Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería en
Electrónica Industrial y Automática

DISEÑO DE UN MEDIDOR IOT DOMÉSTICO

Autor:
Olivar Ruiz, Paloma

Tutor:
González de la Fuente, José Manuel
Dpto. Tecnología Electrónica
Valladolid, Abril 2021

Resumen y Palabras Clave

Diseño e implementación de un sistema de medida de temperatura y humedad para el hogar basado en Internet de las cosas, con conexión WiFi, y la posibilidad de acceder a los datos desde el móvil a partir de una aplicación Android.

El sistema de medida está constituido por la placa de desarrollo NodeMCU, que contiene el módulo ESP8266, y el sensor de temperatura y humedad DHT22. Para la visualización y el control de los datos se ha trabajado con la plataforma IoT ThingSpeak, y se ha diseñado una aplicación móvil utilizando el entorno de desarrollo MIT App Inventor.

Internet de las cosas, NodeMCU, Sensor, Plataforma IoT, Android

Abstract and Keywords

Design and implementation of a temperature and humidity measurement system for the home based on the Internet of Things, with WiFi connection, and the possibility of accesing the data from the mobile with an Android application.

The measurement system consists of the NodeMCU development board, which contains the ESP8266 module, and the DHT22 temperature and humidity sensor. For the visualization and control of the data, the IoT ThingSpeak plataform has been used, and a mobile application has been designed using the MIT App Inventor development environment.

Internet of Things, NodeMCU, Sensor, IoT plataform, Android

INDICE

ILUSTRACIONES

1. Introducción y objetivos

1.1. Introducción y descripción del proyecto

Hoy en día el Internet de las cosas es un concepto que forma parte de la vida de muchas personas, su aplicación y desarrollo están en pleno auge, por lo que es complicado imaginar un mundo sin dispositivos que empleen esta tecnología. Incluso en los próximos años se espera que su uso se extenderá masivamente cumpliendo con el objetivo de mejorar y facilitar la vida de un mayor número de usuarios, si se gestiona de manera adecuada.

Se diseñará un medido IoT de ámbito doméstico, concretamente un medidor de temperatura y humedad, a partir de un sensor y una placa de desarrollo NodeMCU. Este dispositivo se podrá conectar a través de WiFi a una plataforma IoT enviando datos y permitiendo al usuario visualizarlos en tiempo real.

Además, se creará una aplicación móvil a través de la cual, será posible visualizar los datos e interactuar con el dispositivo implementando alguna orden.

1.2. Objetivos de este TFG

Los principales objetivos de este proyecto son, realizar un estudio del concepto IoT, entender la relación que tiene con la domótica, y aprender a diseñar un medidor que funcione con esta tecnología.

Para diseñar el medidor se necesitará realizar un estudio previo sobre diferentes placas de desarrollo y sensores de temperatura y humedad que se pueden encontrar en el mercado, con el objetivo de escoger los que más se adecuen a las necesidades de la aplicación.

Se llevará a cabo un estudio sobre diferentes plataformas IoT, para posteriormente, hacer una comparativa entre ellas y elegir la que más posibilidades puede ofrecer al diseño.

Otro de los objetivos es desarrollar una aplicación móvil que permita visualizar los datos e interactuar con el dispositivo IoT, para lo que será necesario familiarizarse con la herramienta MIT App Inventor antes de implementar la aplicación Android.

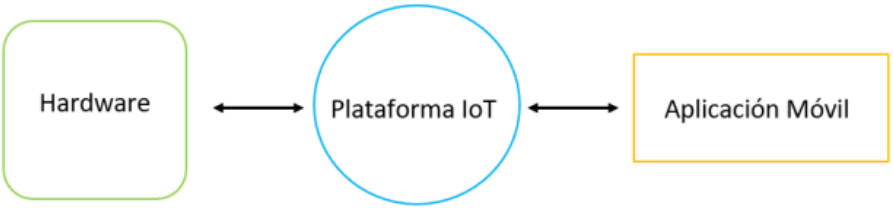


Ilustración 1.- Diagrama de objetivos

1.3. Contenidos de la memoria

En este apartado se van a introducir los diferentes capítulos en los que se divide este TFG, incluyendo una breve descripción de su contenido:

- **Capítulo 1**, este apartado consiste en una introducción y descripción del propio trabajo y los objetivos a llevar a cabo.
- **Capítulo 2**, en este apartado se realizará una introducción tanto del concepto IoT como de domótica, incluyendo algunos ejemplos comerciales y las líneas futuras que se esperan de ambos.
- **Capítulo 3**, se llevará a cabo un estudio sobre el hardware que formará el sistema de medida.
- **Capítulo 4**, se realizará un estudio sobre algunas de las plataformas IoT que existen, y se elegirá la más adecuada para el proyecto después de llevar a cabo una comparación entre ellas.
- **Capítulo 5**, se estudiarán algunos de los entornos de desarrollo para diseñar aplicaciones Android, y se elegirá el más adecuado.
- **Capítulo 6**, se describirán tanto el medio de transmisión como el protocolo de comunicación para el envío de datos a la plataforma IoT.
- **Capítulo 7**, en este apartado se realizará el propio diseño del medidor IoT, se llevará a cabo la implementación del hardware, la plataforma IoT y la aplicación Android.
- **Capítulo 8**, se expondrá la conclusión final del trabajo realizado.
- **Capítulo 9**, en este apartado se explicarán algunas de las posibles líneas futuras de investigación.
- **Capítulo 10,** en este último apartado se añadirá toda la bibliografía que ha sido utilizada para el desarrollo del trabajo.

2. Conceptos fundamentales

En este apartado se definirá el concepto de IoT, cómo surgió el término y cuáles son las principales aplicaciones que se pueden encontrar en la actualidad, concretamente en el ámbito de la domótica.

Se nombrarán algunos ejemplos que existen actualmente en el mercado y, por último, se expondrá lo que se espera que ocurra con esta tecnología en los próximos años.

2.1. Introducción a IoT

El internet de las cosas o IoT (Internet of Things, por sus siglas en inglés) es un concepto que se refiere a la interacción entre objetos cotidianos con Internet, empleando para ello sensores y dispositivos que interactúan y forman parte del mundo real.

El término IoT surgió a finales del siglo XX y fue propuesto por Kevin Ashton, directivo de Procter & Gamble, concretamente en el Auto-ID Center, en el Instituto de Tecnologías de Massachussetts (MIT), donde se realizó una investigación sobre la identificación por radiofrecuencia en red y tecnologías de sensores [1].

Actualmente hay millones de dispositivos conectados entre sí a través de las redes de comunicación, y la cantidad de datos que circulan a través de ellas crece exponencialmente.

El objetivo de esta tecnología es mejorar la funcionalidad de los objetos que están permanentemente conectados a Internet, y con ello, mejorar y facilitar la vida de las personas. Por esta razón, IoT está cambiando la forma de hacer negocios, la organización de sectores como el público e incluso el día a día de las personas.

A continuación, se van a enumerar y explicar algunas de las aplicaciones IoT más empleadas y que están generando un cambio significativo en el mundo que nos rodea [2].

Casas Inteligentes (Smart Homes)

Esta aplicación es una de las más populares y con mayor éxito actualmente. La razón de que esto sea así es que resulta más accesible y asequible para los consumidores, ya que cualquier dispositivo del hogar que emplee electricidad puede ser conectado a la red doméstica.

Algunos ejemplos pueden ser, termostatos inteligentes, lavadoras inteligentes o sistemas de seguridad inteligentes.

Estas aplicaciones permiten al consumidor tener un control del consumo energético de su hogar, gracias a que podemos obtener toda la información necesaria de los dispositivos conectados.

La domótica cobra gran importancia en este caso, ya que es posible configurar los dispositivos para que funcionen de forma autónoma dependiendo de la situación en la que se encuentre, sin necesidad de controlarlo de forma manual.

Wearables

En este bloque se pueden encontrar gafas virtuales o pulseras fitness que monitorean el gasto calórico o el ritmo cardiaco, entre otros.

Esto permite a las personas obtener información a un nivel básico sobre su estado físico y de salud mientras llevan a cabo sus acciones cotidianas.

Ciudades Inteligentes (Smart Cities)

Esta idea tiene como objetivo mejorar la calidad de vida en las ciudades, ya que están creciendo y cada vez están más pobladas, por esta razón están evolucionando a ciudades inteligentes. Aunque es algo que aún está en desarrollo, se espera que para 2025 puedan existir alrededor de 40 ciudades inteligentes en todo el mundo [3]

Algunos ejemplos que podemos encontrar en ciudades inteligentes son:

- Aparcamiento inteligente
- Congestión del trafico
- Iluminación inteligente
- Gestión de residuos

Medioambiente inteligente

La monitorización del estado en el que se encuentra el medio ambiente es importante para poder evaluar en qué condiciones se encuentra y poder actuar en consecuencia.

Gestionarlo es esencial para tener un consumo eficiente de recursos y evitar residuos de fábricas o vehículos.

Además de esto, se puede obtener información acerca de la contaminación del agua o del aire, la radiación o la temperatura, sobre todo en las ciudades, para poder actuar con el objetivo de lograr un entorno más verde y sostenible. Con esto se mejoraría la calidad de vida de las personas y sería posible evitar algunos desastres naturales que pudieran producirse en el futuro.

Industria y fabricación inteligente

El término Industria inteligente se conoce como Industria 4.0, ya que se considera la cuarta revolución industrial.

Se incorporan sensores IoT, wearables e inteligencia artificial que permiten disminuir tiempos de proceso, mejorar la seguridad de los trabajadores y generar menos emisiones al medioambiente. En definitiva, el objetivo de emplear esta tecnología en la industria es mejorar la producción total obteniendo mayores beneficios de una forma respetuosa con el medioambiente y los trabajadores.

2.2. Aplicaciones de IoT en domótica

La domótica se puede definir como el conjunto de tecnologías que hacen posible la automatización de procesos en determinados ámbitos.

La combinación de la domótica e IoT permite controlar los dispositivos que se hayan configurado para ello.

Generalmente el ámbito en el que más se aplica la domótica es en los hogares y edificios, siendo estos los entornos en los que empezó a desarrollarse más fuertemente.

Un sistema domótico está constituido por sensores que permiten captar la información, un controlador para gestionar la información recibida del sensor, actuadores que reciben y ejecutan las ordenes cambiando las características del entorno, buses o medios de comunicación para transmitir la información y una interfaz que permite mostrar la información al usuario e incluso hace posible la interacción con el sistema.

El principal objetivo de la domótica es mejorar y facilitar la vida de los usuarios, algunos de los campos que abarca se exponen a continuación [4]:

Eficiencia energética

La gestión de la energía se puede considerar la aplicación más importante y más empleada de la domótica. El objetivo en este caso es utilizar la energía de una vivienda de una forma más eficaz, pudiendo obtener un ahorro energético considerable.

Para llevar a cabo este objetivo, la domótica se centra en realizar un control sobre los dispositivos dentro de un inmueble que permiten dicho ahorro energético, como, por ejemplo, el control de la temperatura, de la caldera o de la iluminación.

Confort

El objetivo de la domótica en esta aplicación es controlar todo el inmueble para que sea un lugar más cómodo según las necesidades de cada usuario, esto incluye que las tareas más repetitivas del hogar se realicen automáticamente solas, también apagar o encender luces, subir o bajar persianas desde cualquier lugar y en cualquier momento, siendo posible hacerlo de forma automática, pero sin perder el control manual.

Algunos ejemplos típicos son: control de riego, control de iluminación o control de la temperatura.

Seguridad

También es posible emplear la domótica para proteger tanto a personas como los inmuebles. Se puede obtener información sobre el estado de las puertas, ventanas, sensores a la entrada y la salida de las viviendas para poder prevenir y actuar si es necesario.

Gracias a la domótica es posible supervisar el hogar a distancia y programar respuestas ante posibles detecciones de intrusión en el sistema de seguridad.

Accesibilidad

Actualmente la domótica pretende desarrollar entornos más accesibles, para que cualquier persona pueda utilizar todos los elementos del hogar independientemente del grado de discapacidad que tenga.

Esta aplicación en los últimos años ha ganado mucha importancia, ya que consigue que la tecnología pueda estar al alcance de todas las personas.

Un ejemplo conocido para este objetivo son los sistemas de acción por voz, que hacen posible ejecutar cualquier acción con un comando de voz especifico.

Comunicación

Consiste en la posibilidad de conectarse al hogar y a los dispositivos domóticos que haya dentro de él desde cualquier lugar, permitiendo el control a distancia de la vivienda domótica. Por esta razón, se puede considerar la base de cualquier sistema domótico.

En el siguiente apartado de este TFG se van a exponer algunos ejemplos comerciales de IoT, y también de manera más específica algunos enfocados a la domótica.

2.3. Ejemplos comerciales

Algunas de las empresas IoT más potentes son Google, Microsoft, Amazon, Intel, IBM o CISCO. Hay infinidad de productos que emplean esta tecnología, tanto a nivel industrial como individual para el hogar, a continuación, se van a nombrar algunos de los productos comerciales que se pueden encontrar actualmente en el mercado:

Nest Thermostat E de Google

Termostato inteligente que permite cambiar la temperatura con la aplicación *Nest* desde móvil, Tablet o portátil. También se puede cambiar con comando de voz si se tiene un asistente doméstico [5]

Es un sistema que permite ahorrar energía, ya que incluye un horario ajustable para ello. Detecta cuando no hay nadie en el hogar bajando automáticamente la temperatura para ahorrar energía durante ese tiempo.

El dispositivo *Heat Link E* se conecta al sistema de calefacción y a *Nest Thermostat E* para encender, apagar o modular la calefacción.

Es compatible con diferentes tipos de calderas. Permite conexión WiFi y Bluetooth de bajo consumo. Es compatible con sistema Android y iOS. Es uno de los últimos lanzamientos de Google.

Su precio es de 219 €.

Ilustración 2.- Nest Thermostad E y Heat Link E [5]

Nest Audio de Google

Este es el nuevo altavoz que ha sacado Google al mercado, que además permite controlar un hogar inteligente como, por ejemplo, encender las luces con un simple comando de voz [5]

Funciona con los dispositivos inteligentes compatibles y se configura con la aplicación *Google Home*. Una característica interesante es que está fabricado con un 70 % de plástico reciclado. Permite conectividad con WiFi y Bluetooth. Es compatible con sistemas operativos Android y iOS.

Su precio es 99,99 €.

Ilustración 3.- Nest Audio [5]

Amazon echo

Altavoz inteligente desarrollado por Amazon. Permite realizar diferentes acciones con un simple comando de voz gracias al servicio de asistente personal inteligente controlado por *Alexa*, como, por ejemplo, escuchar música, consultar la previsión del tiempo, configurar alarmas o controlar dispositivos del hogar inteligentes compatibles como luces o cerraduras [6].

Tiene integrado el controlador del hogar digital *Zigbee*, el cual es compatible con dispositivos que admitan la especificación *Zigbee*, como bombillas, cerraduras, sensores, enchufes o interruptores. Permite conectividad WiFi y bluetooth. La aplicación de Alexa es compatible con dispositivos Fire OS, Android y iOS.

Su precio es 99,99 €.

Ilustración 4.- Amazon Echo (4ª generación) [6]

Wyze Bulb

Se trata de una bombilla inteligente y regulable de 60W a un precio bastante asequible [7]

Entre sus características incluye WiFi incorporado, es compatible con *Google Assistant* y *Amazon Alexa*, por lo que permite encender, apagar, ajustar el brillo o cambiar la temperatura del color de la luz con un comando de voz.

Con la aplicación *Wyze* también es posible controlar las luces de forma individual o en grupo.

El precio de una bombilla es 7,99 $.
El precio de un pack de 4 bombillas es 29,99 $.

Ilustración 5.- Wyze Bulb [7]

August WI-FI Smart Lock

Es una cerradura inteligente que permite bloquear o desbloquear la puerta, verificar el estado de la puerta, saber quién entra y sale, de forma remota [8]

Es compatible con Amazon Alexa, Apple HomeKit, Google Assistant y muchas otras plataformas y dispositivos de terceros, por lo que es posible controlar la puerta con un comando de voz.

Tiene WiFi incorporado y es compatible con sistemas operativos iOS y Android.

El control de la puerta se tiene a través de la aplicación móvil *August* o con Apple Watch, que actúan como llave.

La cerradura August tiene DoorSense, un sensor que indica si la puerta está correctamente cerrada y bloqueada, lo que ofrece mayor tranquilidad y seguridad desde cualquier lugar. También permite configurar alertas para saber si el estado de la puerta ha cambiado.

Su precio es 249,99 $.

Ilustración 6.- August WI-FI Smart Lock [8]

2.4. El futuro de IoT

IoT ha cambiado la forma en la que vivimos y se espera que en los próximos años sigan creciendo el número de dispositivos conectados a internet.

Se prevé que para el año 2025 el valor económico total creado a través de IoT podría alcanzar 11,3 billones de dólares en varios sectores y áreas de aplicación [9].

Una de las preocupaciones de las empresas actualmente se encuentra en el ámbito de la ciberseguridad [10], ya que al aumentar el número de dispositivos conectados a internet crece el riesgo de sufrir ciberataques. Por lo que el desarrollo de IoT estará encaminado hacia esta rama en los próximos años.

Sin embargo, unos de los puntos de inflexión que marcará una diferencia para la tecnología IoT, viene dado por la aparición de la pandemia de la covid-19 durante el último año [11]

La implantación del teletrabajo de manera prácticamente obligatoria fue una de las consecuencias que trajo consigo, y esto ha obligado a muchas empresas a hacer un mayor uso de tecnologías como Cloud Computing, Software de videoconferencia o acceso remoto a ordenadores, entre otras.

Si el teletrabajo ya era una herramienta en crecimiento antes de la pandemia, a partir de ahora se espera que su uso se implante siempre que sea posible, ya que ofrece mayor flexibilidad, menor pérdida de tiempo para los trabajadores y supone un ahorro económico para las empresas.

Debido a la pandemia, otras tecnologías relacionadas con IoT también se han visto impulsadas, como, por ejemplo, sensores para medir la calidad del aire, sistemas de seguimiento y control de servicios de mensajería para el buen funcionamiento del comercio online, utilización de Big Data en el sistema de salud pública, implantación de sistemas para poder hacer seguimiento de pacientes desde su casa.

Con todo esto, se espera que el mercado de IoT alcance un valor mayor del esperado en el año 2021, ya que esta tecnología ha demostrado ser una de las soluciones para muchos problemas y consecuencias de la pandemia.

3. Hardware

En este apartado, se va a realizar un estudio de las principales placas de desarrollo y sensores de temperatura y humedad que se pueden encontrar en el mercado, y se llevará a cabo una comparativa entre las diferentes posibilidades, para elegir la opción que mejor se adapte a las necesidades de este proyecto.

3.1. Placas de desarrollo

Una placa de desarrollo es un dispositivo hardware que contiene toda la electrónica necesaria para la realización de diferentes aplicaciones tecnológicas, incluyen elementos como, una fuente de alimentación, soporte para conectar sensores o actuadores y la unidad de procesamiento que se encarga de la lógica de la aplicación.

Es importante elegir la placa más adecuada en función de las necesidades requeridas por el diseño que se lleve a cabo.

Actualmente existen muchas placas de desarrollo, a continuación, se van a describir las que son más conocidas para el tipo de aplicación IoT que se está desarrollando en este caso.

Raspberry Pi

Raspberry pi es un dispositivo de bajo costo, desarrollado en Reino Unido por la Fundación Raspberry Pi. Su uso libre está permitido a nivel educativo y particular, aunque es un producto con propiedad registrada. En cuanto al software, es open source, permite emplear diferentes sistemas operativos, pero dispone de un sistema operativo oficial basado en Debian llamado Raspbian [12].

Algunos elementos electrónicos que están incluidos en todos los modelos son:

- Un procesador Broadcom.
- Memoria RAM.
- GPU.
- Puertos USB, HDMI, Ethernet (menos el primer modelo).
- 40 pines GPIO.
- Conector RCA (solo en los primeros modelos).

Debido a que ninguna versión incluye memoria, se puede emplear una tarjeta SD en su primera versión y una tarjeta MicroSD en las siguientes versiones.

Soporta diversos lenguajes de programación, como Python, Tiny Basic, C, Ruby y Perl.

Los diferentes modelos que se han lanzado son los siguientes:

- Raspberry Pi 1 modelo A
- Raspberry Pi 1 modelo B Y B+
- Raspberry Pi 2 modelo B
- Raspberry Pi 3 modelo B y B+
- Raspberry Pi 3 modelo A+
- Raspberry Pi 4 modelo B

También existe otra gama de placas llamada Raspberry Pi Zero, que son de menor tamaño y un poco menos potentes que las anteriores, pero esto es precisamente lo que les hace más interesantes para el desarrollo de productos IoT, ya que consumen menos y el precio es mucho menor. Estas placas son Raspberry Pi Zero, Raspberry Pi Zero W y Raspberry Pi WH, estas dos últimas incluyen WiFi y Bluetooth.

Se pueden encontrar a un precio desde 5 €, la Raspberry Pi Zero, hasta 30 € el nuevo modelo Raspberry Pi 4 modelo B, aunque es posible encontrarlas a un mayor precio.

Ilustración 7.- Raspberry Pi 4 modelo B

Arduino

Arduino es un proyecto de código abierto de gran popularidad internacional. Estas placas están diseñadas y fabricadas por la propia empresa, y con ellas es posible construir dispositivos inteligentes capaces de monitorizar y controlar objetos del mundo real [13].

En el mercado se pueden encontrar diferentes placas hardware Arduino que utilizan una variedad extensa de microcontroladores, aunque el más común es el microcontrolador Atmel AVR.

Es posible agregar placas de expansión o Shields empleando los puertos de entrada y salida de la placa base. Esto permite añadir circuitería, sensores, Ethernet o WiFi de forma sencilla mejorando la funcionalidad de Arduino.

La forma más común de alimentar la placa es por USB y puede programarse a través del puerto serie que incorpora.

El lenguaje de programación más empleado para programar un Arduino es C++, aunque también acepta otros lenguajes como Python, Ruby, PHP o Java.

Ofrece un entorno de desarrollo integrado (IDE) que facilita trabajar con la placa, permite programarla, descargar y añadir librerías y componentes de forma oficial, e incluso descargar frameworks para trabajar con placas de desarrollo diferentes a las del propio Arduino. Además, proporciona librerías que facilitan la programación del microcontrolador.

Los diferentes modelos de placas Arduino se diferencian entre sí, principalmente, en sus componentes y características, como por ejemplo el microcontrolador, el número de pines I/O digitales, entradas analógicas o la memoria flash.

Algunas de las placas más conocidas son:

- Arduino UNO
- Arduino Leonardo
- Arduino DUE
- Arduino ZERO

Actualmente existe una gama específica de placas orientadas a desarrollo de productos IoT.

El precio al que se comercializan las placas Arduino va desde aproximadamente 15 € en adelante.

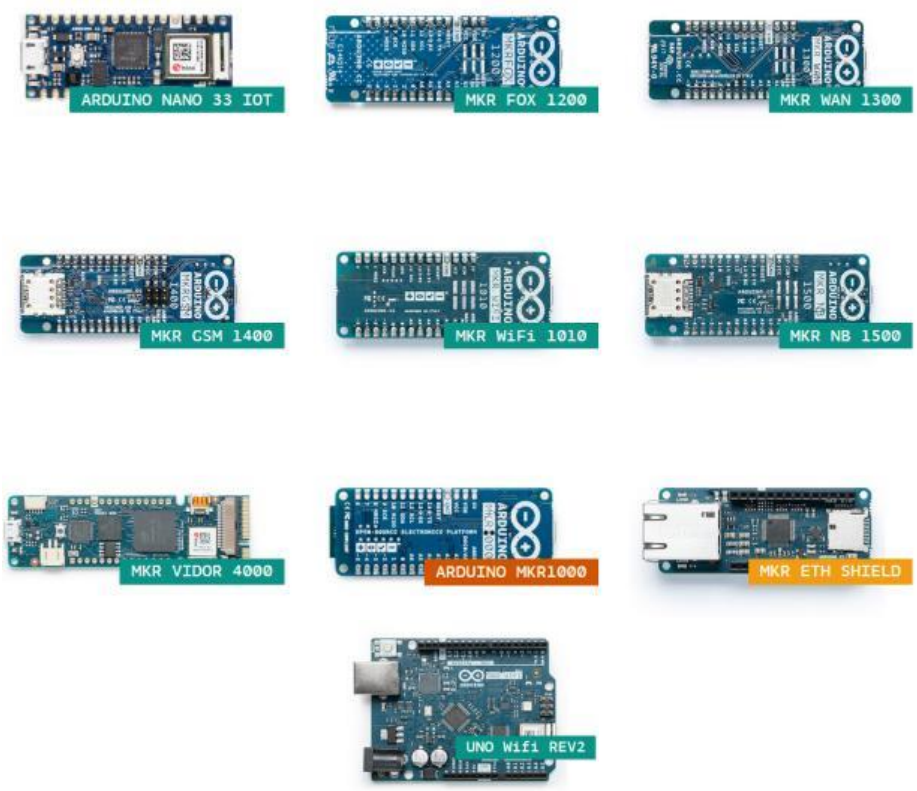


Ilustración 8.- Placas de desarrollo Arduino para IoT

NodeMCU

NodeMCU es una placa de desarrollo libre tanto a nivel hardware como software. Su función principal, al igual que la placa Arduino, es facilitar la programación de un microcontrolador.

Tiene incorporado el chip SoC ESP8266 de Espressif systems, que a su vez contiene un microcontrolador. Este chip permite la conexión WiFi, haciendo posible desarrollar proyectos de IoT con sistemas inalámbricos [14].

Existen diferentes versiones de NodeMCU, basadas en los módulos ESP-12 o ESP-12E, ambos basados en el SOC ESP8266.

Todas las versiones tienen en común las siguientes características:

- Posibilidad de alimentar y programar la placa a través del puerto USB, ya que contiene un convertidor serie-USB.
- Fácil acceso a los pines y posibilidad de alimentar sensores y componentes a través de estos.
- Led para indicar el estado.
- Botón de Reset.

Algunas diferencias entre las versiones son el número de pines al que se tiene acceso y el tamaño de la placa.

Actualmente existen 3 versiones que se pueden encontrar con una nomenclatura diferente, esto se debe a que, al ser una placa de hardware abierto, cualquiera puede fabricar un NodeMCU y comercializarlo. Es posible encontrarlas de la siguiente manera:

Generación	versión	Nombre común
1ª	V0.9	V1
2ª	V1.0	V2
3ª	V1.0	V3

Una de las ventajas más importantes es poder conectar la placa al ordenador a través de USB y tener la posibilidad de programarla desde el IDE Arduino.

Admite varios lenguajes de programación, desde el suyo propio llamado LUA, hasta micropython y C++, lo que le caracteriza de gran versatilidad.

Los principales fabricantes de esta placa son Amica, DOIT y LOLIN, aunque actualmente es posible encontrar muchos otros que ofrecen las mismas garantías.

Esta placa se puede encontrar a un precio menor de 4 €, pero dependerá del fabricante y la plataforma donde se realice la compra.

Con todas las ventajas que ofrece en cuanto a desarrollo de proyectos IoT y su bajo precio, hacen que sea una de las opciones más interesantes para la realización de este proyecto, por lo que será la que se elija para trabajar en el desarrollo del prototipo.

En el apartado 3.3 de este TFG se realizará una explicación en mayor profundidad sobre el funcionamiento y las partes que componen esta placa NodeMCU basada en el SOC ESP8266.

Ilustración 9.- NodeMCU V3 de AZ-Delivery

3.2. Sensores de temperatura y humedad

Un sensor es un dispositivo que transforma una magnitud física, denominada variable de instrumentación, por ejemplo, la temperatura o la distancia, en una variable eléctrica.

Para el diseño del medidor se necesita medir la temperatura, y además si se mide la humedad se obtendrá una predicción más correcta.

A continuación, se muestran algunos sensores de temperatura y humedad junto con las características técnicas más importantes de cada uno, para poder elegir el más adecuado según las necesidades del medidor.

Sensores analógicos:

Un sensor analógico se caracteriza porque su salida puede comprender un rango de valores instantáneos que varían en el tiempo, y que son proporcionales a la magnitud que se está midiendo.

TMP36

Calibrado en grados Celsius.
Voltaje de operación: de 2.7 V a 5.5 V
Rango de temperaturas: -40 ºC a 150 ºC
Precisión (25ºC): ± 2ºC
Conversión: 10 mV / ºC

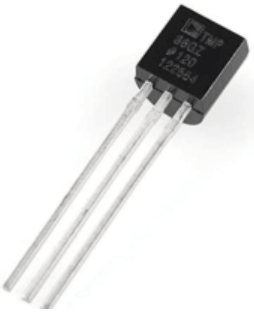


Ilustración 10.- Sensor TMP36

LM35

Calibrado en grados Celsius.
Voltaje de operación: de 2.7 V a 5.5 V
Rango de temperaturas: -55 ºC a 150 ºC
Precisión (25ºC): ± 0.5ºC
Conversión: 10 mV / ºC

Ilustración 11.- Sensor LM35

Sensores digitales:

Los sensores digitales son dispositivos que únicamente adoptan dos estados diferentes, Alto-Bajo (1-0). La ventaja de estos es que son menos sensibles al ruido.

TC74

Voltaje de operación: de 2.7 V a 5.5 V
Rango de temperaturas: -40 ºC a 125 ºC
Precisión: ± 2 ºC (de 25 ºC a 85 ºC) y ± 3 ºC (0 ºC a 125 ºC)
Resolución: 8 bit
Frecuencia de muestreo: 8 muestras/segundo.

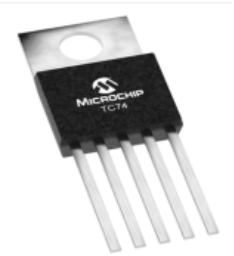


Ilustración 12.- Sensor TC74

DHT11

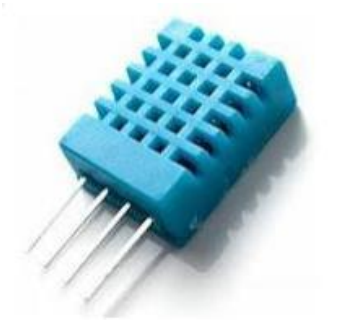


Ilustración 13.- Sensor DHT11

Permite medir temperatura y humedad.
Voltaje de operación: de 3 V a 5.5 V
Rango de temperaturas: 0 ºC a 50 ºC
Precisión de temperatura: ± 2 ºC
Rango de Humedad: 20 – 90 % RH
Precisión de humedad: ± 5 % RH
Resolución: 16 bit
Frecuencia de muestreo: 1 muestra/segundo.

Es de bajo costo.
Arduino incorpora una librería (DHT.h) que permite leer toda la gama de sensores DHT.
Como inconveniente, solo se puede leer la temperatura cada 1 o 2 segundos.

DHT22

Al igual que el anterior, permite medir tanto temperatura como humedad.
Voltaje de operación: de 3.3 V a 6 V
Rango de temperaturas: -40 ºC a 80 ºC
Precisión de temperatura: ± 0.5 ºC
Rango de humedad: 0 – 100 % RH
Precisión de humedad: ± (2 a 5) % RH
Resolución: 16 bit
Frecuencia de muestreo: 0.5 muestras/segundo.

Ilustración 14.- Sensor DHT22

También es un sensor de bajo costo, aunque el precio es un poco más elevado que el DHT11, pero tiene algunas ventajas sobre este, ya que es más preciso, fiable y estable.

Sigue siendo un sensor lento, ya que solo permite obtener una muestra cada dos segundos.

Este último sensor DHT22 será el que se utilizará para trabajar en el prototipo del medidor, ya que, observando las características de algunos de los sensores más conocidos, es la mejor opción teniendo en cuenta su precisión, fiabilidad y estabilidad. Además, cuenta con una librería propia para poder programarlo desde el IDE Arduino, lo que ofrece mayor versatilidad y simplicidad a la hora de manejarlo.

En este caso, se va a diseñar un medidor de temperatura y humedad para el hogar, por lo que los rangos tanto de temperatura (-40 ºC a 80 ºC) como de humedad (0 – 100 % RH) del sensor son válidos.

En cuanto a la frecuencia de muestreo (0.5 Hz) es suficiente, ya que la temperatura en el ambiente no se ve modificada en un margen de tiempo tan pequeño.

Incorpora electrónica de acondicionamiento y es digital, lo cual reduce la sensibilidad al ruido. La conexión se realiza de forma sencilla a partir de 3 pines (VDD, DATA, GND), y, además, es posible encontrarlo a un precio asequible.

En el apartado 3.3 de este TFG se explicará en mayor profundidad algunas de sus características.

3.3. Placa de desarrollo NodeMCU y sensor DHT22

Como se explicó anteriormente, se ha decidido trabajar con la placa de desarrollo NodeMCU V3 y el sensor de temperatura y humedad DHT22, ya que las características técnicas que ambos ofrecen se adecúan a las necesidades del prototipo que se va a diseñar.

En este apartado, se va a explicar en mayor profundidad algunas particularidades y características de ambos componentes, con el objetivo de entender mejor el funcionamiento de los elementos que formarán parte del prototipo y así, poder trabajar con ellos de una manera adecuada.

NodeMCU

Esta placa de desarrollo, como otras, está formada por diferentes módulos que incluyen todos los elementos necesarios para realizar diferentes trabajos o proyectos técnicos de la manera más sencilla posible [14].

El esquema de esta placa concretamente es el que se muestra en la siguiente imagen:

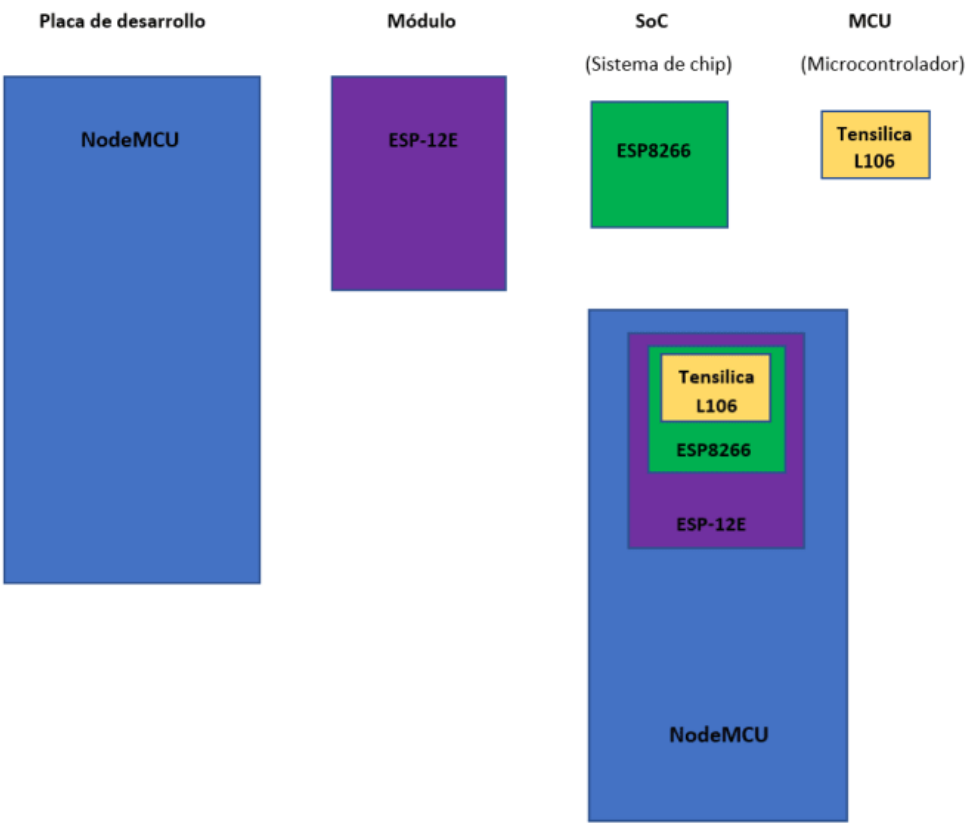


Ilustración 15.- Esquema de las partes que constituyen el NodeMCU

MCU o Microcontrolador:

El microcontrolador es la unidad más básica y en el NodeMCU está integrado en el SoC ESP8266. El MCU se denomina Tensilica L106 de 32-bit, y su función es gestionar todas las entradas, salidas y cálculos necesarios para que el programa que se haya cargado en él funcione correctamente. Este MCU logra un consumo de energía muy bajo y trabaja a una velocidad máxima de 160 MHz.

Incluye las siguientes interfaces [15] :

- Interfaces RAM/ROM programables (iBus), que pueden conectarse con el controlador de memoria y se pueden utilizar para visitar la memoria flash.
- Interfaz RAM (dBus) de datos, que puede realizar la conexión con el controlador de memoria.
- Interfaz AHB que se puede utilizar para visitar el registro.

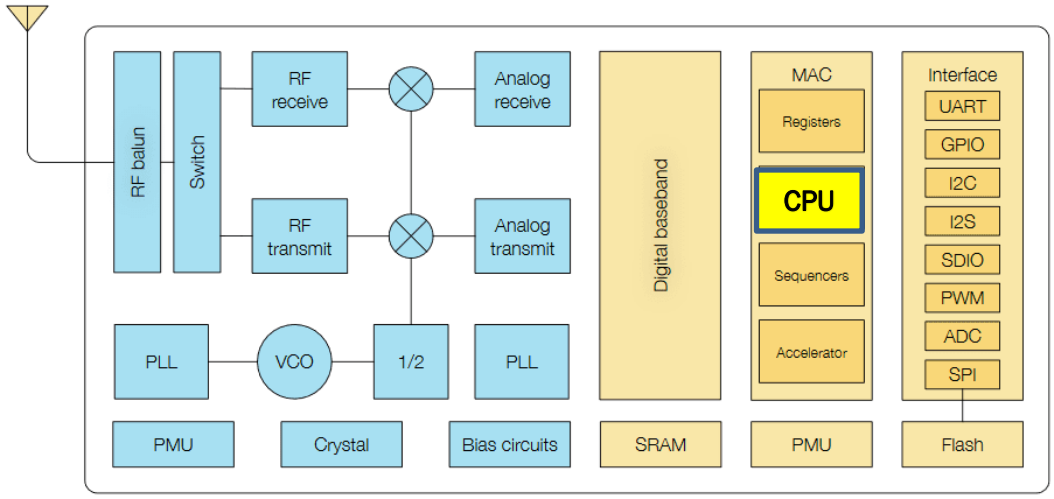


Ilustración 16.- Posición del Microcontrolador (CPU) dentro del SoC ESP8266 [15]

SoC ESP8266:

El nombre técnico es ESP8266EX y consiste en un chip que tiene prácticamente todos los elementos integrados para funcionar de forma autónoma, como si se tratase de un ordenador. El único componente que no tiene es una memoria Flash para el almacenamiento de programas, lo que puede suponer un problema, ya que algunos pines de entrada y salida deberán utilizarse para conectarse a una memoria Flash externa.

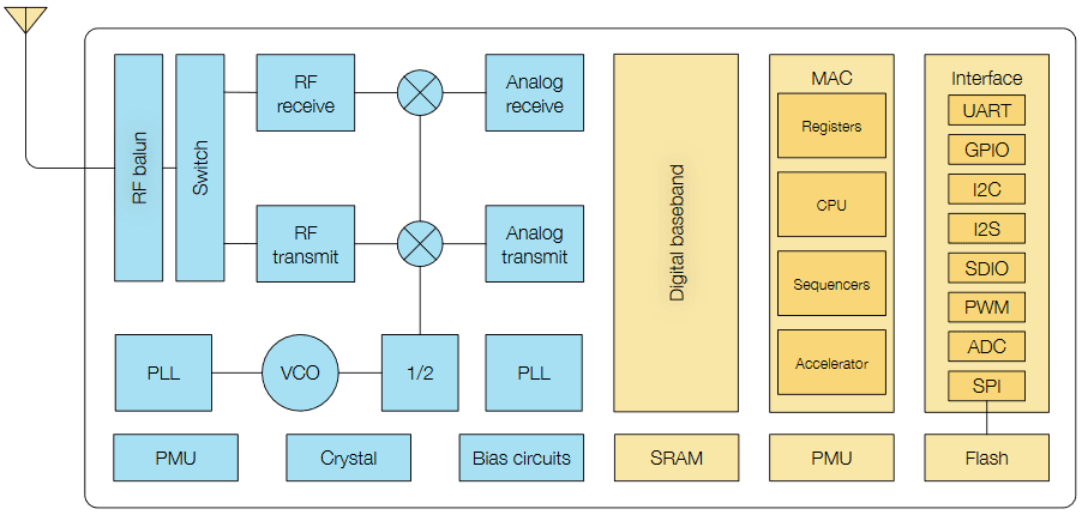


Ilustración 17.- Diagrama de bloques del SoC ESP8266 [15]

Algunas de las características del ESP8266 son las siguientes:

- Incorpora el Microcontrolador Tensilica L106 de 32-bit.
- Memoria RAM de aproximadamente 50 kB.
- No tiene memoria ROM programable, el programa de usuario debe almacenarse en una memoria Flash externa.
- Admite memoria Flash SPI externa para almacenar programas de usuario, admite hasta 16 MB de capacidad de memoria.
- Incorpora módulo WiFi de 2.4 GHz.
- Tiene 17 pines de entrada y salida GPIO (de propósito general).
- 1 entrada analógica de 10-bit (ADC).

Módulo ESP12-E:

En el siguiente nivel se encuentra el módulo ESP-12 y el ESP-12E, dependiendo de la versión del NodeMCU.

Este es el módulo que incorpora la memoria flash para almacenar los programas de usuario, y, además, los pines del ESP8266 están cableados hasta los pines de este módulo internamente, para facilitar su acceso.

La diferencia principal entre este tipo de módulos es el acceso a los pines, dependiendo del módulo con el que se trabaje, se tendrá acceso a unos pines u a otros.

Ilustración 18.- Módulo ESP12-E

NodeMCU:

El último nivel sería la propia placa NodeMCU, que incorpora los componentes necesarios para programar y conectar el módulo y el Microcontrolador.

Existen diferentes versiones de NodeMCU que tienen en común algunas características que ya se nombraron en el apartado 3.1 y difieren en algunas otras.

Versiones del NodeMCU:

Todas las versiones del NodeMCU se basan en los mismos módulos ESP-12 y ESP-12E, que incorporan el SoC ESP8266. Por lo tanto, las diferencias principalmente se encuentran en el número de pines a los que se puede acceder y el tamaño de la placa.

A continuación, se va a explicar brevemente las características principales de cada versión, y se realizará una comparativa entre ellas. Actualmente se pueden encontrar 3 versiones diferentes del NodeMCU en el mercado.

1ª generación / v0.9/ V1

Incorpora el módulo ESP-12 con una memoria Flash de 4MB. Uno de los inconvenientes de esta versión es su tamaño, ya que es mayor que las otras dos versiones.

Es una versión obsoleta que resulta complicado encontrar en el mercado actualmente.

Ilustración 19.- NodeMCU V1

2ª generación / v1.0 / V2

Esta versión se diferencia de la anterior en que su tamaño es notablemente menor, y utiliza el módulo actualizado ESP-12E, lo que permite el acceso a un mayor número de pines. Está diseñada para ofrecer mayor calidad y facilidad en la programación y prototipado, y es posible encontrarla en el mercado, ya que no es tan antigua como la primera versión.

Ilustración 20.- NodeMCU V2

2ª generación / v1.0 / V3

Esta versión aporta ciertas mejoras respecto a las versiones anteriores, algunas son las siguientes:

- Incluye un conversor serial CH340G en lugar del CP2102 de la versión V2, lo que hace que el puerto USB sea más robusto.
- 2 pines que no se utilizan en la versión anterior, se han programado como salida de 5V directa del USB y un GND adicional.
- El tamaño es ligeramente mayor que la V2.

Ilustración 21.- NodeMCU V3.

Es la versión más fácil de encontrar en el mercado, ya que es la más actualizada, y junto con todas las mejoras que ofrece respecto a las anteriores, resulta la más interesante para su utilización en el prototipo que se va a diseñar, por lo que será la que se elija para ello.

Descripción de las características más importantes [16]:

- Programación a través de un cable micro USB-B.
- Permite dos formas de alimentación:
    - Micro USB-B en el puerto USB del ordenador.
    - Micro USB-B en el adaptador de corriente USB de 5V.
- 11 pines de entrada/salidas digitales a 3.3V.
- 1 pin de entrada/salida analógica.
- Modulo ESP-12E procesador con ESP8266 Módulo-WLAN.
- CH340 Interfaz USB.
- Programable a través de Arduino Code y Lua.

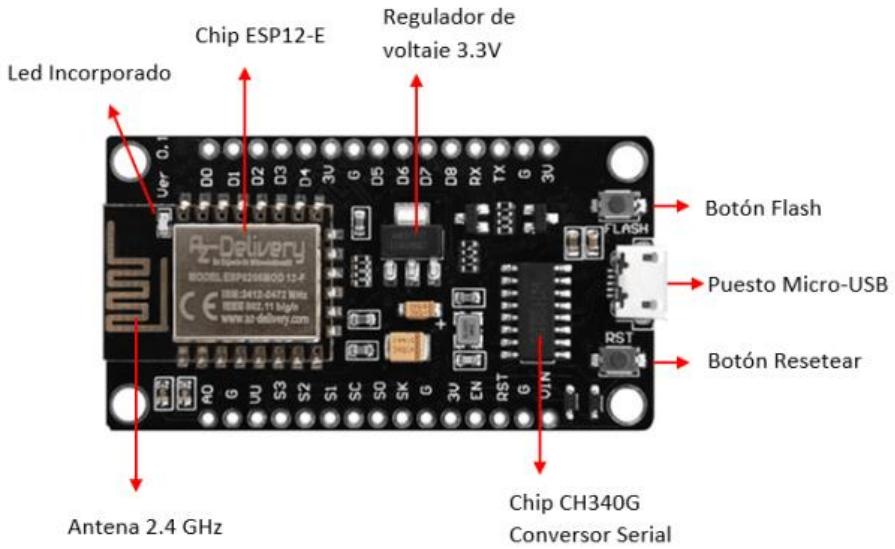


Ilustración 22.- Partes del NodeMCU V3

Esquema pines del NodeMCU V3:

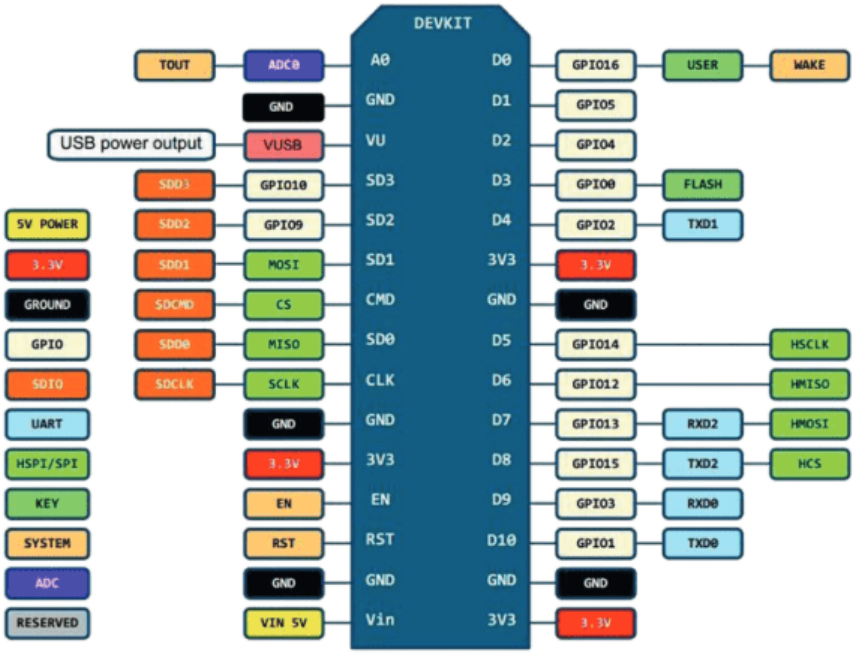


Ilustración 23.- Diagrama de pines del NodeMCU V3 [16]

Entradas y salidas digitales:

El NodeMCU contiene trece pines digitales que se numeran desde el D0 al D12. La nomenclatura que aparece como GPIOxx corresponde al nombre que se les asigna a los pines que conectan con el módulo ESP8266.

Debido a que el SoC ESP8266 no dispone de memoria Flash, es importante dejar algunos pines libres, por lo que no es recomendable el uso de los pines desde GPIO06 al GPIO11. Solamente dos de estos son accesibles a través de la placa, GPIO9 (D11) y GPIO10 (D12), aunque en la placa aparecen con el nombre SD2 y SD3 respectivamente. Se recomienda, siempre que sea posible, no utilizarlos para conectar sensores u otros componentes.

Los pines GPIO3 (D9) y GPIO1 (D10) corresponden con los pines Rx y Tx, los cuales se emplean para recibir y transmitir programas o para comunicar la placa con el ordenador a través del puerto serie, por lo que es importante evitar su uso.

Teniendo en cuenta lo anteriormente explicado, quedarían nueve pines digitales de entrada y salida disponibles para realizar las conexiones, desde el D0 al D8.

Estos pines de E/S digitales están diseñados para trabajar con valores binarios, por lo que solo detectan niveles alto y bajo (1 y 0), por ello, los sensores y actuadores que se conecten a estos deben ser de tipo digital.

Pin Analógico A0:

A diferencia de los pines digitales, que solamente detectan dos estados, Alto o Bajo, el pin analógico tiene un rango de valores que viene determinado por la resolución del convertidor ADC (Convertidor Analógico Digital).

El pin admite un rango de valores de 0 a 3.3 V con una resolución de 10 bits, es decir, puede tomar 2^{10} (1024) valores diferentes. En este caso, podrá distinguir valores entre 0 y 1023.

Alimentación:

El voltaje de operación del NodeMCU es de 3.3 V y se puede alimentar de dos maneras diferentes:

- A través de una fuente de alimentación externa de 5 V conectada al pin Vin.
- Conectando el NodeMCU con un cable USB al propio ordenador, ya que la placa cuenta con un conversor USB a serie.

Cuando la alimentación se lleva a cabo mediante el puerto USB con 5V, el NodeMCU tiene un regulador de tensión interno que permite sacar 3.3 V y 5 V.

Los 3.3 V se utilizan para alimentar la propia placa y para sacarlos por los 3 pines que aparecen marcados en el NodeMCU con ese valor de tensión (3V3), lo que permite alimentar componentes externos a esa tensión.

Los 5 V, que salen del pin VU, se utilizan para alimentar componentes externos que trabajen con esta tensión.

Leds y botones:

NodeMCU tiene un led incorporado conectado al pin D4 que pertenece al módulo ESP-12E. Este led funciona como indicador, ya que parpadea mientras se está cargando un programa en la placa. Además, se puede programar y utilizar.

El botón de reset se utiliza para resetear, no permite eliminar el código, si no que reinicia la ejecución del programa desde el principio.

El botón flash está conectado con el pin GPIO0, correspondiente con el pin D3, y se encarga de poner un pin en un estado conocido.

DHT22

DHT22 es un sensor de temperatura y humedad, de tipo digital, con una precisión bastante alta [17].

Contiene un procesador interno que se encarga de realizar el proceso de medición, proporcionando la medición con una señal digital.

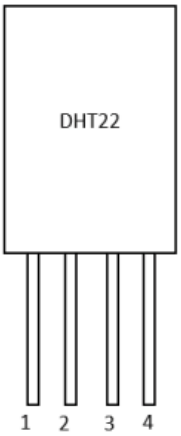
Las características técnicas del sensor son las siguientes:

MAGNITUD	VALORES
Voltaje de operación	3.3-6V DC
Señal de salida	Señal digital a través de single-bus
Elemento sensor	Condensador de polímero
Rango de operación	Humedad: 0-100%RH; Temperatura: -40~80 ºC
Precisión	Humedad: +-2%RH (Max +-5%RH); Temperatura <+-0.5 ºC
Resolución o sensibilidad	Humedad: 0.1%RH; Temperatura: 0.1 ºC
Repetibilidad	Humedad +-1%RH; Temperatura: +-0.2 ºC
Histéresis de humedad	+-0.3%RH
Estabilidad a largo plazo	+-0.5%RH/año
Periodo de detección	Promedio: 2s
Intercambiabilidad	Totalmente intercambiable
Dimensiones	Tamaño pequeño: 14*18*5.5mm Tamaño grande: 22*28*5mm

Pines del sensor:

Dispone de 4 patillas, de las cuales se emplean 3, VDD, DATA y GND. Se puede encontrar suelto, en este caso para realizar la conexión con una placa es necesario añadir una resistencia pull-up, o se puede encontrar en un montaje sobre una placa PCB en la que ya estaría incluida dicha resistencia, y sin el pin 3.

Las conexiones del sensor con el NodeMCU se explicarán en el apartado 7.2 de este TFG en el que se explica toda la conexión harwdware del prototipo.



Pin	Función
1	VDD (Alimentación)
2	DATA (Señal)
3	NULL
4	GND

Ilustración 24.- Pines del sensor DHT22

Las características eléctricas de este sensor son las siguientes:

	Condición	Min	Típico	Max	Unidades
Voltaje de alimentación	DC	3.3	5	6	V
Corriente de alimentación	Medición	1		1.5	mA
	Reposo	40	Null	50	µA
Periodo de recolección	Segundos		2		Segundos

4. Plataformas IoT

Una plataforma IoT es el software que permite conectar toto el sistema IoT, permite monitorizar, almacenar, supervisar y procesar los datos desde la nube.

Estas plataformas ofrecen interfaces estándares al usuario, pero también herramientas para crear interfaces más complejas.

Recogen los datos enviados desde los dispositivos IoT conectados y permiten analizar los datos en tiempo real.

Estas plataformas hacen posible la conectividad con aplicaciones móviles y otros dispositivos, para poder visualizar los datos recibidos desde el dispositivo IoT que está conectado a la plataforma, haciendo posible actuar sobre él desde la propia aplicación móvil.

4.1. Clasificación de plataformas

Algunas plataformas IoT más conocidas que se pueden encontrar en el mercado son las siguientes:

4.1.1. Amazon Web Services IoT

Esta plataforma permite conectar varios dispositivos, crear interacciones entre ellos, procesar los datos recibidos y, además, crear aplicaciones para interactuar con los dispositivos [18].

Ilustración 25.- Logo de Amazon Web Services

Los datos recibidos se pueden filtrar y transformar, ya que esta plataforma facilita la integración con otros servicios de la misma, como Amazon S3, Amazon Machine Learning y AWS Lambda, entre otros.

Para asegurar la privacidad y la veracidad de los datos ofrece una capa de seguridad con mecanismos de autenticación y cifrado integral en todos los puntos de conexión.

Permite conectar los dispositivos a través de los protocolos de comunicación HTTPS, WebSockets o MQTT.

Los datos recibidos en la plataforma se procesan de forma continua y es posible procesarlos, almacenarlos o enviarlos a servicios ajenos a AWS haciendo uso de la herramienta Lambda.

Una de los servicios de AWS que se emplea para desarrollar proyectos IoT es AWS IoT core, que permite conectar dispositivos IoT a la nube de AWS sin la necesidad de administrar servidores. Este servicio admite miles de millones de dispositivos y billones de mensajes, puede procesar y enviar los datos a puntos de enlace de AWS y a otros dispositivos de una manera segura.

La forma de cobro de AWS IoT core depende de la región y de diferentes factores, como se va a ver a continuación:

Conectividad

El precio, en este caso, depende del tiempo total de conexión de los dispositivos a AWS IoT Core.

Mensajería

El cobro se realiza teniendo en cuenta la cantidad de mensajes transmitidos entre los dispositivos y AWS IoT core.

Registro y sombra de dispositivos

El registro se emplea para nombrar y administrar dispositivos, y la sombra de un dispositivo almacena el estado deseado o real de estos. Por lo que el precio en este caso, depende de la cantidad de operaciones que acceden a los datos registrados o la sombra y los modifican.

Motor de reglas

Esta herramienta permite transformar datos de dispositivos a través de operaciones aritméticas o funciones externas como Lambda, y direccionar los datos a un servicio de AWS como Amazon S3, Amazon DynamoDB o Amazon Kinesis. El precio depende del número de veces que se activa una regla y de la cantidad de acciones que se ejecutan dentro de una regla, teniendo en cuenta un mínimo de acción por regla.

Tabla resumen de precios AWS IoT:

Región	Europa (París)
Conectividad	0,08 € (Por millón de minutos de conexión)
Mensajería	0,07 € (Por millón de mensajes) *A partir de 5 mil millones de mensajes*
Registro y sombra de dispositivos	1,24 € (Por millón de operaciones)
Motor de regles	Reglas activadas → 0,15 € (Por millón de reglas activadas o de acciones ejecutadas) Acciones ejecutadas → 0,15 € (Por millón de reglas activadas o de acciones ejecutadas)

AWS ofrece una capa gratuita de una duración de 12 meses en la que se incluyen algunos servicios de la plataforma como Amazon EC2, Amazon S3, Amazon RDS, Amazon Dynamo DB, AWS Lambda e incluso AWS IoT core aunque con algunas restricciones.

La capa gratuita del servicio AWS IoT core incluye lo siguiente:

- 2.250.000 minutos de conexión.
- 500.000 mensajes.
- 225.000 operaciones de registro o sombra de dispositivos.
- 250.000 reglas disparadas y 250.000 acciones ejecutadas.

4.1.2. Azure IoT Hub

Ilustración 26.- Arquitectura de referencia de Azure IoT [19]

Azure es una plataforma que permite comunicación bidireccional, fiable y segura entre los dispositivos IoT conectados y las diferentes soluciones en la nube, algunas de las soluciones que incluye son las siguientes [19]:

- IoT Hub, es el centro de comunicaciones de los dispositivos.
- Stream Analytics, herramienta para analizar y procesar los datos enviados desde los dispositivos.
- Event Hub, para lanzar eventos y llevar a cabo las consecuentes acciones.
- Web Apps, encargadas de la parte visual de los datos.
- Bases de datos, en las que se puede almacenar los datos procesados.

Azure IoT Hub ofrece ciertas características para la comunicación de los dispositivos con la plataforma:

- Tiene un canal de comunicaciones seguro para el envío de los mensajes desde los dispositivos, para ello emplea la autenticación por dispositivo.
- Integra una funcionalidad para redireccionar los mensajes a través del enrutamiento de los mismos, con esto es posible controlar donde se envían los datos de telemetría de los dispositivos.

- IoT Hub se puede integrar con otros servicios de la plataforma para obtener soluciones más amplias, como puede ser con Azure Event Grid, Azure Logic Apps, Azure Machine Learning o Azure Stream Analytics, entre otros.
- Permite configurar y controlar los dispositivos
- Los lenguajes compatibles son C, C insertado, C#, Java, Python y Node.js.
- Los protocolos de comunicación que admite son HTTPS, AMQP, AMQP sobre WebSockets, MQTT, MQTT sobre WebSockets.

El precio asociado a la cuenta creada en Azure IoT depende de la región y dos factores principales, el número de mensajes totales por día y el tamaño medio del medidor de mensajes, ya que los mensajes se fragmentan en bloques del tamaño que se indica en la tabla.

Nivel Basic

TIPO DE EDICIÓN	PRECIO POR UNIDAD DE IOT HUB (AL MES)	NÚMERO TOTAL DE MENSAJES AL DÍA POR UNIDAD DE IOT HUB	TAMAÑO DEL MEDIDOR DE MENSAJES
B1	€8,433	400.000	4 KB
B2	€42,165	6.000.000	4 KB
B3	€421,650	300.000.000	4 KB

Nivel Standard

TIPO DE EDICIÓN	PRECIO POR UNIDAD DE IOT HUB (AL MES)	NÚMERO TOTAL DE MENSAJES AL DÍA POR UNIDAD DE IOT HUB	TAMAÑO DEL MEDIDOR DE MENSAJES
Gratis	Gratis	8.000	0,5 KB
S1	€21,083	400.000	4 KB
S2	€210,825	6.000.000	4 KB
S3	€2108,25	300.000.000	4 KB

Ilustración 27.- Precios Azure IoT

Además, si se utiliza otro servicio o solución de la nube de Azure, se factura a parte, por ejemplo, si se desea procesar los datos hay que recurrir a la herramienta Stream Analytics, la cual utiliza una serie de recursos informáticos denominados Unidades de Streaming (SU), que tienen un precio dependiendo del tiempo de uso.

Unidad de streaming estándar

	ESTÁNDAR	DEDICATED
Resource Type	Stream Analytics Job	Stream Analytics Cluster
Unidades de streaming	€0,093/hora with a 1 SU minimum	€0,093/hora with a 36 SU minimum*
Virtual Network support	No	Yes
C# User-defined functions	Limited to West Central US, North Europe, East US, West US, East US 2 and West Europe	All regions
Custom deserializers	Limited to West Central US, North Europe, East US, West US, East US 2 and West Europe	All regions

Ilustración 28.- Precios por Unidades de Streaming en Azure IoT

4.1.3. Google Cloud IoT Core

Google Cloud IoT Core es un servicio incluido en la plataforma Google Cloud con el fin de poder conectar y administrar de forma segura dispositivos IoT [20].

Es un servicio gestionado para conectar, administrar y adquirir datos de millones de dispositivos de forma segura y sencilla.

Google Cloud permite combinar este servicio con otros de la propia plataforma para obtener una solución más completa, pudiendo recopilar, procesar, analizar y visualizar los datos enviados por los dispositivos en tiempo real, lo que otorga una alta eficiencia operativa en el desarrollo de proyectos IoT.

Permite establecer comunicación bidireccional con los dispositivos, y además emplea un sistema de seguridad integral basado en la autenticación de los dispositivos mediante claves asimétricas y certificados firmados por una autoridad de certificación.

Los componentes principales de Google Cloud IoT Core son los siguientes:

- El administrador de dispositivos, que permite registrar dispositivos en el servicio, para poder monitorizarlos y configurarlos posteriormente de forma segura.
- Dos puentes de protocolo con MQTT y HTTP para que los dispositivos puedan conectarse a la plataforma haciendo uso de ellos.

Los datos de telemetría enviados desde los dispositivos se publican en Cloud Pub/Sub, lo que permite utilizar otros servicios de análisis.

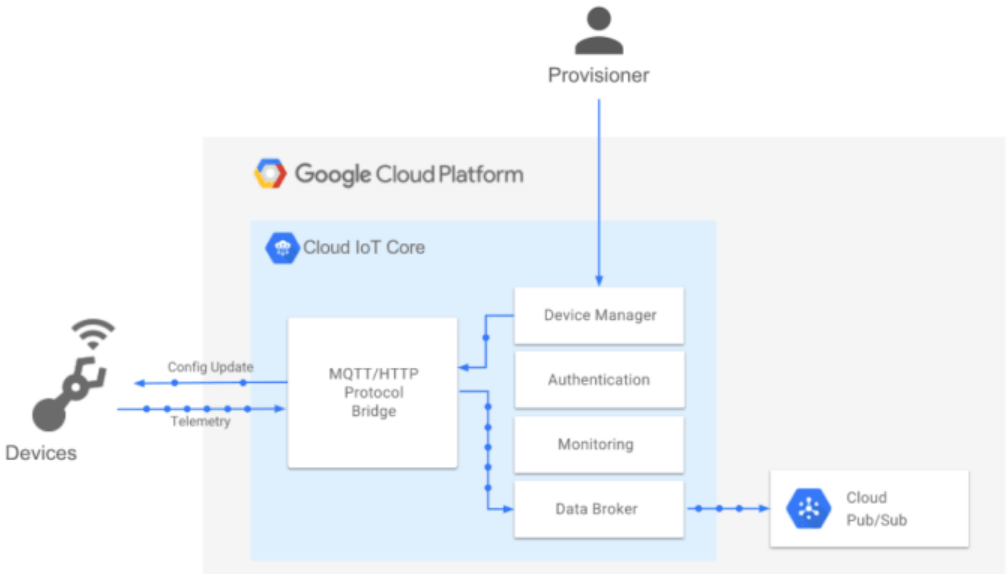


Ilustración 29.- Componentes del servicio y flujo de datos en Google Cloud [20]

El precio de Cloud IoT Core se establece en función del volumen de datos que se emplee por mes:

Volumen de datos mensual	Precio por MB	Dispositivos registrados	Cargo mínimo*
Hasta 250 MB	0,00 $	Ilimitados, dentro de las cantidades máximas de consultas por segundo	1024 bytes
De 250 MB a 250 GB	0,0045 $	Ilimitados, dentro de las cantidades máximas de consultas por segundo	1024 bytes
De 250 GB a 5 TB	0,0020 $	Ilimitados, dentro de las cantidades máximas de consultas por segundo	1024 bytes
5 TB y más	0,00045 $	Ilimitados, dentro de las cantidades máximas de consultas por segundo	1024 bytes

Ilustración 30.- Precios Google Cloud IoT Core

En el caso de utilizar Cloud IoT Core junto con Cloud Pub/Sub, se facturan por separado los recursos que se consuman al emplear este otro servicio, esto ocurre también en caso de utilizar cualquier otro servicio de la plataforma.

4.1.4. Thinger.io

Thinger.io es una plataforma IoT en la nube, de código abierto, que proporciona todas las herramientas necesarias para desarrollar prototipos, y administrar productos conectados de manera simple [21].

La plataforma está compuesta por dos productos principales, un Backend que es el servidor IoT real, y un Frontend basado en la web, que simplifica el trabajo con las funciones usando cualquier ordenador o teléfono inteligente.

En la siguiente imagen se muestran las características principales que proporciona esta plataforma para el desarrollo de proyectos IoT.

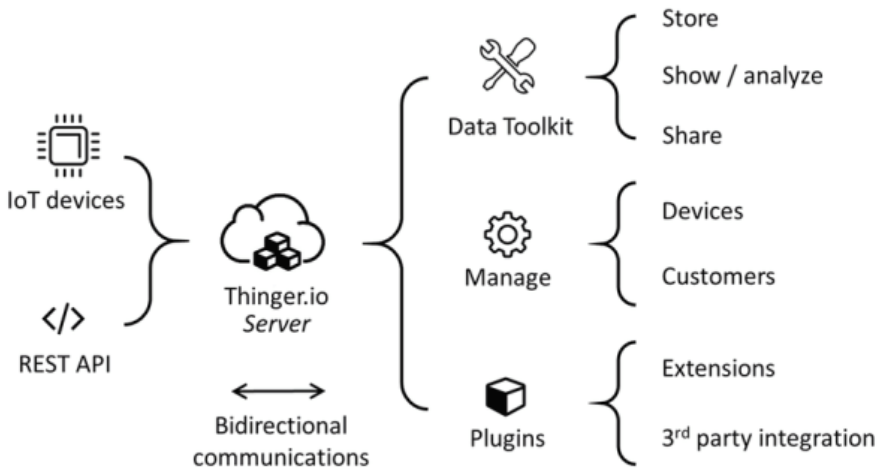


Ilustración 31.- Características de Thinger.io para IoT [21]

La plataforma es compatible con prácticamente cualquier tipo de dispositivo y permite crear comunicación bidireccional con dispositivos Linux, Arduino, Raspberry Pi o MQTT.

Permite agregar los datos del dispositivo en tiempo real y almacenarlos creando depósitos de datos. Estos datos pueden ser mostrados utilizando widgets, tanto en formato de gráfico de tiempo, gráfico de anillo, medidores e incluso es posible diseñar representaciones personalizadas.

Tiene la posibilidad de activar eventos empleando un motor de reglas Node-RED integrado. Además, permite personalizar la apariencia de la interfaz con los colores, logotipos y dominio web que se desee.

Se pueden crear diferentes tipos de cuentas a distintos precios dependiendo de la cantidad de herramientas y servicios de los que se quiera hacer uso.

Desde una cuenta gratuita destinada a estudiantes o desarrolladores aficionados de proyectos IoT, hasta empresas con todos los servicios incluidos de manera ilimitada por 599 €/mes.

La cuenta gratuita incluye los siguientes servicios:

- 2 dispositivos.
- Un único desarrollador.
- Nube comunitaria.
- Características básicas.
    - Gestión de dispositivos.
    - Dashboards.
    - Depósitos de datos.
    - Enpoints.
    - Acceso a tokens.
- Soporte comunitario.

El resumen de precios dependiendo de la cuenta es el siguiente:

Free	Small	Medium	Large	Unlimited
0 €/mes	29 €/mes	149 €/mes	299 €/mes	599 €/mes

4.1.5. Thingspeak

ThingSpeak es un servicio de plataforma de análisis de IoT de MathWorks, los creadores de Matlab y Simulink [22].

Permite agregar, analizar y visualizar datos en tiempo real en la nube desde dispotivos IoT conectados.

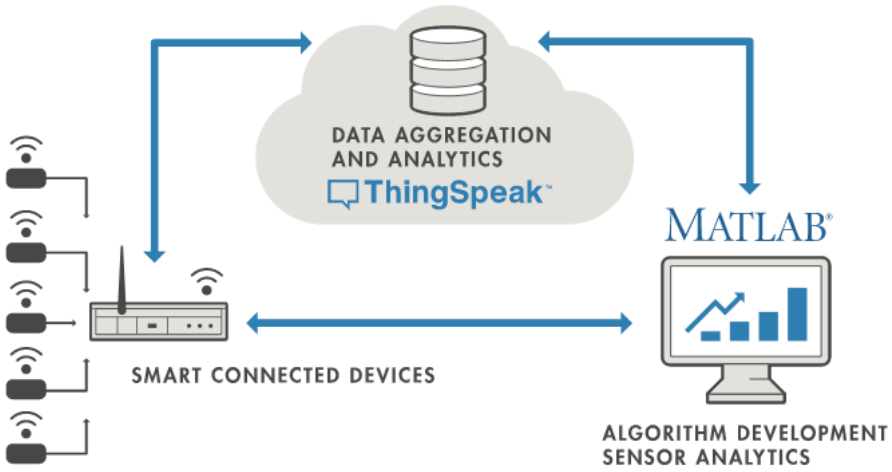


Ilustración 32.- Diagrama de funcionamiento de ThingSpeak

Es compatible con dispositivos como Arduino, Raspberry Pi y el Módulo WiFi ESP8266, entre otros.

Se puede ejecutar código Matlab en ThingSpeak para realizar análisis y procesamiento de datos en tiempo real, y es posible crear sistemas de IoT sin configurar servidores ni software web.

Los datos que se envían desde el dispositivo a la plataforma se almacenen en canales, cada canal almacena hasta 8 campos de datos, y se pueden crear tantos canales como sea necesario. Cada canal está formado por los siguientes campos:

- 8 campos para almacenamiento de datos, por ejemplo, datos enviados por un sensor.
- 3 campos para el almacenamiento de información, para guardar latitud, longitud y elevación.
- Un campo “estado”, para describir la información almacenada en el canal.

Desde la plataforma no se gestionan los dispositivos, ya que el envío de datos se basa en el concepto de canal, como se ha explicado.

Otra característica de la plataforma es que permite acceder a los datos tanto en línea como fuera de línea, ya que ThingSpeak almacena toda la información que se envía en una ubicación central en la nube. Además, los datos privados están protegidos con una clave API que el propio usuario puede controlar, por lo que es posible descargar de forma segura los datos almacenados en la nube.

Es compatible con protocolos como HTTP y MQTT.

Con Matlab integrado en la plataforma, es posible realizar calibraciones, análisis, transformación de datos e incluso crear gráficos personalizados para visualizar dichos datos, aunque por defecto ofrece ya diferentes posibilidades para ello. También cuenta con una serie de indicadores de alarma o displays numéricos.

La plataforma puede configurarse para desencadenar eventos que envíen datos de ThingSpeak a otras aplicaciones, como correos electrónicos o tweets.

Ilustración 33.- Programación de alertas en ThingSpeak

ThingSpeak ofrece los siguientes planes de licencias según las necesidades de cada usuario:

	FREE For small non-commercial projects
Scalable for larger projects	✖ No. Annual usage is capped.
Number of messages	3 million/year (~8200/day)[2]
Message update interval limit	Every 15 seconds
Number of channels	4
MATLAB Compute Timeout	20 seconds
Number of simultaneous MQTT subscriptions	Limited to 3
Private channel sharing	Limited to 3 shares
Technical Support	Community Support

Ilustración 34.- Plan gratuito de ThingSpeak

	STUDENT For students at degree-granting institutions[1]	HOME For personal use only[1]
Scalable for larger projects	✔	✔
Number of messages	33 million/year per unit (~90.000/day per unit)[2]	33 million/year per unit (~90.000/day per unit)[2]
Message update interval limit	Every second	Every second
Number of channels	10 per unit	10 per unit
MATLAB Compute Timeout	20 seconds	20 seconds
Number of simultaneous MQTT subscriptions	50 per unit	50 per unit
Private channel sharing	Unlimited	Unlimited
Technical Support	Community Support	Community Support

Ilustración 35.- Plan para estudiantes y uso personal de ThingPeak

	ACADEMIC For academic use by faculty, staff, or researchers at degree-granting institutions[1]	STANDARD For all commercial, government and revenue generating activities
Scalable for larger projects	✔	✔
Number of messages	33 million/year per unit (~90.000/day per unit)[2]	33 million/year per unit (~90.000/day per unit)[2]
Message update interval limit	Every second	Every second
Number of channels	250 per unit	250 per unit
MATLAB Compute Timeout	60 seconds	60 seconds
Number of simultaneous MQTT subscriptions	50 per unit	50 per unit
Private channel sharing	Unlimited	Unlimited
Technical Support	Standard MathWorks support	Standard MathWorks support

Ilustración 36.- Plan académico y standard de ThingSpeak

Existe una calculadora de precios que permite obtener el precio de cada licencia dependiendo del número de canales y la frecuencia de recopilación de datos, en este caso se ha obtenido el precio de cada licencia con los valores por defecto que ofrece la calculadora:

- Número de canales = 10.
- Frecuencia de recolección de datos = 2 cada minuto.

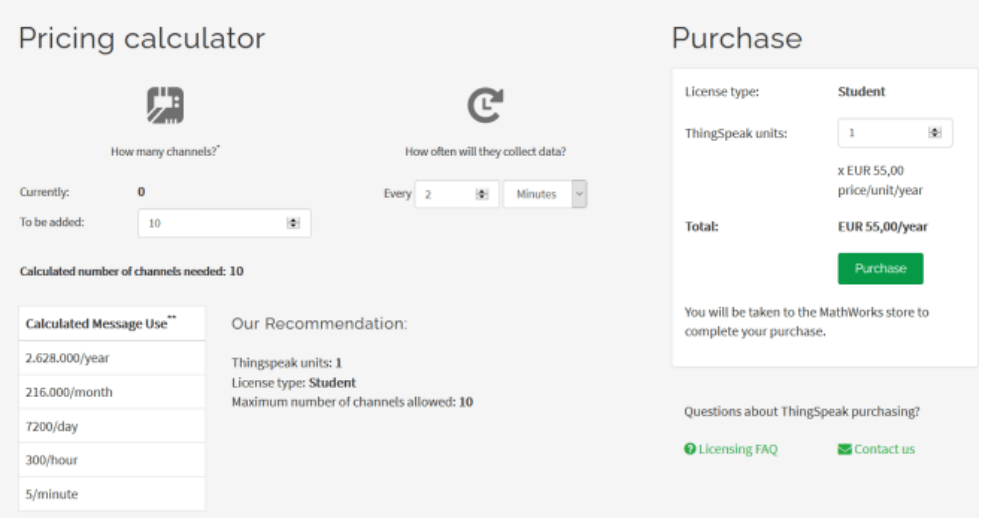


Ilustración 37.- Calculadora de precios de ThingSpeak

Free	Student	Home	Academic	Standard
0 €/año	55 €/año	75 €/año	250 €/año	600 €/año

4.2. Comparativa y elección

En este apartado se va a llevar a cabo una comparativa de las plataformas que se han descrito anteriormente, en su versión gratuita, ya que es la que se va a emplear. Para realizar dicha comparación se tendrán en cuenta algunas de sus características más importantes y las ventajas e inconvenientes que puede tener cada una, con el objetivo de elegir posteriormente la plataforma que mayores posibilidades ofrezca a la hora de diseñar el medidor IoT.

Tabla de características principales de cada plataforma:

Plataformas	Lenguajes	Protocolos de comunicación	Hardware compatible
AWS IoT	C, JavaScript, Java, Python, iOS, Android, C++	MQTT WebSockets HTTP	Prácticamente todos
Azure IoT Hub	C, C insertado, C#, Java, Python, Node.js	MQTT AMQP HTTP	Prácticamente todos
Google Cloud IoT Core	JavaScript, Python, Go, Java, Node.js	MQTT HTTP	Prácticamente todos
Thinger.io	Ninguno	MQTT HTTP	Arduino, ESP8266, NodeMCU, Raspberry Pi, entre otros.
ThinkSpeak	MATLAB	MQTT HTTP	Raspberry Pi, Arduino, Módulo WiFi ESP8266, entre otros.

Tabla de ventajas e inconvenientes de cada plataforma en su versión gratuita:

Plataformas	Ventajas	Inconvenientes
AWS IoT	- Plataforma líder y potente con infinitas posibilidades. - Permite conectar varios dispositivos.	- Requiere cierta formación para usarla correctamente. - Es fácil superar los límites de la cuenta gratuita, por lo que hay que tener cuidado. - Dificultad para crear una cuenta gratuita.
Azure IoT Core	- Plataforma muy completa, con gran cantidad de servicios. - Permite conectar varios dispositivos.	- Hay que tener cuidado de no superar los límites de la cuenta gratuita.
Google Cloud IoT Core	- Escalable. - Posibilidad de usar servicios de Google. - Permite conectar varios dispositivos.	- Hay que tener claro los servicios de la capa gratuita no superarlos. - Dificultad para crear una cuenta gratuita.
Thinger.io	- Es de código abierto. - Librería propia. - Interfaz de usuario amplia para visualización de los datos.	- Funciones bastante más limitadas que con el resto de plataformas. - Solo permite conectar 2 dispositivos.
ThinkSpeak	- Es de código abierto. - Librería propia. - Interfaz amigable para visualizar los datos. - Integración con redes sociales. - Escalable. - Funciones de Matlab	- Menos hardware compatible. - Solo permite conectar 1 dispositivo. - Envío de datos a la plataforma cada 15 s como mínimo. - Funciones más limitadas que las plataformas más potentes.

Como se ha expuesto en la tabla anterior, las plataformas más potentes son AWS, Google Cloud y Azure. Sin embargo, en todas ellas es necesario introducir la tarjeta de crédito en el momento de crear la cuenta gratuita, lo que implica que si se utiliza un servicio o herramienta que no esté incluida en esta capa se cobrará aparte. Esto es un gran inconveniente, ya que estas plataformas suelen emplear recursos internos que no son gratuitos, como ocurre con Azure, por ello es más complicado poder realizar un procesamiento y visualización de los datos con este tipo de plataformas. Sin embargo, son algunas de las opciones más utilizadas a nivel industrial empleando cuentas de pago, ya que ofrecen una cantidad de servicios muy interesante bajo esas circunstancias.

En el caso de las otras dos plataformas, Thinger.io y ThingSpeak, este problema no existe, ya que para la creación de una cuenta gratuita no es necesario introducir la tarjeta de crédito. A pesar de que pueden tener menos funcionalidades que las otras plataformas, en sus versiones gratuitas son perfectamente útiles para el desarrollo de prototipos, como es el caso de este TFG.

De las cinco plataformas que se han estudiado, la más interesante para el desarrollo de este proyecto es ThingSpeak, principalmente por las siguientes razones:

- Es compatible con el hardware que se va a utilizar, NodeMCU.
- Solamente se necesita conectar un dispositivo.
- El envío de datos no es necesario que sea cada menos de 15 segundos, ya que se trata de medir la temperatura de una habitación, por lo que el valor no se modifica en tan poco tiempo.
- Es de código abierto.
- Permite integrarlo con redes sociales de forma sencilla.
- Es posible analizar los datos empleando código Matlab desde la propia plataforma.
- Contiene una interfaz para visualizar los datos bastante completa y permite mejorarla.
- Es más funcional que Thinger.io, gracias a la integración con Matlab.
- Contiene una documentación amplia para los dispositivos hardware que incluye.

En el siguiente apartado, se va a realizar un estudio más exhaustivo de la plataforma que se ha elegido, explicando las principales características y funcionalidades que integra.

4.3. ThingSpeak

ThingSpeak cuenta con un soporte comunitario, una gran cantidad de ejemplos y tutoriales, y una documentación extensa, que sirve de gran ayuda para poder desarrollar cualquier proyecto o prototipo.

A continuación, se van a describir los aspectos y funciones más relevantes de la plataforma, como son, los canales, las aplicaciones para realizar análisis y visualizaciones de los datos, y los servicios para realizar diferentes acciones.

La plataforma almacena los datos enviados desde los dispositivos en canales, como ya se explicó anteriormente, estos canales pueden ser configurados como públicos siendo accesibles y visibles para cualquier persona, o privados, a los que solamente se puede acceder desde una cuenta de usuario en ThingSpeak.

Cada canal tiene asignado un código de identificación único (channel ID), y unas claves de API (API Keys) que permiten escribir datos en un canal o leer datos de un canal. Estas claves son generadas automáticamente por la plataforma en el momento en el que se crea un canal nuevo.

En la configuración de cada canal, se pueden añadir hasta 8 campos que pueden contener cualquier tipo de dato, 3 campos para datos de ubicación y uno para datos de estado del canal.

Una vez se han configurados los canales, y se han recopilado datos, la plataforma permite analizarlos y visualizarlos haciendo uso de las diferentes aplicaciones que ofrece.

Ilustración 38.- Servicios para el análisis de datos de ThingSpeak

MATLAB Analysis

Es una herramienta que permite analizar y procesar los datos haciendo uso de código Matlab desde la propia plataforma. Ofrece diferentes plantillas de código de muestra para realizar análisis de datos. Algunos de los análisis que se pueden realizar son los siguientes:

- Encontrar y eliminar datos incorrectos.
- Convertir datos a diferentes unidades.
- Calcular nuevos datos.
- Construir modelos de datos.

Después del procesamiento de los datos es posible escribirlos en un canal o publicarlos para compartir los resultados obtenidos.

MATLAB Visualizations

Este servicio de ThingSpeak utiliza también código Matlab, pero en este caso enfocado a la visualización de los datos. Contiene plantillas y ejemplos de visualización utilizando código Matlab para obtener diagramas de línea interactivos, y también permite crear diagramas personalizados.

Plugins

Son complementos que permiten crear indicadores, gráficos o pantallas personalizadas de Google utilizando código HTML, CSS y JavaScript. Ofrece plantillas de código de muestras a partir de las cuales se puede construir una solución personalizada.

Estos complementos pueden ser añadidos en un sitio web o en la vista de un canal ThingSpeak.

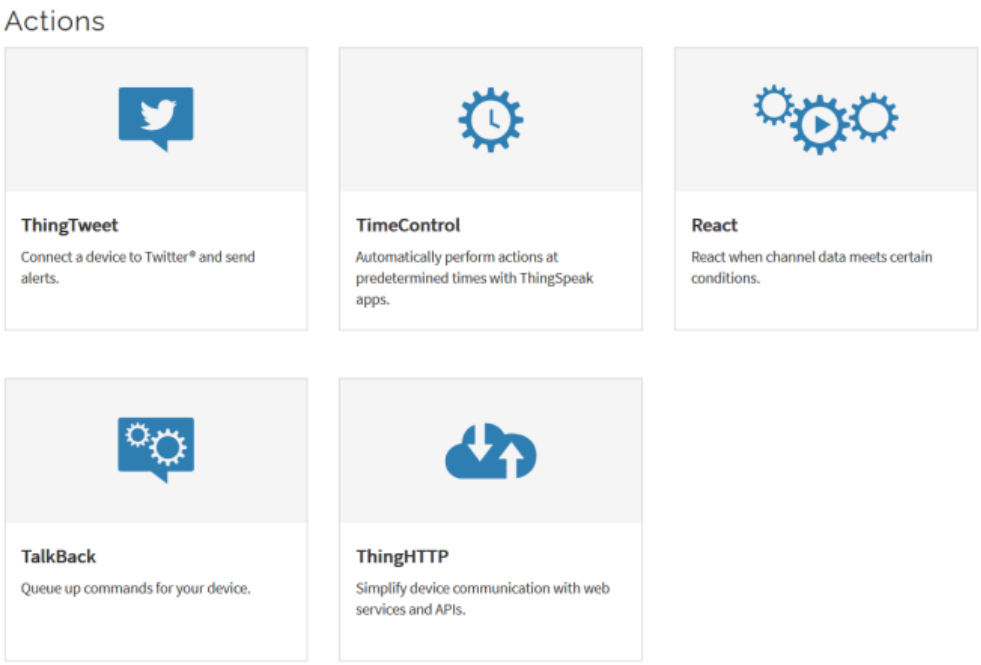


Ilustración 39.- Herramientas para realizar acciones de ThingSpeak

ThingTweet

La aplicación ThingTweet permite vincular una cuenta de Twitter a la cuenta de ThingSpeak. De esta manera es posible enviar alertas a través de Twitter usando ThingTweet.

TimeControl

La herramienta TimeControl funciona con otras aplicaciones de la plataforma y permite realizar una determinada acción en un momento concreto o en un horario regular. Las aplicaciones con las que se utiliza TimeControl son las siguientes:

- ThingHTTP, para comunicarse con dispositivos y sitios o servicios web externos.
- ThingTweet, para enviar alertar a través de Twitter.
- TalkBack, para poner en cola los comandos de un dispositivo.

React

React también trabaja con otras aplicaciones de la plataforma, como ThingHTTP, ThingTweet y MATLAB Analysis. Esta herramienta se encarga de realizar acciones cuando los datos de un canal cumplen una determinada condición que es determinada y configurada por el usuario.

TalkBack

TalkBack es un servicio que permite a cualquier dispositivo actuar sobre los comandos en cola. Por ejemplo, es una herramienta útil para programar una puerta que se abre automáticamente si alguien se acerca y se cierra después de varios minutos de forma automática si no detecta movimiento.

ThingHTTP

Permite la comunicación entre dispositivos y sitios o servicios web externos sin la necesidad de implementar el protocolo en el nivel del dispositivo.

Esta herramienta se puede ejecutar haciendo uso de otras aplicaciones de la plataforma, como TimeControl y React.

Algunos ejemplos de acciones que pueden realizarse con esta herramienta son los siguientes:

- Activar notificaciones de IFTTT.
- Enviar actualizaciones automáticas a través de Prowl.
- Hacer llamadas con Twilio.

5. Entornos de desarrollo para aplicaciones Android

Uno de los objetivos de este TFG es desarrollar una aplicación móvil con la que sea posible visualizar y controlar los datos obtenidos por el medidor IoT.

En este caso se va a desarrollar una aplicación Android, ya que es unos de los sistemas operativos más comunes en el mercado y al que posiblemente pueda acceder un mayor número de usuarios.

Primeramente, se va a explicar en qué consiste este sistema operativo y posteriormente se expondrán algunos entornos de desarrollo más comunes que permiten implementar aplicaciones Android, eligiendo, finalmente, el entorno más adecuado teniendo en cuenta las características de este proyecto.

ANDROID

Android es un sistema operativo basado en Linux, fue desarrollado por Google en el año 2008. Tiene como objetivo ser utilizado en dispositivos como Teléfonos inteligentes, Tablets y Google TV, entre otros [23].

Algunas de sus características son las siguientes:

- Es de código abierto.
- Utiliza SQLite para almacenar datos.
- Navegador web incluido basado en WebKit.
- Soporte de Java y otros formatos multimedia.
- Soporte de HTML, Adobe Flash Player, entre otros.
- Bluetooth.
- Catálogo de aplicaciones gratuitas o de pago.

Su arquitectura está basada en los siguientes bloques:

Aplicaciones: Incluye aplicaciones escritas en lenguaje Java como, correo electrónico, programa de SMS, calendario, mapas, navegador y contactos, entre otras.

Marco de trabajo de aplicaciones: El diseño de la arquitectura Android se basa en la reutilización de componentes, por lo que cualquier aplicación puede hacer públicas sus capacidades y otra aplicación cualquiera puede hacer uso de esas mismas capacidades posteriormente.

Bibliotecas: Incluye un conjunto de bibliotecas de C y C++, que son utilizadas por varios componentes del sistema.

Runtime de Android: Este sistema operativo utiliza su propia máquina virtual llamada Dalvik, en la que se ejecutan todas las aplicaciones.

Núcleo Linux: Android utiliza Linux para los servicios base del sistema, y son los siguiente:

- Seguridad.
- Gestión de memoria.
- Gestión de procesos.
- Pila de red.
- Modelo de controladores.

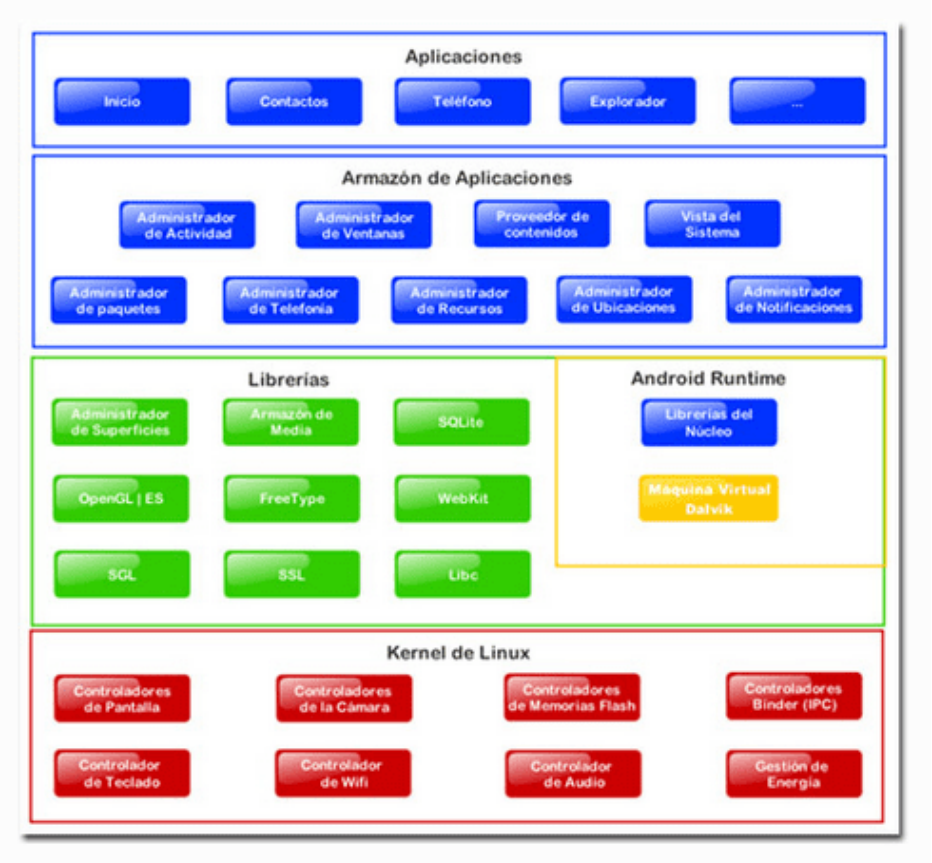


Ilustración 40.- Arquitectura Android [23]

Antes de continuar, se van a explicar algunos tipos de aplicaciones que existen, ya que pueden ser nombradas más adelante [24]

Aplicaciones nativas

Son aquellas creadas específicamente para un sistema operativo concreto, como puede ser iOS o Android. Esto significa que son desarrolladas en el lenguaje de programación oficial de cada sistema, por ejemplo, las aplicaciones para Android se desarrollan en Java y para iOS en lenguaje Swift.

La ventaja de estas aplicaciones es que se adaptan perfectamente a las funcionalidades del móvil y son más eficientes. Sin embargo, el coste de desarrollarlas es más elevado y es necesario implementar una aplicación para cada sistema operativo.

Aplicaciones web

Este tipo de aplicaciones pueden ser ejecutadas en cualquier dispositivo o navegador, lo que implica que están programadas con independencia del sistema operativo. Los lenguajes de programación más comunes para desarrollar estas aplicaciones son HTML y CSS. No necesitan ser instaladas en el propio dispositivo, sino que se adaptan a dicho dispositivo ejecutando el URL desde el navegador.

Aplicaciones híbridas

Estas aplicaciones son una mezcla de las dos anteriores, combinan características de las aplicaciones nativas y de las aplicaciones web y se desarrollan con lenguajes de programación comunes de las aplicaciones web, como HTML o CSS.

Se caracterizan porque su desarrollo es rápido y permite acceder a diferentes sistemas operativos con un esfuerzo menor que las aplicaciones nativas.

Aplicaciones web progresivas

Este tipo de aplicaciones incorporan elementos de las aplicaciones web tradicionales y algunas particularidades de las aplicaciones nativas, por lo que son capaces de realizar ciertas funciones para ofrecer mejores resultados que una aplicación web tradicional.

Suelen estar programadas en lenguaje HTML, CSS o JavaScript.

No es necesario instalarlas en el dispositivo, si no que solo se necesita de conexión a internet y un navegador, además permiten ser ejecutadas en cualquier sistema operativo.

A continuación, se van a explicar algunos de los entornos de desarrollo más conocidos para la creación de aplicaciones Android, y entre todos ellos, se decidirá cuál es el que aporta mayores beneficios a la hora de implementar este proyecto.

5.1. Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para desarrollar aplicaciones Android. Está basado en el software IntelliJ IDEA de JetBrains, esta empresa también desarrolló el lenguaje de programación Kotlin [25].

Contiene un sistema de compilación flexible, un emulador rápido y con funciones, un entorno unificado, compatibilidad con C++, Java y Kotlin. Estas características junto con muchas otras hacen que sea una de las mejores opciones a la hora de implementar una aplicación móvil. Sin embargo, exige tener conocimientos y una formación suficiente para utilizarlo de forma eficiente.

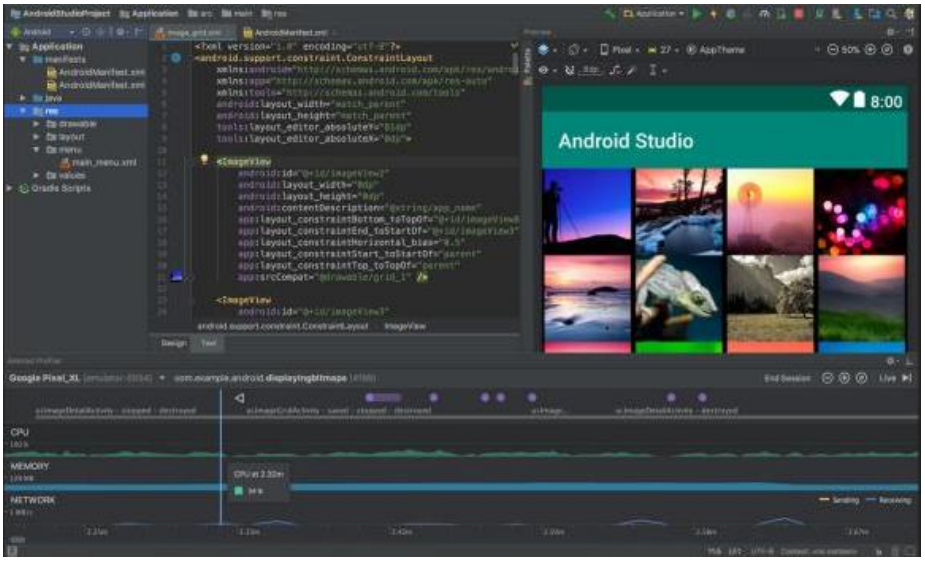


Ilustración 41.- Interfaz de usuario de Android Studio [25]

5.2. Apache Cordova

Apache cordova es un entorno de desarrollo de aplicaciones móviles bastante popular. Fue consolidado como un proyecto de nivel superior en 2012 dentro de la Apache Software Foundation (ASF). Es de código abierto y permite construir aplicaciones para dispositivos móviles con lenguajes de programación como HTML, CSS y JavaScript [26].

Las aplicaciones que se generan son híbridas, lo que significa que no son ni aplicaciones móviles nativas ni basadas únicamente en web.

5.3. Ionic

Ionic es un Kit de desarrollo software o SDK (Software Development Kit) completo de código abierto para la creación de aplicaciones híbridas [27].

Fue desarrollado en 2013 por Max Lynch, Adam Bradley y Ben Sperry de Drifty Co. La versión original fue construida a partir de Apache Cordova y AngularJS.

Para la creación de aplicaciones emplea tecnologías como HTML, CSS y JavaScript, con posible integración de otros marcos populares como Angular, React y Vue.

Permite a los desarrolladores crear aplicaciones que funcionen en múltiples plataformas como iOS, Android, escritorio y Web como aplicaciones web progresivas, todo con una base de código.

Utiliza Capacitor o Cordova para implementar aplicaciones nativas, o se ejecuta en el navegador como una aplicación web progresiva.

La creación de aplicaciones con Ionic es agradable, fácil de aprender y accesible para aquellas personas que ya cuentan con ciertas habilidades en desarrollo web.

5.4. React Native

Es un entorno de trabajo enfocado a la creación de aplicaciones nativas para los sistemas operativos iOS y Android. Es de código abierto y fue creado por Facebook [28].

Está basado en la librería de JavaScript para crear componentes visuales y utiliza las capacidades tanto de JavaScript como de React para desarrollar aplicaciones más complejas.

RectNative puede ser sencillo de aprender si se tienen conocimientos previos de JavaScript, ya que es un entorno intuitivo para el usuario.

Permite desarrollar código nativo utilizando lenguajes como Java o Kotlin para Android y Objective-C para iOS, lo que le caracteriza de mayor versatilidad.

5.5. MIT App Inventor

Es un entorno de programación visual e intuitivo, de código abierto, que permite a cualquier usuario crear aplicaciones completamente funcionales para teléfonos inteligentes y tablets. Fue creado por Google Labs para elaborar aplicaciones destinadas al sistema operativo Android, y está administrado por el Centro del MIT (Instituto Tecnológico de Massachusetts) [29].

Este entorno de desarrollo apareció en 2008 y está enfocado a personas con un nivel de programación básico.

Contiene un conjunto de herramientas básicas basadas en bloques que facilita la creación de aplicaciones complejas de alto impacto en menos tiempo que los entornos de programación tradicionales.

Es un sistema gestionado en la nube, por tanto, las aplicaciones se desarrollan desde un navegador. Contiene principalmente tres herramientas, estas son, un gestor de proyectos, una interfaz para el diseño y un editor de bloques.

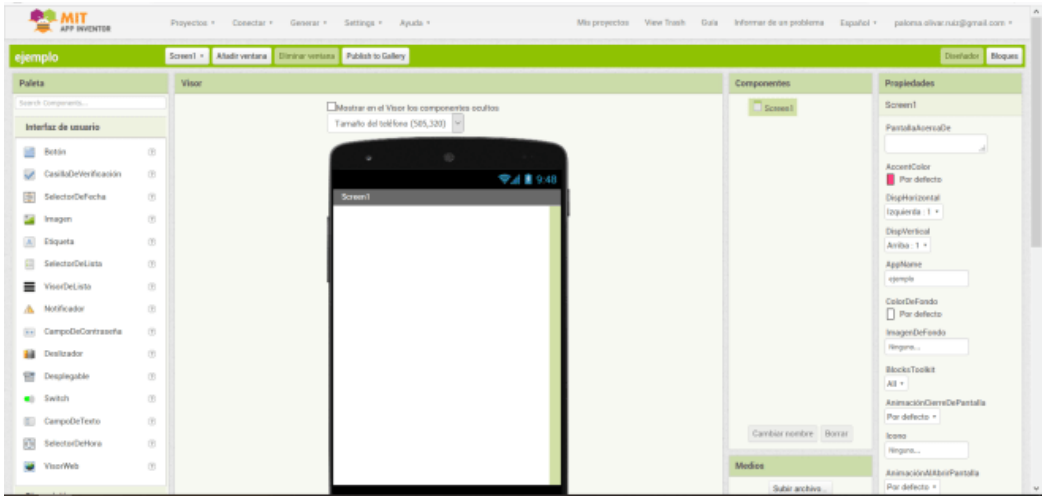


Ilustración 42.- Interfaz de usuario de MIT APP INVENTOR [29]

Las principales ventajas que tiene esta herramienta son, que está basada en un entorno visual que hace que sea muy intuitivo desarrollar aplicaciones perfectamente funcionales, y que no emplea lenguaje de programación como tal, por lo que cualquier usuario sin conocimientos en este ámbito puede hacer uso de ella.

Para el desarrollo de la aplicación móvil de este proyecto se ha elegido este entorno de desarrollo, ya que es una herramienta versátil perfectamente útil para implementar aplicaciones básicas enfocadas a prototipos.

El objetivo de crear una aplicación en este TFG es poder visualizar los datos leídos por el medidor IoT y realizar una acción sobre los mismo, por lo que no es necesario implementar una aplicación profesional haciendo uso de lenguajes de programación tradicionales. Este entorno de desarrollo ofrece todas las herramientas necesarias para desarrollar una aplicación que cumpla con los objetivos de este proyecto.

6. Transmisión y gestión de los datos

A continuación, se van a explicar algunos de los medios de transmisión de datos enfocados en la tecnología IoT que podrían emplearse durante la implementación del prototipo, y se optará por el que mayores beneficios ofrezca.

También se verán los protocolos de comunicación más comunes empleado en IoT, y que son los que van a permitir realizar la comunicación entre el dispositivo y la plataforma IoT.

6.1. Medio de transmisión

La forma en la que se van a enviar los datos desde el dispositivo IoT a la nube es una parte importante del proyecto que se está llevando a cabo, ya que de ello va a depender la elección del propio hardware.

La comunicación inalámbrica es la más común en un sistema IoT. Permite enviar y recibir datos desde el dispositivo a la nube, pudiendo así actuar sobre dicho dispositivo, sin necesidad de utilizar cableado, ya que en los sistemas inalámbricos la comunicación se lleva a cabo a través de ondas electromagnéticas.

Este tipo de medio otorga mayor versatilidad que un sistema cableado en cuanto a movilidad y alcance. Siendo con ello la mejor elección para diseñar un sistema IoT, como es este caso.

A continuación, se van a describir algunas de las tecnologías inalámbricas más comunes y se decidirá cuál es la más apropiada para el caso concreto de este TFG.

Bluetooth

Bluetooth es una especificación industrial orientada a redes inalámbricas de área personal (wireless personal area network o WPAN) [30].

Tiene como principales objetivos eliminar el uso de cables y conectores, conectar dispositivos móviles entre sí, crear pequeñas redes y hacer más fácil la sincronización entre dispositivos personales.

La transmisión de datos entre los diferentes dispositivos se lleva a cabo a través de un enlace por radiofrecuencia en la banda de 2.4 GHz.

Los sectores que más hacen uso de esta tecnología son los relacionados con la informática y las telecomunicaciones.

Los dispositivos que incorporan Bluetooth pueden comunicarse entre sí cuando se encuentran dentro de su alcance, este depende de la potencia de transmisión que tenga cada dispositivo.

Algunas ventajas con las que cuenta esta tecnología son las siguientes:

- Funcionamiento con dispositivos de bajo consumo.
- Es de bajo costo.
- La forma de vincular los dispositivos es sencilla.
- Puede coexistir con otras redes inalámbricas.

Sin embargo, tiene algunos inconvenientes que pueden ser determinantes a la hora de elegir esta especificación:

- El alcance máximo es de aproximadamente 10 m.
- Es posible conectar 8 dispositivos como máximo.
- No siempre el funcionamiento es adecuado, debido a que la banda de 2.4 GHz puede estar saturada, o el número de dispositivos conectados ser elevado.

Zigbee

Zigbee es el nombre de la especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica, y se utiliza con radiofrecuencia digital de bajo consumo [31].

Está basado en el estándar IEEE 802.15.4 de redes inalámbricas de uso personal (WPAN).

Tiene como principal objetivo ser utilizado en aplicaciones que requieren comunicaciones seguras y fiables, tasas de envío de datos bajas y que necesitan aumentar la vida útil de sus baterías.

Cuenta con un rango de alcance comprendido entre 10 y 75 metros, y emplean tecnología inalámbrica con velocidades entre 20 kB/s y 250 kB/s.

Algunas de las ventajas son las siguientes:

- Disminuye los tiempos de espera en el envío y recepción de paquetes.
- Proporciona larga duración a la batería debido al bajo ciclo de trabajo.
- Contiene soporte para diferentes topologías de red.
- Permite conectar hasta 65.000 nodos en una red.

También tiene algunos inconvenientes:

- La tasa de transferencia es baja.
- No es compatible con Bluetooth.
- El precio de los dispositivos con esta tecnología es mayor que los que emplean Bluetooth o WiFi.
- Trabaja en la banda de 2.4 GHz para redes inalámbricas.

WiFi

Es una tecnología para conectar dispositivos de forma inalámbrica, pertenece a la organización comercial Alianza WiFi y cumple con los estándares 802.11 basados en redes inalámbricas de areal local (wireless local area network o WLAN) [32].

Los dispositivos con WiFi, como ordenadores o teléfonos móviles, pueden conectarse a internet o entre sí mediante un punto de acceso de red inalámbrica.

El alcance de una red WiFi depende de la banda de frecuencia que utilice para la transmisión de la información. Si emplea la banda de 2.4 GHz el alcance puede ir desde 45m en interiores hasta 90m en exteriores aproximadamente, cuando utiliza la banda de 5 GHz el alcance puede ser desde 15m en interiores hasta 30m en exteriores. Aunque el alcance también depende de otros factores como el número de barreras físicas, por ejemplo, paredes, puertas o techos.

Algunas de las ventajas con las que cuenta esta tecnología son las siguientes:

- Ofrece mayor alcance que Bluetooth o Zigbee, debido a que es una red de tipo WLAN.
- Es compatible con Bluetooth y redes Ethernet.
- Cuenta con tasas elevadas de transferencia de datos.
- Soporta un gran número de dispositivos conectados.
- Es de bajo costo y está muy extendida.

También tiene algunos inconvenientes:

- Emplea la banda de 2.4 GHz, la cual puede estar saturada y el funcionamiento se puede ver alterado.
- El consumo de energía es elevado.
- Es importante garantizar la seguridad de los datos.
- El alcance se ve afectado por el entorno.

El medidor IoT que se va a diseñar en este TFG está enfocado al ámbito doméstico, por lo que la tecnología WiFi es la más adecuada en este caso, debido a que es la más empleada en los hogares. Además, tiene mayor alcance y menor coste que las otras tecnologías que se han visto. Estas ventajas junto con las demás que se han descrito anteriormente, hacen que sea el medio de transmisión más interesante para el desarrollo del prototipo.

6.2. Protocolo de comunicación IoT

Un protocolo de comunicación es un conjunto de reglas definidas para que dos o más dispositivos o elementos de un sistema de comunicación puedan comunicarse entre sí y transmitir información.

Como ya se ha explicado anteriormente, IoT se basa en la comunicación e interconexión entre diferentes dispositivos, por lo que establecer un protocolo para ello es una parte imprescindible.

Los sistemas IoT exigen una serie de requisitos especiales que hay que tener en cuenta para elegir el protocolo adecuado. Por ejemplo, conexión de varios dispositivos, sistema escalable, interoperabilidad, seguridad y acceso fácil a los dispositivos [33].

A continuación, se van a describir algunos de los protocolos de comunicación más empleados en IoT.

AMQP

El protocolo de cola de mensajes avanzados o AMQP por sus siglas en inglés (Advance Message Queuing Protocol) es un protocolo estándar y abierto de la capa de aplicación [34].

Tiene como principales características la seguridad, la confiabilidad y la interoperabilidad. Por ello, es perfecto para utilizarse en entidades comerciales del sector financiero, aunque también puede utilizarse como protocolo de IoT.

Los elementos que intervienen en el intercambio de mensajes se explican a continuación:

El **publicador** (publisher): es la entidad encargada de crear y enviar el mensaje.

Consumidor (consumer): es la entidad que recibe el mensaje de la cola.

Bróker: es el elemento que actúa como punto intermedio entre el publicador y el consumidor, y se encargará de distribuir el mensaje.

Exchange: se encarga de coger los mensajes enviados al bróker desde el publicador y enrutarlos hacia una o más colas.

Binding: es el elemento que determina las reglas que va a utilizar el Exchange para decidir a qué cola debe enviar cada mensaje.

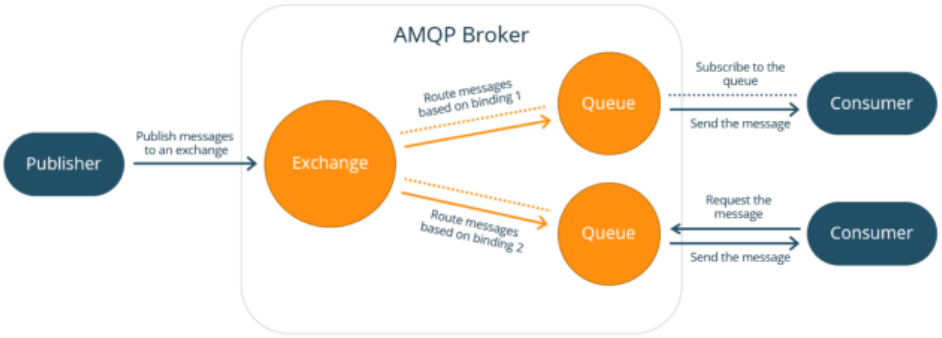


Ilustración 43.- Estructura protocolo de comunicación AMQP [35]

HTTP

El protocolo de transferencia de hipertexto (HTTP, Hypertext Transfer Protocol, en inglés) es un protocolo que permite la transferencia de información de archivos de tipo HTML o XTML en la WWW (World Wide Web). Se considera la base de intercambio de información en la web. Fue diseñado en la década de 1990, y es un protocolo de la capa de aplicación [36].

Se caracteriza por ser un protocolo ampliable, sencillo y sin estado, es decir, no permite guardar la información de conexiones anteriores, para lo que actualmente se emplean las cookies.

Debido a su capacidad de ampliación, además de utilizarse para transmitir documentos de hipertexto (HTML), también se emplea para transmitir imágenes, videos, datos o diferentes tipos de contenido a los servidores.

Es un protocolo de estructura cliente-servidor, y su arquitectura se basa en el siguiente esquema:

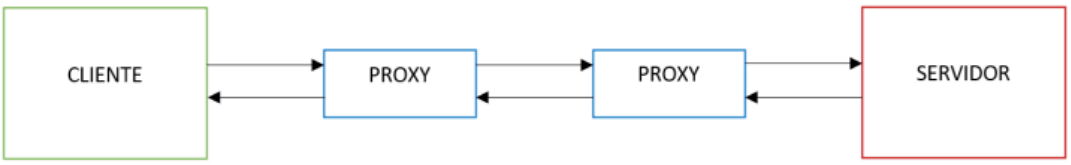


Ilustración 44.- Estructura protocolo de comunicación HTTP

Cliente: Es la entidad que envía las peticiones al servidor, la mayoría de las veces es un navegador web.

Servidor: Es la entidad que se encuentra al otro lado del canal de comunicación y se encarga de gestionar y enviar los datos que ha pedido el cliente.

Proxies: Se trata de programas o dispositivos que hacen de intermediarios entre las peticiones que hace el cliente al servidor, algunas de las funciones que realizan son caching, filtrado (Anti-virus, Control parental), Registro de eventos y Autenticación.

MQTT

MQTT (Message Queue Telemetry Transport) está constituido sobre la pila TCP/IP (protocolo subyacente de internet) y se ha convertido en el estándar más utilizado en las comunicaciones IoT [37].

Fue inventado y desarrollado por IBM en los años 90 y en 2014 pasó a ser oficialmente un estándar abierto.

Es un protocolo de mensajería que soporta la comunicación asíncrona, lo que significa que es capaz de disociar al emisor y al receptor de los mensajes en espacio y tiempo. Se soporta en lenguajes de programación populares haciendo uso de implementaciones de código abierto.

Utiliza el modelo publicación y suscripción (Publish and subscribe) como se muestra en la siguiente imagen:

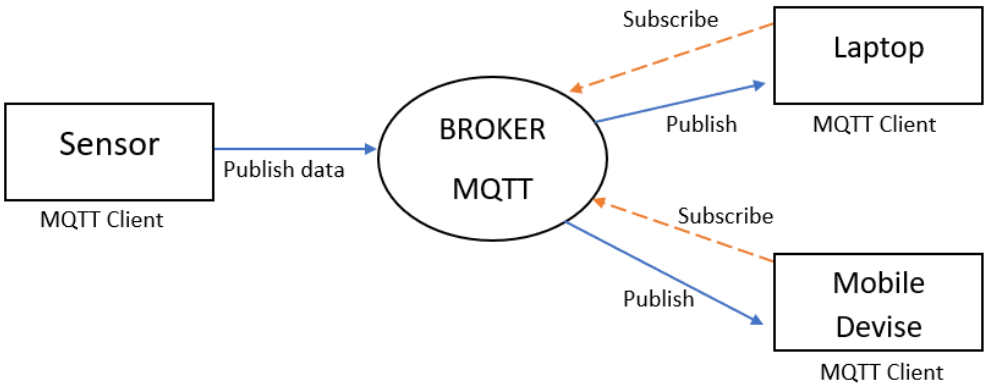


Ilustración 45.- Estructura modelo publish and subscribe de MQTT

En este protocolo se definen dos tipos de entidades principalmente, un intermediario de mensajes y un número de clientes.

Intermediario: Es un servidor que recibe todos los mensajes enviados por los clientes y se encarga de redirigirlos a los clientes.

Cliente: Es cualquier elemento capaz de interactuar con el intermediario para enviar y recibir mensajes. Puede ser un sensor de IoT o una aplicación que procesa datos de IoT.

El cliente se conecta con el intermediario (Broker o Servidor) y se puede subscribir a cualquier “Tema” de mensajes del intermediario.

El cliente publica el mensaje, con un determinado Tema, y envía el mensaje junto con el Tema al intermediario.

A continuación, el intermediario se encarga de redirigir el mensaje a todos los clientes que están subscritos a ese Tema en concreto.

Este protocolo se utilizará para implementar una parte del código que se va a desarrollar en el IDE Arduino, y permitirá realizar la comunicación del dispositivo con la plataforma IoT. Su programación se explicará en el apartado 7.3 del proyecto.

7. Diseño del medidor IoT

En este apartado se va a explicar en qué consiste, de manera concreta, el prototipo del medidor de temperatura y humedad IoT que se ha construido, y cuál ha sido el procedimiento llevado a cabo para su implementación. Se explicará el montaje del hardware, el código desarrollado, de qué manera se ha trabajado con la plataforma IoT, y cómo se ha creado la aplicación móvil que permite visualizar los datos de temperatura y humedad, y actuar sobre el dispositivo IoT.

7.1. Descripción de la aplicación real

Para la construcción del medidor IoT se ha partido de la placa de desarrollo NodeMCU V3 y el sensor de temperatura y humedad DHT22, posteriormente se explicará con detalle el montaje del hardware completo.

El sensor leerá los datos de temperatura y humedad del ambiente y mediante el código que se ha desarrollado en el IDE Arduino, el dispositivo se conectará con la plataforma IoT ThingSpeak, donde se enviarán los datos para ser visualizados y procesados.

Se desarrollará una aplicación móvil empleando el entorno de desarrollo MIT App Inventor, con la que también se podrán visualizar los datos recogidos por el medidor y actuar sobre el propio dispositivo.

La idea de emplear un medidor de temperatura y humedad IoT en un hogar es poder controlar la temperatura de forma remota, lo que se traduce en poder encender o apagar la caldera dependiendo de los datos que se observen. En este caso, se ha empleado un Led de color azul que está conectado con el NodeMCU, y que será el elemento sobre el que se actúe desde la aplicación móvil o desde la plataforma IoT. La función de este Led es simular el encendido o apagado de una caldera, y demostrar con ello, que es posible actuar sobre el dispositivo y los datos de forma remota.

La estructura del diseño del medidor se muestra en la siguiente imagen:

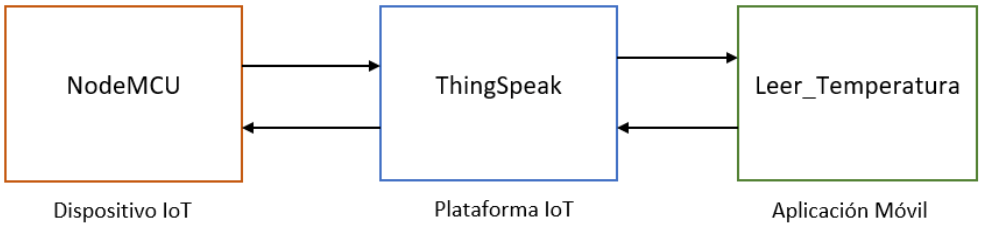


Ilustración 46.- Estructura del diseño del Medidor IoT

En los próximos apartados se va a describir de manera precisa como se ha implementado cada parte de la aplicación.

7.2. Montaje del hardware

Para realizar el montaje completo del hardware, se parte de la placa de desarrollo NodeMCU V3, y del sensor de temperatura y humedad DHT22.

El sensor tiene 3 pines útiles para realizar las conexiones, VDD, DATA y GND, como se explicó en el apartado 3.3.

Para realizar las conexiones entre ambos elementos es importante leer las recomendaciones del Datasheet del sensor, que principalmente son las siguientes:

- La tensión típica de alimentación del sensor es de 5 V, por lo que el pin VDD del sensor debe conectarse al pin VU del NodeMCU, que es el único pin que ofrece 5 V de salida.
- Es aconsejable añadir una resistencia pull-up de 10KΩ entre los pines VDD y DATA del sensor, con el objetivo de asegurar un estado HIGH o alto del pin DATA cuando no haya transmisión de datos.
- Se recomienda emplear un condensador de 100nF entre los pines VDD y GND del sensor para evitar el posible rizado que pudiera tener la señal de alimentación.

Como ya se ha dicho anteriormente, se ha añadido un Led azul con su correspondiente resistencia de 220Ω para limitar la corriente de salida del NodeMCU, y con ello, evitar que se funda el Led.

Este led está conectado al pin D1 del NodeMCU y el pin DATA del sensor al pin D5.

El siguiente esquema muestras las correspondientes conexiones:

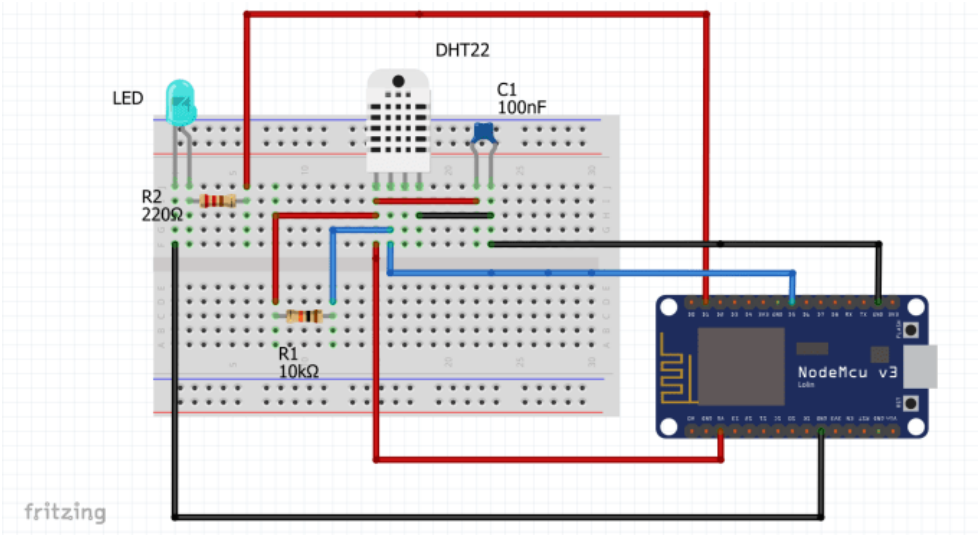


Ilustración 47.- Montaje y conexiones del Hardware [38]

Por último, la placa NodeMCU V3 se va a alimentar conectándola por USB al ordenador, ya que este dispositivo cuenta con un conversor serial, además, servirá para cargar el código implementado en el IDE Arduino al dispositivo.

La siguiente imagen muestra el montaje real del Hardware sobre una protoboard:

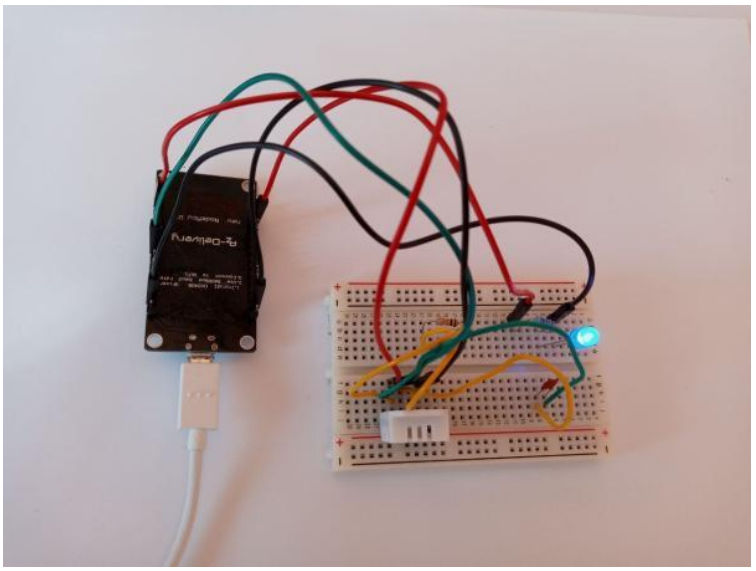


Ilustración 48.- Montaje real del Hardware del prototipo

7.3. Código en Arduino IDE

Para la programación del microcontrolador se ha decidido utilizar el IDE Arduino, ya que es un entorno conocido, permite trabajar con un gran número de placas de desarrollo, y tiene la posibilidad de utilizar e instalar multitud de librerías.

En primer lugar, se ha instalado y configurado la placa NodeMCU en el IDE Arduino, para lo cual se han seguido los siguientes pasos:

1. EL NodeMCU no aparece como tal dentro del entorno de Arduino, por lo que es necesario instalar las placas adicionales que tienen el módulo ESP8266, ya que es el que contiene el microcontrolador. Accediendo a la pestaña Archivo → Preferencias, en gestor de URL adicionales de Tarjetas, añadiendo la siguiente dirección: https://arduino.esp8266.com/stable/package_esp8266com_index.json.

2. Accediendo a la pestaña Herramientas → Placa: → Gestor de tarjetas, y buscando "esp", se instala la que se denomina "esp8266 by ESP8266 Community".

3. Se selecciona la placa con la que se va a trabajar, en la pestaña Herramientas → Placa → "NodeMCU 1.0 (ESP-12E Module)".

4. Configuración de los parámetros de la placa, desde la pestaña Herramientas:

    - CPU Frequency: "80 MHz".
    - Upload Speed: "115200".
    - Flash Size: "4 MB (FS:1MB OTA: ~1019KB)".
    - Debug Port: "Disabled".
    - Debug level:" Ninguno".
    - IwIP Variant: "v2 Lower Memory".

Una vez se ha instalado la placa en cuestión, ya es posible trabajar en el código, cuyo objetivo es realizar la conexión con la plataforma IoT, enviando los datos recogidos por el sensor y recibiendo también datos desde la plataforma para actuar sobre el Led.

En el código se han desarrollado dos métodos diferentes para realizar la conexión con la plataforma IoT, el primero de ellos emplea la librería propia de la plataforma IoT (ThingSpeak), y el segundo método consiste en una estructura publicador-subscriptor basada en el protocolo de comunicación MQTT.

Se ha implementado un código modular con el objetivo de hacerlo más legible y manejable, para poder diferenciar entre estos dos métodos de conexión con la plataforma IoT, y detectar los posibles errores que se puedan generar de manera más sencilla.

El código está dividido en tres subprogramas:

- El programa principal se denomina "App_ThingSpeak", en este se incluyen las librerías necesarias, la definición de variables, las dos estructuras principales del código (Setup () y loop ()), y las funciones para conectarse a internet *WiFiConnection ()* y para leer los datos del sensor *readSensor()*, ya que se ejecutan siempre independientemente del método utilizado para realizar la conexión con la plataforma IoT.

- El segundo subprograma "MQTT_Connection" incluye las funciones necesarias para realizar la conexión con la plataforma siguiendo el método publicador-subscriptor, también incluye las funciones para enviar y recibir los datos.

- El tercer subprograma "ThingSpeak_Library" contiene las funciones de la librería *ThingSpeak.h* para enviar y recibir datos hacia o desde la plataforma.

A continuación, se va a explicar el funcionamiento del código. Para una mejor comprensión se recomienda ver el Anexo en el que está incluido el código completo.

Se instalan las librerías que se van a necesitar en el código y se añaden en el programa principal "App_ThingSpeak":

- *DHT.h*: Librería de la familia de sensores DHT, en la que se encuentra el sensor DHT22 que se va a emplear en este caso.
- *ESP8266WiFi.h*: Para poder realizar la conexión a internet con el módulo ESP8266.
- *ThingSpeak.h*: Librería de la plataforma IoT ThingSpeak.

- *PubSubClient.h*: Librería para implementar el método Publicador-subscriptor.

A continuación, también dentro del programa principal, se definen los pines a los que se han conectado el sensor (D5), el Led (D1), y el tipo de sensor dentro de la familia DHT que se va a utilizar, en este caso DHT22.

Se definen los dos métodos de conexión con la plataforma, Tpeak con el que se realiza la conexión utilizando la librería de la plataforma y MQTT en el que se implemente el protocolo MQTT.

Se añaden las credenciales para realizar la conexión WiFi, es decir, el nombre del WiFi y la contraseña.

Se incluyen tanto los identificadores de los canales de la plataforma proporcionados por la misma, como las claves de lectura y escritura de cada canal.

Se añade el nombre identificador que se ha asignado al dispositivo que actúa como cliente (ESP-Thingspeak), y la clave MQTT proporcionada por la plataforma IoT, que se necesitará para realizar la conexión con la misma.

Se declaran los campos de cada canal que se van a utilizar posteriormente en el código (fields), y el nombre del servidor o bróker de ThingSpeak "mqtt.thingspeak.com".

En primer lugar, se va a explicar el funcionamiento del programa utilizando el método que emplea la librería de ThingSpeak para realizar la conexión con la plataforma, y enviar y recibir la información.

Método Tpeak

El diagrama de flujo es el que se muestra en la siguiente imagen, y muestra el orden en el que se ejecutan las diferentes funciones:

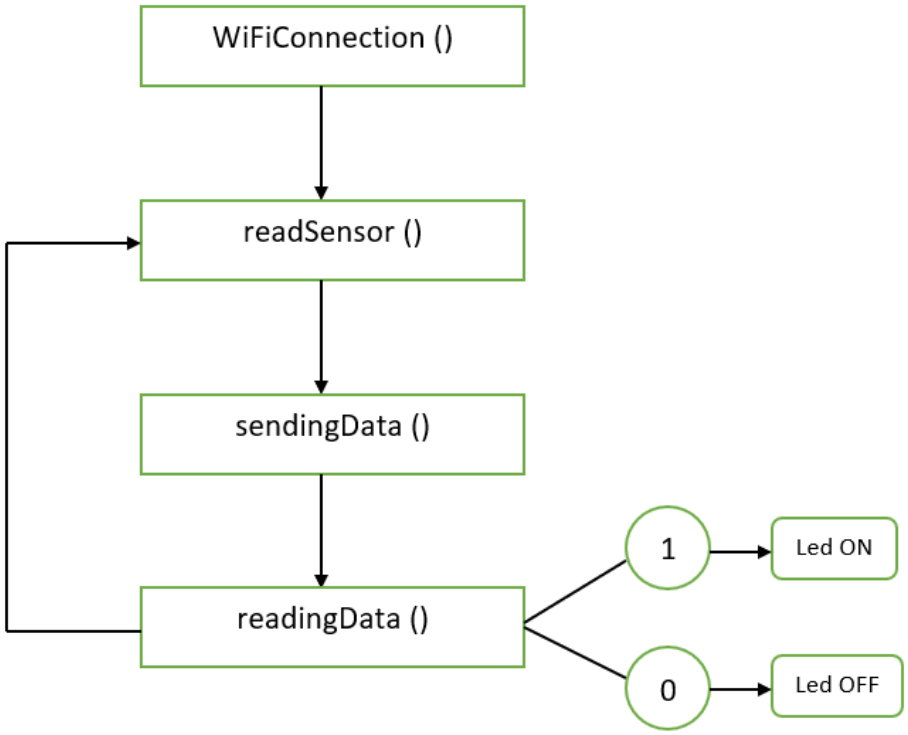


Ilustración 49.- Diagrama para conexión NodeMCU-ThingSpeak con método Tpeak

La función **WiFiConnection ()** es la encargada de realizar la conexión con internet, para lo que utiliza las credenciales WiFi que se han definido al principio del código.

La siguiente función en ejecutarse es **readSensor (),** con la que se leen los datos de temperatura y humedad del sensor, y se almacenan en un vector.

Una vez leído los datos del sensor, y guardados en un vector, se envían como argumentos a la función **sendingData ()**, que además utiliza las claves y los campos de los canales de ThingSpeak necesarios para enviar esta información a la plataforma.

Esta función en primer lugar carga los valores de temperatura y humedad en los campos correspondientes de cada canal y posteriormente lo envía a la plataforma. Para realizar esta acción utiliza las siguientes funciones de la librería *ThingSpak.h*:

ThingSpeak.setField(); → Para cargar los datos en un campo.
ThingSpeak.writeFields(); → Envía y escribe los datos en el canal correspondiente.

La última función en ejecutarse es **readingData ()**, la cual tiene como argumentos las claves y campos necesarios para leer los datos del canal de la plataforma, para lo cual utiliza la siguiente función de la librería *ThingSpeak.h*:

data = ThingSpeak.readFloatField();

El dato leído del canal se almacena en la variable *data*, este valor solo podrá ser 0 o 1. En el caso de que *data* valga 1 se encenderá el Led y si vale 0 se apagará. Para ello, se ha utilizado una sentencia *if else,* como muestra la siguiente imagen:

```
if(data == 1){
  digitalWrite(LED, HIGH);
  Serial.println("LED ON\n");
}
else{
  digitalWrite(LED, LOW);
  Serial.println("LED OFF\n");
}
```

Ilustración 50.- Sentencia de código para encendido o apagado del Led

Después de ejecutarse esta función, el código volvería a empezar leyendo los datos del sensor. En este caso se está realizando una lectura del sensor cada minuto, para ello se utiliza un *delay (60000)*, es decir, una espera de 60 segundos. Sin embargo, para realizar pruebas y ver el funcionamiento del código es recomendable disminuir el tiempo a 15 segundos, ya que es el intervalo mínimo de tiempo que ThingSpeak permite entre lectura de datos.

A continuación, se va a explicar el funcionamiento del programa utilizando el método que emplea el modelo publicador-subscriptor para realizar la conexión con la plataforma, y enviar y recibir los datos.

Método MQTT

El siguiente diagrama de flujo muestra el orden en el que se ejecutan las diferentes funciones para realizar la conexión con ThingSpeak utilizando la estructura publish and subscribe basada en el protocolo MQTT, y que ya se explicó en el apartado 6.2 de este TFG.

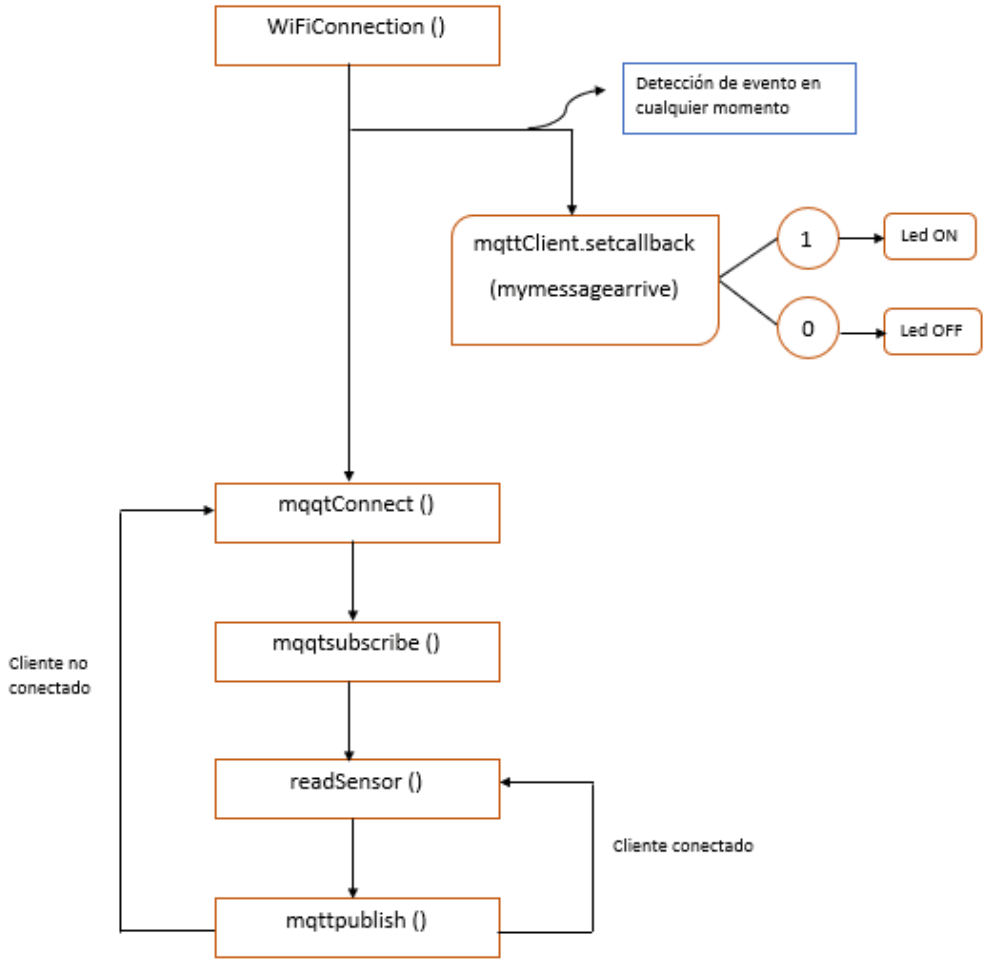


Ilustración 51.- Diagrama para conexión NodeMCU-ThingSpeak con método MQTT

Al igual que ocurría con el primer método de conexión, la primera función en ejecutarse es **WiFiConnection ()** para realizar la conexión del dispositivo a Internet, esta función es exactamente la misma en los dos métodos.

A continuación, se realiza la conexión con el cliente MQTT, es decir, con el NodeMCU. Esto se realiza utilizando la función **mqttConnect ()** que se ha creado.

Esta a su vez hace uso de otra función que pertenece a la librería *PubSubClient.h,* denominada *mqttClient.connect()*, a la cual se le envían como argumentos el identificador único del cliente (clientID), el nombre identificador que se ha establecido del cliente al principio del código (clientName) y la clave MQTT proporcionada por la plataforma (MQTTAPIKey).

El identificador del cliente (clientID), es importante que sea único, ya que de no ser así es posible que no se realice correctamente la conexión. Para ello, se ha creado la función **getID ()**, la cual se encarga de generar un código aleatorio y único que se utilizará como identificador de cliente al realizar la conexión. Esta es una forma de asegurar que el identificador siempre será único.

Una vez el cliente está conectado, se realiza la suscripción al "tema" correspondiente, en este caso, hay que subscribirse al campo del canal de ThingSpeak del que se van a leer los datos. Para ello, se utiliza la función **mqttSubscribe()** que realiza la suscripción utilizando la función mqttClient.subscribe() de la librería *PubSubClient.h* enviándole como argumento el "tema" al que debe subscribirse.

A continuación, se leen del sensor los datos de temperatura y humedad con la función **readSensor ()**, de la misma manera que ocurría con el primer método de conexión. Esta función también se ejecuta de igual manera en ambos métodos, aunque en diferente posición de ejecución.

Para enviar los datos al canal de ThingSpeak, se emplea la función **mqttpublish ()**, a la que se envían como argumentos el identificador del canal, la clave correspondiente de escritura de dicho canal y el vector de datos leídos por el sensor.

Enviamos los datos en una variable denominada *payload* que contendrá la información útil que se va a enviar.

Antes de enviar los datos, se define el "tema" en el que se desean publicar los datos, este se corresponde con los campos del canal de ThingSpeak al que se desean enviar los datos de temperatura y humedad leídos por el sensor.

La función **mqttpublish ()** para enviar dichos datos, utiliza la función mqttClient.publish() de la librería *PubSubClient.h,* a la que se envían como argumentos el "tema" en el que publicar los datos y la variable *payload* que contiene dichos datos de temperatura y humedad.

Si el cliente no se ha desconectado, se volvería a realizar la lectura de los datos del sensor y se enviarían a thingspeak de nuevo. En caso de que el cliente se desconecte, se volvería a realizar la conexión y la subscripción al canal del que se realiza la lectura de datos enviados por la plataforma ThingSpeak.

Por último, para leer los datos del canal en el que se ha subscrito el cliente se utilizan las siguientes funciones:

mqttClient.setServer(server, 1883); → Con esta función se configuran los detalles del servidor de ThingSpeak, como indica la documentación de la plataforma, debe configurarse en el puerto 1883.

mqttClient.setCallback(myMessageArrived); → Esta funciona como manejadora de eventos y se ejecuta cada vez que detecta que ha habido un cambio en el canal del que se realiza la lectura. Como argumentos se le envía la función *myMessageArrived (),* que es la encargada de leer el dato del canal enviado desde la plataforma ThingSpeak. Si se recibe un 1 se encenderá el Led y si se recibe un 0 se apagará.

A diferencia del primer método para conectarse con la plataforma, en el que se leía del canal cada vez que se repetía el bucle, aquí solamente se realiza la lectura en caso de que se produzca un cambio o evento en dicho canal, la función mqttClient.setCallback(myMessageArrived) lo detectará y se actuará sobre el Led.

La repetición del bucle en este caso se realiza cada 14 segundos, ya que la plataforma es capaz de leer datos cada 15 segundos como mínimo.

Uno de los inconvenientes de este método de conexión es que en algunas ocasiones la conexión del cliente con la plataforma se pierde, y durante el tiempo en el que vuelve a realizar la conexión puede haber un número de datos leídos por el sensor que no lleguen a enviarse a la plataforma o incluso que los datos enviados desde la plataforma tampoco lleguen al dispositivo, y en consecuencia no se pueda actuar sobre el Led.

Sin embargo, al enviar los datos con esta frecuencia de 14 segundos, se ha observado que la desconexión entre el cliente y ThingSpeak no ocurre un número de veces tan significativo como para inutilizar este método de comunicación.

En definitiva, ambos métodos de conexión funcionan de forma correcta, el primero es más simple y sencillo, el intercambio de información con la plataforma funciona siempre de manera correcta.

Sin embargo, el código publish and subscribe resulta más versátil debido a que podría emplearse con otras plataformas IoT realizando algunos cambios, algo que no podría hacerse con el primer método.

7.4. Implementación de la plataforma IoT

Como ya se ha explicado anteriormente, la plataforma IoT con la que se va a trabajado es ThingSpeak.

Lo primero que se debe hacer es crear una cuenta en ThingSpeak utilizando un correo electrónico, y después, se deben configuran los diferentes canales que vayan a utilizarse. Una vez se ha realizado esta tarea, se puede continuar trabajando con la plataforma IoT para visualizar los datos, añadir indicadores, crear código Matlab para procesar los datos y enviar tweets o correos electrónicos de alama.

A continuación, se va a explicar cada parte de la plataforma IoT con la que se ha trabajado.

Creación y configuración de los canales

Se han creado tres canales:

- El canal “DHT22”, en el que se recibirán los datos de temperatura y humedad leídos por el sensor.
- El canal “Encendido/Apagado”, que almacenará el valor 0 o 1 dependiendo de si se ha decidido encender o apagar el Led.
- El canal “Control Termostato”, en el que se reciben los datos desde la aplicación móvil, para decidir si encender o apagar el LED.

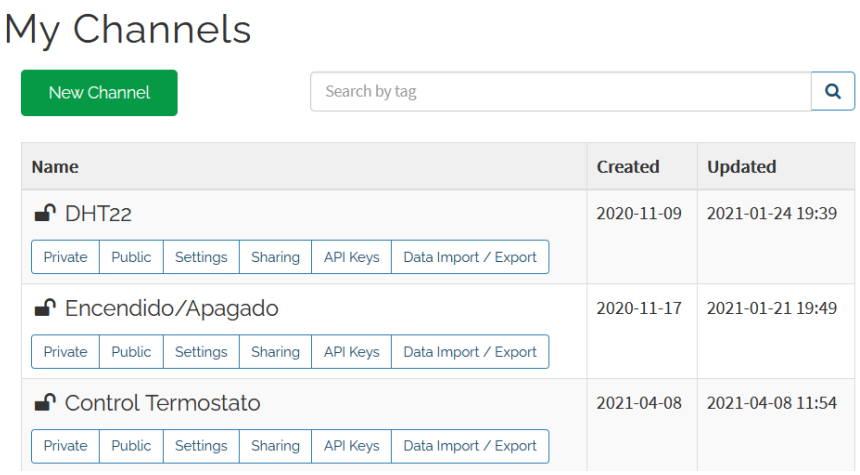


Ilustración 52.- Creación de canales en ThingSpeak

El canal "DHT22" es en el que se van a recibir los datos de temperatura y humedad en los campos 1 y 2 (Field 1 y Field 2) respectivamente desde el dispositivo IoT, la configuración se muestra en la siguiente imagen.

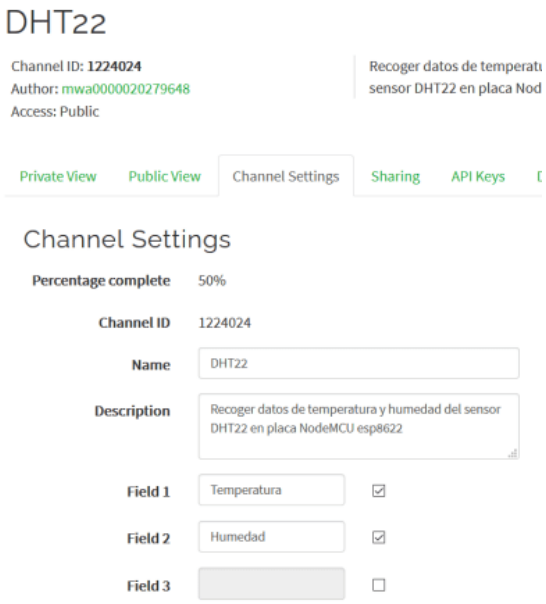


Ilustración 53.- Configuración del canal "DHT22" en ThingSpeak

El canal "Encendido/Apagado" se ha configurado de tal manera que solo se utilice el campo 1 (Field 1), que será el que almacene la información que determine si se enciende o apaga el Led, ya sea desde la propia plataforma IoT o desde la aplicación móvil. Por lo que este campo solamente tendrá el valor de 0 (apagado) o 1 (encendido). La siguiente imagen muestra la configuración de este canal.

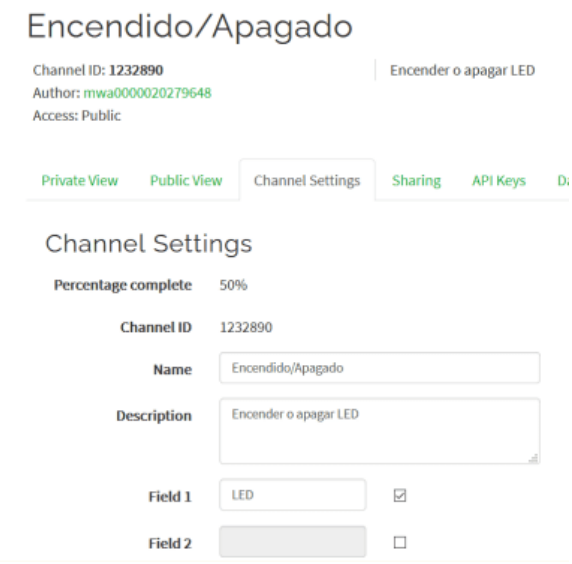


Ilustración 54.- Configuración del canal "Encendido/Apagado" de ThingSpeak

Por último, el canal “Control Termostato” utiliza el primer campo (Field 1) para almacenar los datos enviados desde la aplicación móvil cuando el usuario pulsa los botones “ENCENDER” (se recibe el valor 50) y “APAGAR” (recibe el valor 0). En el segundo campo (Field 2), se almacena el valor de temperatura del termostato que el usuario ha establecido desde la aplicación móvil. La siguiente imagen muestra la configuración del canal.

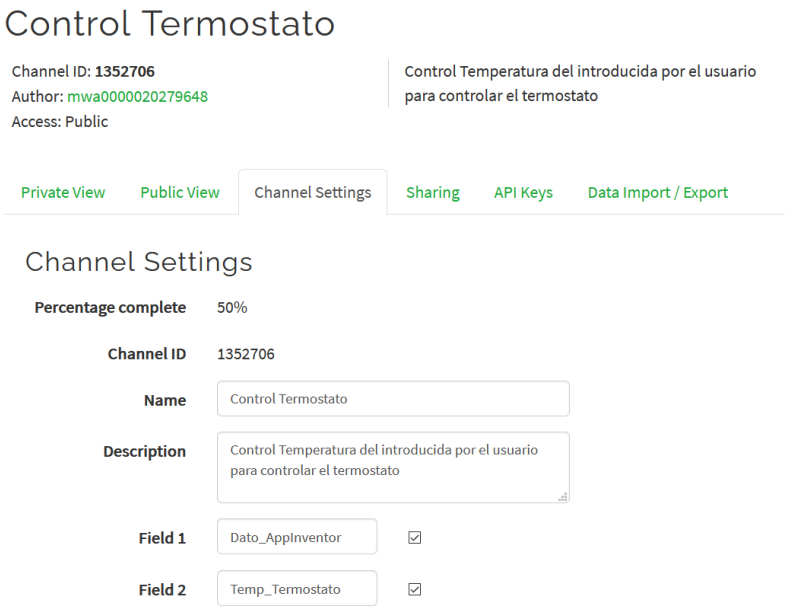


Ilustración 55.- Configuración del canal "Control Termostato" de ThingSpeak

Claves proporcionadas por la plataforma ThingSpeak

Una vez creada la cuenta y los canales en ThingSpeak, la propia plataforma genera de manera automática los identificadores (Channel ID) y claves (API Keys) de cada canal, estos datos son los que se emplean en el código de Arduino para realizar la conexión con la plataforma

.

En la siguiente imagen se pueden ver tanto el identificador del canal “DHT22” como las claves de lectura (Read API Keys) y escritura (Write API Keys) que permitirán leer o escribir datos en el canal.

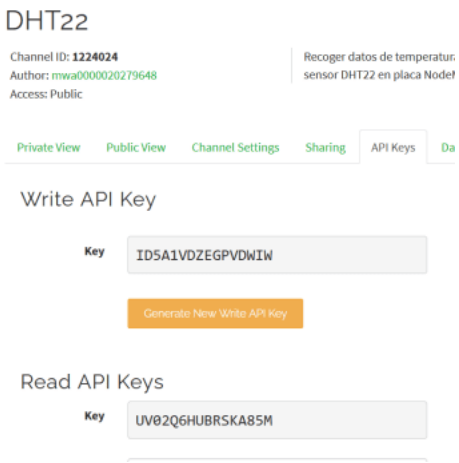


Ilustración 56.- API Keys del canal "DHT22" de ThingSpeak

Para el canal “Encendido/Apagado” el identificador y las claves se generan de la misma manera, como se puede observar en la siguiente imagen.

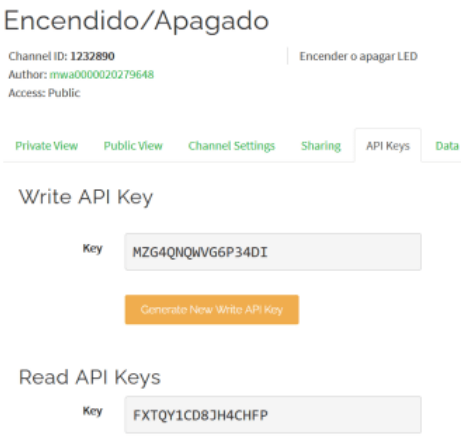


Ilustración 57.- API Keys del canal "Encendido/Apagado" de ThingSpeak

Lo mismo ocurre para el tercer canal “Control Termostato”.

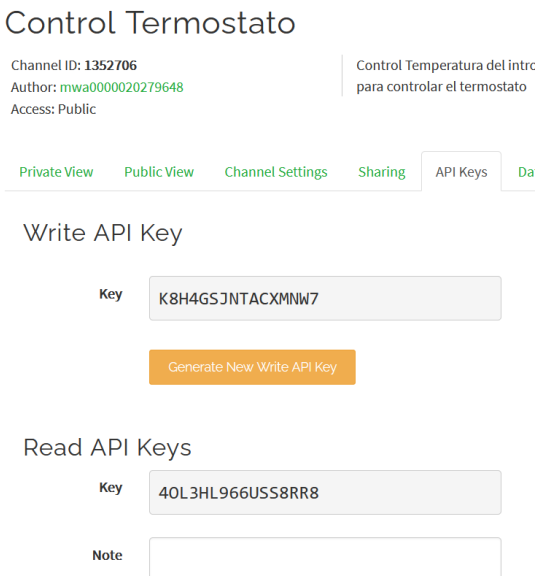


Ilustración 58.- API Keys del canal "Control Termostato" de ThingSpeak

También se crean tres claves de forma automática, a las cuales es posible acceder desde el apartado *My Profile*.

Estas son útiles para realizar la conexión entre el dispositivo IoT y la plataforma a partir del protocolo de comunicación MQTT y poder subscribirse a los canales, como es el caso de “MQTT API Key”, o “Alerts API Key” para enviar corres de alerta, como se verá más adelante en este apartado.

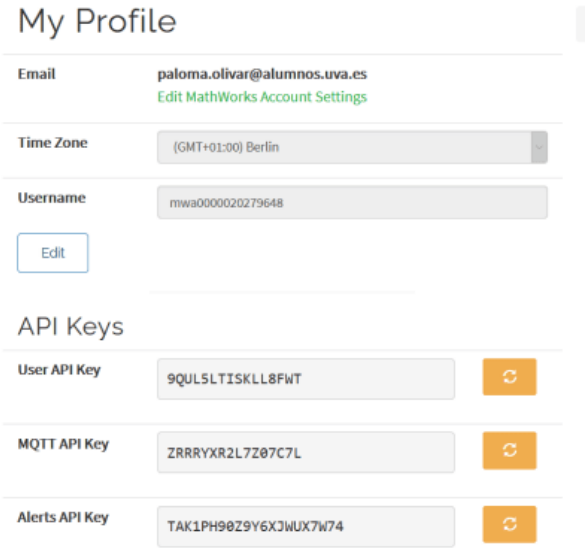


Ilustración 59.- Otras API Keys importantes de ThingSpeak

Visualización de los datos de cada canal

Para visualizar los datos de temperatura y humedad enviados desde el NodeMCU a ThingSpeak, existen dos formas diferentes, privada o pública. En este caso, se ha optado por la vista pública, para que los datos sean de mayor accesibilidad.

En el canal “DHT22” se han añadido dos visualizadores utilizando la pestaña “Add Visualizations”. En el primero de ellos, se representarán los últimos 10 datos de temperatura recibido en el canal, y en el segundo los últimos 10 datos de humedad.

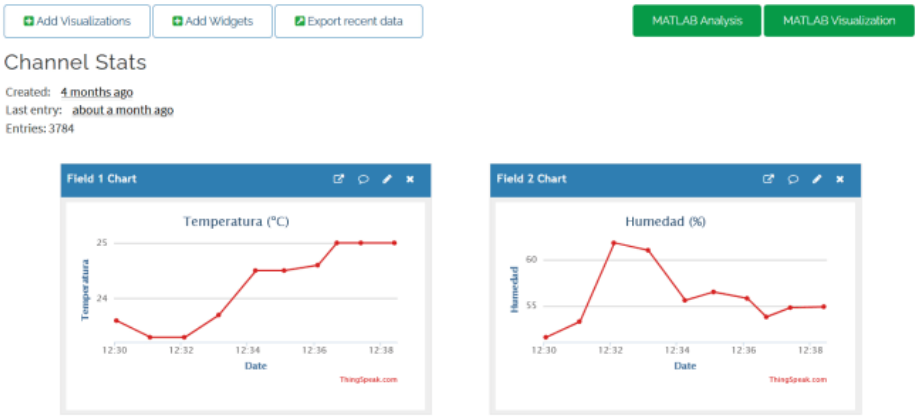


Ilustración 60.- Gráficos canal "DHT22" de ThingSpeak

Se han incluido dos displays utilizando la pestaña “Add Widgets”, en los que se representan los datos tanto de temperatura como de humedad recibidos en el canal en cada instante.

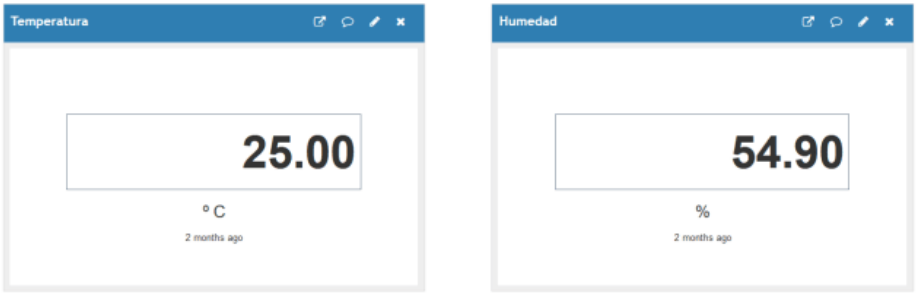


Ilustración 61.- Displays e indicador del canal "DHT22" de ThingSpeak

En el canal “Encendido/Apagado” se ha añadido un visualizador en forma de gráfica de igual manera que el primer canal, en la que se representan los último 10 datos recibidos. También se ha incluido un indicador de color azul, que servirá de ayuda para saber cuándo el Led está encendido o apagado, ya que se ilumina a la vez que este.

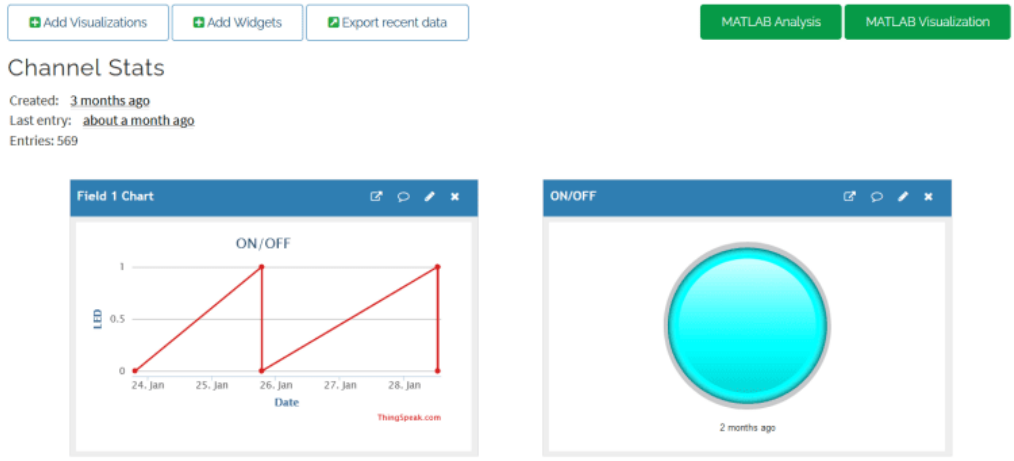


Ilustración 62.- Gráfico e indicador del canal "Encendido/Apagado" de ThingSpeak

En el canal “Control Termostato” se han añadido dos displays, en el primero se indica el valor que ha llegado al campo 1 de este canal desde la aplicación móvil. Si el usuario ha presionado el botón “ENCENDER”, se recibe el valor 50. En caso de que haya presionado el botón “APAGAR”, este campo tendrá el valor de 0.

En el segundo display se muestra la temperatura a la que está el termostato, este valor también lo configura el usuario desde la aplicación móvil, y el dato se envía pulsando el botón “OK”.

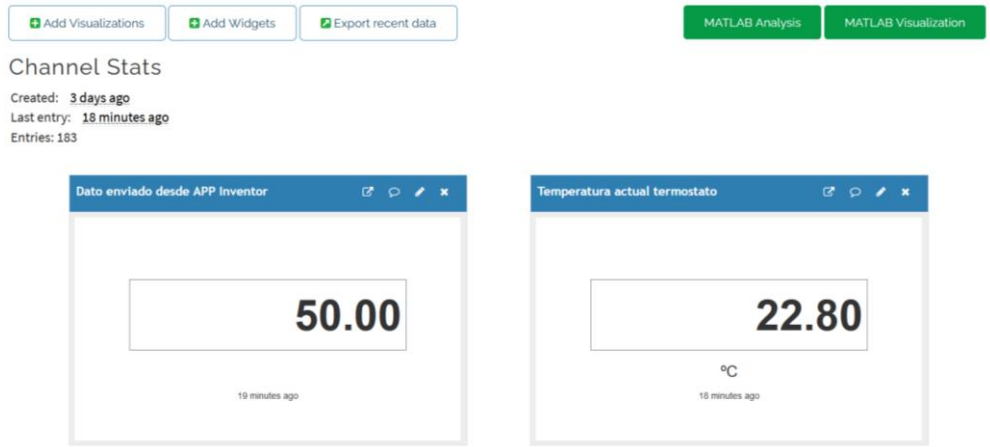


Ilustración 63.- Gráfico y display del canal "Control Termostato" de ThingSpeak

Encender/Apagar LED automáticamente

Desde la plataforma IoT es posible encender o apagar el LED de forma automática cuando la temperatura disminuye de cierto valor predeterminado, para ello se ha implementado el siguiente código con la herramienta MATLAB Analysis de la plataforma.

ThingSpeak™ Channels Apps Support

MATLAB Code

```
% Template MATLAB code for reading data from a private channel, analyzing
% the data and storing the analyzed data in another channel.

% TODO - Replace the [] with channel ID to read data from:
readChannelID = [1224024];

% TODO - Enter the Read API Key between the '' below:
readAPIKey = 'UV02Q6HUBRSKA85M';

% TODO - Replace the [] with channel ID to write data to:
writeChannelID = [1232890];

% TODO - Enter the Write API Key between the '' below:
writeAPIKey = 'MZG4QNQWVG6P34DI';

%% Read Data %%
data = thingSpeakRead(readChannelID,'Fields', [1], 'ReadKey', readAPIKey);

%% Analyze Data %%
% Add code in this section to analyze data and store the result in the

if data < 25

    analyzedData = 1;

else

    analyzedData = 0;

end

thingSpeakWrite(writeChannelID, analyzedData, 'WriteKey', writeAPIKey);
```

Ilustración 64.- Código Matlab para enderder o apagar el Led

Lo que este código hace es leer los datos que llegan a la plataforma en el campo 1 del canal “DHT22”, es decir, los datos de Temperatura obtenidos por el sensor.

Este valor de temperatura se almacena en la variable “data”, y con una sentencia *if else*, se asigna el valor 1 a la variable “analyzedData” cuando la temperatura leída es menor de 25 ºC. En caso contrario, esta variable tendrá el valor de 0.

A continuación, se escribe este valor de la variable “analyzedData” en el campo 1 del canal “Entendido/Apagado”, con lo que el Led se encenderá o apagará de acuerdo con el dato enviado.

Para ejecutar este código se va a utilizar la herramienta de ThingSpeak denominada *REACT*, que permite realizar acciones cuando los datos del canal cumplen una determinada condición.

En este caso, se va a configurar para que el código se ejecute siempre que una lectura del sensor llega a la plataforma, para ello la condición será que la Temperatura sea siempre positiva, como se observa en la siguiente imagen.

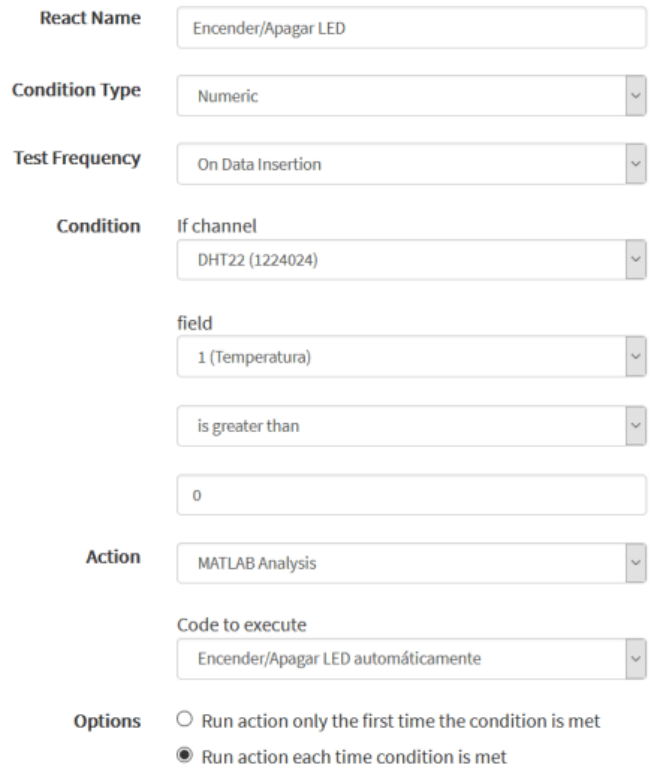


Ilustración 65.- Configuración herramienta React para encender/apagar el Led

Control temperatura termostato

Se ha implementado otro código con MATLAB Analysis para encender o apagar el LED según la temperatura a la que haya decidido el usuario poner el termostato desde la aplicación móvil (Set Point), o si ha pulsado los botones "ENCENDER" o "APAGAR".

Consiste en leer el dato de temperatura del termostato enviado desde la aplicación y compararlo con el dato actual de temperatura que se envía desde el sensor. Si la temperatura actual es menor que el *Set Point* establecido, el LED se enciende, si el dato actual es mayor que dicho *Set Point*, el LED se apaga. Para encender o apagar el led se escribe un 1 o un 0 en el canal "Encendido/Apagado".

Este sistema permite acercar más la funcionalidad a una aplicación real, en la que el usuario decide a qué temperatura quiere tener su hogar, y la caldera se enciende o apaga en función de dicha temperatura.

Además, se ha tenido en cuenta la curva de histéresis que suelen tener los termostatos, con el fin de evitar las oscilaciones, o encendidos y apagados constantes, lo que también se traduce en un ahorro energético. Se ha optado por una histéresis de 0.2 ºC, ya que en el mercado hay termostatos bastante precisos en cuanto a histéresis se refiere.

También se ha tenido en cuenta que, si el usuario pulsa los botones "ENCENDER" o "APAGAR" desde la aplicación móvil, se actúe sobre el LED en consecuencia.

Para ejecutar este código se utiliza la misma configuración de *REACT* que para el código "Encender/Apagar Led automáticamente" que se ha explicado anteriormente, ya que ThingSpeak no permite generar dos Reacts con los mismos parámetros, y dado que no se van a ejecutar los dos códigos a la vez, será necesario decidir cuál de ellos se va a configurar con este REACT.

En la siguiente imagen se puede observar el código desarrollado:

MATLAB Code

```
% Enter your MATLAB Code below
% ENCENDER Y APAGAR LED SEGÚN LA TEMPERATURA DEL TERMOSTATO INTRODUCIDA POR EL USUARIO

% TODO - Replace the [] with channel ID to read data from:
readChannelID = [1224024];

% TODO - Enter the Read API Key between the '' below:
readAPIKey = 'UV02Q6HUBRSKA85M';

% TODO - Replace the [] with channel ID to write data to:
writeChannelID = [1232890];

% TODO - Enter the Write API Key between the '' below:
writeAPIKey = 'MZG4QNQWVG6P34DI';

%% Read Data %%
data = thingSpeakRead(readChannelID,'Fields', [1], 'ReadKey', readAPIKey);

readChannelID_t = [1352706]; % ID canal del que recibo la temperatura del termostato

readAPIKey_t = '4OL3HL966USS8RR8'; % Claves de lectura y escritura del canal

writeAPIKey_t = 'K8H4GSJNTACXMNW7';

% Leo el campo 1 del canal (Encender o Apagar Led con botones desde aplicación)
datoAplicacion = thingSpeakRead(readChannelID_t,'Fields', [1], 'ReadKey', readAPIKey_t);

% Leo el campo  del canal (Temperatura del termostato desde aplicación)
temperatura_t = thingSpeakRead(readChannelID_t,'Fields', [2], 'ReadKey', readAPIKey_t);

% Enciendo el LED si la temperatura actual es menor que la temperatura
% a la que se ha puesto el termostato.
% Se tiene en cuenta una posible histéresis de 0.2ºC.
% Si se pulsan el boton "ENCENDER" en la aplicacion envia un 50 al canal.
% Si se pulsan el boton "APAGAR" en la aplicacion envia un 0 al canal.

if  data <= (temperatura_t - 0.2) || datoAplicacion == 50 && datoAplicacion ~= 0

     analyzed = 1;

     thingSpeakWrite(writeChannelID, analyzed, 'WriteKey', writeAPIKey);

elseif data >= (temperatura_t + 0.2) || datoAplicacion == 0 && datoAplicacion ~= 50

    analyzed = 0;

    thingSpeakWrite(writeChannelID, analyzed, 'WriteKey', writeAPIKey);

end
```

Ilustración 66.- Código Matlab para el control de temperatura del termostato.

Enviar Email

Para enviar un Email de alerta, se ha implementado otro código con la herramienta MATLAB Analysis.

MATLAB Code

```
% Read the soil moisture channel data from the past two weeks.
% Send an email and tell the user to add water if the value
% is in the lowest 10 %.

% Store the channel ID for the moisture sensor channel.
readChannelID = [1224024];
readAPIKey = 'UV02Q6HUBRSKA85M';

% Provide the ThingSpeak alerts API key.  All alerts API keys start with TAK.
alertApiKey = 'TAK1PH90Z9Y6XJWUX7W74';

% Set the address for the HTTTP call
alertUrl="https://api.thingspeak.com/alerts/send";

% webwrite uses weboptions to add required headers.  Alerts needs a ThingSpeak-Alerts-API-Key h
options = weboptions("HeaderFields", ["ThingSpeak-Alerts-API-Key", alertApiKey ]);

% Set the email subject.
alertSubject = sprintf("Alarma de Temperatura");

% Read the recent data.
temperatureData = thingSpeakRead(readChannelID,'Fields', [1], 'ReadKey', readAPIKey);

% Write the email body
if (temperatureData<25)

    %alertBody = ' La Temperatura esta por debajo de 25 ºC ';
    alertBody = sprintf("La Temperatura actual es %0.2f ºC < 25 ºC", temperatureData)

    % Catch errors so the MATLAB code does not disable a TimeControl if it fails
    try
        webwrite(alertUrl , "body", alertBody, "subject", alertSubject, options);

    catch someException
        fprintf("Failed to send alert: %s\n", someException.message);

    end

end
```

Ilustración 67.- Código Matlab para enviar EMAIL

En este caso, se realiza la lectura del campo 1 del canal “DHT22”, es decir, la Temperatura leída por el sensor y enviada a la plataforma. Este valor se almacena en la variable “TemperatureData”.

Cuando este dato es menor de 25 ºC se escribe el cuerpo del mensaje que se va a enviar al correo electrónico, y se almacena en la variable “alertBody”.

Utilizamos una sentencia *try catch* para detectar posibles errores al enviar el mensaje, en caso de que no exista ningún error, se enviará el mensaje utilizando el URL (AlertUrl) y la clave de alertas (alertApikey), proporcionadas por la plataforma IoT, ya que permiten acceder a la página de correo electrónico.

Para ejecutar este código se va a emplear otra herramienta de la plataforma denominada *TimeControl*. Esta permite realizar acciones en un momento específico o en un horario regular. Se ha configurado para que se ejecute cada 10 minutos, y que, en ese caso, compruebe la temperatura actual y envíe un Email si la temperatura es menos de 25 ºC. Este Email se envía al mismo correo electrónico con el que se ha creado la cuenta de ThingSpeak.

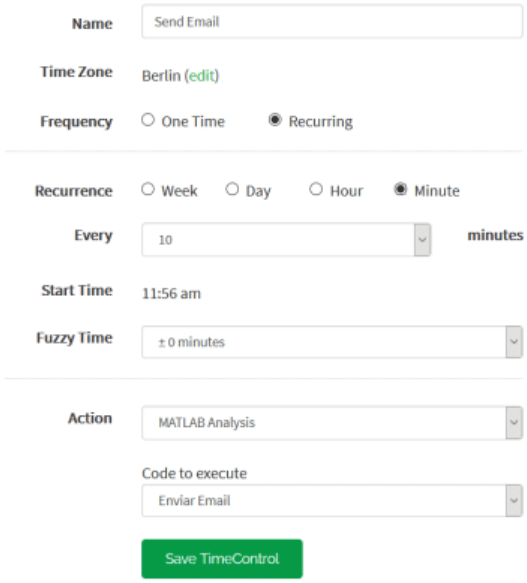


Ilustración 68.-Configuración herramienta TimeControl para enviar EMAIL

La siguiente imagen muestra el Email de alerta que se recibe cuando la condición se cumple.

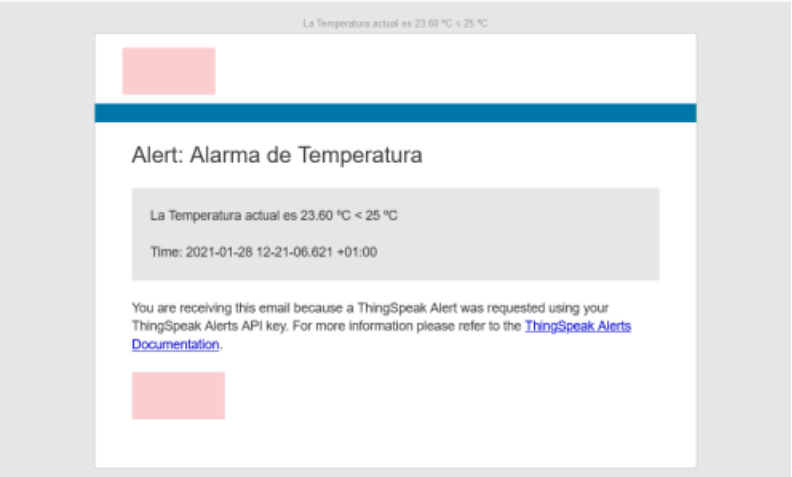


Ilustración 69.- EMAIL de alerta recibido al correo electrónico

Envío de Tweet

Para enviar un Tweet se emplea la herramienta *ThingTweet*, y se debe asociar la cuenta de Twitter a la que se desea enviar la información.

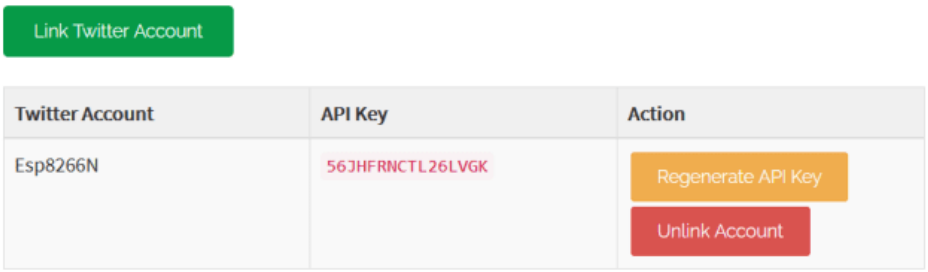


Ilustración 70.- Configuración de la cuenta de Twitter en ThingSpeak

A continuación, se configura otro *REACT* para que esta acción se ejecute cada 10 minutos y compruebe si la temperatura es menor de 25 ° C, en caso de que lo sea, se enviará un Tweet a la cuenta asociada con el mensaje que se haya escrito.

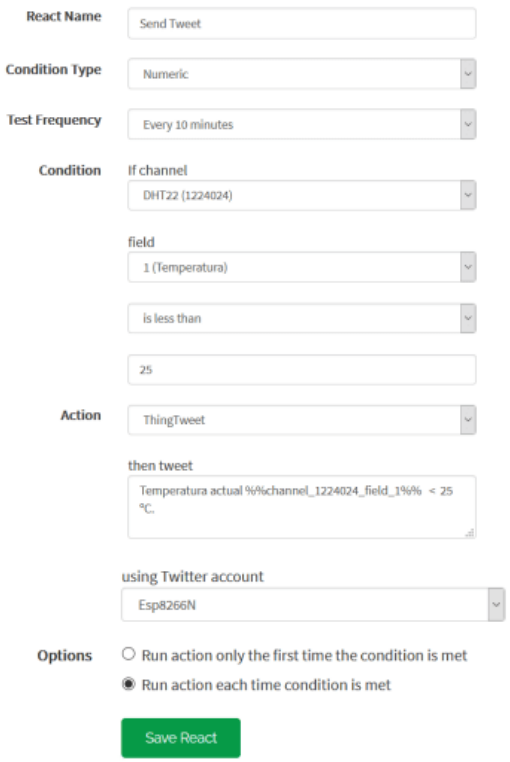


Ilustración 71.- Configuración herramienta React para el envÍ de Tweets

Para enviar el dato leído por el sensor dentro del cuerpo del mensaje, hay que escribir la variable en el siguiente formato:

%%Channel_ID_field_ID%%

La siguiente imagen muestra los tweets recibidos en la cuenta de Twitter cuando la condición se cumple.

Ilustración 72.- Tweets recibidos en la cuenta de Twitter

Importar o Exportar datos

Además, la plataforma también permite descargar o subir datos, por ejemplo, se podrían descargar todos los datos que han sido enviado a la plataforma en el formato CSV como se indica en la siguiente imagen, y posteriormente visualizarlos en Excel.

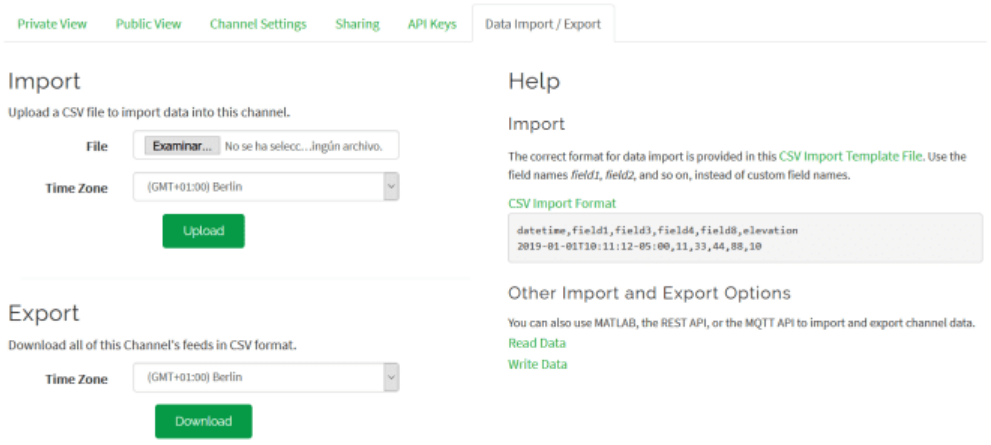
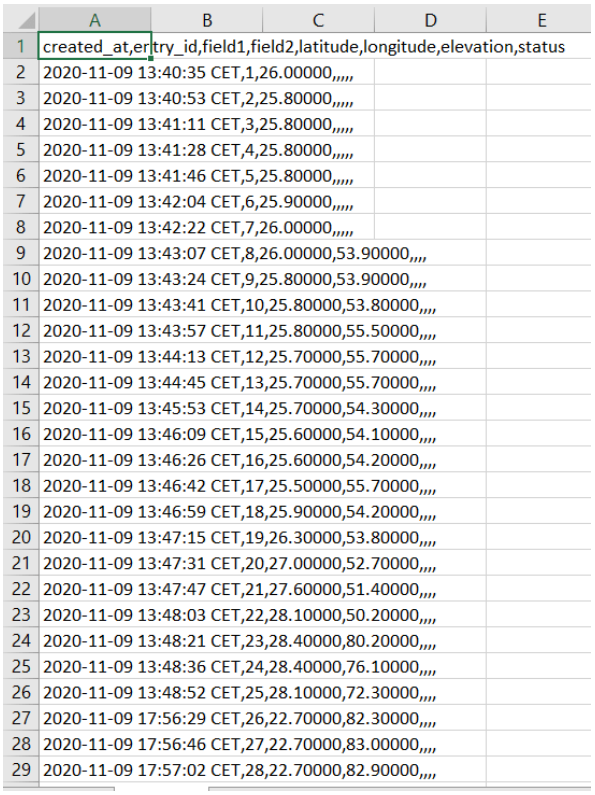


Ilustración 73.- Importar y exportar datos en ThingSpeak



	A	B	C	D	E
1	created_at,entry_id,field1,field2,latitude,longitude,elevation,status				
2	2020-11-09 13:40:35 CET,1,26.00000,,,,,				
3	2020-11-09 13:40:53 CET,2,25.80000,,,,,				
4	2020-11-09 13:41:11 CET,3,25.80000,,,,,				
5	2020-11-09 13:41:28 CET,4,25.80000,,,,,				
6	2020-11-09 13:41:46 CET,5,25.80000,,,,,				
7	2020-11-09 13:42:04 CET,6,25.90000,,,,,				
8	2020-11-09 13:42:22 CET,7,26.00000,,,,,				
9	2020-11-09 13:43:07 CET,8,26.00000,53.90000,,,,				
10	2020-11-09 13:43:24 CET,9,25.80000,53.90000,,,,				
11	2020-11-09 13:43:41 CET,10,25.80000,53.80000,,,,				
12	2020-11-09 13:43:57 CET,11,25.80000,55.50000,,,,				
13	2020-11-09 13:44:13 CET,12,25.70000,55.70000,,,,				
14	2020-11-09 13:44:45 CET,13,25.70000,55.70000,,,,				
15	2020-11-09 13:45:53 CET,14,25.70000,54.30000,,,,				
16	2020-11-09 13:46:09 CET,15,25.60000,54.10000,,,,				
17	2020-11-09 13:46:26 CET,16,25.60000,54.20000,,,,				
18	2020-11-09 13:46:42 CET,17,25.50000,55.70000,,,,				
19	2020-11-09 13:46:59 CET,18,25.90000,54.20000,,,,				
20	2020-11-09 13:47:15 CET,19,26.30000,53.80000,,,,				
21	2020-11-09 13:47:31 CET,20,27.00000,52.70000,,,,				
22	2020-11-09 13:47:47 CET,21,27.60000,51.40000,,,,				
23	2020-11-09 13:48:03 CET,22,28.10000,50.20000,,,,				
24	2020-11-09 13:48:21 CET,23,28.40000,80.20000,,,,				
25	2020-11-09 13:48:36 CET,24,28.40000,76.10000,,,,				
26	2020-11-09 13:48:52 CET,25,28.10000,72.30000,,,,				
27	2020-11-09 17:56:29 CET,26,22.70000,82.30000,,,,				
28	2020-11-09 17:56:46 CET,27,22.70000,83.00000,,,,				
29	2020-11-09 17:57:02 CET,28,22.70000,82.90000,,,,				

Ilustración 74.- visualización de datos en EXCEL descargados desde ThingSpeak

7.5. Desarrollo de la aplicación móvil

En este subapartado se va a explicar cómo se ha desarrollado la aplicación Android utilizando el entorno MIT App Inventor.

El objetivo de crear una aplicación móvil es poder observar los datos de temperatura y humedad leídos por el sensor y actuar sobre el Led, encendiéndolo o apagándolo de forma remota.

Como ya se ha explicado anteriormente, el objetivo de utilizar un Led es demostrar que, en un caso real, desde la aplicación móvil se podría actuar sobre la caldera de un hogar pudiendo encenderla o apagarla, y con ello, controlar la temperatura de la casa.

La aplicación Android que se ha creado tiene 2 pantallas o screens, a continuación, se va a explicar en qué consiste cada una de ellas y cómo se han construido.

Screen 1

La primera pantalla consiste en dos displays que muestran la temperatura y la humedad actual, y un indicador azul que permite saber si el LED está encendido o apagado en cada momento.

En la parte inferior hay tres botones, dos de ellos sirven para el encendido y apagado del LED, y un tercer botón denominado “VER GRAFICAS” que abre la segunda pantalla, en la que se muestras los datos en modo de gráficas pudiendo visualizar las últimas 10 lecturas del sensor.

Se ha añadido la posibilidad de que el usuario pueda modificar la temperatura del termostato, con el fin de acercar más la aplicación a una situación real, ya que lo que interesa es aumentar la temperatura de la estancia.

El usuario decide la temperatura a la que quiere poner el termostato y pulsando el botón “OK” envía el valor a ThingSpeak, donde se realiza el procesamiento para decidir si encender o apagar el LED.

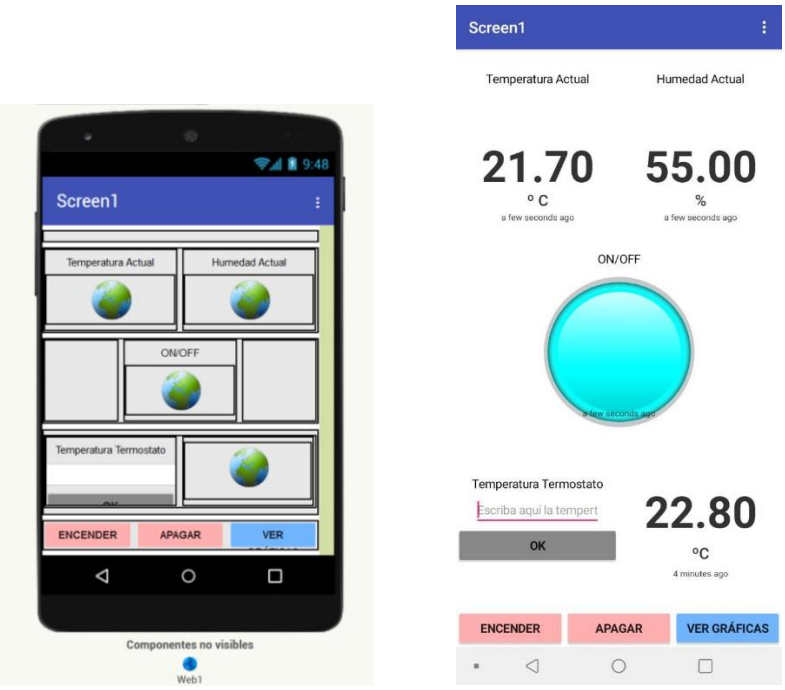


Ilustración 75.- Creación pantalla 1 de la aplicación Android

Para visualizar los displays, el indicador de LED encendido o apagado, y la temperatura actual a la que se ha puesto el termostato, se han añadido 4 *visorWebs*. Estos son componentes que incluye MIT App Inventor y que permiten visualizar páginas webs. El código de bloques correspondiente es el que se muestra en la siguiente imagen, en la que se puede ver que a cada *VisorWeb* se le ha añadido el Url de los gráficos correspondientes de la plataforma ThingSpeak. De esta manera, se puede ver exactamente lo mismo desde la plataforma y desde la aplicación.

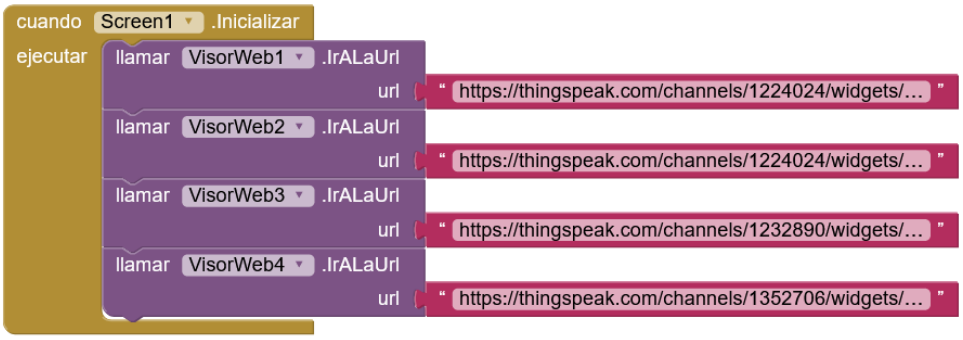


Ilustración 76.- Código de bloques para visualizar los datos en la pantalla 1

Para el encendido y apagado del Led se han añadido dos botones denominados "ENCENDER" y "APAGAR" respectivamente. En el código de bloques correspondiente se observa que dependiendo del botón que se pulse se enviará un 50 o un 0 al campo 1 del canal "Control Termostato" de la plataforma ThingSpeak.

Para realizar esta acción es necesario añadir un componente *Web* (Web 1). Este es un componente no visible que permite gestionar solicitudes HTTP.

Se realiza una llamada a este Web 1, al que se le ha asignado el Url del canal de ThingSpeak en el que se desea escribir, con el campo o Field correspondiente.

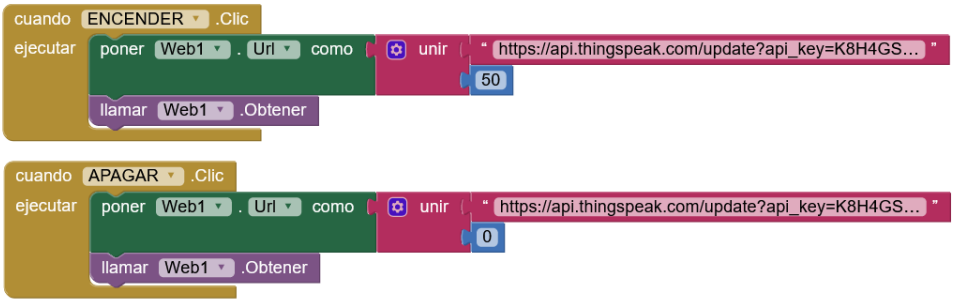


Ilustración 77.- Código de bloques para configurar botones ENCENDER y APAGAR

El botón “VER GRÁFICAS” tiene como función abrir la segunda pantalla o screen 2, en la que se podrán observar los datos en forma de gráfica en el tiempo. En la siguiente imagen se observa el bloque correspondiente.

Ilustración 78.- Código de bloques para configurar botón VER GRÁFICAS

Para introducir el dato de temperatura del termostato por pantalla y enviarlo a ThingSpeak, se utiliza un *CampoDeTexto*, y un botón llamado “OK”.

El dato introducido se envía a la plataforma al pulsar el botón y haciendo una llamada a Web 1, al que se le ha asignado el Url del canal de ThingSpeak en el que se debe escribir este valor.

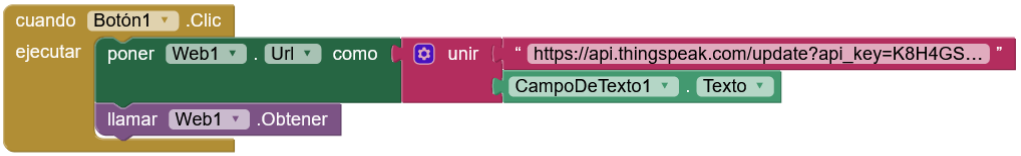


Ilustración 79.- Código de bloques para enviar Temperatura a ThingSpeak

Screen 2

Una vez se accede a la pantalla 2, se podrán observar los últimos diez datos de temperatura y humedad leídos por el sensor, en forma de gráfica en el tiempo.

El botón denominado “ATRÁS” permite regresar a la pantalla 1 de nuevo.

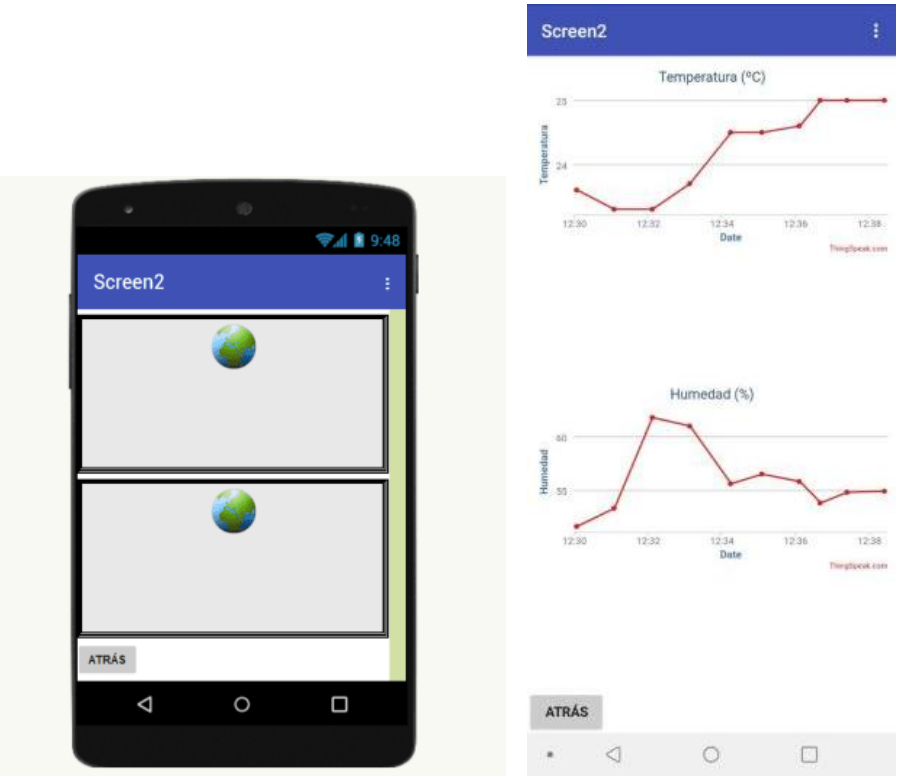


Ilustración 80.- Creación pantalla 2 de la aplicación Android

Para visualizar las dos gráficas, se han añadido otros 2 *VisorWebs*. A cada uno de estos se le ha asignado el Url de los gráficos correspondientes en ThingSpeak. De esta manera, podrán observarse las mismas gráficas en ambos lados, como ocurría en la pantalla 1.

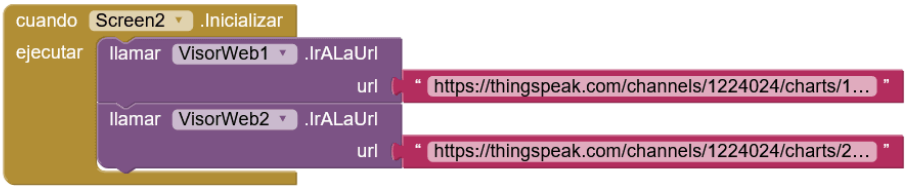


Ilustración 81.- Código de bloques para visualizar los datos en la pantalla 2

Al pulsar el botón "Atrás" se cierra esta pantalla y vuelve a dirigirse a la primera, como se observa en el siguiente bloque de código.

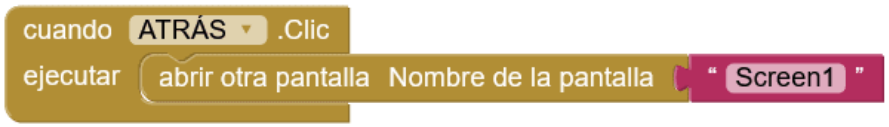


Ilustración 82.- Código de bloques para configurar botón ATRÁS

Una vez se han configurado todos los bloques y se ha finalizado la creación de la aplicación, MIT App Invento ofrece la opción de generar un código QR para instalar el archivo .apk directamente en el móvil. También ofrece una segunda opción, que consiste en guardar el archivo .apk en el ordenador, para posteriormente instalarlo en el móvil.

Una vez se ha instalado la aplicación en el teléfono móvil, es posible acceder a ella, ver los datos y actuar sobre el Led en cualquier momento y desde cualquier lugar.

Ilustración 83.- Código QR para instalar aplicación Android

8. Conclusiones

En este proyecto se ha llevado a cabo el diseño y la implementación de un medidor de temperatura y humedad para el hogar basado en la tecnología IoT. Para ello, ha sido necesario realizar un estudio previo del concepto IoT, su relación con la domótica y cómo se presenta en el mercado actualmente.

Se ha creado un prototipo de medidor de temperatura y humedad basado en la placa de desarrollo NodeMCU V3, la cual incluye el módulo ESP8266 que permite conexión Wifi, y el sensor de temperatura y humedad DHT22.

Para visualizar los datos de temperatura y humedad leídos por el sensor se ha utilizado la plataforma IoT ThingSpeak. Para realizar la conexión entre el dispositivo y dicha plataforma se ha desarrollado un código en Arduino IDE que permite realizar una comunicación bidireccional, pudiendo tanto enviar datos desde el dispositivo a la plataforma como recibir datos desde la plataforma al dispositivo.

Además, ThingSpeak incluye algunas herramientas que ha permitido visualizar los datos de temperatura y humedad enviados desde el dispositivo, procesarlos y generar alertas.

En el Hardware se ha instalado un Led cuya función es simular el encendido o apagado de una caldera en un hogar de forma remota, ya que desde la plataforma IoT es posible enviar datos al dispositivo para que realice esta acción y encienda o apague el Led.

Otra forma de visualizar los datos y tener un control sobre el Led es desde una aplicación Android desarrollada con este fin. Desde el entorno de desarrollo MIT App Inventor se ha creado esta aplicación móvil, que cuenta con visualizadores y botones con los que es posible cumplir el objetivo de observar los datos y actuar sobre el Led desde cualquier lugar.

Finalmente, el prototipo desarrollado ha demostrado cumplir con los objetivos planteados en el trabajo, permitiendo medir y visualizar tanto la temperatura como la humedad de un hogar, y pudiendo realizar una acción para controlar la temperatura de forma remota, con lo que se ha conseguido incluir una parte de domótica en el diseño.

9. Líneas futuras de trabajo

En este apartado se van a plantear algunas de las posibles mejoras que podrían desarrollarse a partir de este proyecto.

Uno de los aspectos que podrían mejorar el medidor IoT construido en este TFG sería utilizar un Hardware que ofrezca mayores ventajas, actualmente existen en el mercado dispositivos diseñados exclusivamente para implementar aplicaciones IoT, por lo que ya cuentan con los aspectos necesarios para obtener una aplicación más profesional, aunque el precio es más elevado.

Otra posible mejora en el diseño sería desarrollar una aplicación nativa utilizando un entorno de desarrollo más profesional, esto implica que sería necesario emplear más tiempo en el desarrollo, pero se tendrían unos beneficios considerables, ya que la aplicación ofrecería mayor versatilidad, más posibilidades y un mejor funcionamiento.

El objetivo de utilizar un medidor de temperatura y humedad en el ámbito doméstico es poder tener un control sobre la temperatura y con ello, obtener un ahorro tanto energético como económico.

Partiendo de este punto, otra posible mejora sería añadir un mayor número de dispositivos y sensores que permitan controlar otros elementos del hogar, con el objetivo de obtener un mayor ahorro. Por ejemplo, empleando sensores de detección de movimiento, se podría controlar el encendido y apagado de la iluminación de una casa, lo que conlleva un ahorro energético y económico considerable.

En definitiva, con esto se obtendría un sistema domótico más amplio, lo que implicaría la necesidad de utilizar una plataforma IoT más potente y posiblemente hacer uso de una cuenta de pago.

Por último, sería interesante investigar la compatibilidad con entornos comerciales como Google Home o Amazon, con lo que sería posible realizar acciones sobre los dispositivos del hogar utilizando un comando de voz.

10. Bibliografía

[1] D. Evans, "Internet de las cosas: Cómo la próxima evolución de internet lo cambia todo". CISCO, 2011.
[Último acceso: Marzo, 2021]
https://www.cisco.com/c/dam/global/es_mx/solutions/executive/assets/pdf/internet-of-things-iot-ibsg.pdf

[2] Fracttal. (2018, Octubre 10). Las 9 aplicaciones más importantes del Internet de las Cosas (IoT) [Online].
[Último acceso: Enero, 2021]
https://www.fracttal.com/es/blog/2018/10/10/9-aplicaciones-importantes-iot

[3] J. Salazar, S. Silvestre, "El mundo Internet of Things (IoT)", MoVET, Primera edicion, 2019.
[Último acceso: Diciembre, 2020]
https://upcommons.upc.edu/bitstream/handle/2117/185120/LM01_R_ES.pdf?sequence=1&isAllowed=y

[4] bitnova, La domótica aplicada a la gestión inteligente de RRHH y a la productividad empresarial [Online].
[Último acceso: Enero, 2021]
https://www.bitnova.es/soluciones-tecnologicas/domotica-e-internet-de-las-cosas

[5] Google Store.
[Último acceso: Enero, 2021]
Nest Themostat E:
https://store.google.com/es/product/nest_thermostat_e
Nest Audio (Altavoz):
https://store.google.com/es/product/nest_audio

[6] Amazon Echo (4ª generación).
[Último acceso: Enero, 2021]
https://www.amazon.es/nuevo-echo-4a-generacion-sonido-de-alta-calidad-controlador-de-hogar-digital-integrado-y-alexa-blanco/dp/B085FXGP5W

[7] Wyze Bulb.
[Último acceso: Enero, 2021]
https://wyze.com/wyze-bulb.html

[8] August Wi-Fi Smart Lock.
[Último acceso: Enero, 2021]
https://august.com/products/august-wifi-smart-lock

[9] J.-Tobias Lorenz, S. Spang, M. Strub, S. Johansson. (2019, Junio 24). Tackling the IoT opportunity for commercial lines isurance. McKinsey & Company [Online].
[Último acceso: Febrero, 2021]
https://www.mckinsey.com/industries/financial-services/our-insights/tackling-the-iot-opportunity-for-commercial-lines-insurance

[10] Kaspersky. ¿En qué consiste el Internet de las cosas (IoT)? Seguridad del IoT [Online].
[Último acceso: Febrero, 2021]
https://www.kaspersky.es/resource-center/definitions/what-is-iot

[11] A. Banafa. (2020, Agosto 24). El Internet de las cosas y COVID-19. BBVA [Online].
[Último acceso: Marzo, 2021]
https://www.bbvaopenmind.com/tecnologia/mundo-digital/internet-de-las-cosas-y-covid-19/

[12] Raspberry Pi.
[Último acceso: Enero, 2021]
https://www.raspberrypi.org/

[13] Arduino.
[Último acceso: Enero, 2021]
https://www.arduino.cc/

[14] L. del Valle Hernández. NodeMCU tutorial paso a paso desde cero. Programafacil.com [Online].
[Último acceso: Marzo, 2021]
https://programarfacil.com/podcast/nodemcu-tutorial-paso-a-paso/

[15] Datasheet ESP8266EX. Espressif Systems.
[Último acceso: Febrero, 2021]
https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf

[16] AZ-Delivery. NodeMCU V3.
[Último acceso: Febrero, 2021]
https://www.az-delivery.de/es/products/copy-of-nodemcu-lua-amica-v2-modul-mit-esp8266-12e?_pos=3&_sid=0ad4a551b&_ss=r

[17] Datasheet DHT22. Aosong Electronics Co.,Ltd.
[Último acceso: Febrero, 2021]
https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

[18] Amazon Web Services IoT.
[Último acceso: Febrero, 2021]
https://aws.amazon.com/es/?nc2=h_lg

[19] Microsoft Azure IoT.
[Último acceso: Febrero, 2021]
https://azure.microsoft.com/es-es/overview/iot/

[20] Google Cloud.
[Último acceso: Febrero, 2021]
https://cloud.google.com/?hl=es

[21] Thinger.io.
[Último acceso: Febrero, 2021]
https://thinger.io/

[22] ThingSpeak.
[Último acceso: Marzo, 2021]
https://thingspeak.com/

[23] Android OS.
[Último acceso: Febrero, 2021]
https://androidos.readthedocs.io/en/latest/data/introduccion/

[24] Yeeply. (2018, Octubre 5). ¿Qué son las Aplicaciones Nativas, Web e Híbridas? [Online].
[Último acceso: Febrero, 2021]
https://www.yeeply.com/blog/tipos-de-app-y-para-que-sirven/

[25] Android Studio.
[Último acceso: Febrero, 2021]
https://developer.android.com/studio/intro

[26] Apache Cordova.
[Último acceso: Febrero, 2021]
https://cordova.apache.org/docs/en/latest/

[27] Ionic Framework.
[Último acceso: Febrero, 2021]
https://ionicframework.com/docs

[28] React Native.
[Último acceso: Febrero, 2021]
https://reactnative.dev/

[29] MIT App Inventor.
[Último acceso: Marzo, 2021]
https://appinventor.mit.edu/

[30] Bluetooth.
[Último acceso: Marzo, 2021]
https://www.bluetooth.com/learn-about-bluetooth/bluetooth-technology/

[31] C. A. Orteha Huembes, D. del Socorro Roque, L. E. Ãšbeda Sequeira. (2008, Julio, 28). "Zigbee: El nuevo estándar global para la domótica e inmótica".
[Último acceso: Marzo, 2021]
https://www.monografias.com/trabajos61/zigbee-estandar-domotico-inmotica/zigbee-estandar-domotico-inmotica2.shtml#xdefin

[32] "An Introduction to WiFi", RABBIT, Product Manual 019-0170 • 090409-B, 2007-2008.
[Último acceso: Marzo, 2021]
http://ftp1.digi.com/support/documentation/0190170_b.pdf

[33] L. Llamas. (2019, Febrero 21). Protocolos de comunicación para IoT [Online].
[Último acceso: Marzo, 2021]
https://www.luisllamas.es/protocolos-de-comunicacion-para-iot/

[34] IONOS. (2019, Mayo 03). AMQP: conoce el Advance Message Queuing Protocol [Online].
[Último acceso: Marzo, 2021]
https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/advanced-message-queuing-protocol-amqp/

[35] SMARTBEAR SUPPORT. (2021, Abril 08). AMQP Testing [Online].
[Último acceso: Marzo, 2021]
https://support.smartbear.com/readyapi/docs/testing/amqp.html

[36] MDN Web Docs. (2021, Abril 09). Generalidades del protocolo HTTP [Online].
[Último acceso: Marzo, 2021]
https://developer.mozilla.org/es/docs/Web/HTTP/Overview

[37] M. Yuan. *IBM.* (2017, Octubre 04). Conozca MQTT [Online].
[Último acceso: Marzo, 2021]
https://developer.ibm.com/es/articles/iot-mqtt-why-good-for-iot/

[38] Fritzing. Versión 0.8.7.0.
[Último acceso: Enero, 2021]
https://fritzing.org/

[39] Datasheet TMP36. Analog Devices.
https://www.analog.com/media/en/technical-documentation/data-sheets/TMP35_36_37.pdf

[40] Datasheet DHT11. Mouser Electronics.
https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

[41] Datasheet LM35. Texas Instruments.
https://www.ti.com/lit/ds/symlink/lm35.pdf

[42] Datasheet TC74. Microchip. https://ww1.microchip.com/downloads/en/DeviceDoc/21462D.pdf

[43] M. Barrio Andrés, "INTERNET DE LAS COSAS". 1ª. edicion. Madrid: REUS, 2018. https://www.editorialreus.es/static/pdf/primeraspaginas_9788429020380_internetdelascosas.pdf

[44] GitHub NodeMCU. https://github.com/nodemcu

ANEXO. CÓDIGO FINAL EN ARDUINO IDE

APP_ThingSpeak

```
/* Diseño de un medidor IoT doméstico

   Este código permite realizar una conexión bidireccional entre
el dispositivo NodeMCU V3 y la plataforma IoT ThingSpeak,
   Pudiendo enviar datos de temperatura y humedad obtenidos por el
sensor DHT22,
   y leer datos enviados desde ThingSpeak para encerder o apagar
un LED.

   Para realizar dicha conexión se han implementado 2 metodos:
    1. Tpeak: Utiliza las funciones de la librería ThingSpeak.h
para realizar la conexion y el intercambio de datos.
    2. MQTT: Utiliza el método publicador-subscriptor basado en el
protocolo MQTT para relizar la conexion y el intercambio de datos.

*/

#include <DHT.h> //Libreria de sensores DHT
#include <ESP8266WiFi.h> // Para realizar la conexión a internet
#include <ThingSpeak.h> //Librería plataforma IoT
#include <PubSubClient.h> //Método publicador-subscriptor

#define DHTPIN D5 //Pin al que se conecta el DHT22
#define DHTTYPE DHT22 //Se define el tipo de sensor
#define LED D1 //Pin al que se conecta el LED

//Se definen los dos métodos para realizar la conexión con la
plataforma
#define Tpeak // Utilizando la libreria de la plataforma IoT
ThingSpeak
//#define MQTT //Utilizando el protocolo MQTT


// Credenciales para la conexion wifi
char* ssid = "***********"; // Nombre del WiFi
char* pass = "********"; // Contraseña

// Datos necesarios de Thingspeak para realizar la conexion del
NodeMCU con la plataforma
unsigned long myChannelNumber1 = 1224024; // Thingspeak número de
canal
char* myWriteAPIKey1 = "ID5A1VDZEGPVDWIW"; // ThingSpeak write API
Key
char* myReadAPIKey1 = "UV02Q6HUBRSKA85M"; // ThingSpeak read API
Key

unsigned long myChannelNumber2 = 1232890; // Thingspeak número de
canal
char* myWriteAPIKey2 = "MZG4QNQWVG6P34DI"; // ThingSpeak write API
Key
char* myReadAPIKey2 = "FXTQY1CD8JH4CHFP"; // ThingSpeak read API
Key
```

```
char* MQTTAPIKey = "ZRRRYXR2L7Z07C7L"; //Clave MQTT de thingspeak
char* clientName = "ESP-Thingspeak";

const unsigned int fieldLED = 1; //campo 1 del canal 2 del que
leemos para cambiar el estado del led
const unsigned int fieldt = 1; // campo 1 del canal 1 al que
enviamos la temperatura
const unsigned int fieldh = 2; // campo 2 del canal 1 al que
enviamos la humedad


char* server = "mqtt.thingspeak.com"; // Servidor o Broker de la
plataforma ThingSpeak



DHT dht(DHTPIN, DHTTYPE); // Se crea el objeto DHT

WiFiClient client; // Se define el tipo de cliente

PubSubClient mqttClient( client ); //Se crea el objeto cliente

int status = WL_IDLE_STATUS; // Variable para deterinar el estado
de conexion WiFi


void setup()
{

  Serial.begin(9600); // Velocidad puerto serie

  delay(10);

  dht.begin(); // Inicia el sensor

  pinMode(LED, OUTPUT); // Se configura el pin como salida

  WiFiConnect(); //Se realiza la conexión WiFi


#ifdef Tpeak

  ThingSpeak.begin(client); // Inicia ThingSpeak

#else

  mqttClient.setServer( server, 1883 ); // Se establecen los
detalles del servidor MQTT.

  mqttClient.setCallback(myMessageArrived); // Detecta eventos en
el canal de lectura

#endif

}
```

```
void loop()
{
  float dataSensor[2]; // vector que almacena los valores de
temperatura y humedad


#ifdef Tpeak

  // Se leen temperatura y humedad
  readSensor(dataSensor);

  // Se envían temperatura y humedad a ThingSpeak
  sendingData(myChannelNumber1, myWriteAPIKey1, fieldt, fieldh,
dataSensor);

  // Se lee del canal de ThingSpeak
  readingData(myChannelNumber2, fieldLED, myReadAPIKey2);

  // Se repite el proceso cada minuto
  delay(60000);

#else

  // Si el cliente MQTT no está conectado se realiza la conexion
  if (!mqttClient.connected())
  {
    mqttConnect();

    // Se realiza la subscripcion al canal de ThingSpeak del que
se van a leer los datos
    if (mqttSubscribe(myChannelNumber2, fieldLED, myReadAPIKey2)
== 1 )
    {
      Serial.println( "Subscrito \n" );

    }
  }

  mqttClient.loop(); // Llamada al bucle para mantener la conexión
con el servidor


  // Le leen temperatura y humedad
  readSensor(dataSensor);

  // Se envian datos a ThingSpeak
  mqttPublish(myChannelNumber1, myWriteAPIKey1, dataSensor);

  // Se repite el proceso cada 14 segundos
  delay(14000);

#endif

}
```

```
//Funcion que se encarga de realizar la conexion wifi

void WiFiConnect()
{
  WiFi.begin(ssid, pass); // Inicia WiFi

  Serial.println();
  Serial.print("Conectando a ");
  Serial.print(ssid);

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

  Serial.println();
  Serial.println("Conectado a WiFi");
  Serial.print("Dirección IP: ");
  Serial.println(WiFi.localIP());
  Serial.print("\n");
}

// Funcion encargada de leer la temperatura y la humedad con el
sensor DHT22 y almacenar los datos en el vector data[]

void readSensor(float data[])
{
  float t;
  float h;

  // Se lee temperatura
  t = dht.readTemperature();

  // Se lee humedad
  h = dht.readHumidity();

  // Se comprueba si las lecturas pueden dar algún fallo mediante
la función isnan()
  // Esta función devuelve un 1 en caso de que el valor no sea
numérico
  while (isnan(t) || isnan(h)) {
    Serial.println("Fallo en la lectura del sensor DHT!");
    delay(1000);
    t = dht.readTemperature();
    h = dht.readHumidity();
  }

  // Se asignan los valores de temperatura y humedad al vector
  data[0] = t;
  data[1] = h;

  Serial.print("Temperatura: ");
  Serial.print(t);
  Serial.print(" °C\t ");

  Serial.print("Humedad: ");
  Serial.print(h);
  Serial.print(" %\t");}
```

MQTT_Connection

```
/*
 * Esta función se encarga de manejar los mensajes recibidos del
canal suscrito a través del broker MQTT
 *   topic - Tema de suscripción para mensaje.
 *   messege - Mensaje recibido en formato byte.
 *   length - Longitud del mensaje.
 */

int myMessageArrived(char* topic, byte* messege, unsigned int
length)
{


  Serial.print("Mensaje recibido en tema: ");
  Serial.println(topic);

  Serial.print("Message:");

  char buffer[128];

  // Convierte el mensaje de bytes a int
  //Forma un string a partir de la carga útil
  memcpy(buffer, messege, length);
  buffer[length] = '\0';

  //Lo convierte en entero
  char *end = nullptr;
  long value = strtol(buffer, &end, 10);

  Serial.println(value);

  // Si un mensaje es recibido en el tema correspondiente (en el
que se subscribe), se cambia el estado del LED según el valor de
dicho mensaje.
  if (String(topic) ==
"channels/1232890/subscribe/fields/field1/FXTQY1CD8JH4CHFP") {

    //Si se recibe un 1 el LED se enciende y si se recibe un 0 se
apaga.
    if(value == 1)
    {
      Serial.println("LED ON\n");
      digitalWrite(LED, HIGH);
    }
    else if(value == 0)
    {
      Serial.println("LED OFF\n");
      digitalWrite(LED, LOW);
    }

  }
}
```

```
/*
 * Esta función realiza la conexion con el cliente
 *   clientID - Identificador unico del cliente, en este caso se
crea uno aleatoriamente con la funcion getID.
 *   clientName - Nombre del cliente que se ha establecido.
 *   MQTTAPIKey - Clave obtenida de ThingSpeak para realizar la
conexion MQTT.
 */

void mqttConnect()
{
    char clientID[ 9 ];

    // El bucle se repite hasta que se realice la conexión
    while ( !mqttClient.connected() )
    {

        getID(clientID,8);

        Serial.println( "Intentando conexion MQTT..." );


        if ( mqttClient.connect(clientID, clientName, MQTTAPIKey
))
        {
            Serial.println( "Conectado con ClienID: " + String(
clientID ) + "\t clientName"+ String( clientName ) + "\t
MQTTAPIKey "+String(  MQTTAPIKey ) );

        }
        else
        {
            Serial.print( "La conexión ha fallado, estado = " );
            Serial.print( mqttClient.state() );
            Serial.println( "Reintentar en 5 segundos" );
            delay( 5000 );
        }
    }
}
/*
 * Esta funcion crea un identificador de cliente aleatorio, para
que sea unico
 *   clientID - Matriz de caracteres para la salida
 *   idLength - Longitud clientID
 */

void getID(char clientID[], int idLength)
{
  static const char alphanum[] ="0123456789"
  "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
  "abcdefghijklmnopqrstuvwxyz";

    // Genera el identificador del cliente (client ID).
    for (int i = 0; i < idLength ; i++)
    {
        clientID[ i ] = alphanum[ random( 51 ) ];
    }
    clientID[ idLength ] = '\0';
}
```

```
/**
 * Funcion que realiza la subscripcion a campos de un canal.
 *   subChannelID - Canal al que se subscribe.
 *   field - Campo al que se subscribe. Valor 0 significa que se
subscibe a todos los campos del canal.
 *   readKey - Read API key del canal de subscripcion.
 *
 */

int mqttSubscribe(unsigned long subChannelID, int field, char*
readKey)
{
    String myTopic;

    if (field==0)
    {
      //Tema para subscribirse a todos los campos del canal
        myTopic="channels/"+String( subChannelID
)+"/subscribe/json/"+String( readKey );
    }
    else
    {
      //Tema para subscribirse a un campo del canal
        myTopic="channels/"+String( subChannelID
)+"/subscribe/fields/field"+String( field )+"/"+String( readKey );
    }


    Serial.println( "Subscribiendose a " +myTopic );

    return mqttClient.subscribe( myTopic.c_str() ,0 );
}

/*
 * Funcion que publica datos a un canal
 *   pubChannelID - Canal en el que se publica.
 *   pubWriteAPIKey - Write API key del canal al que publica.
 *   dataSend[] - Array con los datos que se van a publicar de
temperatura y humedad
 *
 */

void mqttPublish(unsigned long pubChannelID, char* pubWriteAPIKey,
float dataSend[])
{

  float temp;
  float hum;

  temp = dataSend[0];
  hum = dataSend[1];

 //Extraemos en una variable la informacion útil que se va a
enviar
  String payload="field1=";
  payload+=temp;
  payload+=" &field2=";
  payload+=hum;
  payload+=" &status=MQTTPUBLISH";
```

```
  // Tema en el que publicar.
  String topicString ="channels/" + String( pubChannelID ) +
"/publish/"+String( pubWriteAPIKey );
  // Publicacion de los datos en el tema correspondiente
  mqttClient.publish( topicString.c_str(), payload.c_str() );
  Serial.println( "\nPublicado en el canal " + String(
pubChannelID ) + "\n");

}
```

ThingSpeak_Library

```
/*
* Esta funcion es la encargada de enviar los datos a ThingSpeak
*   myChannelNumber1 - Número identificador del canal de
ThingSpeak al que se envían los datos.
*   myWriteAPIKey1 - Clave de escritura del canal
*   fieldt - Campo del canal en el que se escribe la temperatura
*   fieldh - Campo del canal en el que se escribe la humedad
*   dataSend[] - Datos de temperatura y humedad leidos por el
sensor
*
*/

void sendingData(unsigned long myChannelNumber1, char *
myWriteAPIKey1,const unsigned int fieldt,const unsigned int
fieldh, float dataSend[])
{
  float temp;
  float hum;

  temp = dataSend[0];
  hum = dataSend[1];

  // Carga los valores a enviar
  ThingSpeak.setField(fieldt, temp); //field 1
  ThingSpeak.setField(fieldh, hum); //field 2


  // Escribe todos los campos a la vez.
  ThingSpeak.writeFields(myChannelNumber1, myWriteAPIKey1);

  //Escribe primero en el campo 1 y despues en el campo 2

  //ThingSpeak.writeField(myChannelNumber, fieldt, temp,
myWriteAPIKey1);
  //ThingSpeak.writeField(myChannelNumber, fieldh, hum,
myWriteAPIKey1);


  Serial.println("\n¡Datos enviados a ThingSpeak!");

}



/*
* Esta funcion se encargada de  leer los datos enviados desde
ThingSpeak
*   myChannelNumber2 - Número identificador del canal de
ThingSpeak del que se van a leer los datos.
*   fieldLED - Campo del canal del que se van a leer los datos
*   myReadAPIKey2 - Clave de lectura del canal
*
*/

void readingData(unsigned long myChannelNumber2, unsigned int
fieldLED, char* myReadAPIKey2)
```

```
{

    int data = 0;

  // Se leen los datos del canal de ThingSpeak
    data = ThingSpeak.readFloatField(myChannelNumber2, fieldLED,
myReadAPIKey2);
    Serial.println("\nDato leido de ThingSpeak "+String(data));

  //Si se recibe un 1 el LED se enciende y si se recibe un 0 se
apaga
    if(data == 1){
      digitalWrite(LED, HIGH);
      Serial.println("LED ON\n");
    }
    else{
      digitalWrite(LED, LOW);
      Serial.println("LED OFF\n");
    }

}
```