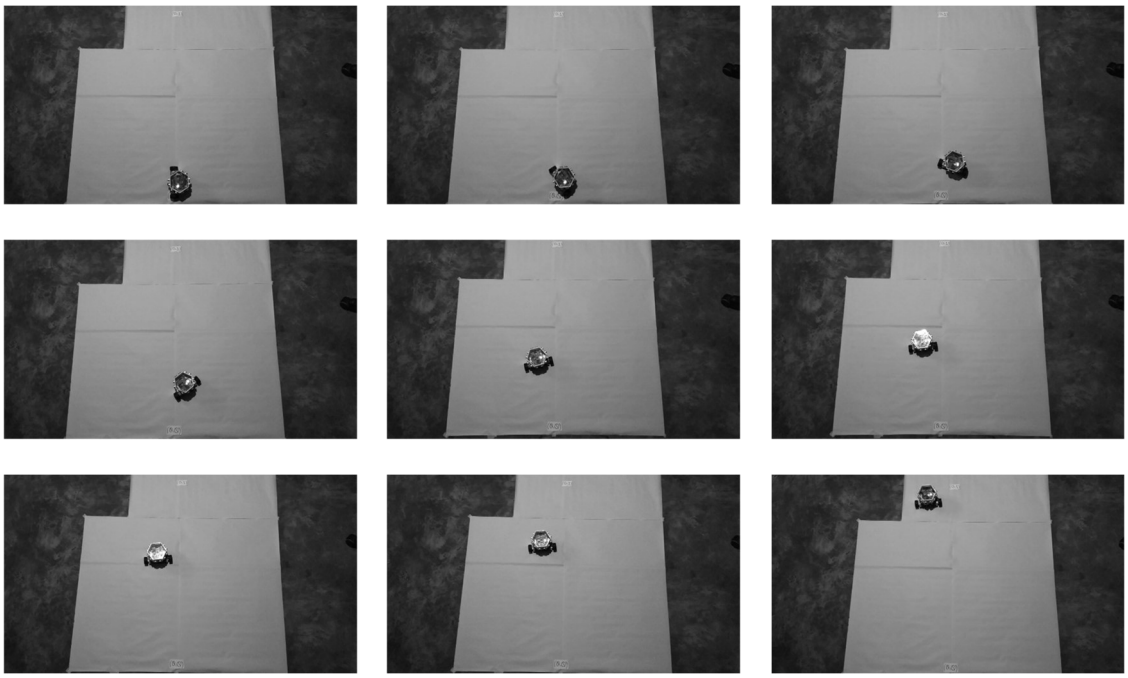
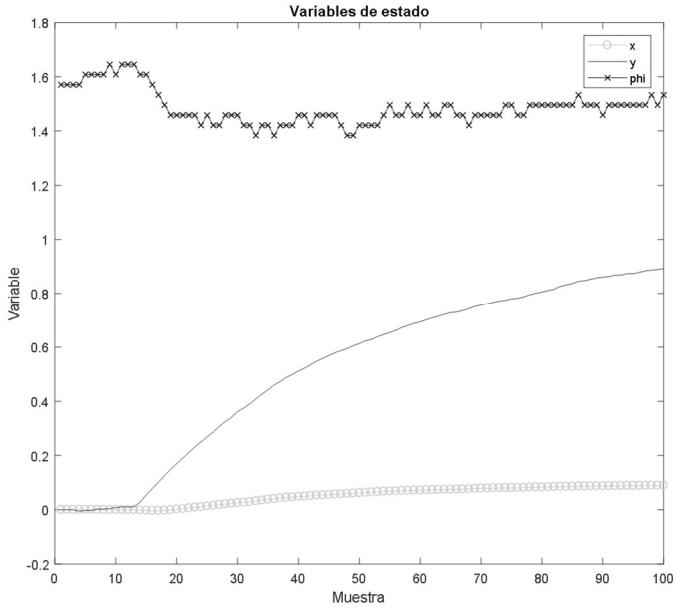


Figura 301. Prueba 4 controlador proporcional.



Para la prueba 4 del controlador proporcional se utilizó kP_omega = 10; 100 muestras; T = 0.3; punto de partida: (0,0); ángulo inicial: pi/2 rad; meta: (0.1, 1).

Figura 302. Secuencia de imágenes de la trayectoria seguida durante la prueba 4 de control proporcional

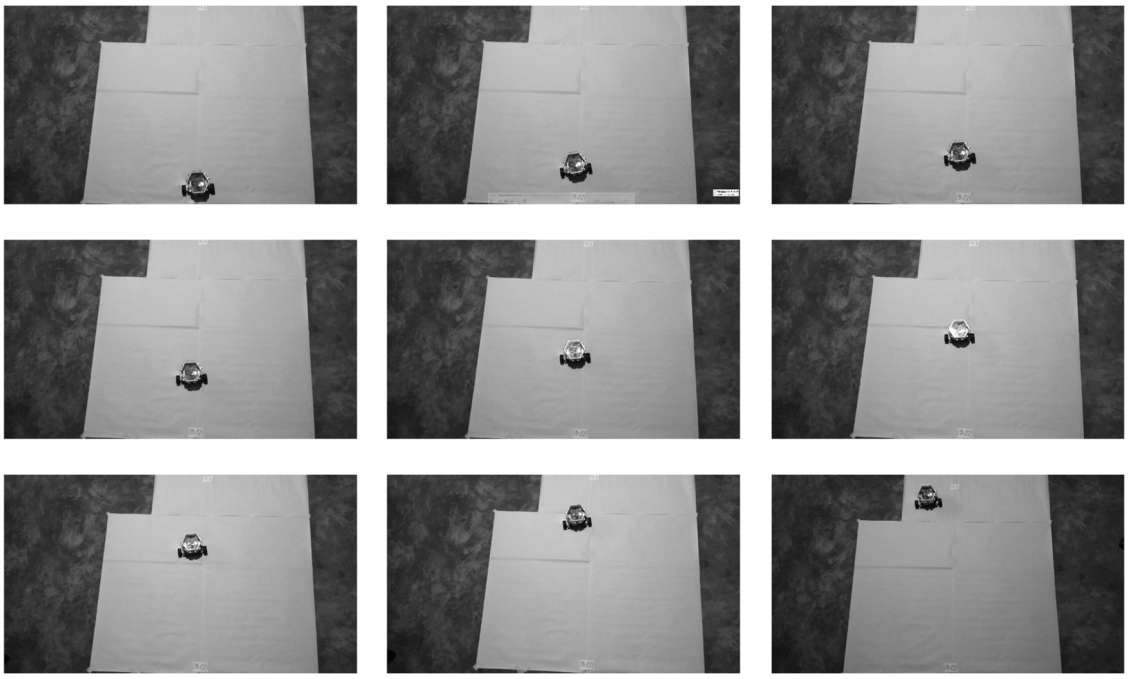
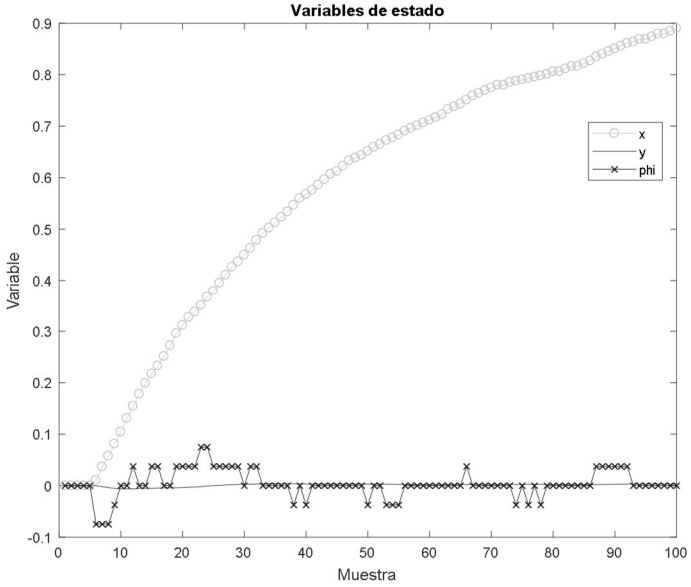


Figura 303. Prueba 5 controlador proporcional



Para la prueba 5 del controlador proporcional se utilizó kP_omega = 10; 100 muestras; T = 0.3; punto de partida: (0,0); ángulo inicial: 0 rad; meta: (1,0).

Figura 304. Secuencia de imágenes de la trayectoria seguida durante la prueba 5 de control proporcional

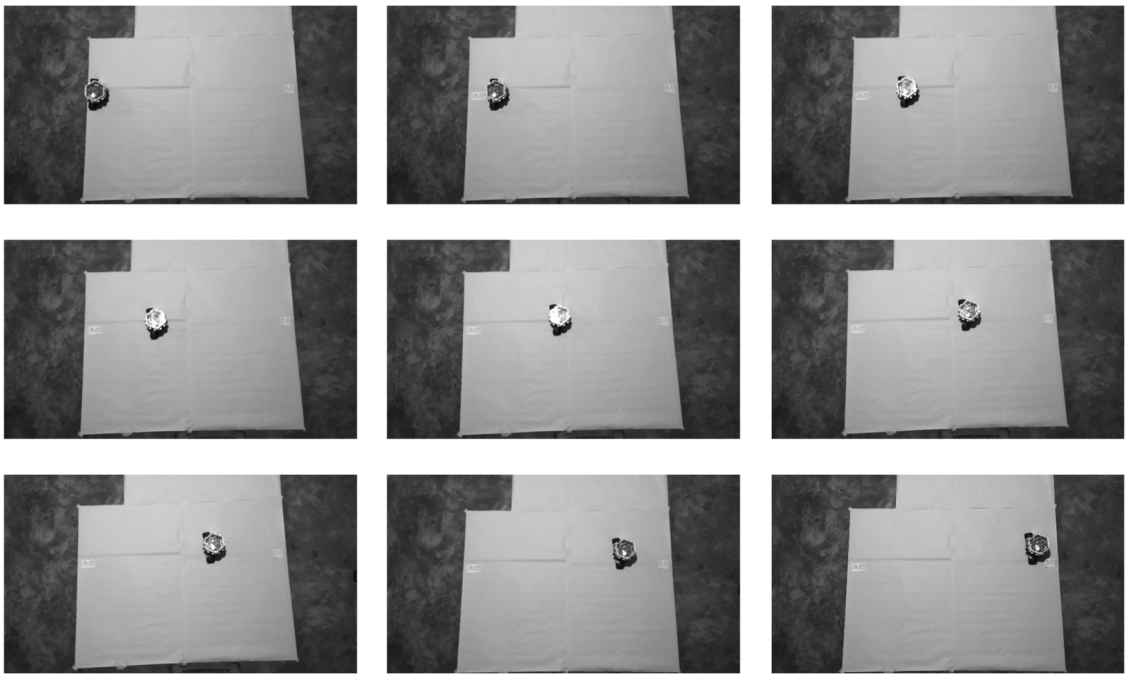
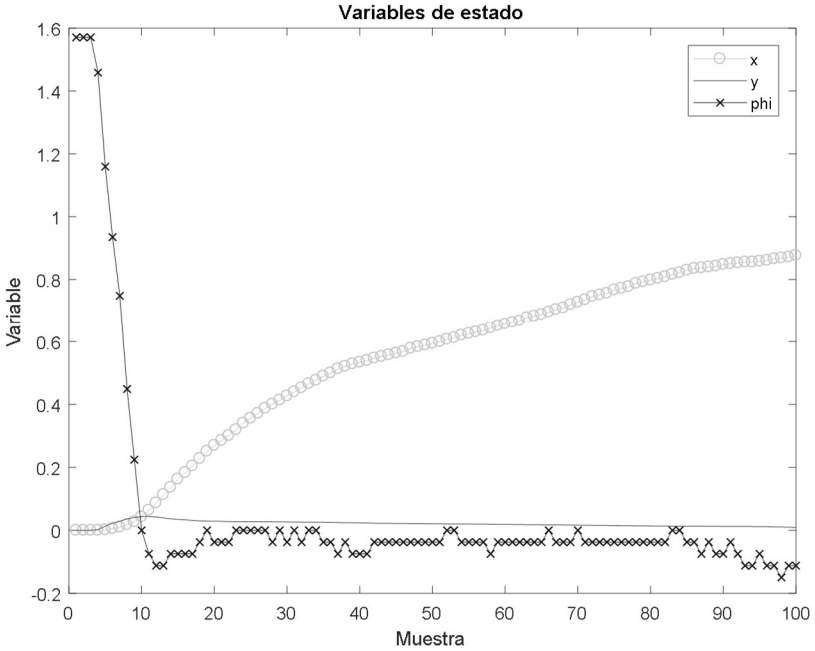
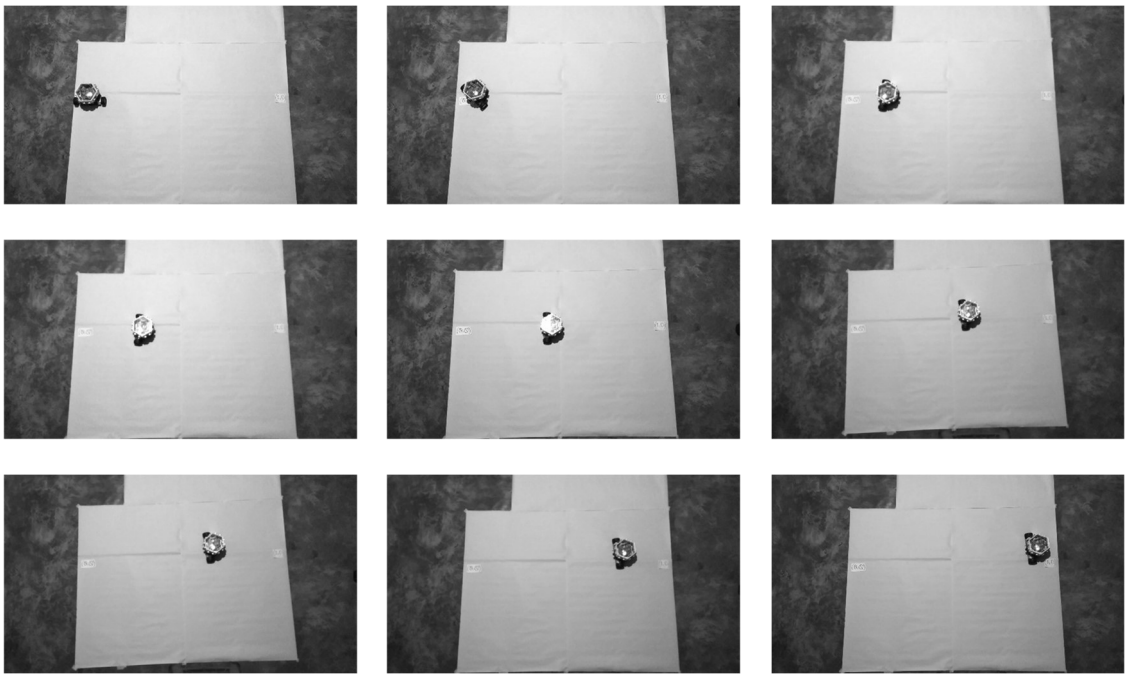


Figura 305. Prueba 6 controlador proporcional.



Para la prueba 6 del controlador proporcional se utilizó kP_omega = 10; 100 muestras; T = 0.3; punto de partida: (0,0); ángulo inicial: pi/2 rad; meta: (1,0).

Figura 306. Secuencia de imágenes de la trayectoria seguida durante la prueba 6 de control proporcional



El valor del *Duty Cycle* del PWM que se envía al motor como control de velocidad se limitó entre 180 y 230. El primero representa lo más rápido que puede ir, y el segundo representa lo más lento. De esa forma se implementó en otro módulo del presente megaproyecto.

Se inició con una ganancia de error proporcional de 50. Sin embargo, hay *overshoot* en el ángulo, como se observa en la Figura 295. La diferencia entre el ángulo deseado y el ángulo actual se amplifica más de lo que debería, y por tanto, la velocidad angular que se obtiene del control proporcional es grande, y provoca un giro errático. Al bajar la ganancia de error proporcional a 10, el controlador hace su trabajo. En la

Figura 297 se aprecia que x, y tratan de alcanzar la meta establecida, pasando de (0,0) a (1,1), quedando cerca del punto deseado. La secuencia de la Figura 298 muestra que el robot llega cerca del punto deseado.

Con la misma constante se hacen las siguientes pruebas, y el rastreo de la meta se logra con el control proporcional. Sin embargo, cuando se coloca la meta (1,0) partiendo del punto (0,0), surgen problemas. Matlab utiliza la función atan para calcular ángulos a partir de coordenadas x, y. La arcotangente de un ángulo involucra una división entre (y-y1)/(x-x1). Se debe tener cuidado de que la diferencia entre la coordenada x de la meta no sea exactamente igual a la coordenada x de la meta. Si esto pasa, en Matlab ocurre una división entre cero. Conceptualmente, se conoce que la arcotangente de algo dividido entre cero, tiende a pi/2.

Matlab no reconoce esto, y devuelve un NaN (*Not a Number*), generando problemas en el momento en que se quiere utilizar el valor del ángulo a partir de la arcotangente. El comportamiento del robot se altera, volviéndose errático. Por eso es que en la prueba 3 del controlador se hizo que la salida fuera (0,0), pero la meta fue (0,1, 1). De esta forma se garantiza que Matlab no devolverá un ángulo deseado incorrecto, sino que devolverá algo parecido a pi/2. El robot sigue una trayectoria accidentada cuando se le pide que no cambie su coordenada en x, solo que se desplace de forma vertical.

Se debe tener cuidado con la calibración de los *encoders*. Es recomendable verificar que, al girar una llanta, el encoder devuelva dos señales cuadradas. Este problema, unido al *drift* de las llantas cuando el suelo es liso, provoca que el robot esté más lejos de llegar a su referencia. También es importante que la conexión WiFi se mantenga estable por varios minutos, para que el flujo de información entre quien procesa la información (computadora) y quien recibe órdenes y recolecta datos de sensores (robot), se mantenga estable y el algoritmo de control se mantenga funcionando.

IX. SILLA

A. DISEÑO EXPERIMENTAL DEL SISTEMA DE CONTROL

La función que se implementó recibe como parámetro la dirección de giro, este parámetro obtendrá su valor directamente del control inalámbrico. Por lo cual se trabajó con rangos de valores, para asegurarse que, si se desea girar a favor de las agujas del reloj, en contra de las agujas del reloj o simplemente no se está movilizando el control.

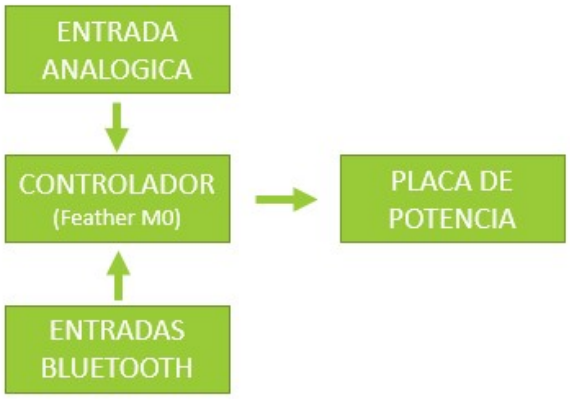
En el siguiente cuadro se describen las entradas y las salidas de la función que se definió para el movimiento de los triángulos.

1. Resultados rutina para los triángulos.

Tabla LXII. Función "motores", entradas y salidas.

Función	Entradas	Salidas
motores	Control: tendrá un valor que varía de 0 a 1024. Dependiendo del valor actual, se sabrá la dirección de giro de los triángulos.	BRK1 y BRK2: si los motores no se deben mover se colocan en 0 para frenar los triángulos, si se deben mover se coloca en 1 para dejar girar los triángulos.
		DIR1 y DIR2: se colocar en 1 si los triángulos deben girar a favor de las agujas del reloj y viceversa.

Figura 307. Diagrama de flujo de las rutinas de los triángulos.



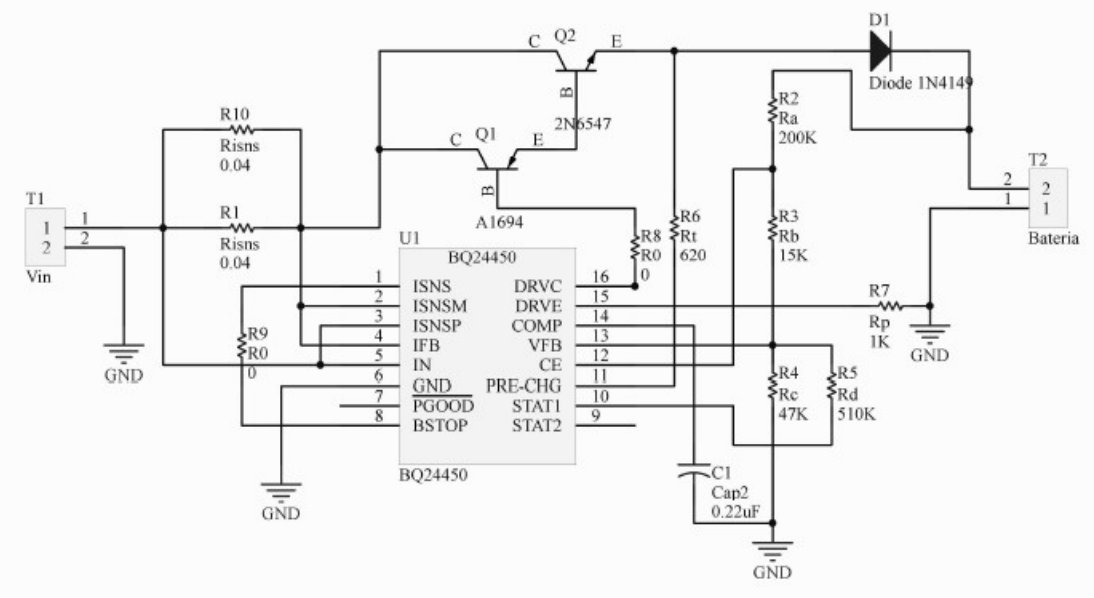
B. DISEÑO EXPERIMENTAL CARGADOR DE BATERÍA

Después de realizar los cálculos necesarios, se obtuvieron los siguientes valores de resistencias. Dichos valores se aproximaron a valore comerciales.

Tabla LXIII. Resistencias calculadas para circuito de carga de batería.

Componente	Valor teórico	Valor comercial
R_A	208kΩ	200kΩ
R_B	15.8kΩ	15.0kΩ
R_C	46.0kΩ	47.0kΩ
R_D	572kΩ	510kΩ
R_T	634Ω	620Ω
R_{ISNS}	0.041Ω	0.010Ω*4

Figura 308. Esquemático del circuito de carga de batería.



El esquemático del circuito encargado de cargar la batería se muestra en la Figura 309, se puede observar que se utiliza la configuración cuasi-darlington. Se le conoce así, ya que utiliza la combinación de un transistor NPN y un transistor PNP. Al contrario, una configuración Darlington como tal maneja dos transistores del mismo tipo.

El diseño del PCB está basado en el esquemático de la Figura 309. Como se mencionó anteriormente, el ancho de las pistas tiene que ser mínimo de 7.38mm, esto para las pistas que conducirán 5 amperios, el resto de pistas trabajaran en el rango de los miliamperios.

Las pistas que trabajaran a 5 amperios son las que corresponden a GND, Vin, Vout, la pista correspondiente al colector del transistor de paso y la pista correspondiente al emisor del transistor de paso. En la Figura 310 y la Figura 311 se puede observar que se utilizaron polígonos para interconectar las pistas mencionadas anteriormente, esto para una mayor disipación de calor

Figura 309. PCB Top Layer.

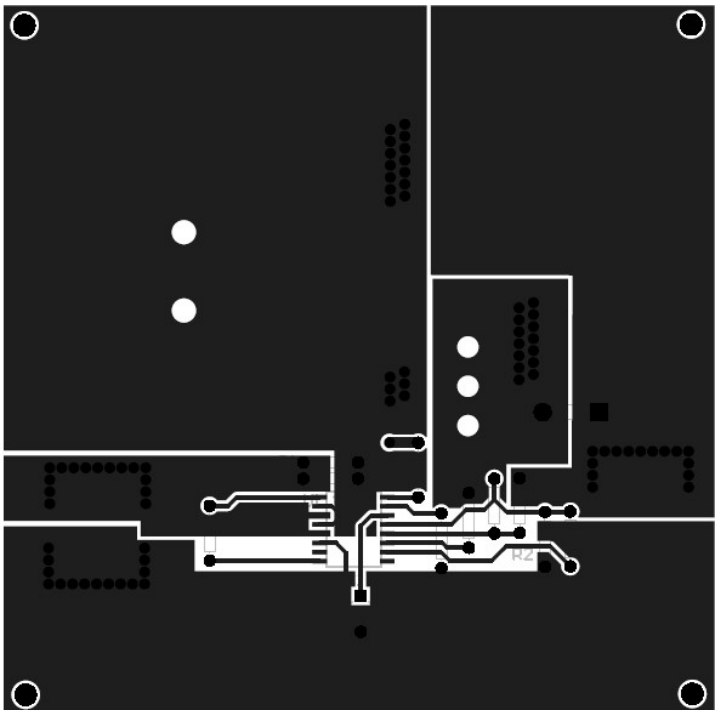


Figura 310. PCB Bottom Layer.

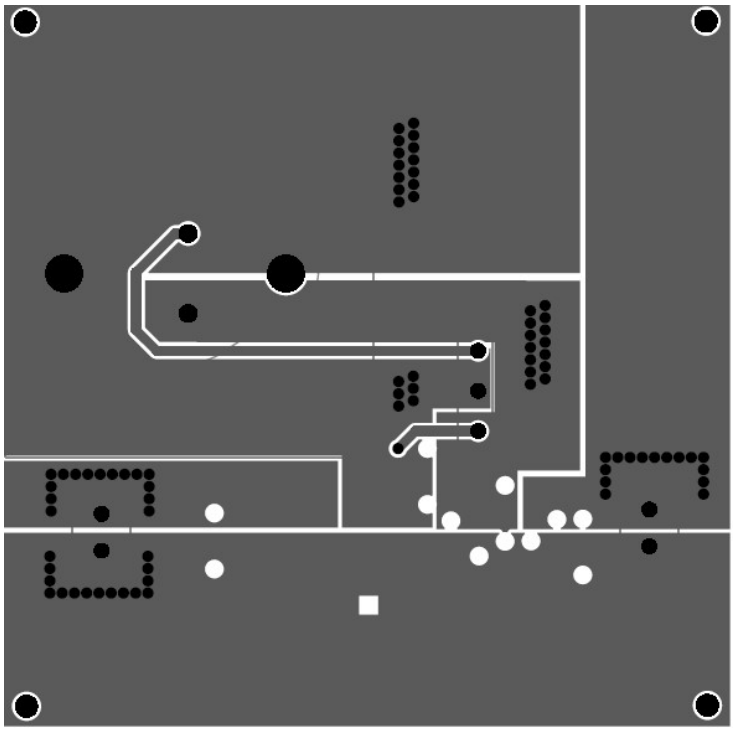


Figura 311. Placa soldada, top layer.

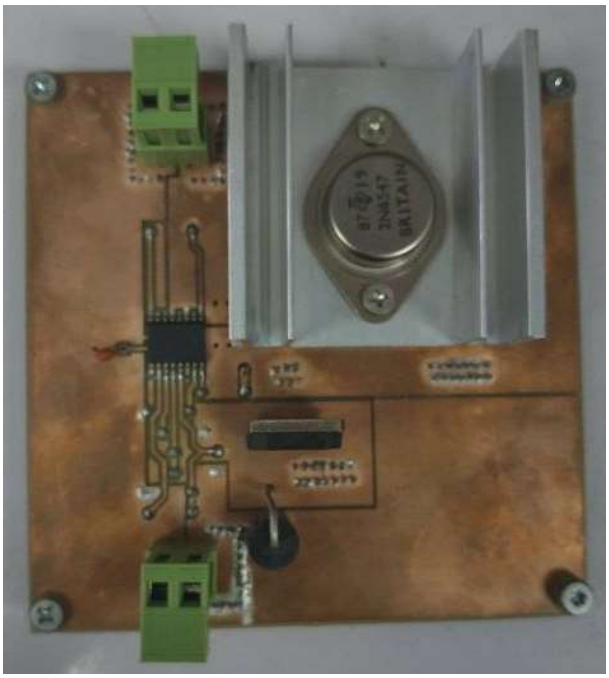
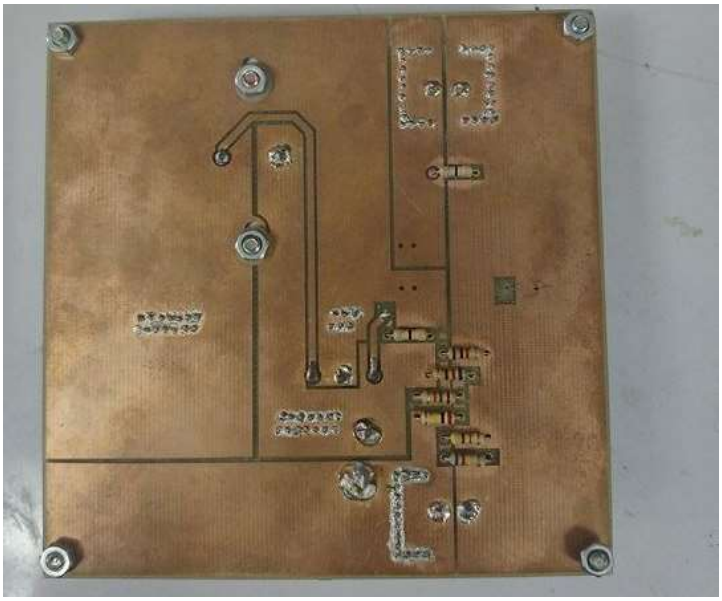


Figura 312. Placa soldada, bottom layer.



X. ROBOT EXPLORADOR

A. DISEÑO EXPERIMENTAL DEL MÓDULO

En el subsistema de potencia y alimentación, se analizó los requerimientos necesarios con respecto al consumo de potencia total que tendrá el robot. Los dos consumidores fueron los motores de corriente continua, encargados del movimiento del explorador y la electrónica implementada, conformada por los controladores y sensores.

Se utilizó dos motores DC de limpia brisas de carro para proveer el movimiento del robot explorador. Por la naturaleza de los mismos, no se tuvo una hoja de datos o alguna ficha técnica para conocer sus características, por lo que se realizó una estimación de la potencia consumida por estos. Para las mediciones del consumo de potencia de los motores se realizaron las siguientes pruebas: primero, se estudió como variaba la potencia dependiendo del voltaje de alimentación del motor sin aplicarle carga. Para esto, se hizo un muestreo de la corriente consumida por el motor al variar el voltaje, con un rango de 1 a 12V con un paso de 0.5V. Seguidamente, se repitió el procedimiento pero incluyendo cargas en el eje del motor, variándolas para encontrar el modelo matemático que describe el consumo de potencia en relación a estas cargas. Se fabricó una plataforma plana, impresa en PLA (ácido poliático), que se acopló al eje del motor, en la cual se montaron las cargas. Para aplicar las cargas se utilizó un kit de masas PASCO, el cual cuenta con masas de 250 y 500 gramos.

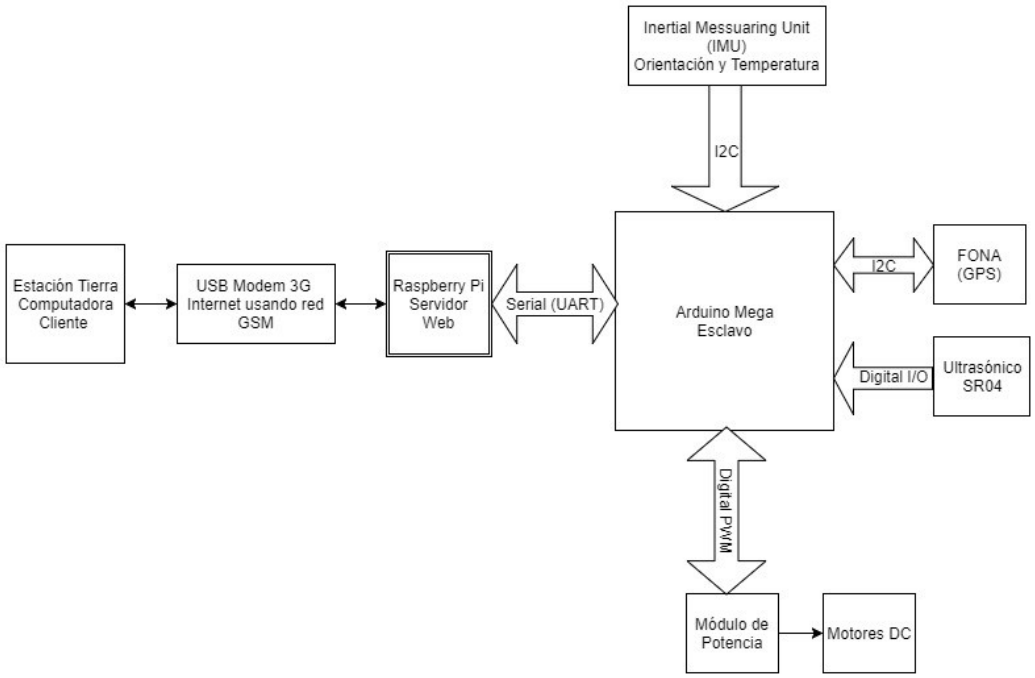
Para el segundo procedimiento mencionado, se fijó un valor de voltaje y se midió la corriente consumida por el motor al variar la magnitud de la carga, las cuales variaron desde los 0.5 Kg hasta 3.5Kg, con un paso de 0.5. Esta prueba se realizó para 4, 5 y 6V. Se graficó cada una de estas pruebas para obtener la función que modela al motor en cada caso. Luego, se extrapoló utilizando esa función hasta un valor de 30kg (valor aproximado del peso del robot explorador). Con esto, se obtuvo una relación de voltaje contra potencia incluyendo el peso estimado. Se encontró una nueva función matemática con la cual se calculó la potencia cuando los motores se alimenten de 12V y tienen 30Kg de carga. Con este valor, se seleccionó el módulo de potencia eléctrica a implementar con el controlador para el movimiento de los motores.

Para la medición de consumo de potencia de la electrónica, se utilizó un cargador externo de 5V y 2.5A. Se hizo uso de un amperímetro para medir en una de las líneas del cargador la corriente demandada. Se realizaron diferentes medidas según el trabajo que realizó la raspberry pi: solamente la raspberry pi en *stand by*, con el arduino conectado por medio del puerto serial, y ejecutando el servidor web con el arduino conectado. Con esto se obtuvo información de máximos consumos y picos de potencia.

El subsistema de control y comunicación se deriva en diferentes bloques. Estos incluyeron: la programación de la aplicación cliente-servidor, la conexión del robot explorador a internet por medio de la

red GSM y la programación del controlador; encargado del movimiento de los motores y el manejo de la información de sensores. En el siguiente diagrama se presenta una vista general del subsistema de control:

Figura 313: Subsistema de control



Primero, se detallará los pasos a seguir para la instalación del servidor web incrustado en la raspberry pi. Después, la programación de la aplicación web en HTML que utilizará el cliente y la ejecución de *scripts* para el envío de comandos por medio del puerto serial al controlador. Luego, la programación del controlador encargado de la obtención de datos de los sensores y del control de los motores del robot explorador.

Se instaló el sistema operativo *Raspbian Jessie* en la raspberry pi. Este se tomó de la página oficial de raspberry pi y bastó con crear una memoria SD ejecutable usando la imagen de disco (extensión de archivo iso) descargada.

Para la aplicación web, se hizo uso del servidor *Apache* en la Raspberry Pi. Al ser instalado, este funciona de manera automática a través del puerto 80 de conexión a internet de la raspberry. Siendo una red local, se ingresó a la dirección IP asignada a la raspberry y con esto se comprobó el funcionamiento correcto del servidor. Con su instalación, se creó un archivo index.html en la dirección /var/www/html/ de la raspberry, en esta ruta se tiene albergado el sitio de la aplicación web y es este es el archivo que el servidor entrega al cliente. Se dividió el programa en tres grandes bloques. Primero, se tuvo el diseño de la interfaz gráfica y los botones. Se crearon imágenes que funcionaron como método de interacción con el usuario. Cada imagen, debidamente identificada según su función, es un botón.

Como segunda parte, se tuvo lo relacionado con el control de movimiento y obtención de valores de los sensores. El archivo HTML mencionado, funciona como programación de *frontend*; es decir, como una interfaz gráfica para el usuario que controla el robot. Esto impide que pueda tener una conexión directa con los puertos físicos o el puerto UART de la raspberry pi, necesaria para el envío de datos al controlador esclavo. El fin fue mandar caracteres a través del puerto UART de la raspberry al arduino, para que este, al leer el carácter proveniente, tomara una acción.

Para esto, se hizo uso de AJAX (Asynchronous JavaScript And XML) en combinación con jQuery, la cual es una biblioteca de JavaScript que permite tener una interacción más sencilla en HTML y con archivos externos. AJAX permite el desarrollo de aplicaciones web interactivas, ejecutándose en el navegador del cliente mientras mantiene una comunicación con el servidor web, con lo que permite tener cambios específicos en la aplicación sin tener que recargar la página completa. Por medio de AJAX, se realizó un jQuery que ejecuta un script desarrollado en PHP (Hypertext Preprocessor) encargado de controlar el puerto serial de la raspberry pi. Para este último paso, se utilizó una clase llamada PhpSerial, encontrada en el repositorio de GitHub. Esta clase crea la conexión entre el archivo PHP y el puerto UART de la raspberry pi y permite diferentes métodos, como escribir o leer en el puerto serial. En la raspberry pi, el grupo llamado *dialout* es el encargado de administrar los dispositivos periféricos, incluidos USB y puertos seriales, por lo que se tuvo que entregar permisos a los scripts de PHP para poder interactuar con estos. Se incluyó al usuario www-data, que administra lo proveniente del servidor apache en el grupo de *dialout* para obtener dichos permisos. El archivo de la raspberry pi, contenedor de la conexión serial al controlador, se denomina *ttyACM0*, por lo cual fue el utilizado en las configuraciones antes mencionadas. [20]

Cada imagen de la interfaz gráfica tuvo asignada una función en JavaScript diferente, encargada de la llamada a través de AJAX para la ejecución del script de PHP, por lo que se creó un script PHP por cada acción que se deseaba implementar. Adelante se detallará la programación del controlador esclavo.

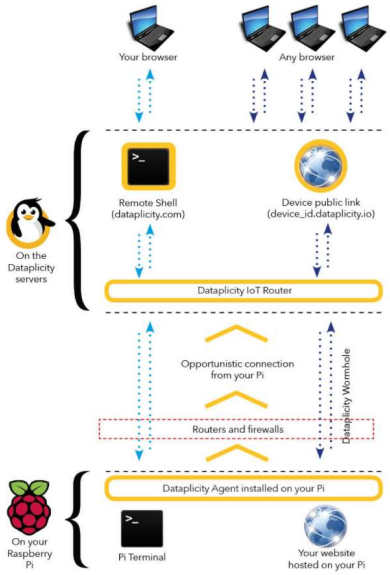
Por último, tenemos la obtención del *stream* de video en la aplicación web. Se hizo uso de funciones JavaScript para el funcionamiento de este. La idea general fue mantener un proceso abierto en la raspberry pi que estuviera capturando fotos cada cierto tiempo, y que el HTML recargara de manera automática dicha imagen. Así, con lapsos de tiempo pequeños entre la captura de imágenes y la recarga automática, se tuvo un stream de imágenes que resulta en un video. Para lograr que el HTML recarga las imágenes, se implementó una función de JavaScript que toma la imagen actual, la despliega y luego le añade un valor mediante la función de fecha actual de HTML. Esto hace que el HTML detecte que hay una nueva imagen que mostrar, siendo esta la nueva imagen tomada por el proceso de captura antes mencionado. Para este proceso se hizo uso del comando *raspistill*, encargado de captura de imágenes usando el módulo de cámara de raspberry pi. La instalación de la cámara y el módulo implementado se mencionarán más adelante.

Para la conexión remota a la raspberry pi se hizo uso del proveedor de servicios *dataplicity*. Este nos provee de métodos para poder acceder remotamente a dispositivos utilizando Linux (en este caso, la raspberry con *raspian*). La principal ventaja es la facilidad que ofrece para poder realizar dicha conexión. En términos

generales, entablar rutas de comunicación entre dispositivos finales que se ubican en diferentes redes tiene muchas complicaciones. Muchas veces, es necesario involucrar al proveedor de servicios de internet para tener direcciones IP estáticas a las cuales acceder, lo que supone pagos extras; además, en este proyecto, la raspberry estuvo conectada al internet por medio de una red GSM/GPRS, por lo que no solo la dirección de esta está en constante cambio, también es complejo conocer cuando cambia y cuál es la nueva dirección IP. [18]

Dataplicity funciona como un servidor proxy, que crea y mantiene una conexión HTTPS (Hyper Text Transfer Protocol Secure) entre la raspberry pi y un router (Dataplicity IoT Router):

Figura 314: Conexión realizada para comunicación con Raspberry Pi



[18]

Para la instalación del agente de *dataplicity* en la raspberry pi, se creó una cuenta en su página web (18). Los datos de inicio de sesión son:

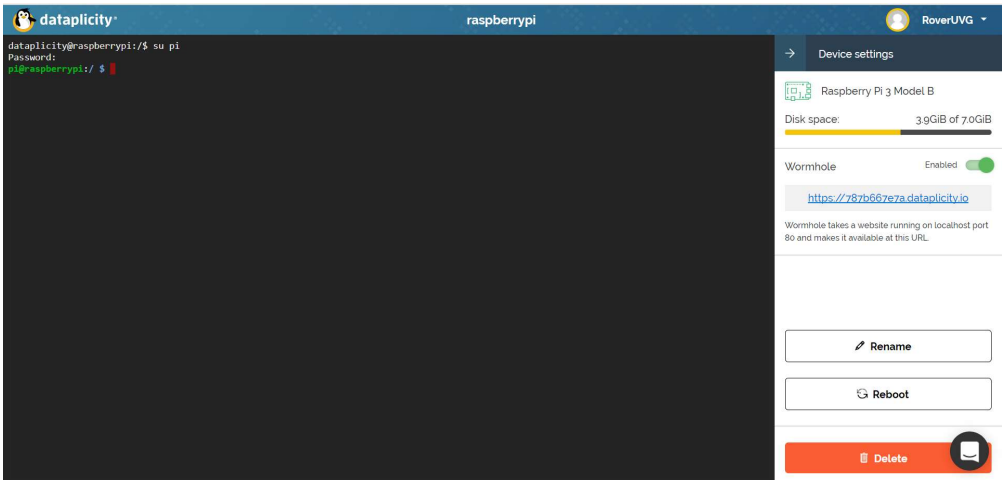
- Email Address: roveruvg@gmail.com
- Contraseña: robotexplorador

Después de haber iniciado sesión, obtenemos una línea de código para la instalación del agente en la raspberry pi, la cual se debe ejecutar desde la terminal de comandos de esta:

Figura 315: Código para la instalación de agente en Raspberry Pi

Luego de instalar el agente, la raspberry pi figuró en la lista dispositivos instalados en la cuenta en la página web. Se ingresó al dispositivo y para obtener acceso a la raspberry pi se ejecutó el comando “su pi” y luego se ingresa la contraseña: “raspberry”.

Figura 316: Sesión iniciada en la Raspberry Pi desde Dataplicity

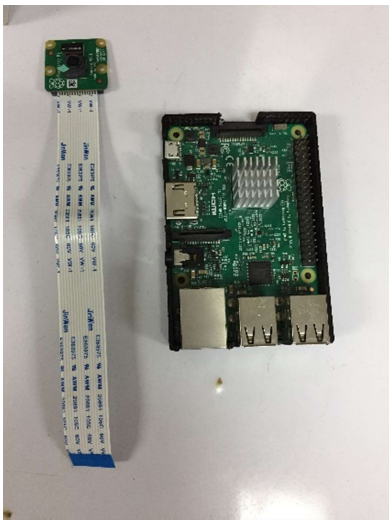


De esta manera se tuvo instalado correctamente el agente de *dataplicity* y el acceso a la terminal de comandos de la raspberry pi desde cualquier red.

Luego, se procedió a la instalación del servidor web. *Dataplicity* nos permite hacer uso de un “*wormhole”* enlazado con el dispositivo, el cual es un URL específico. Este permite que la raspberry funcione como un host con conexión en el puerto 80. Desde esta dirección web se pudo acceder al servidor y a archivos específicos (HTML) que contienen la aplicación web que ejecutamos en el cliente. Para el desarrollo de la aplicación web se utilizó la aplicación MJPG-Streamer, servidor web diseñado específicamente para el *streaming* de video. Esta aplicación se encarga de recopilar la información obtenida de la cámara y enviarla a través de una sesión HTTP.

Primero, se hizo la instalación de la cámara. Se optó por el uso del módulo de cámara V2 para raspberry pi:

Figura 317: Módulo de cámara V2 para Rasbperry Pi



Esta se conecta por medio de un *flat flex* al puerto designado para la cámara en la Raspberry pi. La etiqueta azul debe de estar frente al puerto Ethernet:

Figura 318: Conexión del módulo de cámara a Raspberry pi

Luego de realizar la conexión física, se habilitó el uso de la cámara en la raspberry pi. Para probar el funcionamiento de la cámara, se tomó una imagen utilizando el comando raspistill. Después, se configuró la cámara como un dispositivo USB de video, pues la aplicación *mjpg-streamer* lo requiere. Se cargó el módulo "Video for Linux 2" (V4L2) utilizando el hardware BCM2835, el cual es el encargado de los módulos periféricos en el modelo raspberry pi utilizada. Se debe de mencionar que esta configuración persiste en la raspberry pi hasta que esta se apaga; es decir, cada vez que se reinicie el dispositivo, se debe de volver a realizar la configuración como USB. Para evitar esto, se añadió la línea de código a un script de inicio del sistema operativo; en este caso al archivo *rc-local***.** Este archivo se encuentra en la dirección /etc/. En el

editor se añadió debajo de los comentarios la línea de código de configuración del BCM. Con esto se garantizó que la configuración se realice automáticamente al encender la raspberry pi.

Luego, se verificó que se haya instalado el dispositivo de video USB. Para esto se buscó el archivo video0 en la dirección /dev. La existencia de este archivo nos permitió saber que se realizó la instalación de manera correcta. Se instaló otras dependencias que requiere *mjpg-streamer* para su funcionamiento. Estas son: *libjpeg8-dev*, *imageMagick* y *libv4l-dev*. El primer paquete es una librería para poder manejar archivos con extensión JPEG, *ImageMagick* es un software que incluye herramientas para la manipulación, conversión y despliegue de imágenes en distintos formatos y por último el paquete *libv4l* incluye librerías que añaden una capa de abstracción a dispositivos que utilizan *Video for Linux 2*, con el fin de darle compatibilidad a una gran variedad de dispositivos sin tener que programar códigos diferentes para cada uno.

Teniendo ya todos los prerrequisitos configurados e instalados, se prosiguió a la instalación de la aplicación *mjpg-streamer*. Esta aplicación no se puede instalar desde un paquete ejecutable existente, por lo que primero se descargó el archivo comprimido y luego se compiló. En la sección de anexos se incluye las líneas de código ejecutadas, en donde se puede encontrar la página web de descarga de dicho archivo. Luego, se creó un enlace simbólico entre librerías utilizadas para compilar la aplicación. Un enlace simbólico indica que un archivo tiene referencia a otro archivo o directorio como acceso. Después, se editó el *Makefile* de la aplicación para deshabilitar el *plugin* input*_gspcav1*, ubicado en la carpeta creada al compilar. Por último se creó un archivo que fuese ejecutable para poder activar el servidor desde la línea de comandos. De esta forma se instaló la aplicación que permitió tener el servidor en la raspberry pi y el *stream* de video a través del módulo de cámara. El paso faltante consistió en iniciar el servidor haciendo uso del archivo ejecutable mencionado. Esto se hizo con la siguiente instrucción:

sudo ./mjpg_streamer -i "./input_uvc.so -f 10 -r 640x320 -n -y" -o "./output_http.so –w ./www -p 80" .

En la siguiente tabla se presenta la explicación de la línea de comando para la inicialización del servidor:

Tabla LXIV: Parámetros para inicio del servidor

Parámetro	**Descripción de uso**
-i es una opción para indicar los parámetros del *plugin* de entrada	
-f	*Framerate* - se usa para indicar los frames por segundo del video.
-r	*Resolution* – indica el tamaño de la imagen que vamos a capturar y mostrar
-n	Elimina errores que surgen debido a que no se tiene un control físico sobre la cámara. Estos no tienen ninguna incidencia en el stream de video en vivo
-y	Formato YUV en el video de salida
-o Indica los parámetros para el *plugin* de salida	
-w	Indica la ruta de acceso de donde se obtiene el contenido a servir en el sitio web
-p	Indica en que puerto se sirve la aplicación web

Con el servidor inicializado y después de habilitar el *wormhole*, se pudo acceder a la aplicación web desde *dataplicity*, pulsando en dicho enlace encontrado en la interfaz de *dataplicity*.

Cuando el cliente accede al *wormhole*, ejecuta el archivo "index.html", ubicado en la ruta: /home/pi/mjpg-streamer/www/, en el cual se tiene un menú para poder indagar en las acciones que permite *mjpg-streamer*, tales como: captura de imagen o stream de video, entre otras. Para el proyecto, nos interesó tener el stream de video, que encontramos en el archivo denominado "stream_simple.html". De este archivo se tomó el código para el *streaming*.

Para la conexión con la red GSM se utilizó un modem USB 3G, marca Huawei de la compañía Claro Guatemala. Este funcionó como un teléfono móvil prepago, por lo que fue necesario acreditarle saldo para su conexión a internet. El número telefónico es 5561-9748. Estos tienen también compartimiento para SD y así funcionar como una memoria de almacenamiento. Fue necesario configurarlo como modem 3G en la raspberry pi, utilizando la aplicación *PPP* y *wdial*.

La programación del controlador arduino se dividió en dos bloques generales: control de motores y manejo de sensores. Para el funcionamiento de este, se hizo uso de la comunicación serial por medio del puerto UART con la raspberry pi. El controlador espera a recibir un valor en dicho puerto y, dependiendo del valor recibido, realiza diferentes acciones.

Las librerías utilizadas fueron: librerías del proveedor de los módulos (adafruit) para la IMU y el FONA (encargado de tomar datos GPS), librería *Software Serial* de arduino para la comunicación serial con el FONA, y la librería PWM para el uso de control de motores. Luego, se declararon las siguientes variables para definir los pines de conexión con los módulos:

Tabla LXV: Asignación de pines

Nombre de la Variable	Número de Pin	Descripción	I/O
velM1	11	PWM para control de velocidad del motor 1	Output
velM2	6	PWM para control de velocidad del motor 2	Output
dirM1	22	PIN digital que define dirección del motor 1	Output
dirM2	23	PIN digital que	Output
Trig1	26	Pin digital para el trigger del sensor ultrasónico 1	Output
Trig2	28	Pin digital para el trigger del sensor ultrasónico 2	Output
Eco1	27	Pin receptor echo del sensor ultrasónico 1	Input
Eco2	29	Pin receptor echo del sensor ultrasónico 2	Input
TX	10	Pin para el TX del FONA	Input
RX	2	Pin para el RX del FONA	Output
RST	4	Pin para el RST del FONA	Output
SDA	21	SDA comunicación I2C de la IMU	I/O
SCL	20	SCL comunicación I2C de la IMU	Output

Luego, se instanció la comunicación serial por medio del *Software Serial* para la comunicación entre el FONA y el arduino. Se crearon los objetos FONA y BNO055 que implemetaron las funciones de la librería proporcionadas por Adafruit. Posteriormente, Se inicializó el puerto serial para la comunicación con la

raspberry a una tase de transferencia de 9600 baudios y se configuraron los pines a utilizar para los sensores ultrasónicos y el control del motor como entradas o salidas según correspondía.

Se hizo uso de la estructura de control de tipo *switch-case*. Luego de verificar que se recibieron datos en el puerto serial, se lee dicho caracter y se utiliza como parámetro para el *switch*, y así ejecutar el *case* correspondiente según el valor recibido. Por ejemplo, si en el puerto serial se recibió el caracter "a", entonces ejecuta el código correspondiente al *case* 'a', que en este caso fue avanzar hacia adelante.

La siguiente tabla muestra los cases definidos con su caracter correspondiente y la acción a desarrollar:

Tabla LXVI: Cases y acciones

Caracter recibido	Módulo correspondiente	Acción
a	Control de motores	Avanzar hacia adelante
r	Control de motores	Retroceder
d	Control de motores	Giro a la derecha
I	Control de motores	Giro a la izquierda
s	Control de motores	Detener ambos motores
b	FONA	Obtener porcentaje de carga de la batería del FONA
n	FONA	Obtener estado de red celular (si existe una conexión a la red)
i	FONA	Se obtiene el estado RSSI que indica intensidad de señal celular
l	FONA	Obtener coordenadas GPS (latitud y longitud)
u	Ultrasónico frontal	Obtener mediciones del sensor ultrasónico frontal
U	Ultrasónico trasero	Obtener mediciones del sensor ultrasónico trasero
T	IMU	Temperatura interna del robot

Para tener mejor estructurado el código, se desarrollaron las siguientes funciones:

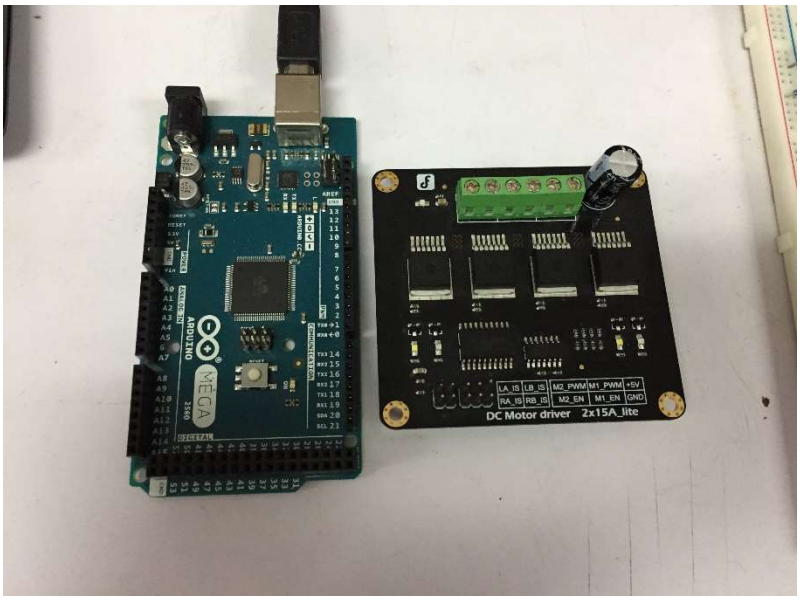
Tabla LXVII: Funciones programadas

Nombre de la función	Tipo de Retorno	Parámetros	Descripción y uso
serialFona	void	Ninguno	Inicializar la comunicación serial entre el arduino y el FONA
networkStat	void	Ninguno	Realizar conexión a red
movimiento	void	char v1, char v2, char d1, char d2	Encargada del control de los motores según parámetros recibidos
Readline	Unsigned int de 8 bits	Buff, maxbuff, timeout	Función usada en librería FONA,

Como se mencionó en la sección de potencia, se implementó un driver capaz de suministrar la potencia necesaria para ambos motores. Este utiliza un control por medio de *pulse width modulation* (PWM) usando lógica combinacional y puentes en H, con los que se pudo variar tanto la velocidad, como la dirección

de giro del motor. Se requirió únicamente de un bit que define el sentido de giro y una salida de PWM, que maneja la velocidad, por cada motor. Este driver hace uso de cuatro integrados BTS7960, que se conforman de medio puente H usando de transistores MOSFET de alta potencia.

Figura 319: Driver para motor DC



Para todos los cases de control de motor, se utilizó una misma función: movimiento. Los parámetros que recibe son:

- V1, que corresponde al ancho de pulso para definir la velocidad del motor 1 (un valor entre 0 y 255).
- V2, que corresponde al ancho de pulso para definir la velocidad del motor 2 (0-255)
- D1, que indica la dirección de giro del motor 1 (valor lógico 1 o 0)
- D2, que indica la dirección de giro del motor 2 (valor lógico 1 o 0)

En esta función únicamente se cambia el estado de los pines de salida para controlar las direcciones, y se generan los PWM para la velocidad de cada motor. De esta manera, solo fue necesario llamar a la función movimiento () y enviar los parámetros de interés para el control del motor. Por ejemplo, si ejecutamos el case “a”, que indica avance hacia adelante, llamamos a la función movimiento (), con valores de v1 y v2 en 255 para la velocidad máxima, y con valores de d1 y d2 en un 1 lógico para indicar un mismo sentido de giro. Para el caso “s”, que indica detener los motores, se llama a la función movimiento con todos los parámetros en 0.

Luego tenemos la obtención de datos de los sensores y el manejo de esta información. Se explicará la implementación de cada uno por separado.

Se utilizó el sensor ultrasónico HC-SR04, para la medición de distancias a posibles obstáculos. Este es un sensor de bajo costo, de fácil acceso y que cumple con la tarea de dichas mediciones.

Figura 320: HC-SR04

Consiste en 4 pines: tierra, voltaje, *trigger* y *echo*. El pin trigger funcionó como el disparador encargado de iniciar el envío de la onda ultrasónica. *Echo* es la salida del sensor, que funciona entregando un flanco cuando se capta la onda ultrasónica de rebote. Para la obtención de la distancia se colocó un uno lógico en el disparador, se dejó en ese estado durante 10 microsegundos y luego se escribió un cero lógico. Seguido, se usó la función de arduino *pulseIn*. Esta detecta un pulso en un pin definido. Se utilizó el valor HIGH (1 lógico), por lo que la función escribió un 1 lógico en el pin y esperó a que se produjera un pulso 0, generado cuando la onda rebota. Con esto se obtuvo el tiempo de viaje de la onda ultrasónica y haciendo uso de la ecuación No. 1, presentada en la sección Marco Teórico, se calculó la distancia, en centímetros. Se implementaron dos sensores en el robot explorador.

Seguido tenemos la obtención del posicionamiento GPS. Como ya se mencionó con anterioridad, se hizo uso del FONA808:

Figura 321: FONA808

[19]

La descripción de los pines es la siguiente:

- Vio: voltaje de entrada para su funcionamiento (5V proveniente del Arduino).
- GND: referencia, conectada a tierra del arduino.
- Key: Este pin se utiliza para encender o apagar el módulo. Haciendo una conexión a un 1 lógico durante dos segundos se enciende o lo apaga. Se conectó a GND para no apagar el módulo durante la ejecución del código
- 5V: Este pin indica si se conecta un cable microUSB para la alimentación. En nuestro caso se alimentó el módulo con una batería externa de litio. Con este pin a 5V del arduino se recargar dicha batería.
- Reset (RST): Este pin realiza un *hard reset* al módulo en caso haya algún problema mayor. Se conectó al pin 4 del arduino para usos de la librería.
- RX y TX: El módulo se comunica por medio de un puerto serial. RX es para la entrada del FONA y se conectó al pin 2 del arduino y el TX es para la salida del FONA y se conectó al pin 10.

El módulo incluye diferentes indicadores LED:

- PWR: de color azul, indica que el módulo arrancó y que está funcionando.
- NET: de color rojo, indica diferentes estados de la red: 64ms encendido y 800ms apagado indica que el módulo está corriendo pero que no ha realizado la conexión a la red. 64ms encendido y 3 segundos apagado, indica que el módulo se conectó con la red celular. 64ms encendido y 300ms apagado, indica que hay una conexión GPRS activa.
- Charging: de color naranja, indica que la batería está cargando.
- Done: de color verde, indica carga completa de la batería.

Este potente módulo ofrece una gran variedad de opciones y librerías para su uso. Se utilizó la librería proporcionada por *adafruit*, sitio en donde se adquirió el módulo. Esta contiene las funciones necesarias para hacer uso del mismo. En la comunicación serial, el FONA toma automáticamente el valor del *baudrate*, por lo que no requiere de una configuración extra en el módulo más que la realizada en el código al inicializar la conexión. Siempre que se ejecuta alguna función de la librería utilizada, este no solo imprime en el puerto serial los valores finales deseados, también muestra información de la conexión e información de estado del módulo. Esto es un problema debido a que, para la obtención de datos por parte del servidor web y el archivo PHP, nos interesó tener únicamente la información final y desechar lo restante. Además, cada vez que se inicializa el puerto serial del arduino, también se reinicia la conexión serial con el FONA, lo que causa problemas con la comunicación con el servidor, debido a que, cuando este hacían los *request* por medio de PHP al puerto serial, fue necesario abrir y cerrar la comunicación serial entre dicho archivo y el arduino.

Para evitar que la conexión serial del FONA se ejecutara cada vez que se establece la comunicación con el archivo PHP, se decidió crear dicha conexión únicamente cuando se tienen los cases

correspondientes del FONA. Es por esto, que no se tiene conexión al módulo en la función setup () ni en los cases que no corresponden a este.

Por último, se diseñó y se fabricaron cuatro placas de circuitos impresos (PCB, por sus siglas en inglés) para realizar la interconexión entre el controlador y los diferentes módulos: la primera para interconectar el arduino con el driver del motor, el FONA y la IMU; la segunda, para situar la IMU y las otras dos para colocar los ultrasónicos y realizar las conexiones con el controlador.

B. RESULTADOS

a. Mediciones de consumo de potencia de motores

Tabla LXVIII: Potencia consumida sin carga al variar voltaje

Voltaje (V)	Corriente (A)	Potencia (W)
0	0	0
1	1.276	1.276
1.5	1.38	2.07
2	1.46	2.92
2.5	1.82	4.55
3	1.9	5.7
3.5	2.18	7.63
4	2.26	9.04
4.5	2.47	11.115
5	2.774	13.87
5.5	2.832	15.576
6	2.84	17.04
6.5	3.056	19.864
7	3.16	22.12
7.5	3.226	24.195
8	3.294	26.352
9	3.556	32.004
10	3.64	36.4
11	4	44
11.5	4.128	47.472

Figura 322: Potencia vs. voltaje

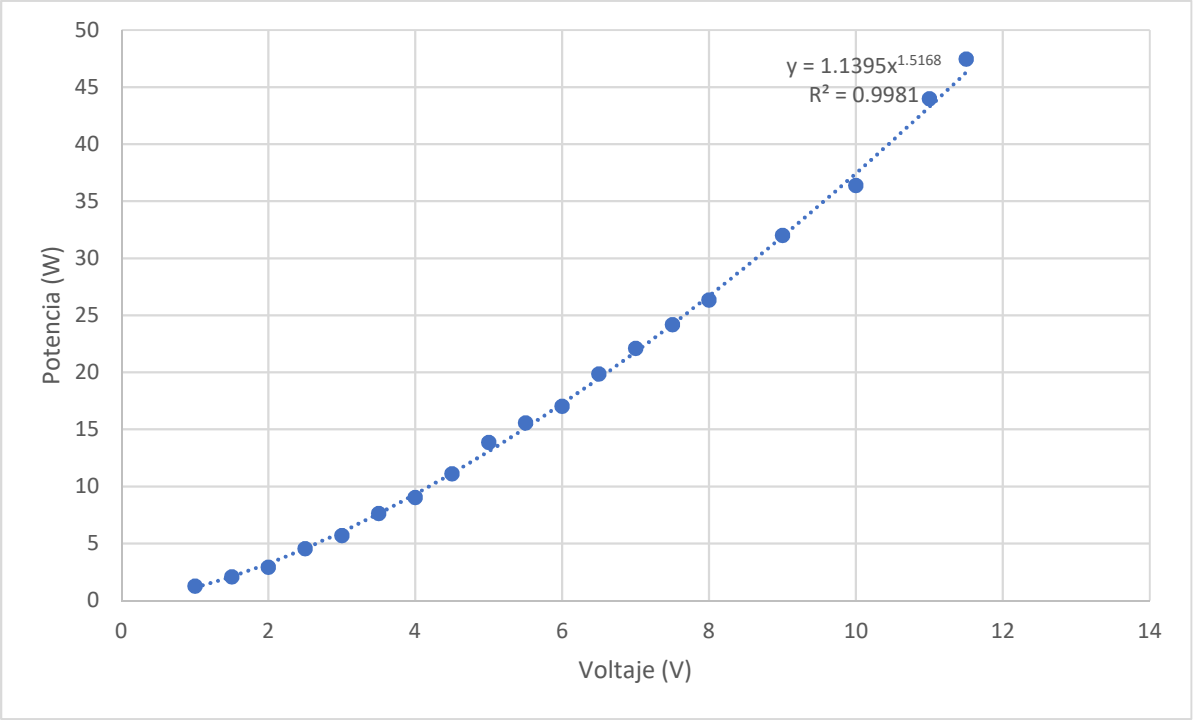


Tabla LXIX: Potencia al variar la masa alimentando el motor con 4V

Masa (kg)	Potencia (W)
0.25	9.28
0.5	9.288
1	9.24
1.5	9.4
2	9.488
2.5	9.504
3	9.624
3.5	9.6

Figura 323: Potencia al variar la masa alimentando el motor con 4V

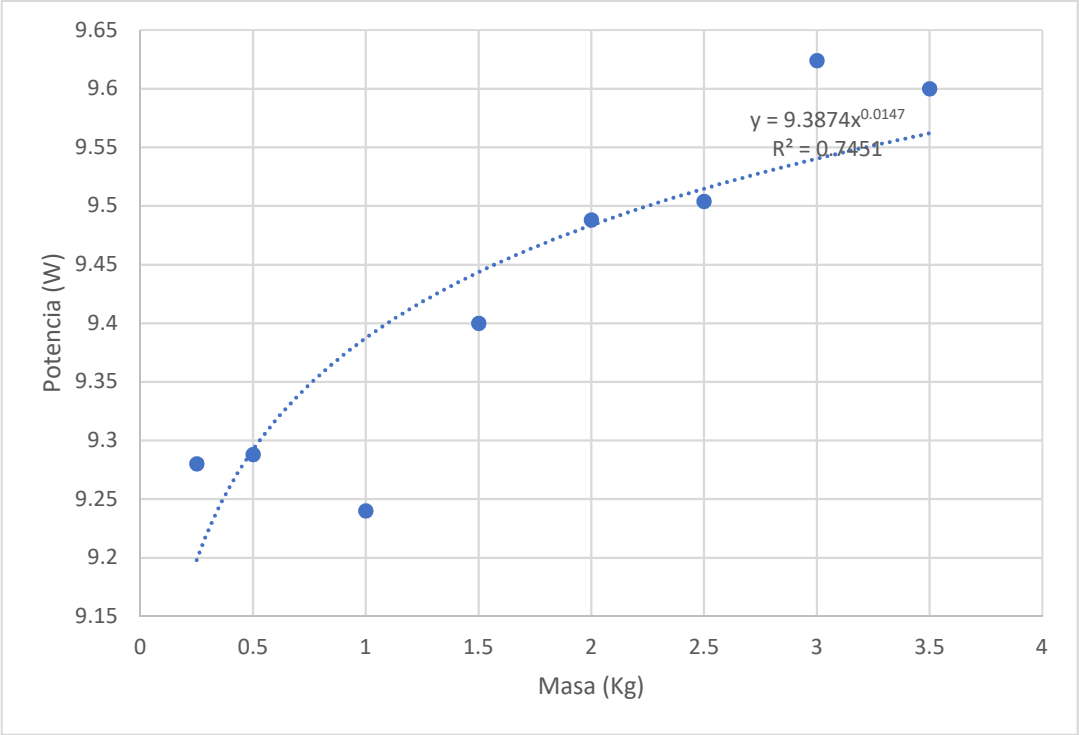


Tabla LXX: Potencia al variar la masa alimentando el motor con 5V

Masa (Kg)	Potencia (W)
0.25	13.2
0.5	13.3
1	13.33
1.5	13.4
2	13.5
2.5	13.5
3	13.5
3.5	13.6

Figura 324: Potencia al variar la masa alimentando el motor con 5V

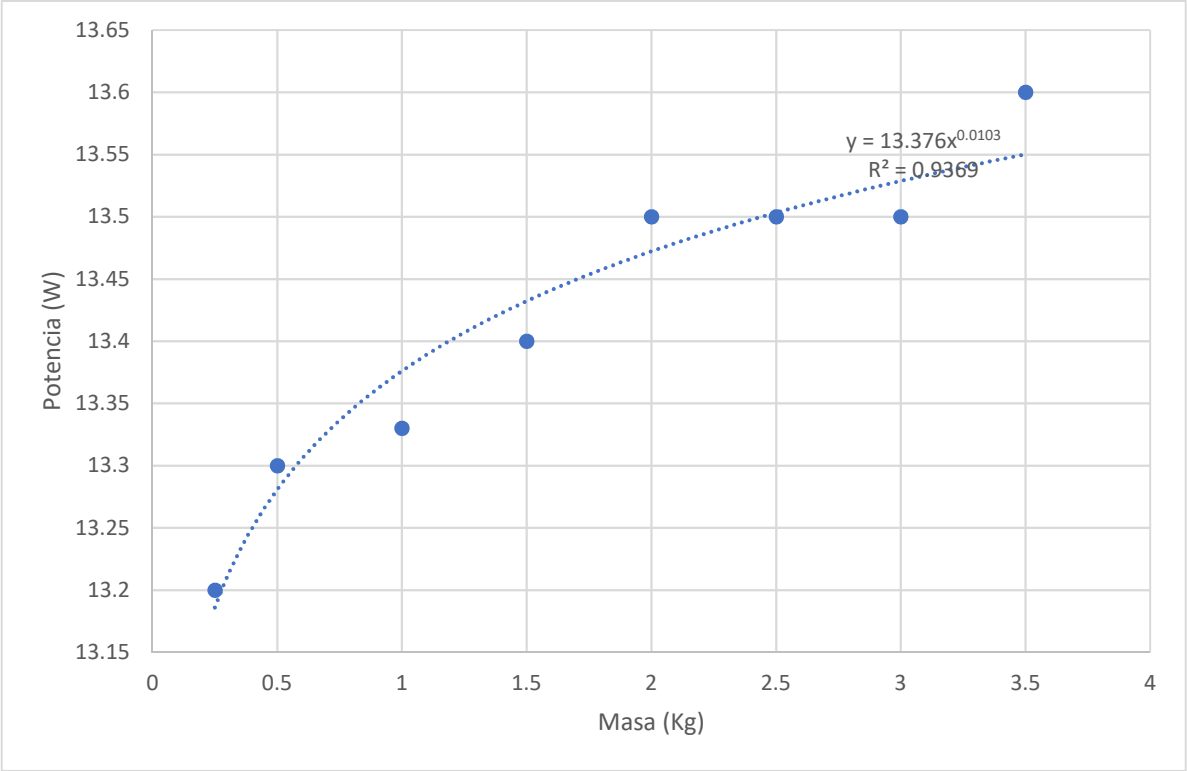
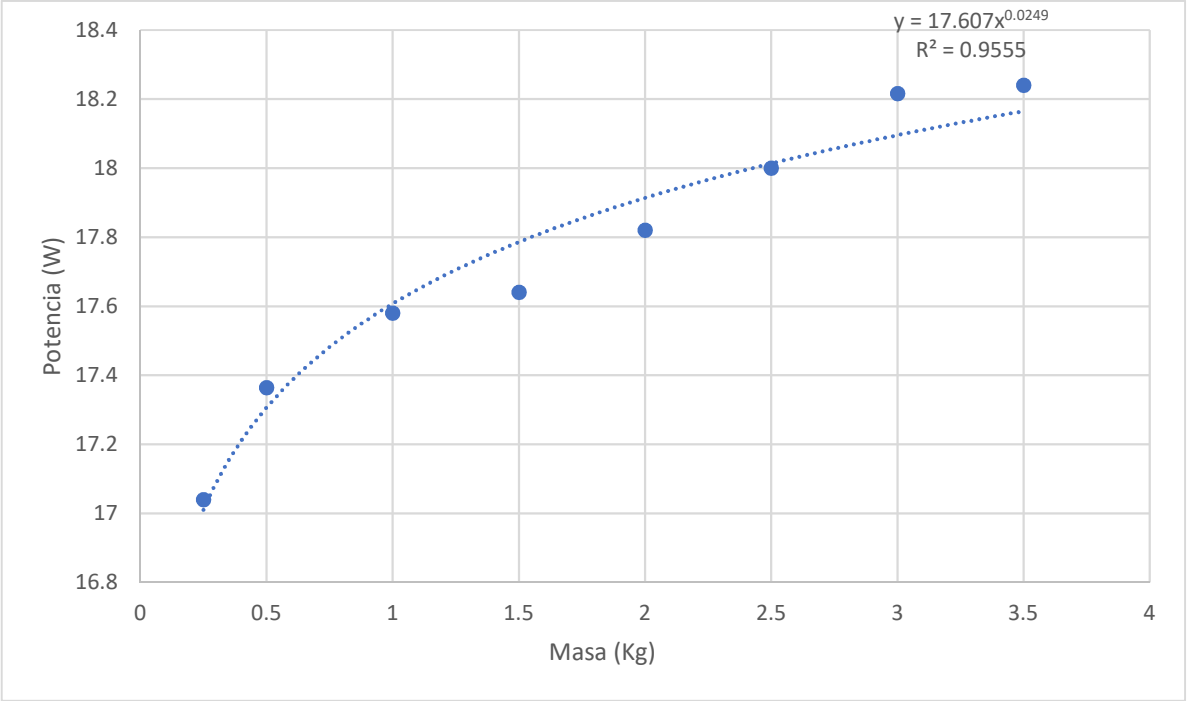


Tabla LXXI: Potencia al variar el peso alimentando el motor con 6V

Masa (Kg)	Potencia (W)
0.25	17.04
0.5	17.364
1	17.58
1.5	17.64
2	17.82
2.5	18
3	18.216
3.5	18.24

Figura 325: Potencia al variar la masa alimentando el motor con 6V

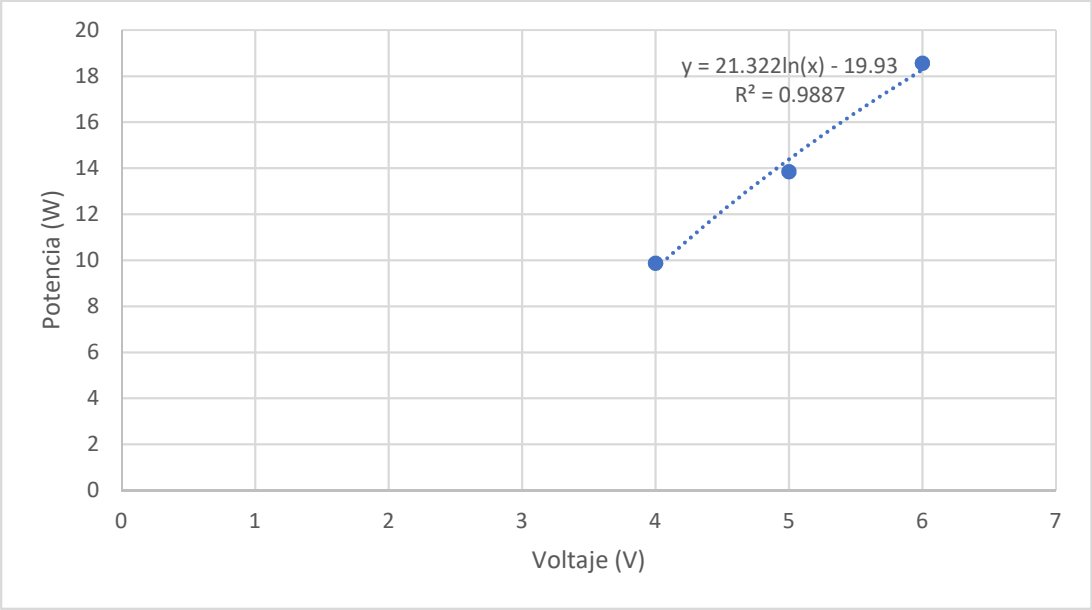


Se calculó la potencia estimada que el motor consumiría usando estos voltajes con un peso de 30 Kg para poder tener una relación de la potencia con el voltaje con el peso del robot explorador:

Tabla LXXII: Valores de potencia con carga de 30 Kg a diferentes voltajes

Función de Voltaje	Potencia con 30Kg (W)
4V	9.86867
5V	13.85289
6V	18.567752

Figura 326: Relación voltaje-potencia con 30Kg de carga



Utilizando la función obtenida en esta relación, estimamos el valor de la potencia consumida con carga de 30Kg con 12V de alimentación:

$$Potencia = 21.322 \ln(x) - 19.93$$

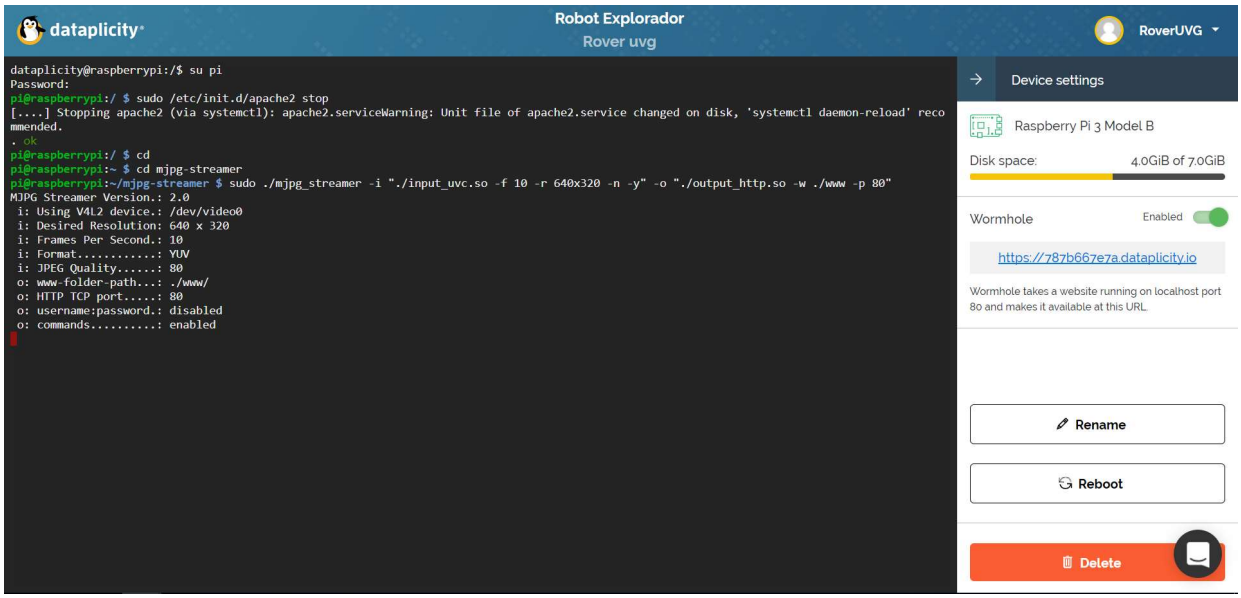
Evaluando la función en x = 30, se obtiene una potencia de 52.59W.

Tabla LXXIII: Consumo de corriente de los módulos electrónicos

	Arranque (A)	Tiempo estacionario (A)
Raspberry	0.4	0.22
Raspberry y Arduino	0.65	0.31
Servidor y Arduino	1.2	0.43

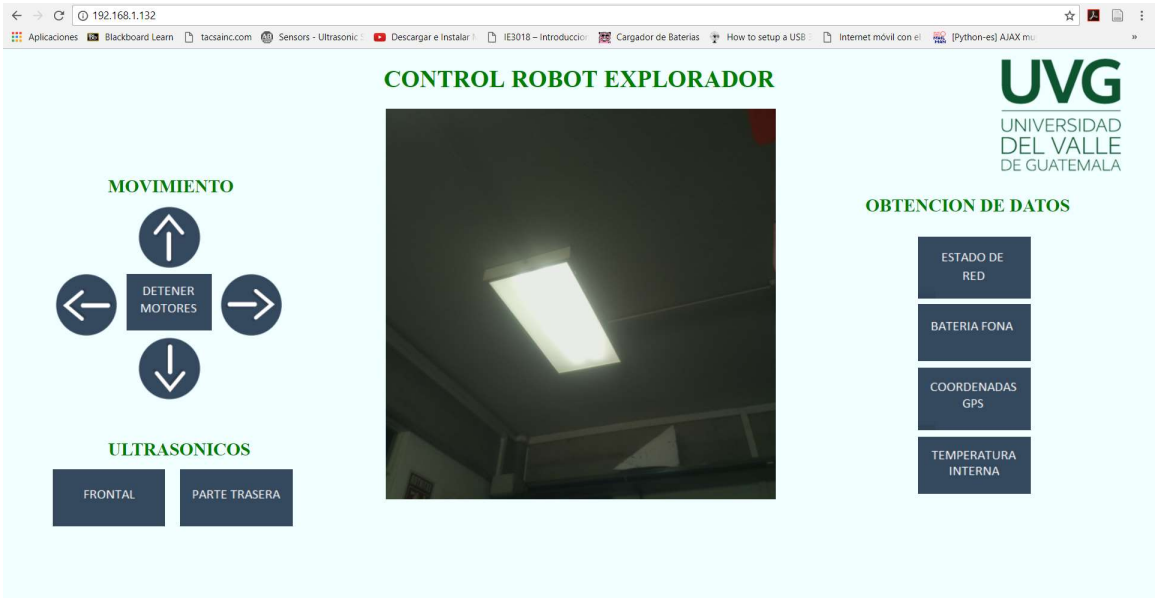
b. Activación del servidor e información:

Figura 327: Activación del servidor Mjpg-Streamer



c. Controles en interfaz gráfica con servidor Apache:

Figura 328: Controles e interfaz de usuario



d. Control de motores. Se presentan diferentes casos para el ciclo de trabajo del PWM y como este varió según el valor utilizado como parámetro en el código del controlador.

Figura 329: PWM con valor de 128, ciclo de trabajo del 50%

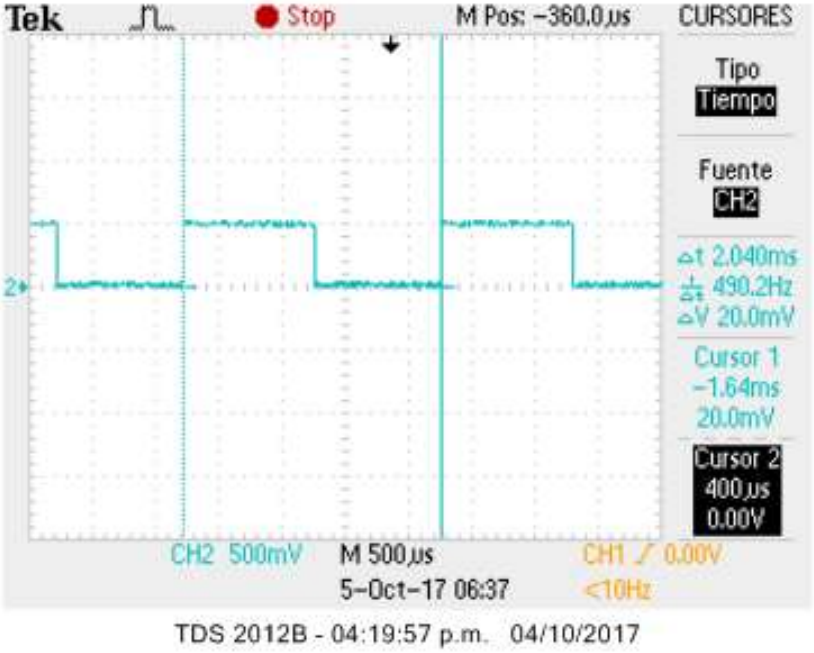
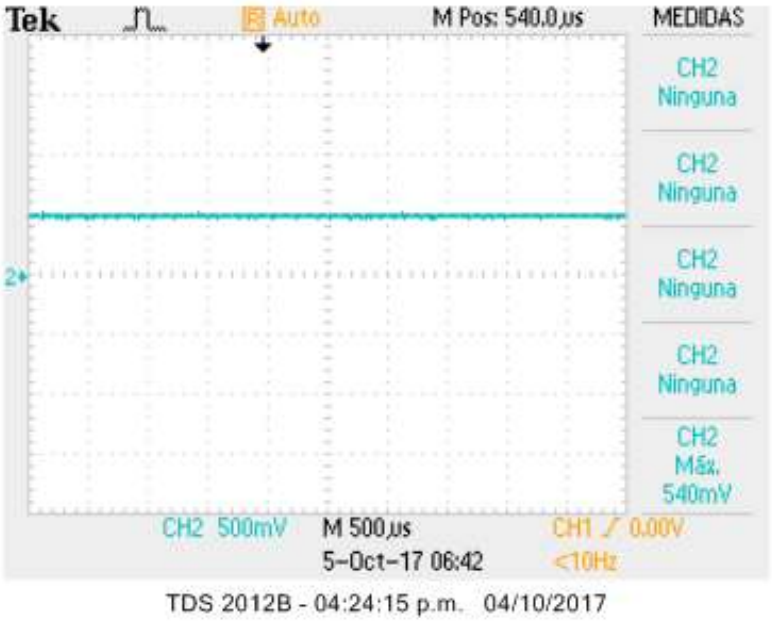


Figura 330: PWM con valor de 255, ciclo de trabajo de 100%



El PWM generado por el arduino tiene un periodo de 2.040 ms. Variamos su ciclo de trabajo mediante la relación:

$$Ciclo\ de\ Trabajo\ (\%) = \frac{Valor\ PWM * 100}{255}$$

$$Tiempo\ en\ Alto\ (ms) = \frac{Valor\ PWM * 2.040}{255}$$

En Donde:

- Valor PWM es un valor entre 0 y 255 en el código de arduino.

e. Obtención de valores de sensores por medio del serial de una computadora:

Cases para el FONA:

Figura 331: Verificación de conexión con FONA

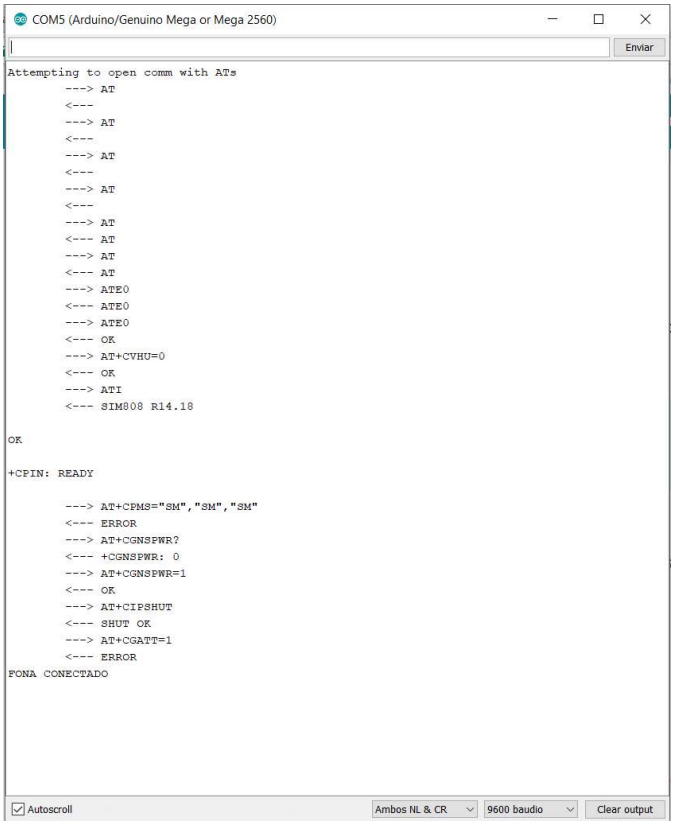


Figura 332: Verificación de estado de red (registrado)

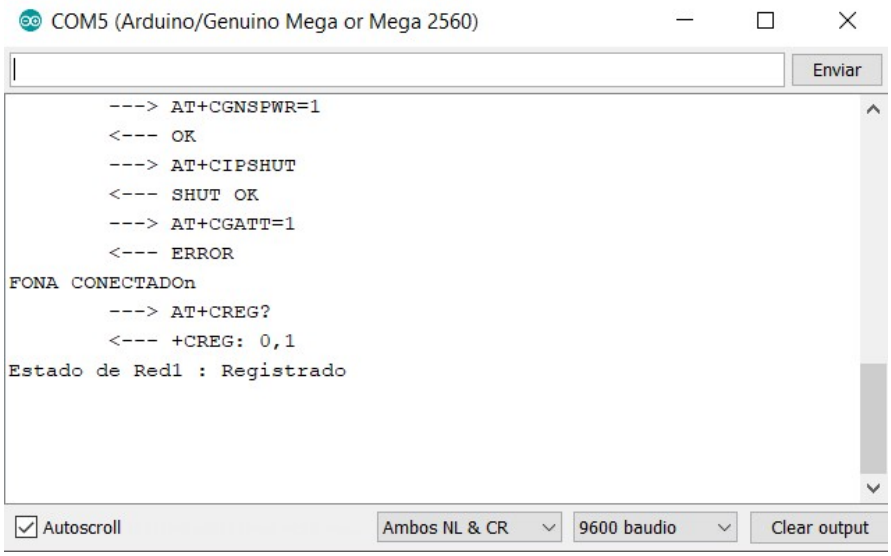
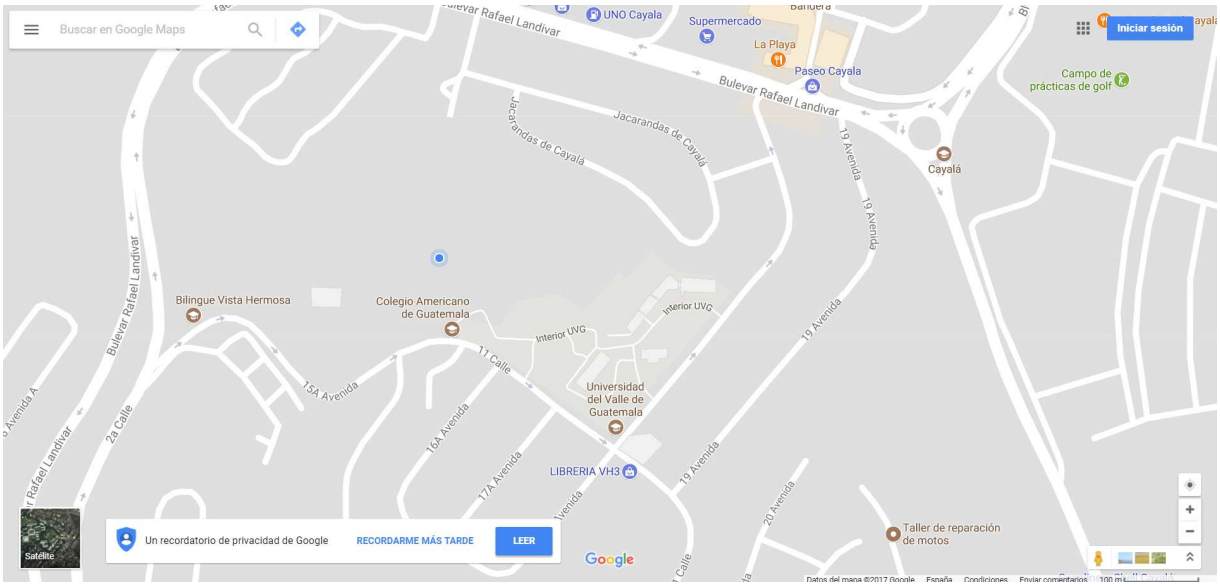


Figura 333: Coordenadas GPS según mapas de google



Latitud de 14.6053523, longitud de -90.4893221.

Figura 334: Coordenadas GPS según FONA808

Latitud de 14.608231, longitud de -90.487244.

Figura 335: Obtención de mediciones sensores ultrasónicos

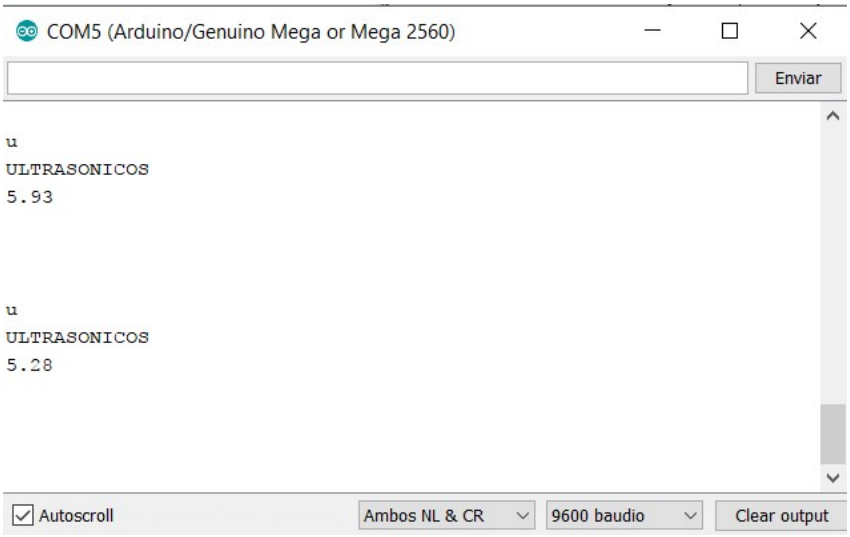


Figura 336: Diseño de placa para interconexión entre Arduino y el módulo de potencia, FONA y la unidad de medición inercial

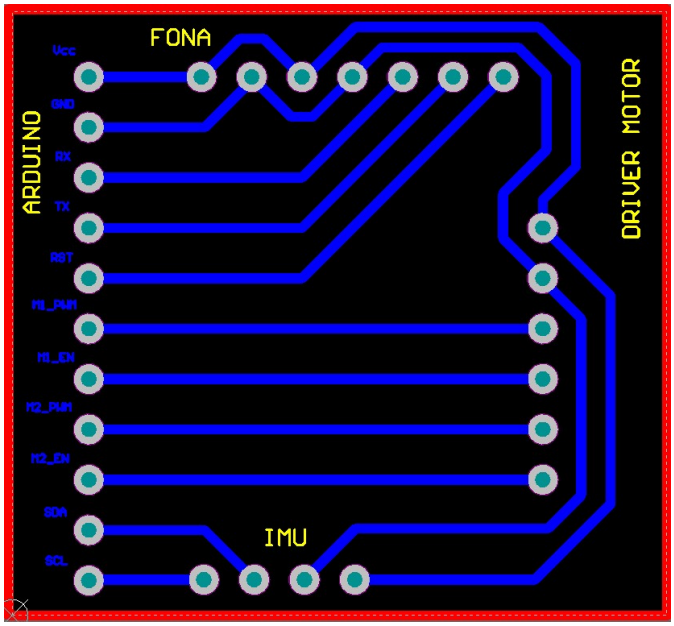
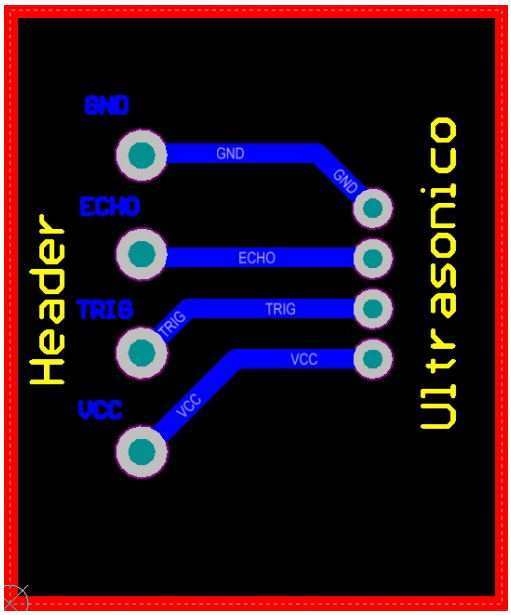


Figura 337: Diseño de placa para colocar sensores ultrasónicos HC-SR04



C. ANÁLISIS DE RESULTADOS

La estimación de potencia para los motores con alimentación de 12V y con una carga de 30kg es de 52.49W, haciendo uso de una regresión logarítmica, esto considerando el comportamiento y la naturaleza de los motores. Como se puede observar en la sección de resultados, los valores de R^2 obtenidos con estas regresiones son bastante aceptables, teniendo un valor mínimo en 0.7451 para la regresión donde el voltaje

fijo fue de 4V. Para las otras regresiones, los valores oscilaron entre el 0.95 y el 0.998, lo que indica que las funciónes se aproximan de muy buena manera al comportamiento de los datos muestreados. El segundo motivo del uso de este tipo de regresión es que se espera que la curva de consumo de potencia de motor al variar la carga tenga una asíntota horizontal en algún punto. Es decir, el consumo de potencia crecerá en cuanto crezca la carga o el voltaje de alimentación, pero llegará a un valor en donde las mismas limitantes físicas y eléctricas del motor no le permitirán seguir aumentando. Por esto no se seleccionó una regresión lineal, exponencial o potencial, pues estas, aun cuando representaban bien los datos, tienden a infinito conforme la carga crece, lo que es físicamente imposible.

El principal problema es que no se tuvo información alguna de las características de los motores. Esto provoca muchos otros conflictos, pues hay que invertir tiempo en este tipo de pruebas que ciertas veces no resultan ser las óptimas. Probablemente se pueden obtener mejores muestreos utilizando mayor cantidad de pesos aplicados al motor, o bien, valores mayores de peso; aunque esto implica una plataforma más grande y elaborada y la obtención de dichos pesos. Estos motores habían sido utilizados en el megaproyecto “Robot Autárquico” y de igual forma no se tenía un valor teórico del consumo de potencia que este tiene según la carga aplicada. Aunque podemos tomar el valor de la prueba para el dimensionamiento, fue necesario aplicar un factor de seguridad para la selección del *driver* para el controlador. Partiendo de los valores obtenidos, se utilizó una batería recargable de motocicleta, de 12V-9Ah, *W Standard*. Esto teniendo en cuenta que cada motor consumirá aproximadamente 4.38A. Es preferible el uso de una sola batería que alimente a ambos motores y no tener un arreglo de estas en paralelo, por diversas razones. Principalmente, debido a cuestiones físicas y químicas, las baterías no son exactamente iguales la una con la otra, por lo que no se puede garantizar que estas se descarguen a la misma tasa. Si una de estas se descarga más rápido que la otra, terminarán gastando de manera incorrecta la carga, afectando también su tiempo de vida útil. Además de utilizar una única batería para los motores, también se utilizó una batería independiente para la electrónica. Ambas referencias deben de ir interconectadas entre sí. Partiendo de todos estos factores, se decidió implementar un driver para motor DC capaz de manejar hasta 15A por cada canal. Esto nos brinda un factor de seguridad de 3.4 sobre el valor teórico que tenemos de la corriente que consumirán, para evitar problemas de sobrecarga en el driver. Para la alimentación de la electrónica, se utilizó una batería convencional BS de motocicleta de 12V-5Ah modelo BB5L-B.

En el subsistema de control y manejo de sensores, se optó por el uso de la raspberry pi debido a su capacidad de procesamiento, tomando en cuenta que se debía de utilizar un dispositivo capaz de mantener activo un servidor para una aplicación web, que sirviera de video a una interfaz de un usuario en sincronía con la comunicación al controlador esclavo.

En una red local, se pudo unificar en una misma aplicación web el stream de video y el control del robot explorador. Se logró obtener la comunicación serial por medio de la clase *PhpSerial*. Esta librería está en desarrollo, por lo que presenta varias debilidades. Se tuvo que modificar esta clase para la conexión de dispositivos de tipo *tty* (conexiones periféricas de las raspberry). La escritura de valores no

presentó mayor problema; sin embargo, la lectura sí. Esto sucedió debido a que el comando *readPort* () de la clase *PhpSerial* lee todo el contenido en el puerto serial, por lo que, al ejecutar conexiones con el FONA, se recibía mucha información irrelevante y no solo la última línea, que contenía el mensaje de importancia. Para solucionar esto se decidió terminar la conexión del puerto serial en el arduino antes de realizar la conexión al FONA y comenzarlo hasta haber terminado dicha conexión para luego escribir en el puerto serial únicamente la línea con la información obtenida, además de agregar un tiempo de espera antes de la lectura en el PHP luego de mandar el valor serial para asegurar existencia de información. Dicho tiempo varió según él *case* a ejecutar.

Se detectó que cuando se ejecutaba la función de lectura del puerto en el archivo PHP, el arduino releía el puerto serial y ejecutaba acciones nuevamente al encontrar un carácter vinculado a un case. Para mitigar esto, se implementó un retraso de 4 segundos en el arduino luego de escribir respuestas en el puerto serial. Es decir, después de escribir el mensaje con el valor del sensor, se cerró el puerto serial y se hizo el retraso, que correspondía al tiempo en el que el archivo PHP realizaba la lectura. Luego de dicho tiempo y con la lectura del PHP terminada, se reabrió el puerto para futuros comandos. Esta solución permitió la comunicación serial entre la raspberry pi y el arduino en ambas vías; sin embargo, se queda sujeto a tiempos muertos de espera que hace que la obtención de los valores no se dé de manera instantánea. El tiempo de espera máximo fue para la obtención de coordenadas GPS, esto debido a que le toma tiempo al FONA obtener dichos valores. El tiempo fue de 21 segundos. Para el caso de la obtención de porcentaje de carga de batería del FONA y el estado de la red, los tiempos fueron de 9 y 12 segundos respectivamente.

El stream de video por medio del método de recargar las imágenes resulta en un video con retraso, mínimo pero perceptible; no obstante, tiene la importante ventaja de unificar el stream de video con el control del robot, que entrega una aplicación web completa. Para mantener la toma de fotos en la raspberry se utilizó el comando raspistill con el atributo –tl que define la cantidad de tiempo antes de cada captura. El valor utilizado fue de 100 ms. El control por medio de la red local requiere conocer la dirección IP de la raspberry pi. Esto se puede encontrar desde la misma raspberry con el comando *ifconfig* o se puede utilizar un programa como *Angry Ip Scanner* que permite conocer la dirección de los dispositivos conectados a la red local. También, se puede crear una red local por medio del modem USB 3G implementado o utilizando un router como punto de acceso.

Se logró tener el acceso desde una red remota a la raspberry pi, incluso si esta tiene la conexión a internet por medio del modem celular 3G. En esta modalidad no se pudo implementar el control del puerto serial de la raspberry pi, pero si se logró tener el stream de video, activando el servidor Mjpg-Streamer y accediendo al stream añadiendo "/stream.html" a la dirección del *wormhole* proporcionada por *dataplicity*. Es importante detener el servidor apache2 antes, pues este se ejecuta directamente en el puerto número 80, correspondiente a la dirección del *wormhole* y no permitirá la ejecución del otro servidor. No se logró la implementación del control pues la aplicación Mjpg-Streamer no puede manejar los archivos PHP debido a permisos de acceso en la raspberry py. Además, si se ejecuta por medio del servidor apache, como

en una red local, no se pudo entregar los permisos necesarios al agente *dataplicity* para la ejecución y el manejo de los archivos PHP. Por medio del servidor Mjpg-streamer se obtuvo satisfactoriamente el stream con una buena calidad de imágenes y con retrasos en el video mínimos. Toda la información correspondiente a los códigos de programación, cuentas, números telefónicos, scripts, etc., se encuentran en un repositorio de *GitHub* en el enlace. [21]

Sobre la conexión a la red celular, el modem 3G cuenta con su propio medidor de transferencia de datos. Este se utilizó para estimar el coste de mantener un stream de video con la aplicación web. Utiliza aproximadamente 135 Mb durante una hora de transmisión con 8 frames por segundo en la calidad de la imagen en el Mjpg-Streamer, lo que es relativamente barato.

El uso de tecnologías como Arduino brinda una importante versatilidad para futuros proyectos integrados en el robot explorador. El módulo FONA utilizado para la obtención de valores GPS es una herramienta que brinda una gran cantidad de utilidades. Su principal atractivo es la facilidad de programación partiendo de la librería proporcionada por el distribuidor. Pensando en costos, probablemente es mejor el uso de un módulo GPS dedicado, con ese único fin específico, aun cuando la programación resulte ser más compleja. Arduino no es precisamente una herramienta utilizada a nivel industrial, pero en enfoques mayormente académicos, amigables al usuario y de demostración, le da a los proyectos capacidad de expansión y nuevas implementaciones. Estudiantes pueden aplicar conocimientos directamente en el robot explorador, programando códigos nuevos para la realización de diferentes tareas. El proyecto se enfocó en permitir que el robot explorador funcionara como una versión de demostración, fácil de utilizar y de controlar. Futuras generaciones tienen la oportunidad de desarrollar y aplicar nuevos proyectos en él y así lograr el cometido que era la recolección de basura en las playas de Guatemala.

XI. CONCLUSIONES

A. PRÓTESIS BIÓNICA

1. Se determinó que el método más apropiado de control para cumplir con los objetivos propuestos es una combinación de movimiento biomecánico con sistemas electrónicos ya que este provee la capacidad de controlar la mano de manera intuitiva.
2. Se seleccionó el microcontrolador Pro Trinket de Adafruit por que este posee la cantidad de entradas y salidas digitales, así como la cantidad de entradas analógicas requeridos por el método de control.
3. El implementó un método de control capaz de entregar hasta 6 posiciones distintas en la mano, más no permite el control de la velocidad de movimiento durante la transición entre ellas.
4. Se realizó un nuevo diseño del antebrazo que provee a los circuitos internos con mayor protección contra impactos y mejora la apariencia física, pero limita el espacio disponible para la instalación de nuevos circuitos.
5. Se implementaron agujeros en la parte inferior del antebrazo que le permite ser compatibles con el codo trabajado en módulos anteriores, en caso se desee retomar su uso para la prótesis en el futuro.
6. Se logró establecer la comunicación entre el controlador y la prótesis con el chip nRF24L01+ para el envío de mensajes codificados que indiquen la señal obtenida por el método de control diseñado.
7. Se descartó la programación de lo microcontroladores por medio del manejo directo de los registros ya que este presentó problemas de compatibilidad con la librería del módulo de radiofrecuencia.
8. Se realizaron placas PCB que integran tiras de pines los cuales permiten conectar y desconectar algunos componentes ya que esto posibilita la reparación pronta del controlador, en caso algún elemento falle, sin la necesidad de volver a fabricar la placa.
9. Se encontró que el mejor método para modelar el acrílico, es a través del uso de una pistola de calor. Debido a que ésta puede proveer la temperatura en la cual este material entra a la zona plástca y permite adoptar la forma deseada.
10. Se implementó un hilo trenzado para evitar el estiramiento de los hilos que controlan el movimiento de los dedos, gracias al pre estiramiento que se realiza durante su fabricación.
11. Se encontró que la utilización de pines sólidos como guías para la unión de las falanges conlleva a evitar que alguna parte de este componente se quede dentro de las falanges.
12. Se seleccionó una fuente de voltaje de 5 amperios que puede proveer la corriente necesaria para alimentar los servomotores y demás componentes electrónicos.
13. Se diseñaron dos prototipos de un case tomando en cuenta un dimensionamiento que permita alojar sin ningún problema a los componentes utilizados en el PCB y permitiéndole un fácil acceso.
14. Se logró transmitir la presión hacia las yemas utilizando silicón de uso general.

B. SWARM ROBOTICS

1. Se logró diseñar e implementar cada uno de los módulos de manera correcta, logrando así tener un robot eficiente y funcional capaz de implementar sistemas básicos de control.
2. Se seleccionó el Teensy 3.2 como el microcontrolador más apropiado para la implementación del firmware debido a su velocidad de funcionamiento y su capacidad de procesamiento de datos.
3. Se identificaron múltiples problemas relacionados con el envío de datos y el manejo de instrucciones debido al uso del protocolo TCP que fueron resueltos, pero deben tomarse en cuenta a futuro.
4. Se fabricó un PCB que incluye un decodificador de 3 a 8 líneas que permite controlar el Trigger de 6 sensores ultrasónicos HC-SR04 utilizando únicamente 3 pines de salida digitales de un microcontrolador; y utiliza lógica combinacional para que el Echo de los sensores llegue a un solo pin de entrada digital.
5. Se programó un Teensy 3.2, que posee un microcontrolador basado en ARM Cortex, y se agregó prioridad en el NVIC a las interrupciones externas del puerto E, donde se conecta el Echo de los sensores ultrasónicos, para realizar mediciones de ancho de pulso que representan la distancia hacia un objeto.
6. Se midieron distancias de objetos radialmente simétricos con la falda de sensores ultrasónicos. En distancias de 4 cm y 10 cm, al menos el 76% de las mediciones se ubican entre en un rango de ±1cm alrededor del valor esperado, mientras que, en distancias de 24 cm y 34 cm, al menos el 72% de las mediciones se ubica en un rango de ±2cm alrededor del valor esperado.
7. Se implementó una trayectoria punto a punto con un controlador proporcional para la velocidad angular, con una ganancia de 10, usando la información de los encoders, y el modelo uniciclo como descripción matemática del robot.
8. Se comprobó que el error como diferencia entre ángulo yaw deseado y ángulo yaw actual no es útil, porque el algoritmo de control no entiende la naturaleza cíclica de los ángulos, y el comportamiento del robot se vuelve errático. Para ello es necesario trasladar el error al intervalo $[-\pi, \pi]$, y con ello se logró que el robot alcance la meta dentro un radio de 10 cm.
9. No es necesario un circuito balanceador de celdas en este sistema porque el desbalanceo experimentado no es significativo ya que es solo de un 5.25% de el voltaje nominal de cada celda.
10. No es recomendado utilizar un regulador LDO en este tipo de aplicación ya que las eficiencias son muy bajas.
11. Se logró diseñar e implementar una estructura funcional para los seis ultrasónicos HC-SR04.
12. Se utilizó placas de MDF para construir una base que albergará el módulo de motores DC miniQ, encoders y baterías tipo LiPo.
13. Se diseñó y construyó una PCB que genera las potencias necesarias para el funcionamiento óptimo del robot destinado a trabajar en enjambre.
14. Es recomendable diseñar este tipo de robots de manera modulable ya que esto facilita más su reparación, armado y desarmado.
15. Se utilizó de manera exitosa el lenguaje Arduino para implementar los motores miniQ con sus encoders.
16. Mediante piezas 3D, piezas de MDF y acrílico, tornillos infinitos M3 y tuercas M3 se logró diseñar y ensamblar una estructura modulable fácil de armar y de desarmar.

17. Se experimentó que los motores DC miniQ con el driver doble puente H DRV8833 tienen una respuesta no lineal ante la modulación del ciclo de trabajo de su señal PWM de control.
18. Al momento de imprimir en 3D se experimentó que el llenado que nos dio una pieza para los ultrasónicos más fuerte es el de perfil rectangular 25 %.

C. SILLA

1. Utilizando un enclave mecánico, se solucionó el problema del giro sobre el propio eje de la silla.
2. Se logró comunicar por medio de bluetooth los controladores y con esto lograr realizar las subrutinas para los giros de motores, tanto en el control manual como en el control inalámbrico.
3. Se implemento un cargador interno específico para baterías de ácido-plomo, que es el tipo de batería con el que cuenta la silla.
4. Se evitaron daños a las baterías a la hora de cargarlas, esto debido a que se cargan según los parámetros de la misma.
5. El tiempo de carga total de una batería es de aproximadamente 5 a 6 horas.

D. ROBOT EXPLORADOR

6. Se implementó un servidor incrustado en una Raspberry Pi, haciendo uso de la aplicación MJPG-Streamer y el servidor proxy dataplicity para obtener una aplicación cliente-servidor que permite el stream de video.
7. Se implementó un servidor incrustado Apache2, en una red local, que permitió tener en una aplicación web el control de movimiento del robot, la obtención de valores de los sensores y stream de video.
8. Se logró tener conexión remota al robot explorador a través de internet y una conexión a la red celular por medio de un modem 3G.
9. Se utilizó una batería recargable de motocicleta de 12V y 9Ah para la alimentación de los motores y una batería de 12V 5Ah para la alimentación de la electrónica.
10. Con el uso del driver de motor DC, se tuvo un control eficiente de la orientación de giro y la velocidad del eje del motor.
11. Se obtuvo las coordenadas de posicionamiento global del robot explorador, distancias hacia obstáculos cercanos y la temperatura interna.

XI. RECOMENDACIONES

A. PRÓTESIS BIÓNICA

1. Se recomienda rediseñar los circuitos del controlador y prótesis haciendo uso de componentes de superficie que permitan hacer placas PCB más compactas.
2. Se recomienda diseñar una carcasa para los controladores que provean protección y se acomoden a la forma anatómica del tobillo para que este se pueda usar con mayor comodidad.
3. Se recomienda rediseñar el antebrazo para que la forma sea aún más parecida a la anatómica real y se investigue el uso de material distintos para su fabricación que otorguen características adicionales deseables.
4. Se recomienda hacer un análisis profundo sobre la interacción entre los dedos, las posiciones y los objetos que se desean sujetar para obtener un modelo más completo sobre las fuerzas aplicadas durante el agarre que permita modificar con precisión la cantidad de fuerza aplicada
5. Se recomienda hacer uso de los sensores resistivos colocados en las yemas de los dedos para diseñar una prótesis de lazo cerrado que mejore el movimiento y agarre de objetos por medio de técnicas de control moderno.
6. Se recomienda investigar la posibilidad de modificar el esquema actual de movimiento para otorgar al usuario control directo sobre la velocidad de cierra y apertura de la mano.
7. Es necesario realizar revisiones a la prótesis, por lo menos cada dos veces al año, principalmente a los hilos. Si bien las propiedades de estos son excelentes para el trabajo que realizan, al ser hilos trenzados poseen como desventaja que su final de segmento pueda deshilarse según el movimiento que se realice. Si se encuentra algún rastro de que esto sucede es necesario cambiar el segmento de hilo por uno que tenga un acabado plano.
8. La fuente de voltaje es de uso único en interiores y es recomendable no exponerla a ambientes húmedos.
9. Si se desea retirar algún pin o separación de falange y este se llegase a perder, se recomienda que antes de insertar el reemplazo deba de adelgazarse un poco el nuevo pin antes de introducirlo. Esto es posible a través de una lija o dremel con la herramienta apropiada con el fin de facilitar su inserción y deslizamiento a través de las falanges.
10. Bajo condiciones de operación normal es necesario mantener la compuerta de los servos cerrada para evitar la introducción de partículas de polvo o de cualquier otra sustancia no deseada.
11. Es necesario rediseñar la mano para poder tener un funcionamiento en conjunto con la muñeca, e inclusive para definir mejores trayectorias de los hilos sí se planean seguir utilizando.
12. Es necesario diseñar alojamientos específicos para los sensores, esto implicaría rediseñar las falanges distales. De esta manera se evitaría dejar sus conexiones en la parte externa.
13. Cambiar el tipo de banda dentada utilizada en el mecanismo de la muñeca por una más comercial.

14. Emplear otro mecanismo para transmitir las presiones a los sensores en las yemas que no sea utilizar silicón ya que los resultados en este proceso poseen demasiadas variaciones como lo son la forma obtenida de la yema, tiempo de secado, rugosidad del molde de arcilla, etc.
15. Incluir un mecanismo retráctil para los servomotores del antebrazo para devolverlos a la posición de tensado cuando la muñeca no esté en uso.

B. SWARM ROBOTICS

1. Se recomienda que el punto de partida y la meta de la trayectoria punto a punto no sean exactamente iguales en la coordenada x. Esto se debe a que al usar una arcotangente en el algoritmo de control, se genera una división entre cero si las coordenadas en x son las mismas. Para probar solo desplazamiento en el eje y, se colocó una diferencia de 0.1 entre las coordenadas x.
2. En la fabricación de un PCB se recomienda dejar un ancho de pista mayor al calculado, para garantizar que los requerimientos de incremento de temperatura, corriente y caída de voltaje se cumplirán. Para la fresadora de la UVG no se recomiendan anchos de pista menores a 10 mil. La precisión de la máquina permite fabricar placas con esas características, pero la soldadura se hace de forma artesanal y se pueden levantar tracks, o se puede perder continuidad si una pista se rompe con un movimiento brusco de las manos o alguna otra herramienta.
3. Se recomienda experimentar con varias faldas de sensores para probar cómo interfieren entre ellas cuando muestrean una a la vez. Una sola falda funciona bien usando puerto serial USB para recopilar datos. Sin embargo, al usar el módulo WiFi, aparece con más frecuencia el timeout.
4. Se recomienda probar una IMU de 9 DOF para complementar la lectura de los encoders y reducir el error de las mediciones de posición que crea el arrastre de las llantas. No se logró medir el ángulo yaw con la IMU de 6DOF, por eso se recomienda una que tenga acelerómetro, giroscopio y magnetómetro para obtener este ángulo, que es importante por la dirección del robot.
5. Se recomienda probar las capacidades del microcontrolador del módulo NodeMCU, que posee un ESP8266 integrado, para manejar el robot. Se usó un módulo ESP8266 con un Teensy por la amplia documentación que existe para este microcontrolador, pero se recomienda probar el nodeMCU para probar la transmisión de datos, y ver su desempeño al controlar el robot.
6. Se recomienda calibrar los encoders antes de realizar pruebas con el robot. Estos son cruciales, porque a partir de su información se desarrolla todo el algoritmo de control. En futuras versiones, se puede probar con cámaras motion capture, y descartar los encoders.
7. Se recomienda probar controladores para seguimiento de trayectorias. Esto se recomienda para hacer demostraciones en público, por ser vistosas.

8. Se recomienda probar otros tipos de sensores de proximidad en la falda de sensores, para discernir si un sensor más robusto tiene un mejor balance entre desempeño y precio que el módulo utilizado en el presente megaproyecto.
9. Luego de analizar el desempeño de los reguladores tipo LDO se llegó a la conclusión que no fueron la mejor selección. Por ello se sugiere utilizar un regulador de tipo step down switching regulator (buck). Se sugieren este tipo de reguladores ya que estos tienen una eficiencia típicamente arriba del 70 % por ciento, no disipan la potencia perdida como calor. Se recomienda específicamente utilizar los modelos TPS54308 y LM2596 ya que estos modelos cumplen con los requerimientos del proyecto. Ambos poseen una muy alta eficiencia como podemos ver en la Figura 339 y en la Figura 340. La desventaja de utilizar esta solución es que ambos reguladores mencionados anteriormente y casi siempre estos tipos de reguladores requieren un circuito exterior para su funcionamiento, esto aumentara la complejidad. Así también se recomienda seguir las especificaciones de diseño en PCB que recomienda el fabricante en la datasheet.

Figura 338. Eficiencia de TPS54308.

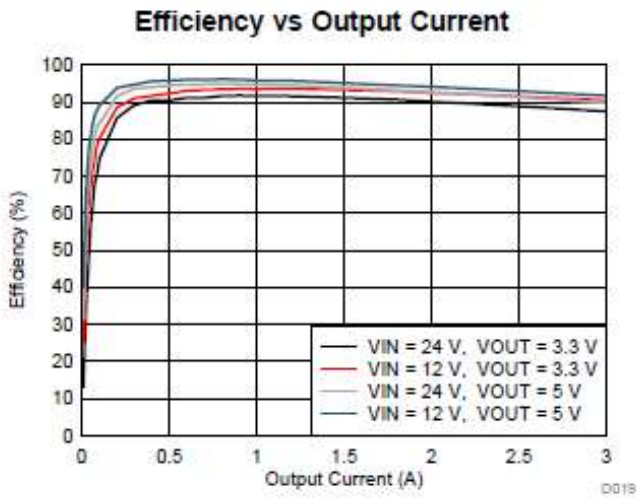
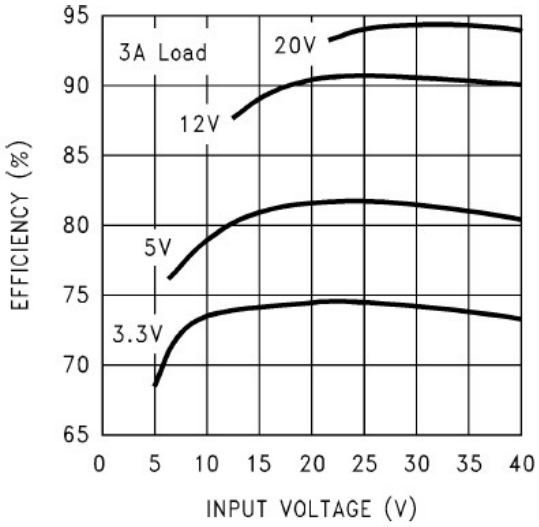


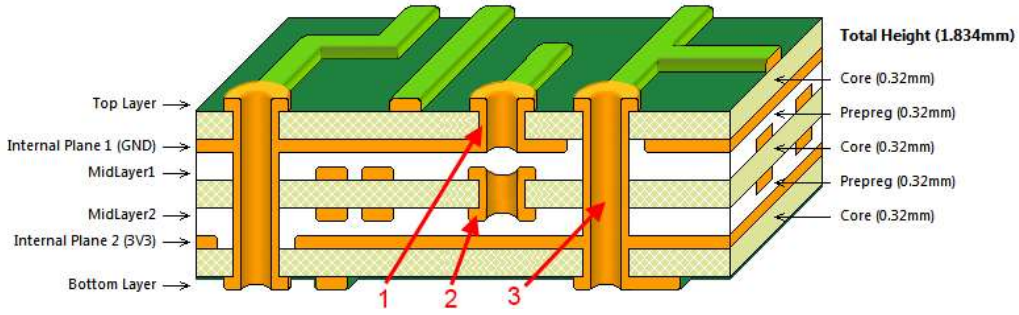
Figura 339. Eficiencia para LM2596.



Otra opción que se recomienda es utilizar la herramienta de diseño de buck converter de Texas instruments llamada WEBENCH. Esta herramienta en línea nos proveerá de una interfaz donde deberemos de colocar los parámetros que necesitamos para nuestra aplicación luego ella generará 3 tipos de soluciones para nuestro caso. La ventaja de utilizarlo es que tendremos una solución específica para nuestro sistema y probablemente sea la más optima, la desventaja es que este generara en la mayoría de los casos circuitos más complicados y con componentes solo disponibles en Texas instruments. [131]

10. En este proyecto se diseñó una PCB de dos capas, se recomienda al ser una placa que varios componentes de switcheo rápido utilizar un diseño de varias capas. De esta manera se pueden colocar capaz de solo voltaje y otras de solo tierra. Esto mejorara el desempeño del circuito ya que de ser bien colocados los planos de poder y tierra puede haber un blindaje de ondas electromagnéticas entre líneas o de onda provenientes del exterior. Otro beneficio que se obtiene con esto podrá colocarse más componentes en un espacio más pequeño. Podemos ver un ejemplo de este tipo de diseño en la Figura 341. [6]

Figura 340. Ejemplo de una PCB con multi capas.



[92]

11. Implementar una manera de desacoplar eléctricamente los motores con el resto del robot cuando este está apagado, de manera que cuando se apague el robot las corrientes inducidas por los motores no retornen al circuito y generen voltaje.

C. SILLA

1. Implementar un sistema de control en lazo cerrado, teniendo en cuenta el sensor y actuador adecuado para unir el proceso de movimiento lineal de la silla y el movimiento rotacional de los triángulos para subir gradas.

D. ROBOT EXPLORADOR

1. Diseñar un filtro para eliminar las señales de ruido inducidas por los motores al circuito.
2. Debido a la importancia, se recomienda utilizar motores para los cuales se posea una ficha técnica o una hoja de datos. A partir de estos se puede hacer un diseño más robusto.
3. Para el cálculo de la potencia eléctrica consumida, utilizar el mecanismo llamado “freno de Prony”, con el cual se puede medir la potencia de salida del motor haciendo uso de un abrazo con fuerzas aplicadas. Teniendo este valor del trabajo del eje de salida, utilizar la eficiencia del motor eléctrico para poder tener un segundo valor teórico para la potencia de consumo del mismo.
4. Utilizar una cámara IP o USB, por el largo de la línea física que la conecta con la Raspberry, para poder situarla cómodamente en el robot explorador sin que esta línea afecte.
5. Utilizar la herramienta HardwareSerial para independizar la comunicación serial del FONA y así lograr reducir los tiempos de espera en la obtención de datos.
6. Implementar lazos de control cerrado para el movimiento del robot explorador, utilizando coordenadas GPS o información de trayectorias generadas por la IMU utilizando Python en la raspberry pi.

XIII. BIBLIOGRAFÍA

[1] A. Bottomley, A. Wilson y A. Nightingale. 1963. «*Muscle substitutes and myo-electric control*» IEEE Radio and Electronic Engineer. XXVI (6): 439-448.

[2] A., A., C., P.-E., D., F., & K., A. (N.A. de N.A. de N.A.). *Wifi*. Obtenido de New Tech for Noobs: https://newtech4noobs.wordpress.com/choose-the-language/the-english-site/wifi/

[3] Adafruit, *Adafruit Pro Trinket*. https://www.adafruit.com/product/2000. [septiembre 2017].

[4] Akshay, Jha 2014. *Wireless Remote Using 2.4 Ghz NRF24L01*. http://www.instructables.com/id/Wireless-Remote-Using-24-Ghz-NRF24L01-Simple-Tutor/ [septiembre 2017].

[5] Al-Aubidy, K. (2015). *Embedded Systems Design*. Retrieved septiembre 2017, from Philadelphia University: https://www.philadelphia.edu.jo/academics/kaubaidy/uploads/ES-Slids-lec3.pdf

[6] Amitron Corp. (10 de 4 de 2017). Obtenido de http://www.amitroncorp.com/printed-circuit-boards/multilayer.html

[7] Arduino. *Arduino NANO*. https://store.arduino.cc/arduino-nano. [septiembre 2017].

[8] Arduino. *Arduino UNO Rev3*. [En línea]. Recuperado de: https://store.arduino.cc/arduino-uno-rev3. [septiembre 2017].

[9] ARM developer. (n.d.). *Architecture*. Retrieved septiembre 2017, from https://developer.arm.com/products/architecture

[10] ARM developer. (n.d.). *Cortex M0+*. Retrieved septiembre 2017, from https://developer.arm.com/products/processors/cortex-m/cortex-m0-plus

[11] Atmel. 2016. *Atmega328/P Datasheet Complete*. http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf. [octubre 2017].

[12] Barrios, Dysmart. *Características biomecánicas de la estructura articular. Dinámica y cinemática articular.* http:// http://www.sld.cu/sitios/rehabilitacion-bio/temas.php?idv=18661 [Consultado el 20 de septiembre de 2017]

[13] Battery univesity. (31 de 7 de 2017). Obtenido de http://batteryuniversity.com/learn/article/serial_and_parallel_battery_configurations

[14] Bayer, N., Sivchenko, D., Xu, B., Rakocevic, V., & Habermann, J. (2006). *Transmission timing of signaling messages in IEEE 802.16 based Mesh Networks.* Wireless Conference 2006 - Enabling Technologies for Wireless Multimedia Communications (European Wireless) (págs. 46-50). Atenas: IEEE.

[15] Beni, G., Sahin, E., & Spears, W. M. (2005). *Swarm Robotics*. Springer-Verlag.

[16] Boneham & Turner Ltd. *Pasador cilíndrico de acero inoxidable.* http://www.directindustry.es/prod/boneham-turner/product-115053-1173791.html [Consultado el 10 de septiembre de 2017]

[17] Bowker, John. 1992. *Atlas of Limb Prosthetics.* 2da ed. Editorial Mosby year book. 950 págs.

[18] Brad. (1 de 10 de 2017). Obtenido de http://circuitcalculator.com/wordpress/2006/01/31/pcb-trace-width-calculator/

[19] Brambilla, M., Ferrante, E., Birattari, M., & Dorigo, M. (2013). *Swarm robotics: a review from the swarm engineering perspective*. Nueva York: Springer.

[20] Bryce, Catriona. 2016. *Ideas Cross Centuries and Changes Lives.* https://www.nla.gov.au/blogs/trove/2016/01/08/ideas-cross-centuries-and-change-lives. [septiembre 2017].

[21] Byte Paradigm. 2016. *Introduction to I2C and SPI protocols*. http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols. [septiembre 2017].

[22] *Carga de baterías*. http://www.electroimpulso.com.ar/ENERGIASOLAR/BAT.pdf [octubre de 2017]

[23] Chadwell, Alix, et al. 2016, «*The Reality of Myoelectric Prostheses: Understanding What Makes These Devices Difficult for Some Users to Control*» Frontier in Neurorobotics.

[24] Cook, Roger. 1995. *US Patent 6148597* A. Spirit Lake, Iowa: U.S. Patent and Trademark Office.

[25] Crowder, RM. 1995. *Tactile sensing*. Enero 1995. http://www.soton.ac.uk/~rmc1/robotics/artactile.htm. [septiembre 2017].

[26] Day, M. (2006). *Understanding Low Drop Out (LDO) Regulators*. Texas Instruments, 10. Obtenido de http://www.ti.com/lit/ml/slup239/slup239.pdf

[27] De Smith, N. (2010). *ANSI PCB trace width calculator*. Retrieved agosto 2017, from http://www.desmith.net/NMdS/Electronics/TraceWidth.html

[28] De St. Germain, J. (n.d.). *Computer Architecture*. Retrieved septiembre 2017, from University of Utah: http://www.cs.utah.edu/~germain/PPS/Topics/architecture.html

[29] DEWESoft. *PID Control*. https://www.dewesoft.com/pro/course/pid-control-53 [octubre de 2017]

[30] DFRobot. (2017). *6 DOF Sensor MPU6050*. Retrieved septiembre 2017, from https://www.dfrobot.com/wiki/index.php/6_DOF_Sensor-MPU6050_(SKU:SEN0142)

[31] Digikey. (17 de 9 de 2017). Obtenido de https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-battery-life

[32] Earl, B. (5 de 3 de 2016). *Learn Adafruit*. Obtenido de https://learn.adafruit.com/multi-cell-lipo-charging?view=all

[33] Egerstedt, M. (2008). *Coursera*. Retrieved septiembre 2017, from Georgia Institute of Technology: https://www.coursera.org/learn/mobile-robot

[34] ElecFreaks. (n.d.). *Ultrasonic Ranging Module HC-SR04*. Retrieved marzo 2017, from http://elecfreaks.com/store/download/HC-SR04.pdf

[35] Espressif Inc. (N.A. de N.A. de 2017). *ESP8266EX*. Obtenido de Datasheet: http://espressif.com/en/products/hardware/esp8266ex/resources

[36] Exarchitects. *Introducción a la impresión 3d.* http://exarchitects.com/introduccion-la-impresion-3d/ [Consultado el 13 de septiembre de 2017]

[37] Fairhurst, G. (11 de enero de 2009). *Transport Layer Protocol.* Obtenido de Electronics Research Group of the University of Aberdeen: http://www.erg.abdn.ac.uk/users/gorry/course/inet-pages/transport.html

[38] Fowler, Richard. 1986. *Electricidad: principios y aplicaciones*. 1ra ed. España: Reverte. 378 págs.

[39] Franti, Eduard, et al. 2012. «*Methods of Acquisition and Signal Processing for Myoelectric Control of Artificial Arms*» Romanian Journal of Information Science and Technology. XV (2): 91-105.

[40] Freescale. (2012). *K20 Sub-Family. Technical Data*.

[41] Freescale. (2014). *Kinetis KL26 Sub-Family. Technical data.*

[42] Fried, Limor. *Introducing Pro Trinket Adafruit*. https://learn.adafruit.com/introducing-pro-trinket/pinouts. [septiembre 2017].

[43] Geethanjali, Purushothaman. 2016. «*Myoelectric control of prosthetic hands: state-of-the-art review*». Dove Medical Press Limited. IX (1): 247-255.

[44] Godse, D., & Godse, A. (2009). *Microcontrollers*. Pune, India: Technical Publications.

[45] Guangzhou HC Information Technology Co., Ltd. (N.A. de N.A. de 2012). *HC Serial Bluetooth Products User Instructional Manual*. Obtenido de Makezine HC User Instruction Manuals: https://cdn.makezine.com/uploads/2014/03/hc_hc-05-user-instructions-bluetooth.pdf

[46] Hauert, S., Winfield, A., Liu, W., & Meyer, J. (14 de febrero de 2017). *Swarm Robotics*. Obtenido de Bristol Robotics Laboratory: http://www.brl.ac.uk/researchthemes/swarmrobotics.aspx

[47] IEEE. (2016). 802.11-2016 - *IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY)*. New York City: IEEE.

[48] Interempresas. 2017. *Heller Máquina-Herramienta*, S.L. http://www.interempresas.net/MetalMecanica/FeriaVirtual/Producto-Fresadora-CNC-Helfer-F4-CNC-49510.html [Consultado el 20 de septiembre de 2017]

[49] Interlink Electronics, «*Force Sensing Resistor Integration Guide And Evaluation Parts Catalog*» California. 26 págs.

[50] *Introduccion a los sistemas de control.* http://www.isa.cie.uva.es/~felipe/docencia/ra12itielec/tema1_trasp.pdf [octubre de 2017]

[51] InvenSense. (2013). *MPU-6000/MPU-6050 Register map and descriptons*. Retrieved septiembre 2017, from https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf

[52] ITEAD. (24 de marzo de 2017). *Serial Port Bluetooth Module (Master/Slave): HC-05*. Obtenido de ITEAD Product Wiki: https://www.itead.cc/wiki/Serial_Port_Bluetooth_Module_(Master/Slave)_:_HC-05

[53] *Jasmine robot platform*. (26 de 5 de 2015). Obtenido de http://www.swarmrobot.org/

[54] Jayakrishnan, H. (2014). *Goal-to-goal*. Retrieved septiembre 2017, from Github: https://github.com/hjjayakrishnan/go-to-goal

[55] Kansas City Bone & Joint Clinic. *Problemas ortopédicos en español*. https://www.pinterest.com/pin/127015651968195784/ [Consultado el 1 de octubre de 2017]

[56] Kuniholm, Jonathan. 2009. «*Open Arms*». IEEE Spectrum. IEEE. 46 (3): 36-41

[57] Kuo, Benjamin. 1996. *Sistemas de control automático*. 7ª ed. México: Pearson Educación. 897 págs.

[58] Kurose, J. F., & Ross, K. W. (2016). *Computer Networking a Top-Down Approach*. Boston: Pearson.

[59] Kyberd, Peter. 1995. «*MARCUS: a two degree of freedom hand prosthesis with hierarchical grip control*» IEEE Transactions on rehabilitation Engineering, III (1); 70-76.

[60] Learning about electronics. (2 de 7 de 2017). Obtenido de http://www.learningaboutelectronics.com/Articles/What-is-a-bypass-capacitor.html

[61] Lee, J.-S., Su, Y.-W., & Shen, C.-C. (2007). *A Comprative Study of Wireless Protocols: Bluetooth, UWB, Zigbee, and Wi-Fi.* The 33rd Annual Conference of the IEEE Industrial Electronics Society (págs. 46-51). Taipei: IEEE.

[62] Levy, S. (2017). *MPU6050*. Retrieved Julio 2017, from Github: https://github.com/simondlevy/MPU6050

[63] Loaiza, Jair; Arzola, Nelson. 2011. «*Evolución y tendencias en el desarrollo de prótesis de mano*». Dyna. [Medellín] 78 (169): 191-200

[64] Lopez-Buedo, S., & Boemo, E. (n.d.). *Electronick Packaging Technologies*. Retrieved septiembre 2017, from Universidad Autonoma de Madrid: http://www.doe.carleton.ca/~tjs/10-Packaging.pdf

[65] Loughhead, P. (2017). *From Idea to manufacture-Driving a PCB design through Altium Designer*. Retrieved from Altium Designer Documentation: http://www.altium.com/documentation/17.1/display/ADES/From+Idea+to+Manufacture+-+Driving+a+PCB+Design+through+Altium+Designer

[66] Lucas, Jim. 2015. *What Is Electromagnetic Radiation*?. https://www.livescience.com/38169-electromagnetism.html. [septiembre 2017].

[67] Margtecnología. *Baterías de ácido-plomo y cargadores*. http://margtecnologia.blogspot.com /2013/07/baterias-de-acido-plomo-y-cargadores.html [octubre de 2017]

[68] Marrakchi, D. (2016). *Top 5 PCB Design Guidelines every PCB designer needs to know*. Retrieved septiembre 2017, from Altium: https://resources.altium.com/pcb-design-blog/top-pcb-design-guidelines-every-pcb-designer-needs-to-know

[69] Marshall, John. 2015. *The history of prosthetics*. http://unyq.com/the-history-of-prosthetics/. [octubre 2017].

[70] Mazariegos, Pablo. 2012. «*Diseño e implementación de un nuevo modelo de la mano de la prótesis Biónica Transhumeral*» Tesis Universidad del Valle de Guatemala, 151 págs.

[71] Mazzone, Virginia. 2002. *Controladores PID*. Universidad Nacional de Quilmes. 12 págs.

[72] MCBtec. *Control PID*. http://www.mcbtec.com/que_es_un_control_PID.html [octubre de 2017]

[73] McLeod, R. (2014). *Lecture 9 - Transistors*. Retrieved abril 2017, from University of Colorado: http://ecee.colorado.edu/~mcleod/pdfs/IADE/lectures/ECEN%201400%20Lecture%209%20Transistors.pdf

[74] McLurkin, J., Rykowski, J., John, M., Kaseman, Q., & Lynch, A. J. (2013). *Using Multi-Robot Systems for Engineering Education: Teaching and Outreach With Large Numbers of an Advanced, Low-Cost Robot*. IEEE Transactions on Education, 24-33.

[75] Microchip. 2001. *PIC16F84A* Datasheet. http://ww1.microchip.com/downloads/en/DeviceDoc/35007b.pdf [octubre 2017]

[76] Microchip. *PIC16AF84A*. http://www.microchip.com/wwwproducts/en/PIC16F84A. [septiembre 2017].

[77] Mikkelsen, S., & Jespersen, R. (2010). *Robotic Swarm - Development and Formation*. Aalborg University, Aalborg.

[78] Mondada, F. (3 de 3 de 2011). *Ascens*. Obtenido de http://blog.ascens-ist.eu/category/robots/

[79] Morris, Phil. 2014. *The nRF24L01+ RF Module Part 1*. http://www.lydiard.plus.com/hardware_pages/nrf24l01_1.htm. [septiembre 2017].

[80] Mott, Robert L. 2006. *Diseño de elementos de máquinas*. 4ta ed. México. 944 págs.

[81] Mouser Electronics. (n.d.). *Texas Instruments SN74HC04DR*. Retrieved septiembre 2017, from http://www.mouser.com/ProductDetail/Texas-Instruments/SN74HC04DR/?qs=sGAEpiMZZMutVWjHE%2fYQw0qCn7H6iZbYRqjLH55J6sQ%3d

[82] Mouser Electronics. (n.d.). *Texas Instruments SN74HC04N*. Retrieved septiembre 2017, from http://www.mouser.com/ProductDetail/Texas-Instruments/SN74HC04N/?qs=sGAEpiMZZMutVWjHE%2fYQw0qCn7H6iZbYIvN0Omn30H4%3d

[83] National Instruments. (2010*). PCB design fundamentals: overview*. Retrieved septiembre 2017, from http://www.ni.com/tutorial/10580/en/

[84] National Instruments. (2017). *Best practices in printed circuit board design*. Retrieved septiembre 2017, from http://www.ni.com/tutorial/6894/en/

[85] Navarro, I., & Matía, F. (2013). *An introduction to swarm robotics. International Scholarly Research Notices*, 2013(1), 10.

[86] Nelson, V. (2017). *ARM and STM32F4xx Operating modes & interrupt handling*. Retrieved from Auburn university: http://www.eng.auburn.edu/~nelson/courses/elec5260_6260/slides/ARM%20STM32F407%20Interrupts.pdf

[87] Nordic Semiconductor. *nRF24L01+, Ultra low power 2.4Ghz RF transceiver IC*. https://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01P. [Septiembre 2017].

[88] Norton, Kim. 2007. «*Un breve recorrido por la historia de la protésica*,» inMotion. XVII (7): .

[89] Ober, C. (2006). *Advanced packaging technology for leading edge microelectronics and flexible electronics*. Retrieved septiembre 2017, from Cornell University: http://people.ccmr.cornell.edu/~cober/mse542/page2/files/Fillion%20GE.pdf

[90] Ogata, Katsuhiko. 2003. *Ingeniería de control moderna*. 4ª ed. Madrid: Pearson Educación. 965 págs.

[91] Oliver, John. 2017. *Muscles in the posterior compartment of the forearm*. http://teachmeanatomy.info/upper-limb/muscles/posterior-forearm/. [Septiembre 2017].

[92] PCB power. (3 de 4 de 2015). Obtenido de http://www.pcbpower.com/important-considerations-designing-multi-layer-board/

[93] Pérez, Francisco; J. Juan. 2010. *Apuntes de electricidad aplicada a los buques*. 2ª ed. San Vicente: Club Universitario. 360 págs.

[94] Pérez, Mario, et al. 2007. *Introduccion a los sistemas de control y modelo matemático para sistemas lineales invariantes en el tiempo*. Universidad Nacional de San Juan. 69 págs.

[95] PJRC Electronic Projects. (N.A. de N.A. de 2017*). Teensy USB Development Board*. Obtenido de PJRC Store: https://www.pjrc.com/store/teensy32.html

[96] PJRC Electronics. (13 de marzo de 2017). *Teensy LC - Low Cost*. Obtenido de PJRC Electronic Projects: https://www.pjrc.com/teensy/teensyLC.html

[97] Prometec. *Qué es un servo*. https://www.prometec.net/servos/# [Consultado el 1 de octubre de 2017]

[98] Pylatiuk, Christian, et al. 2004. «*Two Multiarticulated Hydraulic Hand Prostheses*» Artificial Organs. XXVIII (11): 980-986.

[99] Quadra. (19 de 02 de 2008). Obtenido de https://web.archive.org/web/20060822171843/http://quadrasol.co.uk/cadsys/inventor.php

[100] Quinayás, Burgos. 2010. «*Diseño y Construcción de una Prótesis Robótica de Mano Funcional Adaptada a Varios Agarres*». Tesis Universidad del Cauca. 94 págs.

[101] Rafiquzzaman, Mohamed. 2014. *Fundamental of Digital Logic and Microcontrollers*. 5ta ed. Nueva Jersey: John Wiley & Sons. 840 págs.

[102] Raspberry Pi Organization. *Raspberry Pi Zero*. https://www.raspberrypi.org/products/raspberry-pi-zero/#buy-now-modal. [septiembre 2017].

[103] Reed, Frances. *How Do Servo Motors Work*. https://www.jameco.com/jameco/workshop/howitworks/how-servo-motors-work.html [Consultado el 1 de septiembre de 2017]

[104] Renobat. *Cargadores de baterías y métodos de carga*. http://www.renobat.eu/es/comunicacion/articulos/183-cargadores-de-baterias-y-metodos-de-carga [octubre de 2017]

[105] researchgate. (5 de 6 de 2017). Obtenido de https://www.researchgate.net/figure/318827556_fig16_Figure-18-LiPo-battery-structure-Skylightupowercom-2017

[106] Rocha, José; E. Lara. 2010. *Introduccion a los sistemas de control*. Universidad Autónoma de Nuevo León. 10 págs.

[107] Rosenberg, Paul. 1999. *The basics of data communications*: Copper media. http://www.ecmweb.com/cee-news-archive/basics-data-communications-copper-media [Octubre 2017]

[108] Rozenblat, L. (1 de 10 de 2017). *SMPS*. Obtenido de http://www.smps.us/layout.html

[109] Rubenstein, M., Ahler, C., Hoff, N., Cabrera, A., & Nagpal, R. (2014). *Kilobot: A low cost robot with scalable operations designed for collective behaviors*. En Robotics and Autonomous Systems (págs. 966-975). Nueva York: Elsevier.

[110] Santamaría, Germán; A. Castejón. 2011. *Baterías y acumuladores*. Madrid: Editex. 48 págs.

[111] Scheneider, B. (12 de 09 de 2017). Roger's Hobby Center. Obtenido de https://rogershobbycenter.com/lipoguide/

[112] Schuba, C. L., Krsul, I. V., Kuhn, M. G., Spafford, E. H., Sundaram, A., & Zamboni, D. (1997). *Analysis of a denial of service attack on TCP*. 1997 IEEE Symposium on Security and Privacy (págs. 208-223). Oakland: IEEE.

[113] SIDEI Ingenieros LTDA. 2011. *Conceptos básicos de carga de baterías y algoritmos de carga*. Chile:SIDEI. 4 págs.

[114] Smith, N. (13 de 6 de 2010). Obtenido de desmith: http://www.desmith.net/NMdS/Electronics/TraceWidth.html

[115] Smith, N. *ANSI PCB Trace Width Calculator*. http://www.desmith.net/NMdS/Electronics/TraceWidth.html [octubre de 2017]

[116] Sosa, Liliana. 2016. *Compuertas lógicas en Tecnología TTL – Niveles Lógicos*. http://unicrom.com/compuertas-logicas-tecnologia-ttl-niveles-logicos/. [Septiembre 2017].

[117] South Australian Medical Heritage Society. *The hand of corporal Coles*. http://samhs.org.au/Virtual%20Museum/Surgery/hand%20of%20Corporal%20Coles/hand%20of%20Corporal%20Coles.html. [Octubre 2017].

[118] Sparkfun Electronics. (N.A. de N.A. de N.A.). *WiFi Module - ESP8255*. Obtenido de Sparkfun Electronics: https://www.sparkfun.com/products/13678

[119] SparkFun. 2015. *SparkFun Transceiver Breakout - nRF24L01+*. https://www.sparkfun.com/products/691. [Septiembre 2017].

[120] Stoffegren, P. (2013). *Parallel GPIO on Teensy 3.0*. Retrieved septiembre 2017, from Forum PJRC: https://forum.pjrc.com/threads/23950-Parallel-GPIO-on-Teensy-3-0?p=34158&viewfull=1#post34158

[121] Stoffregen, P. (2013). *AVR like ISR*. Retrieved septiembre 2017, from Forum PJRC: https://forum.pjrc.com/threads/17760-AVR-like-ISR

[122] Stoffregen, P. (n.d.). *Teensy LC USB development board*. Retrieved septiembre 2017, from PJRC: https://www.pjrc.com/store/teensylc.html

[123] Stoffregen, P. (n.d.). *Teensy Schematic*. Retrieved Septiembre 2017, from PJRC: https://www.pjrc.com/teensy/schematic.html

[124] Stoffregen, P. (n.d.). *Teensy technical specifications*. Retrieved from PJRC: https://www.pjrc.com/teensy/techspecs.html

[125] Stoffregen, P. (n.d.). *Teensy USB development board*. Retrieved septiembre 2017, from PJRC: https://www.pjrc.com/teensy/index.html

[126] Sun Microsystems Inc. (2000). *System Administration Guide*, Volume 3. Palo Alto: Sun Microsystems.

[127] Texas Instruments. (2015). *SNx4HC04 Hex Inverters*. Retrieved abril 2017, from http://www.ti.com/lit/ds/symlink/sn74hc04.pdf

[128] Texas Instruments. (2016). *SNx4HC08 Quadruple 2-Input Positive-AND Gates*. Retrieved abril 2017, from http://www.ti.com/lit/ds/symlink/sn74hc08.pdf

[129] Texas Instruments. (2016). *SNx4HC138 3-Line To 8-Line Decoders/Demultiplexers*. Retrieved abril 2017, from http://www.ti.com/lit/ds/symlink/sn74hc138.pdf

[130] Texas Instruments. (2016). *SNx4HC32 Quadruple 2-Input Positive-OR Gates*. Retrieved abril 2017, from http://www.ti.com/lit/ds/symlink/sn74hc32.pdf

[131] Texas Instruments. (2017 de 10 de 4). Obtenido de www.ti.com/Webench

[132] TMRh20. 2014. *Optimized High Speed NRF24L01+ Driver Class Documenation.* http://tmrh20.github.io/RF24/index.html. [octubre 2017].

[133] Torres, J. C. (13 de 9 de 2017). Universidad De Granada. Obtenido de http://lsi.ugr.es/~cad/teoria/Tema1/RESUMENTEMA1.PDF

[134] Traister, J. (2012). *Design guidelines for Surface Mount Technology*. Bentonville, Virginia: Elsevier.

[135] Treffers, C., & Wietmarschen, L. v. (2016). *Position and orientation determination of a probe with use of the IMU MPU9250 and a AT328 microcontroller*. TU Delft, Delft.

[136] Universidad Politécnica de Valencia. *Materiales Poliméricos y Compuestos*. https://www.upv.es/materiales/Fcm/Fcm15/fcm15_3.html [Consultado el 30 de septiembre de 2017]

[137] Varteresian, J. (2002). *Fabricating Printed Circuit Boards*. Woburn, Massachussets: Newnes.

[138] Veevus. G.S.P Thread. http://veevus.com/gsp-thread/18-gsp-thread.html [Consultado el 16 de septiembre de 2017]

[139] W3 Organization. (N.A. de junio de 2007). *Hypertext Transfer Protocol -- HTTP/1.1*. Recuperado el 1 de agosto de 2017, de https://www.w3.org/Protocols/HTTP/1.1/rfc2616bis/draft-lafon-rfc2616bis-03.html

[140] Wang, H., Jin, Y., & Wang, W. (2003). *The Performance Comparison of PRSCTP, TCP and UDP for Mpeg-4 Multimedia Traffic in Mobile Network*. International Conference on Communication Technology (págs. 403-406). Beijing: IEEE.

[141] Wikipedia. (13 de 9 de 2017). Obtenido de https://es.wikipedia.org/wiki/Autodesk_Inventor

[142] Wikipedia. (13 de 9 de 2017). Obtenido de https://www.google.com.gt/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FSwarm_robotic_platforms&psig=AFQjCNG2jp3t84WME0pOahI8dZmheilI2Q&ust=1505365778923628

[143] Woodford, C. (14 de 7 de 2017). *Explain that stuff!* Obtenido de http://www.explainthatstuff.com/how-lithium-ion-batteries-work.html

[144] Zarya. (30 de noviembre de 2014). *ESP8266*. Obtenido de NURDSpace: https://nurdspace.nl/ESP8266

XIV. ANEXOS

A. PRÓTESIS BIÓNICA

Programación del controlador en Pro Trinket:

```
#include <SPI.h>
#include "RF24.h"

RF24 radio(4,8); //ce y cs respectivamente
unsigned long tiempo_aux, tiempo, mensaje;

//variables
int lectura, check, cambio;
unsigned long start_time;

//Canales de comunicación
byte addresses[][6] = {"1Node","2Node"}; //deben ser iguales en ambos lados

void setup() {
 Serial.begin(115200);
 radio.begin();
 radio.setPALevel(RF24_PA_LOW); //Se diminuye la potencia de la antena
 //configurar canales de entrada y salida
 radio.openWritingPipe(addresses[0]);
 radio.openReadingPipe(1,addresses[1]);
}

void loop() {
 lectura = analogRead(A0); //leer el sensor
 //ver si el sensor esta suficiente presionado
 if(lectura > 500){
  check = 1;
 } else {
  check = 0;
 }  //verificar cambios en el sensor positivos
 if(check == 1 && cambio == 0){
```

```
cambio = 1;
   tiempo_aux =  millis();
  }
  if(check == 0 && cambio == 1){ // y negativos
   cambio = 0;
   tiempo = millis() - tiempo_aux;
  }
  //clasificacion de tiempo de las senales
  if(tiempo != 0){
   Serial.println("enviando mensaje");
   if(tiempo > 100){
    if(tiempo <600){
     Serial.println("PULSO CORTO");
     mensaje = 11695 + 1; //ID del transmisor + mensaje
     //11695 para el emisor 1 y 13785 para el emisor 2
    }else if(tiempo <1000){
     Serial.println("PULSO MEDIO");
     mensaje = 11695 + 2;
    }else{
     Serial.println("PULSO LARGO");
     mensaje = 11695 + 3;
    }
   }

   //enviando el mensaje
   if (!radio.write(&mensaje,sizeof(unsigned long) )){
    Serial.println(F("F"));
   }
   tiempo = 0; //limpiar variables
  }
}
```

Programación del microcontrolador principal:

```
#include <SPI.h>
#include "RF24.h"
#include <Servo.h>
```

```
//modulo de radiofrecuencia
RF24 radio(4,8); //ce y cs respectivamente

//Variables
unsigned long mensaje;
Servo pulgar, indice, medio, anular_menique, muneca;
int s_pulgar, s_indice, s_medio, s_anular, s_menique; //pines analógicos
double f_pulgar, f_indice, f_medio, f_anular, f_menique; //fuerzas calculadas
const double e = 2.71828;

//Canales de comunicacion
byte addresses[][6] = {"1Node","2Node"}; //deben ser iguales en ambos lados


void setup() {
  Serial.begin(115200);
  radio.begin();
  radio.setPALevel(RF24_PA_LOW); //se disminuye la potencia de la antena

  //configurar canales de entrada y salida
  radio.openWritingPipe(addresses[1]);
  radio.openReadingPipe(1,addresses[0]);
  //actuadores
  pulgar.attach(3);
  pulgar.attach(5);
  pulgar.attach(6);
  pulgar.attach(9);
  pulgar.attach(10);
  //Sensores
  s_pulgar = A0;
  s_indice = A1;
  s_medio = A2;
  s_anular = A3;
  s_menique = A4;
}
void loop() {
  f_pulgar = fuerza(analogRead(s_pulgar));
```

```
f_indice = fuerza(analogRead(s_indice));
f_medio = fuerza(analogRead(s_medio));
f_anular = fuerza(analogRead(s_anular));
f_menique = fuerza(analogRead(s_menique));

if(radio.available()){
  while(radio.available()){
    radio.read(&mensaje,sizeof(unsigned long)); //lee el mensaje y limpia el bus
  }
  Serial.print("Mensaje recibido: ");
  Serial.println(mensaje);
}

if(mensaje != 0){
  if(mensaje == 11696){ //emisor 1
    pulgar.write(25);
    indice.write(15);
    medio.write(15);
    anular_menique.write(15);
  } else if(mensaje == 11697){
    pulgar.write(25);
    indice.write(15);
    medio.write(120);
    anular_menique.write(120);
  } else if(mensaje == 11698){
    pulgar.write(80);
    indice.write(120);
    medio.write(15);
    anular_menique.write(15);
  }
  if(mensaje == 13786){ //emisor 2
    pulgar.write(60);
    indice.write(80);
    medio.write(80);
    anular_menique.write(80);
  } else if(mensaje == 13787){
    pulgar.write(25);
```

```
    indice.write(100);
    medio.write(100);
    anular_menique.write(100);
   } else if(mensaje == 13788){
    indice.write(120);
    medio.write(120);
    anular_menique.write(120);
    pulgar.write(60);
   }
   mensaje = 0;
  }
}

double fuerza(int lectura){
  double resultado;
  resultado = pow(e,(lectura + 81.315)/42.886);
  return resultado;
}
```

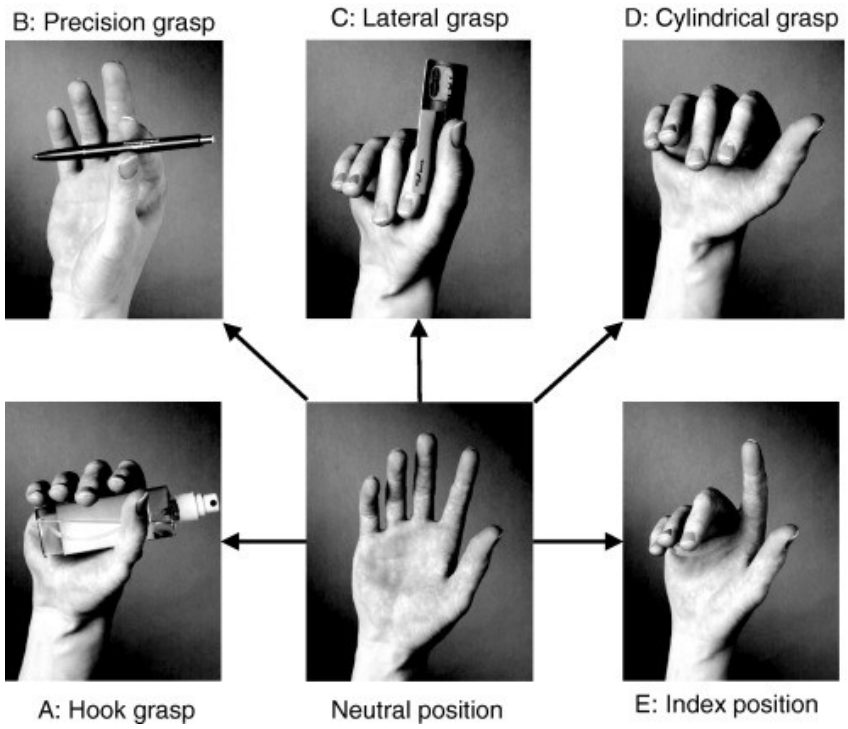
A continuación, se presenta una tabla en la cual se muestra el rango de movimiento, en términos angulares para el Arduino, sobre la cual se trabajó para las posiciones. El valor más bajo corresponde la posición extendido del dedo y el valor alto a la posición tensada. Estos rangos fueron obtenidos experimentalmente.

Tabla LXXIV. Rango de movimiento angular de los servomotores.

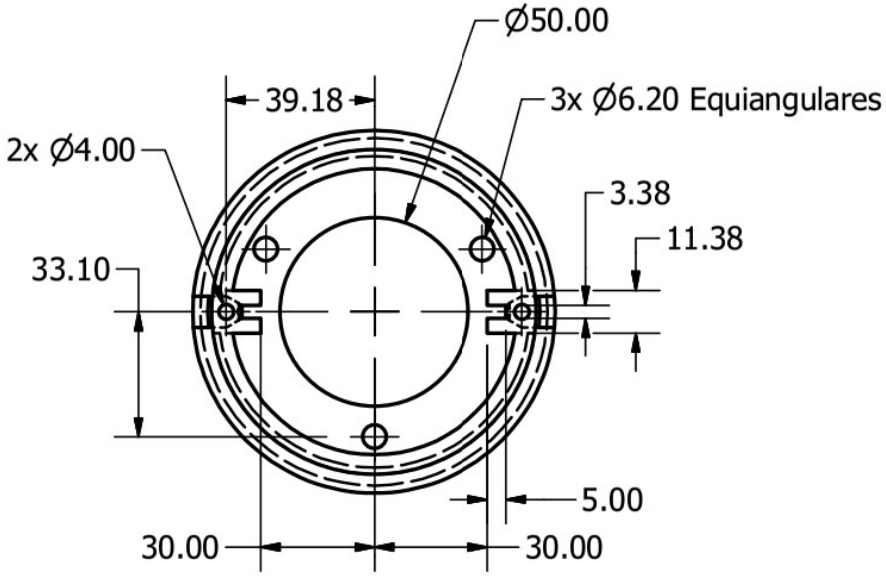
Servomotor	**Rango**	
	mínimo	**Máximo**
Pulgar	20	100
Indice	10	140
Medio	10	140
Anular y meñique	10	140
Muñeca	30	150

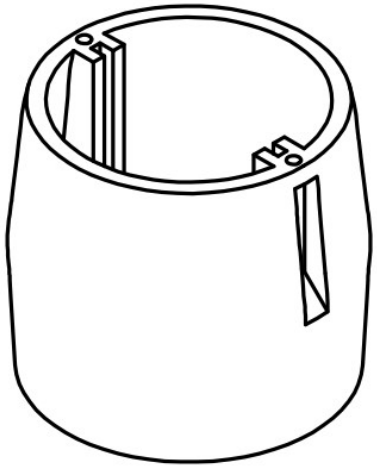
Las posiciones que se desean obtener son las siguientes:

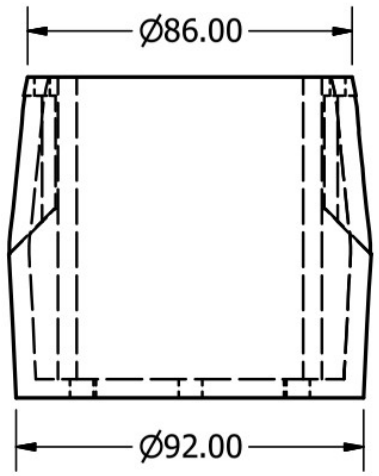
Figura 341. Poses de la mano.

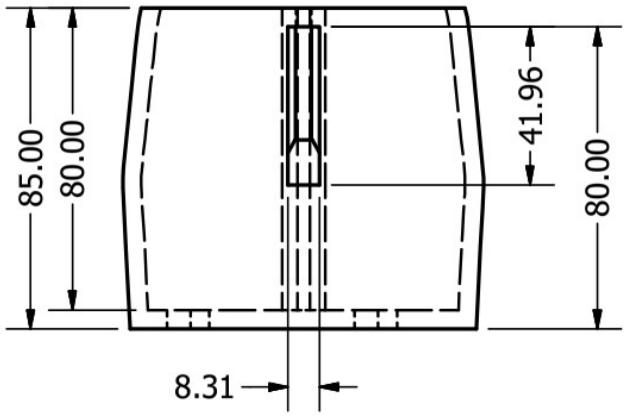


[10]









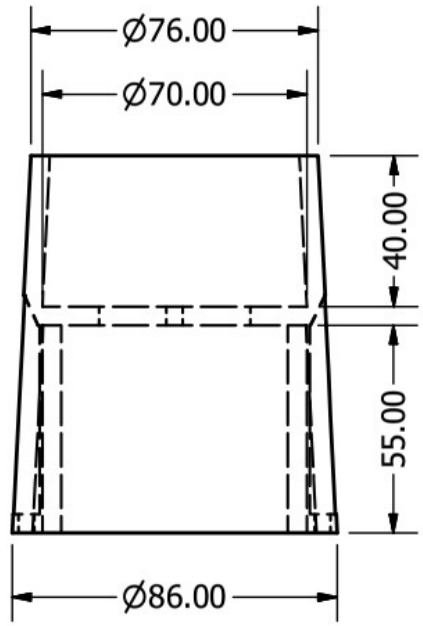
Nota: A menos que se indique lo contario, todas las medidas son en milímetros.

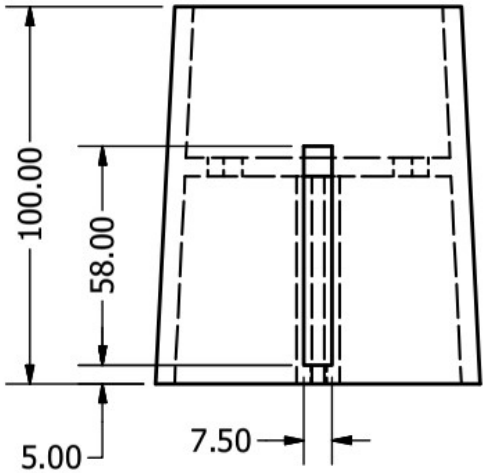
Diseñado por:	Dibujado por:	Fecha	Escala
Erick Ivan Hernández Woc 13197	Erick Ivan Hernández Woc 13197	2017-10-05	1 : 2

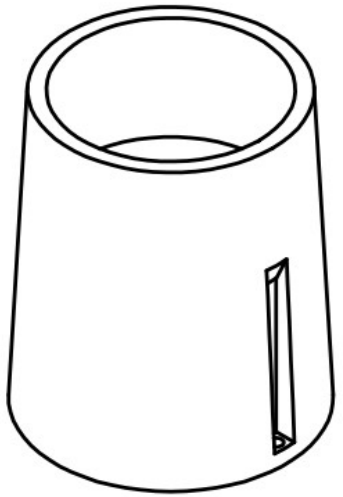
Universidad del Valle de Guatemala

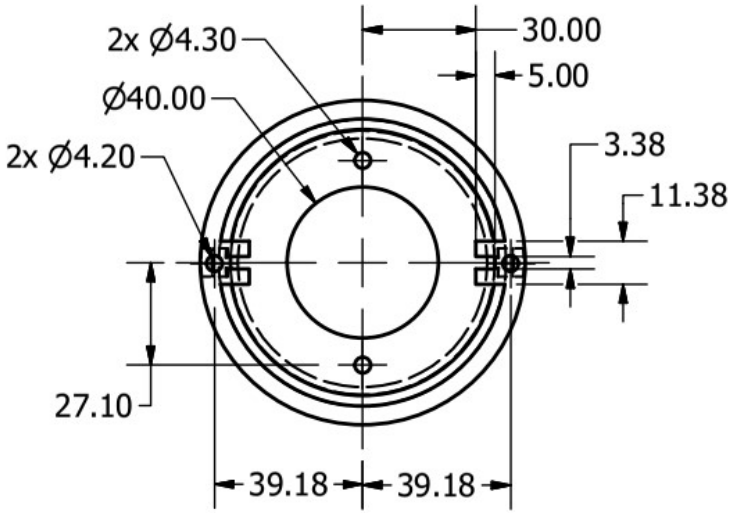
Sección inferior del antebrazo

Título del proyecto: Protolife









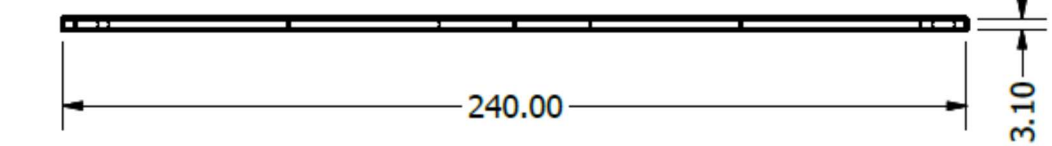
Nota: A menos que se indique lo contario, todas las medidas son en milímetros.

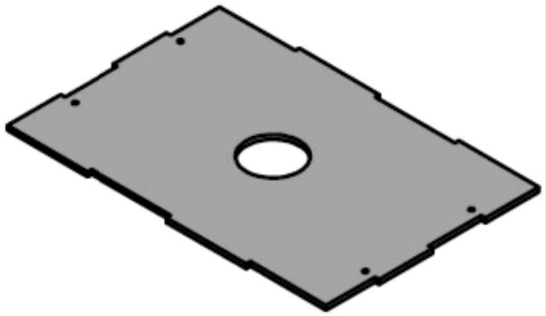
Diseñado por:	Dibujado por:	Fecha	Escala
Erick Ivan Hernández Woc 13197	Erick Ivan Hernández Woc 13197	2017-10-05	1 / 2

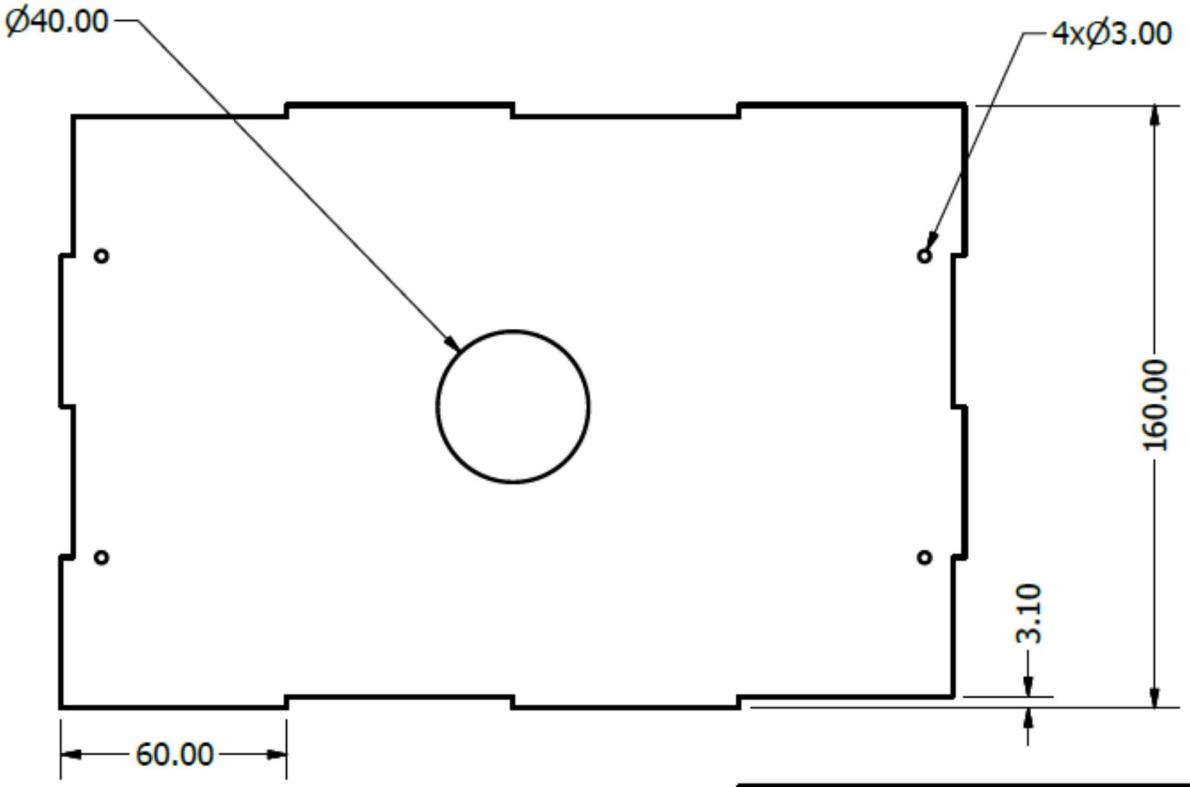
Universidad del Valle de Guatemala

Sección superior del antebrazo

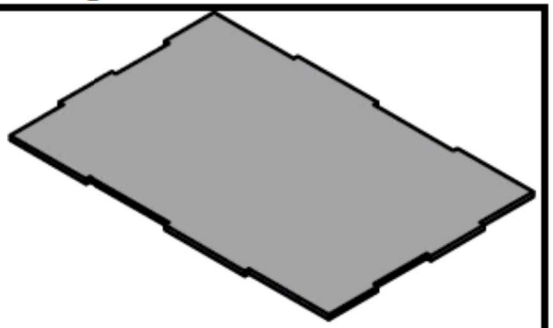
Título del proyecto: Protolife

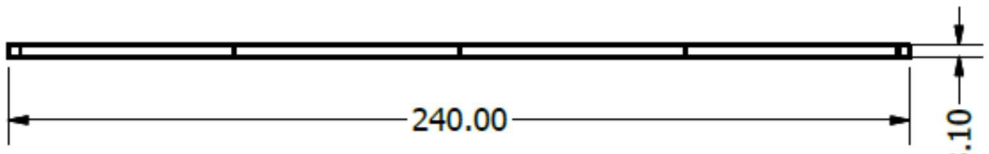


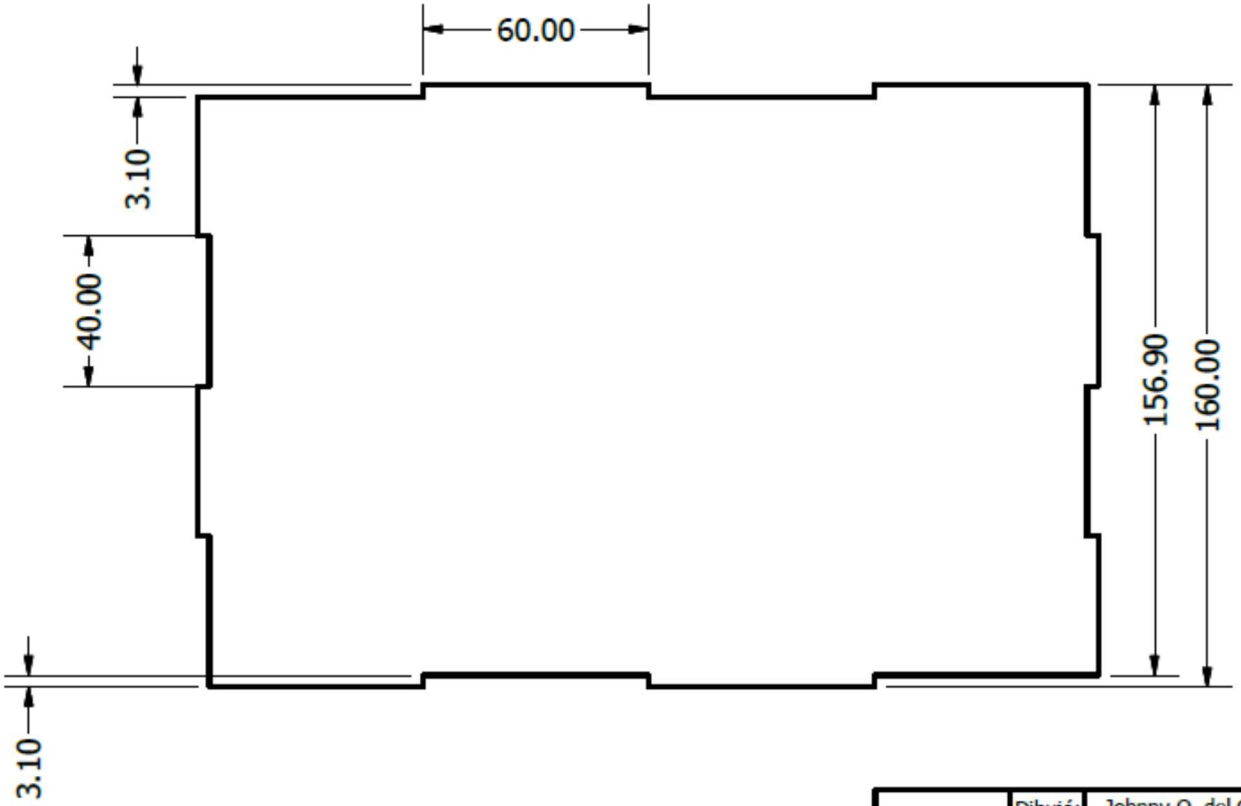




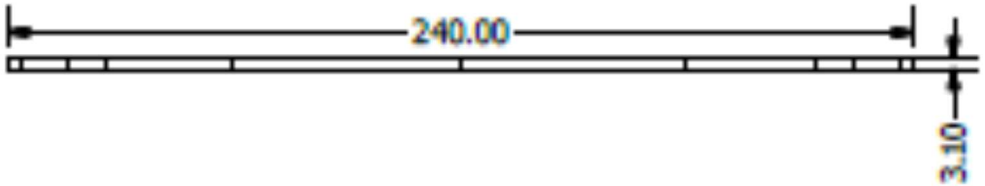
PROTOLIFE	Dibujó:	Johnny O. del Cid G. 13032	Fecha	Universidad Del Valle de Guatemala
	Diseñó:	Johnny O. del Cid G. 13032	05/10/2017	
Escala: 1/2	**Tapadera**			

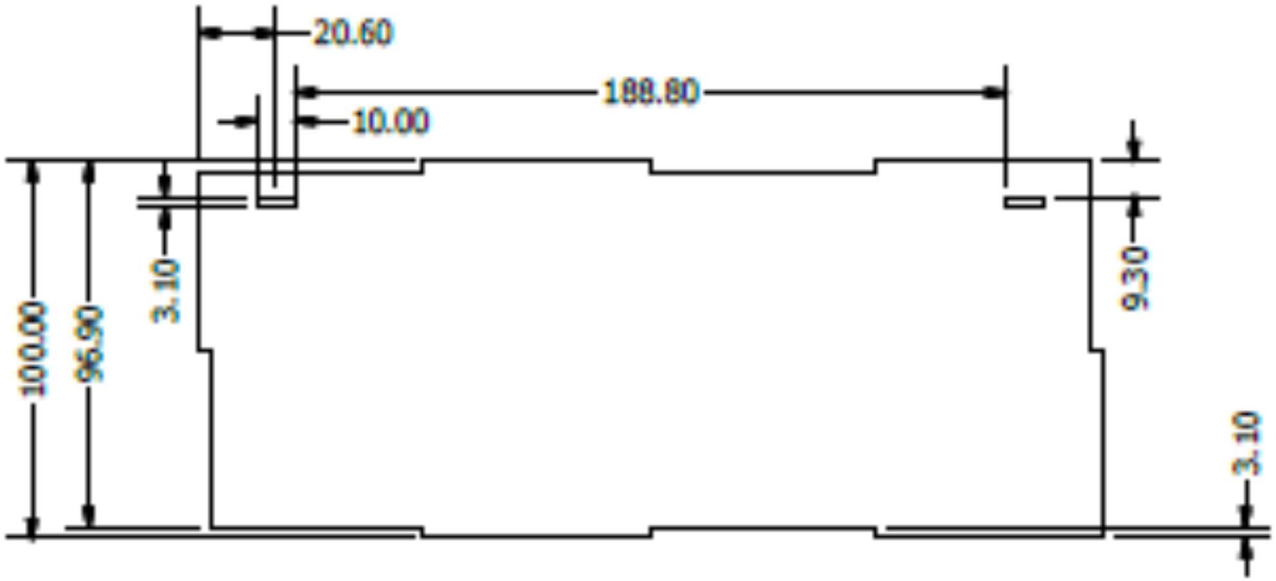




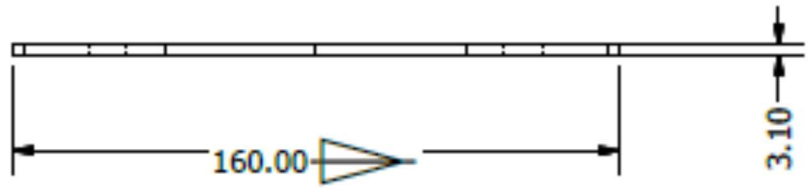


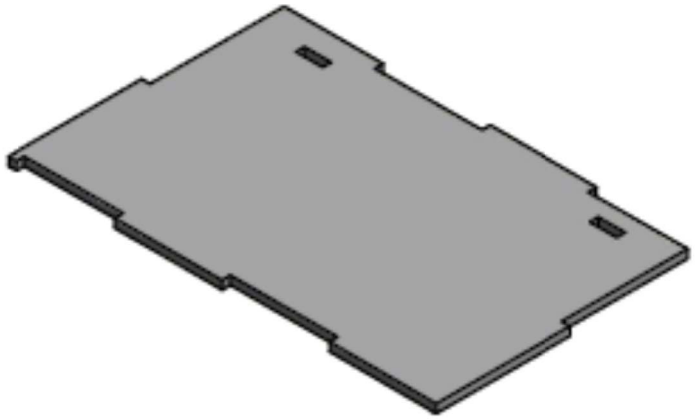
PROTOLIFE	Dibujó:	Johnny O. del Cid G. 13032	Fecha	Universidad Del Valle de Guatemala
	Diseñó:	Johnny O. del Cid G. 13032	05/10/2017	
Escala: 1/2	**Base**			

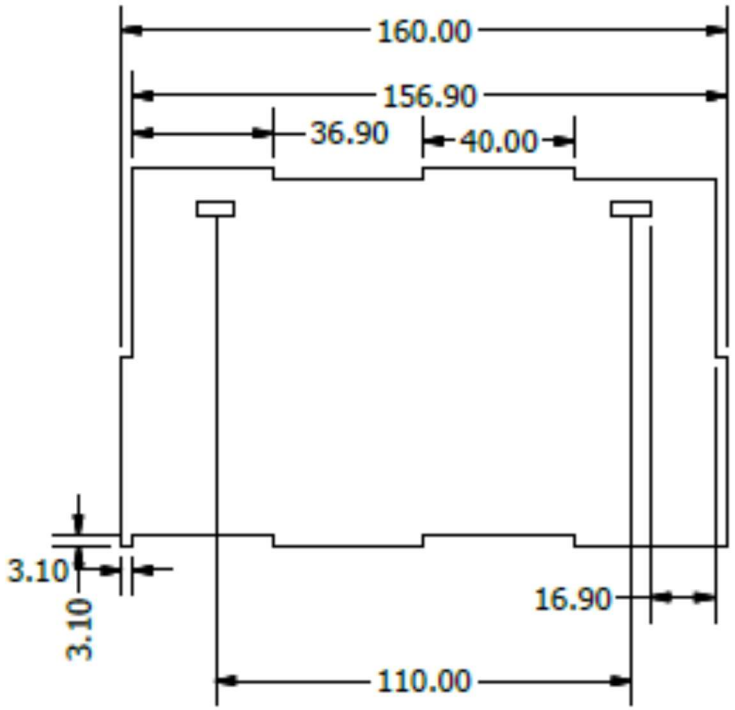




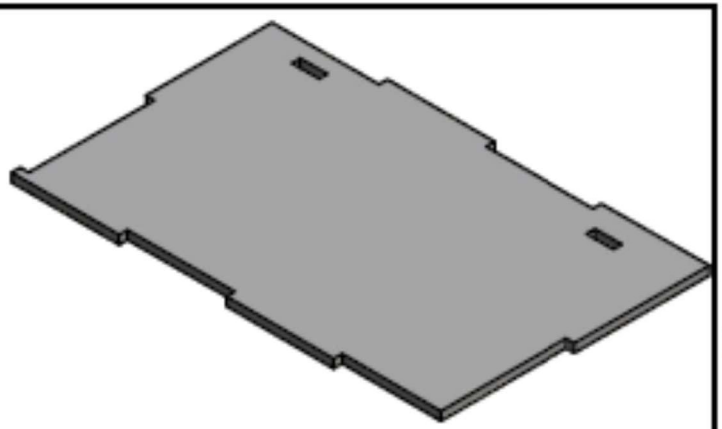
PROTOLIFE	Dibujó:	Johnny O. del Cid G. 13032	Fecha	Universidad Del Valle de Guatemala
	Diseñó:	Johnny O. del Cid G. 13032	05/10/2017	
Escala: 1/2	**Parte Frontal**			

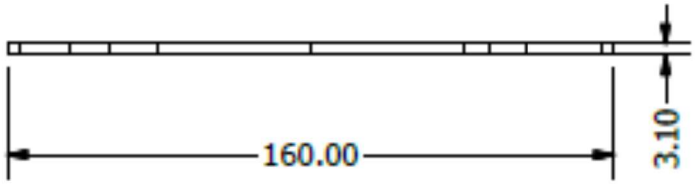


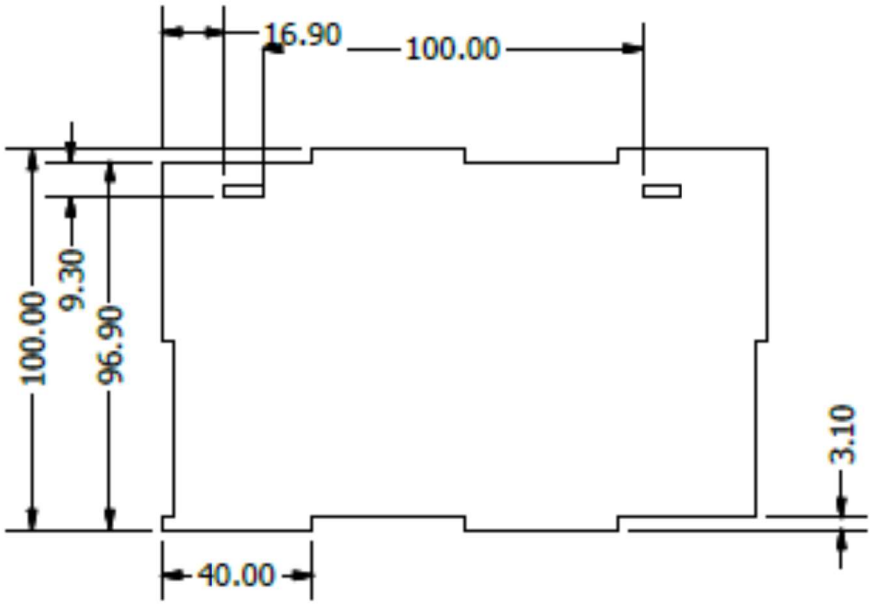




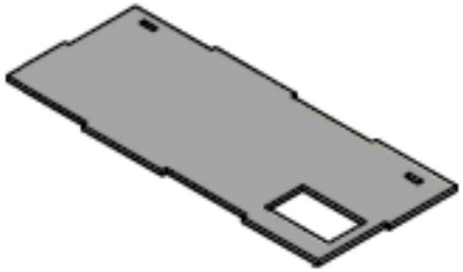
PROTOLIFE	Dibujó:	Johnny O. del Cid G. 13032	Fecha	Universidad Del Valle de Guatemala
	Diseñó:	Johnny O. del Cid G. 13032	05/10/2017	
Escala: 1/2	**Lateral Izquierdo**			

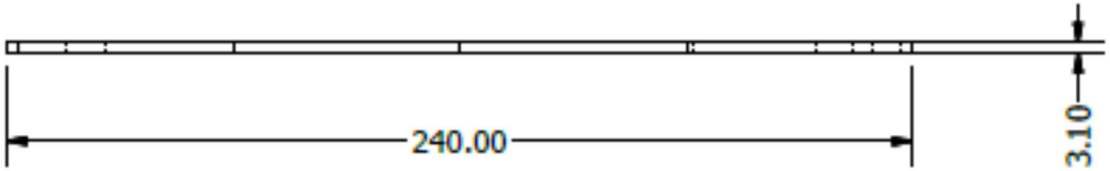


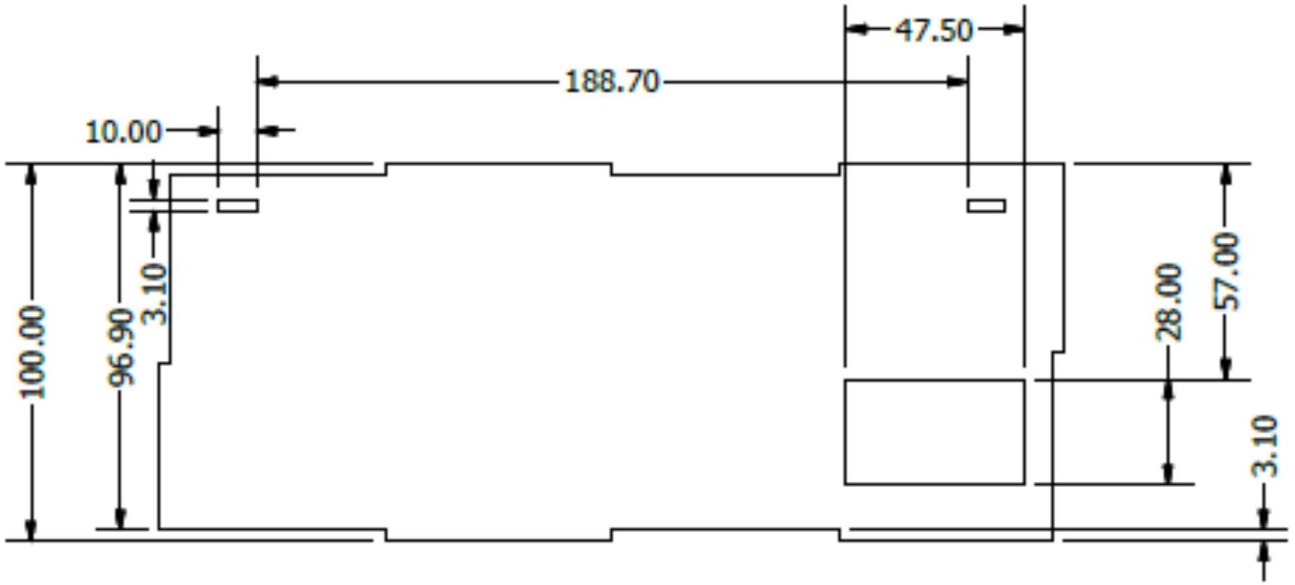




PROTOLIFE	Dibujó:	Johnny O. del Cid G. 13032	Fecha	Universidad Del Valle de Guatemala
	Diseñó:	Johnny O. del Cid G. 13032	05/10/2017	
Escala: 1/2	**Lateral Derecho**			







PROTOLIFE	Dibujó:	Johnny O. del Cid G. 13032	Fecha	Universidad Del Valle de Guatemala
	Diseñó:	Johnny O. del Cid G. 13032	05/10/2017	
Escala: 1/2	**Parte Trasera**			

A. SWARM ROBOTICS

a. Programa realizado para el manejo del microcontrolador y su comunicación a través del módulo ESP-8266.

```
/*
************************************************************
Titulo: SwarmBot_Final.ino
Autores: William Orozco,Otto Wantland y Vidal Villegas
Funcion: Programa para el manejo del teensy.
************************************************************
*/



/*
************************************************************
                    Variables
************************************************************
*/



#define BUFFER_SIZE 1024                                // Tamanio de nuestro Buffer
/*
*****************************************************************************************************
***
**********************************************ESP-
8266**************************************************
*****************************************************************************************************
***
*/
```

```
#define SSID  "dd-wrt"                                  // SSID de nuestra Red
#define PASS  ""                                        // Contrasenia
#define PORT  "8080"                                    // Definimos nuestro puerto a usar
#define TIME_OUT_SERIAL_AVAILABLE 200000

#define DESBORDES_TO_HCSR04 6

//Conexion
char buffer[BUFFER_SIZE];                               // Buffer de los datos transmitidos
int vez = -1;
int data[] = {0,0,0,0,0,0,0,0};                         // Array de los datos de nuestros distintos sensores
int options[] = {-1,-1};                                // Maneja las posibles acciones dentro de nuestra interrupcion
String input;                                           // Guarda nuestros datos de entrada del serial
int largo;                                              // Guarda el length de nuestros datos de salida
int max_v = -1;                                         // Variables para el manejo de las ordenes
int max_i = 0;
char OKrn[] = "OK\r\n";                                 // Respuesta esperada tras cada instruccion al ESP-8266
int finished = -1;
char character;
unsigned long t_esp_resp, tPrev_serialAvailable, t_serialAvailable;
unsigned long timeOut_serialAvailable;
bool found;
int len_esp_resp;
int index_resp;
```

```
String message;
int lengo_data_send;

volatile bool flagRecepcionSerial = false;                  //Bandera de recepcion serial
boolean flagDataSend = false;
volatile bool flagDriverMotores = false;
volatile byte contador_desbordes_to_hcsr04 = 0;
/*
**********************************************************************************************
***
**********************************************Falda
Sensores**********************************************
**********************************************************************************************
***
*/
#define TRIG_PIN_0 6
#define TRIG_PIN_1 5
#define TRIG_PIN_2 4
#define ECHO_PIN_0 7
#define TIME_OUT_PULSE 50000

const byte SENSOR_HCSR04[] = {0,1,2,3,4,5}; //orden en que se hace el barrido de los
sensores ultrasonicos

unsigned int duration_0, duration_1 , duration_2, duration_3, duration_4, duration_5;
unsigned int distance0, distance1, distance2, distance3, distance4, distance5;


volatile byte index_sensor_hcsr04;
```

```
volatile bool flagFallingEcho, flagSerialSend, flagCambioCodigoUltrasonico, flagIMU;
volatile unsigned long t_actual, t_anterior;
IntervalTimer echoTimeOut, timerEncoder; //objetos timer



//*************************************************************************************************
**
//*****************************************Motores*************************************************
****
//*************************************************************************************************
**
int velL = 255;
int velR = 255;
bool dirL = 1;
bool dirR = 1;
const byte fr = 80;                                    // A pin -> the interrupt pin 0
const byte encoder0_pinA = 2;                          // A pin -> the interrupt pin 0
const byte encoder0_pinB = 3;                          // B pin -> the digital pin 4
const byte encoder1_pinA = 11;                         // A pin -> the interrupt pin 0
const byte encoder1_pinB = 12;                         // B pin -> the digital pin 4
const byte driver_1A = 22;                             // B pin -> the digital pin 4
const byte driver_1B = 23;                             // B pin -> the digital pin 4
const byte driver_2A = 9;                              // B pin -> the digital pin 4
const byte driver_2B = 10;                             // B pin -> the digital pin 4
byte encoder0_pinALast_1,encoder0_pinALast_2;
volatile int duracion1,duracion2,velocidad1,velocidad2,p1,p2;
bool Direction1,Direction2;
bool flagE1, flagE2 = false;
```

```
bool Lstate1, val1, Lstate2, val2;
volatile String direccion;

/*
*****************************************************************
                    Funciones - Comunicacion
*****************************************************************
*/

//Funcion para establecer un timeout para las respuestas del ESP8266
byte Timeout(int timeout, char* term=OKrn)
{
  t_esp_resp=millis();
  found=false;
  index_resp=0;
  len_esp_resp=strlen(term);
  while(millis()<t_esp_resp+timeout)                    // Espera el tiempo de tiemout o a recibir Okrn
  {
    if(Serial1.available())
    {
      buffer[index_resp++]=Serial1.read();              // Si esta recibiendo algo lo cargamos a nuestro Buffer
      if(index_resp>=len_esp_resp)                      // Si ya llegamos al tamanio de nuestra respuesta, revisamos
      {
        if(strncmp(buffer+index_resp-len_esp_resp, term, len_esp_resp)==0)          // Comparamos lo recibido con el termino
        {
```

```
            found=true;
            break;
          }
        }
      }
    }
    buffer[index_resp]=0;
    return found;
  }

  //Funcion para establecer nuestra conexion WIFI a traves del ESP 8266
  void setupWiFi()
  {
    Serial1.println("AT+RST");                    // Reseteamos el modulo
    Timeout(3000);
    delay(500);
    Serial1.println("AT+CWLAP");                  // Listamos las redes disponibles para nuestro modulo
    Timeout(5000);
    Serial1.print("AT+CWJAP=\"");                 // Nos unimos a la red que queremos
    Serial1.print(SSID);
    Serial1.print("\",\"");
    Serial1.print(PASS);
    Serial1.println("\"");
    Timeout(20000);
  }
```

```
//Funcion para establecer nuestro robot como servidor para el modo uno a uno
void setupTCP()
{
  Serial1.println("AT+CWMODE=1");                    // Indicamos que queremos el modo 1, direccion estatica
  Timeout(1000);
  Serial1.println("AT+CWMODE?");                    // Indicamos que queremos el modo 1, direccion estatica
  Timeout(1000);
  Serial1.println("AT+CIPMUX=1");                   // Establecemos que queremos una conexion unica
  Timeout(1000);
  Serial1.print("AT+CIPSERVER=1,");                  // Establecemos el servidor TCP-IP
  Serial1.println(PORT);
  Timeout(1000);
}

//Funcion para el envio de datos desde nuestro Teensy hasta nuestra sesion de MAtlab
void DataSend(int data_data_send[], int largo_data_send)
{
  message="";
  digitalWrite(13,0);
  delayMicroseconds(100);
  for (int i = 0; i < largo_data_send; i++)          // Colocamos cada byte para el envio
  {
    message.concat(data_data_send[i]);                // Creamos nuestro String de envio con nuestros datos
```

```
            message.concat(",");
          }
          lengo_data_send = message.length();                    // Dictamos la cantidad de
bytes a enviar
          Serial1.print("AT+CIPSEND=0,");                        // Enviamos nuestra String de
datos
          Serial1.println(lengo_data_send);
          Timeout(50);
          Serial1.println(message);
          Timeout(50);
          //Serial1.println("AT+CIPCLOSE=0");                     // Cerramos la comunicacion
TCP-IP
          //Timeout(50);
        }

        /*
        ****************************************************************
                        Funciones - Falda de sensores
        ****************************************************************
        */

        //Funcion que permite la toma de datos de nuestra falda de sensores ultrasonicos
        void codigoUltrasonico()
        {
          if (flagCambioCodigoUltrasonico){
            flagCambioCodigoUltrasonico = false;

            if (SENSOR_HCSR04[index_sensor_hcsr04]==0){
                if (flagFallingEcho)
```

```
      {flagFallingEcho=false;
      duration_0 = t_actual-t_anterior;
      if (duration_0>TIME_OUT_PULSE){
        duration_0 = TIME_OUT_PULSE;}}
    else
     {duration_0 = TIME_OUT_PULSE;}
    distance0 = (duration_0/60) ;
    digitalWrite(TRIG_PIN_0, HIGH);
    digitalWrite(TRIG_PIN_1, LOW);
    digitalWrite(TRIG_PIN_2, LOW);
  }
else if (SENSOR_HCSR04[index_sensor_hcsr04]==1){
    if (flagFallingEcho){
      flagFallingEcho=false;
      duration_1 = t_actual-t_anterior;
      if (duration_1>TIME_OUT_PULSE){
        duration_1 = TIME_OUT_PULSE;}}
    else
     {duration_1 = TIME_OUT_PULSE;}
    distance1 = (duration_1/60) ;
    digitalWrite(TRIG_PIN_0, LOW);
    digitalWrite(TRIG_PIN_1, HIGH);
    digitalWrite(TRIG_PIN_2, LOW);


    //flagIMU = imu.getMotion6Counts(&ax, &ay, &az, &gx, &gy, &gz);
  }
else if (SENSOR_HCSR04[index_sensor_hcsr04]==2){
```

```
    if (flagFallingEcho){
      flagFallingEcho=false;
      duration_2 = t_actual-t_anterior;
      if (duration_2>TIME_OUT_PULSE){
        duration_2 = TIME_OUT_PULSE;}}
    else
     {duration_2 = TIME_OUT_PULSE;}
    distance2 = (duration_2/60) ;
    digitalWrite(TRIG_PIN_0, HIGH);
    digitalWrite(TRIG_PIN_1, HIGH);
    digitalWrite(TRIG_PIN_2, LOW);

  }
else if (SENSOR_HCSR04[index_sensor_hcsr04]==3){
    if (flagFallingEcho)
      {flagFallingEcho=false;
      duration_3 = t_actual-t_anterior;
      if (duration_3>TIME_OUT_PULSE){
        duration_3 = TIME_OUT_PULSE;}}
    else
     {duration_3 = TIME_OUT_PULSE;}
    distance3 = (duration_3/60) ;
    digitalWrite(TRIG_PIN_0, LOW);
    digitalWrite(TRIG_PIN_1, LOW);
    digitalWrite(TRIG_PIN_2, HIGH);


    //flagIMU = imu.getMotion6Counts(&ax, &ay, &az, &gx, &gy, &gz);
```

```
  }
else if (SENSOR_HCSR04[index_sensor_hcsr04]==4){
    if (flagFallingEcho)
      {flagFallingEcho=false;
      duration_4 = t_actual-t_anterior;
      if (duration_4>TIME_OUT_PULSE){
        duration_4 = TIME_OUT_PULSE;}}
    else
     {duration_4 = TIME_OUT_PULSE;}
    distance4 = (duration_4/60) ;
    digitalWrite(TRIG_PIN_0, HIGH);
    digitalWrite(TRIG_PIN_1, LOW);
    digitalWrite(TRIG_PIN_2, HIGH);
  }
else if (SENSOR_HCSR04[index_sensor_hcsr04]==5){
    if (flagFallingEcho)
      {flagFallingEcho=false;
      duration_5 = t_actual-t_anterior;
      if (duration_5>TIME_OUT_PULSE){
        duration_5 = TIME_OUT_PULSE;}}
    else
     {duration_5 = TIME_OUT_PULSE;}
    distance5 = (duration_5/60) ;
    digitalWrite(TRIG_PIN_0, LOW);
    digitalWrite(TRIG_PIN_1, LOW);
    digitalWrite(TRIG_PIN_2, LOW);

    //flagIMU = imu.getMotion6Counts(&ax, &ay, &az, &gx, &gy, &gz);
```

```
            }


        }
         noInterrupts();
         data[0] = distance0;                                    // Asignamos nuestros datos al array
para envio
         data[1] = distance1;
         data[2] = distance2;
         data[3] = distance3;
         data[4] = distance4;
         data[5] = distance5;
         data[6] = duracion1;
         data[7] = duracion2;
         interrupts();
         }

        void changeEchoIsr(){
          if (digitalRead(ECHO_PIN_0)==HIGH){//rising
             t_anterior=micros();
           }
          else{
             flagFallingEcho=true;
             t_actual=micros();
           }
         }

        void timeOutEchoIsr(){
         index_sensor_hcsr04=index_sensor_hcsr04+1;
```

```
 if (index_sensor_hcsr04>5)
 {index_sensor_hcsr04=0;}
 flagCambioCodigoUltrasonico = true;
 contador_desbordes_to_hcsr04 = contador_desbordes_to_hcsr04+1;
 if (contador_desbordes_to_hcsr04>=DESBORDES_TO_HCSR04){
    contador_desbordes_to_hcsr04=0;
    //flagDataSend=true;
  }

 }
/*
************************************************************
                Funciones - Motores
************************************************************
*/

//Funcion para dictar direccion y velocidad a nuestros motores.
/*void drivers(int direccion, int p1, int p2)
{
 switch (direccion) {
  case 1 :                                          //"adelante L"
    //analogWrite(driver_1B,p1);
    //analogWrite(driver_1A,255);
    //analogWrite(driver_2A,p2);
    //analogWrite(driver_2B, 255);
    analogWrite(driver_1B,p1);
    analogWrite(driver_1A,255);
    break;
```

```
  case 2 :                                  //"atras L"
   //analogWrite(driver_1B,255);
   //analogWrite(driver_1A,p1);
   //analogWrite(driver_2A,255);
   //analogWrite(driver_2B, p2);
   analogWrite(driver_1B,255);
   analogWrite(driver_1A,p1);
  break;
  case 3 :                                  //"Adelante R"
   //analogWrite(driver_1B,p1);
   //analogWrite(driver_1A,255);
   //analogWrite(driver_2A,255);
   //analogWrite(driver_2B, p2);
   analogWrite(driver_2A,p1);
   analogWrite(driver_2B, 255);

  break;
  case 4:                                   //"Atras R"
   //analogWrite(driver_1B,255);
   //analogWrite(driver_1A,p1);
   //analogWrite(driver_2A,p2);
   //analogWrite(driver_2B, 255);
   analogWrite(driver_2A, 255);
   analogWrite(driver_2B, p1);
  break;

 }
}*/
```

```
void drivers(bool dL, int p1, bool dR, int p2)
{
  //noInterrupts();
  if (dL){
     analogWrite(driver_1B,p1);
     analogWrite(driver_1A,255);
    }
  else{
     analogWrite(driver_1B,255);
     analogWrite(driver_1A,p1);
    }
  if (dR){
     analogWrite(driver_2A,p2);
     analogWrite(driver_2B, 255);
    }
  else{
     analogWrite(driver_2A, 255);
     analogWrite(driver_2B, p2);
    }
  //interrupts();
  }


//Funcion para la asignacion de los valores de nuestros encoders en el array para envio
void countreset()
{
    data[6] = duracion1;
```

```
    data[7] = duracion2;
    //duracion1 = 0;
    //duracion2 = 0;
    //flagDriverMotores = true;
}

//Funciones de manejo de la velocidad de una de las ruedas
void wheelSpeed2()
{
  //flagE2 = true;
  Lstate2 = digitalRead(encoder1_pinA);
  if((encoder0_pinALast_2 == LOW) && Lstate2==HIGH)
  {
    val2 = digitalRead(encoder1_pinB);
    if(val2 == LOW && Direction2)
    {
      Direction2 = false; //Reverse
    }
    else if(val2 == HIGH && !Direction2)
    {
      Direction2 = true;  //Forward
    }
  }
  encoder0_pinALast_2 = Lstate2;
  //Direction1 = false;
  if(!Direction2)  duracion2 ++;
  else  duracion2 --;
}
```

```
void wheelSpeed1()
{
  //flagE1 = true;
  Lstate1 = digitalRead(encoder0_pinA);
  if((encoder0_pinALast_1 == LOW) && Lstate1==HIGH)
  {
    val1 = digitalRead(encoder0_pinB);
    if(val1 == LOW && Direction1)
    {
      Direction1 = false;                                   //Reverse
    }
    else if(val1 == HIGH && !Direction1)
    {
      Direction1 = true;                                    //Forward
    }
  }
  encoder0_pinALast_1 = Lstate1;
  if(!Direction1)  duracion1 ++;
  else  duracion1--;
}


void wheelSpeed1C(){
  noInterrupts();
  Lstate1 = digitalRead(encoder0_pinA);
  if((encoder0_pinALast_1 == LOW) && Lstate1==HIGH)
```

```
  {
    val1 = digitalRead(encoder0_pinB);
    if(val1 == LOW && Direction1)
    {
      Direction1 = false;                                   //Reverse
    }
    else if(val1 == HIGH && !Direction1)
    {
      Direction1 = true;                                    //Forward
    }
  }
  encoder0_pinALast_1 = Lstate1;
  if(!Direction1)  duracion1 ++;
  else  duracion1--;
  interrupts();
}

void wheelSpeed2C()
{
  noInterrupts();
  Lstate2 = digitalRead(encoder1_pinA);
  if((encoder0_pinALast_2 == LOW) && Lstate2==HIGH)
  {
    val2 = digitalRead(encoder1_pinB);
    if(val2 == LOW && Direction2)
    {
      Direction2 = false; //Reverse
    }
```

```
    else if(val2 == HIGH && !Direction2)
    {
      Direction2 = true;  //Forward
    }
  }
  encoder0_pinALast_2 = Lstate2;
  //Direction1 = false;
  if(!Direction2)  duracion2 ++;
  else  duracion2 --;

  interrupts();
}



/*
*****************************************************************
                    Setup
*****************************************************************
*/

void setup()
{
  //El ESP8266 se comunica a 115200 a traves del serial.
  Serial1.begin(115200);  // Serial 1 del Teensy
  //Serial.begin(9600);    // Comunicacion con la computadora
  pinMode(13,OUTPUT);
  digitalWrite(13,0);
```

```
//Esperamos un momento e inicializamos nuestro ESP8266
delay(1000);

//Serial.println("begin.");
setupWiFi();
setupTCP();

// Mostramos la direccion IP asignada
//Serial.print("device ip addr: ");
Serial1.println("AT+CIFSR");
Timeout(5000);

//Definimos los pines de nuestra falda de sensores
pinMode(TRIG_PIN_0, OUTPUT);
pinMode(TRIG_PIN_1, OUTPUT);
pinMode(TRIG_PIN_2, OUTPUT);
pinMode(ECHO_PIN_0, INPUT);

//falda de sensores
duration_0=0;
duration_1=0;
duration_2=0;
duration_3=0;
duration_4=0;
duration_5=0;

distance0=0;
distance1=0;
```

```
distance2=0;
distance3=0;
distance4=0;
distance5=0;

t_actual=0;
t_anterior=0;
flagFallingEcho = false;
flagSerialSend = false;
flagCambioCodigoUltrasonico = false;
flagIMU=false;
index_sensor_hcsr04=0;

digitalWrite(TRIG_PIN_0, HIGH);
digitalWrite(TRIG_PIN_1, HIGH);
digitalWrite(TRIG_PIN_2, HIGH);
delayMicroseconds(20);
attachInterrupt(ECHO_PIN_0, changeEchoIsr, CHANGE);
echoTimeOut.begin(timeOutEchoIsr, TIME_OUT_PULSE); //timer para la espera maxima de cada pulso

//recepcion serial
flagRecepcionSerial=false;

//PWM
 analogWriteFrequency(driver_1A , fr);
 analogWriteFrequency(driver_1B , fr);
 analogWriteFrequency(driver_2A , fr);
```

```
 analogWriteFrequency(driver_2B , fr);

 //Encoder
 timerEncoder.begin(countreset, 1000000);
 //Timer1.initialize(1000000);
 //Timer1.attachInterrupt(countreset);
 Direction1 = true;//default -> Forward
 Direction2 = true;
 pinMode(encoder0_pinA,INPUT);
 pinMode(encoder0_pinB,INPUT);
 pinMode(encoder1_pinA,INPUT);
 pinMode(encoder1_pinB,INPUT);

 attachInterrupt(encoder0_pinA, wheelSpeed1, CHANGE);
 attachInterrupt(encoder1_pinA, wheelSpeed2, CHANGE);

 //valores menores significan prioridad mayor
 NVIC_SET_PRIORITY(IRQ_PORTD, 32); //el pin 7 de echo esta en puertoD, 2
 NVIC_SET_PRIORITY(IRQ_PORTC, 48); //los pines de lectura de encoder estan en puerto C, 1 2 3 y 4.

 drivers(1,255,1,255); //se inicializan los motores
 //digitalWrite(13,1);
}

/*
*****************************************************************
            Loop
```

```
**************************************************************
*/
void loop() {
  digitalWrite(13,1);

  /****Toma las medidas de la falda de sensores****/
  codigoUltrasonico();

  /*****Funciones de encoders*****/
  if (flagE1 == true){
   // flagE1 = false;
   // wheelSpeed1C();
  }

  if (flagE2 == true){
    //flagE2 = false;
    //wheelSpeed2C();
  }

  /*****Recepcion de datos*****/
  recepcionSerial();

  /*****Manda la orden a los motores****/
  if (flagDriverMotores){
    flagDriverMotores=false;
    drivers(dirL,velL,dirR,velR);
  }
```

```
  /*****Envio de datos*****/
  if (flagDataSend){
    flagDataSend=false;
    largo = sizeof(data);
    largo = largo/sizeof(data[0]);
    DataSend(data, largo);
  }
}




/*****************************************************************
 * Interrupcion por entrada de comunicacion con el modulo WI-FI
*****************************************************************/
void serialEvent1(){
  flagRecepcionSerial=true;
}



//funcion que recibe la informacion del puerto serial
void recepcionSerial(){
  if (flagRecepcionSerial){
    flagRecepcionSerial=false;
    digitalWrite(13,0);
    finished = -1;
    tPrev_serialAvailable=millis();
```

```
            while (Serial1.available())                    // Tomamos todos los datos de entrada
            {
              character = (char)Serial1.read();
              input += character;
              finished = input.lastIndexOf("CLOSED");
              t_serialAvailable=millis()-tPrev_serialAvailable;
              /*if(t_serialAvailable>TIME_OUT_SERIAL_AVAILABLE){
                 //finished=-1;
                 Serial1.flush();
                 Serial1.clear();
                 buffer[0]='\0';
                 break;
               }*/
              //if (finished>-1){
               //break;
               //}
            }
            if(finished > -1)                              // Tomamos solamente el envio exitoso
de datos
            {
              //flagDataSend=true;

              options[0] = input.lastIndexOf("CONNECT");          // Revisamos la instruccion
enviada
              options[1] = input.lastIndexOf("Motores");
              max_v=-1;
              max_i = 0;
              for ( int i = 0; i < sizeof(options)/sizeof(options[0]); i++ )
```

```
{
  if ( options[i] > max_v )
  {
    max_v = options[i];
    max_i = i;
  }
  if (i>100){
      max_i = 2;
      break;
    }
}

switch (max_i)
{
  case 0 :                                    // Pedida de datos de la falda de sensores
    //largo = sizeof(data);
    //largo = largo/sizeof(data[0]);
    //DataSend(data, largo);
    flagDataSend=true;
    break;
  case 1 :                                     // Asignamos valores a nuestros motores
    noInterrupts();
    dirL     =  (input.charAt(options[1] + 7) - 48);
    velL  =  (input.charAt(options[1] + 8) - 48)*100;
    velL  += (input.charAt(options[1] + 9) - 48)*10;
    velL  += (input.charAt(options[1] + 10) - 48);

    dirR     =  (input.charAt(options[1] + 11) - 48);
```

```
        velR  =  (input.charAt(options[1] + 12) - 48)*100;
        velR  += (input.charAt(options[1] + 13) - 48)*10;
        velR  += (input.charAt(options[1] + 14) - 48);
        interrupts();



        //drivers(Order,movement,movement1);
        flagDriverMotores = true;
        flagDataSend=true;


        break;
      case 2:
        flagDataSend=true;
        break;
     }
    }


    //Serial1.flush();
    Serial1.clear();
    }
   }
```

Programa creado para la librería de Matlab, encargado de transmitir datos al microcontrolador.

```
function valor = TCPTX(Message,IP,TO,port)
%Funcion que se encarga de enviar un mensahe al swarm robot
%Recibe la ip del robot, su puerto y el timeout esperado
   %t = tcpclient(IP,port,'Timeout',TO);
```

```
    data = uint8(Message);


    write(t,data);
end
```

Programa creado para la librería de Matlab, encargado de recibir datos del microcontrolador.

```
function valor = TCPRX(IP,TO,port)
%Funcion que se encarga de recibir los valores del swarm robot
%Recibe la ip del robot, su puerto y el timeout esperado
    flow = 1;
    time = 10000;
    t = tcpclient(IP,port,'Timeout',TO);
    while (flow > 0)
        if(t.BytesAvailable == 0)
            flow = 1;
            time = time -1;
            if (time ==0)
                flow = 0;
                valor = 99999;
            end
        else
            valor = read(t);
            valor = char(valor);
            flow = 0;
        end
    end
end
```

Programa creado para la librería de Matlab, encargado de recolectar la información de la falda de sensores del robot y convertir los datos a información procesable en Matlab.

```
function valores = SensorCheck(IP,TO,port)
%Funcion para obtener los valores de los 6 sensores ultrasonicos
%Regresa una matriz de 6 valores

   valores = zeros(1,6);
   for i = 1:6
      val = TCPRX(IP,TO,port);
      x = str2num(val);
      valores(1,i) = x;
      pause(0.001);
   end
end
```

Programa creado para la librería de Matlab, encargado de controlar los motores del robot.

```
function Message = MotorControl(DirL, VelL, DirR, VelR)
%Funcion encargada de convertir instrucciones y valores en un mensaje para
%el control de los motores de nuestro robot
%Opciones para ordenes:
%FWD (1): Movimiento hacia el frente
%BCK (0): Movimiento en retroceso

%nargin se refiere al numero de parametros de entrada en la funcion
if nargin == 2
   DirL = 1;
   VelL = 255;
```

```
    DirR = 1;
    VelR = 255;
end

if nargin == 1
    DirL = 1;
    VelL = 255;
    DirR = 1;
    VelR = 255;
end
if nargin == 0
    DirL = 1;
    VelL = 255;
    DirR = 1;
    VelR = 255;
end

Message = ['Motores',int2str(DirL),int2str(VelL),int2str(DirR),int2str(VelR)];

End
```

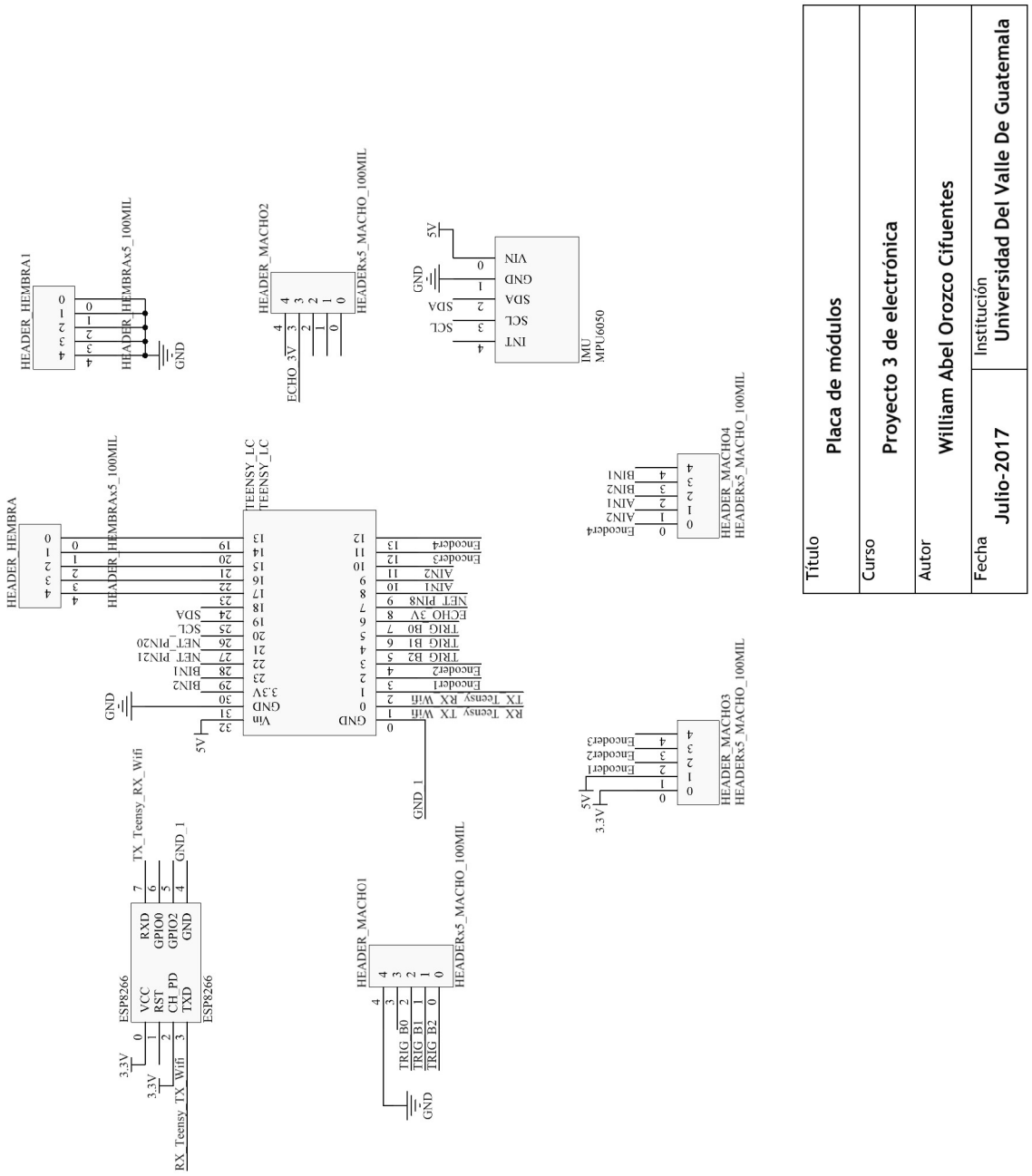
Programa creado para la librería de Matlab, encargado de proveer los datos relevantes de dirección IP y timeout para cada robot de acuerdo con su identificador.

```
function [IP, TO, Port] = RoboIdentify(Name)
%Funcion encargada de cargar las variables de IP, TO y Port a utilizar
%de acuerdo al robot elegido.

switch Name
```

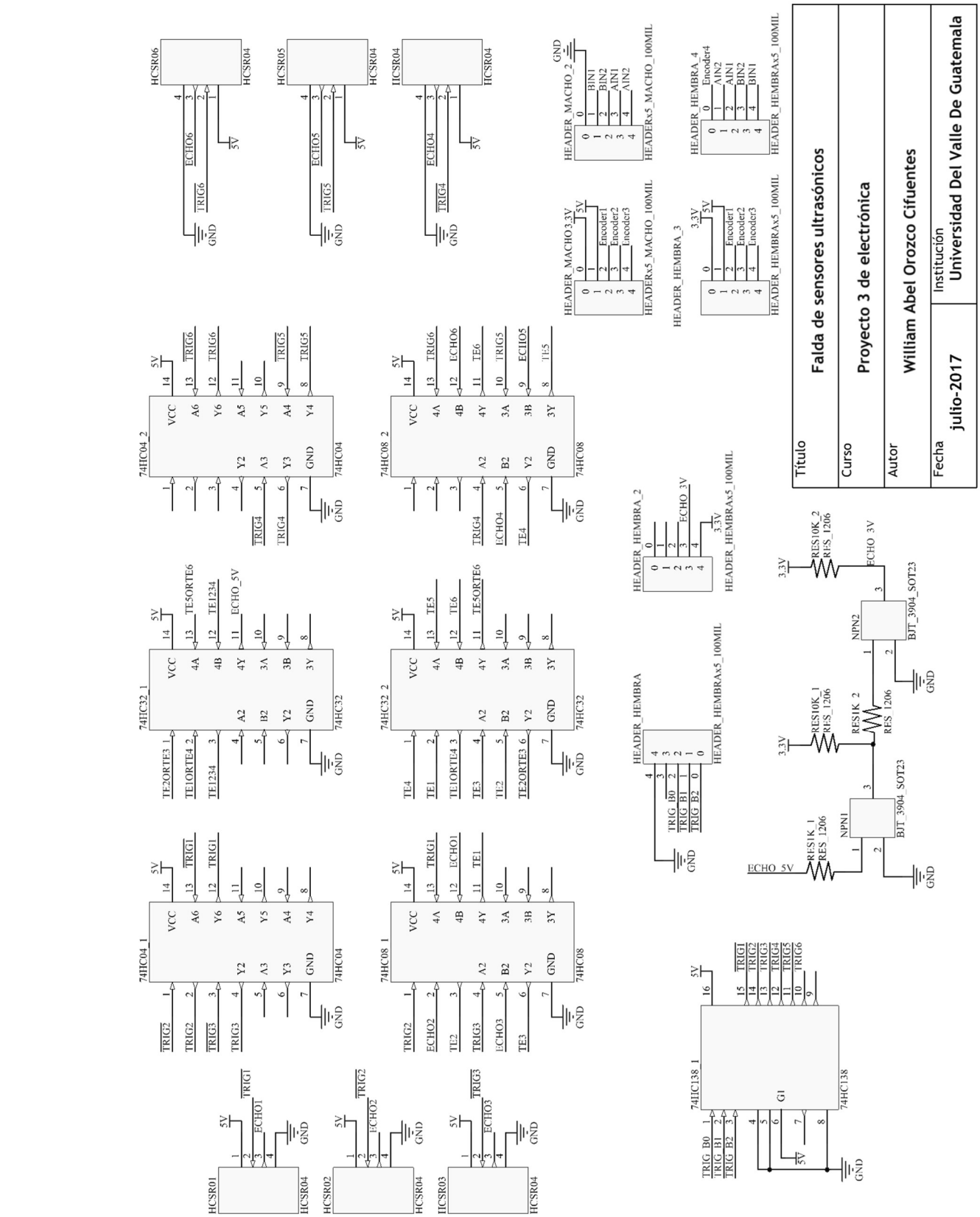
```
    case 'Robot 1'
        IP = '192.168.0.186';
        TO = 100;
        %Port = 20;
        Port = 8080;
    case 'Robot 2'
        IP = '192.168.1.105';
        TO = 100;
        %Port = 20;
        Port = 8080;
    otherwise
        disp('Nombre Incorrecto');
end
end
```

b. Esquemático de la placa de módulos



Título	Placa de módulos
Curso	Proyecto 3 de electrónica
Autor	William Abel Orozco Cifuentes
Fecha	Julio-2017
Institución	Universidad Del Valle De Guatemala

c. Esquemático de la falda de sensores



d. Código de Arduino para la IMU y la falda de sensores

```
#Referencia: https://github.com/simondlevy/MPU6050
#La librería MPU6050.h se obtiene de este repositorio.
#include "MPU6050.h"

#define TRIG_PIN_0 6 //pines conectados al decoder
#define TRIG_PIN_1 5
#define TRIG_PIN_2 4
#define ECHO_PIN_0 7 //pin donde se recibe el echo

//5000 uS equivalen a 85cm
//si velocidad del sonido es 340 m/s = 0.034cm/uS
#define TIME_OUT_PULSE 5000 //tiempo máximo de espera
#define TIME_SERIAL_SEND 100000 // espera entre envio de info, en
uS

unsigned int duration0, duration1 , duration2, duration3,
duration4, duration5;
unsigned int distance0, distance1, distance2, distance3, distance4,
distance5;

String dataUltrasonicos;

volatile int contador;
volatile bool flagFallingEcho, flagSerialSend,
flagCambioCodigoUltrasonico, flagIMU;
volatile int t_actual, t_anterior;

int16_t ax, ay, az, gx, gy, gz;
int x_acc, gyro;

IntervalTimer echoTimeOut, serialTimer; //objetos timer
MPU6050 imu;

void setup() {
  Serial.begin(9600);
  pinMode(TRIG_PIN_0, OUTPUT);
  pinMode(TRIG_PIN_1, OUTPUT);
  pinMode(TRIG_PIN_2, OUTPUT);
  pinMode(ECHO_PIN_0, INPUT);

  duration0=0;
  duration1=0;
  duration2=0;
  duration3=0;
  duration4=0;
  duration5=0;

  distance0=0;
  distance1=0;
  distance2=0;
  distance3=0;
  distance4=0;
  distance5=0;
```

```
  //el pin 7 de echo esta en puertoD (2)
  NVIC_SET_PRIORITY(IRQ_PORTD, 32);

  t_actual=0;
  t_anterior=0;
  flagFallingEcho = false;
  flagSerialSend = false;
  flagCambioCodigoUltrasonico = false;
  flagIMU=false;

  contador=0;

  digitalWrite(TRIG_PIN_0, HIGH);
  digitalWrite(TRIG_PIN_1, HIGH);
  digitalWrite(TRIG_PIN_2, HIGH);
  delayMicroseconds(20);
  attachInterrupt(ECHO_PIN_0, change_echo_isr, CHANGE);

  //timer para la espera maxima de cada pulso
  echoTimeOut.begin(timeOut_echo_isr, TIME_OUT_PULSE);
  //timer para envio de data via serial
  serialTimer.begin(serialSend_isr, TIME_SERIAL_SEND);

  Wire.begin(); //inicio de comunicación i2C

  //inicializacion de la IMU
  if (!imu.begin(AFS_2G, GFS_250DPS)) {
        Serial.println("MPU6050 is online...");
  }
  else {
        Serial.println("Failed to init MPU6050");
        //while (true);
  }
}

void loop(){
  codigo_ultrasonico(); //rutina de ultrasonicos
  serialSend_IMU();     //obtener datos de la IMU
}

//interrupcion del pin de echo
void change_echo_isr(){
    if (digitalRead(ECHO_PIN_0)==HIGH){//rising
        t_anterior=micros();
      }
    else{
        flagFallingEcho=true;
        t_actual=micros();
      }
  }

//timeout si no se detecta un objeto cercano
void timeOut_echo_isr(){
  contador=contador+1;
  if (contador>5)
  {contador=0;}
  flagCambioCodigoUltrasonico = true;
```

```
  }
//envio de información
void serialSend_IMU(){
  if (flagSerialSend && flagIMU){
    //Escalamiento de mediciones. Se hace en Matlab.
    //ax = ax/16384*10;
    //ay = ay/16384*10;
    //az = az/16384*10;
    //gx = gx/131;
    //gy = gy/131;
    //gz = gz/131;

    Serial.print(ax);
    Serial.print(",");
    Serial.print(ay);
    Serial.print(",");
    Serial.print(az);
    Serial.print(",");
    Serial.print(gx);
    Serial.print(",");
    Serial.print(gy);
    Serial.print(",");
    Serial.print(gz);
    Serial.print(",");

    dataUltrasonicos = String(distance0, DEC);
    dataUltrasonicos = dataUltrasonicos + ",";
    dataUltrasonicos = dataUltrasonicos + String(distance1, DEC);
    dataUltrasonicos = dataUltrasonicos + ",";
    dataUltrasonicos = dataUltrasonicos + String(distance2, DEC);
    dataUltrasonicos = dataUltrasonicos + ",";
    dataUltrasonicos = dataUltrasonicos + String(distance3, DEC);
    dataUltrasonicos = dataUltrasonicos + ",";
    dataUltrasonicos = dataUltrasonicos + String(distance4, DEC);
    dataUltrasonicos = dataUltrasonicos + ",";
    dataUltrasonicos = dataUltrasonicos + String(distance5, DEC);

    Serial.println(dataUltrasonicos);

    flagSerialSend = false;
    flagIMU=false;
  }
}

//se puede enviar tiempo entre pulsos de echo, o distancia
procesando la duracion del pulso
void codigo_ultrasonico() { //ALERTA: cuidado con las
combinaciones, deben coincidir con el cableado.
 if (flagCambioCodigoUltrasonico){
   flagCambioCodigoUltrasonico = false;

   if (contador==0){
        if (flagFallingEcho)
          {flagFallingEcho=false;
          duration0 = t_actual-t_anterior;
          if (duration0>TIME_OUT_PULSE){
            duration0 = TIME_OUT_PULSE;}}
```

```
            else
             {duration0 = TIME_OUT_PULSE;}
            distance0 = (duration0/60) ;
            digitalWrite(TRIG_PIN_0, HIGH);
            digitalWrite(TRIG_PIN_1, LOW);
            digitalWrite(TRIG_PIN_2, LOW);
          }
        else if (contador==1){
            if (flagFallingEcho){
              flagFallingEcho=false;
              duration1 = t_actual-t_anterior;
              if (duration1>TIME_OUT_PULSE){
                duration1 = TIME_OUT_PULSE;}}
            else
             {duration1 = TIME_OUT_PULSE;}
            distance1 = (duration1/60) ;
            digitalWrite(TRIG_PIN_0, LOW);
            digitalWrite(TRIG_PIN_1, HIGH);
            digitalWrite(TRIG_PIN_2, LOW);

       //tomar datos de la imu
            flagIMU = imu.getMotion6Counts(&ax, &ay, &az, &gx, &gy,
&gz);
          }
        else if (contador==2){
            if (flagFallingEcho){
              flagFallingEcho=false;
              duration2 = t_actual-t_anterior;
              if (duration2>TIME_OUT_PULSE){
                duration2 = TIME_OUT_PULSE;}}
            else
             {duration2 = TIME_OUT_PULSE;}
            distance2 = (duration2/60) ;
            digitalWrite(TRIG_PIN_0, HIGH);
            digitalWrite(TRIG_PIN_1, HIGH);
            digitalWrite(TRIG_PIN_2, LOW);

          }
        else if (contador==3){
            if (flagFallingEcho)
              {flagFallingEcho=false;
              duration3 = t_actual-t_anterior;
              if (duration3>TIME_OUT_PULSE){
                duration3 = TIME_OUT_PULSE;}}
            else
             {duration3 = TIME_OUT_PULSE;}
            distance3 = (duration3/60) ;
            digitalWrite(TRIG_PIN_0, LOW);
            digitalWrite(TRIG_PIN_1, LOW);
            digitalWrite(TRIG_PIN_2, HIGH);

            //tomar datos de la imu
            flagIMU = imu.getMotion6Counts(&ax, &ay, &az, &gx, &gy,
&gz);
          }
        else if (contador==4){
            if (flagFallingEcho)
```

```
                {flagFallingEcho=false;
                duration4 = t_actual-t_anterior;
                if (duration4>TIME_OUT_PULSE){
                  duration4 = TIME_OUT_PULSE;}}
              else
               {duration4 = TIME_OUT_PULSE;}
              distance4 = (duration4/60) ;
              digitalWrite(TRIG_PIN_0, HIGH);
              digitalWrite(TRIG_PIN_1, LOW);
              digitalWrite(TRIG_PIN_2, HIGH);
            }
          else if (contador==5){
              if (flagFallingEcho)
                {flagFallingEcho=false;
                duration5 = t_actual-t_anterior;
                if (duration5>TIME_OUT_PULSE){
                  duration5 = TIME_OUT_PULSE;}}
              else
               {duration5 = TIME_OUT_PULSE;}
              distance5 = (duration5/60) ;
              digitalWrite(TRIG_PIN_0, LOW);
              digitalWrite(TRIG_PIN_1, LOW);
              digitalWrite(TRIG_PIN_2, LOW);

              //tomar datos de la imu
              flagIMU = imu.getMotion6Counts(&ax, &ay, &az, &gx, &gy,
&gz);
            }
       }
      }
```

e. Código de Matlab para ilustrar los resultados de la IMU

```
delete(instrfindall)

%Referencias
%https://electronics.stackexchange.com/questions/142037/calculating-angles-from-mpu6050
%https://robologs.net/2014/11/04/arduino-y-matlab-v-leer-una-imu-por-serial/
%Valores de conversion
A_R = 16384.0;
G_R = 131.0;

%Conversion de radianes a grados 180/PI
RAD_TO_DEG = 57.295779;

Acc=zeros(1,3);
Gy=zeros(1,3);
Angle=zeros(1,3);

arduino = serial('COM7')
fopen(arduino)
Datos = zeros(1,6);

v = VideoWriter('imu.avi');
open(v);
axis tight manual
set(gca,'nextplot','replacechildren');


    for k = 1:500
        valor = fscanf(arduino);
        valor = char(valor);
        Data = strsplit(valor,',');
        Data(length(Data)) = [];

        for i = 1:length(Datos)
            x = Data(i);
            Datos(1,i) = str2double(x);
        end

        AcX=Datos(1);
        AcY=Datos(2);
        AcZ=Datos(3);

        roll=Datos(4);
        pitch=Datos(5);
        yaw=Datos(6);

        %Escalar datos de aceleracion
        AcX=AcX/A_R;
        AcY=AcY/A_R;
        AcZ=AcZ/A_R;

        %angulo Z
```

```
            Acc(3)          =          atan(sqrt(power(AcY,2)          +
power(AcX,2))/(AcZ))*RAD_TO_DEG;
            %angulo Y
            Acc(2)       =       atan(-1*AcX/sqrt(power(AcY,2)       +
power(AcZ,2)))*RAD_TO_DEG;
            %angulo X
            Acc(1)        =        atan(AcY/sqrt(power(AcX,2)        +
power(AcZ,2)))*RAD_TO_DEG;

            %Escalar datos de giro
            %giro X
            Gy(1) = roll/G_R;
            %giro Y
            Gy(2) = pitch/G_R;
            %giro Z
            Gy(3) = yaw/G_R;

            %filtro para neutralizar el ruido en las mediciones
            Angle(3) = 0.95 *(Angle(3)+Gy(3)*0.010) + 0.05*Acc(3);
            Angle(1) = 0.95 *(Angle(1)+Gy(1)*0.010) + 0.05*Acc(1);
            Angle(2) = 0.95 *(Angle(2)+Gy(2)*0.010) + 0.05*Acc(2);

            IMU_plotCube(Angle(1), Angle(2), Angle(3))
            drawnow

            frame = getframe(gcf);
            writeVideo(v,frame);
            clf

        end

    close(v);
    fclose(arduino)
```

f. Función complementaria para ilustrar los resultados de la IMU

```
function IMU_plotCube(roll, pitch, yaw)
 %Referencia:
 %https://robologs.net/2014/11/04/arduino-y-matlab-v-leer-una-
imu-por-serial/
 vertex_matrix = [0 0 0
1 0 0
1 1 0
0 1 0
0 0 1
1 0 1
1 1 1
0 1 1]-0.5;

faces_matrix = [1 2 6 5
2 3 7 6
3 4 8 7
4 1 5 8
1 2 3 4
5 6 7 8];

subplot(1,3,1)
axis([-1 1 -1 1 -1 1]);
axis equal off;
cube                                                                    =
patch('Vertices',vertex_matrix,'Faces',faces_matrix,'FaceColor',
'green');
rotate(cube, [1,0,0], roll);
rotate(cube,[0,1,0], pitch);
%rotate(cube,[0,0,1], yaw);
view(0,0);

end
```

g. Código de Matlab para implementar el controlador proporcional

```
% Referencia: https://github.com/hjjayakrishnan/go-to-goal
clear all
close all
clc

% Time step
T = 0.3;

% Referencias de posición (GOAL)
%DISCLAIMER: EVITAR QUE xg-x sea 0, porque se usa una tangente
inversa, %que incluye una división entre delta y/delta x.
x_g = 1;
y_g = 1;

% Ganancias del control de velocidad
kP_v = 1; % constante de ganancia proporcional
kI_v = 0; % constante de ganancia integral
kD_v = 0; % constante de ganancia derivativa

%Ganancias del control de ángulo
kP_omega = 10; % constante de ganancia proporcional
kI_omega = 0; % constante de ganancia integral
kD_omega = 0; % constante de ganancia derivativa

% Condición inicial del vector de estado x(t0) = x0
x_0 = 0;
y_0 = 0;
phi_0=0;



%valores para entrar al loop
omega_0 = 0;
v_0 = 0;
NL_prev = 0;
NR_prev = 0;
vL=0;
vR=0;

e_xy_past=0;
E_xy_past=0;
e_phi_past=0;
E_phi_past=0;

% iteraciones
k_max = 100; % número máximo de iteraciones

%valores máximos permitidos
max_v = 60;
min_pwm = 230; %los pwm están 'al revés', así se implementan en
Teensy
max_pwm = 180;
```

```
%Parámetros dimensionales del robot (en metros)
L=0.14; %distancia entre llantas
R=0.02; %radio de la llanta
%cantidad de "ticks" por revolución en la llanta
%Según robotshop, N =48
%Según dfrobot, N=12
%Según pruebas propias, N=24
N=24;



% Inicializar estados
x = x_0; % posición en x
y = y_0; % posición en y
v = v_0; % velocidad traslacional
phi = phi_0; % posición angular
omega = omega_0; % velocidad angular

%parámetros de conexión tcp
IP = '192.168.1.105';
TO = 100;
port = 8080;

%inicialización de la conexión tcp
for i = 1:10
    tcp_obj = tcpclient(IP,port,'Timeout',TO);
    pause(T);
    read(tcp_obj);
end
%un envío más de conexión para obtener datos
tcp_obj = tcpclient(IP,port,'Timeout',TO);
pause(T);
valor=read(tcp_obj);

%Vectores para almacenar x, y, ángulo.
X = [];
Y = [];
Theta = [];

for t = 1:k_max
    %------------------------------------------------------------
------
    %           Sección de escritura y lectura de datos
    %------------------------------------------------------------
------
    tic;
    %No permitir valores mayores o menores a los que los actuadores
    %pueden manejar
    if vL>max_v
        vL=max_v;
    end
    if vL<(-max_v)
        vL=-max_v;
    end
    if vR>max_v
        vR=max_v;
    end
```

```
            if vR<(-max_v)
                vR=-max_v;
            end

            %mapeo a valor pwm de vL
            vL_pwm = (max_pwm - min_pwm)/(max_v)*(abs(vL)-max_v)+max_pwm;
            %mapeo a valor pwm de vR
            vR_pwm = (max_pwm - min_pwm)/(max_v)*(abs(vR)-max_v)+max_pwm;

            %Se forma el mensaje para enviar al Teensy
            if vL>0
                if vR>=0
                    Message                                                 =
['Motores',int2str(1),int2str(abs(vL_pwm)),int2str(1),int2str(vR_pwm)];
                else
                    Message                                                 =
['Motores',int2str(1),int2str(abs(vL_pwm)),int2str(0),int2str(vR_pwm)];
                end
            else
                if vR>=0
                    Message                                                 =
['Motores',int2str(0),int2str(abs(vL_pwm)),int2str(1),int2str(vR_pwm)];
                else
                    Message                                                 =
['Motores',int2str(0),int2str(abs(vL_pwm)),int2str(0),int2str(vR_pwm)];
                end
            end


            %*******escritura de velocidades*********
            data = uint8(Message);  %conversion de datos a uint8
            write(tcp_obj,data);


            %********lectura de sensores********
            flow = 1;
            time = 10000;
            while (flow > 0)
                if(tcp_obj.BytesAvailable == 0)
                    flow = 1;
                    time = time -1;
                    if (time ==0)
                        flow = 0;
                    end
                else
                    valor = read(tcp_obj);
                    valor = char(valor);
                    flow = 0;
                end
            end

            Data = strsplit(valor,',');
            Data(length(Data)) = [];
            Datos = zeros(1,8);
```

```
            for i = 1:8
                x_rx = Data(i);
                Datos(1,i) = str2double(x_rx);
            end


            %Calculo de los D para las llantas
            NL=Datos(7);
            NR=Datos(8);
            DL=2*pi*R*(NL-NL_prev)/N;
            DR=2*pi*R*(NR-NR_prev)/N;
            Dc=(DL+DR)/2;


            %-------------------------------------------------------------
------
            %           Sección del controlador
            %-------------------------------------------------------------
------
            % Se actualizan estados
            x = x + Dc*cos(phi);
            y = y + Dc*sin(phi);
            phi = phi + (DR-DL)/L;

            %Se almacenan los estados para graficarlos después.
            X = [X,x];
            Y = [Y,y];
            Theta = [Theta,phi];

            % Se calcula el error de posición
            e_xy = norm([x_g;y_g] - [x;y],2);
            eD_xy = e_xy-e_xy_past;
            E_xy = E_xy_past+e_xy;
            % Se calcula el valor de v a partir de un PID
            v = kP_v*e_xy + kI_v*E_xy + kD_v*eD_xy;

            e_xy_past = e_xy;
            E_xy_past = E_xy;


            % Se calcula el angulo deseado
            %CUIDADO, USAR atan2() para normalizar a un intervalo entre
            %[-pi,pi]
            switch x_g >= x
                case 1
                    phi_d = atan((y_g - y)/(x_g - x));
                case 0
                    switch y_g >= y
                        case 1
                            phi_d = (pi/2) + atan(abs(y_g - y)/abs(x_g -
x));
                        case 0
                            phi_d = -(pi/2) - atan(abs(y_g - y)/abs(x_g -
x));
                    end
```

```
    end
    phi_d = atan2(sin(phi_d),cos(phi_d));

    % Se calcula el error en el angulo
    e_phi = phi_d - phi;
    %normalizar al intervalo [-pi,pi]
    e_phi = atan2(sin(e_phi),cos(e_phi));


    eD_phi = e_phi-e_phi_past;
    E_phi = E_phi_past+e_phi;

    % Se calcula el valor de velocidad angular a partir de un PID
    omega = kP_omega*e_phi + kI_omega*E_phi + kD_omega*eD_phi;

    e_phi_past = e_phi;
    E_phi_past = E_phi;

    %se calculan las velocidades de cada llanta
    vL=floor((2*v-omega*L)/(2*R));
    vR=floor((2*v+omega*L)/(2*R));

    NL_prev=NL;
    NR_prev=NR;



    %""""""""""""""""""TIEMPO DE MUESTREO""""""""""""""
    elapsed = toc;
    pause(T-elapsed);
end

%Se grafican los resultados
plot (X, 'c-o');
hold on;
plot (Y, 'r')
plot (Theta, 'b-x')
title('Variables de estado')
ylabel('Variable')
xlabel('Muestra')
legend('x','y','phi');
```

Mouser #	Mfr. #	Manufacturer	Customer #	Description	Order Qty.	Price (USD)
595-SN74HC08D	SN74HC08D	Texas Instruments		Logic Gates Quad 2-Input	12	$0.31
595-SN74HC32D	SN74HC32D	Texas Instruments		Logic Gates Quad 2-Input	12	$0.31
595-SN74HC138DR	SN74HC138DR	Texas Instruments		Encoders, Decoders, Multiplexers & Demultiplexers 3-to-8 Line Decoder	10	$0.323
511-L7805CDT-TR	L7805CDT-TR	STMicroelectronics		Linear Voltage Regulators 5.0V 1.0A Positive	6	$0.68
108-1M31T1B1M1QE-EVX	108-1M31T1B1M1QE-EVX	Mountain Switch		Toggle Switches SWITCH TOGGLE 3PDT	4	$4.74
595-SN74HC04D	SN74HC04D	Texas Instruments		Inverters Hex	10	$0.31
863-NCP1117LPST33T3G	NCP1117LPST33T3G	ON Semiconductor		LDO Voltage Regulators LOW DROPOUT REGULATORS	10	$0.332
532-573100D00010G	573100D00010G	Aavid Thermalloy		Heat Sinks SLV HEATSINK TO-252	4	$1.04
485-1769	1769	Adafruit		Adafruit Accessories JST-PH 2-pin SMT Right Angle Connect	10	$0.75
603-RC1206FR-071KL	RC1206FR-071KL	Yageo		Thick Film Resistors - SMD 1K OHM 1%	50	$0.024
603-RC1206FR-0710KL	RC1206FR-0710KL	Yageo		Thick Film Resistors - SMD 10K OHM 1%	50	$0.024
603-RC1206FR-07100RL	RC1206FR-07100RL	Yageo		Thick Film Resistors - SMD 100 OHM 1%	30	$0.024
603-RC1206FR-07330RL	RC1206FR-07330RL	Yageo		Thick Film Resistors - SMD 330 OHM 1%	30	$0.024
80-C1206C104K5R	C1206C104K5RACTU	KEMET		Multilayer Ceramic Capacitors MLCC - SMD/SMT 50volts 0.1uF X7R 10%	10	$0.042
77-VJ1206Y334KXJTBC	VJ1206Y334KXJTW1BC	Vishay		Multilayer Ceramic Capacitors MLCC - SMD/SMT 1206 0.33uF 16volts X7R 10	10	$0.10
77-VJ1206V106ZXQTBC	VJ1206V106ZXQTW1BC	Vishay		Multilayer Ceramic Capacitors MLCC - SMD/SMT 1206 10uF 10volts Y5V +80-	10	$0.10
140-VE101M1CTR0605	VE-101M1CTR-0605	Lelon		Aluminum Electrolytic Capacitors - SMD 16 Volts 100uF 20% 6.3x5.3	10	$0.17
647-UWT0J102MNL1S	UWT0J102MNL1GS	Nichicon		Aluminum Electrolytic Capacitors - SMD 6.3volts 1000uF 8x10 20%	10	$0.238
603-RC1206JR-072KL	RC1206JR-072KL	Yageo		Thick Film Resistors - SMD 2K OHM 5%	10	$0.018
604-APT3216QBC/D	APT3216QBC/D	Kingbright		Standard LEDs - SMD Blue 470nm Water Clear 100mcd	10	$0.213
604-APT3216SECKJ3PRV	APT3216SECK/J3-PRV	Kingbright		Standard LEDs - SMD SMD CHIP LED LAMP 3.2x1.6mm HYPER RED	10	$0.283
863-MMBT3904LT1G	MMBT3904LT1G	ON Semiconductor		Bipolar Transistors - BJT NPN GENERAL PURPOSE	20	$0.098
604-APL3015CGCK-F01	APL3015CGCK-F01	Kingbright		Standard LEDs - SMD GREEN WATER CLEAR	21	$0.154

i. Control de motores

```
void drivers(bool dL, int p1, bool dR, int p2)
{
 //noInterrupts();
 if (dL){
    analogWrite(driver_1B,p1);
    analogWrite(driver_1A,modoM);
  }
 else{
    analogWrite(driver_1B,modoM);
    analogWrite(driver_1A,p1);
  }
 if (dR){
    analogWrite(driver_2A,p2);
    analogWrite(driver_2B, modoM);
  }
 else{
    analogWrite(driver_2A, modoM);
    analogWrite(driver_2B, p2);
  }
 //interrupts();
 }
```

j. Lectura de encoders

```
void wheelSpeed1()
{
 //flagE1 = true;
 Lstate1 = digitalRead(encoder0_pinA);
```

```
 if((encoder0_pinALast_1 == LOW) && Lstate1==HIGH)
 {
  val1 = digitalRead(encoder0_pinB);
  if(val1 == LOW && Direction1)
  {
    Direction1 = false;                                  //Reverse
  }
  else if(val1 == HIGH && !Direction1)
  {
    Direction1 = true;                                   //Forward
  }
 }
 encoder0_pinALast_1 = Lstate1;
 if(!Direction1)  duracion1 ++;
 else  duracion1--;
}
```

k. Anexo swarm

B. SILLA

a. Código de la rutina para el giro de los triángulos.

```
//Definicion de puertos
int DIR1=23;
int DIR2=24;
int ENA1=17;
int ENA2=16;
```

```
int BRK1=19;
int BRK2=18;

void setup() {
  //se declaran los puertos como salidas
  pinMode(DIR1, OUTPUT);
  pinMode(DIR2, OUTPUT);
  pinMode(ENA1, OUTPUT);
  pinMode(ENA2, OUTPUT);
  pinMode(BRK1, OUTPUT);
  pinMode(BRK2, OUTPUT);
}

void loop() {
  //los enables se dejan fijos en activos, para dejarles el control a los break
  digitalWrite(ENA1, HIGH);
  digitalWrite(ENA2, HIGH);
  //se llama a la funcion motores, con el parametro control.
  triangulos(control);
}

void triangulos(int control){
  //dependiendo del valor de la variable control, se cambiara la direccion del motor
  //o se frenara el giro del motor
  if(control<490){
    digitalWrite(DIR1, HIGH);
    digitalWrite(DIR2, LOW);
    digitalWrite(BRK1, HIGH);
```

```
  digitalWrite(BRK2, HIGH);
  }
 else if(control>530){
  digitalWrite(DIR1, HIGH);
  digitalWrite(DIR2, LOW);
  digitalWrite(BRK1, HIGH);
  digitalWrite(BRK2, HIGH);
  }
 else{
  digitalWrite(BRK1, LOW);
  digitalWrite(BRK2, LOW);
  }
 }
```

C. ROBOT EXPLORADOR

a. Líneas de código ejecutadas en SSH de Raspberry Pi para la instalación de Mjpg-Streamer:

```
raspi-config
rapistill /home/pi/prueba.jpg
sudo modprobe bcm2835-v4l2
sudo nano /etc/rc.local
ls /dev | grep vid
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install libjpeg8-dev imagemagick libv4l-dev
wget http://terzo.acmesystems.it/download/webcam/mjpg-streamer.tar.gz
tar -xvzf mjpg-streamer.tar.gz
sudo ln -s /usr/include/libv4l1-videodev.h /usr/include/linux/videodev.h
cd mjpg-streamer/
```

sudo nano Makefile

sudo ./mjpg_streamer -i "./input_uvc.so -f 10 -r 640x320 -n -y" -o "./output_http.so –w ./www -p 80" .

D. CÓDIGO DEL CONTROLADOR (ARDUINO)

```
//Universidad del Valle de Guatemala

//Librerías

#include <Adafruit_BNO055.h> //Liberia para FONA GPS

#include <Adafruit_FONA.h>  //Libreria para IMU

#include <Adafruit_Sensor.h> //Libreria para IMU

#include <utility/imumaths.h>

#include <Wire.h> //Libreria para comunicacion i2c

#include <SoftwareSerial.h> //Libreria para comunicacion serial

#include <PWM.h>

#define BNO055_SAMPLERATE_DELAY_MS (100)



//Pines

int velM1 = 11;  //pwm control de velocidad para el motor 1

int velM2 = 6;  //pwm control de velocidad para el motor 2

int dirM1 = 22; //Direccion del motor 1

int dirM2 = 23; //Direcciond el motor 2

int trig1 = 26; //ULTRASONICOS

int trig2 = 28;

int eco1 = 27;

int eco2 = 29;


//Variables
```

```
int banderaF=1;
char replybuffer[255];
long tiempoSU1, tiempoSU2;
float distanciaSU1, distanciaSU2;
//Conexion Serial
SoftwareSerial fonaSS = SoftwareSerial (10, 2); //10 Tx del Fona, 2 RX del Fona
SoftwareSerial *fonaSerial = &fonaSS; //pointer recibe address de fonaSS

//Objetos
Adafruit_FONA fona = Adafruit_FONA(4); //pin de Reset 4
uint8_t readline(char *buff, uint8_t maxbuff, uint16_t timeout = 0);
uint8_t type;
Adafruit_BNO055 bno = Adafruit_BNO055();

void setup() {
 Serial.begin(9600); //iniciar serial UART con 9600
 //Configuracion pines del Motor**********
 pinMode (9,OUTPUT);
 pinMode (6,OUTPUT);
 pinMode (22,OUTPUT);
 pinMode (23,OUTPUT);
 digitalWrite (velM1, LOW);
 digitalWrite (velM2, LOW);

 //Configuracion Ultrasonicos
 pinMode (trig1, OUTPUT);
 pinMode (trig2, OUTPUT);
 pinMode (eco1, INPUT);
```

```
  pinMode (eco2, INPUT);


  //iniciamos BNO055
  bno.begin();
}


void loop() {
  if (Serial.available()>0){ //verificamos si se recibe datos
    char accion = Serial.read(); //tomamos el caracter serial para saber que funcion ejecutar
    //Switch Case
    switch (accion){
      //CASOS MOTORES MOVIMIENTO ----------
      case 'a':{  //Encender ambos motores avanzar hacia adelante
        Serial.begin(9600);
        movimiento(255,255,1,1);
        break;
      }
      case 'r':{  //Ambos motores en retroceso
        movimiento(255,255,0,0);
        break;
      }
      case 'd':{  //Giro derecha
        movimiento(100,100,1,0);
        break;
      }
      case 'I':{  //Giro izquierda
        movimiento(100,100,0,1);
        break;
```

```
}
case 's':{  //Apagar ambos motores
  movimiento(0,0,0,0);
  break;
}
//ULTRASONICOS
case 'u':{  //Ultrasonico frontal
  digitalWrite(trig1, LOW);
  delayMicroseconds(5);
  digitalWrite (trig1,HIGH);
  delayMicroseconds(10);
  digitalWrite (trig1,LOW);
  tiempoSU1 = pulseIn(eco1, HIGH);
  tiempoSU1 = tiempoSU1/2;
  Serial.println (tiempoSU1);
  distanciaSU1 = tiempoSU1*0.0343;
  Serial.print ("Ultrasonico Frontal (cm): ");
  Serial.println (distanciaSU1);
  Serial.end ();
  delay (4000);
  Serial.begin(9600);
  break;
}
case 'U':{ //Ultrasonico trasero
  digitalWrite (trig2, HIGH);
  delayMicroseconds (10);
  digitalWrite (trig2,LOW);
  tiempoSU2 = pulseIn(eco2, HIGH);
```

```
  tiempoSU2 = tiempoSU2/2;
  distanciaSU2 = tiempoSU2*0.0343;
  Serial.print ("Ultrasonico Trasero: ");
  Serial.println (distanciaSU2);
  Serial.end ();
  delay (4000);
  Serial.begin(9600);
  break;
}
//TEMPERATURA
case 'T':{
  int8_t temp = bno.getTemp();
  Serial.print("TEMPERATURA INTERNA: ");
  Serial.print(temp);
  Serial.println(" C");
  Serial.end ();
  delay (3000);
  Serial.begin (9600);
  break;
}
//CASOS FONA GPS ----------------------
case 'b':{  //Muestra porcentaje de bateria de Fona
  if (banderaF ==1){
    Serial.end();
    serialFona ();
    networkStat();
    uint16_t voltajeBat;
    fona.getBattPercent(&voltajeBat);
```

```
    delay (3000);
    Serial.begin(9600);
    Serial.print ("Porcentaje Bateria Fona (%): ");
    Serial.println (voltajeBat);
    Serial.end ();
    delay (4000);
    Serial.begin (9600);
    break;
   }
   else {
    Serial.begin (9600);
    Serial.println ("REALIZE CONEXION A FONA");
    Serial.end ();
    delay (5000);
    Serial.begin (9600);
    break;
   }
   break;
  }
  case 'n':{
   if (banderaF ==1) {
    Serial.end ();
    serialFona ();
    networkStat ();
    delay (9000);
    uint8_t estado = fona.getNetworkStatus ();
    Serial.begin (9600);
    Serial.print ("Estado de Red");
```

```
    Serial.print (" : ");
    if (estado == 1) Serial.println("Registrado");
    if (estado == 2) Serial.println("Buscando");
    Serial.end ();
    delay (4000);
    Serial.begin (9600);
   }
   else {
    Serial.begin(9600);
    Serial.println ("REALIZE CONEXION A FONA");
   }
   break;
 }
 case 'i':{ //Estado RSSI, intensidad de senal de red
   if (banderaF == 1) {
    Serial.end ();
    uint8_t n = fona.getRSSI();
    int8_t r;
    if (n == 0) r = -115;
    if (n == 1) r = -111;
    if (n == 31) r = -52;
    if ((n >= 2) && (n <= 30)) {
      r = map(n, 2, 30, -110, -54);
    }
    Serial.begin (9600);
    Serial.print("Nivel de Intensidad de Señal: ");
    Serial.print(r);
    Serial.println(" dBm");
```

```
  }
  else {
    Serial.begin(9600);
    Serial.println("REALIZE CONEXION");
  }
  break;
}
case 'l':{ //Devuelve Coordenadas GPS
  if (banderaF == 1){
    Serial.end ();
    serialFona ();
    networkStat ();
    delay (11000);
    fona.enableGPRS (true);
    uint16_t returncode;
    if (!fona.getGSMLoc(&returncode, replybuffer, 250)){
      Serial.begin(9600);
      Serial.print(F("Failed!"));
    }
    if (returncode == 0) {
      Serial.begin (9600);
      Serial.print ("Longitud: ");
      for (int i = 0; i < 10; i++){
        Serial.print(replybuffer[i]);
      }
      Serial.print (" Latitud: ");
      for (int i = 11; i <20; i++){
        Serial.print(replybuffer[i]);
```

```
          }
          Serial.println("");
        } else {
          Serial.begin (9600);
          Serial.print(F("Fail code #")); Serial.println(returncode);
        }
      Serial.end ();
      delay (4000);
      Serial.begin (9600);
      }
      else {
        Serial.begin (9600);
        Serial.print ("REALIZAR CONEXION");
      }

      break;
    }
  }
 }
}

//FUNCION REGISTRO DE RED
void networkStat(void){
  uint8_t netW = fona.getNetworkStatus();
}

//FUNCION MOVIMIENTO DE MOTORES
//(255,255,1,1) avanza ; (255,255,0,0) retroceso
```

```
//(100,100,0,1) izquierda; (100,100,1,0) derecha
void movimiento (char v1, char v2, char d1, char d2){
  analogWrite (velM1, v1);  //velocidad de movimiento segun PWM
  digitalWrite (dirM1, d1); //direccion
  analogWrite (velM2, v2);  //255 maxima velocidad
  digitalWrite (dirM2, d2);
}
//FUNCION PARA SERIAL CON FONA
void serialFona (void){
  fonaSerial->begin(4800); //iniciar el serial del fona con 4800
  if (! fona.begin(*fonaSerial)){ //si no esta conectado el fona
    Serial.println(F("No encontrado"));
  }
  else {
    fona.enableGPS (true);
    fona.enableGPRS (true);
    Serial.print("FONA CONECTADO");
  }
}
//readline----------Codigo obtenido del proveedor Adafruit, escrito por Limor Fried/Ladyada para Adafruit.
uint8_t readline(char *buff, uint8_t maxbuff, uint16_t timeout) {
  uint16_t buffidx = 0;
  boolean timeoutvalid = true;
  if (timeout == 0) timeoutvalid = false;

  while (true) {
    if (buffidx > maxbuff) {
```

```
  //Serial.println(F("SPACE"));
  break;
 }

 while (Serial.available()) {
  char c =  Serial.read();

  //Serial.print(c, HEX); Serial.print("#"); Serial.println(c);

  if (c == '\r') continue;
  if (c == 0xA) {
   if (buffidx == 0)   // the first 0x0A is ignored
    continue;

   timeout = 0;         // the second 0x0A is the end of the line
   timeoutvalid = true;
   break;
  }
  buff[buffidx] = c;
  buffidx++;
 }

 if (timeoutvalid && timeout == 0) {
  //Serial.println(F("TIMEOUT"));
  break;
 }
 delay(1);
}
```

```
  buff[buffidx] = 0;  // null term
  return buffidx;
}
```

XV. GLOSARIO

Conductancia: propiedad eléctrica de los materiales que describe la facilidad al paso de corriente eléctrica

Dieléctrico: propiedad de un material a tener una baja conductividad eléctrica, comúnmente se conocen como aislantes.

Elastómero: es un tipo de material compuesto, no metálico, que presenta un comportamiento elástico ante fuerzas de deformación.

Equiángular: término utilizado para describir una igualdad de ángulos entre distintas características geométricas.

Grado de libertad: cantidad de reacciones que se pueden realizar en una estructura, se descompone en 3 rotaciones y 3 traslaciones geométricas.

Mioacústico: propiedad de los músculos a generar señales auditivas durante su movimiento natural.

Mioeléctrica: propiedad eléctrica de los músculos.

Neuroeléctrico: dispositivo que crea una interfaz entre nervios, neuronas o el sistema nervioso.

Permitividad: es una magnitud física que representa la interacción entre un campo eléctrico y su entorno.

PLA: poliácido láctico, es un termoplástico biodegradable y bioactivo derivado de materiales renovables como almidón de maíz o caña de azúcar.

Registro: son espacios de memoria reservados dentro de un microcontrolador para el manejo de operaciones del procesador o para su usó por el usuario.

Transhumeral: se refiere a una amputación realizada arriba del codo, cortando el húmero.

Transradial: se refiere a una amputación realizada en el antebrazo.