PROTOTIPO DE DISPOSITIVO DE MEDIDA DE RENDIMIENTO EN DEPORTES DE CONTACTO BASADO EN UN ACELERÓMETRO TRIAXIAL Y COMUNICACIÓN A DISPOSITIVO MÓVIL

Autor: Adrián Luis Arándiga Martínez
Tutor: Salvador Coll Arnau
Julio 2016

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Universidad Politécnica de Valencia
Escuela Técnica Superior de Ingeniería del Diseño
Grado en Ingeniería Electrónica Industrial y Automática

ÍNDICE

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

1. Introducción

En este documento se redactará el trabajo de fin de grado requerido para concluir el grado en ingeniería electrónica industrial y automática. Esto implica la obtención del título en la Escuela Técnica Superior de Ingeniería del Diseño de la Universidad Politécnica de Valencia.

1.1 Objeto del proyecto

Se diseñará un sistema para la recopilación de datos significativos en los deportes de contacto basándose en un acelerómetro triaxial. El hardware será ligero y de tamaño reducido para que pueda ser colocado en la muñeca.

Se desarrollará una aplicación para dispositivos móviles que procesará dicha información y proporcionará interacción con el usuario empleando como prototipo base el kit de desarrollo CC2541 SensorTag.

1.2 Antecedentes

Cada vez es más común la integración de la tecnología a la vida diaria. Así pues el concepto de monitorizar la actividad física y hacer los datos accesibles al usuario es manifiesto tanto en el ámbito de la salud como en el ocio.

El dispositivo que se trata en este documento, al que denominaremos Smartfight de aquí en adelante, se categoriza dentro de la denominada tecnología ponible. Accesorios cotidianos como gafas, muñequeras, textiles, relojes y zapatillas entre otros que integran electrónica para ampliar sus funcionalidades.

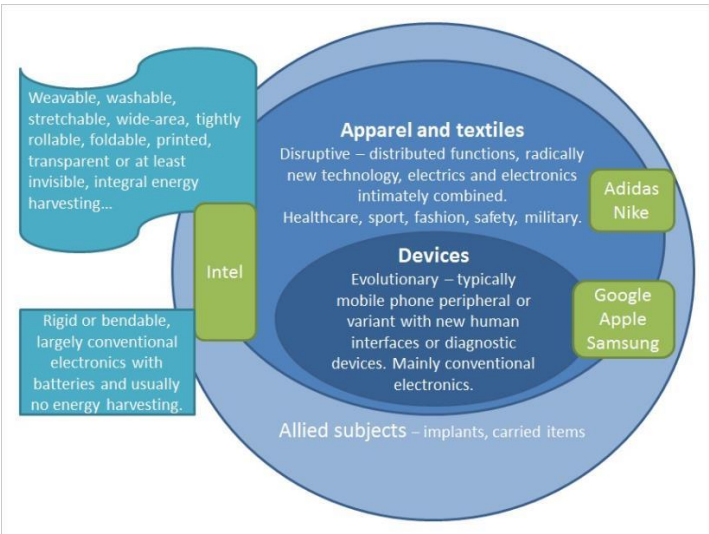


Imagen 1 – Tipos y características de las tecnologías ponibles

Dentro de este tipo de tecnología, el Smartfight une el deporte y la tecnología para ayudar al control del rendimiento personal y a la superación de objetivos, ya sea como aficionado del deporte o a nivel profesional.

Se pueden encontrar dispositivos similares en el mercado que integran la tecnología y el deporte:

- Smartwatch: Es un reloj de pulsera cuya funcionalidad va más allá de comprobar la fecha y hora. Modelos recientes son capaces de sincronizarse con un Smartphone para recibir las llamadas y notificaciones al smartwatch, reproducir música, controlar la cámara del dispositivo móvil y monitorizar actividad física mediante el uso de podómetros.

- SmartBand: Comparte similitudes con el smartwatch, siendo capaz de sincronizarse con un Smartphone y recibir notificaciones. Comparte el concepto del Smartfight de monitorizar la actividad física, enviar los datos mediante Bluetooth al Smartphone y procesar y presentar los datos en una aplicación para que el usuario pueda analizarlos. Suelen contar con un pulsímetro y un acelerómetro para recolectar la información sobre la actividad del usuario.

- Portable Punch Analyzer Lucerys GXP: Este dispositivo es comparable al Smartfight ya que su aplicación es exclusiva a los deportes de contacto. Mide la aceleración y fuerza de los golpes si es usado golpeando un saco de boxeo y los representa en una pantalla integrada en el dispositivo. El dispositivo es cerrado y no se comunica con un Smartphone. Dejo de producirse y no existen unidades a la venta en la actualidad.

Imagen 2 – Lucerys GXP

Podría considerarse al Smartfight una combinación entre la Smartband y el Lucerys GXP puesto que se trata de un dispositivo focalizado en los deportes de combate y se comunica con un Smartphone para la interacción con el usuario.

1.3 Justificación del proyecto

El Smartfight sirve el propósito de focalizar la tecnología ponible en una categoría deportiva concreta, los deportes de contacto y concretamente en este documento, el boxeo.

Se pretende llevar estos dispositivos más allá de una monitorización de la actividad diaria general y explorar las posibilidades que ofrece esta tecnología en el ámbito deportivo.

La aceptación de la tecnología ponible por parte del consumidor está comprobada. Las cifras de venta de estos dispositivos crecen cada año y se prevé que seguirán creciendo, debido a las campañas publicitarias que fomentan la tecnología ponible como un nuevo estilo de vida.

Las ventas estimadas del 2015 y previsión de los siguientes años según Gartner es la siguiente:

Device	2015	2016	2017
Smartwatch	30.32	50.40	66.71
Head-mounted display	0.14	1.43	6.31
Body-worn camera	0.05	0.17	1.05
Bluetooth headset	116.32	128.50	139.23
Wristband	30.15	34.97	44.10
Smart garment	0.06	1.01	5.30
Chest strap	12.88	13.02	7.99
Sports watch	21.02	23.98	26.92
Other fitness monitor	21.07	21.11	25.08
Total	**232.01**	**274.59**	**322.69**

Tabla 1 – Estimación de ventas mundiales de dispositivos ponibles (millones de unidades)

Así pues el Smartfight supone una adición a un mercado en expansión cubriendo un área de aplicación donde no existen alternativas en la actualidad. Debido a la amplia aceptación de los Smartphone el hecho de que el Smartfight requiera de la posesión de uno no supone una limitación, según un informe realizado por Simon Kemp el 51% de la población posee un Smartphone.

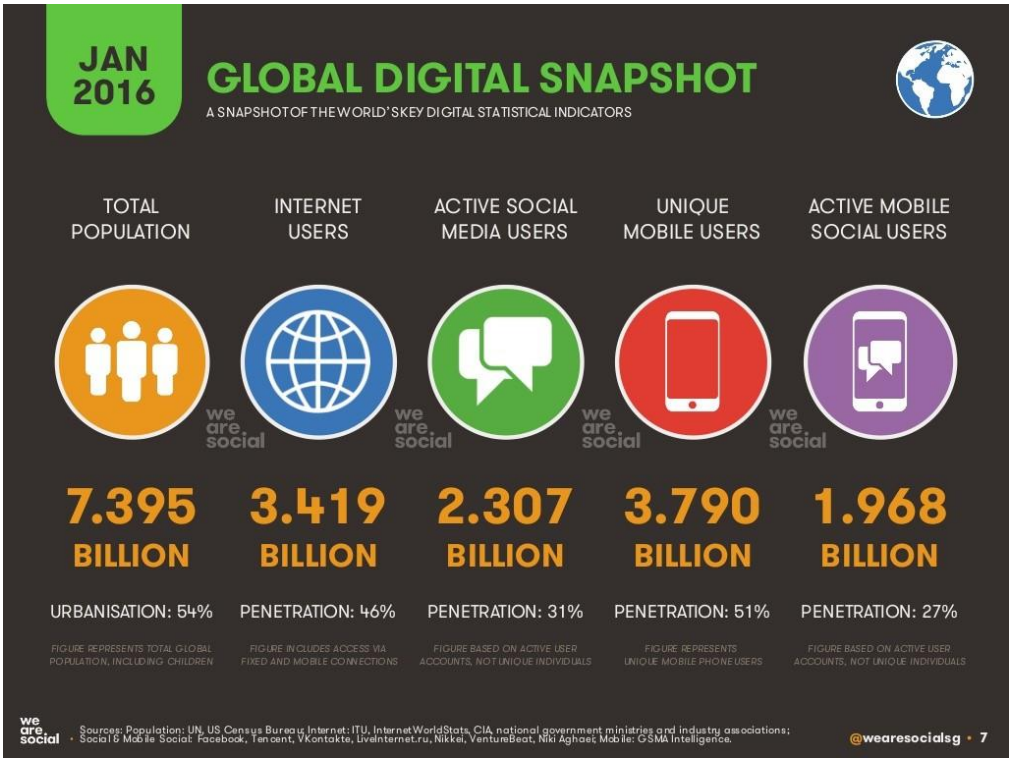


Imagen 3 – Población mundial y uso de tecnología

En España en concreto, ya en 2014 un 87% de la población poseía un Smartphone como se observa en un informe realizado por Ditrendia sobre Móviles en España y el mundo en 2015:

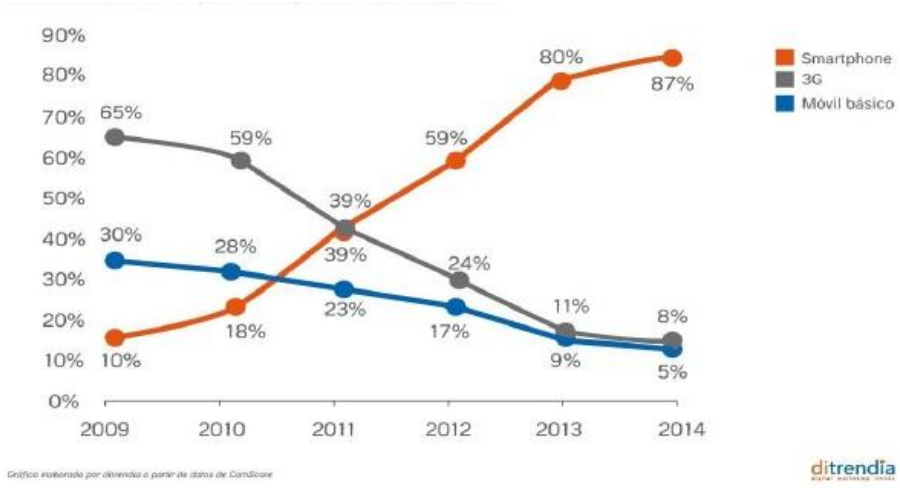


Imagen 4 – Uso de Smartphone en España

Además las ventas se mantienen estables rondando las 340 millones de unidades cuatrimestrales mundiales según IDC en 2015/2016.

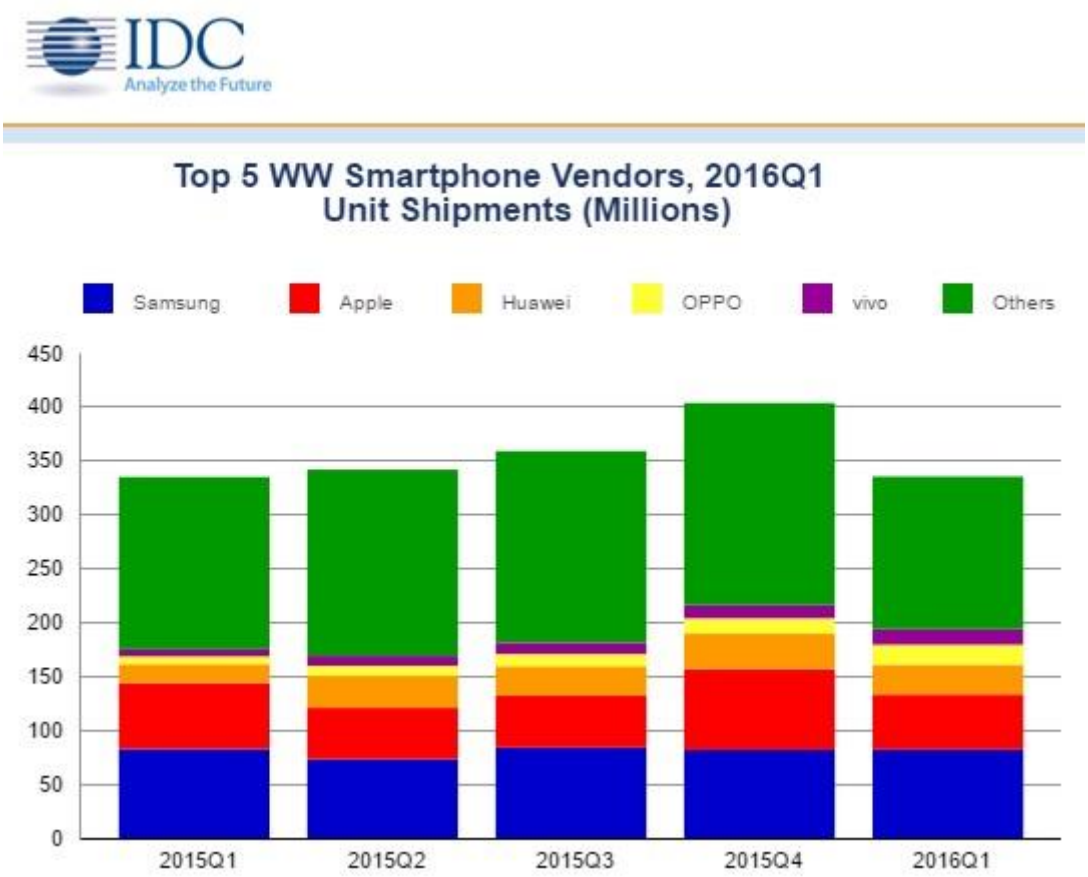


Imagen 5 – Ventas mundiales de Smartphones (millones de unidades)

Con los datos disponibles el Smartfight supone introducir un dispositivo cuya funcionalidad es única dentro de los dispositivos ponibles y que se respalda en una tecnología ya arraigada como son los Smartphones.

2. Descripción de soluciones alternativas.

En este apartado se expondrán distintas opciones de diseño para el dispositivo de medida de rendimiento en deportes de contacto. Las alternativas se plantearán con respecto al siguiente diagrama de bloques:

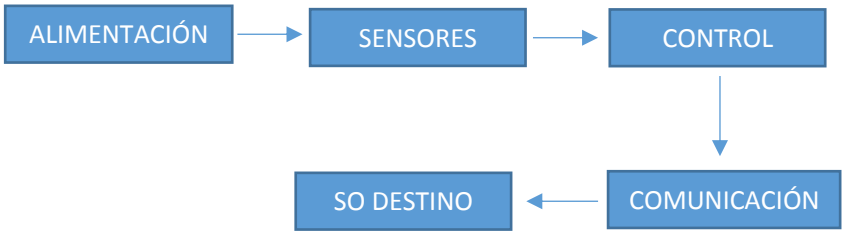


Imagen 6 - Diagrama de bloques de funcionamiento general

2.1 Alimentación

Siendo uno de los objetivos hacer del Smartfight una tecnología ponible, será necesario encontrar una solución a la alimentación independiente de la red eléctrica.
Para otorgarle autonomía pueden utilizarse:

2.1.1 Pilas

Son un dispositivo que transforma energía química en energía eléctrica. Su estructura fundamental consiste en dos electrodos metálicos introducidos en una disolución conductora de la electricidad.

Dentro del amplio abanico de tipos de pilas normalizadas que existen, El modelo más adecuado para el Smartfight sería la pila de botón debido a su tamaño reducido.
Una de las ventajas de esta solución es su simplicidad de diseño, no es necesario complementar la pila con nada más que un conector. Por lo tanto, al minimizar elementos en el diseño, se reduce el precio de producción.

Como desventajas el uso de pilas implica la necesidad de elementos externos para que el dispositivo funcione, además su impacto medioambiental es mayor.

2.1.2 Baterías

Dispositivo con la capacidad de almacenar energía que puede utilizarse para proporcionar tensión eléctrica a un circuito. Ofrecen mayor libertad de diseño ya que pueden encontrarse en variedad de formas y tamaños. Además cuentan con una vida útil extensa por lo que no necesitan reemplazarse con regularidad. Tienen diferentes características dependiendo de los materiales de fabricación.

2.1.2.1 Nickel Cadmio

El cátodo se fabrica con hidróxido de níquel, el ánodo con un compuesto de cadmio y el electrolito con hidróxido de potasio.

La descarga completa de la batería no afecta al funcionamiento de la batería, no obstante es susceptible a la pérdida de capacidad si no se realiza una carga completa (efecto memoria). Su densidad de energía es baja por lo que tienen poca capacidad.

2.1.2.2 Ion-Litio

Utilizan un ánodo de grafito y un cátodo de óxido de cobalto. Una descarga completa de la batería afecta negativamente a su rendimiento. Pero no les afecta el efecto memoria y su densidad de energía es elevada

2.1.2.3 Polímero de Litio

Se diferencia de las baterías convencionales en que emplea como electrolito un polímero sólido.

Como puntos negativos su densidad de energía es inferior a la que ofrecen las baterías de Ion-litio, sus costes de manufactura son mayores y la descarga completa merma su rendimiento.

Por otra parte cuenta con ventajas notables: no les afecta el efecto memoria, pueden fabricarse con dimensiones más reducidas que el resto, son más ligeras ya que el electrolito sólido elimina la necesidad de añadir una placa metálica en el empaquetado y son más resistentes a sobrecargas.

Uno de los requerimientos de la tecnología ponible es que sea ligera y lo más cómoda posible para el usuario. El tamaño reducido y el peso liviano de la batería de polímero de litio son las características determinantes que la convertirán en la elegida para encargarse de la alimentación del Smartfight.

Imagen 7 – Tamaño batería de Polímero de Litio

2.2 Control

La función del bloque de control consistirá en recibir los datos captados por los sensores y acondicionarlos de acuerdo con el método de comunicación elegido para que puedan ser recibidos por el dispositivo móvil sin problemas. Es crucial que el tamaño sea reducido y sea independiente de elementos externos, así pues la solución que cumple con las necesidades requeridas es el microcontrolador.

2.2.1 Microcontrolador

Es un sistema con periféricos, memoria y procesador que se usa como sistema embebido.
Se emplean en sistemas donde el espacio es limitado debido a sus pequeñas dimensiones. Existe gran variedad de modelos con diversos periféricos lo que permite adaptarlos a aplicaciones concretas. Además cuentan con un nivel de procesamiento elevado y bajo consumo.

Existe una gran cantidad de fabricantes y modelos de microprocesadores entre ellos:

2.2.1.1 PIC Microchip

Son una familia de microcontroladores fabricados por Microchip. Algunas de las características de su arquitectura son:

- Área de código y de datos separados (Arquitectura Harvard)
- Reducido número de instrucciones de longitud fija
- La mayoría de las instrucciones duran un solo tiempo de instrucción
- Pequeña cantidad de espacio de datos direccionables.

Al ser un tipo de microcontroladores muy extendido existe gran cantidad de información sobre ellos, y el propio fabricante ofrece un entorno de programación gratuito.

2.2.1.2 Arduino

Consiste en una plataforma de código abierto basada en hardware y software sencillos.

Sus productos son placas de desarrollo ya ensambladas por lo que la libertad de diseño es limitada, el precio es superior al de otros microcontroladores aislados, y en general su tamaño hace que no sean apropiados para un dispositivo ponible, aunque si existe catálogo de placas de tamaño reducido enfocadas a la tecnología ponible, están limitadas a USA.

Como punto favorable, al ser una plataforma de código abierto, existe una gran cantidad de herramientas y librerías disponibles y cuenta con una comunidad muy abierta.

Imagen 8 – Wereable Arduino

2.2.1.3 Otros fabricantes

El mercado de microcontroladores es muy amplio y son muchos los fabricantes (Texas Instruments, Zilog, Analog Devices, STMicroelectronics…) cada uno con una gran cantidad de modelos con distintas características a elegir según las necesidades del diseño. Para el Smartfight, por motivos de disponibilidad se elegirá un microcontrolador del fabricante Texas Instruments que cumpla con los requisitos.

2.3 Sensores

Se encargarán de recolectar los datos de los golpes. Las medidas más interesantes para determinar el rendimiento en deportes de contacto serán la aceleración y la orientación, por ello los siguientes sensores son los más adecuados para este proyecto:

2.3.1 Acelerómetros monoaxiales

Son dispositivos electromecánicos capaces de medir aceleraciones, ya sean estáticas como la gravedad, o dinámicas como vibraciones y movimientos. Los acelerómetros monoaxiales captan las aceleraciones en un solo eje.

2.3.2 Acelerómetros triaxiales

Su función es la misma que la de los acelerómetros monoaxiales, no obstante esta variante mide las aceleraciones en 3 ejes orientados 90º respecto a cada uno. Esto permite que pueda determinarse la orientación del mismo gracias a la fuerza estática de la gravedad.

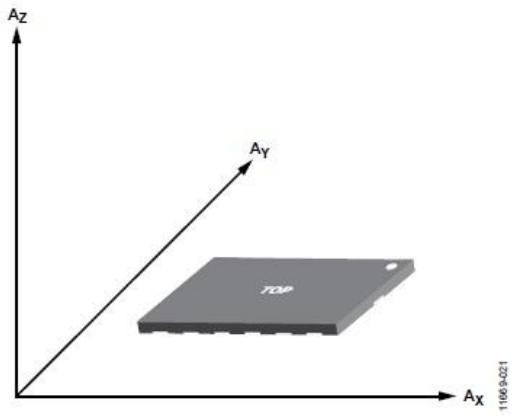


Imagen 9 – Ejes acelerómetro triaxial (la tensión de salida aumenta cuando se producen aceleraciones en el eje correspondiente)

Además con los acelerómetros triaxiales pueden determinarse los vectores tridimensionales de aceleración que generan los golpes en los deportes de contacto.

La orientación del dispositivo será útil para conocer la postura inicial y guardia del usuario.

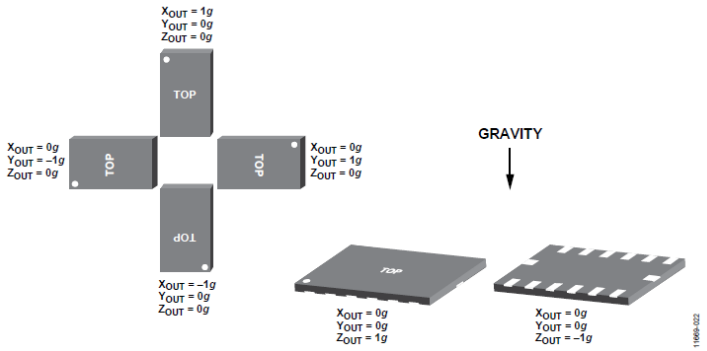


Imagen 10 – Respuesta del acelerómetro triaxial a la orientación

Y con los vectores de aceleración puede diferenciarse el tipo de golpe realizado (straight, hook, uppercut).

Debido a estas características se seleccionará un acelerómetro triaxial para recolectar los datos en el Smartfight.

2.3.3 Giroscopios

El uso de giroscopios permite determinar la rotación de un objeto en relación a los 3 ejes.

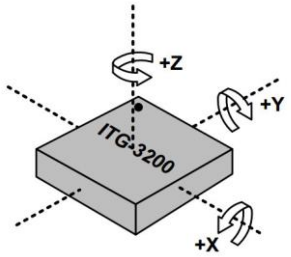


Imagen 11 – giroscopio

Este tipo de sensor sería útil para clasificar golpes en los deportes de contacto ya que cada técnica implica unas rotaciones distintas. Existen varios tipos de giroscopios afectados por principios físicos distintos (mecánicos, ópticos y electrónicos.)

2.4 Comunicación

El módulo de comunicación se encargará de transmitir la información del Smartfight al dispositivo móvil para que, una vez allí, sea procesada y representada.

Puesto que el objetivo del proyecto es diseñar un dispositivo cómodo de usar que permita llevar a cabo una sesión de entrenamiento ordinaria, sin limitar el movimiento del usuario, es imperativo que la comunicación sea inalámbrica. Además ha de ser compatible con la tecnología integrada en los Smartphones. Esto puede obtenerse con:

2.4.1 Bluetooth

Es un estándar de comunicación inalámbrica que conecta dispositivos dentro de cierta distancia. Requiere de un emparejamiento entre los dispositivos que deseen conectarse mediante Bluetooth por motivos de seguridad y permite crear una red de hasta 8 dispositivos conectados simultáneamente donde uno actúa como maestro y el resto como esclavos.

Opera mediante radiofrecuencias de 2,4 a 2,48GHz con amplio espectro y saltos de frecuencia con posibilidad de transmitir full dúplex (comunicación bidireccional simultánea).

Permite transmitir información a un máximo de 720kbit/s a un rango de 10 metros en los dispositivos Bluetooth de clase 2 que son los más comunes en uso doméstico.

Ventajas de este protocolo de comunicaciones son:

- Ampliamente extendido, Casi todos los teléfonos, portátiles y tablets cuentan con tecnología Bluetooth.
- Bajo consumo, lo que lo convierte en una opción a considerar en el diseño de sistemas que requieren una autonomía extensa.
- Sencillez de uso, El consumidor solo debe, en la mayoría de los casos, emparejar los dispositivos y comenzar con la comunicación.
- Económico, pueden encontrarse dispositivos Bluetooth para añadir a un proyecto a bajo precio.

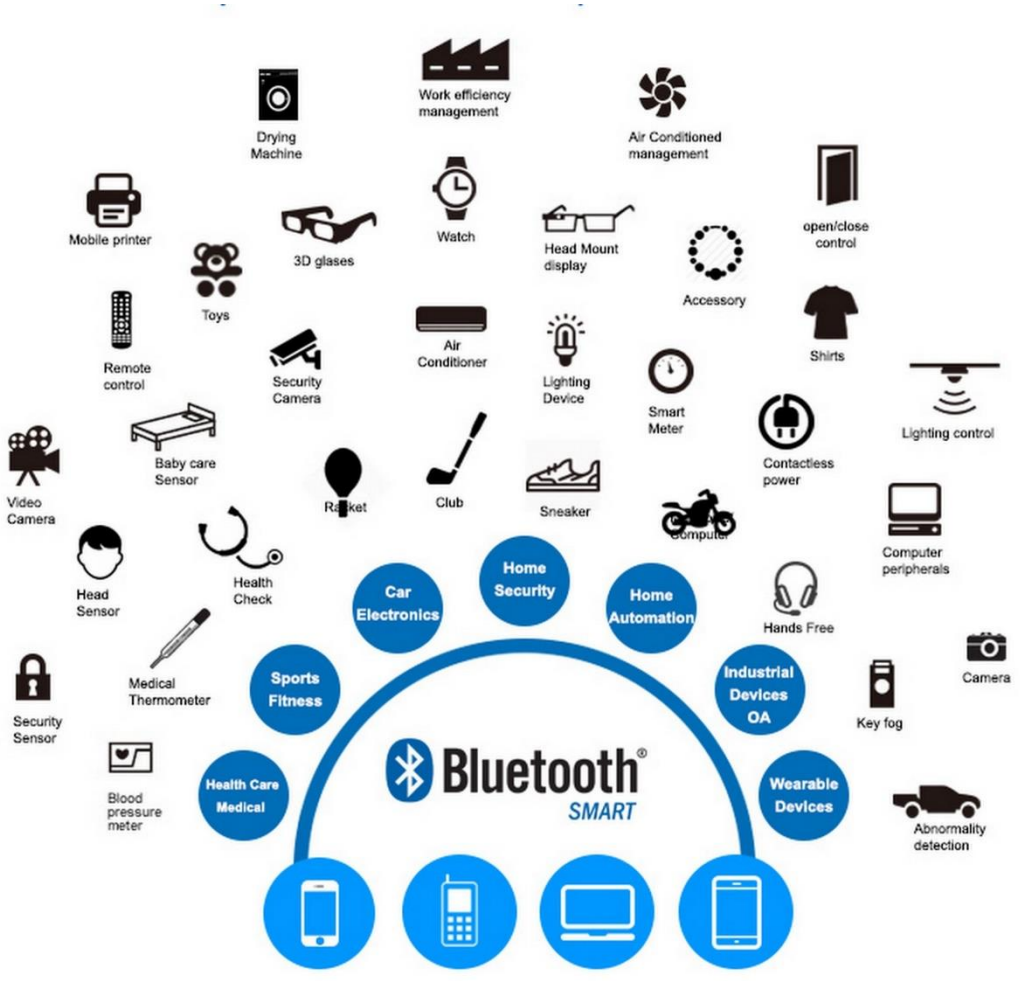


Imagen 12 – Productos que incorporan Bluetooth

El Bluetooth se ajusta a las especificaciones requeridas en el Smartfight, bajo consumo, sencillo, permite emparejar 2 dispositivos (Smartfight y Smartphone) y la mayoría de los Smartphones del mercado cuentan con esta tecnología. Por estos motivos se elegirá el Bluetooth para la comunicación de datos.

2.4.2 Wifi

Es una tecnología que permite la conexión de dispositivos electrónicos a una red LAN inalámbrica. Provee al usuario la capacidad de conectarse a internet desde cualquier lugar donde haya un punto de acceso. Con un módulo wifi podría accederse a un servidor o a la nube para compartir los datos captados por el Smartfight con el dispositivo móvil.

La principal desventaja es que requiere de un punto de acceso a internet en el lugar donde se vayan a tomar las medidas. Asimismo fallos del servidor o la nube pueden dar lugar a pérdidas de información.

Por otra parte con esta tecnología es posible acceder a los datos desde un ordenador personal.

2.5 Sistema operativo

Es el software que provee una interfaz entre el resto de programas, los dispositivos hardware y el usuario. Proporciona las rutinas básicas para controlar los dispositivos del equipo y permite administrar y realizar interacción de tareas.

En los Smartphone, existe variedad de sistemas operativos según el fabricante y el modelo del mismo. Una de las decisiones a la hora de diseñar una aplicación móvil es el sistema operativo en el que operará.

Uno de los objetivos finales del Smartfight será que el software de la aplicación móvil pueda ejecutarse en multitud de plataformas y sistemas operativos. No obstante inicialmente se elegirá un sistema operativo destino para las etapas preliminares de desarrollo de entre los siguientes:

2.5.1 Android

Actualmente en posesión y siendo desarrollado por la compañía Google, basado en Linux kernel y diseñado principalmente para dispositivos móviles táctiles como Smartphones y tablets.

Su interfaz con el usuario se basa principalmente en la manipulación directa, empleando gestos como toques en la pantalla, arrastrar y pellizcar. Cuenta con una gran cantidad de aplicaciones diseñadas para funcionar en esta plataforma que pueden encontrarse en su tienda de aplicaciones Google Play.

Algunas de sus características son:

- Navegación web
- Herramientas operadas mediante reconocimiento de voz
- Multitoque
- Multitarea
- Captura de pantalla
- Soporte multilingüe.

Además cuenta con una comunidad activa de desarrolladores que participan en la evolución del sistema creando aplicaciones en código abierto o versiones modificadas del mismo sistema operativo libres de ser modificadas por el usuario.

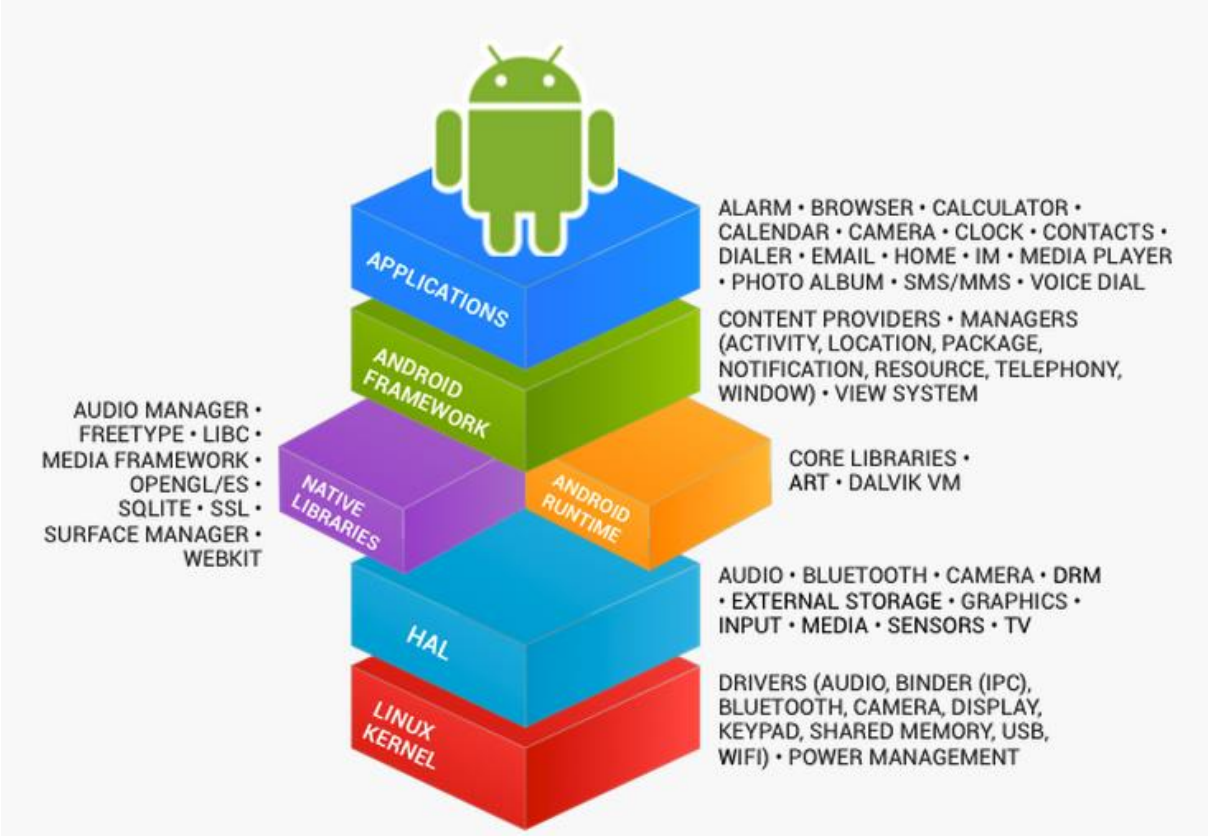


Imagen 13 - Android stack

Por ello existe abundante cantidad de información en la red o libros publicados que permiten inicializarse y ampliar conocimientos sobre el desarrollo de aplicaciones para esta plataforma.

Cuenta en la actualidad con la base de usuarios más elevada de entre todos los sistemas operativos para dispositivos móviles disponibles, siendo especialmente dominante en España según Kantar Worldpanel.

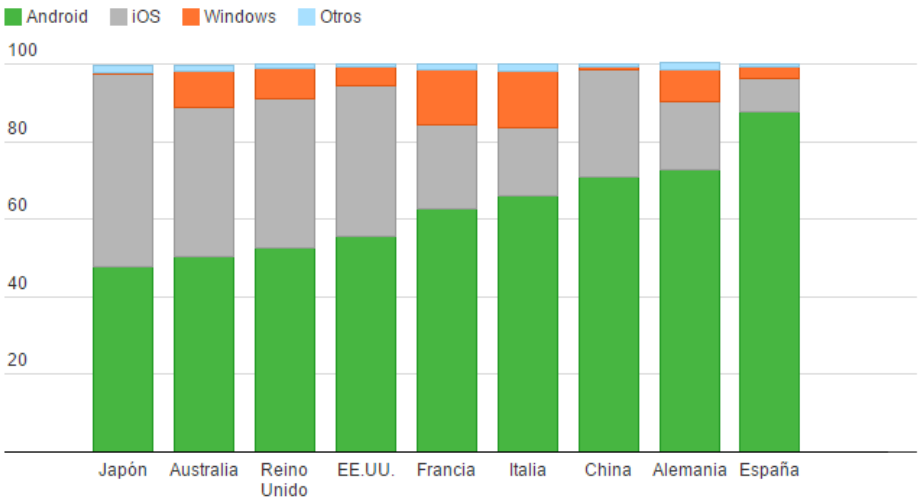


Imagen 14 – Porcentaje de SO empleados por los Smartphones (2014-2015)

Por la mayor base de usuarios y la accesibilidad de esta plataforma, las versiones iniciales de la aplicación del dispositivo móvil que se detallarán en este documento se lanzarán en Android.

2.5.2 iOS

Sistema operativo creado y desarrollado por Apple y de uso exclusivo de los dispositivos Apple. La respuesta a los gestos del usuario para interactuar con el dispositivo es similar a Android. Cuenta con un catálogo de aplicaciones, música, películas y otros medios de entretenimiento al que se accede a través de la Apple Store. Entre las propiedades de este sistema operativo pueden listar:

- Motor de búsqueda integrado
- Reconocimiento de gestos
- Reproductor multimedia
- Acceso directo a Applestore
- Compatibilidad con el servicio Apple Cloud
- Sistema de carpetas
- Multitarea

Se coloca en segunda posición en cuanto a base de usuarios. A diferencia de Android no impulsa el código abierto.

2.5.3 Windows Phone

Sistema operativo desarrollado por Microsoft. Se distingue por su interfaz de usuario que se basa en el tipo de diseño Metro y se organiza en hubs que combinan contenido local y online.

Imagen 15 – Hub música y video Windows phone

Otras de sus características son:

- Navegador web
- Soporte multimedia
- Microsoft Office preinstalado
- Multitarea
- Sincronización entre el dispositivo móvil y el PC personal.

Su base de usuarios es muy reducida en comparación con Android e iOS.

3. Descripción de la solución adoptada

En esta sección se listarán y estructurarán los elementos que formarán parte del diseño final del sistema en dos organigramas, uno para el hardware y otro para el software.

3.1 Hardware

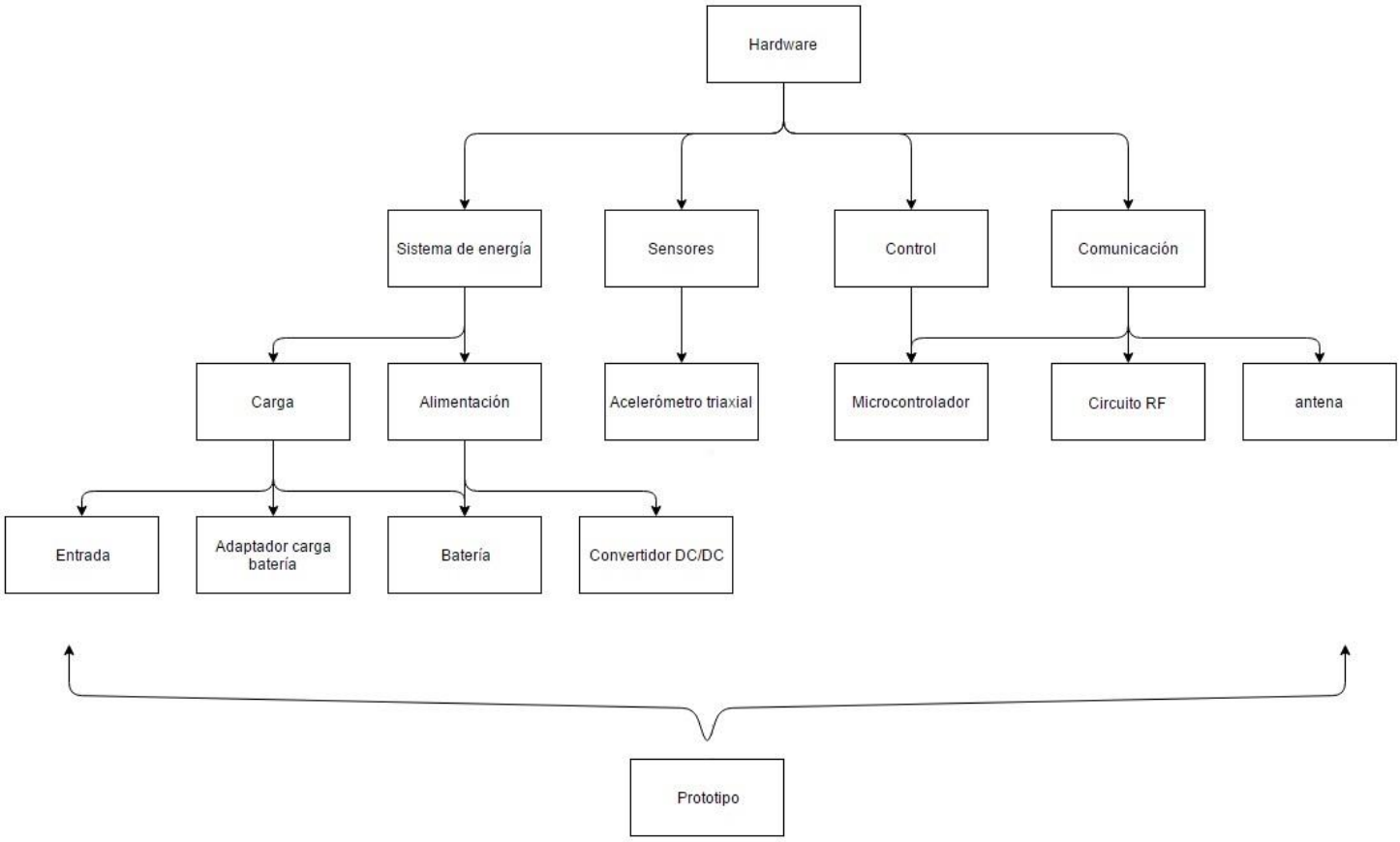


Imagen 16 – Organigrama hardware

La carga del Smartfight se realizará mediante usb, ya sea con cargador o a través de otro dispositivo. Para ello se instalará un puerto micro usb de tipo B (10104110-0001LF) que se conectará a un adaptador de usb/batería (MAX1555) que se encargará de la carga de la batería de polímero de litio (LP381018381119401019). La batería se conectará a un convertidor Buck (TPS2733) que proporcionará tensión eléctrica a un acelerómetro triaxial (ADXL375) y al microcontrolador (CC2541). El acelerómetro comunicará los datos de aceleración al microprocesador. Para la transmisión inalámbrica se empleará una antena (2450AT42A100) cuya conexión al microcontrolador se realizará a

través de un circuito RF encapsulado (2450BM15A0002). Como prototipo se elegirá el CC2541 SensorTag Development Kit de Texas Instruments.

3.2 Software

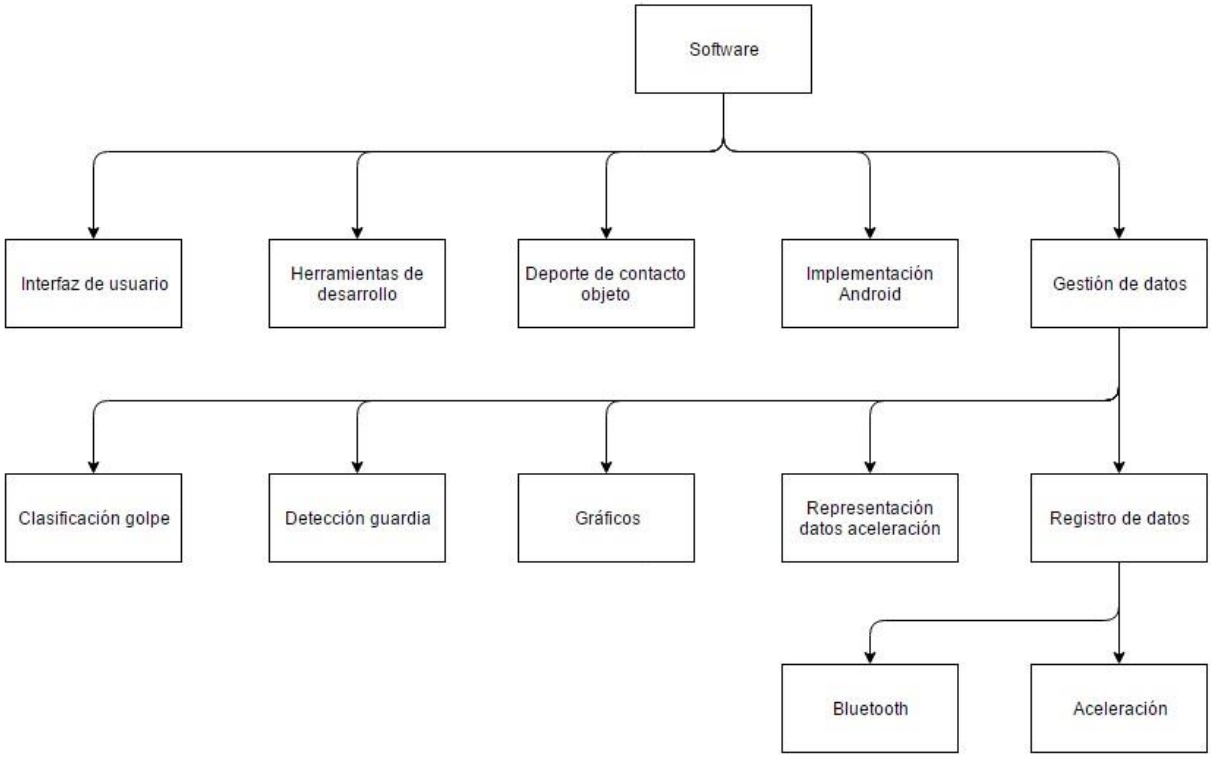


Imagen 17 – Organigrama Software

Se desarrollará una aplicación móvil enfocada a analizar el rendimiento en el boxeo. Como entorno de desarrollo se elegirá el EvothingsStudio. La aplicación diseñada se encargará de emparejar el Bluetooth del microprocesador con el del Smartphone, recibir los datos de aceleración y procesarlos. La información tratada se mostrará en la interfaz de usuario que permitirá conocer:

- Datos de aceleración: se representarán las lecturas del acelerómetro y los máximos alcanzados.
- Gráficos: los datos de aceleración se representarán en gráficos dinámicos y estáticos.
- Detección de guardia: la interpretación de datos determinará la guardia empleada por el usuario de entre las posiciones de boxeo.
- Clasificación de golpe: se compararán los datos con una base de datos para determinar qué tipo de golpe se ha realizado y se contabilizarán.

Finalmente se creará un proyecto compatible con la plataforma Android utilizando el framework Apache Cordova.

4. Justificación detallada de los elementos o componentes de la solución adoptada

En este apartado se justificará y detallará la elección de componentes y las conexiones entre subsistemas del hardware. En cuanto al software se precisará información sobre las herramientas utilizadas en el desarrollo y se especificarán las funciones principales de la aplicación.

4.1 Hardware

El diseño electrónico del Smartfight se explicará siguiendo el siguiente diagrama de bloques:

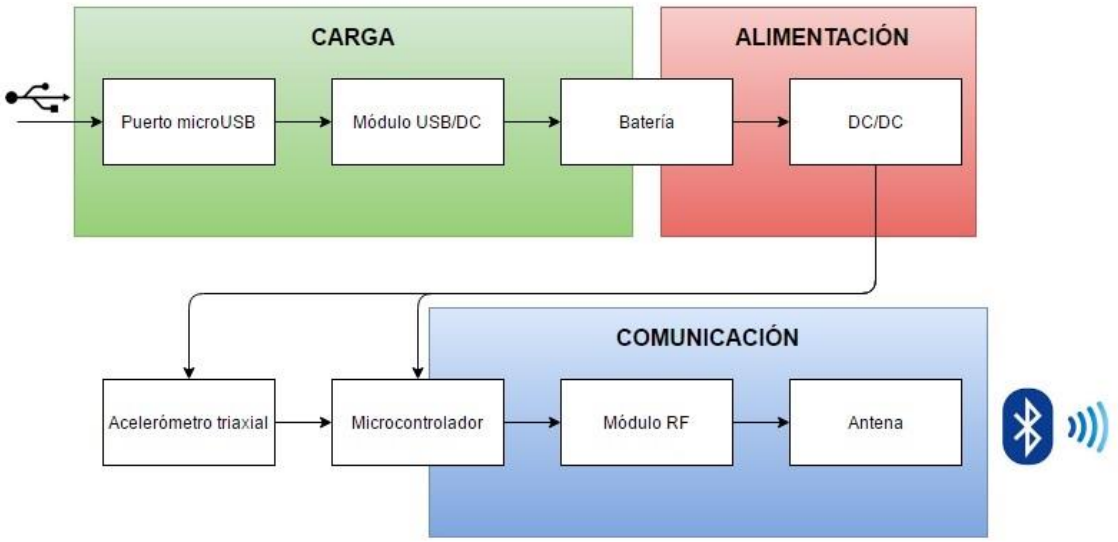


Imagen 18 – Diagrama de bloques del Hardware

4.1.1 Sistema de energía

Este bloqué será responsable de administrar y proporcionar la energía eléctrica del sistema.

4.1.1.1 Carga

Se suministrará la carga de la batería a través de un puerto micro usb de tipo B, que proveerá una tensión eléctrica de 5 voltios y una corriente de 100 miliamperios. Se instalará el conector hembra 1469072 ya que se trata de uno de los modelos más económicos.

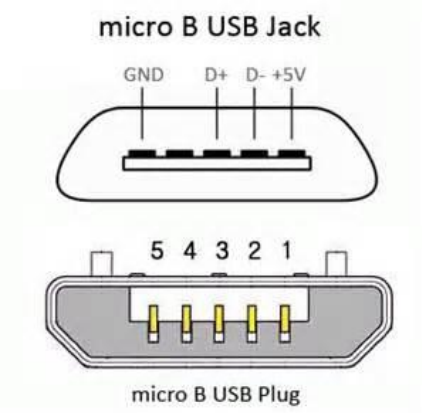


Imagen 19 – Configuración de pines micro USB

La tensión la proporcionará el pin 1 del microusb que se conectará al circuito integrado MAX1555. Este chip gestionará la carga de la batería ofreciendo un ratio de carga óptimo y protección ante cambios de polaridad, además de simplificar el diseño. Alimentará a la batería una tensión típica de 4.2V y una corriente de 90mA.

BAT		MIN	TYP	MAX	
BAT Regulation Voltage	V_{DC} or V_{USB} = 5V	4.158	4.2	4.242	V
DC Charging Current	V_{BAT} = 3.3V, V_{USB} = 0, V_{DC} = 5V	220	280	340	mA
USB Charging Current	V_{BAT} = 3.3V, V_{DC} = 0, V_{USB} = 5V	80	90	100	mA
BAT Prequal Threshold	V_{BAT} rising, 100mV hysteresis	2.9	3	3.1	V
Prequalification Charging Current	V_{BAT} = 2.8V	20	40	80	mA
BAT Leakage Current	V_{DC} = V_{USB} = 0, V_{BAT} = 4.2V			5	µA

Tabla 2 – Datasheet MAX1555 carga batería

Se instalarán condensadores de desacoplo según las especificaciones del fabricante. Al pin CHG se le conectará una resistencia de 100Ω y un LED en serie de 2,2V y 20mA para indicar el estado de carga de la batería cuando el dispositivo esté conectado a una fuente de alimentación.

$$Resistencia\ LED = \frac{Voltaje\ alimentacion * Caída\ voltaje\ LED}{Rango\ de\ Corriente\ LED} = \frac{4{,}2-2{,}2}{0{,}02} = 100\Omega$$

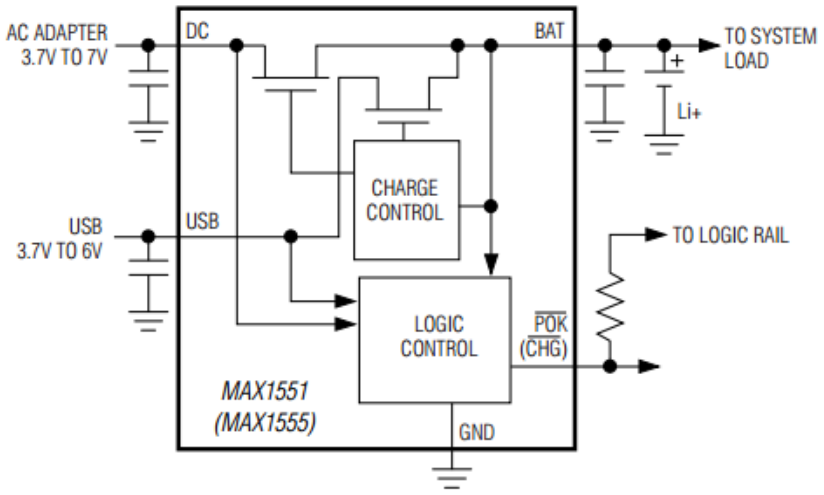


Imagen 20 – Circuito típico de aplicación MAX1555

Se conectará el pin BAT del MAX1555 al componente central del bloque de sistema de energía, la batería. Se trata del modelo de polímero de litio LP381018381119401019. La capacidad de esta batería es de 50 mAh con un tiempo de carga de entre 3 y 4.5 horas y proporciona un voltaje de 3.7V. Sus reducidas dimensiones de 4x11x19mm y su ligero peso de 1,3g son determinantes para la elección de este modelo, ya que permiten reducir el tamaño del diseño del circuito para conseguir un ajuste del dispositivo a la muñeca del usuario más natural.

Para conocer la autonomía del sistema con esta batería primero calcularemos el consumo total:

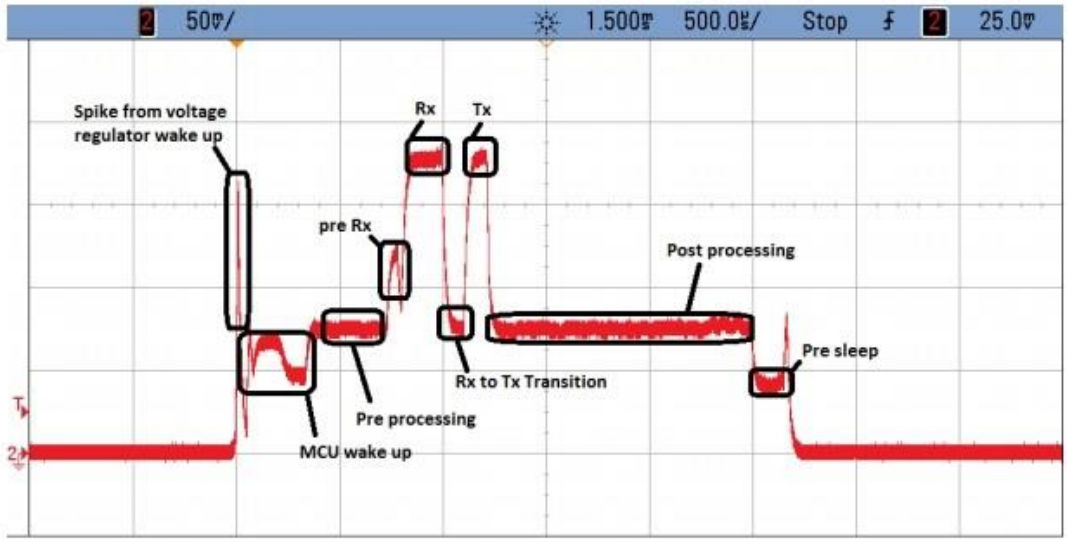
El convertidor Buck TPS62733 requiere 25µA en el modo DC/DC. El acelerómetro ADXL375 configurado en bajo consumo requiere 90µA para una velocidad de transmisión de datos de 400Hz.

Low Power Mode (T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V)

Rate Bits	Output Data Rate (Hz)	Bandwidth (Hz)	I_{DD} (µA)
1100	400	200	90
1011	200	100	60
1010	100	50	50
1001	50	25	45
1000	25	12.5	40
0111	12.5	6.25	35

Tabla 3 – Consumo ADXL375

En cuanto al microcontrolador CC2541, el fabricante proporciona información detallada sobre el consumo del componente (Anexo 1.1 "Consumo IC2 CC2541)), para el cálculo se elegirá el peor de los casos. El consumo durante un evento de conexión puede observarse en el siguiente gráfico:



	Time [µs]	Current [mA]
State 1 (wake-up)	400	6.0
State 2 (pre-processing)	315	7.4
State 3 (pre-Rx)	80	11.0
State 4 (Rx)	275	17.5
State 5 (Rx-to-Tx)	105	7.4
State 6 (Tx)	115	17.5
State 7 (post-processing)	1325	7.4
State 8 (pre-Sleep)	160	4.1

Imagen 21 – consumo CC2541

Los valores de la corriente serán menores en el Smartfight ya que el convertidor BUK TPS62733 permite lograr un mayor ahorro energético según los datos facilitados por el fabricante (Anexo 1.2 "Consumo IC3 TPS62733):

State	Current (mA)	
	DC/DC OFF	DC/DC ON
State 1 (wake-up)	7.1	5.4
State 2 (pre-processing)	8.4	6.9
State 3 (pre-Rx)	11.6	10.7
State 4 (Rx)	18.9	15.4
State 5 (Rx-to-Tx)	9.2	5.8
State 6 (Tx)	18.3	15.1
State 7 (post-processing)	8.1	6.7

Tabla 4 – Ahorro energético en evento de conexión CC2541

Con estos datos puede realizarse el cálculo de la autonomía del sistema.

Primero se calculará el consumo total durante un evento de conexión del microcontrolador.

$$Consumo\ evento\ de\ conexión = \frac{\sum_{i=1}^{n}(State\ i\ Time * State\ i\ Current)}{\sum_{i=1}^{n} State\ i\ Time}$$

$$Consumo\ evento\ de\ conexión = \frac{(5{,}4 * 400) + (6{,}9 * 315) + (10{,}7 * 80) + (15{,}4 * 275) * (5{,}8 * 105) + (15{,}1 * 115) + (6{,}7 * 1325) + (4{,}1 * 160)}{400 + 315 + 80 + 275 + 105 + 115 + 1325 + 160}$$

$$Consumo\ evento\ de\ conexión = 7{,}6769mA$$

El siguiente paso será calcular el consumo en el intervalo de conexión completo, teniendo en cuenta el tiempo que el dispositivo está en modo reposo.

$$Consumo\ intervalo = \frac{(T\ Intervalo - T\ Despierto) * Consumo\ reposo + T\ Despierto\ * Consumo\ evento\ conexión}{T\ intervalo}$$

$$Consumo\ intervalo = \frac{(1000 - 2{,}775) * 0.001 + 2.775 * 7{,}6769}{1000} = 0{,}0223mA$$

$$Consumo\ total = Consumo\ CC2541 + Consumo\ ADXL375 + Consumo\ TPS62733$$

$$Consumo\ total = 0{,}0223 + 0{,}09\ +\ 0{,}025 = 0{,}1373mA$$

Conociendo el consumo, se podrá obtener el tiempo aproximado que la batería podrá alimentar al sistema en funcionamiento continuo.

$$Autonomía = \frac{Capacidad\ batería}{Consumo\ total} = \frac{50}{0{,}1373} = 364{,}17\ horas$$

Por último se multiplicará por un factor de 0,7 que representa pérdidas por factores externos.

$$Autonomía\ estimada = 364{,}17 * 0{,}7 = 254{,}92\ horas$$

Con una autonomía de 254,92 horas puede funcionar 0,7 años si se realiza una sesión de entrenamiento diaria de 1 hora sin necesidad de cargar el dispositivo.

4.1.1.2 Alimentación

Se conectará la batería al circuito integrado TPS62733 que proporcionará la tensión eléctrica al resto de componentes del sistema. El TPS62733 es un convertidor Buck de alta frecuencia optimizado para aplicaciones inalámbricas de bajo consumo. Reduce el consumo de batería en el modo TX y RX con su eficiente conversión de voltaje.

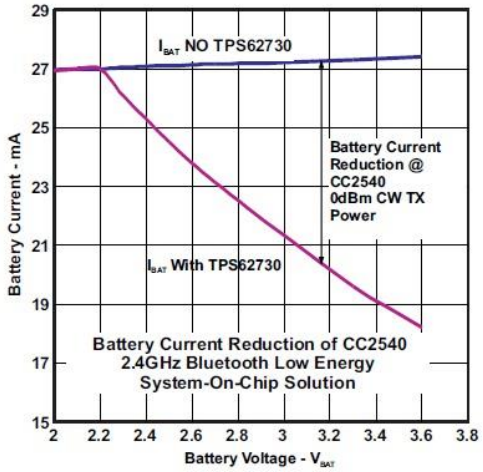


Imagen 22 – Comparación consumo CC2540 con y sin TPS62730

Provee de una corriente de hasta 100mA y con una tensión de entrada de entre 1,9V y 3,9V el dispositivo es compatible con baterías basadas en el Litio. Además puede configurarse en un modo de consumo ultra bajo rondando los 30nA cuando el pin ON/BYP se conecta a nivel lógico bajo. Esto ofrece soporte a microcontroladores con modo reposo, en este modo se puentea la circuitería interna y la salida del conversor se conecta a la batería.

En modo DC/DC, que será el usado principalmente en el sistema diseñado, proporciona una tensión de salida fija de 2,3V y tiene un consumo de 25µA.

El encapsulado se integrará al circuito general usando la configuración recomendada por el fabricante.

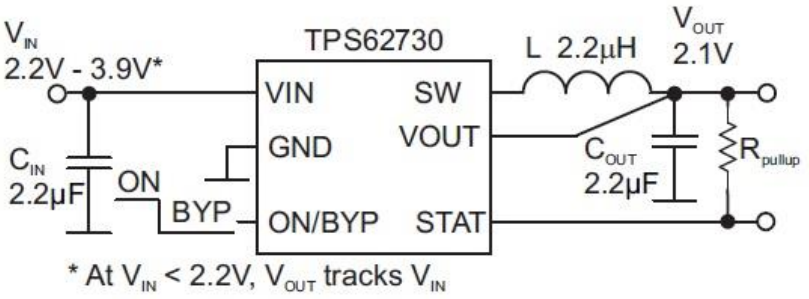


Imagen 23 – Circuito típico de aplicación TPS6273x

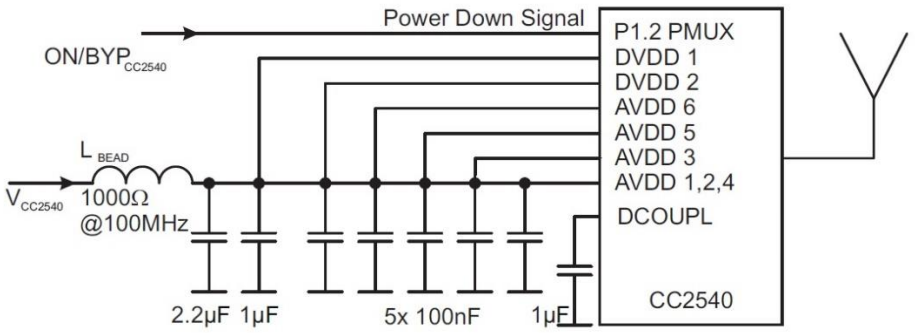


Imagen 24 – Condensadores de desacoplo TPS6273x a CC254x

4.1.2 Sensores

Se usará el acelerómetro triaxial ADXL375 para capturar los datos que permitirán evaluar los golpes. Para determinar el rango de medida necesario se consultó un estudio publicado en el British journal of sports medicine que analiza las fuerzas resultantes en la cabeza tras recibir un golpe directo de boxeo. (Anexo 3.1 "Biomechanics of the head for Olympic bóxer punches to the face").

Las medidas se toman tras lanzar un straight right a la mandíbula de un maniquí con diferentes sensores instalados.

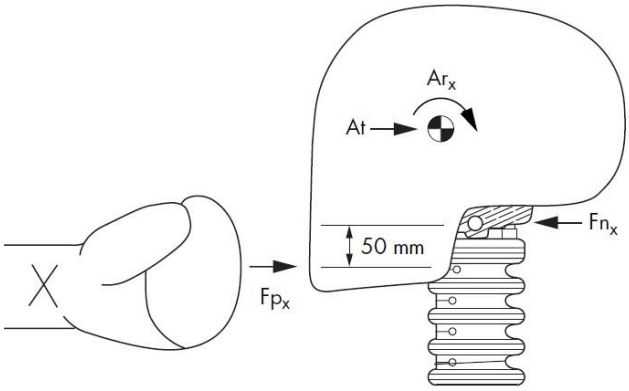


Imagen 25 – Toma de datos del estudio

La fuerza que se tomará como referencia para la elección del rango adecuado será la aceleración lineal (At) de la cabeza. La aceleración lineal de la mano será similar a la observada en la cabeza cuyos valores se obtendrán de la tabla de resultados que el mismo estudio proporciona.

Las conclusiones son que el valor medio del pico de aceleración es de 58g, el máximo registrado es de 78g y el mínimo es de 33g. Los datos se contrastaron con otro estudio realizado por la universidad estatal de Wayne cuyo objetivo es comprobar la eficacia de los protectores de cabeza en el boxeo para evitar lesiones (Anexo 3.2 "Efectiveness of boxing headgear for limiting injury"). Los

datos de aceleración lineal de la cabeza obtenidos en este estudio son que el máximo pico medido es de 78.04g sin protección y 51.73g con protección.

Por ello para que el Smartfight pueda ser utilizado a nivel profesional, el acelerómetro debe tener un rango de medida de al menos ±80g. El ADXL375 cumple esta especificación con un rango de ±200g.

Class	ID	Wgt, lb	Wrist pos.	Eff. hand mass, kg	Linear accel, *g*	Dur., ms
Flyweight						
	us018-1	112	Rigid	2.1	67	13
	us018-2	112	Rigid	3.0	57	13
	us018-3	112	Rigid	4.4	68	13
	us034-2	112	Flexed	1.3	49	11
	us034-3	112	Flexed	2.0	40	12
	us036-1	112	Flexed	1.4	58	12
	us036-2	112	Rigid	1.9	65	11
	Avg.			2.31	58	12.1
	SD			1.06	10	0.9
Light welterweight						
	us019-1	139	Rigid	2.1	37	11
	us019-2	139	Rigid	2.1	50	11
	us019-3	139	Rigid	3.9	59	12
	Avg.			2.70	49	11.3
	SD			1.04	11	0.6
Middleweight						
	us033-1	165	Flexed	0.6	33	10
	us033-2	165	Flexed	1.0	47	11
	us033-3	165	Flexed	0.9	51	9
	Avg.			0.81	44	10.0
	SD			0.19	9	1.0
Super heavyweight						
	us020-2	201	Flexed	2.7	70	11
	us020-3	201	Flexed	8.2	71	10
	us035-1	240	Rigid	4.3	68	10
	us035-2	240	Rigid	2.9	78	11
	us035-3	240	Rigid	6.8	67	15
	Avg.			4.97	71	11.4
	SD			2.44	4	2.1
Significance with respect to weight				0.042	0.074	0.513
All avg.				2.86	58	11.4

Tabla 5 – Medidas estudio

El ADXL375 es un acelerómetro triaxial digital que realiza la conversión analógica/digital internamente y soporta comunicación I²C y SPI.

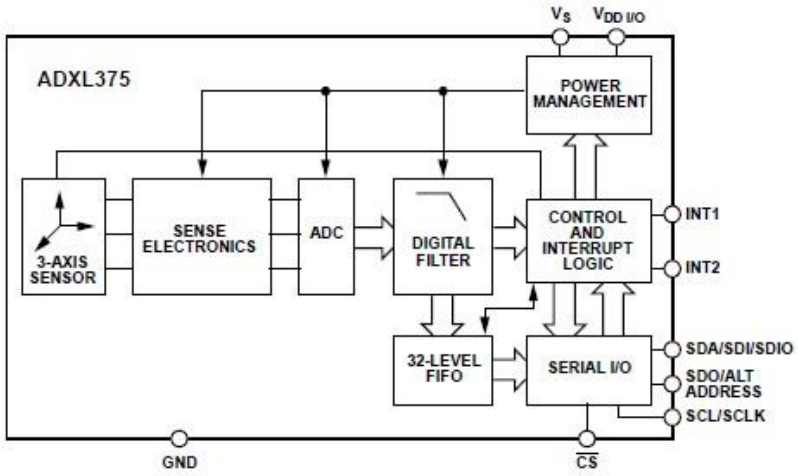


Imagen 26 – Diagrama de bloques ADXL375

Permite un captar en detalle las curvas de aceleración ya que puede configurarse la frecuencia de muestreo en un rango de [0,1Hz - 3,2kHz]. Su consumo varía entre los 35µA y los 145µA según la frecuencia de muestro seleccionada y 0,1 µA en modo reposo.

Tiene un sistema de gestión de memoria integrado que permite guardar datos reduciendo así la actividad del procesador al que se conecte. Y con su reducido tamaño de 3mm x 5mm x 1mm cumple las características requeridas para ser integrado en un dispositivo ponible.

Además cuenta con funciones propias para la detección de picos, que son comunicados al microcontrolador por los pines de interrupción del ADXL375, y cuyo umbral puede configurarse a través de registros.

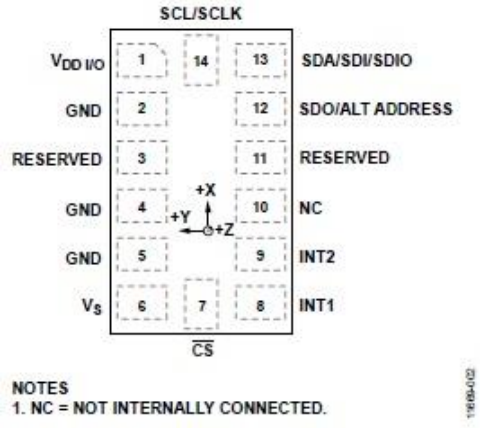


Imagen 27 – Pines ADXL375

Se conectaran condensadores entre VDD y masa y Vs y masa que reducirán ruidos como recomienda el fabricante.

El acceso a los datos del ADXL375 y la configuración del mismo se realizará a través de registros.

Address		Register Name	Access Type	Reset Value	Description
Hex	Decimal				
0x00	0	DEVID	R	11100101	Device ID
0x01 to 0x1C	1 to 28	Reserved	N/A	N/A	Reserved; do not access
0x1D	29	THRESH_SHOCK	R/$\overline{W}$	00000000	Shock 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	Shock duration
0x22	34	Latent	R/$\overline{W}$	00000000	Shock latency
0x23	35	Window	R/$\overline{W}$	00000000	Shock 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
0x2A	42	SHOCK_AXES	R/$\overline{W}$	00000000	Axis control for single shock/double shock
0x2B	43	ACT_SHOCK_STATUS	R	00000000	Source of single shock/double shock
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	Interrupt source
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

Tabla 6 – Mapa de registros ADXL375

4.1.3 Control

La mayor carga del proceso de datos la realizará el dispositivo móvil al que se vincule el Smartfight, no obstante será necesaria la integración de un componente de control cuya función será recibir los datos de aceleración y preparar la comunicación inalámbrica.

Se elegirá un microcontrolador que cumpla con los requerimientos mínimos del Smartfight: número de puertos de entrada y salida de datos ajustados a las necesidades del sistema, tamaño reducido, bajo consumo, comunicación I^2C o SPI, soporte Bluetooth. Un microcontrolador que se adapta a estas necesidades es el CC2541 del fabricante Texas Instruments. Se trata de una solución de sistema en chip para configuraciones Bluetooth de bajo consumo que combina un transceptor RF con un microprocesador 8051 MCU.

Entre sus características destacables:

- Bluetooth de bajo consumo 2.4GHz y su sistema en chip RF integrado.
- Potencia de salida programable hasta 0dBm.
- Pocos componentes externos requeridos.
- Tamaño de 6x6x1mm.
- Bajo consumo.
- Amplio rango de suministro de voltaje (2V – 3.6V).
- Microcontrolador de alto rendimiento y bajo consumo 8051.
- Memoria Flash programable (128 o 256KB).
- 8KB de RAM.
- Se conserva la configuración de registros relevantes independientemente del modo de consumo.
- 2 USARTs potentes que soportan varios protocolos de comunicación serie.
- 23 pines de Entrada/Salida.

El fabricante proporciona información detallada sobre los componentes externos requeridos para el correcto funcionamiento del CC2541, que se incluirán al integrar el microcontrolador al circuito general del sistema.

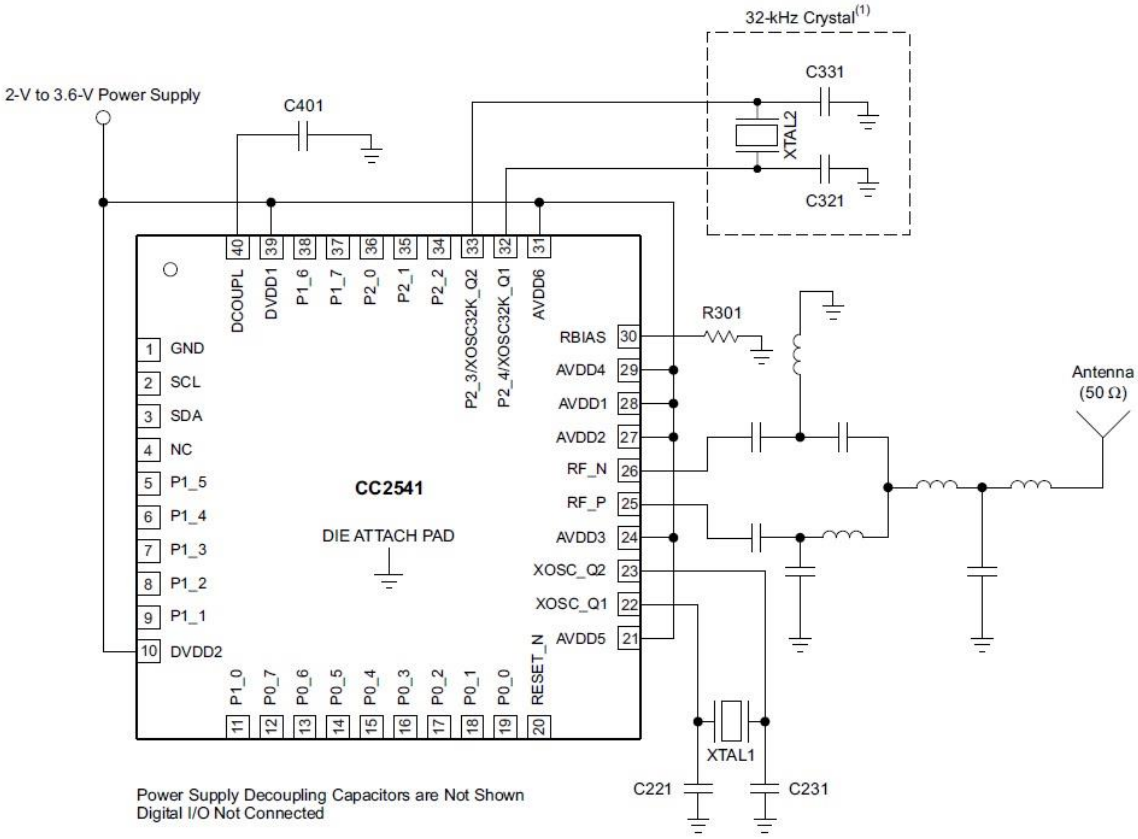


Imagen 28 – Componentes externos CC2541

Al puerto 1.0 de Entrada y salida digital se conectará un pulsador cuya función será de conectar el sistema cuando se pulse estando apagado. O resetear los valores y apagar el sistema si se pulsa en marcha.

4.1.4 Comunicación

Para que pueda realizarse la comunicación Bluetooth adecuadamente, debe conectarse una antena al microcontrolador a través de un circuito RF como se indica en la imagen 35. En lugar de diseñar un circuito con componentes discretos se optará por el encapsulado 2450BM15A0002 de Johanson technology, especialmente diseñado para la familia de microcontroladores CC254x entre otros.

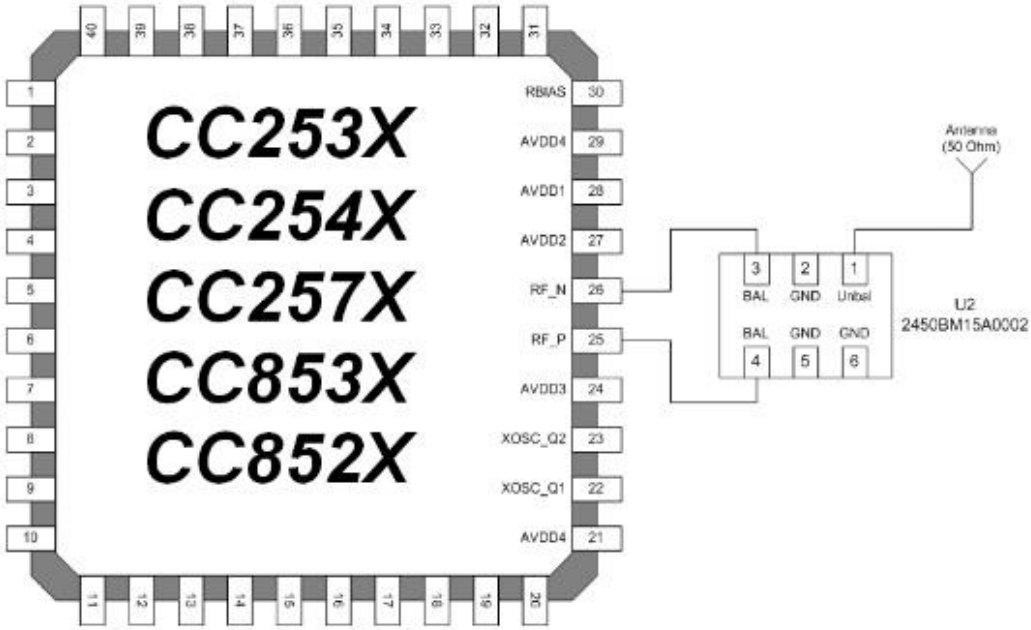


Imagen 29 – Conexión 2450BM15A0002

Con unas dimensiones de 2x1.25x0.7mm, simplifica el diseño del circuito y ocupa menos espacio que la solución alternativa con componentes discretos. Además con la incorporación del 2450BM15A0002 se consiguen mejoras en el segundo y tercer armónico.

		Datasheet CC2530EM discrete **10** passive components *typ*	Measured CC2530EM **w/2450BM15A0002** **1** passive component *typ*	unit
Receiver sensitivity	PER =1% as specified by [1] [1] requires -85dBm	-97	-96.6	dBm dBm
Ouput Power (0xF5)	Delivered to a single ended 50Ω load through a balun using max recomended output setting (0xF5) [1] requires minimum -3dBm	50 4.5	50 3.3	Ω dBm dBm
Spurius Emission	25MHz-1000MHz (outside restricted bands) 25MHz-2400MHz (within FCC restricted bands) 25MHz-1000MHz (within ETSI restricted bands) 1800-1900MHz (ETSI restricted band) 5150-5300MHz (ETSI restricted band) At 2xfc and 3xfc (FCC restricted band) At 2xfc and 3xfc (ETSI EN 300-440 and EN300-328) 1GHz-12.75GHz (Outside restricted bands) At 2483.5MHz and above (FCC restricted bands) fc=2480MHz	-60 -60 -60 -57 -55 -42 -31 -53 -42	-64 -64 -64 -64 -55 -47.3 -38.2 -57.4 -51.6	dBm dBm dBm dBm dBm dBm dBm dBm dBm
EVM	Measured as defined by [1] using maximum recommended output power setting [1] Requires maximum 35%	2	2	%
Current Consumption	32MHz XOSC running, radio in RX mode, -50dBm input power No peripherials active, CPU idle	20.5	21.3	mA
	32MHz XOSC running, radio in RX mode, -100dBm input power No peripherials active, CPU idle	24.3	24.6	mA
	32MHz XOSC running, radio in Tx mode, output power 0xD5 No peripherials active, CPU idle	28.7	29.1	mA
	32MHz XOSC running, radio in Tx mode, output power 0xF5 No peripherials active, CPU idle	33.5	33.3	mA

Tabla 7 – Comparación circuito RF con componentes discretos o 2450BM15A0002

La antena seleccionada será el modelo 2450AT42A100 de Johanson technology. Tiene un tamaño de 2x5x1.1mm y un rango de frecuencias entre 2.4GHz y 2.5GHz, una ganancia de 0dBi y pérdida de retorno de -9.5 dB. Es la antena utilizada en la placa diseñada por Texas Instruments con todos los componentes externos necesarios para el microcontrolador CC2541 (CC2541 Postage Stamp Reference Design).

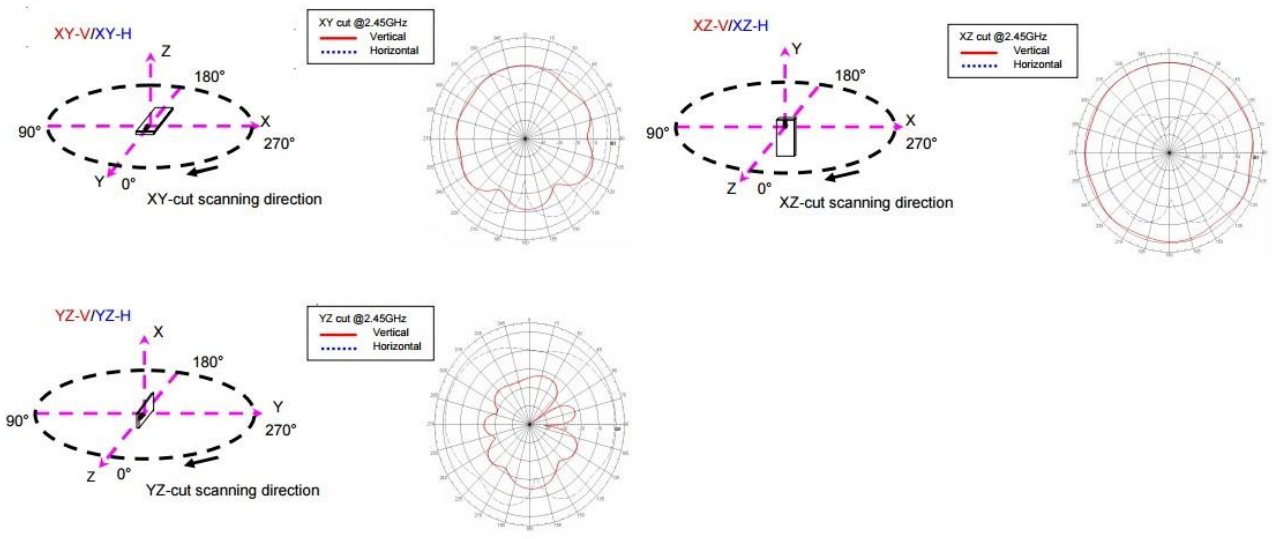


Imagen 30 – Radiación típica antena 2450AT42A100

4.1.5 Esquema del circuito y PCB

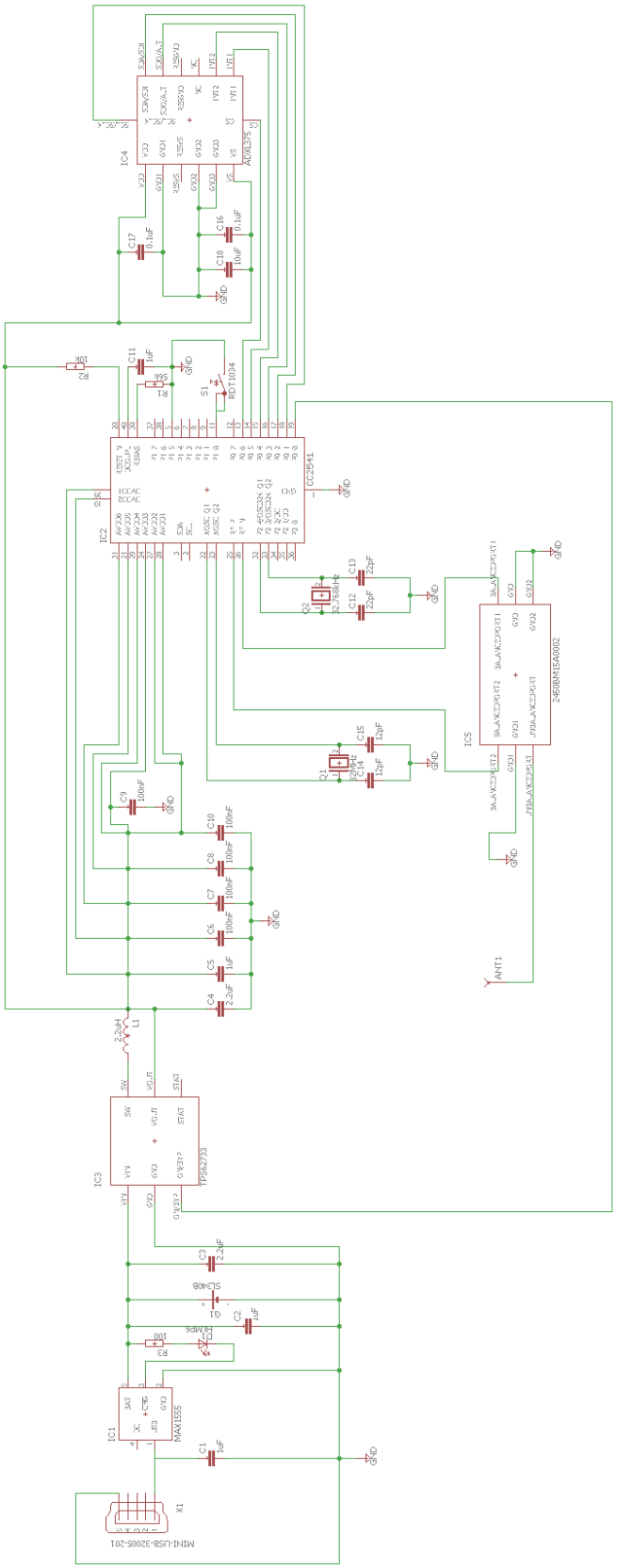


Imagen 31 – Esquema eléctrico Smartfight

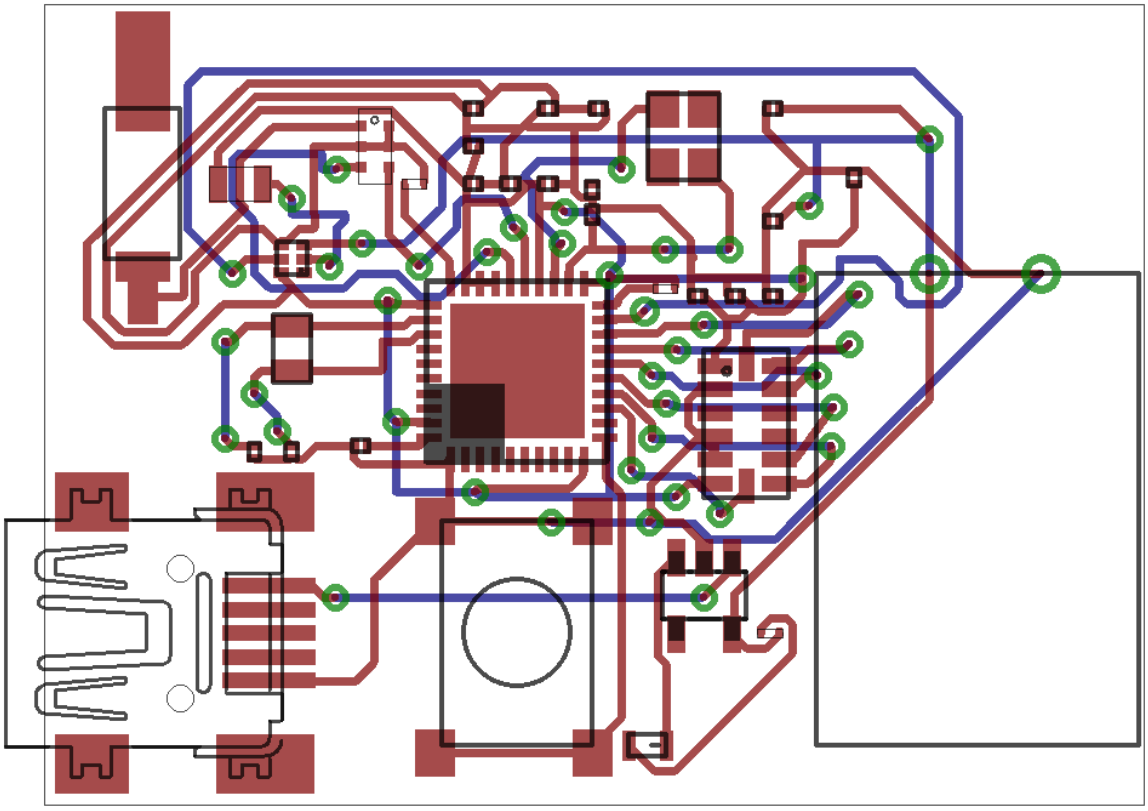


Imagen 32 – PCB Smartfight 38x28mm

4.1.6 Prototipo

Para las pruebas iniciales y primeras aproximaciones a la aplicación móvil se empleará el kit de desarrollo de Texas Instruments CC2541 SensorTag. Se trata de un dispositivo diseñado para el desarrollo de aplicaciones centradas en la adquisición y proceso de datos captados por sensores y con comunicación Bluetooth.

Comparte con el Smartfight como núcleo del dispositivo el microprocesador CC2541 e incorpora los siguientes sensores:

- Sensor de temperatura por infrarrojos Texas Instruments TMP006
- Sensor de humedad Sensirion SHT21
- Giroscopio Invensense IMU-3000
- Acelerómetro Kionix KXTJ9
- Magnetómetro Freescale MAG3110
- Barómetro Epcos T5400
- Temperatura interna (incluido en el CC2541)

El dispositivo se alimenta mediante una pila de botón. Pueden encontrarse diversas aplicaciones en Play Store diseñadas en torno a este dispositivo, entre

ellas una que proporciona el propio fabricante que permite visualizar datos de los sensores y su representación gráfica.

Se elegirá el Sensortag CC2541 ya que en cuanto a funcionalidad es similar al Smartfight. Es un producto comercializado accesible y económico. Cuenta con una cubierta de material plástico protegiendo la placa del circuito que le otorga robustez. Por ello resulta cómodo y seguro realizar pruebas de impacto con este dispositivo.

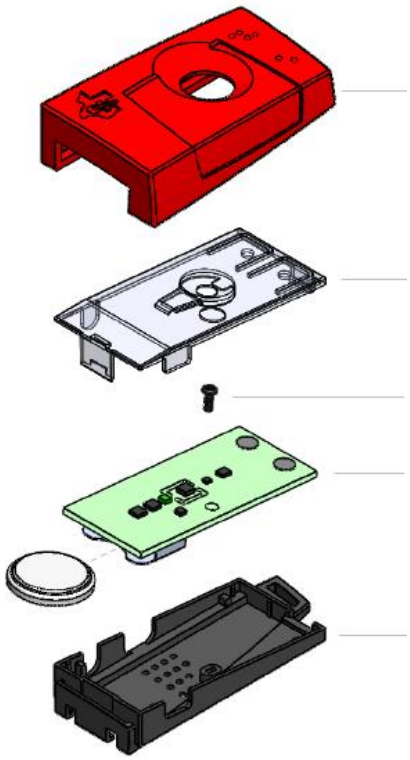


Imagen 33 – Piezas TI CC2541 SensorTag

Como punto negativo el acelerómetro que integra no está diseñado para captar los valores extremos que se manifiestan durante un golpe. Con un rango máximo de ±8g apenas cubre el 10% de las exigencias del sistema. Y su frecuencia de muestreo está limitada a 10Hz, demasiado bajo para aplicaciones a altas velocidades como son los golpes en los deportes de contacto.

No obstante es suficiente para explorar distintas posibilidades de desarrollo de la aplicación móvil. El nivel de exigencia puede reducirse controlando la potencia de los golpes durante las pruebas, con lo que pueden crearse aplicaciones que podrán trasladarse a las especificaciones del Smartfight con ligeras modificaciones.

4.2 Software

En este apartado se explicarán las preparaciones previas al desarrollo de la aplicación móvil así como el código empleado para sus funciones principales y la interacción con el usuario.

4.2.1 Deporte de contacto objeto

Las versiones iniciales de la aplicación móvil se centrarán en el boxeo como deporte de contacto soportado. Se elegirá esta modalidad deportiva porque su reglamento prohíbe emplear el tren inferior para golpear al oponente, con lo que todos los golpes pueden ser registrados colocando el Smartfight en la muñeca del usuario.

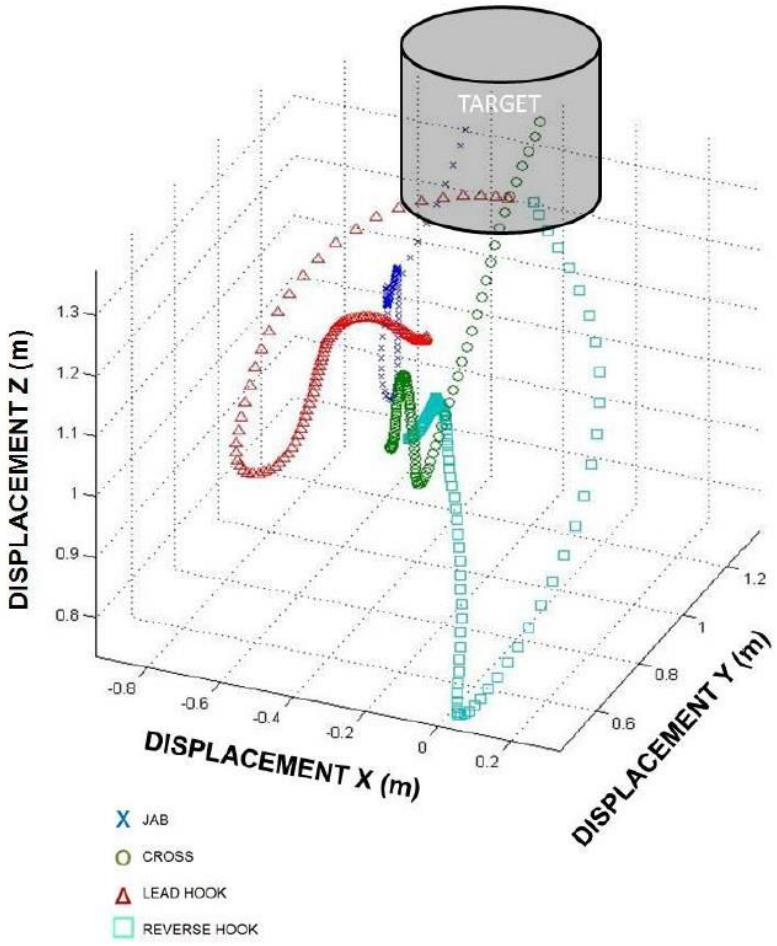


Imagen 34 – Desplazamiento típico de la mano en diferentes golpes de boxeo

Además la mayor diferencia entre las diferentes posiciones defensivas reside en la colocación de los brazos por lo que será posible identificar las guardias del usuario. La aplicación se desarrollará enfocándose en entrenamientos de shadow boxing.

4.2.2 Herramientas de desarrollo

Con el fin de facilitar la implementación de la aplicación móvil y reducir notablemente el tiempo de desarrollo se ha trabajado con el Evothings Studio. Se trata de una aplicación móvil para el Internet of Things que permite sincronizar el ordenador con el Smartphone. Puede trabajarse con la aplicación abierta en el móvil mientras se realizan modificaciones en el código desde el ordenador y visualizar los cambios instantáneamente. Gracias a esta característica se agiliza el proceso ya que no es necesario cargar la aplicación de nuevo al móvil o a un emulador para realizar comprobaciones.

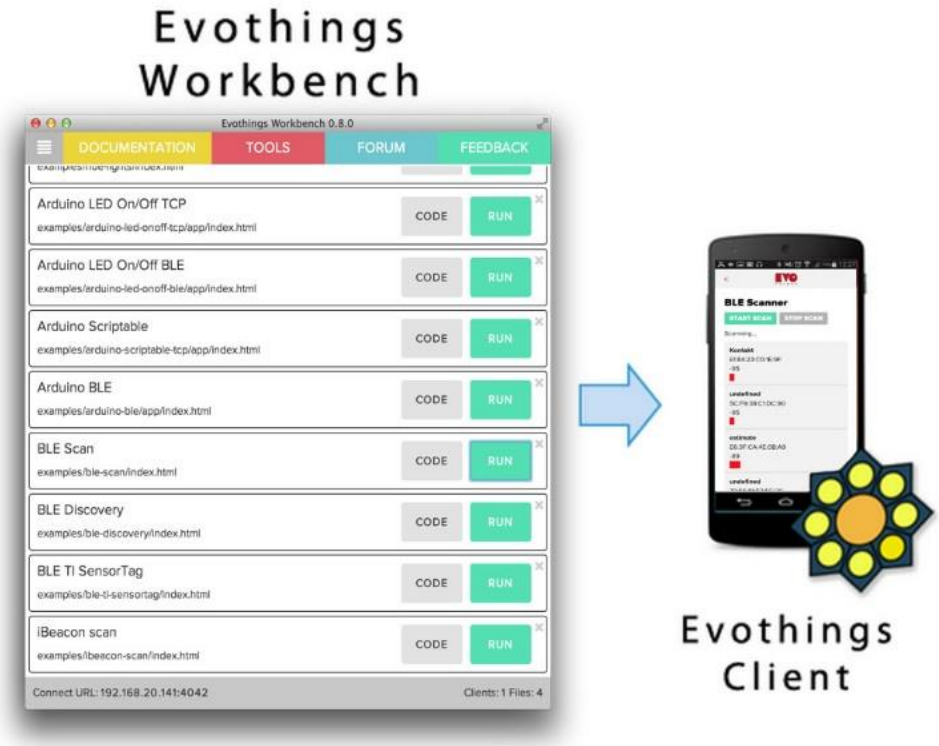


Imagen 35 – Evothings Studio en ordenador y móvil

Soporta código en HTML5, CSS y Javascript y proporciona numerosas librerías y ejemplos además de compatibilidad con Android e iOS.

Para poder utilizar esta herramienta solo se necesita:

- Conocimientos básicos de HTML y Javascript.
- El programa para el ordenador Evothings Studio.
- La aplicación móvil de Evothings.
- Un editor de texto.
- Conexión a internet en el ordenador y el Smarthpone.

Y los pasos a seguir para crear un proyecto son los siguientes:

- Descargar y ejecutar Evothings Studio al ordenador.
- Descargar y ejecutar la aplicación móvil de Evothings.
- Crear una carpeta y añadir los archivos del proyecto (al menos debe contener un fichero HTML).
- Arrastrar el archivo HTML principal a la ventana del programa Evothings.
- Vincular la aplicación móvil al ordenador apretando en el botón SCAN FOR WORKBENCH de la aplicación móvil o escribiendo directamente la dirección en el cuadro que hay debajo.
- Clicar el botón RUN en el proyecto que se desea ejecutar en la interfaz del ordenador.

Listo, si se realizan modificaciones en el código mientras están vinculados se trasladarán al Smartphone si se pulsa RUN de nuevo.

4.2.3 Gestión de datos

Se describirá el proceso de recolección de los datos enviados por el Smartfight mediante Bluetooth y como se trata la información obtenida para mostrar al usuario el rendimiento en su sesión de entrenamiento.

4.2.3.1 Registro de datos

Para recibir los datos de aceleración y gestionar la comunicación Bluetooth se usarán las librerías easy-ble y ti-sensortag. Cuando se inicia la aplicación, Se llama a la función InitialiseSensortag.

```
function initialiseSensortag()
{
sensorTag
        .statusCallback(statusHandler)
        .errorCallback(errorHandler)
        .accelerometerCallback(accelerometerHandler, 100)
        .connectToClosestDevice()
}
```

Esta función crea un objeto SensorTag que se usará para habilitar el sensor. accelerometerCallback actualizará los datos del acelerómetro con el periodo que se seleccione en milisegundos, 100 en eeste caso.

connectToClosestDevice hará que la aplicación escanee y se conecte al SensorTag con la RSSI más elevada.

De manejar los datos de aceleración se encargará la función accelerometerHandler. Que convierte los datos recibidos a fuerzas g y los guarda en un vector que utiliza para realizar llamadas a otras funciones que se explicarán en siguientes apartados.

```
var AccelValueXSensorTag = 0
var AccelValueYSensorTag = 0
var AccelValueZSensorTag = 0
function accelerometerHandler(data)
{
var x = data[0] / 16
var y = data[1] / 16
var z = data[2] / 16
var g = Math.sqrt((x*x)+(y*y)+(z*z))
AccelValueXSensorTag = x
AccelValueYSensorTag = y
AccelValueZSensorTag = z
var AccelArray = [x,y,z]
displayValue(
        'AccelerometerData',  'x = ' + x + '<br/>y = ' + y + '<br/>z = ' + z + '<br/>g = ' + g)

MaxAccelerations(x,y,z)
PunchDataToggle(AccelArray,InitialStance)
datacollectx (AccelArray,InitialStanceCheck)
datacollecty (AccelArray,InitialStanceCheck)
datacollectz (AccelArray,InitialStanceCheck)
}
```

La función displayValue crea un atributo ID y le asigna un valor.

```
function displayValue(elementId, value){
        document.getElementById(elementId).innerHTML = value
}
```

4.2.3.2 Representación datos de aceleración

La función MaxAccelerations se encargará de registrar los valores máximos y mínimos de aceleración alcanzados.

```
var MaxAccelValueXSensorTag = 0
var MaxAccelValueYSensorTag = 0
var MaxAccelValueZSensorTag = 0
var MinAccelValueXSensorTag = 0
var MinAccelValueYSensorTag = 0
var MinAccelValueZSensorTag = 0
function MaxAccelerations(x,y,z){
if (MaxAccelValueXSensorTag < x){
        MaxAccelValueXSensorTag = x;
}
if (MaxAccelValueYSensorTag < y){
        MaxAccelValueYSensorTag = y;
}
if (MaxAccelValueZSensorTag < z){
        MaxAccelValueZSensorTag = z;
}
if (MinAccelValueXSensorTag > x){
        MinAccelValueXSensorTag = x;
}
if (MinAccelValueYSensorTag > y){
        MinAccelValueYSensorTag = y;
}
if (MinAccelValueZSensorTag > z){
        MinAccelValueZSensorTag = z;
}
displayValue('MaxXAccelerometerData', 'Maximo X = ' + MaxAccelValueXSensorTag +
'<br/>Minimo X = ' + MinAccelValueXSensorTag)
displayValue('MaxYAccelerometerData', 'Maximo Y = ' + MaxAccelValueYSensorTag +
'<br/>Minimo Y = ' + MinAccelValueYSensorTag)
displayValue('MaxZAccelerometerData', 'Maximo Z = ' + MaxAccelValueZSensorTag +
'<br/>Minimo Z = ' + MinAccelValueZSensorTag)
}
```

Las funciones datacollectx, datacollecty y datacollectz almacenarán los datos de aceleración en vectores. Las tres funciones cumplen la misma función cada una para cada eje de aceleración, aunque en datacollectx se añade código para reiniciar los vectores.

```
function datacollectx(accelarray,check){
if (check == 0){
        if (DataHold == 1){
        //En estas condiciones acaba de lanzarse un nuevo golpe
        arrayX = [0]
        arrayY = [0]
        arrayZ = [0]
        DataHold = 0
        }
        arrayX.push(accelarray[0])
if (arrayX.length > 25)
arrayX = [0]
        }
}
```

check y DataHold son variables que se definen en otra función que se detallará más adelante. Cuando se detecta que el Smartfight está en la posición inicial desde la que parte el puñetazo, el valor de check y DataHold es 1. Cuando el Smartfight no está en la posición inicial, check es 0. Asi pues está función reinicia los datos del vector de aceleración cada vez que se realiza un nuevo golpe y registra uno nuevo con .push().

4.2.3.3 Gráficos

Se representarán 6 gráficas en la aplicación:

3 gráficas dinámicas donde se mostrarán los valores de aceleración de los ejes x, y, z que se actualizarán automáticamente.

3 gráficas estáticas donde se visualizarán los vectores de aceleración de los ejes x, y, z que capturarán las aceleraciones desencadenadas por el golpe del usuario. Para ello se incluirá la librería Highcharts que permite la visualización de datos en HTML de forma gratuita para aplicaciones no comerciales.

El código para la implementación de la gráfica dinámica es el siguiente:

```
$(function () {
    $(document).ready(function () {
        Highcharts.setOptions({
            global: {
                useUTC: false
            }
        });

        $('#containerx').highcharts({
            chart: {
                type: 'line',
                animation: Highcharts.svg, // don't animate in old IE
                marginRight: 10,
                events: {
                    load: function () {

                        // set up the updating of the chart each second
                        var series = this.series[0];
                        setInterval(function () {
                            var x = (new Date()).getTime(), // current time
                                y = AccelValueXSensorTag ;//AQUI EL VALOR DE LA ACELERACIÓN
                            series.addPoint([x, y], true, true);
                        }, 400); // tiempo de muestreo
                    }
                }
            },
            title: {
                text: 'Eje X'
            },
            xAxis: {
                type: 'datetime',
                tickPixelInterval: 150
            },
```

```
yAxis: {
    title: {
        text: 'Aceleracion (g)'
    },
    plotLines: [{
        value: 0,
        width: 1,
        color: '#808080'
    }]
},
tooltip: {
    formatter: function () {
        return '<b>' + this.series.name + '</b><br/>' +
            Highcharts.dateFormat('%Y-%m-%d %H:%M:%S', this.x) + '<br/>' +
            Highcharts.numberFormat(this.y, 2);
    }
},
legend: {
    enabled: false
},
                plotOptions: {
    line: {
        marker: {
            enabled: false
        }
    }
},
exporting: {
    enabled: false
},
series: [{
    name: 'x',
    data: (function () {
```

```
                // generate an array of random data
                var data = [],
                    time = (new Date()).getTime(),
                    i;

                for (i = -19; i <= 0; i += 1) {
                    data.push({
                        x: time + i * 400,
                        y: 0 // Asi se inicializa a 0
                    });
                }
                return data;
            }())
        }]
    });
});
```

Y para representar la gráfica estática:

```
function RefreshX(){
$(function () {
    $('#staticx').highcharts({
        chart: {
            type: 'line'
        },
        title: {
            text: 'Eje X'
        },
        yAxis: {
            title: {
                text: 'Aceleracion (g)'
            }
        },
```

```
            exporting: {
            enabled: false
        },
        series: [{
          name: 'X',
          data: arrayX
        }]
    });
});
}
```

El mismo código se empleará para los otros ejes con pequeñas variaciones al elegir variables o nombrar elementos. Todas las gráficas estarán vinculadas a botones que las mostrarán u ocultarán cuando se pulsen.

4.2.3.4 Detección de guardia

La función GetStance se encargará de determinar la guardia del usuario e indicará si es una de las posiciones defensivas típicas del boxeo. Esta aplicación cuenta con los datos necesarios para determinar si se está utilizando una de las siguientes tres posiciones:

- Guardia clásica: la más utilizada, el hombro encarando al oponente rodillas ligeramente flexionadas y espalda recta, pie frontal dirección al oponente y el trasero a unos 45º, los codos pegados a los costados, cabeza gacha, las manos protegiendo las mejillas y la mandíbula con las palmas hacia dentro.
- Peak-a-Boo: Similar a la clásica pero manteniendo un perfil más bajo y con los brazos por delante cubriendo el torso y la cara.
- Philly Shell: El brazo izquierdo bajo cubriendo el cuerpo, el hombro izquierdo elevado y el brazo derecho protegiendo la mandíbula.

Podrá detectarse la guardia utilizada ya que el sensor triaxial permite conocer la posición del dispositivo gracias a la fuerza de la gravedad.

Imagen 36 – Guardias boxeo (Clásica, Peak-a-Boo, Philly Shell)

```
var InitialStance = [0,0,0]

function GetStance(x,y,z)

{

var Classic = [-0.8125,-0.3125,0.5]

var Peakaboo = [-0.9375,-0.1875,0]

var Phillyshell = [-0.4375,-0.875,0.3125]

var MovRange = 0.25

InitialStance = [x,y,z]

if (Classic[0]- MovRange < InitialStance[0] && InitialStance[0] < Classic[0] + MovRange &&
Classic[1]- MovRange < InitialStance[1] && InitialStance[1] < Classic[1] + MovRange &&
Classic[2]- MovRange < InitialStance[2] && InitialStance[2] < Classic[2] + MovRange)

displayValue('StartStance', 'Classic guard')

        else if (Peakaboo[0]- MovRange < InitialStance[0] && InitialStance[0] < Peakaboo[0] +
MovRange && Peakaboo[1]- MovRange < InitialStance[1] && InitialStance[1] < Peakaboo[1] +
MovRange && Peakaboo[2]- MovRange < InitialStance[2] && InitialStance[2] < Peakaboo[2] +
MovRange)

displayValue('StartStance', 'Peak-a-boo guard')

else if (Phillyshell[0]- MovRange < InitialStance[0] && InitialStance[0] < Phillyshell[0] +
MovRange && Phillyshell[1]- MovRange < InitialStance[1] && InitialStance[1] < Phillyshell[1] +
MovRange && Phillyshell[2]- MovRange < InitialStance[2] && InitialStance[2] < Phillyshell[2] +
MovRange)

displayValue('StartStance', 'Philly Shell guard')

else

displayValue('StartStance', 'Custom guard')

}
```

Se realiza la llamada a esta función mediante un botón que el usuario pulsa una vez está preparado. Classic, Peakaboo y Phillyshell son los valores de posición del

acelerómetro en las distintas posiciones defensivas. MovRange es el valor elegido para determinar un rango en que la guardia será aceptada.

Se realizan comparaciones con los datos del acelerómetro para comprobar si está dentro del rango de las defensas registradas o en otra distinta y se guardarán los datos de posición en el vector InitialStance.

4.2.3.5 Detección de golpe

La función PunchDataToggle se encargará de detectar cuando se inicia un puñetazo. Se considerará que este evento ocurre cuando se abandona la posición inicial.

```
function PunchDataToggle(array1,array2)
{
var prueba = [0,0,0]
var check = [0,0,0]
var arrayx = [0]
var length = prueba.length
var Range = 0.25
var pos = 'en movimiento'
for (var i = 0; i < length; i++) {
        prueba[i] = array2[i]-array1[i]
        if (-Range < prueba[i] && prueba[i] < Range){
        check[i] = 1
        }
        else{
        check[i] = 0
        }
}
InitialStanceCheck = check[0] * check[1] * check[2]
if (InitialStanceCheck == 1){
pos = 'preparado'
PeakDetect (arrayX,arrayY,arrayZ)
ValleyDetect (arrayX,arrayY,arrayZ)
        if (DataHold == 0){
// En estas condiciones acaba de realizarse un golpe y se ha vuelto a la posición inicial
```

```
            RefreshX()
            RefreshY()
            RefreshZ()
            PunchClassification (peak,valley,DataHold)
            }
DataHold = 1
}
if (InitialStanceCheck == 0){
pos = 'en movimiento'
}
displayValue('togglepruebax', 'Posicion: ' + pos)
}
```

Array 1 recibe un vector con la los datos de aceleración de los ejes x, y, z. A array 2 se le asignan el vector guardado con la posición inicial. Se restan los valores para comprobar si coinciden o si están dentro de un rango admisible (Range) para considerar si el Smartfight se encuentra en la posición inicial. Si lo están se devuelve un 1 si no, un 0 que se guardará en el vector check. InitialStanceCheck será la multiplicación de los valores del vector check, si el resultado es 1 es que las posiciones coinciden y se encuentra en la posición inicial, si es un 0 no coinciden.
DataHold se comparte con la función datacollectx y permite conocer si el movimiento inicia o finaliza. pos es una variable que contiene una cadena de variables y que cambia según InitialStanceCheck para mostrar en la pantalla si el dispositivo está preparado para captar el golpe o en movimiento.

4.2.3.6 Clasificación de golpe

Existen 4 golpes básicos en el boxeo:

- Jab: puñetazo de ejecución rápido empleado como preparación para otros golpes más potentes. Se lanza con el brazo más adelantado, típicamente el izquierdo, se desplaza el puño en un movimiento recto hacia adelante acompañado por una rotación del codo.
- Cross: con el brazo más retrasado, el derecho para los diestros, se mueve el puño hacia adelante en línea recta hacia el oponente y a su vez se pivota la pierna atrasada, las caderas y el torso.
- Hook: Se realiza con el brazo adelantado, se pivota el pie frontal, las caderas y el torso y se acompaña el movimiento colocando el brazo

paralelo al torso formando un ángulo entre el antebrazo y el bíceps ligeramente superior a 90º con el codo arriba.

- Uppercut: Puede realizarse con ambos brazos, se baja la mano, se realiza una rotación del cuerpo y se dirige el puño al objetivo en un movimiento vertical respecto al suelo.

Imagen 37 – Golpes básicos boxeo

Se contó con la ayuda del experto César Borja Mollá, campeón de Europa y del mundo de kickboxing en 2005 y ya retirado de la modalidad deportiva, para la toma de datos. Se realizaron 10 medidas de los golpes que se ejecutan con la mano derecha, el cross y el uppercut que se utilizarán como base de datos (Anexo 4 "Medidas golpes boxeo"). Tras la observación de los modelos obtenidos, se decidirá implementar un sistema de comparación de picos para clasificar el puñetazo.

La función PeakDetect recibirá los vectores de aceleración arrayX, arrayY y arrayZ y detectará los picos.

```
var peak = [0,0,0]
function PeakDetect (arrayx,arrayy,arrayz){
var npeakx = 0
var npeaky = 0
var npeakz = 0
var filterpeak = 0.25
var minpeak = 0.5
for (var i = 2; i <= arrayx.length-1; i++){
        if (arrayx[i] > arrayx[i-1] + filterpeak && arrayx[i] > arrayx[i+1] + filterpeak && arrayx[i] > minpeak){
        npeakx++
        }
```

```
        if (arrayy[i] > arrayy[i-1] + filterpeak && arrayy[i] > arrayy[i+1] + filterpeak && arrayy[i] > minpeak){

        npeaky++

        }

        if (arrayz[i] > arrayz[i-1] + filterpeak && arrayz[i] > arrayz[i+1] + filterpeak && arrayz[i] > minpeak){

        npeakz++

        }

}

peak = [npeakx,npeaky,npeakz]

}
```

Las variables filterpeak y minpeak permiten filtrar pequeñas vibraciones que se producen en la toma de datos y no son picos determinantes. La función ValleyDetect opera bajo el mismo concepto pero comprueba si los valores adyacentes a arrayx[i] son mayores para detectar los valles.

El número de picos y valles son utilizados por la función PunchClassification para determinar que golpe se ha realizado.

```
var straightCounter = 0

var uppercutCounter = 0

var otherCounter = 0

var punchCounter = 0

var punch = 'golpe actual'

var MinStraighttime = 3

var MinUppercuttime = 3

var MinOthertime = 3


function PunchClassification (peak,valley){

 var straight = [2,1,0,1,0,1]

 var straightcheck = 0

 var uppercut = [1,0,0,1,1,1]

 var uppercutcheck = 0

 var punchtype = 0

 var punchtime = 0

 var MeasuredPunch = [peak[0],peak[1],peak[2],valley[0],valley[1],valley[2]]
```

```
straightcheck = Math.abs(straight[0] - MeasuredPunch[0]) + Math.abs(straight[1] - MeasuredPunch[1]) + Math.abs(straight[2] - MeasuredPunch[2]) + Math.abs(straight[3] - MeasuredPunch[3]) + Math.abs(straight[4] - MeasuredPunch[4]) + Math.abs(straight[5] - MeasuredPunch[5])

uppercutcheck = Math.abs(uppercut[0] - MeasuredPunch[0]) + Math.abs(uppercut[1] - MeasuredPunch[1]) + Math.abs(uppercut[2] - MeasuredPunch[2]) + Math.abs(uppercut[3] - MeasuredPunch[3]) + Math.abs(uppercut[4] - MeasuredPunch[4]) + Math.abs(uppercut[5] - MeasuredPunch[5])

punchtime = arrayX.length * 0.1

if (uppercutcheck > 1 && MeasuredPunch [0] == 2 && MeasuredPunch[1] >= 1){

        straightCounter++

        punchCounter++

        punch = 'Straight'

        punchtype = 1

        document.getElementById("Img").src = "images/STRAIGHT.png"

        displayValue('Punchinfo', 'Numero de golpes = ' +  straightCounter + ' de ' +
        punchCounter + '<br/>Porcentaje = ' +
        parseFloat(Math.round(straightCounter/punchCounter*100)).toFixed(0) + '%')

        document.getElementById("ExTime").innerHTML = 'Tiempo de ejecucion = ' +
        punchtime

                if(punchtime <= MinStraighttime){

                        MinStraighttime = punchtime

                        jQuery('#recordIMG').show()

                }

                else{

                        jQuery('#recordIMG').hide()

                }

}

if (uppercutcheck <= 1){

        uppercutCounter++

        punchCounter++

        punch = 'Uppercut'

        punchtype = 2

        document.getElementById("Img").src = "images/UPPERCUT.png"

        displayValue('Punchinfo', 'Numero de golpes = ' +  uppercutCounter + ' de ' +
        punchCounter + '<br/>Porcentaje = ' +
        parseFloat(Math.round(uppercutCounter/punchCounter*100)).toFixed(0) + '%')
```

```
            document.getElementById("ExTime").innerHTML = 'Tiempo de ejecucion = ' +
            punchtime

                        if(punchtime <= MinUppercuttime){

                                    MinUppercuttime = punchtime

                                    jQuery('#recordIMG').show()

                        }

                        else{

                                    jQuery('#recordIMG').hide()

                        }

}

if (arrayX.length > 7 && punchtype == 0){

            otherCounter++

            punchCounter++

            punch = 'Otro'

            punchtype = 3

            document.getElementById("Img").src = "images/OTRO.png"

            displayValue('Punchinfo', 'Numero de golpes = ' +  otherCounter + ' de ' + punchCounter
            + '<br/>Porcentaje = ' +
            parseFloat(Math.round(otherCounter/punchCounter*100)).toFixed(0) + '%')

            document.getElementById("ExTime").innerHTML = 'Tiempo de ejecucion = ' +
            punchtime

                        if(punchtime <= MinOthertime){

                                    MinOthertime = punchtime

                                    jQuery('#recordIMG').show()

                        }

                        else{

                                    jQuery('#recordIMG').hide()

                        }

}

displayValue('Punchcounter','Tipo de golpe: ' + punch + '<br/>Tiempo de ejecucion: ' +
punchtime + 's' + '<br/>Straight = ' + straightCounter + '<br/>Uppercut = ' + uppercutCounter +
'<br/>Otros = ' + otherCounter + '<br/>Golpes Totales = ' + punchCounter )

}
```

Las variables straight y uppercut contienen los picos y valles correspondientes a cada tipo de golpe obtenido de la base de datos. MeasuredPunch junta el número de picos y valles del movimiento captado para estructurar los datos

como en straight y uppercut. Straightcheck y uppercutcheck es el resultado de la resta de absolutos entre MeasuredPunch y la base de datos para comprobar con qué tipo de golpe coincide y contabilizarlos.

Se modificaron las condiciones para que se detecte un straight ya que, las pruebas realizadas por usuarios inexpertos, daban lugar a aceleraciones en el eje Z caóticas. La variable punchtime calcula el tiempo de ejecución del golpe multiplicando la longitud del vector de aceleración por el periodo de muestreo.

4.2.4 Interfaz de usuario

La pantalla principal mostrará el botón de posición inicial que guardará la posición actual del dsipositivo y la tomará como punto de partida y final en los golpes realizados. Aparecerán también 2 pestañas VISUAL y DATOS que corresponde a los métodos de visualización de la aplicación, una focalizada en datos y otro más intuitiva y visual. Finalmente el botón Salir se usará para cerrar la aplicación.

La pestaña datos contendrá una interfaz focalizada en la representación de datos recogidos por el sensor. Por pantalla se mostrarán los valores instantáneos de aceleración en fuerzas g de los ejes x, y, z así como el módulo. Se mostrarán los máximos y mínimos alcanzados. Las gráficas dinámicas se mostraran u ocultarán al pulsar el botón correspondiente. El texto Posición inicial mostrará si la guardia adoptada se corresponde a una de las defensas típicas o a una personalizada. Se mostrará el estado del dispositivo, listo para medir o tomando medidas del movimiento. Exhibirá también el número de golpes contabilizados de cada tipo y su tiempo de ejecución así como la representación gráfica del último golpe registrado que podrá verse u ocultarse pulsando un botón.

La pestaña Visual representarálos datos del último golpe registrado de manera sencilla. Una imagen aparecerá dependiendo del golpe realizado. Bajo de esta, texto indicando el número del tipo de golpe realizado sobre el total así como el porcentaje. Finalmente se representará el tiempo de ejecución de cada golpe, y si es el menor registrado dentro de la categoría del golpe, se mostrará una pequeña copa a la izquierda del tecto para indicar nuevo récord.

El código de la interfaz es el siguiente:

Definición de estilos:

```
<style type="text/css">
        body {
        background: rgb(25,25,25);
```

```
            color: rgb(255,255,255);
            font-size: 100%;
            padding: 0px 20px;
}
.SensorData {
            font-size: 120%;
            font-weight: bold;
}
.SensorData span {
            font-weight: normal;
}
${demo.css}
</style>
<style>
ul.tab {
   list-style-type: none;
   margin: 0;
   padding: 0;
   overflow: hidden;
   border: 2px solid #00aedb;
   background-color: #3B90AF;
}
/* Float the list items side by side */
ul.tab li {float: left;}
/* Style the links inside the list items */
ul.tab li a {
   display: inline-block;
   color: white;
   text-align: center;
   padding: 14px 16px;
   text-decoration: none;
   transition: 0.3s;
   font-size: 17px;
```

```
    font-Weight: bold;
}
/* Create an active/current tablink class */
ul.tab li a:focus, .active {
    background-color: #48d1cc;
}

/* Style the tab content */
.tabcontent {
    display: none;
    padding: 6px 12px;
    border: 2px solid #00aedb;
    border-top: none;
    background-color: #07101f;
}
</style>
```

Cuerpo HTML:

```
</head>
<body>
<h1>Smartfight</h1>
<div class="SensorData">Estado: <span id="StatusData">Ready to connect</span></div>
<button style ='margin-top:15px;margin-bottom:15px'
onclick="GetStance(AccelValueXSensorTag,AccelValueYSensorTag,AccelValueZSensorTag)">Posi
cion Inicial</button>
<ul class="tab">
  <li><a href="#" class="tablinks" onclick="openTab(event, 'Data')">DATOS</a></li>
  <li><a href="#" class="tablinks" onclick="openTab(event, 'Visual')">VISUAL</a></li>
</ul>

<div id="Data" class="tabcontent">
<div class="SensorData">Datos de aceleracion:<br/><span id="AccelerometerData">[Waiting
for value]</span></div>
<div class="SensorData">Aceleracion maxima X:<br/><span
id="MaxXAccelerometerData">[Waiting for value]</span></div>
```

```
<div class="SensorData">Aceleracion maxima Y:<br/><span
id="MaxYAccelerometerData">[Waiting for value]</span></div>

<div class="SensorData">Aceleracion maxima Z:<br/><span
id="MaxZAccelerometerData">[Waiting for value]</span></div>

<div class="SensorData">Graficas de aceleracion dinamicas<br/></div>

<div id="containerx" style="display:none; min-width: 280px; height: 400px; margin: 0
auto"></div>

<input type='button' style ='margin-right:15px' id='hideshowx' value='EJE X'>

<div id="containery" style="display:none; min-width: 280px; height: 400px; margin: 0
auto"></div>

<input type='button' style ='margin-right:15px' id='hideshowy' value='EJE Y'>

<div id="containerz" style="display:none; min-width: 280px; height: 400px; margin: 0
auto"></div>

<input type='button' style ='margin-right:15px' id='hideshowz' value='EJE Z'>

<br>


<div class="SensorData">Posicion inicial:<br/><span id="StartStance">[Waiting for
value]</span></div>

<div class="SensorData">Estado:<br/><span id="togglepruebax">[Waiting for
value]</span></div>

<div class="SensorData">Golpes lanzados:<br/><span id="Punchcounter">[Waiting for
value]</span></div>


<div class="SensorData">Graficas del ultimo golpe registrado<br/></div>


<div id="staticx" style="display:none; min-width: 280px; height: 400px; margin: 0 auto"></div>

<input type='button'  style ='margin-right:15px'id='hideshowsx' value='EJE X'>

<div id="staticy" style="display:none; min-width: 280px; height: 400px; margin: 0 auto"></div>

<input type='button' style ='margin-right:15px' id='hideshowsy' value='EJE Y'>

<div id="staticz" style="display:none; min-width: 280px; height: 400px; margin: 0 auto"></div>

<input type='button' style ='margin-right:15px' id='hideshowsz' value='EJE Z'>

<br>

</div>


<div id="Visual" class="tabcontent">

<img id= "Img" alt="Punch Icon" style="width:128px;height:180px;">
```

```
<div class="SensorData">Datos golpe:<br/><span id="Punchinfo">[Waiting for value]</span></div>

<img id= "recordIMG" src="images/record.png" style="display:none;float:left;padding: 16px 0px 5px 0;" width="24" height="24">

<p id="ExTime" style="font-size:120%"></p>

</div>

<br>

<button onclick="navigator.app.exitApp()">Salir</button>
```

Javascript:

```
//Muestra u oculta las gráficas cuando se pulsa el botón

jQuery(document).ready(function(){
   jQuery('#hideshowx').on('click', function(event) {
       jQuery('#containerx').toggle();
   });
});


jQuery(document).ready(function(){
   jQuery('#hideshowy').on('click', function(event) {
       jQuery('#containery').toggle();
   });
});


jQuery(document).ready(function(){
   jQuery('#hideshowz').on('click', function(event) {
       jQuery('#containerz').toggle();
   });
});
//estatica
jQuery(document).ready(function(){
   jQuery('#hideshowsx').on('click', function(event) {
       jQuery('#staticx').toggle();
   });
});
```

```
jQuery(document).ready(function(){
    jQuery('#hideshowsy').on('click', function(event) {
        jQuery('#staticy').toggle();
    });
});

jQuery(document).ready(function(){
    jQuery('#hideshowsz').on('click', function(event) {
        jQuery('#staticz').toggle();
    });
});
function openTab(evt, Name) {
    var i, tabcontent, tablinks;
//Oculta los elementos con la clase tabcontent
    tabcontent = document.getElementsByClassName("tabcontent");
    for (i = 0; i < tabcontent.length; i++) {
        tabcontent[i].style.display = "none";
    }
//Selecciona los elementos con la clase tablinks y remueve la clase active
    tablinks = document.getElementsByClassName("tablinks");
    for (i = 0; i < tablinks.length; i++) {
        tablinks[i].className = tablinks[i].className.replace(" active", "");
    }
//Muestra la pestaña actual y añade la clase activa al link que abre la pestaña
    document.getElementById(Name).style.display = "block";
    evt.currentTarget.className += " active";
}
```

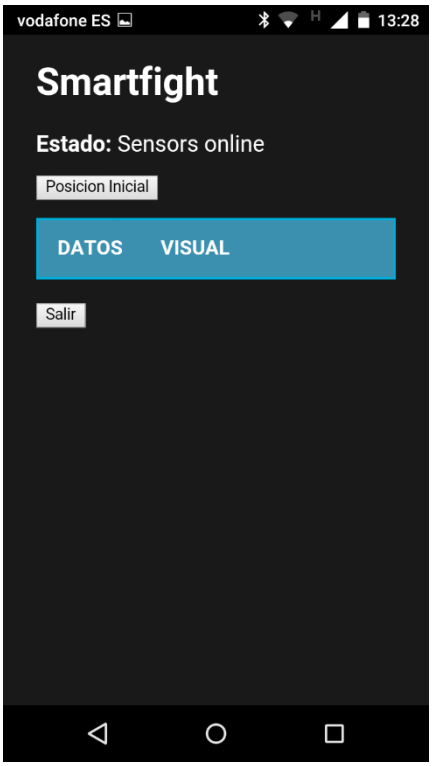


Imagen 38 – Interfaz de usuario pantalla principal

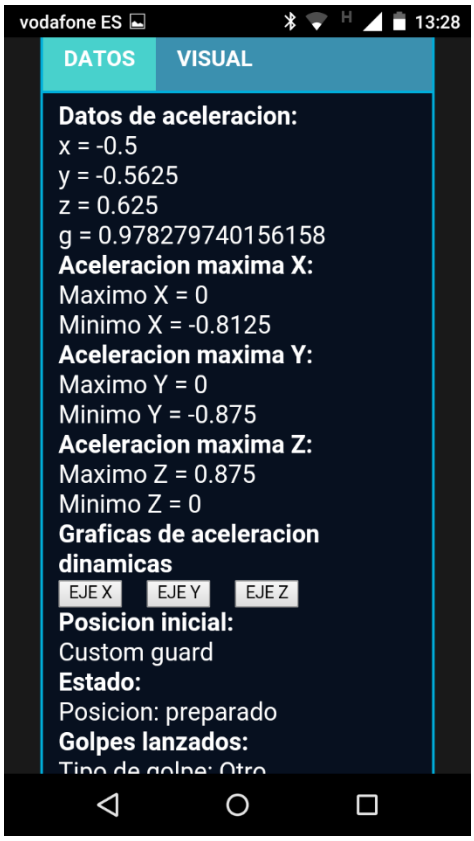


Imagen 39 – Interfaz de usuario DATOS 1

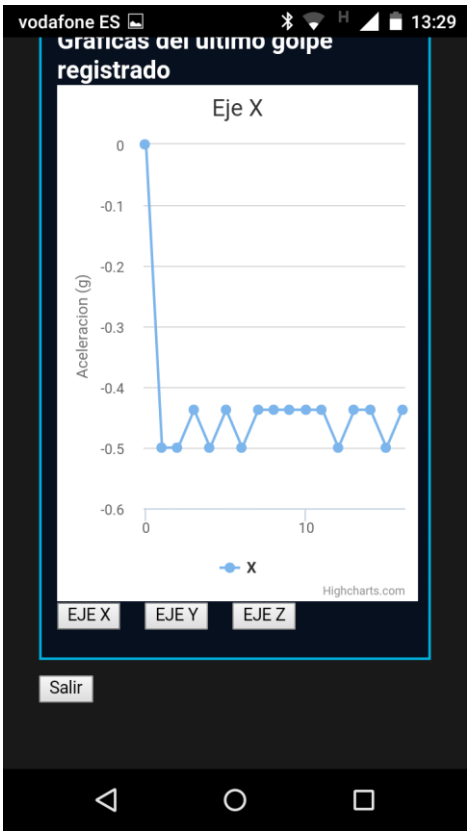


Imagen 40 – Interfaz de usuario en DATOS 2

Imagen 41 – Interfaz de usuario VISUAL 1

Imagen 42 – Interfaz de usuario VISUAL 2

Imagen 43 – Interfaz de usuario VISUAL 3

5. Implementación Android

Para trasladar el proyecto a la plataforma Android se utilizará Apache Cordova. Se trata de un framework de código abierto para desarrollo de aplicaciones móviles. Permite utilizar tecnologías estándar web como HTML5, CSS3 y Javascript.

Feature	Android[35]
Accelerometer	Yes
Camera	Yes
Compass	Yes
Contacts	Yes
File	Yes
Geolocation	Yes
Media	Yes
Network	Yes
Notification (alert, sound, vibration)	Yes
Storage	Yes

Imagen 52 – Características soportadas por cordova

Se necesita la instalación de herramientas previas antes de poder desarrollar aplicaciones móviles con Cordova:

- Node.js.
- Git.
- Cordova.
- SDK Java.
- Apache Ant.
- Android SDK Tools.

Una vez se cumplan los requisitos, los pasos a seguir para implementar la aplicación en Android son los siguientes:

- Abrir una pantalla de comandos (CMD).
- Crear un proyecto Cordova.
- Añadir los plugins necesarios.
- Añadir la plataforma Android.
- Compilar el proyecto.
- Enviar la aplicación al Smartphone.

Para el proyecto de este documento las líneas de código que se utilizaron en el CMD son las siguientes:

```
//Ubicación del proyecto
cd /desktop/tfg/cordova_projects

//Creación del proyecto Cordova
cordova create smartfight com.evothings.p7 Smartfight
//Se sustituyen los elementos de la carpeta WWW creada en el proyecto nuevo por los archivos html y js del proyecto que se desea implementar en android

//Ubicación de la carpeta creada
cd smartfight

//Añadir plugin BLE
cordova plugin add cordova-plugin-ble

//Añadir plugin splash screen
Cordova plugin add cordova-plugin-splashscreen

//Añadir plataforma
cordova platform add Android

//Construir proyecto
cordova build Android

//Ubicación app Android
cd platforms\android\build\outputs\apk

//Instalar la aplicación en el dispositivo móvil conectado al puerto USB
adb -d install android-debug.apk
```

Para poder realizar este paso deben habilitarse las opciones de desarrollador en el Smartphone, los pasos a seguir varían según el modelo.

Para añadir iconos y splash screens propios, se deben sustituir los elementos por defecto que aparecen al crear el proyecto por los personalizados en las carpetas siguiendo la siguiente ruta:

Ruta del proyecto/platforms/Android/res

Los iconos deberán tener las siguientes medidas para adaptarse a los distintos modelos de pantalla.

- 36x36px ldpi
- 48x48px mdpi
- 72x72px hdpi

- 96x96px hdpi

Imagen 44 – Icono Smartfight

En cuanto a las splash sreens deberán tener las siguientes dimensiones:

- 200x320px port ldpi
- 320x200px land ldpi
- 320x480px port mdpi
- 480x320px land mdpi
- 480x800px port hdpi
- 800x480px land hdpi
- 720x1280px port xhdpi
- 1280x720px land xhdpi

Imagen 45 – Splash screen Smartfight

6. Código

```
<!DOCTYPE html>
<!--
	Demonstration of the TI SensorTag JavaScript library.
-->
<html>
<head>
	<meta charset="utf-8" />
	<meta name="format-detection" content="telephone=no" />
	<meta name="viewport" content="width=device-width, initial-scale=1.0" />

	<title>SmartfightTabs</title>

	<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
	<link rel="stylesheet" href="common.css" />
	<style type="text/css">
		body {
		background: rgb(25,25,25);
		color: rgb(255,255,255);
		font-size: 100%;
		padding: 0px 20px;
	}
	.SensorData {
		font-size: 120%;
		font-weight: bold;
	}
	.SensorData span {
		font-weight: normal;
	}
${demo.css}
	</style>
```

```
        <style>

ul.tab {
    list-style-type: none;
    margin: 0;
    padding: 0;
    overflow: hidden;
    border: 2px solid #00aedb;
    background-color: #3B90AF;
}

/* Float the list items side by side */
ul.tab li {float: left;}

/* Style the links inside the list items */
ul.tab li a {
    display: inline-block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
    transition: 0.3s;
    font-size: 17px;
    font-Weight: bold;
}
/* Create an active/current tablink class */
ul.tab li a:focus, .active {
    background-color: #48d1cc;
}

/* Style the tab content */
.tabcontent {
    display: none;
```

```
    padding: 6px 12px;
    border: 2px solid #00aedb;
    border-top: none;
    background-color: #07101f;
}
</style>


            <script type="text/javascript">
$(function () {
    $(document).ready(function () {
        Highcharts.setOptions({
            global: {
                useUTC: false
            }
        });

        $('#containerx').highcharts({
            chart: {
                type: 'line',
                animation: Highcharts.svg, // don't animate in old IE
                marginRight: 10,
                events: {
                    load: function () {

                        // set up the updating of the chart each second
                        var series = this.series[0];
                        setInterval(function () {
                            var x = (new Date()).getTime(), // current time
                                y = AccelValueXSensorTag ;//AQUI EL VALOR DE LA ACELERACIÓN
                            series.addPoint([x, y], true, true);
                        }, 400); // tiempo de muestreo
                    }
                }
```

```
},
title: {
   text: 'Eje X'
},
xAxis: {
   type: 'datetime',
   tickPixelInterval: 150
},
yAxis: {
   title: {
      text: 'Aceleracion (g)'
   },
   plotLines: [{
      value: 0,
      width: 1,
      color: '#808080'
   }]
},
tooltip: {
   formatter: function () {
      return '<b>' + this.series.name + '</b><br/>' +
         Highcharts.dateFormat('%Y-%m-%d %H:%M:%S', this.x) + '<br/>' +
         Highcharts.numberFormat(this.y, 2);
   }
},
legend: {
   enabled: false
},
                        plotOptions: {
   line: {
      marker: {
         enabled: false
      }
```

```
            }
        },
        exporting: {
            enabled: false
        },
        series: [{
            name: 'x',
            data: (function () {
                // generate an array of random data
                var data = [],
                    time = (new Date()).getTime(),
                    i;

                for (i = -19; i <= 0; i += 1) {
                    data.push({
                        x: time + i * 400,
                        y: 0 // Asi se inicializa a 0
                    });
                }
                return data;
            }())
        }]
    });
  });
});
            </script>
<script type="text/javascript">
$(function () {
   $(document).ready(function () {
      Highcharts.setOptions({
         global: {
            useUTC: false
         }
```

```
});

$('#containery').highcharts({
    chart: {
        type: 'line',
        animation: Highcharts.svg, // don't animate in old IE
        marginRight: 10,
        events: {
            load: function () {

                // set up the updating of the chart each second
                var series = this.series[0];
                setInterval(function () {
                    var x = (new Date()).getTime(), // current time
                        y = AccelValueYSensorTag ;//AQUI EL VALOR DE LA ACELERACIÓN
                    series.addPoint([x, y], true, true);
                }, 400); // tiempo de muestreo
            }
        }
    },
    title: {
        text: 'Eje Y'
    },
    xAxis: {
        type: 'datetime',
        tickPixelInterval: 150
    },
    yAxis: {
        title: {
            text: 'Aceleracion (g)'
        },
        plotLines: [{
            value: 0,
```

```
        width: 1,
        color: '#808080'
    }]
},
tooltip: {
    formatter: function () {
        return '<b>' + this.series.name + '</b><br/>' +
            Highcharts.dateFormat('%Y-%m-%d %H:%M:%S', this.x) + '<br/>' +
            Highcharts.numberFormat(this.y, 2);
    }
},
legend: {
    enabled: false
},
                        plotOptions: {
    line: {
        marker: {
            enabled: false
        }
    }
},
exporting: {
    enabled: false
},
series: [{
    name: 'y',
    data: (function () {
        // generate an array of random data
        var data = [],
            time = (new Date()).getTime(),
            i;

        for (i = -19; i <= 0; i += 1) {
```

```
                data.push({
                    x: time + i * 400,
                    y: 0 // Asi se inicializa a 0
                });
            }
            return data;
        }())
    }]
  });
});
        </script>
<script type="text/javascript">
$(function () {
  $(document).ready(function () {
    Highcharts.setOptions({
      global: {
        useUTC: false
      }
    });

    $('#containerz').highcharts({
      chart: {
        type: 'line',
        animation: Highcharts.svg, // don't animate in old IE
        marginRight: 10,
        events: {
          load: function () {

            // set up the updating of the chart each second
            var series = this.series[0];
            setInterval(function () {
              var x = (new Date()).getTime(), // current time
```

```
                y = AccelValueZSensorTag  ;//AQUI EL VALOR DE LA ACELERACIÓN
            series.addPoint([x, y], true, true);
          }, 400); // tiempo de muestreo
        }
    }
},
title: {
    text: 'Eje Z'
},
xAxis: {
    type: 'datetime',
    tickPixelInterval: 150
},
yAxis: {
    title: {
        text: 'Aceleracion (g)'
    },
    plotLines: [{
        value: 0,
        width: 1,
        color: '#808080'
    }]
},
tooltip: {
    formatter: function () {
        return '<b>' + this.series.name + '</b><br/>' +
            Highcharts.dateFormat('%Y-%m-%d %H:%M:%S', this.x) + '<br/>' +
            Highcharts.numberFormat(this.y, 2);
    }
},
legend: {
    enabled: false
},
```

```
                    plotOptions: {
            line: {
                marker: {
                    enabled: false
                }
            }
        },
        exporting: {
            enabled: false
        },
        series: [{
            name: 'z',
            data: (function () {
                // generate an array of random data
                var data = [],
                    time = (new Date()).getTime(),
                    i;

                for (i = -19; i <= 0; i += 1) {
                    data.push({
                        x: time + i * 400,
                        y: 0 // Asi se inicializa a 0
                    });
                }
                return data;
            }())
        }]
    });
  });
});

                </script>
```

```
                    <script type="text/javascript">
function RefreshX(){
$(function () {
   $('#staticx').highcharts({
      chart: {
         type: 'line'
      },
      title: {
         text: 'Eje X'
      },
      yAxis: {
         title: {
            text: 'Aceleracion (g)'
         }
      },
                    exporting: {
            enabled: false
      },
      series: [{
         name: 'X',
         data: arrayX
      }]
   });
});
}

function RefreshY(){
$(function () {
   $('#staticy').highcharts({
      chart: {
         type: 'line'
      },
```

```
        title: {
            text: 'Eje Y'
        },
        yAxis: {
            title: {
                text: 'Aceleracion (g)'
            }
        },
                        exporting: {
                text: false
        },
        series: [{
            name: 'Y',
            data: arrayY
        }]
    });
});
}

function RefreshZ(){
$(function () {
    $('#staticz').highcharts({
        chart: {
            type: 'line'
        },
        title: {
            text: 'Eje Z'
        },
        yAxis: {
            title: {
                text: 'Aceleracion (g)'
            }
        },
```

```
                    exporting: {
            enabled: false
        },
        series: [{
            name: 'Z',
            data: arrayZ
        }]
    });
});
}
                    </script>

            <script>
            // Redirect console.log to Evothings Workbench.
            if (window.hyper) { console.log = hyper.log; }
            </script>
            <script src="cordova.js"></script>
            <script src="easy-ble.js"></script>
            <script src="ti-sensortag.js"></script>
</head>
<body>
            <h1>Smartfight</h1>
            <script src="highcharts.js"></script>
            <script src="exporting.js"></script>
            <script src="jspdf.js"></script>
            <script src="FileSaver.js"></script>




            <div class="SensorData">Estado: <span id="StatusData">Ready to connect</span></div>
            <button style ='margin-top:15px;margin-bottom:15px'
onclick="GetStance(AccelValueXSensorTag,AccelValueYSensorTag,AccelValueZSensorTag)">Posicion
Inicial</button>
```

```
<ul class="tab">
  <li><a href="#" class="tablinks" onclick="openTab(event, 'Data')">DATOS</a></li>
  <li><a href="#" class="tablinks" onclick="openTab(event, 'Visual')">VISUAL</a></li>

</ul>

<div id="Data" class="tabcontent">

        <div class="SensorData">Datos de aceleracion:<br/><span id="AccelerometerData">[Waiting for value]</span></div>

        <div class="SensorData">Aceleracion maxima X:<br/><span id="MaxXAccelerometerData">[Waiting for value]</span></div>

        <div class="SensorData">Aceleracion maxima Y:<br/><span id="MaxYAccelerometerData">[Waiting for value]</span></div>

        <div class="SensorData">Aceleracion maxima Z:<br/><span id="MaxZAccelerometerData">[Waiting for value]</span></div>

        <div class="SensorData">Graficas de aceleracion dinamicas<br/></div>

        <div id="containerx" style="display:none; min-width: 280px; height: 400px; margin: 0 auto"></div>

        <input type='button' style ='margin-right:15px' id='hideshowx' value='EJE X'>

        <div id="containery" style="display:none; min-width: 280px; height: 400px; margin: 0 auto"></div>

        <input type='button' style ='margin-right:15px' id='hideshowy' value='EJE Y'>

        <div id="containerz" style="display:none; min-width: 280px; height: 400px; margin: 0 auto"></div>

        <input type='button' style ='margin-right:15px' id='hideshowz' value='EJE Z'>

        <br>


        <div class="SensorData">Posicion inicial:<br/><span id="StartStance">[Waiting for value]</span></div>

        <div class="SensorData">Estado:<br/><span id="togglepruebax">[Waiting for value]</span></div>

        <div class="SensorData">Golpes lanzados:<br/><span id="Punchcounter">[Waiting for value]</span></div>


        <div class="SensorData">Graficas del ultimo golpe registrado<br/></div>
```

```
        <div id="staticx" style="display:none; min-width: 280px; height: 400px; margin: 0 auto"></div>
        <input type='button'  style ='margin-right:15px'id='hideshowsx' value='EJE X'>
        <div id="staticy" style="display:none; min-width: 280px; height: 400px; margin: 0 auto"></div>
        <input type='button' style ='margin-right:15px' id='hideshowsy' value='EJE Y'>
        <div id="staticz" style="display:none; min-width: 280px; height: 400px; margin: 0 auto"></div>
        <input type='button' style ='margin-right:15px' id='hideshowsz' value='EJE Z'>
        <br>
        </div>


<div id="Visual" class="tabcontent">
        <img id= "Img" alt="Punch Icon" style="width:128px;height:180px;">
        <div class="SensorData">Datos golpe:<br/><span id="Punchinfo">[Waiting for
value]</span></div>
        <img id= "recordIMG" src="images/record.png" style="display:none;float:left;padding: 16px
0px 5px 0;" width="24" height="24">
        <p id="ExTime" style="font-size:120%"></p>
</div>
<br>
<button onclick="navigator.app.exitApp()">Salir</button>


        <script>
        var sensorTag = TISensorTag.createInstance()
        var AccelValueXSensorTag = 0
        var AccelValueYSensorTag = 0
        var AccelValueZSensorTag = 0
        var MaxAccelValueXSensorTag = 0
        var MaxAccelValueYSensorTag = 0
        var MaxAccelValueZSensorTag = 0
        var MinAccelValueXSensorTag = 0
        var MinAccelValueYSensorTag = 0
        var MinAccelValueZSensorTag = 0
        var InitialStance = [0,0,0]
        var InitialStanceCheck = 1
        var DataHold = 1
```

```
            var arrayX = [0]
            var arrayY = [0]
            var arrayZ = [0]
            var peak = [0,0,0]
            var valley = [0,0,0]

            var straightCounter = 0
            var uppercutCounter = 0
            var otherCounter = 0
            var punchCounter = 0
            var punch = 'golpe actual'
            var MinStraighttime = 3
            var MinUppercuttime = 3
            var MinOthertime = 3

            //Muestra u oculta las gráficas cuando se pulsa el botón
            jQuery(document).ready(function(){
   jQuery('#hideshowx').on('click', function(event) {
       jQuery('#containerx').toggle();
   });
});

            jQuery(document).ready(function(){
   jQuery('#hideshowy').on('click', function(event) {
       jQuery('#containery').toggle();
   });
});
```

```
        jQuery(document).ready(function(){
  jQuery('#hideshowz').on('click', function(event) {
      jQuery('#containerz').toggle();
  });
});
//estatica
        jQuery(document).ready(function(){
  jQuery('#hideshowsx').on('click', function(event) {
      jQuery('#staticx').toggle();
  });
});

        jQuery(document).ready(function(){
  jQuery('#hideshowsy').on('click', function(event) {
      jQuery('#staticy').toggle();
  });
});

        jQuery(document).ready(function(){
  jQuery('#hideshowsz').on('click', function(event) {
      jQuery('#staticz').toggle();
  });
});
function openTab(evt, Name) {
  var i, tabcontent, tablinks;
        //Oculta los elementos con la clase tabcontent
  tabcontent = document.getElementsByClassName("tabcontent");
  for (i = 0; i < tabcontent.length; i++) {
    tabcontent[i].style.display = "none";
  }
        //Selecciona los elementos con la clase tablinks y remueve la clase active
  tablinks = document.getElementsByClassName("tablinks");
  for (i = 0; i < tablinks.length; i++) {
```

```
    tablinks[i].className = tablinks[i].className.replace(" active", "");
  }
        //Muestra la pestaña actual y añade la clase activa al link que abre la pestaña
  document.getElementById(Name).style.display = "block";
  evt.currentTarget.className += " active";
}

        function initialiseSensortag()
        {
                // Here sensors are set up.
                // First parameter is the callback function.
                // Accelerometer, Magnetometer and Gyroscope take a
                // millisecond update interval as the last parameter.

                sensorTag
                        .statusCallback(statusHandler)
                        .errorCallback(errorHandler)
                        //Periodo del acelerometro en milisegundos, minimo 100
                        .accelerometerCallback(accelerometerHandler, 100)

                        .connectToClosestDevice()
        }

        function statusHandler(status)
        {
                displayValue('StatusData', status)
        }

        function errorHandler(error)
        {
                displayValue('StatusData', 'Error: ' + error)
                if ('disconnected' == error)
                {
```

```
                // If disconneted attempt to connect again.
                setTimeout(
                        function() { sensorTag.connectToClosestDevice() },
                        1000)
        }
}

//Datos del acelerometro

function accelerometerHandler(data)
{
        //console.log('length: ' + data.length)
        //console.log('acceldata: ' + data[0] + ' ' + data[1] + ' ' + data[2])

        var x = data[0] / 16
        var y = data[1] / 16
        var z = data[2] / 16
        var g = Math.sqrt((x*x)+(y*y)+(z*z))
        AccelValueXSensorTag = x
        AccelValueYSensorTag = y
        AccelValueZSensorTag = z

        var AccelArray = [x,y,z]

        //Cada escalón es de 0.0625g si mide entre +-8g




        displayValue(
                'AccelerometerData',
                'x = ' + x + '<br/>y = ' + y + '<br/>z = ' + z + '<br/>g = ' + g)
```

```
            MaxAccelerations(x,y,z)
            PunchDataToggle(AccelArray,InitialStance)
            datacollectx (AccelArray,InitialStanceCheck)
            datacollecty (AccelArray,InitialStanceCheck)
            datacollectz (AccelArray,InitialStanceCheck)
        }

        //valores máximos y mínimos de aceleración

        function MaxAccelerations(x,y,z){

            if (MaxAccelValueXSensorTag < x){
                MaxAccelValueXSensorTag = x;
            }
            if (MaxAccelValueYSensorTag < y){
                MaxAccelValueYSensorTag = y;
            }
            if (MaxAccelValueZSensorTag < z){
                MaxAccelValueZSensorTag = z;
            }
            if (MinAccelValueXSensorTag > x){
            MinAccelValueXSensorTag = x;
            }
            if (MinAccelValueYSensorTag > y){
                MinAccelValueYSensorTag = y;
            }
            if (MinAccelValueZSensorTag > z){
                MinAccelValueZSensorTag = z;
            }
        displayValue('MaxXAccelerometerData', 'Maximo X = ' + MaxAccelValueXSensorTag +
'<br/>Minimo X = ' + MinAccelValueXSensorTag)
        displayValue('MaxYAccelerometerData', 'Maximo Y = ' + MaxAccelValueYSensorTag +
'<br/>Minimo Y = ' + MinAccelValueYSensorTag)
        displayValue('MaxZAccelerometerData', 'Maximo Z = ' + MaxAccelValueZSensorTag +
'<br/>Minimo Z = ' + MinAccelValueZSensorTag)
```

```
}



// Define la posicion inicial


function GetStance(x,y,z)
{


        var Classic = [-0.8125,-0.3125,0.5]
        var Peakaboo = [-0.9375,-0.1875,0]
        var Phillyshell = [-0.4375,-0.875,0.3125]
        var MovRange = 0.25


        InitialStance = [x,y,z]



        if (Classic[0]- MovRange < InitialStance[0] && InitialStance[0] < Classic[0] + MovRange && Classic[1]- MovRange < InitialStance[1] && InitialStance[1] < Classic[1] + MovRange && Classic[2]- MovRange < InitialStance[2] && InitialStance[2] < Classic[2] + MovRange)
        displayValue('StartStance', 'Classic guard')
        else if (Peakaboo[0]- MovRange < InitialStance[0] && InitialStance[0] < Peakaboo[0] + MovRange && Peakaboo[1]- MovRange < InitialStance[1] && InitialStance[1] < Peakaboo[1] + MovRange && Peakaboo[2]- MovRange < InitialStance[2] && InitialStance[2] < Peakaboo[2] + MovRange)
        displayValue('StartStance', 'Peak-a-boo guard')
        else if (Phillyshell[0]- MovRange < InitialStance[0] && InitialStance[0] < Phillyshell[0] + MovRange && Phillyshell[1]- MovRange < InitialStance[1] && InitialStance[1] < Phillyshell[1] + MovRange && Phillyshell[2]- MovRange < InitialStance[2] && InitialStance[2] < Phillyshell[2] + MovRange)
        displayValue('StartStance', 'Philly Shell guard')
        else
        displayValue('StartStance', 'Custom guard')
}


//Detecta si el dispositivo esta en la posicion inicial


function PunchDataToggle(array1,array2)
```

```
{
var prueba = [0,0,0]
var check = [0,0,0]
var arrayx = [0]
var length = prueba.length
var Range = 0.25
var pos = 'en movimiento'
        for (var i = 0; i < length; i++) {
                prueba[i] = array2[i]-array1[i]
                if (-Range < prueba[i] && prueba[i] < Range){
                check[i] = 1
                }
                else{
                check[i] = 0
                }
        }
InitialStanceCheck = check[0] * check[1] * check[2]
        if (InitialStanceCheck == 1){
        pos = 'preparado'
        PeakDetect (arrayX,arrayY,arrayZ)
        ValleyDetect (arrayX,arrayY,arrayZ)
                if (DataHold == 0){
                // En estas condiciones acaba de realizarse un golpe y se ha vuelto a la
posicion inicial
                RefreshX()
                RefreshY()
                RefreshZ()
                PunchClassification (peak,valley,DataHold)
                }
        DataHold = 1
        }
        if (InitialStanceCheck == 0){
        pos = 'en movimiento'
        }
```

```
displayValue('togglepruebax', 'Posicion: ' + pos)
}

//Crea el vector de aceleracion X

function datacollectx(accelarray,check){

if (check == 0){
        if (DataHold == 1){
        //En estas condiciones acaba de lanzarse el golpe
        arrayX = [0]
        arrayY = [0]
        arrayZ = [0]
        DataHold = 0
        }
arrayX.push(accelarray[0])
        if (arrayX.length > 25)
        arrayX = [0]

}
}

//Crea el vector de aceleracion Y

function datacollecty(accelarray,check){

if (check == 0){
arrayY.push(accelarray[1])
        if (arrayY.length > 25)
        arrayY = [0]

}
}
```

```
//Crea el vector de aceleracion Z

function datacollectz(accelarray,check){

if (check == 0){
arrayZ.push(accelarray[2])
        if (arrayZ.length > 25)
        arrayZ = [0]

}
}

//Deteccion de picos

function PeakDetect (arrayx,arrayy,arrayz){

var npeakx = 0
var npeaky = 0
var npeakz = 0
var filterpeak = 0.25
var minpeak = 0.5


for (var i = 2; i <= arrayx.length-1; i++){
        if (arrayx[i] > arrayx[i-1] + filterpeak && arrayx[i] > arrayx[i+1] + filterpeak && arrayx[i]
> minpeak){
        npeakx++
        }
        if (arrayy[i] > arrayy[i-1] + filterpeak && arrayy[i] > arrayy[i+1] + filterpeak && arrayy[i]
> minpeak){
        npeaky++
        }
```

```
                if (arrayz[i] > arrayz[i-1] + filterpeak && arrayz[i] > arrayz[i+1] + filterpeak && arrayz[i]
> minpeak){
                npeakz++
                }
        }
        peak = [npeakx,npeaky,npeakz]
        }

        //Deteccion de valles

        function ValleyDetect (arrayx,arrayy,arrayz){

        var nvalleyx = 0
        var nvalleyy = 0
        var nvalleyz = 0
        var filtervalley = 0.25
        var minvalley = -0.5


        for (var i = 2; i <= arrayx.length-1; i++){
                if (arrayx[i] < arrayx[i-1] - filtervalley && arrayx[i] < arrayx[i+1] - filtervalley && arrayx[i]
< minvalley){
                nvalleyx++
                }
                if (arrayy[i] < arrayy[i-1] - filtervalley && arrayy[i] < arrayy[i+1] - filtervalley &&
arrayy[i] < minvalley){
                nvalleyy++
                }

                if (arrayz[i] < arrayz[i-1] - filtervalley && arrayz[i] < arrayz[i+1] - filtervalley && arrayz[i]
< minvalley){
                nvalleyz++
                }
        }
        valley = [nvalleyx,nvalleyy,nvalleyz]
```

```
}

//Clasificacion de golpes

function PunchClassification (peak,valley){

 var straight = [2,1,0,1,0,1]
 var straightcheck = 0
 var uppercut = [1,0,0,1,1,1]
 var uppercutcheck = 0
 var punchtype = 0
 var punchtime = 0

 var MeasuredPunch = [peak[0],peak[1],peak[2],valley[0],valley[1],valley[2]]

straightcheck = Math.abs(straight[0] - MeasuredPunch[0]) + Math.abs(straight[1] -
MeasuredPunch[1]) + Math.abs(straight[2] - MeasuredPunch[2]) + Math.abs(straight[3] -
MeasuredPunch[3]) + Math.abs(straight[4] - MeasuredPunch[4]) + Math.abs(straight[5] -
MeasuredPunch[5])

uppercutcheck = Math.abs(uppercut[0] - MeasuredPunch[0]) + Math.abs(uppercut[1] -
MeasuredPunch[1]) + Math.abs(uppercut[2] - MeasuredPunch[2]) + Math.abs(uppercut[3] -
MeasuredPunch[3]) + Math.abs(uppercut[4] - MeasuredPunch[4]) + Math.abs(uppercut[5] -
MeasuredPunch[5])

punchtime = arrayX.length/10

        if (uppercutcheck > 1 && MeasuredPunch [0] == 2 && MeasuredPunch[1] >= 1){
        straightCounter++
        punchCounter++
        punch = 'Straight'
        punchtype = 1
        document.getElementById("Img").src = "images/STRAIGHT.png"
        displayValue('Punchinfo', 'Numero de golpes = ' +  straightCounter + ' de ' +
punchCounter + '<br/>Porcentaje = ' +
parseFloat(Math.round(straightCounter/punchCounter*100)).toFixed(0) + '%')
```

```
                document.getElementById("ExTime").innerHTML = 'Tiempo de ejecucion = ' +
punchtime
                        if(punchtime <= MinStraighttime){
                        MinStraighttime = punchtime
                        jQuery('#recordIMG').show()
                        }
                        else{
                        jQuery('#recordIMG').hide()
                        }
                }
                if (uppercutcheck <= 1){
                uppercutCounter++
                punchCounter++
                punch = 'Uppercut'
                punchtype = 2
                document.getElementById("Img").src = "images/UPPERCUT.png"
                displayValue('Punchinfo', 'Numero de golpes = ' +  uppercutCounter + ' de ' +
punchCounter + '<br/>Porcentaje = ' +
parseFloat(Math.round(uppercutCounter/punchCounter*100)).toFixed(0) + '%')
                document.getElementById("ExTime").innerHTML = 'Tiempo de ejecucion = ' +
punchtime
                        if(punchtime <= MinUppercuttime){
                        MinUppercuttime = punchtime
                        jQuery('#recordIMG').show()
                        }
                        else{
                        jQuery('#recordIMG').hide()
                        }
                }

                if (arrayX.length > 7 && punchtype == 0){
                otherCounter++
                punchCounter++
                punch = 'Otro'
                punchtype = 3
```

```
                document.getElementById("Img").src = "images/OTRO.png"

                displayValue('Punchinfo', 'Numero de golpes = ' +  otherCounter + ' de ' +
punchCounter + '<br/>Porcentaje = ' +
parseFloat(Math.round(otherCounter/punchCounter*100)).toFixed(0) + '%')

                document.getElementById("ExTime").innerHTML = 'Tiempo de ejecucion = ' +
punchtime

                        if(punchtime <= MinOthertime){

                        MinOthertime = punchtime

                        jQuery('#recordIMG').show()

                        }

                        else{

                        jQuery('#recordIMG').hide()

                        }

                }

                displayValue('Punchcounter','Tipo de golpe: ' + punch + '<br/>Tiempo de ejecucion: ' +
punchtime + 's' + '<br/>Straight = ' + straightCounter + '<br/>Uppercut = ' + uppercutCounter +
'<br/>Otros = ' + otherCounter + '<br/>Golpes Totales = ' + punchCounter )

        }


                function displayValue(elementId, value)

        {

                document.getElementById(elementId).innerHTML = value

        }


        document.addEventListener('deviceready', initialiseSensortag, false)


        </script>

</body>

</html>
```

7. Posibles ampliaciones

En cuanto al hardware una opción sería incorporar un giroscopio que ayudaría a diferenciar el tipo de golpe realizado con mayor consistencia al incorporar nuevas variables de comparación. En el proceso de toma de datos inicialmente se iba a incluir el right hook de corto alcance pero su similitud con la gráfica del uppercut hizo que se descartara su inclusión. Un giroscopio quizá podría ayudar a distingirlos.

En cuanto a funcionalidad podría diseñarse un sistema vinculando 2 Smartfights, uno tomando medidas del brazo derecho y otro del izquierdo. Con este sistema y una base de datos completa sería posible realizar análisis más completos de rendimiento y añadir nuevas características como un contador de combos (Jab, Jab), (Jab, Cross), (Jab, Cross, Hook, Cross)…

Es posible aumentar el número de modalidades de deportes de contacto soportadas por el Smartfight si se obtienen y analizan sus gráficas de aceleración. Por lo que podría incorporarse a la interfaz de usuario un selector del arte marcial que se desea complementar.

Otra posibilidad es enfocarlo a un ámbito deportivo fuera de los deportes de contacto, como ejemplo, en el baloncesto podría detectar pases y distintos lanzamientos.

8. BIBLIOGRAFÍA

Ventas Wearables:

http://www.forbes.com/sites/paullamkin/2016/02/03/smartwatch-sales-to-soar-apparently/#3acd27ba22d8

http://www.gartner.com/newsroom/id/3198018

http://www.idtechex.com/research/reports/wearable-technology-2015-2025-technologies-markets-forecasts-000427.asp?viewopt=showall

http://fitness-trackers.specout.com/

Ventas Smartphone:

http://www.idc.com/getdoc.jsp?containerId=prUS41216716

http://www.xatakamovil.com/movil-y-sociedad/espana-territorio-smartphone

Portable Punch Analyzer:

https://www.amazon.co.uk/LUCERIS-Portable-Punch-Analyzer/dp/B00JFM3RVI

Baterías:

https://es.wikipedia.org/wiki/Bater%C3%ADa_el%C3%A9ctrica

http://batteryuniversity.com/learn/article/is_lithium_ion_the_ideal_battery

http://www.digikey.com/en/articles/techzone/2012/sep/a-designers-guide-to-lithium-battery-charging

https://www.maximintegrated.com/en/app-notes/index.mvp/id/3241

Microcontroladores:

https://www.futureelectronics.com/en/Microcontrollers/microcontrollers.aspx

https://www.arduino.cc/en/Guide/Introduction

http://www.ti.com/product/CC2541

Acelerómetros:

http://www.dimensionengineering.com/info/accelerometers

http://www.livescience.com/40102-accelerometers.html

http://www.analog.com/en/products/mems/mems-accelerometers/adxl375.html

http://www.analog.com/en/products/landing-pages/001/accelerometer-specifications-definitions.html

Giroscopio:

https://learn.sparkfun.com/tutorials/gyroscope/how-a-gyro-works

Bluetoth:

https://en.wikipedia.org/wiki/Bluetooth

https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics

Wifi:

http://www.wifinotes.com/wifi-features.html

Android:

https://source.android.com/index.html

http://www.androidcentral.com/

https://www.android.com/intl/es_es/

Circuito RF:

http://www.johansontechnology.com/2450bm15a0002-matched-balun-for-t-i-253x-family-chipsets.html

Guardias boxeo:

http://www.livestrong.com/article/418454-different-types-of-boxing-stances/

https://www.youtube.com/channel/UCrYqUsxL8UbAbk6wP-9_tig

Estudios deportivos:

http://bjsm.bmj.com/

http://www.researchgate.net/publication/19779430_Kinematic_analysis_of_human_upper_extremity_movements_in_boxing

http://www.researchgate.net/publication/7438738_Concussion_in_professional_football_comparison_with_boxing_head_impacts--part_10

Sensortag:

http://processors.wiki.ti.com/index.php/CC2541_SensorTag

http://www.ti.com/tool/CC2541SENSORTAG-RD

http://processors.wiki.ti.com/index.php/SensorTag_User_Guide

Evothings:

https://evothings.com/

https://evothings.com/quick-guide-to-making-a-mobile-app-for-the-ti-sensortag-using-javascript/

https://evothings.com/doc/examples/ble-ti-sensortag-sensors.html

Apache cordova:

https://cordova.apache.org/

https://cordova.apache.org/docs/es/latest/guide/overview/

https://evothings.com/doc/build/cordova-install-windows.html

https://evothings.com/doc/build/cordova-guide.html

http://cordova.apache.org/plugins/?q=bluetooth

http://developer.android.com/intl/es/tools/help/adb.html

Publicación app Android:

http://ionicframework.com/docs/guide/publishing.html

Salarios:

http://www.seg-social.es/Internet_1/Trabajadores/CotizacionRecaudaci10777/Basesytiposdecotiza36537/index.htm

http://www.tusalario.es/main/salario/comparatusalario

Fabricación PCB:

http://www.2cisa.com/index.php?com=presupuestos