Escuela Politécnica Superior de Linares

Grado en ingeniería telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

DISEÑO DE UN SISTEMA DE TELEMONITORIZACIÓN DE SIGNOS VITALES

Alumno: **Maria Josefa Fernández Guillén**

Tutor: Prof. D. Manuel Ángel Gadeo Martos
Depto.: Ingeniería de Telecomunicación

Junio, 2021

Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

Curso 2020-2021

DISEÑO DE UN SISTEMA DE TELEMONITORIZACIÓN DE SIGNOS VITALES

Alumno: Maria Josefa Fernández Guillén

Tutor: Prof. D. Manuel Ángel Gadeo Martos

Depto.: Ingeniería de Telecomunicación

Junio, 2021

Visto bueno a la defensa del TFG

Firma del autor/a

Fernández Guillén, Maria Josefa
DNI:15521109L

Firma del tutor

GADEO MARTOS MANUEL ANGEL - 25987883E

Firmado digitalmente por GADEO MARTOS MANUEL ANGEL - 25987883E
Fecha: 2021.06.21 13:37:19 +02'00'

Junio, 2021

Agradecimientos: A Pedro Aguilar Aguilar que sin su ayuda no hubiese finalizado el trabajo de fin de grado para la fecha propuesta.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. RESUMEN

El objetivo de este trabajo de fin de grado trata del diseño de un sistema de telemonitorización de los signos vitales con interfaz para la interacción médico-paciente, que permita monitorizar signos vitales de pacientes en el área local, es decir, en su domicilio. Se ha diseñado un dispositivo portátil implementado en el cuerpo del paciente con sensores que miden los signos vitales (frecuencia cardiaca, saturación de oxígeno, temperatura corporal, aceleración, etc). El dispositivo cuenta con una base de Arduino Due, el cual procesa y guarda en memoria, los datos de los signos vitales tomados para consecuentemente ser enviados en tiempo real, mediante un módulo Wifi ESP8266-01 localmente. Los datos son enviados al servidor XAMPP y guardados en la base de datos MySQL, y posteriormente, visualizarlos en la aplicación web destinada para el médico, denominada “Centro SigVit”. Finalmente, el sistema incluye un método para determinar automáticamente el nivel de alerta del estado del paciente utilizándose un sistema borroso basado en reglas.

CAPÍTULO 2. INTRODUCCIÓN

2.1 Introducción

La telemonitorización es una especialidad dentro de telemedicina cuyo propósito es hacer un seguimiento de algunos parámetros biológicos del paciente desde su domicilio. En esencia es un proceso similar al que se hace en una Unidad de Cuidados Intensivos, donde el paciente tiene conectado un monitor y un profesional sanitario hace el seguimiento de unos parámetros: pulso, etc.

En el caso de la telemonitorización el paciente está en su casa y de forma periódica (diaria, semanal, etc) envía sus datos biológicos a un servidor y una base de datos para su posterior visualización en una aplicación. Cuando un parámetro presenta alguna anormalidad, el profesional sanitario se pone en contacto con el paciente, para indicar lo que tiene que hacer. Esta comunicación la inicia el profesional y el objetivo es actuar ante una alerta y por lo tanto no hay que confundirlo con una llamada telefónica, que es otra fórmula diferente de telemedicina. Más información [1].

Los signos vitales son parámetros fisiológicos que indican la condición física en la que se encuentran la persona. Entre los principales parámetros son el oxígeno en sangre, la temperatura, la frecuencia respiratoria y la frecuencia cardíaca.

2.2 Estado del arte

La actual pandemia ha sacado a la luz numerosas deficiencias en materia de salud, sobre todo en la que respecta a nuestros mayores. Por ello, actualmente existen muchas opciones para dar soluciones a la hora de implementar sistemas de telemonitorización en el hogar y controlar signos vitales.

La telemedicina está al orden del día, la causa de ello es el Covid19, por lo que los médicos desde la distancia controlan algunos parámetros de los pacientes comprobando su estado de salud y evitando la saturación en los hospitales.

Un inconveniente de esos parámetros es la gran incertidumbre de que sean exactos, ya que el médico no estará en ese momento para comprobar su correcto funcionamiento.

Por ello se propone el diseño de un sistema de telemonitorización de signos vitales, tales como, el oxígeno en sangre, la frecuencia cardíaca, etc. A su vez se

utiliza un sistema inteligente denominado sistema borroso basado en reglas, para que el rango de incertidumbre sea menor.

Finalmente, es un diseño para la red de área local (LAN), es decir, la red doméstica, que facilita la comunicación mediante red Wifi, junto a la red de sensores implementados en el paciente y utilizando un servidor local, para el envío de los datos a una base de datos y posteriormente sean visualizados por el médico.

CAPÍTULO 3. OBJETIVOS

En este capítulo, se describen los objetivos para la realización del diseño. Se han expuesto los objetivos generales, tanto los objetivos específicos que se han ido incorporando al proyecto durante el proceso de su realización.

- Aprendizaje de los conocimientos básicos de signos vitales.
- Búsqueda/Aprendizaje de las placas Arduino e IDE (en inglés “Integrated Development Environment”) Arduino.
- Selección de los componentes para la obtención de los signos vitales de personas y para la comunicación inalámbrica el envío de datos
- Búsqueda de información para la implementación del sistema borroso basado en reglas.
- Diseño de la aplicación web para la iteración médico-paciente mediante la visualización de signos vitales durante el alojamiento en el domicilio.
- Configurar y programar un nodo de una red de sensores inalámbricos para capturar periódicamente datos sobre los signos vitales de una persona.
- Diseñar un protocolo de comunicaciones para transmitir los datos capturados por los nodos sensores de la red a un servidor de comunicaciones.
- Diseño de un algoritmo que a partir de las variables de los signos vitales obtenidos permita detectar situaciones de deterioro de la salud, y dispare los correspondientes mensajes de alarma.
- Diseñar un servidor web que permita el almacenamiento, acceso de los datos capturados, así como la gestión de los nodos sensores.

CAPÍTULO 4. TECNOLOGÍAS UTILIZADAS

En este capítulo se muestran las tecnologías que se replantearon antes de la elección y las que se escogen para la realización del diseño.

4.1 Introducción

Una red de sensores inalámbrica se basa en dispositivos de bajo coste y consumo, que son capaces de obtener información de su entorno, procesarla localmente, y comunicarla a través de enlaces inalámbricos. La red de sensores inalámbrica está formada por numerosos nodos distribuidos, que utilizan sensores para leer los distintos parámetros en distintos puntos, entre ellos la temperatura, nivel de oxígeno en sangre, frecuencia cardiaca, etc.

4.2 Tipos de placas

Se opta por la plataforma de hardware libre denominada Arduino, que se basa en una placa que tiene un microcontrolador y un entorno de desarrollo por medio de software. Las ventajas de Arduino son amplias, por ejemplo: bajo coste, plataforma abierta, gran variedad de placas, etc., más ventajas en [2]. Para programar la placa se necesita el IDE Arduino.

4.2.1 Arduino Uno

Esta placa tiene todo lo necesario para manejar el controlador, simplemente conectamos al computador por medio del cable USB o una fuente de poder externa, que puede ser un adaptador AC-DC o una batería, cabe aclarar que si se alimenta a través del cable USB en el ordenador no es necesario una fuente externa (**Figura 4.1**). Sus características son: [3]

- **Microcontrolador:** ATmega328
- **Voltaje Operativo:** 5v
- **Voltaje de Entrada (Recomendado):** 7-12 v
- **Pines de Entradas/Salidas Digital:** 14 (de las cuales 4 pueden ser utilizados para salidas PWM)
- **Pines de Entradas Análogas:** 6
- **Memoria Flash:** 32 KB (ATmega328) de los cuales 0,5KB es usado por Bootloader.
- **SRAM:** 2 KB (ATmega328)
- **EEPROM:** 1 KB (ATmega328)
- **Velocidad del Reloj:** 16 MHZ

- **Precio:** 19 y 20 euros. [4]

FIGURA 4.1 ARDUINO UNO

4.2.2 Arduino Due

Arduino Due fue la primera placa controladora Arduino con un procesador de 32 bits. A diferencia de otras tarjetas, trabaja con 3,3V, pudiendo tolerar un voltaje máximo en sus pines I/O de 3.3V. Por lo que alimentar los mismos con voltajes más altos, como por ejemplo 5V, puede dañar la tarjeta (**Figura 4.2**). Características: [5]

- **Microcontrolador:** AT91SAM3X8E
- **Voltaje de operación:** 3.3V
- **Voltaje recomendado de entrada (pin Vin):** 7-12V
- **Pines de entrada y salida digitales:** 54 pines I/O, de los cuales 12 proveen salida PWM.
- **Pines de entrada análogos:** 12 (12 bits)
- **Pines de salida análogos:** 2 (12 bits)
- **Corriente de salida total en los pines I/O:** 130mA
- **Corriente DC máxima en el pin de 3.3V:** 800mA
- **Corriente DC máxima en el pin de 5V:** 800mA
- **Memoria Flash:** 512 KB toda disponible para aplicaciones del usuario.
- **SRAM:** 96 KB (en dos bancos de: 64KB y 32KB)
- **4 BUS Serie.**
- **1 BUS SPI.**
- **2 BUS I2C.**
- **Velocidad de reloj:** 84 MHz
- **Precio:** 33.60 euros [6]

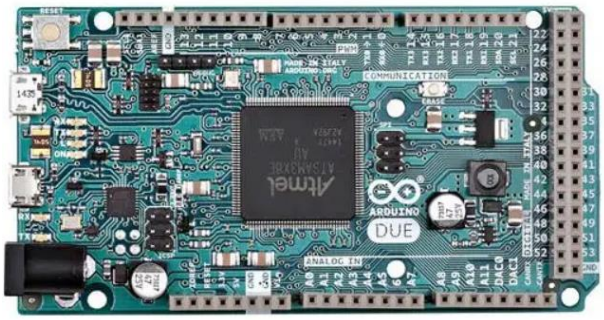


FIGURA 4.2 ARDUINO DUE

4.2.3 Arduino Nano

Tiene funcionalidades similares a las de Arduino Uno. Aunque la principal diferencia entre ellos es que el Arduino Uno se presenta en forma de PDIP (Plastic Dual-In-Line Package) con 30 pines y la Arduino Nano está disponible en TQFP (Plastic Quad Flat Pack) con 32 pines. Los dos pines adicionales de la Arduino Nano sirven para las funcionalidades del ADC, mientras que Uno tiene 6 puertos ADC, pero la Nano tiene 8 puertos ADC. (**Figura 4.3**). Sus características: [7]

- **Microcontrolador:** ATmega328
- **Arquitectura:** AVR
- **Voltaje de operación:** 5V
- **Memoria Flash:** 32 KB de los cuales 2 KB utilizados por bootloader
- **SRAM:** 2 KB
- **Velocidad del reloj:** 16 MHz
- **Pines de E/S analógicas:** 8
- **EEPROM:** 1KB
- **Corriente continua por pin entrada-salida:** 40mA
- **Voltaje de entrada:** 7-12V
- **Pines de E/S digitales:** 22
- **Salida PWM:** 6
- **Consumo de energía:** 19mA
- **Precio:** 24.20euros [8]

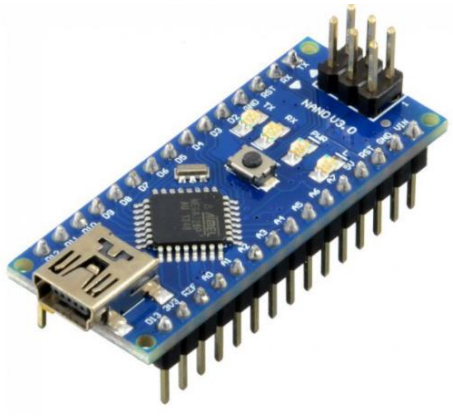


FIGURA 4.3 ARDUINO NANO

4.2.4 Elección

Se ha optado por la placa Arduino Due ya que contiene un procesador de 32 bits. Proporciona una velocidad de reloj interno de 84MHz, con el sistema de programación sencillo de Arduino. Contiene memoria de 96 KB de RAM (en dos bancos de 64 KB y 32 KB) y 512 KB de FLASH. Ver apartado **Arduino Due 4.2.2**.

4.3 Tecnologías inalámbricas

La tecnología inalámbrica se utiliza para transmitir los parámetros recogidos de los sensores del paciente al servidor. A continuación, se presentan los puntos de referencia de los estándares inalámbricos más importantes para seleccionar la opción más adecuada. Más información [9].

4.3.1 Wifi

Tecnología de transmisión de datos inalámbrica usada principalmente para Internet. Está basada en el estándar 802.11 y permite la conexión inalámbrica entre diferentes tipos de dispositivos electrónicos como ordenadores, móviles, etc.

Sus características:

- **Ancho de banda:** 2.4GHz y 5 GHz.
- **Distancia:** alcanza hasta los 100 metros.
- **Dispositivos:** requiere de configuración previa para poder utilizarla.
- **Consumo de energía:** bajo
- **Velocidad:** buena
- **Calidad transferencia datos:** buena.

4.3.2 Bluetooth

Bluetooth es una tecnología de radio corto alcance, de pequeña escala y bajo coste que permite la comunicación entre dispositivos en red y entre dispositivos e Internet a alta velocidad; también simplifica la sincronización de datos entre dispositivos. Esta tecnología se basa en el estándar IEE 802.15.1

Es un protocolo para la comunicación inalámbrica entre dispositivos que aporta sencillez de conexión y favorecer la interacción entre dispositivos sin cable. La versión actual es la 5.0, permiten que dos dispositivos se conecten entre sí mediante transmisiones de radiofrecuencia. Además, permite la transferencia de voz y datos punto a punto sin conexión y está orientada a la conexión entre dos dispositivos digitales.

Estándar de tecnología inalámbrica creado para la comunicación de corto alcance. Características:

- **Ancho de banda:** 2.4GHz
- **Distancia:** menos de 10 metros
- **Dispositivos:** fácil de emparejar los dispositivos
- **Consumo de energía:** alto
- **Velocidad:** más lento que el wifi
- **Calidad de transferencia de datos:** peor que por wifi

4.3.3 Elección

Una vez realizado un análisis comparativo de las tecnologías inalámbricas más importantes o utilizadas, se ha optado por utilizar la tecnología Wifi, para la comunicación entre nodos y servidores por las siguientes razones:

- **Bajo coste:** infraestructura presente en varios entornos de la vida cotidiana.
- **Velocidad de transferencia:** Alta.
- **Fiabilidad:** Alta.
- **Consumo energético:** bajo (mA).

4.4 Módulos ESP8266-XX

El módulo ESP8266 se trata de un chip integrado con conexión Wifi y compatible con el protocolo TCP/IP. El objetivo principal es dar acceso a cualquier microcontrolador a una red. La gran ventaja del ESP8266 es su bajo consumo.

FIGURA 4.4 TIPOS DE MODELOS ESP8266

De los doce modelos que existen del ESP8266 (**Figura 4.4**), se comentarán a continuación tres de ellos. Más información [10].

4.4.1 ESP8266-01

Se trata del módulo más popular, contiene dos pines GPIO digitales para la comunicación, el pin Rx (pin de recepción) y Tx (pin de transmisión). La disposición de los ocho pines que contiene el módulo, no permiten conectarlo directamente a la protoboard. (**Figura 4.5**).

Características principales:

- **Wifi:** 2.4GHz
- **Soporta** WPA/WPA2
- **Modo Deep Sleep:** <10 uA
- **Amplio alcance:** Se obtiene por el estándar 802.11 b/g/n
- **Múltiples aplicaciones, menor costo**.
- **Tres modos de operación:** Wifi AP, Wifi client (STA) y ambos simultáneamente (AP+STA).
- **Comunicación:** Se realiza a través de dos pines GPIO
- **Integrado TCP/IP**
- **Precio:** 2-5 euros. [11]

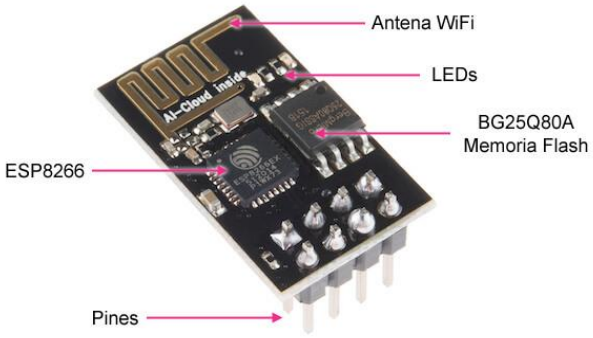


FIGURA 4.5 MODELO ESP8266-01

4.4.2 ESP8266-05

Este módulo está basado en el ESP8266 con una sencilla interfaz de 5 pines con las que puedes agregar Wifi a cualquier proyecto. Puede ser el módulo más simple de todos los modelos, y la posición de los cinco pines nos permite la conexión directa con la protoboard. Aunque no dispone de ningún puerto GPIO accesible. (**Figura 4.6**).

Características principales:

- **No dispone de puerto GPIO accesible**
- **Conexión** recomendable a 3.3V
- **Precio:** 3-4 euros

FIGURA 4.6 MODELO ESP8266-05

4.4.4 NodeMCU

NodeMCU es una placa de desarrollo basada en el ESP12E y expone las funcionalidades y capacidad del mismo. Lo más interesante de este módulo es que

tiene para descargar un firmware que permite programar lenguajes como LUA, Python, Basic o JavaScript. (**Figura 4.7**). Características:

- **Puerto** micro USB y conversor Serie-USB
- **Programación** sencilla a través del Micro-USB
- **Alimentación** a través del USB
- **Terminales**(pines) para facilitar la conexión
- **LED y botón de reset** integrados
- **Precio:** ~6 euros

FIGURA 4.7 NODEMCU

Existen tres versiones de NodeMCU, la versión 1 no está a la venta, la versión 2 es la "oficial" y la versión 3 es muy parecida a la versión 2, solo cambia el conversor serial.

4.4.5 Elección

El dispositivo elegido para este prototipo es el módulo Wifi ESP8266-01, para realizar la conexión inalámbrica de los sensores. Debido a las siguientes características:

- Calidad
- Flexibilidad
- Bajo coste
- Gran cantidad de documentación disponible en la red
- Código abierto

4.5 Sensores

Un sensor es un objeto cuyo propósito es detectar eventos o cambios en su entorno, y luego proporcionar una salida correspondiente, es decir, nos proporciona datos a través de su entrada [12].

El término sensor se usa para referirse a un elemento que produce una señal en respuesta a la detección o a la medida de alguna propiedad, como la presión, la fuerza, la temperatura, la humedad, etc. Según la señal de salida, los sensores se clasifican en sensores analógicos, que son los que producen una señal analógica continua, como el voltaje. Y los sensores digitales que tienen salidas numéricas o digitales, que su salida varia en forma de saltos o pasos discretos, este tipo de sensores tienen mayor fidelidad y fiabilidad, y a veces mucha exactitud. Para seleccionar un sensor, existen diversos factores, como saber la naturaleza de la medida y de salida requeridas. Luego, se identifica el sensor teniendo en cuenta algunos factores, como alcance, precisión, linealidad, velocidad, robustez, costo, etc.

Los parámetros que se necesitan obtener son: Oxígeno en Sangre, frecuencia cardíaca, temperatura, posición, aceleración, etc.

4.5.1 Sensor Oxígeno en Sangre y Frecuencia cardiaca

Para la obtención de estos dos parámetros en un mismo sensor se ha encontrado el sensor max30100.

4.5.1.1 GY-Max30100

Para la medición de la saturación de oxígeno en sangre y la frecuencia cardiaca se utiliza el dispositivo MAX30100, que por el reducido tamaño y el bajo costo lo hace perfecto para su utilización. Este dispositivo combina dos Leds (led rojo 660nm y led infrarrojo 920 nm), un fotodetector y procesamiento de señal analógica de bajo ruido para detectar pulsioximetría y señales de frecuencia cardiaca. Más información en la hoja de datos “datasheet” [13].

El MAX30100 presenta un interfaz en serie de 2-Wire compatible con I2C que consta de línea de datos serie (SDA) y una línea de reloj serie (SCL). SDA Y SCL facilitan la comunicación a velocidades de reloj de hasta 400kHz. (**Figura 48**).

Algunas características:

- **Voltaje de funcionamiento:** 3-5V
- **Interface:** I2C
- **Temperatura de operación:** -40ºC ~ 85ºC

FIGURA 4.8 SENSOR MAX30100

4.5.2 Sensores Temperatura

Recoge la temperatura del exterior y la transforma en señal digital o electrónica que envía a una placa electrónica como puede ser una placa Arduino. A continuación, se muestran los sensores que se plantearon y la elección.

4.5.2.1 LM35

La salida de este sensor es analógica y el calibrado lo hace directamente en grados Celsius. La temperatura que admite oscila entre los 2ºC y los 150ºC. Sensor muy económico. (**Figura 4.9**).

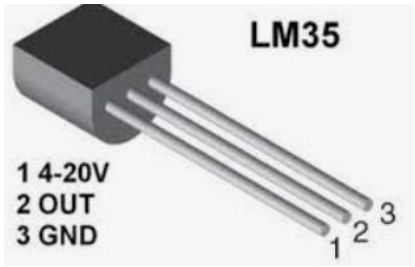


FIGURA 4.9 SENSOR LM35

4.5.2.2 DS18B20

Es un sensor digital y está sellado en un envoltorio, lo que permite sumergirlo en líquidos o/y protegerlo, Tiene una precisión de ±0.5ºC con resolución de 12 bits. Rango de temperatura de -55ºC a 125ºC. (**Figura 4.10**). Más información [14].

FIGURA 4.10 SENSOR DS18B20

4.5.2.3 max30100

Es un sensor de pulsioximetría y monitor de frecuencia cardíaca. Además, posee un sensor de temperatura integrado El sensor de temperatura tiene una asombrosa precisión de ±0.6ºC. Ver **figura 4.8**.

4.5.2.4 Elección

En un principio se opta por el sensor DS18B20, pero tiene un inconveniente, su diseño es incómodo para la colocación del mismo. Además, de ser muy preciso, el sensor elegido ha sido el "max30100". Por lo tanto, con este sensor obtenemos tres parámetros con un solo sensor, lo que hace que se abarate los costes.

4.5.3 Sensor acelerómetro

Un acelerómetro sirve para controlar la aceleración y la velocidad de las vibraciones. En este diseño se quiere utilizar el sensor acelerómetro para detectar un movimiento brusco, la posición en la que se encuentra el paciente, etc., ver **apartado 5.1.1.2**. Se muestra los sensores acelerómetros que se barajan para la elección.

4.5.3.1 mma7361

Acelerómetro analógico de tres ejes (x, y, z). Nos permite medir la aceleración, o la inclinación de una plataforma con respecto al eje terrestre. Además, obteniendo el valor de los tres ejes, se pueden realizar múltiples funciones [15].

Especificaciones:

- **Bajo consumo de corriente:** 400uA
- **Bajo voltaje de la operación:** 2.2 ~ 3.6 V
- **Alta sensibilidad**

- **0g-Detect** para la protección de caída libre

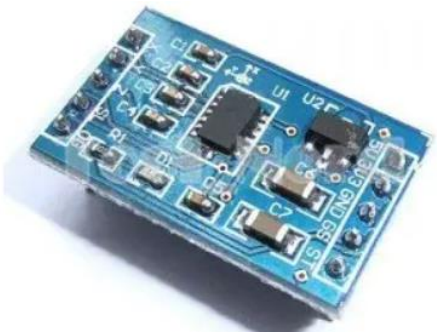


FIGURA 4.11 SENSOR MMA7361

4.5.3.2 ADXL345

Acelerómetro capacitivo de tres ejes digitales que detecta la aceleración en los ejes x, y, z. Más información [16].

Especificaciones:

- **Interfaz digital** I2C y SPI
- **Voltaje de entrada:** 3.3~5V
- **Voltaje de operación:** 2~3.6V
- **Corriente de operación:** 140uA
- **Detección** caída libre
- **Frecuencia de reloj interna:** 400KHz

FIGURA 4.12 SENSOR ADXL345

4.5.3.3 Elección

El sensor adxl345 puede detectar de -16 a 16 g (unidad que mide la fuerza de gravedad, 1g es 9,8m/s^2), por lo que no proporciona una salida precisa y en el caso de detección del movimiento no se necesita tanto rango.

Se ha optado por la elección del sensor mma7361, tiene un rango seleccionable de 1.5 o 6 g, por lo que es bueno para detectar rango y sensibilidad.

4.6 Servidor

Un servidor es un equipo que forma parte de una red y provee servicios a otros equipos, por lo cual, se necesita para recibir los datos obtenidos de los sensores y almacenarlos en una base de datos para su posterior procesamiento y/o visualización a través de una aplicación web.

4.6.1. Elección del servidor

Se ha elegido XAMPP como servidor para este diseño, ya que es el entorno más popular de desarrollo de aplicaciones que incorpora el servidor Apache, un sistema gestor de base de datos MySQL y un lenguaje de programación PHP, ver f**igura 4.14**. Más información [17].

Además, recibe y responde peticiones vía HTTP, ver **figura 4.13**, en la cual, se presenta la arquitectura del servidor XAMPP para una mejor visualización de su funcionamiento.

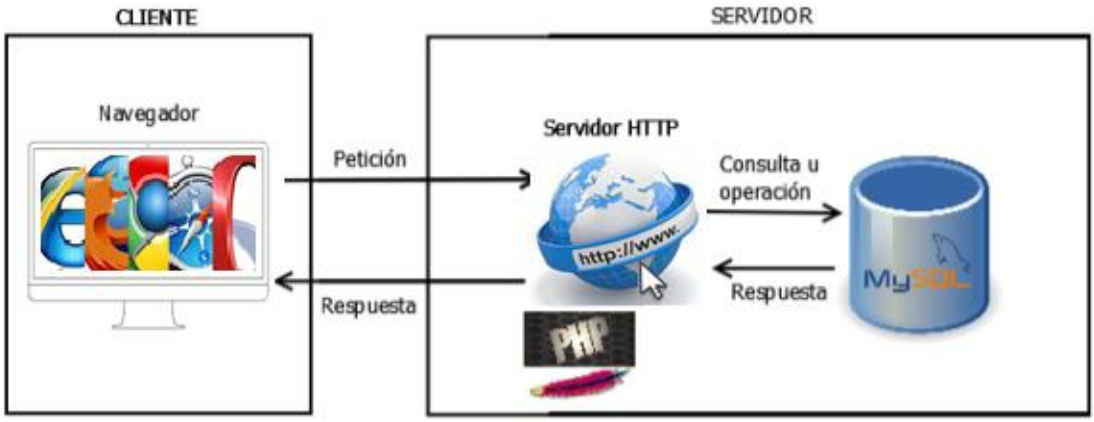


FIGURA 4.13 ARQUITECTURA XAMPP

El paquete de instalación de XAMPP ha sido diseñado para ser fácil de instalar y usar.

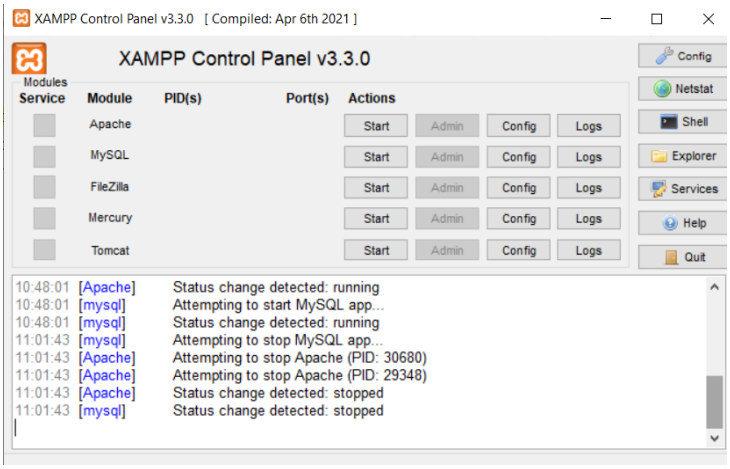


FIGURA 4.14 XAMPP SIN INICIAR

4.6.1.1 Panel de control

En el panel de control se controlan todas las acciones y donde se activan/desactivan los módulos por separado con un click en el botón “Start” o “Stop”, ver **figuras 4.14** y **4.15**.

Dispone de diversas utilidades:

- **Config:** Para configurar XAMPP u otros componentes.
- **Netstat:** Muestra todos los procesos en funcionamiento en el ordenador local.
- **Shell:** Lanza una ventana de comandos UNIX.
- **Explorer:** Abre la carpeta XAMPP en el explorador de Windows.
- **Services:** Muestra todos los servicios en funcionamiento.
- **Help:** Incluye enlaces a foros de usuarios.
- **Quit:** Se usa para salir del panel de control.

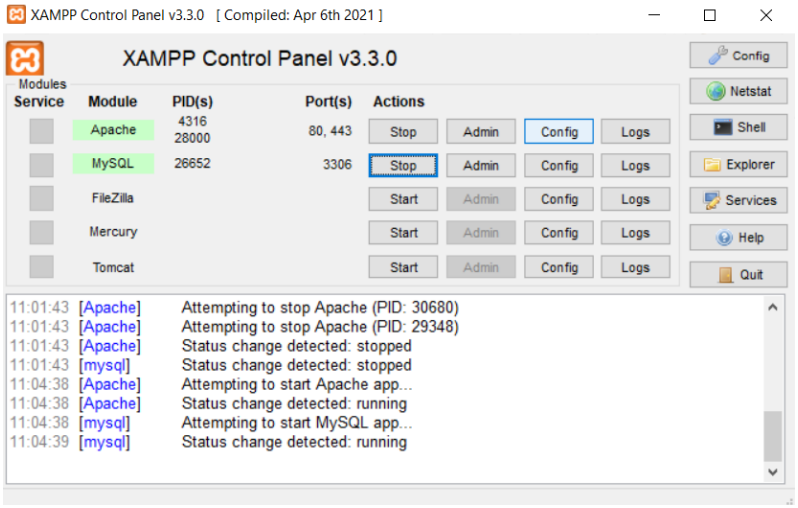


FIGURA 4.15 XAMPP INICIADO

4.7 Base de datos

Una base de datos es una colección organizada de información estructurada, o datos, típicamente almacenados electrónicamente en un sistema de computadora. La mayoría de las bases de datos utilizan lenguaje de consulta estructurado (SQL) para escribir y consultar datos.

El almacenamiento de datos se debe realizar en una estructura flexible que nos permita introducir, modificar y extraer información de forma organizada, evitando duplicidades y errores.

4.7.1 Elección del sistema gestor de base de datos

Se ha optado por MySQL porque es la base de datos de código abierto más famosa y utilizada en todo el mundo. MySQL es un sistema de gestión de bases de datos que cuenta con una doble licencia. Una de las principales características es que trabaja con base de datos relacionales, es decir, utiliza múltiples de tablas que se interconectan entre sí para almacenar la información y organizarla correctamente. Cabe destacar que es un sistema desarrollado en C y C++, su entorno es gratuito y modificable. Dan forma y facilitan la comunicación entre webs y servidores [18].

Se accede a la base de datos (**figura 4.16**) haciendo click en el botón “Admin” del módulo MySQL de XAMPP. Ver **figura 4.15**.

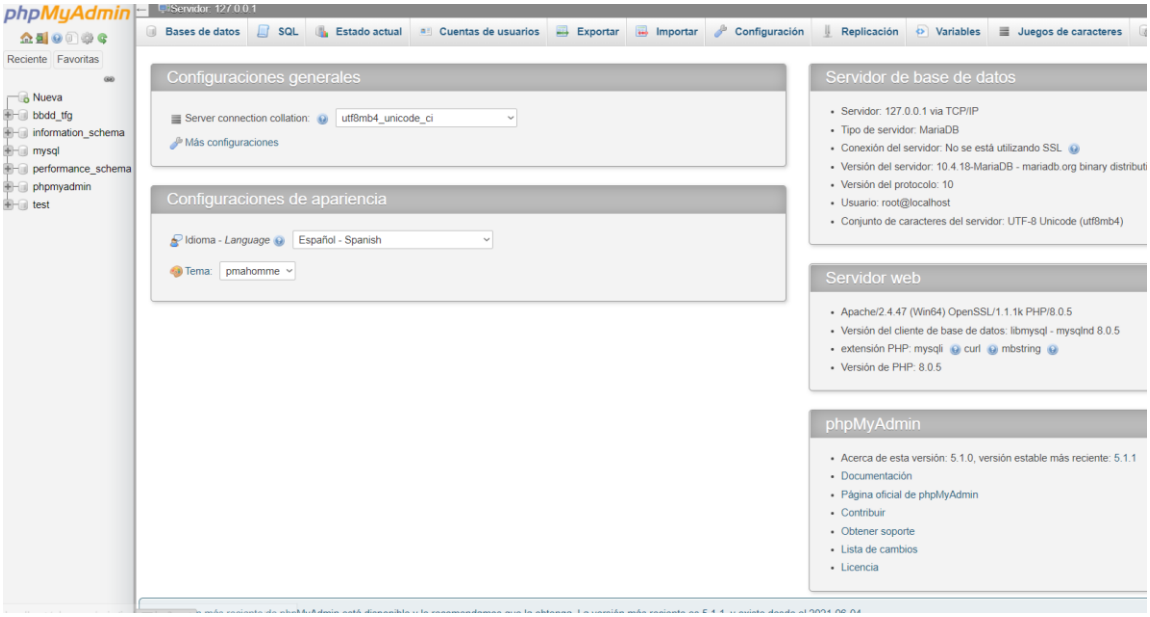


FIGURA 4.16 PHPMYADMIN

4.8 Aplicación web

Hay una serie de factores que influyen en el desarrollo de la aplicación web, como son: los lenguajes de programación, bibliotecas, estructura de datos, entornos de desarrollo y estilos de programación, etc. Pero estos aspectos se discutirán en los siguientes apartados.

4.8.1 Entorno de desarrollo

Un entorno de desarrollo es un conjunto de procedimientos y herramientas que se utilizan para desarrollar un código fuente o programa.

La elección de un IDE, un entorno de desarrollo, está estrechamente relacionado con la tecnología utilizada. Como lenguaje de programación, PHP se usa en el lado del servidor y las versiones HTML y CSS se usan para el desarrollo de scripts, y sitio web.

Hay muchos IDE que satisfacen estas necesidades de diseño, como NetBeans, Eclipse, Visual Studio Code. Tanto NetBeans como Eclipse proporcionan los módulos necesarios, mientras que NetBeans debe descargarse individualmente, Eclipse los proporciona de forma predeterminada.

4.8.1.1 NetBeans

NetBeans es un entorno de desarrollo integrado libre, orientado principalmente al desarrollo de aplicaciones Java. La plataforma NetBeans permite el desarrollo de aplicaciones estructuradas mediante un conjunto de componentes denominados “módulos”.

4.8.1.2 Eclipse

Es un entorno de desarrollo integrado, de código abierto y multiplataforma. Se utiliza mayormente para desarrollar aplicaciones de cliente, basado en el lenguaje Java. Eclipse tiene un módulo para descargar denominado: Spring MVC, el cual está dirigido a facilitar y optimizar el proceso de creación de aplicaciones web utilizando el patrón MVC (Modelo-Vista-Controlador). (**Figura 4.17**).

- El **Modelo:** representa los datos o información que manejará la aplicación web.
- La **Vista:** son todos los elementos de la interfaz del usuario, es decir, con ellos el usuario interactúa con la aplicación. Ejemplo: botones, campos de texto, etc.
- El **Controlador:** el encargado de manipular los datos en base de la interacción del usuario.

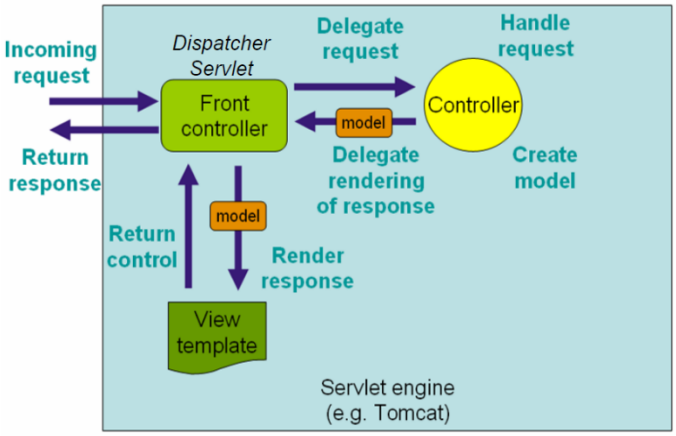


FIGURA 4.17 SPRING MVC

4.8.1.3 Visual Studio Code

Visual Studio Code es un editor de código fuente que funciona con muchos lenguajes de programación, tales como: C++, C#, PHP, Python, Java, etc., y admite la gestión de accesos directos únicos y la refactorización de código. Es de código abierto, gratuito y proporciona utilidades para descargar y administrar extensiones para personalizar y mejorar sus herramientas.

4.8.1.4 Elección entorno de desarrollo

Finalmente, Visual Studio Code ha sido elegido para el desarrollo de la página web, al ser gratuito, conveniente y simple.

4.8.2. Estructura de datos

La estructura de datos está representada por una forma determinada que tenemos de organizar los datos de un equipo informático para que podamos utilizarlos de la manera más efectiva posible.

Para la elaboración de la aplicación web, se ha aplicado un conjunto de lenguajes, tanto HTML, como PHP, y para hacer los estilos de la página CSS. A continuación, se definen.

4.8.2.1 PHP

PHP es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollador web y que puede ser incrustado en HTML. Como podemos ver en la siguiente **figura 4.18**:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Ejemplo</title>
    </head>
    <body>

        <?php
            echo "¡Hola, soy un script de PHP!";
        ?>

    </body>
</html>
```

FIGURA 4.18 EJEMPLO PHP DENTRO DE HTML

Lo que distingue a PHP es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP, por lo que los usuarios no puedan saber nada. Lo mejor de utilizar PHP es su extrema simplicidad, pero a su vez ofrece muchas características avanzadas para los programadores [19].

4.8.2.2 HTML

HTML (Hypertext Markup Language o lenguaje de marcas hipertexto) es un lenguaje de marcación que sirve para definir el contenido de las páginas web. Se compone a base de etiquetas, también llamadas marcas o tags, con las cuales conseguimos expresar las partes de un documento, cabecera, cuerpo, encabezados, párrafos, etc. En definitiva, el contenido de una página web. Más información [20].

4.8.2.3 CSS

CSS (Cascading StyleSheets o hojas de estilo en cascada), es un lenguaje que permite presentar de manera estructurada un documento que es escrito en un lenguaje marcado. Especialmente es usado en el diseño visual de la página web, escrita en XML o HTML, en este caso, escrita en HTML.

CSS se encarga de la descripción de las formas y de la sintaxis del lenguaje marcado. De esta manera describe cómo se tienen que renderizar (generar las imágenes) los elementos que aparecen en pantalla.

El diseño del CSS posibilita establecer una separación entre el contenido y la forma de presentación del documento (dada por las fuentes, colores y las capas empleadas). Así se puede lograr que muchos documentos HTML compartan la apariencia, utilizando una única hoja de estilo para todos (que se especifica en un archivo .css). Gracias a ello, se evita repetir código en la estructura.

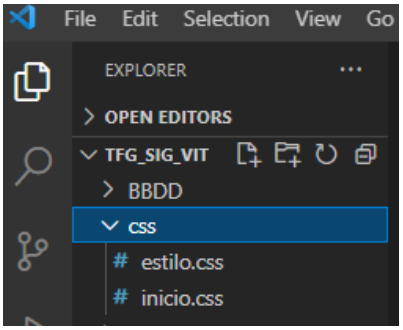


FIGURA 4.19 CARPETA CSS

Con CSS se pueden crear páginas web con un diseño que pueda resultar atractivo y agradable. Hay tres niveles de CSS, el nivel 1 ya no se emplea, el nivel 2 funciona como recomendación y el nivel 3 se divide en varios módulos, lenguaje estándar. Más información [21].

4.8.3 Modelo cliente-servidor

Las comunicaciones entre el cliente y el servidor se realizan a cabo a través del esquema cliente-servidor, en el cual, el cliente realiza peticiones al servidor, y este le responde. Ver **figura 4.20**. Para ello el cliente debe saber la dirección IP y el puerto, al cual realiza las peticiones. Ver **apartado de implementación 5**, concretamente en el la **figura 5.16** para observar la petición http y la respuesta del servidor.

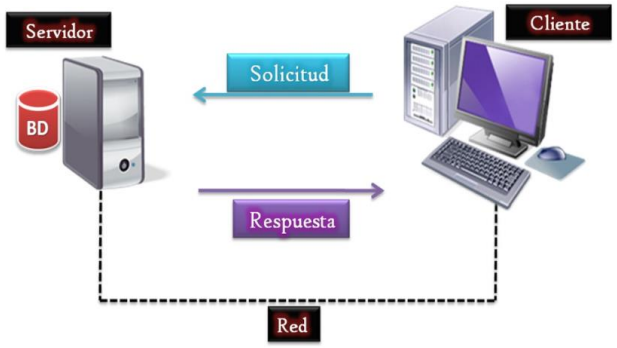


FIGURA 4.20 CLIENTE-SERVIDOR CON BASE DE DATOS.

4.8.3.1 HTTP

El protocolo HTTP (HyperText Transfer Protocol), se ha elegido para transportar las peticiones y respuestas, el cual pertenece al nivel de aplicación del modelo OSI. Ver **figura 4.21**.

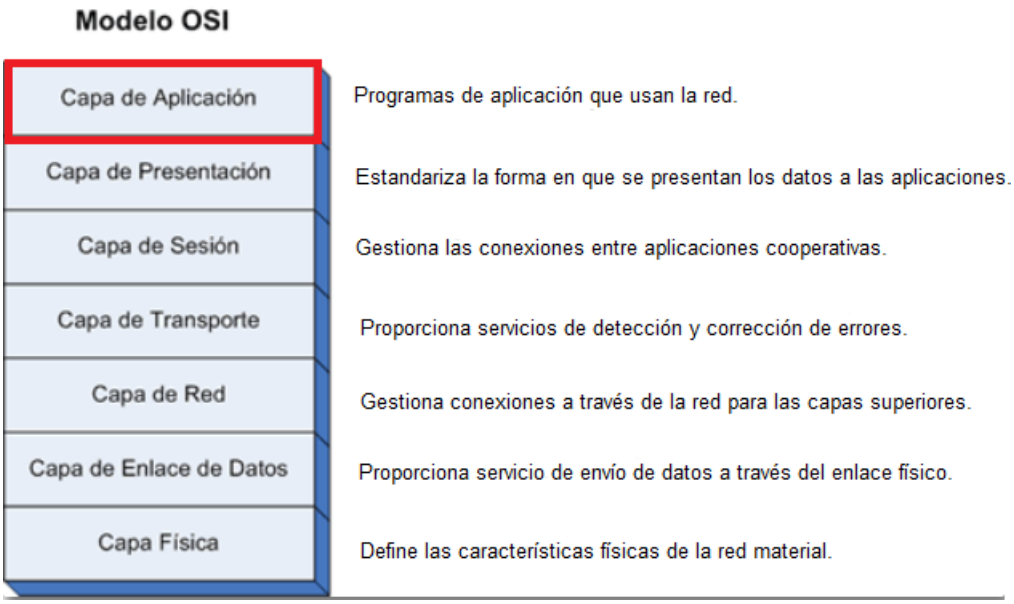


FIGURA 4.21 MODELO OSI

Este protocolo formado por una serie de cabeceras, nos permiten definir parámetros, tanto de la petición como de la respuesta, así como añadir información adicional. Posteriormente, se añaden los datos requeridos en el “Request Body” o Cuerpo del mensaje. Ver **figura 4.22**.

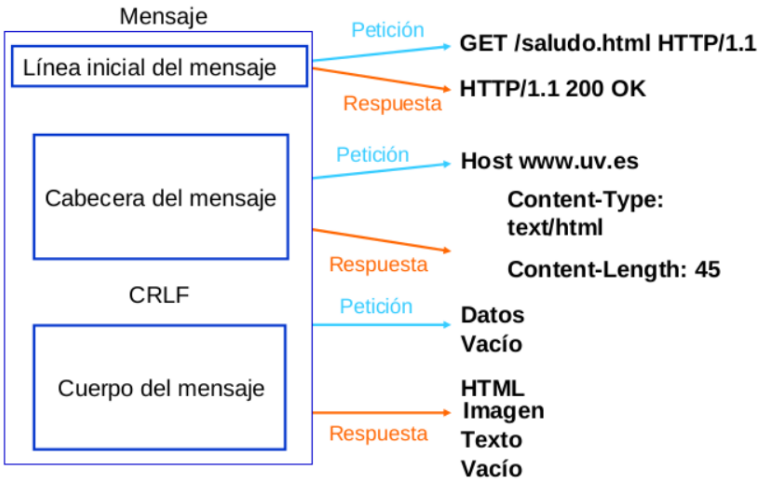


FIGURA 4.22 FUNCIONAMIENTO DEL PROTOCOLO HTTP

Depende de la petición, se utiliza distintos métodos, los más utilizados son:

- **POST:** Manda los datos en el cuerpo del mensaje.
- **GET:** Solicita un documento al servidor. Se envía datos en la URL.
- **DELETE:** Elimina el documento referenciado en la URL.
- **HEAD:** Similar a GET, pero sólo pide las cabeceras de HTTP.

Las respuestas, aparte de la serie de cabeceras, se definen por la primera línea, que nos da un código de estado, el cual, nos indica la existencia de errores o no.

Información del código	*Código de estado*
Respuestas informativas	100-199
Respuestas satisfactorias	200-299
Redirecciones	300-399
Errores de los clientes	400-499
Errores de los servidores	500-599

Tabla 4.1 Código de estado

Más información [22].

4.8.4 Librerías de representación gráfica

Una de las funciones de la aplicación web es la representación de los datos del paciente, a través de tablas o gráficos. Para tal finalidad, se ha decidido utilizar una librería que se pudiese ejecutar de forma atractiva e interactiva.

4.8.4.1 Highcharts

Librería basada en JavaScript destinada a mejorar las aplicaciones web mediante la adición de la capacidad de gráficos interactivos. Proporciona una amplia variedad de gráficos. Por ejemplo, gráficos de líneas, gráficos de barras, gráficos de área, etc. Más información [23].

4.8.4.2 Charts.js

Destacada por ser gratuita y por estar diseñada específicamente para HTML5. Además, ofrece ocho tipos distintos de gráficos configurables: barra, línea, área, circular, etc. Más información [24].

4.8.4.3 Elección

Para la representación de las gráficas se ha optado por la librería Highcharts, ya que es de código abierto y de uso gratuito. Además, los gráficos son interactivos, por lo cual, hace entretenida para el usuario la aplicación web.

4.9 Sistemas borrosos basados en reglas

Los sistemas borrosos basados en reglas o FRBS (Fuzzy Rule-Based Systems), utilizan la lógica difusa como herramienta para representar diferentes formas de conocimiento sobre el problema en cuestión, así como para modelar interacciones y relaciones que existen entre sus variables.

Hay dos tipos de sistemas borrosos, Mamdani y TSK, para el desarrollo de este proyecto se ha optado el sistema borroso Mamdani ya que es flexible y sencillo, para formular las normas mediante el lenguaje natural. Se propone un método llamado “Sistema borroso basado en reglas Mamdani” que nos permite plasmar los datos de los sensores con incertidumbre en un algoritmo, cuya salida sea el nivel de alerta del estado del paciente.

4.9.1 Sistema borroso basado en reglas Mamdani

Con el sistema borroso basado en reglas, se busca plasmar el pensamiento humano mediante unos conjuntos de reglas léxicas formadas por el par. “Si”, “Entonces”. Por ejemplo: Si “x” es grande e “y” es alto entonces “z” es bueno. El primer término, indica las condiciones necesarias para que se produzca una acción o hecho, determinada por el segundo término.

Indicar que tanto las condiciones de entrada y de salida, son variables, a las cuales se les asigna un subconjunto según su valor, pero estos subconjuntos no están definidos con un grado de pertenencia binario, en el cual un valor puede pertenecer o no a un subconjunto, sino que se aplica una clasificación progresiva, entre 0 y 1, siendo 0 totalmente externo y 1 totalmente perteneciente. Por lo tanto, un valor puede pertenecer a dos subconjuntos con distintos grados de pertenencia. Se puede utilizar varios modelos geométricos como triángulos, trapezoide, etc., ver **figura 4.23**.

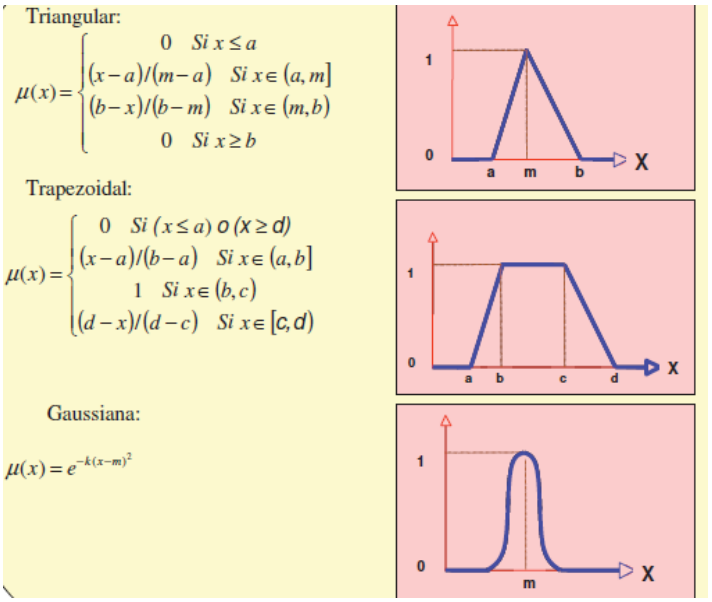


FIGURA 4.23 TIPOS DE FUNCIONES DE PERTENENCIA

El sistema es formado por los siguientes bloques, ver **figura 4.24**:

- **Entrada y salida:** La entrada es uno o más parámetros que se toman, en este caso las mediciones de los sensores. La salida, es un valor entre el [0, 1], sobre el nivel de alerta del estado de salud del paciente.
- **Normalizador:** En este módulo se realiza la transformación de los valores de entrada entre [0, 1].
- **Fuzzificador:** Se transforma un valor en una información borrosa, asignándole un grado de pertenencia a cada uno de los conjuntos borrosos definidos en

cada variable. El grado de pertenencia se define mediante una función característica asociada a cada conjunto borroso; de manera que para cada valor de “x” que pueda tomar una variable de entrada, la función característica “μA(x)”, proporciona el grado de pertenencia de este valor “x” al conjunto borroso “A”. Valor en el eje de ordenadas.

- **Base de conocimiento:** Elemento funcional subdividido en dos tipos de información, la base de datos que contiene la definición de los conjuntos borrosos y la base de reglas que contiene el conjunto de reglas.
- **Motor inferencia:** Con los conjuntos borrosos de las variables de entrada y la información de la base de reglas, se obtiene el un conjunto borroso para cada variable de salida.
- **Defuzzificador:** Elemento funcional que transforma los conjuntos borrosos inferidos, para cada variable de salida, en un valor puntual, que pueda ser aplicado al sistema o proceso a controlar. A partir de los valores borrosos de salida, extrae la salida real del sistema.
- **Desnormalizador:** Este módulo transforma los valores tomados por las variables de salida del controlador, que están comprendidos en el intervalo [0, 1], al rango de valores adecuado, para que dichas variables de salida, hagan actuar correctamente el sistema controlado.

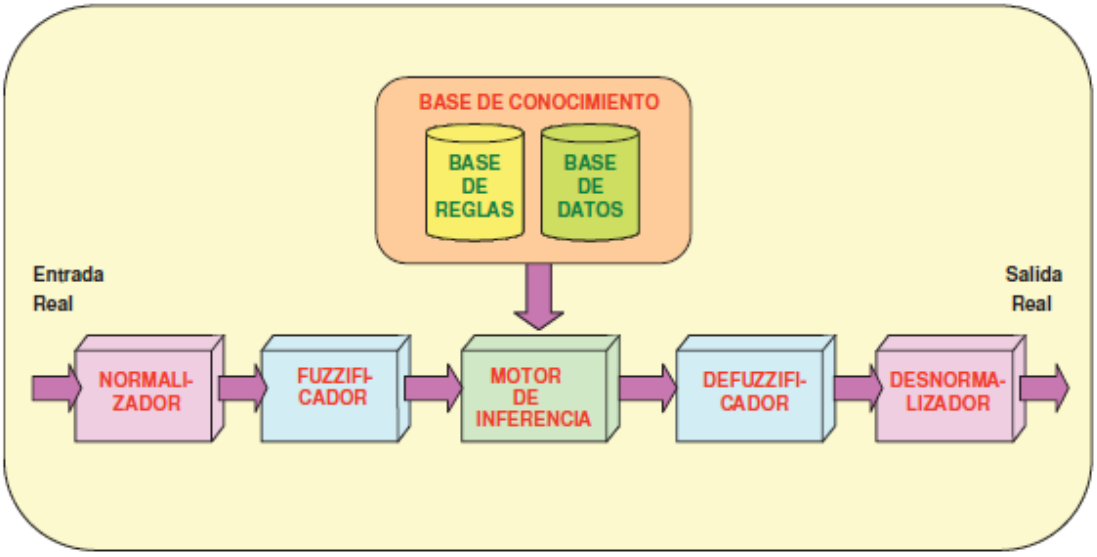


FIGURA 4.24 ESTRUCTURA GENÉRICA DE UN CONTROLADOR BORROSO

A continuación, en la **figura 4.25** se muestra un ejemplo del proceso del motor inferencia. Usa el modo B-FITA para calcular el mínimo valor de cada regla,

obteniendo el grado de pertenencia mínimo de los valores borrosos de entrada y finalmente se suman las salidas individuales, se obtiene el centro de gravedad como salida. En este diseño la salida será el nivel de alerta del paciente según los parámetros captados por los sensores [25].

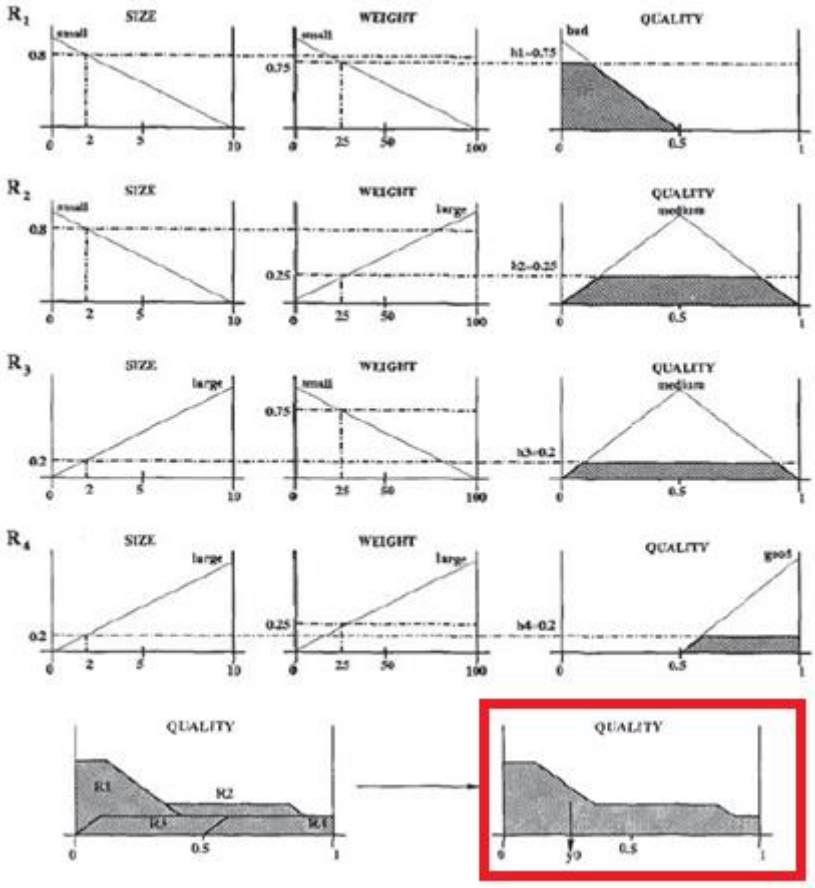


FIGURA 4.25 MODO DE FUNCIONAMIENTO DEL MOTOR INFERENCIA

CAPÍTULO 5. IMPLEMENTACIÓN

Este capítulo procede a mostrar/explicar los procedimientos y algunos códigos implementados junto a las tecnologías elegidas en el capítulo anterior.

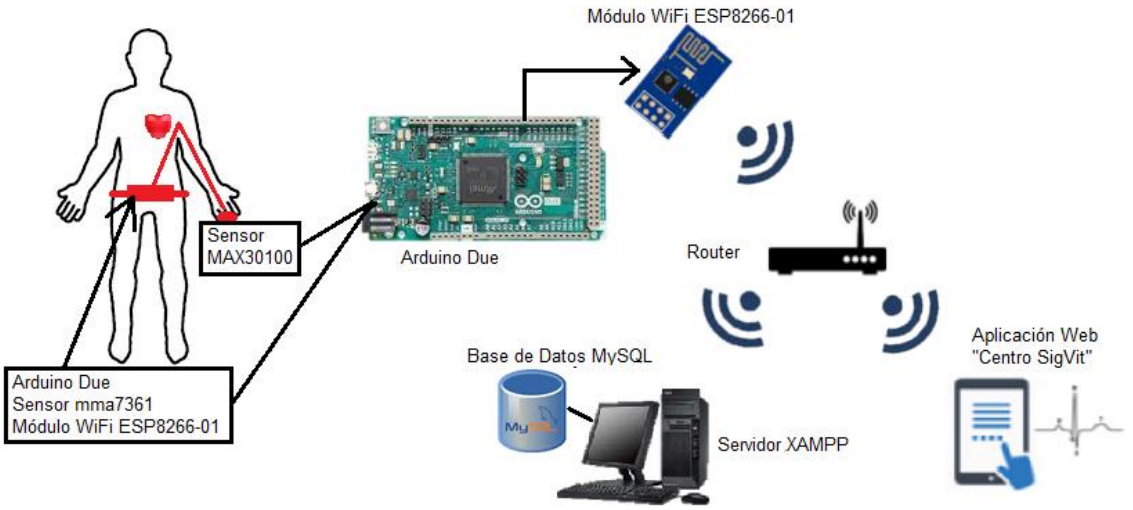


FIGURA 5.1 ESQUEMA GENERAL

Se observa en la **figura 5.1**, un esquema general de cómo queda la red de sensores implementada junto con el servidor XAMPP, la base de datos MySQL y la aplicación web denominada “Centro SigVit”. El paciente lleva implementado el sensor max30100 en el dedo índice, el Arduino Due junto con el sensor mma7361 y el módulo wifi ESP8266-01 en la cintura. Mediante el módulo wifi se conecta al servidor y a la base de datos, a través del router. Y por último desde cualquier dispositivo electrónico conectado a la red wifi del domicilio se puede visualizar la aplicación web “Centro SigVit”.

5.1 Programación en Arduino

Antes de empezar a programar se puede observar que al abrir el IDE de Arduino contiene dos funciones denominadas “void setup()” y “void loop()”, ver **figura 5.2**. Los comandos declarados en el “void setup()” sólo se ejecutarán una vez al inicio del sistema, por lo contrario, en el “void loop()” o bucle, es la función principal del programa. El lugar donde se tiene que poner los comandos que se ejecutan mientras la placa Arduino esté habilitada.

FIGURA 5.2 FUNCIONES ARDUINO

5.1.1 Sensores

Para obtener los distintos parámetros de los sensores elegidos en el **Capítulo 4**, se ha realizado los siguientes métodos.

5.1.1.1 Sensor max30100

Con este sensor se utiliza la librería “MAX30105.h”, (**figura 5.3**), permitiendo obtener los siguientes parámetros: temperatura, frecuencia cardiaca y oxígeno en sangre.

```
#include "MAX30105.h"
#include "spo2_algorithm.h"
#include "heartRate.h"

MAX30105 particleSensor;
```

FIGURA 5.3 LIBRERÍA DEL MAX30100

Una vez incluida la librería, se abre un ejemplo de código para probar su funcionamiento. Durante la prueba del sensor se encuentra un problema, el cual consiste en que el sensor no recoge los datos correctamente de los leds rojo e infrarrojo, no permitiendo el correcto funcionamiento. Tras observar el código del sensor varios días y visualizando los datos de manera gráfica (opción del IDE de Arduino, denominada “Serial plotter”), se detecta qué modificando las variables de

recogida de los leds el sensor comienza a dar los datos correctos, es decir, el led infrarrojo “getIR()” lo recoge como el led rojo “getRed()” y el led rojo como infrarrojo, ver **figura 5.4**.

```
redBuffer[i] = particleSensor.getIR(); //Intercambio
irBuffer[i] = particleSensor.getRed(); // Intercambio
```

FIGURA 5.4 INTERCAMBIO DE RECOGIDA DE DATOS MAX30100

Para obtener el parámetro del oxígeno en sangre (Spo2), se ejecuta un “buffer” que al inicio recoge cien valores del led infrarrojo y el rojo. Una vez iniciado, se elimina los veinticinco valores recogidos al inicio y guarda los veinticinco valores nuevos para calcular de nuevo el parámetro. Ver **figura 5.5**.

```
for (byte i = 75; i < 100; i++){
  while (particleSensor.available() == false){ //do we have new data?
    particleSensor.check(); //Check the sensor for new data
  }
  redBuffer[i] = particleSensor.getIR();// Intercambio
  irBuffer[i] = particleSensor.getRed();// Intercambio
  particleSensor.nextSample(); //We're finished with this sample so move to next sample
}
//After gathering 25 new samples recalculate HR and SP02
maxim_heart_rate_and_oxygen_saturation(irBuffer,bufferLength,redBuffer,&spo2,&validSPO2,&heartRate,&validHeartRate);
```

FIGURA 5.5 LOOP SPO2

A continuación, se explica cómo se obtiene el parámetro de frecuencia cardiaca (pulso), aunque en la **figura 5.5**, se puede observar que ya se obtiene en la variable “heartRate”. Hay un problema, el cual, no es fiable el valor de esa variable, ya que varía mucho, por lo que se hizo otro método para obtener el parámetro. Se recoge el valor del led rojo y va detectando los latidos del corazón durante 30 segundos, mientras tanto hay un contador, ese contador se multiplica por dos, para llegar al minuto de latidos y finalmente se obtiene el pulso. Ver **figura 5.6**.

```
Pini=millis();
while((Pactu<=(Pini+30000))){
  Pactu=millis();
  long irValue = particleSensor.getRed();
  if ( checkForBeat(irValue) == true ){
     contador++;
  }
}
pulso = contador*2;
```

FIGURA 5.6 CÓDIGO PULSO

Y el tercer parámetro a obtener es el de la temperatura, que se llama a la función “readTemperature()”. Ver **figura 5.7**.

```
double temp = particleSensor.readTemperature();
```

FIGURA 5.7 CÓDIGO TEMPERATURA

La conexión del sensor con el Arduino Due, se muestra en la **figura 5.5**.

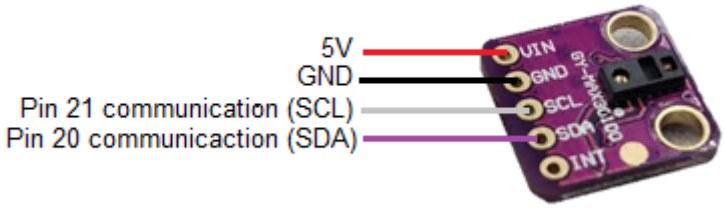


FIGURA 5.8 CONEXIÓN MAX30100 CON ARDUINO

5.1.1.2 Sensor mma7361

Con este sensor se utiliza la librería “#include <math.h>” para operaciones matemáticas básicas, y los parámetros que se obtienen son: posición, tiempo transcurrido en la misma posición, tiempo en otra posición, movimiento brusco, prevención de posible carilla (patología producida por la inmovilización del paciente durante un tiempo prolongado) en las posiciones cuando el paciente esté tumbado.

En la **figura 5.9**, recogen los valores de las variables x, y, z, con la función “analogRead()” y la función “map()” hace que los valores varíen entre valores positivos y negativos [26].

```
xVal = analogRead(0);
yVal = analogRead(1);
zVal = analogRead(2);
xVal = map(xVal, 0, 1023, 500, -500);
yVal = map(yVal, 0, 1023, 500, -500);
zVal = map(zVal, 0, 1023, 500, -500);
```

FIGURA 5.9 OBTENCIÓN VARIABLES X, Y, Z

Ahora hay que “clasificar” los valores de las variables para obtener las distintas posiciones/estados que puede optar una persona a lo largo del dia. En este diseño se ha implementado siete estados: De pie(P), tumbado lado izquierdo(I), tumbado lado derecho(D), acostado hacia arriba(A), acostado hacia abajo(B), con la cabeza hacia el suelo(C), no reconocido(N). Además, se puede detectar movimiento brusco(M) de la misma forma que se detecta el estado de la persona, con un “if{ } else{ }”. Si el estado anterior es distinto al estado de ese momento se llama a una función denominada “muestraTiempoOtraPisicion()” para que muestre el tiempo transcurrido en la anterior posición, pudiendo detectar la posible carilla, y si el estado es el mismo que el anterior se muestra el tiempo que lleva. Finalmente, para la prevención de carillas se recomienda cambiar de posición al paciente cada 3 horas, aunque en la **figura 5.10** está en 50 segundos.

```
estadoAnterior=estado;
if((-230<=xVal && xVal<=-170) and (-135<=yVal && yVal<=40) and (-80<=zVal && zVal<=190)){
  estado='P'; //De pie
  if(estado!=estadoAnterior){
    muestraTiempoOtraPosicion();
    Tinicio=millis(); //Vuelva a coger de nuevo el valor actual en ese momento
  }
  Tactual=millis();
  muestraTiempoTranscurrido();
  carillas='N';
}else if((-170<=xVal && xVal<=50) and (-280<=yVal && yVal<=-160) and (40<=zVal && zVal<=220)){
  estado='I'; //Tumbado lado izquierdo
  if(estado!=estadoAnterior){
    muestraTiempoOtraPosicion();
    Tinicio=millis(); //Vuelva a coger de nuevo el valor actual en ese momento
  }
  Tactual=millis();
  muestraTiempoTranscurrido();
  if(Ttrans> 50000){//50s -> 3h: 1,08e+7 ms
    carillas='S';
  }else{carillas='N';}
```

FIGURA 5.10 CÓDIGO MMA7361

La conexión del sensor con el Arduino Due, se muestra en la **figura 5.11**.

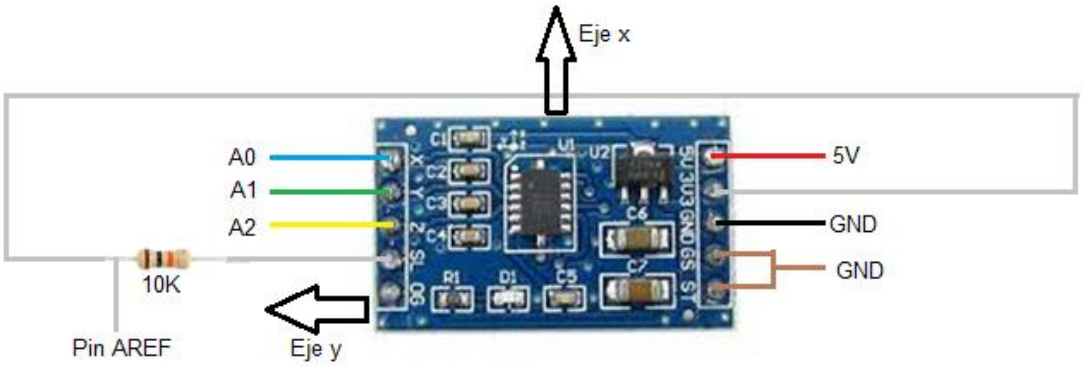


FIGURA 5.11 CONEXIÓN MMA7361 CON ARDUINO

5.1.2 Interfaz inalámbrica

La interfaz inalámbrica permite la comunicación sin cable y se puede transferir datos a otros equipos que estén conectados a la misma red.

5.1.2.1 Módulo wifi

El módulo Wifi ESP8266-01 se maneja mediante comandos AT. A continuación, se detalla qué comandos se utilizan para la conexión y envío de datos, ver **tabla 5.1**. Para más información sobre comandos AT [27].

Comando	Respuesta	Función
AT	OK	Prueba del correcto funcionamiento del módulo.
AT+CWMODE=1	OK	Establece el módulo en: 1=modo estación (cliente) 2=modo AP (huésped) 3=modo AP+estación (modo dual)
AT+CWJAP=\"nombre_red\",\"contraseña\"	OK	El módulo se conecta a la red con el nombre de la red y la contraseña
AT+CIPMUX=0	OK	Modo en: 0=conexión única 1=múltiples conexiones

AT+CIPSTART=\"tcp\",\"ip\",puerto	OK	Empieza una conexión como cliente, puede ser "TCP" o "UDP", luego se introduce la IP remota y el puerto remoto.
AT+CIPSEND=peticionHTTP.length()	SEND OK	Establece la longitud de datos a enviar.
		Espera a recibir "**>**" para enviar la petición http.
		Se envía la petición http, ver **figura 5.12**.
AT+CIPCLOSE	OK	Cierra la conexión "TCP" o "UDP"

Tabla 5.1 Comandos AT

Observando en la **figura 5.12**, se visualiza que primero se envía la longitud de la petición y se espera a recibir el símbolo, ">", para poder enviar el mensaje completo, es decir, la línea inicial del mensaje (método POST), las cabeceras del mensaje (Host, Accept, etc.), espacio en blanco (recuadro azul en la figura) y por último el cuerpo del mensaje (data). Además, el esquema del protocolo HTTP ha sido explicado en el **capítulo 4**, concretamente en la **figura 4.21**.

```
String data = tempe+"&"+heartRa+"&"+oxigenSan+"&"+fuz+"&"+estadAct+"&"+estadAnt+"
String peticionHTTP= "POST /TFG_Sig_Vit/insertar_Datos_Sensores.php HTTP/1.1\r\n";
peticionHTTP=peticionHTTP+"Host: 192.168.1.141\r\n";
peticionHTTP=peticionHTTP+"Accept: */*\r\n";
peticionHTTP=peticionHTTP+"Content-Type: application/x-www-form-urlencoded\r\n";
peticionHTTP=peticionHTTP+"Content-Length: "+data.length()+"\r\n\r\n";
peticionHTTP=peticionHTTP+data;
//Enviamos el tamaño en caracteres de la peticion http:
Serial3.print("AT+CIPSEND=");
Serial3.println(peticionHTTP.length());

//esperamos a ">" para enviar la petcion  http
if(Serial3.find(">")) // ">" indica que podemos enviar la peticion http
  Serial.println("Enviando HTTP . . .");
  Serial3.println(peticionHTTP);
  if(Serial3.find("SEND OK")){
    Serial.println("Peticion HTTP enviada:");
    Serial.println();
    Serial.println(peticionHTTP);
    Serial.println("Esperando respuesta...");
```

FIGURA 5.12 PETICIÓN HTTP ESP8266

La conexión del módulo Wifi con el Arduino Due, se muestra en la **figura 5.13**.

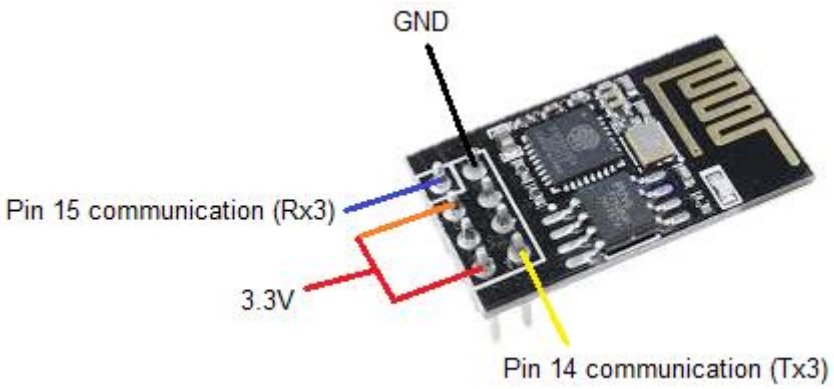


FIGURA 5.13 CONEXIÓN ESP8266-01 CON ARDUINO

Una vez realizada la conexión y conociendo los comandos AT, se comprueba manualmente el funcionamiento del módulo Wifi. El código para introducir los comandos manualmente se observa en la **figura 5.14** y la respuesta recibida del módulo junto a la conexión del servidor XAMPP, ver la **figura 5.15**.

```
void setup() {
  Serial.begin(9600);
  Serial3.begin(9600);
}

void loop() {
  if (Serial3.available()) {
    Serial.write(Serial3.read());
  }
  if (Serial.available()) {
    Serial3.write(Serial.read());
  }
}
```

FIGURA 5.14 CÓDIGO COMPROBACIÓN ESP8266-01

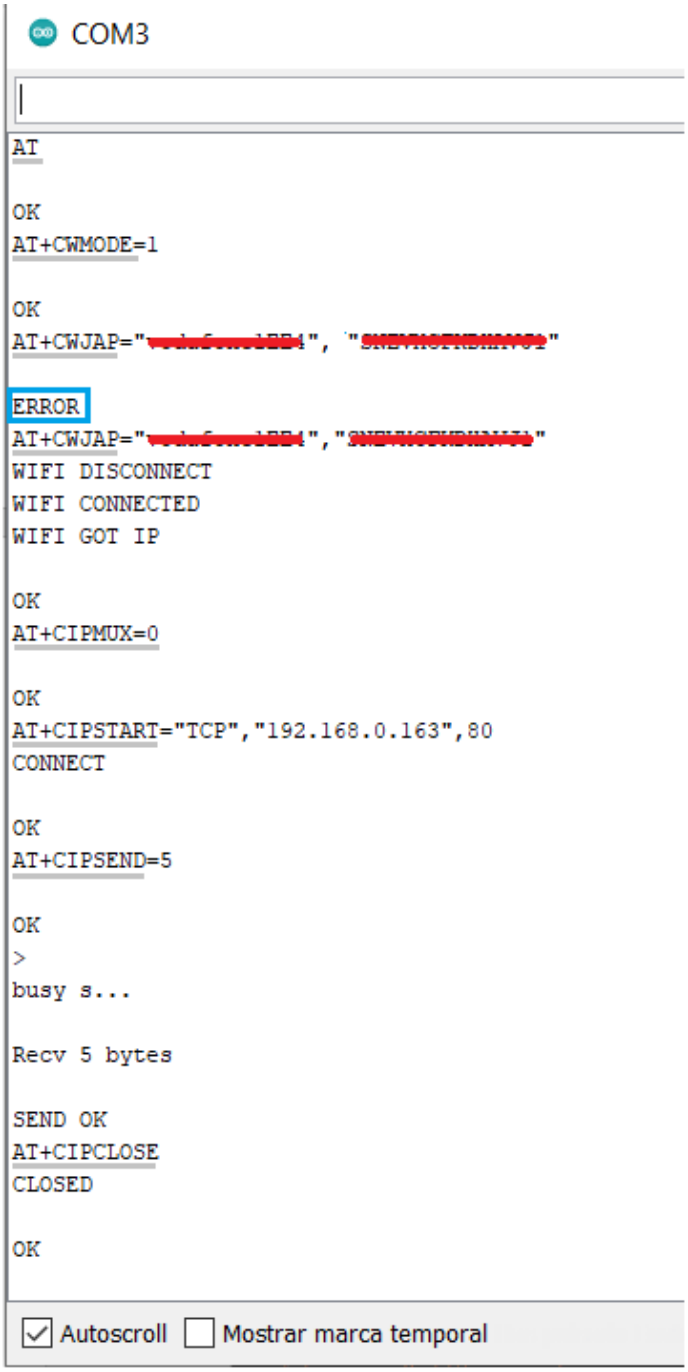


FIGURA 5.15 COMANDOS AT MANUALMENTE Y RESPUESTA MÓDULO WIFI

El recuadro azul de la **figura 5.15**, se debe a causa de no escribir bien el anterior comando AT, dejando un espacio en blanco entre el nombre de la red y su contraseña.

En la siguiente **figura 5.16**, si se conecta el Arduino Due con el cable usb al puerto del ordenador, se visualiza la petición http enviada junto con los datos de los sensores al servidor y la respuesta del servidor, en este caso, se recibe un 200 OK.

Mirando en la **tabla 4.1**, se sabe que es una respuesta satisfactoria, es decir, todo correcto.

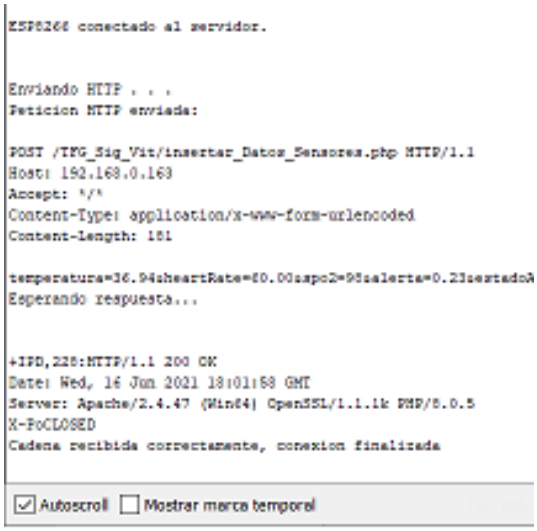


FIGURA 5.16 RESPUESTA HTTP

Se ha insertado un led rojo y otro verde, para visualizar el correcto funcionamiento del envío de los datos al servidor, sin necesidad de tener conectado el Arduino mediante el cable USB al ordenador. Se encienden durante un segundo y posteriormente se apagan. Cuando el módulo wifi no puede iniciar la comunicación con el servidor se enciende el led rojo, por el contrario, el led verde.

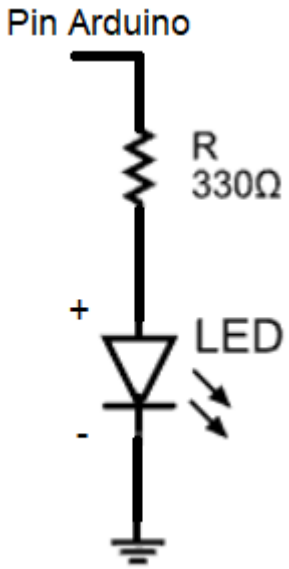


FIGURA 5.17 CONEXIÓN LED CON ARDUINO

5.1.3 Sistema borroso basado en reglas

Como se explica en el **apartado 4.9**, el algoritmo de nivel de alerta se basa en un sistema borroso basado en reglas, donde las variables de entrada consisten en los parámetros obtenidos a través de los sensores.

Para implementar el sistema borroso en el Arduino, se incluye la librería denominada “motorfuzzy.h”, ver **figura 5.18**.

```
#include <motorfuzzy.h>

motorfuzzy mot;
```

FIGURA 5.18 INCLUIR LIBRERÍA

Los parámetros de entrada son: temperatura, frecuencia cardiaca (pulso) y oxígeno en sangre (spo2), ver **figura 5.19**. Y la variable de salida es el nivel de alerta del paciente, según los valores de entrada.

```
double fuzzy=mot.MotorInfiere(temp,pulso,spo2);
```

FIGURA 5.19 FUZZY EN ARDUINO

5.1.3.1 Conjuntos borrosos

Los conjuntos borrosos se definen con el modelo geométrico trapezoide tanto para las entradas como para la salida y así indicar el grado de pertenencia de los valores anteriormente normalizados en [0, 1].

Se observa en la **figura 5.20**, el trapezoide definido por cuatro puntos: a, b, c y d. Los puntos a, d, son pertenencia cero y los puntos b, c, son pertenencia uno.

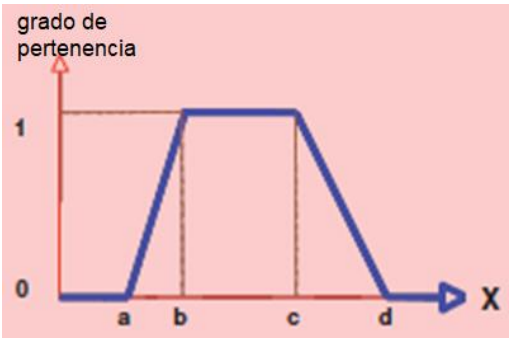


FIGURA 5.20 TRAPEZOIDE

5.1.3.1.1 Temperatura

La temperatura corporal es el equilibrio entre el calor producido y el que pierde el organismo, por lo cual, la temperatura corporal normal para un adulto sano varía entre 36.5ºC y 37,2ºC, valores superiores a lo normal es considerado fiebre y valores inferiores causan hipotermia, hace que el organismo se vuelva lento, por disminución del flujo sanguíneo, del suministro de oxígeno en los tejidos, y a su vez disminuye el metabolismo de los alimentos. Para el sistema borroso el rango de valor de la temperatura corporal es de 34ºC a 39ºC.

5.1.3.1.2 Pulso

La frecuencia cardiaca es la cantidad de latidos por unidad de tiempo, se basa en el número de contracciones de los ventrículos. La frecuencia cardiaca puede ser demasiada rápida (taquicardia) o demasiado lenta (bradicardia). El pulso es un abultamiento de una arteria por ondas de sangre que recorren los vasos sanguíneos cada vez que late el corazón. El pulso se suele tomar en la muñeca para estimar la frecuencia cardiaca. Para el sistema borroso el rango de valor del pulso es de 40 a 120 BPM.

5.1.3.1.3 Spo2

Se conoce como saturación de oxígeno a la medición de cantidad de oxígeno en sangre. Esta medición se lleva a cabo con un oxímetro que mide los niveles de oxígeno que transporta la sangre, a través de la hemoglobina. En personas sanas, y que viven a nivel del mar, la saturación de oxígeno será del 95-100%. Para el sistema borroso el rango de valor del oxígeno en sangre es de 81 a 101%.

5.1.3.1.4 Salida

La salida es el nivel de alerta comprendido entre [0, 1], en el que se encuentra en ese momento el paciente, se divide los posibles valores en cinco: nivel bajo, nivel medio-bajo, nivel medio, nivel medio-alto, nivel alto. La salida sufre un proceso de desnormalización para poder ser usado por nuestro sistema. Se procederá al mismo elemento geométrico, al trapezoide, para definir los conjuntos borrosos.

En el conjunto borroso nivel bajo: a=0.00, b=0.01, c=0.1, d=0.16.

En el conjunto borroso nivel medio-bajo: a=0.14, b=0.2, c=0.3, d=0.36.

En el conjunto borroso nivel medio: a=0.34, b=0.4, c=0.6, d=0.66.

En el conjunto borroso nivel medio-alto: a=0.64, b=0.7, c=0.8, d=0.86.

En el conjunto borroso nivel alto: a=0.84, b=0.89, c=0.99, d=1.00.

5.1.3.2 Base de reglas

Para completar la base de conocimientos, ver **figura 4.24**, se tiene que definir las reglas para completar el funcionamiento de nuestro sistema borroso. Estas reglas deben definir para todas las posibles combinaciones de entradas, su salida. Para este diseño se han realizado dieciocho reglas.

Conjunto de reglas:

Regla 1: Si la temperatura es baja, la frecuencia cardíaca es baja y el oxígeno en sangre es bajo entonces el nivel de alerta es alta.

Regla 2: Si la temperatura es baja, la frecuencia cardíaca es normal y el oxígeno en sangre es bajo entonces el nivel de alerta es medio-alta.

Regla 3: Si la temperatura es baja, la frecuencia cardíaca es alta y el oxígeno en sangre es bajo entonces el nivel de alerta es alta.

Regla 4: Si la temperatura es alta, la frecuencia cardíaca es baja y el oxígeno en sangre es bajo entonces el nivel de alerta es alta.

Regla 5: Si la temperatura es alta, la frecuencia cardíaca es normal y el oxígeno en sangre es bajo entonces el nivel de alerta es medio-alta.

Regla 6: Si la temperatura es alta, la frecuencia cardíaca es alta y el oxígeno en sangre es bajo entonces el nivel de alerta es alta.

Regla 7: Si la temperatura es normal, la frecuencia cardíaca es baja y el oxígeno en sangre es bajo entonces el nivel de alerta es medio-alta.

Regla 8: Si la temperatura es normal, la frecuencia cardíaca es normal y el oxígeno en sangre es bajo entonces el nivel de alerta es medio.

Regla 9: Si la temperatura es normal, la frecuencia cardíaca es alto y el oxígeno en sangre es bajo entonces el nivel de alerta es medio-alta.

Regla 10: Si la temperatura es baja, la frecuencia cardíaca es baja y el oxígeno en sangre es normal entonces el nivel de alerta es medio-alta.

Regla 11: Si la temperatura es baja, la frecuencia cardíaca es normal y el oxígeno en sangre es normal entonces el nivel de alerta es bajo.

Regla 12: Si la temperatura es baja, la frecuencia cardíaca es alta y el oxígeno en sangre es normal entonces el nivel de alerta es medio-alta.

Regla 13: Si la temperatura es alta, la frecuencia cardíaca es bajo y el oxígeno en sangre es normal entonces el nivel de alerta es medio.

Regla 14: Si la temperatura es alta, la frecuencia cardíaca es normal y el oxígeno en sangre es normal entonces el nivel de alerta es baja.

Regla 15: Si la temperatura es alta, la frecuencia cardíaca es alta y el oxígeno en sangre es normal entonces el nivel de alerta es medio.

Regla 16: Si la temperatura es normal, la frecuencia cardíaca es baja y el oxígeno en sangre es normal entonces el nivel de alerta es medio-bajo.

Regla 17: Si la temperatura es normal, la frecuencia cardíaca es normal y el oxígeno en sangre es normal entonces el nivel de alerta es bajo.

Regla 18: Si la temperatura es normal, la frecuencia cardíaca es alta y el oxígeno en sangre es normal entonces el nivel de alerta es medio-bajo.

5.1.3.3 Librería motorfuzzy

Para facilitar el uso del motor borroso, se implementa la librería (**figura 5.17**), que efectúa todas las operaciones, y devuelve el valor de salida (en este diseño, el nivel de alerta de salud del paciente). La librería está formada por dos archivos denominados: “motorfuzzy.h” y “motorfuzzy.cpp”.

- **Motorfuzzy.h:** Se definen los objetos y los métodos a utilizar. Existen dos objetos, el principal que es la clase motorfuzzy (realiza los cálculos) y la clase peso (objeto auxiliar que almacena los datos parciales, además del total de los cálculos).

- **Motorfuzzy.cpp:** Se implementan los métodos. El principal es “motorInfiere()”, a partir de los datos de los sensores y de las reglas, calcula la salida. Y los otros dos métodos se denominan “pertenencia()” y “calculaPeso()”.

-**Pertenencia(X, a, b, c, d)→** Es una función que calcula la pertenencia (eje y) del grupo borroso al valor X que es definido por los puntos a, b, c, d. Ecuaciones utilizadas, **figura 5.21**.

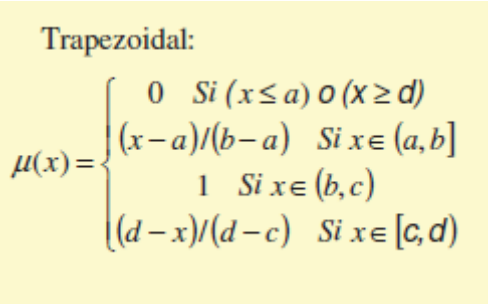


FIGURA 5.21 ECUACIÓN PERTENENCIA

-**CalculaPeso(altura, a, b, c, d)→**Calcula el área y centro del conjunto borroso con forma de trapecio junto al valor de altura, que es el valor mínimo de pertenencia del conjunto borroso y las variables de entrada. Ver **figura 5.22**.

```
double r2=(double)2.0/(double)3.0; //Ecuación
//Geometría triángulo
double centro,area,h,c1,a1,c2,a2,c3,a3;
peso result = peso();
h=altura;
centro=0.0; area=0.0;
//Centros
c1=a+(b-a)*r2; //Centro del triángulo izquierdo al dividir el trape
c2=b+(c-b)/2;//Centro del rectángulo al dividir el trapecio
c3=c+(1-r2)*(d-c);//Centro del triángulo derecho, al dividir el tra
//Áreas
a1=(b-a)*h/2; //Área del triángulo izquierdo al dividir el trapecio
a2=(c-b)*h; //Área del rectángulo central al dividir el trapecio
a3=(d-c)*h/2; //Área del triángulo derecho al divir el trapecio

area=a1+a2+a3; //Área total->Suma de las áreas
```

FIGURA 5.22 ECUACIÓN CALCULAPESO()

-**MotorInfiere(temp, heartRate, oxi)→** Contiene la base de datos con las reglas, indica los puntos de los conjuntos borrosos. Primero se normalizan las variables de entrada que son: temperatura, pulso y oxígeno en sangre. Luego se extrae la regla de la base de datos para cada entrada y se obtiene el grado de pertenencia al conjunto borroso para guardar el mínimo valor, llamando a la función “Pertenencia()”. Finalmente se llama a la función “CalculaPeso()” para

obtener el centro de la suma de todos los conjuntos borrosos. Ver **figura 4.25** para el esquema, y para el código ver **figura 5.23**.

```
/Para cada regla que coincide con número de filas(la longitud) de la base
for (iregla=0; iregla < num_reglas; iregla++ ) {
  grado[iregla] = 1.0; //Inicialmente es 1, después se mira cada antecede
  for (i=0;i<((num_puntos-4)/4);i++){ //En cada fila , 4 puntos por conju
                                      //son los del conjunto borroso de s
                                      // y despues se divide entre 4 para
    iC=i*4; //Valor inicial (punto inicial: "v1") del conjunto borroso qu
    // se extrae los puntos que definen el conjunto borroso
    v1 = BDm[iregla][iC];
    v2 = BDm[iregla][iC+1];
    v3 = BDm[iregla][iC+2];
    v4 = BDm[iregla][iC+3];
    //Se extrae el grado de pertenencia
    if(i==0){
      grado_aux = Pertenencia(tempe, v1, v2, v3, v4);
    } else if(i==1){
      grado_aux = Pertenencia(heartRa, v1, v2, v3, v4);
    }  else if (i==2){
      grado_aux = Pertenencia(oxige, v1, v2, v3, v4);
    }
    //Se comprueba si es menor al que tenía, si es así, es nuevo grado de
    if ( grado_aux < grado[iregla] ){
      grado[iregla] = grado_aux;
    }
  }
  //Ahora se extrae los puntos que definen el conjunto borroso de salida,
  //que serán los últimos 4 puntos de cada fila
  iC= num_puntos-4;
  v1 = BDm[iregla][iC];
  v2 = BDm[iregla][iC+1];
  v3 = BDm[iregla][iC+2];
  v4 = BDm[iregla][iC+3];
  //Se calcula el peso de la regla en función del grado mínimo de pertene
  resultado=CalculaPeso ( grado[iregla],v1, v2, v3, v4);
  //Se va sumando el producto del centro de gravedad por la masa (área) c
  suma_centro_por_area = suma_centro_por_area + resultado.centro*resultac
  //Se va sumando la masa (área) de cada conjunto borroso
  sumaarea=sumaarea + resultado.masa;
}
```

FIGURA 5.23 MOTORINFIERE()

Por último, una vez se obtiene el nivel de alerta “fuzzy” comprendido entre [0, 1] en el código del Arduino, se subdivide en rangos de 0.2 y según sea el nivel de alerta poder obtener los datos de los sensores en un determinado tiempo. A continuación, se describe:

- Para “fuzzy” comprendido entre mayor que 0 y menor que 0.2, el nivel de alerta es bajo, por lo que la próxima recogida de datos será dentro de 10 minutos.

- Para “fuzzy” comprendido entre mayor o igual que 0.2 y menor que 0.4, el nivel de alerta es medio bajo, por lo que la próxima recogida de datos será dentro de 8 minutos.
- Para “fuzzy” comprendido entre mayor o igual que 0.4 y menor que 0.6, el nivel de alerta es media, por lo que la próxima recogida de datos será dentro de 6 minutos.
- Para “fuzzy” comprendido entre mayor que 0.6 y menor que 0.8, el nivel de alerta es medio alta, por lo que la próxima recogida de datos será dentro de 4 minutos.
- Para “fuzzy” comprendido entre mayor o igual que 0.8 y menor que 1, el nivel de alerta es alta, por lo que la próxima recogida de datos será dentro de 10 minutos.
- Para “fuzzy” igual que 0.00, el nivel de alerta no se reconoce debido a que los datos de los sensores no son buenos, por lo que la próxima recogida de datos será en el mismo momento.

5.2 Servidor

En este apartado se trata del desarrollo de las distintas clases, vistas y estilos que se han utilizado en el servidor XAMPP junto con la base de datos.

La carpeta que contiene el código desarrollado de la página web, denominada “TFG_Sig_Vit”, el cuál debe de estar en el directorio de XAMPP, dentro de la carpeta “htdocs”. Ver **figura 5.24**.

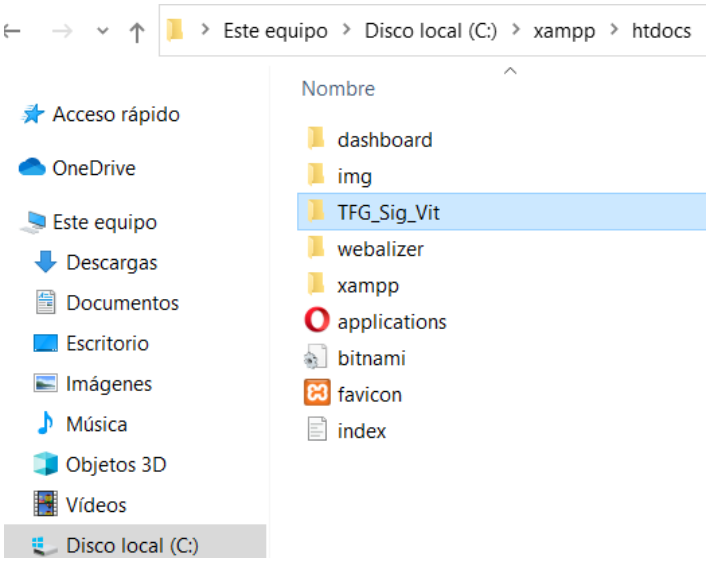


FIGURA 5.24 DIRECTORIO SERVIDOR

Una vez que se ha creado la carpeta dentro del servidor, se procede a desarrollar el código, en Visual Studio Code. En la **figura 5.25**, se puede observar la estructura, que contiene cuatro carpetas:

- **BBDD:** Contiene un fichero .txt, con el código para crear la base de datos.
- **Css:** Contiene dos ficheros .css, uno de ellos es el estilo de la página principal y el otro para el estilo de la página web.
- **Img:** Contiene las imágenes de fondo de pantalla utilizadas en los ficheros css y el logo utilizado, llamado “icono.png”.
- **Plantillas:** Contiene tres ficheros .php, en los cuales, se estructuran la página web, en “encabezado.php”, “cuerpo.php” y “piePag.php”.

  -**Encabezado.php**→ Situado en la parte superior como su propio nombre indica. Contiene el nombre de la aplicación, denominada “Centro SigVit”, el logo del centro y un botón para salir de la aplicación.

  -**Cuerpo.php**→ Situado entre el encabezado y el pie de página. Contiene el menú de la aplicación, es decir, una vez identificado, se puede elegir qué visualizar.

  -**piePag.php**→ Situado en la parte inferior. Contiene el nombre del trabajo de fin de grado y quién lo ha realizado.

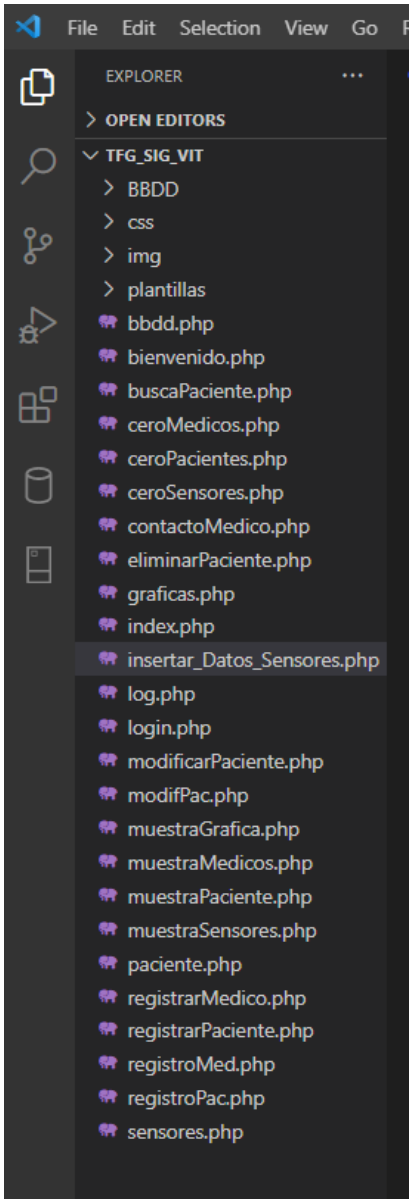


FIGURA 5.25 ESTRUCTURA DE LOS PHP

Hasta ahora sólo se han descrito las carpetas, por lo que a continuación, se describen todas las funciones:

- **Bbdd.php:** Realiza la conexión a la base de datos.
- **Bienvenido.php:** Contiene la página principal una vez identificado, en la cual se expone las funcionalidades de la aplicación.
- **buscaPaciente.php:** Secuencia “SQL”, en el cuál, selecciona el paciente buscado por su nombre y su identificador (“id”).
- **ceroMedicos.php:** Muestra un mensaje de que no existen médicos registrados.

- **ceroPacientes.php:** Muestra un mensaje de que no existen pacientes registrados y contiene un botón para agregar un paciente nuevo.
- **ceroSensores.php:** Muestra un mensaje de que no existen datos de los sensores para mostrar.
- **contactoMedico.php:** Secuencia “SQL”, en el cuál, selecciona los médicos registrados para mostrar todos sus datos.
- **eliminarPaciente.php:** Secuencia “SQL”, en el cuál, elimina el paciente seleccionado a través del identificador “id”.
- **graficas.php:** Hay que insertar el nombre del paciente para mostrar algunos de sus datos en una gráfica interactiva.
- **index.php:** Página principal en el que se debe de elegir si registrarse o identificase, en el caso de tener cuenta o no.
- **insertar_Datos_Sensore.php:** Secuencia “sql” para insertar todos los datos recibidos del módulo Wifi ESP8266 conectado al Arduino Due junto con los demás sensores.
- **log.php:** Secuencia “SQL” se comprueba si el usuario y contraseña introducido, están registrados en la base de datos para otorgarle al médico la identificación correcta, para poder entrar en la aplicación.
- **login.php:** Muestra el formulario en el cual se tiene que introducir el usuario y contraseña y contiene un botón para retroceder a la anterior página/vista.
- **modificarPaciente.php:** Muestra un formulario para modificar los datos del paciente seleccionado y contiene un botón para regresar a la vista anterior.
- **modifPac.php:** Secuencia “SQL”, en el cuál, se actualiza los datos modificados del paciente seleccionado.
- **muestraGrafica.php:** Muestra la gráfica con los datos del paciente que se ha seleccionado anteriormente.
- **muestraMedicos.php:** Muestra una tabla con los datos de los médicos registrados.
- **muestraPaciente.php:** Muestra una tabla con los datos de los pacientes registrados. Además, contiene un botón para agregar un nuevo paciente, modificar o eliminar el paciente seleccionado por su identificador “id”.
- **muestraSensores.php:** Muestra una tabla con los datos de los sensores de los pacientes.
- **paciente.php:** Secuencia “SQL”, en el cual, se selecciona todos los datos de los pacientes.

- **registrarMedico.php:** Muestra un formulario para registrar al médico, en caso de que no tenga una cuenta ya registrada. Además, contiene un botón para regresar a la vista anterior.
- **registrarPaciente.php:** Muestra un formulario para el registro de un paciente y contiene un botón para regresar a la vista anterior.
- **registroMed.php:** Secuencia “SQL” para insertar datos de un médico en la base de datos.
- **registroPac.php:** Secuencia “SQL” para insertar datos de un paciente en la base de datos.
- **sensores.php:** Secuencia “SQL” para seleccionar los datos de los sensores y ordenarlos en modo ascendente por la fecha y hora.

Una vez descritas las funciones que forman la página web, se muestra el diagrama de flujo en las **figuras 5.26** y **5.27**, para una mejor visualización. Cuando el médico se ha identificado, se muestra la vista “bienvenido”, además de esa vista, aparece en la parte izquierda un menú con cinco opciones, ver **figura 5.37** (más delante), que está estático en todo momento hasta que se cierre el navegador web o se pulse el botón de salir. La carpeta plantillas contiene tres ficheros que son los que dan forma de encabezado, cuerpo y pie de página a la página web, explicado más arriba.

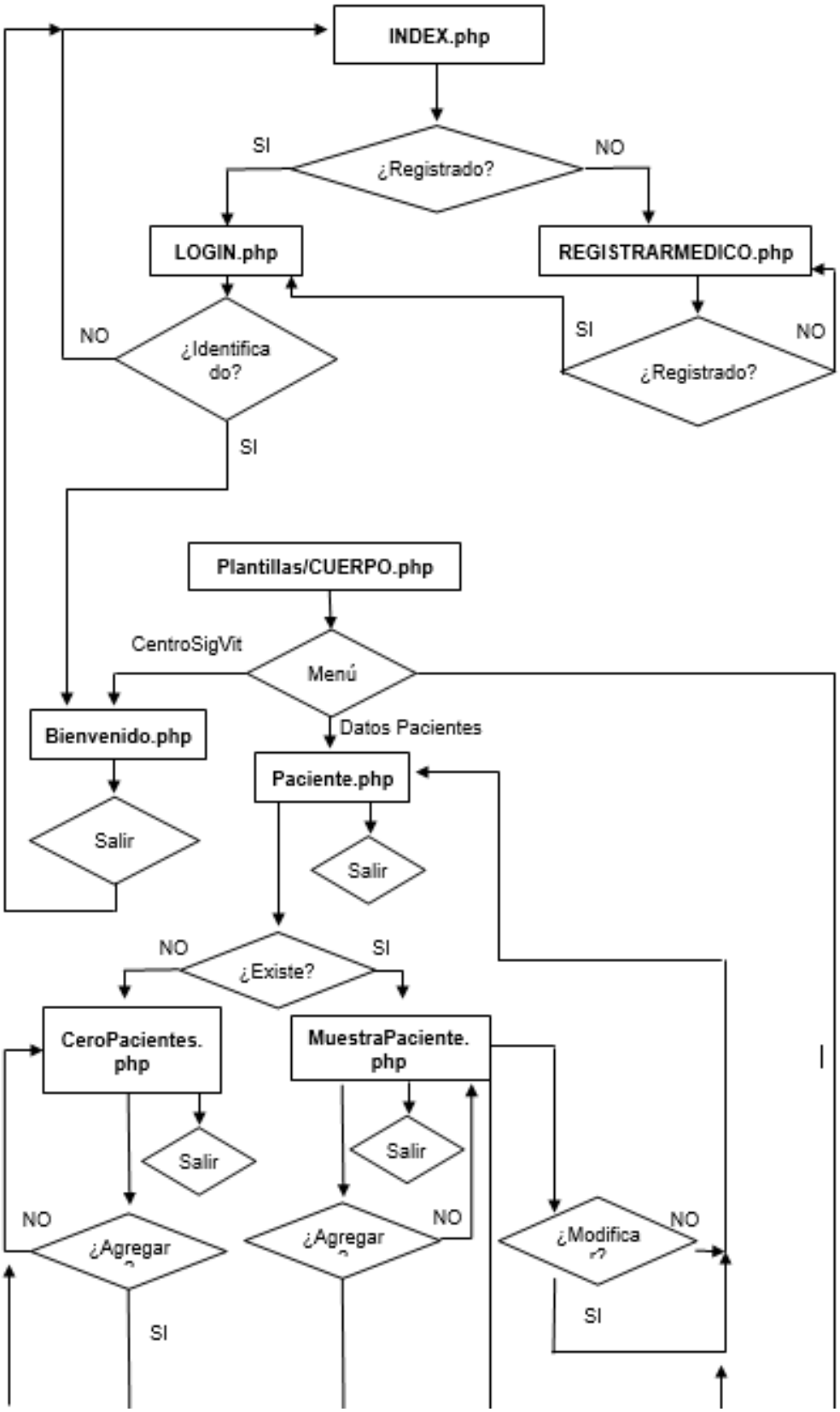


FIGURA 5.26 DIAGRAMA DE FLUJO DE LA PÁGINA WEB EN PHP -1

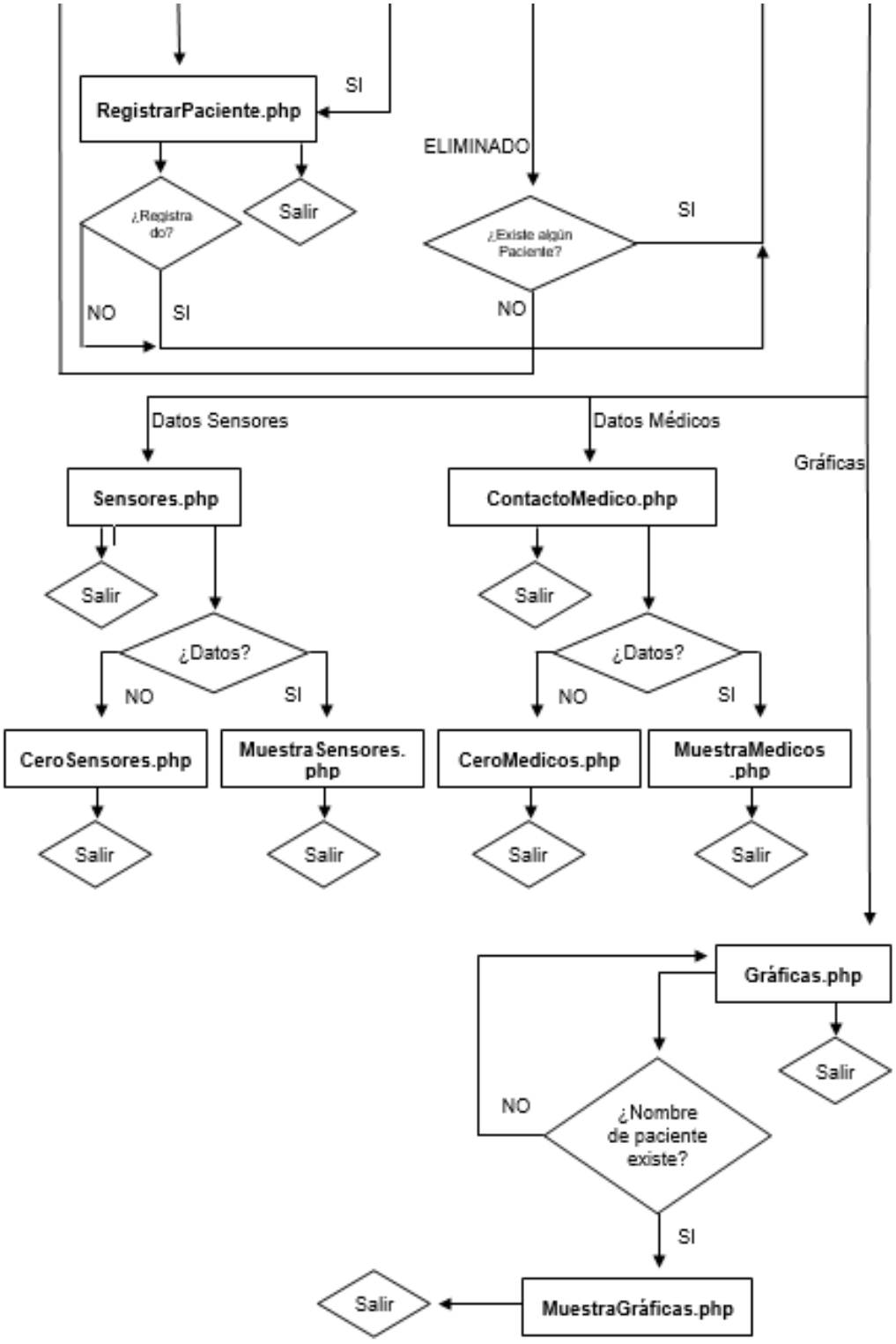


FIGURA 5.27 DIAGRAMA DE FLUJO DE LA PÁGINA WEB EN PHP -2

5.3 Base de datos

Implementada por el servidor XAMPP, por lo que en este apartado se trata de los aspectos relacionados con el diseño de la base de datos, denominada “bbdd_tfg”, como se puede observar en la **figura 5.28**. La base de datos la componen tres tablas, “médicos”, “pacientes”, “sensores”.

FIGURA 5.28 ESTRUCTURA BASE DE DATOS

La tabla “medicos” hace referencia a la información de los médicos registrados en la aplicación web, para llevar un control de los pacientes. Compuesta por seis campos, siendo clave primaria “consultaMed” y unique key “usuarioMed” como se observa en la **figura 5.29**.

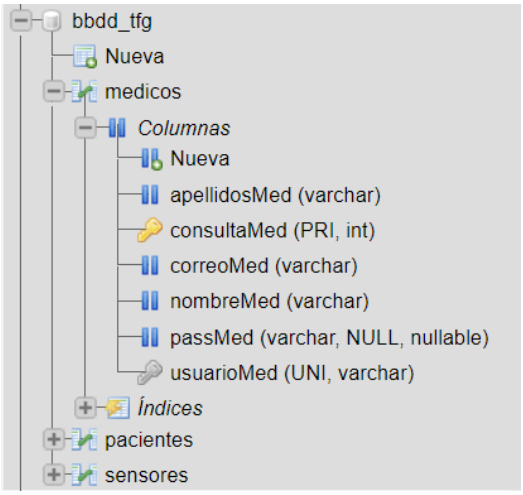


FIGURA 5.29 TABLA MEDICOS

La tabla “pacientes” hace referencia a la información personal del paciente y contiene siete campos, siendo la clave primaria “id”. Ver **figura 5.30**.

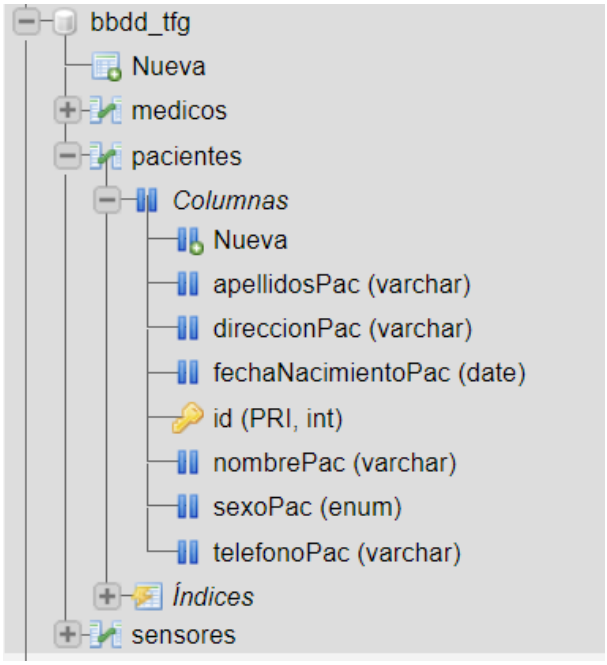


FIGURA 5.30 TABLA PACIENTES

La tabla “sensores” obtiene toda la información de los sensores del paciente, contiene doce campos, la clave primaria es “diaHora” y el campo “idPac” es clave foránea de “id” de la tabla “pacientes”. Ver **figura 5.31.**

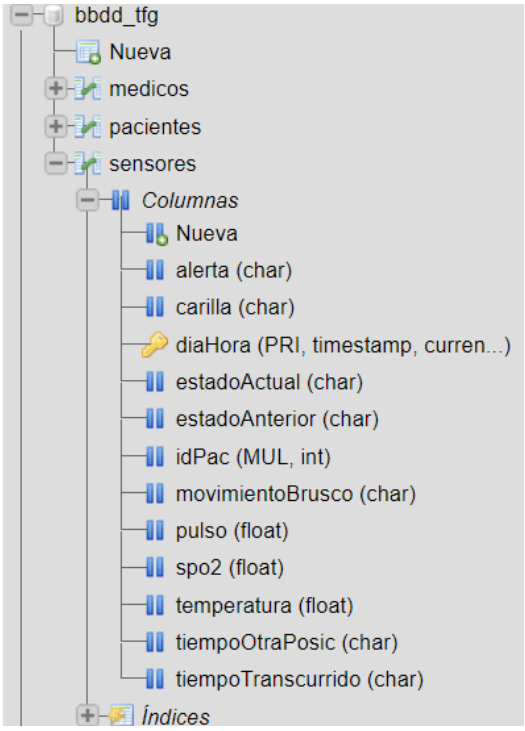


FIGURA 5.31 TABLA SENSORES

5.4 Aplicación Web

Para poder ver la página web, antes de nada hay que asegurarse de que el servidor está arrancado, continuadamente se abre un navegador web y a continuación, se escribe en la url la dirección. Hay dos formas de escribir la dirección, dependiendo desde donde estemos llamandolo:

- Si el navegador se abre desde el mismo dispositivo donde está el servidor Xampp arrancado se introduce “localhost/tfg_sig_vit/”.

- Si el navegador se abre de distinto dispositivo de donde se encuentra el servidor Xampp instalado, entonces, se debe de saber la IP de donde se encuentra el servidor. Para ello, se introduce “cmd” o “ símbolo de sistema” en el buscador de windows en el dispositivo donde está instalado el servidor, ver **figura 5.32,** y seguidamente introducir el comando “ipconfig” y se obtiene la “IP” conforme se observa en la **figura 5.33**. Una vez conocida la “IP” se introduce en el navegador web de cualquier otro dispositivo que no sea el que tiene instalado el servidor “192.168.1.141/tfg_sig_vit”.

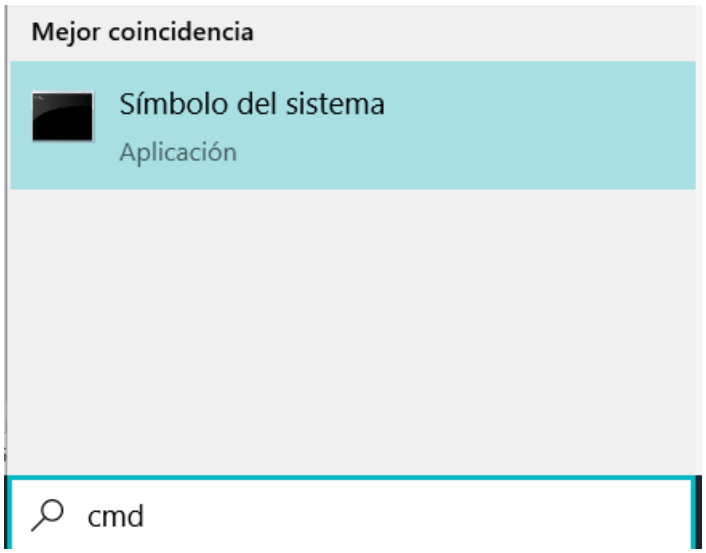


FIGURA 5.32 ABRIR SÍMBOLO DE SISTEMA

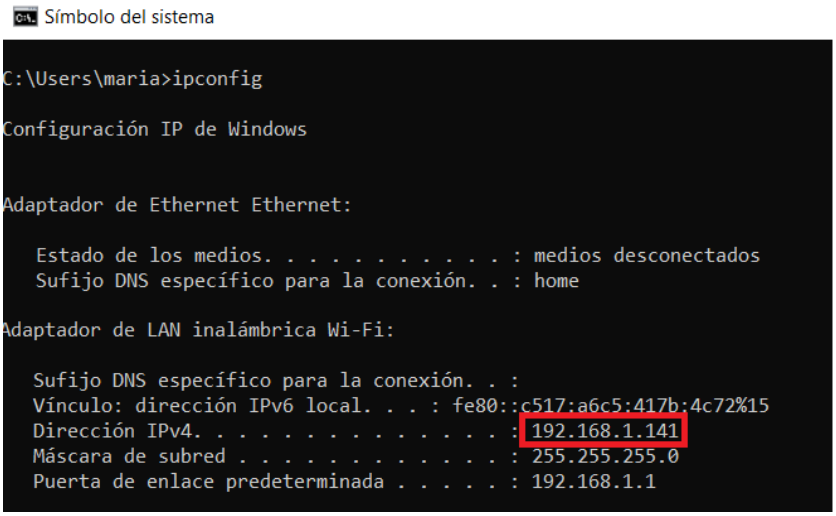


FIGURA 5.33 CMD PARA CONOCER LA IP DEL SERVIDOR

Seguidamente se muestra la página web, donde se tiene que identificar o registrarse para poder entrar a ver los datos de los pacientes. Ver **figura 5.34**.

FIGURA 5.34 INDEX.PHP

Si no está registrado tiene que rellenar los campos que se muestran en la **figura 5.35** y una vez registrado, sólo tiene que introducir su nombre de usuario y su contraseña, ver **figura 5.36**.

FIGURA 5.35 REGISTRO.PHP

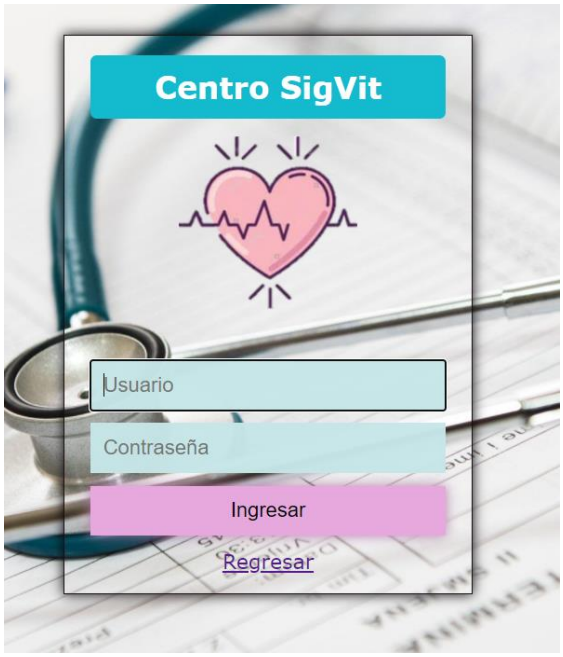


FIGURA 5.36 LOGIN.PHP

Acto seguido a la autenticación y los datos introducidos sean los correctos, se muestra la bienvenida de la aplicación denominada “Centro SigVit”. En ella se puede ver los datos de los pacientes, los datos de los sensores de cada paciente

identificandolos por el “id”, los datos de los médicos y por último, se muestra una gráfica con los datos del paciente. Ver **figura 5.36**.

FIGURA 5.37 BIENVENIDO.PHP

Para poder ver la gráfica, se tiene que introducir el nombre del paciente que esté registrado, ver **figura 5.38**.

FIGURA 5.38 GRAFICA PACIENTES

En la **figura 5.39**, el nombre introducido es “Pepe” ya que está registrado y la base de datos contiene datos.

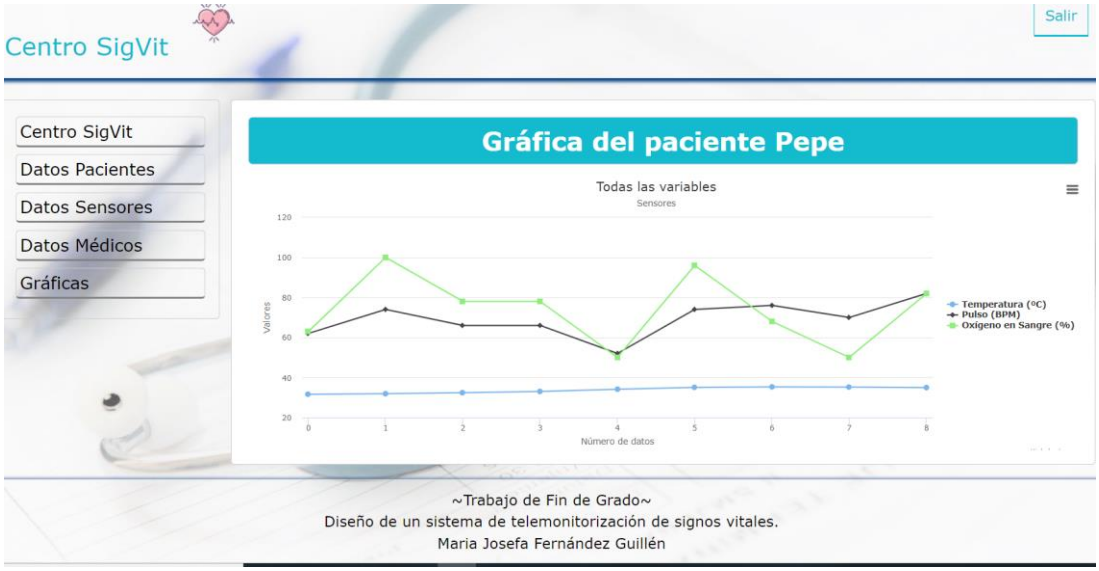


FIGURA 5.39 GRÁFICA DATOS

5.4 Implementación de sensores en el paciente

En este apartado se comenta la implementación de los sensores en el cuerpo del paciente, ya que según la posición implementada puede dar datos erróneos.

Primeramente, se define la colocación del sensor MAX30100 en el dedo índice con una banda elástica y que no tenga mucha presión. Por otro lado, el Arduino Due junto con el módulo Wifi ESP8266-01 se sujeta con una cinta/cinturón a la cintura. Y, por último, el más importante de colocar es el sensor MMA7361 ya que al contener tres ejes es importante para el correcto funcionamiento, donde el eje x tiene que estar en el eje vertical y el eje y en el horizontal, ver en la **figura 5.40**.

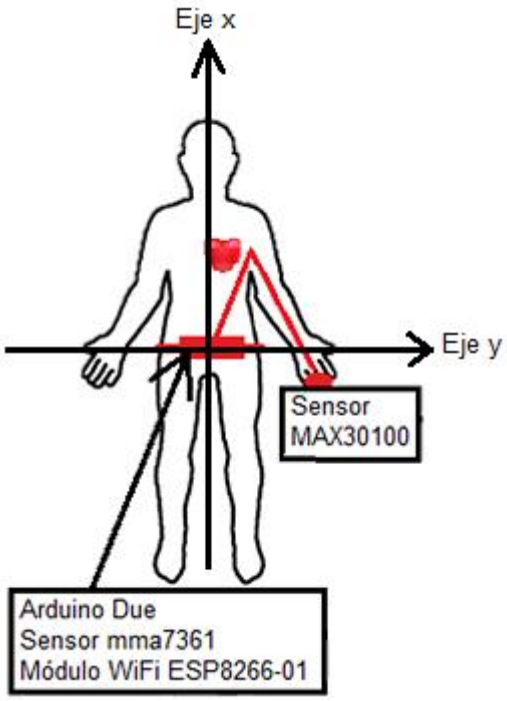


FIGURA 5.40 SENSORES COLOCADOS EN EL PACIENTE

CAPÍTULO 6. CONCLUSIONES

6.1 Conclusión

Una vez realizado el diseño y comprobado su correcto funcionamiento, se concluye que se ha realizado los objetivos propuestos con éxito.

- La red de sensores es una tecnología eficiente y útil.
- Sistema con elementos simples, baratos y funcionan a un buen nivel.
- Son tecnologías altamente usadas, por lo que es de gran facilidad desarrollar proyectos por la gran cantidad de documentación, aunque para la placa Arduino Due no hay tanta documentación, pero si es cierto que hay gran apoyo de la comunidad.
- Se implementa la técnica basada en sistema borroso basado en reglas, el cual, permite detectar situaciones de deterioro de la salud para una variabilidad de los datos de entrada.

CAPÍTULO 7. LÍNEAS DE FUTURO

7.1 Líneas de futuro

Cumplidos los objetivos que se habían planteado y viendo la viabilidad del proyecto, se plantean unas futuras mejoras y funcionalidades para el sistema:

- Gestionar el consumo de energía y abastecer el consumo del nodo y los sensores con pilas recargables.
- Calcular cada cuánto tiempo hay que recargar las pilas.

CAPÍTULO 8. REFERENCIAS BIBLIOGRÁFICAS

[1] TELEMONITORIZACIÓN. *Telemonitorización pacientes en su domicilio.* [Consulta: 28 de mayo 2021] Disponible en: https://canaldiabetes.com/la-telemonitorizacion/

[2] TIPOS DE PLACAS. *Ventajas Arduino.* [Consulta: 28 de mayo 2021] Disponible en: https://www.thegreenmonkey.es/barriodesalamanca/ventajas-de-arduino/#:~:text=Es%20de%20código%20abierto%2C%20por,plataformas%20informáticas%2C%20o%20casi%20todas.

[3] ARDUINO UNO. *Características.* [Consulta: 28 de mayo 2021] Disponible en: https://pluselectric.wordpress.com/2014/09/21/arduino-uno-especificaciones-y-caracteristicas/

[4] ARDUINO UNO. *Precio.* [Consulta: 3 de junio 2021] Disponible en: https://www.arduino.cc/search?tab=&q=arduino+uno

[5] ARDUINO DUE. *Características.* [Consulta: 28 de mayo 2021] Disponible en: https://arduino.cl/producto/arduino-due/

[6] ARDUINO DUE. *Precio.* [Consulta: 26 de noviembre 2020] Disponible en: https://www.amazon.es/gp/product/B00A6C3JN2/ref=ox_sc_act_title_2?smid=A1LKZYWRVOF5T2&psc=1

[7] ARDUINO NANO. *Características.* [Consulta: 2 de junio 2021] Disponible en: https://www.electrogeekshop.com/arduino-nano-pinout-y-caracteristicas/

[8] ARDUINO NANO. *Precio.* [Consulta: 2 de junio 2021] Disponible en: https://www.amazon.es/Arduino-A000005-ARDUINO-Nano/dp/B0097AU5OU

[9] Wifi y BLUETOOTH. *Características.* [Consulta: 3 de junio 2021] Disponible en: https://www.adslzone.net/reportajes/wifi/wifi-vs-bluetooth/

[10] MÓDULOS ESP8266. *Tipos de ESP8266.* [Consulta: 3 de junio 2021] Disponible en:https://programarfacil.com/podcast/esp8266-wifi-coste-arduino/#Que_es_el_ESP8266

[11] ESP8266-01. *Precio.* [Consulta: 3 de junio 2021] Disponible en: https://www.amazon.es/AZDelivery-Esp8266-ESP-01-Arduino-Raspberry/dp/B01LK83TX0/ref=sr_1_14?__mk_es_ES=ÅMÅŽÕÑ&dchild=1&keywords=esp8266-01&qid=1622707582&refinements=p_36%3A1323855031&rnid=1323854031&s=computers&sr=1-14

[12] SENSORES. *¿Qué es un sensor?* [Consulta: 2 de junio 2021] Disponible en: https://proyectosconarduino.com/sensores/

[13] GY-MAX30100. *Datasheet.* [Consulta: 2 de junio 2021] Disponible en: https://www.didacticaselectronicas.com/index.php/sensores/biomedicos/sensor-medidor-de-concentración-de-oxígeno-y-ritmo-cardiaco-max30100-sensores-de-

concentracion-de-oxigeno-y-ritmo-cardiaco-pulsioxímetros-pulsioximetro-pulsioximetría-pulsioximetria-gy-max3010-detail#:~:text=Sensor%20de%20concentración%20de%20oxígeno%20y%20ritmo%20cardiaco%20MAX30100.,detectar%20pulsioximetría%20y%20ritmo%20cardiaco.
[14] DS18B20. *Información.* [Consulta: 4 de junio 2021] Disponible en: https://tienda.bricogeek.com/sensores-temperatura/510-sensor-ds18b20-estanco.html#:~:text=El%20sensor%20DS18B20%20permite%20medir,a%20la%20distancia%20del%20cableado.
[15] MMA7361. *Información.* [Consulta: 4 de junio 2021] Disponible en: https://hetpro-store.com/TUTORIALES/mma7361-sensor-acelerometro/
[16] ADXL345. *Información.* [Consulta: 4 de junio 2021] Disponible en: https://moviltronics.com/tienda/acelerometro-adxl345/
[17] XAMPP. *Información.* [Consulta: 5 de junio 2021] Disponible en: https://www.12caracteristicas.com/xampp/
[18] BASE DE DATOS. *MySQL.* [Consulta: 5 de junio 2021] Disponible en: https://neoattack.com/neowiki/mysql/
[19] PHP. *Introducción.* [Consulta: 7 de junio 2021] Disponible en: https://www.php.net/manual/es/intro-whatis.php
[20] HTML. *Introducción.* [Consulta: 7 de junio 2021] Disponible en: https://desarrolloweb.com/home/html
[21] CSS. *Definición.* [Consulta: 7 de junio 2021] Disponible en: https://definicion.de/css/
[22] HTTP. *Información.* [Consulta: 7 junio 2021] Disponible en: https://plataforma.josedomingo.org/pledin/cursos/flask/curso/u01/
[23] HIGCHARTS. *Definición.* [Consulta: 8 de junio 2021] Disponible en: https://www.ecured.cu/Highcharts
[24] CHARTS. *Definición.* [Consulta: 8 de junio 2021] Disponible en: http://www.chartjs.org
[25] Cordón, Oscar & Herrera, Francisco & Hoffmann, Frank & Magdalena, Luis. (2001). Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases. World Sci, Adv Fuzzy Sys-Appl Theory. 19. 10.1142/4177. [Consulta: 28 de enero 2021]
[26]MMA7361. *Función map.* [Consulta: 14 de junio 2021] Disponible en: https://hetpro-store.com/TUTORIALES/mma7361-sensor-acelerometro/
[27] ESP8266. *Comandos AT*. [Consulta: 15 de junio 2021] Disponible en: https://naylampmechatronics.com/blog/21_tutorial-esp8266-parte-i.html

CAPÍTULO 9. ANEXOS

9.1 Manuales

En esta sección, se presentan dos manuales, un manual de usuario donde se explica de forma simplificada al médico qué se tiene que insertar, por ejemplo, para cambiar la red de sensores de un paciente a otro, etc., y un manual de mantenimiento para el correcto funcionamiento.

9.1.1 Manual usuario

Este apartado trata de los pasos a realizar en el código y en la página web para añadir o modificar un paciente, ya sea nuevo o por otro motivo.

En primer lugar, el médico se identifica en la página web y una vez dentro selecciona “Datos Pacientes”:

- Si el paciente está registrado, tiene que fijarse en la columna “identificación” y tiene que coincidir con el número asociado en la placa Arduino Due.

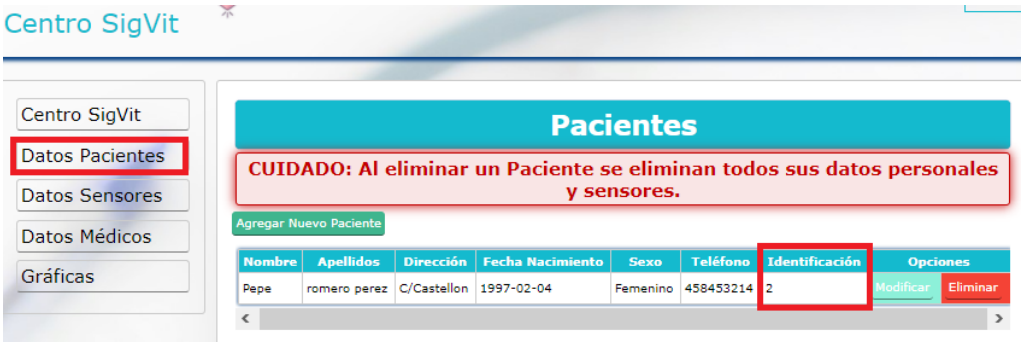


FIGURA 9.1 IDENTIFICACIÓN PACIENTE

- Si se registra un nuevo paciente, se tiene que rellenar un formulario, como se observa en la **figura 9.2**. Y en el campo de “Identificación”, se debe de introducir el número de identificación que está incluido en la placa de Arduino Due. Por lo que, para cada placa de Arduino hay un identificador distinto, así se logra obtener para cada paciente sus datos identificados, ver **figura 9.3**.

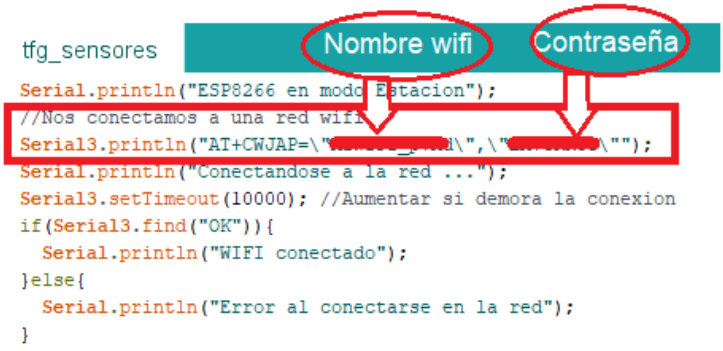
FIGURA 9.2 REGISTRAR PACIENTE

FIGURA 9.3 IDENTIFICACIÓN ARDUINO

- Si al introducir un nuevo paciente, el médico se equivoca al escoger el identificador de la placa Arduino, debe de eliminar ese paciente y volver a introducir de nuevo, por lo que no es posible modificar el campo de identificador.

- Si el paciente cambia de compañía que le ofrece servicios Internet o el router lo configuran de nuevo, haciendo cambiar el nombre del wifi y/o la contraseña,

hay que configurar el sketch de Arduino para que el módulo wifi se pueda conectar a la red y transmitir los datos de los sensores. El cambio a realizar es en el comando AT del código ESP8266, es decir, del módulo wifi. Se observa en la **figura 9.4** y para el envío de datos al servidor la dirección ip del mismo, ver **figura 9.5**. Para esta situación se necesita la intervención de un técnico.




```
Serial.println("ESP8266 en modo Estacion");
//Nos conectamos a una red wifi
Serial3.println("AT+CWJAP=\"[redacted]\",\"[redacted]\"");
Serial.println("Conectandose a la red ...");
Serial3.setTimeout(10000); //Aumentar si demora la conexion
if(Serial3.find("OK")){
  Serial.println("WIFI conectado");
}else{
  Serial.println("Error al conectarse en la red");
}
```

FIGURA 9.4 CONFIGURACIÓN WIFI COMANDOS AT EN ARDUINO

```
//Nos conectamos con el servidor:
Serial3.println("AT+CIPSTART=\"TCP\",\"192.168.1.141\",80");
if(Serial3.find("OK")){
  Serial.println();
  Serial.println();
  Serial.println("ESP8266 conectado al servidor.");
  Serial.println();
  Serial.println();
  //Armamos el encabezado de la peticion http POST
  String tempe = "temperatura="+String(temp);
  String heartRa= "heartRate="+String(pulso);
  String oxigenSan= "spo2="+String(spo2);
  String fuz= "alerta="+String(fuzzy);
  String estadAct= "estadoActual="+String(estado);
  String estadAnt= "estadoAnterior="+String(estadoAnterior);
  String tiempOtraPo= "tiempoOtraPosicion="+String(TiempOtraPos);
  String tiempTrans= "tiempoTranscurrido="+String(TiempoTrans);
  String estadMovBrus= "estadoMovBrusco="+String(estadoMovBrus);
  String carilla= "carillas="+String(carillas);
  String data = tempe+"&"+heartRa+"&"+oxigenSan+"&"+fuz+"&"+estadAct+"&"+estadAnt+"&"+tiempOtraP
  String peticionHTTP= "POST /TFG_Sig_Vit/insertar_Datos_Sensores.php HTTP/1.1\r\n";
  peticionHTTP=peticionHTTP+"Host: 192.168.1.141\r\n";
  peticionHTTP=peticionHTTP+"Accept: */*\r\n";
  peticionHTTP=peticionHTTP+"Content-Type: application/x-www-form-urlencoded\r\n";
  peticionHTTP=peticionHTTP+"Content-Length: "+data.length()+"\r\n\r\n";
  peticionHTTP=peticionHTTP+data;
  //Enviamos el tamaño en caracteres de la peticion http:
  Serial3.print("AT+CIPSEND=");
  Serial3.println(peticionHTTP.length());

  //esperamos a ">" para enviar la petcion  http
  if(Serial3.find(">")){ // ">" indica que podemos enviar la peticion http
    Serial.println("Enviando HTTP . . .");
    //Serial.println(peticionHTTP);
```

FIGURA 9.5 CONFIGURACIÓN IP COMANDOS AT EN ARDUINO

9.1.2 Manual de mantenimiento

En este apartado, se va a explicar el mantenimiento necesario para mantener el funcionamiento correcto.

Por un lado, se debe de comprobar el cableado y el patillaje tanto en la placa Arduino como en los sensores. Además de revisar la correcta colocación del sensor mma7361.

Por otro lado, si la red de sensores se va a proporcionar energía con baterías o pilas, debe de haber un mantenimiento de cargar la batería o las pilas para que no se quede sin conexión.

9.2 Presupuesto

En este apartado se muestra el coste del equipamiento utilizado:

EQUIPO	UNIDADES	PRECIO UNITARIO	PRECIO TOTAL
ARDUINO DUE	1	33,6 €	33,60 €
SENSOR MAX30100	1	5,27 €	5,27 €
SENSOR MMA7361	1	8,79 €	8,79 €
MÓDULO WIFI ESP8266-01	1	5,29 €	5,29 €
PACK DE CABLES ARDUINO	1	5,49 €	5,49 €
ALIMENTACIÓN ARDUINO	1	4,24 €	4,24 €
RESISTENCIA 10KΩ	1	0,15 €	0,15 €
RESISTENCIA 330Ω	2	0,15 €	0,30 €
LED	2	0,15 €	0,30 €
PROTOBOARD	1	3,66 €	3,66 €
TOTAL			67,09 €

Tabla 9.1 Presupuesto materiales utilizados

ACTIVIDAD	HORAS	PRECIO UNITARIO	PRECIO TOTAL
DISEÑO	300	30 €	9.000 €
CONFIGURACIÓN	200	30 €	6.000 €
INSTALACIÓN	1	30 €	30 €
TOTAL			15.030 €

Tabla 9.2 Presupuesto recursos humanos

ÍTEM	PRECIO
GASTOS EN EQUIPAMIENTO	67,09 €
GASTO EN RECURSOS HUMANOS	15.030 €
BASE IMPONIBLE	15.097,09 €
IVA (21%)	3.170,389 €
TOTAL	18.267,479 €

Tabla 9.3 Presupuesto total incluido IVA.

Precio total es de **DIECIOCHO MIL DOSCIENTOS SESENTA Y SIETE EUROS CON CUATROSCIENTOS SETENTA Y NUEVE CÉNTIMOS.**

Costes recurrentes a pagar cada mes, para el acceso a Internet del servidor.

ÍTEM	PRECIO MENSUAL
CONEXIÓN INTERNET	20,90 €
TOTAL	20,90 €

Tabla 9.4 Presupuesto conexión internet

Precio de los costes recurrentes es de **VEINTE EUROS CON NOVENTA CÉNTIMOS.**

9.3 Pliego de condiciones

Para poder desarrollar este proyecto, se ha de tener en cuenta las siguientes consideraciones:

- Los sensores deben estar implementados sobre un Arduino Due, u otro que sea compatible, debido a que según el dispositivo que se utilice el código puede no ser compatible. Además, la placa Arduino Due lleva incorporado una memoria, en el caso de que no se utilice este modelo hay que incorporar una.
- El servidor XAMPP debe de estar conectado a internet permanente, así como los programas introducidos en la memoria, para su correcta instalación y ejecución, como son la base de datos “MySQL” y el servidor Apache Tomcat.
- Para configurar los dispositivos hay que tener un ordenador, tener instalado Arduino IDE para su configuración a través de un adaptador USB.