UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DISEÑO DE UN DISPOSITIVO WEREABLE CON FUNCIÓN DE PODÓMETRO DE PRECISIÓN PARA SU APLICACIÓN EN CALZADO DEPORTIVO

TRABAJO FINAL DEL

Grado en Ingeniería Electrónica Industrial y Automática

REALIZADO POR

Néstor García García

TUTORIZADO POR

Nicolás Laguarda Miró

CURSO ACADÉMICO: 2019/2020

Agradecimientos

Lo primero quiero agradecer a mi familia por estar siempre a mi lado y apoyarme durante los años en aquello que necesitaba gran esfuerzo y constancia por mi parte.

A mi madre y a mi hermana Julia por no dudar un solo momento y por estar siempre ahí cuándo necesitaba de consejo y darle otro enfoque para seguir con la motivación que necesitaba.

A mis amigos más cercanos con los que siempre he disfrutado de los buenos momentos cuando más hacía falta, vuestro apoyo no cae en saco roto.

A mis compañeros de carrera con los que de alguna manera formamos un equipo capaz de resolver las situaciones más difíciles durante estos últimos años.

Finalmente quisiera agradecer a mi tutor Nicolás Laguarda Miró por su implicación y cobertura en la realización de este Trabajo de Fin de Grado a pesar de los tiempos difíciles que se presentan este 2020.

Muchas gracias a todos.

Resumen

La tecnología wearable ha evolucionado el concepto de medición del estado físico y medición del ejercicio realizado proporcionando funcionalidades de gran utilidad y precisión en comparación a su pequeño tamaño debido por ejemplo al reducido tamaño de sus baterías.

El proyecto desarrollado se centra en el diseño de un dispositivo wearable cuya función principal es el conteo de pasos del usuario, es decir, se trata de un dispositivo con función de podómetro cuya aplicación en el usuario se localiza en el calzado del mismo, algo poco habitual ya que este tipo de dispositivos se suelen situar en otras zonas del cuerpo como por ejemplo en la cintura o muñecas.

En esta memoria del Trabajo de Fin de Grado se expone el desarrollo de este dispositivo a través de la tecnología Arduino que gestiona la información recibida de un sensor acelerómetro y la implementación de una aplicación para el sistema operativo Android que servirá como interfaz de usuario que recibe los datos necesarios a través de conexión bluetooth.

Palabras clave: wearable, calzado, Arduino, acelerómetro, app, Android.

Abstract

Wearable technology has evolved the concept of fitness measurement and exercise measurement by providing very useful and accurate functionality compared to its small size due for example to the small size of its batteries.

The project developed focuses on the design of a wearable device whose main function is to count the user's steps, that is, a device with a pedometer function whose application to the user is located in the user's footwear, something unusual since this type of device is located usually in other areas of the body such as the waist or wrists.

In this Final Degree Project, the development of this device is presented through the Arduino technology that manages the information received from an accelerometer sensor and the implementation of an application for the Android operating system that will serve as a user interface that receives the necessary data through Bluetooth connection.

Keywords: wearable, footwear, Arduino, accelerometer, app, Android.

Índice

1. Objeto

1.1 Presentación.

En el presente documento se va a exponer el desarrollo de un dispositivo de tipo wearable con función de podómetro, es decir, un cuentapasos, con interfaz de usuario a través de una aplicación móvil para el sistema operativo Android, y aplicación en el calzado del usuario. En cuanto a la parte de hardware y software el proyecto va a estar basado en el microcontrolador Arduino Nano.

1.2 Objetivos

Durante los últimos años, la tecnología electrónica aplicada a vestimentas se ha desarrollado y evolucionado de manera significativa dando como resultado un amplio abanico de dispositivos versátiles a menudo con varias funcionalidades distintas. La tecnología wearable ha dejado de ser un complemento para convertirse en aparatos con una gran funcionalidad capaces de registrar a través de sensores nuestra presión sanguínea, la actividad física realizada y mucho más.

En este caso el objetivo de este proyecto es el desarrollo de un podómetro digital aplicado al calzado deportivo del usuario con el objeto de conseguir una alta precisión en los resultados a través de un sensor tipo acelerómetro, conteo de los pasos realizados y la distancia recorrida del usuario haciendo uso del sensor de ubicación del dispositivo móvil para su cálculo.

1.3 Justificación

Con el paso del tiempo la realización de ejercicio físico se ha vuelto cada vez una actividad más controlada por los deportistas, se tienen en cuenta con más detenimiento aspectos como la duración del ejercicio, la intensidad de este, la zona del cuerpo a ejercitar, las calorías quemadas, etc.

En este caso el proyecto busca la medición de los pasos realizados y la distancia recorrida a través del desarrollo de un dispositivo aplicable al calzado del usuario, ya que es en esta zona donde se puede obtener una mayor precisión en las medidas del ejercicio con respecto a otras distintas como puede ser la muñeca en el caso de los smartwatches, además de que sea aplicable y extraíble al calzado, ya que en el caso del calzado inteligente el sistema de medición se trata de la zapatilla en sí.

Por otro lado la implementación de una aplicación móvil para mostrar los resultados del ejercicio busca satisfacer la comodidad del usuario durante la realización de este, y que así no se encuentre pendiente de interactuar con el dispositivo situado en el calzado.

2. Generalidades

2.1 Tecnología wearable

La tecnología vestible o wearable hace referencia al conjunto de aparatos o dispositivos electrónicos (con microprocesador incorporado) que se incorporan en alguna prenda o parte de nuestro cuerpo interactuando de forma continua con el usuario y/o con otros dispositivos con la finalidad de realizar o medir una función concreta ya sea reflejar el estado de salud de un paciente, la seguridad de las personas que se exponen a determinados peligros, el entrenamiento de atletas, etc.

2.1.1 Historia de los wearables

Por wearable entendemos un dispositivo electrónico de pequeño tamaño, pero si nos remontamos a los orígenes podemos situarnos a finales del siglo XIX. En 1895 se da la invención del audífono eléctrico inventado por Miller Reese, el "Acusticón".

En cuanto a la era moderna, a finales de los años 70 se empezaron a comercializar los primeros relojes calculadora, además de la llegada de los walkman el cual cambiaría para siempre la forma de escuchar música de la sociedad. En 1981 Seiko comercializó el UC 2000 Wrist PC, considerado como el primer PC portátil pudiendo almacenar hasta 2kb de información.

Ilustracion1: Seiko UC 2000 Wrist PC

Con los años fueron avanzando este tipo de relojes inteligentes, pero fue a partir de la entrada del milenio cuando la tecnología daría el gran salto con la proliferación de prototipos comercializables. En 2004 se presentó en el Festival CyberArt de Bilbao la que sería el primer ejemplo de ropa tecnológica. Se trataba de la Hug Shirt, que permitía transmitir a distancia la sensación del tacto mediante un dispositivo bluetooth.

En los años siguientes la empresa desarrollaría la idea con el M Dress, un vestido que incorporaba una tarjeta SIM y permitía hacer y recibir llamadas sin usar un teléfono móvil; y con el TshirtOS, la primera prenda programable y controlable por una aplicación, en este caso de iOS.

A partir de ahí empezarían a llegar los dispositivos hoy de todos conocidos, el primer auricular bluetooth de Nokia en 2002, el iPod en 2001, el Fitbit en 2008, las Google Glass en 2013, entre los más conocidos.

2.1.2 Tendencias

El futuro de esta tecnología es muy esperanzador. Las compañías encargadas de crear hardware están constantemente trabajando en las nuevas generaciones de wearables, añadiendo y mejorando las funcionalidades. Hay artículos que declaran que este tipo de tecnología se ha adelantado 5 años a nuestro tiempo, debido a que con estos dispositivos vamos a poder hacer la compra, abrir puertas, etc.; aunque para que este tipo de funcionalidades puedan ser aprovechadas al cien por cien todavía falta mucho. Pese a las críticas negativas hacia los smartwatches se prevé que inunden el mercado en los próximos años.

Se espera en un futuro próximo en el que más del 50 por ciento de los deportistas habituales sean usuarios de dispositivos wearables capaces de monitorizar la actividad física debido al aumento del ratio de aceptación de este tipo de productos.

Los fines futuros de la tecnología wearable pueden llegar ser muy variados como por ejemplo gafas con más visión, cascos con sistema de crecimiento capilar, máscaras de belleza inteligentes, asistentes de las condiciones del hogar, etc.

2.1.3 El podómetro común

El podómetro común se trata de un dispositivo electrónico o electromecánico cuya funcionalidad es el conteo de los pasos que realiza una persona detectando el movimiento de sus caderas. Originalmente utilizado por los deportistas habituales, los podómetros se están popularizando como un medidor y motivador del ejercicio cotidiano. Normalmente este tipo de dispositivos se calibran para establecer la distancia de la zancada del usuario para poder así calcular la distancia recorrida (distancia = número de pasos x longitud de zancada), aunque es común que los podómetros registren erróneamente movimientos que no sean un paso como dar un salto o agacharse para atarse los cordones.

Ilustración 2: Podómetro básico Omron Aerobic

En el presente proyecto se busca encontrar una alta precisión en el conteo de los pasos aplicando el dispositivo podómetro al calzado del usuario para así eliminar el máximo número posible de conteos erróneos, a pesar de que el movimiento de saltos ligeros pueda considerarse como un paso en dirección vertical.

2.1.4 Wearables en calzado deportivo

La tecnología wearable con relación al calzado es muy reciente, los primero modelos se desarrollaron hace apenas cinco años, y en cuanto al sistema se trata de la zapatilla en sí, no de un dispositivo que se pueda aplicar a este.

El objetivo de este tipo de wearables es contabilizar el trabajo realizado o ayudar en el control de potencia que se está realizando, es decir, monitorizan la actividad física como distancia, velocidad, calorías quemadas etc. El atractivo de estas prendas es debido a que la tecnología incorporada es invisible y no se distinguen de las prendas convencionales.

Ilustración 3: Zapatilla inteligente Samsung IOFIT

La diferencia entre el dispositivo a desarrollar en este proyecto y este tipo de wearables se trata de la integración de el podómetro. En el calzado inteligente el sistema es la zapatilla en sí, mientras que en el podómetro a diseñar se trata de un dispositivo que se aplica a cualquier tipo de zapatilla convencional además de ser extraíble.

2.2 Hardware

Dentro de este apartado se van a exponer los elementos de hardware utilizados para llevar a cabo el proyecto, tanto el microcontrolador escogido y la historia a cerca de estos, tanto como del sensor tipo acelerómetro utilizado y el módulo bluetooth necesario para establecer la conexión con el smartphone.

2.2.1 Origen y aplicaciones de los microcontroladores

Inicialmente cuando no existían los microprocesadores, las personas se las ingeniaban en diseñar sus circuitos electrónicos y los resultados estaban expresados en diseños que implicaban muchos componentes y cálculos matemáticos. Un circuito lógico simple requería de muchos elemento electrónicos como transistores y resistencias. Al principio se creía que el manejo de un microprocesador era para aquellas personas con un coeficiente intelectual muy alto; por lo contrario con la aparición de este circuito integrado todo sería mucho más fácil de comprender y los diseños electrónicos serían mucho más pequeños y simplificados.

El microcontrolador fue inventado en Texas Instruments en la década de 1970 , alrededor del mismo tiempo que el primer microprocesador estaba siendo inventado en Intel. Los primeros microcontroladores eran simplemente microprocesadores con una función de memoria, como la memoria RAM y ROM. Más tarde, los microcontroladores se desarrollaron en una amplia gama de dispositivos diseñados para aplicaciones de sistemas embebidos específicos en dispositivos tales como automóviles, teléfonos móviles y electrodomésticos.

Ilustración 4: Primer microprocesador de Intel, el Intel 4004 (1971)

Durante la década de 1990, microcontroladores con memorias eléctricamente borrable y programable ROM (EEPROM), tales como la memoria flash, se hicieron disponibles . Estos microcontroladores pueden ser programados, borrar y volver a programar utilizando sólo señales eléctricas. Antes de los dispositivos eléctricamente reprogramables, los microcontroladores a menudo requieren programación especializada y hardware borrado, lo que requiere que el dispositivo se retira de su circuito, retardando el desarrollo de software y hacer el esfuerzo más caro. Con esta limitación eliminada, los microcontroladores fueron capaces de ser programados y reprogramados en un circuito de modo que los dispositivos con microcontroladores podrían ser actualizados con nuevo software sin tener que ser devueltos al fabricante.

Ilustración 5: microcontrolador mega AVR ATmega 16 AU

En resumen un microcontrolador se trata del elemento en un sistema que se encarga de recibir la información proporcionada por los sensores, la procesa y ordena las acciones necesarias a los actuadores del sistema según este haya sido programado para cumplir un propósito general.

2.2.2 Microcontrolador utilizado

Debido a las necesidades de el proyecto a realizar se ha escogido un microcontrolador de la compañía Arduino debido a la sencillez y versatilidad a la hora de programar y diseñar el circuito del dispositivo. Por otro lado, las placas de Arduino, su software y comunidad se han convertido en una plataforma de trabajo que ofrecen un aprendizaje relativamente rápido y sencillo.

De entre las diferente opciones teniendo en cuenta las necesidades para del proyecto y de las limitaciones de cada una se ha escogido la placa de desarrollo Arduino Nano original que hace uso del microcontrolador Atmega328P, el mismo que se utiliza en la placa Arduino Uno ofreciendo las mimas funcionalidades que este a excepción de incorporar una fuente de alimentación, la cual no se requiere para este proyecto.

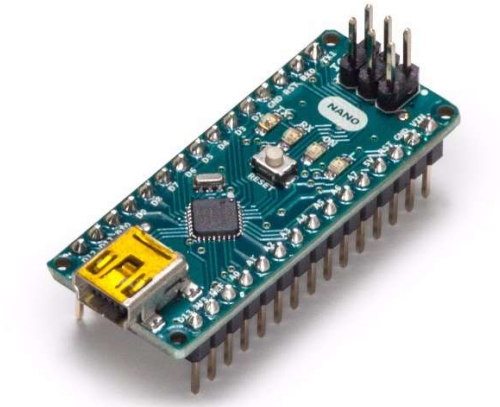


Ilustración 6: Placa Arduino Nano

Ilustración 7: Placa Arduino UNO

Se trata de una placa de tamaño muy reducido (18 x 45 mm) perfecta para el proyecto en cuestión, con una tensión de funcionamiento de 5V, sin embargo la tensión puede variar de 7 a 12 V. Además contiene 14 pines digitales, 8 pines analógicos, 2 pines de reinicio y 6 pines de potencia. Cada uno de estos pines digitales y analógicos tienen asignadas múltiples funciones, pero su función principal debe configurarse como entrada o salida (E/S), actúan como pines de entrada cuando está interconectados con los sensores.

Esta placa es bastante similar a otras placas Arduino disponibles en el mercado, pero su pequeño tamaño hace que destaque sobre las demás, como es el caso ya que el tamaño de los elementos electrónicos en este proyecto es de gran importancia (por ejemplo, las dimensiones del Arduino Uno son de 68.6 x 53.4 mm).

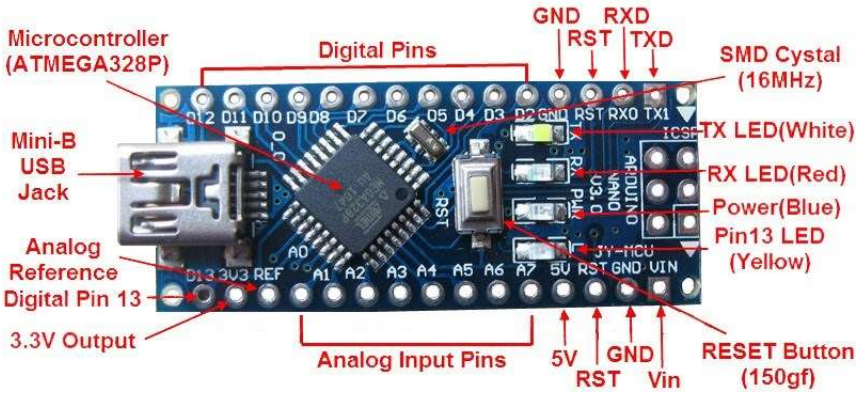


Ilustración 8: Descripción elementos Arduino Nano

A parte de las placas Arduino convencionales cabe mencionar las placas destinadas a usos concretos como son la placas Arduino LilyPad las cuales están destinadas para el desarrollo de proyecto e-textiles e incluso wearables, sin embargo no se ha escogido este tipo de las debido a su mayor desconocimiento en la comunidad y menores prestaciones, además de estar pensados para su integración en ropa, se está buscando un dispositivo de gran versatilidad y de fácil programación y tamaño reducido dando como mejor opción al Arduino Nano.

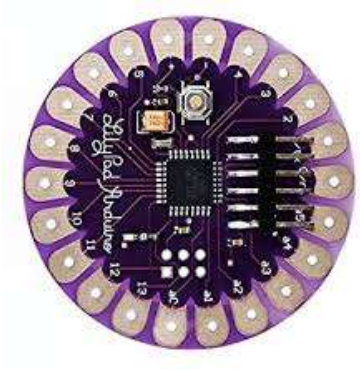


Ilustración 9: Placa Arduino Lilypad

Finalmente se va a presentar una tabla comparativa de las principales características de las placas Arduino más comunes:

Nombre	Procesador	Tensión	Frecuencia CPU	Entradas analógicas	Salidas analógicas	E/S digitales	Salidas PWM	SRAM (KB)	Flash (KB)	UART
Nano	ATmega328	5V/7-9V	16MHz	8	0	14	6	2	32	1
UNO	ATmega328	5V/7-12V	16MHz	6	0	14	6	2	32	1
Pro-mini	ATmega328	5V/7-12V	16MHz	6	0	14	6	1	16	1
LEONARDO	ATmega32u4	5V/7-12V	16MHz	12	0	20	7	2,5	32	1
MEGA	ATmega2560	5V/7-12V	16MHz	16	0	54	15	8	256	4
DUE	AT91SAM3X8E	3,3/7-12V	84MHz	12	2	54	12	96	512	4

Tabla 1: Comparativa de las especificaciones técnicas de 6 diferentes tipos de placas Arduino (incluyendo Nano)

2.2.3 Módulo acelerómetro ADXL345

Teniendo en cuenta el objetivo del proyecto, diseñar un podómetro de precisión utilizando como microcontrolador el anterior mencionado Arduino Nano, se ha escogido como sensor del sistema uno de tipo acelerómetro, concretamente el módulo acelerómetro ADXL345.

El modelo ADXL345 es un acelerómetro micromecanizado (MEMS) capacitivo de 3 ejes independientes (3 DOF) que puede ser fácilmente conectado a un procesador tipo Arduino. Detecta la aceleración en los ejes X, Y, Z; también es posible detectar la orientación del sensor, gracias a la acción de la fuerza de gravedad. Disponen de una alta resolución, pueden medir hasta +/- 16g perfecto para medir la aceleración dinámica ocasionada por algún movimiento o impacto, en este caso el impacto de la pisada del usuario.

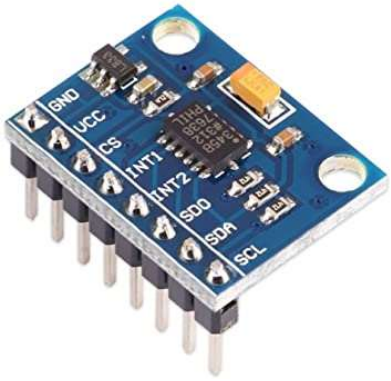


Ilustración 10: Módulo acelerómetro ADXL345

La comunicación puede realizarse tanto por bus SPI como por bus I2C, por lo que es sencillo obtener los resultados medidos. La tensión de alimentación es de bajo voltaje entre 2.0 y 3.6V (incluye además un regulador de voltaje para alimentarse directamente a los 5V del Arduino).

En cuanto al funcionamiento de un acelerómetro capacitivo este es capaz de convertir la aceleración a su respectivo valor de voltaje. Esto se consigue con el uso de condensadores (ilustración 12), conforme la aceleración mueve la esfera metálica del interior del sensor (ilustración 11), el condensador interno presenta cambios basados en el movimiento obteniendo como resultado un voltaje variable.

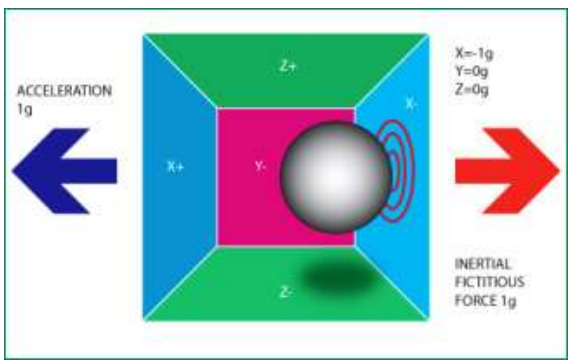


Ilustración 11: Comportamiento de la esfera metálica

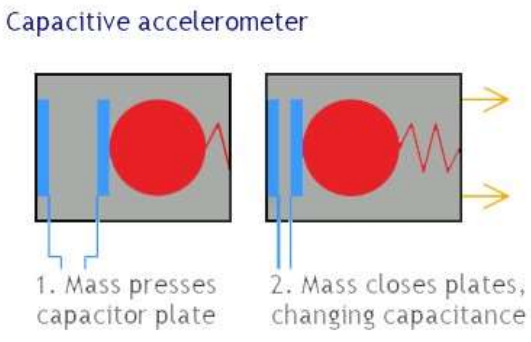


Ilustración 12: Funcionamiento del acelerómetro capacitivo

2.2.4 Módulo bluetooth HC-06

Debido a que se está desarrollado un sistema que conforma un dispositivo inalámbrico wearable con enlace a un aplicación móvil propia, es necesario establecer la conexión entre ambos elementos a través de un medio inalámbrico. El medio escogido es la conexión Bluetooth a través del módulo HC-06 conectado a la placa Arduino del sistema.

Este módulo bluetooth se comporta como esclavo esperando peticiones de conexión, es decir, si algún dispositivo se conecta, el módulo HC-06 transmite a este todos los datos que recibe de en este caso el Arduino y viceversa. En resumen la finalidad de este componente es pasar la información que recoge el acelerómetro antes y la transmite al dispositivo móvil, más concretamente envía la información a la aplicación Android para que la procese y lleve a cabo sus correspondientes funciones en torno al sistema de podómetro.

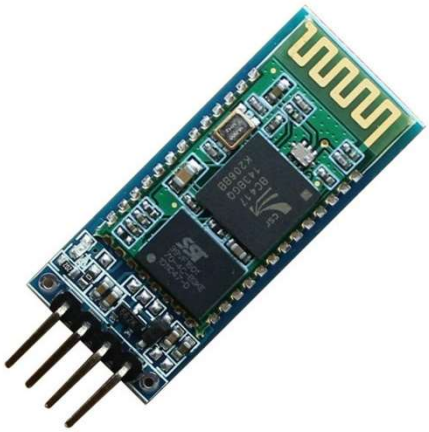


Ilustración 13: Módulo bluetooth HC-06

A diferencia de el módulo HC-05 el cual puede trabajar tanto como maestro como esclavo, el módulo HC-06 tan solo tiene función de esclavo. Sin embargo posee un menor número de conexiones y al no ser necesario el uso del módulo bluetooth como maestro la opción seleccionada es módulo HC-06.

En cuanto a las características principales de este módulo cabe decir que integra tecnología bluetooth 2.0 + EDR (Enhanced Data Rate), permite la configuración mediante comandos AT y posee una antena de PCB incorporada para la transmisión con un alcance de 5 a 10 m, perfecto para el uso que se le va a dar. En cuanto a seguridad posee autentificación por contraseña (por defecto 1234).

Posee un regulador de voltaje y 4 pines suministrando acceso a alimentación, tierra, entrada de datos y salida de datos (TXD y RXD). Su voltaje de operación es de 3.6 a 6 V.

Más adelante se desarrollarán más aspectos acerca de la conexión bluetooth en torno al sistema completo a parte de las características de hardware aquí mencionadas.

2.2.5 Hardware restante utilizado

Además de los tres componentes mencionados anteriormente, se ha hecho uso de otros componentes para el diseño del dispositivo.

Con la finalidad de que el dispositivo sea totalmente inalámbrico se ha utilizado una pila alcalina de 9V para alimentar la placa Arduino, además del uso de un conector de clip para una mejor conexión de la pila a la placa.

Ilustración 14: Pila alcalina de 9V

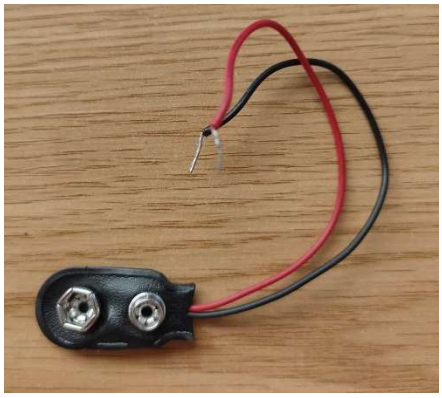


Ilustración 15: Clip conexión pila alcalina

Por otro lado se ha utilizado una pequeña protoboard de 170 pines para conectar los diferentes módulos y la pila ya que no se ha realizado el diseño de un circuito integrado esta era la mejor opción para tener una conexión de pequeño tamaño entre los elementos.

Además se han utilizado cables para conectar los elementos y se ha requerido del uso de estaño y de un soldador de estaño para fijar las conexiones del sensor acelerómetro.

Finalmente se ha utilizado una tira de tela con velcro situada en la cara inferior de la pila que se colocará bajo la protoboard para poder ajustar el dispositivo al calzado del usuario y se mantenga sujeto durante su uso.

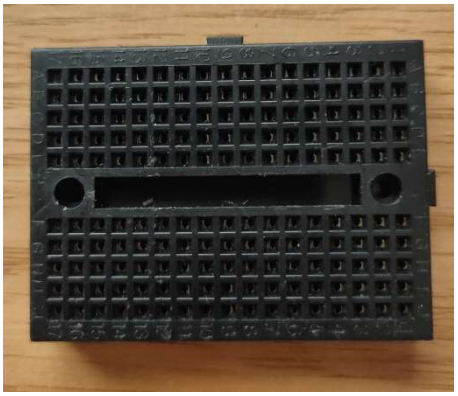


Ilustración 16: Protoboard de 170 pines

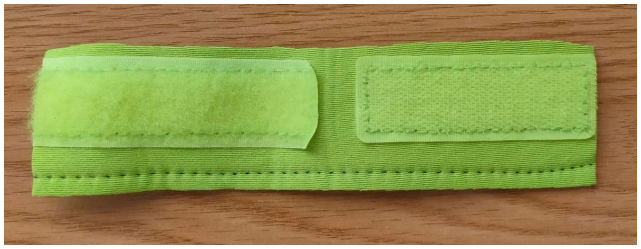


Ilustración 17: Tira de tela con velcro

2.3 Software

Una vez descrito los elementos hardware utilizados en el proyecto pasamos a exponer los programas de software utilizados para llevar a cabo la funcionalidad completa del proyecto. En este caso se ha hecho el uso de tres programas de software distintos para cumplir con los objetivos. En primer lugar se ha utilizado el entorno de desarrollo de la plataforma Arduino para programar la placa, en segundo lugar se ha utilizado el programa Fritzing para realizar esquemas de las conexiones del proyecto de manera preliminar antes de su montaje físico, y finalmente se ha desarrollado la aplicación móvil a través de la plataforma MIT app inventor.

En las siguiente secciones de software se van a presentar ejemplos ajenos al proyecto, ya que este será detallado en el apartado de desarrollo práctico más adelante en la memoria.

2.3.1 Arduino IDE

Con el objetivo de que se realicen la mediciones y se transmitan es necesario dictar la instrucciones que son necesarias que el microcontrolador opere junto al sensor y el transmisor, es decir, este debe ser programado.

Arduino IDE se trata del entorno de desarrollo integrado (Integrated Development Enviroment) de la plataforma Arduino que se basa en el lenguaje de programación Java, aunque también admite la utilización de los lenguajes C y C++ utilizando reglas especiales de estructuras de códigos. Dentro de las diferentes plataformas disponibles para programar la placa Arduino se ha escogido el software oficial de la marca, ya que posee una interfaz sencilla y se trata de un programa de fácil aprendizaje para principiantes. La versión utilizada es la versión 1.8.12.

Ilustración 18: Ventana información de versión de Arduino IDE

Ilustración 19: Logo comunidad Arduino

Comúnmente los programas creados en este entorno de desarrollo contienen una estructura general común en el apartado de desarrollo práctico. Los programas, llamados sketch poseen una estructura dividida en tres bloques diferenciados. A continuación se van a detallar la funcionalidad de estos bloques mostrando un sencillo ejemplo de programa sketch.

- Primero el apartado de inclusión de librerías y declaración de variables que se van a utilizar en el programa, así como de la declaración de la dirección del sensor a utilizar para establecer una correcta comunicación entre el sensor y el microcontrolador y la asignación de los pines de la placa que se van a utilizar para el proyecto. Las librerías son fundamentales para el funcionamiento de los sensores que se van a utilizar.

```
/**************/
/*  SEMAFORO */
/**************/

//** Definiciones **//
#include <Wire.h>
int rojo=2;      //definimos el valor del pin para el led rojo
int amarillo=4;  //definimos el valor del pin para el led amarillo
int verde=7;     //definimos el valor del pin para el led verde
```

Ilustración 20: Ejemplo apartado de inclusión de librerías y declaraciones en un sketch basado en un semáforo

- En segundo lugar se encuentra el bloque "void setup()" en el que se escriben las instrucciones que se realizan tan solo una vez cuando el programa se ejecuta. En este bloque debido a su funcionamiento se establece el comportamiento de los pines, es decir, su configuración, ya sea como entrada o salida (input u output), se inicializan la comunicaciones en serie necesarias haciendo uso comúnmente de los elementos que necesitan de las librerías para su correcto funcionamiento.

```
void setup() {
  pinMode(verde,OUTPUT);    //declaramos el pin verde como salida
  pinMode(amarillo,OUTPUT);//declaramos el pin amarillo como salida
  pinMode(rojo,OUTPUT);     //declaramos el pin rojo como salida
}
```

Ilustración 21: Ejemplo apartado "void setup()" con la declaración de las salidas en un sketch basado en un semáforo

- El último bloque del sketch se trata de la sección "void loop()" en el que finalmente se escribe el código del programa en sí tras haber declarado todos los aspectos anteriores. El comportamiento de este bloque se basa en un bucle infinito en el que todas las acciones escritas en su interior y las funciones antes definidas se ejecutan continuamente una vez tras otra mientras el programa se encuentre en funcionamiento, por lo que en el caso del proyecto se establecerán mediciones del sensor acelerómetro continuamente y así no se perderá en este caso el conteo de pasos.

```
void loop() {
 digitalWrite(verde,HIGH); //encendemos el led rojo
 delay(2000);              //esperamos 2 segundos
 digitalWrite(verde,LOW);  //apagamos el led rojo
 delay(500);               //esperamos medio segundo

 digitalWrite(amarillo,HIGH); //encendemos el led amarillo
 delay(2000);                 //esperamos 2 segundos
 digitalWrite(amarillo,LOW);  //apagamos el led amarillo
 delay(500);                  //esperamos medio segundo

 digitalWrite(rojo,HIGH); //encendemos el led verde
 delay(2000);              //esperamos 2 segundos
 digitalWrite(rojo,LOW);  //apagamos el led verde
 delay(500);               //esperamos medio segundo
}
```

Ilustración 22: Ejemplo apartado "void loop()" con las acciones a repetir en un sketch basado en un semáforo

2.3.2 Fritzing

Fritzing se trata de un software libre de diseño electrónico cuyo objetivo es proporcionar ayuda a los diseñadores para realizar simulaciones y esquemas de los montajes del proyecto para asegurar su funcionamiento antes de pasar a su montaje físico. Es un software muy utilizado en los proyectos de Arduino ya que posee bibliotecas con un gran número de componentes por lo que es muy recomendable hacer uso de este programa para comenzar con el diseño de un proyecto.

En este caso tan solo se han simulado las conexiones entre la placa el sensor y el módulo bluetooth, pero el programa también permite realizar el diseño de elementos PCB ampliando la versatilidad en el diseño.

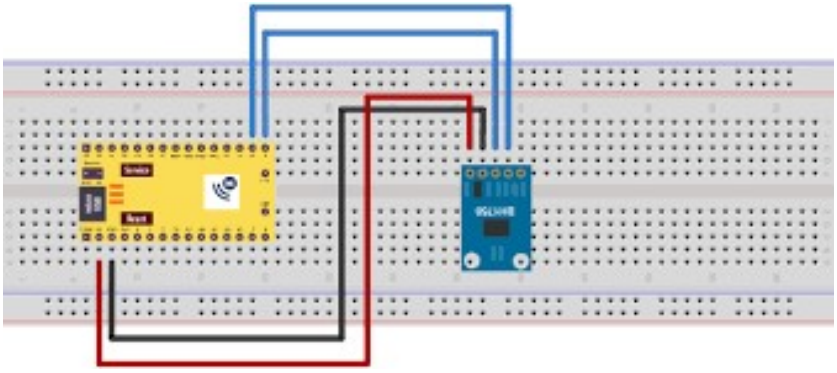


Ilustración 23: Ejemplo esquema circuito conexión sensor lumínico BH1750

Además de este programa se podría haber hecho uso de otros tipos de software que cubre sus funciones como es el caso de Proteus, sin embargo debido a la simplicidad de Fritzing y a que incluye la mayoría de los elementos relacionados con la plataforma Arduino se convierte en la opción más recomendable.

2.3.3 MIT App Inventor

Esta plataforma online se trata de un entorno de desarrollo de aplicaciones móvil desarrollado por Google y el Instituto Tecnológico de Massachusetts. Las aplicaciones funcionan con el sistema operativo Android. La característica principal de esta plataforma se trata de la programación de las aplicaciones a través de bloques funcionales permitiendo diseñar aplicaciones de una manera mucho más fácil y visual comparado con una programación estándar, sin embargo, como cabe esperar de este tipo de métodos de diseño de software, las aplicaciones creadas están limitadas a la simplicidad de la plataforma.

Ilustración 24: Logo MIT App Inventor

Además de la programación en bloques, la plataforma ofrece recursos para el diseño de la interfaz de la aplicación. Incluye cuadros de textos, botones, edición de tamaños de los elementos y las fuentes, imágenes, uso de la cámara, uso de los altavoces, uso de la grabadora etc. Además permite la inclusión de mapas, el uso de memorias de almacenamiento e incluir el uso de sensores del propio dispositivo móvil como son el reloj, termómetro, GPS, o giroscopio entre otros.

Entre las funcionalidades que ofrece en nuestro caso la más importante se trata del uso del servicio de conexión bluetooth del dispositivo para conectarlo al podómetro para, en este caso recibir la información acerca de los pasos realizados por el usuario.

Ilustración 25: Logo MIT Media Lab

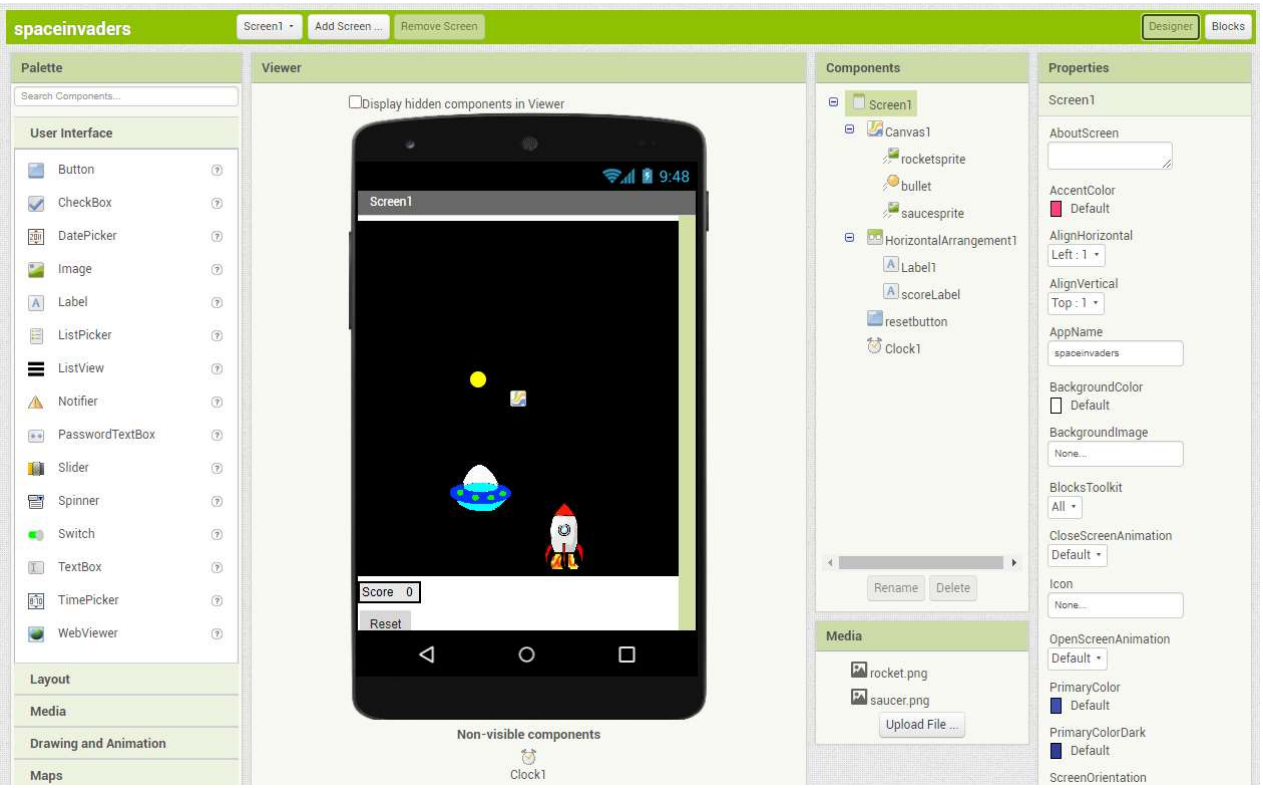


Ilustración 26: Ejemplo diseño de interfaz en MIT App Inventor (Juego Space invaders)

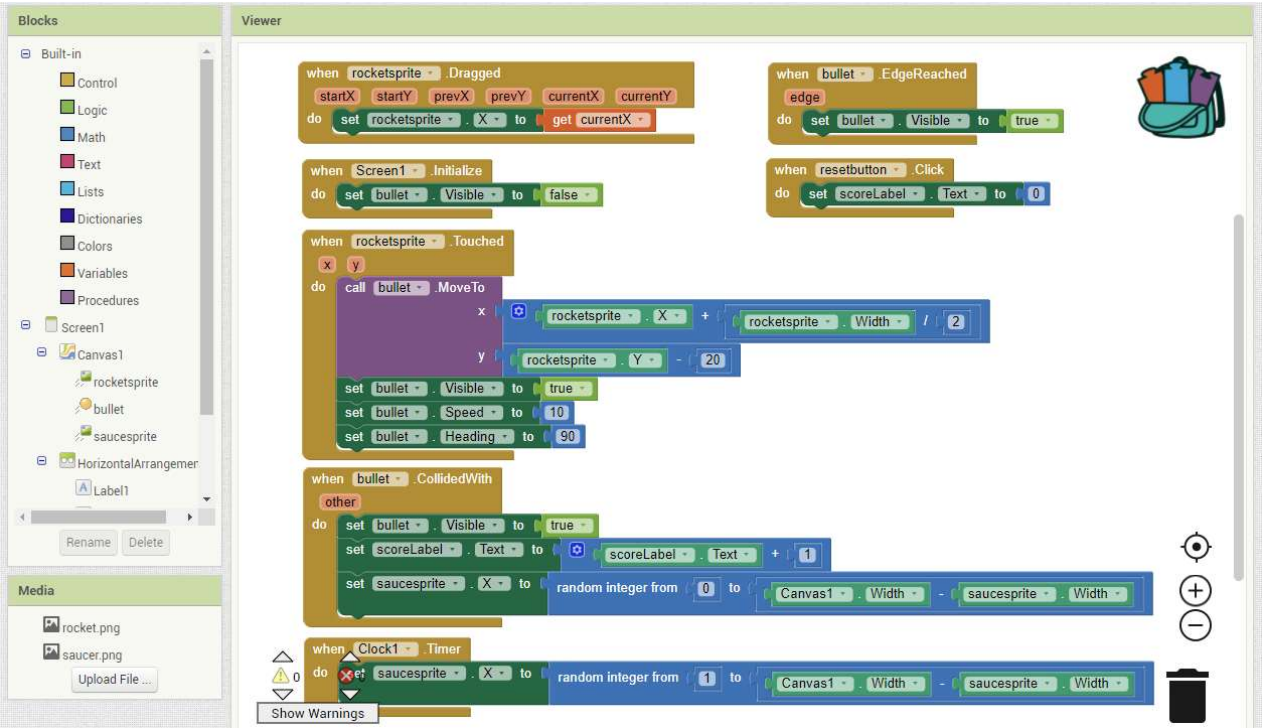


Ilustración 27: Ejemplo programación a través de bloques en MIT App Inventor (Juego Space invaders)

2.4 Sistemas de comunicación (Arduino)

En esta sección se van a presentar los diferentes tipos de sistema de comunicación de los que dispone el sistema Arduino. Debido a que Arduino dispone de una amplia variedad de diferentes componentes como sensores o displays, que se conectan entre sí, es necesario que posea diferentes medios para la transferencia o comunicación de los datos con el microcontrolador a estos componentes.

En el caso de este proyecto se han utilizado tres sistemas de comunicación, el puerto serie o comunicación serial, el bus I2C haciendo uso de la librería "Wire.h" y la conexión bluetooth para transferir los datos del microcontrolador al dispositivo móvil.

2.4.1 Puerto Serie

A comunicación a través del puerto serie es un componente fundamental en gran cantidad de proyectos de Arduino, se trata de uno de los elementos básicos en el aprendizaje de esta plataforma.

El término puerto denomina a los interfaces, físicos o virtuales, que permiten la comunicación entre dos ordenadores o dispositivos. Un puerto serie envía la información mediante una secuencia de bits. Para realizar la comunicación de datos es necesario utilizar al menos dos conectores, un conector de recepción Rx, y uno de transmisión Tx, sin embargo puede ser necesario utilizar otros conductores de referencia.

A diferencia de un puerto en serie, un puerto en paralelo envía información mediante múltiples canales de forma simultánea, para ello requiere de más conductores de comunicación. Esta conexión permite enviar mayor cantidad de bits simultáneamente, sin embargo presenta mayores errores de transferencia en torno a la velocidad del reloj y la longitud de los cables.

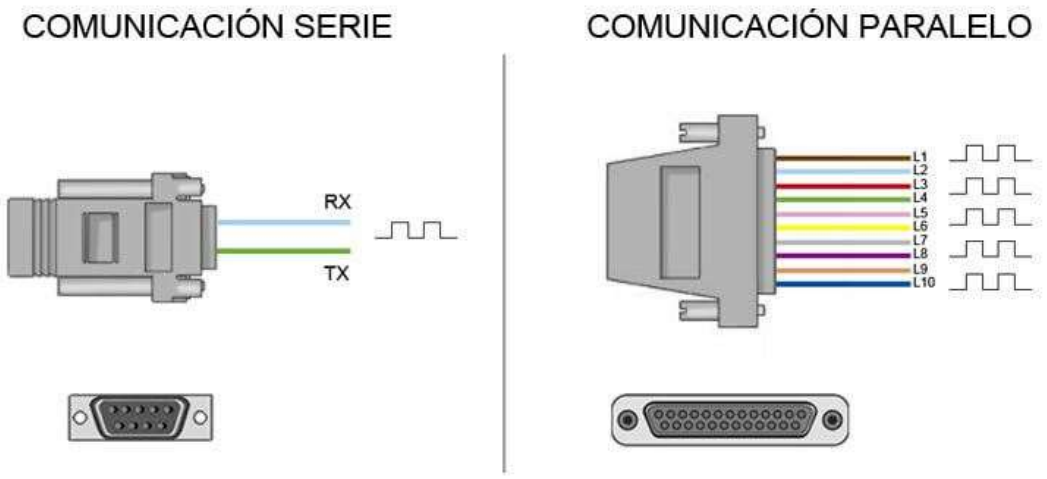


Ilustración 28: Esquema ilustrativo comunicación en serie y en paralelo

Originalmente ambos tipos de comunicación eran comúnmente utilizados, sin embargo, a medida que los procesadores se hicieron más rápidos los puertos en serie fueron desplazando progresivamente a los puertos paralelos en la mayoría de la aplicaciones. Existen una gran cantidad adicional de tipos de puertos serie, como por ejemplo el RS-485, I2C, SPI, Serial Ata, Pcie Express, Ethernet o FireWire, entre otros.

Los puertos serie de la placa Arduino están físicamente unidos a distinto pines, por lo que mientras usamos los puertos serie no podemos usar como entradas o salidas digitales los pines asociados con el puerto en serie en uso. La mayoría de los modelos de Arduino disponen de un conector USB o Micro USB conectado a uno de los puertos serie, simplificando el proceso de conexión con el ordenador. Para transmitir correctamente la información ambos sistemas deben utilizar un código común y estar configurados a la misma velocidad de transmisión, normalmente de 9600 baudios.

A través de la librería "Serial" en la programación del Arduino, resulta sencillo el envío de menajes desde la placa al ordenador utilizando el comando "Serial.print()", el cual envía los datos contenidos dentro del paréntesis mostrándose en el monitor serial IDE.

```
if(pasos > record){
    record = pasos;
  }
Serial.print("Pasos: ");
Serial.print(pasos);
Serial.print(" ; Record: ");
Serial.println(record);

Serial.print("Xa= ");
Serial.print(X_out);
Serial.print("   Ya= ");
Serial.print(Y_out);
Serial.print("   Za= ");
Serial.println(Z_out);

 ;
Serial.print("Tiempo transcurrido: ");
Serial.print(h, 0);
Serial.print("h ");
Serial.print(m, 0);
Serial.print("m ");
Serial.print(s, 0);
Serial.print("s ");
Serial.print(ms, 0);
Serial.println("ms");
Serial.println();
delay(200);
```

Ilustración 29: Líneas de código utilizando comando Serial.print para enviar los datos contenidos

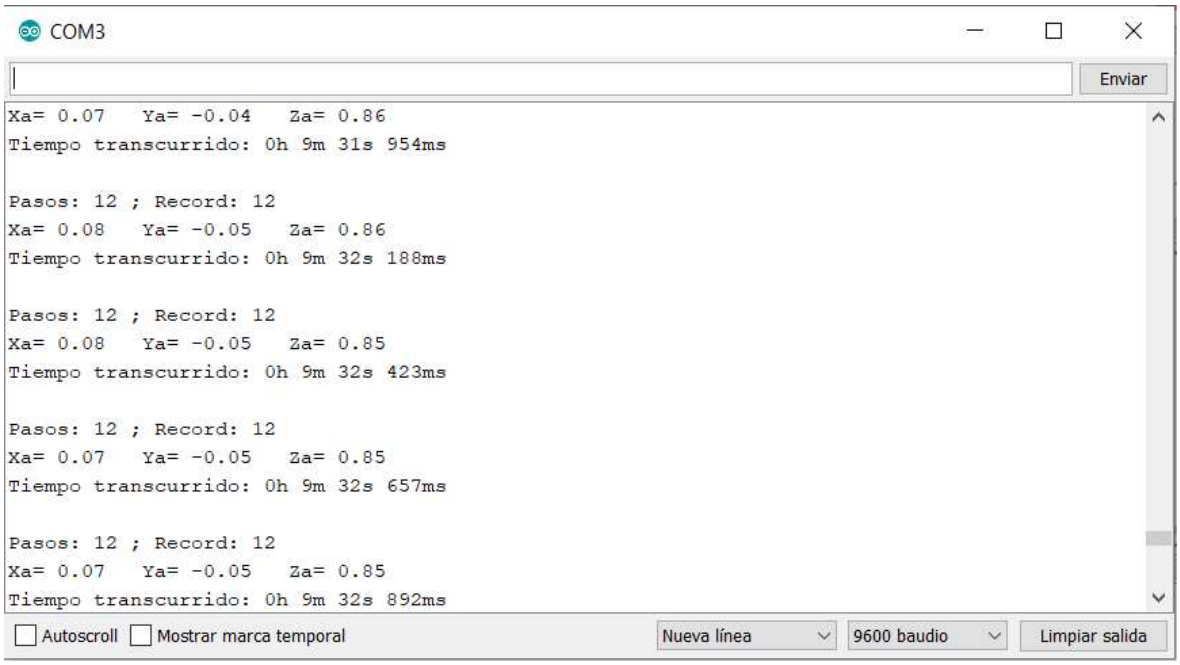


Ilustración 30: Monitor serie de Arduino mostrando los datos enviados a través de los comandos anteriores

2.4.2 El bus I2C

Es uno de los sistemas de comunicación que incluyen gran cantidad de dispositivos además del puerto serie y el bus SPI. El bus I2C requiere únicamente dos cables para su funcionamiento, uno para la señal del reloj CLK (SCL) y otro para el envío de datos SDA.

En el bus cada dispositivo dispone de una dirección propia que se utiliza para acceder al dispositivo en cuestión de forma individual. En general esta dirección debe ser única, en el caso de varios dispositivos con la misma dirección se tendría que cambiar la dirección o implementar un bus secundario.

La arquitectura de el bus I2C es de tipo maestro-esclavo, el dispositivo maestro inicia la comunicación con los esclavos permitiendo enviar y recibir datos de estos. Los esclavos no pueden iniciar la comunicación ni comunicarse entre sí.

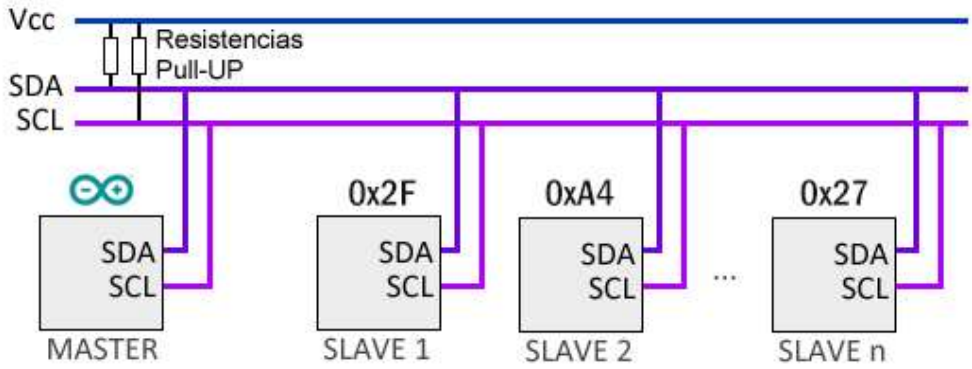


Ilustración 31: Esquema conjunto de una conexión bus I2C

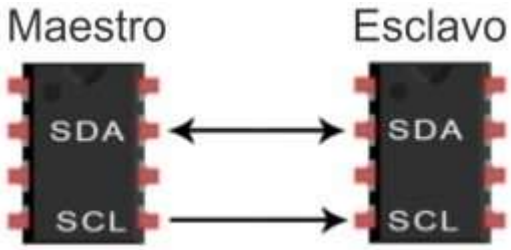


Ilustración 32: Esquema explicativo comportamiento maestro-esclavo

El bus I2C es síncrono, es decir, el maestro proporciona una señal de reloj que mantiene sincronizados todos los dispositivos del bus eliminando la necesidad de que cada dispositivo tenga su propio reloj. El protocolo necesita resistencias pull-up en las líneas Vcc, en el caso de Arduino, la plataforma posee la librería "Wire.h" que activa las resistencias internas de pull-up.

Gracias al funcionamiento del protocolo es posible conectar hasta un máximo de 112 dispositivos, ya que cada uno tiene asociada una dirección de 7 bits (128), las 16 direcciones restantes se reservan para usos especiales.

```
void setup() {
  Serial.begin(9600); // Iniciar comunicación serie para imprimir los resultados en el monitor serie
  Wire.begin(); // Iniciar la librería Wire
  // Establecer modo medicion al ADXL345
  Wire.beginTransmission(ADXL345); // Empezar comunicación con el dispositivo
  Wire.write(0x2D); // Accerder al registro POWER_CTL
  // Habilitar medición
  Wire.write(8); // (8dec -> 0000 1000 binario) Disponible bit D3 para medición
  Wire.endTransmission();
  delay(1000);
}
```

Ilustración 33: Uso de comandos Wire del bloque "void setup" para la comunicación con el acelerómetro

Por otro lado también se encuentra la posibilidad de utilizar el bus SPI que también tiene un comportamiento maestro-esclavo, sin embargo la comunicación se da a través de dos líneas independientes, se trata de una comunicación Full Duplex en la que el maestro puede recibir y enviar datos simultáneamente. Sin embargo en el caso del proyecto no se ha requerido su uso, se ha preferido utilizar tan solo el bus I2C debido a que se requiere la conexión de más pines en el bus SPI y el espacio es limitado.

2.4.3 Comunicación Bluetooth

La tecnología bluetooth suple la necesidad de establecer conexión inalámbrica entre dos sistemas diferentes. Su objetivo de desarrollo era obtener un transmisor de bajo consumo, de tamaño y coste reducidos que fuera capaz de integrarse en dispositivos de pequeño y gran tamaño.

Bluetooth tiene la ventaja de estar integrado de fábrica en la mayoría de los dispositivos portátiles, además no requiere de un sistema operativo concreto para su uso por lo que se convierte en uno de los mejores medios para comunicarse de forma inalámbrica con Arduino. Se trata del protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo, que requieren corto alcance de emisión y basados en transceptores de bajo costo.

Ilustración 34: Logo bluetooth

Los dispositivos pueden comunicarse cuando están dentro de su alcance, la comunicación se realiza a través de radiofrecuencia de forma que los dispositivos no tienen la necesidad de estar alienados o en el mismo espacio cerrado.

En nuestro caso la versión bluetooth utilizada es la 2.0 la cual posee un ancho de banda de 3 Mbit/s con un rango de 10 m sin añadir repetidores. Permite la posibilidad de transmitir en Full Duplex con un máximo de 1600 saltos por segundo.

Los módulos HC-05 y HC-06 poseen Bluetooth 2.0, sim embargo ha surgido el protocolo bluetooth 4.0 o Low Energy diseñado para reducir las necesidades de energía en los dispositivos que los usan. Se suele llamar también BLE por Bluetooth Low Energy, que disminuye el consume previo, sin embargo reduce la distancia de conexión.

Ilustración 35: Módulo Bluetooth 4.0 AT-09 BLE

3. Desarrollo práctico

En esta sección se va a exponer el montaje y desarrollo del proyecto completo tanto a nivel de hardware como de software en torno a sus características técnicas y a su principio de funcionamiento para dar como resultado el dispositivo wearable con función de podómetro junto a su aplicación móvil.

Comenzando por las conexiones de los elementos y los esquemas creados en fritzing mostrando finalmente el dispositivo podómetro completo, siguiendo con la explicación de la programación del Arduino y terminando con la explicación del desarrollo de la aplicación móvil a través de la plataforma MIT App Inventor así como de una vista general del proyecto.

3.1 Montaje del dispositivo

A continuación se van a mostrar los esquemas de montaje de los elementos principales del dispositivo así como de una descripción de sus conexiones y montaje físico. Al final de esta sección se mostrará el montaje final del dispositivo wearable con función de podómetro.

3.1.1 Conexión Arduino – acelerómetro

Para conectar el acelerómetro ADXL345 a la placa de Arduino se requieren establecer cuatro conexiones entre ambos elementos:

- La conexión de tierra GND.
- Conectar los cinco voltios de salida de la placa al terminal Vcc del acelerómetro para alimentarlo.
- Conectar el pin SDA del acelerómetro al pin A4 de la placa Arduino para habilitar el envío de datos (bus I2C).
- Conectar el pin SCL del acelerómetro al pin A5 del Arduino para sincronizar la señal de reloj.

La orientación de los ejes del acelerómetro cuando el dispositivo se encuentra en funcionamiento, es decir, en relación con el calzado y el movimiento del usuario, los ejes quedan de la siguiente manera:

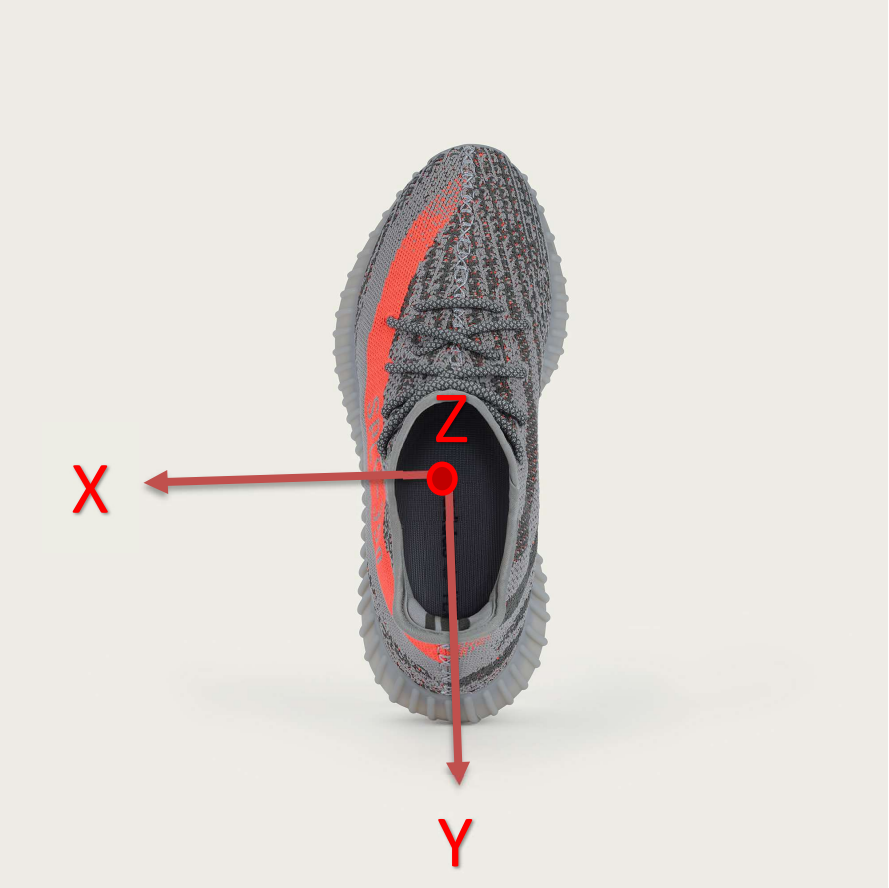


Ilustración 36: Disposición de los ejes de referencia del acelerómetro respecto al calzado

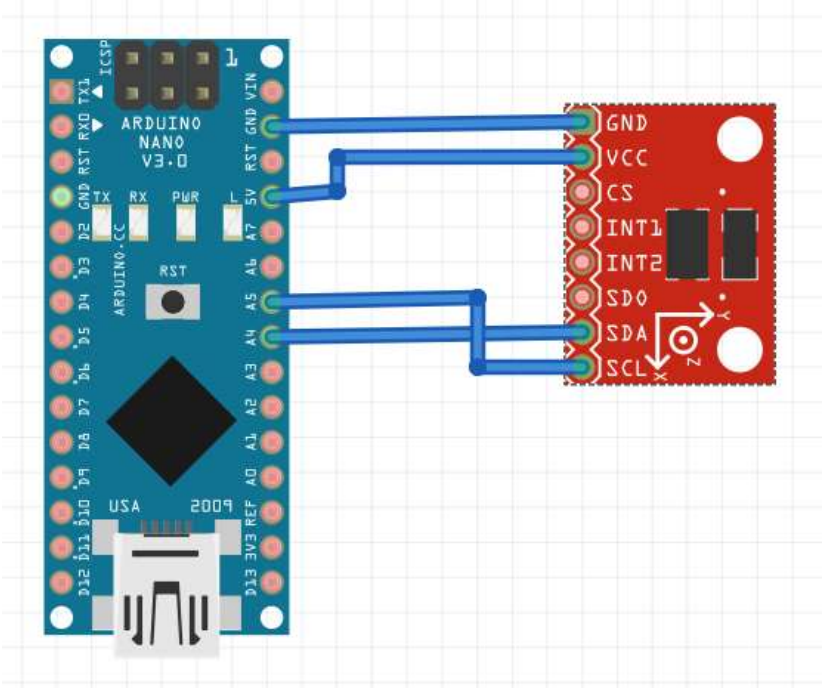


Ilustración 37: Esquema de conexiones entre acelerómetro ADXL345 y Arduino Nano

3.1.2 Conexión módulo bluetooth – Arduino

Al igual que sucede con el acelerómetro tan solo se requieren cuatro conexiones para enlazar correctamente el módulo bluetooth HC-06 con la placa Arduino:

- Conectar los cinco voltios de salida de la placa Arduino a el pin de alimentación Vcc del módulo.
- La conexión de tierra GND.
- Conectar el terminal TXD del módulo bluetooth al terminal RX0 de la placa Arduino para establecer la comunicación en serie.
- Conectar el terminal RTX del módulo bluetooth al terminal TX1 de la placa Arduino para establecer la comunicación en serie con el módulo HC-06 como esclavo y a la placa Arduino Nano como maestro.

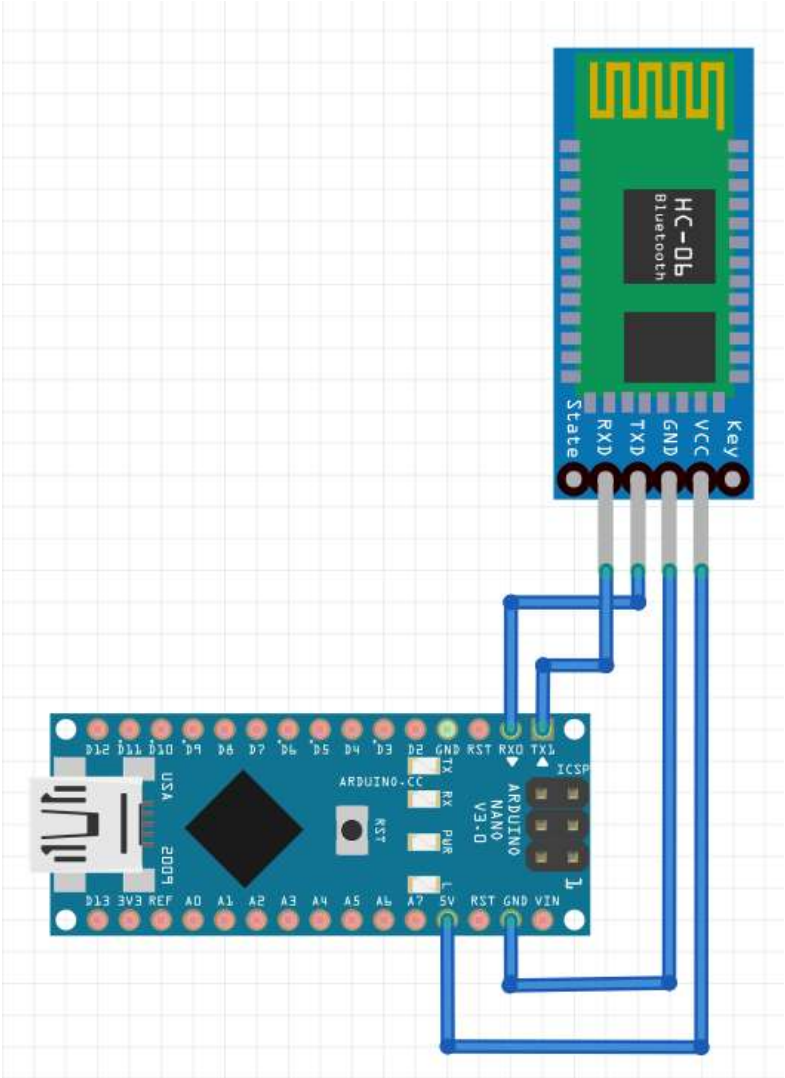


Ilustración 38: Esquema de conexiones entre módulo bluetooth HC-06 y Arduino Nano

3.1.3 Circuito completo

En la primera imagen se muestra el esquema de Fritzing del circuito completo de la parte electrónica del dispositivo podómetro incluyendo la batería de 9V para proporcionarle al dispositivo alimentación de manera inalámbrica, mientras que en la segunda imagen se muestra el montaje físico de el esquema de la primera imagen:

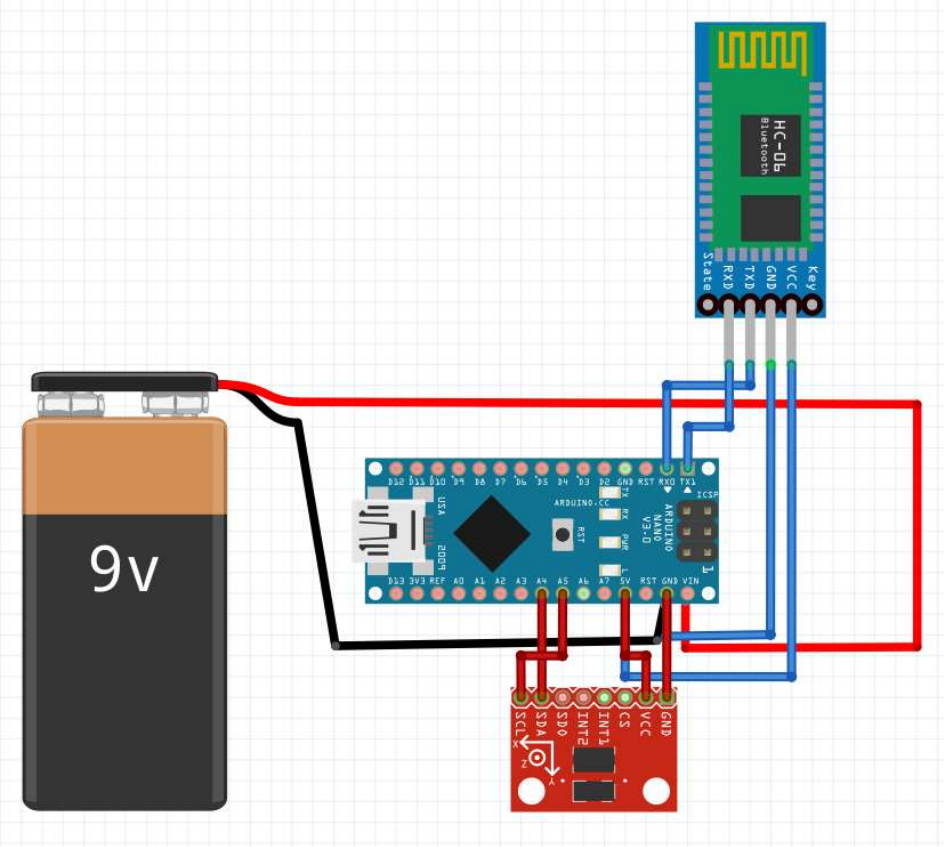


Ilustración 39: Esquema completo de la parte electrónica del dispositivo wearable

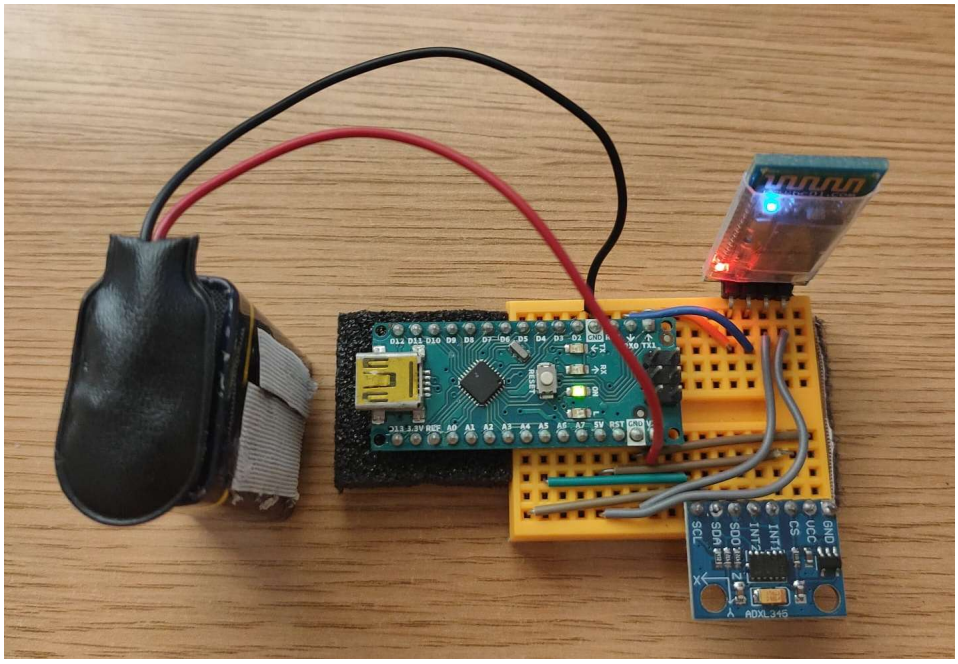


Ilustración 40: Montaje de la parte electrónica del dispositivo wearable

3.1.4 Montaje final del dispositivo

Como parte final de esta sección se muestra el montaje final del dispositivo wearable. Tras montar la parte electrónica se han colocado sus elementos de manera más compacta y se ha añadido la tira de tela con velcro para incluir el método de sujeción del dispositivo.

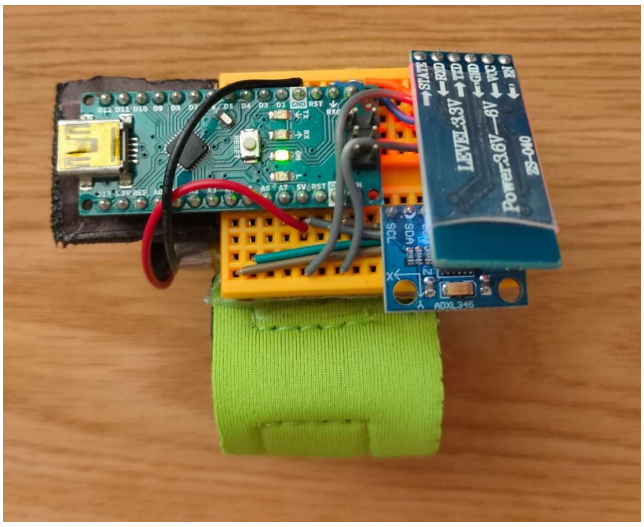


Ilustración 41: Vista superior del dispositivo completo

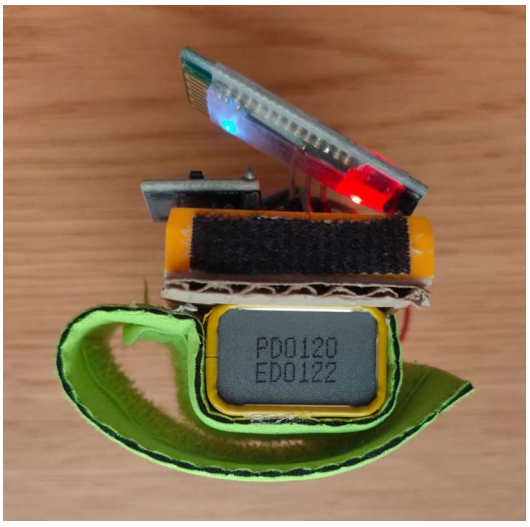


Ilustración 42: Vista lateral del dispositivo completo

Finalmente se muestra una imagen con el dispositivo acoplado al calzado utilizando la tira de tela con velcro:

Ilustración 43: Vista superior dispositivo acoplado a calzado

Ilustración 44: Vista lateral dispositivo acoplado a calzado

3.2 Código sketch Arduino

Una vez se ha expuesto toda la parte de hardware del proyecto pasamos a comentar la parte de software del dispositivo, es decir, de el código escrito en la placa Arduino, cómo se han utilizado los recurso descritos anteriormente para cumplir con las necesidades en este apartado.

Comenzando por la inclusión de librerías y la declaración de variables, en el sketch se han incluido las librerías "SoftwareSerial.h" para la comunicación en serie y la librería "Wire.h" utilizada para la comunicación I2C. A continuación se ha declarado la dirección del acelerómetro ADXL345 (dirección 0x53) y finalmente se han declarado las variables utilizadas para el conteo de pasos y reloj.

```
#include <SoftwareSerial.h>
#include <Wire.h>// Librería Wire - utilizada para la comunicación I2C
int ADXL345 = 0x53;// Dirección I2C del sensor ADXL345
float X_out, Y_out, Z_out;// Salidas
int pasos=0;
float h, m, s, ms;// Variables reloj
unsigned long over,elapsed;
```

Ilustración 45: Inclusión de librerías y declaración de variables en el sketch de Arduino

En el bloque de "void setup()" se ha iniciado la comunicación serie para el monitor serie a 9600 baudios, se ha iniciado la librería Wire y se ha iniciado la comunicación con el acelerómetro además de habilitar la medición a través del uso de comandos propios de la librería Wire.

```
void setup() {
  Serial.begin(9600);// Iniciar comunicación serie para imprimir los resultados en el monitor serie
  Wire.begin();// Iniciar la librería Wire
  // Establecer modo medicion al ADXL345
  Wire.beginTransmission(ADXL345);// Empezar comunicación con el dispositivo
  Wire.write(0x2D);// Accerder al registro POWER_CTL
  // Habilitar medición
  Wire.write(8);// (8dec -> 0000 1000 binario) Disponible bit D3 para medición
  Wire.endTransmission();
  delay(1000);
}
```

Ilustración 46: Bloque void setup en el sketch de Arduino

Finalmente en el bloque “void loop ()” se encuentran las partes más fundamentales del funcionamiento del programa.

Se han redactado las líneas de código que representan el funcionamiento del reloj haciendo uso de la función “millis()” para calcular las horas minutos y segundos transcurridos desde el comienzo del funcionamiento del programa.

A continuación pasamos a la lectura de datos del acelerómetro comenzando con el uso de 6 registros para almacenar los valores de los ejes X, Y, Z leídos, los cuales para obtener un valor entre +/- 2g es necesario dividir el valor registrado entre 256.

```
void loop() {

  elapsed = millis();
  h = int(elapsed / 3600000); // Cálculo horas reloj
  over = elapsed % 3600000;
  m = int(over / 60000); over = over % 60000; // Cálculo minutos reloj
  s = int(over / 1000); ms = over % 1000; // Cálculo segundos reloj


  // === Leer datos acelerómetro === //

  Wire.beginTransmission(ADXL345);
  Wire.write(0x32);// Comenzar con el registro 0x32 (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(ADXL345, 6, true);// Leer 6 registros, cada eje se almacena en 2 registros

  X_out = ( Wire.read()| Wire.read() << 8);// Valor eje X
  X_out = X_out/256;// Para un rango de +-2g, necesitamos dividir los valores entre 256, según la datasheet
  Y_out = ( Wire.read()| Wire.read() << 8);// Valor eje Y
  Y_out = Y_out/256;
  Z_out = ( Wire.read()| Wire.read() << 8);// Valor eje Z
  Z_out = Z_out/256;
```

Ilustración 47: Parte del bloque void loop para cálculo reloj y cálculo valores acelerómetro

En la última parte del bloque de “void loop ()” se han escrito las líneas para enviar los datos en el monitor serie. Si se manda un ‘1’ como comando al monitor serie se mostrarán los pasos realizados desde que se ha arrancado el programa además del tiempo transcurrido.

Finalmente se ha utilizado una estructura “if ()” para establecer el rango de valor mínimo para que el movimiento detectado por el acelerómetro sea interpretado como un paso (teniendo en cuenta los valores del eje Y y el eje Z), y a continuación envíe un ‘1’ al monitor serie o en el caso de no estar conectado al ordenador, mediante la comunicación bluetooth envíe el ‘1’ a la aplicación y esta lo interprete como un pase realizado y así contabilice los pasos totales un su interfaz.

```
  if(Serial.read() == '1'){ // Mostrar pasos totales y tiempo transcurrido al usar el comando '1'
   Serial.print("Pasos: ");// Mostrar pasos totales
   Serial.println(pasos);
   Serial.print("Tiempo transcurrido: "); // Mostrar tiempo transcurrido
   Serial.print(h, 0);
   Serial.print("h ");
   Serial.print(m, 0);
   Serial.print("m ");
   Serial.print(s, 0);
   Serial.print("s ");
   Serial.print(ms, 0);
   Serial.println("ms");
   Serial.println();
   delay(200);

  }

  if(Z_out > 1.5  || Y_out > 0.8)
  {
    pasos = pasos +1;
    delay(200);
  Serial.println("1");//Enviar '1' al monitor serie
  delay(100);

  }
}
```

Ilustración 48: Parte del bloque void loop con la escritura de datos en el monitor serie e interpretación de conteo de pasos

Ilustración 49: Ventana monitor serie mostrando la señal cada paso, su conteo y el tiempo transcurrido

3.3 Diseño aplicación móvil

Para completar los objetivos de funcionalidad del proyecto se ha desarrollado una App móvil para el control manual y visualización de los datos recogidos del dispositivo podómetro. El propósito principal de esta aplicación es disponer de una interfaz que muestre al usuario los datos y resultados esenciales de un dispositivo podómetro para poder así tener un ejercicio más controlado y personalizado.

Como se ha mencionado anteriormente la aplicación ha sido desarrollada a través de la plataforma MIT App inventor la cual crea un archivo apk de la aplicación desarrollada para el sistema operativo Android. La App se ha nombrado “WearPod” (podómetro wearable).

Su logo es el siguiente:

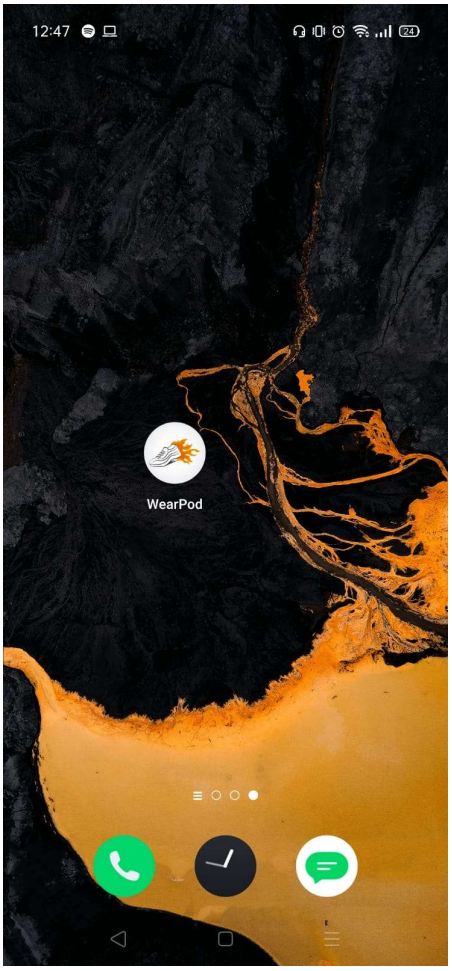


Ilustración 50: Icono app pantalla móvil

Ilustración 51: Logo aplicación WearPod

A continuación se van a exponer los elementos de la interfaz de usuario y sus funcionalidades además de los bloques de programación utilizados para el diseño de la app.

3.3.1 Interfaz y funcionalidades

La aplicación se resume en una sola interfaz en la que se encuentran todos los datos y funciones de que dispone para hacer de esta algo sencillo de manejar y comprender.

Las funcionalidades son las siguientes:

- Botón de selección de dispositivo ("Sel.BT") a conectar a la aplicación (podómetro) mediante bluetooth. Se selecciona el dispositivo "HC-06" del listado emergente.
- Botón "Salir" para cerrar la aplicación.
- Mensaje superior mostrando el estado de conexión con el dispositivo, conectado o desconectado.
- Botón "Start" (verde) para comenzar la medición del ejercicio. Se inician el reloj, el conteo de pasos y la distancia recorrida.
- Botón "Stop" (amarillo) para detener la medición del ejercicio. Se detienen el reloj, y el conteo de pasos.
- Botón "Reset" (rojo) para reiniciar los datos, pone a "0" el reloj, el conteo de pasos y la distancia recorrida.
- Display de reloj mostrando el tiempo transcurrido del ejercicio.
- Display del conteo de pasos.
- Display de la distancia recorrida en el ejercicio.
- Display de la marcas más altas (record) de el número de pasos realizados y la distancia recorrida. Cuando se está superando el record, este aparece en naranja.

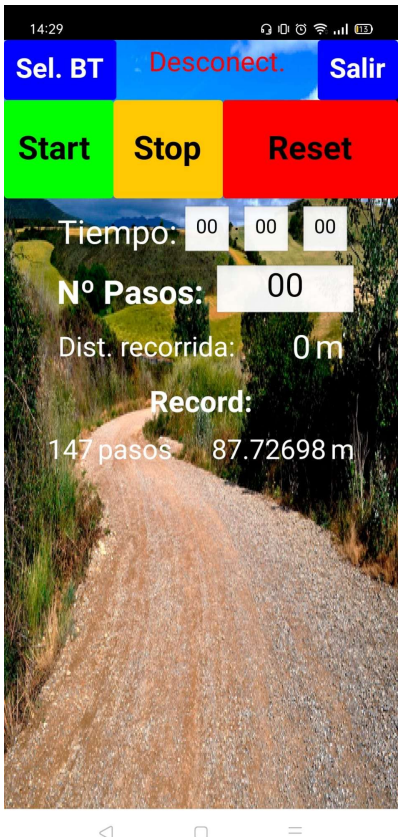


Ilustración 52: Interfaz aplicación con dispositivo desconectado

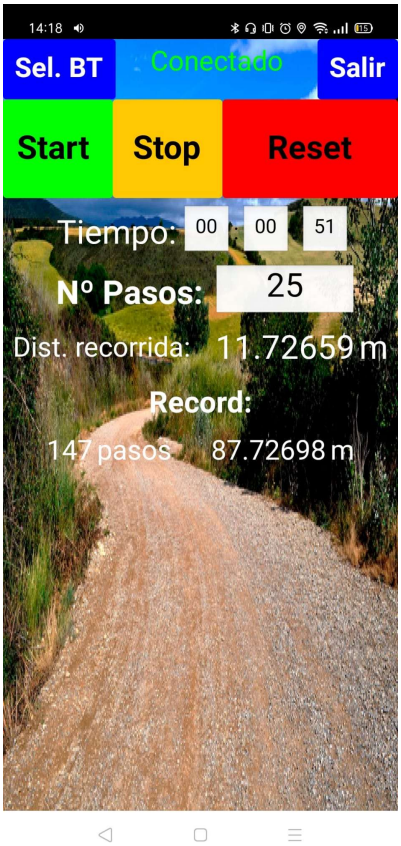


Ilustración 53: Interfaz aplicación con dispositivo conectado

En el caso de los displays del reloj y del número de pasos, ambos valores pueden ser modificados por el usuario según convengan sus necesidades.

3.3.2 Bloques de diseño

En esta sección se van a comentar las estructuras creadas en la programación de bloques de la plataforma para crear las funcionalidades de la aplicación.

Comenzando por por la parte superior de la intefáz nos encontramos con los botones de selección de dispositivo bluetooth , “Sel.BT” y botón “salir” para cerrar la aplicación.

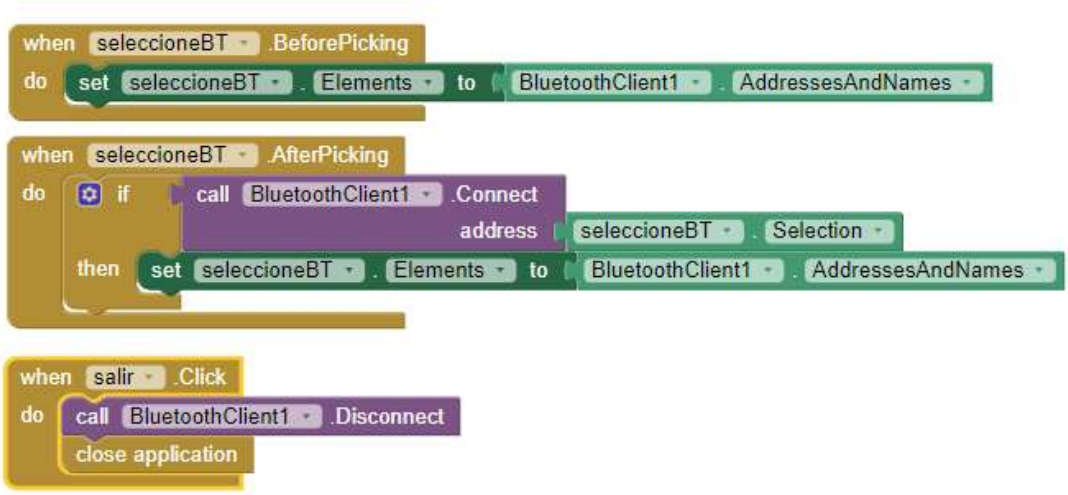


Ilustración 54: Programación de bloques de los botones Sel.BT y Salir

El primer bloque establece los elementos del botón a la lista de dispositivos bluetooth disponibles. El segundo bloque trata las acciones tras pulsar el botón, se establece la conexión con el dispositivo seleccionado. El tercer bloque representa el botón “salir”, una vez pulsado se desconecta el dispositivo bluetooth y se cierra la aplicación.

En el diseño de la aplicación se ha requerido el uso de dos relojes distintos, uno representa las funciones a llevar a cabo cuando el reloj de la interfaz está en funcionamiento y el segundo se utiliza para controlar el estado de conexión del dispositivo y mantener las marcas de record escritas en la interfaz, es decir funciones que se pregunta el programa constantemente.

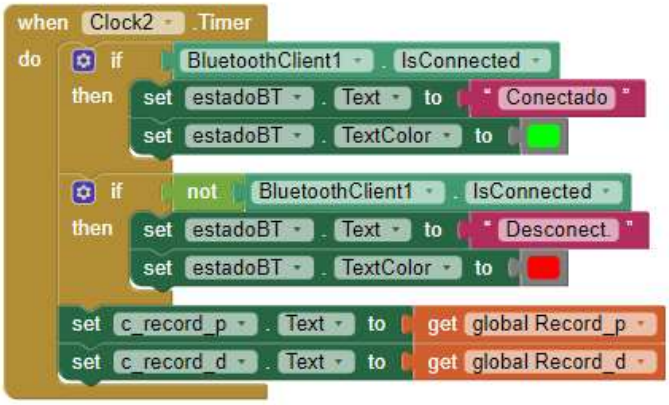


Ilustración 55: Programación de bloques del segundo reloj

Las estructuras “if” del bloque se preguntan si el dispositivo está conectado, en el caso de estarlo el cuadro de texto superior de la interfaz se mostrará como “Conectado” en color verde, mientras que si no lo está se mostrará como “Desconect.” En color rojo. Las dos últimas líneas del bloque se ocupan de escribir el valor de las variables de los records de pasos y distancia en los cuadros de textos correspondientes de la interfaz.

A continuación pasamos a comentar la programación de los botones referentes al reloj principal, es decir, los botones “Start”, “Stop” y “Reset”.

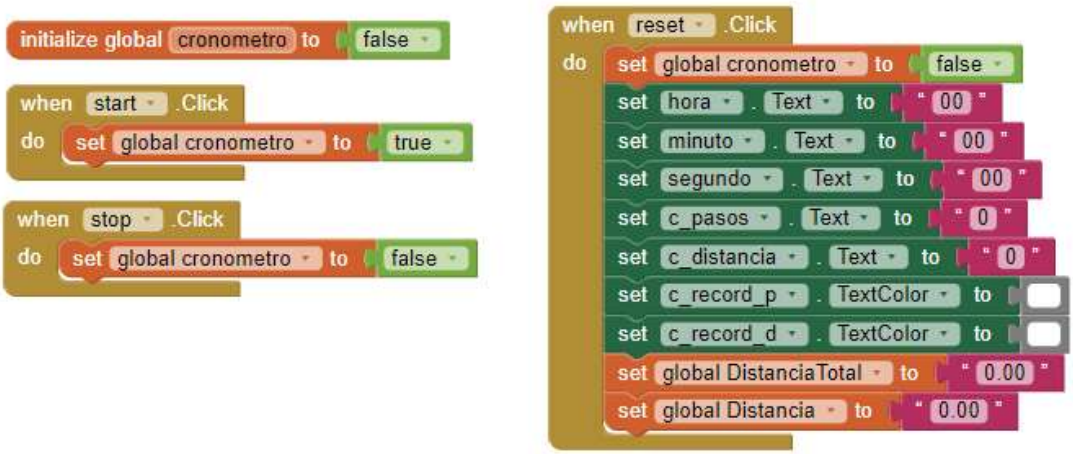


Ilustración 56: Programación de bloques de los botones Start, Stop y Reset

La primera línea declara una variable booleana “cronometro” como falsa, representado si el reloj principal está en marcha o no.

El siguiente bloque representa el botón “Start”, cuando se presiona la variable cronómetro pasa a verdadera; mientras que el siguiente bloque representa el botón “Stop” que al ser presionado la variable cronometro pasa a estado falso.

El bloque de la derecha representa todas las consecuencias de presionar el botón “Reset”, la variable cronometro cambia su valor a falso, se establecen a ‘0’ los valores del reloj, el contador de pasos y la distancia recorrida; el texto de las casillas de record de la interfaz pasan a color blanco ya que si se estaba batiendo el record estas casillas aparecen en color naranja. Las dos últimas líneas ponen a ‘0’ el valor de las variables “DistanciaTotal” y “Distancia”, utilizadas para el cálculo de la distancia recorrida por el usuario.

La siguiente parte del diseño de bloques se trata del núcleo principal de la aplicación, el cual gira en torno al reloj principal. Este bloque puede dividirse en dos partes, el conteo de pasos y funcionamiento del reloj, y el registro de los valores de record y el salvado de estos valores en la memoria de la aplicación.

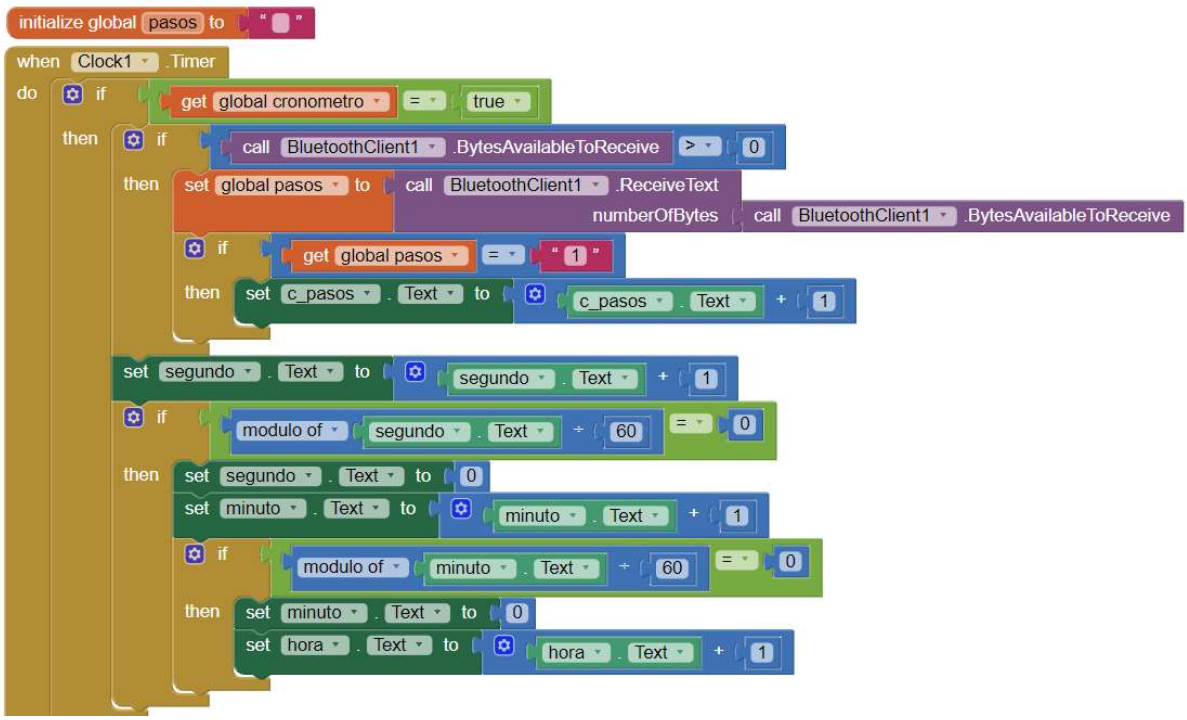


Ilustración 57: Programación de bloques del funcionamiento del reloj principal y el conteo de pasos

En la primera línea se inicializa una variable “pasos” sin valor definido para el número de pasos realizados.

La estructura del reloj principal se pone en funcionamiento cuando la variable “cronometro” se encuentra en estado verdadero. Dentro de esta afirmación se encuentra el bloque para el conteo de pasos que se activa cuando detecta que hay información recibida del módulo bluetooth del dispositivo podómetro, en ese caso establece este valor en la variable “pasos” y cuando la información recibida se trata de un ‘1’ el texto de la interfaz que representa la cifra de pasos realizados suma 1.

El resto de las líneas componen el comportamiento del reloj, el texto de los segundo va aumentando su valor de uno en uno, cuando llega a 60 los minutos suman uno y este se resetea y cuándo los minutos llegan a 60 , las horas suman uno y los anteriores se resetean.

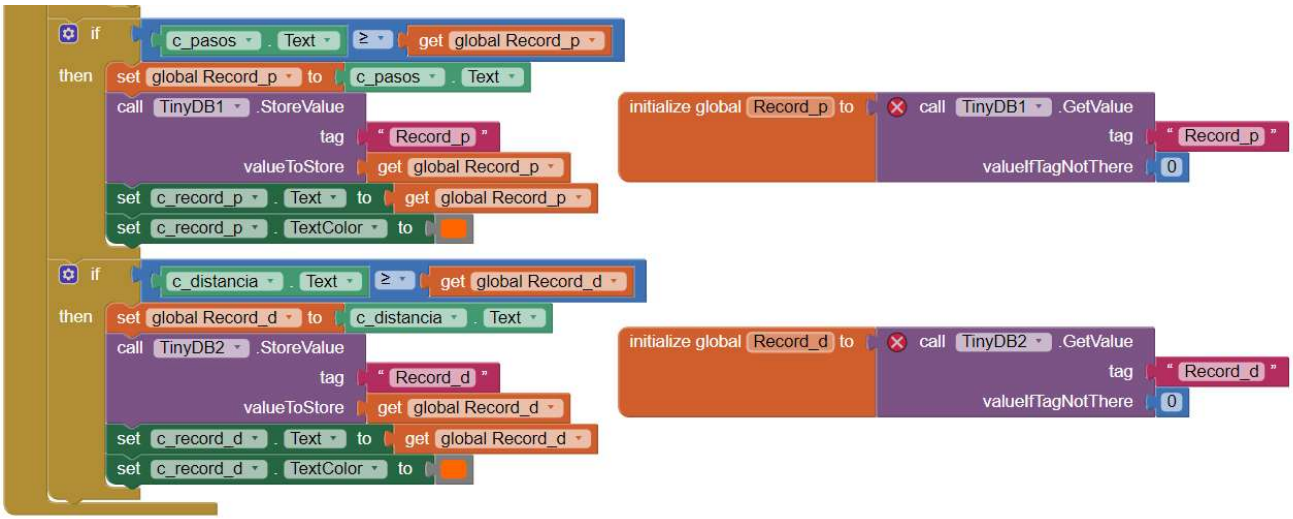


Ilustración 58: Programación de bloques de registro y salvado de los valores de record

En la segunda parte de las funciones llevadas a cabo cuando el reloj principal está en funcionamiento encontramos los bloques que registran y guardan los valores de record del número de pasos y la distancia recorrida.

Se tratan de dos bloques “if” de estructura idéntica los cuales se llevan a cabo cuando el valor de pasos o distancia es igual o mayor al valor de record anterior. Si se cumple esta condición el texto de record pasa a ser naranja mientras se esté superando este, además se guarda este valor a través de una variable que se asigna a un bloque de memoria “TinyDB” enlazado en los bloques que se encuentran en la parte derecha de la imagen anterior.

Tan solo resta comentar la parte del diseño de bloques a cerca del cálculo de la distancia recorrida haciendo uso del sensor de localización GPS del dispositivo móvil y del cálculo de una fórmula (ecuación del semiverseno o de Haversaine) utilizando los valores de latitud y longitud que recoge el sensor.

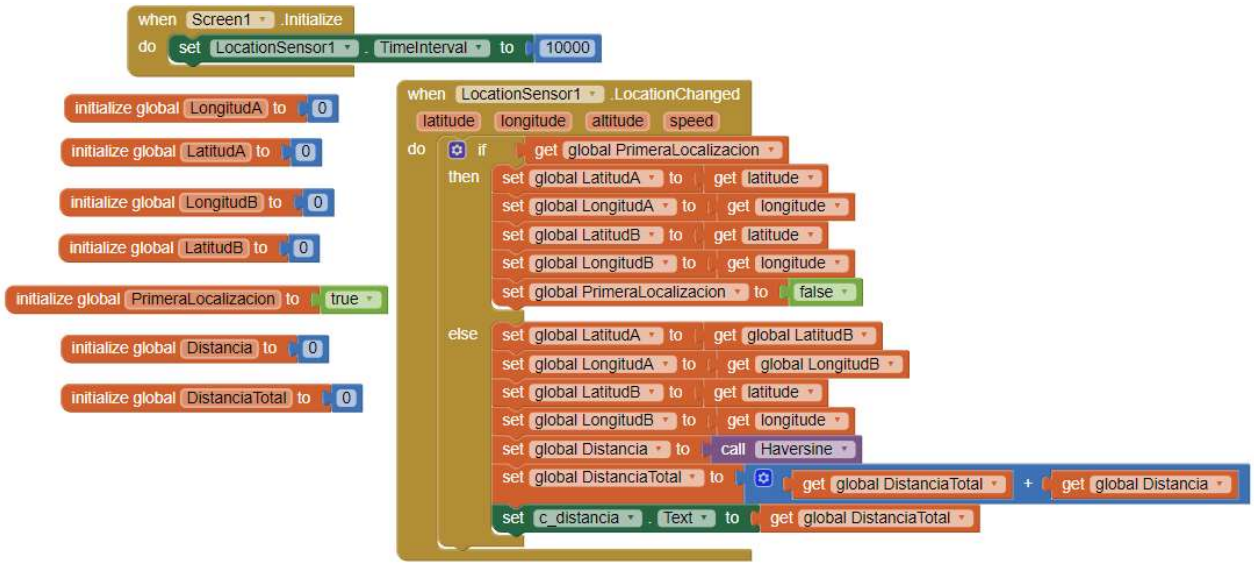


Ilustración 59: Programación de bloques para el cálculo de la distancia recorrida

El primer bloque se utiliza para establecer un intervalo de 10 segundos en las mediciones del sensor de localización, aunque en la práctica no se comporta con repeticiones constantes. En la parte izquierda de la imagen se han declarado las variables globales para la primera localización del sensor, la distancia de una medición, la distancia total, y los valores de longitud y latitud de un punto A y un punto B entre los cuales se hace el cálculo de distancia recorrida de una medición.

En el bloque de la derecha se lleva a cabo el proceso general para el cálculo de la distancia recorrida. Este se activa cuando el sensor de localización detecta un cambio, en el caso de que sea la primera localización toma medida de las longitudes y latitudes de A y B y marca que ya no se encuentra en la primera localización. En el caso de que ya no se encuentre en la primera localización pasa los valores anteriores del punto B al punto A y mide un nuevo punto B; a continuación calcula la distancia entre puntos ejecutando la ecuación de Harvesein y esa distancia resultante la suma a la distancia total calculada desde el punto inicial. Finalmente se escribe en el cuadro de texto de la interfaz el valor de la distancia total recorrida.

En último lugar cabe el cometario de la ecuación de Haversine. Se trata de una ecuación utilizada para el cálculo de distancias entre dos puntos de un globo sabiendo la longitud y su latitud de ambos puntos. La fórmula es la siguiente:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Ilustración 60: Fórmula de la ecuación de Haversine

Donde r es el radio de la Tierra, d la distancia recorrida, ϕ representa a las latitudes y λ a las longitudes.

El equivalente a esta ecuación en la programación de bloques es lo siguiente:

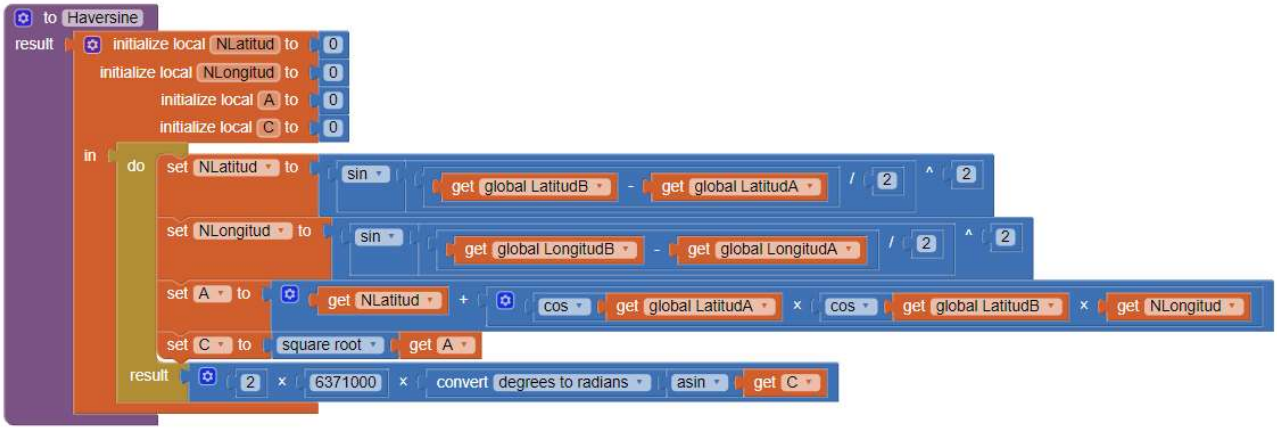


Ilustración 61: Programación de bloques de la ecuación de Haversine

Finalmente se muestra una imagen con toda la programación de bloques implementada en el diseño de la aplicación:

Ilustración 62: Vista completa de la programación de bloques de la aplicación WearPod

3.3.3 Alternativas

Cabe mencionar las alternativas del diseño de la aplicación. En este caso se trata del método de medición de la distancia recorrida, el cual en primer lugar se diseñó incluyendo en la interfaz un cuadro para escribir la longitud de la zancada del usuario y así multiplicar esta cifra por los pasos realizados para obtener la distancia recorrida. Sin embargo se trata de un método menos preciso que en el caso del uso del sensor de localización.

Ilustración 63: Interfaz con opción de insertar longitud de zancada

Ilustración 64: Alternativa interfaz aplicación final

4. *Presupuesto*

En la siguiente sección se van a presentar las tablas de presupuesto resultante del desarrollo del presente Trabajo de Fin de Grado. Estas tablas componen los gastos de hardware, es decir los materiales, y los gastos de personal para el diseño e implementación del dispositivo y la aplicación móvil.

COMPONENTES

Descripción	Coste	Cantidad	Coste total
Placa Arduino Nano (Atmega 328)	23,50 €	1	23,50 €
Módulo acelerómetro ADXL 345	2,29 €	1	2,29 €
Módulo HC-06 bluetooth 2.0	6,95 €	1	6,95 €
Placa breadboard 170 pin	1,60 €	1	1,60 €
Pila alcalina 9V	2,13 €	1	2,13 €
Conector clip pila 9V	0,80 €	1	0,80 €
Cables macho/macho para prototipado	0,03 €	8	0,24 €
Soldador de estaño eléctrico 30W	6,95 €	1	6,95 €
Tira de tela con velcro	1,25 €	1	1,25 €
TOTAL			**44,46 €**

MANO DE OBRA

Descripción	Coste	Cantidad	Coste total
Montaje dispositivo	24 €/h	1	24 €
Diseño de software	30 €/h	3	90 €
Desarrollo aplicación	28 €/h	8	224 €
TOTAL			**338 €**

PRESUPUESTO EJECUCIÓN MATERIAL

Descripción	Coste
Componentes	44,46 €
Mano de obra	338 €
TOTAL	**383,71 €**

PRESUPUESTO TOTAL

Descripción	Coste
Presupuesto ejecución material	383,71 €
Beneficio industrial (20%)	76,74 €
TOTAL SIN IVA	460,45 €
IVA (21%)	96,69 €
TOTAL	**557,14 €**

5. Conclusiones

Con la realización del presente Trabajo de Fin de Grado se ha desarrollado el diseño e implementación de un dispositivo wearable con función de podómetro con aplicación en el calzado del usuario además de una aplicación móvil para monitorizar el ejercicio realizado.

Se ha hecho uso de los conocimientos adquiridos durante los años componentes del Grado en Ingeniería Electrónica Industrial y Automática. Se han aplicado tanto en la selección de los componentes de hardware adecuados como en el desarrollo del proyecto a nivel de software, es decir, en la programación de un microcontrolador, en este caso Arduino.

Por otro lado se han establecido varios sistemas de comunicación de datos entre microcontrolador, los sensores utilizados y el dispositivo móvil.

Además se han adquirido nuevos conocimientos en el diseño de una aplicación móvil en el sistema operativo Android y su pertinente comunicación bluetooth con el dispositivo wearable.

Se ha comprobado el correcto funcionamiento de las diferentes partes y sus conexiones dando como resultado un sistema que cumple con todos los objetivos que se han planteado al comienzo del desarrollo del mismo.

Cabe mencionar la posibilidades en cuanto a mejores que presenta el proyecto, lo cual se desarrolla en el apartado siguiente de líneas futuras, no se trata de un diseño perfecto a pesar de cumplir con los objetivos.

En cuanto al coste de desarrollo teniendo en cuenta tanto el precio de los componentes utilizados tanto como de la mano de obra se calcula un coste de 383,71 €.

6. Líneas futuras

Como punto final en este proyecto cabe mencionar las posibilidades a futuro del dispositivo y la aplicación móvil desarrollados ya que los diseños desarrollados son bastante primitivos.

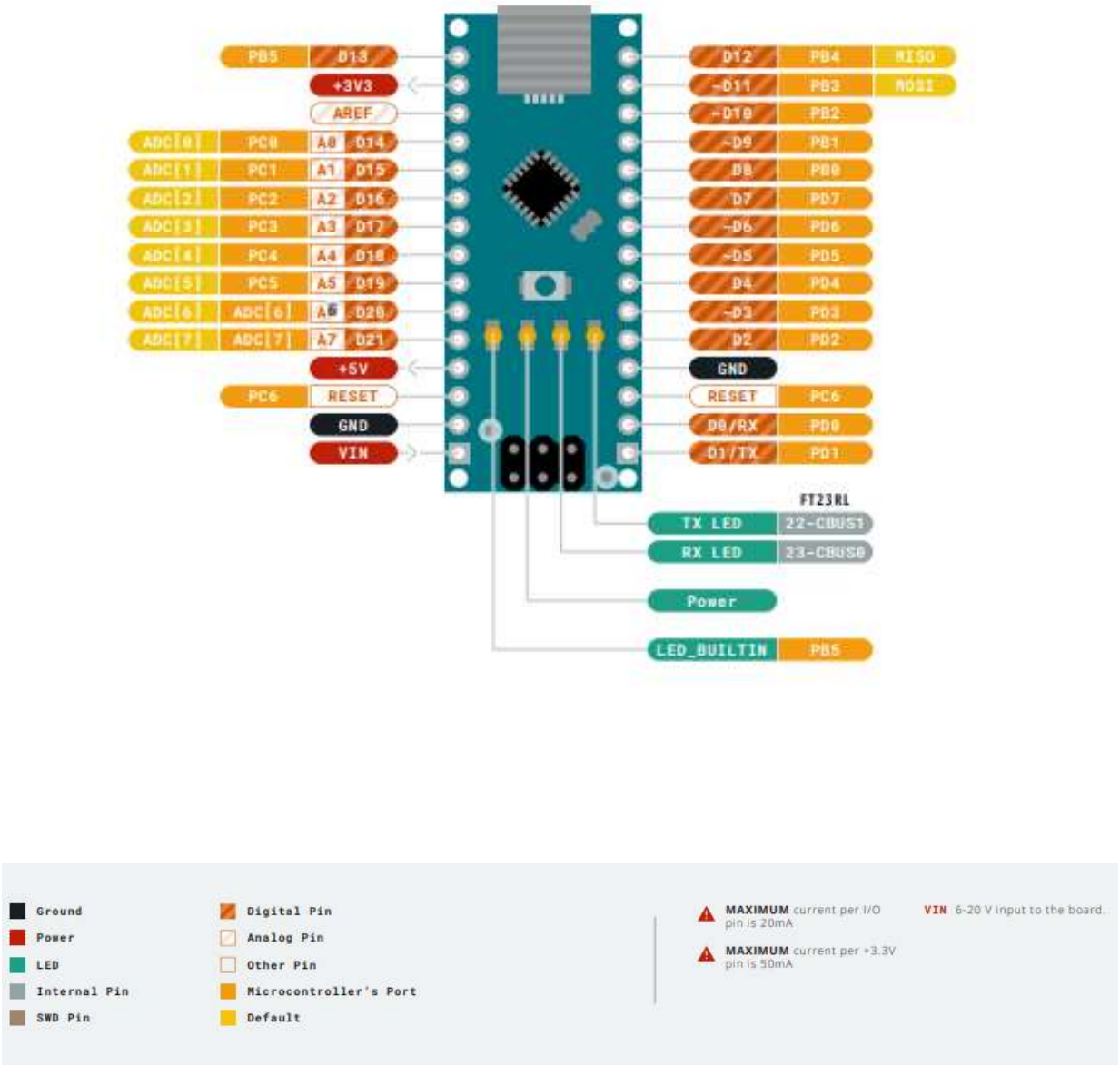
En cuanto al dispositivo es lógico buscar un rediseño de este con el objetivo de reducir su tamaño ya que se aplica a una zona de espacio bastante limitado. Como solución, se puede realizar por ejemplo el diseño de un circuito impreso PCB con todos los elementos necesarios, el cual reduciría considerablemente el tamaño, además, buscar un método de sujeción discreto y más sencillo de ajustar ayudaría a la comodidad de uso.

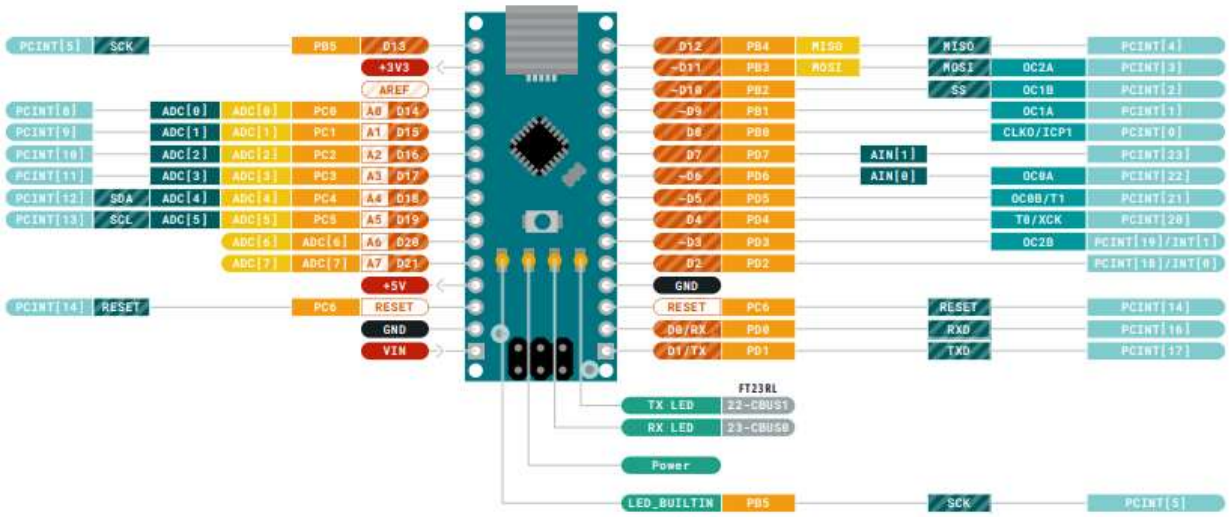
A parte de el tamaño del dispositivo cabe mencionar la incorporación en el circuito de un botón de apagado y encendido, además de reemplazar la pila alcalina por un batería recargable para así tener un uso más doméstico en cuanto a la gestión de la batería.

En lo referente a la aplicación móvil es común que a futuro se produzcan evoluciones en múltiples aspectos tanto como la velocidad de la conexión de la comunicación con el dispositivo como en la mejora de las funciones que ofrece, de la inclusión de nuevas así como de la actualización de esta y su rediseño para hacerla compatible con otros sistemas operativos como por ejemplo iOS.

7. Anexos

7.1 Configuración pines placa Arduino Nano:





Ground
Power
LED
Internal Pin
SWD Pin

Digital Pin
Analog Pin
Other Pin
Microcontroller's Port
Default

Analog
Communication
Timer
Interrupt
Sercom

MAXIMUM current per I/O pin is 20mA

MAXIMUM current per +3.3V pin is 50mA

VIN 6-20 V input to the board.

7.2 Configuración pines y registros acelerómetro ADXL345

PIN CONFIGURATION AND FUNCTION DESCRIPTIONS

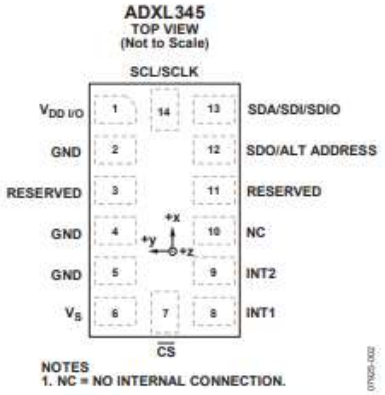


Figure 3. Pin Configuration (Top View)

Table 5. Pin Function Descriptions

Pin No.	Mnemonic	Description
1	$V_{DD\ I/O}$	Digital Interface Supply Voltage.
2	GND	This pin must be connected to ground.
3	RESERVED	Reserved. This pin must be connected to V_S or left open.
4	GND	This pin must be connected to ground.
5	GND	This pin must be connected to ground.
6	V_S	Supply Voltage.
7	$\overline{CS}$	Chip Select.
8	INT1	Interrupt 1 Output.
9	INT2	Interrupt 2 Output.
10	NC	Not Internally Connected.
11	RESERVED	Reserved. This pin must be connected to ground or left open.
12	SDO/ALT ADDRESS	Serial Data Output (SPI 4-Wire)/Alternate I^2C Address Select (I^2C).
13	SDA/SDI/SDIO	Serial Data (I^2C)/Serial Data Input (SPI 4-Wire)/Serial Data Input and Output (SPI 3-Wire).
14	SCL/SCLK	Serial Communications Clock. SCL is the clock for I^2C, and SCLK is the clock for SPI.

REGISTER MAP

Table 19.

Address					
Hex	Dec	Name	Type	Reset Value	Description
0x00	0	DEVID	R	11100101	Device ID
0x01 to 0x1C	1 to 28	Reserved			Reserved; do not access
0x1D	29	THRESH_TAP	R/$\overline{W}$	00000000	Tap threshold
0x1E	30	OFSX	R/$\overline{W}$	00000000	X-axis offset
0x1F	31	OFSY	R/$\overline{W}$	00000000	Y-axis offset
0x20	32	OFSZ	R/$\overline{W}$	00000000	Z-axis offset
0x21	33	DUR	R/$\overline{W}$	00000000	Tap duration
0x22	34	Latent	R/$\overline{W}$	00000000	Tap latency
0x23	35	Window	R/$\overline{W}$	00000000	Tap window
0x24	36	THRESH_ACT	R/$\overline{W}$	00000000	Activity threshold
0x25	37	THRESH_INACT	R/$\overline{W}$	00000000	Inactivity threshold
0x26	38	TIME_INACT	R/$\overline{W}$	00000000	Inactivity time
0x27	39	ACT_INACT_CTL	R/$\overline{W}$	00000000	Axis enable control for activity and inactivity detection
0x28	40	THRESH_FF	R/$\overline{W}$	00000000	Free-fall threshold
0x29	41	TIME_FF	R/$\overline{W}$	00000000	Free-fall time
0x2A	42	TAP_AXES	R/$\overline{W}$	00000000	Axis control for single tap/double tap
0x2B	43	ACT_TAP_STATUS	R	00000000	Source of single tap/double tap
0x2C	44	BW_RATE	R/$\overline{W}$	00001010	Data rate and power mode control
0x2D	45	POWER_CTL	R/$\overline{W}$	00000000	Power-saving features control
0x2E	46	INT_ENABLE	R/$\overline{W}$	00000000	Interrupt enable control
0x2F	47	INT_MAP	R/$\overline{W}$	00000000	Interrupt mapping control
0x30	48	INT_SOURCE	R	00000010	Source of interrupts
0x31	49	DATA_FORMAT	R/$\overline{W}$	00000000	Data format control
0x32	50	DATAX0	R	00000000	X-Axis Data 0
0x33	51	DATAX1	R	00000000	X-Axis Data 1
0x34	52	DATAY0	R	00000000	Y-Axis Data 0
0x35	53	DATAY1	R	00000000	Y-Axis Data 1
0x36	54	DATAZ0	R	00000000	Z-Axis Data 0
0x37	55	DATAZ1	R	00000000	Z-Axis Data 1
0x38	56	FIFO_CTL	R/$\overline{W}$	00000000	FIFO control
0x39	57	FIFO_STATUS	R	00000000	FIFO status

7.3 Código sketch del proyecto

```
#include <SoftwareSerial.h>
#include <Wire.h> // Librería Wire - utilizada para la comunicación I2C
int ADXL345 = 0x53; // Dirección I2C del sensor ADXL345
float X_out, Y_out, Z_out; // Salidas
int pasos=0;
float h, m, s, ms; // Variables reloj
unsigned long over,elapsed;

void setup() {
  Serial.begin(9600); // Iniciar comunicación serie para imprimir los resultados en el monitor serie
  Wire.begin(); // Iniciar la librería Wire
  // Establecer modo medicion al ADXL345
  Wire.beginTransmission(ADXL345); // Empezar comunicación con el dispositivo
  Wire.write(0x2D);// Accerder al registro POWER_CTL
  // Habilitar medición
  Wire.write(8);// (8dec -> 0000 1000 binario) Disponible bit D3 para medición
  Wire.endTransmission();
  delay(1000);
}

void loop() {

  elapsed = millis();
  h = int(elapsed / 3600000); // Cálculo horas reloj
  over = elapsed % 3600000;
  m = int(over / 60000); over = over % 60000; // Cálculo minutos reloj
  s = int(over / 1000); ms = over % 1000; // Cálculo segundos reloj


  // === Leer datos acelerómetro === //

  Wire.beginTransmission(ADXL345);
  Wire.write(0x32); // Comenzar con el registro 0x32 (ACCEL_XOUT_H)
  Wire.endTransmission(false);
```

```
  Wire.requestFrom(ADXL345, 6, true); // Leer 6 registros, cada eje se almacena en 2 registros


  X_out = ( Wire.read()| Wire.read() << 8);// Valor eje X
  X_out = X_out/256;// Para un rango de +-2g, necesitamos dividir los valores entre 256, según la datasheet
  Y_out = ( Wire.read()| Wire.read() << 8);// Valor eje Y
  Y_out = Y_out/256;
  Z_out = ( Wire.read()| Wire.read() << 8);// Valor eje Z
  Z_out = Z_out/256;


   if(Serial.read() == '1'){ // Mostrar pasos totales y tiempo transcurrido al usar el comando '1'
    Serial.print("Pasos: "); // Mostrar pasos totales
    Serial.println(pasos);
    Serial.print("Tiempo transcurrido: "); // Mostrar tiempo transcurrido
    Serial.print(h, 0);
    Serial.print("h ");
    Serial.print(m, 0);
    Serial.print("m ");
    Serial.print(s, 0);
    Serial.print("s ");
    Serial.print(ms, 0);
    Serial.println("ms");
    Serial.println();
    delay(200);


   }


  if(Z_out > 1.5  || Y_out > 0.8)
  {
    pasos = pasos +1;
    delay(200);
  Serial.println("1"); //Enviar '1' al monitor serie
  delay(100);


  }
}
```