Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Internet de las Cosas en dispositivos de bajo coste

Internet of Things in low cost devices

Karmele Victoria Bencomo Martín

La Laguna, *12* de *Marzo* de 2021

D. **Jonay Tomas Toledo Carrillo**, con N.I.F. 78698554-Y profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

C E R T I F I C A (N)

Que la presente memoria titulada:

“Internet de las Cosas en dispositivos de bajo coste”

ha sido realizada bajo su dirección por D. **Karmele Victoria Bencomo Martín**,

con N.I.F. 43380705-Z.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 21 de noviembre de 2018

Agradecimientos

Para empezar, quiero agradecer a mi tutor de la asignatura, Jonay Toledo, por su paciencia, constancia y dedicación durante la realización de este trabajo de fin de grado.

Agradecer también a La Universidad de La Laguna y a sus profesores por todo el esfuerzo, demostrando que siempre hay una solución a todos los problemas.

También quiero agradecerle a mi familia por no perder la confianza en mi.

Y finalmente, a mis amigos y compañeros, tanto de la carrera como de fuera por el apoyo durante estos trabajosos años haciendo que sea posible llegar hasta aquí.

Licencia

Resumen

El Internet de las Cosas se refiere a la interconexión de dispositivos informáticos interrelacionados y a la capacidad de transferir datos a través de una red sin requerir interacción de persona a persona o de persona a computador.

El **IoT** (Internet de las Cosas) ha revolucionado el panorama tecnológico actual y está consiguiendo mayor presencia en nuestro día a día. Una de las finalidades de este campo es que los dispositivos sean de bajo coste y de bajo consumo y eso es lo que se intenta conseguir en este TFG.

A partir de diferentes nodos, que se encargarán de recoger la temperatura y la humedad de manera intermitente, enviarán cada cierto tiempo la información a un servidor que proporciona estos datos a los usuarios que lo requieran.

Para que el funcionamiento de este TFG haya sido posible, primero se ha tenido que hacer un estudio previo de las herramientas y conceptos para conseguir su funcionalidad de la manera más óptima posible. Se ha estudiado una placa que sea capaz de mantener una conexión wifi y pueda realizar su labor de envío de datos a través de dicha conexión. Una vez analizado el hardware, desarrollamos el software, es decir, el código que recoge la información, el servidor que envía los datos a la base de datos, la configuración de la base de datos y, finalmente el código del servidor web.

Palabras clave: Arduino, sensor DHT22, WeMos D, Internet de las Cosas...

Abstract

Internet of Things refers to the interconnection of interrelated computing devices and the ability to transfer data over a network without requiring person-to-person or person-to-computer interaction.

IoT (Internet of Things) has changed the current technological world. It is achieving greater presence day by day. One of the goals of this field is that the devices are low cost and low consumption, so, this is what we are trying to achieve with this project.

From different nodes, which will be working on collecting temperature and humidity intermittently. Each one of them will be sending the information from time to time to a server that will provide this data to the users who require it.

So that the functionality of this tfg has been possible, first we had to look for information about the tools we have chosen, and to choose which things are better for the most optimal performance.
We have studied about a board that could work with an WiFi connection to share the information with the server. Once we had analyzed the hardware we are going to use, we developed software, it means that the code that is taking the temperature and humidity information, server code, that sends data to the server, database configuration and finally, web server code.

Keywords: Arduino, DHT22, temperature sensor, WeMos D1, Internet of Things ...

Índice general

Capítulo 1. Introducción **10**

Capítulo 2. Conceptos Básicos. **12**

Internet de las Cosas 12

Historia 12

Aplicaciones 13

Protocolos 13

UDP 14

TCP 15

HTTP 16

Sockets 17

EEPROM 18

Capítulo 3. Herramientas **19**

WeMos D1 y ESP8266 19

DHT22 21

Arduino IDE 21

MySQL 22

Lenguajes de programación 22

Python 23

HTML 23

SQL 24

Capítulo 4. Objetivos **25**

Alcance 25

Antecedentes 26

Capítulo 5. Desarrollo, Implementación y Resultados **27**

WIFI DINÁMICA 29

RECOGIDA DE DATOS SENSOR-PLACA 30

ENVÍO DE DATOS AL SERVIDOR 30

DEEP SLEEP Y AHORRO DE ENERGÍA 32

SERVIDOR UDP 32

CREACIÓN DE LA BASE DE DATOS 33

ENVÍO DE DATOS DEL SERVIDOR A LA BASE DE DATOS 34

SERVIDOR WEB 35

Capítulo 6. Conclusiones y líneas futuras **36**

Conclusiones 36
Líneas futuras 37
Página Web 37
Avisos por cambios bruscos 37
Hardware más óptimo 37

Capítulo 7. Summary and Conclusions 39
Summary 39
Future work plans 39
Website 40
Setting alerts when big changes comes 40
Hardware optimizing 40

Capítulo 8. Presupuesto 41
Material 41
Mano de obra 41

Apéndice 1. Programa Arduino IDE 42

Apéndice 2. Servidor UDP. 48

Apéndice 3. Servidor web 50

Bibliografía 51

Índice de figuras

Figura 1: Internet de las cosas
Figura 2: Cabecera protocolo UDP
Figura 3: Conexión TCP
Figura 4: Esquema de comunicación HTTP
Figura 5: Protocolo HTTP
Figura 6: Arquitectura del chip ESP8266
Figura 7: Como funciona MySQL
Figura 8: Python
Figura 9: Conexión entre placa WeMosD1 y sensor DHT22
Figura 10: Escaneo de redes en serial de arduino
Figura 11: Insertar password en el serial de arduino
Figura 12: Recogida de datos del sensor de temperatura y humedad
Figura 13: Estableciendo conexión UDP desde Arduino
Figura 14: Envío de datos al servidor UDP
Figura 15: Creación del socket UDP
Figura 16: Socket recibiendo datos y mostrándose por pantalla
Figura 17: SHOW TABLES en MySQL
Figura 18: Creación de la conexión entre servidor - BD
Figura 19: Consulta SQL en servidor
Figura 20: SELECT de la tabla data

Capítulo 1. Introducción

Hoy en día, la posibilidad de tenerlo todo bajo control está al alcance de nuestra mano, solo tocando la pantalla de nuestro móvil.

El **Internet de las cosas**, nace de la necesidad de la digitalización de nuestro entorno, es por ello, que este ha sido un campo que ha evolucionado rápidamente en muy poco tiempo.

Se trata de potenciar los objetos cotidianos como microondas, neveras e incluso bombillas, para que puedan transmitir información, sin necesidad de intervención humana de por medio.

Es un concepto que nos rodea continuamente y le da importancia a los datos. Los dispositivos pueden proporcionar datos útiles sobre su estado o los procesos de su entorno. *Alexa*, para empezar es un buen ejemplo. Es un dispositivo capaz de comunicarse con otros que se encuentren en su casa, sin necesidad de cableado, solo mediante su conexión a la red.

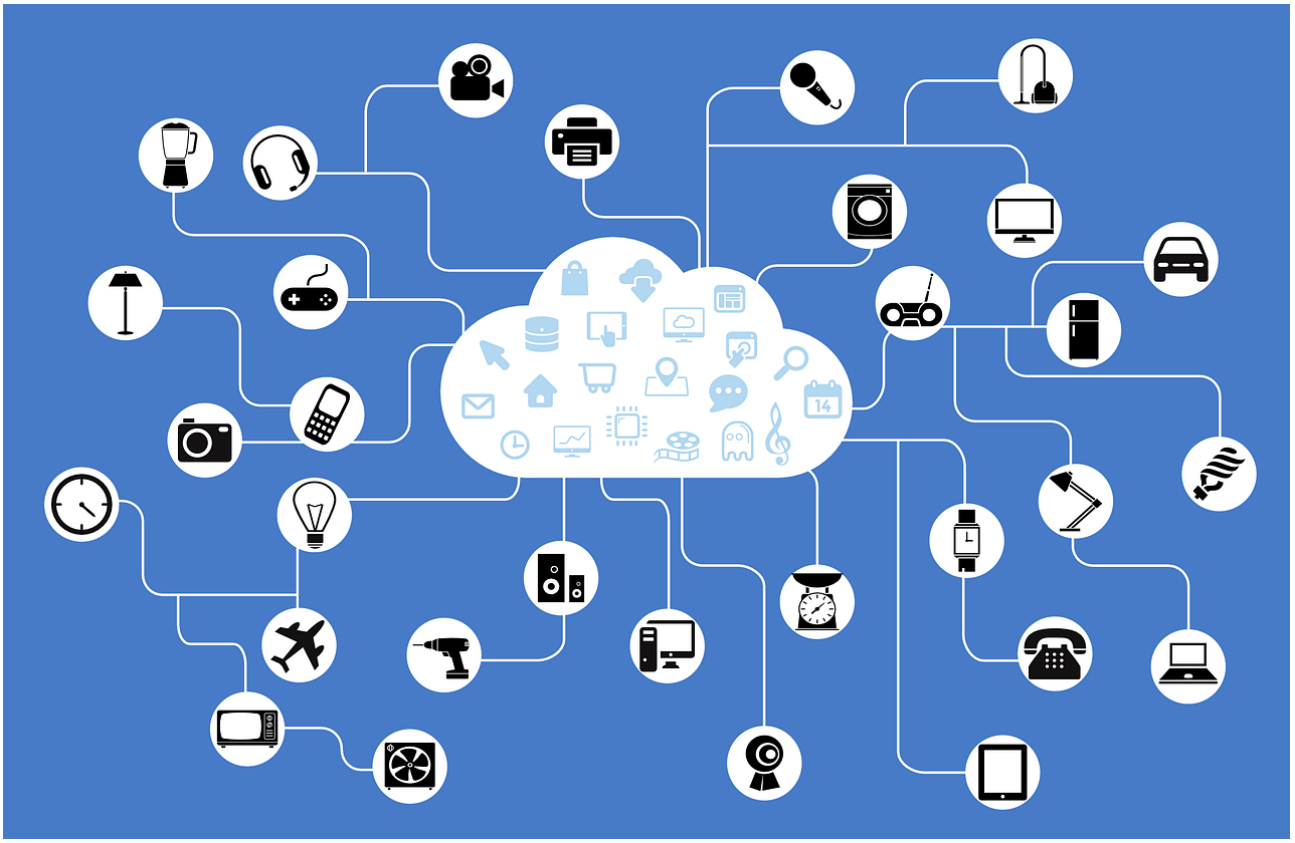

Figura 1: Internet de las Cosas

Fuente: pixabay.com

En este proyecto, investigaremos sobre este tema tan amplio para proponer soluciones útiles. Para ello, estudiaremos con qué dispositivos nos conviene trabajar. Se hace un estudio de los diferentes productos del mercado y cuál es más conveniente para nuestro objetivo. Tendremos en cuenta también, los protocolos a usar a la hora de conectar este dispositivo a la red, creando una red segura.

La idea principal consiste en que varios nodos de dispositivos conectados a un servidor, sean capaces de enviar la información y quede guardada en una base de datos.

Además ofreceremos, la posibilidad de acceder a estos datos a través de un servidor web para facilitar la información a quien la requiera.

"Si tuviésemos ordenadores que fuesen capaces de saber todo lo que pudiese saberse de cualquier cosa –usando datos recolectados sin intervención humana– seríamos capaces de hacer seguimiento detallado de todo, y poder reducir de forma importante los costes y malos usos. Sabríamos cuando las cosas necesitan ser reparadas, cambiadas o recuperadas, incluso si están frescas o pasadas de fecha. El Internet de las Cosas tiene el potencial de cambiar el mundo como ya lo hizo Internet. O incluso más." - Kevin Ashton

Capítulo 2. Conceptos Básicos.

Al tratarse de un tema muy técnico y novedoso, explicaremos primero muchos de los conceptos que utilizaremos a lo largo de esta memoria para comprender nuestro trabajo.

1.1 Internet de las Cosas

1.1.1 Historia

En el siglo XIX, se desarrollaron los primeros experimentos de **telemetría**, que consiste en la medición remota de magnitudes físicas y el posterior envío de la información a un sistemas.

Específicamente, en 1874, un dispositivo instalado por un equipo de científicos franceses en el *Mont Blanc*, enviaba información metereológica y de profundidad a París a través de un enlace de radio de onda corta.

En 1926, Nikola Tesla no se quedó atrás, en una entrevista en la revista *Colliers* habló de la conexión a nivel global, refiriéndose al mundo como un "**gran cerebro**". Este concepto, lo podemos entender hoy en día como Internet.

Más tarde, sobre la década de los 70, el Departamento de Defensa de EEUU envió el primer mensaje a través de **ARPANET,** su red operativa. Esto permitió, el desarrollo de los primeros protocolos de comunicaciones, en busca de comunicaciones más rápidas, menos costosas y compatibles entre ellas.

Fue en 1982, cuando se conectó el primer objeto a Internet. Una máquina de refrescos de la Universidad Carnegie Mellon en Pittsburgh, Pensilvania, mediante una serie de microswitches se informaba de la temperatura de dicha máquina y del número de bebidas que contenía. Cualquier persona, conectada a ARPANET, en ese momento, podía ver el estado de la máquina de refrescos.

La **revolución de Internet** en los 90, fue un paso muy importante para la tecnología. Fue en ese entonces cuando se creó la primera conexión funcional entre cliente y servidor con el protocolo TCP.

También, se conectó por primera vez una tostadora a Internet, de manera que cualquiera conectado a la web, podía controlar su encendido, apagado y el tiempo de tostar. La primera webcam conectada, por la Universidad de Cambridge, que actualizaba la imagen de la cafetera cada tres minutos, para comprobar si tenía

café o no. La primera cámara portátil conectada a la web… Todo desembocó en una revolución de objetos conectados en los 2000.

Con la evolución de Internet y de las tecnologías, seguían apareciendo nuevos objetos conectados a la red, hasta que en 2009, por primera vez, se utilizó públicamente el término de Internet de las Cosas, por el profesor del MIT Kevin Ashton en el RFID journal.

A partir de ahí, Google empezó a trabajar en un proyecto para el desarrollo de un coche autónomo, se empezó a utilizar en la medicina, con el primer implante cardiaco monitorizado por IoT, electrodomésticos inteligentes… Hasta que se extendió a todos los campos posibles. A día de hoy, estamos rodeados de “cosas” conectadas a Internet.

1.1.2 Aplicaciones

Como mencionamos anteriormente, el Internet de las Cosas nos rodea, de tal manera que lo podemos aplicar a casi cualquier campo.

La aplicación más famosa y común es la domótica. Cada vez más y más gente se encuentra en la búsqueda de un hogar inteligente, así como bien dijimos antes, Alexa, se trata de un claro ejemplo de buscar la comodidad en casa. Podemos llegar a ser capaces de controlar la temperatura, la comodidad y la seguridad de nuestra casa gracias a este concepto.

Los wearables, que tienen una finalidad más de ocio. Apple con el smart watch, las gafas de realidad aumentada para las consolas, etc…

En el campo de Smart Cities, IoT tiene una gran trascendencia, desde gestionar el tráfico, los residuos, la distribución de aguas, la seguridad urbana… El avance a un mundo más tecnológico, sencillo y cómodo, hace que este tipo de dispositivos sean completamente necesarios para una sociedad como la de hoy.

También lo podemos relacionar al mundo Industrial, al automotriz con el desarrollo de los coches autónomos, en la agricultura, para los estudios y el control meteorológico, del suelo, de sustancias químicas…,etc.

Y finalmente, la que más cercana tenemos, en la salud. Gracias a ello, se permite la monitorización y control del estado de los pacientes, permite la recolección de datos que pueden ser útiles para un futuro próximo, una gestión más eficaz de los recursos… Es un campo lleno de oportunidades y con retos aún más complicados.

1.2 Protocolos

En informática, los protocolos son lenguajes o códigos de comunicación entre sistemas informáticos que establecen la conexión entre estos, de una manera eficaz y ordenada. Se implementan mediante hardware y software.

1.2.1 UDP

El Protocolo de Datagrama de Usuario (User Datagram Protocol) pertenece al nivel de transporte y se basa en el intercambio de datagramas (Paquete de datos que constituye el mínimo bloque de información).

Permite la transmisión de datos sin establecer una conexión previa. Esto es posible, gracias a que la cabecera del datagrama lleva la suficiente información para el direccionamiento.

Se trata de un protocolo que destaca por la rapidez de transmisión de datos, al no necesitar una conexión previa, no se pierde tiempo en la confirmación de la conexión ni en la espera de respuesta por la otra parte.

Por otro lado, al no tener control de flujo, los paquetes pueden adelantarse unos a otros, también, al no recibir ningún tipo de confirmación, cabe la posibilidad de que los paquetes no lleguen.

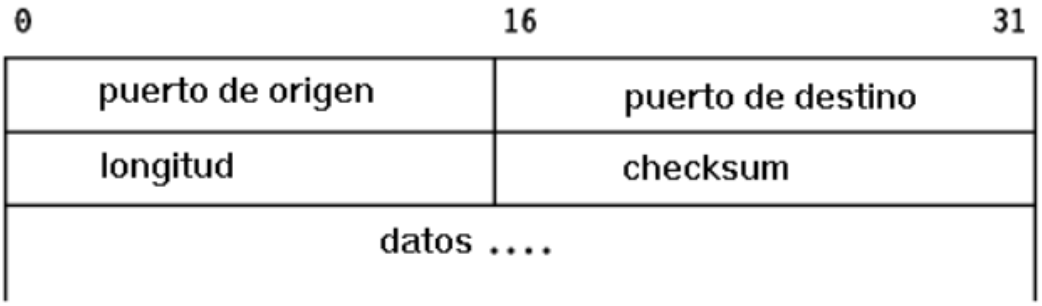

Figura 2: Cabecera protocolo UDP
Fuente: personales.upv.es

El puerto de origen, indica el puerto de donde envía; el puerto de destino, el puerto al que envía la información; la longitud en bytes del datagrama (cabecera incluida); y, el checksum, que es un campo opcional de 16 bits en complemento a uno. Es la suma en complemento a uno de una cabecera pseudo-IP, la cabecera UDP y los datos UDP.

Este protocolo es útil en aplicaciones como en las de este proyecto, las llamadas aplicaciones basadas en best effort delivery, aquellas en las que les basta con una transmisión poco fiable de la información, porque la están repitiendo constantemente de la misma manera.

También en aplicaciones ligeras, debido a su baja sobrecarga; en aplicaciones multicast, ya que soporta transmisiones multidifusión IP y no únicamente las

one-to-one. Y las aplicaciones en tiempo real gracias a los mecanismos que permiten la transmisión de información en tiempo real.

1.2.2 TCP

El protocolo de Control de Transmisión (Transmission Control Protocol) permite establecer una conexión entre dos puntos terminales en una red informática para el intercambio mutuo de información.

A diferencia de UDP, se considera que es un protocolo fiable ya que garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en el que se transmitieron.

Requiere que exista una conexión y autorización previa a la transferencia de datos entre cliente y servidor. Una vez ambas partes hayan autorizado la transmisión, se iniciará el envío y recepción de datos.

Normalmente, el protocolo TCP se combina con el protocolo IP, formando el TCP/IP, el protocolo que forma la base de la transmisión de datos de la gran mayoría de las redes y servicios de internet.

Se trata de una conexión bidireccional, ambos sistemas pueden enviar o recibir datos.

Para que una conexión sea posible se necesita dirección IP y puerto, de manera ordenada.

Un servidor puede dar servicio a varios clientes, siempre y cuando se utilicen diferentes números de puertos.

Para que se establezca una conexión TCP, debe ocurrir lo siguiente:

- Si un cliente quiere establecer la conexión, envía al servidor un paquete o segmento SYN (synchronize) con un número de secuencia individual y aleatorio, lo que garantizará que la transmisión de datos se complete en el orden correcto y sin duplicados.
- Cuando el servidor recibe el paquete, confirma la conexión enviando un paquete SYN-ACK (acknowledgement), con el número de secuencia del cliente tras haberle sumado uno.
- El cliente finalmente, confirma la recepción del paquete mediante el envío de un paquete ACK, que cuenta con el número de secuencia del servidor tras haberle sumado uno y, se pueden enviar los primeros datos al servidor.

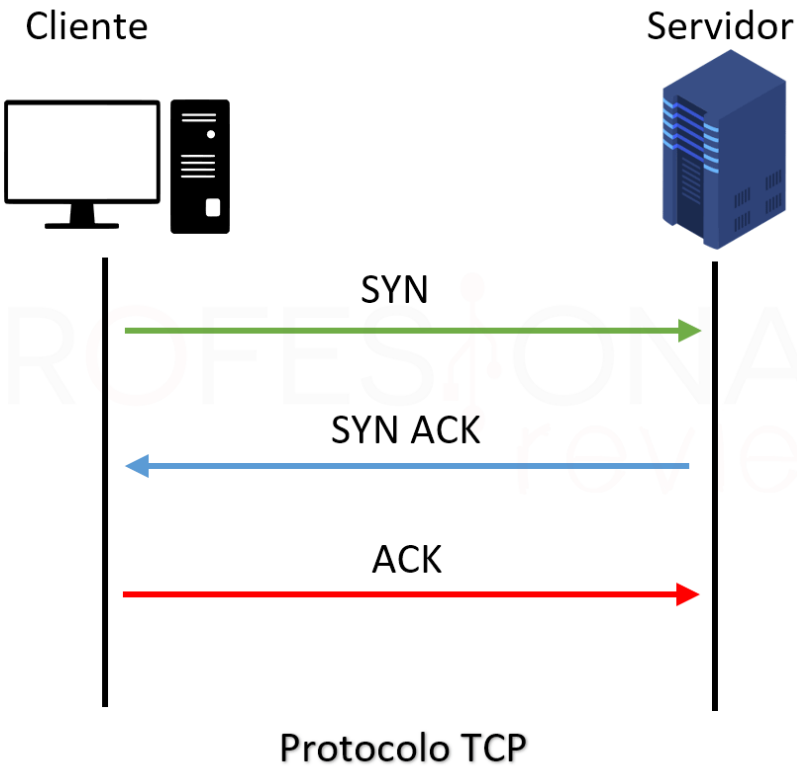

Figura 3: Conexión TCP
Fuente: profesionalreview.com

1.2.3 HTTP

El protocolo de Transferencia de Hipertexto (Hypertext Transfer Protocol) pertenece a la capa de aplicación. Es la base de la comunicación de datos para la **World Wide Web** (www).

Sigue un esquema de petición-respuesta entre un cliente y un servidor. El cliente envía un mensaje de solicitud HTTP con un formato específico al servidor, y el servidor, que proporciona recursos como archivos HTML, responde al cliente. Esta respuesta contiene información sobre el estado de la solicitud.

Se trata de un protocolo “sin estado”, ya que no lleva registro de las visitas anteriores, si no que cada vez que se inicia, empieza de nuevo. La información anterior, está almacenada en *cookies*.

Funciona tal que, cuando escribes una dirección web en el navegador, el navegador envía la solicitud HTTP al servidor que administra el dominio de la dirección web. El servidor web, recibe la petición. La petición es un mensaje requiriendo cierto archivo, ese archivo es buscado por el servidor y si lo encuentra, responde a la solicitud con el archivo, a lo que el navegador, lo recibe y lo muestra

en forma de página web.

Figura 4: Esquema de comunicación HTTP.
Fuente: ionoes.es

En un principio, el protocolo HTTP, servía para solicitar archivos HTML a un servidor web, ahora, se puede solicitar cualquier tipo de archivo.

También se utiliza en APIs basadas en REST para controlar los servicios web, para las comunicaciones entre máquina-máquina, reproductores multimedia y bases de datos.

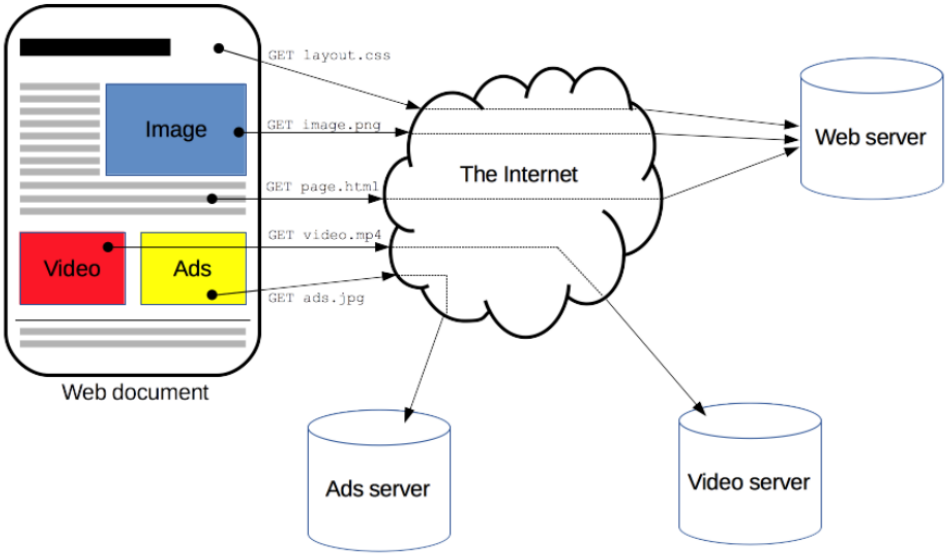

Figura 5: Protocolo HTTP
Fuente: developer.mozilla.es

1.3 Sockets

Es un túnel de comunicación que ayuda a que dos programas se comuniquen entre sí, ya sean de la misma o de diferentes máquinas, es decir, un tipo de software que actúa como un punto final en un canal de comunicación bidireccional.

Son la base de internet y de protocolos como el **HTTP**.

Los Sockets son el soporte que da el sistema operativo para la comunicación basada en redes TCP/IP, por ello, pueden cambiar cualquier flujo de datos de manera fiable y ordenada. Además, simplifican el funcionamiento de un programa ya que, a nivel de programación, solo habría que preocuparse de manipular sus propias funciones.

Sin embargo, existen los sockets UDP. Estos no están orientados a la conexión. Por lo tanto, un programa puede abrir un socket y ponerse a escribir mensajes en él o leer, sin necesidad de esperar a que alguien se conecte en el otro extremo del socket.

1.4 EEPROM

Electrically Erasable Programmable Read-Only Memory o ROM (Memoria de solo lectura) programable y borrable eléctricamente.

Es un tipo de memoria, que puede ser programada, borrada y reprogramada eléctricamente, no como la EPROM, que para poder borrarla se necesitaría un aparato que emite rayos ultravioleta.

Se trata de una memoria no volátil.

Es un chip de memoria que retiene su contenido sin energía.

La gran desventaja de este tipo de memorias es que tiene un número limitado de escrituras (100.000 - 1.000.000 veces), por lo que habría que tenerlas en cuenta para calcular su vida útil con nuestro programa

La memoria flash es una forma avanzada de EEPROM, por lo que la EEPROM es más lenta a la hora de leer y escribir en ella.

Las celdas de memoria de una EEPROM están constituidas por un transistor MOS, que tiene una compuerta flotante (estructura SAMOS), su estado normal está cortado y la salida proporciona un 1 lógico.

Capítulo 3. Herramientas

A continuación, se detallarán las herramientas utilizadas y la finalidad de cada una de ellas durante el desarrollo de este proyecto.

1. WeMos D1 y ESP8266

La herramienta principal e indispensable para que este proyecto se haya podido llevar a cabo, es una placa WeMos D1 con un chip WiFi ESP8266 integrado.

Este tipo de placas tiene el formato de un Arduino UNO pero incluyendo el chip ESP8266. Este chip mantiene la compatibilidad de programación desde el entorno Arduino IDE.

A diferencia del resto de placas de Arduino, esta tiene únicamente una entrada analógica (A0) y 14 pines digitales (D0-D13), además, todos los pines funcionan a 3.3V, por lo que no será compatible con periféricos que funcionan a 5V.

Podemos resumir los pines de la placa en esta tabla:

PIN	ESP8266	Función
TX	TXD	UAT TX
RX	RXD	UART RX
A0	A0	Entrada/Salida analógica
D0	GPIO16	IO
D1	GPIO5	IO, SCL
D2	GPIO4	IO, SDA
D3	GPIO0	IO, 10k Pull-up
D4	GPIO2	IO 10k Pull-up, LED
D5	GPIO14	IO, SCK
D6	GPIO12	IO, MISO

D7	GPIO13	IO, MOSI
D8	GPIO15	IO, 10k Pull-down, SS
G	GND	Tierra
5V	-	5V
3V3	3.3V	3.3V
RST	RST	Reset

Tabla 1: Conexiones de WeMos D1 y ESP8266

En resumidas cuentas, la placa WeMos D1, es gobernada por el chip ESP8266, que se encarga del procesamiento y control del WiFi. ¿Pero qué es el ESP8266?.

Es un sistema que integra en un encapsulado un procesador **RISC** de propósito general y de bajo consumo (32 bits) con conectividad WiFi completa (con pila **TCP/IP** integrada).

Ofrece una RAM con 32 KB para instrucciones, otros 32 KB para caché y 80 KB para los datos.

Tiene 16 pines GPIO, PWM en todos los pines (10b), un conversor analógico-digital de 10b, un voltaje de operación de 3.0V a 3.6V y un consumo medio de 80mA.

Además, en reposo llega a tener un consumo menor de 10 mW, por lo que lo hace ideal para proyectos enfocados a IoT, como es este caso.

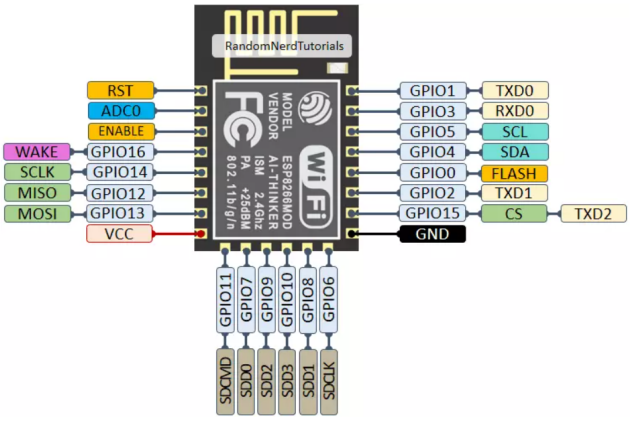

Figura 6: Arquitectura del chip ESP8266

Fuente: hardzone.es

Este tipo de módulos tienen múltiples usos en la vida cotidiana, unos ejemplos son:

- Los electrodomésticos conectados
- La domótica
- Automatización de la industria

- Las cámaras IP
- Redes de sensores
- Wearables
- IoT & IIoT (especificado en el sector industrial).

2. DHT22

Se trata de un sensor de temperatura y humedad de buen rendimiento y bajo costo.

Ofrece una salida digital con un single-bus bidireccional, por lo que no serán necesarias las entradas analógicas.

Se suele utilizar en aplicaciones de control automático (como esta) de temperatura, aire acondicionado, monitoreo ambiental en agricultura y demás.

Existen también otras opciones como el DHT11, sin embargo, el DHT22 presenta mayor precisión y mejor resolución.

A la hora de ponerlo en funcionamiento, se trata de un proceso muy sencillo ya que a nivel hardware, solo sería necesario conectar el pin VCC de alimentación a 3-5V, el GND a la Tierra (0V) y el pin de datos a un pin digital de nuestra placa WeMos D1.

Es capaz de medir unos rangos de temperatura de entre -40ºC a 80ºC con una precisión <±0.5ºC; un rango de humedad de 0 a 100% RH, con una precisión del 2% RH.

Envía los datos cada 2 segundos y su peso y tamaño son insignificantes, por lo que todo son ventajas a la hora de utilizar este tipo de sensores.

3. Arduino IDE

Entorno de Desarrollo Integrado de Arduino (Integrated Development Environment), es decir, un programa informático compuesto por un conjunto de herramientas de programación, en el que se pueden usar uno o más lenguajes.

Se trata de un software completamente gratuito y muy sencillo de usar que ofrece herramientas para manejar de una manera más fácil nuestra placa a través de un programa.

Una vez tengas el programa funcional, lo cargaremos a través del USB, el IDE se encargará de validar el código, compilarlo y cargarlo en el microcontrolador de nuestra placa, que posteriormente si no se detecta ningún error, ejecutará.

En caso de error, se mostrará por pantalla.

Además, proporciona un monitor que sirve como interfaz de entrada y salida para nuestra placa. Nos permite la entrada de datos y la salida de mensajes.

El lenguaje de programación que soporta Arduino se basa en C/C + +, y se simplifica con el uso de la biblioteca Arduino que incorpora el IDE.

4. MySQL

Sistema de gestión de bases de datos relacional de código abierto (RDBMS). RDBMS es un software utilizado para crear y administrar bases de datos basadas en un modelo relacional.

Desarollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo.

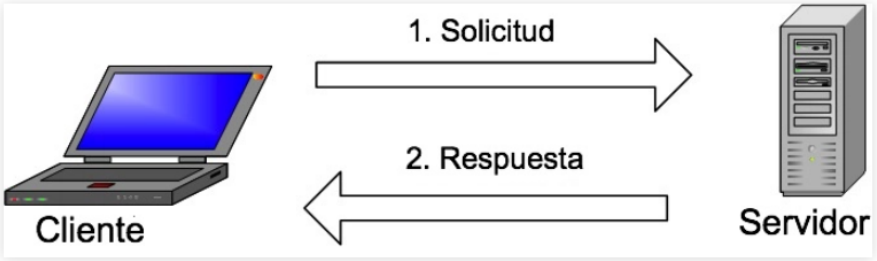

Figura 7: Cómo funciona MySQL
Fuente: hostinger.es

MySQL fue desarrollado por MySQL AB, una compañía sueca, en 1994. Sun Microsystems se hizo cargo de MySQL AB en 2008 y, posteriormente fue comprada por Oracle Corporation en 2010, hasta hoy.

Se trata de una base de datos flexible y fácil de usar, es gratuita y tienes total libertad para modificar el código fuente a tu antojo.

Además tiene una instalación simple y corta.

Ofrece un alto rendimiento y un amplio compendio de servidores de clúster que lo respalda.

Una de sus características principales es la seguridad que proporciona, ya que los datos, son importantes. Por ello, contiene un sistema de privilegios de acceso y administración de cuentas de usuario.

5. Lenguajes de programación

A continuación, describiré los lenguajes de programación utilizados para el desarrollo del proyecto. Cada uno de ellos se aplica a entornos diferentes.

5.1. Python

Lenguaje de programación versátil multiplataforma y multiparadigma, es decir, que permite diferentes estilos de programación, entre ellos, la programación orientada a objetos, la imperativa y la funcional.

Destaca por su código legible y limpio.

Figura 8: Python
Fuente: escuelapython.com

Una de sus características principales es que al contar con una licencia de código abierto y al tratarse de un lenguaje con mucho trayecto, se encuentra en muchas aplicaciones y sistemas operativos. Se encarga de la ejecución de servidores en aplicaciones iOS, Android, Linux, Windows o Mac.

Ventajas de programar con Python:

- Simple y rápido. Se suele utilizar para scripts.
- Elegante y flexible. Más adaptado al programador a la hora de interpretarlo.
- Fácil de aprender. Suele ser uno de los lenguajes de iniciación en programación.
- Organizado y limpio. Al no existir ; que separen las sentencias, el código tiene que estar correctamente separado en bloques claros.

5.2. HTML

Lenguaje de marcas de hipertexto (HyperText Markup Language) hace referencia al lenguaje de marcado para la realización de páginas web.

Realmente, no se trata de un lenguaje de programación si no de un lenguaje de marcado que define la estructura de su contenido. Mediante etiquetas de encierre se define el contenido según como quieres que se vea o se comporte a la hora de mostrarlo.

Estas etiquetas pueden hacer de una palabra o una imagen, un hipervínculo a otro sitio, puedes crear tablas, poner letras en cursiva, etc… Permite al usuario crear y estructurar secciones, párrafos, encabezados, enlaces y elementos para páginas web y aplicaciones.

HTML nos ofrece una gran adaptabilidad, una estructuración lógica y una fácil interpretación tanto por humanos como por máquinas.

Se considera el lenguaje web más importante, ya que, ha sido el motivo de la aparición, desarrollo y expansión de la World Wide Web (www).
Se ha impuesto como estándar para las visualizaciones de páginas web en todos los navegadores.

5.3. SQL

El lenguaje de consulta estructurada (Structured Query Language) es un lenguaje de dominio específico, utilizado en programación para administrar y recuperar información de sistemas de gestión de bases de datos relacionales.
Ayuda a solucionar problemas relacionados con la definición, manipulación e integridad de la información.

Se trata de un lenguaje de alto nivel orientado a conjuntos de registros, por lo que con una única consulta puede devolver cantidades de datos.

Lo más interesante de este lenguaje es que alguno de los aspectos están basados en cálculo relacional y otros en álgebra relacional, con el fin de hacer consultas y recuperar de una forma más sencilla la información de la base de datos y realizar cambios en ellas.

Gracias a SQL, en una base de datos se puede:

- Consultar, actualizar y organizar datos.
- Crear, modificar y borrar la estructura.
- Controlar los accesos.

Capítulo 4. Objetivos

En este contexto, el objetivo principal del proyecto será, que una placa WeMos D1, con un chip ESP8266, sea capaz de recoger los datos a partir de un sensor de temperatura y humedad como lo es el sensor DHT22. A partir de ahí, la placa estará programada para enviar dicha información a un servidor. Dicho servidor será el encargado de guardar los datos en una base de datos, a modo historial. Finalmente, el último paso sería que un servidor web, facilitará al cliente dicha información.

Cada nodo, constará de una placa WeMos D1 con un chip ESP8266 conectada mediante hardware a un DHT22. Cada nodo, deberá estar programado correctamente y ser funcional.

El servidor, que será un servidor UDP, estará a la espera y escucha de los datos entrantes, almacenando así, la información, cada vez que la reciba.

El almacenaje será en una base de datos creada en MySQL y la constituirá una tabla que contenga los datos recibidos, es decir, la temperatura y la humedad, la fecha y la hora a la que se recibieron los datos, ya que continuamente va a estar recibiendolos; y, la dirección ip y el puerto origen, de esta manera se podrá diferenciar de qué nodo proviene dicha información.

En el caso de que alguna de las temperaturas sea sospechosa, se podrá detectar el nodo y el problema en menos tiempo y con menos trabajo.

Para finalizar, la creación de un servidor web, que mantenga una página web en localhost, a la que, quien acceda, podrá observar los datos del último día y si lo requiere, un historial de todos los datos que han llegado al servidor de cada nodo.

Cada nodo, se despertará cada cierto tiempo, se configurará automáticamente, enviará los datos al servidor y se volverá a dormir. De esta manera, habremos conseguido crear una red de nodos que facilitan la información de la temperatura y la humedad durante un tiempo limitado por la pila de la placa.

Sin embargo, al programarla para que se duerma durante largos periodos de tiempo y se despierte únicamente para enviar la información, se trata de un dispositivo de bajo consumo energético, ya que apenas gasta energía, por lo que se tratará de un periodo medianamente largo.

1. Alcance

El alcance de este proyecto consistirá en desarrollar un programa en

Arduino IDE capaz de recoger los datos del sensor de temperatura y humedad y, mediante las librerías que ofrece Arduino, implementar una conexión UDP con el servidor para enviar la información siempre que se recoja.

El servidor UDP desarrollado en Python, tendrá consultas SQL a la base de datos de MySQL para almacenar dicha información.

Finalmente, el servidor web estará implementado en Python, se tratará de una conexión HTTP, ya que se trata de un servicio web, con HTML embebido que será el encargado de la apariencia de la página y de mostrar los datos correctamente.

2. Antecedentes

En la actualidad, la automatización y digitalización de nuestro entorno, es una realidad. Vivimos en un mundo conectado y no podemos quedarnos atrás, debido a esa necesidad es que surgió el Internet de las cosas, que se basa en la conectividad de objetos o dispositivos cotidianos a través de internet.

A medida que avanza la sociedad y la tecnología, avanzan nuestras necesidades. Por ello, la información y los datos son una parte fundamental del aprendizaje. El empirismo se conoce por defender que la base del conocimiento es la experiencia. Es verdad que sin conocimientos teóricos previos es imposible llegar hasta el punto en el que nos encontramos hoy tecnológicamente, pero eso no significa que a partir de cierto punto, la experiencia e información anterior nos ayude a comprender de una forma mejor nuestro entorno, y esto es lo que hace tan necesaria en nuestra vida la conexión entre los diferentes dispositivos. Gracias a eso, tenemos la posibilidad de conocer los datos de los lugares donde no llegamos y tener un historial de ellos.

Capítulo 5. Desarrollo, Implementación y Resultados

En este capítulo se describe el diseño, desarrollo, implementación y resultados obtenidos en el desarrollo del proyecto desde que la placa recoge la información hasta que el servidor web lo muestra.

Para conseguir nuestros objetivos, primero había que investigar qué herramientas nos serían más útiles y sencillas de utilizar.

En primer lugar, necesitamos una placa programable que fuese capaz de conectarse de manera inalámbrica.

Se ha escogido la tecnología WiFi para llevar a cabo la conexión porque a diferencia del Bluetooth soporta la conexión entre varios dispositivos simultáneamente, ya que el Bluetooth está orientado a conexiones de punto a punto.

La principal ventaja de la tecnología WIFI es su ancho de banda, y que se encuentra implantada en muchos lugares con conexión directa a internet. Es el estándar que podemos encontrar en muchos hogares e industrias y facilita mucho la implantación de la tecnología.

Otras tecnologías como Lora y Zigbee que también se plantearon inicialmente, son tecnologías de bajo consumo, con una velocidad de transferencia muy baja y capacidad de cubrir grandes áreas. Estas tecnologías, a pesar de que son más propicias para instalaciones de IoT, haría falta implantarlas en las zonas de pruebas, sin embargo en este proyecto, en el que se pretende desarrollar dispositivos de bajo costo, una de las claves es que el costo de instalación sea lo menor posible, con lo que la tecnología seleccionada es WIFI.

Dentro de la tecnología WIFI, la primera opción sería una placa Arduino UNO corriente con un chip WiFi integrado. Sin embargo, la placa WeMos D1 ofrece todo lo que necesitamos. Se trata de una placa que en comparación con Arduino, tiene mayor capacidad de procesamiento, consume menos energía y es más económica. Por lo que ofrece más prestaciones por menos dinero. Además, WeMos D1 es compatible con el entorno de Arduino para poder programarla, por lo que finalmente, nos decidimos por la placa WeMos D1 con el módulo ESP8266 (el más barato del mercado) integrado por sus numerosas ventajas. Sin embargo no son todo ventajas, solo ofrece un pin analógico (A0) y un número reducido de pines de Entrada/Salida.

Por otro lado, necesitamos un sensor de temperatura y humedad, en este caso, las opciones son numerosas, pero queremos que tengan salida digital, por lo que las opciones se reducen bastante. Además, el factor económico nos hizo reducir a dos posibilidades, un sensor DHT11 o un sensor DHT22, ya que son muy económicos en

comparación al resto y con una buena relación calidad-precio. En este caso, el DHT22 es un poco más caro pero ofrece unos datos más precisos y robustos.

Para programar la placa, primero se realiza la conexión hardware entre el sensor DHT22 y la placa WeMos D1 que se lleva a cabo con unos cables dupont macho-hembra.

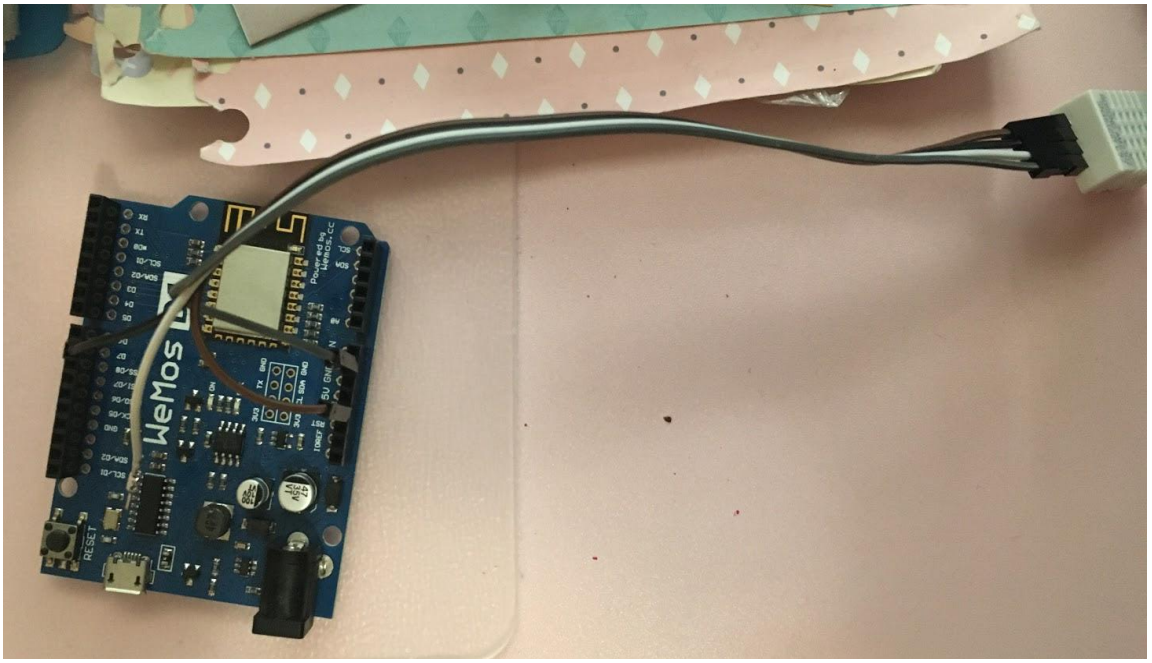

Figura 9: Conexión entre placa WeMos D1 y sensor DHT22

Según los estudios realizados, para que la conexión de la placa y el sensor fuera funcional, era necesario conectar una resistencia al sensor. Sin embargo, se utilizó una alternativa a través de la resistencia interna de pull up del microcontrolador. Por lo que los pines del sensor quedan conectados a la placa así:

- Pin 1 → VCC. Conexión del cable marrón.
- Pin 2 → D7. Conexión del cable negro. Lo conectamos a la entrada digital 7.
- Pin 3 → No se conecta a nada, es el cable blanco.
- Pin 4 → GND. Se conecta a Tierra. Conexión del cable gris.

Una vez tenemos el hardware conectado, enchufamos mediante un cable USB, la placa al ordenador.

Antes de empezar con el desarrollo software de la placa, recordamos que al no tratarse de un Arduino, hay que configurar el entorno Arduino IDE para que sea compatible con nuestra tarjeta.
Para ello, accedemos al gestor de tarjetas y seleccionamos que se trata de una placa WeMos D1.

Para iniciar el proyecto, en primer lugar, habría que añadir las librerías que incluyeran tanto el sensor como nuestro chip ESP8266.

```
#include <DHT.h>
#include <ESP8266WiFi.h>
```

A continuación, inicializamos el sensor, el módulo WiFi ESP8266 y el serial de Arduino. También definimos el pin D7 como el pin de entrada de los datos del sensor.

WIFI DINÁMICA

Para que funcione la conexión entre placa y servidor, se conecta primero la placa a una red WiFi. Queremos que se trate de una WiFi dinámica, es decir, que rastree todas las redes cercanas, las muestre por el serial y en caso de no elegir ninguna de las escaneadas, se elija automáticamente la última red a la que estuvo conectada la placa.

Si queremos que esto sea posible, deberemos hacer uso de la memoria EEPROM. Por ello, debemos añadir la librería que incluye la memoria EEPROM e inicializarla posteriormente.
Cada vez que nos conectemos a una red, se guardará en esta memoria la información relacionada con dicha red, es decir, el nombre de la red y su contraseña.
Si al escanear las redes, no se selecciona ninguna, el programa leerá la zona de memoria en la que se almacenó la anterior vez los datos y se conectará directamente.

Si decides elegir otra red, te mostrará este mensaje por el serial y tendrás que elegir una opción:

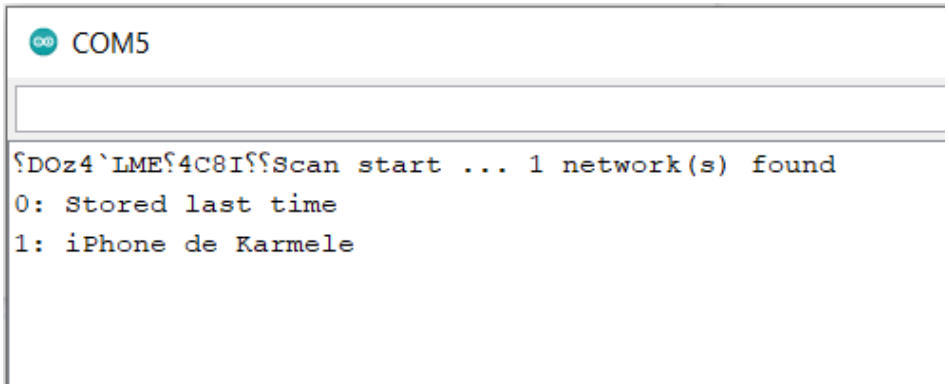

Figura 10: Escaneo de redes en serial de arduino

Se ha declarado un timeout de 10 segundos, lo que quiere decir que si en 10 segundos no recibe ninguna opción, se conectará automáticamente a la anterior.
Si por el contrario, eliges una, se te pedirá la contraseña por el serial, se guardarán los datos en la memoria EEPROM y se conectará a la red.

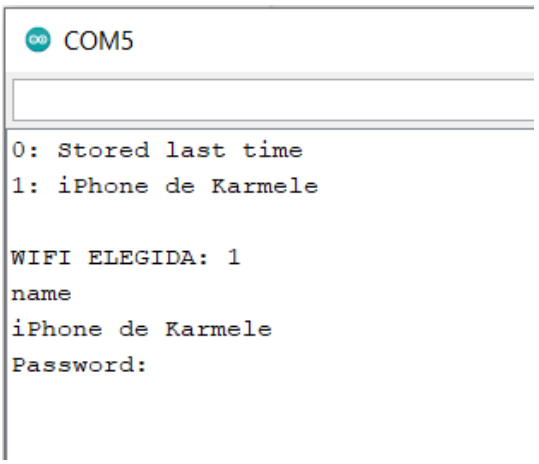

Figura 11: Insertar password en el serial de arduino

Con esto, hemos programado una placa que se conecta dinámicamente al WiFi.

RECOGIDA DE DATOS SENSOR-PLACA

Centrándonos en nuestro objetivo, tenemos que establecer una conexión con el servidor y enviar los datos.

La recogida de datos del sensor, se establece en el pin D7, por lo que usamos las funciones establecidas en la librería DHT para leer tanto la temperatura como la humedad y guardarlas en diferentes variables.

```
// Leemos la humedad relativa
float humedad = dht.readHumidity();
// Leemos la temperatura en grados centígrados (por defecto)
float temperatura = dht.readTemperature();

Serial.println(humedad);
Serial.println(temperatura);
Serial.println(" ");

// Comprobamos si ha habido algún error en la lectura
if (isnan(humedad) || isnan(temperatura) ) {
  Serial.println("ERROR: DHT22 DATA IS NOT AVAILABLE");
}
```

Figura 12: Recogida de datos del sensor de temperatura y humedad

ENVÍO DE DATOS AL SERVIDOR

Entonces, ha llegado el momento de establecer la conexión entre el servidor y la placa. Para ello, hay que decidir el protocolo a utilizar, que en este caso, aplicaremos el protocolo UDP.

El protocolo UDP no nos aporta seguridad a la hora de saber si los datos han llegado al servidor, sin embargo, al ser una aplicación tan sencilla y que estará configurada para que envíe los datos constantemente, no preocupa que se pierda algún paquete, por lo que utilizar el protocolo UDP, que tiene menos sobrecarga, redundará en el consumo final de la aplicación.

El módulo WiFi ESP8266 tiene soporte UDP a través de la librería WifiUDP que da soporte al protocolo UDP, por lo que añadimos:

```
#include <WiFiUDP.h>
```

Y lo inicializamos posteriormente. Esto, nos permitirá establecer una conexión UDP con un servidor UDP funcional. La información será enviada a una dirección IP y a un puerto específicos. En este caso hemos decidido que el servidor estará escuchando en el puerto 1234, por lo que la información se enviará a este.

```
//CONEXIÓN UDP
  udp.begin(port);
  Serial.print("Listening on UDP port");
  Serial.print(port);

  Serial.print("Connecting to ");
  Serial.print(host);
  Serial.print(':');
  Serial.println(port);
```

Figura 13: Estableciendo conexión UDP desde Arduino

Y se envían los datos:

```
// CONEXIÓN UDP
//Envío de datos
udp.beginPacket(host, port);
udp.write(hum);
udp.write(h);
udp.write(temp);
udp.write(t);
udp.write(c);
udp.endPacket();

int packetSize = udp.parsePacket();

// Recepción de datos
if(packetSize){
  Serial.print("Paquete recibido! Size: ");
  Serial.println(packetSize);
  int len = udp.read(packet, 1024);
  if (len>0){
    packet[len] = '\o';
  }
}
```

Figura 14: Envío de datos al servidor UDP

DEEP SLEEP Y AHORRO DE ENERGÍA

Finalmente, uno de los objetivos principales de este proyecto es su alimentación por medio de baterías, con lo que hay que reducir el consumo, entonces, si está encendido continuamente, no lo cumplirá. Por lo tanto, la placa, cada vez que envía la información, se dormirá durante 20 segundos. Al despertarse, tendrá que configurarse automáticamente y volver a enviar los datos y así sucesivamente.

A nivel software, añadimos esta línea de código en el setup, que activará nuevamente la Wifi :

```
WiFi.setSleepMode(WIFI_NONE_SLEEP);
```

Y una vez la información haya sido enviada al servidor:

```
ESP.deepSleep(20e6, RF_DEFAULT);
```

Que hará que se duerma durante 20 segundos.

A nivel hardware, necesitamos un cable dupont que conecte el pin D0 con el pin RST para que las líneas anteriores de código sean funcionales. De esta manera, la llamada a deepSleep, provocará el reseteo de la placa tras el tiempo establecido. Durante el tiempo que la placa se encuentra en deepSleep, el consumo es prácticamente nulo, de unos 20 microAmperios.

SERVIDOR UDP

En segundo lugar, tenemos un servidor que recibirá la información de la placa. Estará programado en Python y usaremos sockets UDP para la recepción de datos.

Python es un lenguaje compatible con casi todas las tecnologías, lo que lo hace más flexible a la hora de programar, además, se define como un lenguaje seguro y más fácil para el programador.

Por tanto, importamos las funciones de sockets y lo configuramos para que se mantenga a la escucha en el puerto 1234, donde la placa estará enviando la información. La dirección IP será: ‘’, de esta manera, el servidor escuchará la información que llegue desde cualquier dirección IP.

La diferencia entre un socket corriente y un socket UDP a la hora de programar es que estos reciben y/o envían datagramas de tamaño limitado, sin tener en cuenta el orden ni tener la fiabilidad de si la información ha llegado, tal y como se define el protocolo UDP.

Procedemos creando el socket UDP, que se pondrá a la espera de recibir datos. Cuando los reciba, los mostrará por pantalla y seguirá escuchando sin cerrar la conexión, puesto que la placa volverá a encenderse y a enviar los datos.

```
#·Create·a·datagram·socket

UDPServerSocket·=·socket.socket(family=socket.AF_INET,·type=socket.SOCK_DGRAM)

#·Bind·to·address·and·ip

UDPServerSocket.bind((localIP,·localPort))

print("UDP·server·up·and·listening")
```

Figura 15: Creación del socket UDP

```
while(True):

····bytesAddressPair·=·UDPServerSocket.recvfrom(bufferSize)

····message·=·bytesAddressPair[0]
····address·=·bytesAddressPair[1]

····m·=·message.decode("utf-8");

····me·=·m.split();
····hum·=·me[1];
····temp·=·me[3];


····clientMsg·=·"Message·from·Client:{}".format(message)
····clientIP··=·"Client·IP·Address:{}".format(address)
····
```

Figura 16: Socket recibiendo datos y mostrándose por pantalla

CREACIÓN DE LA BASE DE DATOS

La información es importante. Hay que tener en cuenta, que la creación de una base de datos en un proyecto como este, es indispensable.

Para que todo esto sea útil, habrá que almacenar estos datos ya sea para obtener un estudio de ellos, unas gráficas o simplemente mostrarlo al cliente.

Por ello, necesitamos un gestor de bases de datos que sea simple de utilizar y nos

permita tanto escribir como leer los datos.

Sobra decir, que la seguridad e integridad de los datos a la misma vez que la eficiencia y la rapidez son características fundamentales.

Los gestores de bases de datos más sencillos son SQLite y MySQL, sin embargo, MySQL es más flexible cuando se trata de los tipos de datos, además tiene una estructura fácilmente escalable y permite el acceso a más de un usuario.

Entonces, el primer paso es instalar MySQL en el ordenador. Una vez listo, acceder a este con el usuario y la contraseña y crear la base de datos usando el lenguaje SQL. Lo único que habría que hacer desde la terminal de mysql, sería la creación de la base de datos y de la tabla con sus correspondientes atributos.

En este caso, será una tabla que contenga la información recibida desde la placa, es decir, el día y la hora, la dirección IP origen y el puerto origen, la temperatura y la humedad.

Una vez creada, se puede empezar a insertar datos en ella.

Nombre de la base de datos: tfg.

Nombre de la tabla: data.

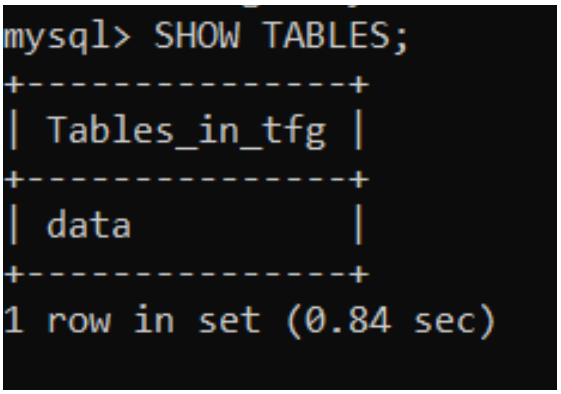

Figura 17: SHOW TABLES en MySQL

ENVÍO DE DATOS DEL SERVIDOR A LA BASE DE DATOS

Una vez montada la base de datos, está preparada para almacenar la información. Para que esto funcione, desde el propio servidor de Python, habrá que insertar los datos mediante consultas SQL.

No obstante hay que importar las funciones incluidas en `mysql.connector` y crear la conexión. Como se trata de una base de datos con suficiente seguridad, hay que incluir el usuario, la contraseña y la base de datos que quieres utilizar para poder leer/modificar. A partir de ahí, cada vez que el servidor reciba datos, generará una setencia con los datos correspondientes a insertar en la base de datos. Posteriormente, hará un commit de la conexión hasta que se vuelva a recibir datos.

```
cnx = mysql.connector.connect(user='root', password='tfgKarmele', database='tfg')
cursor = cnx.cursor()
```

Figura 18: Creación de la conexión entre servidor - BD

```
    add_sql = """INSERT INTO data
    (fecha, tiempo, puerto, direccion_ip, temperatura, humedad)
    VALUES (%s, %s, %s, %s, %s, %s);"""
    add_val = (fecha, hora, localPort, address[0], temp, hum)

    cursor.execute(add_sql, add_val)

    cnx.commit()

cursor.close()
cnx.close()
```

Figura 19: Consulta SQL en servidor

Lo que finalmente daría resultado a una tabla tal que así:

```
mysql> SELECT * FROM data;
+------------+----------+--------+--------------+-------------+---------+
| fecha      | tiempo   | puerto | direccion_ip | temperatura | humedad |
+------------+----------+--------+--------------+-------------+---------+
| 2021-03-02 | 19:59:24 |   1234 | 172.20.10.5  | 22.9*C      | 58.60   |
| 2021-03-02 | 20:01:18 |   1234 | 172.20.10.5  | 25.0*C      | 86.00   |
| 2021-03-02 | 20:01:18 |   1234 | 172.20.10.5  | 26.1*C      | 91.70   |
| 2021-03-02 | 20:01:18 |   1234 | 172.20.10.5  | 23.2*C      | 61.60   |
| 2021-03-02 | 20:01:18 |   1234 | 172.20.10.5  | 23.9*C      | 60.80   |
| 2021-03-02 | 20:01:18 |   1234 | 172.20.10.5  | 23.6*C      | 61.60   |
| 2021-03-02 | 20:01:18 |   1234 | 172.20.10.5  | 23.5*C      | 61.20   |
| 2021-03-02 | 20:01:18 |   1234 | 172.20.10.5  | 23.5*C      | 61.30   |
| 2021-03-02 | 20:01:18 |   1234 | 172.20.10.5  | 23.4*C      | 61.50   |
| 2021-03-02 | 20:01:18 |   1234 | 172.20.10.5  | 23.4*C      | 62.10   |
| 2021-03-02 | 20:01:18 |   1234 | 172.20.10.5  | 23.4*C      | 62.50   |
| 2021-03-02 | 20:01:18 |   1234 | 172.20.10.5  | 23.4*C      | 61.60   |
```

Figura 20: SELECT de la tabla data

SERVIDOR WEB

La última parte del proyecto sería dar la posibilidad a la persona de acceder y obtener los datos recogidos. La idea es que no se necesite tener conocimientos técnicos para poder acceder a ellos, es decir, ofrecer la posibilidad de ver los datos sin necesidad de acceder a la base de datos.

Esta tarea se llevará a cabo mediante el desarrollo de un servidor web HTTP montado en localhost.

Se basa en un programa desarrollado en Python que establecerá una conexión HTTP con localhost y una conexión con la base de datos, de tal manera que, a la misma vez que

recoge la información de la base de datos, la muestra en una tabla en el servidor web.

Para que esto sea posible, hay que importar las funciones de http.server y las que incluyen la conexión con la base de datos como hicimos anteriormente con el servidor UDP.

Esta se conectará en localhost, en un puerto cualquiera, en esta caso 8080 y cuando un cliente se conecte, mostrará la información que haya recopilado de ese mismo día.

Una vez haya establecido la conexión HTTP y con la base de datos, leerá cada línea de la tabla en cuestión y lo imprimirá insertandolo en una línea de código HTML.

```
            for row in records:
                self.wfile.write(bytes("<tr><td>", "utf-8"))
                self.wfile.write(bytes(str(row[0]), "utf-8"))
                self.wfile.write(bytes("</td>", "utf-8"))
                self.wfile.write(bytes("<td>", "utf-8"))
                self.wfile.write(bytes(str(row[1]), "utf-8"))
                self.wfile.write(bytes("</td>", "utf-8"))
                self.wfile.write(bytes("<td>", "utf-8"))
                self.wfile.write(bytes(str(row[2]), "utf-8"))
                self.wfile.write(bytes("</td>", "utf-8"))
                self.wfile.write(bytes("<td>", "utf-8"))
                self.wfile.write(bytes(str(row[3]), "utf-8"))
                self.wfile.write(bytes("</td>", "utf-8"))
                self.wfile.write(bytes("<td>", "utf-8"))
                self.wfile.write(bytes(str(row[4]), "utf-8"))
                self.wfile.write(bytes("</td>", "utf-8"))
                self.wfile.write(bytes("<td>", "utf-8"))
                self.wfile.write(bytes(str(row[5]), "utf-8"))
                self.wfile.write(bytes("</td></tr>", "utf-8"))
```

Figura 21: Mostrando los datos de la base de datos en la web

A lo que finalmente, si escribimos en nuestro navegador web: `localhost:8080`. Nos mostrará:

Temperatura y Humedad

DATOS DE HOY

FECHA	HORA	PUERTO	DIRECCION IP	TEMPERATURA	HUMEDAD
2021-03-02	19:59:24	1234	172.20.10.5	22.9*C	58.60
2021-03-02	20:01:18	1234	172.20.10.5	25.0*C	86.00
2021-03-02	20:01:18	1234	172.20.10.5	26.1*C	91.70
2021-03-02	20:01:18	1234	172.20.10.5	23.2*C	61.60
2021-03-02	20:01:18	1234	172.20.10.5	23.9*C	60.80
2021-03-02	20:01:18	1234	172.20.10.5	23.6*C	61.60
2021-03-02	20:01:18	1234	172.20.10.5	23.5*C	61.20
2021-03-02	20:01:18	1234	172.20.10.5	23.5*C	61.30
2021-03-02	20:01:18	1234	172.20.10.5	23.4*C	61.50
2021-03-02	20:01:18	1234	172.20.10.5	23.4*C	62.10
2021-03-02	20:01:18	1234	172.20.10.5	23.4*C	62.50
2021-03-02	20:01:18	1234	172.20.10.5	23.4*C	61.60
2021-03-02	20:01:18	1234	172.20.10.5	23.3*C	61.80
2021-03-02	20:01:18	1234	172.20.10.5	23.2*C	62.00
2021-03-02	20:01:18	1234	172.20.10.5	23.2*C	62.10
2021-03-02	20:01:18	1234	172.20.10.5	23.2*C	62.10

Figura 22: Tabla de resultados

Capítulo 6. Conclusiones y líneas futuras

1. Conclusiones

El Internet de las Cosas es un campo muy completo y lleno de nuevas tendencias, ideas y futuro. Es muy importante tener en cuenta el factor de que estamos rodeados por tecnologías conectadas entre sí y cuanto más las conozcamos, más poder sobre ellas tenemos.

Es importante que la conexión entre dos o más dispositivos sea segura, estable, fiable y rápida, de esta manera nos ahorraremos tiempo, dinero y problemas futuros.

En este proyecto, he realizado la interconexión desde una placa que recoge datos, en este caso, temperatura y humedad, pero podría utilizarse para cualquier otro tipo de aplicación, siempre y cuando se cambie el programa de la placa, hasta un servidor web que muestra la información.

Por un lado, hemos desarrollado la conexión hardware de la placa WeMos D1 con el sensor DHT22.

Y por otro lado, hemos desarrollado la conexión software mediante WiFi entre la placa y el servidor UDP, entre el servidor UDP y la base de datos en mySQL y entre el servidor web y la base de datos también.

La finalidad de este TFG es crear esta conexión de la que hablábamos con un dispositivo de bajo consumo. Ambas finalidades han sido conseguidas con éxito, de tal manera que si un cliente solicita querer saber la temperatura de varios lugares a la vez, este proyecto funciona y él podría obtener los datos desde el servidor web.

La elaboración de este proyecto, me ha permitido ampliar mis conocimientos sobre programación, tanto de Python, como del lenguaje de Arduino, como HTML. También he aprendido, sobre la historia del Internet de las cosas, sobre las conexiones entre distintas redes, conexiones servidor-cliente, y he entendido más a fondo los conceptos que he aprendido durante la carrera.

Además, he refrescado conceptos teóricos como los protocolos, el uso de los lenguajes, bases de datos, etc.

2. Líneas futuras

A pesar de que el proyecto ha completado los objetivos que tenía previstos, existen una serie de mejoras que se pueden aplicar, para poder hacerlo más realista, escalable y usable. A continuación, se describen algunas de ellas:

a. Página Web

Para hacer este proyecto dirigido más a una interfaz humana, se podría crear una página web que muestre los datos de una manera más visible y creativa a los ojos de las personas. Sin embargo, tenemos un servidor web montado en localhost.

b. Avisos por cambios bruscos

Al tratarse de una aplicación que recoge datos que podrían ser muy importantes, una buena mejora sería que avisara siempre que ocurrieran cambios drásticos de un un envío a otro. Estos avisos podrían ser al móvil, email…

Simplemente con poder acceder a la base de datos y comparar aquellos que tienen la misma IP y puerto de origen. Si hay un cambio de temperatura o humedad muy brusco, puede tratarse de un incendio u otro incidente, por lo que no estaría de más tener control sobre los datos.

c. Hardware más óptimo

Finalmente, el hardware que proporcionamos no es el más óptimo ya que carece de carcasa y protección en general. Los cables no aseguran su conexión puesto que con cualquier movimiento de la placa, pueden desprenderse.

Una posible solución a esto, podría ser una carcasa que lo protegiera del entorno, golpes, etc. Y otra sería soldar los cables a los pines, de esta manera, no se soltarán.

Capítulo 7. Summary and Conclusions

1.Summary

The Internet of Things is a new trend, new ideas and a future field that has to develop more than we think. It's very important to know that around us everything is connected to technology and the more we know about them, the more power we have over them.
A connection between two or more devices must be a secure, stable, reliable and fast connection, that way, we will save time, money and future problems.

In this project, we have made the interconnection from a board that collects data, in this case, temperature and humidity, but could be used for any other application, as long as the program of the board is changed, to a web server that shows its information in real time.

On the one hand, a hardware connection between WeMos D1 board and DHT22 sensor has been developed.

And on the other hand, software has been developed too. The connection between the board and UDP server, also between the UDP server and the database, and the web server and database as well.

The purpose of this project is to create the connection we were talking about before with a low-power device. Both purposes have been successfully achieved, in such a way that if a client requests to want to know the temperature of different places to make an experiment, the project would work perfectly and he could obtain the results from the web server.

The development of this project has helped me to expand my knowledge of programming, both in Python and the Arduino language, such as HTML. I have also learned about IoT history, connections between different networks and devices, server-client communication and I have learned more about some concepts that I have learnt during my grade.

2. Future work plans

Despite the fact that the project has achieved its objectives, there are a number of improvements that can be applied to make it more realistic, scalable and usable. Some of them are described below:

a. Website

To make this project aimed more at a human interface, you could create a website that displays the data in a more creative way to people's eyes. However, we have a web server mounted on localhost.

b. Setting alerts when big changes comes

Collecting data is a very important thing, a good improvement would be to warn whenever big changes occur from one shipment to another. These alerts could be to the mobile, email ...
Simply by being able to access the database and compare those data that have the same source IP and source port, if there is a very sudden change in temperature or humidity, it could become a fire or another incident, so it would be useful to control this kind of data.

c. Hardware optimizing

Finally, our hardware is not the most optimal since it lacks a casing, protection in general and dupont wires don’t ensure their connection since with any slight movement of the board, they can detach.
So a possible solution to this could be a case that protects it from the environment, blows, etc. Another would be to solder wires to pins, in this way, they will not come loose.

Capítulo 8. Presupuesto

En este apartado calcularemos el presupuesto de los materiales y del desarrollo para la elaboración de este proyecto, teniendo en cuenta las horas de investigación previas como una persona con experiencia en el campo.

1. Material

El hardware en este caso sería barato y escaso.

Material	Coste
Placa WeMos D1 con ESP8266	10,99€
Cables dupont (20 unidades)	1€
Sensor DHT22	8,99€
Total	**20,98€**

Tabla 2: Estimación de precios de los materiales

2. Desarrollo

En la tabla de a continuación se especifican las horas invertidas aproximadamente en la realización de cada tarea, tomando como referencia que el coste por hora se calcula en base a 10€.

Tareas	Horas empleadas
Revisión bibliográfica y estado del arte	60 horas
Análisis y prueba de las herramientas a utilizar	45 horas
Diseño, desarrollo y pruebas de la aplicación	150 horas
Documentación de la memoria del proyecto	65 horas
Total	**320 horas**

Tabla 3: Estimación Tareas/Horas

Finalmente el presupuesto de dicho trabajo, por un único nodo funcional, teniendo en cuenta el trabajo de desarrollo, el presupuesto final se detalla a continuación:

Tareas	Descripción	Coste
Programación/Desarrollo informático	320 horas x 10€/hora	3200 €
Material	Coste de las herramientas	20,98 €
	Total	**3220,98€**

Tabla 4: Estimación Tareas/Horas

Cabe destacar que el uso de los materiales ha sido facilitado por el tutor de la asignatura, en este caso.

Apéndice 1. Programa Arduino IDE

El programa que se ejecuta en el entorno Arduino IDE para que la placa cumpla correctamente con su funcionalidad:

```
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <WiFiUDP.h>
#include <EEPROM.h>

// Definimos el pin digital donde se conecta el sensor
static const uint8_t D7   = 13;
#define DHTPIN D7

// Dependiendo del tipo de sensor
#define DHTTYPE DHT22

//Declaramos las variables globales
String ssid1;
String password1;

char ssid[30];
char pass[30];

const char* host = "172.20.10.4";
const uint16_t port = 1234;

// Inicializamos el sensor DHT11
DHT dht(DHTPIN, DHTTYPE);

//Inicializamos UDP
WiFiUDP udp;
char packet[1024];

//-------------------------------------SETUP---------------------------------
void setup() {

  // Inicializamos comunicación serie
  Serial.begin(9600);
```

```
const int timeout=10000;
Serial.setTimeout(timeout);

// Inicializamos el sensor DHT
dht.begin();

//Inicializamos la EEPROM
EEPROM.begin(512);

//Sleep mode WIFI
WiFi.setSleepMode(WIFI_NONE_SLEEP);

// Escaneo de las redes
Serial.print("Networks scan starting ... ");
int n = WiFi.scanNetworks();
if (n > 0){
  Serial.print(n);
  Serial.println(" networks found");
  Serial.println("0: Stored on EEPROM");
  for (int i = 0; i < n; i++)
  {
    Serial.print(i+1);
    Serial.print(": ");
    Serial.print(WiFi.SSID(i));
    Serial.println("");
  }
  Serial.println();
} else {
  Serial.println("No networks found");
}
delay(5000);

// Lee la elección de red
if (Serial.available() > 0){

  int chosen_wifi = Serial.parseInt();
  if (Serial.read() == '\n') {
    Serial.print("Chosen network: ");
    Serial.println(chosen_wifi);
  }
if (chosen_wifi != 0){

    for ( int i = 0; i <= n; i++)
    {
      if (i == chosen_wifi){
        Serial.println(WiFi.SSID(i-1));
```

```
        ssid1 = WiFi.SSID(i-1);
      }
    }

    // Lee el password de la red
    Serial.println("Password: ");
    password1 = Serial.readString();

  // Formateamos los datos
  password1.trim();
  ssid1.toCharArray(ssid, sizeof(ssid));
  password1.toCharArray(pass, sizeof(pass));

  for ( int i = 0; i < sizeof(ssid); i++){
    EEPROM.put(i, ssid[i]);
  }
  int val = 0;
  for (int j = 50; j < 80; j++){
    EEPROM.put(j, pass[val]);
    val++;
  }

  // Conectando a la wifi
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.print(ssid);
  Serial.print(pass);

  WiFi.begin(ssid, pass);
  }
}else {

  // Conectando a la wifi
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  for(int i=0; i < sizeof(ssid);i++){
    EEPROM.get(i, ssid[i]);
    Serial.print(ssid[i]);
  }

  Serial.println("");
  int val = 0;
  for (int j=50; j<80;j++){
```

```
      EEPROM.get(j, pass[val]);
      Serial.print(pass[val]);
      val++;
    }
    int tam = sizeof(pass) + 50;

    WiFi.begin(ssid, pass);
  }
  EEPROM.commit();

    // Conectando a la wifi

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.print(WiFi.localIP());
  Serial.println("/");

  delay(500);

//CONEXIÓN UDP
  udp.begin(port);
  Serial.print("Listening on UDP port");
  Serial.print(port);

  Serial.println("");

  Serial.print("Connecting to ");
  Serial.print(host);
  Serial.print(':');
  Serial.println(port);

 //----------

  char h[6];
  char t[6];
  // Leemos la humedad relativa
  float humedad = dht.readHumidity();
  // Leemos la temperatura en grados centígrados (por defecto)
  float temperatura = dht.readTemperature();
```

```
Serial.println(humedad);
Serial.println(temperatura);
Serial.println(" ");

// Comprobamos si ha habido algún error en la lectura
if (isnan(humedad) || isnan(temperatura) ) {
  Serial.println("ERROR: DHT22 DATA IS NOT AVAILABLE");
}

// Esperamos 10 segundos entre medidas
delay(1000);

// Convertimos a string los datos
dtostrf(humedad, 4, 2, h);
dtostrf(temperatura, 5, 1, t);

char hum[] = "Humedad: ";
char temp[] = "            Temperatura: ";
char c[] = "*C";

// CONEXIÓN UDP
//Envío de datos
udp.beginPacket(host, port);
udp.write(hum);
udp.write(h);
udp.write(temp);
udp.write(t);
udp.write(c);
udp.endPacket();

int packetSize = udp.parsePacket();

// Recepción de datos
if(packetSize){
  Serial.print("received packet! Size: ");
  Serial.println(packetSize);
  int len = udp.read(packet, 1024);
  if (len>0){
    packet[len] = '\o';
  }
}

delay(200);

ESP.deepSleep(20e6, RF_DEFAULT);
```

```
}


//---------------------------------LOOP-------------------------------------
void loop() {



}
```

Apéndice 2. Servidor UDP.

Programa que recibe la información desde la placa y la almacena en la base de datos. Desarrollado en Python. Ejecuta un servidor UDP.

```
import socket
from datetime import datetime
import mysql.connector

fecha = datetime.now().date()
hora = datetime.now().time()

localIP = ''
localPort = 1234
bufferSize  = 1024

msgFromServer = "Hello UDP Client"
bytesToSend = str.encode(msgFromServer)

# Create a datagram socket
UDPServerSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)

# Bind to address and ip
UDPServerSocket.bind((localIP, localPort))
print("UDP server up and listening")

# Listen for incoming datagrams
cnx = mysql.connector.connect(user='root', password='tfgKarmele', database='tfg')
cursor = cnx.cursor()

while(True):
```

```
    bytesAddressPair = UDPServerSocket.recvfrom(bufferSize)
    message = bytesAddressPair[0]
    address = bytesAddressPair[1]

    m = message.decode("utf-8");

    me = m.split();
    hum = me[1];
    temp = me[3];

    clientMsg = "Message from Client:{}".format(message)
    clientIP  = "Client IP Address:{}".format(address)

    # Sending a reply to client
    UDPServerSocket.sendto(bytesToSend, address)

    add_sql = """INSERT INTO data
    (fecha, tiempo, puerto, direccion_ip, temperatura, humedad)
    VALUES (%s, %s, %s, %s, %s, %s);"""
    add_val = (fecha, hora, localPort, address[0], temp, hum)

    cursor.execute(add_sql, add_val)

    cnx.commit()

cursor.close()
cnx.close()
```

Apéndice 3. Servidor web

Programa que establece la conexión HTTP en localhost, consulta los datos en la base de datos y los muestra en la web. Servidor web desarrollado en Python con fragmentos en HTML para la visualización de la web.

```
from http.server import BaseHTTPRequestHandler, HTTPServer

import mysql.connector

from datetime import datetime


hostName = "localhost"

serverPort = 3152


class MyServer(BaseHTTPRequestHandler):

    def do_GET(self):

        self.send_response(200)

        self.send_header("Content-type", "text/html")

        self.end_headers()


self.wfile.write(bytes("<html><head><title>https://TFG_KARMELE</title></head>",
"utf-8"))

         self.wfile.write(bytes("<style> h2 { text-align: center;} table, th, td
{  border:  1px  solid  black;  border-collapse:  collapse;}  table.center  {
margin-left: auto; margin-right: auto;} th, td { padding: 5px;} th { text-align:
center;} #t01 { width: 100%; background-color: #f1f1c1;} #t01 tr:nth-child(odd)
{  background-color:  #fff;}  #t01  th  {  background-color:  black;  color:  white;}
</style>", "utf-8"))

           self.wfile.write(bytes("<body> <h2>Temperatura y Humedad</h2> <table
class='center'  id='t01'  style='width:50%'>  <caption>DATOS  DE  HOY</caption>",
"utf-8"))


self.wfile.write(bytes("<tr><th>FECHA</th><th>HORA</th><th>PUERTO</th><th>DIRECC
ION IP</th><th>TEMPERATURA</th><th>HUMEDAD</th>", "utf-8"))
```

```
        try:
                  cnx = mysql.connector.connect(host = 'localhost',user='root',
password='tfgKarmele', database='tfg')
                sql_select_query = """select * from data where fecha = (SELECT
curdate())"""

            cursor = cnx.cursor()
            cursor.execute(sql_select_query)

            # get all records
            records = cursor.fetchall()
            print("Total number of rows in table: ", cursor.rowcount)

            print("\nPrinting each row")
            i=0
            for row in records:
                self.wfile.write(bytes("<td>", "utf-8"))
                self.wfile.write(bytes(str(row[0]), "utf-8"))
                self.wfile.write(bytes("</td>", "utf-8"))
                self.wfile.write(bytes("<td>", "utf-8"))
                self.wfile.write(bytes(str(row[1]), "utf-8"))
                self.wfile.write(bytes("</td>", "utf-8"))
                self.wfile.write(bytes("<td>", "utf-8"))
                self.wfile.write(bytes(str(row[2]), "utf-8"))
                self.wfile.write(bytes("</td>", "utf-8"))
                self.wfile.write(bytes("<td>", "utf-8"))
                self.wfile.write(bytes(str(row[3]), "utf-8"))
                self.wfile.write(bytes("</td>", "utf-8"))
                self.wfile.write(bytes("<td>", "utf-8"))
                self.wfile.write(bytes(str(row[4]), "utf-8"))
                self.wfile.write(bytes("</td>", "utf-8"))
                self.wfile.write(bytes("<td>", "utf-8"))
                self.wfile.write(bytes(str(row[5]), "utf-8"))
                self.wfile.write(bytes("</td></tr>", "utf-8"))
```

```
                i+1

            self.wfile.write(bytes("</table></body></html>","utf-8"))


            cursor.close()
            cnx.close()

        except mysql.connector.Error as e:
            print("Error reading data from MySQL table", e)
        finally:
            if cnx.is_connected():
                cnx.close()
                cursor.close()
                print("MySQL connection is closed")

if __name__ == "__main__":
    webServer = HTTPServer((hostName, serverPort), MyServer)
    print("Server started http://%s:%s" % (hostName, serverPort))

try:
    webServer.serve_forever()
except KeyboardInterrupt:
    pass


    webServer.server_close()
    print("Server stopped.")
```

Bibliografía

General:

https://www.redhat.com/es/topics/internet-of-things/what-is-iot

https://www.adslzone.net/reportajes/tecnologia/estandares-conexion-inalambrica/

Servidor UDP:

https://www.luisllamas.es/esp8266-protocolo-udp/

https://stackoverflow.com/questions/39314086/what-does-it-mean-to-bind-a-socket-to-any-address-other-than-localhost/39314221

https://blog.educacionit.com/2017/12/22/php-vs-python-cual-es-mejor-para-programacion-del-lado-del-servidor/

https://www.bejob.com/7-razones-para-programar-en-python

http://www.chuidiang.org/clinux/sockets/udp/udp.php

ESP8266:

https://www.esp8266.com/

https://www.luisllamas.es/tag/esp8266/

https://www.luisllamas.es/guia-de-programacion-del-esp8266-en-entorno-arduino/

https://www.luisllamas.es/programar-esp8266-con-el-ide-de-arduino/

https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/udp-examples.html

https://www.arduino.cc/en/Tutorial/WiFiSendReceiveUDPString

https://electronicsprojectslab.wordpress.com/trucos-recomendaciones-o-consejos-con-esp8266/

https://techtutorialsx.com/2017/02/25/esp8266-scanning-wifi-networks/

https://programarfacil.com/esp8266/esp8266-deep-sleep-nodemcu-wemos-d1-mini/

https://es.wikipedia.org/wiki/EEPROM

Base de datos:

https://guiadev.com/mysql-vs-sqlite/

https://fp.uoc.fje.edu/blog/por-que-elegir-el-gestor-de-base-de-datos-mysql/