UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TRABAJO FIN DE GRADO

Grado en Ingeniería Electrónica Industrial y Automática

DESARROLLO DE UN GUANTE CON SENSORES PARA LA PRÁCTICA DEL BILLAR

Memoria y Anexos

Autor: Carlos Gil Delgado
Director: Javier Farreres
Co-Director: Sebastián Tornil

Convocatoria: Junio 2019

Resum

Aquest treball es basa en crear un dispositiu per a la millora de la tècnica del billar sense afectar al seu joc mitjançant un dispositiu integrat en un guant. S'ha desenvolupat un dispositiu capaç de mesurar la pressió exercida sobre el tac emprant una tela amb reactància, la rotació, acceleració i desviació de la mà dominant emprant un acceleròmetre. Es transmeten aquestes dades mitjançant Bluetooth a l'ordinador on a través d'un programa de suport es emmagatzem i es visualitzen els errors i les possibles millores sobre la tècnica de forma visual. Per a aquesta millora es comparen els mesuraments de cada usuari amb els paràmetres d'un professional, permetent de forma fàcil corregir els errors que posseeixin mitjançant l'ús d'imatges.

Resumen

Este trabajo se basa en crear un dispositivo para la mejora de la técnica del billar sin afectar a su juego mediante un dispositivo integrado en un guante. Se ha desarrollado un dispositivo capaz de medir la presión ejercida sobre el taco empleando una tela con reactancia, la rotación, aceleración y desviación de la mano dominante empleando un acelerómetro. Se transmiten dichos datos mediante Bluetooth al ordenador donde a través de un software de apoyo se guardan y se visualizan los errores y las posibles mejoras sobre la técnica de forma visual. Para dicha mejora se comparan las mediciones de cada usuario con los parámetros de un profesional, permitiendo de forma fácil corregir los errores que se posean mediante el empleo de imágenes.

Abstract

This work is focus on create a device to improve the billar technique without affect to the play using a device integred in a glove. It is created a technological tool that is able to measure the pressure on the cue using a fabric with reactance, the rotation, acceleration and desviation of the dominant hand using an accelerometer. The measures are send by bluetooth to the computer, where with a software it is possible save them and it shows the user's mistakes and the posible improves of his technique using a visual form. To get this improve the software compares the user's measures with the proffesional's parameters, allowing in a easy way correct the user's mistakes using images.

Agradecimientos

El autor agradece, primeramente, al ingeniero Javier Farreres por la asignación y la oportunidad de realizar el trabajo, el cual posee varios aspectos interesantes a desarrollar. En segundo lugar, a la Universidad Politécnica de Cataluña (UPC) – Escuela de Ingeniería de Barcelona Este (EEBE), Barcelona, España, por brindar las herramientas y los conocimientos para el desarrollo de dicho proyecto aportados a lo largo de todo el grado.

Como mención especial se debe destacar el personal del grupo FabLab de la EEBE gracias a los cuales se pudo realizar una impresión 3D liviana y resistente empleando una de las impresoras 3D que posee nuestra facultad. Otro agradecimiento se dirige hacia el grupo de costureras de la cadena " La Yaya Costurera " las cuales han tenido la paciencia y dedicación de ayudar en los aspectos del guante a lo largo de todo el desarrollo del dispositivo.

Por último y no menos importante, el autor agradece el apoyo incondicional y la confianza aportada por sus padres a lo largo de estos años lejos de su domicilio con el objetivo de graduarse como ingeniero electrónico en una de las mejores universidades de España.

Índice

1. Prefacio

1.1. Origen del trabajo

El billar es un deporte que abarca un público tan amplio desde cualquier persona sin conocimientos técnicos sobre él que lo emplea como un medio de entretenimiento, hasta profesionales que dedican su jornada completa a dicho deporte, incluyendo el término medio que acoge a semiprofesionales que compaginan su jornada laboral con dicho deporte, los cuales poseen una técnica superior al resto de la población que lo emplea como entretenimiento.

El caso del profesor Javier Farreres es el último nombrado y gracias a sus conocimientos de ingeniería y del billar propuso compaginarlos en un dispositivo que ayude a dicha práctica a nivel profesional. En la convocatoria de 2018 el alumno Luis Felipe realizó una primera versión de dicho trabajo que poseía diferentes características y medios empleados, dicho proyecto medía la presión a través del sensor de presión FSR406, el montaje se realizó en una Mini Board, se transmitían los datos mediante cable y el software estaba desarrollado en Matlab.

1.2. Requerimientos previos

Para la realización de dicho prototipo se necesitan conocimientos que abarcan desde lenguajes de programación para el entorno del software, hasta circuitos y electrónica para la creación del prototipo físico y su conexionado, la mayor parte de estos se han adquirido a lo largo del grado y el resto han sido aprendidos mediante aprendizaje autónomo.

2. Introducción

En este capítulo se explica el funcionamiento del deporte del billar, los factores más importantes de su técnica, los dispositivos que se encuentran en el mercado compartiendo el mismo objetivo que el prototipo a desarrollar. También se realiza el análisis de requerimientos del proyecto y la organización de tareas a realizar para su desarrollo, estableciendo los periodos de cada etapa. Por último se introducen aspectos tales como el entorno de programación " Python ", las tarjetas " Arduino " y el empleo de sensores a la hora de medir parámetros físicos.

2.1. El billar y los factores más importantes de su técnica

Según la RAE [1] el billar es un " juego de destreza que se ejecuta impulsando con tacos bolas de marfil, o de otro material semejante, en una mesa forrada de paño, rodeada de barandas elásticas y con troneras o sin ellas ".

La destreza de los jugadores es asociada a su técnica en el desempeño del juego, los factores más importantes a la hora de realizar correctamente la ejecución del ataque sobre la bola son tales como la presión de la mano dominante, la desviación, la rotación y la aceleración de la misma a la hora de realizar la tacada.

El objetivo principal del dispositivo es la mejora de dichos aspectos de la técnica [2] por lo que se deben especificar los aspectos óptimos que se desarrollan a continuación. El taco debe sostenerse con la mano dominante (*grip*) y la presión ejercida debe ser firme, pero no excesiva, el movimiento que debe realizar tiene que mantenerse en línea recta sobre la bola a tacar y el destino deseado a impactar, evitando cualquier desviación tanto si es en el eje horizontal como vertical. La muñeca no debe rotar haciendo que el taco gire sobre sí mismo y debe realizar el movimiento de ataque con una aceleración adecuada siguiendo una progresión de mayor a menor. Por último, los pies deben estar situados a la altura de los hombros, estos aspectos se observan en la figura 2.1.

Figura 2.1. Variables a medir en el proyecto [3].

2.1.1. Dispositivos del mercado

A modo comparativo se explican algunos dispositivos de los tantos que existen en el mercado que poseen el mismo objetivo que el proyecto a desarrollar, la mejora de la técnica del billar. Dichos dispositivos abarcan un gran abanico incluyendo precios bajos hasta altos y dispositivos que son muy intrusivos hasta los que casi no lo son.

Uno de los analizados es el dispositivo denominado " Cue Action Trainer " [4], el cual se observa en las imágenes de la figura 2.2. y consiste en un soporte que se sitúa en las mesa y en el cual se introduce el taco y permite realizar una tacada perfectamente alineada y con una velocidad adecuada gracias a un sistema de dos agarres pudiendo incluir unas bolas indicadoras sobre dichos soportes tal y como se muestra en la imagen.

Figura 2.2. Dispositivo Cue Action Trainer en dos de sus posibles usos [4].

Aunque mejora la técnica del usuario, es muy intrusivo debido a sus dimensiones y espacio empleado en su uso, por lo que no se puede incluir en el juego habitual. Otro dispositivo que sí es menos intrusivo es el " QMD3 Training Tool " [5], el cual consiste en una cámara con sensores que se instala en el taco y envía al móvil (con una aplicación gratuita) la velocidad y desviación del movimiento realizado de forma gráfica y fácil de entender. Dicho dispositivo y su comportamiento se pueden observar en las imágenes de la figura 2.3.

Figura 2.3. Aspecto y funcionamiento del dispositivo QMD3 Training Tool [5].

Dicho dispositivo se debe destacar porque emplea unos conceptos similares a los desarrollados por el prototipo creado en este proyecto, la diferencia es que el prototipo de este proyecto se posee más prestaciones, ya que abarca parámetros que el QMD3 no mide tales como la presión de la mano dominante sobre el taco y la rotación sobre el eje del movimiento.

2.2. Análisis de requerimientos

2.2.1. Funcionales

- Medir la presión ejercida por la mano dominante sobre el taco mediante telas con reactancia.
- Medir la aceleración, rotación y desviación de la mano empleando un giroscopio.
- Posibilidad de comenzar y detener mediciones por parte del usuario.
- Visualización de los errores del usuario mediante imágenes.
- Permitir observar errores del historial de la jugada guardada en las imágenes.
- Acceder ventana de gráficos de aceleración, velocidad y posición del historial.

2.2.2. No funcionales

- El lenguaje de programación debe ser Python.
- Posibilidad de modificar el programa introducido en la tarjeta Arduino.
- Tiempo de respuesta rápido debido a la velocidad de la tacada.
- Tolerancia de desviación y rotación permitida dentro de unos márgenes establecidos.
- Enviar los datos al ordenador vía Bluetooth.
- Minimizar el peso y tamaño para no interferir en el juego habitual del usuario.

2.3. Alcance del trabajo

El proyecto busca, al ser finalizado, crear un prototipo totalmente funcional para la persona que practique billar de forma profesional. Se pretende crear una herramienta que brinde la ayuda suficiente para que el usuario pueda identificar errores durante la práctica de forma gráfica y sencilla y de esta forma poder corregirlos.

2.3.1. Secuencia de tareas

- **Asignación del proyecto**

La idea del proyecto fue desarrollada por el profesor y jugador de billar Javier Farreres, extrayendo las experiencias obtenidas de su anterior versión y se identifican las posibles mejoras tanto de hardware como de software.

- **Planificación**

Se hace un planteamiento inicial de las tareas a realizar, el tiempo que abarcarán y los materiales que se necesiten.

Una vez realizado esto se realiza un diagrama de Gantt, el cual se muestra en el apartado 2.2.2, se muestran las fechas deseada de inicio y culminación de las mismas.

- **Diseño**

  - Diseño de prototipo: Gracias a los conocimientos de electrónica adquiridos a lo largo del grado y teniendo en cuenta el aspecto de no interferir en la práctica del billar se ha diseñado el montaje que posea menores dimensiones.

  - Evaluación de materiales: Se realiza una exhaustiva búsqueda en internet de los diferentes sensores para cada variable, con el objetivo de encontrar los adecuados al proyecto. Se piden con la suficiente antelación y se recopila información acerca de su conexionado, especificaciones, limitaciones y su correcto funcionamiento.

También se escoge la plataforma que abarque el software de la aplicación que ayuda al funcionamiento del prototipo, al final se determina que el programa ideal es uno de acceso libre y de fácil entendimiento tal y como es el lenguaje " Python ".

- Cálculos del circuito: Teniendo en cuenta el diseño anteriormente realizado y sumado a las especificaciones de los materiales previamente evaluados, se realizan los distintos cálculos que permitan una posterior construcción del prototipo de manera eficaz y sin errores de montaje.

- **Construcción**:

  - Hardware: Una vez adquiridos los materiales y componentes necesarios se realiza el montaje y la soldadura del diseño previamente realizado. Este proceso se realiza con sumo cuidado y siguiendo las especificaciones de todos los datasheets. Se realizan pruebas con cada componente por separado previo a su incorporación al conjunto, con el objetivo de comprobar su correcto funcionamiento y no poner en peligro la integridad del conjunto.

    Se estudian las normativas correspondientes con las normativas que abarcan la electrónica y su comercialización, ya que se recalca el objetivo de su posterior comercialización.

  - Software: Siendo el intermediario entre el prototipo físico y el usuario, el software es igual de importante que el hardware, el objetivo principal de su creación es el fácil entendimiento de forma visual de cuáles son los errores cometidos durante la ejecución del ejercicio y como rectificarlos.

    Dicho algoritmo recopila diferentes aspectos del juego desde el momento del inicio de la jugada hasta su finalización, los guarda y los representa de tal forma que cualquier persona con o sin conocimientos del programa pueda entender los resultados.

- **Perfeccionamiento**:

Una vez realizado el primer prototipo del proyecto se realizan una serie de modificaciones, tales como cambiar la protoboard por una placa de topos con la finalidad de reducir las dimensiones y el peso del dispositivo. El objetivo de este proceso es reducir las posibles interferencias o molestias que puedan modificar la forma de juego del usuario.

- **Registro de señales**:

Desde el origen del trabajo hasta su cúspide se declaran una serie de variables a estudiar, tales como la presión ejercida por la mano dominante sobre taco, la desviación, la rotación y la aceleración de esta a lo largo del proceso de ataque. Cuando se ha desarrollado el prototipo final se realizan las lecturas de dichas señales en el caso de un profesional de dicho deporte con la finalidad de poder usarlas como medio de comparación para la mejora de los usuarios.

- **Revisión y mejoras**:

Gracias a los conocimientos, experiencia y la vocación del director del proyecto respecto al proyecto ha sido muy fácil realizar mejoras y perfeccionamiento de las ideas originales. Dicho profesor ha estado revisando de forma periódica tanto la memoria como el trabajo físico, brindando un total apoyo y dirección con la finalidad de culminar el proyecto de la forma más óptima.

- **Presupuesto**:

Se estudia la inversión inicial para el desarrollo del prototipo que se necesita, contando las horas de mano de obra y los materiales y recursos empleados. Una vez realizado esto se contacta con empresas para realizar un presupuesto de producción a media escala y así determinando el punto de equilibrio.

- **Redacción final**:

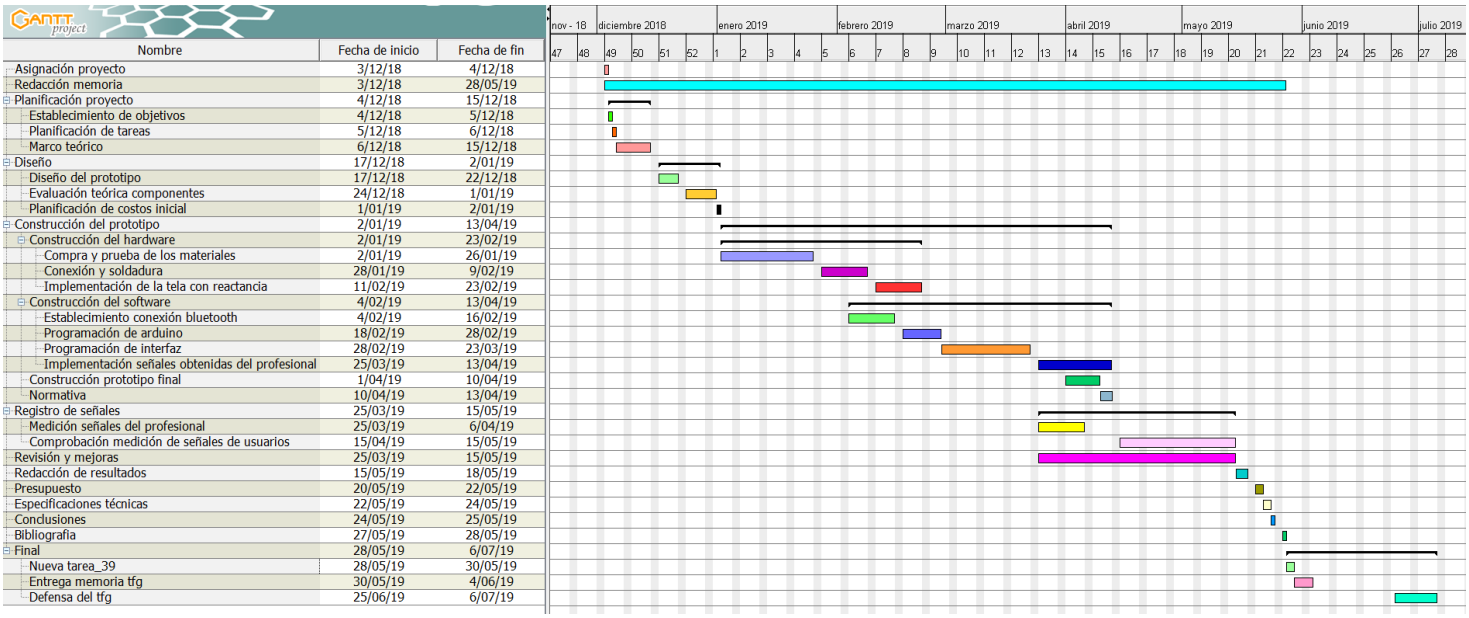
Una vez obtenido las mediciones del dispositivo final se añaden las conclusiones alcanzadas al presente documento. Se determina si cumple con los objetivos establecidos inicialmente, su funcionamiento actual y sus posibles mejoras a llevar a cabo a posterior.

- **Entrega final**:

Se realiza la revisión final del proyecto tanto de la memoria como del prototipo y se entrega dicho documento para su posterior evaluación. Se prepara una presentación que se lleva a cabo delante de un tribunal que califica los diferentes aspectos del proyecto, incluyendo errores y mejoras que pueda tener el dispositivo, finalizando de esta forma el proyecto final de grado.

2.3.2. Planificación del proyecto

A continuación se adjunta el diagrama de Gantt en la figura 2.2. realizado al comienzo del proyecto, en el cual se establecen todas las tareas y sus duraciones con el fin de culminar en el plazo de entrega establecido por la universidad.



Nombre	Fecha de inicio	Fecha de fin
Asignación proyecto	3/12/18	4/12/18
Redacción memoria	3/12/18	28/05/19
Planificación proyecto	4/12/18	15/12/18
Establecimiento de objetivos	4/12/18	5/12/18
Planificación de tareas	5/12/18	6/12/18
Marco teórico	6/12/18	15/12/18
Diseño	17/12/18	2/01/19
Diseño del prototipo	17/12/18	22/12/18
Evaluación teórica componentes	24/12/18	1/01/19
Planificación de costos inicial	1/01/19	2/01/19
Construcción del prototipo	2/01/19	13/04/19
Construcción del hardware	2/01/19	23/02/19
Compra y prueba de los materiales	2/01/19	26/01/19
Conexión y soldadura	28/01/19	9/02/19
Implementación de la tela con reactancia	11/02/19	23/02/19
Construcción del software	4/02/19	13/04/19
Establecimiento conexión bluetooth	4/02/19	16/02/19
Programación de arduino	18/02/19	28/02/19
Programación de interfaz	28/02/19	23/03/19
Implementación señales obtenidas del profesional	25/03/19	13/04/19
Construcción prototipo final	1/04/19	10/04/19
Normativa	10/04/19	13/04/19
Registro de señales	25/03/19	15/05/19
Medición señales del profesional	25/03/19	6/04/19
Comprobación medición de señales de usuarios	15/04/19	15/05/19
Revisión y mejoras	25/03/19	15/05/19
Redacción de resultados	15/05/19	18/05/19
Presupuesto	20/05/19	22/05/19
Especificaciones técnicas	22/05/19	24/05/19
Conclusiones	24/05/19	25/05/19
Bibliografía	27/05/19	28/05/19
Final	28/05/19	6/07/19
Nueva tarea_39	28/05/19	30/05/19
Entrega memoria tfg	30/05/19	4/06/19
Defensa del tfg	25/06/19	6/07/19

Figura 2.2. Diagrama de Gantt del proyecto.

2.4. Lenguaje Python

Tal y como se desarrolla en el artículo " El lenguaje de programación Python " de la Universidad de Holguín, Cuba [6], el software libre se ha convertido en uno de los movimientos tecnológicos de mayor auge en el siglo XXI, para su desarrollo ha sido necesario contar con un grupo de herramientas que hagan óptima su utilización y sean fáciles de aprender.

Python fue creado por Guido van Rossum, un programador holandés a finales de los 80 y principio de los 90 cuando se encontraba trabajando en el sistema Operativo Amoeba. Primariamente se concibe para manejar excepciones y tener interfaces con Amoeba como sucesor del lenguaje ABC. De todas las características que posee Python, una de las más importantes es su capacidad de reutilizar código escrito en los lenguajes C y C++. Existen mecanismos que hacen muy sencilla la tarea de envolver funciones y clases hechas en estos lenguajes, entre los que se encuentran Boost, Sip y Shiboken. La importancia de esta integración es relevante, ya que las bases de código en lenguajes como C y C++

son las más grandes disponibles por el software libre hoy en día, y permiten no tener que duplicar código ya existente.

Python es liberado bajo una licencia propia llamada Python Licence que ha sido certificada por el movimiento Open Source, y es compatible con la GPL (GNU Public Licence) de la Free Software Foundation (Fundación del Software Libre). Google, por ejemplo, siendo uno de los gigantes de la informática e Internet, ha empleado el Python para construir una gran parte de sus algoritmos de búsqueda de webs.

Debido a todo ello se ha escogido Python como lenguaje para la programación del entorno del software que abarca dicho proyecto por las razones ya nombradas, su fácil entendimiento y gran compatibilidad, destacando su libre acceso.

2.5. Sensores para la obtención mediciones

La importancia que poseen los sensores a la hora de obtención mediciones de variables físicas es desarrollada por el doctor ingeniero industrial Ramón Pallás Areny de la UPC [7] en su artículo " Sensores y acondicionadores de señal ", en el cual recalca que las aplicaciones de la electrónica no serían posibles sin la ayuda de los sensores integrados en ellas.

Los sensores han experimentado un gran cambio en su uso desde 1989, en sus orígenes eran empleados en aplicaciones de control de procesos continuos (temperatura, presión, caudal y nivel), y en la actualidad abarcan el mercado de la robótica, automóviles, control de niveles de contaminación, electrodomésticos, etc.

La utilización de sensores es indispensable en los procesos de automatización de diversos sectores, incluso los equipos de gestión de datos incorporan internamente sensores para comprobar su correcto funcionamiento. Los sensores han pasado a ser componentes de un subsistema controlado por los microprocesadores y microcontroladores, pero a pesar de dicho avance y cambio, apenas han variado los sensores clásicos tal y como se conocían.

En el caso de este proyecto se emplea una tela con capacidad de variar su resistencia interna a medida que se ejerce presión sobre ella y un acelerómetro y un giroscopio que se encuentran integrados en un módulo y es controlado mediante un microprocesador.

2.6. Arduino

Tal y como se muestra en la página web oficial de Arduino [8] se narran diferentes aspectos de la estructura de dicha empresa, es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software flexible y fácil de utilizar para los creadores y desarrolladores.

El proyecto tuvo su origen en 2003, cuando varios estudiantes del Instituto de Diseño Interactivo de Ivrea, Italia, con el fin de facilitar el acceso y uso de la electrónica y programación. Tenían como objetivo que los estudiantes de electrónica tuviesen una alternativa más económica a las populares BASIC Stamp, unas placas que por aquel entonces valían más de cien dólares, y que no todos se podían permitir.

El resultado del proyecto fue Arduino, una placa con todos los elementos necesarios para conectar periféricos a las entradas y salidas de un microcontrolador, y que puede ser programada tanto en Windows como macOS y GNU/Linux.

El Arduino es una placa basada en un microcontrolador ATMEL. Los cuales se pueden programar a través del programa gratuito Arduino IDE, dichas instrucciones permiten crear programas que interactúan con los circuitos de la placa y los periféricos conectados a ella.

Existen un gran número de placas Arduino, tales como Arduino Yún, Leonardo, Due, etc. En este proyecto se emplea una tarjeta Arduino Nano debido al objetivo de poseer las menores dimensiones y peso posibles, dicha tarjeta ofrece las prestaciones necesarias para desenvolver la totalidad del proyecto.

3. Diseño del proyecto

Todo proceso de creación de un dispositivo debe pasar por una fase previa de diseño del hardware y software, en la primera de estas se estudian las variables a medir y los materiales a emplear realizándose la comparación entre las diferentes opciones del mercado. En la segunda se estudia la mejor forma de visualizar dichas variables, se crea un diagrama de las ventanas del programa a desarrollar y un algoritmo de este.

3.1. Diseño del hardware

3.1.1. Variables a medir con el dispositivo

Tal y como se desarrolla en la web de " Aprender billar " [2] : " la empuñadura se refiere a momento de agarre del taco con la mano dominante en la parte trasera del taco de billar." Aunque dicho proceso parece simple existen una serie de factores que son determinantes en la técnica de dicho deporte y es necesario un estudio sobre ellos y su correcta ejecución.

El agarre se debe realizar con el índice y mayor, también se puede agregar el anular, el pulgar se sitúa en el otro lado cerrando con el índice. Nunca se debe poner el pulgar por encima del taco. La presión de la empuñadura tiene que ejercerse de forma frontal a la mano, mientras que los dedos traseros deben abrirse a medida que se aleja el taco de la mesa y luego vuelven a su posición original, tal y como se muestra en la figura 3.1.

Figura 3.1. Colocación de los dedos para un correcto agarre [9].

Debido a este agarre especificado se han situado las zonas de medición en dos partes concretas de la mano, la ubicada en intermedio del índice y el pulgar y el extremo contrario de la mano, formado por el meñique y la zona correspondiente de la palma de la mano.

Cuando se realice el movimiento de golpeo nunca se debe rotar la muñeca hacia el cuerpo, y la mano debe encontrarse debajo del codo. La empuñadura debe ser confortable y relajada, no muy firme ni rígida, pero tampoco muy floja. Para tiros que se requieren mucha fuerza como el saque o break, la empuñadura debe estar más firme de lo normal pero no rígida. La muñeca debe estar bastante relajada y colgando hacia abajo, con los nudillos hacia el suelo.

Gracias a los conocimientos aportados por el director del trabajo y la información recopilada en internet se han determinado cuales son las variables más importantes en el desarrollo de la práctica del billar y son sensadas por el dispositivo. A continuación se enumeran dichas variables y se visualizan en la figura 3.2.

1. Presión de la mano dominante sobre el taco.

2. Rotación de la muñeca de la mano dominante durante el ataque.

3. Desviación de la mano dominante durante el ataque.

4. Aceleración de ataque.

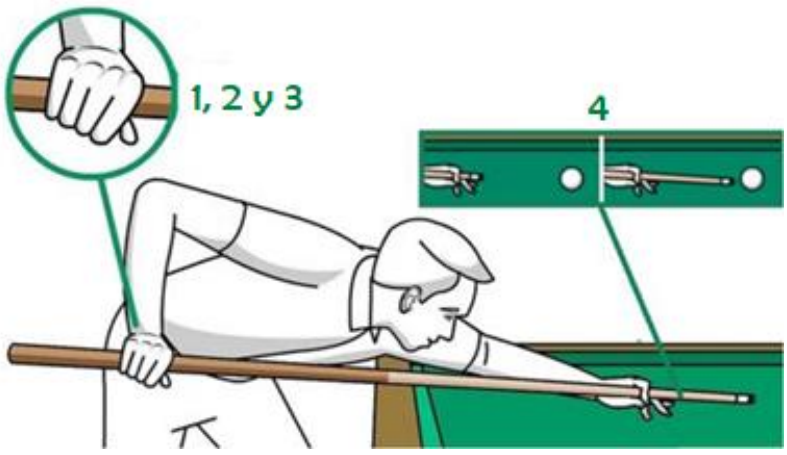


Figura 3.2. Variables a medir con el dispositivo [10].

3.1.2. Presión mano dominante

Para la medición de la presión en la mano dominante hay muchos sensores en el mercado, pero la elección se centra entre dos principalmente, los cuales son nombrados y desarrollados a continuación.

1. Sensor FSR406.

2. Tela con reactancia

- **Sensor de fuerza FSR406**

Dicho sensor permite medir la presión física que se ejerce sobre él, este procedimiento se realiza mediante su alimentación y la modificación de su resistencia interna a medida que se ejerce presión sobre él, lo que genera diferentes voltajes en su salida.

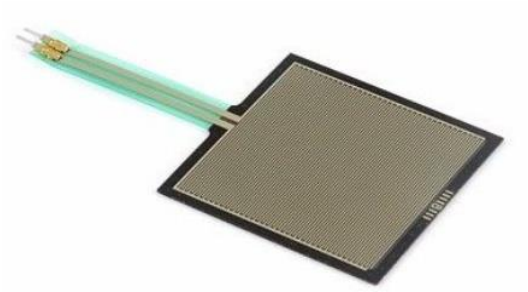


Figura 3.3. Sensor de fuerza FSR406 [11].

La grafica adjuntada a continuación muestra el comportamiento de dicho dispositivo a medida que se ejerce fuerza sobre él., cuando no existe ninguna fuerza dicho sensor actúa como una resistencia "infinita".

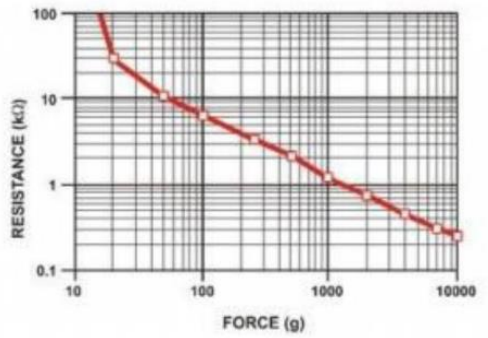


Figura 3.4. Cambio de resistencia del sensor FSR406 [12].

- **Tela con reactancia**

Se trata de un tejido con microfibras conductoras que actúan como sensores piezoresistivos para empelar en sensores dinámicos, con el objetivo de mapear y medir la presión, la flexión, el estiramiento y torsión de su superficie.

A medida que se ejerce fuerza sobre ella se modifica su resistencia interna y conductividad, actuando como un sensor. En la figura 3.5. se adjuntan las posibles telas a escoger, fabricadas por diferentes empresas.

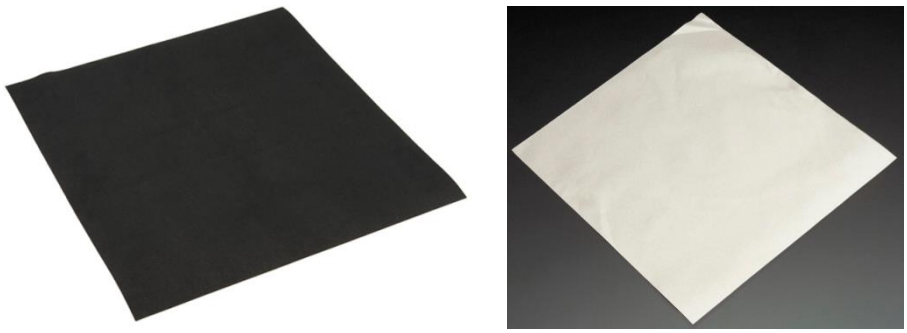


Figura 3.5. Posibles telas con reactancia, izquierda Sparkfun [13], derecha Adafruit [14].

- **Elección**

Se ha seleccionado el uso de las telas con reactancia, en concreto la fabricada por Sparkfun [13], debido a su facilidad de modificar su forma y así ajustarse a la forma del guante deseada. También se ha adquirido un hilo conductor para la transmisión de las señales desde las telas con reactancia hasta el dispositivo Arduino.

A continuación se adjunta una imagen de dicho hilo empleado en el prototipo:

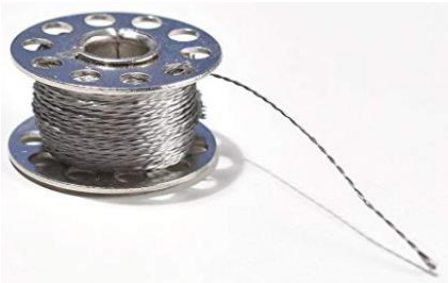


Figura 3.6. Hilo conductor desarrollado por Adafruit [15].

3.1.3. Rotación de la muñeca de la mano dominante

Teniendo como objetivo medir la rotación de la mano dominante sobre el eje de movimiento durante el proceso de ataque se estudian dos dispositivos que se nombran y desarrollan a continuación.

1. Giroscopio MPU6050.

2. Sensor infrarrojo.

- **Giroscopio MPU6050**

Este módulo de Sodial para Arduino combina un giroscopio de 3 ejes y un Procesador Digital de Movimiento (DMP) en un solo dispositivo compacto y liviano, en la figura 3.7. se adjunta una foto de dicho módulo.

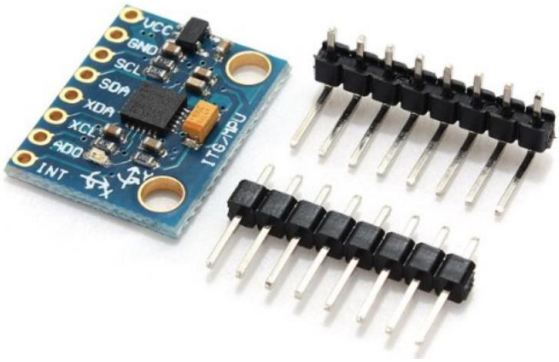
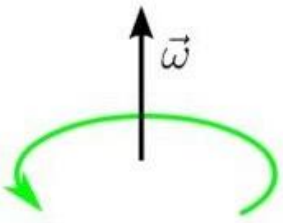


Figura 3.7. Giroscopio MPU6050 [16].

Teniendo en cuenta que la velocidad angular (ω) es la tasa de cambio del desplazamiento angular (θ) por unidad de tiempo (t), es decir, que tan rápido gira un cuerpo alrededor de su eje, tal y como se expresa en la Ec. 3.1.



$$\omega = \frac{d\theta}{dt} \qquad \text{(Ec. 3.1)}$$

Figura 3.8. Velocidad angular, ilustración y ecuación [17].

Para medir la velocidad angular el giroscopio emplea un MEMS (MicroElectroMechanical System) usando el efecto Coriolis, dónde se genera una fuerza ficticia que aparece sobre un cuerpo en movimiento cuando se encuentra en un sistema de rotación. El principio de un funcionamiento de un GVC (Giroscopio de Vibración de Coriolis) se basa en que un objeto tiende a vibrar en el mismo plano incluso si este rota, esta vibración ejerce una fuerza que es medida y así se determina la rotación del objeto. Ciertas partes del cuerpo MEMS se someten a vibración por resonancia y el efecto de la fuerza de Coriolis deforma la estructura, lo cual puede ser medido por la vibración de la capacitancia del sistema.

- **Sensor infrarrojo**

Se compone de dos partes, la primera es un emisor, el cual es un LED infrarrojo que expulsa luz que no se encuentra dentro del espectro visible y la segunda es un fotodiodo, el cual es un semiconductor que actúa como receptor y de acuerdo a su área fotosensible cambia su capacidad de regular la corriente que circula a través de él.

Figura 3.9. Sensor infrarrojo [18].

- **Elección**

Se evita el uso del sensor infrarrojo debido a su baja compatibilidad con el proyecto, ya que se producirían errores de medida debido a la distancia entre la mano dominante (emisor) y la mano de apoyo (receptor), a la susceptibilidad a la luz ambiente y a la necesidad de otros componentes tales como resistencias, timer 555 y otros, por último depende de dos puntos para la medición, situados en ambas manos, por lo que complica la programación.

En cambio, el sensor MPU6050 es un módulo para Arduino con lo que existe una fácil compatibilidad y también permite captar la rotación deseada solo necesitando la mano dominante para los registros.

3.1.4. Desviación de la mano dominante durante el ataque

- **Elección**

Tal y como se ha mencionado anteriormente, el módulo MPU6050 posee un acelerómetro de 3 ejes. Dicho dispositivo se basa en la segunda ley de Newton, la cual indica que, en un cuerpo con masa constante, la aceleración del cuerpo es proporcional a la fuerza que actúa sobre él mismo.

Los acelerómetros internamente poseen un MEMS, que de forma similar a un sistema masa - resorte permite medir la aceleración, tal y como se muestra en la figura 3.10. El movimiento acelerado sobre el conjunto se traduce a una fuerza sobre los resortes causando que uno se contraiga y otro se extienda, lo que hace que la posición relativa de la masa dentro del sensor varíe. Se ha de tener en cuenta que, a pesar de que exista movimiento, siempre el acelerómetro estará sensando la aceleración de la gravedad en su eje z.

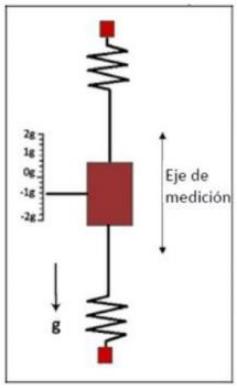


Figura 3.10. Sistema masa – resorte en el que se basa la medición del acelerómetro [17].

Teniendo en cuenta su funcionamiento y que el movimiento del ataque se realiza sobre el eje x del giroscopio MPU6050, para observar si existe desviación, es decir, que no se realiza el movimiento en línea recta, se miden las aceleraciones en los otros ejes, las cuales deberían ser nulas dentro de unos márgenes de permisión.

3.1.5. Aceleración de ataque

- **Elección**

Basándose en el mismo principio empleado para la medición de desviación, si se centra la medida en la aceleración del eje x, el cual pertenece al eje de realización de movimiento de ataque, se puede medir la aceleración del impacto en el momento de la tacada. Por lo tanto, para dicha variable se emplea el mismo módulo MPU6050 que abarca las variables de desviación y rotación de la mano dominante durante la ejecución del movimiento.

3.1.6. Placa Arduino

Para la medición de todas las variables, su procesamiento y envío se escoge una tarjeta Arduino debido a su gran compatibilidad con el resto de módulos. Dentro de todas las placas que fabrica dicha compañía se ha elegido la tarjeta Arduino Nano debido a sus pequeñas dimensiones y peso liviano, disminuyendo su efecto en el juego habitual y posee la suficiente versatilidad para poder abarcar todos los aspectos de dicho proyecto, en la figura 3.11. se adjunta una foto de dicha placa.

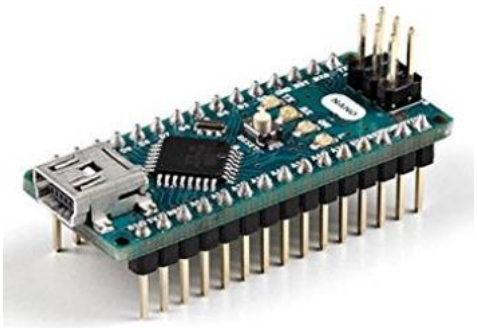


Figura 3.11. Placa Arduino Nano [19].

3.1.7. Transmisión de mediciones

Respecto a la transmisión de datos hay dos posibilidades con sus diferentes ventajas y desventajas, dichas opciones son:

1. Conexión mediante cable.

2. Conexión vía Bluetooth.

- **Elección**

Aunque la opción más fácil a nivel de montaje es utilizar un cable, se tiene en cuenta uno de los objetivos principales del proyecto, no afectar al juego habitual se ha seleccionado la conexión mediante bluetooth.

Para transmitir los datos vía Bluetooth se necesita un módulo DSD Tech HC-05 para Arduino, el cual posee una fácil conexión y tamaño compacto, el cual se puede observar en la figura 3.12.

Figura 3.12. Módulo DSD Tech HC-05 para Arduino [20].

3.1.8. Superficie para el soporte del circuito

Para realizar el montaje del circuito hay varias opciones en el mercado entre las que destacan las siguientes:

1. Protoboard.

2. Placa de topos.

- **Elección**

Aunque las protoboards tienen mucha facilidad para montar y modificar los montajes, poseen unas grandes dimensiones y un peso, incluso las MiniBoards con menores dimensiones del mercado. Dichas características afectan al juego habitual del usuario del proyecto, ya que dicho peso en la mano afecta tanto provocando molestias al jugador como distorsionando las mediciones del giroscopio.

Por lo tanto se decide la utilización de una placa de topos, en concreto una placa fabricada con fibra de vidrio, la cual posee unas dimensiones mínimas, debido a que se recorta al tamaño deseado y un peso despreciable en comparación con los módulos soldados en ella. Dicha placa se muestra en la figura 3.13.

Figura 3.13. Placa de topos fabricada con fibra de vidrio [21].

3.1.9. Diagrama de bloques del circuito

En la figura 3.14. se adjunta un diagrama de bloques que resume los dispositivos empleados y la conexión que existe entre ellos, siendo el sensor de presión las telas con reactancia cosidas al guante, los sensores de aceleración y rotación incluidos en el módulo MPU6050 cableado a la tarjeta Arduino Nano. El modulo HC-05 se encuentra conectado al puerto serie de Arduino y se conecta por Bluetooth al ordenador.

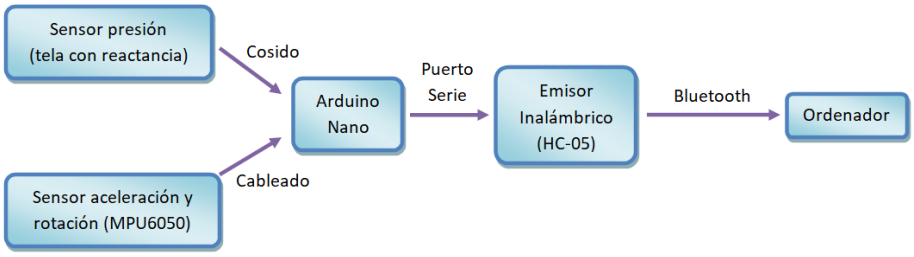


Figura 3.14. Diagrama de bloques del proyecto.

3.2. Diseño del software

Teniendo como objetivo que sea un programa de libre acceso a cualquier usuario se descartan programas con grandes prestaciones tales como Matlab, debido a que posee una licencia de pago para su empleo. Otro aspecto a estudiar es su compatibilidad con el dispositivo físico, por lo tanto se ha escogido para el desarrollo del software el lenguaje Python, en concreto el programa " PyCharm ", el cual tiene acceso gratuito a cualquier usuario que lo desee y tiene librerías tanto para Arduino como para los módulos empleados.

3.2.1. Visualización de las mediciones

El objetivo principal del programa es realizar un entorno visual que sea de fácil entendimiento para cualquier usuario, por lo tanto las variables anteriormente mencionadas deben visualizarse de forma clara y precisa. Para esa labor se emplean una seria de imágenes para la visualización de cada variable y errores producidos, a continuación se comentan los aspectos de las imágenes para cada una de las variables.

- **Presión de la mano dominante**

En la figura 3.15. se muestra la imagen empleada para la señalización de los niveles de presión de dicha mano sobre el taco durante el movimiento para un jugador diestro, para la versión de jugadores zurdos se ha invertido previamente empleando Photoshop.

Figura 3.15. Imagen mano derecha a emplear en el programa [22].

Para señalizar de forma clara y sencilla se diferencian dos zonas de agarre, una presión ejercida en la zona conjunta del pulgar e índice y la zona del meñique y el lateral de la palma correspondiente a dicha zona. Tal y como se aclara en la leyenda, los colores representan una correcta presión (verde) la cual se encuentra dentro de unos márgenes establecidos, una presión insuficiente (azul) ya que se encuentra por debajo del umbral inferior y una presión excesiva (rojo) ya que se haya por encima del umbral superior establecido. Dichas señalizaciones se pueden observar en la figura 3.16. en la cual se exponen los diferentes casos para diestros y zurdos.

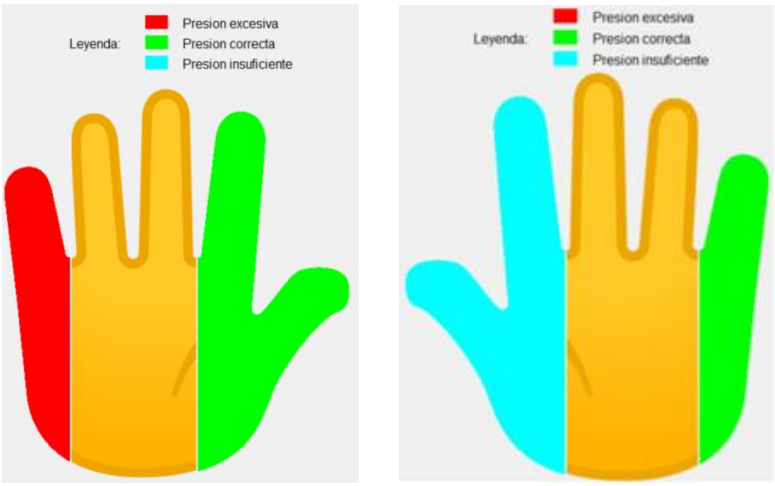


Figura 3.16. Señalización de presión correcta, insuficiente o excesiva.

- **Rotación de la muñeca de la mano dominante durante el ataque**

Uno de los fallos más comunes es la rotación de la muñeca a la hora de realizar la tacada, dicha rotación se realiza llevando la muñeca hacia el cuerpo. Se emplean dos imágenes para obtener una doble perspectiva de dicha muñeca dominante, teniendo en cuenta las versiones diestras y zurdas del usuario, dichas imágenes se adjuntan en la figura 3.17, para sus versiones zurdas se han invertido en Photshop previamente y se han guardado.

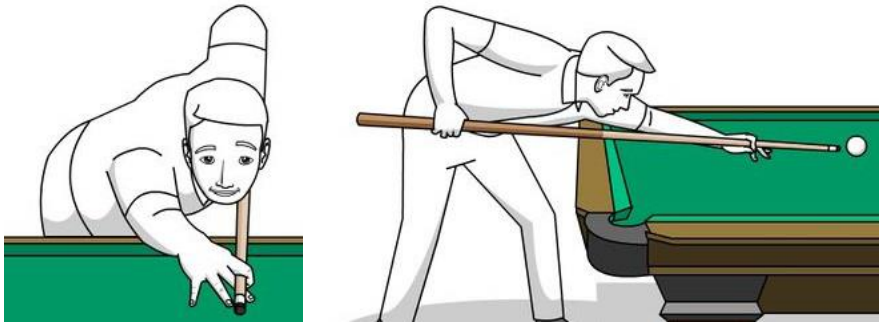


Figura 3.17. Imágenes para observar la desviación de la muñeca dominante [10].

Modificando las imágenes originales se añaden unas flechas que indican la rotación de la muñeca hacia el interior, tal y como se muestra en las imágenes de la figura 3.18.

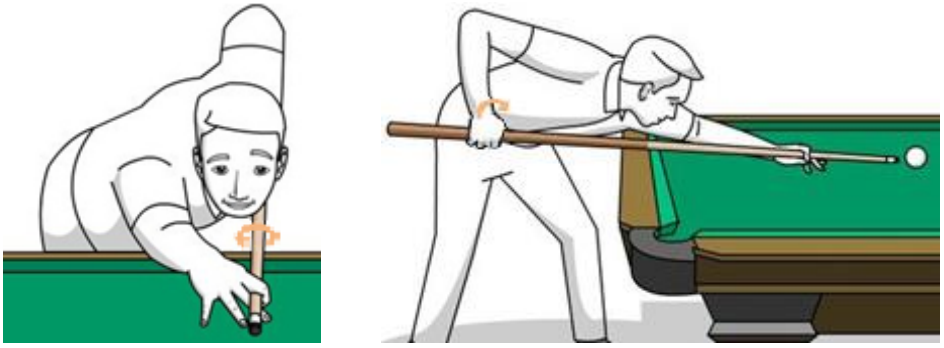
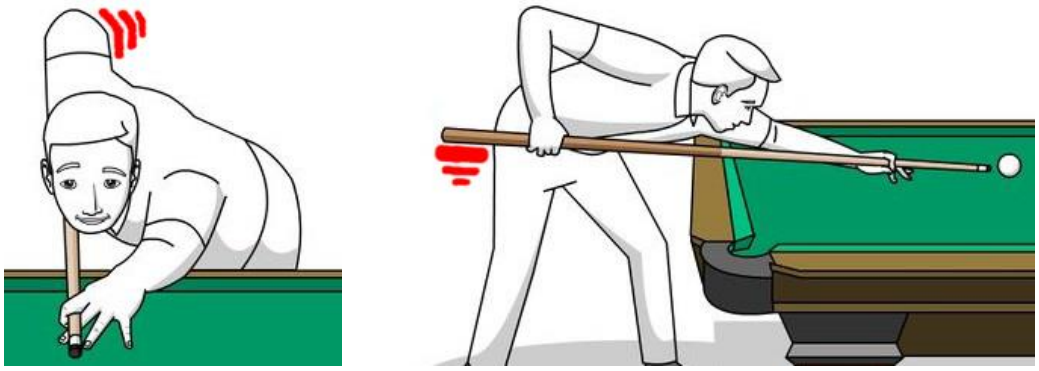


Figura 3.18. Señalización de rotación de la muñeca dominante.

- **Desviación de la mano dominante durante el ataque**

El movimiento de ataque deseado se realiza en el eje x del módulo MPU6050, por lo tanto cualquier movimiento en el eje vertical y horizontal se trata de un fallo en la técnica (dentro de unos márgenes de permisión). Para ello se emplean las imágenes de la figura 3.17. en las cuales se pueden observar de forma clara los ejes horizontal y vertical.

Para la representación de dichas desviaciones se modifican las imágenes originales, en las cuales se señaliza en rojo la dirección de la desviación del movimiento del usuario, teniendo en cuenta también si se trata de un jugador diestro o zurdo.



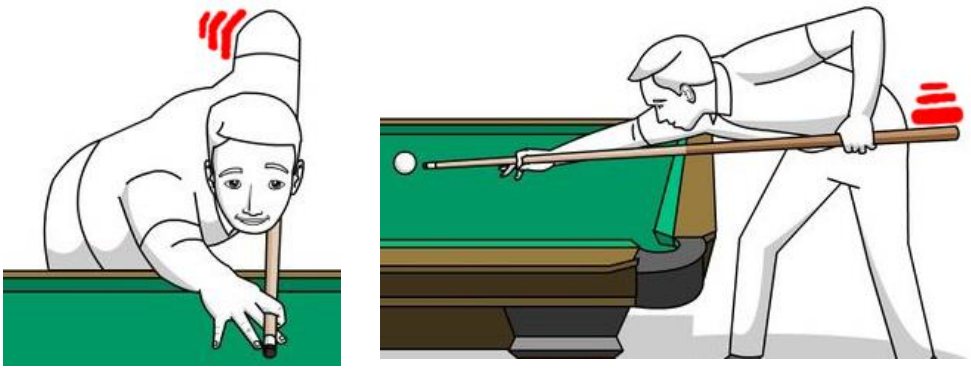


Figura 3.19. Señalización desviación durante el movimiento.

- **Aceleración de ataque**

La aceleración con la que se realiza la tacada es un factor muy importante a la hora de poseer una buena técnica de juego, a través de dicha aceleración se puede obtener la velocidad final con la que se golpea la bola. Para representar dicha velocidad se emplea una barra horizontal en la que se visualiza dicho valor, la cual se sitúa en la imagen lateral debajo de la zona de golpeo tal y como se muestra en la figura 3.20.

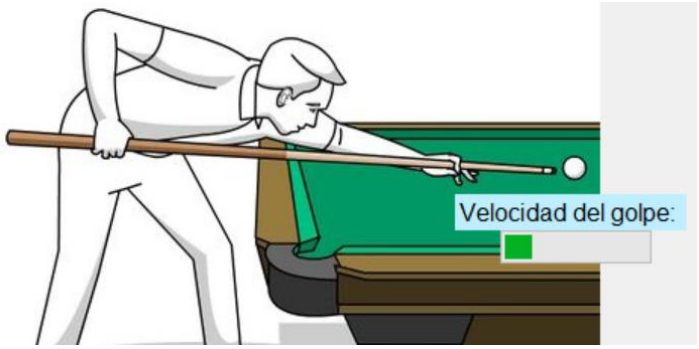


Figura 3.20. Representación de la velocidad de golpeo.

Los valores mínimo y máximo de dicha barra representativa se establecen acorde a los valores estándares que se midan experimentalmente.

3.2.2. Diagramas de la aplicación

Para la creación del entorno del software se debe realizar un diagrama de navegación de las ventanas que posea, la cual se muestra en la figura 3.21.

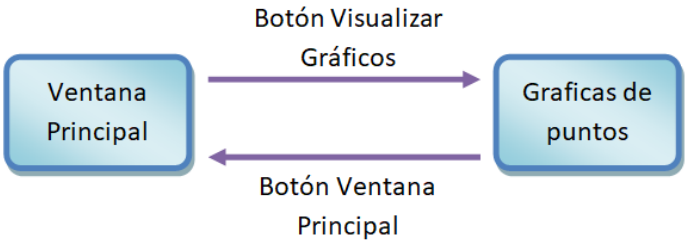


Figura 3.21. Diagrama de navegación de pantallas del programa.

Una vez realizado el diagrama de navegación entre las dos ventanas del programa se debe crear un algoritmo en el cual se base el programa en general, especificando el funcionamiento deseado tal y como se muestra a continuación.

INICIO

Se selecciona si el usuario es diestro o zurdo:

Si el usuario es zurdo se muestran las imágenes previamente invertidas.

Se inician las mediciones:

Se leen las lecturas del puerto serie y las guarda en la lista mediciones.

Se comparan los valores de las resistencias, rotación y aceleraciones de los 3 ejes con los umbrales:

Se muestran las imágenes sin errores o señalizando cuáles son los errores de forma visual.

Se guardan las variables en un listado de historial.

Se guardan la aceleración del eje x (movimiento deseado) en un listado de aceleraciones.

Se borran los datos de la lista mediciones para guardar las nuevas mediciones.

Si se detienen las mediciones:

Se puede seleccionar un momento del historial empleando un slider:

Se enseñan las imágenes de las variables con o sin errores de dicho momento.

Se puede cambiar la ventana principal con la ventana de gráfica de puntos pulsando un botón:

Se calcula la velocidad integrando el listado de aceleraciones y después se vuelve a integrar para obtener las posiciones.

Se grafican las aceleraciones, velocidades y posiciones.

FIN

4. Implementación del proyecto

En este apartado se desarrolla la implementación del diseño previamente estudiado y desarrollado, para ello se explican los conexionados de cada componente con la tarjeta Arduino y su montaje final, el cual engloba desde toda la circuitería hasta cómo alimentarla, protegerla e instalarla en los guantes. También se explica la programación que abarca el entorno visual, tanto el programa integrado en la tarjeta Arduino como en el ordenador, el cual sirve de enlace entre los usuarios y el dispositivo.

4.1. Conexionado

A continuación se analiza el conexionado de cada componente con la placa Arduino Nano, se realiza un listado de los materiales necesarios para cada tarea y se añade un esquema con una pequeña explicación de dicho montaje, por último, se engloba todo en el montaje final.

4.1.1. Telas con reactancia

A continuación se realiza un listado de los materiales necesarios para la conexión de las telas.

- **Materiales necesarios**:
    - Papel de patrón de telas.
    - Tela con reactancia fabricado por Sparkfun.
    - Hilo conductor fabricado por Adafruit.
    - Tarjeta Arduino Nano.
    - 3 resistencias 1 kΩ.
    - Cables.

El primer paso para el empleo de dicha tela es realizar un patrón en el guante con el trazo deseado, en este caso se centra la medición en los dos extremos de la mano que es donde se centra el agarre del taco. Después de realizar el recorte de la tela se necesitan los servicios de una costurera profesional debido a la dificultad de abrir el guante, coser las telas con la máquina de coser y volver a

cerrar el guante, quedando el guante con su forma original, pero con los fragmentos de tela agregados tal y como se observa en la figura 4.1.

Figura 4.1. Guantes con los fragmentos de tela con reactancia cosida.

Las telas con reactancia se conectan con la tarjeta Arduino mediante el hilo conductor fabricado por Adafruit [15] ya nombrado con anterioridad y con cables soldados a dichos hilos. Para realizar la medición se realiza un circuito basado en un divisor de tensión, en el cual se encuentran dos resistencias, una conocida (1 kΩ) y la segunda es la resistencia interna variable del fragmento de tela del guante. Dichas telas se conectan un extremo a 5 V y otro al puerto analógico de la tarjeta Arduino Nano, mientras que las resistencias conocidas se encuentran conectadas a las entradas analógicas y al terminal GND de la placa. El esquema del montaje se adjunta en la figura 4.2. en el cual se emplean diferentes colores en los cables para facilitar su entendimiento.

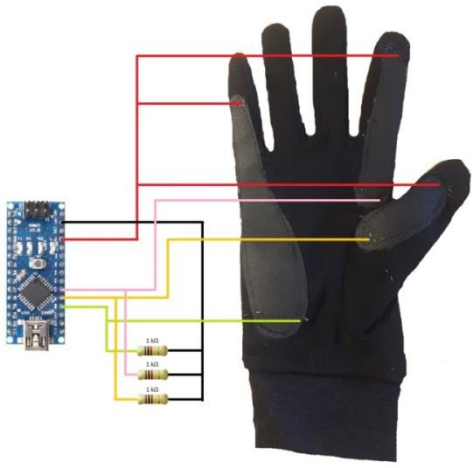


Figura 4.2. Esquema montaje Arduino Nano y telas con reactancia.

4.1.2. Giroscopio MPU6050

A continuación se hace un listado de los materiales necesarios para el conexionado del módulo MPU6050.

- Materiales necesarios:
    - Giroscopio MPU6050.
    - Tarjeta Arduino Nano.
    - Cables.

Como se ha mencionado con anterioridad el módulo MPU6050 permite medir la aceleración en los tres ejes espaciales y el ángulo de rotación en cualquiera de ellos. Para realizar la conexión de dicho módulo se deben conectar los pines de VCC y GND a sus respectivos pines de la tarjeta Arduino para alimentarlo y los pines SCL y SDA del módulo MPU6050 a las entradas analógicas A5 y A4 de la placa, tal y como se muestra en la figura 4.3.

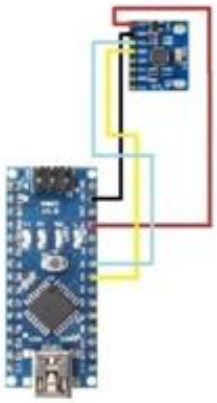


Figura 4.3. Esquema montaje Arduino Nano y módulo MPU6050.

4.1.3. Módulo HC-05

A continuación se crea una lista de los materiales necesarios para establecer las conexiones del módulo HC-05.

- Materiales necesarios:
    - Módulo Bluetooth HC-05.
    - Tarjeta Arduino Nano.
    - Cables.
    - Interruptor 1C2P.

Para realizar el conexionado de dicho módulo se debe alimentar usando los pines VCC y GND a sus respectivos terminales de la tarjeta Arduino y el pin " TXD " del HC-05 con el pin " RXD " de la tarjeta Arduino Nano y viceversa, debido a que la transmisión de los datos se realiza a través del puerto serie de la placa. Este conexionado se muestra en la figura 4.4.

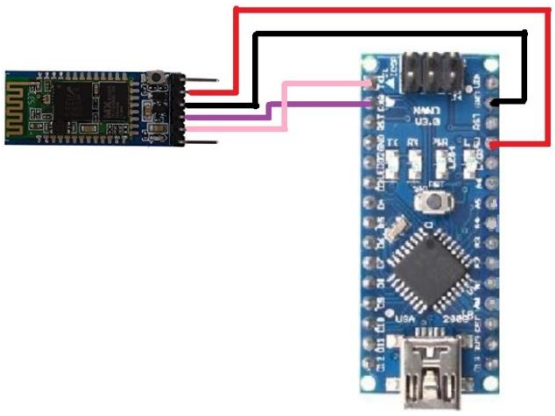


Figura 4.4. Esquema montaje Arduino Nano y módulo HC-05.

- **<u>Corrección montaje</u>**

Cuando se importa un programa desde Arduino IDE a la tarjeta Arduino se realiza a través del puerto serie, en dicho puerto se encuentra conectado al módulo HC-05, por lo tanto el montaje inicial necesita una modificación que permita conectar y desconectar dicho módulo de forma fácil y rápida para poder cargar nuevas modificaciones del programa sin tener que modificar el montaje.

Para ello se añade un interruptor en el cable que alimenta al modulo HC-05, permitiendo dos posiciones, una que habilita la importación de programas a la tarjeta Arduino a través del puerto

serie y la segunda alimenta el módulo HC-05 y transmite datos vía Bluetooth, dicho montaje se puede observar en la figura 4.5.

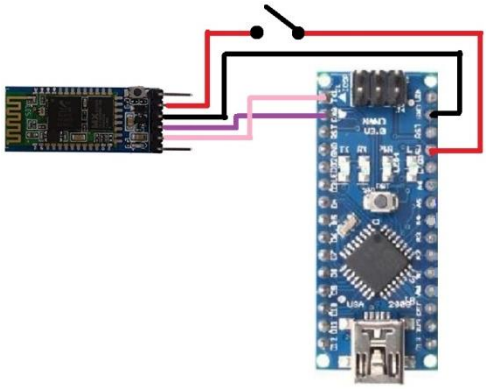


Figura 4.5. Montaje con interruptor Arduino Nano y módulo HC-05.

El interruptor seleccionado para dicha función es el conmutador de dos posiciones (1C2P) el cual es especifico para circuitos impresos que se observa en la figura 4.6.

Figura 4.6. Conmutador dos posiciones [23].

4.2. Montaje final

Una vez se ha visualizado el montaje parcial de cada módulo se deben soldar todos en la placa de topos ya recortada, con la finalidad para poseer las menores dimensiones posibles y así interferir en menor medida al juego habitual. Posteriormente se crea una carcasa que sirve tanto de protección como para facilitar la costura del mismo al guante. Una vez finalizado estos procesos solo queda establecer la conexión vía Bluetooth con el ordenador, con el objetivo de crear el entorno visual una vez lograda dicha conexión.

4.2.1. Montaje del circuito

A continuación se muestra el listado completo de los materiales necesarios para el montaje completo del circuito.

- Materiales necesarios:
    - Guantes con la tela con reactancia cosida.
    - Tarjeta Arduino Nano.
    - 3 resistencias 1 kΩ.
    - Giroscopio MPU6050.
    - Modulo Bluetooth HC-05.
    - Placa de topos.
    - Interruptor 1C2P.
    - Cables.

Para conectar todos los dispositivos en la placa de topos se han soldado intentando ocupar el menor espacio posible para minimizar su peso y disminuir la incomodidad que pueda producir un gran volumen en la mano dominante.

En la figura 4.7. se adjunta el esquema del montaje del circuito completo, el cual engloba los esquemas parciales mostrados en las figuras 4.2, 4.3, 4.5.

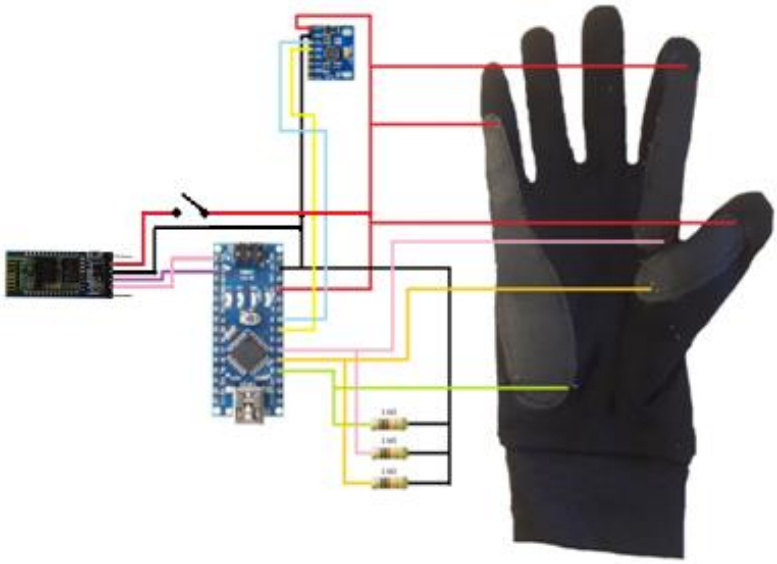


Figura 4.7. Esquema del montaje completo.

En la figura 4.8. se adjuntan dos foto del montaje real para observar su tamaño compacto y el conexionado de su parte superior e inferior.

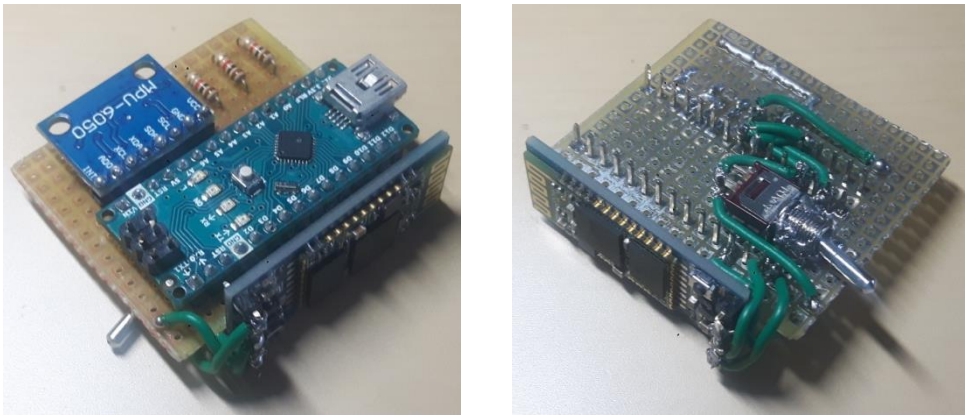


Figura 4.8. Fotos del montaje completo.

En la tabla 4.1. se muestra de forma esquemática y sencilla todos los pines de conexión entre la placa Arduino Nano y el resto de módulos.

Tabla 4.1. Guía de conexionado entre la tarjeta Arduino Nano y los módulos MPU6050 y HC-05.

Sensores	HC-05 Bluetooth				Giroscopio MPU6050			
Pines Arduino Nano	5 V	GND	TXD	RXD	5 V	GND	A 5	A 4
Pines sensor	VCC	GND	RXD	TXD	VCC	GND	SCL	SDA

Sensores	Tela 1		Tela 2		Tela 3	
Pines Arduino Nano	5 V	A0	5 V	A1	5 V	A2
Pines sensor	Extremo 1	Extremo 2	Extremo 1	Extremo 2	Extremo 1	Extremo 2

4.2.2. Alimentación circuito

Una vez conectada toda la circuitería se debe alimentar de tal forma que no deba estar conectado mediante el cable USB Mini B – USB limitando su rango de movimiento y por lo tanto interfiriendo en el juego habitual.

Tal y como comenta Luis Llamas en su web [24] existen varias posibilidades para alimentar una tarjeta Arduino con pilas o baterías, entre todas las opciones que se hallan en el mercado se ha elegido la pila de 9 V, debido a poseer la mejor relación voltaje – tamaño para poder situarse en la muñeca sin afectar al jugador. Para el conexionado de dicha pila se necesita un portapilas como el que se muestra en la figura 4.9.

Figura 4.9. Portapilas de 9 V [25].

Para poder sujetar la pila de 9 V junto con el portapilas conectado es necesario la creación de una pulsera que lo ciña a la muñeca, la cual ha sido cosida por una profesional y se consiguió una comodidad que no influye al usuario, dichas pulseras se adjuntan en una foto en la figura 4.10.

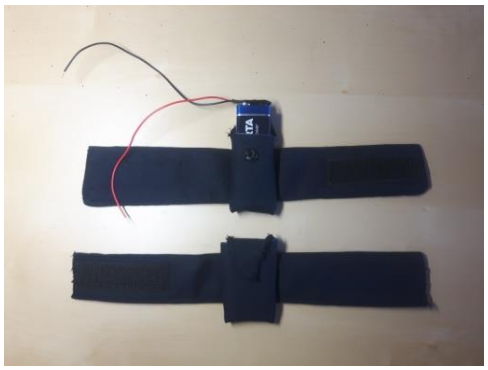


Figura 4.10. Fotos de las pulseras contenedoras de las pilas de 9 V.

4.2.3. Carcasa protectora

- **Diseño**

Tal y como se ha mencionado con anterioridad, se diseña una carcasa que a la vez protege el circuito y permite su costura a los guantes. Dicha carcasa se ha diseñado a través del programa SolidWorks y se trata de una caja rectangular que posee una serie de orificios para permitir ciertas acciones.

La primera es la entrada de los cables provenientes de las telas con reactancia, otra acción necesaria es la modificación del conmutador de dos posiciones para poder cargar un programa en la placa sin tener que extraerla. Para dicho almacenamiento del programa se permite la conexión a la entrada USB Mini Tipo B a través de otro orificio. Por último posee una serie de perforaciones en la base con el objetivo de servir como medio de anclaje para su costura al guante.

Dicho diseño se puede visualizar en la figura 4.11. que se trata de dos capturas para poder observar la pieza desde distintos ángulos, en la captura derecha se puede observar las inscripciones de " Load " y " Play " en el orificio perteneciente al conmutador para conocer las acciones que conllevan ambas posiciones.

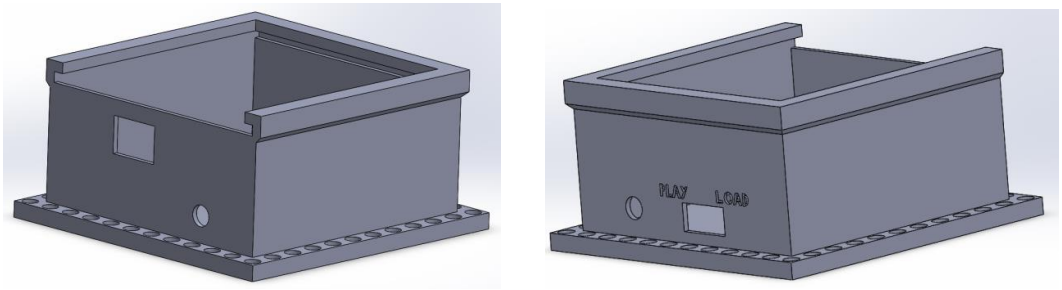


Figura 4.11. Capturas del diseño de la carcasa protectora en SolidWorks.

También se realizó el diseño de la tapa propia de la carcasa, para comprobar su correcto ajuste con la carcasa se realiza un ensamble con el objetivo de retocar los parámetros de dicha tapa, una vez ajustada se adjunta una captura en la figura 4.12. para observar su aspecto final.

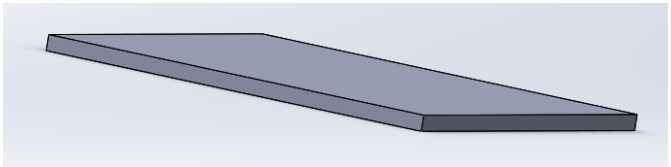


Figura 4.12. Captura del diseño de la tapa en SolidWorks.

- **Impresión**

Para la impresión de dichas piezas se han requerido los servicios del personal de FabLab de la universidad, los cuales han colaborado con una gran actitud y explicando el proceso a seguir para dicha impresión.

La primera acción a realizar para imprimir las piezas previamente diseñadas es importarlas como un archivo STL, el cual es un conjunto de puntos en 3D, es aconsejable modificar los parámetros estándares y ponerlos en la máxima calidad posible, para ello se debe desplazar ambos sliders al extremo derecho tal y como se muestra en la figura 4.13, este conjunto de puntos en 3D es procesable por cualquier impresora 3D del mercado.

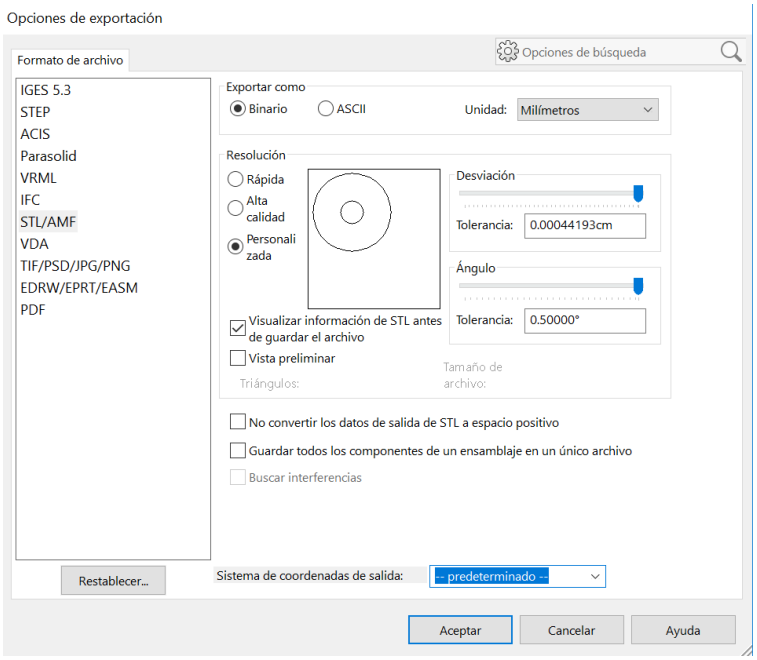


Figura 4.13. Captura del establecimiento parámetros óptimos en la importación de las piezas.

Una vez guardado el archivo como un conjunto de puntos 3D, se debe abrir en el programa denominado " Cura ", el cual es de libre acceso, dicho programa convierte estos puntos en un conjunto de órdenes que son procesados por cualquier impresora 3D. En este programa se pueden especificar los parámetros de impresión, tales como dureza de las paredes (dependiendo de las necesidades mecánicas de las piezas), la creación de soportes para las partes horizontales sin base física donde apoyarse, la velocidad de impresión, etc.

Todas estas características dependen del objetivo que cumplen dichas piezas, en el caso de este proyecto al tratarse de unas piezas sin necesidades mecánicas no es necesaria una gran densidad de sus paredes. Se necesitan soportes debido a que hay partes de la pieza sin base física y todo ello conlleva a una impresión de 3 horas y 45 minutos de duración por copia (se necesitan dos ya que existe una versión de usuario diestro y zurdo).

Una vez han sido impresas las piezas se vuelve a requerir de los servicios de una costurera profesional para la costura de dichas piezas a los guantes, obteniendo como resultado el observable en la foto adjuntada en la figura 4.14. solo necesitando como retoque final la soldadura de los hilos conductores provenientes de las telas.

Figura 4.14. Foto del resultado final de la impresión y cosido a los guantes.

4.2.4. Establecimiento conexión bluetooth

Para establecer la conexión mediante Bluetooth empleando el módulo HC-05 se debe alimentar a un voltaje entre 3,6 y 6 V, en el caso del proyecto se conecta al pin 5V y GND de la tarjeta Arduino Nano. Después se debe activar el Bluetooth del ordenador y buscar dicho dispositivo (nombre: HC-05), la contraseña de enlace por defecto es " 1234 ". Una vez realizado dichos pasos debería aparecer el puerto COM en Arduino IDE de la misma forma que al conectarlo por cable, pero si no aparece la opción de seleccionar dicho puerto se debe configurar manualmente el puerto de comunicaciones.

Para realizarlo lo primero que se debe hacer es abrir el " Administrador de dispositivos " del ordenador y en la pestaña " Acción " se escoge " Agregar hardware heredado " tal y como se muestra en la figura 4.15.

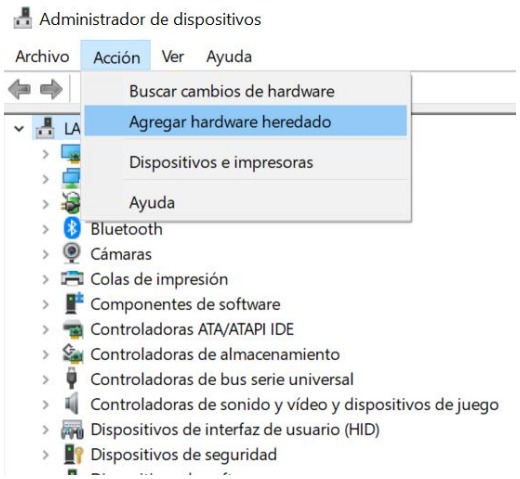


Figura 4.15. Captura agregación hardware heredado en Administrador de dispositivos.

En las siguientes ventanas se debe seleccionar la opción de " Siguiente " hasta que aparece la ventana de la figura 4.16. en la cual se debe seleccionar el tipo de hardware a instalar, escogiendo la opción " Puertos (COM y LPT) ".

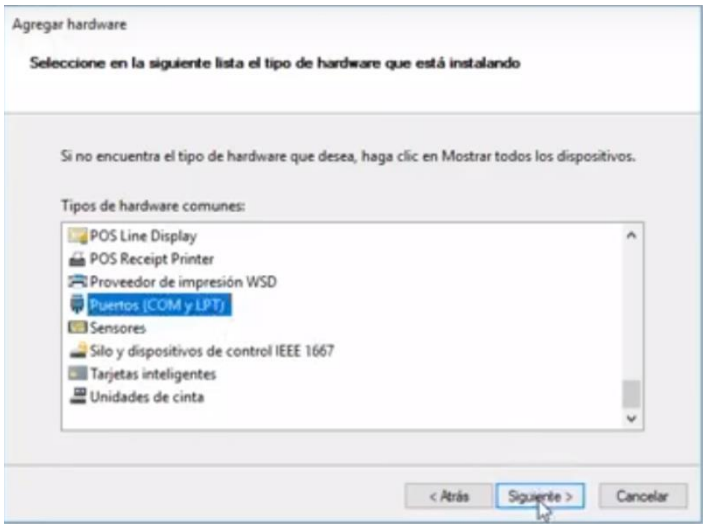


Figura 4.16. Captura selección de hardware a instalar.

En la siguiente ventana se debe seleccionar “ Puerto de comunicaciones ” tal y como se observa en la figura 4.17, clicando en “ Siguiente ” en las siguientes ventanas ya se establece la creación del puerto COM deseado.

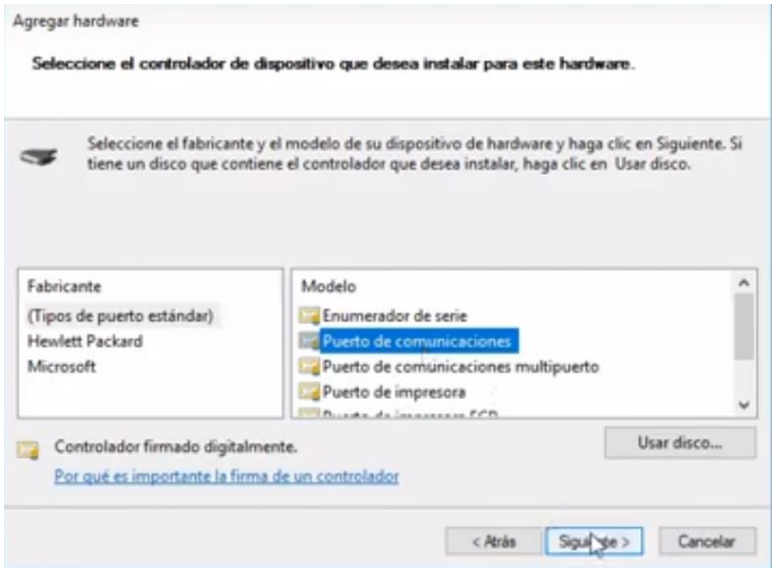


Figura 4.17. Captura de establecimiento de puerto de comunicaciones.

Una vez realizado todo el proceso ya se puede seleccionar el puerto COM en el programa Arduino IDE.

4.3. Programación

En este apartado se explica el procedimiento a seguir a la hora de programar tanto el programa de la tarjeta Arduino Nano empleando Arduino IDE, como el entorno visual empleando PyCharm que emplea el lenguaje Python. Se desarrollan aspectos tan importantes como la declaración de variables a emplear, establecimiento de librerías necesarias, procedimientos y cálculos del programa.

4.3.1. Arduino

Respecto a la tarjeta Arduino Nano se necesita descargar el programa Arduino IDE, el cual es gratis y posee una gran compatibilidad con cualquier tarjeta y módulo. El primer paso a realizar es el reconocimiento de la placa por parte del ordenador, para ello se debe conectar la placa empleando un cable USB – USB Mini Tipo B.

- **Selección tarjeta**

Una vez conectado y abierto el entorno de Arduino IDE se deben establecer los parámetros de nuestra placa, para ello se debe acceder a la pestaña de " Herramientas ", luego se selecciona " Placa " y posteriormente se escoge la tarjeta Arduino Nano (en el caso de este proyecto), este paso se puede observar en la figura 4.18.

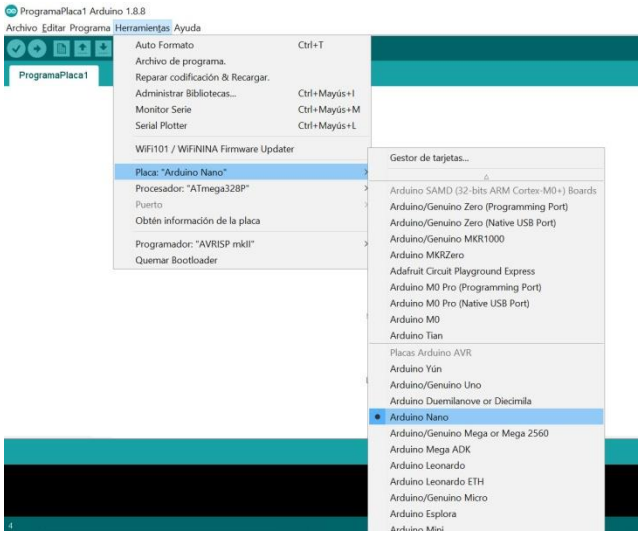


Figura 4.18. Captura selección placa en Arduino IDE.

Una vez seleccionada la placa a emplear se debe escoger el puerto COM que usa la tarjeta, para ello en la misma pestaña de " Herramientas ", se escoge en " Puerto ", esta opción solo es visible mientras se encuentre conectada la tarjeta con el ordenador, se puede visualizar en la figura 4.19.

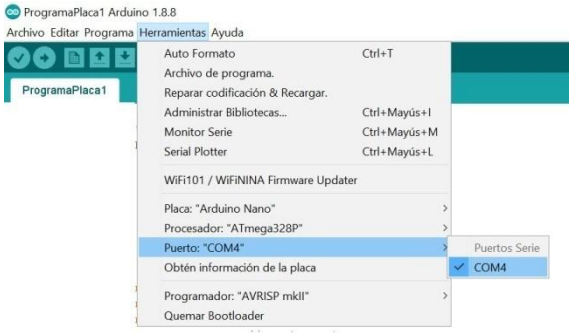


Figura 4.19. Captura selección Puerto COM en Arduino IDE.

Otro parámetro que se debe escoger es el procesador, en el caso de la tarjeta Arduino Nano es " Atmega328P " tal y como se muestra en la figura 4.20.

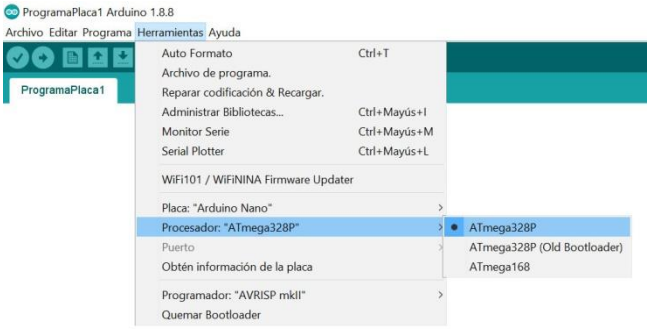


Figura 4.20. Captura selección procesador en Arduino IDE.

- **Librerías**

Una vez establecidos los parámetros de la placa a emplear ya se puede desarrollar el programa en sí, comenzando por la instalación de las librerías necesarias. Debido al uso de los módulos MPU6050 y HC-05 se necesitan librerías específicas para su uso, las cuales son " I2Cdev.h ", " MPU6050.h " y " Wire.h ".

Para la instalación de dichas librerías se deben descargar de internet (libre acceso) y añadir al programa Arduino IDE siguiendo el procedimiento que se muestra en la figura 4.21. en la pestaña de " Programa ", se selecciona " Incluir Librería " y a continuación " Añadir biblioteca .ZIP ", por último se selecciona la ubicación de las librerías descargadas en el ordenador y se guardan en el programa.

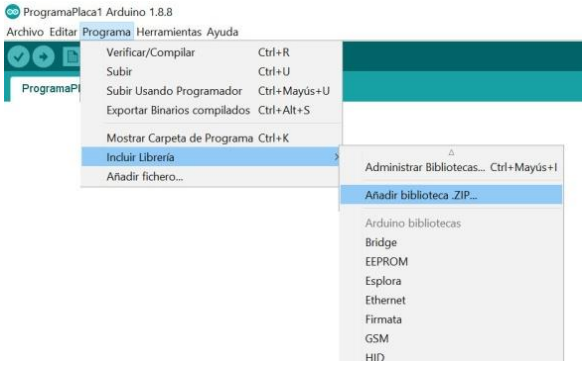


Figura 4.21. Captura importación librerías nuevas descargadas en Arduino IDE.

Una vez añadidas al Arduino IDE se deben importar en el programa que se desean implementar, para dicha acción se emplea la orden " #include " seguido de las librerías a importar, tal y como se muestra en la captura de la figura 4.22.

```
// Librerias
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
```

Figura 4.22. Captura de declaración de librerías en Arduino IDE.

- **Variables**

Una vez importadas las librerías se deben declarar las variables a emplear en el programa, se necesitan 3 variables enteras para establecer los valores de las resistencias externas ya medidas a través de un multímetro de mano y 3 variables reales para guardar los voltajes asociados a las entradas analógicas (A0, A1 y A2).

También se declaran 3 variables reales en las que se guardan los valores de las resistencias de las telas una vez calculadas y por último, 6 variables enteras para guardar los valores de las aceleraciones y giros sobre los 3 ejes espaciales. En esta etapa también se debe declarar el sensor MPU6050, todas las variables mencionadas se observan en la captura de la figura 4.23.

```
// Variables
int resistencia0 = 986;           // Resistencia fija entrada A0
int resistencia1 = 980;           // Resistencia fija entrada A1
int resistencia2 = 985;           // Resistencia fija entrada A2
float voltaje1 = analogRead(A0);  // Lectura voltaje entrada A0
float voltaje2 = analogRead(A1);  // Lectura voltaje entrada A1
float voltaje3 = analogRead(A2);  // Lectura voltaje entrada A2
float r1;                         // Resistencia sensor 1
float r2;                         // Resistencia sensor 2
float r3;                         // Resistencia sensor 3
int ax, ay, az;                   // Aceleraciones ejes x, y, z
int gx, gy, gz;                   // Giroscopio eje x
MPU6050 sensor;                   // Sensor MPU6050
```

Figura 4.23. Captura declaración de variables en Arduino IDE.

- **Inicialización**

Una vez declaradas las variables del programa se deben iniciar una serie de aspectos del programa, para ello se incluyen dentro del método " setup ( ) ". Se debe iniciar el puerto serie y especificar la velocidad de transmisión, en el caso de la tarjeta Arduino Nano es 9.600 baudios, también se deben iniciar el I2C y el sensor MPU6050. Los pines analógicos a emplear se añaden para que el programa detecte sus lecturas, los pines A4 y A5 son los asociados al giroscopio, y por último se realiza un test de prueba de dicho sensor para comprobar su correcta conexión, todo esto se observa en la figura 2.24.

```
void setup()
{
  Serial.begin(9600);             // Iniciando el puerto serial
  Wire.begin();                   // Iniciando I2C
  sensor.initialize();            // Iniciando el sensor

  pinMode(A0,INPUT);              // Resistencia meñique
  pinMode(A1,INPUT);              // Resistencia indice
  pinMode(A2,INPUT);              // Resistencia indice
  pinMode(A4,INPUT);              // SDA MPU6050
  pinMode(A5,INPUT);              // SCL MPU6050

  if (sensor.testConnection()) Serial.println("Sensor iniciado correctamente");
  else Serial.println("Error al iniciar el sensor");
}
```

Figura 4.24. Captura método setup en Arduino IDE.

- **Mainloop**

Una vez declaradas las variables e iniciado los parámetros necesarios para el programa principal se pueden calcular las resistencias correspondientes a las telas con reactancia (r1, r2 y r3) mediante un divisor de tensión y se guardan sus valores. También se leen y se guardan los valores del sensor MPU6050, es decir, las aceleraciones y las rotaciones en los 3 ejes espaciales.

Una vez medido y calculado todas las variables deseadas se deben enviar empleando el puerto serie COM asignado a la placa, para dicha acción se emplea el comando " Serial.println " y se envían todos escritos en formato json, con el objetivo de facilitar su posterior procesamiento. Por último se establece un tiempo de espera (delay) para volver a realizar dicho bucle, todo ello se observa en la captura de la figura 4.25.

```
void loop()
{
  // Calculamos las resistencias de los sensores
  r1 = resistencia0 * voltaje1 / (5.0 - voltaje1);
  r2 = resistencia1 * voltaje2 / (5.0 - voltaje2);
  r3 = resistencia2 * voltaje3 / (5.0 - voltaje3);

  // Leer las aceleraciones y velocidades angulares
  sensor.getAcceleration(&ax, &ay, &az);
  sensor.getRotation(&gx, &gy, &gz);

  // Se emplea serial println
  Serial.println("{\"r1\":"+String(r1) + ", \"r2\":"+String(r2) + ", \"r3\":"+String(r3)
  + ", \"ax\":"+String(ax) + ", \"ay\":"+String(ay) + ", \"az\":"+String(az)
  + ", \"gx\":"+String(gx) + "}");

  delay(500);
}
```

Figura 4.25. Captura programa principal en Arduino IDE.

El formato json permite almacenar los datos en formato diccionario, por lo tanto después se puede buscar la variable deseada y el valor que devolverá será el asociado a dicha variable.

4.3.2. Python

Respecto al entorno visual situado en el ordenador se ha escogido el desarrollador PyCharm debido a su gran versatilidad, compatibilidad y su portabilidad libre de acceso para cualquier usuario. La primera acción a realizar previa a la creación del programa es la descarga e importación de las librerías necesarias.

- **Librerías**

Para descargar una librería en un programa que desarrollo en Python se necesita descargar de internet o emplear unos comandos específicos en la ventana de comandos, pero en PyCharm se descargan desde el mismo programa. Se selecciona la pestaña " File ", dentro de ella la opción " Settings " se debe seleccionar el intérprete, en el caso del proyecto " Python 3.7 ". Una vez realizado esto se debe clicar en el " + " situado en la parte superior derecha, para añadir una nueva librería y dentro de la nueva ventana se realiza la búsqueda de dicha librería tal y como se muestra en la figura 4.26.

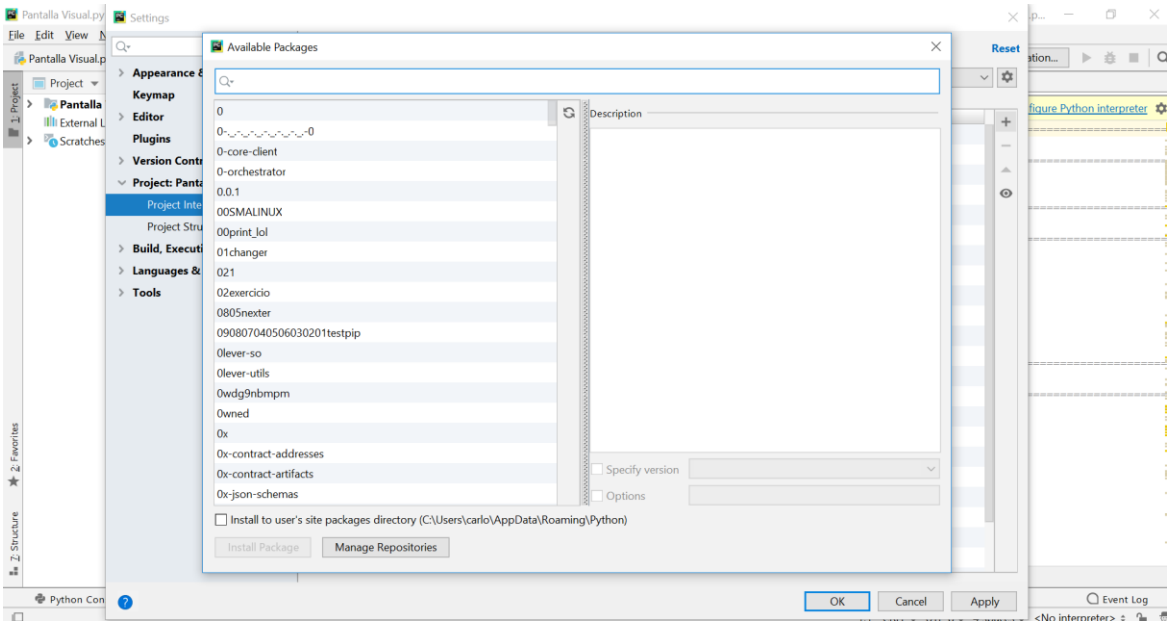


Figura 4.26. Captura descarga de librerías nuevas en PyCharm.

Una vez descargadas las librerías se deben importar en el programa, para ello se emplea el comando " import ", para poder realizar el programa se necesitan las librerías de " tkinter " para el entorno gráficos, " simplejson ", " serial " y " time " para la comunicación serie con Arduino y el resto englobadas en " matplotlib ", " numpy " " integrate " y " sys " para realizar cálculos y la creación de gráficos, todo ello se muestra en la figura 4.27.

```
1   # ================================================================
2   #                              Librerias
3   # ================================================================
4   from tkinter import *
5   from tkinter import font
6   from tkinter import ttk
7   import serial, time
8   import simplejson
9   import matplotlib.pyplot as plt
10  import matplotlib as mpl
11  import numpy as np
12  from sympy import integrate
13  import matplotlib.backends.tkagg as tkagg
14  from matplotlib.backends.backend_agg import FigureCanvasAgg
15  import sys
```

Figura 4.27. Captura importación librerías en PyCharm.

- **Parámetros de la ventana**

Una vez importadas las librerías el siguiente paso es definir los parámetros de la ventana principal, tales como su título, dimensiones y la imposibilidad de modificar dicho tamaño (resizable = False). También se deben definir dos fuentes a emplear, la de las letras de cualquier label y la perteneciente a los botones, todo ello se observa en la figura 4.28.

```
17  # ================================================================
18  #                        Parametros de la ventana
19  # ================================================================
20  ventana=Tk()
21  ventana.title(" Programa de ayuda para la practica del billar ")
22  ventana.resizable(False,False)
23  ventana.geometry("1270x650")
24  fuenteBoton=font.Font(family="Helvetica", size=17, weight="bold")
25  fuenteNormal=font.Font(family="Helvetica", size=12)
26
```

Figura 4.28. Captura establecimiento parámetros de la ventana principal en PyCharm.

- **Variables**

Una vez establecidos los parámetros de la ventana principal se necesitan declarar las variables a emplear durante el programa las cuales se observan en la figura 4.29. Se crea una variable para guardar la selección de la mano dominante del usuario (varMano), una variable que guarde el momento que se desea visualizar del historial con el slider (varVisualizacion) teniendo como valor máximo de dicho slider el total de mediciones guardadas (varMomento).

Por otra parte, para guardar la velocidad del momento de tacada y después visualizar dicho valor en la barra de velocidad se emplea una variable ” varVelocidad “, también se emplea una variable que establezca el inicio y parada del periodo de medición (varMedicion) la cual inicialmente estará en falso para que no se produzca la medición. Se establecen los umbrales de presión máxima permitida (presionMax) y presión mínima (presionMin) que delimitan los tres posibles estados (insuficiente, correcta, excesiva). También se guardan la rotación máxima y las desviaciones máximas permitidas en los ejes espaciales (rotaciónMax, desviaciónMax y desviaciónMax2).

Por último se necesitan una serie de listas, una para el guardado de los valores obtenidos de la medición (medicion), otra para guardar en un historial todos los valores medidos (historial). Dos listas para el procesamiento y guardado de la aceleración medida (aceleración y aceleración2), al igual que con la velocidad obtenida integrando la aceleración (velocidad y velocidad2) y finalmente, una lista para el guardado de las posiciones calculadas a partir de la integración de las velocidades (posición).

```
27  # ============================================================
28  #                          Variables
29  # ============================================================
30  varMano = 1
31  varVisualizacion = IntVar()
32  varVelocidad = IntVar()
33  varMomento = DoubleVar()
34  varMedicion = False
35  presionMax = IntVar()
36  presionMin = IntVar()
37  desviacionMax = IntVar()
38  desviacionMax2 = IntVar()
39  rotacionMax = IntVar()
40  medicion = {}
41  historial = []
42  aceleracion = []
43  aceleracion2 = []
44  velocidad = []
45  velocidad2 = []
46  posicion = []
```

Figura 4.29. Captura declaración de variables en PyCharm.

- **Imágenes**

Una vez declaradas las variables se deben importar las imágenes a emplear durante el programa, para dicha labor se crea un diccionario que las engloba todas con el objetivo de minimizar líneas de código. Para la importación de dichas imágenes se emplea el comando " PhotoImage " continuado del nombre del archivo, en el caso de las imágenes empleadas son todas " .png " debido a la necesidad estética de solapar unas a otras y que no se observe el contorno que las envuelve.

Las primeras categorías creadas son " presión ", " rotación ", " desviación " y " leyenda ". Dentro de la categoría presión se divide entre diestro y zurdo, el cual es determinado como ya se ha mencionado con la variable " varMano ", dentro de ambas opciones se divide entre " mano " para cargar la imagen sin ninguna señalización e " índice " y " menique ", en las cuales se diferencia entre las imágenes que señalizan la presión correcta, desmesurada o insuficiente para ambas zonas de presión.

En la categoría " desviación " se divide entre " horizontal " y " vertical ", cada una centrada en un eje espacial, dentro de los cuales se diferencia entre diestro y zurdo, cargando las imágenes para señalizar si existe alguna desviación o no. Dentro de la categoría " rotación " al centrarse solo en el eje del movimiento no existe subdivisión alguna y se cargan las imágenes que señalizan si existe alguna rotación o no, solo variando si se trata de un jugador diestro o zurdo.

Por último en la categoría " leyenda " se cargan las imágenes de la leyenda, señalizando los significados de cada color, todo ello se puede observar en la figura 4.30.

```
52  # ================================================================================
53  #                    Las imagenes que se van a emplear
54  # ================================================================================
55  imagenes = {
56      'presion': {
57          # Diestro
58          1: {
59              'mano': PhotoImage(file="mano derecha.png"),
60              'indice': {
61                  'correcta': PhotoImage(file="mano derecha verde.png"),
62                  'rojo': PhotoImage(file="mano derecha rojo.png"),
63                  'azul': PhotoImage(file="mano derecha azul.png")
64                  },
65              'menique': {
66                  'correcta': PhotoImage(file="mano derecha verde 2.png"),
67                  'rojo': PhotoImage(file="mano derecha rojo 2.png"),
68                  'azul': PhotoImage(file="mano derecha azul 2.png")
69                  }
70          },
71          # Zurdo
72          2: {
73              'mano': PhotoImage(file="mano izquierda.png"),
74              'indice': {
```

```
                'correcta': PhotoImage(file="mano izquierda verde.png"),
                'rojo': PhotoImage(file="mano izquierda rojo.png"),
                'azul': PhotoImage(file="mano izquierda azul.png")
                },
            'menique': {
                'correcta': PhotoImage(file="mano izquierda verde 2.png"),
                'rojo': PhotoImage(file="mano izquierda rojo 2.png"),
                'azul': PhotoImage(file="mano izquierda azul 2.png")
                }
        },
    },
    'desviacion': {
        'vertical': {
            # Diestro
            1: {
                'correcta': PhotoImage(file="billar.png"),
                'arriba': PhotoImage(file="billar rojo arriba.png"),
                'abajo': PhotoImage(file="billar rojo abajo.png")
            },
            # Zurdo
            2: {
                'correcta': PhotoImage(file="billar zurdo.png"),
                'arriba': PhotoImage(file="billar zurdo rojo arriba.png"),
                'abajo': PhotoImage(file="billar zurdo rojo abajo.png")
            }
        },
        'horizontal': {
            # Diestro
            1: {
                'correcta': PhotoImage(file="billar 2.png"),
                'derecha': PhotoImage(file="billar 2 rojo derecha.png"),
                'izquierda': PhotoImage(file="billar 2 rojo izquierda.png")
            },
            # Zurdo
            2: {
                'correcta': PhotoImage(file="billar 2 zurdo.png"),
                'derecha': PhotoImage(file="billar 2 zurdo rojo izquierda.png"),
                'izquierda': PhotoImage(file="billar 2 zurdo rojo derecha.png")
            }
        }
    },
    'rotacion': {
        'correcta': PhotoImage(file="rotacion blanco.png"),
        'diestro': PhotoImage(file="rotacion izquierda.png"),
        'zurdo': PhotoImage(file="rotacion derecha.png")
    },
    'leyenda': {
        'verde': PhotoImage(file="verde.png"),
        'rojo': PhotoImage(file="rojo.png"),
        'azul': PhotoImage(file="azul.png")
    }
}
```

Figura 4.30. Captura declaración del diccionario de imágenes en PyCharm.

- **Comunicación Arduino**

Una vez declaradas las imágenes se debe declarar el puerto serie que emplea Arduino, para ello se crea una variable " arduino " y se emplea el comando " serial.Serial " en el cual se debe establecer el número del puerto y la velocidad de transmisión, en el caso de la tarjeta Arduino Nano son 9.600 baudios, después se dejan 2 segundos para que se inicialice el puerto serie, tal y como se observa en la figura 4.31.

```
116  # ========================================================================
117  #                    Establecimiento puerto serie
118  # ========================================================================
119  arduino=serial.Serial('COM4', 9600)
120  time.sleep(2)
```

Figura 4.31. Captura declaración del puerto serie para Arduino en PyCharm.

- **Labels**

Una vez declarado todos los parámetros, librerías, variables e imágenes ya se pueden diseñar los aspectos gráficos del programa, la primera acción es la creación de una serie de textos fijos y entradas de texto. Tales como los textos " Nombre " y " Jugada " , acompañados de sus respectivas entradas de textos. Otros textos incluidos en el programa son los correspondientes a la leyenda explicativa de las imágenes de la presión de la mano, los cuales se acompañan con imágenes de los colores correspondientes, color rojo con presión excesiva, verde con correcta y azul con insuficiente, todo ello se muestra en la figura 4.32.

```
131  # ========================================================================
132  #                    Entradas de texto y labels
133  # ========================================================================
134  Label(ventana, text = "Nombre: ", font=fuenteNormal).place(x=600, y=20)
135  cuadroNombre = Entry(ventana, font=fuenteNormal).place(x=670, y=20)
136
137  Label(ventana, text = "Jugada: ", font=fuenteNormal).place(x=900, y=20)
138  cuadroJugada = Entry(ventana, font=fuenteNormal).place(x = 970, y = 20)
139
140  Label(ventana, text="Leyenda : ", font=fuenteNormal).place(x=90, y=50)
141  Label(ventana, image=imagenes['leyenda']['rojo']).place(x=185, y=25)
142  Label(ventana, image=imagenes['leyenda']['verde']).place(x=185, y=50)
143  Label(ventana, image=imagenes['leyenda']['azul']).place(x=185, y=75)
144  Label(ventana, text="Presion excesiva", font=fuenteNormal).place(x=230, y=25)
145  Label(ventana, text="Presion correcta", font=fuenteNormal).place(x=230, y=50)
146  Label(ventana, text="Presion insuficiente", font=fuenteNormal).place(x=230, y=75)
```

Figura 4.32. Captura creación de labels de la leyenda en PyCharm.

- **Selección mano dominante**

La primera acción a realizar por el usuario en el programa es la selección de su mano dominante, que por defecto se encuentra en la opción diestro, pero a través de un " Radiobutton " se puede escoger entre las dos opciones. Dicha selección hace que se modifiquen las imágenes para mostrar las versiones para jugadores diestros o zurdos, dicha función se observa en la figura 4.33.

```
148 # ============================================================================================================
149 #                    Seleccion de la mano dominante
150 # ============================================================================================================
151 def imagenMano():
152     Label(ventana, image=imagenes['presion'][varMano]['mano']).place(x=0, y=100)
153     Label(ventana, image=imagenes['rotacion']['diestro']).place(x=990, y=170)
154     Label(ventana, image=imagenes['desviacion']['horizontal'][varMano]['correcta']).place(x=990, y=200)
155     Label(ventana, image=imagenes['desviacion']['vertical'][varMano]['correcta']).place(x=500, y=200)
156
157
158 Label(ventana, text = "Mano Dominante: ", font = fuenteNormal).place(x = 500, y = 95)
159 Radiobutton(ventana, text = "Derecha", variable=varMano, value=1, command=imagenMano,
160             font = fuenteNormal).place(x=660, y=80)
161 Radiobutton(ventana, text = "Izquierda", variable=varMano, value=2, command=imagenMano,
162             font = fuenteNormal).place(x=660, y=110)
```

Figura 4.33. Captura función selección mano dominante en PyCharm.

- **Comenzar y detener medición**

Con el objetivo de comenzar y detener la obtención de mediciones se crean dos botones que poseen asociados las funciones " empezar_Medición " y " parar_Medición " que modifican la variable " varMedicion ", solo produciéndose la medición si el valor de dicha variable es True, es decir, si se pulsa el botón " Empezar ", tal y como se observa en la figura 4.34.

```
166 # ============================================================================================================
167 #                    Comenzar y detener medicion
168 # ============================================================================================================
169 def empezar_Mediciones():
170     varMedicion = True
171
172
173 def parar_Mediciones():
174     varMedicion = False
175
176
177 Button(ventana, text="Empezar", command=empezar_Mediciones(), width=10, height=2, relief="ridge",
178        borderwidth=10, font=fuenteBoton, bg="pale green").place(x=825, y=70)
179 Button(ventana, text="Parar", command=parar_Mediciones(), width=10, height=2, relief="ridge",
180        borderwidth=10, font=fuenteBoton, bg="salmon").place(x=1000, y=70)
```

Figura 4.34. Captura creación botones para comenzar y detener medición en PyCharm.

- **Slider**

Teniendo como objetivo poder visualizar las mediciones del historial se crea un slider cuya longitud es el total de mediciones acumuladas en dicha lista desde el inicio a la parada del proceso (varMomento). Deslizando el slider se varía el valor del momento a visualizar con la variable " varVisualizacion " y con ello es observable la presión, rotación, desviación y velocidad del momento elegido en las imágenes de la ventana principal, dicho código se observa en la figura 4.35.

```
# ============================================================================================
#                                    Slider
# ============================================================================================
Scale(ventana, orient=HORIZONTAL, length=400, width=10, sliderlength=5, from_=0, to=varMomento,
      bg = "LightBlue1", variable=varVisualizacion).place(x=450, y=550)
```

Figura 4.35. Captura slider para visualizar las variables de un momento en PyCharm.

- **Barra velocidad**

Uno de los parámetros más importantes de la técnica del billar es la velocidad con la que se ejerce la tacada, para la representación de dicha variable se emplea una barra, cuyo llenado se encuentra asociado a la variable " varVelocidad ", en la cual se guarda el valor justo en el momento del impacto, dicho código se muestra en la figura 4.36.

```
# ============================================================================================
#                                Barra velocidad
# ============================================================================================
Label(ventana, text="Velocidad del golpe: ", font=fuenteNormal, bg="LightBlue1").place(x=820, y=330)
ttk.Progressbar(ventana, value=varVelocidad).place(x=850, y=354)
```

Figura 4.36. Captura creación barra de velocidad de golpeo en PyCharm.

Una vez realizado todos estos procesos ya se abarca el bucle principal del programa, en el cual se deben destacar una serie de partes importantes.

- **Guardado de mediciones**

Empleando el comando " arduino.readline() " se leen los datos obtenidos del puerto serie ya declarado y después se realiza un " try " para comprobar la correcta lectura cuando la variable " varMedicion " sea True, por lo tanto, se producen mediciones. Una vez se han leído se guardan los valores en el listado " medición ", dichos valores son [resistencia 1 : valor, resistencia 2 : valor, resistencia 3 : valor, aceleración eje x : valor, aceleración eje y : valor, aceleración eje z : valor, rotación eje x : valor] tal y como se muestra en la figura 4.37.

```
174  # ================================================================================================================
175  #                                Main loop
176  # ================================================================================================================
177  while True:
178      jsonResult=arduino.readline()
179      try:
180          if varMedicion==True:
181              medicion.extend(simplejson.loads(jsonResult))    #{'r1': valor, 'r2', 'r3', 'ax', 'ay', 'az', 'gx'}
182      except Exception:
183          pass
```

Figura 4.37. Captura guardado mediciones en PyCharm.

- **Medición de presión**

Una vez guardados los datos del puerto serie en el listado " medición " ya este no se encuentra vacío y se comienza su procesamiento, la primera acción a realizar es la medición de la presión. Se diferencia entre usuarios diestros y zurdos debido a las diferentes imágenes a emplear y sus diferentes posicionamientos. Para cada una de las presiones (zona del meñique y zona del índice) se compara con los umbrales mínimo y máximo para catalogar el valor medido como correcto, insuficiente o excesivo. Todo ello se puede observar en la figura 4.38.

```
195      if medicion != {}:
196          # MEDICION DE PRESION
197          if varMano == 1:      # Caso jugador diestro
198              if (medicion['r1']) > presionMax :      # Presion indice
199                  Label(ventana, image=imagenes['presion'][varMano]['indice']['rojo']).place(x=250, y=100)
200              elif (medicion['r1']) < presionMin :
201                  Label(ventana, image=imagenes['presion'][varMano]['indice']['azul']).place(x=250, y=100)
202              else:
203                  Label(ventana, image=imagenes['presion'][varMano]['indice']['verde']).place(x=250, y=100)
204
205              if (medicion['r2']) > presionMax :      # Presion menique
206                  Label(ventana, image=imagenes['presion'][varMano]['menique']['rojo']).place(x=0, y=100)
207              elif (medicion['r2'])<presionMin:
208                  Label(ventana, image=imagenes['presion'][varMano]['menique']['azul']).place(x=0, y=100)
209              else:
210                  Label(ventana, image=imagenes['presion'][varMano]['menique']['verde']).place(x=0, y=100)
```

```
221         else:                       # Caso jugador zurdo
222             if (medicion['r1']) > presionMax:       # Presion indice
223                 Label(ventana, image=imagenes['presion'][varMano]['indice']['rojo']).place(x=0, y=100)
224             elif (medicion['r1'])<presionMin:
225                 Label(ventana, image=imagenes['presion'][varMano]['indice']['azul']).place(x=0, y=100)
226             else:
227                 Label(ventana, image=imagenes['presion'][varMano]['indice']['verde']).place(x=0, y=100)
228 
229             if (medicion['r2']) > presionMax:       # Presion menique
230                 Label(ventana, image=imagenes['presion'][varMano]['menique']['rojo']).place(x=353, y=100)
231             elif (medicion['r2'])<presionMin:
232                 Label(ventana, image=imagenes['presion'][varMano]['menique']['azul']).place(x=353, y=100)
233             else:
234                 Label(ventana, image=imagenes['presion'][varMano]['menique']['verde']).place(x=353, y=100)
```

Figura 4.38. Captura medición de presión en PyCharm.

- **Desviación**

Una vez realizada la medición de presión se debe comprobar la existencia de desviación en la realización del movimiento de ataque. Teniendo en cuenta que el movimiento deseado se realiza a lo largo del eje x, cualquier movimiento en los otros dos ejes espaciales sería un fallo de técnica, permitiendo cierto margen permisible, por lo que para ello se observan los valores de las aceleraciones de dichos ejes y se comparan los valores con los umbrales establecidos anteriormente, todo ello se observa en la figura 4.39.

```
227         # DESVIACION EN EL ATAQUE
228         if (medicion['ay']) > desviacionMax:     # Variable ay ( arriba/abajo)
229             Label(ventana, image=imagenes['desviacion']['vertical'][varMano]['arriba']).place(x=500, y=200)
230         elif (medicion['ay']) < desviacionMax2:
231             Label(ventana, image=imagenes['desviacion']['vertical'][varMano]['abajo']).place(x=500, y=200)
232 
233         if(medicion['az']) > desviacionMax:      # Variable az ( derecha/izquierda)
234             Label(ventana, image=imagenes['desviacion']['horizontal'][varMano]['derecha']).place(x=990, y=200)
235         elif(medicion['az']) < desviacionMax2:
236             Label(ventana, image=imagenes['desviacion']['horizontal'][varMano]['izquierda']).place(x=990, y=200)
```

Figura 4.39. Captura medición de desviaciones en PyCharm.

- **Rotación muñeca**

El siguiente aspecto a medir se trata de la rotación de la muñeca, teniendo en cuenta que el eje x es en el que se realiza el movimiento deseado, la rotación indeseada se realiza en torno a dicho eje. Se permite un rango de margen y señalizando si sobrepasa el umbral especificado, tal y como se emplea en los otros parámetros se diferencia entre jugadores diestros y zurdos, todo ello se muestra en la figura 4.40.

```
238         # ROTACION MUNECA
239         if (medicion['gx']) > rotacionMax:       # Variable gx
240             if varMano == 1:       # Caso jugador diestro
241                 Label(ventana, image=imagenes['rotacion']['diestro']).place(x=990, y=170)
242             else:                  # Caso jugador zurdo
243                 Label(ventana, image=imagenes['rotacion']['zurdo']).place(x=990, y=170)
244         else:
245             Label(ventana, image=imagenes['rotacion']['correcta']).place(x=990, y=170)
246
247         historial.append(medicion)
248         aceleracion.append(medicion['ax'])
249         medicion = []
```

Figura 4.40. Captura medición de rotación en PyCharm.

Una vez se han procesado todas las variables de la lista “ medicion ” se copian dichos valores en la lista “ historial ” y se importa el valor de la aceleración del eje x (ax) al listado “ aceleración ” y por último se borran los valores que se encontraban en la lista “ medicion ” para guardar la siguiente medición.

- **Gráficos**

Después de haberse guardado las mediciones de aceleraciones del eje x en el listado de “ aceleración ” se realiza una copia en la lista “ aceleracion2 ” para su integración y guardado de las velocidades obtenidas en el listado denominado “ velocidad ”. Se repite el mismo proceso para realizar una copia de las velocidades en el listado “ velocidad2 ” y la obtención de las posiciones y su guardado en “ posición ” después de su integración.

Una vez obtenido los 3 listados se crea una figura y se emplea el comando subplot para realizar un graficado de los 3 parámetros, los cuales comparten el eje x debido a que es el eje temporal. Para cada variable se ha creado un label en el eje vertical señalizando que variable se representa en cada gráfico. En la figura 4.41 se observa el código descrito que permite dicha graficación.

```
# MOSTRAR GRAFICOS
def mostrar_graficos():
    ventana.iconify()
    aceleracion2 = aceleracion[:]
    velocidad = it.cumtrapz(aceleracion2,initial=0)
    velocidad2 = velocidad[:]
    posiciones = it.cumtrapz(velocidad2,initial=0)

    plt.figure("Gráficos")

    ax1 = plt.subplot(311)
    plt.plot(aceleracion, 'c')
    plt.xticks(visible=False)
    plt.ylabel(" Aceleración ( m/s^2 ) ")
    plt.grid()
    plt.title(" Gráficas ")

    ax2 = plt.subplot(312, sharex=ax1)
    plt.plot(velocidad, 'b')
    plt.xticks(visible=False)
    plt.ylabel(" Velocidad ( m/s ) ")
    plt.grid()

    ax3 = plt.subplot(313, sharex=ax1)
    plt.plot(posiciones, 'g')
    plt.ylabel("Posición ( m )")
    plt.xlabel("Tiempo ( s ) ")
    plt.grid()

    plt.show()


Button(ventana, text="Observar graficos movimiento", command=mostrar_graficos, width=30,
       height=2, relief="ridge", borderwidth=10, font=fuenteNormal, bg="khaki1").place(x=900, y=500)

ventana.update_idletasks()
ventana.update()
ventana.mainloop()
```

Figura 4.41. Captura creación de gráficos en PyCharm.

La última parte del programa principal es el mantenimiento de la parte grafica mientras se producen los cálculos y procesamiento de las mediciones, para dicha acción se emplean los comandos " ventana.update_idletasks() " y " ventana.update() ", siendo " ventana " la correspondiente a la parte gráfica.

Una vez comentado todas las partes pertenecientes al código del programa se adjuntan las capturas de las ventana creadas, en primer lugar en la figura 4.42 se adjunta una captura de la ventana principal.

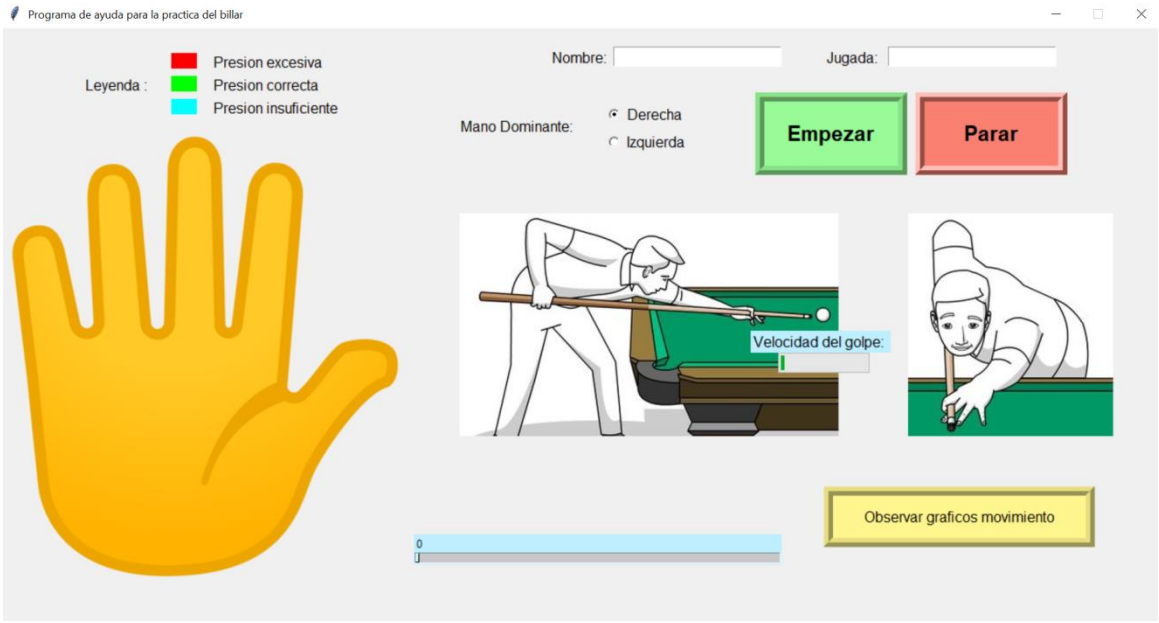


Figura 4.42. Captura ventana principal en PyCharm.

Y por último en la figura 4.43. se adjunta una captura de la ventana de gráficos.

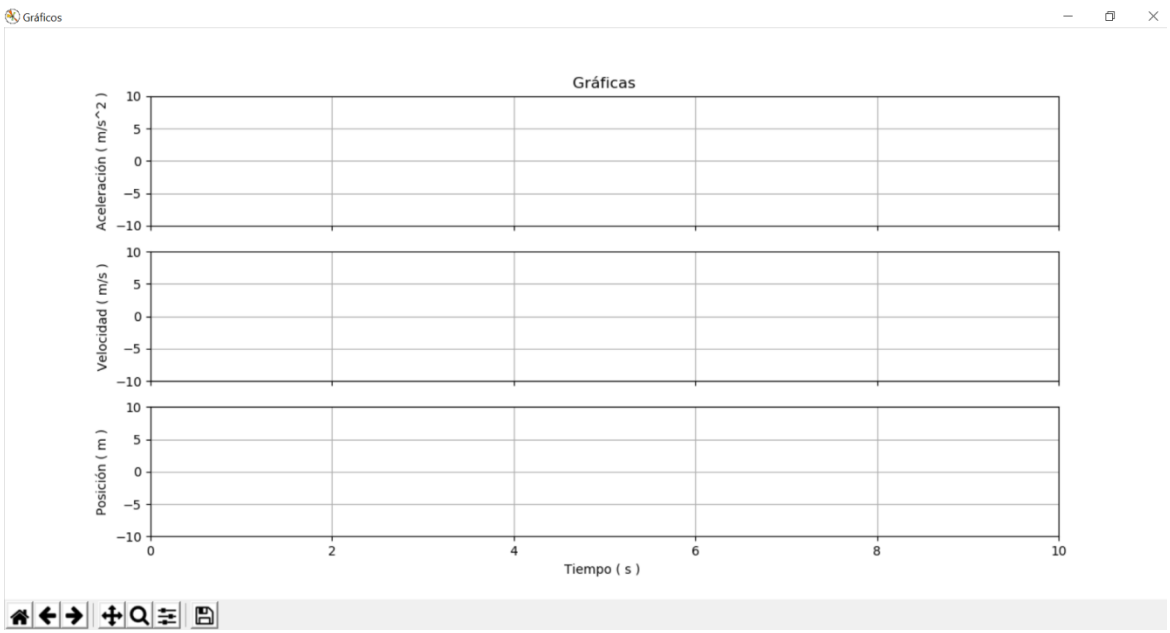


Figura 4.43. Captura ventana de gráficos en PyCharm.

5. Análisis del impacto ambiental

El dispositivo implementado no genera ningún riesgo ambiental directamente, pero de forma indirecta sí, ya que para su creación es necesaria la fabricación de ciertos componentes electrónicos cuyas materias primas son extraídas mediante la minería. Como residuos de su fabricación se generan mercurio, plomo, cianuro, etc. Los cuales son altamente peligrosos para el agua, suelo y aire, creando un malestar para la salud humana y ambiental [26].

Otro factor a analizar es la gestión de los desechos, la cual aumenta la contaminación ambiental y existe un riesgo asociado a los componentes que poseen mercurio, fósforo, cadmio o bromo, siendo los productores de efectos colaterales tanto en los manipuladores de dichos componentes como al medio ambiente. Según el Programa de Naciones Unidad para el Medio Ambiente (PNUMA), en un informe del año 2010, se expuso el dato que en el mundo se producen alrededor de 40 millones de toneladas de residuos electrónicos anuales [27], se estima que hay un incremento cada cinco años de chatarra electrónica entre el 16 % y 28 %, es por esta razón, que se han creado una serie de normativas para la protección ambiental que exigen el uso, producción y distribución de artefactos tecnológicos de manera responsable [28].

Aunque se realiza el presupuesto de producción a media escala, el dispositivo actualmente implementado posee un bajo riesgo ambiental, si se realiza la producción tal y como se ha estudiado sí que se provocarían dicho efectos por lo que habría que tener las convenidas precauciones y depositar las residuos en los sitios destinados para ello.

6. Presupuesto

En este apartado se calcula el coste perteneciente a la producción del prototipo, especificando el precio de cada uno de los componentes y también se calcula la mano de obra para su desarrollo. Una vez especificado el coste de un prototipo se realiza un presupuesto para saber el precio orientativo de una producción en masa.

6.1. Prototipo implementado

6.1.1. Materiales

En la tabla 6.1. se adjunta el resumen de los materiales empleados en la creación del prototipo, sin contabilizar los recursos que ya se poseían por ejemplo el soldador.

Tabla 6.1. Presupuesto materiales empleados en el proyecto y sus costes de adquisición.

Material	Costo
Arduino NANO (2 Uds.)	59,04 €
Dsd Tech HC-05 Bluetooth (2 Uds.)	16,98 €
Sensor MPU6050 (2 Uds.)	3,40 €
Guantes	1,60 €
Tela con reactancia	11,59 €
Resistencia 1 k (6 Uds.)	0,12 €
Pila Alcalina 9 V (2 Uds.)	9,58 €
Porta Pila 9 V Tipo Clip (2 Uds.)	0,54 €
Placa de Fibra de Vidrio	2,27 €
Conmutador 1CC Palanca (2 Uds.)	1,82 €
Cable USB - Mini USB Tipo B	2,93 €
Cables	1,20 €
Impresión 3D (2 Uds.)	3,50 €
Hilo conductivo	15,39 €
TOTAL	**129,96 €**

6.1.2. Mano de obra

En la tabla 6.2. se muestran las horas empleadas tanto por el propietario del prototipo en la administración, planeación, investigación, ingeniería y redacción de la memoria, como de la empresa de costura de la cual se han necesitado sus servicios para coser las telas con reactancia y la carcasa al guante.

Tabla 6.2. Presupuesto mano de obra.

Descripción	Costo por horas	Horas	Costo total
Administración del proyecto	15 €	15	225 €
Planeación	15 €	15	225 €
Investigación	20 €	80	1.600 €
Ingeniería	35 €	220	7.700 €
Costurera	10 €	6,5	65 €
Redacción	15 €	100	1.500 €
TOTAL			**11.315 €**

6.1.3. Presupuesto total

Finalmente, en la tabla 6.3 se adjunta la tabla del coste total del prototipo implementado, sumando el precio de los materiales empleados y las horas de mano de obra.

Tabla 6.3. Presupuesto total del proyecto.

Descripción	Costo total
Materiales	129,96 €
Mano de obra	11.315 €
TOTAL	**11.444,96 €**

6.2. Producción en masa

Debido a que este proyecto abarca un campo que no ha sido poco desarrollado y que el prototipo desarrollado es poco intrusivo se ha decidido contactar con varias empresas y explicarles el deseo de producir el montaje a media escala, por lo que han facilitado sus presupuestos para dicho objetivo.

Para ello dichas empresas han realizado un presupuesto de la producción de 100 y 1.000 unidades de PCB, a los cuales se deben añadir el coste de las impresiones 3D de las carcasas, las telas con reactancia y los guantes. También se debe añadir el precio de la mano de obra de la empresa de costura encargada de coser las telas con reactancia y las carcasas a los guantes.

Las empresas que han ofrecido un mejor presupuesto para las unidades y especificaciones aportadas de las PCB es " Prototyme ", todos los presupuestos recibidos cumplían las mismas características con un gasto de primera fabricación y sin incluir impuestos ni transporte, pero Prototyme posee el precio más competitivo.

Respecto a la impresión de las carcasas aunque la opción lógica sería contactar con empresas de prensado ya que no posee gran detalle dichas piezas, como presupuesto ficticio inicial se ha recurrido a empresas de impresión 3D empleando los archivos STL (conjunto de puntos 3D) que ya se poseían para su impresión inicial. Finalmente después de comparar presupuestos se ha escogido el más asequible que concuerda con el ya empleado servicio de impresión de la universidad, el cual abarcan los empleados de FabLab.

Respecto a las empresas de costura aunque todos los precios oscilaban sobre la misma cantidad se ha escogido el presupuesto de " La Yaya Costurera " debido a que ya se han empleado sus servicios y se ha comprobado su velocidad y profesionalidad del trabajo realizado. Dicha empresa ha ofrecido aplicar un 10 % de descuento sobre las producción de 100 unidades y un 20 % sobre 1.000 unidades.

Respecto a las telas fabricadas por " Sparkfun " se debe aprovechar la totalidad de la tela para producir la mayor cantidad de dispositivos posibles, llegando a obtener de una tela las formas necesarias para equipar 12 guantes. Por lo que para realizar 100 guantes se necesitan 9 telas y en el caso de 1.000 guantes 84 telas. Cabe destacar que se obtiene un 10 % de descuento a la hora de adquirir 1.000 unidades, en cuanto a los guantes no se obtiene algún descuento.

Otro factor a tener en cuenta es la proporción diestros y zurdos, siendo la cantidad de zurdos correspondiente al 10 % de la población mundial [29], para la cantidad sobrante de guantes zurdos se ha consultado y es factible la posibilidad de reconstruir guantes diestros empleando los zurdos como materia prima.

6.2.1. Producción 100 prototipos

Gracias a los presupuestos orientativos que han realizado las empresas nombradas con las que se ha contactado se ha ideado el coste orientativo de dicha labor y se adjuntan en la tabla 6.4.

Tabla 6.4. Presupuesto producción de 100 unidades.

Descripción (100 Uds.)	Costo total
Impresión PCB	5.600 €
Impresión Carcasa 3D	489 €
Guantes	80 €
Telas con reactancia	104,31 €
Costura	3.100 2.790 €
TOTAL	**9.063,31 €**

6.2.2. Producción 1.000 prototipos

Las mismas empresas facilitaron los costes de la fabricación de 1.000 unidades aparte del caso anterior descrito y se adjuntan los valores en la tabla 6.5, cabe destacar los precios reducidos que se aprecian en las impresiones de las PCB y de la compra de las telas con reactancia gracias a la gran cantidad de productos a adquirir.

Tabla 6.5. Presupuesto producción de 1.000 unidades.

Descripción (1.000 Uds.)	Costo total
Impresión PCB	13.900 €
Impresión Carcasa 3D	4.890 €
Guantes	800 €
Telas con reactancia	922,63 €
Costura	24.800 €
TOTAL	**45.312,63 €**

6.2.3. Punto de equilibrio

Una vez observado los presupuestos facilitados por las empresas y teniendo en cuenta la inversión inicial de la creación del proyecto calculada en la tabla 6.3. se debe calcular el precio de equilibrio por unidades para recuperar el sumatorio de los gastos iniciales y los gastos de producción de dichas cantidades. En la tabla 6.6. se calcula el precio por unidades produciendo 100 unidades, teniendo en cuenta que en dichas cantidades solo se plica un 10 % de descuento presupuestado por la empresa de costura, el resto no han ofrecido descuentos debido a la baja cantidad de unidades.

Tabla 6.6. Cálculo precio por unidad de la producción de 100 unidades.

Descripción (100 Uds.)	Costo
Inversión Inicial	11.444,96 €
Gastos Producción	9.063,31 €
PRECIO UNIDAD	**205,08 €**

En la tabla 6.7. se adjunta el cálculo del precio por unidad realizando la producción de 1.000 unidades, en dicho presupuesto se han producido descuentos en la impresión de las PCB, en la adquisición de las telas con reactancia y en la costura de ellas, con lo cual se han abaratado costes.

Tabla 6.7. Cálculo precio por unidades de la producción de 1.000 unidades.

Descripción (1.000 Uds.)	Costo
Inversión Inicial	11.444,96 €
Gastos Producción	45.312,63 €
PRECIO UNIDAD	**56,76 €**

Conclusiones

Una vez finalizado el proyecto se puede llegar a la conclusión de que se han cumplido los objetivos propuestos en un principio, se ha desarrollado un prototipo que permite medir las variables más importantes de la técnica del billar sin interferir en el juego habitual.

Se han implementado las telas con reactancia, se ha soldado toda la circuitería a una placa de fibra de vidrio y se ha protegido con una carcasa impresa en 3D, todo con el objetivo de reducir al máximo las dimensiones y el peso del hardware.

La transmisión de datos se ha realizado a través de comunicación Bluetooth por lo que brinda al usuario de una total libertad de movimientos y se han tenido en cuenta factores tan simples, pero importantes como la creación de dos versiones, una versión para usuarios diestros y otra para zurdos.

La interfaz gráfica del programa es Open-Source por lo que no es necesaria ninguna licencia para emplear dicho programa, los errores se visualizan de forma simple y gráfica empleando imágenes para su total comprensión. También se brinda la posibilidad de poder visualizar el historial de la jugada realizada, por lo cual el usuario puede centrarse en el juego y realizar una inspección de sus fallos una vez realizado el movimiento a estudiar.

Se ha realizado un presupuesto en masa contactando con varias empresas, tanto de impresión de PCB, como de piezas 3D, telas con reactancia, empresas de costura y productoras de guantes, llegando a la conclusión de cuál sería el punto de equilibrio, son observables los descuentos proporcionados por las empresas a partir de la producción de 1.000 unidades, ya que a menores cantidades no se aplican, con la excepción de la empresa de costura que aplica descuentos en ambos casos. Cuanto más competitivo se desee que sea el precio de salida al mercado mayor es la cantidad de producción requerida, ya que comparando las producciones de 100 y 1.000 unidades se ha disminuido un 70 % el precio del punto de equilibrio.

Bibliografía

[1] Real Academia Española, " *Billar* ". [En línea]. Disponible en: https://dle.rae.es/?id=5WASn7H [Acceso el 28 de Diciembre de 2018].

[2] Aprender Billar, " *La empuñadura en el billar* ". [En línea]. Disponible en: http://aprenderbillar.com/nivel/aprendiz/la-empunadura-en-el-billar/ [Acceso el 28 de Diciembre de 2018].

[3] Imagen tomada de: Coldeportes, " *Guía deportiva: billar-técnica* ", pp. 2, 2014. [En línea]. Disponible en: https://www.guao.org/sites/default/files/biblioteca/Billar.T%C3%A9cnica.pdf. [Acceso el 30 de Diciembre de 2018].

[4] The Snooker Gym, " *Cue Action Trainer* ". [En línea]. Disponible en: https://www.thesnookergym.com/cue-action-trainer. [Acceso el 26 de Mayo de 2019].

[5] PoolDawg, " *QMD3 Training Tool* ". [En línea]. Disponible en: https://www.pooldawg.com/product/cue-md3-training-tool. [Acceso el 26 de Mayo de 2019].

[6] I Challenger-Pérez, Y Díaz-Ricardo, " *El lenguaje de programación Python* ". [En línea]. Disponible en: https://www.redalyc.org/html/1815/181531232001/ [Acceso el 2 de Enero de 2019].

[7] Ramón Pallás Areny, " *Sensores y acondicionadores de señal* ". [En línea]. Disponible en : https://books.google.es/books?hl=es&lr=&id=Eevyk28_fVkC&oi=fnd&pg=PR11&dq=sensores&ots=JXkO59Evac&sig=Sdy9TZXyltNlIkImth3ocOBjbOA#v=onepage&q=sensores&f=true [Acceso el 2 de Enero de 2019].

[8] Arduino, " *¿ Qué es Arduino ?* ". [En línea]. Disponible en: https://www.arduino.cc/en/Guide/Introduction [Acceso el 8 de Enero de 2019].

[9] Imagen tomada de: 147 Academy, " *Grip* ". [En línea]. Disponible en: http://www.147academy.com/index.php?option=com_k2 &view=item&layout=item&id=371&Itemid=952 [Acceso el 8 de Enero de 2019].

[10] Imagen tomada de: Adaptado de: WikiHow, "*Cómo jugar billar como profesional*". [En línea]. Disponible en: https://es.wikihow.com/jugar-billar-como-profesional. [Acceso el 8 de Enero de 2019].

[11] Imagen tomada de: Moviltronics SAS, " *Sensor fuerza 1.5'' FSR 406* ". [En línea]. Disponible en: https://moviltronics.com.co /sensores/133-sensor-de-fuerza-15.html. [Acceso el 10 de Enero de 2019].

[12] Imagen tomada de: Interlink Electronics, " *FSR 406 Data Sheet* ". [En línea]. Disponible en: https://www.trossenrobotics.com/ productdocs/2010-10-26-datasheet-fsr406-layout2.pdf . [Acceso el 10 de Enero de 2019].

[13] Imagen tomada de: Sparkfun, " *EeonTex Conductive Fabric* ". [En línea]. Disponible en: https://www.sparkfun.com/products/14110. [Acceso el 14 de Enero de 2019].

[14] Imagen tomada de: Adafruit, " *Woven Conductive Fabric* ". [En línea]. Disponible en: https://www.adafruit.com/product/1168 [Acceso el 14 de Enero de 2019].

[15] Imagen tomada de: Amazon, " *Adafruit Stainless Thin Conductive Thread – 2 ply – 23 meter / 76 ft [640]* ". [En línea]. Disponible en: https://www.amazon.es/gp/product/B00CIVPYFA/ref=ppx_yo_dt_b_asin_title_o02_s00?ie=UTF8&psc=1. [Acceso el 16 de Enero de 2019].

[16] Imagen tomada de: Amazon, " *Sodial (R) Módulo MPU6050 de 3 ejes giroscopio + 3 Modulo Acelerómetro GY-521 Eje para Arduino* ". [En línea]. Disponible en: https://www.amazon.es/gp/product/B00K67X810/ref=ppx_yo_dt_b_asin_title_o04_s00?ie=UTF8&psc=1. [Acceso el 16 de Enero de 2019].

[17] Naylamp Mechatronics, " *Tutorial MPU6050, Acelerómetro y Giroscopio* ". [En línea]. Disponible en: https://naylampmechatronics.com/blog/45_Tutorial-MPU6050-Aceler%C3%B3metro-y-Giroscopio.html. [Acceso el 21 Enero 2019].

[18] Imagen tomada de: Banggood, " *5mm 940nm IR Infrarrojo Diodo Emisor Receptor LED* ". [En línea]. Disponible en: https://www.banggood.com/es/10pcs-5mm-940nm-IR-Infrared-Diode-Launch-Emitter-Receive-Receiver-LED-p-1099984.html?akmClientCountry=ES&cur_warehouse=CN. [Acceso el 21 de Enero de 2019].

[19] Imagen tomada de: Amazon, " *Arduino A000005 módulo digital y analógico i / o – Digital & Analog I / O modules* ". [En línea]. Disponible en: https://www.amazon.es/gp/product/B01873ITV8/ref=ppx_yo_dt_b_asin_title_o03_s00 ?ie=UTF8&psc=1. [Acceso el 24 de Enero de 2019].

[20] Imagen tomada de: Amazon, " *DSD Tech HC-05 módulo Bluetooth inalámbrico con botón de comunicación para Arduino* ". [En línea]. Disponible en: https://www.amazon.es/gp/product/B01G9KSAF6/ref=ppx_yo_dt_b_asin_title_o03_s00?ie=UTF8&psc=1. [Acceso el 24 de Enero de 2019].

[21] Imagen tomada de: Diotronic, " *Placa 120 x 80 fibra de vidrio virgen* ". [En línea]. Disponible en. https://diotronic.com/placa-120x80-fibra-vidrio-virg_6215/. [Acceso el 25 de Enero de 2019].

[22] Imagen tomada de: Eco Icon, " *Icono Mano, con, dedos, extendidas Gratis* ". [En línea]. Disponible en: https://icon-icons.com/es/icono/mano-con-dedos-extendidas/111475. [Acceso el 25 de Enero de 2019].

[23] Imagen tomada de: Diotronic, " *C11435CI Conmut. CI 1C2P SPDT* ". [En línea]. Disponible en: https://diotronic.com/c11435ci-conmut-ci-1c2p-spdt_2023/. [Acceso el 25 de Enero de 2019].

[24] Luis Llamas, " *Opciones para alimentar Arduino con baterías* ". [En línea]. Disponible en: https://www.luisllamas.es/alimentar-arduino-baterias/. [Acceso el 26 de Enero de 2019].

[25] Imagen tomada de: Diotronic, " *11445 Portapilas pila 9 V recto* ". [En línea]. Disponible en: https://diotronic.com/11445-portapilas-pila-9v-recto_554/. [Acceso el 26 de Enero de 2019].

[26] Medio Ambiente, " *La contaminación y la minería* ". [En línea]. Disponible en: https://www.medioambiente.net/la-contaminacion-y-la-mineria/. [Acceso el 15 de Mayo de 2019].

[27] A. Fernández , " *Basura Electrónica, un grave problema ambiental* ", 2014. [En línea]. Disponible en: http://www.consumer.es/web/es/medio_ambiente/urbano/2014/03/10/219489.php. [Acceso el 15 de Mayo de 2019].

[28] L. Hidalgo, Facultad de Ciencias de la Ingeniería, Universidad Tecnológica Equinoccial, " *La basura electrónica y la contaminación ambiental* ". [En línea]. Disponible en: https://www.researchgate.net/publication/277147221_La_basura_electronica_y_la_contaminacion_ambiental/fulltext/559caa5708ae898ed651fed1/277147221_La_basura_electronica_y_la_contaminacion_ambiental.pdf. [Acceso el 15 Mayo de 2019].

[29] Zurdos.cl, " Estadísticas ". [En línea]. Disponible en: http://www.zurdos.cl/estadisticas.html. [Acceso el 16 de Mayo de 2019].

Anexo A: Datasheets dispositivos empleados

A1. Tarjeta Arduino Nano

Specifications:

Microcontroller	Atmel ATmega168 or ATmega328
Operating Voltage (logic level)	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	8
DC Current per I/O Pin	40 mA
Flash Memory	16 KB (ATmega168) or 32 KB (ATmega328) of which 2 KB used by bootloader
SRAM	1 KB (ATmega168) or 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) or 1 KB (ATmega328)
Clock Speed	16 MHz
Dimensions	0.73" x 1.70"

Power:

The Arduino Nano can be powered via the Mini-B USB connection, 6-20V unregulated external power supply (pin 30), or 5V regulated external power supply (pin 27). The power source is automatically selected to the highest voltage source.

Input and Output

Each of the 14 digital pins on the Nano can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip.
External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Nano has 8 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the analogReference() function. Additionally, some pins have specialized functionality:

I²C: 4 (SDA) and 5 (SCL). Support I²C (TWI) communication using the Wire library (documentation on the Wiring website).

There are a couple of other pins on the board:

AREF. Reference voltage for the analog inputs. Used with analogReference().
Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and ATmega168 ports.

A2. Módulo MPU6050

A.2.1. Giroscopio

InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.4 Release Date: 08/19/2013

6 Electrical Characteristics

6.1 Gyroscope Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	FS_SEL=0		±250		°/s	
	FS_SEL=1		±500		°/s	
	FS_SEL=2		±1000		°/s	
	FS_SEL=3		±2000		°/s	
Gyroscope ADC Word Length			16		bits	
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(°/s)	
	FS_SEL=1		65.5		LSB/(°/s)	
	FS_SEL=2		32.8		LSB/(°/s)	
	FS_SEL=3		16.4		LSB/(°/s)	
Sensitivity Scale Factor Tolerance	25°C	-3		+3	%	
Sensitivity Scale Factor Variation Over Temperature			±2		%	
Nonlinearity	Best fit straight line; 25°C		0.2		%	
Cross-Axis Sensitivity			±2		%	
GYROSCOPE ZERO-RATE OUTPUT (ZRO)						
Initial ZRO Tolerance	25°C		±20		°/s	
ZRO Variation Over Temperature	-40°C to +85°C		±20		°/s	
Power-Supply Sensitivity (1-10Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		°/s	
Power-Supply Sensitivity (10 - 250Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		°/s	
Power-Supply Sensitivity (250Hz - 100kHz)	Sine wave, 100mVpp; VDD=2.5V		4		°/s	
Linear Acceleration Sensitivity	Static		0.1		°/s/g	
SELF-TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	1
GYROSCOPE NOISE PERFORMANCE	**FS_SEL=0**					
Total RMS Noise	DLPFCFG=2 (100Hz)		0.05		°/s-rms	
Low-frequency RMS noise	Bandwidth 1Hz to10Hz		0.033		°/s-rms	
Rate Noise Spectral Density	At 10Hz		0.005		°/s/ √ Hz	
GYROSCOPE MECHANICAL FREQUENCIES						
X-Axis		30	33	36	kHz	
Y-Axis		27	30	33	kHz	
Z-Axis		24	27	30	kHz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		256	Hz	
OUTPUT DATA RATE						
	Programmable	4		8,000	Hz	
GYROSCOPE START-UP TIME	**DLPFCFG=0**					
ZRO Settling (from power-on)	to ±1°/s of Final		30		ms	

1. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*

A.2.2. Acelerómetro

InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.4 Release Date: 08/19/2013

6.2 Accelerometer Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY						
Full-Scale Range	AFS_SEL=0		±2		*g*	
	AFS_SEL=1		±4		*g*	
	AFS_SEL=2		±8		*g*	
	AFS_SEL=3		±16		*g*	
ADC Word Length	Output in two's complement format		16		bits	
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/*g*	
	AFS_SEL=1		8,192		LSB/*g*	
	AFS_SEL=2		4,096		LSB/*g*	
	AFS_SEL=3		2,048		LSB/*g*	
Initial Calibration Tolerance			±3		%	
Sensitivity Change vs. Temperature	AFS_SEL=0, -40°C to +85°C		±0.02		%/°C	
Nonlinearity	Best Fit Straight Line		0.5		%	
Cross-Axis Sensitivity			±2		%	
ZERO-G OUTPUT						
Initial Calibration Tolerance	X and Y axes		±50		mg	1
	Z axis		±80		mg	
Zero-G Level Change vs. Temperature	X and Y axes, 0°C to +70°C		±35			
	Z axis, 0°C to +70°C		±60		mg	
SELF TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	2
NOISE PERFORMANCE						
Power Spectral Density	@10Hz, AFS_SEL=0 & ODR=1kHz		400		µg/√Hz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		260	Hz	
OUTPUT DATA RATE						
	Programmable Range	4		1,000	Hz	
INTELLIGENCE FUNCTION INCREMENT			32		mg/LSB	

1. Typical zero-g initial calibration tolerance value after MSL3 preconditioning
2. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*

A3. Módulo HC – 05 Bluetooth

2、Feature

- Sensitivity (Bit error rate) can reach -80dBm,The change range of output's power: -4 - +6dBm.
- Has an EDR module; and the change range of modulation depth: 2Mbps - 3Mbps.
- Has a build-in 2.4GHz antenna; user needn't test antenna.
- Has the external 8Mbit FLASH
- Can work at the low voltage (3.1V~4.2V). The current in pairing is in the range of 30～40mA.
- PIO control can be switched.
- This module can be used in the SMD.
- It's made through RoHS process.
- The board PIN is half hole size.
- Has a 2.4GHz digital wireless transceiver.
- Bases at CSR BC04 Bluetooth technology.
- Has the function of adaptive frequency hopping.
- Small (27mm×13mm×2mm)
- Peripherals circuit is simple.
- It's at the Bluetooth class 2 power level.
- Storage temperature range: -40 ℃ - 85℃， work temperature range: -25 ℃ - +75℃
- Any wave inter Interference: 2.4MHz， the power of emitting: 3 dBm.
- Bit error rate: 0. Only the signal decays at the transmission link, bit error may be produced. For example, when RS232 or TTL is being processed, some signals may decay.