industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

DISEÑO Y DESARROLLO DE UN SISTEMA DE MONITORIZACIÓN DE PARÁMETROS DE INTERÉS DURANTE LA REALIZACIÓN DE TERAPIAS DE EXPOSICIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: **BRAHYAN FABIAN PORTILLA FLOREZ**
Director: JUAN ANTONIO LÓPEZ RIQUELME
Codirector: NIEVES PAVÓN PULIDO

Cartagena, 10 de octubre de 2018

Universidad Politécnica de Cartagena

Índice

Lista de figuras

Lista de tablas

Capítulo 1

Introducción

1.1 Contexto

Los dispositivos móviles inteligentes, también conocidos como *Smarthphones*, irrumpieron a principios del presente siglo con gran fuerza, impulsados por la posibilidades y versatilidades que ofrecían a los usuarios. Estos dispositivos han seguido mejorando, aumentado sus capacidades como la memoria, las comunicaciones, los periféricos conectables, apoyado por sistemas operativos muy capaces y con precios asequibles para gran parte de las personas.

Como resultado de todo lo anterior, potentes máquinas de procesamiento y conectividad en un reducido espacio estás a disposición de millones de personas, que se pueden aprovechar para el desarrollo de nuevos sistemas que puedan ser de gran utilidad para los usuarios. En este contexto, en el ámbito *e-Health* y clínico es muy interesante el desarrollo de sistemas de monitorización para pacientes como el pulso cardiaco, la temperatura corporal, entre otros, que ayuden en el tratamiento o terapias de pacientes.

Bajo este abanico de posibilidades de las nuevas tecnologías, este proyecto consiste en el diseño y desarrollo de un sistema de monitorización de parámetros para terapias de exposición compuesto por elementos hardware y software.

En el apartado hardware, será necesario la selección de un sistema de adquisición de los parámetros para la terapia, dispositivos para el almacenamiento, tratamiento y visualización de datos, y los acondicionamientos electrónicos de señal necesarios.

Para él apartado software, será necesario el desarrollo de la arquitectura que permita las labores de gestión, transporte y almacenamiento de datos entre los diferentes dispositivos hardware implicados, siendo necesario el uso de tecnologías inalámbricas y cableadas.

1.2 Objetivos

Este trabajo tiene como objeto diseñar y desarrollar una arquitectura hardware y software para la monitorización de parámetros de interés durante la realización de terapias de exposición, utilizando un dispositivo móvil, un sistema de adquisición de parámetros biomédicos y la utilización de tecnología *Bluethooth Low Energy*. Para obtener el sistema propuesto, se plantean los siguientes subobjetivos:

- Estudio de la metodología de terapia de exposición.
- Selección de los sensores más importantes para la monitorizar el estado del paciente durante la realización de la terapia de exposición.
- Estudio y selección del sistema hardware, entre todas las opciones tecnológicas disponibles en el mercado, para la comunicación con *Bluetooth Low Energy*.
- Estudio y elección de los componentes hardware que compondrán el sistema.
- Estudio del software y lenguajes de programación necesarios para configurar los diferentes dispositivos hardware que componen el sistema.
- Análisis y diseño de una aplicación para dispositivos móviles.
- Realización de pruebas de funcionamiento del sistema propuesto para la validación de su funcionamiento.

1.3 Desarrollo de la memoria

1.3.1 Capítulo 1. Introducción

En este capítulo se introduce la motivación de este trabajo, se exponen los objetivos propuestos y se describe la estructura de la memoria.

1.3.2 Capítulo 2. Estado del Arte

En este capítulo se exponen las diferentes tecnologías que se ven involucradas en la realización del proyecto. Se describen los diferentes protocolos de comunicación disponibles para dispositivos móviles, se presentan las alternativas hardware y software de los sistemas empotrados existentes en el mercado y sus plataformas de desarrollo.

1.3.3 Capítulo 3. Descripción del Sistema

Este capítulo presenta las tecnologías elegidas del capítulo anterior y que compondrán la configuración del sistema. Posteriormente, se detallan con mayor profundidad cada una de las opciones elegidas.

1.3.4 Capítulo 4. Descripción del Hardware y Software Desarrollado

En este capítulo se explica con detalle el trabajo desarrollado para la realización de este proyecto, se exponen las soluciones adoptadas a nivel hardware y software.

1.3.5 Capítulo 5. Conclusiones y Trabajos futuros

En este capítulo se presentan, en primer lugar, las conclusiones obtenidas tras la realización del trabajo y, en segundo lugar, ideas de trabajos futuros a desarrollar.

Capítulo 2

Estado del Arte

2.1 Introducción

El proyecto consiste en diseñar y desarrollar una arquitectura hardware y software para la monitorización de parámetros de interés durante la realización de terapias de exposición, utilizando un dispositivo móvil, un sistema de adquisición de parámetros biomédicos y la utilización de tecnología *Bluethooth Low Energy*.

Para lograr el objetivo propuesto, será necesario realizar una solución hardware y software que requerirá diferentes elementos que compondrán el sistema.

En concreto, en este capítulo se realizará un análisis de los diferentes sistemas empotrados disponibles en el mercado, que será parte fundamental para el desarrollo hardware que ira unido en la placa y se conectará con el teléfono móvil inteligente. También, se tratará en este capítulo los sistemas operativos móviles más utilizados y los protocolos de comunicación disponibles para la comunicación los dispositivos hardware que componen el sistema.

2.2 Terapia de exposición

La terapia de exposición es un proceso muy utilizado que se engloba dentro de la terapia cognitivo conductual. Consiste en el acercamiento a situaciones que generan estrés, miedo o fobias para modificar la sensibilidad antes los estímulos generados.

En la terapia de exposición los pacientes son expuestos, de forma controlada y sistemática a situaciones que les generan ansiedad, miedo o reacciones negativas, con el objetivo de establecer un nuevo aprendizaje que permita acostumbrarse a la situación y reducir el conflicto que les generan. Por lo general, la exposición se realiza de forma gradual y se tiene en cuenta características del paciente.

La exposición continua ante estímulos negativos en el paciente, lo que ocurre generalmente es el fenómeno de habitación, a la vez, ayuda al que el individuo reflexione sobre los temores que le generan los estímulos y el nivel de amenaza que en realidad le supone.

La habituación consiste en la familiarización con el estímulo que genera la fobia o reacción negativa, en términos psicofisiológicos, disminución de la reactividad automática, y términos subjetivos, respuesta subjetiva a la ansiedad.

2.2.1 Funcionamiento de la terapia de exposición

Las sesiones generalmente tienen una duración de entre 30 a 120 minutos, ya que puede resultar agotador para los pacientes, y las fases de las que consta una terapia de exposición son generalmente las que se presentan a continuación.

- La persona se expone a los estímulos que le generan incomodidad del modo que terapeuta considere apropiada de acuerdo con historia clínica y valoración psicológica.
- Cuando la paciente considera que la ansiedad empieza a ser incontrolable, se recomienda pausar la terapia temporalmente hasta reducir los niveles de estrés y posteriormente, reincorporarse a la situación.
- En caso de retirarse de manera temporal de la actividad, las técnicas de respiración y el movimiento físico pueden ayudar a reducir los niveles de ansiedad.
- La exposición a situaciones de estrés es continua, progresivamente se aumenta la intensidad o la situación para generar una mayor tolerancia en el paciente. Las frecuencia y tiempo pueden modificarse de acuerdo con la valoración psicológica.

2.3 Protocolos de comunicación

Los protocolos de comunicación son el conjunto de pautas que posibilitan que diferentes elementos que forman parte de un sistema establezcan una vía de comunicación y puedan intercambiar información. Son quienes instituyen los parámetros que determinan cuál es la semántica y la sintaxis que debe utilizarse en el proceso de comunicación en cuestión.

En las computadoras, los protocolos de comunicación son los que determinan como se transmiten y deben circular los datos en un proceso de intercambio de información dentro de una red.

A continuación, se detallan tres protocolos de comunicación que podrían ser válidos para la solución del problema que en este proyecto se plantea.

2.3.1 Wi-Fi

Wi-Fi [1] (Wireless Fidelity) es un conjunto de estándares para redes inalámbricas basado en las especificaciones IEEE 802.11, creado para redes locales inalámbricas que también es utilizado para conectarse a Internet.

Es un sistema que permite conectar redes y ordenadores sin la necesidad de cables, mediante ondas de radio. WiFi tiene como objetivo fomentar las conexiones inalámbricas y facilitar la compatibilidad entre los diferentes dispositivos.

Como todos los protocolos de comunicación, WiFi tiene algunas ventajas y desventajas que se detallan a continuación:

Ventajas:

-Conectividad inalámbrica

-Facilidad de conexión, ya que cualquier dispositivo con acceso a la red puede conectarse a ella siempre que esté en el rango de espacio donde opera.

-Elección entre señales libres o con seguridad.

-Una vez configuradas, las redes WiFi permiten el acceso de diferentes dispositivos sin gasto de infraestructura.

-Compatibilidad asegurada entre dispositivos que cuenten con la certificación WiFi.

Desventajas:

-Se pueden generar fallos en la conexión.

-Distancia limitada.

-Aunque en la actualidad se cuenten con sofisticados protocolos de protección de redes WiFi, son vulnerables al hackeo.

-Se consigue menor velocidad en comparación a las logradas por sistemas cableados debido a interferencias y perdidas de señal.

-No es compatible con otros protocolos de comunicación no cableados como el Bluetooth y GPRS, entre otros.

2.3.2 Bluetooth

La tecnología Bluetooth [2] fue creada por Ericsson en 1994 y actualmente es propiedad de un grupo de empresas llamado SIG. Esta tecnología permite que dos dispositivos se puedan conectar entre sí de forma inalámbrica intercambiando información por medio de ondas de radio.

Podemos distinguir entre dos categorías de Bluetooth, que son la clase 1 y la clase 2, y que se distinguen por la potencia de la transmisión de datos. La clase 1 es menos habitual y permite conectar dos dispositivos hasta 100 metros de distancia y la clase 2, la más implementada en dispositivos móviles, permite conectar dispositivos en distancias de hasta 10 metros.

A continuación, se describen las principales ventajas y desventajas de esta tecnología:

Ventajas:

-Está muy extendido y casi todos los dispositivos móviles que se encuentran en el mercado implementan esta tecnología.

-Funcionamiento sencillo, no se necesita grandes conocimientos para poder utilizarla.

-Es gratuito

-Permite él envió de información entre dispositivos sin la necesidad de cables.

-Permite tener un cierto nivel de seguridad y control de la información que se intercambia.

Desventajas:

-Aumenta el consumo de la batería en el dispositivo mientras se está usando.

-Velocidad de transferencias de información bajas en comparación con otras tecnologías.

-Limitación de alcance en la conexión.

2.3.3 BLE (Bluetooth Low Energy)

Bluetooth Low Energy [3] también conocido como "Bluetooth Smart" fue introducido como parte de las especificaciones del estándar Bluetooth 4.0. Se solapa en algunos aspectos con el Bluetooth clásico, pero es una tecnología diferente que nació en Nokia y que luego fue adquirida por SIG.

Es una tecnología que permite la transferencia de información inalámbricamente con velocidades menores que el Bluetooth clásico. En concreto, trabaja en la banda de los 2,4 GHz y tiene un alcance de hasta 100 metros, pero su principal característica es el bajo consumo, ya que con BLE podemos emitir señales durante semanas e incluso meses sin la necesidad de recargar o cambiar la batería del dispositivo emisor o receptor.

En el apartado de seguridad, BLE cuenta con un cifrado llamado AES 128 con CCM.

Se detallan algunas ventajas y desventajas de esta tecnología a continuación.

Ventajas:

-Sin duda una de las grandes ventajas de esta tecnología es su bajo consumo.

-Aumenta el número de dispositivos que pueden conectarse simultáneamente en comparación con Bluetooth clásico.

-Apoyo por parte de las principales empresas.

-Es una tecnología barata y flexible que permite su utilización en dispositivos como teléfonos móviles, tablets, smartTV, cepillos de dientes y monitores de ritmo cardiaco, entre otros.

Desventajas:

-No es compatible con Bluetooth clásico.

-Menores velocidades de transferencia de información en comparación con otras tecnologías.

2.3.3.1 GAP

Es el acrónimo para el Generic Access Profile, y se encarga de controlar las conexiones y los anuncios en BLE. El GAP que permite que un dispositivo sea público hacia el exterior y determina como dos dispositivos pueden (o no) interactuar entre ellos.

El GAP define dos define varios roles entro los dispositivos, pero hay que tener claro que se tendrán dispositivos centrales y periféricos.

- Los periféricos son dispositivos pequeños, de baja potencia, de bajos recursos, que pueden conectarse a dispositivos centrales mucho más potentes. Un ejemplo de periférico puede ser un glucómetro, un medidor de pulsaciones, un beacon, etc.
- Un dispositivo central se corresponde normalmente con un teléfono móvil o una Tablet, que suelen tener una capacidad de procesamiento mucho mayor.

La transmisión de información en el GAP se realiza de dos maneras que son: El *Advertising Data* payload y el *Scan Response* payload.

2.3.3.2 GATT

GATT es el acrónimo de Generic Attribute Profile, y define la manera en que dos dispositivos BLE pueden comunicarse usando los Servicios y Características. La comunicación se realiza mediante un protocolo conocido como ATT, que se usa para almacenar los servicios, características y datos relacionados en una tabla usando identificadores de 16 bits para cada entrada en la tabla.

GATT entra en juego una vez se ha establecido una conexión dedicada entre dos dispositivos, lo que significa que ya hemos pasado previamente por el GAP.

Lo más importante a tener en cuenta con el GATT y las conexiones, es que las conexiones son exclusivas. Un periférico BLE sólo puede ser conectado a un dispositivo central (un teléfono móvil, etc.) a la vez. Tan pronto como un periférico se conecta a un dispositivo central, dejará de anunciarse y otros dispositivos ya no podrán verlo o conectarse a él, hasta que se finalice la conexión existente.

Establecer una conexión también es la única forma de permitir la comunicación bidireccional, en la que el dispositivo central puede enviar datos al periférico y viceversa.

En el intercambio de información en el GATT, al periférico se conoce como el servidor GATT, que contiene los datos de búsqueda ATT y las definiciones de servicio y características, y el cliente GATT (el teléfono/tableta), que envía solicitudes a este servidor.

Todas las transacciones son iniciadas por el dispositivo maestro, el GATT Client (central), que recibe la respuesta del dispositivo esclavo, el GATT Server (periférico).

2.4 Sistemas empotrados

Un sistema empotrado es un sistema informático de uso específico que esta encapsulado totalmente por el dispositivo que controla y que está dedicado a aplicaciones de control. Se denominan empotrados porque normalmente forman parte de otros sistemas más completos y con funciones más generales. Los sistemas empotrados se encuentran en multitud de aplicaciones, desde la electrónica de consumo hasta el control de complejos procesos en la industria.

Están muy presentes en nuestra vida cotidiana como, nuestros teléfonos móviles y ordenadores, los automóviles, los aviones, etc. Todos cuentan con algún dispositivo empotrado para realizar alguna de sus funciones.

Los sistemas empotrados están diseñados para llevar a cabo tareas específicas que le son programadas. Esta tarea puede no ser única y por tanto puede incluir varias opciones más, por ejemplo, los diferentes programas de lavado de una lavadora. En la actualidad, con el gran avance que vive la microelectrónica, los dispositivos empotrados han evolucionado y son muchos los que ofrecen funcionalidades cercanas a un ordenador y a precios muy asequibles.

En el mercado podemos disponer de multitud de dispositivos empotrados y a continuación detallaremos algunos de ellos y que pueden servir para la resolución del problema planteado en este trabajo.

2.4.1 Alternativas hardware para comunicación BLE

A continuación, se presenta algunos de los sistemas empotrados disponibles con soporte para la tecnología BLE.

2.4.1.1 BLE Nano

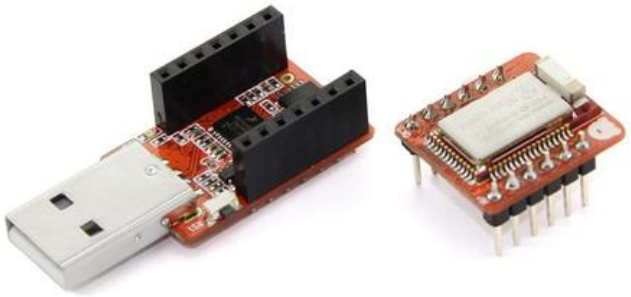

Ilustración 2.1.- Módulo BLE Nano y placa de desarrollo.

BLE Nano (ver Ilustración 2.1) es una pequeña tarjeta para desarrollo que cuenta con Bluetooth 4.1 Low Energy (BLE). Implementa un core Nordic nRF51822 (un ARM Cortex-M0 SoC + Capacidad BLE) con frecuencia de reloj a 16 MHz con muy bajo consumo.

El desarrollo de aplicaciones para este dispositivo es fácil, se pude programar con Nordic nRF51822 BLE SDK, mbed's Bluetooth Low Energy API y Arduino Library para nRF51822. BLE Nano puede operar entro los 1,8 y los 3,3 V y es compatible con bastantes componentes electrónicos, teléfonos móviles y los SO más populares en el mercado.

2.4.1.2 ESP32

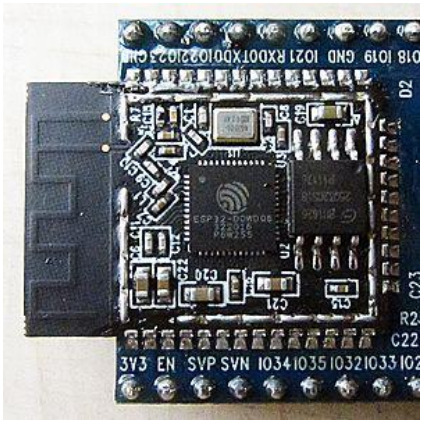

Ilustración 2.2.- Módulo ESP32.

El ESP32 es una serie de sistemas empotrados que destacan por su bajo consumo de energía, es un dispositivo que cuenta con un microcontrolador con Wi-Fi integrado y Bluetooth de modo dual. Además, por sus prestaciones y coste es una solución muy idónea para proyectos en el marco de IoT.

A continuación, se detallan sus características a nivel de procesador y conectividad.

-Procesador:

-CPU: Xtensa dual-core (or single-core) 32-bit LX6 microprocesador, trabaja a 160 ó 240 MHz y puede llegar hasta los 600 DMIPS.

-Ultra bajo consumo (ULP).

-Memoria: 520 KiB SRAM.

-Conectividad:

-Wi-Fi: 802.11 b/g/n/e/i.

-Bluetooth: v4.2 BR/EDR y BLE.

2.4.1.3 Arduino + BLE Shield

Ilustración 2.3.- Placa BLE Shield para Arduino.

BLE Shield en un dispositivo diseñado para trabajar con placas Arduino o compatibles, incluido Arduino Uno, Mega 2560, Leonardo y Due. Este shield, permite conectar una placa de Arduino por medio de Bluetooth Low Energy con otros dispositivos, como teléfonos móviles, ordenadores o tablets.

Entre sus características más importantes cabe destacar las siguientes:

-Opera entre los 3,3 y los 5 V, esto hace que pueda ser compatible con muchas placas Arduino.

-Microcontrolador Nordic nRF8001 Bluetooth Low Energy IC.

2.4.2 Sistemas empotrados de código abierto

Los grandes avances de las últimas décadas, junto a los esfuerzos por traer la electrónica al alcance de todos para poder aprender y desarrollar proyectos de cualquier tipo, permiten que hoy en día tengamos al alcance muchas alternativas tanto hardware como software, en este apartado se detallaran algunas de ellas.

2.4.2.3 Arduino

Arduino [4] es una plataforma de código abierto que está basada en hardware y software libre donde una de sus grandes ventajas es el bajo coste y su fácil uso. Su entorno de desarrollo es Arduino IDE que está basado en el lenguaje C++.

Nació en el año 2003 por un grupo de estudiantes del Instituto de Diseño Interactivo de Ivrea, Italia, con el objetivo de facilitar el uso y acceso a la electrónica y programación. A continuación, se detallan algunas de sus características principales:

- Plataforma de código abierto.
- Arduino es una placa hardware libre, lo que permite que cualquier persona pueda diseñar u obtener una placa personalizada basada en Arduino.
- Bajo coste.
- Es muy popular, ofrece muchas alternativas y está apoyada por una gran comunidad.
- Diferentes tipos de placas según las necesidades.

Arduino Uno

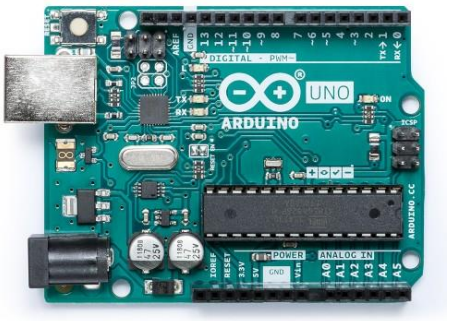

Ilustración 2.4.- Placa Arduino UNO.

Es uno de los modelos más extendidos que actualmente está disponible en su versión Arduino UNO R3, y sus características más destacables son las siguientes:

- Microprocesador: ATmega328
- Voltaje Operativo: 5 V
- Voltaje de Entrada (Recomendado): 7 – 12 V
- Pines de Entradas/Salidas Digital: 14 (De las cuales 6 son salidas PWM)
- Pines de Entradas Análogas: 6
- Memoria Flash: 32 KB (ATmega328) de los cuales 0,5 KB es usado por Bootloader.
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Velocidad del Reloj: 16 MHz

Arduino Pro Mini

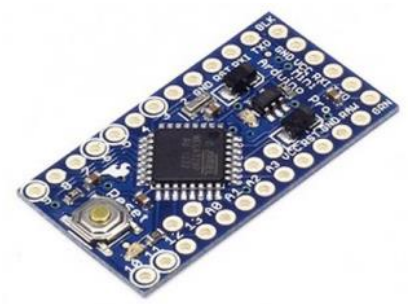

Ilustración 2.5.- Arduino Pro Mini

Es otro modelo disponible para trabajar con Arduino. Entre sus características principales destaca sus reducidas dimensiones y menor consumo energético, manteniendo las mismas capacidades que modelos como la Arduino UNO, lo que la hace ideal para proyectos donde el tamaño es importante. A continuación, sus características más destacables:

- Microprocesador: ATmega328
- Voltaje de Entrada (Recomendado): 3,35 – 12 (Modelo 3,3 V) o 5 – 12 V (modelo 5V)
- Voltaje Operativo: 5 ó 3,3 V (depende del modelo)
- Pines de Entradas/Salidas Digital: 14 (De las cuales 6 son salidas PWM)
- Pines de Entradas Análogas: 6

- Memoria Flash: 32 KB (ATmega328P) de los cuales 0,5 KB es usado por Bootloader.
- SRAM: 2 KB
- EEPROM: 1 KB
- Velocidad del Reloj: 8 MHz (modelo 3,3 V) 16 MHz (modelo 5 V).

2.4.2.4 Raspberry Pi

Raspberry Pi [5] es un microordenador de bajo coste o como se conoce por sus siglas en ingles SBC (*Single Board Computer*). Esto quiere decir que son placas con sistemas embebidos donde se integran todos o la gran mayoría de componentes de un ordenador funcional, pudiéndosele conectar interfaces de entrada y salida como pueden ser teclados, pantallas, altavoces, etc.

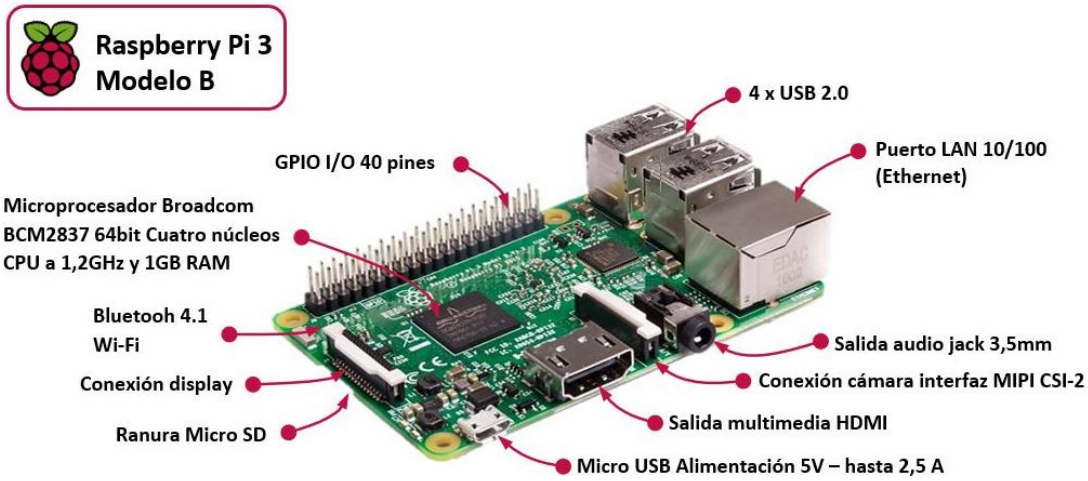

Ilustración 2.6.- Raspberry Pi 3

Para su funcionamiento será necesario instalar un sistema operativo en una tarjeta microSD para las cuales tiene soporte.

A continuación, algunas de sus características más destacables:

- Microprocesador Broadcom BCM2387 a 1,2 GHz de cuatro núcleos ARM Cortex-A53.
- 1 GB LPDDR2.
- Dual Core VideoCore IV ® Multimedia Co-procesador. Proporciona Open GL ES 2.0, OpenVG acelerado por hardware, y 1080p30 H.264 de alto perfil de decodificación.

- Capaz de 1 Gpixel / s, 1.5Gtexel / s o 24 GFLOPs con el filtrado de texturas y la infraestructura DMA.
- Ethernet socket Ethernet 10/100 BaseT.
- 802.11 b / g / n LAN inalámbrica y Bluetooth 4.1 (Classic Bluetooth y LE).
- HDMI rev 1.3 y 1.4.
- RCA compuesto (PAL y NTSC).
- 40-clavijas de 2,54 mm (100 milésimas de pulgada) de expansión.
- Proporciona 27 pines GPIO, así como 3,3 V, +5 V y GND.

2.5 Dispositivos móviles inteligentes

Hoy en día es habitual contar con uno o varios dispositivos denominamos inteligentes (Smartphone). Son dispositivos que se conectan a Internet, y que son capaces de ejecutar tareas que hasta no hace mucho estaban reservadas para los ordenadores personales.

Su implantación ha supuesto un salto enorme en la forma de comunicarnos con el mundo, y han abierto un mar de nuevas posibilidades, que han permitido que mientras que, por ejemplo, antes hiciera falta varios aparatos distintos para llevar a cabo cada tarea, ahora sólo es necesario un único dispositivo.

Un sistema operativo móvil o SO móvil es un sistema operativo que controla un dispositivo móvil. Los dispositivos móviles tienen sus sistemas operativos, tales como Android e iOS, entre otros. Los sistemas operativos móviles son mucho más simples y están más orientados a la conectividad inalámbrica, los formatos multimedia para móviles y las diferentes maneras de introducir información en ellos. Mencionar también que los sistemas operativos utilizados en los dispositivos móviles están basados en el modelo de capas.

2.5.1 Sistemas Operativos para el móvil

Un sistema operativo móvil es un conjunto de programas de bajo nivel que permite la abstracción de las propiedades del hardware especifico que compone un teléfono móvil y provee servicios a las aplicaciones móviles que se ejecutan sobre él. Los sistemas operativos móviles están enfocados en la movilidad, la conectividad inalámbrica, los formatos multimedia para móviles y la optimización de forma óptima los recursos.

En la actualidad estamos rodeados de multitud de *smartphones de* muchas marcas y diseños, pero todos tienen algo en común, y es que estos dispositivos cuentan con un

sistema operativo. En la actualidad hay dos OS que destacan sobre el resto y tienen la mayor cuota de mercado, son Android y iOS.

2.5.1.1 Android

El sistema operativo Android [6] es sin duda uno de los líderes en cuanto a SO móviles. Sus orígenes se remontan al 2003, cuando la empresa Android Inc. decide crearlo como sistema operativo que daría vida a cámaras digitales. En 2005 es comprado por Google, posteriormente modificado para ser utilizado en teléfonos móviles inteligentes y fue lanzado al público en 2007, cuando Google fundó la Open Handset Alliance formada por un grupo de 78 compañías de hardware, software y telecomunicaciones dedicadas al desarrollo de estándares abiertos para dispositivos móviles. Juntos desarrollaron Android, la primera plataforma móvil completa, abierta y libre. En la actualidad este SO da vida a multitud de dispositivos móviles a parte de los teléfonos, como son relojes, televisores, PCs y hasta neveras.

Android está basado en el kernel de Linux y la programación de las aplicaciones se escriben y desarrollan en Java, aunque con APIs propias, la interacción entre el kernel y las aplicaciones se lleva a cabo mediante una máquina virtual (capa intermedia) que hace posible este proceso.

Android destaca por ser un software libre y de código abierto, permite que multitud de empresas y fabricantes puedan modificarlo y adaptarlo según sus necesidades, lo que hace que sea una de sus grandes virtudes, aunque en mucho de los dispositivos vendidos, gran parte del software incluido es software propietario y de código cerrado.

2.5.1.2 iOS

iOS, en sus orígenes iPhone OS, es propiedad de Apple Inc. y es el SO que da vida a los dispositivos móviles de la compañía y es el segundo sistema para teléfonos móviles inteligentes más usado en el mundo detrás de Android.

Destaca por ser un sistema operativo cerrado y controlado por su fabricante, está basado en el Darwin, o lo que es lo mismo, el kernel del sistema operativo para ordenadores de la compañía, Mac OS X. Destaca por su gran optimización de los recursos hardware, su sencillez de su interfaz y la fluidez.

En la actualidad es el SO que da vida a los iPhones, iPad, iPod y demás dispositivos móviles de Apple. En concreto, las aplicaciones para iOS se escriben y desarrollan en lenguaje Swift usando el IDE Xcode.

2.5.2 Componentes de un Smartphone

La evolución del hardware de los *Smartphone* viene dada por la miniaturización de los componentes electrónicos que lo forman y una mejora en el proceso de producción/fabricación con menor consumo y mayores velocidades.

Las velocidades de micro-procesamiento guardan una relación directa con el número de transistores incluidos sobre el chip, y cuanto más pequeño sea el transistor, mayor cantidad de ellos podrán ser empleados dentro de un mismo chip.

2.5.2.1 Procesadores ARM

ARM es la responsable de la rápida evolución en los últimos años de los dispositivos móviles en una gran parte de los casos. Los procesadores ARM se basan en el modelo RISC y están licenciados por la compañía británica ARM Holdings, que realizó su introducción en el mercado en su primer modelo en el año 1985. Desde entonces, la tecnología ARM se ha actualizado de forma constante hasta alcanzar la madurez suficiente para convertirse en la arquitectura de 32 bits más exitosa del mundo. De hecho, cerca del 75 % de los procesadores de 32 bits montados en cualquier tipo de dispositivo poseen este chip en su núcleo (*tablets*, videoconsolas, *routers*, etc.) y está presente en más del 95 % de los *Smartphone* actuales.

El diseño de los procesadores de los Smartphone está muy relacionado con el desarrollo del concepto multi-núcleo y disminución del proceso de fabricación en nm. Por ejemplo, 45nm de tamaño en el proceso de fabricación es más pequeño que 65nm, y a menor tamaño menos calor y menor consumo eléctrico, y al ser más pequeños permiten un mayor número de ellos en el chip y se gana un mejor rendimiento.

El microprocesador es la parte más importante de cualquier equipo electrónico, y desde hace unos años la tendencia es duplicar, triplicar e incluso cuadriplicar el núcleo de dicho microprocesador.

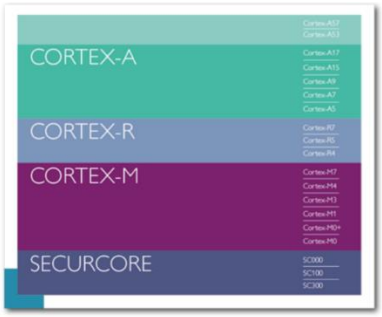

Ilustración 2.7: Familia de Procesadores ARM Cortex

2.5.2.2 GPU

La unidad de procesamiento gráfico o GPU (*Graphics Processing Unit*) es un coprocesador dedicado al procesamiento de gráficos u operaciones de coma flotante. Existe básicamente para aligerar la carga de trabajo en videojuegos o en aplicaciones 3D interactivas. De esta forma, mientras gran parte de lo relacionado con los gráficos se procesa en la GPU, la unidad central de procesamiento (CPU) puede dedicarse a otro tipo de cálculos (como la inteligencia artificial o los cálculos mecánicos en el caso de los videojuegos).

La GPU implementa ciertas operaciones gráficas llamadas primitivas optimizadas para el procesamiento gráfico. Una de las primitivas más comunes para el procesamiento gráfico en 3D es el *antialiasing* o suavizado de bordes (evita el *aliasing* que es un efecto visual tipo "sierra" o "escalón"). Las GPU actualmente disponen de gran cantidad de primitivas, buscando mayor realismo en los efectos.

Las GPU modernas son descendientes de los chips gráficos monolíticos (circuitos integrados que están fabricados en un solo monocristal, habitualmente de silicio) de finales de la década de 1970 y 1980.

La GPU en los *Smartphone* está especializada en mostrar los gráficos de la interfaz de usuario, efectos 3D y 2D, reproducción de vídeo en HD Ready (720p) o full HD (1080p), reproducción de gráficos avanzados 3D y 2D en videojuegos.

2.5.2.3 Memoria RAM

La memoria RAM es uno de los componentes críticos del móvil, junto con los núcleos de procesamiento de la CPU y GPU. Sin RAM, cualquier tipo de sistema de computación sería incapaz de realizar tareas básicas y acceder a los archivos de su memoria secundaria sería inaceptablemente lento.

Los archivos críticos que necesita el procesador se almacenan en la memoria RAM, que siempre ha de estar lista en espera de ser leída o escrita. Estos archivos críticos para el dispositivo pueden ser: los componentes del sistema operativo, datos de aplicaciones y gráficos de un juego, o en general cualquier cosa a la que se deba acceder a velocidades mayores que las de acceso a memorias de almacenamiento secundario.

El tipo de memoria RAM que se utiliza en móviles *Smartphone* es, técnicamente, DRAM (RAM Dinámica). La estructura de la DRAM es tal que cada condensador de la placa de RAM almacena un bit, y estos condensadores requieren de un constante "refresco" o actualización de los datos que están almacenados. El contenido del módulo de memoria DRAM se puede cambiar rápida y fácilmente para almacenar diferentes datos.

2.5.2.4 Pantalla Táctil

La tecnología actual para la función táctil de las pantallas se conoce como capacitiva. Se basa en conectar el toque del usuario a través de una lámina protectora que se encuentra al frente de la pantalla, de tal manera que cada unidad pueda instalarse por detrás de materiales de protección o vidrio anti-roturas. El sistema integrado resistente a golpes, rayones y vandalismo, además de no verse afectado por agentes tales como humedad, calor, lluvia, nieve o granizo y líquidos corrosivos. El *touchscreen* -sólido y estable- y su controladora, aumentan el nivel de calidad y vida útil del equipo ofreciendo una respuesta rápida y libre de error, además de requerir bajos niveles de mantenimiento y recalibración.

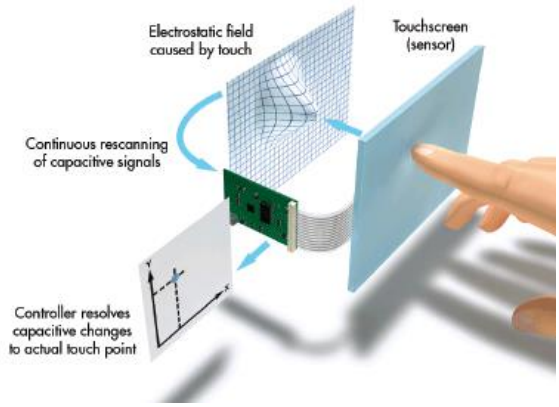

Ilustración 2.8: Funcionamiento de una pantalla táctil capacitiva

El componente principal que proporciona el campo electrostático es transparente, por lo que en la mayoría de las pantallas táctiles no es posible ver la cuadrícula de electrodo capacitivo en la capa de sistema integrado.

Las principales tecnologías de pantallas táctiles son:

- LCD: Una pantalla de cristal líquido o LCD (siglas del inglés *Liquid Crystal Display*) es una pantalla delgada y plana formada por un número de píxeles en color o monocromos colocados delante de una fuente de luz o reflectora.

- OLED: Acrónimo de *Organic Light-Emitting Diode*. En este caso lo que se usa son emisores LED, pero en cuyos componentes se incluye el carbono y de ahí la palabra orgánico en su nombre. Estas pantallas prometen menor consumo, mejores tiempos de respuesta y colores más realistas.

2.5.2.5 Sensores

Los dispositivos móviles inteligentes están equipados con algunos sensores, siendo los más comunes los siguientes:

- Acelerómetro: Se denomina acelerómetro a cualquier dispositivo capaz de detectar las fuerzas de aceleración a las que se ve sometida una masa. Se puede utilizar para detectar la inclinación, la vibración, el movimiento, el giro y el choque. Actualmente es posible construir acelerómetros de tres ejes (X, Y, Z) en un sólo chip de silicio, incluyendo en el mismo la parte electrónica que se encarga de procesar las señales.

- Giroscopio: El giróscopo o giroscopio es un dispositivo mecánico formado esencialmente por un cuerpo con simetría de rotación que gira alrededor de su eje de simetría. Detecta la rotación a la que se somete el dispositivo.

- Brújula Digital: La brújula es un instrumento que sirve de orientación y nos permite conocer la dirección del *Smartphone* en relación a los polos magnéticos de la Tierra o al norte geográfico.

- Sensor de Luz: Este pequeño componente se encargará de recoger información sobre la luz ambiental y pasársela a nuestro dispositivo para que éste ajuste el brillo de nuestra pantalla con el fin de hacer que la visibilidad y comodidad de uso sea lo más satisfactoria posible.

- Sensor de Proximidad: El sensor de proximidad es un transductor que detecta objetos o señales que se encuentran cerca del elemento sensor. Estos sensores de proximidad se basan en un LED infrarrojo y también en un receptor IR.

2.6 Plataformas e-health open source

Hasta no hace mucho tiempo, pensar en salud y en sistemas biomédicos hacía pensar en hospitales o centros especializados con grandes y costosos sistemas de monitorización de parámetros para nuestro cuerpo para la prueba y diagnóstico de pacientes, pero esta realidad ha cambiado con las nuevas tecnologías y las posibilidades que nos ofrecen.

IoT permite que, por ejemplo, las personas interesadas en estos sistemas de monitorización puedan disponer de uno por no mucho dinero apoyados por la comunidad maker que ofrece muchísimas posibilidades.

En este apartado, se detallarán algunas de esas plataformas que disponen de más o menos sensores para la obtención de información biomédica y que están disponibles en el mercado:

2.6.1 Bitalino

Ilustración 2.9.- Bitalino

Bitalino [7] es una plataforma open-source muy popular para desarrolladores que cuenta con una gran variedad de sensores biomédicos, algunos de estos son, sensor de electromiografía (EMG), sensor de electrocardiograma (ECG), sensor LUX, acelerómetros y sensor electro dermal (EDA). Esta plataforma está basada en un microcontrolador Atmega328 y ofrece soporte Bluethooth.

Es una alternativa de bajo coste y que ofrece una gran versatilidad a la hora de trabajar. Otra de sus características es que trabaja por módulos independientes que se pueden acoplar según las necesidades requeridas, haciéndolo muy versátil.

2.6.2 E-Health Sensor Platform

E-health es una plataforma open-source [8] disponible en el mercado. Una de sus principales características es que ofrece una gran cantidad de sensores biomédicos que se pueden conectar a ella. Además, es compatible tanto con Arduino como con Raspberry Pi, lo que la hace muy versátil e ideal para el desarrollo hardware.

Los sensores soportados por E-Health son los siguientes:

- Sensor de pulso
- Sensor de oxígeno en sangre
- Sensor de temperatura
- Sensor ECG

- Sensor de flujo de aire
- Sensor de respuesta galvánica (GSR)
- Glucómetro
- Acelerómetro
- Sensor presión en sangre
- Sensor EMG

Todos los sensores son de uso no invasivos y están conectados a pines específicos en la placa los cuales permiten una monitorización en tiempo real.

Otra de las posibilidades que ofrece esta plataforma, es poder conectar la placa a otros Shields BLE, Wi-Fi, GPRS, 3G o ZigBee, para poder almacenar o enviar la comunicación de forma remota y la visualización de la información en una pantalla LCD que ofrece el propio fabricante y que se comunica por una interfaz serie muy sencilla.

Capítulo 3

Descripción del sistema

3.1 Introducción

En el capítulo anterior se presentaron algunas tecnologías y dispositivos relevantes disponibles en el mercado, de las cuales, se han elegido para el desarrollo de este proyecto la placa de sensores e-health de cooking hacks, el microcontrolador Arduino Mini Pro, el módulo ESP32, y Android como sistema operativo sobre el que estará construida la aplicación.

A lo largo de este capítulo se detallan más a fondo las tecnologías y dispositivos elegidos. También se expone y se ofrece información del software necesario para la utilización de algunas de las opciones elegidas para el proyecto.

3.2 Arquitectura de hardware y software del sistema

El sistema se compone de dos partes muy bien diferenciadas: la arquitectura hardware y la configuración software.

3.2.1 Arquitectura Hardware

La arquitectura está formada por todos los elementos que intervienen desde la medición de los sensores hasta el dispositivo móvil que controla el sistema. Los diferentes componentes se presentan a continuación y serán desarrollados en profundidad en este capítulo:

- Plataforma de sensores *e-Health* de Cooking Hacks: Funciona como *shield* para la placa Arduino Pro Mini a la cual van conectados los sensores y permiten la recepción de las señales.
- Arduino Pro Mini: Placa que se encarga de procesar la información de los sensores y que se comunica por medio de UART con un microcontrolador ESP32.
- ESP32: Microcontrolador que se comunica con el dispositivo móvil inteligente por medio de la tecnología *Bluethooth Low Energy*.
- Dispositivo móvil inteligente: Parte del sistema que permite la gestión e interfaz de usuario.

3.2.2 Arquitectura Software

La configuración software para el proyecto se realiza en diferentes plataformas y entornos de desarrollos según lo requiere los dispositivos que componen la arquitectura hardware descrita en el apartado anterior. A continuación, se presenta los elementos que componen la arquitectura software:

- Aplicación para dispositivos móviles: Aplicación para dispositivos Android y programada en su entorno de desarrollo, *Android Studio*.
- Programas o *Sketchs* para el Arduino Pro Mini y el microcontrolador ESP32: Ambos programados en el IDE de Arduino para el funcionamiento de las placas.
- Base de datos: Donde se almacena y gestiona la información para la realización de las terapias.
- Servicios WEB: Permite gestionar los accesos y el envió de información entre la aplicación y la base de datos de forma segura.

3.3 Plataforma de sensores e-Health de Cooking Hacks

Para el desarrollo del proyecto se emplea el kit de desarrollo *e-Health* de Cooking Hacks (ver Ilustración 3.1), en su versión 2.0. Se compone de un *Shield* completamente compatible con Arduino y Raspberry Pi. A continuación, una lista con las características de esta plataforma.

Ilustración 3.1.- Kit e-health utilizado.

- Se compone de ocho sensores médicos no invasivos y un sensor medico invasivo.
- Medición de glucosa y almacenamiento de los valores.
- Medición de señales de electrocardiograma (ECG).
- Medición de señales electromiografía (EMG).
- Medición de flujo de aire
- Medición de la temperatura corporal.
- Medición de la respuesta galvánica en la piel.
- Detención de la posición del cuerpo.
- Medición del pulso y oxígeno en la sangre.
- Medición de la presión sanguínea.
- Visualización de información en pantalla.
- Compatibilidad y comunicación con otros dispositivos mediante UART.

El suministro eléctrico de la placa puede hacerse mediante el PC con una conexión USB, alimentarse del Arduino, Raspberry Pi o directamente a una fuente externa de energía de 12 V y 2 A.

3.3.1 Librería

La plataforma cuenta con una librería para el IDE de Arduino en C++que permite obtener fácilmente la medida de lo sensores y enviarla a otros dispositivos. En el caso de Raspberry Pi, el fabricante usa la librería en ArduPi para su correcto funcionamiento.

3.3.1.1 Uso de la librería en Arduino

La librería que incluye la plataforma e-health es de alto nivel y permite fácilmente la utilización de la placa y los sensores. Para su implementación y uso hace falta descargar el archivo .ZIP desde la web de Cooking Hacks [9], la cual tiende dentro dos carpetas llamadas "eHealth" y "PinChangeInt", esta última es necesaria solo en caso se utiliza el pulsioxímetro. Posteriormente, se copian estos archivos en la carpeta "libraries" del IDE de Arduino.

3.3.2 Sensores

A continuación, se describen los sensores que incluye la plataforma con las características más destacadas.

3.3.2.1 Pulsómetro y oxígeno en sangre (SPO2)

El pulsioxímetro es un sensor no invasivo que permite obtener la saturación de oxigeno arterial de la hemoglobina funcional de la sangre. La saturación de oxigeno se define como la cantidad de oxígeno disuelto en la sangre basado en la detención de hemoglobina y desoxihemoglobina.

El medidor de pulso cardiaco es útil en todos los casos y configuración, incluso cuando la oxigenación del usuario o paciente es inestable. Para la utilización del sensor basta con conectarlo al Arduino o Raspberry Pi sin necesidad de baterías externas.

La utilización del sensor de hace poniendo la pinza en el dedo índice como muestra la siguiente imagen y para la gestión de los datos es necesario las librerías eHealth y PinChangeInt.

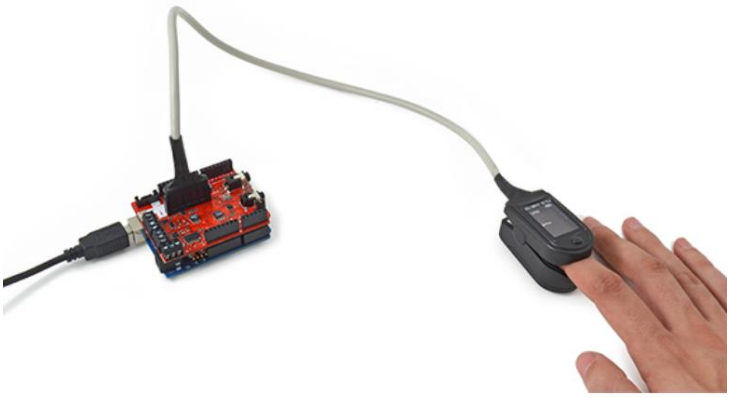

Ilustración 3.2.- Pulsioxímetro

3.3.2.2 Sensor de electrocardiograma ECG

El electrocardiograma es una herramienta de diagnóstico que permite evaluar el estado muscular y eléctrico del corazón. En la actualidad, el electrocardiograma es muy utilizado para el diagnóstico y seguimiento de enfermedades y patologías cardiacas.

Para la medición del electrocardiograma es necesario la utilización de tres electrodos que van conectados a la placa y la utilización de las librerías eHealth.

Ilustración 3.3.- Sensores ECG

3.3.2.3 Sensor de flujo de aire

El ritmo respiratorio es un parámetro importante y que puede ayudar a determinar el estado en que se encuentra un paciente, una alteración de este parámetro puede indicar, por ejemplo, alguna alteración psicológica o una situación de inestabilidad del paciente. En este caso la plataforma de desarrollo eHealth cuenta con un sensor nasal de flujo de aire que permite medir la frecuencia de la respiración de un paciente.

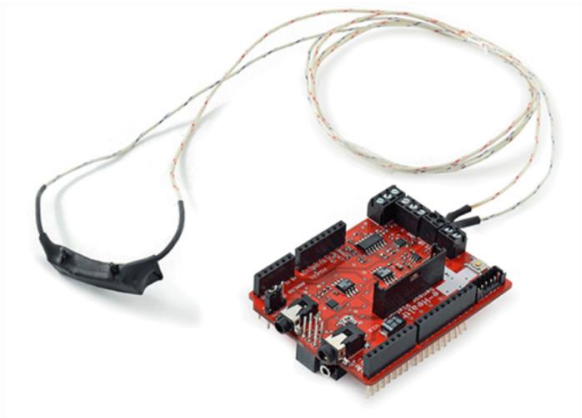

Ilustración 3.4.- Sensor de flujo de aire

Como se muestra en la figura, el sensor se compone de un dispositivo ajustable que se pone en la nariz del paciente y mediante dos hilos se conecta a la placa. Para la gestión de la información que obtienen los sensores es necesario la librería "eHealth".

3.3.2.4 Sensor de temperatura corporal

La medición de la temperatura corporal es un parámetro en el cual influyen diferentes factores como puede ser el lugar del cuerpo donde se lleve a cabo la medida, la hora del día, el tipo de paciente o la edad. Al ser una medida muy importante para conocer el estado de una persona el fabricante, en este caso, nos ofrece unos valores para determinar el estado del paciente según la temperatura corporal que registre.

- Hipotermia: valores < 35,0 ºC
- Normal: 36.5-37,5 ºC
- Fiebre: Valores > 37,5-38,3 ºC
- Hiperpirexia: Valores > 40-41,5 ºC

En la utilización del sensor, y dependiendo de la precisión requerida, la plataforma permite recalibrar el sensor con el objetivo de tener los valores más exactos posibles. En este caso es necesario un polímetro y requiere medir las resistencias una serie de resistencias y actualizar sus valores en la librería eHealth función "getTemperature".

```
231
232   //!*****************************************************************************
233   //!  Name:    getTemperature()                                                  *
234   //!  Description: Returns the corporal temperature.                             *
235   //!  Param : void                                                               *
236   //!  Returns: float with the corporal temperature value.                        *
237   //!  Example: float temperature = eHealth.getTemperature();                     *
238   //!*****************************************************************************
239
240   float eHealthClass::getTemperature(void)
241   {
242     //Local variables
243     float Temperature; //Corporal Temperature
244     float Resistance;  //Resistance of sensor.
245     float ganancia=5.0;
246     float Vcc=3.3;
247     float RefTension=3.0; // Voltage Reference of Wheatstone bridge.
248     float Ra=4700.0; //Wheatstone bridge resistance.
249     float Rc=4700.0; //Wheatstone bridge resistance.
250     float Rb=821.0; //Wheatstone bridge resistance.
251     int sensorValue = analogRead(A3);
252
```

Ilustración 3.5.- Fragmento de código librerías e-Health

En la figura (ver Ilustración 3.5) se observa los valores de las resistencias Ra, Rc, Rb que se deben calcular y RefTension, que es el voltaje de referencia del puente de Wheatstone, en este caso es de 3.0V.

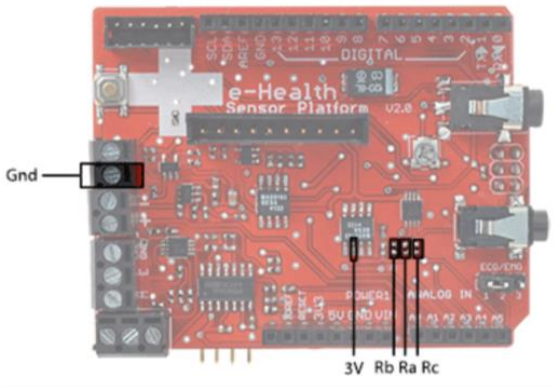

Ilustración 3.6.- Ubicación resistencias del sensor de temperatura

Con la ayuda de un polímetro se calculan los valores de las resistencias referidas a la masa GND y se actualizan sus valores en el fichero eHealth.

Para utilizar el sensor, hace falta ubicar el contacto en una de las zonas del cuerpo, puede ser la mano o un pie y para la gestión de los datos es necesario la librería eHealth.

3.3.2.5 Presión sanguínea

La monitorización de la presión sanguínea regularmente puede ser necesaria para algunos pacientes, ya que es un parámetro que de ser muy elevado puede presentar problemas como la hipertensión o ataque al corazón, entre otras enfermedades. Para la plataforma eHealtlh, se utiliza un sistema de medida Kodea que se conecta a la placa y mediante el uso de la librería eHealth para Arduino permite la lectura y gestión de los datos.

Ilustración 3.7.- Sensor de presión sanguínea

3.3.2.6 Sensor de posición del cuerpo

Consta de una banda que se ajusta en el pecho que lleva dentro un acelerómetro que ayuda a detectar la posición del cuerpo mediante un sistema de tres ejes. El número de posiciones que detecta son cinco (sentado/de pie, tumbado boca arriba, tumbado boca abajo, derecha, izquierda).

El sensor está conectado a la placa por medio de una conexión de 6 hilos y para la gestión de los datos se requiere la librería eHealth y las funciones que provee.

Ilustración 3.8.- Sensor de posición

3.3.2.7 Sensor de respuesta galvánica en la piel

La respuesta galvánica en la piel es un método que permite medir la conductancia eléctrica de la piel, la cual varia con la humedad. Esto es muy interesante si tenemos en cuenta la relación que existe entre la sudoración de la piel y el sistema nervioso. La conductividad eléctrica de la piel y el nivel de sudoración variara según nuestro estado emocional o de excitación. El principio en el que se basa este sistema es en la variación de la conductancia eléctrica con la sudoración, de tal forma que, a mayor sudoración los

valores de la conductancia eléctrica o resistencia son inferiores y si la piel se encuentra más seca, los valores de la resistencia son más altos.

El sensor que disponemos es no invasivo y permite obtener la medida GSR mediante dos puntos que estarán en contacto con la piel, concretamente dos dedos y que funciona básicamente como un óhmetro.

Ilustración 3.9.- Sensor GSR y como su colocación

3.3.2.8 Electromiograma

El electromiograma es una medida de la actividad muscular, su variación entre un estado de reposo y una contracción. Son señales que se miden y utilizan con frecuenta en aplicaciones médicas y permite evaluar el estado neuromuscular para detectar enfermedades como el desorden de control motor. En este caso, el sistema de medida se compone por electrodos, sensores no invasivos, que se conectan a la piel y permite conocer el estado y actividad eléctrica de un musculo en concreto durante un tiempo determinado.

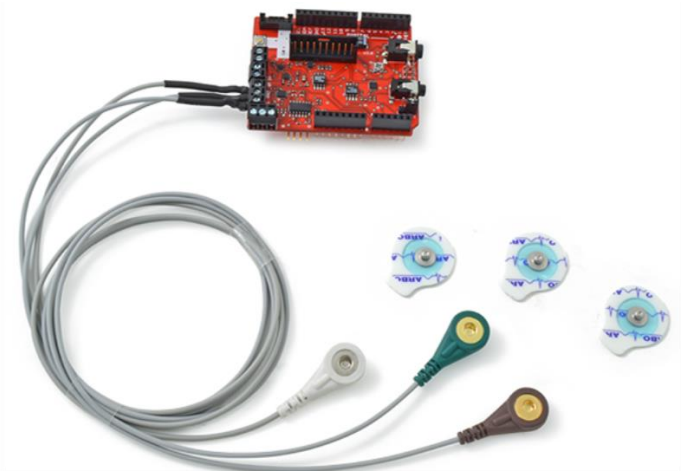

Ilustración 3.10.- Sensor de Electromiograma

Para su uso, se requiere limpiar la piel donde se ubicarán los electrodos y a ser posible aplicar gel para un mejor contacto. A continuación, en la imagen (ver Ilustración 3.11), un ejemplo de cómo se conectarían los electrodos.

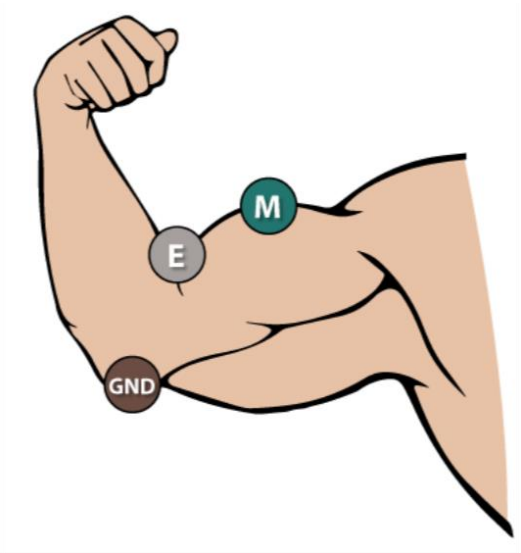

Ilustración 3.11.-Colocación de los electrodos de electromiograma

3.3.2.9 Glucómetro

Es un dispositivo que ayuda a medir el valor aproximado de la concentración de glucosa en la sangre. Este es el único sistema de medida invasivo que dispone la plataforma, en este caso, es necesario una gota de sangre que se deposita en una tira para poner en un lector de glucosa como se muestra en la imagen (ver Ilustración 3.12).

Ilustración 3.12.- Glucómetro

El dispositivo permite almacenar medidas, borrar y cambiar la configuración según requiera el usuario. Para la utilización de los datos de medida, es necesario configurar el almacenamiento de medidas previamente y después conectar mediante un cable a la placa eHealth y por medio de la librería del mismo nombre gestionar los datos recibidos en Arduino.

Ilustración 3.13.- Placa e-Health conectada al glucómetro

3.4 Arduino Pro Mini

Arduino Pro Mini es un hardware Open-Source, el fabricante de Arduino tiene disponibles la documentación y esquemáticos necesarios para la fabricación de una placa con las mismas características o modificada. Para el desarrollo del proyecto se utiliza una placa Arduino Pro Mini fabricada por Arduino.

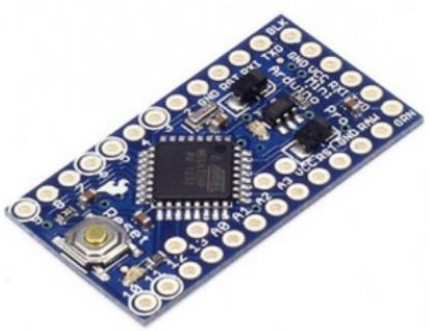

Ilustración 3.14.- Arduino Pro Mini

La alimentación de esta placa puede hacerse por conexión de un cable FTDI, mediante el acoplamiento de una placa de arranque conectado al cabecero de seis pines o por conexión de 3.3V- 5V (dependerá del modelo) en los pines de entrada estándar que ofrece.

El microcontrolador encargado de dar vida al Arduino Pro Mini es un ATmega328P que dispone de 32kB de memoria flash para almacenar código. Además, cuenta con un 2 kB de SRAM y 1 kB de EEPROM.

3.4.1 Entradas y salidas

Esta placa cuenta con 14 pines digitales que pueden ser configurados como pines de entrada y salida indistintamente según se requiera mediante las funciones pinMode, digitalWrite y digitalRead. Cada pin cuenta con resistencias internas de pull-up que varían entre 20-50 kOhms. Algunos pines tienen características específicas como son:

- Serial: pin 0(RX) y pin 1(TX) usados para enviar y recibir información por comunicación serial TTL.
- Interrupciones externas: pines 2 y 3. Estos pines pueden ser configurados para generar interrupciones, para esta función es necesario el uso de la librería attachInterrupt.
- PWM: pines 3, 5, 6, 9, 10 y 11. Proporciona 8-bit PWM con la función analogWhite.
- Comunicaciones SPI: pines 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). La configuración de estos pines permite la comunicación SPI.
- Led: pin 13.
- Dispone de 8 salidas analógicas, cada una de ellas con una resolución de 10 bits.
- Comunicación I2C: pines A4 y A5. Soporte para este protocolo de comunicación usando la librería Wire.

- Reset: Disponible para hacer reset de la placa.

3.4.2 Comunicaciones

EL Arduino Pro Mini ofrece facilidades para la comunicación con otros dispositivos, PCs o microcontroladores. Dispone de una comunicación UART habilitada en los pines 0 y 1, pudiéndosele configurar alguna más mediante software. También dispone de un monitor serie que permite enviar, recibir y visualizar datos en pantalla. Además, el microcontrolador ofrece soporte para los protocolos de comunicación I^2C y SPI.

3.4.3 Programación y entorno de desarrollo

Para la programación de la placa el fabricante ofrece el IDE oficial de Arduino que se encuentra disponible en la web. Una herramienta basada en el lenguaje C, que se caracteriza por ser ligereza y sencillez.

Ilustración 3.15.- IDE Arduino

Al ser una plataforma *Open Source* cuenta con una gran comunidad detrás y el aporte de multitud de librerías, shields y repositorios para trabajar. Para instalar nuevas librerías, hay que guardar los archivos dentro de la ruta C:\Users\user\Documents\Arduino\libraries.

Los proyectos o Sketch (como se conocen popularmente en Arduino), son los ficheros que compilara el IDE para cargarlos en el microcontrolador.

3.5 ESP32

El ESP32 es un SoC (System on Chip) de sus siglas en inglés, es creado y desarrollado por la empresa Espressif Systems y fabricado por TSMC (Taiwan Semiconductor Manufacturing Company Limited) con tecnología de 40nm. Es el sucesor del SoC ESP82266. Integra en su interior un procesador de 32 bits de doble núcleo Tensilica LX6, que trabaja normalmente a 160Mhz pudiendo llegar a los 240Mhz, dispone de una memoria ROM de 448 kB y SRAM de 520kB.

Es un microcontrolador orientado y destinado para el uso de IoT (Internt of Things). Entre sus características más destacables se encuentra la conectividad que ofrece, el número de puertos entrada y salida, las comunicaciones, su modo ultra bajo consumo y las diferentes plataformas soportadas para su programación.

3.5.1 Entradas y salidas

Dispone de 48 pines que se ofrecen las siguientes características:

- 18 pines ADC de 12 bits
- pines DAC de 8 bits
- 10 pines de sensores de contacto
- 16 pines para PWM
- 20 pines entradas/salidas digitales

3.5.2 Comunicaciones

El ESP32 ofrece muchas variantes y facilidades de comunicación, siendo uno de los apartados más destacables en este pequeño microcontrolador ya que ofrece soporte para:

- Conectividad Wi-Fi: 802.11 b/g/n, Wifi Direct (P2P), P2P Discovery and P2P power management
- Conectividad Bluetooth (v4.2): BR/EDR y Bluetooth Low Energy (BLE)
- Comunicación SPI de hasta 4 conexiones
- Interfaz I2S de hasta 2
- Interfaz I2C de hasta 2
- Soporte para UART de hasta 3
- CAN bus 2.0
- Ethernet

3.5.3 Programación y entorno de desarrollo

En este apartado, las posibilidades para programar el dispositivo son variadas, se encuentran disponibles diferentes entornos de desarrollo y dependerá de las necesidades que requiera el proyecto a realizar la elección entre uno u otro. Los entornos más destacables y que ofrecen un gran soporte son:

- Arduino IDE por medio del Core ESP32 Arduino.
- Espressif IoT Development Framework, es el entorno de desarrollo nativo que ofrece el fabricante.
- Espruino, JavaScript SDK.
- MicroPython.

3.5.3.1 Arduino Core para ESP32

La instalación del Core para Arduino se encuentra disponible para dispositivos Windows, Mac, Linux, Fedora entre otros. Para el desarrollo de este proyecto se utiliza un PC con el sistema operativo Windows 10. El proceso para la instalación del Core en Windows es el siguiente, pero es similar en otras plataformas.

1. En primer lugar, es necesario la instalación del IDE de Arduino que se encuentra disponible en la web Arduino.cc
2. En segundo lugar, se requiere la descarga e instalación de Git disponible en git.scm.com.
3. Una vez instalado, se ejecuta el programa y da la opción de "Clone Existing Repository"

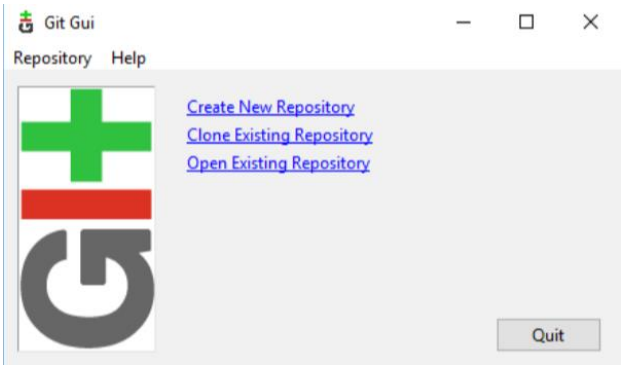

Ilustración 3.16.- Pantalla 1 instalación Core ESP32 para Arduino IDE

4. Posteriormente, aparece una ventana donde se selecciona un directorio en el PC, en mi caso el C:\Users\bflor\Documents\Arduino. Se selección en "Source Location" https://github.com/espressif/arduino-esp32.git como se muestra en la imagen.

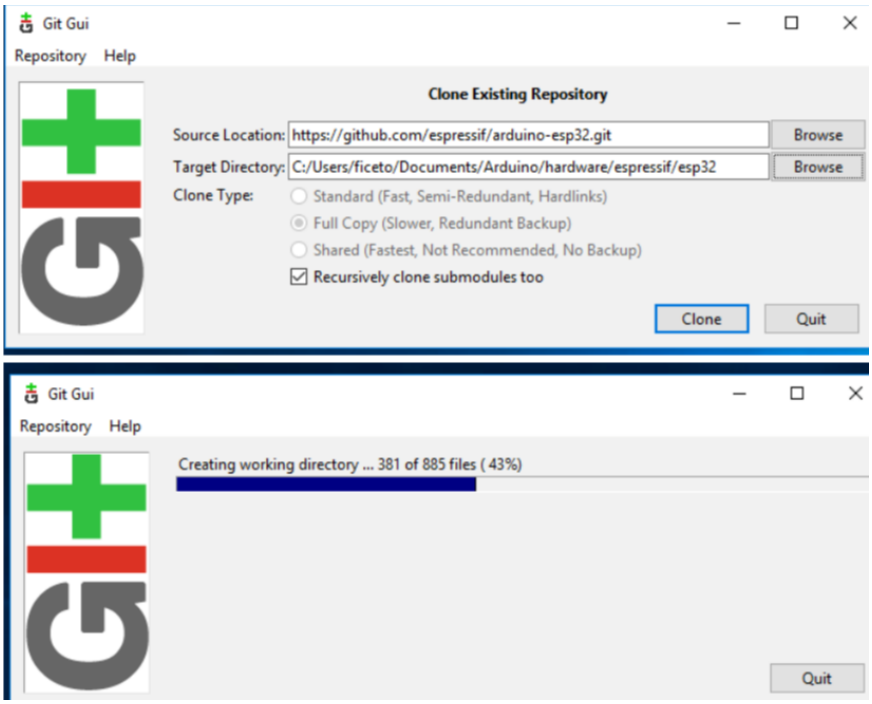

Ilustración 3.17.- Pantalla 2 instalación Core ESP32 para Arduino IDE

5. En el caso que nos ocupa es necesario ir a la dirección:

C:\Users\bflor\Documents\Arduino\hardware\espressif\esp32, y ejecutar el archivo submodule Update. Después, ir a:

C:\Users\bflor\Documents\Arduino\hardware\espressif\esp32\tools, y hacer doble click en el archivo get.exe.

6. Una vez acabado este proceso, se procede a verificar la instalación, para esto, es necesario abrir el IDE Arduino, una vez dentro, en la barra de tareas pulsar en herramientas -> tarjetas y seleccionar la tarjeta ESP32.
7. Para finalizar, se configuración del puerto COM y se compilan los proyectos como si de un Sketch de Arduino se tratara.

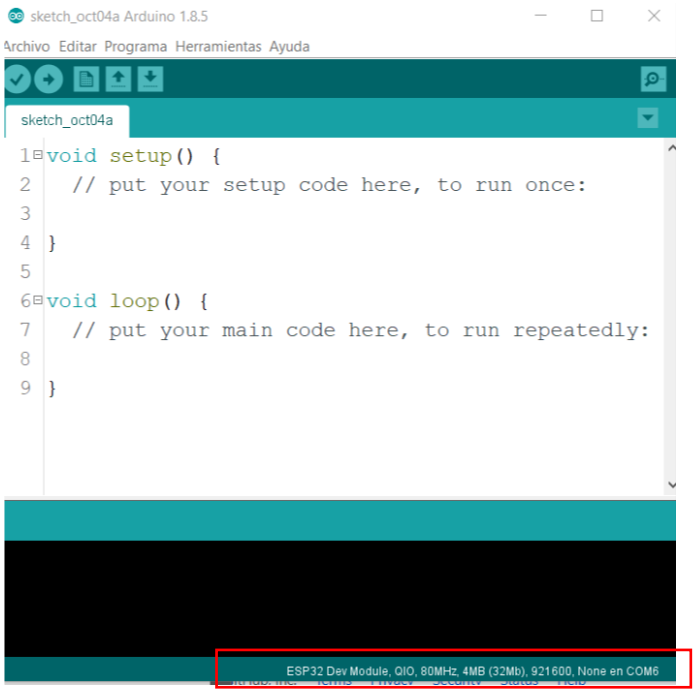

Ilustración 3.18.- Arduino IDE configurado para el ESP32

3.6 Sistema Operativo Android

El Sistema Operativo Android (Android OS) es diseñado y empleado en dispositivos móviles, por lo generan con pantalla táctil, como teléfonos móviles inteligentes, tabletas, relojes inteligentes, televisores y coches.

En sus orígenes, Android OS fue creado por Android Inc, posteriormente, la empresa fue comprada por Google en el año 2005, actual propietaria y desarrolladora del sistema. Android está basado en núcleo Linux y fue presentado oficialmente al público en el año 2007. El primer móvil en salir al mercado bajo Android fue el HCT Dream en el año 2008. En la actualidad, es el sistema operativo más utilizado del mundo con una cuota de mercado cercana al 80% [10].

Android está basado en Open Source, esto significa que el sistema y el software en general está disponible y puede ser consultado por cualquiera que lo desee. Bajo esta filosofía, la versión básica del sistema operativo que ofrece el fabricante es conocida como AOSP (Android Open Source Project). El sistema operativo tiene como objetico la universalidad y acceso a los teléfonos móviles inteligentes en todo el mundo, por este motivo, en 2017 Google presento otra versión del sistema llamada Android Go que se caracteriza por ser más ligera y consumir menos recursos que la versión estándar.

3.6.1 Características

A continuación, algunas de las características más destacables del sistema operativo.

- Es de código abierto.
- Núcleo del sistema basado en el Kernel de Linux
- Sistema adaptable y con soporte para multitud de pantallas y resoluciones disponibles en el mercado.
- Utiliza SQLite para el almacenamiento de datos
- Soporte para diferentes formas de mensajería como SMS, MMS, FCM o GCM.
- Navegador web basado en el motor de renderizado de código abierto WebKit.
- Soporte para Java, en este caso, el sistema no cuenta con una máquina virtual Java en la plataforma y para poder ejecutar bytecode Java el sistema cuenta con una máquina virtual Dalvik especialmente diseñada para Android que se encarga de compilar el código. Dalvik fue utilizado hasta la versión 5.0 cuando fue reemplazado por Android Runtime (ART).
- Soporte de múltiples formatos multimedia como son WebM, AMR, AAC, HE-ACC, JPEG, WAV entre muchos otros.
- Soporte para HTML, HTMIL5, Adobe Flash Player.
- Soporte para de hardware adicional como cámaras, pantallas táctiles, sistemas de navegación GPS, acelerómetros, termómetros, acelerómetros, giroscopios, etc.
- Cuenta con el entorno de desarrollo oficial Android Studio para la emulación, creación de apps, análisis de rendimiento entre otras funciones. En sus orígenes el software utilizado era Eclipse.
- Tienda de aplicaciones Google Play, cuenta con más de un millón de aplicaciones gratis y de pago.
- Soporte para multitarea real en la ejecución del sistema y aplicaciones.
- Soporte para tecnología Bluetooth, Wi-Fi, redes de datos móviles.

3.6.2 Arquitectura del sistema

Los componentes principales que conforman el sistema operativo Android son:

- Aplicaciones: Pueden estar escritas en los lenguajes de programación Java o Kotlin (soportado oficialmente por Google a partir de la API 28 del sistema). El sistema por defecto cuenta con aplicaciones base de correo, mensajes, contactos entre otras.
- Framework o marco de trabajo de aplicaciones: Los desarrolladores tienen acceso a las mismas API del entorno de trabajo de las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de los componentes y este mismo mecanismo hace posible que los usuarios componentes serán reemplazados por los usuarios.
- Bibliotecas: El sistema operativo incluye un conjunto de librerías C/C++ utilizadas por varios componentes del sistema.
- Runtime de Android: Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik, escrita de tal forma que permita correr múltiples máquinas virtuales simultáneamente en el dispositivo. Desde la versión 5.0 de Android se utiliza ART, que compila completamente al momento de instalar las aplicaciones.
- Núcleo Linux: Funciona como capa de abstracción entre el hardware del sistema y el resto de la pila de software. El sistema operativo depende de Linux para la utilización de servicios base como el sistema de seguridad, gestión de la memoria, gestión de los procesos, pila de red.

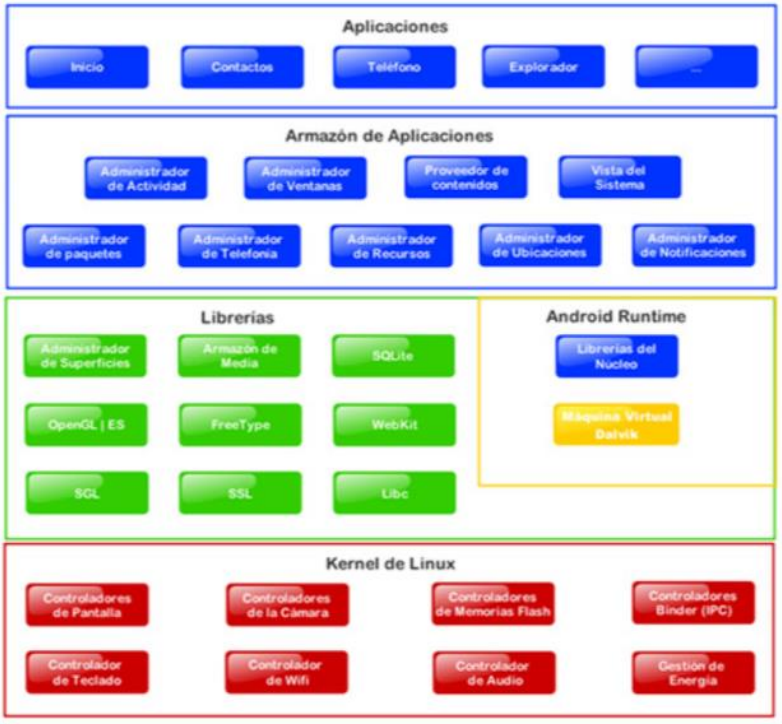

Ilustración 3.19.- Esquema de la arquitectura del sistema operativo Android

3.6.3 Versiones del sistema operativo

Las mejoras en el sistema operativo son constantes y la evolución desde su primera versión es notable. Cada una de las actualizaciones trae consigo mejoras de rendimientos y nuevas funcionalidades.

Un aspecto muy característico de las versiones de Android son sus nombres, que se encuentran numeradas con el añadido del nombre de un postre popular en ingles ordenados en orden alfabético desde la primera versión. A continuación, una lista con las versiones del sistema ordenadas por las más recientes (verTabla 1).

Tabla 1.- Versiones del sistema operativo Android

Nombre	Versión	Traducción
Apple Pie	1.0	Tarta de manzana
Banana Bread	1.1	Pan de plátano
Cupcake	1.5	Magdalena
Donut	1.6	Donut
Éclair	2.0 / 2.1	Palo de crema
Froyo	2.2	Yogur helado
Gingerbread	2.3	Pan de jengibre
Honeycomb	3.0 / 3.1 / 3.2	Panal
Ice Cream Sandwich	4.0	Sándwich de helado
Jelly Bean	4.1 / 4.2 / 4.3	Gominola
KitKat	4.4	Kit Kat
Lollipop	5.0 / 5.1	Piruleta
Marshmallow	6.0 / 6.0.1	Malvavisco
Nougat	7.0 / 7.1 / 7.1.1 / 7.1.2	Turrón
Oreo	8.0 / 8.1	Oreo
Pie	9.0	Pastel

3.6.4 Aplicaciones

Las aplicaciones en Android están programadas y desarrolladas mayoritariamente en lenguaje de programación Java a través del IDE oficial Android Studio lanzado en el año 2013, en sus últimas versiones Android Studio también soporta Kotlin como lenguaje nativo para la programación de apps. Otras herramientas disponibles para la programación de aplicaciones son por medio de Google App inventor, con aplicaciones o extensiones C/C++ o por medio de entorno de desarrollo multiplataforma como Xamarin entre otras opciones.

Para empezar a programar aplicaciones en Android no es necesario el conocimiento profundo de Java, con algunas bases y fundamentos de programación es posible desarrollar algunas apps sencillas. Las aplicaciones se empaquetan y comprimen en formato APK el cual se instala en los dispositivos móviles.

3.6.5 Entorno de desarrollo Android Studio

Android Studio [11] es el entorno de desarrollo (IDE de sus siglas en inglés) oficial para el desarrollo de aplicaciones Android. Anunciado en mayo del año 2013 en la conferencia anual para desarrolladores de Google como reemplazo para, el hasta entonces, entorno de desarrollo Eclipse Android Depelopment Tools. Esta construido sobre JetBrains'IntelliJ IDEA y cuenta con versiones para los sistemas operativos de Windows, macOS y Linux. La versión más reciente de Android Studio es la numero 3.2 lanzada en septiembre de 2018. Algunas de sus funciones más destacadas son:

- Un sistema de compilación basado en Gradle flexible
- Un emulador rápido con varias funciones
- Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android.
- Instant Run para aplicar cambios mientras tu app se ejecuta sin la necesidad de compilar un nuevo APK.
- Integración de plantillas de código y GitHub para ayudarte a compilar funciones comunes de las apps e importar ejemplos de código.
- Gran cantidad de herramientas y frameworks de prueba.
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versión, etc.
- Compatibilidad con C++ y NDK.
- Soporte incorporado para Google Cloud Platform, lo que facilita la integración de Google Cloud Messaging y App Engine.

3.6.5.1 Estructura de los proyectos

Los proyectos (o aplicaciones) en Android Studio están compuestos por uno o más módulos con archivos de código fuente y archivos de recursos dentro. Entre estos módulos se encuentra los siguientes:

- Módulos de aplicaciones para Android.
- Módulos de bibliotecas.
- Módulos de Google App Engine.

Los archivos de compilación para los proyectos son visibles en el panel izquierdo, en el nivel superior de los comandos Gradle donde cada módulo se compone de las siguientes carpetas:

- Manifest: contiene el archivo AndroidManifest.xml, aquí es donde se configuran y se declaran las pantallas y permisos que tiene la aplicación en el dispositivo.
- Java: Contiene todos los archivos y código fuente de la aplicación en Java.
- Res: En esta carpeta se encuentran alojados todos los recursos de diseño, variables, colores, letras, textos, diseños XLM que requiera la aplicación.

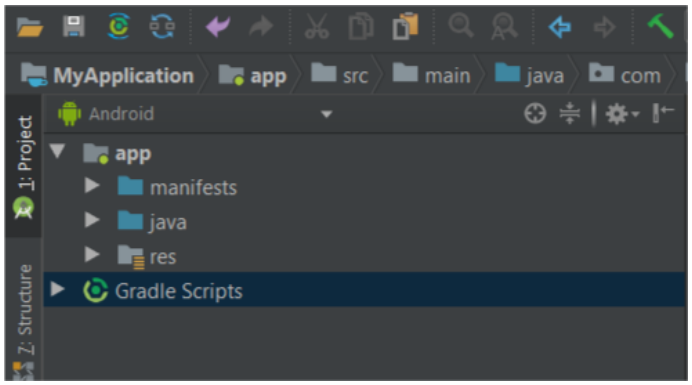

Ilustración 3.20.- Archivos de compilación de un proyecto Android Studio

3.6.5.2 Interfaz de usuario

La interfaz de usuario se compone de los siguientes elementos:

- Barra de herramientas: Permite realizar acciones como la ejecución de la app y el inicio de herramientas Android.
- Barra de navegación: Permite explorar el proyecto y abrir archivos para su edición. Genera una vista compacta del proyecto.
- Ventana de editor: Área donde se crea y modifica el código.
- Barra de la ventana de Herramientas: Contiene los botones que permiten expandir o contraer la ventana de herramientas individuales.
- Ventana de herramientas: Permite acceder a tareas específicas como la administración de proyectos, búsqueda y consulta de versiones entre otras opciones.
- Bara de estado: Muestra el estado del proyecto y del IDE, también lanza mensajes de errores o advertencias.

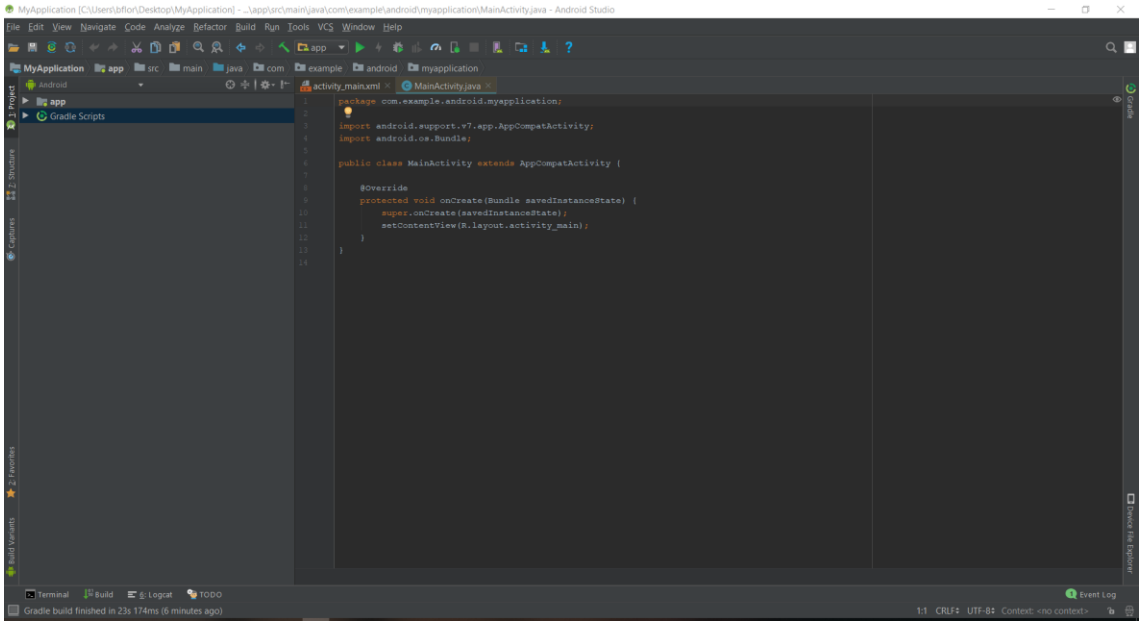

Ilustración 3.21.- Entorno de desarrollo Android Studio

3.7 Bases de datos

Las bases de datos se definen como un conjunto de información o datos organizados y relacionados entre sí que se encuentra agrupados, que pueden ser recolectados o exportados. Las bases de datos tienen como principales características las siguientes:

- Independencia lógica y física de los datos.
- Redundancia mínima de información.
- Permite el acceso concurrente a la información por diferentes puntos.
- Asegura la integridad de los datos.
- Optimización de las consultas.
- Seguridad de acceso y auditoria.
- Acceso a través de lenguajes de programación estandarizados.

3.7.1 Sistemas de gestión de bases de datos

Los sistemas de gestión de base de datos (SGBD) es un tipo de software específico que sirve como interfaz entre bases de datos, los usuarios y las aplicaciones que utilizan las bases de datos. Se compone principalmente de un lenguaje para la definición de datos, otro para la manipulación de datos y un lenguaje de consulta.

3.7.2 Tipos de bases de datos

Existen diferentes tipos de sistemas gestores de bases de datos, entre las cuales se encuentran:

- MySQL: Base de datos con licencia GPL basada en un servidor. Es rápida y no se recomienda su uso para la gestión de muchos datos.
- PostgreSql y Oracle: Sistema de base de datos potente, robusto y utilizado para grandes volúmenes de datos.
- Microsoft SQL Server: Base de datos desarrollada por Microsoft.

3.7.3 Modelo entidad-relación

El modelo entidad-relación es una herramienta para el modelado de datos de un sistema de información a través de diagramas. Los modelos son expresados en entidades relevantes para un sistema de información y sus interrelaciones.

3.7.3.1 Elementos del Modelo entidad relación

- Entidades: Representa cosas y objetos que se diferencian claramente entre sí y se representan en el diagrama con un rectángulo.
- Atributos: Se definen atributos a las características de las entidades. Cada entidad contiene un conjunto de atributos y se representan con círculos que descienden de las entidades.
- Relación: Es el vínculo que permite definir una dependencia entre varias entidades y se representan en el diagrama con rombos.

3.7.3.2 Tipos de relaciones

- Uno a uno: Una entidad se relaciona únicamente con otra y viceversa.
- Uno a varios o Varios a uno: Determina que un registro de una entidad puede estar relacionad con varios de otra entidad, pero esta entidad solo una vez.
- Varios a varios: Determina que una entidad está relacionada con otra con varios o ningún registro y viceversa.

3.7.3.3 Claves

Atributo de una entidad, que se le aplica una restricción que lo distingue de los demás registros y se le aplica un vínculo de tipo:

- Superclave: aplica a una clave o restricción a varios atributos de la entidad.
- Clave primaria: identificador único de un solo atributo que no se repite en una entidad.
- Clave externa: campo estrictamente relacionado con la clave primaria de otra entidad.

3.7.4 phpMyAdmin

Se trata de una herramienta de software libre escrita en PHP diseñada para la administración de bases de datos MySQL a través de un navegador web de internet. Ofrece las funciones de crear y eliminar bases de datos, crear y administrar tablas en una base de datos, ejecutar secuencias SQL, exportar datos entre otras funciones de administración de bases de datos. El software se encuentra disponible bajo licencia GPL V2.

3.7.4.1 Especificaciones y características

Algunas de las funciones y características destacables de phpMyAdmin son:

- Gestor de bases de datos grafico a través de la web
- Soporte para base de datos MySQL, MariaDB y Drizzle
- Se pueden importar datos en formato CSV y SQL
- Se puede exportar datos en formatos como SCV, SQL, XLM y PDF.
- Administración de servidores múltiple.
- Creación de gráfico de base de datos
- Búsqueda global en las bases de datos
- Interfaz gráfica para la creación de base de datos
- Soporte para sentencias SQL para la creación de tablas y bases de datos

3.8 Servicios Web

Los servicios web se definen como vías de comunicación e interoperabilidad entre maquinas conectadas a la red, siendo muy populares en el mundo de internet. Un servicio web es un componente al que se puede acceder mediante protocolos Web estándar que usa

XLM o JSON para el intercambio de información. Se pueden utilizar para integrar aplicaciones escritas en diferentes lenguajes y que se ejecutan en plataformas diferentes. Los servicios Web son independientes de lenguaje y de la plataforma gracias a que los servidores han admitido estándares comunes de Servicios Web. El funcionamiento de los servicios web se basa en el envío y recepción de información entre un cliente y un servidor para intercambiar datos, de forma que, un cliente envía una solicitud a un servidor, que la procesa y genera una respuesta para enviar de vuelta al cliente. Los servicios web se entienden como el tráfico de mensajes y datos entre dos máquinas.

3.8.1 Características

Las características deseables de un Servicio Web son:

- Debe poder ser accesible a través de la Web. Para ello debe utilizar protocolos de transporte estándares como HTTP y codificar los mensajes en un lenguaje estándar que pueda conocer cualquier cliente que quiera utilizar el servicio.
- Debe contener una descripción de sí mismo. De esta forma, una aplicación podrá saber cuál es la función de un determinado Servicio Web y poder utilizarlo de forma automática sin la intervención del usuario.
- Debe poder ser localizado. Mediante algún mecanismo un servicio Web debe poder encontrarse y que realice una determinada función.

3.8.2 Tipos de servicios Web

Los servicios pueden implementarse de varias formas. Por este motivo, podemos distinguir dos tipos de servicios Web: los denominados servicios Web "grandes" (*"big" Web Services*), los llamaremos servicios Web SOAP, y servicios Web RESTful.

3.8.2.1 Servicios Web SOAP

Los servicios Web SOAP, o servicios Web *"big"*, utilizan mensajes XML para intercomunicarse siguiendo el estándar SOAP (*Simple Object Access Protocol*), que es un lenguaje XML que define la arquitectura y formato de los mensajes. Sistemas que generalmente tienen una descripción legible por la máquina de la descripción de las operaciones ofrecidas por el servicio, escrita en WSDL (*Web Services Description Language*), que es un lenguaje basado en XML para definir las interfaces sintácticamente.

El formato de mensaje SOAP y el lenguaje de definición de interfaces WSDL está muy extendido y, por ejemplo, herramientas desarrollo como Netbeans, pueden reducir la complejidad de desarrollar aplicaciones de servicios web.

El diseño de un servicio web SOAP debe establecer una estructura formal para describir la interfaz que ofrece el servicio. WSDL puede utilizarse para describir los detalles del contrato, que pueden incluir mensajes, operaciones, *bindings*, y la localización del servicio Web.

3.8.2.2 Servicios Web RESTful

Los servicios Web RESTful (*Representational State Transfer Web Services*) son más útiles en escenarios de integración básicos *ad-hoc*. REST se integra mejor con HTTP que los servicios basado en SOAP, ya que no requieren mensajes XML o definiciones del servicio en forma de fichero WSDL

Los servicios web REST hace referencia a un conjunto de principios para el diseño de arquitecturas en la red, que resumen como se definen y diseccionan los recursos. REST no es un estándar, ya que solo es un estilo de arquitectura, pero si está basado en estándares como HTTP, URL, XML, HTML, MINE entre otros. Su estructura es más ligera y permite que los servicios se construyan utilizando herramientas de forma mínima.

Los objetivos del modelo de arquitectura REST para servicios web se enlistan de la siguiente forma:

- Escalabilidad de la integración con los componentes. Aún con el crecimiento de la web en los últimos años, el rendimiento de REST no se ha visto afectado, muestra de esto es la variedad de clientes con acceso a la web como estaciones de trabajo, dispositivos móviles, sistemas industriales, etc.
- Generalidad de interfaces. Por medio del protocolo HTTP un cliente pueda interactuar con cualquier servidor HTTP sin la necesidad de una configuración especial. Lo diferencia de otras alternativas como SOAP.
- Puesta en funcionamiento diferente. Los clientes y servidores son elementos con periodos de funcionamiento muy largos, por este motivo, REST asegura la retrocompatibilidad entre sistemas nuevos y antiguos. HTTP hace posible la extensibilidad mediante el uso de cabeceras a través de URIs.
- Compatibilidad con componentes intermedios, como proxys, Gateway, firewall. La compatibilidad con intermedios ayuda a reducir la latencia de interacción, refuerza la seguridad y encapsulamiento de otros sistemas.

La implementación de servicios web REST sigue cuatro principios de diseño fundamental que son:

- Utiliza métodos HTTP de forma explícita. Es una de las características claves de REST, permite que los desarrolladores utilicen los métodos HTTP de tal manera que resulte consisten con la definición del protocolo. Este diseño establece una asociación uno-a-uno entre operación crear, leer, borrar, actualizar y los métodos HTTP de la siguiente manera:
  - POST se utiliza para crear recursos en el servido.
  - GET se utiliza para obtener recurso en el servidor.
  - PUT se utiliza para cambiar el estado de un recurso o actualizarlo.
  - DELETE se utiliza para eliminar un recurso.
- No mantienen el estado. Los clientes de servicios web REST envían peticiones completas e independientes, se deben enviar peticiones que incluyan todos los datos necesarios para cumplir con el pedido, de forma que los servidores sean independientes, proceses y gestionen la carga sin necesidad de mantener el estado localmente. Esto hace que un servidor no tenga que recuperar ninguna información de contexto o estado para procesar una petición.
- Expone URIs con forma de directorios. Las URIs de los servicios web REST deben ser intuitivas, que sean adivinables fácilmente. Para hacerlo posible, las URIs tienen una estructura similar a los directorios de forma jerárquica, con una única ruta raíz que se va abriendo a las diferentes ramas y sub-rutas para exponer las áreas principales del servicio. Un ejemplo de URI REST es el siguiente:

  http://www.miservicio.org/discusion/temas/{tema}, donde *discusion* es la raíz, con un nodo *tema*, que tiene un subconjunto de nombres de temas.

- Trasfiere XML, JavaScript Object Notation (JSON) o ambos. Formatos estandarizados y muy extendidos para el intercambio de información, se basan en cadenas de texto para representar los datos de manera estructurada, datos que se intercambian entre las aplicaciones y los servicios en las peticiones/respuestas. Se caracterizan por ser fácilmente

interpretables por el ser humano. Un par de ejemplo de cadenas XML y JSON.

XML

```
<pieza tipo="A">
        <nombre>Tornillo</nombre>
       <descripcion>Cilindro mecanico con un cabeza utilizado en
       la fijación temporal de unas piezas con otras
       </descripcion>
        <caracateristica>
                    <tipo>metal</tipo>
              <tamanyo>10</tamanyo>
       </caracateristica>
        <vacio></vacio>
</pieza>
```

JSON

```
{
“pieza”: {
“tipo”: “A”
       “nombre”: “Tornillo”,
        “descripcion”:  “Cilindro  mecánico  con  un  cabeza
       utilizado en la fijación temporal de unas piezas con
       otras”,
       “caracteristica”: {
              “tipo”: “metal”
              “tamanyo”: 10
       },
       “vacio”: “”
 }
}
```

Capítulo 4

Descripción del hardware y software desarrollado

4.1 Introducción

En capítulos anteriores se describieron las tecnologías disponibles en el mercado para poder realizar el proyecto, y posteriormente se habló de la arquitectura hardware y software necesarias para su desarrollo. Por consiguiente, en este cuarto capítulo se explica el hardware y software desarrollado para la realización del proyecto.

En primer lugar, se describe la estructura hardware configurada para el proyecto para, posteriormente, detallar el software y la programación necesaria para cada uno de los dispositivos, así como la base de datos y los servicios web necesarios.

4.2 Descripción de la arquitectura

Como solución a los objetivos establecidos para este proyecto, el sistema estará compuesto por los siguientes subsistemas:

- Placa Arduino Pro Mini + Shield e-Health. Subsistema que se encarga de recolectar la información de los sensores, establecer una conexión UART con el microcontrolador ESP32 y enviar la información.
- Microcontrolador ESP32. Subsistema que crea los perfiles BLE necesarios para comunicarse con el dispositivo Android y gestiona la información de los sensores.
- Dispositivo móvil basado en Android OS. Subsistema compuesto por la aplicación Android para el caso de estudio del proyecto, realización y monitorizado de pacientes en terapias de exposición. La aplicación cuenta con las funciones de realizar y consultar de las terapias por parte del paciente, acceder y consultar los pacientes por parte del terapeuta.
- Base de datos. Subsistema que contiene la base de datos remota con la información de los usuarios del sistema, las terapias y sus contenidos.
- Servicios web. Subsistema con los servicios web necesarios para el intercambio de datos entre la base de datos y el dispositivo móvil inteligente Android.

A lo largo de este capítulo se detallará más a fondo cada uno de los subsistemas mencionados para una mejor comprensión de la arquitectura planteada.

4.3 Arduino Pro Mini + Shield e-Health

Este subsistema es el encargado de recibir la información de los sensores y enviarlos por medio de una conexión UART a la placa ESP32. Entre los diferentes sensores biomédicos que se pueden conectar al *Shield e-Health*, se han elegido los sensores de pulso cardiaco y resistencia galvánica en la piel como parámetros a controlar en las terapias de exposición debido a la relevancia e importancia que tienen sus variaciones en el estado de un paciente.

4.3.1 Hardware

Cabe mencionar también, que el *Shield e-health* está diseñado para la placa Arduino UNO, microcontrolador que se ha sustituido por un Arduino Pro Mini, las razones para efectuar el cambio están basadas en:

- Obtener un sistema de bajo consumo y en este apartado el Arduino Pro Mini tiene un consumo energético inferior que el UNO.
- Un sistema con pocas dimensiones.

Para efectuar la sustitución de la placa y que la compatibilidad entre el *Shield* y el Arduino Pro Mini sea plena, se realizó la conexión entre las dos por medio de cables. Para que los pines entre el Arduino UNO y el Pro Mini sean homónimos se ha consultado la librería *eHealth* para el IDE de Arduino en el archivo "eHealth.cpp", referenciando los pines que utiliza las funciones correctamente.

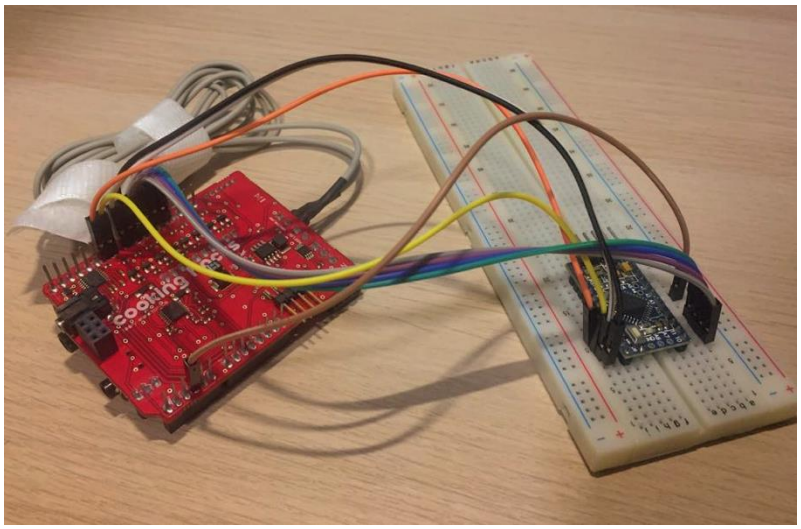

Ilustración 4.1.- Arduino Pro Mini + e-Health Platform

4.3.2 Software

El software o sketch se ha programado en el IDE de Arduino y se han empleado las librerías *eHealth* y *PinChangeInt* proporcionadas por el fabricante Cooking Hacks.

El funcionamiento del programa que ejecuta el Arduino es el siguiente:

- En primer lugar, se declaran una serie de valores que son necesario para almacenar los datos de pulso cardiaco (HRM) y resistencia galvánica en la piel (GSR).

- Por otra parte, en el "setup" del programa se habilita el "Serial" y se inicia una serie de funciones necesarias para las medidas HRM tomadas del ejemplo para medir estos valores.

```
void setup() {

  Serial.begin(115200);
  eHealth.initPulsioximeter();
  //Attach the inttruptions for using the pulsioximeter.
  PCintPort::attachInterrupt(6, readPulsioximeter, RISING);
}
```

Ilustración 4.2.- Función Setup Arduino

- En la función "loop", el sistema refresca los valores de HRM y GSR, posteriormente verifica si la conexión serial está disponible y lee la trama que recibe que llega del ESP32. Si la trama es correcta, se convierten los datos en bytes y se envían por el puerto serial al ESP32.

```
void loop() {

  RefrescarValores();

  if(Serial.available()>0){
    estado = Serial.read();
    if(estado == 'E'){
      dtostrf(ResistanceVol, 7, 3, outstr);
      Serial.write(outstr);
      delay(5);
      Serial.write(highByte(BPM));
      Serial.write(lowByte(BPM));
    }
  }
  // wait for 3 second
  delay(3000);
}
```

Ilustración 4.3.- Función Loop Arduino

4.4 Microcontrolador ESP32

El microcontrolador está conectado al Arduino Mini Pro por dos hilos (TX y RX), medio para intercambiar la información.

4.4.1 Entorno de desarrollo

El fabricante del ESP32 da soporte para el IDE de Arduino, entorno de desarrollo sencillo pero limitado en algunas funciones o herramientas. Como una alternativa más potente se encuentra Visual Studio Community 2017, un software de licencia libre propiedad de Microsoft, que cuenta con funciones como autocompletado, programación estructurada, entre muchas otras que son de ayuda a la hora de programar. Visual Studio tiene soporte para extensiones de terceros, entre ellas se encuentra disponible Visual Micro, que permite importar el IDE de Arduino en Visual Studio para la programación de los microcontroladores soportados con la ayuda de las herramientas de VS.

Frente a las ventajas y opciones extra que ofrece Visual Studio frente al IDE de Arduino, la programación del ESP32 se realizó gracias a la extensión de Visual Micro en Visual Studio.

4.4.2 Software desarrollado

El código que ejecuta el microcontrolador tiene las siguientes características:

- Comunicación con el Arduino Pro Mini que se realiza por medio de una UART extra que se ejecuta por medio de la librería HardwareSerial.
- Gestión de los datos que recibe por parte del Arduino.
- Creación y configuración del servidor BLE con servicios y características necesarias.
- Envío de información con el dispositivo Android por medio de BLE.

4.4.2.1 Comunicación UART y Arduino

Esta comunicación se hace a través de una UART extra creada a través de la librería "HardwareSerial.h", la cual es declarada en la cabecera del *sketch*, ya que el puerto serial por defecto ha sido usado para imprimir en el terminal información de interés en la programación para facilitar la depuración de los módulos software desarrollados.

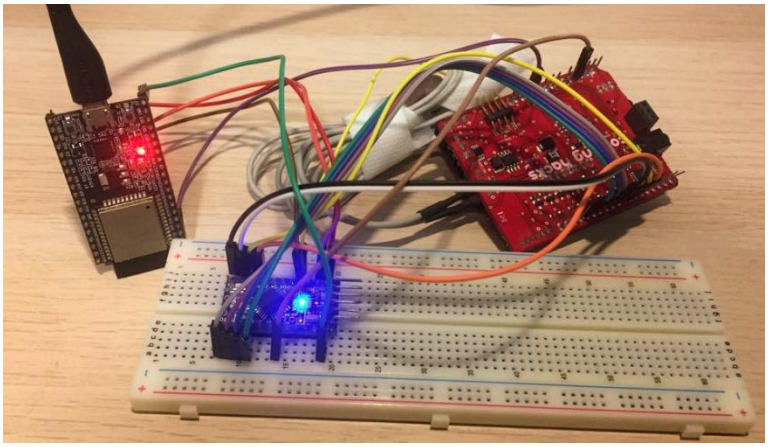

Ilustración 4.4.- ESP32 conectado al Arduino

La UART llamada "MySerial" es declarada en la funcion "setup" y se utiliza en la función "loop" para enviar tramas al Arduino.

```
void setup() {

    // ...
    InitBLE();
    Serial.begin(115200);
    MySerial.begin(115200, SERIAL_8N1, 16, 17);
    Serial.println("Start");

}
```

Ilustración 4.5.-Función Setup del ESP32

4.4.2.2 Configuración BLE

Como se especificó en el capítulo dos, la tecnología Bluetooth Low Energy define una serie de perfiles con sus de funciones y características tanto en la capa de control como en la capa de datos. En el apartado de *e-health* concretamente, se encuentran disponibles perfiles específicos para su implementación como son los perfiles de glucosa, temperatura, entre muchos otros que son recomendables de usar.

Para el desarrollo del proyecto se utiliza algunos de estos perfiles específicos estándar, concretamente el perfil definido Heart Rate Measurement (HRM) para enviar los datos del pulso cardiaco entre dispositivos. Para la Resistencia Galvánica en la Piel (GSR) no hay un perfil establecido y estándar. Por este motivo, el envió de información se realiza por medio un perfil genérico para dispositivos *e-health* que define Bluetooth SIG y se encuentra en su web.

Los perfiles específicos están basados en GATT, lo que quiere decir que utilizan los procedimientos y modelos operativos del perfil GATT como base. Cada uno de los perfiles define la solución al describir el comportamiento de ambos extremos del enlace, como

servidor GATT y como cliente GATT. En el caso del servidor GATT, define todos los servicios que contiene.

Los servicios definen características que determinan su funcionamiento. Por tanto, un perfil define una serie de servicios y su utilización, a su vez estos cuentan una serie de características. En la imagen se describe la jerarquía genérica de los perfiles BLE basados en GATT.

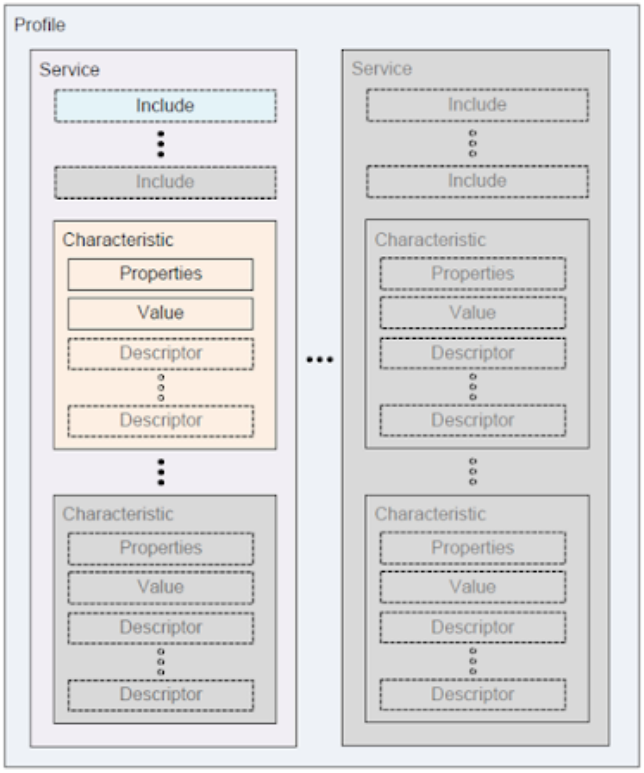

Ilustración 4.6.- Jerarquía de perfiles basados en GATT

Para la solución del proyecto, el perfil creado en el ESP32 funciona como servidor, cuenta con dos servicios y a su vez estos con 2 características. La siguiente tabla resume el esquema (ver Tabla 2).

Tabla 2.- Estructura del servidor BLE para el ESP32

BLE Server	
Servicio: Heart Rate	**Servicio:** Galvanic Skin Resistance
Caracteristica : Heart Rate Measurement	**Caracteristica:** Galvanic Skin Resistance Measurement
Caracteristica: HR Sensor Position	**Caracteristica:** HR Sensor Position

Donde cada uno de los servicios y características tiene un identificador único UUID de 128 bits, que son los tomados para dispositivos *ehealth* de la página web de Bluetooth SIG.

La creación del servidor BLE, sus servicios y características está basado en el ejemplo "BLE_server_Arduino.ino" y las librerías disponibles al instalar el Core del ESP32 en Arduino. Concretamente, se han utilizado las siguientes librerías (ver Ilustración 4.7).

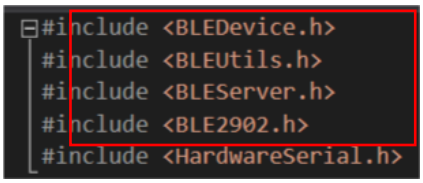

Ilustración 4.7.- Librerias BLE para ESP32

Posteriormente, es necesario declarar los valores constantes de las servicios y características con sus propiedades por medio de las funciones que ofrecen las librerías BLE.

```
bool _BLEClientConnected = false;

#define heartRateService BLEUUID((uint16_t)0x180D)
#define galvSkinResponServ BLEUUID((uint16_t)0x107F) //Usage Sensor UUID BLE Serv

//Definicion de caracteristicas perfil HR

BLECharacteristic heartRateMeasurementCharacteristics(BLEUUID((uint16_t)0x2A37), BLECharacteristic::PROPERTY_NOTIFY);
BLECharacteristic sensorPositionCharacteristic(BLEUUID((uint16_t)0x2A38), BLECharacteristic::PROPERTY_READ);
BLEDescriptor heartRateDescriptor(BLEUUID((uint16_t)0x2901));
BLEDescriptor sensorPositionDescriptor(BLEUUID((uint16_t)0x2901));

//Definicion de caracterisiticas perfil GSR
BLECharacteristic galvSkinResponCharacteristics(BLEUUID((uint16_t)0x2A98), BLECharacteristic::PROPERTY_NOTIFY);
BLECharacteristic GSRsensorPositionCharacteristic(BLEUUID((uint16_t)0x2A99), BLECharacteristic::PROPERTY_READ);
BLEDescriptor galvSkinResponDescriptor(BLEUUID((uint16_t)0x2901));
BLEDescriptor GSRsensorPositionDescriptor(BLEUUID((uint16_t)0x2901));
```

Ilustración 4.8.- Definición características de los servicios BLE

También es necesario una función que se encarga de actualizar estado del servidor.

```
class MyServerCallbacks : public BLEServerCallbacks {
    void onConnect(BLEServer* pServer) {
      _BLEClientConnected = true;
    };
    void onDisconnect(BLEServer* pServer) {
      _BLEClientConnected = false;
    }
};
```

Ilustración 4.9.- Función estado del servidor ESP32

Por último, para finalizar la implementación del BLE en el ESP32 debemos ejecutar en el "Setup" del sketch la función "InitBLE", que se encarga de crear y habilitar todos los

servicios y características para que estén disponibles. La estructura de la función es la siguiente (ver Ilustración 4.10).

```
void InitBLE() {

  BLEDevice::init("e-Healt Device");

  // Create the BLE Server
  BLEServer *pServerHR = BLEDevice::createServer();
  BLEServer *pServerGSR = BLEDevice::createServer();
  pServerHR->setCallbacks(new MyServerCallbacks());
  pServerGSR->setCallbacks(new MyServerCallbacks());


  // Create the BLE Service HR
  BLEService *pHeart = pServerHR->createService(heartRateService);

  pHeart->addCharacteristic(&heartRateMeasurementCharacteristics);
  heartRateMeasurementCharacteristics.addDescriptor(&heartRateDescriptor);
  heartRateMeasurementCharacteristics.addDescriptor(new BLE2902());

  pHeart->addCharacteristic(&sensorPositionCharacteristic);
  sensorPositionCharacteristic.addDescriptor(&sensorPositionDescriptor);

  pServerHR->getAdvertising()->addServiceUUID(heartRateService);
  pHeart->start();

  // Create the BLE Service GSR
  BLEService *pSkin = pServerGSR->createService(galvSkinResponServ);

  pSkin->addCharacteristic(&galvSkinResponCharacteristics);
  galvSkinResponCharacteristics.addDescriptor(&galvSkinResponDescriptor);
  galvSkinResponCharacteristics.addDescriptor(new BLE2902());

  pSkin->addCharacteristic(&GSRsensorPositionCharacteristic);
  GSRsensorPositionCharacteristic.addDescriptor(&GSRsensorPositionDescriptor);

  pServerGSR->getAdvertising()->addServiceUUID(galvSkinResponServ);
  pSkin->start();

  // Start advertising
  pServerHR->getAdvertising()->start();
  pServerGSR->getAdvertising()->start();

}
```

Ilustración 4.10.- Configuración de los servicios BLE ESP32

4.4.2.3 Envió de información por BLE

El envío de información por medio de BLE se hace a través de una función llamada "enviarDatosBLE", que se ejecuta periódicamente en el "loop" del sketch. La estructura que tiene es la siguiente (ver Ilustración 4.11).

```
void enviarDatosBLE() {

    //Funcion que envia los datos por BLE
    heartRateMeasurementCharacteristics.setValue(heart, 4); //set
    heartRateMeasurementCharacteristics.notify();
    sensorPositionCharacteristic.setValue(hrmPos, 1);

    delay(8);

    galvSkinResponCharacteristics.setValue(gsr, 2);
    galvSkinResponCharacteristics.notify();
    GSRsensorPositionCharacteristic.setValue(gsrPos, 1);


}
```

Ilustración 4.11.- Función enviarDatosBLE

Como se observa, por medio de las funciones "setValue" y "notify", las características preparan y notifican al cliente, en este caso el dispositivo Android.

4.5 Dispositivo móvil basado en Android OS

Este subsistema se compone principalmente por una aplicación para dispositivos Android que permitirá realizar las funciones de:

- Acceder a la base de datos como paciente o terapeuta. Si se accede como paciente permite ver las terapias y actividades programadas, la realización de las actividades, así como la gestión de los datos de las actividades. Si se accede como terapeuta, permite ver los pacientes que tiene a su cargo, la información de sus terapias y actividades.
- Recepción de los datos que envía el ESP32, gestión y visualización en graficas.

4.5.1 Creación del proyecto (aplicación) en Android Studio

El primer paso fue crear un nuevo proyecto en Android Studio siguiendo los siguientes pasos:

- A través de la ruta Archivo/nuevo proyecto en Android Studio, abrimos el asistente para crear un nuevo proyecto. Se asigno como nombre del proyecto *AppFinalTFG*.
- La API seleccionada para el proyecto fue API 22: Android 5.1 (Lollipop), ya que el dispositivo de pruebas para el proyecto es una Tablet que corre esa versión del sistema.

- Se seleccionó una *Empy Activity* como actividad de inicio y los demás parámetros de configuración se dejaron por defecto y se ejecutó la creación del proyecto.

Una vez finalizado el proceso del asistente de creación de nuevos proyectos, Android Studio abre el fichero "MainActivity.java", espacio de trabajo para empezar a programar la aplicación.

4.5.2 Estructura de la aplicación Android

Como se vio en el capítulo anterior, los proyectos de Android Studio están compuestos por módulos muy bien diferenciados que alojan todo el código de la aplicación. A continuación, se detalla el contenido de cada módulo.

4.5.2.1 Manifest

En este módulo se encuentra el archivo "AndroidManifest.xlm", archivo muy importante donde se aloja la información de la aplicación, los colores y los temas que utiliza, todos los permisos que necesita la aplicación, las actividades y servicios para poder funcionar correctamente. La forma de programar este archivo es usando lenguaje XML.

Para este proyecto se requieren de los siguientes permisos:

- INTERNET: Es necesario el acceso a Internet para poder conectar el dispositivo a la base de datos remota que se encuentra alojada en un servidor web Apache local (esto se detalla con más detenimiento en el apartado de Servicios Web de este capítulo).
- BLUETOOTH: Este permiso permite a la aplicación poder enviar y recibir información a través de tecnologías Bluetooth.
- BLUETOOTH_ADMIN: Es un permiso más avanzado que el anterior y permite dotar a la aplicación de mayores privilegios como son administrar los dispositivos para conectarse, activar o desactivar el Bluetooth del dispositivo. Estas funciones que se requieren en el desarrollo de la aplicación.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

Ilustración 4.12.- Permisos del Manifest

Como se mencionó antes, otros elementos importantes que se deben declarar en este archivo son las actividades y servicios que necesite la aplicación. La forma de declararlos es mediante las etiquetas:

- <activity android:name = " *Nombre de la actividad*" />, para las actividades. La aplicación cuenta con varias pantallas y actividades.
- <service/ android:name = "*Nombre del serivicio*" />, para los servicios, concretamente se utiliza un servicio llamado *Service_BLE_GATT*, necesario para controlar los eventos Bluetooth.

4.5.2.2 Gradle

Este módulo se compone del Gradle, el sistema de compilación de aplicaciones que usa Android Studio. Un paquete de herramientas muy avanzado para automatizar y administrar el proceso de compilación, a la vez que permite configuraciones flexibles y personalizadas.

Para este proyecto, ha sido necesario modificar dos de los archivos que componen este compilador, que son el "Proyect Level build.gradle" y el "App level build.gradel". En el primero de estos archivos hay que añadir dentro de la función "allprojects" la siguiente línea, quedando de la forma mostrada en la Ilustración 4.13):

```
allprojects {
    repositories {
        maven { url 'https://jitpack.io' }
        google()
        jcenter()
    }
}
```

Ilustración 4.13.- Configuracion Gradle 1

En el segundo, el "App level: build.gradel", se añaden las dependencias en el apartado "dependencies", que son las siguientes:

- Volley: Es una librería HTTP que permite hacer comunicaciones para aplicaciones Android por medio de este estándar de forma fácil y rápida. En este proyecto se implementa Volley para el envío y recepción de datos con la base de datos a través de servicios web.
- Recyclerview: Librería para implementar las listas *Recyclerview*. En la aplicación se utilizan *Recyclerview* para visualizar en pantalla la lista de pacientes, terapias y actividades de los pacientes.

- MPAndroidChart: Librería que permite implementar graficas en aplicaciones Android. En la aplicación se utiliza esta librería para realizar las gráficas de los datos obtenidos de las actividades de los pacientes.

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:27.1.1'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    implementation 'com.android.volley:volley:1.1.1'
    implementation 'com.android.support:recyclerview-v7:27.1.1'
    implementation 'com.github.PhilJay:MPAndroidChart:v3.0.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
}
```

Ilustración 4.14.- Estructura del Gradle 2, implements

4.5.2.3 Java

Este módulo está compuesto por todos archivos .java que componen la aplicación que se encuentran distribuido de la siguiente forma:

- MainActiviy.java: Es la actividad principal y con la cual inicia la aplicación cuando se ejecuta en el dispositivo. Se encarga de enseñar la primera interfaz, consta de un botón para paciente y otro para terapeuta que lanzará otra actividad en función de la elección del usuario.

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    private Button btn_ter;
    private Button btn_pac;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btn_pac = findViewById(R.id.btn_paciente);
        btn_ter = findViewById(R.id.btn_terapeuta);

        btn_pac.setOnClickListener(this);
        btn_ter.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()){

            case R.id.btn_paciente:
                Intent intent = new Intent( packageContext: this, paciente_2_1.class);
                startActivity(intent);
                break;
            case R.id.btn_terapeuta:
                Intent intent1 = new Intent( packageContext: this, terapeuta_1_1.class);
                startActivity(intent1);
                break;
        }
    }
}
```

Ilustración 4.15.- MainActivity.Java, programación de activities en Android Studio

- PacienteActivities: Es un paquete compuesto por 6 archivos, clases Java que se enlazan con los *layuotActivities.xlm*. Es donde se programa la lógica y funcionalidad que tienen las *activities* de paciente en la aplicación. En la figura siguiente se muestra parte del código de una de las actividades, como conectarse a un base de datos y recibir información utilizando la dependencia Volley.

```
private void iniciarSesion() {

    String ip = "http://192.168.1.108";
    String url = ip+"/tfg/android/inicio_sesion.php?dni="+cajaDni.getText().toString()+
            "&pwd="+cajaPwd.getText().toString()+"&usuario=paciente";
    jsonObjectRequest = new JsonObjectRequest(Request.Method.GET, url, jsonRequest: null, listener: this, errorListener: this);
    volleySingleton.getInstanciaVolley(getApplicationContext()).addToRequestQueue(jsonObjectRequest);

}

@Override
public void onErrorResponse(VolleyError error) {

    Toast.makeText( context: this, text: "No se encontro el usuario. " + error.toString(), Toast.LENGTH_SHORT).show();
}

@Override
public void onResponse(JSONObject response) {

    Toast.makeText( context: this, text: "Se ha encontrado el usuario: " + cajaDni.getText().toString(),
            Toast.LENGTH_SHORT).show();

    user usuario = new user();
    JSONArray jsonArray = response.optJSONArray( name: "datos"); //Se llama datos para relacionr con el json 'datos' de "sesion.php"
    JSONObject jsonObject = null;

    try {
        jsonObject = jsonArray.getJSONObject( index: 0);
        usuario.setDni(jsonObject.optString( name: "dni"));
        usuario.setPwd(jsonObject.optString( name: "pwd"));
        usuario.setNames(jsonObject.optString( name: "nombre"));
    } catch (JSONException e) {
        e.printStackTrace();
    }

    Intent intent = new Intent( packageContext: this, paciente_2_2.class);
    intent.putExtra( name: "nombres", usuario.getNames());
    intent.putExtra( name: "dni", usuario.getDni());
    startActivity(intent);
}
```

Ilustración 4.16.- Código para una pantalla "Paciente", como conectar y sercibir información con una base de datos.

- TerapeutaActivities: Es un paquete compuesto por 4 archivos, clases Java que se enlazan con los *layuotActivities.xlm*. Es donde se programa la lógica y funcionalidad que tiene las *activities* de terapeuta en la aplicación.
- Utils: Contiene un archivo llamado Utils.java y su propósito es almacenar valores constantes y funciones que se repiten en algunos procesos de la aplicación con el fin de optimizar el código. Para este proyecto, el archivo se compone de:
    - Constates de los UUID, identificadores de los servicios Bluetooth.
    - Función llamada *checkBluetooth,* que verifica si el bluetooth está conectado y disponible en el dispositivo.
    - Función *requestUserBluetooth,* que verifica el estado de la conexión Bluetooth y lanza una activity.

- Función *Toast*, sirve para imprimir mensajes de corta duración en pantalla al usuario.

```
public static UUID HEART_RATE_MEASUREMENT = UUID.fromString("00002a37-0000-1000-8000-00805f9b34fb");
public static UUID GALVANIC_SKIN_RESPONSE_MEASUREMENT = UUID.fromString("00002a77-0000-1000-8000-00805f9b34fb");

public static final String EXTRA_NAME = "android.1.Activity_BTLE_Services.NAME";
public static final String EXTRA_ADDRESS = "android.Activity_BTLE_Services.ADDRESS";
public static final String EXTRA_DNI = "dniPaciente";
public static final String EXTRA_ACTIVIDAD = "actividadPaciente";
public static final String EXTRA_REPOSITORIO = "idRepositorio";

public static boolean checkBluetooth(BluetoothAdapter bluetoothAdapter) {
    // Ensures Bluetooth is available on the device and it is enabled. If not,
    // displays a dialog requesting user permission to enable Bluetooth.
    if (bluetoothAdapter == null || !bluetoothAdapter.isEnabled()) {
        return false;
    }
    else {
        return true;
    }
}
```

Ilustración 4.17.- Fragmento código Utils.java

- WebServicesAnswer: Paquete de clases Java que sirve para almacenar las respuestas que llegan de las consultas de los servicios web. Se basan en una estructura de declaración de variables y sus respectivos *Getters y Setters*.

```
public class datosPaciente {

    String dniPaciente, nombrePaciente;

    public String getDniPaciente() { return dniPaciente; }

    public void setDniPaciente(String dniPaciente) { this.dniPaciente = dniPaciente; }

    public String getNombrePaciente() { return nombrePaciente; }

    public void setNombrePaciente(String nombrePaciente) { this.nombrePaciente = nombrePaciente; }
}
```

Ilustración 4.18.- Ejemplo de clase Java en el paquete WebServicesAndwear

- Adaptadores: Paquete de clases Java que almacena los *adapters* necesarios para los *Recyclerviews* que implementa la aplicación para la visualización de listas de pacientes, terapias y actividades. Son clases que se heredan de *Recyclerview.Adapter* para, a través de sus métodos, gestionar y preparar la información que se visualiza en los *Recyclerview* o listas. Además, implementa la interfaz *OnClickListener*, que sirve para dotar a los ítems de la lista de funciones cuando se pulse sobre ellos. En concreto, en la aplicación las pulsaciones sobre ítems hacen que la aplicación se mueva hacia otra pantalla o *activity*.

- ConexionBLE: Paquete con clases Java para implementar el funcionamiento de BLE en la aplicación. Está compuesta de los siguientes archivos.
    - BTLE_Device: clase Java que almacena información de un dispositivo Bluetooth, en la figura (ver Ilustración 4.19) el código.
    - ListAdapter_BTLE_Devices: clase Java que hereda de ArrayAdapter<BLE_Device>, y su función es mostrar una lista con los dispositivos Bluetooth disponibles en la aplicación.
    - Scanner_BTLE: clase Java que sirve para escanear los dispositivos Bluetooth que se encuentran cerca y almacenar su información. en la figura (ver Ilustración 4.20) se observa un fragmento del código.
    - BroadcastReceiver_BTState: clase Java que se usa para actualizar el estado de conexión de Bluetooth.
    - Service_BTLE_GATT: clase Java que implementa los métodos para conectarse a un dispositivo BLE, recibir y gestionar la información que reciben.

```
public class BTLE_Device {

    private BluetoothDevice bluetoothDevice;
    private int rssi;

    public BTLE_Device(BluetoothDevice bluetoothDevice) { this.bluetoothDevice = bluetoothDevice; }

    public String getAddress() { return bluetoothDevice.getAddress(); }

    public String getName() { return bluetoothDevice.getName(); }

    public void setRSSI(int rssi) { this.rssi = rssi; }

    public int getRSSI() { return rssi; }
}
```

Ilustración 4.19.- Clase BTLE_Device

```
private void scanLeDevice(final boolean enable) {
    if (enable && !mScanning) {
        Utils.toast(ma.getApplicationContext(), string: "Starting BLE scan...");

        // Stops scanning after a pre-defined scan period.
        mHandler.postDelayed(new Runnable() {
            @Override
            public void run() {
                Utils.toast(ma.getApplicationContext(), string: "Stopping BLE scan...");

                mScanning = false;
                mBluetoothAdapter.stopLeScan(mLeScanCallback);

                ma.stopScan();
            }
        }, scanPeriod);

        mScanning = true;
        mBluetoothAdapter.startLeScan(mLeScanCallback);
          mBluetoothAdapter.startLeScan(uuids, mLeScanCallback);
    }
    else {
        mScanning = false;
        mBluetoothAdapter.stopLeScan(mLeScanCallback);
    }
}
```

Ilustración 4.20.- Función scanLeDevice, fragmento de la clase Java Scanner_BTLE

4.5.2.4 Res

En este módulo se encuentran los *layoutAcivities.xlm*, archivos que contienen la interfaz gráfica de la aplicación y están enlazados con los *Activities.Java,* descritos anteriormente. Para la aplicación se han utilizado 10 *layuout*, que son el total de pantallas navegables que componen la aplicación. En la Ilustración 4.21 se muestra la estructura de un archivo XML de este tipo.

Los *layouts* se construyen por una cabecera que define el tipo de contenedor gráfico. Para el proyecto, se utilizó siempre *LinearLayout*, y dentro de estos contenedores van los elementos visuales que ofrece Android Studio como son *TextViews*, *EditText*, *Buttons*, *RecyclerViews*. A todos los elementos se les pueden establecer un tamaño de letra, dimensiones, espaciado y demás opciones. Entre los parámetros configurables, los identificadores "id" son muy importante, ya que son necesarios para que puedan ser enlazados en las clases *Activities.Java.*

Finalmente, la Ilustración 4.22 muestra una de las pantallas finales de la App realizada, tal y como se observaría al ejecutar la aplicación en el dispositivo móvil inteligente.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:paddingTop="52dp"
        android:text="Bienvenido usuario, elija una opcion."
        android:textAlignment="center"
        android:textSize="21dp"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:gravity="center"
        android:paddingTop="21dp">

        <Button
            android:layout_width="150dp"
            android:layout_height="wrap_content"
            android:text="Terapeuta"
            android:id="@+id/btn_terapeuta"/>
        <Button
            android:layout_width="150dp"
            android:layout_height="wrap_content"
            android:text="Paciente"
            android:id="@+id/btn_paciente"/>

    </LinearLayout>

</LinearLayout>
```

Ilustración 4.21.- Código archivo XLM para una de las pantallas de la aplicación.

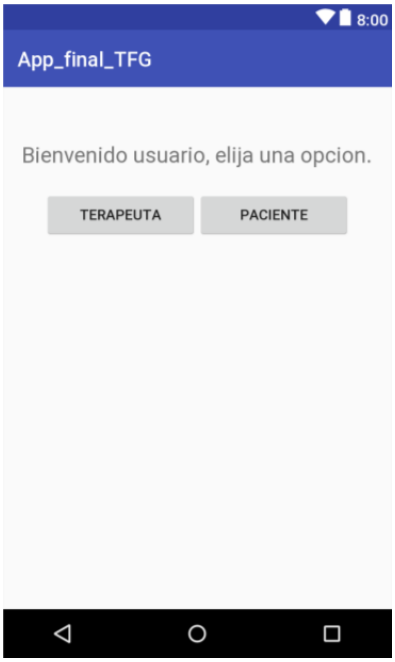

Ilustración 4.22.- Pantalla de Inicio App

4.6 Base de datos

Este subsistema está compuesto por la base de datos donde se aloja la información de los pacientes, terapeutas, terapias y actividades. La base de datos creada para este proyecto se basa en el modelo relacional descrito en el capítulo anterior. Las razones de su elección son porque es una herramienta muy conocida y usada en el ámbito del modelado de datos.

4.6.1 Estructura de la base de datos

Las entidades más relevantes son las siguientes:

- Paciente: Entidad con la información de cada paciente registrado en la base de datos como es su nombre, apellido, numero seguro. Tiene un identificador único, que es el DNI, que le permite relacionarse con otras entidades.
- Terapeuta: Entidad con la información de los terapeutas registrados con sus respectivos datos como son el nombre, apellido. Su identificador único es el DNI, el cual le permite relacionarse con otras tablas.
- Terapias: Entidad con la lista de terapias que se pueden realizar, cuenta con un identificador único y una descripción.
- Actividad: Entidad con la información de una actividad, tiene atributos de tiempo, descripción y un identificador.
- Repositorio multimedia: Entidad que almacena los enlace web multimedia.
- Lecturas: Entidad que almacena las lecturas de HR, GSK y el tiempo.

Las relaciones de la base de datos son las siguientes:

- Paciente-Terapeuta: Relación muchos a muchos.
- Paciente-Terapia: Relación muchos a muchos.
- Paciente-Lecturas: Relación uno a muchos.
- Terapeuta-Terapias: Relación muchos a muchos.
- Terapia-Actividad: Relación muchos a muchos.
- Actividad-Reposito Multimedia: Relación muchos a muchos.
- Actividad-Lecturas: Relación uno a muchos.

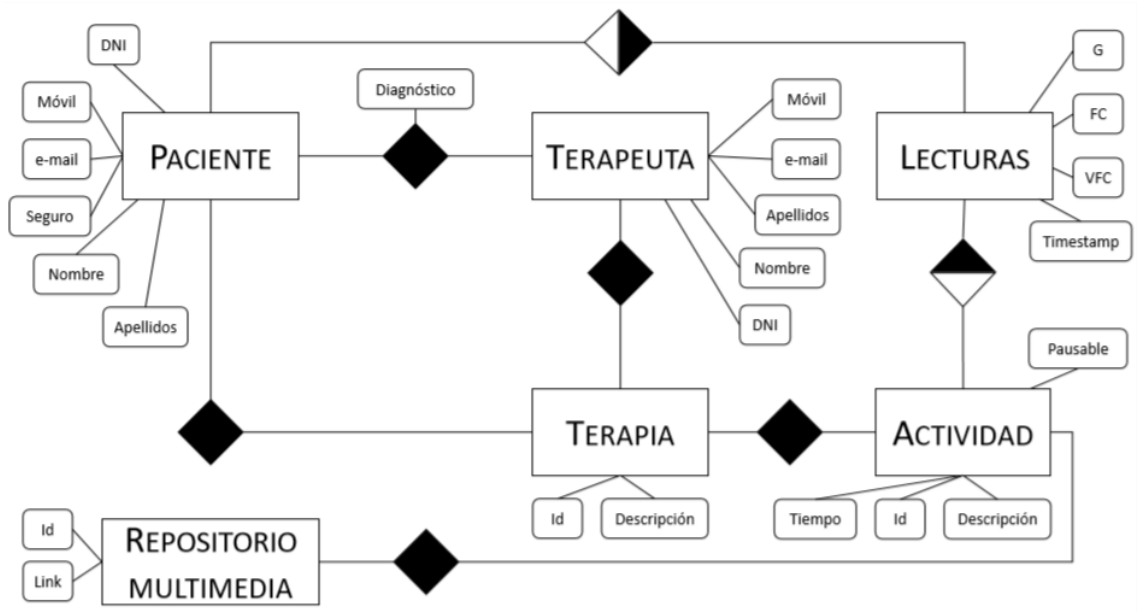

Ilustración 4.23.- Diagrama Entidad-Relación de la base de datos

4.6.2 Software XAMPP

XAMPP es un paquete de software libre, que está compuesto por un servidor web Apache, un sistema de gestión de base de datos MySQL y los intérpretes para lenguajes PHP y Perl. En la versión más reciente el gestor de base de datos es MariaDB, un folk MySQL con licencia GPL.

Para el desarrollo de este proyecto se utiliza XAMPP como gestor de base de datos y para crear un servidor web local. El software que hay que instalar previamente para el desarrollo del proyecto, que utiliza como máquina principal Windows, requiere los siguientes pasos para su funcionamiento:

- Descargar XAMPP desde la web www.apachefriends.org.
- Hacer doble clic sobre el archivo descargado y a continuación, se inicia el asistente. Para continuar se da clic en "siguiente".

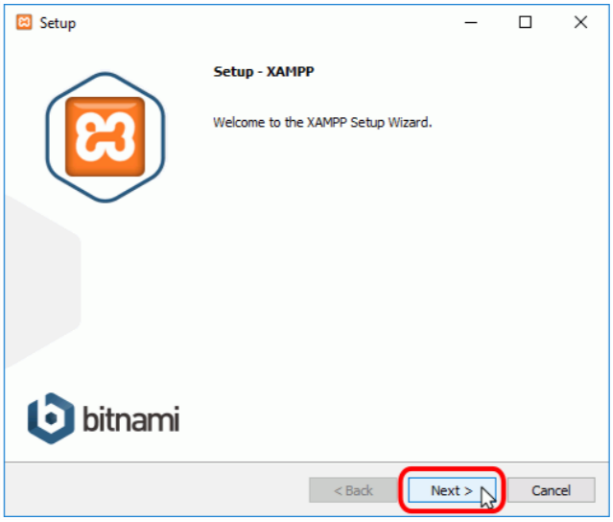

Ilustración 4.24.- Instalación XAMPP (1)

- Se seleccionan los componentes que queremos instalar, para nuestro caso se selecciona el servidor Apache y MySQL. En el apartado "lenguaje de programación" se selecciona PHP y phpMyAdmin, para finalmente hacer clic en "siguiente".

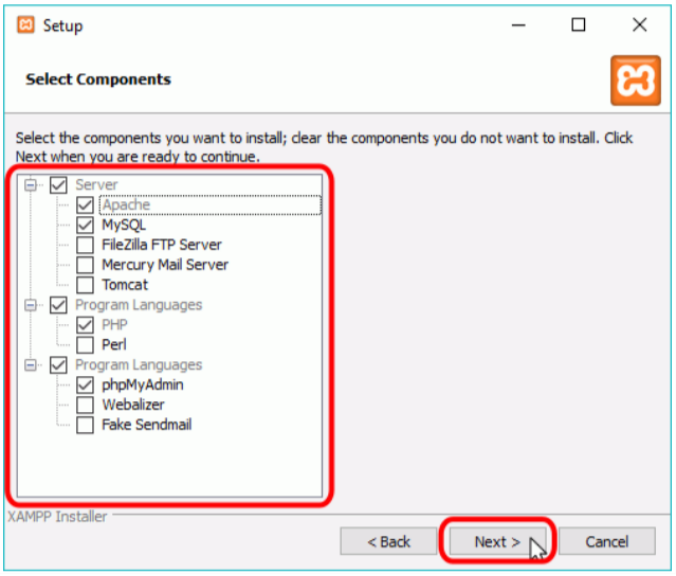

Ilustración 4.25.- Instalación XAMPP (2)

- Después el asistente nos muestra la carpeta de instalación, dejamos la carpeta por defecto y después pulsamos "siguiente".

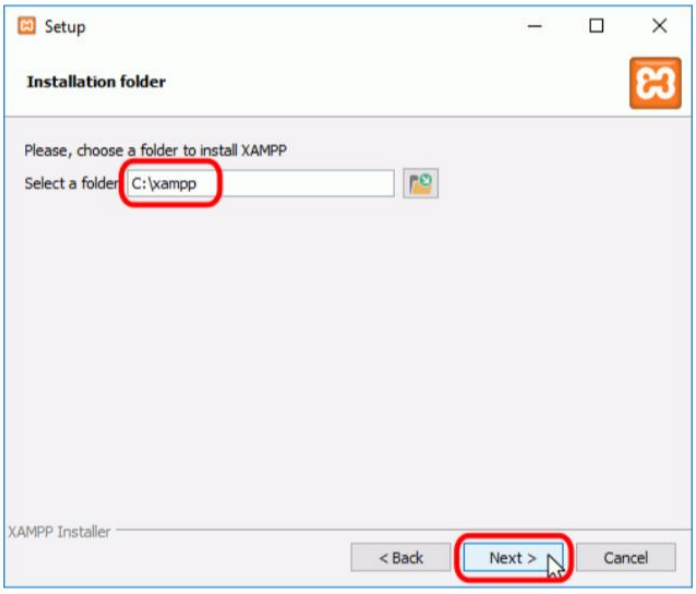

Ilustración 4.26.- Instalación XAMPP (3)

- Las siguientes opciones se da "siguiente" y el programa empieza a instalar, cuando salta un mensaje del cortafuegos, clicamos en "Permitir Acceso".

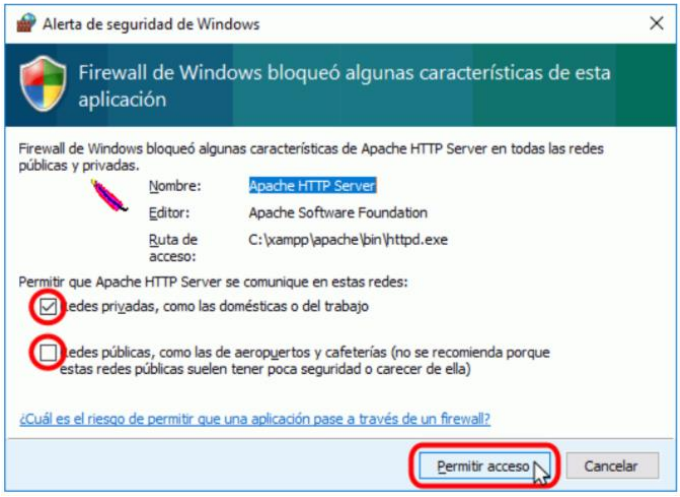

Ilustración 4.27.- Instalación XAMPP (4)

- Si todo está correcto, XAMPP se instalará correctamente.

Una vez instalado el software, debemos configurarlo para poder habilitar las herramientas de base de datos y el servidor Apache. Para que esto, se ejecuta el programa "XAMPP Control Panel" que tiene la siguiente interfaz.

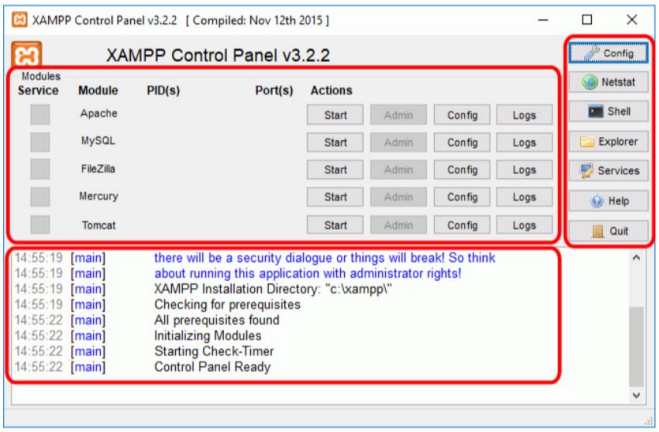

Ilustración 4.28.- Instalación XAMPP (5)

Para activar el servidor Apache hay que hacer clic en el botón "start" en el apartado de "Actions", proceso que es el mismo para el servicio de MySQL. Una vez hecho estos pasos ya estamos en disposición de trabajar XAMPP.

Algo importante que se debe hacer es deshabilitar los servicios Apache y MySQL una vez se dejen de utilizar para que no quede ejecutados siempre.

4.6.3 Base de datos con XAMPP

Una vez realizados los pasos del apartado anterior, ya se puede utilizar el gestor de base de datos de XAMPP. Para acceder al mismo hay que abrir el navegador web, en mi caso Opera Web Browser, y en la barra de direcciones poner "localhost", abriéndose una página web de bienvenida, donde hay que pulsar en "phpMyAdmin".

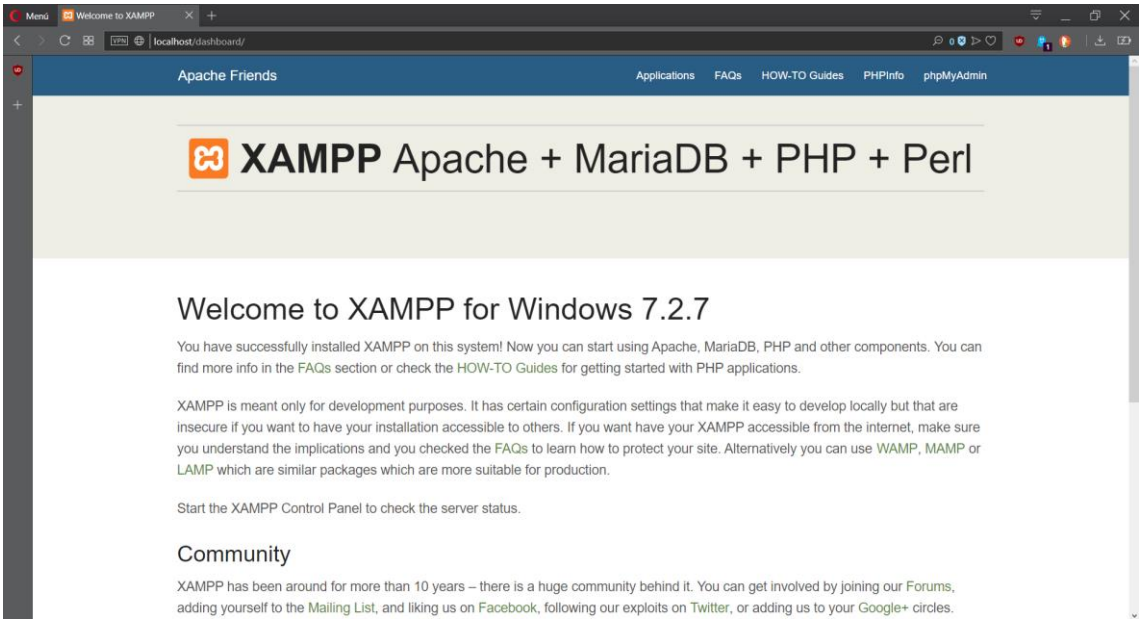

Ilustración 4.29.- Pagina bienvenida del Localhost

Luego de haber realizado este proceso, ya aparecerá el gestor de bases de datos.

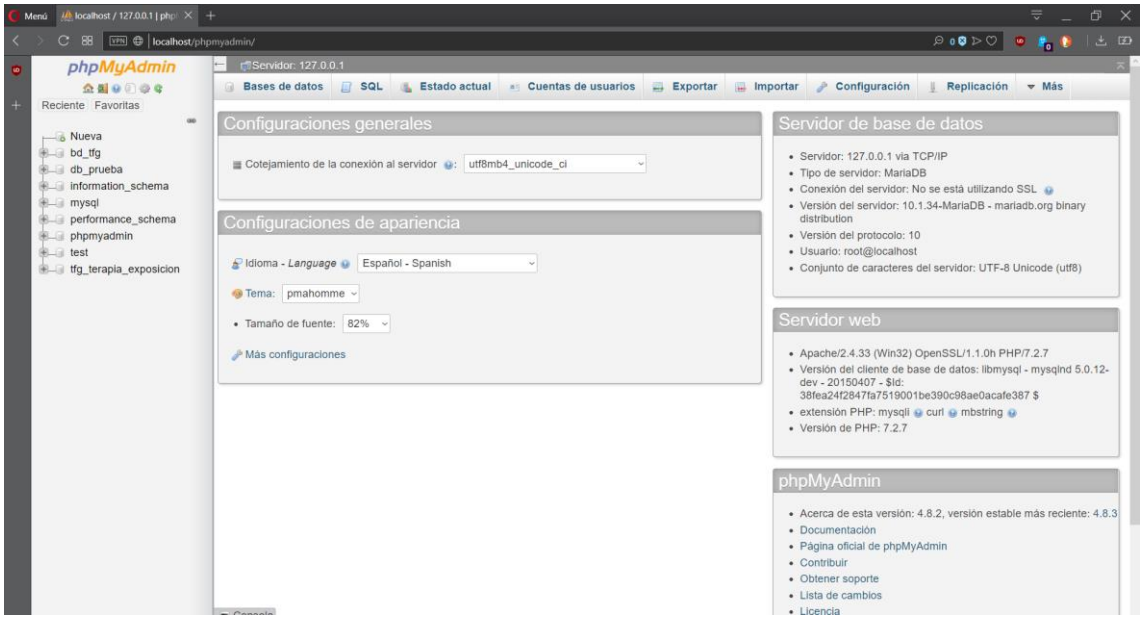

Ilustración 4.30.- Panel de control del gestor de bases de datos MySQL

4.6.3.1 Creación de una base de datos

Para crear una nueva base de datos hay que ir al panel izquierdo y hacer clic en "nueva". Posteriormente, se abre un asistente para poner el nombre y la codificación de la base de datos. Para este proyecto, se ha creado la base de datos con el nombre "bd_tfg" y se ha seleccionado la codificaicón *uft8_general_ci* para asegurar la compatibilidad del cotejamiento de datos.

Ilustración 4.31.- Creación base de datos (1)

Una vez realizado los pasos anteriores, ya tenemos la base de datos y hay que empezar a añadir las tablas de entidades y relaciones (en el caso de relaciones muchos es necesaria una tabla adicional). El proceso para crear una tabla es el siguiente:

- Hay que dar clic en la base de datos y posteriormente se abre apartado con las tablas creadas y la opción de añadir una nueva, donde hacemos clic.
- Ponemos el nombre y el número de columnas (serán de atributos que componen la entidad) y damos clic en "continuar".

Ilustración 4.32.- Creacion base de datos (2)

- A continuación, se abre el asistente grafico para rellenar los valores de la tabla. Se eligen los nombres de los atributos, el tipo de dato y en el caso de las ID o DNI se configuran como clave primaria. Al finalizar se da clic en guardar.

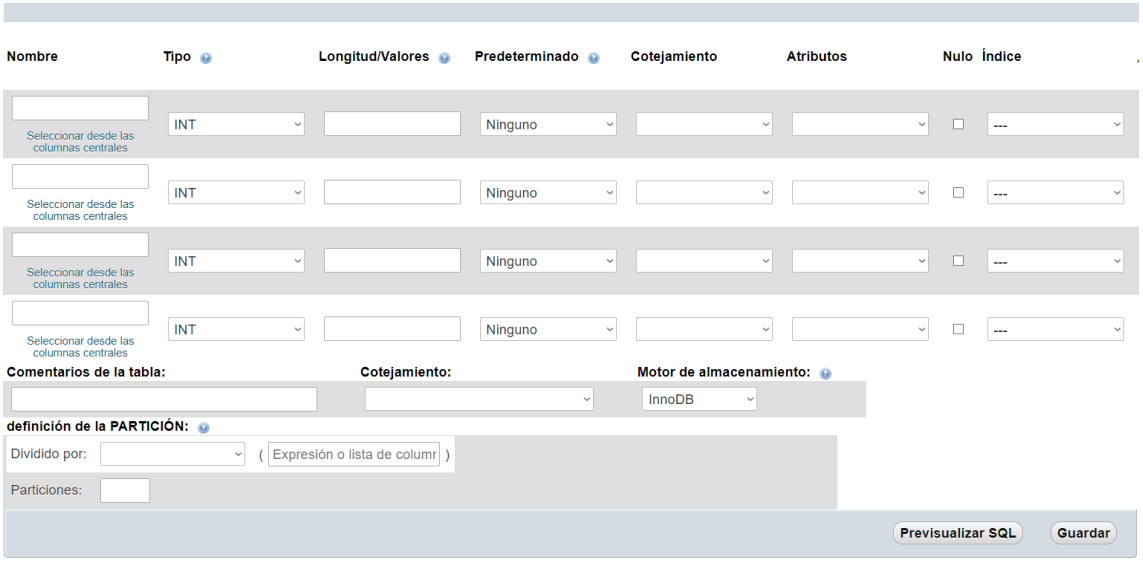

Ilustración 4.33.- Creación base de datos (3)

- El proceso es similar para todas las tablas que requiera la base de datos, para el proyecto se han creado las descritas en el primer apartado.

4.7 Servicios web

El ultimo subsistema es el de los servicios web. Como se ha descrito anteriormente, estos son los encargados de acceder a la base de datos para extraer o poner información en

ella. El tipo de servicios web usados son los servicios tipo REST. El motivo de su elección es por la sencillez y robustez que ofrecen.

En el caso del presente trabajo, los servicios creados serán utilizados por la aplicación Android y se han creado servicios REST tipo GET para extraer información de la base de datos, así como de tipo POST para insertar información en la misma.

4.7.1 Creación de un servicio web REST

Los servicios web se han programado en PHP, un lenguaje de código abierto muy popular en el desarrollo web, como es este el caso. El entorno de desarrollo utilizado en este caso es Visual Studio Code, un software libre de Microsoft que da soporte para PHP.

La instalación del software es muy sencilla. Tras descargar el archivo de instalación de https://code.visualstudio.com/, luego hay que hacer doble clic en el archivo descargado. Posteriormente, se abrirá el asistente de instalación y siguiendo los pasos que se indican por defecto se instala el programa.

4.7.1.1 Servicios web REST tipo GET

El proyecto consta de diferentes servicios web que han sido creados para acceder a la base de datos. En primer lugar, hay que establecer la ruta en el que estarán almacenados, ya que las URIs deben apuntar hacia alguna dirección web.

Para el desarrollo del proyecto se ha usado un servidor local Apache creado con XAMPP. Por este motivo, los servicios web están alojados en la maquina principal donde se ejecuta. La ruta de almacenamiento es la siguiente C:\xampp\htdocs\tfg\android. En concreto, en esta carpeta se encuentran todos los archivos *webservice.php*, los cuales se crean de la siguiente manera:

- Primero se ejecuta VS Code. Una vez abierto el programa por medio de la siguiente ruta, archivo/nuevo archivo, se crea un nuevo archivo con el nombre del servicio, por ejemplo "inicio_sesion.php", y se debe guardar en la ruta descrita en el párrafo anterior.
- Una vez creado el archivo procedemos a su programación.
- En la cabecera se declara unos valores constantes que es la información de la base de datos y una variable "array" llamado "json", que será el dato que devuelva el servicio en caso de ser ejecutado.

```php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "bd_tfg";

$json=array();
```

Ilustración 4.34.- Estructura servicio web (1)

- Después de esto, se declara un condicional *if*, que se ejecuta en caso de recibir los parámetros "dni, pwd y usuario" por medio del método _GET. Una vez dentro del condicional, el servicio prepara el acceso a la base de datos por medio de los métodos "mysqli_connect" y "mysqli_query".

```php
if(isset($_GET["dni"]) && isset($_GET["pwd"]) && isset($_GET["usuario"])){
    $dni=$_GET['dni'];
    $pwd=$_GET['pwd'];
    $usuario=$_GET['usuario'];

    $conexion = mysqli_connect($servername, $username, $password, $dbname);
    $consulta="SELECT dni, pwd, nombre FROM $usuario WHERE dni= '{$dni}' AND pwd= '{$pwd}'";
    $resultado=mysqli_query($conexion,$consulta);
```

Ilustración 4.35.- Estructura servicio web (2)

- Una vez ejecutada la consulta y en caso de ser exitosa, el servicio se encarga de empaquetar la información en el array "json" y la codifica en tipo JSON. Se cierra el acceso a la base de datos y devuelve el array como respuesta. En caso de que la respuesta a la consulta sea negativa, el servicio empaqueta un el array "json" vacío y lo envía.

```php
if($consulta){

    if($reg=mysqli_fetch_array($resultado)){
        $json['datos'][]=$reg;
    }
    mysqli_close($conexion);
    echo json_encode($json);
}

else{
    $results["dni"]='';
    $results["pwd"]='';
    $results["nombre"]='';
    $json['datos'][]=$results;
    echo json_encode($json);
}
```

Ilustración 4.36.- Estructura servicio web (3)

- Todos los servicios web REST tipo GET utilizados en el proyecto están programados de forma similar y siguiendo la misma estructura.

Capítulo 5

Conclusiones y trabajos futuros

5.1 Análisis de los resultados

Después de realizar el estudio, diseño y desarrollo del sistema expuesto en capítulos anteriores, se procedió a realizar su montaje para evaluar los resultados obtenidos y si son satisfactorios.

Las configuraciones realizadas para las pruebas fueron las siguientes:

- La base de datos se llenó de forma manual con datos de pacientes, terapeutas y terapias ficticios.
- Como actividad de pruebas se utilizó un vídeo explicativo de una terapia para pacientes con fobia a los insectos.

En la mayoría de los casos, las medidas de los sensores se efectuaron sin problemas, con algunas fluctuaciones en los valores, pero estos datos dependen de la precisión que ofrece la plataforma e-Health utilizada.

La realización de la actividad prueba se realizó satisfactoriamente, el registro de los datos en la base de datos también se realizó de forma correcta. En concreto, se probó desde la aplicación instalada en una Tablet, en el emulador de Android Studio y desde un navegador web.

Las gráficas de los datos fue la esperada según los datos recibidos por el sistema.

Por otra parte, las comunicaciones entre el ESP32 y el dispositivo Android se realizaron correctamente con algunos pequeños fallos que se encontraron, y que fueron depurados de forma satisfactoria usando el debugger de Android Studio.

5.2 Conclusiones

Realizadas las pruebas descritas en el apartado anterior, las conclusiones que se obtienen para este trabajo se describen a continuación:

- Se han evaluado las tecnologías disponibles para la realización del proyecto.
- Se han seleccionado dos parámetros relevantes en el seguimiento de pacientes en terapias de exposición.
- Se han seleccionado los dispositivos necesarios para el desarrollo del sistema.
- Se ha acondicionado el sistema para obtener las características requeridas.
- Se ha estudiado el diseño y arquitectura del sistema operativo Android y su entorno de desarrollo de aplicaciones para dispositivos móviles inteligentes.
- Se han programado todos los dispositivos y microcontroladores que componen el sistema.
- Se han creado la base de datos remota para alojar la información del proyecto.
- Se han creado los servicios web necesario para acceder de forma segura a la base de datos.
- Se realizaron las pruebas de funcionamiento necesarias para comprobar el funcionamiento.

Con lo anteriormente expuesto, se puede concluir que los objetivos planteados para el proyecto se han alcanzado de manera satisfactoria.

5.3 Trabajos Futuros

En el desarrollo del proyecto surgieron nuevas necesidades e ideas sobre el alcance de solución aportada. Siendo el resultado obtenido satisfactorio para los objetivos planteados, como trabajos futuros que quedaron fuera del alcance para este proyecto, se proponen los siguientes:

- Actualización de la plataforma e-Health. En la actualizad se encuentra disponible una nueva versión que ofrece mayores posibilidades.
- Diseño de un sistema empotrado para encapsular los componentes del Arduino y ESP32.
- Implementación de un sistema móvil de alimentación para las placas Arduino y ESP32.
- Incorporar algún parámetro para la identificación del estado del paciente. La incorporación de otros sensores aportaría más información sobre el estado del paciente, importante en el seguimiento de terapias de exposición.
- Aportar más funciones a la aplicación como puede ser el registro de usuarios, terapias y actividades, alarmas y notificaciones para la realización de terapias.
- Desarrollo de la aplicación para otros sistemas operativos como iOS.
- Exportar la base de datos a un servidor web de Internet o implementar servicios en la nube para alojar la información.
- Creación de formulario basado en HTML para insertar información a la base de datos de una forma más sencilla.

Referencias bibliográficas

Referencias

[1] "Wi-FI", *AulaClic*, 2005. [En línea]. Disponible en: https://www.aulaclic.es/articulos/wifi.html. [Accedido: 15-mar2018]

[2] "Bluetooth: ¿Qué es?", *Tecnologia-informatica*. [En línea]. Disponible en: https://tecnologia-informatica.com/bluetooth/. [Accedido: 15-mar2018]

[3] Jesus Macías, "Bluetooth BLE: el conocido desconocido", *Solidgeargroup*, 2017. [En línea]. Disponible en: https://solidgeargroup.com/bluetooth-ble-el-conocido-desconocido?lang=es. [Accedido: 15-mar2018]

[4] Enrique Crespo, "Qué es Arduino", *Aprendiendo Arduino*. [En línea]. Disponible en: https://aprendiendoarduino.wordpress.com/2016/09/25/que-es-arduino/. [Accedido: 15-mar2018]

[5] Roberto Carlos Cruceira, "Raspberry Pi: características y aplicaciones", *Ingenierate*, 2017. [En línea]. Disponible en: https://ingenierate.com/2017/10/03/raspberry-pi-caracteristicas-aplicaciones/. [Accedido: 15-mar2018]

[6] "Android", Blog Historia de la informática, 2012. [En línea]. Disponible en: http://histinf.blogs.upv.es/2012/12/14/android/. [Accedido: 20-mar2018]

[7] Marco Schwartz, "Review of Open-Source Healthcare Platforms & Sensors", *Openhomeautomation*, 2016. [En línea]. Disponible en: https://openhomeautomation.net/review-of-open-source-healthcare-platforms-sensors. [Accedido: 15-mar2018]

[8] "e-Health Platform", *Cooking Hacks*. [En línea]. Disponible en: https://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical. [Accedido: 20-feb2018]

[9] "Cooking hacks", *Cooking Hacks*. [En línea]. Disponible en: https://www.cooking-hacks.com/about-us/. [Accedido: 20-feb2018]

[10] Rubén Andrés, "Android mantiene su liderazgo como sistema más usado", *Computerhoy*, 2018. [En línea]. Disponible en: https://computerhoy.com/listas/tecnologia/android-mantiene-su-liderazgo-como-sistema-mas-usado-pero-iphone-8-vende-mas-281739. [Accedido: 28-agos2018]

[11] "Conoce Android Studio", *Android Developers*, 2018. [En línea]. Disponible en: https://developer.android.com/studio/intro/. [Accedido: 03-mar2018]

Bibliografía web Recomendada

Arduino. *Arduino Pro Mini*. Recuperado el 20 marzo 2018 de https://store.arduino.cc/arduino-pro-mini

Norficpc. *Versiones del sistema operativo Android*. Recuperado de https://norfipc.com/celulares/todas-versiones-sistema-operativo-android.php

Pedro Gutiérrez. (2013, noviembre 5). *Fundamento de las bases de datos: Modelo entidad-relación.* Recuperado de https://www.genbeta.com/desarrollo/fundamento-de-las-bases-de-datos-modelo-entidad-relacion

Damián Pérez Valdés. (2007, octubre 26). *¿Qué son las bases de datos?* Recuperado de http://www.maestrosdelweb.com/que-son-las-bases-de-datos/

phpMyAdmin. *Bringing MySQL to the web*. Recuperado de https://www.phpmyadmin.net/

Leonardo de la Seta. (2008, noviembre 13). *Introducción a los servicios web RESTful.* Recuperado de https://dosideas.com/noticias/java/314-introduccion-a-los-servicios-web-restful

Luis del Valle Hernández. *XML y JSON: Intercambiar información.* Recuperado de https://programarfacil.com/blog/xml-y-json-intercambiar-informacion/

Jose Mª Baquero García. (2015, agosto 2). *¿Qué son los web services y qué tecnología usar en su desarrollo?* Recuperado de https://www.arsys.es/blog/programacion/diseno-web/web-services-desarrollo/

Cristian Henao. (2017, febrero 17). *Curso Android desde Cero.* Recuperado de https://www.youtube.com/playlist?list=PLAg6Lv5BbjjdvIcLQdVg4ROZnfuuQcqXB

Mohammad Afaneh. (2017, junio 27). *Bluetooth GATT: How to Design Custom Services & Characteristics.* Recuperado de https://www.novelbits.io/bluetooth-gatt-services-characteristics/

Bluetooth SIG. *GATT Specifications.* Recuperado de https://www.bluetooth.com/specifications/gatt

Sgologier. (2012, julio 29). *Tareas en segundo plano en Android (I): Thread y AsyncTask.* Recuperado de http://www.sgoliver.net/blog/tareas-en-segundo-plano-en-android-i-thread-y-asynctask/

Myriam Gómez Obregón. *La Terapia de Exposición, en qué consiste.* Recuperado de https://www.psicoactiva.com/blog/la-terapia-de-exposicion-en-que-consiste/

Silicon Labs. (2015, junio 8). *BLE master/slave, GATT client/server, and data RX/TX basics.* Recuperado de https://www.silabs.com/community/wireless/bluetooth/knowledge-base.entry.html/2015/08/06/_reference_ble_mas-gviy

Abhay Anand. (2016, junio 6). *Android Line Chart or Line Graph using MpAndroid Library Tutorial.* Recuperado de https://www.studytutorial.in/android-line-chart-or-line-graph-using-mpandroid-library-tutorial

Kasia Mikoluk. (2013, septiembre 18). *XAMPP Tutorial: How to Use XAMPP to Run Your Own Web Server.* Recuperado de https://blog.udemy.com/xampp-tutorial/

Espressif Systems. *Arduino core for ESP32 WiFi chip*. Recuperado de https://github.com/espressif/arduino-esp32

Garcia, S. Garzón, L. Camargo, L. (1997). La auto exposición y prevención de respuesta en un caso de trastorno obsesivo-compulsivo con rituales de comprobación. *Revista de Psicopatología y Psicología Clínica*. Recuperado de https://dialnet.unirioja.es/descarga/articulo/4016780.pdf

Robert, C. (2011, enero – junio). Revisión de dispositivos electrónicos para la determinación de estrés a partir de variables fisiológicas. *Revista Visión Electrónica*. Recuperado de http://www.aepcp.net/arc/Vol.%202.%20N1,%20pp.%2083-96,%201997.pdf