UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

CIENCIA APLICADA AL DEPORTE

Trabajo de graduación en modalidad de megaproyecto presentado por
Alejandro Noé Díaz Vargas
Javier Leonel Búcaro Argueta
José Alejandro Sagastume Pinto
Pablo Danilo Argueta Juárez
para optar al grado académico de Licenciados en Ingeniería Mecatrónica

Guatemala,
2017

CIENCIA APLICADA AL DEPORTE

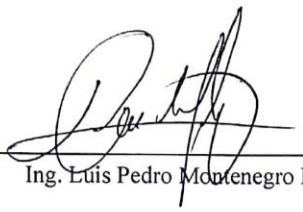
UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

CIENCIA APLICADA AL DEPORTE

Trabajo de graduación en modalidad de megaproyecto presentado por
Alejandro Noé Díaz Vargas
Javier Leonel Búcaro Argueta
José Alejandro Sagastume Pinto
Pablo Danilo Argueta Juárez
para optar al grado académico de Licenciados en Ingeniería Mecatrónica

Guatemala,
2017

Vo. Bo.:



(f) _____.
Ing. Luis Pedro Montenegro Mejicanos

Directores de los estudiantes que trabajaron el:Megaproyecto

(f) _____.
Msc. Carlos Alberto Esquit Hernández

Fecha de aprobación: Guatemala 22 de noviembre de 2017

ÍNDICE

LISTA DE CUADROS

LISTA DE FIGURAS

RESUMEN

La ciencia aplicada permite estudiar los movimientos y el desempeño del deportista con herramientas y deben tener ciertas características técnicas que permitan obtener datos relevantes de las disciplinas para obtener buenos resultados. Es por ello que el costo de las herramientas es muy elevado y para el Comité Olímpico Guatemalteco casi imposible de obtener. A raíz de este problema surge el Megaproyecto de *Ciencia Aplicada al Deporte*, con el fin de desarrollar herramientas y programas que permitan estudiar el movimiento y desempeño de un deportista, con dispositivos de bajo costo. En biomecánica se desarrolló un sensor inercial de bajo costo, que permite obtener la trayectoria de una persona que ejecute una caminata rectilínea. Este proyecto permite que el deportista cuantifique un movimiento rectilíneo y en base a ello pueda mejorar su técnica. En biofeedback se desarrolló un dispositivo que permite medir tiempos de reacción del deportista. También se desarrolló un dispositivo que cuantifica los tiros en la disciplina de tiro con armas de caza. Estos proyectos son para mejorar el desempeño de los deportistas mediante la medición de tiempos. Y se obtuvieron muy buenos resultados con deportistas en sus respectivas disciplinas. Se recomienda continuar estos proyectos, ya que se obtuvo buenos resultados en esta primera fase, aún con limitantes de tiempo y recursos.

I. INTRODUCCIÓN

Actualmente Guatemala no está posicionado como uno de los países pioneros en el deporte. Muchos de los países que se encuentran en esta categoría han hecho innumerables esfuerzos por desarrollar a sus deportistas, tanto en rendimiento como en técnica en cada una de sus disciplinas. Entre las actividades que han obtenido mejores resultados de los distintos países en el desempeño de los deportistas, se encuentra la ciencia aplicada al deporte.

El megaproyecto *Ciencia Aplicada al Deporte* surgió como una iniciativa del Comité Olímpico Guatemalteco con el objetivo de mejorar el desempeño de los deportistas guatemaltecos a nivel nacional e internacional. De ello surgieron dos grupos de proyectos: Biomecánica y Biofeedback, con el objetivo de desarrollar a los deportistas en distintos aspectos.

De biomecánica surgió el proyecto del sensor inercial de bajo costo con el objetivo de cuantificar la técnica de los deportistas, para posteriormente analizarla y mejorar. Se logró construir el sensor y obtener las trayectorias de dicho sensor enfocado a la caminata rectilínea.

De biofeedback se realizaron dos prototipos con el fin de ayudar a distintas disciplinas de alto desempeño en Guatemala. El primer prototipo consistió en diseñar un dispositivo que tiene como funcionalidad capturar el tiempo de reacción ante un estímulo, con el fin de retroalimentar al atleta y poder realizar un análisis con estos. El segundo prototipo se diseñó utilizando un acelerómetro, teniendo como funcionalidad capturar, desplegar y almacenar el tiempo de reacción y de disparo de los atletas de tiro con armas de caza, en las modalidades Fosa y Skeet.

II. OBJETIVOS

A. Objetivo general del megaproyecto

1. Diseñar herramientas especializadas de bajo costo para aplicaciones deportivas en temas de biomecánica y biofeedback.

B. Objetivos generales de los módulos

1. Diseñar y construir una placa de circuito impreso para el sensor inercial para aplicación deportiva.

2. Obtener datos de una Unidad de Medición Inercial y obtener medidas cinemáticas del movimiento realizado por el dispositivo biomecánico respecto a un marco global.

3. Diseñar un dispositivo de bajo costo utilizando LEDs y un sensor táctil, capaz de medir el tiempo que se tarda en reaccionar el atleta ante un estímulo con el fin de controlar la evolución de este durante los entrenamientos y que dicha información sea almacenada en una base de datos para luego ser analizada.

4. Diseñar una solución de bajo costo capaz de medir el tiempo de reacción de los atletas de tiro con armas de caza, desde que el blanco es lanzado hasta ser identificado por el atleta.

C. Objetivos específicos de los módulos

1. Diseñar una placa de circuito impreso que no supere las medidas de 5.00 x 5.00 cm de largo y ancho.

2. Implementar un ATMEGA328P-AU, una IMU BNO055 con acelerómetro y giroscopio, un Xbee y un módulo de cargador de batería de ser requerido, en la placa de circuito impreso.

3. Determinar las corrientes requeridas por el circuito para dimensionar adecuadamente los caminos de conducción de la placa de circuito impreso.

4. Determinar la viabilidad y necesidad de replicar un Arduino con un diseño de circuito impreso personalizado para el sensor inercial.

5. Validar los datos de la Unidad de Medición Inercial mediante la utilización de otros sensores.

6. Obtener la trayectoria del sensor inercial sin desfases.

7. Interpretar los datos desde la computadora para obtener la posición, velocidad y aceleración lineal en los tres ejes cartesianos en cualquier instante.

8. Graficar la trayectoria del dispositivo y obtener posiciones veraces respecto a un marco global.

9. Desarrollar un programa que implemente la función de capturar el tiempo que se tarda un atleta en reaccionar ante un estímulo.

10. Implementar comunicación inalámbrica entre el dispositivo y una computadora para poder utilizarlo de forma portátil.

11. Crear una base de datos sencilla que cumpla con almacenar de forma segura los datos de cada atleta.

12. Implementar un sensor táctil capacitivo para la captura del tiempo.

13. Realizar pruebas de captura correcta del tiempo, obteniendo información útil para los atletas.

14. Implementar funciones como tiempo máximo, tiempo mínimo y tiempo promedio para retroalimentar a los atletas.

15. Diseñar una estructura sencilla, portátil y de dimensiones adecuadas para utilizarlas dentro de los entrenamientos.

16. Determinar los requerimientos de tamaño, precisión y variables relevantes a medir, para obtener los datos necesarios y no afectar al atleta durante sus prácticas.

17. Seleccionar un sensor que cumpla con los requerimientos mencionados previamente, capaz de proporcionar la información relevante.

18. Diseñar la placa de circuito impreso para el sensor, respetando las dimensiones establecidas por los atletas y entrenadores.

19. Desarrollar e implementar un algoritmo capaz de medir el tiempo de reacción de un atleta mediante el movimiento del arma y disparo del arma, posteriormente desplegar la información.

20. Investigar e implementar módulos de comunicación inalámbrica de largo alcance.

III. JUSTIFICACIÓN

Como se puede observar en las competiciones deportivas a nivel mundial, cada vez se rompen más récords y se crean nuevas expectativas de las distintas disciplinas. Los países desarrollados principalmente, están invirtiendo muchos recursos en la mejora del desempeño de los deportistas mediante la utilización de la tecnología. Esta tecnología permite a los deportistas analizar su técnica y desempeño al llevar a cabo su disciplina deportiva.

La ciencia aplicada permite estudiar los movimientos y el desempeño del deportista con herramientas que obtienen información que es en muchas ocasiones imperceptible para un entrenador, o para el mismo deportista. Existen grandes empresas que venden tecnología para el desarrollo del deportista, pero su precio es sumamente alto. Es por ello que surge la iniciativa de desarrollar herramientas tecnológicas de bajo costo, para el desarrollo del deporte nacional en competiciones locales e internacionales.

La ciencia aplicada al deporte cubre muchas áreas del análisis deportivo, pero las dos principales son biomecánica y biofeedback. La biomecánica estudia el movimiento del cuerpo humano, y en la ciencia aplicada al deporte le permite al deportista analizar su técnica y mediante el estudio de la misma mejorar su desempeño. Biofeedback estudia el estado del cuerpo del deportista al ejecutar du disciplina, lo que permite al deportista rendir más y de mejor manera al realizar su disciplina.

IV. MARCO TEÓRICO

A. Ciencia aplicada al deporte

La ciencia aplicada al deporte es una conglomeración de distintas disciplinas con un enfoque primario en el desarrollo de ejercicios a través de la ciencia. Esta aplicación combina distintas ramas de la ciencia incluyendo fisiología, psicología, biomecánica y nutrición (Nelson Mandela University, 2017).

El principal propósito de la ciencia aplicada al deporte es asistir a los atletas en maximizar su potencial con el menor riesgo de padecer una lesión. Por otro lado, su enfoque se centra en realizar pruebas durante fin de temporada, pre-temporada y temporada de competición. Los componentes para medir el estado físico incluyen: capacidad aeróbica, capacidad anaeróbica, potencia muscular, fuerza, resistencia, agilidad y rapidez (Times Higher Education's, 2016).

Cabe mencionar que también se realizan pruebas de los deportes, se identifican las áreas de trabajo y se analizan los trabajos que realizan los atletas, se monitorea los niveles de estado físico y se incorporan técnicas de recuperación (The British Association of Sport and Excercise Sciences, 2017).

B. Psicología del deporte

Es una ciencia del deporte que estudia los factores psicológicos asociados con la participación y el rendimiento en el deporte, el ejercicio y otros tipos de actividad física. Utiliza conocimientos psicológicos y habilidades para obtener un rendimiento óptimo y para el bienestar de los atletas (American Psychological Association, 2017).

La psicología del deporte está dedicada a estudiar el cómo, porqué y bajo qué condiciones los deportistas, entrenadores y personas relacionadas con el atleta y el deporte se comportan en el modo que lo hacen, así como investigar la mutua influencia entre actividad física y la participación en el deporte y bienestar psico-físico, la salud y el desarrollo personal (Comité Olímpico Mexicano, 2017) .Los factores más importantes para un deportista de alto rendimiento que apoya la psicología del deporte son:

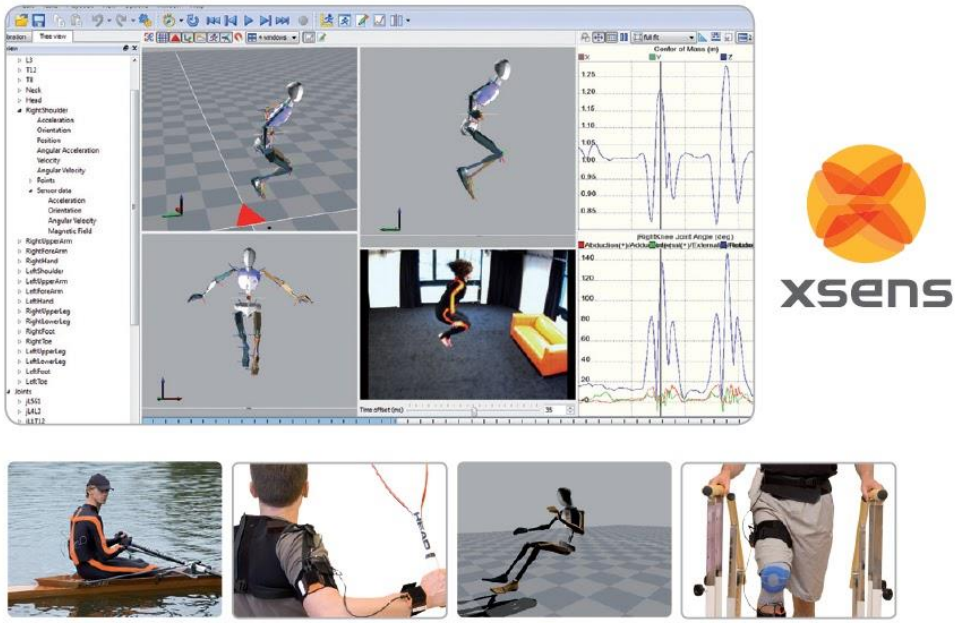
- Hacer frente a presiones de entrenamientos y competencias.
- Mejorar el rendimiento.
- Recuperación de lesiones.
- Impulsar el desarrollo personal.
- Prevención y/o apoyo emocional.

C. Biomecánica deportiva

La biomecánica es el estudio de la anatomía y las bases mecánicas del movimiento de los seres humanos. Actualmente la biomecánica es una de las principales herramientas y técnicas que se utilizan al momento de realizar un estudio a los atletas en alguna de las disciplinas en las que se desenvuelven. Esto se debe a que gracias al estudio de la biomecánica es posible estudiar con mayor precisión la técnica de los atletas, poder mejorar dicha técnica y evitar posibles lesiones en el proceso, así como hacer más eficiente y rápida la recuperación de lesiones.

La técnica de la biomecánica deportiva proviene de la aplicación de las leyes físicas al estudio del movimiento humano. Al realizar algún deporte se utiliza tecnología y equipo especial de fisioterapia, los cuales permiten observar, grabar y explicar de una forma clara y detallada el movimiento realizado por el atleta. De esta forma es posible identificar la técnica del atleta en la disciplina en la que se desenvuelve y poder mejorarla para evitar posibles lesiones y mejorar su rendimiento en dicha disciplina deportiva. En la Figura 1. se puede observar un análisis biomecánico con un equipo y software profesional.

Figura 1. Imágenes de análisis biomecánico con equipo y software profesional XSENS



Al ser una de las metodologías más utilizada por las instituciones deportivas más grandes del mundo, tanto federaciones nacionales como instituciones privadas, la biomecánica ha llegado a ser una de las razones principales del éxito deportivo en muchos países y clubes deportivos. Una de las grandes ventajas es que la biomecánica se puede aplicar tanto de forma individual como en deportes de equipos, logrando no solo un

mejor rendimiento individual, sino que también grupal. Por ello muchos de los mejores equipos deportivos del mundo utilizan la biomecánica en sus entrenamientos.

En los análisis biomecánicos aplicados a deporte se estudian tres tipos de movimiento, la traslación, la rotación y la combinación de ambos. Con ello se estudian parámetros como distancia, desplazamiento, aceleración y velocidad. Por lo tanto, es posible estudiar y analizar la cinemática lineal y angular de los atletas. Adicionalmente, al obtener estos parámetros, por medio de fórmulas físicas como las de Newton, se pueden obtener datos como fuerza, impulso, trabajo, energía, masa y potencia, los cuales son necesarios para un análisis completo de biomecánica deportiva.

D. Biofeedback

Es una técnica utilizada en entrenamientos que ayudan a controlar las funciones corporales involuntarias, como ritmo cardíaco, presión arterial, tensión muscular, y temperatura corporal. Para poder aprender a controlar dichas funciones internas es necesaria la ayuda de herramientas que miden y despliegan procesos psicológicos y mentales. Con biofeedback se utilizan sensores eléctricos que reciben información (feedback) sobre el cuerpo (bio).

Esta retroalimentación ayuda a crear cambios en nuestro cuerpo, como relajar ciertos músculos, para llegar a cumplir con los resultados propuestos. En esencia, biofeedback nos da el poder de usar nuestros pensamientos para controlar nuestro cuerpo, en muchos casos con el fin de mejorar la condición física (University of Maryland, 2017).

1. Tipos de biofeedback.

Existen diversos métodos utilizados para la realización de técnicas de biofeedback, dentro de los más utilizados se encuentran los siguientes.

a. Electroencefalogramas (EEG).

El encefalograma es un estudio de la función cerebral que recoge la actividad eléctrica del cerebro en situación basal y con métodos de activación, como la hiperventilación y la foto estimulación. La señal eléctrica recogida se amplifica y representa en forma de líneas, interpretándose la actividad de las distintas áreas cerebrales a lo largo del tiempo (Clínica Universidad de Navarra, 2015).

b. Respiración.

La respiración es el proceso vital para la vida por el cual el oxígeno del aire inhalado entra en la sangre, y el dióxido de carbono es exhalado a la atmósfera. El oxígeno es transportado por las arterias y venas hasta los tejidos; en las células que forman estos las sustancias nutritivas orgánicas son “quemadas” en un proceso de combustión y se libera energía calórica, agua y dióxido de carbono. La energía calórica producida es utilizada por el cuerpo para realizar sus diferentes funciones (ABC Color, 2017).

c. Electrocardiograma (ECG). El electrocardiograma es un estudio de rutina que se realiza para observar la actividad eléctrica del corazón. El electrocardiograma puede suministrar mucha información sobre el corazón y su funcionamiento. Con este estudio es posible averiguar más sobre el ritmo cardíaco, el tamaño y funcionamiento de las cavidades del corazón y el músculo cardíaco. (Texas Heart Instititute,2016)

d. Electromiografía (EMG). La electromiografía es el registro mediante una aguja o mediante electrodos de la actividad eléctrica muscular. Las fibras musculares, al contraerse, producen descargas que, recogidos por estos electrodos, dan unos patrones normales o indicativos de lesión distintos niveles del sistema neuromuscular. Localiza el área lesionada, concretando si es un problema de una mano, brazo o pierna, o si es algo más difuso y, definiendo si la lesión es de un músculo, nervio, tronco o raíz nerviosa, o de más de uno. (Clínica Universidad de Navarra, 2015)

e. Electrodermografía (EDG). La electrodermografía es un estudio que mide la resistencia, conductancia, potencial e impedancia eléctrica de la piel, para advertir sobre la presencia de ansiedad. (Clínica Universidad de Navarra, 2015)

f. Temperatura. Es la medida relativa de calor o frío asociado al metabolismo del cuerpo humano y su función es mantener activos los procesos biológicos. (EcuRed,2017)

E. Tiro con armas de caza

El deporte consiste en disparar a discos de arcilla de 4 y 5 pulgadas de diámetro, y 1/8 de pulgada de grosor con una escopeta calibre 0.12 y cartuchos de 24 gramos de plomos, en una cancha al aire libre. En la Figura 2. se observan los cartuchos y arma que utilizan. Los discos son lanzados por diversas máquinas y su trayectoria puede ser conocida o no dependiendo de la modalidad. Para que el disparo realizado al disco se tome como bueno debe existir al menos un fragmento observable del mismo. Las modalidades son skeet y foso olímpico. Cada vez que los atletas realizan un disparo, los atletas deben recargar la escopeta y ubicarse en la siguiente posición. (CDAG,2017)

Figura 2. Arma y municiones utilizadas en tiro con armas de caza.

(Sichling, 2017)

Aparte de la escopeta los atletas cuentan con gafas para proteger los ojos en caso exista rebote de perdigones, y tapones u orejeras para proteger los oídos del ruido del impacto de disparo. Adicionalmente cuentan con un chaleco deportivo que utilizan para llevar municiones en los bolsillos y proteger al atleta del

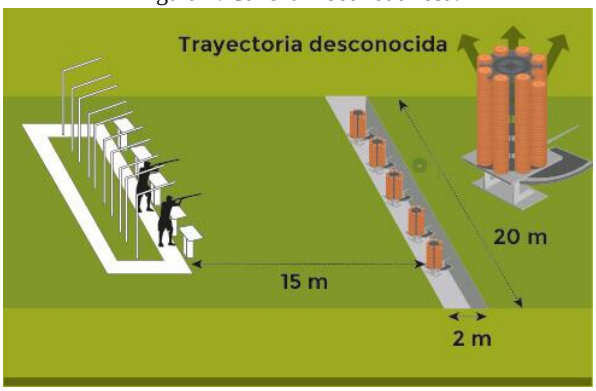
culatazo. En la Figura 3. se observa un atleta con la indumentaria mencionada previamente, así como la posición correcta de sostener la escopeta. (Marca, 2017)

Figura 3. Atleta de tiro con indumentaria reglamentaria utilizada.

(COG,2015)

1. Fosa. Consiste en disparar a 125 platos, en 5 series de 25 platos cada una en la categoría masculina y 75 platos, en 3 series de 25 platos en la categoría femenina. Los discos son lanzados aleatoriamente de 15 máquinas posibles, configuradas previamente a determinados ángulos. El atleta tiene derecho a realizar uno o dos disparos para intentar romper el disco. En cada serie el tirador inicia en la primera estación ubicada en el extremo izquierdo del campo. La Figura 4. contiene las dimensiones de un campo de Fosa, así como la ubicación de las estaciones, siendo la última estación en la que se encuentra ubicado el atleta. (Federación Venezolana de Tiro, 2017)

Figura 4. Cancha modalidad fosa.



(Marca,2017)

2. Skeet. Al igual que la modalidad Fosa se disparan 125 platos en 5 series de 25 cada una para la categoría masculina y 75 para la categoría femenina. Los platos son lanzados por una caseta alta y otra caseta baja, el tirador debe recorrer 9 puntos en donde se pueden realizar lanzamientos simples o dobles. En la

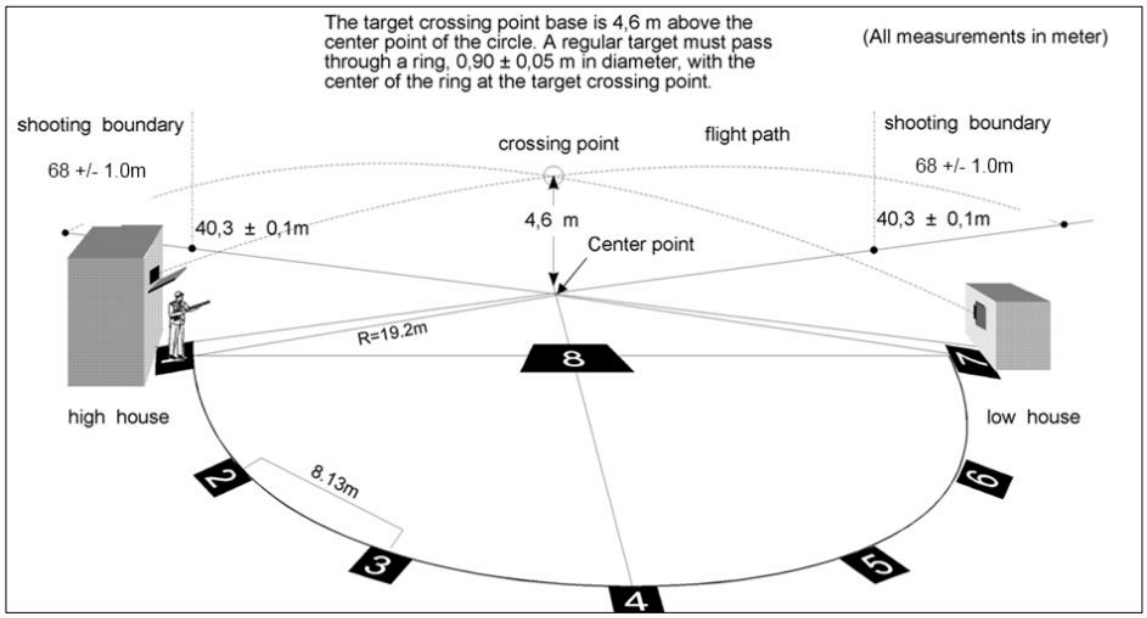
Figura 5. se observa las dimensiones de un campo de Skeet, así como la ubicación de cada una de las estaciones, estas estaciones se encuentran ubicadas al mismo nivel. En el lanzamiento simple se lanza un solo disco, y en el doble dos discos al mismo tiempo. Cada atleta solo tiene oportunidad de realizar un disparo

por disco. En el Cuadro 1. se observan los tipos de lanzamientos que se hacen en cada estación (Federación Venezolana de Tiro, 2017)

Cuadro 1. Lanzamientos por estación modalidad skeet

Estación	Tipo de lanzamiento	
	Caseta alta	Caseta baja
1	Simple	Doble
2	Simple	Doble
3	Simple	Doble
4	Simple	Simple
5	Simple	Doble
6	Simple	Doble
7	Doble	
4	Doble	Doble
8	Simple	Simple

Figura 5. Cancha modalidad skeet.



(ISSF,2013)

F. Taekwondo

El Taekwondo es el arte marcial tradicional de Corea que tiene unos 400 años de antigüedad. El origen del Taekwondo primitivo se asocia con la necesidad de protegerse de los adversarios. La sistematización del Taekwondo moderno empezó con la fundación de la asociación Coreana de Taekwondo en 1961.

Es un arte marcial en que se practica la técnica y el entrenamiento del cuerpo y la mente. Es un proceso de perfección personal mediante el desarrollo de la disciplina mental y el cuerpo con las técnicas de los pies y manos. Dicha disciplina mental, la confianza en sí mismo y el autocontrol, proporcionan al practicante una absoluta firmeza y estabilidad, tanto del cuerpo como de la mente. La palabra taekwondo tiene como significado:

- "Tae", técnica con los pies
- "Kwon", técnica con las manos
- "Do", filosofía, arte y forma de vida.

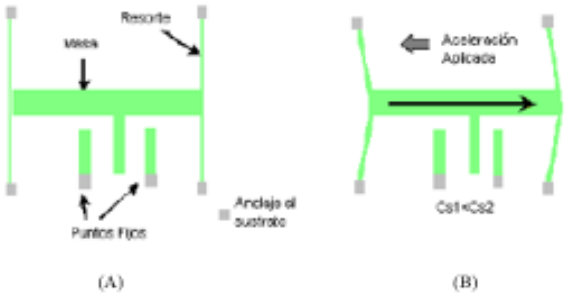
La Federación Nacional de Taekwondo de Guatemala, fue fundada en el año 1988, reconocida legalmente ante la Federación Mundial de Taekwondo (WTF) en el año 1991 (Federación Nacional de Taekwondo, 2013).

Figura 6. Competencia taekwondo

G. Acelerómetro

Es un dispositivo electromecánico, capaz de medir la aceleración estática o dinámica, en uno, dos o tres ejes. Los acelerómetros llevan dentro de ellos una cantidad de masa conocida denominada también masa sísmica o masa de prueba, conectada a un sistema de soportes y una estructura con propiedades de amortiguamiento. De tal manera que cuando el objeto experimenta cualquier aceleración la masa sísmica debe experimentar la misma. Una mejor descripción de esto se puede apreciar en la Figura 7. (Arenas,2008)

Figura 7. Estructura de un acelerómetro capacitivo. (A) Sensor en reposo. (B)Respuesta a una aceleración aplicada



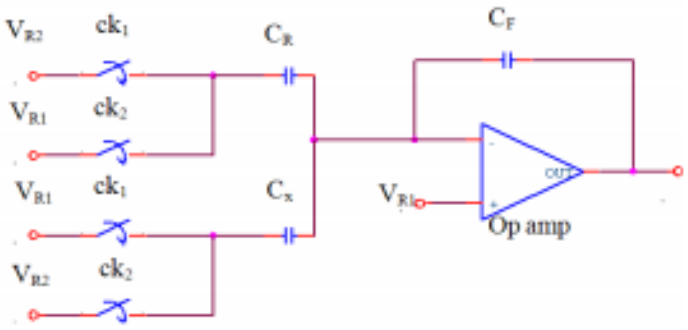
(Arenas,2008)

Existen diversos tipos de acelerómetros que utilizan diversas tecnologías: mecánicos, piezoeléctricos, piezoresistivos y capacitivos. Sin embargo, el principio de funcionamiento es el mismo. Los acelerómetros mecánicos se encuentran compuestos por una masa inerte, resortes elásticos y galgas extensiometricas. La aceleración produce una deformación en las galgas que se traduce en variaciones en la corriente detectada por el puente te Wheatstone. En estos acelerómetros la deformación es directamente proporcional a la aceleración. (Arenas,2008)

Los acelerómetros piezoeléctricos, se encuentran compuestos por cristales piezoeléctricos, la masa sísmica y una carcasa. El cristal se coloca entre la masa y la carcasa, para que cuando ocurra una aceleración la masa ejercerá una fuerza sobre el cristal produciendo diferencia de potencial que indicará la aceleración del objeto. A diferencia de los piezoeléctricos los piezoresistivos utilizan un sustrato en lugar del cristal. Sin embargo, el principio es similar ya que cuando la masa ejerza una fuerza sobre el sustrato variara su resistencia, la cual se encuentra conectada a un puente de Wheatstone en el cual se mide la intensidad de la corriente. (Arenas,2008)

Los capacitivos cuentan con una masa, resortes y placas capacitivas, las cuales dos se encuentran acopladas a la carcasa del sensor y otra a la masa sísmica. El funcionamiento de este tipo de acelerómetros se basa en medir las variaciones de la capacitancia, ya que se sabe que la capacitancia de un condensador se encuentra dada, entro otros, por la distancia que separa las placas. De esta manera cuando la masa experimenta una fuerza de aceleración ejerce un movimiento, que hace variar la distancia entre placas, provocando así las variaciones mencionadas previamente. Dichas variaciones son detectadas y procesadas por un circuito convertidor de capacitancia en voltaje. Produciendo así un voltaje en la salida. (Arenas,2008)

Figura 8. Convertidor de capacitancia a voltaje.



(Alam, *et al*, 2010)

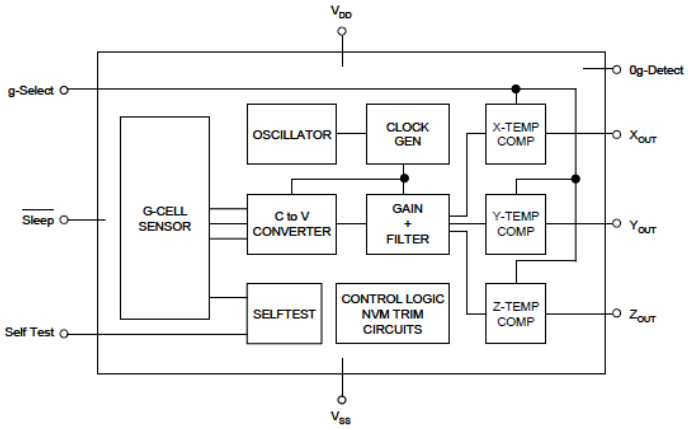
En la Figura 8. se puede observar una versión de un circuito convertidor de capacitancia en voltaje o CVC por sus siglas en inglés. CX es el valor de capacitancia detectado por el sensor y CR y CF son valores previamente definidos. VR1 es el voltaje de modo común y VR2 es el voltaje de referencia. CK1 y CK2 son

señales de reloj independientes. Cuando la señal CK2 se encuentra en un estado lógico alto, el voltaje de referencia va a cargar el capacitor CX, mientras que el capacitor CF va a almacenar el voltaje offset del OpAmp. Cuando la señal CK1 se encuentra en un estado lógico alto, el voltaje de referencia carga el capacitor CR y el capacitor CF está conectado a la salida. Si se sigue el principio de conservación de la carga, el voltaje de salida Vo se expresa de la siguiente manera: (Alam, *et al*, 2010)

$$Vout = \frac{(C_X - C_R)}{C_F} * (V_{R2} - V_{R1}) \quad (1)$$

1. Acelerómetro MMA7361. Es un acelerómetro capacitivo de baja potencia y bajo perfil que cuenta con un filtro pasa baja de un polo, un circuito acondicionar de señal, compensación de temperatura. En la Figura 9. se puede apreciar el diagrama de bloques simplificado de este acelerómetro y en el Cuadro 2. una breve descripción de los pines. (Freescale,2008).

Figura 9. Arquitectura acelerómetro MMA7361



(Freescale,2008)

Cuadro 2. Descripción de pines MMA7361

Nombre del Pin	Descripción
Xout	Salida de voltaje del eje X.
Yout	Salida de voltaje del eje Y.
Zout	Salida de voltaje del eje Z.
Vss	Tierra de alimentación.
Vdd	Voltaje de alimentación.
Sleep	Entrada digital, en la cual si el estado es alto el sensor entra en modo reposo.
0g-Detect	Salida digital, para aplicaciones en caída libre.
g-Select	Entrada digital, en la cual si el estado es bajo el rango de medición es de 1.5g y la sensibilidad es de 800mV/g. Si el estado es alto el rango es de 6g y la sensibilidad es de 206mV/g.
Self-Test	Entrada digital, si el estado es alto se inicia la verificación del funcionamiento mecánico y eléctrico del sensor.

(Freescale,2008)

H. Giroscopios

Los giroscopios son instrumentos de medición que cuantifican la razón de cambio de los ángulos en la triada x, y, z del instrumento de medición. La velocidad angular es medida, al igual que en los acelerómetros por medio de sistemas de micro-electromecánica. Cada línea del giroscopio mide la rotación del sensor en su eje correspondiente, en un marco de referencia local. Por medio de esta razón de cambio de la posición angular es posible obtener una posición angular preliminar, y combinada con otros sensores se puede obtener dicha posición con menos ruido y con respecto a un marco de referencia global. El dispositivo que se utilizará funciona con el protocolo I2C, por lo tanto, por medio de dicho direccionamiento se puede acceder a la información del sensor utilizando un microcontrolador.

I. Magnetómetros

Los magnetómetros miden la dirección y magnitud de los campos magnéticos. La Tierra posee un campo magnético, varía con la localidad y puede ser cuantificado por medio de un magnetómetro. Por medio de los magnetómetros de sistemas micro-electromecánicos se puede obtener la dirección de estos campos magnéticos, lo cual es de gran utilidad para calcular la orientación del sensor. La ventaja de dichos sensores es que la orientación que se obtiene de ellos es con respecto al margo global de la Tierra. Las mediciones de este instrumento pueden ser fusionadas con los sensores anteriormente descritos para obtener una orientación más robusta y con respecto a un marco global. Este sensor es de mucha utilidad, ya que, en muchos de los algoritmos de fusión de sensores para orientación, su papel es de corrección o reajuste, lo cual brinda una orientación más certera con el marco global.

J. Unidad de medición inercial

Una Unidad de Medición Inercial (IMU) es un dispositivo que cuenta con varios sensores inerciales en un mismo circuito. Estos poseen un bus para direccionamiento I2C, por medio del cual se puede accesar individualmente cada uno de los sensores, y de esta manera obtener las mediciones de interés. La cantidad de grados de libertad que ofrece una Unidad de Medición Inercial indica la cantidad de sensores que posee. Para la obtención de una orientación correcta se necesita un mínimo de 9 grados de libertad, es decir que se necesita cada uno de los sensores anteriormente descritos, para una orientación y trayectoria certeros.

K. Comunicación Serial

La comunicación serial es un protocolo muy común utilizado para la comunicación entre dispositivos que se incluye de manera estándar en cualquier computadora. Su concepto es sencillo, el puerto serial envía y recibe bytes de información un bit a la vez, este método de comunicación es ideal para transmisión de datos

a grandes distancias. Típicamente, la comunicación serial es utilizada para trasmitir datos en el formato ASCII, y utiliza tres líneas de trasmisión: una tierra o referencia, una de transmisión y otra de recepción. Para que dos puertos se puedan comunicar entre sí, es necesario que las siguientes características sean iguales: baud rate o velocidad de trasmisión, indica el número de bits por segundo que se transfieren. Bits de datos, indica la cantidad de bits en la transmisión, bits de parada, establece el final de comunicación de un solo paquete. Finalmente, la paridad que es una forma sencilla de verificar si hay errores en la transmisión serial. (National Instruments, 2006).

1. UART – Universal Asynchronous Receiver/Transmitter. El Transmisor/Receptor Universal Asíncrono, mejor conocido como UART por sus siglas en inglés, es el componente electrónico encargado de la comunicación serial entre computadoras, microcontroladores o dispositivos. El UART toma bytes de información y los transmite bit por bit de manera secuencial. En la terminal de destino, existe un segundo UART capaz de traducir la información en bytes nuevamente. (Durda, 2014).

La comunicación asíncrona permite que los datos sean transmitidos sin tener que enviar al receptor una señal de reloj. En su lugar el emisor y el receptor deben acordar los parámetros de temporización previamente y se deben añadir bits especiales a cada palabra que se utilizan para sincronizar el emisor y el receptor. Cuando una palabra es enviada al UART, el bit de inicio es añadido al inicio de la misma, el cual le indica al receptor que está por recibir una palabra, posteriormente son enviados los bits iniciando con el bit menos significativo, cada bit es transmitido exactamente la misma cantidad de tiempo. Posteriormente son enviados los bits de paridad y el bit de parada. (Durda,2014)

2. SPI – Serial Peripheral Interface. La interfaz serial periférica o SPI, por sus siglas en inglés, es un estándar de comunicaciones utilizada para la transferencia de paquetes de datos de forma serial. Normalmente es utilizad para la comunicación entre dispositivos y periféricos externos. Los dispositivos conectas al bus SPI pueden transmitir y recibir datos al mismo tiempo, ya que dentro de su arquitectura cuenta con una línea para transmisión (MOSI) y otra línea para recepción de datos (MISO), adicionalmente cuenta con una línea para la señal de reloj (SLCK) y otra para la selección del destinatario o esclavo (SS). (Texas Instruments, 2012)

El protocolo SPI se comunica utilizando una relación maestro-esclavo, en donde el maestro es el que siempre inicia la comunicación y selecciona con cual esclavo desea comunicarse. Cuando el maestro genera una señal de reloj y determina con que esclavo se desea comunicar, los datos son trasferidos en ambas direcciones, es decir tanto el maestro envía datos al esclavo, como el esclavo al maestro. Depende del maestro y esclavo determinar si el byte recibido es representativo. (Kalinsky & Kalinsky, 2002)

Existen cuatro formas de enviar la información en un bus SPI, cada modo va a depender del estado de dos parámetros basados en la señal de reloj. El primero es la polaridad del reloj, que determina el estado de la señal en los momentos en los que no se transmite el segundo es la fase del reloj que determina el momento en el pulso de reloj en el que se debe realizar la toma de datos. En la Cuadro 3. se realiza una breve descripción de cada modo. La configuración de independiente para cada esclavo es decir que cada esclavo puede tener una configuración distinta, por lo que el maestro se debe adaptar a la configuración de cada esclavo. (Navarro, 2014)

Cuadro 3. Modos de comunicación SPI

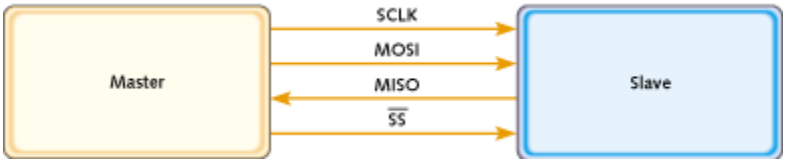
Modo	Polaridad de reloj (CPOL)	Fase de reloj (CPHA)	Descripción
0	0	0	El estado del reloj permanece en cero y la información se envía en cada transición de bajo a alto.
1	0	1	El estado del reloj permanece en cero y la información se envía en cada transición de alto a bajo.
2	1	0	El estado del reloj permanece en alto y la información se envía en cada transición de bajo a alto.
3	1	11	El estado de reloj permanece en alto y la información se envía en cada transición de alto a bajo.

(Navarro, 2014)

En este protocolo se puede realizar una conexión punto-punto como la de la Figura 10. en donde solo existe un maestro y un esclavo, en este caso la línea SCLK es la encargada de transmitir la señal de reloj, la línea MOSI envía datos del maestro al esclavo y la línea MISO función de manera contraria, es decir envía datos del esclavo al maestro. Otra forma de realizar una conexión de este tipo es de punto-multipunto, en donde existe un solo maestro y diversos esclavos. Esta conexión puede ser paralela o encadenada. En la configuración tipo encadenada el maestro envía datos solo al primer esclavo y este se encarga de enviarle datos al siguiente y así hasta que el ultimo esclavo le envíe los datos al maestro. Además, se utiliza una única línea de selección de esclavo en conexión paralela a cada esclavo. (Navarro, 2014).

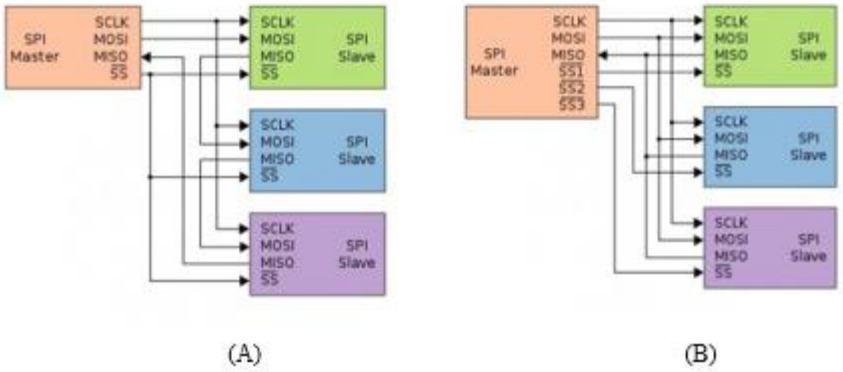
A diferencia de la configuración encadenada, la configuración paralela permite que el maestro envíe y reciba datos de cada esclavo de manera independiente, para seleccionar con que esclavo se comunicara existe una línea de selección de esclavo en el maestro por cada esclavo que exista en el sistema. En la Figura 11. se puede observar la diferencia entre ambas configuraciones. (Navarro, 2014)

Figura 10. Conexión punto-punto mediante SPI



(Kalinsky & Kalinsky, 2002)

Figura 11. Conexión punto-multipunto mediante SPI A) Configuración encadenada (B) Configuración paralela



(Navarro,2014)

L. Redes inalámbricas de área personal (WPAN)

Las redes inalámbricas de área personal o WPAN por sus siglas en ingles son redes utilizadas para cubrir distancias cortas, utilizadas para conectar varios dispositivos portátiles o personales sin tener que utilizar cables. Por lo genera la comunicación de estos dispositivos no requiere de altos índices de transmisión de datos, por lo que el consumo de energía es bajo. Siendo la tecnología WPAN ideal para el uso de dispositivos pequeños que funcionen con baterías. (Camargo,2009)

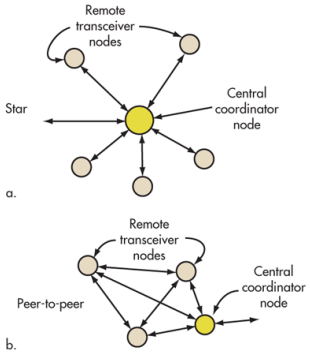
1.IEEE 802.15.4. Es un estándar desarrollado por el grupo 802.15 que se caracteriza por su flexibilidad de red y bajo consumo de energía por lo que es muy utilizado para aplicaciones que requieran una tasa baja de transmisión de datos. El estándar define la capa física, PHY por sus siglas en inglés, y la capa de control de acceso a medios, MAC, del modelo de interconexión de sistemas abiertos. La capa física define la frecuencia, la potencia, la modulación y otras condiciones inalámbricas del enlace, mientras que la capa de control de acceso a medios define el formato del tratamiento de datos. (Frenzel, 2013)

El objetivo principal de este estándar es proveer un formato base al que otros protocolos y características se puedan agregar en las capas superiores del modelo de interconexión de sistemas abiertos. El estándar puede operar en tres frecuencias distintas, aunque la más utilizada es la de 2.4 GHz, es altamente tolerante al ruido y la interferencia. También cuento un acceso múltiple de sentido de portadora con evitación de colisión, el cual permite a múltiples usuarios o nodos acceder al mismo canal en diferentes momentos sin que existe interferencia. El rango de transmisión varía dependiendo de la naturaleza del camino que debe ser en su mayor parte línea de visión. Bajo las mejores condiciones el alcance puede ser de 1000 metros con un camino claro al aire libre. (Frenzel, 2013)

IEEE 802.15.4 define dos topologías, siendo estas una estrella básica, y una peer-to-peer, como se observa en la Figura 12. En la topología estrella toda la comunicación entre nodos debe ser mediante un nodo central,

mientras que en la topología peer-to-peer cualquier nodo puede comunicarse con cualquier otro nodo. Esta topología puede ser expandida en otras, como por ejemplo la topología mesh. (Frenzel, 2013)

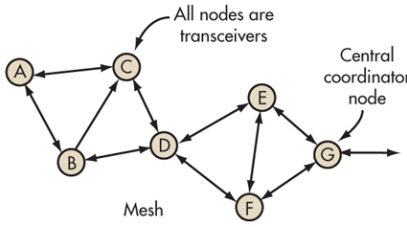
Figura 12.Modelos de topología definidos por IEE 802.15.4



(Frenzel,2013)

2. ZigBee.Es un protocolo normalizado para los WPAN's diseñado para soportar un diverso mercado de aplicaciones con una conectividad sofisticada. Este estándar utiliza las capas 3 en adelante del modelo de interconexión de sistemas abiertos, para definir funciones de comunicación adicionales, dentro de estas mejoras se encuentra la autenticación con nodos válidos, encriptación para seguridad y una capacidad de enrutamiento y reenvío de datos que permite la creación de redes mesh. El estándar es comúnmente utilizado para la creación de redes de sensores inalámbricos configurados en la topología mesh, en donde cada nodo se puede comunicar con el más cercano. (Freenzel,2013) (Camargo,2009)

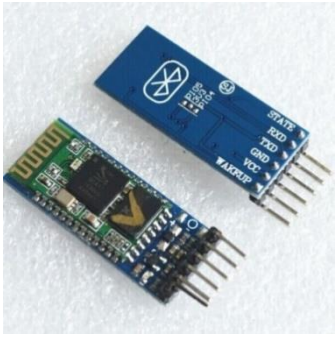
Figura 13.Topología mesh utilizada en redes ZigBee.



(Freenzel,2013)

3. Bluetooth HC-05. Es un módulo bluetooth que puede trabajar en dos modos, datos y comandos. En el modo de datos, el módulo se encarga de conducir la información hacia los puertos UART, es decir envía y recibe datos con otro módulo. Cuando se encuentra en modo comandos, se puede cambiar la configuración del módulo dependiendo de la aplicación en la que se va a utilizar. Para configurarlo se utilizan los comandos AT y se puede configurar con un adaptador que facilita su manejo. El HC-05 puede ser configurado como maestro o esclavo, sin embargo, sin importar cuál sea su configuración solo puede trabajar en una red punto-punto. (López, 2016)

Figura 14. Módulo HC-05



(Prometec, 2017)

M. Xbee

Según Digi, los módulos Xbee son soluciones integradas que brindan un medio inalámbrico para la interconexión y comunicación entre dispositivos. Estos módulos utilizan el protocolo IEEE 802.15.4 y se basan en el protocolo ZigBee. Los módulos Xbee permiten crear redes punto a multipunto; o redes puno a punto. Dependiendo de la serie del módulo y la antena seleccionada las propiedades del mismo pueden llegar a ser significativas ya que existen módulos con antena RP-SMA que llegan a tener un alcance 1600 metros. Los módulos Xbee operan en una banda de frecuencia de 2.4 GHz, aunque también pueden operar en bandas de 900 y 868 MHz, estos módulos con una antena de alta ganancia pueden llegar a tener un alcance de 24 kilómetros. Para configurar un módulo Xbee es necesario contar con el explorador USB de Xbee y el software XCTU (Xbee, 2017)

1. Xbee Serie 2. Los Xbee's Serie 2 son módulos de radiofrecuencia que operan dentro de la banda de frecuencia de ISM 2.4GHz. Su tasa de datos es de 250 250 Kbps en RF y 1Mbps de forma serial. En ambientes urbanos pueden llegar a tener un alcance de 30 metros mientras que en ambientes exteriores pueden llegar hasta los 1,200 metros. El voltaje de operación de estos módulos es de 2.1 a 3.6 y cuentan con versiones con antena de cable, antena de chip y antena RPSMA. (Digi,2017)

Figura 15. Módulo Xbee serie 2 con antena de cable.

(Digi, 2017)

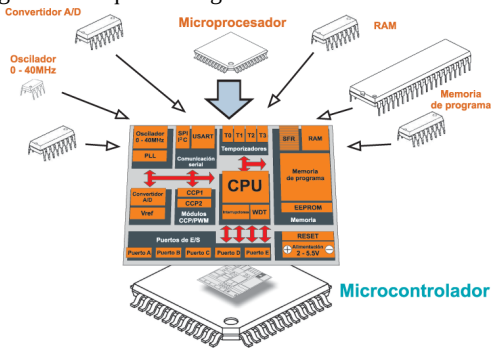
2. XCTU. Es una aplicación multiplataforma gratuita, que fue diseñada para permitir a los desarrolladores interactuar con los módulos RF de Digi, mediante una interfaz gráfica sencilla. Cuenta con diversas herramientas que facilitan la configuración y pruebas de los Módulos Xbee. Con esta plataforma se pueden configurar múltiples módulos Xbee a la vez, y verificar que la configuración realizada haya sido la correcta. La interfaz gráfica cuenta con una sección de configuración de módulos, otra sección para abrir la

consola e iniciar comunicación entre módulos y una tercera sección para la vista gráfica de red, en la cual se puede observar la red Xbee junto con la intensidad de la señal de cada conexión. El XCTU es compatible con los sistemas operativos de Windows, Linux y Mac OS X (Digi, 2017).

N. Microcontroladores

Un microcontrolador se puede describir con un pequeño ordenador que contiene en su interior una unidad de procesamiento central, encargada de ejecutar las instrucciones de operaciones lógicas, aritméticas y movimiento de datos. Una unidad de memoria para guardar datos o programas, soporte (reloj y reset) y periféricos de entrada y salida. Además de estas características los microcontroladores pueden llevar otras características más sofisticadas para incrementar la potencia del mismo, dentro de ellas se encuentran, interrupciones, puertos analógicos de entradas/salidas, puertos seriales e interfaces con memoria externa. (Velasco, 2006)

Figura 16. Arquitectura general de un microcontrolador.



(Mikroelectronika,2010)

1. Arduino Uno. El Arduino Uno es una tarjeta basada en el microcontrolador ATmega328 de Atmel. Tiene 14 pines digitales de entrada/salida, de los cuales 6 pueden ser utilizados como salidas de ancho de pulso modulable o PWM, por sus siglas en inglés. Tiene 6 entradas analógicas, un cristal oscilador de 16 MHz, una conexión USB, un conector de alimentación, pines para conexión ICSP y un botón de reset. Adicionalmente cuenta con un chip ATmega16U2 programada como un convertidor USB a serial. (Arduino,2017)

El voltaje de operación del Arduino Uno es de 5V, el cual se puede obtener de un cable USB conectado a una computadora. También se puede conectar una fuente de alimentación externa en los pines Vin y Gnd, o conectar una batería de 9V en el conector Power. La tarjeta puede operar con alimentación externa de 6 a 20V sin embargo no se recomienda esto, ya que si se alimenta con menos de 7V el pin de 5V entregaría menos de 5V y la tarjeta puede ser inestable. Por otro lado, si se alimenta con más de 12V el regulador de voltaje se puede calentar y dañar la placa. Por lo que lo recomendable es alimentarlo en un rango de 7 a 12V (Arduino,2017)

Los pines digitales pueden ser utilizados como entradas o salidas, configurándolos en las funciones pinMode (), digitalWrite () y digitalRead (). Los pines operan a 5 voltios y pueden proveer o recibir un máximo de 40mA. Los pines 0 y 1 están conectados a al módulo UART del microcontrolador y al chip ATmega16U2 como RX y TX, para comunicarse de manera serial con otro dispositivo que cuente con un módulo UART. Los pines 2 y 3 pueden ser configurados por interrupciones externas por cambios de valor o cambios en la señal. Los pines 3, 5, 6, 9, 10 y 11 pueden ser utilizados como un PWM de 8 bits. Los pines 10, 11 y 12 se pueden utilizar con el módulo SPI. Los pines analógicos A0 a A45 poseen una resolución de 10 bits, es decir pueden obtener 1024 valores diferentes, y otorgan un valor desde 0 hasta 5 voltios. (Arduino,2017)

Figura 17. Arduino Uno.

(Arduino,2017)

2. Arduino Nano. El Arduino Nano es una tarjeta pequeña basada en el microcontrolador ATmega328 si es la versión 3.0 o ATmega168 si es la versión 2.x. Ofrece las mismas especificaciones que el Arduino UNO, pero en una tarjeta de menor tamaño. También el cable de alimentación que utiliza es un USB tipo B. (Arduino, 2017)

Figura 18. Arduino Nano.

(Arduino,2017)

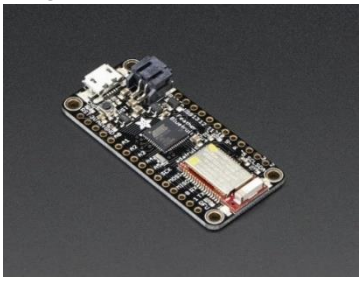
Cuadro 4.Comparación Arduino UNO vs. Arduino Nano

Especificaciones	Arduino UNO	Arduino NANO
Microcontrolador	ATmega328P	ATmega328
Voltaje de operación	5 V	5 V
Voltaje de entrada	7 – 12V	7 – 12 V
Pines digitales I/O	14	22
Pines digitales PWM salida	6	6
SRAM	2 KB	2 KB
Memoria flash	32 KB	32 KB
Velocidad de reloj	16 MHz	16 MHz
EEPROM	1 KB	1 KB
Corriente DC por pin I/O	20 mA	40 mA
Dimensiones PCB	53.4 x 68.6 mm	18 x 45 mm

3. SoftwareSerial Library. La librería SoftwareSerial fue desarrollada para permitir comunicación serial en otros pines digitales del Arduino. Con esta librería es posible tener varios puertos de comunicación serial, sin embargo, solo uno puede recibir datos a la vez. Para utilizar esta librería se utiliza el comando #include <SoftwareSerial.h> y se declara un objeto tipo SoftwareSerial en donde los parámetros del mismo son los pines a utilizar como transmisor y receptor. El resto de funciones son las mismas utilizadas por la librería Serial. (Arduino,2017)

4. Adafruit Feather BLE. El Feather BLE es una placa diseñada por la compañía Adafruit, utiliza el microcontrolador ATmega32u4, y trae incorporado un módulo bluetooth de baja energía. Los pines lógicos del controlador funcionan con 3.3 V, cuenta con 20 pines GPIO, de los cuales 7 pueden producir salidas PWM y 10 como entradas analógicas. También cuenta con un módulo para cargar la batería y un indicador LED cuando la carga de la batería se encuentra baja. La placa se puede programar con el IDE de Arduino, para ello es necesario instalar el complemento Adafruit AVR Boards, el cual permite al IDE reconocer placas no fabricadas por Arduino. Para configurar el módulo bluetooth del Feather se utilizan los comandos AT. (Adafruit,2017)

Figura 19. Adafruit feather 32u4



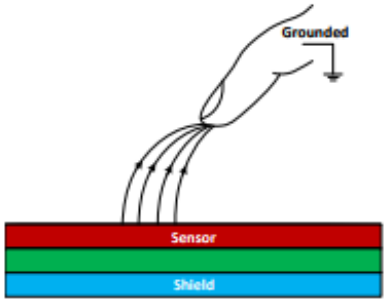
(Adafruit, 2017)

O. Sensor táctil capacitivo

La detección capacitiva es una tecnología basada en el acoplamiento capacitivo que toma la capacitancia producida por el cuerpo humano como entrada. Un sensor capacitivo utiliza cualquier material conductor y detecta cualquier cosa que conduce o que tiene una conductividad eléctrica distinta del aire (Lion Precision, 2012).

La cantidad de corriente está determinada por la capacitancia, y la capacitancia está determinada por el área y la proximidad del objeto conductor. A medida que el objeto es más grande y se encuentra más cercano al sensor este causa una mayor corriente. La capacitancia también se ve afectada por el tipo de material no conductor en el espacio entre los objetos (Texas Instrument, 2014).

Figura 20. Esquema sensor capacitivo



La capacitancia puede ser interpretada matemáticamente de la siguiente manera:

$$C = \frac{Q}{V} \tag{2}$$

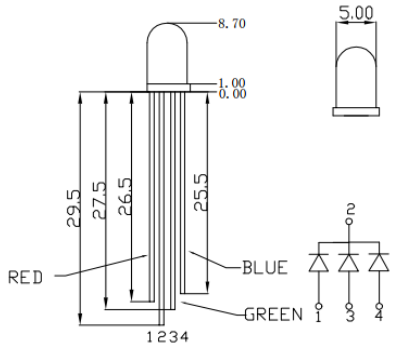
Dónde C está relacionada con la carga almacenada a un voltaje dado.

P. LED RGB

Un Diodo Emisor de Luz o LED es un tipo de diodo especializado que contiene una capa delgada dopada de un material semiconductor. Dependiendo del material semiconductor usado cuando el LED se encuentra en configuración directa emitirá luz a ciertas longitudes de onda espectrales. Debido a la capa delgada una razonable cantidad de fotones son expulsados y producen una luz de color. Al aplicarle una pequeña corriente eléctrica produce luz y dicha luz es de un determinado color dependiendo del material semiconductor (Iluminación LED RGB SAC, 2017).

La mayoría de LEDs producen un único color, sin embargo, existen LEDs multicolores que producen un rango de diferentes colores en un solo dispositivo. La mayoría de estos LEDs son fabricados de manera que un paquete contiene de dos a tres LEDs.

Figura 21. Esquema dimensiones LED



Q. Interfaz gráfica

La interfaz gráfica de usuario, GUI por sus siglas en inglés, son los elementos gráficos que permiten la comunicación entre un humano y un sistema o estructura. Esta surge como la evolución de la línea de comandos de los primeros sistemas operativos. El objetivo principal de una interfaz gráfica es que al usuario se le facilite la ejecución de acciones que desea realicen el sistema con el que está tratando. Para la ejecución de estas acciones las GUI utilizan una serie de comandos dictaminados por el mouse o teclado del ordenador. En la actualidad existen diversas herramientas y lenguajes para el desarrollo de interfaces de usuario, como por ejemplo GLADE, Java o Python. (Luna, 2004).

1. Java. Es un lenguaje de programación, que surge en los años 90, orientado a objetos, tipificado estáticamente, compilado, multiprocesos, robusto, seguro y ampliable que permite el desarrollo de aplicaciones en cualquier sistema de computación. También es un lenguaje independiente de plataforma, ya que los programas desarrollados en este lenguaje pueden correr en cualquier sistema sin cambios. Esto se logra ya que cuando se compila un programa se genera un archivo llamado byte-code, el cual puede ser leído y ejecutado por cualquier computadora que tenga un intérprete de java. (Bell, 2003)

   a. Librería PanamaHitek_Arduino. Está librería es un conjunto de métodos ordenados por el ingeniero Antony García, de la Universidad Tecnológica de Panamá que facilitan el proceso de comunicación entre Arduino-Java y viceversa. Es una librería de acceso público por lo que cualquier persona puede hacer uso de ella. Para poder utilizar está librería se debe importar la librería a los archivos del proyecto. (García, 2016).

   b. Librería Apache POI. Es una librería que permite manipular diversos formatos de archivo basados en los estándares de Office Open XML y en el formato de documentos de OLE 2 de Microsoft. Permite leer y escribir archivos de Excel, Power Point y Word utilizando el lenguaje de Java. (Oliver, *et Al, 2017*).

2. Processing. Es un software flexible de código abierto, que relaciona conceptos de principios de forma visual con movimiento e interacción. El lenguaje de Processing es un lenguaje de programación de texto diseñado para generar y modificar imágenes. Processing permite realizar operaciones de dibujo vectorial, procesamiento de imágenes, modelos de color, eventos de teclado y ratón, comunicación en red y programación orientada a objetos de una manera sencilla. Debido a que posee diversas bibliotecas se pude ampliar la capacidad de Processing para generar sonido, enviar/recibir datos en diversos formatos e importar/exportar formatos de archivo 2D y 3D. (Reas, 2014)

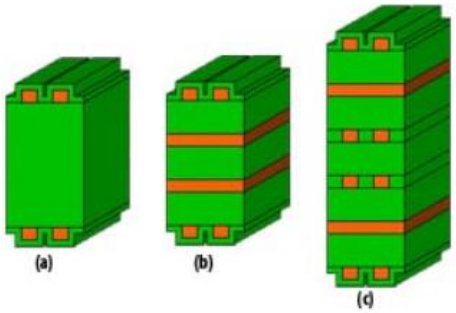
R. PCB – Printed Boarded Circuit

Una placa de circuito impreso o PCB, por sus siglas en inglés, es una placa utilizada para realizar las interconexiones eléctricas entre los componentes de un circuito eléctrico. La placa está constituida por un material no conductor, generalmente fibra de vidrio, recubierto por un material conductor, generalmente cobre. El material conductor en la superficie del material no conductor forma caminos o pistas entre los componentes del circuito para las interconexiones entre ellos.

Existen varios tipos de placas de circuito impreso, de una capa, de doble capa y de múltiple capa. Las placas de circuito impreso de una capa consisten en placa cuyo material conductor únicamente está en la una de las superficies de la placa, las placas de doble capa tienen material conductor en las ambas superficies y las placas de múltiple capa tienen material conductor en ambas superficies y tienen capas dentro del material no conductor. Sin embargo, las placas más utilizadas actualmente son las placas de circuito impreso de doble capa. En la

Figura 22. se puede observar los tipos de placas explicados anteriormente.

Figura 22.Ejemplo de placas de 2 capas (a), 4 capas (b) y 6 capas (c)

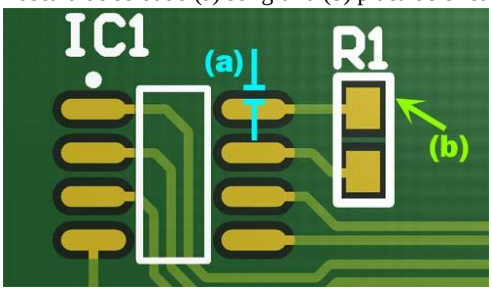


En el diseño y manufactura de las placas de circuito impreso se utilizan varias técnicas y conceptos que ayudan para que la placa tenga un mejor acabado, este mejor identificada y sea más eficiente. Entre estas técnicas están la máscara de soldado, serigrafía, pads, caminos de cobre, perforaciones metalizadas, entre otras. La máscara de soldado consiste en un barniz especial, el cual se aplica a los circuitos impresos en la etapa de fabricación. Este barniz previene que la soldadura provoque un corto circuito accidental entre nodos,

en la Figura 23. se puede observar la máscara de soldado alrededor del nodo. Estos barnices frecuentemente son de color verde, rojo o azul.

La serigrafia es utilizada para realizar impresiones sobre la máscara de soldado. Estas impresiones sirven para identificar características de la placa como orientación, posición, puntos de prueba y referencia de los componentes del circuito para facilitar su identificación y ensamblado. También se utiliza para colocar información importante del diseñador o fabricante, como, nombre del producto, compañía, diseñador, breves instrucciones. En la Figura 23. se puede observar la implementación de la serigrafia en una placa de circuito impreso.

Figura 23. Máscara de soldado (a) serigrafia (b) placa de circuito impreso



Los componentes tienen distintos tipos de empaquetados o encapsulados, es común encontrar a un mismo componente con diferentes tipos de encapsulados. Existe una gran variedad, pero hay tres que son los más utilizados, a través de orificio (Thru-hole), montaje superficial (SMD) y arreglo de bolas en grilla (BGA). Los componentes *Thru-hole* son aquellos que tienen pines para ser ensamblados en perforaciones de la palca de circuito impreso, los cuales son soldados en la capa opuesta. Los componentes SMD son aquellos que se ensamblan de forma superficial a la placa de circuito impreso. Estos componentes por lo general son mucho más pequeños que sus contrapartes *Thru-hole*, por lo que permiten diseñar placas mucho más compactas. Los componentes BGA son aquellos que tienen una gran cantidad de pines, más de 300 pines. Para el ensamble de estos componentes es necesario maquinaria especializada, ya que los pines son bolas de soldadura que se funden para conectarse a los pads. En las Figuras 24, 25 y 26 se muestran los tres tipos de empaquetado *Thru-hole*, SMD y BGA, respectivamente.

Figura 24. Componentes tipo Thru-hole

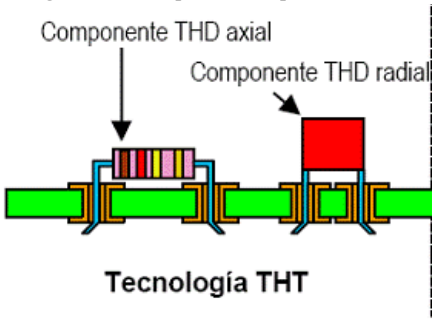


Figura 25. Componentes tipo SMD

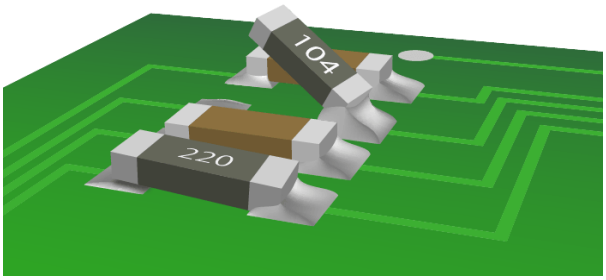
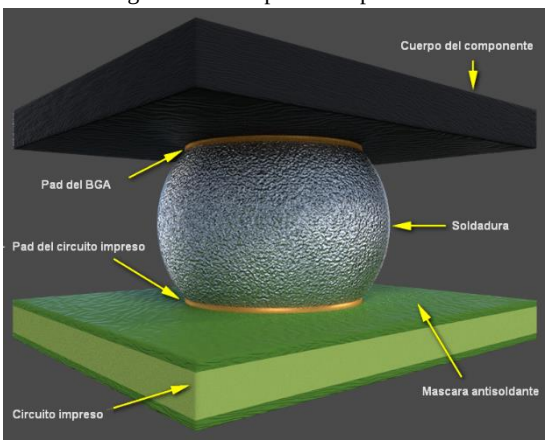
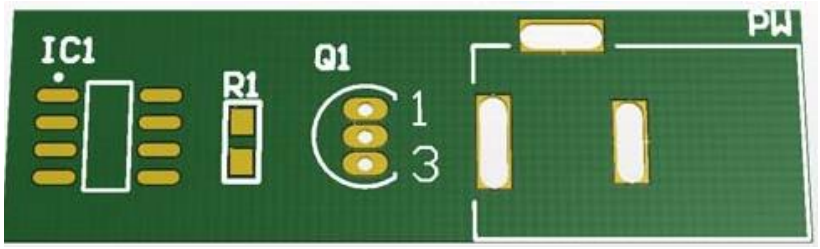


Figura 26. Componente tipo BGA



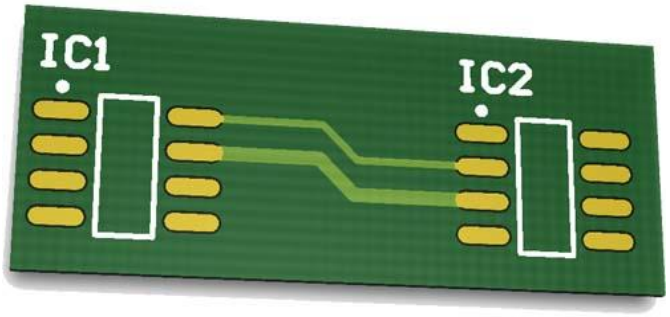
Los pads en una placa de circuito impreso son una pequeña superficie de cobre que es utilizada para soldar los componentes a la placa por medio de los pines de dichos componentes. Estos pads deben ser específicos para el tipo de componente que se va a soldar para que se ajuste a los pines de los componentes, por ello hay dos tipos de pads, los pads *Thru-hole* y los SMD (montaje de superficie). Como su nombre lo indica los pads *Thru-hole* son para componentes tipo *Thru-hole* como los de la Figura 27. y los pads SMD son para componentes tipo SMD como los de la Figura 27. Como se puede observar en la Figura 27. los pads de los componentes IC1 y R1 son pads SMD y los pads de los componentes Q1 y PW son pads *Thru-hole*.

Figura 27. Tipo de pads en placas de circuito impreso



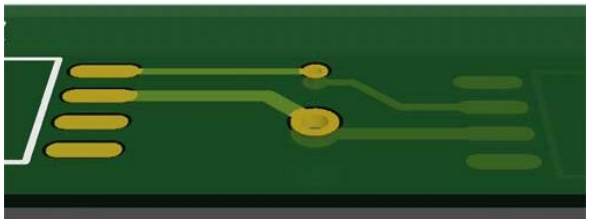
Los caminos de material conductor son conocidos como pistas o tracks, los cuales son utilizados para interconectar los pads entre componentes del circuito electrónico. En la Figura 28. se pueden observar dos pistas interconectando pads de tipo SMD del componente IC1 y el componente IC2 entre ellos.

Figura 28. Interconexión de los pads de dos componentes mediante pistas.



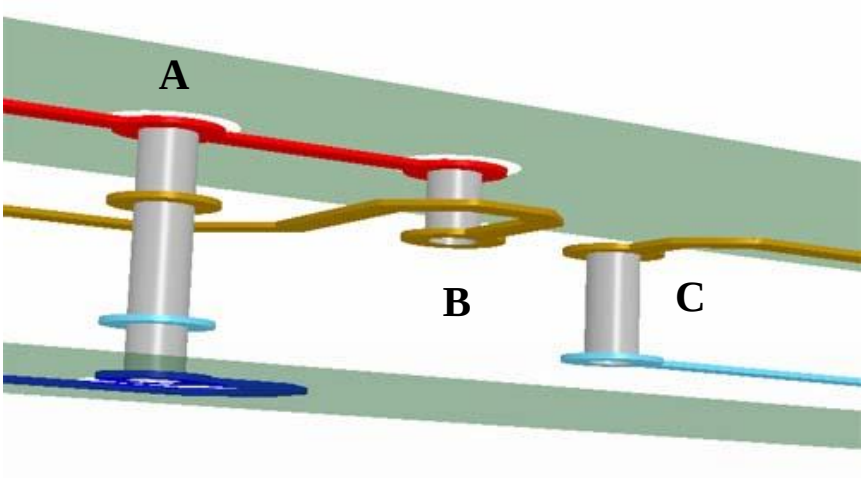
Como se explicó anteriormente, las placas de circuito impreso más comunes son las placas de doble capa. Estas placas tienen la ventaja de poder reducir el tamaño de las placas ensamblando componentes en sus dos superficies, superior e inferior. Para realizar la interconexión de componentes se utilizan vías. Una vía es una perforación metalizada con material conductor que permite la conducción eléctrica entre los tracks en caras opuestas de la placa, este tipo de vías es conocido como vía *Thru-hole*, ya que atraviesa la placa en su totalidad. En la Figura 29. se puede observar dos vías conectando las caras superior e inferior de una placa de circuito impreso.

Figura 29. Conexión entre superficie superior e inferior de un PCB.



Cuando se trabajan placas de circuito impreso de múltiples capas, es decir con capas internas que no son visibles en la superficie, se utilizan vías *Thru-hole*, ciegas y enterradas. Con estos diferentes tipos de vías es posible interconectar dos o más de las capas de la placa de circuito impreso entre ellas. Las vías ciegas son aquellas que comienzan en una capa exterior y termina en una capa interna de la placa de circuito impreso; mientras que las vías enterradas o también conocidas como vías ocultas son aquellas que no son visibles en el exterior de la placa, sino que interconectas dos capas internas de la placa. Al observar la Figura 30. de izquierda a derecha, podemos observar los tres tipos de vías, la vía del lado izquierdo es una vía *Thru-hole*, la vía del medio es una vía ciega y la vía derecha es una vía oculta o enterrada.

Figura 30. Tipos de vías, A) Thru-hole B) ciega C) oculta



La manufactura de las placas de circuito impreso puede ser mediante técnicas caseras o automatizadas. El método casero más común es diseñar el circuito sobre la superficie de cobre de la placa con tinta de marcador permanente y luego con ácido se correo el cobre que queda expuesto, de esta forma se tienen los caminos de cobre del circuito y se taladran los agujeros necesarios para el circuito. Luego se realiza la soldadura de los componentes con estaño de forma manual.

La manufactura automatizada de las placas de circuito impreso consiste en realizar un diseño de la placa en un software y por medio de una máquina CNC, la cual realiza las perforaciones y remueve el material conductor. Como resultado se obtienen las pistas que interconectan los componentes del circuito y las posiciones de cada uno de ellos. La soldadura puede realizarse de forma manual o automatizada con un proceso similar al de la fabricación de la placa de circuito impreso. En la producción en masa se utiliza los métodos automatizados, ya que el proceso de manufactura es bastante rápido.

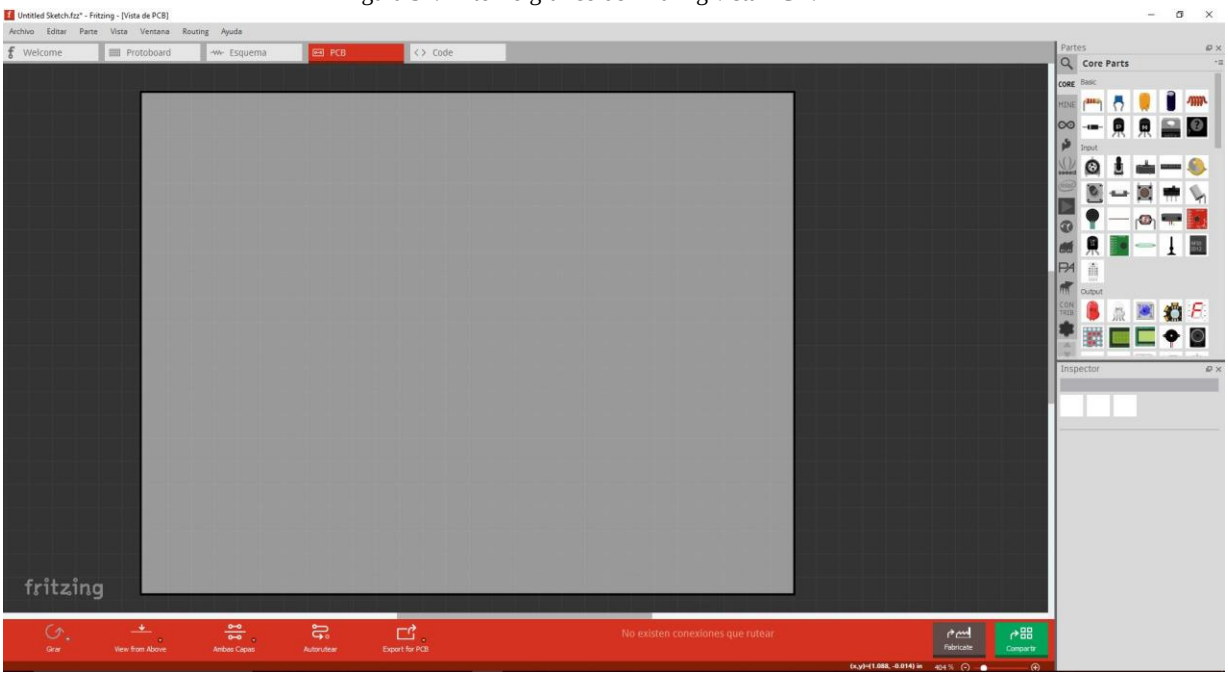
Al diseñar y manufacturar tanto placas de circuito impreso como cualquier otro tipo de componente eléctrico o electrónico se debe de tomar en cuenta las restricciones y normas estandarizadas a nivel global. Una de las normas más importantes que se debe cumplir al momento de manufacturar componentes eléctricos o electrónicos es la norma/directiva RoHS (Restriction of hazardous Substances, por sus siglas en inglés). La directiva RoHS es una restricción de algunas sustancias peligrosas en aparatos eléctricos y electrónicos. Esta consiste en 6 sustancias que son nocivas para el ser humano y se tiene prohibido su uso en componentes electrónicos.

Estas sustancias son: mercurio, plomo, cromo VI, cadmio, PBB y PBDE. Esta directiva tiene implicaciones en la industria electrónica de forma significativa. Esto se debe que al evitar que las industrias de desarrollo de componentes electrónicos utilicen estos 6 materiales los precios de algunos componentes se elevan. Además, cada cierto tiempo esta directiva se actualiza extendiéndose a más industrias de desarrollo

de componentes electrónicos. La última vez que esta se actualizó fue el 2 de enero de 2013, donde se mantuvo los 6 materiales, pero se extendió a una mayor cantidad de pequeñas y medianas empresas. Esto afecta directamente en los costos de producción de las empresas debido al cambio de materiales que deben hacer algunas de estas empresas para poder mantener su producto en el mercado. El parlamento europeo es la entidad que aprueba o no las propuestas de revisión de la directiva RoHS.

1. Fritizing. Es un software de código abierto sencillo que provee las herramientas que facilitan la fabricación de los PCB's. El software cuenta con 3 vistas disponibles para trabajar los circuitos, la primera vista es la de un protoboard, la segunda un esquemático y la tercera la del PCB. La principal facilidad de este software es la vista protoboard ya que permite al usuario traspasar el circuito físico ensamblado en el protoboard a esta vista y así generar el PCB de manera más sencilla, sin embargo, no es necesario realizar este procedimiento ya que se puede diseñar directamente el circuito impreso en la vista PCB. Frtizing cuenta con diversos componentes por defecto y adicionalmente se pueden descargar componentes elaborados por otros usuarios. (Sparkfun, 2017)

Figura 31. Entorno gráfico de Fritzing vista PCB.



S. Altium Designer

Cuando se diseñan placas de circuito impreso se debe seguir un orden de pasos para que el diseño sea más sencillo y efectivo de llevar a cabo. Lo mejor es trabajar en alguno de los softwares para diseño de placas de circuito impreso, tales como NI Multisim, Altium Designer, KiCad, Easy EDA, ExpressPCB, entre otros. Sin embargo, uno de los softwares más eficientes es Altium Designer, ya que es uno de los más completos y se

puede llevar todo el proceso de diseño en un único software donde no se requiere exportar los archivos a otros softwares para seguir las fases de diseño.

Altium Designer es un software de diseño de circuitos electrónicos con una gran variedad de funciones como diseño de PCB, diseño de diagramas esquemáticos de circuitos eléctricos, creación de componentes electrónicos, simulación de circuitos, documentación de diseño circuitos, acceso a base de datos de proveedores, diseño 3D de PCB, entre otras funciones que tiene. Dicho software fue lanzado en el año 2005 por la empresa Altium. Esta es una de las herramientas más utilizadas por empresas desarrolladoras de circuitos electrónicos y universidades en todo el mundo donde se enseña diseño de circuitos electrónicos.

El software Altium Designer tiene todas las herramientas necesarias para realizar la manufactura de una PCB pasando del diseño y el concepto a la manufactura de la misma. Las herramientas principales en este proceso cuando se comienza el diseño de una placa son el diseño esquemático del circuito, la simulación del circuito en caso sea requerida, diseño del Layout del PCB y creación de archivos Gerber para su manufactura en una CNC.

El esquemático de un circuito es una representación visual de las interconexiones entre los componentes del circuito electrónico en forma de diagrama. Con el esquemático de un circuito se busca reflejar de forma exacta el diseño del circuito electrónico mostrando componentes, fuentes de alimentación, pines de entrada y salida, y la interconexión entre componentes. En dicho diagrama debe ser posible la identificación de la información pertinente del circuito y sus componentes, como tipo, valor, tolerancia y huella, y generar una lista detallada con esta información de cada componente. El esquemático de un circuito es la base para diseñar el Layout de la placa de circuito impreso.

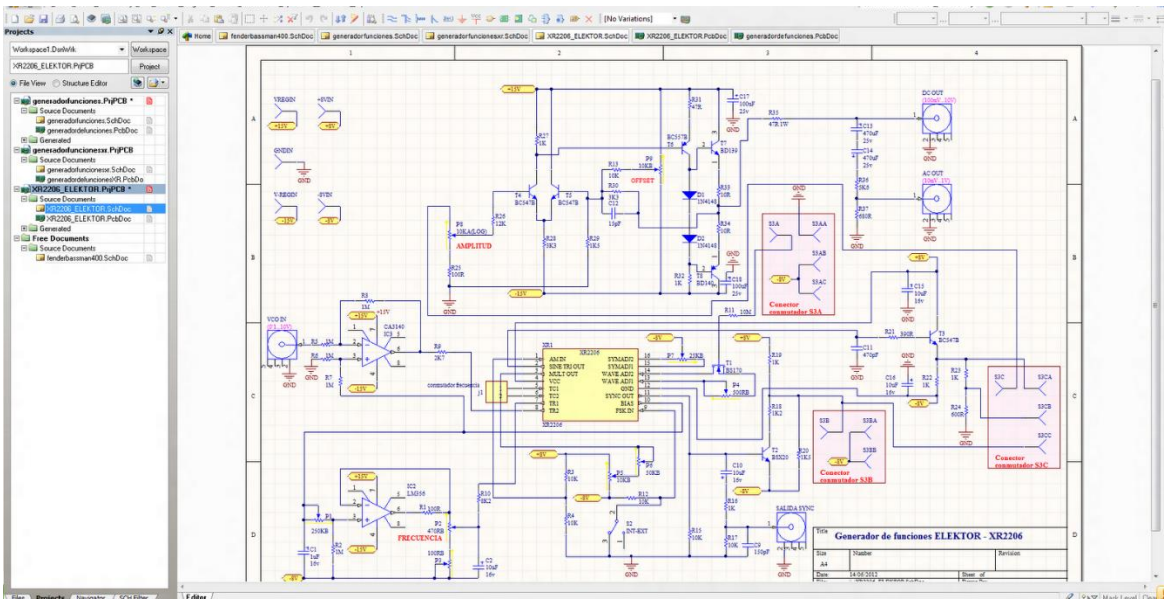
En el momento de realizar el esquemático de un circuito en el software Altium Designer se tiene acceso a una gran variedad de librerías, las cuales proporcionan los componentes electrónicos para la creación del circuito. Si en caso los componentes que se necesitan no existen en ninguna librería, Altium Designer ofrece la opción para que el usuario cree sus propias librerías. Con estas opciones el usuario puede crear cualquier tipo de circuito que desee, configurando las salidas y entradas del mismo, así como los valores, empaquetado y configuración de cada uno de los componentes a utilizar.

Al diseñar el esquemático de un circuito en un software se debe tomar consideración de normas de diseño. En el esquemático de un circuito los componentes deben estar organizados, alineados e identificados de manera que sea legible. De ser posible se deben colocar las señales en orden de izquierda a derecha, donde las entradas quedan ubicadas del lado izquierdo, mientras que las salidas del lado derecho. No se deben realizar cruces entre señales y asegurar que este lo más claro posible todas las interconexiones de los

componentes del circuito. La mejor forma de hacer que el esquemático de un circuito este ordenado es la separación de los componentes por módulos.

Al haber terminado el esquemático del circuito se tiene la posibilidad de realizar simulaciones de todo tipo. Entre los análisis que se pueden hacer están punto de operación, transiente, barrido voltaje DC, AC señales pequeñas, ruido, polos y zeros, función de transferencia, barrido de temperatura, barrido de parámetros y monte carlo. Estos análisis se configuran según las necesidades del usuario, con los cuales se pueden identificar posibles fallas del circuito o formas de optimizarlo, ya que se puede observar cómo será el comportamiento del circuito con una aproximación muy cercana a la realidad. En la Figura 32. se muestra un ejemplo del diseño del esquemático de un circuito en el software Altium Designer.

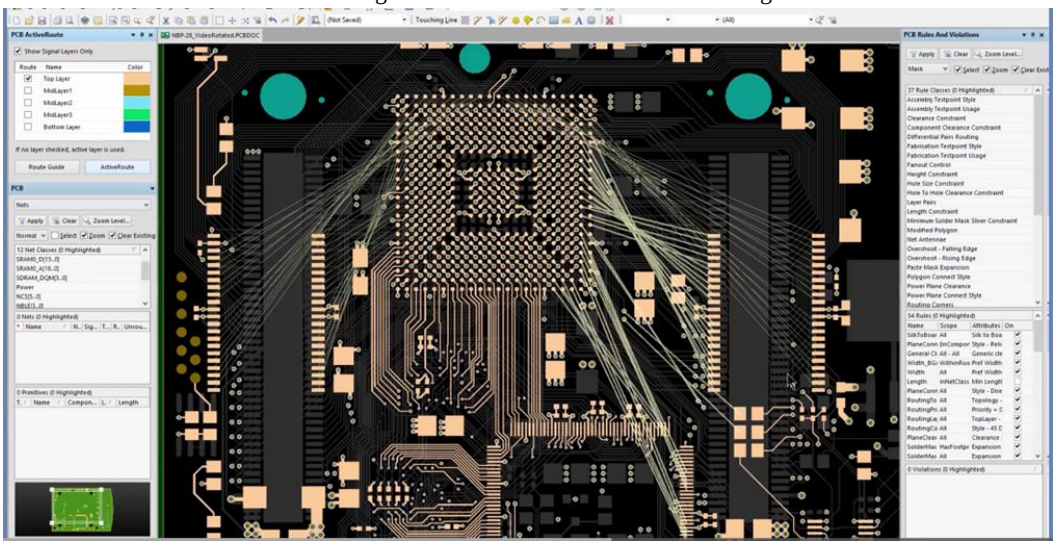
Figura 32.Esquemático de un circuito en Altium Designer



Con el diseño del esquemático de un circuito se procede a realizar el diseño del Layout de la placa de circuito impreso. El Layout de la placa de circuito es el diseño de cómo será la placa de circuito impreso físicamente. Este contiene todos los datos necesarios tales como, tamañas, posiciones y características de cada uno de los componentes del circuito, y de los pads, pistas, capas, serigrafia, máscara de soldado y las dimensiones de la misma placa.

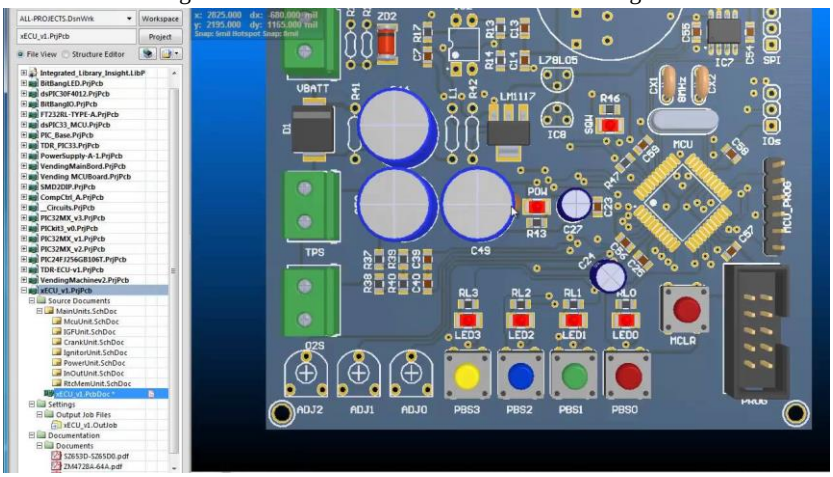
Como se tienen las dos herramientas en el mismo software únicamente se exporta el diseño esquemático a la interfaz del Layout de la placa de circuito impreso, donde el mismo software indica las conexiones de cada uno de los componentes del circuito. También tiene la opción que el software realice el diseño de la PCB automáticamente. En esta interfaz se puede seleccionar que tipo de PCB se desea diseñar, una capa o de múltiples capas, así como el tamaño de los caminos de cobre, posición de los componentes, tamaño de la PCB, etc. Además de una variedad de herramientas para que el diseño de la PCB sea lo más optimo posible. En la Figura 33. se muestra un ejemplo del diseño de una PCB en el software Altium Designer.

Figura 33. Diseño de PCB en Altium Designer



Una de las herramientas más útiles de Altium Designer al momento de diseñar placas de circuito impreso es su opción de modelo 3D de la placa que se está diseñando. Con esta opción el usuario tiene la oportunidad de observar cómo se verá la placa físicamente cuando sea construida. Esto es sumamente útil al momento de diseñar, ya que puede corroborar tamaños, formas, tipo de componentes, etc. para evitar cualquier tipo de error relacionado con aspectos físicos de la placa de circuito impreso. En la Figura 34. se puede ver un ejemplo de los diseños 3D en Altium Designer.

Figura 34. Diseño 3D de un PCB en Altium Designer



De la misma forma que con el esquemático de un circuito se deben seguir una serie de normas y pasos para que el diseño del Layout de la placa de circuito impreso este diseñado de forma adecuada. Lo primero que se debe hacer es ajustar el tamaño de la PCB al máximo que se desea diseñar. Los primeros componentes

que se deben colocar son aquellos que estén relacionados con las entradas y las salidas del circuito, ya que de esta forma se fijan en la posición deseada y se diseña alrededor de estos. Luego se deben agrupar los demás componentes en bloques lógicos, buscando minimizar el largo de las pistas para evitar efectos de resistencias y capacitancias parasitas en el circuito.

Cuando se diseñan las pistas del circuito se deben de tomar en consideración algunas características para que estas funcionen de forma óptima. La principal es la selección del ancho de la pista, ya que este debe permitir el paso de la corriente sin perjudicar la señal y sin calentarse. Por ejemplo, para una corriente de 0.5A se utiliza un ancho de al menos 0.2mm o su aproximado equivalente de 8mils.

El algoritmo para calcular el ancho de una pista de placa de circuito impreso está basado en la norma ANSI-IPC 2221 para el diseño de circuitos impresos desarrollado por el IPC (Association connecting electronics industries). Para poder realizar el cálculo del ancho de una pista se necesitan tres datos sobre la pista que se desea calcular:

- Corriente máxima que pasará por la pista, expresada en amperios
- Incremento máximo de calor que se desea permitir. Lo más común es diseñar las pistas para que nunca sobrepasen 10°C que la temperatura ambiente con la que se diseñó. Esta temperatura debe expresarse en grados Celsius.
- Grosor de la pista. El grosor de una pista se refiere a la altura de la pista. Esta medida se expresa en onzas por pie cuadrado. Donde lo más común son pistas de 1 onza por pie cuadrado, que es equivalente a 35 micras de grosor. Las otras medidas estándar para el grosor son de 2 y 3 onzas por pie cuadrado.

Al obtener los tres datos anteriormente mencionados con la dimensional adecuada se procede a utilizar las fórmulas (3) y (4), respectivamente para calcular el ancho de la pista. Cabe mencionar que los resultados obtenidos en estos cálculos están en mils, medida imperial estandarizada para diseño de placas de circuito impreso.

$$Ancho = \frac{Área}{L * 1.378} \tag{3}$$

Donde:

L es el grosor de la pista.
El área en este caso no se puede conocer, ya que no se conoce el ancho de la pista. Por ello se utiliza la formula (2) para el área, el área obtenida tiene como dimensional mils cuadrados.

$$Área = \left(\frac{I}{k_1 * \Delta T^{k_2}}\right)^{1/k_3} \tag{4}$$

Donde:

I es la corriente máxima que pasará por la pista

ΔT es la diferencia de temperatura que se va a permitir en la pista.

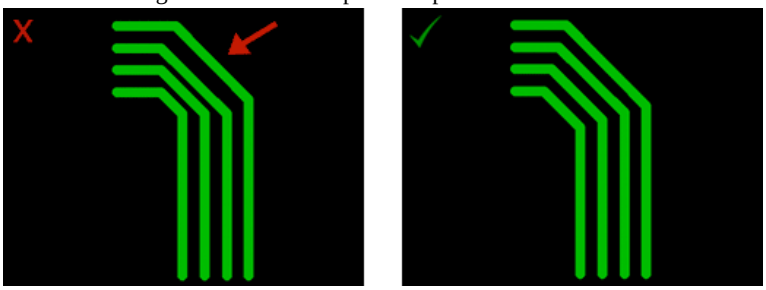
k_1, k_2 y k_3 son constantes que varían su valor si la pista se encuentra en una capa externa o interna de la placa de circuito impreso. Estos valores se pueden ver en el Cuadro 5.

Cuadro 5.Valores de k1, k2 y k3 para pistas internar y externas

	Pista Externa	Pista Interna
k_1	0.0647	0.0150
k_2	0.4281	0.5453
k_3	0.6732	0.7349

Otro punto importante al momento del diseño de pistas es la separación entre ellas. La separación depende de la tensión que contengan, en caso sean dos o más pistas paralelas estas deben mantener la distancia de separación de forma uniforme, tal y como se muestra en la Figura 35. Para tensiones entre 5 y 10 volts se debe tener una separación de al menos 0.3mm o su aproximado equivalente de 12mils.

Figura 35. Diseño de pistas en paralelo en una PCB



Al momento de trazar las pistas se debe evitar que las pistas formen ángulos de 90° en vez de ello se deben utilizar ángulos de 45° y cuando se realiza una bifurcación se debe suavizar la entrada a la pista, tal y como se muestra en la Figura 36. y 37. respectivamente. Esto se debe a que la curva de 90° genera un efecto de punta, un efecto de punta es un efecto físico donde se produce una acumulación de energía; esto significa que al haber una acumulación de energía la temperatura aumenta. Para señales que pasan los GHz la impedancia es afectada y puede generar interferencia para estas señales. Adicionalmente, al momento de realizar la manufactura con una máquina CNC un ángulo de 90° es más difícil de maquinar que un ángulo de 45°.

Figura 36. Representación de cómo trazar pistas al realizar cambio de dirección de 90°

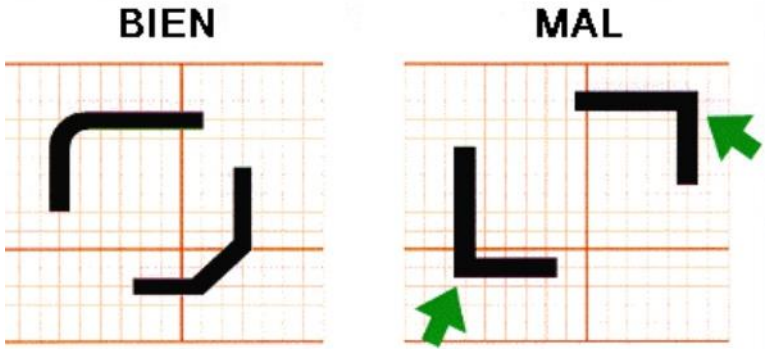
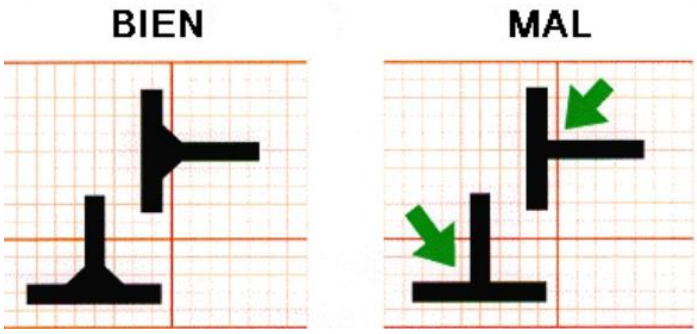


Figura 37. Representación de cómo suavizar la unión perpendicular de dos pistas



Por último, se procede a exportar el diseño de la PCB a un archivo Gerber o también conocido como código G. Este es un lenguaje de bajo nivel con el cual se programan las máquinas CNC, dicho lenguaje de programación está basado en programación vectorial que describe acciones simples y entidades geométricas sencillas con los respectivos parámetros de maquinado que debe llevar a cabo la máquina CNC. Al descargar el programa del código G del diseño de la PCB a una máquina CNC de placas PCB, esta realiza la manufactura de la PCB.

T. Inventor

Es la herramienta ideal para diseño de detalle de dispositivos mecánicos de alta ingeniería, ya que logra validar los prototipos digitales por medio de análisis de elementos finitos y simulación dinámica. Optimiza la etapa de diseño de producto, validando el correcto funcionamiento de los elementos, y permitiendo la parametrización de variables de validación, para la mejora constante.También permite la actualización automática del modelo 3D, según la dependencia de este a las variables asignadas. Permite la extracción de información de ingeniería por medio de planos detallados, facilitando a los departamentos de diseño la comunicación con los departamentos de manufactura. (AutoDesk,2017).

Figura 38.Logotipo AutoDesk Inventor.

U. Cinemática

La cinemática es el estudio del movimiento sin considerar las causas de dicho movimiento, sino únicamente la geometría de dicho movimiento. (Hibbeler, 2004) Es de mucha utilidad obtener la trayectoria de los sensores, para posteriormente permitirle a los entrenadores realizar un análisis biomecánico con mediciones reales del movimiento con información relevante. Para obtener una trayectoria con respecto a un marco global será necesario conocer la pose del sensor en todo tiempo, por lo que se planteará la cinemática el sensor desde un punto de vista de robótica. Considerando todas las rotaciones locales y globales del sensor.

1. Cinemática de masas puntuales. Una masa puntual es una suposición del cuerpo de estudio en la cual dicho cuerpo se modela como un punto sin dimensiones, y esto permite muchos de los cálculos que se necesarios para describir una trayectoria. El sensor inercial de bajo costo servirá para modelar una parte del cuerpo, y para ello se necesita saber la pose de dicho sensor. La tarea de obtener la orientación y posición de dicho sensor es posible modelando el sensor como una masa puntual.

2. Conceptos de cinemática. La cinemática, como se mencionó anteriormente, es el estudio de la geometría de un movimiento. Esto implica obtener las aceleraciones, velocidades y posiciones en el tiempo a lo largo de toda la trayectoria a evaluar. La velocidad es la razón de cambio de la posición con respecto al tiempo, como se plantea en la ecuación 2. La aceleración es la derivada de la velocidad con respecto al tiempo y por lo tanto la doble derivada de la posición con respecto al tiempo, como se puede ver en la ecuación 5. Una Unidad de Medición Inercial puede medir aceleraciones. Para obtener el resto de las funciones cinemáticas se deben utilizar las siguientes ecuaciones, y para resolverlas se deberá integrar numéricamente.

$$a(t) = \frac{dv(t)}{dt} = \frac{d^2x(t)}{dt^2} \tag{5}$$

$$v(t) = \frac{dx(t)}{dt} = v_0 + \int a\,(t)dt \tag{6}$$

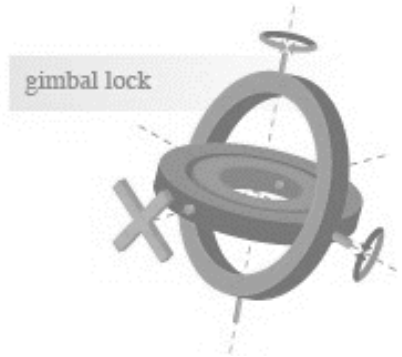
$$x(t) = x_0 + \int v\,(t)dt = x_0 + \int (v_0 + \int a(t)\,dt)\,dt \tag{7}$$

3. Cuaterniones. Los cuaterniones en un principio fueron diseñados para crear una extensión de los números complejos a dimensiones superiores. (Rodríguez, 2012) Pero en el contexto de la robótica y la cinemática es una manera muy eficiente de representar las matrices de rotación. Este se compone de un vector de cuatro componentes, por lo tanto, el álgebra de cuaterniones es muy simple. Es posible realizar rotaciones a vectores por medio de cuaterniones de una manera muy simple, y con pocas operaciones aritméticas. El hecho de tener un cuarto componente permite representar rotaciones dentro de una esfera de radio 1 (Favieri, 2008), lo que tiene una gran ventaja sobre los ángulos de Euler, los cuales tienen singularidades, como se discutirá a continuación. En la ecuación 8 se muestra la forma general de un cuaternión.

$$q = w + x\hat{\imath} + y\hat{\jmath} + z\hat{k} \tag{8}$$

a. Singularidades de rotación. La representación de las rotaciones por medio de un trío de ángulos es fácil de visualizar, pero trae consigo singularidades. Estas singularidades o bloqueo de ejes ocurren cuando se alinean el eje de rotación del segundo término con cualquiera de los otros dos ejes. Es aquí donde se pierde la orientación en dicho eje, y es imposible obtener de nuevo la orientación del objeto estudiado. Para visualizarlo un poco más fácilmente, el bloqueo ocurre cuando la rotación sobre el eje z de la nave es ±90º, de manera que cuando dos ejes de rotación se alinean, no se puede saber con respecto a cuál de los dos ejes gira el objeto estudiado. Este problema se evita con la representación por medio de cuaterniones.

Figura 39.Bloqueo de ejes (Gimbal lock)



(Gimbal Systems, 2017)

V. Cálculo de cinemática

La Unidad de Medición Inercial tiene muchas aplicaciones, desde estabilizar dispositivos, obtener orientaciones, desplazamientos, tiempos de reacción, entre otros. En el caso del cálculo de la trayectoria es necesario obtener la orientación, para lo cual se utiliza los acelerómetros, giroscopios y magnetómetros. Posteriormente se rotan las mediciones de los acelerómetros por medio de la orientación obtenida, para luego por medio de integración numérica obtener la posición efectiva.

1. Orientación del sistema. La orientación del sistema es de suma importancia para obtener una posición correcta, ya que esta repercute en la trayectoria que se obtenga. Como se explicó anteriormente se necesita utilizar una representación de la orientación que sea representativa y que no importe las rotaciones que realice el sensor, no se perderá la orientación. Es por esto que es necesario representar la orientación por medio de cuaterniones. La orientación del sistema se utiliza para saber en todo momento el sentido del movimiento que se cuantifica con los acelerómetros. Es decir, los acelerómetros miden constantemente la aceleración que experimentan, pero para darle una interpretación en el marco global se debe saber la orientación de dichas mediciones.

a. Algoritmos para la obtención de la orientación. Mediante la utilización una Unidad de Medición Inercial es posible obtener las mediciones de aceleración, velocidad angular y campo magnético. Estas tres mediciones se pueden fusionar para obtener una orientación efectiva. Esto para un análisis aplicado a ingeniería aeroespacial, a robótica, navegación o análisis biomecánico. Los algoritmos que permiten realizar esta función de sensores son llamados AHRS, por sus siglas en inglés que significan: Sistemas de Referencia de Altitud y Dirección. Al utilizar giroscopios MEMS, por sus siglas en inglés, como una alternativa a los giroscopios de precisión se obtendrá un error acumulado en la orientación, pero los acelerómetros y magnetómetros, que pueden medir por medio del campo magnético y la dirección de la gravedad una referencia absoluta de la orientación. (Madgwick, 2010) El objetivo de los algoritmos anteriormente mencionados es obtener una orientación absoluta del sensor fusionando las mediciones de los giroscopios, acelerómetros y magnetómetros. Existen muchas maneras de abordar este problema, pero las más accesibles por cuestiones de eficiencia computacional son filtros de fusión que tienen enfoque en los filtros Kalman, lo cual permite que se analicen las mediciones desde una perspectiva de observador. Estos algoritmos tienen mejor resultado con una frecuencia alta de medición. En los anexos se encuentran los algoritmos de Madgwick y Mahony que funcionan mediante la fusión de sensores anteriormente mencionada.

2. Rotación de acelerómetros. Las mediciones de los acelerómetros están orientadas en su pose local, pero si se integra estas aceleraciones se obtienen velocidades y posiciones con respecto a un marco local, y si el sistema sufre de un cambio en su orientación, cambia el marco de referencia. Es por esto que es de suma importancia a medida que la pose cambia con respecto a la pose inicial, rotar también las mediciones de los acelerómetros para integrar y obtener velocidades y posiciones verídicas con respecto a un marco global. Si se dispone de la orientación representada por medio de cuaterniones, es necesario rotar el vector para ubicarlo en la dirección que se ubica el sensor. Este cálculo es posible por medio de la ecuación 6. Interpretando la ecuación, se multiplica el cuaternion por el vector en forma de cuaternión, es decir, colocando la primera componente como cero, y esto por el cuaternión conjugado. Esto da como resultado el vector rotado, es decir, conserva su magnitud, pero la dirección será ahora en el sentido del cuaternión unitario.

$$V_{rotado} = q * q(V) * q^{-1} \quad (9)$$

3. Resolución de la ecuación diferencial. Para obtener todas las ecuaciones de cinemática a partir de la medición de los acelerómetros es necesario integrar numéricamente. Existen varios algoritmos de integración, para solucionar dicha ecuación diferencial. Entre los algoritmos más utilizado y con mejores resultados está el más simple que es Euler, Trapecios y Simpson 1/3, como se muestra a continuación en las ecuaciones 10, 11 y 12.

$$x_i = x_{i-1} + v_i * \Delta t \quad (10)$$

$$x_i = x_{i-1} + \frac{1}{2}(v_i + v_{i-1}) * \Delta t \quad (11)$$

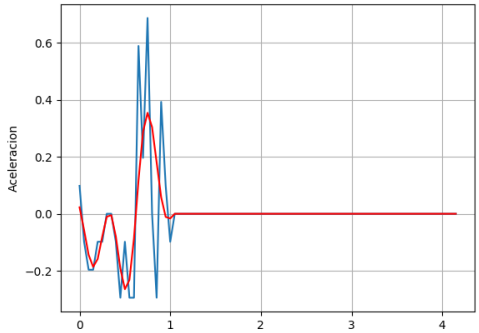
$$x_i = x_{i-1} + \frac{1}{3}(v_i + 4 * v_{i-1} + v_{i-2}) * \Delta t \quad (12)$$

W. Acumulación de error

Al obtener la trayectoria del sensor siguiendo el proceso anteriormente descrito, y utilizando alguno de los algoritmos indicados, comúnmente existe error entre la trayectoria real y la obtenida matemáticamente. Esto debido a ruido en los instrumentos de medición, pérdida de orientación en los mismos y desfases en la integración. A continuación, se analizará cada uno de estos posibles errores en la trayectoria.

1. Ruido. El ruido es un error sistemático de alta frecuencia, que tiene como resultado mediciones inestables. Este error es muy fácil de identificar, ya que, realizando un análisis frecuencial, son armónicos o grupos de frecuencias que están por encima de la frecuencia de Nyquist que caracteriza el sistema. Es decir, son cambios en las mediciones que se dan más rápido de lo que cambia la variable estudiada. Realizando así mismo el análisis en el tiempo, es una medición que cambia muy rápido en el tiempo, como se puede observar en la Figura 40.

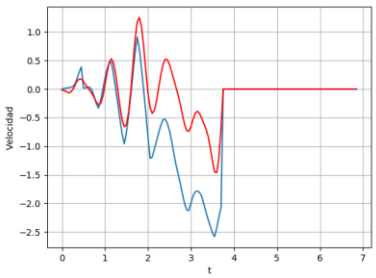
Figura 40. Medición de acelerómetro ideal vs real.



2. Pérdida de la orientación. La pérdida de la orientación es un error que se da cuando se tiene la medición de la aceleración de un sistema, pero la orientación que se tiene es aún de la iteración pasada. Uno de los inconvenientes al calcular la orientación a partir de las mismas mediciones que servirán para obtener la trayectoria es que la orientación debe ser una medición que se calcula con una mayor frecuencia, ya que esta será la que le de la dirección al vector de aceleraciones. El otro inconveniente de este sistema es el error dinámico de los sensores, es decir, la diferencia entre la cantidad indicada por el sensor en un período de tiempo y el verdadero valor del parámetro de medición. Esta incertidumbre de la medición repercute en pérdida de la orientación en un período corto de tiempo.

3. Desfases. Los sensores tienen un error dinámico intrínseco, como se planteó anteriormente. Al integrar las mediciones, se integra también el error junto con la medición. Parte del error que se genera al integrar se puede observar como desfases en función integrada, es decir, que la función tiene una longitud mucho mayor a la real. El gran inconveniente de esto es que no es un error sistemático, lo que indica que no será el mismo error en los tres ejes, ni en todas las mediciones, por lo que se debe encontrar una solución genérica para mitigar estos desfases y de esta manera obtener una trayectoria sin dichos desfases. A continuación, en la Figura 41. se muestra una señal con un desfase contra la variable real del sistema.

Figura 41. Aceleración con desfase vs. aceleración sin desfase.



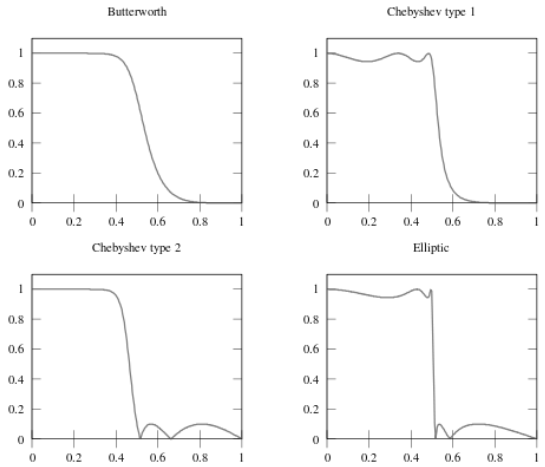
X. Mitigación de errores

Como se planteó anteriormente, existen muchos tipos de error, que pueden tener como resultado una trayectoria poco acertada al realizar los cálculos. Es sumamente complicado definir un método que aplique para todos los posibles movimientos de un sensor inercial. Existen muchas soluciones a los errores anteriormente descritos, a continuación, se menciona una de ellas.

1. Filtros digitales. Los filtros digitales son algoritmos que procesan una señal digital, a la cuál, dependiendo sus características, pueden variar su amplitud, frecuencia o fase. (Hamming, 1989) Los filtros digitales pueden eliminar contenido frecuencial, y dejar pasar únicamente las frecuencias que son de interés. Esto quiere decir que primero se debe hacer un análisis frecuencial de una señal, para saber qué frecuencias son las que se debe eliminar. Esto puede ser de suma utilidad si se logran identificar los armónicos de las frecuencias que son información y así mismo, identificar las frecuencias que únicamente son ruido, o

información que se verá reflejada como desfases al integrar. Existen muchos tipos de filtros, como se muestra en la siguiente figura.

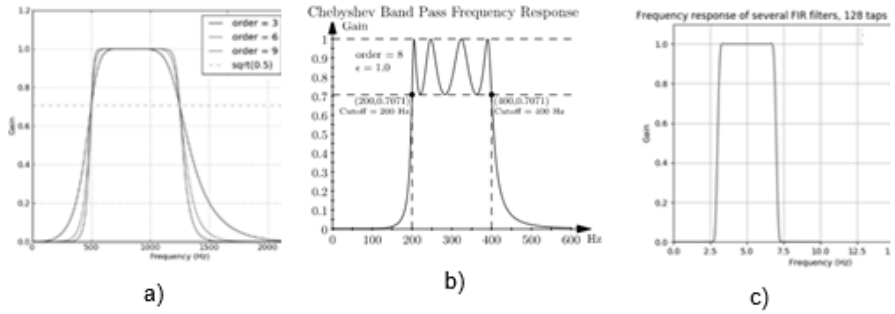
Figura 42. Respuesta en frecuencia de algunos filtros digitales



a. Tipos de filtros. Existen muchos tipos de filtros, con características distintas. Existen filtros que eliminan cierto contenido frecuencial y dejan pasar otro. Entre los filtros con esta característica se encuentran:

- Filtro pasa bajas: Estos filtros dejan pasar únicamente el contenido de frecuencias por debajo de una frecuencia establecida, la cuál es la frecuencia de corte.
- Filtro pasa altas: Estos filtros dejan pasar únicamente el contenido de frecuencias por encima de su frecuencia de corte.
- Filtro pasa banda: Estos filtros básicamente son la unión de un filtro pasa bajas y uno pasa altas. Esto quiere decir que dejarán pasar una serie de frecuencias se encuentren por encima de la frecuencia de corte de la pasa alta y a la vez por debajo de la frecuencia de corte del filtro pasa bajas. El filtro pasa banda permite seleccionar frecuencias específicas de una señal. A continuación, se mostrarán tres filtros digitales y se discutirá las características principales de cada uno.

Figura 43. Filtros Pasabanda a) Butterworth b) Chebychev c) FIR (Firwin)



Como se puede observar, los filtros pasabanda permiten pasar un conjunto de frecuencias, pero cada filtro tiene características distintas. El filtro Butterworth deja pasar un conjunto de frecuencias mucho más amplio, y las frecuencias están en el orden de 10^3. Mientras que el filtro Chebychev tiene una banda mucho más angosta, con frecuencias en el orden de 10^2. Por último, se tiene el filtro FIR (Firwin), que permite el paso de frecuencias en el orden de 10^0.

b. Análisis frecuencial. Para analizar señales digitales y ver que filtro es el que mejor se ajusta a la necesidad es necesario ver la señal desde el punto de vista de las frecuencias que la componen. Para ello se tiene la Transformada Rápida de Fourier (TRF). Este algoritmo permite ver las frecuencias que componen una señal.

Y. Cuantificación del error

La veracidad de la trayectoria se puede cuantificar de varias maneras, entre ellas se encuentran la magnitud y la forma de la trayectoria. Para la magnitud se sigue un proceso muy sencillo, y es medir el desplazamiento total en forma vectorial, y posteriormente se obtiene la magnitud del vector que será la magnitud de la trayectoria. A esta magnitud es sumamente sencillo calcularle el porcentaje de error si se conoce previamente la totalidad del desplazamiento del sensor. Para cuantificar la forma de la trayectoria se utiliza un método un tanto más complicado, este es calcular una regresión lineal del conjunto de coordenadas x, y. A esta regresión lineal se le obtiene el coeficiente de correlación, y será una información muy relevante, ya que indica que tan rectilínea es la trayectoria. Un coeficiente de correlación cercano a 1 indica que la trayectoria fue casi rectilínea. Las ecuaciones a continuación indican cómo calcular dichos indicadores de error.

$$|\text{Trayectoria}| = \sqrt{x^2 + y^2 + z^2} \tag{12}$$

$$\rho_{X,Y} = \frac{\sigma_{XY}}{\sigma_X * \sigma_Y} = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{(n-1)s_x * s_y} \tag{13}$$

V. METODOLOGÍA

A. PCB sensor inercial

El módulo de Diseño de las placas de circuito impreso para sensores inerciales de bajo costo del proyecto de Ciencia Aplicada al Deporte se separó en cuatro fases fundamentales para su desarrollo. De esta forma se desarrolló cada una de las fases con sus respectivas sub-fases para su elaboración de la siguiente manera:

1. Prototipo
    a. Construcción de réplica de Arduino en protoboard
    b. Descarga de bootloader y programas al ATMEGA328P-PU del circuito en protoboard
    c. Pruebas con la IMU-BNO055 y Xbee
2. Diseño de placa de circuito impreso
    a. Diseño del esquemático del circuito en el software Altium Designer
    b. Cálculo de pistas de PCB
    c. Diseño de PCB en el software Altium Designer
3. Manufactura
    a. Fresado de placa de circuito impreso
    b. Soldado de placa de circuito impreso
    c. Descarga de bootloader al ATMEGA328P-AU de la placa de circuito impreso
4. Pruebas de la placa de circuito impreso

Este proceso se llevó acabo de forma cronológico, ya que cada una de las fases depende de la anterior y de una investigación previa de temas específicos de cada una de las fases. Para lograr esto se realizó un cronograma de actividades que se muestra en el Cuadro 6 , cabe mencionar que paralelamente a este proceso se estuvo desarrollando el trabajo escrito.

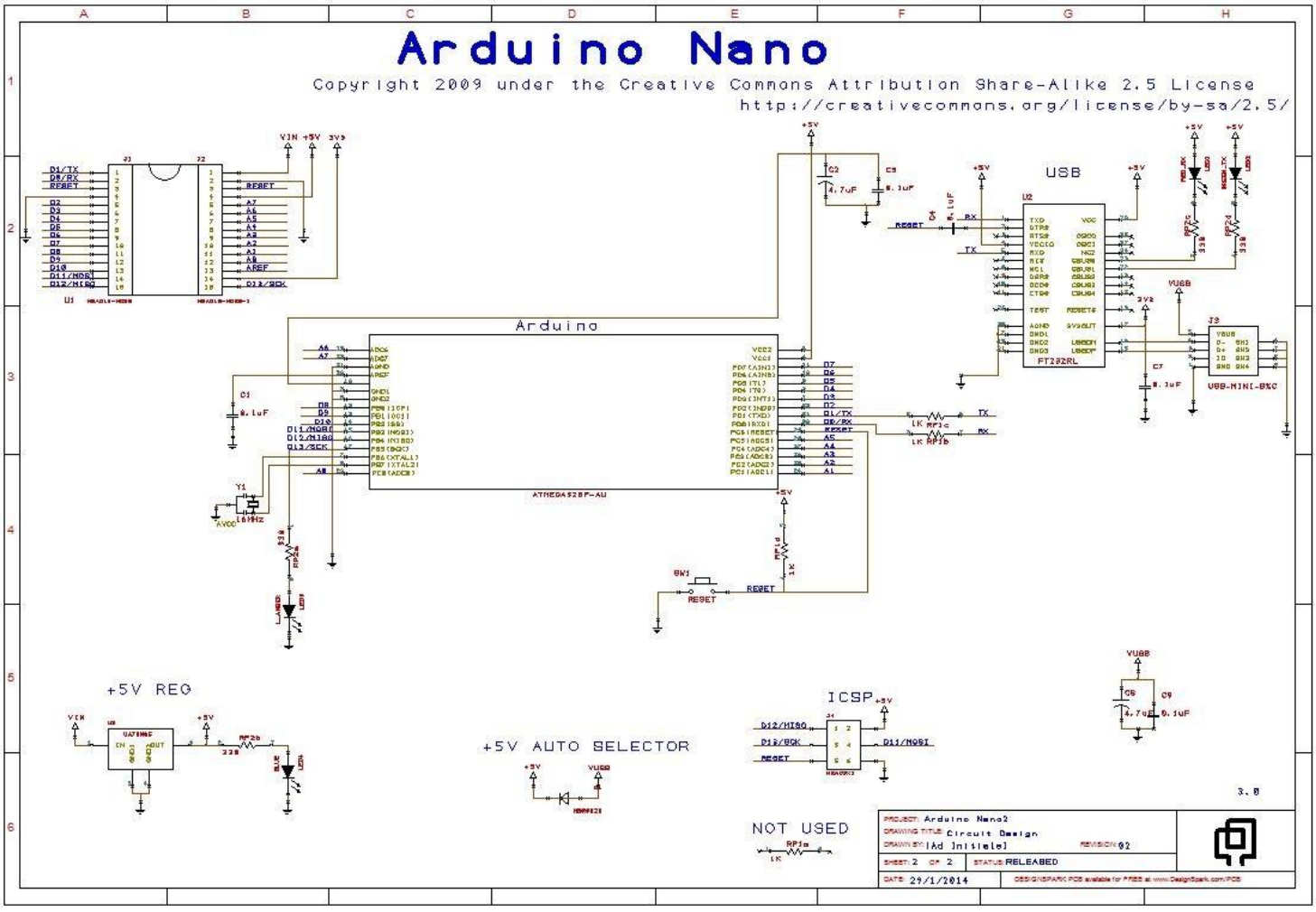
Cuadro 6. Actividades para el desarrollo del proyecto.

Listado de Actividades			
No.	Actividad	Fecha inicio	Fecha finalización
1	Investigación general sobre los temas esenciales del proyecto	03/07/2017	31/07/2017
2	Desarrollo, pruebas y validación de prototipo	24/07/2017	07/08/2017
3	Diseño del diagrama esquemático de la placa de circuito impreso en Altium Designer	07/08/2017	21/08/2017
4	Diseño de PCB de la placa de circuito impreso en Altium Designer	21/08/2017	04/09/2017
5	Validación del diseño realizado	04/09/2017	11/09/2017
6	Manufactura de la placa de circuito impreso	11/09/2017	25/09/2017
7	Pruebas, validación y correcciones a la placa de circuito impreso. (según sea necesario)	25/09/207	02/10/2017
8	Pruebas y resultados finales de la placa de circuito impreso	02/10/2017	06/10/2017
9	Entrega final y presentación final del proyecto	09/10/2017	09/11/2017

Cada una de las fases anteriormente mencionadas por haberse realizado de forma cronológica los resultados se presentará en cada una de las fases para facilitar al lector la comprensión y análisis de cada una de las fases.

1. Réplica de Arduino en protoboard. La plataforma Arduino al ser open-source se tiene acceso a la documentación necesaria para poder replicarlo. Para este proyecto, en el cual se implementa Arduino, se realizó un prototipo en protoboard para poder realizar las pruebas necesarias y corroborar que efectivamente replicar el Arduino es la opción más viable para el proyecto. Para lograr esto se estudió el esquemático del circuito del Arduino NANO, que se puede observar en la Figura 44. Luego de haber estudiado dicho esquemático se pudo concluir que materiales son los mínimos necesarios para el óptimo funcionamiento del Arduino para poder replicarlo. Estos materiales se pueden ver en el Cuadro 7.

Figura 44. Esquemático Arduino Nano.

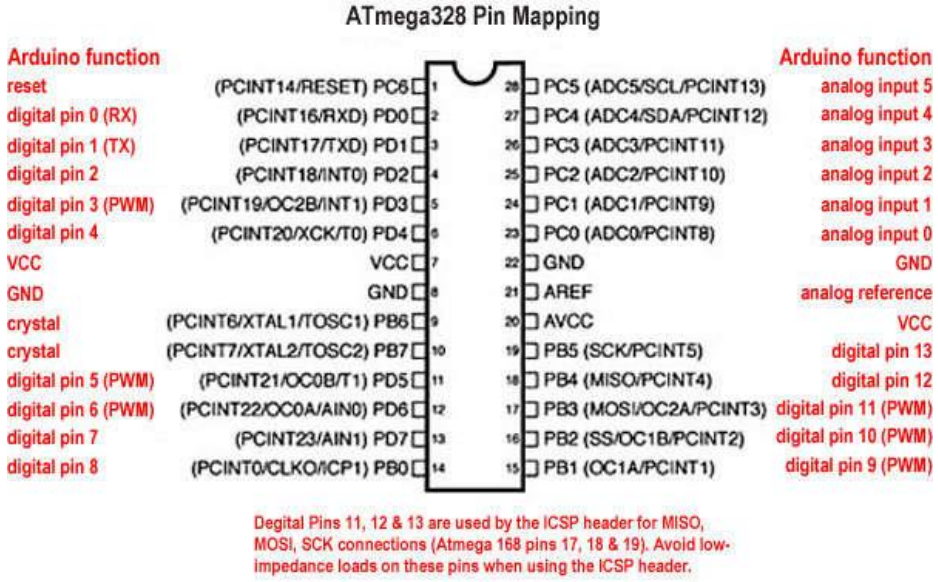


Cuadro 7. Materiales para replicar Arduino Nano.

Cantidad	Materiales
1	ATMEGA328P-PU
1	Regulador de voltaje con salida de 5V (7805)
2	LED
2	Resistencia de 220 Ω
2	Resistencia de 10KΩ
2	Capacitor de 10uF
2	Capacitor de 22pF
1	Cristal oscilador de 16MHz
1	Botón

El ATMEGA328P-PU es el microcontrolador que utiliza el Arduino. En la Figura 45. se puede observar el mapa de pines del microcontrolador y en el Cuadro 7. se especifica que se conecta en cada uno de los pines del microcontrolador ATMEGA328P-PU. Cabe mencionar que los pines que no están especificados en el Cuadro 8. no es necesario conectarlos para que la réplica del Arduino funcione. El PIN 13 no es indispensable conectarlo, pero se utiliza como referencia para saber si el circuito está funcionando correctamente y el PIN 21 tampoco es indispensable, este únicamente es necesario conectarlo si se utilizará la función ADC del microcontrolador ATMEGA328P-PU. En la Figura 45. se puede observar el esquemático del circuito completo para poder realizar una réplica de un Arduino con los componentes básicos para que este funcione.

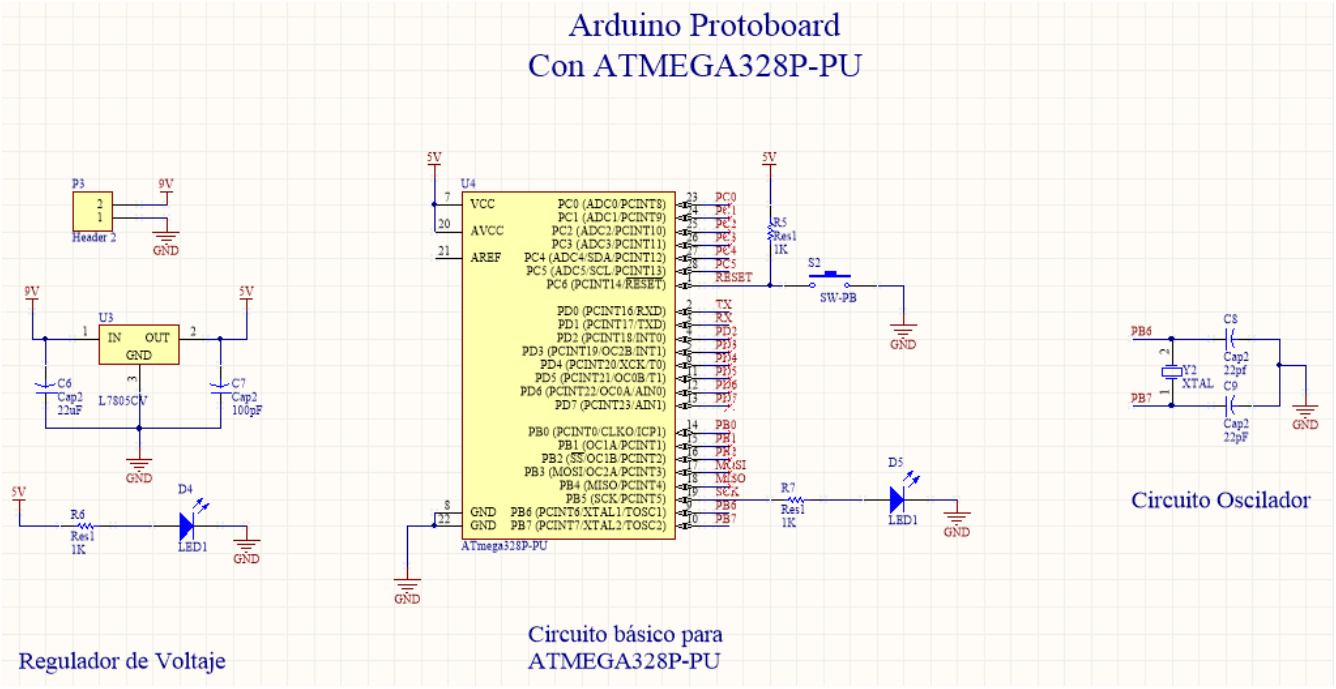
Figura 45. Mapa de pines del microcontrolador ATMEGA328



Cuadro 8. Listado de pines, identificación, función y conexión de cada uno

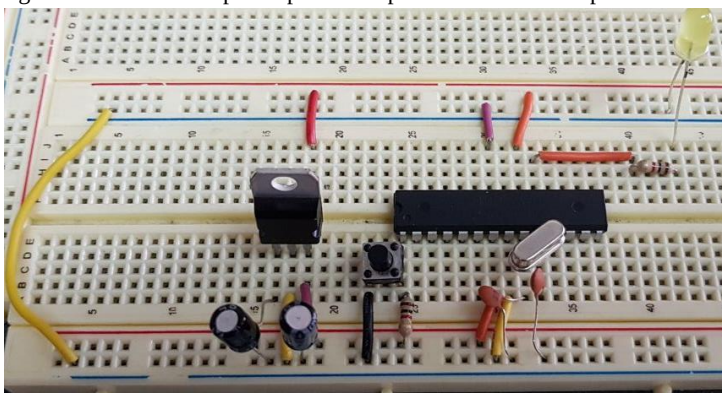
PIN	Identificación de PIN	Función	Conexión
1	RESET	Reset del ATMEGA328P-PU	Botón de forma Pull-up
7	VCC	Voltaje positivo para alimentar el ATMEGA328P-PU	Salida de 5V del Regulador de voltaje (7805)
8	GND	Tierra	Tierra del circuito
9	TOSC1	Cristal oscilador	Un extremo del cristal oscilador
10	TOSC2	Cristal oscilador	Un extremo del cristal oscilador
13	PD7	Pin digital	Resistencia con LED
20	AVCC	Voltaje positivo para el ADC del ATMEGA328P-PU	Salida de 5V del regulador de voltaje (7805)
21	AREF	Voltaje de referencia	Salida de 5V del regulador de voltaje (7805)
22	GND	Tierra	Tierra del circuito

Figura 46. Esquemático del circuito con los componentes básicos para un ATMEGA328P



En la Figura 46. se puede observar el circuito terminado de la réplica del Arduino en Protoboard. El microcontrolador utilizado en este circuito tenía un programa previamente cargado que únicamente encendía el LED del PIN 13 por 2 segundos y luego lo dejaba apagado 1 segundo. De esta forma se realizó la primera prueba al prototipo para verificar que el diseño si funciona de la forma esperada.

Figura 47. Circuito del prototipo de la réplica del Arduino en protoboard



2. Bootloader y programas ATMEGA328P-PU. El microcontrolador ATMEGA328P al ser adquirido de fábrica no trae ningún tipo de programa previamente instalado, por lo que se utiliza como un microcontrolador cualquiera; es decir, no puede programarse con el IDE y el lenguaje de programación de Arduino. Para poder utilizar el ATMEGA328P como un Arduino es necesario descargarle un programa llamado: Bootloader.

Una de las grandes ventajas que Arduino sea un software *open-source*, es que el código del Bootloader se puede encontrar fácilmente en la página oficial de Arduino. El programa Bootloader esta adjunta en la sección de Anexos de este trabajo. Para poder cargar el Bootloader a un ATMEGA328P se requiere realizar el circuito de la Figura 47. y utilizar un Arduino para alimentar y descargarlo al ATMEGA328P. En el Cuadro 9. se puede observar los pines que se conectan de un Arduino UNO a un ATMEGA328P para realizar la descarga del Bootloader. En la Figura 47. se puede observar el diagrama pinout del Arduino UNO y en la Figura 48. la conexión del Arduino UNO con el circuito del microcontrolador ATMEGA328P de forma gráfica.

Figura 48. Circuito básico para poder cargar el bootloader a un ATMEGA328

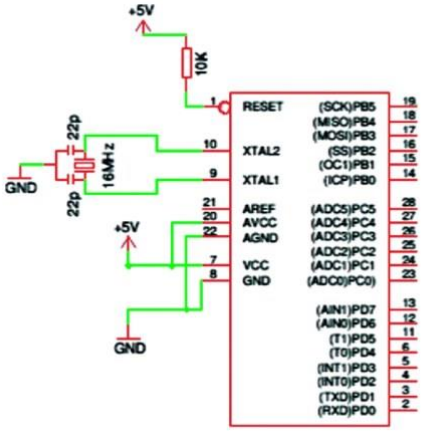


Figura 49. PinOut de Arduino UNO

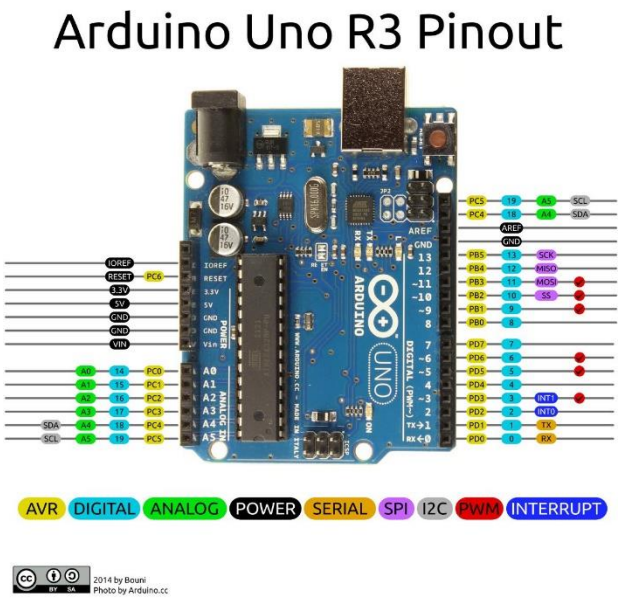
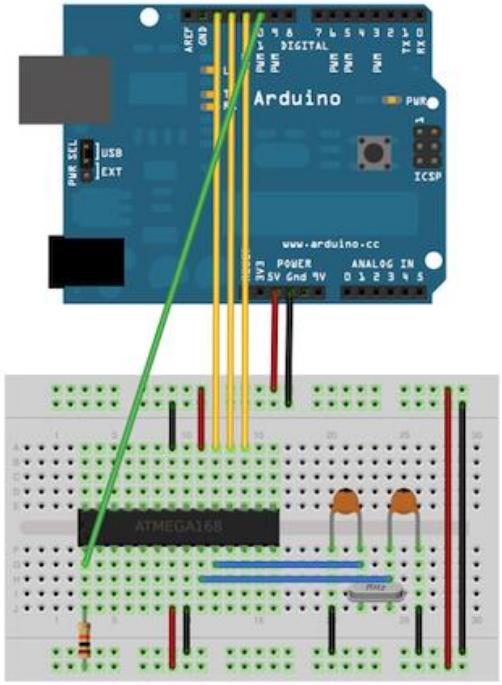


Figura 50. Diagrama gráfico de conexiones para la descarga del bootloader



Cuadro 9. Conexión de pines para la descarga del bootloader

Arduino UNO		ATMEGA328P	
No. PIN	Identificación PIN	No. PIN	Identificación PIN
10	PB2/SS	1	RESET
11	PB3/MOSI	17	MOSI
12	PB4/MISO	18	MISO
13	PB5/SCK	19	SCK
-	5V	7 y 20	5V
-	GND	8 y 22	GND

Hay varias formas de realizar el proceso de Bootloader en un microcontrolador ATMEGA. En este caso se utilizó el que recomienda Arduino en su página oficial, https://www.arduino.cc/en/Tutorial/ArduinoToBreadboard. Este método se conoce como programación ArduinoISP y los pasos para hacer este bootloader son los siguientes:

1. Abrir el programa ArduinoISP. Este se encuentra en el IDE de Arduino en: Archivo/Ejemplo/ArduinoISP, esto se puede ver en la Figura 49.
2. Configurar el Arduino que se utilizará para realizar el bootloader. Esto es en el IDE de Arduino en Herramientas/Placa. En esta opción se debe seleccionar el Arduino que se va a utilizar. Eso se puede ver en la Figura 51. en rojo.
3. Realizar el cableado de la Figura 52.
4. Seleccionar “Arduino as ISP” en Herramientas/Programador. Eso se puede ver en la Figura 52.en rojo.
5. Seleccionar “Quemar Bootloader” en Herramientas. Eso se puede ver en la Figura 52. en azul.

Figura 51. Proceso de descarga de bootloader paso 1

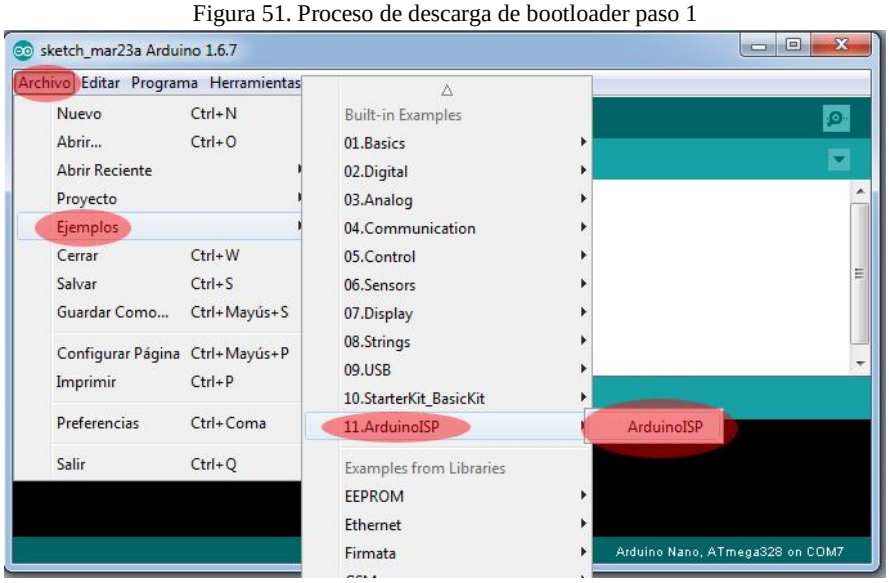
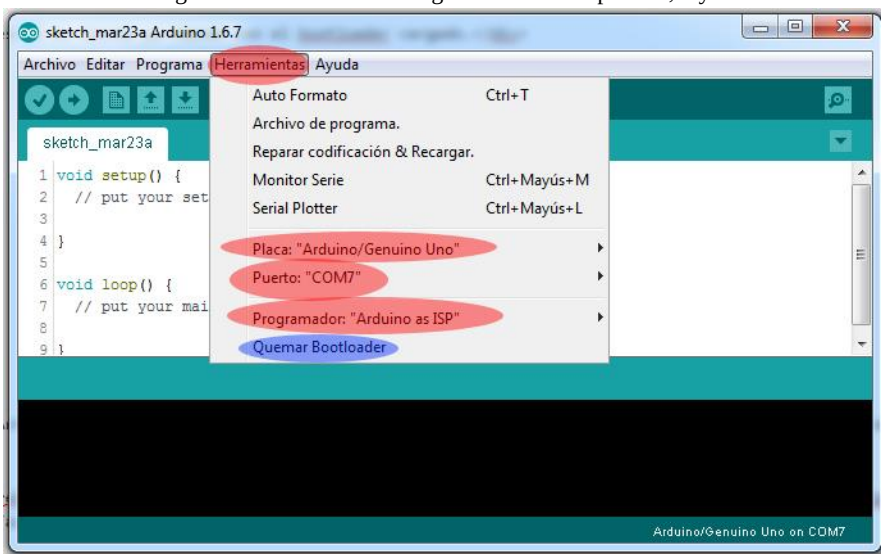


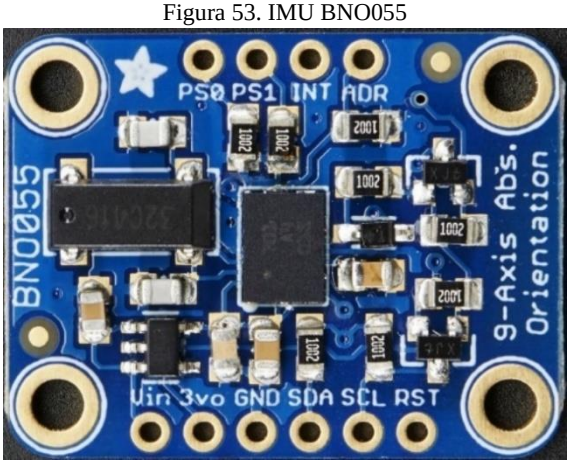
Figura 52. Proceso de descarga de bootloader pasos 2, 4 y 5



Una vez se llevó a cabo este proceso el prototipo de réplica de Arduino en protoboard que se realizó está listo para su programación y uso como un Arduino de cualquier tipo de los que se describió en el marco teórico.

3. Pruebas con IMU BNO055 y Xbee. Por último, para terminar la validación del prototipo y asegurar que es factible replicar el Arduino de forma que facilite su implementación en el Proyecto se realizaron pruebas directamente con los componentes que se van a utilizar en el proyecto. En este caso se utilizaron los módulos IMU BNO055 y el Xbee pro S2C.

Figura 53. IMU BNO055



En la Figura 55. se puede observar el circuito del prototipo de la réplica de Arduino en protoboard conectado a los módulos IMU BNO055 y Xbee S2c. Para esto se descargó el programa de prueba del módulo Recolección de datos de una unidad de medición inercial y cálculo de la cinemática del sensor inercial de bajo costo del Proyecto. Las pruebas realizadas fueron satisfactorias, por lo que se procedió a realizar el diseño de una placa de circuito impreso con un ATMEGA328P-AU, empaquetado SMD.

Figura 54. Prototipo final con IMU BNO055 y Xbee

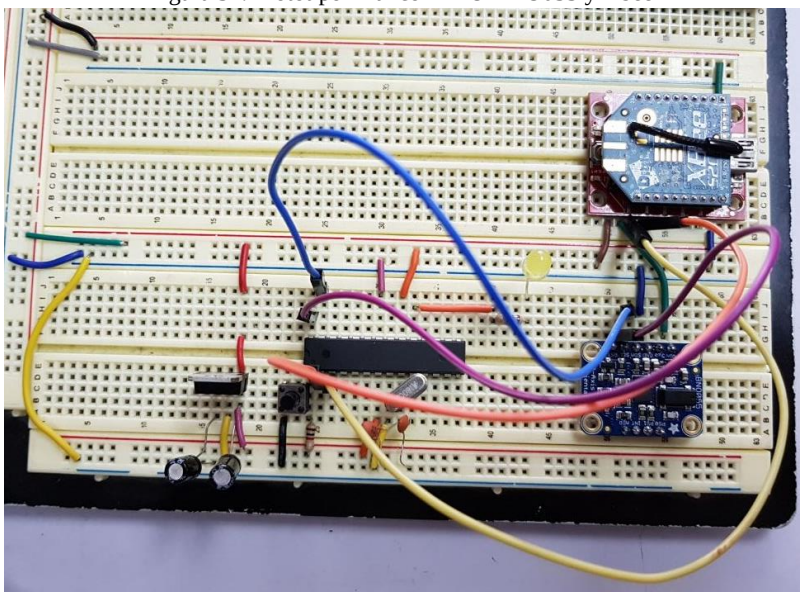
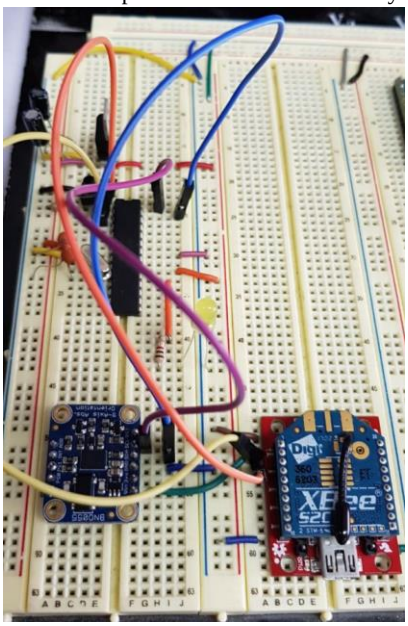
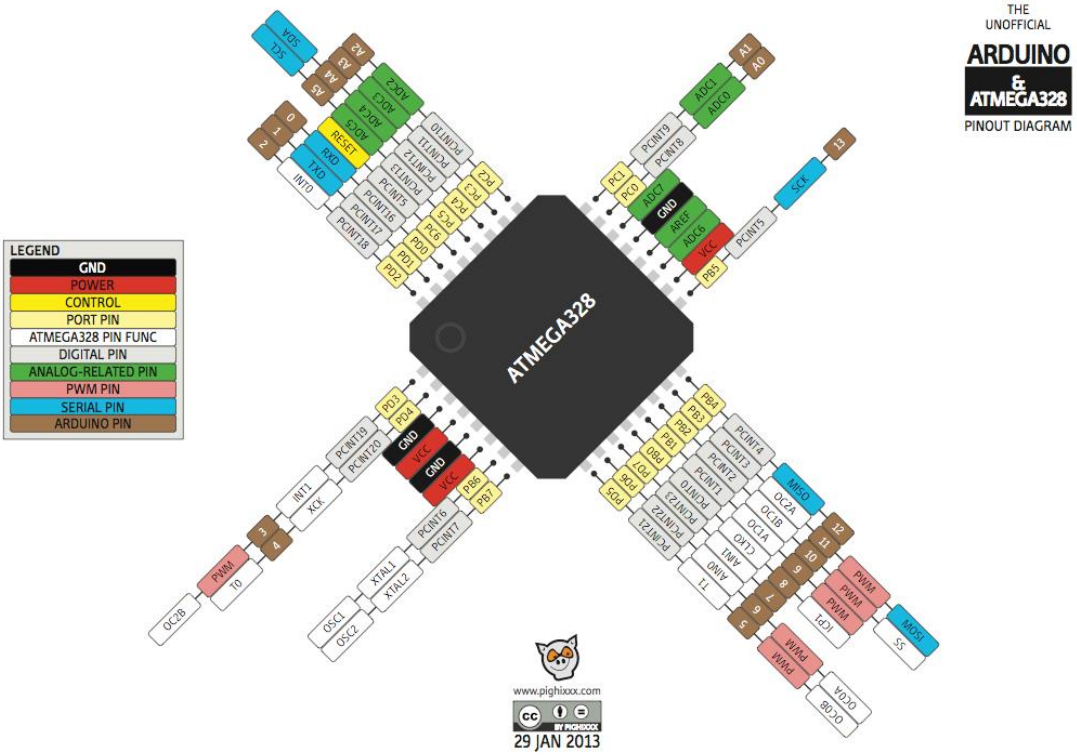


Figura 55. Prototipo final con IMU BNO055 y Xbee



4. Diseño del esquemático del circuito. El diseño del esquemático del circuito se llevó a cabo en el software Altium Designer. Para poder realizar el esquemático del circuito se tomó como referencia el esquemático del circuito del prototipo de réplica de Arduino en protoboard. En este caso al esquemático se trabajó con un ATMEGA328P-AU, es decir un ATMEGA328P en empaquetado SMD. Además, se agregaron pin header macho para el bootloader y la descarga de programación, borneras de tornillo para conectar una batería de 9V y los módulos IMU BNO055 y el Xbee Explorer Regulator. En la Figura 56. se puede observar el mapa pinout del microcontrolador ATMEGA328P-AU que se utilizó.

Figura 56. Pinout de microcontrolador ATMEGA328P-AU



En el Cuadro 10. se pueden observar todos los componentes del esquemático del circuito con su respectiva información. En esta tabla se incluyó un comentario con el nombre del componente, una breve descripción del mismo, el designado que lo representa en el esquemático del circuito de la Figura 57. la cantidad de cada componente, el valor del componente, su precio (varía según fabricante) y la cantidad de pines de cada componente.

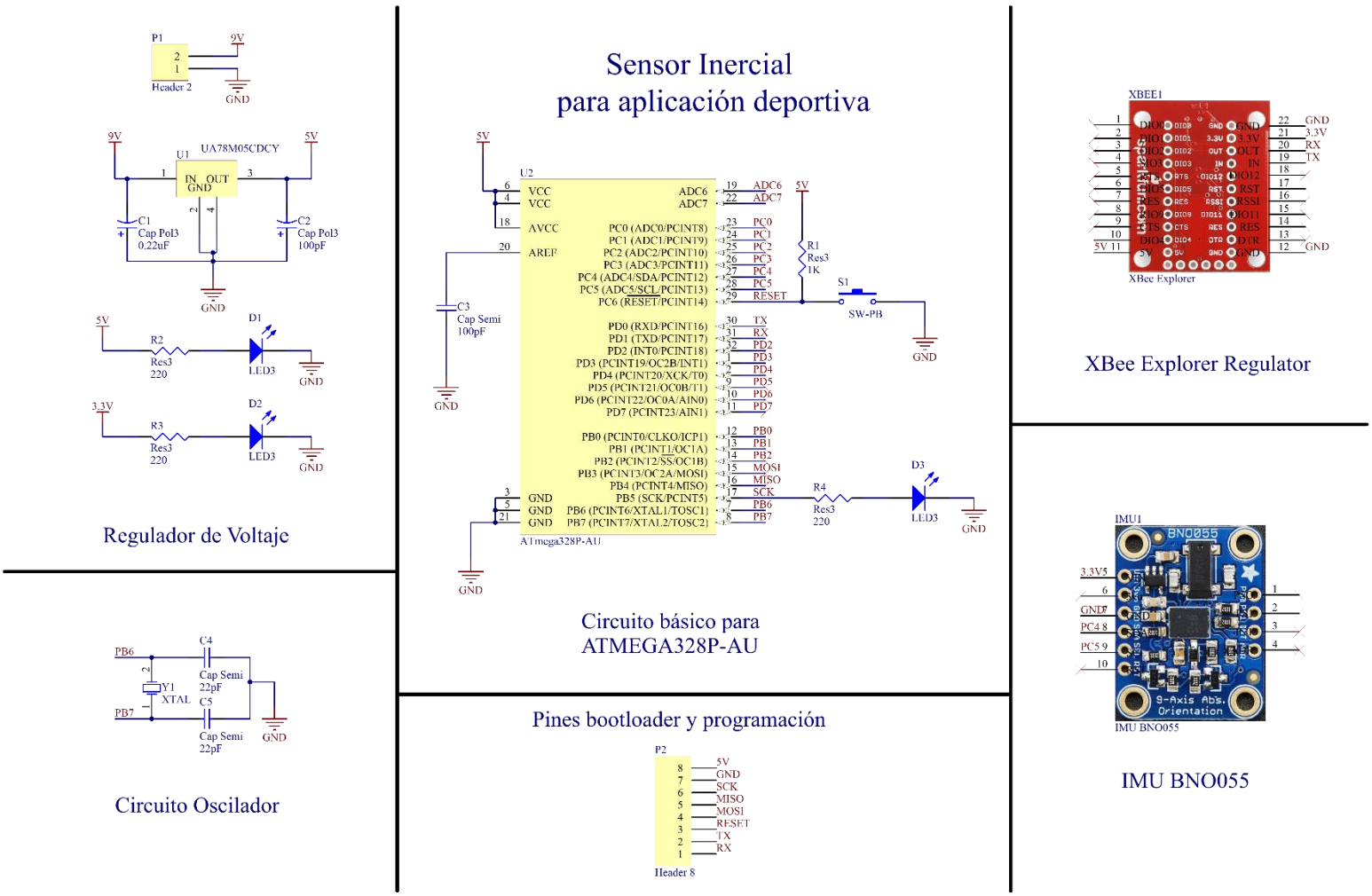
Cuadro 10. Listado de materiales para construcción de sensor inercial.

Comentario	Descripción	Designador	Cantidad	Valor	Precio	Pines
ATmega328P-AU	8-bit AVR Microcontroller, 32KB Flash, 1KB EEPROM, 2KB SRAM, 32-pin TQFP, Industrial Grade (-40°C to 85°C)	U2	1		Q 47.60	32
Cap Pol3	Polarized Capacitor (Surface Mount)	C1	1	0.22uF	Q 4.80	2
Cap Pol3	Polarized Capacitor (Surface Mount)	C2	1	100pF	Q 4.80	2
Cap Semi	Capacitor (Semiconductor SIM Model)	C3	1	100pF	Q 3.80	2
Cap Semi	Capacitor (Semiconductor SIM Model)	C4, C5	2	22pF	Q 3.80	2
Header 2	Header, 2-Pin	P1	1		Q 3.00	2
IMU BNO055	IMU BNO055	IMU1	1		Q 480.00	10
LED3	Typical White SiC LED	D1, D2, D3	3	Blanco	Q 21.80	2
Res3	Resistor	R1	1	1K	Q 1.80	2
Res3	Resistor	R2, R3, R4	3	220	Q 1.80	2
SW-PB	Switch	S1	1		Q 24.60	2
Xbee Explorer	Xbee Explorer Regulator	Xbee1	1		Q 120.00	22
XTAL	Crystal Oscillator	Y1	1		Q 11.60	2
µA78M05CKC	Positive-Voltage Regulator	U1	1		Q 16.60	3
					Q 797.00	

Cuando se diseña el esquemático de un circuito en Altium Designer se deben descargar las librerías con los componentes electrónicos para poder utilizarlos. Sin embargo, aunque este software cuenta con una gran variedad de librerías de muchos proveedores en ocasiones no se encuentran los componentes que se requieren.

Para este diseño se tuvo que crear tres componentes, el regulador de voltaje UA78M05CDCY, el Xbee Explorer Regulator y la IMU BNO055. Esos componentes se pueden observar en la sección de anexos. También se pueden observar en la Figura 57. donde se encuentra el esquemático del circuito completo, estos componentes se identifican en el esquemático del circuito como U1, Xbee1 e IMU1, respectivamente.

Figura 57. Esquemático del circuito de sensor inercial.



Como se puede observar en la Figura 57. el esquemático del circuito se dividió en 6 módulos según las funciones que tiene cada uno de ellos para mantener un orden y sea más sencilla su comprensión. Estos módulos son:

- Regulador de Voltaje
- Circuito Oscilador
- Circuito básico para ATMEGA328P-AU
- Pines bootloader y programación
- Xbee Explorer Regulator
- IMU BNO055

El módulo Circuito básico para ATMEGA328P-AU es el módulo principal del circuito esquemático. Todos los demás módulos se conectan a este. Como su nombre lo dice, este módulo contiene el circuito básico que se requiere para que un ATMEGA328P-AU funcione como un Arduino. En este caso únicamente

se realizaron las conexiones que se van a utilizar, todas las conexiones que no se usaran se indicaron con una "X" roja.

El módulo de regulador de voltaje cuenta con cuatro partes. La primera consta de una bornera de tornillo de 2 entradas donde en una recibe la tierra del circuito y en la otra la entrada de 9V que le proporciona la batería. La segunda parte es el regulador de voltaje como tal, donde recibe la señale de 9V y de salida se tienen 5V. Por último, tiene 2 LEDs uno conectado a la salida de 9V y el otro a la salida de 3.3V, que proviene del Xbee Explorer Regulator, esto para poder verificar visualmente que efectivamente hay 5V y 3.3V en el circuito al momento de tenerlo físico.

El módulo de circuito oscilador consiste en un cristal oscilador de 16MHz conectado con dos capacitores cerámicos de 22pF. Este módulo es necesario, ya que el microcontrolador ATMEGA328P depende una frecuencia de oscilación para funcionar adecuadamente. Aunque este consta de un oscilador interno es preferible conectar uno externo, debido a que el interno es de 8MHz, es decir, tiene la mitad de velocidad que el externo.

El módulo de pines para bootloader y programación consta de 8 pines. Estos pines están basados en la fase de prototipo "Descarga de bootloader y programas al ATMEGA328P", ya que tiene los pines para la alimentación del circuito, para la descarga de bootloader y la descarga del programa al ATMEGA328P. Este módulo está pensado para que sea fácil la identificación de los pines para las tareas mencionadas.

Por último, consta de los dos módulos de Xbee Explorer Regulator y la IMU BNO055. El módulo de Xbee es el que se utiliza para poder realizar la transferencia de datos de forma inalámbrica, mientras que el módulo IMU BNO055 contiene un acelerómetro y giroscopio para realizar las mediciones del circuito.

5. Cálculo de pistas para placa de circuito impreso. Al momento de realizar el diseño de una placa de circuito impreso se debe tomar en consideración una serie normas y cuidados para que el circuito funcione de forma adecuada. Una de las normas más importantes es el ancho de las pistas de la placa de circuito impreso. Esto se debe a que si las pistas son muy pequeñas y la corriente que pasa a través es muy grande la pista tiende a calentarse y se despega el cobre de la placa.

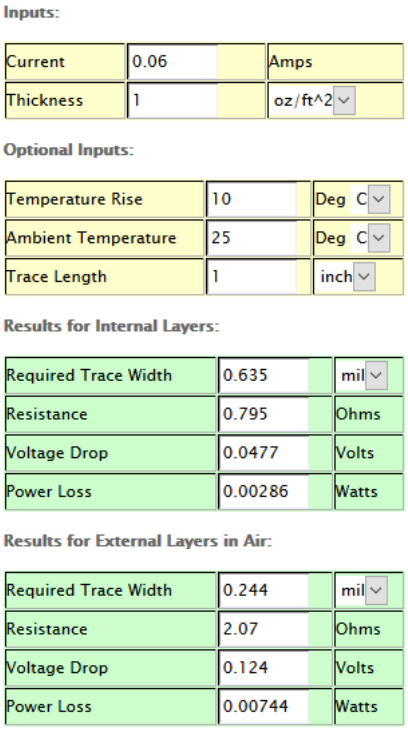
Para poder realizar el cálculo de las pistas de la placa de circuito impreso se midieron las corrientes del circuito en el prototipo de réplica de Arduino en protoboard cuando se realizaron las pruebas con los módulos Xbee e IMU BNO055. En estas mediciones se obtuvo que la corriente más grande es en la entrada de los 9V provenientes de la batería, donde se midió una corriente de 60mA. En el resto del circuito se obtuvo una medición de corriente menor a los 30mA. Como se sabe que el ATMEGA328P tiene un límite de corriente

de 40mA se sobredimensiono a esta medida para las pistas de voltaje de 5V y 3.3V, así como al de la pista de GND.

Se utilizaron las fórmulas 3 y 4, especificadas en el marco teórico, para obtener un ancho de pista para la placa de circuito impreso. Se obtuvo que para la pista de los 9V se requiere un ancho mínimo de 0.25 mils. Para las pistas de 5V, 3.3V y GND se obtuvo un ancho mínimo de 0.15 mils. De esta forma se realizó la medición aproximada, ya que no se tenía el largo exacto de cada una de las pistas.

Para verificar los cálculos realizados con las ecuaciones 3 y 4, se utilizó una página web, la cual calcula el ancho de las pistas de una placa de circuito impreso según los datos que se ingresen para el cálculo, tomando en cuenta variables que no tiene las ecuaciones 3 y 4, siendo esto más exacto para realizar el cálculo. En la Figura 58. se puede observar una captura de pantalla de dicha página web con uno de los cálculos realizados. Esta verificación se llevó a cabo cuando ya se tuvo el diseño de la PCB para poder colocar el largo de las pistas y tener un dato más exacto que el anterior. En la Figura 59. se realizó nuevamente el cálculo suponiendo que el regulador de voltaje estaría en su máxima capacidad de 0.5 amperios, obteniendo un ancho de pista de 4.55mils.

Figura 58. Cálculo de ancho de pistas en página web, con una corriente de 60mA



Inputs:

Current	0.06	Amps
Thickness	1	oz/ft^2

Optional Inputs:

Temperature Rise	10	Deg C
Ambient Temperature	25	Deg C
Trace Length	1	inch

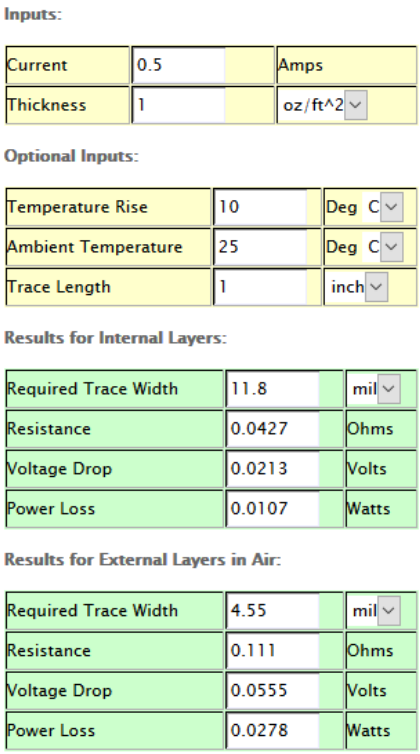
Results for Internal Layers:

Required Trace Width	0.635	mil
Resistance	0.795	Ohms
Voltage Drop	0.0477	Volts
Power Loss	0.00286	Watts

Results for External Layers in Air:

Required Trace Width	0.244	mil
Resistance	2.07	Ohms
Voltage Drop	0.124	Volts
Power Loss	0.00744	Watts

Figura 59. Cálculo de ancho de pistas en página web, con una corriente de 0.5A



Inputs:

Current	0.5	Amps
Thickness	1	oz/ft^2

Optional Inputs:

Temperature Rise	10	Deg C
Ambient Temperature	25	Deg C
Trace Length	1	inch

Results for Internal Layers:

Required Trace Width	11.8	mil
Resistance	0.0427	Ohms
Voltage Drop	0.0213	Volts
Power Loss	0.0107	Watts

Results for External Layers in Air:

Required Trace Width	4.55	mil
Resistance	0.111	Ohms
Voltage Drop	0.0555	Volts
Power Loss	0.0278	Watts

6. PCB Altium. Luego de finalizar el diseño del esquemático del circuito y haber calculado los anchos mínimos de las pistas de la placa de circuito impreso se procede a realizar el diseño del PCB en el software Altium Designer. La ventaja de hacer en este software tanto el esquemático del circuito como el PCB es que únicamente se debe exportar al área de diseño de PCB para hacerlo.

Lo primero que se hizo fue verificar los footprints de cada uno de los componentes para que fueran los correctos. En este caso se encontró que los footprints de los LEDs, botón SMD y cristal oscilador no eran los adecuados, por lo que se procedió a realizar el diseño de cada uno de estos footprints. A cada uno de estos 6 footprints que se diseñaron se les agregó su modelo 3D, los cuales se pueden observar en la sección de anexos.

Teniendo todos los footprints correctos se procedió a ordenar los componentes. Al estar buscando diseñar una placa de circuito impreso que no supere las medidas de 5 x 5 cm de largo y ancho se definió que la placa será de doble capa, es decir, se colocaran componentes en la capa superior e inferior de la placa y se utilizaran vías *Thru-hole* para interconectar ambas capas. Para ahorrar espacio se decidió colocar el Xbee Explorer Regulator en la capa superior y la IMU BNO055 en la capa inferior, adicionalmente a esto se tomó en cuenta dejar el espacio suficiente en la capa inferior para poder colocar la batería de 9V.

Al haber ordenado todos los componentes en un espacio menor a 5x5 cm se realizaron las conexiones de los componentes con las pistas, tomando en cuenta los anchos anteriormente mencionados, así como evitar

ángulos en 90° y distancia entre pistas alejadas unas de otras, entre otras normas de diseño que se consideraron.

En la Figura 60. se puede observar el diseño del PCB del sensor inercial. Las pistas de color rojo son aquellas que se encuentran en la capa superior de la placa de circuito impreso, mientras que las pistas de color azul son las que se encuentran en la capa inferior. Adicionalmente, en las Figuras 61,62,63 y 64 se puede observar el diseño 3D de la placa de circuito impreso, en las cuales se muestra la capa superior e inferior de diferentes ángulos. El tamaño final de la placa de circuito impreso es de 4.91x4.06 cm, cumpliendo de esta forma el objetivo de diseñar una placa menor a 5x5cm de largo y ancho.

Figura 60. Diseño del PCB, pistas rojas en capa superior, pistas azules en capa inferior

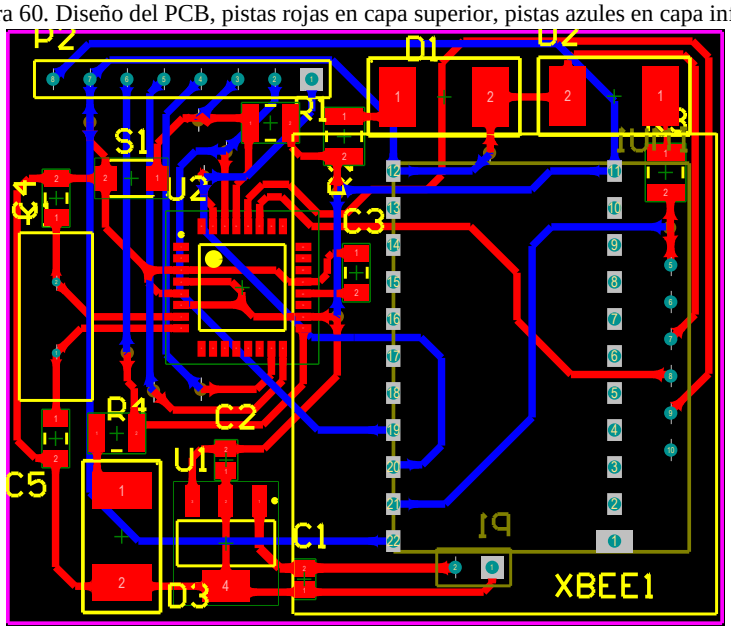


Figura 61. Vista superior del diseño 3D

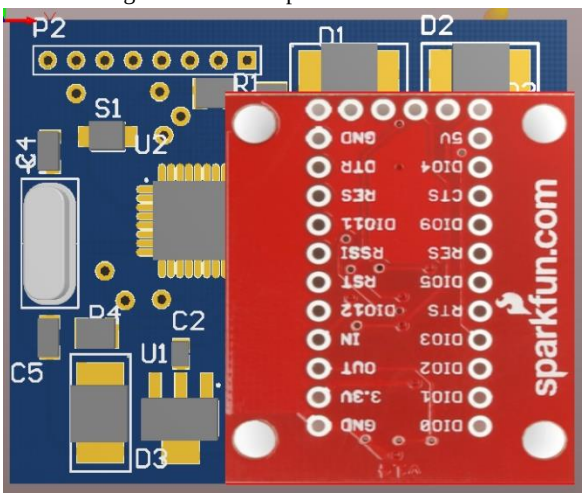


Figura 62. Vista inferior del diseño 3D

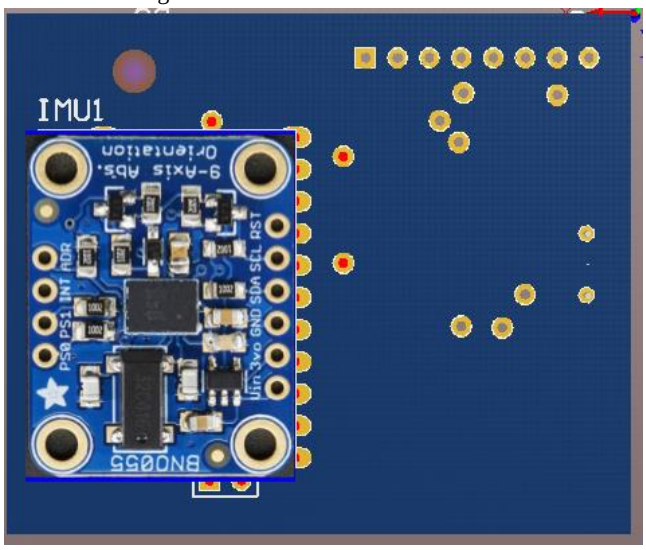


Figura 63. Vista ortogonal de la parte superior del diseño 3D

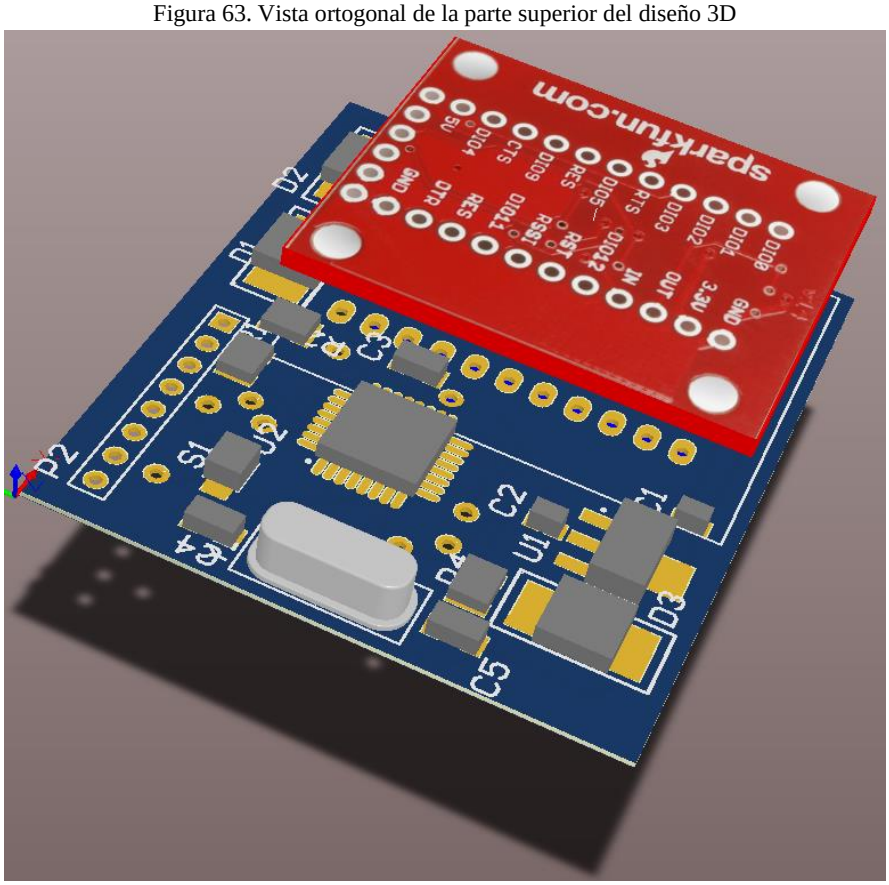
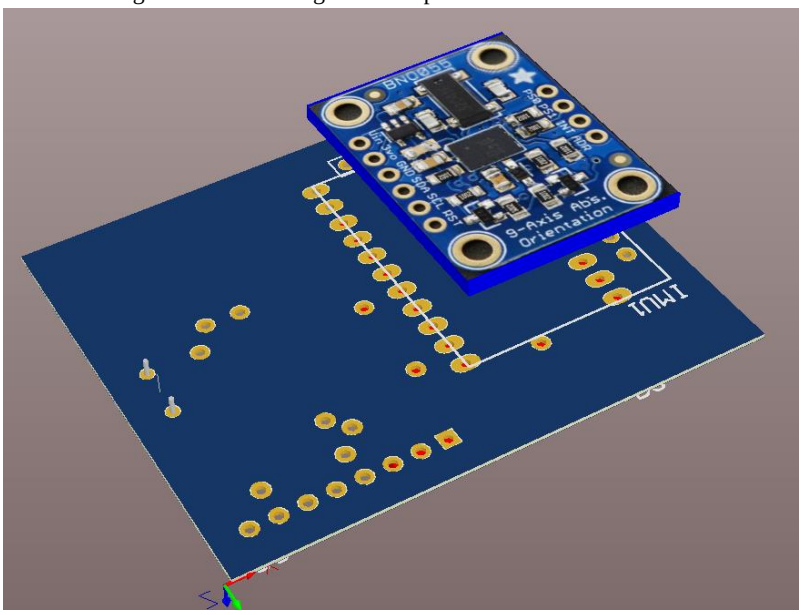


Figura 64. Vista ortogonal de la parte inferior del diseño 3D



7. Manufactura. Al terminar el diseño de la placa de circuito impreso en el Software Altium Designer se pudo proceder a la manufactura de la misma. Para poder llevar a cabo la manufactura de la placa de circuito impreso, esta se dividió en tres fases.

1. Fresado de la placa de circuito impreso
2. Soldado de la placa de circuito impreso
3. Descarga de bootloader y programa a la placa de circuito impreso

a. Fresado de la placa de circuito impreso. El fresado de la placa de circuito impreso se llevó a cabo en una computadora de control numérico (CNC por sus siglas en inglés). Para poder comunicarse con la CNC se utilizaron archivos conocidos como Gerber, los cuales se obtienen con el software Altium Designer. El software Altium Designer tiene la funcionalidad de poder generar estos archivos Gerber para el maquinado y fresado de las placas de circuito impreso que se diseñan.

Para poder llevar a cabo la manufactura con la CNC se exportaron cuatro tipos de archivos Gerber. Dos de ellos son las capas superior e inferior de la placa de circuito impreso, estas tienen la configuración de cómo se debe de maquinar el circuito de la placa. El otro archivo es el que le indica a la CNC el tamaño de la placa. Por último, el cuarto archivo es el que tiene las coordenadas de los agujeros que se deben realizar en la placa de circuito impreso. De esta forma la CNC puede fabricar una placa de circuito impreso en un corto periodo de tiempo según el tamaño y detalle que se desee en la misma. En las Figuras 65 y 66 se puede observar como se ve los archivos Gerber de la capa superior (rojo), inferior(azul) y el que delimita el tamaño de la placa (fucsia) y el de los agujeros, respectivamente. En la Figura 66. los puntos celestes son los agujeros que deben ser taladrados por la CNC. En la Figura 67 y 68 se puede observar la capa superior e inferior, respectivamente, de la placa de circuito impreso diseñada.

Figura 65. Archivo Gerber capa superior (rojo), inferior(azul) y mecánica (fucsia)

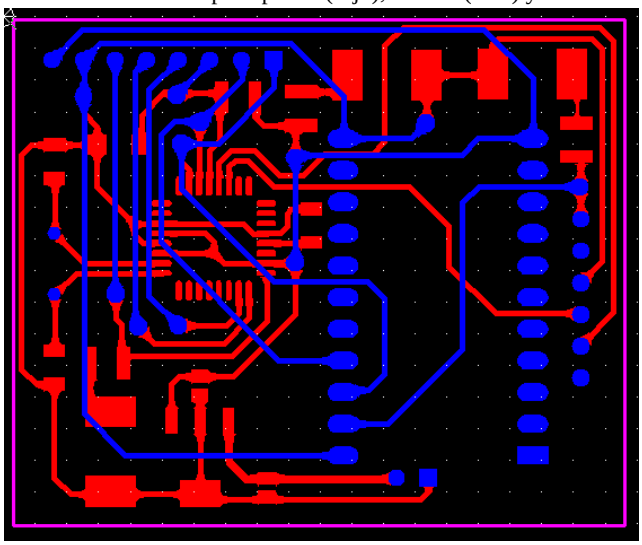


Figura 66. Archivo Gerber del taladrado que debe llevar a cabo la CNC

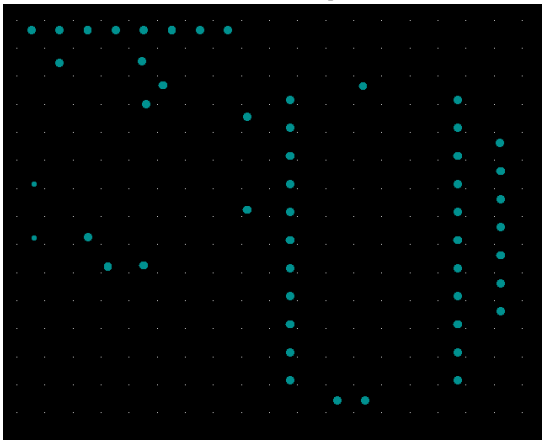


Figura 67. Capa superior de la placa de circuito impreso diseñada

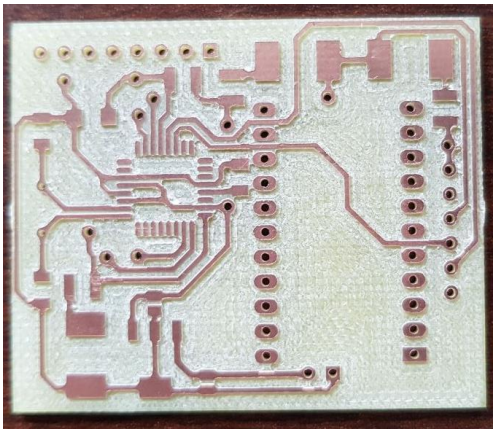
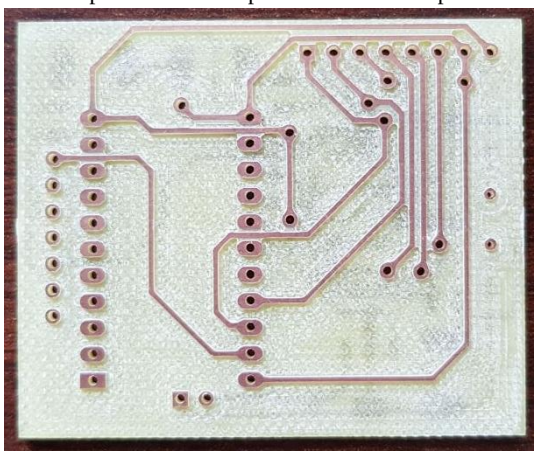


Figura 68. Capa inferior de la placa de circuito impreso diseñada



b. Soldado de placa de circuito impreso. Para realizar la soldadura de la placa esta se separó en tres tipos de soldado diferente. Estos fueron los siguientes: soldado de vías, soldado de componentes SMD y soldado de componentes Thru-hole. Estas soldaduras se llevaron a cabo en el orden anteriormente mencionado.

Para realizar la soldadura final se tenían tres opciones: no utilizar headers hembras para colocar el Xbee Explorer Regulator y la IMU BNO055, utilizar únicamente de 1 de los 2 lados y utilizar de los 2 lados. Antes de realizar la soldadura se realizaron mediciones para verificar las diferentes alturas que puede tener el sensor, las cuales son de 19.72mm, 26.01mm y 33.42mm respectivamente. En las Figuras 69,70 y 71 se puede observar las tres opciones respectivamente. La batería en la Figura 71Figura 61. es demostrativa para representar el tamaño del sensor.

Figura 69. Sensor sin headers hembra

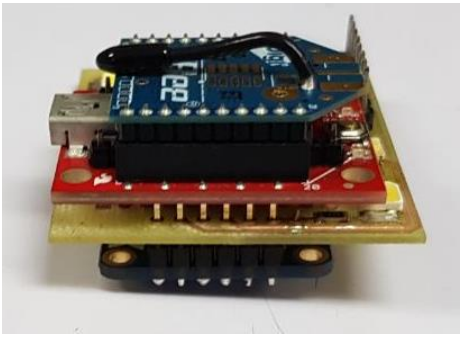


Figura 70. Sensor con headers hembra únicamente de un lado

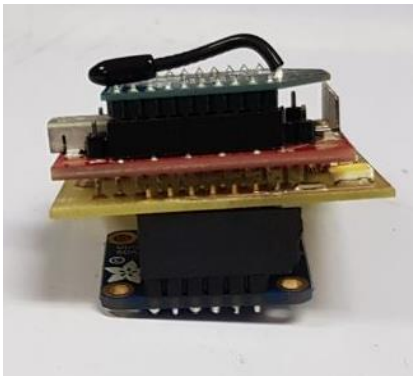
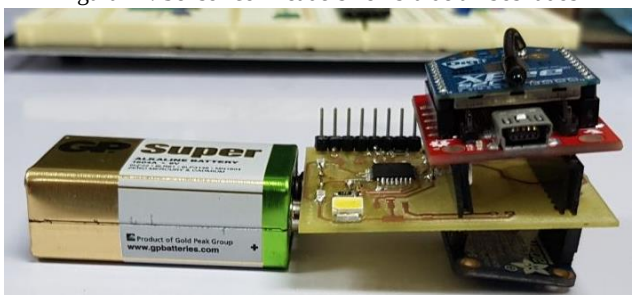


Figura 71. Sensor con headers hembra de ambos lados



Ya que los componentes Xbee Explorer Regulator y la IMU BNO055 pertenecen a la Universidad del Valle de Guatemala se decidió soldar los headers en ambos lados de la placa para que los componentes pudieran ser removidos con facilidad.

Las soldaduras de vías son aquellas que se utilizaron para conectar las capas superior e inferior entre ellas, por lo que requería introducirse un cable en la vía y soldarlo en ambas superficies de la placa de circuito impreso para realizar la conexión entre ellas. Las soldaduras de superficie son aquellas de los componentes con empaquetado SMD, estas estaban únicamente en la capa superior y son las soldaduras más pequeñas de la placa, por lo que se tuvo mucho cuidado al momento de realizar la soldadura. Por último, se llevó a cabo la soldadura de los componentes *Thru-hole*, en este caso si se tenía soldadura de este tipo de componentes en ambas superficies. En las Figuras 72 y 73 se puede observar las capas superior e inferior con el proceso de soldado terminado.

Figura 72. Vista superior del PCB con el proceso de soldado terminado

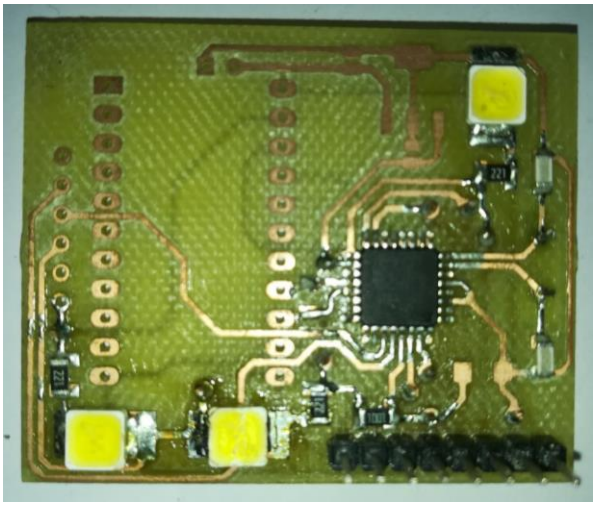
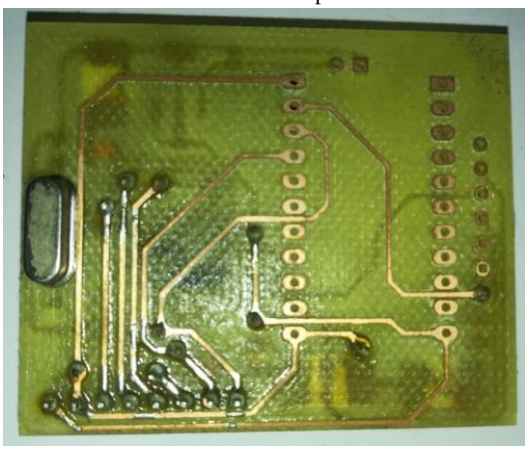


Figura 73. Vista inferior del PCB con el proceso de soldado terminado



8. Bootloader y programación PCB.

Para la descarga del bootloader y del programa no se debía tener la placa completamente soldada con todos los componentes, no se debía tener soldado el módulo de regulador de voltaje, IMU BNO055 y Xbee Explorer Regulator, explicados en la sección de diseño del esquemático del circuito de la placa de circuito impreso en la metodología.

No se deben tener estos módulos, ya que para la descarga del bootloader y del programa a la placa del circuito impreso la alimentación proviene del Arduino con el que se está llevando a cabo la descarga. Respecto a los otros dos módulos mencionados, estos no son necesarios para este proceso, por lo que únicamente estorbarían al momento de descargar el bootloader y el programa. En las Figuras 74 y 75 se puede observar la capa superior e inferior de la placa de circuito impreso, respectivamente.

Figura 74. Vista superior del PCB listo para descarga de bootloader y programa

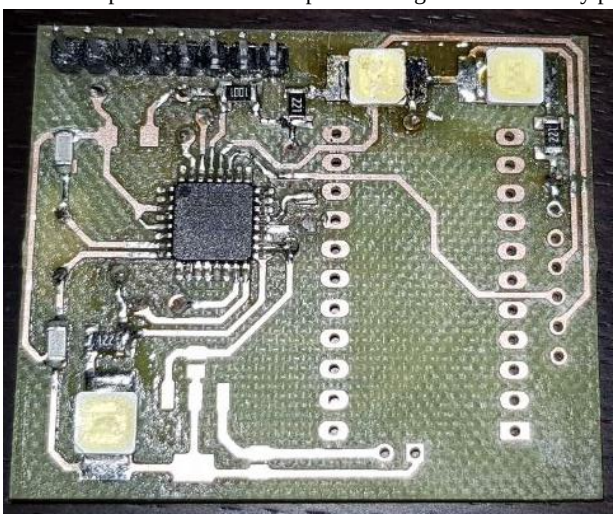
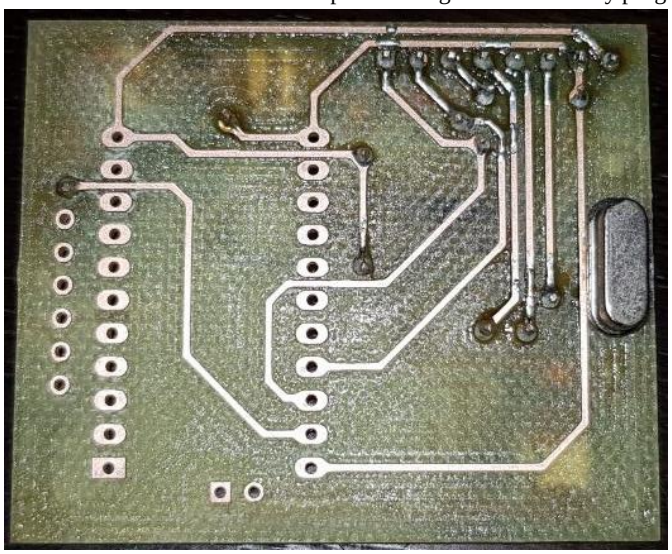


Figura 75. Vista inferior del PCB listo para descarga de bootloader y programa



Para la descarga del bootloader y del programa se utilizaron un Arduino UNO y un Arduino NANO. Para poder llevar esto a cabo se debió realizar la conexión de los pines mencionados en el Cuadro 11 con el Arduino NANO y Arduino UNO. En la Figura 76. se puede ver un diagrama de la conexión del Arduino NANO con un ATMEGA328P-AU y su respectivo circuito para realizar la descarga del bootloader, sin incluir las conexiones de 5V y GND. El proceso de bootloader se probó realizar con ambas placas de Arduino, en la Figura 76. y Figura 77. se puede observar la conexión entre la placa de circuito impreso la placa de Arduino NANO

Cuadro 11 Conexión de pines entre ATMEGA328P-AU con distintos Arduino

Conexión de pines entre ATMEGA328P-AU con las placas de Arduino UNO y Arduino NANO		
ATMEGA328P-AU	ARDUINO UNO	ARDUINO NANO
15	11	11
16	12	12
17	13	13
29	10	10

Figura 76. Esquemático para descarga de bootloader con Arduino NANO y ATMEGA328P-AU

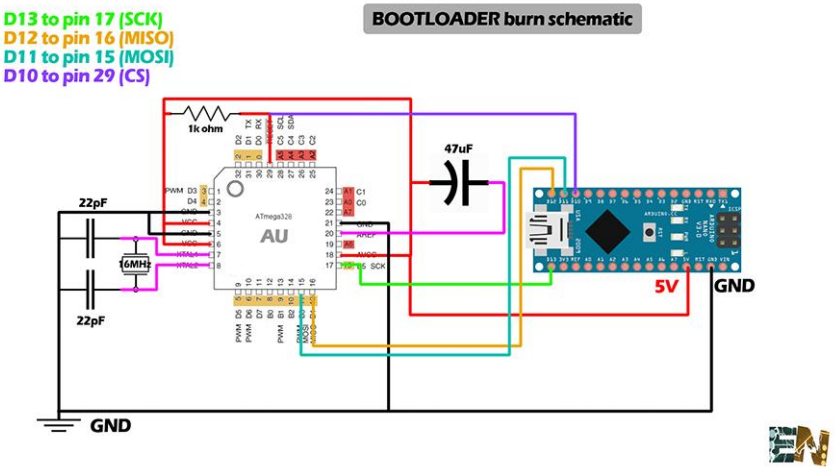


Figura 77. Conexión entre PCB y Arduino NANO para la descarga del bootloader

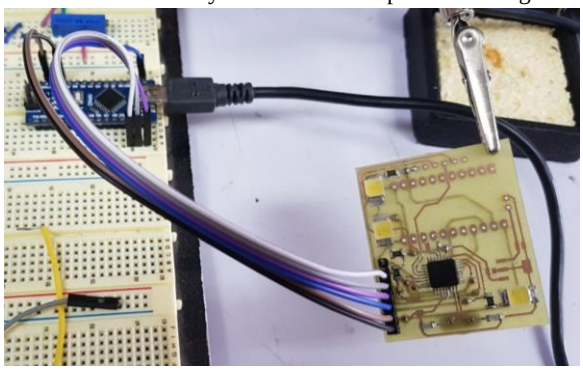
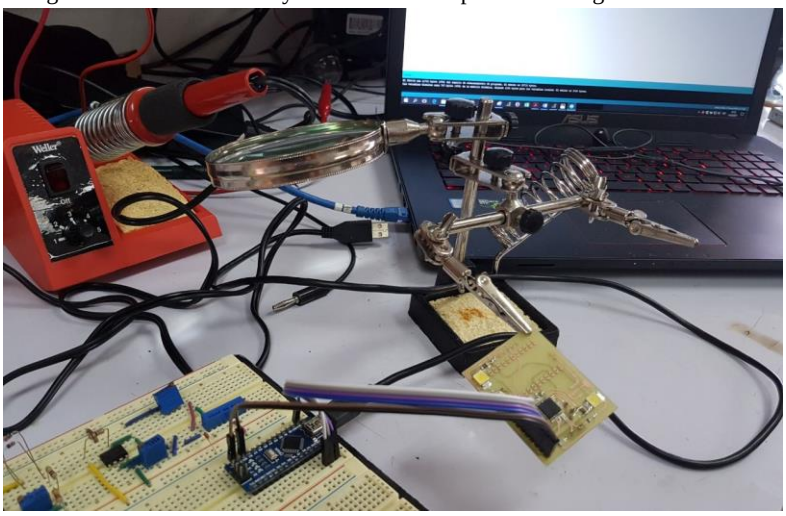


Figura 78. Conexión PCB y Arduino NANO para la descarga del bootloader



Para la descarga de programas se utiliza únicamente el Arduino UNO por su capacidad de poder retirar el microcontrolador ATMEGA328P-PU de la palca. Lo primero que se debe hacer es retirar el microcontrolador ATMEGA328 del Arduino UNO. Luego para las conexiones de los pines solo se debe conectar la alimentación de 5V y GND en los pines correspondientes y conectar los pines TX y RX del Arduino UNO y el microcontrolador ATMEGA328P-AU entre ellos. Es decir, TX con TX y RX con RX. En las Figuras 59 se puede observar dicha conexión.

Figura 79. Conexión de PCB con Arduino UNO para la descarga del programa

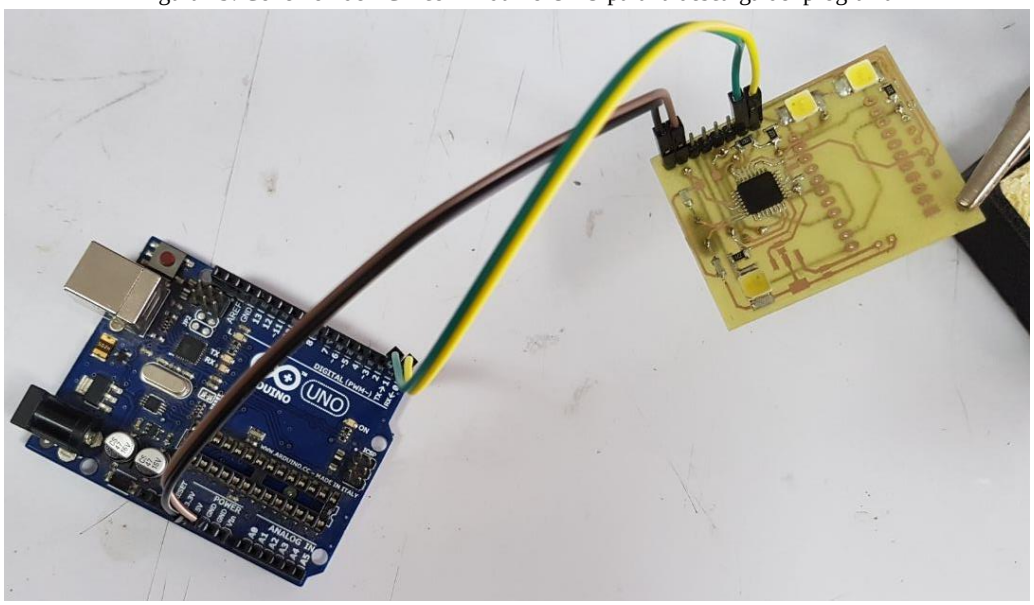
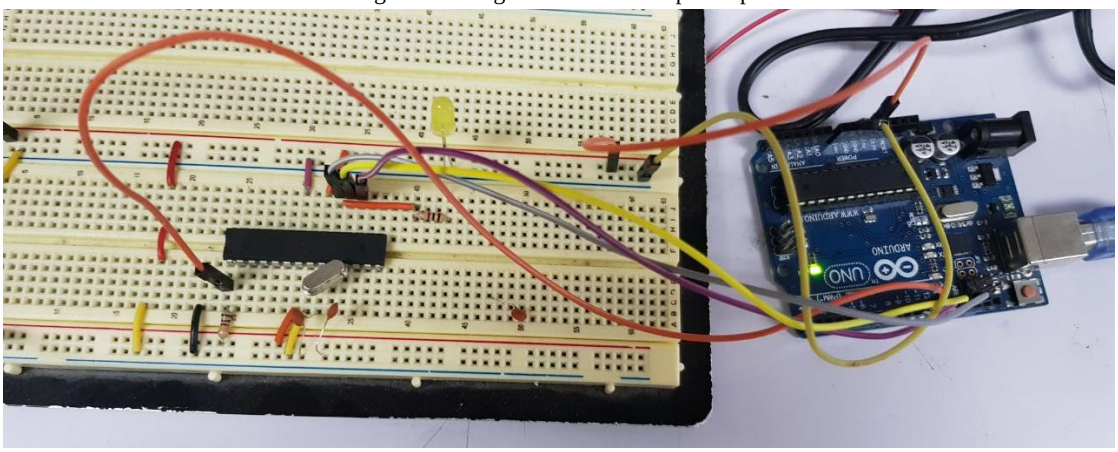


Figura 80. Carga de bootloader a prototipo



Al terminar la construcción, descarga de bootloader y descarga de programa se realizaron pruebas sencillas para verificar que todo funcionará como se esperaba. Para las pruebas se descargaron varios programas que variaban la duración de encendido y apagado del LED conectado al PIN 19 / PB5 / SCK del ATMEGA328P conectado al protoboard, siendo satisfactorias todas las pruebas realizadas.

B. Recolección de datos sensor inercial

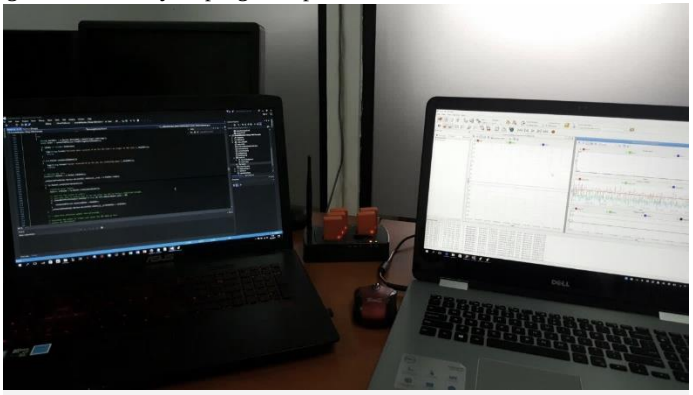
El proyecto tuvo inicios en el 2016 por una proposición que personas del Comité Olímpico Guatemalteco (COG) hicieron al departamento de electrónica y mecatrónica de la Universidad del Valle. Esto con el fin de elevar el nivel de los competidores olímpicos guatemaltecos mediante la utilización de ciencia aplicada al deporte. El objetivo era utilizar, reparar y elaborar equipo de biomecánica y biofeedback para el desarrollo de dichos deportistas. En un comienzo se llegó a las instalaciones del COG para reconocer el equipo que se disponía llevar a cabo el proyecto. También se realizó un semestre de investigación sobre los avances de otros países en el campo de la ciencia aplicada al deporte para proponer ideas que generen un impacto en el deporte guatemalteco. También se investigó los distintos equipos que se utilizan a nivel profesional para el desarrollo de los deportistas por medio de la ciencia y los resultados que se obtienen con los distintos equipos. El COG indicó que contaba con un presupuesto para comprar equipo de biomecánica y biofeedback, y que necesitaban asesoría en cuanto al equipo que debían comprar. Es por ello que se realizó una proposición de los equipos que son costeables que pudieran dar resultados significativos y de esta manera aportar información valiosa a los entrenadores y a los deportistas para mejorar su desempeño. El equipo de estudiantes del Megaproyecto estaba conformado por ocho estudiantes de ingeniería mecatrónica, y se decidió que para que se realizaran proyectos con avances significativos y hubiera trabajo suficiente para todos los módulos, se dividió el grupo en cuatro de biofeedback y cuatro de biomecánica.

El equipo de biomecánica comenzó proponiendo equipo para que el COG comprar y de esta manera realiza una investigación que tuviera valor para el deporte guatemalteco. Como se mencionó con anterioridad, se llegó en varias ocasiones a las instalaciones del COG a reconocer cierto equipo que ellos disponían. Entre ese equipo se tomó un equipo de análisis biomecánico llamado MuscleLab, el cual se encontraba averiado. Se realizó un proceso de configuración de los sensores y finalmente se restauró la mayoría de los sensores de dicha estación.

Figura 81. Restauración de equipo MuscleLab

El COG decidió por su parte pedir un equipo llamado XSens, el cuál es para análisis biomecánico. Cuando el equipo vino a Guatemala a principio del 2017 resulta que el equipo vino con seis sensores inerciales, los cuales son de muy alta calidad, pero el software que acompañaba al equipo tenía muchas limitaciones. Era un equipo orientado para el desarrollador, lo cual para el grupo de biomecánica fue algo muy positivo y conveniente. Es por ello que el objetivo del proyecto tomó un giro drástico y ahora sería elaborar un software que permitiera obtener información relevante de dichos sensores y posteriormente utilizando este software realizar una investigación con las mediciones obtenidas.

Figura 82. XSens y su programa para el desarrollador en C++



Una de las grandes limitaciones al trabajar con estos sensores es que no se podía hacer en cualquier lugar, por motivos de pruebas con los sensores, sino solamente en las instalaciones del COG. Se estuvo llegando a

trabajar en dicho equipo de mediados de enero a finales de marzo. En este período se realizaron muchas pruebas y se comenzó a desarrollar el programa propuesto en base a un programa contenido en la carpeta para el desarrollador que ofrecía el fabricante. A finales de marzo se indicó por parte del COG que ya no se permitiría seguir trabajando a los estudiantes con equipo del COG por cuestiones legales, pero que su interés de trabajar en conjunto continuaba.

Es por ello que como equipo de biomecánica se tuvo que volver a proponer proyectos para el desarrollo de la ciencia aplicada al deporte con el fin de mejorar el desempeño de los deportistas. Con este fin se propuso el desarrollo e implementación de sensores inerciales que permitieran realizar un análisis biomecánico, como se pretendía hacer con los XSens. A continuación, se pulió la idea y resultó en sensores inerciales de bajo costo, con el fin de que al contar con un conjunto de ellos se pudiera realizar una aplicación que brindara mucha información a los entrenadores y deportistas sobre la técnica del movimiento realizado. Este proyecto se dividió dentro de los cuatro estudiantes asignados a biomecánica, y los roles dentro del proyecto fueron: interfaz gráfica, obtención de la trayectoria, diseño mecánico del sensor y diseño electrónico del sensor. Esta investigación es sobre la obtención de la trayectoria de los sensores inerciales.

1. Selección de componentes. La finalidad principal del megaproyecto es elaborar sensores inerciales de bajo presupuesto, para ello es de suma importancia seleccionar los componentes que conformarán dicho sensor. Los principales componentes que se utilizarán son: Transmisor inalámbrico para obtener los datos en tiempo real, unidad de medidas inerciales para obtener todas las mediciones necesarias para obtener la trayectoria y también un microcontrolador que pueda realizar ciertos cálculos para luego enviar los datos de una manera estándar. Para la selección de cada uno de los componentes a utilizar se utilizarán matrices de decisión, las cuales ponderan de igual manera todos los criterios a evaluar y al realizar la sumatoria de cada uno de los candidatos se tomará la decisión del ganador.

a. Selección microcontrolador. El microcontrolador es una parte muy importante en el sensor inercial, ya que este realiza los cálculos necesarios para que los acelerómetros que se envían a la computadora sean con respecto a un marco global, y a su vez se encarga de enviarle los datos al transmisor inalámbrico para luego ser recibidos en la computadora. Este dispositivo debe realizar las tareas anteriormente descritas dentro de un tiempo establecido, para que el muestreo sea representativo y que tenga congruencia. Se evaluarán ciertos criterios para seleccionar el microcontrolador adecuado por medio de una matriz de decisión. Entre los dispositivos a evaluar se encuentran, Arduino Nano y Teensy 3.2.

Como se planteó con anterioridad, se utilizará una matriz de decisión para la selección del microcontrolador a utilizar. Los criterios de selección del microcontrolador son: Disponibilidad de librerías, programación, portabilidad, precio y disponibilidad. El 10 se le asigna al candidato que cumple el requerimiento de la mejor manera. Dependiendo del cumplimiento del otro dispositivo respecto al criterio a evaluar se le coloca el punteo entre 1 y 10. Todos los criterios tienen el mismo peso. El candidato seleccionado será el que se implementará en el dispositivo inercial.

Cuadro 12. Matriz de decisión del microcontrolador

Criterio	Arduino Nano	Teensy 3.2
Librerías	10	10
Programación	10	10
Portabilidad	9	10
Precio	10	7
Disponibilidad	10	7
Total	49	44

Como se puede apreciar en la matriz de decisión, el candidato ganador es el Arduino Nano. Los dos criterios decisivos para su selección fueron la disponibilidad y el precio. El teensy es un microcontrolador muy útil en el análisis de señales, ya que puede funcionar a velocidades muy altas. Pero para esta investigación el microprocesador del Arduino Nano cumple con los requerimientos planteados.

2. Selección de transmisor inalámbrico. Existen muchos dispositivos y protocolos para enviar datos de manera inalámbrica. En este caso se necesita desacoplar una señal entre el microcontrolador a utilizar y la computadora, que por defecto se intercomunican por medio de protocolo serial en alguno de los puertos de la computadora. La selección del dispositivo a utilizar se hará por medio de una serie de criterios, en los que se ponderará las características de los candidatos a evaluar. Entre los dispositivos a analizar se encuentran, Xbee y bluetooth.

a. Matriz de decisión de dispositivo inalámbrico. Como se planteó con anterioridad, se utilizarán ciertos criterios para la selección del dispositivo inalámbrico a utilizar. Entre los criterios de selección del dispositivo de transmisión inalámbrico a utilizar se encuentran: máxima distancia de alcance entre dispositivos, interconexión multipunto, portabilidad, precio y disponibilidad. El 10 se le asigna al candidato que cumple el requerimiento de la mejor manera. Dependiendo del cumplimiento del otro dispositivo respecto al criterio a evaluar se le coloca el punteo entre 1 y 10. Todos los criterios tienen el mismo peso. El candidato seleccionado será el que se implementará en el dispositivo inercial.

Cuadro 13. Matriz de decisión de dispositivo inalámbrico

Criterio	Xbee	HC-05
Distancia Máxima	10	5
Interconexión Multipunto	10	0
Portabilidad	5	10
Precio	5	10
Disponibilidad	10	10
Total	40	35

Como se puede ver, el dispositivo que mejor cumple los requerimientos es el Xbee, por lo tanto, será el seleccionado a utilizar. Como se mencionó anteriormente hay muchos modelos de Xbees, pero se debe seleccionar uno específico, ya que los distintos modelos no son compatibles entre sí.

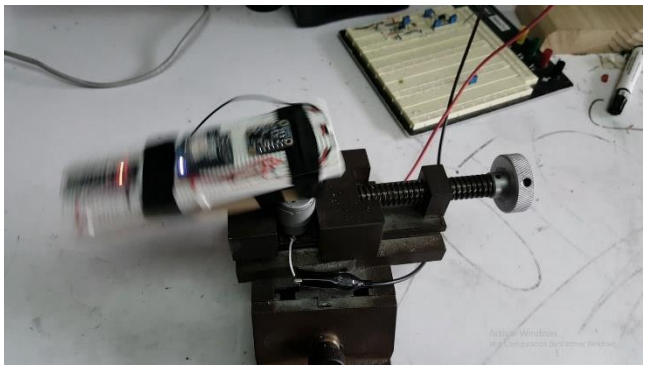
Los dos modelos disponibles en el Departamento de Electrónica y Mecatrónica de la Universidad del Valle de son: Xbee Pro S1 y Xbee S2. De los cuales se seleccionó el Xbee S2 por los siguientes criterios:

- El Xbee S2, como se indica en el Datasheet, tiene un rango máximo de alcance de 120m, mientras que el Xbee Pro S1 tiene un rango máximo de 90m. El dispositivo inercial debe ser capaz de enviar datos del deportista en movimiento, por lo tanto, es sumamente importante que el rango de alcance sea el máximo posible.
- Se realizaron pruebas con una serie de datos aleatorios concatenados a una velocidad de envío entre datos de 10ms. En la computadora se recibieron los datos en tiempo real, y el resultado fue el siguiente: El Xbee Pro S1 no lograba mantener un envío constante de datos, sino enviaba una serie de datos, y luego paraba, mientras que el Xbee S2 mantuvo el envío constante de datos.
- El Xbee S2 se puede comunicar con varios dispositivos a la vez con una red ZigBee, la cual se puede utilizar para una comunicación punto a multipunto. Esto es de suma utilidad, ya que en un futuro se podrían implementar varios sensores inerciales en una misma persona y medir distintos puntos del atleta. Para esto, se tendría que tener un dispositivo maestro conectado a la computadora y varios esclavos en los distintos puntos del atleta.

3. Selección de la unidad de medidas inerciales. La unidad de medidas inerciales juega un papel muy importante en el sensor inercial de bajo costo que se realizará. De sus mediciones depende le resultado de la trayectoria a obtener. Es por ello que los criterios a utilizar serán exhaustivamente evaluados y analizados. También será más amplia la lista de candidatos, ya que se pretende obtener la mejor, en un rango de precios accesible. Al igual que con los componentes anteriores, la selección de la Unidad de Medidas Inerciales se hará por medio de una serie de criterios, en los que se ponderará las características de los candidatos a evaluar.

Por ser un componente tan trascendental en el resultado de la presente investigación, se realizó una prueba de desempeño de cada una de las Unidades de Medidas Inerciales. En esta prueba de se colocó en una galleta todos los componentes anteriormente seleccionados, los cuales son, el Arduino Nano y el Xbee S2 y las distintas Unidades de Medidas Inerciales. Posteriormente la galleta con todos los dispositivos, a la cual se referirá como el sensor inercial, ya que tiene todos los componentes que se consolidarán en el sensor final, se colocó sobre una base. Esta base se colocó sobre un motor de corriente directa girando a una velocidad constante.

Figura 83. Simulador de giro



Para cada IMU se adecuó el algoritmo de Madgwick y se colocó el programa en el Arduino. El algoritmo de Madgwick, como se explicó en el marco práctico, es un algoritmo para el cálculo de cuaterniones a partir de las medidas la IMU. Los datos de la medición se enviaron en tiempo real a la computadora, para luego realizar una transformación hacia los ángulos de Euler. De los ángulos de Euler el único que varió, el cual es el ángulo que se mide por su giro en el eje Z, fue Yaw. Este ángulo teóricamente cambió de manera periódica sobre su eje, a velocidad constante, por lo que la medición debía representar en todos los casos una señal senoidal. Posteriormente, se realizó un análisis frecuencial de cada una de las mediciones, para evidenciar la calidad de la medición realizada. Los resultados de esta prueba se muestran a continuación.

Figura 84. Medición con SKU: SEN0140 a) Señal b) Espectro

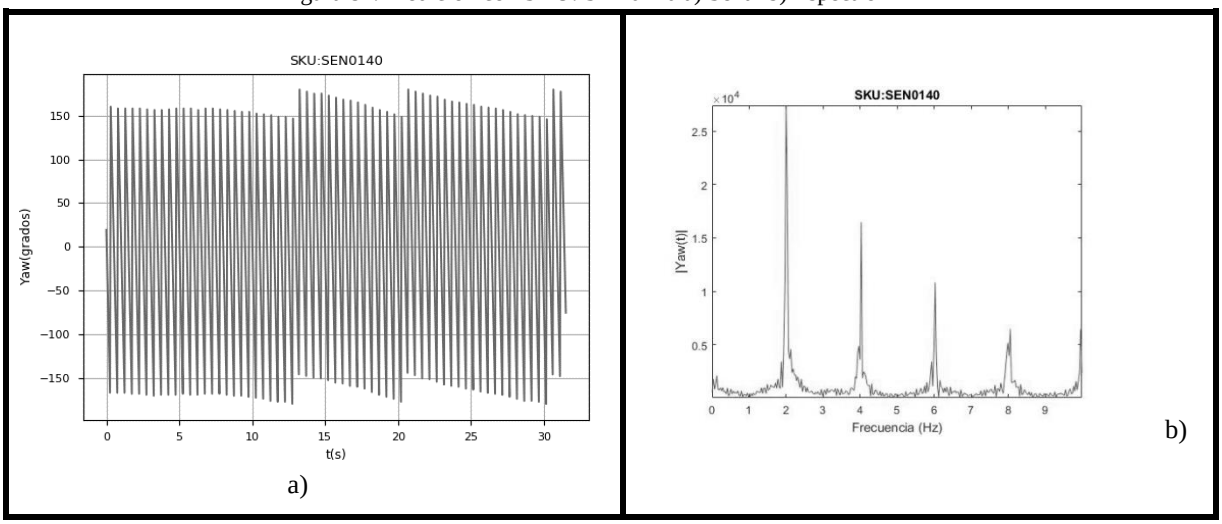


Figura 85. Medición con MPU6050 a) Señal b) Espectro

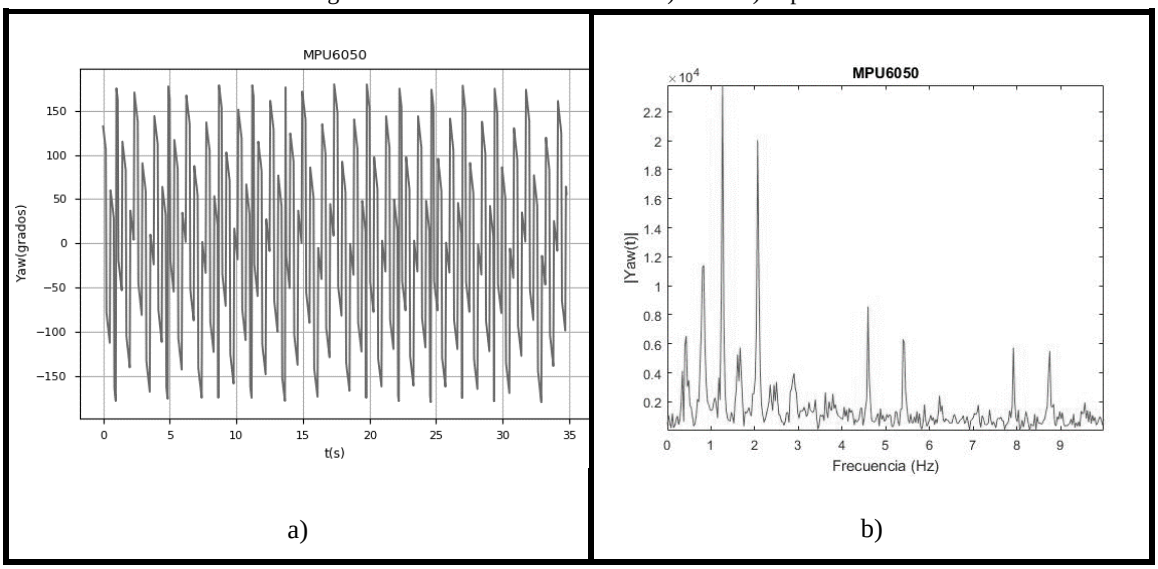
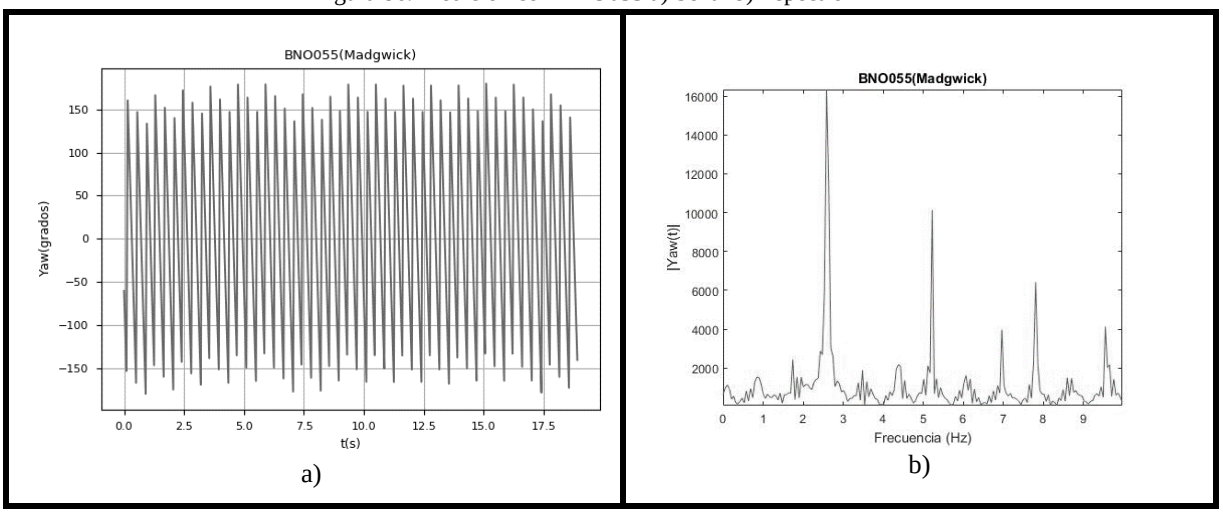


Figura 86. Medición con BNO055 a) Señal b) Espectro



Como se puede observar, se realizó por cada Unidad de Medidas Inerciales una medición, a la cuál también se le analzó el espectro por medio de TRF. En el análisis individual de cada IMU se discutirá cada resultado de este prueba.

a. DFRobot SKU:SEN0140. Esta es una Unidad de Medidas Inerciales muy completa, ya que cuenta con Acelerómetros Giroscopios y Magnetómetros. Esta IMU está diseñada para estabilizar drones y otros sistemas de alta precisión. Entre sus principales características se encuentran:

- Portabilidad: El tamaño de PCB es 26mm x 18mm
- Librerías: No cuenta con librería propia, pero se puede utilizar una Open Source para Arduino
- Precio: Incluyendo precio de importación Q400
- Disponibilidad: Esta IMU no está disponible en las electrónicas de Guatemala, por lo que se mandó a traer por internet.

- Desempeño: La prueba realizada a esta IMU tuvo un muy buen resultado, como se puede ver en la Figura 84. La señal muestra una forma periódica y en el espectro se puede contemplar que hay únicamente cuatro frecuencias que la componen. Es de mencionar que la Unidad de Medidas Inerciales se encontraba lo más centrada posible al eje de giro del simulador, pero como se puede observar, esto es físicamente imposible. Esta es una de las causas por la que no se encuentra únicamente un armónico en el análisis espectral de las señales. Pero el hecho que haya pocos armónicos es una señal de la precisión del sensor.

Figura 87. SKU: SEN0140

(DFRobot, 2017)

b. MPU6050. Esta es una Unidad de Medidas Inerciales para proyectos muy básicos, ya que únicamente cuenta con Acelerómetros y Giroscopios. Esta IMU está hecha para sencillos, sin necesidad de una mayor precisión, y por lo tanto es muy accesible. Sus principales características son:

- Portabilidad: El tamaño de PCB es 20.26mm x 15.56mm
- Librerías: Cuenta con librería propia para Arduino
- Precio: Incluyendo precio de importación Q150
- Disponibilidad: Esta IMU no está disponible en las electrónicas de Guatemala, por lo que se mandó a traer por internet.
- Desempeño: La prueba realizada a esta IMU no tuvo muy buen resultado, como se puede ver en la Figura 85. Esta es una IMU de bajo costo, por lo tanto, la construcción de sus sensores no es de muy buena calidad. La señal sí muestra una forma periódica, pero en el espectro se puede ver que hay muchas frecuencias que la componen, lo que indica que aparte de la medición se tiene mucho ruido.

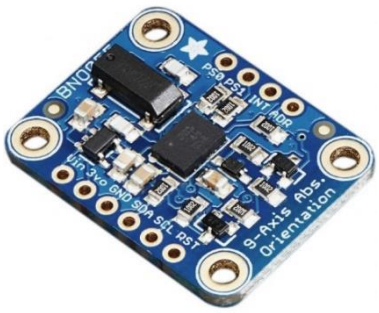
Figura 88. MPU6050

(Invensense, 2017)

c. BNO055. Esta es una Unidad de Medidas Inerciales muy completa. Esta cuenta con Acelerómetros Giroscopios y Magnetómetros. Esta IMU ha sido utilizada en múltiples investigaciones. Entre sus principales características se encuentran:

- Portabilidad: El tamaño de PCB es 26.67 mm x 20.32mm
- Librerías: Cuenta con librería propia para Arduino
- Precio: Incluyendo precio de importación Q480
- Disponibilidad: Esta IMU no está disponible en las electrónicas de Guatemala, por lo que se mandó a traer por internet.
- Desempeño: Con esta IMU se obtuvo muy buen resultado, como se puede observar en la Figura 86. La señal muestra una forma periódica y en el espectro se puede contemplar que hay únicamente cuatro frecuencias que la componen, al igual que con la BNO055.

Figura 89. BNO055



(Adafruit, 2017)

4. Matriz de decisión unidad inercial. Como se realizó con los otros dispositivos, se utilizaron ciertos criterios para la selección de la Unidad de Medición Inercial a utilizar. Entre los criterios de selección del dispositivo de transmisión inalámbrico a utilizar se encuentran: portabilidad, librerías, precio, disponibilidad y desempeño. El criterio de selección será el mismo:

Cuadro 14. Matriz de decisión de la IMU

Criterio	SKU: SEN0140	MPU6050	BNO055
Portabilidad	8	10	9
Librerías	7	10	10
Precio	9	10	8
Disponibilidad	10	10	10
Desempeño	10	2	10
Total	44	42	47

Finalmente se selecciona la IMU BNO055. Como se puede ver en la matriz de decisión, tiene un muy buen punteo en todos los criterios. La IMU de DFRobot (SKU: SEN0140) obtuvo también un muy buen punteo, pero el hecho de no tener una librería propia, le afectó su resultado.

5. Obteniendo la trayectoria. El gran objetivo de la presente investigación es obtener la trayectoria que realice el sensor inercial. Como se planteó anteriormente, los componentes del sensor ya están definidos, por lo que ahora se debe utilizar los algoritmos correctos para obtener una trayectoria congruente con las mediciones.

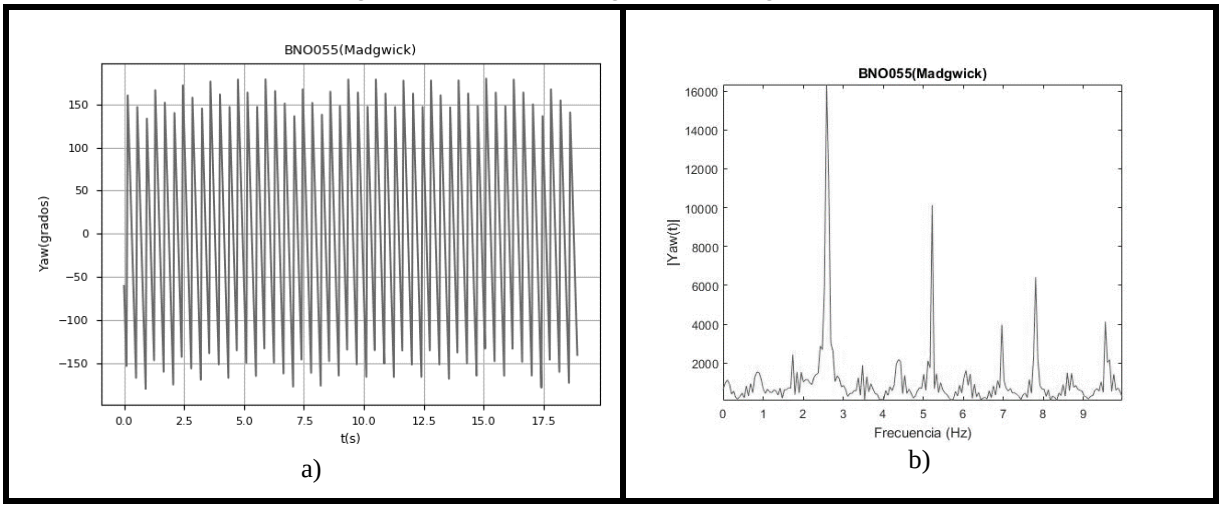
a. Programación del microcontrolador. El microcontrolador será el encargado de tomar los datos de la Unidad de Medición Inerciales, para luego procesarlas y mandar la información a la computadora, donde se realizará el cálculo de la trayectoria. El papel que desempeña dicho componente es muy importante, ya que las medidas que entrega la Unidad de Medición Inercial son con respecto a su marco local, es decir, que la IMU entrega exclusivamente aceleraciones de sus sensores, sin darle ninguna interpretación. Es en el Arduino donde se deberá tomar las mediciones y procesarlas para tener mediciones respecto a un marco global. Para ello el proceso a seguir es el siguiente:

- Primero se debe obtener las mediciones de los acelerómetros sin la gravedad
- Al mismo tiempo, se debe calcular la orientación del sensor, por medio de cuaterniones
- Luego se debe tomar las mediciones de los acelerómetros y rotarlas para obtener aceleraciones respecto a un marco global

b. Mediciones de los acelerómetros. Como se puede observar en la sección de anexos, al inicio del loop se obtiene el vector de los acelerómetros. Se obtienen utilizando una función desarrollada en la librería de la IMU BNO055. Esta función regresa en el vector indicado las aceleraciones. El único inconveniente con esta función es que éste vector no se puede operar con otros vectores, ya que es un vector de tipo IMU, por lo que, para obtener las mediciones en un vector operable, únicamente hay que extraer las mediciones de este vector, llamándolos por cada uno de los ejes: vector X, vector Y, y vector Z El gran inconveniente al leer las mediciones de los acelerómetros es que constantemente miden la gravedad. Es por ello que se debe restar la gravedad de las mediciones para saber el movimiento efectivo del sensor inercial. Afortunadamente, en la librería de la IMU desarrollaron una función para calcular la dirección de la gravedad en g. Es por esto que, al colocar las mediciones de los acelerómetros en un vector normalizado, se le resta a cada eje la medición de la gravedad. Es de esta manera como se logró obtener una medición de las aceleraciones experimentadas por el sensor inercial, sin la constante de la gravedad.

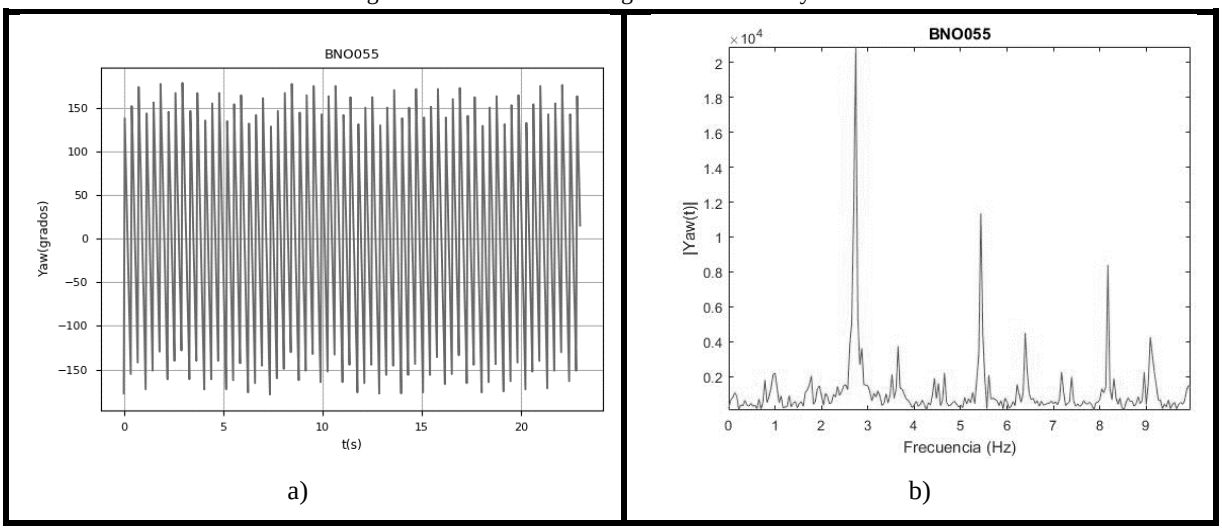
c. Obtención de la orientación. La orientación es una parte muy crítica, por lo que se le hará un especial énfasis. La orientación es la que servirá para tener aceleraciones respecto a un marco global, y no a un marco local. Es por ello que una pequeña diferencia en la orientación generará un desplazamiento al integrar en la dirección equivocada, y por consiguiente una trayectoria errónea. Es por ello que se realizó una prueba para seleccionar el algoritmo con mejor rendimiento. La prueba que se realizó es exactamente la misma que validó a la BNO055 como la Unidad de Medición Inercial a utilizar. Es decir, se colocó el sensor inercial en el simulador de giro y se evaluó la calidad de la medición. Únicamente, que en esta ocasión lo que varió entre una medición y la otra fue el algoritmo de cálculo del cuaternión en el microcontrolador. Los dos algoritmos que se utilizaron fue el de Madgwick y Mahony. Estos dos algoritmos toman las mediciones de la IMU y calculan en un proceso iterativo el cuaternión, que describe la orientación del sensor inercial. Los resultados de la experimentación fueron los siguientes:

Figura 90. Resultados con algoritmo de Madgwick



a) b)

Figura 91. Resultados con algoritmo de Mahony



Como se puede ver en la experimentación realizada, se realizó una corrida con el simulador de giro. La señal tiene un patrón muy parecido con los dos algoritmos, pero la diferencia se ve en el análisis frecuencial. Como se puede ver, con el algoritmo de Mahony se tiene la misma cantidad frecuencias principales que con el algoritmo de Madgwick. Pero las frecuencias menores tienen menos magnitud con el algoritmo de Mahony que con el de Madgwick. Por esta razón se utilizará el algoritmo de Mahony para el cálculo de la orientación. La IMU BNO055 en su librería tiene implementada una función para obtener el cuaternión. Esta función utiliza el algoritmo de Mahony, por lo que se utilizará esa función para obtener la orientación. Esto se puede ver en el loop del código, cuando se obtiene el cuaternión como la variable "quat" en un vector tipo IMU.

d. Rotando los acelerómetros. Teniendo las aceleraciones sin la gravedad constante, es decir, únicamente las aceleraciones de la medición y la orientación, se puede ahora obtener las aceleraciones respecto a un marco global. El proceso es sencillo, pero se necesita conocer álgebra de cuaterniones para realizar la operación. Básicamente se necesita realizar la operación que se observa en la Ecuación 5. El código se encuentra en el loop del anexo 2. Para el proceso primero se debe colocar el vector de aceleraciones en forma de cuaternión. Por lo que la primera componente del vector de cuatro componentes de las aceleraciones irá como cero. A continuación, se debe obtener el resto de vectores que se necesitan para realiza la operación. Por ello se obtiene el conjugado del cuaternión, el cual únicamente tiene negadas las tres últimas componentes del cuaternión original. Posteriormente, se multiplica el vector normalizado de las aceleraciones por el cuaternión.

Dicha multiplicación ahora se multiplica por el cuaternión conjugado. El resultado de dicha multiplicación será un vector de cuatro componentes, en el cual las últimas tres componentes serán las aceleraciones respecto al marco global, y como verificación la primera componente del vector debe ser cero. Este vector es el que posteriormente se envía junto con el cuaternión por protocolo serial, para ser procesado en la computadora.

6. Procesamiento de datos. Mediante la utilización de Xbees se envían los datos del Arduino en tiempo real a la computadora, en la cual se realiza el procesamiento debido para obtener la trayectoria. Este programa grafica en tiempo real la trayectoria, permitiendo ver el movimiento de sensor inercial conforme se realiza el movimiento. Éste programa tiene varios componentes importantes, los cuales se describen a continuación.

a. Obtención de los datos. Los datos se leen en tiempo real en la computadora por medio del puerto serial, por lo tanto, debe haber alguna manera de guardar el historial de toda la medición para poder realizar la trayectoria completa. Es por ello que, como se puede observar en el Anexo 1, se guardan directamente los datos en un archivo de texto llamado Datos.txt. Posteriormente se obtiene el archivo total de las mediciones y se crean arreglos para tener por separado cada medición, y luego otro arreglo para tener las aceleraciones en sus tres ejes y los cuatro componentes de los acelerómetros en un solo arreglo. Luego se guarda la medición de cada eje en un arreglo individual para realizar los cálculos de cada medición individualmente.

b. Obtención de la posición. Como se puede observar en la sección de anexos, lo que sigue es integrar los datos para obtener la velocidad, y luego la posición. Antes de esto se realiza un criterio con la magnitud del vector de aceleraciones, aquí se obtiene la magnitud de la aceleración y se utiliza la siguiente condición: si la magnitud es menor a 0.5 m/s2 no se toma en cuenta la medición y se guarda cero en la velocidad y la posición anterior en el vector de posición. Este criterio es para evitar desfases en la trayectoria cuando no hay movimiento. Posteriormente, si existe movimiento se utiliza el algoritmo de integración de Euler para calcular la velocidad y posición actual a partir de las mediciones anteriores. Es de mencionar que existen varios algoritmos que utilizan más iteraciones para calcular la integración, pero la diferencia en la trayectoria fue indistinguible, por lo que se utilizó el algoritmo de Euler.

Todo esto se realiza dentro de una función llamada “gen”, la cual tiene como parámetro de salida una matriz que contiene la trayectoria de cada uno de los ejes. Al final de este código se utiliza una función llamada “animation”, la cual genera la trayectoria en tiempo real, con un efecto de animación y por último se muestra dicha gráfica.

c. Calibración de flitros. Las mediciones, como se mencionó en el marco teórico, tienen información del movimiento, pero tienen ruido y desfases. Es por ello que se realizó una experimentación para determinar si es posible eliminar el ruido y los desfases mediante el uso de filtros. Como se planeó en el marco teórico, existen muchos tipos de filtros, pero como el objetivo es deshacerse de las frecuencias que son ruido, las cuales son frecuencias altas, y también de los desfases, que son frecuencias bajas, se necesitará un filtro pasa banda. De la variedad de filtros pasa banda se necesita uno que tenga una banda de paso angosta, ya que las frecuencias de las mediciones varían entre 0 a 10 Hz. Es por esto que se utilizará el filtro FIR (Firwin), cuya respuesta frecuencial se muestra en la Figura 90. Primero se quería realizar un set de mediciones con atletas, para que el movimiento realizado fuera lo más metódico y correcto posible.

La experimentación constaba de tres partes, en las cuales se recorría una distancia total de 100 metros en línea recta en la pista del estadio Doroteo Guamuch. Se tomó una muestra de tres participantes entre quince y dieciocho años que practicaran alguna rama de atletismo. Cada atleta debía recorrer los cien metros en tres modalidades: caminando, corriendo y trotando. Para cada modalidad se realizaron dos mediciones. En total se obtuvo dieciocho mediciones, a las cuales se les realizó un análisis frecuencia mediante la TRF para determinar las frecuencias de corte de los filtros a utilizar. A continuación, se muestra una imagen de la experimentación realizada:

Figura 92. Mediciones en el estadio Dotoreo Guamuch

C. Dispositivo emisor de luz RGB para tiempo de reacción

Para la realización de este dispositivo se realizó una investigación detallada acerca de las ventajas, de los estudios y de los países que emplean la ciencia en los deportes para mejorar el rendimiento de los atletas. Así mismo, también se investigo acerca de la psicología de la ciencia aplicada al deporte que se encarga de aplicar técnicas de retroalimentación (Biofeedback). Así como las ventajas que estas presentan en el deporte a nivel mundial y los distintos métodos que se pueden utilizar, los cuales ayudan a los atletas a desarrollarse en las distintas disciplinas.

1. Reunión con psicólogos. Se realizaron reuniones con psicólogos de las instalaciones para las cuales se realizará el proyecto de ciencia aplicada al deporte. Con el fin de Proponer y formular los objetivos que se desean cumplir durante la elaboración del prototipo. También se buscó seleccionar los deportes a los cuáles se desea que los atletas mejoren su rendimiento durante las competencias.

2. Trabajo de campo. Se realizaron visitas a los establecimientos para conocer las necesidades de cada uno de los entrenadores y psicólogos de los deportes seleccionados. Posteriormente se visitaron las instalaciones donde los atletas entrenan para observar y tener un mayor conocimiento acerca de su disciplina, y así proponer ideas que se pudieran implementar para poder optimizar los entrenamientos.

3. Investigación del mercado. Se realizó una segunda investigación detallada acerca de dispositivos, plataformas y componentes que ayuden a la realización de los prototipos, tomando en cuenta el factor de bajo costo. Se buscaron dispositivos ya fabricados que realicen las mismas funciones que se desea tener en el prototipo y así realizar una comparación de las características de cada uno y determinar si el prototipo cumple con las especificaciones deseadas. Finalmente se investigó acerca de formas más sencillas para la elaboración del prototipo que implicaran un bajo costo de fabricación.

4. Elaboración del prototipo. Se realizó un prototipo sencillo y que sea fácil de utilizar para los usuarios. Se inició trabajando con el diseño de la parte electrónica el cual consiste en construir un interruptor que desactive el LED buscando formas sencillas pero que sean llamativas para el usuario, investigando distintas formas de realizar el interruptor; todo el circuito está alimentado y conectado al Arduino. En las siguientes dos imágenes se muestran los diagramas eléctricos de las propuestas de interruptores.

Figura 93. Diagrama eléctrico LED con pushbutton

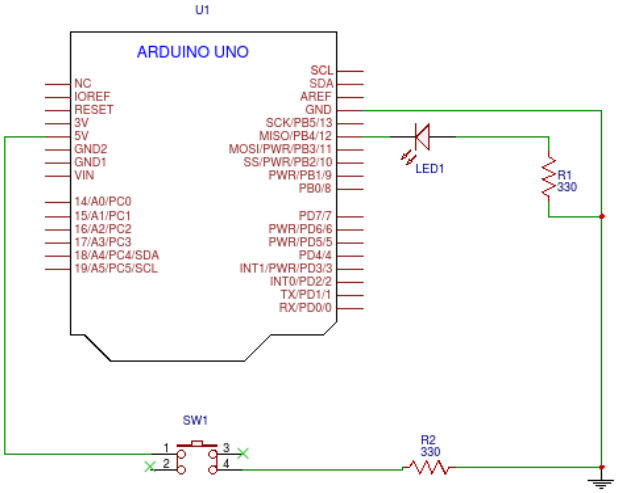
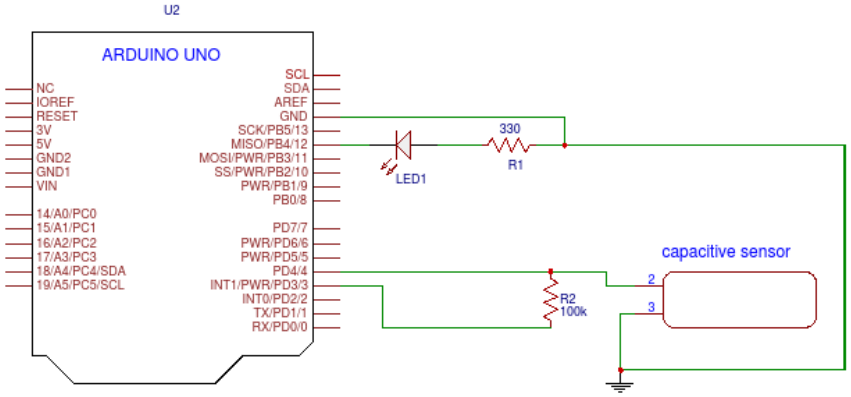


Figura 94. Diagrama eléctrico LED con sensor capacitivo



Posteriormente se trabajó la interfaz la cual debía ser amigable para el usuario, utilizando un software gratuito como lo es Python, Processing, Java, entre otros. Para graficar se realizó un ejemplo de cómo sería la interacción con el usuario. Se procedió a la creación de una base de datos la cual debía contener los datos obtenidos de las pruebas realizadas en los atletas. La base se crea en un archivo de Excel porque es un programa que dispone de muchas herramientas para interpretación de datos, utilizando métodos estadísticos.

Finalmente se incorporó el circuito y el aparato electrónico ya aplicado en uno de los módulos del proyecto el cual consiste en la comunicación inalámbrica el prototipo y la computadora que almacenará la base de datos.

Figura 95. Diagrama de conexión de Xbee a computadora

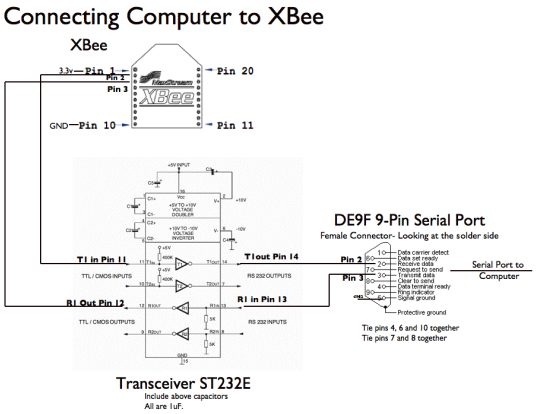
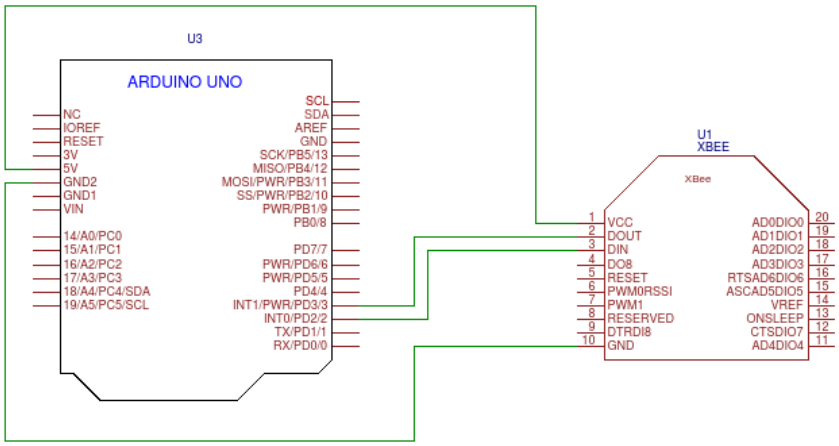


Figura 96. Diagrama de conexión de Xbee a Arduino



5. Realización de pruebas y obtención de datos. Se realizó una serie de pruebas que permitiera verificar la funcionalidad del prototipo, así como también pruebas que permitieran diferenciar entre utilizar un interruptor normal y un sensor capacitivo. Adicionalmente se verifico que el tiempo obtenido fuera correcto al igual que el despliegue y almacenamiento en la base de datos. Finalmente se realizó un análisis de esfuerzos en el prototipo para los escenarios importantes en donde se utilizaría. A partir de los resultados se verificó el cumplimiento de los objetivos propuestos.

D. Sensor tiro con armas de caza

Para lograr construir el equipo que logra medir tiempo de reacción y de disparo, se tuvieron diversas reuniones los representantes de la COG, representantes de tiro, psicólogo y entrenador, y atletas. En dichas reuniones se establecieron los parámetros de diseño, se presentaban los prototipos para su validación y se realizaban pruebas con atletas para la calibración del sensor y revisar el funcionamiento correcto del equipo. Los criterios establecidos por los representantes de tiro fueron los siguientes, el equipo no debe interferir en el campo de visión del atleta, el sensor utilizado no debe molestar al atleta, el peso del equipo no debe afectar el balance de la escopeta. El equipo no debe tener dimensiones mayores a 20x6 centímetros, se debe contar con algo que despliegue los resultados de manera inmediata, los resultados deben ser almacenados para su posterior análisis, la transmisión de datos del equipo debe superar el diámetro de la cancha de skeet. Finalmente, el equipo debe ser capaz de funcionar en modalidad fosa y skeet.

Partiendo con la información planteada con anterioridad se estableció que el equipo debía poseer un dispositivo capaz de medir el tiempo de reacción y de disparo en ambas modalidades, con la capacidad de despliegue y almacenamiento de datos, que no afectase al atleta en su campo de visión y balance. Para ello primero se estableció un mapa mental sobre la lógica para medir tiempos de reacción y disparo sin importar el sensor a utilizar, dicho mapa mental partió de la premisa de que el atleta debe realizar dos o tres movimientos para realizar uno o dos disparos. Es decir, el primer movimiento representa la reacción,

entendiéndose este como el momento en el que el atleta realiza un movimiento con la escopeta debido a que ya posee su objetivo identificado dentro de su campo visual, y un segundo y tercer movimiento correspondientes a él o los disparos realizados para acertar el objetivo. En dicho mapa mental se estableció que el programa debía tener una variable para identificar estos movimientos para clasificar cada tiempo.

1. Selección de sensor. Partiendo de la premisa de que la solución debía ser capaz de medir el tiempo de disparo de los atletas se realizó una lluvia de ideas con los posibles sensores que establecerían el momento en el que el atleta realizaba el disparo. Luego de tener la serie de sensores propuestos se procedió a seleccionar aquellos que se consideraban fáciles de montar en la escopeta o el atleta, los sensores seleccionados fueron los siguientes. Sensor de fuerza colocado en la culata de la escopeta, sensor de flexión colocado en el dedo índice del atleta, sensor infrarrojo colocado en el cañón de la escopeta y un acelerómetro colocado en la escopeta.

Como se mencionó anteriormente dentro de los parámetros de diseño se encontraba establecido que no se debía afectar la visión del atleta ni colocar ningún sensor que les provocará alguna molestia, por ello se procedió a descartar los sensores de flexión e infrarrojos. El sensor de flexión fue descartado ya que para ser colocado en el dedo del atleta se debía realizar un guante o un dedal en el cual se montará el sensor y sacar los cables de referencia y salida hacia el controlador, pudiendo esto afectar la conducta del atleta, según el psicólogo de tiro. El sensor infrarrojo se descartó ya que para ser colocados en el cañón se debían realizar modificaciones a la escopeta, o bien realizar un acople para la misma que los pudiera contener, modificando así el campo visual natural del atleta. Adicionalmente se tenía como otro parámetro que los tiempos que se debían desplegar eran los de disparo y reacción con ello se descartó el sensor de fuerza, ya que este solo sería capaz de dar el tiempo de disparo, por lo que se procedió a seleccionar un acelerómetro.

2. Módulo de comunicación y microcontrolador. Teniendo el sensor seleccionado, se realizó un trade study sobre los módulos de comunicación investigados. Los criterios utilizados en dicho estudio fueron los siguientes, costo, precio, dimensiones, alcance, documentación disponible, disponibilidad y compatibilidad. Los primeros cuatro criterios partieron de los parámetros de diseño mencionados con anterioridad mientras que los otros partieron de un parámetro implícito, el tiempo, ya que los prototipos debían estar finalizados con la suficiente antelación para poder contar con la participación de los atletas en las pruebas de los mismos.

A cada uno de los criterios mencionados previamente se les asigno un peso. En el Cuadro 15. se puede observar el valor que se le asignó a cada criterio siendo 1 lo menos importante y 9 lo más importante. Posteriormente se le asignó un valor a cada módulo de comunicación con respecto a cada criterio, dicho valor se dividió dentro de la suma de todos los valores asignados al módulo, a ese valor se le denomino punteo. Ese valor se multiplicó por el peso establecido previamente y ese dio como resultado el punteo pesado, el

cual se utilizó para seleccionar la opción más conveniente. En el Cuadro 16. se observan los módulos seleccionados y el valor que obtuvo cada uno. Los módulos Bluetooth y Xbee S2 fueron los que mejor puntearon en el estudio por lo que se procedió a realizar pruebas con cada uno de los módulos.

Cuadro 15. Peso por criterio

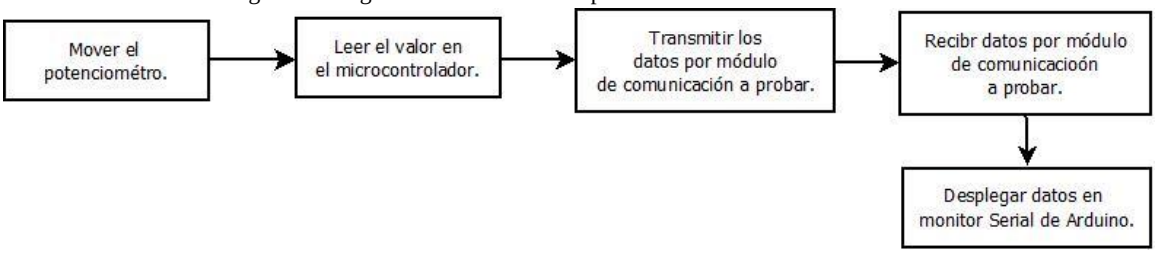
Criterio	Peso
Costo	8
Alcance	7
Documentación Disponible	5
Disponibilidad	8
Dimensiones	9
Peso	7
Compatibilidad	6

Cuadro 16. Trade study, módulos de comunicación

Criterio	Peso	Bluetooth		NRF24L01		Xbee S2		RF 433MHz	
		Punteo	Punteo Pesado	Punteo	Punteo Pesado	Punteo	Punteo Pesado	Punteo	Punteo Pesado
Costo	16%	0.1316	0.0211	0.1143	0.0183	0.0667	0.0107	0.1176	0.0188
Alcance	14%	0.0526	0.0074	0.1429	0.0200	0.2000	0.0280	0.1176	0.0165
Documentación Disponible	10%	0.1842	0.0184	0.2000	0.0200	0.1556	0.0156	0.2059	0.0206
Disponibilidad	16%	0.2368	0.0379	0.0286	0.0046	0.2000	0.0320	0.0294	0.0047
Dimensiones	18%	0.0263	0.0047	0.1429	0.0257	0.0889	0.0160	0.1471	0.0265
Peso	14%	0.1316	0.0184	0.1143	0.0160	0.0889	0.0124	0.1176	0.0165
Compatibilidad	12%	0.2368	0.0284	0.2571	0.0309	0.2000	0.0240	0.2647	0.0318
Sumatoria			0.1363		0.1354		0.1387		0.1353
Punteo Final			98.30		97.66		100.00		97.57

Estas pruebas consistían en seguir los pasos listados en la Figura 97. El valor establecido por la posición del potenciómetro sería leído por un microcontrolador enviado por el módulo de comunicación con el que se estaban realizando las pruebas. Este valor sería capturado por el receptor y desplegado en el monitor serial del IDE de Arduino. En la sección de anexos se encuentra el código utilizado en estas pruebas. Finalmente, la parte transmisora se fue alejando cada vez un metro de distancia, con el fin de determinar hasta que distancia existía comunicación en los módulos. Estas pruebas se realizaron en un entorno al aire libre, en dónde no existían obstáculos que interfirieran entre los módulos.

Figura 97. Lógica de funcionamiento pruebas módulos de comunicación.



a. Feather 32u4. Primero se realizaron pruebas con el microcontrolador Feather 32u4, en donde el transmisor fue el módulo bluetooth incorporado a la placa y el receptor el bluetooth incorporado en el ordenador.

b. HC-05. Con el módulo HC-05 se realizaron dos variantes de prueba. En la primera prueba se sustituyó el microcontrolador por un Arduino Uno, al cual se le conecto el módulo transmisor y el módulo receptor se conectó a un USB-TTL. En la segunda prueba se conectó el receptor a un Arduino Nano. Para configurar los módulos HC-05 se utilizaron los comandos AT.

c. Xbee S2. La última prueba que se realizó fue con los módulos Xbee S2, se realizaron pruebas con dos topologías de red. Una con la topología punto a punto en d configurar los Xbee se utilizó el Software XCTU y la guía presente en la Figura 98. Posteriormente se verificó el funcionamiento de los módulos en el mismo software.

Figura 98. Parámetros de configuración módulos Xbee.

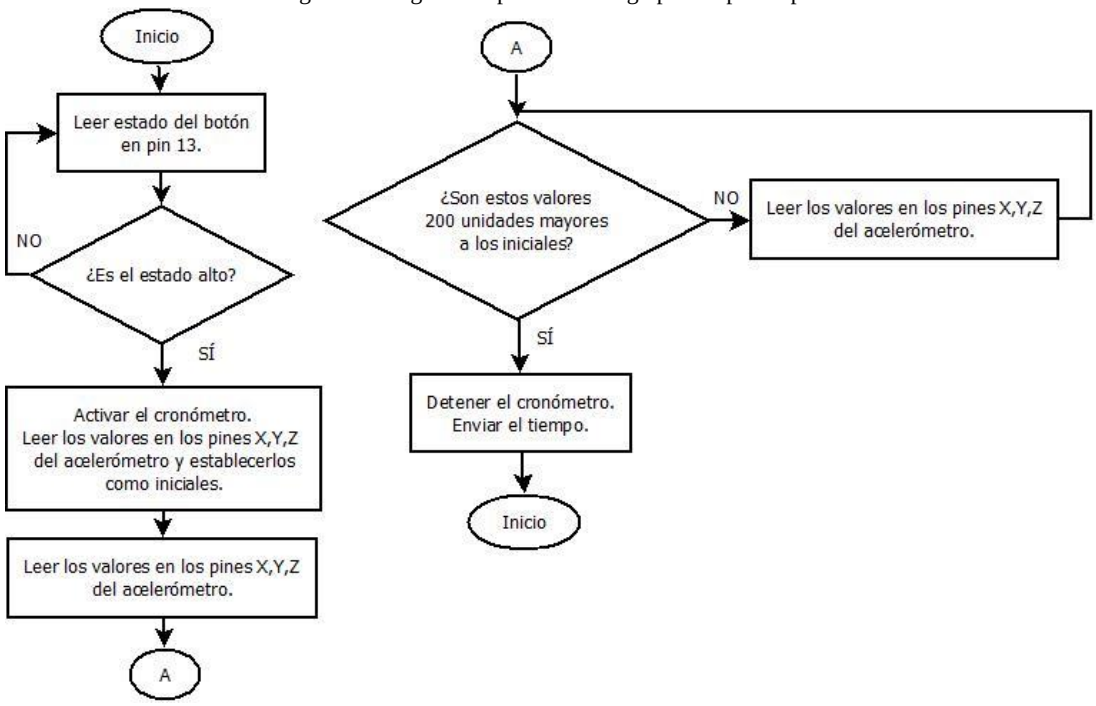
XBee A Valores	XBee B Valores
DH 13A200	DH 13A200
DL 4076E267	DL 4076E26E
MY AAAA	MY AAAA
SH 13A200 (viene por defecto)	SH 13A200 (viene por defecto)
SL 4076E26E (viene por defecto)	SL 4076E267 (viene por defecto)
CE 1 -Coordinator	CE 0 -End Device Serie 1 Pro

Se procedió a seleccionar el módulo que habría logrado un mayor alcance. Ya con el sensor y los módulos de comunicación seleccionados se procedió a seleccionar los microcontroladores que se utilizarían, para ello se seleccionaron las dos placas Arduino utilizadas. El arduino Nano estaría montado en la escopeta y el Arduino Uno emularía la máquina que lanza los discos. Ambas placas fueron seleccionadas debido que se tomaron en cuenta los criterios de disponibilidad, es decir se contaba con las placas físicamente, documentación disponible y lenguaje de programación. El lenguaje de programación que las placas utilizan es de alto nivel por lo que, las pruebas se estarían elaborando en el menor tiempo posible.

Al tener los microcontroladores, sensor y módulo de comunicación seleccionad se procedió a realizar diversas iteraciones del prototipo en las siguientes secciones de este capítulo se describen detalladamente cada uno de ellos.

3. Primer prototipo. El primer prototipo consistió en implementar la lógica presente en la Figura 99. para ello se conectó un botón al pin 13 del Arduino Nano el cual simulaba el lanzamiento de un disco permitiendo así la activación del cronometro. Este prototipo solo era capaz de detectar tiempo de reacción el cual se enviaba por el Serial del Arduino hacia la computadora y era desplegado por el monitor Serial de Arduino IDE. Este prototipo fue utilizado para explicar al entrenador y psicólogo de tiro la idea general sobre el funcionamiento del equipo que se deseaba desarrollar.

Figura 99. Lógica de operación código primer prototipo.



4. Segundo prototipo. En este segundo prototipo se agregó la activación del cronometro de manera remota, utilizando un Arduino Uno, al cual se le trasladó el botón del Arduino Nano. Al Arduino Uno se le implementó la lógica de la Figura 100. Al arduino Nano se le modificó la lógica anterior agregándole la habilidad de poder detectar tiempo de disparo 1, esta nueva lógica se observa en la Figura 101. Para lograr la activación del cronómetro como el despliegue de información en el ordenador se utilizaron tres módulos Xbee, conectados en una red circular. En la Figura 102. se observa la configuración de los mismos.

Figura 100.Diagrama de flujo Arduino Uno prototipo 2.

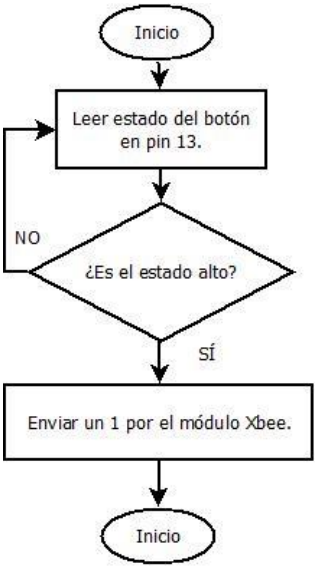


Figura 101. Diagrama de flujo Arduino Nano prototipo 2.

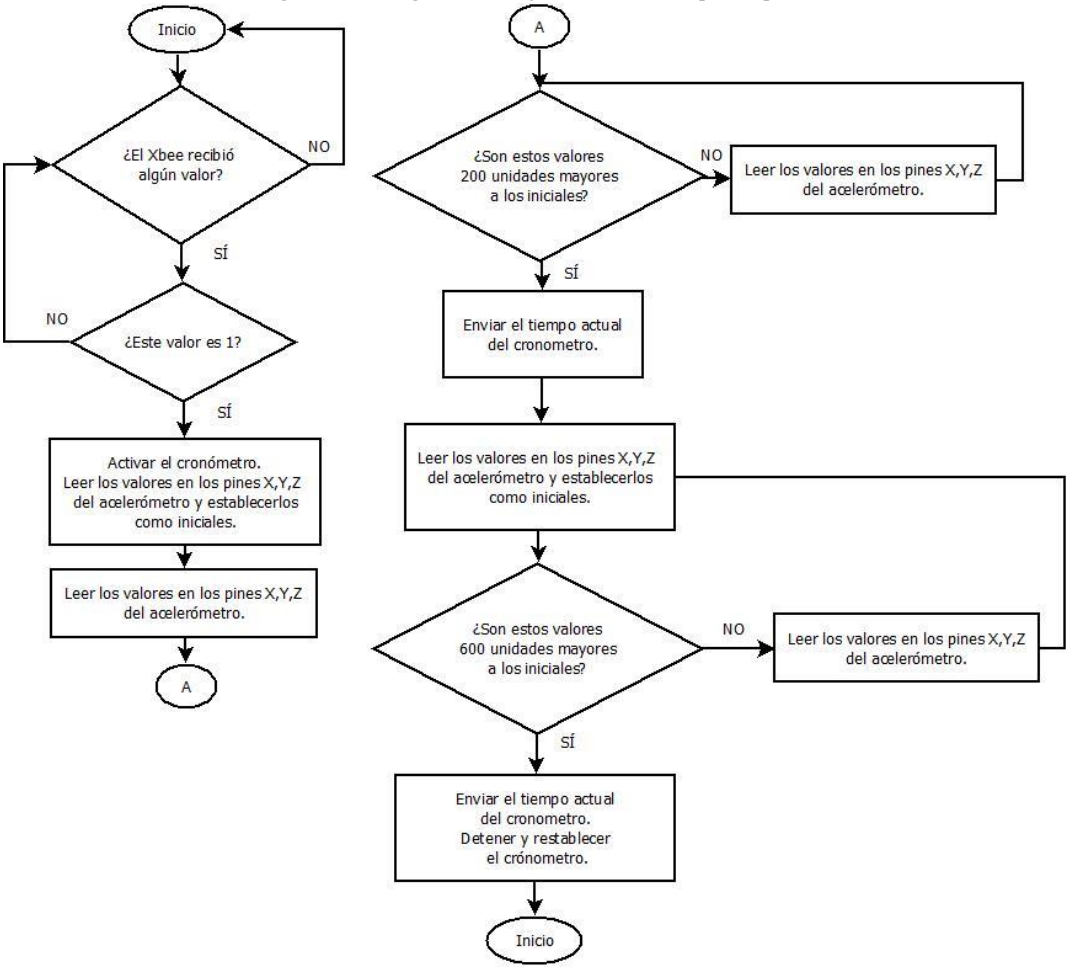
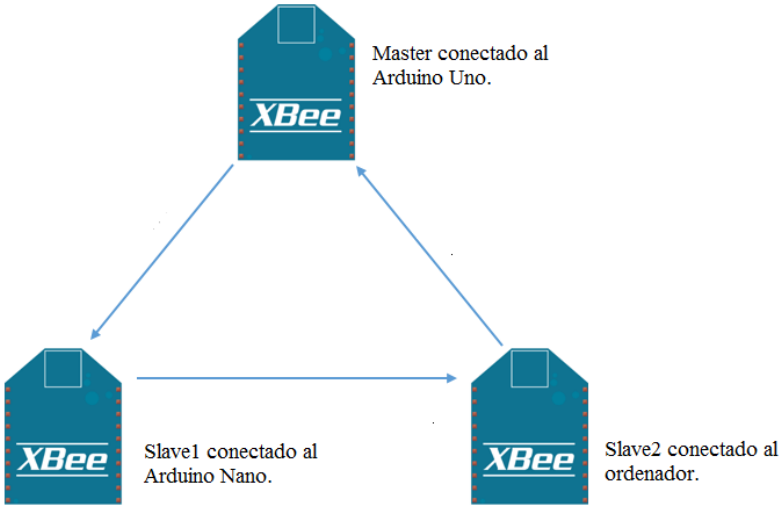
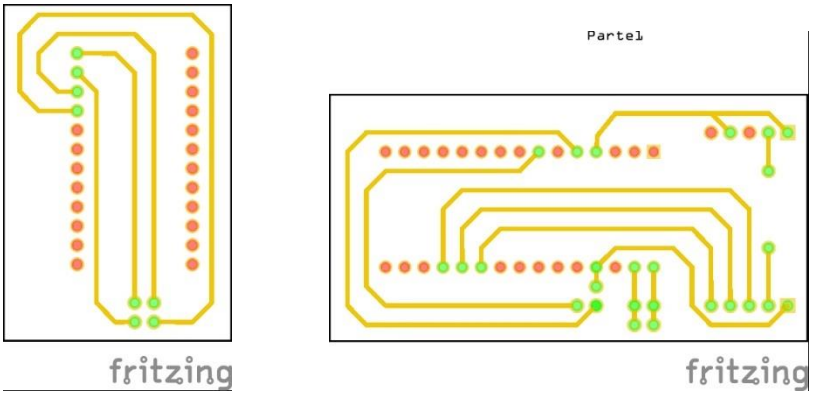


Figura 102. Configuración módulos Xbee prototipo 2.



Para verificar el funcionamiento se realizaron pruebas con Jean Pierre Brol, atleta de la modalidad Fosa. Para realizar estas pruebas se diseñaron dos PCB's en el software Fritzing, uno para portar el módulo Xbee conectado al Arduino Nano y otro para portar el Arduino y acelerómetro. Para definir el ancho de las pistas se midió la corriente entregada por la batería y se utilizó una calculadora en línea. En la Figura 103. se pueden observar ambos diseños de las placas. Estas placas se fabricaron utilizando el método de planchado del PCB.

Figura 103. Diseño de placa prototipo 2. A) Placa Xbee B) Placa Arduino Nano y acelerómetro.



A) B)

5. Tercer prototipo. Este prototipo utilizó la topología anterior de los Xbee's, así como el PCB. Las variantes en este prototipo radicaban en el Arduino Uno ya que se le agregó un botón que permitía el cambio entre la modalidad Fosa y Skeet, otro para indicar si el disparo realizado se consideraba válido y un tercer

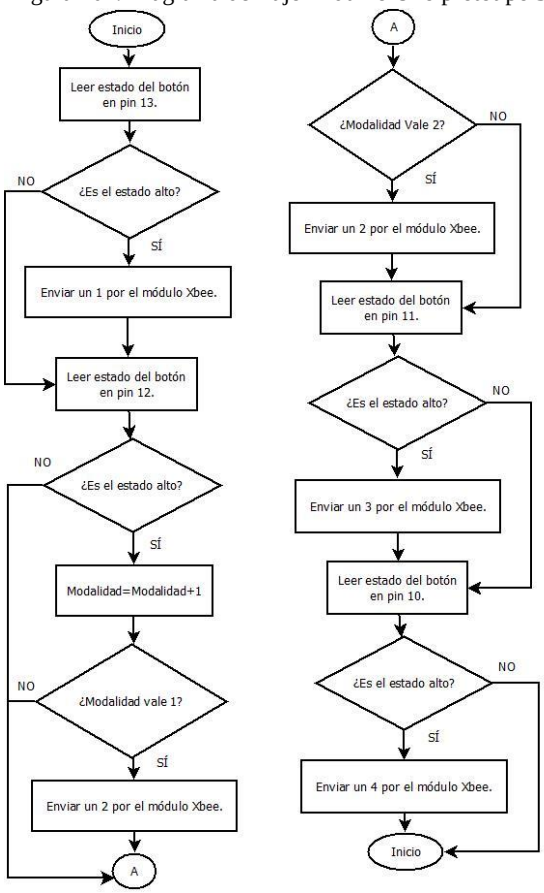
botón que enviaba los siguientes datos: cantidad de discos lanzados, cantidad de disparos realizados, cantidad de disparos acertados. El diagrama de la Figura 104. muestra la nueva lógica de operación del

Arduino Uno. En el código del Arduino Nano se agregaron nuevas condicionales que permitían ejecutar funciones determinadas en dependencia de los caracteres recibidos, así mismo a la modalidad Fosa se le agregó la opción de detectar dos disparos. En el siguiente cuadro se la función que, a ejecutar en base al carácter recibido, estos dígitos corresponden al valor ASCII del carácter enviado.

Cuadro 17. Funciones a ejecutar en el Arduino Nano prototipo 3.

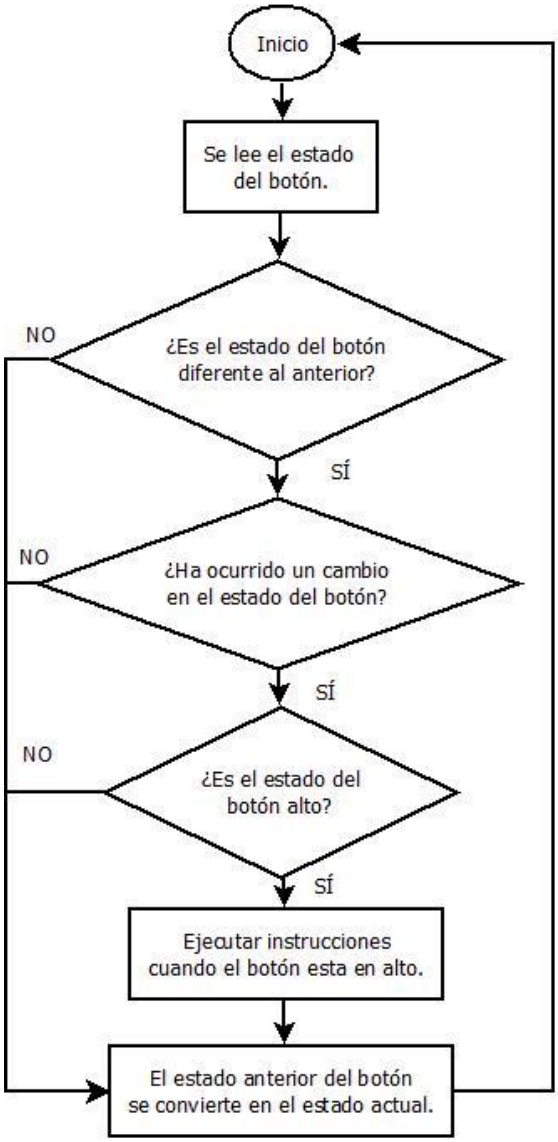
Carácter Recibido por Arduino Nano	Función que ejecutar
49	Activar el cronómetro.
50	Establecer la modalidad Fosa como la modalidad para la toma de datos.
51	Establecer la modalidad Skeet como la modalidad para la toma de datos.
52	Incrementar en número de aciertos.
53	Enviar los siguientes resultados: -Cantidad de discos lanzados. -Cantidad de discos acertados. -Cantidad de disparos realizados.

Figura 104. Diagrama de flujo Arduino Uno prototipo 3.



6. Cuarto prototipo. Este prototipo utilizó el mismo código de operación que el anterior, con la variante que los botones utilizados se sustituyeron por una interfaz realizada en el software Processing y el objeto PrintWriter, ya que este prototipo buscaba almacenar los datos en un documento de Excel. A esta interfaz se le agregaron cuadros de texto que permitieron el despliegue de los datos recibidos. Adicionalmente se sustituyó el filtro RC utilizado por un debounce implementado en código por Arduino en la Figura 105. se puede observar la lógica de este código. Para qué el usuario recibiera una confirmación de que el dispositivo se encontraba conectado y que estaba listo para trabajar la modalidad deseada se pintaron los botones de color verde, luego que el Arduino Uno recibía un carácter que indicaba que el Arduino Nano iba a ejecutar la función deseada.

Figura 105. Diagrama de flujo debounce botón.



Para que los datos recibidos se desplegaran en los cuadros correspondientes a las funciones mencionadas previamente del Arduino Nano se le agregaron caracteres para identificar el cuadro en donde se debe desplegar el dato correspondiente. En el Cuadro 18. se observa el carácter identificador y su dato correspondiente.

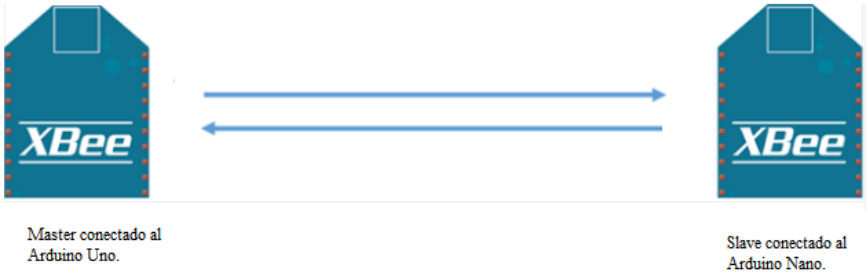
Cuadro 18. Carácter identificador con su dato correspondiente.

Carácter Identificador	Dato Recibido
R	Tiempo de Reacción
U	Tiempo de Disparo 1
X	Tiempo de Disparo 2
L	Cantidad de discos Lanzados
W	Cantidad de discos Acertados
Z	Cantidad de disparos realizados

A diferencia de los prototipos anteriores este solo contó con dos módulos Xbee en topología punto a punto ya que se utilizó el Arduino Uno y el puerto Serial como la interfaz para transmitir los datos a desplegar en el ordenador, esto se logró debido a que se decidió que el Arduino Uno ya no se iba colocar directamente en la máquina sino se colocaría en el cerebro de la misma. En la

Figura 106. se observa el diagrama de la topología utilizada. Este prototipo se probó con el entrenador Pedro Zaya en la modalidad Fosa, y con el atleta Rodrigo Zachrisson en la modalidad Skeet.

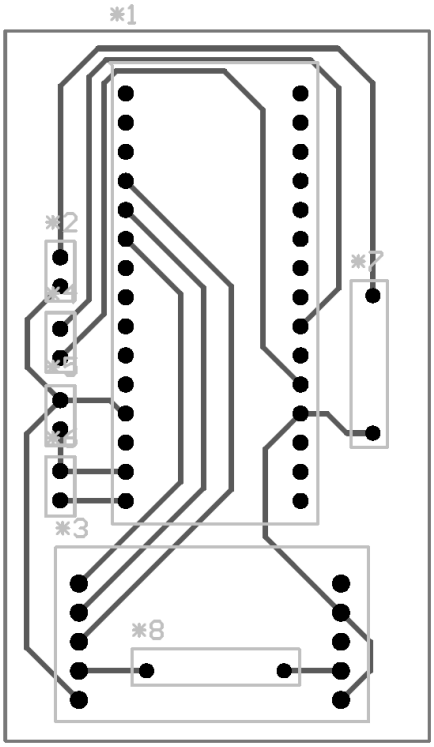
Figura 106. Configuración módulos Xbee prototipo 4.



7. Quinto prototipo. En este prototipo se sustituyó la interfaz gráfica desarrollada en Processing por una nueva interfaz realizada en JAVA utilizando el software de Netbeans, el cual facilitó la creación de la interfaz. A esta interfaz se le realizó una variante la cual consistió en agregar un cuadro de texto para que se colocará el nombre del atleta y un botón para enviar los datos, el cual enviaría los datos a un nuevo Excel en caso no existiese o bien agregaría una nueva hoja en caso ya existiese el Excel y en esta nueva pestaña se desplegarían los datos. En la sección de Anexos se encuentran los diagramas de flujo de la interfaz gráfica y de operación del código de este prototipo. Así como el código desarrollado.

Este prototipo contó con una nueva placa diseñada en Altium Designer, para diseñarla se basó en la placa anterior y solo se realizaron modificaciones para poder conectar un LED a la misma. En la Figura 107. se observa el diseño del nuevo PCB. Este prototipo se probó con el atleta Juan Ramón Schaeffer en la modalidad de Skeet y con el atleta Dany Brol en la modalidad de Fosa, para la modalidad de Fosa se realizó una prueba realizando un disparo y otra realizando dos disparos.

Figura 107. Diseño PCB prototipo 5.



VI. RESULTADOS

A. Sensor Inercial

Como se planteó anteriormente, se realizó una experimentación en la cual se obtuvo la trayectoria de los atletas en tres distintas modalidades: corriendo, trotando y caminando. A las mediciones de los acelerómetros de cada eje se les hizo un análisis frecuencial para determinar los patrones de frecuencias que se puedan identificar. A partir de estos patrones se busca filtrar la señal para obtener un mejor resultado en la obtención de la trayectoria. Cada una de las mediciones de cada tipo se colocó concatenada una después de la otra en un archivo de texto, para realizar un análisis que fuera válido para cada uno de los atletas estudiados. Posteriormente se utilizó el programa de TRF para realizar el análisis frecuencial correspondiente. A continuación, se muestran los resultados de dicho análisis:

Figura 108. Medición de eje X en la modalidad caminando

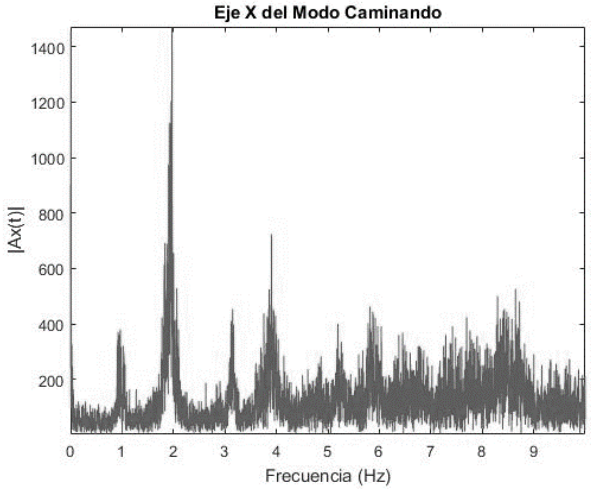


Figura 109. Medición de eje Y en la modalidad caminando

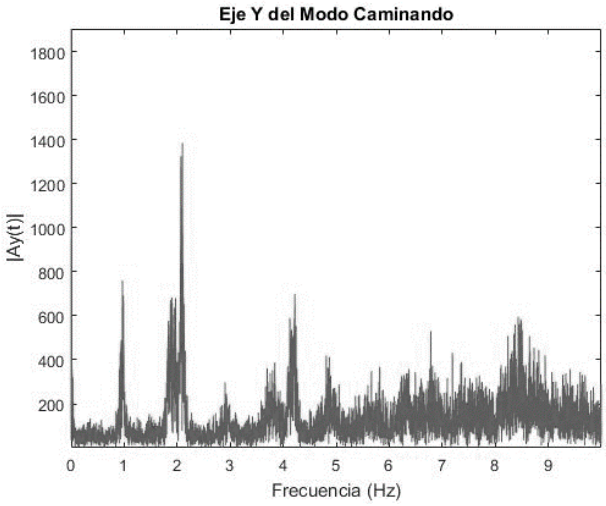


Figura 110. Medición de eje Z en la modalidad caminando

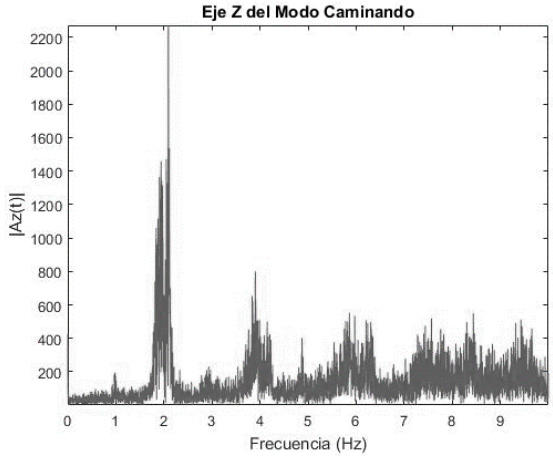


Figura 111. Medición de eje X en la modalidad trotando

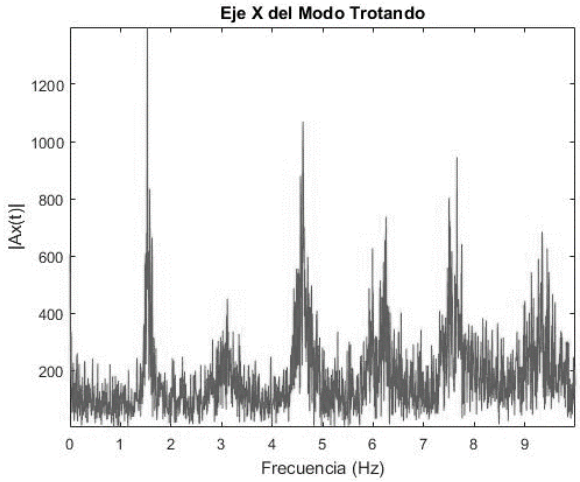


Figura 112. Medición de eje Y en la modalidad trotando

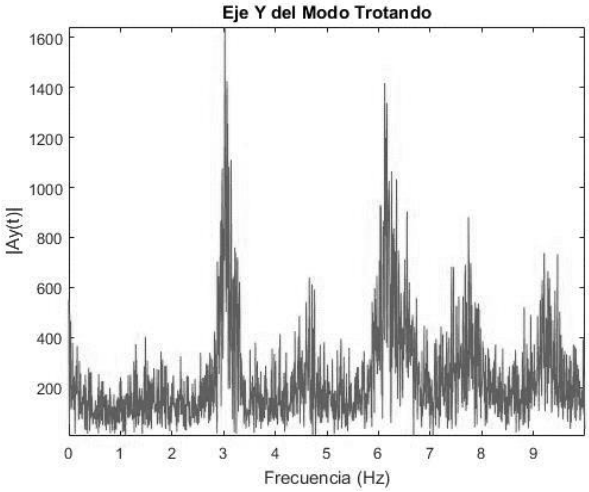


Figura 113. Medición de eje Z en la modalidad trotando

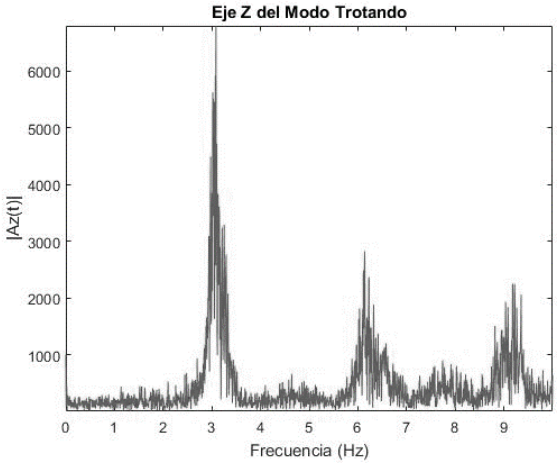


Figura 114. Medición de eje X en la modalidad corriendo

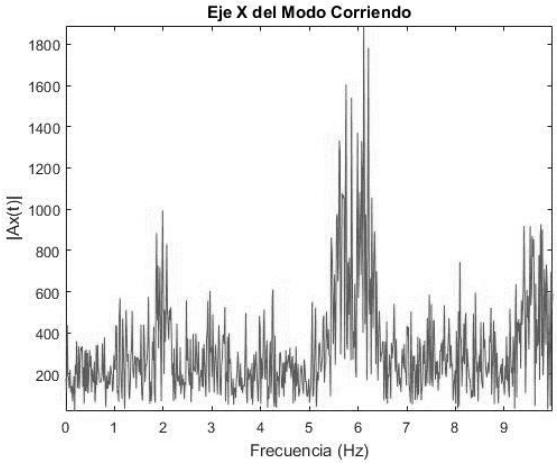


Figura 115. Medición de eje Y en la modalidad corriendo

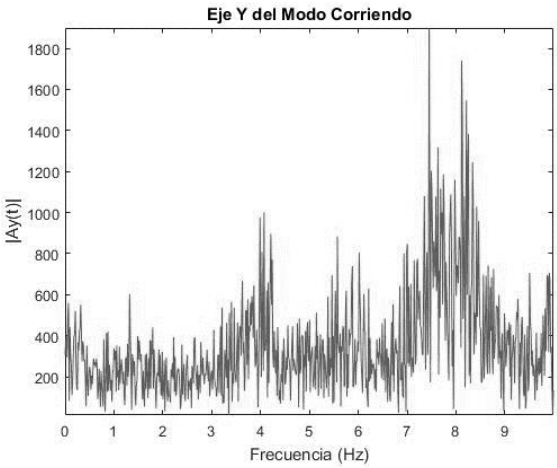
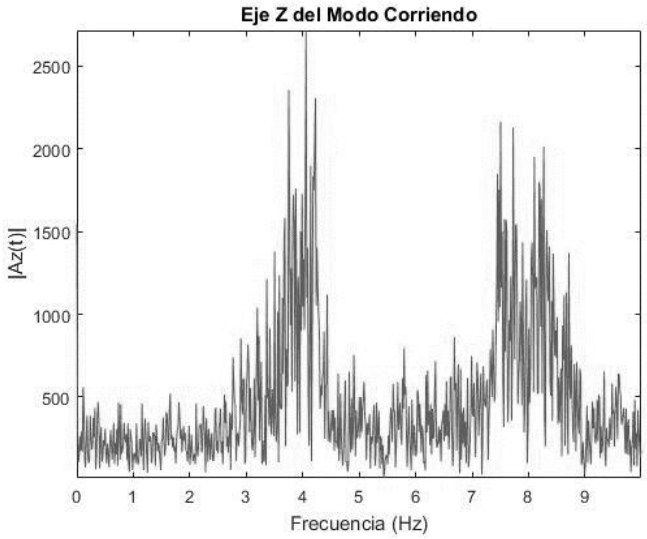
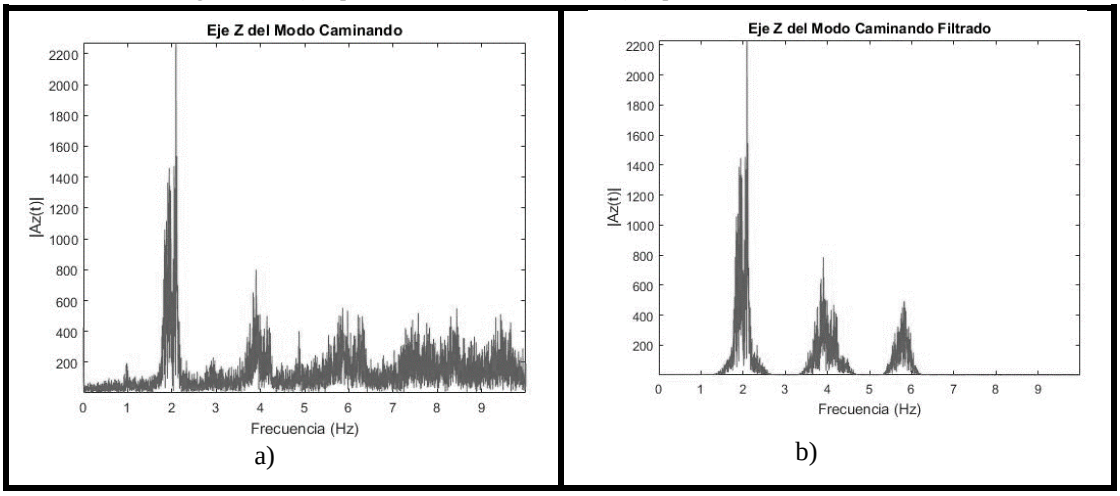


Figura 116. Medición de eje Z en la modalidad corriendo



Como se planteó con anterioridad, el objetivo de estas gráfica era identificar las frecuencias principales que caracterizan al movimiento y rechazar las que no. Esto para obtener un movimiento sin ruido ni desfases que repercutan en una trayectoria errónea. Primero se debe identificar el eje x y el eje y deben tener los mismos filtros, ya que las trayectorias no se realizarán en un solo eje, sino en los dos. El eje z será el más fácil de filtrar, porque ya se sabe el resultado que se debe obtener, que es un desplazamiento final nulo, con oscilaciones, las cuales son los pasos que realiza la persona. Por ello se debe establecer el rango deseado de frecuencias que se van a permitir pasar, para filtrarlas de esta manera. A continuación, se muestra gráficamente lo que se pretende lograr al filtrar las mediciones mediante un análisis frecuencial.

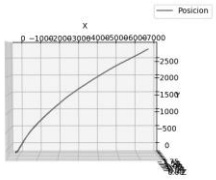
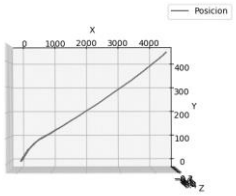
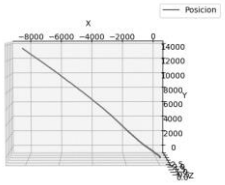
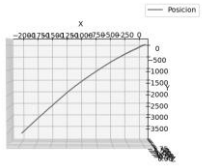
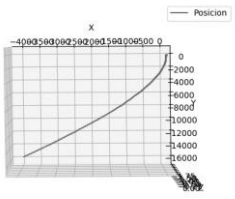
Figura 117. a) Espectro de los datos medidos b) Espectro de las mediciones filtradas



Al observar las gráficas del análisis realizado con la TRF se procedió a realizar la calibración de los filtros correspondiente. El enfoque de los mismos será en las frecuencias armónicas de la modalidad caminando. Se implementarán los filtros en base a la información observada en las gráficas. A continuación, se analizará el efecto de los filtros en la trayectoria final.

Las trayectorias obtenidas se deben analizar para adecuar los filtros a las frecuencias que contienen información, y desechar las frecuencias que son ruido o que generan desfases. Para ello primero se analizarán las trayectorias con las técnicas descritas en el marco teórico, es decir, la magnitud de trayectoria y el coeficiente de correlación. En base a ello se filtrarán los datos y luego se analizarán los datos de nuevo. Como se planeó anteriormente, el porcentaje de error se obtendrá entre la magnitud estudiada y una trayectoria de 100 m. Y el coeficiente de correlación se obtendrá entre los ejes X y Y, que son los dos ejes que tendrán la trayectoria.

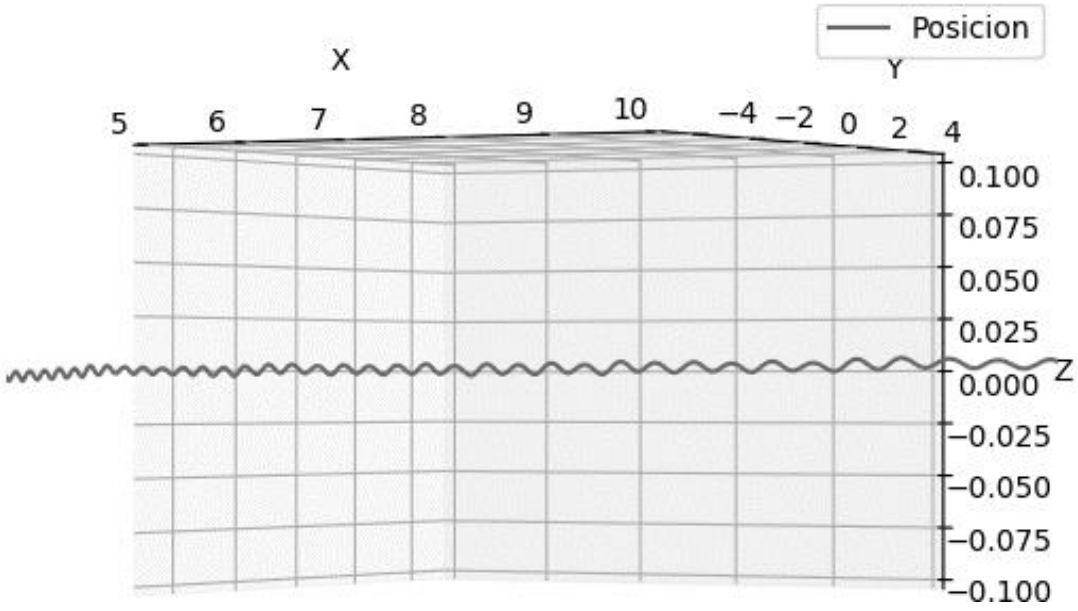
Cuadro 19. Análisis de trayectorias sin filtrar

Prueba	R^2	Longitud	% Error
	0.97187	7231m	7131
	0.99373	4395m	4295
	0.99862	15871m	15771
	0.99080	4292m	4192
	0.94830	16515m	16415

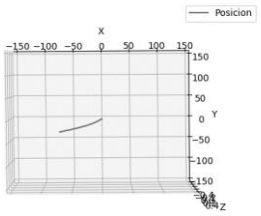
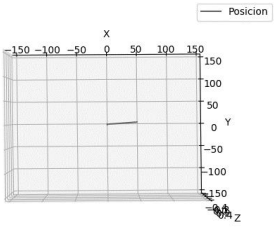
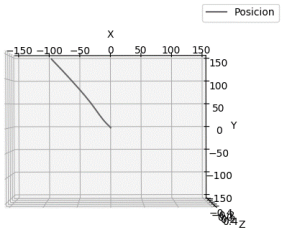
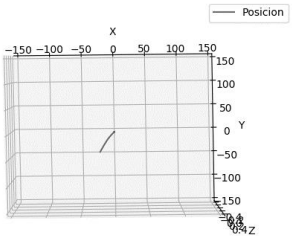
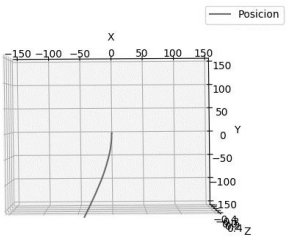
Cuadro 20. Frecuencias de corte de los filtros

Eje por Filtrar	Frecuencia de Corte de Pasa Bajas	Frecuencia de Corte de Pasa Altas
X	4.5 Hz	6.5 Hz
Y	4.5 Hz	6.5 Hz
Z	3.5 Hz	6.5 Hz

Figura 118. Acercamiento para ver oscilaciones en el eje Z



Cuadro 21. Análisis de trayectorias filtradas

Prueba	R^2	Longitud	% Error
	0.97184	77.93m	22.07
	0.99439	52.41m	47.59
	0.99862	149.03m	49.03
	0.99091	56.43m	43.57
	0.94837	145.34m	45.34

B. Dispositivo emisor de luz RGB para tiempo de reacción

Desde la Figura 119 hasta la Figura 121 podemos observar un ejemplo de un posible programa cuya función es capturar los datos recibidos a través de la plataforma Arduino la cual contiene el tiempo que se tarda una persona en presionar un pulsador en respuesta a un estímulo. La Figura 119 y la Figura 120 muestran el programa principal el cual contiene la lógica para obtener dichos datos y la creación de la base de datos.

Como podemos observar en la Figura 119 la parte importante del programa que define la interfaz entre el usuario y el programa es la de ingresar el nombre del usuario a realizar las pruebas y el número de pruebas, ya que a partir de este es que definimos la estructura de nuestra base de datos la cual lleva el nombre del usuario. El número de pruebas define la cantidad de datos que capturará el programa y almacenará en la base de datos. Otro punto importante es la creación de estilos, ya que este nos ayuda a tener de manera organizado los datos para poder manejarlos al gusto del usuario, ya sea para aplicar métodos estadísticos, generar gráficas o simplemente almacenar los datos.

En la Figura 120 observamos la continuación del programa principal y lo más importante que se debe recalcar es que el mismo programa calcula el promedio de las pruebas, el valor máximo de los tiempos y el valor mínimo. Esto se realiza para tener un mejor manejo de los datos y para poder analizar factores que pudieron haber afectado al usuario, cabe mencionar que estos análisis serían elaborados por un experto que pueda interpretar los datos.

Por último, en la Figura 121 observamos una subrutina la cual es utilizada al finalizar el programa y consiste en generar un nuevo usuario si se desea o modificar uno ya existente. Importante explicar que sucede cuando ya existe un usuario y se desea realizar modificaciones, dichas modificaciones consisten únicamente en agregar una nueva corrida, es decir crear una nueva hoja en el archivo de Excel, el cual tiene como nombre la hora que se creó. Este programa como se ha mencionado con anterioridad es un ejemplo de cómo se vería un programa el cual es diseñado al gusto del usuario. Este programa cumple con una de las funciones principales de este prototipo y es poder capturar los datos recibidos a través de Arduino y almacenarlos en una base de datos.

Figura 119. Diagrama de flujo interfaz-usuario

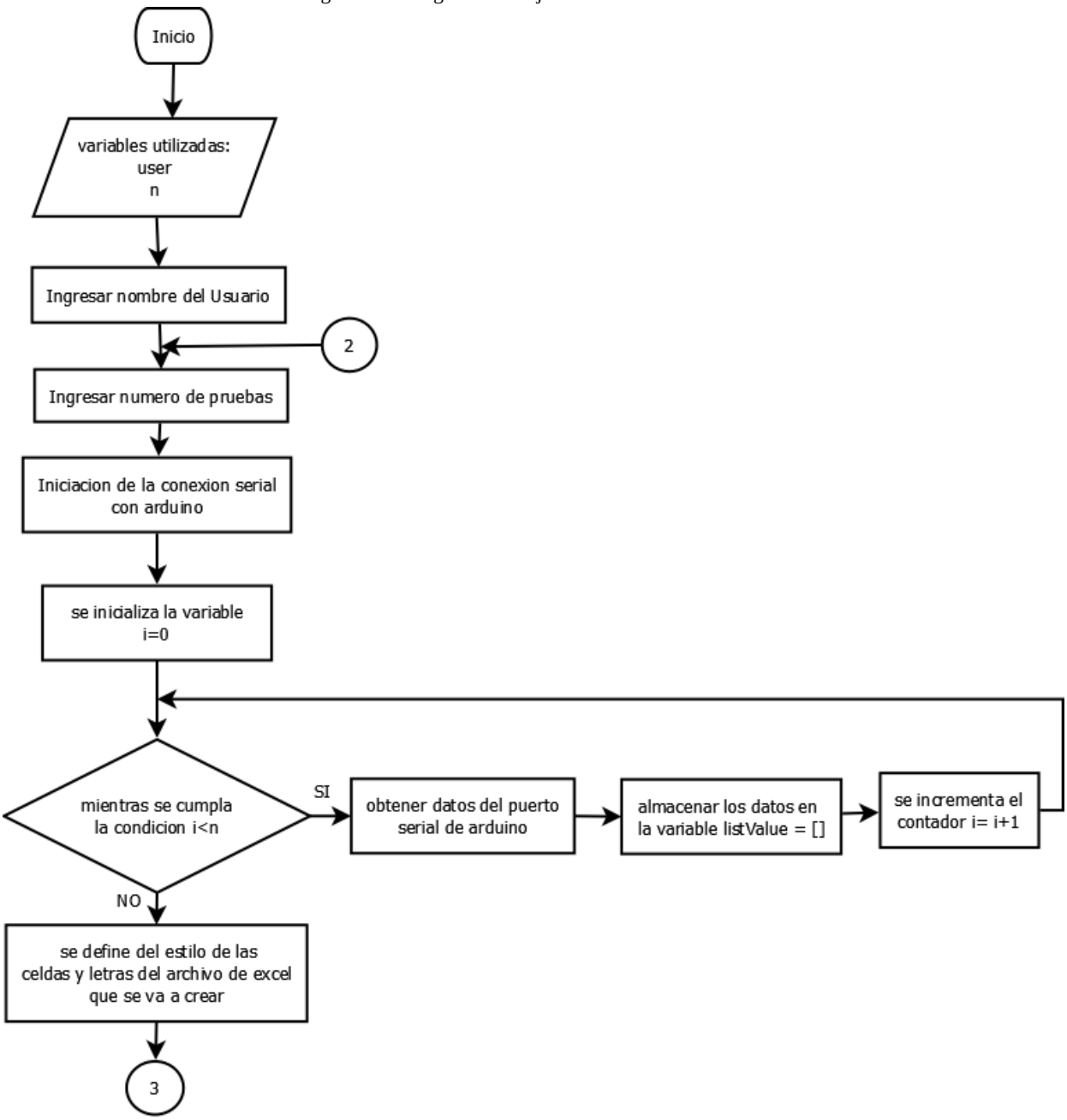


Figura 120. Continuación diagrama de flujo interfaz-usuario

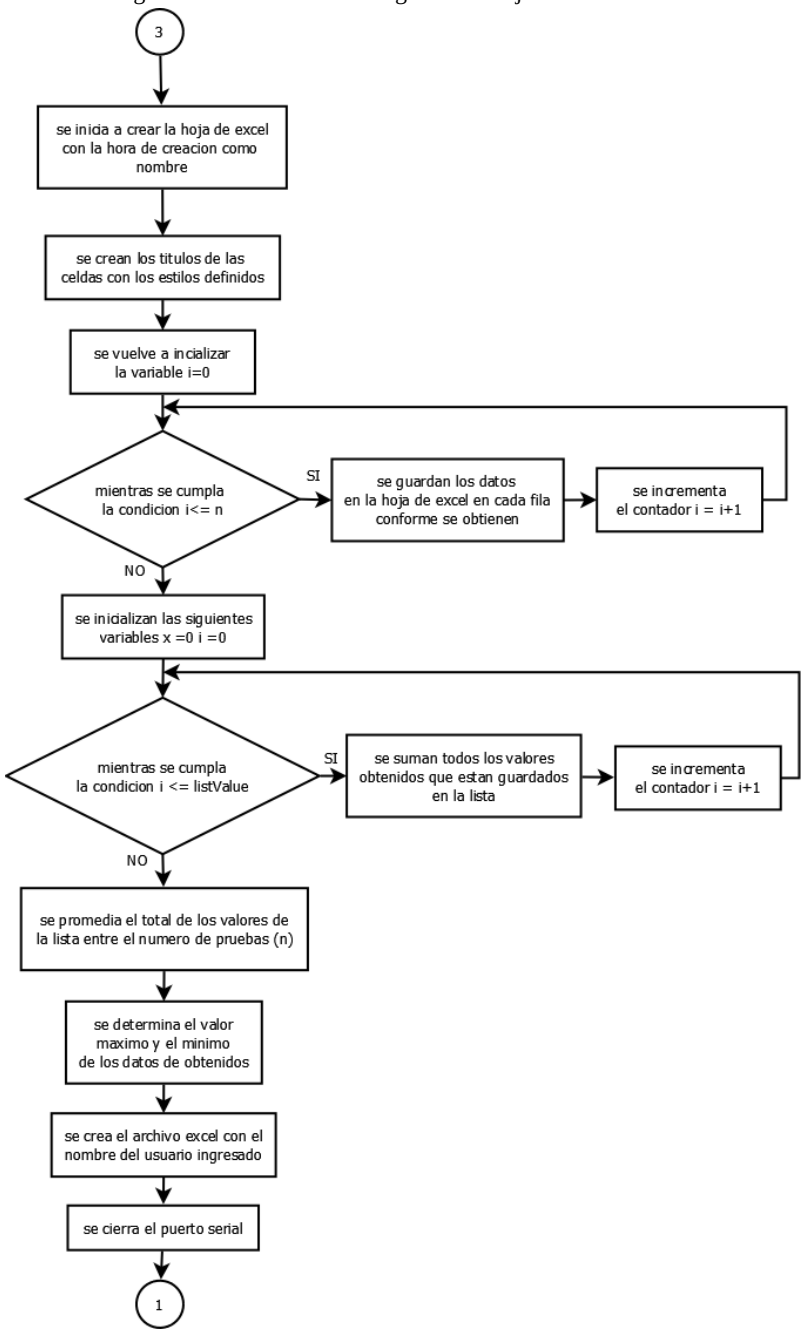
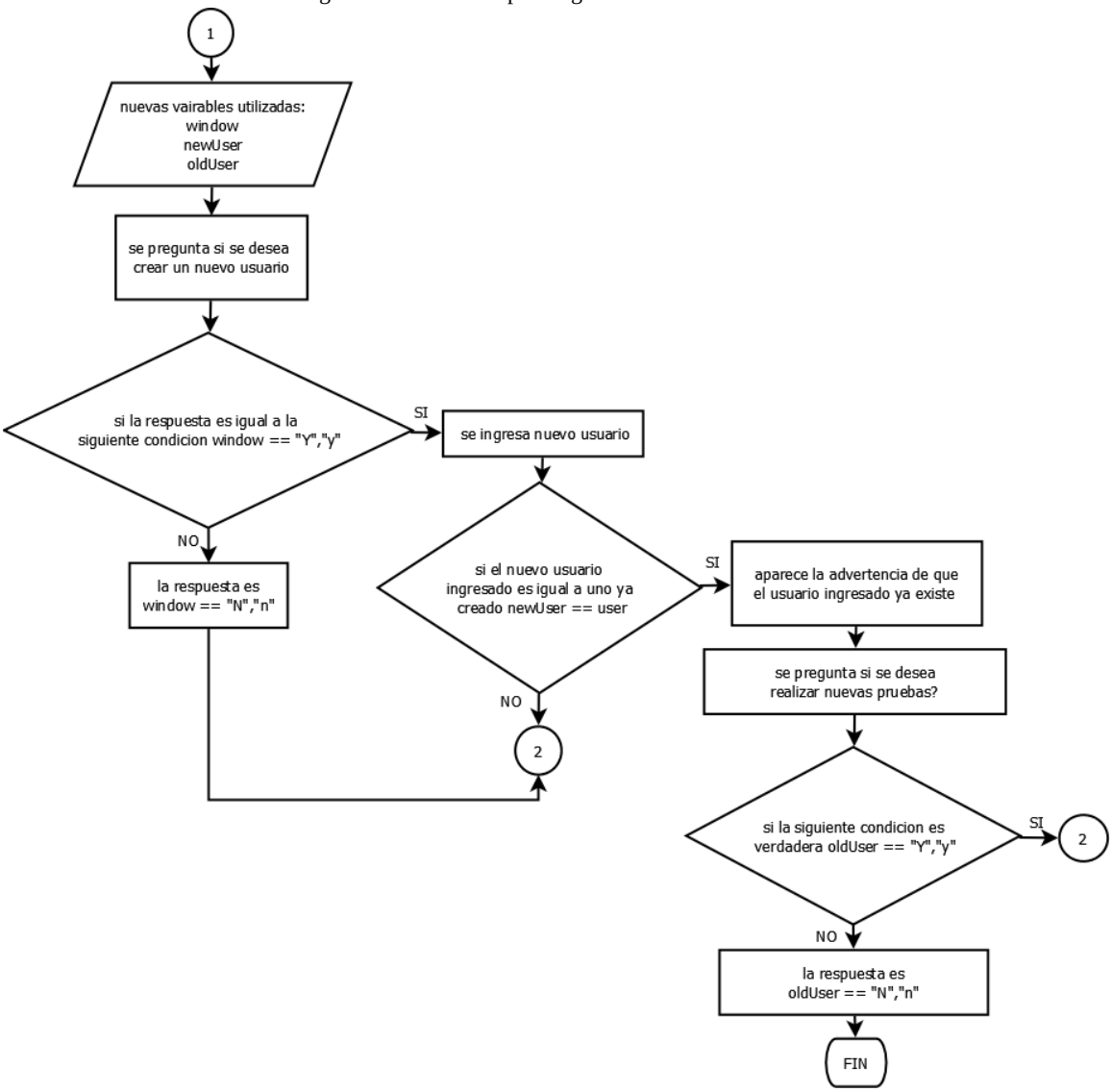


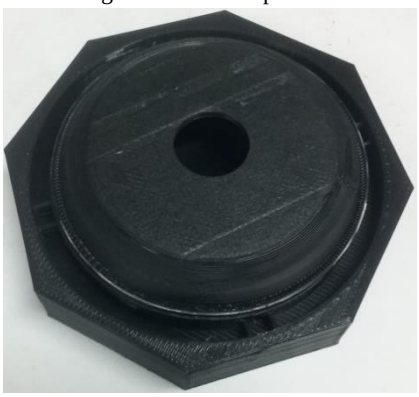
Figura 121. Subrutina para ingresar o modificar usuario



El Cuadro 22 muestra la comparación entre un dispositivo cuyas funciones son similares a las del prototipo, este es el caso del FitLight, en el cual se enumeran ciertas especificaciones importantes en común que describen ambos dispositivos. La primera característica comparada es el peso, ya que es un prototipo y los elementos y materiales que se utilizan son de bajo costo, dicha característica se ve afectada. Al comparar los tamaños observamos que estos tienen las mismas dimensiones las cuales representan una ventaja a la hora de comparar ambos dispositivos. Algo importante a recalcar en el prototipo es el rango inalámbrico que este tiene, ya que es mayor que el del dispositivo existente.

Algo importante a recalcar es el precio el cual se basa en los componentes utilizados y materiales para la elaboración del prototipo. El valor total de generar el prototipo tiene un aproximado de $130. El costo del Fitlight el cual es el dispositivo comparado tiene un valor aproximado de $400 por unidad.

Figura 122. Prototipo final



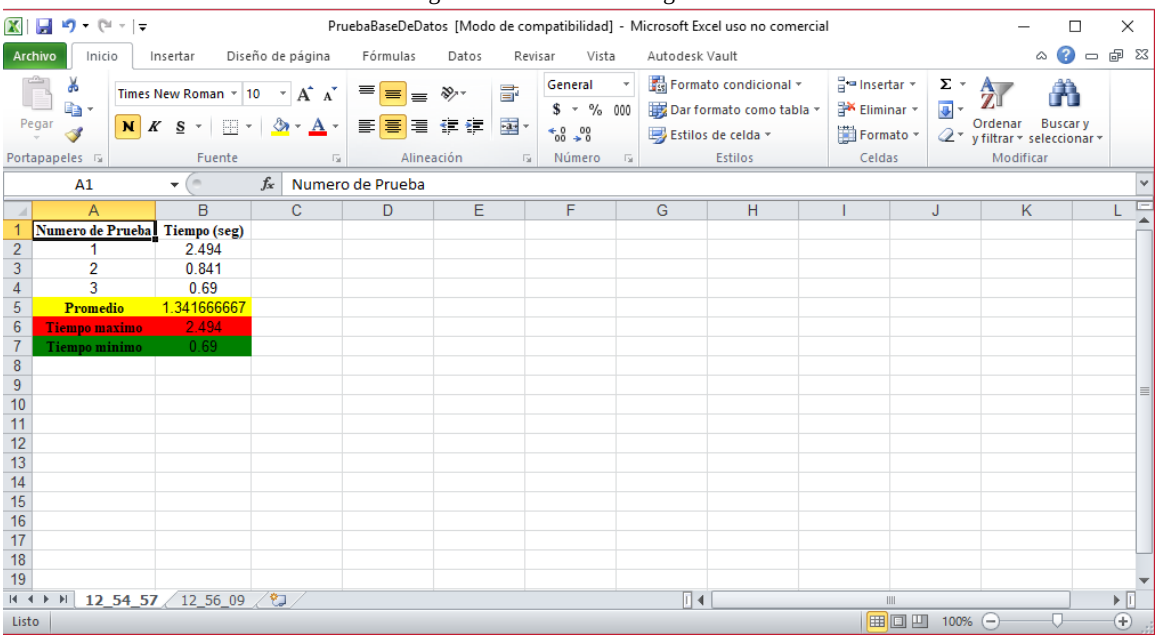
Cuadro 22. Comparación FitLight vrs. Prototipo

Especificaciones	FitLight	Prototipo
Peso	0.3 kg	0.11 kg
Tamaño	10 cm	10 cm
Software	Programable	Programable
Duración baterías	1.5 horas	1.12 horas
Rango inalámbrico	50 metros	60 metros
Costo (aproximado)	$400	$130

En la Figura 123 observamos la generación de la base de datos la cual es generada a través del software el cual se explicó anteriormente. En dicha base de datos se observa que este genera los títulos, los colores de las casillas, el formato de tipo de letra, el nombre del archivo el cual tiene como referencia el nombre del usuario y el nombre de las hojas que tienen la hora en la que se generó la base de datos. Por otro lado, se observa que el programa genera e inserta el promedio, el tiempo máximo y el tiempo mínimo del número de corridas que realiza el usuario. También se observa los tiempos de las pruebas que se realizaron en una corrida.

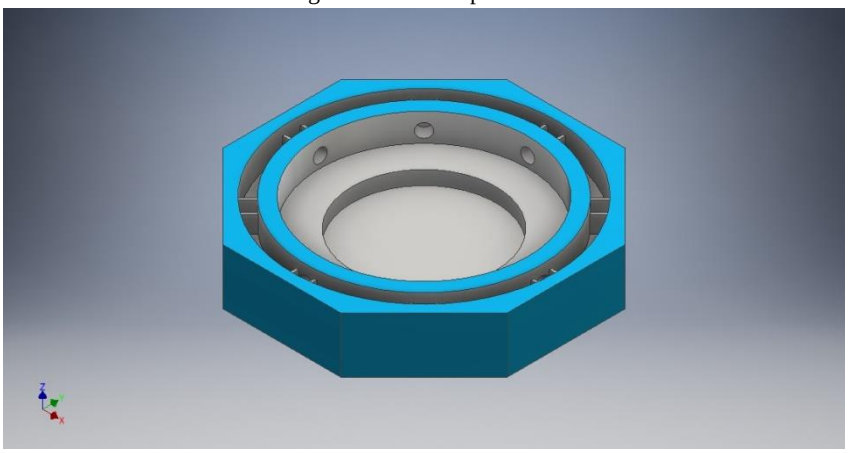
El mismo es capaz de generar una amplia cantidad de bases de datos, así como modificar las mismas, el cual es agregar nuevas hojas de cálculos en el archivo. Como podemos observar en el nombre de la hoja, este tiene la hora de generación la cual es una forma organizada de generar los archivos para tener una base de datos segura, confiable y que permite un mejor control de los usuarios. Así mismo hay que tener en cuenta la base de datos es un ejemplo realizado de cómo podría llegar a ser la base de datos, pero este puede ser modificado como el usuario lo desee.

Figura 123. Base de datos generada



En la siguiente figura observamos el primer prototipo, el cual a diferencia del final este tiene más agujeros para los LEDs y tiene una abertura circular en la cual se iba a colocar la placa que contiene el circuito eléctrico. Otra diferencia es la dimensión la cual tiene un aproximado de 15cm de diámetro.

Figura 124. Prototipo inicial



En las siguientes tres Figuras se observa el diseño final del prototipo, desde la base Figura 125 donde se colocará la placa que contiene al Arduino, el módulo de comunicación, y los LEDs RGB. En la Figura 126 observamos la tapadera la cual contiene el pulsador que al ser presionada determina el tiempo que se tarda en desactivarlo. Y la Figura 127 muestra cómo se vería el prototipo una vez ensamblado.

La base la cual se observa en la Figura 125 contiene el circuito eléctrico que se utilizará, el cual será colocado en el rectángulo que se observa. En los alrededores observamos unos agujeros en los cuales irán los LEDs, que iluminarán el prototipo. El material que se utilizará para crear el prototipo es de plástico (PLA) material utilizado en las impresiones 3D. La siguiente Figura importante es la tapadera la cual tendrá

únicamente el interruptor del prototipo, colocado en la parte superior del mismo en el agujero que se puede observar. Por último, se puede observar el ensamble de las piezas mencionadas, las cuales irán unidas a través de tornillos para el ensamble y desensamble. La dimensión final del prototipo es de 10cm de diámetro aproximadamente. El diseño octagonal es para que se pueda colocar de forma vertical u horizontal.

Figura 125. Diseño de la base del dispositivo

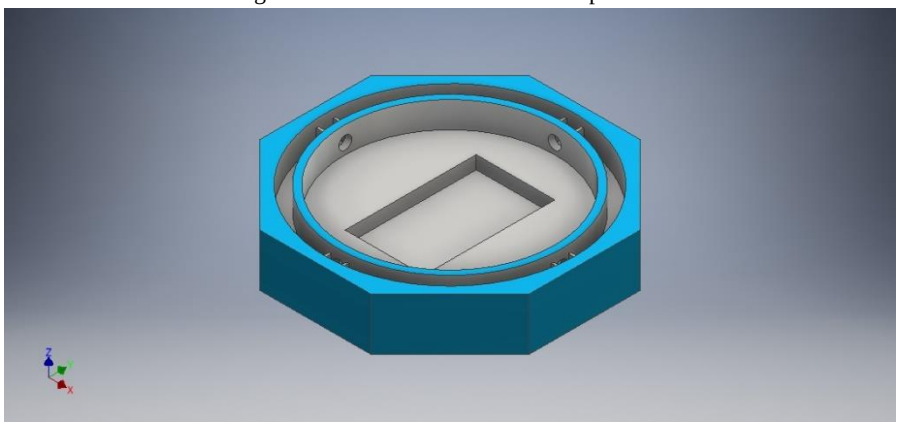


Figura 126. Diseño parte superior o tapadera del dispositivo

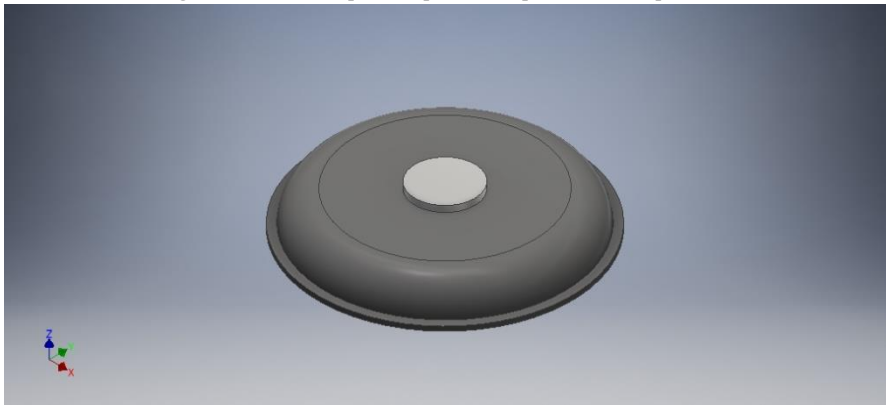
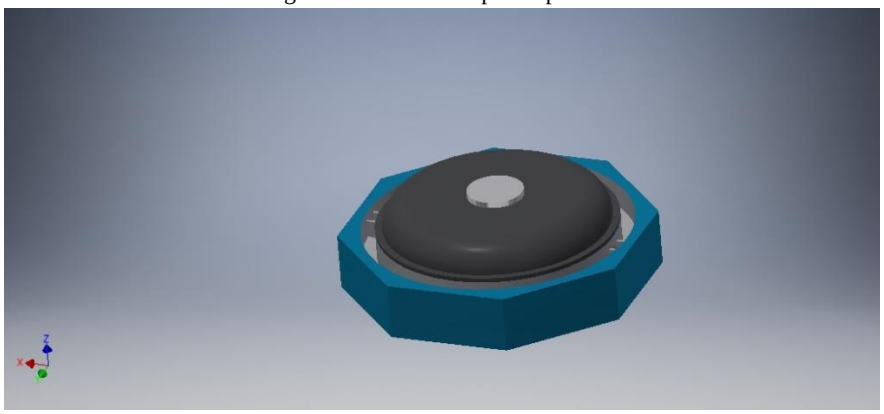
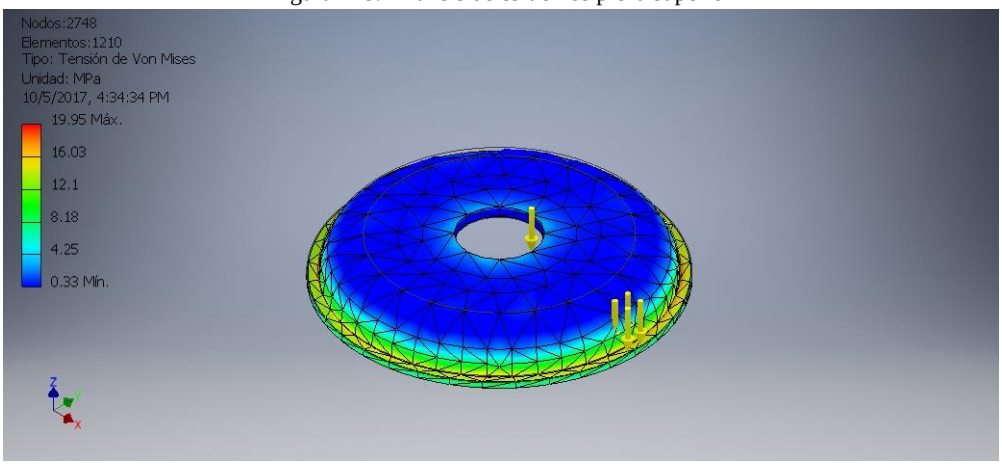


Figura 127. Ensamble prototipo final



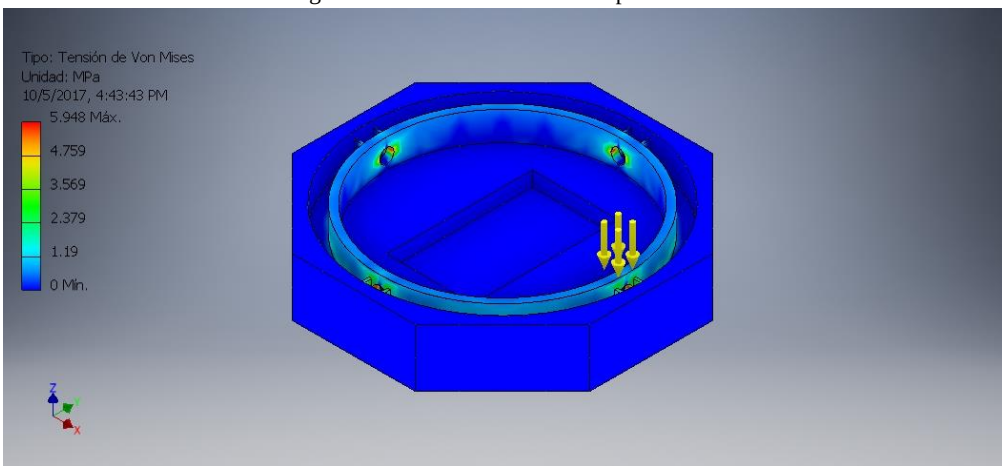
En la siguiente figura observamos un análisis de esfuerzos realizado a la parte superior del diseño, tomando en cuenta distintos escenarios en los que se utilizará el prototipo. Ya que es un prototipo que se utiliza en el campo, es decir durante los entrenamientos, este puede llegar a sufrir ciertas fuerzas debido a presionar el interruptor y en caso de sufrir algún accidente. Este análisis se realizó en la plataforma de Inventor, el cual proporciona un simulador en el cual se puede simular esfuerzos debido a alguna fuerza.

Figura 128. Análisis de esfuerzos pieza superior



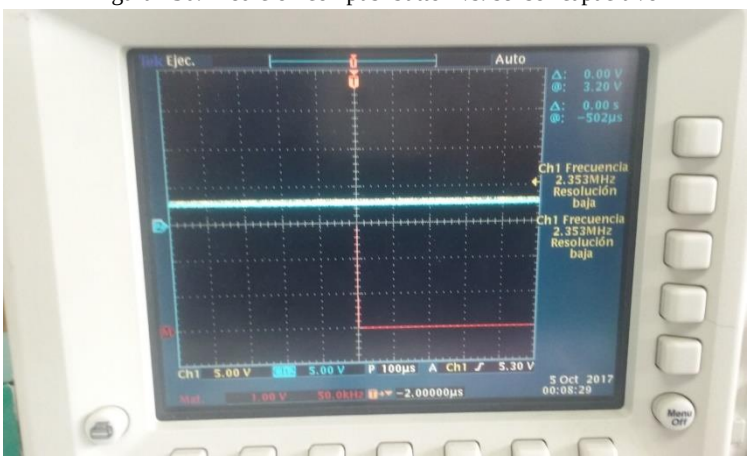
En la Figura 129 podemos observar el análisis de esfuerzos que se realizó a la base del prototipo. Esta únicamente sufre la misma presión generada en la parte de la unió entre la base y la pieza superior. En ella se puede observar que la mayor concentración de esfuerzos se encuentra en la parte superior de las aberturas donde se encuentran los LEDs.

Figura 129. Análisis de esfuerzos pieza base



En la siguiente figura observamos la medición que se realizó para comparar y diferenciar el utilizar un pushbutton y un sensor capacitivo, ambas señales de color amarillo y azul respectivamente se encuentra en el momento en que son presionadas.

Figura 130. Medición con pushbutton vs. sensor capacitivo



C. Sensor tiro con armas de caza

En los siguientes tres cuadros puede observar el costo de los materiales utilizados en cada prototipo, así como un listado de los mismos. En estos valores no se incluyen costos de manufactura y diseño. Los prototipos 4 y 5 poseen el mismo costo ya que la única diferencia radica en la implementación de la interfaz gráfica.

Cuadro 23. Costos totales prototipo 2.

Componente	Cantidad	Costo Unitario (Q)	Costo Total (Q)
Xbee S2 Wire Antenna	3	127.58	382.73
Arduino Nano	1	99.44	99.44
Arduino Uno	1	149.23	149.23
Acelerómetro MMA7361	1	74.58	74.58
Xbee Arduino Uno Shield	1	108.99	108.99
Xbee USB Explorer	2	181.89	363.77
Batería 9V	2	24.86	49.72
Cable USB Mini	1	7.22	7.22
Broche Batería 9V	2	2.99	5.98
Push Button 2 Pines	1	3.28	3.28
Resistencia 1/4 W	2	1.02	2.04
Capacitor 0.1 micros	1	3.50	3.50
PCB Xbee	1	22.13	22.13
PCB Arduino Nano y Sensor	1	25.52	25.52
Costo Total			1,298.10

Cuadro 24. Costos totales prototipo 3.

Componente	Cantidad	Costo Unitario (Q)	Costo Total (Q)
Xbee S2 Wire Antenna	3	127.58	382.73
Arduino Nano	1	99.44	99.44
Arduino Uno	1	149.23	149.23
Acelerometro MMA7361	1	74.58	74.58
Xbee Arduino Uno Shield	1	108.99	108.99
Xbee USB Explorer	2	181.89	363.77
Batería 9V	2	24.86	49.72
Cable USB Mini	1	7.22	7.22
Broche Batería 9V	2	2.99	5.98
Push Button 2 Pines	4	3.28	13.12
Resistencia 1/4 W	5	1.02	5.10
Capacitor 0.1 micros	4	3.50	14.00
PCB Xbee	1	22.16	22.16
PCB Arduino Nano y Sensor	1	25.52	25.52
Costo Total			1,321.53

Cuadro 25. Costos totales prototipos 4 y 5.

Componente	Cantidad	Costo Unitario(Q)	Costo Total(Q)
Xbee S2 Wire Antenna	1	127.58	127.58
Xbee RP-SMA	1	145.44	145.44
Duck Antenna RP-SMA	1	72.54	72.54
Arduino Nano	1	99.44	99.44
Arduino Uno	1	149.23	149.23
Acelerómetro MMA7361	1	74.58	74.58
Xbee Explorer Regulated	2	72.54	145.07
Batería 9V	1	24.86	24.86
Broche Batería 9V	1	2.99	2.99
Resistencia 1/4 W	2	0.87	1.75
LED	1	1.82	1.82
PCB Arduino Nano y Sensor	1	25.52	25.52
Costo Total			870.79

En la Figura 131 se observa el prototipo 2 montado en la escopeta de Jean Pierre Brol Luego de realizar las pruebas. Con respecto a este prototipo Jean Pierre sugirió lo siguiente:

- Buscar la manera de considerar el tiempo de un segundo disparo.

Figura 131. Jean Pierre Brol luego de realizar pruebas con el prototipo 2.

En el siguiente cuadro se pueden observar los alcances máximos obtenidos por los módulos de comunicación. El módulo Bluetooth del Adafruit se encuentra como N/A debido a que no se logró configurar un módulo para la recepción de datos.

Cuadro 26. Alcances módulos de comunicación.

Módulo	Alcance máximo (Metros)
Adafruit Bluetooth Low Energy	N/A
Bluetooth HC-05	7
Xbee S2 Wire Antenna	80
Xbee S2 RP-SMA	80

La siguiente Figura presenta la interfaz gráfica utilizada en el prototipo 3. En el Cuadro 27. se observan los datos agregados a Excel por medio de esta interfaz, en una de las pruebas realizadas por Pedro Zaya.

Figura 132. Interfaz gráfica Processing prototipo 4.

Cuadro 27. Excel generado por Processing.

Tiempo de reacción (m:s:ds:cs)	Tiempo de disparo 1 (m:s:ds:cs)	Tiempo de disparo 2 (m:s:ds:cs)
0:0:1:37	0:0:1:38	
0:0:3:38	0:0:0:0	0:0:0:0
0:1:2:04	0:0:0:0	
0:0:1:45	0:0:0:0	
0:0:0:0	0:0:0:0	0:0:0:0

El atleta Rodrigo Zachrisson realizo las siguientes sugerencias con respecto al diseño y funcionamiento del equipo:

- Colocar un indicador que permita saber cuándo el equipo ya se encuentra encendido.
- Agregar una modalidad a la interfaz gráfica que permita ingresar el tiempo máximo y mínimo de un atleta por estación en la modalidad Skeet y un indicador que le notifique al atleta si el tiempo obtenido en cada estación se encuentra dentro de este rango.
- Disminuir la altura de la placa.

Las siguientes Figuras reflejan la interfaz gráfica del prototipo 5 así como el PCB diseñado en Altium, y los componentes montados en el mismo, respectivamente.

Figura 133. Interfaz gráfica en Java prototipo 5.

Figura 134. PCB Arduino Nano y sensor

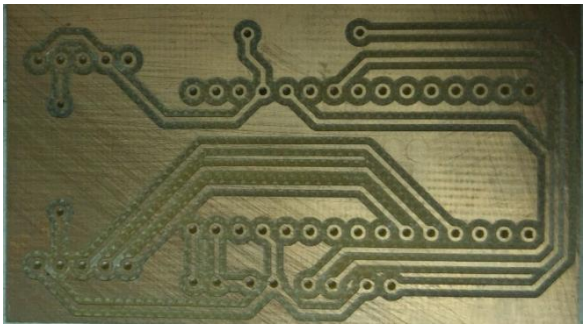


Figura 135. Esquemático prototipo 5.

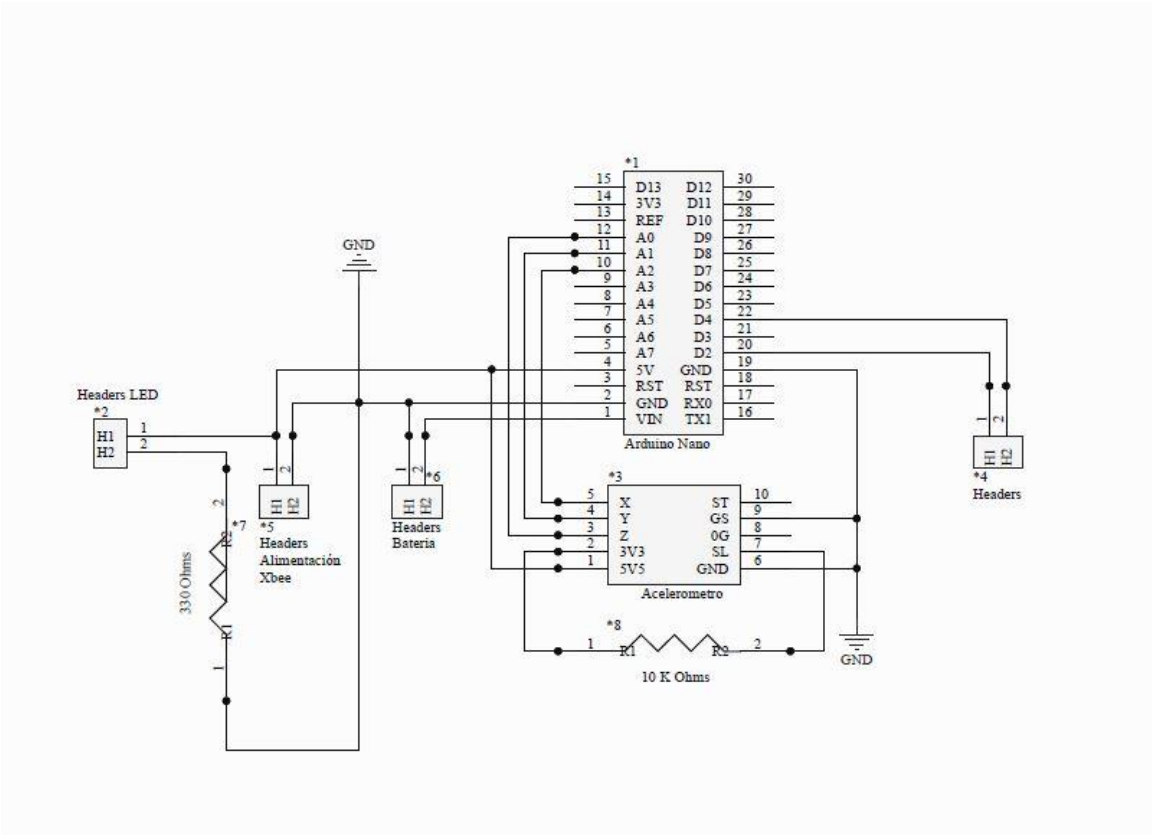
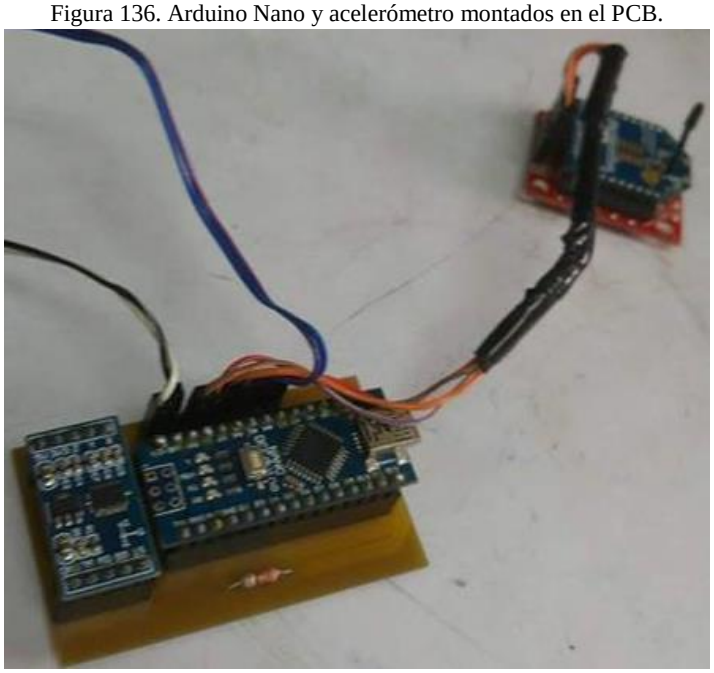


Figura 136. Arduino Nano y acelerómetro montados en el PCB.



En los siguientes cuatro cuadros se pueden observar los datos obtenidos por el dispositivo y la forma en cómo se almacenaron los mismos en el documento de Excel, de las pruebas de Juan Ramón Schaeffer y Dany Brol.

Cuadro 28. Entreno skeet Juan Ramón Schaeffer ronda 1.

Tiempo de reacción (m:s:ds:cs)	Tiempo de disparo 1 (m:s:ds:cs)	Tiempo de disparo 2 (m:s:ds:cs)
0:0:0:30	0:0:0:32	
0:1:6:92	0:2:4:60	0:2:4:60
0:0:4:46	0:0:4:48	
0:0:1:80	0:0:9:68	0:4:8:39
0:0:0:42	0:0:0:46	
0:0:0:30	0:0:0:35	0:0:2:51
0:0:0:03	0:0:4:43	
0:0:0:15	0:0:0:15	
0:0:0:64	0:0:0:65	
0:0:0:27	0:0:1:30	0:0:1:30
0:0:0:49	0:0:1:51	
0:0:0:26	0:0:2:48	0:0:2:48
0:0:1:53	0:0:2:63	
0:0:0:60	0:0:0:60	0:0:0:63
0:0:0:31	0:0:0:45	0:0:0:48
0:0:0:69	0:0:1:46	
0:0:0:31	0:0:0:35	

Cuadro 29. Entreno skeet Juan Ramón Schaeffer ronda 2.

Tiempo de reacción (m:s:ds:cs)	Tiempo de disparo1 (m:s:ds:cs)	Tiempo de disparo2 (m:s:ds:cs)
0:0:0:30	0:0:0:32	
0:1:6:92	0:2:4:60	0:2:4:60
0:0:4:46	0:0:4:48	
0:0:1:80	0:0:9:68	0:4:8:39
0:0:0:42	0:0:0:46	
0:0:0:30	0:0:0:35	0:0:2:51
0:0:0:03	0:0:4:43	
0:0:0:15	0:0:0:15	
0:0:0:64	0:0:0:65	
0:0:0:27	0:0:1:30	0:0:1:30
0:0:0:49	0:0:1:51	
0:0:0:26	0:0:2:48	0:0:2:48
0:0:1:53	0:0:2:63	
0:0:0:60	0:0:0:60	0:0:0:63

Cuadro 30. Entreno fosa Dany Brol 1 disparo.

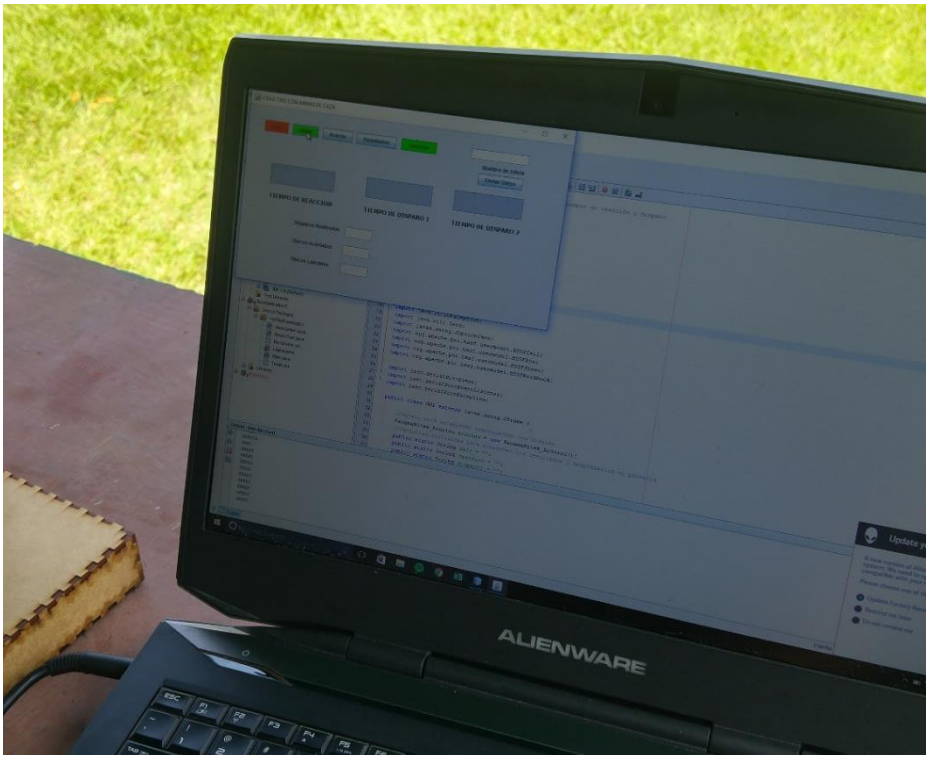
Tiempo de reacción (m:s:ds:cs)	Tiempo de disparo 1 (m:s:ds:cs)
0:0:0:73	0:0:6:00
0:0:0:37	0:0:4:57
0:0:0:57	0:0:2:61
0:0:0:92	0:0:4:14
0:0:0:68	0:0:2:70
0:0:0:24	0:0:4:48
0:0:2:59	0:0:4:63
0:0:0:46	0:0:0:48
0:0:0:63	0:0:8:09
0:0:2:50	0:0:4:63
0:0:0:05	0:0:6:34
0:0:0:18	0:0:0:19
0:0:0:47	0:0:6:70
0:0:0:13	0:0:2:24

Cuadro 31. Entreno fosa Dany Brol 2 disparos.

Tiempo de reacción (m:s:ds:cs)	Tiempo de disparo1 (m:s:ds:cs)	Tiempo de disparo2 (m:s:ds:cs)
0:0:0:69	0:0:2:00	0:0:2:35
0:0:1:37	0:0:4:55	
0:0:0:87	0:0:2:31	0:0:2:47
0:0:1:55	0:0:4:14	0:0:4:22
0:0:1:68	0:0:2:57	
0:0:0:74	0:0:2:84	0:0:3:01

Figura 137. Entreno Juan Ramón Schaeffer con prototipo 5.

Figura 138. Comunicación interfaz gráfica sensor en entreno de Juan Ramón Schaeffer.



VII. DISCUSIÓN

A. PCB sensor inercial

En el presente trabajo se valida el diseño de la placa de circuito impreso de un sensor inercial de bajo costo para aplicación deportiva, ya que se obtuvo el resultado esperado al momento de realizar las pruebas del sensor inercial. Adicionalmente, se considera que el proceso que se llevó a cabo es el adecuado para un trabajo de diseño y desarrollo como el del sub-módulo de Diseño de las placas de circuito impreso para sensores inerciales de bajo costo del módulo de biomecánica del Megaproyecto de Ciencia Aplicada al Deporte de la Universidad del Valle de Guatemala.

Luego de la investigación de los temas más importantes para poder llevar a cabo este módulo de diseño de las placas de circuito impreso, el desarrollo de un prototipo al inicio del proyecto fue una de las fases más cruciales para la elaboración del mismo. Esto se debe a que con el prototipo realizado se pudo aprender la mejor forma de llevar a cabo la implementación de un microcontrolador ATMEGA328P en el proyecto.

Algunos de los conceptos más importantes aprendidos gracias al desarrollo del prototipo fueron, el circuito básico que se debe implementar para que un microcontrolador ATMEGA328P funcione de manera adecuada, la descarga del bootloader con un Arduino para poder utilizar el microcontrolador ATMEGA328P con la plataforma de Arduino y por último, la descarga de programas al microcontrolador ATMEGA328P sin necesidad de tener una conexión directa con la computadora por medio de un puerto USB o similares.

Al momento de realizar los diseños esquemáticos, diseños de footprints, diseño 3D y diseño de PCB, tanto del circuito completo que se implementó como el de los componentes electrónicos utilizados, la implementación del Software Altium Designer fue la mejor decisión que se pudo tomar. Esto debido a que este software de diseño de circuitos eléctricos y electrónicos es uno de los más completos que hay. El poder realizar el diseño del esquemático del circuito y el diseño del PCB en el mismo software es una facilidad que no incluyen otros softwares de diseño de circuitos eléctricos.

Adicionalmente, la capacidad que tiene el software Altium Designer para la creación de componentes que no existen en sus librerías fue una función muy útil al momento de diseñar el circuito. Los diseños y creación de componentes es una tarea que toma tiempo de hacer y se debe ser cuidadoso al hacerla para cumplir con las medidas exactas y configuración de pines adecuada pero una vez creados los componentes se pueden utilizar tanto en el proyecto actual como en proyectos futuros donde se necesite implementar los componentes creados. Respecto al diseño de PCB, la facilidad proporcionada por el software Altium Designer para estos diseños significó un avance más rápido y mejor del diseño final de placa de circuito impreso implementado.

El diseño de la placa de circuito impreso se decidió hacer con varios componentes de superficie, tales como, resistencias, capacitores, LEDs y el mismo microcontrolador ATMEGA328P para poder reducir el tamaño de la placa lo más que se pudiera y cumplir con las medidas propuestas, menor a 5 x 5 cm de ancho y largo, respectivamente. Esta decisión tuvo repercusiones tanto negativas como positivas a lo largo del diseño de la placa de circuito impreso.

Lo positivo que se obtuvo de esta decisión principalmente fue la reducción significativa de la placa de circuito impreso, obteniendo un tamaño de 4.05 x 4.93 cm de ancho y largo, respectivamente. Además, al momento de realizar el diseño de PCB en el software Altium Designer, los footprints de los componentes con empaquetado SMD son más pequeños que sus contrapartes con empaquetados *Thru-hole*, por lo que se facilitó el diseño de la PCB al ordenar los componentes.

Lo negativo de este diseño con una gran cantidad de componentes con empaquetado SMD se dio en el proceso de manufactura. Para comenzar las soldaduras de superficie son más complicadas y difíciles de realizar que una soldadura de un componente *Thru-hole*. Otro aspecto importante es que los componentes SMD son más susceptibles al incremento de la temperatura al momento de realizar la soldadura, por lo que algunos componentes tienden a arruinarse si se calientan por un tiempo corto al momento de soldarse. Esto repercutió directamente en la manufactura de la palca de circuito impreso, ya que efectivamente los LEDs utilizados y un microcontrolador ATMEGA328P-AU se arruinaron al momento de soldarse.

Adicionalmente, los microcontroladores ATMEGA328P-AU, es decir, los microcontroladores ATMEGA328P de empaquetado SMD, tienen una forma distinta de cargar el bootloader al de su contraparte de empaquetado *Thru-hole* de la cual hay una menor cantidad de documentación. Esto perjudicó significativamente el proyecto, atrasando las pruebas finales de la placa, ya que no se logró realizar el bootloader en los primeros intentos.

Por último, esta decisión también dificultó la forma de obtener los componentes para el proyecto, ya que al ser componentes SMD en Guatemala no son utilizados con regularidad por lo que se necesitó mandar a pedir todos los componentes SMD al extranjero, si se hubiera optado por la opción de tener en su mayoría componentes *Thru-hole* se hubiera podido obtenerlos de forma local.

La dificultad de cargar el bootloader al microcontrolador fue que el microcontrolador que se utilizó en la placa fue el ATMEGA328-AU y no el ATMEGA328P-AU, ya que este fue el que se pudo conseguir. El inconveniente es que el IDE de Arduino tiene una diferente configuración para ambos empaquetados, por lo que no reconocía el ATMEGA328-AU en la placa diseñada. Se trató de solucionar realizando la configuración adecuada para que lo reconociera y poder programarlo, pero el Arduino UNO y el Arduino NANO que se utilizaron tienen ATMEGA328P, por lo que el IDE de Arduino no los reconocía al momento

de realizar la descarga del programa con la nueva configuración. En la sección de anexos se muestra una captura de pantalla con el mensaje de error que muestra el IDE de Arduino. Lo que se cambia en la configuración es el “device signature” para que reconozca el ATMEGA328 o ATMEGA328P.

Un aspecto importante de mencionar es el tamaño de las pistas utilizadas para la placa de circuito impreso. Al final, se utilizó un tamaño de 20mils para todas las pistas de la palca de circuito impreso. Esta decisión se tomó debido a la precisión de maquinado de la CNC, ya que si se diseñaban con un ancho menor a 20mils podría generar problemas al maquinarlo. Al tener suficiente espacio en la placa y verificar que las pistas de 20mils en todo el circuito no afectan negativamente, cumpliendo con las normas de diseño requeridas se tomó la decisión de implementar todas de este tamaño.

Una decisión que fue sumamente significativa para el desarrollo del proyecto fue la de realizar una placa de circuito impreso de doble capa. De no haber realizado la placa de circuito impreso de doble capa no hubiera sido posible cumplir con las dimensiones que se plantaron como objetivo, ya que se pudo hacer una distribución de los componentes de una forma más compacta en ambas superficies realizando conexiones a través de vías.

Al momento de realizar la manufactura se optó indiscutiblemente por realizar la manufactura de la placa de circuito con una CNC. Esto se debe a la precisión con la que trabajan estas máquinas obteniendo un acabado fino y muy preciso para la placa de circuito impreso. Otra ventaja importante de esta decisión fue el tiempo de manufactura que llevó hacer la placa de circuito impreso con una CNC comparado con métodos convencionales, ya que el tiempo de manufactura se reduce significativamente.

B. Obtención de datos sensor inercial

La elaboración de las gráficas anteriores fue un proceso trabajoso, pero como resultado se tienen gráficas que representan todo el set de mediciones de cada modalidad. Los resultados de dicho análisis son los siguientes: mientras más lento es el movimiento, más fácil es distinguir sus frecuencias principales, esto se puede ver claramente en la comparación de la Figuras 108, 111 y 114, las cuales corresponden al eje X.

También se puede observar que mientras más lento es el movimiento, menos frecuencias principales se tiene. En el caso de las corridas del modo caminando se puede observar en los tres ejes una frecuencia principal y otras frecuencias secundarias con menos de la mitad de la amplitud de la primera. Mientras que trotando ya se tiene un promedio de tres frecuencias principales que tienen casi la misma amplitud, lo que indica que el movimiento es más complejo, y con frecuencias que no siempre serán las mismas. En el caso de la modalidad corriendo, se puede ver que no hay un patrón definido y no se pueden distinguir frecuencias principales claras en ninguno de los ejes. Esto tiene sentido, ya que al correr cada persona lo hace de una

manera muy distinta, con una zancada distinta y a un ritmo distinto, mientras que la caminata es un movimiento que se realiza casi a la misma frecuencia en la mayoría de personas. Es por esto que la investigación se enfocará en describir la trayectoria de una caminata. Ya que es fácil de identificar el set de frecuencias que se estudiará y tratar de obtener una trayectoria representativa mediante el uso de filtros que permitan el paso únicamente de las frecuencias de interés.

Como se puede observar en el Cuadro 19, las trayectorias, en su mayoría tienen un coeficiente de correlación cercano a 1. Esto quiere decir que tanto las mediciones como el cálculo de la orientación se realizaron de manera correcta. La interpretación de este coeficiente es que las mediciones están muy cercanas a una línea recta. Los filtros se ajustaron para la magnitud de las mediciones, ya que se tiene en la mayoría, una magnitud con un porcentaje de error por encima del 1000%. Teóricamente no hay porcentajes arriba del 100% pero como se puede ver, la trayectoria debía ser de 100 metros y resultó siendo por encima de los 1000 metros en la mayoría de las mediciones.

Para ajustar los filtros que darán un mejor resultado, se verificaron las gráficas de las Figuras 108, 109 y 110. Donde se realiza el análisis frecuencial de las mediciones que se están analizando. En estas gráficas se puede apreciar que las frecuencias con mayor magnitud están a partir de 1 Hz y terminan en 6.5 Hz para casi los tres ejes. Es por ello que se realizó una serie de pruebas en las que se fue variando la frecuencia del filtro pasa bajas, es decir, la menor frecuencia que se permitirá pasar. Se llegó a la conclusión que los primeros armónicos eran únicamente desfases por integración, y estos tenían como resultado una trayectoria con una magnitud muy grande. Es por ello que se quitaron los primeros armónicos y las frecuencias de corte de los filtros que dieron el mejor resultado fueron:

Es importante mencionar que al filtrar las señales se identificó que todas tenían en común que cada paso realizado tenía un factor de amplificación en común. Esto quiere decir que el problema de los desfases no se encontraba solamente en los acelerómetros sino al integrar. Es por ello que se realizó un análisis para determinar el paso promedio en las trayectorias, y el paso real del atleta, que tenía un promedio de 0.63m. Realizando esta comparación se determinó que el factor que compartían las trayectorias era de 20, es decir, que cada paso estaba amplificado por 20 cuando se integraba. Es por ello que todas las trayectorias se atenuaron por este factor al obtener la posición, para obtener una trayectoria con unas medidas que sean verídicas. Los resultados de filtrar las trayectorias y quitarles el factor de amplificación por la integración fueron los siguientes:

Como se puede ver en el Cuadro 21, a pesar de que los filtros cambian la información de la trayectoria, el coeficiente de correlación no se vio afectado, es decir, que las trayectorias conservaron sus características de forma, mas no de tamaño. Ahora se tienen trayectorias de una magnitud dentro de un rango cercano a la medición real. Se tienen porcentajes de error que oscilan entre el 6% hasta casi el 50%. La complejidad del

error en la magnitud es que la misma varía mucho y no de una manera sistemática. Es por esto que no se puede diseñar un solo filtro que sea válido para todas las mediciones que se realicen.

Es importante mencionar adicional a los filtros en los acelerómetros, se filtró la velocidad en el eje z. Esto es porque de antemano se sabía que el desplazamiento final en el eje z debía ser cero. Por lo que debía haber oscilaciones en este eje. Se colocó un filtro pasa banda con frecuencia baja de 4Hz y alta de 9.5Hz. Esto quitó cualquier desfase en el eje z de las trayectorias, como aprecia en la Figura 117.

C. Dispositivo emisor de luz RGB para tiempo de reacción

En la Figura 119, Figura 120, Figura 121 se encuentra los diagramas de flujo del programa que genera la base de datos. Para realizar el código se había iniciado utilizando la plataforma por la facilidad de generar una interfaz, pero debido a que se dificultaba la generación de la base de datos se optó por utilizar Python que, a pesar de no tener la facilidad de generar una interfaz, tiene una facilidad de generar un archivo de Excel, base de datos, y es posible modificarla sin ninguna dificultad. Por otro lado, hay que mencionar que la extensión utilizada da la opción de interactuar con los formatos de las celdas por lo que permite organizar de forma sencilla los datos recibidos. Para este caso las bases de datos se crearán en la misma carpeta donde se encuentra el programa, ya que por ser un ejemplo de un posible programa se diseñó de manera sencilla.

También cabe mencionar que no se creó un ejecutable ni una interfaz vistosa sino únicamente se creó un programa sencillo capaz de generar la base de datos, ya que el objetivo es demostrar que en plataformas gratuitas como lo son Python, Processing y Java es posible crear una base de datos y poder manejar de cualquier manera los datos recibidos a través de Arduino, el cual contiene los tiempos.

En la Figura 123 podemos observar la base de datos que genera el programa. Entre lo más importante a tener en cuenta es que la base de datos es generada en Excel por las herramientas que este tiene disponible para hacer análisis estadísticos y generar gráficos los cuales pueden llegar a ser de gran ayuda para los psicólogos. Por otro lado, cabe mencionar que el nombre de la hoja de cálculo es la hora de creación, esto para tener un mejor control del momento en que se realizó las pruebas, no se decidió colocar la fecha y hora por cuestiones de visibilidad, ya que si colocamos la fecha y hora se vería amontonado e inentendible.

En el Cuadro 22 podemos observar una tabla que enumera una serie de características importantes a tomar en cuenta, la más importante y la que cumple con el objetivo del módulo, es el precio que podría llegar a tener en caso de lanzarlo al mercado. Este valor de $130.67 aproximadamente se basa en material y los componentes utilizados. Los programas no se tomas en cuenta debido a que los utilizados se encuentran gratuitos en la Internet. Por otro lado, no se toma como significativo el costo de comprar los componentes, ya que, a diferencia del material, la batería y el módulo de comunicación son muy bajos. El material utilizado

es PLA un plástico utilizado para las impresiones 3D el cual, si se llega a imprimir en alguna empresa, este tiene un valor aproximado de Q.300, aproximadamente $40.50, incluyendo el material y el tiempo, que tuvo un aproximado de 4 horas. Otro aspecto es la batería que tiene un costo de Q.66.00, un aproximado de $9, ya que es una batería de Litio de 9V con una capacidad de 800mAh y son recargables. Por último, hay que tomar en cuenta el costo de los módulos de comunicación, ya que tienen un aproximado de $67.57. Por lo que el total incluyendo el tiempo utilizado para diseñarlo, y la construcción es de aproximadamente $130.

En la Figura 124 y Figura 125 observamos el prototipo inicial y el prototipo final respectivamente. El primer prototipo propuesto tenía un diámetro de 15cm esto debido a que si se deseaba colocar tanto el módulo de comunicación y la batería en el mismo lugar que la placa, que es el agujero circular, las dimensiones de la base aumentaban 5cm más en comparación al segundo prototipo. Para el segundo prototipo, el cual es el prototipo final, las dimensiones eran de 10cm de diámetro. El inconveniente al reducir el tamaño del prototipo es que el módulo de comunicación y la batería se ubicarían encima de la placa. Por lo mismo que se redujo el tamaño del prototipo, ya no era necesario utilizar una placa circular y los componentes se podían colocar más juntos y reduciríamos el espacio. Las dimensiones finales de la placa son de 29.5x50 mm, que incluyen el Arduino, una bornera de tornillo para la batería y la conexión del módulo de comunicación.

El prototipo inicial por otro lado tenía 8 aberturas para colocar los LEDs RGB, pero debido a que estás tenían un consumo mayor a la capacidad de las baterías, se redujo a 4 LEDs. El consumo de los LEDs RGB es de 40mA por color, es decir si estás estaban funcionando a su máximo desempeño consumirían un aproximado de 980mA incluyendo el consumo general del Arduino. Si utilizamos una batería de Litio de 9V con una capacidad de 800mAh y utilizábamos los 8 LEDs la duración de la batería sería de aproximadamente 50min por lo que se redujo el número de LEDs a 4, lo que haría durar la batería un aproximado de 1hora 15min.

En la Figura 126 podemos observar el diseño de la tapadera está tiene un agujero en la parte superior en la que se colocará el accionamiento. Está diseñado de esa manera porque se desea colocar el domo de una bocina, que es un material conductor pero lo bastante resistente para soportar la presión generada por la mano, esta se puede observar como la parte de aluminio mostrada en dicha Figura.

Por último, en la Figura 130 podemos observar una medición en el osciloscopio de la diferencia entre usar un pushbutton y el sensor capacitivo. Como podemos observar en la pantalla de este mismo no hay alguna diferencia en cuanto a funcionalidad, pero si existe diferencia alguna. Dicha diferencia se observa en el diseño y el costo de cada uno. Para el diseño observamos que utilizar un sensor capacitivo es necesario tener únicamente un material conductor sin importar las dimensiones, es decir se puede utilizar un material con un área grande y su funcionamiento no varía. A diferencia del pushbutton que las dimensiones son menores y requiere un costo el poder adquirirlo.

Por otro lado, se intercambió el pushbutton por el sensor capacitivo para realizar un prototipo atractivo para el usuario, pero también para cumplir parte de los objetivos que es generar un prototipo sencillo, con componentes de bajo costo. En la Figura 128. y Figura 129. podemos observar el análisis realizado al prototipo esto para prevenir cualquier tipo de percance durante el entrenamiento y se toma en cuenta la fuerza aplicada al momento de accionar el interruptor del prototipo. Para el análisis de esfuerzos se tomaron en cuenta dos casos, el primero es la fuerza que se aplica sobre la parte superior del prototipo, se realizó un aproximado de la fuerza máxima que aplicaría la mano sobre el prototipo. Para esto se utilizó una pesa eléctrica en la cual se presionó a tal punto de generar una fuerza aproximada aplicada en el punto del interruptor, este tiene un valor de masa de 130.5g y utilizando la fórmula de:

$$F = m * a. \quad (13)$$

Se determinó la fuerza máxima y con esto notamos que dicha fuerza no genera alguna deformación significativa. Por otro lado, se aplicó presión entre la unión de la base y la parte superior esto tomando en cuenta algún percance durante los entrenamientos en los que se pise el prototipo. La fórmula aplicada para este análisis es:

$$P = \frac{F}{A}. \quad (14)$$

Por lo tanto, la fuerza aplicada en el área superficial de la unión entre la base y la pieza superior genera una presión aproximada de 2MPa, que si notamos en la Figura 128 el área con mayor concentración de esfuerzos es en la parte inferior de la unión lo cual al mismo tiempo genera esfuerzos mayores en las aberturas donde se colocarán los LEDs. Por último, observamos que la pieza tiene una estructura rígida que podría soportar fuerzas más grandes que las que deberían de ser aplicadas.

D. Sensor tiro con armas de caza

1. Primer prototipo. El prototipo uno contaba con un pulsador simulando ser la señal de la máquina para activar el cronometro. Sin embargo, el funcionamiento del mismo no fue el correcto ya que, cuando el pulsador se presionaba el cronometro varias veces en un tiempo infinitesimal. Esto ocurrió debido a que cuando se activa un pulsador, dos partes de metal se unen entre sí. Sin embargo, esta no se realiza de manera inmediata, al pulsador le lleva un intervalo de tiempo infinitesimal cerrarse por completo (Christoffersen, 2015), a esto se le conoce como rebote. Con el fin de corregir este problema al prototipo dos se le agregó un filtro RC, permitiendo que el cronometro se activará solo una vez. A pesar de esta falla el prototipo se consideró como satisfactorio ya que fue capaz de medir el tiempo de reacción de un individuo.

2. Segundo prototipo. Este prototipo cumplió el objetivo de obtener el tiempo de reacción y de un disparo para la modalidad Fosa. Adicionalmente también cumplió con el parámetro de que la ubicación del

mismo no debía afectar al atleta, ya que si se observa la Figura 131. notará que el dispositivo se encuentra colocado en la cara inferior de la escopeta de Jean Pierre Brol, manifestando el mismo que no le afecto durante el entrenamiento.

Uno de los objetivos que no satisface este prototipo es el costo del mismo ya que sobrepasa los Q1000.00 siendo este de Q1298.10, como se observa en el Cuadro 23. equivalente a USD 176.06 a una tasa de cambio de 7.29. El alto costo de este prototipo se debió a la cantidad de módulos Xbee utilizados, así como los shields utilizados para conectar los Arduinos a los módulos. Una forma de disminuir los costos de este prototipo podría ser utilizar solo dos módulos Xbee en configuración punto a punto como se detalla en el capítulo de metodología en la sección cuarto prototipo, o bien eliminar el shield utilizado para conectar el módulo al Arduino Uno y diseñar un PCB que permita esto, también se podría rediseñar el PCB del Xbee que va conectado al Arduino Nano y así eliminar un Xbee USB Explorer. Por otro lado, debido a la recomendación realizada por el atleta Jean Pierre Brol, se decidió agregar una nueva variable para llevar control de los movimientos realizados por los atletas.

3. Tercer prototipo. Este prototipo cumplió con el objetivo de obtener tiempo de reacción y de disparo para las modalidades Fosa y Skeet. Por lo que la implementación de los caracteres identificadores presentados en el capítulo de metodología como satisfactoria. Sin embargo, al igual que el prototipo anterior no logra satisfacer el objetivo relacionado con el costo ya que este es el obtuvo el costo más alto el cual fue de Q1321.53, como se observa en el Cuadro 24. equivalente a USD 181.28 a una tasa de 7.29. El incremento en el costo de este prototipo se debió a la implementación de los botones que permitieron el cambio de modalidades, el control de aciertos y solicitud de resultados. Este costo se podría disminuir aplicando la alternativa presentada en la sección anterior de este capítulo. Otra alternativa para disminuir este valor podría ser la implementación del debounce presentado en la sección cuarto prototipo del capítulo de metodología.

4. Cuarto prototipo. Como se menciona anteriormente el utilizar el debounce en código elimina el uso de filtros RC provocando así una disminución en los costos. Sin embargo, el decremento más representativo en los costos de este prototipo se ve reflejado en la sustitución de tres módulos Xbee en topología circular a dos en topología punto a punto. En el

Cuadro 25. se observa que el costo de este prototipo satisface el objetivo trazado de los Q 1,000.00.

Una falencia que se presentó en este dispositivo fue la generación de datos ya que en el Cuadro 27. se observa un desfase en las columnas en las que deberían de estar colocados los resultados. Esto ocurrió debido que el objeto PrintWriter no cuenta con método que permita establecer en qué posición del documento se desean agregar los datos. Adicionalmente este objeto genera un archivo nuevo cada vez que se generan los datos, por lo que llevar el control de los atletas para los entrenadores se torna molesto ya que deben buscar

en varios archivos. Esto se debe a que el objeto PrintWriter genera un archivo .CSV el cual no se puede modificar una vez realizado.

Con respecto a las observaciones proporcionadas por el atleta Rodrigo Zachrisson, en el nuevo diseño de la placa se agregó una sección que permita colocar un LED, el cual se conectó al selector On/Off permitiendo que el LED se encienda cuando el dispositivo se encienda.

La implementación de la segunda observación realizad por Rodrigo, se puede lograr si se agregan dos cuadros de texto a la interfaz gráfica que se habiliten cuando se selecciona la modalidad Skeet, estos valores se deben enviar al Arduino Nano y así comparar si el resultado obtenido cumple. Para que el atleta sea alertado del resultado se puede agregar un LED RGB, que se torne Rojo cuando el atleta falle y Verde cuando cumpla.

Finalmente, para disminuir la altura de la placa se podrían soldar los componentes directamente en la placa, sin embargo, esto no permitiría realizar variaciones en el código o sustituir los componentes en dado caso se lleguen a dañar. Otra alternativa sería sustituir el Arduino Nano por el microcontrolador ATMega328P con bootloader, eso permite utilizar el código ya realizado en el IDE de Arduino, y el acelerómetro por uno que sea SMD. Esto no solo disminuiría el tamaño de la placa significativamente, sino que también disminuiría el costo total del dispositivo.

5. Quinto prototipo. La implementación de la interfaz en Java se considera exitosa ya que si se observa el Cuadro 28. notara que los datos fueron agregados correctamente dentro del archivo Excel a diferencia del prototipo anterior. Adicionalmente la interfaz de Java permite que se genere un archivo por atleta y que en el mismo se agreguen pestañas con los nuevos datos, en la sección de Anexos puede encontrar el archivo Excel de Juan Ramón presentado en el Cuadro 28. y Cuadro 29. de la sección de resultados. A pesar de que la interfaz se considera exitosa existen dos desventajas primordiales de utilizarla este método, la primera es que el usuario cierre la interfaz sin haber guardado los datos, provocando una pérdida de los mismos. Esto se puede corregir agregando una opción que le pregunte al usuario si almaceno los datos antes de salir del programa, si lo hizo cerrar el mismo y si no realizó esta acción dejar el programa abierto para que el usuario los almacene. La segunda desventaja es que el usuario tenga el libro de Excel abierto, esto no permite la edición del mismo y lleva al programa a un error provocando la pérdida de información.

El desarrollo del algoritmo para la modalidad de Skeet es válido ya que cumple con los objetivos de capturar tiempos de reacción y disparo para cada estación. Esto se puede observar en el Cuadro 28. en donde se observa que los resultados obtenidos de una sesión de entrenamiento con el atleta Juan Ramón Schaeffer, en el cual se evidencia claramente que se sigue la lógica presentada en el Cuadro 1. presente en el capítulo marco teórico. Si se observa el Cuadro 29. correspondiente a la segunda ronda del mismo entrenamiento

observará que existen datos faltantes, esto debido a que durante dicha ronda la máquina lanzo repetidas veces discos fracturados, obligando al atleta a repetir el lanzamiento en determinadas posiciones. Esto afecta la variable que lleva el control de la estación en la que encuentra el atleta, ya que esta incrementa cada vez que la máquina realiza un lanzamiento. Una forma de corregir esto, sería agregar un botón en la interfaz gráfica que al ser presionado le indique al Arduino Nano que el disparo fue exitoso y con esto incrementar la variable de control de estación, o bien agregar un botón que si se presiona reste una unidad a la variable de las posiciones, llevando al Arduino Nano a ejecutar las instrucciones del lanzamiento que se debe repetir.

Al realizar las pruebas a un disparo en modalidad Fosa el dispositivo respondió de buena manera en la captura de los datos de 15 lanzamientos de 25, esto se observe en el Cuadro 30. La falta de los otros 10 lanzamientos se debe a que el dispositivo no se activó en los mismos, esto debido a que, como se menciona en la metodología todos los prototipos contaron con un botón el cual simulaba el lanzamiento de los discos. Al no ser este presionado correctamente el cronometro no se activó ni se tomaron datos. Sin embargo, este error no debe estar presente cuando se conecté el Arduino Uno con el controlador de las máquinas lanzadoras de discos, con esta consideración se toma la implementación del algoritmo en modalidad de Fosa un disparo como exitosa.

Por otro lado, al realizar pruebas a dos disparos en modalidad Fosa. Cuadro 31. Se observa que existe falta de datos, ya que el dispositivo no siempre capturó la ejecución de los mismos y solo permitió capturar 7 datos debido a que la batería se quedó sin carga. La descarga de la batería no se considera como un falló dentro del diseño ya que se estableció que la misma no debía ser de tipo recargable para no incrementar los costos del prototipo y que debía ser una batería que se pudiese conseguir fácilmente en el mercado.

La falta de captura de los datos del segundo disparo se debe a que el Arduino Nano cuenta con una condicional que verifica si el segundo disparo supera en más de un segundo al primer disparó, si lo hace omite el dato y si no lo hace lo envía para su almacenamiento. Sin embargo, esta condicional no puede ser omitida del programa ya que sin ella el programa captura cualquier movimiento que se realice después del primer disparo y lo tome como segundo disparó.

6. Módulo de comunicación. El alcance del módulo de comunicación seleccionado, Xbee S2, cumplió con las expectativas esperadas ya que logró cubrir en su totalidad la longitud completa de una cancha de Skeet, 39 metros. Esto se puede observar en el Cuadro 26. en la Figura 137. y Figura 138. En donde se observa el sensor conectado en la escopeta de Juan Ramón y la interfaz Gráfica con los botones Skeet y Conectar de color verde, lo cual indica que si existe comunicación entre ambos módulos. En el Cuadro 26. se observa que ambos módulos Xbee lograron dicho alcance, por lo que, si se hubiesen utilizado dos módulos Wire Antenna, el costo final del prototipo habría disminuido en Q90.00. El haber utilizad módulos HC-05

habría disminuido el costo final del prototipo en Q270.00, ya que estos módulos tienen un valor de Q75.00 en el mercado local. Sin embargo, el dispositivo no hubiese sido capaz de enviar datos de todas las estaciones.

VIII. CONCLUSIONES

1. Se diseñó una placa de 4.06 x 4.91 cm, las cuales están acorde con las medidas propuestas. Comparado con el Xbee Add-on para Arduino NANO que tiene dimensiones de 9.4 x 2.8 cm, lo cual es una reducción significativa en el largo de la placa.

2. Se determinó que la corriente máxima del circuito es de 60mA.

3. Se calculó que las pistas del circuito deben ser mayores a 4.55mils / 0.1156 mm, calculando el caso donde el regulador de voltaje opera con su corriente máxima de salida (0.5A).

4. Se obtuvo una altura de 33.42mm debido a la implementación de pin headers hembra en ambas superficies de la placa para la IMU BNO055 y el Xbee Explorer Regulator. De haberse soldado directamente a la placa la altura se reduciría a 19.72mm.

5. Se encontró que la mejor opción para cargar el bootloader es el proceso de ArduinoISP, debido a su facilidad de pasos, documentación y las herramientas se encuentran en el IDE de Arduino.

6. Es posible obtener la trayectoria rectilínea de una persona realizando un movimiento de caminata utilizando sensores inerciales.

7. Las trayectorias obtenidas tienen un porcentaje de error debajo del 50%, lo cual muestra que las mediciones son veraces, mas no confiables para una experimentación que requiera precisión.

8. Los filtros digitales son de gran uso para el procesamiento de señales. Esto es de gran utilidad cuando se quiere permitir el paso de frecuencias específicas de una señal.

9. Mediante la utilización de Xbees es posible lograr una comunicación inalámbrica constante con el objetivo de obtener mediciones en tiempo real.

10. El error al obtener una trayectoria utilizando sensores inerciales tiene características sistemáticas y no sistemáticas.

11. Es posible mitigar las características sistemáticas del error en la obtención de la trayectoria, como lo es, el desfase en el eje z por medio de la utilización de filtros digitales.

12. Es posible mitigar gran parte del error no sistemático de una trayectoria mediante la identificación de las frecuencias que contienen información e implementando filtros digitales que permitan el paso exclusivo de dichas frecuencias.

13. Mediante la fusión de las mediciones de una IMU se puede obtener una correcta orientación del sistema.

14. Es posible, utilizando la representación de la orientación mediante cuaterniones, evitar pérdidas de orientación.

15. Se logró implementar un programa capaz de capturar el tiempo que se tarda un atleta en reaccionar ante un estímulo, creando una interfaz sencilla. El programa es capaz de capturar un tiempo mínimo de 0.001 segundo, lo cual es muy difícil de alcanzar para un atleta, pero es un programa eficiente para la captura de datos.

16. Se implementó la comunicación inalámbrica para facilitar el manejo del prototipo ubicándolo a 60 metros de distancia entre el dispositivo y una computadora. El área de los entrenamientos para los cuales se realizó el prototipo es de 8x8m es evidente que el dispositivo cumple con dichas dimensiones.

17. Se logró crear la base de datos sencilla utilizando Excel para fácil manejo de datos y para análisis de los mismos siendo una base de datos segura.

18. Se implementó el sensor táctil capacitivo para la captura del tiempo por la facilidad que tienen el diseñarlo. El botón no presenta ningún rebote por lo que un botón se puede implementar de igual manera, pero el área de contacto es menor.

19. La captura correcta de tiempo se cumplió, porque la precisión de la captura es 0.001 segundos lo cual es de gran ayuda a la hora de realizar las pruebas.

20. Se logró implementar en el programa las funciones de tiempo máximo, tiempo mínimo y tiempo promedio con el fin de ayudar al atleta a determinar factores que llegan a afectarle.

21. Se diseñó un dispositivo portátil y con las dimensiones adecuadas, 10cm de diámetro, para poder utilizarse en el campo cumpliendo con las especificaciones requerida. Por las dimensiones del prototipo el transporte de este es fácil, ya que cabe en la palma de la mano muy fácilmente.

22. Se logró diseñar un dispositivo con un costo total de Q870.79 (USD 119.45)

23. El módulo de comunicación seleccionado posee un alcance máximo de 80 metros, superando el necesario para transmitir información en una cancha de tiro con armas de caza.

24. Se logró diseñar un equipo que cumple con los parámetros de diseño establecidos por entrenadores y atletas de tiro.

25. El algoritmo desarrollado es capaz de proporcionar correctamente tiempo de reacción y tiempo del primer disparo para la modalidad Fosa.

26. El algoritmo desarrollado proporciona el tiempo del segundo disparo en la modalidad Fosa, siempre que este no sea mayor en un segundo al tiempo del primer disparo.

27. El algoritmo desarrollado es capaz de proporcionar correctamente tiempo de reacción, tiempo de disparo 1 y tiempo de disparo 2 para la modalidad Skeet.

28. El algoritmo desarrollado permite que la información sea desplegada en una interfaz gráfica y almacenada en un documento de Excel.

IX. RECOMENDACIONES

1. El principal inconveniente que se tuvo fue el de realizar el bootloader al microcontrolador ATMEGA328P-AU, por lo que se recomienda que al momento de comprar dicho microcontrolador se adquiera uno que tenga la capacidad de ser utilizado con la plataforma de Arduino para evitar este paso. El tener un microcontrolador ATMEGA328P-AU con el bootloader precargado de fábrica simplifica el diseño de la placa de circuito impreso, ya que reduce la cantidad de pistas y pines que se deben utilizar la misma.

2. Se recomienda evaluar la implementación de componentes SMD, ya que se considera que esto afecto más de forma negativa que positiva por la poca experiencia que se tenía trabajando con este tipo de componentes y se considera que pudo obtenerse al menos el mismo tamaño de 4.05 x 4.94 cm de ancho y largo con componentes Thru-hole. Adicionalmente, el obtener componentes SMD es más complicado que los componentes Thru-hole por su poco uso en Guatemala.

3. Al utilizar componentes SMD, se recomienda practicar y aprender la mejor forma de realizar este tipo de soldadura para evitar inconvenientes en fases siguientes a la soldadura.

4. Si se desea realizar alguna modificación significativa al diseño, se recomienda mantener el proceso de investigación y desarrollo que se llevó a cabo en este proyecto. Es decir, comenzar con investigación y proseguir con realizar un prototipo para tener claro que se va a implementar.

5. Se recomienda rediseñar el circuito realizando la conexión del Xbee directamente a la placa y al microcontrolador ATMEGA328P, evitando el uso del Xbee Explorer Regulator como intermediario entre ambos. Esto para reducir el alto del diseño de la placa.

6. Se recomienda rediseñar el circuito e implementar un módulo para cargar baterías para hacer más efectivo el sensor.

7. Se recomienda mantener el diseño de una placa de circuito impreso de doble capa o rediseñar la placa para que sea de múltiple capa, con capas internas.

8. Se recomienda mantener el uso del Software Altium Designer al momento de realizar modificaciones o nuevos diseños.

9. Se recomienda fuertemente mantener la manufactura de las placas de circuito impreso por medio de una CNC, ya que esto facilita enormemente este proceso.

10. Se recomienda la utilización de sensores con mayor exactitud para obtener mejores resultados en la obtención de la trayectoria.

11. Se recomienda la fusión de las mediciones de la IMU con otros sensores, como los GPS para obtener una trayectoria corregida, y, por consiguiente, con menos error respecto al marco global.

12. Se recomienda utilizar librerías para graficar la trayectoria en tiempo real que tengan menos retraso de la señal, para ver de una manera continua la trayectoria. Esta tarea se realiza de una manera muy lenta con la librería de Python Matplotlib.

13. Se recomienda investigar otras formas de alimentar al Arduino para reducir espacio y poder tener un dispositivo que internamente este más ordenado.

14. Se recomienda investigar formas de diseñar un Arduino, utilizando únicamente las partes más importantes y las cuales se van a utilizar para reducir el tamaño de la placa significativamente y tener un dispositivo diseñado de mejor manera.

15. Buscar otras alternativas para la captura de datos, ya sea realizando un distinto prototipo o modificando el ya existente.

16. Para versiones posteriores se recomienda realizar una encuesta a todos los atletas de tiro con armas de Caza en las modalidades Fosa y Skeet sobre funciones adicionales que debe llevar el equipo.

17. Se recomienda realizar pruebas con el microcontrolador ATMega328P con bootloader.

18. Se recomienda en versiones posteriores agregar a la interfaz gráfica un cuadro que permita establecer la diferencia de tiempo máxima entre el disparo 1 y disparo 2 de un atleta.

19. Se recomienda agregarle a la interfaz gráfica un indicador de la estación en la que se encuentra el atleta, así como una solución que permita repetir el lanzamiento y borrar el dato erróneo en caso se haya lanzado un disco fracturado.

20. Se recomienda en versiones posteriores agregar a la interfaz gráfica un cuadro que permita establecer el valor máximo y mínimo de tiempo de disparo y de tiempo de reacción por atleta, así como un LED RGB que le indique al atleta si el tiempo obtenido se encuentra dentro del rango establecido.

21. Se recomienda utilizar componentes SMD para disminuir las dimensiones del PCB, así como los costos del equipo

22. Se recomienda realizar pruebas con módulos de comunicación más económicos que tengan un alcance máximo de 45 metros, con el fin de disminuir el costo del equipo.

23. Se recomienda realizar mediciones de las aceleraciones en distintos puntos de referencia de la escopeta, con la finalidad de determinar el punto clave para la diferenciación entre el primer y segundo disparo.

24. Se recomienda realizar mediciones sobre las aceleraciones de los atletas en las distintas posiciones que se ven involucradas en los disparos, que permita encontrar patrones sobre las mismas. Esto con el objetivo de establecer rangos de aceleraciones que afinen la detección de los movimientos de reacción, primer y segundo disparo.

25. Se recomienda utilizar una batería de 9V tipo recargable o bien dos baterías tipo LI-PO de 3.3V en conectadas en serie. Si se opta por utilizar las baterías LI-PO se debe adquirir el dispositivo encargado de recargarlas.

26. Se recomienda realizar modificaciones a la interfaz gráfica que permita la parametrización de los valores de las aceleraciones de acuerdo a los patrones encontrados en cada atleta.

X. BIBILOGRAFÍA

1. ABC Color. *La Respiración Humana*. http://www.abc.com.py/articulos/la-respiracion-humana-784432.html [25/9/17]
2. Adafruit. 2017. *Adafruit BNO055 Absolute Orientation Sensor* https://cdn-learn.adafruit.com/downloads/pdf/adafruit-bno055-absolute-orientation-sensor.pdf [06/06/2017]
3. Adafruit. *Adafruit Feather 32u4 Bluefruit LE* https://learn.adafruit.com/adafruit-feather-32u4-bluefruit-le/overview [23/01/2017]
4. Adafruit. *BNO055* https://cdn-shop.adafruit.com/datasheets/BST_BNO055_DS000_12.pdf [03/03/17]
5. Adafruit. *Installing BLE Library*. https://learn.adafruit.com/adafruit-feather-32u4-bluefruit-le/installing-ble-library [23/01/2017]
6. Addicore. Teensy 3.2 *Dev Board*. https://www.addicore.com/Teensy-3-2-p/ad288.htm [28.7.17]
7. Alam, Zahirual, *et al*. 2010. «Design of Capacitance to Voltage Converter for Capacitive Sensor Transducer». *American Journal of Applied Sciences*. VII (10): 1353-1357.
8. Altium – Import 3D model into footprint – PCB 3D. (s. f.). Recuperado 6 de julio de 2017, a partir de https://www.pcb-3d.com/knowledge-base/altium-import-3d-model-into-footprint/
9. Altium Designer Evaluation Guide: PCB Preparation and Design Transfer. (s. f.). Recuperado 6 de julio de 2017, a partir de http://go.altium.com/AD-evaluation-guide-4-pcb-preparation-and-design-transfer.html
10. American Psychological Association. *Sport Psychology*. http://www.apa.org/ed/graduate/specialize/sports.aspx [18/08/17]
11. ANSI PCB Track Width Calculator. (s. f.). Recuperado 17 de septiembre de 2017, a partir de http://www.desmith.net/NMdS/Electronics/TraceWidth.html
12. Arduino - ArduinoToBreadboard. (s. f.). Recuperado 20 de agosto de 2017, a partir de https://www.arduino.cc/en/Tutorial/ArduinoToBreadboard
13. Arduino - Bootloader. (s. f.). Recuperado 25 de julio de 2017, a partir de https://www.arduino.cc/en/Hacking/Bootloader
14. Arduino - MiniBootloader. (s. f.). Recuperado 25 de julio de 2017, a partir de https://www.arduino.cc/en/Hacking/MiniBootloader
15. Arduino. *Arduino Nano*. https://store.arduino.cc/arduino-nano [18/07/17]
16. Arduino. *Arduino UNO Rev3* from https://store.arduino.cc/arduino-uno-rev3 [25/08/17]
17. Arduino. *Arduino Uno*. http://www.arduino.org/products/boards/arduino-uno [17/07/2017]
18. Arduino. *Capacitive Sensing Library*. http://playground.arduino.cc/Main/CapacitiveSensor?from=Main.CapSense [25.08.17]
19. Arduino. *Getting Started with the Arduino Nano*. https://www.arduino.cc/en/Guide/ArduinoNano [17/07/2017]

20. Arduino. *SoftwareSerial Library* https://www.arduino.cc/en/Reference/SoftwareSerial [17/07/2017]
21. Arenas, Marta. 2008. «Sistema para la adquisición y monitorización de aceleraciones mediante microprocesador.». Tesis Universidad de Sevilla. 187 págs.
22. Atmega328 Pinout. (s. f.). Recuperado 20 de julio de 2017, a partir de http://www.learningaboutelectronics.com/Articles/Atmega328-pinout.php
23. Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf. (s. f.). Recuperado a partir de http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf
24. Bell, Douglas. 2003. *Java Para Estudiantes.* 3ª ed. México: Pearson. 664 págs.
25. BST_BNO055_DS000_12.pdf. (s. f.). Recuperado a partir de https://cdn-shop.adafruit.com/datasheets/BST_BNO055_DS000_12.pdf
26. Burn Arduino Bootloader on Atmega-328 TQFP and DIP Chips on Breadboard. (s. f.). Recuperado 23 de julio de 2017, a partir de http://www.instructables.com/id/Rapid-programming-of-Atmega-328-TQFP-and-DIP-chips/
27. Burn Bootloader ATmega328p AU on SMD chip. (s. f.). Recuperado 3 de agosto de 2017, a partir de http://www.electronoobs.com/eng_arduino_tut6.php
28. Burning the Bootloader on ATMega328 Using Arduino UNO as ISP. (s. f.). Recuperado 3 de agosto de 2017, a partir de http://www.instructables.com/id/Burning-the-Bootloader-on-ATMega328-using-Arduino-/
29. Cálculo de ancho de pista en una PCB. (2010, octubre 19). Recuperado 13 de septiembre de 2017, a partir de https://cuningan.wordpress.com/2010/10/19/calculo-de-ancho-de-pista-en-una-pcb/
30. Camargo, José Luis. 2009 «Modelo de cobertura para redes inalámbricas de interiores.». Tesis Universidad de Sevilla 182 págs.
31. Christoffersen, Jens. *Switch Bounce and How to Deal with it.* https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it/ [19/08/2017]
32. Clínica Universidad de Navarra. *Electroencefalograma.* http://www.cun.es/enfermedades-tratamientos/pruebas-diagnosticas/electroencefalograma [18/09/17]
33. Clínica Universidad de Navarra. *Electromiografía.* http://www.cun.es/enfermedades-tratamientos/pruebas-diagnosticas/electromiografia-electroneurografia [25/09/17]
34. CogniFit. *Tiempo de Reacción.* https://www.cognifit.com/es/habilidad-cognitiva/tiempo-de-respuesta [18/08/17]
35. Cohen, P. (s. f.). Electrosoft Ingenieria Limitada. Recuperado 14 de agosto de 2017, a partir de http://www.pcb.electrosoft.cl/04-articulos-circuitos-impresos-desarrollo-sistemas/01-conceptos-circuitos-impresos/conceptos-circuitos-impresos-pcb.html

36. Comité Olímpico Guatemalteco. *Tiradores de skeet cierran la participación de Guatemala en la Copa del Mundo de Escopeta.* http://www.cogant.cog.org.gt/noticias/2015/3/tiradores-de-skeet-cierran-la-participaci%C3%B3n-de-guatemala-en-la-copa-del-mundo-de-escopeta.aspx [04/09/2017]
37. Comité Olímpico Mexicano. *Psicología del Deporte.* http://www.com.org.mx/psicologia-del-deporte/ [18.08.17]
38. Customer cases. (s. f.). Recuperado 18 de junio de 2017, a partir de https://www.xsens.com/customer-cases/
39. Demay, V. (2014, marzo 6). Running Atmega328 in a standalone mode without Arduino Shield. Recuperado 20 de julio de 2017, a partir de http://www.homautomation.org/2014/03/06/running-atmega328-in-a-standalone-mode-without-arduino-shield/
40. DFRobot. *10 DOF Mems IMU Sensor*. https://www.dfrobot.com/product-818.html [18/06/17]
41. Díaz, Emmanuel. *Las ciencias aplicadas al deporte y sus beneficios* http://planoinformativo.com/planodeportivo/nota_deportes/id/12894/noticia/las-ciencias-aplicadas-al-deporte-y-sus-beneficios.html [05/08/2017]
42. Digi Internarional. *XBP24.* https://www.digikey.com/product-detail/en/digi-international/XBP24-AWI-001/XBP24-AWI-001-ND/935968 [12.07.17]
43. Digi. *XCTU Next Generation Configuration Platform for Xbee/RF Solutions.* https://www.digi.com/products/Xbee-rf-solutions/xctu-software/xctu#productsupport [04/08/2017]
44. Digi. *Digi Xbee Zigbee* https://www.digi.com/products/Xbee-rf-solutions/2-4-ghz-modules/Xbee-zigbee#specifications [22/07/2017]
45. Digi. *Sleep Mode* http://docs.digi.com/display/XbeeArduinoCodingPlatform/Sleep+mode [23/08/2017]
46. DirectivaRoHSaprobadaporelparlamentoeuropeo24112010.pdf. (s. f.). Recuperado a partir de http://www.relec.es/RECICLADO_ELECTRONICO/Legislacion/DirectivaRoHSaprobadaporelparlamentoeuropeo24112010.pdf
47. Durda, Frank. *Serial and Uart Tutorial.* https://www.freebsd.org/doc/en_US.ISO8859-1/articles/serial-uart/ [16/07/2017]
48. EcuRed. *Microcontrolador.* https://www.ecured.cu/Microcontrolador [25.08.17]
49. EcuRed. Temperatura Corporal. https://www.ecured.cu/Temperatura_corporal#Medici.C3.B3n_de_la_temperatura_corporal [25/09/17]
50. Electronoobs. (s. f.). *Burn bootloader to ATmega328p AU with Arduino NANO*. Recuperado a partir de https://www.youtube.com/watch?v=dpgcBsl9D4k
51. Embedded Systems Tutorials. (s. f.). *Altium Designer Tutorial 3- Creating Gerber files*. Recuperado a partir de https://www.youtube.com/watch?v=UlKPZ0K8mAM

52. Errores comunes en el diseño de PCB con componentes SMT. (2014, octubre 15). Recuperado 13 de julio de 2017, a partir de http://microensamble.com/errores-comunes-en-el-diseno-de-pcbs-usando-componentes-smt/
53. EUR-Lex - 32011L0065 - EN - EUR-Lex. (s. f.). Recuperado 6 de octubre de 2017, a partir de http://eur-lex.europa.eu/legal-content/ES/TXT/?uri=celex%3A32011L0065
54. Federación Nacional de Taekwondo. *Historia del Taekwondo*. http://www.tkdguatemala.net/nosotros/historia.aspx [18/08/17]
55. Federación Venezolana de Tiro. 2015. *Manual de Introducción al deporte del tiro.* Comité Olímpico de Venezuela. Venezuela. 33 págs.
56. FitLight. *The Fitlight Trainer*. https://www.fitlighttraining.com/the-fitlight-trainer/ [05/10/17]
57. Freescale Semiconductor. *2008*. *MMA7361L.* Arizona. 11 págs.
58. Frenzel, Low. *What's the Difference Between IEEE 802.15.4 And ZigBee Wireless?* http://www.electronicdesign.com/what-s-difference-between/what-s-difference-between-ieee-802154-and-zigbee-wireless [22/07/2017]
59. Gallardo, Omar. 2015 «Fabricación de circuito impreso con Proteus.». Tesis Universidad de Valladolid 119 págs.
60. García, Antony. *Librería PanamaHitek_Arduino, v2.8.2* http://panamahitek.com/libreria-panamahitek_arduino/ [05/08/17]
61. Gimbal Systems. *Gimbal Systems*. http://gimbalsystems.com/about-gimbal-systems/ [23/07/17]
62. González, A. G. (2015, mayo 20). ¿Qué es Arduino y para qué se utiliza? Recuperado 13 de julio de 2017, a partir de http://panamahitek.com/que-es-arduino-y-para-que-se-utiliza/
63. Hamming, Richard. 1989. *Digital Filters.* 3ª ed. Nueva York: Prentice-Hall. 290 págs.
64. Hibbeler, Ross. 2004. *Mecánica Vectorial Para Ingenieros: Dinámica.* 10ª ed. México D.F.: Pearson Educación. 692 págs.
65. Humberto Higinio. (s. f.-a). *Construye tu propio Arduino UNO - Carga de Bootloader*. Recuperado a partir de https://www.youtube.com/watch?v=XgFwf5iJjWc
66. Humberto Higinio. (s. f.-b). *Construye tu propio Arduino UNO - DIY Arduino Clone*. Recuperado a partir de https://www.youtube.com/watch?v=XgFwf5iJjWc
67. iLED PERU. *Tecnología LED.* http://www.iledperu.com/tecnologialed [25/08/17]
68. Ingenieria Asistida por Computadora. *AutoDesk.* http://www.iac.com.co/autodesk-inventor/ [05/10/17]
69. Invensense. *MPU-6000 and MPU-6050 Product Specification Revision 3.4*. https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf [12/07/17]
70. ISSF. 2013. *Official Statutes Rules and Regulations.* ISFF. Munich. 497 págs.
71. Jarvis, Matt. 2006. «Introduction». Sports Psychology A Student's Handbook. Canada: Routledge. págs. 1-11.

72. Kalinsky, David; R. Kalinsky. *Introduction to Serial Peripheral Interface.* http://www.embedded.com/electronics-blogs/beginner-s-corner/4023908/Introduction-to-Serial-Peripheral-Interface [17/07/2017].
73. LION Precision. *Capacitive Sensor Operation and Optimization.* http://www.lionprecision.com/tech-library/technotes/cap-0020-sensor-theory.html [25/08/17]
74. López, Oliverio. 2016. «Diseño y elaboración de un control remoto para lámparas de alumbrado público.» Tesis Universidad Nacional Autónoma de México 89 págs
75. Luis Vasquez. (13:54:02 UTC). *Tecnicas de elaboracion de PCI - PCB*. Education. Recuperado a partir de https://www.slideshare.net/mcmax911/tecnicas-de-elaboracion-de-pci-pcb
76. Luna, Lizbeth. 2004. «El diseño de interfaz gráfica de usuario para publicaciones digitales». *Revista Digital Universitaria.* V (7): 2-12
77. Lutovac, Miroslav; D. Tošić y B. Evans. 2001. *Filter Design for Signal Processing Using MATLAB and Mathematica*, Belgrado: Singidunum University.
78. Madgwick, Sebastian. 2010. *An effcient orientation fillter for inertial and inertial/magnetic sensor arrays.* http://x-io.co.uk/res/doc/madgwick_internal_report.pdf [07/08/17]
79. Marca. *Tiro Olímpico.* http://www.marca.com/juegos-olimpicos/tiro-olimpico/todo-sobre.html [04/09/17]
80. MaxStream. *Xbee Series 2 OEM RF Modules.* http://www.farnell.com/datasheets/27606.pdf [17/08/17]
81. Mayo Clinic. *Biorretroalimentación.* http://www.mayoclinic.org/es-es/tests-procedures/biofeedback/home/ovc-20169724 [25/09/17]
82. MCI Electronics. *¿Qué es Xbee?* from http://Xbee.cl/que-es-Xbee/ [25.08.17]
83. MikroElektronika. *Introducción Al Mundo De Los Microcontroladores.* https://learn.mikroe.com/ebooks/microcontroladorespicbasic/chapter/introduccion-al-mundo-de-los-microcontroladores/ [17/07/2017].
84. MVN BIOMECH - Products. (s. f.). Recuperado 18 de junio de 2017, a partir de https://www.xsens.com/products/mvn-biomech/
85. National Instruments. *Comunicación Serial: Conceptos Generales.* http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1 [16/07/2017].
86. Navarro, Kiara.*¿Cómo funciona el protocolo SPI?* http://panamahitek.com/como-funciona-el-protocolo-spi/ [17/07/2017].
87. Nelson Mandela University. *Sport Science.* http://hms.mandela.ac.za/Biokinetics-Sport-Science-Unit/Sport-Science [18/08/17]
88. news_pdf_ar_157696.pdf. (s. f.-a). Recuperado a partir de https://www.tuv.com/es/argentina/about_us_ar/press_ar/news_pdf_ar_157696.jsp
89. news_pdf_ar_157696.pdf. (s. f.-b). Recuperado a partir de https://www.tuv.com/es/argentina/about_us_ar/press_ar/news_pdf_ar_157696.jsp

90. Oliver, Andrew, *et Al.* 2017. *Apache POI- The Java Api For Microsoft.* https://poi.apache.org/index.html [05/08/2017].
91. PCB Design Software | Circuit Design | PCB Tools | Altium. (s. f.). Recuperado 3 de septiembre de 2017, a partir de http://www.altium.com/
92. Pérez. J. J. 2014. <<El Tiempo de Reacción Específico Visual en Deportes de Combate>>. Tesis Universidad Autónoma de Madrid. 293 págs.
93. Placas de circuito impreso (PCB). (s. f.). Recuperado 4 de agosto de 2017, a partir de http://www.lcardaba.com/projects/placas/placas.htm
94. Prometec, *Módulo Bluetooth HC-05*. https://www.prometec.net/producto/modulo-bluetooth-hc-05/ [05/08/2017].
95. Prometec. *HC Serial Bluetooth Products User Instructional Manual*. https://cdn.makezine.com/uploads/2014/03/hc_hc-05-user-instructions-bluetooth.pdf [11/07/17]
96. Ramón, Gustavo. 2009. *Biomecánica deportiva y control del entrenamiento.* Medellín: Funámbulos. 134 págs.
97. Reas, Casey y B. Fry. 2014. *Processing: A programming handbook for visual designers and artists.* 2a ed. Massachusetts: MIT Press. 720 págs.
98. Rodriguez, Juan; M. Tena. 2012. «ECG Recording and Heart Rate Detection with Textile-Electronic Integrated Instrumentation. » Tesis Universidad de Boras 51 págs.
99. Rodríguez, Victor. 2012. *Sobre los cuaterniones, álgebras de Lie, y matrices de Pauli*. Universidad de Oviedo. España: Editorial de la Universidad de Oviedo. 18 págs.
100. RoHS2 (Directiva – 2011/65/CE) – Restricción de sustancias peligrosas en aparatos eléctricos y electrónicos | ar | TÜV Rheinland. (s. f.). Recuperado 3 de octubre de 2017, a partir de https://www.tuv.com/es/argentina/about_us_ar/press_ar/noticias_detalle_157696.html
101. Sensores biomecanicos, aplicaciones analisis emg. (s. f.). Recuperado 18 de junio de 2017, a partir de http://www.biomech-solutions.com/sensores-biomecanicos-noraxon.html
102. SENSORES INERCIALES - Análisis de Movimiento - ISEN - STT. (s. f.). Recuperado 25 de junio de 2017, a partir de http://www.biomec.com.co/productos/sensores-inerciales.html
103. Sichling & CO. 2017. *Escopetas.* http://www.armeriasichling.com/12-escopetas
104. snva419c.pdf. (s. f.). Recuperado a partir de http://www.ti.com/lit/an/snva419c/snva419c.pdf
105. SparkFun Xbee Explorer Regulated - WRL-11373 - SparkFun Electronics. (s. f.). Recuperado 17 de agosto de 2017, a partir de https://www.sparkfun.com/products/11373
106. Sparkfun. 2017. *Make your Own Fritzing Parts.* https://learn.sparkfun.com/tutorials/make-your-own-fritzing-parts [05/08/2017]
107. Starlino. *A Guide To using IMU (Accelerometer and Gyroscope Devices) in Embedded Applications.* http://www.instructables.com/id/Accelerometer-Gyro-Tutorial/ [16/07/17]
108. Suarez, R. (2009). Biomecánica deportiva y control del entrenamiento.

109. TecBolivia. (s. f.). *Arduino Simple y de Bajo Costo: Breadbo-Arduino*. Recuperado a partir de https://www.youtube.com/watch?v=3D5gwxtuSlI
110. Texas Heart Institute. *Electrocardiograma.* http://www.texasheart.org/HIC/Topics_Esp/Diag/diekg_sp.cfm [25/09/17]
111. Texas Instrument Incorporated. 2014. *Basics of Capacitive Sensing and Applications.* Texas, Dallas. 11 págs.
112. Texas Instruments. *2012. KeyStone Architecture Serial Peripheral Interface (SPI)*. Texas. 51 págs.
113. The British Association of Sport and Excercise Sciences. *About Sport and Exercise Science.* http://www.bases.org.uk/About-Sport-and-Exercise-Science [18/08/17]
114. Times Higher Education's. *What can you do with a sports science degree?* https://www.timeshighereducation.com/student/subjects/what-can-you-do-sports-science-degree [18/09/17]
115. UA78M 500-mA, 25-V, Linear Voltage Regulators | TI.com. (s. f.). Recuperado 27 de septiembre de 2017, a partir de http://www.ti.com/product/UA78M?dcmp=dsproject&hqs=pf
116. Universidad de Buenos Aires. *2003. Introducción a los Cuaterniones*. Universidad de Buenos Aires. Argentina: Universidad de Buenos Aires. 6 págs.
117. Universidad Nacional de Colombia. 2002. Métodos Numéricos Para Ingeniería. Universidad Nacional de Colombia. Colombia: Universidad Nacional de Colombia. 69 págs.
118. Universidad Tecnológica Nacional. 2008. Introducción a los Cuaterniones. Universidad Tecnológica Nacional. Argentina: Editorial de la Universidad Tecnológica Nacional. 34 págs.
119. University of Cambridge. 2007. An introduction to inertial navigation. University of Cambridge. England: University of Cambridge. 37 págs.
120. University of Maryland. *Biofeedback.* http://www.umm.edu/health/medical/altmed/treatment/biofeedback [18/08/17]
121. Velasco, Nicolás. «Sistema embebido para la conexión de un PLC Siemens S7-200 a la red GSM.». Tesis Universidad de Sevilla 146 págs.
122. viacurrents.pdf. (s. f.). Recuperado a partir de http://www.ultracad.com/articles/viacurrents.pdf
123. Vicente, Miguel¸ *et Al.* 1998. *Educación física e deporte no século XXI.* Coruña: Universidad de Coruña. 161 págs.
124. Xbee. 2017 *¿Qué es Xbee?* http://Xbee.cl/que-es-Xbee/ [22/07/2017]
125. Xbee. 2017 *Xbee Configuración Serial* http://Xbee.cl/Xbee-serie-1-configuracion/ [20/02/2017]
126. XIO Technologies. *Open source IMU and AHRS algorithms*. http://x-io.co.uk/open-source-imu-and-ahrs-algorithms/ [16/05/17]
127. Yucha, Carolyn y C. Gilbert. 2004. *Evidence-Based Practice In Biofeedback and Neurofeedback.* Florida: AAPB. 59 págs.

XI. ANEXO

A. PCB

A continuación, encontrará imágenes relevantes al diseño de los PCB’s de este megaproyecto.

1. Sensor Inercial

Figura 139. Diseño del esquemático del circuito del componente IMU BNO055 en Altium Designer

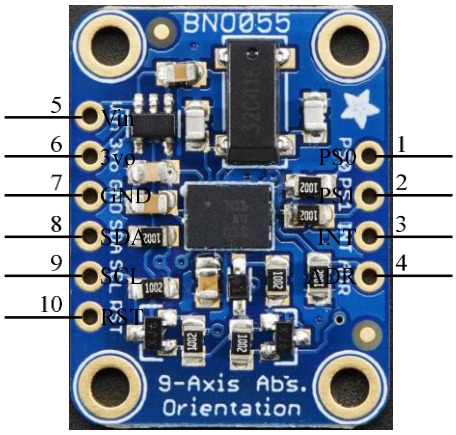


Figura 140. Diseño de footprint del componente IMU BNO055 en Altium Designer

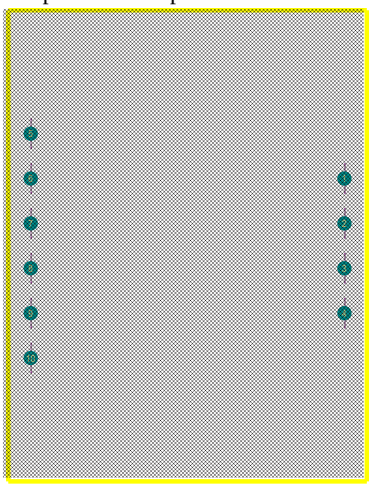


Figura 141. Diseño de 3D del componente IMU BNO055 en Altium Designer

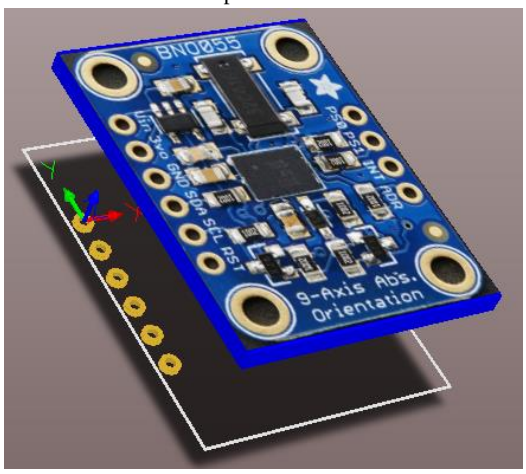


Figura 142. Esquemático regulador de voltaje UA78M05CDCY en Altium Designer

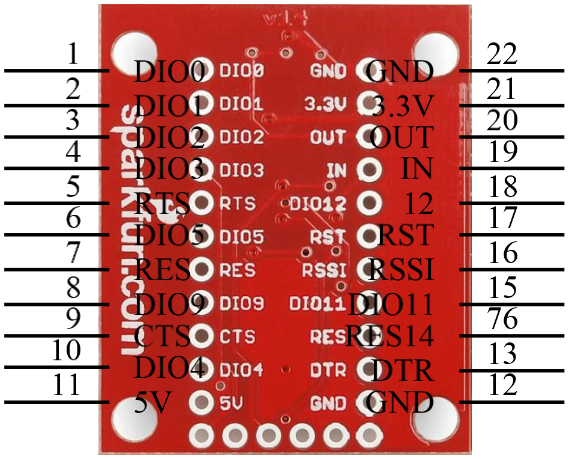


Figura 143. Footprint Xbee Explorer Regulator en Altium Designer

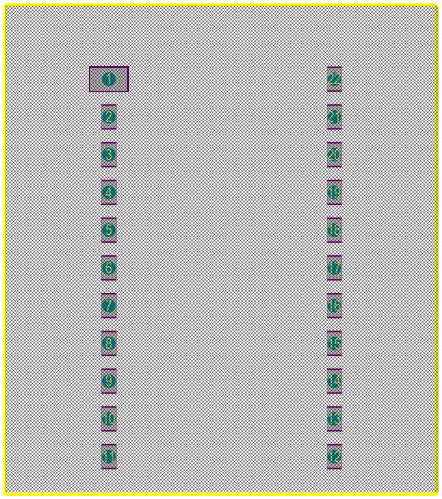


Figura 144. Diseño de 3D del componente Xbee Explorer Regulator en Altium Designer

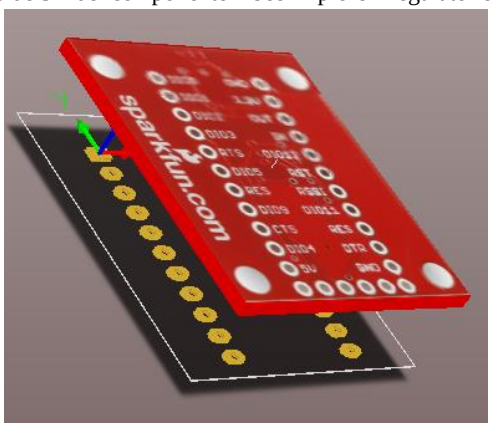


Figura 145. Esquemático del circuito del componente Regulador de voltaje UA78M05CDCY en Altium Designer

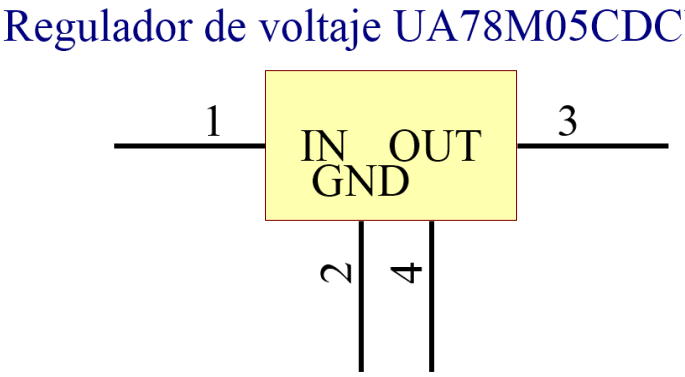


Figura 146. Footprint regulador de voltaje UA78M05CDCY en Altium Designer

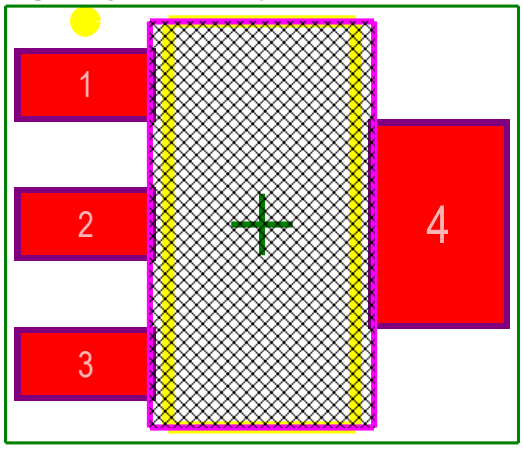


Figura 147. Diseño 3D del componente regulador de voltaje UA78M05CDCY en Altium Designer

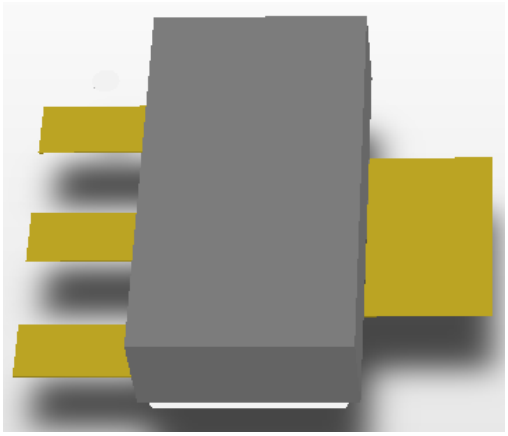


Figura 148. Footpirnt cristal oscilador de 16MHz empaquetado Thru-hole en Altium Designer

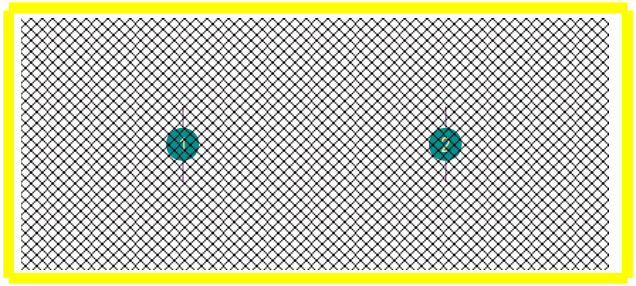


Figura 149. Diseño 3D cristal oscilador de 16MHz empaquetado Thru-hole, vista superior

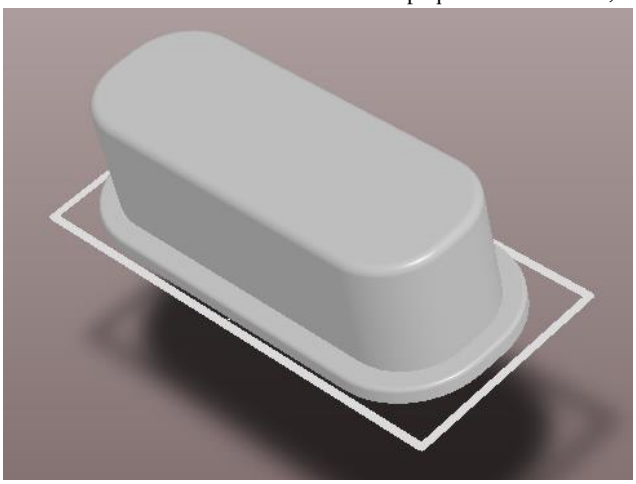


Figura 150. Diseño 3D cristal oscilador de 16MHz empaquetado Thru-hole, vista inferior

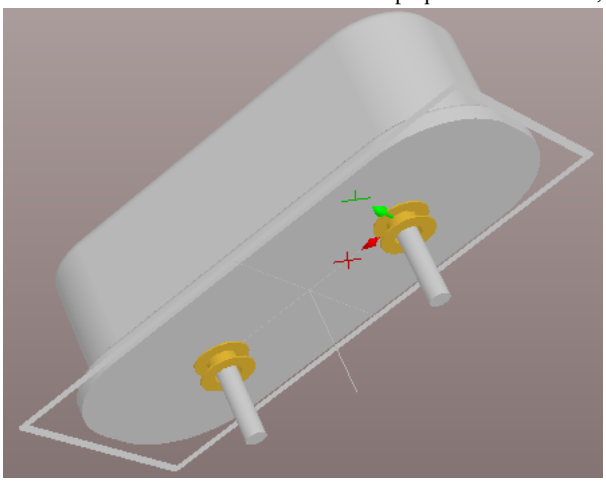


Figura 151. Footprint del componente botón SMD en Altium Designer

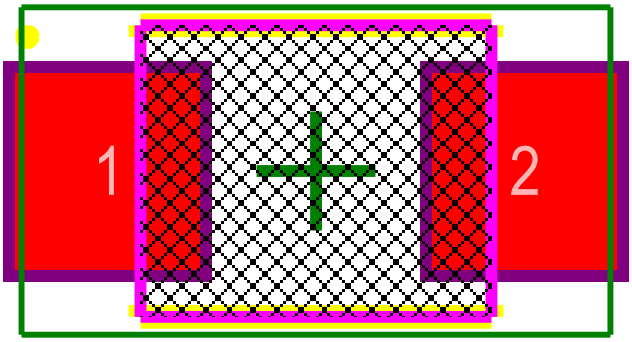


Figura 152. Diseño 3D del componente botón SMD en Altium Designer

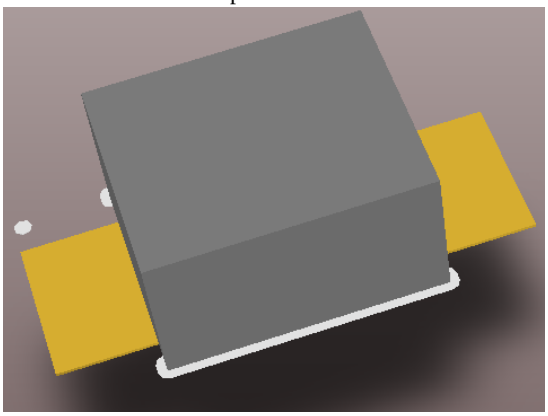


Figura 153. Diseño de footprint del componente LED SMD en Altium Designer

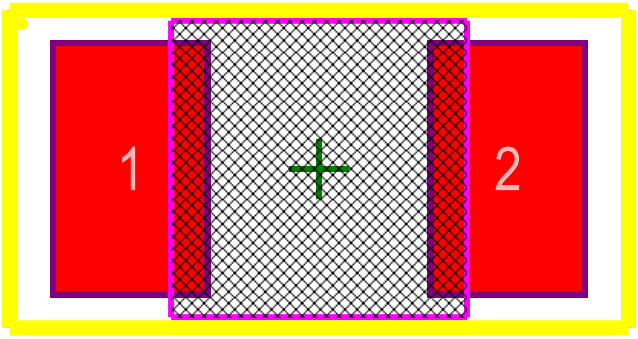
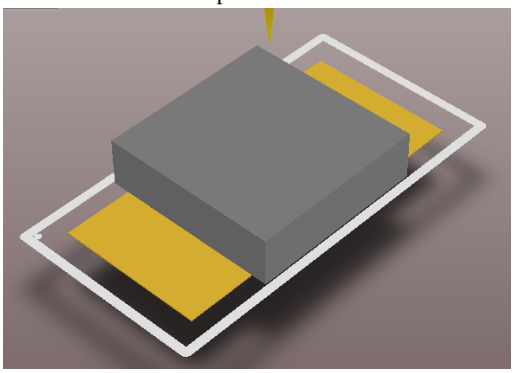


Figura 154. Diseño 3D del componente LED SMD en Altium Designer



2. Sensor tiro con armas de caza

Figura 155. Diseño PCB Bottom Layer.

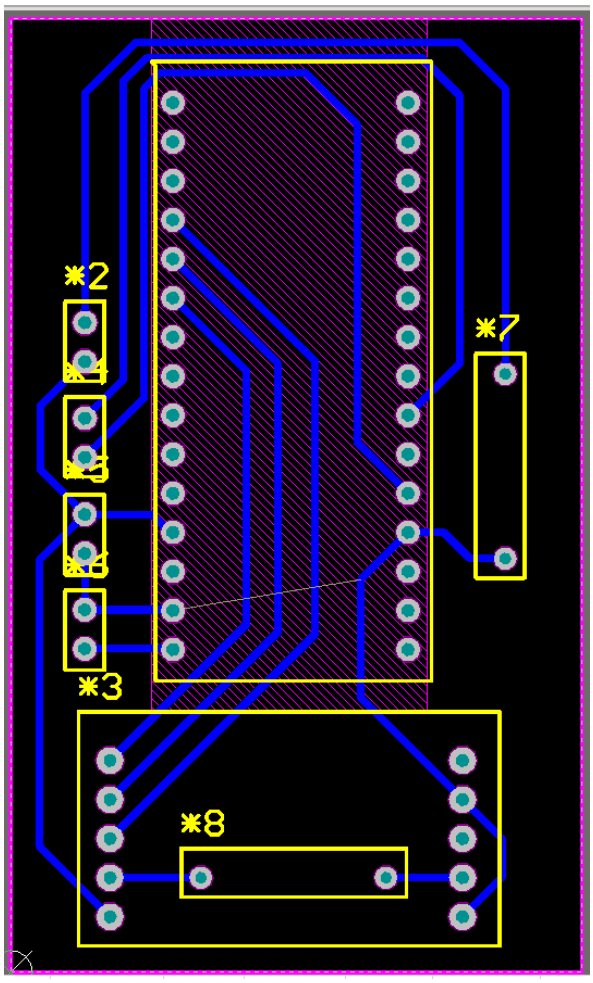
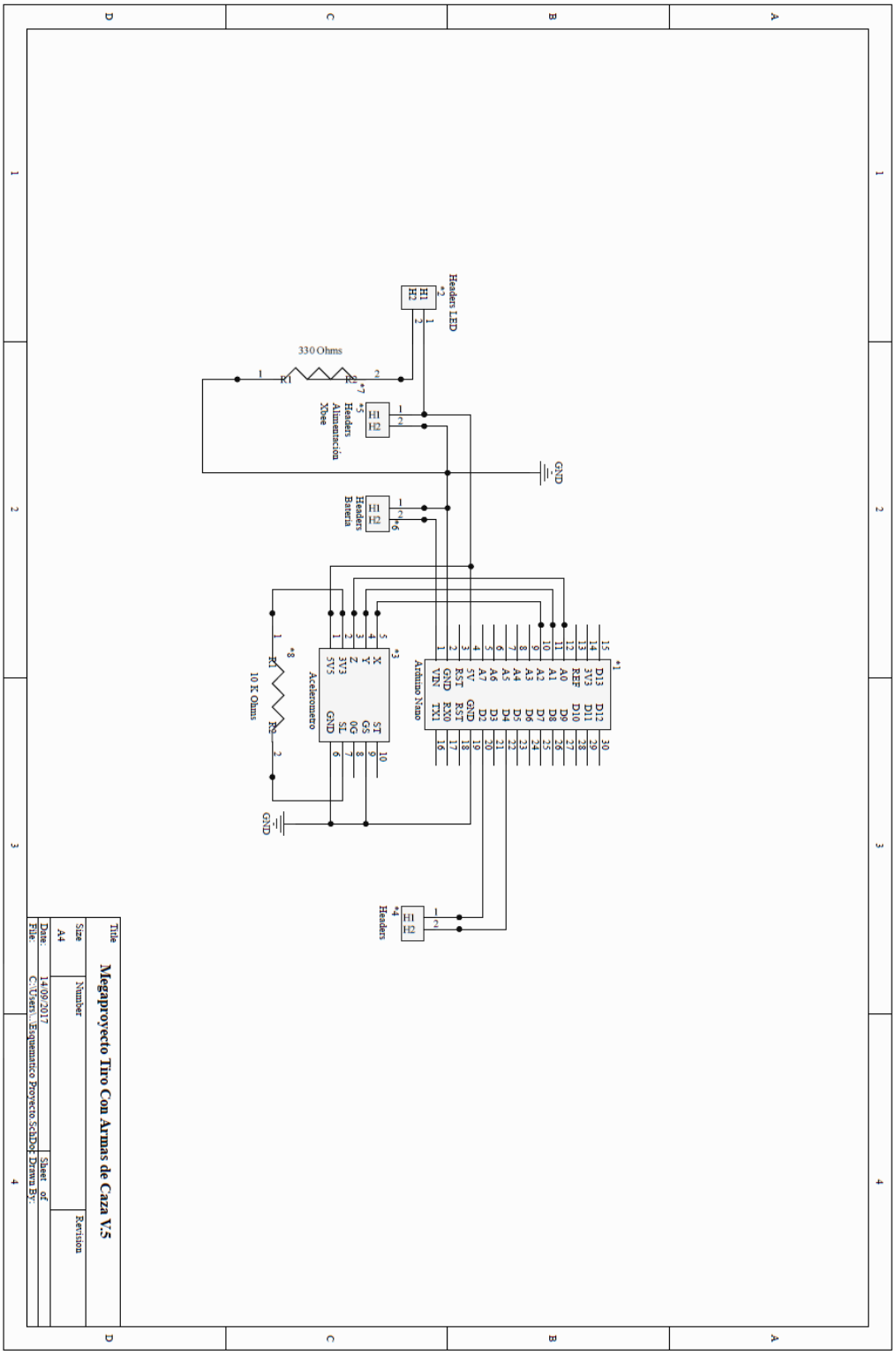


Figura 156. Esquemático PCB.



B. Códigos de programación

1. Bootloader ATMEGA 328P

```
// ArduinoISP
// Copyright (c) 2008-2011 Randall Bohn
// If you require a license, see
// http://www.opensource.org/licenses/bsd-license.php
//
// This sketch turns the Arduino into a AVRISP using the following Arduino pins:
//
// Pin 10 is used to reset the target microcontroller.
//
// By default, the hardware SPI pins MISO, MOSI and SCK are used to communicate
// with the target. On all Arduinos, these pins can be found
// on the ICSP/SPI header:
//
//               MISO °. . 5V (!) Avoid this pin on Due, Zero...
//               SCK   . . MOSI
//                     . . GND
//
// On some Arduinos (Uno,...), pins MOSI, MISO and SCK are the same pins as
// digital pin 11, 12 and 13, respectively. That is why many tutorials instruct
// you to hook up the target to these pins. If you find this wiring more
// practical, have a define USE_OLD_STYLE_WIRING. This will work even when not
// using an Uno. (On an Uno this is not needed).
//
// Alternatively you can use any other digital pin by configuring
// software ('BitBanged') SPI and having appropriate defines for PIN_MOSI,
// PIN_MISO and PIN_SCK.
//
// IMPORTANT: When using an Arduino that is not 5V tolerant (Due, Zero, ...) as
// the programmer, make sure to not expose any of the programmer's pins to 5V.
// A simple way to accomplish this is to power the complete system (programmer
// and target) at 3V3.
//
// Put an LED (with resistor) on the following pins:
// 9: Heartbeat   - shows the programmer is running
// 8: Error       - Lights up if something goes wrong (use red if that makes sense)
// 7: Programming - In communication with the slave
//

#include "Arduino.h"
#undef SERIAL


#define PROG_FLICKER true

// Configure SPI clock (in Hz).
// E.g. for an ATtiny @ 128 kHz: the datasheet states that both the high and low
// SPI clock pulse must be > 2 CPU cycles, so take 3 cycles i.e. divide target
```

```
// f_cpu by 6:
//     #define SPI_CLOCK           (128000/6)
//
// A clock slow enough for an ATtiny85 @ 1 MHz, is a reasonable default:

#define SPI_CLOCK     (1000000/6)


// Select hardware or software SPI, depending on SPI clock.
// Currently only for AVR, for other architectures (Due, Zero,...), hardware SPI
// is probably too fast anyway.

#if defined(ARDUINO_ARCH_AVR)

#if SPI_CLOCK > (F_CPU / 128)
#define USE_HARDWARE_SPI
#endif

#endif

// Configure which pins to use:

// The standard pin configuration.
#ifndef ARDUINO_HOODLOADER2

#define RESET     10 // Use pin 10 to reset the target rather than SS
#define LED_HB    9
#define LED_ERR   8
#define LED_PMODE 7

// Uncomment following line to use the old Uno style wiring
// (using pin 11, 12 and 13 instead of the SPI header) on Leonardo, Due...

// #define USE_OLD_STYLE_WIRING

#ifdef USE_OLD_STYLE_WIRING

#define PIN_MOSI  11
#define PIN_MISO  12
#define PIN_SCK   13

#endif

// HOODLOADER2 means running sketches on the ATmega16U2 serial converter chips
// on Uno or Mega boards. We must use pins that are broken out:
#else

#define RESET       4
#define LED_HB      7
#define LED_ERR     6
#define LED_PMODE   5

#endif

// By default, use hardware SPI pins:
#ifndef PIN_MOSI
```

```
#define PIN_MOSI  MOSI
#endif

#ifndef PIN_MISO
#define PIN_MISO  MISO
#endif

#ifndef PIN_SCK
#define PIN_SCK   SCK
#endif

// Force bitbanged SPI if not using the hardware SPI pins:
#if (PIN_MISO != MISO) ||  (PIN_MOSI != MOSI) || (PIN_SCK != SCK)
#undef USE_HARDWARE_SPI
#endif


// Configure the serial port to use.
//
// Prefer the USB virtual serial port (aka. native USB port), if the Arduino has one:
//   - it does not autoreset (except for the magic baud rate of 1200).
//   - it is more reliable because of USB handshaking.
//
// Leonardo and similar have an USB virtual serial port: 'Serial'.
// Due and Zero have an USB virtual serial port: 'SerialUSB'.
//
// On the Due and Zero, 'Serial' can be used too, provided you disable autoreset.
// To use 'Serial': #define SERIAL Serial

#ifdef SERIAL_PORT_USBVIRTUAL
#define SERIAL SERIAL_PORT_USBVIRTUAL
#else
#define SERIAL Serial
#endif


// Configure the baud rate:

#define BAUDRATE  19200
// #define BAUDRATE 115200
// #define BAUDRATE 1000000


#define HWVER 2
#define SWMAJ 1
#define SWMIN 18

// STK Definitions
#define STK_OK      0x10
#define STK_FAILED  0x11
#define STK_UNKNOWN 0x12
#define STK_INSYNC  0x14
#define STK_NOSYNC  0x15
#define CRC_EOP     0x20 //ok it is a space...

void pulse(int pin, int times);
```

```
#ifdef USE_HARDWARE_SPI
#include "SPI.h"
#else

#define SPI_MODE0 0x00

class SPISettings {
  public:
    // clock is in Hz
    SPISettings(uint32_t clock, uint8_t bitOrder, uint8_t dataMode) : clock(clock) {
      (void) bitOrder;
      (void) dataMode;
    };

  private:
    uint32_t clock;

    friend class BitBangedSPI;
};

class BitBangedSPI {
  public:
    void begin() {
      digitalWrite(PIN_SCK, LOW);
      digitalWrite(PIN_MOSI, LOW);
      pinMode(PIN_SCK, OUTPUT);
      pinMode(PIN_MOSI, OUTPUT);
      pinMode(PIN_MISO, INPUT);
    }

    void beginTransaction(SPISettings settings) {
      pulseWidth = (500000 + settings.clock - 1) / settings.clock;
      if (pulseWidth == 0)
        pulseWidth = 1;
    }

    void end() {}

    uint8_t transfer (uint8_t b) {
      for (unsigned int i = 0; i < 8; ++i) {
        digitalWrite(PIN_MOSI, (b & 0x80) ? HIGH : LOW);
        digitalWrite(PIN_SCK, HIGH);
        delayMicroseconds(pulseWidth);
        b = (b << 1) | digitalRead(PIN_MISO);
        digitalWrite(PIN_SCK, LOW); // slow pulse
        delayMicroseconds(pulseWidth);
      }
      return b;
    }

  private:
    unsigned long pulseWidth; // in microseconds
};

static BitBangedSPI SPI;
```

```
#endif

void setup() {
  SERIAL.begin(BAUDRATE);

  pinMode(LED_PMODE, OUTPUT);
  pulse(LED_PMODE, 2);
  pinMode(LED_ERR, OUTPUT);
  pulse(LED_ERR, 2);
  pinMode(LED_HB, OUTPUT);
  pulse(LED_HB, 2);

}

int error = 0;
int pmode = 0;
// address for reading and writing, set by 'U' command
unsigned int here;
uint8_t buff[256]; // global block storage

#define beget16(addr) (*addr * 256 + *(addr+1) )
typedef struct param {
  uint8_t devicecode;
  uint8_t revision;
  uint8_t progtype;
  uint8_t parmode;
  uint8_t polling;
  uint8_t selftimed;
  uint8_t lockbytes;
  uint8_t fusebytes;
  uint8_t flashpoll;
  uint16_t eeprompoll;
  uint16_t pagesize;
  uint16_t eepromsize;
  uint32_t flashsize;
}
parameter;

parameter param;

// this provides a heartbeat on pin 9, so you can tell the software is running.
uint8_t hbval = 128;
int8_t hbdelta = 8;
void heartbeat() {
  static unsigned long last_time = 0;
  unsigned long now = millis();
  if ((now - last_time) < 40)
    return;
  last_time = now;
  if (hbval > 192) hbdelta = -hbdelta;
  if (hbval < 32) hbdelta = -hbdelta;
  hbval += hbdelta;
  analogWrite(LED_HB, hbval);
}
```

```
static bool rst_active_high;

void reset_target(bool reset) {
  digitalWrite(RESET, ((reset && rst_active_high) || (!reset && !rst_active_high)) ? HIGH : LOW);
}

void loop(void) {
  // is pmode active?
  if (pmode) {
    digitalWrite(LED_PMODE, HIGH);
  } else {
    digitalWrite(LED_PMODE, LOW);
  }
  // is there an error?
  if (error) {
    digitalWrite(LED_ERR, HIGH);
  } else {
    digitalWrite(LED_ERR, LOW);
  }

  // light the heartbeat LED
  heartbeat();
  if (SERIAL.available()) {
    avrisp();
  }
}

uint8_t getch() {
  while (!SERIAL.available());
  return SERIAL.read();
}
void fill(int n) {
  for (int x = 0; x < n; x++) {
    buff[x] = getch();
  }
}

#define PTIME 30
void pulse(int pin, int times) {
  do {
    digitalWrite(pin, HIGH);
    delay(PTIME);
    digitalWrite(pin, LOW);
    delay(PTIME);
  } while (times--);
}

void prog_lamp(int state) {
  if (PROG_FLICKER) {
    digitalWrite(LED_PMODE, state);
  }
}

uint8_t spi_transaction(uint8_t a, uint8_t b, uint8_t c, uint8_t d) {
  SPI.transfer(a);
  SPI.transfer(b);
```

```
  SPI.transfer(c);
  return SPI.transfer(d);
}

void empty_reply() {
  if (CRC_EOP == getch()) {
    SERIAL.print((char)STK_INSYNC);
    SERIAL.print((char)STK_OK);
  } else {
    error++;
    SERIAL.print((char)STK_NOSYNC);
  }
}

void breply(uint8_t b) {
  if (CRC_EOP == getch()) {
    SERIAL.print((char)STK_INSYNC);
    SERIAL.print((char)b);
    SERIAL.print((char)STK_OK);
  } else {
    error++;
    SERIAL.print((char)STK_NOSYNC);
  }
}

void get_version(uint8_t c) {
  switch (c) {
    case 0x80:
      breply(HWVER);
      break;
    case 0x81:
      breply(SWMAJ);
      break;
    case 0x82:
      breply(SWMIN);
      break;
    case 0x93:
      breply('S'); // serial programmer
      break;
    default:
      breply(0);
  }
}

void set_parameters() {
  // call this after reading parameter packet into buff[]
  param.devicecode = buff[0];
  param.revision   = buff[1];
  param.progtype   = buff[2];
  param.parmode    = buff[3];
  param.polling    = buff[4];
  param.selftimed  = buff[5];
  param.lockbytes  = buff[6];
  param.fusebytes  = buff[7];
  param.flashpoll  = buff[8];
  // ignore buff[9] (= buff[8])
```

```
  // following are 16 bits (big endian)
  param.eeprompoll = beget16(&buff[10]);
  param.pagesize   = beget16(&buff[12]);
  param.eepromsize = beget16(&buff[14]);

  // 32 bits flashsize (big endian)
  param.flashsize = buff[16] * 0x01000000
                  + buff[17] * 0x00010000
                  + buff[18] * 0x00000100
                  + buff[19];

  // AVR devices have active low reset, AT89Sx are active high
  rst_active_high = (param.devicecode >= 0xe0);
}

void start_pmode() {

  // Reset target before driving PIN_SCK or PIN_MOSI

  // SPI.begin() will configure SS as output, so SPI master mode is selected.
  // We have defined RESET as pin 10, which for many Arduinos is not the SS pin.
  // So we have to configure RESET as output here,
  // (reset_target() first sets the correct level)
  reset_target(true);
  pinMode(RESET, OUTPUT);
  SPI.begin();
  SPI.beginTransaction(SPISettings(SPI_CLOCK, MSBFIRST, SPI_MODE0));

  // See AVR datasheets, chapter "SERIAL_PRG Programming Algorithm":

  // Pulse RESET after PIN_SCK is low:
  digitalWrite(PIN_SCK, LOW);
  delay(20); // discharge PIN_SCK, value arbitrarily chosen
  reset_target(false);
  // Pulse must be minimum 2 target CPU clock cycles so 100 usec is ok for CPU
  // speeds above 20 KHz
  delayMicroseconds(100);
  reset_target(true);

  // Send the enable programming command:
  delay(50); // datasheet: must be > 20 msec
  spi_transaction(0xAC, 0x53, 0x00, 0x00);
  pmode = 1;
}

void end_pmode() {
  SPI.end();
  // We're about to take the target out of reset so configure SPI pins as input
  pinMode(PIN_MOSI, INPUT);
  pinMode(PIN_SCK, INPUT);
  reset_target(false);
  pinMode(RESET, INPUT);
  pmode = 0;
}

void universal() {
```

```
  uint8_t ch;

  fill(4);
  ch = spi_transaction(buff[0], buff[1], buff[2], buff[3]);
  breply(ch);
}

void flash(uint8_t hilo, unsigned int addr, uint8_t data) {
  spi_transaction(0x40 + 8 * hilo,
                  addr >> 8 & 0xFF,
                  addr & 0xFF,
                  data);
}
void commit(unsigned int addr) {
  if (PROG_FLICKER) {
    prog_lamp(LOW);
  }
  spi_transaction(0x4C, (addr >> 8) & 0xFF, addr & 0xFF, 0);
  if (PROG_FLICKER) {
    delay(PTIME);
    prog_lamp(HIGH);
  }
}

unsigned int current_page() {
  if (param.pagesize == 32) {
    return here & 0xFFFFFFF0;
  }
  if (param.pagesize == 64) {
    return here & 0xFFFFFFE0;
  }
  if (param.pagesize == 128) {
    return here & 0xFFFFFFC0;
  }
  if (param.pagesize == 256) {
    return here & 0xFFFFFF80;
  }
  return here;
}


void write_flash(int length) {
  fill(length);
  if (CRC_EOP == getch()) {
    SERIAL.print((char) STK_INSYNC);
    SERIAL.print((char) write_flash_pages(length));
  } else {
    error++;
    SERIAL.print((char) STK_NOSYNC);
  }
}

uint8_t write_flash_pages(int length) {
  int x = 0;
  unsigned int page = current_page();
  while (x < length) {
```

```
    if (page != current_page()) {
      commit(page);
      page = current_page();
    }
    flash(LOW, here, buff[x++]);
    flash(HIGH, here, buff[x++]);
    here++;
  }

  commit(page);

  return STK_OK;
}

#define EECHUNK (32)
uint8_t write_eeprom(unsigned int length) {
  // here is a word address, get the byte address
  unsigned int start = here * 2;
  unsigned int remaining = length;
  if (length > param.eepromsize) {
    error++;
    return STK_FAILED;
  }
  while (remaining > EECHUNK) {
    write_eeprom_chunk(start, EECHUNK);
    start += EECHUNK;
    remaining -= EECHUNK;
  }
  write_eeprom_chunk(start, remaining);
  return STK_OK;
}
// write (length) bytes, (start) is a byte address
uint8_t write_eeprom_chunk(unsigned int start, unsigned int length) {
  // this writes byte-by-byte, page writing may be faster (4 bytes at a time)
  fill(length);
  prog_lamp(LOW);
  for (unsigned int x = 0; x < length; x++) {
    unsigned int addr = start + x;
    spi_transaction(0xC0, (addr >> 8) & 0xFF, addr & 0xFF, buff[x]);
    delay(45);
  }
  prog_lamp(HIGH);
  return STK_OK;
}

void program_page() {
  char result = (char) STK_FAILED;
  unsigned int length = 256 * getch();
  length += getch();
  char memtype = getch();
  // flash memory @here, (length) bytes
  if (memtype == 'F') {
    write_flash(length);
    return;
  }
  if (memtype == 'E') {
```

```
    result = (char)write_eeprom(length);
    if (CRC_EOP == getch()) {
      SERIAL.print((char) STK_INSYNC);
      SERIAL.print(result);
    } else {
      error++;
      SERIAL.print((char) STK_NOSYNC);
    }
    return;
  }
  SERIAL.print((char)STK_FAILED);
  return;
}

uint8_t flash_read(uint8_t hilo, unsigned int addr) {
  return spi_transaction(0x20 + hilo * 8,
                         (addr >> 8) & 0xFF,
                         addr & 0xFF,
                         0);
}

char flash_read_page(int length) {
  for (int x = 0; x < length; x += 2) {
    uint8_t low = flash_read(LOW, here);
    SERIAL.print((char) low);
    uint8_t high = flash_read(HIGH, here);
    SERIAL.print((char) high);
    here++;
  }
  return STK_OK;
}

char eeprom_read_page(int length) {
  // here again we have a word address
  int start = here * 2;
  for (int x = 0; x < length; x++) {
    int addr = start + x;
    uint8_t ee = spi_transaction(0xA0, (addr >> 8) & 0xFF, addr & 0xFF, 0xFF);
    SERIAL.print((char) ee);
  }
  return STK_OK;
}

void read_page() {
  char result = (char)STK_FAILED;
  int length = 256 * getch();
  length += getch();
  char memtype = getch();
  if (CRC_EOP != getch()) {
    error++;
    SERIAL.print((char) STK_NOSYNC);
    return;
  }
  SERIAL.print((char) STK_INSYNC);
  if (memtype == 'F') result = flash_read_page(length);
  if (memtype == 'E') result = eeprom_read_page(length);
```

```
  SERIAL.print(result);
}

void read_signature() {
  if (CRC_EOP != getch()) {
    error++;
    SERIAL.print((char) STK_NOSYNC);
    return;
  }
  SERIAL.print((char) STK_INSYNC);
  uint8_t high = spi_transaction(0x30, 0x00, 0x00, 0x00);
  SERIAL.print((char) high);
  uint8_t middle = spi_transaction(0x30, 0x00, 0x01, 0x00);
  SERIAL.print((char) middle);
  uint8_t low = spi_transaction(0x30, 0x00, 0x02, 0x00);
  SERIAL.print((char) low);
  SERIAL.print((char) STK_OK);
}
////////////////////////////////////////
////////////////////////////////////////


///////////////////////////////////
///////////////////////////////////
void avrisp() {
  uint8_t ch = getch();
  switch (ch) {
    case '0': // signon
      error = 0;
      empty_reply();
      break;
    case '1':
      if (getch() == CRC_EOP) {
        SERIAL.print((char) STK_INSYNC);
        SERIAL.print("AVR ISP");
        SERIAL.print((char) STK_OK);
      }
      else {
        error++;
        SERIAL.print((char) STK_NOSYNC);
      }
      break;
    case 'A':
      get_version(getch());
      break;
    case 'B':
      fill(20);
      set_parameters();
      empty_reply();
      break;
    case 'E': // extended parameters - ignore for now
      fill(5);
      empty_reply();
      break;
    case 'P':
      if (!pmode)
```

```
    start_pmode();
  empty_reply();
  break;
case 'U': // set address (word)
  here = getch();
  here += 256 * getch();
  empty_reply();
  break;

case 0x60: //STK_PROG_FLASH
  getch(); // low addr
  getch(); // high addr
  empty_reply();
  break;
case 0x61: //STK_PROG_DATA
  getch(); // data
  empty_reply();
  break;

case 0x64: //STK_PROG_PAGE
  program_page();
  break;

case 0x74: //STK_READ_PAGE 't'
  read_page();
  break;

case 'V': //0x56
  universal();
  break;
case 'Q': //0x51
  error = 0;
  end_pmode();
  empty_reply();
  break;

case 0x75: //STK_READ_SIGN 'u'
  read_signature();
  break;

// expecting a command, not CRC_EOP
// this is how we can get back in sync
case CRC_EOP:
  error++;
  SERIAL.print((char) STK_NOSYNC);
  break;

// anything else we will return STK_UNKNOWN
default:
  error++;
  if (CRC_EOP == getch())
    SERIAL.print((char)STK_UNKNOWN);
  else
    SERIAL.print((char)STK_NOSYNC);}}
```

2. Calculo de la trayectoria a partir de mediciones de los acelerómetros sensor inercial

```
#Se importan todas las librerías
from matplotlib import pyplot as plt
import numpy as np
import mpl_toolkits.mplot3d.axes3d as p3
from matplotlib import animation
import serial
import numpy as np
from scipy.signal import lfilter, firwin
#Se define el puerto que se estará utilizando
ser = serial.Serial('COM3', 9600, timeout=1)
ser.flushInput()
ser.flushOutput()
#Se inicializan las variables para las gráficas en tiempo real
fig = plt.figure()
ax = p3.Axes3D(fig)
#Se crea el archivo de texto que contendrá toda la información de las mediciones
open('Datos.txt','w')
#Se verifica que los datos que esén entrando contengan información
while ser.readline() == "":
    a=0
dataSerial = ser.readline()
#Se desechan los primeros cuatro datos, para obtener mediciones estables
while ser.readline() == dataSerial:
    a=0
dataSerial = ser.readline()
while ser.readline() == dataSerial:
    a=0
dataSerial = ser.readline()
datos = open('Datos.txt','a')
datos.write(dataSerial)
while ser.readline() == dataSerial:
    a=0
dataSerial = ser.readline()
datos = open('Datos.txt','a')
datos.write(dataSerial)
while ser.readline() == dataSerial:
    a=0
dataSerial = ser.readline()
datos = open('Datos.txt','a')
datos.write(dataSerial)
#Se comienza a medir
print("Midiendo")
#Función que calcula la trayectoria del sensor
```

```
def gen():
    #Se lee el dato en el puerto serial
    dataSerial = ser.readline()
    #Se coloca el dato al final del archivo de texto
    datos = open('Datos.txt','a')
    datos.write(dataSerial)
    #Se lee el archivo de texto completo
    graph_data = open('Datos.txt','r').read()
    #Se crea un arreglo que contiene todas las mediciones
    lines = graph_data.split('\n')
    #Se inicializan todas las variables que se estarán utilizando
    xs = []
    ys = []
    zs = []
    tMuestreo = 0.05
    aX =[]
    aY =[]
    aZ =[]
    vX =[]
    vY =[]
    vZ =[]
    posX = []
    posY = []
    posZ = []
    cont = 0
    aX.append(0)
    aY.append(0)
    aZ.append(0)
    vX.append(0)
    vY.append(0)
    vZ.append(0)
    posX.append(0)
    posY.append(0)
    posZ.append(0)
    #Se recorre el arreglo que contiene las mediciones
    for line in lines:
      if len(line)>1:
        #Se incrementa el contador
        cont+=1
        #Se crea un arreglo que contiene los datos individuales de una medicion
        datos = line.split(',')
        #Se guarda en accPrevia las aceleraciones en los tres ejes
        accFin = [accPrevia[0]*9.81,accPrevia[1]*9.81,accPrevia[2]*9.81]
        #Se guardan los valores individuales de aceleraómetros en sus ejes
        aX.append(accFin[0])
        aY.append(accFin[1])
        aZ.append(accFin[2])
        #Se filitran los acelerómetros
        aX = filtrar(4.5,6.5,aX)
        aY = filtrar(4.5,6.5,aY)
        aZ = filtrar(3.5,6.5,aZ)
        #Se obtiene la magnitud de la aceleración
        accMag = (accFin[0]**2+accFin[1]**2+accFin[2]**2)**0.5
        #Si la magnitud de la aceleración es menor a 0.5 m/s^2 se deascarta la
        #medición. Ésto para evitar desfases en las mediciones
        if(abs(accMag)<0.5):
            #En el caso que se descarte la medición del acelerómetro se le coloca
            #cero al vector de velocidad y el valor anterior al vector de posición
            vX.append(0)
            vY.append(0)
            vZ.append(0)
            posX.append(posX[cont-1])
            posY.append(posY[cont-1])
            posZ.append(posZ[cont-1])
```

```
        else:
            #Si la magnitud es mayor a 0.5 m/s^2, se realizan los cálculos
            #Se utiliza el algoritmo de Euler para integrar
            vX.append(vX[cont-1] + (aX[cont])*tMuestreo)
            vY.append(vY[cont-1] + (aY[cont])*tMuestreo)
            vZ.append(vZ[cont-1] + (aZ[cont])*tMuestreo)
            #Se filtra la velocidad en el eje Z
            vZ = filtrar(4,9.5,vZ)
            #Se integra de nuevo para obtener la posición
            posX.append(posX[cont-1] + (vX[cont])*tMuestreo*0.05)
            posY.append(posY[cont-1] + (vY[cont])*tMuestreo*0.05)
            posZ.append(posZ[cont-1] + (vZ[cont])*tMuestreo*0.05)
        #Se ingresa la posición el los vectores que se graficarán
        xs.append(posX[cont])
        ys.append(posY[cont])
        zs.append(posZ[cont])
    #Se recorren los vectores de posición y se colocan las triadas en una matriz
    for k in range(0, len(xs)-1):
        x=float(xs[k])
        y=float(ys[k])
        z=float(zs[k])
        yield np.array([x,y,z])

#Función que actualiza la posición actual
def update(num, data, line):
    #Se obtiene la matriz con todas las posiciones
    data = np.array(list(gen())).T
    line.set_data(data[:2, :num])
    line.set_3d_properties(data[2, :num])
#Función que filtra los datos con un filtro pasa banda

def filtrar(l,h,x):
    #La frecuencia de muestreo es de 20 Hz
    fs = 20.0
    #Se ingresa la frecuencia de corte del pasa bajas y pasa altas
    lowcut = l
    highcut = h
    ntaps = 128
    taps_hamming = bandpass_firwin(ntaps, lowcut, highcut, fs=fs)
    #Se quita el desfase de los datos agregandole ceros al vector de mediciones
    for i in range(0,63):
        x.append(0)
    #Se filtran los datos
    filtrado = lfilter(taps_hamming, 1.0, x)
    filtFin=[]
    #Al vector de datos filtrados se le quitan los ceros que se le adicionaron
    for i in range(0,len(filtrado)-63):
        filtFin.append(filtrado[i+63])
    return filtFin

#Filtro pasa banda Firwin
def bandpass_firwin(ntaps, lowcut, highcut, fs, window='hamming'):
    nyq = 0.5 * fs
    taps = firwin(ntaps, [lowcut, highcut], nyq=nyq, pass_zero=False,
                  window=window, scale=False)
    return taps

#Se obtienen los datos a graficar
data = np.array(list(gen())).T
#Se grafican los datos
line, = ax.plot(data[0, 0:1], data[1, 0:1], data[2, 0:1])
#Se invoca la función que genera las mediciones en tiempo real
ani = animation.FuncAnimation(fig, update, fargs=(data, line),interval=10)
#Se muestra la gráfica
plt.show()
```

3. Obtención de las mediciones de los sensores y envío de datos sensor inercial

```
//Se llaman a todas las librerías: Librería para I2C y librerías de la IMU
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>
//Se inicializa la IMU
#define BNO055_SAMPLERATE_DELAY_MS (100)
Adafruit_BNO055 bno = Adafruit_BNO055();
//Se crean todas la variables a utilizar en los cálculos
float angles[6];
float q[4];
float quat[4];
float qConj[4];
float accNorm[4];
float qMult[4];
float qMult1[4];
float accFin[3];
float accSig2[4];
float accSig[4];
float g[3];

void setup(void)
{
  //Se usará una BaudRate de 9600
  Serial.begin(9600);
  //Se espera a que se inicialice la IMU
  if(!bno.begin())
  {
    while(1);
  }
//Función que multiplica dos cuaterniones                              sensores
 void q_Mult(float q[],float a[], float b[]){
  q[0] = a[0] * b[0] - a[1] * b[1] - a[2] * b[2] - a[3] * b[3];
  q[1] = a[0] * b[1] + a[1] * b[0] + a[2] * b[3] - a[3] * b[2];
  q[2] = a[0] * b[2] - a[1] * b[3] + a[2] * b[0] + a[3] * b[1];
  q[3] = a[0] * b[3] + a[1] * b[2] - a[2] * b[1] + a[3] * b[0];
}
```

```
void loop(void)
{
  //Se leeel vector de aceleración
  imu::Vector<3> acel = bno.getVector(Adafruit_BNO055::VECTOR_ACCELEROMETER);
  //Se lee el vector de gravedad
  imu::Vector<3> grav = bno.getVector(Adafruit_BNO055::VECTOR_GRAVITY);
  //Se lee el vector de cuaterniones
  imu::Quaternion quat = bno.getQuat();
 //Se crean vectores estánar de los cuaterniones y de gravedad
  q[0] = quat.w();
  q[1] = quat.x();
  q[2] = quat.y();
  q[3] = quat.z();
  //Adicionalmente, se convierte el vector de gravedad a g
  g[0] = grav.x()/9.81;
  g[1] = grav.y()/9.81;
  g[2] = grav.z()/9.81;
  //Se obtiene el vector de conjugado el cuaternión
  qConj[0] = q[0];
  qConj[1] = -q[1];
  qConj[2] = -q[2];
  qConj[3] = -q[3];
  //Se coloca la aceleración en un vector con la forma de cuaternión
  //Adicionalmente, se le resta la gravedad a los ejes para que únicamente
  //tengamos las aceleraciónes externas, y no la de gravedad
  accNorm[0] = 0;
  accNorm[1] = acel.x()/9.81-g[0];
  accNorm[2] = acel.y()/9.81-g[1];
  accNorm[3] = acel.z()/9.81-g[2];
  //Se realiza la rotación del vector de acelerómetros por el cuaternión
  q_Mult(qMult1,q,accNorm);
  q_Mult(qMult,qMult1,qConj);
  //Finalemente se crea el vector con las tres aceleraciónes rotadas
  accFin[0] = qMult[1];
  accFin[1] = qMult[2];
  accFin[2] = qMult[3];
  //Se envía por el puerto serial los datos de las aceleraciones y los cuaterniones
  Serial.print(accFin[0]);
  Serial.print(",");
  Serial.print(accFin[1]);
  Serial.print(",");
  Serial.print(accFin[2]);
  Serial.print(",");
  Serial.print(q[0]);
  Serial.print(",");
  Serial.print(q[1]);
  Serial.print(",");
  Serial.print(q[2]);
  Serial.print(",");
  Serial.print(q[3]);
  Serial.print("\n");
  delay(50);
}
```

4. Algoritmo de Mahony para el cálculo de cuaterniones

```
void MahonyAHRSupdateIMU(float gx, float gy, float gz, float ax, float ay, float az) {
    float recipNorm;
    float halfvx, halfvy, halfvz;
    float halfex, halfey, halfez;
    float qa, qb, qc;

    // Compute feedback only if accelerometer measurement valid (avoids NaN in accelerometer normalisation)
    if(!((ax == 0.0f) && (ay == 0.0f) && (az == 0.0f))) {

        // Normalise accelerometer measurement
        recipNorm = invSqrt(ax * ax + ay * ay + az * az);
        ax *= recipNorm;
        ay *= recipNorm;
        az *= recipNorm;

        // Estimated direction of gravity and vector perpendicular to magnetic flux
        halfvx = q1 * q3 - q0 * q2;
        halfvy = q0 * q1 + q2 * q3;
        halfvz = q0 * q0 - 0.5f + q3 * q3;

        // Error is sum of cross product between estimated and measured direction of gravity
        halfex = (ay * halfvz - az * halfvy);
        halfey = (az * halfvx - ax * halfvz);
        halfez = (ax * halfvy - ay * halfvx);

        // Compute and apply integral feedback if enabled
        if(twoKi > 0.0f) {
            integralFBx += twoKi * halfex * (1.0f / sampleFreq);    // integral error scaled by Ki
            integralFBy += twoKi * halfey * (1.0f / sampleFreq);
            integralFBz += twoKi * halfez * (1.0f / sampleFreq);
            gx += integralFBx;  // apply integral feedback
            gy += integralFBy;
            gz += integralFBz;
        }
        else {
            integralFBx = 0.0f; // prevent integral windup
            integralFBy = 0.0f;
            integralFBz = 0.0f;
        }

        // Apply proportional feedback
        gx += twoKp * halfex;
        gy += twoKp * halfey;
        gz += twoKp * halfez;
    }

    // Integrate rate of change of quaternion
    gx *= (0.5f * (1.0f / sampleFreq));     // pre-multiply common factors
    gy *= (0.5f * (1.0f / sampleFreq));
    gz *= (0.5f * (1.0f / sampleFreq));
    qa = q0;
    qb = q1;
    qc = q2;
    q0 += (-qb * gx - qc * gy - q3 * gz);
    q1 += (qa * gx + qc * gz - q3 * gy);
    q2 += (qa * gy - qb * gz + q3 * gx);
    q3 += (qa * gz + qb * gy - qc * gx);

    // Normalise quaternion
    recipNorm = invSqrt(q0 * q0 + q1 * q1 + q2 * q2 + q3 * q3);
    q0 *= recipNorm;
    q1 *= recipNorm;
    q2 *= recipNorm;
    q3 *= recipNorm;
}
```

5. Algoritmo Madgwick para el cálculo de cuaterniones

```
void MadgwickAHRSupdateIMU(float gx, float gy, float gz, float ax, float ay, float az) {
    float recipNorm;
    float s0, s1, s2, s3;
    float qDot1, qDot2, qDot3, qDot4;
    float _2q0, _2q1, _2q2, _2q3, _4q0, _4q1, _4q2 ,_8q1, _8q2, q0q0, q1q1, q2q2, q3q3;

    // Rate of change of quaternion from gyroscope
    qDot1 = 0.5f * (-q1 * gx - q2 * gy - q3 * gz);
    qDot2 = 0.5f * (q0 * gx + q2 * gz - q3 * gy);
    qDot3 = 0.5f * (q0 * gy - q1 * gz + q3 * gx);
    qDot4 = 0.5f * (q0 * gz + q1 * gy - q2 * gx);

    // Compute feedback only if accelerometer measurement valid (avoids NaN in accelerometer normalisation)
    if(!((ax == 0.0f) && (ay == 0.0f) && (az == 0.0f))) {

        // Normalise accelerometer measurement
        recipNorm = invSqrt(ax * ax + ay * ay + az * az);
        ax *= recipNorm;
        ay *= recipNorm;
        az *= recipNorm;

        // Auxiliary variables to avoid repeated arithmetic
        _2q0 = 2.0f * q0;
        _2q1 = 2.0f * q1;
        _2q2 = 2.0f * q2;
        _2q3 = 2.0f * q3;
        _4q0 = 4.0f * q0;
        _4q1 = 4.0f * q1;
        _4q2 = 4.0f * q2;
        _8q1 = 8.0f * q1;
        _8q2 = 8.0f * q2;
        q0q0 = q0 * q0;
        q1q1 = q1 * q1;
        q2q2 = q2 * q2;
        q3q3 = q3 * q3;
        // Gradient decent algorithm corrective step
        s0 = _4q0 * q2q2 + _2q2 * ax + _4q0 * q1q1 - _2q1 * ay;
        s1 = _4q1 * q3q3 - _2q3 * ax + 4.0f * q0q0 * q1 - _2q0 * ay - _4q1 + _8q1 * q1q1 + _8q1 * q2q2 + _4q1 * az;
        s2 = 4.0f * q0q0 * q2 + _2q0 * ax + _4q2 * q3q3 - _2q3 * ay - _4q2 + _8q2 * q1q1 + _8q2 * q2q2 + _4q2 * az;
        s3 = 4.0f * q1q1 * q3 - _2q1 * ax + 4.0f * q2q2 * q3 - _2q2 * ay;
        recipNorm = invSqrt(s0 * s0 + s1 * s1 + s2 * s2 + s3 * s3); // normalise step magnitude
        s0 *= recipNorm;
        s1 *= recipNorm;
        s2 *= recipNorm;
        s3 *= recipNorm;

        // Apply feedback step
        qDot1 -= beta * s0;
        qDot2 -= beta * s1;
        qDot3 -= beta * s2;
        qDot4 -= beta * s3;
    }

    // Integrate rate of change of quaternion to yield quaternion
    q0 += qDot1 * (1.0f / sampleFreq);
    q1 += qDot2 * (1.0f / sampleFreq);
    q2 += qDot3 * (1.0f / sampleFreq);
    q3 += qDot4 * (1.0f / sampleFreq);

    // Normalise quaternion
    recipNorm = invSqrt(q0 * q0 + q1 * q1 + q2 * q2 + q3 * q3);
    q0 *= recipNorm;
    q1 *= recipNorm;
    q2 *= recipNorm;
    q3 *= recipNorm;
}
```

6. Cálculo de coeficiente de correlación sensor inercial

```
from scipy import stats
import numpy as np

graph_data = open('Datos.txt','r').read()
lines = graph_data.split('\n')
x = []
y = []
for line in lines:
  if len(line)>1:
    datos = line.split(',')
    x.append(float(datos[0]))
    y.append(float(datos[1]))

slope, intercept, r_value, p_value, std_err = stats.linregress(x,y)
print "Coeficiente:", r_value**2
```

7. Transformada rápida de Fourier sensor inercial

```
signal = load('Datos.txt');
N = length(signal);
fs = 20; % 62.5 samples per second
fnyquist = fs/2; %Nyquist frequency

X_mags = abs(fft(signal));
bin_vals = [0 : N-1];
fax_Hz = bin_vals*fs/N;
N_2 = ceil(N/2);
plot(fax_Hz(1:N_2), X_mags(1:N_2))
xlabel('Frecuencia (Hz)')
ylabel('|Yaw(t)|');
title('SKU:SEN0140');
axis tight
```

8. Interfaz gráfica sensor RGB

```
#!/usr/bin/python
    # -*- coding: iso-8859-1 -*-
    import time
    import serial
    import xlwt
    import xlrd
    from time import strftime
    import sys
    from xlrd import open_workbook
    import openpyxl

    # Interacción usuario
    user = str(raw_input("Nombre del Usuario: "))
    n = int(input("Ingrese el numero de pruebas: "))

    # Funcion que define la conexion entre arduino
    def serialConection():
        arduinoPort = serial.Serial('COM6', 9600, timeout=1)
        arduinoPort.flushInput()
        arduinoPort.setDTR()
        time.sleep(0.3)
        return arduinoPort

    # Funcion que define la captura de datos
    def dataCapture(x):
        listValue = []
        i=0
        while i<n:
            getSerialValue = x.readline()
            Value = getSerialValue.replace("\r\n","")
            if (Value!=""):
                listValue.append(Value)
                i = i + 1
        return listValue

    # Estilo para las celdas
    style0 = xlwt.easyxf('font: name Times New Roman, colour black, bold on; align: vert center, horiz
center')
    style1 = xlwt.easyxf('align: vert center, horiz center')
    style2 = xlwt.easyxf('align: vert center, horiz center; pattern: pattern solid, fore_colour yellow')
    style3 = xlwt.easyxf('align: vert center, horiz center; pattern: pattern solid, fore_colour red')
    style4 = xlwt.easyxf('align: vert center, horiz center; pattern: pattern solid, fore_colour green')
    style5 = xlwt.easyxf('align: vert center, horiz center; pattern: pattern solid, fore_colour yellow; font:
name Times New Roman, colour black, bold on')
    style6 = xlwt.easyxf('align: vert center, horiz center; pattern: pattern solid, fore_colour red; font: name
Times New Roman, colour black, bold on')
    style7 = xlwt.easyxf('align: vert center, horiz center; pattern: pattern solid, fore_colour green; font: name
Times New Roman, colour black, bold on')

    # Funcion que define la creacion de la hoja de Excel
    def createWorkBook():
        wb = xlwt.Workbook()
```

```
    return wb

# Funcion que define la creacion de las hojas
def createSheet(b):
    date = str(strftime("%H_%M_%S"))
    ws = b.add_sheet(date,cell_overwrite_ok=True)
    return ws

# Funcion que define el almacenaje de los datos en el archivo de Excel
def savingData(x,y,n,w,l):
    w.write(0, 0, 'Numero de Prueba', x)
    w.write(0, 1, 'Tiempo (seg)', x)
    for i in range(n):
        w.write(i+1, 0, i+1 ,y)
        w.write(i+1, 1, float(l[i]),y)

# Funcion que define las operacione de los datos
def averageMaxMin(l,n,a,b,c,z,w,p,s):
    x = 0
    for i in l:
        x = x + float(i)

    average = x/n
    s.write(n+1,0,'Promedio',a)
    s.write(n+1,1,average,z)

    maxValue = float(max(l))
    s.write(n+2,0,'Tiempo maximo',b)
    s.write(n+2,1,maxValue,w)

    minValue = float(min(l))
    s.write(n+3,0,'Tiempo minimo',c)
    s.write(n+3,1,minValue,p)

# Se mandan a llamar a las funciones
a = serialConection() ## arduino
l = dataCapture(a) ## lista
wb = createWorkBook() ## workbook
ws = createSheet(wb) ## worksheet
savingData(style0,style1,n,ws,l) ## almacenaje
averageMaxMin(l,n,style5,style6,style7,style2,style3,style4,ws) ## promedio, maximo y minimo

# Hoja de excel guardada
excelWorkBook = wb.save(user + '.xls')

# Cerrando puerto serial
a.close()

window = str(raw_input("Desea ingresar un nuevo usuario (Y o N): "))

if(window == 'Y' or window == 'y'):
    newUser = str(raw_input("Nombre del Usuario: "))

    if(newUser == user):
        print "Ya existe ese usuario"
        oldUser = str(raw_input("Desea agregar una nueva corrida (Y o N): "))
```

```
        if(oldUser == 'Y' or oldUser == 'y'):
            open_workbook(user + '.xls',formatting_info=True)
            n = int(input("Ingrese el numero de pruebas: "))
            a = serialConection()
            l = dataCapture(a)
            ws = createSheet(wb)
            savingData(style0,style1,n,ws,l)
            averageMaxMin(l,n,style5,style6,style7,style2,style3,style4,ws)
            excelWorkBook = wb.save(user + '.xls')

        elif(oldUser == 'N' or 'n'):
            sys.exit(0)
    else:
        print "Nuevo usuario"
        n = int(input("Ingrese el numero de pruebas: "))
        a = serialConection()
        l = dataCapture(a)
        wb = createWorkBook()
        ws = createSheet(wb)
        savingData(style0,style1,n,ws,l)
        averageMaxMin(l,n,style5,style6,style7,style2,style3,style4,ws)
        excelWorkBook = wb.save(newUser + '.xls')

elif(window == 'N' or window == 'n'):
    oldUser = str(raw_input("Desea agregar una nueva corrida al archivo ya existente (Y o N): "))

    if(oldUser == 'Y' or oldUser == 'y'):
        open_workbook(user + '.xls',formatting_info=True)
        n = int(input("Ingrese el numero de pruebas: "))
        a = serialConection()
        l = dataCapture(a)
        ws = createSheet(wb)
        savingData(style0,style1,n,ws,l)
        averageMaxMin(l,n,style5,style6,style7,style2,style3,style4,ws)
        excelWorkBook = wb.save(user + '.xls')
        sys.exit(0)

    elif(oldUser == 'N' or 'n'):
        sys.exit(0)
```

9. Arduino sensor RGB

```
/*
* Megaproyecto FitLight Prototipo
* Javier Búcaro 13033
*/

#include <CapacitiveSensor.h>
CapacitiveSensor sensor = CapacitiveSensor(4,3);
int ledPin1 = 12; //R
int ledPin2 = 11; //G
int ledPin3 = 10; //B
int startTime;
```

```
int elapsedTime;
int converter;
int fractional;
int ON_OFF;
long randOn = 0;

void setup(){                  // run once, when the sketch starts
  Serial.begin(9600);
  randomSeed (analogRead (0));    // randomize
  pinMode(ledPin1, OUTPUT);       // sets the digital pin as output
  pinMode(ledPin2, OUTPUT);
  pinMode(ledPin3, OUTPUT);
}

void loop(){                   // run over and over again

  long lectura = sensor.capacitiveSensor(30);

  if(lectura < 1 && ON_OFF == false){

    randOn = random(1000,5000);
    delay(randOn);
    digitalWrite(ledPin1, LOW);
    //digitalWrite(ledPin2, LOW);
    //digitalWrite(ledPin3, LOW);
    startTime = millis();
    ON_OFF = true;

  }

  else if(lectura > 1 && ON_OFF == true){

    elapsedTime = millis() - startTime;
    digitalWrite(ledPin1, HIGH);
    //digitalWrite(ledPin2, HIGH);
    //digitalWrite(ledPin3, HIGH);
    ON_OFF = false;
    converter = secondConverter(elapsedTime);

  }
}

int secondConverter (int x){

      Serial.print((int)(x/1000L));        // divide by 1000 to convert to seconds - then cast to an int to print
      Serial.print(".");                   // print decimal point

      // use modulo operator to get fractional part of time
      fractional = (int)(x % 1000L);

      if (fractional == 0)
        Serial.print("000");     // add three zero's
      else if (fractional < 10)    // if fractional < 10 the 0 is ignored giving a wrong time, so add the zeros
        Serial.print("00");      // add two zeros
      else if (fractional < 100)
        Serial.print("0");       // add one zero
```

```
    Serial.print(fractional);    // print fractional part of time
}
```

10. Módulos de Comunicación

```
#include <SoftwareSerial.h> //Librería Utilizada para comunicación Serial en Pines Digitales.
SoftwareSerial mod(10,11); //RX_Arduino,TX_Arduino
int pot=0; //Potenciometro colocado en el pin A0 del Arduino
void setup(){
  mod.begin(9600); //Se inicia la comunicación Serial
  mod.println("Conectado"); //Permite verificar que la comunicación inicio con éxito.
}
void loop(){
  pot=analogRead(0); //Se lee el valor del potenciómetro.
  mod.println(pot);//Se envía el valor del potenciómetro por el puerto serial en los pines digitales.
  delay(100);
}
```

11. Máquina lanzadora de discos tiro con armas de caza

```
#include <SoftwareSerial.h>

int Activar_cronometro=0;
int temporal=0;
int maquina1=13;
boolean conectar=false;
boolean EstadoAnterior=LOW;
boolean EstadoActual;
char valor =0;
SoftwareSerial Maquina(4,2); //RX TX

void setup() {

  Serial.begin(9600); //Permite el intercambio de información entre ordenador y Arduino
  Maquina.begin(9600); //Permite el intercambio de información entre ambos módulos Arduino

  //Configuración del Pin Utilizado para detectar el lanzamiento de un Disco
  pinMode(maquina1,INPUT);
  digitalWrite(maquina1, LOW);
}

void loop() {

//El dato recibido por el módulo Xbee es enviado por el puerto Serial Alámbrico para ser desplegado
  if(Maquina.available())
  {
    Serial.print(String(Maquina.readString()));
  }

//El dato recibido por el puerto serial, es verificado para determinar que acción desea ejecutar el operador,
posteriormente se
//envía un valor identificador por el módulo Xbee.
  if(Serial.available()>0)
```

```
  {
   {
    valor = Serial.read();
   }
  }
//Selección de Modalidad Fosa
  if(valor=='F')
  {
    Maquina.println(2);
    valor=0;
  }
//Seleccion de Modalidad Skeet
  if(valor=='S')
  {
    Maquina.println(3);
    valor=0;
  }
//Conexión entre ambos módulos
  if(valor=='Z')
  {
    Maquina.println(4);
    valor=0;
    temporal = millis();
    conectar=true;
  }
  if(millis()>temporal+1000 && conectar==true)
  {
    temporal=millis();
    Maquina.println(5);
  }

//Debounce utilizado para evitar que el ruido afecte la activación del cronometro
  int lectura = digitalRead(maquina1);
  if(lectura != EstadoAnterior)
  {
     if(lectura != EstadoActual)
     {
        EstadoActual=lectura;
        if (EstadoActual == HIGH)
        {
          Activar_cronometro = 1;
          Maquina.println(Activar_cronometro);
        }
     }
  }
  EstadoAnterior = lectura;
}
```

12. Detección movimiento de escopetas tiro con armas de caza

```
#include <SoftwareSerial.h>

//Declaracion de variables booleanas
boolean Fosa=false;
```

```
boolean Skeet=false;
boolean Simple=false;
boolean Doble=false;
boolean Coneccion=false;
boolean DX = false;
boolean DY = false;
boolean DZ = false;
boolean DVelX = false;
boolean DVelY = false;
boolean DVelZ = false;
boolean cronReaction_Time = false;
boolean cronShoot_time=false;
boolean Reaction_Display=false;
boolean Display_One=false;
boolean Display_Two=false;

//Declaración de Variables enteras
int LED=13;
int Plato=0;
int temporal=0;
int temporal1=0;
int EstacionSkeet=1;
int X = 0;
int Y = 0;
int Z = 0;
int Xini = 0;
int Yini = 0;
int Zini = 0;
int movimiento = 0;
int minutos = 0;
int segundos = 0;
int decimas = 0;
int centesimas=0;
int datoRecibido=0;

//Delcaración de Variables tipo String para concatenar los tiempos para su envío.
String ind ="";
String Reaccion1="";
String Reaccion2="";
String Reaccion3="";
String Reaccion4="";
String Reaccion5="";
String ReaccionT="";

String Disparo11="";
String Disparo12="";
String Disparo13="";
String Disparo14="";
String Disparo15="";
String Disparo1T="";

String Disparo21="";
String Disparo22="";
String Disparo23="";
String Disparo24="";
String Disparo25="";
```

```
String Disparo2T="";

unsigned long milisegundos = 0;

SoftwareSerial BT(4,2); //RX | TX


void setup()
{
  pinMode(LED, OUTPUT);
  BT.begin(9600);
}


//Método elaborado como cronometro, es capaz de contar minutos, segundos y decimas de segundo.
void cronometro()
{
 milisegundos = millis();
 if (milisegundos % 100 == 0)
 {
  decimas = decimas + 1;
  if (decimas == 10)
  {
   decimas = 0;
   segundos = segundos + 1;
  }
  if (segundos == 60)
  {
   segundos = 0;
   minutos = minutos + 1;
  }
  if (minutos == 60)
  {
   minutos = 0;
  }
 }
}

//Método que lee constantemente los valores en los ejes X,Y,Z del acelerometro.
void leerDatos()
{
 X = analogRead(A0);
 Y = analogRead(A1);
 Z = analogRead(A2);

 X = map(X,0,1023,0,4500);
 Y = map (Y,0,1023,0,4500);
 Z = map (Z,0,1023,0,4500);
}


//Método reinicia la detección de un disparo.
void Reset_Shoot_Time()
{
    DX = false;
    DY = false;
```

```
    DZ = false;
}
//Método que reinicia la detección del tiempo de reacción
void Reset_Reaction_Time()
{
    cronReaction_Time = false;
    Reset_Shoot_Time();
}
//Método para actulizar los valores de los ejes del acelerometro
void Actualizar_Valores()
{
    Xini =  X;
    Yini = Y;
    Zini = Z;
}
//Metodo que reinicia el coronometro
void Reset_Cron()
{
    digitalWrite(LED,LOW);
    milisegundos = 0;
    decimas = 0;
    segundos = 0;
    minutos = 0;
    cronShoot_time = false;
    movimiento=0;
    BT.println(" ");
}

//Metodo para imprimir los valores del Cronometro
//Los datos son concatenados en un solo String para no tener problemas con su envio.
//A cada dato se le agrega un caracter identificador que permite establecer que dato se esta enviando.
void display_timer()
{
 String sep=":";
 if(Reaction_Display==true)
 {
  ind = "R";
  Reaccion1=minutos+ind;
  Reaccion2=segundos+ind;
  Reaccion3=decimas+ind;
  Reaccion4=((milisegundos%10000)%1000/100);
  Reaccion5=(((milisegundos%10000)%1000)%100/10);
  ReaccionT=Reaccion1+Reaccion2+Reaccion3+Reaccion4+Reaccion5;
  Reaction_Display=false;
 }
 if(Display_One==true)
 {
  ind = "U" ;
  Disparo11=minutos+ind;
  Disparo12=segundos+ind;
  Disparo13=decimas+ind;
  Disparo14=((milisegundos%10000)%1000/100);
  Disparo15=(((milisegundos%10000)%1000)%100/10);
  Disparo1T=Disparo11+Disparo12+Disparo13+Disparo14+Disparo15;
  Display_One=false;
 }
```

```
 if(Display_Two==true)
 {

  ind = "X";
  Disparo21=minutos+ind;
  Disparo22=segundos+ind;
  Disparo23=decimas+ind;
  Disparo24=((milisegundos%10000)%1000/100);
  Disparo25=(((milisegundos%10000)%1000)%100/10);
  Disparo2T=Disparo21+Disparo22+Disparo23+Disparo24+Disparo25;
  Display_Two=false;
 }
 ind = "";
}


//Método que permiete la obtención del tiempo de reacción. Al existir una variacion detectada por el acelerometro,
//se obtiene el tiempo actual del cronometro y se almacena cmo tiempo de reacción.
void reaction_time()
{
   if (X > Xini + 100 || X < Xini - 100)
   {
    DX = true;
   }
   if (Y > Yini + 100 || Y < Yini - 100)
   {
    DY = true;
   }
   if (Z > Zini + 100 || Z < Zini - 100)
   {
    DZ = true;
   }
   if (DX == true || DY == true || DZ == true)
   {
    Reaction_Display=true;
    display_timer();
    movimiento=1;
    leerDatos();      //Se lee el valor actual del acelerometro
    Actualizar_Valores(); //Se establece el valor leido como nuevo marco de referencia
    Reset_Reaction_Time();
    }
}

//Método que permiete la obtención del tiempo del primer disparo. Al existir una variacion detectada por el acelerometro,
//se obtiene el tiempo actual del cronometro y se almacena cmo tiempo de disparo 1.
void shoot_time1()
{
   if (X > Xini + 800 || X < Xini - 800)
   {
    DX = true;
   }
   if (Y > Yini + 800 || Y < Yini - 800)
   {
    DY = true;
```

```
    }
    if (Z > Zini + 800 || Z < Zini - 800)
    {
      DZ = true;
    }
    if ((DX == true || DY == true || DZ == true))
    {
      Display_One=true;
      display_timer();
      temporal=segundos;
      movimiento=2;
      leerDatos();
      Actualizar_Valores();
      Reset_Shoot_Time();
    }
}

//Método que permiete la obtención del tiempo del segundo disparo. Al existir una variacion detectada por el acelerometro,
//se obtiene el tiempo actual del cronometro y se almacena cmo tiempo de disparo 2.
void shoot_time2()
{
    if (X > Xini + 2000|| X < Xini - 2000)
    {
      DX = true;
    }
    if (Y > Yini + 2000 || Y < Yini - 2000)
    {
      DY = true;
    }
    if (Z > Zini + 2000 || Z < Zini - 2000)
    {
      DZ = true;
    }
    if ((DX == true || DY == true || DZ == true))
    {
      Display_Two=true;
      display_timer();
      temporal1=segundos;
      movimiento=3;
      Reset_Shoot_Time();
    }
}

//Método que permite la obtención de tiempo de reacción y disparo de los lanzamientos Simples en la modalidad de Skeet.
void Skeet_Simple()
{
  if (cronReaction_Time == true)
  {
    leerDatos();
    cronometro();
    reaction_time();
  }
  if(cronShoot_time == true && movimiento==1)
  {
```

```
    leerDatos();
    cronometro();
    shoot_time1();
    if (movimiento==2)
    {
      EstacionSkeet=EstacionSkeet+1;
      Reset_Cron();
      movimiento=0;
      Plato=1;
      BT.println(ReaccionT);
      delay(100);
      BT.println("\n");
      delay(10);
      BT.println("\n");
      delay(10);
      BT.println("\n");
      delay(30);
      BT.println(Disparo1T);

    }
  }
}

//Método que permite la obtención de tiempo de reacción y disparo de los lanzamientos Dobles en la modalidad de Skeet.
void Skeet_Doble()
{
   if (cronReaction_Time == true)
  {
    leerDatos();
    cronometro();
    reaction_time();
  }
  if(cronShoot_time == true && movimiento==1)
  {
    leerDatos();
    cronometro();
    shoot_time1();
 }
  if(cronShoot_time == true && movimiento==2)
  {
    leerDatos();
    cronometro();
    shoot_time2();
    if (movimiento==3)
    {
      EstacionSkeet=EstacionSkeet+1;
      movimiento=0;
      BT.println(ReaccionT);
      delay(100);
      BT.println("\n");
      BT.println("\n");
      BT.println("\n");
      delay(30);
      BT.println(Disparo1T);
      delay(100);
```

```
      BT.println("\n");
      delay(10);
      BT.println("\n");
      delay(10);
      BT.println("\n");
      delay(30);
      BT.println(Disparo2T);
      Reset_Cron();
      Plato=3;

    }
  }
}



void loop()
{
   leerDatos();
   if(BT.available())
   {
     datoRecibido=BT.read();
   }

   //Condicion Para iniciar el cronometro
   if(datoRecibido==49)
    {
     cronReaction_Time = true;
     cronShoot_time = true;
     BT.println("A\n");
     digitalWrite(LED,HIGH);
     Actualizar_Valores();
     Simple=true;
     Doble=true;
     datoRecibido=0;
    }

    //Condicion Para Activar la modalidad Fosa
    if (datoRecibido==50)
    {
     Fosa=true;
     BT.println("FOSA") ;
     datoRecibido=0;
     Skeet=false;
    }

    //Condicion para activar la modalidad Skeet
    if(datoRecibido==51)
    {
     Skeet=true;
     BT.println("SKEET");
     datoRecibido=0;
     Fosa=false;
     }

    if(datoRecibido==52)
```

```
    {
      BT.println("M");
      datoRecibido=0;
    }



   //Condición utilizad para la modalidad Skeet
    if(Skeet==true)
    {
      if (EstacionSkeet==1 || EstacionSkeet==3 || EstacionSkeet==5 || EstacionSkeet==7 || EstacionSkeet ==8
|| EstacionSkeet ==9 || EstacionSkeet ==11|| EstacionSkeet ==16 || EstacionSkeet ==17)
      {
        Skeet_Simple();
      }
      if(EstacionSkeet==2 || EstacionSkeet==4 || EstacionSkeet==6 || EstacionSkeet==10 ||
EstacionSkeet==12  || EstacionSkeet==13 || EstacionSkeet==14 ||EstacionSkeet==15)
      {
        Skeet_Doble();
      }

      if(EstacionSkeet>17)
      {
        EstacionSkeet=1;
      }
    }

   //Condicion para la modalidad Fosa.
   if(Fosa==true)
   {
     //Se obtiene el tiempo de reacción
     if (cronReaction_Time == true)
     {
       leerDatos();
       cronometro();
       reaction_time();
     }
     //Se obtiene el tiempo del primer disparo
     if(cronShoot_time == true && movimiento==1)
     {
       leerDatos();
       cronometro();
       shoot_time1();
       cronometro();
     }
    //Se obtiene el tiempo del segundo disparo
    if(cronShoot_time==true && movimiento==2)
    {
        leerDatos();
        cronometro();
        shoot_time2();
        if (movimiento==3)
        {
          //Condicional para verificar si el tiempo obtenido del segundo disparo es una dato valido o no
          //Si este no supera por más de un segundo al del primer disparo es valido, de lo contrario este valor
se descarta.
```

```
          if((temporal1-1==temporal || temporal1==temporal))
          {
            BT.println(ReaccionT);
            delay(100);
            BT.println("\n");
            BT.println("\n");
            BT.println("\n");
            delay(30);
            BT.println(Disparo1T);
            delay(100);
            BT.println("\n");
            delay(10);
            BT.println("\n");
            delay(10);
            BT.println("\n");
            delay(30);
            BT.println(Disparo2T);
            Reset_Cron();
          }
          else
          {
            delay(30);
            BT.println(ReaccionT);
            delay(100);
            BT.println("\n");
            delay(10);
            BT.println("\n");
            delay(10);
            BT.println("\n");
            delay(30);
            BT.println(Disparo1T);
            Reset_Cron();
          }
        }
    }
  }
}
```

13. Interfaz gráfica sensor tiro con armas de caza

```
import jssc.*;
import java.awt.*;
import java.io.*;
import java.util.*;
import java.util.logging.*;
import javax.swing.JOptionPane;
//Librerias para establecer la comunicación con Arduino
import com.panamahitek.ArduinoException;
import com.panamahitek.PanamaHitek_Arduino;
//Liberria para Generar el Archivo de Excel
import org.apache.poi.hssf.usermodel.*;

public class GUI extends javax.swing.JFrame {

    //Objeto que permite establecer comunicación con Arduino
    PanamaHitek_Arduino arduino = new PanamaHitek_Arduino();
```

```
//Variables utilizadas para almacenar los resultados y desplegarlos en pantalla
public static String dato = "";
public static String reaccion = "";
public static String disparo1 = "";
public static String disparo2 = "";
public static String lanzados = "";
public static String acertados="";
public static String realizados="";
public static String nombre = "";
//Variable para establecer el nombre a las pestañas de Excel
public static Date fecha = new Date();
//Variables para determinar la modalidad en la que se estan obteniendo datos.
public boolean fosa=false;
public boolean skeet=false;
//Variabales para llevar control de resultados.
public int aciertos=0;
public int Dlanzados=0;
public int Drealizados=0;
public int ESkeet=0;
public int EFosa=0;

public static int i = 0;
public static int m =0;
public static int p=0;
public static int j=0;
//Variables para inicializar el archivo de Excel y almacenar los datos.
public HSSFWorkbook libro;
public HSSFSheet hoja;
public HSSFRow fila;
public HSSFCell celda;
public FileOutputStream elFichero;
public String data[] = new String[1000];
public String posicion[] = new String[1000];
SerialPortEventListener listener;

public GUI() {

   this.listener = (SerialPortEvent spe) -> {
      try {
         //Se revisa si existe algún mensaje serial en Arduino
         if (arduino.isMessageAvailable()) {
            //Se imprime el mensaje recibido en la consola
            dato = arduino.printMessage();
            System.out.println(dato);
            //El valor M establece que existe conección entre ambos módulos
            if (dato.equals("M")) {
               Conectar.setBackground(Color.green);
               Skeet.setBackground(Color.red);
               Fosa.setBackground(Color.red);
            }
            //Valor de confirmación que el otro micrcontrolador opera en modalidad Fosa.
            if (dato.equals("FOSA")) {
               Fosa.setBackground(Color.green);
               Skeet.setBackground(Color.red);
               fosa=true;
               skeet=false;
```

```
}
//Valor de confirmación que el otro micrcontrolador opera en modalidad Skeet
if (dato.equals("SKEET")) {
   Skeet.setBackground(Color.green);
   Fosa.setBackground(Color.red);
   fosa=false;
   skeet=true;
}
//Valor de confirmación que se realizó el lanzamiento de 1 o 2 discos
if (dato.equals("A")) {
   reaccion = "";
   disparo1 = "";
   disparo2 = "";
   Display_Reaccion.setText(reaccion);
   Display_Disparo1.setText(disparo1);
   Display_Disparo2.setText(disparo2);
   if(fosa==true)
   {
      posicion[p]=Integer.toString(EFosa);
   }
   if(skeet==true)
   {
      posicion[p]=Integer.toString(ESkeet);
   }
   p=p+1;

}
//Valor del tiempo de reacción se susituye el caracter identificador por ":"
if (dato.contains("R")) {
   String temp = dato.replace("R", ":");
   reaccion = reaccion + temp;
   if(reaccion.startsWith("0"))
   {
      Display_Reaccion.setText(reaccion);
   }
   else
   {
      reaccion="0"+reaccion;
      Display_Reaccion.setText(reaccion);
   }
   //Si se recibe todo el valor el dato se agrega a la matriz de resultados.
   if (reaccion.length()>7)
   {
      data[i]=reaccion;
      i=i+1;
      //Mensaje de confirmación que se agrego el dato a la matriz.
      System.out.println("Dato Agregado");
   }
}

//Valor de tiempo de disparo 1 se sustituye le caracter identificador por ":"
if (dato.contains("U")) {
   String temp2 = dato.replace("U", ":");
   disparo1 = disparo1 + temp2;
   if(disparo1.startsWith("0"))
   {
```

```
                Display_Disparo1.setText(disparo1);
            }
            else
            {
                disparo1="0"+disparo1;
                Display_Disparo1.setText(disparo1);
            }
            //Si se recibe todo el valor el dato se agrega a la matriz de resultados.
           if(disparo1.length()>7)
            {
                data[i]=disparo1;
                //Se corre la matriz dos posiciones ya que no siempre se realizan dos disparos.
                //Si no se realiza el segundo disparo, en el documento de excel la celda queda vacia.
                i=i+2;
                System.out.println("Dato Agregado2");
                //Condicionales para llevar el control de discos lanzados y disparos realizados.
                 if(fosa==true)
                {
                    Dlanzados=Dlanzados+1;
                    Drealizados=Drealizados+1;
                }
                if(skeet==true)
                {
                    Dlanzados=Dlanzados+1;
                    Drealizados=Drealizados+1;
                }
            }

        }
        //Valor de tiempo de disparo 1 se sustituye le caracter identificador por ":"
        if (dato.contains("X")) {
            String temp3 = dato.replace("X", ":");
            disparo2 = disparo2 + temp3;
            if(disparo2.startsWith("0"))
            {
                Display_Disparo2.setText(disparo2);
            }
            else
            {
                disparo2="0"+disparo2;
                Display_Disparo2.setText(disparo2);
            }
            if(disparo2.length()>7)
            {
                //Se retorna una posición de la matriz para almacenar el segundo disparo.
                //Los datos ocupan tres posiciones en la matriz sin importar si se realiza o no el segundo
disparo.
                i=i-1;
                data[i]=disparo2;
                i=i+1;
                System.out.println("Dato Agregado3");
                if(skeet==true)
                {
                    Dlanzados=Dlanzados+1;
                    Drealizados=Drealizados+1;
                }
```

```
                }

            }

        }

    }
    //Mensaje para desplegar que no se logro establecer comunciación con el Arduino
    catch (SerialPortException | ArduinoException ex) {
        Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);
        JOptionPane.showMessageDialog(null, "No se logró iniciar la comunicación");

    }
};

try {

    arduino.arduinoRXTX("COM4", 9600, listener);

}
//Mensaje para desplegar que no se logro establecer comunciación con el Arduino
    catch (ArduinoException ex) {
    Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);
    JOptionPane.showMessageDialog(null, "No se logró iniciar la comunicación revise que el cable se
encuentre conectado.");
}

initComponents();

}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    Fosa = new javax.swing.JButton();
    Skeet = new javax.swing.JButton();
    Acierto = new javax.swing.JButton();
    Resultados = new javax.swing.JButton();
    Conectar = new javax.swing.JButton();
    LabelTiempoR = new javax.swing.JLabel();
    LabelTiempoD2 = new javax.swing.JLabel();
    LabelTiempoD1 = new javax.swing.JLabel();
    jScrollPane1 = new javax.swing.JScrollPane();
    Display_Reaccion = new javax.swing.JTextPane();
    jScrollPane5 = new javax.swing.JScrollPane();
    Display_Disparo1 = new javax.swing.JTextPane();
    jScrollPane6 = new javax.swing.JScrollPane();
    Display_Disparo2 = new javax.swing.JTextPane();
    LabelRealizados = new javax.swing.JLabel();
    LabelAcertados = new javax.swing.JLabel();
```

```
LabelLanzados = new javax.swing.JLabel();
jScrollPane2 = new javax.swing.JScrollPane();
Disparos_Realizados = new javax.swing.JTextPane();
jScrollPane3 = new javax.swing.JScrollPane();
Discos_Acertados = new javax.swing.JTextPane();
jScrollPane4 = new javax.swing.JScrollPane();
Discos_Lanzados = new javax.swing.JTextPane();
Nombre = new javax.swing.JTextField();
LabelNameAtleta = new javax.swing.JLabel();
Excel = new javax.swing.JButton();
jMenuBar1 = new javax.swing.JMenuBar();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("CDAG TIRO CON ARMAS DE CAZA");
setAutoRequestFocus(false);
setBackground(new java.awt.Color(29, 218, 213));
setFocusCycleRoot(false);
setFocusTraversalPolicyProvider(true);
setMaximumSize(new java.awt.Dimension(800, 450));
setMinimumSize(new java.awt.Dimension(800, 450));

Fosa.setText("Fosa");
Fosa.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        FosaActionPerformed(evt);
    }
});

Skeet.setText("Skeet");
Skeet.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        SkeetActionPerformed(evt);
    }
});

Acierto.setText("Acierto");
Acierto.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        AciertoActionPerformed(evt);
    }
});

Resultados.setText("Resultados");
Resultados.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        ResultadosActionPerformed(evt);
    }
});

Conectar.setText("Conectar");
Conectar.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        ConectarActionPerformed(evt);
    }
});
```

```
LabelTiempoR.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
LabelTiempoR.setText("TIEMPO DE REACCION");

LabelTiempoD2.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
LabelTiempoD2.setText("TIEMPO DE DISPARO 2");

LabelTiempoD1.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
LabelTiempoD1.setText("TIEMPO DE DISPARO 1");

Display_Reaccion.setBackground(new java.awt.Color(204, 204, 204));
Display_Reaccion.setBorder(new javax.swing.border.MatteBorder(null));
Display_Reaccion.setFont(new java.awt.Font("Times New Roman", 1, 24)); // NOI18N
Display_Reaccion.setAlignmentX(2.5F);
Display_Reaccion.setAlignmentY(2.5F);
jScrollPane1.setViewportView(Display_Reaccion);

Display_Disparo1.setBackground(new java.awt.Color(204, 204, 204));
Display_Disparo1.setBorder(new javax.swing.border.MatteBorder(null));
Display_Disparo1.setFont(new java.awt.Font("Times New Roman", 1, 24)); // NOI18N
jScrollPane5.setViewportView(Display_Disparo1);

Display_Disparo2.setBackground(new java.awt.Color(204, 204, 204));
Display_Disparo2.setBorder(new javax.swing.border.MatteBorder(null));
Display_Disparo2.setFont(new java.awt.Font("Times New Roman", 1, 24)); // NOI18N
jScrollPane6.setViewportView(Display_Disparo2);

LabelRealizados.setText("Disparos Realizados");

LabelAcertados.setText("Discos Acertados");

LabelLanzados.setText("Discos Lanzados");

jScrollPane2.setViewportView(Disparos_Realizados);

jScrollPane3.setViewportView(Discos_Acertados);

jScrollPane4.setViewportView(Discos_Lanzados);

LabelNameAtleta.setText("Nombre de Atleta");

Excel.setText("Enviar Datos");
Excel.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        ExcelActionPerformed(evt);
    }
});
setJMenuBar(jMenuBar1);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
            .addGroup(layout.createSequentialGroup()
                .addGap(68, 68, 68)
```

```
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 164,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(78, 78, 78)
                .addComponent(jScrollPane5, javax.swing.GroupLayout.PREFERRED_SIZE, 164,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(52, 52, 52)
                .addComponent(jScrollPane6, javax.swing.GroupLayout.PREFERRED_SIZE, 164,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGroup(layout.createSequentialGroup()
                .addGap(68, 68, 68)
                .addComponent(LabelTiempoR)
                .addGap(80, 80, 80)
                .addComponent(LabelTiempoD1)
                .addGap(53, 53, 53)
                .addComponent(LabelTiempoD2))
            .addGroup(layout.createSequentialGroup()
                .addGap(140, 140, 140)
                .addComponent(LabelLanzados)
                .addGap(30, 30, 30)
                .addComponent(jScrollPane4, javax.swing.GroupLayout.PREFERRED_SIZE, 70,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGroup(layout.createSequentialGroup()
                .addGap(41, 41, 41)
                .addComponent(Fosa)
                .addGap(10, 10, 10)
                .addComponent(Skeet)
                .addGap(10, 10, 10)
                .addComponent(Acierto)
                .addGap(10, 10, 10)
                .addComponent(Resultados)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(Conectar)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(Excel)
                            .addComponent(LabelNameAtleta))
                        .addGap(9, 9, 9))
                    .addComponent(Nombre, javax.swing.GroupLayout.PREFERRED_SIZE, 137,
javax.swing.GroupLayout.PREFERRED_SIZE))))
            .addContainerGap(35, Short.MAX_VALUE))
        .addGroup(layout.createSequentialGroup()
            .addGap(140, 140, 140)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(LabelAcertados)
                    .addGap(26, 26, 26)
                    .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED_SIZE, 70,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(82, 407, Short.MAX_VALUE))
                .addGroup(layout.createSequentialGroup()
                    .addComponent(LabelRealizados)
                    .addGap(13, 13, 13)
```

```
                .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 70,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(90, 407, Short.MAX_VALUE))))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(31, 31, 31)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(Fosa)
                    .addComponent(Skeet)
                    .addComponent(Acierto)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(Resultados)
                        .addComponent(Conectar)
                        .addComponent(Nombre, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(6, 6, 6)
                .addComponent(LabelNameAtleta)
                .addGap(5, 5, 5)
                .addComponent(Excel)
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 47,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jScrollPane5, javax.swing.GroupLayout.PREFERRED_SIZE, 47,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jScrollPane6, javax.swing.GroupLayout.PREFERRED_SIZE, 47,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(LabelTiempoR)
                    .addComponent(LabelTiempoD1)
                    .addComponent(LabelTiempoD2))
                .addGap(38, 38, 38)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(LabelRealizados)
                    .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(LabelAcertados)
                    .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(LabelLanzados)
                    .addComponent(jScrollPane4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
        );

        pack();
    }// </editor-fold>
//A cada botón se le agrega un identificador que se le envia al otro microcontrolador Cuando este es
presionado.
    private void FosaActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        try {
            arduino.sendData("F");
        } catch (ArduinoException | SerialPortException ex) {
            Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    private void ConectarActionPerformed(java.awt.event.ActionEvent evt) {

        try {
            arduino.sendData("Z");
        } catch (ArduinoException | SerialPortException ex) {
            Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    private void SkeetActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            arduino.sendData("S");
        } catch (ArduinoException | SerialPortException ex) {
            Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    private void AciertoActionPerformed(java.awt.event.ActionEvent evt) {
            aciertos=aciertos+1;

    }
//Método para Desplegar los resultados en la GUI.
    private void ResultadosActionPerformed(java.awt.event.ActionEvent evt) {
            Disparos_Realizados.setText(Integer.toString(Drealizados));
            Discos_Lanzados.setText(Integer.toString(Dlanzados));
            Discos_Acertados.setText(Integer.toString(aciertos));

    }

 //Método que crea el documento de Excel
    private void ExcelActionPerformed(java.awt.event.ActionEvent evt) {
        // Se obtiene el nombre con el que se llamará o se llama el excel.
        nombre=Nombre.getText();
        String archivo = nombre+".xls";
        File fichero = new File ("C:\\Users\\Alejo\\Documents\\10mo
Semestre\\Megaproyecto\\Códigos\\Interfaz Java\\Interfaz\\"+archivo);
        //Se verifica si exsite el archivo.
        //Si este existe se abre y se crea una nueva pestaña.
        //Si no existe se crea un archivo nuevo.
        if(fichero.exists())
        {
            JOptionPane.showMessageDialog(null, "Abriendo Archivo Existente");
            try {
                String sheet_name = fecha.toString();
                //Se divide la fecha ya que solo interesa el día, mes y la hora.
                sheet_name=sheet_name.substring(0, 19);
                sheet_name=sheet_name.replace(':', ' ');
                FileInputStream file = new FileInputStream(new File("C:\\Users\\Alejo\\Documents\\10mo
Semestre\\Megaproyecto\\Códigos\\Interfaz Java\\Interfaz\\"+archivo));
```

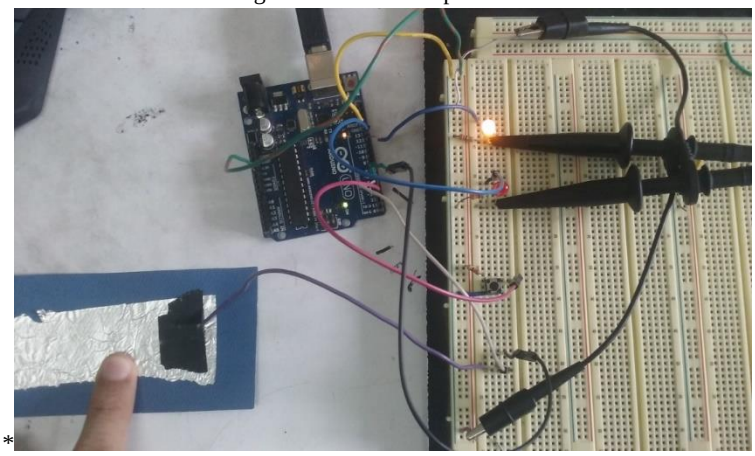
```
            libro = new HSSFWorkbook(file);
            //Se crea una nueva hoja en el archivo existente
            hoja = libro.createSheet(sheet_name);
            //Se le colocan los títulos a las celdas
            fila = hoja.createRow(0);
            celda = fila.createCell(0);
            celda.setCellValue("Tiempo de Reaccion");
            celda = fila.createCell(1);
            celda.setCellValue("Tiempo de Disparo1");
            celda = fila.createCell(2);
            celda.setCellValue("Tiempo de Disparo2");
            celda = fila.createCell(3);
            celda = fila.createCell(4);
            celda = fila.createCell(5);
            celda.setCellValue("Platos Acertados");
            celda = fila.createCell(6);
            celda.setCellValue(aciertos);
            celda = fila.createCell(7);
            celda.setCellValue("Platos Lanzados");
            celda = fila.createCell(8);
            celda.setCellValue(Dlanzados);
            celda = fila.createCell(9);
            celda.setCellValue("Disparos Realizados");
            celda = fila.createCell(10);
            celda.setCellValue(Drealizados);

            //Se recorre la matriz de resultados y se copian los valores en cada una de las celdas
correspondientes.
            for (int x = 1; x<=i;x++)
            {
               fila = hoja.createRow(x);
               for(int j=0;j<=2;j++)
               {
                  celda = fila.createCell(j);
                  celda.setCellValue(data[m]);
                  m=m+1;
               }
            }

            //Se cierra el archivo
            file.close();
            FileOutputStream new_file = new FileOutputStream(new
File("C:\\Users\\Alejo\\Documents\\10mo Semestre\\Megaproyecto\\Códigos\\Interfaz
Java\\Interfaz\\"+archivo));
            libro.write(new_file);
            new_file.close();
            //Se le indica al usuario que la operación fue exitosa.
            JOptionPane.showMessageDialog(null, "Archivo Existente Modificado");
         } catch (FileNotFoundException ex) {
            Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(null, "No se puedo realizar la operación, revise que el archivo
este cerrado");
         } catch (IOException ex) {
            Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);
            JOptionPane.showMessageDialog(null, "No se puedo realizar la operación, revise que el archivo
este cerrado");
```

```
        }


    }
    else
    {
        String sheet_name = fecha.toString();
        sheet_name=sheet_name.substring(0, 19);
        sheet_name=sheet_name.replace(':', ' ');
        libro = new HSSFWorkbook();
        hoja = libro.createSheet(sheet_name);
        fila = hoja.createRow(0);
        celda = fila.createCell(0);
            celda.setCellValue("Tiempo de Reaccion");
            celda = fila.createCell(1);
            celda.setCellValue("Tiempo de Disparo1");
            celda = fila.createCell(2);
            celda.setCellValue("Tiempo de Disparo2");
            celda = fila.createCell(3);
            celda = fila.createCell(4);
            celda = fila.createCell(5);
            celda.setCellValue("Platos Acertados");
            celda = fila.createCell(6);
            celda.setCellValue(aciertos);
            celda = fila.createCell(7);
            celda.setCellValue("Platos Lanzados");
            celda = fila.createCell(8);
            celda.setCellValue(Dlanzados);
            celda = fila.createCell(9);
            celda.setCellValue("Disparos Realizados");
            celda = fila.createCell(10);
            celda.setCellValue(Drealizados);

             for (int x = 1; x<=i;x++)
            {
                fila = hoja.createRow(x);
                for(int j=0;j<=2;j++)
                {
                    celda = fila.createCell(j);
                    celda.setCellValue(data[m]);
                    m=m+1;
                }

            }
        try {
            try (FileOutputStream Fichero = new FileOutputStream(nombre+".xls")) {
                libro.write(Fichero); //Se general el fichero
                JOptionPane.showMessageDialog(null, "Generando Nuevo Archivo Excel");
                //Se cierra el archivo
            } //Se general el fichero
        JOptionPane.showMessageDialog(null, "Nuevo Archivo Generado");
    } catch (HeadlessException | IOException e) {
    }
    }
}
```

```
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {

        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        //</editor-fold>
        java.awt.EventQueue.invokeLater(() -> {
            new GUI().setVisible(true);
        });

    }


    // Variables declaration - do not modify
    private static javax.swing.JButton Acierto;
    private static javax.swing.JButton Conectar;
    private javax.swing.JTextPane Discos_Acertados;
    private javax.swing.JTextPane Discos_Lanzados;
    private javax.swing.JTextPane Disparos_Realizados;
    private javax.swing.JTextPane Display_Disparo1;
    private javax.swing.JTextPane Display_Disparo2;
    private javax.swing.JTextPane Display_Reaccion;
    private javax.swing.JButton Excel;
    private static javax.swing.JButton Fosa;
    private javax.swing.JLabel LabelAcertados;
    private javax.swing.JLabel LabelLanzados;
    private javax.swing.JLabel LabelNameAtleta;
    private javax.swing.JLabel LabelRealizados;
    private javax.swing.JLabel LabelTiempoD1;
    private javax.swing.JLabel LabelTiempoD2;
    private javax.swing.JLabel LabelTiempoR;
    private javax.swing.JTextField Nombre;
    private static javax.swing.JButton Resultados;
    private static javax.swing.JButton Skeet;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
    private javax.swing.JScrollPane jScrollPane4;
    private javax.swing.JScrollPane jScrollPane5;
    private javax.swing.JScrollPane jScrollPane6;
    // End of variables declaration

}
```

C. Circuito sensor RGB

Figura 157. Sensor capacitivo



*

Figura 158. Pushbutton

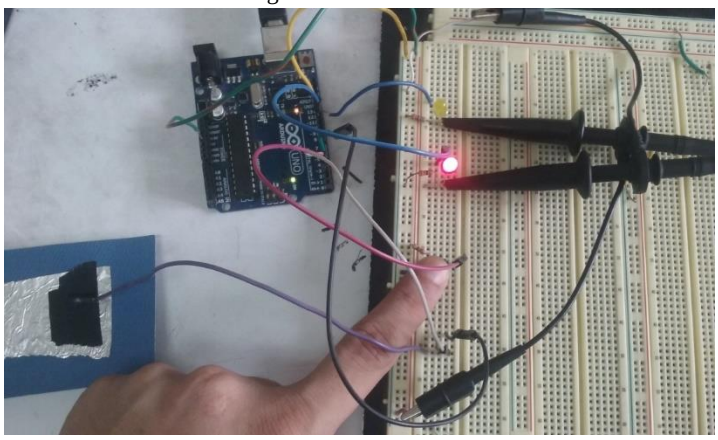
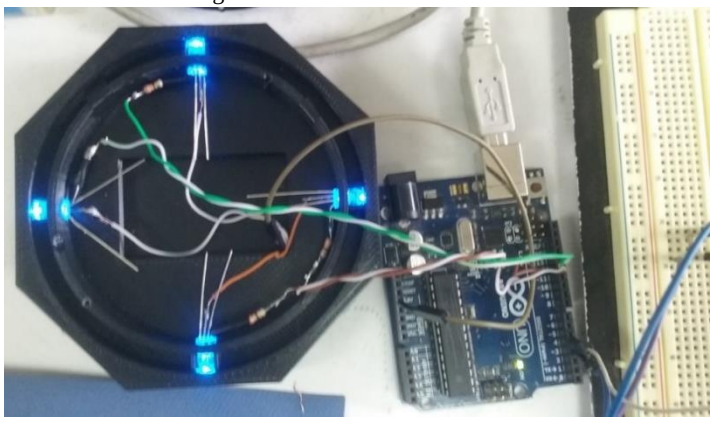


Figura 159. Prueba de LEDs RGB



D. Diagramas de flujo sensor tiro con armas de caza

1. Arduino Uno

Figura 160. Máquina lanzadora de discos parte 1.

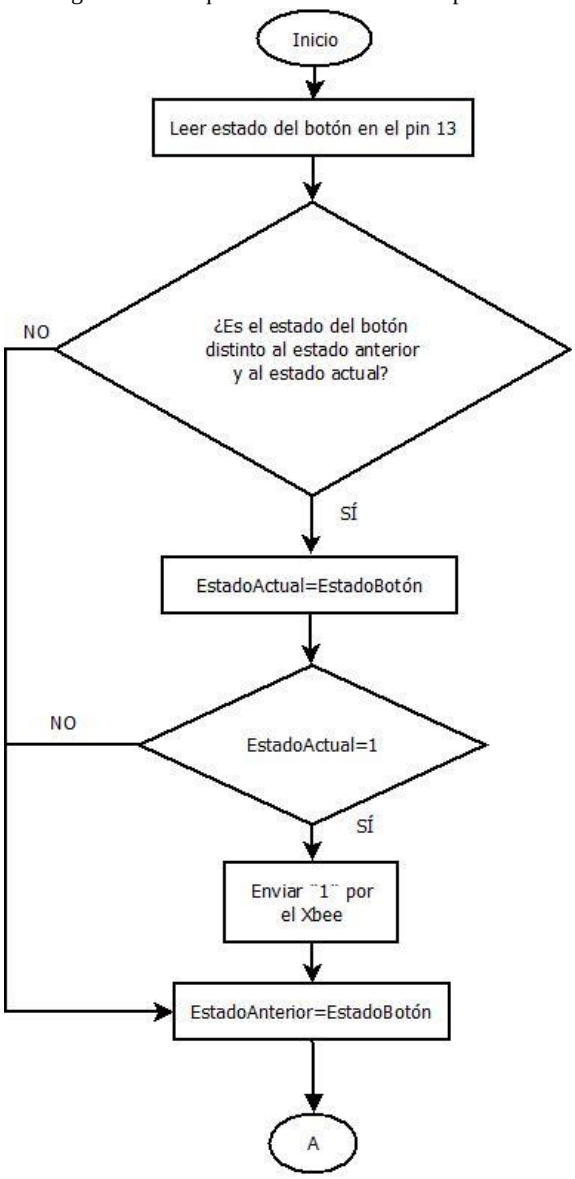
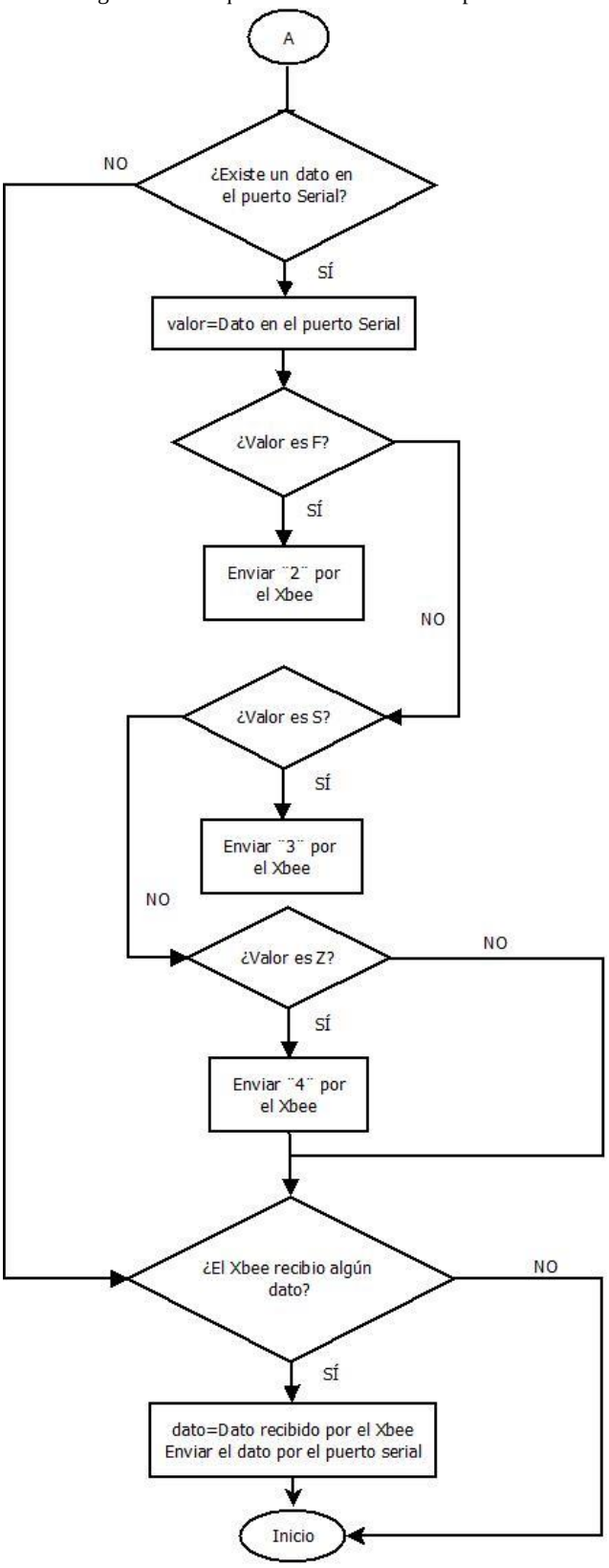


Figura 161. Máquina lanzadora de discos parte 2.



2. Arduino Nano

Figura 162. Selector de modalidad

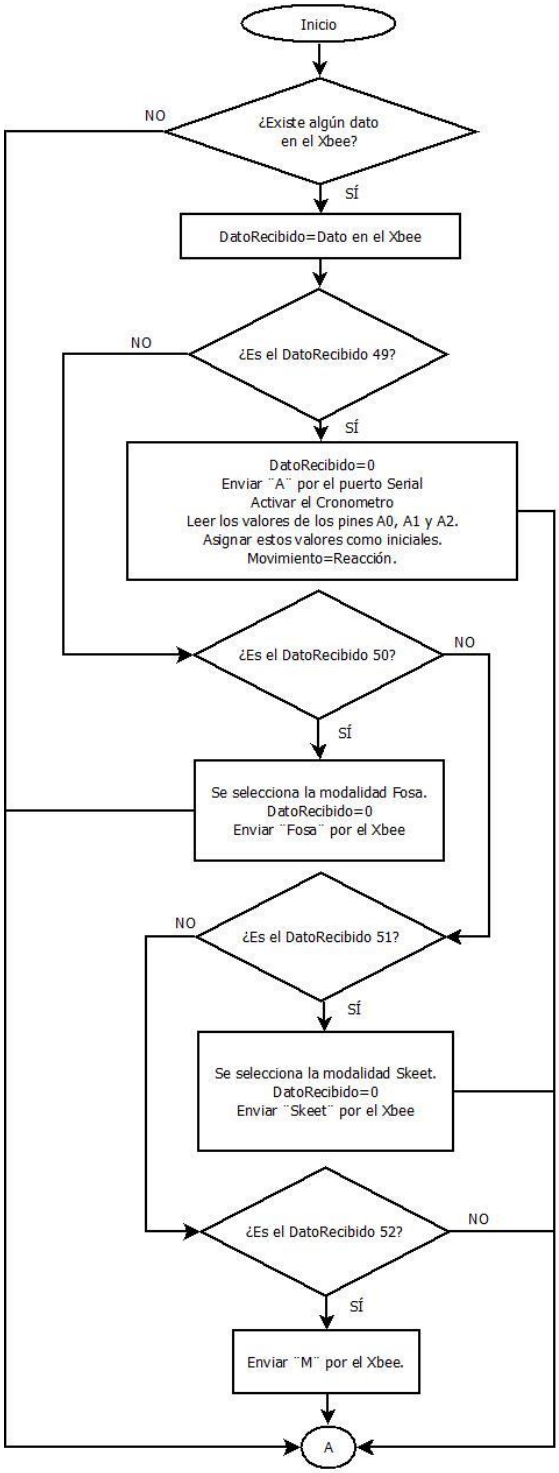


Figura 163. Modalidad skeet lanzamiento simple.

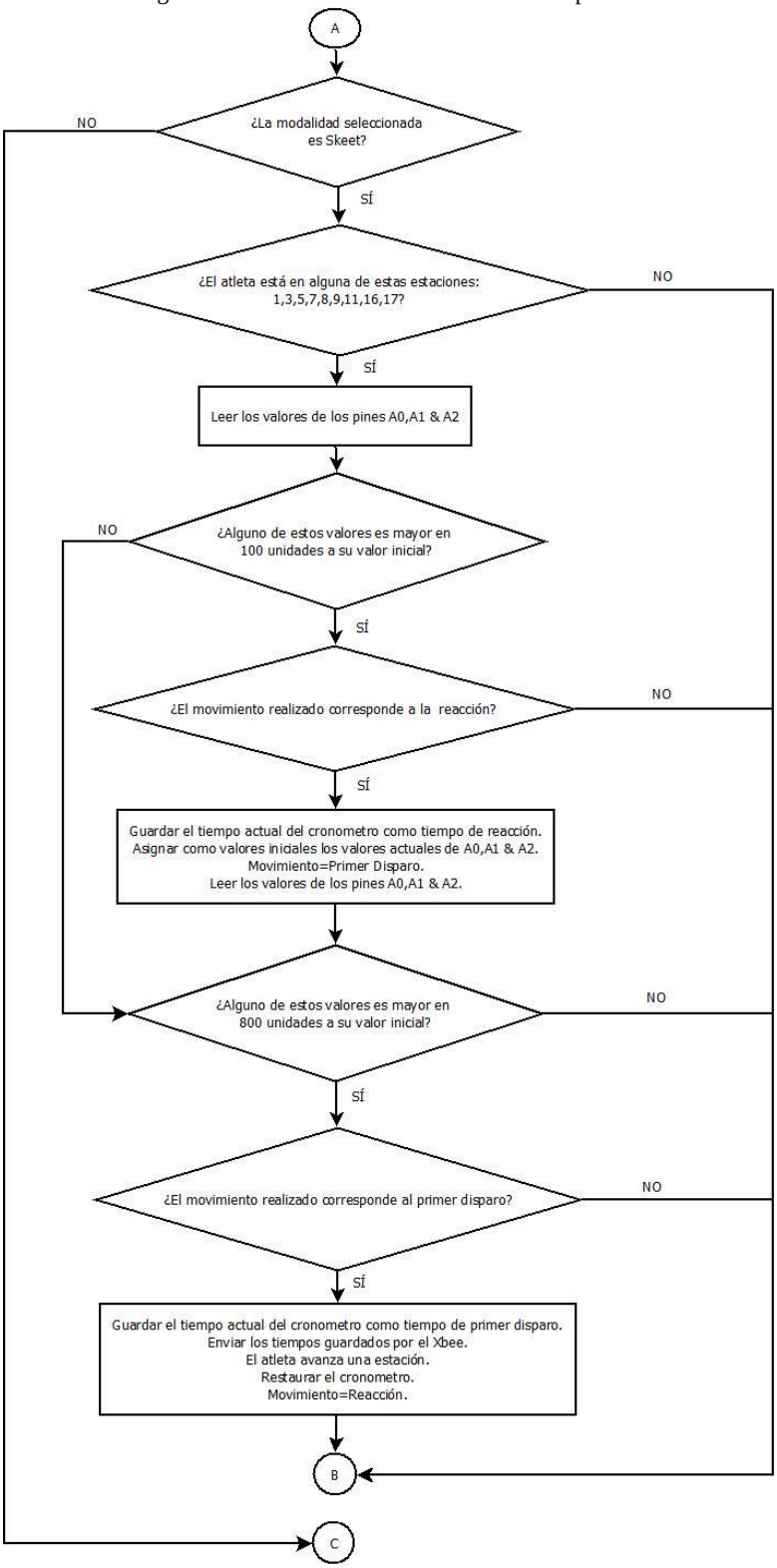


Figura 164. Modalidad skeet lanzamiento doble.

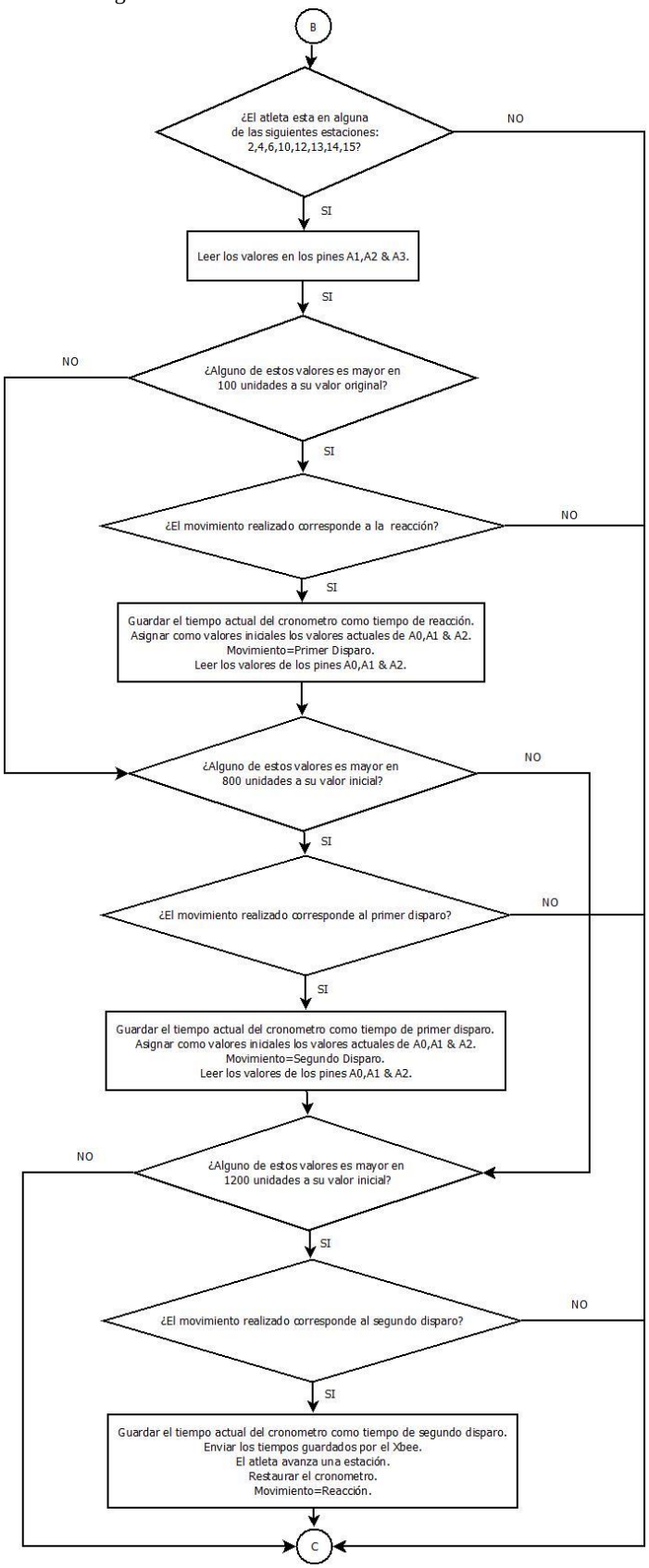
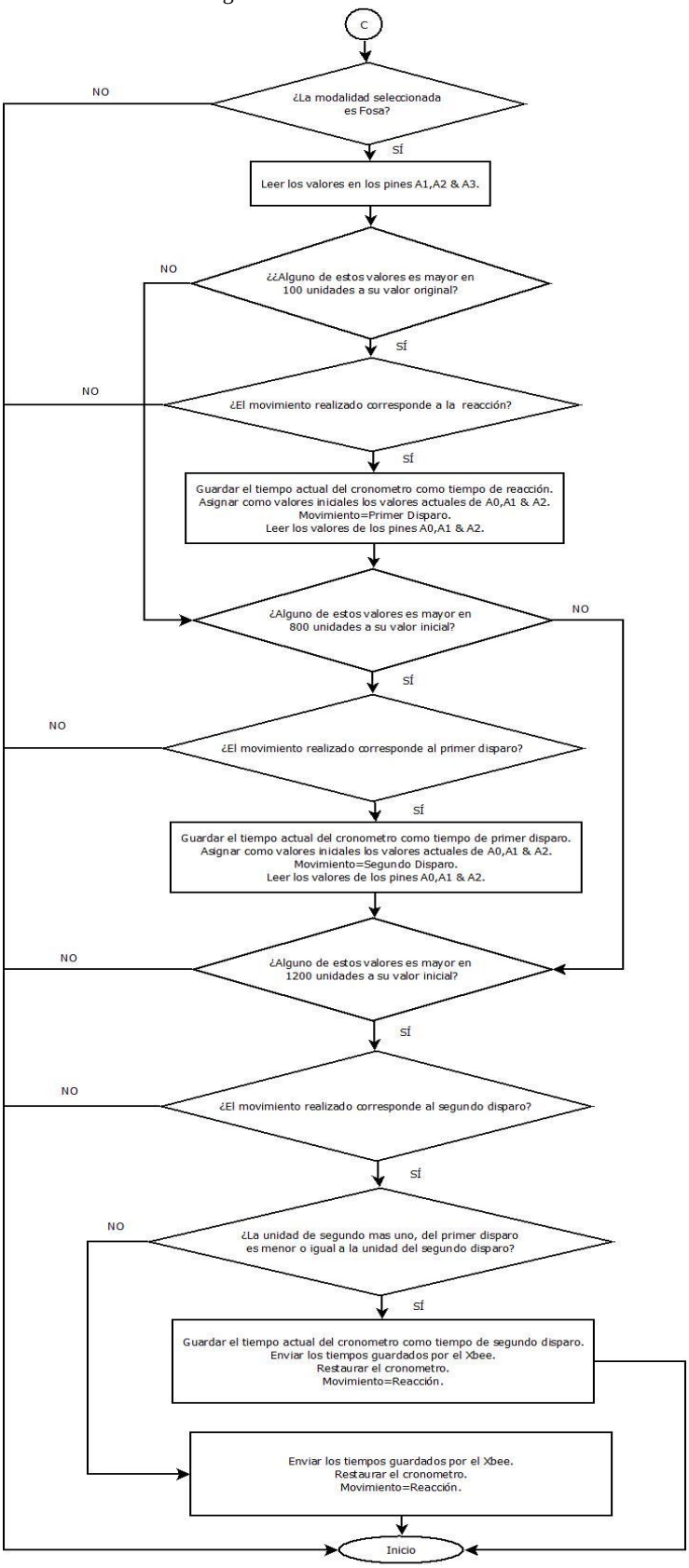


Figura 165. Modalidad fosa.



3. Interfaz gráfica

Figura 166. Interfaz gráfica parte 1.

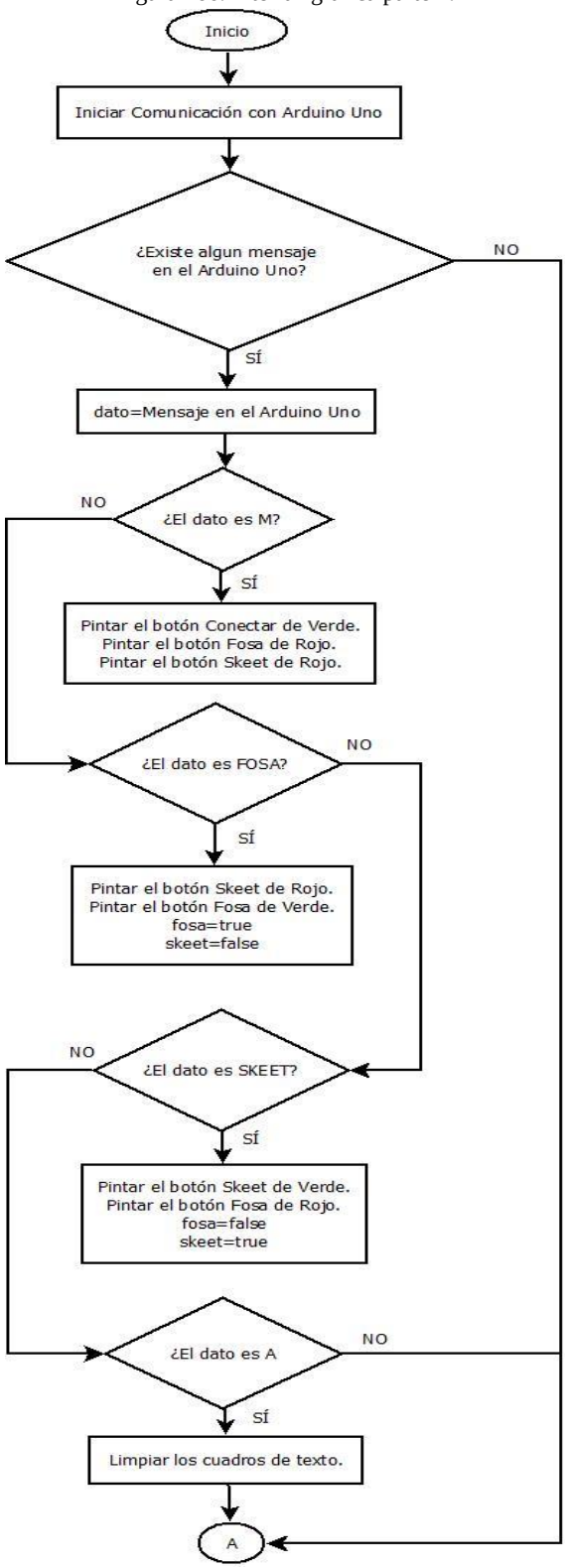


Figura 167. Interfaz gráfica parte 2.