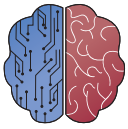
UNIVERSIDAD DE BURGOS

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de la Salud



INGENIERÍA DE LA SALUD

TFG del Grado en Ingeniería de la Salud

título del TFG
Documentación Técnica

Presentado por Naiara Gadea Rodríguez Gómez
en la Universidad de Burgos

1 de julio de 2023

Tutor: Pedro Luis Sánchez Ortega

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

Para el correcto desarrollo del proyecto se seguirá una planificacion temporal y se desarrollara una planificacion economica para ver el coste economico aproximado del producto desarrollado a lo largo del proyecto. Además, se dispondrá de un apartado donde consultar la viabilidad legal del desarrollo del proyecto.

A.2. Planificación temporal

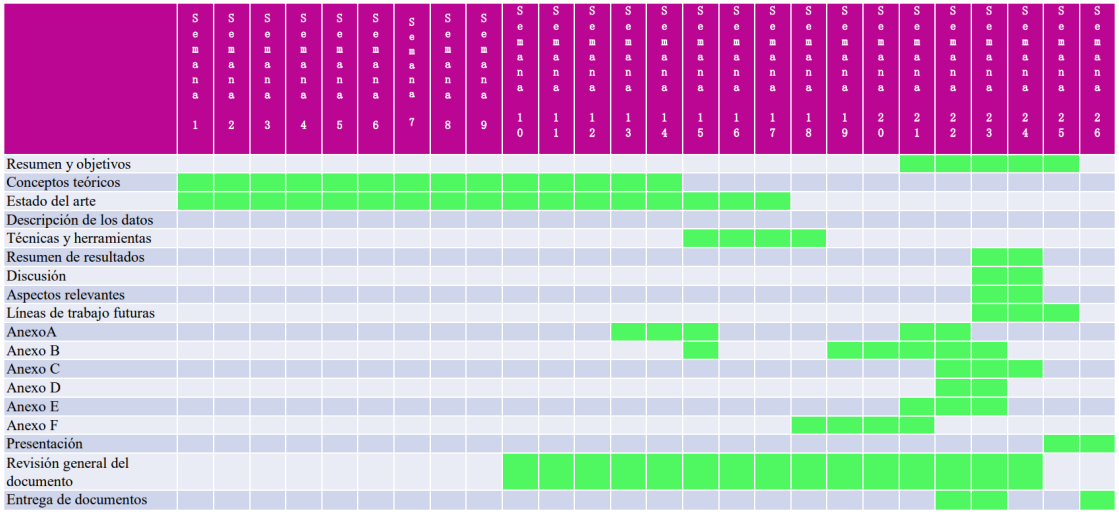


Figura A.1: Planificación temporal seguida para la realización de este proyecto.

En la planificación temporal se puede ver cada apartado de la memoria y los anexos desarrollados a lo largo de las semanas.

A.3. Planificación económica

Para obtener una buena planificación económica se deberán identificar los costes tanto personales como materiales y de desarrollo y los ingresos relacionados con el prototipo.

Los costes personales se calcularán en función del salario medio de un Ingeniero de la Salud durante un periodo aproximado de 6 meses durante media jornada[1], lo que suma unas 540 horas de trabajo totales. Teniendo en cuenta que el sueldo neto medio de un ingeniero con menos de 3 años de experiencia ronda los 1392,8€ al mes a jornada completa [1], a media jornada cobrará unos 696,4€ al mes. Por lo tanto, los costes de personal se podrían resumir en la siguiente tabla.

	Costes
Sueldo neto mensual	696,4€
Sueldo bruto mensual	799,79€
Total en los 6 meses	4798,74€

Tabla A.1: Resumen de costes personales.

Como primeros gastos relacionados con los equipos de desarrollo, el ordenador que se ha empleado y la licencia de Office que se ha empleado para la realización de algunas partes del proyecto, con una amortización[2] [2] en unos 4 años. No se incluyen gastos de otros softwares, porque se han empleado herramientas de código abierto y gratuitas.

	Costes	**Amortización**
Ordenador ASUS	700€	105€
Office 365 personal anual	69€	10,35€
Total en los 6 meses	769€	115,35€

Tabla A.2: Resumen de costes de equipos de desarrollo amortizados en 4 años.

[1]La media jornada constará de unas 4 horas al día durante 5 días a la semana, es decir, 20 horas semanales.

[2]La amortización se calcula con la diferencia del coste del equipo y del coste residual a los 4 años (un 40 % del coste del equipo) todo ello divido entre el periodo de amortización, que en este caso serán unos 4 años.

Para el cálculo del gasto de los materiales del desarrollo del prototipo, se tienen en cuenta los costes de los componentes y los gastos de realización del prototipo, mientras que para obtener los posibles ingresos se tendrá en cuenta el beneficio, gastos imprevistos e I+D del producto, para posibles mejoras en e futuro. La suma de gastos e ingresos nos devolverá el precio final aproximado del dispositivo. No se incluyen los gastos personales, ni de equipos, estos gastos se incluirían en caso de producción del dispositivo final.

	Cálculos	**Versión 1**	**Versión 2**
Gastos de los componentes	Suma de los precios de cada componente del producto	27€	32.5€
Gastos de producción	10 % del precio de los componentes	2.7€	3.25€
Ingresos destinados a beneficio	5 % del total de gastos	1.48€	1.79€
Gastos imprevistos e I+D	10 % del total de gastos	2.97€	3.58€
Precio Total	Suma de los gastos e ingresos	**34.15€**	**41.10€**

Tabla A.3: Resumen de gastos y precio total del producto

El prototipo de la versión 1, en la que se emplea el sensor SW520D[3], tiene un coste final aproximado[3] de unos 34.15€, precio que podría ser menor al crear nuestro propio microcontrolador o utilizar una alternativa similar a arduino, puesto que es el elemento que más aumenta el precio de la solución, siendo el precio del resto de los componentes aproximadamente unos 3€.

Por el contrario el prototipo de la versión 2, en el que se emplea el módulo MPU-6050[4, 5], tiene un coste final aproximado de unos 41.10€, precio que también podría disminuir al crear nuestro propio microcontrolador o utilizar una alternativa a arduino, ya que sin el microcontrolador Arduino el precio ronda los 8.5€.

[3]La planificación económica será variable en el tiempo y durante el desarrollo del producto, por lo que se trata de precios totales aproximados. Igual para el prototipo versión 2.

Desglose de los precios de los componentes del prototipo Versión 1

Se han tenido en cuenta los precios más bajos encontrados de cada componente necesario. Estos precios incluyen el porcentaje de IVA y se han calculado en base a los precios del proveedor Amazon España[6].

- Arduino UNO R3: 24€
- Resitencias (330 Ω, 2x220 Ω, 33 Ω, 1000 Ω): 0.05€
- Zumbador pasivo: 0.25€
- Motor de vibración: 1€
- Transistor: 0.05€
- SW520D: 0.5€
- Led azul: 0.02€
- Pulsador: 0.05€
- Otros elementos variados: 1€

Desglose de los precios de los componentes del prototipo Versión 2

Se han tenido en cuenta los precios más bajos encontrados de cada componente necesario. Estos precios incluyen el porcentaje de IVA y se han calculado en base a los precios del proveedor Amazon España[6].

- Arduino UNO R3: 24€
- Resitencias (2x330 Ω, 2x220 Ω, 33 Ω, 1000 Ω): 0.06€
- Zumbador pasivo: 0.25€
- Motor de vibración: 1€
- Transistor NPN: 0.05€
- MPU-6050: 6€
- Led azul: 0.02€

- 2xPulsador: 0.10€
- Otros elementos variados: 1€

A.4. Viabilidad legal

Se debe tener en cuenta en todo momento que el dispositivo sea completamente seguro y no afecte negativamente al usuario. Para ello, existen legislaciones específicas a cada fase del desarrollo y comercialización del producto que se deben cumplir para obtener un dispositivo seguro y regulado.

Se pueden diferenciar 3 fases, una primera fase de creación de la idea, diseño y desarrollo y realización de pruebas, una segunda fase de comercialización y la última fase de posventa, donde se incluyen las demandas y la gestión de los datos de los usuarios.

Durante la primera fase de creación de la idea, diseño y desarrollo del producto y realización de pruebas, todos los movimientos que se realicen se deberán ajustar a las siguientes legislaciones:

- Ley 24/2015[7], Ley de Patentes, dónde se regula todo lo relacionado con la protección de invenciones empleando patentes, desde el registro de las patentes, invenciones patentables, el derecho a la patente y los procedimientos para pedir una patente.
- Real Decreto Legislativo 1/1996[8] relativo Ley de Propiedad Intelectual que regulariza la protección del derecho de autor y de derechos similares.
- Los productos sanitarios se rigen por la Agencia española de medicamentos y productos sanitarios (AEMPS)[9]. En este proyecto nos interesan especialmente el Real Decreto 1591/2009[10] que regula todo lo relativo a los productos sanitarios, desde su desarrollo a su venta, y el Real Decreto 437/2002[11] establece las pautas para la concesión de licencias de fabricación y desarrollo de productos sanitarios.
- Reglamento de la UE 2017/745[12] de Productos Sanitarios de la Union Europea este reglamento establece requisitos y regula la comercialización de productos sanitarios en la Unión Europea, con el fin de garantizar un dispositivo de calidad, eficaz y completamente seguro.
- Además, durante todo el desarrollo del producto se deberá cumplir con la normativa laboral española[13], que incluye leyes y reglamentos como pueden

ser el Estatuto de los Trabajadores, la Ley de Prevención de Riesgos Laborales o la Ley de Igualdad.

Si se consigue crear el dispositivo en base a todas las leyes anteriores y se quisiera sacar a mercado se deberán cumplir también con los siguientes requisitos legales:

- La Ley 34/2002[14] de Servicios de la Sociedad de la Información y de Comercio Electrónico, en caso de que se realice una tienda web oficial de comercialización del dispositivo.
- Además, se deberan tener en cuenta otras leyes[15] como la Ley 7/1996[16] de Ordenación del Comercio Minorista.

Por último, si el dispositivo se ha puesto a la venta se debe pensar en los requisitos legales que se necesitarán cumplir a partir del momento de la primera venta. Alguno de estos requisitos serán:

- Ley Orgánica 3/2018[17] de Protección de Datos Personales y garantía de los derechos digitales. Para poder proteger cualquier información que identifique a una persona, de forma confidencial. Además, el usuario debe estar correctamente informado del tratamiento de sus datos, ademas el acceso al tratamiento de sus datos debe ser claro y accesible.

  El usuario tendrá derecho al acceso de sus datos, derecho de rectificación y supresión de sus datos, derecho a la limitación del tratamiento de sus datos, derecho a la portabilidad de sus datos y el derecho a oponerse al tratamiento de sus datos. Por todo ello el tratamiento de sus datos debe ser tras la confirmación clara del consentimiento informado del tratamiento de sus datos.
- Reglamento UE 2016/679[18] relativo a Protección de las Personas Físicas en lo que respecta al tratamiento de datos personales y circulación de estos Datos. Donde se define que se debe garantizar la protección de los datos con los que se trabaja, además de notificar brechas de seguridad o exposición de datos al usuario.

Además el dispositivo deberá contar con un certificado CE[19], que garantizará que el dispositivo cumple con los requisitos de seguridad, protección y sanidad europeos. Una vez se obtenga el certificado el dispositivo podrá ser comercializado legalmente en la Unión Europea.

Apéndice B

Documentación de usuario

B.1. Requisitos software y hardware para ejecutar el proyecto.

En esta sección del anexo se definen varios requisitos tanto de software como de hardware que requiere el dispositivo objeto de este trabajo. Cuantos más requisitos cumpla el prototipo más exitoso resultará el dispositivo.

Requisitos funcionales

RF-01	**Aplicación**
Descripción	La solución deberá contar con una aplicación, ya sea una aplicación de escritorio, web o móvil, para simplificar la experiencia de uso y la visualización de resultados por parte del usuario.
Importancia	Alta, es la base de la visualización del seguimiento de la persona que utiliza el dispositivo.
Prioridad	Alta

Tabla B.1: Requisito Funcional 1 'Aplicación'

RF-02	**Iniciar grabación**
Descripción	La solución deberá contar con una opción de grabación, con la cual el profesional o el usuario tendrán la posibilidad de comenzar y finalizar el registro de la postura. Los resultados durante la grabación se almacenarán en la plataforma para su análisis.
Importancia	Alta, ya que la grabación de las respuestas permitirá al profesional analizarlas de forma detallada con el objetivo de obtener conclusiones y determinar el grado y evolución de la afectación.
Prioridad	Alta

Tabla B.2: Requisito Funcional 2 'Iniciar grabación'

RF-03	**Identificación de perfiles**
Descripción	La aplicación debe ser capaz de diferenciar a diferentes perfiles, en el caso de uso de una organización o un profesional, y una única identificación en el caso de que se trate de un usuario particular.
Importancia	Media, una vez se obtenga la base del dispositivo y su funcionamiento se puede dividir a los usuarios entre profesionales o particulares, con distintas funciones para cada uno de ellos.
Prioridad	Media

Tabla B.3: Requisito Funcional 3 'Identificación de perfiles'

RF-04	**Detección de la postura**
Descripción	La solución deberá ser capaz de detectar los cambios en la postura. Para ello, se deberá implementar un algoritmo que filtre en función de los datos en crudo recogidos, una postura correcta o incorrecta. Esta medición se podría obtener en forma de 'porcentaje de buena postura'.
Importancia	Alta, dado que es la base que permitirá definir si la persona lleva una buena postura o no, y en base a ello, realizar la comunicación correspondiente y obtener las estadísticas necesarias para la toma de decisiones.
Prioridad	Alta

Tabla B.4: Requisito Funcional 4 'Detección postural'

RF-05	**Comunicar una postura incorrecta**
Descripción	La solución debe poder comunicar mediante, vibración, sonido u otra manera una mala postura continuada durante un periodo de tiempo predefinido.
Importancia	Alta, es necesario que el usuario conozca en todo momento su situación, para poder corregir su postura cuando sea necesario.
Prioridad	Media

Tabla B.5: Requisito Funcional 5 'Comunicar una postura incorrecta'

RF-06	**Realizar seguimiento**
Descripción	La información registrada por el dispositivo debe quedar almacenada para valorar y evaluar la postura del paciente, con el fin de modificar o no el tratamiento o fisioterapia o tomar otro tipo de decisiones. La visualización de la información recogida se reflejará en forma de gráficos y tablas. Esto permitirá analizar la información de manera clara y sencilla
Importancia	Alta, ya que será clave para la toma de decisiones por parte del especialista en cuanto a la personalización del tratamiento y rehabilitación.
Prioridad	Alta

Tabla B.6: Requisito Funcional 6 'Realizar seguimiento'

RF-07	**Manual de usuario**
Descripción	La solución deberá incluir unas instrucciones que se entreguen al usuario que lo vaya a utilizar. Esto supone un apoyo durante todo el proceso de uso del dispositivo y de la aplicación por parte del usuario.
Importancia	Media, puesto que supone un apoyo para el usuario que lo utilice.
Prioridad	Baja

Tabla B.7: Requisito Funcional 7 'Manual de usuario'

RF-08	Batería
Descripción	El dispositivo debe disponer de una batería para poder utilizarlo de forma telemática. Además, la batería del dispositivo debe ser suficiente para el uso previsto.
Importancia	Media, se debe incluir para mayor comodidad y libertad del paciente al utilizar el dispositivo.
Prioridad	Media

Tabla B.8: Requisito Funcional 8 'Batería'

Requisitos no funcionales

- **Accesibilidad**: la aplicación debe ser accesible para el mayor grupo de personas posible, tengan o no algún tipo de discapacidad.
- **Seguridad**: el dispositivo electrónico debe ser seguro y la información manejada en la aplicación debe estar protegida.
- **Compatibilidad**: la aplicación debe ser compatible con distintos dispositivos.
- **Eficiencia**: la aplicación debe permitir al usuario lograr sus objetivos, con un coste computacional y temporal bajo.
- **Efectividad**: la aplicación debe cumplir con exactitud los requisitos funcionales.
- **Errores**: la aplicación debe presentar una tasa de error baja, además, debe mostrar posibles soluciones en caso de anomalías.
- **Usabilidad**: tanto el uso del dispositivo electrónico como de la aplicación debe ser sencillo, es decir, se debe poder usar de forma intuitiva.
- **Memorabilidad**: tanto el funcionamiento del dispositivo electrónico como de la aplicación debe ser fácil de recordar, tras no haberlos utilizado durante un tiempo.
- **Satisfacción**: el usuario debe estar satisfecho con el dispositivo electrónico y la aplicación, tanto por su comodidad, estética y usabilidad.

B.2. Instalación / Puesta en marcha

Como se han creado dos versiones los requisitos de instalación y puesta en marcha serán diferentes.

Para la primera versión del prototipo dónde se emplea el sensor SW520D[3], es necesario tener instalado el software de Arduino IDE[20, 21, 22], sin necesidad de instalar cualquier librería adicional.

Mientras que para la segunda versión no solo se ha de tener instalado el software de Arduino IDE, sino que se deberá igualmente tener instaladas las librerías I2Cdev[23], MPU6050[24] y Wire[25]:

Una vez tenemos instalado el software y las librerías necesarias, el siguiente paso será el de montar el prototipo siguiendo los esquemas especificados en las figuras *Figura B.1* y *Figura B.3*.

Incluir El repositorio de GitHub

Si se ha montado la primera versión del prototipo se deberá descargar el programa que se encuentra en el repositorio de GitHub[26] bajo el nombre de 'PrototipoV1.ino'. En el caso de haber montado la segunda versión, se deberá descargar el programa del repositorio de GitHub[26] con el nombre de 'PrototipoV2.ino'. Estos programas son los que abriremos y se enviarán al microprocesador Arduino UNO R3[22], el prototipo montado, que se encontrará conectado con el cable USB al ordenador. Una vez cargados los programas se podrá realizar el control postural, de forma que se consiga una ejecución exitosa.

Versión 1, empleando el sensor SW520D

Como primera versión se ha creado un prototipo empleando Arduino[21] y el sensor SW520D[3].

Se ha incluido un botón de encendido, un led de color azul que indica que el dispositivo se encuentra encendido, un zumbador pasivo que actuará como señal sonora y un motor de vibración que actuará como aviso vibratorio.

Cuando el sensor detecta que la persona se ha inclinado hacia delante, se detecta una mala postura y salta una alerta sonora (melodía modificable empleando el zumbador) y vibratoria (motor de vibración).

Se pueden observar los componentes y las conexiones realizadas en esta versión en los diagramas *Figura B.1* y *Figura B.2*.

MOVER código al anexo C¿?

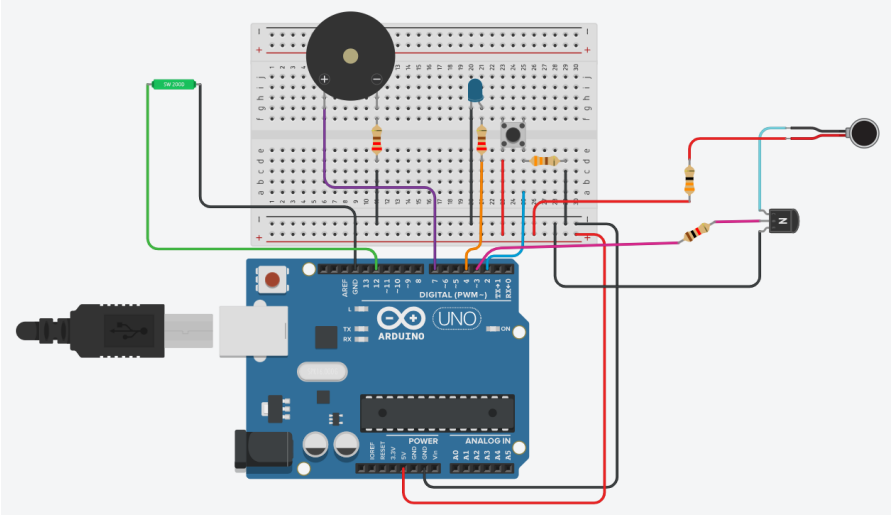


Figura B.1: Diagrama de la primera versión del prototipo, empleando el sensor SW520D

El código empleado para el funcionamiento de esta primera versión ha sido el siguiente:

```
1  // Led simple + Boton + Zumbador + Tilt + Motor Vibracion
2  // Naiara Gadea Rodriguez Gomez
3  //
4
5  int led = 4; // Seleccion del pin del led (pin digital)
6  int  boton = 2; // Seleccion del pin del boton (pin digital
       )
7  int zum = 7;// Seleccion del pin del zumbador
8  int tilt = 12;// Seleccion del pin del sensor SW250D
9  int motor = 3; //Seleccion del pin del motor de vibracion
10
11 int estado; // Estado del boton
12
13 void setup() {
14   Serial.begin(9600);
15   pinMode(led, OUTPUT); // inicializacion del pin led.
16   pinMode(boton, INPUT); // inicializacion del pin del
       boton.
17   pinMode(zum, OUTPUT); // inicializacion del pin del
       zumbador
```

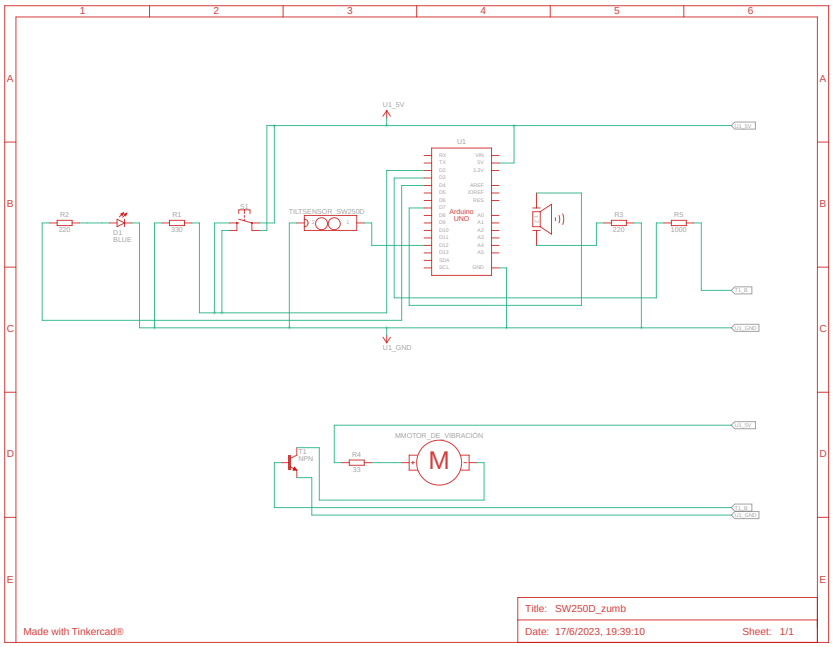


Figura B.2: Diagrama de las realizadas para la implementación de la primera versión del prototipo.

```
18   pinMode(tilt, INPUT); // inicializacion del pin del
      sensor tilt
19   digitalWrite(tilt , HIGH); // Sensor tilt
20   pinMode(motor, OUTPUT);// inicializacion del pin del
      motor de vibracion.
21 }
22
23 void loop() {
24   Serial.println(digitalRead(tilt));// Comprobacion en el
      Serial Monitor.
25   if (estado == LOW && digitalRead(boton)){
26     // Si se presiona el boton se enciende el dispositivo
27     digitalWrite(led, HIGH); // Encendido
28       delay(1000); // Durante 1 segundo (1000 ms)
29       estado = HIGH; // Cambia el estado del boton a
      encendido.
```

```
30
31   }
32   if(estado == HIGH){
33     // Si el sensor tilt hace contacto, el usuario tiene
     una mala postura y el dispositivo manda un aviso,
     musical o de vibracion.
34     if (digitalRead(tilt))  {
35       // Vibracion intermitente
36       digitalWrite(motor, HIGH); // vibracion
37       delay(500);  // delay 0.5 seconds
38       //digitalWrite(motor, LOW);  // stop vibracion
39       //delay(500); //wait 0.5 seconds.
40
41       melodia(); // En este caso es un aviso sonoro, pero
     teniendo un motor de vibracion se puede utilizar un
     aviso vibratorio.
42
43     }  else  {
44       // Si no hay contacto con el sensor tilt, no suena la
     melodia
45       noTone(zum); // El zumbador ya no emite ruido
46       //delay(3000);
47       digitalWrite(motor, LOW);// Paramos el motor
48     }
49
50     // Si se presiona el boton se apaga el dispositivo
51     if (digitalRead(boton)){
52       digitalWrite(led, LOW); // Apagado
53       delay(1000); // Durante 1 segundo (1000ms)
54       estado = LOW; // Cambia el estado del boton a apagado
     .
55     }
56
57   }
58
59 }
60
61 // Definimos las notas
62 int Do = 261;
63 int Re = 293;
```

```
64 int Mi = 329;
65 int Fa = 349;
66 int Sol = 392;
67 int La = 440;
68 int Si = 493;
69
70 void melodia(){
71   // Escala de musica con el zumbador, se puede modificar
     en funcion del gusto del programador.
72     tone(zum, Fa, 500);
73     delay(700);
74     tone(zum, Sol, 500);
75     delay(700);
76     tone(zum, Sol, 500);
77     delay(700);
78     tone(zum, La, 1000);
79     delay(1700);
80     tone(zum, Sol, 500);
81     delay(700);
82     tone(zum, Fa, 500);
83     delay(700);
84     tone(zum, Sol, 500);
85     delay(700);
86     //tone(zum, Do, 1000);
87     //delay(1700);
88     //tone(zum, Fa, 500);
89     //delay(700);
90     //tone(zum, La, 500);
91     //delay(700);
92     //tone(zum, Fa, 500);
93     //delay(700);
94     //tone(zum, Re, 1000);
95     //delay(1700);
96
97 }
```

Durante el desarrollo de esta primera versión del proyecto donde se empleaba el sensor SW520D[3], el más sencillo, se observó que sí que es capaz de realizar un control postural, pero no se trata de un dispositivo de gran precisión, ya que es muy sensible a vibraciones.

Aunque este sensor no cumple con el requisito de la precisión esta primera versión ha permitido analizar las posibilidades de mejora y las ventajas que supone trabajar con un sensor de mayor precisión que se emplea en la fase 2.

Versión 2, empleando el sensor MPU-6050

La segunda versión se ha creado en base al microprocesador Arduino[21] y el módulo MPU6050[4, 5].

Esta segunda versión mantiene el botón de encendido, el led de color azul que indica el estado del dispositivo, el zumbador pasivo y el motor de vibración que realizarán el feedback a través de un aviso sonoro o vibratorio.

Además, esta versión cuenta con un botón de calibración para mayor precisión. Este botón deberá presionarse cuando el dispositivo se encienda cuando se coloque por primera vez en la espalda del usuario con una postura correta o en el caso de que el usuario sea consciente de que el dispositivo no esté funcionando correctamente.

El sensor detecta que la persona se ha inclinado más del ángulo umbral[1] respecto a su posición inicial correcta, se detecta una mala postura y se ofrece el biofeedback sonoro (melodía modificable empleando el zumbador) y vibratorio (motor de vibración) para que el usuario modifique su postura. Una vez la persona recupera su postura natural correcta el dispositivo deja de sonar y vibrar, indicando que se encuentra en una postura correcta.

Se pueden observar los componentes y las conexiones realizadas en esta versión en los diagramas *Figura B.3* y *Figura B.4*.

MOVER código al anexo C¿?

[1]En caso de que el usuario no hay indicado un ángulo umbral, se tomará como valor por defecto 15º

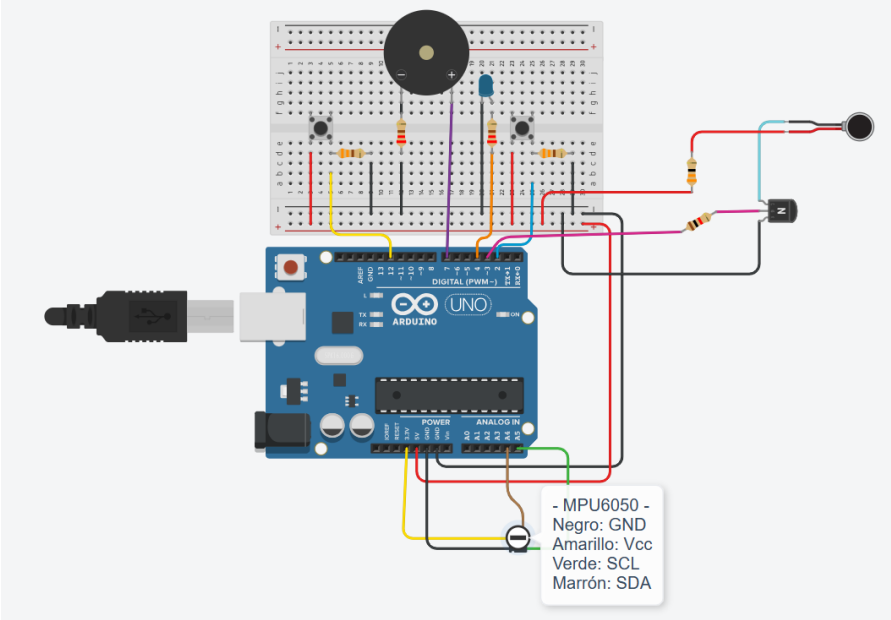


Figura B.3: Diagrama de la segunda versión del prototipo, empleando el módulo MPU-6050

El código empleado para el funcionamiento de esta segunda versión es el siguiente:

```
1 // Led simple + Boton + Zumbador + MPU6050 + (Motor de
    vibracion)
2 // Naiara Gadea Rodriguez Gomez
3 //
4 // Nota Si se tiene motor de vibracion descomentar las
    lineas de codigo correspondientes.
5
6 // Librerias I2C para controlar el sensor mpu6050
7 // La libreria MPU6050.h necesita I2Cdev.h y la libreria
    I2Cdev.h necesita Wire.h
8 #include "I2Cdev.h"
9 #include "MPU6050.h"
10 #include "Wire.h"
11
12
13 // La direccion del MPU6050 puede ser 0x68 o 0x69,
    dependiendo del estado de AD0. Si no se especifica, 0x68
     estara implicito.
```

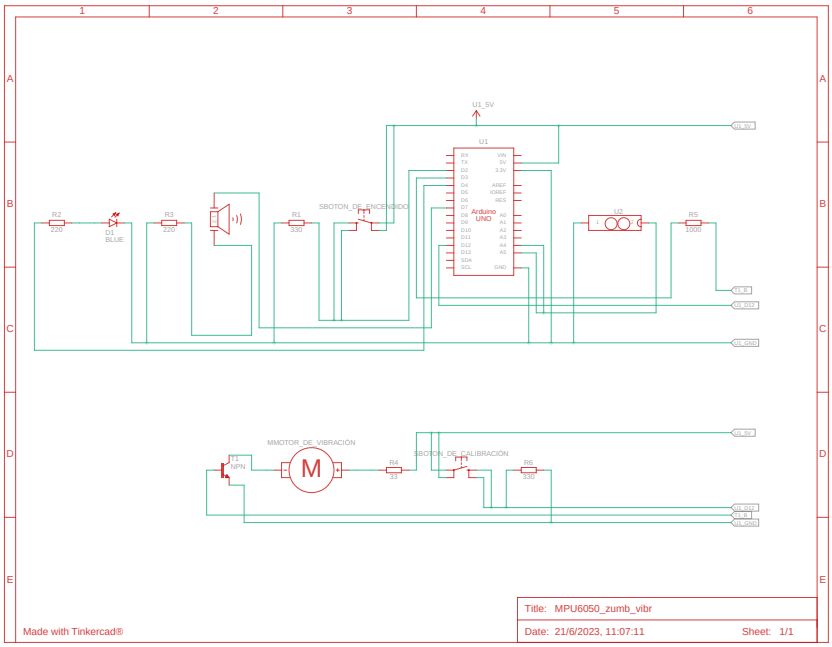


Figura B.4: Diagrama de las conexiones realizadas para la implementación de la segunda versión, siendo el módulo MPU-6050, U2.

```
14 // Se crea la variable del sensor. En este caso se esta
     trabajando con 0x68. Si se quiere trabajar con 0x69 hay
     que poner MPU6050 sensor(0x69)
15 MPU6050 sensor(0x68);
16
17 // Valores en crudo del acelerometro y giroscopio en los
     ejes x,y,z
18 int ax, ay, az; // acelerometro
19 int gx, gy, gz; // giroscopio
20
21 float accel_ang_x, accel_ang_y, accel_ang_z; // Variables
     correspondientes a los angulos de inclinacion. Son los
     que nos interesan principalmente.
22
23 int  boton1 = 2; // Seleccion del pin del boton (pin
     digital)
```

```
24 int estado1; // Estado del boton ON/OFF
25 int led = 4; // Seleccion del pin del led (pin digital)
26 int zum = 7;// Seleccion del pin del zumbador
27 // int motor = 3; //Seleccion del pin del motor -Cuando se
      tenga el motor-
28 int boton2 = 12; //Boton de calibrado
29 int estado2;//Estado del boton de calibrado
30
31 // Otras variables
32 bool calibrar = false; // Variable que indica cuando seguir
       o no calibrando.
33 bool origen = true; // Para guardar los datos del primer
      valor tras la calibracion y comparar la diferencia de
      angulo de inclinacion.
34
35 // Variables usadas para filtro y promedio durante la
      calibracion.
36 long f_ax,f_ay, f_az;
37 int p_ax, p_ay, p_az;
38 long f_gx,f_gy, f_gz;
39 int p_gx, p_gy, p_gz;
40 int counter=0;
41
42 //Valor de los offsets.
43 int ax_o,ay_o,az_o;
44 int gx_o,gy_o,gz_o;
45
46 float first_x, first_y, first_z; // Si no funciona
      correctamente desde el principio, hay que calibrar, o
      inicializar desde el principio.
47
48
49 // Variables para obtener los valores medios cada cierto
      tiempo.
50 float sum_ax, sum_ay, sum_az;
51 float sum_gx, sum_gy, sum_gz;
52 float media_ax, media_ay, media_az;
53 float media_gx, media_gy, media_gz;
54 int cont = 0;
55
```

```
56 // Angulo umbral y tiempo de aviso
57 int ang_aviso = 15;
58 int t_aviso = 5;
59
60 // Definimos las notas que se emplearan para la melodia de
     aviso del zumbador pasivo.
61 int Do = 261;
62 int Re = 293;
63 int Mi = 329;
64 int Fa = 349;
65 int Sol = 392;
66 int La = 440;
67 int Si = 493;
68
69
70 void setup() {
71   Serial.begin(9600); // Inicializacion del monitor.
72   pinMode(led, OUTPUT); // Inicializacion del pin digital
      led.
73   pinMode(boton1, INPUT); // Inicializacion del pin digital
       del boton.
74   pinMode(boton2,INPUT); // Inicializacion del pin boton de
       calibrado.
75   pinMode(zum, OUTPUT); // Inicializacion del pin digital
      del zumbador.
76   //pinMode(motor, OUTPUT);// Inicializacion del pin
      digital del motor de vibracion.
77
78   Wire.begin(); // Inicializacion de la comunicacion I2C
79   sensor.initialize(); // Inicializacion del sensor MPU6050
80
81   // Al inicializar el sensor los rangos seran:
82   // Acelerometro: -2g a +2g
83   // Giroscopio: -250deg/sec a +250deg/sec
84
85   // Se comprueba que se ha inicializado correctamente el
      sensor.
86   if (sensor.testConnection()) Serial.println("Sensor
      iniciado correctamente");
87   else Serial.println("Error al iniciar el sensor");
```

```
88
89  // Definicion de offsets iniciales iniciales
90  ax_o=sensor.getXAccelOffset();
91  ay_o=sensor.getYAccelOffset();
92  az_o=sensor.getZAccelOffset();
93  gx_o=sensor.getXGyroOffset();
94  gy_o=sensor.getYGyroOffset();
95  gz_o=sensor.getZGyroOffset();
96
97  Serial.println("Offsets iniciales:");
98  Serial.print(ax_o); Serial.print("\t");
99  Serial.print(ay_o); Serial.print("\t");
100 Serial.print(az_o); Serial.print("\t");
101 Serial.print(gx_o); Serial.print("\t");
102 Serial.print(gy_o); Serial.print("\t");
103 Serial.print(gz_o); Serial.println("\t");
104
105
106 Serial.println("Introduce un angulo umbral(deg). ");
107 while (Serial.available()==0){
108 }
109 int num = Serial.parseInt();
110
111 if (num!=0){
112   Serial.println("Entra");
113   ang_aviso = num;
114 }
115 Serial.println("Angulo umbral: "); Serial.println(
    ang_aviso);
116 Serial.println("Introduce un tiempo de respuesta(sec). ")
    ;
117 delay(5000);
118 while (Serial.available()==1){
119 }
120
121 int num2 = Serial.parseInt();
122
123 if (num2!=0){
124   t_aviso = num2;
125 }
```

```
126   Serial.println("Tiempo de espera para aviso: "); Serial.
      println(t_aviso);
127
128 }
129
130 void loop() {
131   if (estado1 == LOW && digitalRead(boton1)){
132     // Si se presiona el boton se ENCIENDE EL DISPOSITIVO y
      por lo tanto se enciende el led.
133     digitalWrite(led, HIGH); // Encendido
134     //delay(1000); // Durante 1 segundo (1000 ms)
135     estado1 = HIGH; // Cambia el estado del boton a
     encendido.
136
137   }
138   if(estado1 == HIGH){
139     // Si el dispositivo se encuentra encendido
140     // Lectura de las aceleraciones y velocidades angulares
      y guardado en sus variables correspondientes.
141     sensor.getAcceleration(&ax, &ay, &az);
142     sensor.getRotation(&gx, &gy, &gz);
143
144     if (digitalRead(boton2) && estado2==LOW){
145       // Si se presiona el boton de calibracion, cambia el
     estado para indicar que hay que calibrar.
146       calibrar = true;
147     }
148
149     if (calibrar){
150       // Si se quiere calibrar se calibra.
151       calibracion();
152       origen = true; // Se modifican los angulos iniciales
     de inclinacion.
153     }else{
154
155       if (cont < t_aviso/0.5){
156         lecturas(); // Se realiza la lectura de los valores
      cada t_aviso tiempo de aviso)
157         sum_ax = sum_ax + accel_ang_x;
158         sum_ay = sum_ay + accel_ang_y;
```

```
159         sum_az = sum_az + accel_ang_z;
160
161         cont++;
162
163       } else{
164         media_ax = sum_ax/(t_aviso/0.5);
165         Serial.print("Valor medio de inclinacion en X: ");
    Serial.println(media_ax);
166         media_ay = sum_ay/(t_aviso/0.5);
167         Serial.print("Valor medio de inclinacion en Y: ");
    Serial.println(media_ay);
168         media_az = sum_az/(t_aviso/0.5);
169
170
171         // reestablecemos los sumatorios
172         sum_ax = 0;
173         sum_ay = 0;
174         sum_az = 0;
175         cont = 0;
176
177         if(abs(first_x - media_ax)<ang_aviso & abs(first_y
    - media_ay)<ang_aviso){// & abs(first_z - media_az)<15){
178           Serial.println("Buena postura");
179           noTone(zum); // NO se produce alerta
180           //digitalWrite(motor, LOW);// Paramos el motor -
    Cuando se tenga motor.
181         }else{
182           Serial.println("Mala postura, ponte recto");
183           //digitalWrite(motor, HIGH); //vibracion -Cuando
    se tenga motor.
184           //delay(500);  // -Cuando se tenga motor.
185           melodia(); // Se produce alerta
186
187         }
188
189       }
190       if(origen){
191         first_x = accel_ang_x;
192         first_y = accel_ang_y;
193         first_z = accel_ang_z;
```

```
194         Serial.print("Valor primera medida en eje X: ");
    Serial.println(first_x);
195         Serial.print("Valor primera medida en eje Y: ");
    Serial.println(first_y);
196         Serial.print("Valor primera medida en eje Z: ");
    Serial.println(first_z);
197
198         origen = false;
199       }
200
201     }
202
203     // Si se presiona el boton se apaga el dispositivo
204     if (digitalRead(boton1)){
205       digitalWrite(led, LOW); // Apagado
206       delay(1000); // Durante 1 segundo (1000ms)
207       estado1 = LOW; // Cambia el estado del boton a
    apagado.
208     }
209
210   }
211
212 }
213
214 void lecturas(){
215   // Si queremos pasar las lecturas del acelerometro a m/s
    ^2 hay que multiplicar las lecturas por (9.81/16384.0).
216   // En la componente Z se deben encontrar mediciones
    aproximadas a los 9.8 m/s^2
217   // Si queremos pasar las lecturas del giroscopio a deg/s
    (grados/s) hay que multiplicar las lecturas por
    (250.0/32768.0)
218   Serial.print("a[x y z]   Incl X   Incl Y  g[x y z]:\t");
219   Serial.print(ax*(9.81/16384.0)); Serial.print("\t"); //
    En m/s^2
220   Serial.print(ay*(9.81/16384.0)); Serial.print("\t"); //
    En m/s^2
221   Serial.print(az*(9.81/16384.0)/2); Serial.print("\t"); //
     En m/s^2
222
```

```
223   accel_ang_x=atan(ax/sqrt(pow(ay,2) + pow(az,2)))
       *(360.0/3.14); // En angulos de inclinacion
224   Serial.print(accel_ang_x); Serial.print("\t");
225   accel_ang_y=atan(ay/sqrt(pow(ax,2) + pow(az,2)))
       *(360.0/3.14); // En angulos de inclinacion
226   Serial.print(accel_ang_y); Serial.print("\t");
227   accel_ang_z=atan(az/sqrt(pow(ax,2) + pow(ay,2)))
       *(360.0/3.14); // En angulos de inclinacion
228   Serial.print(accel_ang_z); Serial.print("\t");
229
230   // Esto no es necesario
231   Serial.print(gx*(250.0/32768.0)); Serial.print("\t"); //
       En grados/s
232   Serial.print(gy*(250.0/32768.0)); Serial.print("\t"); //
       En grados/s
233   Serial.println(gz*(250.0/32768.0)); // En grados/s
234
235   delay(500); // Mide cada 0,5 segundos. Cuando se calcule
       la media para comprobar que la buena postura sera cada
       t_aviso*2
236
237 }
238
239 void calibracion(){
240   // Primero se filtran las lecturas
241   f_ax = f_ax-(f_ax>>5)+ax;
242   p_ax = f_ax>>5;
243
244   f_ay = f_ay-(f_ay>>5)+ay;
245   p_ay = f_ay>>5;
246
247   f_az = f_az-(f_az>>5)+az;
248   p_az = f_az>>5;
249
250   f_gx = f_gx-(f_gx>>3)+gx;
251   p_gx = f_gx>>3;
252
253   f_gy = f_gy-(f_gy>>3)+gy;
254   p_gy = f_gy>>3;
255
```

```
256   f_gz = f_gz-(f_gz>>3)+gz;
257   p_gz = f_gz>>3;
258
259   // Cada 100 lecturas se corrige el offset
260   if (counter==100){
261     // Mostramos las lecturas promedio para ver como va con
      la calibracion
262     Serial.print("promedio:"); Serial.print("\t");
263     Serial.print(p_ax); Serial.print("\t");
264     Serial.print(p_ay); Serial.print("\t");
265     Serial.print(p_az); Serial.print("\t");
266     Serial.print(p_gx); Serial.print("\t");
267     Serial.print(p_gy); Serial.print("\t");
268     Serial.println(p_gz);
269     // Basicamente se modifica constantemente el offset
     para que sea 0, como medida real.
270     // Se ajusta el Offset del acelerometro
271     if (p_ax>0) ax_o--;
272     else {ax_o++;}
273
274     if (p_ay>0) ay_o--;
275     else {ay_o++;}
276
277     if (p_az-16384>0) az_o--;
278     else {az_o++;}
279
280     sensor.setXAccelOffset(ax_o);
281     sensor.setYAccelOffset(ay_o);
282     sensor.setZAccelOffset(az_o);
283
284     // Se ajusta el Offset del giroscopio
285     if (p_gx>0) gx_o--;
286     else {gx_o++;}
287
288     if (p_gy>0) gy_o--;
289     else {gy_o++;}
290
291     if (p_gz>0) gz_o--;
292     else {gz_o++;}
293
```

```
294       sensor.setXGyroOffset(gx_o);
295       sensor.setYGyroOffset(gy_o);
296       sensor.setZGyroOffset(gz_o);
297
298       counter=0;
299
300       if (p_ax>-10 & p_ax<10 & p_ay>-10 & p_ay<10 & p_az
      >32757 & p_az<32777 & p_gx>-10 & p_gx<10 & p_gy>-10 &
      p_gy<10 & p_gz>-10 & p_gz<10){
301       //if (p_ax>-10 & p_ax<10 & p_ay>-10 & p_ay<10 & p_az
      >16374 & p_az<16394 & p_gx>-10 & p_gx<10 & p_gy>-10 &
      p_gy<10 & p_gz>-10 & p_gz<10){
302         Serial.println("DISPOSITIVO CALIBRADO!!!");
303         // El dispositivo Pita 3 veces ara indicar que se ha
      calibrado correctamente.
304         tone(zum, Sol, 500);
305         delay(200);
306         noTone(zum);
307         delay(200);
308         tone(zum, Sol, 500);
309         delay(200);
310         noTone(zum);
311         delay(200);
312         tone(zum, Sol, 500);
313         delay(200);
314         noTone(zum);
315         calibrar = false;
316       }else{
317         Serial.println("Calibrando...");
318       }
319     }
320     counter++;
321
322 }
323
324
325
326
327 void melodia(){
328   // Escala de musica con el zumbador
```

```
329     tone(zum, Fa, 500);
330     delay(700);
331     tone(zum, Sol, 500);
332     delay(700);
333     // Si se quiere realizar una melodia mas larga.
334     //tone(zum, Sol, 500);
335     //delay(700);
336     //tone(zum, La, 1000);
337     //delay(1700);
338     //tone(zum, Sol, 500);
339     //delay(700);
340     //tone(zum, Fa, 500);
341     //delay(700);
342     //tone(zum, Sol, 500);
343     //delay(700);
344     //tone(zum, Do, 1000);
345     //delay(1700);
346     //tone(zum, Fa, 500);
347     //delay(700);
348     //tone(zum, La, 500);
349     //delay(700);
350     //tone(zum, Fa, 500);
351     //delay(700);
352     //tone(zum, Re, 1000);
353     //delay(1700);
354
355 }
```

Esta versión proporciona un resultado satisfactorio, porque realiza la función del control postural correctamente, aunque hay que calibrarlo cada vez que se enciende el dispositivo y la calibración puede llevar varios minutos. Además, es posible modificar el ángulo de aviso que se quiere tener de umbral de correcta o incorrecta postura. También se puede modificar el tiempo de espera que se quiere tener para dar el aviso de una postura incorrecta, por defecto el tiempo es de 5 segundos.

Por otro lado, en esta versión se ha improbisado una base de calibración empleando material reciclado de cartón para mantener el sensor de forma plana. Además se improvisado también la forma de acoplarlo a la espalda del usuario para que se realicen mediciones correctas. Si no se coloca correctamente el sensor

habría que modificar los ejes en el código, ya que no detectara de forma correcta los ángulos y esto puede ser algo complejo.

B.3. Manuales y/o Demostraciones prácticas

si a la vez que preparas la demostración grabas los vídeos y los incluyes puede ser bastante valorado .. además de al tribunal puedes decir que le servirá al usuario para entender como debe realizar las acciones.

Las demostraciones prácticas se realizarán en base a una serie de pasos que se deberán realizar. Si estos pasos transcurren sin ningún contratiempo se obtendrá una ejecución exitosa. En el futuro se podrán observar y analizar los resultados.

Se incluirá la dirección a los videos de ejemplo.

Demostración de la primera versión del prototipo

En primer lugar el usuario se deberá colocar el prototipo, si es necesario puede solicitar ayuda a otra persona.

Una vez se encuentre colocado el dispositivo se encenderá presionando el botón de encendido, tras encender el prototipo se encenderá un led azul. El dispositivo empezará a controlar la postura del usuario.

Incluir imagen del prototipo colocado y el led

Para poder comprobar su correcto funcionamiento el usuario se inclinará hacia delante y el dispositivo al detectar un postura incorrecta dará feedback a través de sonido, y en el caso de tener añadido en el dispositivo un motor de vibración, también vibrará. Tras unos segundos el usuario volverá a una posición correcta, al corregir su postura el dispositivo dejará de sonar y de vibrar.

Incluir imagen de postura correcta y de postura incorrecta

Se volverá a comprobar que funciona correctamente haciendo que el usuario se vuelva a inclinar hacia delante, el dispositivo volverá a indicar que se ha establecido una mala postura y el usuario gracias al aviso sabrá que deberá corregir su postura, una vez el usuario haya corregido su postura el dispositivo dejará de pitar.

Se realizarán pruebas en las que el usuario se agache o salte o ande para ver cuando da problemas este dispositivo, debido a su sensibilidad a vibraciones. Los problemas detectados son aquellos que se intentarán solucionar en la segunda version donde se empleará un sensor más complejo.

Por último, el dispositivo se apagará empleando el botón de encendido/ apagado.

Demostración de la primera versión del prototipo

Para la demostración de esta segunda versión en la que se emplea un sensor más complejo y preciso, el sensor MPU6050[4, 5], el primer paso es que el usuario se coloque el prototipo.

Imagen del dispositivo colocado

Una vez el dispositivo se encuentre correctamente colocado el sistema pedirá el ángulo de aviso y el tiempo de aviso, si el usuario no indica estos datos se toma como valor por defecto 15º y 5 segundos. Se debe encender el dispositivo presionando el botón de encendido. Al encender el prototipo se encenderá un led de color azul y el dispositivo comenzará a controlar la postura. Si es la primera vez que se conecta al portatil el dispositivo deberá calibrar el dispositivo. En ese momento el usuario deberá presionar el botón de calibración para calibrar el dispositivo. Durante la calibración el dispositivo no debe moverse y puede tardar varios minutos. Una vez calibrado, el dispositivo pitará 3 veces o vibrará tres veces.

Incluir imagen de la pantalla cuando el sistema solicita el ángulo y el tiempo

Incluir imagen del dispositivo calibrando.

Tras la calibración el prototipo comenzará el control postural de forma correcta.

Para comprobar su funcionamiento la persona se inclinará ligeramente hacia delante, si supera el umbral, en este caso 15º, el dispositivo pitará y vibrará a modo de feedback. La persona gracias al biofeedback corregirá su postura, una vez adoptada una postura correcta el dispositivo dejará de emitir sonido y vibración. Se volverá a realizar una medición de prueba pero está vez el usuario se encontrará andando.

Incluir imagen del dispositivo colocado y de la persona en postura correcta e incorrecta

Incluir pantallazo del sistema cuando indica una postura correcta y otra incorrecta

El usuario se inclinará de nuevo ligeramente y el dispositivo indicará una mala postura, la persona corregirá su postura tras conocer su estado gracias al dispositivo, y una vez corregida la postura el dispositivo dejará de emitir la alerta.

Se puede probar la mayor precisión de esta versión realizando los movimientos que perturbaban en la versión anterior, como agacharse o saltar donde el sensor deberá no deberá emitir alerta.

Queda comprobado que con esta versión se obtiene un dispositivo más preciso y personalizado y, además, se puede emplear tanto para uso en personas sentadas o en movimiento (andando).

Por último, se deberá apagar el dispositivo con el botón de encendido/apagado una vez ya no se requiera el control postural.

Apéndice C

Manual del desarrollador / programador / investigador.

C.1. Estructura de directorios

Eliminar aquellos anexos que no se hayan realizado o entregado en la memoria final

En el repositorio de GitHub encontraremos los siguientes ficheros y directorios:

- **Carpeta img**: carpeta dónde se incluyen todas las imágenes que se han empleado en el desarrollo del proyecto.
- **Carpeta tex**:
  - **1_objetivos.tex**: documento LaTex que contiene la información acerca de los objetivos.
  - **2_introduccion.tex**: documento LaTex que contiene la información acerca de la introducción del trabajao, se incluyen conceptos teóricos básicos y el estado del arte.
  - **3_metodologia.tex**: documento LaTex que contiene la información acerca de la metodología empleada, dónde se incluyen descripción de los datos con los que se trabajan y técnicas y herramientas.
  - **4_conclusiones.tex**: documento LaTex que contiene las conclusiones del proyecto, se puede encontrar un resumen de resultados, discusión y aspectos relevantes.

    - **5_lineas_futuras.tex**: documento LaTex que contiene la información acerca de las líneas de trabajo futuras.
    - **A_planificacion.tex**: documento LaTex que contiene información del anexo A, donde se incluye la planificación temporal, la planificación económica y la viabilidad legal en España del trabajo.
    - **B_manual_usuario.tex**: documento LaTex que contiene la información del anexo B que incluye los requisitos funcionales y no funcionales, la instalación y puesta en marcha y manuales o demostraciones prácticas.
    - **C_manual_programador.tex**: documento LaTex que contiene la información del anexo C que contiene la estructura de los directorios entregados, la información acerca de la ejecución del proyecto y las instrucciones de mejora del proyecto.
    - **D_datos.tex**: documento LaTex que contiene la información acerca de los datos utilizados en el proyecto.
    - **E_diseno.tex**: documento LaTex que contiene la información acerca del diseño del prototipo realizado.
    - **F_requisitos.tex**: documento LaTex que contiene la información acerca los casos de uso definidos.
    - **G_experimental.tex**: documento LaTex que contiene la información acerca del estudio experimental realizado.
    - **readme.txt**:
- **Carpeta videos**: carpeta dónde se encuentran los vídeos de demostración del proyecto.
- **Carpteta Arduino**: carpeta dónde se encuentran los programas de arduino empleados en las distintas versiones del prototipo del proyecto.
    - **ProgramaV1.ino**: Programa que se ha de cargar en la placa de Arduino de la primera versión del prototipo del dispositivo de control postural, dónde se emplea el sensor SW520D.
    - **ProgramaV2.ino**: Programa que se ha de cargar en la placa de Arduino de la segunda versión del prototipo del dispositivo de control postural, dónde se emplea el módulo MPU-6050.
- **README.md**: archivo de presentación del repositorio de GitHub.
- **anexos.pdf**: documento PDF que contiene los anexos completos.
- **anexos.tex**: archivo LaTex que contiene la estructura del documento pdf de los anexos.

- **bibliografia.bib**: archivo que recoge la bibliografía de la memoria.
- **bibliografiaAnexos.bib**: archivo que recoge la bibliografía de los anexos.
- **memoria.pdf**: documento PDF que contiene la memoria completa.
- **memoria.tex**: archivo LaTex que contiene la estructura del documento de la memoria.

C.2. Instrucciones para la modificación o mejora del proyecto.

Al ser este proyecto un prototipo hay varios aspectos de mejora para el futuro desde añadir componentes al prototipo a crear una carcasa o crear una base de datos donde se guarden los datos adquiridos.

Durante el desarrollo de la primera fase se observó que el sensor empleado no era de gran precisión y por ello se decidió crear una segunda versión con un sensor de mayor precisión, el sensor MPU-6050[4, 5], que supusiera una ventaja al proyecto.

La segunda versión tiene la necesidad de realizar una calibración cada vez que se conecta el prototipo al portátil que tiene instalado el software de Arduino IDE[20], como primera propuesta de mejora se podría modificar el código del programa para que no fuese necesario calibrar el dispositivo con tanta frecuencia y para que no fuese necesario presionar un el botón de calibración siempre.

El sensor resultó exitoso, sin embargo, por las características del sensor MPU6050 este prototipo no puede ser utilizado durante largos periodos de tiempo ya que se crea deriva y se distorsionan las mediciones. Por ello, como propuesta de mejora se podrían estudiar el uso de otros sensores similares que resolviesen este problema uno de los sensores que se podrían emplear sería el MPU9250[**MPU9250_1**, **MPU9250_2**].

Además, para el prototipo se podría crear una placa electrónica donde se encuentren todos los componentes del prototipo sin la necesidad de emplear una placa de pruebas. Asimismo, se podría crear una carcasa, por ejemplo, empleando impresión 3D que protegiese el hardware y que se adaptase de manera ergonómica al cuerpo de la persona.

Al prototipo se podrían incluir otros elementos como pueden ser una batería y un módulo Bluetooth para mejorar la interacción con el usuario, eliminando cables que pueden ser incómodos para el usuario además que pueden limitar la libertad de los movimientos. Disponer de un módulo Bluetooth también permitiría transmitir los datos que recoge el sensor sin necesidad de conexión USB.

Este último punto nos lleva a otro punto a mejorar, el almacenamiento y tratamiento de los datos. Se puede implementar una forma de almacenaje de los datos para que se pueda trabajar con ellos obteniendo distintas estadísticas que serían útiles en el futuro. Y que se realice un trabajo estadístico de esos mismos datos para conecer más información a parte de la corrección en tiempo real de la postura.

Apéndice D

Descripción de adquisición y tratamiento de datos

Como se ha mencionado en la memoria en este proyecto se han empleado **datos de elaboración propia**, no se han obtenido datos externos de bases de datos existentes.

Por un lado se trabaja con los datos personales proporcionados por el usuario:

- Nombre
- Apellidos
- Edad
- Correo electrónico
- Nombre de usuario
- Constraseña

Por el otro lado, en la segunda versión del prototipo también se trabaja con los datos recopilados por el dispositivo. Se realiza una medición cada 0,5 segundos[1] y en cada medición se obtienen los siguientes datos:

- Aceleración eje X en m/s^2.
- Aceleración eje Y en m/s^2.

[1]En el futuro se puede modificar y se puede trabajar con medidas medias cada cierto tiempo.

- Aceleración eje Z[2] en m/s^2.
- Inclinación eje X en deg.
- Inclinación eje Y en deg.
- Rotación eje X en deg/s.
- Rotación eje Y en deg/s.
- Rotación eje Z en deg/s.

Las inclinaciones son las que se emplean en el segundo prototipo para diferenciar entre una postura correcta o incorrecta.

La acumulación de estos datos en bases de datos abrirá el camino a la realización de estadísticas que aporten más información al usuario sobre el estado de su postura y su evolución al emplear el dispositivo.

Todos estos datos deberán ser protegidos conforme a la legislación español definida en el *Anexo A*.

De igual modo, en el futuro, se pueden acumular datos de gran cantidad de personas y de esta forma también se podrían realizar estudios estadísticos con diferentes cohortes para obtener más información sobre la postura y el control postural.

[2]esta aceleración deberá aproximarse a la gravedad de la tierra, a 9.81 m/s^2

Apéndice E

Manual de especificación de diseño

Si es necesario. Se incluirán los planos de las distintas versiones de arduino.

No se ha realizado ningún diagrama de despliegue para ninguna de las versiones propuestas puesto que no se ha llegado a crear la apliación móvil de interacción de usuario-prototipo, tampoco se han definido clases en los programas de Arduino[20] creados para cada una de las versiones.

Para cada una de las versiones se han creado únicamente funciones y atributos. Se pueden ver las distintas funciones y atributos en el siguiente diagrama:

Añadir diagrama con los atributos y funciones de cada uno de los programas de los 2 prototipos

Planos (Si procede) Diseño arquitectonico (Si procede) Diagrama de clases, diagrama de despliegue

E.1. Planos

Si procede

E.2. Diseño arquitectónico

Si procede.

Diagramas de clases, diagramas de despliegue ...

Apéndice F

Especificación de Requisitos

F.1. Diagrama de casos de uso

En la *Figura F.1* se puede ver el diagrama de casos de uso definido para el desarrollo de la aplicación de usuario.

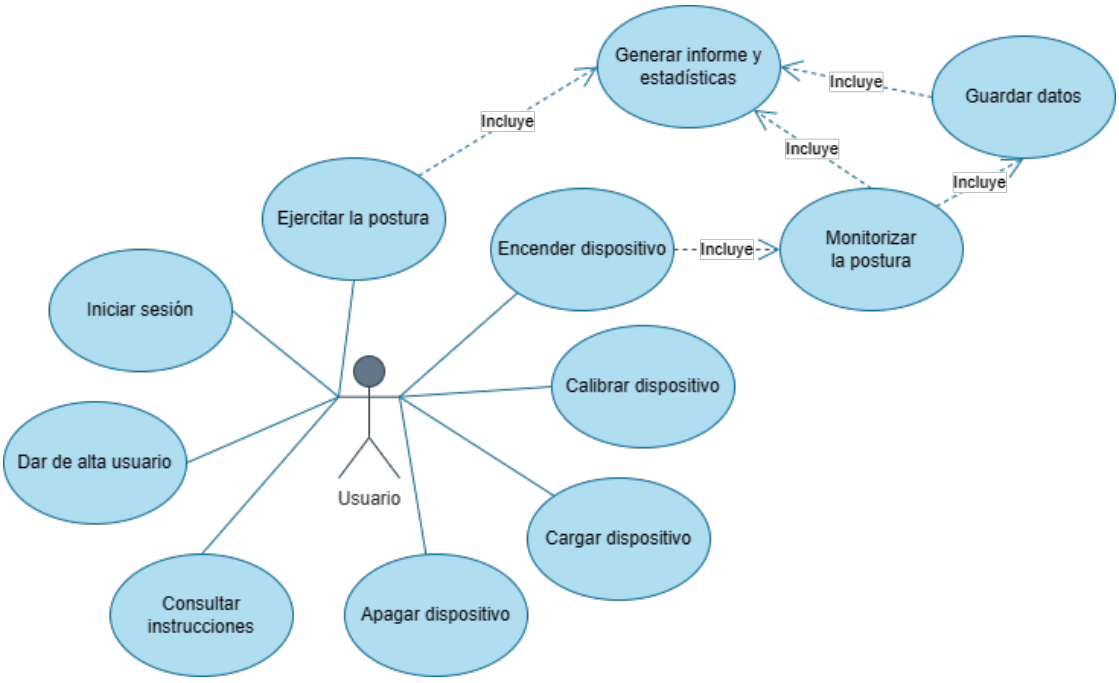


Figura F.1: Diagrama de casos de uso

F.2. Explicación casos de uso.

CS-01	<Encender dispositivo>
Versión	1.0
Autor	Naiara Gadea Rodíguez Gómez
Descripción	Puesta en marcha del dispositivo hardware y su conexión con la aplicación. El dispositivo no deberá estar siempre conectado con la aplicación, pero si se conecta a la aplicación se puede obtener más información.
Secuencia Normal	1. Se coloca el dispositivo en contacto sobre la piel, como se indica en las instrucciones. 2. Se enciende el dispositivo con el botón ON/OFF. 3. Cuando se encienda un led verde indicará que el dispositivo está encendido. 4. Se conecta el dispositivo al dispositivo que tiene instalada la aplicación vía Bluetooth o vía WIFI. 5. El usuario accede a la aplicación software instalada en el dispositivo móvil o en el ordenador, que deberán estar conectados a la misma red WIFI o Bluetooth para permitir la comunicación. 6. Una vez se realiza la conexión, el led verde del dispositivo cambia a color a azul. Además, en la aplicación aparece el dispositivo como conectado.
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Requisitos Funcionales Relacionados	RF-02
Casos de uso relacionados	CS-02, CS-03, CS-04, CS-07, CS-08, CS-10

Tabla F.1: CU-01. Encender dispositivo.

CS-02	**<Apagar dispositivo>**
Versión	1.0
Autor	Naiara Gadea Rodíguez Gómez
Descripción	Apagado del dispositivo.
Secuencia Normal	1. Se apaga el dispositivo con el botón ON/OFF. 2. Una vez el dispositivo se encuentre apagado el led azul o verde se apagará. 3. El usuario se puede quitar el dispositivo y puede cargarlo.
Frecuencia	Alta
Importancia	Alta
Urgencia	Media
Requisitos Funcionales Relacionados	RF-02
Casos de uso relacionados	CS-01, CS-03
Comentarios	El dispositivo también se apaga cuando se acaba la batería, en cuyo caso el caso de uso comienza en el paso 2.

Tabla F.2: CU-02. Apagar dispositivo.

CS-03	**<Cargar dispositivo>**
Versión	1.0
Autor	Naiara Gadea Rodíguez Gómez
Descripción	Carga de la batería del dispositivo empleando la estación de carga.
Secuencia Normal	1. Actúa el CS-02. 2. Una vez que el dispositivo se encuentre apagado el usuario coloca el dispositivo sobre la estación de carga enchufada a la corriente. 3. Se enciende un led rojo intermitente. 4. Cuando la batería del dispositivo se encuentre completamente cargada, el led rojo intermitente deja de ser intermitente. 5. El dispositivo está completamente cargado y puede desconectar la estación de carga de la corriente o desconectar el dispositivo. 6. El dispositivo estará listo para su uso.
Frecuencia	Alta
Importancia	Alta
Urgencia	Media
Requisitos Funcionales Relacionados	RF-08
Casos de uso relacionados	CS-02

Tabla F.3: CU-03. Cargar dispositivo.

CS-04	**<Calibrar dispositivo>**
Versión	1.0
Autor	Naiara Gadea Rodíguez Gómez
Descripción	Se recalculan los valores en reposo para que el dispositivo devuelva medidas lo más precisas posible. Se debe calibrar el dispositivo cada cierto tiempo para que los valores medidos sean lo más precisos posible.
Secuencia Normal	1. El dispositivo debe estar encendido y sobre una superficie plana. 2. Se presionará el botón de calibrado. Sobre el dispositivo o de la propia aplicación. 3. El sistema recalibra los valores medidos por el dispositivo. 4. Una vez el dispositivo se haya calibrado correctamente se encenderá 3 veces el led verde o azul, en función si la calibración se ha realizado a través de la aplicación o directamente sobre el dispositivo. 5. El dispositivo se encuentra correctamente calibrado y listo para su uso.
Frecuencia	Baja
Importancia	Alta
Urgencia	Baja
Requisitos Funcionales Relacionados	RF-01, RF-04
Casos de uso relacionados	CS-01, CS-07

Tabla F.4: CU-04. Calibrar dispositivo.

CS-05	**<Dar de alta usuario>**
Versión	1.0
Autor	Naiara Gadea Rodíguez Gómez
Descripción	El usuario crea una cuenta para cada recopilar los datos recopilados por el dispositivo. Se creará una cuenta por persona que vaya a utilizar el dispositivo.
Secuencia Normal	1. El caso de uso comienza cuando el facultativo clica en registrar. 2. El sistema solicita el nombre de usuario. 3. El usuario introduce el nombre de usuario. 4. El sistema solicita los nombre y apellidos del usuario. 5. El usuario introduce su nombre y apellidos. 6. El sistema solicita una contraseña. 7. El usuario introduce una contraseña. 8. El sistema guarda y almacena los datos introducidos. Se ha creado la cuenta del usuario.
Frecuencia	Baja
Importancia	Media
Urgencia	Media
Requisitos Funcionales Relacionados	RF-01, RF-03
Casos de uso relacionados	CS-06

Tabla F.5: CU-05. Dar de alta usuario.

CS-06	**<Iniciar sesión>**
Versión	1.0
Autor	Naiara Gadea Rodíguez Gómez
Descripción	El usuario inicia sesión en la aplicación para poder acceder a sus datos, informes y ejercicios. El usuario tiene la opción de mantener la sesión iniciada.
Secuencia Normal	1. El sistema solicita nombre de usuario y contraseña. 2. El usuario introduce su nombre de usuario y contraseña. 3. El sistema pregunta si desea mantener la sesión abierta. 4. El usuario introduce si desea mantener la sesión abierta. 5. El sistema compara con su base de datos y si encuentra coincidencia accede a los datos del usuario. En caso de que no encuentre al usuario el sistema imprime por pantalla 'Contraseña o usuario incorrectos'. 6. El sistema muestra las estadísticas y datos del paciente.
Frecuencia	Media
Importancia	Alta
Urgencia	Media
Requisitos Funcionales Relacionados	RF-01, RF-03, RF-06
Casos de uso relacionados	CS-05, CS-07, CS-08, CS-09, CS-10
Comentarios	Si el usuario ha indicado que desea mantener la sesión abierta, la sesión se mantiene abierta, aunque se cierre la aplicación. El usuario deberá cerrar manualmente su sesión.

Tabla F.6: CU-06. Iniciar sesión.

CS-07	**<Realizar monitoreo de la postura>**
Versión	1.0
Autor	Naiara Gadea Rodíguez Gómez
Descripción	Una vez el dispositivo se encuentre encendido se realiza la monitorización de la postura. Si se detecta una mala postura el dispositivo emitirá una vibración. En el caso de que el dispositivo se encuentre conectado con la aplicación los datos.
Secuencia Normal	1. El caso de uso comienza cuando se enciende el usuario enciende el dispositivo. 2. Pasos del CU El sistema obtiene los datos proporcionados por el sensor del dispositivo. 3. El sistema identifica entre una buena o mala postura, gracias al algoritmo. 4. Si se detecta mala postura el dispositivo emita una señal vibratoria para que el usuario modifique su postura. 5. Interviene el CS-08. 6. El caso de uso finaliza cuando se apaga el dispositivo o cuando se gasta la batería del dispositivo.
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Requisitos Funcionales Relacionados	RF-02, RF-04, RF-05
Casos de uso relacionados	CS- 01, CS-04, CS-06, CS-08, CS-09, CS-10

Tabla F.7: CU-07. Realizar monitoreo de la postura.

CS-08	**<Guardar datos>**
Versión	1.0
Autor	Naiara Gadea Rodíguez Gómez
Descripción	Se archivan los datos recopilados a través del dispositivo. Los datos se van guardando en tiempo real si el dispositivo se encuentra conectado a la aplicación.dices que 'puede ser en tiempo real' deberá de explicar en que casos no es en tiempo real, o que pasa con el almacenamiento.
Secuencia Normal	1. El sistema registra los datos enviados vía Bluetooth o WIFI por el dispositivo. 2. Interviene el CS-07. 3. El sistema guarda los datos y el informe generado en la base de datos integrada. Que se podrán consultar posteriormente o en tiempo real.
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Requisitos Funcionales Relacionados	RF-01, RF-02, RF-04, RF-06
Casos de uso relacionados	CS-06, CS-07, CS-09

Tabla F.8: CU-08. Guardar datos.

CS-09	**<Generar informe>**
Versión	1.0
Autor	Naiara Gadea Rodíguez Gómez
Descripción	A partir de los datos recopilados del sensor, se realiza un informe que puede ser en tiempo real que incluye toda la información relevante y estadísticas calculadas en función de los datos. CS-09 dices que 'puede ser en tiempo real' deberá de explicar en que casos no es en tiempo real, o que pasa con el almacenamiento.
Secuencia Normal	1. El caso de uso comienza tras el CS-08. 2. El sistema incluye los datos obtenidos. 3. El sistema crea varias estadísticas utilizando gráficas que resumen visualmente los datos recogidos y la evolución del paciente. 4. Se puede obtener un informe en formato pdf.Incluir donde secrea, como se crea..
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Requisitos Funcionales Relacionados	RF-01, RF-02, RF-04, RF-06
Casos de uso relacionados	CS-06, CS-07, CS-08, CS-10

Tabla F.9: CU-09. Generar informe.

CS-10	**<Ejercitar la musculatura de la postura>**
Versión	1.0
Autor	Naiara Gadea Rodíguez Gómez
Descripción	Se realizan distintos juegos o ejercicios incluidos en la aplicación software, que, mediante la interacción con el dispositivo hardware, permitirán al usuario aumentar y mejorar la musculatura que se necesita para una buena postura. en realizan distintos juegos o ejercicios incluidos en la aplicación softwarezo subdibidiría en 1. y 2. con una breve descripción por lo menos de los mas relevantes. auqneu puedes indicar que hay más si no describes todos.
Secuencia Normal	1. El usuario selecciona dentro de la pestaña de juegos y ejercicios, el juego o el ejercicio que desee realizar. 2. El sistema muestra el juego o ejercicio seleccionado. 3. El usuario realiza el juego o ejercicio seleccionado. 4. Una vez finalizado el juego o el ejercicio el sistema vuelve a la pantalla donde se incluyen los ejercicios y juegos disponibles para mejorar la musculatura o la postura.
Frecuencia	Media
Importancia	Baja
Urgencia	Baja
Requisitos Funcionales Relacionados	RF-01, RF-02, RF-04, RF-05
Casos de uso relacionados	CS-01, CS-04, CS-06, CS-07, CS-09, CS-11

Tabla F.10: CU-10. Ejercitar la musculatura de la postura.

CS-11	**<Consultar instrucciones de uso>**
Versión	1.0
Autor	Naiara Gadea Rodíguez Gómez
Descripción	El usuario puede revisar las instrucciones de uso del dispositivo, y de esa forma puede obtener información para la puesta en marcha para poder monitorizar su postura.
Secuencia Normal	1. El usuario accede a su perfil y clica sobre el icono '?' donde se puede consultar las instrucciones. 2. El sistema abre una pestaña con las instrucciones de uso que puede seguir el usuario. Las instrucciones serán sencillas y deberán redirigir a un vídeo explicativo de la puesta en marcha y uso del dispositivo. 3. El usuario cierra la ventana de instrucciones cuando deje de necesitar su consulta.
Frecuencia	Baja
Importancia	Baja
Urgencia	Baja
Requisitos Funcionales Relacionados	RF-02, RF-07
Casos de uso relacionados	CS-01, CS-02, CS-03, CS-04, CS-05, CS-06, CS-07, CS-09, CS-10

Tabla F.11: CU-11. Consultar instrucciones de uso.

F.3. Prototipos de interfaz o interacción con el proyecto.

Se ha realizado un prototipo de interfaz de aplicación móvil, en base a las aplicaciones de los dispositivos existentes.

La interfaz creada consta de 6 pantallas principales:

1. Pantalla de inicio de sesión.
2. Pantalla de inicio con información en tiempo real.
3. Pantalla con las estadísticas en forma de gráfica de distintos periodos de tiempo.
4. Pantalla con juegos y ejercicios de mejora de la postura.
5. Pantalla de ajustes del dispositivo conectado.
6. Pantalla del perfil del usuario.

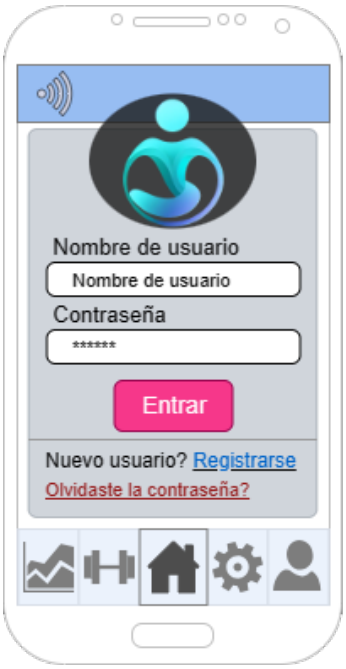


Figura F.2: Pantalla de inicio de sesión.

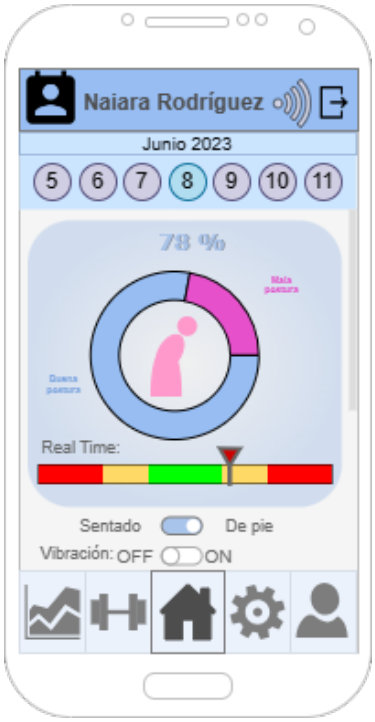


Figura F.3: Pantalla de inicio con información en tiempo real.

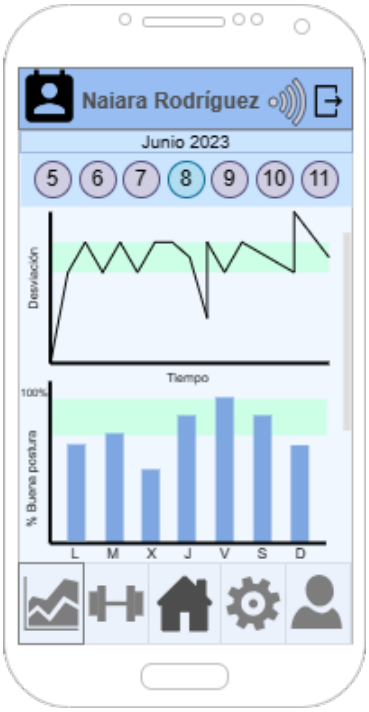


Figura F.4: Pantalla con las estadísticas en forma de gráfica de distintos periodos de tiempo.

Figura F.5: Pantalla con juegos y ejercicios de mejora de la postura.

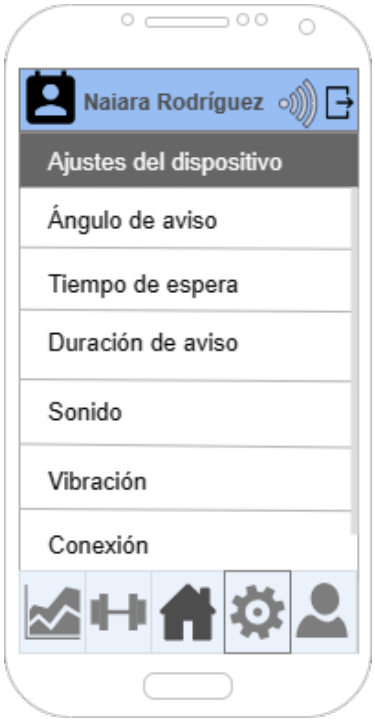


Figura F.6: Pantalla de ajustes del dispositivo conectado.

Figura F.7: Pantalla del perfil del usuario.

Apéndice G

Estudio experimental

En este trabajo no se ha podido realizar un estudio experimental, pero es posible que en el futuro se puedan realizar este tipo de estudios en base al seguimiento de los resultados de diferentes pacientes y la explotación estadística que se puede realizar de los datos recopilados empleando el dispositivo.

Bibliografía

[1] Jobted.es. *¿Cuánto Cobra un Ingeniero? (Sueldo 2023)*. 2023. URL: https://www.jobted.es/salario/ingeniero#:~:text=Un%5C%20Ingeniero%5C%20reci%5C%C3%5C%A9n%5C%20egresado%5C%2C%5C%20con%5C%20menos%5C%20de%5C%203,de%5C%20alrededor%5C%20de%5C%2020.450%5C%20%5C%E2%5C%82%5C%AC%5C%20brutos%5C%20por%5C%20a%5C%C3%5C%B1o..

[2] D. Frederick. *Amortización*. 2023. URL: https://enciclopediaeconomica.com/amortizacion/.

[3] Luis Llamas. *Medir inclinación con Arduino y sensor tilt SW-520d*. 2015. URL: https://www.luisllamas.es/medir-inclinacion-con-arduino-y-sensor-tilt-sw-520d. (accessed: 17.05.2023).

[4] Naylamp Mechatronics. *Tutorial MPU6050, Acelerómetro y Giroscopio*. URL: https://naylampmechatronics.com/blog/45_tutorial-mpu6050-acelerometro-y-giroscopio.html.

[5] Luis Llamas. *Determinar la orientación con Arduino y el IMU MPU-6050*. URL: https://www.luisllamas.es/arduino-orientacion-imu-mpu-6050/.

[6] amazon.es. *Amazon España*. URL: https://www.amazon.es/.

[7] BOE. "Ley 24/2015, de 24 de julio, de Patentes (BOE-A-2015-8328)." En: *Boletín Oficial del Estado* 177 (2015). URL: https://www.boe.es/buscar/act.php?id=BOE-A-2015-8328.

[8] BOE. "Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia. (BOE-A-1996-8930)." En: *Boletín Oficial del Estado* 97 (1996). URL: https://boe.es/buscar/act.php?id=BOE-A-1996-8930.

[9] AEMPS. *Legislación sobre productos sanitarios.* 2022. URL: https://www.aemps.gob.es/productos-sanitarios/legislacion-sobre-productos-sanitarios/.

[10] BOE. "Real Decreto 1591/2009, de 16 de octubre, por el que se regulan los productos sanitarios. (BOE-A-2009-17606)." En: *Boletín Oficial del Estado* 268 (2009). URL: https://www.boe.es/buscar/doc.php?id=BOE-A-2009-17606.

[11] BOE. "Real Decreto 437/2002, de 10 de mayo, por el que se establecen los criterios para la concesión de licencias de funcionamiento a los fabricantes de productos sanitarios a medida.(BOE-A-2002-10228)." En: *Boletín Oficial del Estado* 128 (2002). URL: https://www.boe.es/buscar/doc.php?id=BOE-A-2002-10228.

[12] EUR-Lex. "Reglamento (UE) 2017/745 del Parlamento Europeo y del Consejo, de 5 de abril de 2017, sobre los productos sanitarios, por el que se modifican la Directiva 2001/83/CE, el Reglamento (CE) n.° 178/2002 y el Reglamento (CE) n.° 1223/2009 y por el que se derogan las Directivas 90/385/CEE y 93/42/CEE del Consejo (Texto pertinente a efectos del EEE.)" En: *Diario Oficial de la Unión Europea* (2017). URL: https://www.boe.es/buscar/act.php?id=BOE-A-2015-8328.

[13] Agencia Estatal Boletín Oficial del Estado. *Código Laboral y de la Seguridad Social.* URL: https://www.boe.es/biblioteca_juridica/codigos/codigo.php?id=93&modo=2&nota=0&tab=2.

[14] BOE. "Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico.(BOE-A-2002-13758)." En: *Boletín Oficial del Estado* 166 (2002). URL: https://www.boe.es/buscar/act.php?id=BOE-A-2002-13758.

[15] Agencia Estatal Boletín Oficial del Estado. *Código de Comercio y legislación complementaria.* URL: https://www.boe.es/biblioteca_juridica/codigos/codigo.php?id=35&modo=2&nota=0&tab=2.

[16] BOE. "Ley 7/1996, de 15 de enero, de Ordenación del Comercio Minorista. (BOE-A-1996-1072)." En: *Boletín Oficial del Estado* 15 (1996). URL: https://www.boe.es/buscar/act.php?id=BOE-A-1996-1072.

[17] BOE. "Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales. (BOE-A-2018-16673)." En: *Boletín Oficial del Estado* 294 (2018). URL: https://www.boe.es/buscar/doc.php?id=BOE-A-2018-16673.

[18] EUR-Lex. "Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE (Reglamento general de protección de datos)". En: *Diario Oficial de la Unión Europea* (2016). URL: https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=celex%5C%3A32016R0679.

[19] Your Europe. *Marcado CE*. URL: https://europa.eu/youreurope/business/product-requirements/labels-markings/ce-marking/index_es.htm.

[20] Arduino. *Arduino IDE 2.1.0, Downloads*. URL: https://www.arduino.cc/en/software.

[21] Arduino. *What is Arduino?* URL: https://www.arduino.cc/en/Guide/Introduction. (accessed: 17.03.2023).

[22] Arduino. *UNO R3, Arduino documentation*. URL: https://docs.arduino.cc/hardware/uno-rev3. (accessed: 17.03.2023).

[23] Jeff Rowberg (jrowberg). *I2C Device Library, I2Cdev*. URL: https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/I2Cdev.

[24] Jeff Rowberg (jrowberg). *I2C Device Library, MPU6050*. URL: https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050.

[25] Arduino. *Wire*. URL: https://www.arduino.cc/reference/en/language/functions/communication/wire/.

[26] GitHub. *GitHub*. URL: https://github.com/github.