Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

UTILIZACIÓN DE GOOGLE CLOUD PARA EL ALMACENAMIENTO Y PROCESADO DE DATOS MONITORIZADOS REMOTAMENTE

Alumno: Fernando José Lara Cardeñas

Tutor 1: Prof. D. José Enrique Muñoz Expósito
Depto.: Ingeniería de Telecomunicación
Tutor 2: Prof. D. Sebastián García Galán
Depto.: Ingeniería de Telecomunicación

UNIVERSIDAD DE JAÉN

ESCUELA POLITÉCNICA SUPERIOR DE LINARES

TRABAJO FIN DE GRADO
Curso 2019-2020

UTILIZACIÓN DE GOOGLE CLOUD PARA EL ALMACENAMIENTO Y PROCESADO DE DATOS MONITORIZADOS REMOTAMENTE

Alumno: Fernando José Lara Cardeñas
Tutor 1: José Enrique Muñoz Expósito
Departamento: Ingeniería de Telecomunicación
Tutor 2: Sebastián García Galán
Departamento: Ingeniería de Telecomunicación

Firma del alumno	**Firma Tutor 1**	**Firma Tutor 2**
LARA CARDEÑAS FERNANDO JOSE - 78683494B Firmado digitalmente por LARA CARDEÑAS FERNANDO JOSE - 78683494B Fecha: 2020.06.16 11:35:40 +02'00'	MUÑOZ EXPOSITO JOSE ENRIQUE - 27386513E Firmado digitalmente por MUÑOZ EXPOSITO JOSE ENRIQUE - 27386513E Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-27386513E, givenName=JOSE ENRIQUE, sn=MUÑOZ EXPOSITO, cn=MUÑOZ EXPOSITO JOSE ENRIQUE - 27386513E Fecha: 2020.06.16 11:51:33 +02'00'	

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. RESUMEN

1.1 Resumen

Este trabajo fin de grado (TFG) tiene como objetivo la creación de aplicaciones y servicios en Google Cloud Platform para almacenar y procesar datos que se han enviado remotamente a través de diferentes nodos. Se ha realizado el desarrollo de un sistema que permite la monitorización de la temperatura de manera remota y centralizada, así como la temperatura y la humedad.

Los nodos monitorizan la temperatura en periodos no inferiores a un minuto y estos a su vez la envían a un punto central o encaminador, que es el encargado de enviarlo a un servidor para su almacenamiento. En este caso el servidor es el IoT Core de Google.

Dados los requisitos del trabajo, se implementa una red de nodos inalámbricos de bajo consumo. Los nodos que se utilizan son gestionados por microcontroladores ESP32 y Xbee que monitorizan diferentes sensores conectados a ellos. Considerando el requisito de escalabilidad y despliegue rápido, para su almacenamiento se emplea un sistema Cloud, en este caso Google Cloud Platform. Los nodos deben ser capaces de comunicarse a distancias considerables para poder reducir la cantidad de estos y a su vez eliminar la necesidad de cualquier tipo de cableado.

Se diseña una función en Cloud Functions para poder almacenar la información en una base de datos como es Cloud SQL, para después con App Engine programar diferentes aplicaciones para visualizar la información, y también se envía esta información a un gran almacén de datos como es BigQuery, que permite su posterior análisis. La aplicación desarrollada en PYTHON funciona de pasarela entre estos servicios y una página web que ha sido diseñada con HTML 5 para poder visualizar los datos almacenados.

Finalmente se utiliza la plataforma Grafana para crear un cuadro de mando donde se visualiza también la información almacenada. Es una solución muy sencilla y que puede ser adoptada por usuarios avanzados que no dominan la programación.

1.2 Abstract

The main aim of this final degree Project (TFG) is to create applications and services on the Google Cloud Platform to store and process data that has been sent remotely through different nodes. The development of a system that allows remote and centralized temperature monitoring, as well as temperature and humidity, will be carried out.

The nodes monitor the temperature in periods of no less than one minute and these in turn send it to a central point or router, which is in charge of sending it to a server for storage. In this case, the server will be Google´s Iot Core.

According to the job requirements, a network of low consumption wireless nodes are implemented. The nodes to be used will be ESP32, Arduino Uno and XBee microcontrollers that will monitor different sensors connected to them. Considering the requirement of increasing and fast deployment, a Cloud system is used for its storage, in this case Google Cloud Platform. The nodes must be able to communicate over appreciable distances in order to reduce the number of these and in turn remove the need for any type of wiring.

Furthermore, a function is designed in Cloud Functions to be able to store the information in the Cloud Sql database and then with the App Engine program different applications to display the information, and this information is also sent to a large data warehouse such as BigQuery, which allows its subsequent analysis. An application developed in Python that will function as a gateway among these services and a web page that has been designed with Html 5 to be able to visualize the stored data.

To conclude, the Grafana platform is used to create a dashboard where we will also view the stored information. It is a very easy solution and it can be adopted by advanced users who do not master programming.

2. INTRODUCCIÓN

En este punto se expondrá el contexto de este trabajo, así como su desarrollo, además se describe la estructura del documento y se presenta el contenido de cada capítulo que constituye este documento.

2.1 Justificación y contexto del TFG

En la actualidad, vivimos en un mundo conectado en el que podemos conocer todo lo que ocurre tanto a nuestro alrededor como en un lugar remoto. La obtención de información es importante para poder actuar de forma rápida ante cualquier cambio de esta y que mejor que toda esta información sea accesible desde cualquier sitio sin necesidad de tener una infraestructura propia o simplemente teniendo una solución híbrida, que combine hardware propio nuestro con una infraestructura externa como puede ser Google Cloud Platform. Esta infraestructura permite acceder, gestionar y utilizar sus herramientas con un coste de solo por uso, en lugar de tener que realizar una gran inversión en infraestructuras físicas.

El diseño de nuevas redes de alta velocidad, la necesidad de tener cada vez más datos en tiempo real, así como el bajo coste que suponen los nuevos sistemas de almacenamiento han propiciado la utilización del Cloud Computing, que propone utilizar recursos y servicios para formar una red de ordenadores en la nube.

2.2 Estructura del documento

Este documento contiene una relación de puntos en los que se detalla el escenario general utilizado, para la realización de una serie de servicios y aplicaciones en Google Cloud Platform, así como la puesta en marcha de Grafana para un control de datos. Estos puntos se distribuyen de la siguiente manera:

- **Resumen:**
  Breve definición del origen del presente proyecto
- **Introducción:**
  Exposición del alcance del proyecto.
- **Objetivos:**
  Definición punto por punto de servicios o tareas que han de ser cubiertas con el presente proyecto.
- **Materiales y métodos:**
  Detalle de todos los materiales y sus especificaciones, así como las técnicas utilizada para lograr los objetivos marcados.

- **Resultados y discusión:**
  Valoración de las consecuencias obtenidas con el presente proyecto en base a los objetivos marcados.
- **Conclusiones:**
  Análisis del resultado obtenido con el desarrollo del proyecto.
- **Líneas futuras:**
  Análisis de las posibilidades de ampliación y evolución en el tiempo del presente proyecto.
- **Bibliografía:**
  Listado de fuentes documentales, principalmente electrónicas empleadas como fuentes de información.
- **Anexos:**
  Documentos adicionales que complementan la información expuesta en el presente documento

3. OBJETIVOS

El principal objetivo de este TFG como se ha dicho anteriormente es la realización de una serie de aplicaciones y servicios en Google Cloud Platform para almacenar y procesar datos que se obtienen remotamente de varios dispositivos. Los parámetros registrados son la temperatura y la humedad. Para poder llevar a cabo el principal objetivo se han desarrollado una serie de objetivos específicos:

- Creación de una red de dispositivos remotos, que envíen datos de forma inalámbrica.
- Creación de los servicios de Google Cloud Platform
- Conectar, administrar e ingerir datos procedentes de dispositivos remotos en el servicio Cloud IoT Core.
- Programar aplicaciones basadas en eventos con Cloud Fuctions.
- Desplegar aplicaciones que hacen uso de un almacén de datos masivo con Cloud DataFlow y BigQuery
- Administrar un servicio de base de datos que facilite la configuración y el mantenimiento de datos relacionados con Cloud SQL.
- Publicar aplicaciones web en línea, realizando un Front-End y un Back-End para la visualización de datos con App Engine.
- Conexión de la aplicación Grafana para crear un cuadro de mandos.

4. MATERIALES Y MÉTODOS

4.1 Antecedentes

En lo referente a sensores, atrás quedan los aparatos de medida analógicos, como punto a su favor nos encontramos con su robustez e independencia de fuentes de energía eléctrica para su funcionamiento. En contra y mayormente de ahí su evolución hasta los captadores digitales es difícilmente practicable la lectura de valores de manera automatizada en el tiempo.

No obstante, estos medidores analógicos, ya sean termómetros para temperatura o higrómetros para la medición de la humedad ambiente u en objetos, los medidores analógicos no quedan completamente descartados por su citada robustez. De tal modo que son frecuentemente empleados en laboratorios en ocasiones como segundo dispositivo de medición para comprobar e incluso para calibrar otros medidores digitales que lo precisen.

Otro factor que provoca el paso a tecnologías electrónicas, pese a la dependencia eléctrica de los mismos es la facilidad de lectura que proporcionan los sensores digitales frente a los analógicos.

La evolución de los sensores digitales conlleva con ellos el aumento de la fragilidad y su miniaturización, esto evoluciona en la separación del equipo de control, dispositivo electrónico que procesa la medida obtenida en el sensor y muestra en un reloj, display o pantalla el valor equivalente en la unidad de medida configurada, del dispositivo captador o sensor, entendiendo por sensor un captador electrónico semiconductor sensible a una magnitud física capaz de obtener variaciones proporcionales a la misma con equivalencia eléctrica, ya sea de forma digital o analógica.

Dentro del contexto de este proyecto que se toman como referencia magnitudes físicas tales como temperatura y humedad, lo más próximo existente en la actualidad son las centrales meteorológicas, en este caso pudiendo llegar a obtener algún valor adicional como es la presión barométrica. La introducción de estas centrales manifiesta la complejidad de una instalación cableada cuando los sensores quieren ser ubicados en exterior para que los valores sean lo más representativos posibles. Es por esto que no tardan en surgir los sensores inalámbricos, mayormente estos sistemas suelen ser cerrados, propiedad de los fabricantes y no facilitan información del desarrollo de la comunicación inalámbrica puesto que están pensados para que un receptor sea utilizado únicamente son sus sensores.

Como señal inalámbrica, se basan en las mismas tecnologías inalámbricas empleadas en mandos de garaje, bandas generalmente de 333Mhz, 433Mhz 866Mhz

933Mhz, pudiendo variar dependiendo del país de comercialización para adaptarse a la legislación radioeléctrica correspondiente.

Pese a transmitir la señal del sensor a la central de manera inalámbrica en sus inicios no estaba previsto el almacenamiento de los valores a modo histórico, de hecho, la mayor parte de las centrales de este tipo que aun hoy están en el mercado no almacenan los valores, siendo una minoría consideradas premium o de gama alta las que están dotadas de algún tipo de conexión con PC, generalmente USB, para que los muestreos realizados sean almacenados en el PC.

Este tipo de dispositivos actualmente a continuado evolucionando con la introducción de lo que podemos considerar lo más próximo al consumidor que ha llegado de forma masiva la Inteligencia Artificial, IA, hasta ahora. Esto son los asistentes de voz, Alexa, Google, Cortana, Siri, entre otros. Actualmente más que una IA propiamente dicha es una interfaz de voz, si bien esta en desarrollo continuado, y es un producto por el que los principales referentes tecnológicos están apostando fuertemente. Gracias a esto, surge una nueva evolución en lo referente a captadores ambientales y su conectividad, es posible actualmente encontrar dispositivos dotados de conectividad inalámbrica tanto para internet como de una banda de largo alcance y bajo consumo y bajo ancho de banda puesto que están pensados para ser captadores o dispositivos autónomos.

Este tipo de sensores son los más apropiados para el presente proyecto. Aproximan sus características a los requisitos del proyecto, pero aun así no llegan a tener capacidades de controlar el almacén de datos, en la mayoría de las veces los tiempos entre muestreo son fijados por el fabricante, no existe control sobre el software, tienen compatibilidad con las plataformas de control por voz más extendidas, pero no son de código abierto. No obstante, algunos de ellos utilizan el mismo tipo de hardware que se selecciona para el desarrollo del proyecto, pero con un coste superior por dispositivo. Esto descarta la posibilidad de utilizar sensores comerciales en este caso por el coste, es necesaria una reprogramación del firmware propietario del fabricante, además del coste de adquisición y no tener el control sobre la permanencia en el mercado del mismo, existencias o modificaciones a nivel de hardware que decidiera hacer el fabricante y que afectaran al proyecto una vez estuviera en producción.

Pese a enviar los datos para su procesamiento a los asistentes de voz, o poner la capacidad de lectura a disposición de este, en adelante serán considerados enviados los datos a Cloud. Los datos no son almacenados de forma que el usuario del servicio tenga control sobre ellos generalmente o al menos control total, así mismo no se puede configurar el sistema para ningún otro sensor que no haya sido previsto por el fabricante.

Si a esto se intenta realizar un procesamiento de la información de miles de sensores de manera coordinada y centralizada, estos sensores comercializados no son válidos.

Es por lo tanto necesario el desarrollo de un hardware basado en microcontroladores orientados a desarrollo de dispositivos de los denominados Internet of Things, IoT. Esto va a permitir controlar la frecuencia con la que se muestrea la información, sobre qué tipo de dispositivos se muestrea, ya sea digitales o analógicos y como es procesada una vez obtenida. Si bien esto otorga una gran flexibilidad en cuanto a la conexión del proyecto con el mundo real sigue existiendo el problema del almacenamiento puesto que almacenar muestreos en el propio dispositivo tiene capacidades muy limitadas a medio-largo plazo. Toma relevancia en este aspecto los servicios Cloud.

En lo que a Cloud se refiere desde su inclusión hace años, ha evolucionado introduciendo multitud de servicios intermedios que no pertenecen propiamente a la segmentación que en su origen se hacía sobre los sistemas Cloud a nivel de Infraestructura, plataforma o servicio. Cada vez los servicios prestados por los sistemas Cloud están más orientados a soluciones específicas e interconectan otros servicios más básicos prestados por el mismo Cloud.

Dentro del ámbito del Cloud Computing, existe gran variedad de proveedores a día de hoy. Los más populares son los proporcionados por los principales proveedores digitales, en este caso van a ser contemplados tres de ellos. Estos son:

Microsoft Azure, Amazon AWS y Google Cloud. No obstante, es un referente también IBM disponiendo de una división específica para IoT. Así mismo los principales fabricantes de componentes hardware IoT disponen de algún tipo de servicio Cloud para permitir expandir la conectividad de su hardware y las posibilidades en el desarrollo de proyectos. Muchos de estos son gratuitos, renunciando generalmente a privacidad, puesto que gran parte de las veces que un servicio es proporcionado de forma gratuita es porque realmente el producto son los datos o la entidad que utiliza el servicio.

El objetivo en cuanto al almacenamiento y procesamiento de la información es poder disponer de control sobre el mismo permitiendo ser empleado para la toma de decisiones, consulta e incluso como realimentación a un sistema más complejo que sea capaz de digerir en tiempo real el gran volumen de datos generado de manera global por todos los dispositivos conectados a la red. Es necesario que el sistema Cloud pueda prestar servicios orientados a BigData. Queda así preparada la infraestructura que desarrolla el presente proyecto para la interconexión con proyectos más complejos que requieran la información facilitada por la red de sensores.

Considerando esto, los proveedores Cloud capaces de prestar el conjunto de servicios que las ambiciones de este proyecto necesitan, quedan en los citados inicialmente Azure, AWS, Google.

Amazon Web Services: plataforma muy utilizada, de las primeras que empezaron a ofrecer los servicios cloud, con muchísimas funcionalidades y servicios, quizás el que más. Tiene un pequeño inconveniente y es que ofrece solo tecnologías diseñadas por ellos mismos, además de ser una de las más caras.

Microsoft Azure: está apostando por los servicios cloud muy fuerte en los últimos años, además de ofrecer productos de Microsoft, también están ofreciendo servicios open-source. Es una plataforma híbrida y por lo tanto se puede integrar muy fácilmente con otras plataformas privadas. Indicar que el manejo de esta plataforma es más complicado que las otras.

Google Cloud: Es de las plataformas más nuevas, y más fáciles de utilizar, además de ser una de más económicas, aunque de las tres nombradas aquí es la que menos servicios ofrece.

La decisión en este momento llega a ser puramente económica o bien por facilidad y agilidad de desarrollo. Dentro de este último es conocido el acuerdo existente entre la Universidad de Jaén, para lo cual es desarrollado el proyecto, y Google otorgando crédito gratuito para utilizar los servicios de su plataforma Cloud.

Basado en este acuerdo, para el desarrollo del proyecto se emplea Google Cloud Platform. La diversidad de servicios más allá de los propios ya contemplados para el desarrollo facilita la futura expansión del proyecto de una manera rápida, sin necesidad de interconectar sistemas Cloud o migrarlos.

4.2 Esquema de interconexión

Teniendo presente el esquema de funcionamiento (Figura 4.1) del sistema expuesto anteriormente, a continuación, se procede a la explicación del mismo por partes comenzando desde la toma de datos en los sensores, y siguiendo el flujo que la información llevaría a través del esquema hasta llegar al Front-End, sitios web habilitados para ser consultada por el usuario.

El escenario principal (Figura 4.1) que se plantea en este trabajo es el siguiente:

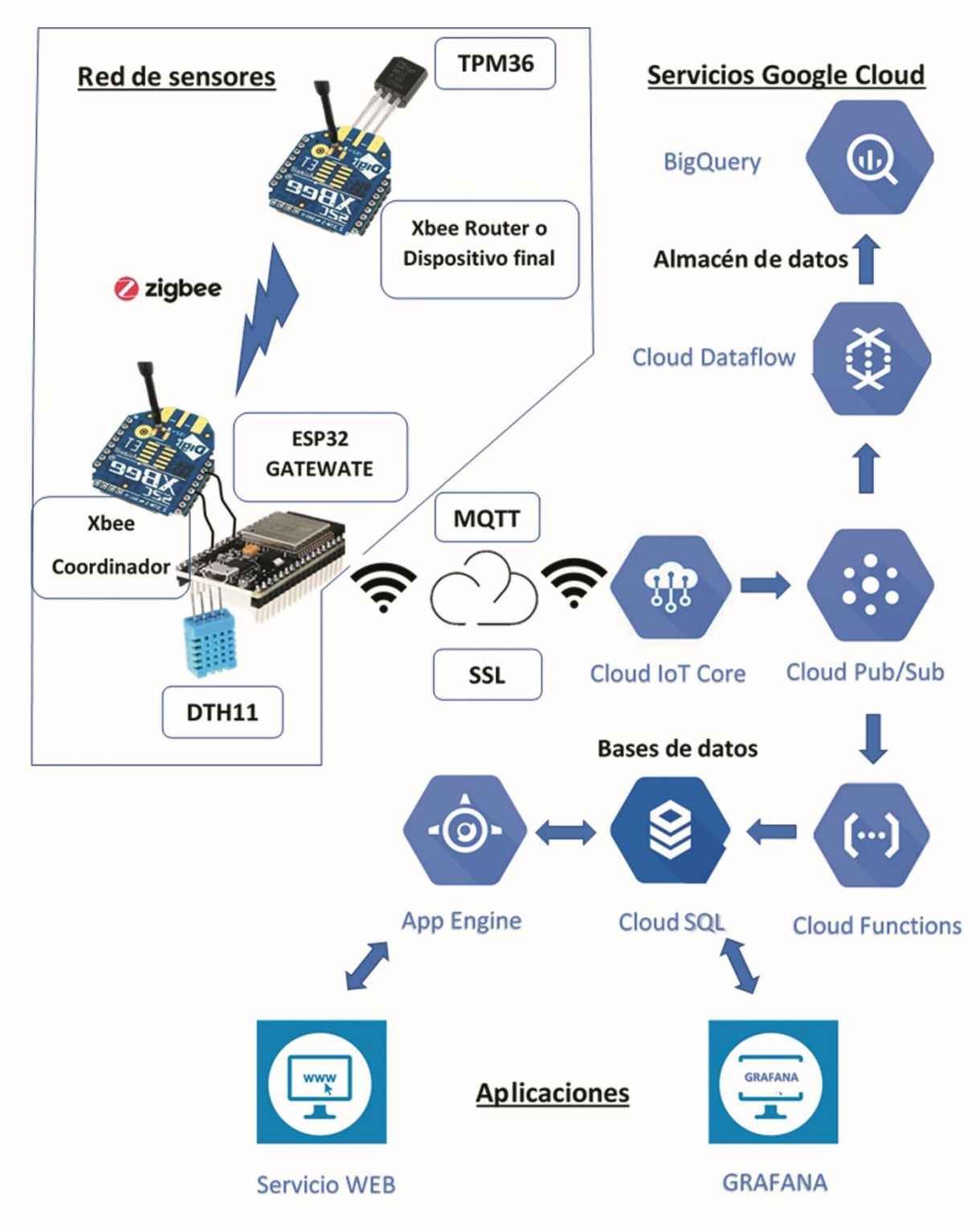


Figura 4.1 Escenario principal

La información se genera en los sensores, que realizan un muestreo, en este caso de parámetros ambientales, temperatura y humedad. La muestra de temperatura tomada

de manera remota en un dispositivo Xbee, a intervalos de 60 segundos, es transmitida hacia el nodo coordinador, bien de forma directa o bien pasando por otros dispositivos Xbee configurados para muestrear en el rol de router, lo que permite extender la cobertura de la red ZigBee. Una vez que la muestra alcanza el dispositivo Xbee coordinador, este está configurado en modo API, y transmite toda la información recibida por un canal serie, en este caso hacia el dispositivo ESP32, que dispone de dos pines GPIO configurados mapeados como uno de los tres puertos series que soporta este dispositivo. El ESP32 se encarga de interpretar todas las tramas recibidas por el canal serie del Xbee coordinador. En las tramas está contenida la dirección MAC del dispositivo origen de la misma, así como el muestreo realizado de los pines a los que está conectado el sensor, en este caso TMP36 sensor analógico de temperatura.

El dispositivo ESP32 interpreta la lectura del sensor, obteniendo según las especificaciones del fabricante, la equivalencia en grados centígrados. Esta lectura tiene una precisión de dos decimales, es pasada a entero y se compone un JSON para transmitir la información a su registro, para ello se le integra a la información MAC del nodo y el muestreo de la temperatura, la MAC del ESP32 que hace la función de encaminador, y el identificador de zona, un identificador estático definido en la instalación que permite la agrupación de los grupos de sensores o conjuntos de ellos. En este caso el parámetro humedad que no es muestreado se transmite a 0. Este último identificador es introducido a efectos de poder clasificar, agrupar y ordenar lecturas si se realiza el despliegue del presente sistema múltiples veces gestionando la información de todos ellos de manera centralizada en el mismo Cloud. En este momento es cuando las tramas son etiquetadas con la huella de tiempo, timestamp. Esto es así porque los dispositivos empleados no disponen de reloj en tiempo real, y para poder etiquetarlos se emplea un servidor NTP que facilita la actualización del tiempo, esta característica no se puede implementar en el dispositivo Xbee remoto, por lo que se integrara un error en el marcado de tiempo de la muestra equivalente al tiempo que tarde en llegar la trama desde el dispositivo Xbee remoto al puerto serie del ESP32.

Una vez la información esta formateada en JSON esta es introducida en el payload de un paquete MQTT para ser transmitido al servidor empleando el estándar PUB/SUB, en este caso Pub. El servidor MQTT está ubicado en el Cloud de Google, por lo tanto, el dispositivo ESP32 integra la configuración del certificado de seguridad auto-firmado y previamente registrado como autorizado para realizar la conexión con HTTPS/MQTT identificación/publicación del dispositivo en Google Cloud (en adelante GC).

Paralelamente, el dispositivo ESP32 dispone de un sensor DHT11, sensor digital para muestreo de temperatura y humedad, este es muestreado cada minuto y de igual manera se compone un JSON para su envío por MQTT a GC. En este caso coincide el ID del dispositivo con el ID del encaminador siendo en ambos casos la dirección MAC del ESP32.

La información muestreada de todos los tipos de dispositivos llegados a este punto se encuentra ya en el Cloud IoT, Pub/Sub almacenada. Falta implementar una función que permita su almacenaje permanente de manera estructurada en una base de datos, siendo en este trabajo MySQL.

Por ello se emplean las Cloud Function.

Una Cloud Function es vinculada a un evento que la dispara, en este caso, a una suscripción de MQTT realizada al topic sobre el que se publican los muestreos en este caso "environment", de esta forma la función será ejecutada cada vez que se reciba un nuevo evento. A modo depuración se crea una función inicial que solamente muestra la información en mensajes de información dentro del log de GC, y finalmente se crea una función que lee el topic, obtiene el payload un JSON en este caso y del que obtiene los Identificadores del origen de la trama, zona, gateway, dispositivo, muestreo y tiempo. Esta función conecta con la base de datos e inserta los datos.

Pub/sub a su vez tiene configurado el servicio Cloud Dataflow y este BigQuery. Este servicio va a permitir la consulta masiva a la información en tiempo real. Es necesario dada la proyección, de aplicarlo a un gran número de sensores muestreando cada minuto durante 24h 7dias, lo que genera volúmenes de datos considerables para un único sensor, para una red de sensores el volumen de datos no es manejable de forma práctica empleando simplemente una base de datos SQL sin realizar filtrados específicos.

Desde la perspectiva de la base de datos SQL, la información es consumida por el usuario empleando una página web que esta facilitada por el APP Engine o bien por un servidor externo que implemente Grafana accediendo para ello con las credenciales de usuario a crear una conexión con la base de datos SQL.

La aplicación web que está desplegada en el APP Engine está desarrollada empleando el framework Flask, que es un framework para Python, en la parte dinámica de la web en el lado del servidor, en lo que respecta al lado del cliente, la web emplea HTML5, CSS3 y JavaScript. Se desarrolla para que el tráfico entre cliente y servidor sea el menor posible mejorando así la experiencia de usuario al reducir los tiempos de espera de carga del sitio web. Esta funcionalidad se consigue empleando las peticiones asíncronas de JavaScript. HTML5 estructura la información de manera semántica y CSS3 se encarga de la visualización y gráficos.

Seguidamente, se especifica con más detalle los elementos del hardware implicado en la realización de los muestreos ambientales según se ha explicado anteriormente.

4.3 Elementos de hardware (Red de sensores)

Para poder obtener datos del exterior es necesario la creación de una red de sensores, conectados a unos microcontroladores que serán los que se conectarán a la infraestructura externa. Estos sensores estarán conectados en lugares remotos.

La red de sensores (Figura 4.2) diseñada se muestra seguidamente:

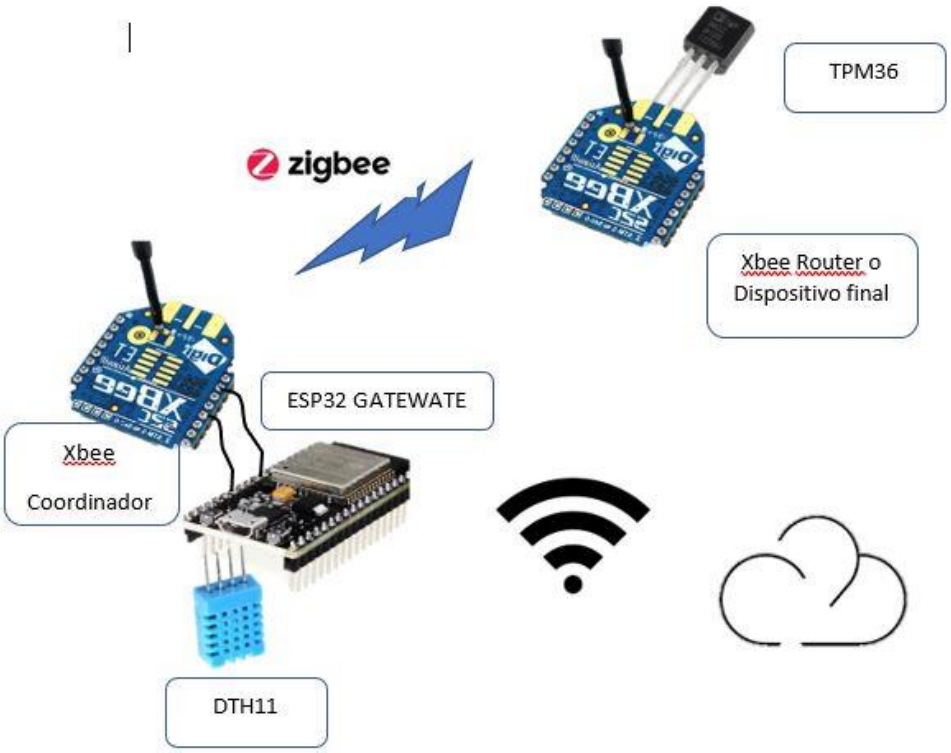


Figura 4.2 Red de sensores

El nodo principal y equipo que se utiliza como pasarela para la conexión y el envío de datos a una infraestructura externa, es el microcontrolador ESP32. Este tendrá conectado directamente un sensor de temperatura y humedad y un módulo Xbee que será el coordinador de una red ZigBee para el envío inalámbrico de otros datos de sensores que estén conectados a otros Xbee que sean dispositivos finales. Los módulos Xbee se utilizan para llegado el momento poder expandir una red de sensores inalámbricos.

El sensor DHT11 es el que se ha utilizado para la temperatura y humedad, los Xbee son los modelos S2C y el sensor que tendrá conectado unos de los Xbee será el TMP36 que solo mide la temperatura. (Figura 4.3)

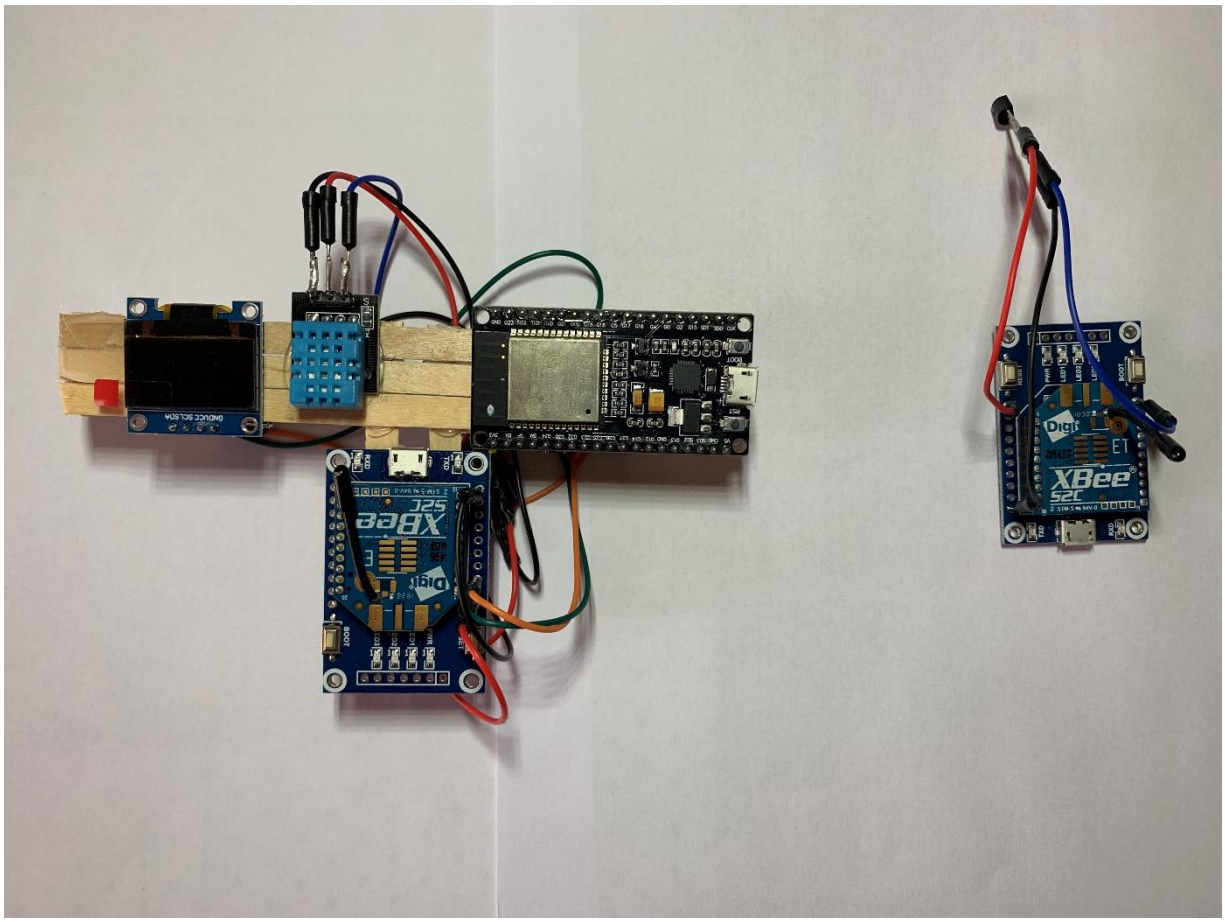


Figura 4.3 Red de sensores

A continuación, se detallan las características de los sensores encargados de tomar las muestras citados anteriormente sensores DHT11, TMP36, para pasar a continuación a detallar las características del microcontrolador ESP32, así como los dispositivos Xbee empleados y sus modos de funcionamiento.

Como últimos subapartados (4.1.4 y 4.1.5), se incluyen los inconvenientes hallados en el proceso de desarrollo que dieron lugar a la necesidad de descartar y sustituir componentes de hardware ya que sus prestaciones no cumplían con los requisitos mínimos necesarios y su implementación era inviable. Así mismo se incluye el detalle del mantenimiento necesario previsto del hardware expuesto en los puntos anteriores.

4.3.1 Sensores

4.3.1.1 Sensor Temperatura y Humedad DHT11

El sensor digital DHT11 (Figura 4.4) mide la temperatura y la humedad relativa es de bajo coste y fácil de usar. Incorpora un sensor de humedad capacitivo, para medir el aire circundante un termistor y para poder mostrar los datos, utiliza en el pin de datos una

señal digital. Se conecta a 3-5V el pin VCC de alimentación, a Tierra el pin GND y el pin de datos a uno digital de un microcontrolador. [1]

Una de las desventajas que presenta el DHT11 es que para poder muestrear nuevos datos es necesario que pasen 2 segundos. Para la comunicación entre el sensor y el microcontrolador se utiliza un solo cable teniendo este como máximo una longitud de 20 metros.

Este sensor digital se conectará al microcontrolador ESP32.

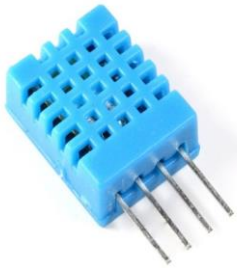


Figura 4.4 Sensor DHT11

4.3.1.2 Sensor Temperatura TMP36

El TMP36 (Figura 4.5) es un sensor de temperatura de tipo analógico, de bajo consumo y también de bajo coste. Trabaja a bajo voltaje de entre 2,7V y 5,5V y está calibrado directamente en grados Celsius. Su precisión es de +-2ºC, linealidad de +-0.5ºC. Es capaz de trabajar desde los -40ºC hasta los 125ºC. [2]

Tiene 3 pines de conexión uno para la alimentación a 5v, otro para la salida del dato de temperatura y otro que se conecta a la tierra.

Este sensor estará conectado a los Xbee

Figura 4.5 Sensor TMP36

4.3.2 Microcontrolador ESP32

4.3.2.1 Descripción del dispositivo

El ESP32 (Figura 4.6) es un microcontrolador con conexión wifi y bluetooth y torre de protocolos IP. Esta comunicación es gestionada desde la propia CPU. Respecto al wifi, soporta los estándares 802.11 b/g/n.

Es un potente dispositivo preparado para IoT ya que cuenta con un procesador Xtensa Dual-Core LX6 de 32 bits a 160 o 240 MHz y al tener dos núcleos dedica uno de ellos a la comunicación IP y WiFi y el otro al resto de procesos. Tiene una memoria RAM de 520 kB, accesible por ambos procesadores y puede utilizar memoria RAM externa adicional de hasta 8 MB. La memoria ROM interna es de 448 kB, utilizada sobre todo para las funciones internas. La memoria flash, ha de ser externa y soporta hasta 16 MB.

Este microcontrolador puede comunicarse utilizando protocolos UART, I2C y SPI.

La comunicación UART (Transmisor-Receptor Asíncrono Universal) trasmisión y recepción asíncronas universales, es una de las más utilizadas en los microcontroladores. Para transmitir utiliza una línea simple de datos y para recibir utiliza otra línea.

I2C es una comunicación síncrona que usa dos líneas, una línea para el reloj (SCL) y otra línea para el dato (SDA), por un mismo cable envían datos tanto el maestro como el esclavo, siendo está controlada siempre por el que crea la señal del reloj, que no es otro que el maestro. I2C utiliza direccionamiento en lugar de selección de esclavo.

SPI es una comunicación simple, la señal de reloj es enviada por el maestro que después de cada pulso de reloj envía un bit al esclavo y este le devuelve un bit. Para las señales se utilizan los nombres de SCK para el reloj, MISO para el Maestro In Esclavo Out y MOSI para el Maestro Out Esclavo In. [3]

Además, utiliza software para implementar el protocolo de cifrado TLS 1.2, pero esto aumentará en gran medida la carga del procesador.

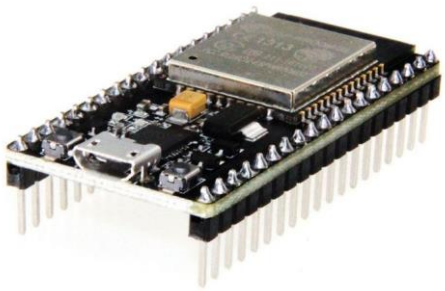


Figura 4.6 Microcontrolador ESP32

4.3.2.2 Pines utilizados

Los pines (Figura 4.7) [5] que se han utilizado en este trabajo son los siguientes:

GPIO 27 datos del DHT11

GPIO 34 y GPIO 35 – puerto serie a 9600 para recibir datos del Coordinador Xbee en modo API

GPIO 21 y GPIO22 – SLC SDA para conectar una pantalla OLED

A modo de prueba se puede conectar empleando el puerto USB a un powerbank.

Todos los elementos conectados son alimentados a través de 3.3V de la propia placa.

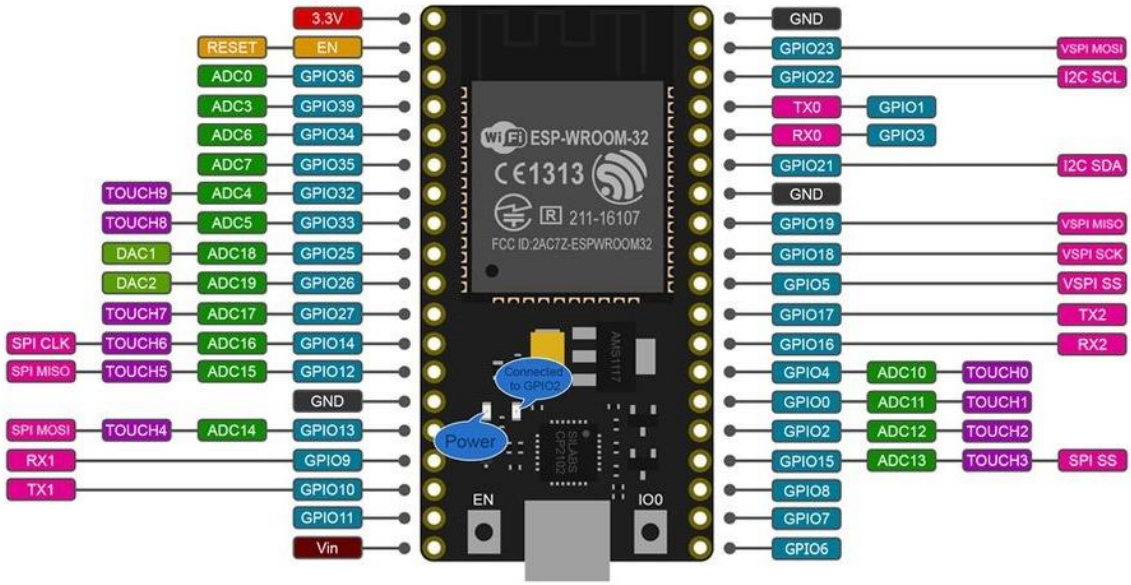


Figura 4.7 Pines ESP32 – Fuente: prometec.net

4.3.2.3 Software utilizado

Para establecer comunicación con el ESP32 se ha empleado el entorno de desarrollo Arduino IDE, esto es posible puesto que el microcontrolador es compatible con Arduino. No obstante, es necesario incluir sus librerías y dependencias en el entorno para que incluya la configuración de este hardware, así como los controladores del adaptador USB que integre la placa de desarrollo empleada, aunque son dos tipos de controladores USB los más empleados entre este tipo de placas.

Una vez conectado el dispositivo al puerto USB del PC, es generado un puerto COM virtual gracias a los drivers USB instalados compatibles con el controlador USB de la placa de desarrollo. Al instalar las dependencias para esta placa en el entorno de desarrollo, facilita el reconocimiento de las características del hardware para configurar el

compilador y la comunicación. No obstante, el puerto COM generado es necesario seleccionarlo en el entorno de desarrollo, así como la velocidad de la comunicación serie a emplear. En este caso se ha utilizado 115200. Una vez realizado esto ya es posible compilar y subir el código al dispositivo ESP32. Así mismo permite la apertura de una ventana de consola de comunicación con el puerto serie al que se ha conectado el ESP32 a modo de depuración, en este proyecto.

4.3.3 Xbee

4.3.3.1 Descripción del dispositivo

Los Xbee (Figura 4.8) son dispositivos inalámbricos fabricados por Digi International utilizan el protocolo de comunicación IEEE 802.15.4 para diseñar redes punto a punto o redes punto a multipunto, son robustos, de bajo costo, bajo consumo y tienen un alcance en sus distintos modelos entre 60 metros y los 10 kilómetros. Está preparado para aplicaciones que tienen un tráfico de datos bastante alto, baja latencia y tiempo de comunicación predecible. Es una implementación de Digi basada en el protocolo Zigbee.

Existen diferentes series de Xbee, en este trabajo se van a utilizar los Xbee Series S2C, que son fáciles de usar, debido a que no es necesario configurarlos y tiene un módulo de transmisión de datos inalámbrico a 2mW.

Algunas de las especificaciones son:

Potencia Max: 9,82 mW (9,92 dBm)

Frecuencia: separación de canales de 5 MHz, que comienza en 2405 MHz y termina en 2480 MHz. [4]

En el interior puede llegar a tener un alcance máximo de 60 metros y un rango en línea de vista de 1200 metros en condiciones ideales.

Xbee consta de 20 pines de los cuales 13 son entradas y salidas digitales y tiene 4 entradas analógicas.

Existen 3 roles para estos microcontroladores: Coordinador, Router y Dispositivos finales.

Figura 4.8 Xbee S2C

Para configurar los Xbee, se necesita un XBee USB Adaptador Explorer (Figura 4.9), ya que los pines del Xbee no se ajustan a una protoboard normal. Este adaptador se encarga de comunicarse directamente con los pines seriales del Xbee

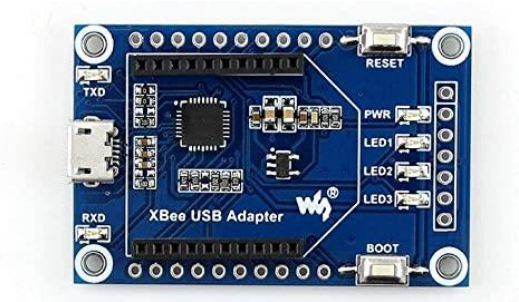


Figura 4.9 Xbee Adaptador USB

4.3.3.2 Software utilizado

XCTU

Este software es el que se utiliza para configurar los Xbee y es propio de DIGI, su nombre XCTU y se trabaja con la versión 6.5.1 [8]. Para configurarlos lo más importante son los parámetros (Figura 4.10):

-"PAN ID" o identificador de red, que se puede poner el valor que se quiera.

-Coordinador activado, que simplemente servirá si queremos que un Xbee sea coordinador.

-Número de serie (alto y bajo), identifica al dispositivo que se va a configurar.

-Dirección de destino con el dispositivo que queremos comunicarnos (alto y bajo), si se identifica con un 0, solo se puede comunicar con el coordinador y si se quiere conectar con otro dispositivo, se deberá de poner el número de serie del mismo. Si lo que se quiere es enviar un mensaje de broadcast se pondrá en la dirección de destino bajo un 0 y en la dirección de destino alto FFF, con esto llegará el mensaje a todos los equipos activos que estén en la red.

-API activado, por defecto viene en modo transparente para comunicarnos por comando AT y si se activa se habilita la opción de enviar los datos por medio de tramas. Para este trabajo, se habilitará la opción de API.

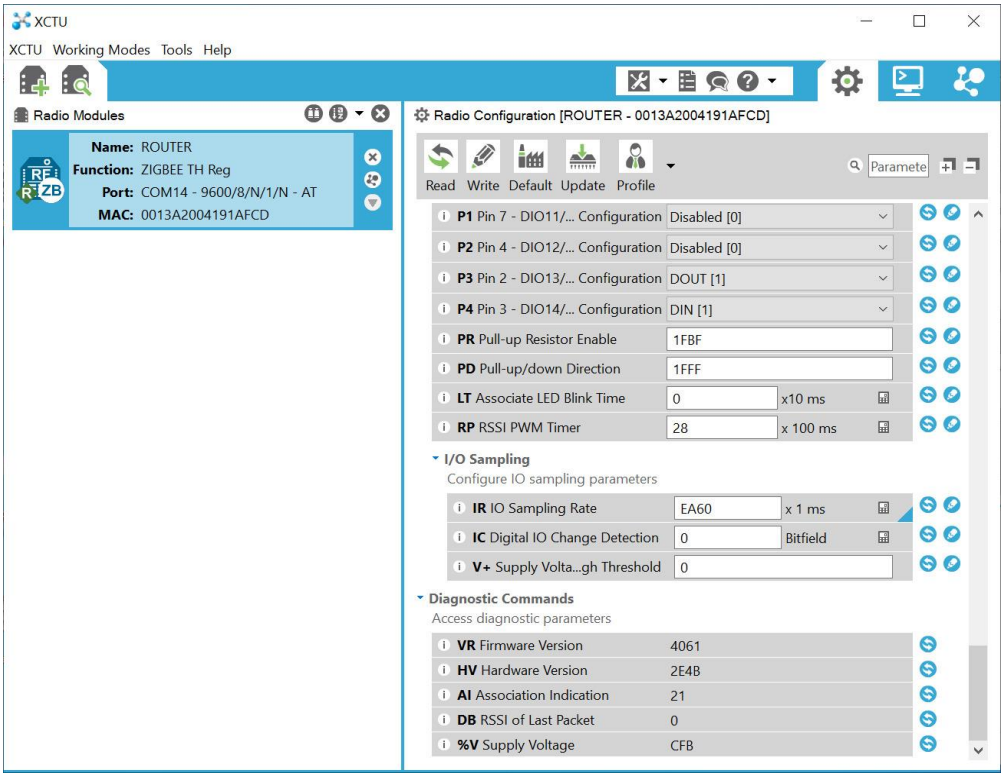


Figura 4.10 Configuración software del Xbee

4.3.3.3 Configuración Xbee

Los parámetros que configurar en los Xbee son:

Coordinador:

PAN ID [SIEMPRE EL MISMO]

DL FFFF

NI Coordinador

CE Enable

Router

PAN ID [SIEMPRE EL MISMO]

JV Enable

CE Disable

NI Router

Además, hay que habilitar el pin a muestrear donde se ha conectado el sensor y establecer el tiempo entre muestras.

También se ha habilitado la seguridad por defecto en todos los dispositivos.

Haciendo esto para agregar un nuevo dispositivo el sistema ESP32 reconocería automáticamente el ID del dispositivo Xbee y lo podría enviar al cloud quedando identificado de manera automática.

4.3.4 Problemas de conexionado

El planteamiento inicial de este trabajo era tener dos pasarelas para el envío de datos, por un lado, conectar el ESP32 con el sensor DHT11 y por otro conectar los Xbee con Arduino Uno (Figura 4.11) y este con su adaptador wifi que actuará de pasarela, para conectar con el servidor, pero al leer toda la documentación se observó que existía un problema de incompatibilidad entre los requisitos de seguridad en las conexiones impuestas por Google Cloud y Arduino Uno. Para solventar este problema se decide utilizar el dispositivo ESP32 como única pasarela para el envío de datos. Google Cloud no admite conexiones que no sean SSL, es un requisito expreso de características mínimas de Google Cloud y por esto se decide utilizar este microcontrolador.

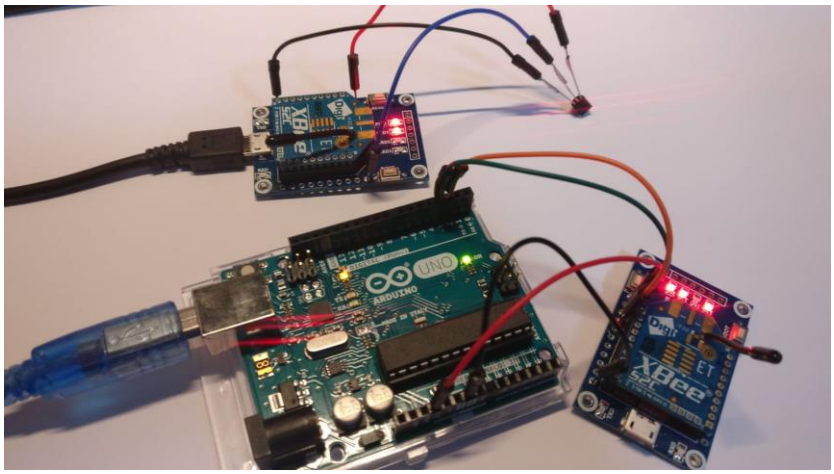


Figura 4.11 Primera conexión con Arduino Uno

Los microcontroladores que se podrían utilizar son el MKR1000 de arduino, Arduino UNO Wifi rev2 que lleva integrado un ESP8266, el ESP8266 propiamente y el ESP32. El MKR1000 al no disponer de él se deshecha la opción y entre el ESP8266 y ESP32 Se decide utilizar al ESP32 ya que ofrece mejores prestaciones.

En particular se emplea para la conexión del Xbee un puerto serie adicional del microcontrolador ESP32, aunque casi cualquier par de pines libres en este dispositivo podrían ser implementados como puerto serie.

En este caso se emplea el referenciado por el fabricante como UART 2 Tx UART 2 Rx mapeándolo a los pines 34 y 35 que en este caso no está en uso.

La implementación de la comunicación Xbee ha sido desarrollada de manera modular, esto ha permitido moverse de manera rápida hacia otra solución sin que afecte al desarrollo que había sido realizado para la comunicación entre dispositivos bajo la implementación del protocolo Zigbee. De esta manera el nodo central que desarrolla la función de coordinador en modo API, es conectada a su salida de datos, recibido por la red Zigbee, por puerto serie al ESP32 en lugar del Arduino Uno como estaba previsto.

La prueba de conexión final puede verse en las Figuras 4.12 y 4.13 que se encuentran a continuación:

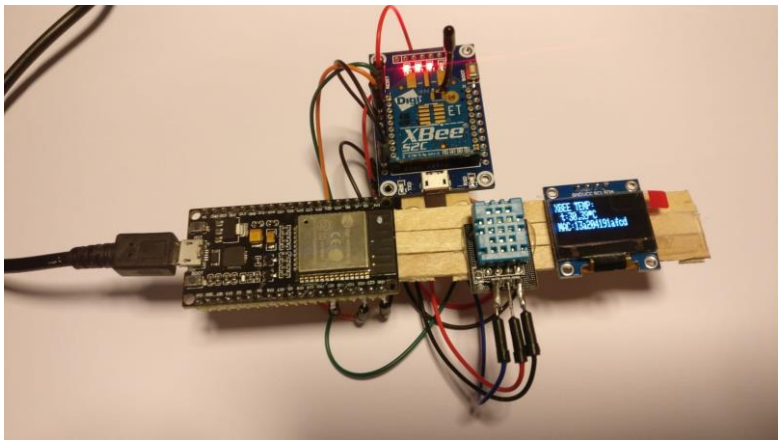


Figura 4.12 Conexión Xbee con ESP32

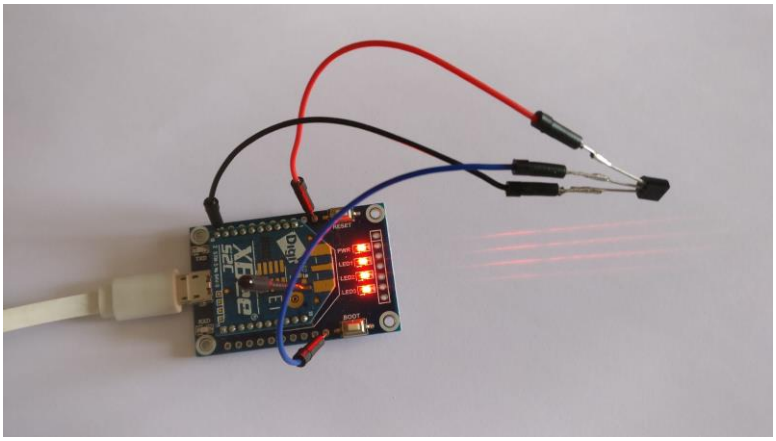


Figura 4.13 Conexión Xbee con TMP36

Esta conexión podrá ser realizada con cualquier ESP32, ya sea de manera dedicada para este propósito o encargada de realizar muestreos adicionales con otros sensores DHT11 o de otro tipo si fuera demandado.

Asimismo, para identificar a los dispositivos se referenciarán utilizando sus direcciones MAC, las cuales deben ser únicas de manera global por requisitos de fabricación. Aumentan los requisitos de tamaño de direccionamiento, pero soluciona colisiones si se quisiera crecer de forma global como se plantea. Así quedarán identificados los dispositivos de manera univoca.

La dirección MAC de los dispositivos Xbee responde a 16 dígitos hexadecimales.

4.3.5 Mantenimiento hardware

El hardware es completamente silencioso en su funcionamiento, es decir no existe ningún indicador externo de que este realizando su trabajo correctamente o no. Por esto y dado que existen pines suficientes de entrada y salida en el microcontrolador ESP32, a

modo de depuración se ha incorporado principalmente para el equipo que ostenta la tarea de coordinador una pantalla OLED la cual se conecta usando I2C (Figura 4.14, 4.15 y 4.16).

Esta pantalla permite la visualización en cuatro líneas desde gráficos, hasta las lecturas que va recibiendo de los distintos dispositivos Xbee, así como muestreo propio, también informa de la intensidad de la señal WiFi recibida.

La incorporación de este elemento supone como daño colateral el acortamiento de la batería puesto que es un elemento adicional consumiendo energía, es por esto por lo que se contempla como una herramienta que asiste a la implantación inicial o a la depuración de errores de un sistema en producción, pero no se contempla como un elemento a incorporar a ciclo permanente cuando el ahorro energético sea un aspecto crucial. De ser imprescindible dotar de pantalla de monitorización de funcionamiento al sistema y mantener a su vez el consumo al mínimo, existen disponibles pantallas de tinta electrónica, las que el hecho de mantener los elementos visualizados supone un consumo ínfimo y el consumo lo realiza al actualizar la información mostrada. Por contra el coste es, en el mejor de los casos, más de diez veces superior.

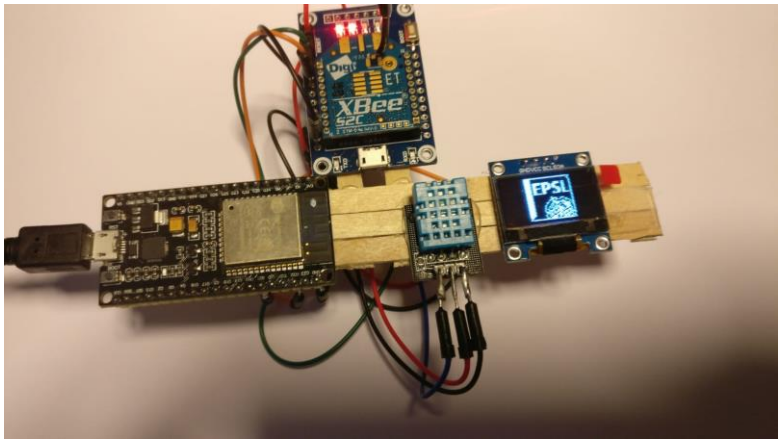


Figura 4.14 Ensamblado de pantalla Oled

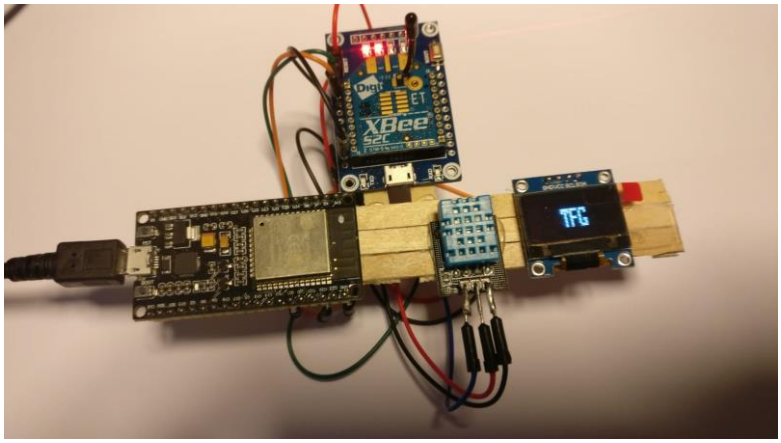


Figura 4.15 Ensamblado de pantalla Oled

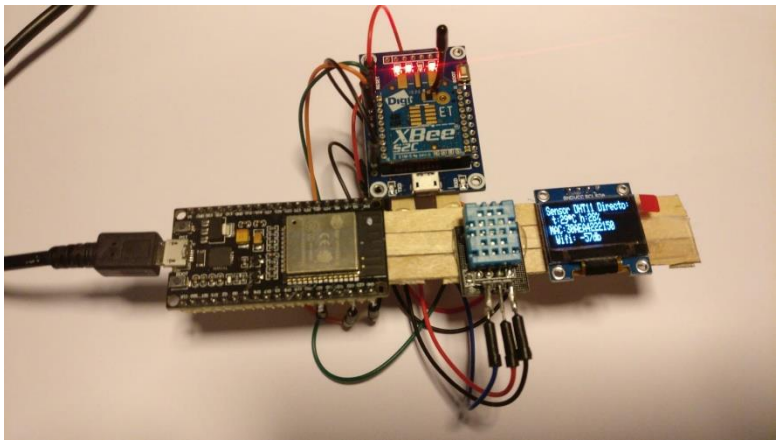


Figura 4.16 Visionado en pantalla de los datos de los sensores

Los nodos según su implementación permiten la conexión de sensores adicionales, ya sean de tipo analógico como digital, así mismo es posible implementar la activación de algún dispositivo a través del mismo nodo. Cualquiera de estas modificaciones, requerirán que sea revisado el tiempo entre mantenimientos de la batería del nodo, por el incremento de consumo que ello supone, o bien el redimensionamiento de la batería del nodo.

Para ello se considera el consumo medio de energía de todos los nodos, sensores incluidos, por cada tipo de hardware empleado.

Se debe considerar también el factor de auto-descarga de la batería empleada, y el rango de temperaturas a las que este va a estar expuesto durante su funcionamiento, puesto que temperaturas excesivamente bajas, impiden el funcionamiento de las baterías y temperaturas excesivas pueden dañar las baterías. En estos casos deben ser dotados los nodos de aislamiento adecuado o bien de ventilación.

4.4 Conectividad

En este apartado se explica cómo conectar la parte hardware y como se comunican los diferentes sensores con el microcontrolador que actuará de pasarela para el envío de datos.

El sistema planteado en el presente proyecto emplea dos tipos de redes inalámbricas diferentes que se detallan a continuación, por un lado, una red ZigBee y por otro lado una red WiFi.

4.4.1 Red ZigBee

Teniendo en cuenta que es necesario economizar energía, para conseguir el bajo mantenimiento solicitado a nivel de hardware, se orienta principalmente la tecnología de comunicación de uno de los nodos a una de las definidas en 802.15.4. Para este caso en particular se selecciona ZigBee [6] puesto que proporciona la flexibilidad requerida para el trabajo gracias a su posibilidad de despliegue en sistema mesh.

La tecnología mesh que implementa el protocolo ZigBee se basa en tres configuraciones distintas o roles que desempeñan los dispositivos conectados a la red. El dispositivo central o controlador, es el encargado de establecer los parámetros de configuración de la red, generación y difusión de la clave de encriptado, funcionalidad que el protocolo gestiona de manera transparente, el registro de nuevos dispositivos y el encaminamiento de todo el tráfico.

Por otro lado, están los dispositivos que realizan la función de router, estos envían el tráfico que generan al controlador, pero a su vez encaminan el tráfico generado por otros dispositivos hacia el controlador cuando estos no lo tienen a su alcance, esta funcionalidad permite incrementar de manera considerable el alcance de la red con una relación coste eficacia optima. La tecnología establece un límite de 15 reenvíos antes de alcanzar el controlador de la red.

Finalmente, un dispositivo puede tener el rol de dispositivo final, como ventaja en este caso se puede implementar una función de ahorro de energía pasando al dispositivo a modo sueño cuando no tiene que estar transmitiendo o procesando peticiones propias. En su contra este método no permite extender la cobertura de la red, ya que el nodo no realiza función de encaminamiento en este modo de configuración

Una red debe contener solo un controlador, pero es imprescindible que algún dispositivo este configurado en este modo. Sin embargo, no es necesario que algún dispositivo sea configurado como dispositivo final.

Para el funcionamiento, es imprescindible una conexión a internet, esta puede ser fija, por cable, o bien nómada basada en LTE o equivalente, teniendo en cuenta el consumo de datos.

No obstante, la información que se envía está dividida en múltiples paquetes con un mínimo de carga válida para transportar, es por esto por lo que el peso del empaquetado de los paquetes es necesario considerarlo.

Para paliar la sobrecarga que supone el encabezado tradicional de TCP o UDP siendo en el mejor de los casos superior a 20 Bytes, para la implementación del presente proyecto se selecciona MQTT.

Topologías

Soporta 3 tipos de redes (Figura 4.17): Inicial, árbol y mesh [7]

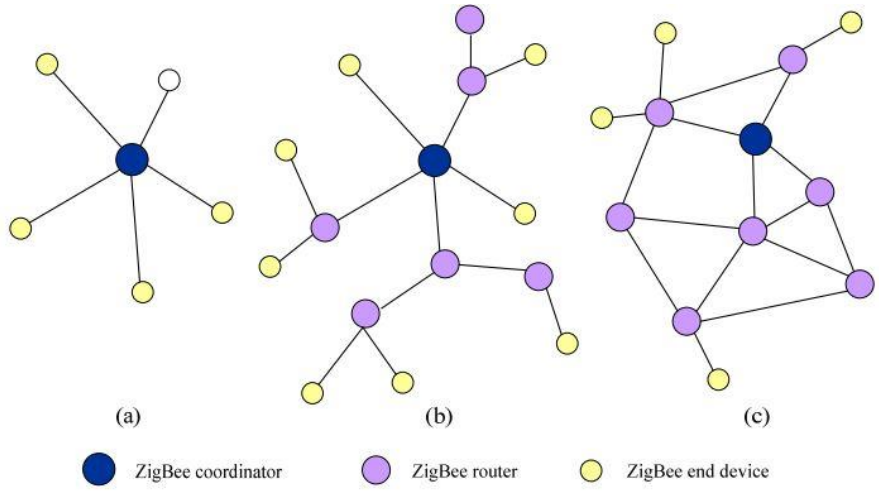


Figura 4.17 Tipos de redes - Fuente: http://ocw.nctu.edu.tw/course/ws992/E1.pdf

La topología que se utiliza en este trabajo es mesh.

4.4.2 Red Wifi

Se establece un segundo tipo de nodo que integra una pasarela directa contra el router por medio de la cobertura wifi, a este nodo en caso de que en alguna de las zonas de despliegue lo requiera por requisitos de posicionamiento del controlador ZigBee, puede ser conectado el dispositivo ZigBee. Para la implementación de este tipo de nodos se ha seleccionado el hardware ESP32, por sus características de eficiencia energética y rendimiento. Este tipo de encaminador aporta un valor añadido puesto que existe la posibilidad de implementar una red de ESP32 en modo mesh igual que con el protocolo ZigBee, pero sobre wifi, de este modo se amplía aún más la posibilidad de integrar en la red otro tipo de sensores que requieran una capacidad o necesidad de reprocesado de la lectura realizada.

En este caso los dispositivos ESP32 se conectan directamente por Wifi al router y envían al Cloud de Google empleando MQTT bajo identificación HTTPS con certificado digital auto-firmado, las lecturas realizadas a los sensores digitales de temperatura y humedad DHT11, cubriendo así todos los requisitos solicitados. A este tipo de dispositivo también cabe la posibilidad de conectar sensores adicionales o algún otro tipo de dispositivo de actuación o monitorización, como pudiera ser una pantalla para auditar la información procesada.

En este esquema se muestra el conexionado de todos los equipos hardware utilizados en este trabajo, así como las conexiones y tecnologías utilizadas que hacen posible el envío de datos por parte de los sensores hacia la plataforma de Google Cloud.

En el siguiente apartado (4.3) se pasa a detallar los métodos empleados para almacenar la información, la base de datos estructurada MySQL, los atributos y sus características para la definición de la estructura de datos contenedora de la información para su transferencia DTO, así como el formato, JSON, empleado para transferir la información entre las partes del proyecto y el protocolo de transferencia empleado MQTT.

4.5 Transmisión de información y Almacenamiento

En este apartado se detalla el funcionamiento del dispositivo ESP32 que actúa como pasarela (Gateway), envía los datos procedentes de los diferentes sensores a la plataforma de Google Cloud.

Los sensores que se han utilizado en este trabajo son el DHT11 y el TMP36. Se leen los datos, uno digital y otro analógico y se compone una cadena JSON (Javascript Objet Notation) para enviar los datos por MQTT a Google Cloud. Se empieza explicando cómo se obtienen las muestras de datos y se procede a la explicación de forma más amplia de los formatos y protocolos empleados, en último lugar (el siguiente punto 4.4) pasa a explicar la configuración de los servicios Cloud de Google implicados.

Muestreo

Las muestras de los sensores utilizados son tomadas y enviadas a registro por MQTT al menos una vez cada minuto. El dispositivo que hace la función de Gateway se conecta al Google Cloud empleando el servicio Cloud IoT Core, y usando identificación segura mediante certificado digital autofimado SSL. Este certificado es necesario registrarlo identificando el dispositivo en el Cloud e instalarlo también en el código del dispositivo ESP32 que vaya a realizar la función de pasarela. Cada dispositivo debe tener un certificado independiente. Posteriormente todos podrían estar enviando sus datos al mismo topic de MQTT o bien usar uno diferente para clasificar por dispositivos, no obstante, el sistema está desarrollado para hacer más ágil el despliegue por lo que no es necesario emplear distintos topic de publicación ya que las tramas enviadas contienen la identificación completa del origen donde se generaron, así como del dispositivo que se encargó de registrarlas en el sistema.

Como se ha explicado anteriormente, este trabajo presenta un sensor conectado directamente a la pasarela de envido de datos y solo tiene una configuración, pero para

extender la red de sensores si fuese necesario se han utilizado dispositivos Xbee, pues para las muestras de estos dispositivos, se ha propuesto un dispositivo que realiza la función de coordinador, este debe ser único, ningún otro debe ser configurado en este modo. El coordinador no va a realizar en este caso muestreo de sensores, solamente va a encaminar todas las muestras recibidas del resto de dispositivos Xbee configurados tanto como elementos finales como routers.

La configuración de dispositivos Xbee como routers permite expandir la cobertura de la red, pero aumenta su consumo. En principio los dispositivos Xbee solo muestrean un sensor de temperatura analógico TMP36. El intervalo de muestras se establece cada 60 segundos.

4.5.1 Estructura de almacenamiento - DTO (Data Transfer Object)

Para dar cabida al sistema que se quiere implementar, se diseña la siguiente estructura para el almacenamiento de la información.

Identificador de Zona

FFFFFF: 6 dígitos en hexadecimal

Identificador de Gateway

FFFFFFFFFFFF: 12 dígitos en hexadecimal, donde se especifica la MAC del Gateway

Identificador de Dispositivo

FFFFFFFFFFFFFFFF: 16 dígitos en hexadecimal, donde se especifica la MAC del dispositivo (Xbee es mayor a ESP32)

Lectura

IDZona: el id de la Zona

IDGW: id del Gateway

IDDV: id del Dispositivo

timestamp: marca de tiempo

temp: temperatura

hum: humedad

El direccionamiento de 6 dígitos hexadecimales en el id de zona, proporciona una capacidad de registro de 16.777.216 unidades. Se reservan 0 y la última para funciones específicas de test y administración.

Posibilita la monitorización y expansión más amplia evitando a su vez problemas por reutilización de identificadores por avería de dispositivos o limitaciones de movilidad de sensores en escenarios de despliegue temporales que supongan finalmente la duplicidad de ids en casos en los que se compare el escenario en producción con el histórico de datos. Esto provocaría que los datos se corrompieran ya que un dispositivo nuevo reutilizaría el id de un dispositivo que fue eliminado en el tiempo o sustituido, confundiendo las lecturas en el histórico.

El empleo de este tipo de id, posibilita la vinculación con el identificador físico de red de los dispositivos, el cual debe ser único, y facilita el mantenimiento en un escenario creciente de esas magnitudes.

4.5.2 JSON – JavaScript Object Notation

JSON es un formato que almacena información estructurada y se utiliza para transferir datos entre un cliente y un servidor. Este es el formato que se ha empleado para la transferencia de datos entre el ESP32 y Google Cloud.

El objeto JSON tiene dos elementos principales, una de ella es la clave y la otra el valor. Las claves son cadenas de caracteres (string) que están delimitadas por unas comillas y los valores son objetos de tipo string, booleano, número, objeto, array o nulo.

Los objetos comienzan con una llave "{" y terminan con otra llave "}" pudiendo tener entre ellas varios pares de claves y valores, separados por una coma, y después de cada clave se especifica el valor separados por dos puntos. [9]

Ej:

{"temp":3200, "hum":3500}

TIPOS DE VALORES

String: Cadena de caracteres Unicode y van entre comillas dobles

Array: también llamado vector y es una lista ordenada de valores separadas por comas, y van entre corchetes

Número: Podría ser un número entero o un punto flotante

Booleano: los valores posibles serán verdadero o falso

Nulo: para indicar que no tiene valor

En los años 90 XML era el formato más utilizado para el intercambio de información, pero había problemas de lentitud a la hora de enviar un fichero de gran volumen, por esto

surgieron otros formatos para el intercambio de información más rápidos y ligeros y entre ellos apareció JSON.

El formato JSON es el empleado para formatear los datos de la carga válida de los mensajes MQTT.

JSON simplifica la tarea de intercambiar datos y es por esto por lo que también se ha usado en este trabajo para la salida de datos del back-end realizado

4.5.3 MQTT

4.5.3.1 MQTT – PROTOCOLO DE TRANSFERENCIA EMPLEADO

MQTT (MQ Telemetry Transport), fue diseñado inicialmente para conectar dispositivos utilizados en la industria petrolera en el año 1999 por Andy Stanford-Clark y Arlen Nipper. En un primer momento fue un formato en propiedad, pero en el año 2010 se consiguió liberar y ya en el año 2014 la OASIS (Organization for the Advancement of Structured Information Standards) lo consideró como un formato estándar. [10]

Es un protocolo publish/subscribe de mensajes entre cliente y servidor que trabaja sobre TCP. Es ligero y fácil de implementar, el tamaño de un paquete de este protocolo oscila entre 12 Bytes y 7 Bytes. Está pensado para dispositivos de baja potencia como los utilizados en IoT, que son los que se están utilizando en este TFG.

Debido a que utiliza poco ancho de banda el protocolo MQTT está indicado para comunicar sensores, además se puede usar mayormente con dispositivos que requieren o necesitan pocos recursos. El modelo de arquitectura de este protocolo utiliza una topología en estrella, basándose en un servidor o “bróker” que actúa de nodo central. Este “bróker” mantiene la comunicación activa ya que se encarga de la gestión de la red y de transmitir todos los mensajes.

Para la comunicación, el cliente que quiere publicar un mensaje crea unos “topics” o unos temas y los equipos que quieran recibir estos mensajes debe de subscribirse a estos “topics”. Esta comunicación podría ser bien de uno a uno o bien de uno a muchos. Para representar un “topic” se realiza a través de una cadena teniendo esta una estructura jerárquica y siendo separada por el carácter “/” para cada jerarquía.

El protocolo MQTT está basado en TCP/IP (Figura 4.18) por lo que, tanto el cliente como el servidor o broker necesitan tener una pila con estas características:

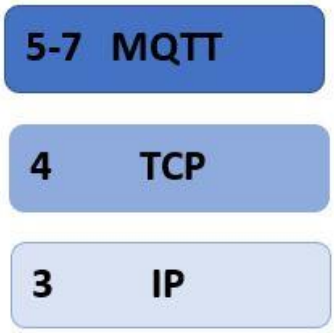


Figura 4.18 Torre protocolos MQTT

4.5.3.2 Estructura de un mensaje MQTT

La definición y el tipo de mensaje, es de lo más importante en el protocolo MQTT, ya que es lo que le hace ser rápido. Cada mensaje tiene 3 fracciones como puede observarse en la siguiente imagen (Figura 4.19):

Fija Siempre		Opcional	Opcional
Cabecera fija		Cabecera opcional	Contenido del mensaje
Código Control	Longitud Mensaje		
1 byte	1-4 bytes	0-X bytes	0-256Mb

Figura 4.19 Estructura mensaje MQTT

Cabecera MQTT

- **Cabecera fija**. Es obligatorio, y sirve para saber el tipo de mensaje y su longitud.
- **Cabecera variable**. Opcional, solo para algunos mensajes
- **Contenido (payload)**. Contenido del mensaje propiamente dicho. Aunque el tamaño puede ser hasta 256MB, el real no excede de 4Kb.

Mensajes y códigos de control del protocolo MQTT:

Mensaje	Código
CONNECT	0x10
CONNACK	0x20
PUBLISH	0x30
PUBACK	0x40
PUBREC	0x50
PUBREL	0x60
PUBCOMP	0x70
SUSCRIBE	0x80
SUBACK	0x90
UNSUSCRIBE	0xA0
UNSUBACK	0xB0
PINGREQ	0xC=
PINGRESP	0xD0
DISCONNECT	0xE0

Tabla 4-1 Mensajes MQTT [10]

4.5.3.3 Seguridad en MQTT

Unos de los factores más importantes en un sistema de comunicación M2M es la seguridad y este protocolo también dispone de ellas para poder proteger toda comunicación.

Para el transporte de datos se utiliza SSL/TLS y autentificación por usuario y contraseña o certificado. Alguna vez la autentificación se realiza enviando en texto plano un usuario y una contraseña.

Es importante conocer los riesgos de utilizar uno u otro para que el sistema que se implemente en MQTT sea eficiente.

4.5.4 MySQL - Almacenamiento de datos

MySQL pertenece a la empresa Oracle Corporation y es un sistema de administración de bases de datos relacional basada en código abierto, aunque Oracle

también gestiona una versión comercial. Es un programa que puede almacenar grandes cantidades de diferentes datos y repartirlos para satisfacer las posibles necesidades que pudiera tener cualquier organización [19]. Contiene elementos para realizar la protección del sistema, así como copias de seguridad, volcado de datos y crear diferentes niveles de acceso para los usuarios. Se puede utilizar en una gran mayoría de lenguajes de programación y ejecutarse en muchos de los sistemas operativos disponibles hoy día. MySQL maneja SQL como lenguaje de consulta estructurado, está preparado para bases de datos relacionales, donde se puede crear, insertar, borrar datos en cualquier base de datos en función de unos criterios.

4.5.4.1 Características MySQL

Algunas características de MySQL son:

Arquitectura Cliente y Servidor: MySQL funciona con el modelo cliente y servidor. La comunicación es diferencia entre cada cliente y cada servidor para tener un mejor rendimiento. Cada cliente puede realizar las consultas que desee de forma independientemente.

Compatibilidad con SQL: Como lenguaje de consulta estructurado.

Escalabilidad. Puede soportar hasta 50 millones de registros con 200.000 tablas.

Conectividad. Utilizando socket TCP/IP, un cliente de cualquier plataforma podría conectarse.

Localización: Se pueden enviar mensajes en muchos idiomas desde el servidor.

Clientes y herramientas: MySQL incluye algunos programas de utilidad tanto en líneas de comandos, como formato gráfico y uno de los más utilizados es el MySQL Workbench.

4.5.4.2 Principales Sentencias MySQL

MySQL utiliza sentencias del lenguaje SQL, las más básicas son las siguientes:

Select: Usada para consultar datos.

Distinct: Utilizada para eliminar duplicados en una consulta de datos.

Where: Para poner condiciones en los datos consultados.

Order By: Ordena los datos obtenidos de una consulta.

Insert: Para insertar datos.

Update: Sirve para actualizar datos.

Delete: Usada para borrar datos.

4.6 Servicios Google Cloud Platform

El almacenamiento y procesamiento de la información en este proyecto se realiza íntegramente en servicios Cloud Computing, en este caso desplegados en Google Cloud. A continuación, se enuncian los servicios empleados, y en los distintos apartados referidos a cada servicio en concreto se detalla la configuración del mismo. Se tendrá presente el esquema presentado inicialmente donde se establece la relación entre el flujo de datos y los servicios en el orden en el que son utilizados durante un ciclo de ejecución.

De este modo, se inicia el proceso con **IoT Core**, que dispone de los dispositivos identificados y realiza el registro de los mismos, a su vez este hace de servidor **Pub/Sub** donde los dispositivos suben los muestreos, para ser procesado por **Cloud Function** como subscriptor al servicio anterior y hacer la inserción de datos en el almacén de datos **Cloud SQL** sobre una base de datos **MySQL**, paralelamente al servicio **Pub/Sub** es también consumido por **DataFlow** que prepara los datos para ser consumidos a través del servicio **BigQuery**. En último lugar están los servicios de visualización **App Engine** que ejecuta una web dinámica para tomar los datos almacenados y ofrecerlos al usuario.

4.6.1 Comunicación con Google Cloud Platform

La mejor manera de conseguir que unos dispositivos que están situados en diferentes localizaciones puedan comunicarse entre ellos es realizar un servicio de notificaciones centralizado y sobre todo externo. En definitiva, tener un servidor central que reciba los datos de los dispositivos emisores y llegado el momento poder de distribuirlos a los receptores que los soliciten. En este trabajo, solo se dispone de una red de sensores que envían los datos al servidor para poder ser procesados por quien los solicite. Este servidor tiene el nombre de "Broker" y el que se va a utilizar para este trabajo es el Cloud IoT Core que ofrece Google Cloud Platform.

Unas de las metodologías más utilizadas para esta recepción y envío de notificaciones, mensajes o datos es la Publish/Suscribe (Pub/Sub), donde el Publish puede publicar mensajes, y el Suscribe puede recibirlos siempre que este suscrito. Será el Servidor el encargado de distribuir estos mensajes. Esta metodología es utilizada por el protocolo MQTT.

Otra de las metodologías que se puede utilizar es la de cliente servidor o petición-respuesta que utiliza el protocolo HTTP. Este protocolo crea una conexión cuando necesita enviar una petición. La cabecera más pequeña de un paquete HTTP es de 26 Bytes.

MQTT presenta un ahorro considerable en el tráfico de datos procedente de IoT. Y es por ello por lo que se ha decantado para la realización de este TFG por el protocolo

MQTT que es el que mejor se adapta a esta metodología Pub/Sub, además de como se ha explicado anteriormente el tamaño del paquete es bastante menor.

4.6.2 Servicio - IoT Core

Cloud IoT Core es un servicio que permite ingresar datos de dispositivos que están deslocalizados [11]. Como se ha explicado anteriormente los clientes enviarán datos a un servidor, en adelante “bróker”. El "broker" que se utiliza para este TFG es el servidor IoT Core de Google Cloud Platform, ofreciendo el uso de MQTT para el intercambio de mensajes cifrados, así como el registro individual de los dispositivos que se vayan a usar. Los clientes son los sensores conectados a los microcontroladores que empleando librerías MQTT se conectarán al servidor.

El primer paso que se ha realizado en el servidor es registrar los dispositivos que se han utilizado, a través de los llamados "Registros de dispositivos" que estarán relacionados para poder compartir “topics”.

Se selecciona IoT Core y se accede a la página de inicio (Figura 4.20):

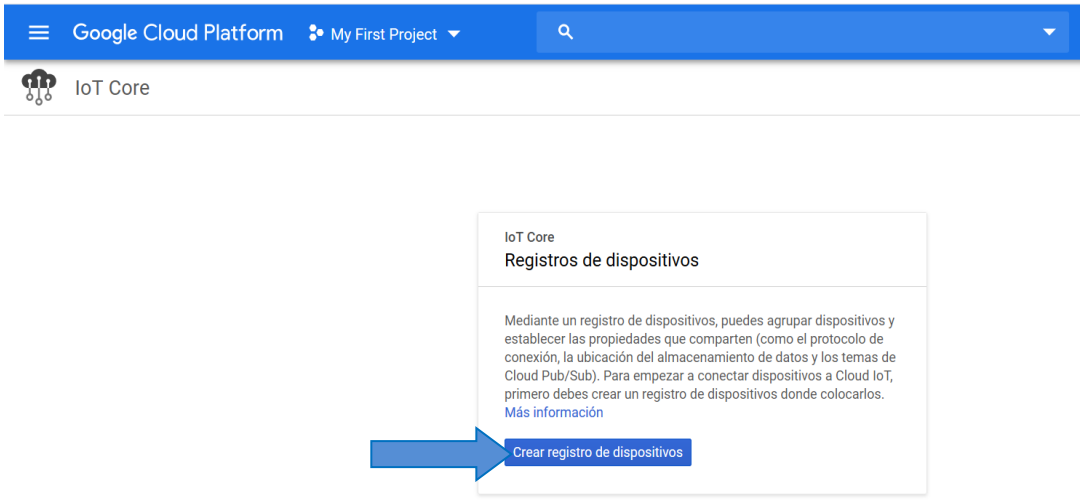


Figura 4.20 Registro de Dispositivos en el Cloud IoT Core

Se pulsa “Crear registro de dispositivos” y se completa con la siguiente información:

ID de registro: devices-environment

Región: europe-west1

Tema de Cloud Pub/Sub: environment

Protocolos: HTTP, MQTT

Tema de estados de dispositivos: environment

Después de completar todos los detalles, el nuevo registro de dispositivo estará creado y puede comprobarse en la Figura 4.21:

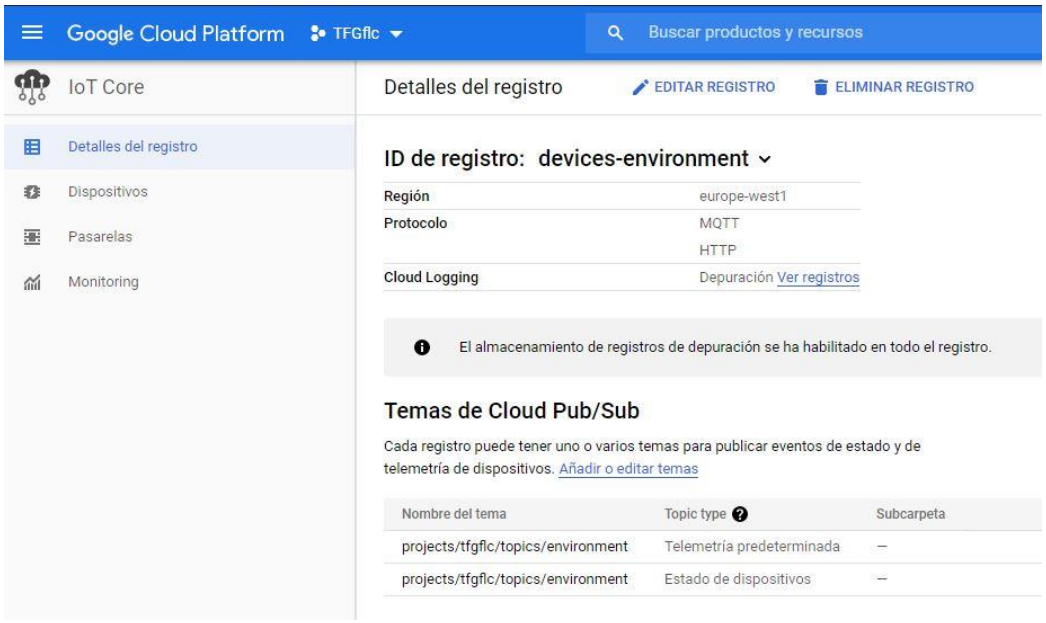


Figura 4.21 Registro creado en Cloud IoT Core

Para añadir dispositivos, se pulsa la opción “Dispositivos” de la figura anterior (Figura 4.21) y en el nuevo menú que aparece, se pulsaría en “+ CREATE A DEVICE”. (Figura 4.22)

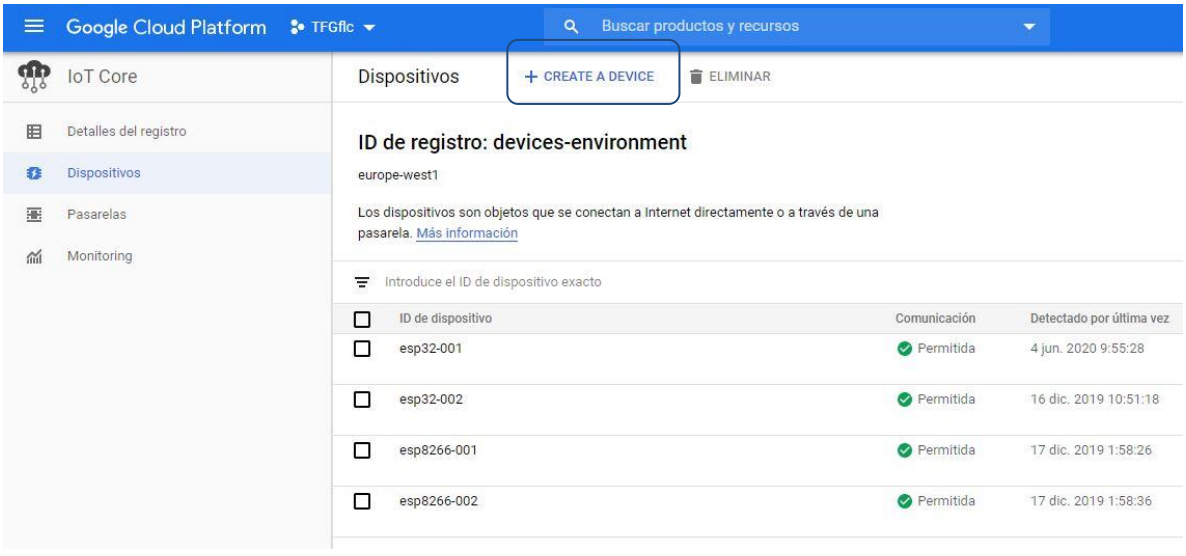


Figura 4.22 Añadir dispositivos en Cloud IoT Core

En la página del dispositivo, (Figura 4.23) se debería de ver un menú en el que se introducirán varios datos, como puede ser el nombre del dispositivo, el permiso para la comunicación de este y la autenticación.

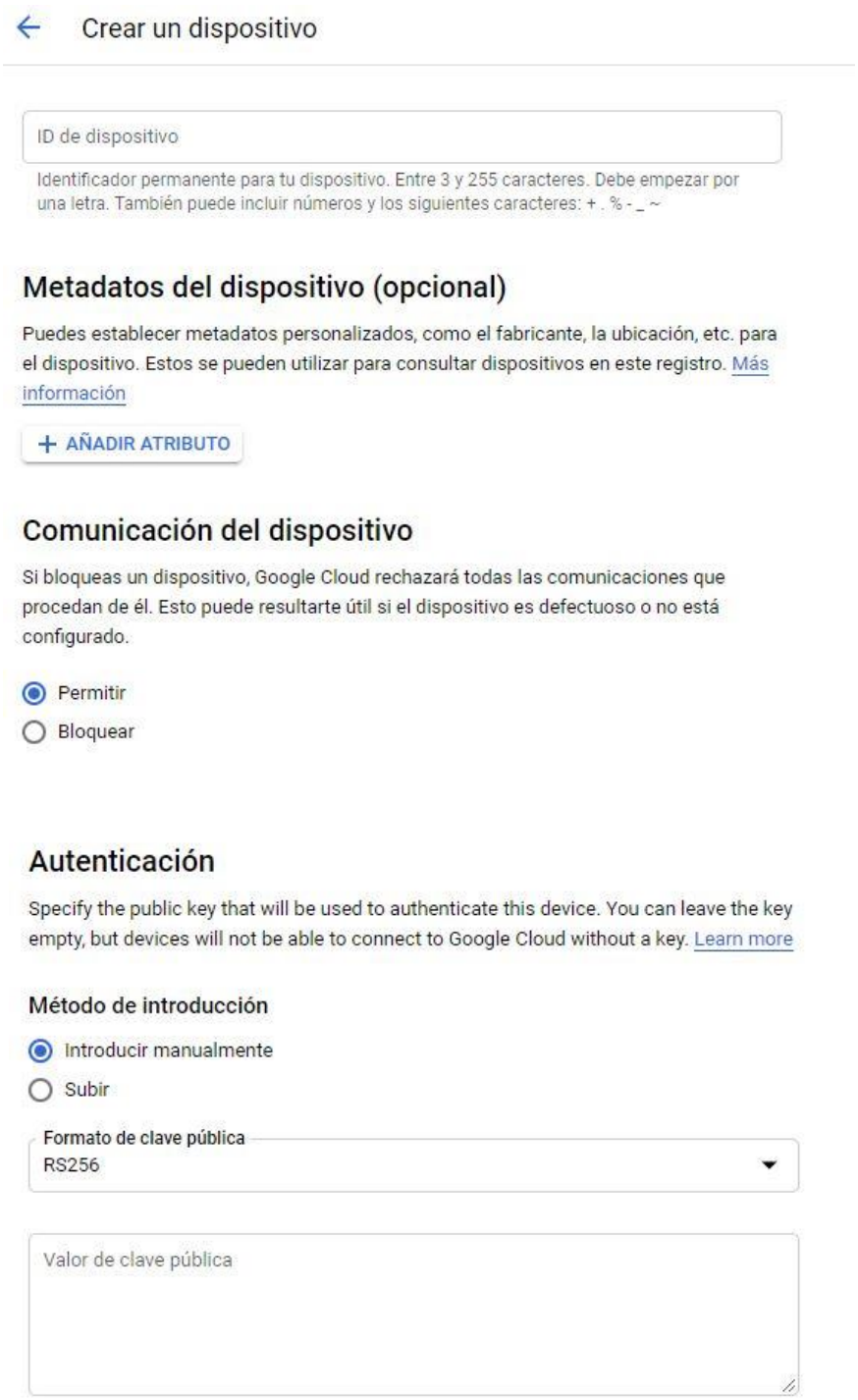


Figura 4.23 Alta de nuevo dispositivo en Cloud IoT Core

Ahora, se debe de agregar una nueva clave pública. Para generar un par de claves públicas/privada, se necesita tener disponible la línea de comandos del cloud Shell y usar el siguiente comando:

openssl req -x509 -newkey rsa:2048 -keyout rsa_private.pem -nodes -out rsa_cert.pem -subj "/CN=unused"

Una vez ejecutado ese comando se habrán creado dos archivos:

esp32-001-rsa_cert.pem, clave pública.

esp32-001-rsa_private.pem, clave privada.

Estos certificados no se compartirán, ya que de lo contrario cualquier persona que los consiga puede conectarse a Google IoT Core como dispositivo y comenzar a publicar datos. Una vez creados, se adjuntan al dispositivo que se ha creado en IoT Core.

En la página del dispositivo se incorporará la clave pública, editándola y copiándola en la interfaz de la figura anterior. El formato de clave pública que se ha utilizado es RS256_X509. Además, se ha indicado una caducidad para esta clave pública.

Finalmente se pulsa sobre el botón de crear y una vez que la clave pública se agregó exitosamente, se puede conectar a la nube usando la clave privada. Una forma de descargarse la clave privada es desde el editor de código (Figura 4.24)

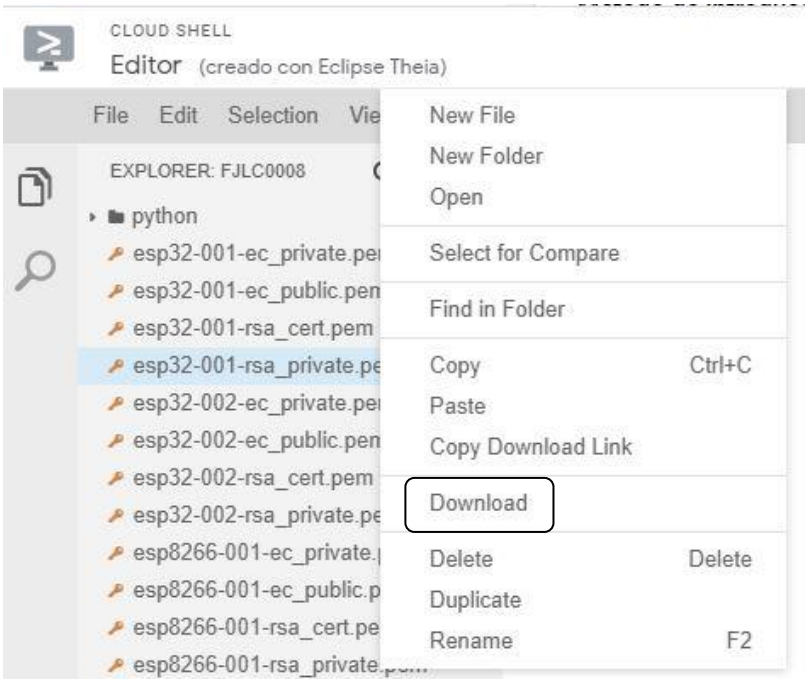


Figura 4.24 Descarga de la clave privada desde el Cloud Shell del IoT Core

Para probar que la configuración anterior ha sido correcta, se utiliza la red de sensores de este trabajo y desde el IDE de Arduino se cargará la librería “ciotc_config.h” donde se especifica los detalles de la conexión wifi, los detalles del Cloud IoT y los certificados de conexión.

Los detalles del cloud son los siguientes:

```
const char *ssid = "Redwf_1Y38";                    //red wifi
const char *password = "******************";    //clave de la red, está oculta
const char *project_id = "tfgflc";               //nombre del proyecto
const char *location = "europe-west1";     // localización
```

```
const char *registry_id = "devices-environment";     //registros de dispositivos
const char *device_id = "esp32-001";                  //id del dispositivo
const char *private_key_str =                         //claves
   "3*:**:**:**:**:";  (oculto)
const char *root_cert =
   "-----BEGIN CERTIFICATE-----\n"
"**********************************************************"(oculto)
"-----END CERTIFICATE-------\n";
```

El tipo de algoritmo es RS256: RSA signatura with SHA-256. Es un método de cifrado asimétrico muy eficiente. Este algoritmo primero calcula un hash único de los datos de entrada usando el algoritmo SHA256 y a continuación el hash se encripta con una clave privada utilizando el algoritmo RSA.

Para poder enviar datos en formato MQTT se ha utilizado la librería “esp32-mqtt.h”.

Ejecutando este código, se debe de haber conectado con IoT Core y estará listo para recibir los datos.

Para comprobar que el dispositivo se está comunicando correctamente con IoT Core se usa su sistema de logging. Para ello, se edita la configuración del dispositivo para establecer el valor de logging en “Depuración” (Figura 4.25 y 4.26). Esto permite visualizar todos los envíos de mensajes desde el dispositivo a la plataforma. Se consulta el intercambio de mensajes visualizando los registros del sistema de logging (Figura 4.27) y se verá con detalle como se aprecia en la siguiente imagen (Figura 4.28)

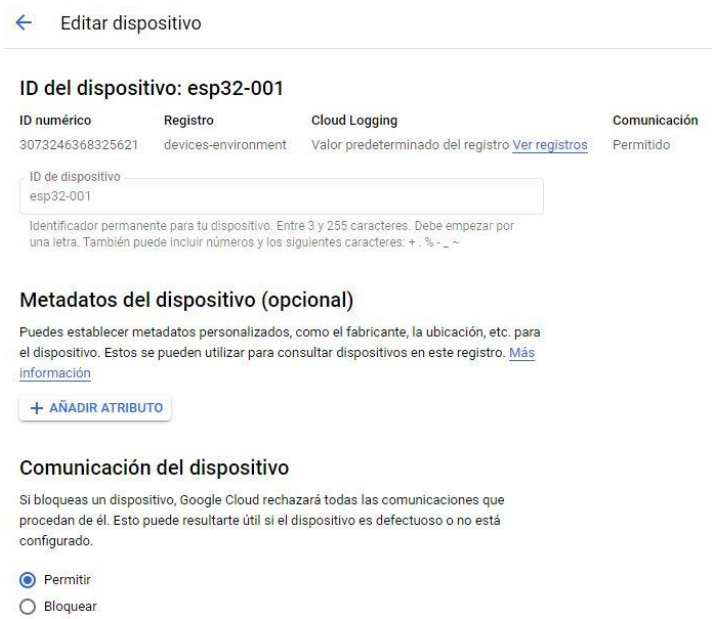


Figura 4.25 Activación de captura de actividad de los dispositivos en el IoT Core

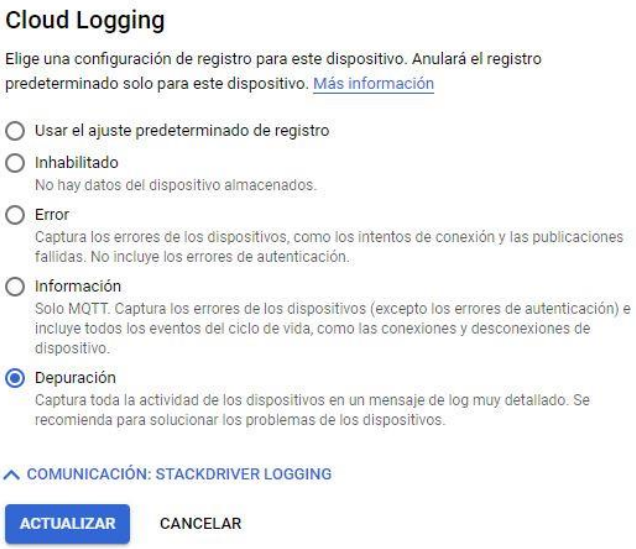


Figura 4.26 Activación de captura de actividad de los dispositivos en el IoT Core

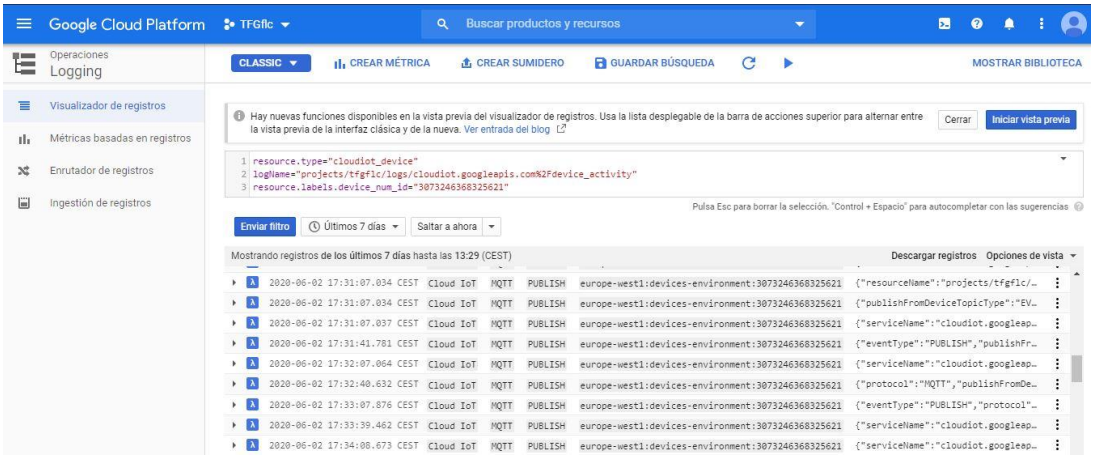


Figura 4.27 Registro de actividad del dispositivo en IoT Core

Figura 4.28 Detalle registro de actividad del dispositivo en IoT Core

4.6.3 Servicio - Cloud Function

Google Cloud Function es uno de los servicios de Google Cloud que sirve para crear aplicaciones sin servidores dentro de la propia infraestructura [12]. Una ventaja que presenta es que se paga por el tiempo que el código se esté ejecutando, de manera que es ideal para soluciones pequeñas. De este modo se puede dar respuesta a todas las necesidades de proyectos en el momento exacto sin necesidad de mantener una estructura mantenida en el tiempo.

En esta función se programa a partir de un evento, o sea, cada vez que en el IoT Core se genera una Pub/Sub, la función se ejecuta.

Se puede usar Cloud Functions combinado con Internet of Things y más servicios. Se puede procesar y analizar en tiempo real datos telemétricos de dispositivos IoT. La Figura 4.29 muestra un ejemplo donde un sensor adquiere continuamente la temperatura y envía mensajes utilizando la infraestructura IoT Core. A través del servicio Pub/Sub el programa escrito a través de Cloud Functions detecta que se ha alcanzado una temperatura máxima y envía un mensaje para que se ponga en marcha un ventilador que baje la temperatura. (Figura 4.29)

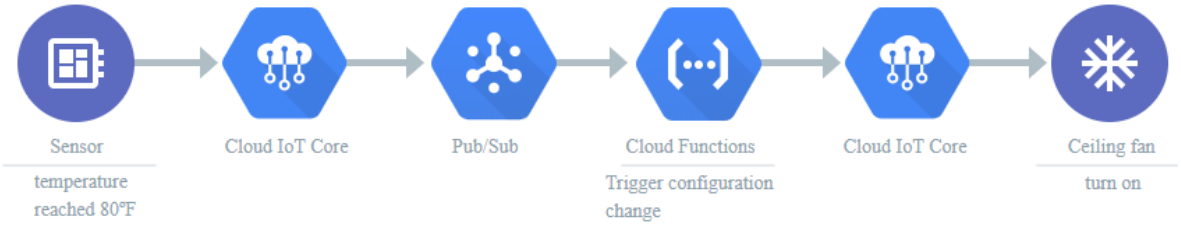


Figura 4.29 Cloud Functions combinado con IoT Core

Para implementar las Cloud Functions se accede desde el menú principal del Cloud y se pulsa en “Cloud Functions” (Figura 4.30)

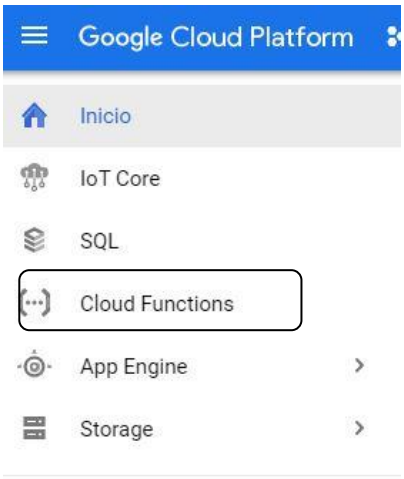


Figura 4.30 Acceso a Cloud Functions

Y se pulsa en “Crear función” (Figura 4.31)

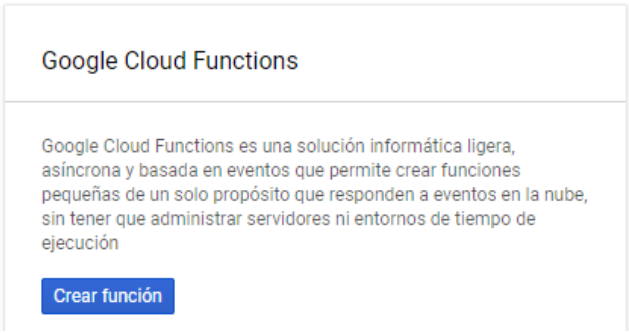


Figura 4.31 Crear nueva función en Cloud Functions

Se completa con la siguiente información:

Nombre: fdb

Memoria asignada: 128Mb

Activador: Cloud Pub/Sub

Tema de Cloud Pub/Sub: environment

Código fuente: Editor insertado

Tiempo de Ejecución: Python 3.7

Roll que ejecutar: addDatamysqldb_pubsub (nombre del método que se exportará)

Región: europe-west1

Después de completar todos los detalles, el formulario debe de tener el aspecto siguiente (Figura 4.32):

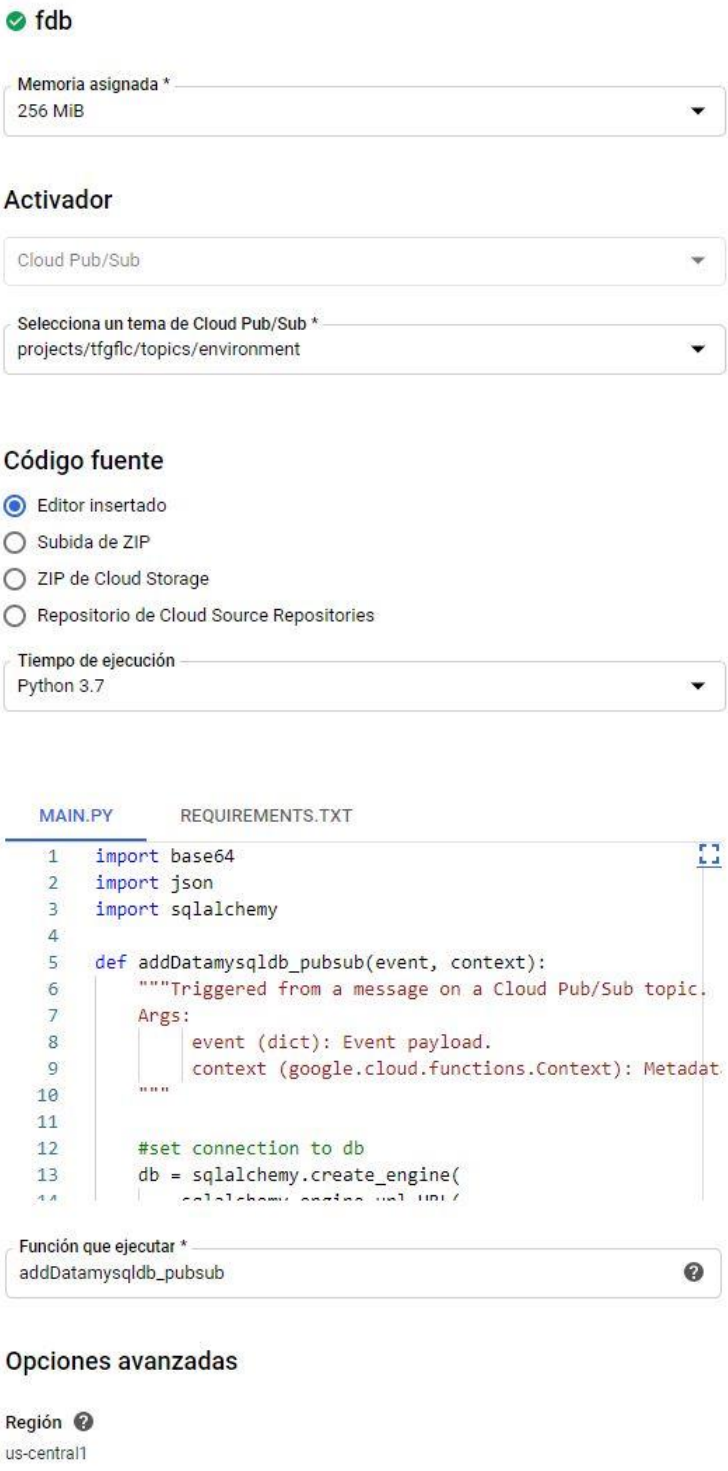


Figura 4.32 Aspecto función en Cloud Fuctions

Se pulsa sobre el botón de “Crear” y se tendrá la nueva función ya terminada. (Figura 4.33)

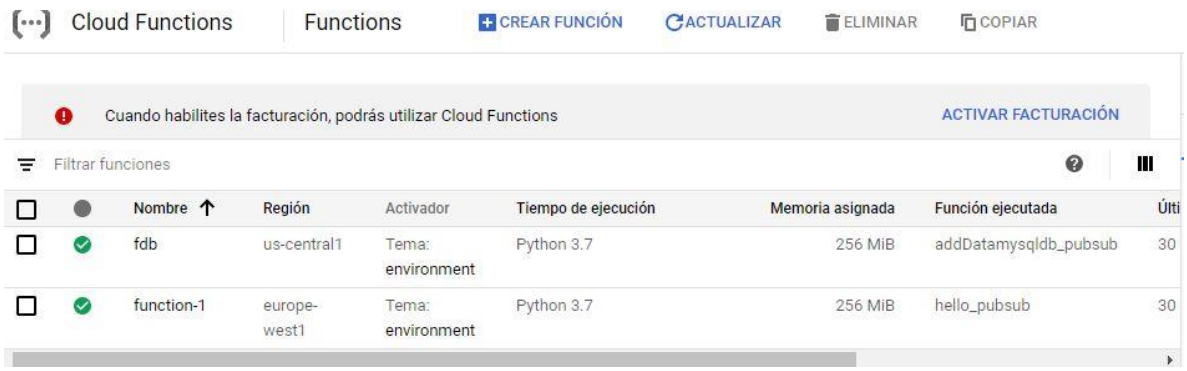


Figura 4.33 Función “fdb” creada en Cloud Functions

Si se quiere comprobar la actividad de la función, se lanza el cliente anteriormente diseñado y en el menú de Cloud Functions se elige la opción “Ver registros” (Figura 4.34)

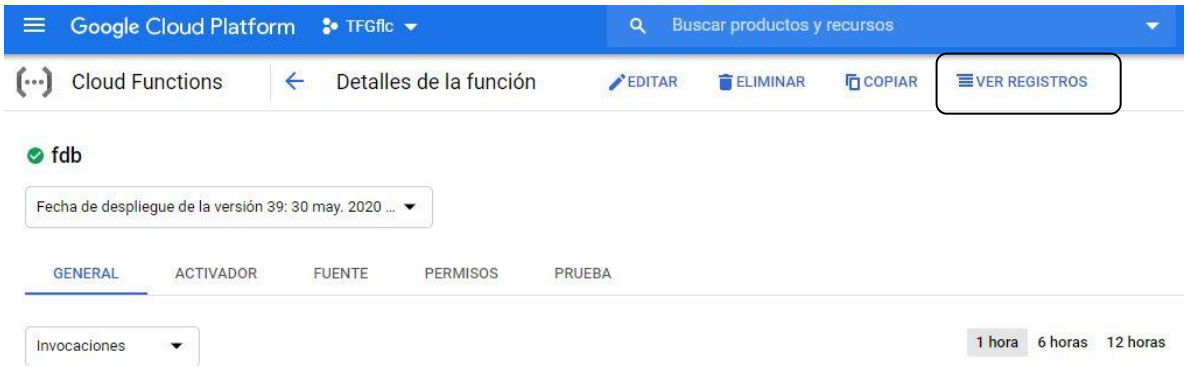


Figura 4.34 Visualización del registro de la función en Cloud Functions

Si todo va bien, se puede ver la actividad de la función, su ejecución y finalización. (Figura 4.35)

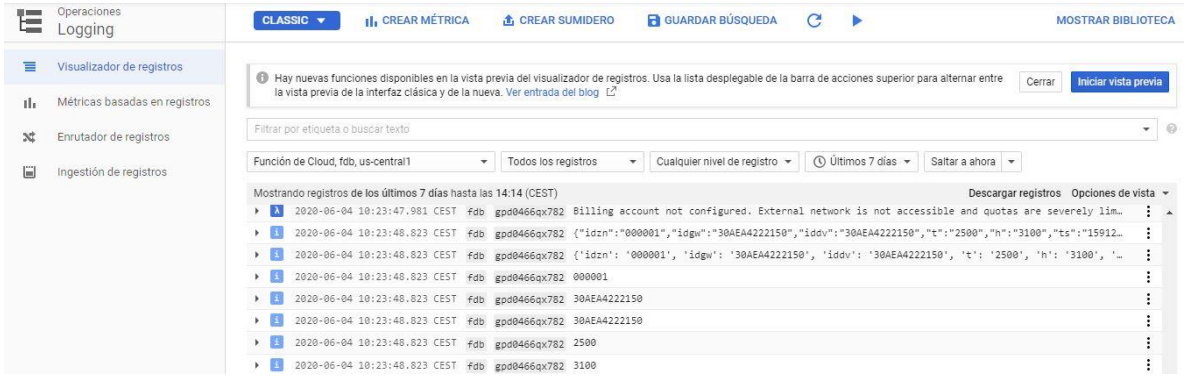


Figura 4.35 Comprobación de la actividad de la función en Cloud Functions

Una vez comprobado que la función funciona correctamente, se incorpora un código que permita guardar los datos del mensaje recibido en una base de datos que más adelante se creará en Cloud SQL.

Los datos que se han almacenado en la tabla son:

idzn: para guardar el ID de la zona del dispositivo que envía datos.

idgw: para almacenar el ID del dispositivo que funciona como pasarela.

iddv: para almacenar el ID del dispositivo que envía el dato.

temp: para guardar la temperatura.

hum: para almacenar la humedad.

time: que contendrá la fecha de adquisición de los valores.

El código necesita una biblioteca de funciones llamada “mysql”, que no se tiene que incorporar, pero si indicar en un archivo llamado “REQUIREMENTS.TXT” que el código depende de ella. Se incluye en el apartado “REQUIREMENTS” como se aprecia en la figura 4.36

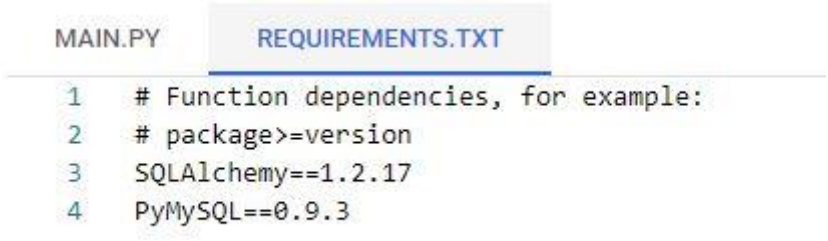


Figura 4.36 Dependencias mysql para acceder al Cloud SQL

Al código que se genera automática al crear una función, se le ha añadido las siguientes líneas de código con el siguiente propósito:

Utilizar las funciones del paquete “sqlalchemy”. Además de incorporar los paquetes base64 y json.

```
import base64
import json
import sqlalchemy
```

- Extraer los datos del mensaje que envía el dispositivo

```
pubsub_message = base64.b64decode(event['data']).decode('utf-8')
print(pubsub_message)
data=json.loads(pubsub_message)
idzn=data["idzn"]
idgw=data["idgw"]
iddv=data["iddv"]
temp=data["t"]
```

```
hum=data["h"]
time=data["ts"]
```

- Conectarse a la base de datos

```
db = sqlalchemy.create_engine(
    sqlalchemy.engine.url.URL(
        drivername='mysql+pymysql',
         host="146.148.114.59",
         username="root",
         password="**************",
         database="data"
     )
)
```

- Introducir datos en la tabla creada de la base de datos

```
sql="INSERT INTO envtemphum (idzn,idgw,iddv,temp,hum,ts) VALUES
('" + idzn + "', '" + idgw + "', '" + iddv + "', '" + temp + "',
'" + hum + "', FROM_UNIXTIME(" + time + "));"
with db.connect() as conn:
    conn.execute(sql)
    conn.close()
```

4.6.4 Servicio - Almacén de datos BigQuery

BigQuery es un servicio web de Cloud Computing que admite almacenar y consultar datos masivamente. El uso de este servicio es muy sencillo, permitiendo en tiempo real, estudiar las bases de datos los agentes implicados como pueden ser un desarrollador o un analista [13].

En pocos segundos se pueden realizar consultas SQL sobre estos datos que pueden llegar a contener infinidad de información. Si se conoce el lenguaje SQL, realizar una consulta es muy fácil. Los datos extraídos de estas consultas pueden ser por una lado guardados en unas tablas o bien exportados para un posterior análisis.

Lo primero que se hace es acceder al servicio desde el menú izquierdo y dentro de los servicios de “BIG DATA”, se selecciona “BigQuery” según la Figura 4.37.

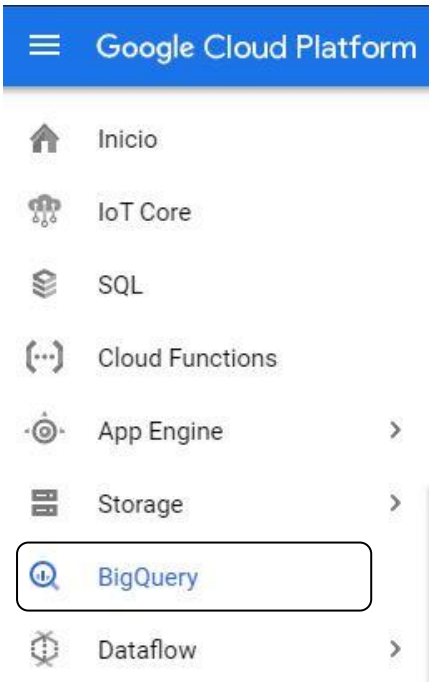


Figura 4.37 Acceder a BigQuery

Se accede a la página de inicio y el siguiente paso es fijar el proyecto, utilizando la opción “+AÑADIR DATOS” (Figura 4.38).

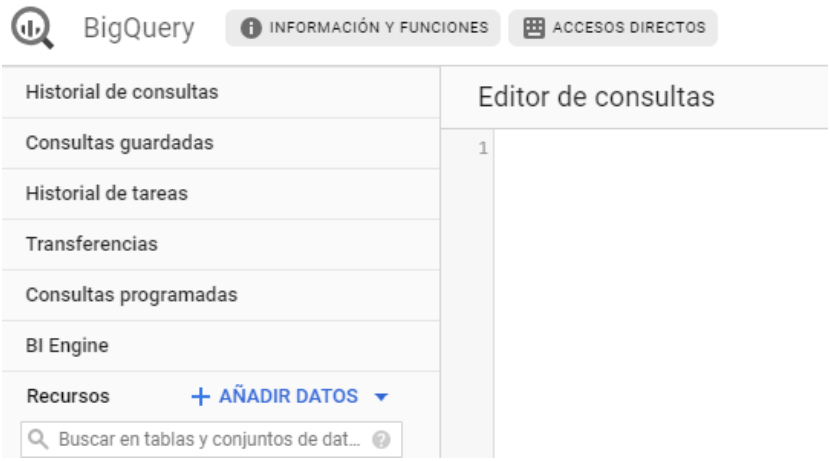


Figura 4.38 Menú gestión BigQuery

Aparece un desplegable (Figura 4.39) y se elige el nombre del proyecto en el que se está trabajando. En este caso “tfgflc”

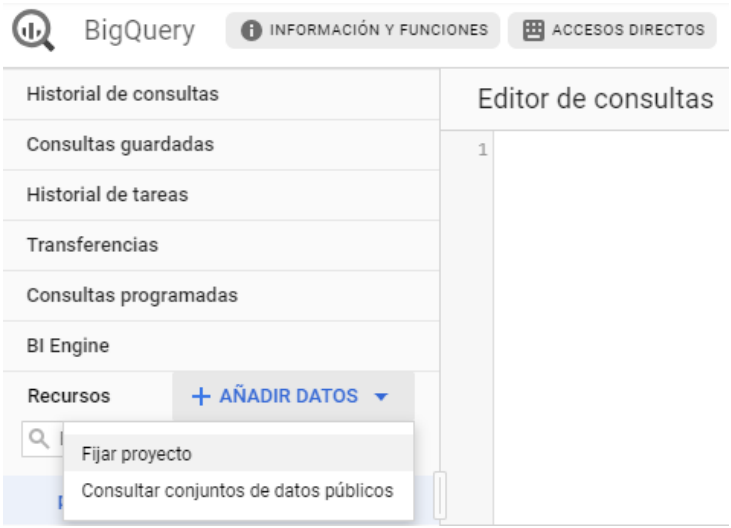


Figura 4.39 Fijación del proyecto en BigQuery

A continuación, se crea un “Conjunto de datos” (Figura 4.40).

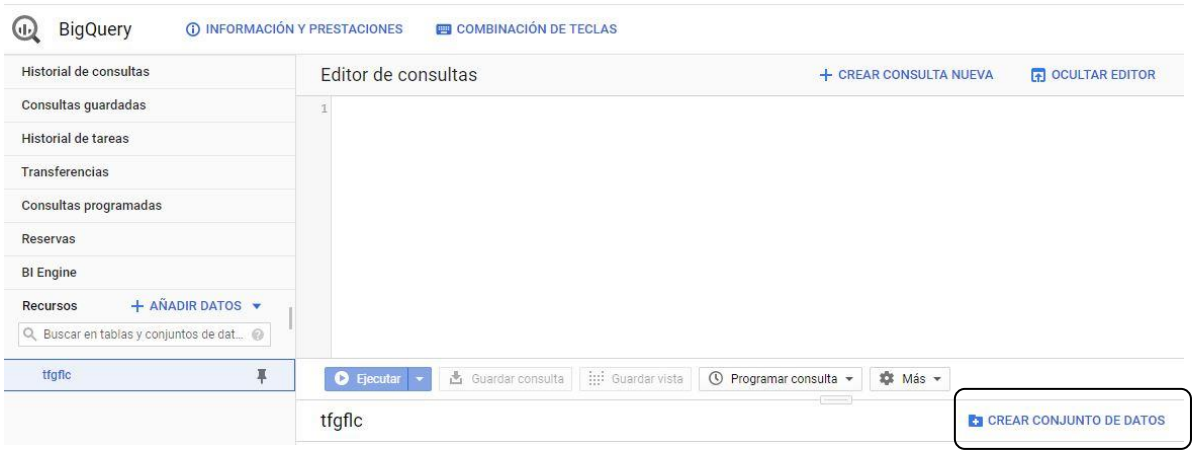


Figura 4.40 Crear un conjunto de datos BiqQuery

Se le puede dar cualquier nombre, por ejemplo “IA”. En ubicación de datos, se deja la opción “predeterminada” (Figura 4.41).

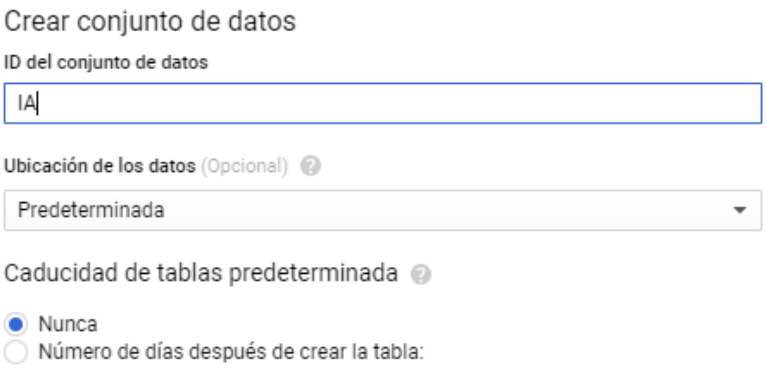


Figura 4.41 Crear conjunto de datos BigQuery

Cuando se ha creado, en la parte izquierda aparece. (Figura 4.42)

Figura 4.42 Conjunto de datos creados en BigQuery

Si se pulsa sobre el icono “IA”, aparece un nuevo menú para crear tablas en el conjunto de datos.

Figura 4.43 Menú para crear tablas en BigQuery

Pulsando sobre la opción de “CREAR TABLA”, (Figura 4.43) se puede crear una tabla en este caso, llamada “DATOS”. Esta tabla se crea a partir de una tabla vacía, el destino es el nombre del proyecto que se ha realizado y el conjunto de datos el creado anteriormente “IA”. El tipo de tabla es “tabla nativa” (Figura 4.44) y va a tener 6 campos. (Figura 4.45)

idzn: para guardar el ID de la zona del dispositivo que envía datos. Es de tipo String.

idgw: para almacenar el ID del dispositivo que funciona como pasarela. Es de tipo String.

iddv: para almacenar el ID del dispositivo que envía el dato. Es de tipo String.

temp: para guardar la temperatura. Es de tipo Entero.

hum: para almacenar la humedad. Es de tipo Entero.

time: que contendrá la fecha de adquisición de los valores. Es de tipo TimeStamp.

Figura 4.44 Crear tabla BigQuery

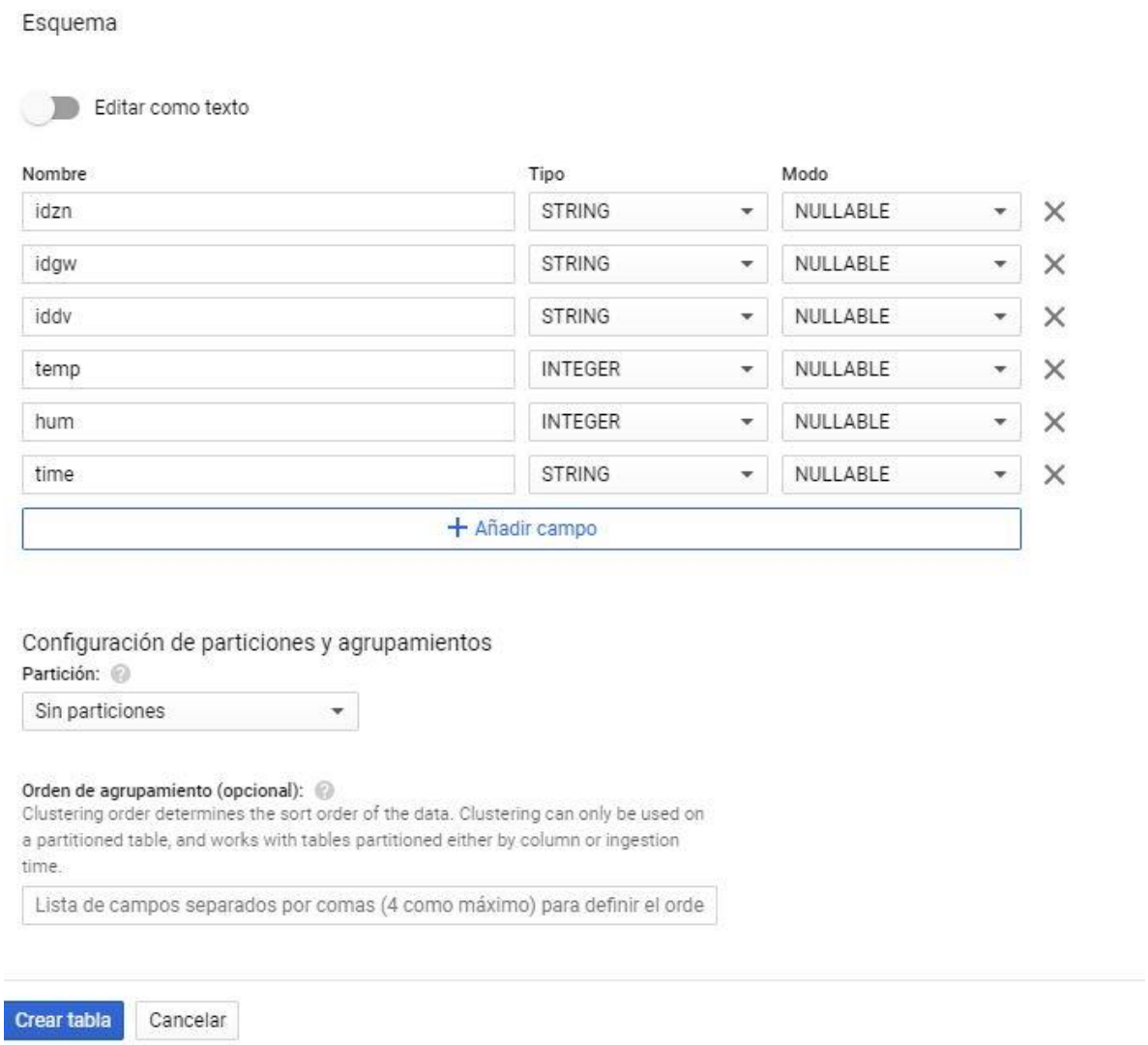


Figura 4.45 Campos de la tabla BigQuery

Los campos de la tabla deben de coincidir con los atributos de los mensajes publicados por los dispositivos clientes. Se puede observar que el aspecto de un mensaje publicado es el siguiente (Figura 4.46):

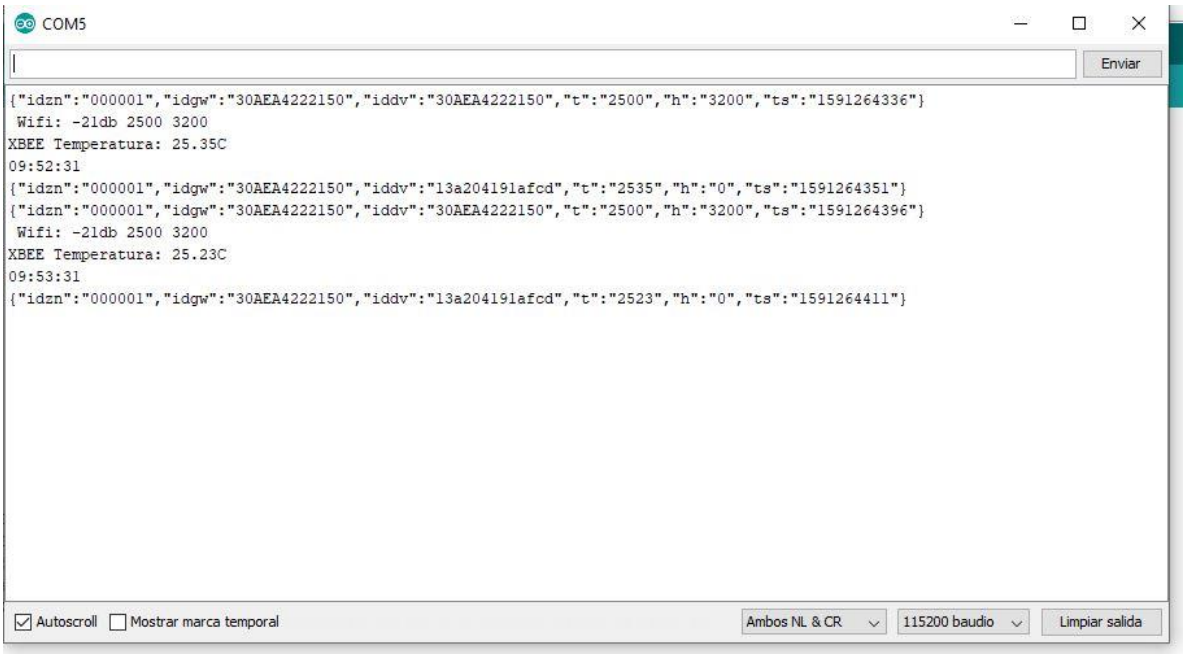


Figura 4.46 Aspecto de un formato publicado de datos

En la Figura 4.46, se puede identificar los atributos que se han descrito anteriormente.

Si en el “editor de consultas” (Figura 4.47) se escribe la siguiente instrucción SQL:

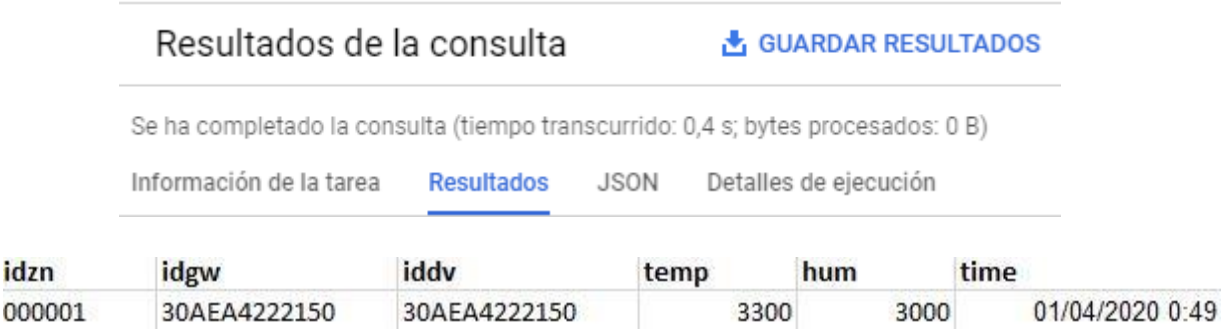
INSERT INTO IA.DATOS (idzn, idgw, iddv, temp, hum, time) VALUES ('000001', '30AEA4222150','30AEA4222150', 3300, 3000, '2020-04-01 00:49:53');

Figura 4.47 Editor de consultas con instrucción SQL

Se obtiene como resultado la inserción de estos valores en la tabla creada. Para poderlo comprobar, se puede escribir en el editor el siguiente código:

SELECT * FROM tfgflc.IA.DATOS LIMIT 1000 y pulsando en la opción “CONSULTAR TABLA” (Figura 4.48) se puede observar si se ha introducido en la tabla los datos de la consulta anterior (Figura 4.47). Se ha introducido “*” para que muestre los valores de todos los atributos. (Figura 4.49)

Figura 4.48 Consultar tabla en BigQuery



idzn	idgw	iddv	temp	hum	time
000001	30AEA4222150	30AEA4222150	3300	3000	01/04/2020 0:49

Figura 4.49 Datos introducidos en la tabla BigQuery

4.6.5 Servicio - Aplicación con Dataflow

Dataflow es un servicio de Google Big Data y permite la integración entre IoT Core con servicios como BigQuery y Google Cloud Storage, facilitando el procesado de datos. [14]

El cliente de este trabajo actúa como suscriptor y a través de MQTT envía datos a IoT Core, Dataflow puede adquirir los mensajes Pub/Sub que llegan a un topic y reconducir el flujo de datos para que éstos se almacenen directamente en un conjunto de datos BigQuery. El flujo de trabajo puede verse en la Figura 4.50, igualmente puede observarse

como los datos enviados por la red de sensores se almacenan en BigQuery por si se quiere analizar.

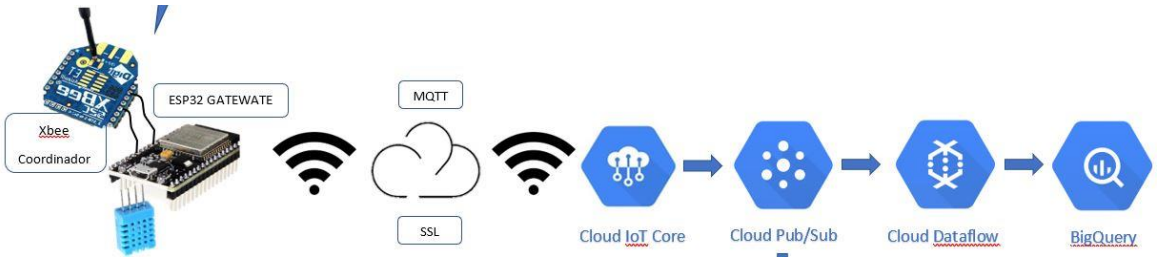


Figura 4.50 Procesos conectados con Dataflow

La primero de todo es acceder al servicio, para ello en el menú de la izquierda y dentro de los servicios de “BIG DATA”, se selecciona “Dataflow”. (Figura 4.51)

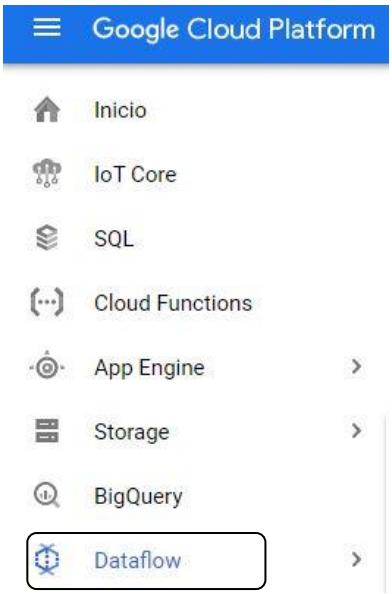


Figura 4.51 Servicio Dataflow

El siguiente paso es crear una tarea, utilizando la opción “+CREAR TAREA A PARTIR DE PLANTILLA”. (Figura 4.52)

Figura 4.52 Creación de tareas en Dataflow

Se han insertado los siguientes datos para crear una nueva tarea (Figuras 4.53 y 4.54):

Nombre de la tarea: DatosDev

Plantilla de cloud Dataflow: Pub/Sub Subscription to BigQuery

Punto de conexión regional: europe-west1

Cloud Pub/Sub input topic: projects/tfgflc/subscription/environments

BigQuery output table: tfgflc:IA.DATOS

Ubicación temporal: gs://misegmento/tmp

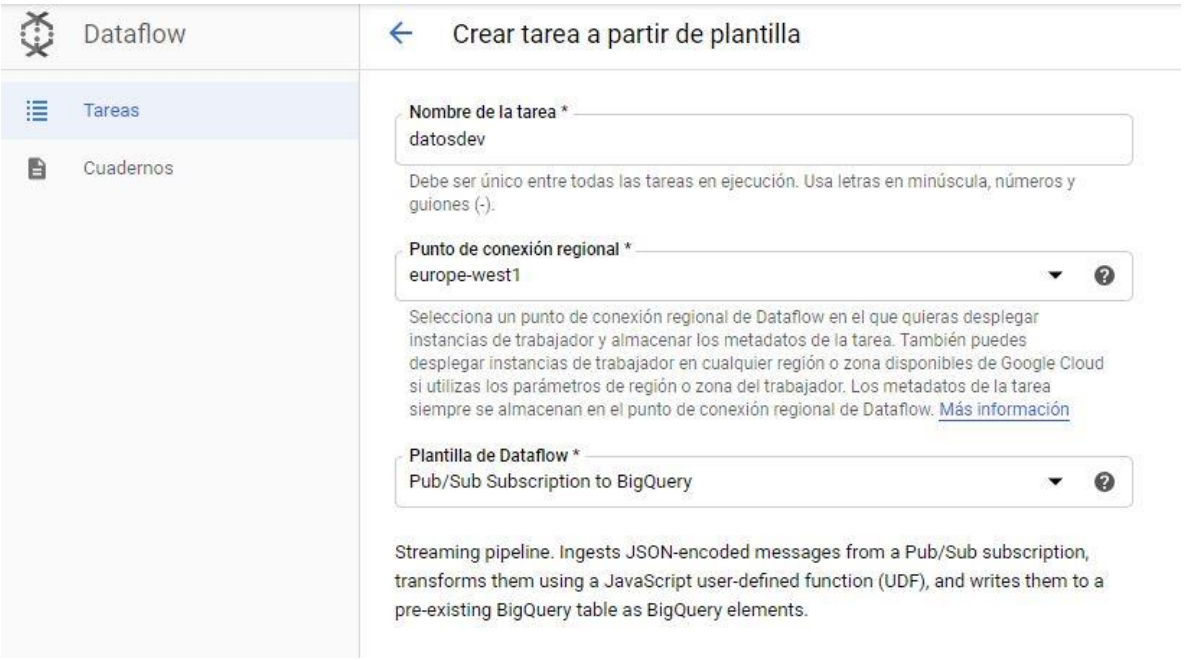


Figura 4.53 Datos creación tarea en Dataflow

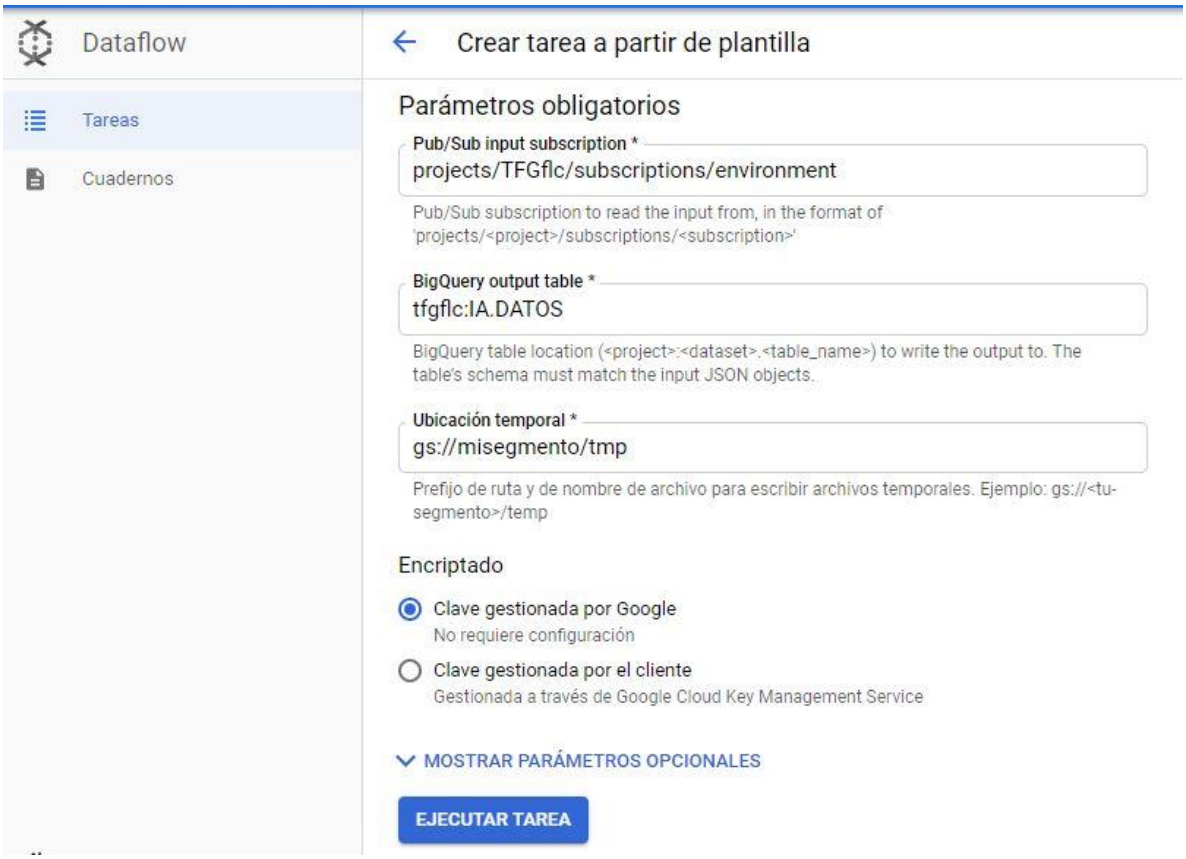


Figura 4.54 Datos creación tarea en Dataflow

Unos de los datos que se debe de proporcionar para crear una tarea, es el prefijo de ruta y nombre de archivo para escribir archivos temporales. Para ello se ha utilizado el servicio “Cloud Storage”. Igual que el anterior en el menú de la izquierda y dentro de los servicios de “ALMACENAMIENTO”, se selecciona “Storage” (Figura 4.55) y se accede a la página de inicio:

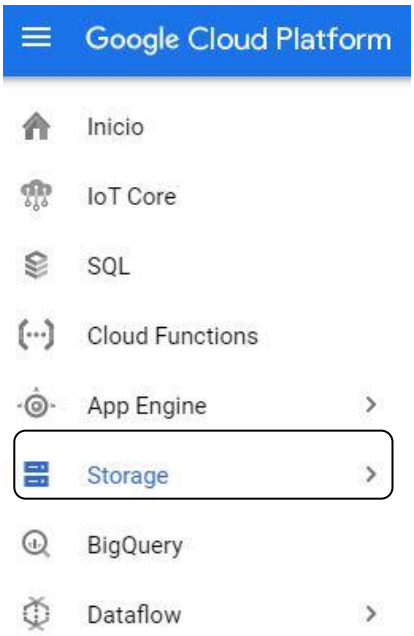


Figura 4.55 Servicio Storage

Una vez seleccionado se crea el segmento (Figura 4.56) y dentro de él un directorio temporal llamado /tmp (Figura 4.58)

Figura 4.56 Servicio Storage - Crear segmento

Se le tiene que asignar un nombre al segmento (Figura 4.57) y por último pulsar el botón de “Crear” (Figura 4.57)

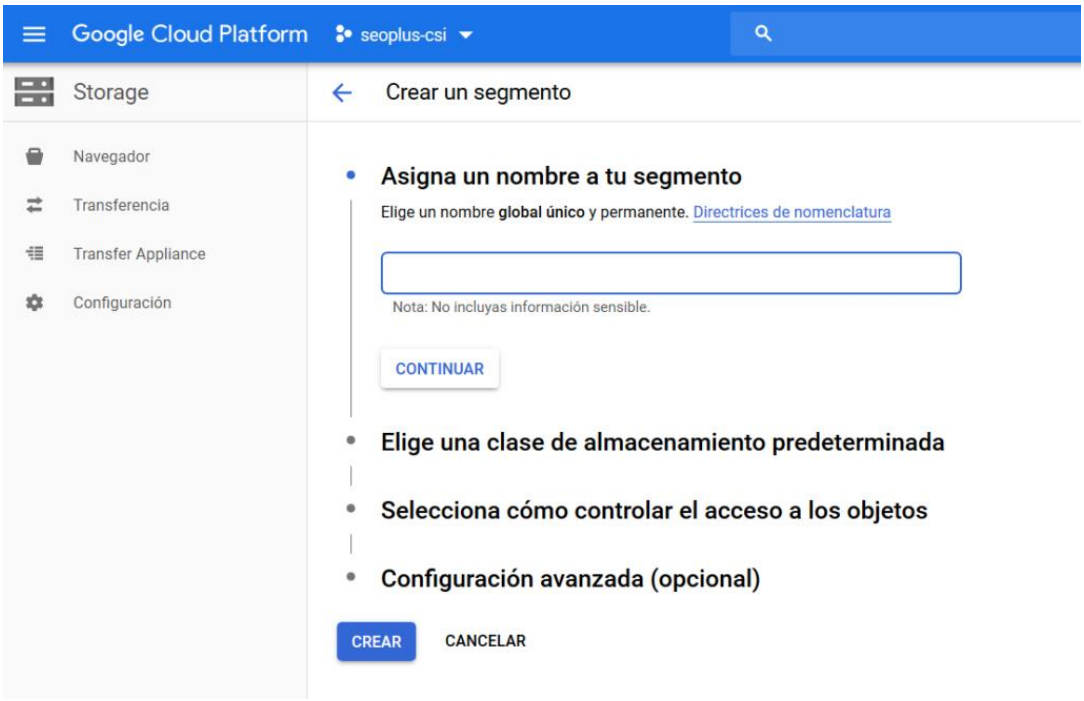


Figura 4.57 Nuevo Segmento en Storage

Y por último se crea la carpeta temporal “tmp”

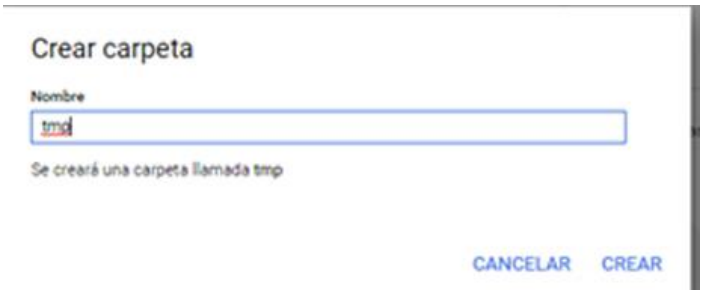
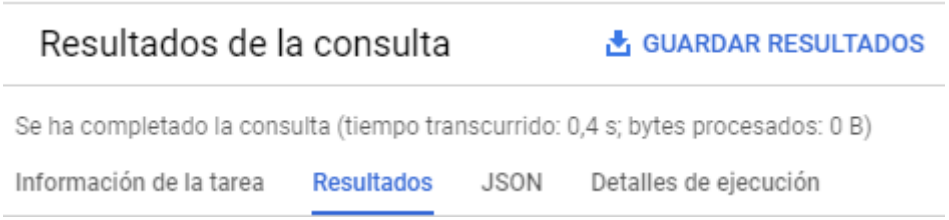


Figura 4.58 Creación carpeta temporal dentro del segmento en Storage

Una vez creada la carpeta temporal dentro del nuevo segmento, ya se puede regresar al servicio de Dataflow y terminar de crear la tarea con todos los datos necesarios.

Para comprobar que la tarea ha funcionado, se ejecuta la red de sensores y se accede a BigQuery, para comprobar que en la tabla “tfgflc.AI.DATOS” aparecen los nuevos datos que los dispositivos clientes han enviado. Realizando una consulta (Figura 4.59) se podrán comprobar que los datos se están almacenando (Figura 4.60):

Figura 4.59 Consulta a BigQuery para comprobar los nuevos datos introducidos



idzn	idgw	iddv	temp	hum	time
000001	30AEA4222150	30AEA4222150	3000	3500	07/04/2020 11:40
000001	30AEA4222150	13a204191afcd	3100	3550	07/04/2020 11:41
000001	30AEA4222150	30AEA4222150	3200	3400	07/04/2020 11:42
000001	30AEA4222150	13a204191afcd	3000	3450	07/04/2020 11:43
000001	30AEA4222150	30AEA4222150	3100	3400	07/04/2020 11:44
000001	30AEA4222150	13a204191afcd	3110	3370	07/04/2020 11:45
000001	30AEA4222150	30AEA4222150	3120	3340	07/04/2020 11:46

Figura 4.60 Datos introducidos en la nueva tarea

Estos resultados se pueden guardar simplemente pulsando la opción de “GUARDAR RESULTADOS”. Se muestra un menú con distintas opciones de exportación (Figura 4.61):

Figura 4.61 Formatos posibles para exportar los datos

Se puede elegir a CSV (archivo local) y descargar los datos en formato CSV separado por comas para posteriores análisis. Estos datos se pueden ver en la siguiente imagen (Figura 4.62)

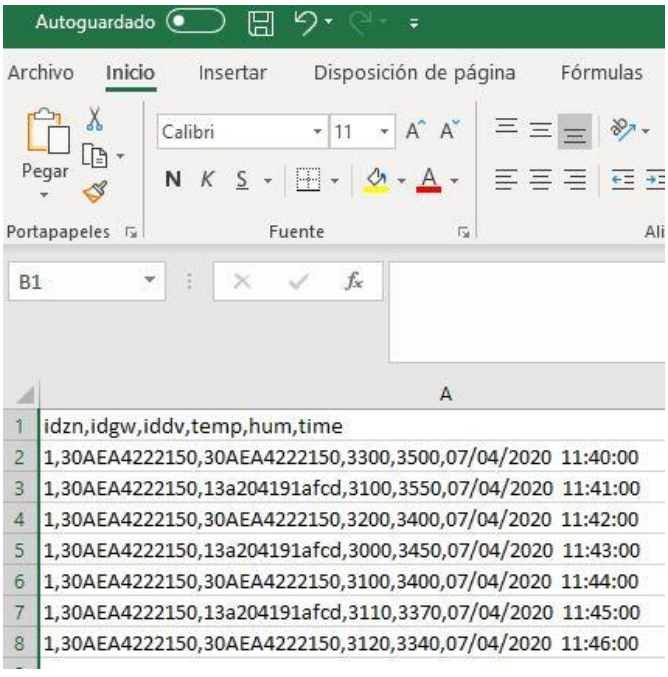


Figura 4.62 Archivo de datos en formato CSV

4.6.6 Servicio - Cloud SQL

Cloud SQL es un servicio de base de datos totalmente administrado que puede configurar, mantener y administrar fácilmente bases de datos relacionales en Google Cloud Platform. Cloud SQL se puede usar con MySQL o Postgre SQL [15]. Para este trabajo se ha utilizado MySQL.

Cloud SQL proporciona un alto nivel de rendimiento, escalabilidad y comodidad. Debido a que está alojado en Google Cloud Platform, proporciona una infraestructura de base de datos para aplicaciones que pueden ejecutarse en cualquier lugar. Es fácil de usar y no requiere ninguna instalación de software. Todos los sistemas de copia de seguridad, replicación y actualización se pueden automatizar y garantizar la disponibilidad en todo el mundo.

Cloud SQL proporciona la función de usar instancias de bases de datos y las llama "instancias SQL" para definir el sistema de administración de bases de datos. Al crear una instancia, lo que se está haciendo es crear una máquina virtual que contenga el sistema de administración de la base de datos.

Siguiendo el flujo de trabajo del escenario principal, una vez que el cliente está enviado los datos al Cloud IoT Core, mediante la programación de la función que se ha realizado en el Cloud Functions, estos datos se almacenarán en una tabla que se creará en el Cloud SQL, para su posterior análisis y visualización de estos. Como se ha visto anteriormente la función que se ejecutaba en el Cloud Functions, conectaba con una base de datos e introducía una serie de valores en una tabla llamada “Data”, cada vez que se producía la entrada de un dato (Pub/Sub), el Cloud Functions se activaba. (Figura 4.63)

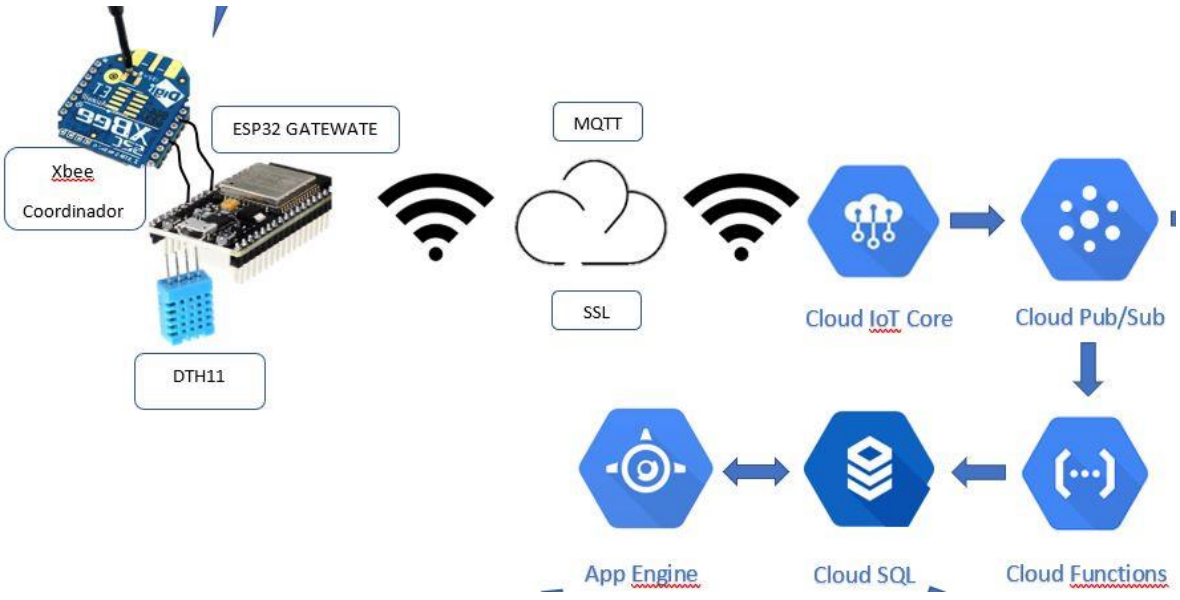


Figura 4.63 Escenario Cloud SQL

Primeramente, se crea una instancia SQL y para ello se accede al menú SQL. (Figura 4.64)

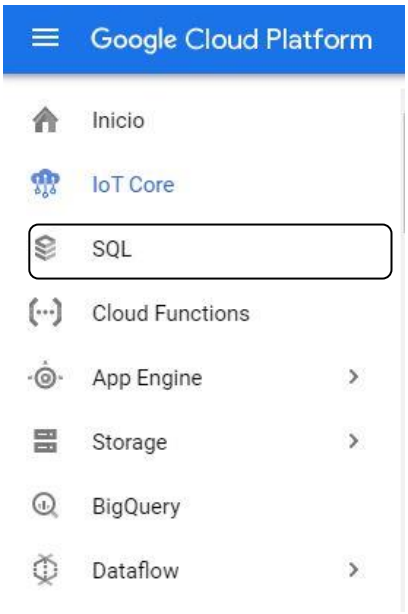


Figura 4.64 Cloud SQL

Una vez dentro del Cloud SQL se selecciona la opción de “Crear una instancia”, (Figura 4.65) seguidamente se selecciona MySQL y se pulsa sobre la etiqueta “Choose MySQL”, (Figura 4.66)

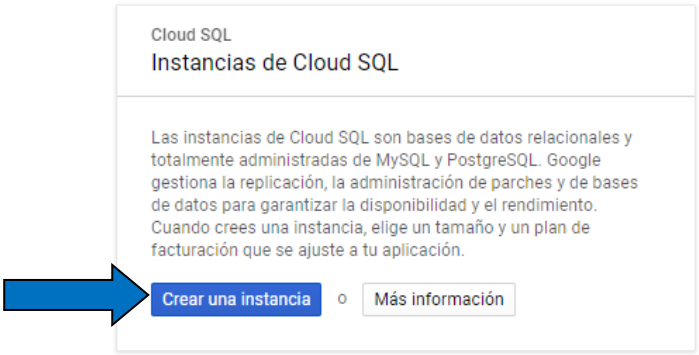


Figura 4.65 Crear instancia de Cloud SQL

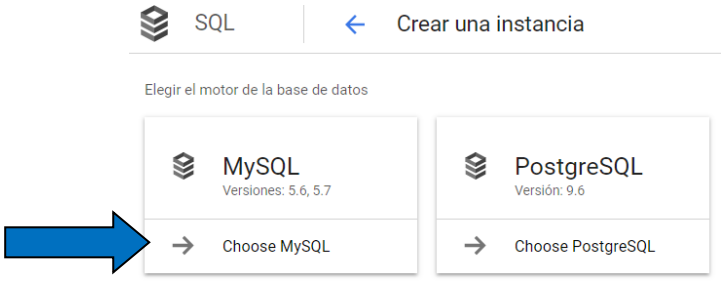


Figura 4.66 Elección MySQL para la nueva instancia Cloud SQL

Seguidamente se muestra el asistente para crear una instancia, (Figura 4.67) se debe elegir un nombre o ID de instancia, una contraseña del usuario root, que será el administrador de la instancia y elegir una zona geográfica.

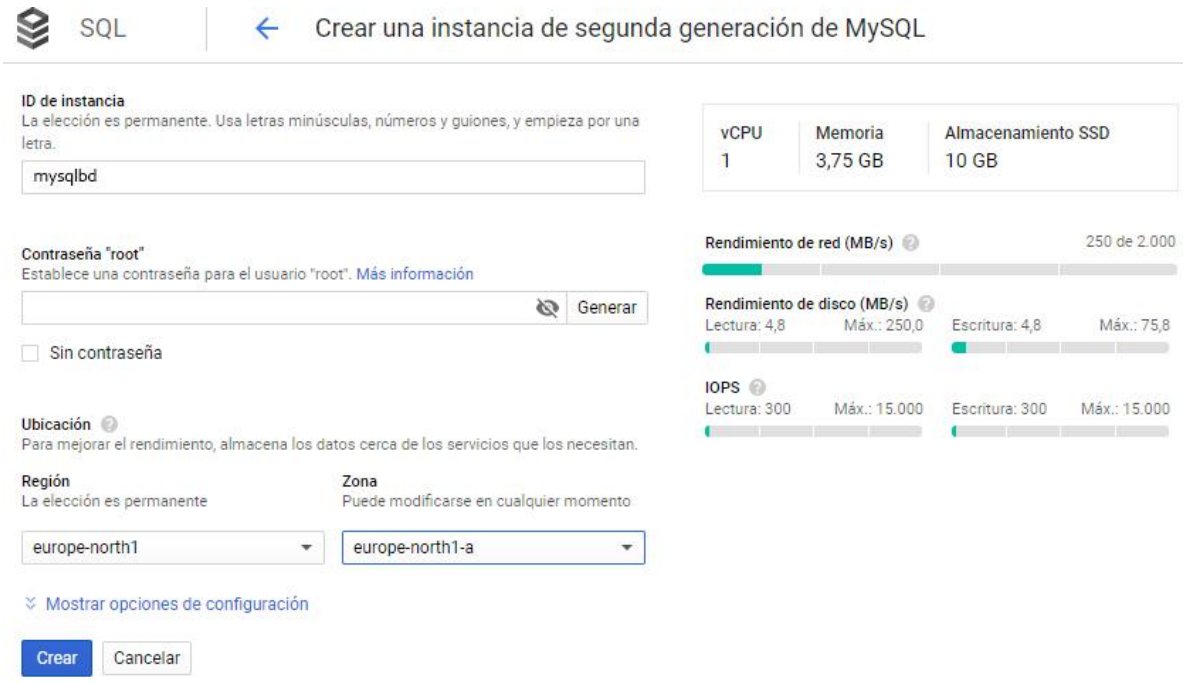


Figura 4.67 Datos de la instancia SQL

Una vez creada se puede apreciar en la imagen que sigue (Figura 4.68).

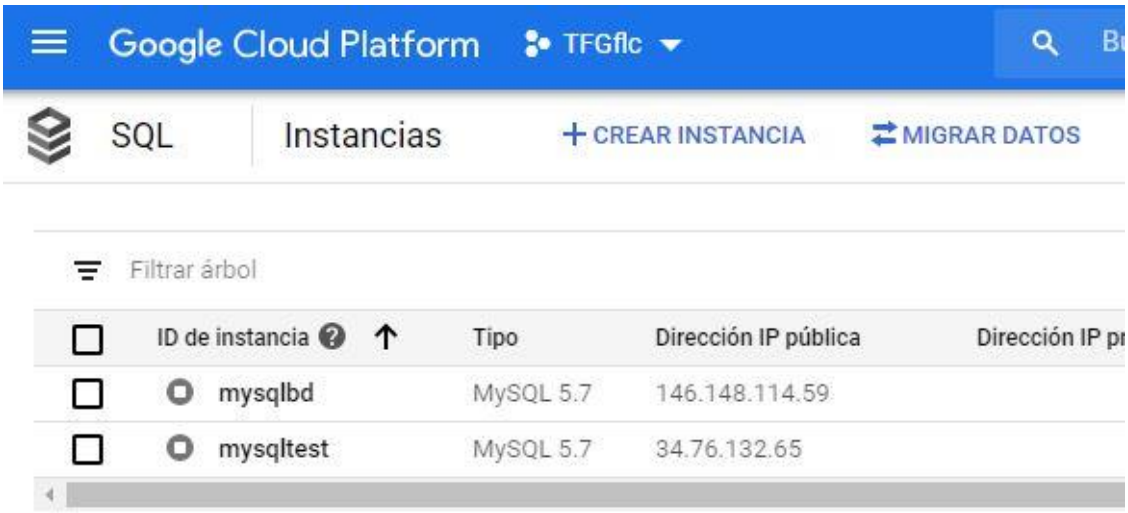


Figura 4.68 Instancia creada en Cloud SQL

Una vez que ya se ha creado la instancia, ya está preparada para recibir datos desde el cliente.

Una de las opciones que ofrece Google y como se puede ver el escenario anterior en la Figura 4.63, es la realización de una aplicación web con un servicio de Google que

no es otro que App Engine, obteniendo los datos desde Cloud SQL, pero también existe la posibilidad de que un cliente externo quiera acceder a esta base de datos para poder representar esta misma información. En este trabajo se explicará como un cliente externo llamado Grafana, se conecta a esta base de datos y representa los datos almacenados en ella.

Pero para que una aplicación externa, no alojada en Google Cloud Platform pueda acceder a esta información se accede a los detalles de la instancia creada anteriormente y en el menú “CONEXIONES” (Figura 4.69) se pulsa en añadir red.

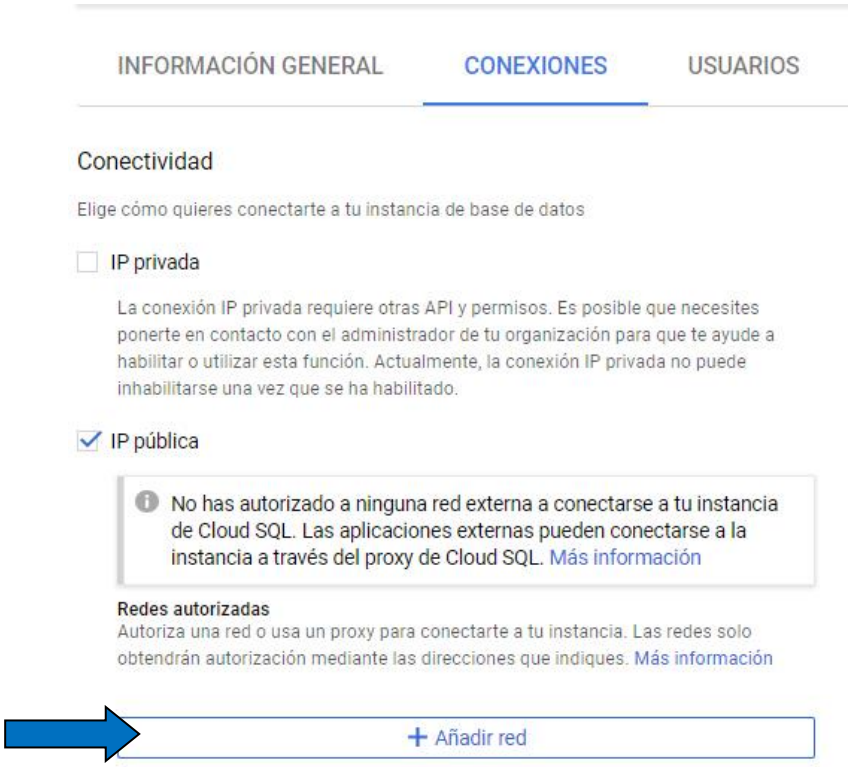


Figura 4.69 Añadir red en la instancia SQL

Si se pulsa sobre el botón “+Añadir red” esto muestra una nueva ventana (Figura 4.70), donde se especifica un nombre y las direcciones IP de red que quiera que se conecten. Esta opción se debe de tener un cuidado especial, ya que la introducción de una ip de red que no se quiera podría dar acceso indeseado a la instancia creada.

Por lo tanto, especifica la dirección ip de la red que se desee conectar. En este trabajo a efectos de prueba se ha incorporado la red 0.0.0.0/0, dando acceso a todas las direcciones IP externas. Para ello en el campo etiquetado como “Red” se introduce 0.0.0.0/0.

Una vez introducidos los datos de red, se pulsa en el botón “Listo” y seguidamente en guardar (Figura 4.70), con esto ya se puede acceder a la instancia desde la red que se quiera conectar.

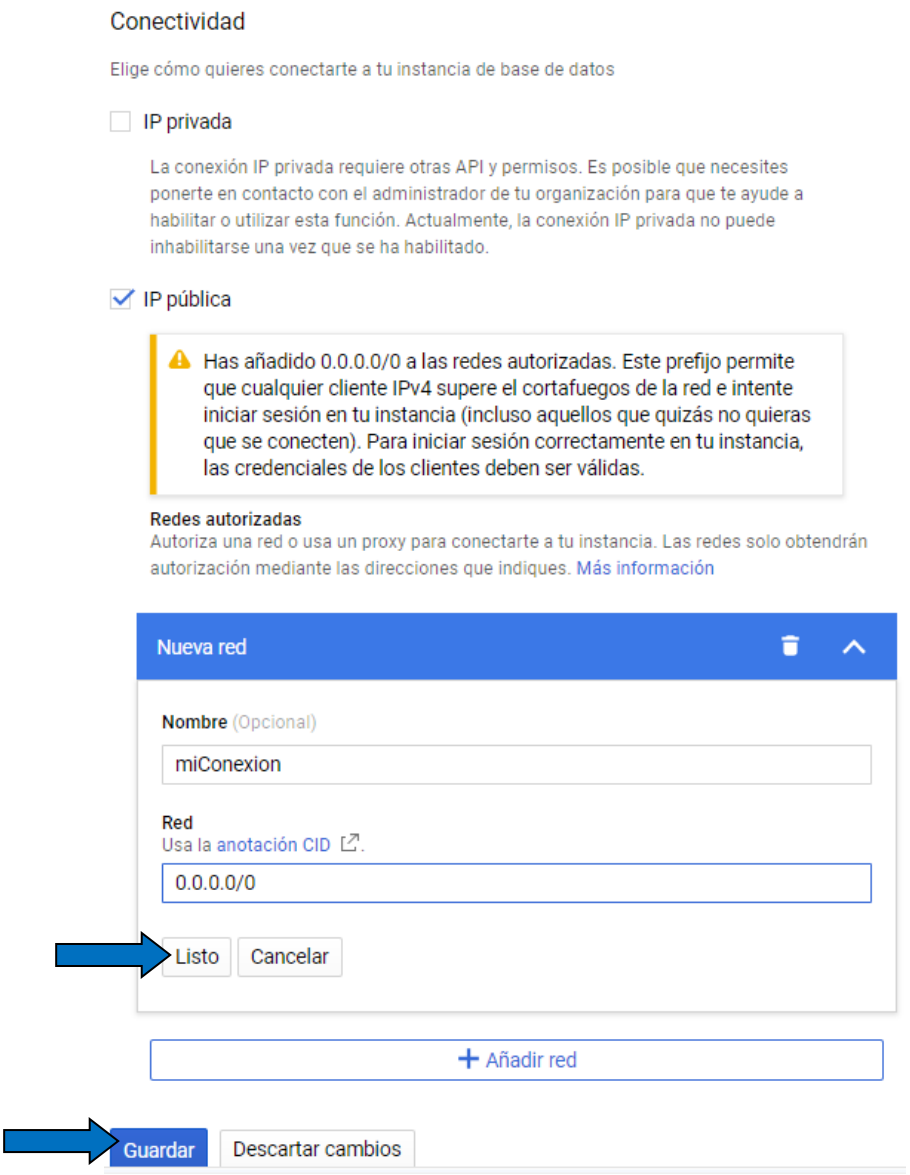


Figura 4.70 Direcciones IP que permite la conexión a la instancia SQL

4.6.7 Servicio - App Engine

Con App Engine se obtiene la posibilidad de desplegar aplicaciones. Este servicio permite publicar aplicaciones web en línea sin necesidad de preocuparse por la parte de la infraestructura, por lo tanto, se puede diseñar, mantener y escalar la aplicación conforme vaya haciendo falta. Se ejecuta en entornos seguros sin depender ni del sistema operativo ni de la ubicación física del servidor [16].

Unas de las características más importantes es que las aplicaciones pueden programarse con diferentes lenguajes de programación como pueden ser Python, Java o Javascript.

Siguiendo el escenario principal, ahora se realiza la aplicación web que va a mostrar todos los datos que han sido enviado por el cliente y almacenados en una base de datos como Cloud SQL. (Figura 4.71). Posteriormente son representados en una gráfica los datos consultados y devueltos en una consulta a la base de datos y estos se van actualizando en tiempo real.

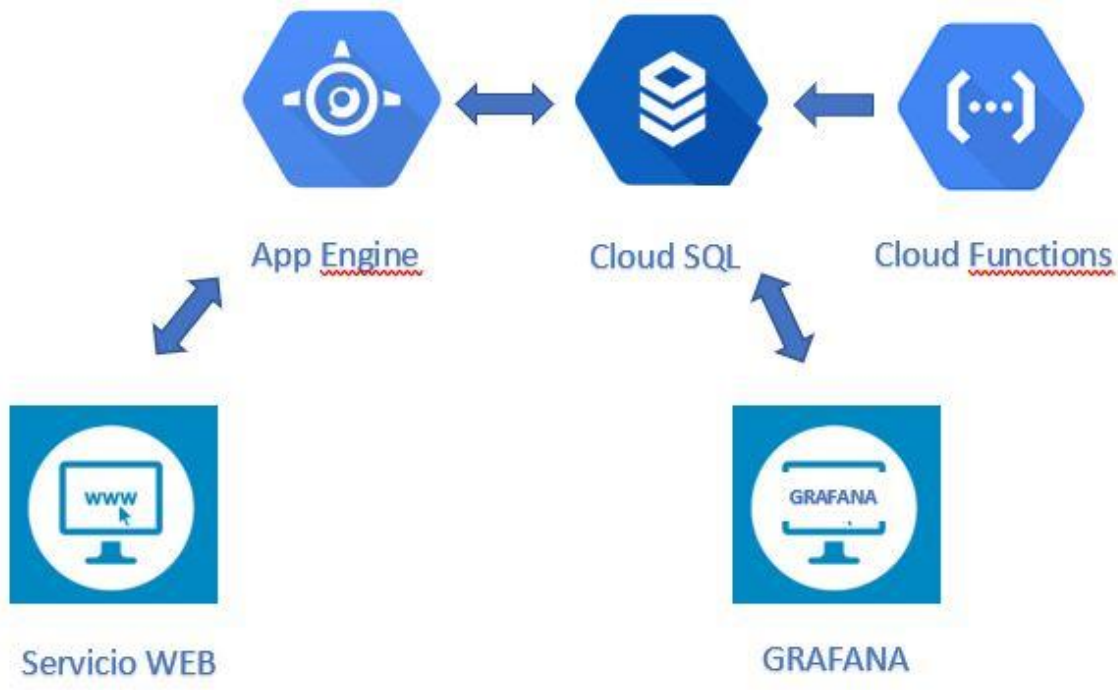


Figura 4.71 Escenario App Engine

Para la aplicación se ha elegido el lenguaje de programación Python. Esta aplicación se comunica con el servidor utilizando el estándar CGI. Este servidor recibe peticiones POST o GET que se procesan y producen una respuesta HTTP que es enviada al navegador web.

Para el desarrollo de la aplicación lo primero que se utiliza es la interfaz de comandos Google Cloud Shell Figura 4.72

Figura 4.72 Google Cloud Shell

y pulsando sobre "Abrir editor" se abre Google Cloud Editor (Figura 4.73), donde se empieza a diseñar la aplicación, pudiendo crear directorios y archivos que se necesite para ello.

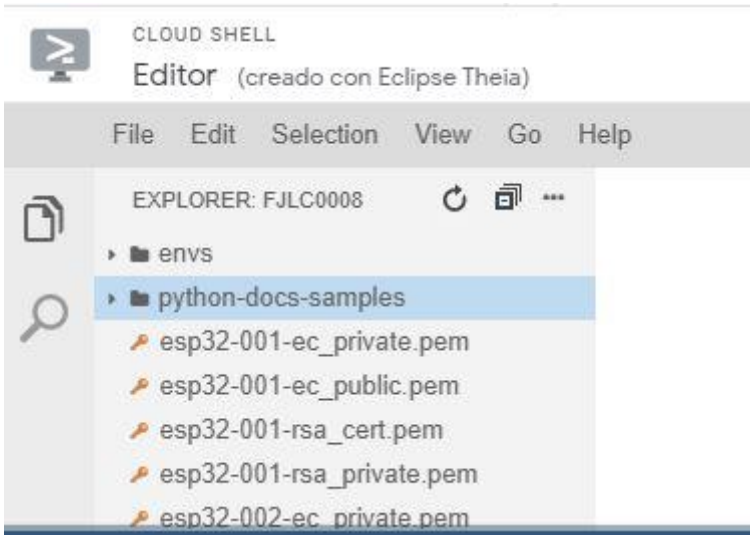


Figura 4.73 Google Cloud Editor

En este editor gráfico, sencillo de utilizar, se pueden crear directorios, generar archivos nuevos, subirlos desde un pc, editarlos y hasta descargarlos como se puede ver en las Figuras 4.74 y 4.75.

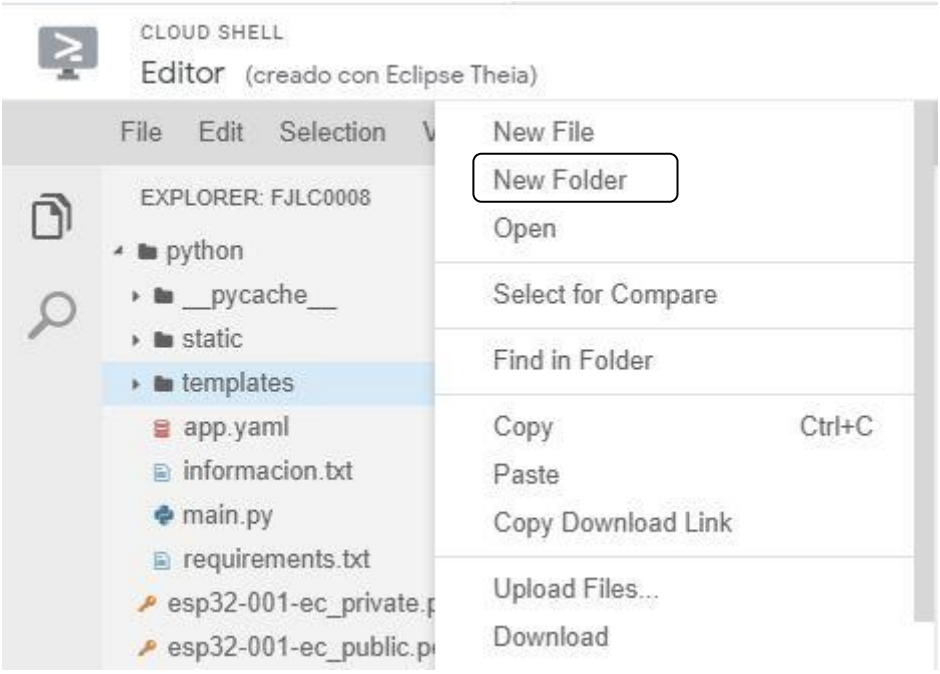


Figura 4.74 Crear directorio con Google Cloud Editor

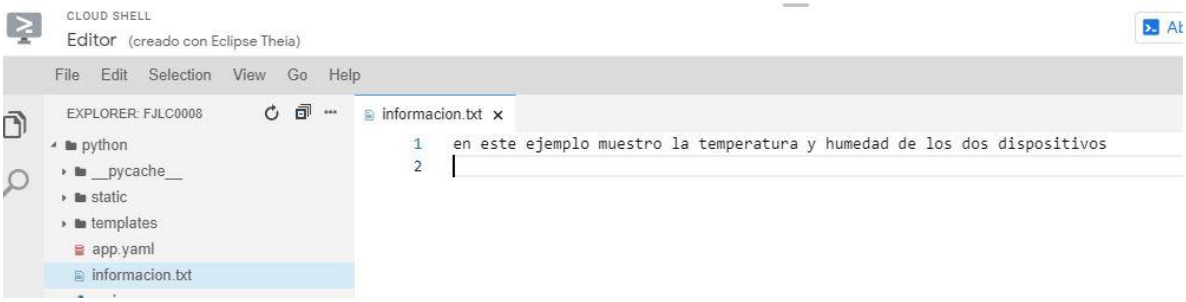


Figura 4.75 Editar archivos con Google Cloud Editor

Para diseñar la aplicación está debe tener una estructura de directorios prefijada. Existe un directorio principal que se llama "pyapp" que contiene la aplicación principal, un directorio “templates” que contiene la web a mostrar con “main.html” y un directorio "static" donde se guardan todos los archivos estáticos tipo fotos y archivos css. Para poder desplegar la aplicación se necesita un archivo que se llama "app.yaml" y por último un archivo "requerimets.txt" que se utiliza para recoger los nombres de los paquetes que deben ser instalados para que funcione la aplicación, ya que estos paquetes han sido utilizados en la misma. (Figura 4.76d)

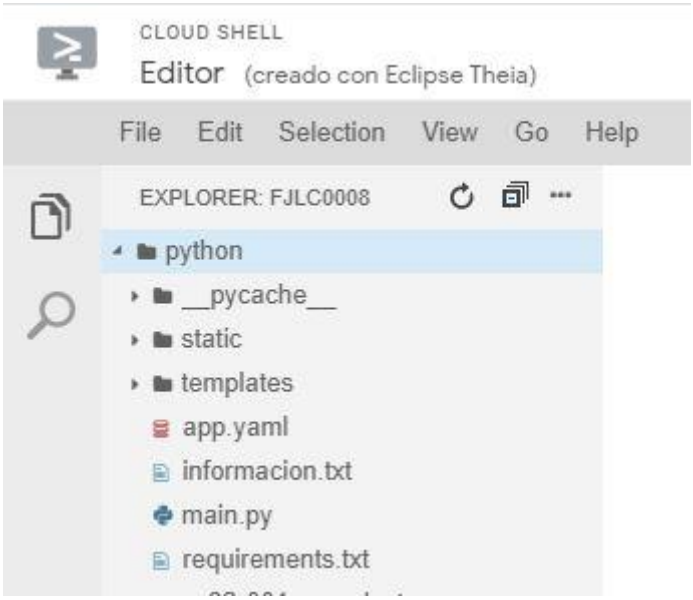


Figura 4.76 Archivos de la Aplicación App Engine

A través de App Engine, se implementa un front-end que permite al usuario acceder a la información recolectada de manera clasificada e histórica almacenada en Cloud SQL, a través de una página web.

El sistema cloud permite el redimensionamiento automático de las prestaciones en función de la demanda del servicio. De esta manera el coste de hardware a nivel de red y de servidores se convierte en variable y evita la necesidad de hacer un desembolso inicial pasando a ser de pago por uso.

Así mismo reduce el tiempo de despliegue en más de la mitad, si se considera el tiempo de adquisición e instalación del hardware dedicado a servidores.

Cuando se quiere realizar una aplicación web existen algunas metodologías, una de ellas es la Modelo-Vista-Controlador. Que divide la aplicación en tres capas:

Modelo: Conjunto de elementos que interactúan con los datos del programa.

Vista: Conjunto de elementos que presentan las interfaces de usuario.

Controlador: Es la aplicación principal, la que se encarga de que funcione la aplicación.

Para este trabajo, el modelo son los datos almacenados en Cloud SQL, las vistas son los documentos HTML que se almacenan en la carpeta “templates” y el controlador es la aplicación “main.py”

Para poder desplegar la aplicación web es necesario tener un archivo “app.yaml” que contiene la información de configuración. Se indica que se utiliza un entorno de ejecución Python, o sea, que se va a utilizar para la aplicación web el lenguaje de programación Python, y para ello se añade las siguientes líneas:

```
runtime: python37
```

```
handlers:
- url: /images
  static_dir: static/images
- url: /assets
  static_dir: static/assets
```

“Python37” indica que se utilice Python en su versión 3.7 y con “handlers” se crea una asociación entre el nombre que aparece en la url de los recursos solicitados y su ubicación real en otro directorio. Por ejemplo “/images” su directorio real en la aplicación es “static/images”.

El archivo “requirements.txt” contiene los nombres de las dependencias que deben instalarse. En este caso la aplicación hace uso del framework Flask y por lo tanto se debe indicar.

La aplicación principal es el archivo main.py que contiene el siguiente código fuente:

```
from flask import Flask, render_template, jsonify
import sqlalchemy


app = Flask(__name__)
db = sqlalchemy.create_engine(
    sqlalchemy.engine.url.URL(
        drivername='mysql+pymysql',
        host="146.148.114.59",
        username="root",
        password="****************",
        database="data"
    )
)


@app.route("/", methods=['GET'])
def principal():
    return render_template('main.html')
```

```
@app.route('/datos/<iddv>', methods=['GET'])
def getDatosIDdv(iddv):
    # iddv contine el id para buscar
    listamedidas = []
    with db.connect() as conn:
        # Ejecuta la consulta y obtiene los resultados
        result = conn.execute(
            "SELECT ts,temp,hum FROM envtemphum " +
            "WHERE iddv='" + iddv + "' ORDER BY ts DESC LIMIT 150
"
        ).fetchall()
        # convertir resultados a lista
        for row in result:
            listamedidas.append({
                'ts': str(row[0]),
                'temp': str((row[1])/100),
                'hum': str((row[2])/100),
            })
    return jsonify(listamedidas)


@app.route('/datos/recent/<iddv>', methods=['GET'])
def getDatosRecentIDdv(iddv):
    # iddv contine el id para buscar
    listamedidas = []
    with db.connect() as conn:
        # Ejecuta la consulta y obtiene los resultados
        result = conn.execute(
            "SELECT ts,temp,hum FROM envtemphum " +
            "WHERE iddv='" + iddv + "' ORDER BY ts DESC LIMIT 300
"
        ).fetchall()
        # Convierte los resultados en una lista
        for row in result:
            listamedidas.append({
                'ts': str(row[0]),
```

```
                'temp': str((row[1])/100),
                'hum': str((row[2])/100),
            })
    return jsonify(listamedidas)


@app.route('/datos/last/<iddv>', methods=['GET'])
def getLastDatoIDdv(iddv):
    # iddv contine el id para buscar
    listamedidas = []
    with db.connect() as conn:
        # Ejecuta la consulta y obtiene los resultados
result = conn.execute(
            "SELECT temp,hum, ts FROM envtemphum " +
            "WHERE iddv='" + iddv + "' ORDER BY ts DESC LIMIT 1"
        ).fetchall()

    # convierte los resultados en una lista
    for row in result:
        listamedidas.append({
            'ts': str(row[2]),
            'temp': str(row[0]),
            'hum': str(row[1]),
        })

    return jsonify(listamedidas)


# lista de dispositivos
@app.route('/datos/listdv', methods=['GET'])
def getListDV():
    # iddv contine el id para buscar
    listdev = []
    with db.connect() as conn:
        # Ejecuta la consulta y obtiene los resultados
        result = conn.execute(
            "SELECT DISTINCT iddv FROM envtemphum "
```

```
        ).fetchall()
        # Convierte los resultados en una lista
        for row in result:
            listdev.append({
                'iddv': str(row[0]),
            })
            # Return archivo JSON
    return jsonify(listdev)


@app.route('/datos/todas/<iddv>', methods=['GET'])
def medidas(iddv):
    listamedidas = []
    with db.connect() as conn:
        # Ejecuta la consulta y obtiene los resultados
        result = conn.execute(
            "SELECT ts,temp,hum FROM envtemphum " +
            "WHERE iddv='" + iddv + "' ORDER BY ts DESC"
        ).fetchall()
        # Convierte los resultados en una lista
        for row in result:
            listamedidas.append({
                'ts': str(row[0]),
                'temp': str(row[1]),
                'hum': str(row[2]),
            })
    return jsonify(listamedidas)


if __name__ == '__app__':
    app.run()
```

Este código fuente lo primero que realiza es la conexión a la instancia de la base de datos creada en el Cloud SQL

```
db = sqlalchemy.create_engine(
    sqlalchemy.engine.url.URL(
        drivername='mysql+pymysql',
        host="146.148.114.59",
```

```
            username="root",
            password="****************",
            database="data"
        )
)
```

Y seguidamente realiza una serie de servicios que pueden ser solicitados por la aplicación web y que más adelante se explicarán.

Para poder probar la aplicación web, en primer lugar, se ha desplegado el proyecto creado en App Engine, pudiéndose hacer desde Google Cloud Shell. La aplicación se encuentra en un directorio que se ha creado y se llama “python” por lo tanto se deben ejecutar las siguientes sentencias:

$ cd python

$ gcloud app deploy - -project tfgflc (tfgflc es el nombre del proyecto)

Se puede comprobar que se ha creado un servicio y que este acepta las peticiones de un navegador para ejecutar la aplicación. (Figura 4.77)

Figura 4.77 Servicio generado en App Engine

Una vez generado el servicio, se tiene que publicar la aplicación con la sentencia:

$ gcloud app browse - -project ftgflc

Esto devuelve una url donde se puede conectar para poder ver la aplicación web y que en este caso es: https://tfgflc.appspot.com

Como se ha dicho anteriormente dentro del directorio “templates” está el fichero “main.html” que es el que se ejecuta para poder ver la web principal.

Dentro del directorio “static” están los ficheros de “main.js” y “style.css” para aplicar estilos a la web y ejecutar código JavaScript que muestre los datos almacenados en el Cloud SQL.

4.6.7.1 Servicios REST

En la creación del código anterior del fichero “main.py” se ha utilizado servicios REST que no es otra cosa que un recurso direccionable o accesible a través de una web, y que puede ser trasferido entre cliente y servidor [17].

Se utiliza HTTP para comunicarse y los datos son transferidos mediante TCP/IP.

Los formatos que se pueden utilizar para estos servicios son: Texto plano, HTML, XML y JSON.

Los métodos utilizados son:

GET: Es una operación de solo lectura y que sirve para recuperar información del servidor.

PUT: Solicitud al servidor para almacenar una información.

DELETE: Para eliminar recursos.

POST: Para modificar el servicio de forma exclusiva enviando, o no, información en la petición

HEAD: Igual que GET, pero devuelve una cabecera asociada a la petición

OPTIONS: Sirve para solicitar información sobre las opciones de un recurso.

Una URI (Uniform Resource Identifier), es un enlace a un recurso que sirve para intercambiar información entre un cliente y un servidor y no cambia nunca.

Para el código anterior una URI podría ser: `/datos/listdv` utilizando el método GET.

En este trabajo el formato JSON es el empleado para entregar la información por los servicios REST implementados.

4.6.7.2 Back-End

El Back-End es la aplicación del lado del servidor que procesa una entrada y devuelve una salida.

Se ha empleado para el desarrollo el framework para Python Flask. Como motor de base de datos un sistema MySQL, y el despliegue se ha realizado en Google Cloud empleando su App Engine y su soporte para base de datos Cloud SQL. Esto permite la posibilidad de habilitar la escalabilidad automática según demanda del sistema manteniendo la calidad de servicio al usuario final.

Las salidas de datos del back-end se realizan en formato JSON, esto aumenta la eficiencia en la transmisión de información respecto a otros métodos como XML, ya que transmite menos caracteres pertenecientes a la definición de estructura.

Los servicios ofrecidos han sido modularizados usando REST, esto permite que el sistema pueda ser expandido empleando distintos sistemas y plataformas para visualizar los datos que proporciona sin que sea necesario modificar la implementación de esta parte.

Se han implementado los siguientes servicios REST:

-Obtención de lista de dispositivos que tienen al menos un registro en la base de datos de muestras.

```
('/datos/listdv', methods=['GET'])
```

-Obtención de los últimos 300 registros de un dispositivo dado

```
('/datos/recent/<iddv>', methods=['GET'])
```

-Obtención del último registro de un dispositivo dado

```
('/datos/last/<iddv>', methods=['GET'])
```

Todos estos servicios ejecutan sentencias SQL que se envían al gestor de base de datos y este devuelve los datos solicitados.

4.6.7.3 Front-End

Esta es la parte del lado del cliente, que permite ejecutar aplicaciones cliente por ejemplo en un navegador, obteniendo información del servidor, que ejecuta el Back-End diseñado anteriormente.

La idea es emplear el menor tráfico posible, para ello se apoya el desarrollo en Java Script y las peticiones Asíncronas. Para la base de desarrollo se emplea HTML5 que usa etiquetas descriptivas y el diseño lo delega a CSS3.

Java Script realizara las peticiones en modo asíncrono a los diferentes servicios REST facilitados por el Back-End y se encargara de formatearlo con CSS3 y visualizarlo editando el contenido de la web en modo local sin que sea necesario recargar el contenido completamente para agregar elementos.

Así mismo se integran diversas librerías JavaScript para usar graficas interactivas, para la visualización de los datos facilitados en JSON.

Esta metodología de diseño permite que la web sea válida y se auto adapte al contenido cambiante según se integren nuevos datos y sensores al sistema.

El Front-End puede ser accesible desde

https://tfgflc.appspot.com

Una vez ejecutado muestra el contenido del archivo “main.html” que está dentro del directorio “templates” En la Figura 4.78 Se puede observar que, en la pantalla inicial, lo que muestra es un menú con una lista de dispositivos clientes.

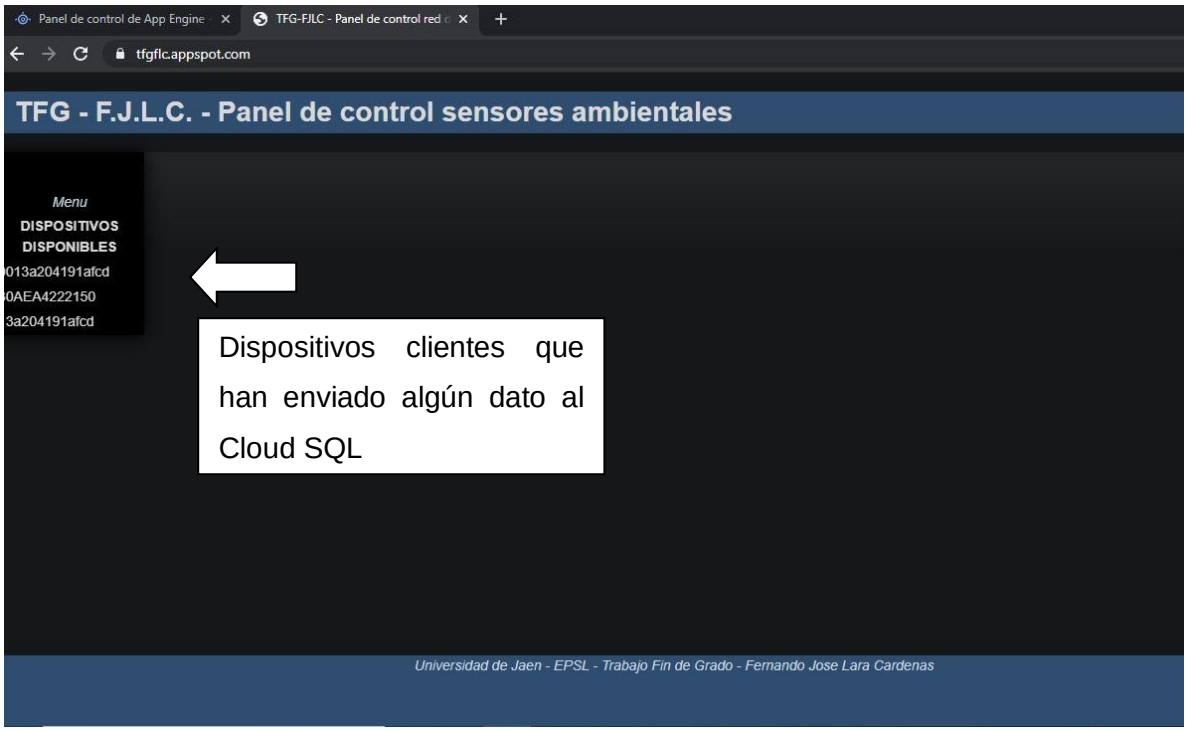


Figura 4.78 Pagina inicio aplicación App Engine "main.html"

Para comprobar que algunos de los servicios REST funcionan correctamente, por ejemplo, los dispositivos clientes que envían datos al servidor se deben escribir https://tfgflc.appspot.com/datos/listdv. La vista devolverá los datos que se han introducido en la instancia que se había creado en el servicio Cloud SQL. (Figura 4.79)

Figura 4.79 Servicio que muestra una lista de dispositivos que han enviado datos a GCP y han sido almacenado en Cloud SQL

El código que se ejecuta para esto es:

```
@app.route('/datos/listdv', methods=['GET'])
def getListDV():
    # iddv contine el id para buscar
    listdev = []
    with db.connect() as conn:
        # Ejecuta la consulta y obtiene los resultados
        result = conn.execute(
```

```
            "SELECT DISTINCT iddv FROM envtemphum "
        ).fetchall()
        # Convierte los resultados en una lista
        for row in result:
            listdev.append({
                'iddv': str(row[0]),
            })
            # Return archivo JSON
    return jsonify(listdev)
```

Conecta con la base de datos y realiza la consulta con “SELECT DISTINCT”, distinguiendo entre los “iddv” o dispositivos clientes que han enviado datos. Como se puede observar, los dos dispositivos que se presentaban en el escenario principal como clientes y enviaban datos de los sensores.

Si se quiere mostrar los últimos datos introducidos por un sensor, por ejemplo, el que tiene como iddv: 30AEA4222150, se debe escribir en el navegador https://tfgflc.appspot.com/datos/recent/30AEA4222150. Puede apreciarse en la siguiente imagen (Figura 4.80).

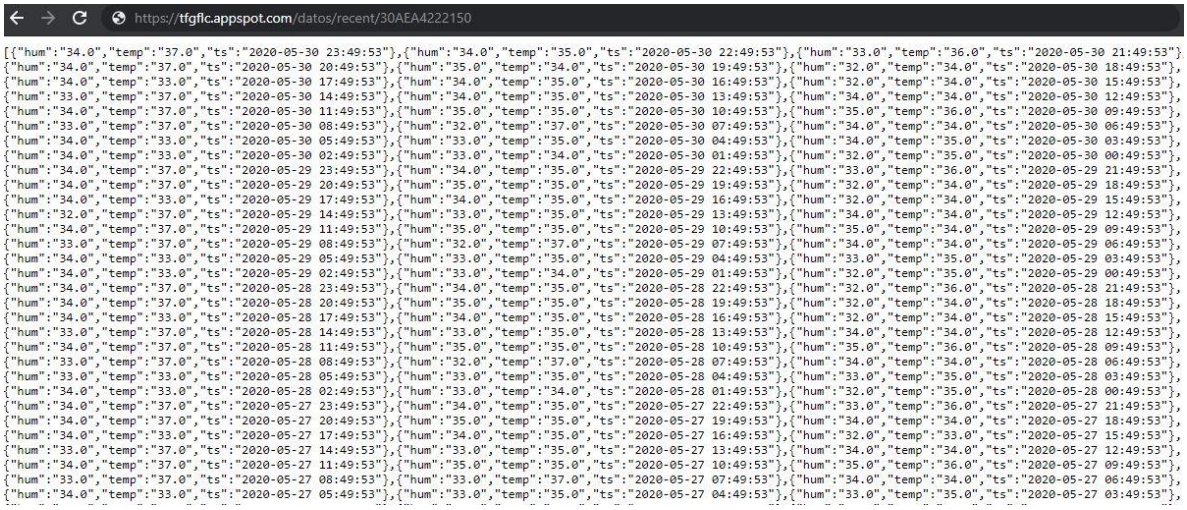


Figura 4.80 Servicio que muestra los últimos 300 registros enviados por un cliente y han sido almacenado en Cloud SQL

El código que se ejecuta es el siguiente:

```
@app.route('/datos/recent/<iddv>', methods=['GET'])
def getDatosRecentIDdv(iddv):
    # iddv contine el id para buscar
    listamedidas = []
    with db.connect() as conn:
```

```
        # Ejecuta la consulta y obtiene los resultados
        result = conn.execute(
            "SELECT ts,temp,hum FROM envtemphum " +
            "WHERE iddv='" + iddv + "' ORDER BY ts DESC LIMIT 300
"
        ).fetchall()
        # Convierte los resultados en una lista
        for row in result:
            listamedidas.append({
                'ts': str(row[0]),
                'temp': str((row[1])/100),
                'hum': str((row[2])/100),
            })
    return jsonify(listamedidas)
```

Igual que la anterior consulta con la base de datos y muestra los últimos 300 datos introducidos por este dispositivo consultado.

4.7 - Visualización de datos – Front-End

La implementación de este proyecto proporciona dos formas de visualizar los datos almacenados. Empleando un sitio web desplegado con la aplicación de Google desarrollado usando el framework Flask, ambas tecnologías son detalladas a continuación en el subapartado 4.5.2 o bien conectando de forma remota al servidor SQL con una instalación de Grafana, esta opción es explicada detalladamente en el punto 4.5.1.

4.7.1 Aplicación Grafana

4.7.1.1 Qué es Grafana

Grafana es una herramienta de software libre que ofrece soluciones de análisis y monitoreo de una base de datos sin importar donde esté almacena, mediante gráficos. En este caso se ha utilizado la base de datos que está creada en el Cloud SQL. Básicamente ofrece una herramienta donde poder visualizar y monitorizas mediante gráficos la información almacenada en la base de datos.

Fue diseñado por Torkel Ödegaard e implementada en el año 2014. Programada en Lenguaje Go (diseñado por Google) y Node.js LTS y con una fuerte Interfaz de Programación de Aplicaciones (API). Su código fuente está publicado en GitHub [18].

Grafana se puede ejecutar en GNU/Linux, Windows y MacOS y es muy utilizado por empresas tan conocidas como PayPal, Uber, etc.

Admite más de 30 fuentes de código abierto, así como bases de datos/fuentes de datos comerciales que incluyen MySQL, PostgreSQL, InfluxDB, Elasticsearch, OpenTSDB, Prometheus y Graphite.

Una visión de lo que puede representar Grafana es esta (Figura 4.81):

Figura 4.81 Visión general de Grafana

Ofrece gráficos elegantes, rápidos y flexibles con numerosas opciones, además de paneles dinámicos y reutilizables.

4.7.1.2 Instalando Grafana

La instalación de Grafana se ha realizado sobre un pc con un sistema operativo Windows 10 de 64 bits. Para la instalación se ha descargado el software desde su propia página web https://grafana.com/grafana/download, seleccionando la versión Código abierto, el archivo que se ha descargado para ejecutarlo es (grafana-6.7.3.windows-amd64.msi). (Figura 4.82)

Figura 4.82 Pantalla inicio de instalación de Grafana

Una vez ejecutado e instalado Grafana arranca un servicio para poder iniciar el servidor

4.7.1.3 Iniciando Grafana

Grafana instala su propio servidor HTTP y pone el puerto 3000 a la escucha. Por lo tanto, para iniciar Grafana, una vez iniciado el servidor, se abre una ventana del navegador y se escribe la dirección siguiente: http://localhost:3000, con esto ya se puede acceder a la interfaz web y dará acceso para poder introducir las credenciales de usuario. Estas son en el primer inicio Usuario: admin y Password: admin (Figura 4.83)

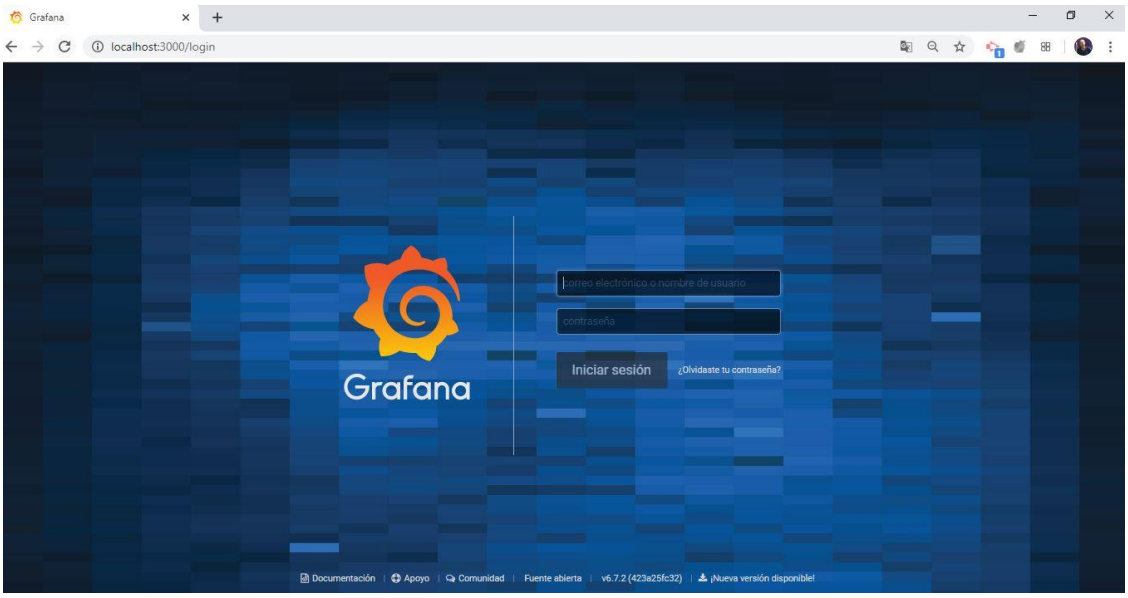


Figura 4.83 Entrada usuario Grafana

Una vez introducidos los datos de acceso la aplicación solicita una nueva contraseña para mayor seguridad.

La primera pantalla que se ve es la siguiente: (Figura 4.84)

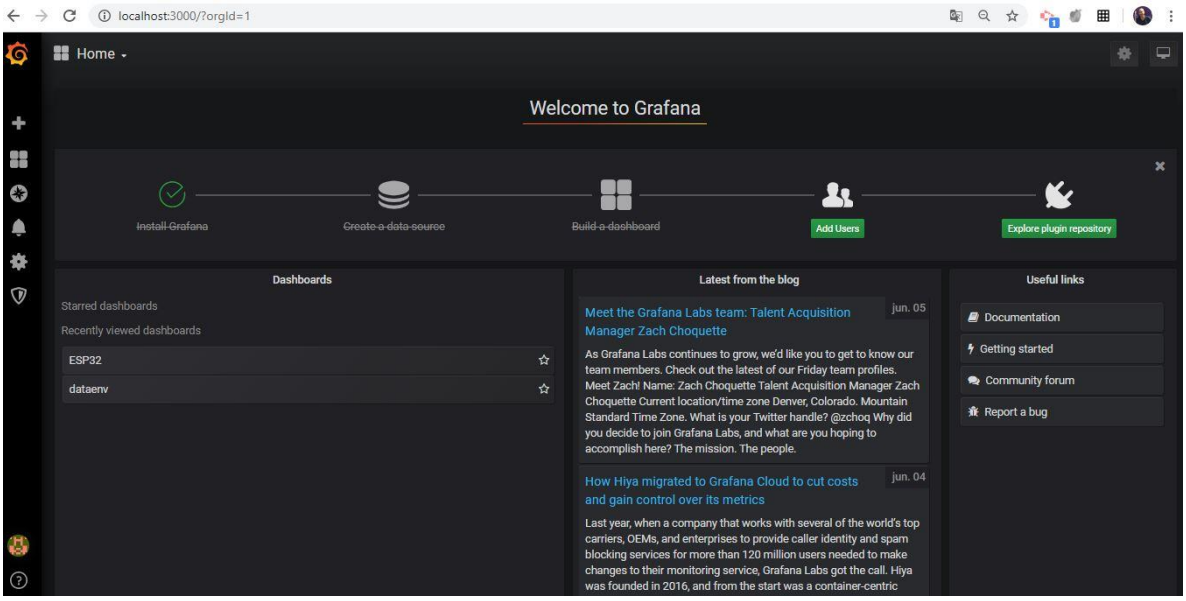


Figura 4.84 Pantalla inicial de Grafana

4.7.1.4 Visualización de los datos

Para visualizar datos en una gráfica, previamente se debe de definir donde se encuentran estos datos o, dicho de otra manera, cual es el origen de datos. A la izquierda existe un menú de configuración donde se puede pulsar “Data Sources” (Figura 4.85) y añadir el origen de datos. (Figura 4.86)

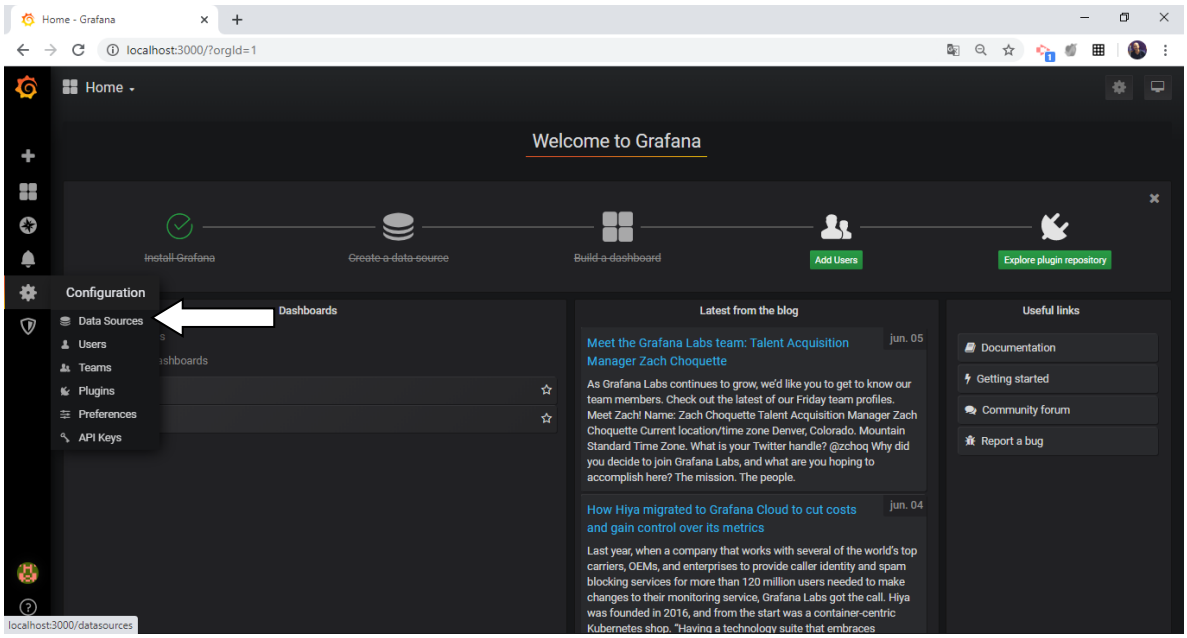


Figura 4.85 Menú Configuración-Data Sources en Grafana

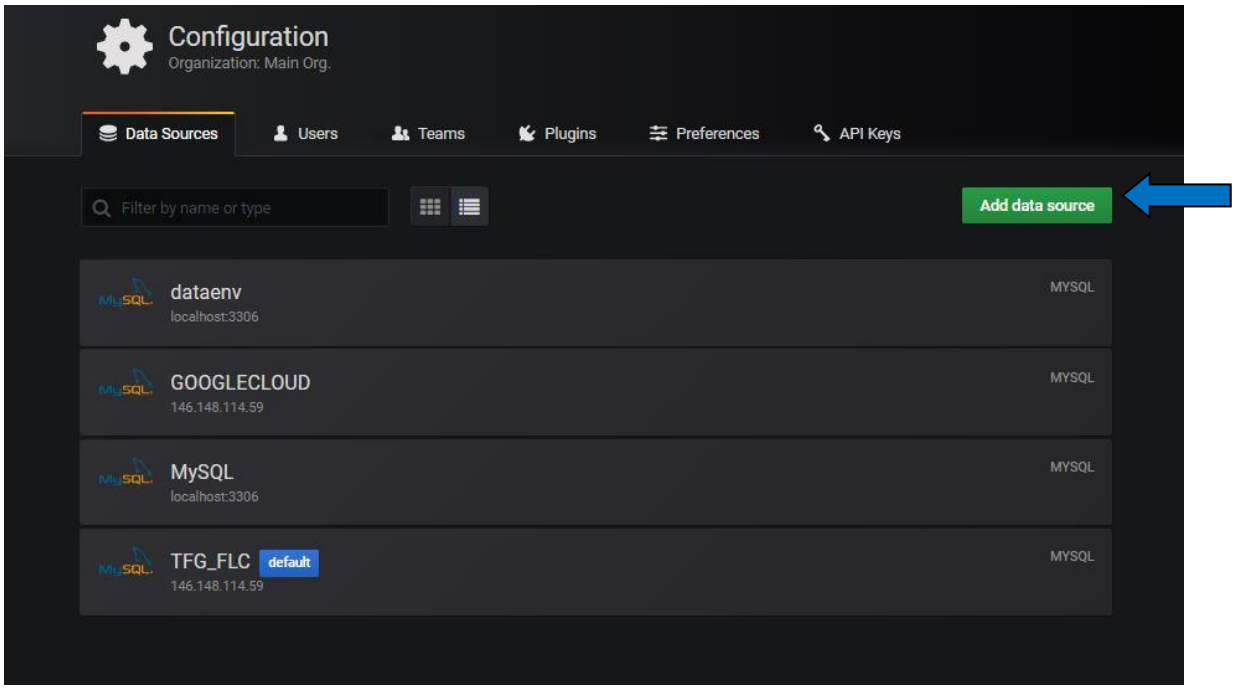


Figura 4.86 Añadir fuente de datos en Grafana

En este caso se ha realizado para el Cloud SQL que como se ha dicho anteriormente la instancia creada es de tipo MySQL. (Figura 4.87)

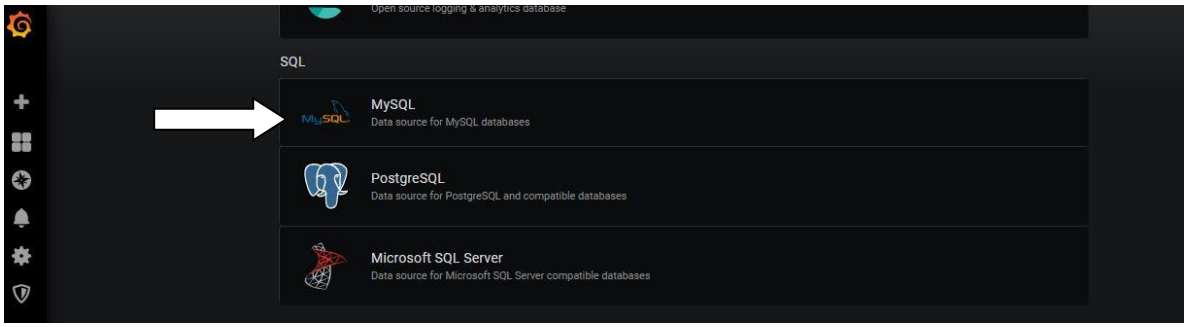


Figura 4.87 Tipos origen de datos de la instancia SQL

Elegido el tipo de origen de datos, se introducen los parámetros de configuración (Figura 4.88)

Figura 4.88 Configuración origen de datos de la instancia SQL

Los datos que se han introducido son los siguientes:

Name: Nombre que se le dará al origen de datos. En este caso es TFG_FLC

Host: Dirección donde está la instancia Cloud SQL.

Database: Nombre de la base de datos “data”

User: Usuario de la base de datos “root”

Password: Contraseña de la base de datos

Una vez introducidos se pulsa el botón de “Save and test” y si la conexión se ha realizado correctamente, debe aparecer un mensaje de “Database Connection OK” (Figura 4.89)

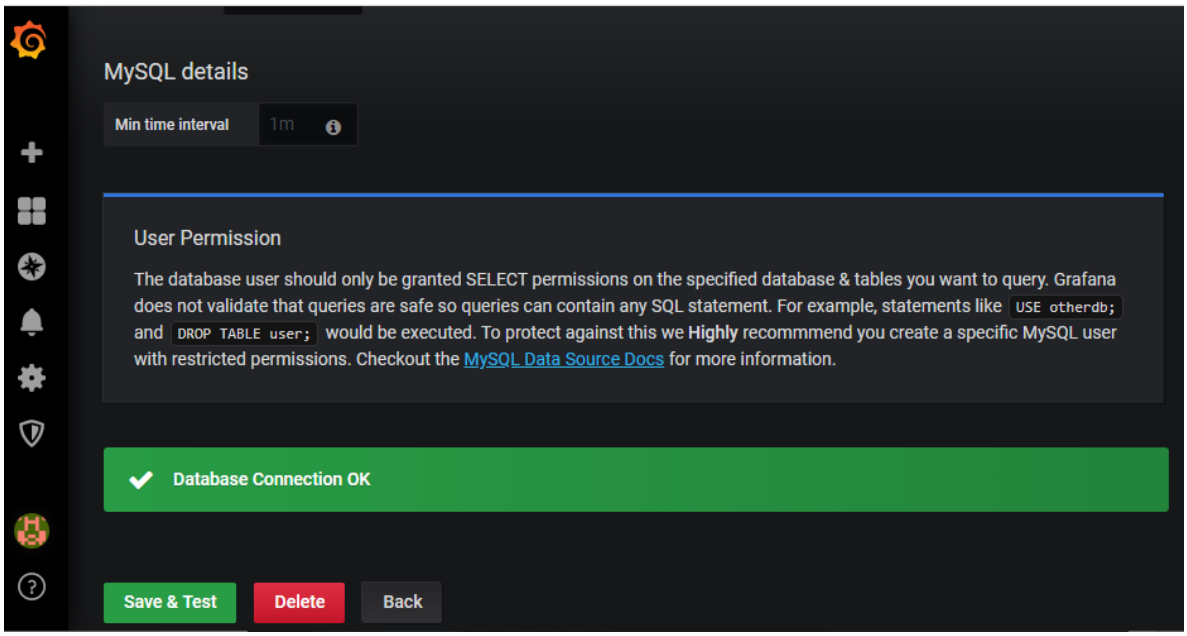


Figura 4.89 Base de datos conectada correctamente en Grafana

Una vez que ya está creado el origen de datos, se realiza la representación gráfica de los datos almacenados. Para ello en el menú de la izquierda se selecciona “Dashboard”, (Figura 4.90)

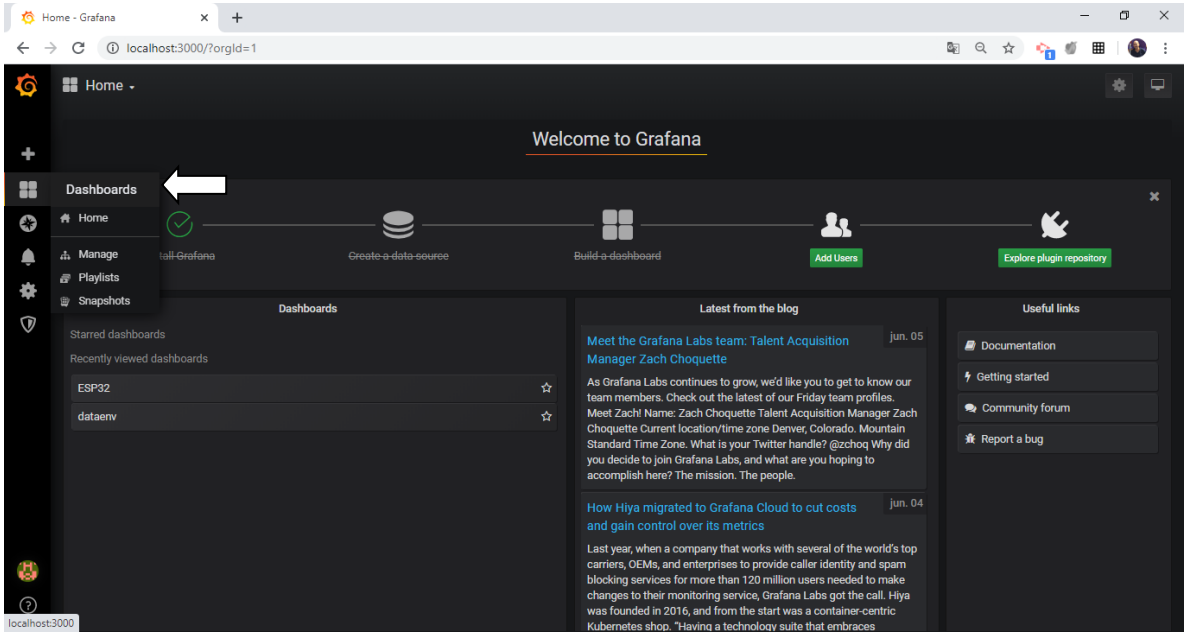


Figura 4.90 Crear un Dashboard en Grafana

Una vez pulsado este botón se muestra una ventana “New Panel” (Figura 4.91), se pulsa en el botón “Add Query” para crear una visualización de datos.

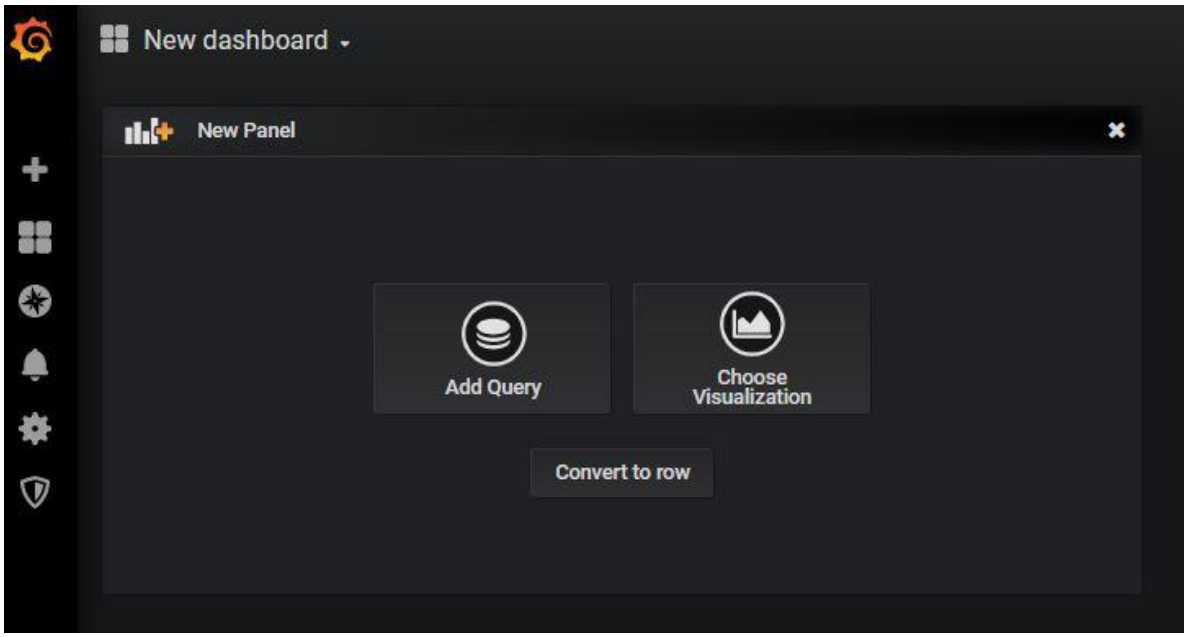


Figura 4.91 Nuevo panel para este Dashboard

Una vez pulsado sobre este botón se puede configurar este nuevo dashboard seleccionando el origen de datos en “Query” que para este caso es el creado anteriormente “TFG_FLC” y se selecciona el atributo que se quiere mostrar, en el campo del SELECT. Para este trabajo se muestran la temperatura y la humedad de diferentes clientes. Los clientes se pueden seleccionar mediante el parámetro “iddv” el campo de WHERE, (Figura 4.92) que para este caso es “30AEA4222150”. En el eje Y se mostrará los valores de temperatura y humedad y para el eje X el de tiempos.

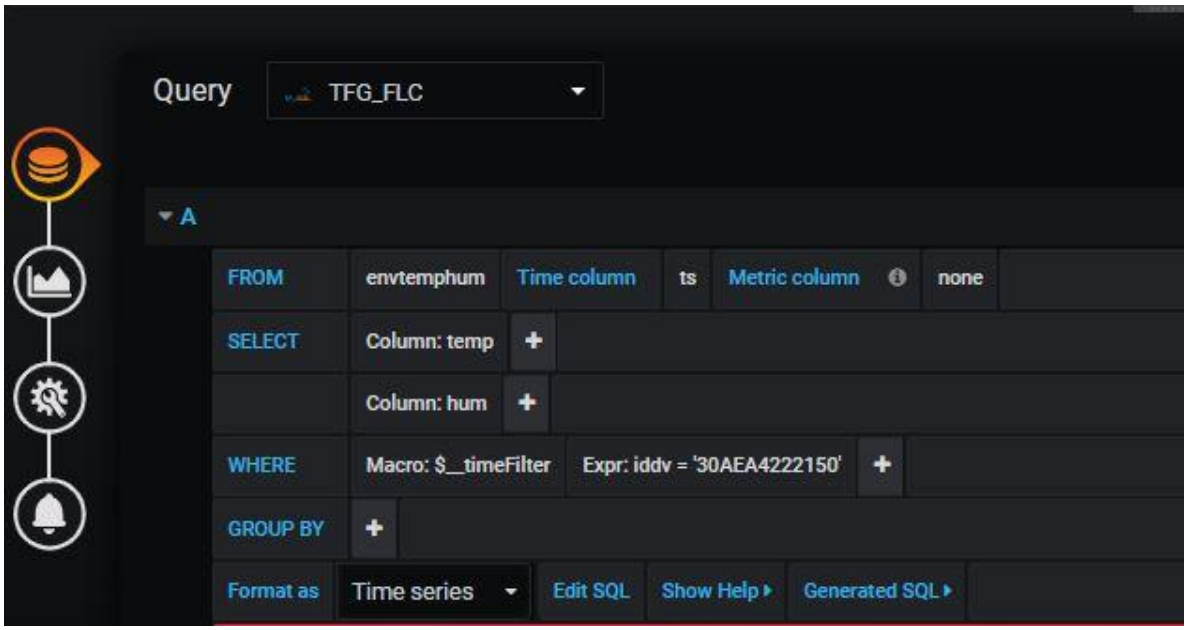


Figura 4.92 Selección de la consulta del Dashboard

Una vez realizada esta configuración, se guarda este dashboard (Figura 4.93) y se le pone un nombre (Figura 4.94), en este caso como el dispositivo cliente con la iddv=30AEA4222150" es el DHT11, pues se le pone este nombre.

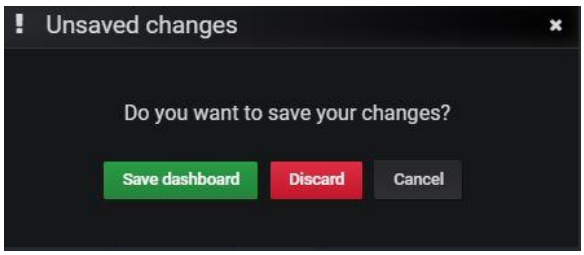


Figura 4.93 Guardar cambios del Dashboard creado

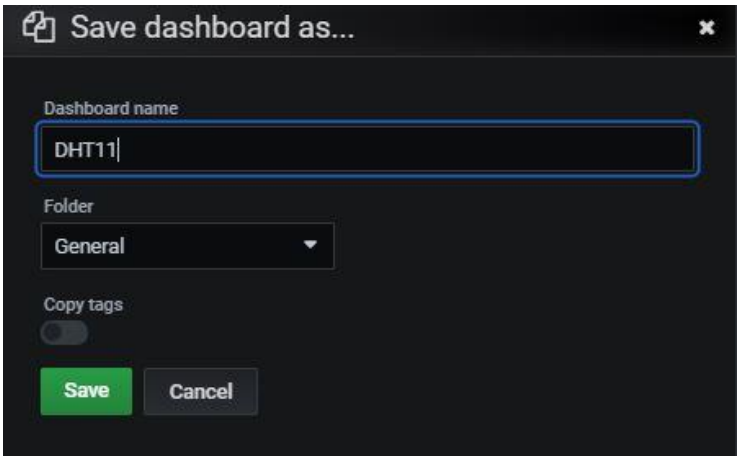


Figura 4.94 Nombre dado al Dashboard creado

Grafana muestra los datos de la instancia creada en Cloud SQL y filtrados mediante la selección anterior. (Figura 4.95)

Figura 4.95 Visualización de datos con Grafana

También es posible cambiar una serie de opciones de visualización, como puede ser el tipo de gráfico, los títulos de los ejes, nombre del panel. (Figura 4.96)

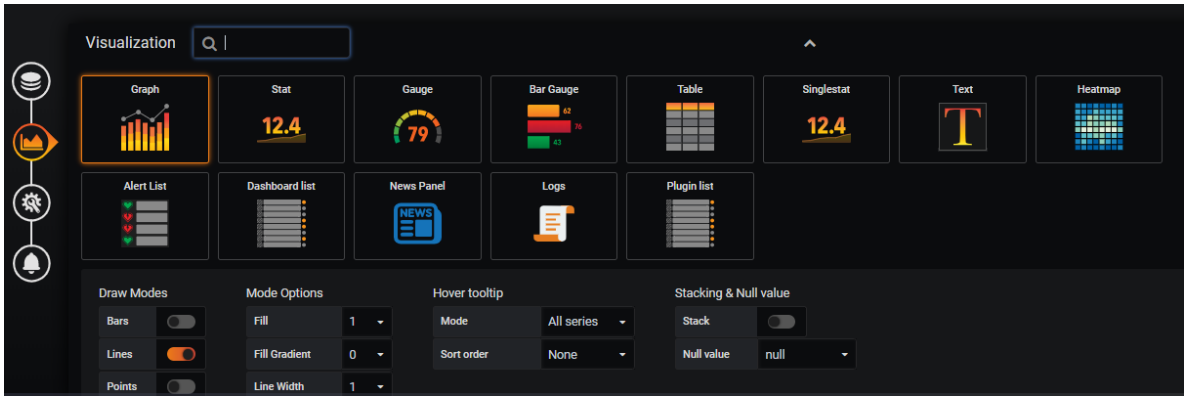


Figura 4.96 Opciones de visualización de gráficas con Grafana

4.7.2 App Engine – Sitio Web de visualización

La aplicación cargada en el servicio Cloud App Engine es desarrollada con los lenguajes Python y su framework Flask, ambos son detallados a continuación:

4.7.2.1 Python

Python nació a principios de los 90 y es un lenguaje de programación multiplataforma, de código abierto y orientado a objetos, que permite integrar sistemas de forma muy efectiva. Con una sintaxis fácil de leer y de usar, simplificando mucho el funcionamiento de la aplicación de que programe. Contiene una gran cantidad de tareas de programación como conectarse a servidores web, leer y modificar archivos, etc. [20].

Se puede ejecutar en cualquier sistema operativo Unix, MAC Os X, Windows, y Linux, y con alguna compilación sin ser oficial en iOS y Android.

Antes de desarrollar la aplicación web en App Engine, se ha programado el código en máquinas locales. Para esto se ha instalado PyCharm que es un IDE de Python que proporciona herramientas para desarrollar y programar en Python.

PyCharm dispone de tres versiones, Comunity y Edu que son de código abierto y gratuitas, y la versión Professional con bastantes más funciones que las anteriores y en formato comercial.

Antes de programar en Python con PyCharm, se ha descargado e instalado Python desde la web oficial www.python.org y elegir la plataforma deseada.

Una vez instalado Python, se ha descargado PyCharm e instalado en la máquina local para poder empezar a programar, las características mínimas para poder trabajar con este programa son de disponer de 4gb de memoria Ram así como 3,5 gb de almacenamiento en disco duro. La descarga se ha realizado desde la web oficial www.jetbrains.com/pycharm/download eligiendo la plataforma del sistema operativo del equipo, que para este caso será Windows. La versión que se ha descargado es la Comunity y el archivo descargado es este: pycharm-community-2019.3.4.exe.

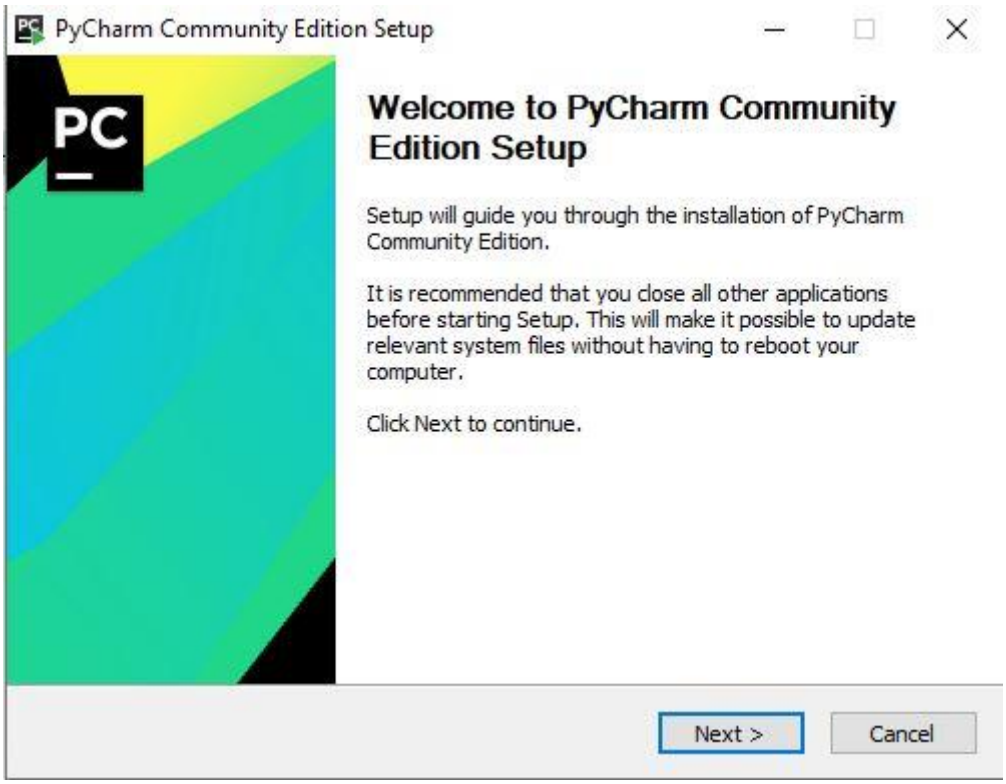


Figura 4.97 Instalación de Pycharm

Pulsando el botón “Next” (Figura 4.97) comienza la instalación del programa, donde solo se tiene que especificar la ruta de instalación. Una vez instalado la pantalla principal de inicio es (Figura 4.98)

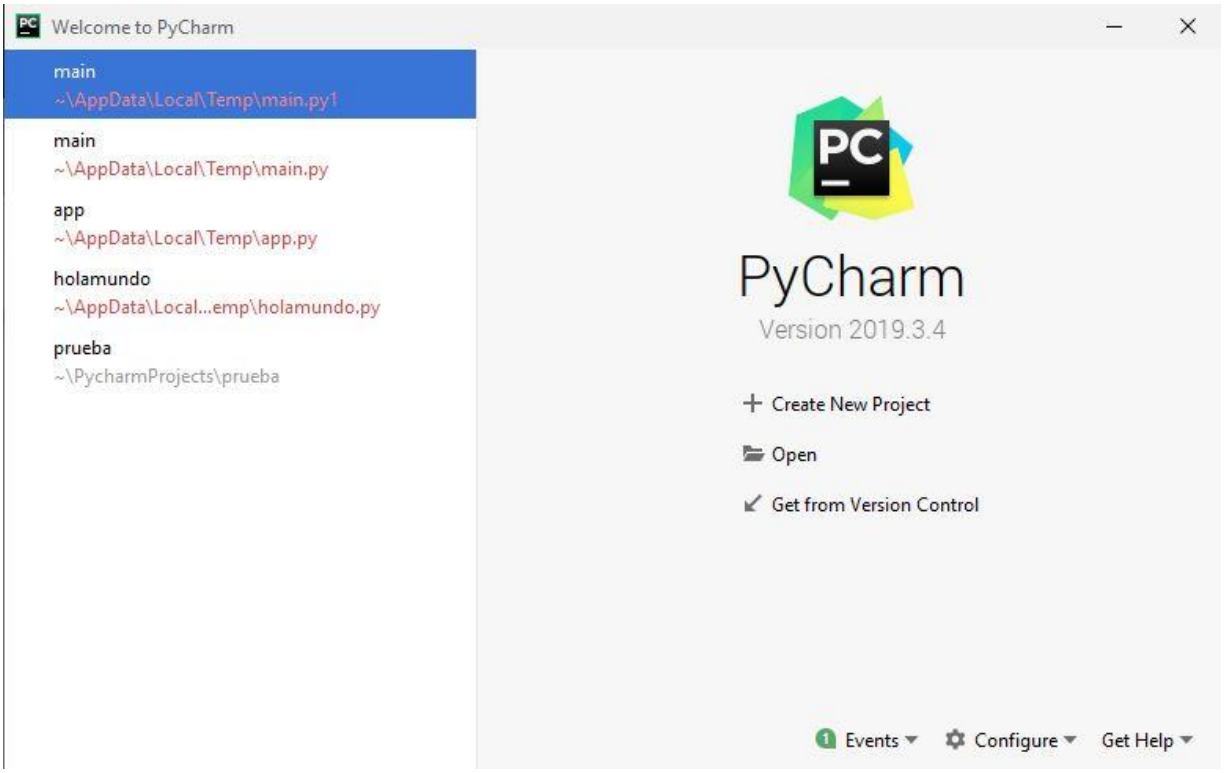


Figura 4.98 Ventana inicio PyCharm

En esta pantalla de inicio, se puede crear un nuevo proyecto, abrir uno existente o abrir las opciones de configuración del programa. La pantalla inicial para un nuevo proyecto puede verse en la siguiente imagen (Figura 4.99).

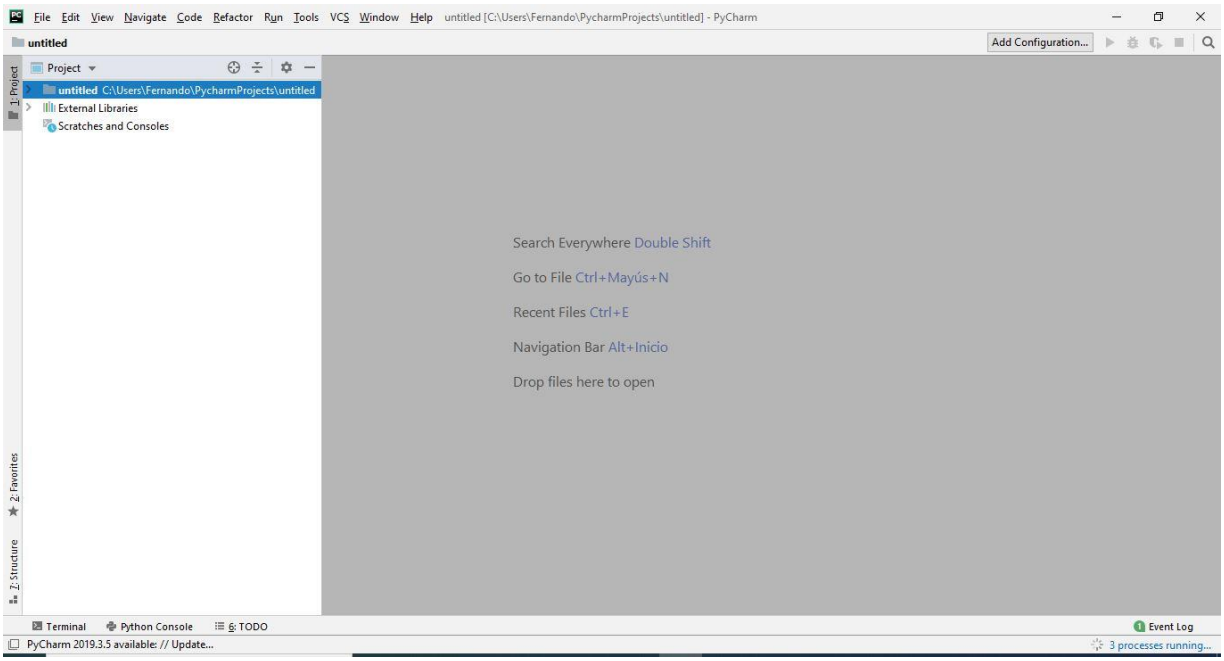


Figura 4.99 Ventana nuevo proyecto

Para empezar a programar se tiene que crear un nuevo archivo, ponerle un nombre y elegir el formato de Python. Una vez creado se genera un área para la programación. (Figura 9.100)

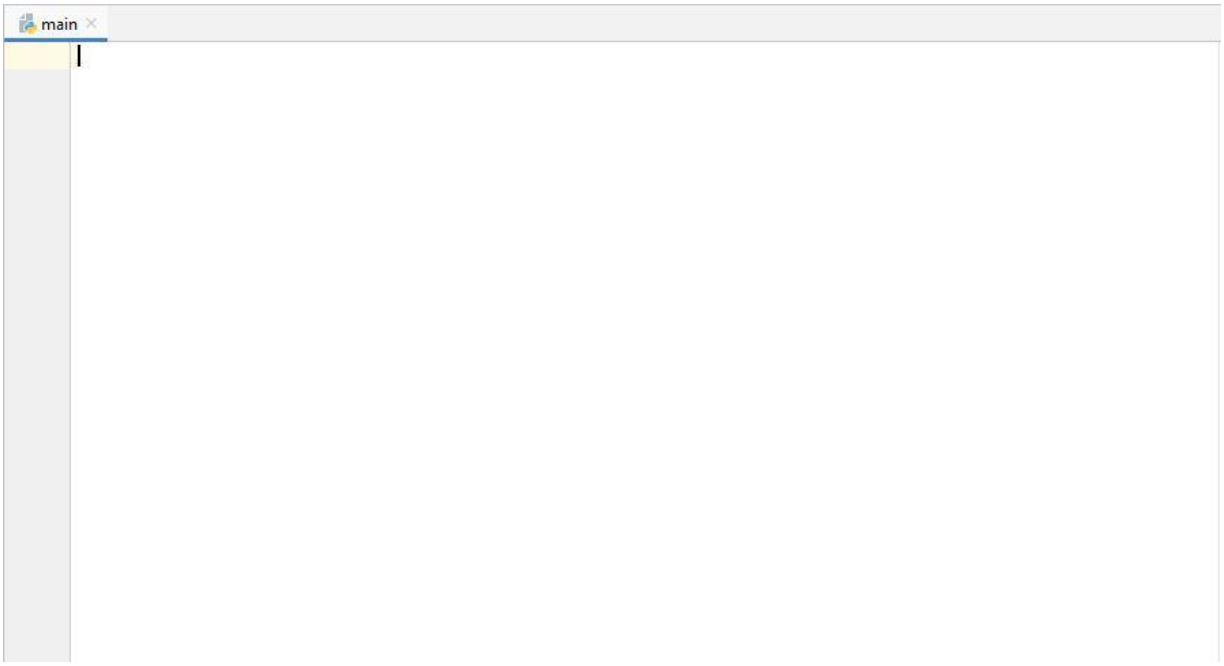


Figura 4.100 Área de programación

4.7.2.2 Flask

Flask es un framework de Python para crear aplicaciones web que se basa en el patrón Modelo-Vista-Controlador. Un framework es un entorno que proporciona un marco de trabajo y una serie de utilidades y funciones que ayudan a construir aplicaciones web [21].

5. RESULTADOS Y DISCUSIÓN

5.1 Presentación de los datos App Engine – Front-End

Para realizar una representación de los datos obtenidos de los sensores y poder tomar decisiones de actuación dependiendo de estos, se utilizará el código desarrollada con App Engine.

La pantalla principal se puede visualizar en la Figura 5.1.:

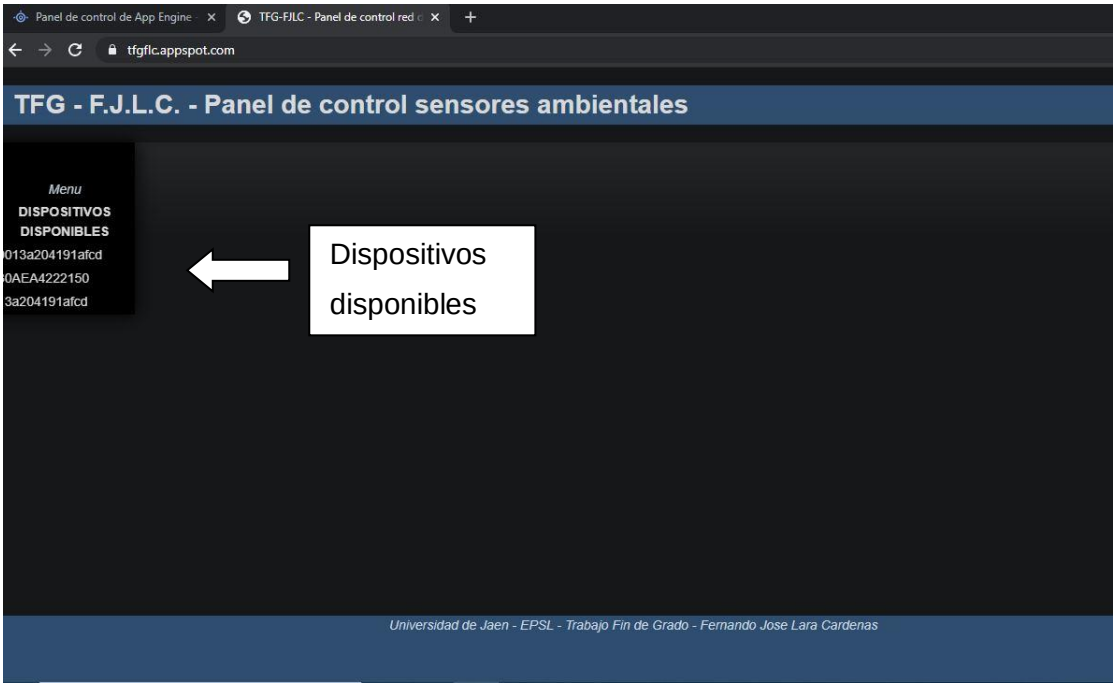


Figura 5.1 Pantalla principal de la aplicación web creada con App Engine

En la aplicación aparece una cabecera con el título de la web. En la parte izquierda aparece un menú de dispositivos disponibles y son cada uno de los dispositivos clientes que han enviado algún dato a Google Cloud. Pulsando en uno de los dispositivos aparecen las gráficas con los datos enviados. En la parte del pie de la página aparece los datos de este trabajo.

Si se pulsa sobre el dispositivo “30AEA4222150” deben aparecer los datos del sensor/cliente DHT11 (Figura 5.2). Como se ha explicado anteriormente este sensor envía dos tipos de datos, temperatura y humedad, por lo tanto, deben aparecer dos medidas en la gráfica.

La gráfica muestra los valores con una serie de puntos, cada punto corresponde con un valor enviado por el sensor/cliente. Además, la aplicación muestra en gráficos circulares el último valor enviado/leído por este sensor. El gráfico circular en color rojo mostrará la temperatura y el gráfico en color azul la humedad.

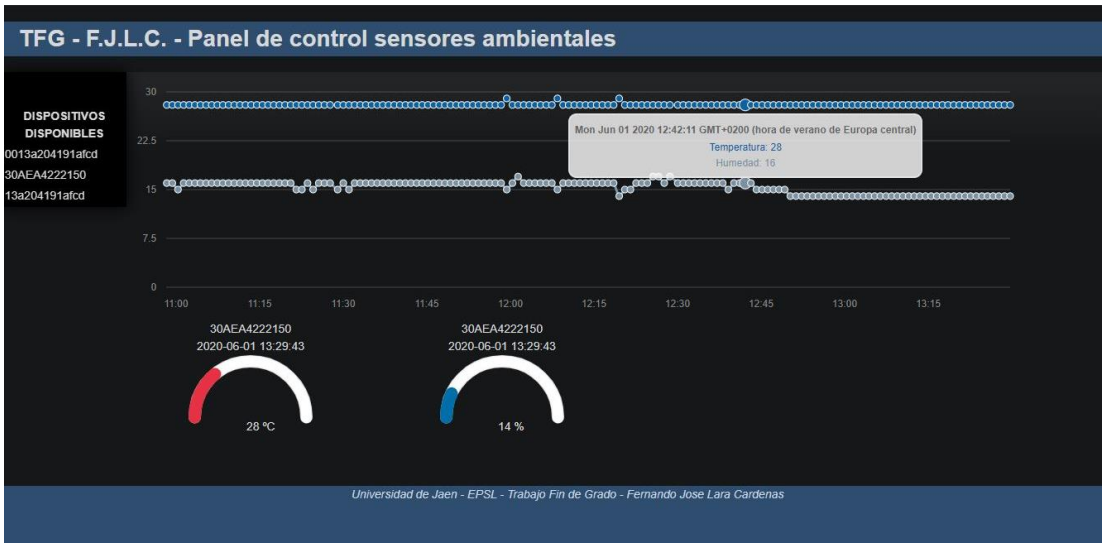


Figura 5.2 Datos temperatura y humedad del sensor/cliente DHT11

En la Figura 5.3 pueden verse los datos del sensor/cliente TMP36 enviada por el Xbee con iddv="13a204191afcd". Este, solo envía datos de temperatura por lo tanto solo aparece un tipo de dato. Igual que el anterior la gráfica es por puntos y para el último dato enviado por este sensor se mostrará en un gráfico de tipo circular. Como se puede observar solo muestra el de color rojo ya que como se ha dicho anteriormente solo envía temperatura.

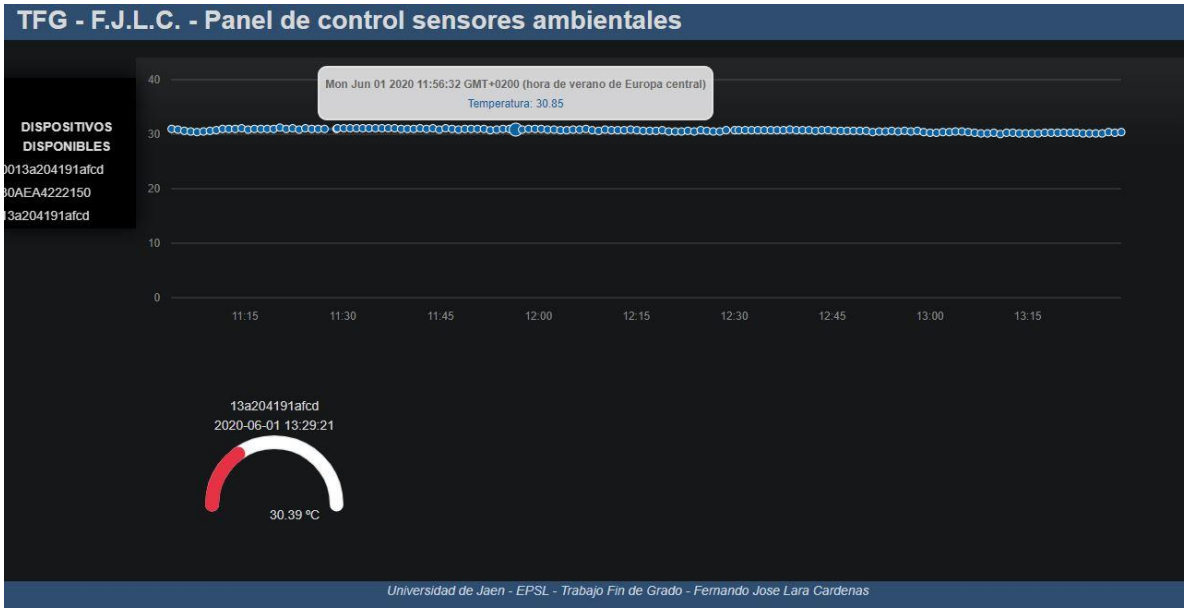


Figura 5.3 Datos temperatura enviados por el Xbee con el sensor TMP36

La aplicación esta accesible desde cualquier lugar en la dirección siguiente: https://tfgflc.appspot.com

5.2 Presentación de los datos con Grafana

En Grafana se pueden crear muchos paneles con gráficas como se ha explicado anteriormente uno de ellos puede ser la mostrada en la Figura 5.4 que muestra los valores en conjunto de todos los clientes. Se aprecian dos líneas de puntos verdes, que son datos de temperatura y que equivalen a los dos sensores que envían datos. Las líneas amarillas son datos de humedad, como uno de ellos no envía humedad, pues unas de las líneas están a cero.

Figura 5.4 Datos de los dos sensores solapados con Grafana

También se han creado paneles con gráficas con los datos enviados por un solo sensor, en este caso el DHT11 (Figura 5.5). Como se puede apreciar aparecen los datos de temperatura en color verde y los de humedad en color amarillo. Indicar, ahora que se aprecian más los datos, que al enviar un dato cada minuto en muchas ocasiones si el dato del minuto anterior es el mismo que el del siguiente en lugar de aparecer puntos puede apreciarse como una línea, pero en realidad son puntos. En la Figura 5.6 se puede apreciar mejor esos puntos, ya que existe una mayor dispersión de estos.

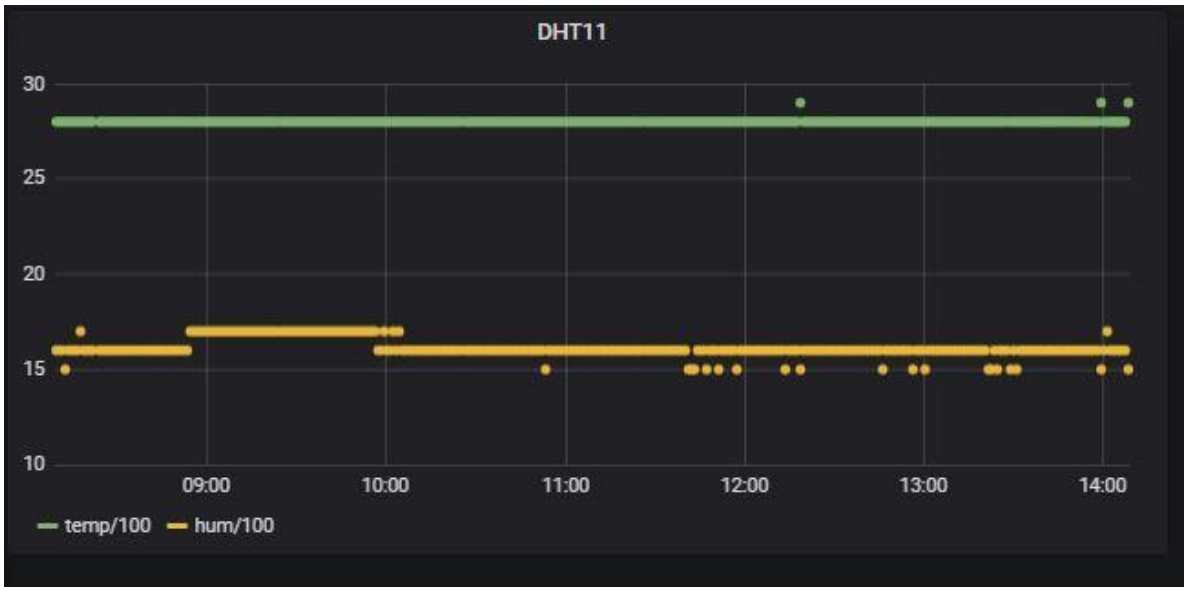


Figura 5.5 Datos Temperatura y humedad del sensor DHT11 con Grafana

En la Figura 5.6 se puede apreciar los datos enviados por el sensor TMP36 conectado al Xbee. La línea de puntos es de color verde y como se ha explicado anteriormente solo envía temperatura. En este ejemplo se puede ver mejor la dispersión de puntos, ya que existe más disparidad de valores. En esta gráfica se ha realizado un zoom más grande para que se pueda aprecia mejor los puntos.

Figura 5.6 Datos enviados por el sensor TMP36 con Grafana

En la Figura 5.7 se puede apreciar los dos paneles juntos de los dos sensores distintos que contiene la red creada.

Figura 5.7 Datos con dos paneles de los dos sensores con Grafana

5.3 Comparaciones

Con las representaciones de App Engine y Grafana se ha querido mostrar la similitud a la hora de visualizar los datos con dos aplicaciones diferentes. Una de ellas, Grafana, sin necesidad de tener conocimientos de programación y solo conociendo los datos de acceso a la base de datos donde están alojados estos.

5.4 Estudio económico

Una parte importante en el estudio de viabilidad de un proyecto es el aspecto económico del mismo, puesto que puede ser técnicamente correcto, pero económicamente inviable. Partiendo de esta base se detalla a continuación un estudio económico centrándose en el coste mínimo de implantación del proyecto, coste de mantenimiento mensual, así como una estimación del coste al incrementar volumen de tráfico.

Se distinguen varios capítulos económicos:

Mano de obra: compensa el coste de desarrollo del proyecto, así como su prototipado y especificación para paso a producción.

Mano de obra	Cantidad	Precio	Importe
Horas de Ingeniería	300	18	5.400,00 €
Horas técnico instalación	0,5	8	4,00 €

Tabla 5-1 Mano de obra proyecto

Hardware: componentes electrónicos necesarios.

Coste de hardware	Cantidad	Precio	Importe	Proveedor
Sensor DHT11	1	6,11	6,11	https://es.rs-online.com/web/p/kits-de-desarrollo-de-sensores/1743237/

Sensor TMP36	1	1,546	1,546	https://es.rs-online.com/web/p/sensores-de-humedad-y-temperatura/0427351/
Dispositivo ESP32	1	15,05	15,05	https://es.rs-online.com/web/p/modulos-wlan/1697594/?sra=pmpn
Dispositivo Xbee	2	23,07	46,14	https://es.rs-online.com/web/p/kits-de-desarrollo-de-radio-frecuencia/1359730/
Total			**68,85 €**	

Tabla 5-2 Coste Hardware

Hardware de uso requerido en el proyecto, pero descartado por inviable (Este total supone un ahorro respecto de los requisitos iniciales):

Coste de hardware	Cantidad	Precio	Importe	Proveedor
Arduino UNO	1	22,54	22,54	https://es.rs-online.com/web/p/kits-de-desarrollo-de-procesador-y-microcontrolador/7697409/
Shield Arduino Ethernet	1	25,8	25,8	https://es.rs-online.com/web/p/kits-de-desarrollo-de-procesador-y-microcontrolador/8732285/
Total			**- 48,34 €**	

Tabla 5-3 Coste Hardware requisitos iniciales

Coste de pack de batería para los dispositivos que no tienen opción de conexión a alimentación. Uno por cada dispositivo.

Coste paquete batería	Cantidad	Precio	Importe	Proveedor
Convertidor dc - dc no aislado, Salida 5V dc, 1.5A, 7.5W	1	3,43	3,43	https://es.rs-online.com/web/p/reguladores-de-conmutacion/1676213/
Batería recargable de Ión-Litio, 7.4V, 5.2Ah, 4 celdas 78 x 68 x 19 mm, terminación en cable	1	39,97	39,97	https://es.rs-online.com/web/p/baterias-recargables/1449412/
Total			**43,4 €**	

Tabla 5-4 Coste pack batería

Servicios Cloud: servicios de procesamiento de información ubicados en internet. Para su utilización es necesaria la existencia de una conexión a internet. Dada la naturaleza de estos servicios, se ha empleado la calculadora de precios proporcionada por el Google para su plataforma con la configuración mínima de los servicios utilizados.

https://cloud.google.com/products/calculator/#id=f58faa30-e5cb-4308-a0ad-bd253b2ea073

Your Estimated Bill *

Estimated Monthly Cost: USD 443.54

Servicio	Detalle	Almacenamiento	Importe	Moneda
Iot Core	Data Exchange	1024 GiB	3.48	USD
Cloud Functions	AddBD	80000	8.77	USD
Pub/Sub	Message Volume	1 GiB	0.00	USD
Pub/Sub	Retained Acknowledged Backlog	1 Gib	0.27	USD
Pub/Sub	Snapshot Backlog	1 Gib	0.27	USD
Dataflow	1 x n1-standard-1 workers in Batch Mode	1	0.07	USD
Almacenamiento muestreos	Total Number of instance	730h	430.61	USD
	BigQuery	1 Gib	0.05	USD
App Engine standard environment	Instances	730 Instance Hours	0.00	USD
		Total Estimated Monthly Cost	**443.54**	**USD**

Tabla 5-5 Coste mensual Servicios Cloud

Coste implantación:

Compensación del coste de mano de obra de la instalación en campo, así como del desarrollo del proyecto. No se incluye desembolso de servicios Cloud puesto que no tiene coste de contratación inicial solamente por uso.

- Mano de obra e instalación: 5404,00 €
- Coste de hardware: 68,85 €
- Coste de baterías: 86,80 €
- **Total** **5.559,65 €**

Coste de mantenimiento por mes:

Coste de los servicios Cloud principalmente, puesto que el intervalo de mantenimiento de los sensores es de larga duración. Los paquetes de baterías son recargables de litio, lo que implica que solo es necesario sustituir por un paquete cargado para realizar una recarga. Se estima duración de batería superior a un año.

- **Coste mensual de servicios de Google Cloud*** **393,96 €**

** Es un coste estimado puesto que varía en función del uso y demanda del servicio. Este es el coste máximo que podrían tener todos los servicios funcionando sin parar a lo largo de un mes*.

Se observa que el coste de mayor repercusión en el proyecto lo supone el servicio de base de datos, no obstante, es el más crítico en cuanto al rendimiento, seguridad, puesto que alberga toda la información y escalado en función de la demanda. Por lo tanto, es

recomendable el empleo de este servicio virtualizado porque no es solamente el hecho de almacenar la información porque se está pagando. Y aunque la simulación esta realizada con la instalación mínima viable, el proyecto está orientado a la gestión de miles e incluso millones de sensores.

Considerando esta observación del funcionamiento de facturación de los servicios Cloud se realiza una segunda simulación donde la base de datos almacenaría 1TB, y se observa que el incremento del coste no es significativo en comparación con el incremento del volumen simulado. Esto representa que mayor coste es el coste mensual inicial, considerando que el entorno Cloud esta dimensionado principalmente para grandes demandas de volumen el coste mensual inicial es asumible.

Almacenamiento muestreos	Total Number of instance	730h	430.61 USD	10GB (mínimo)
Almacenamiento muestreos	Total Number of instance	730h	598.91 USD	1 TB

Tabla 5-6 Estimación segunda simulación

6. CONCLUSIONES

El presente proyecto establece una infraestructura flexible que permite su dimensionamiento dinámico y en cortos periodos de tiempo sin necesidad de realizar inversiones en infraestructura adicionales para ello.

El sistema gracias al soporte de la tecnología Cloud es posible configurarlo en un dimensionamiento dinámico. Las cantidades de información generadas son grandes por lo tanto el mayor coste se deriva de su almacenamiento, el sistema de base de datos albergado en el Cloud, y por otro lado y no menos importante a medio y largo plazo el procesado de este gran volumen de información para la toma de decisiones, aprendizaje, estimaciones entre otras posibilidades esto está contemplado en el proyecto implementando los servicios BigQuery y DataFlow.

El proyecto presenta un planteamiento básico de una infraestructura mínima global que realiza la implementación inicial de unos servicios con gran potencial y que permite su adaptación rápida según las demandas del usuario o la aplicación del mismo en un corto periodo de tiempo y reduciendo al máximo tanto la inversión necesaria en infraestructura como los costes fijos asociados al sistema.

El planteamiento modular del proyecto permite la actualización de una de las partes o bloques, sin que el resto de la implementación se vea afectado, esto permite que el proyecto sea viable durante mayores periodos de tiempo puesto que el sector tecnológico y en especial el sector de las comunicaciones está sujeto a cambios constantes y en periodos relativamente cortos de tiempo.

Dos formas de acceder a información

BigQuery y SQL

Dos formas demostradas de visualización

App Engine sitio adaptable

Conexión y consulta directa SQL - Grafana

En resumidas líneas, entre proyecto despliega la infraestructura para la implementación de una red escalable de sensores de ágil configuración que permite auto reconocer dispositivos nuevos apoyándose en servicios Cloud Computing.

7. LÍNEAS FUTURAS

Con el despliegue actual el proyecto establece una infraestructura que posibilita el muestreo masivo a nivel global, actualmente de hasta dos magnitudes, temperatura y humedad, permitiendo identificar de manera concreta el dispositivo en el que fueron tomadas, así como el terminal que realizo su registro en el sistema.

Analizando los dispositivos empleados, se disponen en todos los elementos de hardware de conectividad para la inclusión de sensores adicionales, o bien actuadores. Estos podrán ser tanto analógicos o digitales bien conectados directamente o con el soporte de electrónica adicional.

La principal característica que aporta interés de expansión a este proyecto es la flexibilidad. Flexibilidad en cuanto al número de dispositivos que realicen muestreos, en el despliegue y en la conectividad tanto para entrada, escenario demostrado en la configuración actual para tanto para captadores digitales y analógicos.

Para el despliegue del presente sistema de manera funcional solamente es necesaria la existencia de acceso a internet por vía wifi, y colocar el nodo coordinador en el rango de cobertura de esta. Desde aquí el rango de cobertura de este nodo empleando tecnología ZigBee se extiende de nodo a nodo permitiendo comunicar a los nodos más extremos con el nodo coordinador.

Considerando lo citado anteriormente, el proyecto permite amplio rango de aplicaciones prácticas reales.

Por ejemplo combinando actuadores (relés/triac) con sensores de movimiento y luminosidad, es viable establecer una red inteligente de iluminación urbana o interurbana, permitiendo que las luminarias permanezcan apagadas cuando no exista movimiento y al detectarse el movimiento active la iluminación y esta a su vez transmita la información al resto de nodos permitiendo la iluminación del tramo de carretera interurbana antes de que el vehículo llegue a la zona lo que permite que el usuario no sea consciente de que la iluminación estaba apagada antes de su llegada y permite facilitar un campo de visión igual que si hubieran estado permanentemente encendidas.

Así mismo esta información se almacena y permite su posterior análisis para poder dimensionar vías en función del uso, densidad de tráfico y franjas horarias.

Es posible dotar a los nodos de sensores de calidad de aire, o bien aplicarlo en campos como la agricultura para establecer un mayor aprovechamiento de los recursos hídricos como ya se está haciendo en invernaderos y cultivos a pequeña escala pero con posibilidad de aplicar a campo abierto, permitiendo documentar el entorno ambiental al que ha estado expuesto durante todo el cultivo una planta en concreto y con el uso herramientas adicionales facilitadas por el Cloud de Google no usadas en el presente proyecto para

análisis de datos aplicando modelos e inteligencia artificial establecer medidas para incrementar producción con menores recursos año tras año.

La próxima implantación de 5G e incluso el actual LTE facilita el despliegue del sistema de monitorización en cualquier ubicación donde exista cobertura de estas bandas y de forma segura. Esto es posible por el empleo de Cloud como sistema de almacenamiento y procesado de información en lugar de emplear servidores propios locales.

Dejando a un lado la expansión del proyecto en el ámbito del hardware. El sistema recolecta una cantidad de información realmente grande. Lo que posibilita de forma rápida la aplicación de los módulos de Inteligencia Artificial (IA) que facilita Google para realizar análisis, predicciones sobre los datos registrados y permitir aprender sobre la información recogida, lo cual nos lleva a un campo inmenso en expansión actualmente.

Estos módulos de Google de IA se apoyan sobre BigQuery herramienta que está ya implementada en la configuración actual del proyecto.

Cualquiera de los escenarios planteados, actualmente, permitiría ser implementado partiendo de la solución actual en un muy corto periodo de tiempo, desde horas hasta solo días.

Por otro lado, como alternativa a la visualización de la información, existe la posibilidad de implementación de una app para la visualización de la información, la aplicación puede ser desarrollada para cualquier plataforma independientemente del lenguaje que sea necesario emplear para su implementación, ya que accederá a la información empleando los servicios REST.

Con perspectivas de futuro, la evolución del hardware y el desarrollo de un sistema estrechamente vinculado a él, está condenado inevitablemente a la obsolescencia, sin embargo, el presente proyecto está desarrollado sobre dispositivos compatibles con Arduino, y otros elementos electrónicos de propósito general esto apoyado sobre protocolos estándar, permite que el proyecto pueda ser implementado utilizando hardware que reúnan características similares si llegara a estar discontinuado o retirado. Viéndose afectado solamente la parte de los dispositivos de hardware y permitiendo coexistir con dispositivos ya existentes en la red.

8. BIBILIOGRAFÍA

[1]. **HARDWARE LIBRE.** *DHT11: todo sobre el sensor para medir temperatura y humedad***.** Disponible en: https://www.hwlibre.com/dht11/ [Consulta: 12-01-2020]

[2]. **ANALOG DEVICE.** *TMP36 Sensor.* Disponible en: https://www.analog.com/en/products/tmp36.html [Consulta: 12-01-2020]

[3]. **THE INTERNET OF THINGS WITH ESP32.** *Features and Specifications ESP32.* Disponible en: https://www.esp32.net. [Consulta: 12-01-2020]

[4]. **DIGI.** *Digi Xbee Ecosytem.* Disponible en: https://www.digi.com/xbee. [Consulta: 12-01-2020]

[5]. **PROMETEC.** *El despertar del ESP 32, pines.* Disponible en: https://www.prometec.net/blog-curso-esp32-pin-tactil/. [Consulta: 13-01-2020]

[6]. **DOMODESK.** *A fondo Zigbee.* Disponible en: https://www.domodesk.com/216-a-fondo-zigbee.html. [Consulta: 14-01-2020]

[7]. **Y.C. TSENG.** *Zigbee/IEEE 802.15.4 Overview.* Disponible en: http://ocw.nctu.edu.tw/course/ws992/E1.pdf. [Consulta: 14-01-2020]

[8]. **DIGI.** *XCTU. Next Generation Configuration Platform for XBee/RF Solutions.* Disponible en: https://www.digi.com/products/embedded-systems/digi-xbee/digi-xbee-tools/xctu. [Consulta: 02-02-2020].

[9]. **JSON.** *Introducción a JSON.* Disponible en: https://www.json.org/json-es.html. [Consulta: 17-02-2020].

[10]. **MQTT.** *MQTT*. Disponible en: https://www.mqtt.org [Consulta: 17-02-2020]

[11]. **GOOGLE.** *Cloud IoT Core.* Disponible en: https://cloud.google.com/iot-core?hl=es [Consulta: 05-02-2020]

[12]. **GOOGLE.** *Cloud Functions.* Disponible en: https://cloud.google.com/functions?hl=es [Consulta: 05-02-2020]

[13]. **GOOGLE.** *BigQuery.* Disponible en: https://cloud.google.com/bigquery?hl=es [Consulta: 05-02-2020]

[14]. **GOOGLE.** *Dataflow.* Disponible en: https://cloud.google.com/dataflow?hl=es [Consulta: 05-02-2020]

[15]. **GOOGLE.** *Cloud SQL.* Disponible en: https://cloud.google.com/sql?hl=es [Consulta: 06-02-2020]

[16]. **GOOGLE.** *App Engine.* Disponible en: https://cloud.google.com/appengine?hl=es [Consulta: 06-02-2020]

[17]. **FIELDING, ROY THOMAS.** *Architectural Sytles and the Designo of Network-based Software Architectures.* Doctoral dissertation, University of California, Irvine, 2000.

Disponible en: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm [Consulta: 14-02-2020]

[18]. **GRAFANA LABS.** *Grafana Documentations.* Disponible en: https://grafana.com/docs/grafana/latest/ [Consulta: 15-02-2020]

[19]. **MYSQL.** *MySQL 8.0 Reference Manual.* Disponible en: https://dev.mysql.com/doc/refman/8.0/en/ [Consulta: 15-02-2020]

[20]. **JETBRAINS.** *PyCharm.* Disponible en: https://www.jetbrains.com/es-es/pycharm/ [Consulta: 17-02-2020]

[21]. **PALLETS.** *Flask web development, one drop at a time.* Disponible en: https://flask.palletsprojects.com/en/1.1.x/ [Consulta: 17-02-2020]

9. ANEXOS

9.1 Código fuente App Engine

9.1.1 main.html

El código fuente que ejecutará el archivo principal de la web será el siguiente, como podemos apreciar que contiene etiquetas HTML5, además ejecuta código JavaScript (“main.js”) y hojas de estilos (“style.css”)

```
<html>
    <head>
        <title>TFG-FJLC - Panel de control red de sensores</title>
        <meta name="description" content="">
        <link rel="stylesheet" href="assets/style.css">
        <link rel="stylesheet" href="assets/morris.css">
        <script src="assets/jquery.min.js"></script>
        <script src="assets/raphael-min.js"></script>
        <script src="assets/morris.min.js"></script>
        <script src="assets/gauge.min.js"></script>
        <script src="assets/main.js"></script>
    </head>
    <body onload="listDv()">
        <header>
            <h1>TFG - F.J.L.C. - Panel de control sensores
ambientales </h1>
        </header>
        <nav class="sidemenu sidemenu__top">
            <div id="cnt"></div>
        </nav>
        <main class="main-view">
            <section id="sec_graf">
                <article id="graf_dev"></article>
            </section>
            <section id="sec_gauge">
                <article id="gauge_temp">
                    <div id="gTemp">
```

JavaScript fuente externa

Hoja de estilos utilizada para representar las gráficas

Llamada a la función que carga los dispositivos, para que aparezcan en el menú

```
                        <label id="ID_gaugeTemp">ID DE
DISPOSITIVO</label>
                        <canvas id="gaugeTemp"></canvas>
                        <label id="lbl_gaugeTemp">[temp]</label>
                    </div>
                </article>
                <article id="gauge_hum">
                    <div id="gHum">
                        <label id="ID_gaugeHum">ID DE
DISPOSITIVO</label>
                        <canvas id="gaugeHum"></canvas>
                        <label id="lbl_gaugeHum">[hum]</label>
                    </div>
                </article>
            </section>
        </main>
        <footer>
            <p>Universidad de Jaén - EPSL - Trabajo Fin de Grado -
Fernando Jose Lara Cardenas</p>
        </footer>
    </body>
</html>
```