UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Ciencia Aplicada al Deporte

Módulo: Obtención y procesamiento de datos para sensor de tiempo de reacción en tiro con armas de caza.

Trabajo de Graduación en modalidad de Megaproyecto presentado por:
Alejandro Noé Díaz Vargas
(Licenciatura en Ingeniería Mecatrónica)

Guatemala, noviembre 2017

Vo. Bo.:

(f)_____
Ing. Edwin Daniel Gamboa Palacios

Tribunal Examinador:

(f)_____
Ing. Edwin Daniel Gamboa Palacios

(f)_____
MSc. Carlos Alberto Esquit Hernández

(f)_____
Ing. Luis Pedro Montenegro Mejicanos

Fecha de aprobación: Guatemala……

ÍNDICE

LISTA DE CUADROS

LISTA DE FIGURAS

RESUMEN

El Megaproyecto Ciencia Aplicada al deporte, en la modalidad Tiro con armas de caza, buscaba desarrollar una solución de bajo costo que le proporcionara a un atleta de tiro sus tiempos de reacción y de disparo, para las modalidades Fosa y Skeet. Para llegar a la solución se establecieron dos módulos. El primer módulo buscaba obtener, procesar, desplegar y almacenar los tiempos de reacción y tiro. El segundo buscaba desarrollar la parte física, y el acondicionamiento de la señal para activar el cronómetro. El primer módulo se desarrolló en cinco prototipos, cuyo costo fuese menor a Q1000.00.

Los primeros dos prototipos fueron capaces de medir el tiempo de reacción para Fosa. Sin embargo, el costo de los mismos fue mayor a los Q1000.00 establecidos como objetivo. El tercer prototipo fue capaz de obtener los tiempos de reacción, primer y segundo disparo, correctamente para Skeet, y el tiempo de reacción y primer disparo para Fosa. Este prototipo obtuvo el costo más elevado.

El cuarto prototipo, fue capaz de desplegar datos de forma correcta, mas no de almacenarlos. La implementación de la GUI permitió la reducción de costos, cumpliendo así con el objetivo trazado. La versión final contó con una nueva GUI, permitiéndole desplegar y almacenar los tiempos de reacción, primer y segundo disparo para Skeet, y tiempos de reacción y primer disparo para Fosa de forma correcta. Esta versión cumplió con el objetivo de costo inferior a los Q1,000.00 ya que contó con los componentes de la versión anterior.

I. INTRODUCCIÓN

El tiro con armas de caza es un deporte que pone a prueba la precisión y concentración de un atleta. El deporte cuenta con las modalidades Fosa y Skeet en donde el objetivo principal es acertar la mayor cantidad de platos en el menor tiempo posible. Sin embargo, existen dos diferencias principales entre ambas modalidades. En la modalidad Fosa, el atleta desconoce la trayectoria a recorrer por el disco y decide la cantidad de disparos a realizar por disco, un disparo o dos. Mientras que en la modalidad Skeet, el atleta conoce la trayectoria del disco y en cada lanzamiento se le imponen los disparos a realizar por disco.

El presente trabajo forma parte del Megaproyecto Ciencia Aplicad al Deporte, específicamente el módulo de diseño e implementación de una solución de bajo costo capaz de medir, desplegar y almacenar el tiempo de reacción y de disparo de los atletas de tiro con armas de caza de las modalidades Fosa y Skeet.

Para lograr el desarrollo de la solución realizó una serie de propuestas que se presentaron y validaron en una reunión con los psicólogos del Comité Olímpico Guatemalteco, el psicólogo y el entrenador de tiro. Con la propuesta definida y los parámetros de diseño establecidos se procedió a realizar 5 iteraciones de prototipo, las cuales se probaron en los entrenamientos de varios atletas de tiro de ambas modalidades.

La versión final del prototipo logró obtener, desplegar y almacenar de forma correcta los tiempos de reacción, primer disparo y segundo disparo para la modalidad Skeet. En la modalidad Fosa, solo fue capaz de obtener, desplegar y almacenar de forma correcto los tiempos de reacción y el primer disparo. Esta versión tuvo un costo de Q870.79, inferior a los Q1,000.00 establecidos en los parámetros de diseño.

II. OBJETIVOS

A. Generales

1. 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.

B. Específicos

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

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

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

4. 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.

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

III. JUSTIFICACIÓN

El tiro con armas de caza ha sido uno de los deportes que más éxitos ha cosechado para el país, sin embargo, aún tiene una asignatura pendiente y es ganar una medalla durante las justas olímpicas. Es por ello que se busca desarrollar herramientas que permitan a los atletas, mediante la obtención de información objetiva, mejorar su rendimiento.

Actualmente existen diversos equipos en el mercado utilizados por atletas y entrenadores de renombre internacional, sin embargo, para el Comité Olímpico Guatemalteco el proceso de importar esos equipos al país es sumamente costos y extenso, y el tiempo es un factor clave para la preparación de los atletas.

Por ello se plantea una solución de bajo costo que sea capaz de medir el tiempo de reacción y disparos de los atletas de tiro con armas de caza. Dicha solución busca que los atletas logren mejorar su rendimiento en tiempo de reacción al momento de que un blanco es lanzado. Esta será realizada de tal manera que no incomode al atleta durante sus prácticas y que proporcione la información necesaria para el posterior análisis por entrenadores y atletas.

El sistema debe ser capaz de realizar varias tareas, para lo cual se ha divido en varios módulos, esto con el objetivo de darle prioridad a los módulos críticos, ya que como se mencionó previamente en la preparación de un atleta el tiempo es un factor clave. El tamaño del sistema es un requerimiento de suma importancia para lograr implementar esta solución que no perjudique al atleta durante sus entrenamientos. Otro requerimiento importante es la comunicación entre los datos obtenidos y una fuente de almacenamiento, la misma debe ser de manera remota para no colocar cables que interfieran con el movimiento natural del atleta.

IV. MARCO TEÓRICO

A. 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 .12 y cartuchos de 24 gramos de plomos, en una cancha al aire libre. En la Figura *1*. 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 1. Arma y Municiones utilizadas en Tiro con Armas de Caza.

(Sichling, 2017)

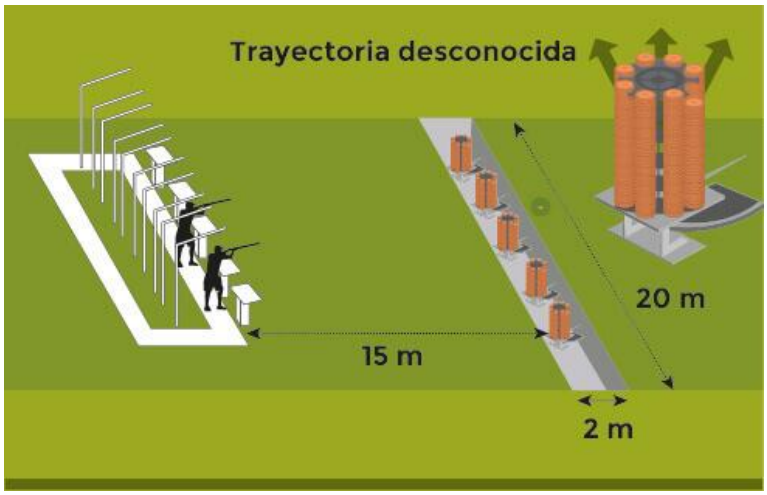
Aparte de la escopeta los atletas cuentan con gafas para proteger los ojos en caso exista rebote de perdigones, y tapones u orejeras para proteger los oídos del ruido del impacto de disparo. Adicionalmente cuentan con un chaleco deportivo que utilizan para llevar municiones en los bolsillos y proteger al atleta del culatazo. En la Figura *2*. se observa un atleta con la indumentaria mencionada previamente, así como la posición correcta de sostener la escopeta. (Marca, 2017)

Figura 2. 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 3. 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 3. Cancha Modalidad Fosa.



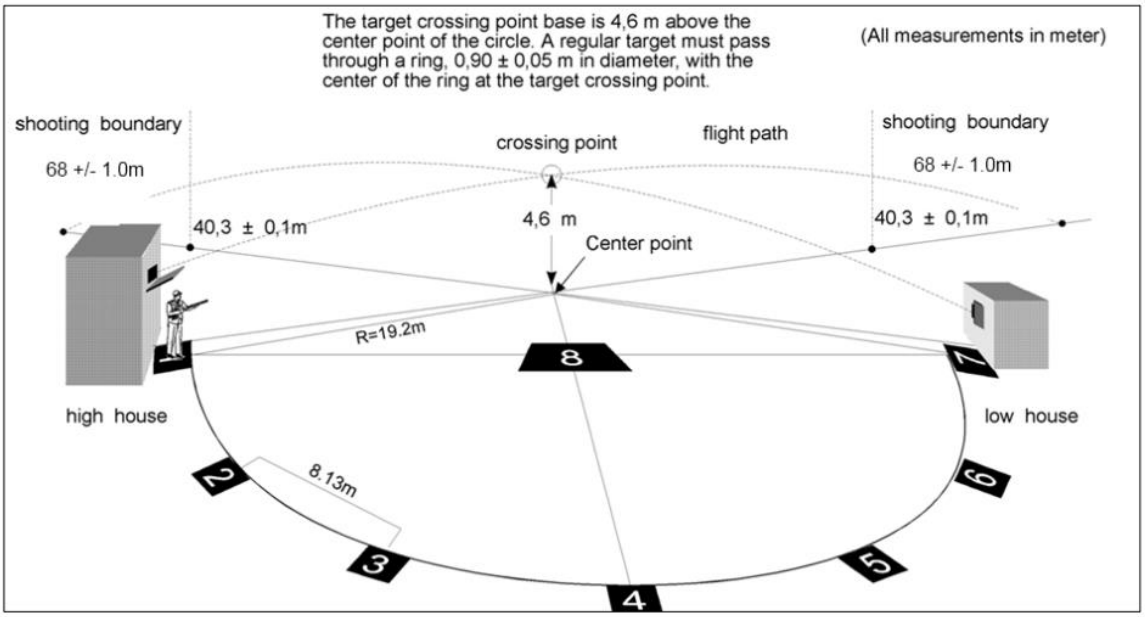
(Marca,2017)

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 4. 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 4. Cancha Modalidad Skeet.



(ISSF,2013)

B. CIENCIAS DE LA ACTIVIDAD FISICA

Las ciencias aplicadas al deporte o ciencias de la actividad física son el conjunto de disciplinas integradas que estudia la función del cuerpo humano durante el ejercicio, cómo el deporte y la actividad física promueven la salud y el rendimiento de un atleta desde diversas perspectivas. Estas disciplinas se pueden agrupar en tres grandes grupos, físico, fisiológico y cultural. (Díaz, 2017) (Pedraz, 1998)

1. Psicología del Deporte Según la Federación Europea de la Psicología Deportiva, la psicología del deporte es el estudio psicológico de la base, procesos y efectos en el deporte. Se encuentra divida en la psicología académica, la cual se enfoca en todos los factores que afectan la participación y el rendimiento en el deporte y en la psicología aplicada, que se centra en la aplicación psicológica para mejorar el rendimiento atlético. Los temas tratados por la psicología del deporte son personalidad, actitudes, agresión, estrés, ansiedad, dinámica de grupo motivación y adquisición de habilidades. (Jarvis, 2006)

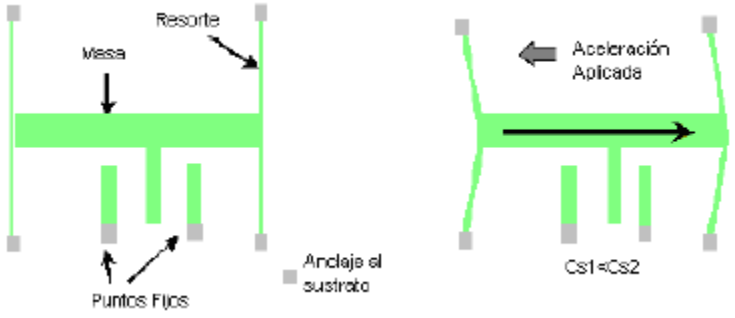
    a. Biofeedback Es una terapia del cuerpo y la mente que utiliza instrumentos electrónicos para ayudar a los individuos a ganar conciencia y control sobre sus procesos psicofisiológicos. Los instrumentos utilizados miden la actividad muscular, la temperatura de la piel, la actividad electro dermal, respiración, ritmo cardiaco, variabilidad cardiaca, presión sanguínea, actividad eléctrica cerebral y flujo sanguíneo cerebral. Esta técnica es efectiva para tratar una variedad de desórdenes médicos y psicológicas. Las terapias de Biofeedback, guían a las personas para facilitar el aprendizaje del control voluntario sobre su cuerpo y mente. (Yucha, 2004)

    b. Biomecánica Deportiva Es una disciplina científica que aplica las leyes mecánicas para estudiar las fuerzas que actúan sobre el cuerpo humano y los efectos que las mismas producen. Esta disciplina busca responder a las interrogantes de ¿cómo se debe determinar los factores que determinan la técnica de un campeón?, O ¿Cuáles son las limitantes que afectan a un atleta? (Martínez, 2009)

C. 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 5. (Arenas,2008)

Figura 5. Estructura de un acelerómetro capacitivo. (A) Sensor en reposo. (B)Respuesta a una aceleración aplicada



(A) (B)

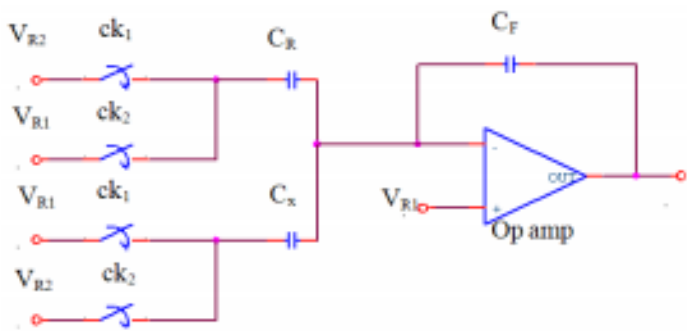
(Arenas,2008)

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 6. Convertidor de Capacitancia a Voltaje.



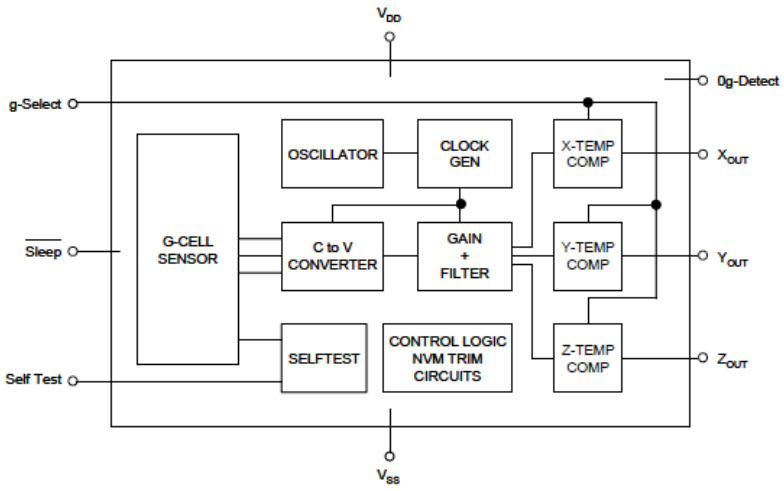
(Alam, et al, 2010)

En la Figura 6. se puede observar una versión de un circuito convertidor de capacitancia en voltaje o CVC por sus siglas en inglés. C_X es el valor de capacitancia detectado por el sensor y C_R y C_F son valores previamente definidos. V_{R1} es el voltaje de modo común y V_{R2} es el voltaje de referencia. C_{K1} y C_{K2} son señales de reloj independientes. Cuando la señal C_{K2} se encuentra en un estado lógico alto, el voltaje de referencia va a cargar el capacitor C_X, mientras que el capacitor C_F va a almacenar el voltaje offset del OpAmp. Cuando la señal C_{K1} se encuentra en un estado lógico alto, el voltaje de referencia carga el capacitor C_R y el capacitor C_F está conectado a la salida. Si se sigue el principio de conservación de la carga, el voltaje de salida V_o 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 7. 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 7. 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)

D. 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).

2. 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)

3. 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

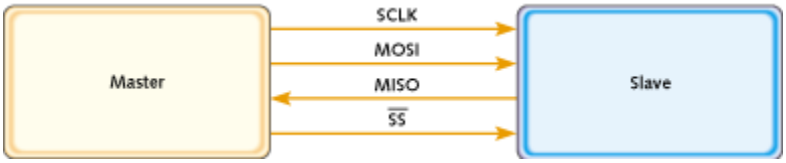
Modo	Polaridad de Reloj (CPOL)	Fase de Reloj (CPHA)	Descripción
0	0	0	El estado del reloj permanece en cero y la información se envía en cada transición de bajo a alto.
1	0	1	El estado del reloj permanece en cero y la información se envía en cada transición de alto a bajo.
2	1	0	El estado del reloj permanece 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 8. 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).

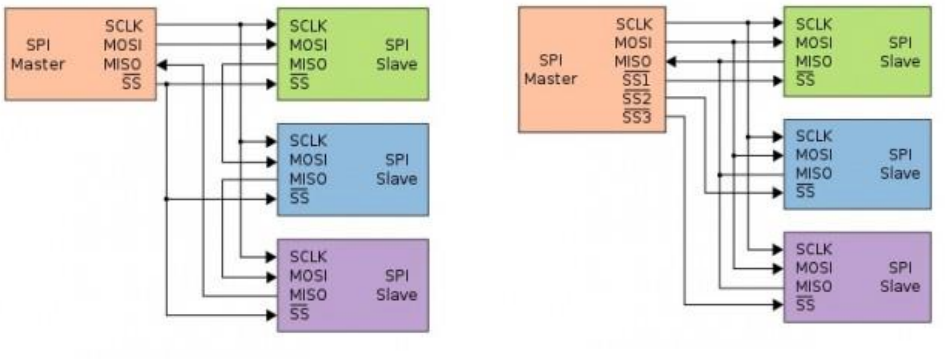
A diferencia de la configuración encadenada, la configuración paralela permite que el maestro envíe y reciba datos de cada esclavo de manera independiente, para seleccionar con que esclavo se comunicara existe una línea de selección de esclavo en el maestro por cada esclavo que exista en el sistema. En la Figura 9. se puede observar la diferencia entre ambas configuraciones. (Navarro, 2014)

Figura 8. Conexión Punto-Punto mediante SPI



(Kalinsky & Kalinsky, 2002)

Figura 9. Conexión Punto-Multipunto mediante SPI A) Configuración Encadenada (B) Configuración Paralela



(A) (B)

(Navarro,2014)

E. REDES INALAMBRICAS DE AREA 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)

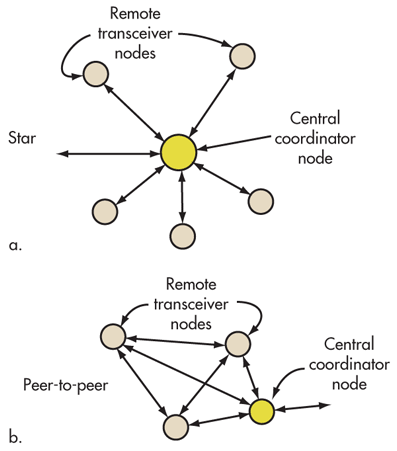
El objetivo principal de este estándar es proveer un formato base al que otros protocolos y características se puedan agregar en las capas superiores del modelo de interconexión de sistemas abiertos. El estándar puede operar en tres frecuencias distintas, aunque la más utilizada es la de 2.4 GHz, es 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,

Figura 10. 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)

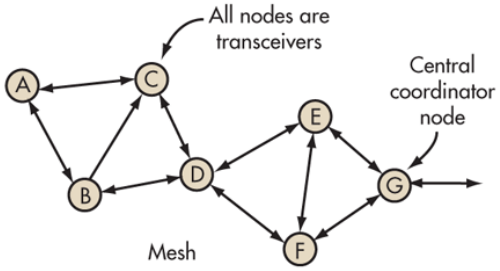
Figura 10. Modelos de topología definidos por IEE 802.15.4



(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)

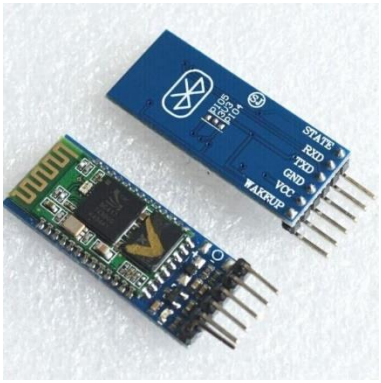
Figura 11. Topología Mesh utilizada en redes ZigBee.



(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 12. Módulo HC-05



(Prometec, 2017)

F. 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 13. Módulo Xbee serie 2 con antena de cable.

(Digi, 2017)

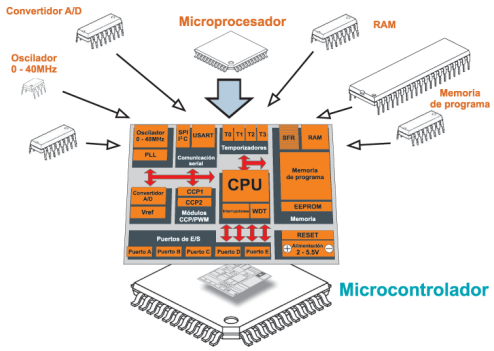
2. XCTU Es una aplicación multiplataforma gratuita, que fue diseñada para permitir a los desarrolladores interactuar con los módulos RF de Digi, mediante una interfaz gráfica sencilla. Cuenta con diversas herramientas que facilitan la configuración y pruebas de los Módulos Xbee. Con esta plataforma se 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).

G. 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 14. Arquitectura general de un microcontrolador.



(Mikroelectronika,2010)

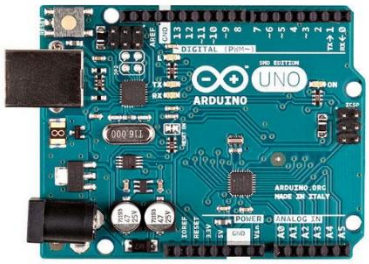
1. Arduino Uno El Arduino Uno es una tarjeta basada en el microcontrolador ATmega328 de Atmel. Tiene 14 pines digitales de entrada/salida, de los cuales 6 pueden ser utilizados como salidas de ancho de pulso modulable o PWM, por sus siglas en inglés. Tiene 6 entradas analógicas, un cristal oscilador 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)

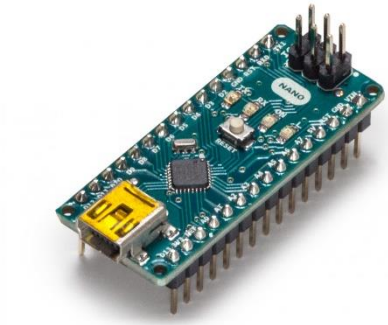
Figura 15. Arduino Uno.



(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 16. Arduino Nano.

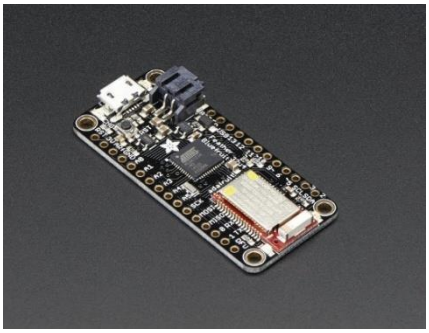


(Arduino,2017)

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 17. Adafruit Feather 32u4



(Adafruit, 2017)

H. INTERFÁZ 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*).

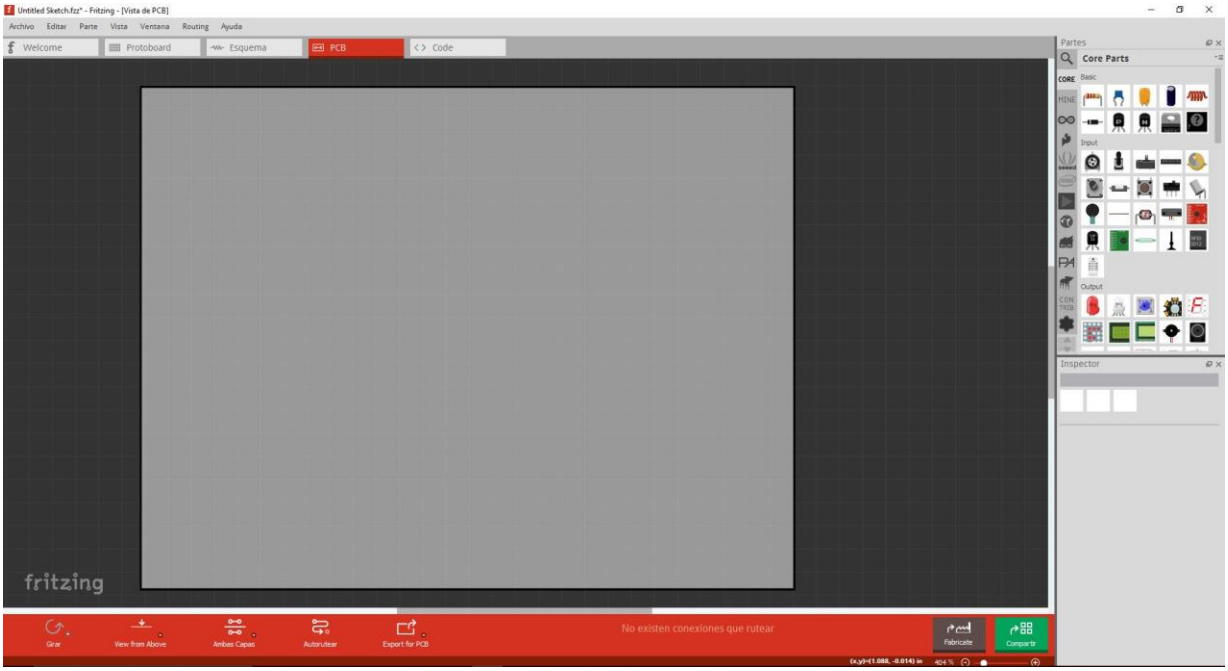
2. Processing Es un software flexible de código abierto, que relaciona conceptos de principios de forma visual con movimiento e interacción. El lenguaje de Processing es un lenguaje de programación de texto diseñado para generar y modificar imágenes. Processing permite realizar operaciones de dibujo 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)

I. PCB – Printed Boarded Circuit

Un PCB o circuito impreso es un soporte de material aislante en donde se conectan componentes de un circuito eléctrico entre sí, utilizando pistas conductoras. El circuito suele servir de soporte físico para la colocación y soldadura de componentes. Los circuitos pueden estar fabricados sobre una de las capas de la placa o en 2 más capas, dependiendo de la complejidad del circuito. En la actualidad para fabricar un circuito impreso existen diversas metodologías, sin embargo, todas cuentan con un factor en común y es el uso de un software que facilite la conexión de los circuitos. El software utilizado puede variar dependiendo de los componentes a utilizar y la complejidad del circuito. (Gallardo, 2015).

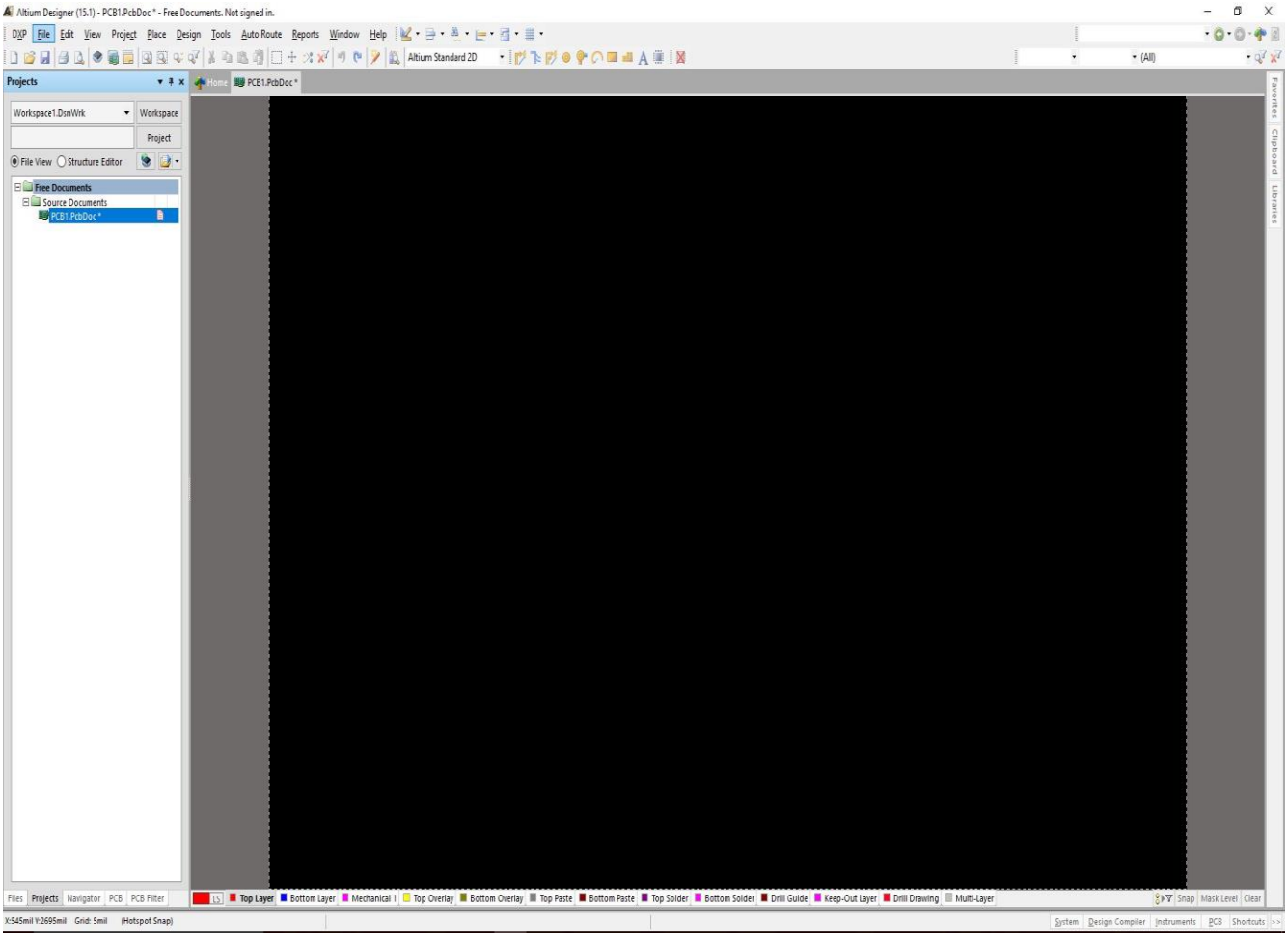
1. Fritizing Es un software de código abierto sencillo que provee las herramientas que facilitan la fabricación de los PCB's. El software cuenta con 3 vistas disponibles para trabajar los circuitos, la primera vista es la de un protoboard, la segunda un esquemático y la tercera la del PCB. La principal 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 18. Entorno Gráfico de Fritzing vista PCB.



2. Altium Designer Es un entorno unificado para el desarrollo de productos electrónicos. Cuenta con herramientas para edición útiles para realizar casi cualquier paso en el proceso de desarrollo de un producto electrónico. El entorno de Altium Designer permite al usuario configurar el espacio de trabajo de forma personalizada. Para realizar un PCB en Altium es necesario crear los propios componentes a utilizar, utilizar los predeterminados en sus librerías, o bien descargar componentes realizados por otras personas. Luego de esto se crea un archivo con extensión Prj1PCB y en este se crea el esquemático del circuito, posteriormente este esquemático se traslada a un archivo con extensión PcbDoc y en este se realizan las pistas de los componentes. Altium permite exportar estos archivos para ser manufacturados en máquinas de control numérico. (Rodríguez, 2012)

Figura 19. Entorno Gráfico Altium Designer V. 15.1.



V. METODOLOGÍA

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.

A. 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.

B. Módulo de Comunicación & 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 4. 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 5. 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 4. Peso Por Criterio

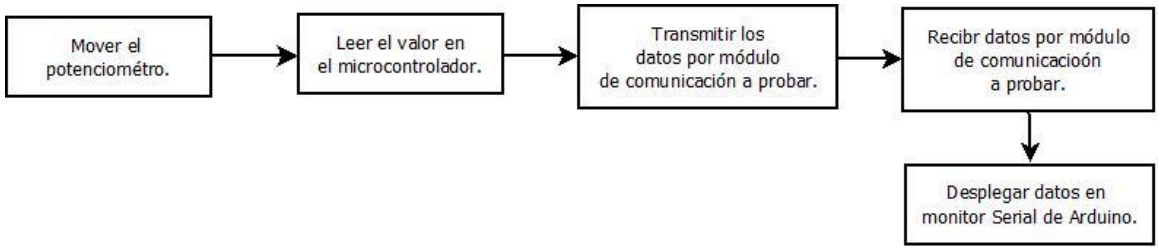
Criterio	Peso
Costo	8
Alcance	7
Documentación Disponible	5
Disponibilidad	8
Dimensiones	9
Peso	7
Compatibilidad	6

Cuadro 5. 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	10%	0.1842	0.0184	0.2000	0.0200	0.1556	0.0156	0.2059	0.0206
Disponible									
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 20. 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 20. Lógica de funcionamiento pruebas módulos de comunicación.



1. 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.

2. 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.

3. 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 21. Posteriormente se verificó el funcionamiento de los módulos en el mismo software.

Figura 21. Parámetros de Configuración Módulos Xbee.

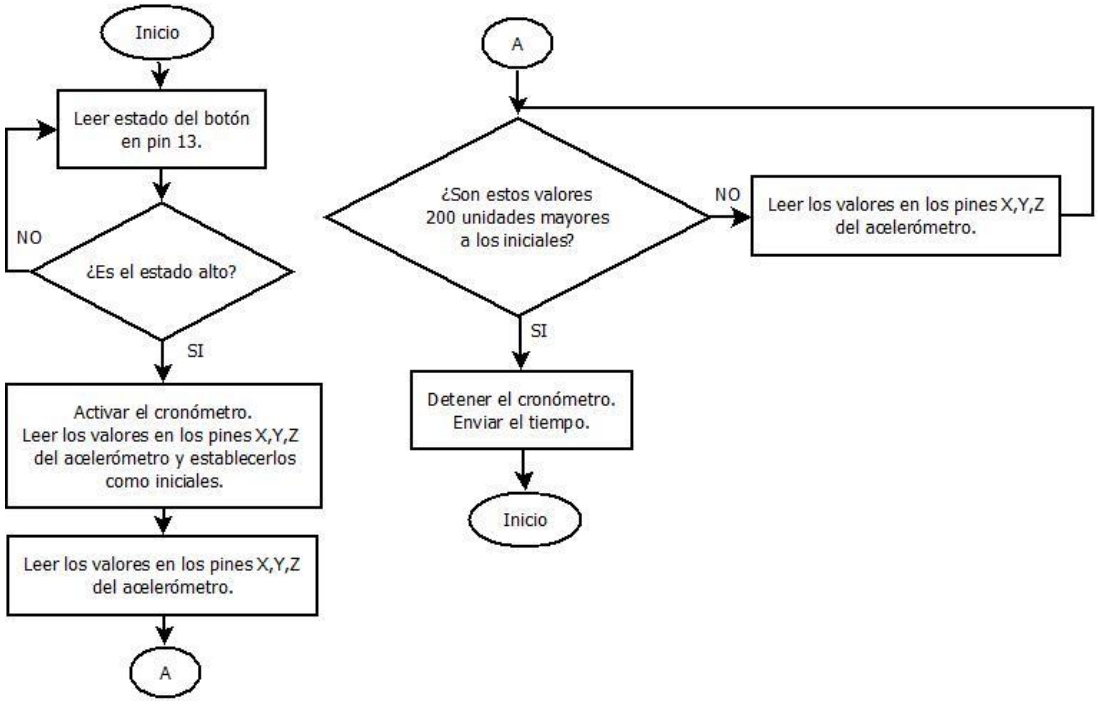
Se procedió a seleccionar el módulo que habría logrado un mayor alcance. Ya con el sensor y los módulos de comunicación seleccionados se procedió a seleccionar los microcontroladores que se utilizarían, para ello se seleccionaron las dos placas Arduino utilizadas. El arduino Nano estaría montado en la 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.

C. Primer Prototipo

El primer prototipo consistió en implementar la lógica presente en la Figura 22. 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 22. Lógica de operación código primer prototipo.



D. 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 23. 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 24. 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 25. se observa la configuración de los mismos.

Figura 23.Diagrama de Flujo Arduino Uno Prototipo 2.

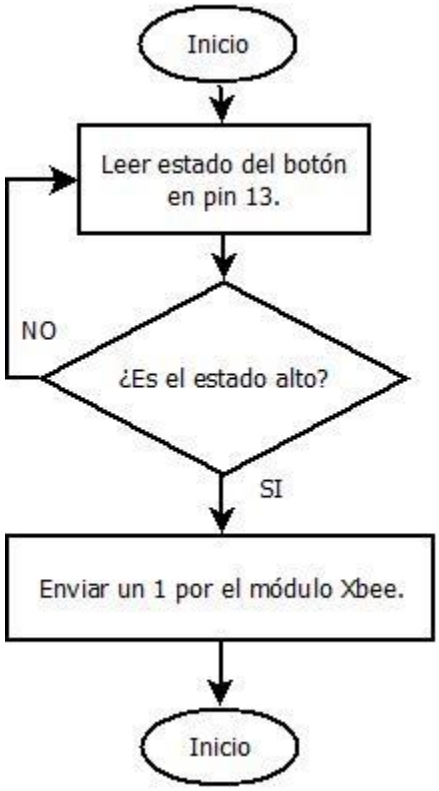


Figura 24. Diagrama de Flujo Arduino Nano Prototipo 2.

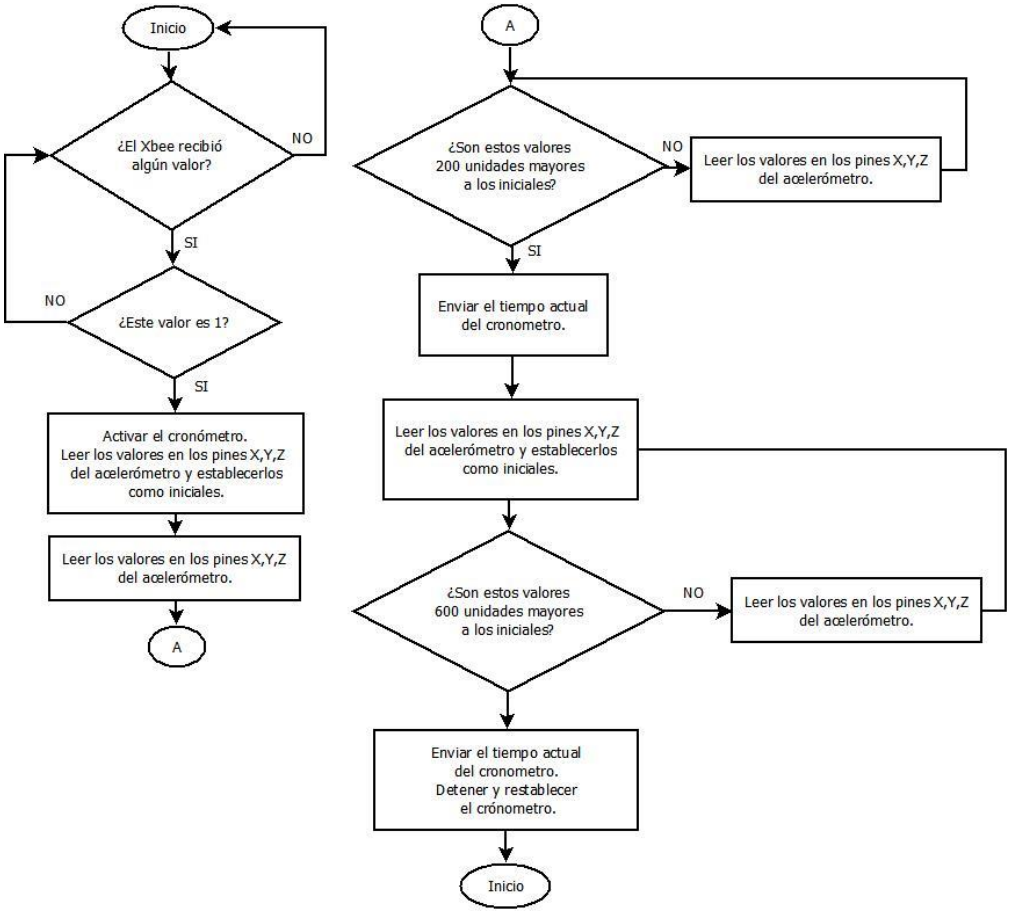
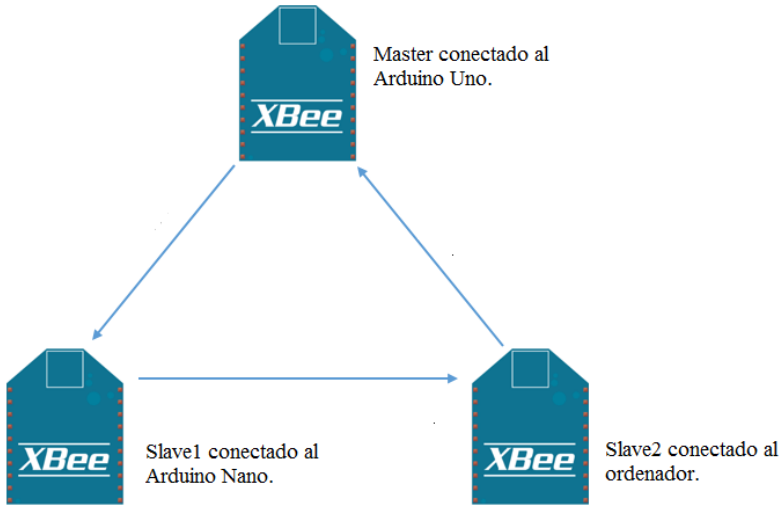
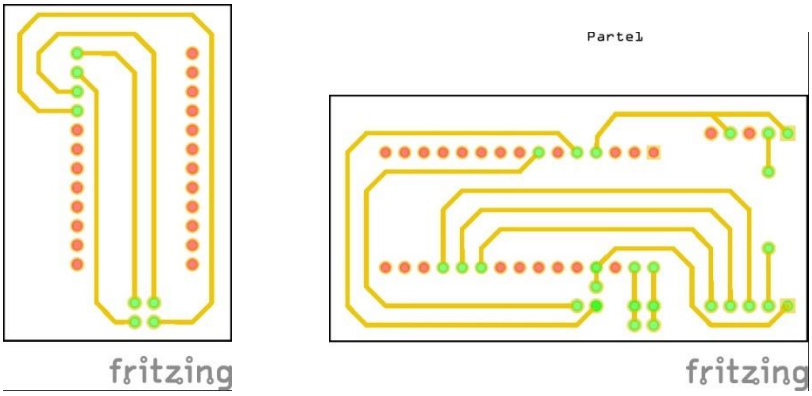


Figura 25. Configuración Módulos Xbee Prototipo 2.



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 26. se pueden observar ambos diseños de las placas. Estas placas se fabricaron utilizando el método de planchado del PCB.

Figura 26. Diseño de Placa Segundo Prototipo. A) Placa Xbee B) Placa Arduino y Acelerómetro.



A) B)

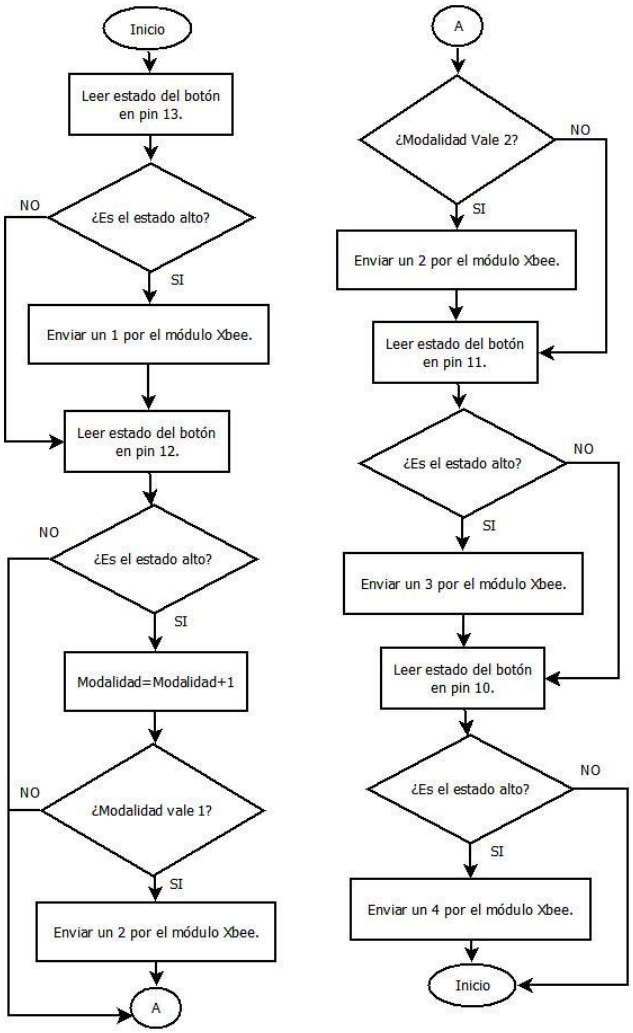
E. Tercer Prototipo

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 & 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 *27*. 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 6. Funciones a Ejecutar en el Arduino Nano Prototipo 3.

Carácter Recibido por Arduino Nano	Función a Ejecutar
49	Activar el cronometro.
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.

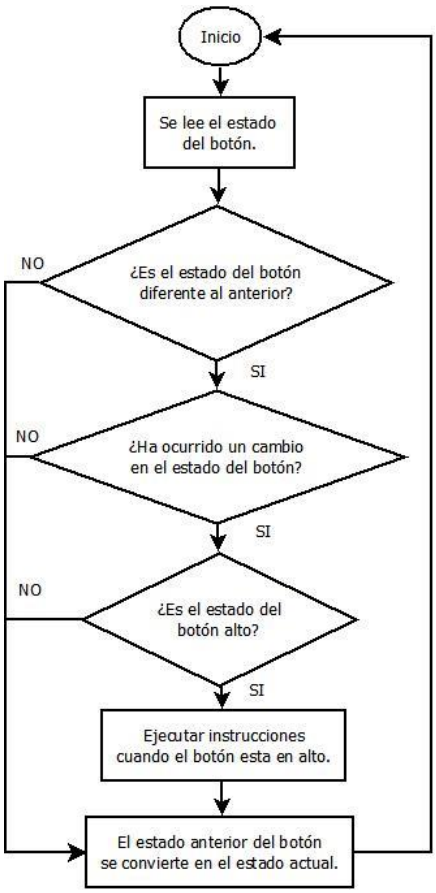
Figura 27. Diagrama de Flujo Arduino Uno Prototipo 3.



F. 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 28. 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 28. 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 7. se observa el carácter identificador y su dato correspondiente.

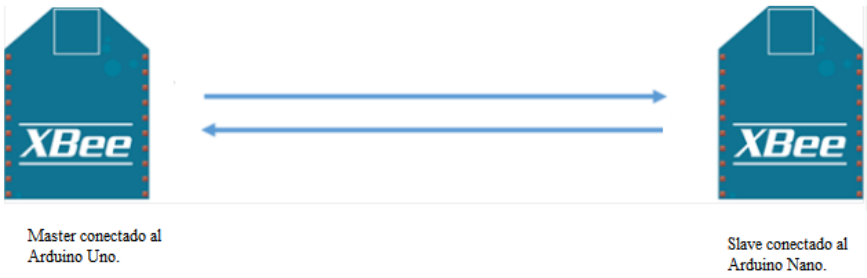
Cuadro 7. Carácter Identificador con su dato correspondiente.

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 *29*. 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 29. Configuración Módulos Xbee Prototipo 4.

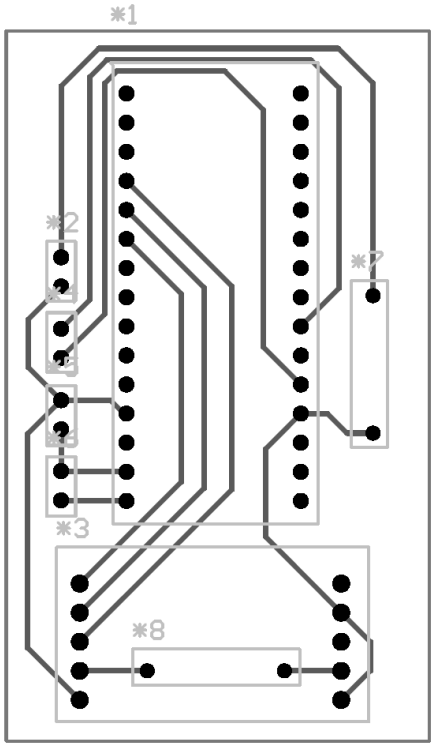


G. Quinto Prototipo

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 30. 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 30. Diseño PCB Prototipo 5.



VI. RESULTADOS

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 8. 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 9. 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 10. 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 31. 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 31. 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 11. 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 12. se observan los datos agregados a Excel por medio de esta interfaz, en una de las pruebas realizadas por Pedro Zaya.

Figura 32. Interfaz Gráfica Processing Prototipo 4.

Cuadro 12. 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 33. Interfaz Gráfica en Java Prototipo 5.

Figura 34. PCB Arduino Nano y Sensor

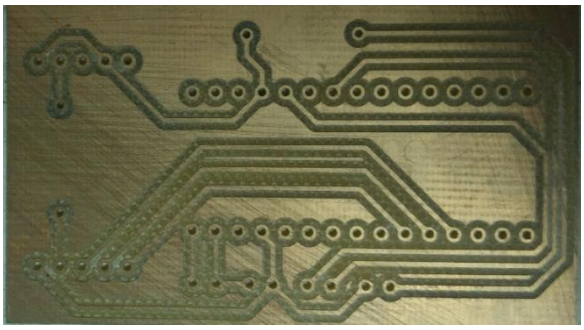


Figura 35. Esquemático Prototipo 5.

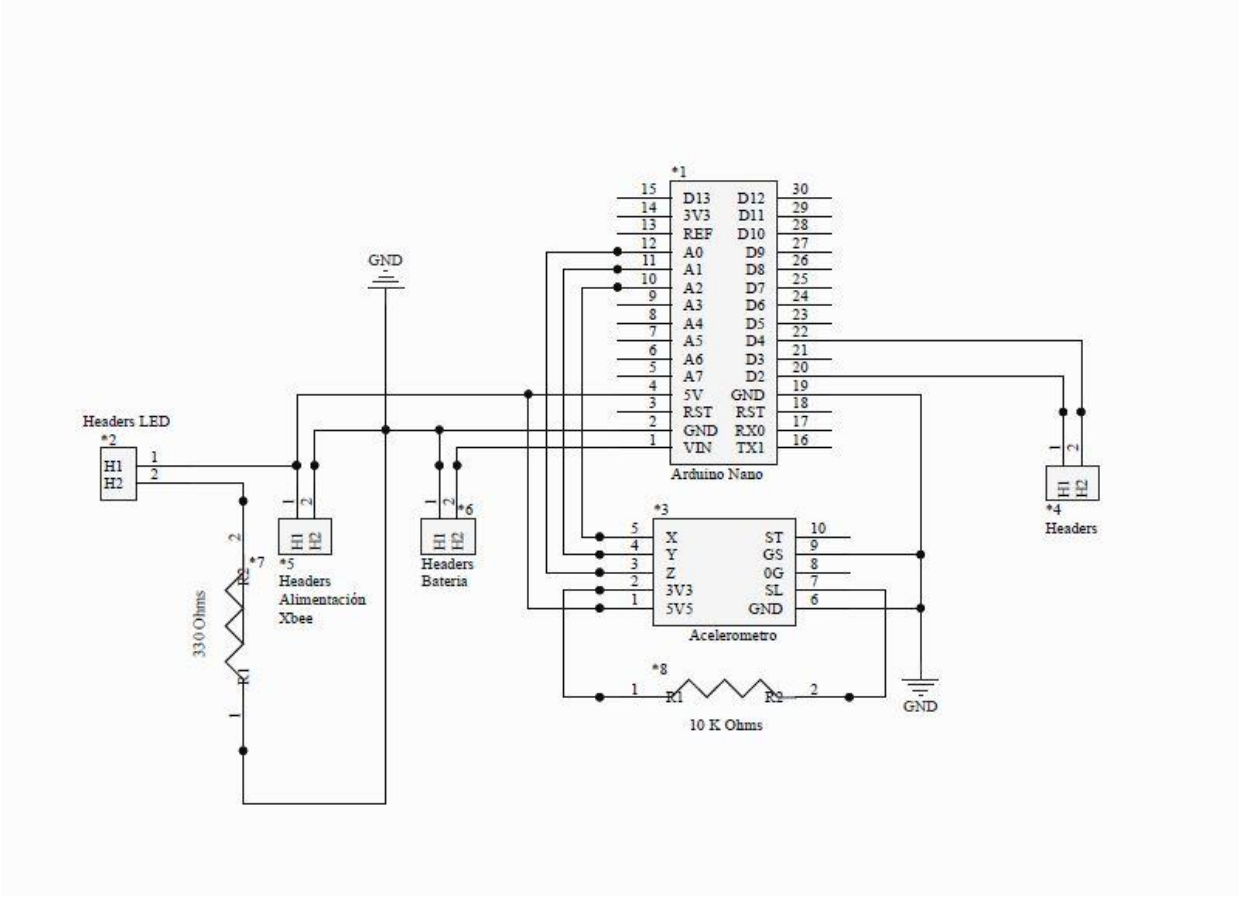
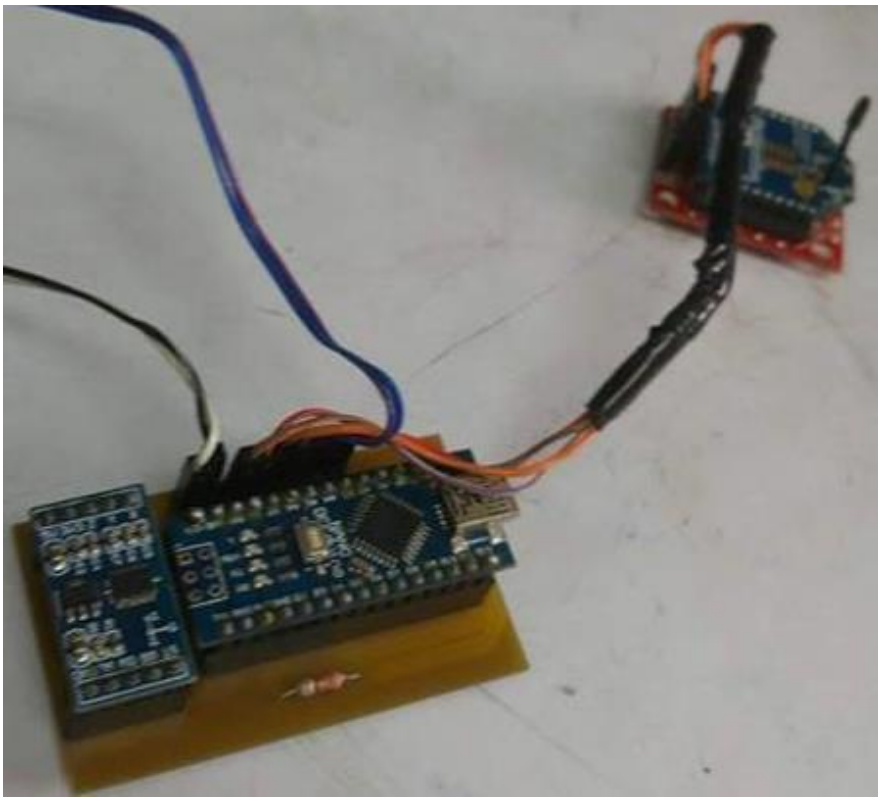


Figura 36. Arduino Nano y Acelerómetro montados en el PCB.



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 13. 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 14. 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 15. Entreno Fosa Dany Brol 1 Disparo.

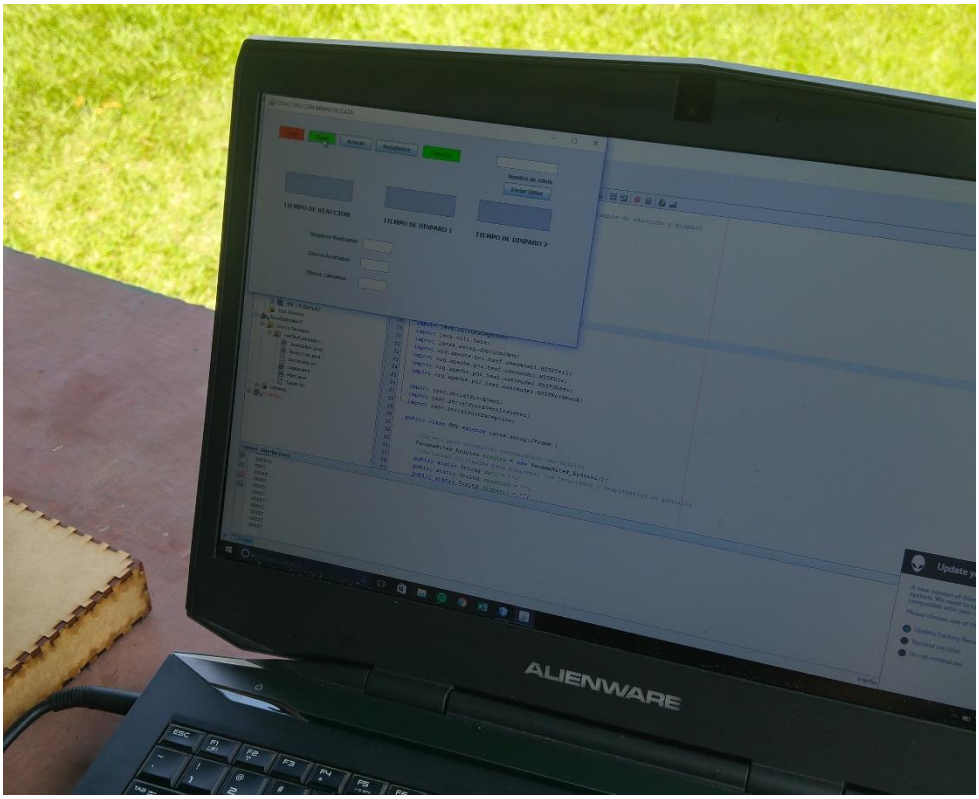
Tiempo de Reacción (m:s:ds:cs)	Tiempo de Disparo 1 (m:s:ds:cs)
0:0:0:73	0:0:6:00
0:0:0:37	0:0:4:57
0:0:0:57	0:0:2:61
0:0:0:92	0:0:4:14
0:0:0:68	0:0:2:70
0:0:0:24	0:0:4:48
0:0:2:59	0:0:4:63
0:0:0:46	0:0:0:48
0:0:0:63	0:0:8:09
0:0:2:50	0:0:4:63
0:0:0:05	0:0:6:34
0:0:0:18	0:0:0:19
0:0:0:47	0:0:6:70
0:0:0:13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

Cuadro 16. 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 37. Entreno Juan Ramón Schaeffer con prototipo 5.

Figura 38. Comunicación Interfaz Gráfica Sensor en Entreno de Juan Ramón Schaeffer.



VII. DISCUSIÓN

A. 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.

B. 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 31. 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 8. 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.

C. 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 9. 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.

D. 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 10. 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 12. 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.

E. Quinto Prototipo

La implementación de la interfaz en Java se considera exitosa ya que si se observa el Cuadro 13. 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 13. & Cuadro 14. 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 13. 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 14. 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 15. 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 16. 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.

F. 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 11. en la Figura 37. & Figura 38. 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 11. 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 logró diseñar un dispositivo con un costo total de Q870.79 (USD 119.45)

2. 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.

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

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

5. 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.

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

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

IX. RECOMENDACIONES

1. 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.

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

3. 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.

4. 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.

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

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

7. 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.

X. BIBILOGRAFÍA

1. Adafruit. *Adafruit Feather 32u4 Bluefruit LE* https://learn.adafruit.com/adafruit-feather-32u4-bluefruit-le/overview [23/01/2017]
2. Adafruit. *Installing BLE Library*. https://learn.adafruit.com/adafruit-feather-32u4-bluefruit-le/installing-ble-library [23/01/2017]
3. Alam, Zahirual, *et al*. 2010. «Design of Capacitance to Voltage Converter for Capacitive Sensor Transducer». *American Journal of Applied Sciences*. VII (10): 1353-1357.
4. Arduino. *Arduino Uno*. http://www.arduino.org/products/boards/arduino-uno [17/07/2017]
5. Arduino. *Getting Started with the Arduino Nano.* https://www.arduino.cc/en/Guide/ArduinoNano [17/07/2017]
6. Arduino. *SoftwareSerial Library* https://www.arduino.cc/en/Reference/SoftwareSerial [17/07/2017]
7. Arenas, Marta. 2008. «Sistema para la adquisición y monitorización de aceleraciones mediante microprocesador.». Tesis Universidad de Sevilla. 187 págs.
8. Bell, Douglas. 2003. *Java Para Estudiantes.* 3ª ed. México: Pearson. 664 págs.
9. Camargo, José Luis. 2009 «Modelo de cobertura para redes inalámbricas de interiores.». Tesis Universidad de Sevilla 182 págs.
10. 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]
11. 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]
12. 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]
13. 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]
14. Digi. *Digi Xbee Zigbee* https://www.digi.com/products/xbee-rf-solutions/2-4-ghz-modules/xbee-zigbee#specifications [22/07/2017]
15. Digi. *Sleep Mode* http://docs.digi.com/display/XBeeArduinoCodingPlatform/Sleep+mode [23/08/2017]
16. Durda, Frank. *Serial and Uart Tutorial.* https://www.freebsd.org/doc/en_US.ISO8859-1/articles/serial-uart/ [16/07/2017]

17. Federación Venezolana de Tiro. 2015. *Manual de Introducción al deporte del tiro.* Comité Olímpico de Venezuela. Venezuela. 33 págs.
18. Freescale Semiconductor. *2008*. *MMA7361L.* Arizona. 11 págs.
19. 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]
20. Gallardo, Omar. 2015 «Fabricación de circuito impreso con Proteus.». Tesis Universidad de Valladolid 119 págs.
21. García, Antony. *Librería PanamaHitek_Arduino, v2.8.2* http://panamahitek.com/libreria-panamahitek_arduino/ [05/08/17]
22. ISSF. 2013. *Official Statutes Rules and Regulations.* ISFF. Munich. 497 págs.
23. Jarvis, Matt. 2006. «Introduction». Sports Psychology A Student's Handbook. Canada: Routledge. págs. 1-11.
24. 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].
25. López, Oliverio. 2016. «Diseño y elaboración de un control remoto para lámparas de alumbrado público.» Tesis Universidad Nacional Autónoma de México 89 págs
26. Luna, Lizbeth. 2004. «El diseño de interfaz gráfica de usuario para publicaciones digitales». *Revista Digital Universitaria.* V (7): 2-12
27. Marca. *Tiro Olímpico.* http://www.marca.com/juegos-olimpicos/tiro-olimpico/todo-sobre.html [04/09/17]
28. MikroElektronika. *Introducción Al Mundo De Los Microcontroladores.* https://learn.mikroe.com/ebooks/microcontroladorespicbasic/chapter/introduccion-al-mundo-de-los-microcontroladores/ [17/07/2017].
29. National Instruments. *Comunicación Serial: Conceptos Generales.* http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1 [16/07/2017].
30. Navarro, Kiara.*¿Cómo funciona el protocolo SPI?* http://panamahitek.com/como-funciona-el-protocolo-spi/ [17/07/2017].
31. Oliver, Andrew, *et Al.* 2017. *Apache POI- The Java Api For Microsoft.* https://poi.apache.org/index.html [05/08/2017].
32. Prometec, *Módulo Bluetooth HC-05*. https://www.prometec.net/producto/modulo-bluetooth-hc-05/ [05/08/2017].
33. Ramón, Gustavo. 2009. *Biomecánica deportiva y control del entrenamiento.* Medellín: Funámbulos. 134 págs.
34. Reas, Casey y B. Fry. 2014. *Processing: A programming handbook for visual designers and artists.* 2a ed. Massachusetts: MIT Press. 720 págs.

35. Rodriguez, Juan; M. Tena. 2012. «ECG Recording and Heart Rate Detection with Textile-Electronic Integrated Instrumentation. » Tesis Universidad de Boras 51 págs.
36. Sichling & CO. 2017. *Escopetas.* http://www.armeriasichling.com/12-escopetas
37. Sparkfun. 2017. *Make your Own Fritzing Parts.* https://learn.sparkfun.com/tutorials/make-your-own-fritzing-parts [05/08/2017]
38. Texas Instruments. *2012. KeyStone Architecture Serial Peripheral Interface (SPI)*. Texas. 51 págs.
39. Velasco, Nicolás. «Sistema embebido para la conexión de un PLC Siemens S7-200 a la red GSM.». Tesis Universidad de Sevilla 146 págs.
40. Vicente, Miguel¸ *et Al.* 1998. *Educación física e deporte no século XXI*. Coruña: Universidad de Coruña. 161 págs.
41. Xbee. 2017 *¿Qué es Xbee?* http://xbee.cl/que-es-xbee/ [22/07/2017]
42. Xbee. 2017 *Xbee Configuración Serial* http://xbee.cl/xbee-serie-1-configuracion/ [20/02/2017]
43. Yucha, Carolyn y C. Gilbert. 2004. *Evidence-Based Practice In Biofeedback and Neurofeedback.* Florida: AAPB. 59 págs.

XI. ANEXO

A. Diagramas de Flujo

En las figuras presentadas a continuación encontrará la lógica desarrollada para la implementación de los algoritmos del prototipo 5.

1. Arduino Uno Los siguientes diagramas corresponden al código implementado en el arduino Uno.

Figura 39. Diagrama de Flujo Máquina Lanzadora de Discos Parte 1.

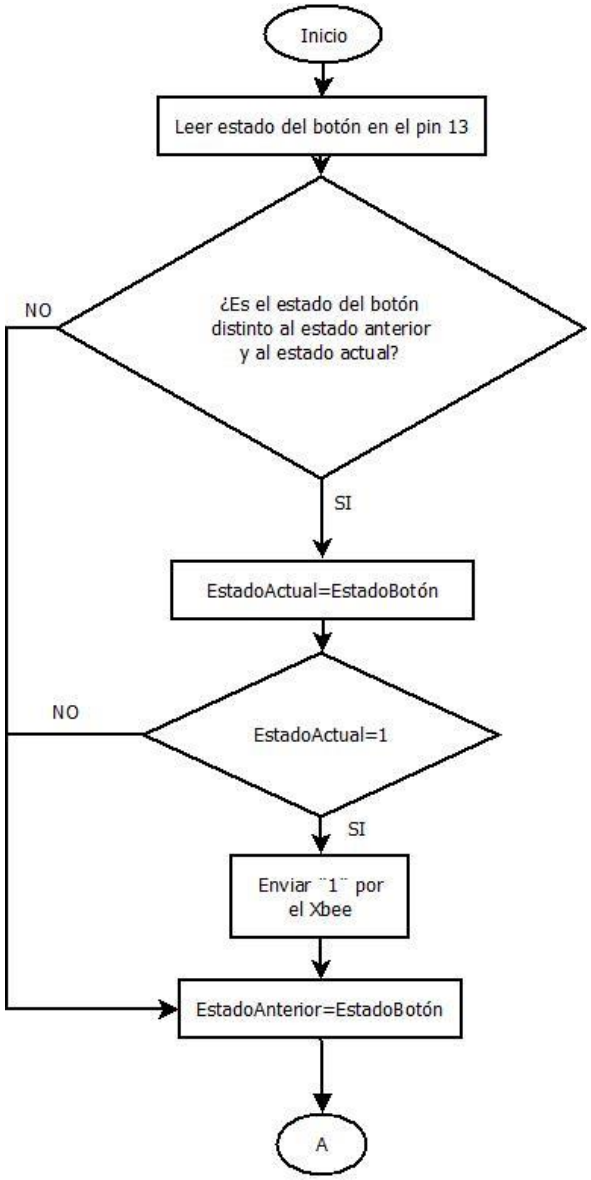
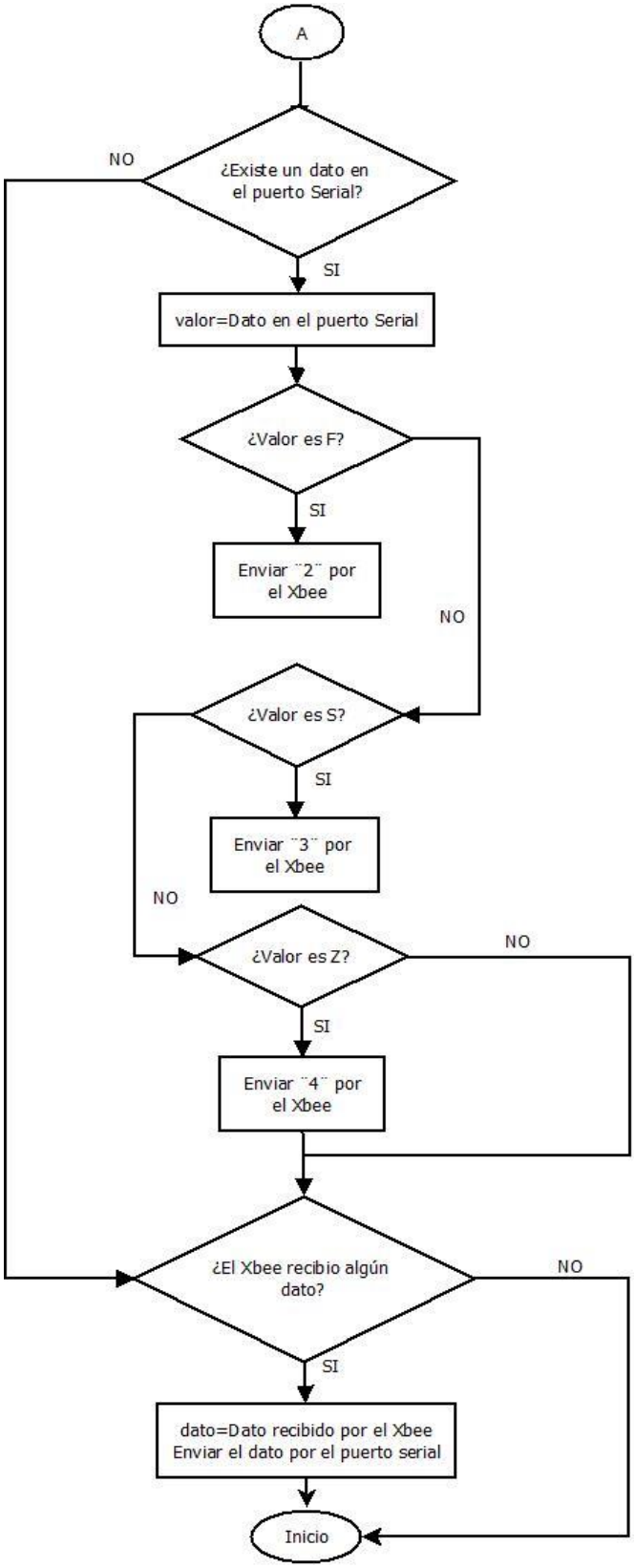


Figura 40. Diagrama de Flujo Máquina Lanzadora de Discos Parte 2.



2. Arduino Nano Las siguientes figuras corresponden al código implementado en el Arduino Nano.

Figura 41. Diagrama de Flujo Selector de Modalidad.

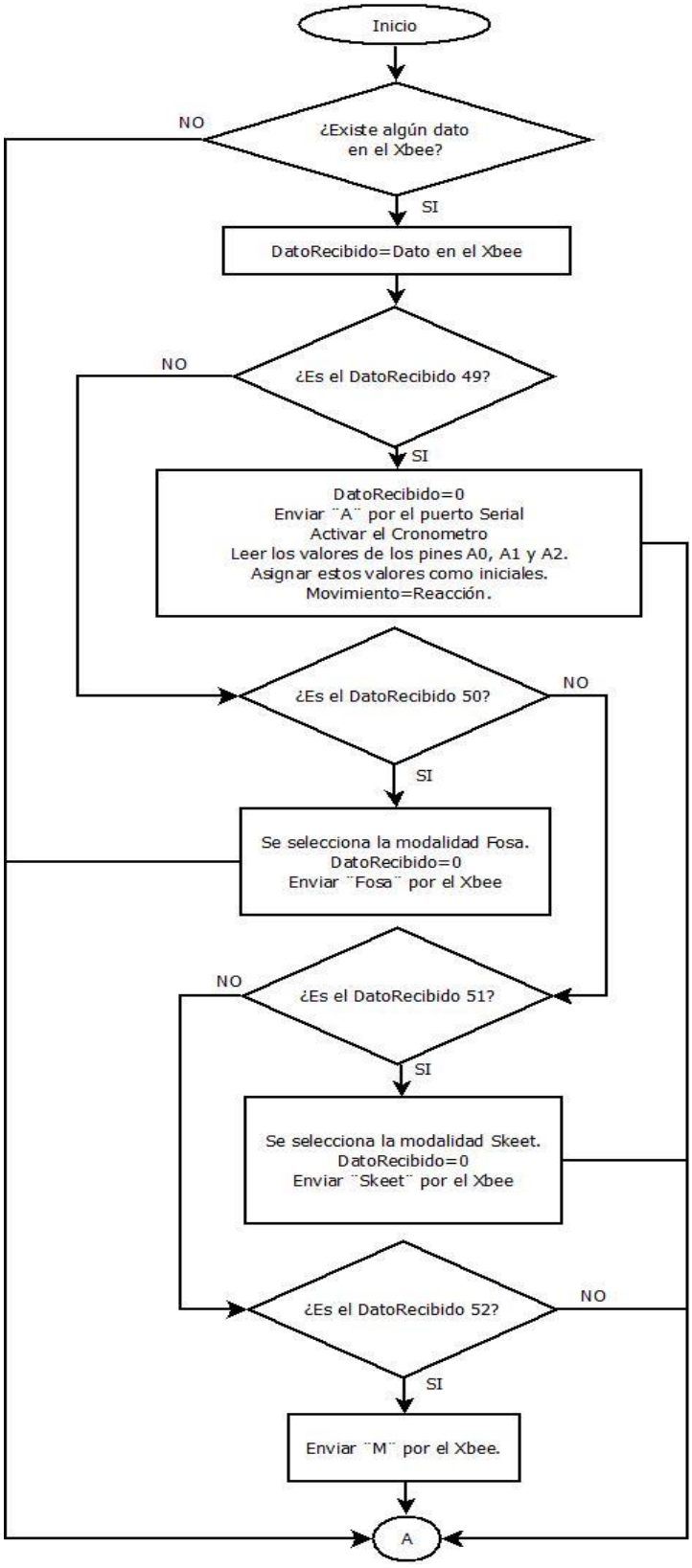


Figura 42. Diagrama de Flujo Modalidad Skeet Lanzamiento Simple.

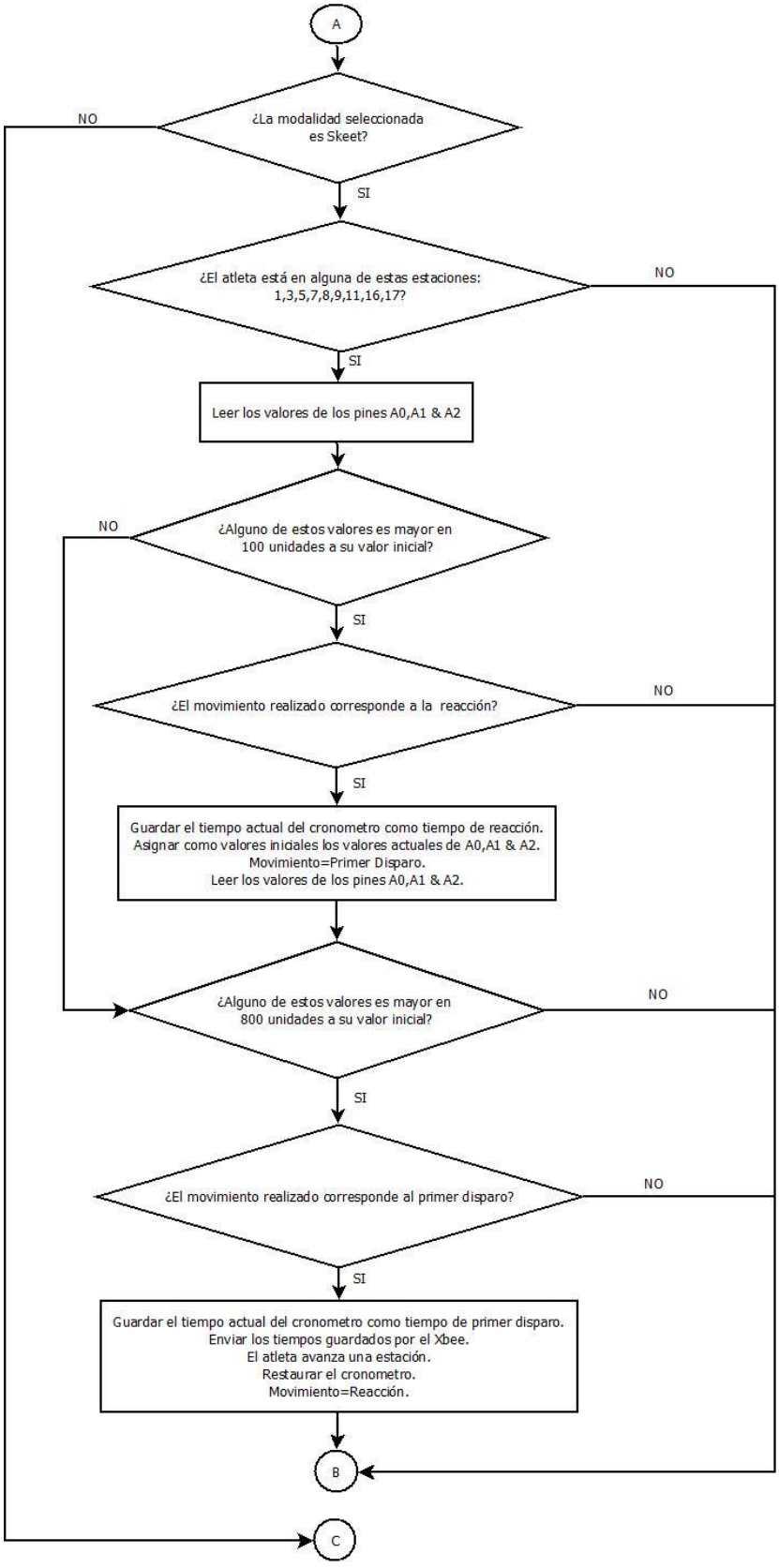


Figura 43. Diagrama de Flujo Modalidad Skeet Lanzamiento Doble.

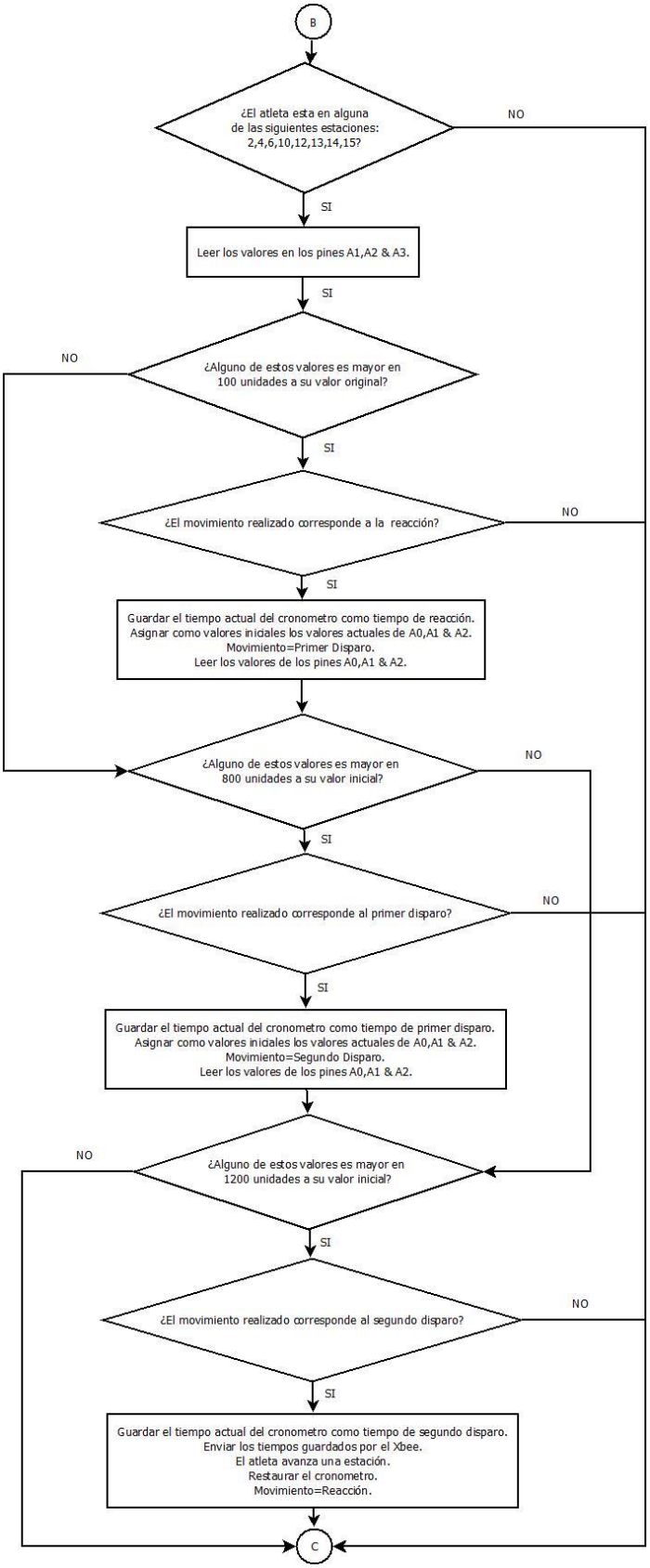
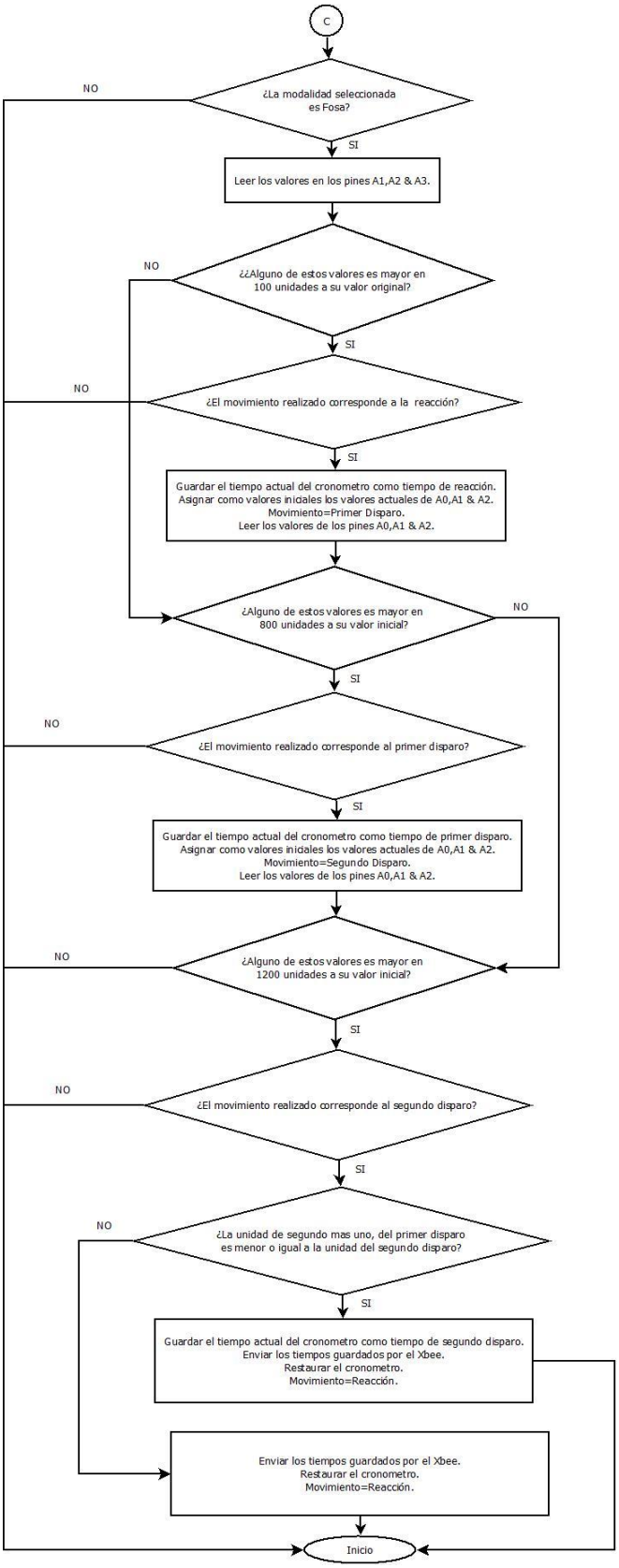


Figura 44. Diagrama de Flujo Modalidad Fosa.



3. Interfaz Gráfica los siguientes diagramas corresponden a la lógica utilizada para implementar la interfaz gráfica.

Figura 45. Diagrama de Flujo Interfaz Parte 1.

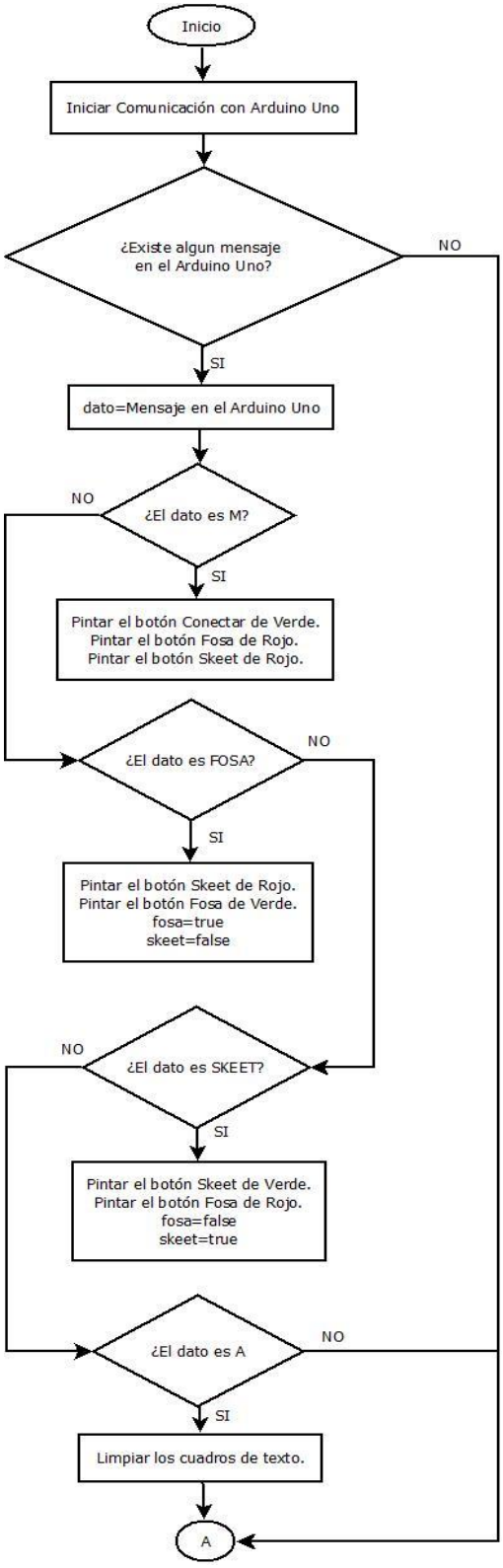
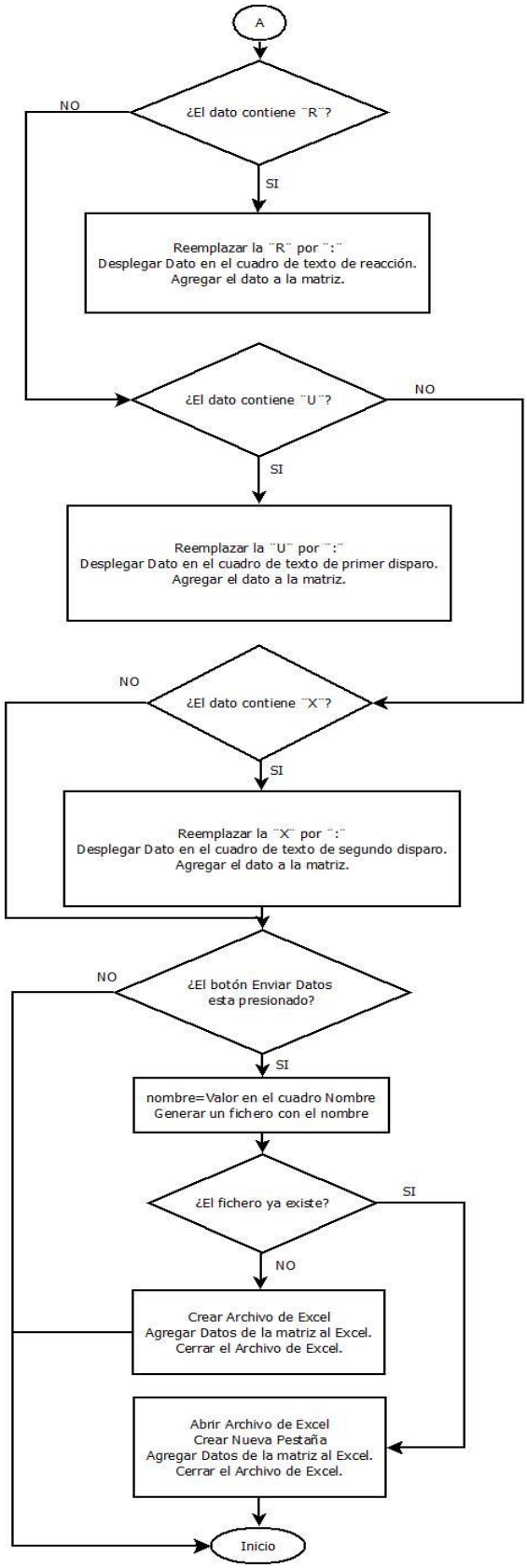


Figura 46. Diagrama de Flujo Interfaz Gráfica Parte 2.



B. PCB

A continuación, se encuentra el esquemático del PCB, así como el diseño realizado en la capa inferior del mismo.

Figura 47. Diseño PCB Bottom Layer.

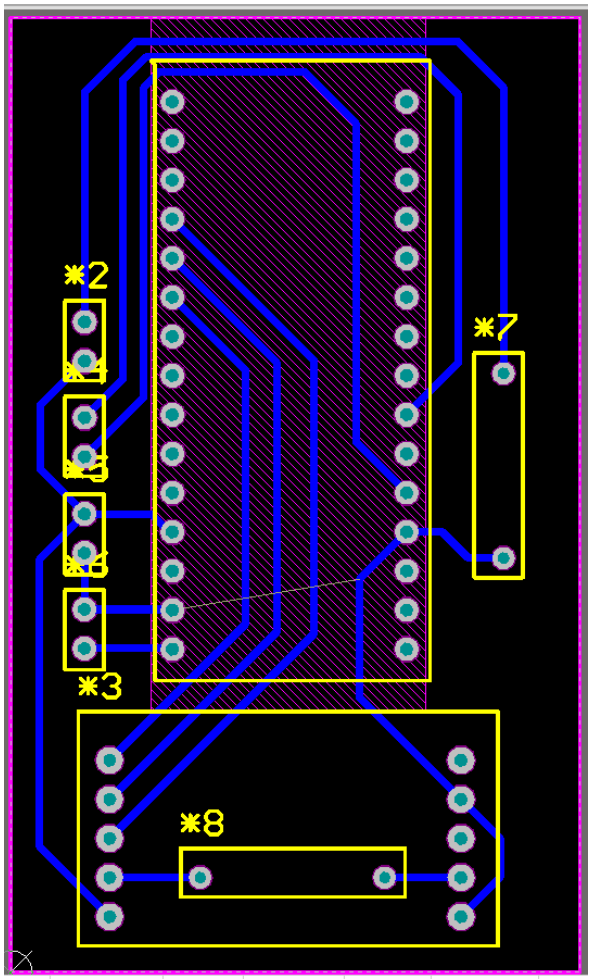
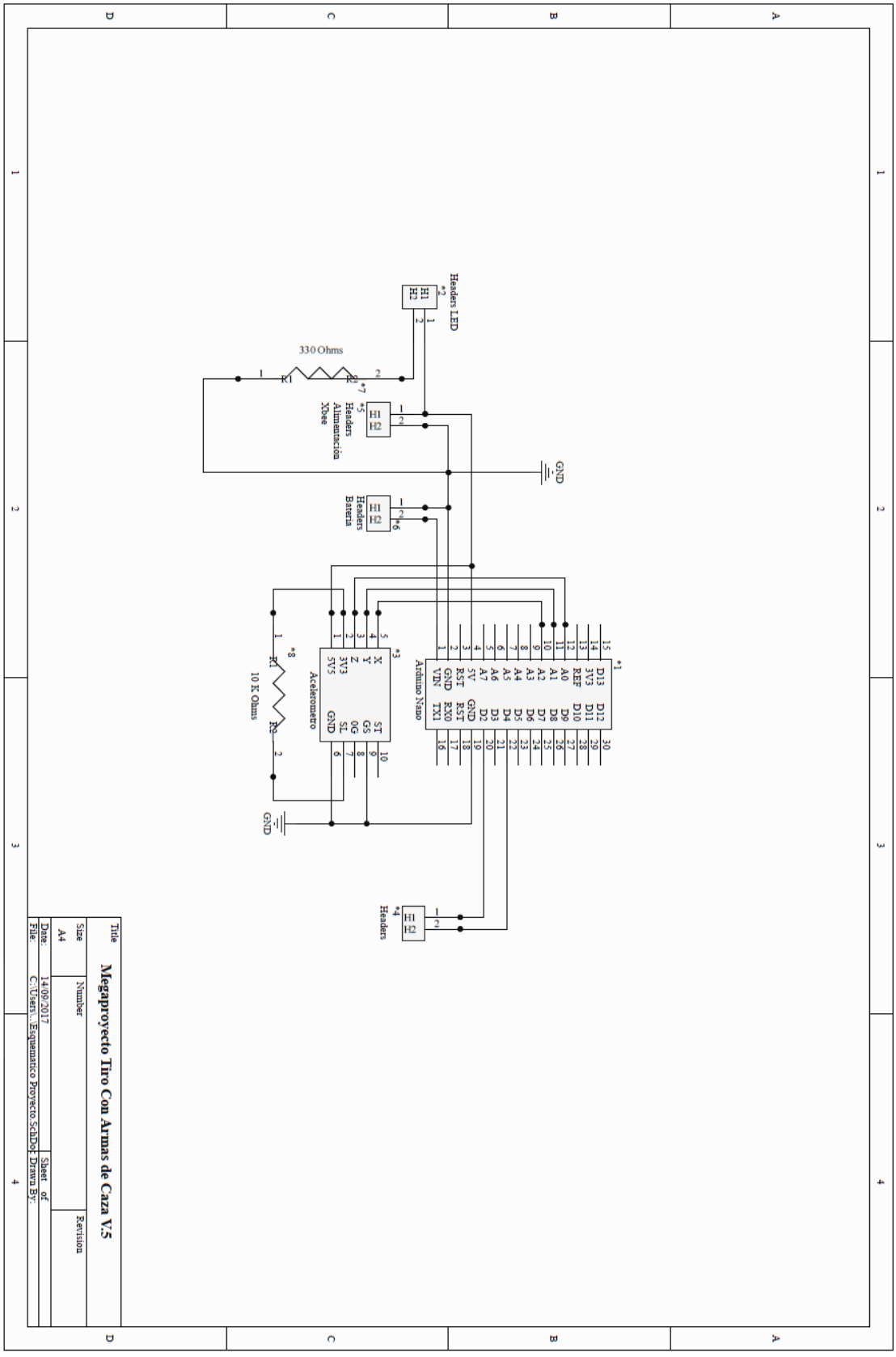


Figura 48. Esquemático PCB.



C. Archivos de Excel

1. Juan Ramón Schaeffer En el siguiente enlace se encuentra el archivo de Excel del atleta Juan Ramón Schaeffer para su consulta.
   - https://www.dropbox.com/s/x94ttsor9tz1zqt/Juan_Ramon.xls?dl=0

2. Danny Brol En el siguiente enlace se encuentra el archivo de Excel del atleta Danny Brol para su consulta.
   - https://www.dropbox.com/s/dzq2aew1kd64z1a/Danny_Brol.xls?dl=0

D. Códigos de Programación.

1. Módulos de Comunicación A continuación encuentra el código utilizado para realizar las pruebas con los 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);
}
```

2. Maquina A continuación encuentra el código desarrollado para controlar los lanzamientos de los discos en las modalidades fosas y skeet, este código se cargó en el Arduino Uno.

```
#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;
}
```

3. Escopeta A continuación, se encuentra el código desarrollado para la detección de los tiempos de reacción, primer y segundo disparo de las modalidades Fosa y Skeet.

```
#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();
             }
          }
     }
   }
}
```

4. Interfaz Gráfica. A continuación, encuentra el código desarrollado para el despliegue y almacenamiento de datos de los atletas de tiro.

```
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.htm
l
         */
        //</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

}
```