ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

APLICACIÓN DE TECNOLOGÍAS WIRELESS A LOS DEPORTES DE FUERZA COMO MEJORA DE RENDIMIENTO

(Application of Wireless technologies to the strength sports as performance improvement)

Para acceder al Título de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Samuel Pérez Fernández

Sep - 2018

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Samuel Pérez Fernández
Director del TFG: Adolfo Cobo García
Título: " Aplicación de tecnologías wireless a lso deportes de fuerza como mejora del rendimiento "
Title: "Applications of Wireless Technologies to the strength sports as performance improvement"

Presentado a examen el día:

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Adolfo Cobo García
Secretario (Apellidos, Nombre): Francisco Javier Madruga Saavedra
Vocal (Apellidos, Nombre): Roberto Sanz Gil

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Trabajo Fin de Grado Nº
(a asignar por Secretaría)

Índice

Índice de figuras

Índice de tablas

Resumen

En este documento se presenta el diseño de un dispositivo formado por una placa Arduino y un sensor láser inalámbricos para el ámbito del deporte, concretamente para realizar entrenamientos de fuerza de forma eficiente midiendo la velocidad de ejecución en tiempo real.

El continuo crecimiento de los deportes de fuerza a lo largo de los últimos años ha llevado a la comunidad científica a ver qué relaciones hay de estos entrenamientos y cómo se puede entrenar de forma eficiente.

Dado que los dispositivos utilizados son excesivamente caros, esto llevó a la curiosidad de qué precisión podrían tener las prestaciones de una tecnología low- cost como son las que nos ofrecen la placa Arduino Wemos D1 mini pro y el sensor VL53L0X. Con los cuales mediremos la velocidad de ejecución en tiempo real y realizaremos un Access Point (AP) pa[r]a crear un servidor web en la misma placa.

Para poder discutir los resultados obtenidos mediante simulación, en este proyecto se ha llevado a cabo una comparativa entre el dispositivo Speed4Lifts, por cámara y con el propio proyecto.

Abstract

In this document the design of a device compound by an Arduino board and a wireless laser sensor is presented for the sport field, specifically to perform strength training efficiently measuring the execution speed in real time.

The continuous growth of the strength sports over the last few years has taken the scientific community to see what relationships there are in these trainings and how they can train efficiently.

Due to the devices used are excessive expensive, this led to the curiosity of how accurate could be a low- cost technology, such as offer the board Arduino Wemos D1 Mini Pro and the VL53L0X sensor. With which we will measure the execution speed in real time and perform an Access Point (AP) to create a web server in the same board.

In order to be able to discuss the results obtained by simulation, in this project a comparison has been made between the Speed4Lifts device, by camera a with the project itself.

Agradecimientos

Primero de todo quiero quisiera agradecer a Adolfo por su ayuda a lo largo de este proyecto, ya que gran parte de él se realizó en un Erasmus y la comunicación a veces dificultaba las cosas.

Agradecer también a Sergio y Vidal del club Fuerza Norte León por ayudarme con el testeo del proyecto con el Speed4Lifts y al centro deportivo Supera por dejarnos sus instalaciones.

Por supuesto agradecer a mi familia, en particular a mis padres y abuelos por su apoyo y sacrificio que han hecho para que pueda estudiar lo que me gusta, además de brindarme la oportunidad de poder realizar parte de ello en el extranjero.

They could not miss, all the friends I have made in Ireland, my lads, really thank you for participating in the best 4 months of my life. Even if I had a lot of work and stress, all the moments that I have shared with ye were worth it. I had a really good craic so… Sláinte!

Por último, mis agradecimientos a mis amigos y compañeros de clase que al final he pasado más tiempo con ellos que con mi propia familia. Gracias por toda la ayuda mutua y los buenos tiempos que estamos pasando, que no son pocos.

1.Introducción

En este primer capítulo se explicará la motivación de llevar a cabo este trabajo, una explicación de los conceptos del entrenamiento de fuerza, así como los objetivos que se quieren alcanzar. Finalmente, se detalla la estructura que sigue el documento, con el fin de adelantar y entender los diferentes elementos que de tratan en la memoria de este proyecto.

1.1 Planteamiento del problema

En estos últimos años, gracias a las redes sociales, a los pioneros de los deportes de fuerza (halterofilia y powerlifting) y a la revolución del Crossfit, el número de atletas de deportes de fuerza se ha visto en aumento. Ya de antemano, González Badillo [1] mostraba interés por la dosificación del entrenamiento en los años 80, pero fue hasta los años 90 donde ya desarrollaron con proyectos de la universidad un hardware y software propios para la medición de velocidad que se estaba llevando a cabo a través de una cámara.

El desarrollo de estos equipos como SmartCoach o T-Force, cuyo precio oscila entre 2000- 3000€, hizo que jóvenes talentos desarrollaran los suyos propios, comercializándose Speed4Lifts en 2017, el cual tiene un coeficiente de correlación (R^2) con T-Force de 0.9681 y con Gold Estándar 0.9655 [2]. Sin embargo, el precio de este ha subido desde los 250€ de su lanzamiento hasta los 366€ Es por esto por lo que se llegó a la curiosidad de cuán lejos podía llegar algo con menos prestaciones, mucho más barato, 100% inalámbrico y el conocimiento de un estudiante de ingeniería de telecomunicación aficionado a estos deportes.

En definitiva, este trabajo se propone un nuevo enfoque para medir las velocidades de ejecución en los levantamientos de potencia con una tecnología Wireless y low- cost.

1.2 Objetivos

En este proyecto se llevará a cabo la realización del hardware y del software de la placa que controlará tanto un sensor láser como un servidor web en un AP. Además, se realizará la comparativa de las medidas tomadas por el proyecto, con las tomadas tanto por Speed4Lifts y con lo calculado mediante la cámara de un Smartphone Samsung Galaxy S7 en resolución Full-HD a 60 FPS.

La placa a utilizar se programará en lenguaje Arduino, ya que es una Wemos D1 mini pro, la cual actuará como AP con IP 192.168.4.1 donde se ejecutará un pequeño servidor web HTML (HyperText Markup Language) donde se reflejarán los resultados, ya sean los medidos por el sensor VL53L0X o los que solicite el cliente a través de dicho servidor a través de un dispositivo con conexión Wi- Fi.

1.3 Introducción al entrenamiento de fuerza

Para poder entender mejor qué es lo que se está representando en este proyecto es necesario unos conocimientos y conceptos básicos acerca del entrenamiento de fuerza.

La forma de cuantificar la carga en uno de estos deportes se denomina RM (Repetición Máxima), por consiguiente, se deduce que es el peso máximo con el que hacer una repetición, 2RM es el peso máximo con el que hacer 2 repeticiones, no obstante, también se denota por porcentajes el equivalente de 1RM sería el 100%RM. Gracias a esto es más fácil para los entrenadores establecer los entrenamientos a sus atletas; por ejemplo, realizar 4 series de 3 repeticiones al 85%RM.

Sin embargo, esto va un poco más allá, ya que cada día nuestro RM varía por diversas razones (comida, descanso, activación, momento de la temporada, etc), por ello, decir que por un día haber tenido de 1RM 100kg no significa que siempre sea así. Fue entonces cuando González Badillo se dio cuenta que el %RM dependía de la VMP (Velocidad Media Propulsiva), es decir, la velocidad media en la fase concéntrica (cuando se hace el mayor esfuerzo). Además, el esfuerzo que representa cada porcentaje de 1RM es distinto según los ejercicios [1], sin embargo, a priori, un mismo atleta levantará su 1RM a la misma velocidad (aunque los atletas de élite pueden levantar un máximo a una velocidad menor de lo normal). Más allá de todo esto, si un atleta reduce su ROM (Range of Movement) cancelando así su punto de estancamiento, puede aumentar su RM.

1.4 Estructura de la memoria

A continuación, se detalla la estructura del documento, revisando brevemente el contenido de los capítulos.

- En el capítulo 2 se presentan los componentes utilizados durante el trabajo con sus principales características.

- En el capítulo 3 se muestran los conceptos previos necesarios para desarrollar y entender el trabajo. Se explican aspectos teóricos como aceleración, velocidad y posición de cinemática, ya que serán datos por representar en el trabajo. La regla de Simpson, como método de cálculo, el concepto de Wi- Fi, que será la tecnología de conectividad usada, el chip ESP8266, que brinda grandes aplicaciones. Además, se explicarán de forma sencilla las librerías utilizadas en el proyecto para facilitar su comprensión. Y por último se explicará la conexión del bus I2C que será el protocolo de comunicaciones entre el sensor y la placa.

- El capítulo 4 supone la parte central del trabajo. En primer lugar, se presentan los materiales que han sido necesarios. A continuación, se detalla cómo se ha realizado el software del trabajo y su desarrollo final.

- Por último, en el capítulo 5, se obtienen los resultados a partir del Speed4Lifts, por cámara, y por el proyecto. Posteriormente se detallan las conclusiones que se han alcanzado tras el análisis de los resultados obtenidos previamente y la explicación de las líneas futuras del uso de este tipo de tecnología.

2.Características de los componentes

Este capítulo recoge las características principales de los componentes que se han utilizado durante la realización de este proyecto. De esta forma el lector puede adquirir unos conocimientos básicos para tratar con el diferente hardware que se ha utilizado.

2.1 Acelerómetro ADXL335

Este sensor acelerómetro nos permite medir la aceleración en los tres ejes espaciales X, Y, Z. Es un sensor compacto y de tamaño muy reducido (20.3 mm x 15.7 mm). Para operar correctamente con este sensor debe estar conectado a una alimentación de 3.3 V y normalmente se encuentra en forma de módulo con los componentes necesarios para su conexión directa con placas Arduino.

El rango en el que es capaz de operar es de -3G a +3G [3] (1G= 9.8m/s^2). Por otra parte, podremos medir cada uno de los tres ejes conectando cada entrada analógica (una por eje) a la placa, además de las correspondientes conexiones a VCC y GND.

Con los valores de aceleración podemos medir la velocidad, detectar sacudidas, gestos o incluso analizar movimientos como andar o correr, tal y como hacen los podómetros de los smartphones de hoy en día. No sólo eso, sino que también es muy utilizado en videoconsolas como la Wii, que incluye acelerómetros y giroscopios.

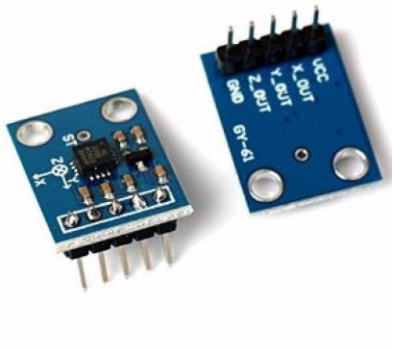

Figura 2.1: Sensor ADXL3355

2.2 Sensor Láser VL53L0X

El VL53L0X es un sensor de distancia infrarrojo láser de última generación, que junto con un hardware operador (preferiblemente Arduino) podemos medir distancias de 50 a 200 cm de forma precisa.

Este sensor opera incluso con gran luz ambiental infrarroja, e incorpora un sistema de compensación para la medición que lo permite hacer funcionar incluso detrás de un cristal protector.

Posee una precisión superior a sensor de ultrasonidos en infrarrojos y no se ve alterado por las condiciones del ambiente como ecos o reflectancia de los objetos. Sin embargo, el ángulo de medición es relativamente estrecho. Aunque lo más común es medir la distancia justo enfrente del sensor.

Pertenece a la familia de los sensores ToF (Time of Flight). Su funcionamiento se basa en un foto- emisor y un foto- detector, con el que se envía un pulso láser de luz infrarroja y mide el tiempo necesario en el haz en volver al sensor.

Este integrado incorpora el emisor láser de 940 nm VCSEL (Vertical Cavity Surface- Emitting Laser), un detector SPAD (Single Photon Avalanche Diodes) y una electrónica interna que realiza los cálculos necesarios.

El ángulo de medición o FOV (Field of View) es de 25º, lo cual se traduce en un área de medición de 0.44m de diámetro a una distancia de 1m. Por otra parte, el rango de medición depende de las condiciones del entorno (interior o exterior), de las características del objeto a reflejar y del modo de funcionamiento, el estándar es de 50 a 120cm y el modo ampliado llega hasta los 200cm.

Reflectancia Objetivo	Condiciones	Indoor	Outdoor
Objetivo Blanco	Típico	200cm	80cm
	Mínimo	120cm	60cm
Objetivo Gris	Típico	80cm	50cm
	Mínimo	70cm	40cm

Tabla 2.2.1: Rangos de mediciones

Debido a la precisión, que depende del entorno, objetivo y modo de funcionamiento, obtenemos las siguientes precisiones de referencia para los modos de funcionamiento.

	Indoor			Outdoor		
Reflectancia objetivo	Distancia	33ms	66ms	Distancia	33ms	66ms
Objetivo blanco	A 120cm	4%	3%	A 60cm	7%	6%
Objetivo gris	A 70cm	7%	6%	A 40cm	12%	9%

Tabla 2.2.2: Precisiones de referencia para los distintos modos de operación

El montaje de este sensor es sencillo, ya que la comunicación de datos se realizará a través de un bus I2C (el cual se explicará más adelante). Con lo cual el esquema de conexiones será VCC a la conexión de 3.3V de nuestra placa, GND a GND y SCL al pin D1 y SDA al pin D2 para nuestro caso.

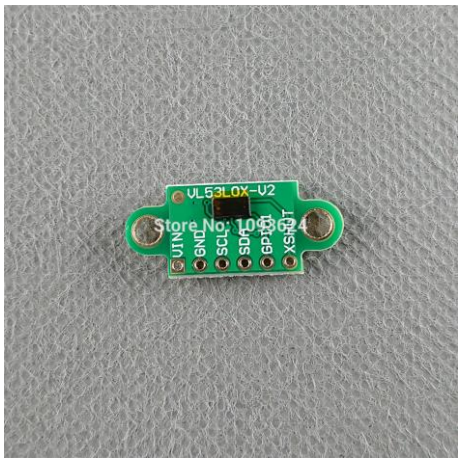

Figura 2.2: Sensor VL53L0X

2.3 Placa Wemos D1 Mini Pro

La placa Wemos D1 Mini Pro es la versión compacta de la familia de placas WeMos basadas en el chip Wi- Fi ESP8266. Incorpora un procesador central de 32 bits con un completo stack IP + Wi- Fi, pero además se monta una placa con pines al igual que en una protoboard.

Podemos decir que esta placa está gobernada por el chip ESP8266EX, que se encarga de las tareas de procesamiento y del control de Wi- Fi todo en un único chip, lo cual dice mucho de su potencia para su pequeño tamaño.

Esta placa se programa directamente desde el plugin IDE de Arduino, con la comodidad de poder incluirlo directamente en la protoboard o en sus placas de prototipado.

Sus características principales son las siguientes:

- 11 pines digitales entrada/ salida (I/O)
- 1 entrada analógica (max 3.3 V input)
- Interrupciones/ PWM/ I2C/ One- Wire
- 16Mbytes Flash
- Conector de antena externa
- Antena de cerámica
- Chip Cp2104 para la conexión USB- TO- UART (Universal Asynchronous Receiver-Transmitter).
- Longitud 34.2mm
- Ancho 25.6mm
- Peso 2.5g

Además, nos incluye un conjunto de tres tipos de conectores (los cuales requieren soldadura) que permiten varias configuraciones de conexión y que los "shields" sean fácilmente añadidos o removidos. Esta placa está diseñada para permitir que los "shields" compatibles con WeMos mini se conecten a la placa de una manera similar a la plataforma de desarrollo Arduino.

Figura 2.3.1: Placa WeMos D1 Mini Pro

Debido a los requerimientos necesitados, se vio preciso incluir un dispositivo de alimentación externo, ya que no tener alimentación propia, la placa debería estar conectada mediante conexión USB a un ordenador, lo cual podría resultar incómodo a la hora de realizar los levantamientos.

Gracias a los innumerables "shields" que mencionábamos antes, se pudo utilizar uno que se adecuaba a nuestras necesidades, en concreto el Battery Shield V1.2.0 para WeMos D1 Mini, el cual permite conectar una batería de litio y cargarla. Las principales características de este shield son las siguientes:

- Voltaje de carga: Max 10V, recomendado 5V.
- Amperaje máximo de carga 1A.
- Voltaje de la batería de litio: 3.3- 4.2V.
- Alimentación: 5V, max 1A.

Sólo posee los pines de 5V y GND, pero sin embargo posee distintos puertos:

- PH2-2.0MM: Conectada a la batería de litio de 3.3-4.2V.
- Micro USB: Puerto de carga.
- LED verde: Indica carga completa.
- LED rojo: Indica batería en carga.
- J1: Establece la carga máxima en 0.5 o 1A.
- J2: Conecta la batería con A0.

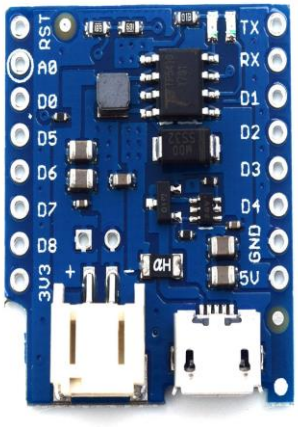

Figura 2.3.1: Battery Shield V1.2.0 Mini

3.Conceptos teóricos

Este capítulo recoge una introducción de cada uno de los conceptos en los que se apoya el trabajo. De este modo el lector obtendrá unos conocimientos básicos para comprender los diferentes conceptos tratados a lo largo del proyecto.

3.1 Aceleración, velocidad y posición

La aceleración es una magnitud derivada vectorial que nos indica la variación de velocidad por unidad de tiempo, la cual es expresada en m/s^2. Normalmente se representa por **a**, no obstante, para este proyecto se utilizará la magnitud **G**, ya que nos desplazamos principalmente en el plano Y.

Una partícula en movimiento rectilíneo solo puede variar su velocidad bajo la acción de una aceleración en la misma dirección de su velocidad (en el mismo sentido acelera, en el contrario desacelera). De este modo definimos en mecánica clásica la aceleración como la variación de la velocidad respecto al tiempo:

$$\mathbf{a} = \frac{d\,\mathbf{V}}{d\,t}$$

La llamada aceleración de la gravedad de la Tierra es la aceleración que produce la fuerza gravitatoria terrestre gracias a su gran masa, su valor en la superficie del planeta es, aproximadamente, 9,8 m/s^2. Con esto podemos decir que, si un objeto cae, omitiendo las fuerzas de rozamiento posibles, su velocidad aumentaría a razón de 9,8 m/s por cada segundo.

De la misma forma, definimos la velocidad como una magnitud física vectorial que relaciona el cambio de posición con el tiempo. Se representa por **v** y sus unidades son m/s.

La velocidad media, en nuestro caso VMP, se define como el cambio de posición durante un intervalo de tiempo. Se calcula dividiendo el vector desplazamiento entre el escalar tiempo:

$$\bar{\mathbf{v}} = \frac{\Delta \mathbf{r}}{\Delta t}$$

Por otra parte, nos interesa la velocidad instantánea, que es un vector tangente a la trayectoria, que corresponde con la derivada del vector posición respecto al tiempo. Esto nos permitirá conocer la velocidad de un móvil que se desplaza sobre una trayectoria cuando el intervalo de tiempo es infinitamente pequeño, siendo entonces el espacio recorrido muy pequeño, representando un punto de la trayectoria:

$$\mathbf{v} = \lim_{\Delta t \to 0} \frac{\Delta \mathbf{r}}{\Delta t} = \frac{d\mathbf{r}}{dt}$$

La posición de una partícula indica su localización en el espacio o en el espacio-tiempo. Se representa mediante el sistema de coordenadas X, Y, Z. Además, llamaremos distancia recorrida o ROM, a la distancia recorrida durante el intervalo de tiempo en que calculamos la velocidad.

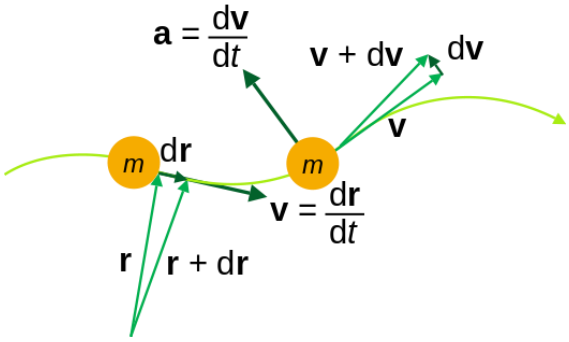

Figura 3.1: Magnitudes de interés en la cinemática

3.2 Regla de Simpson

En análisis numérico, la regla de Simpson es un método de integración numérica, al igual que las sumas de Riemann, o la regla del trapecio, que se utiliza para obtener la aproximación de la integral:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]$$

La regla de Simpson aproxima por subintervalos de f mediante polinomios de segundo grado, tomándole como polinomio interpolador. Este polinomio aproxima a la función integrando.

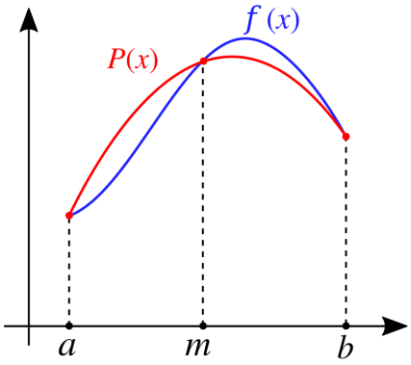

Figura 3.2: La función f(x) es aproximada por la función cuadrática P(x)

Hay distintas Reglas de Simpson distintas para calcular la aproximación de la integral, en nuestro caso utilizamos la Regla de Simpson 3/8 compuesta. En ella utilizamos polinomios de Lagrange de tercer orden, dividiendo el intervalo de integración en más subintervalos n. Para ello tenemos en consideración el paso $h = \frac{(b-a)}{n}$ con la condición de que n sea múltiplo de 3 y que en cada sumatorio se tomen los valores de i= i+3, obteniendo así la siguiente expresión para la aproximación [4]:

$$I \approx \frac{3h}{8}\left[f(x_0) + 3\sum_{i=1,4,7\ldots}^{n-2} f(x_i) + 3\sum_{i=2,5,8\ldots}^{n-1} f(x_i) + 2\sum_{i=3,6,9\ldots}^{n-3} f(x_i) + f(x_n)\right]$$

3.3 Redes Wi- Fi

En 1999 existía un estándar, sin embargo, existían productos de diferentes fabricantes que no trabajaban correctamente juntos. Por esto surge la necesidad de establecer una tecnología que interconecte inalámbricamente distintos dispositivos.

Fue entonces cuando se crea la WECA (Wireless Ethernet Compability Alliance), formada por 3COM, Aironet (ahora Cisco) Lucent Technologies (ahora Agere), Intersil, Nokia y Symbol Technologies, actualmente llamada la Alianza Wi- Fi, con el objetivo de designar una marca que permitiese fomentar más fácilmente la tecnología inalámbrica y asegurar la compatibilidad de equipos, siendo así el estándar global en todos los segmentos del mercado. [5]

De esta forma, en abril de 2000 WECA certifica la interoperabilidad de equipos según la norma IEEE 802.11b, bajo la marca Wi-Fi. Esto quiere decir que el usuario tiene la garantía de que todos los equipos que tengan el sello Wi-Fi pueden trabajar juntos sin problemas, independientemente del fabricante de cada uno de ellos.

En el año 2002, la asociación WECA estaba formada ya por casi 150 miembros en su totalidad. La familia de estándares 802.11 ha ido naturalmente evolucionando desde su creación, mejorando el rango y velocidad de la transferencia de información, su seguridad, entre otras cosas.

La norma IEEE 802.11 fue diseñada para sustituir el equivalente a las capas físicas y MAC de la norma 802.3 (Ethernet). Esto quiere decir que en lo único que se diferencia una red wifi de una red Ethernet es en cómo se transmiten las tramas o paquetes de datos; el resto es idéntico. Por tanto, una red local inalámbrica 802.11 es completamente compatible con todos los servicios de las LAN (Local Area Network) de cable Ethernet.

Existe un abanico de diversos tipos de wifi, basado en el estándar 802.11:

- Estándares que operan en la banda de 2.4 GHz: IEEE 802.11 b/g/n con velocidades de 11, 54 y 300 Mbit/s, respectivamente.

- Estándares que operan en la banda de 5 GHz: IEEE 802.11 a/n/ac con tasas binarias de 54, 600, 3200 Mbps, dado que operan en una frecuencia mayor y poseen tasas binarias superiores, tienen menos alcance que los estándares que trabajan a 2,4 GHz.
- Estándares que operan en la banda de 60 GHz: IEEE 802.11 ad cuya tasa asciende los 7 Gbps, aunque por el mismo motivo que los de la banda de 5 GHz, si alcance es menor a 5m.

Aunque esta tecnología es muy versátil, y está prácticamente en todos los dispositivos, un gran porcentaje de estas redes se instalas sin considerar el aspecto de la seguridad de la red, convirtiéndose así en redes abiertas a las cuales puede acceder una tercera persona. Muchas de las configuraciones de los dispositivos wifi (routers) son muy inseguras, dado que a partir de la MAC se puede conocer la contraseña de acceso de éste, por lo que acceder y manipular la información no resulta de gran complejidad. Una forma de hacer la red un poco más segura es tomando alguna de las siguientes precauciones:

- Cambios frecuentes de contraseña, utilizando números, mayúsculas y minúsculas.
- Modificar el SSID (Service Set Identifier) preterminado.
- Desactivar la difusión de SSID y DHCP (Dynamic Host Configuration Protocol).
- Configurar los dispositivos conectados según la dirección MAC a través de la configuración del router.
- Utilizar un cifrado, por ejemplo, el WPA2.

Podemos clasificar los dispositivos wifi en dos grupos: los dispositivos de distribución, entre los que se incluyen APs, routers y repetidores, y los dispositivos terminales, que suelen ser tarjetas receptoras para conectar a un ordenador, ya sean internas con un PCI o un USB.

En este proyecto nos centraremos exclusivamente en los AP, que son dispositivos que generan una red wifi a la que se pueden conectar otros dispositivos. Un AP permite conectar dispositivos de forma inalámbrica a una red existente. Pueden agregarse más puntos de acceso a una red para generar redes de cobertura más amplia, o conectar antenas más grandes que amplifiquen la señal.

3.4 Chip ESP8266

Es un chip low- cost que incluye conectividad Wi-Fi y soporta la pila de protocolos TCP/IP, por lo que nos permite conectar nuestro Arduino a cualquier red, en

particular soporta 802.11 b/g/n. Este pequeño chip posee un microcontrolador y un procesador Tensilica Xtensa LX106 a 80 MHz (160 MHz haciendo overclock), lo cual nos deja una gran capacidad de cómputo para lo pequeño que es.

Este sensor fue un gran éxito en la comunidad Arduino ya que antes de que apareciese este chip, para que una placa se conectase a una red Wi- Fi, se debía integrar un shield que oscilaba entre 3 y 4 veces el precio de la placa Arduino. Otra de sus grandes ventajas es su consumo, posee unos consumos muy bajos en comparación con el resto del mercado.

Estas son ciertas de sus características principales que han sido notorias para la realización del trabajo:

- CPU 32 bit a 80 MHz.
- RAM de instrucción de 64KB, RAM de datos de 96 KB.
- Capacidad de memoria externa flash de 512 KB a 16 MB (la cual posee la placa utilizada).
- Soporte de IEEE 802.11 b/g/n con soporte de autenticación WEP y WPA/ WPA2.
- 16 pines GPIO.
- I2C.
- Pines dedicados a UART, para la transmisión se habilita a través del pin GPIO2.
- 1 conversor ADC de 10 bit. [6]

3.5 Librería ESP8266WiFi

La librería ESP82266Wifi es una librería de código abierto en lenguaje C++ que está basada en la librería Wifi.h para dar soporte a los shields de WiFi de Arduino.

Gracias a esta pequeña librería nos será mucho más fácil iniciar nuestro servidor (siempre y cuando entendamos la librería), ya que gracias a ella podemos iniciar y modificar parámetros como:

- Dirección IP.
- Modo de funcionamiento STA/ AP/ Scan/ Generic.
- Crear un cliente/ servidor.
- SSID/ Password/ RSSI/ BSSID/ canal. [7]

3.6 Librería WiFiClient

La librería WiFiClient va de la mano junto a la anterior, igualmente es de código abierto y escrita en C++. La particularidad de esta librería es que nos permitirá saber cuál es nuestro socket, es decir, saber qué dirección IP y qué puerto estamos utilizando (por defecto se usa el puerto 80 el cual escucha las peticiones HTTP). [8]

3.7 Librería ESP8266WebServer

La librería ESP8266WebServer se utiliza para simplificar la creación de un servidor web y va de la mano junto con la librería WiFiClient. Nos permite crear un objeto servidor WEB que escuchará en el puerto que le digamos (en nuestro caso puerto 80 ya que es una petición HTTP) gracias a la instrucción:

ESP8266WebServer server(80);

Además, gracias a server.on indicamos una función como respuesta del servidor cuando el navegador solicita cierta URL (Uniform Resource Locator). El ESP8266 nos permite crear diferentes subdirectorios de la URL y configurar diferentes funciones para estos como, por ejemplo:

server.on("/estimasquat",estimasquat);

server.on("/estimabanca",estimabanca);

server.on("/pushspeed",pushspeed);

server.on("/pullspeed",pullspeed);

Cuando el navegador se conecte a la placa se ejecutará ese código que hemos puesto dentro del *server.on* y que en el caso de que la URL apunte a la raiz (/) es una llamada a la función *server.send*, que lo que hace es responder al navegador. Para ello, hay que darle tres argumentos:

- Código de respuesta: es un código que se utiliza en el protocolo HTTP para indicar diferentes eventos en las conexiones. En nuestro caso daremos el código 200, es el que se devuelve cuando sí que se encuentra el contenido, a diferencia de cuando encontramos un enlace roto que nos aparecería el código 404.
- MIME-type: es un código que indica el tipo de contenido que se devuelve, puede ser texto, imagen…
- Respuesta: El contenido de la respuesta en sí, ya puede ser texto, una función, o redirigir a otro subdirectorio.

Para iniciar nuestro servidor web utilizaremos la función de esta librería *server.begin();* y en la otra parte obligatoria de todo Arduino (loop()) escuchamos las conexiones entrantes *server.handleClient();* Otro de los aspectos más importantes es que en las funciones nos permite realizar GET y POST, para obtener datos o subirlos a nuestro servidor. La diferencia entre los métodos GET y POST radica en la forma de enviar los datos a la página cuando se pulsa el botón “Enviar”. Mientras que el método GET envía los datos usando la URL, el método POST los envía de forma que no podemos verlos (en un segundo plano u "ocultos" al usuario).

Por último otra de las funciones que nos ofrece esta librería de gran utilidad es la función *server.arg(“”);* la cual nos permite obtener el valor de un *name* del código HTML del directorio del servidor. [9]

3.8 Librería Wire

Esta librería se utiliza para comunicar la placa arduino con dispositivos que trabajan mediante el protocolo I2C/TWI. Este sistema de comunicación utiliza dos líneas de transmisión: SDA (datos serie) y SCL (reloj serie) conectadas a dos resistencias tipo pull-up a 5 voltios. En la placa Wemos D1 mini pro los pines SDA/ SCL corresponden con los pines D2/ D1 respectivamente.

La librería "Wire" utiliza 7 bits para identificar la dirección del dispositivo y el octavo bit indica si se lee o se escribe en él. Si el periférico que queremos controlar utiliza 8 bits de dirección, debemos desechar el bit de menos peso (por ejemplo, desplazando el valor 1 bit a la derecha), trabajando entonces con una dirección entre 0 y 127.

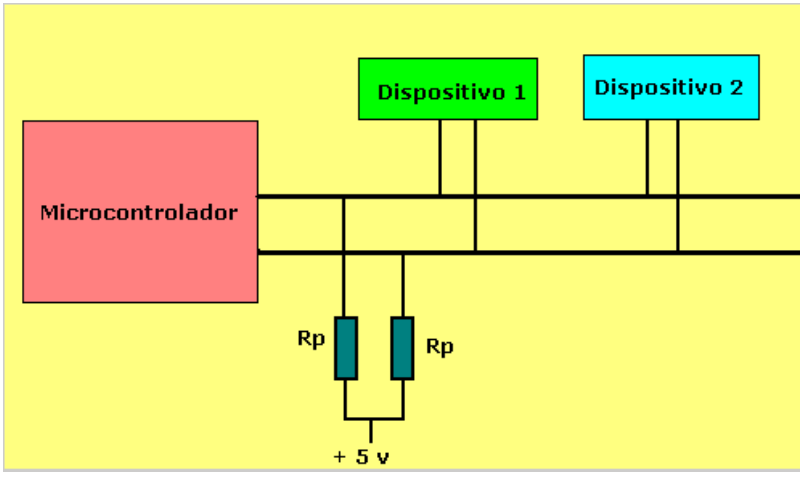

Figura 3.8: Esquema de conexión del protocolo I2C

Con la instrucción *Wire.begin()* inicializamos la librería y une el bus I2C como maestro o esclavo, normalmente sólo se llama una sola vez. [10] [11]

3.9 Librería VL53L0X

Como ya comentamos con anterioridad, para este proyecto hemos utilizado el sensor VL53L0X para medir la distancia en milímetros hasta el objeto más cercano a través del puerto I2C. Para utilizarlo Adafruit ha desarrollado esta librería la cual nos permite un usó fácil del sensor.

Esta librería tiene dos posibles modos, el modo de realizar un solo disparo y el de estar continuamente midiendo la distancia, lo cual se amolda perfectamente a nuestras necesidades. La función que nos dará el resultado será *sensor.readRangeContinuousMillimeters()* y simplemente asignándoselo a una variable ya tendremos esos mm de distancia. Tal y como se dijo el chip emite un fotón que se refleja en la superficie y llega al fotodetector la distancia será igual al tiempo de vuelo del fotón entre 2 multiplicado por la velocidad de la luz tal y como se indica en la figura. [12]

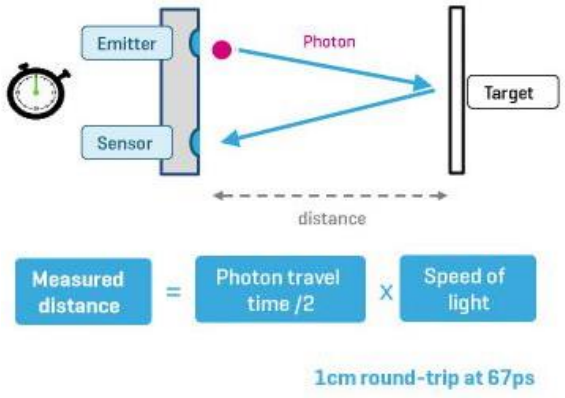

Figura 3.9: Esquema de cómo mide la distancia el sensor

3.10 Bus I2C

En cualquier sistema de microcontroladores de Arduino tenemos dos principales tipos de comunicación serie:

- UART, uno de los protocolos más utilizados y hardware que disponen la mayoría de los microcontroladores. Utiliza una línea de datos para transmitir y otra para recibir. Por otra parte, la forma de transmitir los datos es usando flags, de manera que manda un bit de inicio a nivel bajo, a continuación 8 bits de datos y un bit final a nivel alto. Podemos diferenciar UART de SPI (Serial Peripheral Interface) e I2C ya que es asíncrono a diferencia de los otros dos y la velocidad está limitada a 2Mbps.
- SPI, en el cual tenemos una comunicación maestro- esclavo, el maestro se encarga de enviar la señal de reloj, y tras cada pulso de reloj envía un bit al esclavo y recibe un bit de este.

El sistema de transmisión de datos entre el sensor y la placa usa el protocolo I2C, un bus de comunicaciones en serie. Este protocolo utiliza 2 cables, uno para el reloj (SCL) y otro para los datos (SDA). Esto conlleva que el maestro y el esclavo envían datos por el mismo cable, controlado por el maestro que crea la señal de reloj.

El nombre de este tipo de bus viene por *Inter-Integrated Circuit*, cuyo diseñador es Philips y sus primeras versiones datan en el año 1992. Posee una velocidad de 100 kbit/s en modo estándar, aunque también permite velocidades superiores de 3.4Mbit/s. Este tipo de conexiones es muy utilizado para comunicar microcontroladores y periféricos en sistemas integrados, sobre todo en la industria. Generalizando aún más, permite comunicar circuitos integrados entre sí que se encuentran dentro de un mismo circuito impreso, por lo que lo podemos encontrar en cualquier electrodoméstico de hoy en día, mandos, smartphones, smartwatches, televisores…

Las dos primeras líneas son drenador abierto, por lo que necesitan resistencias de pull-up. Dos o más señales a través del mismo cable pueden causar conflicto, y ocurrirían problemas si un dispositivo envía un 1 lógico al mismo tiempo que otro envía un 0. Por tanto, el bus es "cableado" con dos resistencias para poner el bus a nivel alto, y los dispositivos envían niveles bajos. Si quieren enviar un nivel alto simplemente lo comunican al bus.

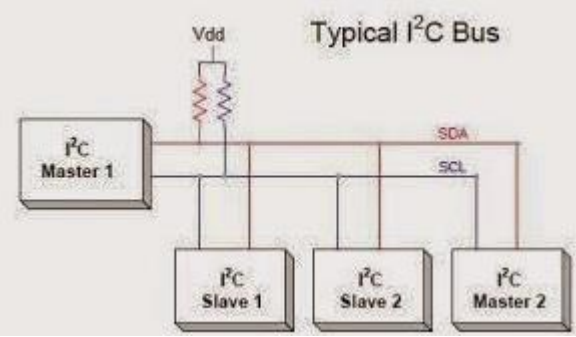

Figura 3.10.1: Esquema Típico de un Bus I2C

Los dispositivos conectados a este tipo de buses tienen una dirección única para cada uno de 7 bits, es decir, que en teoría podemos conectar 2^7= 128 dispositivos. Obligatoriamente uno de ellos debe ser maestro para controlar el reloj e iniciar la transferencia de datos, aunque el rol de maestro puede cambiar de uno a otro, si un dispositivo tiene la capacidad de ser maestro, este rol puede ir pasandose. Esta caracteristica nos da pie a denominar a este bus multimaestro.

El poceso de comunicación comienza con el maestro, enviando un patrón llamado "start condition", el cual alerta a los dispositivos esclavos para la espera de transacción. Posteriormente el maestro envía un byte con los siete bits (A7-A1) para dirigirse al dispositivo con el que quiere hablar, y un octavo bit (A0) que corresponde con la operación a realizar (L/E), lectura=1 (recibir del esclavo) y escritura=0 (enviar al esclavo). A continuación la dirección enviada es comparada por cada uno de los dispotivos esclavos del bus con su propia dirección, cuando una dirección de los esclavos coincide con la enviada, el esclavo se considera direccionado como esclavo-transmisor o esclavo-receptor, dependiendo del bit L/E. Cada byte leído/ escrito por el maestro debe

ser reconocido por un bit de ACK por el dispositivo maestro/esclavo. Para finalizar la comunicación, el maestro transmite una “stop condition” para dejar libre el bus.

Figura 3.10.2: Proceso de comunicación en I2C

Cuando los datos son enviados a través del SDA, el SCL envía los pulsos de reloj para mantener el maestro y el esclavo sincronizados. Los datos son enviados como un bit en cada pulso de reloj, por lo que la transferencia de datos es un octavo de la frecuencia de reloj. Originalmente la frecuencia de reloj se puso a 100KHz y la mayoría de microcontroladores soportan esta velocidad. En posteriores actualizaciones, se introdujo una fast speed de 400 KHz y una high speed de 1.7 a 3.4 MHz. Arduino puede soportar la velocidad estándar y fast speed, BeagleBoard tiene tres buses I2C cada uno a una velocidad distinta y tanto BeagleBoard como Raspberry Pi soportan velocidad estándar y fast speed. Fast speed corresponde a una velocidad de transferencia de 50Kbytes/sec lo que puede ser una velocidad muy baja para algunas aplicaciones de control. Podemos encontrar este bus por otras nomenclaturas, como IIC o I^2C, y también como TWI (Two Wire Interface), aunque todas se refieren a lo mismo [13]. Como ya hemos indicado en anteriores puntos, nuestro Arduino soporta de fábrica I2C con una librería estándar que utiliza dos pines analógicos para las funciones SDA y SCL, la librería I2C en Arduino se llama Wire, y gestiona el protocolo de comunicaciones al completo. [14]

4. Implementación Práctica

En este capítulo se expondrán las pautas que han sido seguidas para el despliegue de la experimentación. Se relatan las herramientas que han sido necesarias para la configuración de la placa y de los sensores: desde la programación llevada a cabo, que no deja de ser el núcleo de este proyecto (con pseudocódigo para que sea de mayor sencillez para el lector), hasta el desarrollo hardware efectuado para su correcto funcionamiento,

4.1 Hardware

Para que el correcto funcionamiento del proyecto se han necesitado distintos materiales, ciertos de ellos ya han sido nombrados:

- Placa Arduino Wemos D1 Mini Pro
- Battery Shield para Arduino Mini V.1.2.0
- Sensor acelerómetro ADXL335 (que finalmente no fue implementado).
- Sensor de vuelo VL53L0.
- Cables.
- 3 fundas termocontractoras para cables.
- 2 expansiones de pines.
- Estaño.
- Soldador.
- Desoldador de estaño de vacío.
- Pistola de silicona.
- Plástico.
- Percha.
- Batería recargable de 3.6V y 300mAh de Ni-Cd.

En primer lugar, se han soldado las dos expansiones para los pines a la placa Wemos D1 Mini Pro para posteriormente soldar el shield de la batería. A continuación, se ha situado el sensor VL53L0 (previamente soldado el patillaje) con silicona en la parte de debajo de la placa a la altura de la antena de cerámica para que no interfiera con ningún componente de la propia placa. A continuación, se ha conectado el VIN, GND, SCL y SDA a los correspondientes pines de la placa con unos cables a los que se les ha añadido una funda termocontractora para proteger las soldaduras. Por último, se ha añadido silicona a las expansiones de los pines para incluir una pieza de plástico que sirva de sujeción a la percha con silicona con el fin de poder anclar la placa a una barra olímpica para poder medir la velocidad de ejecución, que al fin y al cabo es el propósito de este proyecto.

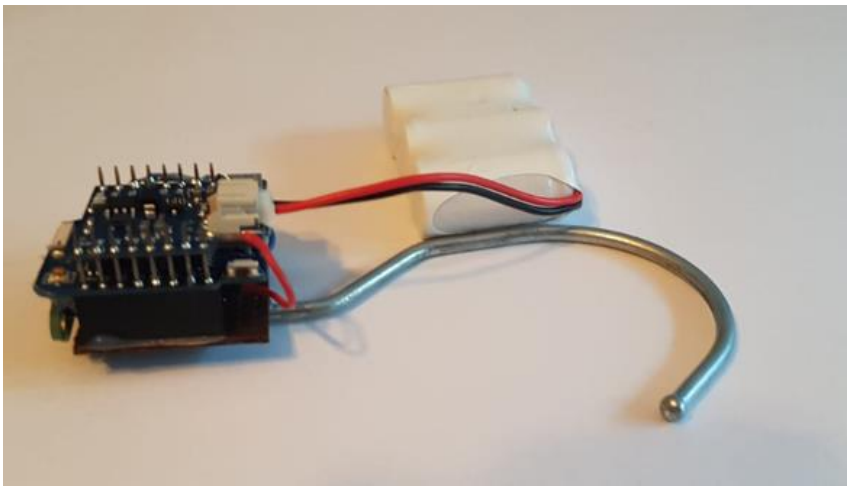

Figura 4.1: Desarrollo final del proyecto

4.2 Software

Para la implementación del software vamos a distribuir los siguientes subapartados en los siguientes:

- Software del acelerómetro ADXL335.
- Software del sensor ToF VL53L0.
- Software del servidor.

4.2.1 Software del acelerómetro ADXL335

Tal y como se nombró con anterioridad, este sensor nos mide la aceleración en los tres ejes XYZ, sin embargo, esta placa sólo dispone de un pin analógico, con lo cual no podemos medir el módulo del movimiento, sino sólo calcular lo que ocurre en el eje Y que será el más relevante para la aplicación que queremos realizar. Ya que finalmente esto no se llegó a implementar debido al ruido y a la falta de otros medios, se explicará de forma sencilla con el propósito de que el lector tenga unas nociones de lo que se realizó con este código. Como aclaración inicial, recordar que G es 9.8 m/s^2 aproximadamente, por lo cual si la medida no es igual a G significa que hay movimiento, una vez aclarado esto, proseguimos:

If (medida > G+ margen o medida < G-margen)
Empezamos a contar tiempo.
If(es la primera medida o no es la primera medida pero cumple los requisitos para realizarse la regla de Simpson, es decir, ser una muestra múltiplo de 3 y mayor que 6)
Restamos G a la medida
Realizamos un LPF (Low Pass Filter) que tome el promedio de 5 muestras para eliminar ruido.
If(es la quinta muestra)
Almacenamos valor.

If(La muestra es múltiplo de 3 y el valor es menor que 0.03) //Tomamos que un valor de referencia de 0.03 indica que se ha vuelto a parar.
Damos pie a que pueda salir del ciclo
Finalizamos el cronómetro y calculamos el tiempo transcurrido.
Implementamos la regla de Simpson con el vector de las medidas definitivas.
Calculamos el área con la regla de Simpson.
Calculamos velocidad y posición.

Como ya se ha dicho (y presupuso al inicio del proyecto), estos valores no son fiables, con lo que se pasó a implementar el software del sensor VL53L0.

4.2.2 Software del sensor ToF VL53L0

Con el objetivo de tener gran precisión, finalmente se utilizó este sensor de distancia que se basa en el ToF o Time of Flight, tal y como se ha explicado anteriormente. Además, se recuerda al lector que conectaremos el sensor por I2C gracias a la librería *wire.h* y seremos capaz de obtener directamente la distancia en mm hasta el objeto más cercano con la librería *VL53L0X.h* con lo que simplemente al lector se le explicará a continuación el pseudocódigo de las funciones *calculavelocidadpush()*, la cual mide la VMP de un empuje y de *calculavelocidadpull()*, que mide la VMP de un tirón.

4.2.2.1 Calculavelocidadpush()

Para que la explicación del pseudocódigo sea más sencilla de seguir para el lector, se va a explicar cómo es un movimiento de empuje de forma rápida. Hay que tener en mente una sentadilla o un press de banca, que a fin de cuentas es lo que en principio va a medir esta función. En ambos movimientos el atleta empieza el recorrido en el punto más alto del movimiento, para luego empezar la fase excéntrica e ir descendiendo la barra poco a poco hasta llegar a su punto más bajo. Finalmente empezará la fase concéntrica hasta el punto más alto en el que estábamos en el inicio, una vez que el atleta realice este ciclo, el movimiento se da por terminado. Esta función se encarga de medir la VMP y el ROM del recorrido.

Figura 4.2.2.1: A la izquierda inicio y final del movimiento, a la derecha el atleta se sitúa en el punto más bajo que coincide con el inicio de la fase concéntrica

While(No ha terminado la repetición)

If(Es la primera muestra, realizamos un ciclo de gran duración para estabilizar el punto más alto del movimiento que será al que retornará)
Realizamos un LPF para eliminar ruido
If(medidaprevia> medidadefinitiva)
Está bajando
Else if(medidaprevia< medidadefinitiva, no es la primera iteración Y se ha desplazado más de 10 cm)
Está subiendo
Calculamos los puntos más altos y bajos (en teoría el alto está calculado, pero por si hay alguna anomalía)
If(está subiendo Y la medidadefinitiva está por encima del punto bajo)
Empezamos a cronometrar en microsegundos con la función micros();
If(Está subiendo Y medidadefinitiva>= puntoalto- 1cm) //Tomamos 1cm de margen en caso de que se produzca alguna anomalía
Paramos cronómetro y calculamos velocidad y ROM.
Terminarepeticion = true.

Un aspecto a tratar sobre esta función y la siguiente, es la explicación de la función *micros()*. Esta función devuelve el número de microsegundos desde la que placa Arduino empezó a ejecutar el programa actual. Este número se desbordará (volverá a

cero), después de aproximadamente 70 minutos, pero dado que este proyecto toma valores muy cortos, sería casi imposible coincidir con el minuto 70, con lo que no habría problema al usar esta función. En placas Arduino de 16 MHz (por ejemplo, Duemilanove y Nano), esta función tiene una resolución de cuatro microsegundos (es decir, el valor devuelto es siempre un múltiplo de cuatro). En las placas Arduino 8 MHz (por ejemplo, la LilyPad), esta función tiene una resolución de ocho microsegundos. Esta función retorna el valor en formato long sin signo. [15]

4.2.2.2 Calculavelocidadpull()

Al igual que se hizo en el apartado anterior, se procede a explicar en qué consiste un movimiento de tirón: El atleta traccionará de un objeto, ya se encuentre en el suelo o por encima de su cabeza, el primer caso correspondería con un peso muerto y el segundo con una dominada. En ambos casos el atleta se encuentra en el punto más bajo del movimiento al inicio, comienza a tirar, por lo que comienza también la fase concéntrica, hasta llegar al punto más alto del movimiento. Posteriormente, empieza la fase excéntrica para volver al punto inicial. Al igual que en la anterior función, la función mide la VMP de la fase concéntrica y el ROM realizado.

Figura 4.2.2.2: A la izquierda inicio del movimiento de un peso muerto, a la derecha la fase final

While(no ha terminado la repetición)

If(Es la primera muestra, realizamos un ciclo de gran duración para estabilizar el punto más bajo del movimiento)
Realizamos un LPF para eliminar ruido
If(medidaprevia>medidadefinitiva)
Está bajando y no subiendo
Else If(medidaprevia<medidadefinitiva Yno es la primera iteración)
Está subiendo y no bajando
Calculamos los puntos más altos y bajos
If(Está subiendo Y medidadefinitiva> puntobajo + 1cm) //Tomamos 1cm de margen en caso de que se produzca alguna anomalía.
Empezamos a cronometrar en microsegundos con la función micros();

If(Está bajando Y puntoalto – medidadefinitiva > 2 cm) //Tomamos 2cm porque significa que está bajando de verdad y no que por falta de fuerza llegue a un steaking point en el que decaiga un poco según la medida del sensor.
Paramos cronómetro y calculamos velocidad y ROM.
Terminarepeticion = true.

4.2.3 Software del servidor

Para la realización de este proyecto, plasmaremos todos los datos de interés en un servidor web integrado dentro de la placa, al cual nos conectaremos mediante conectividad Wi- Fi a la IP 192.168.4.1 gracias a la función *WiFi.softAP(ssid,password);* que nos permite arrancar la placa en modo AP.

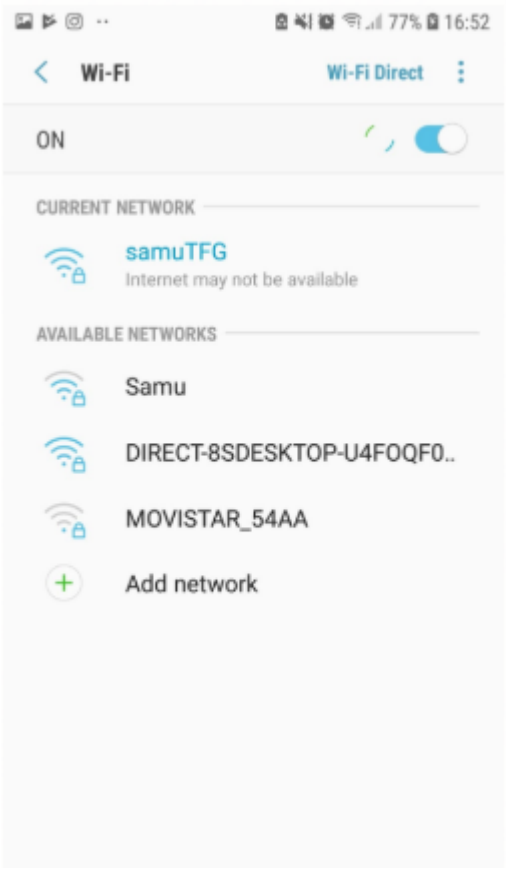

Figura 4.2.3.1: Conexión a la placa mediante Wi- Fi

Además, gracias a la librería ESP8266WebServer explicada anteriormente, realizamos un menú inicial mediante un subdirectorio donde se muestran todas las opciones disponibles en el servidor. Para ello, con esta librería y con la instrucción *server.on("/", handleRoot);* que creará este subdirectorio, es decir, deberíamos escribir en la barra de navegación 192.168.4.1/ para acceder a este subdirectorio, el cual contiene la función *handleRoot*, que tomamos como directorio raíz.

```
236 void handleRoot() {  //Esta subrutina es llamada cuando pones la IP en el navegador, es el root del server
237
238 String html = "<!DOCTYPE html> <html> <body> <h1>Conectado al BitxoFusion</h1></h1>";
239 html += "<meta name = \"viewport\" content = \"width = device-width, initial-scale = 1.0, maximum-scale = 1.0, user-scalable=0\">";
240 html += "<br><br><button><a href=\"estimasquat\">Estima SQ</a></button><br>";
241 html += "<br><button><a href=\"estimabanca\">Estima BP</a></button><br>";
242 html += "<br><button><a href=\"ecuacionRM\">Estima RM</a></button><br>";
243 html += "<br><button><a href=\"estimasquatreal\">Estima Squat Real</a></button><br>";
244 html += "<br><button><a href=\"estimabenchpressreal\">Estima Bench Press Real</a></button><br>";
245 html += "<br><button><a href=\"estimadeadliftreal\">Estima Dead Lift Real</a></button><br>";
246 html += "<br><button><a href=\"pushspeed\">Calcula Velocidad de Empuje</a></button><br>";
247 html += "<br><button><a href=\"pullspeed\">Calcula Velocidad de Tiron</a></button><br>";
248 html += "</body></html>";
249 html += "<meta http-equiv=\"refresh\" content=\"2\">";
250   server.send(200, "text/html", html);
251
252 }
```

Figura 4.2.3.2: Código del directorio raíz.

Como se puede intuir, este código es un mero HTML, con la peculiaridad de que en Arduino no se puede escribir HTML directamente, sino que está adaptado para Arduino con ciertas peculiaridades. Aunque son más notorias si añadimos algún tipo de CSS (Cascading Style Sheets), para evitar complicaciones innecesarias, este servidor sólo dispone de código HTML, ya que al fin y al cabo este proyecto podría ser un prototipo de algo mucho mayor. Para que el lector entienda mejor como se manejan los subdirectorios, toda instrucción *server.on()* está formada por un directorio y una función que no debe tener ningún parámetro de entrada/ salida, por lo que ha dificultado la creación de ciertos subdirectorios, sin embargo, se ha podido solventar con el uso de variables globales.

Con esto realizamos una pequeña botonera que nos lleva a los distintos directorios posibles para mejor manejo del servidor. La peculiaridad es la instrucción de la línea 239, que simplemente ajustamos las medidas tanto del texto como del HTML a las dismensiones del dispositivo que se conecta a la placa, para evitar distorsiones. Por último, "mandamos" la función *handleRoot* al servidor con la instrucción *server.send* tal y como se hace en la línea 250.

Figura 4.2.3.3: Resultado final del directorio raíz.

Tanto para las funciones de Estima SQ como Estima BP hemos hecho uso de las ecuaciones de Badillo para estimar el 1RM en base a los datos que nos da el usuario de cuántos kilos y a qué velocidad se han levantado. Esto es posible ya que se ha realizado un formulario de tipo POST, haciendo así los datos no visibles en la URL. Por otra parte, de nuevo gracias a la librería ESP8266WebServer, disponemos de la instrucción *server.arg();* que nos permite obtener el valor de un name en un string, para posteriormente manejarlo como si de un valor normal dentro del programa se tratase con la intrucción .toFloat(); que nos permite tranformar un valor a un float.

Para la función Estima SQ la ecuación de Badillo de estimación del RM es:

%1RM = -5.961*VPM^2- 50.71* VPM +117. Dicha ecuación tiene un coeficiente de correlación $R^2 = 0.981$ y una desviación del %1RM del 3.56%, por lo que es bastante exacta.

Para la función Estima BP la ecuación de Badillo de estimación del RM es:

%1RM = 8.4326* VPM^2- 73.501* VPM + 112.33 . Dicha ecuación tiene un coeficiente de correlación $R^2 = 0.954$ y una desviación del %1RM del 4.02%, por lo que también goza de gran exactitud.

Figura 4.2.3.4: Desarrollo final de la función *Estima SQ* (*Estima BP* es semejante pero con su ecuación propia).

Pese a la gran precisión de estas ecuaciones, estas están hechas en base a una media de población en un estudio, es decir, no están individualizadas. Sin ir más lejos, está demostrado que atletas profesionales, pueden mover %RM a velocidades menores que alguien novato, por lo que estas ecuaciones para todo el mundo no tendría un R^2 de 0.981 y 0.954 sino que podría ser de mucho menos. Es por eso que se ha realizado un subdirectorio para crear una ecuación propia del RM para cada atleta en base a cada ejercicio, incluso para un tirón como peso muerto que no hay ecuación propia, esto se

consigue obteniendo los valores de velocidad para unos %RM que nos interesan, en concreto desde el 70 al 100%RM. Teniendo los porcentajes y las velocidades, podemos establecer una ecuación propia con un método de mínimos cuadrados. El método de mínimos cuadrados calcula a partir de los N pares de datos experimentales (Xi,Yi) los valores de la pendiente b y del punto de corte con el eje a que mejor ajustan los datos a una recta, Y= a+ bX, donde X es el %RM e Y es la velocidad.

$$a=\frac{(\sum_{i=1}^{N} Y_i)(\sum_{i=1}^{N} X_i^2)-(\sum_{i=1}^{N} X_i)(\sum_{i=1}^{N} X_i Y_i)}{N(\sum_{i=1}^{N} X_i^2)-(\sum_{i=1}^{N} X_i)^2}$$

Donde a se obtiene de la expresión:

$$b=\frac{N(\sum_{i=1}^{N} X_i Y_i)-(\sum_{i=1}^{N} X_i)(\sum_{i=1}^{N} Y_i)}{N(\sum_{i=1}^{N} X_i^2)-(\sum_{i=1}^{N} X_i)^2}$$

Y b la pendiente se obtiene:

A nivel de código simplemente se ha realizado un formulario de tipo POST donde se piden estos valores y el ejercicio para el cual quiere dicha ecuación, en nuestro caso disponemos de sentadilla, press de banca y peso muerto. Al igual que antes, obtendremos los valores con la función *server.arg* y los transformaremos a float con *.toFloat();* por último se realiza un triple if para comprobar a qué ejercicio van los parametros dados.

Figura 4.2.3.5: Desarrollo final del subdirectorio *ecuacionRM*

Seguidamente, creamos 3 subdirectorios, uno por ejercicio para estimar el 1RM real del día y de un atleta en específico. Al igual que antes creamos un formulario con el método POST, donde pediremos al usuario la carga y la velocidad ejecutada, previamente habremos pasado los parámetros que hacen posible la ecuación de mínimos cuadrados. Dicha ecuación nos dará qué porcentaje del RM estamos manejando en base a la VMP y posteriormente con una regla de 3 y la carga, obtendremos el valor en kilogramos del 1RM del día.

Figura 4.2.3.6: Desarrollo final del subdirectorio *estimasquatreal* (hay otros dos semenjantes para los demás ejercicios)

Obviamente no se puede hacer uso de estos directorios si previamente no hemos calculado la VMP con el software desarrollado con el sensor. Como ya se ha explicado como funciona este código, simplemente en el sevidor se ha realizado un subdirectorio para empujón y otro para tirón, y nada más acceder a estos empieza a funcionar su correspondiente código. El subdirectorio simplemente mostrará un título, la velocidad de ejecución en m/s y el ROM en mm.

Figura 4.2.3.7: Desarrollo final de *pushspeed*

Figura 4.2.3.8: Desarrollo final de *pullspeed*

5. Conclusiones y líneas futuras

En este capítulo se completa la memoria, exponiendo las conclusiones a las que se han llegado y un resumen en comparación con Speed4Lifts y por cámara a 60pfs. En la sección 5.2 se explican las líneas de trabajo por las que se puede continuar este trabajo para su mejora respecto a este tipo de tecnologías.

5.1 Conclusiones

Debido al auge de los deportes de fuerza (powerlifting y halterofilia) en los últimos años, en gran parte gracias al crossfit y a referentes en redes sociales, ha surgido la necesidad de cómo hacer más eficiente este tipo de entrenamientos. Gracias a Juan José González Badillo, se han llevado a cabo diversos estudios que relacionan el %RM con la velocidad de ejecución. Universidades han desarrollado costosos dispositivos para medir este tipo de datos: VMP, ROM, potencia, fatiga… Sin embargo, unos jóvenes talentos desarrollaron un modelo más barato a la par que eficaz (R^2= 0.9681 con el T-Force) siguiendo con la misma tecnología añadiendo la conectividad Wi-Fi.

El estudio que se ha llevado en este proyecto es verificar cuán eficaces son las tecnologías inalámbricas para medir este tipo de datos en el levantamiento de cargas en powerlifting. Utilizando una placa Arduino con conectividad Wi-Fi, a la cual accederemos a un servidor web con IP 192.168.4.1, con su correspondiente HTML para poder manejar la placa y el sensor que se ha utilizado, un ToF VL53L0X, con láser infrarrojo que se conectará a la placa con el protocolo I2C. Previamente a este sensor, se utilizó un acelerómetro ADXL335, sin embargo, no se obtuvieron los resultados deseados ya que tenía demasiada sensibilidad y con las prestaciones de esta placa no se podía calcular el módulo del movimiento por no disponer de 3 pines analógicos y la posibilidad de implantar un convertidor ADC era inviable.

Se ha hecho uso de diversas librerías tanto para la conectividad Wi-Fi, I2C y para la creación del servidor, usando funciones propias de las librerías que facilitan el desarrollo. En este servidor incluimos un directorio raíz para acceder a los diversos subdirectorios a través de una botonera, entre estos subdirectorios incluimos las ecuaciones de Badillo para sentadilla y press de banca que estiman el RM en base a sus estudios metiendo como parámetros la velocidad de ejecución y la carga levantada. Además, se ha creado un subdirectorio para crear una ecuación propia para el atleta y para cada ejercicio, ya que el estudio de Badillo es una media entre distintos tipos de atletas y puede no ser 100% exacta para cada levantador, esta recta requiere ciertos datos como son la velocidad a la que el atleta levanta X%RM, el usuario proporcionará estos datos a través de un formulario HTML tipo POST. Finalmente, se han creado dos subdirectorios para medir la velocidad de ejecución, uno para los empujes y otro para los tirones, con el fin de saber la velocidad de ejecución en tiempo real en los entrenos.

Como se ha explicado anteriormente se ha pretendido realizar una comparación de este proyecto con el Speed4Lifts y por cámara a 60 fps, sin embargo, hubo ciertos problemas a mitad del estudio, ya que las soldaduras no aguantaron bien y por un golpe contra el soporte el sensor dejó de hacer contacto bien con la placa y no se pudo medir más. No obstante, se recogen en esta tabla los 3 levantamientos que dieron tiempo a estudiar en un press de banca con un atleta el cual su 1 RM aproximadamente era de 102.5 kg:

Carga	%RM	VMP Cámara (m/s)	VMP Speed4Lifts (m/s)	VMP proyecto (m/s)	ROM (cm)
60kg	58.53	0.81	0.83	0.76	35/34/38
70kg	68.28	0.65	0.66	0.60	34/34/37
80kg	78.04	0.51	0.5	0.47	34/35/35

Tabla 5.1: Comparación de las distintas tecnologías

En conclusión, como se puede apreciar, aunque sólo haya podido ser en 3 muestras, a medida que el %RM sube, la correlación entre las velocidades incrementa debido a que tiene más muestras y la función del proyecto puede detectar más fácilmente cuál es el punto más alto. Respecto al ROM no hay grandes diferencias y la razón sigue siendo la misma, al realizar el levantamiento con gran velocidad, posiblemente se haya tomado una muestra superior a la que tomó el Speed4Lifts, es decir, que el muestreo no pasase por el mismo punto que pasó en el inicio del levantamiento, sino que pasó directamente a un valor más alto.

5.2 Líneas futuras

En esta sección se introducen algunas de las posibles líneas futuras que quedan abiertas para continuar con el trabajo realizado en este proyecto.

A pesar de que las pruebas realizadas con el sensor VL53L0X han sido pocas y por tanto no suficientes para sacar nada en claro al completo, una posible mejora sería utilizar un sensor más reciente de la misma familia, el VL53L1X, que puede medir hasta 4m de longitud, ya que este, tal y como se especificó, sólo tiene 120 cm de rango, con lo cual movimientos como la arrancada o el clean and jerk de halterofilia no se podrían medir ya que el atleta parte con la barra desde el suelo y la pasa por encima de su cabeza, con lo que la altura sería mayor a 120 cm.

Además, el trabajo de soldar lo debería hacer un personal más cualificado y experimentado para evitar los problemas que se han tenido en el testeo, con el fin de que el sensor siempre pueda medir bien. Para reforzar este punto se podría crear un encapsulado con una impresora 3D evitando así golpes en la placa, en el sensor y demás componentes, añadiendo así más seguridad.

Una vez que se ha hecho esto y comprobado que las medidas para 1 repetición son correctas, se podría mejorar el código para poder medir X repeticiones midiendo así el % de fatiga intra-serie, haciendo así aún más eficiente el entreno y no sólo sabiendo el RM del día. Esta opción resultaría de gran utilidad a atletas de élite, que tienen incluso 2 o 3 entrenos diarios y se debe administrar correctamente el volumen y la intensidad de los entrenos para que el atleta se recupere de una sesión a otra.

Por último, para una mejor representación se podría añadir un CSS, con el que podríamos incluir gráficos, viendo así el avance del atleta con el tiempo. Además, ya que este dispositivo lo pueden utilizar varios atletas, la creación de una base de datos, y por consiguiente un inicio de sesión dentro del mismo servidor, sería de vital importancia para que cada atleta tenga sus entrenos y no se mezclen con los de otro atleta.

Lista de acrónimos

AP	Access Point
R^2	Coeficiente de Correlación
RM	Repetición Máxima
VMP	Velocidad Media Propulsiva
ROM	Range Of Movement
G	Aceleración de la Gravedad (1G= 9.8m/s^2)
ToF	Time of flight
VCSEL	Vertical Cavity Surface- Emitting Laser
SPAD	Single Photon Avalanche Diodes
FOV	Field of View
WECA	Wireless Ethernet Compability Alliance
LAN	Local Area Network
SSID	Service Set Identifier
DHCP	Dynamic Host Configuration Protocol
URL	Uniform Resource Locator
UART	Universal Asynchronous Receiver-Transmitter
SPI	Serial Peripheral Interface
LPF	Low Pass Filter
HTML	HyperText Markup Language
CSS	Cascading Style Sheets

Bibliografía

[1]Juan José González Badillo, Luis Sánchez Medina, Fernando Pareja Blanco, David Rodríguez Rosell. *La velocidad de ejecución como referencia para la programación, control y evaluación del entrenamiento de fuerza*. 2014.

[2] Borja Albalá Gómez. *Validez y fiabilidad de un sensor basado en acelerometría y de un transductor lineal de posición para medir la velocidad de ejecución en el ejercicio de press de banca.*

[3] Data- Sheet del componente.

[4] Wikipedia: Regla de Simpson; Última visita: 20/7/2018.

[5] Grupo de Ingeniería Telemática UC. *Redes de Área Local Inalámbricas: el estándar IEEE 802.11.*

[6] Wikipedia: ESP8266; Última visita: 20/7/2018.

[7] GitHub: Librería ESP8266WiFi.h; Última visita: 20/7/2018.

[8] GitHub: Librería WifiClient.h; Última visita: 20/7/2018

[9] GitHub: Librería ESP8266WebServer; Última visita: 20/7/2018.

[10] GitHub: Librería Wire.h; Última visita: 20/7/2018

[11] Arduino.cc: Wire; Última visita: 20/7/2018

[12] GitHub: Librería VL53L0X; Última visita: 20/7/2018

[13] Foro de Arduino: Aprendiendoarduino; Última visita: 20/7/2018.

[14] Prometec, página de tecnología especializada en Arduino.

[15] Foro personal de Arduino: Manueldelgadocrespo; Última visita: 20/7/2018.