

## 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 1. 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
<b>1. Análisis de mercado y especificación de requisitos</b>												
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												
<b>2. Diseño del sistema</b>												
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												
<b>3. Desarrollo HW v1</b>												
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												
<b>4. Fase pruebas</b>												
4.1. Desarrollo firmware básico												
4.2. Pruebas dispositivo												
4.3. Conclusiones												
<b>5. Desarrollo HW v2</b>												
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												
<b>6. Desarrollo SW</b>												
6.1. Desarrollo del firmware de la electrónica												
6.2. Desarrollo de la interfaz de usuario												
<b>7. Diseño</b>												
7.1. Diseño carcasa y correa												
7.2. Impresión 3D												
7.3. Mejoras carcasa												
<b>8. Fase pruebas Generales</b>												
8.1. Pruebas sistema completo												
8.2. Calibración sensores												
8.3. Pruebas autonomía												
8.4. Pruebas carga												

Tabla 2. 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](http://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.



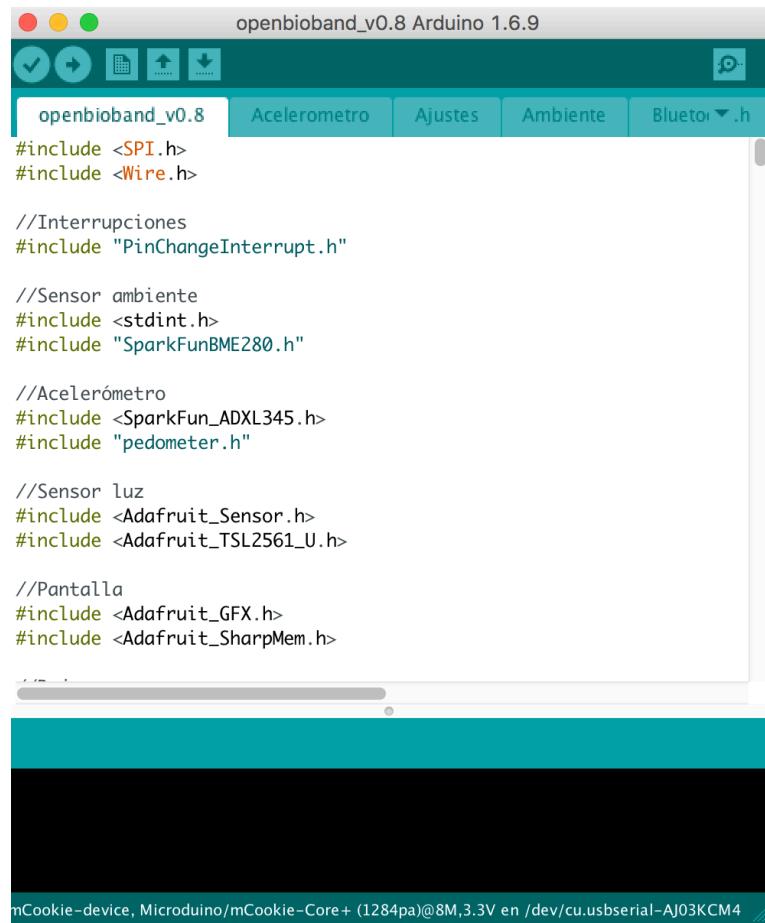
**Abrir:** Presenta un menú de todos los programas sketch de su "sketchbook" (librería de sketch). Un clic sobre uno de ellos lo abrirá en la ventana actual.



**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 is a toolbar with standard icons: file, open, save, upload, download, and a settings gear. A horizontal tab bar is visible, with the "Acelerometro" tab currently selected. The main code editor window contains the following C++ code:

```
#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>
```

At the bottom of the code editor, a status message reads "mCookie-device, Microduino/mCookie-Core+(1284pa)@8M,3.3V en /dev/cu.usbserial-AJ03KCM4".

Fig. 1 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.

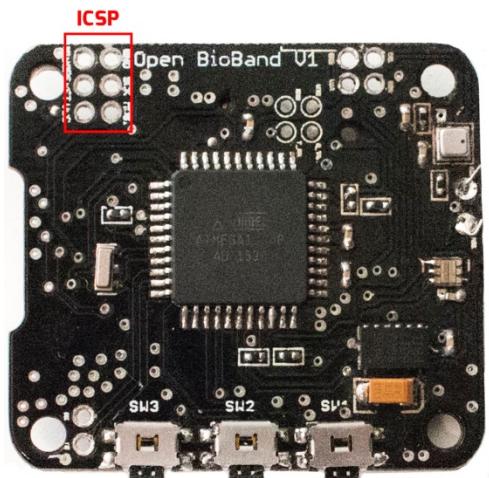


Fig. 2 Conector ICSP PCB

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.

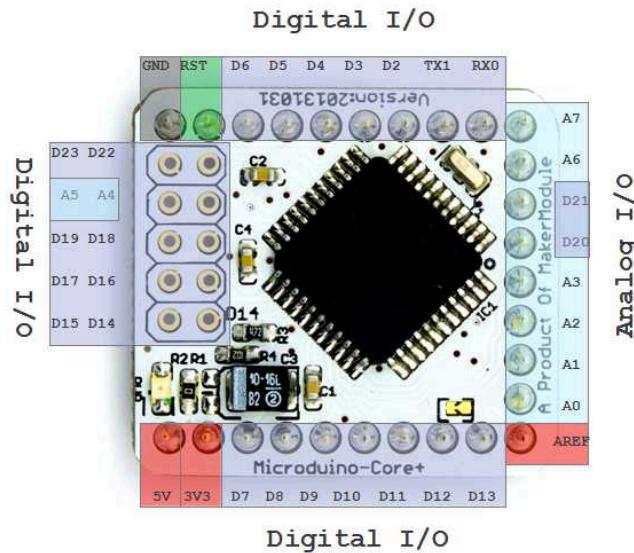


Fig. 3 Microduino

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.

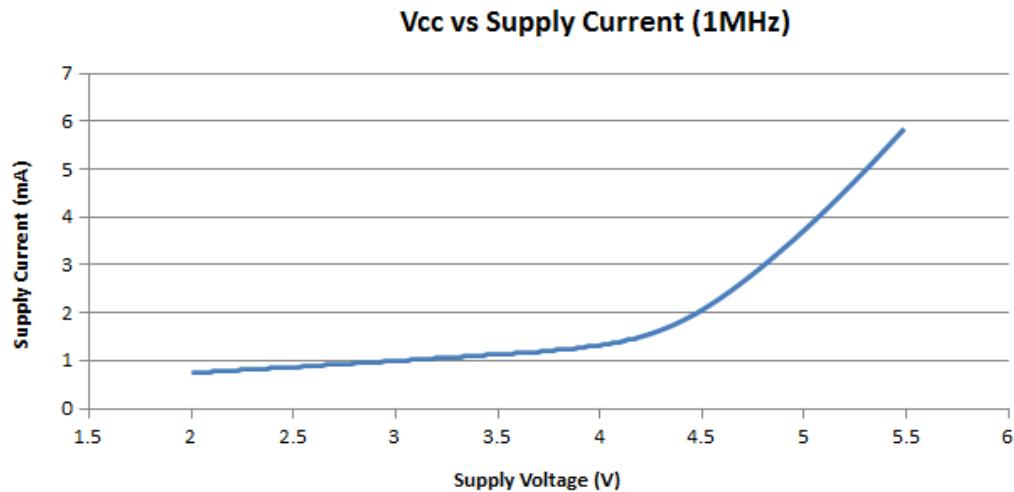


Fig. 4 Gráfica Tensión vs Intensidad

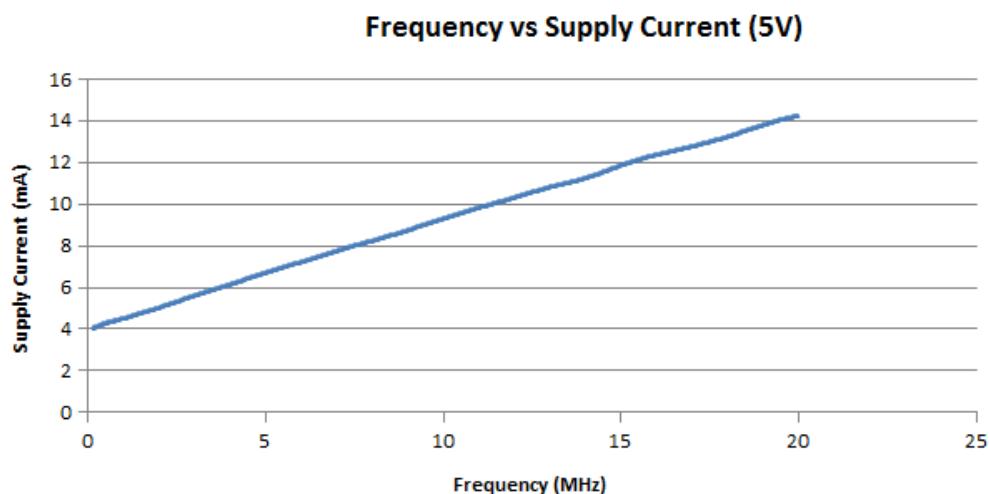


Fig. 5 Gráfica Frecuencia vs Intensidad

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.

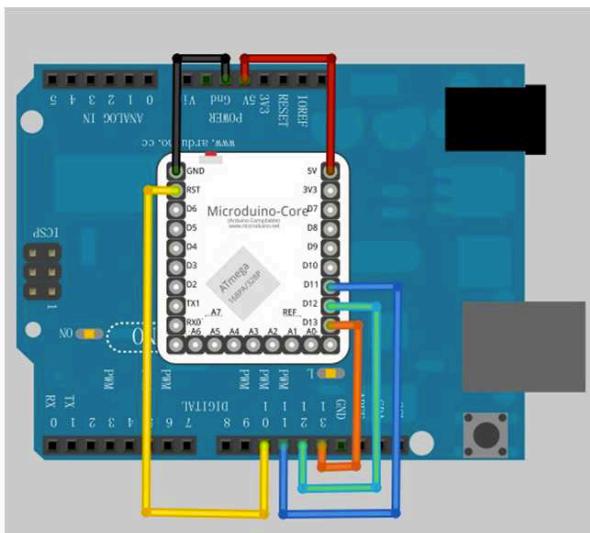


Fig. 6 Arduino bootloader

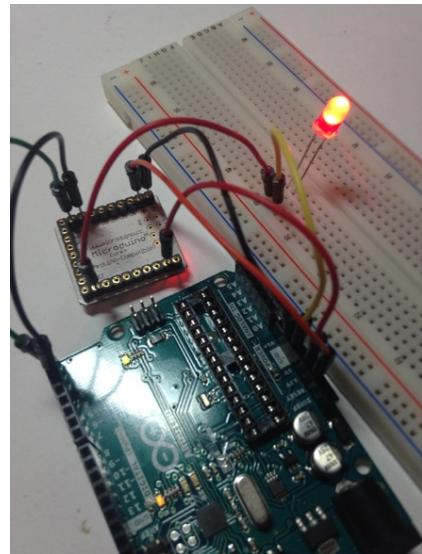


Fig. 7 Arduino como programador

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.

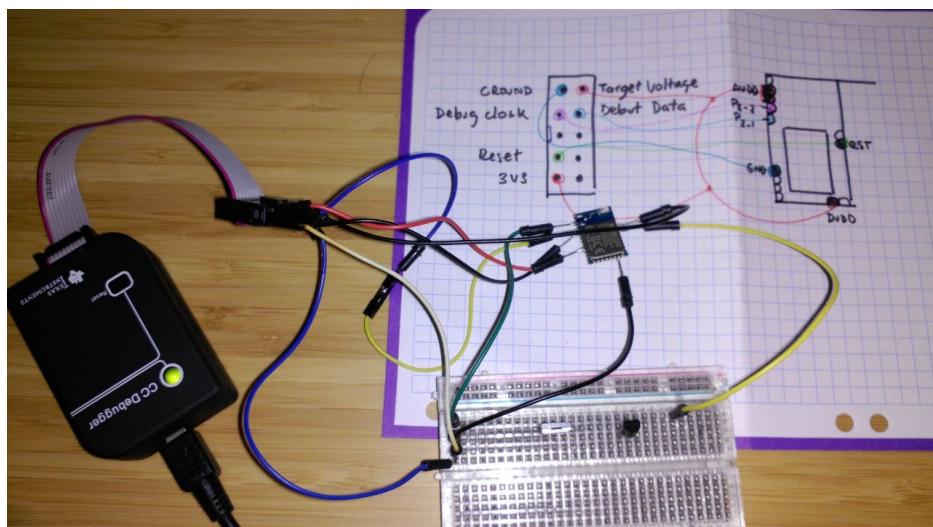


Fig. 8 Firmware BLE112

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.

## Pulsera biométrica Open Source para la monitorización del usuario

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.

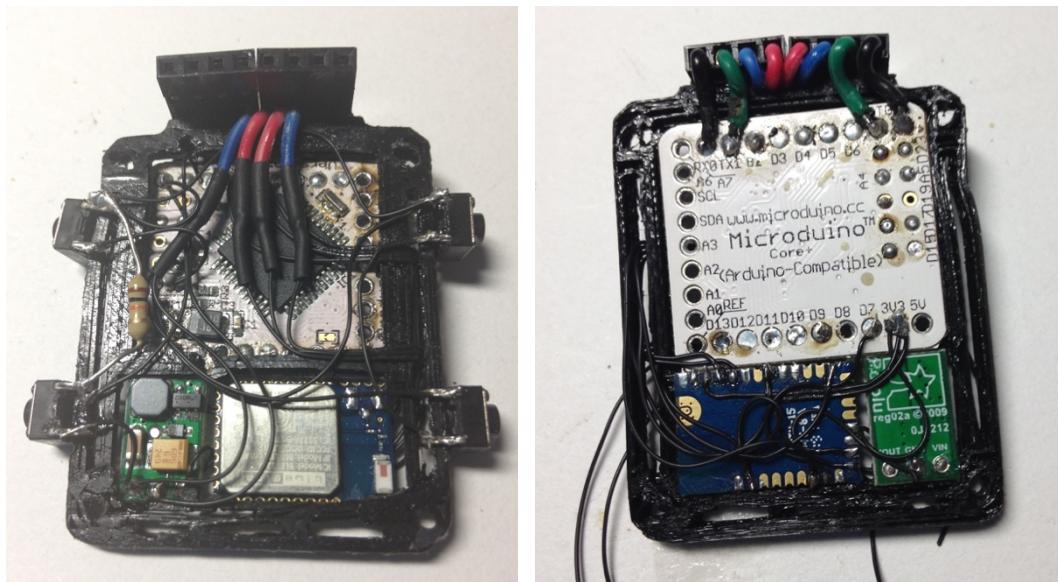


Fig. 9 Montaje pulsera versión 0

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.

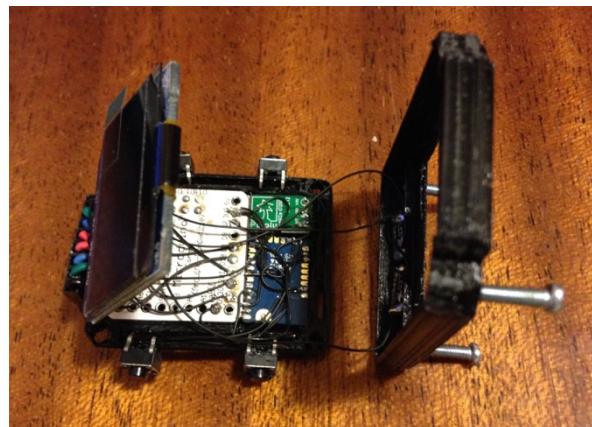


Fig. 10 Montaje pulsera versión 0

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.

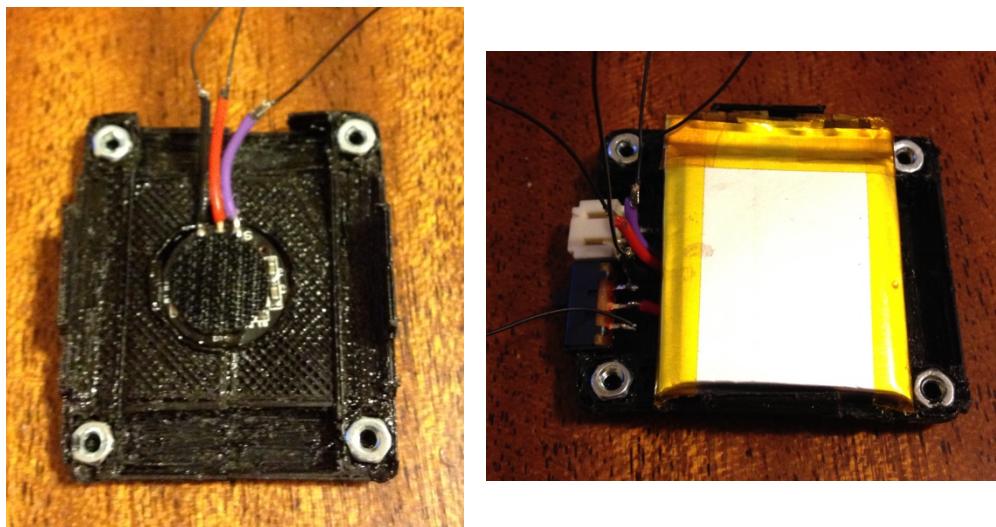


Fig. 11 Montaje pulsera versión 0

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.

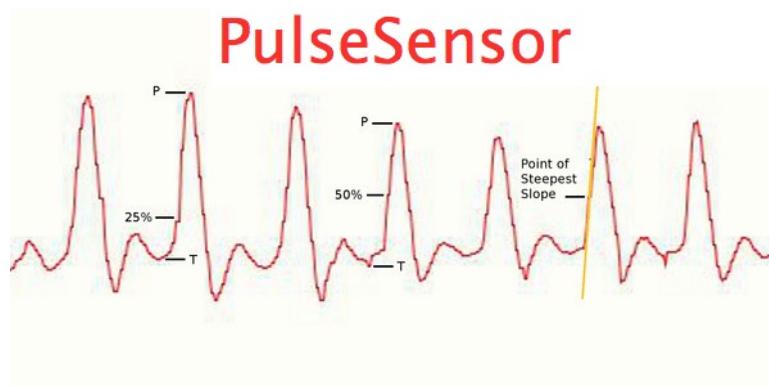


Fig. 12 Gráfica pulso

## 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. 13 Piezas 3D pulsera versión 0

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



Fig. 14 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 menuIluminacion = 1;
byte estadoIluminacion = 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
#define BME280 sensorAmbiente;

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

// Control sensor luz TSL2561
#define 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(CHGPIN, INPUT_PULLUP);  
 //attachPCINT(digitalPinToPCINT(CHGPIN), 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;
```

## Pulsera biométrica Open Source para la monitorización del usuario

```
sensorAmbiente.settings.humidOverSample = 1;
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 lecturaDatosTemp;

for (int rowi = 8; rowi < 16; rowi++)
{
    DEBUG_PRINT("0x");
    DEBUG_PRINT(rowi, HEX);
    DEBUG_PRINT("0:");
    for (int col = 0; col < 16; col++)
    {
        lecturaDatosTemp = sensorAmbiente.readRegister(contadorMem);
        DEBUG_PRINT((lecturaDatosTemp >> 4) & 0x0F, HEX); // Imprimir
        primera parte en hexadecimal
        DEBUG_PRINT(lecturaDatosTemp & 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 inicialización 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 digito,
* 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: const 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_PRINTLN(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,
    0xc0, 0x0, 0x30,
    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, 0xf0, 0xf, 0xff, 0xff, 0x0,
    0x3, 0xff, 0xf8, 0x1f, 0xff, 0xff, 0x80,
    0x3, 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, 0xff, 0xfe, 0x0,
    0x0, 0x1f, 0xff, 0xf0, 0xff, 0xff, 0xfc, 0x0,
    0x0, 0xff, 0xf0, 0xff, 0xff, 0xf8, 0x0,
    0x0, 0x7, 0xff, 0xf0, 0xff, 0xff, 0xf0, 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, 0x0, 0xf, 0xff, 0xff, 0xf0, 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, 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, 0x0, 0x7c, 0x0,
    0x1f, 0xff, 0xfc, 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, 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,
    0x1f, 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, 0xc1, 0xf1, 0x80,
    0xc7, 0xc1, 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, 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,  
        0xff, 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  
    };
```

```

// 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 *****

//! **** Nome: 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;
        }
    }
}

```

```

// 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 (botonArriba.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);
}
```

```

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  
    }  
}  
*****
```

```
*****  
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.

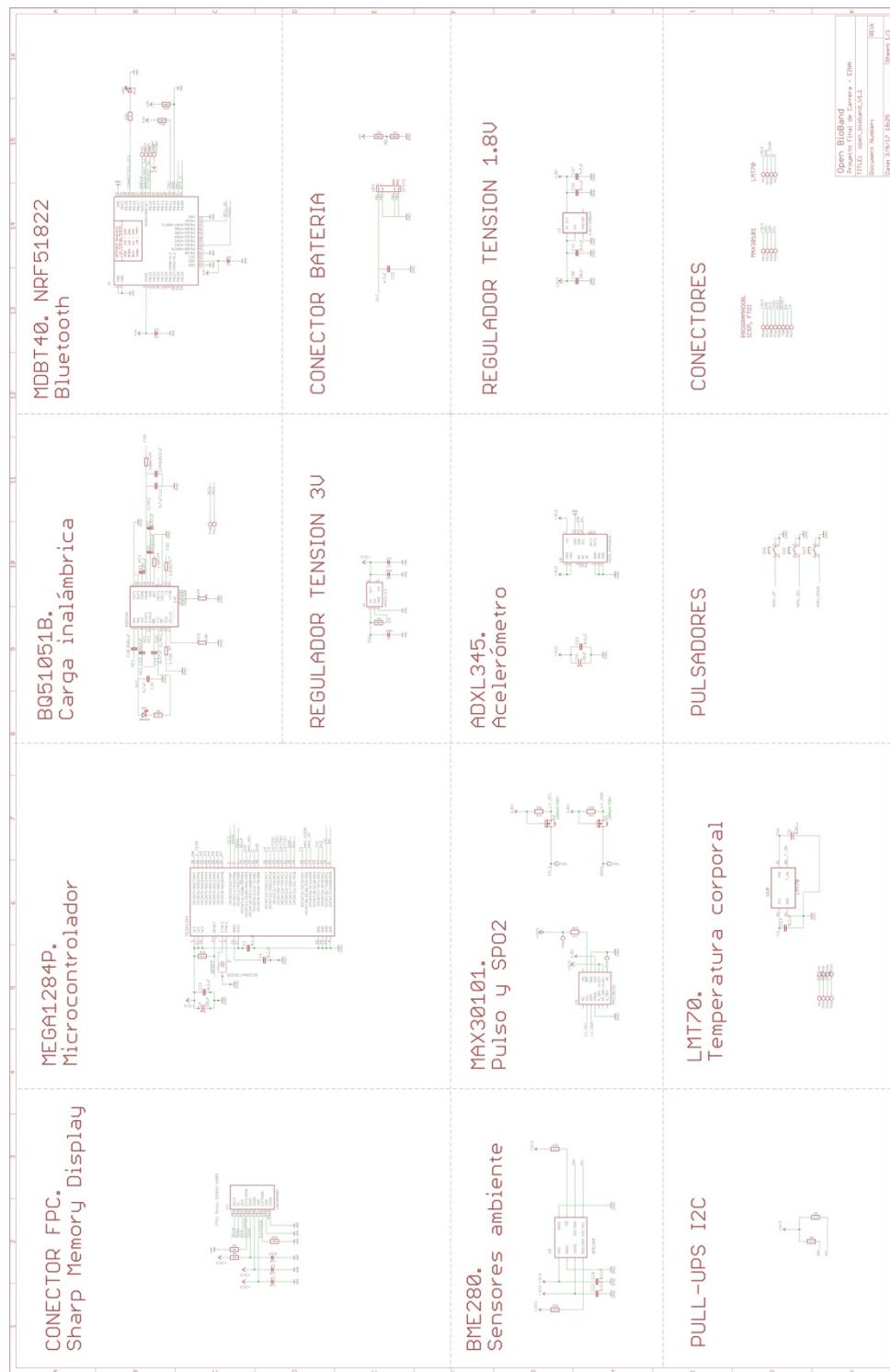


Fig. 15 Esquemático Open Bioband

## 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.

Identificación	Valor	Componente	Encapsulado	Descripción
BQ510X	BQ51051	BQ51051	R-PQFP-N20	Charge Circui
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 Polarizac
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 Polarizac
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_0603	LED
CHG1	BLUE	LED0603	CHIPLED_0603	LED
IC2	LS013B4DN04	LS013B4DN04	FPC_10PIN_52892-1095	Sharp Mono Memory LCD
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	ADXL3450RIC	ADXL3450RIC	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 symbc
C27	0.1uF	0.1UF-0402-16V-10%	402	0.1µF ceramic capacitors
U10	LMT70LMT70	LMT70LMT70	BGA4C40P2X2_77X77X5t	LOW-POWER SINGLE INVERTER GATE

*Tabla 3. 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 type="checkbox"/> Otros
Grosor PCB :	<input checked="" type="checkbox"/> 1mm	<input type="checkbox"/> 1.6mm	<input type="checkbox"/> Otros
	<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"/> Rojo	<input type="checkbox"/> Azul	<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 4. Requerimientos fabricación PCB*

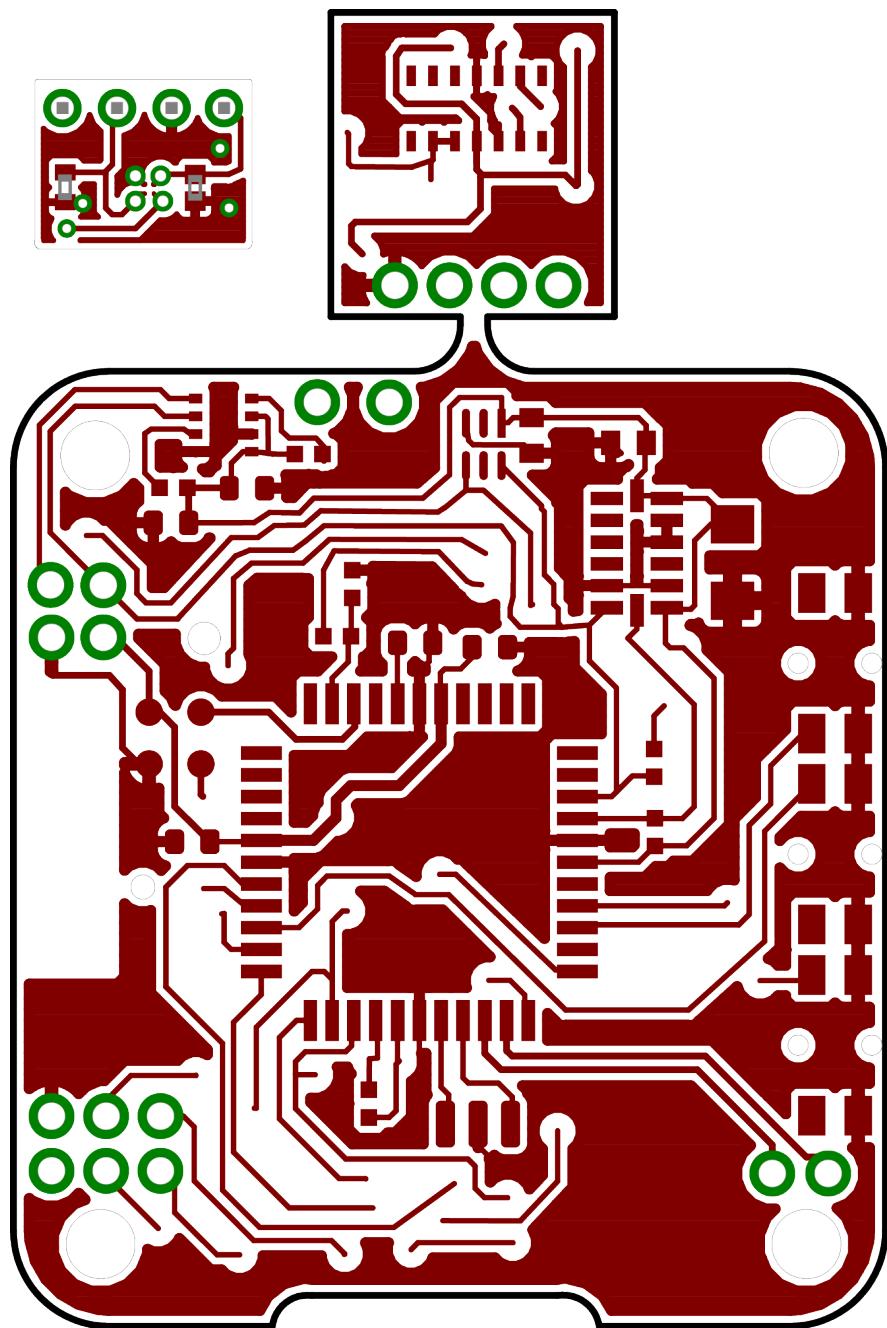


Fig. 16 Plano pistas cara TOP

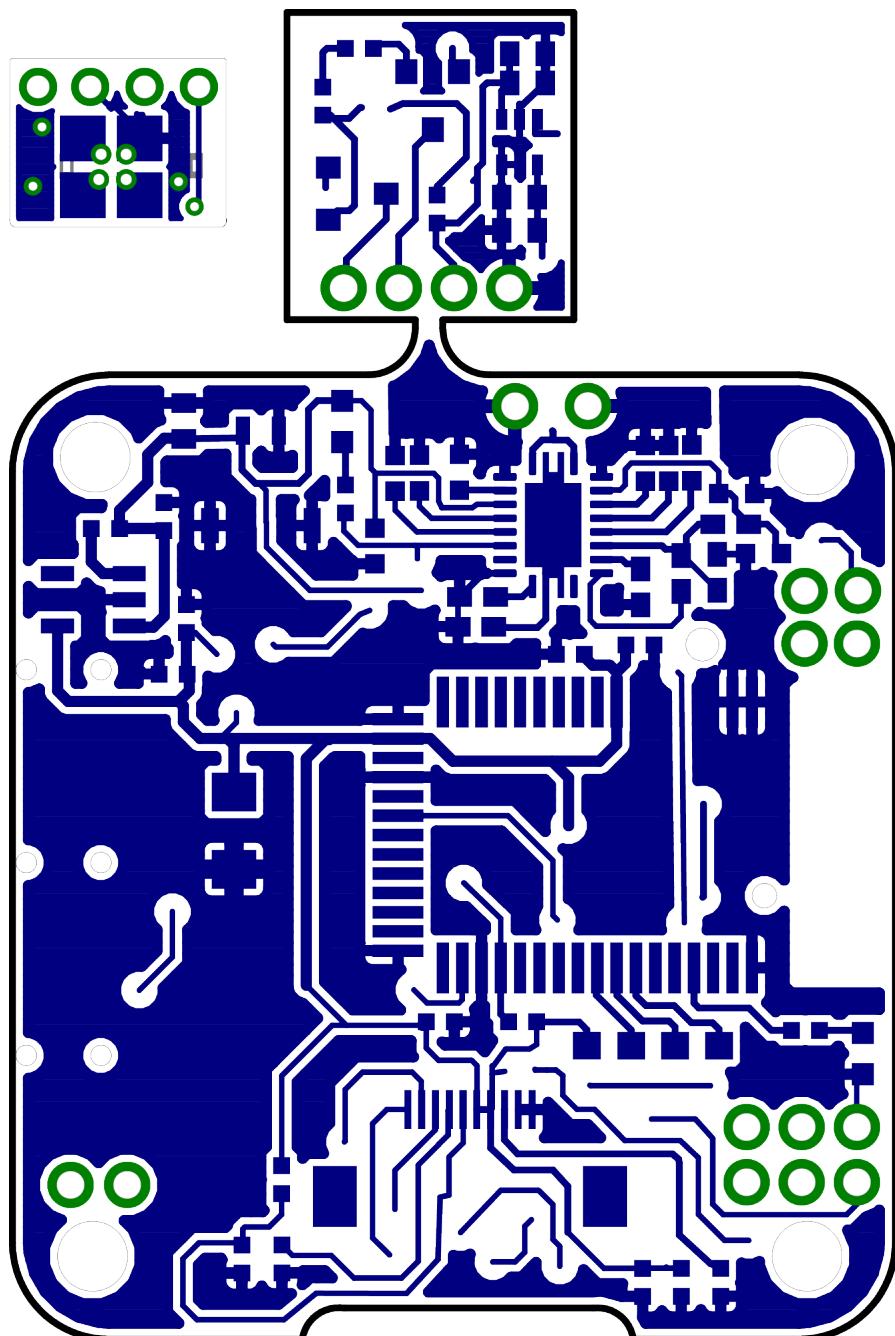


Fig. 17 Plano pistas cara BOTTOM

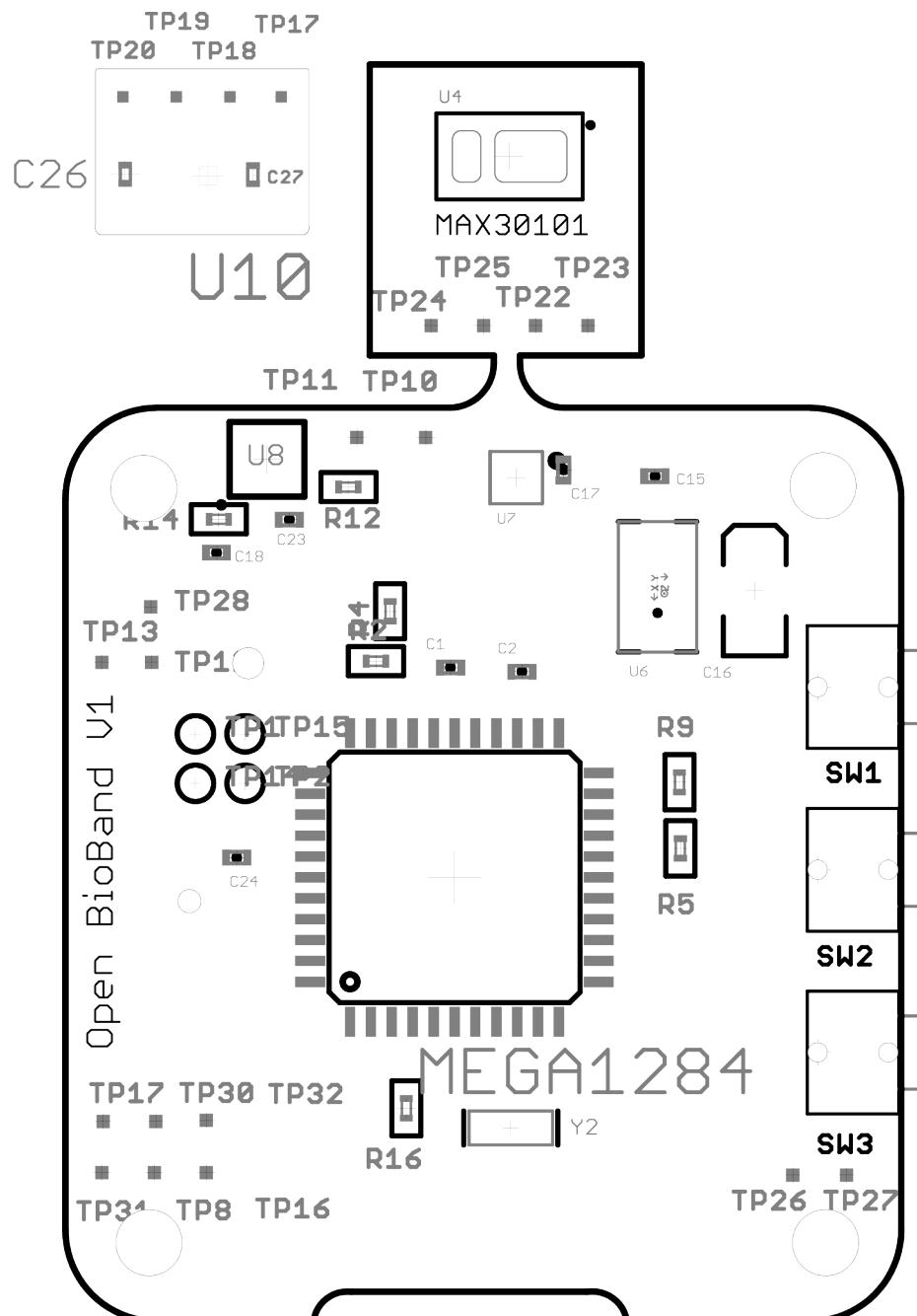


Fig. 18 Plano componentes cara TOP

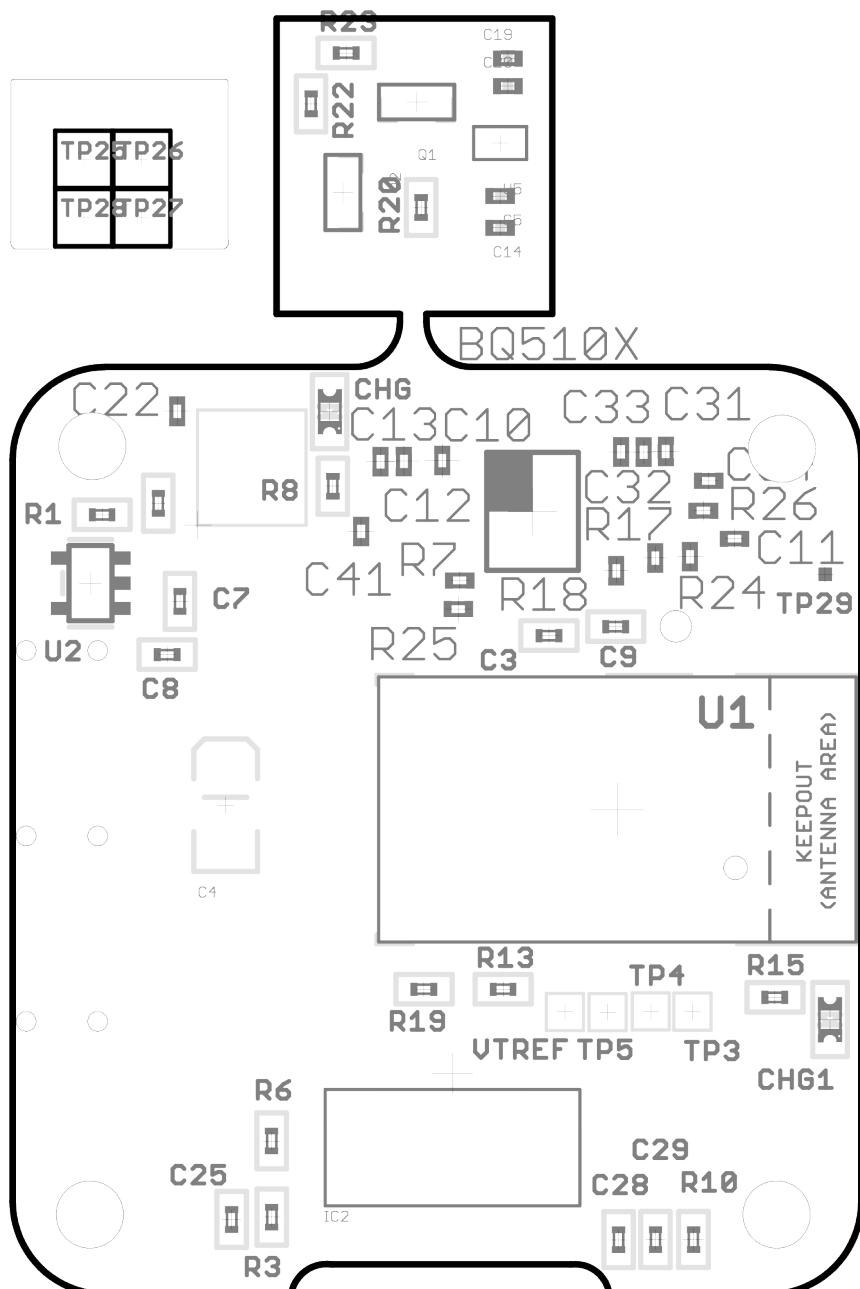


Fig. 19 Plano componentes cara BOTTOM

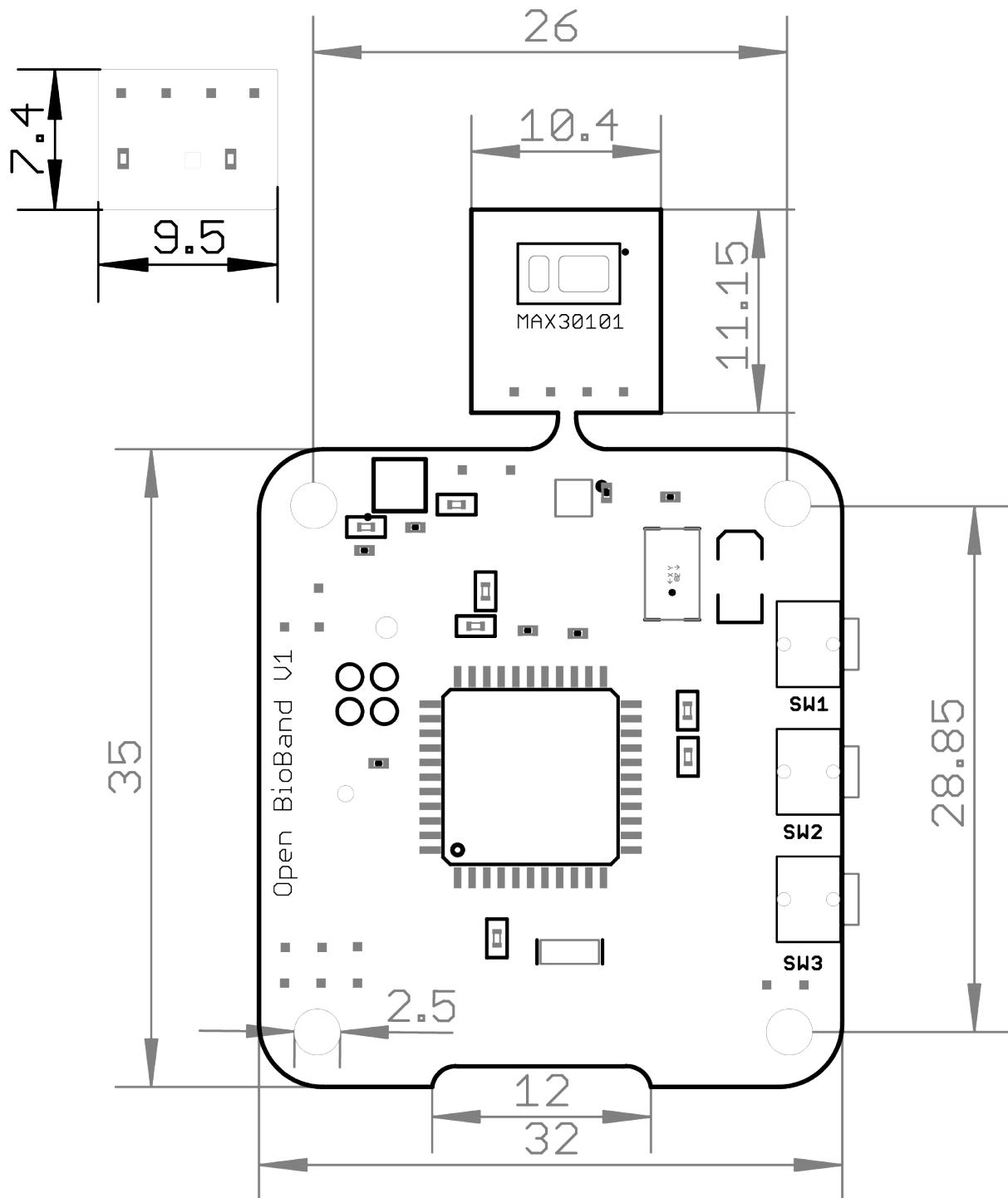
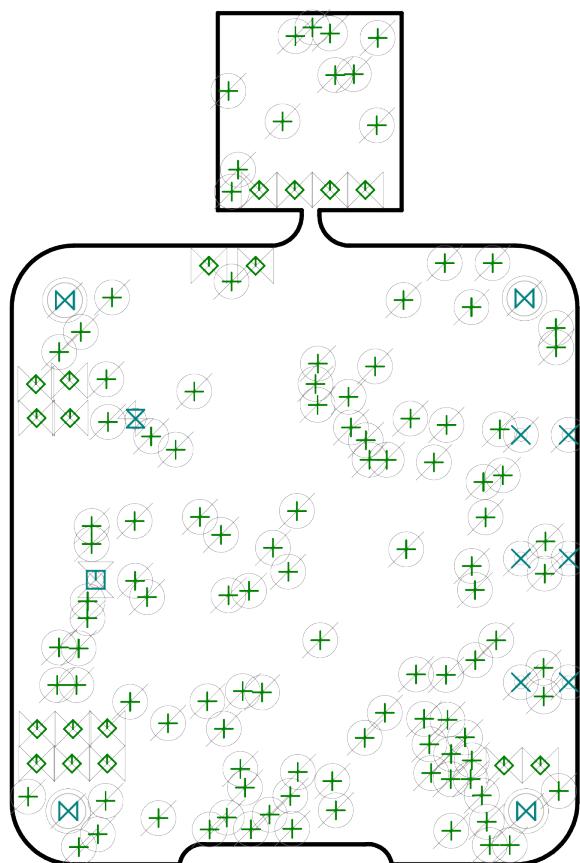
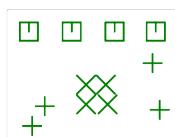


Fig. 20 Plano de dimensiones



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. 21 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:

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

Tabla 5 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_POL120€	C4	EIA3216	Capacitor Polarized	x		Generic Distributor	Standart Capacitor	0,08 €	0,16 €
28	CAP0402-CAF	C13	0402-CAF	Capacitor	x		Generic Distributor	Standart Capacitor	0,01 €	0,32 €
1	LS013B4DN0z	IC2	FPC_10PIN_5289z	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	BS5138	Q2	SOT23-3	Common NMOSFET Part:	87007	1	Future Electronics	5596457	0,01 €	0,02 €
2	APT1608SEC	CHG	CHIPLED_0603	LED	x		Generic Distributor	APT1608SEC	0,24 €	0,48 €
24	RESISTOR_0402	R26	_0402	Resistors	x		Generic Distributor	Standart 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	RESONATORS-MC	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 coi	x	1	Generic Distributor	760308201	1,89 €	1,89 €

Tabla 6. 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	<a href="http://www.analog.com/media/en/technical-documentation/datasheets/ADXL345.pdf">http://www.analog.com/media/en/technical-documentation/datasheets/ADXL345.pdf</a>
bq51051b	<a href="http://www.ti.com/lit/ds/symlink/bq51050b.pdf">http://www.ti.com/lit/ds/symlink/bq51050b.pdf</a>
TSL2561	<a href="https://cdn-shop.adafruit.com/datasheets/TSL2561.pdf">https://cdn-shop.adafruit.com/datasheets/TSL2561.pdf</a>
BME280	<a href="http://www.mouser.com/ds/2/783/BST-BME280_DS001-11-844833.pdf">http://www.mouser.com/ds/2/783/BST-BME280_DS001-11-844833.pdf</a>
MAX30101	<a href="https://datasheets.maximintegrated.com/en/ds/MAX30101.pdf">https://datasheets.maximintegrated.com/en/ds/MAX30101.pdf</a>
ATmega 1284p	<a href="http://www.atmel.com/images/doc8059.pdf">http://www.atmel.com/images/doc8059.pdf</a>
MDBT40	<a href="http://www.raytac.com/download/MDBT40/MDBT40%20spec-Version%20A7.pdf">http://www.raytac.com/download/MDBT40/MDBT40%20spec-Version%20A7.pdf</a>
LMT70	<a href="http://www.ti.com/lit/ds/symlink/lmt70.pdf">http://www.ti.com/lit/ds/symlink/lmt70.pdf</a>
SHARP MEMORY	<a href="https://cdn-shop.adafruit.com/datasheets/LS013B4DN04-3V_FPC-204284.pdf">https://cdn-shop.adafruit.com/datasheets/LS013B4DN04-3V_FPC-204284.pdf</a>

Tabla 7. Datasheet componentes