



Proyecto Fin de Carrera

Pulsera biométrica Open Source para la monitorización del usuario

Ingeniería industrial especialización Automatización industrial y Robótica/ Electrónica

Autor/es

Esther Borao Moros

Director/es

Luis Antonio Martin Nuez
Roberto Casas Nebra

Escuela de Ingeniería y Arquitectura
Curso 2016-2017

*Gracias a todas esas personas
que creen en mí,
que me ayudan a crecer y
que siempre están ahí.*

Resumen

Este proyecto ha sido realizado con el fin de obtener una nueva herramienta y dispositivo disponible para la comunidad de investigadores, emprendedores y artistas. Personas que cada día buscan más apoyo en proyectos y plataformas Open Source, abriendo también las puertas al usuario particular de las tecnologías vestibles de medición biométricas a un bajo coste.

El objeto del proyecto es el desarrollo de una pulsera biométrica Open Source para la monitorización del usuario. Un sistema electrónico basado en la plataforma Arduino, capaz de recoger información acerca de los parámetros de un usuario, tanto del estado de su organismo como de su entorno inmediato. El proyecto se apoyará en las bases definidas por los productos del movimiento Open Source: contenido libre, accesibilidad, bajo coste...

El primer paso consistirá en la búsqueda de las necesidades del usuario, investigación de sistemas ya existentes, y estudio de sensores en el mercado. Esto permitirá definir unos requisitos mínimos de la plataforma, con los que poder seguir trabajando. Una vez definidos los sensores se deben diseñar las diferentes etapas, el hardware formado por la electrónica de adaptación necesaria para los sensores elegidos. Con respecto al software, se implementará un firmware con múltiples funciones y una interfaz de control, con el fin de facilitar su programación y el uso al usuario final. El resultado final pretende ser un dispositivo real, que acerque la tecnología de la monitorización médica a otros lugares a un precio asequible, y que sirva de base a futuros desarrollos o investigaciones.

En cuanto a los conocimientos obtenidos en el transcurso de la realización de PFC, cabe destacar su carácter multidisciplinar, el haber adquirido los conocimientos necesarios para cada una de las fases de la producción de un producto: desde monitorización y medición biomédica, hasta la fabricación y análisis de costes.

Contenido

Resumen	3
Contenido	4
Índice de figuras	7
Índice de tablas	9
1. Introducción	10
1.1. Justificación	10
1.1.1. Objeto	10
1.1.2. Motivación del proyecto	11
1.2. Objetivos.....	12
1.2.1. Alcance	13
1.3. Antecedentes.....	14
1.4. Metodología y plan de trabajo.....	15
2. Campo de aplicación.....	16
2.1. Posibles aplicaciones del sistema	16
2.2. Estado del arte	17
2.2.1. Wearables.....	17
3. Especificación de los requisitos de aplicación.....	29
3.1. La tecnología de partida: ARDUINO	29
3.2. Especificaciones básicas iniciales	29
3.3. Especificaciones basadas en el estudio de mercado	30
3.3. Especificaciones basadas en el estudio de la normativa	30
3.4. Especificaciones basadas en el estudio de las comunicaciones inalámbrica	31
3.5. Diseño final	31
4. Diseño hardware	32
4.1. Metodología	32
4.2. Diagrama de bloques del sistema.....	33
4.3. Selección de componentes	33
4.3.1. Sensores	33
4.3.2. Pantalla.....	50
4.3.3. Bluetooth	52
4.3.4. Microcontrolador.....	61
4.3.5. Carga.....	63
4.4. Decisión final componentes	65
4.5. Esquemático	67
4.5.1. Pinout AtMega1284p.....	67
4.5.2. Circuitos utilizados.....	70
4.6. Diseño de la PCB.....	79
4.7. Fabricación.....	85
4.7.1. Versión 0	85
4.7.2. OPEN BIOBAND versión 1	86
4.7.3. OPEN BIOBAND versión 2	87

5. Diseño software.....	88
5.1. Introducción	88
5.2. Diseño de interacción con el usuario	89
5.3. Implementación	90
5.3.1. Código principal	90
5.3.2. Pantallas secundarias.....	94
5.3.3. Librerías.....	115
6. Diseño mecánico.....	120
6.1. Introducción	120
6.2. Tinkercad	120
6.3. Modelos.....	122
6.3.1. PLA.....	122
6.3.2. FILAFLEX.....	123
6.3.3. Diseño final.....	124
7. Montaje.....	126
8. Pruebas realizadas	129
8.1. Introducción	129
8.2. Pruebas generales	129
8.3. Calibración de sensores.....	130
8.4. Pruebas de consumo.....	131
8.5. Pruebas de compatibilidad	133
9. Conclusiones.....	134
9.1. Conclusiones personales.....	135
10. Bibliografía	136

Anexos

ANEXO 1. CRONOGRAMA	137
ANEXO 2. NORMATIVA FDA	139
ANEXO 3. LA TECNOLOGÍA DE PARTIDA: ARDUINO	145
ANEXO 4. BOOTLOADER PCB.....	150
ANEXO 5. PRIMERAS PRUEBAS: PULSERA VERSIÓN 0	152
ANEXO 6. FIRMWARE PRINCIPAL	159
ANEXO 7. FIRMWARE PANTALLAS SECUNDARIAS	173
ANEXO 8. ESQUEMÁTICO FINAL	210
ANEXO 9. LISTADO DE COMPONENTES: BOM.....	211
ANEXO 10. DISEÑO PCB FINAL	213
ANEXO 11. COSTES FABRICACIÓN	220
ANEXO 12. HOJAS DE CARACTERÍSTICAS	223

Índice de figuras

<i>Fig. 1 Evolución wearables-----</i>	14
<i>Fig. 2 Colores dependiendo de las emociones del público-----</i>	19
<i>Fig. 3 Representación datos de las pulseras. Festival Cannes -----</i>	19
<i>Fig. 4 Diagrama de bloques del sistema-----</i>	33
<i>Fig. 5 Pruebas acelerómetro -----</i>	35
<i>Fig. 6 Señal típica de electrocardiograma-----</i>	37
<i>Fig. 7 Funcionamiento fotopletismógrafo-----</i>	37
<i>Fig. 8 Pruebas pulso -----</i>	39
<i>Fig. 9 Fabricación PCB de prueba y bootloader ATtiny-----</i>	39
<i>Fig. 10 PCB sensor de pulso MAX30101 -----</i>	40
<i>Fig. 11 Smartband con GSR-----</i>	44
<i>Fig. 12 Pruebas ambiente encima del radiador-----</i>	47
<i>Fig. 13 Pruebas ambiente en exterior-----</i>	47
<i>Fig. 14 Pruebas ambiente en interior -----</i>	48
<i>Fig. 15 Pruebas Readbear Blend Micro -----</i>	60
<i>Fig. 16 Pruebas nRF8001 -----</i>	60
<i>Fig. 17 Pruebas Readbear Nano -----</i>	60
<i>Fig. 18 PCB con ATmega1284p -----</i>	62
<i>Fig. 19 PCB con CórTEX M0-----</i>	62
<i>Fig. 20 Cableado Arduino Zero para utilizarlo como programador -----</i>	62
<i>Fig. 21 Carga inalámbricaATmega1284p -----</i>	63
<i>Fig. 22 Carga por USB CórTEX M0 -----</i>	63
<i>Fig. 23 Circuito de carga por USB -----</i>	64
<i>Fig. 24 Cargador moto360-----</i>	64
<i>Fig. 25 Bobina y adhesivo magnético moto360 -----</i>	64
<i>Fig. 26 Diagrama de bloques del sistema detallado-----</i>	67
<i>Fig. 27 Esquemático Open Bioband-----</i>	69
<i>Fig. 28 Esquemático acelerómetro-----</i>	70
<i>Fig. 29 Conexión datasheet acelerómetro -----</i>	70
<i>Fig. 30 Gráfica condensadores de bypass -----</i>	70
<i>Fig. 31 Esquemático sensor de pulso -----</i>	71
<i>Fig. 32 Esquemático sensor de temperatura corporal -----</i>	72
<i>Fig. 33 Esquemático sensor de ambiente -----</i>	72
<i>Fig. 34 Esquemático sensor de luz -----</i>	73
<i>Fig. 35 Esquemático pantalla -----</i>	74
<i>Fig. 36 Conexión datasheet pantalla -----</i>	74
<i>Fig. 37 Esquemático Bluetooth -----</i>	75
<i>Fig. 38 Esquemático microcontrolador -----</i>	76
<i>Fig. 39 Gráfica tensión segura según la frecuencia -----</i>	76
<i>Fig. 40 Esquemático carga por inducción -----</i>	77
<i>Fig. 41 Conexiones datasheet carga inalámbrica -----</i>	77
<i>Fig. 42 Diseño PCB sensor de pulso, sensor de temperatura y Open Bioband-----</i>	79
<i>Fig. 43 Diseño acelerómetro en datasheet-----</i>	80
<i>Fig. 44 PCB sensor de pulso-----</i>	80
<i>Fig. 45 Diseño PCB sensor de temperatura -----</i>	80
<i>Fig. 46 Cableado y diseño LMT70 -----</i>	81
<i>Fig. 47 PCB sensor de temperatura corporal -----</i>	81

<i>Fig. 48 PCB detalles sensores de ambiente y luminosidad -----</i>	81
<i>Fig. 49 Diseño pantalla datasheet -----</i>	82
<i>Fig. 50 PCB detalle diseño pantalla -----</i>	82
<i>Fig. 51 PCB detalle microcontrolador -----</i>	82
<i>Fig. 52 Datasheet diseño bluetooth -----</i>	83
<i>Fig. 53 PCB diseño bluetooth-----</i>	83
<i>Fig. 54 PCB detalle carga por inducción -----</i>	84
<i>Fig. 55 Datasheet diseño PCB -----</i>	84
<i>Fig. 56 Pulsera versión cero con Microduino -----</i>	85
<i>Fig. 57 Prototipo Bioband con Arduino -----</i>	85
<i>Fig. 58 Open Bioband versión 1 – parte trasera -----</i>	86
<i>Fig. 59 Open Bioband versión 1 - parte delantera -----</i>	86
<i>Fig. 60 Sensor de pulso -----</i>	86
<i>Fig. 61 Sensor de temperatura-----</i>	86
<i>Fig. 62 Open Bioband versión 2 - parte trasera-----</i>	87
<i>Fig. 63 Open Bioband versión 2 - parte delantera -----</i>	87
<i>Fig. 64 Sensores de pulso (izquierda) y temperatura corporal (derecha)-----</i>	87
<i>Fig. 65 Img2Code. Conversor de bitmap a hexadecimal-----</i>	89
<i>Fig. 66 Pulsera en modo carga-----</i>	94
<i>Fig. 67 Pantalla principal-----</i>	94
<i>Fig. 68 Pantalla menú-----</i>	96
<i>Fig. 69 Pantalla biométrica -----</i>	98
<i>Fig. 70 Pantalla ambiente-----</i>	100
<i>Fig. 71 Pantalla movimiento -----</i>	102
<i>Fig. 72 Pantalla cronómetro-----</i>	104
<i>Fig. 73 Pantalla ajustes -----</i>	106
<i>Fig. 74 Pantalla juego -----</i>	108
<i>Fig. 75 Pantalla Game Over -----</i>	108
<i>Fig. 76 Pantalla iluminación -----</i>	110
<i>Fig. 77 Pantalla reloj -----</i>	112
<i>Fig. 78 Pantalla notificaciones -----</i>	114
<i>Fig. 79 Despiece 3D dispositivo -----</i>	121
<i>Fig. 80 Diseño pulsera en PLA -----</i>	122
<i>Fig. 81 Diseño correa en PLA -----</i>	122
<i>Fig. 82 Correas en Filaflex-----</i>	123
<i>Fig. 83 Pulsera en Filaflex -----</i>	123
<i>Fig. 84 Partes carcasa final impresas en PLA -----</i>	124
<i>Fig. 85 Partes carcasa final en Tinkercad -----</i>	124
<i>Fig. 86 Carcasa final y en 3D -----</i>	125
<i>Fig. 87 Correa final en Filaflex. Estándar 22 mm -----</i>	125
<i>Fig. 88 Sensores y luz externos a PCB principal -----</i>	126
<i>Fig. 89 PCB junto el resto de elementos -----</i>	126
<i>Fig. 90 Montaje metal sensor de temperatura y bobina -----</i>	126
<i>Fig. 91 Montaje sensores-----</i>	127
<i>Fig. 92 Montaje luz, batería y PCB principal-----</i>	127
<i>Fig. 93 Open Bioband -----</i>	128
<i>Fig. 94 Gráfica sensor de pulso -----</i>	130

Índice de tablas

<i>Tabla 1. Smartwatch -----</i>	<i>20</i>
<i>Tabla 2. Smartband -----</i>	<i>21</i>
<i>Tabla 3. Wearables Open Source-----</i>	<i>21</i>
<i>Tabla 4. Comparativa Smartwatch - Microcontrolador-----</i>	<i>22</i>
<i>Tabla 5. Comparativa Smartband - Microcontrolador-----</i>	<i>22</i>
<i>Tabla 6. Comparativa Open Source - Microcontrolador-----</i>	<i>23</i>
<i>Tabla 7. Comparativa Smartwatch - Batería -----</i>	<i>23</i>
<i>Tabla 8. Comparativa Smartband - Batería-----</i>	<i>23</i>
<i>Tabla 9. Comparativa Open Source - Batería-----</i>	<i>24</i>
<i>Tabla 10. Comparativa Smartwatch - Pantalla -----</i>	<i>24</i>
<i>Tabla 11. Comparativa Smartband - Pantalla-----</i>	<i>24</i>
<i>Tabla 12. Comparativa Open Source - Pantalla-----</i>	<i>25</i>
<i>Tabla 13. Comparativa Smartwatch - Sensores -----</i>	<i>25</i>
<i>Tabla 14. Comparativa Smartband - Sensores -----</i>	<i>26</i>
<i>Tabla 15. Comparativa Open Source - Sensores -----</i>	<i>26</i>
<i>Tabla 16. Comparativa Smartwatch - Diseño -----</i>	<i>27</i>
<i>Tabla 17. Comparativa Smartband - Diseño-----</i>	<i>27</i>
<i>Tabla 18. Comparativa Open Source - Diseño-----</i>	<i>28</i>
<i>Tabla 19. Características acelerómetros -----</i>	<i>34</i>
<i>Tabla 20. Pruebas acelerómetro -----</i>	<i>35</i>
<i>Tabla 21. Análisis clínico de los valores de SPO2 -----</i>	<i>36</i>
<i>Tabla 22. Ritmo cardíaco según la edad -----</i>	<i>37</i>
<i>Tabla 23. Características sensor de pulso -----</i>	<i>38</i>
<i>Tabla 24. Pruebas pulso -----</i>	<i>38</i>
<i>Tabla 25. Características sensores de temperatura corporal-----</i>	<i>41</i>
<i>Tabla 26. Pruebas sensor de temperatura corporal -----</i>	<i>41</i>
<i>Tabla 27. Características sensores GSR -----</i>	<i>43</i>
<i>Tabla 28. Pruebas GSR -----</i>	<i>43</i>
<i>Tabla 29. Características sensores ambiente-----</i>	<i>46</i>
<i>Tabla 30. Pruebas sensores ambiente -----</i>	<i>46</i>
<i>Tabla 31. Características sensores de luz -----</i>	<i>49</i>
<i>Tabla 32. Pruebas sensores de luz -----</i>	<i>49</i>
<i>Tabla 33. Características pantallas -----</i>	<i>51</i>
<i>Tabla 34. Pruebas pantallas -----</i>	<i>51</i>
<i>Tabla 35 Características Bluetooth -----</i>	<i>59</i>
<i>Tabla 36. Características microcontroladores -----</i>	<i>61</i>
<i>Tabla 37. Decisión final componentes -----</i>	<i>65</i>
<i>Tabla 38. Consumos teóricos-----</i>	<i>132</i>
<i>Tabla 39. Pruebas señal con carcasa-----</i>	<i>133</i>
<i>Tabla 40. Pruebas señal sin carcasa-----</i>	<i>133</i>

1. Introducción

1.1. Justificación

1.1.1. Objeto

El objeto del proyecto es el estudio del arte del actual mercado emergente de las tecnologías “wearables” conectadas al cuerpo humano, junto al diseño y desarrollo de un dispositivo autónomo, inalámbrico y Open Source para la monitorización de las constantes vitales del ser humano.

El desarrollo de la pulsera tocará un gran número de tecnologías y campos de aplicación, además de integrar tecnología Bluetooth Low Energy, para poder tener un control diario en un teléfono móvil, llevará diferentes sensores, desde posibles sensores de monitorización del ambiente (luz, humedad, temperatura ...) hasta de monitorización del usuario (temperatura corporal, SPO2, pulso ...).

Como resultado se espera un producto de bajo coste, para continuar con el compromiso de acercar la tecnología al mayor número de usuarios posible y que además sea libre, sobre el que cualquier desarrollador del mundo pudiera trabajar, tanto software como hardware, consiguiendo así una mejora continua del producto.

Para ello, se comenzará con una investigación de los diferentes wearables que existen en la actualidad y sus aplicaciones. De ahí se obtendrá información de las mejores características para la pulsera y con ello se realizarán pruebas con distintos sensores y módulos Bluetooth Low Energy. Por último, se terminará con el diseño, fabricación y prueba del producto desarrollado, probando su correcto funcionamiento.

1.1.2. Motivación del proyecto

En la actualidad, estamos en un mundo en el que la tecnología forma parte de nuestro día a día. Y en todo este desarrollo tecnológico, los wearables se han hecho un gran hueco siendo un indispensable para los deportistas.

Éstas son seis necesidades que satisfacen estos dispositivos:

1. Ayudan a que la vida diaria sea más fácil y eficiente (por ejemplo, para abrir puertas).
2. Reflejan el comportamiento y dan sugerencias para mejorar la calidad de vida (por ejemplo, mediante el monitoreo de la actividad física).
3. Se conectan de manera remota con familia, amigos o grupos para compartir experiencias o temas de interés mutuo (por ejemplo, compartir el tiempo que te ha costado hacer un recorrido en bicicleta).
4. Ayudan con tareas específicas a incrementar el rendimiento (por ejemplo, en aplicaciones para deportes).
5. Permiten el acceso a datos personales a cambio de un beneficio (por ejemplo, monitoreo de salud para las compañías de seguros).
6. Uso de los wearables para lucir y sentirse bien, expresando identidad personal.

PwC realizó un estudio a los consumidores estadounidenses sobre su comportamiento hacia el mercado wearable en 2014 [1] y 2016 [2]. Los principales objetivos del estudio eran identificar las tendencias, descubrir oportunidades y aplicaciones industriales. Dicho estudio en 2014 dice que el 56% de la población cree que la esperanza de vida crecerá 10 años gracias a la monitorización de nuestros signos vitales, el 46% cree que esta tecnología disminuirá la obesidad gracias a la monitorización del ejercicio y un 42% cree que la habilidad atlética mejorará gracias al monitoreo del progreso en deporte. Dos años después se ve como esta tendencia ha incrementado, hay mucha más población que utiliza wearables y ahora es el 70%, el que cree que nuestra vida mejorará gracias a los wearables.

En vista de cubrir las necesidades de las personas y mejorar su calidad de vida, se pretende desarrollar un sistema Open Source económico y con monitoreo del usuario. Un sistema que pueda llegar a distintas personas, personas que quieran usarlo para desarrollar, para aprender o para el monitoreo de sus constantes vitales.

1.2. Objetivos

El objetivo principal de este PFC es el desarrollo, construcción, programación y puesta a punto de un producto electrónico autónomo para la monitorización de parámetros biométricos, y que disponga de otras numerosas funcionalidades como la visualización de parámetros en tiempo real o el envío de datos de manera inalámbrica.

Podemos distinguir también una serie de objetivos secundarios relacionados con la finalización del proyecto:

- Investigación en un primer lugar todo lo relacionado con el ámbito del proyecto (redes inalámbricas, la tecnología de partida,...).
- Investigación y prueba de los sensores utilizados y su mercado. Adquiriendo de esta manera los conocimientos necesarios para la realización de elecciones óptimas.
- Compresión en profundidad el hardware utilizado (datasheet, documentación técnica...). Diseño de la electrónica asociada a los sensores y al resto de componentes el prototipo. Compromiso entre costes y calidad.
- Diseño de un sistema adecuado, garantizando su autonomía y minimizando el consumo.
- Evaluación y pruebas del sistema desarrollado.

1.2.1. Alcance

En estos últimos años, la tecnología basada en hardware y software libre, como por ejemplo Arduino [3] (sistema desarrollado basado en microcontroladores ATmega), ha iniciado un movimiento muy importante entre la sociedad electrónica, impulsado por las numerosas ventajas que este tipo de dispositivos facilitan. Ya sea en el ámbito personal como en el profesional, las personas comparten sus avances con el resto de la comunidad, lo que permite tener una referencia continua para la realización de distintos proyectos.

La domótica médica, robótica, todas las formas de instrumentación y las nanotecnologías son las principales ramas por las que se mueve el futuro de la raza humana. El mercado de la electrónica personal está creciendo de una manera incalculable, y servirá para crear nuevos productos que monitoricen nuestro entorno. Tendremos que ver y controlar nuestro entorno completo para vivir una vida sana y prolongada.

1.3. Antecedentes

Desde hace años el ser humano ha tratado de mejorar su cuerpo. Ya sea por motivos estéticos, como tatuajes o por motivos prácticos.

Actualmente, hay muchas grandes empresas abriéndose camino en el mundo del wearable. Sin embargo, su historia comenzó hace varios siglos.

Medir el tiempo siempre ha sido una de las preocupaciones del ser humano ya que es algo que facilita la organización. Los huevos de Nuremberg a principios del s.XVI, fue uno de los primeros dispositivos portables capaces de hacer esto aunque no muy precisos.

Después aparecieron los relojes de bolsillo y en 1812, apareció el primer reloj pulsera.

Pero el aspecto de wearable como se entiende ahora, no aparece hasta 1972, cuando Hamilton lanza Pulsar P1 por 2100 dólares.

Tres años después, la misma compañía lanzó Pulsar Calculator Watch, el primer reloj que incluía calculadora aunque sus botones solo podían ser pulsados por algunos dedos.

En 1982 Seiko lanza el primer reloj conectado, pero en este caso a un televisor. Contaba con dos pantallas, una para sintonizar distintas frecuencias del televisor y otra para la hora y alarmas.

Sin embargo, no es hasta el siglo XXI cuando se empieza a ver el verdadero potencial de los wearables.

En 2006, Nike en una colaboración con Apple desarrollan una aplicación gracias a unos sensores en las zapatillas y el iPod Nano. Dos años después, Fitbit apareció en el mercado marcando un antes y un después hasta la llegada de Pebble, el cual fue un éxito en Kickstarter y dio un paso más en las aplicaciones de estos dispositivos.

A día de hoy, estos dispositivos han evolucionado exponencialmente y todas las compañías van apareciendo con nuevas y distintas funcionalidades en sus dispositivos.

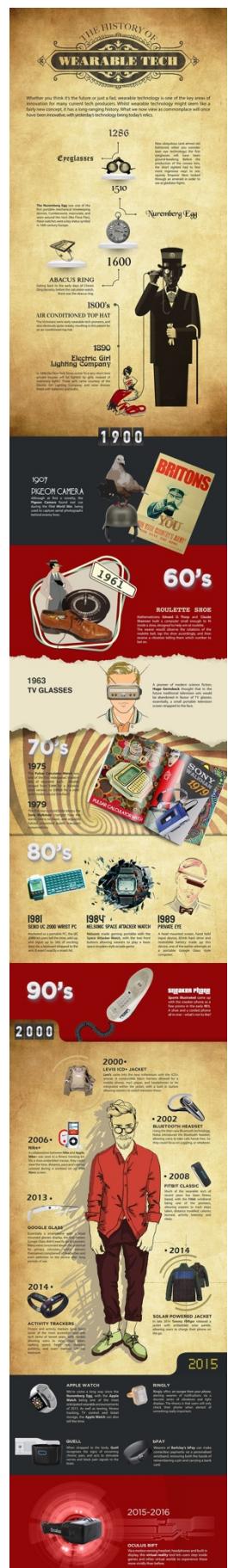


Fig. 1 Evolución wearables

1.4. Metodología y plan de trabajo

Se ha diseñado un plan de trabajo basado en los siguientes hitos:

1. Estado del arte: Realización de un estudio del estado del arte referido a equipos relacionados con la idea conceptual planteada.
2. Diseño de requisitos de aplicación: Diseño de requisitos de la aplicación y los equipos basados en el punto anterior.
3. Diseño preliminar: Realización de un diseño preliminar del Hardware y software basado en componentes comerciales.
4. Diseño final basado en PCB propia: Tecnologías y módulos implementados, en base a los requisitos.
5. Desarrollo: La fase de desarrollo se dividirá en 3 sub-tareas predeterminadas:
 - a. Desarrollo Hardware: Se integrará en una sola electrónica, todos los componentes necesarios para integrar los módulos y sensores empleados.
 - b. Desarrollo software: Se desarrollará un software capaz de mostrar todos los parámetros en pantalla.
 - c. Desarrollo Carcasa: desarrollo de una carcasa realizada en impresión 3D para proteger la electrónica.
6. Integración pruebas y resultados: Una vez especificados los requisitos y desarrollado el prototipo en cuestión de software y hardware, se realizarán pruebas para validar la funcionalidad de la aplicación y verificar que su funcionamiento es correcto.
7. Conclusiones técnicas y trabajo futuro.

2. Campo de aplicación

2.1. Posibles aplicaciones del sistema

El proyecto se enfoca a personas, desarrolladores e innovadores, inmersos en el mundo Open Source que quieran aprender y desarrollar proyectos en esta línea, sirviéndoles de una herramienta con la que empezar a desarrollar y poder trabajar con medidas biométricas y ambientales en sus proyectos y/o productos.

El proyecto se enfoca al uso de particulares que deseen monitorizar distintos parámetros médicos o del ambiente o artistas que quieran dar vida a su instalación a través de dichos parámetros.

Los beneficios del sistema son entre otros:

- Monitorización de parámetros biomédicos.
- Monitorización del ambiente del usuario.
- Interacción remota gracias a la utilización de comunicación inalámbrica.
- Implementación de rutinas para auto-diagnóstico dado su carácter “libre” tanto en hardware como en software.
- Posibilidad de monitorización continua.
- Obtención de los parámetros biométricos en bruto, los que son aplicables para interactuar con proyectos de diversas índoles (como por ejemplo proyectos de arte).

2.2. Estado del arte

2.2.1. Wearables

La tecnología wearable consiste en llevar en la ropa o en los complementos como relojes o collares distintos sensores o características para hacernos nuestra vida diaria más fácil. La principal ventaja que suponen los wearables es que no tenemos que llevarlos en la mano o en el bolsillo, sino que vienen directamente con nosotros, ofreciéndonos todas las utilidades de las que disponen.

Otra ventaja de este tipo de innovadores dispositivos es que además de ofrecernos información sobre nuestra actividad, también interactúan con nosotros. Si por ejemplo realizamos ejercicio, además de mostrarnos nuestras pulsaciones, nos avisarán si nuestro ritmo es demasiado alto. Otro ejemplo es cuando utilizamos nuestros dispositivos móviles como GPS, pues gracias a los wearables no hará falta mirar el mapa que muestra la pantalla, sino que ligeras vibraciones nos informarán hacia dónde tenemos que girar.

Los wearables, como vemos no son exclusivos en el mundo de la salud, pero es uno de los escenarios en los que mayor evolución pueden tener.

En general, y según el modelo permiten medir la frecuencia cardiaca, contabilizar el número de pasos que hemos dado al cabo del día, contabilizar los km recorridos en una actividad física de tipo carrera o ciclismo (y obtener el correspondiente consumo de calorías) y cuantificar el número de horas que dormimos al día y la calidad de este descanso (reflejando el porcentaje de tiempo inmóvil). Se complementan con una app en nuestro Smartphone, con el que se comunican vía bluetooth, en la que podemos ver el histórico de nuestros progresos, consultar todo tipo de estadísticas y gestionar nuestros objetivos mediante técnicas de gamificación. Estamos hablando de marcas y modelos como Fitbit, Jawbone, Withings, Nike Fuelband, Misfit Shine, Polar Loop, Garmin, y modelos que mezclan la función de pulsera cuantificadora con las funciones de smartwatch, como Sony Smartband, Samsung Gear Fit, LG LifeBand Touch o Huawei TalkBand B1.

Éstas básicamente se componen de acelerómetros para la cuantificación del movimiento, algunos GPS para cálculo de rutas, monitores ópticos de frecuencia cardíaca que permiten determinar el pulso a través de un proceso llamado fotopletismografía, algunos sensores de respuesta galvánica de la piel que permiten detectar por ejemplo el inicio de una actividad física, termómetros, sensores de detección de luz ambiental, sensores de luz ultravioleta, sensores de bioimpedancia, etc.

Por otro lado, existen grandes apuestas de los principales proveedores tecnológicos para fomentar el seguimiento y la gestión de la salud personal, y por lo tanto, su interacción con los wearables. Éstos son Apple con Health Kit [4], y Google a través de Google Fit [5]. En ambos casos se propone un entorno de gestión de la salud para la integración de dispositivos y apps de terceros con los terminales móviles iOS y Android, los cuales incluyen por defecto estos entornos. Esto implica que los dos principales sistemas operativos de Smartphone y tablets del mercado se están posicionando para actuar de “pasarela” e integradores, inicialmente, de todo el mercado de wearables relacionados con la salud.

Otra opción de estos dispositivos, como Sony o Pebble, es la de crear Smartwatch que son Open Source en software, es decir, liberan su software a través de una API para que los desarrolladores desde su casa puedan crear nuevas aplicaciones, y de esta forma produzcan mejoras en sus productos.

Por último, otra de las opciones que nos proporcionan dichos wearables, es el uso de los datos para instalaciones artísticas. Por ejemplo, un estudio artístico de Londres [6] desarrolló unas pulseras con GSR (Galvanic Skin Response). Estas pulseras fueron dadas a todo el público y gracias a ello, podían ver en tiempo real las emociones del público al ver los videos proyectados en pantalla. Dichas emociones eran representadas por colores desde más calmado (en azul) a más excitado (en rosa).



Fig. 2 Colores dependiendo de las emociones del público



Fig. 3 Representación datos de las pulseras. Festival Cannes

Visto que son los wearables y sus diferentes aplicaciones centrándonos en los wearables de pulsera, los clasificaremos en tres categorías dependiendo de su uso:

- a) **Smartwatch.** Dichos dispositivos además de proporcionar información sobre el rendimiento del usuario durante el ejercicio físico, ofrecen información del teléfono móvil y es utilizado más por sus funcionalidades extras que por analizar el estado físico de la persona.
- b) **Smartband.** Dichos dispositivos suelen ser comprados por ciclistas o atletas en general. Contienen sensores para llevar un seguimiento de su rendimiento y estado físico.
- c) **Open Source.** Dichos dispositivos proporcionan datos de los sensores sin procesamiento para permitir a los investigadores aplicar sus algoritmos, seguir desarrollando nuevas aplicaciones o que desarrolladores hagan sus propias aplicaciones para uso propio.

En la siguiente sección, se presentan los dispositivos a analizar según las categorías descritas y unas tablas comparativas donde se analizarán distintos aspectos técnicos de éstas como: Microcontrolador, batería, pantalla, sensores y diseño.

2.2.1.1. Smartwatch

Apple Watch		Sony Smartwatch 3	
Pebble Time		Asus Zen Watch	
Pebble Steel		Huawei Watch	
Moto 360		Samsung Gear S	
LG G Watch R			

Tabla 1. Smartwatch

Pulsera biométrica Open Source para la monitorización del usuario

2.2.1.2. Smartband

Xiaomi Mi Band 1s		Jawbone UP3	
Garmin VivoSmart		Withings Pulse O2	
Fitbit Charge HR			

Tabla 2. Smartband

2.2.1.3. Open Source

Meta Health		Retrowatch	
Totem		BlueIOT	
Hexiwear			

Tabla 3. Wearables Open Source

2.2.1.4. Tablas comparativas

MICROCONTROLADOR

- Smartwatch

	Apple Watch	Pebble Time	Pebble Steel	Alcatel One Touch	Moto 360	LG G Wath R	Sony Smartwatch 3	Asus Zen Watch	Huawei Watch	Samsung Gear S
CPU	Apple 51 chip	ARM Cortex-M4 COU 199Mhz	ARM Cortex-M3 CPU 64Mhz	STM 429	1 Ghz Cortex A8	Qualcomm Snapdragon Quad-core 1.2 Ghz	1.2 Ghz Quad ARM A7	1.2 Ghz Qualcomm Snapdragon 400	1.2 Ghz Qualcomm Snapdragon 400	Dual Core 2Ghz Dual-core Snapdrago n 400 processor
Almacenamiento	8GB limitado 2GB música y 75MB fotos	No limites apps	Límite 8 apps	4 GB	4GB	4GB	4GB	4GB	4GB	4GB
RAM			128KB		512MB	512MB				512MB

Tabla 4. Comparativa Smartwatch - Microcontrolador

- Smartband

	Xiaomi Mi Band	Garmin VivoSmart		Fitbit Charge HR	Jawbone UP3	Fitbit Flex	Withings Pulse O2
CPU	ARM Cortex M0 32bit	ARM Cortex M0 32bit		ARM Cortex M3	ARM Cortex M3	ARM Cortex M3 32MHz	
Almacenamiento	256KB	256KB		512KB	8MB	128KB	
RAM	16KB	16KB				16KKB	

Tabla 5. Comparativa Smartband - Microcontrolador

Pulsera biométrica Open Source para la monitorización del usuario

- Open Source

	Meta Health	Totem	Hexiwear	Retrowatch	blueIOT
CPU	ARM Cortex-M0	ARM Cortex-M0	ARM Cortex-M4 120MHz	Arduino Pro Mini ATmega328 8MHz	ATmega328 8MHz
Almacenamiento	128-256KB		1MB	32KB	32KB
RAM	16-32KB		256KB	2KB	2KB

Tabla 6. Comparativa Open Source - Microcontrolador

BATERIA

- Smartwatch

	Apple Watch	Pebble Time	Pebble Steel	Alcatel One Touch	Moto 360	LG G Wath R	Sony Smartwatch 3	Asus Zen Watch	Huawei Watch	Samsung Gear S
Batería	205mAh	150	130	210	320	410	420	370	300	300
Duración	18h (hasta 72h)	5-7 días	5-7 días		1 día	2 días				2 días
Cargador	Inalámbrico	Magnético	Magnético	Integrado USB	Inalámbrico	Magnético	Micro USB	Magnético	Magnético	Cargador de base

Tabla 7. Comparativa Smartwatch - Batería

- Smartband

	Xiaomi Mi Band 1s	Garmin VivoSmart	Fitbit Charge HR	Jawbone UP3	Fitbit Flex	Withings Pulse O2
Batería	45mAh	200mAh		38mAh		
Duración	20 días	7 días	5 días	7 días	5 días	14 días
Cargador	Magnético	Magnético	Inalámbrico	USB magnético	Cargador USB	Micro-USB a USB

Tabla 8. Comparativa Smartband - Batería

- Open Source

	Meta Health	Totem	Hexiwear	Retrowatch	bluelOT
Batería	CR2032	CR2032	190mAh	110mAh	CR2031
Duración		40 días		7 horas	
Cargador	-		Integrado USB		

Tabla 9. Comparativa Open Source - Batería

PANTALLA

- Smartwatch

	Apple Watch	Pebble Time	Pebble Steel	Alcatel One Touch	Moto 360	LG G Wath R	Sony Smartwatch 3	Asus Zen Watch	Huawei Watch	Samsung Gear S
Tipo	Retina	Color e-ink	B&W e-ink	LCD	LCD	OLED	LCD transreflective	AMOLED	AMOLED	Super AMOLED
Medida	38mm:1.5" 42mm:1.65"	1.25"	1.25"	1.22"	1.56"	1.3"	1.6"	1.64"	1.4"	2"
Resolución	340x272 390x312	144x168	204x204	320x290	320x320	320x320	320x320	320x320	400x400	480x360
Cristal	Ion-X-Glass	Gorilla 3	Gorilla	Corning Glass	Gorilla 3	Gorilla 3	Gorilla 3	Gorilla 3	Sapphire	Gorilla 3

Tabla 10. Comparativa Smartwatch - Pantalla

- Smartband

	Xiaomi Mi Band 1s	Garmin VivoSmart	Fitbit Charge HR	Jawbone UP4	Fitbit Flex	Withings Pulse O2
Tipo	No, leds colores	OLED	OLED	No, leds Azul-sueño Naranja-Actividad Blanco-Notificac	No, 5 leds	OLED
Medida	-	1.35"	0.7"	-	-	1.3x7"
Resolución	-	128x16	32x72	-	-	128x32
Cristal	-	-	-	-	-	-

Tabla 11. Comparativa Smartband - Pantalla

Pulsera biométrica Open Source para la monitorización del usuario

- Open Source

	Meta Health	Totem	Hexiwear	Retrowatch	bluelOT
Tipo	-	-	OLED	OLED	Sharp Memory LCD
Medida	-	-	-	0.96"	1.3"
Resolución	-	-	-	128x64	96x96
Cristal	-	-	-	-	-

Tabla 12. Comparativa Open Source – Pantalla

SENSORES

- Smartwatch

	Apple Watch	Pebble Time	Pebble Steel	Alcatel One Touch	Moto 360	LG G Wath R	Sony Smartwatch 3	Asus Zen Watch	Huawei Watch	Samsung Gear S
Pulso	x				x	x	x	x	x	x
SPO2	x									
Acelerómetro	x	x	x	x	x	x	x	x	x	x
Giroscopio	x			x		x	x	x	x	x
Barómetro	x					x	x	x	x	x
Sensor de luz	x	x	x		x					x
Tacto	x									
Altímetro				x						
Compás		x	x							x
UV										x

Tabla 13. Comparativa Smartwatch - Sensores

- Smartband

	Xiaomi Mi Band 1s	Garmin VivoSmart	Fitbit Charge HR	Jawbone UP3	Fitbit Flex	Withings Pulse O2
Pulso	x		x	x		x
SPO2						x
Acelerómetro	x	x	x	x	x	x
Giroscopio						
Barómetro						x
Sensor de luz						
Tacto						
Altímetro			x			x
Compás						
UV						

Tabla 14. Comparativa Smartband - Sensores

- Open Source

	Meta Health	Totem	Hexiwear	Retrowatch	bluelOT
Pulso	x		x		
SPO2					
Acelerómetro	x	x	x		x
Giroscopio	x	x	x		
Barómetro			x		x
Sensor de luz			x		
GSR					
Altímetro					
Compás			x		
UV					x

Tabla 15. Comparativa Open Source - Sensores

DISEÑO

- Smartwatch

	Apple Watch	Pebble Time	Pebble Steel	Alcatel One Touch	Moto 360	LG Watch R	Sony Smartwatch 3	Asus Zen Watch	Huawei Watch	Samsung Gear S
Correa	Custom	Standard 22mm	Custom	Custom	22mm-custom	Standard 22mm	Custom	Standard 22mm	Standard 18/22mm	Custom
Carcasa	Sport: Aluminio Watch: acero inoxidable	Plástico o inoxidable	acero inoxidable	acero inoxidable	acero inoxidable	acero inoxidable	Plástico o inoxidable	acero inoxidable	acero inoxidable	acero inoxidable
Dimensiones	38mm: 38.6x33.3x 10.5 42mm: 42x35.9x 10.5	40.5x37.5x 9.5	38x34.1x1 0.3	41.8x10.5	46x11.5	46.4x53.6x 9.7	36 x 51 x 10	39.9x51x7.9	42 x 11.3	39.9x58.1x 12.5
Peso	25 - 69g (depende modelo)	42.5g	56g	60g	49g	62g	45g	50g	-	67-84g
Precio	350€	70€	200€	200€	320€	150€	100€	90€	220€	340€

Tabla 16. Comparativa Smartwatch - Diseño

- Smartband

	Xiaomi Mi Band 1s	Garmin VivoSmart	Fitbit Charge HR	Jawbone UP3	Fitbit Flex	Withings Pulse O2
Correa	Silicona	Goma	-	Caucho	goma	Silicona
Carcasa	Aluminio y policarbonato	-	-	Aluminio	Acero inoxidable	metal
Dimensiones	22.5x13.6x9.9 mm	Pequeña: 127-172mm Grande: 155-221 mm	204.4x21.3x12.7	220x12.2x3-9.3mm	204.4x11x12.7	43x22x8mm
Peso	-	Pequeña: 18.7g Grande: 19g	23.8g	29g	23.5g	11.8g
Precio	27€	120€	150€	66€	66€	89€

Tabla 17. Comparativa Smartband - Diseño

- Open Source

	Meta Health	Totem	Hexiwear	Retrowatch	bluelOT
Correa	-	-	-	-	-
Carcasa	-	-	-	-	-
Dimensiones	10x5x10mm	4.7x2.9x0.7mm	-	3.4x3.2x1.2 mm	-
Peso	28g	-	-	-	-
Precio	-	-	-	-	-

Tabla 18. Comparativa Open Source - Diseño

3. Especificación de los requisitos de aplicación

Una vez especificado el objetivo de este proyecto, y analizados los puntos planteados en el estado del arte, se profundizará en las características de la aplicación a desarrollar y se fijarán unos objetivos concretos relativos a cada punto técnico del proyecto.

3.1. La tecnología de partida: ARDUINO

Arduino es una plataforma de hardware libre (Open Source) basada en la arquitectura de los microcontroladores AVR de la empresa ATMEL. Se trata de una sencilla placa con entradas y salidas (E/S) analógicas y digitales que incluye un entorno de desarrollo para implementar el código en lenguaje C-C++ y diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

El principal objetivo del proyecto es que todo lo desarrollado en este proyecto sea con el IDE de Arduino, de hardware y software libre y que esté al alcance de cualquier persona, bien para que aprenda, para que le pueda servir en sucesivos desarrollos de la misma o en sus propios desarrollos.

Para conocer más sobre dicha plataforma, sus ventajas y software puede irse al ANEXO 3.

3.2. Especificaciones básicas iniciales

El dispositivo será un módulo que funcionará de forma autónoma y que tendrá las siguientes características:

La tarjeta electrónica encargada de la recogida de las señales, tanto de los sensores, como de la comunicación inalámbrica, contará con al menos X entradas analógicas y X entradas/salidas digitales con 2 puertos UART y, que permita la simulación de UARTs virtuales mediante E/S digitales. (las entradas y salidas dependerán del número de sensores, actuadores .. del sistema)

El micro trabajará al menos a 8MHz y tendrá una batería de litio recargable. La tensión de trabajo del microprocesador será de 3.3V. Se procurará que la electrónica sea lo más compacta posible para que quede una pulsera lo más pequeña posible. El micro será compatible con el SDK de Arduino, y habrá que contar con un puerto ICSP para su programación.

3.3. Especificaciones basadas en el estudio de mercado

Según el estudio de mercado podemos observar que los Smartband, a excepción de Withings, compañía comprada recientemente por Nokia Health [7] se centran principalmente en la medición de la actividad física por medio de sensores de movimiento y pulso y los Smartwatch tienen muchas más funcionalidades aunque por el contrario, su precio es mayor.

Por otro lado, a excepción de alguno como Pebble [8] o Samsung [9] que es de software libre, ninguno proporciona una plataforma abierta a los desarrolladores de hardware y software. Hexiwear [10] sería uno de los desarrollos que más se asemeja al dispositivo a desarrollar. Un dispositivo capaz de capturar diferentes señales físicas y del ambiente y proporcionan un acceso abierto a los datos capturados (la parte hardware no es totalmente libre). Por otro lado, el microcontrolador se baraja la posibilidad de un Cortex o un ATmega, ambos compatibles con Arduino, pantalla OLED o LCD, batería de unos 200mAh vistos los rangos de otros wearables y estudiar la posibilidad de carga inalámbrica, ya que ningún wearable Open Source tiene esta característica.

3.3. Especificaciones basadas en el estudio de la normativa

- Normativa FDA wearables salud

Dado el incremento de tecnologías wearables relacionadas con la salud y el bienestar, la FDA (agencia estadounidense del medicamento) ha publicado una normativa no jurídicamente vinculante, pero que sí establece unos criterios de legislación.

Debido a que la pulsera desarrollada no es para diagnosticar o tratar una enfermedad, sino para monitorizar al usuario y el ambiente en el que se rodea, se trataría de un producto apto según dicha normativa. Se puede ver en el ANEXO 2.

3.4. Especificaciones basadas en el estudio de las comunicaciones inalámbricas

El módulo de comunicaciones utilizará tecnología 2,4 GHz Bluetooth 4.0 (BLE). Dicho módulo debe ser lo más compacto posible y de bajo consumo. Además incorporará un sistema que permita actualizar el firmware en caso necesario.

Además, el módulo será Open Source y permitirá realizar desarrollos en su funcionalidad, así como en las características de las tramas enviadas, y todo ello presentará documentación accesible al público.

3.5. Diseño final

Se partirá de los requisitos previamente comentados, aunque a lo largo del proceso del desarrollo pueden ser rechazados o mejorados en función de la situación y los datos analizados.

Tras los distintos estudios se han añadido los siguientes requisitos de cara al diseño final:

- Una pulsera en la que se intentará un equilibrio entre los sensores y su tamaño, ya que se quiere que haya gran variedad de sensores sin que ésta sea demasiado grande.
- Trabajar con tecnología SMD únicamente dado el pequeño tamaño del dispositivo y el lugar donde va a ir colocado. Esto reducirá en gran parte las dimensiones del prototipo, y facilitará el diseño del ruteado.
- Integrará distintos sensores de ambiente y biométricos:
 - Sensores ambiente (luz, temperatura, humedad, altitud, presión)
 - Sensores biométricos (pulso, spo2, GSR, temperatura corporal)
 - Acelerómetro
- Con posibilidad de integrar carga inalámbrica ya que ninguna pulsera Open Source tiene esta característica.
- Pantalla donde se mostrarán los datos en tiempo real.
- Módulo Bluetooth Low Energy con antena y certificado para que a futuro pueda tener su propia app o enviar datos al ordenador de forma inalámbrica.

4. Diseño hardware

El hardware de este proyecto conforma la creación y diseño del dispositivo de monitorización del usuario y su ambiente. En esta placa se incluyen los componentes necesarios para su correcto funcionamiento.

4.1. Metodología

Partiendo de la información proporcionada por los fabricantes de los distintos sensores y los conocimientos adquiridos de la investigación, se realizó el diseño del circuito impreso de la placa. Se han seguido una serie de pautas para su diseño, con el fin de evitar posibles problemas de diseño o fabricación:

- Plantear el diseño en papel
- Estudio intensivo de los circuitos del Datasheet con el fin de encontrar la configuración que se adapta mejor a nuestra aplicación.
- Cumplir las normas básicas de ruteado de circuitos (distancia entre pistas, tamaño de pistas...)
- Fijar unas dimensiones de base sobre las que trabajar.
- Una buena elección de componentes, documentada, para evitar problemas futuros.

Para el diseño de la placa por ordenador se ha recurrido al programa Eagle [11], el cual, aunque no es libre, dispone de una licencia gratuita para uso doméstico. Es un programa sencillo, muy utilizado en el mundo del hardware libre.

Una vez diseñadas las placas, y fabricadas por una empresa, se procedió a soldar los componentes y probar paso a paso su correcto funcionamiento.

4.2. Diagrama de bloques del sistema

El sistema, alimentado por una batería, consiste en un microcontrolador que procesa datos de distintos tipos de sensores: sensores biométricos, acelerómetro y sensores ambientales. Por otro lado, estos datos pueden ser representados a través de una pantalla o por medio de Bluetooth Low Energy en el móvil o en un ordenador.

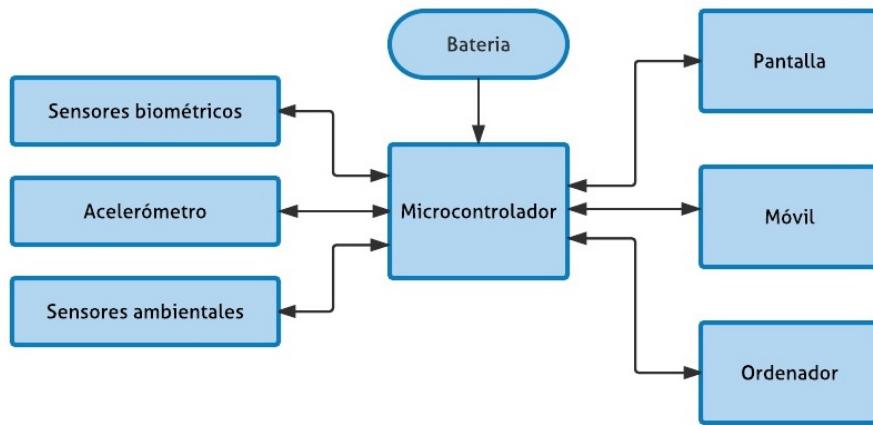


Fig. 4 Diagrama de bloques del sistema

4.3. Selección de componentes

4.3.1. Sensores

4.3.1.1. Acelerómetro

Los acelerómetros son dispositivos que miden la aceleración, que es la tasa de cambio de la velocidad de un objeto. Esto se mide en metros por segundo al cuadrado (m/s^2) o en las fuerzas G (g). La sola fuerza de la gravedad para nosotros aquí en el planeta Tierra es equivalente a $9,8\text{ m/s}^2$, pero esto varía ligeramente con la altitud (y será un valor diferente en diferentes planetas, debido a las variaciones de la atracción gravitatoria). Los acelerómetros son útiles para detectar las vibraciones en los sistemas o para aplicaciones de orientación.

Los acelerómetros se comunican a través de un convertidor analógico, digital, o interfaz de conexión modulada por ancho de impulsos (PWM) .

Los acelerómetros con una interfaz analógica entregan un voltaje proporcional a la aceleración en cada uno de sus ejes (hablando de uno de 3 ejes), que normalmente fluctúan entre tierra y el valor de alimentación Vcc. Éstos suelen ser más baratos que los digitales y mucho más fáciles de usar.

Los acelerómetros con una interfaz digital pueden comunicarse a través de los protocolos de comunicación de SPI o I²C. Estos tienden a tener más funcionalidades y ser menos susceptibles al ruido que acelerómetros analógicos .

Dispositivo	Rango	Interfaz	Ejes	Requerimientos de energía	Características especiales
BMA220	±2g ±4g ±8g ±16g	Digital SPI/ I ² C	3	2-3.6V Bajo consumo	Un pin de interrupción Detección de golpe
MMA8452	±2g ±4g ±8g	Digital I ² C	3	1.95-3.6V 6-165uA	2pines de interrupción Detección de golpe Reconocimiento orientación
ADXL362	±2g ±4g ±8g	Digital SPI	3	1.6-3.5V Bajo consumo	2pines de interrupción Detección de caída libre Sensor de temperatura
ADXL345	±2g ±4g ±8g ±16g	Digital SPI/ I ² C	3	2-3.6V 0.1-40uA	2pines de interrupción Detección de golpe/ doble golpe Detección de caída libre

Tabla 19. Características acelerómetros

A la hora de elegir entre un acelerómetro u otro debemos considerar varios requerimientos: entre los principales se encuentra el nivel de alimentación y el tipo de comunicación, además del alcance, que puede variar de $\pm 1\text{g}$ hasta $\pm 250\text{g}$ (utilizando un acelerómetro de gama pequeña proporcionará datos más detallados que uno de 250g) o algunas características especiales como puede ser la detección de caída libre, la compensación de temperatura o la detección de un golpe con el dedo.

El uso de un acelerómetro nos permite conocer datos como el número de pasos diarios, las calorías que ha gastado el usuario o la monitorización del sueño.

Pruebas acelerómetro

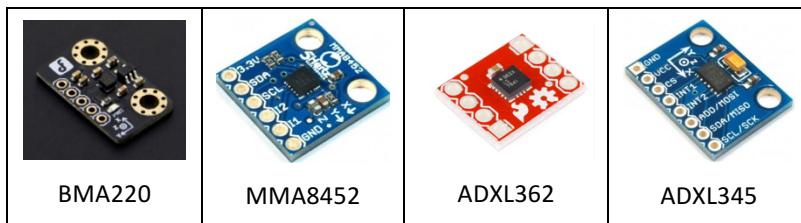


Tabla 20.
acelerómetro

Pruebas

Para las pruebas con los acelerómetros se utilizó un programa hecho en Processing [12], un software de visualización de datos, donde se podía ver en tiempo real cómo afectaba los cambios de posición en el sensor.

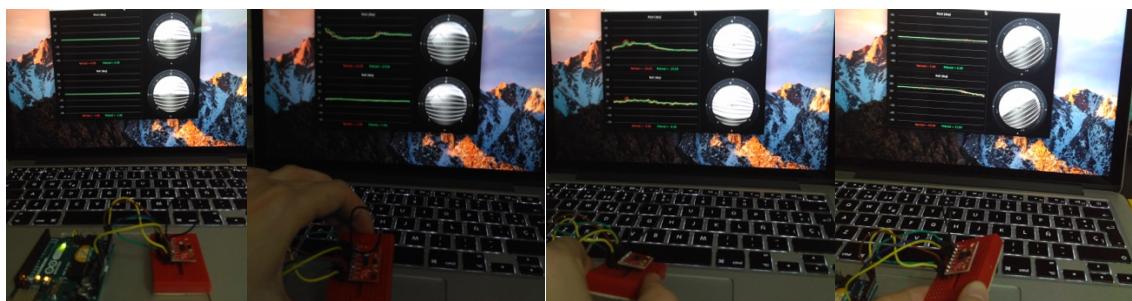


Fig. 5 Pruebas acelerómetro

Entre el sensor adxl362 y el adxl345 no había grandes diferencias en cuanto a precisión, funcionalidades o tamaño, pero finalmente por documentación encontrada y precio se optó por usar el sensor adxl345.

4.3.1.2. Pulso/SPO₂

La pulsioximetría es la medición no invasiva del oxígeno transportado por la hemoglobina en el interior de los vasos sanguíneos. Se realiza con un aparato llamado pulsioxímetro, que tiene un emisor de luz y un fotodetector, mediante los cuales calcula la absorción de oxígeno por parte de la sangre.

La medición de este parámetro es interesante para pacientes con medidas de oxígeno inestables. En la tabla es posible observar la peligrosidad de estos descensos de la saturación de oxígeno en sangre.

% Saturación	Actuación
> 95 %	No actuación inmediata.
95-90 %	Tratamiento inmediato y monitorización de la respuesta al mismo, según ésta, valorar derivación al hospital. Los pacientes con enfermedad respiratoria crónica toleran bien saturaciones en torno a estos valores.
< 90 %	Enfermo grave. Hipoxia severa. Oxigenoterapia + tratamiento y traslado al hospital.
< 80 %	Valorar intubación y ventilación mecánica.

Tabla 21. Análisis clínico de los valores de SPO₂

El punto crítico que debe dar la señal de alarma es el de saturaciones inferiores al 95% (inferiores al 90 o 92% cuando existe patología pulmonar crónica previa) estos pacientes deben recibir tratamiento inmediato.

Pulsera biométrica Open Source para la monitorización del usuario

De este mismo sensor podemos obtener la medida de pulso. El pulso son los “saltos” palpables del flujo sanguíneo que se aprecian en diversos puntos del cuerpo. Es obtenido por lo general en partes del cuerpo donde las arterias se encuentran más próximas a la piel, como en las muñecas o el cuello. Es un indicador del estado circulatorio.

El ritmo cardíaco es medido en pulsos por minuto. En la tabla 22 aparecen los valores normales según la edad.

Edad	Ritmo cardíaco (latidos por minuto)
Lactantes	120-160
Niños pequeños	90-140
Preescolares	80-110
Niños en edad escolar	75-100
Adolescentes	60-90
Adultos	60-100

Tabla 22. Ritmo cardíaco según la edad

Un fotopletismógrafo es un dispositivo utilizado para capturar el pulso de una persona tras colocar un emisor de luz (infrarroja o visible) y un receptor en contraposición con un obstáculo en medio (el cual sería en la mayoría de los casos la punta de los dedos). Cuando se produce un bombeo de sangre del corazón, la presión sanguínea en el dedo aumenta por lo que se produce una minúscula interrupción del haz de luz.

La señal de pulso del corazón que sale de un fotopletismógrafo es una fluctuación en el voltaje y tiene una forma de onda predecible como se muestra en la figura conocida como Registro de actividad eléctrica cardiaca, ECG, EKG o registro electrocardiográfico.

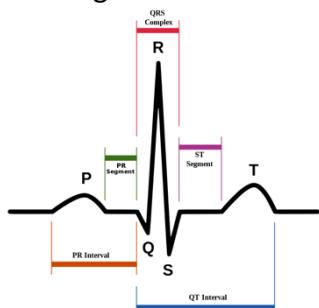


Fig. 6 Señal típica de electrocardiograma

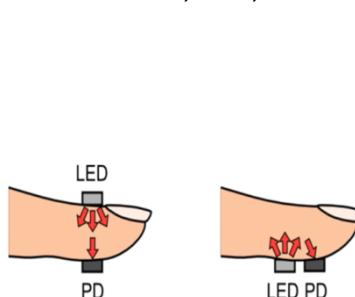
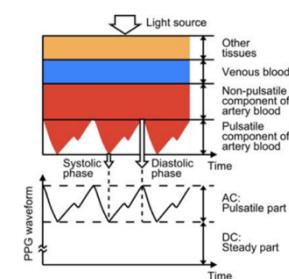


Fig. 7 Funcionamiento fotopletismógrafo



Dispositivo	Interfaz	Requerimientos de energía	Características
 Finger clip	Digital I ² C/SPI	3-3.6V	Se basa en el sensor PAH8001EI-2G, que integra una luz verde para el cálculo del pulso
 Pulse sensor	Analógico	3-5V	Consta de un led verde, un sensor de luz apds9008 y un circuito de amplificación para el cálculo de pulso
 MAX30101	Digital I ² C	1.7-2V Led driver 3.1-5.25V	Pequeño e integra 3 leds (IR, rojo y verde) para el cálculo de pulso y SPO2

Tabla 23. Características sensor de pulso

Pruebas pulso



Tabla 24. Pruebas pulso

Se han realizado pruebas con los distintos sensores de pulso arriba y debajo de la muñeca.

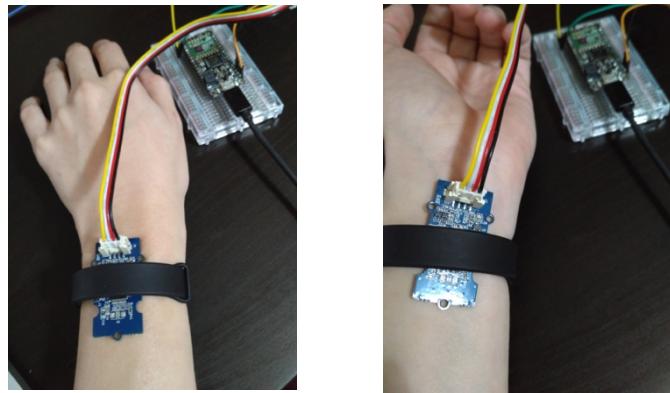


Fig. 8 Pruebas pulso

Para probar el sensor max30101 se diseñó una PCB y se fabricó en el OpenArt [13], un Fablab de Zaragoza de uso libre a la ciudadanía.

Se utilizó el programa Eagle para su diseño y el programa Circuit Pro, propio de la máquina Protomat E33 [14] para su fabricación.

Durante el desarrollo de la misma, se aprendió su uso y se hizo un tutorial para que cualquier persona que no supiera usarla pudiera seguir los pasos y fabricarse sus propios diseños.

En este [enlace](#) se puede ver tanto las imágenes como el tutorial con el ejemplo de un corazón que late en la oscuridad gracias a la programación de un Attiny.

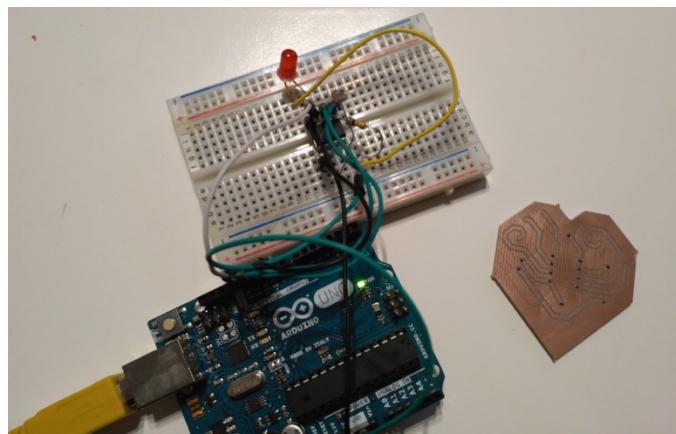


Fig. 9 Fabricación PCB de prueba y bootloader ATTiny

Este es el resultado de la fabricación de la PCB y soldado de los componentes:

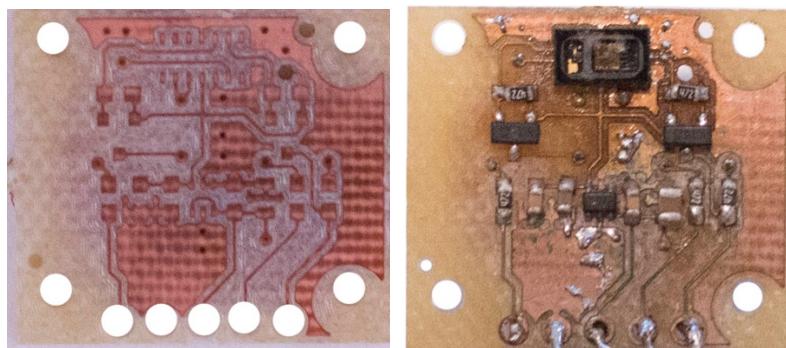


Fig. 10 PCB sensor de pulso MAX30101

Tras las distintas pruebas con ambos sensores, se decidió utilizar el MAX30101 debido a su tamaño, ya que es un sensor pequeño, que te mide tanto pulso como SPO2 con sus 3 leds y por su precisión.

4.3.1.3. Temperatura corporal

La temperatura corporal depende de la zona del cuerpo donde se realiza la medición, además de la hora del día y el nivel de actividad de la persona. Esto es debido a que diferentes partes del cuerpo tienen diferentes temperaturas ya que están más o menos expuestas a la temperatura ambiente.

La temperatura corporal normal aceptada (tomada internamente) es 37.0 ° C (98.6 ° F). En los adultos sanos, la temperatura del cuerpo oscila alrededor de 0.5 ° C (0,9 ° F) durante todo el día, con temperaturas más bajas en la mañana y temperaturas más altas en la tarde y la noche, a medida que cambian las necesidades y actividades del cuerpo. Es de gran importancia médica medir la temperatura corporal ya que una serie de enfermedades se acompañan de cambios característicos en la temperatura corporal. Del mismo modo, el curso de ciertas enfermedades se puede monitorizar midiendo la temperatura corporal, y la eficacia de un tratamiento iniciado puede ser evaluada por el médico.

Este sensor es de los más complicados a la hora de tener una medida exacta ya que tiene que estar en buen contacto con la piel y la muñeca no es uno de los sitios más comunes donde medir la temperatura corporal.

Dispositivo	Exactitud	Interfaz	Requerimientos de energía	Características
	0.1º	Digital I²C	2.7-3.3V 600uA	Especial para wearables y aplicaciones médicas
	0.05º	Analógico	2-5.5V 9.2uA	Especial para wearables y aplicaciones médicas
	0.1º	Analógico	2.7-5V	Especial para aplicaciones médicas. Aconsejable utilizar en el dedo de la mano o el pie

Tabla 25. Características sensores de temperatura corporal

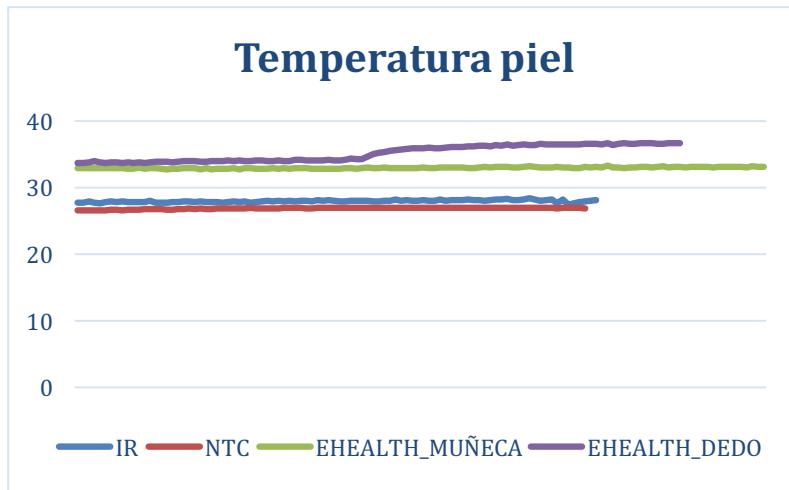
Pruebas temperatura corporal



Tabla 26. Pruebas sensor de temperatura corporal

Se realizaron distintas pruebas, no sólo con estos sensores, sino también con otros para ver que efecto tenían en la muñeca, como un ntc o infrarrojo, pero eran sensores de temperatura para exteriores u objetos y no daban una medida adecuada para la piel. La muñeca es una zona bastante problemática para medir temperatura por lo que en un primer momento como mejor opción se utilizó el sensor de temperatura de ehealth.

En la gráfica se pueden observar los distintos valores de temperatura según el sensor:



Siguiendo con la investigación finalmente se optó por el sensor LMT70 ya que es más pequeño y con mayor precisión.

Para probarlo de forma adecuada se diseñó una placa teniendo en cuenta una serie de especificaciones para que hiciera mejor contacto con la piel vistas en el punto 4.8.

4.3.1.4. GSR

La conductancia de la piel, también conocida como respuesta galvánica de la piel (GSR), es un método para medir la conductancia eléctrica de la piel, que varía con su nivel de humedad. Esto es de interés porque las glándulas sudoríparas son controladas por el sistema nervioso simpático, por lo que en momentos de fuerte emoción, cambia la resistencia eléctrica de la piel.

La conductancia de la piel se utiliza como una indicación de la excitación psicológica o fisiológica, el sensor de respuesta galvánica de la piel (GSR - sudoración) mide la conductancia eléctrica entre 2 puntos, y es esencialmente un tipo de ohmímetro.

El principio o teoría detrás del funcionamiento del sensor de respuesta galvánica es medir la resistencia eléctrica de la piel basada en el sudor producido por el cuerpo. Cuando se produce un nivel elevado de sudoración, la resistencia eléctrica de la piel disminuye. Una piel seca registra mucha mayor resistencia.

Pulsera biométrica Open Source para la monitorización del usuario

Dispositivo	Interfaz	Requerimientos de energía	Características
 GSR GROVE	Digital I ² C	3.3-5V	Se coloca en los dedos del usuario
 GSR E-HEALTH	Analógico	2.7-5V	Se coloca en los dedos del usuario

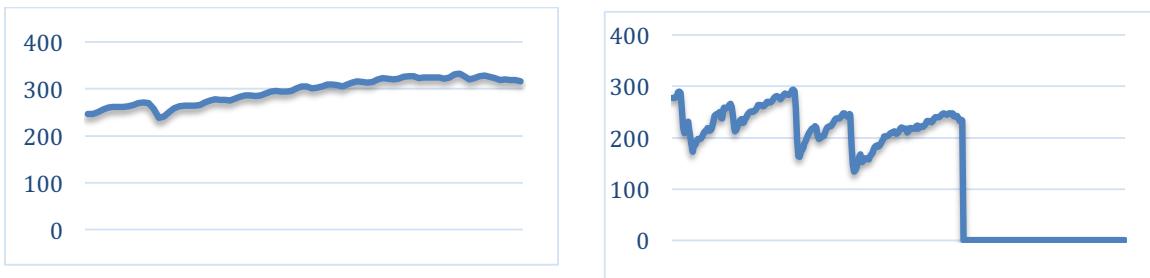
Tabla 27. Características sensores GSR

Pruebas GSR



Tabla 28. Pruebas GSR

Se realizaron pruebas con los distintos sensores de GSR, ambos para los dedos de la mano. En las gráficas se puede observar la información que nos da este sensor. La primera gráfica sería cuando nos encontramos en estado reposo y la segunda vemos como baja la curva de repente, que es equivalente a cuando respiramos profundamente.



Tras ello, se realizó una investigación para ver como otras pulseras que tienen sensor de GSR lo incorporaban y cómo se podía incluir. Algunos ejemplos de otras pulseras:



Fig. 11 Smartband con GSR

Al investigarlo se vio que hacía falta una gran superficie de contacto, normalmente más de dos puntos para que sea posible esta medición sin mucho error. En uno de los casos está en la correa pero en el prototipo planteado la pulsera está separada de la correa. Debido a que en la zona en contacto con la piel ya se introducen tanto el sensor de pulso como el de temperatura, se ve inviable la utilización también de este sensor por lo que finalmente ha quedado descartado.

4.3.1.4. Ambiente

No solo es bueno monitorizar tus parámetros biomédicos sino también tu entorno, ya que éste te puede afectar de diferentes formas. Por ejemplo:

Cuando la presión barométrica disminuye y aumenta la humedad, los tejidos se hinchan, un fenómeno que pasa generalmente desapercibido excepto para quienes sufren de artritis, ya que hasta la más mínima inflamación en las articulaciones les resulta muy dolorosa.

Un tiempo cambiante y fluctuante no sólo es incómodo a la hora de decidir sobre la necesidad de llevar paraguas. Para quienes sufren de migrañas, puede resultar una auténtica pesadilla, ya que se calcula que hasta un 60% de las mismas pueden estar desencadenadas por los cambios de presión. El motivo es que el calor dilata venas y capilares, mientras que el frío las contrae, y esa alternancia puede desencadenar jaquecas en las personas propensas a ellas.

De esta forma afectan los distintos parámetros:

Presión atmosférica. Se trata de una constante que afecta en sus variaciones de forma evidente al cuerpo, sobre todo al aparato cardiovascular y al sistema nervioso central. La tensión arterial se puede ver afectada de forma notable. Las personas hipertensas son sensibles a las variaciones, cuando están a nivel del mar o cuando están en clima de montaña. Las alteraciones más llamativas se producen cuando se superan los 3.000 metros de altitud, a partir de los cuales puede aparecer el mal de altura, que se caracteriza por: trastornos respiratorios, taquicardia, sensación de falta de aire, dolor de cabeza, pesadez, apatía, disminución de la agudeza auditiva y fatiga muscular.

Temperatura. Éste es un factor climático fundamental y sus variaciones extremas (frío o calor intensos) producen importantes y, a veces, graves trastornos de salud. El hombre es un animal de sangre caliente que mantiene una temperatura constante entre 36,5-37ºC. Esto lo hace a través de diferentes mecanismos metabólicos, vasculares y cutáneos, que funcionan para almacenar o perder calor según la temperatura ambiente. El frío intenso provoca una vasoconstricción periférica intensa con aumento del metabolismo basal y producción de calor. El calor produce vasodilatación periférica, sudoración abundante, pérdida de agua y electrólitos a través de la piel. En situaciones muy extremas se puede dar el llamado golpe de calor, que causa hipertermia, deshidratación, dolor de cabeza y afectación del sistema nervioso central.

Humedad ambiental. Es un factor climático íntimamente ligado a la temperatura, y es la característica que más diferencia el clima continental del marino. La humedad intensa dificulta entre otras cosas la sudoración y aumenta la eliminación de líquidos a través del riñón, lo que hace que en ambientes naturalmente húmedos con frío o con calor intensos, éstos se toleren peor.

Dispositivo	Parámetros	Rango y Exactitud	Interfaz	Requerimientos de energía	Características
DS18B20	Temperatura Humedad	-55 a 125°C $\pm 0.5^\circ$	Digital (1-wire)	3-5.5V	
NTC 10K	Temperatura	No muy precisos	Analógico	2-5V	Baratos Cambian rápido de temperatura
BME280	Temperatura Humedad Presión Altitud	Temp: -40 – 85° Hum: 0 – 100 Pres: 30.000 – 110.000Pa Alt: 0 – 30.000 ft(9.2km)	Digital SPI/ I ² C	3.3V <1mA	
SHT15	Temperatura Humedad	Hum: 0-100% RH $\pm 2\%$ RH Temp: $\pm 0.3^\circ\text{C}$ $@ 25^\circ\text{C}$	Digital 2-Wire, parecido I ² C	2.4-5.5V 30uW	Alta precisión pero alto precio

Tabla 29. Características sensores ambiente

Pruebas ambiente

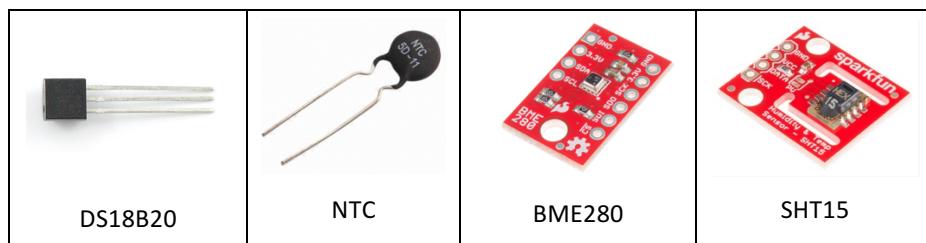


Tabla 30. Pruebas sensores ambiente

Se realizó un programa que almacenaba en una tarjeta SD todos los datos de temperatura tomados cada 5 segundos en formato .CSV para abrir como tabla Excel. Y poder analizar de esta forma como se comportan todos los sensores ante los cambios de temperatura.

Se han realizado pruebas en distintas condiciones en invierno, encima del radiador, en la calle y en la habitación con la calefacción encendida:

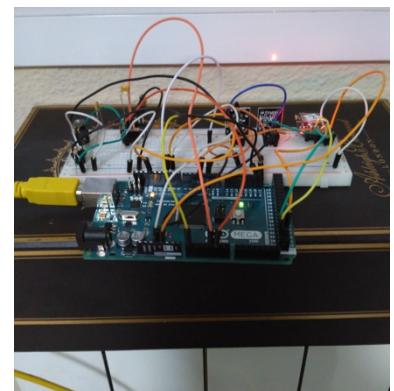
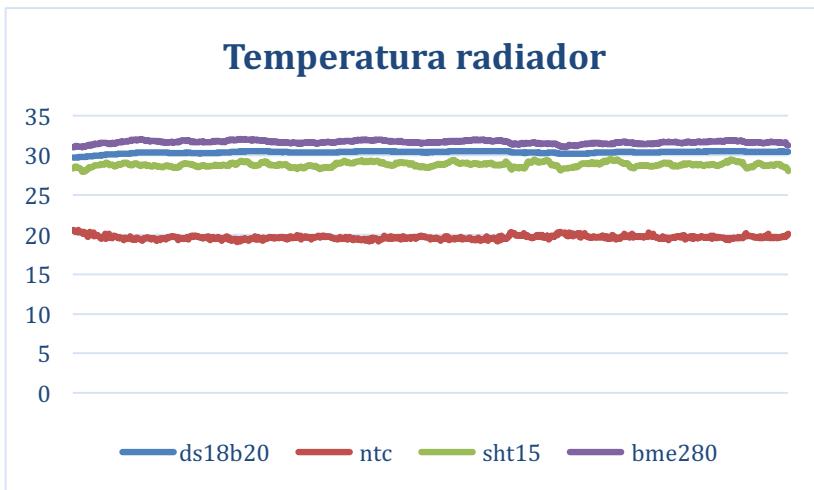


Fig. 12 Pruebas ambiente encima del radiador

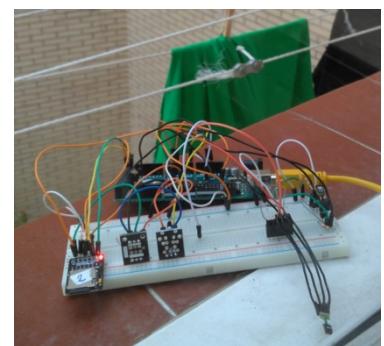
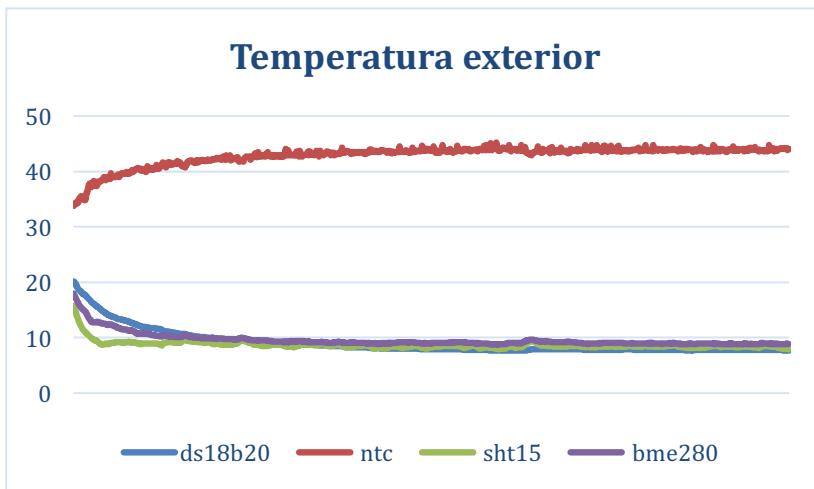


Fig. 13 Pruebas ambiente en exterior

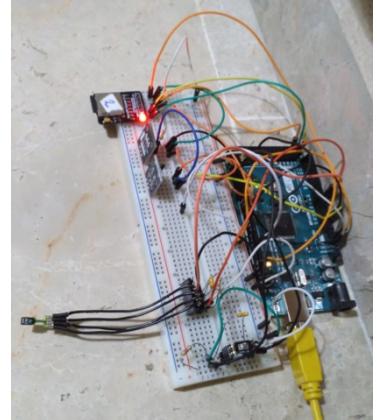
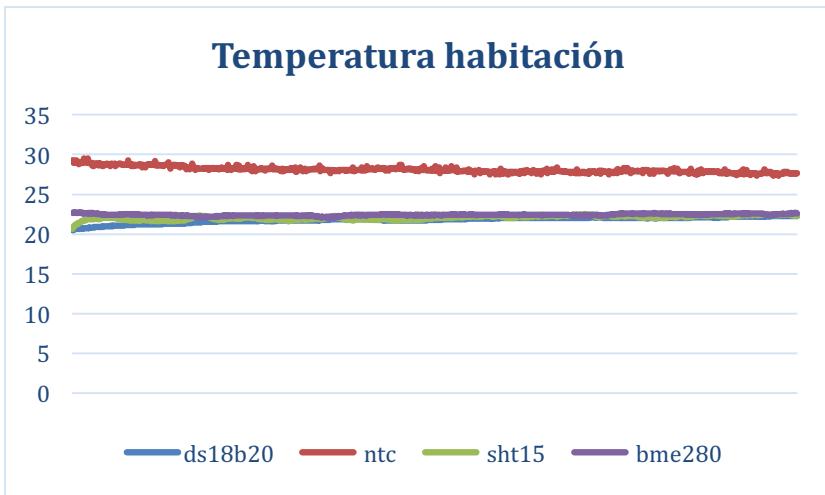


Fig. 14 Pruebas ambiente en interior

Dado que los sensores tenían una respuesta similar, se descartó el sht15 o Sensirion por su alto coste y se optó por utilizar el BME 280 ya que gracias a este sensor también se podían obtener datos de humedad, altitud y presión.

4.3.1.5. Sensor de luminosidad

Añadiendo un sensor de luz es posible añadir más funcionalidades en la pulsera:

- regular el brillo de la pantalla para que no gaste tanto cuando haya luminosidad.
- Junto al sensor de temperatura detectar cuando has pasado mucho tiempo al sol.
- Encender un led a modo linterna cuando enciendes la pulsera y estás a oscuras.

Pulsera biométrica Open Source para la monitorización del usuario

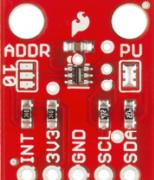
Dispositivo	Interfaz	Requerimientos de energía	Características
 TEMT6000	Analógico	3.3-5V	No tiene efecto infrarrojo, ultravioleta o cualquier luz que no se ve directamente.
 LDR	Analógico	3-150V	Se satura con mucha luz Tarda en regularse
 TSL2561	Digital I ² C	2.7-3.6V <0.6mA	Detecta la luz y el infrarrojo. Proporciona los luxes. 0.1-40000lux

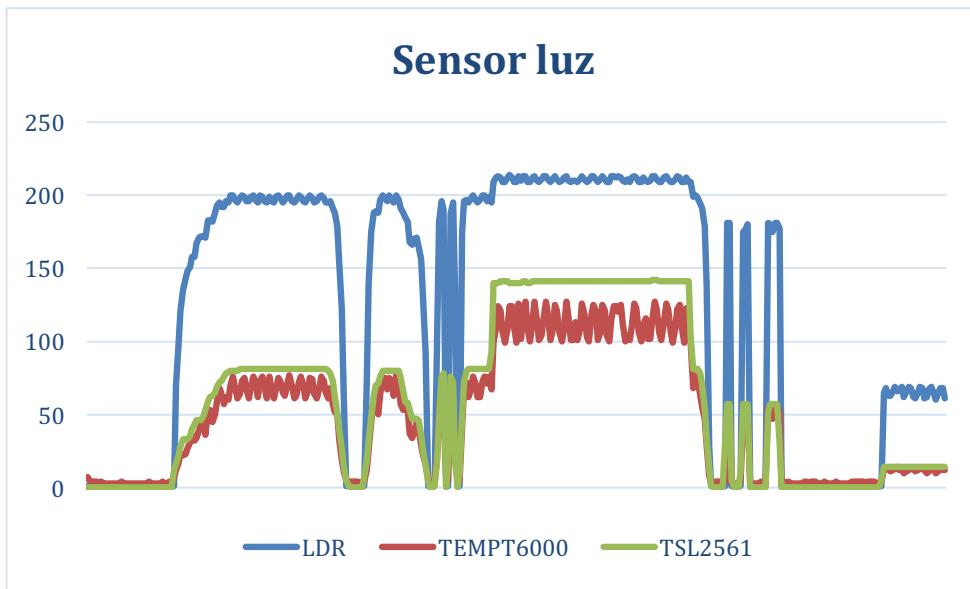
Tabla 31. Características sensores de luz

Pruebas sensor de luz



Tabla 32. Pruebas sensores de luz

Al igual que con los sensores de ambiente se realizó otro programa para guardar los datos en una SD con Arduino. Para determinar los distintos grados de luminosidad en esta prueba se utilizó una lámpara con regulador de intensidad.



Podemos observar en la gráfica que el LDR para baja luminosidad funciona bien pero en cuanto es más grande se satura por lo que fue el primero en ser descartado.

Finalmente se optó por el tsl2561 ya que es más preciso, ve el espectro invisible al ojo humano y y funciona por I²C por lo que nos proporciona los luxes directamente.

4.3.2. Pantalla

Se quería que fuera un dispositivo autónomo, por lo que finalmente se ha decidido usar una pantalla donde se puedan mostrar los datos en tiempo real. Para ello será necesario un menú donde poder seleccionar los datos que quieras ver y pulsadores para realizar esta selección ya que una pantalla táctil incrementaba el precio final y tamaño de la pulsera.

Pulsera biométrica Open Source para la monitorización del usuario

Dispositivo	Interfaz	Requerimientos de energía	Características
OLED 64X48	Digital I ² C	3.3 V <20mA	Se ve en la oscuridad
OLED 128X64	Digital I ² C /SPI	3.3V <40mA	Se ve en la oscuridad
SHARP MEMORY 96X96	Digital SPI	3.3V 4uA	Mezcla de LCD y e-paper. No refleja, no se ve en la oscuridad.

Tabla 33. Características pantallas

Pruebas pantalla



Tabla 34. Pruebas pantallas

Se realizaron pruebas en distintos modelos de pantallas a un solo color y finalmente por su tamaño y su bajo consumo, además de que se conecta se optó por usar la pantalla Sharp Memory.

Otro aspecto que la diferencia de las otras es que dispone de conector y si se estropea puede cambiarse fácilmente. En los otros casos la pantalla va soldada directamente a las vías.

4.3.3. Bluetooth

Pese a que esta tecnología es relativamente reciente, existen multitud de alternativas en el mercado, con diferentes configuraciones de chipset, memoria y pinouts.

Estudio de las comunicaciones inalámbricas

Dentro de los dispositivos comerciales analizados, es posible distinguir tres tipos de categorías en módulos Bluetooth Smart, los que requieren de un chip externo, los System on Chip (SoC) y los módulos Bluetooth.

Requieren de un chip externo:

Los módulos que requieren de un chip externo tienen el inconveniente de que el trabajo del desarrollador es mayor debido a la necesidad de convertir y tratar con los drivers de bajo nivel pero a cambio, el desarrollador puede usar el microcontrolador que mas se ajusta a sus necesidades, en lugar del que viene en el módulo.

NRF8001

El nRF8001 es un dispositivo independiente que requiere de un microcontrolador externo, pero hace funcionar a la pila del BLE en forma interna de modo que la mayoría de las tareas de bajo nivel ya están manejadas. Es controlado enviando comandos y datos y recibiendo eventos desde el dispositivo. El SPI es una interfaz muy común de modo que es muy fácil conectarla a la mayoría de los microcontroladores. La ventaja de usar tal dispositivo es que puedes usar el microcontrolador que más te encaje, siempre que cumpla con todos los requisitos de su aplicación. Sin embargo, el uso de un microcontrolador externo puede significar una solución más cara. Algunas de las especificaciones:

- Bluetooth compatible: Bluetooth v4.0 Modo Único
- Solución BLE con pila BLE integrada.
- Requiere microcontrolador externo.
- 6x6mm QFN
- \$1.96/1ku

CC256x

EL CC256x es el único capaz de ser compatible con lo que típicamente se denomina Bluetooth Clásico en una configuración de Modo Dual. Esto le permite transmitir audio y ser compatible con la funcionalidad típica cuando se piensa en Bluetooth. El CC256x está disponible inmediatamente a cualquiera ya que TI ha proporcionado kits de Evaluación y Desarrollo como también software. Las soluciones Modo Dual de otros vendedores no son accesibles para pequeños fabricantes.

- Bluetooth compatible: Bluetooth de baja energía v4.0 y Bluetooth Clásico – Modo Dual
- Solución Modo Dual hasta el HCI Level
- Requiere de un microcontrolador externo operando un stack de Bluetooth
- 8.10×7.83mm
- \$3.00/1ku

Una cosa importante sobre el CC256x es que requiere de una pila Bluetooth que opera con un microcontrolador externo. Esto quiere decir que usted necesitará asignar para él decenas de espacios de kB de Flash, lo que podría ser significativo. También significa que usted sea cuidadoso con su selección del microcontrolador porque los lentes no pueden satisfacer los requerimientos para transmitir audio usando A2DP. Por ejemplo, emparejado con el MSP430, el CC256x puede fácilmente procesar el perfil SPP para enviar datos pero no podrá procesar la transmisión de audio. Sin embargo, si se lo empareja con un Cortex-M3 o Cortex-M4, la transmisión de audio a parlantes puede ser fácilmente lograda. A3DP puede cambiar esta realidad.

System on chip:

Los SoC son microchips desarrollados por compañías especializadas en circuitos integrados, que crean un microchip con la implementación de la tecnología Bluetooth pero requieren de más componentes para que funcione. Por ejemplo, en algunos dispositivos es necesario añadir la antena, condensadores y resistencias en los pines, un escudo de RF, un reloj de cristal de cuarzo de 32kHz y otro de 32MHz, además de tener que pasar las certificaciones FCC, IC, ETSI. Los fabricantes de SoC para Bluetooth Smart más extendidos son Texas Instruments con los modelos CC2540/2541 y Nordic con el NRF51822.

CC2540/2541

El CC2540 y su versión mejorada, la CC2541, han sido muy populares durante varios años. Ambos dispositivos presentan un procesador 8051 que permite a su aplicación funcionar sin un microcontrolador externo, de modo que el costo total del sistema es más bajo. Esto tiene también un aspecto negativo que es común a la mayoría de las soluciones integradas. Su aplicación estará ligada al procesador 8051, para mejor o para peor. En muchas aplicaciones esto no será un problema, pero en algunos casos su producto requerirá de mayor rendimiento.

He aquí algunas especificaciones más detalladas:

- Bluetooth compatible: Bluetooth v4.0 Modo Único
- Sistema completo de solución de chip incluyendo Bluetooth Low Energy integrado, microcontrolador y periféricos.
- 8051 integrado sobre el Chip con 256kB o 128kB flash y 8kB RAM
- Paquete QFN 6x6mm
- Máxima potencia de salida 0dBm
- Sensibilidad de recepción -94dBm
- Analógico: 12-bit ADC, comparador, sensor de temperatura y monitor de batería.
- \$3.29/1ku

Un aspecto importante que tiene que tenerse en cuenta con el CC2540 y el CC2541 son las herramientas de desarrollo. La mayoría de los dispositivos Bluetooth de chip único son en realidad radios definidos por el software. En realidad es este firmware el que maneja parte de la funcionalidad de bajo nivel del Bluetooth de baja energía. Esta flexibilidad puede ser una bendición y una maldición. Por un lado, el software definido quiere decir que TI puede proveer firmware nuevo y mejorado. Por otra parte, los archivos BLE necesitan estar integrados a su software por el linker, lo que significa que tiene que estar conectado con su compilador específico. Texas Instruments provee del software para Bluetooth Low Energy (BLE) para IAR's 8051 IDE. Después de un período de evaluación de 30 días usted básicamente tiene que pagar \$3.000 – \$4.000. El IAR es un muy buen compilador pero para usarlo como hobby es un precio muy elevado.

El CC2541 no es solamente un radio BLE sino que puede también operar protocolos registrados, lo que el CC2540 no es capaz de hacer. Entre los otros cambios se encuentran:

- RX en el modo de potencia mas bajo reducido de 19.6mA a 17.9mA
- El CC2541 no tiene interfaz USB
- CC2541 tiene una interfaz de hardware I²C
- El CC2541 puede transmitir a 0dBm versus 4dBm del CC2540

NRF51822

Nordic Semiconductor lanzó el nRF51822 después del nRF8001 y es uno de los dispositivos de Bluetooth Low Energy (BLE) más populares. Es una solución integrada que atrae a los desarrolladores debido a su pequeño tamaño. Incorpora un CPU Cortex-M0, lo que significa que hay muchas opciones de compilador para el desarrollo. De hecho, Nordic explica como configurar un medio de desarrollo usando un compilador ARM GCC, Eclipse y otras pocas herramientas. Esto le puede permitir a cualquier desarrollador usar el RF51822 con herramientas gratis. Para el desarrollo del producto se recomienda usualmente usar una cadena de herramientas de un vendedor tales como IAR o Keil. Algunas de las características del dispositivo son:

- Bluetooth compatible: Bluetooth v4.1 Modo Único
- Solución BLE completa con pila BLE integrada.
- ARM Cortex-M0 integrado en el Chip con 256kB o 128kB flash y 16kB o 32kB RAM
- Disponible en paquetes 6x6mm QFN y 3.5x3.8mm WLCSP.
- Voltaje 1.8V – 3.6V
- \$2.31/1ku

El chipset BLE de Nordic es en gran parte un dispositivo multiprotocolo. Soporta funcionalidad de radio patentada estándar de 2.4GHz con tres diferentes tasas de transferencia de datos de hasta 2Mbps. Esto puede ayudarlo a proteger el producto de futuros cambios si alguna vez se necesita una comunicación patentada. Es completamente compatible con las radios nRF24L de Nordic que han sido usadas en bastante productos.

Nordic anunció recientemente el nRF51822 que funciona con IPv6, lo que abre la puerta a controlar por Internet productos con el nRF51822, por oposición al uso de Wi-Fi (y a un costo mucho menor). Esto puede llegar a ser extremadamente atractivo para los desarrolladores de productos.

Módulos BLE:

Por otro lado, fabricantes como Bluegiga, Panasonic o Connect Blue, integran los SoC anteriores en módulos Bluetooth equipados con antena, relojes y todos los componentes correctamente certificados y testeados para asegurar su funcionamiento. Estos módulos Bluetooth están listos para su incorporación en un sistema y vienen provistos, por parte del fabricante, de herramientas para su configuración.

Según su configuración, se pueden distinguir dos tipos, módulos como los de Bluegiga que tienen un firmware para facilitar la programación sin perder sus características y módulos como el hm10 o el de Readbear que abstraen y empaquetan la conexión BLE en una conexión serie.

Bluegiga 112

Módulos BLE112 y BLE113 de Bluegiga se basan en el CC2540 / CC2541 de Texas Instruments. Incluyen soporte para BGScript, lo que le permite programar ciertos tipos de aplicaciones utilizando archivos XML simples. Bluegiga también proporciona una API en C para trabajar con estos módulos utilizando una MCU externa.

La principal diferencia entre el BLE112 y BLE113, es que el BLE113 es un módulo más reciente, tiene el consumo de energía ligeramente inferior y añade un puerto I²C, que se puede utilizar para hablar con una amplia variedad de sensores de bajo coste (sensores de temperatura, acelerómetros, giroscopios, sensores de presión, etc.)

Estos módulos tienen las certificaciones reglamentarias para CE / ETSI (Europa), FCC, Canadá, Japón y Corea del Sur y están disponibles para comprar en la mayoría de vendedores principales (DigiKey, Mouser, Farnell, etc.).

Sus dimensiones son de 18.1mm x 12mm x 2.3mm.

El precio es de unos 10€.

Bluetooth 4.0 BLE Module de Smart Prototyping:

El módulo Bluetooth 4.0 BLE Module de Smart Prototyping está basado en el chipset Bluetooth 4.0 Clase 2 CC2540 de Texas Instruments.

Éste transmite a 4dBm y tiene una sensibilidad en recepción de -84dBm, con antena integrada en el propio módulo. La interfaz de comunicación con el micro se realiza a través de la UART con comandos AT.

Sus dimensiones son de 26.9mm x 13mm x 2.2mm.

El precio es de unos 10€.

BLE Mini de ReadBearLab

El módulo BLE Mini de ReadBearLab es una versión mejorada del BLE Shield de la misma compañía.

Éste incorpora el chipset CC2540 de Texas Instruments, y las dimensiones son considerablemente más reducidas que su versión anterior. El módulo incorpora ciertos elementos programables, (ya que éste en si, puede trabajar sin microprocesador adicional) EEPROM de 512kB, LEDs y pulsador.

Permite conexiones de hasta 8 clientes y en rangos de hasta 30 y 150 metros, en función del entorno, con +3dBm de potencia en transmisión y -93dBm de sensibilidad en recepción.

Sus dimensiones son de 39mm x 18.5mm x 3.8mm.

Su precio es de unos 25€.

HM10

El HM-10 es un módulo Bluetooth Low Energy (BLE, Bluetooth 4.0 o Bluetooth Smart) basado en los chips de Texas Instruments CC2540 o CC2541.

Un aspecto importante de este módulo es que una vez configurado, puede funcionar de forma autónoma sin necesidad de estar conectado a un microcontrolador que lo gestione. Simplemente alimentando el módulo (pines Vcc y GND)

Sus dimensiones son 26.9mm x 13mm x 2.2 mm. Su precio es de unos 10€.

Introducir un módulo Bluetooth en nuestro dispositivo aporta al proyecto una serie de ventajas y posibilidades:

- Permite que pueda conectarse a nuestro móvil y a través de una app tener un resumen diario de nuestra actividad.
- Permite que estos datos puedan ser almacenados en un ordenador si así lo prefiriéramos.
- Estos datos enviados en tiempo real a un ordenador podrían ser utilizados en un proyecto artístico donde se proyectarían tus constantes vitales por ejemplo.
- Permite que pueda tener más funcionalidades a futuro como notificaciones desde el móvil.

Pulsera biométrica Open Source para la monitorización del usuario

Dispositivo	Módulo	Interfaz	Requerimientos de energía	Características
NRF8001		Digital UART/SPI	1.9-3.6V <12mA	Requiere microcontrolador externo
READBEAR NANO	MDBT40	Digital UART/SPI	1.8-3.3V <15mA	Basado en NRF51822. Incluye CórTEX M0, antena y certificado 18x10x3.2
	BLE112		2-3.6V <27mA	Basado en cc2540. Incluye antena y certificado 18.1x12x2.3
RFDUINO	RFD22301	Digital UART/SPI	1.9-3.6V <18mA	Incluye CórTEX M0, antena y certificado. 15x15x3.5

Tabla 35 Características Bluetooth

Pruebas bluetooth

Se han probado distintos módulos Bluetooth a través de las librerías de los fabricantes para ver cual encajaba mejor con la pulsera. Debido a espacio y facilidad de uso, se buscaba un módulo con librería, antena y certificado y que fuera pequeño. Además como se quería usar la pantalla Sharp Memory, se buscaba un módulo que funcionará por UART. En un primer momento se eligió el módulo de RFduino enviando tramas por UART pero continuando con las pruebas y mejoras, se estudió el módulo Bluefruit de la empresa Adafruit, que con su firmware y conectada por UART, te da la posibilidad de elegir si quieres enviar datos a la aplicación por comandos AT o por UART.

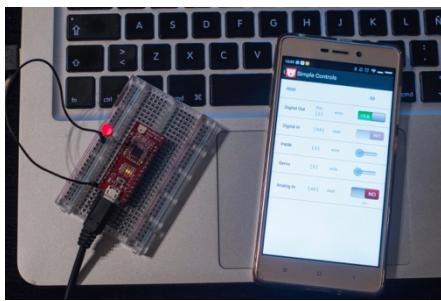


Fig. 15 Pruebas Readbear Blend Micro

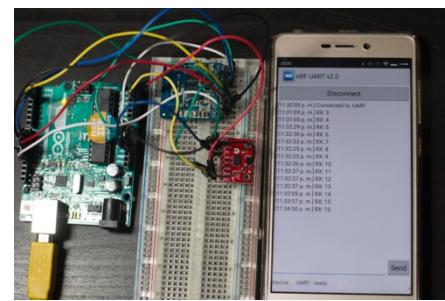


Fig. 16 Pruebas nRF8001

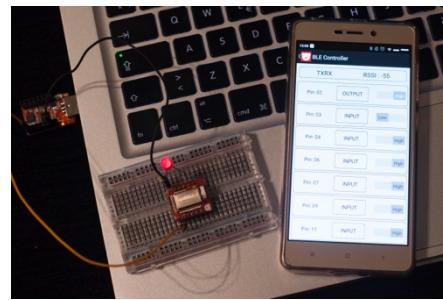


Fig. 17 Pruebas Readbear Nano

4.3.4. Microcontrolador

Un microcontrolador es un circuito integrado programable, capaz de ejecutar las órdenes guardadas en su memoria. Está compuesto de todos los bloques funcionales de un sistema microprocesador en un único encapsulado, los cuales cumplen una tarea específica. Un microcontrolador incluye en su interior las tres principales unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada/salida; los tres se encuentran conectados mediante buses.

La siguiente tabla muestra las principales características de cada uno para la toma de decisión de un microcontrolador u otro.

Dispositivo	Modelo	Requerimientos de energía	CPU	Analógicos IN/OUT	Digitales IO/PWM	UART	EEPROM/SRAM/Flash
MICRODUINO	ATmega 644p	3.3V	8MHz	8/0	32/8	1	2/4/64
ARDUINO MEGA	ATmega 2560	5V/7-12V	16MHz	16/0	54/15	4	4/8/256
ARDUINO UNO	ATmega 328p	5V/7-12V	16MHz	6/0	14/6	1	1/2/32
BLUEFRUIT	Córtex M0	3.3V/7-12V	48MHz	6/1	14/10	2	-/32/256
	ATmega1284p	3.3V	8MHz	8/0	32/8	2	4/16/128

Tabla 36. Características microcontroladores

Pruebas microcontrolador

En el caso del microcontrolador existía la duda si utilizar un Córtex M0, como muchos de los wearables o un ATmega, un micro más común y con mucha más información en internet.

Para probar se utilizó la placa de Adafruit Feather M0 y se cambió su firmware a través del programador del BLE Nano de Readbear, ya que este Bluetooth funciona con un M0. Para cambiar su firmware es necesario cablear los pines: SWDIO, SWCLK, VDD y GND. Visto que existía esta posibilidad, además de la versión 1 de Bioband, se realizó otra placa con un Córtex M0 para realizar pruebas con ella y ver si era una mejor opción.

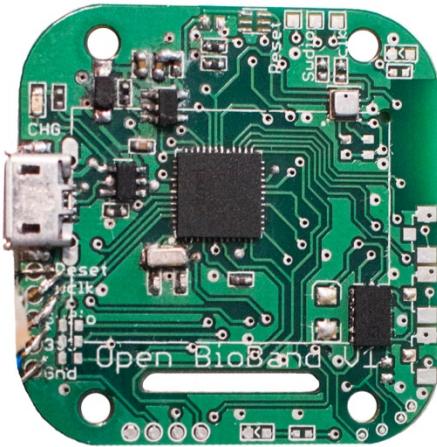


Fig. 19 PCB con Córtext M0

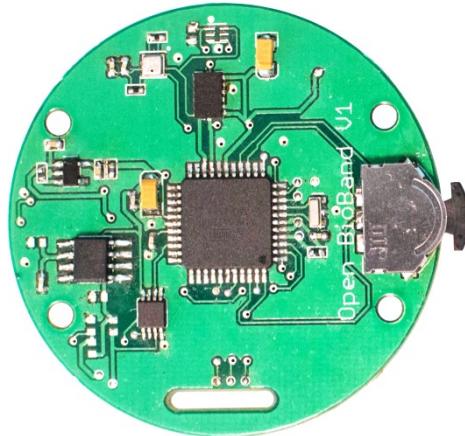


Fig. 18 PCB con ATmega1284p

Sin embargo, al ir a quemar el bootloader no funcionaba. Realizando un nuevo estudio se vio que con el programador del ble nano era posible cambiar el firmware si ya se había quemado anteriormente, pero no la primera vez que se realizaba.

Las opciones que se encontraron en este segundo estudio fueron:

3. Comprar un Córtext M0 quemado ya el bootloader para usar con el IDE de Arduino.
Precio: 7.5€ más gastos
3. Comprar el programador oficial, Atmel ICE. Precio: 160€ más gastos
3. Recablear un Arduino Zero. Precio: 47€ más gastos.

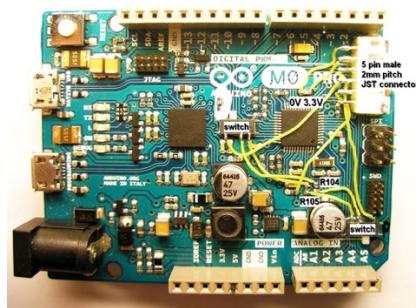


Fig. 20 Cableado Arduino Zero para utilizarlo como programador

Comparando con los micros de ATmega, quemar el bootloader es mucho más económico ya que en el caso de que no tengas un programador externo, esto puede hacerse con un Arduino Uno como ICSP, una placa que cualquier desarrollador, electrónico o maker tiene en su casa. Por lo que finalmente se decide usar un ATmega1284p y el Córtext M0 se guarda para futuros estudios.

4.3.5. Carga

Para la carga del dispositivo había dos opciones: carga por inducción o carga por USB. Para ello, se ha recurrido a controladores de carga integrados, los cuales se encargan de la gestión y protección en el proceso de carga.

En el caso del circuito de carga por USB, se necesitaba la inclusión de un conector de micro USB, lo que de cara a fabricación supone algunos problemas, como posicionamiento o adaptación a la carcasa. Además, si a futuro se planteara la estanqueidad de la pulsera, supondría la búsqueda de algún modelo *Waterproof*.

Por otro lado, la opción inalámbrica utiliza un integrado que gestiona la carga a través del estándar de carga por inducción Qi, ya muy extendido en gran parte de pulseras y terminales móviles actuales. Este fenómeno permite la integración de una etapa de carga por inducción, barata, pequeña y eficiente.

En los diseños anteriores, con Córtext M0 y ATmega se barajaron y probaron las dos posibilidades.

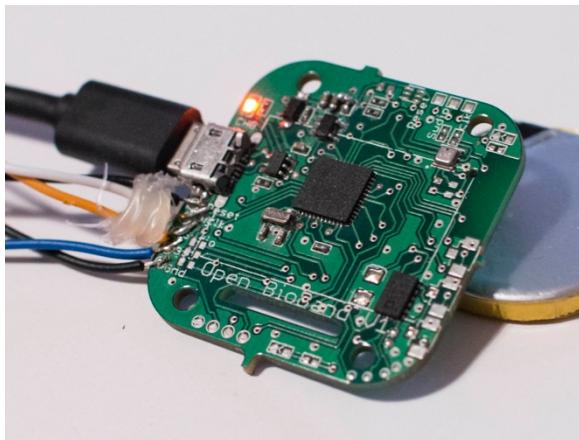


Fig. 22 Carga por USB Córtext M0



Fig. 21 Carga inalámbrica ATmega1284p

Carga por USB

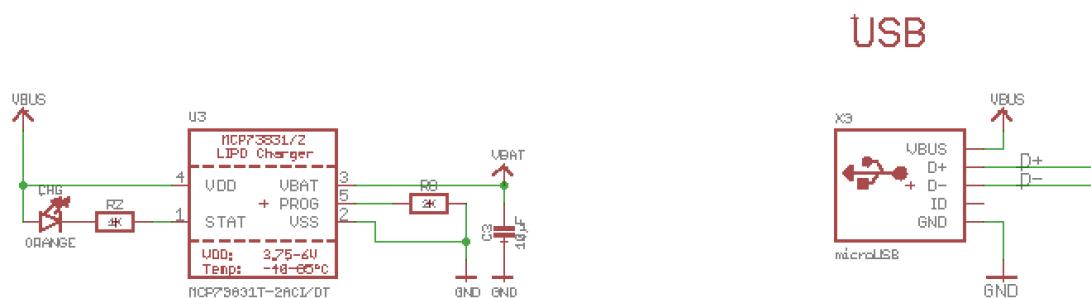


Fig. 23 Circuito de carga por USB

La ventaja del USB frente a la inducción es que además de cargar la batería, te sirve para programar el dispositivo. Sin embargo, no son muchos los wearables que disponen de la opción de carga inalámbrica, por lo que era algo interesante a estudiar.

Tras las pruebas, se observó que el resultado ante ambos circuitos era similar, tanto en calentamiento de la etapa, como en capacidad de suministro de corriente de carga. A su vez, el que tuvieran un coste similar, hizo que se eligiera la opción inalámbrica para el diseño final.

Se decidió utilizar un modelo de integrado de carga, como se ha comentado, ya extendido. Gracias al estudio de mercado y dispositivos actuales, se eligió el componente utilizado en el Smartwatch Moto360. También se utilizó para pruebas de carga el cargador de este mismo modelo de producto.



Fig. 25 Bobina y adhesivo magnético moto360



Fig. 24 Cargador moto360

4.4. Decisión final componentes

Tras realizar las distintas pruebas de cada uno de los componentes se optó por la utilización de los que se muestran en la tabla y se descartó el sensor GSR debido a espacio ya que necesita un buen contacto con la piel.

ACELERÓMETRO	TEMPERATURA PIEL	AMBIENTE
 ADXL345	 LMT70	 BME280
LUZ	PANTALLA	PULSO
 TSL2561	 SHARP MEMORY	 MAX30101
BLUETOOTH	MICRO	CARGA
 MDBT40-NRF51822	 ATMEGA 1284P	 BQ51051B

Tabla 37. Decisión final componentes

Con los componentes elegidos, puede verse ahora un diagrama de bloques más detallado del sistema.

En el diagrama final de producto se pueden distinguir claramente 5 zonas: Alimentación, Sensores, Comunicación, Visualización (pantalla) y Microcontrolador.

- Alimentación: incluyendo la etapa de carga por inducción y las indicaciones tanto de batería como de carga. Permite alimentar el resto de etapas.
- Microcontrolador: Los datos son tomados por el microcontrolador ATmega1284p, el cerebro del dispositivo. Al ser alimentado, se encarga de comunicarse con el resto de etapas para llevar a cabo las distintas funciones del producto final.
- Sensores: etapa que monitoriza distintos parámetros. El microcontrolador se encarga de leer, consultar y enviar a los distintos integrados y circuitos a través de los protocolos de comunicaciones disponibles en el procesador (SPI, I2C) o del conversor ADC integrado.
- Visualización: a través de la pantalla Sharp Memory que se conecta con el microcontrolador por interfaz SPI. La pantalla es utilizada para mostrar al usuario el valor de los sensores, a la vez que sirve de interfaz para elegir la función a ser utilizada. También nos permite de medio de intercambio de datos entre el usuario y la pulsera, por medio de la pantalla de ajustes.
- Comunicaciones: el microcontrolador dispone de comunicación UART con el que se comunica con el módulo BLE. Esto permite tener en todo momento la funcionalidad de mandar datos a un teléfono móvil o recibir parámetros de configuración. Además, estos datos pueden ser subidos a la nube por un tercero. Esta funcionalidad de control de comunicaciones está integrada, pero se deja la implementación final del envío de datos o su recepción a los futuros desarrolladores de aplicaciones finales.

Pulsera biométrica Open Source para la monitorización del usuario

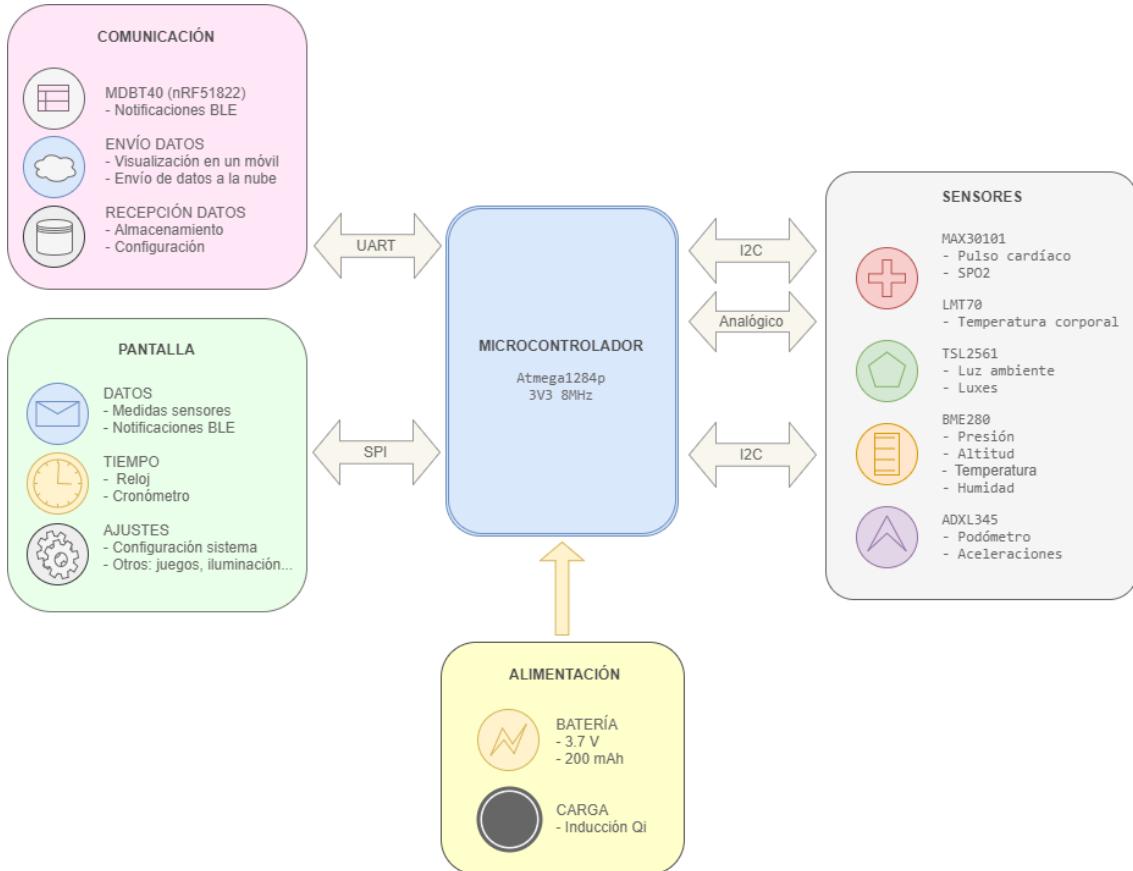


Fig. 26 Diagrama de bloques del sistema detallado

4.5. Esquemático

Una parte relevante de este proyecto es el desarrollo hardware. En este apartado hablaremos de la fabricación de la PCB y de los circuitos utilizados.

4.5.1. Pinout ATmega1284p

En las distintas tablas se puede ver la elección de los pines. El pulsador de selección va a un pin de interrupción para activar la pantalla cuando está en modo reposo. La mayoría de los sensores funcionan por I²C por lo que el microcontrolador no ha limitado el uso de ninguno. El módulo Bluetooth funciona por UART, pero dicho micro tiene dos UARTs, así que podemos usar el serial para ver los datos por el monitor serie de Arduino a la hora de programar el microcontrolador.

Pines digitales

1	Luz led
2	Tx – BLE
3	Rx – BLE
4	DISP – Pantalla
5	T_ON – Temperatura
6	Pulsador selección
7	Indicador carga batería
8	Pulsador UP
9	Pulsador DOWN
10	SCLK – Pantalla
11	MOSI – Pantalla
12	MODO AT/UART - BLE
13	SCS – Pantalla

Pines analógicos

A0	Modo UART/AT - BLE
A1	LMT70 – Temperatura corporal
A2	Nivel de batería
A3	
A4 – SDA Y A5 – SCL INTERFAZ I2C	MAX30101 – Pulso ADXL345 - Acelerómetro BME280 - Ambiente TSL2561 - Luz
A6	-
A7	-

Pulsera biométrica Open Source para la monitorización del usuario

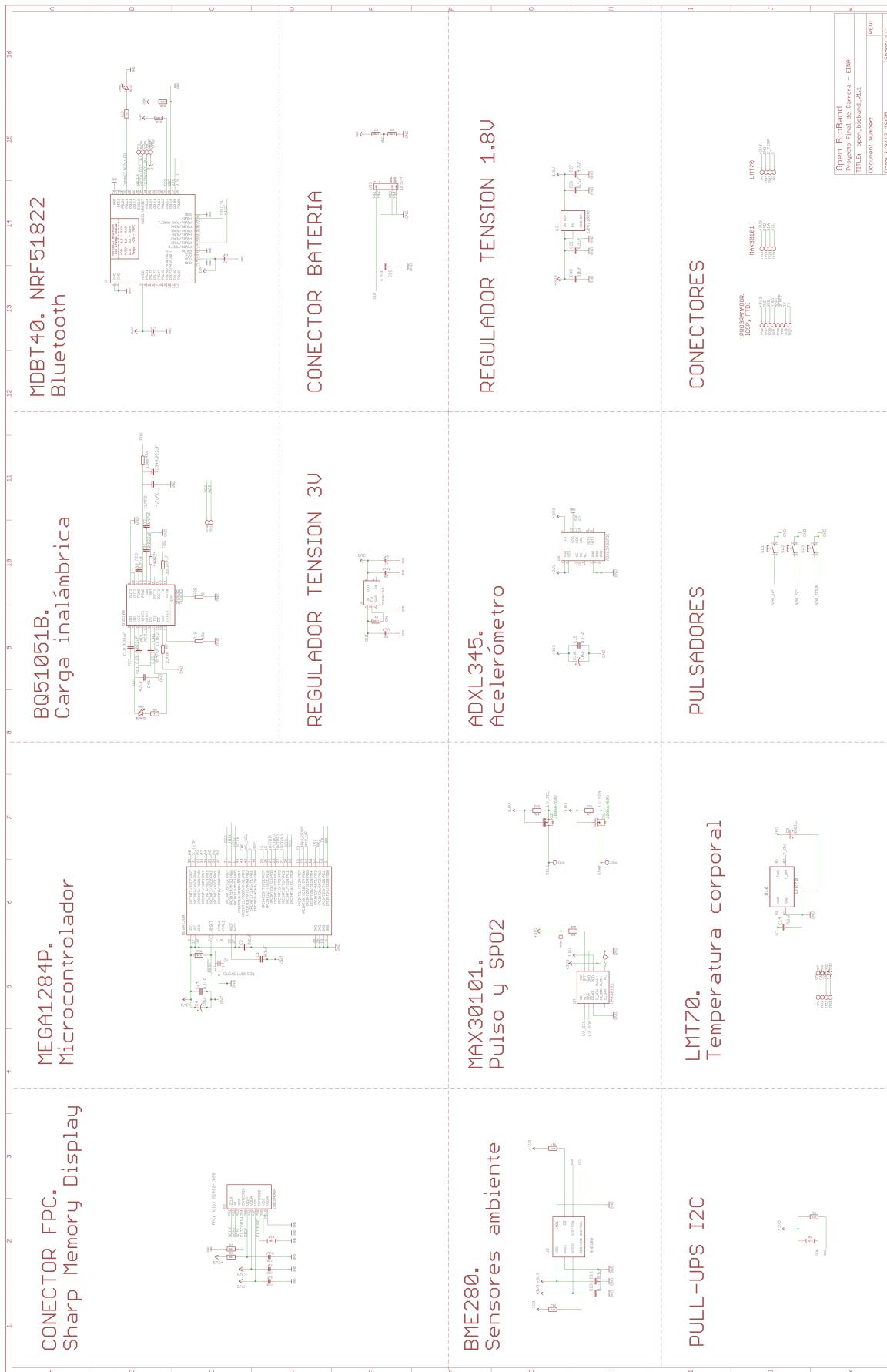


Fig. 27 Esquemático Open Bioband

4.5.2. Circuitos utilizados

Acelerómetro

El ADXL345 es un acelerómetro MEMS pequeño, delgado, de baja potencia y 3 ejes con medición de alta resolución (13 bits) hasta +/- 16 g. Los datos de salida digital se formatean como complemento de 16 bits y se pueden acceder a través de una interfaz digital SPI (3 o 4 hilos) o I²C.

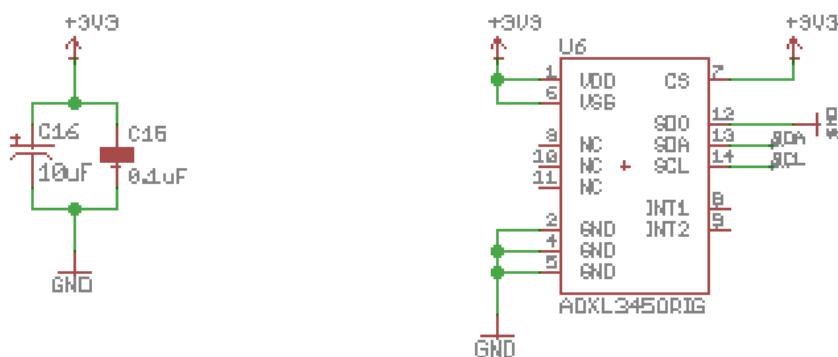


Fig. 28 Esquemático acelerómetro

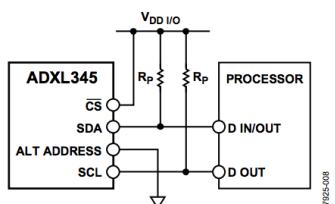


Fig. 29 Conexión datasheet acelerómetro

En este caso lo utilizamos por I²C por lo que el pin CS (chip select) se conecta a 3.3V y SDO a GND, lo que indica que utilizamos la dirección 0x53.

El circuito además necesita, como indica el datasheet, dos condensadores en paralelo para mejorar el ruido de alimentación, un condensador cerámico de 0.1 uF y un condensador electrolítico de 10 uF, cogiendo de esta forma mayor rango de frecuencia.

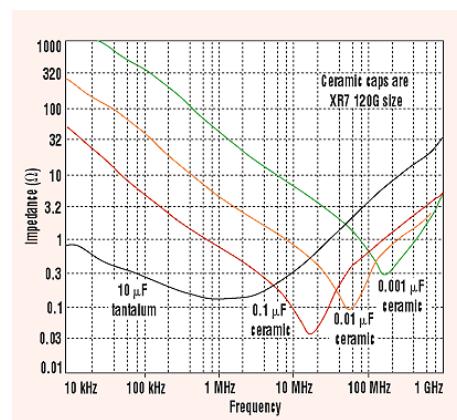


Fig. 30 Gráfica condensadores de bypass

Pulso

El MAX30101 es un sensor de pulso y SPO2. Incluye leds internos, fotodetectores, elementos ópticos y electrónica de amplificación de bajo ruido, y rechazo de la luz ambiental. La idea es encender los diferentes LEDs, y luego detectar el brillo recibido y la dispersión del haz luminoso.

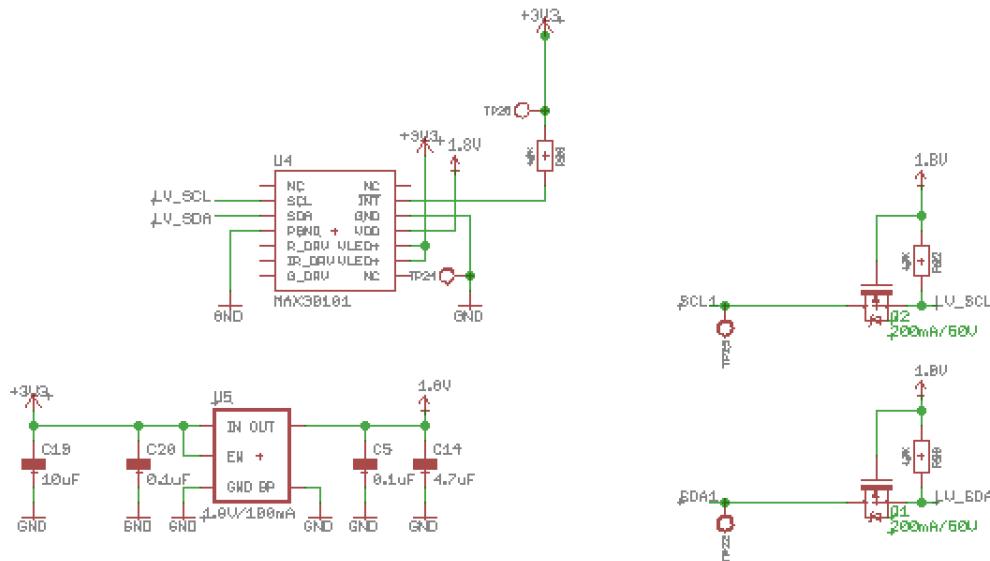


Fig. 31 Esquemático sensor de pulso

La tensión de operación del MAX30101 es entre 1.7V y 2V, por lo que se ha incluido un regulador de tensión, referencia de tensión a 1.8V y dos mosfet de tipo N para que la tensión en el I²C sea de 1.8V y se adapte a la de las señales de SCL y SDA del microcontrolador a 3V3. La tensión del driver de control de los leds es un valor entre 3.1V y 5.25V por lo que utilizamos la alimentación general. Añadimos condensadores en paralelo para eliminar el ruido de alimentación, un condensador cerámico de 0.1 uF y un condensador electrolítico de 10 uF, cogiendo de esta forma mayor rango de frecuencia, en el regulador/referencia de tensión.

Conectamos el pin INT con una resistencia de pullup a 3.3V como indica el datasheet en la configuración de trabajo del sensor.

Temperatura

El LMT70 es un sensor de temperatura analógico CMOS ultra-pequeño, de alta precisión y baja potencia. Las aplicaciones para el LMT70 incluyen prácticamente cualquier tipo de detección de temperatura donde se requieren rentabilidad, alta precisión y baja potencia, como los nodos de sensores de Internet de Cosas (IoT), termómetros médicos, instrumentación de alta precisión y dispositivos con baterías. El LMT70 es también un gran reemplazo para termistores NTC / PTC de precisión.

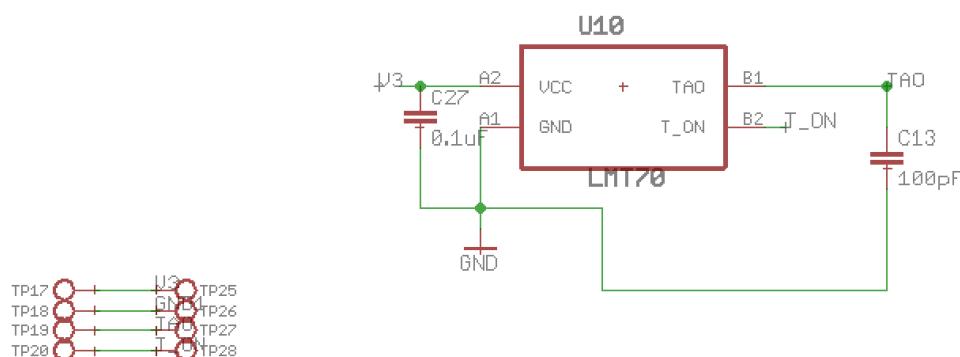


Fig. 32 Esquemático sensor de temperatura corporal

Es recomendado colocar un condensador cerámico de bypass de 0.1uF para eliminar ruido en alimentación y un condensador de carga en T_AO menor de 1.1nF.

Ambiente

El BME280 es un sensor combinado digital de humedad, presión, altitud y temperatura. El sensor proporciona tanto interfaces SPI como I²C.

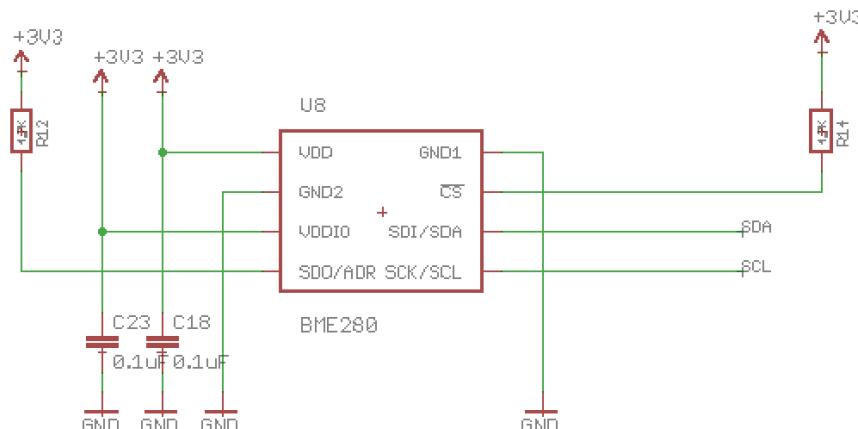


Fig. 33 Esquemático sensor de ambiente

Tal como indica el datasheet se han conectado dos condensadores de 100nF en Vdd y VddIO para mejorar el ruido de alimentación.

Al ser utilizado a través de interfaz I²C, CS deberá ir conectado a 3.3V a través de una resistencia de pull-up.

El pin SDO sirve para configurar la dirección I²C. Como lo hemos conectado a 3.3V de la misma manera con una pull-up, la dirección del BME280 será la 0x76, como nos indica su ficha técnica.

Luz

TSL2561 es un sensor de luz que tiene una respuesta plana en la mayor parte del espectro visible. Mide tanto la luz infrarroja como la visible para aproximar mejor la respuesta del ojo humano. Y debido a que puede absorber luz durante un tiempo predeterminado, es capaz de medir pequeñas y grandes cantidades de luz cambiando el tiempo de integración. Es capaz de comunicación directa I²C y de realizar rangos de luz específicos de 0,1 - 40k Lux.

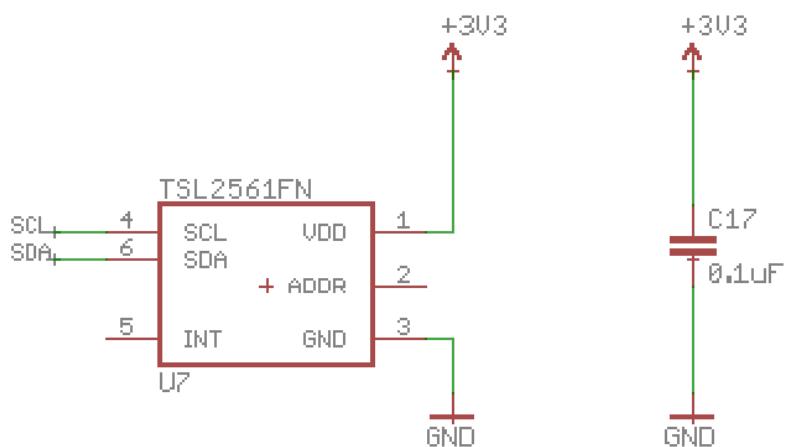


Fig. 34 Esquemático sensor de luz

Las líneas de alimentación deben estar desacopladas con un condensador de 0,1 μF.

El condensador de bypass debe tener baja resistencia efectiva en serie (ESR) y baja efectividad (ESI), tales como los condensadores cerámicos, que proporcionan una baja impedancia a tierra en altas frecuencias para manejar corrientes transitorias causadas por la comutación lógica interna.

Pantalla

La pantalla LCD de 1.3" de memoria SHARP es una mezcla entre eInk (e-paper) y una pantalla LCD. Tiene el bajo consumo de eInk y la actualización rápida de un LCD. Este modelo tiene un fondo de plata mate, donde los píxeles se muestran como pequeños espejos. No tiene luz de fondo, pero es legible por la luz del día.

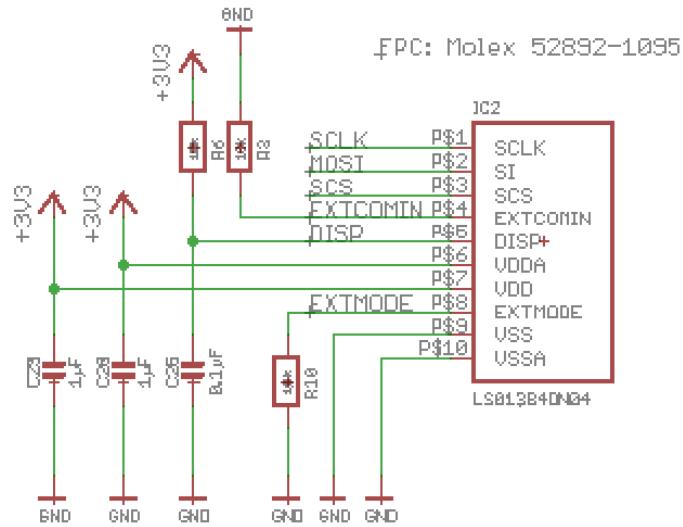


Fig. 35 Esquemático pantalla

Según el datasheet son recomendados los siguientes condensadores por el filtrado de alimentación. C1=0.1uF, C2=1uF y C3=1uF. SCLK, SI y SCS son las conexiones de interfaz de SPI utilizada para la comunicación con el módulo.

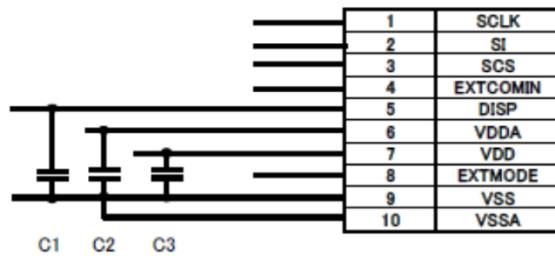


Fig. 36 Conexión datasheet pantalla

El pin DISP se conecta a una señal digital del microcontrolador ya que controla la visualización de datos o no en pantalla.

Las señales EXTMODE y EXTCOMIN que habilitan configuración externa de la pantalla, son conectadas a GND para deshabilitarlo.

Bluetooth

MDBT40 es un módulo BT4.0 y BT4.1 (Bluetooth de baja energía o BLE) diseñado con base en la solución SoC de Nordic nRF581xxx, que incorpora interfaces GPIO, UART, I2C y ADC. Certificado en EEUU, Japón y UE.

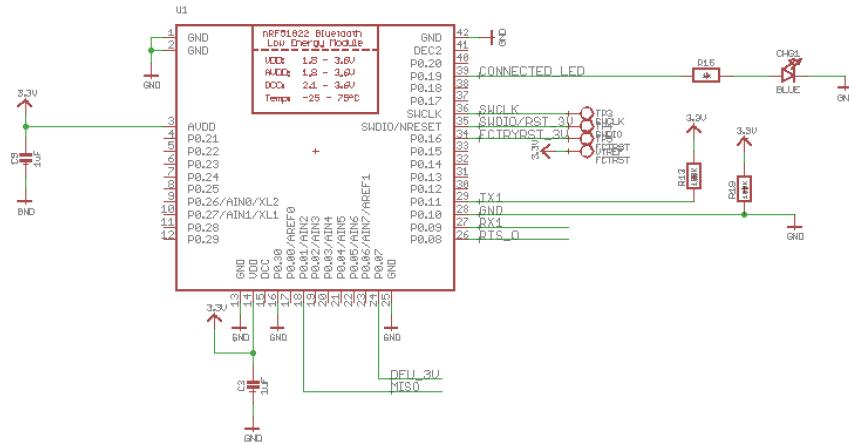


Fig. 37 Esquemático Bluetooth

Se han puesto condensadores de bypass de 1uF en alimentación, AAVDD y VDD para evitar ruido.

Se han dejado disponibles los pines de programación del módulo Bluetooth, swclk y swdio en el caso de que hubiera que cambiarle el firmware.

El pin DFU se utiliza para actualizar el firmware o volver al estado de fábrica. Si se pone este pin a GND antes de encender el módulo puede actualizarse a través de la aplicación de Adafruit llamada Bluefruit LE Connect. Si se pone el mismo pin a GND durante más de 5 segundos una vez encendido el módulo y después a tensión, éste vuelve al estado de fábrica.

MOD selecciona si se quiere programar por comando AT, poniendo este pin a señal digital HIGH o por UART comunicación serie directa, poniéndolo a señal digital LOW.

CTS va a GND para permitir la transferencia de datos. Poniendo FCTRST a GND es otra posibilidad para restaurar el módulo a fábrica.

Por otro lado, se ha añadido un led azul a uno de los pines que nos avisa cuando el Bluetooth está conectado al móvil.

Microcontrolador

ATmega1284P es un microcontrolador CMOS de 8 bits de baja potencia. Al ejecutar instrucciones potentes en un solo ciclo de reloj, el ATmega1284P alcanza rendimientos cercanos a 1MIPS por MHz. Esto permite al diseñador del sistema optimizar el dispositivo para el consumo de energía frente a la velocidad de procesamiento.

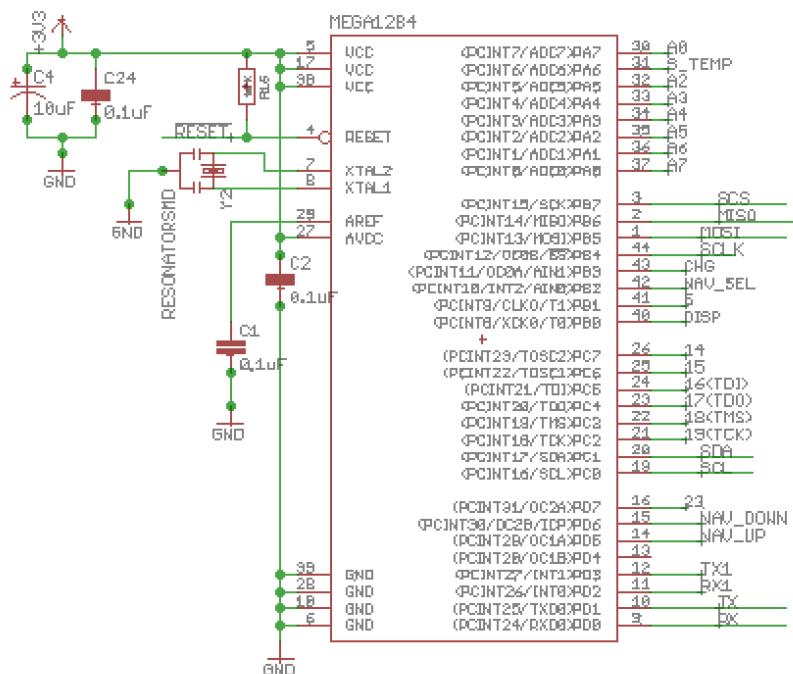


Fig. 38 Esquemático microcontrolador

Se han puesto condensadores cerámicos de 0.1uF en alimentación para eliminar ruido y un condensador de 0.1uF en paralelo a otro de 10uF electrolítico para eliminar ruido en mayor rango de frecuencia. Para que funcione a 3.3V se ha añadido un cristal de 8MHz.

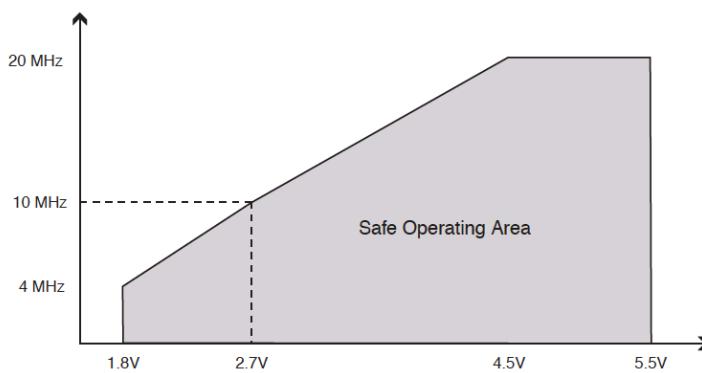


Fig. 39 Gráfica tensión segura según la frecuencia

Carga

El dispositivo bq5105x es un receptor de energía inalámbrico compatible con Qi de alta eficiencia con un controlador de carga de batería Li-Ion / Li-Pol integrado para aplicaciones portátiles. Los dispositivos bq5105xB proporcionan una eficiente conversión de energía AC-DC, integran el controlador digital necesario para cumplir con el protocolo de comunicación Qi v1.2 y proporcionan todos los algoritmos de control necesarios necesarios para una carga de batería Li-Ion y Li-Pol eficiente y segura.

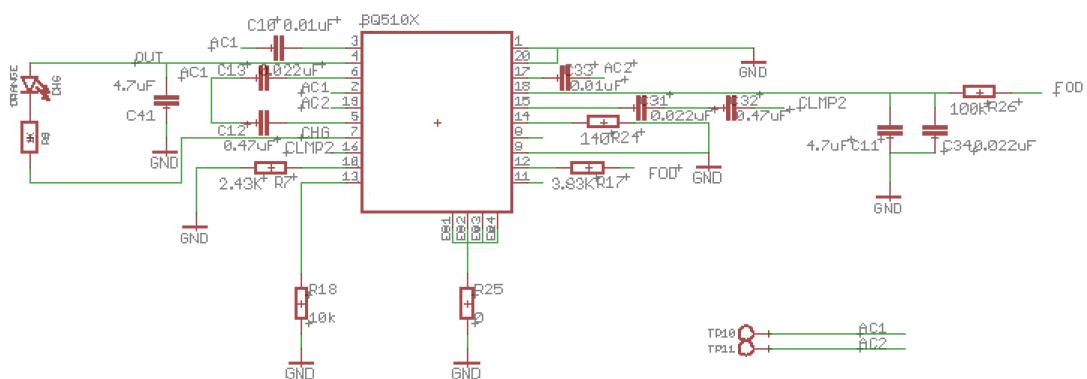


Fig. 40 Esquemático carga por inducción

Para la etapa de carga se ha utilizado una etapa de carga por inducción a través de protocolo Qi, utilizando el módulo BQ51051. Se trata de un receptor inalámbrico integrado para carga de baterías Li-Ion/Li-Pol especialmente diseñado para pequeñas aplicaciones portátiles. Proporciona conversión de alimentación AC-DC.

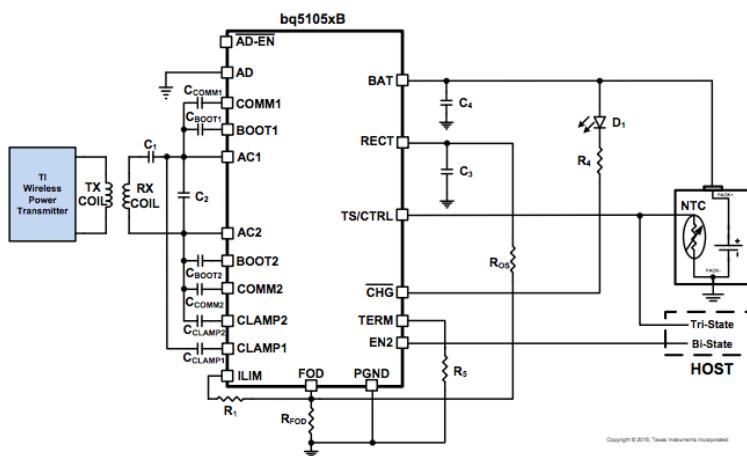


Fig. 41 Conexiones datasheet carga inalámbrica

La etapa utilizada es el montaje típico de funcionamiento. Su diseño y montaje ha supuesto un análisis completo de las distintas etapas, así como sus cálculos asociados.

Caben destacar algunos puntos específicos:

- Se ha utilizado el pin de salida CHG para monitorizar desde el microcontrolador si se está cargando la batería a través de una interrupción. Por otro lado también tiene un led naranja que te indica cuando se está cargando.
- El pin ILIM sirve para configurar la corriente de carga de la etapa a través de una resistencia.

$$R_{lim} = R5 + R_{fod} = 3.83K + 140 = 3.97K$$

$$I_{charge} = K_{ilim}/R_{lim} = 314/3970 = 80mA$$

- Además, con el pin TERM a través de un resistencia a GND se calcula del % de límite aplicable a dicha corriente, como protección.

$$\%I_{bulk} = R_{Term}/K_{Term} = 2.4K/240 = 10\% \text{ (10\% of } 80mA = 8mA)$$

Otros circuitos

- Se han colocado resistencias de pull-up de 4.7K en las señales de I2C ya que es un valor adecuado para la velocidad de transmisión y nos evita, al tener varios integrados que funcionan por interfaz I2C, colisiones entre distintos voltajes ya que operan sobre un bus común. Este valor debe ser grande para que la corriente del pin sea pequeña, pero no excesivamente grande como para que no caiga mucha tensión en la resistencia.
- Regulador de tensión dar alimentación principal al microcontrolador y resto de circuitos, a 3.3V desde la batería y otro regulador de 3.3V a 1.8V para el sensor de pulso.
- Divisor de tensión con resistencias de 100K para calcular la tensión de la batería a través de un pin analógico del micro.
- Conectores varios para conexión rápida y robusta de los distintos elementos: batería, pantalla, circuitos de sensores externos.
- Conectores de programación: FTDI e ICSP.

4.6. Diseño de la PCB

El diseño de la PCB en general ha seguido unas pautas básicas de diseño de circuitos:

- Planos de masa en TOP y BOTTOM conectados a GND.
- Cumplimiento de normas de distancias y agujeros (DRCs) asociadas a las especificaciones indicadas por el fabricante.
- Utilización de vías para permitir la unión de los planos de masa y mejorar la inmunidad de la placa a ruido.
- Pistas directas y sin zonas con codos o de malos ángulos.
- Evitar vías en pads de componentes.

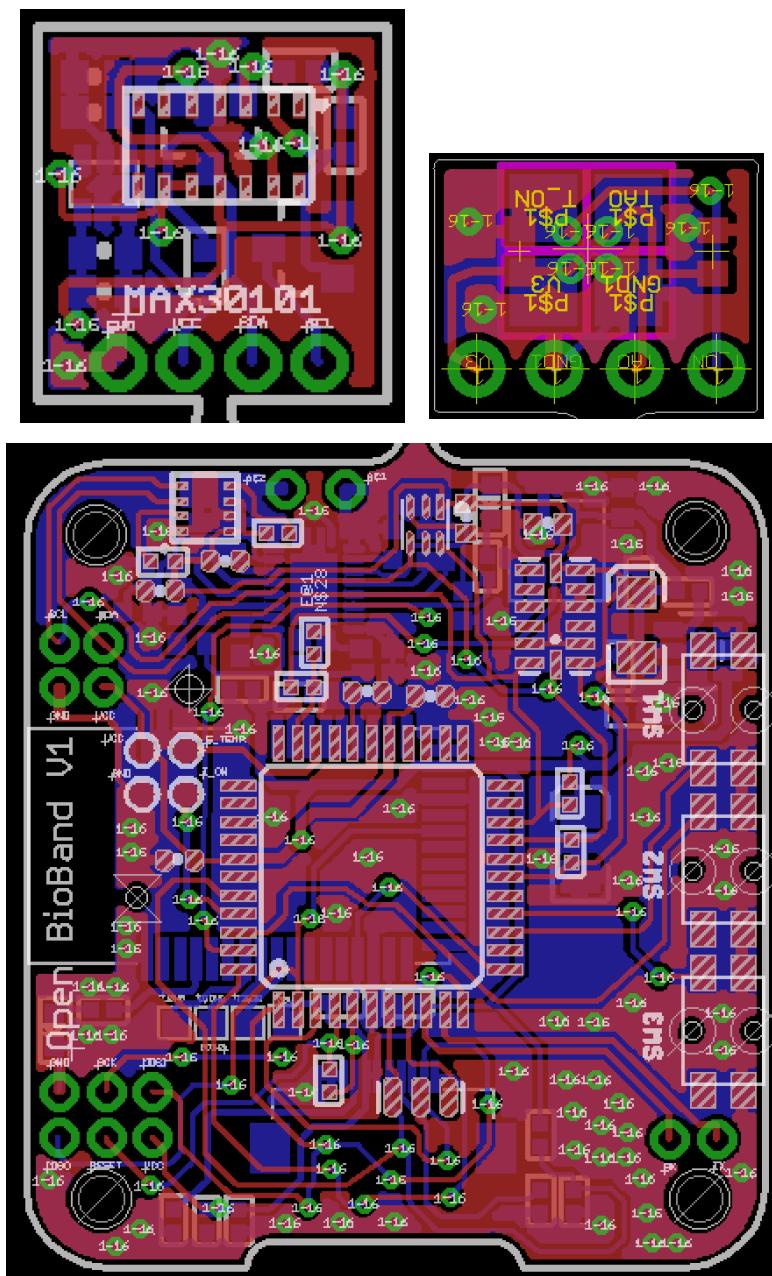


Fig. 42 Diseño PCB sensor de pulso, sensor de temperatura y Open Bioband

Acelerómetro

Se ha colocado en el top de la PCB teniendo en cuenta las especificaciones de los ejes de aceleración en el datasheet.

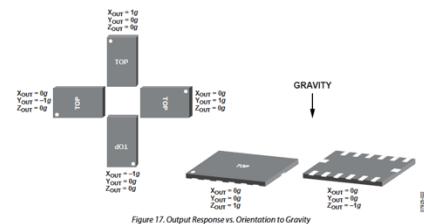
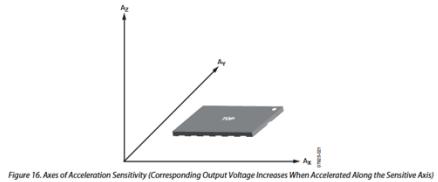


Fig. 43 Diseño acelerómetro en datasheet

Pulso

Este es uno de los sensores que deben ir pegados a la muñeca para que no de valores erróneos por lo que se ha diseñado una pcb individual, sólo con dicho sensor.

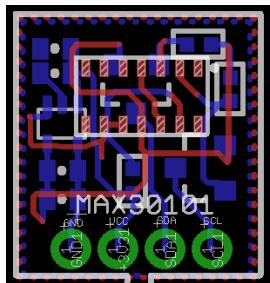


Fig. 44 PCB sensor de pulso

Como se puede ver el sensor va en la cara top de la PCB y el resto de componentes va en la cara bottom para que no molesten a la hora de posicionarlo en la carcasa, ya que en un primer diseño iban componentes por ambas caras y entorpecía tanto a la colocación como a la medición.

Ha sido soldado con plancha caliente debido a su encapsulado de pequeño tamaño y pads en parte inferior.

Temperatura

Al igual que el de pulso, este sensor tiene que ir en una placa a parte, lo más pequeña posible y sin plano de masa. Uno de los factores es que tiene que llegarle toda la temperatura de la piel posible sin que se disipe nada de calor por el camino. [15]



Fig. 45 Diseño PCB sensor de temperatura

Dicha PCB debe tener los pads de cobre en la zona de las patas del sensor y por el otro lado, estos pads deben tener mayor superficie para que haga mejor contacto con la piel. Además, debe ser lo más pequeña posible ya que de esta forma la pérdida de calor es menor. Por otro lado, los cables para unir al microcontrolador deben ser lo más finos posible y de níquel, ya que su conductividad térmica es más baja que la del cobre.

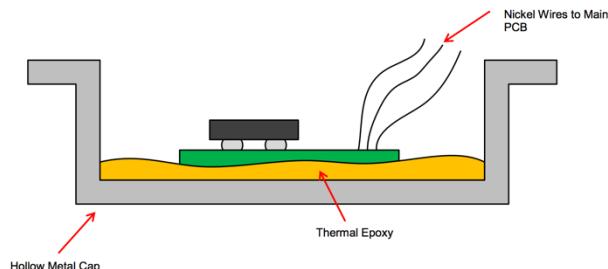


Fig. 46 Cableado y diseño LMT70

Se utiliza plancha caliente, para su soldadura dadas sus dimensiones y encapsulado. Se trata de una labor muy delicada por lo que se tuvo que encargar pantalla de soldadura para poder estañar de manera manual los pads de la placa con pasta de estaño.

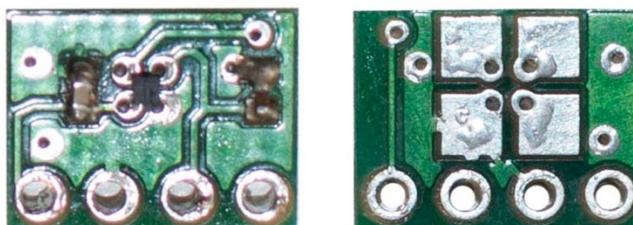


Fig. 47 PCB sensor de temperatura corporal

Ambiente y luz

El sensor de luz al igual que el de ambiente se han colocado en un lateral de la PCB y se le ha diseñado agujeros a la carcasa para que estén lo más cerca posible del exterior.

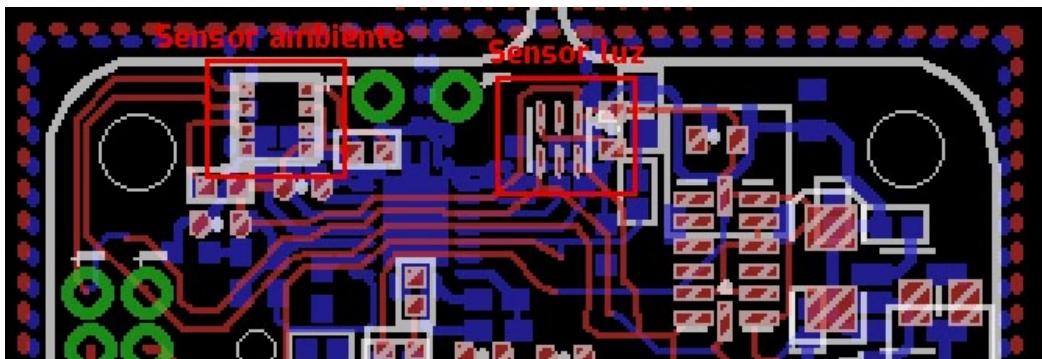


Fig. 48 PCB detalles sensores de ambiente y luminosidad

Pantalla

Para la pantalla se ha diseñado una hendidura en la PCB del tamaño del doblez teniendo en cuenta las características del datasheet.

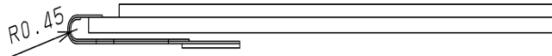


Fig. 49 Diseño pantalla datasheet

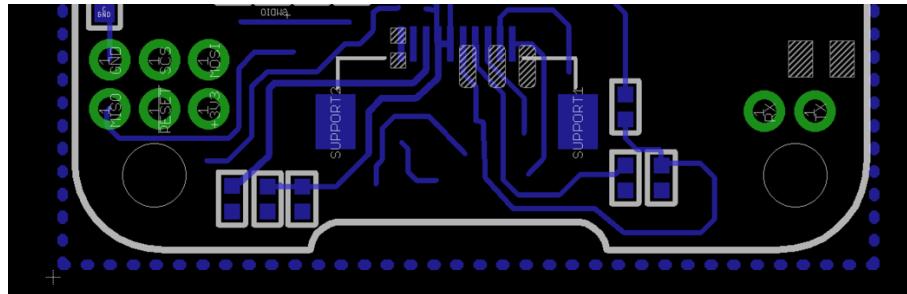


Fig. 50 PCB detalle diseño pantalla

Además, se ha diseñado una pieza impresa junto a la carcasa para sujetar la pantalla sin que toque la PCB. Ésta puede verse más adelante.

Microcontrolador

El microcontrolador ha sido posicionado en la parte central de la PCB para poder hacer pistas cortas a cualquiera del resto de etapas.

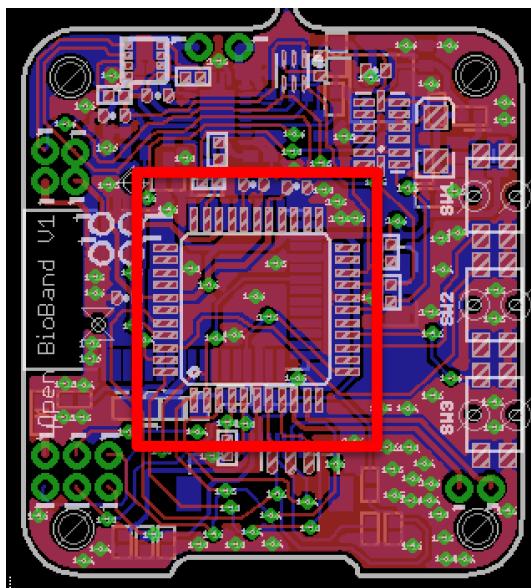


Fig. 51 PCB detalle microcontrolador

Bluetooth

Como es recomendado en el datasheet no se ha puesto masa ni otros componentes en el lugar de la antena del bluetooth, ya que puede afectar al alcance con otros dispositivos.

Las zonas cercanas a la antena han sido protegidas de ruido a través de planos de masas a GND con vías para unirlos en cada TOP y BOTTOM.

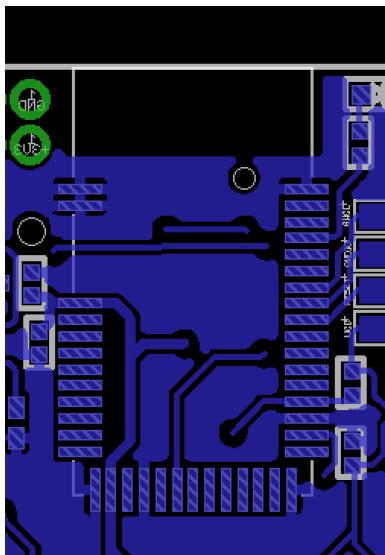


Fig. 53 PCB diseño bluetooth

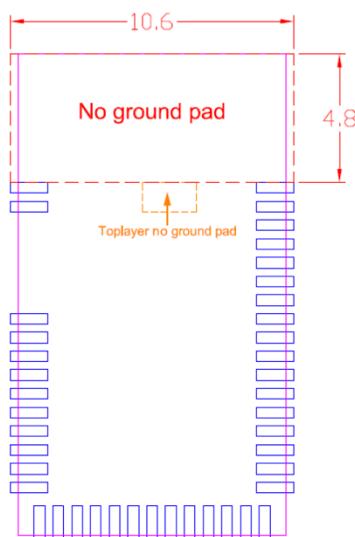


Fig. 52 Datasheet diseño bluetooth

Carga

Esta etapa requiere de un diseño de PCB complejo en cuanto a disposición de componentes y planos de masa, que han sido realizados según las recomendaciones indicadas en su datasheet.

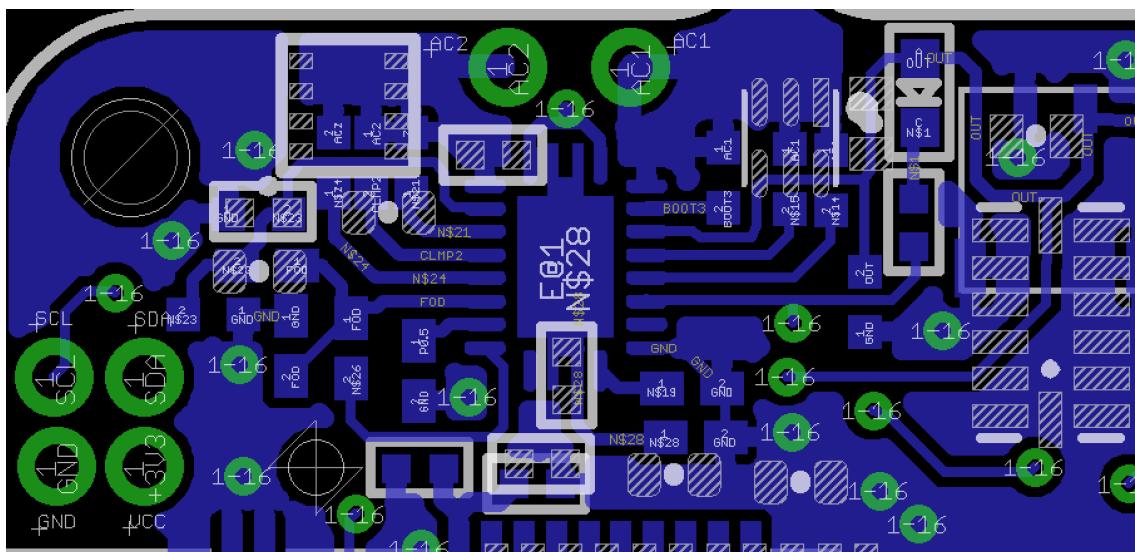


Fig. 54 PCB detalle carga por inducción

Las señales AC1 y AC2 han sido configuradas a través de las etapas capacitivas indicadas en la hoja de características, y conectadas a través de planos de masa. También en la señal BAT por motivos de disipación.

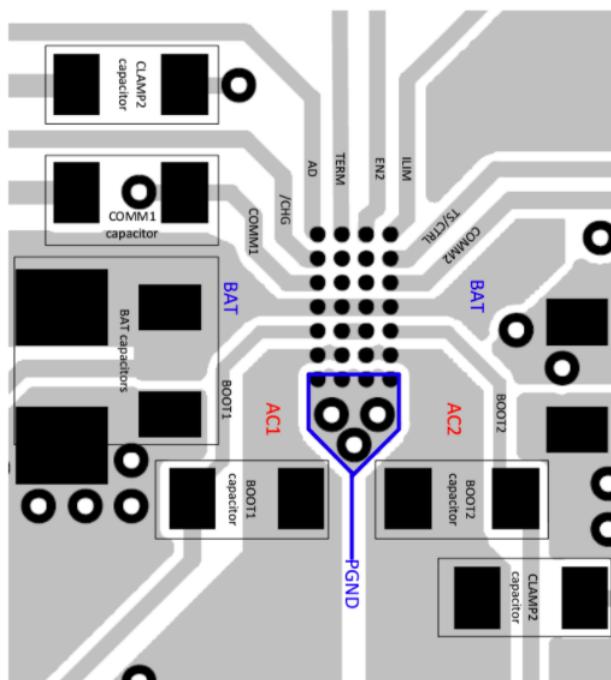


Fig. 55 Datasheet diseño PCB

4.7. Fabricación

4.7.1. Versión 0

En una primera fase se construyó un prototipo de una pulsera realizada con Arduino donde se vieron las primeras posibilidades del proyecto: tamaño, especificaciones y tener un primer contacto con el mundo wearable. Dicho prototipo constaba de un Microduino, batería, pantalla OLED, pulsadores, Bluetooth ble112, conversor de tensión de 3V a 5V y el sensor de pulso pulse sensor.

Ver el ANEXO 5 para el desarrollo de dicho prototipo.



Fig. 56 Pulsera versión cero con Microduino

Tras las pruebas y elección de los sensores se realizó un prototipo con un Arduino Mega, ya que un Arduino Uno no tenía suficiente memoria, para testear el correcto funcionamiento de todos los sensores juntos.

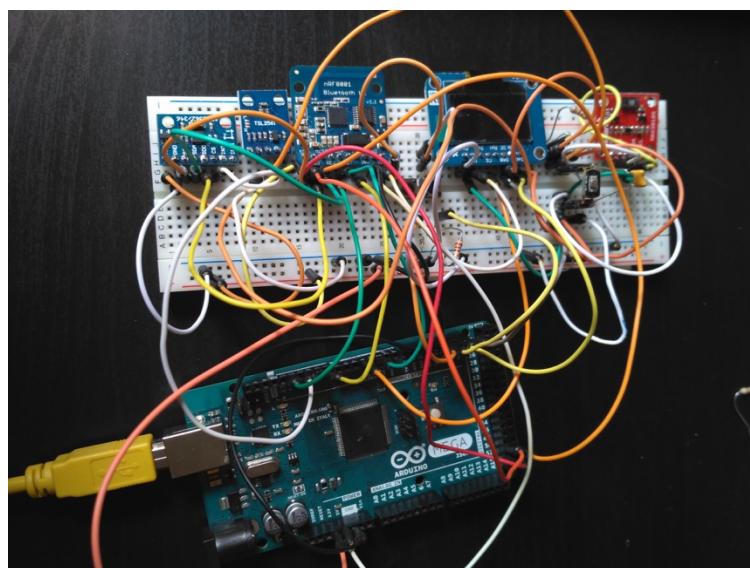


Fig. 57 Prototipo Bioband con Arduino

Una vez se tenía el prototipo definitivo se diseñó la placa de pruebas en el programa de CAD Eagle. Los criterios para crear las placas fueron:

- Usar componentes de encapsulado SMD (surface mounted devices)
- Miniaturizar todo lo posible.
- Minimizar el número de vías y hacerlas lo más cortas posibles.
- Crear buenos planos de masa.

Todas las placas de prueba fueron diseñadas , ensambladas y probadas por el autor del proyecto, a excepción de algunos sensores que es necesaria la cama caliente para realizar la soldadura.

4.7.2. OPEN BIOBAND versión 1

Para esta primera versión se optó por utilizar el sensor de temperatura de ehealth debido a que ya tiene su encapsulado metálico para que haga bien contacto con la piel, el módulo bluetooth de RFduino funcionando por UART para enviar tramas de datos de manera simple y unos botones circulares con desplazamiento ya que el diseño de la PCB es circular y de esta forma no era necesario realizar el encapsulado de los mismos.

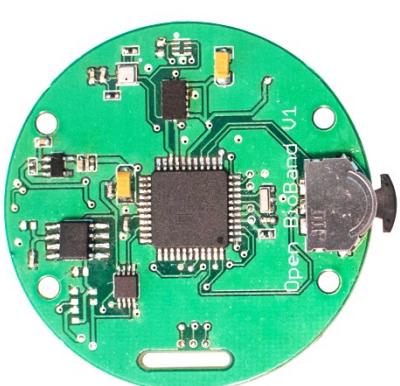


Fig. 59 Open Bioband versión 1 - parte delantera

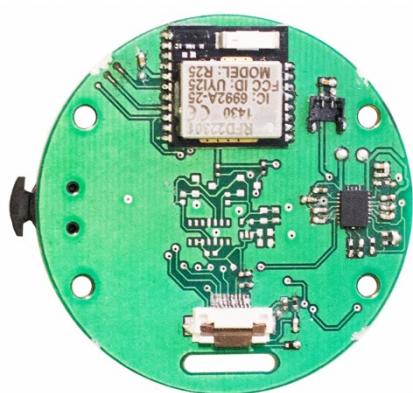


Fig. 58 Open Bioband versión 1 – parte trasera

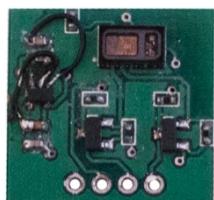


Fig. 60 Sensor de pulso

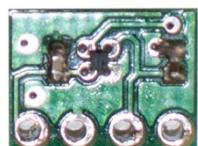


Fig. 61 Sensor de temperatura

4.7.3. OPEN BIOBAND versión 2

Tras la primera versión y la continua investigación se realizaron algunas mejoras en la siguiente versión de la PCB, la mayoría de ellas por el ahorro de espacio y que fuera lo más pequeña posible:

- Placa cuadrada
- Cambio de pulsadores a unos estándar más pequeños
- Cambio de sensor de temperatura por uno más pequeño
- Led para avisar que la pulsera se está cargando
- Cambio de módulo Bluetooth, dicho ble también funciona por UART pero su firmware te da la opción del envío de datos básico o por comandos at.
- Cambio en el diseño de la placa del sensor de pulso

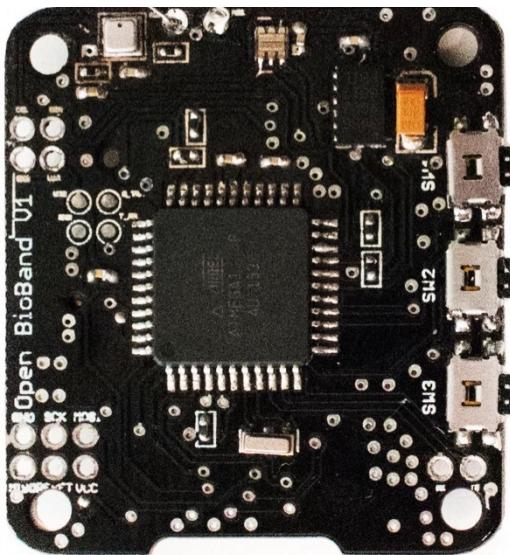


Fig. 63 Open Bioband versión 2 - parte delantera

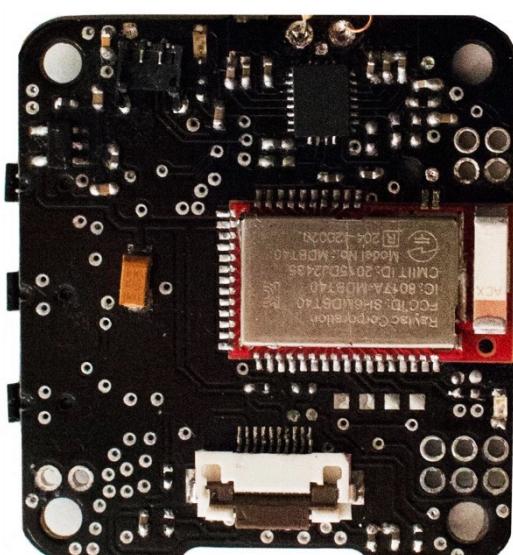


Fig. 62 Open Bioband versión 2 - parte trasera

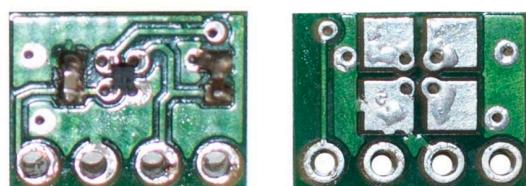
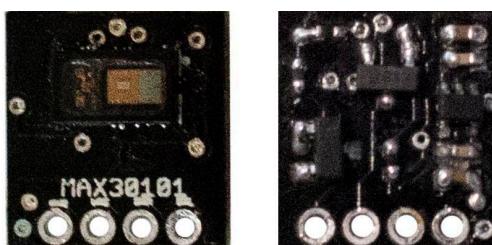


Fig. 64 Sensores de pulso (izquierda) y temperatura corporal (derecha)

5. Diseño software

5.1. Introducción

Se ha desarrollado un software para el dispositivo en el SDK de Arduino, para ello se han usado distintas librerías de código abierto basadas en dicho SDK para cada uno de los sensores, que serán explicadas más adelante. De cara a la integración de algunas etapas se han partido de estas librerías y códigos de terceros, pero han sido configuradas y modificadas para adaptarlas al hardware propio.

En un primer momento se han creado códigos independientes para cada uno de los sensores, de esta forma conforme se soldaba la parte correspondiente al sensor o actuador se iba programando para verificar que todo funcionaba correctamente. Posteriormente se ha creado un código completo con todos los sensores implementados en el dispositivo.

Para realizar cualquier tipo de programación con la tecnología de Arduino, es necesario instalar el IDE (Integrated Development Environment). Se puede encontrar en la página web arduino.cc y está disponible para descargar gratuitamente por cualquier usuario.

El entorno de programación incluye todas las librerías de API necesarias para compilar los programas. Una vez tenemos un programa cargado en el microcontrolador, el funcionamiento de Arduino se basa en el código cargado. La estructura de códigos se divide en dos partes fundamentales: setup y loop. Ambas partes del código tienen un comportamiento secuencial, ejecutándose las instrucciones en el orden establecido. El setup es la primera parte del código que se ejecuta, haciéndolo solo una vez al iniciar el código. En esta parte es recomendable incluir la inicialización de los módulos, alimentación, entre otros, que se vayan a utilizar. El loop es un bucle que se ejecuta continuamente, formando un bucle infinito.

5.2. Diseño de interacción con el usuario

Se ha diseñado la interfaz de usuario pantalla a pantalla para poder disponer de una experiencia de usuario completa al utilizar el dispositivo. Se ha intentado que cada una de ellas sea lo más visual, entendible y sencilla posible para el usuario de manera que, sin tener unos conocimientos previos, el usuario pueda interactuar con ella de forma natural.

Para crear cada uno de los iconos se ha diseñado la imagen en un tamaño adecuado con Photoshop y después, a través de un programa de la empresa Adafruit [16] llamado Img2Code se ha transformado la imagen bitmap a código para poder mostrarla. Este programa genera una tabla de píxeles en hexadecimal que será interpretada por el microcontrolador y la dibujará en la pantalla.

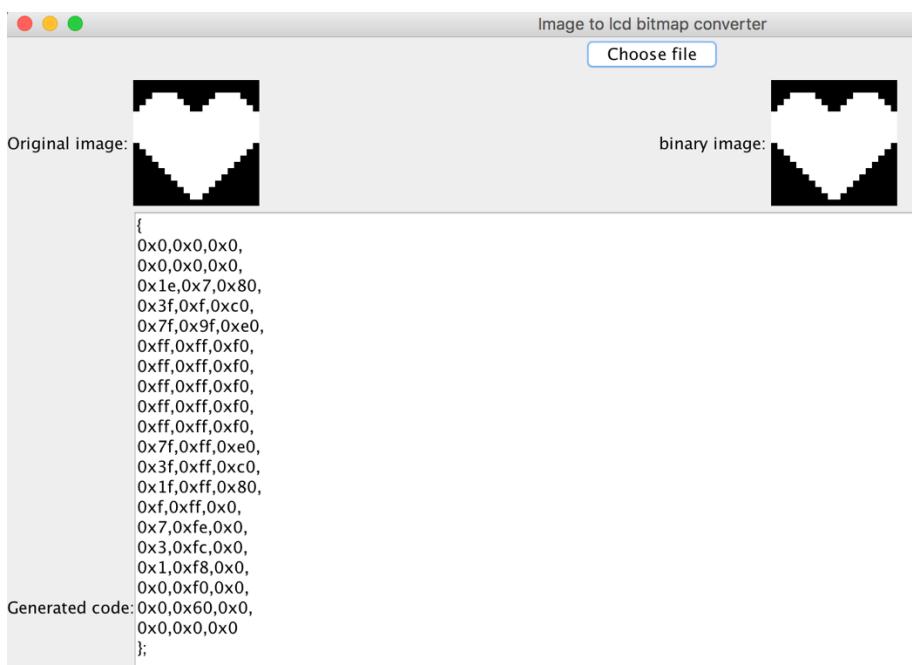


Fig. 65 Img2Code. Conversor de bitmap a hexadecimal

En el siguiente apartado puede verse la interfaz del dispositivo pantalla a pantalla.

5.3. Implementación

Tras comprobar el correcto funcionamiento de todas las etapas y diseñar la interfaz de visualización de los distintos sensores, se diseña el código principal que servirá de “Sistema Operativo” para Open Bioband y como ejemplo completo de programación de todos sus bloques.

Este firmware está dividido en el código principal, el cual es el encargado de configurar y llevar el flujo de trabajo del dispositivo, y las pantallas secundarias, en las cuales son tratados los datos y/o funcionalidades dependiendo de la pantalla en la que te encuentras.

5.3.1. Código principal

Es la parte principal del firmware, ya que incluye el Setup () donde se configuran, definen e inicializan todas las etapas que son utilizadas para el funcionamiento de Open Bioband.

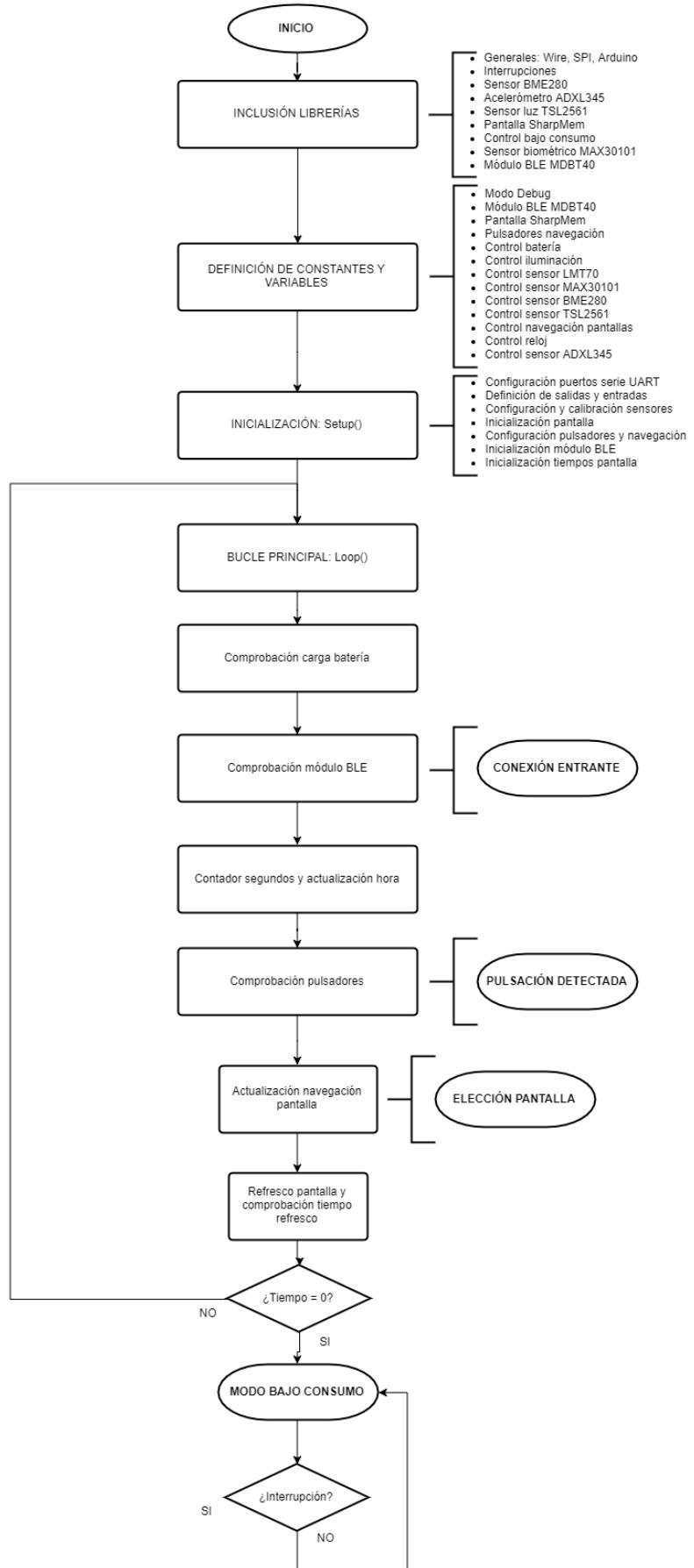
Tras dicha configuración, entra en el Loop(), el bucle principal del firmware. El cual se encarga de realizar las comprobaciones básicas del “Sistema Operativo”:

- Nivel de batería
- Conexión BLE
- Monitorización botones
- Refresco de pantalla
- Tiempo actual

Gracias al tiempo actual, constantemente está pendiente del tiempo pasado desde la última interacción del usuario y es capaz de entrar en modo Bajo Consumo en el momento en el que se ha superado ese límite. Dicho límite está preconfigurado a 15 segundos, pero puede ser cambiado en la pantalla de configuración.

En el caso de entrar en modo Bajo Consumo, se queda en ese estado hasta que detecta una interrupción por pulsación del botón central, lo que supone la interacción del usuario.

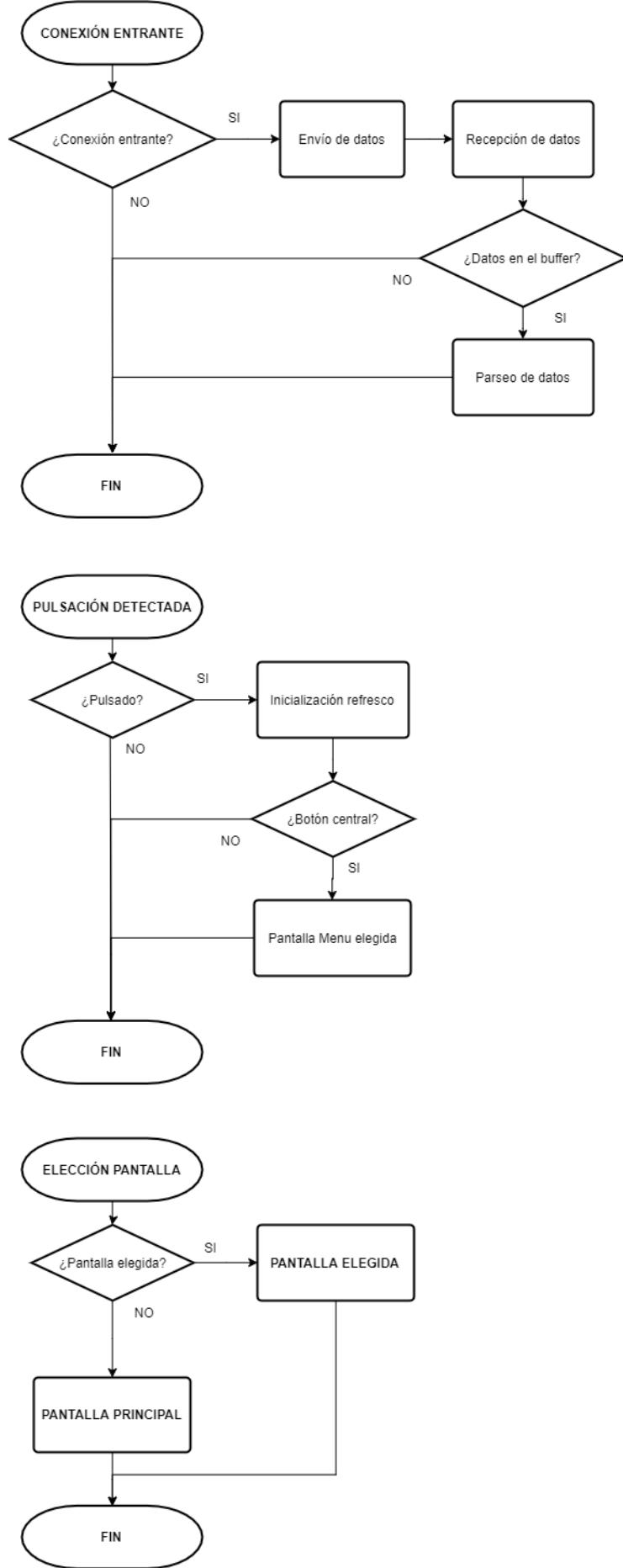
Pulsera biométrica Open Source para la monitorización del usuario



A su vez dentro de este bucle principal existen 3 subrutinas más complejas que monitorizan las distintas señales que pueden interactuar con la visualización y navegación del dispositivo:

- Conexión entrante BLE: en el caso de detectar conexión BLE, monitoriza los datos entrantes y los parsea para ser utilizados para cualquier funcionalidad, a la vez que puede enviar datos si se ha configurado su envío previamente.
- Pulsación detectada: al detectar pulsación, refresca el tiempo de Standby para evitar que entre en estado Bajo Consumo y entra en el menú de selección si se ha presionado el botón central.
- Elección de pantalla: comprueba en todo momento la pantalla en la que está el usuario o si ha cambiado en alguna subrutina. Esta es la rutina principal de navegación y refresco de pantallas.

Pulsera biométrica Open Source para la monitorización del usuario



5.3.2. Pantallas secundarias

PANTALLA 1. PANTALLA PRINCIPAL - pantallaPrincipal()

Esta pantalla es la pantalla principal que aparece cuando se inicia la pulsera o si se va al modo Standby. Se puede ver el porcentaje de batería, si está o no conectado a otro módulo Bluetooth y cuando lo pones a cargar aparece una animación de carga en la batería y el símbolo del rayo.

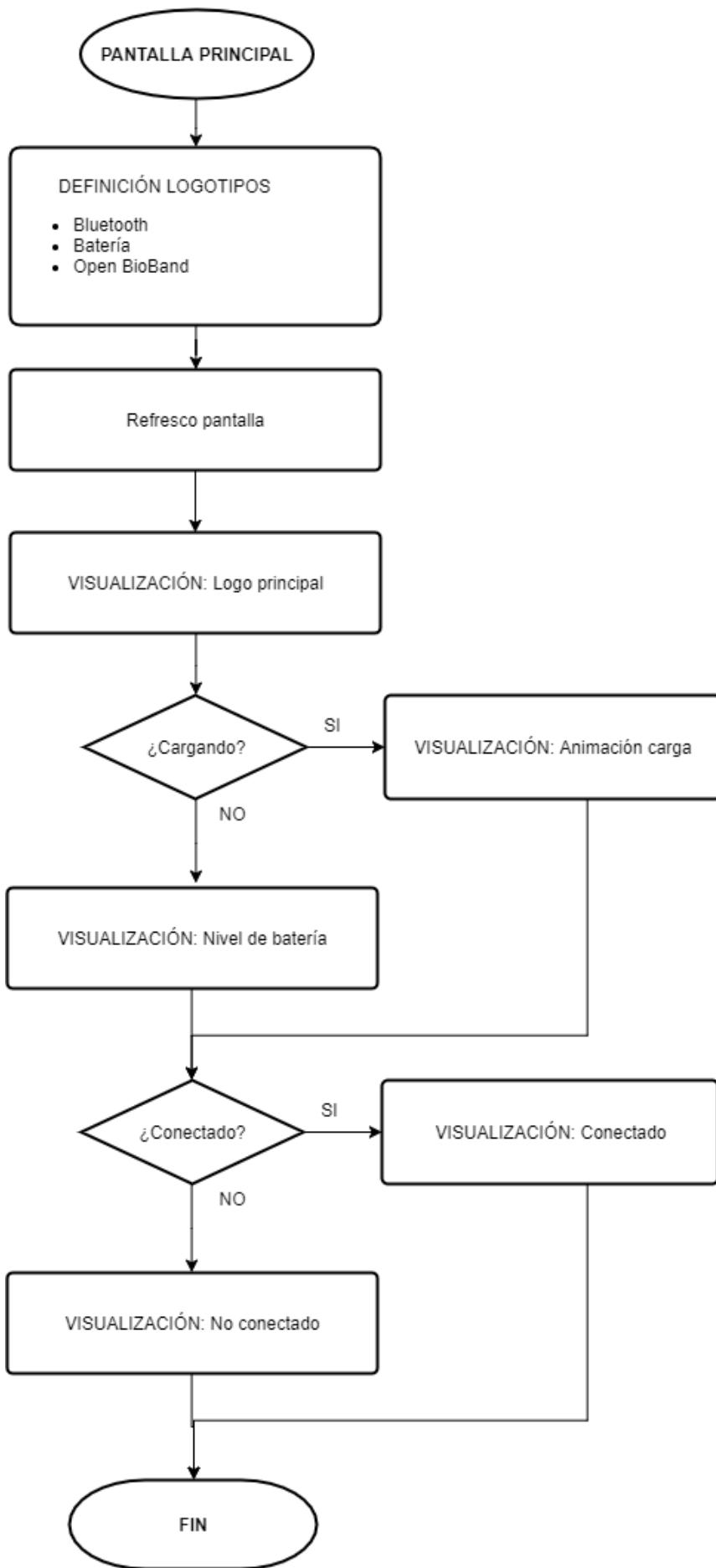
Para entrar al menú debes pulsar el botón de selección.



Fig. 67 Pantalla principal



Fig. 66 Pulsera en modo carga



PANTALLA 2. MENÚ - pantallaMenu()

Esta pantalla es el menú del dispositivo, desde aquí puedes acceder a las diferentes aplicaciones de la pulsera. Con los botones de arriba y abajo puedes navegar por los iconos y con el de selección entrar en la aplicación.

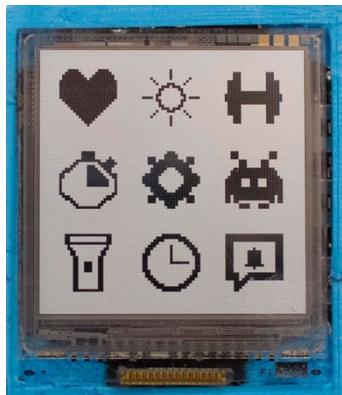
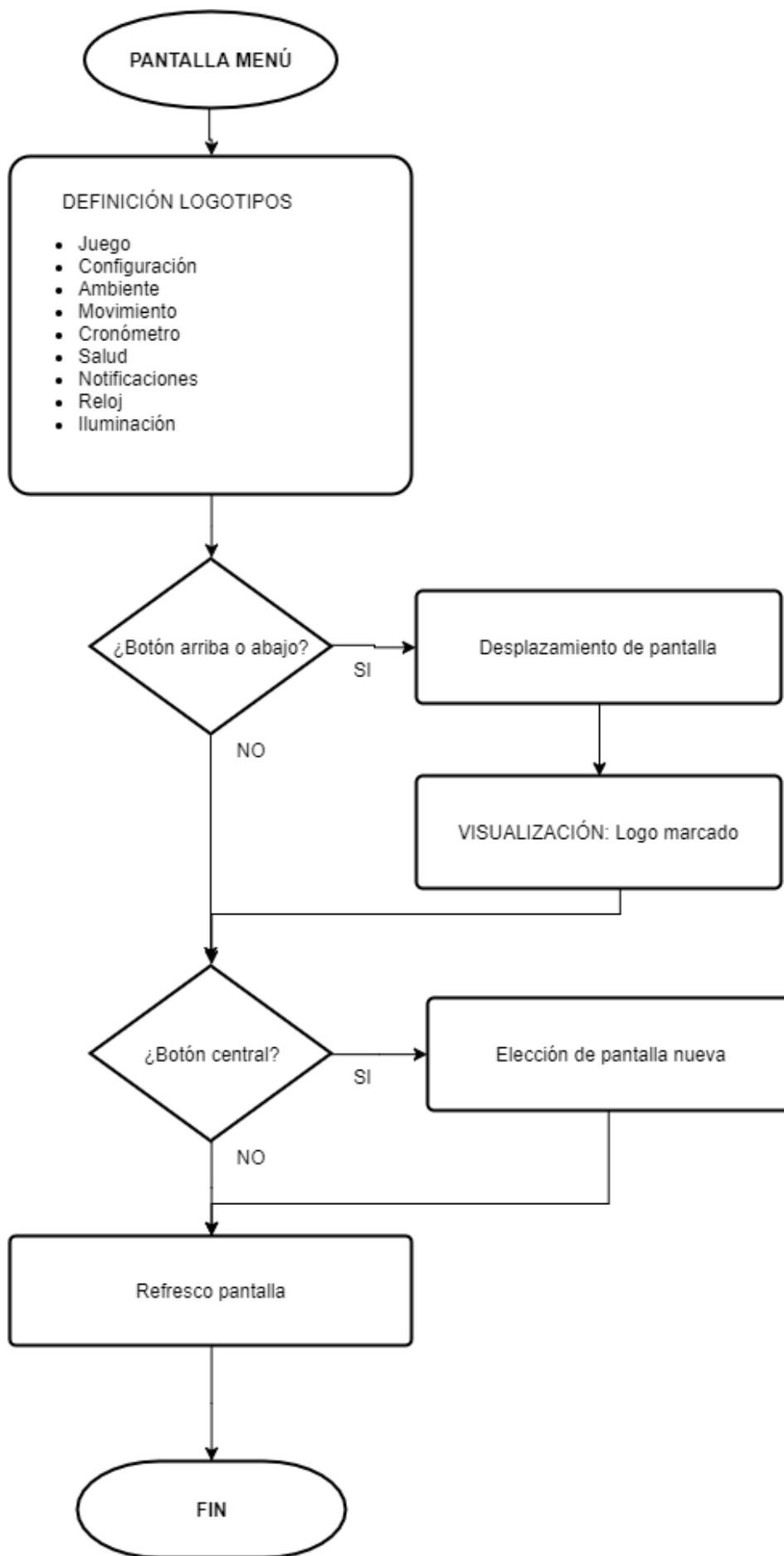


Fig. 68 Pantalla menú

1. El corazón – Aplicación de salud
2. El sol – Aplicación de ambiente
3. Pesas – Aplicación de ejercicio/movimiento
4. Cronómetro – Aplicación de cronómetro
5. Engranaje – Configuración de la pulsera
6. Marciano – Juego
7. Iluminación – Poner la luz de la pulsera
8. Reloj – Reloj
9. Campana – Notificaciones del móvil



PANTALLA 3. SALUD - pantallaBiométrica()

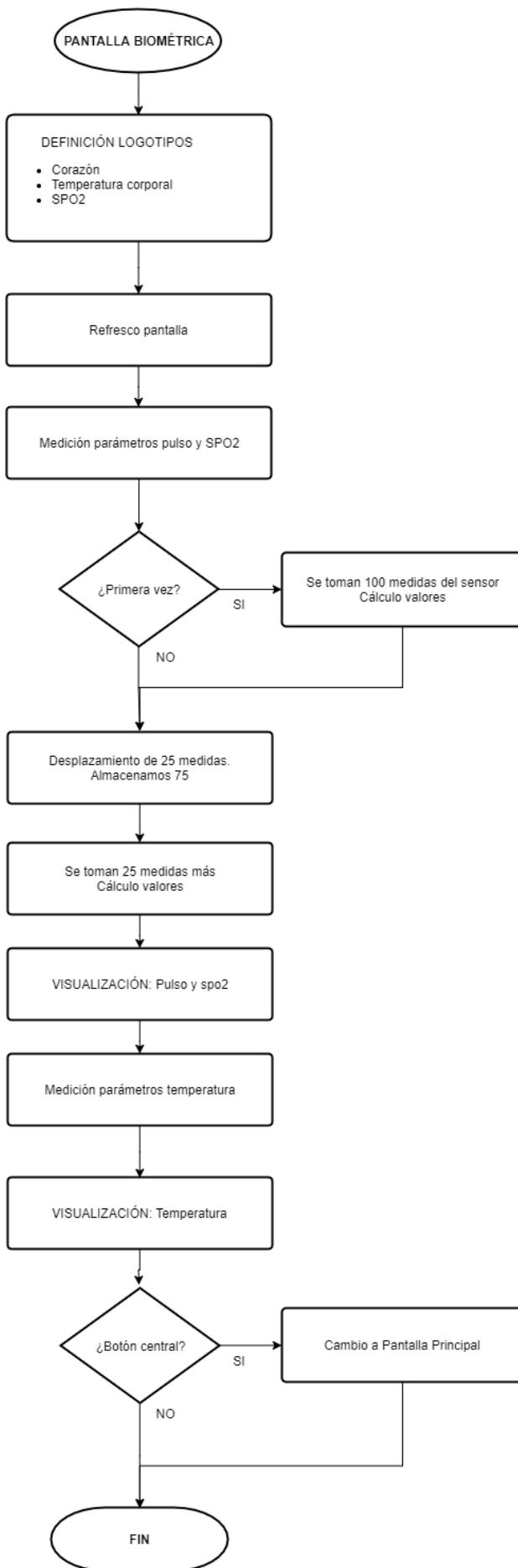


Esta pantalla muestra los valores de los sensores de temperatura, pulso y SPO2. Muestra nuestra temperatura corporal en grados centígrados, pulso en BPM (pulsaciones por minuto) y SPO2 en %.

Para salir de esta pantalla deberás pulsar el botón de selección.

Fig. 69 Pantalla biométrica

Pulsera biométrica Open Source para la monitorización del usuario



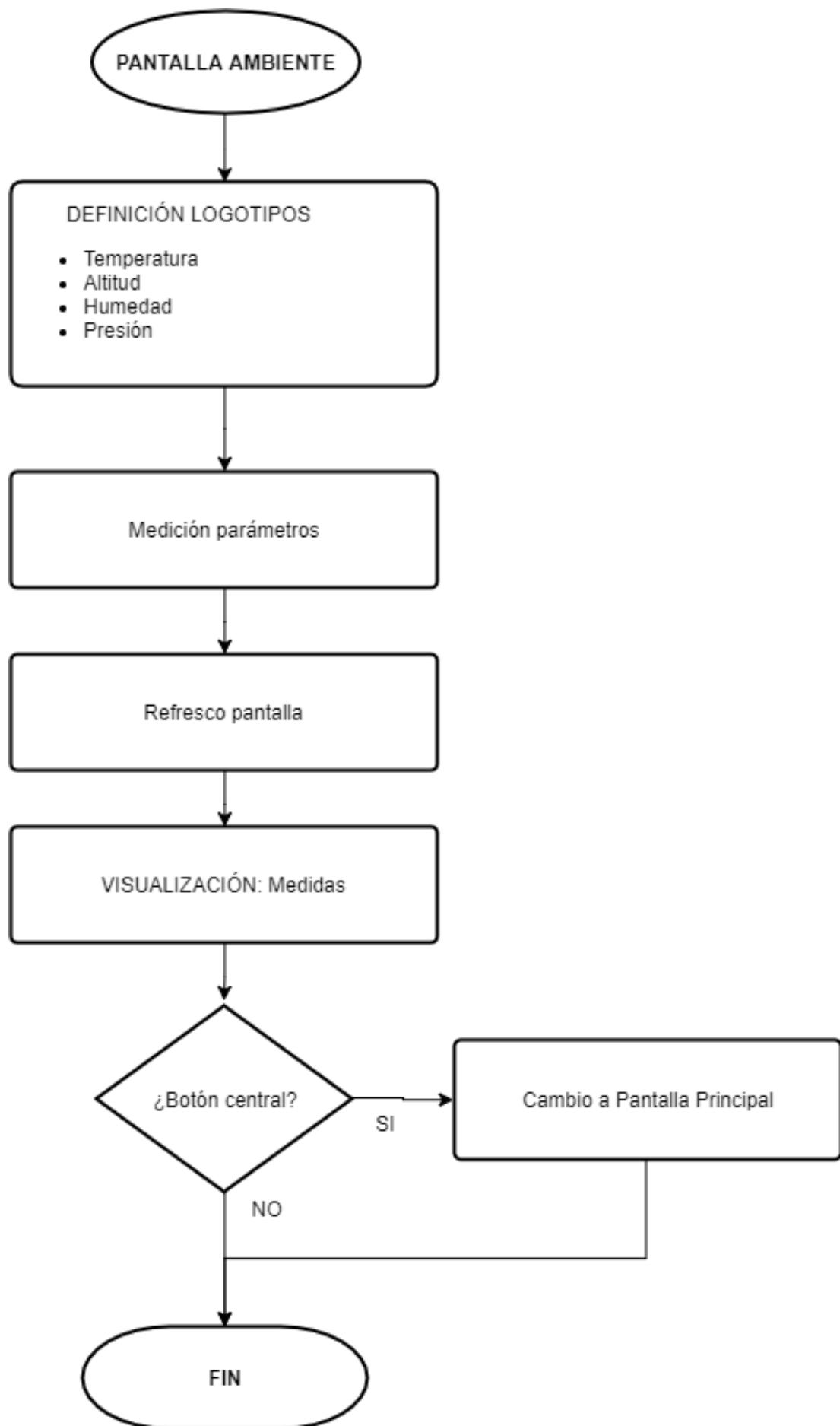
PANTALLA 4. AMBIENTE - pantallaAmbiente()



Esta pantalla muestra los valores del sensor de ambiente, BME280. Muestra la temperatura en grados centígrados, la humedad en % de humedad relativa, la altitud en metros y la presión en KPa.

Para salir de esta pantalla deberás pulsar el botón de selección o esperar que termine el tiempo de Standby.

Fig. 70 Pantalla ambiente



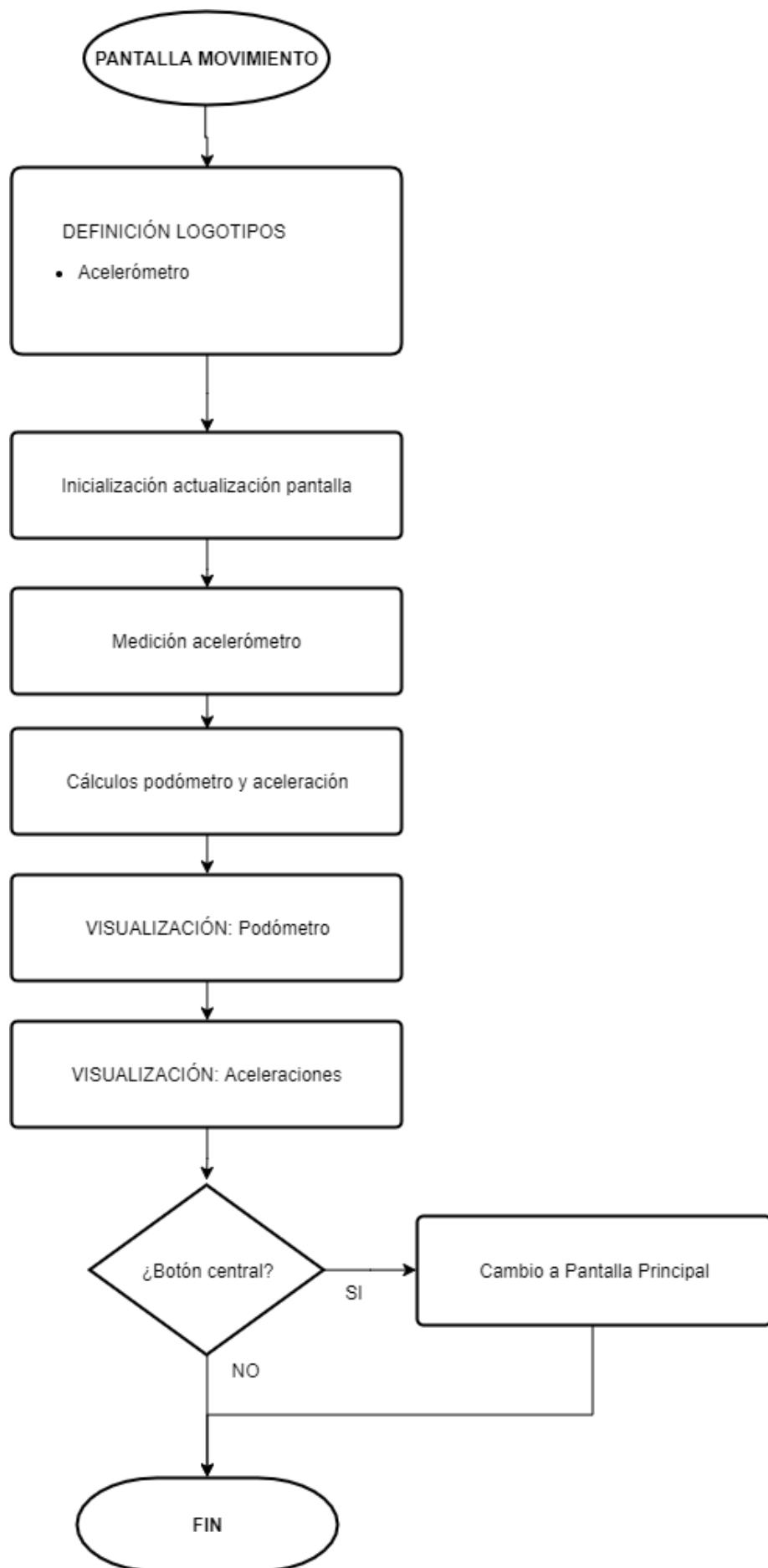
PANTALLA 5. EJERCICIO - pantallaMovimiento()



Esta pantalla muestra los valores del acelerómetro. Muestra por un lado los pasos, las calorías y la distancia recorrida y por otro los valores de x, y, z en g. Para calcular las calorías correctamente es necesario poner el peso y altura del usuario en Ajustes.

Para salir de esta pantalla deberás pulsar el botón de selección.

Fig. 71 Pantalla movimiento



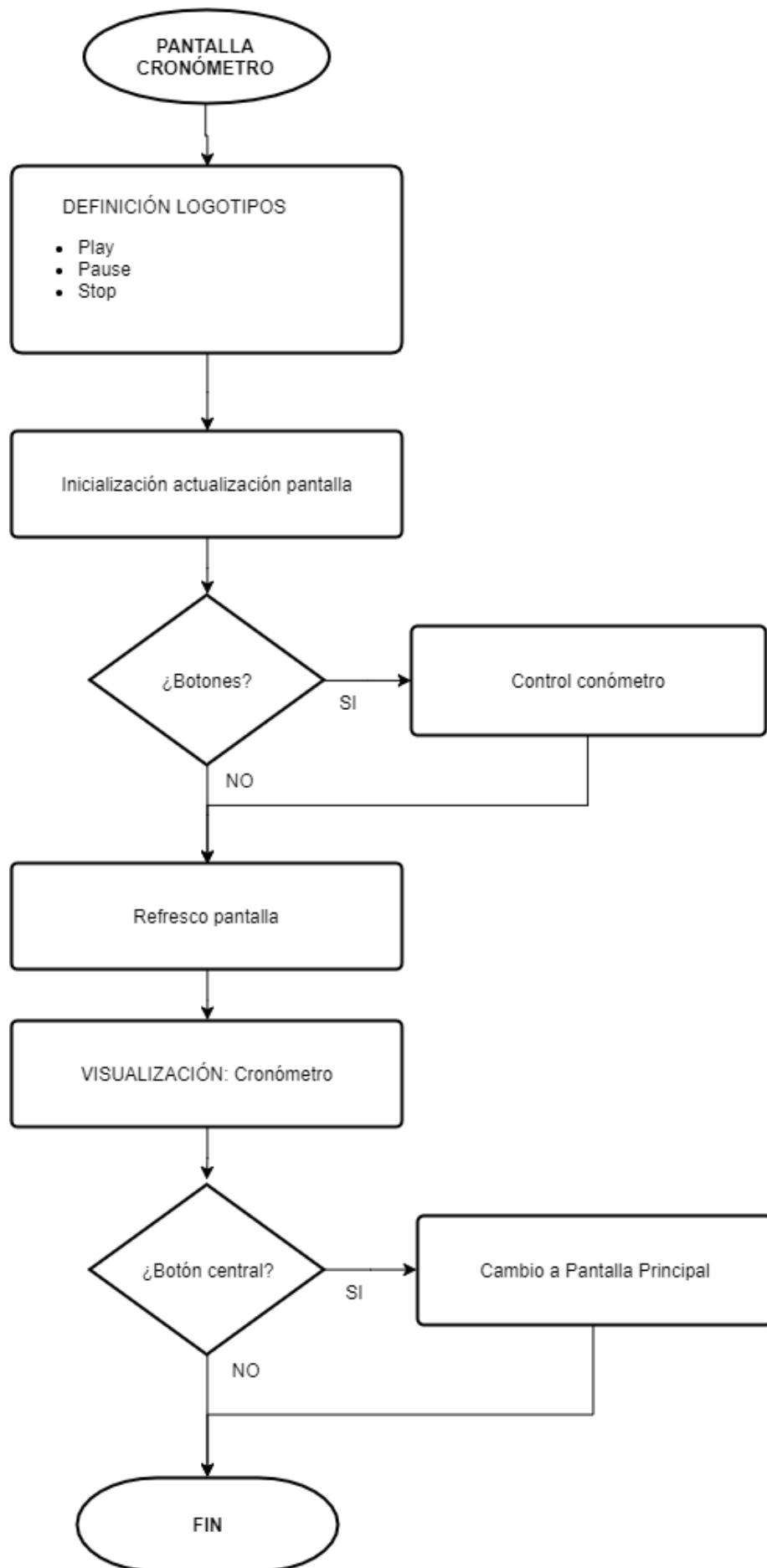
PANTALLA 6. CRONÓMETRO - pantallaCronometro()



Esta pantalla muestra un cronómetro. Si le das al botón de arriba empieza la cuenta y si le vuelves a dar la pausa. Si quieres volver a ponerlo a cero es dándole al botón de abajo.

Para volver al menú deberás pulsar el botón central de selección.

Fig. 72 Pantalla cronómetro



PANTALLA 7. AJUSTES - pantallaAjustes()

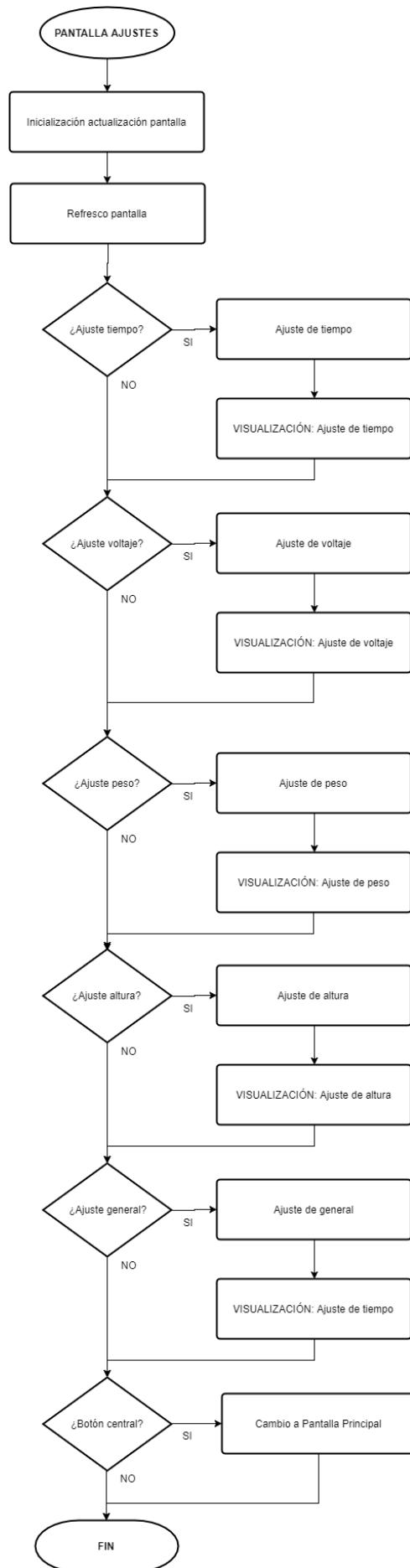
Esta pantalla muestra posibles configuraciones del dispositivo.



Fig. 73 Pantalla ajustes

- i. Subir o bajar el tiempo hasta que la pulsera va a Standby. Por defecto son 15 segundos.
- ii. Mostrar o no la batería en la pantalla principal.
- iii. Configurar tu peso para el cálculo de calorías. Por defecto es 50 kg.
- iv. Configurar tu altura para el cálculo de calorías. Por defecto es 160 cm.
- v. Ok vuelve de nuevo al menú.

Pulsera biométrica Open Source para la monitorización del usuario



PANTALLA 8. JUEGOS - pantallaJuego()

Esta pantalla es un pequeño juego en el que debes utilizar el pulsador de arriba y el de abajo para intentar llevar la pelota por los agujeros. Sino lo consigues pierdes y vuelves a empezar.

Para volver al menú deberás pulsar el botón central de selección.

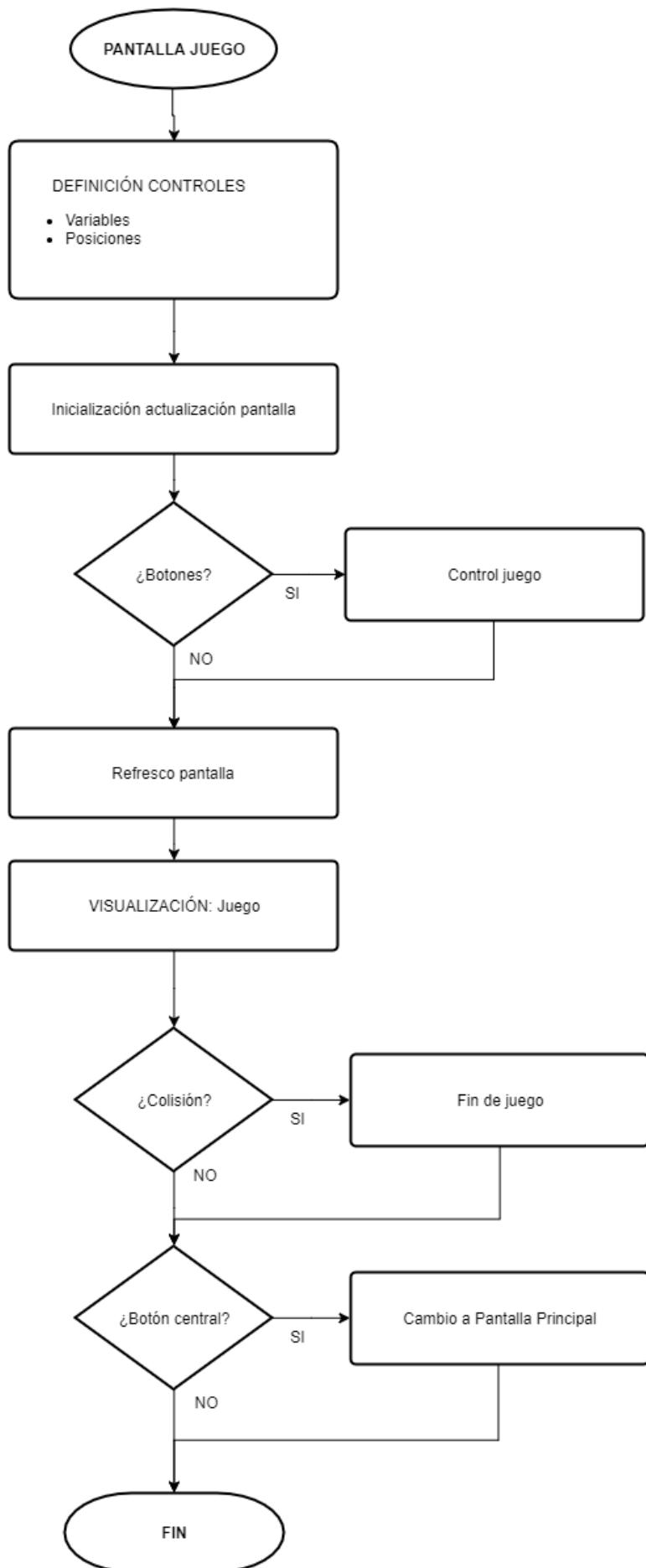


Fig. 74 Pantalla juego



Fig. 75 Pantalla Game Over

Pulsera biométrica Open Source para la monitorización del usuario



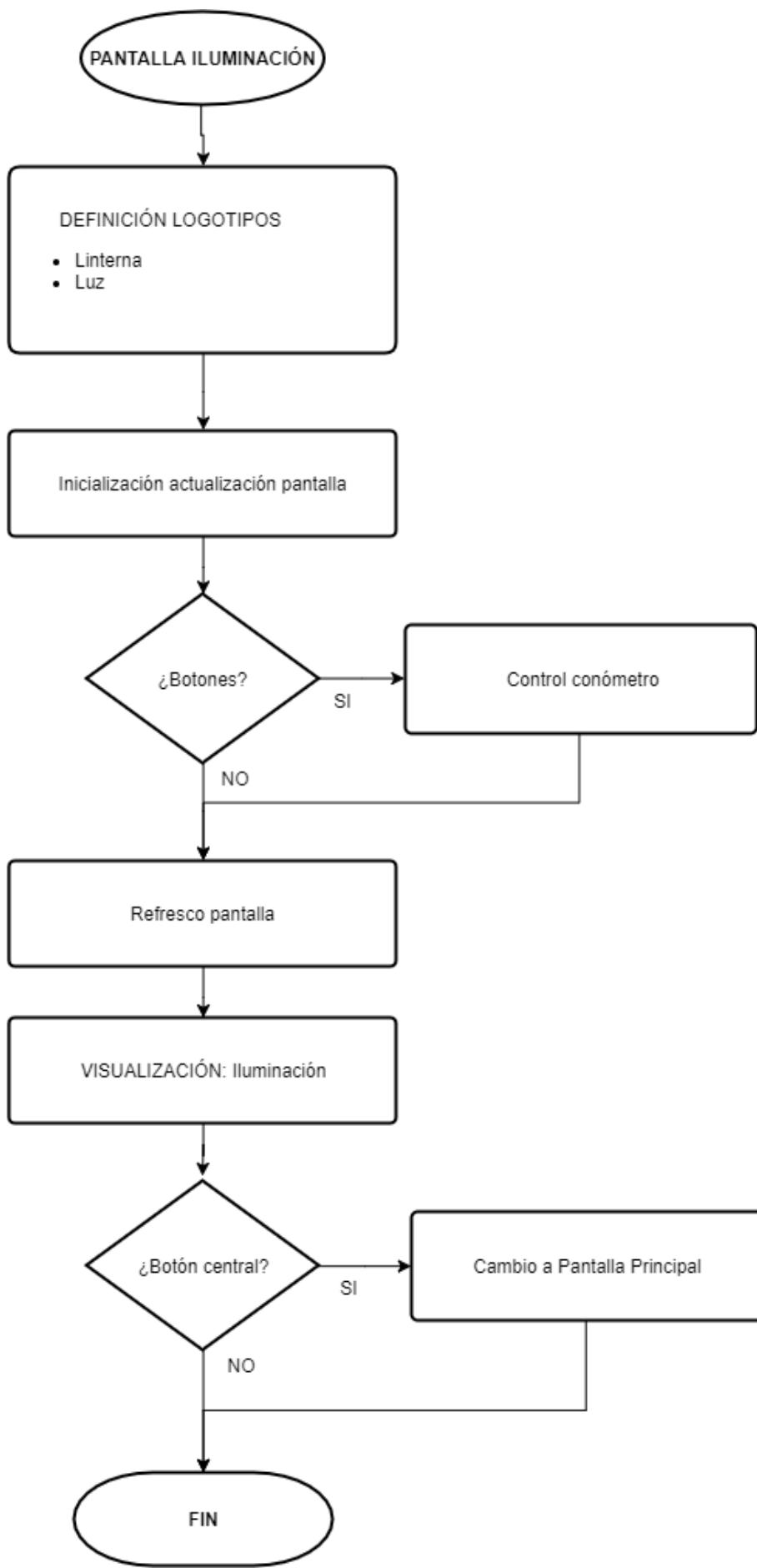
PANTALLA 9. ILUMINACIÓN – pantallalluminación()



Esta pantalla puedes seleccionar encender la luz, apagarla o que se encienda de forma automática dependiendo del sensor. Además te muestra el valor de luxes que hay en tiempo real.

Para salir de esta pantalla deberás pulsar el botón de selección o esperar que termine el tiempo de Standby.

Fig. 76 Pantalla iluminación



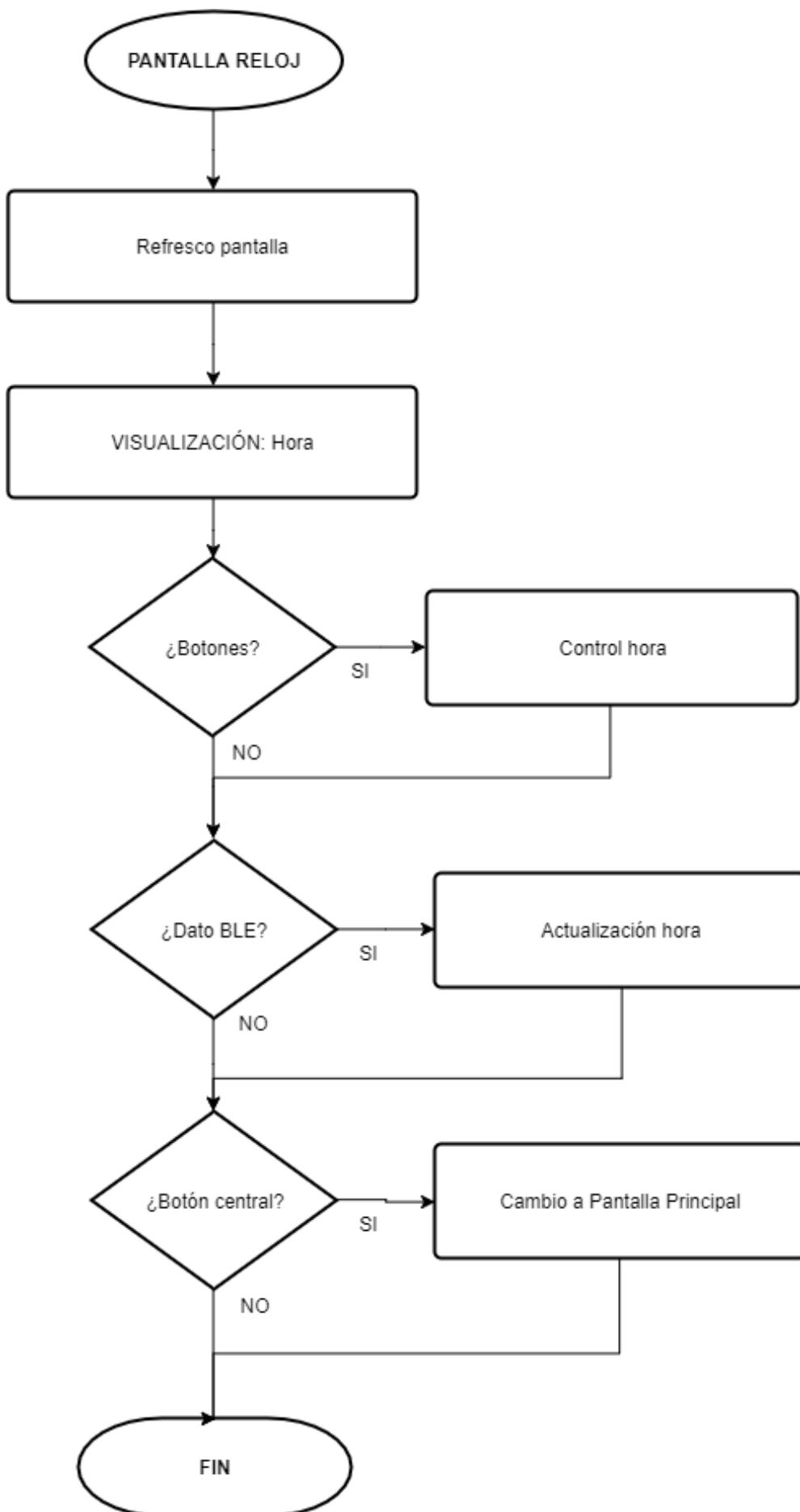
PANTALLA 10. RELOJ - pantallaReloj()



Esta pantalla muestra un reloj que puedes poner en hora con el botón de arriba pones la hora y con el de abajo pones los minutos. La idea es que el móvil sea el que lo actualice.

Para salir de esta pantalla deberás pulsar el botón de selección o esperar que termine el tiempo de Standby.

Fig. 77 Pantalla reloj



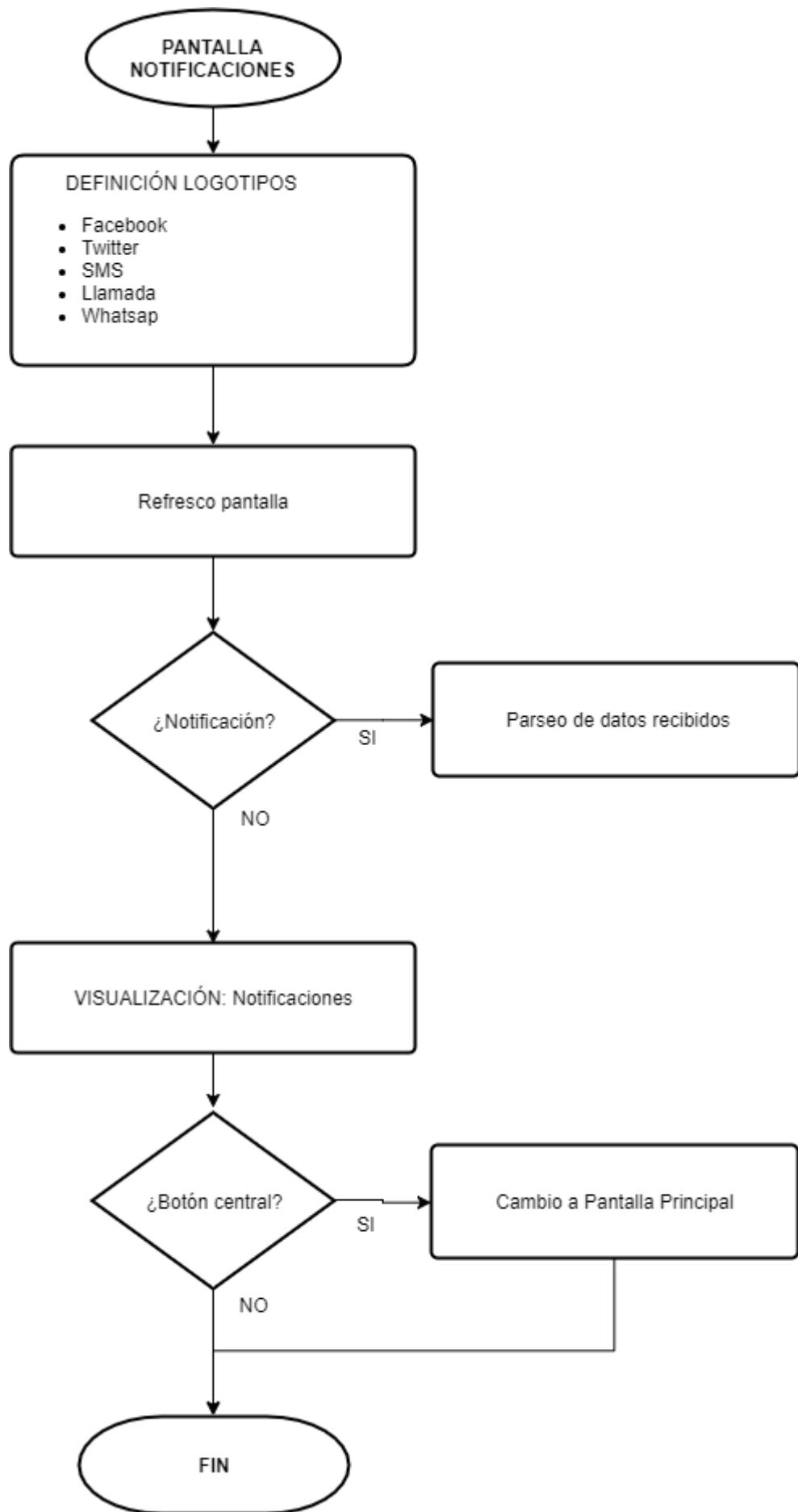
PANTALLA 11. NOTIFICACIONES - pantallaNotificaciones()



Esta pantalla muestra distintas notificaciones que pueden ser actualizadas con el móvil. Estas son: Facebook, Twitter, WhatsApp, mensajes y llamadas.

Para salir de esta pantalla deberás pulsar el botón de selección o esperar que termine el tiempo de Standby.

Fig. 78 Pantalla notificaciones



5.3.3. Librerías

El SDK de Arduino permite importar librerías externas específicas para ciertas acciones de comunicaciones, o manejo de ciertos sensores comerciales.

Para este proyecto se han empleado las siguientes librerías basadas en el SDK de Arduino:

- **PinChangeInterrupt.h:** Esta librería permite utilizar un pin digital del micro como interrupción. Se utiliza para poner el pin de carga de batería como interrupción. De esta forma cuando la batería se ponga en el cargador, se mostrará en pantalla.
 - attachPCINT(digitalPinToPCINT(CHGPIN), despertarMicro, FALLING): activa la interrupción al ponerse a cargar, es decir CHGPIN igual a cero.
- **SparkFunBME280.h:** Esta librería permite la comunicación entre el sensor de ambiente, sensor digital por I²C y el micro.
 - sensorAmbiente.begin(): Inicializa el protocolo de lectura de tramas digitales.
 - sensorAmbiente.readTempC(): Esta función hace transparente al desarrollador el formato de tramas de comunicación y vuelca directamente la temperatura en grados Celsius.
 - sensorAmbiente.readFloatPressure(): Esta función hace transparente al desarrollador el formato de tramas de comunicación y vuelca directamente la presión en Pascales.
 - sensorAmbiente.readFloatAltitudeMeters(): Esta función hace transparente al desarrollador el formato de tramas de comunicación y vuelca directamente la altitud en metros.
 - sensorAmbiente.readFloatHumidity(): Esta función hace transparente al desarrollador el formato de tramas de comunicación y vuelca directamente la humedad en % de humedad relativa.

- **SparkFun_ADXL345.h:** Esta librería permite la comunicación entre el acelerómetro, sensor digital por I²C, y el micro. Dicha librería se utiliza dentro de la librería pedometer.h para el cálculo de pasos.
- **Pedometer.h:** Esta librería realiza la cuenta de los pasos y determina cuando se ha hecho un paso cogiendo los valores de x, y y z del sensor con la función: adxl.readAccel(x, y, z) de la librería de Sparkfun.
 - acelerometro.init(): Inicializa el protocolo de lectura de tramas digitales llamando a la librería Sparkfun_ADXL345.
 - acelerometro.stepCalc(): Esta función realiza el cálculo para determinar cuando se ha dado un paso y aumenta la variable stepCount cuando esto ocurre.
- **SparkFunTSL2561.h:** Esta librería permite la comunicación entre el sensor digital por I²C y el micro.
 - sensorLuz.begin(): Inicializa el protocolo de lectura de tramas digitales.
 - sensorLuz.getData(data0, data1): Esta función devuelve los valores en crudo data 0 y data 1 según lo que capta el sensor.
 - sensorLuz.getLux(gananciaLuz, ms, data0, data1, lux): Esta función convierte estos valores en raw en luxes y devuelve 1 si se ha podido hacer y 0 si ha habido un error. Si el sensor se satura lux será igual a cero.
- **MAX30105.h:** Esta librería permite la comunicación entre el sensor digital por I²C y el micro.
 - sensorBiometrico.begin(Wire, I2C_SPEED_FAST): Inicializa el protocolo de lectura de tramas digitales.
 - sensorBiometrico.setup(brilloLed, mediaMuestreo, modoLed, ratioMuestreo, anchoPulso, rangoAdc): Configuración de los parámetros del sensor.
 - sensorBiometrico.check(): Esta función comprueba que hay nuevos datos de lectura.

- sensorBiometrico.getRed(): Esta función lee los nuevos datos en relación al led rojo.
 - sensorBiometrico.getIR(): Esta función lee los nuevos datos en relación al led IR.
- **Adafruit_GFX.h:** Esta librería permite realizar gráficos en la pantalla.
- pantallaBioBand.fillRect(posX, posY, 20, 8, col): Esta función permite dibujar una línea recta.
 - pantallaBioBand.setCursor(10, 10): Esta función define el punto donde empezar a dibujar en x-y.
 - pantallaBioBand.print("Dist: "): Esta función imprime en pantalla lo que le dices dentro, en este caso "Dist:".
 - pantallaBioBand.setTextSize(1): Esta función define el tamaño de la letra.
 - pantallaBioBand.drawBitmap(10, 65, acc_img, acc_width, acc_height, BLACK): Esta función dibuja una imagen en formato Bitmap.
 - pantallaBioBand.setRotation(0): Define la orientación de la pantalla
- **Adafruit_SharpMem.h:** Esta librería permite la comunicación entre la pantalla por SPI y el micro.
- pantallaBioBand.begin(): Inicializa el protocolo de escritura de tramas digitales.
 - pantallaBioBand.clearDisplay (): Limpia la pantalla de lo que hubiera dibujado anteriormente.
- **LowPower.h:** Esta librería permite bajar el consumo del micro.
- LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF): Envía el micro a modo Standby para ahorrar consumo de energía.

- **Button.h:** Esta librería se utiliza para la lectura de los pulsadores.
 - botonCentral.wasPressed(): Esta función devuelve verdadero o falso dependiendo de si se ha presionado el pulsador.
 - botonCentral(MBUT, true, true, 20): Esta función define el pulsador.
 - botonCentral.read(): Esta función lee el valor del pulsador y devuelve 1 si se ha presionado y 0 si se ha soltado.
- **Adafruit_BLE.h y Adafruit_BluefruitLE_UART:** Librerías de Adafruit para el manejo del módulo bluetooth mdbt40 (nrf518222) a través de UART.
 - ble.begin(MODO_VERBOSE): Inicializa el módulo bluetooth.
 - ble.isConnected(): Comprueba que el módulo se ha conectado a otro bluetooth.
 - Ble.print(): Envía un dato a otro bluetooth
 - ble.waitForOK(): Espera a que el dato se haya enviado correctamente.

6. Diseño mecánico

6.1. Introducción

La última parte del proyecto ha sido el diseño de una carcasa para encajar el dispositivo en impresión 3D. Se realizaron distintos modelos de pulsera según la versión de PCB tanto en PLA como Filaflex hasta optar por la que mejor encajaba con el dispositivo y sus especificaciones.

6.2. Tinkercad

Para el diseño de la carcasa se ha utilizado el programa Tinkercad [17]. Este programa fue la primera plataforma de diseño 3D basado en navegador para las masas. Fue fundada en 2011 por Kai Backman y Mikko Mononen. Se convirtió en parte de Autodesk en junio del año 2013 y se unió a 123D de productos para dar apoyo a todas las personas de distintas clases sociales con la finalidad de diseñar y hacer las cosas que ellas imaginan.

Para utilizar Tinkercad sólo es necesario crear una cuenta gratuita. Es un programa muy sencillo de utilizar y te permite almacenar tu trabajo en línea. Cuenta con diferentes modelos de ejemplo para orientar a todas aquellas personas principiantes que se introducen en el mundo del diseño de modelos 3D y por otro lado, los usuarios son libres de compartir los scripts para que otras personas puedan utilizarlos o bien hacer variaciones del patrón realizado.

En la imagen de abajo puede verse una imagen del despiece 3D del dispositivo final exportada desde dicho programa. De esta forma podemos ver de una manera más clara las distintas partes que van dentro de la carcasa:

De arriba abajo podemos encontrar:

1. Tapa superior (Material PLA)
2. Pantalla Sharp Memory
3. Pieza soporte PCB y pantalla (Material PLA)
4. PCB principal (se engancha a través de sus agujeros en la pieza 3)
5. Batería
- 6.1. Bobina de inducción
- 6.2. Placas de sensores: en el agujero que deja libre la bobina (Sensores T^a y HR)
7. Carcasa principal y botones (dentro van desde 2 hasta 6)

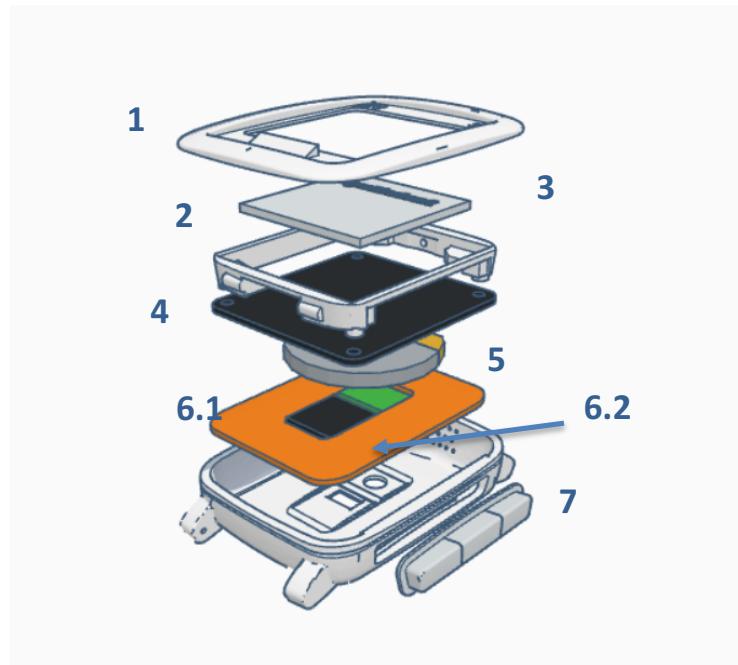


Fig. 79 Despiece 3D dispositivo

6.3. Modelos

Gracias a la web Thingiverse [18] existen gran cantidad de archivos Open Source para imprimir en 3D. En los primeros modelos de PCB, se realizaron distintas pruebas para realizar la carcasa partiendo de estos diseños Open Source. Se realizaron pruebas tanto en PLA como en Filaflex.

6.3.1. PLA

El PLA es un polímero muy usado en impresión 3D por su facilidad de uso. Su temperatura de impresión es de unos 200 grados, es un material frágil a la vez que duro y se fabrica a partir del maíz por lo que no es tóxico.

Se realizaron distintos modelos de pulseras en PLA, basadas en los modelos metálicos de los relojes:



Fig. 80 Diseño pulsera en PLA



Fig. 81 Diseño correa en PLA

6.3.2. FILAFLEX

El Filaflex es un elastómero termoplástico, no es tan fácil de usar ya que necesita de impresoras o boquillas especiales para poder imprimir con este material. Su temperatura de impresión es mayor que la del PLA y tampoco es un material tóxico.

Aunque la pulsera sea de Filaflex, la carcasa de la electrónica debe ser resistente a golpes por lo que tiene que ser en PLA.



Fig. 82 Correas en Filaflex

Otro diseño que se planteó es que la electrónica fuera dentro de una carcasa que pudieras llevarla en el bolsillo por ejemplo y cuando quisieras, a través de una funda, pudieras ponértela como pulsera.



Fig. 83 Pulsera en Filaflex

6.3.3. Diseño final

Finalmente se ha diseñado la carcasa para la PCB final en 4 piezas que pueden verse en la imagen de abajo.



Fig. 84 Partes carcasa final impresas en PLA

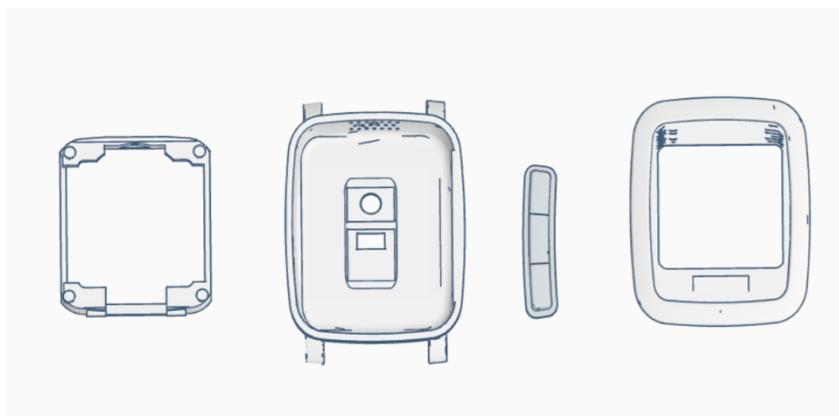


Fig. 85 Partes carcasa final en Tinkercad

La primera sirve de sujeción para la pantalla con los agujeros de la PCB, la segunda es donde se introduce la bobina, la batería, la PCB, pueden verse también los agujeros para el sensor de pulso y el de temperatura y en un lateral se encuentran los pulsadores que es la siguiente pieza. También dispone de agujeros en uno de los laterales para que los sensores de ambiente y luz estén en contacto con el exterior. Y por último la tapa de la carcasa con un agujero del tamaño de la pantalla y una ranura por donde sale luz en el caso de encenderla.

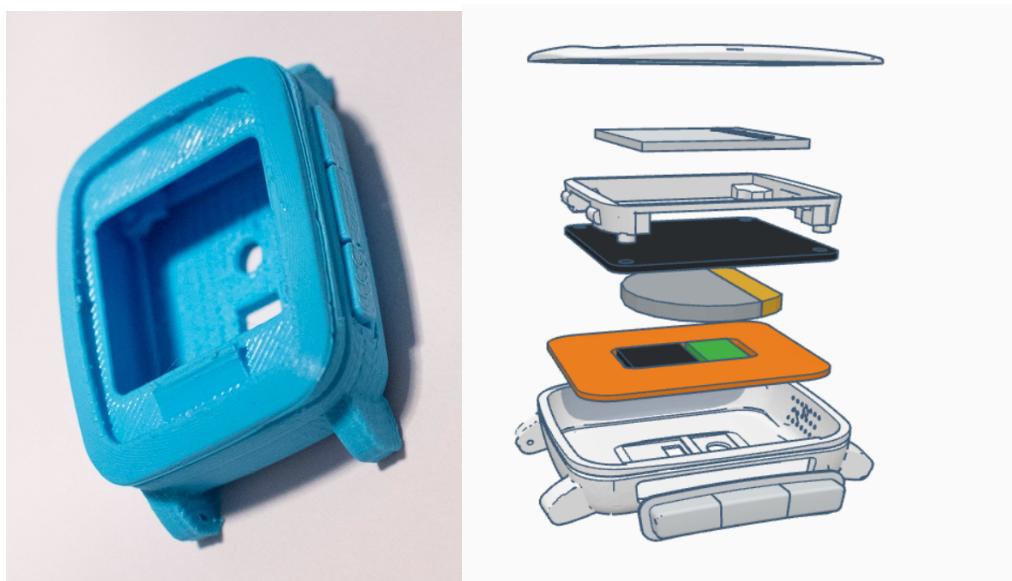


Fig. 86 Carcasa final y en 3D

La correa ha sido diseñada para correas estándar de 22mm. En este caso, se ha escogido la correa vista en los diseños con Filaflex.



Fig. 87 Correa final en Filaflex. Estándar 22 mm

7. Montaje

En este apartado pueden verse imágenes paso a paso del montaje de la pulsera.

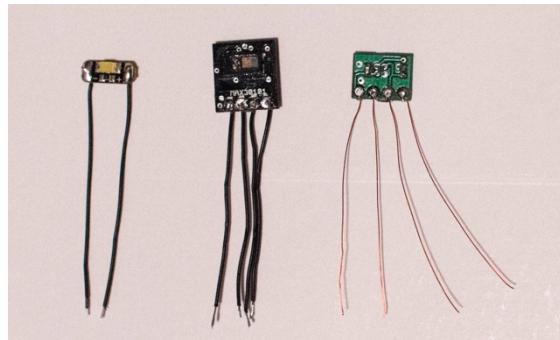


Fig. 88 Sensores y luz externos a PCB principal

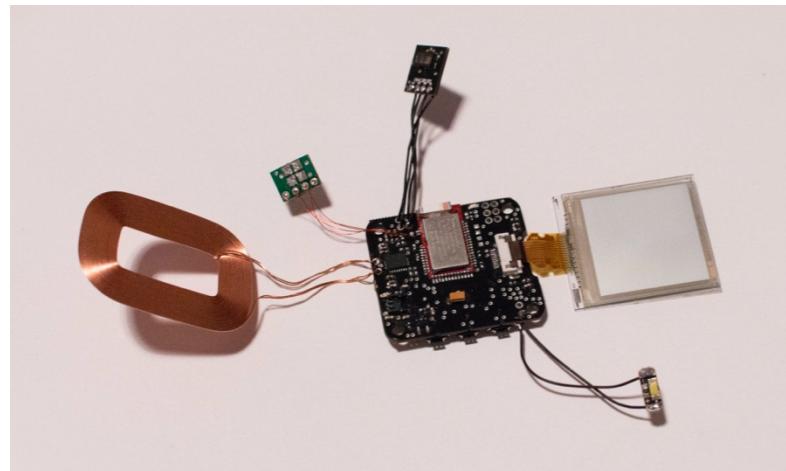


Fig. 89 PCB junto el resto de elementos

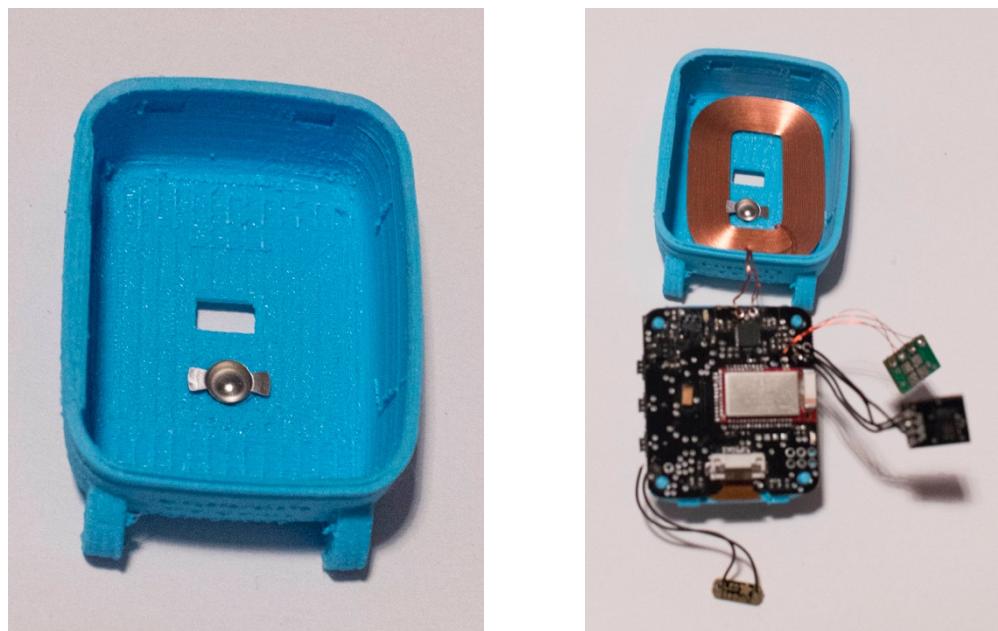


Fig. 90 Montaje metal sensor de temperatura y bobina

Pulsera biométrica Open Source para la monitorización del usuario

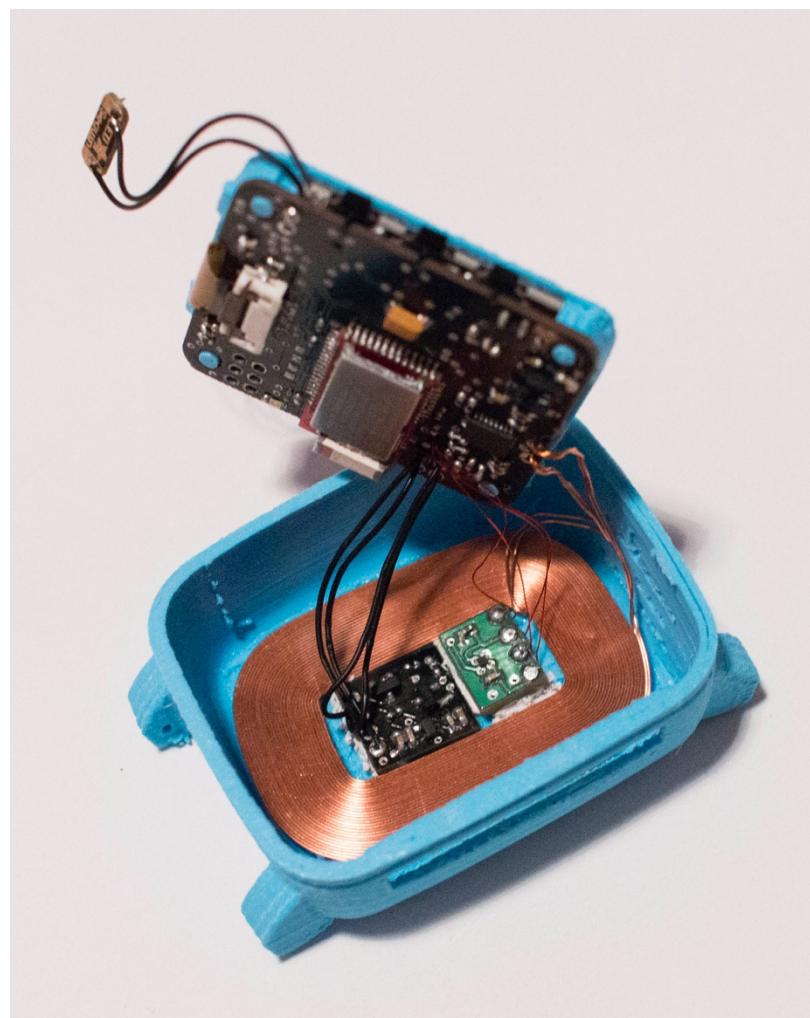


Fig. 91 Montaje sensores



Fig. 92 Montaje luz, batería y PCB principal



Fig. 93 Open Bioband

8. Pruebas realizadas

8.1. Introducción

El diseño final, ha sido testeado en distintos aspectos tanto en funcionalidades básicas, como en cuanto a experiencia de usuario.

Estas segundas no son planteadas ya que no son objetivas completamente, pero han sido tenidas en cuenta para el diseño de producto y sus distintas iteraciones y soluciones finales aplicadas.

Lo mismo ha ocurrido con la navegación final implementada básica en todo lo que atañe a UI (user interface) y UX (user experience).

8.2. Pruebas generales

Para comprobar el correcto funcionamiento de todas y cada una de las etapas del prototipo principal se han ido montando cada etapa de manera independiente como se ha comentado previamente y probándola. Para dar por válida cada una de ellas se han realizado pruebas puntuales sirviéndose de una placa FTDI conectada al conector preparado para esta misión.

Esto ha permitido obtener datos a través de comunicación serie y comprobar que la respuesta de los distintos circuitos o las mediciones tomadas sobre ellos era la correcta.

Se ha utilizado el serial plotter de Arduino para representar los valores en tiempo real de los sensores. Este proceso se ha realizado con varios de los sensores, especialmente con aquellos que permiten representación mediante gráficas como el de pulso.

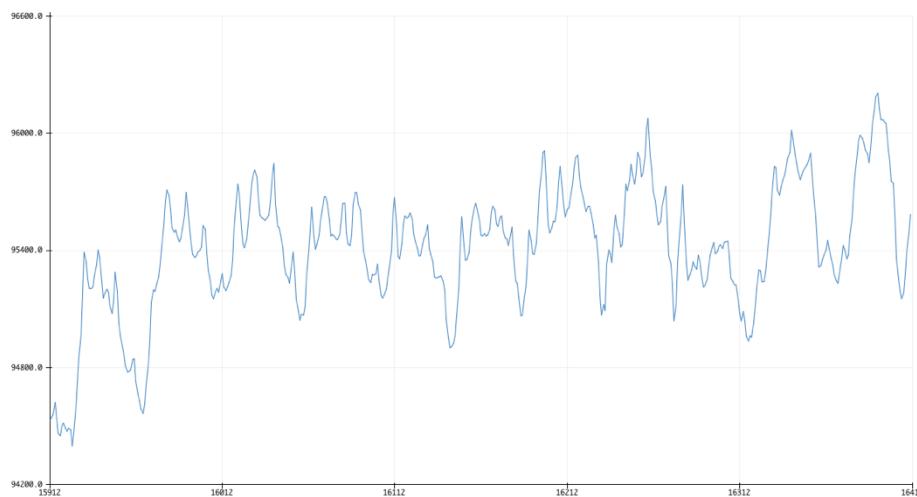


Fig. 94 Gráfica sensor de pulso

Tras el montaje de todo el dispositivo se ha probado el sistema completo revisando que no hubiera ningún fallo al funcionar a la vez.

8.3. Calibración de sensores

Todos los sensores han sido comparados con dispositivos comerciales destinados a la monitorización de parámetros comunes.

Esto ha permitido de manera sencilla y superficial, validar el funcionamiento de los sensores, y en caso de observar alguna desviación, aplicar correcciones por software.

Deberían ser repetidas en una siguiente fase, dichas pruebas, a largo plazo, con el fin de validar el funcionamiento completo del producto.

Esto sería planteado, sobre todo de cara a una fase de comercialización, aunque hay que decir que las propias certificaciones FCC y CE necesarias para su comercialización, supondrían unos estrictos controles y límites para las pruebas.

8.4. Pruebas de consumo

Se ha realizado un estudio teórico tanto de las posibilidades del microcontrolador, como del funcionamiento del dispositivo sobre el consumo de la batería.

Podemos observar en la tabla de abajo lo siguiente:

- Si la pulsera estuviera siempre en funcionamiento, una batería de 210mAh duraría 1 día y 5 horas.
- Si la pulsera estuviera en modo reposo, una batería de 210mAh duraría 3 meses y 15 días.
- Si la pulsera además de estar en modo normal, está enviando datos con el bluetooth, una batería de 210mAh duraría prácticamente un día.

En la práctica, si utilizas la pulsera como un usuario normal, en el que siempre está en reposo menos los momentos en los que quieras realizar alguna medición la batería dura entre 7-10 días y si la pulsera siempre está midiendo estará durará unos 2 o 3 días ya que nunca están todos sensores midiendo a la vez.

MODO ENCENDIDO - FUNCIONAMIENTO NORMAL						
Componente	Nombre	Consumo Máximo (mA)	Factor de utilización	Tiempo consumo	Consumo energético	BATERÍA mAh
Accelerómetro	ADXL345	0,04	1	3600	0,04	210
LED Iluminación	0402 LED	20	0,2	600	0,6	330
Sensor iluminación	TSL2561	0,6	1	3600	0,6	
Sensor ambiente	BME280	0,714	1	3600	0,714	
Sensor biométrico	MAX30101	1,1	1	3600	1,1	
Microcontrolador	Atmega 1284p	4	1	3600	4	
BLE	MDBT40	0,0012	1	3600	0,0012	
Sensor temperatura corporal	LMT70	0,012	1	3600	0,012	
Pantalla	Sharp Memory	0,004	1	3600	0,004	
				Total (mAh)	7,137866667	29,4205516
						46,23230097

MODO BAJO CONSUMO - FUNCIONAMIENTO REPOSO						
Componente	Nombre	Consumo Máximo (mA)	Factor de utilización	Tiempo consumo	Consumo energético	BATERÍA mAh
Accelerómetro	ADXL345	0,0001	1	3600	0,0001	210
LED Iluminación	0402 LED	20	0	3600	0	330
Sensor iluminación	TSL2561	0,015	1	3600	0,015	
Sensor ambiente	BME280	0,005	1	3600	0,005	
Sensor biométrico	MAX30101	0,0025	1	3600	0,0025	
Microcontrolador	Atmega 1284p	0,0043	1	3600	0,0043	
BLE	MDBT40	0,0012	1	3600	0,0012	
Sensor temperatura corporal	LMT70	0,0092	1	3600	0,0092	
Pantalla	Sharp Memory	0,004	1	3600	0,004	
				Total (mAh)	0,0413	5084,745763
						7950,31477

MODO CONECTADO - FUNCIONAMIENTO BLUETOOTH						
Componente	Nombre	Consumo Máximo (mA)	Factor de utilización	Tiempo consumo	Consumo energético	BATERÍA mAh
Accelerómetro	ADXL345	0,04	1	3600	0,04	210
LED Iluminación	0402 LED	20	0,2	600	0,6	330
Sensor iluminación	TSL2561	0,6	1	3600	0,6	
Sensor ambiente	BME280	0,714	1	3600	0,714	
Sensor biométrico	MAX30101	1,1	1	3600	1,1	
Microcontrolador	Atmega 1284p	4	1	3600	4	
BLE	MDBT40	10	1	600	1,666666667	
Sensor temperatura corporal	LMT70	0,012	1	3600	0,012	
Pantalla	Sharp Memory	0,004	1	3600	0,004	
				Total (mAh)	8,803333333	23,85460053
						37,48580083

Tabla 38. Consumos teóricos

8.5. Pruebas de compatibilidad

La telemedicina es el uso de las telecomunicaciones y las tecnologías de la información para obtener apoyo médico a distancia. Esto elimina muchas barreras, por eso un punto principal de Open Bioband es su conectividad. Se ha implementado un módulo Bluetooth para que a futuro puedan compartirse los datos en la nube y realizar un diagnóstico en tiempo real.

Se han realizado pruebas de señal para comprobar el correcto funcionamiento de la etapa, y evitar problemas de desconexión. El datasheet indica que debería funcionar correctamente hasta 80 metros en áreas abiertas.

Open Bioband	Carcasa inferior / Tapa / Batería
DISTANCIA	SEÑAL (dBm)
1 cm	-52
25 cm	-54
50 cm	-58
75 cm	-62
100 cm	-64
150 cm	-68
200 cm	-76

Tabla 39. Pruebas señal con carcasa

Open Bioband	Solo PCB / Alimentación Batería
DISTANCIA	SEÑAL (-dBm)
1 cm	-45
25 cm	-48
50 cm	-52
75 cm	-60
100 cm	-62
150 cm	-66
200 cm	-70

Tabla 40. Pruebas señal sin carcasa

9. Conclusiones

Se ha conseguido el objetivo propuesto en el proyecto, consiguiendo una pulsera capaz de monitorizar al usuario y su ambiente en tiempo real.

Se dispone de un sistema que incorpora la electrónica y el software utilizado para su control y programación. También se han realizado las pruebas pertinentes para verificar su correcto funcionamiento.

La consecución del objetivo se hizo en orden, lo que ha permitido identificar una serie de hitos parciales:

- Se completó un exhaustivo proceso de investigación inicial, analizando todo lo relacionado con los wearables enfocados a pulseras de monitorización, reuniendo información general del tema y comparando sus características.
- Se realizó un estudio básico de la tecnología con la que se iba a trabajar: Arduino y los módulos Bluetooth.
- Se realizó un estudio de distintos componentes que existen en el mercado: sensores, Bluetooth, pantalla, ... realizando pruebas con ellos hasta elegir los que mejor se adaptan al sistema.
- Para el diseño final del módulo, se realizó una serie de prototipos, con un firmware básico para ver que todos los componentes juntos funcionaban correctamente.
- Se diseñó un prototipo final funcional, con su correspondiente firmware definitivo. Se añadieron algunos extras como reloj, cronómetro o juegos.
- Se verificó que todo funcionaba en la placa definitiva, cumplía las expectativas y objetivos preestablecidos.
- Se diseñó una carcasa impresa en 3D acorde con los componentes que deben encajar en su interior.

Lo ideal, es que al ser una pulsera Open Source, ésta sirviera para futuros desarrollos de otras personas y un desarrollador crearía una aplicación donde pudieran verse todos los datos, no solo en tiempo real, sino almacenados por fechas.

Por ello, el proyecto fue uno de los elegidos para participar en la convocatoria Cesar del OpenArt de Zaragoza, anteriormente nombrado. El objetivo de esta convocatoria es el de impulsar la investigación en ciencia ciudadana, acercar la ciencia, la creatividad tecnológica y el arte con nuevos medios a la ciudadanía, además de favorecer el conocimiento colaborativo y consolidar Etopia como centro de producción de proyectos multidisciplinares.

Y toda la información referente al dispositivo podrá verse en mi GitHub [19] y en Hackaday [20]. Actualmente hay parte en un drive público [21].

9.1. Conclusiones personales

Elegí este proyecto porque no sólo quería hacer un proyecto sino porque quería aprender haciéndolo. Me encanta la electrónica porque puedes pensar algo y hacerlo realidad, me encanta el mundo maker y Open Source porque no sólo le das vida a algo sino que además, lo compartes y ya no solo aprendes tú sino que aprendéis todos y la última cosa que me encanta, son los wearables, poder llevar la electrónica a la moda, usarla de manera creativa.

Así que escogí una pulsera biomédica Open Source, aprender cómo se hace un “producto” desde cero, cómo soldar componentes SMD, hacer PCBs en Eagle y diseñar y elegir cada componente y detalle para que encaje por si solo.

Y, aunque no tiene el acabado de una pulsera que puedas comprar en una tienda, me quedo con todo lo aprendido en su desarrollo y que, puedo personalizarla o adaptarla para otros proyectos a futuro. En esta línea, me gustaría seguir investigando en la pulsera con el Cortex M0 y algunas mejoras de diseño final que he ido viendo conforme su desarrollo.

10. Bibliografía

- [1] <https://www.pwc.com/us/en/technology/publications/assets/pwc-wearable-tech-design-oct-8th.pdf>
- [2] <https://www.pwc.es/es/publicaciones/digital/wearable-life-dispositivos-conectados.pdf>
- [3] <https://www.arduino.cc/>
- [4] <https://developer.apple.com/healthkit/>
- [5] <https://www.google.com/fit/>
- [6] <http://saatchi.com/en-us/news/feel-the-reel/>
- [7] <https://health.nokia.com/es/en/>
- [8] <https://developer.pebble.com/open-source/>
- [9] <https://www.simband.io/>
- [10] <http://www.hexiwear.com/>
- [11] <https://www.autodesk.com/products/eagle/overview>
- [12] <https://processing.org/>
- [13] <http://cesaretopia.com/>
- [14] <http://www.lpkf.es/productos/creacion-rapida-prototipos-pcb/plotter-fresadora/protomat-e-33.htm>
- [15] <http://www.ti.com/lit/ug/tiduay7/tiduay7.pdf>
- [16] <https://www.adafruit.com/>
- [17] <https://www.tinkercad.com/>
- [18] <https://www.thingiverse.com/>
- [19] <https://github.com/estherbm/Open-BioBand>
- [20] <https://hackaday.io/>
- [21] https://drive.google.com/drive/folders/0B6BMdabiWqA_ZkdGTE5uWnZoN0O?usp=sharing

ANEXO 1. Cronograma

La planificación del proyecto se muestra en la siguiente página. Dicha planificación, se ha dividido en 5 grandes hitos generales, que han quedado enmarcados de la siguiente manera, en función de la duración estimada de los mismos.

HITO	DURACIÓN
Especificación de requisitos definida	6 semanas
Diseño del sistema completo finalizado	10 semanas
Electrónica del sistema fabricada y disponible	14 semanas
Firmware de la electrónica y aplicación de usuario finalizados	6 semanas
Producto final terminado y validado	12 semanas

Tabla 41. Hitos generales

Debajo puede verse una tabla más detallada de cada uno de los hitos.

Tarea	Mes 1	Mes 2	Mes 3	Mes 4	Mes 5	Mes 6	Mes 7	Mes 8	Mes 9	Mes 10	Mes 11	Mes 12
1. Análisis de mercado y especificación de requisitos												
1.1. Análisis de mercado wearables												
1.2. Análisis de mercado componentes												
1.3. Versión 0												
1.4. Definición de objetivos y especificación de requisitos												
2. Diseño del sistema												
2.1. Diseño de arquitectura del sistema												
2.2. Estudio y pruebas componentes comerciales												
2.3. Diseño preliminar HW con componentes comerciales												
2.4. Estudio y pruebas comunicaciones												
2.5. Diseño estructura SW												
3. Desarrollo HW v1												
3.1. Desarrollo de los esquemas electrónicos												
3.2. Compra y recepción de componentes HW												
3.3. Fabricación de los elementos del prototipo electrónico												
3.4. Puesta a punto												
4. Fase pruebas												
4.1. Desarrollo firmware básico												
4.2. Pruebas dispositivo												
4.3. Conclusiones												
5. Desarrollo HW v2												
5.1. Desarrollo de los esquemas electrónicos												
5.2. Compra y recepción de componentes HW												
5.3. Fabricación de los elementos del prototipo electrónico												
5.4. Puesta a punto												
6. Desarrollo SW												
6.1. Desarrollo del firmware de la electrónica												
6.2. Desarrollo de la interfaz de usuario												
7. Diseño												
7.1. Diseño carcasa y correa												
7.2. Impresión 3D												
7.3. Mejoras carcasa												
8. Fase pruebas Generales												
8.1. Pruebas sistema completo												
8.2. Calibración sensores												
8.3. Pruebas autonomía												
8.4. Pruebas carga												

Tabla 42. Cronograma

ANEXO 2. Normativa FDA

Dado el incremento de tecnologías wearables relacionadas con la salud y el bienestar, la FDA (agencia estadounidense del medicamento) ha publicado una normativa no jurídicamente vinculante, pero que sí establece unos criterios de legislación.

CDHR (Center for Devices and Radiological Health's) define los productos generales de bienestar como productos que cumplen los siguientes factores:

- (1) Están destinados exclusivamente para el bienestar general, como se define en esta guía.
- (2) Presenta un riesgo muy bajo para la seguridad de los usuarios. Estos podrían ser equipos de ejercicio, grabaciones de audio, videojuegos, software y otros productos que son comunes, aunque no exclusivamente, disponibles en establecimientos minoristas (incluyendo los minoristas y distribuidores que ofrecen software para ser descargado directamente en línea).

La primera categoría de bienestar general implica afirmaciones sobre el mantenimiento o la mejora general de las condiciones y funciones asociadas con un estado general de la salud que no hace ninguna referencia a enfermedades o condiciones. Para los propósitos de esta orientación, esta primera categoría se refiere a:

- Control de peso
- Aptitud física, incluidos los productos destinados a uso recreativo
- La relajación o el manejo del estrés
- Agudeza mental
- Autoestima
- Gestión de sueño
- La función sexual

Los siguientes son ejemplos de esta categoría:

- Reclamos para promover o mantener un peso saludable, fomentar la alimentación saludable, o ayudar con objetivos de pérdida de peso;
- Reclamos para promover la relajación o manejar el estrés cuando no hay ninguna referencia a la ansiedad, trastornos u otra referencia a una enfermedad o afección;
- Reclamos para aumentar o mejorar el flujo de qi;
- Reclamos para mejorar la agudeza mental, la instrucción siguiente, la concentración, la solución de problemas multitarea, gestión de recursos, la toma de decisiones, la lógica, el patrón, el reconocimiento o la coordinación ojo-mano;
- Reclamos para promover la salud física, como medir la capacidad aeróbica, mejorar la condición física, desarrollar o mejorar resistencia, fuerza o coordinación, o mejorar la energía;
- Reclamos para promover la gestión del sueño, tales como el seguimiento de las tendencias del sueño;
- Reclamos para promover o impulsar la autoestima
- Reclamos que se ocupan de una estructura específica del cuerpo o función, como para aumentar o mejorar el tamaño de los músculos o el tono corporal, tono o firmeza al cuerpo o músculo, mejorar la función cardíaca , o aumentar o mejorar el rendimiento sexual;
- Reclamos para mejorar la movilidad general o para ayudar a las personas que son problemas de movilidad o que tienen una movilidad limitada en una actividad recreativa; y
- Reclamos para mejorar la participación de una persona en actividades recreativas por el seguimiento de participar en este tipo de actividades, como para monitorear la frecuencia cardíaca o la frecuencia del monitor o el impacto de las colisiones.

Los siguientes son ejemplos de los reclamos que no entran en esta categoría de bienestar general:

- Declararse que un producto va a tratar o diagnosticar la obesidad;
- Declararse que un producto va a tratar un trastorno de la alimentación, como la anorexia;
- Declararse que un producto ayuda a tratar la ansiedad;
- Declararse que un juego de ordenador diagnosticara o tratara el autismo;
- Declararse que un producto va a tratar la atrofia muscular o disfunción eréctil;
- Un reclamo para restaurar una estructura o función deteriorada debido a una enfermedad, por ejemplo, la afirmación de que una prótesis permite a amputados jugar al baloncesto

La segunda categoría de usos de bienestar generales se compone de dos subcategorías:

- 1) Intención de promover, controlar y / o fomentar elección (es), que, como parte de un estilo de vida saludable, puede ayudar a reducir el riesgo de ciertas enfermedades crónicas o condiciones.
- 2) Intención de promover, controlar y / o fomentar elección (es) que, como parte de una estilo de vida saludable, puede ayudar a vivir bien con ciertas enfermedades o condiciones crónicas.

Los siguientes son ejemplos de esta categoría de reclamaciones de bienestar relacionados con la enfermedad:

- **Producto X:** promueve la actividad física que, como parte de un estilo de vida saludable, puede ayudar a reducir el riesgo de presión arterial alta.
- **Producto Y:** software que monitoriza tu consumo de calorías y te ayuda a administrar una alimentación saludable, lo planifica para mantener un peso saludable y una dieta equilibrada. Este peso saludable y la dieta equilibrada te puede ayudar a vivir bien con la presión arterial alta y la diabetes tipo 2.
- **Producto Z:** rastrea los patrones de sueño de actividad y promueve hábitos saludables de sueño, que, como parte de un estilo de vida saludable, puede ayudar a reducir el riesgo de desarrollar diabetes tipo 2.

La política de bienestar general del CDRH no se extiende a los dispositivos que presentan riesgos inherentes a la seguridad del usuario.

Si un dispositivo tiene efectos de bajo riesgo se determina por si el producto:

- Es invasivo
- Implica una intervención o tecnología que puedan suponer un riesgo para la seguridad de un usuario, como los riesgos de exposición a la radiación láser, o implantes
- Plantea nuevas cuestiones de usabilidad
- Plantea cuestiones de biocompatibilidad

Si la respuesta a cualquiera de estas preguntas es sí, el dispositivo no es determinado como producto de bienestar general de bajo riesgo y no está cubierto por esta guía.

Los siguientes son ejemplos de productos que presentan riesgos inherentes a la seguridad del usuario y no serían considerados de "bajo riesgo":

- Lámpara solar promovida para fines de bronceado, debido a los riesgos para la seguridad de los usuarios de la radiación ultravioleta, incluyendo, sin limitación, un mayor riesgo de cáncer de piel.
- Implantes promovidos para mejorar la imagen de sí mismo o la función sexual, debido a los riesgos a los usuarios, tales como la ruptura o reacción adversa a los materiales de implante, y los riesgos asociados con el procedimiento de implantación.
- Un producto láser que pretende mejorar la confianza en la apariencia del usuario, el rejuvenecimiento de la piel. Aunque las afirmaciones de rejuvenecimiento de la piel y mejora la confianza en la apariencia del usuario son afirmaciones generales de bienestar, la tecnología láser pone en riesgo la piel, los ojos y pueden aparecer quemaduras.

Ejemplos de bajo riesgo:

Ejemplo ilustrativo 1:

Una aplicación móvil reproduce música para "calmar y relajar" a un individuo y ayuda a "manejar el estrés."

Estas afirmaciones se refieren sólo a la relajación o el manejo del estrés, no a cualquier enfermedad o estado de salud, y por lo tanto son las reclamaciones de bienestar general. Además, la tecnología para reproducir música no presenta riesgos inherentes a la seguridad del usuario. Por lo tanto, este producto cumple con ambos factores para un producto de la salud en general de bajo riesgo.

Ejemplo ilustrativo 2:

Una aplicación móvil que únicamente registra la actividad diaria de gasto y de entrenamiento cardiovascular para "dotarte de unas actividades para mejorar o mantener una buena salud cardiovascular".

Esta reclamación se refiere a un órgano específico sólo en el contexto de la salud general y no se refieren a una enfermedad o condición médica. Además, la vigilancia o grabación de actividades del ejercicio puede presentar riesgos (como la inexactitud), pero cuando se hace en la ausencia de enfermedad o condición médica, los riesgos para la seguridad del usuario son bajos.

Por lo tanto, este producto cumple con los dos factores para un producto de bienestar general de bajo riesgo.

Ejemplo ilustrativo 3:

A los dispositivos que registran el consumo de alimentos: "Monitorizar la actividad de la dieta para controlar el peso y alertar al usuario, profesional de la salud, o a un miembro de la familia de la actividad de la dieta poco saludable".

Esta reclamación se refiere a los hábitos dietéticos y el control de peso, y por lo tanto es un reclamo de bienestar. Además, la tecnología para la vigilancia o la grabación de datos del consumo de la comida supone un riesgo bajo para la seguridad del usuario. Por lo tanto, este producto cumple con ambos factores para un producto de bienestar general de bajo riesgo.

Ejemplo ilustrativo 4:

Un producto portátil que pretende controlar el pulso de los usuarios durante el ejercicio y el senderismo.

Esta afirmación se refiere únicamente al ejercicio y practicar senderismo y no se refiere a una enfermedad o condición médica. Por lo tanto, es una reivindicación de bienestar general. Además, la tecnología para el monitoreo supone un riesgo bajo para la seguridad del usuario. Por lo tanto, este producto cumple ambos factores para un producto de bienestar general de riesgo bajo.

Ejemplo ilustrativo 5:

Un producto diseñado para exfoliar la cara, las manos y los pies para hacer la piel más suave.

Esta reclamación se refiere a la autoestima y no se refiere a una enfermedad específica, y por lo tanto es un reclamo de bienestar general. Además, la tecnología para exfoliar la cara presenta un bajo riesgo para la seguridad del usuario ya que no penetra en el estrato córneo. Por lo tanto, este producto cumple con ambos factores de riesgo bajo para la salud del usuario.

Nota: Sin embargo, si el producto que exfolia la piel contiene uno o más ingredientes activos farmacéuticos que aplicados penetran el estrato córneo, el producto podría presentar riesgos para la seguridad del usuario debido a su naturaleza invasiva. Por lo tanto, el producto no sería un producto de bajo riesgo para la salud.

ANEXO 3. La tecnología de partida: Arduino

Arduino nació como un proyecto para acercar la electrónica y la programación a personas que no lo desconocen pero quiere incluirlo en sus proyectos, una herramienta sencilla de usar y aprender. Su propósito fue el de ser una placa de desarrollo de fácil programación a través de un lenguajes de muy alto nivel, en comparación con otras plataformas como PIC, que resultaba muy complicada para los iniciados y nuevos estudiantes. Pero Arduino ha transcendido más allá y hoy en día ya no sólo se usa para prototipado, sino que incluso es una placa sobre la que desarrollan pequeños proyectos para empresas y entidades o con la que enseñan electrónica y programación a niños en los colegios.

Arduino simplifica mucho tanto la tarea de creación de hardware, como la posterior tarea de desarrollo de software.

Leer un simple sensor, requiere de configuración de muchos registros del micro-controlador, que hacen de la experiencia de aprendizaje algo pesado para muchos usuarios. En el entorno de Arduino, todo se reduce mediante una única llamada a una función que nos devuelve el valor de la entrada analógica y sin necesidad de leer hojas y hojas del datasheet del micro-controlador.

Otra ventaja es que el diseño de la placa Arduino está disponible en Internet, con que cualquier persona puede construirla en su casa de forma independiente.

Todos los ficheros CAD y las librerías de programación y bootloaders están totalmente disponibles de forma gratuita.

Ventajas de Arduino

Hay muchos otros microcontroladores y plataformas con microcontroladores disponibles para la computación física. Muchos otros ofrecen funcionalidades similares, pero Arduino, ofrece algunas ventajas respecto a otros sistemas a profesores y estudiantes:

- Asequible: Las placas Arduino son más asequibles comparadas con otras plataformas de microcontroladores.
- Multi-Plataforma: El software de Arduino funciona en los sistemas operativos Windows, Mac y Linux. La mayoría de los entornos para microcontroladores están limitados a Windows.
- Entorno de programación simple y directo: El entorno de programación de Arduino es fácil de usar para principiantes y lo suficientemente flexible para los usuarios avanzados.
- Software ampliable y de código abierto: El software Arduino está publicado bajo una licencia libre y preparada para ser ampliado por programadores experimentados. El lenguaje puede ampliarse a través de librerías de C++, y si se está interesado en profundizar en los detalles técnicos, se puede dar el salto a la programación en el lenguaje AVR C en el que está basado.
- Hardware ampliable y de código abierto: Los planos de los módulos están publicados bajo licencia Creative Commons, por lo que diseñadores de circuitos con experiencia pueden hacer su propia versión del módulo, ampliéndolo u optimizándolo

Sin embargo Arduino también tiene algunos críticos. Aunque es evidente que la placa presenta unas características como herramienta de prototipado rápido, con un hardware potente y barato, y con unas herramientas de desarrollo software muy bien cuidadas, hay gente que duda de la capacidad de Arduino para ser una herramienta de aprendizaje real. Ya que muchos detalles están ocultos para los programadores y desarrolladores, y no enseñan las bases de los sistemas basados en micro-controladores. Por decirlo de otra manera, Arduino enseña a usar los micros, pero no enseña “qué” son los micros. Pero aun así, sigue siendo una herramienta muy buena para iniciar a la gente en el mundo de la electrónica y que no dispone de estudios universitarios o superiores.

Arduino IDE

Para realizar cualquier tipo de programación con la tecnología de Arduino, es necesario instalar el IDE (Integrated Development Environment). Se puede encontrar en la página web arduino.cc y está disponible para descargar gratuitamente por cualquier usuario.

El entorno de programación incluye todas las librerías de API necesarias para compilar los programas. Una vez tenemos un programa cargado en el microcontrolador, el funcionamiento de Arduino se basa en el código cargado. La estructura de códigos se divide en dos partes fundamentales: *setup* y *loop*. Ambas partes del código tienen un comportamiento secuencial, ejecutándose las instrucciones en el orden establecido. El *setup* es la primera parte del código que se ejecuta, haciéndolo solo una vez al iniciar el código. En esta parte es recomendable incluir la inicialización de los módulos, alimentación, entre otros, que se vayan a utilizar. El *loop* es un bucle que se ejecuta continuamente, formando un bucle infinito.

En el entorno existen una serie de botones que sirven para un manejo rápido del código:

Verificar: Chequea el código en busca de errores.



Cargar: Compila el código y lo vuelca en la placa E/S de Arduino.



Nuevo: Crea un nuevo sketch.



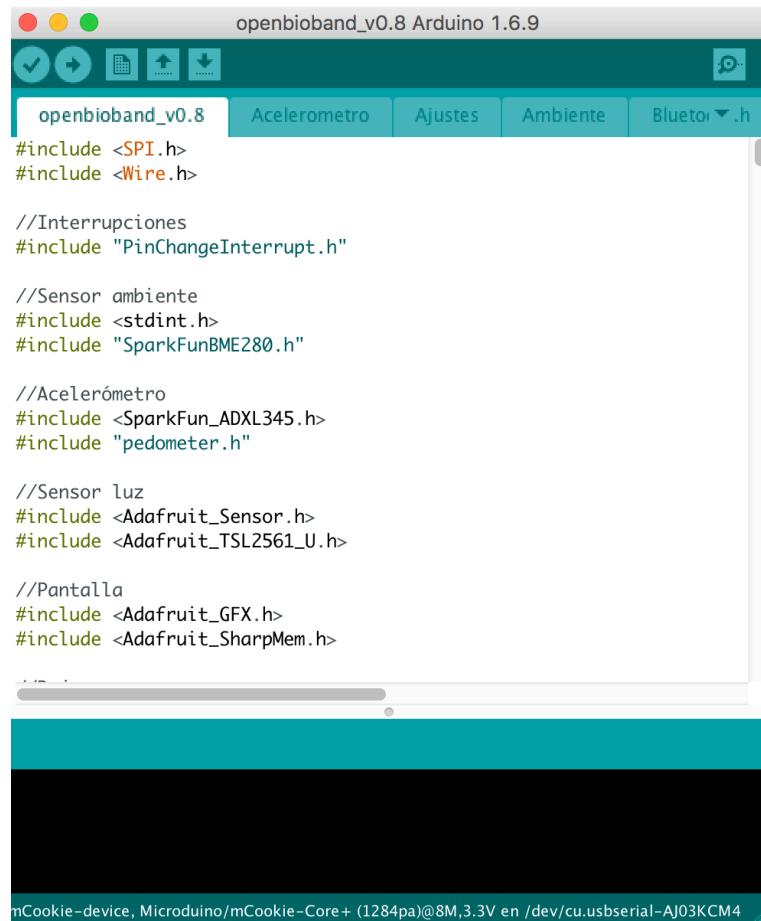
Abrir: Presenta un menú de todos los programas sketch de su "sketchbook" (librería de sketch). Un clic sobre uno de ellos lo abrirá en la ventana actual.



Guardar: Salva el programa sketch.



Monitor Serial: Inicia la monitorización serie.



The screenshot shows the Arduino IDE interface with the title bar "openbioband_v0.8 Arduino 1.6.9". Below the title bar are standard toolbar icons. The main window displays the code for the "openbioband_v0.8" sketch. The code includes headers for SPI, Wire, PinChangeInterrupt, stdint, SparkFunBME280, SparkFun_ADXL345, pedometer, Adafruit_Sensor, Adafruit_TSL2561_U, Adafruit_GFX, and Adafruit_SharpMem. The code is organized into sections for ambient sensor, accelerometer, light sensor, and display. At the bottom of the code editor, a message indicates the connection to "mCookie-device, Microduino/mCookie-Core+ (1284pa)@8M, 3.3V en /dev/cu.usbserial-AJ03KCM4".

```
#include <SPI.h>
#include <Wire.h>

//Interrupciones
#include "PinChangeInterrupt.h"

//Sensor ambiente
#include <stdint.h>
#include "SparkFunBME280.h"

//Acelerómetro
#include <SparkFun_ADXL345.h>
#include "pedometer.h"

//Sensor luz
#include <Adafruit_Sensor.h>
#include <Adafruit_TSL2561_U.h>

//Pantalla
#include <Adafruit_GFX.h>
#include <Adafruit_SharpMem.h>
```

Fig. 95 IDE Arduino

Lenguaje de programación: C/C++

C es un lenguaje de programación orientado a la implementación de Sistemas Operativos. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones. Se trata de un lenguaje débilmente tipificado de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

C++ es un lenguaje de programación diseñado con la intención de extender el exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. Las propiedades de este tipo de lenguajes son:

- Un núcleo del lenguaje simple, con funcionalidades añadidas importantes, como funciones matemáticas y de manejo de archivos, proporcionadas por bibliotecas.
- Es un lenguaje muy flexible que permite programar con múltiples estilos.
- Un sistema de tipos que impide operaciones sin sentido.
- Usa un lenguaje de pre-procesado.
- Acceso a memoria de bajo nivel mediante el uso de punteros.
- Interrupciones al procesador con uniones.
- Un conjunto reducido de palabras clave.
- Tipos de datos agregados (struct) que permiten que datos relacionados se combinen y se manipulen como un todo.

ANEXO 4. Bootloader PCB

Cuando cargamos un programa en Arduino desde el USB con el IDE, estamos haciendo uso del bootloader, se trata de un pequeño programa que ha sido guardado previamente en el microcontrolador de la placa y que nos permite cargar código sin necesidad de hardware adicional. El bootloader solo está activo unos segundos cuando se resetea el Arduino y después comienza el sketch que está cargado en la flash de Arduino y que hemos programado y subido a la placa.

Cuando compramos un microcontrolador, éste viene sin una configuración de fábrica, por lo que para poder programarlo con Arduino, lo primero que tendremos que hacer es quemarle un bootloader. Hay que tener en cuenta el micro, en este caso es el 1284p por lo que habrá que seleccionar una placa en el IDE que utilice ese micro.

Dado que en la primera pulsera se realizó el bootloader de Microduino con el micro 644p, en este caso lo realizaremos también con Microduino con el micro 1284p 8MHz y 3.3V.

Para poder utilizar este placa hay que descargarse un Arduino IDE especial llamado Arduino for Microduino, que puede ser encontrado en su [Wiki](#).

Para cargar o “quemar” el bootloader, necesitaremos un programador externo (in-system programmer), un programador paralelo u otro arduino con un programa adecuado cargado. En este caso para programarlo se ha utilizado un programador externo y se ha dejado una zona de la placa para el ICSP como se puede ver en la fotografía.

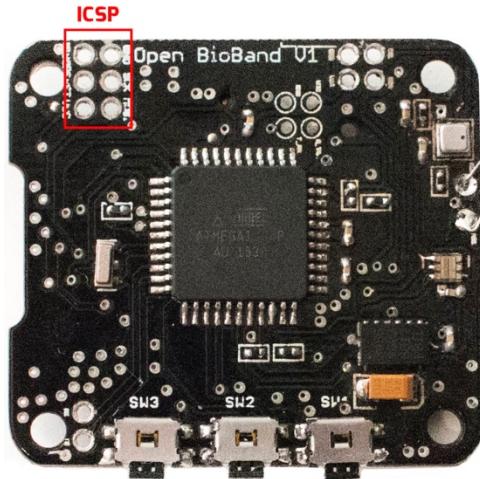


Fig. 96 Conektor ICSP PCB

El programador se conecta al ICSP con la batería puesta, se selecciona la placa y el programador correctos: Microduino 1284p 8M 3.3V y avrisp mkII y por último se le da a quemar bootloader. Este proceso tarda unos 20 segundos.

Una vez quemado el bootloader, el dispositivo ya podrá programarse con el Arduino IDE gracias al programador externo o con un ftdi por UART.

La posibilidad de usar un FTDI por UART permite poder utilizar el monitor serial de Arduino y poder ir probando cada sensor al soldarlo para saber si está todo soldado y conectado correctamente.

ANEXO 5. Primeras pruebas: pulsera versión 0

Como primer prototipo y toma de contacto en el mundo wearable, se ha realizado una pulsera basada en el ‘OS Watch’, un wearable Open Source, al que se le incluye además un sensor de pulso.

Dicho sistema utiliza como componente principal, debido a su bajo coste, su tamaño y memoria, un Microduino, una placa basada en la plataforma Arduino.

Microduino tiene el microcontrolador ATmega644 a 5V y 16MHz, tiene 24 pines digitales, 8 pines analógicos y 2 puertos serie. Tiene 3 pines de interrupción e interfaces SPI e I2C.

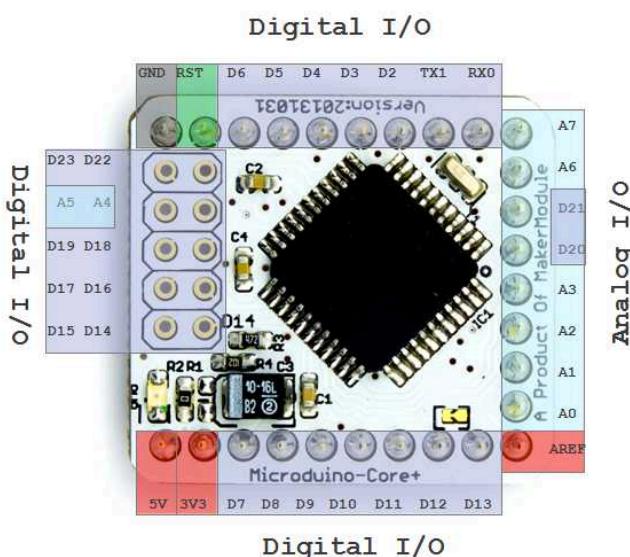


Fig. 97 Microduino

El proceso de la pulsera se divide en 3 partes:

1. Hardware
2. Impresión 3D
3. Firmware

HARDWARE

Antes de empezar con el montaje de Hardware se realizó unas modificaciones en el Microduino ya que funciona a 16MHz y 5V y queríamos que funcionará a 8MHz y 3.3V, ya que el ble funciona a 3.3V y a esta tensión y frecuencia consume menos el micro.

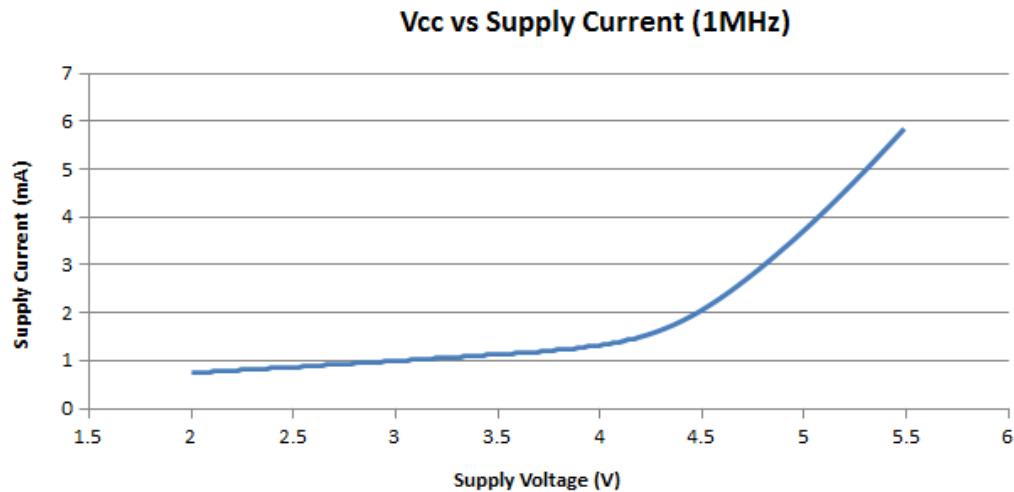


Fig. 98 Gráfica Tensión vs Intensidad

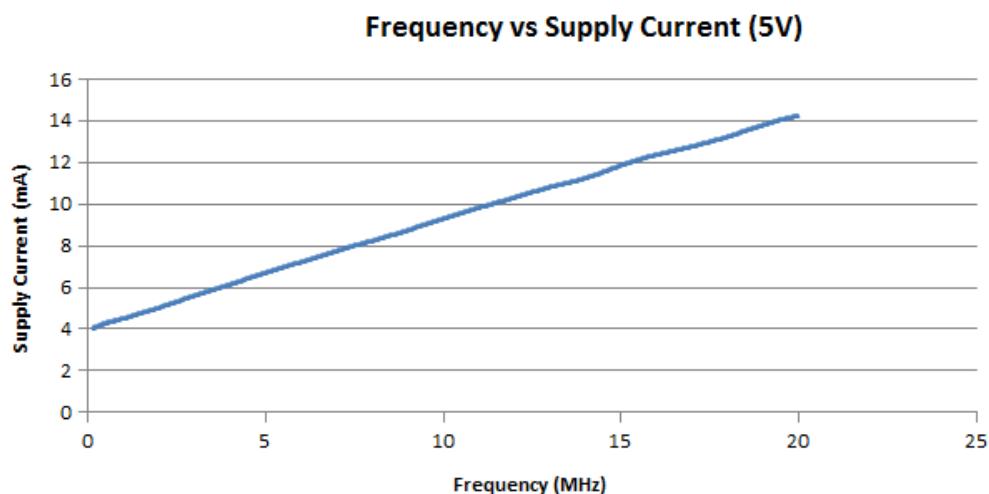


Fig. 99 Gráfica Frecuencia vs Intensidad

Una vez cambiado el cristal del Microduino, se ha quemado el bootloader para que trabaje a dicha frecuencia y tensión y ya puede programarse con el IDE de Arduino. Esto puede hacerse con un FTDI o con un Arduino sin microcontrolador.

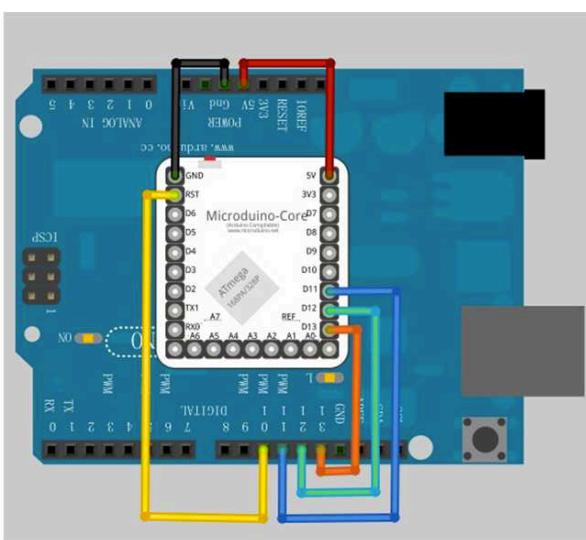


Fig. 100 Arduino bootloader

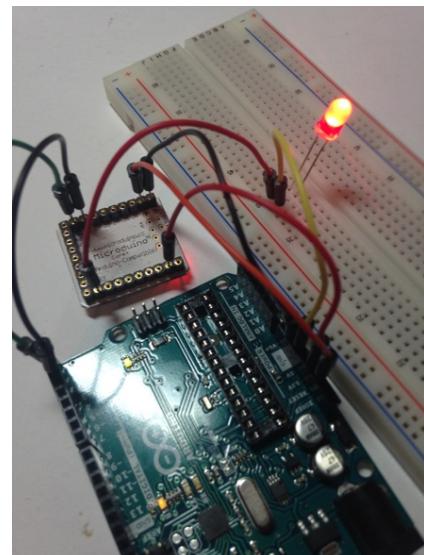


Fig. 101 Arduino como programador

Por otro lado, para poder utilizar el BLE primero se tuvo que instalar el firmware. Para ello se ha utilizado un CC Debugger y el software SW Update tool que se encuentra en la página de Bluegiga.

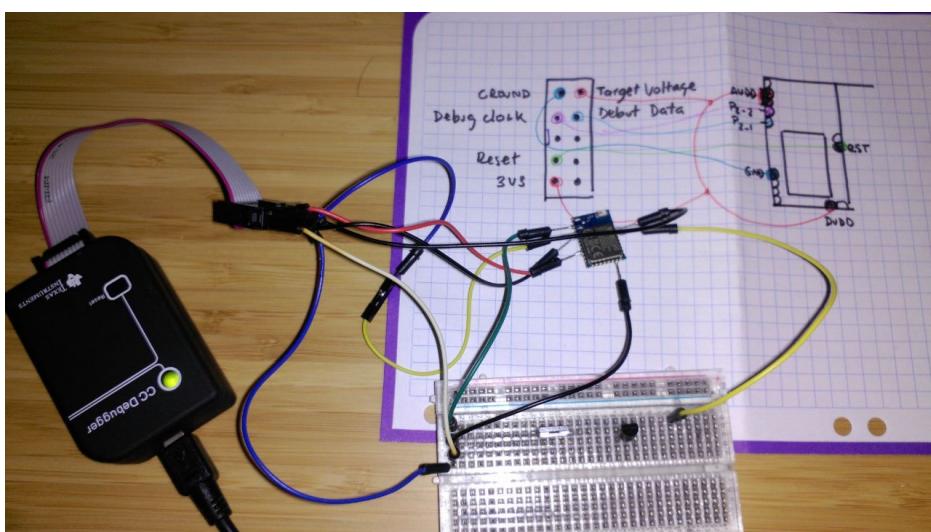


Fig. 102 Firmware BLE112

Al ir a cargar el firmware en el Bluetooth, no se detectaba el BLE. Para solucionarlo se tuvieron que soldar todas las masas debido a que había un problema de ruido.

Una vez listos todos componentes, empezó el proceso de montaje:

El sistema consta de: Microduino, 2 leds, 4 pulsadores, una pantalla OLED128x64, BLE bluegiga, una batería LiPo 3.7v, un interruptor, un conector JST, un regulador de voltaje 3.3V, resistencias de 10k, 33, 1k, un condensador de 0.1uF, un transistor NPN, un diodo, un motor de vibración y un sensor de pulso.

Lo primero que se ha construido ha sido la parte central de la pulsera, donde va situado el Microduino, el BLE, los 4 botones y el regulador de tensión. Además se han añadido unos pines externos a la pulsera para poder programar el microcontrolador sin tener que abrirla.

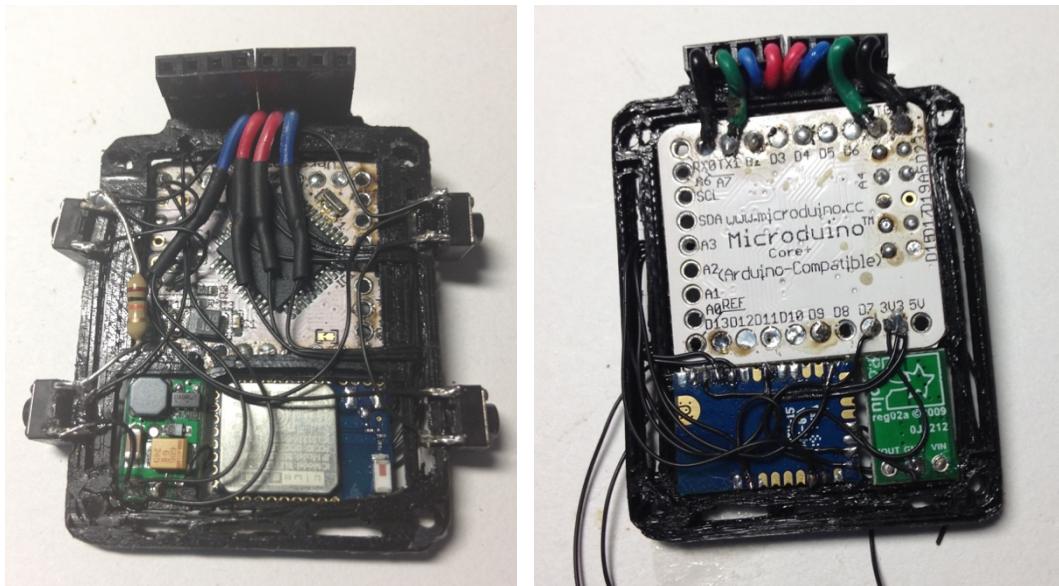


Fig. 103 Montaje pulsera versión 0

Lo siguiente fue construir la parte delantera y unirla a ésta. Donde van la pantalla y dos leds. Uno de ellos será usado como referencia de nuestro pulso.

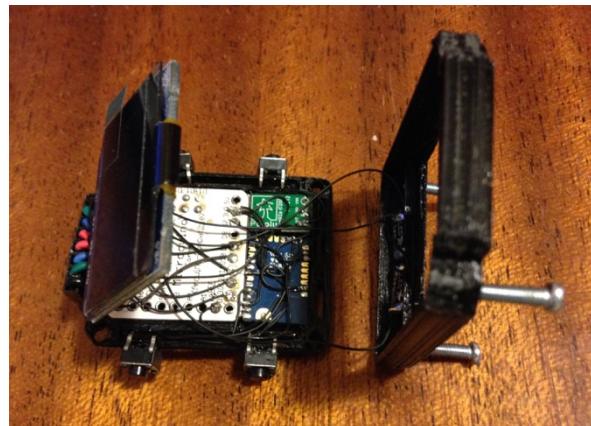


Fig. 104 Montaje pulsera versión 0

Y por último, el montaje de la parte trasera, donde va el motor, el interruptor, el conector JST para cargar la batería, la batería y un sensor de pulso como añadido.

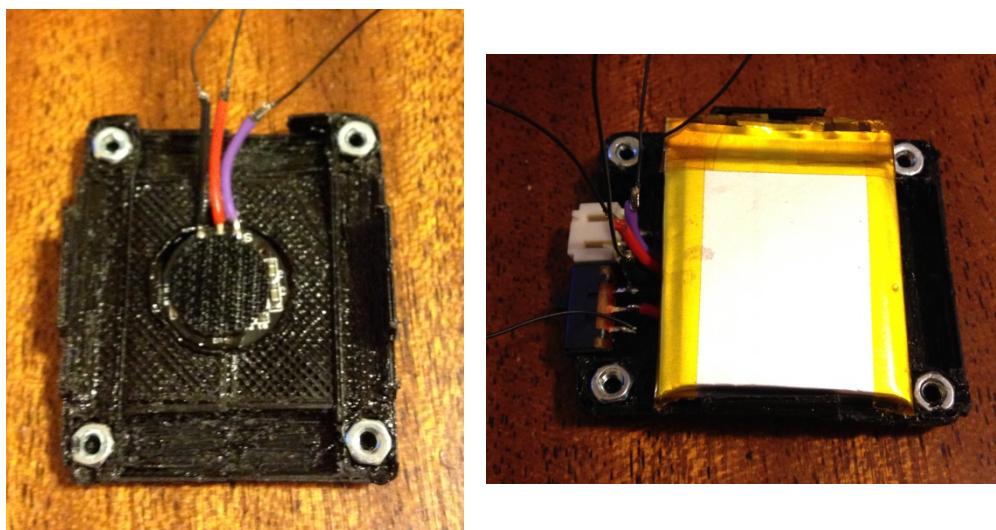


Fig. 105 Montaje pulsera versión 0

Para el sensor de pulso se ha escogido pulse sensor, un sensor óptico construido especialmente para Arduino, con su propia librería e interfaz en processing.

Dicho sensor funciona como un fotopletismógrafo, un dispositivo utilizado para capturar el pulso de una persona tras colocar un emisor de luz (infrarroja o visible) y un receptor en contraposición con un obstáculo en medio (el cual sería en la mayoría de los casos la punta de los dedos). Cuando se produce un bombeo de sangre del corazón, la presión sanguínea en el dedo aumenta por lo que se produce una minúscula interrupción del haz de luz.

La señal de pulso del corazón que sale de un fotopletismógrafo es una fluctuación en el voltaje analógico, y tiene una forma de onda predecible como se muestra en la figura conocida como PPG. El sensor lo que hace es amplificar la señal sin procesar del sensor de pulso anterior, y normaliza la onda de pulso en torno a V/2 (punto medio de la tensión). Además, responde a los cambios relativos en la intensidad de la luz. Si la cantidad de luz que incide sobre el sensor se mantiene constante, el valor de señal permanecerá en (o cerca de) 512 (punto medio del rango ADC). Si hay más luz, la señal sube y si hay menos luz, todo lo contrario. La luz del led verde que se refleja en el sensor cambia durante cada pulso.

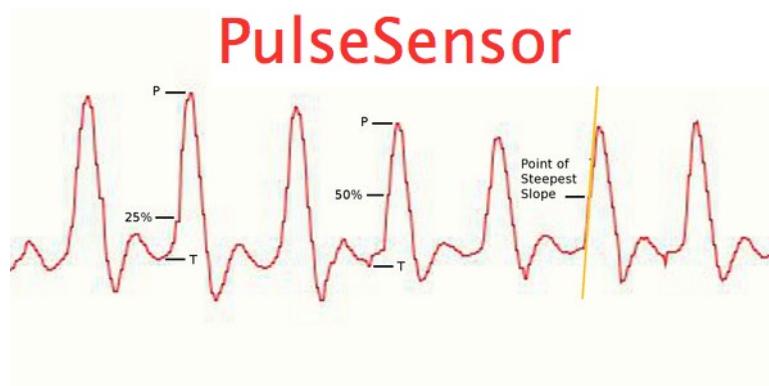


Fig. 106 Gráfica pulso

IMPRESIÓN 3D

Para la carcasa de la pulsera se ha utilizado impresión 3D debido a su bajo coste y su fácil accesibilidad.

El problema de la impresión 3D es que si la impresora no está bien calibrada, las piezas no son lo suficientemente perfectas para su correcto anclaje.

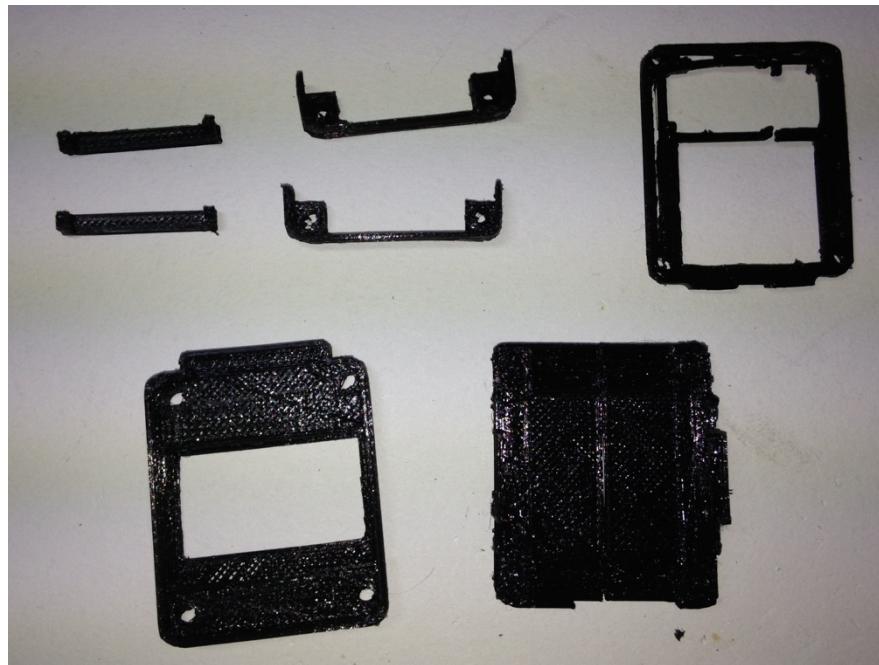


Fig. 107 Piezas 3D pulsera versión 0

En la siguiente imagen puede verse el resultado final de la pulsera:



Fig. 108 Pulsera diseño final

FIRMWARE

Se realizó un software para la pulsera con un menú básico de navegación al que podías acceder con los pulsadores y ver los datos en tiempo real del sensor de pulso.

ANEXO 6. Firmware principal

/*

Open BioBand is a biometric Open Source wristband with BLE (Bluetooth Low Energy) connectivity used to non-invasive monitorization of patients and their environment. It measure several parameters such as pulse, SPO2, body temperature, motion activity, pressure, humidity, temperature...

It allow developers to investigate and learn about wearables and biometric sensors and artists to create new biological performances and projects.

Created as final thesis Industrial Engineering project.

This firmware manage all the circuits and sensors integrated in Open BioBand.

Created by Esther Borao Moros, September 1, 2017.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Version: 0.15
Implementation: Esther Borao Moros

*/

```
*****  
Includes  
*****  
  
// Librerías generales Arduino  
#include <Wire.h>  
#include <SPI.h>  
#include <Arduino.h>  
  
// Librería interrupciones  
#include "PinChangeInterrupt.h"  
  
// Librerías sensor parámetros ambientales BME280  
#include <stdint.h>  
#include "SparkFunBME280.h"  
  
// Librerías sensor acelerómetro ADXL345  
#include <SparkFun_ADXL345.h>  
#include "pedometer.h"  
  
// Librería sensor luz TSL2561  
#include <SparkFunTSL2561.h>  
  
// Librerías pantalla SharpMem  
#include <Adafruit_GFX.h>  
#include <Adafruit_SharpMem.h>  
  
// Librería bajo consumo  
#include <LowPower.h>  
  
// Librería pulsadores  
#include <Button.h>  
  
// Librerías sensor biométrico MAX30105  
#include "MAX30105.h"  
#include "HeartRate.h"  
#include "spo2_algorithm.h"  
  
// Librerías módulo BLE MDBT40 (nRF51822)  
#include "Adafruit_BLE.h"  
#include "Adafruit_BluefruitLE_UART.h"  
*****
```

Pulsera biométrica Open Source para la monitorización del usuario

```
*****  
Defines y variables  
*****  
  
*****  
Modo Debug: descomentar el siguiente comentario para activar el modo.  
*****  
  
//#define BIOBAND_DEBUG  
  
// Define en que salida se imprime el Modo Debug  
#define DEBUG_SERIAL Serial  
  
// Configuración Modo Debug  
#ifdef BIOBAND_DEBUG  
    #define DEBUG_PRINT(...) { DEBUG_DEBUG_PRINT(__VA_ARGS__); }  
    #define DEBUG_PRINTLN(...) { DEBUG_SERIAL.println(__VA_ARGS__); }  
#else  
    #define DEBUG_PRINT(...) {}  
    #define DEBUG_PRINTLN(...) {}  
#endif  
  
*****  
  
// Control módulo BLE MDBT40 (nRF51822)  
#define LONGITUDBUFFER 128 // Tamaño del buffer de recepción de  
datos  
#define MODO_VERBOSE true // Con valor "true" se activa el modo  
debug  
#define MODO_UART_PIN 12 // Pin de control de modo  
  
#ifdef Serial1  
    #define NOMBRE_HW_SERIAL Serial1  
#endif  
  
#define RESET_FABRICA_HABILITADO 1  
#define VERSION_MINIMA_FIRMWARE "0.6.6"  
#define MODO_LED_AVISO "MODE"  
  
Adafruit_BluefruitLE_UART ble(Serial1, MODO_UART_PIN);  
  
// Control pantalla SharpMem  
#define SCK 10  
#define MOSI 11  
#define SS 13  
  
Adafruit_SharpMem pantallaBioBand(SCK, MOSI, SS);  
  
#define NEGRO 0  
#define BLANCO 1  
  
// Control pulsadores  
#define MBOT 6  
#define UBOT 9  
#define DBOT 8  
  
// Control batería  
#define VBATPIN A2  
#define CHGPIN 7
```

```
int porcentajeBateria = 0;
int estadoBateria;

// Control iluminación
#define ILUMINACION 1

byte menuIlluminacion = 1;
byte estadoIlluminacion = 1;
double lux;

// Control sensor temperatura
#define LMT70_TAO A1
#define LMT70_TON 5

float LMT70_AMul = -0.000000001809628;
float LMT70_BMul = -0.000003325395;
float LMT70_CMul = -0.1814103;
float LMT70_DMul = 205.5894;

// Control pulsadores
Button botonCentral(MBOT, true, true, 20);
Button botonArriba(UBOT, true, true, 20);
Button botonAbajo(DBOT, true, true, 20);

// Control sensor biométrico MAX30105
#define BRILLO_MAXIMO 255

MAX30105 sensorBiometrico;

bool primeraMedida;
uint32_t bufferIR[100]; // datos del sensor IR
uint32_t bufferRojo[100]; // datos del sensor Rojo
int32_t longitudBuffer; // longitud de datos
int32_t spo2; // valor SPO2
int8_t validSPO2; // indicador de SPO2 válido
int32_t pulsoCardiaco; // valor ritmo cardíaco
int8_t validPulsoCardiaco; //indicador de ritmo cardíaco válido

// Control sensor parámetros ambientales BME280
BME280 sensorAmbiente;

float temperatura;
float presion;
float altitud;
float humedad;

// Control sensor luz TSL2561
SFE_TSL2561 sensorLuz;

boolean gananciaLuz;
unsigned int ms;

// Control navegación pantallas
unsigned long tiempoStandby;
byte tiempoActivo = 15;
boolean active = false;

byte paginaActual = 0;
byte menuVal = 0;
byte configVal = 0;
```

```
boolean flicker = false;
unsigned long paradaRelojTimer = 0;
boolean paradaRelojActivo = false;
unsigned long paradaRelojMs = 0;

// Control sensor batería
boolean visualizacionVoltaje = true;

// Control reloj
int u_hora = 0;
int u_minuto = 0;
int u_segundo = 0;
int d_hora = 0;
int d_minuto = 0;
int d_segundo = 0;
unsigned long timer1 = 0;
unsigned long timer2 = 0;

// Control sensor acelerómetro ADXL345
Pedometer acelerometro;

int peso = 50;      //kg
int altura = 160;  //cm
float long_paso;
float calorias_milla;
float pasos_milla;
float calorias_quemadas;
float distancia;
float factor;

/********************************************/
```

```
*****  
    Inicialización  
*****  
  
void setup(void) {  
  
    // Configuración puertos serie  
    //Serial.begin(9600);  
    Serial1.begin(115200);  
  
    // Configuración batería  
    pinMode(CHGIN, INPUT_PULLUP);  
    //attachPCINT(digitalPinToPCINT(CHGIN), despertarMicro, FALLING);  
  
    // Configuración sensor luz TSL2561  
    pinMode(ILUMINACION, OUTPUT);  
  
    // Configuración sensor acelerómetro ADXL345  
    acelerometro.init();  
  
    // Configuración sensor biométrico MAX30105  
    if (!sensorBiometrico.begin(Wire, I2C_SPEED_FAST))  
    {  
        DEBUG_PRINT("MAX30101 no fue encontrado. ");  
        DEBUG_PRINTLN("Comprueba el cableado o la alimentacion ");  
        while (1);  
    }  
  
    primeraMedida = 1;  
    byte brilloLed = 60; // 0=apagado to 255=50mA  
    byte mediaMuestreo = 4; // 1, 2, 4, 8, 16, 32  
    byte modoLed = 2; // 1 = Rojo, 2 = Rojo + IR, 3 = Rojo + IR + Verde  
    byte ratioMuestreo = 100; // 50, 100, 200, 400, 800, 1000, 1600,  
    3200  
    int anchoPulso = 411; // 69, 118, 215, 411  
    int rangoAdc = 4096; // 2048, 4096, 8192, 16384  
  
    sensorBiometrico.setup(brilloLed, mediaMuestreo, modoLed,  
    ratioMuestreo, anchoPulso, rangoAdc);  
  
    // Configuración sensor luz TSL2561  
    unsigned char ID;  
    gananciaLuz = 0;  
    unsigned char tiempoLuz = 2;  
  
    sensorLuz.begin();  
    sensorLuz.getID(ID);  
    sensorLuz.setTiming(gananciaLuz, tiempoLuz, ms);  
    sensorLuz.setPowerUp();  
  
    // Configuración y calibración sensor parámetros ambientales BME280  
    sensorAmbiente.settings.commInterface = I2C_MODE;  
    sensorAmbiente.settings.I2CAddress = 0x77;  
    sensorAmbiente.settings.runMode = 3; // Modo Normal  
    sensorAmbiente.settings.tStandby = 0; // Tiempo de Standby  
    sensorAmbiente.settings.filter = 0;  
    sensorAmbiente.settings.tempOverSample = 1;  
    sensorAmbiente.settings.pressOverSample = 1;  
    sensorAmbiente.settings.humidOverSample = 1;
```

Pulsera biométrica Open Source para la monitorización del usuario

```
delay(10); //Requiere 2ms para empezar
sensorAmbiente.begin();
sensorAmbiente.readRegister(BME280_CHIP_ID_REG);
sensorAmbiente.readRegister(BME280_RST_REG);
sensorAmbiente.readRegister(BME280_CTRL_MEAS_REG);
sensorAmbiente.readRegister(BME280_CTRL_HUMIDITY_REG);

uint8_t contadorMem = 0x80;
uint8_t lecturaDatoTemp;

for (int rowi = 8; rowi < 16; rowi++)
{
    DEBUG_PRINT("0x");
    DEBUG_PRINT(rowi, HEX);
    DEBUG_PRINT("0:");
    for (int coli = 0; coli < 16; coli++)
    {
        lecturaDatoTemp = sensorAmbiente.readRegister(contadorMem);
        DEBUG_PRINT((lecturaDatoTemp >> 4) & 0x0F, HEX); // Imprimir
        primera parte en hexadecimal
        DEBUG_PRINT(lecturaDatoTemp & 0x0F, HEX); // Imprimir segunda
        parte en hexadecimal
        DEBUG_PRINT(" ");
        contadorMem++;
    }
    DEBUG_PRINT("\n");
}

sensorAmbiente.calibration.dig_T1;
sensorAmbiente.calibration.dig_T2;
sensorAmbiente.calibration.dig_T3;

sensorAmbiente.calibration.dig_P1;
sensorAmbiente.calibration.dig_P2;
sensorAmbiente.calibration.dig_P3;
sensorAmbiente.calibration.dig_P4;
sensorAmbiente.calibration.dig_P5;
sensorAmbiente.calibration.dig_P6;
sensorAmbiente.calibration.dig_P7;
sensorAmbiente.calibration.dig_P8;
sensorAmbiente.calibration.dig_P9;

sensorAmbiente.calibration.dig_H1;
sensorAmbiente.calibration.dig_H2;
sensorAmbiente.calibration.dig_H3;
sensorAmbiente.calibration.dig_H4;
sensorAmbiente.calibration.dig_H5;
sensorAmbiente.calibration.dig_H6;

// Configuración pantalla SharpMem
pantallaBioBand.begin();
pantallaBioBand.setRotation(0);
pantallaBioBand.clearDisplay();

// Configuración sensor temperatura
pinMode(LMT70_TON, OUTPUT);
```

```
// Configuración pulsadores
pinMode(MBOT, INPUT_PULLUP);
pinMode(UBOT, INPUT_PULLUP);
pinMode(DBOT, INPUT_PULLUP);

attachInterrupt(2, despertarMicro, FALLING);

botonCentral.read();
botonArriba.read();
botonAbajo.read();

// Configuración módulo BLE MDBT40 (nRF51822)
if ( !ble.begin(MODO_VERBOSE) ) {
    error(F("No se puede conectar con el módulo"));
}
DEBUG_PRINTLN( F("Conexion correcta") );

if ( RESET_FABRICA_HABILITADO ) {
    // Reset de fábrica para inicializar los valores
    DEBUG_PRINTLN(F("Reset de fabrica: "));
    if ( !ble.factoryReset() ) {
        error(F("No es posible el reseteo"));
    }
}
// Deshabilita el comando "echo"
ble.echo(false);

// Consulta información del módulo
ble.info();

// Activación led de indicaciones
ble.verbose(false);

if ( ble.isVersionAtLeast(VERSION_MINIMA_FIRMWARE) )
{
    DEBUG_PRINTLN(F("*****"));
    DEBUG_PRINTLN(F("Change LED activity to " MODO_LED_AVISO));
    ble.sendCommandCheckOK("AT+HWModeLED=" MODO_LED_AVISO);
    DEBUG_PRINTLN(F("*****"));
}

// Configuración e inicializacion actualización pantalla
tiempoStandby = millis() + tiempoActivo * 1000;

}

/*****************************************/
```

Pulsera biométrica Open Source para la monitorización del usuario

```
*****  
Bucle Principal  
*****  
  
void loop(void) {  
  
    // Comprobación estado carga batería  
    int carga = digitalRead(CHGPIN);  
  
    // Comprobación si está activa la navegación  
    active = (millis() <= tiempoStandby);  
  
    // Comprobación conexión módulo BLE MDBT40 (nRF51822)  
    if (ble.isConnected()) {  
        ble.print("AT+BLEUARTTX=");  
        ble.println("Connected");  
        // Comprobación de envío correcto de comando  
        if (! ble.waitForOK()) {  
            DEBUG_PRINTLN(F("Fallo al enviar"));  
        }  
  
        // Comprobación de datos entrantes  
        ble.println("AT+BLEUARTRX");  
        ble.readline();  
        if (strcmp(ble.buffer, "OK") == 0) {  
            return;  
        }  
  
        // Datos recibidos. Almacenados en el buffer  
        DEBUG_PRINT(F("[Recv] "));  
        DEBUG_PRINTLN(ble.buffer);  
        ble.waitForOK();  
    }  
  
    // Lectura botón central  
    botonCentral.read();  
  
    // Reseteo del Timer de actualización de pantalla si se detecta  
    pulsación  
    if (active && (botonArriba.read() || botonAbajo.read()))  
    {  
        tiempoStandby = millis() + tiempoActivo * 1000;  
    }  
  
    // Comprobación finalización navegación pantallas y reinicio  
    if (paginaActual == 0 && botonCentral.wasPressed()) {  
        paginaActual = 10;  
        menuVal = 0;  
        botonCentral.read();  
    }  
  
    // Conteo de segundos  
    un_seg();  
  
    // Actualización reloj  
    contador();
```

```
// Actualización navegación pantallas
if (paginaActual == 0 || !active) pantallaPrincipal();
else if (paginaActual == 10) pantallaMenu();
else if (paginaActual == 1) pantallaSalud();
else if (paginaActual == 2) pantallaAmbiente();
else if (paginaActual == 3) pantallaMovimiento();
else if (paginaActual == 4) pantallaCronometro();
else if (paginaActual == 5) pantallaAjustes();
else if (paginaActual == 6) pantallaJuego();
else if (paginaActual == 7) pantallaIluminacion();
else if (paginaActual == 8) pantallaReloj();
else if (paginaActual == 9) pantallaNotificaciones();
else pantallaBioBand.clearDisplay();

// Medición sensor luz TSL2561
unsigned int data0, data1;

sensorLuz.getData(data0, data1);
sensorLuz.getLux(gananciaLuz, ms, data0, data1, lux);

// Comprobación estado iluminación
switch (estadoIluminacion) {
    case 1:
        if (lux < 200) {
            digitalWrite(ILUMINACION, LOW);
        }
        else digitalWrite(ILUMINACION, HIGH);
        break;
    case 2:
        digitalWrite(ILUMINACION, HIGH);
        break;
    case 3:
        digitalWrite(ILUMINACION, LOW);
        break;
}

// Refresco pantalla
pantallaBioBand.refresh();

// Comprobación tiempo Stanby y activación modo bajo consumo
if (!active) {
    paginaActual = 0;
    LowPower.powerDown(SLEEP_FOREVER, ADC_OFF, BOD_OFF);
}
else if (!flicker) digitalWrite(SCK, HIGH); // Elimina error por
"flicker"

}

/********************************************/
```

Pulsera biométrica Open Source para la monitorización del usuario

```
*****  
Otras funciones  
*****  
  
//! ****  
//!     Nombre: despertarMicro  
//!     Descripción: Despierta el microcontrolador del modo de bajo  
consumo  
//!     Parámetro entrada: void  
//!     Parámetro salida: void  
//!     Ejemplo: despertarMicro();  
//! ****  
  
void despertarMicro() {  
  
    noInterrupts();  
    tiempoStandby = millis() + tiempoActivo * 1000; // Resetea Timer  
Standby  
    interrupts();  
}  
*****  
  
//! ****  
//!     Nombre: un_seg  
//!     Descripción: Rutina para el conteo de segundos  
//!     Parámetro entrada: void  
//!     Parámetro salida: void  
//!     Ejemplo: un_seg();  
//! ****  
  
//Reloj  
void un_seg() {  
  
    //Rutina para cada segundo  
    timer2 = (millis() / 1000);  
    if (timer1 != timer2) {  
        timer1 = timer2;  
        u_segundo++;  
    }  
}  
*****
```

```
/*!*****  
//!  Nombre: contador  
//!  Descripción: Rutina que lleva el conteo del reloj  
//!  Parámetro entrada: void  
//!  Parámetro salida: void  
//!  Ejemplo: contador();  
/*!*****  
  
void contador() {  
  
    // Rutina de segundos  
    if ( u_segundo == 10 ) {  
        u_segundo = 0;  
        d_segundo++;  
    }  
    if ( ( d_segundo == 6 ) && ( u_segundo == 0 ) ) {  
        d_segundo = 0;  
        u_minuto++;  
    }  
  
    // Rutina de minutos  
    if ( u_minuto == 10 ) {  
        u_minuto = 0;  
        d_minuto++;  
    }  
    if ( ( d_minuto == 6 ) && ( u_minuto == 0 ) ) {  
        d_minuto = 0;  
        u_hora++;  
    }  
  
    // Rutina de horas  
    if ( u_hora == 10 ) {  
        u_hora = 0;  
        d_hora++;  
    }  
    if ( (d_hora == 2) && (u_hora == 4) ) {  
        u_hora = 0;  
        d_hora = 0;  
    }  
}  
/******
```

Pulsera biométrica Open Source para la monitorización del usuario

```
/* ****
//!    Nombre: dibujarDigito
//!    Descripción: Dibuja en la pantalla un dígito
//!    Parámetro entrada: int posición X, int posición Y, int dígito,
bool color
//!    Parámetro salida: void
//!    Ejemplo: dibujarDigito(10,20,7,1);
/* ****

void dibujarDigito(int posX, int posY, int digito, boolean col) {

    switch (digito) {
        case 0:
            pantallaBioBand.fillRect(posX, posY, 20, 8, col);
            pantallaBioBand.fillRect(posX, posY, 7, 40, col);
            pantallaBioBand.fillRect(posX + 13, posY, 7, 40, col);
            pantallaBioBand.fillRect(posX, posY + 32, 20, 8, col);
            break;
        case 1:
            pantallaBioBand.fillRect(posX + 13, posY, 7, 40, col);
            break;
        case 2:
            pantallaBioBand.fillRect(posX, posY, 20, 8, col);
            pantallaBioBand.fillRect(posX, posY + 16, 20, 8, col);
            pantallaBioBand.fillRect(posX, posY + 24, 7, 8, col);
            pantallaBioBand.fillRect(posX + 13, posY, 7, 16, col);
            pantallaBioBand.fillRect(posX, posY + 32, 20, 8, col);
            break;
        case 3:
            pantallaBioBand.fillRect(posX, posY, 20, 8, col);
            pantallaBioBand.fillRect(posX, posY + 16, 20, 8, col);
            pantallaBioBand.fillRect(posX + 13, posY, 7, 40, col);
            pantallaBioBand.fillRect(posX, posY + 32, 20, 8, col);
            break;
        case 4:
            pantallaBioBand.fillRect(posX, posY, 7, 16, col);
            pantallaBioBand.fillRect(posX, posY + 16, 20, 8, col);
            pantallaBioBand.fillRect(posX + 13, posY, 7, 40, col);
            break;
        case 5:
            pantallaBioBand.fillRect(posX, posY, 20, 8, col);
            pantallaBioBand.fillRect(posX, posY + 16, 20, 8, col);
            pantallaBioBand.fillRect(posX + 13, posY + 24, 7, 8, col);
            pantallaBioBand.fillRect(posX, posY, 7, 16, col);
            pantallaBioBand.fillRect(posX, posY + 32, 20, 8, col);
            break;
        case 6:
            pantallaBioBand.fillRect(posX, posY, 20, 8, col);
            pantallaBioBand.fillRect(posX, posY + 16, 20, 8, col);
            pantallaBioBand.fillRect(posX + 13, posY + 24, 7, 8, col);
            pantallaBioBand.fillRect(posX, posY, 7, 40, col);
            pantallaBioBand.fillRect(posX, posY + 32, 20, 8, col);
            break;
        case 7:
            pantallaBioBand.fillRect(posX, posY, 20, 8, col);
            pantallaBioBand.fillRect(posX + 13, posY, 7, 40, col);
            break;
        case 8:
            pantallaBioBand.fillRect(posX, posY, 20, 8, col);
            pantallaBioBand.fillRect(posX, posY, 7, 40, col);
            pantallaBioBand.fillRect(posX, posY + 16, 20, 8, col);
```

```
pantallaBioBand.fillRect(posX + 13, posY, 7, 40, col);
pantallaBioBand.fillRect(posX, posY + 32, 20, 8, col);
break;
case 9:
pantallaBioBand.fillRect(posX, posY, 20, 8, col);
pantallaBioBand.fillRect(posX, posY, 7, 16, col);
pantallaBioBand.fillRect(posX, posY + 16, 20, 8, col);
pantallaBioBand.fillRect(posX + 13, posY, 7, 40, col);
pantallaBioBand.fillRect(posX, posY + 32, 20, 8, col);
break;
}
}

/*****



//! ****
//!    Nombre: error
//!    Descripción: Imprime el error de inicialización del módulo BLE
//!    Parámetro entrada: cost Respuesta del módulo tras inicialización
//!    Parámetro salida: void
//!    Ejemplo: error(FlashStringHelper);
//! ****

//Control inicialización BLE (Bluetooth Low Energy)
void error(const __FlashStringHelper*err) {

    DEBUG_PRINT(err);
    while (1);
}

/*****



//! ****
//!    Nombre: leerVoltage
//!    Descripción: Lee el valor de la batería Vcc y lo transforma en
mv
//!    Parámetro entrada: void
//!    Parámetro salida: long valor en mv de la batería
//!    Ejemplo: leerVoltage();
//! ****

long leerVoltage() {

    int resultado;
    float voltageMedido = analogRead(VBATPIN);
    voltageMedido *= 2;      // Divide para 2
    voltageMedido *= 3.3;    // Multiplica por 3.3, la referencia de
tensión
    voltageMedido /= 1024;   // Convierte en tensión
    DEBUG_PRINT("VBat: " ); DEBUG_PRINTF(voltageMedido);

    resultado = voltageMedido * 1000; // Calcula Vcc (en mV); 1125300 =
1.1*1023*1000
    return resultado; // Devuelve Vcc en mV
}

*****
```

ANEXO 7. Firmware pantallas secundarias

```
*****  
Pantalla Principal  
*****  
  
//!*****  
//!    Nombre: pantallaPrincipal  
//!    Descripción: Pantalla principal de inicio y estado reposo  
//!    Parámetro entrada: void  
//!    Parámetro salida: void  
//!    Ejemplo: pantallaPrincipal();  
//!*****  
  
void pantallaPrincipal() {  
  
    // Definición logos  
#define anchoBleno 11  
#define altoBleno 14  
  
    static const unsigned char bleno_img[] PROGMEM =  
    {  
        0xc, 0x0,  
        0xa, 0x20,  
        0x9, 0x60,  
        0x8, 0xe0,  
        0x49, 0xc0,  
        0xb, 0x80,  
        0x1f, 0x0,  
        0xe, 0x0,  
        0x1e, 0x0,  
        0x39, 0x0,  
        0x78, 0x80,  
        0xe9, 0x0,  
        0xca, 0x0,  
        0x8c, 0x0  
    };  
  
#define anchoBlesi 10  
#define altoBlesi 14  
  
    static const unsigned char blesi_img[] PROGMEM =  
    {  
        0xc, 0x0,  
        0xa, 0x0,  
        0x9, 0x0,  
        0x8, 0x80,  
        0x49, 0x0,  
        0x2a, 0x0,  
        0x1c, 0x0,  
        0x1c, 0x0,  
        0x2a, 0x0,  
        0x49, 0x0,  
        0x8, 0x80,  
        0x9, 0x0,  
        0xa, 0x0,  
        0xc, 0x0  
    };
```

```
#define anchoBat 22
#define altoBat 11

static const unsigned char bat0_img[] PROGMEM =
{
    0xff, 0xff, 0xf0,
    0xff, 0xff, 0xf0,
    0xc0, 0x0, 0x30,
    0xc0, 0x0, 0x3c,
    0xff, 0xff, 0xf0,
    0xff, 0xff, 0xf0
};

static const unsigned char bat1_img[] PROGMEM =
{
    0xff, 0xff, 0xf0,
    0xff, 0xff, 0xf0,
    0xc0, 0x0, 0x30,
    0xd8, 0x0, 0x3c,
    0xc0, 0x0, 0x30,
    0xff, 0xff, 0xf0,
    0xff, 0xff, 0xf0
};

static const unsigned char bat2_img[] PROGMEM =
{
    0xff, 0xff, 0xf0,
    0xff, 0xff, 0xf0,
    0xc0, 0x0, 0x30,
    0xd9, 0x80, 0x3c,
    0xd9, 0x80, 0x3c,
    0xd9, 0x80, 0x3c,
    0xd9, 0x80, 0x3c,
    0xc0, 0x0, 0x30,
    0xff, 0xff, 0xf0,
    0xff, 0xff, 0xf0
};
```

```
static const unsigned char bat3_img[] PROGMEM =
{
  0xff, 0xff, 0xf0,
  0xff, 0xff, 0xf0,
  0xc0, 0x0, 0x30,
  0xd9, 0x98, 0x3c,
  0xc0, 0x0, 0x30,
  0xff, 0xff, 0xf0,
  0xff, 0xff, 0xf0
};

static const unsigned char bat4_img[] PROGMEM =
{
  0xff, 0xff, 0xf0,
  0xff, 0xff, 0xf0,
  0xc0, 0x0, 0x30,
  0xd9, 0x99, 0xbc,
  0xc0, 0x0, 0x30,
  0xff, 0xff, 0xf0,
  0xff, 0xff, 0xf0
};

static const unsigned char rayo_img[] PROGMEM =
{
  0x1f, 0x80,
  0x3f, 0x0,
  0x3e, 0x0,
  0x7c, 0x0,
  0x78, 0x0,
  0xfe, 0x0,
  0xfe, 0x0,
  0x1c, 0x0,
  0x38, 0x0,
  0x30, 0x0,
  0x60, 0x0,
  0x40, 0x0
};
```

```
static const unsigned char logo_img[] PROGMEM =
{
    0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0,
    0x0, 0x7, 0xf8, 0x0, 0x0, 0x3f, 0xc0, 0x0,
    0x0, 0xf, 0xff, 0x0, 0x0, 0x7f, 0xf0, 0x0,
    0x0, 0x3f, 0xff, 0x80, 0x1, 0xff, 0xfc, 0x0,
    0x0, 0xff, 0xff, 0xe0, 0x7, 0xff, 0xfe, 0x0,
    0x1, 0xff, 0xff, 0xf0, 0xf, 0xff, 0xff, 0x0,
    0x3, 0xff, 0xff, 0xf8, 0x1f, 0xff, 0xff, 0x80,
    0x3, 0xff, 0xff, 0xf8, 0x1f, 0xff, 0xff, 0xc0,
    0x3, 0xff, 0xff, 0xfc, 0x3f, 0xff, 0xff, 0xc0,
    0x7, 0xff, 0xff, 0xfe, 0x7f, 0xff, 0xff, 0xe0,
    0x7, 0xff, 0xff, 0xfe, 0x7f, 0xff, 0xff, 0xe0,
    0xf, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0,
    0xf, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0,
    0xf, 0xff, 0xff, 0x7f, 0xff, 0xff, 0xf0,
    0xf, 0xff, 0xfe, 0x3f, 0xff, 0xff, 0xf0,
    0x1f, 0xff, 0xfc, 0x3f, 0xff, 0xff, 0xf8,
    0x1f, 0xff, 0xfc, 0x1f, 0xff, 0xff, 0xf8,
    0x1f, 0xff, 0xfc, 0x1f, 0xff, 0xff, 0xf8,
    0x1f, 0xff, 0xf8, 0x1f, 0xff, 0xff, 0xf8,
    0x1f, 0xff, 0xf8, 0x1f, 0xff, 0xff, 0xf8,
    0x1f, 0xff, 0xf0, 0x1f, 0xff, 0x81, 0xff, 0xf8,
    0xf, 0xff, 0xf0, 0x9f, 0xff, 0x80, 0xff, 0xf0,
    0xf, 0xff, 0xf1, 0x9f, 0xff, 0x0, 0xff, 0xf0,
    0xf, 0xff, 0xf1, 0x8f, 0x80, 0x0, 0xff, 0xf0,
    0xf, 0xff, 0xf1, 0x8f, 0x0, 0x0, 0xff, 0xf0,
    0x0, 0x0, 0x3, 0xcf, 0x0, 0x0, 0xff, 0xf0,
    0x0, 0x0, 0x7, 0xc6, 0x3f, 0x0, 0xff, 0xe0,
    0x3f, 0xff, 0xff, 0xe6, 0x3f, 0x80, 0xff, 0xc0,
    0x7f, 0xff, 0xff, 0xe6, 0x3f, 0x81, 0xff, 0xc0,
    0x7f, 0xff, 0xff, 0xe0, 0x3f, 0xff, 0xff, 0x80,
    0x7f, 0xff, 0xff, 0xe0, 0x7f, 0xff, 0xff, 0x80,
    0x3f, 0xff, 0xff, 0xe0, 0x7f, 0xff, 0xff, 0x0,
    0x0, 0x7f, 0xff, 0xe0, 0x7f, 0xfe, 0x0, 0x0,
    0x0, 0x1f, 0xff, 0xf0, 0xff, 0xff, 0xfc, 0x0,
    0x0, 0xf, 0xff, 0xf0, 0xff, 0xff, 0xf8, 0x0,
    0x0, 0x7, 0xff, 0xf0, 0xff, 0xf0, 0x0, 0x0,
    0x0, 0x3, 0xff, 0xf0, 0xff, 0xff, 0xc0, 0x0,
    0x0, 0x1, 0xff, 0xf9, 0xff, 0xff, 0x80, 0x0,
    0x0, 0x0, 0xff, 0xff, 0xff, 0xff, 0x0, 0x0,
    0x0, 0x0, 0x7f, 0xff, 0xff, 0xfe, 0x0, 0x0,
    0x0, 0x0, 0x3f, 0xff, 0xff, 0xf8, 0x0, 0x0,
    0x0, 0x0, 0xf, 0xff, 0xff, 0xf0, 0x0, 0x0,
    0x0, 0x0, 0x7, 0xff, 0xff, 0xe0, 0x0, 0x0,
    0x0, 0x0, 0x3, 0xff, 0xff, 0xc0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0xff, 0xff, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x7f, 0xfe, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x3f, 0xfc, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x1f, 0xf8, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0xf, 0xf0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x7, 0xe0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x7, 0xe0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x3, 0xc0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x1, 0x80, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0
};
```

Pulsera biométrica Open Source para la monitorización del usuario

```
// Refresco pantalla
pantallaBioBand.clearDisplay();

// Visualización pantalla principal
pantallaBioBand.drawBitmap(18, 20, logo_img, 62, 54, NEGRO);
pantallaBioBand.setCursor(12, 78);
pantallaBioBand.setTextSize(1);
pantallaBioBand.print("Open Bioband");
if (ble.isConnected()) {
    pantallaBioBand.drawBitmap(3, 2, blesi_img, anchoBlesi, altoBlesi,
NEGRO);
}
else pantallaBioBand.drawBitmap(3, 2, bleno_img, anchoBleno,
altoBleno, NEGRO);

// Visualización nivel de batería
// 4.2 = 100% (máximo voltaje, máxima carga)
// 3.2 = 0%   (mínimo voltaje de seguridad)

if (visualizacionVoltaje) {
    // Evitamos valores negativos
    if (porcentajeBateria <= 0)
    {
        porcentajeBateria = 0;
    }

    // Comprobación estado carga batería
    int carga = digitalRead(CHGPIN);

    // Cálculo de nivel de batería
    if (carga == HIGH) {
        unsigned int batVoltage = leerVoltage();
        porcentajeBateria = ((batVoltage * 0.1) - 320); // Fórmula para
mostrar el porcentaje de carga

        // Visualización de nivel de batería
        pantallaBioBand.setCursor(52, 4);
        pantallaBioBand.setTextColor(NEGRO, BLANCO);
        pantallaBioBand.print(porcentajeBateria);
        pantallaBioBand.print("%");

        if (porcentajeBateria <= 10)
        {
            pantallaBioBand.drawBitmap(72, 2, bat0_img, anchoBat, altoBat,
NEGRO);
            estadoBateria = 0;
        }

        if ((porcentajeBateria > 10) && (porcentajeBateria < 26)) {
            pantallaBioBand.drawBitmap(72, 2, bat1_img, anchoBat, altoBat,
NEGRO);
            estadoBateria = 1;
        }

        if ((porcentajeBateria > 25) && (porcentajeBateria < 51)) {
            pantallaBioBand.drawBitmap(72, 2, bat2_img, anchoBat, altoBat,
NEGRO);
            estadoBateria = 2;
        }
    }
}
```

```
if ((porcentajeBateria > 50) && (porcentajeBateria < 76)) {
    pantallaBioBand.drawBitmap(72, 2, bat3_img, anchoBat, altoBat,
NEGRO);
    estadoBateria = 3;
}

if ((porcentajeBateria > 75) && (porcentajeBateria < 101)) {
    pantallaBioBand.drawBitmap(72, 2, bat4_img, anchoBat, altoBat,
NEGRO);
    estadoBateria = 4;
}
}

// Visualización carga batería
if (carga == LOW) {
    // Inicialización actualización pantalla
    tiempoStandby = millis() + tiempoActivo * 1000;

    // Visualización logo nivel de batería
    pantallaBioBand.drawBitmap(61, 2, rayo_img, 9, 12, NEGRO);
    estadoBateria++;

    if (estadoBateria > 4) {
        estadoBateria = 0;
    }

    switch (estadoBateria) {
        case 0:
            pantallaBioBand.drawBitmap(72, 2, bat0_img, anchoBat,
altoBat, NEGRO);
            delay(100);
            break;
        case 1:
            pantallaBioBand.drawBitmap(72, 2, bat1_img, anchoBat,
altoBat, NEGRO);
            delay(100);
            break;
        case 2:
            pantallaBioBand.drawBitmap(72, 2, bat2_img, anchoBat,
altoBat, NEGRO);
            delay(100);
            break;
        case 3:
            pantallaBioBand.drawBitmap(72, 2, bat3_img, anchoBat,
altoBat, NEGRO);
            delay(100);
        case 4:
            pantallaBioBand.drawBitmap(72, 2, bat4_img, anchoBat,
altoBat, NEGRO);
            delay(200);
            break;
    }
}
}

/*****
```

Pulsera biométrica Open Source para la monitorización del usuario

```
*****  
Pantalla Menú  
*****  
  
//! ****  
//!     Nombre: pantallaMenu  
//!     Descripción: Pantalla menu principal para elegir pantalla de  
//!                         medición  
//!     Parámetro entrada: void  
//!     Parámetro salida: void  
//!     Ejemplo: pantallaMenu();  
//! ****  
  
void pantallaMenu() {  
  
    // Definición logos  
    #define anchoIcono 25  
    #define altoIcono  25  
  
    static const unsigned char juegos_img_1[] PROGMEM =  
    {  
        0x0, 0x0, 0x0, 0x0,  
        0x0, 0x0, 0x0, 0x0,  
        0xe, 0x0, 0x38, 0x0,  
        0xe, 0x0, 0x38, 0x0,  
        0xe, 0x0, 0x38, 0x0,  
        0x1, 0xc1, 0xc0, 0x0,  
        0x1, 0xc1, 0xc0, 0x0,  
        0x1, 0xc1, 0xc0, 0x0,  
        0x7, 0xff, 0xf0, 0x0,  
        0xf, 0xff, 0xf8, 0x0,  
        0xf, 0xff, 0xf8, 0x0,  
        0x1f, 0xff, 0xfc, 0x0,  
        0x3f, 0x1c, 0x7e, 0x0,  
        0x3f, 0x1c, 0x7e, 0x0,  
        0x3f, 0x1c, 0x7e, 0x0,  
        0x3f, 0xff, 0xfe, 0x0,  
        0x3f, 0xff, 0xfe, 0x0,  
        0x37, 0xff, 0xf6, 0x0,  
        0x37, 0xff, 0xf6, 0x0,  
        0x36, 0x0, 0x36, 0x0,  
        0x1, 0xe3, 0xc0, 0x0,  
        0x1, 0xe3, 0xc0, 0x0,  
        0x0, 0x0, 0x0, 0x0,  
        0x0, 0x0, 0x0, 0x0  
    };
```

```
static const unsigned char juegos_img_2[] PROGMEM =
{
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80,
    0xf1, 0xff, 0xc7, 0x80,
    0xf1, 0xff, 0xc7, 0x80,
    0xf1, 0xff, 0xc7, 0x80,
    0xfe, 0x3e, 0x3f, 0x80,
    0xfe, 0x3e, 0x3f, 0x80,
    0xfe, 0x3e, 0x3f, 0x80,
    0xf8, 0x0, 0xf, 0x80,
    0xf0, 0x0, 0x7, 0x80,
    0xf0, 0x0, 0x7, 0x80,
    0xe0, 0x0, 0x3, 0x80,
    0xc0, 0xe3, 0x81, 0x80,
    0xc0, 0xe3, 0x81, 0x80,
    0xc0, 0xe3, 0x81, 0x80,
    0xc0, 0x0, 0x1, 0x80,
    0xc0, 0x0, 0x1, 0x80,
    0xc0, 0x0, 0x1, 0x80,
    0xc8, 0x0, 0x9, 0x80,
    0xc8, 0x0, 0x9, 0x80,
    0xc9, 0xff, 0xc9, 0x80,
    0xfe, 0x1c, 0x3f, 0x80,
    0xfe, 0x1c, 0x3f, 0x80,
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80
};

static const unsigned char conf_img_1[] PROGMEM =
{
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x1c, 0x0, 0x0,
    0x0, 0x3e, 0x0, 0x0,
    0xe, 0x3e, 0x38, 0x0,
    0xe, 0xff, 0xb8, 0x0,
    0xf, 0xff, 0xf8, 0x0,
    0x3, 0xff, 0xe0, 0x0,
    0x7, 0xe3, 0xf0, 0x0,
    0x7, 0xc1, 0xf0, 0x0,
    0x1f, 0x80, 0xfc, 0x0,
    0x3f, 0x0, 0x7e, 0x0,
    0x3f, 0x0, 0x7e, 0x0,
    0x3f, 0x0, 0x7e, 0x0,
    0x1f, 0x80, 0xfc, 0x0,
    0x7, 0xc1, 0xf0, 0x0,
    0x7, 0xe3, 0xf0, 0x0,
    0x3, 0xff, 0xe0, 0x0,
    0xf, 0xff, 0xf8, 0x0,
    0xe, 0xff, 0xb8, 0x0,
    0xe, 0x3e, 0x38, 0x0,
    0x0, 0x3e, 0x0, 0x0,
    0x0, 0x1c, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0
};
```

```

static const unsigned char conf_img_2[] PROGMEM =
{
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xe3, 0xff, 0x80,
    0xff, 0xc1, 0xff, 0x80,
    0xf1, 0xc1, 0xc7, 0x80,
    0xf1, 0x0, 0x47, 0x80,
    0xf0, 0x0, 0x7, 0x80,
    0xfc, 0x0, 0x1f, 0x80,
    0xf8, 0x1c, 0xf, 0x80,
    0xf8, 0x3e, 0xf, 0x80,
    0xe0, 0x7f, 0x3, 0x80,
    0xc0, 0xff, 0x81, 0x80,
    0xc0, 0xff, 0x81, 0x80,
    0xc0, 0xff, 0x81, 0x80,
    0xe0, 0x7f, 0x3, 0x80,
    0xf8, 0x3e, 0xf, 0x80,
    0xf8, 0x1c, 0xf, 0x80,
    0xfc, 0x0, 0x1f, 0x80,
    0xf0, 0x0, 0x7, 0x80,
    0xf1, 0x0, 0x47, 0x80,
    0xf1, 0xc1, 0xc7, 0x80,
    0xff, 0xc1, 0xff, 0x80,
    0xff, 0xe3, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80
};

static const unsigned char tiempo_img_1[] PROGMEM =
{
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x8, 0x0, 0x0,
    0x0, 0x8, 0x0, 0x0,
    0x0, 0x8, 0x0, 0x0,
    0x0, 0x8, 0x0, 0x0,
    0x4, 0x8, 0x10, 0x0,
    0x2, 0x0, 0x20, 0x0,
    0x1, 0x1c, 0x40, 0x0,
    0x0, 0x77, 0x0, 0x0,
    0x0, 0xc1, 0x80, 0x0,
    0x0, 0x80, 0x80, 0x0,
    0x1, 0x80, 0xc0, 0x0,
    0x3d, 0x0, 0x5e, 0x0,
    0x1, 0x80, 0xc0, 0x0,
    0x0, 0x80, 0x80, 0x0,
    0x0, 0xc1, 0x80, 0x0,
    0x0, 0x77, 0x0, 0x0,
    0x1, 0x1c, 0x40, 0x0,
    0x2, 0x0, 0x20, 0x0,
    0x4, 0x8, 0x10, 0x0,
    0x0, 0x8, 0x0, 0x0,
    0x0, 0x8, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0
};

```

```
static const unsigned char tiempo_img_2[] PROGMEM =
{
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xf7, 0xff, 0x80,
    0xff, 0xf7, 0xff, 0x80,
    0xff, 0xf7, 0xff, 0x80,
    0xfb, 0xf7, 0xef, 0x80,
    0xfd, 0xff, 0xdf, 0x80,
    0xfe, 0xe3, 0xbf, 0x80,
    0xff, 0x88, 0xff, 0x80,
    0xff, 0x3e, 0x7f, 0x80,
    0xff, 0x7f, 0x7f, 0x80,
    0xfe, 0x7f, 0x3f, 0x80,
    0xc2, 0xff, 0xa1, 0x80,
    0xfe, 0x7f, 0x3f, 0x80,
    0xff, 0x7f, 0x7f, 0x80,
    0xff, 0x3e, 0x7f, 0x80,
    0xff, 0x88, 0xff, 0x80,
    0xfe, 0xe3, 0xbf, 0x80,
    0xfd, 0xff, 0xdf, 0x80,
    0xfb, 0xf7, 0xef, 0x80,
    0xff, 0xf7, 0xff, 0x80,
    0xff, 0xf7, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80
};

static const unsigned char pasos_img_1[] PROGMEM =
{
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0,
    0x7, 0x0, 0x70, 0x0,
    0x7, 0x0, 0x70, 0x0,
    0x7, 0x0, 0x70, 0x0,
    0x1f, 0x0, 0x7c, 0x0,
    0x1f, 0x0, 0x7c, 0x0,
    0x1f, 0x0, 0x7c, 0x0,
    0x1f, 0xff, 0xfc, 0x0,
    0x3f, 0xff, 0xfe, 0x0,
    0x3f, 0xff, 0xfe, 0x0,
    0x3f, 0xff, 0xfe, 0x0,
    0x1f, 0xff, 0xfc, 0x0,
    0x1f, 0x0, 0x7c, 0x0,
    0x1f, 0x0, 0x7c, 0x0,
    0x1f, 0x0, 0x7c, 0x0,
    0x1f, 0x0, 0x7c, 0x0,
    0x7, 0x0, 0x70, 0x0,
    0x7, 0x0, 0x70, 0x0,
    0x7, 0x0, 0x70, 0x0,
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0
};
```

```

static const unsigned char pasos_img_2[] PROGMEM =
{
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80,
    0xf8, 0xff, 0x8f, 0x80,
    0xf8, 0xff, 0x8f, 0x80,
    0xf8, 0xff, 0x8f, 0x80,
    0xe0, 0xff, 0x83, 0x80,
    0xe0, 0xff, 0x83, 0x80,
    0xe0, 0xff, 0x83, 0x80,
    0xe0, 0xff, 0x83, 0x80,
    0xe0, 0x0, 0x3, 0x80,
    0xc0, 0x0, 0x1, 0x80,
    0xc0, 0x0, 0x1, 0x80,
    0xc0, 0x0, 0x1, 0x80,
    0xe0, 0x0, 0x3, 0x80,
    0xe0, 0xff, 0x83, 0x80,
    0xe0, 0xff, 0x83, 0x80,
    0xe0, 0xff, 0x83, 0x80,
    0xe0, 0xff, 0x83, 0x80,
    0xf8, 0xff, 0x8f, 0x80,
    0xf8, 0xff, 0x8f, 0x80,
    0xf8, 0xff, 0x8f, 0x80,
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xff, 0x80
};

static const unsigned char crono_img_1[] PROGMEM =
{
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x3e, 0x0, 0x0,
    0x0, 0x3e, 0x0, 0x0,
    0x0, 0x3e, 0x8, 0x0,
    0x0, 0xff, 0x9c, 0x0,
    0x1, 0x80, 0xfe, 0x0,
    0x3, 0xe, 0x7c, 0x0,
    0xe, 0xf, 0x38, 0x0,
    0x1c, 0xf, 0x98, 0x0,
    0x30, 0xf, 0xce, 0x0,
    0x30, 0xf, 0xe6, 0x0,
    0x20, 0xf, 0xf2, 0x0,
    0x20, 0xf, 0xf2, 0x0,
    0x20, 0xf, 0xf2, 0x0,
    0x20, 0x0, 0x2, 0x0,
    0x20, 0x0, 0x2, 0x0,
    0x30, 0x0, 0x6, 0x0,
    0x1c, 0x0, 0x1c, 0x0,
    0xe, 0x0, 0x38, 0x0,
    0x3, 0x0, 0x60, 0x0,
    0x1, 0x80, 0xc0, 0x0,
    0x0, 0xff, 0x80, 0x0,
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0
};

```

```
static const unsigned char crono_img_2[] PROGMEM =
{
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xc1, 0xff, 0x80,
    0xff, 0xc1, 0xff, 0x80,
    0xff, 0xc1, 0xf7, 0x80,
    0xff, 0x0, 0x63, 0x80,
    0xfe, 0x7f, 0x1, 0x80,
    0xfc, 0xf1, 0x83, 0x80,
    0xf1, 0xf0, 0xc7, 0x80,
    0xe3, 0xf0, 0x67, 0x80,
    0xcf, 0xf0, 0x31, 0x80,
    0xcf, 0xf0, 0x19, 0x80,
    0xdf, 0xf0, 0xd, 0x80,
    0xdf, 0xf0, 0xd, 0x80,
    0xdf, 0xf0, 0xd, 0x80,
    0xdf, 0xff, 0xfd, 0x80,
    0xdf, 0xff, 0xfd, 0x80,
    0xcf, 0xff, 0xf9, 0x80,
    0xe3, 0xff, 0xe3, 0x80,
    0xf1, 0xff, 0xc7, 0x80,
    0xfc, 0xff, 0x9f, 0x80,
    0xfe, 0x7f, 0x3f, 0x80,
    0xff, 0x0, 0x7f, 0x80,
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xff, 0x80
};

static const unsigned char salud_img_1[] PROGMEM =
{
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0,
    0x7, 0x80, 0xf0, 0x0,
    0xf, 0xc1, 0xf8, 0x0,
    0x1f, 0xe3, 0xfc, 0x0,
    0x1f, 0xf7, 0xfc, 0x0,
    0x3f, 0xff, 0xfe, 0x0,
    0x3f, 0xff, 0xfc, 0x0,
    0xf, 0xff, 0xf8, 0x0,
    0x7, 0xff, 0xf0, 0x0,
    0x3, 0xff, 0xe0, 0x0,
    0x1, 0xff, 0xc0, 0x0,
    0x0, 0xff, 0x80, 0x0,
    0x0, 0x7f, 0x0, 0x0,
    0x0, 0x3e, 0x0, 0x0,
    0x0, 0x1c, 0x0, 0x0,
    0x0, 0x8, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0
};
```

```

static const unsigned char salud_img_2[] PROGMEM =
{
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80,
    0xf8, 0x7f, 0xf, 0x80,
    0xf0, 0x3e, 0x7, 0x80,
    0xe0, 0x1c, 0x3, 0x80,
    0xe0, 0x8, 0x3, 0x80,
    0xc0, 0x0, 0x1, 0x80,
    0xc0, 0x0, 0x3, 0x80,
    0xf0, 0x0, 0x7, 0x80,
    0xf8, 0x0, 0xf, 0x80,
    0xfc, 0x0, 0x1f, 0x80,
    0xfe, 0x0, 0x3f, 0x80,
    0xff, 0x0, 0x7f, 0x80,
    0xff, 0x80, 0xff, 0x80,
    0xff, 0xc1, 0xff, 0x80,
    0xff, 0xe3, 0xff, 0x80,
    0xff, 0xf7, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80
};

static const unsigned char notif_img_1[] PROGMEM =
{
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0,
    0x3f, 0xff, 0xfe, 0x0,
    0x3f, 0xff, 0xfe, 0x0,
    0x3f, 0xff, 0xfe, 0x0,
    0x38, 0x0, 0xe, 0x0,
    0x38, 0x0, 0xe, 0x0,
    0x38, 0x8, 0xe, 0x0,
    0x38, 0x3e, 0xe, 0x0,
    0x38, 0x7f, 0xe, 0x0,
    0x38, 0x8, 0xe, 0x0,
    0x38, 0x0, 0xe, 0x0,
    0x3e, 0x3f, 0xfe, 0x0,
    0x3e, 0x7f, 0xfe, 0x0,
    0x3e, 0xff, 0xfe, 0x0,
    0x7, 0xc0, 0x0, 0x0,
    0x7, 0x80, 0x0, 0x0,
    0x7, 0x0, 0x0, 0x0,
    0x6, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0
};

```

```
static const unsigned char notif_img_2[] PROGMEM =
{
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80,
    0xc0, 0x0, 0x1, 0x80,
    0xc0, 0x0, 0x1, 0x80,
    0xc0, 0x0, 0x1, 0x80,
    0xc7, 0xff, 0xf1, 0x80,
    0xc7, 0xff, 0xf1, 0x80,
    0xc7, 0xf7, 0xf1, 0x80,
    0xc7, 0xc1, 0xf1, 0x80,
    0xc7, 0x80, 0xf1, 0x80,
    0xc7, 0xf7, 0xf1, 0x80,
    0xc7, 0xff, 0xf1, 0x80,
    0xc1, 0xc0, 0x1, 0x80,
    0xc1, 0x80, 0x1, 0x80,
    0xc1, 0x0, 0x1, 0x80,
    0xf8, 0x3f, 0xff, 0x80,
    0xf8, 0x7f, 0xff, 0x80,
    0xf8, 0xff, 0xff, 0x80,
    0xf9, 0xff, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80
};

static const unsigned char reloj_img_1[] PROGMEM =
{
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x7f, 0x0, 0x0,
    0x1, 0xff, 0xc0, 0x0,
    0x3, 0xc1, 0xe0, 0x0,
    0x7, 0x0, 0x70, 0x0,
    0xe, 0x8, 0x38, 0x0,
    0x1c, 0x8, 0x1c, 0x0,
    0x18, 0x8, 0xc, 0x0,
    0x38, 0x8, 0xe, 0x0,
    0x30, 0x8, 0x6, 0x0,
    0x30, 0x8, 0x6, 0x0,
    0x30, 0xf, 0xe6, 0x0,
    0x30, 0x0, 0x6, 0x0,
    0x30, 0x0, 0x6, 0x0,
    0x38, 0x0, 0xe, 0x0,
    0x18, 0x0, 0xc, 0x0,
    0x1c, 0x0, 0x1c, 0x0,
    0xe, 0x0, 0x38, 0x0,
    0x7, 0x0, 0x70, 0x0,
    0x3, 0xc1, 0xe0, 0x0,
    0x1, 0xff, 0xc0, 0x0,
    0x0, 0x7f, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0
};
```

```

static const unsigned char reloj_img_2[] PROGMEM =
{
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80,
    0xff, 0x80, 0xff, 0x80,
    0xfe, 0x0, 0x3f, 0x80,
    0xfc, 0x3e, 0x1f, 0x80,
    0xf8, 0xff, 0x8f, 0x80,
    0xf1, 0xf7, 0xc7, 0x80,
    0xe3, 0xf7, 0xe3, 0x80,
    0xe7, 0xf7, 0xf3, 0x80,
    0xc7, 0xf7, 0xf1, 0x80,
    0xcf, 0xf7, 0xf9, 0x80,
    0xcf, 0xf7, 0xf9, 0x80,
    0xcf, 0xf0, 0x19, 0x80,
    0xcf, 0xff, 0xf9, 0x80,
    0xcf, 0xff, 0xf9, 0x80,
    0xc7, 0xff, 0xf1, 0x80,
    0xe7, 0xff, 0xf3, 0x80,
    0xe3, 0xff, 0xe3, 0x80,
    0xf1, 0xff, 0xc7, 0x80,
    0xf8, 0xff, 0x8f, 0x80,
    0xfc, 0x3e, 0x1f, 0x80,
    0xfe, 0x0, 0x3f, 0x80,
    0xff, 0x80, 0xff, 0x80,
    0xff, 0xff, 0x80,
    0xff, 0xff, 0x80
};

static const unsigned char ILUMINACION_img_1[] PROGMEM =
{
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0,
    0xf, 0xff, 0xf8, 0x0,
    0xc, 0x0, 0x18, 0x0,
    0xc, 0x0, 0x18, 0x0,
    0xf, 0xff, 0xf8, 0x0,
    0x4, 0x0, 0x10, 0x0,
    0x6, 0x0, 0x30, 0x0,
    0x3, 0x0, 0x60, 0x0,
    0x3, 0x0, 0x60, 0x0,
    0x1, 0x80, 0xc0, 0x0,
    0x1, 0x80, 0xc0, 0x0,
    0x1, 0x9c, 0xc0, 0x0,
    0x1, 0x9c, 0xc0, 0x0,
    0x1, 0x9c, 0xc0, 0x0,
    0x1, 0x9c, 0xc0, 0x0,
    0x1, 0x80, 0xc0, 0x0,
    0x0, 0x0, 0x0, 0x0,
    0x0, 0x0, 0x0, 0x0
};

```

```
static const unsigned char ILUMINACION_img_2[] PROGMEM =
{
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80,
    0xf0, 0x0, 0x7, 0x80,
    0xf3, 0xff, 0xe7, 0x80,
    0xf3, 0xff, 0xe7, 0x80,
    0xf0, 0x0, 0x7, 0x80,
    0xfb, 0xff, 0xef, 0x80,
    0xf9, 0xff, 0xcf, 0x80,
    0xfc, 0xff, 0x9f, 0x80,
    0xfc, 0xff, 0x9f, 0x80,
    0xfe, 0x7f, 0x3f, 0x80,
    0xfe, 0x7f, 0x3f, 0x80,
    0xfe, 0x63, 0x3f, 0x80,
    0xfe, 0x63, 0x3f, 0x80,
    0xfe, 0x63, 0x3f, 0x80,
    0xfe, 0x63, 0x3f, 0x80,
    0xfe, 0x7f, 0x3f, 0x80,
    0xfe, 0x7f, 0x3f, 0x80,
    0xfe, 0x7f, 0x3f, 0x80,
    0xfe, 0x7f, 0x3f, 0x80,
    0xfe, 0x0, 0x3f, 0x80,
    0xff, 0xff, 0xff, 0x80,
    0xff, 0xff, 0xff, 0x80

};

// Control menu principal
if (botonArriba.wasPressed()) menuVal--;
if (botonAbajo.wasPressed()) menuVal++;

// Navegación circular
if (menuVal > 9 && menuVal < 99) menuVal = 1;
else if (menuVal > 99) menuVal = 9;

// Control de vuelta a pantalla principal
if (botonCentral.wasPressed()) {
    paginaActual = menuVal;
    botonCentral.read(); // Comprobación de pulsación
}

// Refresco pantalla
pantallaBioBand.clearDisplay();

// Visualización menu principal
pantallaBioBand.setTextSize(1);
pantallaBioBand.setTextColor(NEGRO, BLANCO);

if (menuVal != 1) {
    pantallaBioBand.drawBitmap(5, 5, salud_img_1, anchoIcono,
altoIcono, NEGRO);
} else {
    pantallaBioBand.drawBitmap(5, 5, salud_img_2, anchoIcono,
altoIcono, NEGRO);
}

if (menuVal != 2) {
    pantallaBioBand.drawBitmap(35, 5, tiempo_img_1, anchoIcono,
altoIcono, NEGRO);
```

```

    } else {
        pantallaBioBand.drawBitmap(35, 5, tiempo_img_2, anchoIcono,
altoIcono, NEGRO);
    }

    if (menuVal != 3) {
        pantallaBioBand.drawBitmap(65, 5, pasos_img_1, anchoIcono,
altoIcono, NEGRO);
    } else {
        pantallaBioBand.drawBitmap(65, 5, pasos_img_2, anchoIcono,
altoIcono, NEGRO);
    }

    if (menuVal != 4) {
        pantallaBioBand.drawBitmap(5, 35, crono_img_1, anchoIcono,
altoIcono, NEGRO);
    } else {
        pantallaBioBand.drawBitmap(5, 35, crono_img_2, anchoIcono,
altoIcono, NEGRO);
    }

    if (menuVal != 5) {
        pantallaBioBand.drawBitmap(35, 35, conf_img_1, anchoIcono,
altoIcono, NEGRO);
    } else {
        pantallaBioBand.drawBitmap(35, 35, conf_img_2, anchoIcono,
altoIcono, NEGRO);
    }

    if (menuVal != 6) {
        pantallaBioBand.drawBitmap(65, 35, juegos_img_1, anchoIcono,
altoIcono, NEGRO);
    } else {
        pantallaBioBand.drawBitmap(65, 35, juegos_img_2, anchoIcono,
altoIcono, NEGRO);
    }
    if (menuVal != 7) {
        pantallaBioBand.drawBitmap(5, 65, ILUMINACION_img_1, anchoIcono,
altoIcono, NEGRO);
    } else {
        pantallaBioBand.drawBitmap(5, 65, ILUMINACION_img_2, anchoIcono,
altoIcono, NEGRO);
    }
    if (menuVal != 8) {
        pantallaBioBand.drawBitmap(35, 65, reloj_img_1, anchoIcono,
altoIcono, NEGRO);
    } else {
        pantallaBioBand.drawBitmap(35, 65, reloj_img_2, anchoIcono,
altoIcono, NEGRO);
    }

    if (menuVal != 9) {
        pantallaBioBand.drawBitmap(65, 65, notif_img_1, anchoIcono,
altoIcono, NEGRO);
    } else {
        pantallaBioBand.drawBitmap(65, 65, notif_img_2, anchoIcono,
altoIcono, NEGRO);
    }
}

/*********************************************
/*********************************************

```

Pantalla Biométrica

```
*****  
// ! ***** Nombre: pantallaBiometrica  
// ! Descripción: Pantalla de medición de sensores biométricos  
// ! Parámetro entrada: void  
// ! Parámetro salida: void  
// ! Ejemplo: pantallaBiometrica();  
// ! *****  
  
void pantallaSalud() {  
  
    // Definición logos  
    static const unsigned char corazon_img[] PROGMEM =  
    {  
        0x0, 0x0,  
        0x0, 0x0,  
        0x3c, 0x78,  
        0x7e, 0xfc,  
        0xff, 0xfe,  
        0xff, 0xfe,  
        0xff, 0xfe,  
        0x7f, 0xfc,  
        0x3f, 0xf8,  
        0x1f, 0xf0,  
        0xf, 0xe0,  
        0x7, 0xc0,  
        0x3, 0x80,  
        0x1, 0x0,  
        0x0, 0x0  
    };  
  
    static const unsigned char temp_img[] PROGMEM =  
    {  
        0x18,  
        0x18,  
        0x18,  
        0x1e,  
        0x18,  
        0x18,  
        0x1e,  
        0x18,  
        0x18,  
        0x3c,  
        0x7e,  
        0xff,  
        0xff,  
        0x7e,  
        0x3c  
    };  
}
```

Pulsera biométrica Open Source para la monitorización del usuario

```
static const unsigned char spo_img[] PROGMEM =
{
    0xf0, 0x3c, 0x0,
    0x80, 0x24, 0x0,
    0xf7, 0xa4, 0x0,
    0x14, 0xa4, 0x0,
    0x94, 0xa5, 0xc0,
    0xf7, 0xbc, 0x40,
    0x4, 0x1, 0xc0,
    0x4, 0x1, 0x0,
    0x0, 0x1, 0xc0
};

// Refresco pantalla
pantallaBioBand.clearDisplay();

// Fuerza el tiempo de Standby para que no vaya a la pantalla
principal
tiempoStandby = millis() + tiempoActivo * 1000;

// Medición sensor parámetros biométricos SPO2 y ritmo cardíaco
if (sensorBiometrico.available() == true) { // Comprobación dato
recibido

    // Configura la longitud de buffer en 100 medidas almacenando 4
segundos
    longitudBuffer = 100;

    //Lee las primeras 100 muestras y determina el rango de trabajo de
la señal
    if (primeraMedida == 1) {
        for (byte i = 0 ; i < longitudBuffer ; i++)
        {
            // while (sensorBiometrico.available() == false) // Comprobación dato
recibido
            sensorBiometrico.check(); // Lectura del nuevo dato del sensor

            bufferRojo[i] = sensorBiometrico.getRed();
            bufferIR[i] = sensorBiometrico.getIR();
            sensorBiometrico.nextSample(); // Al acabar con una medida,
pasa a la siguiente

            primeraMedida = 0;

            DEBUG_PRINT(F("red="));
            DEBUG_PRINT(bufferRojo[i], DEC);
            DEBUG_PRINT(F(", ir="));
            DEBUG_PRINTLN(bufferIR[i], DEC);
        }
        // Calcular ritmo cardíaco y SPO2 tras los 4 primeros segundos
de medidas
        maxim_heart_rate_and_oxygen_saturation(bufferIR, longitudBuffer,
bufferRojo, &spo2, &validSPO2, &pulsoCardiaco, &validPulsoCardiaco);
    }
    // Desplazar las 75 últimas medidas, dejando hueco para 25
    for (byte i = 25; i < 100; i++)
    {
        bufferRojo[i - 25] = bufferRojo[i];
        bufferIR[i - 25] = bufferIR[i];
    }
    // Toma 25 nuevas medidas
```

```
for (byte i = 75; i < 100; i++)
{
    // while (sensorBiometrico.available() == false) // Comprobación
dato recibido
    sensorBiometrico.check(); // Lectura del nuevo dato del sensor

    bufferRojo[i] = sensorBiometrico.getRed();
    bufferIR[i] = sensorBiometrico.getIR();
    sensorBiometrico.nextSample(); // Al acabar con una medida, pasa
a la siguiente

    DEBUG_PRINT(F("rojo="));
    DEBUG_PRINT(bufferRojo[i], DEC);
    DEBUG_PRINT(F(", ir="));
    DEBUG_PRINT(bufferIR[i], DEC);

    DEBUG_PRINT(F(", Pulso cardiaco="));
    DEBUG_PRINT(pulsoCardiaco, DEC);

    DEBUG_PRINT(F(", Pulso cardiaco valido="));
    DEBUG_PRINT(validPulsoCardiaco, DEC);

    DEBUG_PRINT(F(", SPO2="));
    DEBUG_PRINT(spo2, DEC);

    DEBUG_PRINT(F(", SPO2 valido="));
    DEBUG_PRINTLN(validSPO2, DEC);
}

// Tras 25 medidas recalcular ritmo cardíaco y SPO2
maxim_heart_rate_and_oxygen_saturation(bufferIR, longitudBuffer,
bufferRojo, &spo2, &validSPO2, &pulsoCardiaco, &validPulsoCardiaco);
}

// Medición sensor temperatura corporal
digitalWrite(LMT70_TON, HIGH);
float LMT70_lectura = analogRead(LMT70_TAO);
LMT70_lectura *= 4.9;
float A_val = LMT70_AMul * (LMT70_lectura * LMT70_lectura *
LMT70_lectura);
float B_val = LMT70_BMul * (LMT70_lectura * LMT70_lectura);
float C_val = LMT70_CMul * LMT70_lectura;
float degC = A_val + B_val + C_val + LMT70_DMul;

// Visualización parámetros biométricos
pantallaBioBand.drawBitmap(15, 13, temp_img, 8, 15, NEGRO);
pantallaBioBand.drawBitmap(10, 38, corazon_img, 15, 15, NEGRO);
pantallaBioBand.drawBitmap(10, 63, spo_img, 18, 9, NEGRO);

pantallaBioBand.setCursor(34, 18);
pantallaBioBand.setTextSize(1);
pantallaBioBand.print(degC + 10);
pantallaBioBand.println(" C");

pantallaBioBand.setCursor(34, 43);
pantallaBioBand.print(pulsoCardiaco);
pantallaBioBand.println(" BPM");
pantallaBioBand.setCursor(34, 65);
pantallaBioBand.print(spo2);
pantallaBioBand.println(" %");
```

Pulsera biométrica Open Source para la monitorización del usuario

```
// Control de vuelta a pantalla principal
if (botonCentral.wasPressed()) {
    paginaActual = 10;
    primeraMedida = 1;
    botonCentral.read(); // Comprobación de pulsación
}
}

/*****************************************/
```

```
*****  
Pantalla Ambiente  
*****  
  
//! ****  
//!     Nombre: pantallaAmbiente  
//!     Descripción: Pantalla de visualización de parámetros ambientales  
//!     Parámetro entrada: void  
//!     Parámetro salida: void  
//!     Ejemplo: pantallaAmbiente();  
//! ****  
  
void pantallaAmbiente() {  
  
    // Definición logos  
    static const unsigned char temp_img[] PROGMEM =  
    {  
        0x30,  
        0x30,  
        0x30,  
        0x38,  
        0x30,  
        0x38,  
        0x30,  
        0x78,  
        0xfc,  
        0x78,  
        0x30  
    };  
  
    static const unsigned char alt_img[] PROGMEM =  
    {  
        0x8, 0x0,  
        0x1c, 0x0,  
        0x1c, 0x0,  
        0x3e, 0x0,  
        0x3e, 0x20,  
        0x7f, 0x70,  
        0x7f, 0xf8,  
        0xff, 0xf8,  
        0xff, 0xfc,  
        0xff, 0xfe  
    };  
  
    static const unsigned char hum_img[] PROGMEM =  
    {  
        0x10,  
        0x38,  
        0x38,  
        0x7c,  
        0x7c,  
        0xfe,  
        0xfe,  
        0xfe,  
        0x7c  
    };  
  
    static const unsigned char pres_img[] PROGMEM =
```

```

{
    0x1f, 0x0,
    0x31, 0x80,
    0x60, 0xc0,
    0x42, 0x40,
    0x44, 0x40,
    0x40, 0x40,
    0x60, 0xc0,
    0x31, 0x80,
    0x1f, 0x0,
    0xa, 0x0,
    0xff, 0xe0,
    0xff, 0xe0
};

// Medición sensor parámetros ambientales
temperatura = sensorAmbiente.readTempC();
presión = sensorAmbiente.readFloatPressure();
altitud = sensorAmbiente.readFloatAltitudeMeters();
humedad = sensorAmbiente.readFloatHumidity();

// Refresco pantalla
pantallaBioBand.clearDisplay();

// Visualización parámetros ambientales
pantallaBioBand.setTextColor(NEGRO, BLANCO);

pantallaBioBand.drawBitmap(6, 20, temp_img, 6, 11, NEGRO);
pantallaBioBand.setCursor(21, 23);
pantallaBioBand.print("Temp:");
pantallaBioBand.print(temperatura, 2); pantallaBioBand.print("C");

pantallaBioBand.drawBitmap(6, 35, hum_img, 7, 10, NEGRO);
pantallaBioBand.setCursor(21, 37);
pantallaBioBand.print("Hum:");
pantallaBioBand.print(humedad, 2); pantallaBioBand.print("%");

pantallaBioBand.drawBitmap(3, 50, alt_img, 15, 10, NEGRO);
pantallaBioBand.setCursor(21, 52);
pantallaBioBand.print("Alt:");
pantallaBioBand.print(altitud, 2); pantallaBioBand.print("m");

pantallaBioBand.drawBitmap(5, 65, pres_img, 11, 12, NEGRO);
pantallaBioBand.setCursor(21, 68);
pantallaBioBand.print("Pre:");
pantallaBioBand.print(presión / 1000, 2);
pantallaBioBand.print("KPa");

delay(50);

// Control de vuelta a pantalla principal
if (botonCentral.wasPressed()) {
    paginaActual = 10;
    botonCentral.read(); // Comprobación de pulsación
}
}

/*****************************************/

```

```
*****  
Pantalla Movimiento  
*****  
  
//! ****  
//!     Nombre: pantallaMovimiento  
//!     Descripción: Pantalla de medición de parámetros asociados al  
//!                 acelerómetro  
//!     Parámetro entrada: void  
//!     Parámetro salida: void  
//!     Ejemplo: pantallaMovimiento();  
//! ****  
*****  
  
void pantallaMovimiento() {  
  
    // Definición logos  
    #define anchoIcono 11  
    #define altoIcono 12  
  
    static const unsigned char acc_img[] PROGMEM =  
    {  
        0x4, 0x0,  
        0xe, 0x0,  
        0x1f, 0x0,  
        0x4, 0x0,  
        0x4, 0x0,  
        0xe, 0x0,  
        0xa, 0x0,  
        0xe, 0x0,  
        0x11, 0x0,  
        0xa0, 0xa0,  
        0xc0, 0x60,  
        0xe0, 0xe0  
    };  
  
    // Fuerza el tiempo de Standby para que no vaya a la pantalla  
    // principal  
    tiempoStandby = millis() + tiempoActivo * 1000;  
  
    // Medición datos acelerómetro  
    acelerometro.stepCalc();  
  
    // Cálculo de pasos  
    long_paso = 0.415 * altura; //cm  
    calorias_milla = 0.57 * (peso * 2.2); //peso en kg  
    pasos_milla = 160934.4 / long_paso; //160934.3 una milla en cm  
    factor = calorias_milla / pasos_milla;  
    calorias_quemadas = acelerometro.stepCount * factor; //cal  
    distancia = (long_paso * acelerometro.stepCount) / 100; //m  
  
    // Cálculo aceleración  
    int valorGravedad = 30;  
    float acelX = (1.0*acelerometro.x)/valorGravedad;  
    float acelY = (1.0*acelerometro.y)/valorGravedad;  
    float acelZ = (1.0*acelerometro.z)/valorGravedad;
```

Pulsera biométrica Open Source para la monitorización del usuario

```
// Visualización datos podómetro
pantallaBioBand.setCursor(10, 10);
pantallaBioBand.clearDisplay();
pantallaBioBand.setTextSize(1);
pantallaBioBand.print("Pasos: ");
pantallaBioBand.print(accelerometro.stepCount);
pantallaBioBand.setCursor(10, 25);
pantallaBioBand.print("Dist: ");
pantallaBioBand.print(distancia);
pantallaBioBand.println(" m");
pantallaBioBand.setCursor(10, 40);
pantallaBioBand.print("Cal: ");
pantallaBioBand.print(calorias_quemadas);

// Visualización aceleraciones
pantallaBioBand.drawBitmap(10, 65, acc_img, anchoIcono, altoIcono,
NEGRO);
pantallaBioBand.setCursor(30, 56);
pantallaBioBand.print("x ");
pantallaBioBand.print(acelX);
pantallaBioBand.setCursor(30, 67);
pantallaBioBand.print("y ");
pantallaBioBand.print(acelY);
pantallaBioBand.setCursor(30, 78);
pantallaBioBand.print("z ");
pantallaBioBand.print(acelZ);

// Control de vuelta a pantalla principal
if (botonCentral.wasPressed()) {
    accelerometro.stepCount = 0;
    paginaActual = 10;
    botonCentral.read(); // Comprobación de pulsación
}
}

/********************************************/
```

```
*****  
Pantalla Cronómetro  
*****  
  
//! ****  
//!     Nombre: pantallaCronometro  
//!     Descripción: Pantalla de función cronómetro  
//!     Parámetro entrada: void  
//!     Parámetro salida: void  
//!     Ejemplo: pantallaCronometro();  
//! ****  
  
void pantallaCronometro() {  
  
    // Definición logos  
    #define anchoIcono 10  
    #define altoIcono 10  
  
    static const unsigned char play_img[] PROGMEM =  
    {  
        0xc0, 0x0,  
        0xf0, 0x0,  
        0xfc, 0x0,  
        0xff, 0x0,  
        0xff, 0x80,  
        0xff, 0x80,  
        0xff, 0x0,  
        0xfc, 0x0,  
        0xf0, 0x0,  
        0xc0, 0x0  
    };  
  
    static const unsigned char pause_img[] PROGMEM =  
    {  
        0x73, 0x80,  
        0x73, 0x80  
    };  
  
    static const unsigned char stop_img[] PROGMEM =  
    {  
        0xff, 0xc0,  
        0xff, 0xc0  
    };
```

Pulsera biométrica Open Source para la monitorización del usuario

```
// Fuerza el tiempo de Standby para que no vaya a la pantalla
principal
tiempoStandby = millis() + tiempoActivo * 1000;

// Control del cronómetro
pantallaBioBand.drawBitmap(80, 80, stop_img, anchoIcono, altoIcono,
NEGRO);

if (botonArriba.wasPressed()) {
    paradaRelojActivo = !paradaRelojActivo;
    if (paradaRelojActivo) {
        pantallaBioBand.drawBitmap(80, 80, play_img, anchoIcono,
altoIcono, NEGRO);
    }
    else
        pantallaBioBand.drawBitmap(80, 80, pause_img, anchoIcono,
altoIcono, NEGRO);
}

if (botonAbajo.wasPressed()) {
    paradaRelojTimer = millis();
    paradaRelojMs = 0;
    pantallaBioBand.drawBitmap(80, 80, stop_img, anchoIcono,
altoIcono, NEGRO);
}

if (!paradaRelojActivo) paradaRelojTimer = millis() - paradaRelojMs;
paradaRelojMs = millis() - paradaRelojTimer;

// Refresco pantalla
pantallaBioBand.clearDisplay();

// Visualización cronómetro
dibujarDigito(2, 20, ((paradaRelojMs / 60000) / 10), 0);
dibujarDigito(24, 20, ((paradaRelojMs / 60000) % 10), 0);
dibujarDigito(52, 20, (paradaRelojMs / 1000 % 60) / 10, 0);
dibujarDigito(74, 20, (paradaRelojMs / 1000 % 60) % 10, 0);

pantallaBioBand.setCursor(4, 63);
pantallaBioBand.setTextColor(NEGRO, BLANCO);
pantallaBioBand.setTextSize(2);

if (paradaRelojMs % 1000 < 10) pantallaBioBand.print(0);
if (paradaRelojMs % 1000 < 100) pantallaBioBand.print(0);

pantallaBioBand.print(paradaRelojMs % 1000);
pantallaBioBand.print("ms");
pantallaBioBand.fillRect(46, 30, 4, 4, 0);
pantallaBioBand.fillRect(46, 46, 4, 4, 0);

// Control de vuelta a pantalla principal
if (botonCentral.wasPressed()) {
    paradaRelojActivo = false;
    paradaRelojMs = 0;
    paginaActual = 10;
    botonCentral.read(); // Comprobación de pulsación
}
}

/********************************************/
```

```
*****  
Pantalla Ajustes  
*****  
  
//! ****  
//!     Nombre: pantallaAjustes  
//!     Descripción: Pantalla de ajustes de parámetros de generales de  
//!                     navegación  
//!     Parámetro entrada: void  
//!     Parámetro salida: void  
//!     Ejemplo: pantallaAjustes();  
//! ****  
  
void pantallaAjustes() {  
  
    // Fuerza el tiempo de Standby para que no vaya a la pantalla  
    principal  
    tiempoStandby = millis() + tiempoActivo * 1000;  
  
    // Variable de control de ajustes  
    static int valorAjustes = 0;  
    if (botonCentral.wasPressed()) valorAjustes++;  
  
    // Refresco pantalla  
    pantallaBioBand.clearDisplay();  
  
    pantallaBioBand.setCursor(0, 2);  
  
    // Visualización de ajuste de tiempo activo  
    if (valorAjustes == 0) {  
        if (botonArriba.wasPressed() || botonArriba.pressedFor(500))  
            tiempoActivo++;  
        if (botonAbajo.wasPressed() || botonAbajo.pressedFor(500))  
            tiempoActivo--;  
        pantallaBioBand.setTextColor(BLANCO, NEGRO);  
        if (tiempoActivo > 60) tiempoActivo = 30;  
        else if (tiempoActivo < 5) tiempoActivo = 5;  
    }  
    else pantallaBioBand.setTextColor(NEGRO, BLANCO);  
  
    // Visualización de ajuste de tiempo activo  
    pantallaBioBand.print(F("Standby: "));  
    pantallaBioBand.print(tiempoActivo);  
    pantallaBioBand.println("s");  
  
    // Control de ajuste de indicación de voltaje  
    if (valorAjustes == 1) {  
        if (botonArriba.wasPressed() || botonAbajo.wasPressed())  
            visualizacionVoltaje = !visualizacionVoltaje;  
        pantallaBioBand.setTextColor(BLANCO, NEGRO);  
    }  
    else pantallaBioBand.setTextColor(NEGRO, BLANCO);  
}
```

Pulsera biométrica Open Source para la monitorización del usuario

```
// Visualización de ajuste de indicación de voltaje
pantallaBioBand.print(F("Voltaje: "));
if (visualizacionVoltaje) {
    pantallaBioBand.println("ON");
}
else {
    pantallaBioBand.println("OFF");
}

// Control de ajuste de peso
if (valorAjustes == 2) {
    if (botonArriba.wasPressed() || botonArriba.pressedFor(500))
        peso++;
    if (botonAbajo.wasPressed() || botonAbajo.pressedFor(500)) peso--;
    pantallaBioBand.setTextColor(BLANCO, NEGRO);
}
else pantallaBioBand.setTextColor(NEGRO, BLANCO);

// Visualización de ajuste de peso
pantallaBioBand.print(F("Peso: "));
pantallaBioBand.print(peso);
pantallaBioBand.println("kg");

// Control de ajuste de altura
if (valorAjustes == 3) {
    if (botonArriba.wasPressed() || botonArriba.pressedFor(500))
        altura++;
    if (botonAbajo.wasPressed() || botonAbajo.pressedFor(500))
        altura--;
    pantallaBioBand.setTextColor(BLANCO, NEGRO);
}
else pantallaBioBand.setTextColor(NEGRO, BLANCO);

// Visualización de ajuste de altura
pantallaBioBand.print(F("Altura: "));
pantallaBioBand.print(altura);
pantallaBioBand.println("cm");

// Control y visualización de ajuste de general
pantallaBioBand.setTextColor((valorAjustes == 4), (valorAjustes != 4));
pantallaBioBand.println(F("OK"));

// Control de vuelta a pantalla principal
if (botonCentral.wasPressed() && valorAjustes == 5) {
    valorAjustes = 0;
    paginaActual = 10;
    menuVal = 4;
    botonCentral.read(); // Comprobación de pulsación
}
}

/*****************************************/
```

```
*****  
Pantalla Juego  
*****  
  
//! ****  
//!     Nombre: pantallaJuego  
//!     Descripción: Pantalla de juego similar a "Flappy bird"  
//!     Parámetro entrada: void  
//!     Parámetro salida: void  
//!     Ejemplo: pantallaJuego();  
//! ****  
  
void pantallaJuego() {  
  
    // Fuerza el tiempo de Standby para que no vaya a la pantalla  
    principal  
    tiempoStandby = millis() + tiempoActivo * 1000;  
  
    // Refresco pantalla  
    pantallaBioBand.clearDisplay();  
  
    //Control juego  
    static boolean activacionJuego = true;  
    static float velocidadY = 0;  
    static int py = 0;  
    static boolean gameOver = false;  
    static byte puntuacion = 0;  
    static int posicionMuro[3] = {100, 143, 186};  
    static int agujeroMuro[3] = {40, 60, 0};  
    static unsigned long ultimoTiempo = millis();  
  
    if (activacionJuego) {  
        velocidadY = py = puntuacion = 0;  
        posicionMuro[0] = 100;  
        posicionMuro[1] = 143;  
        posicionMuro[2] = 186;  
        ultimoTiempo = millis();  
        gameOver = false;  
    }  
  
    float deltaTime = float(millis() - ultimoTiempo);  
  
    velocidadY += deltaTime / 80;  
    py += velocidadY;  
  
    // Visualización juego escenario  
    for (int i = 0; i < 3; i++) { // draw walls  
        pantallaBioBand.fillRect(posicionMuro[i] - 10, 0, 10,  
        agujeroMuro[i], 0);  
        pantallaBioBand.fillRect(posicionMuro[i] - 10, agujeroMuro[i] +  
        30, 10, 96, 0);  
  
        // Detección muro  
        if (posicionMuro[i] > 5 && posicionMuro[i] < 25) {  
            // Detección agujero  
            if (agujeroMuro[i] > py - 5 || agujeroMuro[i] < py - 25)  
                gameOver = true;  
        }  
    }  
}
```

Pulsera biométrica Open Source para la monitorización del usuario

```
// Reinicio muro
if (posicionMuro[i] <= 0) {
    posicionMuro[i] += 129;
    agujeroMuro[i] = random(5, 70);
    puntuacion++;
}
posicionMuro[i] -= deltaTime / 80; // move walls
}

// Visualización personaje
py = constrain(py, 5, 91);
pantallaBioBand.fillCircle(10, py, 5, 0);

// Visualización puntuación
pantallaBioBand.setTextColor(NEGRO, BLANCO);
pantallaBioBand.setCursor(40, 2);
pantallaBioBand.print(F("puntuacion: "));
pantallaBioBand.println(puntuacion);

if (botonArrriba.isPressed()) velocidadY = -3;
ultimoTiempo = millis();

// Visualización fin del juego
if (gameOver) {
    activacionJuego = true;
    pantallaBioBand.clearDisplay();
    pantallaBioBand.setCursor(20, 30);
    pantallaBioBand.println(F("GAME OVER"));
    pantallaBioBand.setCursor(20, 40);
    pantallaBioBand.print(F("puntuacion: "));
    pantallaBioBand.println(puntuacion);
    pantallaBioBand.refresh();
    delay(3000);
}
else activacionJuego = false;

// Control de vuelta a pantalla principal
if (botonCentral.wasPressed()) {
    activacionJuego = true;
    paginaActual = 10;
    botonCentral.read(); // Comprobación de pulsación
}
}

/*****************************************/
```

```
*****  
Pantalla Iluminación  
*****  
  
//! ****  
//!     Nombre: pantallaIluminacion  
//!     Descripción: Pantalla de medición y control de iluminación  
//!     Parámetro entrada: void  
//!     Parámetro salida: void  
//!     Ejemplo: pantallaIluminacion();  
//! ****  
  
void pantallaIluminacion() {  
  
    static const unsigned char lint_img[] PROGMEM =  
    {  
        0x0, 0x0, 0x78,  
        0x0, 0x1, 0xc8,  
        0x0, 0x7, 0x48,  
        0xff, 0xfe, 0x48,  
        0xff, 0xf8, 0x48,  
        0x80, 0x0, 0x48,  
        0x80, 0x0, 0x48,  
        0x81, 0xe0, 0x48,  
        0x80, 0x0, 0x48,  
        0x80, 0x0, 0x48,  
        0xff, 0xf8, 0x48,  
        0xff, 0xfe, 0x48,  
        0x0, 0x7, 0x48,  
        0x0, 0x1, 0xc8,  
        0x0, 0x0, 0x78  
    };  
    static const unsigned char luz_img[] PROGMEM =  
    {  
        0x1, 0x0,  
        0x41, 0x4,  
        0x21, 0x8,  
        0x10, 0x10,  
        0x3, 0x80,  
        0x6, 0xc0,  
        0xc, 0x60,  
        0xe8, 0x2e,  
        0xc, 0x60,  
        0x6, 0xc0,  
        0x3, 0x80,  
        0x10, 0x10,  
        0x21, 0x8,  
        0x41, 0x4,  
        0x1, 0x0  
    };  
  
    // Refresco pantalla  
    pantallaBioBand.clearDisplay();  
  
    // Control iluminación  
    pantallaBioBand.drawBitmap(10, 20, lint_img, 21, 15, NEGRO);  
    pantallaBioBand.drawBitmap(10, 53, luz_img, 15, 15, NEGRO);  
}
```

Pulsera biométrica Open Source para la monitorización del usuario

```
if (botonArriba.wasPressed()) menuIluminacion--;
if (botonAbajo.wasPressed()) menuIluminacion++;

if (menuIluminacion > 3 && menuIluminacion < 99) menuIluminacion =
1;
else if (menuIluminacion > 99) menuIluminacion = 3;

// Visualización iluminación
estadoIluminacion = menuIluminacion;
pantallaBioBand.setTextColor(NEGRO, BLANCO);

pantallaBioBand.setCursor(30, 57);
pantallaBioBand.print(lux);
pantallaBioBand.println("lux");

switch (estadoIluminacion) {
    case 1:
        boolean good;
        pantallaBioBand.setCursor(37, 24);
        pantallaBioBand.print(F("Auto"));
        break;
    case 2:
        pantallaBioBand.setCursor(37, 24);
        pantallaBioBand.print(F("On"));
        break;
    case 3:
        pantallaBioBand.setCursor(37, 24);
        pantallaBioBand.print(F("Off"));
        break;
}

// Control de vuelta a pantalla principal
if (botonCentral.wasPressed()) {
    paginaActual = 10;
    botonCentral.read(); // Comprobación de pulsación
}
}

/****************************************/
*****
```

```
*****  
Pantalla Reloj  
*****  
  
//! ****  
//!     Nombre: pantallaAjustes  
//!     Descripción: Pantalla de ajustes de parámetros de generales de  
//!                     navegación  
//!     Parámetro entrada: void  
//!     Parámetro salida: void  
//!     Ejemplo: pantallaAjustes();  
//! ****  
  
void pantallaReloj() {  
  
    // Refresco pantalla  
    pantallaBioBand.clearDisplay();  
  
    // Visualización de la hora  
    dibujarDigito( 2, 20, d_hora, 0);  
    dibujarDigito(24, 20, u_hora, 0);  
    dibujarDigito(52, 20, d_minuto, 0);  
    dibujarDigito(74, 20, u_minuto, 0);  
  
    pantallaBioBand.fillRect(46, 30, 4, 4, 0);  
    pantallaBioBand.fillRect(46, 46, 4, 4, 0);  
  
    pantallaBioBand.setCursor(4, 63);  
    pantallaBioBand.setTextColor(NEGRO, BLANCO);  
    pantallaBioBand.setTextSize(2);  
    pantallaBioBand.print(d_segundo);  
    pantallaBioBand.print(u_segundo);  
  
    // Control de la hora  
    if (botonArriba.wasPressed() || botonArriba.pressedFor(500)) {  
        u_minuto++;  
    }  
    if (botonAbajo.wasPressed() || botonAbajo.pressedFor(500)) {  
        u_hora++;  
    }  
  
    // Control de vuelta a pantalla principal  
    if (botonCentral.wasPressed()) {  
        paginaActual = 10;  
        botonCentral.read(); // Comprobación de pulsación  
    }  
}  
*****
```

Pulsera biométrica Open Source para la monitorización del usuario

```
*****  
Pantalla Notificaciones  
*****  
  
//! ****  
//!     Nombre: pantallaNotificaciones  
//!     Descripción: Pantalla de indicación de notificaciones del  
//!                     teléfono  
//!     Parámetro entrada: void  
//!     Parámetro salida: void  
//!     Ejemplo: pantallaNotificaciones();  
//! ****  
  
void pantallaNotificaciones() {  
  
    // Definición logos  
    static const unsigned char fb_img[] PROGMEM =  
    {  
        0xf, 0x0,  
        0x3f, 0xc0,  
        0x78, 0xe0,  
        0x7b, 0xe0,  
        0xfb, 0xf0,  
        0xf0, 0xf0,  
        0xfb, 0xf0,  
        0xfb, 0xf0,  
        0x7b, 0xe0,  
        0x7b, 0xe0,  
        0x3f, 0xc0,  
        0xf, 0x0  
    };  
  
    static const unsigned char twit_img[] PROGMEM =  
    {  
        0xf, 0x0,  
        0x3f, 0xc0,  
        0x7f, 0xe0,  
        0x73, 0xe0,  
        0xf0, 0xf0,  
        0xf0, 0xf0,  
        0xf3, 0xf0,  
        0xf0, 0xf0,  
        0x70, 0xe0,  
        0x7f, 0xe0,  
        0x3f, 0xc0,  
        0xf, 0x0  
    };  
}
```

```
static const unsigned char text_img[] PROGMEM =
{
    0xf, 0x0,
    0x3f, 0xc0,
    0x7f, 0xe0,
    0x40, 0x20,
    0xcf, 0x30,
    0xd6, 0xb0,
    0xd9, 0xb0,
    0xdf, 0xb0,
    0x40, 0x20,
    0x7f, 0xe0,
    0x3f, 0xc0,
    0xf, 0x0
};

static const unsigned char call_img[] PROGMEM =
{
    0xf, 0x0,
    0x3f, 0xc0,
    0x7f, 0xe0,
    0x77, 0xe0,
    0xe7, 0xf0,
    0xe7, 0xf0,
    0xf3, 0xf0,
    0xf8, 0x70,
    0x7c, 0xe0,
    0x7f, 0xe0,
    0x3f, 0xc0,
    0xf, 0x0
};

static const unsigned char what_img[] PROGMEM =
{
    0x1f, 0x80,
    0x70, 0xe0,
    0x40, 0x20,
    0xc8, 0x30,
    0x98, 0x10,
    0x98, 0x10,
    0x8c, 0x10,
    0x87, 0x90,
    0xc3, 0x30,
    0x40, 0x20,
    0xb0, 0xe0,
    0xdf, 0x80
};

// Refresco pantalla
pantallaBioBand.clearDisplay();
```

Pulsera biométrica Open Source para la monitorización del usuario

```
// Visualización notificaciones
pantallaBioBand.drawBitmap(13, 5, fb_img, 12, 12, NEGRO);
pantallaBioBand.setCursor(30, 6);
pantallaBioBand.print("0");
pantallaBioBand.drawBitmap(13, 20, twit_img, 12, 12, NEGRO);
pantallaBioBand.setCursor(30, 21);
pantallaBioBand.print("0");
pantallaBioBand.drawBitmap(13, 35, what_img, 12, 12, NEGRO);
pantallaBioBand.setCursor(30, 36);
pantallaBioBand.print("0");
pantallaBioBand.drawBitmap(13, 50, text_img, 12, 12, NEGRO);
pantallaBioBand.setCursor(30, 51);
pantallaBioBand.print("0");
pantallaBioBand.drawBitmap(13, 65, call_img, 12, 12, NEGRO);
pantallaBioBand.setCursor(30, 66);
pantallaBioBand.print("0");

// Control de vuelta a pantalla principal
if (botonCentral.wasPressed()) {
    paginaActual = 10;
    botonCentral.read(); // Comprobación de pulsación
}
}

/***** /
```

ANEXO 8. Esquemático final

Los ficheros asociados a esquemas electrónicos del proyecto se han realizado con el software Cadsoft Eagle. Se adjuntan a continuación las capturas correspondientes a las hojas del esquema final del proyecto.

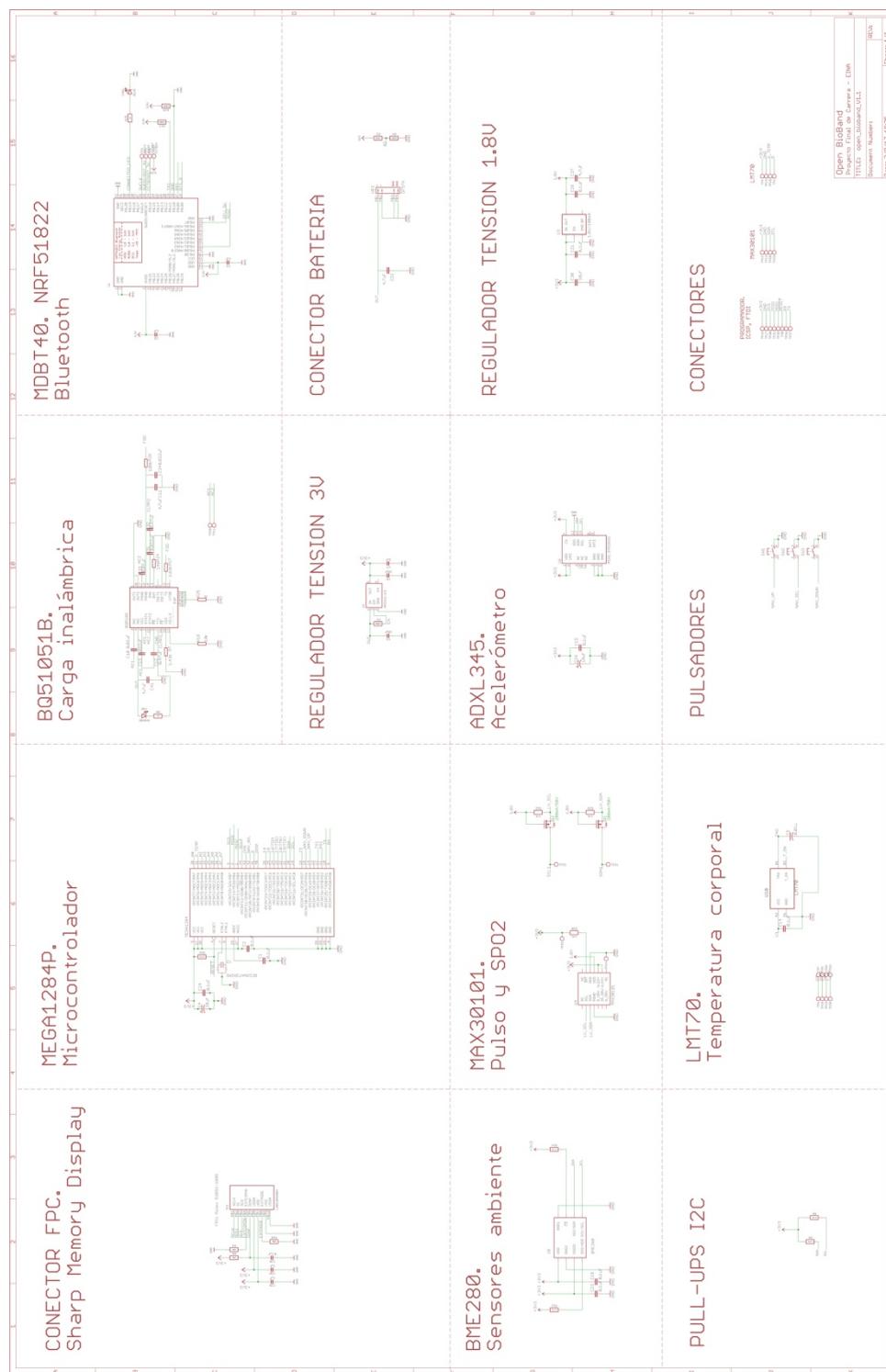


Fig. 109 Esquemático Open Bioband

ANEXO 9. Listado de componentes: BOM

Se incluye el listado clasificado de todos los componentes necesarios para la fabricación del diseño final de pulsera Open Bioband.

Ingeniería Industrial con especialización en Electrónica
Universidad de Zaragoza

Identificación	Valor	Componente	Encapsulado	Descripción
BQ510X	BQ51051	BQ51051	R-PQFP-N20	Charge Circuit
C1	0.1uF	CAP0402-CAF	0402-CAP	Capacitor
C2	0.1uF	CAP0402-CAF	0402-CAP	Capacitor
C3	1uF	CAP0402-CAF	0402-CAP	Capacitor
C4	10uF	CAP_POL120€	EIA3216	Capacitor Polarized
C5	0.1uF	CAP0402-CAF	0402-CAP	Capacitor
C6	10µF	CAP0402-CAF	0402-CAP	Capacitor
C7	1uF	CAP0402-CAF	0402-CAP	Capacitor
C8	10µF	CAP0402-CAF	0402-CAP	Capacitor
C9	1uF	CAP0402-CAF	0402-CAP	Capacitor
C10	0.01uF	CAP0402-CAF	0402-CAP	Capacitor
C11	4.7uF	CAP0402-CAF	0402-CAP	Capacitor
C12	0.47uF	CAP0402-CAF	0402-CAP	Capacitor
C13	0.022uF	CAP0402-CAF	0402-CAP	Capacitor
C14	4.7uF	CAP0402-CAF	0402-CAP	Capacitor
C15	0.1uF	CAP0402-CAF	0402-CAP	Capacitor
C16	10uF	CAP_POL120€	EIA3216	Capacitor Polarized
C17	0.1uF	CAP0402-CAF	0402-CAP	Capacitor
C18	0.1uF	CAP0402-CAF	0402-CAP	Capacitor
C19	10uF	CAP0402-CAF	0402-CAP	Capacitor
C20	0.1uF	CAP0402-CAF	0402-CAP	Capacitor
C22	4.7uF	CAP0402-CAF	0402-CAP	Capacitor
C23	0.1uF	CAP0402-CAF	0402-CAP	Capacitor
C24	0.1uF	CAP0402-CAF	0402-CAP	Capacitor
C25	0.1µF	CAP0402-CAF	0402-CAP	Capacitor
C28	1µF	CAP0402-CAF	0402-CAP	Capacitor
C29	1µF	CAP0402-CAF	0402-CAP	Capacitor
C31	0.022uF	CAP0402-CAF	0402-CAP	Capacitor
C32	0.47uF	CAP0402-CAF	0402-CAP	Capacitor
C33	0.01uF	CAP0402-CAF	0402-CAP	Capacitor
C34	0.022uF	CAP0402-CAF	0402-CAP	Capacitor
C41	4.7uF	CAP0402-CAF	0402-CAP	Capacitor
CHG	ORANGE	LED0603	CHIPLED_060Ξ	LED
CHG1	BLUE	LED0603	CHIPLED_060Ξ	LED
IC2	LS013B4DN02	LS013B4DN02	FPC_10PIN_52892-1095	Sharp Mono Memory LCE
MEGA1284		ATMEGA1284	TQFP44	8-bit Microcontroller with 64K Bytes
Q1	200mA/50V	BSS138	SOT23-3	Common NMOSFET Part
Q2	200mA/50V	BSS138	SOT23-3	Common NMOSFET Part
R1	100k	RESISTOR_0402	_0402	Resistors
R2	100k	RESISTOR_0402	_0402	Resistors
R3	10k	RESISTOR_0402	_0402	Resistors
R4	100K	RESISTOR_0402	_0402	Resistors
R5	4.7K	RESISTOR_0402	_0402	Resistors
R6	10k	RESISTOR_0402	_0402	Resistors
R7	2.43K	RESISTOR_0402	_0402	Resistors
R8	1K	RESISTOR_0402	_0402	Resistors
R9	4.7K	RESISTOR_0402	_0402	Resistors
R10	10k	RESISTOR_0402	_0402	Resistors
R12	4.7K	RESISTOR_0402	_0402	Resistors
R13	100K	RESISTOR_0402	_0402	Resistors
R14	4.7K	RESISTOR_0402	_0402	Resistors
R15	1k	RESISTOR_0402	_0402	Resistors
R16	4.7K	RESISTOR_0402	_0402	Resistors
R17	3.83K	RESISTOR_0402	_0402	Resistors
R18	10k	RESISTOR_0402	_0402	Resistors
R19	100K	RESISTOR_0402	_0402	Resistors
R20	4.7K	RESISTOR_0402	_0402	Resistors
R22	4.7K	RESISTOR_0402	_0402	Resistors
R23	4.7K	RESISTOR_0402	_0402	Resistors
R24	140	RESISTOR_0402	_0402	Resistors
R25	0	RESISTOR_0402	_0402	Resistors
R26	100k	RESISTOR_0402	_0402	Resistors
U\$3	DF57H	DF57H	DF57H-2P	
U1		NRF51822_MODULE_MDBT4C	BLE_MODULE_RAYTAC_MDBT4C	nRF51822 Bluetooth Low Energy Modul
U2	AP2112-3.3	VREG_SOT23-5	SOT23-5	SOT23-5 Fixed Voltage Regulator
U4	MAX30101	MAX30101	OLGA-14	Particle Detection IC
U5	1.8V/100mA	V_REG_SP62141.8V	SC70	
U6	ADXL345ORIC	ADXL345ORIC	LGA14	
U7	TSL2561FN	TSL2561FN	FN-6	TSL2561 illumination senso
U8	BME280	BME280	BME280_LGA	
Y2	RESONATORSMC	RESONATORSMC	RESONATOR-SMC	Resonator
C26	0.01u	C-USC0402	C0402	CAPACITOR, American symb
C27	0.1uF	0.1UF-0402-16V-10%	402	0.1µF ceramic capacitor
U10	LMT70LMT70	LMT70LMT70	BGA4C40P2X2_77X77X5(LOW-POWER SINGLE INVERTER GATE

Tabla 43. Listado componentes PCB

ANEXO 10. Diseño PCB final

Los ficheros Gerber generados durante la fase de diseño poseen un formato estándar que aceptan todos los fabricantes y por lo tanto no debería de suponer un problema replicar el prototipo en ningún aspecto.

Además, de cara a su fabricación se establecieron unas pautas de cara a las características de los prototipos y las posibilidades de fabricación de los distintos fabricantes:

Tecnología de fabricación PCB y requerimientos			
PCB Nombre y versión	Open BioBand V1		
Material PCB :	<input checked="" type="checkbox"/> Rígido)FR-4 <input type="checkbox"/> Flex		
Tamaño PCB :	<input type="checkbox"/> cm <input type="checkbox"/> mm <input type="checkbox"/> inch	R36	
Capas PCB :	<input checked="" type="checkbox"/> 2	<input type="checkbox"/> 4	<input type="checkbox"/> other:()
	<input type="checkbox"/> 0.8mm <input type="checkbox"/> 1.2mm <input checked="" type="checkbox"/> 1mm <input type="checkbox"/> 1.6mm	<input type="checkbox"/> Otros	
Grosor de cobre :	<input type="checkbox"/> 1oz <input type="checkbox"/> 2oz	<input checked="" type="checkbox"/> Otros	
Color PCB :	<input type="checkbox"/> Verde <input type="checkbox"/> Blanco	<input type="checkbox"/> Negro	<input type="checkbox"/> Negro
	<input type="checkbox"/> Rojo <input type="checkbox"/> Azul	<input type="checkbox"/> Otros	<input type="checkbox"/> Otros
Serigrafía PCB :	<input checked="" type="checkbox"/> Blanco	<input type="checkbox"/> Negro	<input type="checkbox"/> Otros
Acabado PADs :	<input checked="" type="checkbox"/> Hasl-lead free	<input type="checkbox"/> Hasl	<input type="checkbox"/> ENIG
Máscara de soldadura :	<input type="checkbox"/> Activada	<input type="checkbox"/> Desactivada	
Distancia mínima	Fabricante	Agujero mínimo	Fabricante
Requerimientos especiales	<input type="checkbox"/> Yes		
Comentarios:			

Tabla 44. Requerimientos fabricación PCB

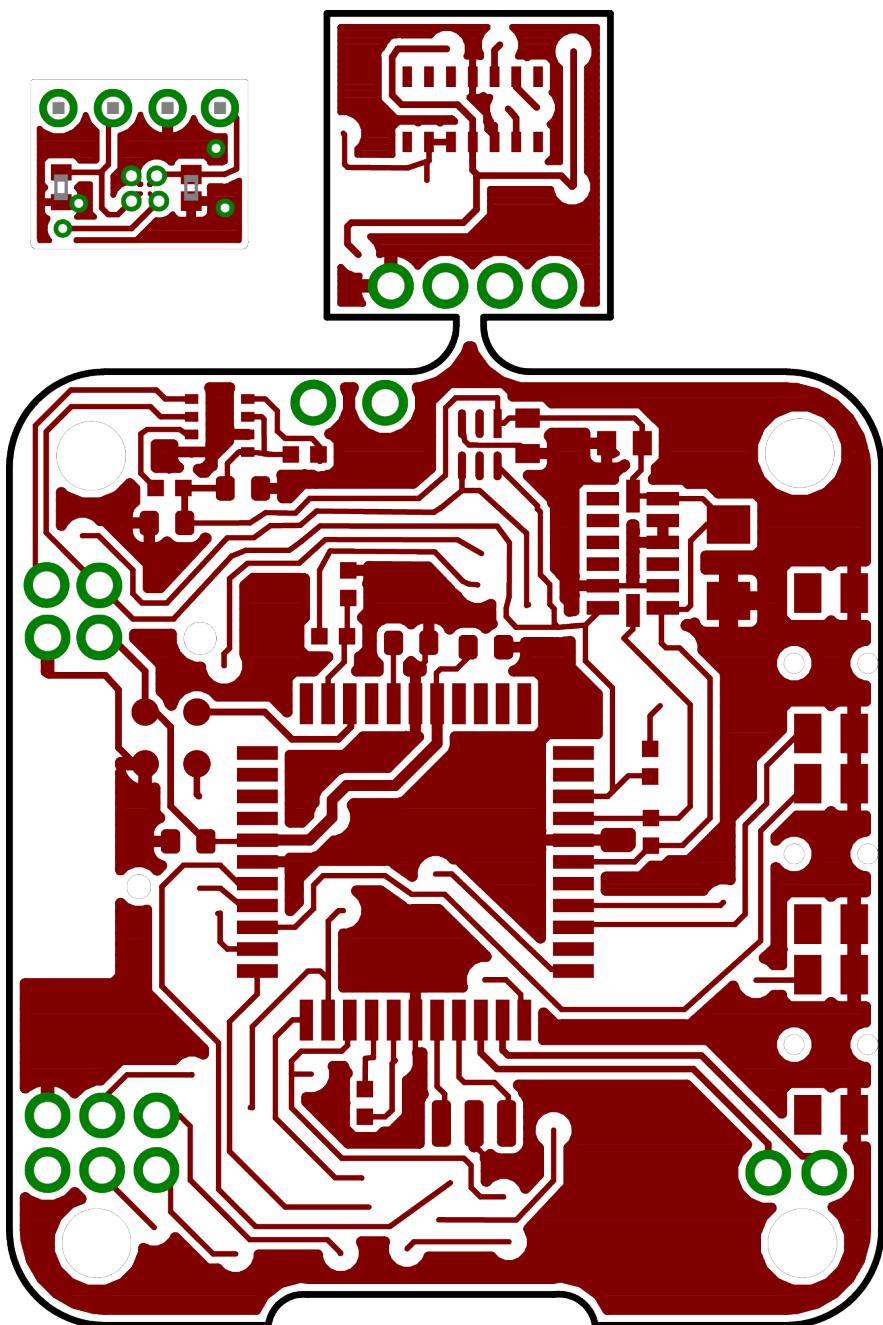


Fig. 110 Plano pistas cara TOP

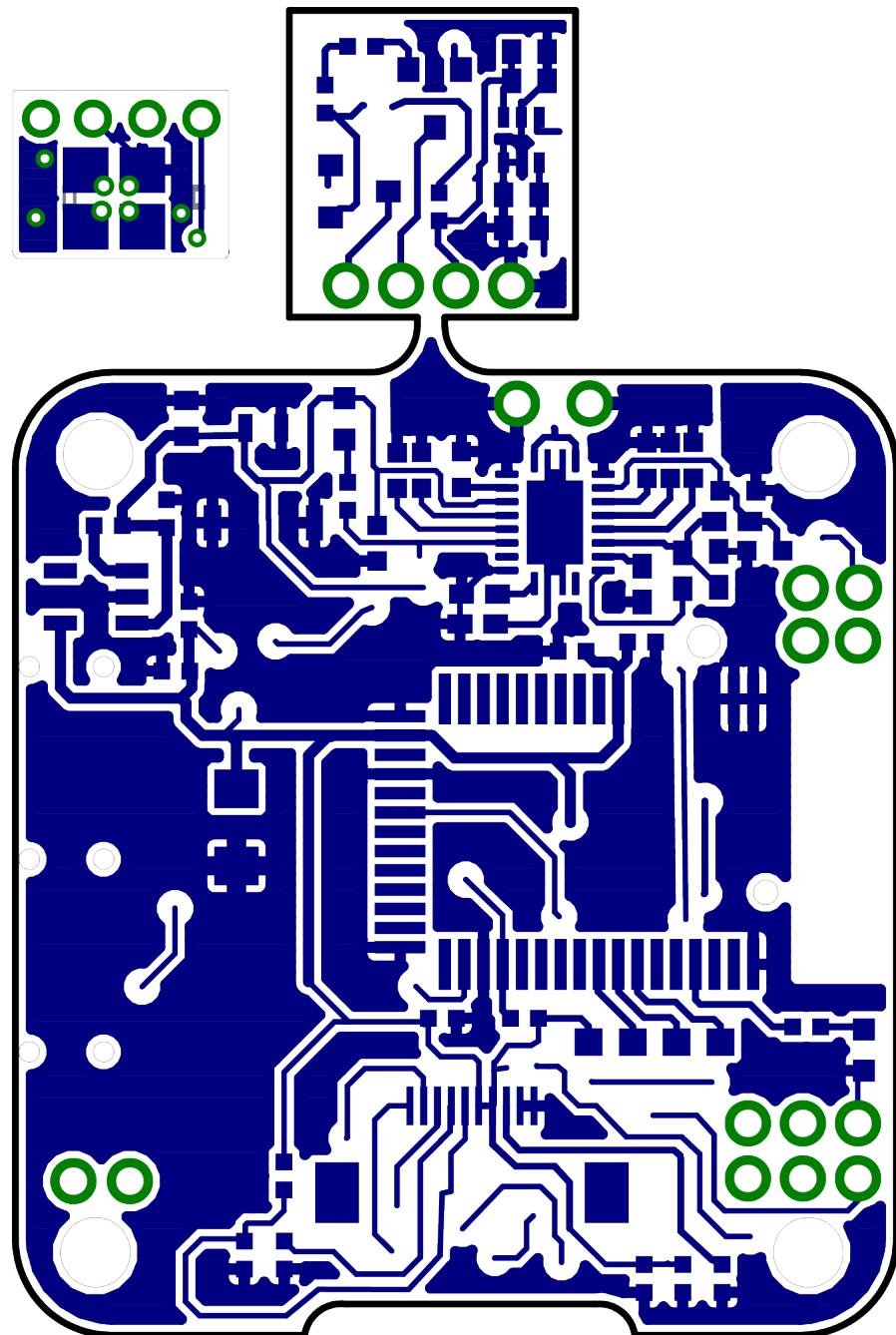


Fig. 111 Plano pistas cara BOTTOM

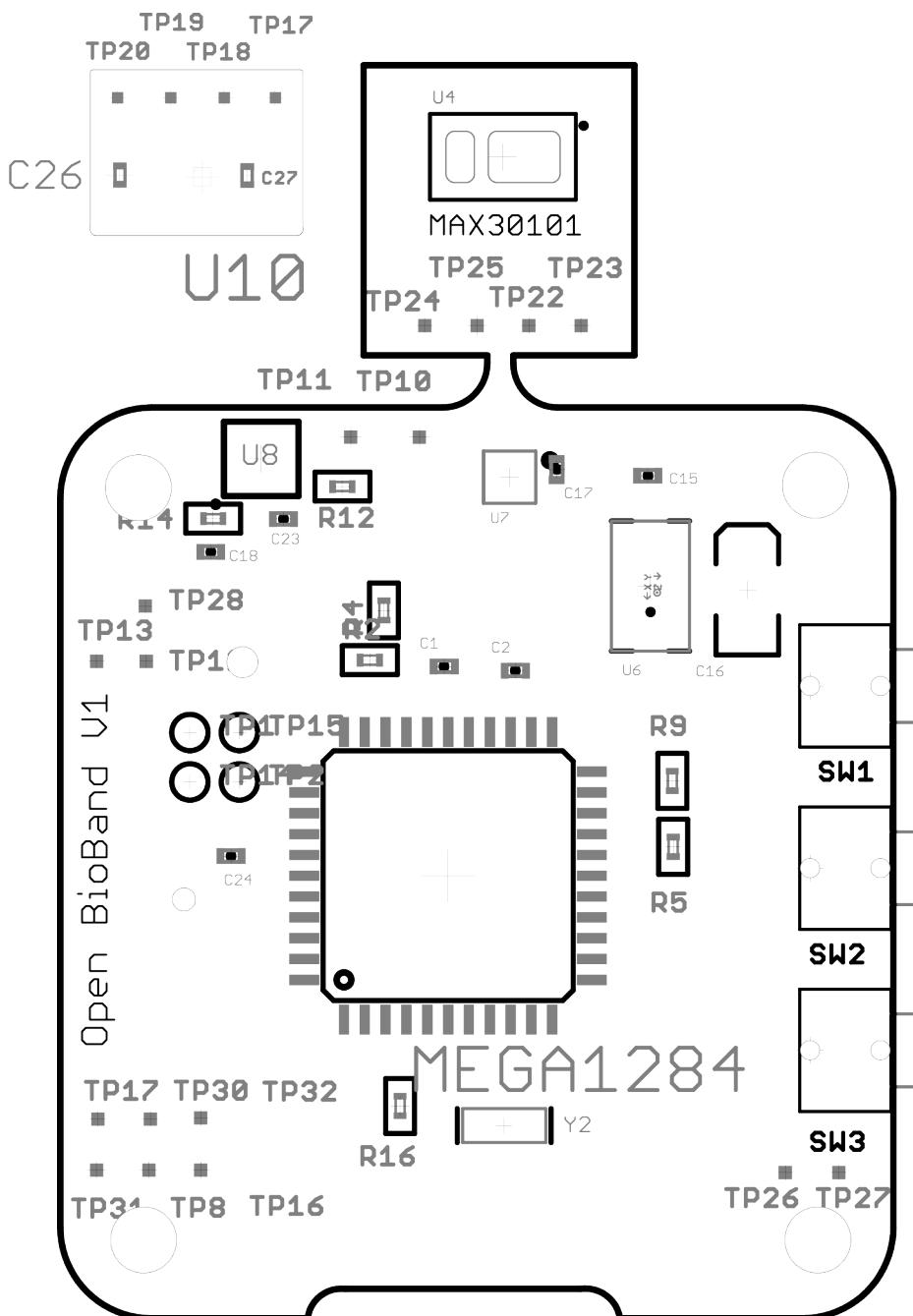


Fig. 112 Plano componentes cara TOP

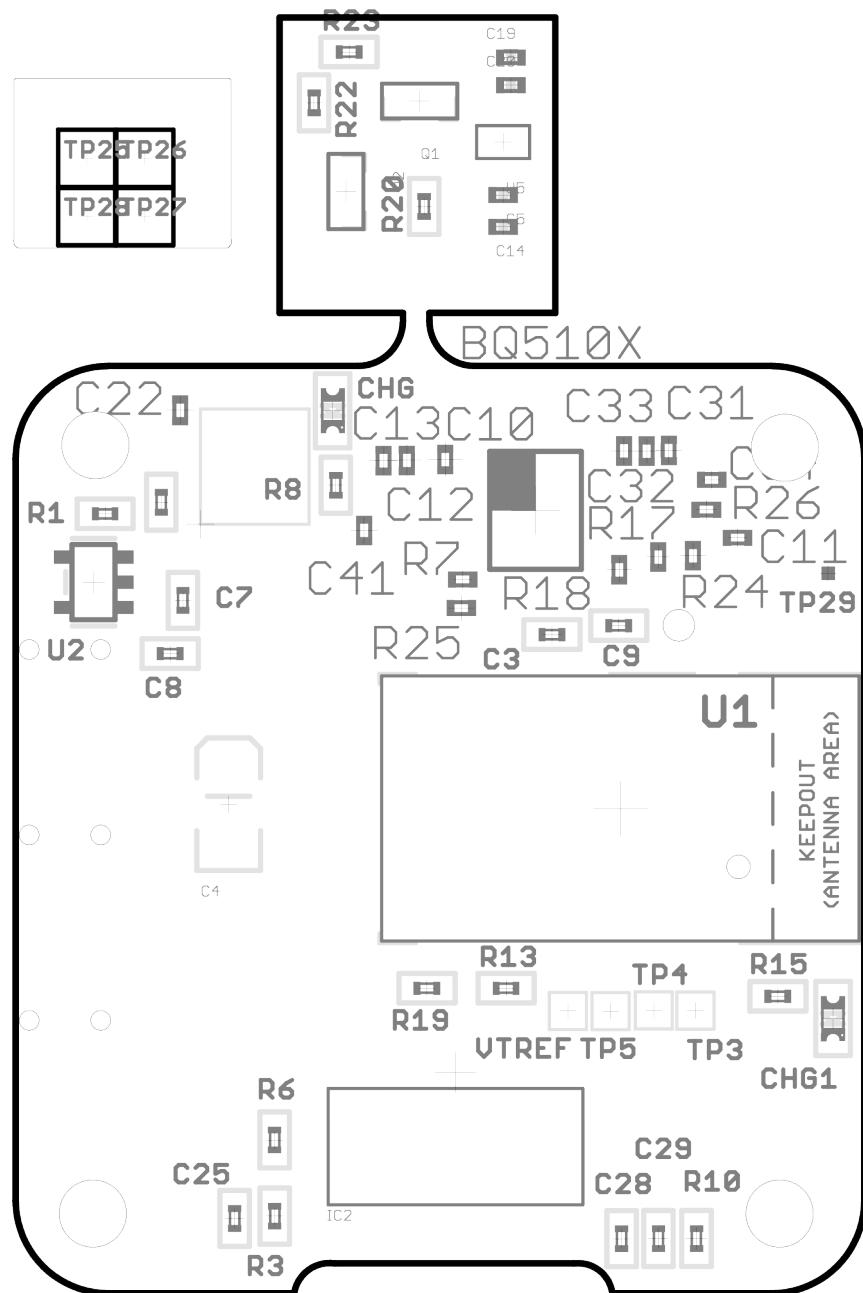


Fig. 113 Plano componentes cara BOTTOM

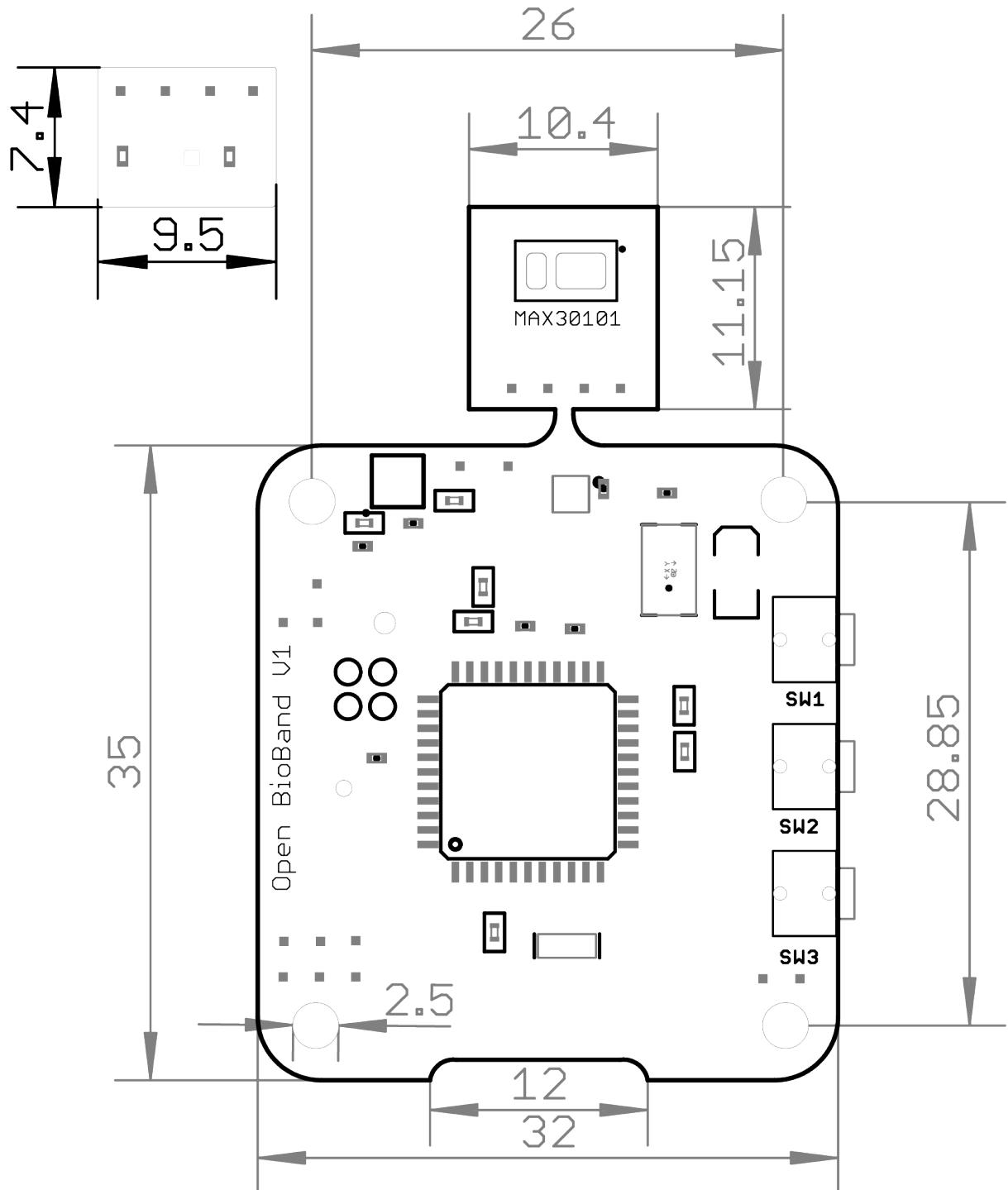
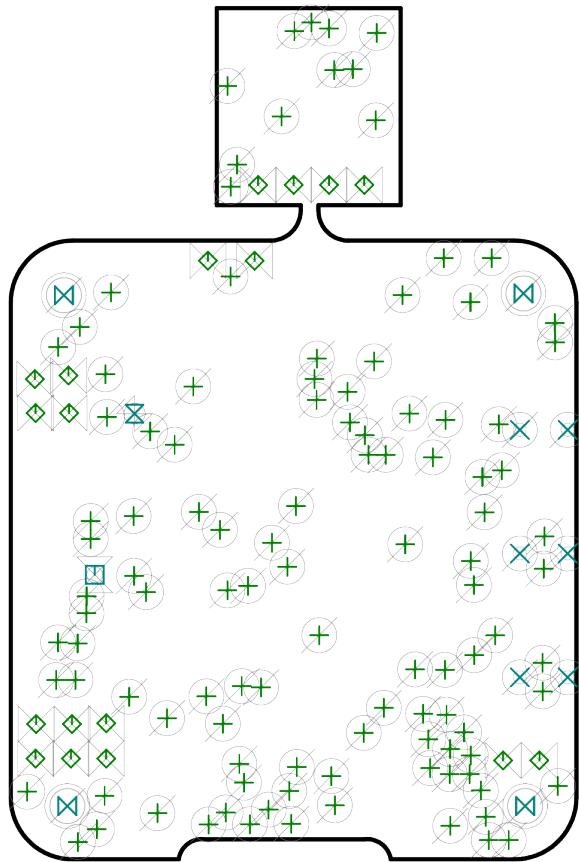
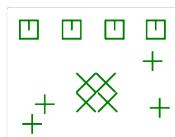


Fig. 114 Plano de dimensiones



LAYER-STACK
01-16
01-20

Sym	Nº	Mils	MM	Qty	Plated
+	1	16	0.40	111	YES
×	2	30	0.75	6	NOT
□	3	35	0.90	1	NOT
◊	4	39	1.00	18	YES
✗	5	47	1.20	1	NOT
⌘	6	100	2.54	4	NOT



LAYER-STACK
01-16

Sym	Nº	Mils	MM	Qty	Plated
+	1	16	0.40	4	YES
×	2	20	0.50	4	YES
□	3	39	1.00	4	YES

Fig. 115 Plano de taladrado

ANEXO 11. Costes fabricación

Uno de los requisitos propuesto en el proyecto, además de la adquisición de señales tanto ambientales como biométricas y la comunicación inalámbrica, era el de crear un producto accesible y que fuera de bajo coste a nivel de fabricación y de componentes.

Esto es básico para poder ser replicado y utilizado como base de trabajo para futuros proyectos en el campo de la investigación o la interacción artística. Por ello, uno de los criterios de selección de componentes era el coste por unidad, tanto para una tirada corta de pocas unidades como su producción en una tirada mayor.

Cuando se diseña un producto, se suele fabricar una pre-serie o prototipo con objeto de realizar pruebas, y una vez probada su funcionalidad, se fabrican un mayor número de unidades en función de la previsión de ventas. A la hora de calcular el valor de mercado de un dispositivo, se tienen en cuenta factores como el coste de los materiales, el coste de desarrollo a nivel de ingeniería, los costes de mano de obra, gastos de comercialización, los costes de mantenimiento y reparación y el margen de beneficio que se desea obtener.

Como se trata de un proyecto de carácter académico, la mayoría de factores anteriormente nombrados no están cuantificados en el coste de fabricación. No obstante, se han incluido el coste de los materiales por tratarse de un factor objetivo y desde el cual se puede realizar una extrapolación del valor de mercado del producto. Únicamente se han realizad estimaciones de coste de algunos elementos principales como carcasa, montaje o elementos mecánicos.

Del anterior ejercicio se puede observar una tabla resumen donde se ha recopilado el coste total de la placa:

Estimación coste fabricación PCB	10,00 €
Coste total componentes	36,80 €
Estimación coste montaje	5,00 €
Estimación coste carcasa y elementos mecánicos	5,00 €
COSTE TOTAL UNITARIO	56,80 €

Tabla 45 Coste unitario

Con estos resultados se puede observar un valor competitivo con los valores analizados en el estudio de estado del arte y estudio de mercado. Hay que tener claro que esto es el precio unitario para una tirada mínima de 1 a 10 unidades. Cuando se compran componentes electrónicos por otras vías, como son los distribuidores oficiales recomendados por el fabricante (normalmente mayoristas), el descuento por compra de grandes cantidades puede ser de hasta el 50%. Por ello, los costes finales de fabricación a nivel de materiales de un diseño comercial basado en este prototipo serían menores.

Al realizar tiradas grandes entran en juego otros factores como disponibilidad de componentes, plazos de entrega... Se han buscado componentes con gran disponibilidad o facilidad de reponerse. Para ello se ha utilizado la herramienta online Octopart que nos indica un factor de reposición de el listado de componentes. Podemos observar este factor y la reducción del precio de componentes para tiradas mayores, calculado con Octopart:

- 1 UNIDAD: 36,78 € *total - 95% BOM coverage*
- 100 UNIDAD: 31,12€ *total - 75% BOM coverage*
- 1000 UNIDAD: 25,11€ *total - 55% BOM coverage*

A continuación se encuentra un desglose completo de los costes por listado de componentes del dispositivo.

Cantidad	Componente	Referencia PCB	Referencia interna	Descripción	En Stock	MOQ	Distribuidor	SKU	Precio unitario	Total
1	BQ51051BYFPT	BQ510X	R-PQFP-N20	Charge Circuit	221	1	Rochester Electronics	BQ51051BYFPT	3,10 €	3,10 €
2	CAP_POL120x	C4	EIA3216	Capacitor Polarized	x		Generic Distributor	Standard Capacitor	0,08 €	0,16 €
28	CAP0402-CAF	C13	0402-CAF	Capacitor	x		Generic Distributor	Standard Capacitor	0,01 €	0,32 €
1	LS013B4DN0z	I _{C2}	FPC_10PIN_5289 _z	Sharp Mono Memory LCC	x		Generic Distributor	LS013B4DN0z	8,39 €	8,39 €
1	ATMEGA48-20AU	MEGA1284	TQFP44	8-bit Microcontroller	36383	1	Freelance Electronics	ATMEGA48-20AU	1,25 €	1,25 €
2	BSS138	Q2	SOT23-3	Common NMOSFET Part:	87007	1	Future Electronics	5596457	0,01 €	0,02 €
2	APT1608SEC	CHG	CHIPLED_060 _z	LED	x		Generic Distributor	APT1608SEC	0,24 €	0,48 €
24	RESISTOR_0402	R26	_0402	Resistors	x		Generic Distributor	Standard SKU	0,01 €	0,20 €
1	DF57H-2P-1.2V(21)	SW3	TACT_SMD2		13814	1	Future Electronics	7066021	0,28 €	0,28 €
1	AP2112K-3.3TRG1	U2	SOT23-5	SOT23-5 Fixed Voltage	4456	1	Arrow	AP2112K-3.3TRG1	0,29 €	0,29 €
1	MAX30101EFD+	U4	OLGA-14	Particle Detection IC	816	1	Digi-Key	MAX30101EFD+	4,21 €	4,21 €
1	SP62141.8V	U5	SC70		x		Generic Distributor	SP62141.8V	0,15 €	0,15 €
1	ADXL345BCC	U6	LGA14		4215	1	Area51	ADXL345BCC	2,94 €	2,94 €
1	TSL2561FN	U7	FN-6	TSL2561 illumination	6248	1	Arrow	TSL2561FN	1,34 €	1,34 €
1	BME280	U8	BME280_LGA		19613	1	Future Electronics	2053946	2,89 €	2,89 €
1	CSTCE8M00G55-RC	Y2	RESONATOR-SMC	Resonator	107206	1	Future Electronics	4428066	0,19 €	0,19 €
1	LMT70LMT70	U10	LMT70LMT70	Temperature sensor	x	1	Texas Instruments	LMT70LMT70	1,81 €	1,81 €
1	BAT520	X	BAT520	Battery	x	1	Generic Distributor	BAT520	2,89 €	2,89 €
1	760308201	X	760308201	Charge coil	x	1	Generic Distributor	760308201	1,89 €	1,89 €

Tabla 46. Coste componentes

ANEXO 12. Hojas de características

Los datasheet de los componentes más relevantes usados se muestran a continuación:

Nombre componente	Hoja de características URL
ADXL345	http://www.analog.com/media/en/technical-documentation/datasheets/ADXL345.pdf
bq51051b	http://www.ti.com/lit/ds/symlink/bq51050b.pdf
TSL2561	https://cdn-shop.adafruit.com/datasheets/TSL2561.pdf
BME280	http://www.mouser.com/ds/2/783/BST-BME280_DS001-11-844833.pdf
MAX30101	https://datasheets.maximintegrated.com/en/ds/MAX30101.pdf
ATmega 1284p	http://www.atmel.com/images/doc8059.pdf
MDBT40	http://www.raytac.com/download/MDBT40/MDBT40%20spec-Version%20A7.pdf
LMT70	http://www.ti.com/lit/ds/symlink/lmt70.pdf
SHARP MEMORY	https://cdn-shop.adafruit.com/datasheets/LS013B4DN04-3V_FPC-204284.pdf

Tabla 47. Datasheet componentes