COJÍN INTELIGENTE PARA MONITORIZACIÓN REMOTA DE PERSONAS DEPENDIENTES

Autor
JOSE IGNACIO VALERO CERDÁ
(jovacer@etsid.upv.es)

Director: JOSÉ ENRIQUE SIMÓ TEN
(jsimo@disca.upv.es)

TRABAJO FIN DE MÁSTER
MÁSTER DE AUTOMÁTICA E INFORMÁTICA INDUSTRIAL
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
Valencia, 01/09/2020

Dedicatoria

Me gustaría dedicar este trabajo de fin de máster a mi familia principalmente, a mis compañeros y, por supuesto, a los profesores que han hecho posible que la calidad del máster se viera lo menos afectada por la situación que vivimos actualmente.

AGRADECIMIENTOS

Agradezco individualmente la ayuda de mi tutor José Enrique Simó Ten, por darme la posibilidad de realizar este proyecto que se adapta perfectamente a lo que yo buscaba en un trabajo de fin de máster.

Por otro lado, agradezco a toda la comunidad que existe en internet envuelta alrededor de la programación en general. Es muy importante el apoyo que existe actualmente al alcance de cualquiera en numerosos foros sin ánimo de lucro y eso es digno de agradecer.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN

El siguiente documento describe la realización de un proyecto llevado a cabo en la Universidad Politécnica de Valencia por un alumno perteneciente al Máster de Automática e Informática Industrial y dirigido por José Enrique Simó Ten, catedrático de Universidad y perteneciente al Dpto. de Informática de Sistemas y Computadores en la Escuela Técnica Superior de Ingeniería Informática.

Durante el desarrollo del Trabajo de Fin de Máster, se plantea la implementación de un sistema de monitorización a distancia de personas dependientes haciendo uso de uno o varios cojines inteligentes situados en la vivienda del individuo. Este cojín inteligente nace de la idea del IoT (*Internet of Things*), en español, Internet de las Cosas.

Por consecuente, este proyecto pretende demostrar el desarrollo de un sistema inteligente de bajo coste capaz de monitorizar y controlar la actividad diaria de una persona dependiente de una manera no intrusiva, haciendo uso de la tecnología implementada en objetos cotidianos.

Para ello, en este documento se presenta el desarrollo electrónico completo y necesario para llevar a cabo un montaje íntegro del dispositivo de monitorización. Comenzando por el sensorizado del entorno hasta el sistema de comunicación. Posteriormente, se plantea la programación de los diferentes microcontroladores y software menester para la implementación. Todo esto acompañado de un listado del hardware utilizado y su debida calibración y programación.

Finalmente, se presentarán los resultados obtenidos tras la elaboración completa de una primera maqueta funcional y las conclusiones obtenidas.

ABSTRACT

The following document describes the realization of a project carried out at the Polytechnic University of Valencia by a student belonging to the Master of Automation and Industrial Informatics and directed by José Enrique Simó Ten, professor at the University and belonging to the Department of Computer Systems and Computers at the Higher Technical School of Computer Engineering.

During the development of the master's Final Project, the implementation of a remote monitoring system for dependent people is proposed using a smart cushion located in the individual's home. This smart cushion was born from the idea of the IoT (Internet of Things).

Consequently, this project aims to demonstrate the development of a low-cost intelligent system capable of monitoring and controlling the daily activity of a dependent person in a non-intrusive way, making use of the technology implemented in everyday objects.

To do this, this document presents the complete electronic development necessary to carry out a complete assembly of the monitoring device. Starting with the sensorization of the environment to the communication system. Subsequently, the programming of the different microcontrollers and necessary software for implementation is proposed. All this accompanied by a list of the hardware used and its proper calibration and programming.

Finally, the results obtained after the complete elaboration of a first functional model and the conclusions obtained will be presented.

1. INTRODUCCIÓN

El Internet de las Cosas, IoT por sus siglas en inglés, se basa en la implementación de una conexión entre objetos utilizados por el ser humano de manera habitual e internet. De este modo, se presenta la posibilidad de realizar una interpretación de las diferentes situaciones cuotidianas de una manera no intrusiva para las personas. Básicamente, se trata de estudiar y gestionar el entorno presentado de la manera más sutil posible, sin interferir en el desarrollo habitual de las acciones diarias mediante la conexión de estos dispositivos a internet.

Actualmente, el IoT presenta numerosas ventajas. Tanto que en ocasiones pasan desapercibidos objetos que pertenecen a este grupo desde hace relativamente poco y, sin embargo, hoy en día no se podría prescindir de ellos. Un claro ejemplo de este caso es un teléfono móvil inteligente, lo que actualmente se conoce como "Smartphone", hace poco más de un par de décadas únicamente servía para para emitir y recibir llamadas telefónicas. A día de hoy es capaz de almacenar miles de fotografías, libros, películas y transmitir toda esta información a internet en segundos.

Este avance tecnológico que está sucediendo hoy en día provoca que cada vez más dispositivos tengan una mayor utilidad si se encuentran conectados a internet. Una nevera podría ser capaz de indicar en tiempo real la calidad del aire situado en su interior, su temperatura, la cantidad de alimentos que hay o qué hace falta comprar a través de una aplicación, por ejemplo. O quizá un sofá conectado a la red podría avisarte si ya es hora de hacer un poco de ejercicio, o si el volumen de tu televisor es demasiado elevado y por lo tanto perjudicial para tus oídos.

Sin embargo, en esta ocasión se plantea una utilidad capaz de hacer un poco más fácil la vida de aquellas personas que poseen una movilidad reducida o que son dependientes y que, por lo tanto, pasan un gran número de horas sentadas sobre una misma posición o en un mismo lugar.

2. JUSTIFICACIÓN

Según los datos proporcionados por el ministerio de Sanidad, en julio del año 2019 se registró un total de 1.331.251 personas dependientes en nuestro país, de las cuales, más de un 25% no recibe ningún tipo de prestación o ayuda por parte del Estado, incluso cumpliendo todos los requisitos para ello. Y si bien estas cifras son elevadas, un estudio presentado por el Presidente Nacional de Lares afirma que estos valores podrían duplicarse en los próximos diez años.

Con la realización de un dispositivo capaz de monitorizar la actividad diaria de estas personas dependientes de forma totalmente no intrusiva para ellas podría aparecer un gran número de ventajas ante múltiples situaciones. Entre ellas nos encontramos la posibilidad de conocer la información relevante acerca de la cantidad de movimiento llevado a cabo durante un día por cada una de estas personas. Además de saber con certeza qué parte de su tiempo ha sido invertido en permanecer sentado, es decir, sin actividad física. Por otro lado, mediante la implementación de diferentes sensores de bajo coste, se plantea tanto la monitorización de actividad como el conocimiento en tiempo de real de datos tan vitales como la temperatura del espacio en el que se encuentran estas personas e incluso el porcentaje de humedad del lugar en el cual están sentados.

La obtención de todos estos datos relacionados con la actividad de personas dependientes conllevaría una ventaja muy importante a la hora de personalizar de una manera más individual los tratamientos necesarios para cada uno de estos individuos. Desde la programación de varios tipos de alarmas que propongan al usuario a levantarse y desplazarse hasta otro punto de la casa o incluso un indicador que nos informe cuando sea necesario un cambio de ropa del paciente debido a un exceso de humedad en su cojín.

Es por eso por lo que se ha decidido realizar este proyecto haciendo uso de microcontroladores y sensores de bajo coste. Un dispositivo útil, sencillo de programar y que aporte una gran versatilidad y personalización para cada una de las personas que así lo requieran.

3. OBJETIVOS

3.1 Objetivo general

El objetivo principal del proyecto se encuentra en la monitorización de la actividad diaria que realiza una persona dependiente. Para ello, se hará uso de una variedad de sensores situados en el entorno de esta persona. Estos sensores aportarán la información necesaria para conocer el estado actual del individuo, ya sea en tiempo real o un historial de momentos anteriores. Por último, toda esta información será presentada de manera gráfica a aquellos usuarios que necesiten conocer dicha actividad, ya sea un familiar o personal sanitario a cargo de la situación.

A través de la realización de esta investigación se pretende conseguir información acerca de la temperatura y humedad del propio cojín, la temperatura del habitáculo y el número de horas que el paciente pasa sentado. Toda esta información será gestionada y almacenada para ser mostrada al usuario que se encuentre a cargo del paciente. Finalmente, se pretende incluir un sistema de creación y gestión de alarmas que sea capaz de provocar respuestas a las diferentes situaciones presentadas en el día a día.

Así pues, se pretende mejorar la comunicación entre la persona y su responsable en cualquier momento y en tiempo real, quedando, además, un registro completo de toda la información recopilada que será accesible para aquellos usuarios con derechos de administración.

3.2 Objetivos específicos

En cuanto a los objetivos específicos que se deberán llevar a cabo para cumplimentar la realización de este proyecto, se ha optado por dividir todo el trabajo en tres amplios puntos de control que servirán para organizar de un modo más estructurado todas y cada una de las tareas a realizar:

Objetivo 1: El primer punto de control a alcanzar en el desarrollo de este proyecto se encuentra en el diseño del primer módulo del sistema final. Este módulo consta de un microcontrolador capaz de recibir los datos obtenidos por diferentes sensores de presión, temperatura y humedad transmitidos mediante radiofrecuencia desde diferentes microcontroladores de bajo coste situados en la habitación que se desea monitorizar. El uso del segundo microcontrolador se debe a que de este modo se simplifica de gran manera la decodificación del mensaje emitido por radiofrecuencia ya que estos microcontroladores trabajan con la misma librería proporcionada por el fabricante.

Objetivo 2: El segundo objetivo del proyecto se encuentra en la recopilación de todos estos datos en un microcontrolador más avanzado que los mencionados anteriormente. Este microcontrolador constará de un sistema empotrado conectado a internet que interpretará los valores obtenidos, gestionará las diferentes alarmas, notificaciones, etc. Posteriormente, el sistema empotrado (RasPi) que soportará la gestión de las alarmas, notificaciones y la programación de perfiles enviará toda la información pertinente a través de comunicación estándar IoT (MQTT) a una aplicación cliente dónde aparecerán las notificaciones y monitorización en tiempo real.

Objetivo 3: La última etapa de la investigación terminará cuando toda la información relacionada con el estado de las alarmas, notificaciones y demás, sea plasmada en una interfaz gráfica sencilla para cualquier usuario a la cual se accederá desde una conexión local a través de una URL proporcionada por el sistema.

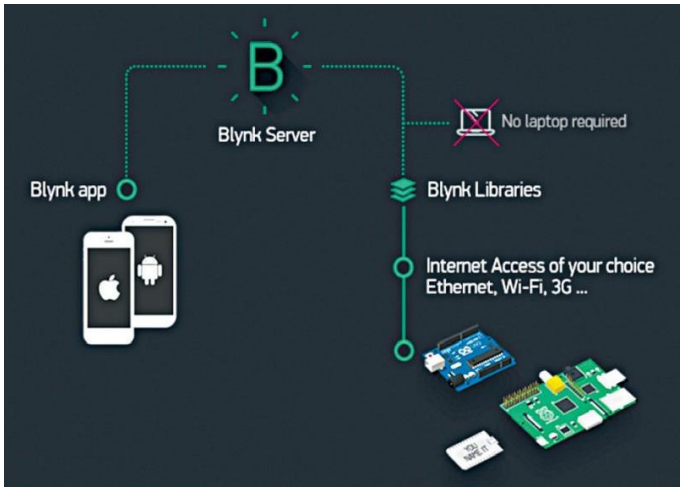
4. ESTADO ACTUAL

En este apartado se presenta una síntesis de conocimientos, investigaciones y/o conceptos que ya se conocen debido a trabajos realizados hasta el momento. Por otro lado, aquellos aspectos de los cuales se posee menos información o que puedan presentar alguna controversia o hagan formular preguntas de respuesta no certera. Finalmente, se describe la contribución que propone este proyecto al relacionar las ideas teóricas con los resultados prácticos.

Para ello, tras realizar una búsqueda bibliográfica, se presentan varios casos en los que se ha llevado a cabo un sistema inteligente de monitorización basado en el IoT. El primero de ellos se trata de un dispositivo capaz de medir el consumo mensual de agua en cada casa para mejorar la distribución de esta. La investigación fue llevada a cabo en 2019 por *Ashwini Kumar Sinha* y pretendía recoger información acerca del gasto de agua diario en su casa y almacenarlo en una base de datos a la cual podría acceder el gobierno.

Fue en ese mismo año (2019) cuando *Biswajit Das* desarrolló un aparato que combinaba el uso de la plataforma *Blink* con un microcontrolador de Arduino para monitorizar en tiempo real la cantidad de polución del aire a través de una aplicación móvil. Para llevar a cabo este diseño, el autor siguió la estructura y estándares del IoT.

Figura 1. Estructura *Blink* con Sistema embebido



Pese a que la cantidad de referencias acerca de este tipo de proyectos es muy grande hoy en día, no resulta tan fácil encontrar trabajos que propongan dispositivos de lectura que "invadan" la zona de confort del usuario. Mayormente se trata de instrumentos que realizan lecturas del entorno de manera estática, es decir, no van a sufrir un movimiento constante del propio sistema en el entorno.

Finalmente, la mayor controversia que presenta la idea del IoT gira entorno a la falta de seguridad que conllevan los sistemas desarrollados. Por un lado, los servicios de red que se ejecutan en los dispositivos y se encuentran conectados a internet presentan una gran vulnerabilidad. Además, la información recolectada en las bases de datos puede ser utilizada de manera poco segura o sin permiso. Así pues, jefe del laboratorio de ESET Latinoamérica, Camilo Gutiérrez, declara que los casos de amenazas a dispositivos IoT aumentarán en los próximos años.

5. METODOLOGIA EXPERIMENTAL

5.1 Planificación del trabajo

Para llevar a cabo la realización de un proyecto como el que se trata en esta memoria, es necesario partir de una planificación estructurada de todas y cada una de las partes que se vayan a ver involucradas durante su desarrollo. Por eso, desde el principio es importante comenzar por una buena organización y estructuración de las diferentes etapas de la investigación que se va a realizar.

En este caso, se plantea la implementación de una de las metodologías de trabajo más tradicionales, la denominada "metodología en cascada" o "waterfall", en inglés. Este método de desarrollo de proyectos consta de un modelo básicamente lineal de construcción y desarrollo de software que se basa en la utilización de un proceso de diseño secuencial. De este modo, el desarrollo del proyecto pasa por cada una de las etapas establecidas hasta alcanzar su parte final. Estas etapas no tienen por qué ser fijas, pero suelen cumplir una estructura similar. Por ejemplo: requisitos del cliente, diseño, implementación, verificación y mantenimiento.

Una de las ventajas principales que aporta esta forma de plantear el trabajo a realizar y por la cual se ha escogido en este proyecto es su clara linealidad durante el desarrollo de las diferentes partes diferenciadas del trabajo, ya que, al tratarse de una investigación llevada a cabo por un único integrante, es más sencillo poseer una visión del avance que se alcanzado y el que todavía queda por abordar. Sin embargo, esta linealidad que aporta la metodología a la realización del documento puede ser un inconveniente a la hora de alterar partes o añadir modificaciones *a posteriori*. Es por eso por lo que no se descarta la aplicación de la metodología en espiral para ocasiones puntuales en las que se planteen cambios no previstos durante el desarrollo ya que aportará al resultado final la posibilidad de retroalimentarse de errores cometidos durante su desarrollo.

Figura 2. Metodología experimental en cascada

Tal y como se aprecia en el gráfico mostrado, se ha optado por abordar cada uno de los tres principales objetivos explicados en el apartado 3.2 de esta memoria de manera lineal, siguiendo las bases de la metodología en cascada. Sin embargo, se observa que se ha planteado como última etapa, el estudio de posibles mejoras sobre el trabajo realizado una vez se tenga una visión global de los objetivos cumplidos.

Este esquema de ataque lineal se aplicará independientemente a cada una de las fases diferenciadas del proyecto:

1. Diseño de un cojín inteligente capaz de recopilar la información necesaria de su entorno y transmisión de esta a un microcontrolador mediante radiofrecuencia.
2. Recopilación de toda la información en un microcontrolador más potente, conectado a internet y publicación de esta a través del protocolo de comunicación MQTT. Gestión de alarmas e interpretación avanzada de los datos obtenidos inicialmente.
3. Gestión de la monitorización y muestra mediante interfaz de uso sencillo al usuario a través de un servidor web.

A continuación, se muestra un esquema global de la estructura principal del proyecto en cuestión dividido por módulos u objetivos:

Figura 3. Esquema de comunicación del primer módulo

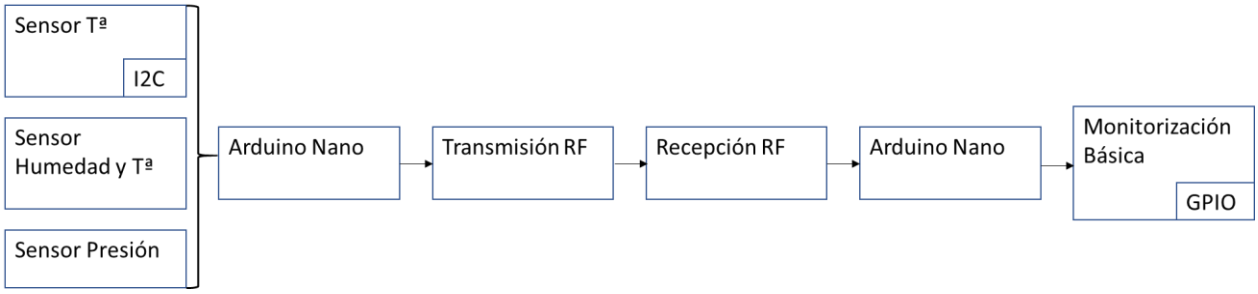


Figura 4. Esquema de comunicación del segundo módulo

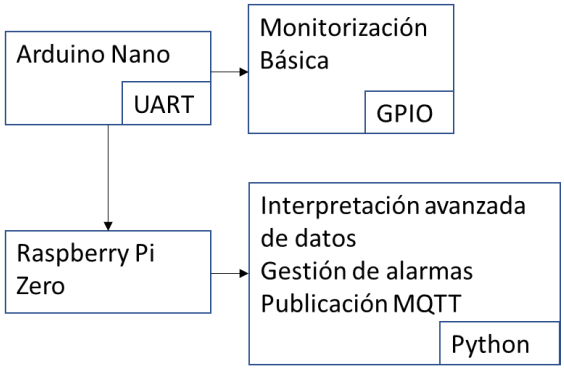
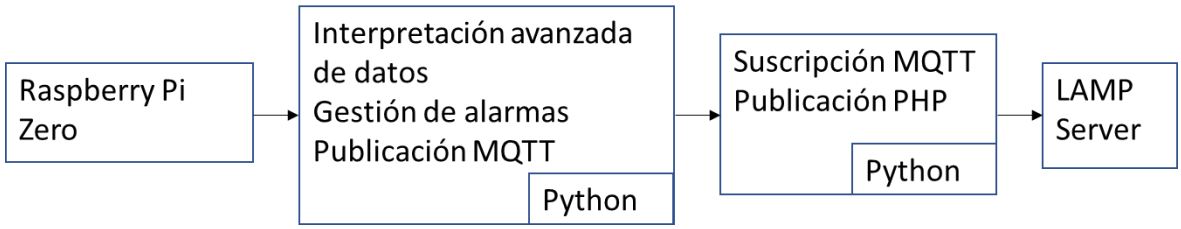


Figura 5. Esquema de comunicación del tercer módulo



Mediante estos tres diagramas se obtiene la máxima simplificación del sistema completo, desde la recogida de información a través del sensorizado de la habitación, la comunicación entre los diferentes controladores y la recogida final en el servidor web.

6. SELECCIÓN DE COMPONENTES Y MONTAJE

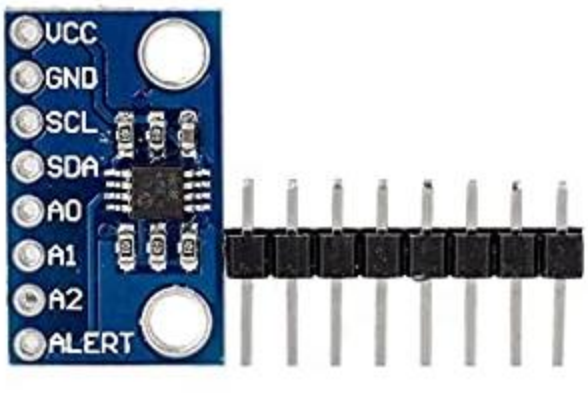
6.1 Sensorizado

Una vez establecidos los objetivos que se desean alcanzar con la realización de este proyecto, el primero de los pasos tras conocer los requisitos del sistema explicados previamente es obtener la cantidad necesaria de información acerca del estado actual del habitáculo. Para ello, como ya se ha explicado anteriormente, se van a utilizar diferentes sensores que nos ofrecerán los datos adecuados. En este apartado se mencionará cada uno de los dispositivos que se verán involucrados. Además, se estudiará brevemente su funcionamiento.

6.1.1 Sensor de temperatura

El primer dato que se desea conocer para realizar la monitorización del entorno del cojín inteligente es la temperatura del habitáculo en el que se encuentra el paciente, por lo tanto, se coloca un sensor de temperatura en la parte exterior del cojín para que esta no se vea afectada por el calor que desprenda tanto la persona sentada encima como el propio microcontrolador. Con el fin de obtener unos valores de temperatura lo suficientemente fiables y de la manera más sencilla posible sin alcanzar un coste de adquisición elevado, se plantea el uso del sensor digital MCP9808.

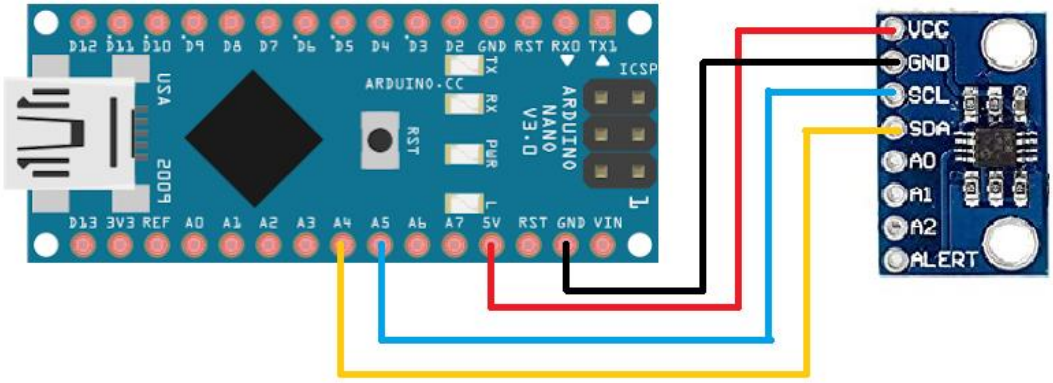
Figura 6. Sensor de temperatura I2C de alta precisión (MCP9808 Adafruit)



Este sensor es capaz de convertir temperaturas entre -20º C y +100º C a valor digital con un error típico de ±0.25º C y un máximo de ±0.5º C. Por otro lado, debido a su gran simplicidad y reducido tamaño, aporta al proyecto una gran versatilidad a la hora de leer los datos de este mediante un microcontrolador también de bajo coste y poder, de este modo, obtener un dispositivo de lectura de temperatura de un tamaño bastante reducido y casi despreciable masa.

Para hacer uso de este sensor, únicamente se necesita conectar el pin de colector de voltaje (VCC), masa (GND), la línea de pulsos o "System clock" (SCL) y la de datos o "System Data" (SDA) con los pines correspondientes del microcontrolador tal y como se muestra en el siguiente diagrama de conexiones con un Arduino Nano:

Figura 7. Diagrama de conexiones MCP9808



Una vez obtenido el conexionado correcto del sensor de temperatura de alta sensibilidad y el microcontrolador en cuestión, únicamente resta la programación del Arduino Nano. Para la misma, se hace uso de la librería "*Wire.h*" y la proporcionada por el fabricante "Adafruit_MCP9808.h". Por lo tanto, tras incluir las librerías en nuestro programa y crear un objeto de tipo "Adafruit_MCP9808" llamado "tempsensor" se procede a configurar la resolución de este mediante la siguiente orden:

```
tempsensor.setResolution(1);  //Ajusta la resolución al
segundo modo de lectura.
```

Los modos de lectura, su resolución y el tiempo de muestreo se encuentran definidos en la siguiente tabla proporcionada por el fabricante del sensor:

Tabla 1. Modos de lectura del sensor MCP9808

Modo	Resolución	Tiempo de muestreo
0	0.5º C	30 ms
1	0.25°C	65 ms
2	0.125°C	130 ms
3	0.0625°C	250 ms

En este caso, se plantea el uso del modo "1", que conlleva un tiempo de muestreo de 65 ms, sin embargo, posteriormente se podrá modificar en función de las necesidades que tenga el sistema. Por último, efectuar la lectura de la temperatura en la habitación mediante las siguientes líneas de código:

```
tempsensor.wake(); //Despierta el sensor
float  temp  =  tempsensor.readTempC();  //Lectura  de  la
temperatura actual
```

En este momento, la temperatura actual del habitáculo se encuentra almacenada en la variable "*temp*" de tipo "float" lista para ser enviada.

6.1.2 Sensor de humedad

Pese a que en el primer planteamiento del proyecto se ofrece una solución a la lectura de los valores de temperatura y humedad conjunta, se ha optado por separar estas mediciones mediante dos sensores independientes. Esto se debe principalmente a que, tras varias pruebas de funcionamiento, se observa que las mediciones de temperatura en el interior del cojín tras varios minutos de funcionamiento no corresponden con los valores reales obtenidos en el exterior de este debido al calor que emite el dispositivo y el propio individuo sentado sobre el cojín.

Es por eso por lo que se propone el uso del sensor DHT22 de temperatura y humedad para comprobar el estado del interior del cojín. En este caso únicamente será necesaria la obtención del valor de la humedad, aunque no está de más obtener un valor de la temperatura interior del cojín para conocer si existen problemas de sobrecalentamiento o similares.

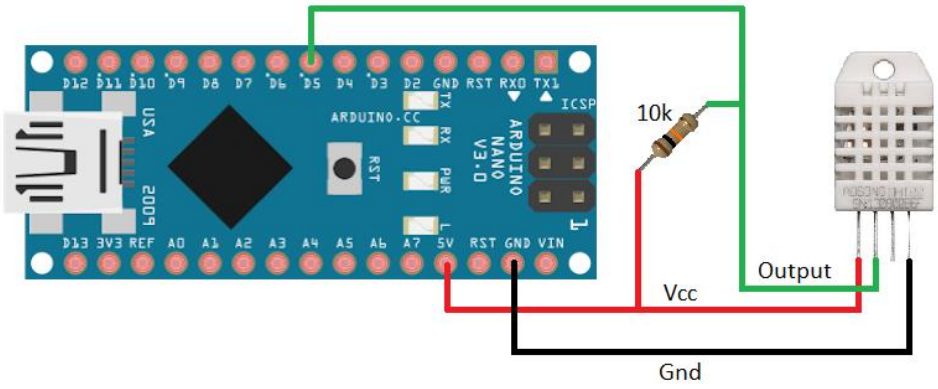
Figura 8. Sensor de temperatura y humedad (DHT22)

Este sensor, pese a que no se encuentra dentro de la familia de los sensores de alta precisión, posee una serie de características que lo convierten en idóneo para este proyecto. Entre ellas, su bajo coste, que sigue la línea de los demás sensores utilizados, así como su reducido tamaño y casi despreciable masa.

En cuanto a sus características técnicas, se trata de un dispositivo capaz de detectar temperaturas entre -40º C y 125º C con una precisión de 0.5º C. Por otro lado, mide la humedad ambiental entre el 0 y el 100% con precisión del 2 al 5%. Y, por último, todo esto con una frecuencia de muestreo de 2 Hz (500 ms).

Para hacer uso de este sensor, se debe conectar tal y como se aprecia en el siguiente diagrama, pudiendo variar el pin de lectura, en este caso el pin digital 5:

Figura 9. Diagrama de conexiones DHT22



Finalmente, es necesario unas breves líneas de código en el programa para obtener los valores proporcionados por el sensor y almacenarlos en una variable de tipo entero. De este modo, tras incluir la librería proporcionada por el fabricante "DHT.h" se hace uso del siguiente código:

```
const int DHTPin = 5; //Pin D5 del Arduino Nano utilizado
dht.begin(); //Arranque del sensor DHT
float hum = dht.readHumidity(); //Lectura del valor de humedad
```

Así pues, se obtiene el valor de la humedad del interior del cojín en porcentaje comprendido entre 0 y 100% en una variable de tipo "*float*" llamada "*hum*" lista para ser enviada.

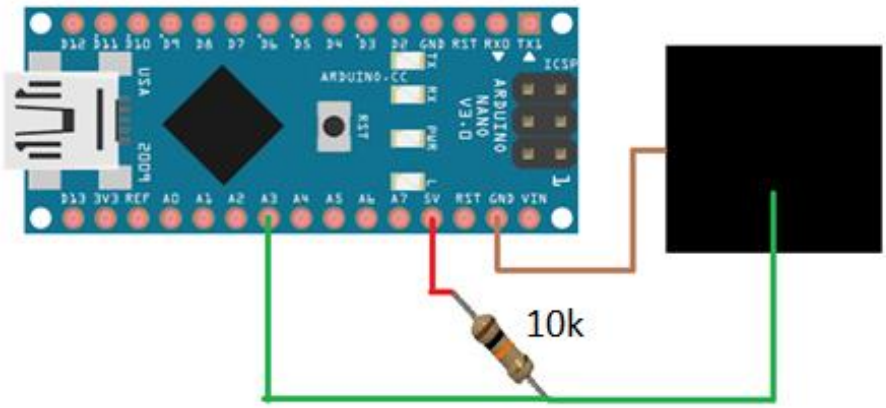
6.1.3 Sensor de presión

Para ser capaces de identificar el tiempo que la persona pasa sentada en el cojín inteligente es necesario detectar la presión ejercida sobre el mismo. Para ello, se plantea el uso de un material conductivo sensible a la presión situado de forma horizontal en el interior del cojín. Se trata de una hoja flexible de color negro conocida como "Velostat" que es capaz de modificar su resistencia al paso de corriente en función de la presión ejercida sobre la misma. Además, presenta la capacidad de ser cortada tantas veces como se desee sin perjudicar su funcionamiento, lo cual aporta una gran versatilidad a la hora de escoger el tamaño necesario.

Figura 10. Hoja conductiva sensible a la presión (Velostat Adafruit)

A la hora de integrar esta hoja conductiva en el sistema de lectura de datos, es necesario hacer uso de lo que se conoce como un divisor de tensión. Para ello, se aplica masa (0 V) en uno de los laterales del material. En el otro extremo conectaremos una resistencia de un valor arbitrario, por ejemplo, 10k Ω (R1). Al extremo contrario de la resistencia aplicamos un voltaje de 5 V y medimos el valor del voltaje de salida entre ambas resistencias mediante un pin de entrada del microcontrolador.

Figura 11. Diagrama de conexiones hoja conductiva Velostat



En este caso no es necesario hacer uso de ninguna librería en concreto, ya que únicamente será menester obtener el voltaje resultante a través del pin de entrada A3 del Arduino Nano para determinar si se está ejerciendo presión sobre el sensor. De este modo, si se calcula en valor de la tensión de salida del divisor con los valores máximo y mínimo de la resistencia proporcionada por la hoja conductiva "R2" (1k Ω al ejercer presión y 20k Ω al no ejercer ninguna presión) obtenemos los siguientes valores:

$$V_{out} = V_{in}\frac{I \cdot R_2}{I(R_1 + R_2)} = \frac{V_{in} \cdot R_2}{(R_1 + R_2)}$$

$$V_{out\,max} = 3.33\,V$$

$$V_{out\,min} = 0.45\,V$$

Por lo tanto, se supone que al aplicar presión sobre el cojín y, por lo tanto, sobre la hoja conductiva, al variar la resistencia frente al paso de corriente de esta, obtendremos un voltaje de entrada en el pin A3 del microcontrolador de 0.45 voltios. Sin embargo, mientras no se ejerza ninguna presión sobre el sistema, el voltaje de entrada sobre el pin seleccionado será de 3.33 V. De este modo se puede distinguir fácilmente cuando la persona dependiente permanece sentada para así ejercer un control sobre el tiempo a lo largo del día y para ello se efectúa la lectura de estos valores mediante el siguiente código:

```
int p = analogRead(A1); //Lectura analógica del pin A3
```

El valor de la variable entera "p" siempre se encontrará entre 296 y 675 debido a los valores mínimo y máximo del voltaje de salida del divisor de tensión. Por lo tanto, se obtendrá un valor de "p" de 296 cuando la persona se encuentre sentada y un valor de 675 cuando no lo esté.

```
int pres;
if (p <= 400){
     pres = 1;}
else{
     pres = 0;}
```

Finalmente, se obtiene un valor de la variable "pres" que puede variar entre "0" y "1" indicando si la persona se encuentra sentada (1) o no (0).

6.2 Recopilación de datos

6.2.1 Microcontrolador Arduino NANO

Tal y como se ha introducido anteriormente, una vez captados los datos necesarios del entorno del paciente, es necesario interpretar los valores obtenidos a través de los sensores y almacenarlos en variables para así poder transmitirlos por medio de comunicación radiofrecuencia a mediante de un transmisor conectado al microcontrolador hasta el sistema principal conectado a internet (Raspberry Pi).

Para ello, se plantea el uso de un microcontrolador mencionado en el apartado anterior, el Arduino Nano. Este se caracteriza por ser un dispositivo de tamaño reducido que, pese a su bajo coste, es capaz de llevar a cabo procesos casi tan pesados como el Arduino UNO.

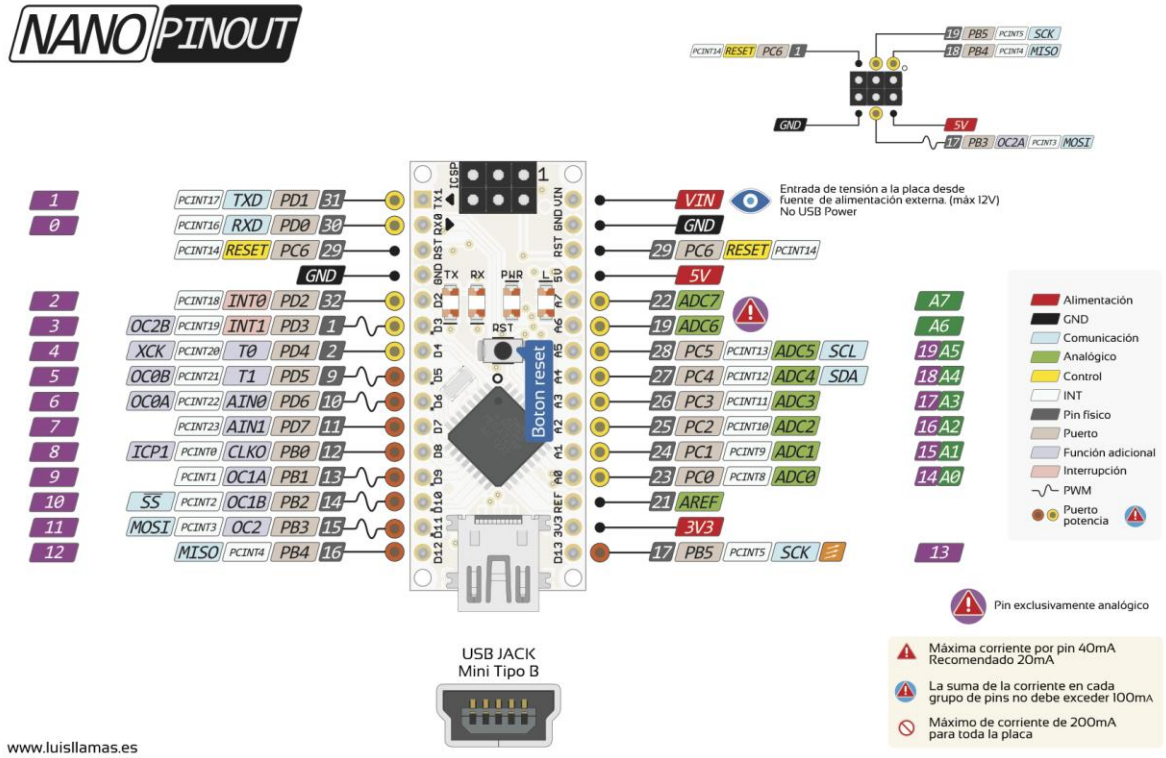
En cuanto a las características técnicas de esta placa, a continuación, se muestra una tabla proporcionada por el fabricante en la que se observa como su tamaño no excede los cinco centímetros de longitud y su masa es de 7 gramos.

Tabla 2. Especificaciones Arduino Nano

Modo	Resolución
Microcontrolador	ATmega328
Arquitectura	AVR
Voltaje Operativo	5 V
Memoria Flash	32 KB
SRAM	2 KB
Velocidad de reloj	16 MHz
Entradas Analógicas	8
Corriente en E/S	40 mA
Voltaje de entrada	7 – 12 V
Pines Digitales E/S	22
Salida PWM	6
Consumo	19 mA
Tamaño	18 x 45 mm
Masa	7 g

"El esquema de patillaje de un dispositivo electrónico, o *pinout*, es uno de los documentos de referencia más útiles y que más frecuentemente consultaremos a la hora de realizar un montaje. El caso de Arduino no es una excepción, más aún teniendo en cuenta la gran cantidad de pines y modelos de placas disponibles." – Luis Llamas 2015

Figura 12. Pinout Diagram Arduino NANO

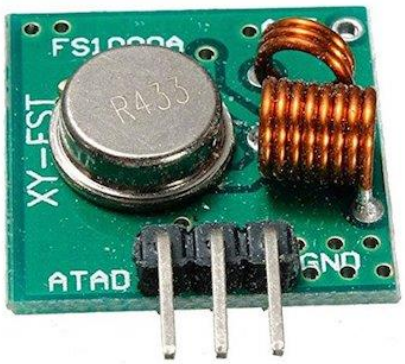


6.3 Emisión de datos mediante RF 433 MHz

A la hora de transmitir datos desde un dispositivo a otro de manera sencilla sin hacer uso de una conexión a internet, se plantean varias soluciones. Entre ellas se encuentra el uso de la tecnología bluetooth, infrarrojos, radiofrecuencia, etc.

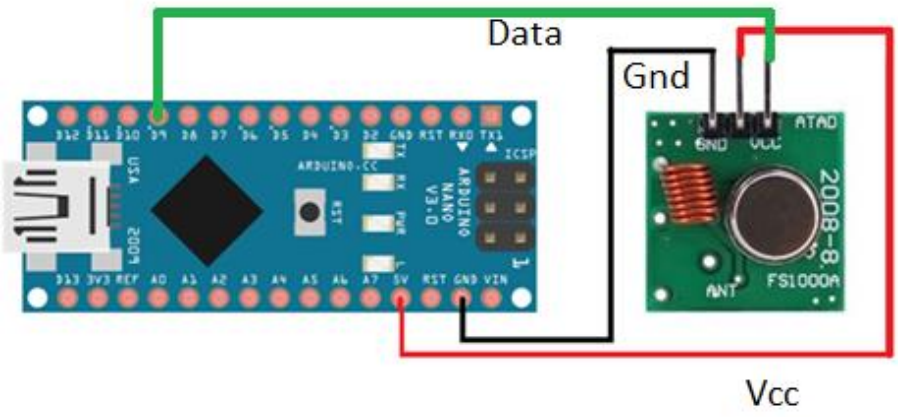
En este caso se ha optado por implementar el uso de emisores y receptores radiofrecuencia. Esto proporciona una gran versatilidad al proyecto gracias a su reducido tamaño de apenas 1 cm, su bajo coste frente a otros sistemas de comunicación y su sencilla implementación por medio de librerías de código libre ya existentes gracias a la comunidad actual.

Figura 13. Emisor radiofrecuencia 433 MHz



Para poder hacer uso del transmisor radiofrecuencia de 433 MHz es necesario seguir el siguiente esquema electrónico para conectarlo al microcontrolador donde se encuentran ya conectados los sensores anteriores:

Figura 14. Esquema de conexiones para el transmisor RF 433 MHz



Una vez conectado correctamente al microcontrolador, es necesario hacer uso de una serie de librerías de código libre para así llevar a cabo una programación sencilla del transmisor radiofrecuencia. Esta librería se conoce como

"*VirtualWire.h*" y acompañada del siguiente código, se consigue emitir mediante radiofrecuencia los valores de las variables obtenidas en los apartados anteriores:

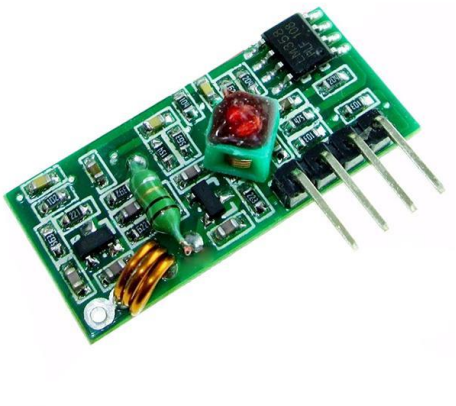
```
const int dataPin = 9;//Se declara el pin 9 como dataPin
vw_setup(2000);
vw_set_tx_pin(dataPin);//Se identifica como pin de salida RF
char buf[VW_MAX_MESSAGE_LEN];//Buffer de tamaño máximo
strtemp = "t" + String(temp);//Se añade el identificador
strtemp.toCharArray(buf,sizeof(buf));//Se convierte en array
vw_send((uint8_t *)buf, strlen(buf));//Envio del paquete
vw_wait_tx();//Esperar a terminar el envio
```

Mediante las líneas de código mostradas previamente, se declara el pin por el cual se va a transmitir la información mediante radiofrecuencia, se crea una cadena de caracteres del tamaño máximo permitido por la librería Virtual Wire, se crea también una variable de tipo "String" que almacena el valor de la temperatura obtenida y un identificador que permitirá diferenciarla posteriormente del resto de variables al ser recibidas en otro dispositivo (en este caso el identificador es la letra "t").

6.4 Recepción de datos mediante RF 433 MHz

Para recibir los datos en el sistema principal, es necesario continuar con el uso de las librerías proporcionadas por el fabricante, "*VirtualWire.h*". En este caso se utilizará un módulo receptor de ondas radiofrecuencia de 433 MHz. Este módulo conectado a otro Arduino Nano que actuará de mediador entre el cojín y el sistema empotrado recibirá constantemente valores representativos del estado actual de la habitación acompañados de un identificador que permitirá la diferenciación entre cada uno de ellos.

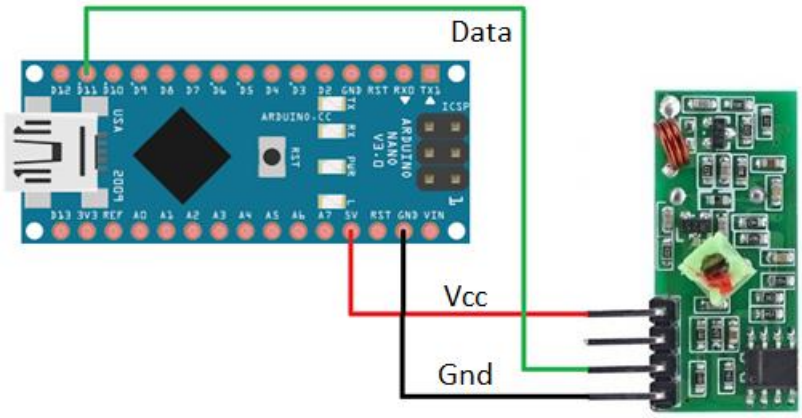
Figura 15. Receptor radiofrecuencia 433 MHz



A continuación, se muestra el esquema de conexiones entre el módulo receptor y el microcontrolador. Únicamente es necesario alimentar el sensor a 5 voltios a través del pin adecuado del microcontrolador, del mismo modo el pin GND a la

masa del dispositivo y el pin por el cual se efectuará la lectura de información recibida, en este caso el pin digital 11 del segundo Arduino Nano.

Figura 16. Esquema de conexiones para el receptor RF 433 MHz



6.5 Monitorización básica

En este momento, se encuentra el final del primer módulo que finalmente formará el sistema completo. Con el fin de poder hacer cada uno de estos módulos "independientes", se plantea la implementación de luces LED de baja intensidad para interpretar la información recogida y poder mostrarla al usuario de manera sencilla.

De este modo, se configura una serie de LED que cambian su color en función de los valores de temperatura, humedad o presión obtenidos por los sensores del cojín inteligente. Así pues, se puede definir por primera vez una serie de rangos de estos valores que serán interpretados como normales, bajos o altos par así alertar al usuario mediante las propias luces.

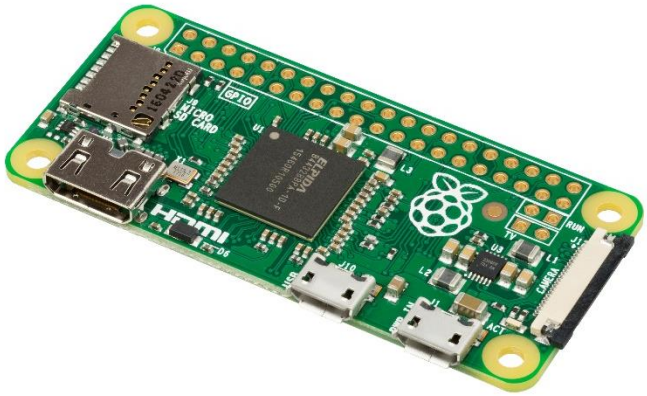
A modo de ejemplo, si la temperatura de la habitación asciende por cualquier motivo y por lo tanto supera los 30º C, una luz roja se encenderá con la etiqueta de "Temperatura". Por otro lado, si la humedad del cojín asciende debido a un derrame de líquidos sobre el mismo, una luz de color rojo se encenderá con la etiqueta de "Humedad". En caso de que todos los valores se encuentren dentro del rango conocido como "Correcto", todas las luces se encontrarán de color verde, indicando, simplemente que el sistema está encendido y funcionando correctamente.

Para la implementación de estas luces LED, basta con hacer uso de las salidas digitales que posee el microcontrolador Arduino Nano y alguna resistencia para controlar la intensidad que pasa por las mismas.

6.6 Comunicación puerto serie

A partir de este punto, se describe lo perteneciente al segundo de los módulos del sistema. Si se observa el avance realizado hasta el estado actual, el sistema se sitúa en un punto en el cual el cojín, lleno de sensores capaces de identificar valores que representen el estado actual del habitáculo en diferentes aspectos, es capaz de obtener dichos datos y transmitirlos mediante un emisor y un receptor radiofrecuencia de 433 MHz y mostrarlos de manera sencilla al usuario. Posteriormente, se requiere hacer llegar esta información hasta la Raspberry Pi con acceso a internet.

Figura 17. Raspberry Pi Zero



Con el fin de obtener una comunicación estable, rápida y segura entre el microcontrolador Arduino Nano (que no posee una capacidad de conexión a internet por defecto) y la Raspberry Pi Zero (en adelante "Raspi") se plantea el uso de la comunicación mediante puerto serie. Este tipo de comunicación está establecida en el Arduino por defecto. Por lo tanto, se trata de unas simples líneas de código para comenzar con este protocolo. En lado de la "Raspi", del mismo modo, se efectuará la lectura del puerto serie conectado a través de la entrada USB. Esta entrada además aportará un voltaje de 5 voltios de salida, por lo tanto, suficiente para alimentar el Arduino.

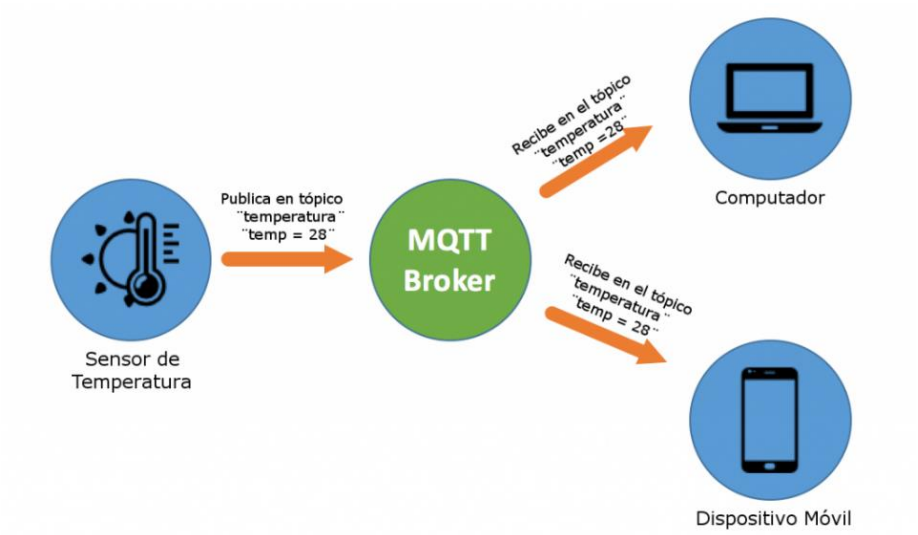
Figura 18. Cable conexión puerto serie (Mini USB - Micro USB)

6.7 Bróker MQTT y cliente publicador

Una vez efectuada la lectura a través del puerto serie, se plantea la publicación de todos estos valores de manera individual a través del estándar de comunicación MQTT. De este modo, cada vez que se efectúe la lectura de unos de los datos, gracias a un identificador que acompañará a cada uno de ellos se podrá organizar mediante el uso de tópicos y sub tópicos y ser publicados para así colocarlos al alcance del que posteriormente será el suscriptor MQTT. Además, gracias a este estándar de comunicación se podría plantear la implementación de varios suscriptores que accedieran a los diferentes tópicos en función de su grado de administración. Por ejemplo, podría haber un suscriptor que únicamente gestionase el estado de las alarmas y su configuración, otro que atendiese a los valores numéricos de cada uno de los datos, etc.

Para hacer posible la publicación mediante el protocolo MQTT se propone el uso de la librería Paho y el lenguaje de programación Python. Así pues, se conseguiría la implementación de un Bróker MQTT, un cliente que suscribiría constantemente los valores obtenidos a través de los sensores y un posible cliente que podría suscribirse a los mismos.

Figura 19. Estructura básica del Bróker MQTT



Por otro lado, dentro de este publicador que recibirá a través del puerto serie una serie de valores numéricos de aspecto abstracto a primera vista, también se plantea la incorporación de la gestión de todos y cada uno de los datos y la interpretación más avanzada. En cierto modo, este programa será el encargado de activar y desactivar las alarmas en función de los rangos de temperatura y humedad aportados por el usuario, así como el tiempo máximo que hará saltar la alerta ocasionada por un periodo elevado en el que el paciente está sentado, etc. Así pues, en este programa se aplicará la inteligencia al sistema, ya que será capaz de interpretar los datos percibidos y hacer saltar las alarmas preconfiguradas.

A continuación, se muestran algunas de las alarmas que vendrían por defecto, sin embargo, todas ellas podrían modificarse fácilmente accediendo al código fuente del sistema:

- Demasiado tiempo en la misma posición: esta alarma actuará cuando el sensor de presión detecte un valor de "1" durante un periodo superior a 60 minutos. El estado por defecto de la alarma será "0" y pasará a "1" cuando esté activa.
- Temperatura de la habitación muy elevada: esta alarma se activará en cualquier momento en el que la temperatura del interior de la habitación supere los treinta grados centígrados.
- Temperatura de la habitación elevada: esta alarma es muy similar a la anterior, con la diferencia de que únicamente se activará cuando la habitación se encuentre por encima de los veintisiete grados centígrados durante un periodo de tiempo superior a diez minutos.
- Temperatura del cojín muy elevada: esta alarma se activará en cualquier momento en el que la temperatura del interior del cojín inteligente supere los treinta y siete grados centígrados.
- Temperatura del cojín elevada: esta alarma únicamente se activará cuando el cojín se encuentre por encima de los treinta y cinco grados centígrados durante un periodo de tiempo superior a diez minutos.
- Humedad elevada: la alarma relacionada con la humedad saltará cuando la humedad del interior del cojín supere el 60%.

Una vez gestionadas todas las alarmas dentro del programa y publicados en el bróker MQTT, queda toda la información organizada según sus correspondientes tópicos. Por ejemplo: *'TFM/dev1/alarm/temp_1'*.

6.8 Cliente suscriptor

Mediante la implementación del cliente suscriptor al bróker MQTT comienza el desarrollo del tercer y último bloque del sistema. En este módulo se encuentra desde el cliente que accederá a cada uno de los tópicos creados en el módulo anterior hasta la interfaz web a la cuál accederá el usuario mediante una URL, pasando por la base de datos en la que se almacenarán todos los valores de los sensores y alarmas y que se irá actualizando constantemente.

Comenzando por el programa suscriptor, este será capaz de reunir la información publicada anteriormente (estado de las alarmas, valores de los sensores, etc.) y situarla de manera ordenada en una base de datos con diferentes tablas dependiendo del sensor que se trate o la alarma que se active. Para esta base de datos se plantea el uso de la herramienta escrita en PHP que permite manejar MySQL a través de sitios web: "*phpMyAdmin*". Esta herramienta permite la creación de bases de datos, tablas, edición de estas y la ejecución de comandos de sentencia SQL. Con la intención de unificar estos servicios con el Sistema Operativo de la Raspberry Pi Zero (Raspbian, distribución del SO GNU/Linux) basado en Debian se opta por la aplicación de un Servidor HTTP como Apache. Este servidor web gratuito aportará al proyecto la posibilidad de servir el contenido de la base de datos al usuario final mediante el desarrollo de una interfaz sencilla escrita en lenguaje de programación HTML.

Todo este conglomerado de Sistema Operativo Linux, Servidor Web Apache, Base de Datos MySQL y el lenguaje de código abierto PHP utilizado para el desarrollo web incrustado en HTML se conoce internacionalmente como LAMP.

Figura 20. Servidor LAMP

Pese a que no han sido creadas expresamente para trabajar entre ellas, estas herramientas conjuntas forman una infraestructura de las más extendidas en el mundo del desarrollo web. Es por ese motivo y porque actualmente es enorme la cantidad de información que se puede encontrar en internet sobre cómo trabajar con ella que se plantea el uso de la pila LAMP.

Una vez introducida la estructura general del tercer y último módulo del sistema, se procede a profundizar en el programa cliente suscriptor que recibirá los datos publicados previamente en el bróker MQTT.

En este programa se propone el uso de la librería "Paho", por supuesto, para poder hacer uso de las funciones necesarias para suscribirse a los tópicos creados, en este caso: 'TFM/#'. Por otro lado, una vez obtenida la información del bróker, es necesario hacerla llegar a la base de datos que se ha creado mediante "*phpMyAdmin*". Para hacerlo posible se plantea la incorporación de la librería "Requests" para HTTP escrita en Python. Esta conocida librería aportará al proyecto la posibilidad de realizar peticiones POST añadiendo una única línea de código tras definir las URL necesarias. Se muestra un ejemplo:

```
import requests
URL1 = http://192.168.1.19/site/temp_ext.php
data = {"Temperatura": "28", "id": "1"}
response = requests.request("POST", URL1, data=data)
```

Así pues, se acaba de realizar una petición POST a la URL definida anteriormente con el contenido de "data" en formato JSON. El destino de la URL se trataría de archivos escritos en PHP que accederían a la base de datos para leerla o modificarla siguiendo la semántica del lenguaje de consultas "*SQL*". Ejemplos INSERT y SELECT:

```
# INSERT INTO ''tabla'' (''columna1'', [''columna2,... '']) VALUES ('
'valor1'', [''valor2,...'']) 

# SELECT Campos FROM Tabla;
```

6.9 Servidor Web

Antes de hablar de la propia base de datos que almacenará la información, es necesario mencionar el propio Servidor Web que hará posible el montaje del sitio web.

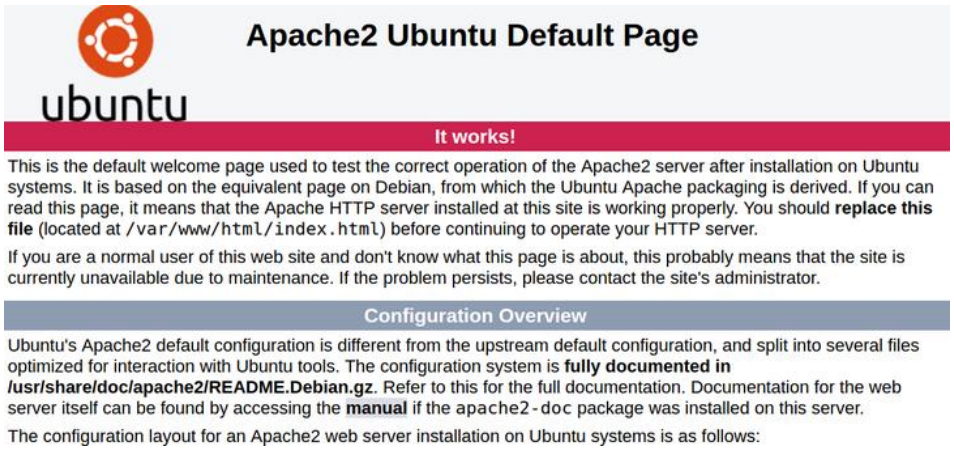
Apache es, hoy en día, el más popular del mundo, posee una enorme cantidad de documentación y es utilizado por multitud de usuarios constantemente a nivel mundial. Es por eso por lo que se plantea el uso de este servidor en concreto.

Para instalar apache basta con hacer uso del paquete *apt* de Ubuntu:

```
$ sudo apt -get update
$ sudo apt -get install apache2
```

Una vez instalado, se podrá comprobar su funcionamiento accediendo a la dirección IP local del dispositivo en el que se encuentre instalado desde el navegador de internet (por ejemplo: Google Chrome), en este caso la Raspberry Pi Zero.

Figura 21. Web predeterminada de Apache y Ubuntu 16.04



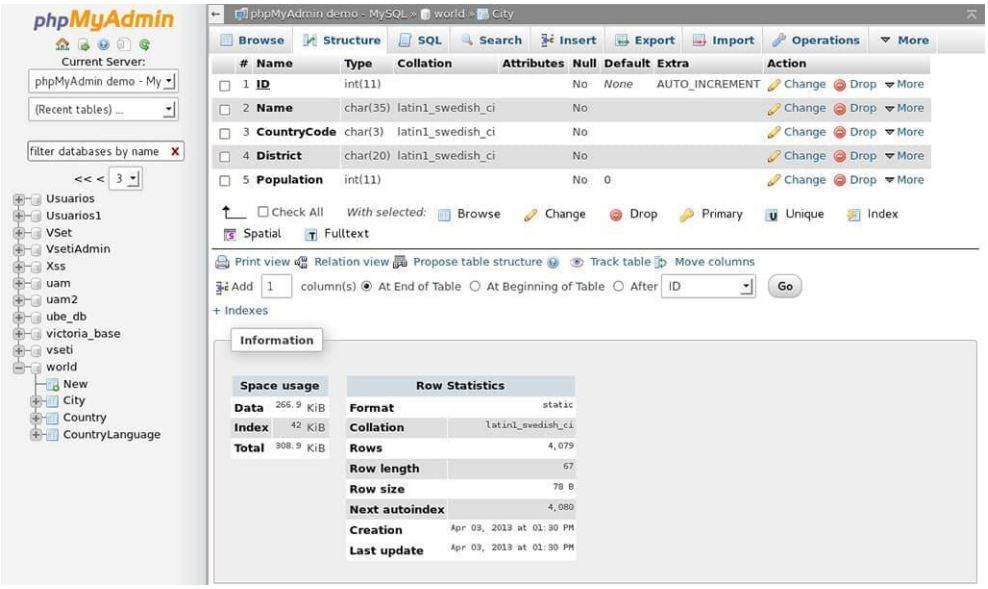
Cuando Apache se encuentre instalado y el servidor esté en marcha no será necesario realizar nada con respecto al mismo, se quedará funcionando en segundo plano para poder continuar con el montaje del resto del sistema.

6.10 Base de datos phpMyAdmin

Con la intención de almacenar, gestionar y poder visualizar los datos que se han ido recopilando previamente (sensorizado del habitáculo, estado de alarmas, etc.) se plantea el uso de la herramienta mencionada en el apartado anterior: "phpMyAdmin".

La implantación de este software al proyecto aportará la posibilidad de trabajar con una interfaz gráfica que proporciona una visualización global del contenido de la base de datos y cada una de las tablas creadas. Así como la facilidad de crear tablas y configurarlas mediante varios sencillos pasos. Además, una vez el sistema se encuentre en marcha. Su sitio web permitirá acceder a la base de datos mediante un usuario y contraseña especificados en su configuración para observar en tiempo real el estado general de la base de datos.

Figura 22. Interfaz gráfica del software phpMyAdmin



En cuanto a la configuración de la base de datos utilizada en el proyecto, posteriormente se incluyen más detalles sobre la misma.

6.11 Procesador PHP

El procesador PHP será la última herramienta necesaria para culminar el montaje del sistema y poder mostrar al usuario final una interfaz sencilla que pueda interpretar con un simple vistazo, sin necesidad de buscar valores en tablas de bases de datos de cientos de celdas. Para hacer esto posible se propone el uso de lenguaje PHP incrustado en HTML. Por lo tanto, es necesario un procesador de código que muestre el contenido dinámico.

Se propone el uso de lenguaje PHP ya que este es capaz de ejecutar secuencias de comandos, acceder a las bases de datos creadas en MySQL para visualizar su contenido y hacerlo llegar hasta el sitio web que abrirá el usuario.

Para instalar PHP en la Raspberry Pi Zero basta con hacer uso de un par de líneas de comandos a través de la terminal del dispositivo. Posteriormente se verán más detalles acerca de su instalación.

7. PROGRAMACIÓN

7.1 Introducción

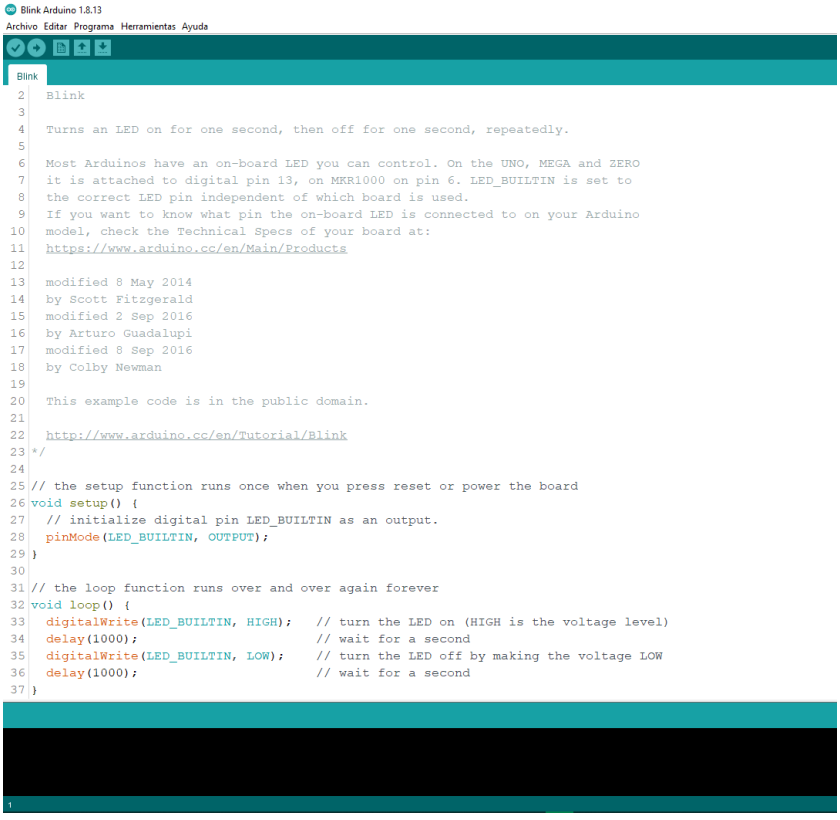
En este apartado de la memoria se presenta en profundidad la programación necesaria para unificar todas las herramientas que se han utilizado en el proyecto.

Para realizar una programación lo más organizada posible, se estructurará según dispositivos o lenguajes de los propios programas. Por lo tanto, se comenzará por la programación de los microcontroladores Arduino Nano, posteriormente la configuración previa que requerirá la Raspi, los archivos publicador y suscriptor escritos en Python y finalmente los archivos con sentencia PHP y SQL.

7.2 Arduino

A continuación, se procede a explicar por completo el desarrollo del código que contendrán ambos microcontroladores Arduino Nano. Para empezar, es conveniente descargar e instalar en un ordenador el Software de Arduino de código libre (IDE) para facilitar así la subida del código a los dispositivos utilizados. Este código se encuentra en la página web oficial de la empresa: https://www.arduino.cc/en/Main/software.

Figura 23. Interfaz de desarrollo de Arduino



Antes de empezar a programar es necesario saber que el lenguaje de programación de Arduino está basado en C++ por lo que se trata de un lenguaje

de nivel medio, además, debido a la implementación de librerías externas, se facilita de gran modo la programación que se va a configurar.

Para llevar a cabo la programación del dispositivo emisor al cual estarán conectados todos los sensores del cojín, se comienza incluyendo las librerías mencionadas en el apartado anterior:

```
#include <VirtualWire.h>
#include "DHT.h"
#include "Adafruit_MCP9808.h"
```

Se define el sensor DHT22, el pin de emisión RF (dataPin), el pin de lectura analógica del sensor de presión (presPin) y el pin de lectura analógica del sensor DHT22 (humPin). Por otro lado, se crea una variable de tipo Adafruit_MCP9808 para efectuar la lectura del sensor de temperatura de la habitación y se vincula el pin del sensor DHT22:

```
#define DHTTYPE DHT22
const int dataPin = 9;
const int presPin = 15;
const int humPin = 5;
Adafruit_MCP9808 tempsensor = Adafruit_MCP9808();
DHT dht(humPin, DHTTYPE);
```

La función *setup* se encarga de llevar a cabo la configuración inicial indicada en el interior de esta:

```
vw_setup(2000);
vw_set_tx_pin(dataPin);//Configura el pin de emisión RF
pinMode(presPin, INPUT);//Configura el pin como entrada

if (!tempsensor.begin(0x18)) {//Gestión de error en el sensor
    Serial.println("Sensor MCP9808 no encontrado.");
    while (1);
  }
tempsensor.setResolution(1);//Ajustar resolución del MCP9808
tempsensor.wake();//Despertar el sensor MCP9808
dht.begin();//Despertar el sensor DHT22
```

Una vez aplicada la configuración inicial, se procede con la lectura de los sensores y la emisión de datos mediante radiofrecuencia. Para ello, tras crear las variables necesarias, se efectúa la lectura de los sensores sobre las mismas:

```
String strtemp, strpres, strhum, strtemp2;//String a enviar
char buf[VW_MAX_MESSAGE_LEN];//Longitud del mensaje a enviar
int pres;//Variable tipo entero, indicador de presion
float temp, hum, temp2;//Variables tipo float restantes
```

```
temp = tempsensor.readTempC();//Lectura de la temperatura de
la habitación
strtemp = "t" + String(temp); // Convertir a String con el
primer caracter identificador
strtemp.toCharArray(buf, sizeof(buf)); // Convertir a char
array
vw_send((uint8_t *)buf, strlen(buf)); // Enviar array
vw_wait_tx(); // Esperar envio

int pres_ = analogRead(presPin); //Lectura valor entrada
sensor de presion (0 - 1023)
if (pres_ > 512){//Umbral de detección del sensor
  pres = 1;
}
else{
  pres = 0;
}
strpres = "p" + String(pres);  //String con identificador
strpres.toCharArray(buf, sizeof(buf));
vw_send((uint8_t *)buf, strlen(buf));
vw_wait_tx();

hum = dht.readHumidity();//Lectura del valor de humedad del
cojin
strhum = "h" + String(hum);// String con identificador
strhum.toCharArray(buf, sizeof(buf));
vw_send((uint8_t *)buf, strlen(buf));
vw_wait_tx();

temp2 = dht.readTemperature();// Lectura del valor de
temperatura interior del cojin
strtemp2 = "g" + String(temp2);//String con identificador
strtemp2.toCharArray(buf, sizeof(buf));
vw_send((uint8_t *)buf, strlen(buf));
vw_wait_tx();
```

Finalmente se añade un delay de 200 milisegundos para no saturar al sistema, ya que no es necesaria una lectura más rápida de 5 veces por segundo.

```
delay(200);
```

Una vez completada la programación del primer dispositivo se procede a conectar el microcontrolador mediante la conexión USB al ordenador y se pulsa el botón de "subir" identificado con una flecha en horizontal en la parte superior de la interfaz.

En cuanto al segundo microcontrolador, receptor de toda esta información, su programación es la siguiente:

Primero, como anteriormente, es necesario incluir la librería "VirtualWire.h". Seguidamente, declarar el pin de lectura radiofrecuencia (dataPin):

```
#include <VirtualWire.h>
const int dataPin = 9;
```

Después, en la configuración inicial se incluyen las mismas líneas necesarias para trabajar con el módulo RF 433MHz:

```
Serial.begin(9600);//Utilizando la salida por defecto del
puerto serie se puede pasar la información fácilmente a la
Raspberry Pi
vw_setup(2000);
vw_set_rx_pin(dataPin);//Pin de lectura RF
vw_rx_start();//Despierta la lectura
```

A continuación, se muestra cómo se realiza una lectura del valor de temperatura de la habitación. Este método se aplicará del mismo modo a todos los valores que se quieran leer. Únicamente es necesario modificar el identificador que se quiera distinguir:

```
if   (vw_get_message((uint8_t   *)buf,&buflen)){//Evento   al
recibir un mensaje RF
  String lectura;
  if((char)buf[0]=='t'){//Busqueda del identificador
    for (int i = 1; i < buflen; i++){//Lectura de todos los
caracteres
      lectura.concat((char)buf[i]);
    }
    float temp = lectura. toFloat ();  // Convertir a float
    Serial.println(temp);//Impresion  por  puerto  serie  del
valor en variable float
}
```

Por lo tanto, posteriormente se añadirán los tres bloques precedidos de "else if" para el resto de los valores necesarios.

7.3 Configuración inicial Raspberry Pi Zero

7.3.1 Sistema Operativo

El sistema operativo que se va a utilizar en la Raspberry Pi Zero es el actualmente conocido como *Raspberry Pi OS (32-bit) Lite* (anteriormente conocido como Raspbian Lite). Existe otra versión más pesada que contiene la capacidad de abrir un escritorio mediante una pantalla con conexión HDMI, sin embargo, en este caso se accederá al contenido del dispositivo a través de una red local, por lo tanto, no se necesita una versión tan pesada.

Este Sistema Operativo se puede descargar desde la página web del fabricante: https://www.raspberrypi.org/downloads/raspberry-pi-os/ . Una vez descargado será necesaria la utilización de un software para crear un USB de arranque. Existen varias opciones disponibles en internet y no es demasiado importante el uso de una u otra. El objetivo es crear una tarjeta de memoria micro USB de arranque con el Sistema Operativo recientemente descargado.

Una vez creada la memoria de arranque, antes de introducirla en la Raspberry Pi Zero, ya que esta no posee una conexión Ethernet y es necesario conectarla a la red local, es menester acceder a la memoria para editar el archivo llamado "*wpa_supplicant.conf*". Se deben buscar las siguientes líneas dentro del mismo y modificarlas con los datos de la red y su contraseña:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=<Insert 2 letter ISO 3166-1 country code here>
network={
 ssid="<Name of your wireless LAN>"
 psk="<Password for your wireless LAN>"
}
```

Para poder acceder al dispositivo mediante "*ssh*" se crea un archivo vacío con el nombre "*ssh*" en el mismo directorio en el que se encuentra el archivo recientemente modificado. Posteriormente, se introduce la tarjeta de memoria en la ranura de la Raspi y se conecta mediante el cable proporcionado para arrancar el sistema. El dispositivo se conectará a la red local mediante Wi-Fi y obtendrá una dirección IP (es recomendable establecer una IP estática desde la configuración del rúter para facilitar el acceso durante su configuración).

7.3.2 Actualización del sistema

En estos momentos, ya es posible acceder al sistema desde un terminal conectado a la misma red local mediante el comando "*ssh*" con el usuario y contraseña establecidos por defecto. Usuario: "*pi*". Contraseña: "*raspberry*". Por lo tanto, se abre un terminal local y se ejecutan los siguientes comandos:

```
ssh pi@192.168.1.19
>>raspberry
pi@raspberrypi:~ $ sudo passwd root
pi@raspberrypi:~ $ 1234
pi@raspberrypi:~ $ 1234
pi@raspberrypi:~ $ sudo su
root@raspberrypi:/home/pi# cd
```

Así pues, se configura la contraseña del usuario root a "1234" para tener acceso como administrador. Se actualiza el sistema:

```
root@raspberrypi:~# apt-get update
root@raspberrypi:~# apt-get upgrade
```

Instalación de Python 3:

```
root@raspberrypi:~# apt install python3-picamera
```

7.3.3 Instalación Servidor LAMP

```
root@raspberrypi:~# apt install apache2
```

Acceder desde el navegador a la IP del dispositivo para comprobar su instalación correcta. Debe aparecer la página web predeterminada de Apache.

```
>>http://192.168.1.19
```

Se instala el procesador PHP y se lleva a cabo un *Hola Mundo*:

```
root@raspberrypi:~# apt install php
root@raspberrypi:~# nano /var/www/html/test.php
<?php echo "HOLA MUNDO!!!"; ?>
root@raspberrypi:~# service apache2 restart
```

Acceder desde el navegador a la IP del dispositivo y ubicación del archivo para comprobar su instalación correcta.

```
http://192.168.1.19/test.php
```

Instalación de MySQL y creación de nuevo usuario con privilegios:

```
root@raspberrypi:~# apt install mariadb-server php-mysql
```

```
root@raspberrypi:~# service apache2 restart
root@raspberrypi:~# mysql_secure_installation
root@raspberrypi:~# mysql --user=root --password
> create user admin@'%' identified by '1234';
> grant all privileges on *.* to admin@'%';
> FLUSH PRIVILEGES;
> exit;
```

Instalación de phpMyAdmin y enlace con MySQL:

```
root@raspberrypi:~# apt install phpMyAdmin
```

Al aparecer la ventana de diálogo, seleccionar la opción de Servidor Apache 2. Más tarde configurar una contraseña de inicio de sesión.

```
root@raspberrypi:~# phpenmod mysqli
root@raspberrypi:~# service apache2 restart
```

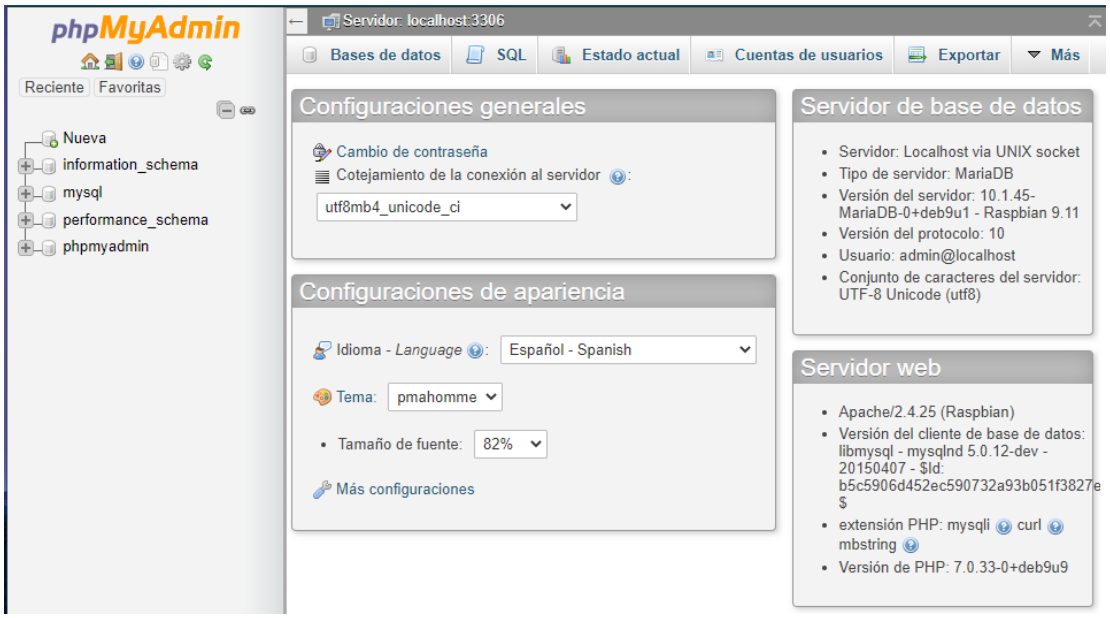
En este momento se debe modificar el archivo de configuración de MySQL para permitir conexiones en remoto:

```
root@raspberrypi:~#     nano     /etc/mysql/mariadb.conf.d/50-
server.cnf
```

Buscar la línea que contenga la "bind-address" y cambiarla por 0.0.0.0. Posteriormente, abrir la siguiente URL desde un navegador para configurar la base de datos:

```
>>http://192.168.1.19/phpmyadmin/
```

Figura 24. Interfaz web de phpMyAdmin



7.3.4 Configuración phpMyAdmin

Para crear la base de datos que posteriormente se utilizará en el proyecto se debe pulsar sobre el icono de "Nueva" situado a la izquierda de la pantalla y crear las siguientes tablas:

- hum_cojin: Almacena los valores del sensor de humedad (DHT22).
- hum_alarm: Almacena el estado de la alarma relacionada con la humedad del cojín. La humedad del cojín es superior al rango establecido.
- pres: Almacena el valor del sensor de presión (0 - 1).
- pres_alarm: Almacena el estado de la alarma relacionada con el sensor de presión. El paciente lleva demasiado tiempo sentado.
- temp_ext: Almacena el valor del sensor de temperatura MCP9808. Temperatura de la habitación.
- temp_alarm_1: Almacena el estado de la primera alarma relacionada con el sensor de temperatura de la habitación. Temperatura superior a 30º C.
- temp_alarm_2: Almacena el estado de la segunda alarma relacionada con el sensor de temperatura de la habitación. Temperatura superior a 27º C durante más de 10 minutos.
- temp_cojin: Almacena el valor del sensor de temperatura DHT22. Temperatura del cojín.
- temp2_alarm_1: Almacena el estado de la primera alarma relacionada con el sensor de temperatura del cojín. Temperatura superior a 37º C.
- temp2_alarm_2: Almacena el estado de la segunda alarma relacionada con el sensor de temperatura del cojín. Temperatura superior a 35º C durante más de 10 minutos.

Cada una de las tablas tendrá dos columnas. La primera llamada "id" y será auto rellenable, para conocer el número de valores obtenidos. La segunda, "value", en la que se almacenará el propio valor de cada una de las variables (sensores y alarmas).

7.4 Ficheros Python

7.4.1 Publicador MQTT

Primero, se va a presentar la programación del archivo publicador MQTT. Este fichero sirve para recibir los datos transmitidos a través del puerto serie entre el Arduino Nano receptor y la Raspberry Pi Zero. Utiliza el lenguaje de programación Python, acompañado de la librería "paho".

Para comenzar, se importan las librerías necesarias para su desarrollo. En este caso, "serial" permite la capacidad de leer a través del puerto serie, "time" para aportar el uso de contadores, "datetime" permite conocer la fecha actual y "paho.mqtt.publish" se utilizará para publicador la información en el bróker MQTT.

```
import serial
import time
from datetime import datetime
import paho.mqtt.publish as publish
```

Posteriormente, tras la configuración de la entrada por puerto serie, se configuran los valores que efectuarán el cambio entre activa e inactiva de las alarmas creadas. También se almacena en una variable la fecha actual.

```
data = serial.Serial('/dev/ttyUSB0', 9600)
date = datetime.now()

temp_value_1 = 30
temp_value_2 = 27
temp_time = 30 #segundos
temp_alarm_1 = 0        #ALARMA 1 ACTIVA
temp_alarm_2 = 0        #ALARMA 2 ACTIVA
temp_now_1 = int(round(time.time()))

hum_value = 60
hum_alarm = 0           #ALARMA ACTIVA

temp2_value_1 = 37
temp2_value_2 = 35
temp2_time = 30 #segundos
temp2_alarm_1 = 0        #ALARMA 1 ACTIVA
temp2_alarm_2 = 0        #ALARMA 2 ACTIVA
temp2_now_1 = int(round(time.time()))

pres_time = 25 #segundos
pres_alarm = 0           #ALARMA ACTIVA
```

Tras una breve bienvenida al inicio del sistema, aunque esta probablemente nunca se aprecie si no se conecta el dispositivo a una pantalla que permita leerla, comienza el código principal. Este código se ejecutará en bucle sin ningún tipo de excepción. Esto es así para hacer el programa lo más sencillo posible. Un dispositivo que: lee por puerto serie, identifica la variable leída, ajusta el valor de las alarmas en función de unos parámetros especificados previamente y publica toda esta información en un bróker MQTT.

```
lectura = data.readline() #Lectura del Puerto Serie

if lectura[0] == 'p': #Identificador variable presión
  pres = int(lectura[1])
  if pres == 1 & pres_alarm == 0: #GESTION ALARMA PRESION
    pres_now_1 = int(round(time.time()))
  if pres == 1:
    pres_now_2 = int(round(time.time()))
    pres_time_ = pres_now_2 - pres_now_1
    if pres_time_ > pres_time:
      pres_alarm = 1
  if pres == 0:
    pres_alarm = 0

  print ("Sentado: " + str(pres))
  print ("Alarma: " + str(pres_alarm))
  print ("--------------------")

  try:
    publish.single('TFM/dev1/pres',    pres,    hostname    =
'localhost') #Publicacion de valor y alarmas de presion
    publish.single('TFM/dev1/alarm/pres',          pres_alarm,
hostname = 'localhost')
  except:
    print("Exception ocurred: Publish error!")
```

El código recientemente mostrado se trata de una sentencia sencilla que identifica la variable leída según su primer carácter (explicado previamente) para así, almacenarla en una variable de tipo entero (0 o 1). Según la activación o no de esta variable, se almacenará el tiempo que permanece en activo para así, si supera el valor establecido previamente en "pres_time" poder activar la alarma "pres_alarm".

Tras hacer esta comparación y activar o desactivar la alarma relacionada, se hace uso del "*try and except*" para intentar publicar el valor de la variable y de la alarma en sus tópicos correspondientes. En este caso, el valor del sensor de presión se publica en *'TFM/dev1/pres'* y el valor de la alarma en *'TFM/dev1/alarm/pres'*.

Así pues, este bloque se multiplica por el número de sensores y alarmas asociadas a los mismos.

7.4.2 Suscriptor MQTT

En cuanto a la programación del suscriptor MQQT encargado de adquirir toda la información publicada en el bróker creado, se plantea un código fuente todavía más sencillo que el anterior. Para este fichero se plantea la incorporación de las librerías "*paho*" y "*requests*" escritas en Python. La librería "*requests*" aporta la posibilidad de realizar una petición POST al servidor web creado para montar la interfaz del usuario final.

Por lo tanto, en las primeras líneas se añaden las librerías pertinentes y se declaran las URL en las cuales se realizarán las peticiones:

```
import ssl
import sys
import paho.mqtt.client
import requests

URL1 = "http://192.168.1.19/site/temp_ext.php"
URL2 = "http://192.168.1.19/site/temp_cojin.php"
URL3 = "http://192.168.1.19/site/hum_cojin.php"
URL4 = "http://192.168.1.19/site/pres.php"
URL5 = "http://192.168.1.19/site/temp_alarm_1.php"
URL6 = "http://192.168.1.19/site/temp_alarm_2.php"
URL7 = "http://192.168.1.19/site/temp2_alarm_1.php"
URL8 = "http://192.168.1.19/site/temp2_alarm_2.php"
URL9 = "http://192.168.1.19/site/pres_alarm.php"
URL10 = "http://192.168.1.19/site/hum_alarm.php"
```

En estas ubicaciones se encuentran los archivos escritos en sentencia PHP que se explicarán en el siguiente apartado.

A continuación, tras inicializar el valor a "0" de las alarmas, se crea la función "*on_connect*", la cual se ejecutará cuando establezca conexión con el bróker. Esta función imprime un mensaje de confirmación para indicar esta conexión y el cliente. Finalmente, se suscribe al tópico deseado, en este caso: *'TFM/#*'.

```
def on_connect(client, userdata, flags, rc):
    print('connected (%s)' % client._client_id)
    client.subscribe(topic='TFM/#', qos=2)
```

Después, del mismo modo, se declara la función "*on_message*". Esta función se llamará cada vez que se reciba un mensaje sobre los tópicos a los que el sistema está suscrito. Se imprime el mensaje por pantalla:

```
def on_message(client, userdata, message):
     print('topic: %s' % message.topic)
     print('payload: %s' % message.payload)
     print('qos: %d' % message.qos)
```

Una vez recibido un mensaje, se comprueba el tópico del cual proviene y se realiza la petición "*POST*" sobre la URL correspondiente. En este caso, puesto que el tópico corresponde con el valor del sensor de temperatura de la habitación, se realiza la petición "*POST*" sobre la "*URL1"*. Esta ruta corresponde con la ubicación del fichero "*temp_ext.php*" que contiene la temperatura exterior en sentencia PHP.

```
if message.topic == 'TFM/dev1/temp':
  data = {
      "Temperatura": message.payload
      }
  response = requests.request("POST", URL1, data=data)
```

Finalmente, se define el "*main*" del programa. En él, se declara el cliente y se llama a la declaración de las funciones "*on_connect*" y "*on_message*":

```
client      =      paho.mqtt.client.Client(client_id='valero',
clean_session=False)
client.on_connect = on_connect
client.on_message = on_message
client.connect(host='localhost', port=1883)
client.loop_forever()
```

7.5 Ficheros PHP

En este apartado se describe la programación necesaria para presentar la información que se posee hasta el momento en la interfaz web a la que accederá el usuario final. La estructura está compuesta de un archivo que proporcionará la configuración necesaria al resto, diferentes archivos PHP para cada uno de los valores de los sensores y las alarmas y el archivo que contendrá la propia interfaz (FRONTEND).

7.5.1 Config.php

Este archivo es común para todos los demás ya que contiene las credenciales para acceder a la base de datos creada previamente.

```
<?php
// Credenciales necesarias para acceder a phpMyAdmin
$dbhost="localhost";
$dbuser="admin";//Nombre de usuario
$dbpass="1234";//Contraseña
$dbname="Prueba_cojin";//Nombre de la base de datos

//Establecer conexión con la base de datos
$con=mysqli_connect($dbhost, $dbuser, $dbpass, $dbname);
?>
```

7.5.2 Temp_ext.php

Por otro lado, los diez archivos correspondientes a las URL creadas en el programa suscriptor. Estos archivos reciben la petición "*POST*" y la almacenan en una variable para, a través de una "*query"*, guardarlos en la tabla correspondiente de la base de datos: (Ejemplo: *temp_ext.php* – Temperatura de la habitación)

```
<?php
require("config.php");
$Temperatura=mysqli_real_escape_string($con,
$_POST['Temperatura']);
$query="insert into temp_ext (value) values ($Temperatura)";
$result=mysqli_query($con,$query);
?>
```

Mediante estas líneas de código escritas en PHP, se llama a la configuración previamente establecida, se almacena el valor de la variable Temperatura y se inserta en la tabla "*temp_ext*", en la columna "*value*" de la base de datos.

7.5.3 Values.php

El último paso del proceso consta de un archivo de sentencia PHP incrustada en HTML. Esto es así para facilitar el diseño de la propia interfaz por medio del uso de tablas. También se plantea el uso de JavaScript para aportar dinamismo al sitio web. Es importante recalcar que este sitio web será el aspecto físico de todo el sistema, ya que será lo único que verá el cliente a diario mientras haga uso del dispositivo.

En cuanto a la programación de esta, no es de gran relevancia. Por lo que no se va a indagar demasiado en su explicación. Únicamente recalcar que del mismo modo que anteriormente se ha escrito sobre determinadas tablas de la base de datos, esta vez se lee de la misma para presentar el último valor obtenido por pantalla.

```
<?php
require("config.php");//Configuración necesaria
$query="Select value From temp_ext";//Seleccion del valor
$datos=array();//Alamecenamiento en array
$rs=mysqli_query($con, $query);//Ejecución de la query
mysqli_close($con);

while ($row=mysqli_fetch_object($rs)){
    $datos[]=$row;
}
$values = sizeof($datos);
$obj_ = json_encode($datos[$values-1]);
$temp = json_decode($obj_);//Ejemplo con valor de temp
?>
<?php
```

En este momento, se obtiene el último valor de la temperatura de la habitación (*temp*) en una variable para ser representado de cualquier modo al usuario. En este caso se ha optado por formar una tabla sencilla que muestre todos los valores de los sensores y las alarmas. Tal y como se muestra a continuación:

Figura 25. Ejemplo del posible diseño de la UI

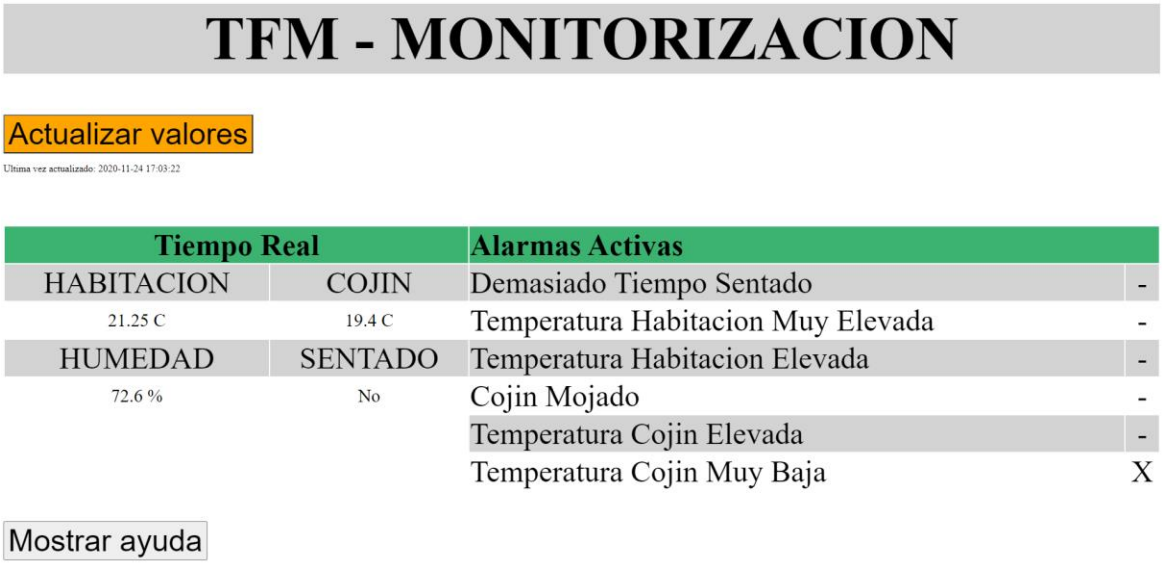
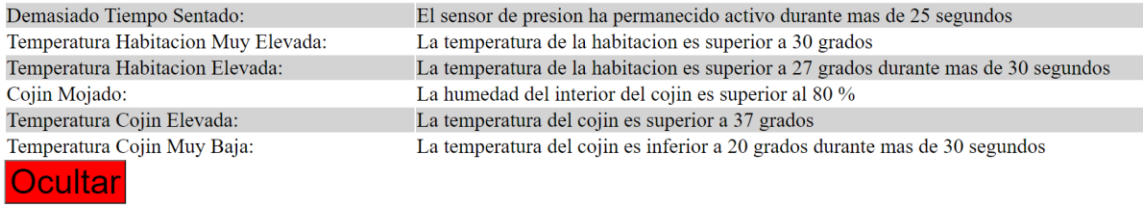


Figura 26. Ayuda interfaz web



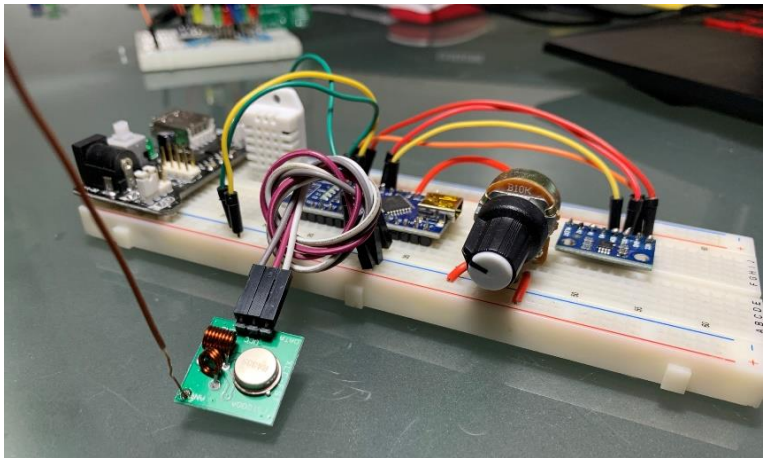
8. ANÁLISIS Y DISCUSIÓN DE RESULTADOS

En el siguiente apartado se presentan los resultados obtenidos al poner en práctica todos los conceptos teóricos constituidos a lo largo de la memoria. Desde el montaje completo del hardware mencionado, su calibración y ajuste para un funcionamiento apropiado hasta la programación completa de los ficheros necesarios a lo largo del sistema, pasando por la configuración de la Raspberry Pi Zero como bloque fundamental del proyecto.

8.1 Hardware

Comenzando por el "Cojín Inteligente", para el primer diseño se plantea el uso de una placa de prototipos hasta conseguir un funcionamiento adecuado, así como una programación funcional. De este modo, se tiene la posibilidad de realizar modificaciones de manera rápida y más sencilla que si se soldaran todos los componentes directamente. Por otro lado, el sensor de presión se simula mediante la implementación de un potenciómetro que hace una función similar a la hoja conductiva utilizada en el proyecto final.

Figura 27. Primer prototipo Cojín Inteligente



Una vez obtenidos los resultados deseados, se diseña mediante el software de SolidWorks una carcasa para posteriormente imprimirla en 3D con plástico PLA. De este modo, se consigue un resultado más profesional y duradero, así como resistente a sacudidas o golpes. También se sustituye el potenciómetro con la hoja conductiva. La carcasa contendrá una pequeña ranura para introducir el cable USB que alimentará el microcontrolador. Para obtener un dispositivo "autónomo" temporalmente se plantea el uso de una batería externa portátil y recargable que dará al cojín varias horas de vida. En un diseño mejorado se plantea el uso de baterías de menor tamaño y mayor capacidad para alargar su vida y disminuir su volumen dentro del cojín. El resultado es el siguiente:

Figura 28. Diseño 3D carcasa Emisor RF

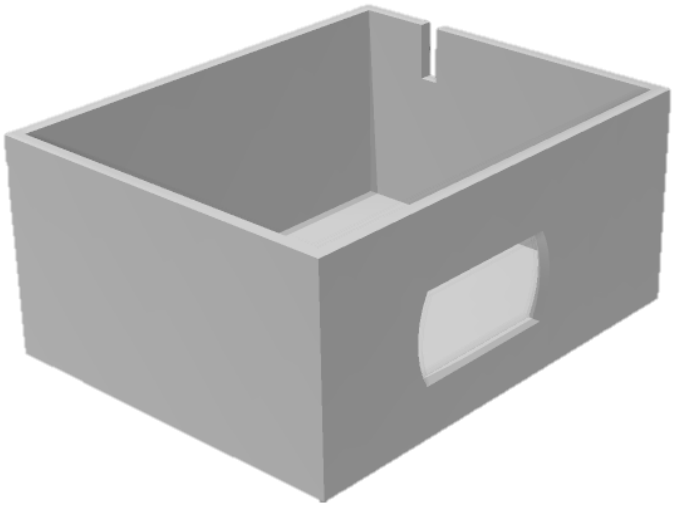
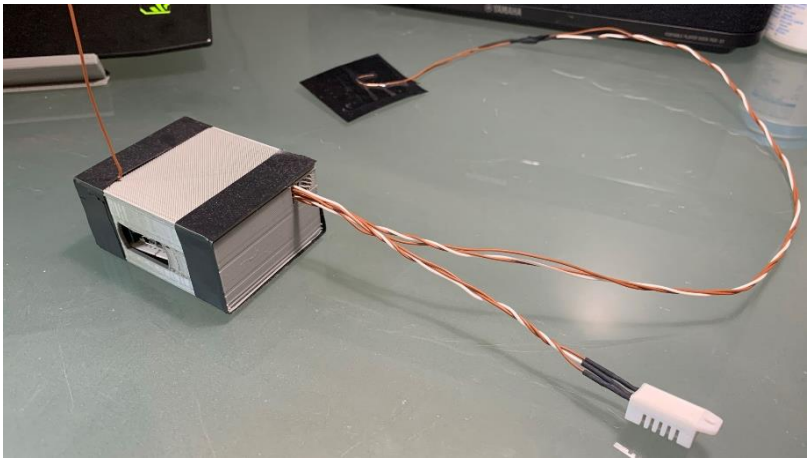
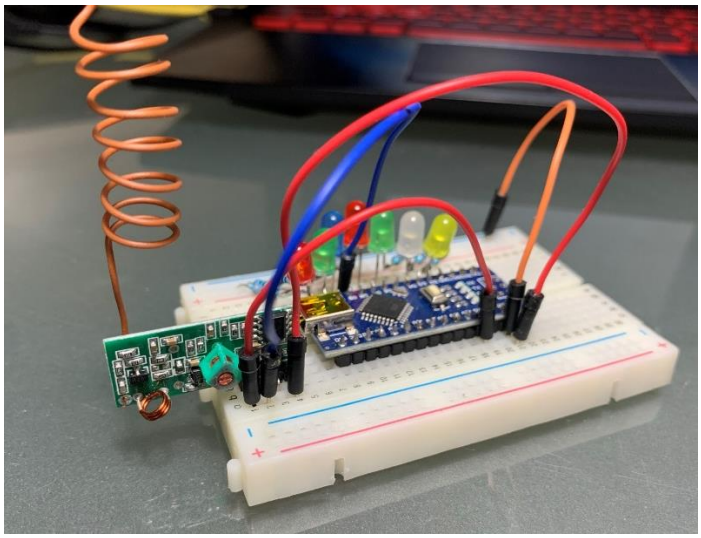


Figura 29. Segundo prototipo Cojín Inteligente



En cuanto al dispositivo receptor radiofrecuencia que será capaz de recibir información de los diferentes cojines situados en el interior de la habitación. También se ha llevado a cabo un primer prototipo de características mínimas para comprobar el funcionamiento de este durante la fase de desarrollo. En este primer prototipo, el dispositivo únicamente consta del propio microcontrolador conectado mediante puerto serie a un ordenador, en el cual imprime por pantalla la información recibida por medio del sistema radiofrecuencia. Posteriormente se plantea la implementación de indicadores LED que simulan el valor de alguno de los sensores en función de varios rangos establecidos.

Figura 30. Primer prototipo Receptor RF



Estas luces LED se encenderán según el valor leído de los sensores conectados al Cojín Inteligente. Para hacer uso de esta función de representación mediante luces LED, por el momento solo es posible interpretar los valores de uno de los cojines. Sin embargo, la incorporación de un mayor número de luces aportaría la posibilidad de interpretar el valor de los sensores del número de cojines deseado.

Una vez obtenido un funcionamiento adecuado del receptor, se ha diseñado una carcasa mediante SolidWorks y ha sido impresa en PLA mediante una impresora 3D para recoger así de una manera más segura y organizada el dispositivo.

Figura 31. Diseño 3D carcasa Receptor RF

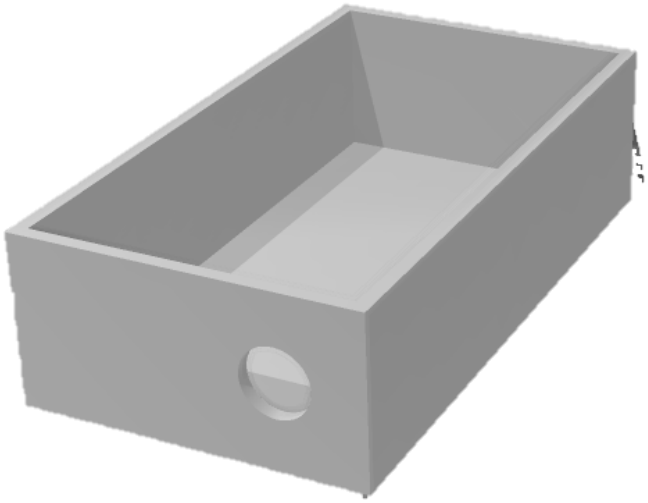


Figura 32. Segundo prototipo Receptor RF

A la hora de comparar los resultados obtenidos con los resultados esperados, es importante recalcar varias situaciones que no resultaron como se esperaba y que por lo tanto ha habido que plantear soluciones alternativas durante la investigación. Entre ellas se destaca:

- Alcance emisor - receptor RF. Pese a que el fabricante de los dispositivos emisor y receptor radiofrecuencia habla de un alcance aproximado de entre 15 y 20 centímetros sin ninguna incorporación de antena, el resultado es un alcance absolutamente nulo. El dispositivo no era capaz de emitir satisfactoriamente ninguna información al receptor pese a encontrarse casi en contacto. Esto ha ocasionado un gran número de horas perdidas debido a que se pensaba que el fallo pudiera estar en la programación de los microcontroladores. Finalmente, se optó por el uso de un cable de telefonía de cobre. Este cable es capaz de amplificar la señal hasta varios metros de distancia. Su coste es prácticamente nulo y su funcionamiento irregular, sin embargo, debido a que el cojín se encuentra en una posición estática, su funcionamiento es más que suficiente. Se plantea la incorporación de antena radiofrecuencia más potente para diseños futuros.

- Temperatura interior del cojín. Al inicio del proyecto se planteaba el uso de un único sensor de temperatura y humedad para conocer tanto el estado de la habitación como el del interior del cojín. A medida que se avanzó en el trabajo se concluyó que era necesario un segundo sensor que midiera únicamente la temperatura de la habitación mientras el sensor de temperatura y humedad serviría para conocer solamente el estado del interior del cojín. Esto resultó ser efectivo, ya que, desde este momento se obtenían dos valores independientes de ambas temperaturas (interior y exterior del cojín). Se pretendía utilizar, además, la temperatura interior

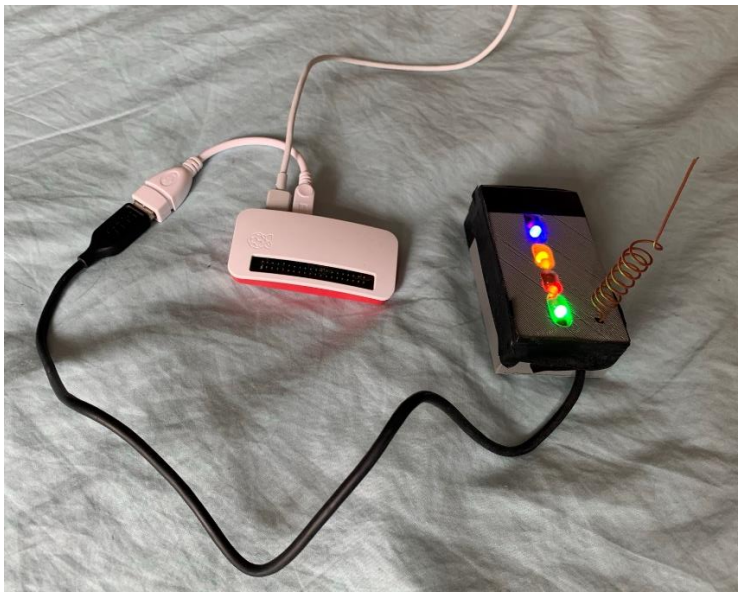
para conocer de manera aproximada el valor de la temperatura corporal del paciente, sin embargo, este valor no se aproxima demasiado a la realidad y no parece resultar útil en cuanto a la monitorización del paciente. Por otro lado, se obtiene un valor de temperatura del cojín que puede ser utilizado para conocer el estado térmico de este.

Por último, al conectar el receptor RF mediante el cable USB a la Raspberry Pi se obtiene el sistema completo en cuanto a Hardware se refiere. El resultado obtenido al aplicar este método de transmisión de datos (puerto serie) es óptimo, ya que, la velocidad es muy alta y constante, evitando así cualquier pérdida de datos tal y como sucede mediante radiofrecuencia. Además, como ya se ha mencionado anteriormente, alimentando la Raspberry mediante un transformador de 5 voltios, se alimenta también el microcontrolador Arduino, por lo que no es necesario el uso de ninguna batería portátil. Esto se debe a que este dispositivo está diseñado para permanecer de manera estática en cualquier punto de la habitación cercano a los cojines inteligentes. El cable de alimentación es el único necesario ya que se conecta al rúter mediante Wi-Fi como se explica en el apartado de programación.

Figura 33. Cojín inteligente semimontado

Figura 34. Cojín inteligente emisor

Figura 35. Sistema receptor



Vistos los resultados obtenidos se puede comprobar que aportan un funcionamiento suficiente al sistema como para trabajar conjuntamente, no obstante, es importante remarcar que los recursos durante el desarrollo del trabajo han sido bastante limitados. No se ha utilizado ningún laboratorio dedicado a este tipo de proyectos ni recursos superiores a un ordenador de características estándar y un soldador de estaño. El único elemento que quizá no se encuentre en cualquier espacio de trabajo y que se ha utilizado es una impresora 3D (Ender 3 Pro-S) de baja gama.

El hecho de que haya sido desarrollado en un espacio de trabajo convencional es algo muy importante teniendo en cuenta los resultados obtenidos. Esto indica que la proyección del dispositivo es enorme.

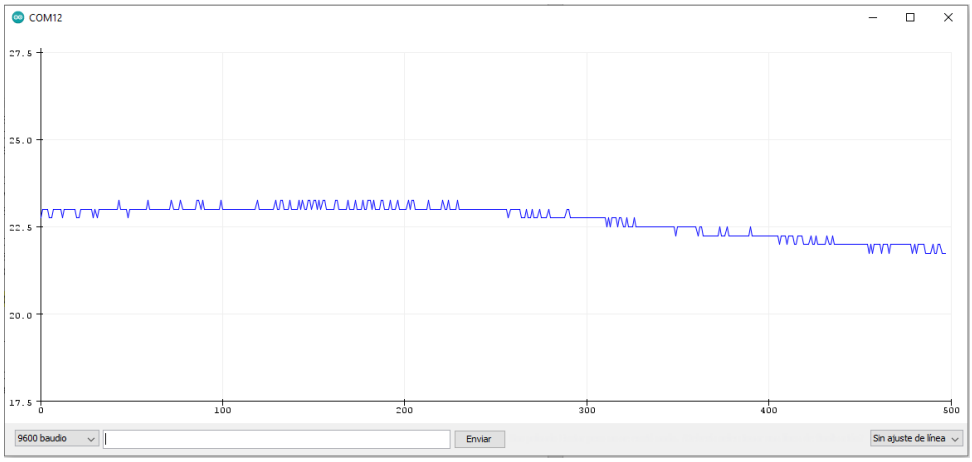
8.2 Software

En cuanto a los resultados obtenidos del software implementado, para comprobar la solidez y estabilidad de este, se ha propuesto una prueba de funcionamiento en una persona durante un periodo de tiempo. A continuación, se muestra lo obtenido.

8.2.1 Arduino emisor

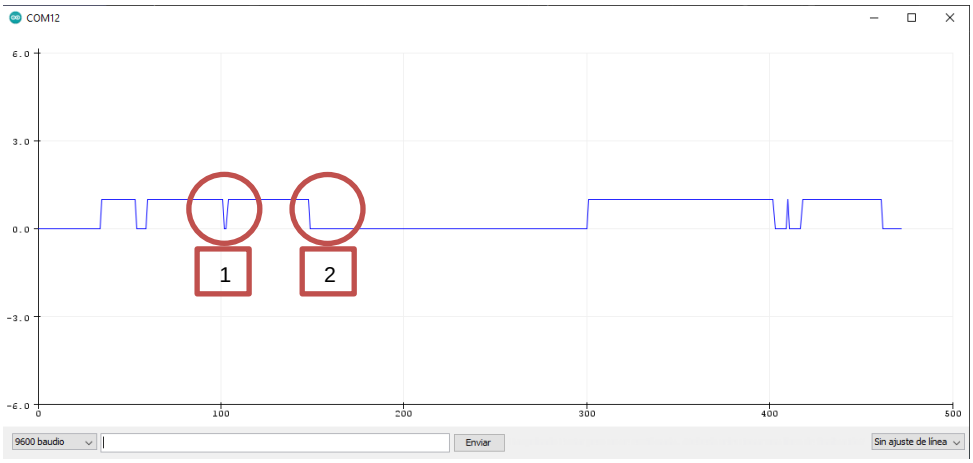
En la primera gráfica se observa la recogida de valores de temperatura de la habitación en la cual se encuentra el cojín inteligente. Se puede comprobar como la temperatura es bastante estable cerca de los 23 º C. Tras abrir una ventana en el habitáculo, la temperatura comienza a descender ligeramente. La sensación dentro del cuarto es agradable para la persona.

Figura 36. Gráfica temperatura habitación



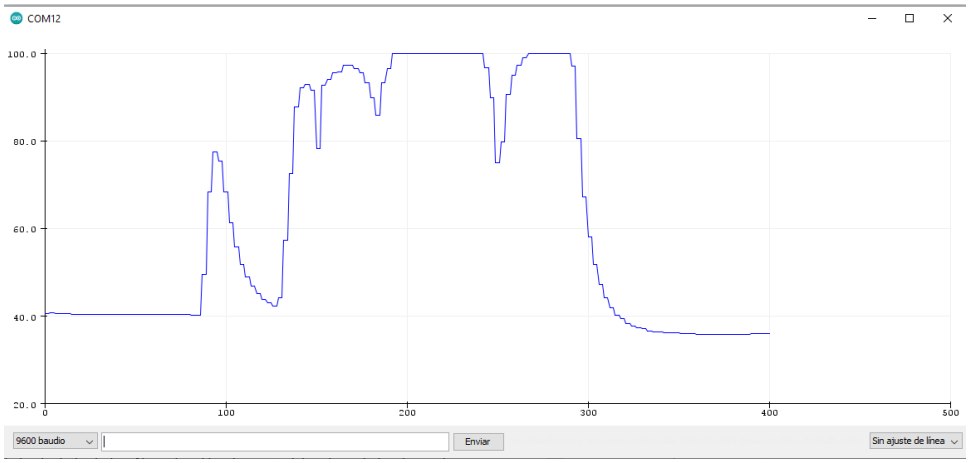
En la siguiente figura se plasma la lectura del sensor de presión situado en el cojín. Este devuelve un “0” cuando no detecta presión y un “1” cuando sí lo hace. Se puede observar cuando el paciente se levanta realmente del sillón (2) y cuando se produce un movimiento sin llegar a incorporarse totalmente (1).

Figura 37. Gráfica sensor de presión



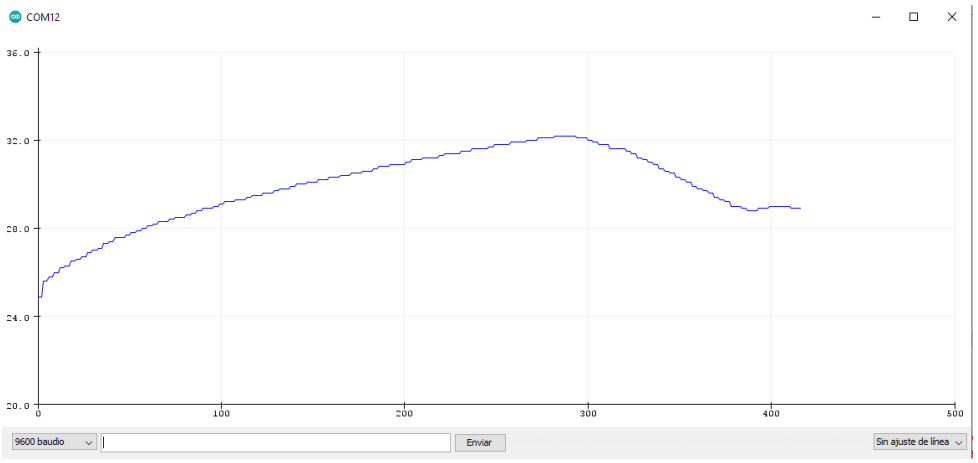
Al contemplar la siguiente gráfica que representa la lectura del sensor de humedad DHT22, se detecta fácilmente el momento en el que se ha humedecido la tela del cojín mediante un difusor de agua. Tras aumentar drásticamente el valor de lectura, se extrae el dispositivo del interior de la maqueta y, por lo tanto, el porcentaje de humedad obtenido desciende rápidamente.

Figura 38. Gráfica sensor de humedad



En esta última figura se representa el valor de lectura del sensor de temperatura situado en el interior del cojín. Este valor pretende aproximarse a la temperatura corporal del individuo. Puesto que el sensor está situado en el interior del dispositivo y no en el paciente, es notable el periodo de tiempo que pasa desde que se sienta hasta que la lectura se acerca a la temperatura real de la persona. Así pues, se observa como gráfica comienza ascendiendo progresivamente hasta que se levanta del sillón y prosigue descendiendo lentamente hasta alcanzar la temperatura de la habitación.

Figura 39. Gráfica temperatura interior del cojín



Finalmente, se ejecuta el programa principal y la única salida del sistema corresponde con el envío periódico de la información recogida junto con el identificador de cada variable, tal y como se ha explicado en los apartados anteriores. Se produce un envío cada 500 ms para no saturar la recepción radiofrecuencia.

Figura 40. Salida del sistema emisor

8.2.2 Arduino receptor

El sistema receptor, que se encuentra continuamente recibiendo información a través del dispositivo radiofrecuencia conectado, únicamente hace de intermediario entre los sensores y la Raspberry Pi, por lo tanto, recibirá exactamente los mismos String que han sido emitidos previamente. Aquí se muestra la información que capta este dispositivo de manera filtrada según su identificador.

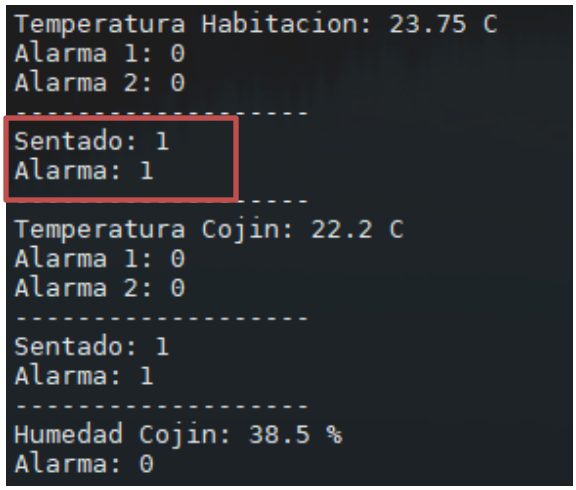
Figura 41. Salida del sistema receptor (Filtrada)

```
13:30:43.684 -> Temperatura habitación: 23.00 °C
13:30:43.730 -> Sentado: 0
13:30:43.823 -> Humedad interior: 39.70 %
13:30:44.473 -> Temperatura habitación: 23.00 °C
13:30:44.520 -> Sentado: 1
13:30:44.612 -> Humedad interior: 39.90 %
13:30:44.705 -> Temperatura interior: 22.50 °C
13:30:45.310 -> Temperatura habitación: 23.00 °C
13:30:45.310 -> Sentado: 1
13:30:45.403 -> Humedad interior: 39.90 %
13:30:46.055 -> Temperatura habitación: 23.00 °C
13:30:46.101 -> Sentado: 1
13:30:46.192 -> Humedad interior: 39.90 %
13:30:46.846 -> Temperatura habitación: 23.00 °C
13:30:46.986 -> Humedad interior: 39.70 %
```

8.2.3 Publicador MQTT

En cuanto al funcionamiento del publicador realizado en Python dentro de la Raspberry Pi Zero, se muestra la salida por pantalla del script. Esta salida corresponde con los valores obtenidos a través del puerto serie y el estado de la alarmas una vez realizada la gestión de estas.

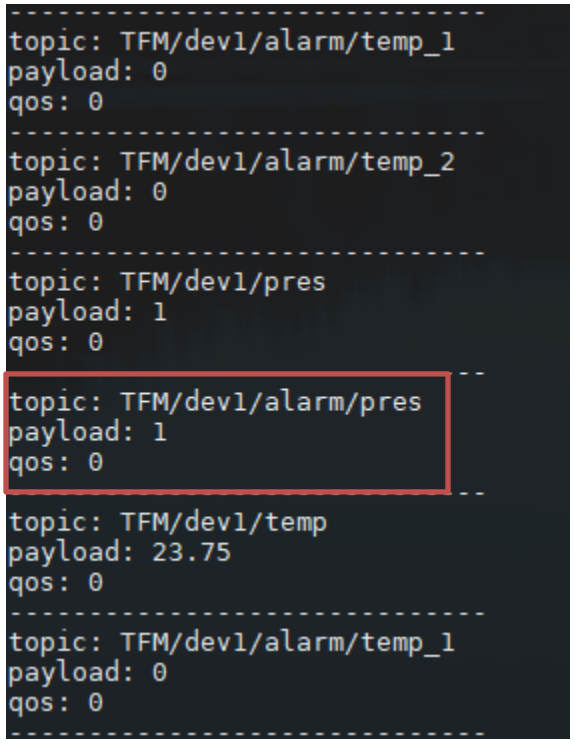
Figura 42. Salida del script Publicador.py



8.2.4 Suscriptor MQTT

En cuanto al suscriptor del bróker MQTT, se observan los diferentes mensajes recibidos según el tópico al que pertenecen. Así pues, se muestra por pantalla el aviso que corresponde con una publicación del script anterior conteniendo el tópico y contenido del mensaje (*payload*). Se puede ver como la alarma relacionada con el sensor de presión está activa, tal y como se ha publicado previamente.

Figura 43. Salida script Suscriptor.py

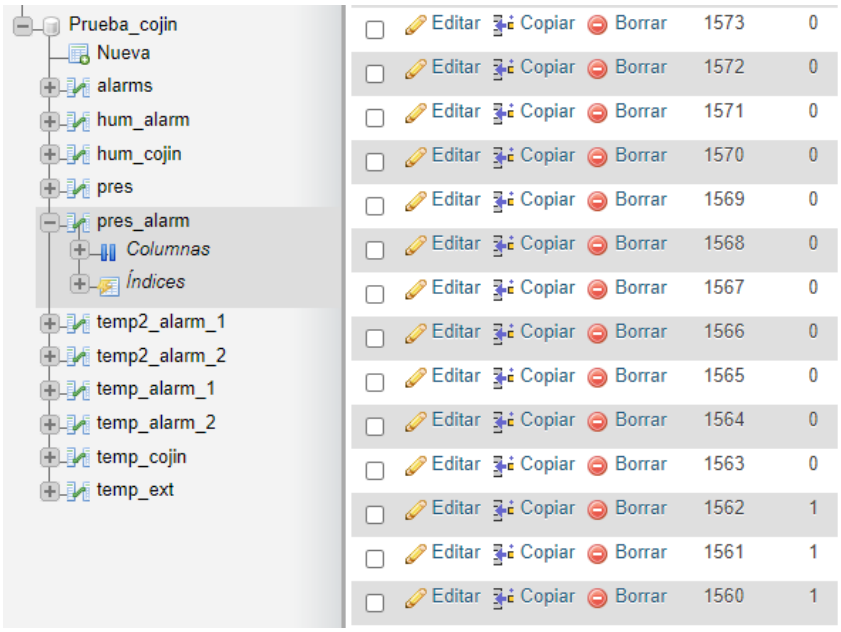


Del mismo modo que en el script anterior, se contempla como el orden de recepción ya no concuerda con el que establece en el primer microcontrolador. Esto se debe a que la emisión y recepción radiofrecuencia no presenta una estabilidad total, sino que ocasionalmente se producen pérdidas de información a causa de las interferencias o falta de alcance del sistema.

8.2.5 Base de datos

Accediendo a la base de datos del sistema mediante el usuario y contraseña configurados anteriormente, resulta que esta se ve actualizada y contiene los últimos valores recibidos junto con el identificador (*id*) que se le asigna automáticamente. En este caso se muestra la tabla que contiene el valor de alarma relacionada con el sensor de presión.

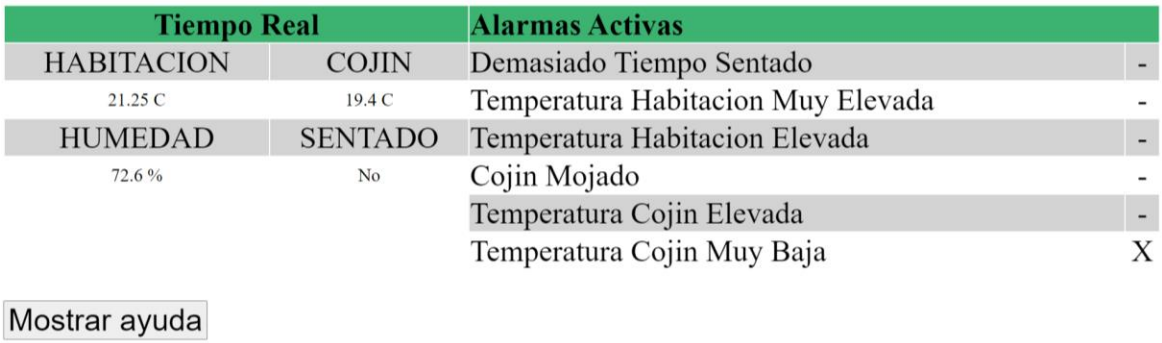
Figura 44. Base de datos phpMyAdmin



8.2.6 Interfaz Web

Finalmente, al abrir la interfaz web, se adquiere la vista resumen que recoge la última actualización de la base de datos y la muestra mediante una tabla de valores y alarmas.

Figura 45. Interfaz Web



9. CONCLUSIONES

Tras la elaboración de la memoria se presentan las conclusiones obtenidas en dos grandes bloques. Según si afectan a aspectos generales de la investigación o específicos del sistema obtenido.

Aspectos generales:

Se ha conseguido diseñar e implementar un dispositivo completo de monitorización de la actividad diaria de una persona haciendo uso únicamente de componentes de bajo coste. Además, se ha logrado que el sistema sea divisible en tres módulos independientes y modificables individualmente.

Aspectos específicos:

Tras el desarrollo y prueba del dispositivo, se ha demostrado que los sensores utilizados poseen calidad y precisión suficiente para dotar proyecto de un funcionamiento correcto. Sin embargo, existen limitaciones reales en la transmisión radiofrecuencia que dificultan la estabilidad del sistema.

Por otro lado, la autonomía de la batería es un condicionante a la hora de hacer un uso continuo y prolongado del aparato inteligente. Asimismo, el tamaño del dispositivo emisor es un inconveniente si se coloca en el interior del cojín.

Finalmente, la seguridad del sistema no está preparada contra ciberataques. También, queda claro que es necesaria una configuración previa para cada usuario en función de sus necesidades y variables corporales. El código es estable y no sufre de bucles infinitos que no lleven a ningún lugar.

10. RECOMENDACIONES Y TRABAJOS FUTUROS

En cuanto a las recomendaciones para trabajos o investigaciones relacionadas se propone:

Implementación de un sistema de comunicación inalámbrica entre el cojín y el dispositivo receptor más estable y que alcance mayor distancia que el llevado a cabo en esta investigación. Igualmente, una mejora en el desarrollo del entorno web interfaz de usuario que permita modificar los rangos de las alarmas para así facilitar la personalización y configuración del sistema para cada paciente.

Por otra parte, es interesante reducir el tamaño del dispositivo emisor situado en el cojín para evitar molestias a la persona que haga uso de este.

Finalmente, se propone una mejora de la seguridad del software para así evitar posibles ataques a la red en la cual se encuentra conectado el sistema principal y la incorporación de un sistema de visión en tiempo real que aporte la posibilidad de observar al paciente o incluso escucharlo en caso de emergencia.

11. REFERENCIAS BIBLIOGRÁFICAS

Ashwini Kumar Sinha (2019) *Smart Wireless Water Meter with Web DB IoT Projects*

https://www.electronicsforu.com/electronics-projects/smart-wireless-water-meter-with-web-db

Biswajit Das (2019) *IoT-Enabled Air Pollution Meter with Digital Dashboard on Smartphone*

https://www.electronicsforu.com/electronics-projects/iot-enabled-air-pollution-meter

Electan Electrónica y Robótica. (2006) *Velostat hoja conductiva sensible a la presión - Linqstat*

https://www.electan.com/pressuresensitive-conductive-sheet-velostatlinqstat-p-7636.html

Eli the Computer Guy (2018) *LAMP Server - What is...*

https://www.youtube.com/watch?v=S079SXX2d2g&ab_channel=ElitheComputerGuy

ESET (2019) *Tendencias 2019: Privacidad e intrusión en la aldea global*

https://www.welivesecurity.com/wp-content/uploads/2018/12/Tendencias-Ciberseguridad-2019-ESET.pdf

Geriatric Area (2018) *Nubolo, una línea de cojines dinámicos "inteligentes" para prevenir la aparición de úlceras por presión*

https://www.geriatricarea.com/2018/02/28/nubolo-una-linea-de-cojines-dinamicos-inteligentes-para-prevenir-la-aparicion-de-ulceras-por-presion/

HetPro Tutoriales. (2020) *MCP9808 con Arduino – Sensor de temperatura I2C*

https://hetpro-store.com/TUTORIALES/sensor-mcp9808-de-temperatura/

Luis Llamas (2016) *Comunicación inalámbrica en Arduino con módulos RF 433 MHz*

https://www.luisllamas.es/comunicacion-inalambrica-en-arduino-con-modulos-rf-433mhz/

Luis Llamas (2015) *Esquema de patillaje (pinout) de Arduino UNO, Nano, Mini y Mega*

https://www.luisllamas.es/esquema-de-patillaje-de-arduino-pinout/

Microchip Technology Inc. (2011) *MCP9808.*

https://ww1.microchip.com/downloads/en/DeviceDoc/25095A.pdf

Miles Web (Actualizado 2020) *Understanding a Web Server and Types of Web Servers*

https://www.milesweb.in/blog/hosting/web-server-types-web-servers/

Negocios y gestión de la dependencia (2019) *El número de dependientes en España se duplicará en una década*

https://gestionydependencia.com/noticia/2598/actualidad/el-numero-de-dependientes-en-espana-se-duplicara-en-una-decada.html

Nociones de (2016) *Introducción a la librería Paho y MQTT para IoT(Internet de las Cosas)*

https://www.nociones.de/introduccion-paho-mqtt-iot/

Peter Waher, Pradeeka Seneviratne, Brian Russell, Drew Van Duren (2016)

IoT: building arduino-based projects : explore and learn about Internet of Things to develop interactive arduino-based Internet projects : a course in three modules

12. ANEXOS

Se muestra el listado de archivos adjuntos mencionados durante el desarrollo de este documento.

```
+--- publicador
     |   +--- publicador.py
+--- README
+--- receive2
     |   +--- receive2.ino
+--- site
     |   +--- config.php
     |   +--- hum_alarm.php
     |   +--- hum_cojin.php
     |   +--- pres.php
     |   +--- pres_alarm.php
     |   +--- temp2_alarm_1.php
     |   +--- temp2_alarm_2.php
     |   +--- temp_alarm_1.php
     |   +--- temp_alarm_2.php
     |   +--- temp_cojin.php
     |   +--- temp_ext.php
     |   +--- values.php
+--- suscriptor
     |   +--- suscriptor.py
+--- transmit
     |   +--- transmit.ino
```