Escola Universitaria Politécnica

UNIVERSIDADE DA CORUÑA

Grado en Ingeniería Electrónica Industrial y Automática

TRABAJO DE FIN DE GRADO

TFG Nº: **770G01A165**

TÍTULO: **DESARROLLO DE UN SISTEMA DE ADQUISICIÓN DE DATOS PARA PLANTA DE BOMBEO FOTOVOLTAICO.**

AUTOR: **MARTÍN NOVOA PELLO**

TUTOR: **MARÍA DEL CARMEN MEIZOSO LÓPEZ**
BENIGNO RODRÍGUEZ GÓMEZ

FECHA: **SEPTIEMBRE DE 2019**

Fdo.: EL AUTOR

Fdo.: EL TUTOR

AGRADECIMIENTOS

Quisiera mostrar mi agradecimiento a todas las personas que han hecho posible la realización de este proyecto:

A mis tutores, Mª Carmen y Benigno por la ayuda y la revisión del trabajo.

A Andrés Piñón, Esteban Jove y Manuel Rivas por la ayuda para la realización de toda la electrónica del proyecto.

A Cesar Añón por realizar el montaje del sistema en la planta de bombeo fotovoltaico.

A Carmen Regueiro por la ayuda a la hora de conseguir implementar y configurar el servidor.

TÍTULO: **DESARROLLO DE UN SISTEMA DE ADQUISICIÓN DE DATOS PARA PLANTA DE BOMBEO FOTOVOLTAICO.**

ÍNDICE

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

Fdo.: MARTÍN NOVOA PELLO

Listado de figuras

Listado de tablas

Listado de códigos de programación

TÍTULO: **DESARROLLO DE UN SISTEMA DE ADQUISICIÓN DE DATOS PARA PLANTA DE BOMBEO FOTOVOLTAICO.**

MEMORIA

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

Fdo.: MARTÍN NOVOA PELLO

Índice del documento MEMORIA

1 OBJETO

Esta propuesta consiste en desarrollar un sistema de adquisición de datos para la planta de bombeo fotovoltaico del Laboratorio de Renovables. Se trata de sensorizar algunas variables adicionales a las que ya se están midiendo tales como: presión a la salida de la bomba, caudal a la salida de la bomba, tensión y corriente a la entrada de la bomba, e incorporar todas las medidas a un sistema de adquisición de datos basado en Arduino, con posibilidad de comunicación remota, para poder registrarlos, procesarlos y mostrarlos a petición del usuario.

El registro de datos se puede llevar a cabo con un servidor local que permita crear una base de datos, o mediante una solución “Platform as a Service” alojada en la “nube”. Esta última tecnología se encuentra a día de hoy en fase de expansión en las empresas, por lo que se considera de especial interés para el TFG.

Además, si se desea monitorizar los datos de instalaciones de este tipo distribuidas en zonas más o menos extensas y sin acceso a red cableada de datos, es interesante disponer de un sistema de adquisición de bajo coste, con sistema de comunicación, y con posibilidad de análisis y consulta en tiempo real, que permita históricos de datos sin necesidad de invertir en hardware específico para este último fin.

En cualquier caso, se desarrollará un servidor de datos local sobre un PC, que salvaguarde el fin último de este trabajo.

2 ALCANCE

Se pretende diseñar e instalar físicamente el sistema de adquisición de datos. Las fases de desarrollo del trabajo podrían ser las siguientes:

1. Análisis y estudio de los diferentes sistemas Arduino
2. Análisis y estudio de los diferentes sensores posibles para esta aplicación.
3. Selección de los elementos hardware y adquisición de los mismos.
4. Montaje de circuitos y pruebas.
5. Programación de un servidor de datos local en PC.
6. Comparativa de distintas plataformas IoT (Internet of Things) para la adquisición, almacenamiento y visualización de datos recogidos por Arduino.
7. Si se encuentra alguna opción gratuita que se ajuste a los requerimientos del sistema, se proporcionará también esta solución mediante la programación de la plataforma y pruebas.

3 ANTECEDENTES

La EUP dispone de una planta de bombeo fotovoltaico directo, de dimensiones reducidas, instalada en Diciembre de 2017, con la que se pretende valorar el rendimiento de estos sistemas bajo diferentes condiciones de disponibilidad de recurso hídrico y radiación solar.

La planta consiste en una bomba, situada en un depósito de agua de 500 l, que se alimenta mediante un panel fotovoltaico a través de un regulador, lo que permite trasvasar el agua a otro depósito situado a cierta altura. Los depósitos están comunicados con una llave de paso de accionamiento manual que permite vaciar el superior sobre el que contiene la bomba. Además se dispone de otra llave de paso también manual que restringe la sección a la salida de la bomba para simular diferentes alturas manométricas.

Actualmente la planta no está automatizada ni en cuanto a operación ni a captura de datos, se están midiendo la tensión y corriente del panel fotovoltaico, así como la radiación solar que recibe el mismo, que se pueden ver en sendos displays, pero no se pueden registrar ni almacenar para hacer históricos o gráficos, esto limita en gran medida las posibilidades de experimentación y divulgación de la planta. El esquema inicial de la planta facilitado por la empresa instaladora Narontec se puede consultar en el plano 1, a continuación se describen las principales características de su equipamiento.

3.1. Panel fotovoltaico - MSM195-AS36

El panel fotovoltaico instalado en la planta es el MSM195-AS36 de la marca MünchenEnergieprodukte. Este panel cuenta con las siguientes características eléctricas:

Potencia de salida	195 W
Eficiencia	15.27 %
Voltaje de salida a Pmax	38.30 V
Corriente a Pmax	5.09 A
Tensión a circuito abierto	45 V
Corriente de cortocircuito	5.45 A

Tabla 3.1 – Especificaciones MSM195-AS36

Figura 3.1 – Panel fotovoltaico

3.2. Bomba de agua

La bomba de agua instalada en la planta es el modelo 9300 de la marca SHURflo. Esta bomba cuenta con las siguientes especificaciones técnicas:

Tipo de motor	Imanes permanentes
Voltaje	24 VDC nominales
Intensidad máxima	4.0 A
Altura máxima	70 m
Profundidad máxima	30 m
Caudal máximo	443 l/h

Tabla 3.2 – Especificaciones bomba SHURflo.

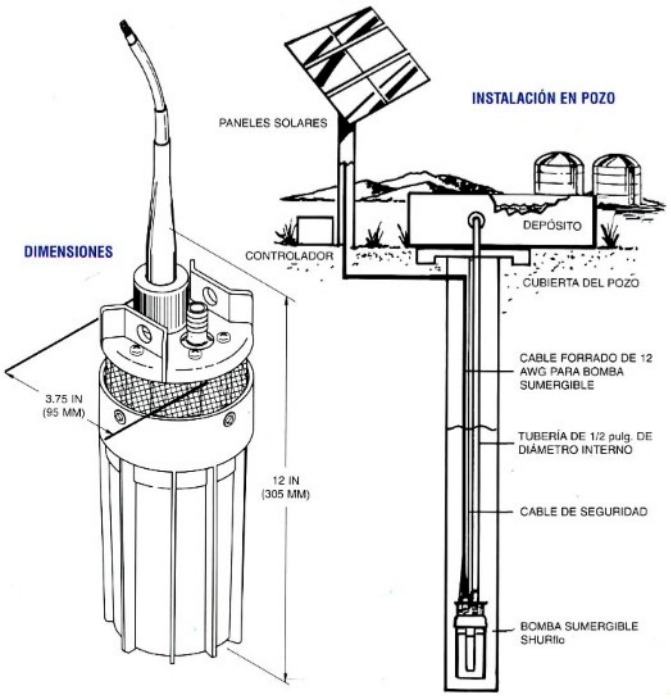


Figura 3.2 – Bomba de agua SHURflo

3.3. Regulador

El regulador instalado en la planta es el modelo 902-200(LCB-G) de la marca SHURflo. Este regulador mantiene el positivo de la placa y el de la bomba comunes y genera una diferencia de potencial entre el negativo de la placa y el de la bomba. De este modo, controlando la tensión y la corriente suministrada a la bomba, consigue optimizar la cantidad de agua bombeada. Además, cuenta con sensores de nivel que impiden que la bomba funcione cuando el nivel de agua es demasiado bajo. Cuando esto ocurre, el positivo y el negativo de la bomba tienen la misma tensión que el panel fotovoltaico, de manera que la diferencia de potencial es cero.

Sus características eléctricas son:

Tensión máxima de entrada	45 V
Tensión de encendido	25V
Tensión de apagado	28 V
Intensidad máxima de salida	7.0 A
Máximo consumo de potencia	25 mA
Potencia máxima de salida	150 W

Tabla 3.3 – Especificaciones regulador 902-200 (LCB-G)

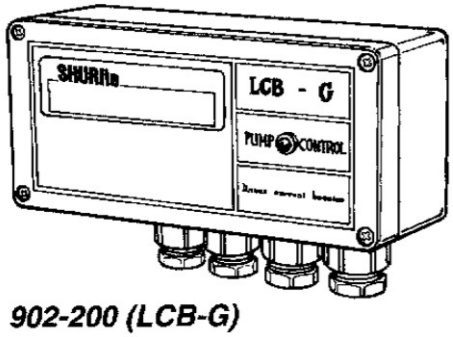


Figura 3.3 – Regulador LCB-G.

3.4. Analizador de red C.C. - AR3DC

Este analizador de red mide las siguientes magnitudes, en una red de corriente continua de baja tensión:

- Potencia activa (kW)
- Energia activa consumida (kWh)
- Energia activa generada (kWh)
- Amperio hora consumido (Ah)

- Amperio hora generado (Ah)

Es posible configurarlo para mostrar estas variables en su display LCD de 4 digitos con signo. Además, cuenta con una salida serie RS-485 con protocolo de comunicación Modbus RTU.

Este equipo se utiliza para mostrar la tensión y la corriente a la salida del panel fotovoltaico.

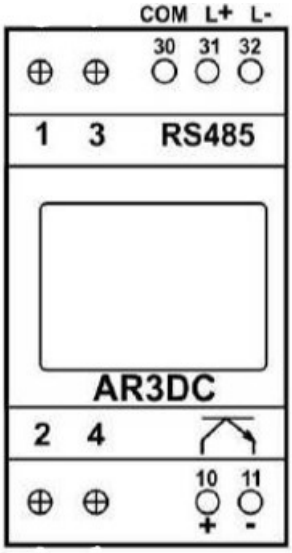


Figura 3.4 – Analizador de red

3.5. Resistencia serie shunt

La resistencia shunt instalada en la planta es una resistencia de 5A 60mV. Esta, se utiliza para medir la corriente de salida proporcionada por el panel fotovoltaico.

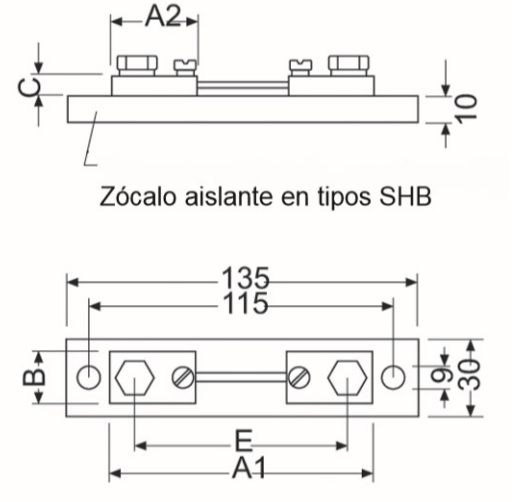


Figura 3.5 – Resistencia shunt cuadro

3.6. Célula calibrada

La célula calibrada, con la que se mide la radiación solar, instalada en la planta de bombeo no se corresponde con la que la empresa instaladora recoge en la documentación, sino que se ha instalado una célula cuyos rango de funcionamiento se desconocen.

3.7. Indicador digital - JUNIOR JR-E

Este es un indicador digital para señales de:

- TENSIÓN DC (±600V, ±200V y ±20V)
- CORRIENTE DC (±5A, ±1A, ±100mV y ±60mV)
- TENSIÓN AC (0-600V, 0-200V y 0-20V)
- CORRIENTE AC (0-5A, 0-1A, 0-100mV y 0-60mV)

Este indicador cuenta con un display de 4 dígitos de 14mm de altura y un rango máximo de lectura -9999 a 9999. En la planta se utiliza para indicar el valor de la radiación solar, si bien, se apreciaba que las medidas se salían del rango (indicación de overflow) en condiciones de sol despejado.

Figura 3.6 – Indicador JUNIOR JR-E

4 NORMAS Y REFERENCIAS

4.1. Disposiciones legales y normas aplicadas

Normativa establecida por la Escuela Universitaria Politécnica para la elaboración de de los Trabajos de Fin de Grado (TFG) en las titulaciones de Grado en Ingeniería Electrónica Industrial y Automática y Grado en Ingeniería Eléctrica.

4.2. Bibliografía

[1] SCHNEIDER ELECTRIC. TELEMECANIQUE; *Manual electrtecnico. Telesquemario* (1999).

4.3. Software utilizado

- Arduino IDE 1.8.9 - Arduino.
- Autocad 2020 - Autodesk.
- Cura 3.4.0 - Ultimaker.
- Excel 1902 - Microsoft.
- Kicad 5.1.0 - Kicad.
- MacSolar Software - MacSolar.
- Matlab R2017a - MathWorks.
- Nx 11.0 - Siemens.
- OrCad Capture 17.2 - Cadence.
- pgAdmin4 4.9 - pgAdmin.
- ThingsBoard - ThingsBoard.

4.4. Otras referencias

[1] *Configure Cloudant NoSQL DB as Historian Data Storage for IBM Watson IoT*. IBM. 1911. [Consulta 25 de junio del 2019]. Disponible en: https://developer.ibm.com/recipes/tutorials/cloudant-nosql-db-as-historian-data-storage-for-ibm-watson-iot-parti/

[2] *Guía de aprendizaje de iniciación*. IBM. 1911. [Consulta 25 de junio del 2019]. Disponible en: https://cloud.ibm.com/docs/services/IoT?topic=iot-platform-getting-started

[3] *Connect a Device*. Thinger.io. 2015. [Consulta 24 de mayo del 2019]. Disponible en: http://docs.thinger.io/arduino/

[4] *Temperature upload over MQTT using Arduino UNO, ESP8266 and DHT22 sensor*. ThingsBoard, Inc. 2016. [Consulta 3 de junio del 2019]. Disponible en: https://thingsboard.io/docs/samples/arduino/temperature/

[5] *Installing ThingsBoard on Windows*. ThingsBoard, Inc. 2016. [Consulta 14 de julio del 2019]. Disponible en: https://thingsboard.io/docs/user-guide/install/windows/

[6] *ThingSpeak*. The MathWorks, Inc. 1994. [Consulta 7 de mayo del 2019]. Disponible en: https://es.mathworks.com/help/thingspeak/

5 DEFINICIONES Y ABREVIATURAS

- ADC: Convertidor Analógico Digital.
- Amplificador operacional rail to rail: amplificador operacional cuya salida satura muy próxima a la tensión de alimentación, del orden de mV.
- Caudalímetro: instrumento de medida del caudal de un fluido.
- Convertidor AC-DC: Convertidor de corriente alterna a corriente continua.
- Convertidor DC-DC: Convertidor de corriente continua a corriente continua, de diferente valor.
- E/S: Entradas/Salidas.
- FS: Fondo de escala. Es la diferencia entre el límite superior y el límite inferior del campo de medida. Para el ADC del Arduino MKR 1010, como el limite inferior es 0 V, el fondo de escala coincide con el límite superior.
- HTTP: protocolo de comunicación que permite la transferencia de información en la World Wide Web.
- LCD: *Liquid Cristal Display* (Monitor de Cristal Liquido).
- LSB: Resolución. Es el incremento mínimo de la variable de entrada que produce una variación medible en la salida.
- Resistencia shunt: resistencia de valor conocido que se emplea para conocer la intensidad de corriente eléctrica, a través de la caída de tensión que produce en ella.

- Manómetro: instrumento de medida de la presión de fluidos.
- MQTT: protocolo de comunicación basado en el modelo publicación-suscripción.
- Sensor de efecto Hall: sensor que, mediante la medición del campo magnético, permite medir magnitudes secundarias como velocidad o caudal.
- Swing from rail: es la diferencia entre la tensión de alimentación y la tensión a la que satura el operacional.
- Vdc: Voltaje en corriente continua.
- WiFi: tecnología que permite la conexión inalámbrica entre dispositivos electrónicos.

6 REQUISITOS DE DISEÑO

Este proyecto se plantea como un sistema de adquisición de datos de bajo coste basado en Arduino, que permita la comunicación inalámbrica con un servidor. Este, deberá almacenar los datos obtenidos de la planta y permitir al usuario exportarlos y visualizarlos a voluntad.

6.1. Magnitudes de la planta.

Para la medida de las diferentes magnitudes de la planta será necesario realizar el acondicionamiento de las diferentes señales para así, ajustarlas al fondo de escala del ADC que incorpore el Arduino. Las magnitudes a medir son las siguientes:

- Tensión del panel fotovoltaico.
- Corriente del panel fotovoltaico.
- Tensión de la bomba de agua.
- Corriente de la bomba de agua.
- Presión a la salida de la bomba de agua.
- Caudal de la bomba de agua.
- Radiación solar.

6.1.1. Panel fotovoltaico

Como se puede ver en la Tabla 3.1 el panel fotovoltaico tiene una tensión máxima de 45 V, en circuito abierto, y una corriente máxima de 5.45 A, en cortocircuito. Sin embargo, la intensidad únicamente superará los 5.09 A en situaciones anómalas.

6.1.2. Bomba de agua

En la Tabla 3.2, vemos que tiene una tensión nominal de 24 Vdc y una intensidad máxima de 4 A; el caudal máximo es de 443 l/min y la altura máxima es de 70m, que según la siguiente ecuación:

$$p = p_0 + \rho g h \tag{6.1}$$

Y sabiendo que:

$$p_0 = p_{atm} = 1{,}013 * 10^5 Pa$$

$$\rho = 1000 kg/m^3$$

$$g = 9,8 m/s^2$$

$$h = h_{bombamax} = 70m$$

Se obtiene que la presión máxima a la salida de la bomba será:

$$p = 1{,}013 * 10^5 + 1000 * 9{,}8 * 70 = 0{,}788 MPa \tag{6.2}$$

Por otra parte, para la medida de la presión y el caudal hay instalados un manómetro de glicerina y un caudalímetro analógico, respectivamente, por lo que habrá que instalar sensores que puedan ser conectados al Arduino.

6.1.3. Radiación

La radiación solar se mide mediante una célula calibrada; siendo la Artesa 1003001 la que aparece en la documentación de la planta. Sin embargo, la instalada no se corresponde con esta célula; teniendo unos rangos de funcionamiento diferentes, llegando a superar el fondo de escala del indicador JUNIOR JR-E. Por lo tanto, será necesario realizar las medidas pertinentes para poder medir la radiación solar con esta célula, de la cual no se tiene información.

6.1.4. Tensiones en modo común.

Como se puede ver en el Plano 1 la resistencia shunt, que aparece representada como un amperímetro dentro del cuatro de protección, está ubicada a la salida de la placa fotovoltaica. Debido a esto y en caso de que la bomba no tuviese suficiente agua y por lo tanto, estuviese apagada; la caída de tensión en la resistencia shunt sería aproximadamente 0V y la tensión a la salida de la placa fotovoltaica podría ser de hasta 45V. Por lo tanto, segun la siguiente ecuación:

$$V_{CM} = \frac{V^+ + V^-}{2} \tag{6.3}$$

Y sabiendo que:

$$V^+ = 45V$$

$$V^- = 45V$$

Se tiene que :

$$V_{CM} = 45V \tag{6.4}$$

Por lo tanto, es necesario que los componentes seleccionados para la realización del acondicionamiento de las señales soporte tal rango de tensiones en modo común.

6.1.5. Alimentación de los componentes

Será necesario seleccionar el método de alimentación del Arduino y los circuitos de acondicionamiento, valorando la alimentación mediante un convertidor AC-DC conectado a la red eléctrica, la cual se encuentra próxima a la planta, o la utilización de un convertidor DC-DC para así, utilizar la propia placa fotovoltaica como alimentación.

7 ANÁLISIS DE LAS SOLUCIONES

7.1. Selección de los sensores

Debido a que los sensores incorporados en la planta son sensores analógicos, será necesaria la adquisición de nuevos sensores de presión y caudal que puedan ser leídos por el Arduino.

7.1.1. Selección del sensor de caudal

Para la medición del caudal de agua impulsado por la bomba, se buscará un sensor con el menor coste posible. Buscando en internet caudalímetros con rosca de media pulgada, se pueden encontrar diferentes modelos que cuentan con características muy similares; como pueden ser los presentes en la Figura 7.1

(a) Latón (b) Plástico

Figura 7.1 – Caudalímetros

Estos sensores contienen un sensor de efecto Hall en su interior; generando así una señal cuadrada, de amplitud igual a la tensión de alimentación, cuya frecuencia es directamente proporcional al caudal del fluido.

En este caso, debido a que las características son similares y cumplen con las necesidades mínimas, nos centraremos en el precio como criterio de selección, escogiendo así, el caudalímetro de latón, de la marca Zeast (Figura 7.1(a)). Sus características técnicas son las siguientes:

Rengo del flujo	1 - 30 l/min
Presión máxima	1.75 MPa
Voltaje de funcionamiento	4.5 - 18 V
Frecuencia	$6 * Q - 8$

Tabla 7.1 – Características sensor de flujo

7.1.2. Selección del sensor de presión

De manera análoga al sensor de caudal, se busca un sensor de presión con el menor coste posible. En este caso, el único sensor de presión a un precio asequible es el siguiente:

Figura 7.2 – Sensor de presión

Este sensor, de la marca Walfront, tiene las siguientes características técnicas:

Rengo de presión	0 - 1.2 MPa
Presión máxima	2.4 MPa
Voltaje de funcionamiento	5 V
Voltaje de salida	0.5 - 4.5 V
Error de la medición	1.5 % FS

Tabla 7.2 – Características sensor de presión.

7.2. Selección del Arduino

Teniendo en cuenta lo visto anteriormente, es necesario realizar siete medidas de las cuales, seis son medidas analógicas y una es una medida digital. Por lo tanto, necesitamos un Arduino con al menos 6 entradas analógicas y una entrada digital.

Arduino ofrece una amplia gama de dispositivos con conectividad inalámbrica, dentro de su apartado IoT. Estos dispositivos cuentan con protocolos de comunicación como WiFi, GSM, SigFox, LoRaWAN o Narrow Band. En la tabla 7.3 se pueden observar las características más importantes, de los dispositivos más interesantes de la gama IoT para la realización de este proyecto.

Producto	Conectividad	Entradas analógicas	E/S digitales	Precio
Arduino MKR WiFi 1010	WiFi 2.4 GHz	7 (8/10/12 bits)	8	27.9 €
Arduino MKR 1000	WiFi 2.4 GHz	7 (8/10/12 bits)	8	30.99 €
Arduino MKR FOX 1200	SigFox	7 (8/10/12 bits)	8	35 €
Arduino MKR WAN 1300	LoRaWAN	7 (8/10/12 bits)	8	35 €
Arduino MKR GSM 1400	GSM	7 (8/10/12 bits)	8	59.9 €
Arduino MKR NB 1500	Narrow Band	7 (8/10/12 bits)	8	66.9€
Arduino UNO WiFi	WiFi	6 (10 bits)	14	38.9 €

Tabla 7.3 – Comparativa Arduino

Como se puede apreciar en la tabla, todos los dispositivos cumplen las necesidades del proyecto. Como en el laboratorio se dispone de conexión WiFi, se selecciona el Arduino MKR 1010 al ser el más barato.

7.3. Método de alimentación

Como se ha comentado en el Apartado 6.1.5 disponemos de dos opciones para la alimentación de los circuitos de acondicionamiento y el Arduino:

- Convertidor AC-DC: la utilización de la red eléctrica para la alimentación del sistema de adquisición supondría que se estuviesen adquiriendo datos las 24 horas del día. Debido a que la única información relevante se produciría de día, sería necesario un filtrado de los datos obtenidos o el aumento de la complejidad de la programación al tener que controlar el momento en el que se produce el amanecer y el anochecer.

- Convertidor DC-DC: la utilización de la placa fotovoltaica como alimentación evitaría el problema de controlar el amanecer y el anochecer, ya que el Arduino se encendería en el momento en el que la placa produzca suficiente energía para ello. Además una planta de bombeo fotovoltaico es especialmente interesante en zonas en las que no se dispone de red eléctrica.

Por lo tanto, la opción más interesante es la utilización de un convertidor DC-DC.

Se busca un convertidor que permita una tensión de entrada de mas de 45 V y que sea capaz de proporcionar suficiente intensidad para alimentar al Arduino y los circuitos de acondicionamiento de las señales. De entre las diferentes posibilidades, se selecciona el convertidor buck de la Figura 7.4, debido al bajo coste del mismo y sus buenas características eléctricas. Este, está basado en el circuito integrado LM2576HV, que tiene las siguientes especificaciones eléctricas:

Rango voltaje de entrada	5 - 60 V
Rango voltaje de salida	1.25 - 26 V
Corriente máx salida	3 A
Potencia máx salida	20 W

Tabla 7.4 – Caracaterísticas del convertidor DC-DC

Además, en la Figura 7.3 se puede ver que su eficiencia es relativamente alta, teniendo en la mayor parte del rango de tensiones de entrada una eficiencia superior al 75 %, llegando incluso al 80 %.

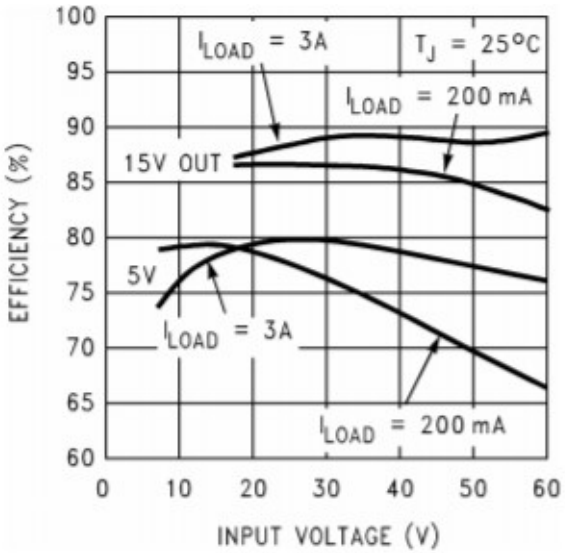


Figura 7.3 – Eficiencia convertidor DC-DC

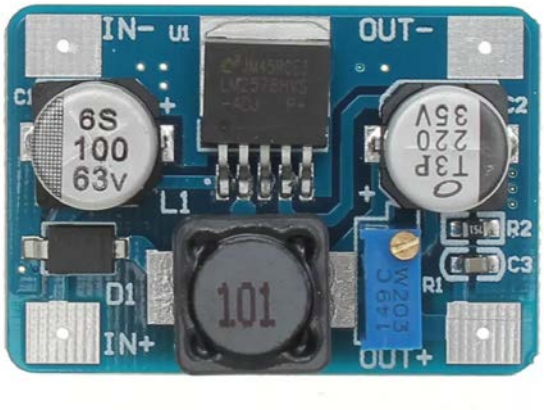


Figura 7.4 – Convertidor DC-DC

7.4. Circuitos de acondicionamiento

Para la lectura de las variables analógicas se utilizará el ADC incorporado en el microcontrolador del Arduino MKR 1010, el cual tiene un fondo de escala de 3.3 V.

Por último, un factor a tener en cuenta a la hora de realizar mediciones por medio de un ADC es la resistencia máxima de la fuente de la señal. En el apartado de Características eléctricas del ADC (página 1001 del datasheet del microcontrolador) se puede ver que su valor máximo es de 3.5 kΩ. Por lo tanto, será necesario incluir, en ciertas ocasiones, seguidores de tensión para ajustar las impedancias.

Debido a que se utiliza un convertidor DC-DC para la alimentación de los circuitos, únicamente disponemos de alimentación unipolar a 5V. Por lo tanto, es necesario escoger un amplificador *rail to rail* para reducir el error debido al *swing from rail*.

7.4.1. Acondicionamiento de la tensión de la placa y de la bomba

Para el acondicionamiento de las tensiones en la placa fotovoltaica se opta por realizar un divisor de tensión, con el fin de convertir la tensión del panel a un valor de tensión apto para el Arduino. Como se puede apreciar en la Tabla 3.1 la tensión de salida del panel fotovoltaico va de 0 a 45 V, que deberá convertirse a un rango de 0 a 3.3 V.

Sin embargo, para la medición de tensión en la bomba de agua, es necesario conocer la tensión diferencial entre los bornes positivo y negativo. Para ello, se pueden optar por dos métodos:

- Medir la tensión diferencial entre el positivo y el negativo de la bomba, lo que supondría una tensión en modo común elevada.
- Medir la tensión en el negativo de la bomba que, conociendo la tensión en la placa fotovoltaica, permitiría obtener la tensión entre los bornes de la bomba.

Debido a que las tensiones en modo común elevadas pueden suponer dificultades para el diseño del circuito de acondicionamiento, se opta por la medición de la tensión en el borne negativo de la bomba.

A pesar de que la tensión nominal de la bomba es de 24 Vdc, cabe la posibilidad de que en el negativo haya hasta 45 V, cuando el nivel de agua no sea suficiente para que la bomba comience a funcionar.

Por lo tanto, ambos circuitos de acondicionamiento son iguales, un divisor de tensión y un seguidor para evitar errores en la medida. Para la realización del seguidor de tensión se selecciona el OPA378, un amplificador de precisión con un *swing from rail* máximo de 8 mV para una resistencia de carga de $10k\Omega$.

7.4.2. Acondicionamiento de la corriente de la placa y en la bomba

La medición de la corriente en la placa se realiza mediante una resistencia *shunt* de 5A 60 mV; que según la Ley de Ohm, aplicada en la Ecuación 11.3, se tiene que el valor de esta resistencia es de $12m\Omega$.

Sin embargo, para la medición de la corriente en la bomba es necesario incorporar una nueva resistencia *shunt*. Como se expone en la Tabla 3.2 la corriente máxima de la bomba es

de 4 A. Por lo tanto, se opta por una resistencia de 5 A 75 mV; que según la Ley de Ohm, aplicada en la Ecuación 11.6, el valor de esta resistencia es de $15m\Omega$.

Como se expone en el Apartado 6.1.4 la tensión en modo común al realizar la medición de la corriente en la placa es elevada. Por ello, se utiliza un INA169, un medidor de corriente que soporta tensiones en modo común de entre 2.7 y 60 V, independientemente de la tensión de alimentación.

Para medir la corriente de la bomba se opta por el uso del mismo integrado. Debido a que este debe tener en su entrada una tensión en modo común de como mínimo 2.7 V, se colocará la resistencia *shunt* antes del borne positivo de la bomba, donde la tensión es igual a la tensión en la placa fotovoltaica.

Como se propone en el apartado 8.2.1 ”Buffering output to drive an ADC” de la hoja de características, se incorpora un seguidor de tensión entre el INA169 y el Arduino. De este modo, la resistencia de entrada del ADC no afecta a la ganancia del INA169.

7.4.3. Acondicionamiento de la presión a la salida de la bomba

Como se puede ver en la Tabla 7.2 el sensor de presión tiene una salida analógica de 0.5 a 4.5 V para un rango de presiones de 0 a 1.2 MPa. Sin embargo, el rango de presiones de funcionamiento de la bomba no llega a 1.2 MPa, sino a los 0.788 MPa que se obtuvieron en la Ecuación 6.2.

Según los cálculos realizados en el Apartado 11.4.1 el rango de tensión que proporcionará el sensor de presión es de 0.5 a 3 V. Por lo tanto, es necesario convertir la curva de calibración roja de la Figura 7.5 en la curva de calibración azul de la misma figura.

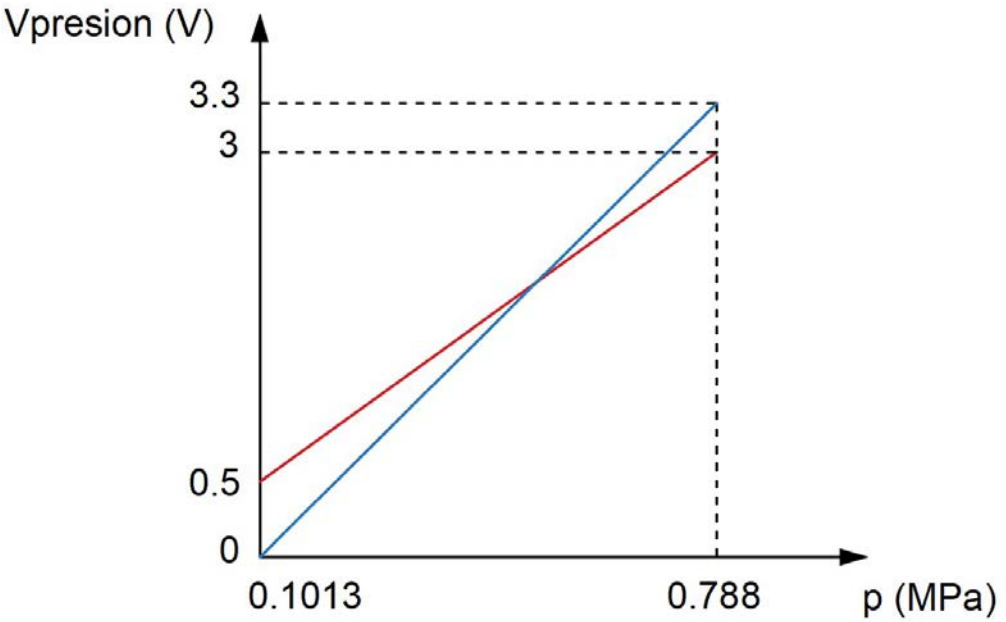


Figura 7.5 – Curvas de calibración del sensor de presión

Para ello, se utilizará un amplificador operacional en modo diferencial, como el que se puede ver en la siguiente imagen:

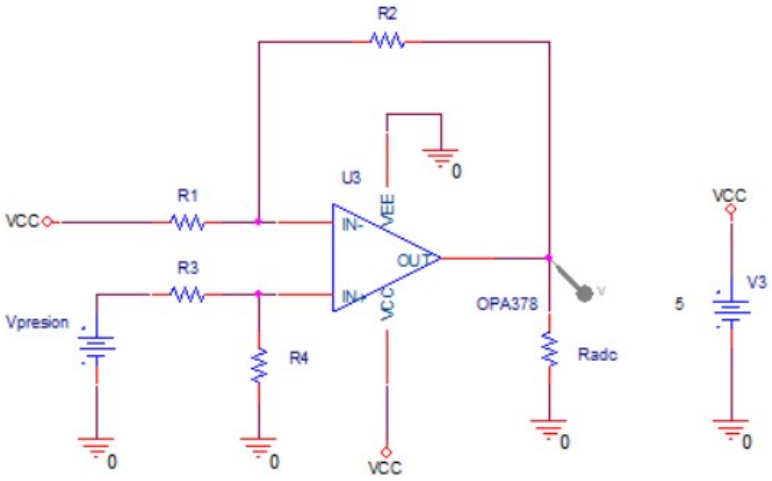


Figura 7.6 – Amplificador diferencial

7.4.4. Acondicionamiento del caudal a la salida de la bomba

Como se explica en el Apardado 7.1.1, el sensor de caudal genera una señal cuadrada cuya frecuencia es proporcional al caudal de agua y con amplitud igual a la tensión de alimentación del sensor. Como en este caso la tensión de alimentación del sensor es de 5 V, es necesario reducir la amplitud de los pulsos. Para ello es necesario tener en cuenta los rangos de tensión de los niveles lógicos del microcontrolador:

- Nivel alto: $V_{IN} \geq 0{,}7 * VDD = 2{,}3V$
- Nivel bajo: $V_{IN} \leq 0{,}3 * VDD = 0{,}99V$

Teniendo en cuenta esto, se reducirá la tensión a un valor en torno a los 3 V. Para ello se realiza un divisor de tensión con un seguidor. Teniendo en cuenta la Ecuación 11.1 y sabiendo que $V_{OUT} = 3V$ y $V_{IN} = 5V$ tenenmos que:

$$R_1 = 0{,}667 * R_2 \tag{7.1}$$

Teniendo en cuenta los valores normalizados de las resistencias con tolerancia E24, se toman los valores $R_1 = 10k\Omega$ y $R_2 = 15k\Omega$.

De este modo:

$$V_{OUT} = 5 * \frac{15 * 10^3}{10 * 10^3 + 15 * 10^3} = 3V \tag{7.2}$$

Como la frecuencia máxima de la señal se encuentran en torno a los 80 Hz, se utilizará en el seguidor de tensión el OPA378, que tiene un Slew Rate es de $0{,}4V/\mu s$, por lo que no va a haber deformaciones en la señal cuadrada.

7.4.5. Acondicionamiento de la Radiación solar

Como se expone en el el Apartado 6.1.3 la medida de la radiación no se realiza con la célula calibrada que aparece en la documentación de la planta. Por lo tanto, se desconoce cual es el valor máximo de la señal de salida de la célula y su relación con la radiación solar, siendo necesario establecer un valor máximo aproximado para el diseño del circuito de acondicionamiento.

En los momentos en los que el medidor JUNIOR JR-E marcaba que la tensión estaba fuera de rango, se llegaron a medir tensiones de 127 mV. Por ello, se establece el valor máximo de la salida de la célula calibrada en 150 mV.

Para el acondicionamiento de la señal se emplea el AD8237, un amplificador de instrumentación *rail to rail*. Este amplificador tiene la siguiente configuración:

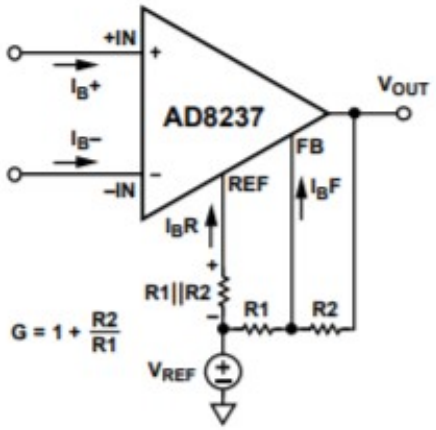


Figura 7.7 – AD8237

Además, debido a que se mide una tensión diferencial que no esta referenciada a masa, es necesario incorporar una resistencia que permita el retorno de las corrientes de bias.

Teniendo en cuenta los cálculos realizados en el Apartado 11.5 se seleccionan las resistencias normalizadas, con tolerancia E24. Estas resistencias son: $R_1 = 1k\Omega$ y $R_2 = 22\Omega$. De este modo se tiene que:

$$V_{IN\_MAX} = 3{,}3 * (1 + \frac{22k}{1k}) = 143{,}48mV \tag{7.3}$$

Debido a que el valor de 150 mV fue establecido de manera aproximada, el valor obtenido debido al uso de resistencias normalizadas es igualmente correcto.

De este modo tenemos el siguiente circuito de acondicionamiento:

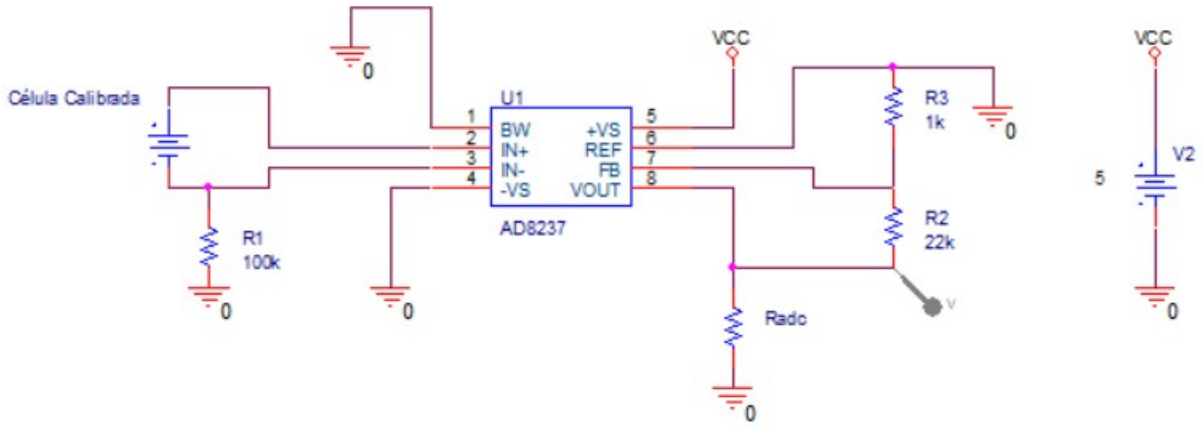


Figura 7.8 – Circuito de acondicionamiento de la radiación solar

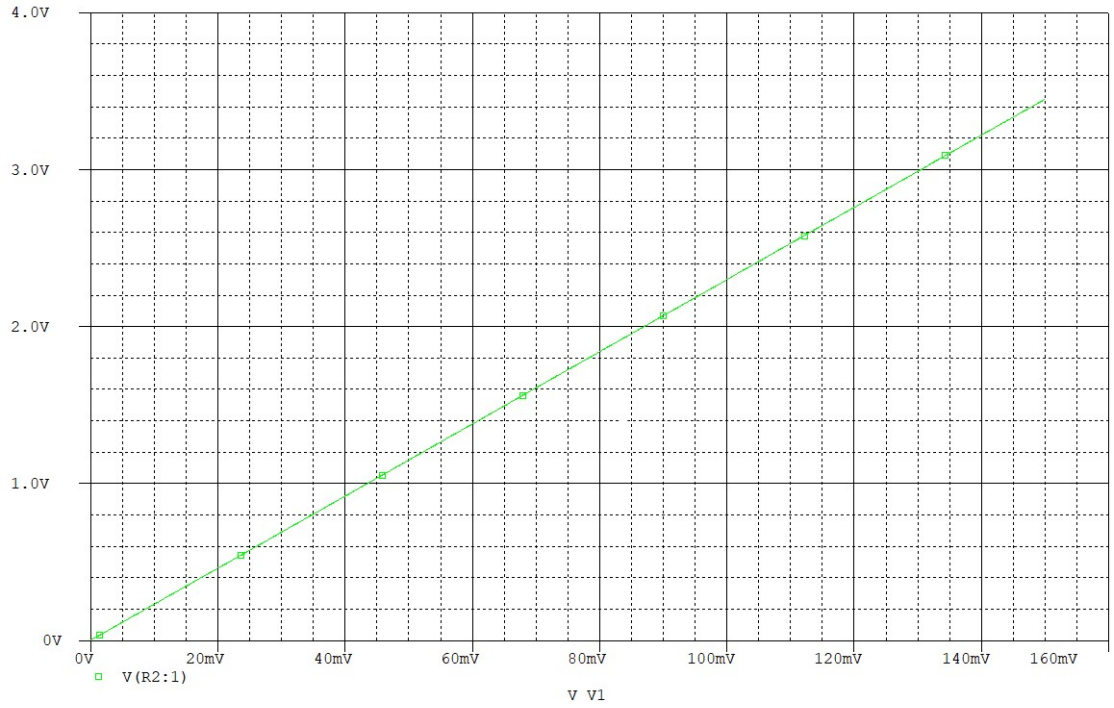


Figura 7.9 – Simulación del circuito de acondicionamiento de la radiación

Ahora, se intentará obtener la relación entre la tensión de salida de la célula calibrarda y la radiación solar. Para ello, se emplea el medidor de radiación solar SLM018C-E de la marca Mac Solar.

Figura 7.10 – Medidor de radiación solar SLM018C-E

Este dispositivo se coloca en la propia estructura del panel fotovoltaico, intentando que ambos tengan la misma inclinación:

Figura 7.11 – Colocación del radiómetro

Para la realización de las medidas se configura el dispositivo en el modo ”sto”, en el cual toma medidas cada 10 segundos. Pasados 6 minutos, realiza la media de los valores obtenidos en el último periodo de tiempo y la almacena en su memoria interna.

Al mismo tiempo se programa el Arduino para realizar el mismo procedimiento de toma de datos. Sin embargo, el almacenamiento de los datos se realiza en una hoja de cálculo de

Google Drive, de manera análoga a como se explica en el Apartado 13.1.

7.5. Plataformas IoT

En este apartado se analizan las plataformas IoT más interesantes para la implementación de este proyecto, exponiendo sus ventajas e inconvenientes. En el Anexo 13 se puede ver una guía de configuración y uso de las diferentes plataformas.

7.5.1. Google Drive

Google Drive es un servicio de alojamiento en la nube que cuenta con herramientas para la realización de tareas como la edición de documentos de texto u hojas de cálculo, entre otras.

Una de las funcionalidades más interesantes para su uso en soluciones IoT son los formularios, que permiten realizar encuestas con respuestas de selección múltiple o entrada de texto. Este último tipo, permite la respuesta mediante una solicitud HTTP, lo que es muy útil para ser empleada con un Arduino. Además, los resultados de las preguntas se pueden almacenar en una hoja de cálculo.

- Ventajas:
  - Gratuita.
  - Configuración sencilla.
- Inconvenientes:
  - La representación de los datos no permite introducir el intervalo de tiempo a graficar.

7.5.2. Google Cloud Platform

Google Cloud Platform es una plataforma en la nube que contiene herramientas enfocadas a áreas como: almacenamiento, computación en la nube, inteligencia artificial o Big Data. Este último tiene especial interés para este proyecto, ya que contiene herramientas especializadas en IoT.

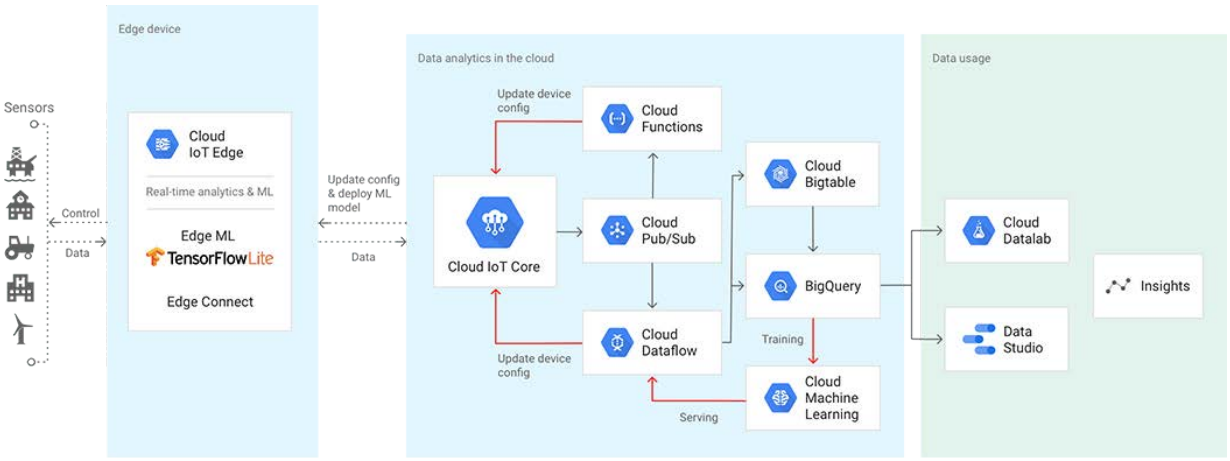


Figura 7.12 – Diagrama de Google Cloud

En la Figura 7.12 se pueden ver los posibles flujos de información en la implementación del internet de las cosas.

- Ventajas:
  - Cuenta con criptografía de curva elíptica, permitiendo la utilización del chip de encriptación ATECC508A integrado en el Arduino MKR 1010 .
  - Es una plataforma con gran seguridad.
- Inconvenientes:
  - Es de pago.
  - Difícil de configurar.
  - Para representar los datos es necesario exportarlos a una plataforma externa.

7.5.3. Thingspeak

Thingspeak es una plataforma IoT de código abierto. Esta plataforma permite el almacenamiento y representación de la información enviada mediante solicitudes HTTP. Además, permite la introducción de código de MATLAB para el análisis de la información y su representación.

En cuanto a la cantidad de información que permite enviar y almacenar, Thingspeak permite enviar 3 millones de mensajes al año, por lo que, la plataforma, recomienda que se envíe un mensaje cada 22 segundos como máximo. Se pueden configurar hasta 4 dispositivos con hasta 8 variables cada uno.

- Ventajas:
  - Es gratuita.
  - Configuración de la plataforma sencilla.

    - Permite el uso de código de MATLAB.

- Inconvenientes:

    - En la representación no se puede graficar un intervalo de tiempo, sino que únicamente se puede seleccionar el número de días y el número de puntos a graficar.
    - Cuando se introduce código de MATLAB para graficar datos de un periodo de tiempo anterior, no se puede seleccionar los datos en función de su fecha. Es necesario conocer cuantos mensajes se han enviado desde ese momento.

7.5.4. Thinger.io

Thinger.io es una plataforma IoT de código abierto que permite el almacenamiento y representación de los datos enviados por un dispositivo mediante solicitudes HTTP. Sin embargo, en la versión web, la cantidad de información que se puede enviar es reducida, únicamente 2.7 kB al mes.

Una de las grandes ventajas de esta plataforma es que se puede descargar e instalar, de manera gratuita, en una máquina con sistema operativo Ubuntu o Raspbian.

- Ventajas:

    - Es gratuita.
    - Configuración sencilla.
    - Se puede instalar en una máquina propia para crear un servidor.

- Inconvenientes:

    - No se puede seleccionar el intervalo de tiempo a representar.
    - La cantidad de información que se puede mandar, cada mes, es reducida.

7.5.5. ThingsBoard

ThingsBoard es una plataforma IoT de código abierto que proporciona un sistema escalable para sistemas de diferentes tipos. Ofrecen cuatro servicios diferentes en la nube:

- ThingsBoard Community Edition: versión de código abierto diseñada para ser instalada en una máquina del usuario o en un servicio en la nube. Actualmente, únicamente está disponible DigitalOcean, pero pronto se podrá instalar en las plataformas Azure, Google Cloud Platform, IBM Cloud, AWS y Alibaba Cloud. Además, cuenta con una versión online, que cuenta con la desventaja de que no hay ninguna base de datos en la que se almacenen los datos y, por lo tanto, no se pueden exportar.

- ThingsBoard Professional Edition: versión de pago de la plataforma que cuenta con diferentes ventajas como una mayor compatibilidad con sistemas de comunicación como

SigFox o LoRaWAN y una mayor variedad de plataformas en la nube para su implementación ya que, en este momento, es posible emplear AWS y Azure. Además, cuenta con soporte técnico proporcionado por la compañía.

- ThingsBoard IoT Gateway: solución de código abierto que permite la conexión de la plataforma ThingsBoard con sistemas de terceros que cuenten con esta plataforma.
- ThingsBoard License Server: servicio que permite a los usuarios de Thingsboard Professional Edition crear licencias online para sus clientes.

En cuanto a sus ventajas e inconvenientes:

- Ventajas
  - Cuenta con una versión gratuita, tanto para instalar en una máquina como en la versión online.
  - Tiene varias posibilidades de uso, de modo que se puede adecuar fácilmente a numerosos proyectos.
  - Todas sus versiones permiten la creación de interfaces muy profesionales.
  - En los gráficos es posible seleccionar el intervalo de tiempo a representar.
- Inconvenientes
  - La versión de prueba online no permite el exportado de los datos.

7.5.6. Ubidots

Ubidots es un plataforma diseñada para la creación de aplicaciones enfocadas a la integración de sistemas IoT, mediante herramientas de recolección de datos, análisis y representación.

En cuanto al medio de conexión, esta plataforma permite la conexión con dispositivos con conectividad WiFi, SigFox LoRaWAN, Narrow Band, LTE-M y Cellular, mediante los protocolos HTTP y MQTT. Se ofrecen cuatro versiones principales, ordenadas de menor a mayor en función de sus capacidades y escalabilidad:

- IoT Entrepreneur
- Professional
- Industrial
- Scale

Además, existe una versión de educación, gratuita, denominada “Ubidots for Education”. En ella se pueden configurar hasta 3 dispositivos y una interfaz en la que representar las variables.

- Ventajas

- Tiene una versión gratuita.
- Tiene diferentes versiones para ajustarse a diferentes proyectos

- Inconvenientes
  - En la representación de los datos, únicamente se puede seleccionar el número de puntos a representar y no el intervalo de tiempo .
  - En la versión de prueba no es posible exportar los datos.

7.5.7. IBM Cloud

IBM Cloud es una plataforma que ofrece numerosos servicios en la nube como: bases de datos, análisis, cálculo en la nube, inteligencia artificial, seguridad, internet de las cosas, o gestión de redes.

A pesar de que es una plataforma de pago, cuenta con numerosos servicios *lite* que son gratuitos. Dentro de estos servicios se encuentran “Internet of Things Platform”, que permite la comunicación de la plataforma con dispositivos IoT como Arduino, y “Cloudant”, una base de datos NoSQL en la que se almacena la información enviada por el dispositivo IoT. Se pueden tener hasta 500 dispositivos conectados y transferir hasta 200 MB al mes.

- Ventajas
  - Cuenta con servicios gratuitos.
  - En “Internet of Things Platform” se pueden crear gráficos en los que se puede seleccionar el intervalo de tiempo a representar.

- Inconvenientes
  - La configuración de la base de datos es compleja.
  - En “Cloudant” los datos únicamente se pueden exportar en formato .json, por lo que se complica la utilización de los datos.
  - Es necesario mantener un uso constante de la plataforma, ya que tras 30 días sin uso se bloquean los servicios.

7.5.8. Cayenne myDevices

Esta es una plataforma que permite monitorizar variables medidas por dispositivos IoT, como Arduino o Raspberry Pi. Además permite la creación de alarmas en función del valor que tomen determinadas variables.

- Ventajas
  - Es gratuita.
  - La configuración de la plataforma es sencilla.
  - Permite seleccionar el intervalo de tiempo a representar.

  - Se pueden exportar los datos en formato .csv.

- Inconvenientes
  - Actualmente, la librería de Arduino no tiene soporte para el MKR 1010, por lo que es necesario crearla.
  - Las interfaces gráficas que se pueden crear son muy simples.

7.5.9. Conclusiones

Una vez estudiadas las plataformas IoT expuestas a lo largo de este apartado, se puede concluir que gran parte de las plataformas IoT disponibles en la nube, de manera gratuita, no cumplen con los requisitos básicos de este proyecto, bien porque no se puedan exportar los datos subidos a la plataforma o porque no se puedan representar a voluntad y de manera cómoda para el usuario. Por lo tanto, se opta por emplear un servidor local.

Para la implementación del servidor local se utilizará un servidor de código abierto. De las plataformas estudiadas durante este apartado, únicamente están disponibles, para la instalación en una máquina propia, Thinger.io y ThingsBoard, cuyas guías de uso se se recogen los apartados 13.4 y 13.5, respectivamente.

De entre estas dos plataformas destaca especialmente ThingsBoard. Esta plataforma, instalada como servidor local, permite la incorporación de diferentes tipos usuarios:

- Administrador del sistema: Únicamente hay un perfil de este tipo. Es el encargado de crear las “Organizaciones”, que se corresponderían con las diferentes plantas configuradas en la plataforma, y configurar los perfiles de administrador de las mismas.
- Administrador de la organización: Estos cuentan con permisos para configurar los dispositivos conectados al servidor y las interfaces gráficas.
- Cliente: Pueden ver los datos enviados por los dispositivos y la representación de los mismos. Sin embargo, no disponen de permisos para realizar modificaciones.

Debido a que este proyecto va a ser empleado académicamente, esto es una característica especialmente interesante, ya que los profesores serían administradores de la organización, pudiendo configurarla a voluntad, y los alumnos tendrían cuentas de cliente, a los que únicamente se les permite ver la información.

7.6. Programación del Arduino

A la hora de programar el Arduino para funcionar como un sistema de adquisición de datos es necesario tener en cuenta los siguientes factores:

- Tipo de señales que van a ser muestreadas.
- Resolución del ADC, en caso de que permita su configuración.
- Frecuencia de muestreo y método para la obtención de la misma.

7.6.1. Medición de las magnitudes

Como se especifica en el Apartado 6.1, hay dos tipos de señales: señales de corriente continua, en las que hay que medir el valor de la tensión, y una señal cuadrada, en la que será necesario medir su frecuencia.

La medición de las magnitudes de C.C. se realiza mediante el ADC incorporado en el Arduino MKR 1010. Este, puede ser configurado para funcionar con una resolución de 8, 10 o 12 bits. Debido a que la frecuencia de muestreo va a ser muy baja, se configurará el ADC para funcionar con 12 bits ya que, en este caso, la velocidad de conversión no es un factor relevante.

En cuanto a la medición de la frecuencia de la señal cuadrada, esta se puede realizar de varios modos: empleando la función “pulseIN”, cuyo tiempo de ejecución es inversamente proporcional a la frecuencia de la señal, y empleando un contador que cuente el número pulsos en un intervalo de tiempo determinado de manera que, conociendo el intervalo de tiempo, se pueda deducir la frecuencia. Debido a que la frecuencia de la señal va a ser reducida, siendo su frecuencia máxima, aproximadamente, 80 Hz, la duración de la función “pulseIN” puede ser relativamente alta. Por ello, se opta por la utilización de un contador.

7.6.2. Frecuencia de muestreo

Debido a que la planta en la que el sistema de adquisición de datos va a estar instalado está alimentada mediante energía solar, la variación de las magnitudes se va a producir de manera progresiva a lo largo del día. Por ello, para reducir la cantidad de datos que deberán ser tratados por los usuarios, se opta por subir al servidor las magnitudes cada minuto. Sin embargo, para no pasar por alto posibles variaciones puntuales, como el paso de una nube por delante del panel solar, se establece un periodo de muestreo de 10 s. Finalmente, al pasar el minuto, se realizará la media de las 6 medidas de cada variable y se enviarán los resultados al servidor.

En cuanto al tiempo que transcurre entre cada muestreo, se opta por poner el microcontrolador en un modo de bajo consumo. Esto reduce en gran medida el consumo energético del Arduino, siendo un factor importante en este tipo de sistemas.

7.6.3. Protocolo de comunicación

Para la comunicación del Arduino con ThingsBoard se pueden emplear los protocolos HTTP y MQTT. Debido a que la alimentación del sistema de adquisición de datos se realiza por medio del panel fotovoltaico, se pretende conseguir la mayor eficiencia energética posible. Por ello, se selecciona el protocolo MQTT, al permitir un uso más eficiente de la energía.

7.6.4. Conexionado

El Arduino MKR 1010 cuenta con 7 pines analógicos, de los cuales 1 es un DAC. Por lo tanto, el conexionado de las 6 magnitudes analógicas se realizará en los pines analógicos que

no contienen el DAC, por posibles futuros usos que se le pueda dar. En cuanto a la magnitud digital, se conectará al pin digital 0 por comodidad a la hora de realizar el circuito. De este modo, se obtienen las siguientes conexiones:

$$V_{PLACA} \rightarrow Pin\ A1$$

$$I_{PLACA} \rightarrow Pin\ A2$$

$$V_{BOMBA} \rightarrow Pin\ A3$$

$$I_{BOMBA} \rightarrow Pin\ A4$$

$$V_{PRESION} \rightarrow Pin\ A5$$

$$V_{RADIACION} \rightarrow Pin\ A6$$

$$Caudal \rightarrow Pin\ 0$$

8 RESULTADOS FINALES

A lo largo de este apartado se realizará una descripción del producto final obtenido, indicando cuales son sus características principales.

8.1. Circuitos de acondicionamiento

8.1.1. Acondicionamiento de la tensión de la placa

Teniendo en cuenta los cálculos realizados en el Apartado 11.1 se seleccionan las resistencias normalizadas, con tolerancia E24, que no superan el fondo de escala del ADC. Estas resistencias son: $R_1 = 130k\Omega$ y $R_2 = 10k\Omega$. De este modo se obtiene el siguiente circuito de acondicionamiento:

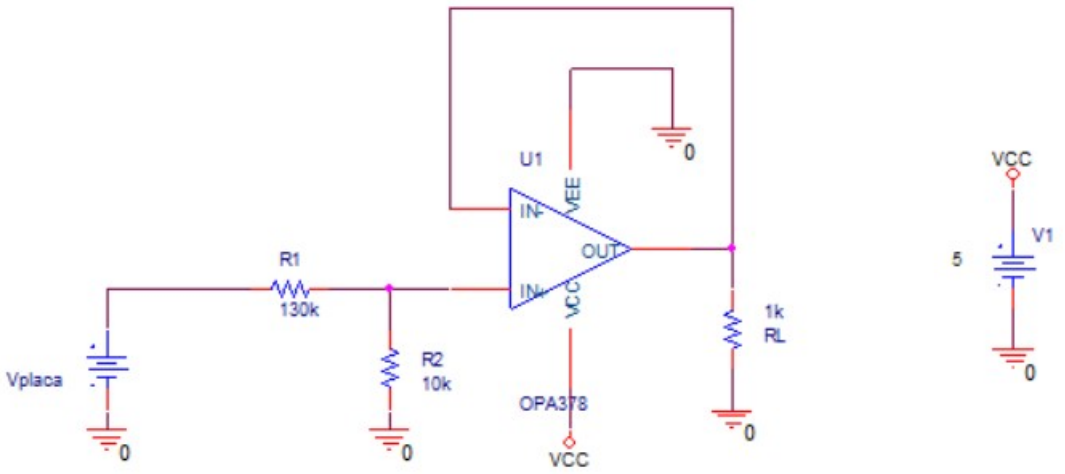


Figura 8.1 – Circuito de acondicionamiento de la tensión de la placa fotovoltaica

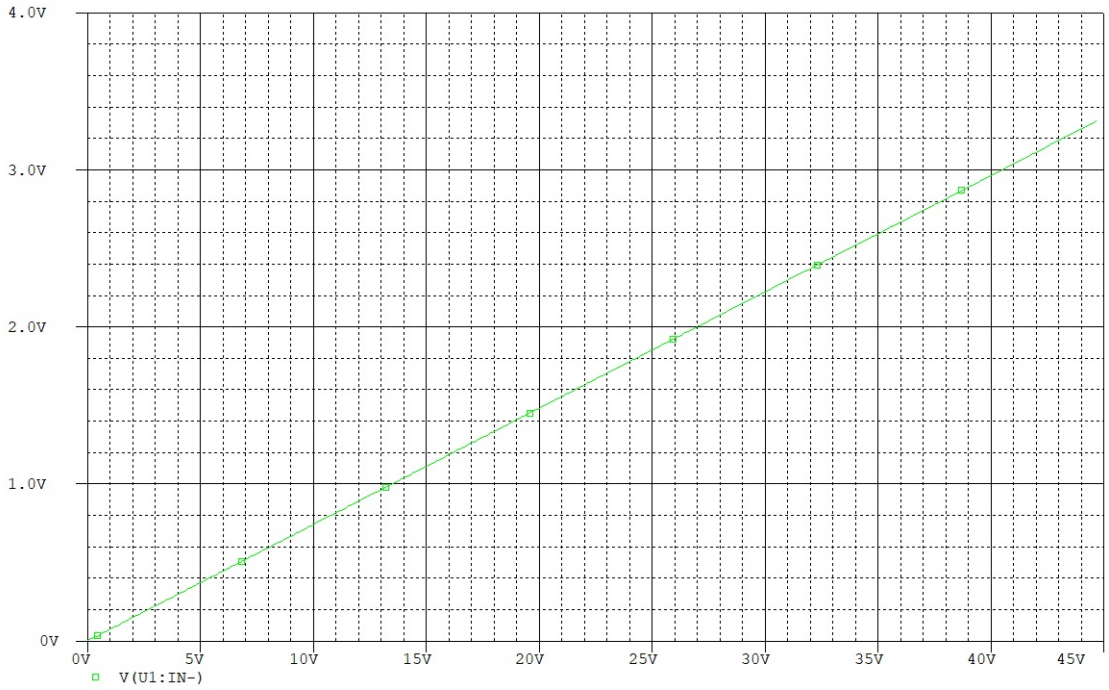


Figura 8.2 – Simulación del circuito de acondicionamiento de la tensión de la placa.

Como se puede ver en la simulación del circuito (Figura 8.2), para tensiones en la entrada en el rango de 0 hasta 45V, se obtienen tensiones en la salida que van de 0 a 3,3V.

Debido al uso de resistencias normalizadas, no es posible realizar la reducción de la tensión de 45 V a 3.3 V exactos, si no a un valor menor. Teniendo en cuenta la ecuación 11.1, se tiene que:

$$V_{OUT} = 45 * \frac{10 * 10^3}{130 * 10^3 + 10 * 10^3} = 3{,}214V \tag{8.1}$$

Por lo tanto, aunque el ADC se ha configurado con 12 bits y el valor digital máximo teórico es $2^{12} = 4096$, el valor máximo digital será de $3{,}214 * 4095/3{,}3 = 3989$ cuentas. Por lo tanto, la

resolución será:

$$LSB = \frac{45V}{3989} = 0{,}01128V = 11{,}28mV \tag{8.2}$$

8.1.2. Acondicionamiento de la corriente de la placa

Teniendo en cuenta los cálculos realizados en el Apartado 11.2 se seleccionan las resistencias normalizadas, con tolerancia E24, que no superen el fondo de escala del ADC. Esta resistencia es $R_L = 51k\Omega$. De este modo se obtiene el siguiente circuito de acondicionamiento.

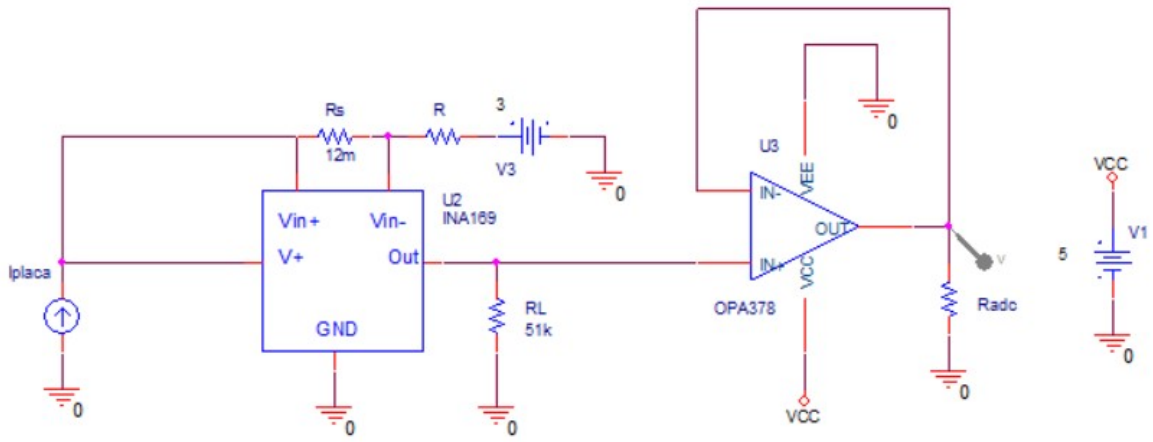


Figura 8.3 – Circuito de acondicionamiento de la corriente de la placa fotovoltaica

Debido a que el rango de tensiones en modo común del INA169 es de 2.7 a 60 V, en la simulación se incorpora una fuente de tensión de 3 V ya que, en la práctica nunca va a haber tensiones en modo común tan bajas. Si no se hiciese esto el operacional mostraría un comportamiento extraño cuando la corriente es cercana a 0 A, al ser la tensión en modo común inferior al mínimo.

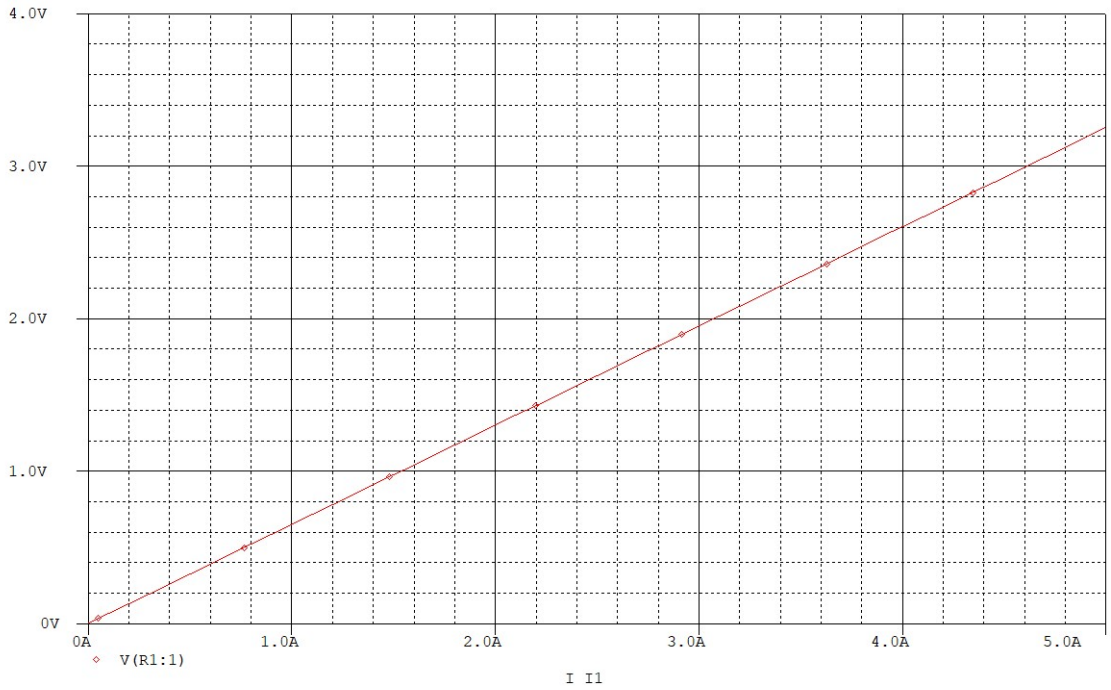


Figura 8.4 – Simulación del circuito de acondicionamiento de la corriente en la placa.

Como se puede ver en la simulación del circuito (Figura 8.4), para una corriente en la entrada en el rango de 0 a 5A ,se obtiene una tensión de salida que va de 0 a 3.3V.

Sin embargo, teniendo en cuenta la ecuación 11.4 se puede ver que, debido al uso de resistencias normalizadas, no se abarca todo el rango del ADC:

$$V_o = \frac{5{,}09 * 12 * 10^{-3} * 51 * 10^3}{1k\Omega} = 3{,}054V$$

Por lo tanto, el valor máximo digital del ADC será $3{,}054 * 4095/3{,}33 = 3790$ y, por lo tanto, la resolución será:

$$LSB = \frac{5{,}09A}{3790} = 1{,}34mA \tag{8.3}$$

8.1.3. Acondicionamiento de la tensión de la bomba

Aunque el circuito de acondicionamiento de la tensión en la bomba es el mismo que el de la tensión de la placa, este se recoge en un apartado diferente. Esto se debe a que el esquema en el que se realiza la simulación se especifica que la tensión medida es en el negativo de la bomba respecto al negativo de la placa fotovoltaica.

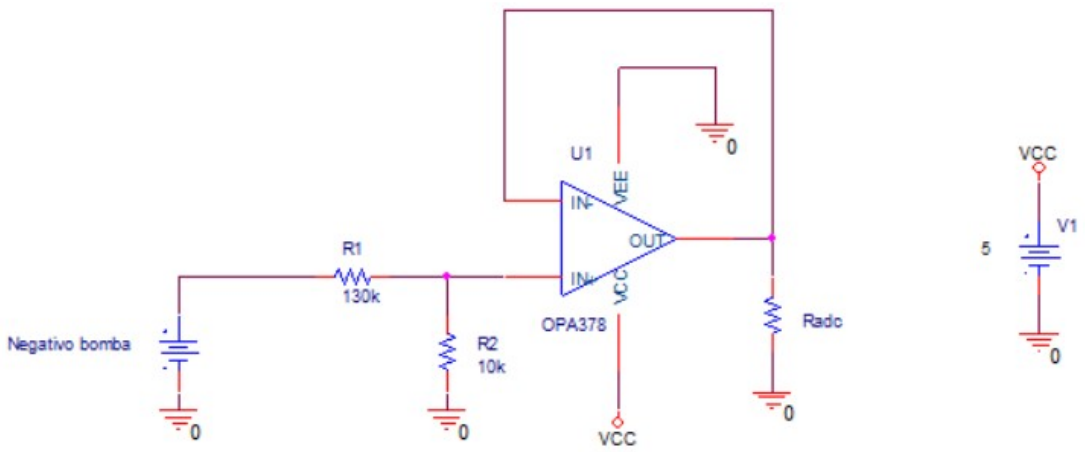


Figura 8.5 – Circuito de acondicionamiento de la tensión de la bomba de agua

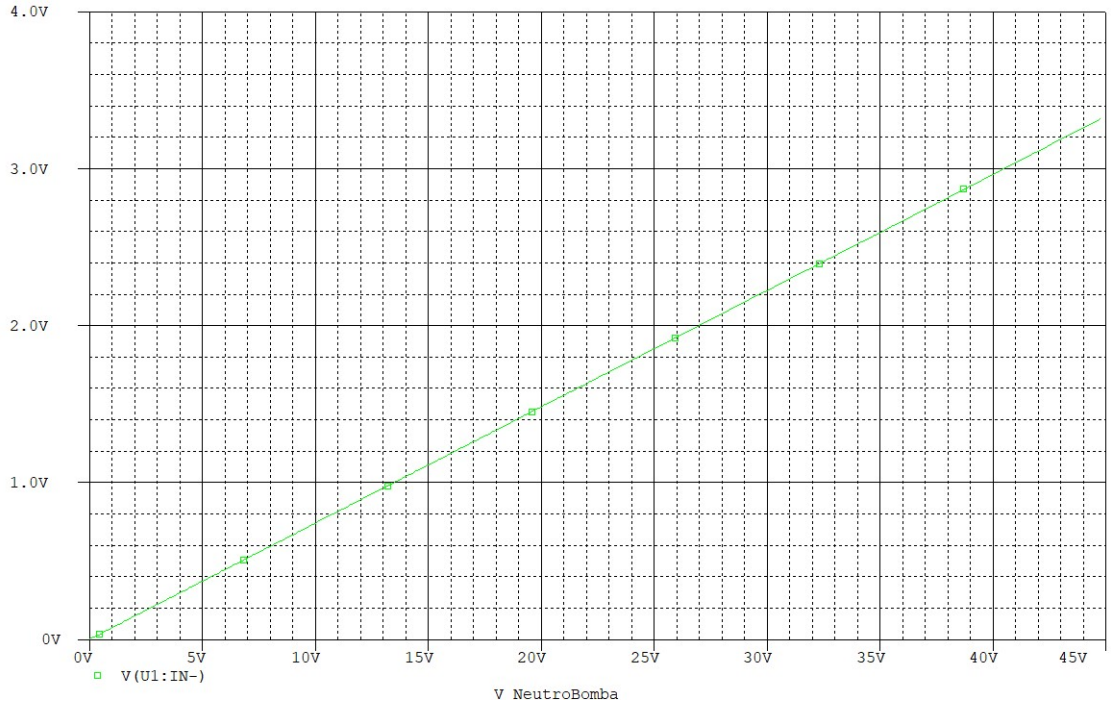


Figura 8.6 – Simulación del circuito de acondicionamiento de la tensión en la bomba

Como se puede ver en la simulación del circuito (Figura 8.6), para una tensión en el rango de 0 a 45V, se obtiene una tensión de salida que va de 0 a 3,3V.

Del mismo modo que en el apartado 8.1.1 se obtiene que:

$$LSB = \frac{45V}{3989} = 0{,}01128V = 11{,}28mV \tag{8.4}$$

8.1.4. Acondicionamiento de la corriente de la bomba

Teniendo en cuenta los cálculos realizados en el Apartado 11.3 se seleccionan las resistencias normalizadas, con tolerancia E24, que no superen el fondo de escala del ADC. Estas

resistencias es $R_{L1} = 51k\Omega$ y $R_{L2} = 3k\Omega$. De este modo se obtiene el siguiente circuito de acondicionamiento.

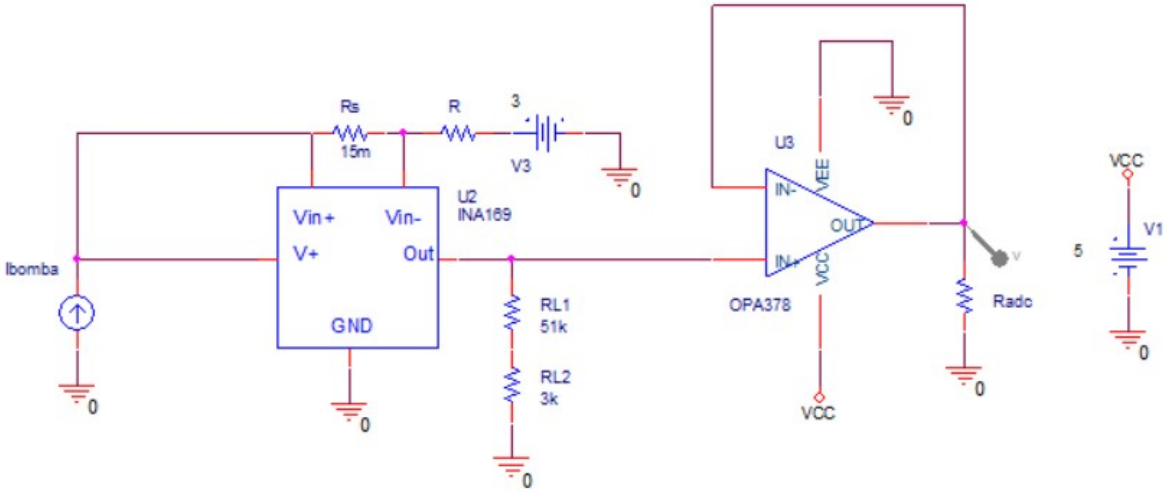


Figura 8.7 – Circuito de acondicionamiento de la corriente de la bomba de agua

Del mismo modo que en el Apartado 8.1.2 se incorpora la fuente de tensión de 3V para evitar comportamientos extraños en la simulación.

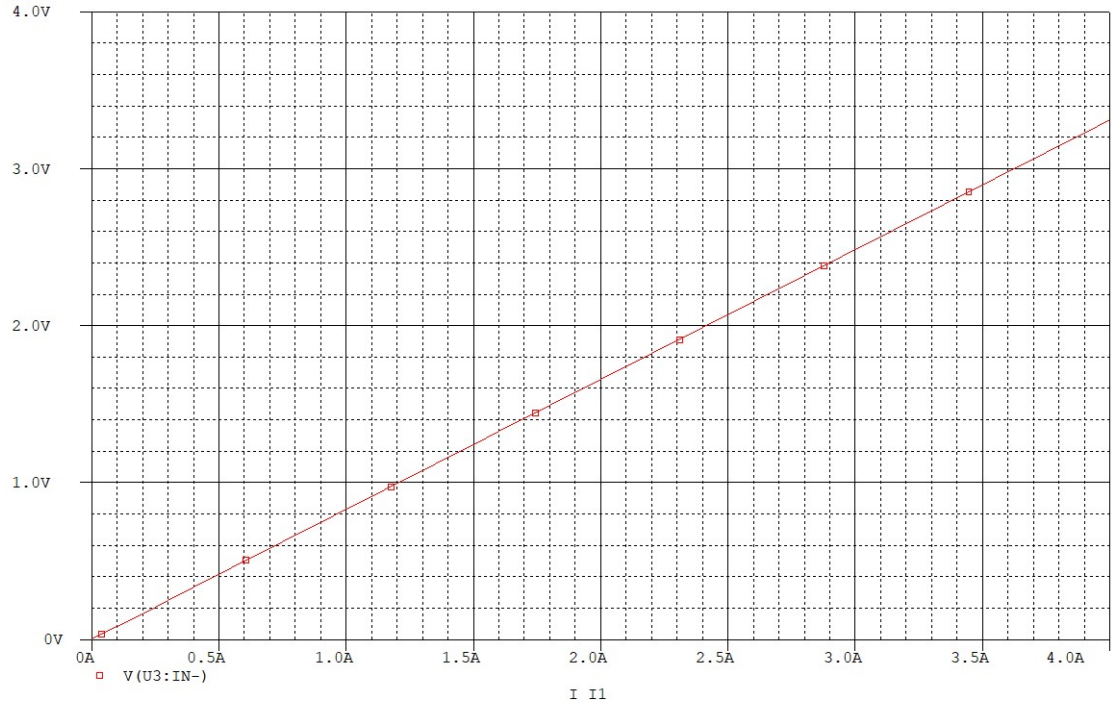


Figura 8.8 – Simulación del circuito de acondicionamiento de la corriente en la bomba

Como se puede ver en la simulación del circuito (Figura 8.8), para una corriente de entrada que va de 0 a 4A, se obtiene una tensión de salida en el rango de 0 a 3.3V.

Sin embargo, teniendo en cuenta la ecuación 11.4 se puede ver que, debido al uso de resistencias normalizadas, no se abarca todo el rango del ADC:

$$V_o = \frac{4 * 15 * 10^{-3} * 54 * 10^3}{1k\Omega} = 3{,}24V$$

Por lo tanto, el valor máximo digital del ADC será $3{,}24 * 4095/3{,}3 = 4021$ y, por lo tanto, la resolución será:

$$LSB = \frac{4A}{4021} = 0{,}995mA \tag{8.5}$$

8.1.5. Acondicionamiento de la señal generada por el sensor de presión a la salida de la bomba

Teniendo en cuenta los cálculos realizados en el Apartado 11.4.2 se seleccionan las resistencias normalizadas, con tolerancia E24, que no superen el fondo de escala del ADC. Estas resistencias son:

$$R_1 = 15k\Omega$$

$$R_2 = 7{,}5k\Omega$$

$$R_3 = 15k\Omega$$

$$R_4 = 110k\Omega$$

$$R_6 = 12k\Omega$$

$$R_7 = 4{,}3k\Omega$$

De este modo se obtiene el siguiente circuito de acondicionamiento.

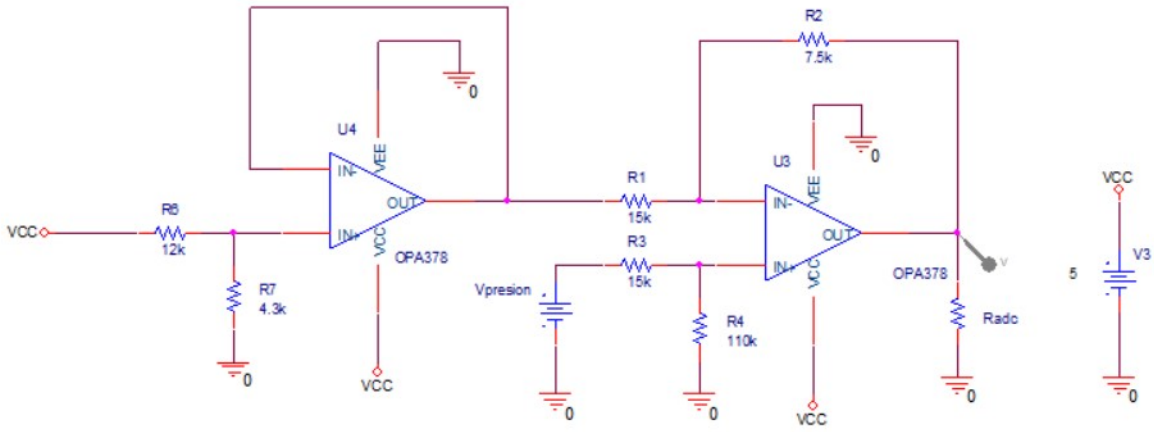


Figura 8.9 – Circuito de acondicionamiento de la presión la salida de la bomba de agua

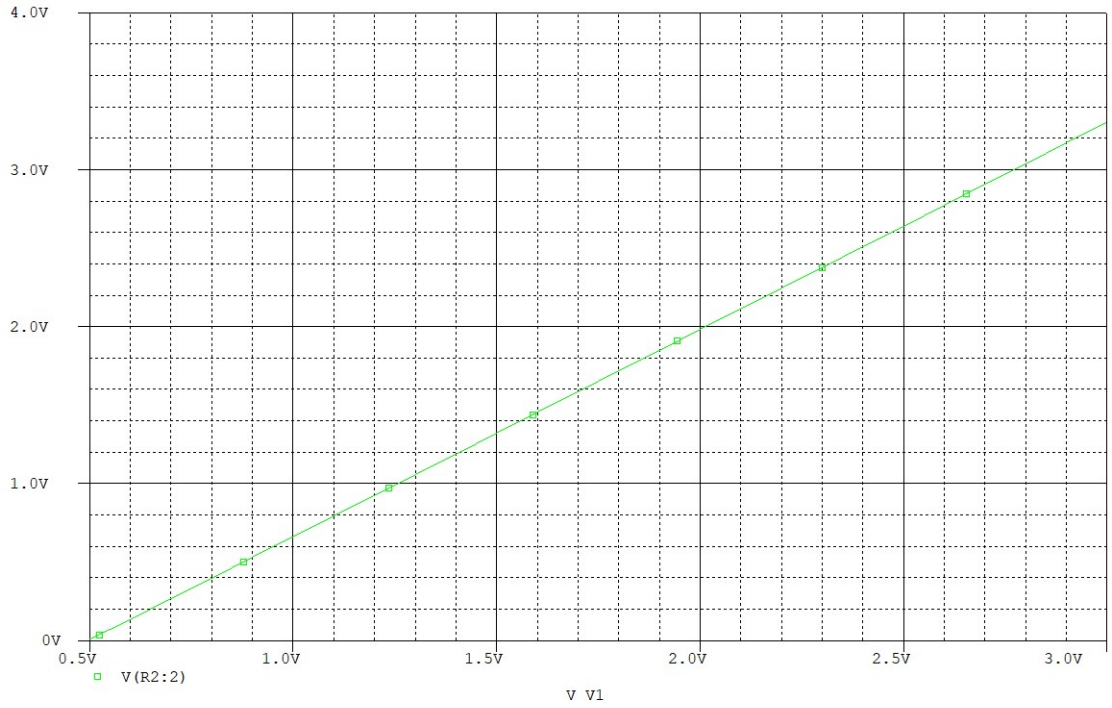


Figura 8.10 – Simulación del circuito de acondicionamiento de la presión

Como se puede ver en la simulación del circuito (Figura 8.10) para una tensión del sensor de 0.5 a 3V la tensión de salida va de 0 a 3.3V.

Teniendo en cuenta la Ecuación 11.10 y los valores de las resistencias obtenidos en el Apartado 7.4.3 se tiene que:

$$V_o = 4{,}5 * \frac{110 * 10^3}{15 * 10^3 + 110 * 10^3} * (1 + \frac{7{,}5 * 10^3}{15 * 10^3}) - 1{,}32 * \frac{7{,}5 * 10^3}{15 * 10^3} = 3{,}3V$$

Por lo tanto, el valor digital máxim del ADC será 4095. De este modo la resolución será:

$$LSB = \frac{0{,}788MPa}{4095} = \frac{7{,}88bar}{4095} = 1{,}92mbar$$

8.1.6. Acondicionamiento de la señal generada por el sensor de caudal

De acuerdo a lo expuesto en el Apartado 7.4.4, se obtiene el siguiente circuito de acondicionamiento:

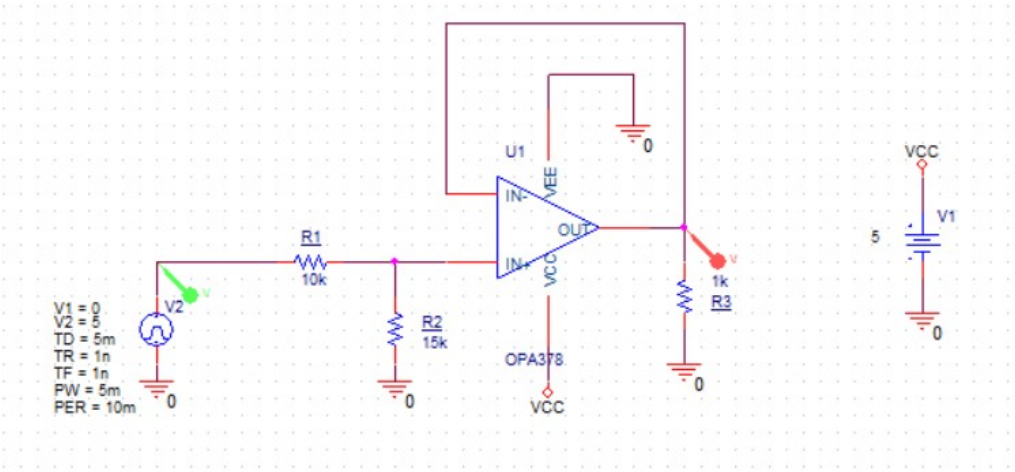


Figura 8.11 – Circuito de acondicionamiento de la señal del sensor de caudal

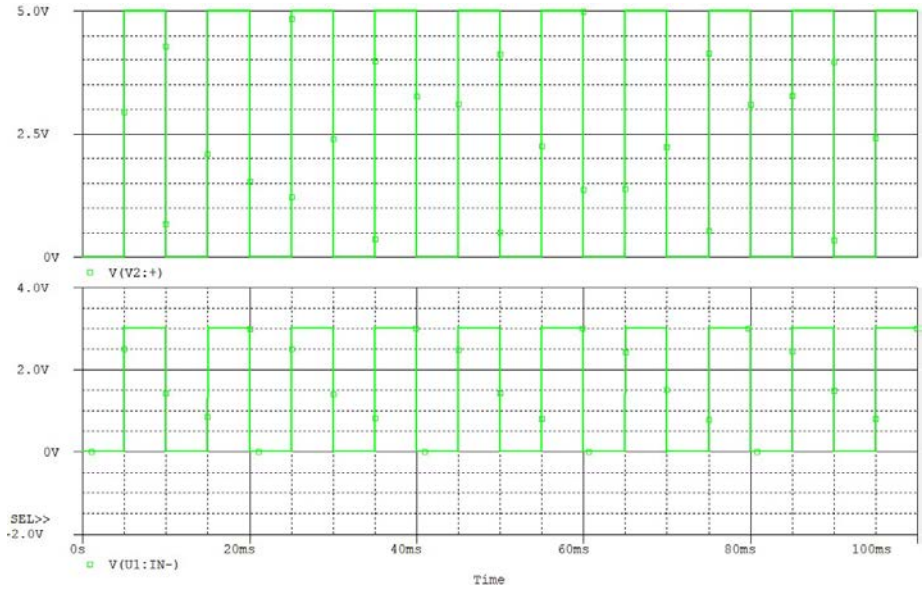


Figura 8.12 – Simulación del circuito de acondicionamiento del caudal

Como se puede ver en la simulación del circuito (Figura 8.12) la amplitud de los pulsos se reduce de 5V a 3V. Además, se puede comprobar que no hay deformaciones en la señal cuadrada.

En cuanto a la obtención del caudal en l/min, se ha observado que la ecuación proporcionada por el fabricante tiene un gran error. Por ello, se prueba el sensor con diferentes caudales, con el fin de obtener la relación aproximada entre el caudal y la frecuencia de la señal. De este modo se obtiene que la relación media es 10.53. Por lo tanto, para la obtención del caudal en l/min habrá que dividir la frecuencia por ese valor.

8.1.7. Acondicionamiento señal generada por la célula calibrada

Como se expone en el Apartado 7.4.5 se ha realizado la medición de la radiación solar mediante el medidor de radiación solar de MacSolar y mediante la célula calibrada, a través

del Arduino, con el fin de poder calibrarla. Tomando valores durante varios días se obtiene la siguiente gráfica, en la que se representa la radiación solar y el valor digital del ADC:

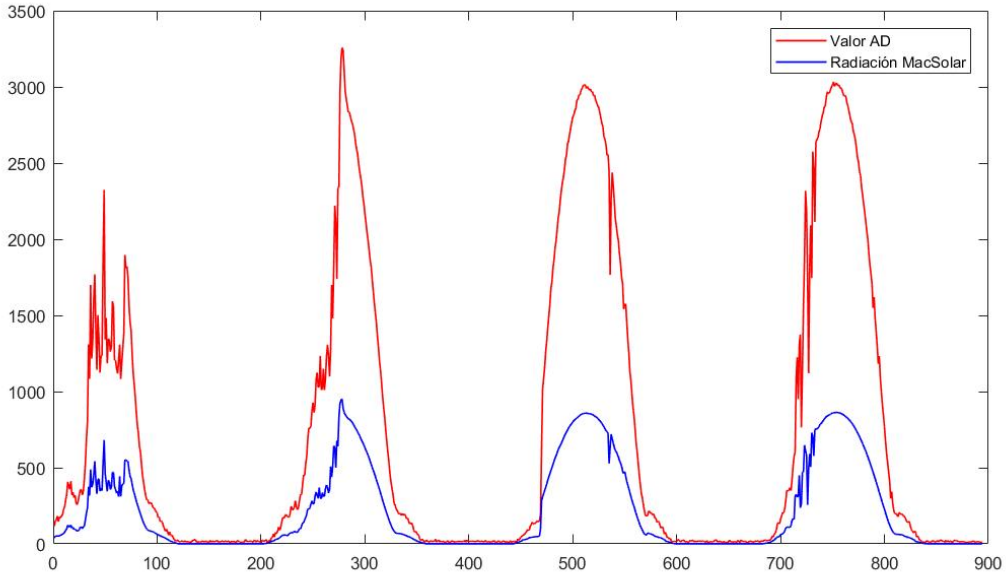


Figura 8.13 – Gráficas de la radiación solar I

Calculando la relación media entre la radiación solar medida por el dispositivo de MacSolar y el valor digital del ADC, se obtiene que esta es 3.33, siendo el error medio de la medida de solo 12.15 W/m^2. Por lo tanto, para la obtención de la radiación solar en W/m^2 habría que dividir el valor digital del ADC entre este valor.

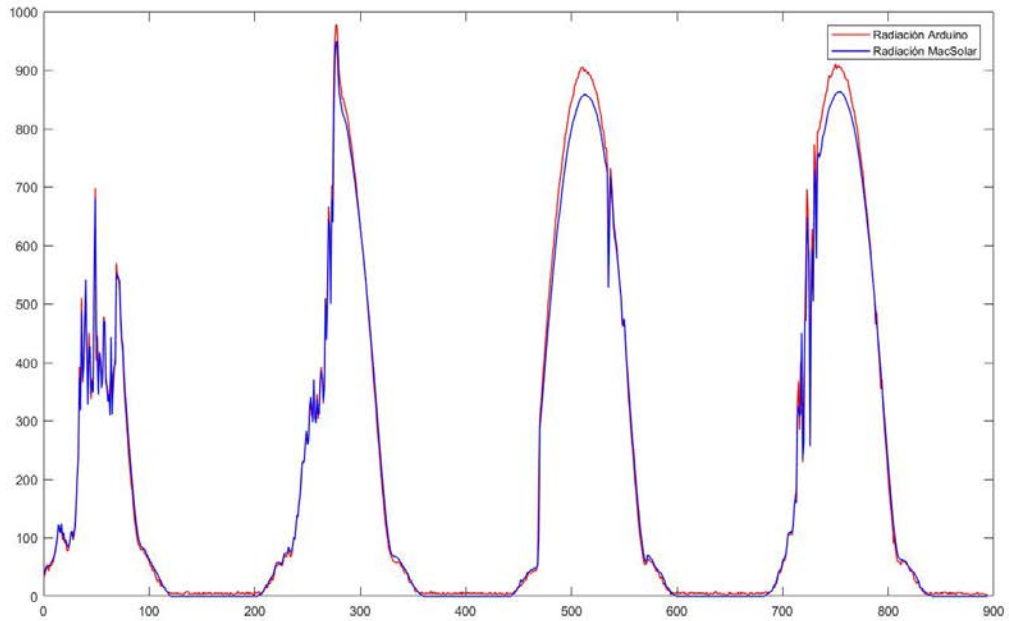


Figura 8.14 – Gráficas de la radiacion solar 2

8.2. Fabricación del PCB

Una vez simulados todos los circuitos de acondicionamiento se procede a la implementación física del sistema de adquisición de datos. Para ello se decide encargar la fabricación de un PCB.

En primer lugar, se realiza el esquemático de los circuitos de acondicionamiento desarrollados previamente, incorporando además los siguientes elementos:

- Condensadores en las alimentaciones de los integrados: $0{,}1\mu F$, estipulados por los fabricantes en las hojas de características.

- Diodos *schottky* en los pines de entrada de lo integrados para su protección. La tensión máxima que pueden soportar los pines del Arduino es de VDD + 0.6V. Por ello, se instala el diodo schottky NSR0530HT1G de ON Semiconductor; cuya caída de tensión, para una corriente de 100 mA, es de 0.37 V.

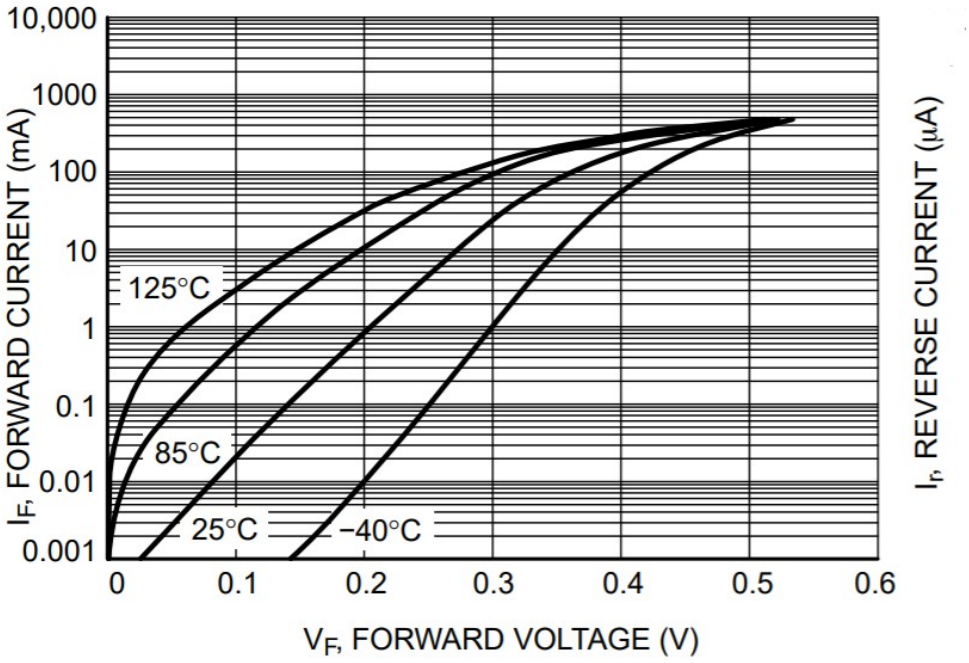


Figura 8.15 – Tensión - Corriente diodo NSR0530HT1G

- Arduino MKR 1010.

- Elementos de conexión. Para la conexión de los conductores, a través de los cuales se realizaran las medidas y la alimentación de los componentes, se decide incluir una borna de circuito impreso. Para los conductores de las señales de entrada es necesario incorporar una borna de 12 contactos. Debido a que encontrar una borna con tantos contactos es complicado, se instalan 6 bornas de 2 contactos. Para los contactos de alimentación de los sensores de caudal y presión, se instalan 2 bornas de 2 contactos.

Figura 8.16 – Borna de dos contactos

Respetando las conexiones del Arduino establecidas en el Apartado 7.6.4, se realiza el esquemático recogido en el Plano 3, con el cual se diseña el PCB recogido en el Plano 4.

Finalmente, se obtiene la siguiente placa de circuito impreso:

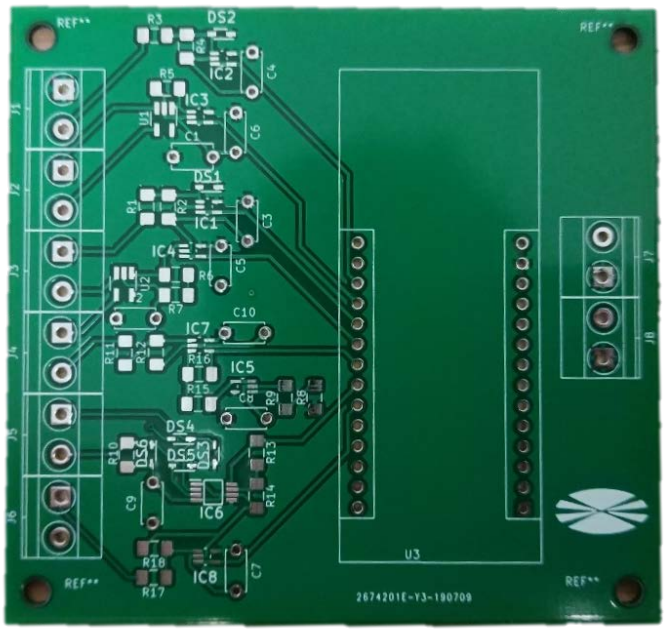


Figura 8.17 – Anverso del PCB

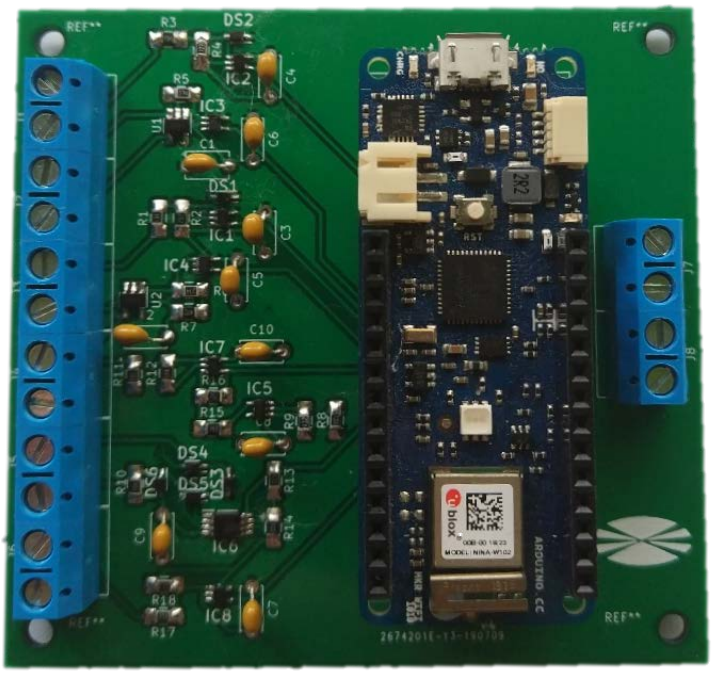


Figura 8.18 – Resultado final del PCB

8.3. Implementación en la planta

Para la implementación del sistema de adquisición de datos en la planta de bombeo fotovoltaico, se incorporan los conductores necesarios, numerándolos del siguiente modo:

- Código comenzando por 3: el conductor procede del cuadro de protecciones.
- Código comenzando por 5: el conductor procede del regulador 902-200 (LCB-G).

De este modo, se obtiene el esquema de conexionado recogido en el Plano 2.

El alojamiento de la placa de circuito impreso y el convertidor DC-DC se realizará en una caja impresa en 3D, diseñada específicamente para tal uso. Debido a que el Arduino estará integrado permanentemente dentro de la caja, esta será diseñada con una apertura en la parte superior, a través de la cual, se podrá introducir un cable microUSB para reprogramar el Arduino, en caso de que sea necesario. La información necesaria para la fabricación de la caja se encuentra recogida en los Planos 5, 6, 7 y 8.

El resultado final de la instalación es el siguiente:

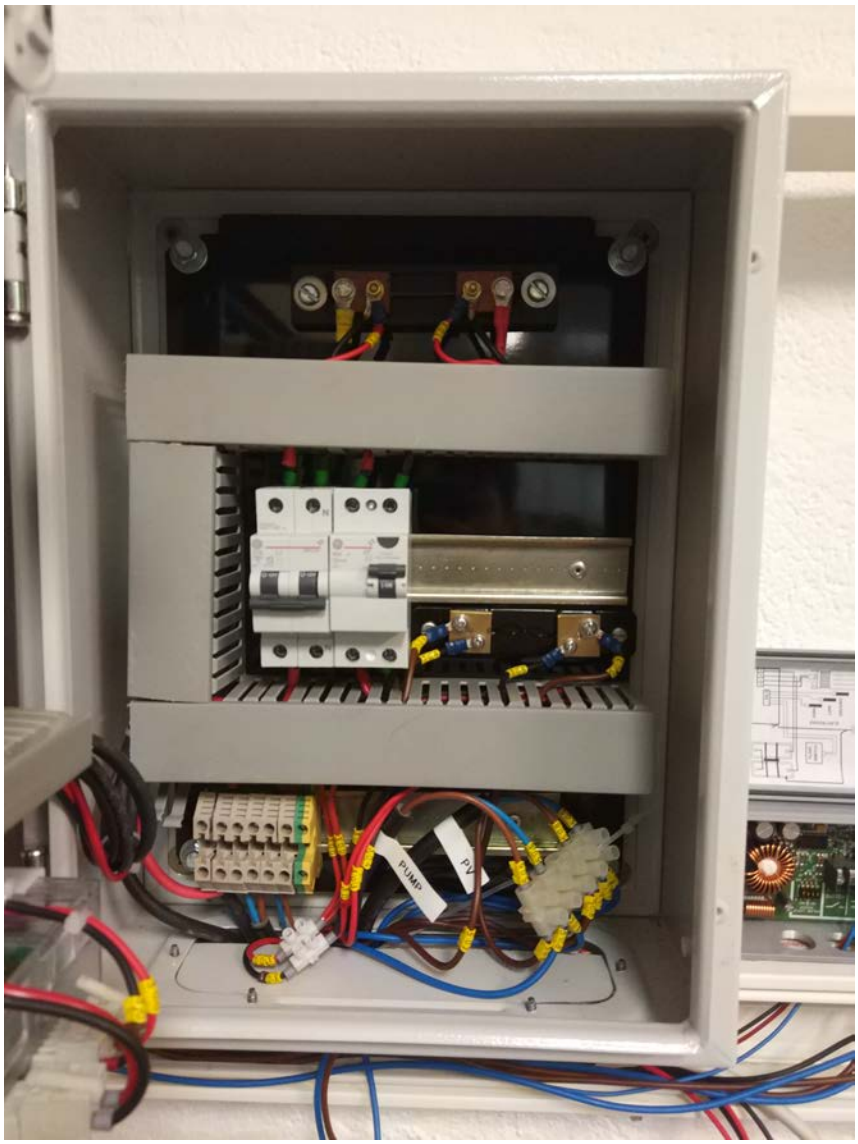


Figura 8.19 – Cuadro de protecciones

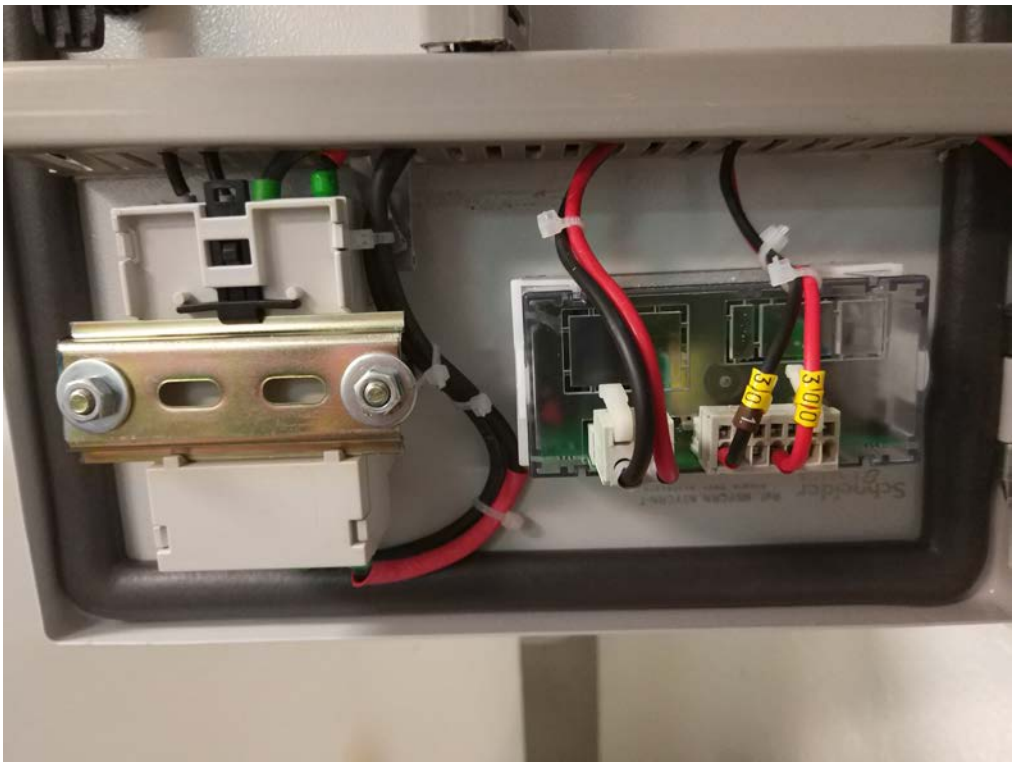


Figura 8.20 – Puerta del cuadro de proteccion

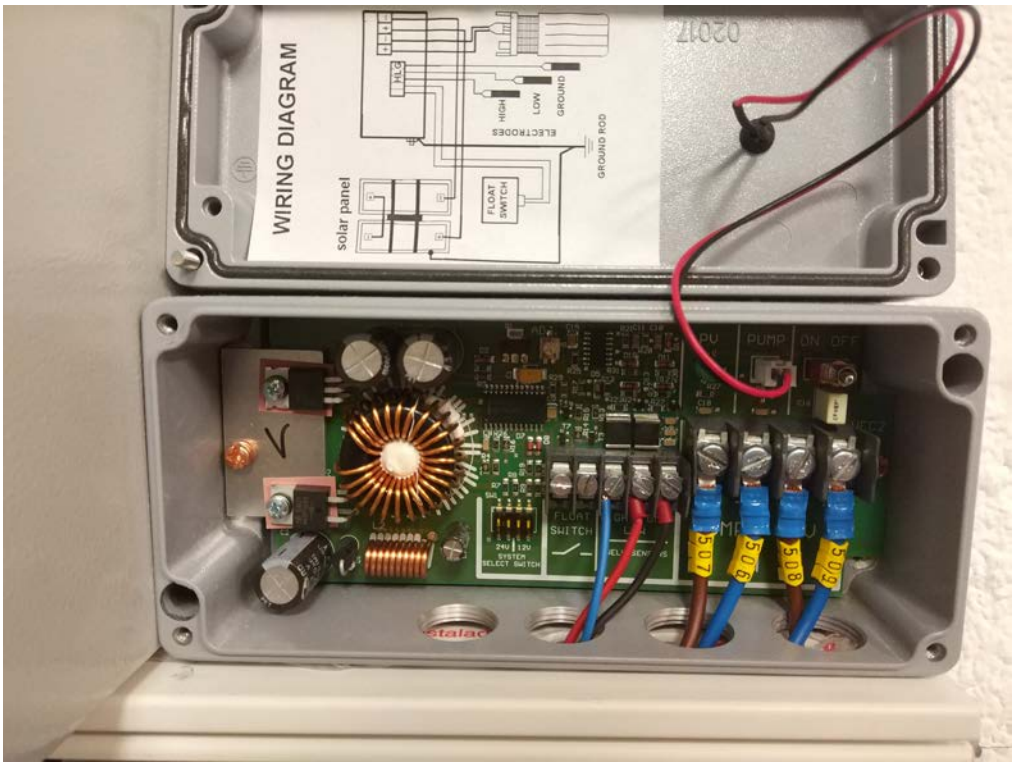


Figura 8.21 – Conexión del regulador

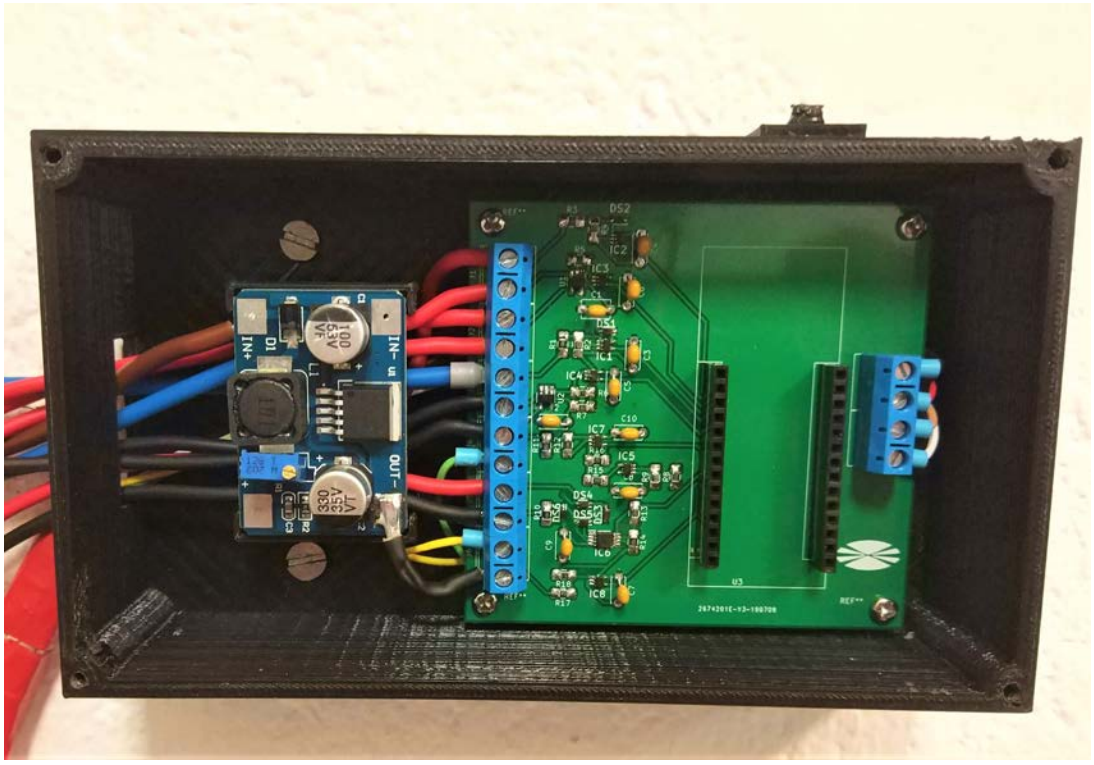


Figura 8.22 – Conexión del convertidor DC-DC y la placa diseñada

8.4. Configuración del servidor

Como se expone en el apartado 7.5.9, la mejor solución para la implementación de un servidor es instalar ThingsBoard en una máquina privada. Por ello, se realiza la instalación del servidor en un ordenador próximo a la planta, siguiendo las indicaciones del Apartado 13.5.2.

Sin embargo, el servidor va a estar conectado a la red “udc.pri¨y el Arduino a la red “udc-portal”. Debido a esto, es imposible que ambos dispositivos se puedan comunicar utilizando ThingsBoard como servidor en una red local. Por ello, se configura el puerto ethernet del ordenador para funcionar con una IP pública, asignándole los siguientes parámetros:

De este modo, Arduino es capaz de comunicarse para realizar el envío de la información. Además, esto proporciona beneficios adicionales, como la posibilidad de acceder al servidor desde cualquier red que tenga conexión a internet.

Finalmente, de manera análoga a como se indica en el Apartado 13.5.1, se configura el servidor para recibir los datos del Arduino y representarlos gráficamente en un panel. De este modo se obtiene la siguiente interfaz gráfica:

Figura 8.23 – Representación gráfica resultante

Como se puede ver en la figura 8.23 las variables se agrupan en 5 gráficas, separándolas según la magnitud a representar:

- Primera gráfica: representación de la tensión en la placa y la bomba.
- Segunda gráfica: representación de la corriente de la placa y la bomba.
- Tercera gráfica: representación de la presión a la salida de la bomba.
- Cuarta gráfica: representación de la radiación solar.
- Quinta gráfica: representación del caudal de la bomba.

8.5. Programación del Arduino

Finalmente, se programa el Arduino para la toma y envío de datos mediante el código de programación 12.1.1. Con este programa se realizan las medidas cada 10 segundos, realizando la conversión a la magnitud correspondiente. Una vez realizadas 6 medidas, se calcula su media y se envía al servidor utilizando la librería “ThingsBoard.h”.

Mediante la modificación de los registros internos del microcontrolador se realizan las siguientes configuraciones de los periféricos:

- Habilitación del modo de bajo consumo.
- Habilitación y configuración del sistema de interrupciones.

- Configuración del RTC para “despertar” al microcontrolador cada 10 s.
- Habilitación y configuración de las interrupciones externas por flanco ascendente en el pin 0.
- Configuración del timer TCC0 para contar el número de pulsos en el pin 0.
- Ajuste del offset y la ganancia del convertidor AD.

El flujo que seguirá la ejecución del programa será:

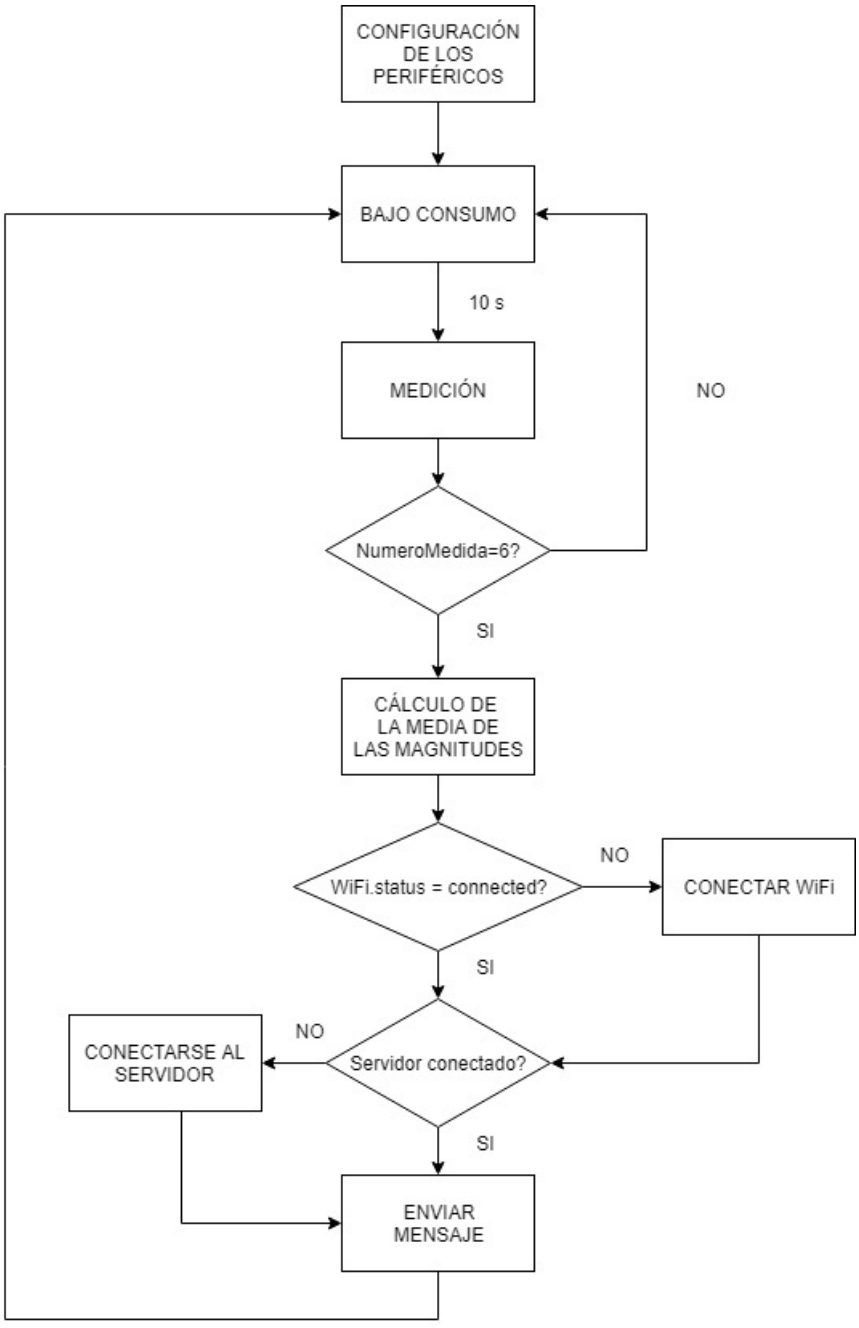


Figura 8.24 – Diagrama de flujo del programa

8.6. Descarga de los datos almacenados en el servidor

El fin último del almacenamiento de los datos de la planta en un servidor es el dar la posibilidad de trabajar con ellos para poder estudiar la planta. Por ello, siguiendo las indicaciones del Apartado 13.5.4.1 se pueden descargar los datos en formato .csv. Sin embargo, este archivo contiene los datos agrupados de un modo que no es cómodo para trabajar con ellos. Por ello, se desarrolla en Matlab el script 12.2.2. Ejecutando este script, se agrupan por columnas las diferentes variables, como se puede ver en la Figura 8.25.

	1 Fecha	2 TensionPlaca	3 CorrientePlaca	4 TensionBomba	5 CorrienteBomba	6 Presion	7 Radiacion
1	""2019-09-...	28.0547	3.2458	16.4737	2.4580	2.1302	737.6877
2	""2019-09-...	28.1205	3.2662	17.8369	2.4493	2.0640	743.2933
3	""2019-09-...	28.1732	3.2417	15.0051	2.4484	2.8602	744.3443
4	""2019-09-...	28.1901	3.2853	17.5361	2.4532	1.9480	744.8948
5	""2019-09-...	28.1525	3.2638	16.5508	2.4532	2.3602	740.0400
6	""2019-09-...	28.0641	3.2524	19.6345	2.4512	1.6892	740.8408
7	""2019-09-...	28.0434	3.2417	18.8655	2.4514	1.8130	741.9419
8	""2019-09-...	28.1186	3.2647	19.4822	2.4547	1.8052	745.5456
9	""2019-09-...	28.1544	3.2219	19.6007	2.4532	1.8175	740.7908
10	""2019-09-...	28.0754	3.2496	19.5274	2.4542	1.6979	743.7938
11	""2019-09-...	28.0679	3.2458	19.1964	2.4549	1.7694	742.3423
12	""2019-09-...	28.1017	3.2504	19.2754	2.4507	1.8588	742.3423
13	""2019-09-...	28.0566	3.2401	20.4205	2.4632	1.6255	744.8448
14	""2019-09-...	28.1092	3.2524	19.7003	2.4561	1.7898	742.1421
15	""2019-09-...	28.2427	3.2485	19.6157	2.4496	1.7892	739.8899

Figura 8.25 – Resultado del tratamiento de datos

8.7. Conclusiones

Como se puede observar en los apartados anteriores; se ha implementado, de manera exitosa, un sistema de adquisición de datos sobre una planta de bombeo fotovoltaico, realizando el almacenamiento de los datos en un servidor. Este sistema facilita enormemente el estudio de este tipo de sistemas, al automatizar el proceso de la toma de datos, ya que hacerlo manualmente es inviable debido a la gran cantidad de variables que es necesario medir simultáneamente.

Debido a que el sistema se va a emplear de forma académica, las posibilidades que otorga ThingsBoard en cuanto a cuentas de usuario, con diferentes permisos, son de gran utilidad a la hora de permitir a un gran número de personas acceder al servidor.

Además, al instalar un servidor cuya IP es publica, se facilita el acceso a la información recogida desde diferentes dispositivos simultáneamente e incluso desde diferentes ubicaciones remotas.

9 ORDEN DE PRIORIDAD ENTRE LOS DOCUMENTOS

1 Planos

2 Pliego de Condiciones

3 Presupuesto

4 Memoria

TÍTULO: **DESARROLLO DE UN SISTEMA DE ADQUISICIÓN DE DATOS PARA PLANTA DE BOMBEO FOTOVOLTAICO.**

ANEXOS

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

Fdo.: MARTÍN NOVOA PELLO

Índice del documento ANEXOS

10 DOCUMENTACIÓN DE PARTIDA

10.1. Popuesta inicial de asignación del Trabajo Fin de Grado

ESCUELA UNIVERSITARIA POLITÉCNICA

ASIGNACIÓN DE TRABAJO FIN DE GRADO

En virtud de la solicitud efectuada por:
En virtud da solicitude efectuada por:

APELLIDOS, NOMBRE: *Novoa Pello,Martín*
APELIDOS E NOME:

DNI: [redacted] **Fecha de Solicitud:** Feb2019
DNI: *Fecha de Solicitude:*

Alumno de esta escuela en la titulación de Grado en Ingeniería en Electrónica Industrial y Automática, se le comunica que la Comisión de Proyectos ha decidido asignarle el siguiente Trabajo Fin de Grado:

O alumno de esta escola na titulación de Grado en Enxeñería en Electrónica Industrial e Automática, comunícaselle que a Comisión de Proxectos ha decidido asignarlle o seguinte Traballo Fin de Grado:

Título T.F.G:Desarrollo de un sistema de adquisición de datos para planta de bombeo fotovoltaico

Número TFG: *770G01A165*

TUTOR:(Titor) *Meizoso Lopez,Maria Del Carmen*

COTUTOR/CODIRECTOR: *Benigno Rodríguez Gómez*

La descripción y objetivos del Trabajo son los que figuran en el reverso de este documento*:*

A descrición e obxectivos do proxecto son os que figuran no reverso deste documento.

Ferrol a Lunes, 26 de Agosto del 2019

Retirei o meu Traballo Fin de Grado o dia _____ de _____ do ano _____

Fdo: Novoa Pello,Martín

DESCRIPCIÓN Y OBJETIVO:La EUP dispone de una planta de bombeo fotovoltaico directo, de dimensiones reducidas, instalada en Diciembre de 2017, con la que se pretende valorar el rendimiento de estos sistemas bajo diferentes condiciones de disponibilidad de recurso hídrico y radiación solar.
La planta consiste en una bomba, situada en un depósito de agua de 500 l, que se alimenta mediante un panel fotovoltaico a través de un regulador, lo que permite trasvasar el agua a otro depósito situado a cierta altura. Los depósitos están comunicados con una llave de paso de accionamiento manual que permite vaciar el superior sobre el que contiene la bomba. Además se dispone de otra llave de paso también manual que restringe la sección a la salida de la bomba para simular diferentes alturas manométricas.
Actualmente la planta no está automatizada ni en cuanto a operación ni a captura de datos, se están midiendo la tensión y corriente del panel fotovoltaico, así como la radiación solar que recibe el mismo, que se pueden ver en sendos displays, pero no se pueden registrar ni almacenar para hacer históricos o gráficos, esto limita en gran medida las posibilidades de experimentación y divulgación de la planta.

OBJETO:
Esta propuesta consiste en sensorizar algunas variables adicionales a las que ya se están midiendo tales como: presión a la salida de la bomba, caudal a la salida de la bomba, tensión y corriente a la entrada de la bomba, e incorporar todas las medidas a un sistema de adquisición de datos basado en Arduino, con posibilidad de comunicación remota, para poder registrarlos, procesarlos y mostrarlos a petición del usuario.
El registro de datos se puede llevar a cabo con un servidor local que permita crear una base de datos, o mediante una solución ?Platform as a Service? alojada en la ?nube?,esta última tecnología se encuentra a día de hoy en fase de expansión en las empresas, por lo que se considera de especial interés para el TFG.
Además, si se desea monitorizar los datos de instalaciones de este tipo distribuidas en zonas más o menos extensas y sin acceso a red cableada de datos, es interesante disponer de un sistema de adquisición de bajo coste, con sistema de comunicación, y con posibilidad de análisis y consulta en tiempo real, que permita históricos de datos sin necesidad de invertir en hardware específico para este último fin.
En cualquier caso, se desarrollará un servidor de datos local sobre un PC, que salvaguarde el fin último de este trabajo.

ALCANCE:
Se pretende diseñar e instalar físicamente el sistema de adquisición de datos. Las fases de desarrollo del trabajo podrían ser las siguientes:
1. Análisis y estudio de los diferentes sistemas Arduino
2. Análisis y estudio de los diferentes sensores posibles para esta aplicación.
3. Selección de los elementos hardware y adquisición de los mismos
4. Montaje de circuitos y pruebas
5. Programación de un servidor de datos local en PC.
6. Comparativa de distintas plataformas IoT (Internet of Things) para la adquisición, almacenamiento y visualización de datos recogidos por Arduino.
7. Si se encuentra alguna opción gratuita que se ajuste a los requerimientos del sistema, se proporcionará también esta solución mediante la programación de la plataforma y pruebas.

11 CÁLCULOS

11.1. Acondicionamiento de la tensión en la placa y en la bomba

Debido a que la ganancia del circuito de acondicionamiento de la placa fotovoltaica y de la bomba de agua es la misma, se trataran las dos juntas en este mismo apartado. En el caso de la placa Vin es la tensión en el positivo de la misma y en la bomba es la tensión en el negativo de esta.

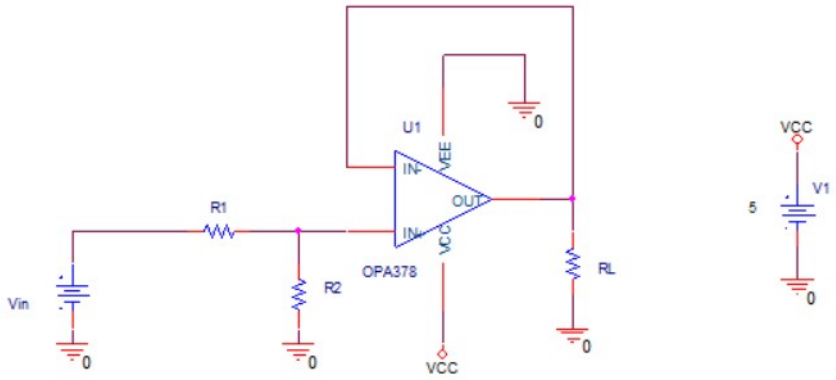


Figura 11.1 – Circuito de acondicionamiento de la tensión de la placa y de la bomba

$$V_{OUT} = V_{IN} * \frac{R_2}{R_1 + R_2} \tag{11.1}$$

Siendo:

$$V_{IN} = 45V$$

$$V_{OUT} = 3{,}3V$$

Se tiene que:

$$G = \frac{R_2}{R_1 + R_2} = \frac{3{,}3}{45} = 0{,}733$$

Así:

$$R_1 = 12{,}33 * R_2 \tag{11.2}$$

11.2. Acondicionamiento de la corriente en la placa

La resistencia *shunt* con la que se va a medir la corriente en la placa es de 5A 60 mV, que según la Ley de Ohm:

$$R_S = \frac{60 * 10^{-3}}{5} = 12m\Omega \tag{11.3}$$

Para el acondicionamiento de la corriente se emplea el INA169 en la siguiente configuración:

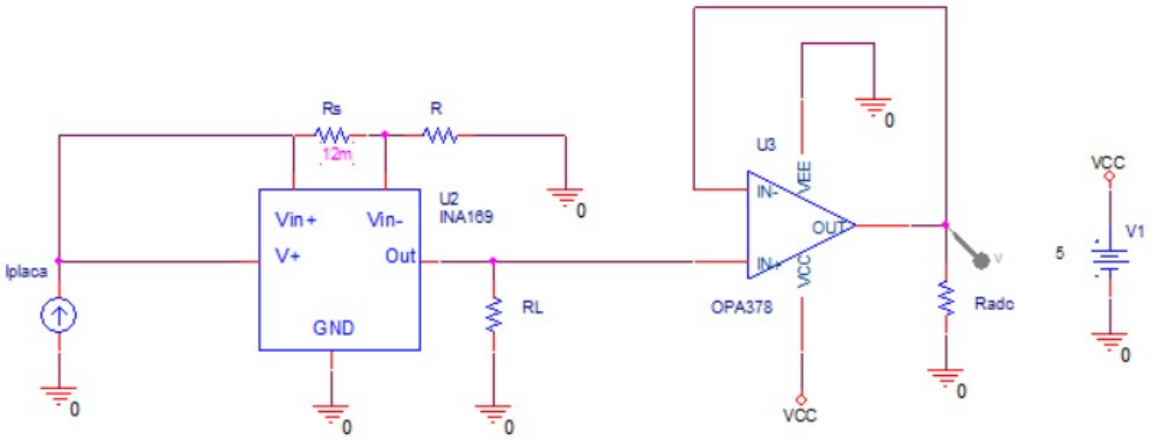


Figura 11.2 – Circuito de acondicionamiento de la corriente de la placa

Sabiendo que la Funcion de transferencia del INA169 es:

$$V_o = \frac{I_S * R_S * R_L}{1k\Omega} \tag{11.4}$$

Y siendo:

$$I_S = 5{,}09A$$

$$R_S = 12m\Omega$$

$$V_o = 3{,}3V$$

Se obtiene que:

$$R_L = \frac{3{,}3 * 1 * 10^3}{5{,}09 * 12 * 10^{-3}} = 54027{,}5\Omega \tag{11.5}$$

11.3. Acondicionamiento de la corriente en la bomba

La resistencia *shunt* con la que se va a medir la corriente en la placa es de 5A 75 mV, que según la Ley de Ohm:

$$R_S = \frac{75 * 10^{-3}}{5} = 15m\Omega \tag{11.6}$$

Para el acondicionamiento de la corriente se emplea el INA169 en la siguiente configuración:

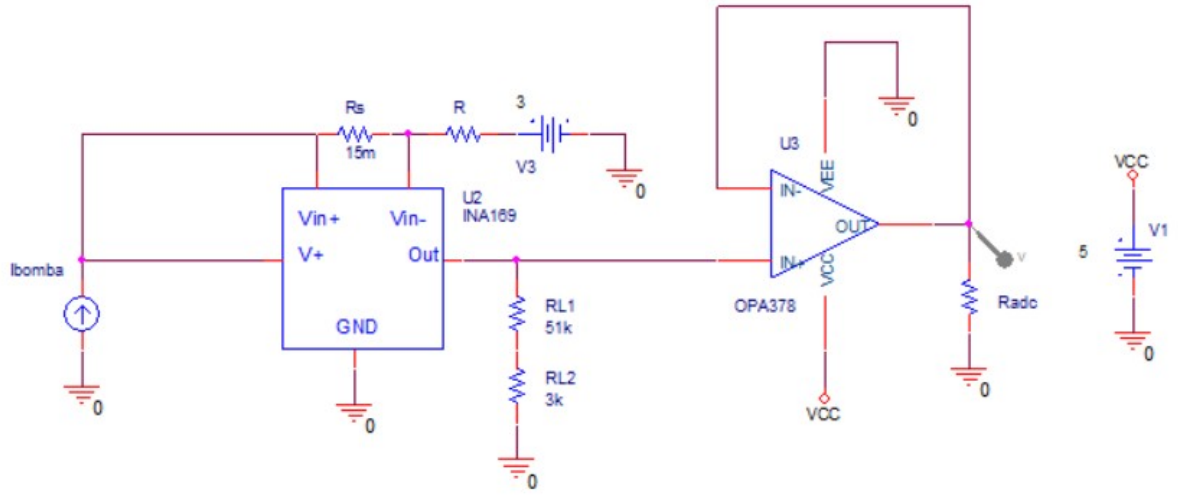


Figura 11.3 – Circuito de acondicionamiento de la corriente de la bomba

Según la Ecuación 11.4 y sabiendo que:

$$I_S = 4A$$

$$R_S = 15m\Omega$$

$$V_o = 3{,}3V$$

Se obtiene que:

$$R_L = \frac{3{,}3 * 1 * 10^3}{4 * 15 * 10^{-3}} = 55000\Omega \tag{11.7}$$

11.4. Acondicionamiento de la señal generada por el sensor de presión a la salida de la bomba

11.4.1. Cálculo del valor máximo proporcionado por el sensor

La curva de calibración del sensor de presión con el que se va a medir la presión a la salida de la bomba es:

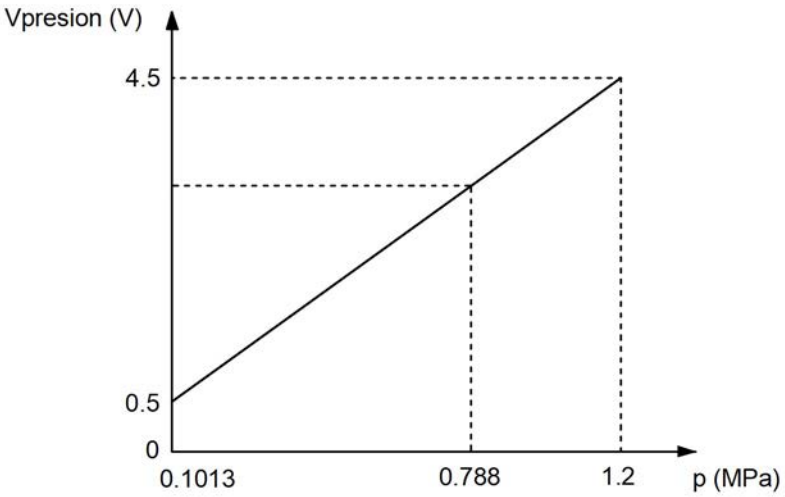


Figura 11.4 – Curva de calibración del sensor de presión

Utilizando los puntos de la Figura 11.4 en la ecuación de la recta que pasa por dos puntos se obtiene la tensión máxima proporcionada por el sensor. Sabiendo que esta ecuación es:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} \tag{11.8}$$

Si:

$$y = V_{presionmax}$$

$$y_1 = 0{,}5$$

$$y_2 = 4{,}5$$

$$x = 0{,}788$$

$$x_1 = 0{,}1013$$

$$x_2 = 1{,}2$$

Se obtiene que:

$$\frac{0{,}788 - 0{,}1013}{1{,}2 - 0{,}1013} = \frac{V_{presionmax} - 0{,}5}{4{,}5 - 0{,}5}$$

$$V_{presionmax} = 3V \tag{11.9}$$

11.4.2. Circuito de acondicionamiento

En el apartado 7.4.3 se propone el siguiente circuito de acondicionamiento para la presión a la salida de la bomba:

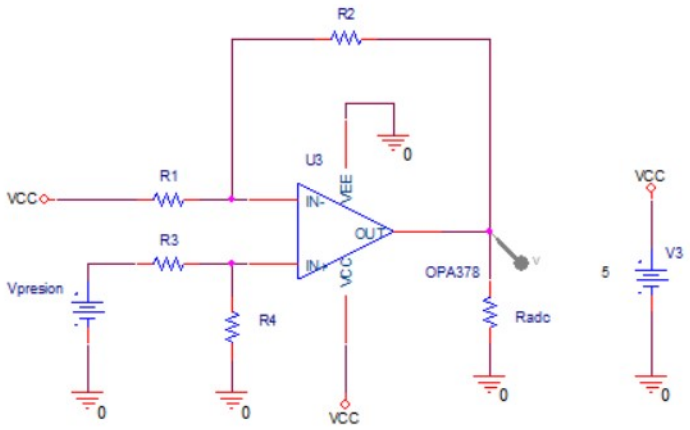


Figura 11.5 – Amplificador diferencial

Por superposición se obtiene que su función de transferencia es:

$$V_o = V_{presion} * \frac{R_4}{R_3 + R_4} * (1 + \frac{R_2}{R_1}) - V_{ref} * \frac{R_2}{R_1} \tag{11.10}$$

Para la calibración se emplearán los puntos inicial y final:

1. Si $V_{presion} = 0{,}5V$, entonces $V_o = 0V$

$$0{,}5 * \frac{R_4}{R_3 + R_4} * (1 + \frac{R_2}{R_1}) - V_{ref} * \frac{R_2}{R_1} = 0$$

Despejando V_{ref} se obtiene que:

$$V_{ref} = 0{,}5 * \frac{R_4}{R_3 + R_4} * (1 + \frac{R_1}{R_2}) \tag{11.11}$$

2. Si $V_{presion} = 3V$, entonces $V_o = 3{,}3V$

$$3 * \frac{R_4}{R_3 + R_4} * (1 + \frac{R_2}{R_1}) - V_{ref} * \frac{R_2}{R_1} = 3{,}3$$

Despejando V_{ref} se obtiene que:

$$V_{ref} = 3 * \frac{R_4}{R_3 + R_4} * (1 + \frac{R_1}{R_2}) - 3{,}3 * \frac{R_1}{R_2} \tag{11.12}$$

Igualando las ecuaciones 11.11 y 11.12 se obtiene:

$$0{,}5 * \frac{R_4}{R_3 + R_4} * (1 + \frac{R_1}{R_2}) = 3 * \frac{R_4}{R_3 + R_4} * (1 + \frac{R_1}{R_2}) - 3{,}3 * \frac{R_1}{R_2}$$

$$3{,}3 * \frac{R_1}{R_2} = 2{,}5 * \frac{R_4}{R_3 + R_4} + 2{,}5 * \frac{R_4}{R_3 + R_4} * \frac{R_1}{R_2}$$

$$(3{,}3 - 2{,}5 * \frac{R_4}{R_3 + R_4}) * \frac{R_1}{R_2} = 2{,}5 * \frac{R_4}{R_3 + R_4} \tag{11.13}$$

Estableciendo que:

$$\frac{R_1}{R_2} = 2 \tag{11.14}$$

Se puede deducir que:

$$(3{,}3 - 2{,}5 * \frac{R_4}{R_3 + R_4}) * 2 = 2{,}5 * \frac{R_4}{R_3 + R_4}$$

$$6{,}6 - 5 * \frac{R_4}{R_3 + R_4} = 2{,}5 * \frac{R_4}{R_3 + R_4}$$

$$\frac{R_4}{R_3 + R_4} = \frac{6{,}6}{7{,}5} = 0{,}88 \tag{11.15}$$

$$R_4 = 0{,}88 * R_3 + 0{,}88 * R_4$$

$$R_4 = 7{,}33 * R_3 \tag{11.16}$$

Sustituyendo los valores de las Ecuaciones 11.14 y 11.15 en la Ecuación 11.11 se obtiene:

$$V_{ref} = 0{,}5 * 0{,}88 * (1 + 2) = 1{,}32V \tag{11.17}$$

Para obtener la tensión de referencia se hará un divisor de tensión con un seguidor. Teniendo en cuenta la ecuación 11.1 se deduce que:

$$R_7 = 0{,}36 * R_6 \tag{11.18}$$

11.5. Acondicionamiento de la señal generada por la célula calibrada

En el Apartado 7.4.5 se propone el siguiente circuito de acondicionamiento:

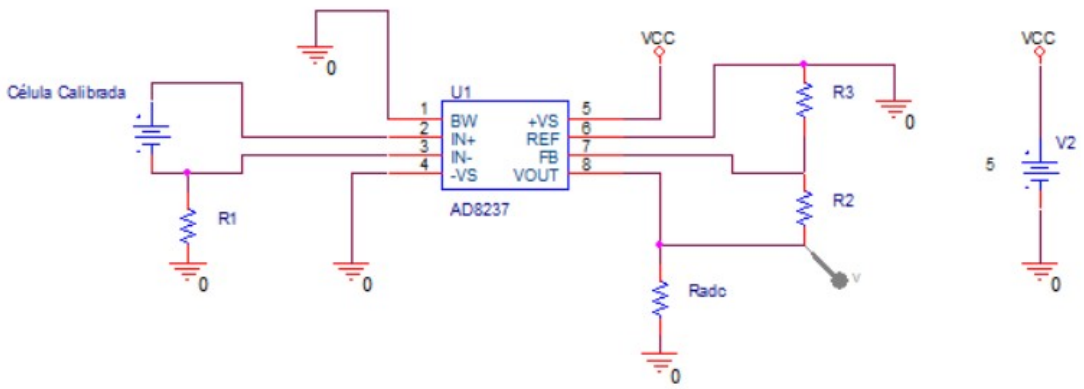


Figura 11.6 – Circuito acondicionamiento de la radiación solar

Sabiendo que la función de transferencia del AD8237 es:

$$V_{OUT} = G(V_{+IN} - V_{-IN}) + V_{REF} \tag{11.19}$$

Siendo:

$$G = 1 + \frac{R_2}{R_3} \tag{11.20}$$

Debido a que en este caso no es necesario ajustar el offset de la señal, se tiene que $V_{REF} = 0V$ y por lo tanto:

$$G = \frac{V_{OUT}}{V_{+IN} - V_{-IN}} = 1 + \frac{R_2}{R3}$$

$$\frac{3{,}3}{150 * 10^{-3}} = 1 + \frac{R_2}{R3}$$

$$\frac{R_2}{R3} = 22 - 1$$

$$R_2 = 21 * R_3 \tag{11.21}$$

12 CÓDIGOS DE PROGRAMACIÓN

12.1. Códigos Arduino

En esta sección se recogen todos los códigos de Arduino empleados durante el desarrollo del proyecto. Las variables que contienen la palabra ”SECRET” deben estar contenidas en el archivo “arduino_secrets.h”. Este archivo puede ser creado con el bloc de notas, definiendo las variables como se indica en la imagen 12.1. Este archivo debe estar almacenado en la carpeta del programa de manera que al abrir el programa se abrirá también el archivo con formato .h.

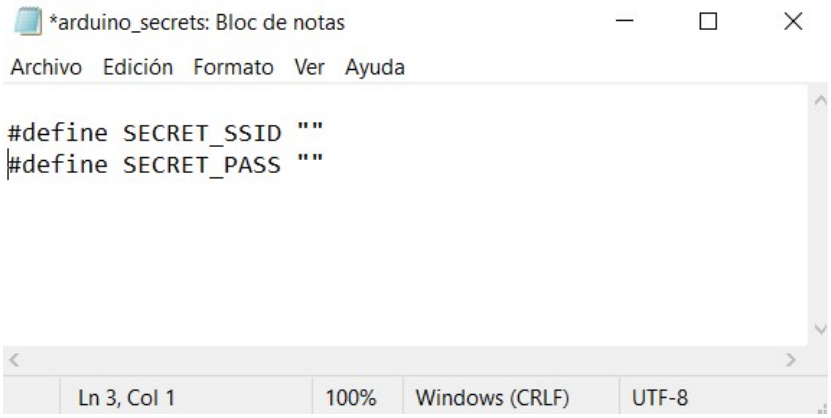


Figura 12.1 – Creación archivo “arduino_secrets.h”

12.1.1. Código final

Código 12.1: Código final

```
   breakatwhitespace

#include <WiFiNINA.h>
#include <ThingsBoard.h>
#include "arduino_secrets.h"

// VARIABLES DE LA RED WiFi

char ssid[] = SECRET_SSID;        // your network SSID (name)
char pass[] = SECRET_PASS;        // your network password (use for WPA, or use as
    key for WEP)
int keyIndex = 0;                 // your network key Index number (needed only for
    WEP)

// VARIABLES DEL SERVIDOR

int status = WL_IDLE_STATUS;
#define TOKEN ""
#define thingsboardServer ""


// INICIALIZACION DE LA LIBRERIA WiFi

WiFiClient client;

// INICIALIZACION DE LA LIBRERIA Thingsboard

//ThingsBoardHttp tb(client, TOKEN, thingsboardServer, thingsboardPort);
  ThingsBoard tb(client);
```

```
// VARIABLES DE USUARIO

byte NumeroMedida=0;

float TensionPlaca, CorrientePlaca, TensionBomba, CorrienteBomba, Presion, Radiacion
    , Caudal;

float TensionPlacaMedia=0;
float CorrientePlacaMedia = 0;
float TensionBombaMedia = 0;
float CorrienteBombaMedia = 0;
float PresionMedia = 0;
float RadiacionMedia = 0;
float CaudalMedio = 0;

float GananciaTension=(130+10)/10;
float GananciaCorrientePlaca=1000/(0.012*51000);
float GananciaCorrienteBomba=1000/(0.015*54000);
float ConversionAD = 3.3/4095;

void setup() {


  delay(10000); //Tiempo para la programacion

  REG_PM_APBCMASK |=  PM_APBCMASK_EVSYS;
  PM->APBCMASK.reg |= PM_APBCMASK_TCC0;


// CONFIGURACION DEL CRISTAL EXTERNO

  SYSCTRL->XOSC32K.reg = SYSCTRL_XOSC32K_ONDEMAND | // Habilita el uso bajo demanda
                         SYSCTRL_XOSC32K_RUNSTDBY | // Habilita el funcionamiento en
                              standby
                         SYSCTRL_XOSC32K_EN32K |    // Enable the crystal oscillator
                              IO pads
                         SYSCTRL_XOSC32K_XTALEN |   // Habilita el oscilador de
                             crital
                         SYSCTRL_XOSC32K_STARTUP(6) | // Establece el tiempo de
                             inicio del cristal
                         SYSCTRL_XOSC32K_ENABLE;    // Habilita el oscilador

// CONFIGURACION DE LA FUENTE DEL RELOJ Y DEL GCLK (gclk.h)

  GCLK->GENDIV.reg =  GCLK_GENDIV_ID(4) | // Selecciona GLCK4
                      GCLK_GENDIV_DIV(4); // Se divide la frecuencia ente (2 ^ (4 +
                          1)) para generar 1.024kHz
  while (GCLK->STATUS.bit.SYNCBUSY);      // Espera a que se sincronice el registro

  GCLK->GENCTRL.reg = GCLK_GENCTRL_ID(4) |        // Selecciona GCLK4
```

```
                        GCLK_GENCTRL_SRC_XOSC32K |  // Selecciona como fuente el
                            cristal de 32.768kHz
                        GCLK_GENCTRL_IDC |          // Mejora el ciclo de trabajo para
                             que sea del 50 % en divisiones impares
                        //GCLK_GENCTRL_RUNSTDBY |   // Habilita el funcionamiento en
                            standby
                        GCLK_GENCTRL_DIVSEL |       // La division de la frecuencia es
                             2 elevado al valor introducido en GENDIV_DIV
                        GCLK_GENCTRL_GENEN;         // Habilita GCLK4
  while (GCLK->STATUS.bit.SYNCBUSY);                // Espera a que se sincronice el
      registro

 GCLK->CLKCTRL.reg = GCLK_CLKCTRL_GEN_GCLK4 |       // Selecciona GCLK4
                GCLK_CLKCTRL_ID_RTC |               // Conecta GCLK4 con el RTC (Real
                    Time Clock)
                GCLK_CLKCTRL_CLKEN;                 // Habilita el GCLK4
  while (GCLK->STATUS.bit.SYNCBUSY);                // Espera la sincronizacion del
      registro

  REG_GCLK_CLKCTRL = GCLK_CLKCTRL_GEN_GCLK4 |       // Selecciona GCLK4
                     GCLK_CLKCTRL_ID_EIC |          // Conecta GCLK4 con el EIC
                     GCLK_CLKCTRL_CLKEN;            // Habilita el GCLK4
  while (GCLK->STATUS.bit.SYNCBUSY);                // Espera la sincronizacion del
      registro

  REG_GCLK_CLKCTRL = GCLK_CLKCTRL_GEN_GCLK4 |       // Selecciona GCLK4
                     GCLK_CLKCTRL_ID_TCC0_TCC1 |    // Conecta GCLK4 con TCC0 y TCC1
                     GCLK_CLKCTRL_CLKEN;            // Habilita el GCLK4
  while (GCLK->STATUS.bit.SYNCBUSY);                // Espera la sincronizacion del
      registro

// CONFIGURACION DEL RTC

  RTC->MODE1.CTRL.bit.ENABLE = 0;  // Inhabilita el RTC (Para configurarlo debe
      estar inactivo)
  while (RTC->MODE1.STATUS.bit.SYNCBUSY);  // Espera la sincronizacion del registro

  RTC->MODE1.CTRL.bit.SWRST = 1;  // Reset por software del RTC
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

  RTC->MODE1.CTRL.reg |= RTC_MODE1_CTRL_PRESCALER_DIV1024 |  // Establece el
      prescaler en 1024 para obtener una frecuencia de 1Hz
                         RTC_MODE1_CTRL_MODE_COUNT16;        // Configura el RTC
                             para funcionar en 16 bits

  RTC->MODE1.PER.reg = RTC_MODE1_PER_PER(9);  // Establece el tiempo de interrupcion
       en 10s: 1Hz/(9 + 1)
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

// CONFIGURACION DE LAS INTERRUPCIONES DEL RTC
```

```
  RTC->MODE1.INTENSET.reg = RTC_MODE1_INTENSET_OVF; // Habilita las interrupciones
      por overflow del RTC

  NVIC_SetPriority(RTC_IRQn, 0);    // Establece la maxima prioridad del NVIC (Nested
       Vector Interrupt Controller) al RTC
  NVIC_EnableIRQ(RTC_IRQn);    // Conecta el RTC al NVIC

// HABILITACION DEL MODO DE BAJO CONSUMO

  SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;  // Cuando se ejecute la funcion __WFI() el
      microcontrolador entrara en bajo consumo
  NVMCTRL->CTRLB.reg |= NVMCTRL_CTRLB_SLEEPPRM_DISABLED;  // Disable auto power
      reduction during sleep - SAMD21 Errata 1.14.2

// HABILITACION DEL RTC

  RTC->MODE1.CTRL.bit.ENABLE = 1;          // Habilita el RTC
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

// CONFIGURACION DE LOS PUERTOS

  PORT->Group[PORTA].PMUX[22 >> 1].reg |= PORT_PMUX_PMUXO_A;      // Conecta el pin
      PA22 (0) al periferico A (EXTINT[6])
  PORT->Group[PORTA].PINCFG[22].reg |= PORT_PINCFG_PMUXEN;        // Habilita la
      multiplexacion de los pines

  pinMode(A1,INPUT);
  pinMode(A2,INPUT);
  pinMode(A3,INPUT);
  pinMode(A4,INPUT);
  pinMode(A5,INPUT);
  pinMode(A6,INPUT);

  // CONFIGURACION DEL ADC

  ADC->CTRLB.bit.RESSEL=0;  //Configura el AD para funcionar a 12 bits.
  ADC->CTRLB.bit.CORREN=1;
  ADC->OFFSETCORR.reg = ADC_OFFSETCORR_OFFSETCORR(60); //Correccion del offset
  ADC->GAINCORR.reg =  ADC_GAINCORR_GAINCORR(2079);    //Correccion de la ganancia


  // CONFIGURACION DEL EIC

  REG_EIC_EVCTRL |= EIC_EVCTRL_EXTINTEO6;          // Habilita lso eventos del pin
      asociado a la interrupcion externa 6
  REG_EIC_CONFIG0 |= EIC_CONFIG_SENSE6_RISE;       // La accion que activa el evento
      es un flanco de subida
  REG_EIC_CTRL |= EIC_CTRL_ENABLE;                 // Habilita el EIC
```

```
  while (EIC->STATUS.bit.SYNCBUSY);                    // Espera la sincronizacion del
      registro


  // CONFIGURACION DEL EVSYS

  REG_EVSYS_USER = EVSYS_USER_CHANNEL(1) |                                    // Asocia
      el evento al canal n=0 (n - 1)
                   EVSYS_USER_USER(EVSYS_ID_USER_TCC0_EV_0);                  //
                       Establece que los eventos activan el timer TCC0

  REG_EVSYS_CHANNEL = EVSYS_CHANNEL_EDGSEL_NO_EVT_OUTPUT |                    // No se
      genera ninguna senal de salida en la deteccion de eventos
                      EVSYS_CHANNEL_PATH_ASYNCHRONOUS |                       //
                          Habilita los eventos asincronos
                      EVSYS_CHANNEL_EVGEN(EVSYS_ID_GEN_EIC_EXTINT_6) |        //
                          Establece como generador del evento a la interrupcion
                          externa 6
                      EVSYS_CHANNEL_CHANNEL(0);                               // Asocia
                          el emisor del evento al canal 0


  // CONFIGURACION TCC0

  REG_TCC0_CTRLA &=~TCC_CTRLA_ENABLE;              // Deshabilita TCC0 para poder
      configurarlo
  while (TCC0->SYNCBUSY.bit.ENABLE);               // Espera la sincronizacion del
      registro


  REG_TCC0_CTRLBCLR |= TCC_CTRLBCLR_DIR;           // Establece que cada evento el
      contador suma 1
  while (TCC0->SYNCBUSY.bit.CTRLB);                // Espera la sincronizacion del
      registro

  REG_TCC0_EVCTRL |= TCC_EVCTRL_TCEI0 |              // Habilita el evento 0 del timer
                     TCC_EVCTRL_EVACT0_COUNT;      // Habilita la cuenta del timer en
                         los eventos

  REG_TCC0_CTRLA |= TCC_CTRLA_RUNSTDBY |           //Habilita el funcionamiento en
      bajo consumo
                    TCC_CTRLA_ENABLE;              // Habilita el TCC0
  while (TCC0->SYNCBUSY.bit.ENABLE);               // Espera la sincronizacion del
      registro

  connectWiFi();
  connectServer();
}

void loop() {
```

```
  for (NumeroMedida=0;NumeroMedida<6;NumeroMedida++){

    __DSB(); // Se espera a que todos los registros esten sincronizados
    __WFI();  // Pone el microcontrolador en bajo consumo
    Medicion();

  }

  CalculoMedia();

  if (WiFi.status() != WL_CONNECTED) {
    connectWiFi();
  }

  if (!tb.connected()){
    connectServer();
  }

  publishMessage();

  if (!client.connected()) {
    client.stop();
  }

}

void connectWiFi() {

  while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
    // failed, retry
    delay(5000);
  }
}

void connectServer() {

    if ( tb.connect(thingsboardServer, TOKEN) ) { //Conexion al servidor
      //Si conecta sale directamente
    } else {
      delay(5000); //Espera 5 segundos antes de volver a intentar
    }
}

void Medicion(){
  for (int i=0;i<=1;i++){
    REG_TCC0_CTRLBSET = TCC_CTRLBSET_CMD_READSYNC;  // Activa la sincronizacion del
        registro que almacena el valor del contador
```

```
  //  __DSB();
    while (TCC0->SYNCBUSY.bit.CTRLB);                // Espera la sincronizacion del
        registro
    while (TCC0->SYNCBUSY.bit.COUNT);                // Espera la sincronizacion del
        registro
  }

  TensionPlaca=analogRead(A1);
  CorrientePlaca=analogRead(A2);
  TensionBomba=analogRead(A3);
  CorrienteBomba=analogRead(A4);
  Presion=analogRead(A5);
  Radiacion=analogRead(A6);

  Caudal=REG_TCC0_COUNT/105.3; // 10(tiempo durante el que cuenta los pulsos)*10.53(
      relacion frecuencia-caudal)
  REG_TCC0_COUNT=TCC_COUNT_RESETVALUE;
  while (TCC0->SYNCBUSY.bit.COUNT);

  TensionPlaca=(TensionPlaca*ConversionAD)*GananciaTension;
  CorrientePlaca=(CorrientePlaca*ConversionAD)*GananciaCorrientePlaca;
  TensionBomba=TensionPlaca-(TensionBomba*ConversionAD)*GananciaTension;
  CorrienteBomba=(CorrienteBomba*ConversionAD)*GananciaCorrienteBomba;
  //Presion=((((Presion*ConversionAD+0.66)/1.32-0.5)/4)*(1.2-0.1013)+0.1013)*10; //
      Presion absoluta.
  Presion=((((Presion*ConversionAD+0.66)/1.32-0.5)/4)*(1.2-0.1013))*10;          //
      Presion manometrica.
  Radiacion=Radiacion/3.33;

  TensionPlacaMedia += TensionPlaca;
  CorrientePlacaMedia += CorrientePlaca;
  TensionBombaMedia += TensionBomba;
  CorrienteBombaMedia += CorrienteBomba;
  PresionMedia += Presion;
  CaudalMedio += Caudal;
  RadiacionMedia += Radiacion;
}

void CalculoMedia(){

  TensionPlacaMedia /= 6;
  CorrientePlacaMedia /= 6;
  TensionBombaMedia /= 6;
  CorrienteBombaMedia /= 6;
  PresionMedia /= 6;
  CaudalMedio /= 6;
  RadiacionMedia /= 6;

}
void publishMessage()
```

```
{
  tb.sendTelemetryFloat("TensionPlaca", TensionPlacaMedia);
  tb.sendTelemetryFloat("CorrientePlaca", CorrientePlacaMedia);
  tb.sendTelemetryFloat("TensionBomba", TensionBombaMedia);
  tb.sendTelemetryFloat("CorrienteBomba", CorrienteBombaMedia);
  tb.sendTelemetryFloat("Presion", PresionMedia);
  tb.sendTelemetryFloat("Radiacion", RadiacionMedia);
  tb.sendTelemetryFloat("Caudal", CaudalMedio);

  TensionPlacaMedia = 0;
  CorrientePlacaMedia = 0;
  TensionBombaMedia = 0;
  CorrienteBombaMedia = 0;
  PresionMedia = 0;
  RadiacionMedia = 0;
  CaudalMedio = 0;
}

void RTC_Handler(void)
{
     RTC->MODE1.INTFLAG.bit.OVF = 1;                                    // Reset the
         overflow interrupt flag
}
```

12.1.2. Calibrado de la célula calibrada

Código empleado para obtener la relación entre tensión a la salida de la célula calibrada y la radiación solar, con la ayuda del medidor de radiación solar de MacSolar.

Código 12.2: Código Arduino para el calibrado de la célula

```
   breakatwhitespace

#include <WiFiNINA.h>
#include "arduino_secrets.h"

// VARIABLES DE LA RED WiFi

char ssid[] = SECRET_SSID;  // your network SSID (name)
char pass[] = SECRET_PASS;  // your network password (use for WPA, or use as key for
     WEP)

// VARIABLES DEL SERVIDOR

int keyIndex = 0;           // your network key Index number (needed only for WEP)
int status = WL_IDLE_STATUS;
char server[] = "docs.google.com";        // name address for Google (using DNS. Se
    podria usar la IP y ocuparia menos memoria)
```

```
String formkey="";          // Identificador del formulario de Google
String NombreCampo = "";  //Nombre del campo en el que se introduce la variable en
    el formulario de Google

// VARIABLES DEL PROGRAMA

unsigned long RadiacionMedia;
String dato;
byte NumeroMedida=0;

// INICIALIZACION DE LA LIBRERIA WiFi

WiFiSSLClient client; // Initialize the Ethernet client library with the IP address
    and port of the server that you want to connect to (port 80 is default for HTTP)


void setup() {


// CONFIGURACION DE LOS PINES Y EL ADC

  pinMode(A6,INPUT);

  ADC->CTRLB.bit.RESSEL=0;  //Configura el AD para funcionar a 12 bits.


// CONFIGURACION DEL CRISTAL EXTERNO

  SYSCTRL->XOSC32K.reg = SYSCTRL_XOSC32K_ONDEMAND | // Habilita el uso bajo demanda
                         SYSCTRL_XOSC32K_RUNSTDBY | // Habilita el funcionamiento en
                             standby
                         SYSCTRL_XOSC32K_EN32K |    // Enable the crystal oscillator
                             IO pads
                         SYSCTRL_XOSC32K_XTALEN |   // Habilita el oscilador de
                            crital
                         SYSCTRL_XOSC32K_STARTUP(6) | // Establece el tiempo de
                            inicio del cristal
                         SYSCTRL_XOSC32K_ENABLE;    // Habilita el oscilador

// CONFIGURACION DE LA FUENTE DEL RELOJ Y DEL GCLK (gclk.h)

  GCLK->GENDIV.reg =  GCLK_GENDIV_ID(4) | // Selecciona GLCK4
                      GCLK_GENDIV_DIV(4); // Se divide la frecuencia ente (2 ^ (4 +
                          1)) para generar 1.024kHz
  while (GCLK->STATUS.bit.SYNCBUSY);      // Espera a que se sincronice el registro

  GCLK->GENCTRL.reg = GCLK_GENCTRL_ID(4) |         // Selecciona GCLK4
                      GCLK_GENCTRL_SRC_XOSC32K |  // Selecciona como fuente el
                          cristal de 32.768kHz
                      GCLK_GENCTRL_IDC |          // Mejora el ciclo de trabajo para
```

```
                        que sea del 50 % en divisiones impares
                    //GCLK_GENCTRL_RUNSTDBY |    // Habilita el funcionamiento en
                        standby
                    GCLK_GENCTRL_DIVSEL |       // La division de la frecuencia es
                         2 elevado al valor introducido en GENDIV_DIV
                    GCLK_GENCTRL_GENEN;         // Habilita GCLK4
  while (GCLK->STATUS.bit.SYNCBUSY);            // Espera a que se sincronice el
      registro
       GCLK->CLKCTRL.reg = GCLK_CLKCTRL_GEN_GCLK4 | // Selecciona GCLK4
                    GCLK_CLKCTRL_ID_RTC |       // Conecta GCLK4 con el RTC (Real
                        Time Clock)
                    GCLK_CLKCTRL_CLKEN;         // Habilita el GCLK4
  while (GCLK->STATUS.bit.SYNCBUSY);            // Espera la sincronizacion del
      registro

// CONFIGURACION DEL RTC

  RTC->MODE1.CTRL.bit.ENABLE = 0;  // Inhabilita el RTC (Para configurarlo debe
      estar inactivo)
  while (RTC->MODE1.STATUS.bit.SYNCBUSY);  // Espera la sincronizacion del registro

  RTC->MODE1.CTRL.bit.SWRST = 1;  // Reset por software del RTC
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

  RTC->MODE1.CTRL.reg |= RTC_MODE1_CTRL_PRESCALER_DIV1024 |  // Establece el
      prescaler en 1024 para obtener una frecuencia de 1Hz
                         RTC_MODE1_CTRL_MODE_COUNT16;       // Configura el RTC
                             para funcionar en 16 bits

  RTC->MODE1.PER.reg = RTC_MODE1_PER_PER(9);  // Establece el tiempo de interrupcion
       en 30s: 1Hz/(29 + 1)
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

// CONFIGURACION DE LAS INTERRUPCIONES DEL RTC

  RTC->MODE1.INTENSET.reg = RTC_MODE1_INTENSET_OVF; // Habilita las interrupciones
      por overflow del RTC

  NVIC_SetPriority(RTC_IRQn, 0);    // Establece la maxima prioridad del NVIC (Nested
       Vector Interrupt Controller) al RTC
  NVIC_EnableIRQ(RTC_IRQn);    // Conecta el RTC al NVIC

// HABILITACION DEL MODO DE BAJO CONSUMO

  SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;  // Cuando se ejecute la funcion __WFI() el
      microcontrolador entrara en bajo consumo
  NVMCTRL->CTRLB.reg |= NVMCTRL_CTRLB_SLEEPPRM_DISABLED;  // Disable auto power
      reduction during sleep — SAMD21 Errata 1.14.2

// HABILITACION DEL RTC
```

```
  RTC->MODE1.CTRL.bit.ENABLE = 1;          // Habilita el RTC
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

}

void loop() {

  if (WiFi.status() != WL_CONNECTED) {
    connectWiFi();
  }
  for (NumeroMedida=0;NumeroMedida<36;NumeroMedida++){

   __DSB(); // Se espera a que todos los registros esten sincronizados
  __WFI();  // Pone el microcontrolador en bajo consumo

    RadiacionMedia += analogRead(A6);
  }

  RadiacionMedia /= 36;
  publishMessage();

  if (!client.connected()) {
    client.stop();
  }

}


void connectWiFi() {

  while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
    delay(5000);
  }

}

void publishMessage() {

  dato="";
  dato=dato + NombreCampo + "=";
  dato=dato + String(RadiacionMedia);

  if (client.connect(server, 443)) {
    // Make a HTTP request:
        client.print("POST /forms/d/e/");
    client.print(formkey);
    client.println("/formResponse? HTTP/1.1");
    client.println("Host: docs.google.com");
//    client.println("User-Agent: MKR1010/1.0");
```

```
    client.println("Content-Type: application/x-www-form-urlencoded");
    client.println("Connection: close");
    client.print("Content-Length: ");
    client.println(dato.length());
    client.println();
    client.print(dato);
    client.println();
  }

}

void RTC_Handler(void)
{
   RTC->MODE1.INTFLAG.bit.OVF = 1;                                    // Reset the
       overflow interrupt flag

}
```

12.1.3. Google Drive

En este programa se realiza, cada 60 s, el envío de una magnitud, medida en el pin A6 del Arduino MKR 1010. Además, durante el tiempo entre muestreos, el Arduino se encuentra en bajo consumo. El envío de la información se realiza mediante solicitudes HTTP.

Código 12.3: Código Arduino Google Drive

```
   breakatwhitespace
#include <WiFiNINA.h>
#include "arduino_secrets.h"

// VARIABLES DE LA RED WiFi

char ssid[] = SECRET_SSID;  // your network SSID (name)
char pass[] = SECRET_PASS;  // your network password (use for WPA, or use as key for
     WEP)

// VARIABLES DEL SERVIDOR

int keyIndex = 0; // your network key Index number (needed only for WEP)
int status = WL_IDLE_STATUS;
char server[] = "docs.google.com";  // name address for Google (using DNS. Se podria
     usar la IP y ocuparia menos memoria)
String formkey = "";  // Identificador del formulario de Google
String NombreCampo = "";  //Nombre del campo en el que se introduce la variable en
    el formulario de Google

// VARIABLES DEL PROGRAMA
```

```
unsigned int variable;
String dato;

// INICIALIZACION DE LA LIBRERIA WiFi

WiFiSSLClient client; // Initialize the Ethernet client library with the IP address
    and port of the server that you want to connect to (port 80 is default for HTTP)


void setup() {

// CONFIGURACION DE LOS PINES Y EL AD

  pinMode(A6,INPUT);

  //ADC->CTRLB.bit.RESSEL=0;  // Configura el AD para funcionar a 12 bits.

// CONFIGURACION DEL CRISTAL EXTERNO

  SYSCTRL->XOSC32K.reg = SYSCTRL_XOSC32K_ONDEMAND | // Habilita el uso bajo demanda
                         SYSCTRL_XOSC32K_RUNSTDBY | // Habilita el funcionamiento en
                              standby
                         SYSCTRL_XOSC32K_EN32K |    // Enable the crystal oscillator
                              IO pads
                         SYSCTRL_XOSC32K_XTALEN |   // Habilita el oscilador de
                             crital
                         SYSCTRL_XOSC32K_STARTUP(6) | // Establece el tiempo de
                             inicio del cristal
                         SYSCTRL_XOSC32K_ENABLE;    // Habilita el oscilador

// CONFIGURACION DE LA FUENTE DEL RELOJ Y DEL GCLK (gclk.h)

  GCLK->GENDIV.reg =  GCLK_GENDIV_ID(4) | // Selecciona GLCK4
                      GCLK_GENDIV_DIV(4); // Se divide la frecuencia ente (2 ^ (4 +
                          1)) para generar 1.024kHz
  while (GCLK->STATUS.bit.SYNCBUSY);      // Espera a que se sincronice el registro

  GCLK->GENCTRL.reg = GCLK_GENCTRL_ID(4) |         // Selecciona GCLK4
                      GCLK_GENCTRL_SRC_XOSC32K |  // Selecciona como fuente el
                          cristal de 32.768kHz
                      GCLK_GENCTRL_IDC |          // Mejora el ciclo de trabajo para
                           que sea del 50% en divisiones impares
                      //GCLK_GENCTRL_RUNSTDBY |   // Habilita el funcionamiento en
                          standby
                      GCLK_GENCTRL_DIVSEL |       // La division de la frecuencia es
                           2 elevado al valor introducido en GENDIV_DIV
                      GCLK_GENCTRL_GENEN;         // Habilita GCLK4
  while (GCLK->STATUS.bit.SYNCBUSY);              // Espera a que se sincronice el
      registro
       GCLK->CLKCTRL.reg = GCLK_CLKCTRL_GEN_GCLK4 | // Selecciona GCLK4
```

```
                        GCLK_CLKCTRL_ID_RTC |        // Conecta GCLK4 con el RTC (Real
                            Time Clock)
                        GCLK_CLKCTRL_CLKEN;          // Habilita el GCLK4
  while (GCLK->STATUS.bit.SYNCBUSY);                 // Espera la sincronizacion del
      registro

// CONFIGURACION DEL RTC

  RTC->MODE1.CTRL.bit.ENABLE = 0;  // Inhabilita el RTC (Para configurarlo debe
      estar inactivo)
  while (RTC->MODE1.STATUS.bit.SYNCBUSY);  // Espera la sincronizacion del registro

  RTC->MODE1.CTRL.bit.SWRST = 1;  // Reset por software del RTC
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

  RTC->MODE1.CTRL.reg |= RTC_MODE1_CTRL_PRESCALER_DIV1024 |  // Establece el
      prescaler en 1024 para obtener una frecuencia de 1Hz
                         RTC_MODE1_CTRL_MODE_COUNT16;        // Configura el RTC
                             para funcionar en 16 bits

  RTC->MODE1.PER.reg = RTC_MODE1_PER_PER(59);  // Establece el tiempo de
      interrupcion en 60s: 1Hz/(59 + 1)
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

// CONFIGURACION DE LAS INTERRUPCIONES DEL RTC

  RTC->MODE1.INTENSET.reg = RTC_MODE1_INTENSET_OVF; // Habilita las interrupciones
      por overflow del RTC

  NVIC_SetPriority(RTC_IRQn, 0);    // Establece la maxima prioridad del NVIC (Nested
       Vector Interrupt Controller) al RTC
  NVIC_EnableIRQ(RTC_IRQn);    // Conecta el RTC al NVIC

// HABILITACION DEL MODO DE BAJO CONSUMO

  SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;  // Cuando se ejecute la funcion __WFI() el
      microcontrolador entrara en bajo consumo
  NVMCTRL->CTRLB.reg |= NVMCTRL_CTRLB_SLEEPPRM_DISABLED;  // Disable auto power
      reduction during sleep - SAMD21 Errata 1.14.2

// HABILITACION DEL RTC

  RTC->MODE1.CTRL.bit.ENABLE = 1;          // Habilita el RTC
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro


}

void loop() {
```

```
  if (WiFi.status() != WL_CONNECTED) {
    connectWiFi();
  }

  __DSB();  // Se espera a que todos los registros esten sincronizados
  __WFI();  // Pone el microcontrolador en bajo consumo

  variable = analogRead(A6);

  publishMessage();

  if (!client.connected()) {
    client.stop();
  }

}


void connectWiFi() {

  while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
    delay(5000);
  }

}

void publishMessage() {

  dato="";
  dato=dato + NombreCampo + "=";
  dato=dato + String(variable);

  if (client.connect(server, 443)) {
    // Make a HTTP request:
        client.print("POST /forms/d/e/");
    client.print(formkey);
    client.println("/formResponse? HTTP/1.1");
    client.println("Host: docs.google.com");
//    client.println("User-Agent: MKR1010/1.0");
    client.println("Content-Type: application/x-www-form-urlencoded");
    client.println("Connection: close");
    client.print("Content-Length: ");
    client.println(dato.length());
    client.println();
    client.print(dato);
    client.println();
  }

}
```

```
void RTC_Handler(void)
{
   RTC->MODE1.INTFLAG.bit.OVF = 1;  // Reset the overflow interrupt flag
}
```

12.1.4. Google Cloud Platform

12.1.4.1. Obtención de la clave pública

Para la obtención de la clave pública que se empleará para la utilización de la plataforma Google Cloud Platform, se utilizara el programa ECCX08JWSPublicKey.ino de la librería “ArduinoECCX08”.

Código 12.4: Obtención clave pública con el ECC508

```
   breakatwhitespace
/*
  ArduinoECCX08 - JWS Public Key

  This sketch can be used to generate a PEM public key for a private key
  generated in an ECC508/ECC608 crypto chip slot.

  If the ECC508/ECC608 is not configured and locked it prompts
  the user to configure and lock the chip with a default TLS
  configuration.

  The user can also select a slot number to use for the private key
  A new private key can also be generated in this slot.

  The circuit:
  - Arduino MKR board equipped with ECC508 or ECC608 chip

  This example code is in the public domain.
*/

#include <ArduinoECCX08.h>
#include <utility/ECCX08JWS.h>
#include <utility/ECCX08DefaultTLSConfig.h>

void setup() {
  Serial.begin(9600);
  while (!Serial);

  if (!ECCX08.begin()) {
    Serial.println("No ECCX08 present!");
    while (1);
  }
```

```
  if (!ECCX08.locked()) {
    String lock = promptAndReadLine("The ECCX08 on your board is not locked, would
        you like to PERMANENTLY configure and lock it now? (y/N)", "N");
    lock.toLowerCase();

    if (!lock.startsWith("y")) {
      Serial.println("Unfortunately you can't proceed without locking it :(");
      while (1);
    }

    if (!ECCX08.writeConfiguration(ECCX08_DEFAULT_TLS_CONFIG)) {
      Serial.println("Writing ECCX08 configuration failed!");
      while (1);
    }

    if (!ECCX08.lock()) {
      Serial.println("Locking ECCX08 configuration failed!");
      while (1);
    }

    Serial.println("ECCX08 locked successfully");
    Serial.println();
  }

  Serial.println("Hi there, in order to generate a PEM public key for your board, we
      'll need the following information ...");
  Serial.println();

  String slot               = promptAndReadLine("What slot would you like to use? (0
       - 4)", "0");
  String generateNewKey     = promptAndReadLine("Would you like to generate a new
      private key? (Y/n)", "Y");

  Serial.println();

  generateNewKey.toLowerCase();

  String publicKeyPem = ECCX08JWS.publicKey(slot.toInt(), generateNewKey.startsWith(
      "y"));

  if (!publicKeyPem || publicKeyPem == "") {
    Serial.println("Error generating public key!");
    while (1);
  }

  Serial.println("Here's your public key PEM, enjoy!");
  Serial.println();
  Serial.println(publicKeyPem);
}
```

```
void loop() {
  // do nothing
}

String promptAndReadLine(const char* prompt, const char* defaultValue) {
  Serial.print(prompt);
  Serial.print(" [");
  Serial.print(defaultValue);
  Serial.print("]: ");

  String s = readLine();

  if (s.length() == 0) {
    s = defaultValue;
  }

  Serial.println(s);

  return s;
}

String readLine() {
  String line;

  while (1) {
    if (Serial.available()) {
      char c = Serial.read();

      if (c == '\r') {
        // ignore
        continue;
      } else if (c == '\n') {
        break;
      }

      line += c;
    }
  }

  return line;
}
```

12.1.4.2. Conexión con el servidor

Para la comunicación con el servidor se utiliza el programa “GCP_IoT_Core_WiFi.ino”de la librería “Arduino Cloud Provider Examples”. En este caso se modifica el programa para leer la temperatura y la humedad mediante el sensor DHT11 y enviar los datos a la nube. El envío de la información se realiza mediante el protocolo MQTT.

Código 12.5: Código Arduino Google Cloud

```
   breakatwhitespace
/*
  GCP (Google Cloud Platform) IoT Core WiFi

  This sketch securely connects to GCP IoT Core using MQTT over WiFi.
  It uses a private key stored in the ATECC508A and a JSON Web Token (JWT) with
  a JSON Web Signature (JWS).

  It publishes a message every 5 seconds to "/devices/{deviceId}/state" topic
  and subscribes to messages on the "/devices/{deviceId}/config" and
  "/devices/{deviceId}/commands/#" topics.

  The circuit:
  - Arduino MKR WiFi 1010 or MKR1000

  This example code is in the public domain.
*/

#include <ArduinoECCX08.h>
#include <utility/ECCX08JWS.h>
#include <ArduinoMqttClient.h>
#include <Arduino_JSON.h>
#include <WiFiNINA.h> // change to #include <WiFi101.h> for MKR1000
#include <DHT.h>

#include "arduino_secrets.h"

/////// Enter your sensitive data in arduino_secrets.h
const char ssid[]        = SECRET_SSID;
const char pass[]        = SECRET_PASS;

const char projectId[]   = SECRET_PROJECT_ID;
const char cloudRegion[] = SECRET_CLOUD_REGION;
const char registryId[]  = SECRET_REGISTRY_ID;
const String deviceId    = SECRET_DEVICE_ID;

const char broker[]      = "mqtt.googleapis.com";

#define DHTPIN 0  // pin digital al que se conecta el sensor
#define DHTTYPE DHT11 //definir tipo de sensor
DHT dht(DHTPIN, DHTTYPE); //Inicializar sensor

WiFiSSLClient wifiSslClient;
MqttClient    mqttClient(wifiSslClient);

unsigned long lastMillis = 0;

void setup() {
```

```
  Serial.begin(9600);
  while (!Serial);

  if (!ECCX08.begin()) {
    Serial.println("No ECCX08 present!");
    while (1);
  }

  // Calculate and set the client id used for MQTT
  String clientId = calculateClientId();

  mqttClient.setId(clientId);

  // Set the message callback, this function is
  // called when the MQTTClient receives a message
  mqttClient.onMessage(onMessageReceived);

  dht.begin(); //Habilita el sensor

}

void loop() {
  if (WiFi.status() != WL_CONNECTED) {
    connectWiFi();
  }

  if (!mqttClient.connected()) {
    // MQTT client is disconnected, connect
    connectMQTT();
  }

  // poll for new MQTT messages and send keep alives
  mqttClient.poll();

  // publish a message roughly every 5 seconds.
  if (millis() - lastMillis > 5000) {
    lastMillis = millis();

    publishMessage();
  }
}

unsigned long getTime() {
  // get the current time from the WiFi module
  return WiFi.getTime();
}

void connectWiFi() {
  Serial.print("Attempting to connect to SSID: ");
  Serial.print(ssid);
```

```
  Serial.print(" ");

  while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
    // failed, retry
    Serial.print(".");
    delay(5000);
  }
  Serial.println();

  Serial.println("You're connected to the network");
  Serial.println();
}

void connectMQTT() {
  Serial.print("Attempting to connect to MQTT broker: ");
  Serial.print(broker);
  Serial.println(" ");

  while (!mqttClient.connected()) {
    // Calculate the JWT and assign it as the password
    String jwt = calculateJWT();

    mqttClient.setUsernamePassword("", jwt);

    if (!mqttClient.connect(broker, 8883)) {
      // failed, retry
      Serial.print(".");
      delay(5000);
    }
  }
  Serial.println();

  Serial.println("You're connected to the MQTT broker");
  Serial.println();

  // subscribe to topics
  mqttClient.subscribe("/devices/" + deviceId + "/config", 1);
  mqttClient.subscribe("/devices/" + deviceId + "/commands/#");
}

String calculateClientId() {
  String clientId;

  // Format:
  //
  //   projects/{project-id}/locations/{cloud-region}/registries/{registry-id}/
      devices/{device-id}
  //

  clientId += "projects/";
```

```
  clientId += projectId;
  clientId += "/locations/";
  clientId += cloudRegion;
  clientId += "/registries/";
  clientId += registryId;
  clientId += "/devices/";
  clientId += deviceId;

  return clientId;
}

String calculateJWT() {
  unsigned long now = getTime();

  // calculate the JWT, based on:
  //   https://cloud.google.com/iot/docs/how-tos/credentials/jwts
  JSONVar jwtHeader;
  JSONVar jwtClaim;

  jwtHeader["alg"] = "ES256";
  jwtHeader["typ"] = "JWT";

  jwtClaim["aud"] = projectId;
  jwtClaim["iat"] = now;
  jwtClaim["exp"] = now + (24L * 60L * 60L); // expires in 24 hours

  return ECCX08JWS.sign(0, JSON.stringify(jwtHeader), JSON.stringify(jwtClaim));
}

void publishMessage() {
  float humedad = dht.readHumidity();
  float temperatura = dht.readTemperature();
  //Comprobamos si ha habido algun error.
  if (isnan(humedad) || isnan(temperatura)){  //Se comprueba que el valor recibido
      sea un numero
    Serial.println("Error obsteniendo los datos del sensor DHT11");
    return;
  }
  String h=String(humedad);
  String t=String(temperatura);
//  String hic=String(indice);
  Serial.println("Publishing message");

  // send message, the Print interface can be used to set the message contents
  //El formato de los datos debe ser: {"variable1": valor1, "variable2": valor2 }");

  mqttClient.beginMessage("/devices/" + deviceId + "/state");
  mqttClient.print("{\"Temperatura\": "+t+", \"Humedad\": "+h+"}");
  //mqttClient.print(millis());
  mqttClient.endMessage();
```

```
}

void onMessageReceived(int messageSize) {
  // we received a message, print out the topic and contents
  Serial.print("Received a message with topic '");
  Serial.print(mqttClient.messageTopic());
  Serial.print("', length ");
  Serial.print(messageSize);
  Serial.println(" bytes:");

  // use the Stream interface to print the contents
  while (mqttClient.available()) {
    Serial.print((char)mqttClient.read());
  }
  Serial.println();

  Serial.println();
}
```

12.1.5. ThingSpeak

En este programa se realiza, cada 20 s, el envío de una magnitud, medida en el pin A6 del Arduino MKR 1010, y el RSSId de la red WiFi. Además, durante el tiempo entre muestreos, el Arduino se encuentra en bajo consumo. El envio de la información se realiza mediante solicitudes HTTP.

Código 12.6: Código Arduino ThingSpeak

```
   breakatwhitespace
#include <WiFiNINA.h>
#include "arduino_secrets.h"

// VARIABLES DE LA RED WiFi

char ssid[] = SECRET_SSID;  // your network SSID (name)
char pass[] = SECRET_PASS;  // your network password (use for WPA, or use as key for
     WEP)

// VARIABLES DEL SERVIDOR
char server[] = "api.thingspeak.com";
String writeAPIKey = "";
int status = WL_IDLE_STATUS;

// INICIALIZACION DE LA LIBRERIA WiFi

WiFiClient client;
```

```
// VARIABLES DEL PROGRAMA

unsigned int variable;


void setup() {

  // CONFIGURACION DE LOS PINES Y EL AD

  pinMode(A6,INPUT);

  //ADC->CTRLB.bit.RESSEL=0;  // Configura el AD para funcionar a 12 bits.


  // CONFIGURACION DEL CRISTAL EXTERNO

  SYSCTRL->XOSC32K.reg = SYSCTRL_XOSC32K_ONDEMAND | // Habilita el uso bajo demanda
                         SYSCTRL_XOSC32K_RUNSTDBY | // Habilita el funcionamiento en
                             standby
                         SYSCTRL_XOSC32K_EN32K |    // Enable the crystal oscillator
                             IO pads
                         SYSCTRL_XOSC32K_XTALEN |   // Habilita el oscilador de
                            crital
                         SYSCTRL_XOSC32K_STARTUP(6) | // Establece el tiempo de
                            inicio del cristal
                         SYSCTRL_XOSC32K_ENABLE;    // Habilita el oscilador

// CONFIGURACION DE LA FUENTE DEL RELOJ Y DEL GCLK (gclk.h)

  GCLK->GENDIV.reg =  GCLK_GENDIV_ID(4) | // Selecciona GLCK4
                      GCLK_GENDIV_DIV(4); // Se divide la frecuencia ente (2 ^ (4 +
                          1)) para generar 1.024kHz
  while (GCLK->STATUS.bit.SYNCBUSY);      // Espera a que se sincronice el registro

  GCLK->GENCTRL.reg = GCLK_GENCTRL_ID(4) |         // Selecciona GCLK4
                      GCLK_GENCTRL_SRC_XOSC32K |  // Selecciona como fuente el
                          cristal de 32.768kHz
                      GCLK_GENCTRL_IDC |          // Mejora el ciclo de trabajo para
                           que sea del 50% en divisiones impares
                      //GCLK_GENCTRL_RUNSTDBY |   // Habilita el funcionamiento en
                          standby
                      GCLK_GENCTRL_DIVSEL |       // La division de la frecuencia es
                           2 elevado al valor introducido en GENDIV_DIV
                      GCLK_GENCTRL_GENEN;         // Habilita GCLK4
  while (GCLK->STATUS.bit.SYNCBUSY);              // Espera a que se sincronice el
      registro
       GCLK->CLKCTRL.reg = GCLK_CLKCTRL_GEN_GCLK4 | // Selecciona GCLK4
                      GCLK_CLKCTRL_ID_RTC |       // Conecta GCLK4 con el RTC (Real
                          Time Clock)
                      GCLK_CLKCTRL_CLKEN;         // Habilita el GCLK4
```

```
  while (GCLK->STATUS.bit.SYNCBUSY);              // Espera la sincronizacion del
      registro

// CONFIGURACION DEL RTC

  RTC->MODE1.CTRL.bit.ENABLE = 0;  // Inhabilita el RTC (Para configurarlo debe
      estar inactivo)
  while (RTC->MODE1.STATUS.bit.SYNCBUSY);  // Espera la sincronizacion del registro

  RTC->MODE1.CTRL.bit.SWRST = 1;  // Reset por software del RTC
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

  RTC->MODE1.CTRL.reg |= RTC_MODE1_CTRL_PRESCALER_DIV1024 |  // Establece el
      prescaler en 1024 para obtener una frecuencia de 1Hz
                         RTC_MODE1_CTRL_MODE_COUNT16;        // Configura el RTC
                             para funcionar en 16 bits

  RTC->MODE1.PER.reg = RTC_MODE1_PER_PER(19);  // Establece el tiempo de
      interrupcion en 20s: 1Hz/(19 + 1)
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

// CONFIGURACION DE LAS INTERRUPCIONES DEL RTC

  RTC->MODE1.INTENSET.reg = RTC_MODE1_INTENSET_OVF; // Habilita las interrupciones
      por overflow del RTC

  NVIC_SetPriority(RTC_IRQn, 0);    // Establece la maxima prioridad del NVIC (Nested
       Vector Interrupt Controller) al RTC
  NVIC_EnableIRQ(RTC_IRQn);    // Conecta el RTC al NVIC

// HABILITACION DEL MODO DE BAJO CONSUMO

  SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;  // Cuando se ejecute la funcion __WFI() el
      microcontrolador entrara en bajo consumo
  NVMCTRL->CTRLB.reg |= NVMCTRL_CTRLB_SLEEPPRM_DISABLED;  // Disable auto power
      reduction during sleep - SAMD21 Errata 1.14.2

// HABILITACION DEL RTC

  RTC->MODE1.CTRL.bit.ENABLE = 1;          // Habilita el RTC
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

  // attempt to connect to Wifi network
  while ( WiFi.status() != WL_CONNECTED) {
    while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
    // failed, retry
      Serial.print(".");
      delay(5000);
    }
    Serial.println();
```

```
    Serial.println("You're connected to the network");
    }
}

void loop() {

  if (WiFi.status() != WL_CONNECTED) {
    connectWiFi();
  }

  __DSB();                                                  // Complete outstanding
     memory operations - not required for SAMD21 ARM Cortex M0+
  __WFI();

  publishMessage();

}

void connectWiFi() {

  while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
    delay(5000);
  }

}

void publishMessage() {
  // read analog pin 0
  int variable = analogRead(A6);
  // read Wi-Fi signal strength (rssi)
  long rssi = WiFi.RSSI();
  // create data string to send to ThingSpeak
  String data = String("field1=" + String(variable, DEC) + "&field2=" + String(rssi,
       DEC));

  // close any connection before sending a new request
  client.stop();

  // POST data to ThingSpeak
  Serial.println("Publishing message");
  Serial.println("\nStarting connection to server...");
  if (client.connect(server, 80)) {
    Serial.println("connected to server");
    client.println("POST /update HTTP/1.1");
    client.println("Host: api.thingspeak.com");
    client.println("Connection: close");
    client.println("User-Agent: ArduinoWiFi/1.1");
    client.println("X-THINGSPEAKAPIKEY: "+writeAPIKey);
    client.println("Content-Type: application/x-www-form-urlencoded");
```

```
    client.print("Content-Length: ");
    client.print(data.length());
    client.print("\n\n");
    client.print(data);

  }
}
  void RTC_Handler(void)
{
     RTC->MODE1.INTFLAG.bit.OVF = 1;       // Reset the overflow interrupt flag

}
```

12.1.6. Thinger.io

Para la prueba de la plataforma se emplea el progrma ArduinoMKR1010.ino de la librería "thinger.io". En este programa se envía a la plataforma el tiempo transcurrido desde el inicio. Además, se puede encender y apagar el LED incorporado en la placa (LED_BUILTIN, conectado al pin 6) desde la plataforma.

Código 12.7: Código Arduino Thinger.io

```
   breakatwhitespace
#include <ThingerWiFiNINA.h>

#define USERNAME ""
#define DEVICE_ID ""
#define DEVICE_CREDENTIAL ""

#define SSID ""
#define SSID_PASSWORD ""

ThingerWiFiNINA thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);

void setup() {
  // configure wifi network
  thing.add_wifi(SSID, SSID_PASSWORD);

  pinMode(LED_BUILTIN, OUTPUT);

  // pin control example (i.e. turning on/off a light, a relay, etc)
  thing["led"] << digitalPin(LED_BUILTIN);

  // resource output example (i.e. reading a sensor value, a variable, etc)
  thing["millis"] >> outputValue(millis());
  delay(60000);
```

```
  // more details at http://docs.thinger.io/arduino/
}

void loop() {
  thing.handle();
}
```

12.1.7. Thingsboard

Para el envio de los datos a la plataforma Thingsboard, ya sea como servidor en la nube o como servidor local, se realiza el envio de datos mediante la librería “ThingsBoard.h”. Esta librería permite la comunicación tanto por HTTP como por MQTT. En ambos casos se realizará por MQTT.

12.1.7.1. Versión demo

Código 12.8: Código Arduino ThingsBoard Demo

```
   breakatwhitespace

#include <DHT.h>
#include <WiFiNINA.h>
#include <ThingsBoard.h>
#include "arduino_secrets.h"

// VARIABLES DE LA RED WiFi

char ssid[] = SECRET_SSID;
char pass[]  = SECRET_PASS;

// VARIABLES DEL SERVIDOR

char thingsboardServer[] = "demo.thingsboard.io";
char TOKEN[] = SECRET_TOKEN;
int status = WL_IDLE_STATUS;

// DHT
#define DHTPIN 0
#define DHTTYPE DHT11

// INICIALIZACION DE LA LIBRERIA WiFi

WiFiClient wifiClient;

//INICIALIZACION DEL SENSOR DHT
```

```
DHT dht(DHTPIN, DHTTYPE);

// INICIALIZACION DE LA LIBRERIA Thingsboard

ThingsBoard tb(wifiClient);

void setup(){

// CONFIGURACION DEL CRISTAL EXTERNO

  SYSCTRL->XOSC32K.reg = SYSCTRL_XOSC32K_ONDEMAND | // Habilita el uso bajo demanda
                         SYSCTRL_XOSC32K_RUNSTDBY | // Habilita el funcionamiento en
                              standby
                         SYSCTRL_XOSC32K_EN32K |    // Enable the crystal oscillator
                              IO pads
                         SYSCTRL_XOSC32K_XTALEN |   // Habilita el oscilador de
                             crital
                         SYSCTRL_XOSC32K_STARTUP(6) | // Establece el tiempo de
                             inicio del cristal
                         SYSCTRL_XOSC32K_ENABLE;    // Habilita el oscilador

// CONFIGURACION DE LA FUENTE DEL RELOJ Y DEL GCLK (gclk.h)

  GCLK->GENDIV.reg =  GCLK_GENDIV_ID(4) | // Selecciona GLCK4
                      GCLK_GENDIV_DIV(4); // Se divide la frecuencia ente (2 ^ (4 +
                          1)) para generar 1.024kHz
  while (GCLK->STATUS.bit.SYNCBUSY);      // Espera a que se sincronice el registro

  GCLK->GENCTRL.reg = GCLK_GENCTRL_ID(4) |         // Selecciona GCLK4
                      GCLK_GENCTRL_SRC_XOSC32K |   // Selecciona como fuente el
                          cristal de 32.768kHz
                      GCLK_GENCTRL_IDC |           // Mejora el ciclo de trabajo para
                           que sea del 50% en divisiones impares
                      //GCLK_GENCTRL_RUNSTDBY |    // Habilita el funcionamiento en
                          standby
                      GCLK_GENCTRL_DIVSEL |        // La division de la frecuencia es
                           2 elevado al valor introducido en GENDIV_DIV
                      GCLK_GENCTRL_GENEN;          // Habilita GCLK4
  while (GCLK->STATUS.bit.SYNCBUSY);               // Espera a que se sincronice el
      registro
       GCLK->CLKCTRL.reg = GCLK_CLKCTRL_GEN_GCLK4 | // Selecciona GCLK4
                      GCLK_CLKCTRL_ID_RTC |        // Conecta GCLK4 con el RTC (Real
                          Time Clock)
                      GCLK_CLKCTRL_CLKEN;          // Habilita el GCLK4
  while (GCLK->STATUS.bit.SYNCBUSY);               // Espera la sincronizacion del
      registro

// CONFIGURACION DEL RTC
```

```
  RTC->MODE1.CTRL.bit.ENABLE = 0;  // Inhabilita el RTC (Para configurarlo debe
      estar inactivo)
  while (RTC->MODE1.STATUS.bit.SYNCBUSY);  // Espera la sincronizacion del registro

  RTC->MODE1.CTRL.bit.SWRST = 1;  // Reset por software del RTC
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

  RTC->MODE1.CTRL.reg |= RTC_MODE1_CTRL_PRESCALER_DIV1024 |  // Establece el
      prescaler en 1024 para obtener una frecuencia de 1Hz
                         RTC_MODE1_CTRL_MODE_COUNT16;        // Configura el RTC
                             para funcionar en 16 bits

  RTC->MODE1.PER.reg = RTC_MODE1_PER_PER(29);  // Establece el tiempo de
      interrupcion en 30s: 1Hz/(29 + 1)
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

// CONFIGURACION DE LAS INTERRUPCIONES DEL RTC

  RTC->MODE1.INTENSET.reg = RTC_MODE1_INTENSET_OVF; // Habilita las interrupciones
      por overflow del RTC

  NVIC_SetPriority(RTC_IRQn, 0);    // Establece la maxima prioridad del NVIC (Nested
       Vector Interrupt Controller) al RTC
  NVIC_EnableIRQ(RTC_IRQn);    // Conecta el RTC al NVIC

// HABILITACION DEL MODO DE BAJO CONSUMO

  SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;  // Cuando se ejecute la funcion __WFI() el
      microcontrolador entrara en bajo consumo
  NVMCTRL->CTRLB.reg |= NVMCTRL_CTRLB_SLEEPPRM_DISABLED;  // Disable auto power
      reduction during sleep - SAMD21 Errata 1.14.2

// HABILITACION DEL RTC

  RTC->MODE1.CTRL.bit.ENABLE = 1;          // Habilita el RTC
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

  dht.begin();
}

void loop(){

  __DSB();  // Se espera a que todos los registros esten sincronizados
  __WFI();  // Pone el microcontrolador en bajo consumo

  if (WiFi.status() != WL_CONNECTED) {
    connectWiFi();
  }

  if ( !tb.connected() ) {
```

```
    connectServer();
  }
  publishMessage();
  tb.loop();
}

void RTC_Handler(void){

  RTC->MODE1.INTFLAG.bit.OVF = 1; // Reset the overflow interrupt flag

}

void connectWiFi() {

  while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
    delay(5000);
  }
}

void publishMessage()
{
  float humidity = dht.readHumidity();  //Lectura de la humedad
  float temperature = dht.readTemperature();  //Lectura de la temperatura

  // Check if any reads failed and exit early (to try again).
  if (isnan(humidity) || isnan(temperature)) {
    return;
  }

  tb.sendTelemetryFloat("temperature", temperature);
  tb.sendTelemetryFloat("humidity", humidity);
}


void connectServer() {

    if ( tb.connect(thingsboardServer, TOKEN) ) { //Conexion al servidor
      //Si conecta sale directamente
    } else {
      delay( 5000 ); //Espera 5 segundos antes de volver a intentar
    }
}
```

12.1.7.2. Servidor local

Código 12.9: Código Arduino Thingsboard en servidor local

```
   breakatwhitespace
#include <DHT.h>
#include <WiFiNINA.h>
#include <ThingsBoard.h>
#include "arduino_secrets.h"

// VARIABLES DE LA RED WiFi

char ssid[] = SECRET_SSID;    // your network SSID (name)
char pass[] = SECRET_PASS;    // your network password (use for WPA, or use as key
    for WEP)
int keyIndex = 0;             // your network key Index number (needed only for WEP)

// VARIABLES DEL SERVIDOR

int status = WL_IDLE_STATUS;
char TOKEN[] = SECRET_TOKEN;
#define thingsboardServer ""
#define thingsboardPort  8080

// DHT
#define DHTPIN 0
#define DHTTYPE DHT11

// INICIALIZACION DE LA LIBRERIA WiFi

WiFiClient client;

//INICIALIZACION DEL SENSOR DHT

DHT dht(DHTPIN, DHTTYPE);

// INICIALIZACION DE LA LIBRERIA Thingsboard

ThingsBoardHttp tb(client, TOKEN, thingsboardServer, thingsboardPort);

void setup() {

// CONFIGURACION DEL CRISTAL EXTERNO

  SYSCTRL->XOSC32K.reg = SYSCTRL_XOSC32K_ONDEMAND | // Habilita el uso bajo demanda
                         SYSCTRL_XOSC32K_RUNSTDBY | // Habilita el funcionamiento en
                              standby
                         SYSCTRL_XOSC32K_EN32K |    // Enable the crystal oscillator
                              IO pads
                         SYSCTRL_XOSC32K_XTALEN |   // Habilita el oscilador de
                             crital
                         SYSCTRL_XOSC32K_STARTUP(6) | // Establece el tiempo de
                             inicio del cristal
                         SYSCTRL_XOSC32K_ENABLE;    // Habilita el oscilador
```

```
// CONFIGURACION DE LA FUENTE DEL RELOJ Y DEL GCLK (gclk.h)

  GCLK->GENDIV.reg =  GCLK_GENDIV_ID(4) | // Selecciona GLCK4
                      GCLK_GENDIV_DIV(4); // Se divide la frecuencia ente (2 ^ (4 +
                          1)) para generar 1.024kHz
  while (GCLK->STATUS.bit.SYNCBUSY);      // Espera a que se sincronice el registro

  GCLK->GENCTRL.reg = GCLK_GENCTRL_ID(4) |         // Selecciona GCLK4
                      GCLK_GENCTRL_SRC_XOSC32K |   // Selecciona como fuente el
                          cristal de 32.768kHz
                      GCLK_GENCTRL_IDC |           // Mejora el ciclo de trabajo para
                           que sea del 50 % en divisiones impares
                      //GCLK_GENCTRL_RUNSTDBY |    // Habilita el funcionamiento en
                          standby
                      GCLK_GENCTRL_DIVSEL |        // La division de la frecuencia es
                           2 elevado al valor introducido en GENDIV_DIV
                      GCLK_GENCTRL_GENEN;          // Habilita GCLK4
  while (GCLK->STATUS.bit.SYNCBUSY);               // Espera a que se sincronice el
      registro
       GCLK->CLKCTRL.reg = GCLK_CLKCTRL_GEN_GCLK4 | // Selecciona GCLK4
                      GCLK_CLKCTRL_ID_RTC |        // Conecta GCLK4 con el RTC (Real
                          Time Clock)
                      GCLK_CLKCTRL_CLKEN;          // Habilita el GCLK4
  while (GCLK->STATUS.bit.SYNCBUSY);               // Espera la sincronizacion del
      registro

// CONFIGURACION DEL RTC

  RTC->MODE1.CTRL.bit.ENABLE = 0;  // Inhabilita el RTC (Para configurarlo debe
      estar inactivo)
  while (RTC->MODE1.STATUS.bit.SYNCBUSY);  // Espera la sincronizacion del registro

  RTC->MODE1.CTRL.bit.SWRST = 1;  // Reset por software del RTC
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

  RTC->MODE1.CTRL.reg |= RTC_MODE1_CTRL_PRESCALER_DIV1024 |  // Establece el
      prescaler en 1024 para obtener una frecuencia de 1Hz
                         RTC_MODE1_CTRL_MODE_COUNT16;        // Configura el RTC
                             para funcionar en 16 bits

  RTC->MODE1.PER.reg = RTC_MODE1_PER_PER(29);  // Establece el tiempo de
      interrupcion en 30s: 1Hz/(29 + 1)
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

// CONFIGURACION DE LAS INTERRUPCIONES DEL RTC

  RTC->MODE1.INTENSET.reg = RTC_MODE1_INTENSET_OVF; // Habilita las interrupciones
      por overflow del RTC
```

```
  NVIC_SetPriority(RTC_IRQn, 0);    // Establece la maxima prioridad del NVIC (Nested
      Vector Interrupt Controller) al RTC
  NVIC_EnableIRQ(RTC_IRQn);    // Conecta el RTC al NVIC

// HABILITACION DEL MODO DE BAJO CONSUMO

  SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;  // Cuando se ejecute la funcion __WFI() el
      microcontrolador entrara en bajo consumo
  NVMCTRL->CTRLB.reg |= NVMCTRL_CTRLB_SLEEPPRM_DISABLED;  // Disable auto power
      reduction during sleep - SAMD21 Errata 1.14.2

// HABILITACION DEL RTC

  RTC->MODE1.CTRL.bit.ENABLE = 1;          // Habilita el RTC
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

  dht.begin();
}

void loop() {

   __DSB(); // Se espera a que todos los registros esten sincronizados
  __WFI();  // Pone el microcontrolador en bajo consumo

  if (WiFi.status() != WL_CONNECTED) {
    connectWiFi();
  }

  publishMessage();

  if (!client.connected()) {
    client.stop();
  }
}

void connectWiFi() {

  while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
    // failed, retry
    delay(5000);
  }
}

void publishMessage()
{

  float humidity = dht.readHumidity();  //Lectura de la humedad
  float temperature = dht.readTemperature();  //Lectura de la temperatura

  // Check if any reads failed and exit early (to try again).
```

```
  if (isnan(humidity) || isnan(temperature)) {
    return;
  }

  tb.sendTelemetryFloat("temperature", temperature);
  tb.sendTelemetryFloat("humidity", humidity);
}

void RTC_Handler(void)
{
     RTC->MODE1.INTFLAG.bit.OVF = 1;                                    // Reset the
         overflow interrupt flag

}
```

12.1.8. Ubidots

Para el envio de datos a Ubidots se emplea la librería “PubSubClient.h”. Esta librería permite la comunicación mediante MQTT con el servidor.

Código 12.10: Código Arduino Ubidots

```
   breakatwhitespace
/****************************************
 * Include Libraries
 ****************************************/
#include <WiFiNINA.h>
#include <PubSubClient.h>
#include <DHT.h>

#define ssid "" // Put your WifiSSID here
#define pass "" // Put your wifi password here
#define TOKEN "A1E-" // Put your Ubidots' TOKEN
#define MQTT_CLIENT_NAME "" // MQTT client Name, please enter your own 8-12
    alphanumeric character ASCII string;
                                          //Puede ser el nombre que quiera el
                                              ususario

/****************************************
 * Define Constants
 ****************************************/
#define VARIABLE_LABEL1 "Temperatura" // Assing the variable label
#define VARIABLE_LABEL2 "Humedad"
#define DEVICE_LABEL "Arduino_MKR" // Assig the device label

#define DHTPIN 0 // pin digital
#define DHTTYPE DHT11 //definir tipo de sensor
DHT dht(DHTPIN, DHTTYPE); //Inicializar sensor
```

```
char mqttBroker[]  = "things.ubidots.com";
char payload[100];
char topic[150];
// Space to store values to send
char str_sensor1[10];
char str_sensor2[10];

/****************************************
 * Auxiliar Functions
 ****************************************/
WiFiClient ubidots;
PubSubClient client(ubidots);

void callback(char* topic, byte* payload, unsigned int length) {
  char p[length + 1];
  memcpy(p, payload, length);
  p[length] = NULL;
  String message(p);
  Serial.write(payload, length);
  Serial.println(topic);
}

/****************************************
 * Main Functions
 ****************************************/
void setup() {

  // CONFIGURACION DEL CRISTAL EXTERNO

  SYSCTRL->XOSC32K.reg = SYSCTRL_XOSC32K_ONDEMAND | // Habilita el uso bajo demanda
                         SYSCTRL_XOSC32K_RUNSTDBY | // Habilita el funcionamiento en
                              standby
                         SYSCTRL_XOSC32K_EN32K |    // Enable the crystal oscillator
                              IO pads
                         SYSCTRL_XOSC32K_XTALEN |   // Habilita el oscilador de
                             crital
                         SYSCTRL_XOSC32K_STARTUP(6) | // Establece el tiempo de
                             inicio del cristal
                         SYSCTRL_XOSC32K_ENABLE;    // Habilita el oscilador

// CONFIGURACION DE LA FUENTE DEL RELOJ Y DEL GCLK (gclk.h)

  GCLK->GENDIV.reg =  GCLK_GENDIV_ID(4) | // Selecciona GLCK4
                      GCLK_GENDIV_DIV(4); // Se divide la frecuencia ente (2 ^ (4 +
                          1)) para generar 1.024kHz
  while (GCLK->STATUS.bit.SYNCBUSY);      // Espera a que se sincronice el registro

  GCLK->GENCTRL.reg = GCLK_GENCTRL_ID(4) |        // Selecciona GCLK4
                      GCLK_GENCTRL_SRC_XOSC32K |  // Selecciona como fuente el
```

```
                        cristal de 32.768kHz
                    GCLK_GENCTRL_IDC |          // Mejora el ciclo de trabajo para
                         que sea del 50 % en divisiones impares
                    //GCLK_GENCTRL_RUNSTDBY |    // Habilita el funcionamiento en
                        standby
                    GCLK_GENCTRL_DIVSEL |       // La division de la frecuencia es
                         2 elevado al valor introducido en GENDIV_DIV
                    GCLK_GENCTRL_GENEN;         // Habilita GCLK4
  while (GCLK->STATUS.bit.SYNCBUSY);            // Espera a que se sincronice el
      registro
       GCLK->CLKCTRL.reg = GCLK_CLKCTRL_GEN_GCLK4 | // Selecciona GCLK4
                    GCLK_CLKCTRL_ID_RTC |       // Conecta GCLK4 con el RTC (Real
                        Time Clock)
                    GCLK_CLKCTRL_CLKEN;         // Habilita el GCLK4
  while (GCLK->STATUS.bit.SYNCBUSY);            // Espera la sincronizacion del
      registro

// CONFIGURACION DEL RTC

  RTC->MODE1.CTRL.bit.ENABLE = 0;  // Inhabilita el RTC (Para configurarlo debe
      estar inactivo)
  while (RTC->MODE1.STATUS.bit.SYNCBUSY);  // Espera la sincronizacion del registro

  RTC->MODE1.CTRL.bit.SWRST = 1;  // Reset por software del RTC
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

  RTC->MODE1.CTRL.reg |= RTC_MODE1_CTRL_PRESCALER_DIV1024 |  // Establece el
      prescaler en 1024 para obtener una frecuencia de 1Hz
                         RTC_MODE1_CTRL_MODE_COUNT16;        // Configura el RTC
                             para funcionar en 16 bits

  RTC->MODE1.PER.reg = RTC_MODE1_PER_PER(29);  // Establece el tiempo de
      interrupcion en 30s: 1Hz/(29 + 1)
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

// CONFIGURACION DE LAS INTERRUPCIONES DEL RTC

  RTC->MODE1.INTENSET.reg = RTC_MODE1_INTENSET_OVF; // Habilita las interrupciones
      por overflow del RTC

  NVIC_SetPriority(RTC_IRQn, 0);   // Establece la maxima prioridad del NVIC (Nested
       Vector Interrupt Controller) al RTC
  NVIC_EnableIRQ(RTC_IRQn);   // Conecta el RTC al NVIC

// HABILITACION DEL MODO DE BAJO CONSUMO

  SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;  // Cuando se ejecute la funcion __WFI() el
      microcontrolador entrara en bajo consumo
  NVMCTRL->CTRLB.reg |= NVMCTRL_CTRLB_SLEEPPRM_DISABLED;  // Disable auto power
      reduction during sleep — SAMD21 Errata 1.14.2
```

```
// HABILITACION DEL RTC

  RTC->MODE1.CTRL.bit.ENABLE = 1;          // Habilita el RTC
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

  dht.begin();


  connectWiFi();
  client.setServer(mqttBroker, 1883);
  client.setCallback(callback);

}

void loop() {

  if (WiFi.status() != WL_CONNECTED) {
    connectWiFi();
  }

  if (!client.connected()) {
   while (!client.connect(MQTT_CLIENT_NAME, TOKEN, "")) {
     delay(500);
   }
 }

  __DSB();  // Se espera a que todos los registros esten sincronizados
  __WFI();  // Pone el microcontrolador en bajo consumo

  sprintf(topic, "%s%s", "/v1.6/devices/", DEVICE_LABEL);
  sprintf(payload, "%s", ""); // Cleans the payload
  sprintf(payload, "{\"%s\":", VARIABLE_LABEL1); // Adds the variable label
  sprintf(payload, "%s", ""); // Cleans the payload
  sprintf(payload, "{\"%s\":", VARIABLE_LABEL2); // Adds the variable label

 float temperatura = dht.readTemperature(); //Lectura de la temperatura
 float humedad = dht.readHumidity();        //LEctura de la humedad

 //El formato de los datos debe ser:{"variable1": valor1, "variable2": valor2}");

 String payload = "{\"temperatura\":";
 payload += temperatura;
 payload += ",\"humedad\":";
 payload += humedad;
 payload += "}";

 client.publish(topic, (char*) payload.c_str()); //Envio de los datos

 /*Se puede descomentar el if siguiente y comentar la linea anterior
```

```
  para saber si el dato se escribe correctamente (en ese caso, no se
  puede poner el Arduino en bajo consumo) */

// if (client.publish(topic, (char*) payload.c_str())) { //Envio de los datos y
    comprobacion
//   Serial.println("Publish ok");
// } else {
//   Serial.println("Publish failed");
// }

  client.loop();
}

void connectWiFi() {

  while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
    delay(5000);
  }
}

void RTC_Handler(void)
{
   RTC->MODE1.INTFLAG.bit.OVF = 1;  // Reset the overflow interrupt flag
}
```

12.1.9. IBM

Código 12.11: Código Arduino IBM

```
   breakatwhitespace
/**
 * Helloworld style, connect an ESP32 to IBM's Watson IoT Platform
 *
 * Author: Anthony Elder
 * License: Apache License v2
 */
#include <WiFiNINA.h>
#include <PubSubClient.h>
#include <DHT.h>

// VARIABLES DE LA RED WiFi

const char* ssid = "";
const char* pass = "";

// VARIABLES DEL SERVIDOR
```

```
#define ORG "" // your organization or "quickstart"
#define DEVICE_TYPE "" // use this default for quickstart or customize to your
    registered device type
#define DEVICE_ID "" // use this default for quickstart or customize to your
    registered device id
#define TOKEN "" // your device token or not used with "quickstart"
//-------- Customise the above values --------

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char topic[] = "iot-2/evt/status/fmt/json";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

// VARIABLES DEL PROGRAMA

#define DHTPIN 0 // pin digital
#define DHTTYPE DHT11 //definir tipo de sensor
DHT dht(DHTPIN, DHTTYPE); //Inicializar sensor

// INICIALIZACION DE LA LIBRERIA WiFi y PubSubClient

WiFiClient wifiClient;
PubSubClient client(server, 1883, wifiClient);

void setup() {

 // CONFIGURACION DEL CRISTAL EXTERNO

  SYSCTRL->XOSC32K.reg = SYSCTRL_XOSC32K_ONDEMAND | // Habilita el uso bajo demanda
                         SYSCTRL_XOSC32K_RUNSTDBY | // Habilita el funcionamiento en
                             standby
                         SYSCTRL_XOSC32K_EN32K |    // Enable the crystal oscillator
                             IO pads
                         SYSCTRL_XOSC32K_XTALEN |   // Habilita el oscilador de
                            crital
                         SYSCTRL_XOSC32K_STARTUP(6) | // Establece el tiempo de
                            inicio del cristal
                         SYSCTRL_XOSC32K_ENABLE;    // Habilita el oscilador

// CONFIGURACION DE LA FUENTE DEL RELOJ Y DEL GCLK (gclk.h)

  GCLK->GENDIV.reg =  GCLK_GENDIV_ID(4) | // Selecciona GLCK4
                      GCLK_GENDIV_DIV(4); // Se divide la frecuencia ente (2 ^ (4 +
                          1)) para generar 1.024kHz
  while (GCLK->STATUS.bit.SYNCBUSY);      // Espera a que se sincronice el registro

  GCLK->GENCTRL.reg = GCLK_GENCTRL_ID(4) |        // Selecciona GCLK4
                      GCLK_GENCTRL_SRC_XOSC32K |  // Selecciona como fuente el
                          cristal de 32.768kHz
```

```
                    GCLK_GENCTRL_IDC |          // Mejora el ciclo de trabajo para
                        que sea del 50 % en divisiones impares
                    //GCLK_GENCTRL_RUNSTDBY |    // Habilita el funcionamiento en
                       standby
                    GCLK_GENCTRL_DIVSEL |        // La division de la frecuencia es
                        2 elevado al valor introducido en GENDIV_DIV
                    GCLK_GENCTRL_GENEN;          // Habilita GCLK4
  while (GCLK->STATUS.bit.SYNCBUSY);             // Espera a que se sincronice el
      registro
       GCLK->CLKCTRL.reg = GCLK_CLKCTRL_GEN_GCLK4 | // Selecciona GCLK4
                    GCLK_CLKCTRL_ID_RTC |        // Conecta GCLK4 con el RTC (Real
                       Time Clock)
                    GCLK_CLKCTRL_CLKEN;          // Habilita el GCLK4
  while (GCLK->STATUS.bit.SYNCBUSY);             // Espera la sincronizacion del
      registro

// CONFIGURACION DEL RTC

  RTC->MODE1.CTRL.bit.ENABLE = 0;  // Inhabilita el RTC (Para configurarlo debe
      estar inactivo)
  while (RTC->MODE1.STATUS.bit.SYNCBUSY);  // Espera la sincronizacion del registro

  RTC->MODE1.CTRL.bit.SWRST = 1;  // Reset por software del RTC
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

  RTC->MODE1.CTRL.reg |= RTC_MODE1_CTRL_PRESCALER_DIV1024 |  // Establece el
      prescaler en 1024 para obtener una frecuencia de 1Hz
                         RTC_MODE1_CTRL_MODE_COUNT16;        // Configura el RTC
                             para funcionar en 16 bits

  RTC->MODE1.PER.reg = RTC_MODE1_PER_PER(4);  // Establece el tiempo de interrupcion
       en 60s: 1Hz/(59 + 1)
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

// CONFIGURACION DE LAS INTERRUPCIONES DEL RTC

  RTC->MODE1.INTENSET.reg = RTC_MODE1_INTENSET_OVF; // Habilita las interrupciones
      por overflow del RTC

  NVIC_SetPriority(RTC_IRQn, 0);   // Establece la maxima prioridad del NVIC (Nested
       Vector Interrupt Controller) al RTC
  NVIC_EnableIRQ(RTC_IRQn);   // Conecta el RTC al NVIC

// HABILITACION DEL MODO DE BAJO CONSUMO

  SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;  // Cuando se ejecute la funcion __WFI() el
      microcontrolador entrara en bajo consumo
  NVMCTRL->CTRLB.reg |= NVMCTRL_CTRLB_SLEEPPRM_DISABLED;  // Disable auto power
      reduction during sleep - SAMD21 Errata 1.14.2
```

```
// HABILITACION DEL RTC

  RTC->MODE1.CTRL.bit.ENABLE = 1;          // Habilita el RTC
  while (RTC->MODE1.STATUS.bit.SYNCBUSY); // Espera la sincronizacion del registro

 dht.begin();
}

void loop() {

  if (WiFi.status() != WL_CONNECTED) {
    connectWiFi();
  }

 if (!client.connected()) {
   while (!client.connect(clientId, authMethod, token)) {
     delay(500);
   }
 }

  __DSB();  // Se espera a que todos los registros esten sincronizados
  __WFI();  // Pone el microcontrolador en bajo consumo

 float temperatura = dht.readTemperature(); //Lectura de la temperatura
 float humedad = dht.readHumidity();        //LEctura de la humedad

 //El formato de los datos debe ser:{"variable1": valor1, "variable2": valor2}");

 String payload = "{ \"d\" : {\"temperatura\":";
 payload += temperatura;
 payload += ",\"humedad\":";
 payload += humedad;
 payload += "}}";

 client.publish(topic, (char*) payload.c_str()); //Envio de los datos

 /*Se puede descomentar el if siguiente y comentar la linea anterior
  para saber si el dato se escribe correctamente (en ese caso, no se
  puede poner el Arduino en bajo consumo) */

// if (client.publish(topic, (char*) payload.c_str())) { //Envio de los datos y
    comprobacion
//   Serial.println("Publish ok");
// } else {
//   Serial.println("Publish failed");
// }
}

void connectWiFi() {
```

```
  while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
    delay(5000);
  }
}
void RTC_Handler(void)
{
   RTC->MODE1.INTFLAG.bit.OVF = 1;  // Reset the overflow interrupt flag
}
```

12.1.10. Cayenne

Para la utilización de esta plataforma, se emplea la librería "CayenneMQTT". Sin embargo, esta librería no tiene soporte para el Arduino MKR 1010. Por lo tanto, es necesario modificar la librería para poder ser utilizada por el MKR 1010. La librería se puede encontrar en el siguiente enlace: https://github.com/martinnp12/TFG/blob/master/CayenneMQTT.zip?raw=true.

Este programa se basa en el programa contenido en la siguiente ubicación $Ejemplos \rightarrow CayenneMQTT \rightarrow Connections \rightarrow MKR1010$.

Código 12.12: Código Arduino Cayenne myDevices

```
   breakatwhitespace

//#define CAYENNE_DEBUG       // Uncomment to show debug messages
#define CAYENNE_PRINT Serial  // Comment this out to disable prints and save space
#include <CayenneMQTTMKR1010.h>
#include "arduino_secrets.h"

// WiFi network info.
char ssid[] = SECRET_SSID;
char wifiPassword[] = SECRET_PASS;

// Cayenne authentication info. This should be obtained from the Cayenne Dashboard.
char username[] = SECRET_USERNAME;
char password[] = SECRET_PASSWORD;
char clientID[] = SECRET_CLIENT_ID;

void setup() {
  Serial.begin(9600);
  Cayenne.begin(username, password, clientID, ssid, wifiPassword);
}

void loop() {
  Cayenne.loop();
}

// Default function for sending sensor data at intervals to Cayenne.
```

```
// You can also use functions for specific channels, e.g CAYENNE_OUT(1) for sending
    channel 1 data.
CAYENNE_OUT_DEFAULT()
{
  // Write data to Cayenne here. This example just sends the current uptime in
      milliseconds on virtual channel 0.
  Cayenne.virtualWrite(0, millis());
  // Some examples of other functions you can use to send data.
  Cayenne.celsiusWrite(1, random(0,40));
  Cayenne.luxWrite(2, random(0,1000));
  Cayenne.virtualWrite(3, 50, TYPE_PROXIMITY, UNIT_CENTIMETER);
}

// Default function for processing actuator commands from the Cayenne Dashboard.
// You can also use functions for specific channels, e.g CAYENNE_IN(1) for channel 1
     commands.
CAYENNE_IN_DEFAULT()
{
  CAYENNE_LOG("Channel %u, value %s", request.channel, getValue.asString());
  //Process message here. If there is an error set an error message using getValue.
      setError(), e.g getValue.setError("Error message");
}
```

12.2. Códigos Matlab

12.2.1. Función para el importado de los datos

Este código es una modificación de la función generada por la herramienta “Importado”de Matlab.

Código 12.13: Modificación del formato de los datos

```
function Datos = importfile(filename)
%IMPORTFILE Import numeric data from a text file as a matrix.
%   DATOS = IMPORTFILE(FILENAME) Reads data from text file FILENAME for the
%   default selection.
%
%   DATOS = IMPORTFILE(FILENAME, STARTROW, ENDROW) Reads data from rows
%   STARTROW through ENDROW of text file FILENAME.
%
% Example:
%   Datos = importfile('Datos.csv', 1, 1737);
%
%    See also TEXTSCAN.

% Auto-generated by MATLAB on 2019/09/02 14:30:52
```

```
%% Initialize variables.
delimiter = ',';
if nargin<=2
startRow = 2;
endRow = inf;
end

%% Read columns of data as text:
% For more information, see the TEXTSCAN documentation.
formatSpec = '%*s%*s%s%*s%*s%*s%*s%s%s%[^\n\r]';

%% Open the text file.
fileID = fopen(filename,'r');

%% Read columns of data according to the format.
% This call is based on the structure of the file used to generate this
% code. If an error occurs for a different file, try regenerating the code
% from the Import Tool.
dataArray = textscan(fileID, formatSpec, ...
    endRow(1)-startRow(1)+1, 'Delimiter', delimiter, ...
    'TextType', 'string', 'HeaderLines', startRow(1)-1, ...
    'ReturnOnError', false, 'EndOfLine', '\r\n');
for block=2:length(startRow)
frewind(fileID);
dataArrayBlock = textscan(fileID, formatSpec, ...
    endRow(block)-startRow(block)+1,'Delimiter', ...
    delimiter, 'TextType','string', 'HeaderLines', ...
    startRow(block)-1, 'ReturnOnError', false, 'EndOfLine', '\r\n');
for col=1:length(dataArray)
dataArray{col} = [dataArray{col};dataArrayBlock{col}];
end
end

%% Close the text file.
fclose(fileID);

%% Convert the contents of columns containing numeric text to numbers.
% Replace non-numeric text with NaN.
raw = repmat({''},length(dataArray{1}),length(dataArray)-1);
for col=1:length(dataArray)-1
raw(1:length(dataArray{col}),col) = mat2cell(dataArray{col}, ...
    ones(length(dataArray{col}), 1));
end
numericData = NaN(size(dataArray{1},1),size(dataArray,2));

% Converts text in the input cell array to numbers. Replaced non-numeric
% text with NaN.
rawData = dataArray{2};
for row=1:size(rawData, 1)
% Create a regular expression to detect and remove non-numeric prefixes and
```

```
% suffixes.

try
result = regexp(rawData(row), regexstr, 'names');
numbers = result.numbers;

% Detected commas in non-thousand locations.
invalidThousandsSeparator = false;
if numbers.contains(',')

if isempty(regexp(numbers, thousandsRegExp, 'once'))
numbers = NaN;
invalidThousandsSeparator = true;
end
end
% Convert numeric text to numbers.
if ~invalidThousandsSeparator
numbers = textscan(char(strrep(numbers, ',', '')), '%f');
numericData(row, 2) = numbers{1};
raw{row, 2} = numbers{1};
end
catch
raw{row, 2} = rawData{row};
end
end


%% Split data into numeric and string columns.
rawNumericColumns = raw(:, 2);
rawStringColumns = string(raw(:, [1,3]));


%% Replace non-numeric cells with NaN
R = cellfun(@(x) ~isnumeric(x) && ~islogical(x),rawNumericColumns);
rawNumericColumns(R) = {NaN}; % Replace non-numeric cells

%% Make sure any text containing <undefined> is properly converted to ...
%%an <undefined> categorical
idx = (rawStringColumns(:, 1) == "<undefined>");
rawStringColumns(idx, 1) = "";

%% Create output variable
Datos = table;
Datos.key = categorical(rawStringColumns(:, 1));
Datos.dbl_v = cell2mat(rawNumericColumns(:, 1));
Datos.created_at = rawStringColumns(:, 2);
```

12.2.2. Script para la modificación del formato de los datos

Código 12.14: Modificación del formato de los datos

```
clear
numero=0;
M=Importar('Datos5.csv');
[filas, columnas]=size(M);
Datos(:,1)=M(1:7:filas,3);
Datos(:,2:8)=array2table(zeros(filas/7,7));
for i=1:7:filas-1
    numero=numero+1;
    for j=i:i+6
       if M.key(j) == '"TensionPlaca"'
           Datos(numero,2)=M(j,2);
       end
       if M.key(j) == '"CorrientePlaca"'
           Datos(numero,3)=M(j,2);
       end
       if M.key(j) == '"TensionBomba"'
           Datos(numero,4)=M(j,2);
       end
       if M.key(j) == '"CorrienteBomba"'
           Datos(numero,5)=M(j,2);
       end
       if M.key(j) == '"Presion"'
           Datos(numero,6)=M(j,2);
       end
       if M.key(j) == '"Radiacion"'
           Datos(numero,7)=M(j,2);
       end
       if M.key(j) == '"Caudal"'
           Datos(numero,8)=M(j,2);
       end
    end
end

header={'Fecha','TensionPlaca','CorrientePlaca','TensionBomba', ...
    'CorrienteBomba', 'Presion','Radiacion','Caudal'};
Datos.Properties.VariableNames = header;
```

13 CONFIGURACIÓN Y USO DE LAS PLATAFORMAS IOT

En este anexo se desarrollara una guía de configuración y uso de las plataformas IoT, estudiadas en el Apartado 7.5, junto con el Arduino MKR 1010.

13.1. Google Drive

Para subir datos a Google drive desde Arduino hay que crear un formulario de Google. Esto se puede hacer yendo a la página web https://drive.google.com/drive/my-drive y, entonces, a $Nuevo \rightarrow Mas \rightarrow Formularios\ \ de\ \ Google$.

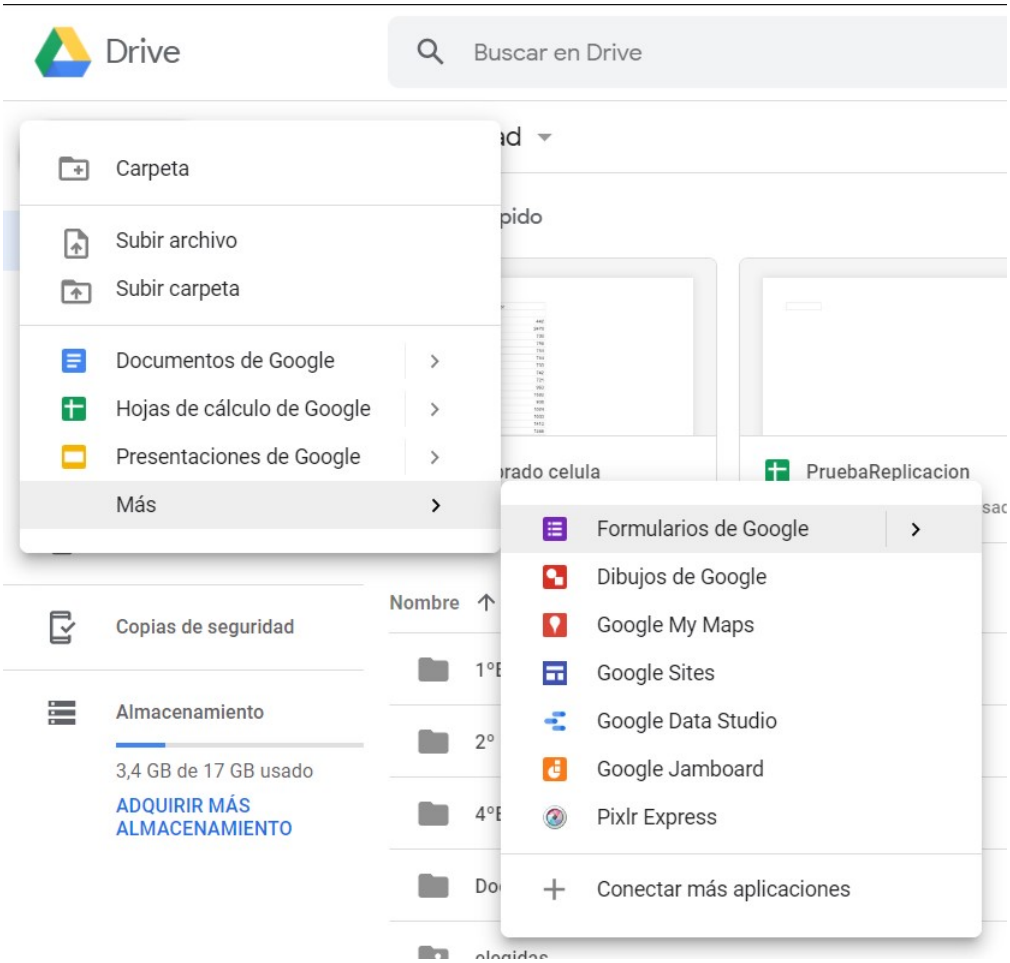


Figura 13.1 – Creación de un formulario en Google Drive

De este modo, se abre la siguiente página:

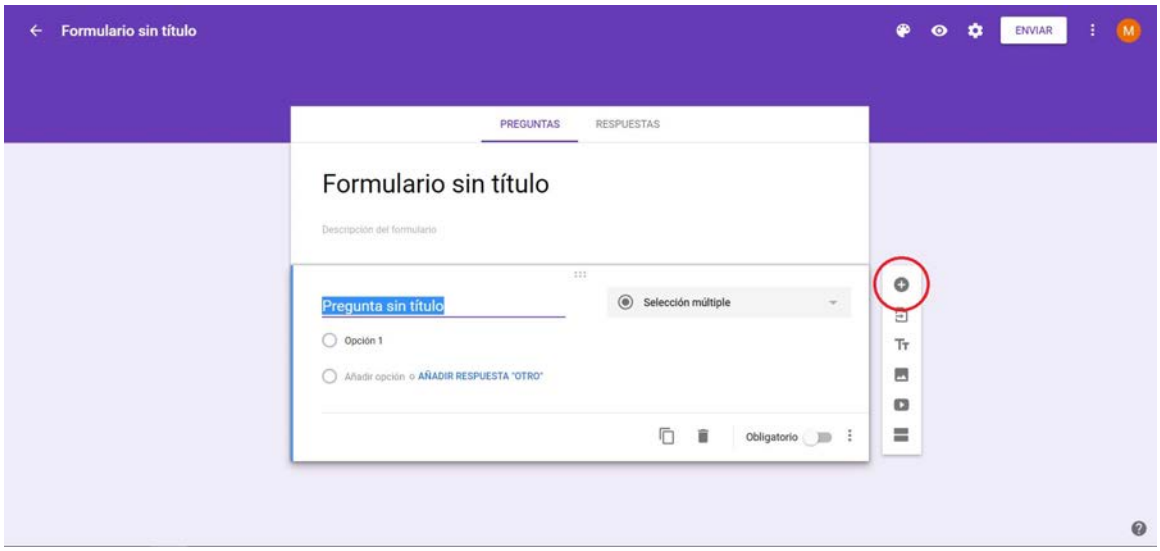


Figura 13.2 – Formulario de Google I

Ahora, se le da un título al formulario y se introducen las preguntas, que serán las variables que van a ser enviadas por el Arduino. Para añadir más variables se clica en el símbolo de + rodeado en rojo en la Figura 13.2.

Para que el Arduino pueda enviar los valores de las variables es necesario establecer el método de respuesta como “Respuesta corta” en el desplegable que se puede ver en la Figura 13.3.

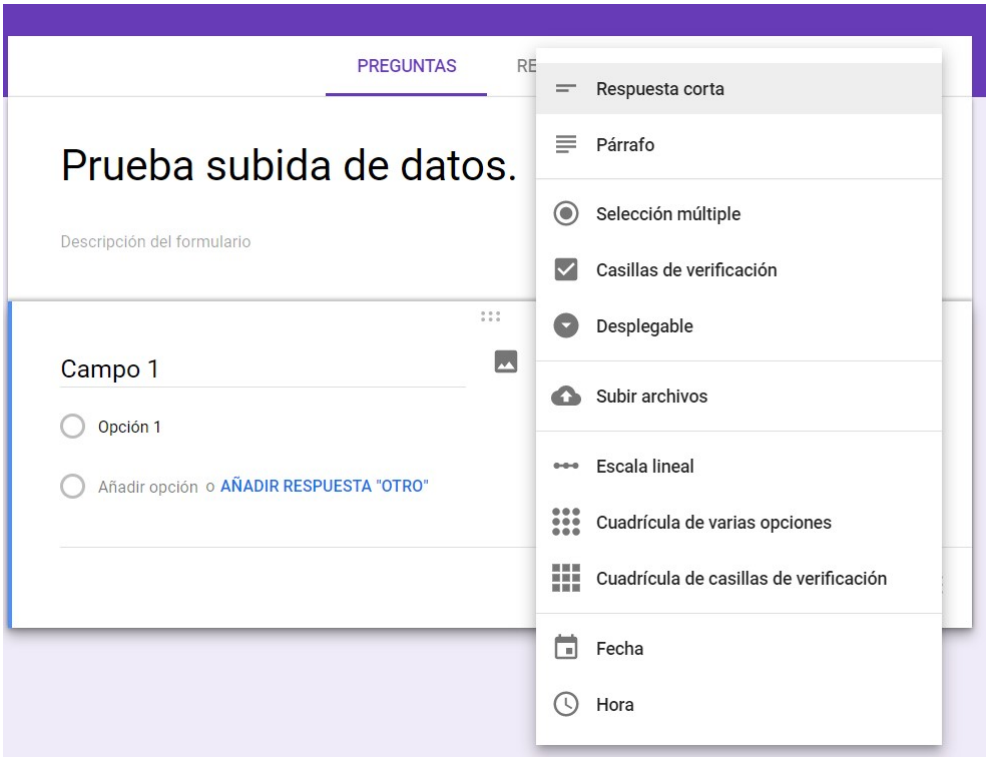


Figura 13.3 – Formulario de Google II

Ahora, para guardar los datos en una hoja de cálculo se clica, dentro de la pestaña de

“RESPUESTAS”, a “Crear hoja de cálculo”.

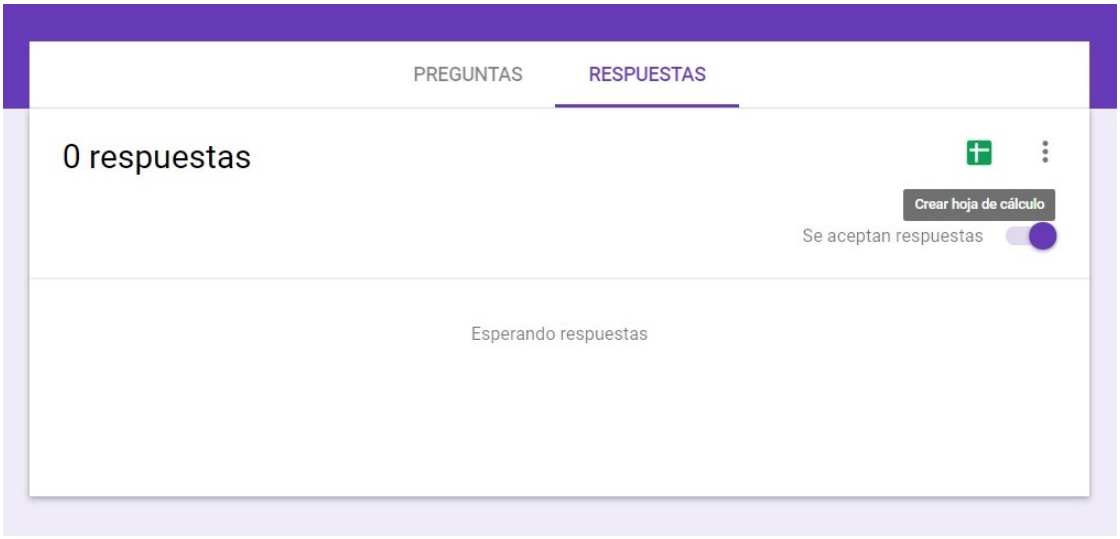


Figura 13.4 – Formulario de Google III

Ahora, en la esquina superior derecha, se clica en visualizar el formulario. Una vez visualizado se hace clic derecho en el campo de entrada y se le da a “inspeccionar elemento”.

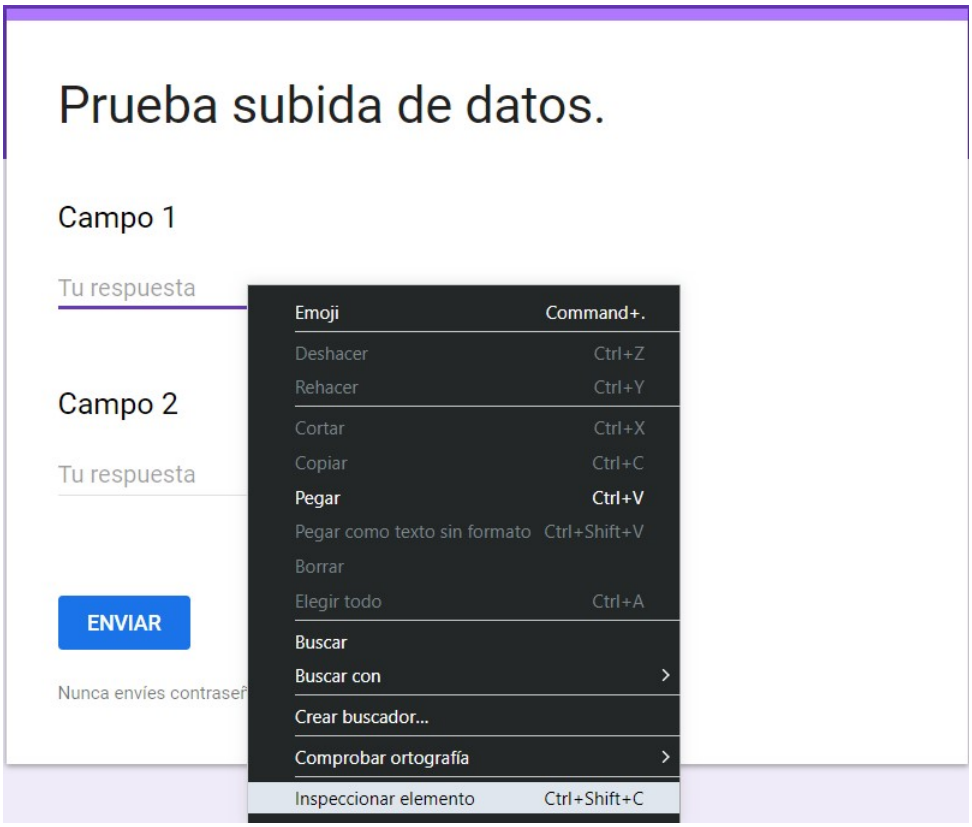


Figura 13.5 – Formulario de Google IV

En el desplegable que se abre a la derecha es necesario copiar el campo que se ve marcado en la siguiente Figura 13.6, ya que se utilizara para enviar los datos a la nube. Esto es necesario hacerlo con todos los campos creados.

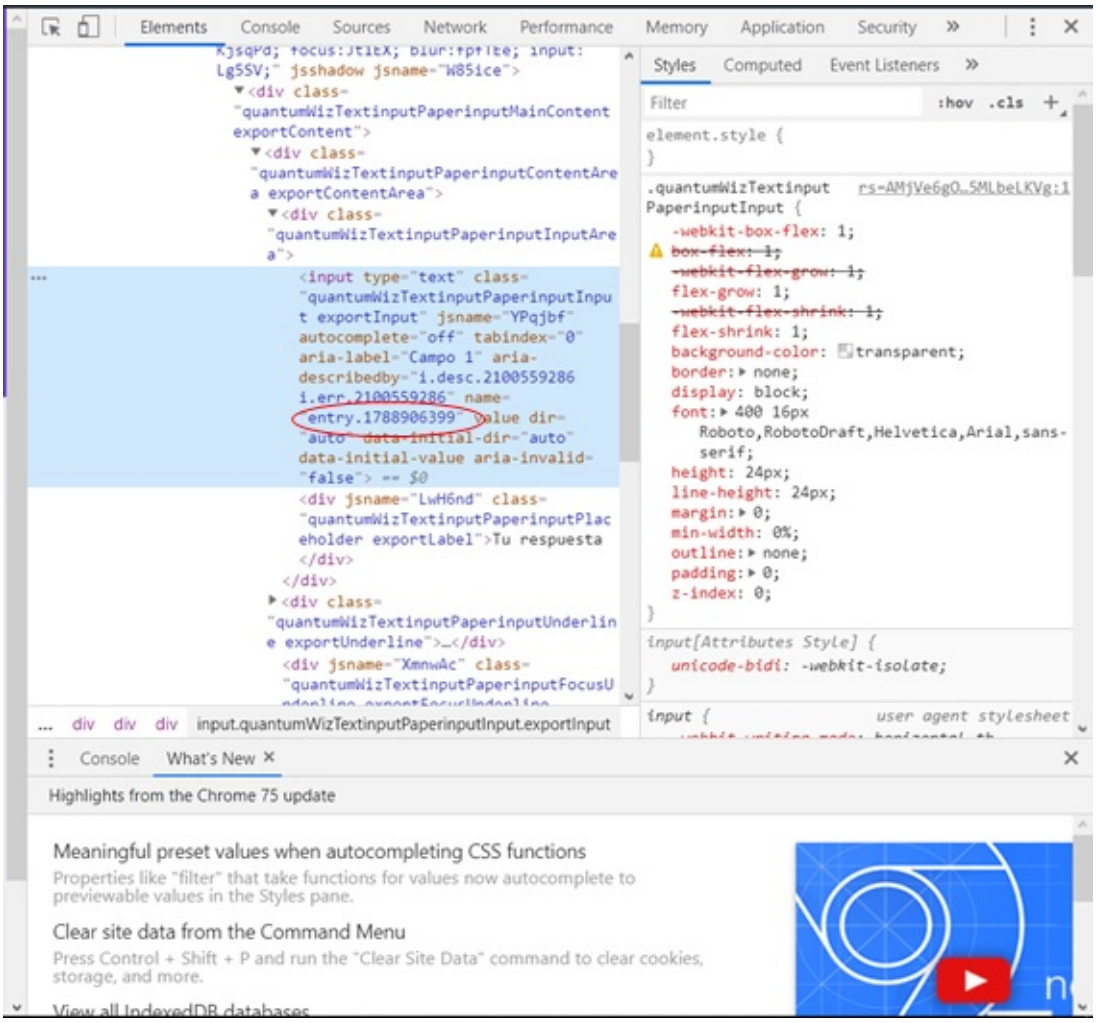


Figura 13.6 – Obtención del nombre del campo de entrada

Por último, se copia el “formkey” del formulario. Este se encuentra en el enlace del formulario.

En este caso el enlace del formulario es el siguiente: https://docs.google.com/forms/d/e/1FAIpQLScoVDKAl9tGjSwgWUAXb83Ge_vs3HpPmX0ZbY5eCKQEowjsKA/viewform

Y, por lo tanto, el *formkey* es:

“1FAIpQLScoVDKAl9tGjSwgWUAXb83Ge_vs3HpPmX0ZbY5eCKQEowjsKA”

Para subir los datos la solicitud http se debe hacer al siguiente enlace: docs.google.com/forms/d/e/<formkey>/formResponse?<campo>=<valor>&submit=Sumbit

En este caso sería el siguiente enlace:

docs.google.com/forms/d/e/1FAIpQLScoVDKAl9tGjSwgWUAXb83Ge_vs3HpPmX0ZbY5eCKQEowjsKA/formResponse?entry.1788906399=<valor>&submit=Sumbit

13.2. Google Cloud Platform

En primer lugar, hay que registrarse en la web https://cloud.google.com y crear un proyecto.

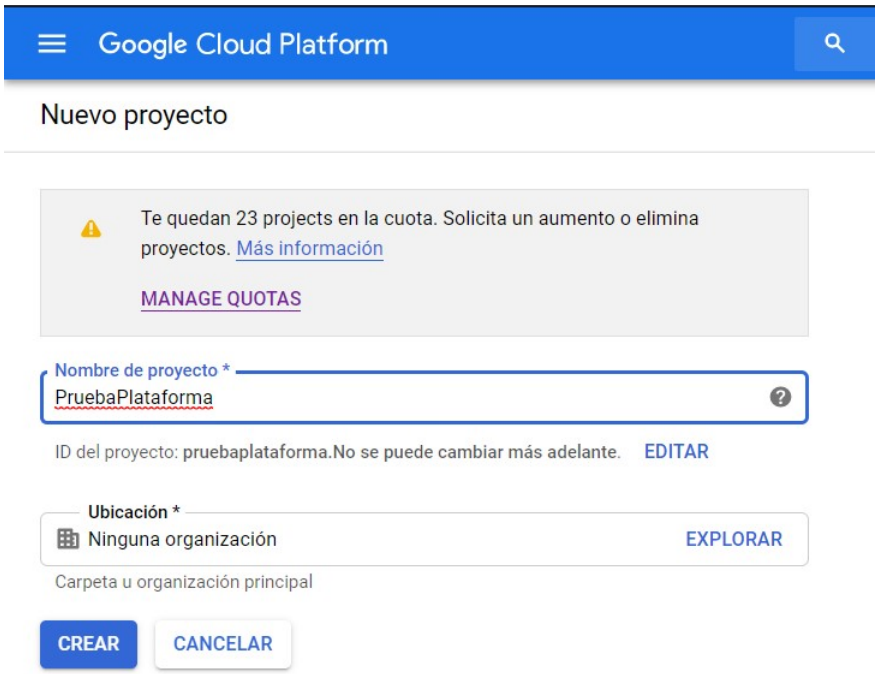


Figura 13.7 – Creación del proyecto

Al crear el proyecto se abre el panel de control, en el que se puede ver información sobre el proyecto.

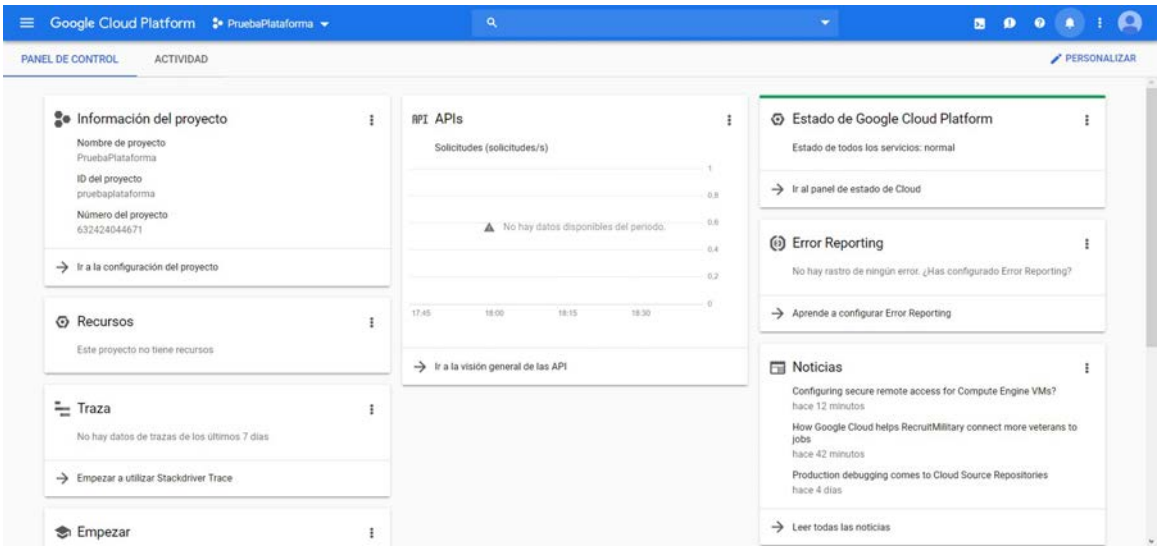


Figura 13.8 – Panel de control de Google Cloud

13.2.0.1. IoT Core

El primer servicio a configurar es “IoT Core”, el cual se encarga de la comunicación entre el Arduino y Google Cloud Platform. Este se puede encontrar en el desplegable de la izquierda.

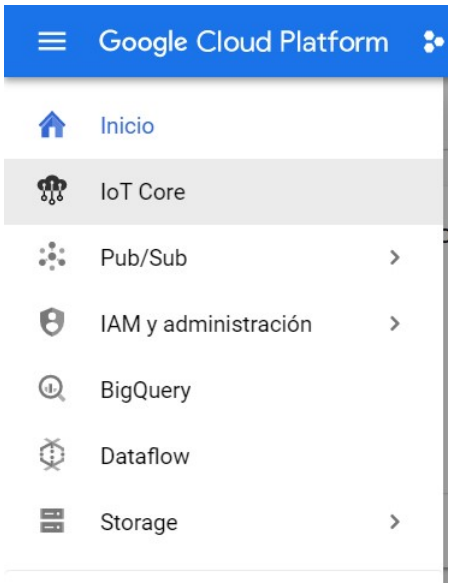


Figura 13.9 – IoT Core

En la ventana que se abre a continuación se clica en “Habilitar”.

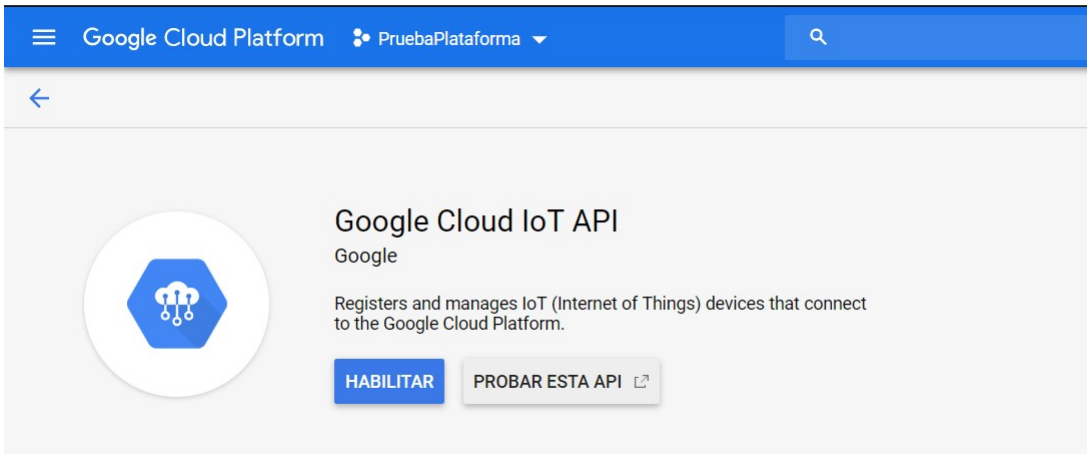


Figura 13.10 – Habilitación de IoT Core

A continuación, se crea un nuevo registro.

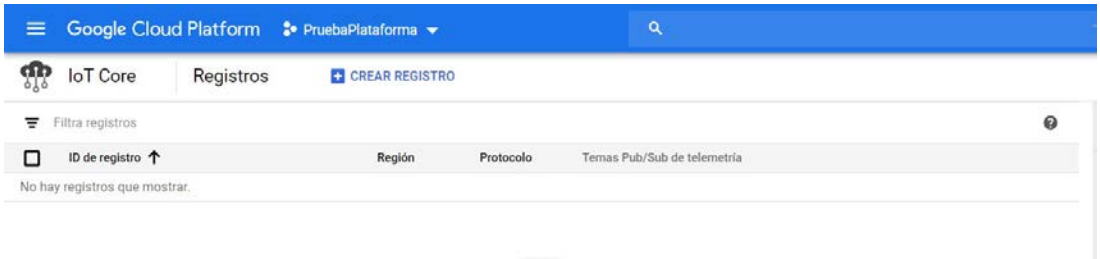


Figura 13.11 – Registros de IoT Core

Ahora, se le asigna un nombre al registro y una ubicación. En cuanto a los protocolos de comunicación, se puede marcar únicamente el que se vaya a utilizar o dejar los dos marcados. En este caso se dejan ambos seleccionados.

Es necesario crear un tema Pub/Sub. Estos se encargan de extraer los datos del IoT Core y mandarlos a otros servicios de la plataforma. Este tema Pub/Sub se debe seleccionar en “tema de telemetría predeterminado”.

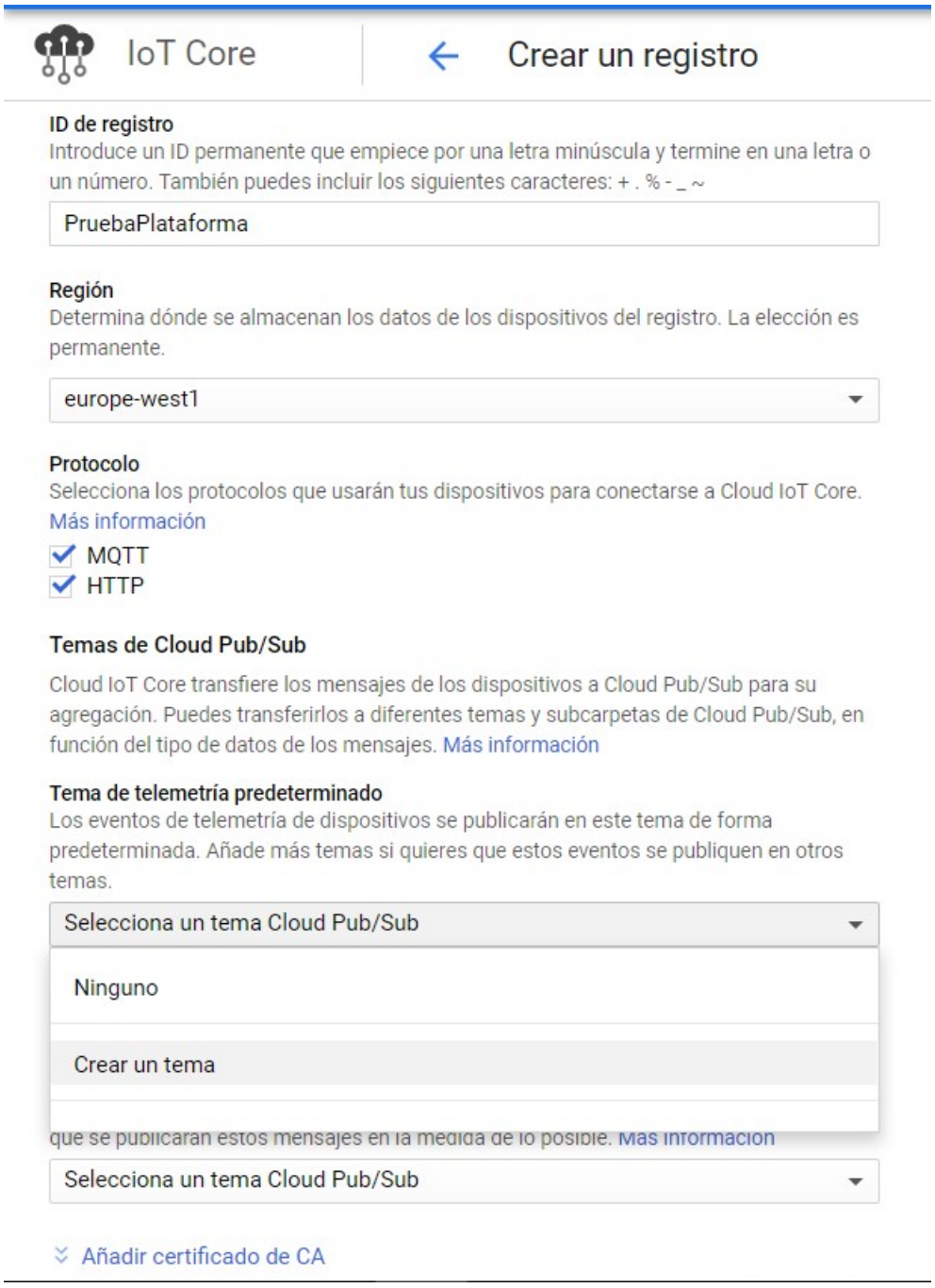


Figura 13.12 – Creación del registro de IoT Core I

Se le da un nombre y se deja seleccionado “Clave administrada por Google”.

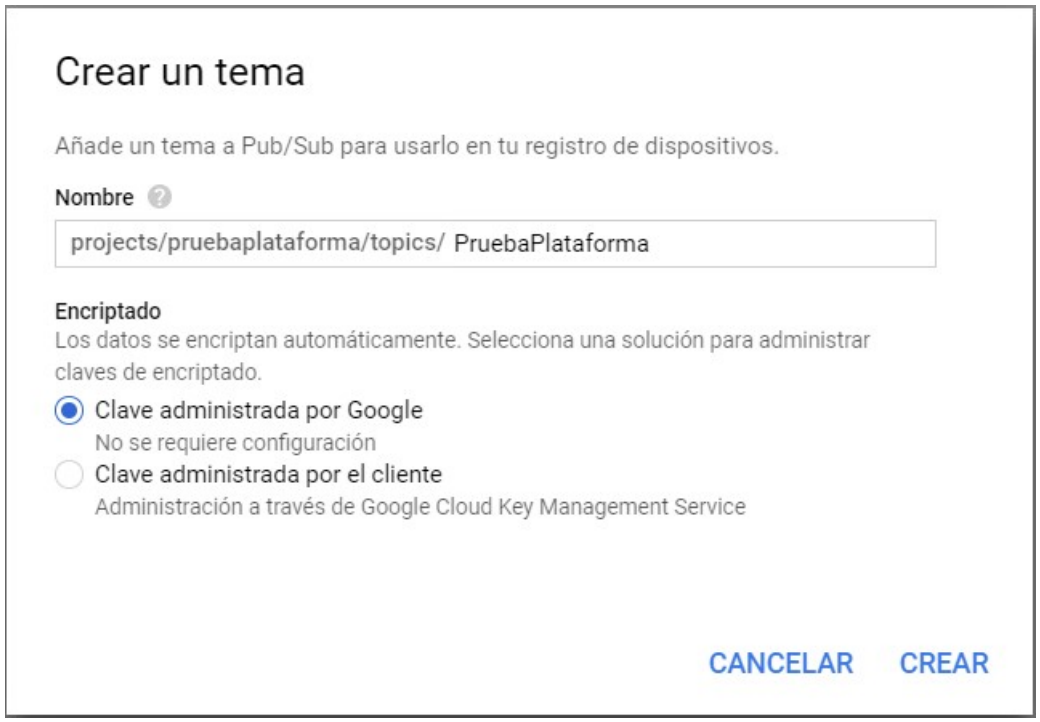


Figura 13.13 – Creación del tema Pub/Sub

Ahora, en “Tema de telemetría predeterminado” se selecciona el tema creado. Sin embargo, en “Tema de estado del dispositivo” y “Stackdriver Loggin” se selecciona “Ninguno”.

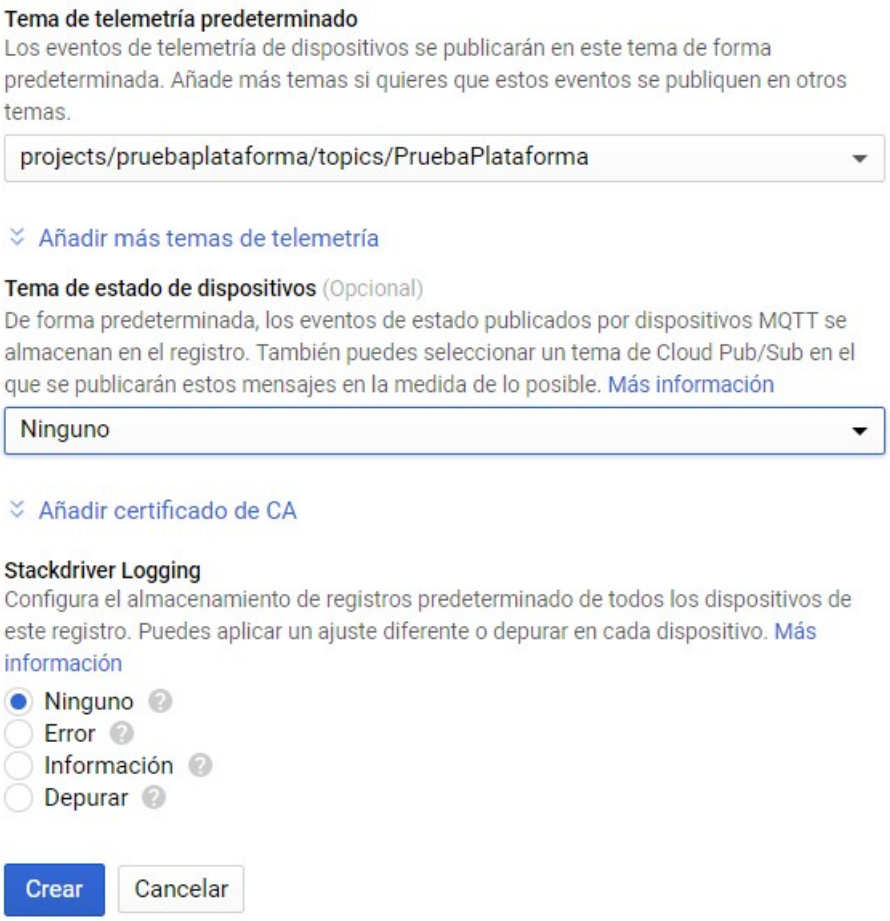


Figura 13.14 – Creación del registro de IoT Core II

De este modo, se crea el registro.

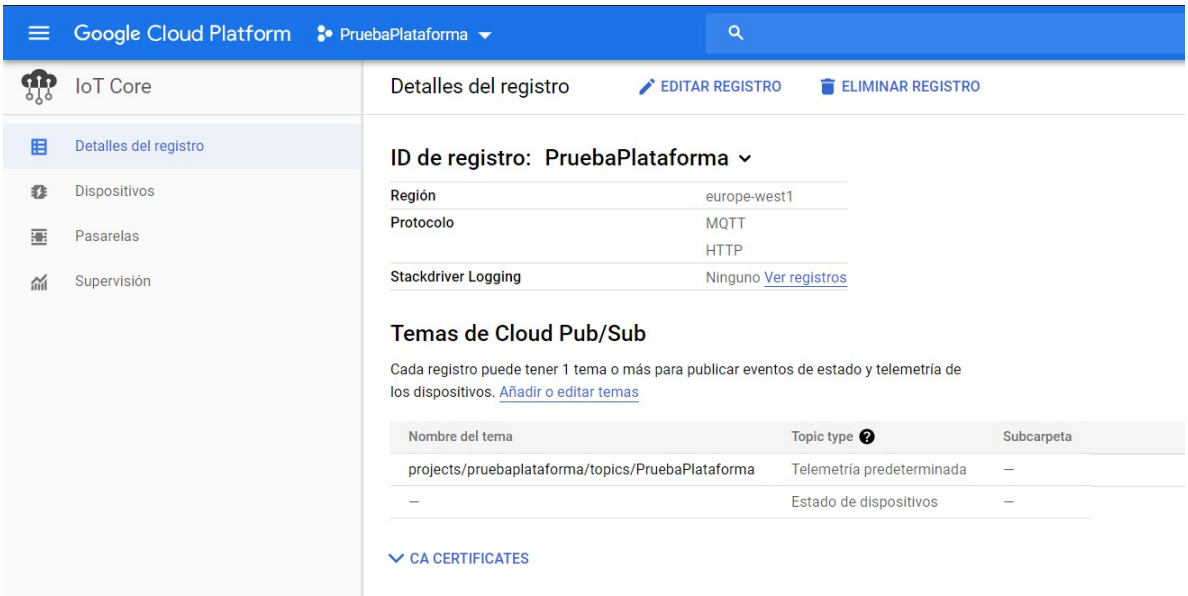


Figura 13.15 – Registro de IoT Core

Ahora, en el apartado de “Dispositivos” se clica en “Crear dispositivo”.

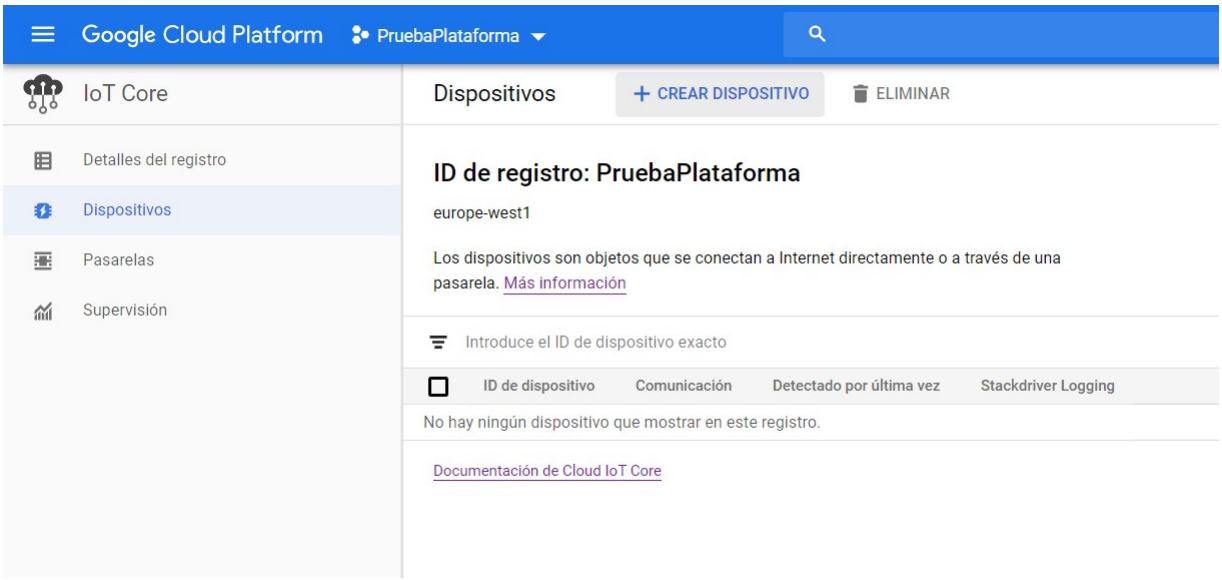


Figura 13.16 – Dispositivos IoT Core

Se le da nombre al dispositivo y en el formato de la clave publica se selecciona “ES256”. Esta clave la obtenemos mediante el código de Arduino del Apartado 12.1.4.1. Este código lo introducimos en la casilla y le damos a crear. En caso de no usar un Arduino MKR 1010 u otro que permita crear las claves, es necesario crearla mediante el sdk de la plataforma.

Figura 13.17 – Creación del dispositivo

13.2.0.2. PubSub

Ahora se procede a la configuración del servicio “Pub/Sub”, que extrae la información de “IoT Core” para que pueda ser utilizada por otros servicios. Este servicio se puede encontrar en el desplegable de la izquierda.

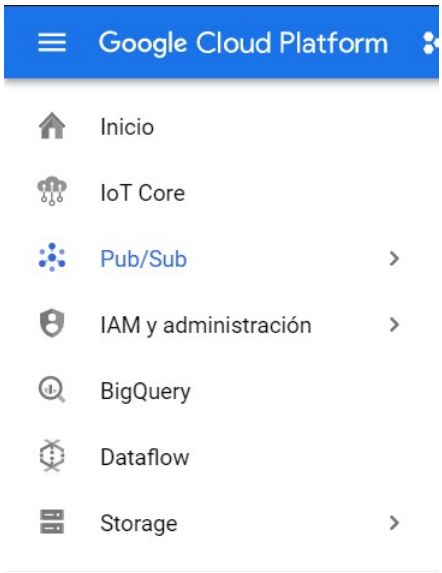


Figura 13.18 – Pub/Sub

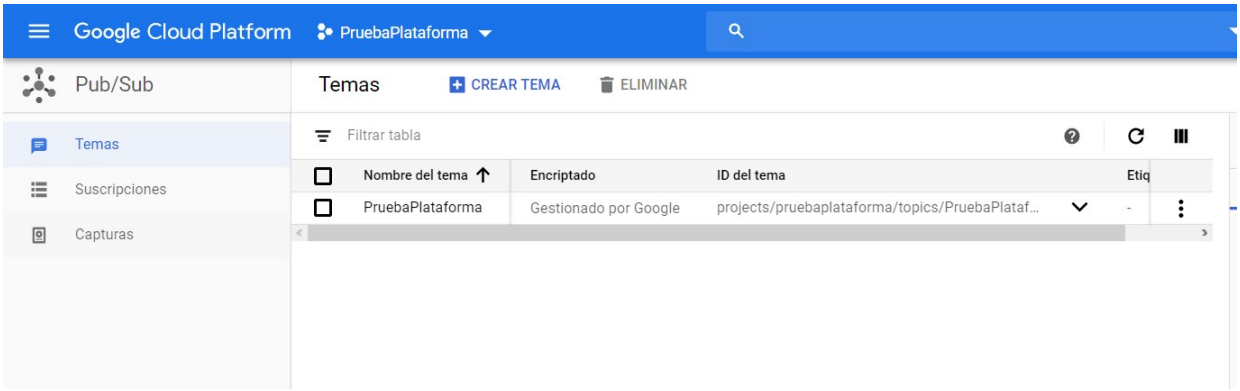


Figura 13.19 – Temas de Pub/Sub

Como se puede ver en la Figura 13.19, ya aparece el Pub/Sub creado previamente. Ahora, se configurará una suscripción al IoT Core creado en el apartado anterior. Por lo tanto, en el apartado de “suscripciones” clica en “Crear suscripción”.

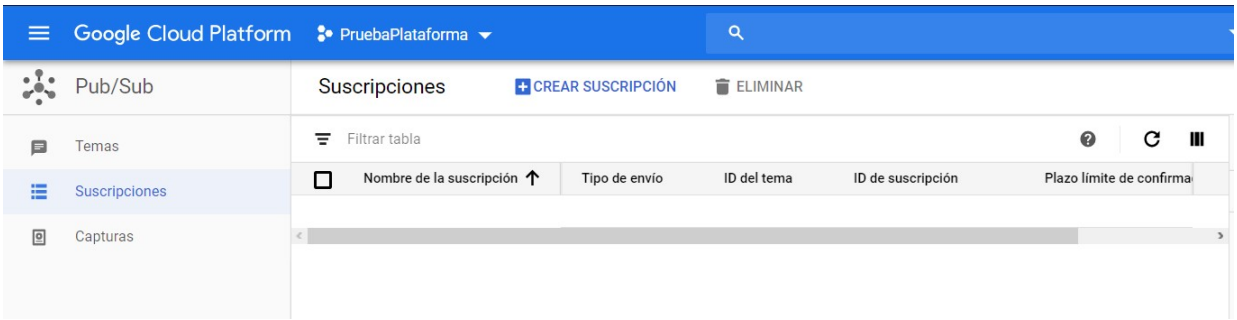


Figura 13.20 – Suscripciones Pub/Sub

Se le da la configuración correspondiente y se crea la suscripción.

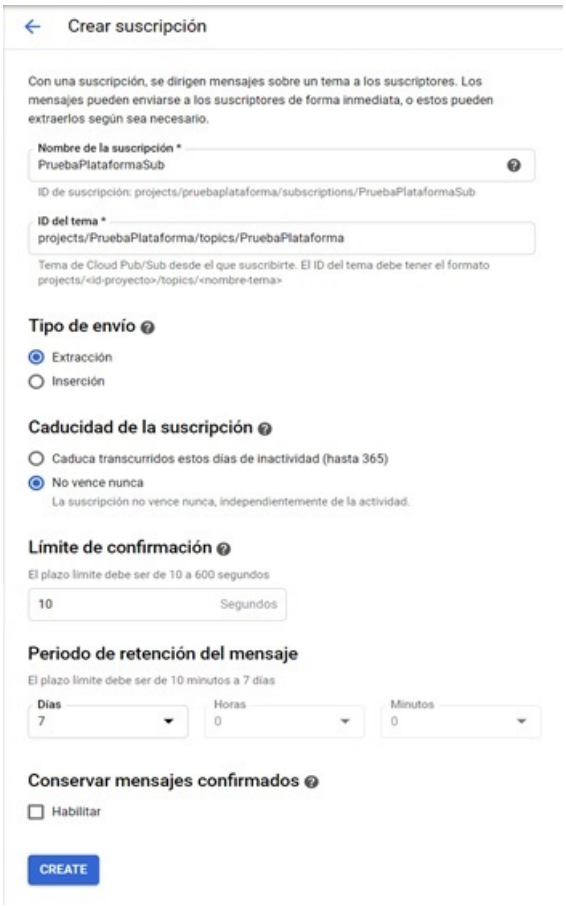


Figura 13.21 – Creación de la suscripción Pub/Sub

Por último, se abre la suscripción y se copia el ID de la suscripción, ya que hará falta en pasos posteriores.

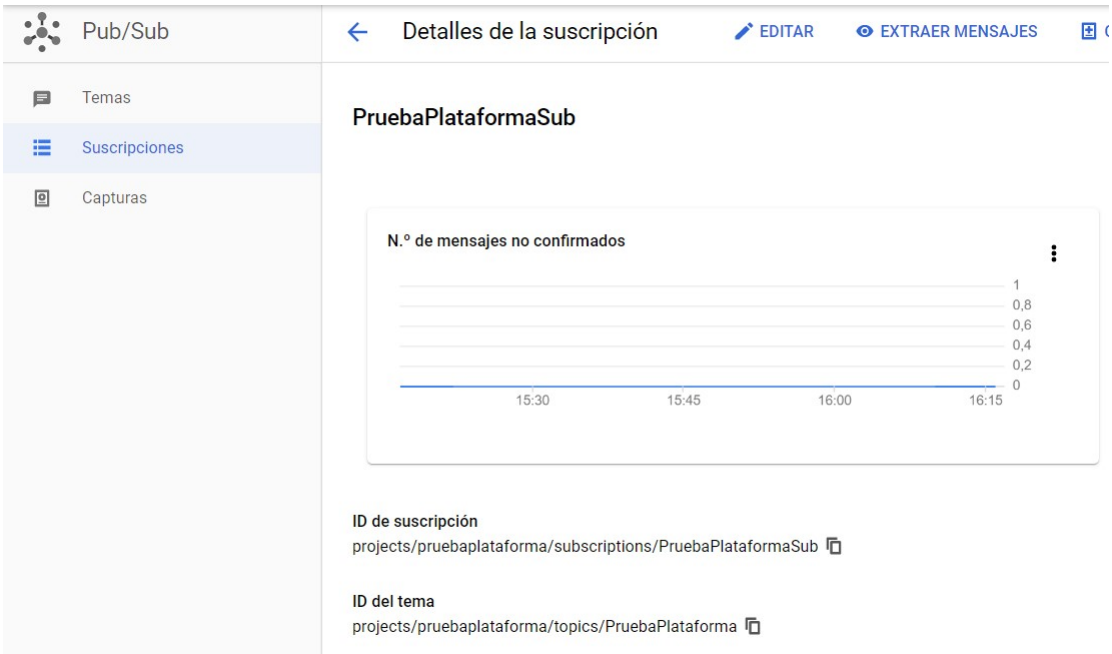


Figura 13.22 – Suscripción al registro de IoT Core

13.2.0.3. BigQuery

Ahora, se procede a configurar la tabla en la que se van a almacenar los datos. Para ello, en el desplegable de la izquierda, se abre el servicio BigQuery.

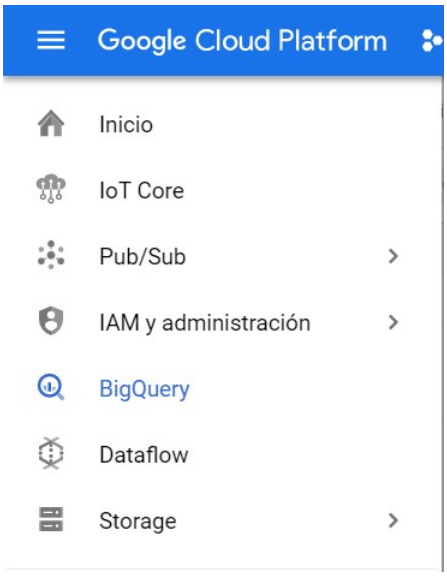


Figura 13.23 – BigQuery

Ahora, en la barra lateral, se clica en el desplegable que tiene el mismo nombre que el proyecto, que se encuentra en el apartado de “Recursos” y se clica en “Crear conjunto de datos”.

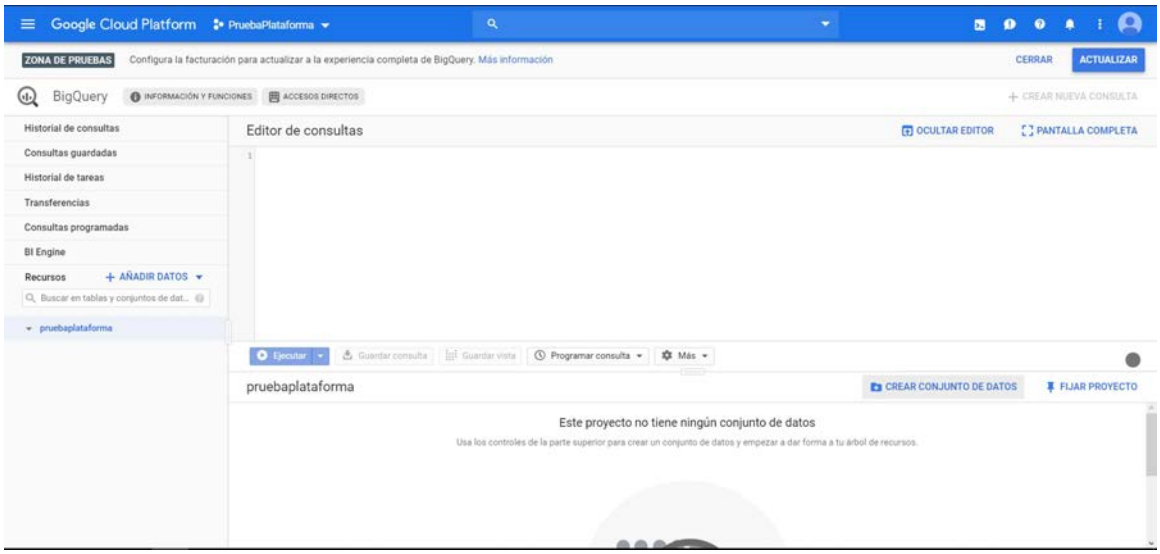


Figura 13.24 – Recurso de BigQuery

Ahora, se le da nombre a la tabla y se crea.

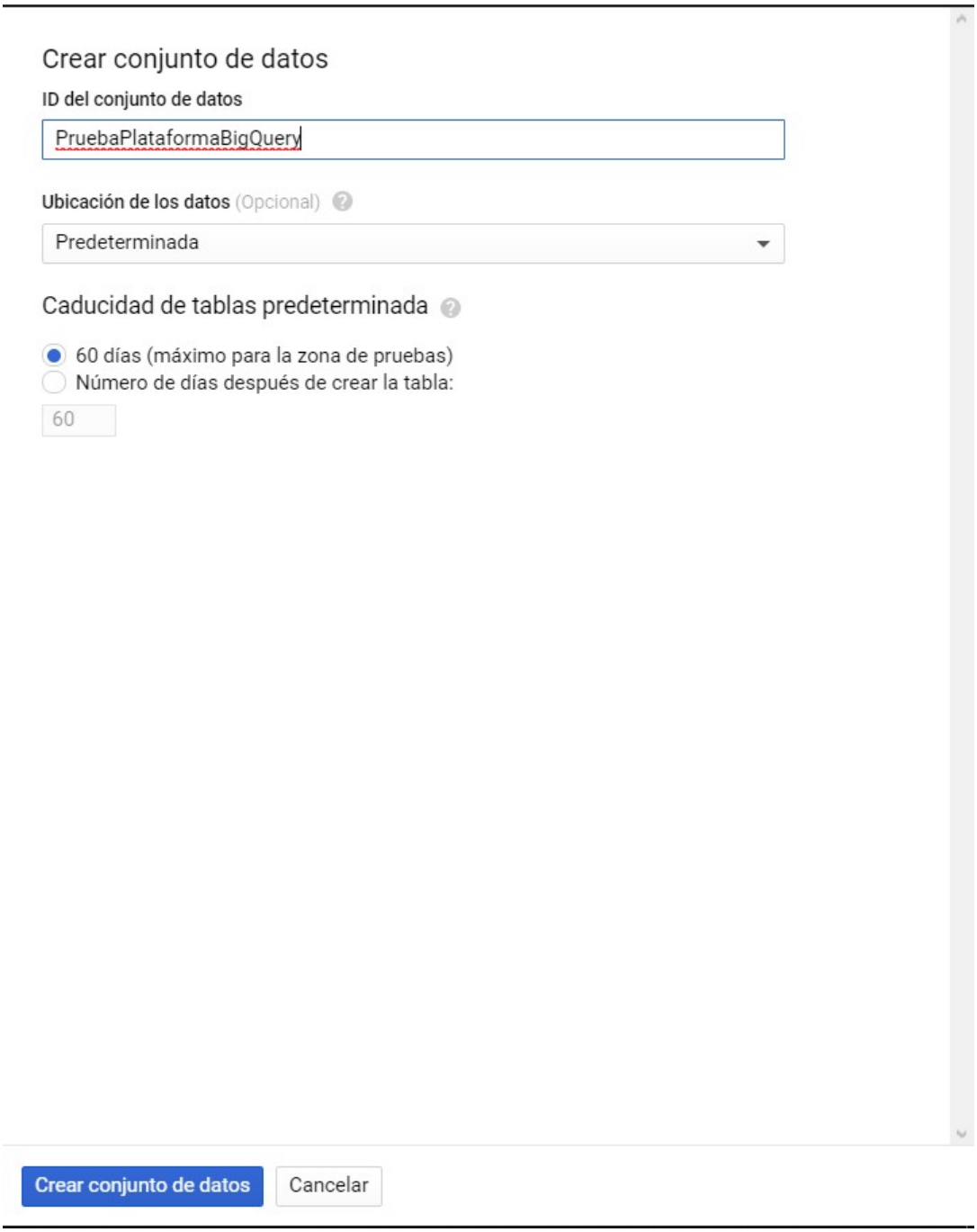


Figura 13.25 – Creación del conjunto de datos

Ahora se abre el conjunto de datos, clicando en él, y se clica en “Crear tabla”.

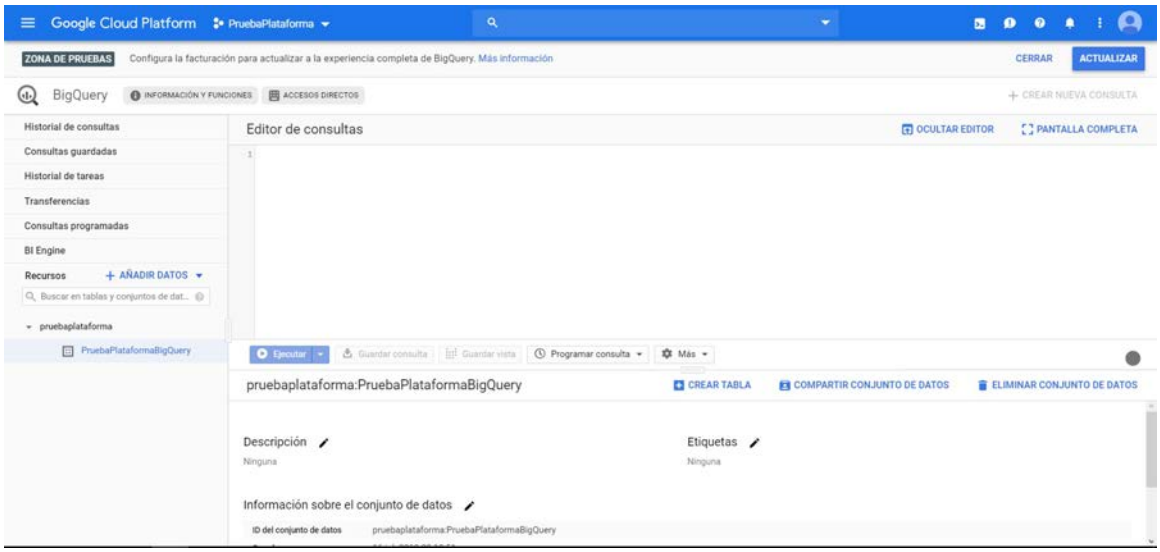


Figura 13.26 – Conjunto de datos

Se le da nombre a la tabla y se introducen las variables que van a ser subidas a la plataforma desde el Arduino. Las variables deben tener el mismo nombre que en el Arduino.

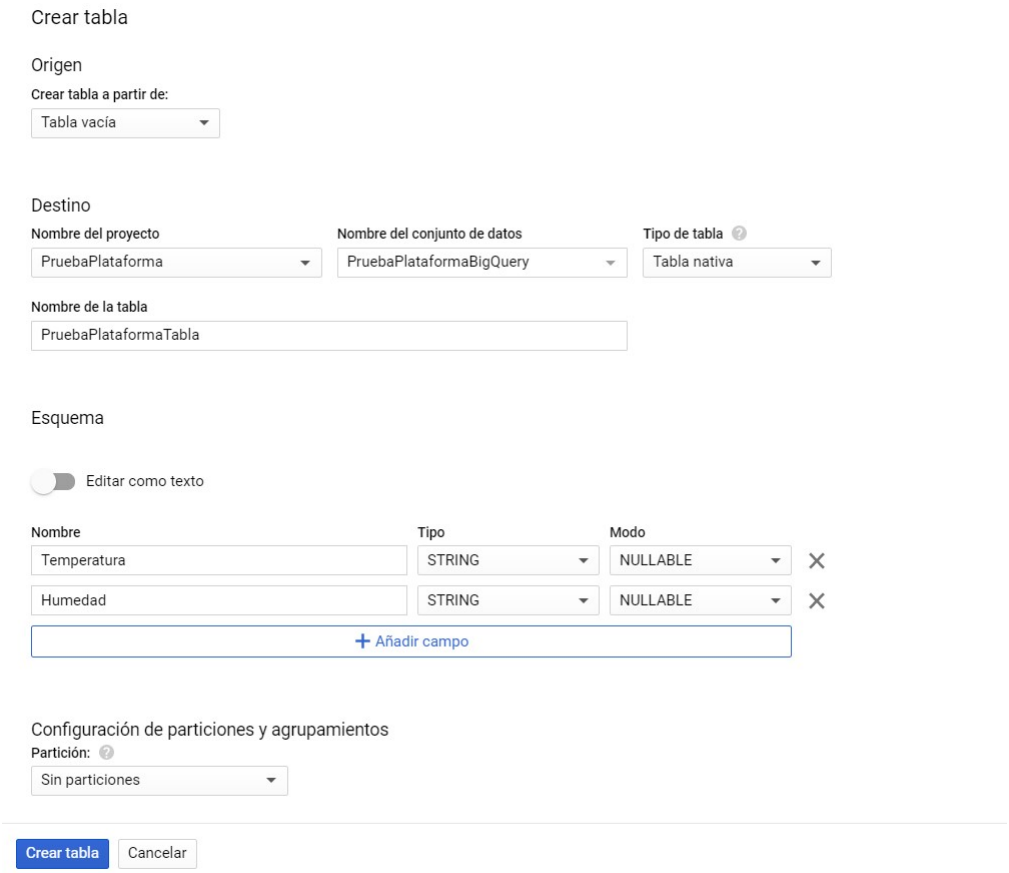


Figura 13.27 – Creación de la tabla

Por último, se abren los detalles de la tabla y se copia su ID, ya que será necesario en pasos posteriores.

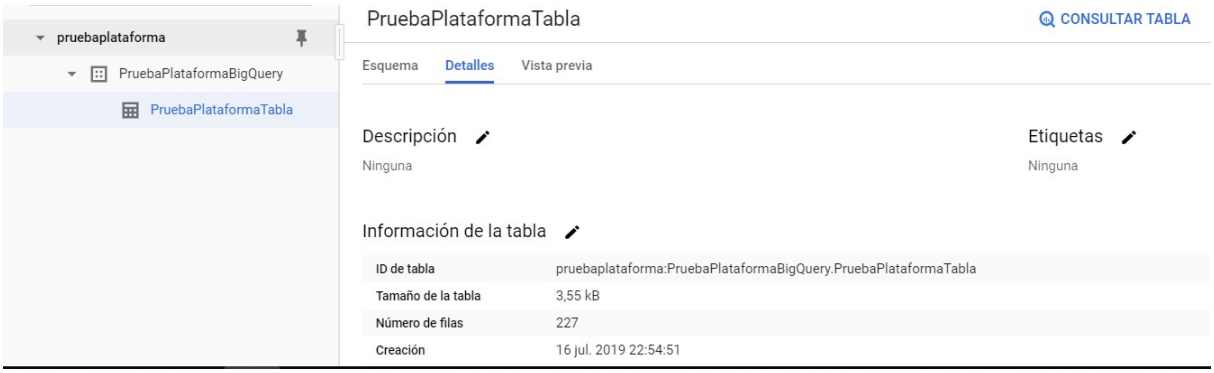


Figura 13.28 – ID de la tabla de BigQuery

13.2.0.4. Storage

Ahora se configurará el almacenamiento de los archivos temporales producto del intercambio de información. Para ello, en el desplegable de la izquierda, se abre el servicio “Storage”.

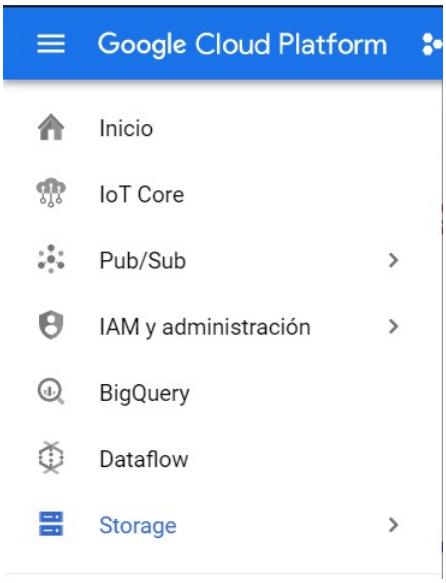


Figura 13.29 – Storaje

Ahora se clica en “Crear segmento”.

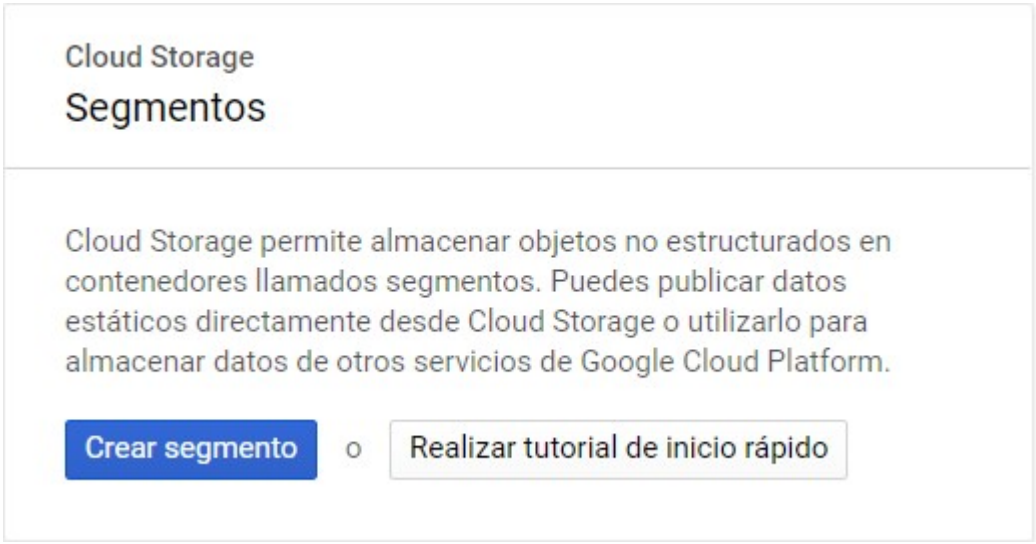


Figura 13.30 – Creación del segmento

Se le da nombre y se introducen las configuraciones correspondientes (el nombre no puede contener mayúsculas).

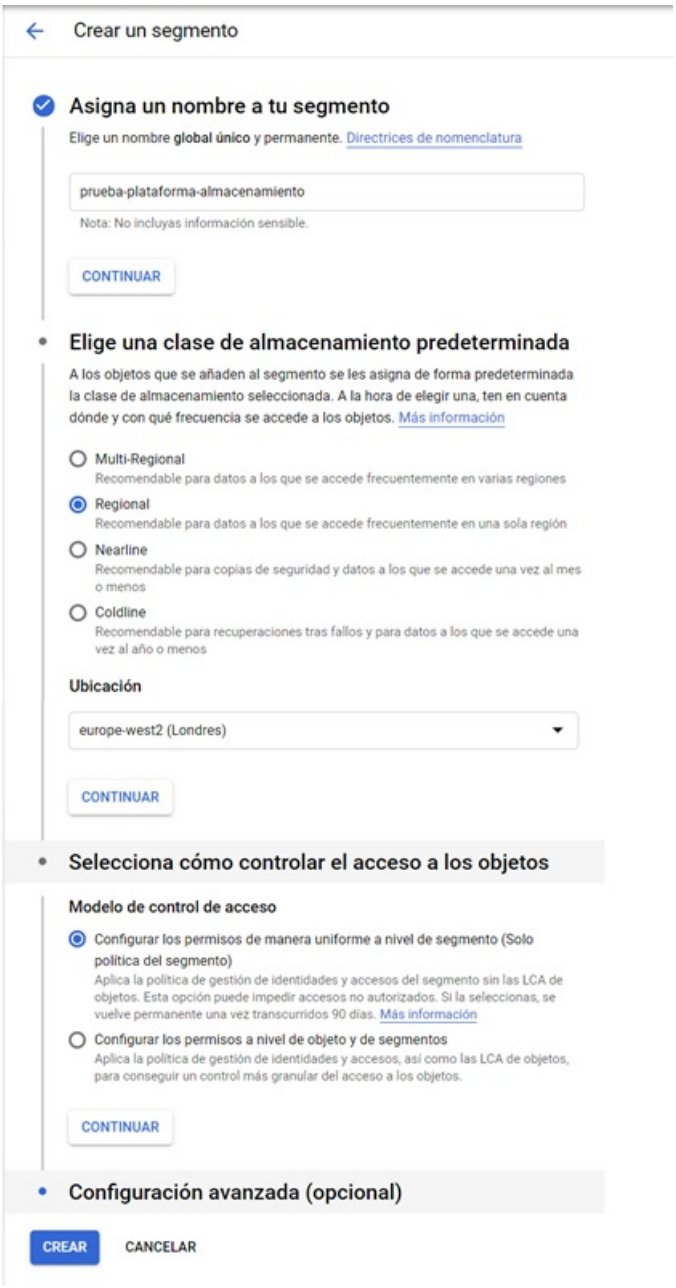


Figura 13.31 – Configuración del segmento

Ahora se clica “Crear carpeta” y se le da nombre. En este caso se nombra como “temporal”.

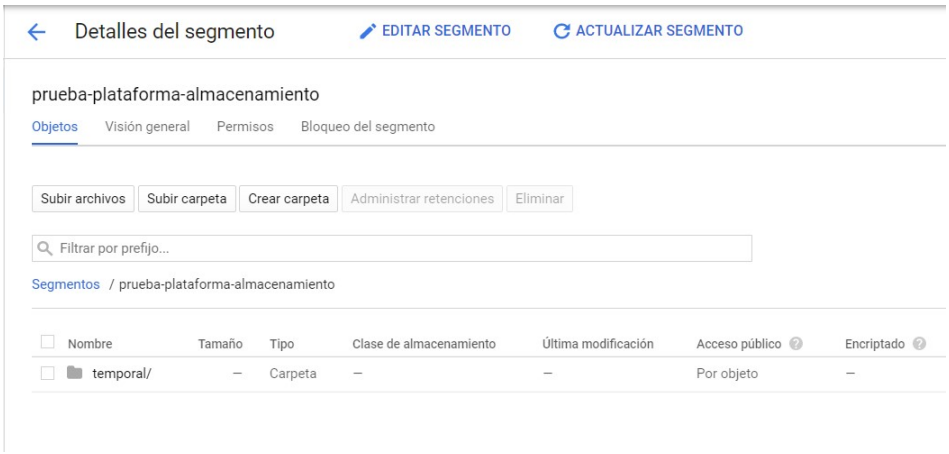


Figura 13.32 – Carpeta archivos temporales

13.2.0.5. Dataflow

Por último, se configura Dataflow, el servicio encargado de conectar el PubSub con BigQuery. Para ello, en el desplegable de la izquierda se abre “Dataflow”.

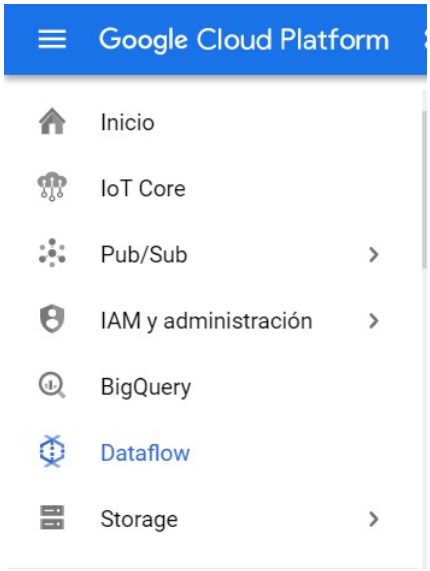


Figura 13.33 – Dataflow

Ahora, se introduce la siguiente configuración:

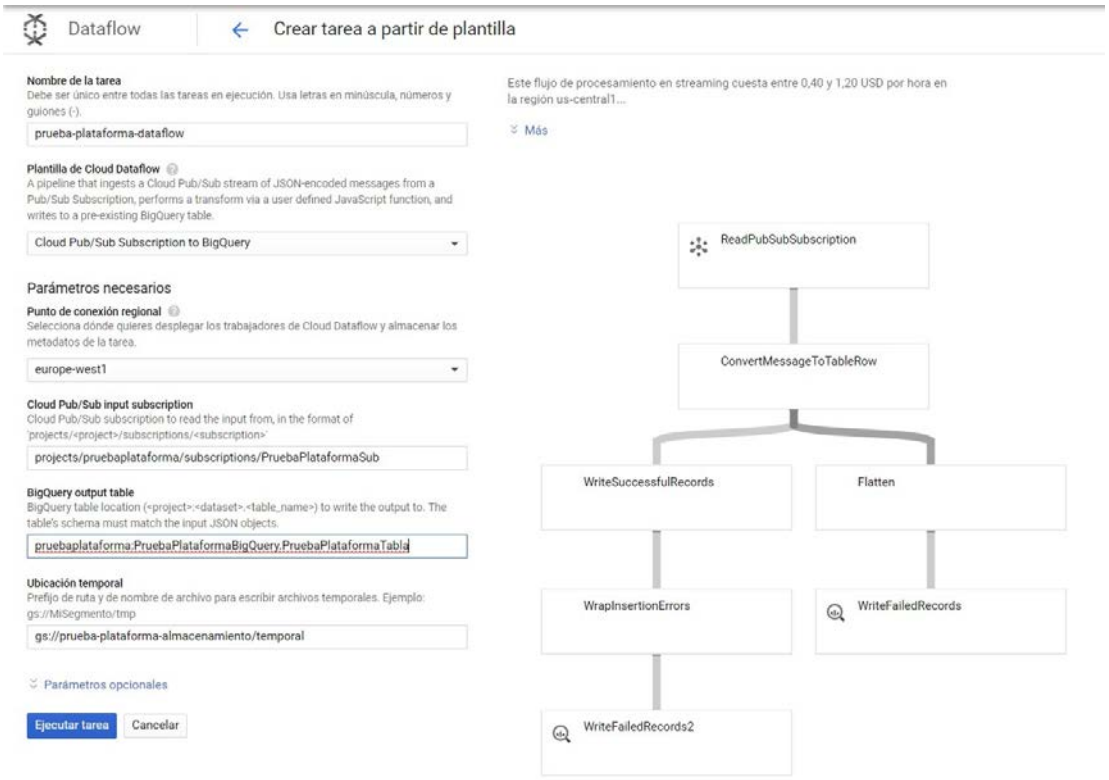


Figura 13.34 – Creación de Dataflow

13.2.0.6. Resultado

Para ver los datos subidos a la nube, es necesario realizar una consulta SQL en la tabla. Para ello, se clica en “Consultar tabla” y se introduce un asterisco entre “SELECT” y “FROM” en la consulta, para obtener la tabla completa.

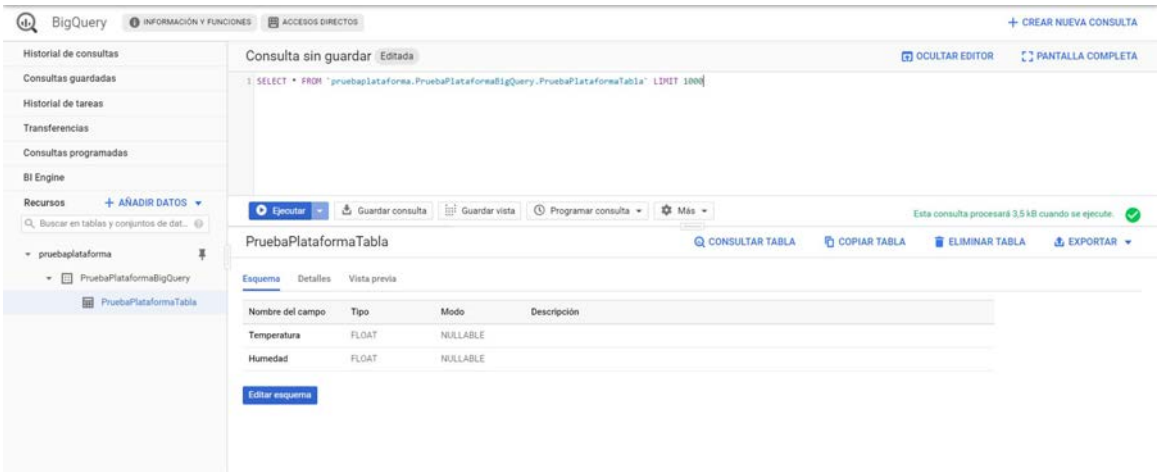


Figura 13.35 – Consulta SQL

Como se puede ver en la Figura 13.36, ahora se pueden ver todos los datos de la tabla.

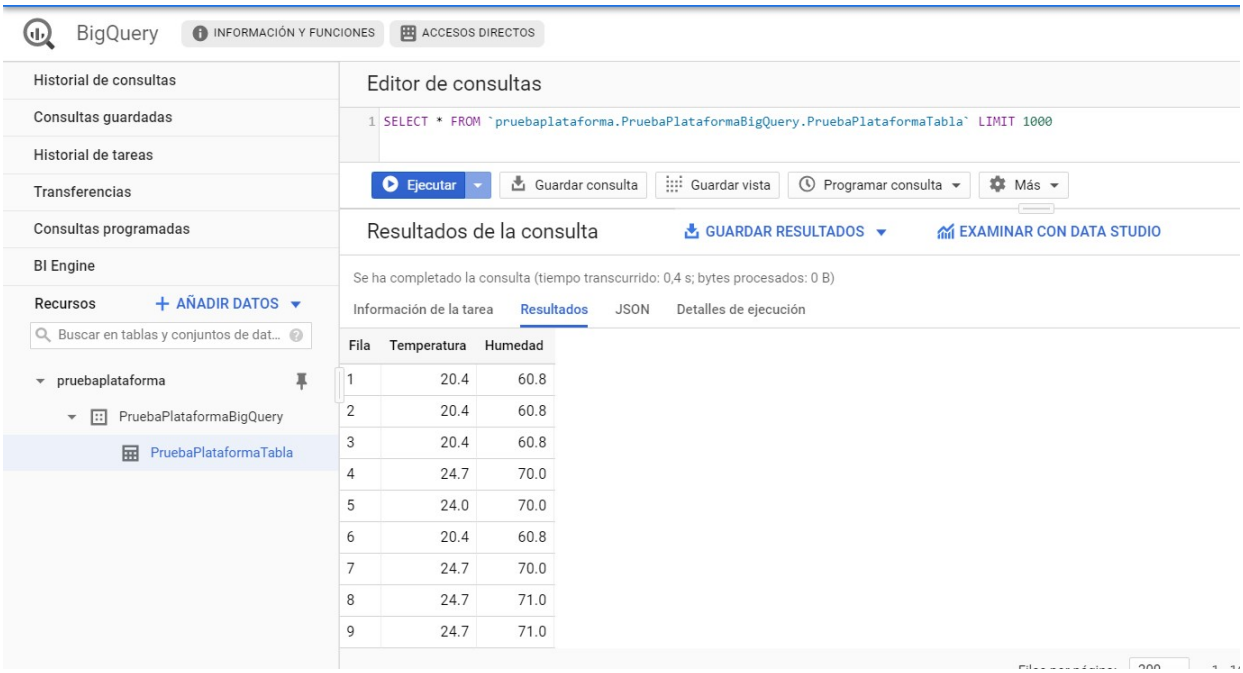


Figura 13.36 – Datos subidos a Google Cloud

Además, existe la posibilidad de exportar los datos para visualizarlos en Google Data Studio

13.3. Thingspeak

En primer lugar, es necesario crear una cuenta en https://thingspeak.com. Una vez hecho esto, se abrirá el panel de control:

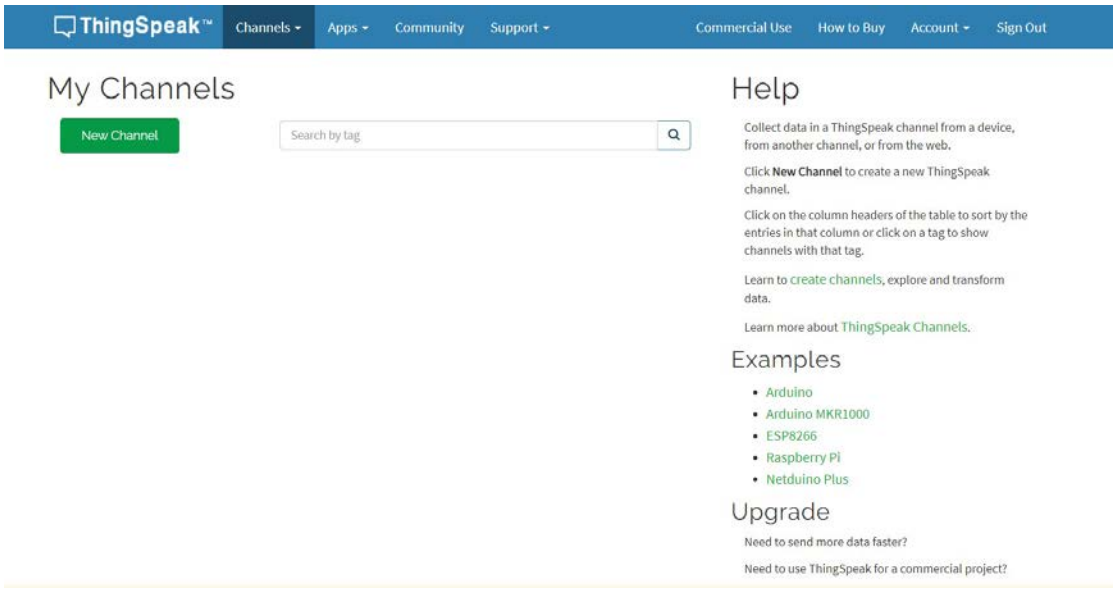


Figura 13.37 – Panel de control ThingSpeak

Ahora, en “New channel” se crea un nuevo canal. Para ello se rellenan los campos correspondientes, como se realiza en el la Figura 13.38

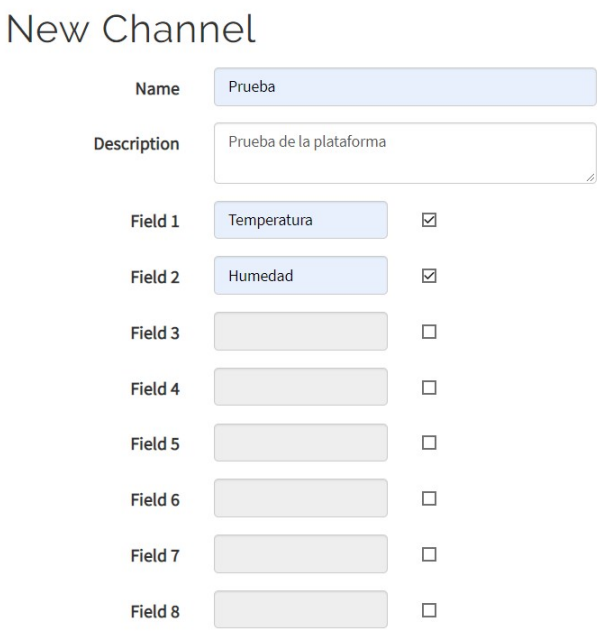


Figura 13.38 – Configuración de un nuevo canal

Ahora, en el apartado de “API Keys”, se copia la clave “Write API Key”, ya que será necesaria para subir los datos mediante solicitudes HTTP.

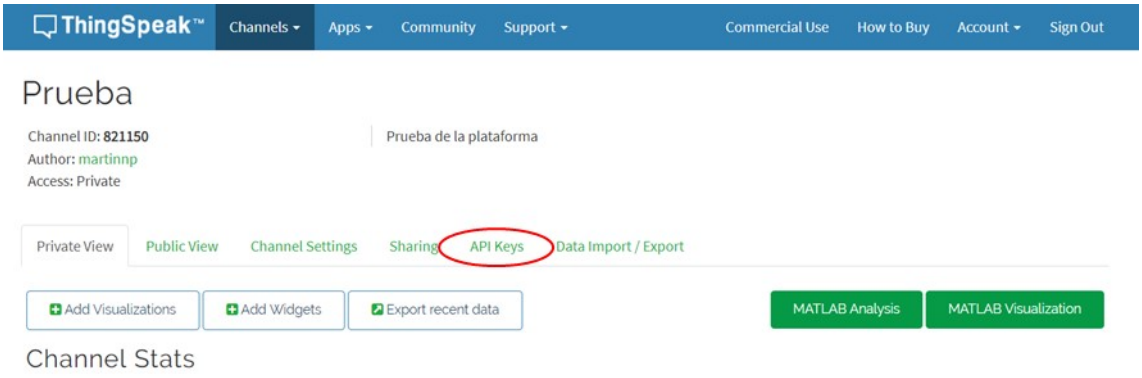


Figura 13.39 – API Keys

Una vez subidos los datos con el código recogido en el Apartado 12.1.5 se nos generan automáticamente las siguientes gráficas:

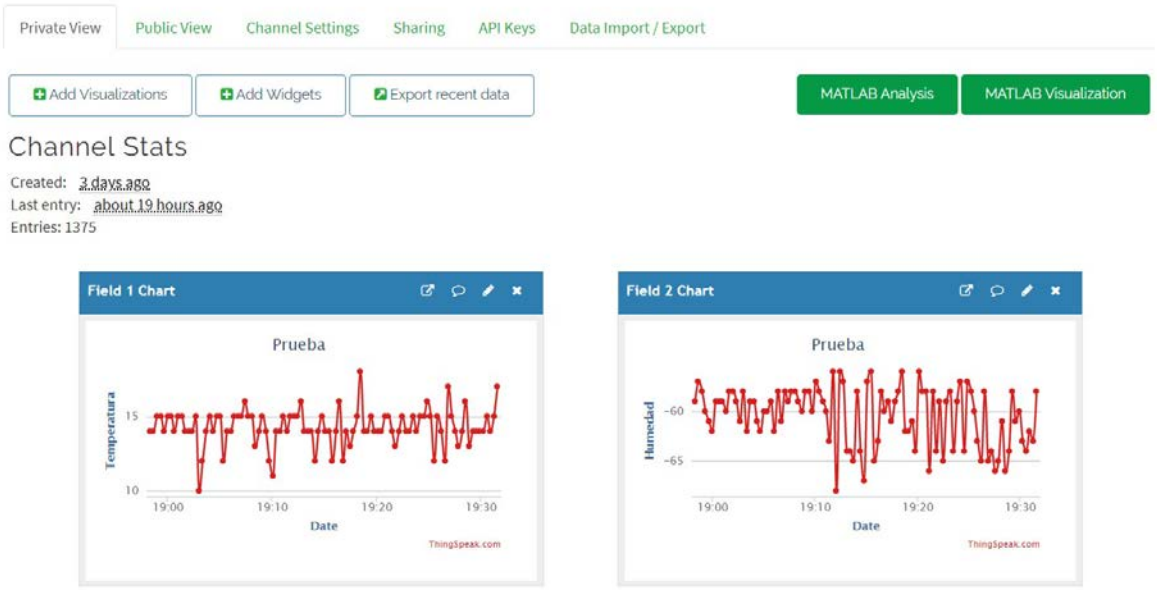


Figura 13.40 – Representación Thingspeak

Clicando en el símbolo del lápiz se abren las opciones de la gráfica:

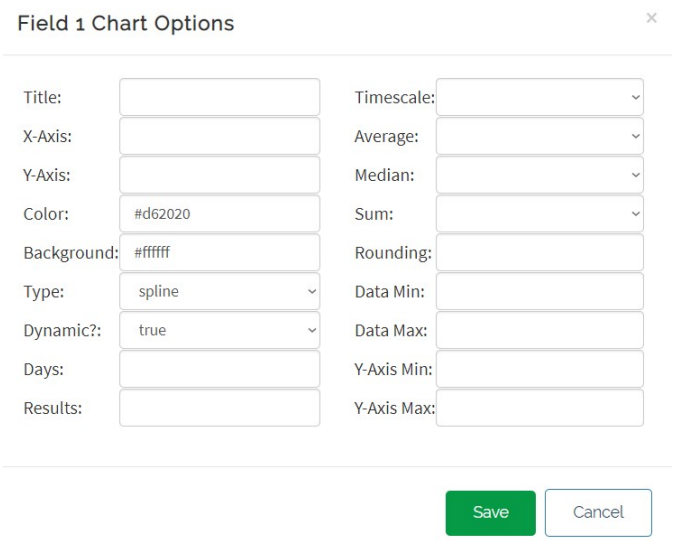


Figura 13.41 – Configuración de la representación

Las opciones que se pueden configurar en la gráfica son las siguientes:

Si se clica en “MATLAB Visualization” y en la siguiente ventana se marca “Custom (no starter code)” y luego se clica en crear, se abre una ventana en la que se puede introducir código de Matlab para visualizar los datos a voluntad.

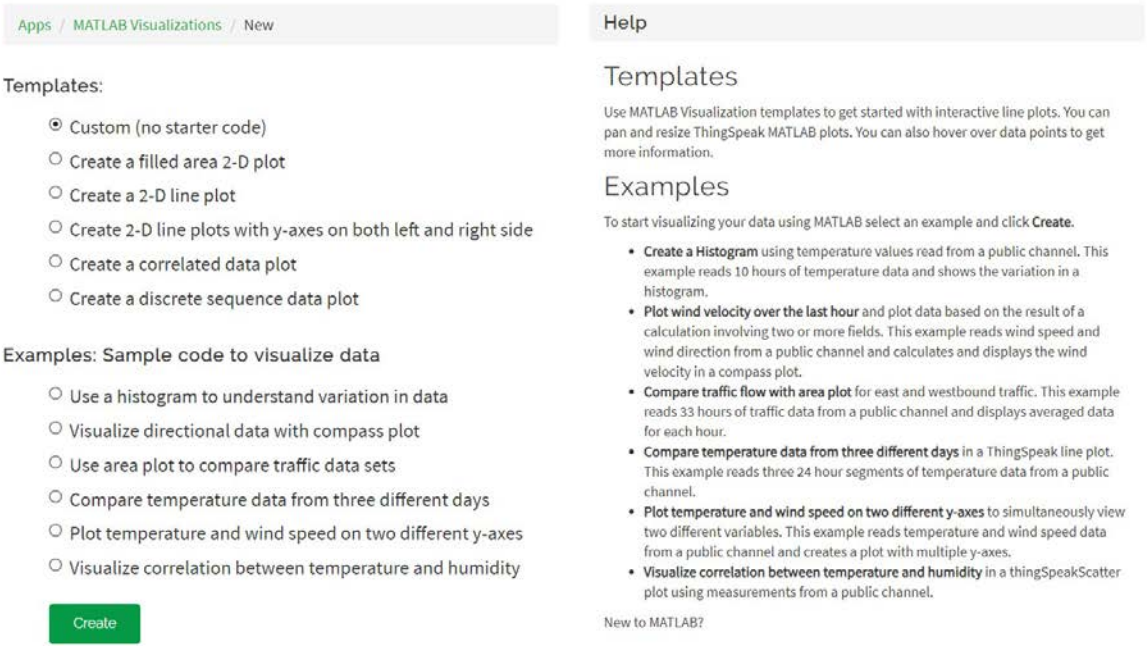


Figura 13.42 – Introducción del código Matlab I

Como por ejemplo el siguiente código permite graficar los últimos 100 valores de la variable 1:

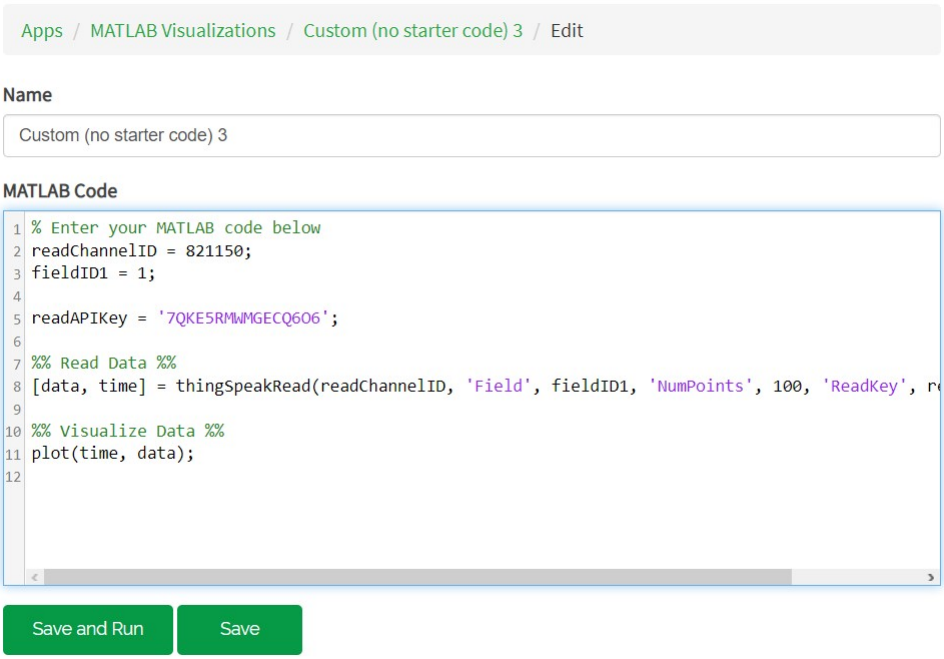


Figura 13.43 – Introducción del código Matlab II

Dándole a “Save and Run” tenemos la siguiente gráfica:

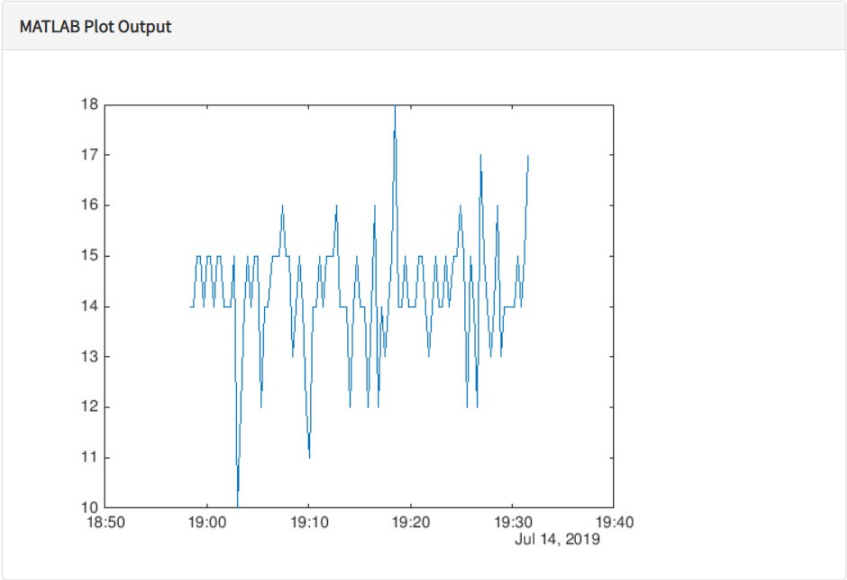


Figura 13.44 – Representación utilizando código Matlab

13.3.1. Exportado de los datos

Para exportar los datos hay que clicar en “Data Import/Export”.

En exportar se clica “Download”, seleccionando la zona horaria que se desee.

Figura 13.45 – Exportado de los datos

De este modo se descargará un archivo en formato .csv que podremos introducir en diferentes programas como Excel o MATLAB.

13.4. Thinger.io

En primer lugar, es necesario registrarse en la web `thinger.io`.

Para configurar un dispositivo es necesario abrir el apartado “Devices”, que se encuentra en el menú desplegable de la izquierda. Una vez abierto, se clica en “Add device”, añadiendo los datos del dispositivo que se desee añadir. Para la credencial, se puede introducir una o crear una aleatoria en “Generate Random Credential”.

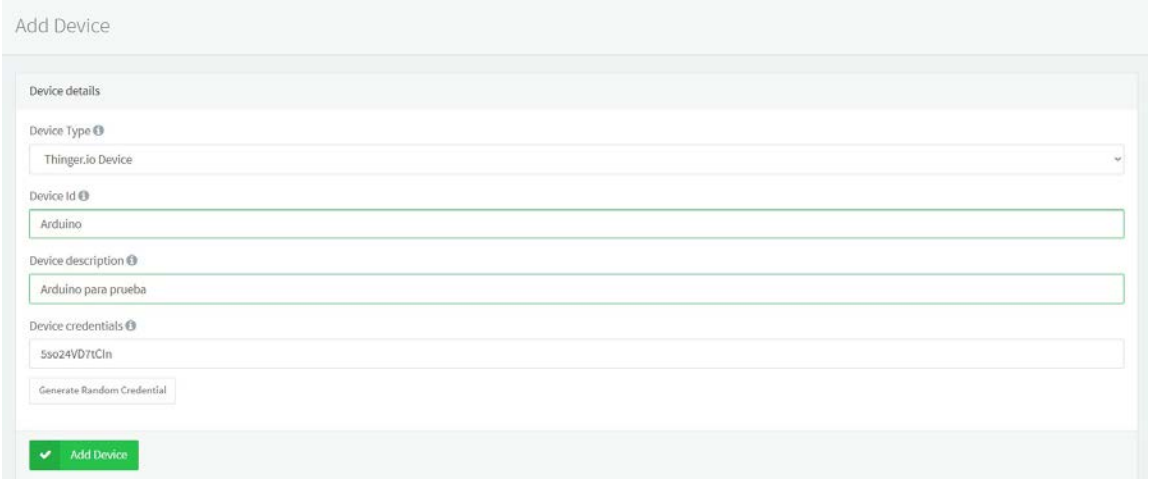


Figura 13.46 – Añadir un dispositivo

Añadiendo el dispositivo se obtendría la siguiente lista que se puede ver en la Figura 13.47. Como se puede ver, es posible ver que dispositivos están conectados.

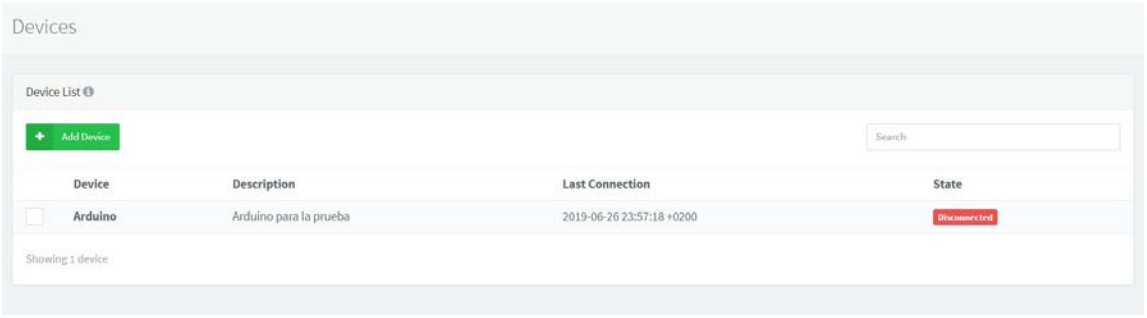


Figura 13.47 – Lista de dispositivos

Seleccionando el dispositivo se abre la siguiente ventana, en la que se pueden ver diferentes parámetros.

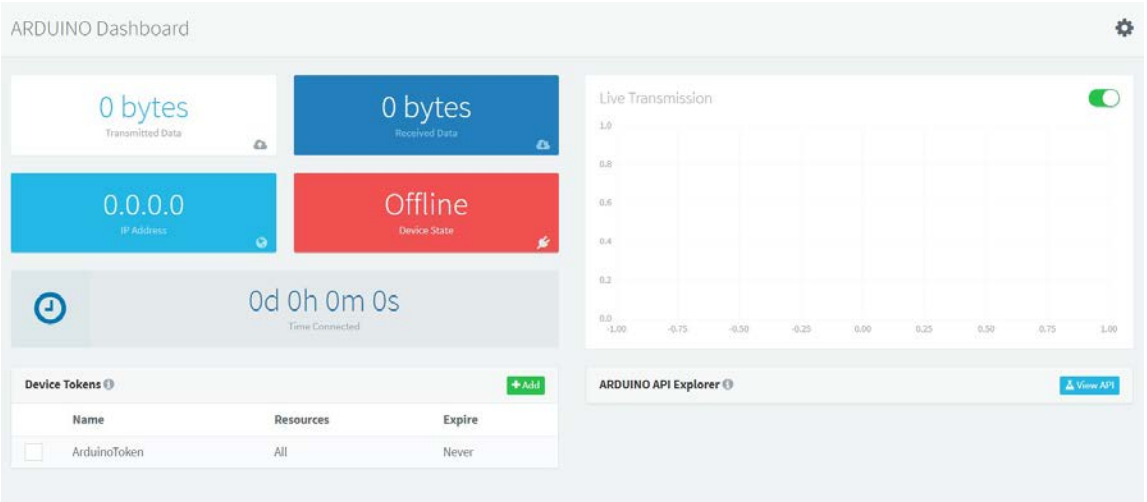


Figura 13.48 – Parámetros del dispositivo configurado

Ahora, en el apartado “Dashboard” se crea una nueva interfaz clicando en “Add dashboard”.

Figura 13.49 – Creación de una interfaz

Esta interfaz se puede configurar a voluntad, introduciendo mapas, gráficas, botones, etc.

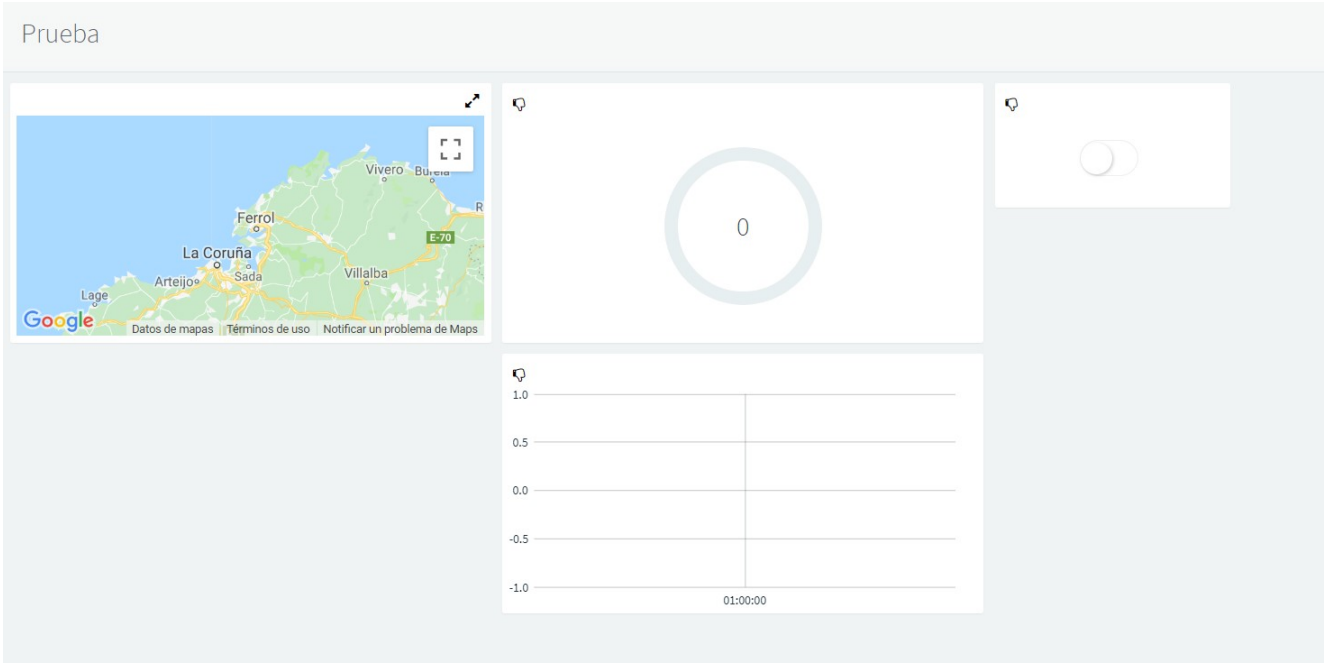


Figura 13.50 – Interfaz

13.5. ThingsBoard

13.5.1. Thingsboard como servidor en la nube

En primer lugar, es necesario registrarse en la web https://demo.thingsboard.io/login.

Una vez registrado, se crea el dispositivo. Para ello se va al apartado de “Dispositivos”, que aparece en el menú lateral y se añade un nuevo dispositivo clicando en el símbolo + rojo situado en la esquina inferior derecha.

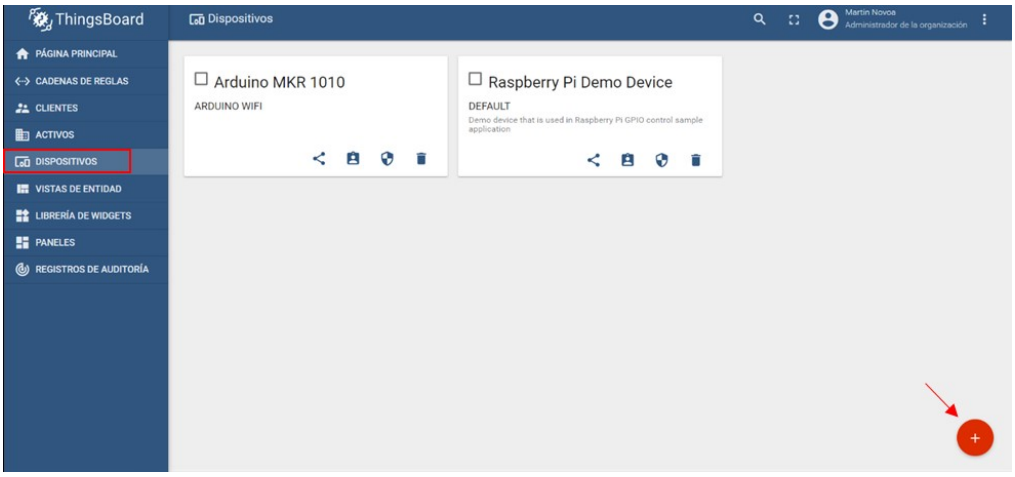


Figura 13.51 – Dispositivos I

De este modo, se abre una ventana en la que es necesario introducir el nombre y tipo de dispositivo.

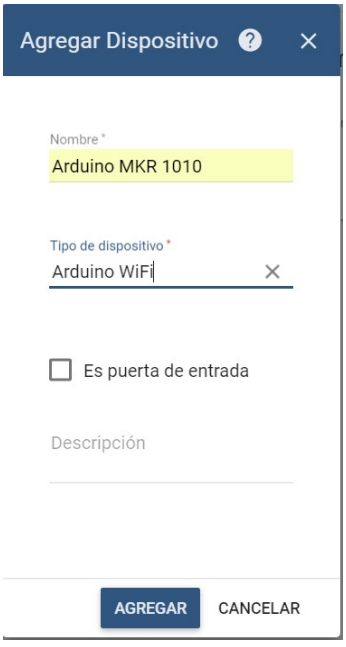


Figura 13.52 – Agregar dispositivo

Dándole a “Agregar” se nos añade a la lista dispositivos.

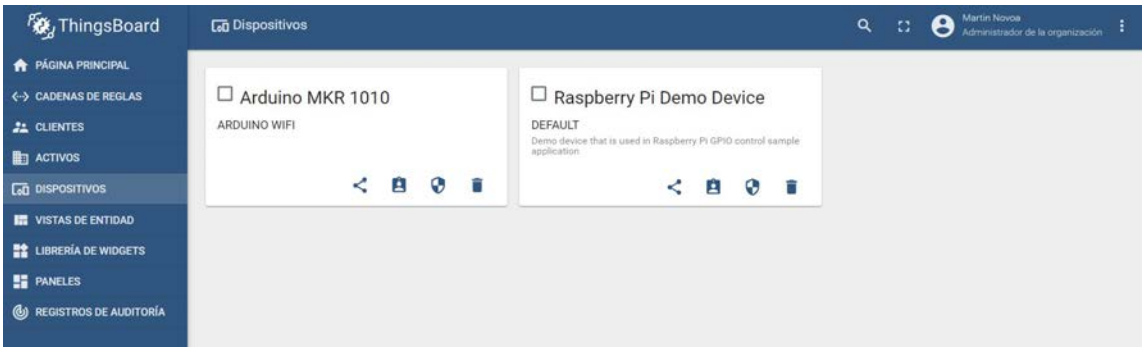


Figura 13.53 – Dispositivos II

Ahora, se copiará el token de acceso del dispositivo, en la sección de “Gestionar credenciales” del dispositivo, ya que se utilizará en la programación del Arduino.

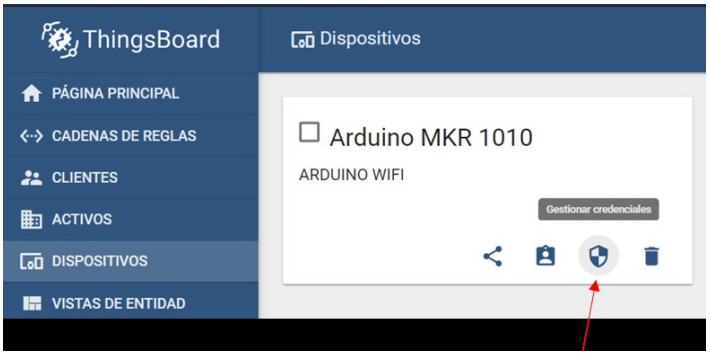


Figura 13.54 – Gestionar credenciales

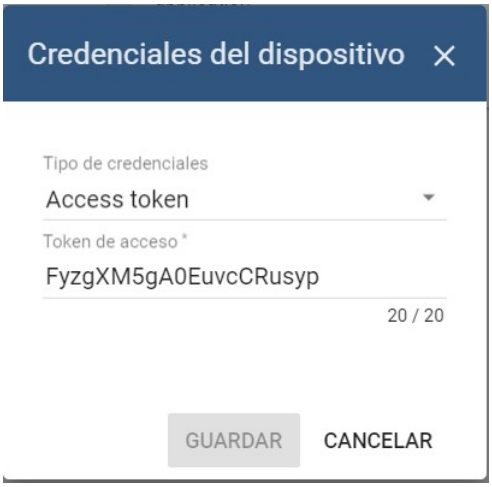


Figura 13.55 – Token de acceso

Por último, se creará la interfaz gráfica, en la que se añadirán los widgets deseados. Para ello, se abre la sección “Paneles” del menú lateral y se clica en “Crear nuevo panel” en la esquina inferior derecha (símbolo +). En la ventana que se abre se introduce el nombre que se desee poner.

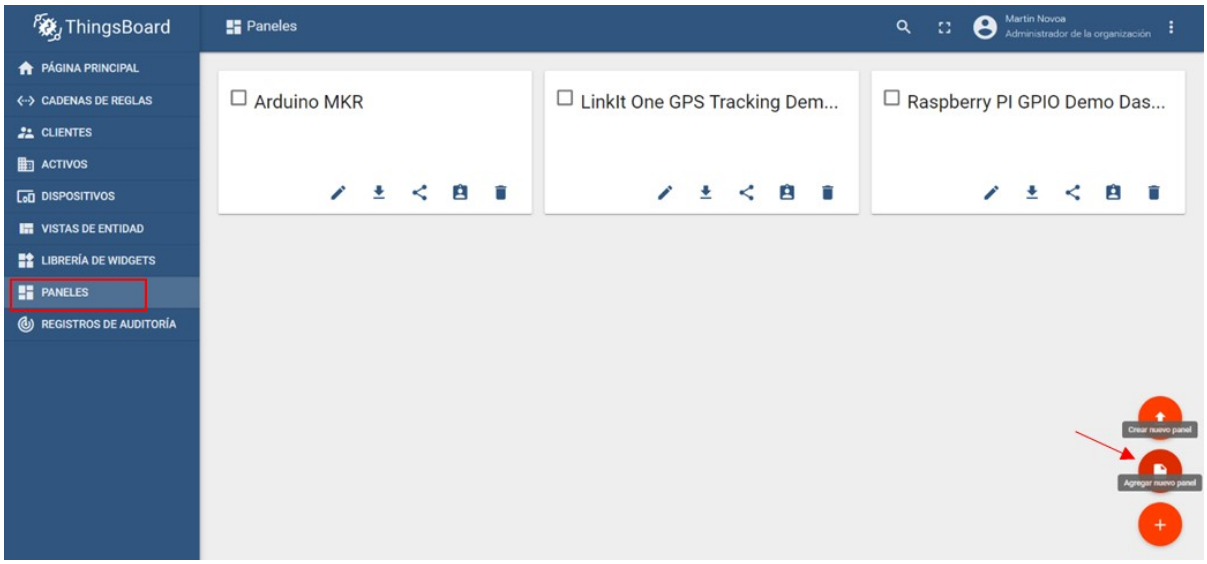


Figura 13.56 – Añadir nuevo panel

Una vez que el Arduino comienza a mandar datos, el sistema detecta automáticamente las variables y permite seleccionarlas en los diferentes widgets. En este caso, hemos realizado el siguiente panel para hacer la prueba:

Figura 13.57 – Interfaz gráfica

Ahora, clicando arriba a la derecha, en el símbolo del reloj, se puede seleccionar diferentes intervalos de tiempo: verlo en tiempo real, ver las últimas n medidas o ver un periodo de tiempo determinado por el usuario.

Figura 13.58 – Configuración de la visualización de los datos I

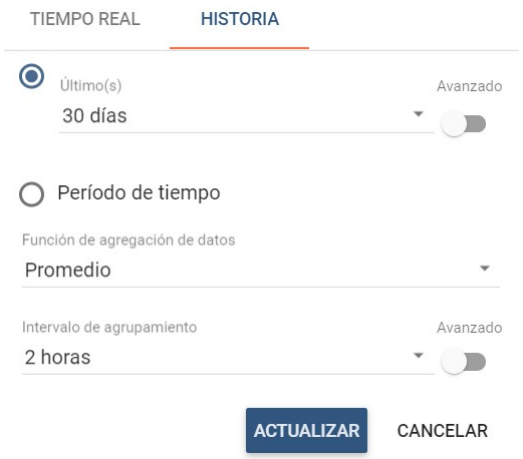


Figura 13.59 – Configuración de la visualización de los datos II

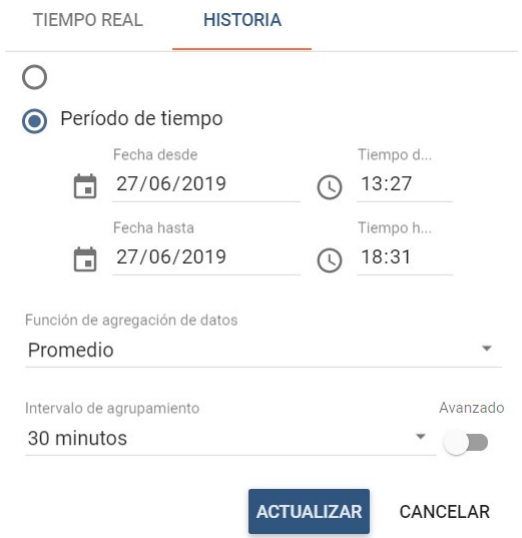


Figura 13.60 – Configuración de la visualización de los datos III

13.5.2. ThingsBoard como servidor local

13.5.2.1. Java OpenJDK

En primer lugar, es necesario descargar la version de java OpenJDK 8 HotSpot en el link: https://adoptopenjdk.net/index.html. Una vez descargado, se procede a su instalación, asegurandose de marcar las opciones de “Añadir al PATH” y “Establecer la variable JAVA_HOME”

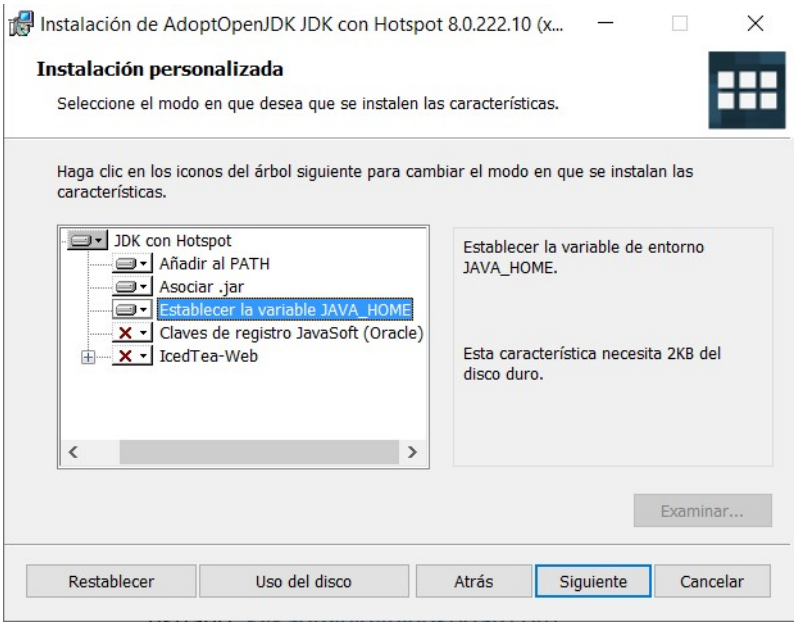


Figura 13.61 – Configuración de la instalación de Java OpenJDK

Una vez instalado, comprobamos que se haya instalado correctamente. Para ello abrimos la aplicación Símbolo del sistema e introducimos el comando “java -version”. Este comando debe devolver la siguiente información:

```
C:\Users\marti>java -version
openjdk version "1.8.0_222"
OpenJDK Runtime Environment (AdoptOpenJDK)(build 1.8.0_222-b10)
OpenJDK 64-Bit Server VM (AdoptOpenJDK)(build 25.222-b10, mixed mode)
```

Figura 13.62 – Conprobación de la versión de java

13.5.2.2. Base de datos (PostgreSQL)

Para el almacenamiento de los datos ThingsBoard recomienda dos posibilidades: -PostgreSQL para soluciones en las que va a haber menos de 5000 mensajes por segundo. -PostgreSQL+Cassandra para soluciones en las que va a haber más de 5000 mensajes por segundo. En este caso, como el periodo del envío de los datos va a ser de 1 minuto, se utilizará PostgreSQL. El enlace de descarga es: https://www.enterprisedb.com/downloads/postgres-postgresql-downloads#windows

Durante la instalación, el programa pide la contraseña para acceder a la base de datos. En este caso, se introdujo la contraseña “root”.

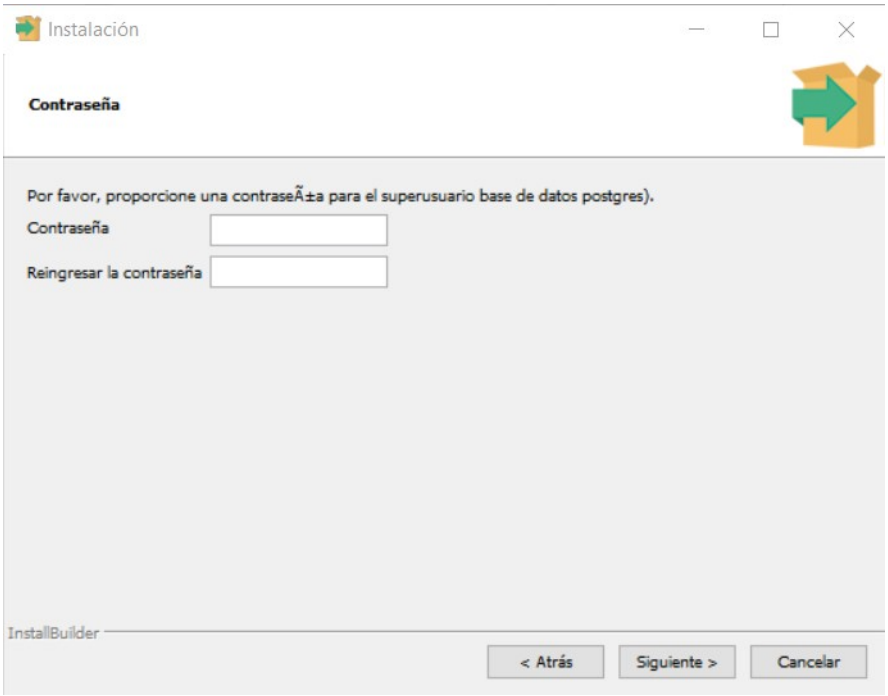


Figura 13.63 – Introducción de la contraseña en la instalación de la base de datos

El puerto se deja el 5432, ya que es el puerto que normalmente está relacionado a PostgreSQL.

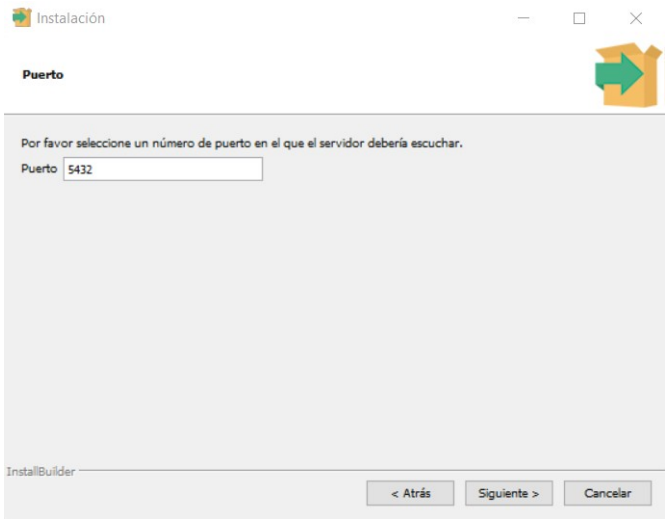


Figura 13.64 – Configuración del puerto en la instalación de la base de datos

Ahora se abre el programa “pgAdmin 4” y se introduce la contraseña.

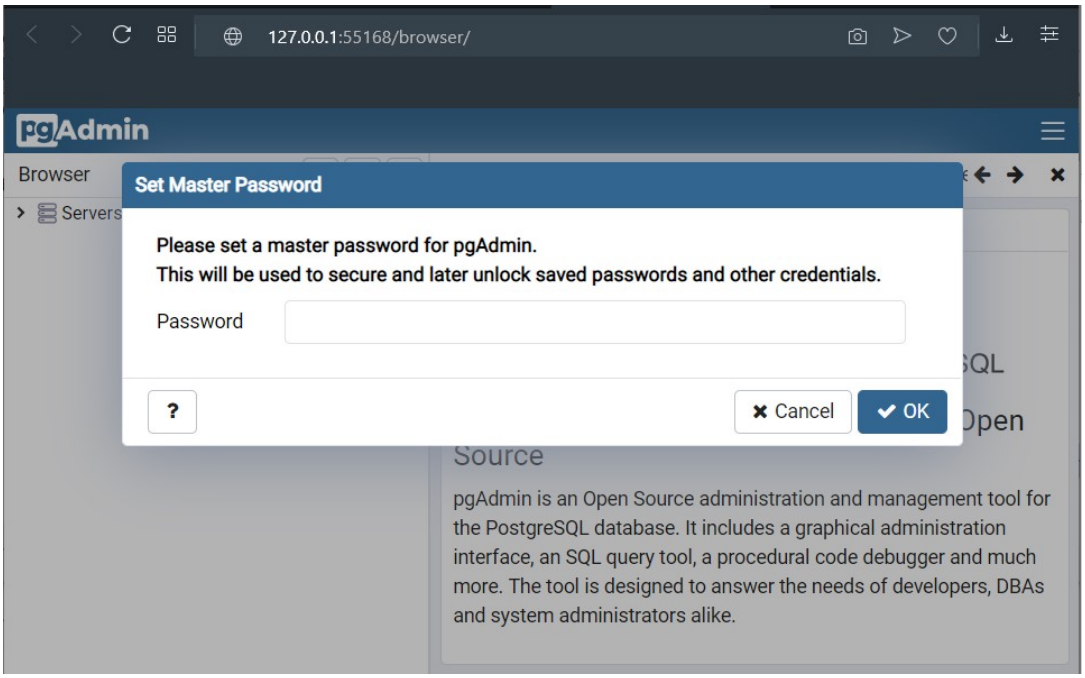


Figura 13.65 – Introducción de la contraseña de la base de datos

Ahora se crea la base de datos “thingsboard” haciendo clic derecho en “Databases” y “Create Database”.

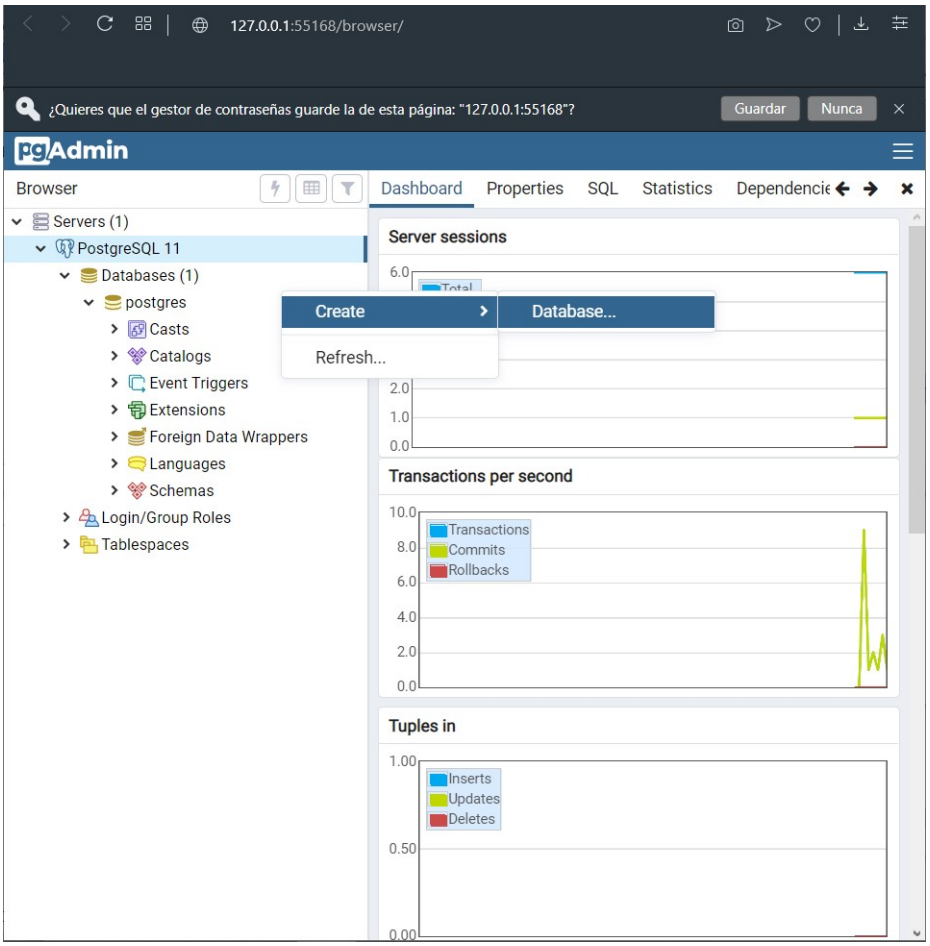


Figura 13.66 – Creación de una base de datos I

Se nombra como “thingsboard” se establece a “postgres” como el dueño, que es el usuario que se crea por defecto.

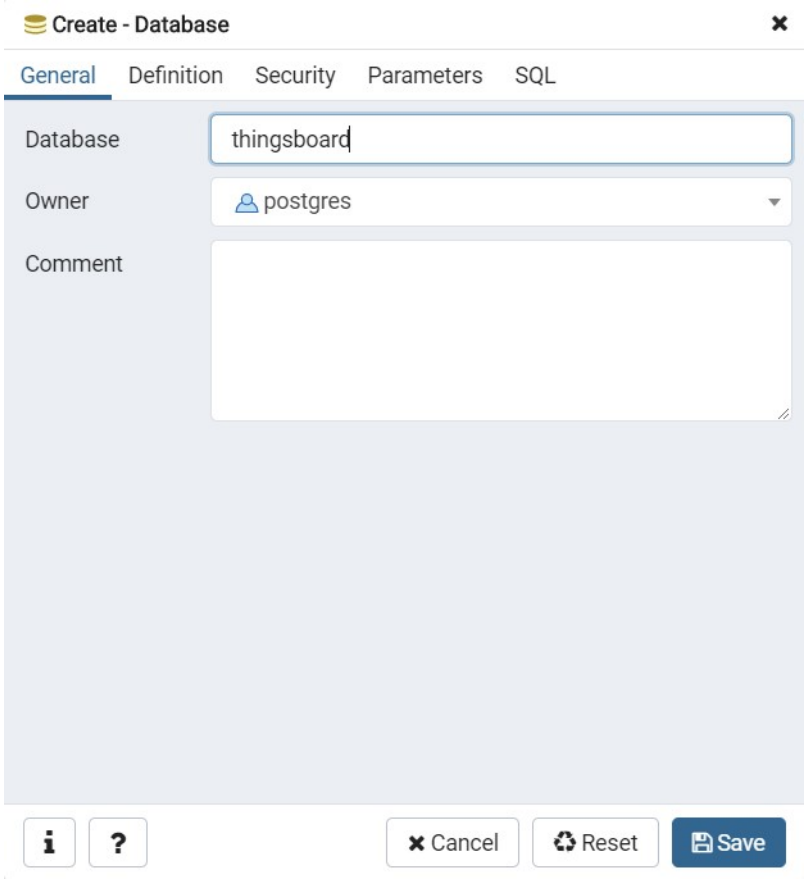


Figura 13.67 – Creación de una base de datos II

13.5.2.3. Thingsboard

Ahora se procede a descargar la plataforma de ThingsBoard. Esta, se puede descargar del siguiente enlace: https://github.com/thingsboard/thingsboard/releases/download/v2.4/thingsboard-windows-2.4.zip

Una vez descargado el paquete se descomprime en la carpeta “Archivos de programa (x86)”, en caso de un sistema de 64 bits, o “Archivos de programa”, para uno de 32 bits, del disco C del ordenador. Una vez hecho esto se abre y el archivo “thingsboard.yml” almacenado en la carpeta “conf”, con el bloc de notas. En este se busca el apartado “#SQL DAO Configuration” y se modifica la contraseña.

```
# SQL DAO Configuration
spring:
  data:
    jpa:
      repositories:
        enabled: "true"
  jpa:
    hibernate:
      ddl-auto: "none"
    database-platform: "${SPRING_JPA_DATABASE_PLATFORM:org.hibernate.dialect.PostgreSQLDialect}"
  datasource:
    driverClassName: "${SPRING_DRIVER_CLASS_NAME:org.postgresql.Driver}"
    url: "${SPRING_DATASOURCE_URL:jdbc:postgresql://localhost:5432/thingsboard}"
    username: "${SPRING_DATASOURCE_USERNAME:postgres}"
    password: "${SPRING_DATASOURCE_PASSWORD:postgres}"
```

Figura 13.68 – Modificación de la contraseña del servidor

Por defecto viene “postgres”, pero se introducirá la contraseña introducida previamente en la base de datos. Esto es debido a que Thingsboard se comunica con el servidor creado por PostgreSQL para enviar los datos. Ahora se procede a la instalación de Thingsboard. Para ello, en la carpeta principal, se ejecuta “install.bat” Una vez termina, abrimos el programa “Símbolo del sistema” como administrador y ejecutamos la línea “net start thingsboard”

```
Microsoft Windows [Versión 10.0.18362.239]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>net start thingsboard
El servicio de ThingsBoard Server Application está iniciándose.
El servicio de ThingsBoard Server Application se ha iniciado correctamente.
```

Figura 13.69 – Inicialización del servidor

Cabe la posibilidad de que, en este punto, la aplicación no se inicie. En ese caso es necesario ir al archivo thingsboard.xml y eliminar la siguiente línea:

$$< startargument > -XX : +UseCondCardMark < /startargument >$$

Figura 13.70 – Eliminación de la linea de código, en caso de error

Como se va a acceder al servicio desde diferentes ordenadores conectados a la red local, es necesario abrir el puerto 808 en el firewall de Windows, ya que es en el que se va a ubicar Thingsboard.

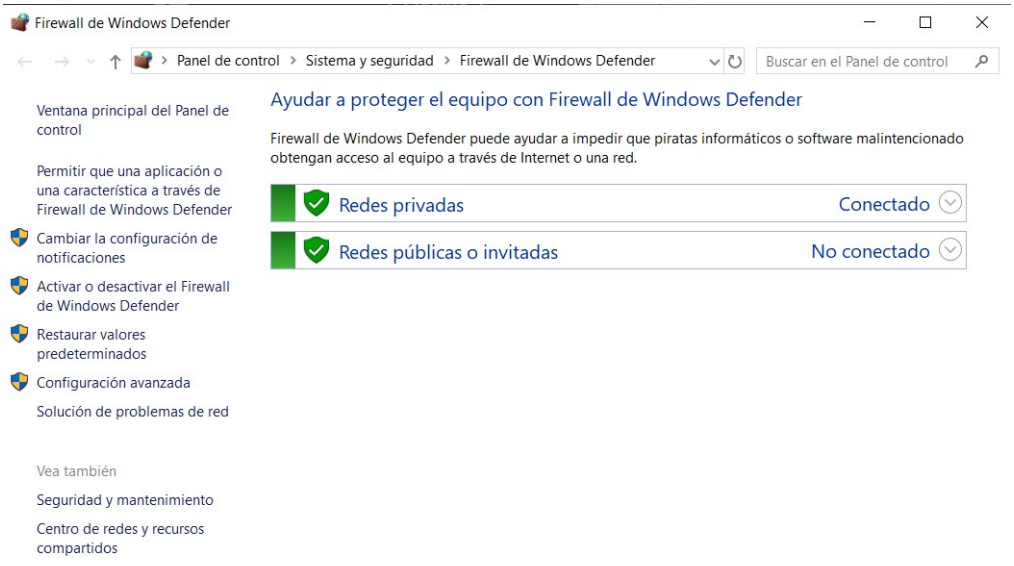


Figura 13.71 – Firewall de Windows

Ahora en Configuración avanzada → Reglas de entrada se clica en “Nueva regla. . . ”

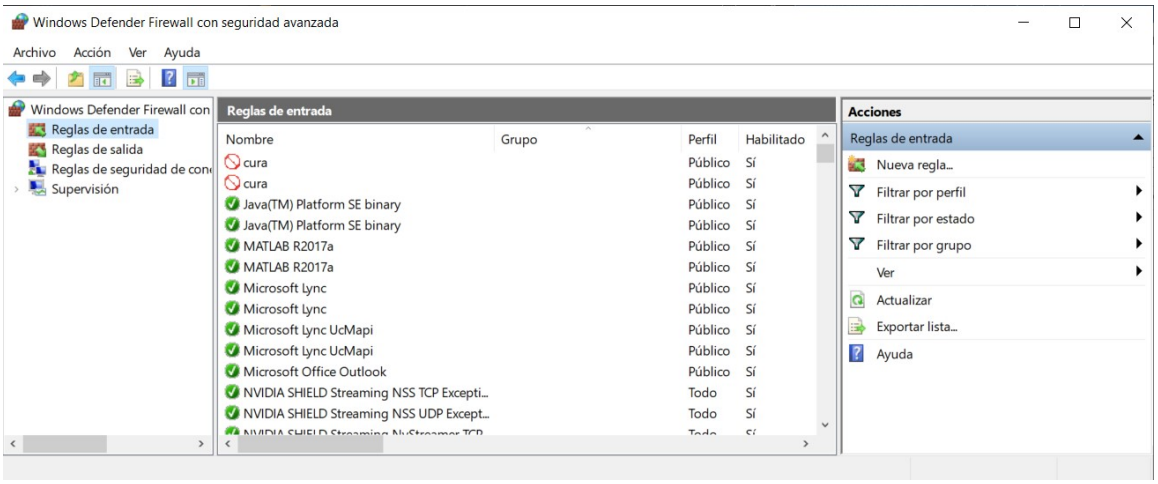


Figura 13.72 – Creación de una nueva regla

Ahora se establece el tipo de regla como “Puerto”.

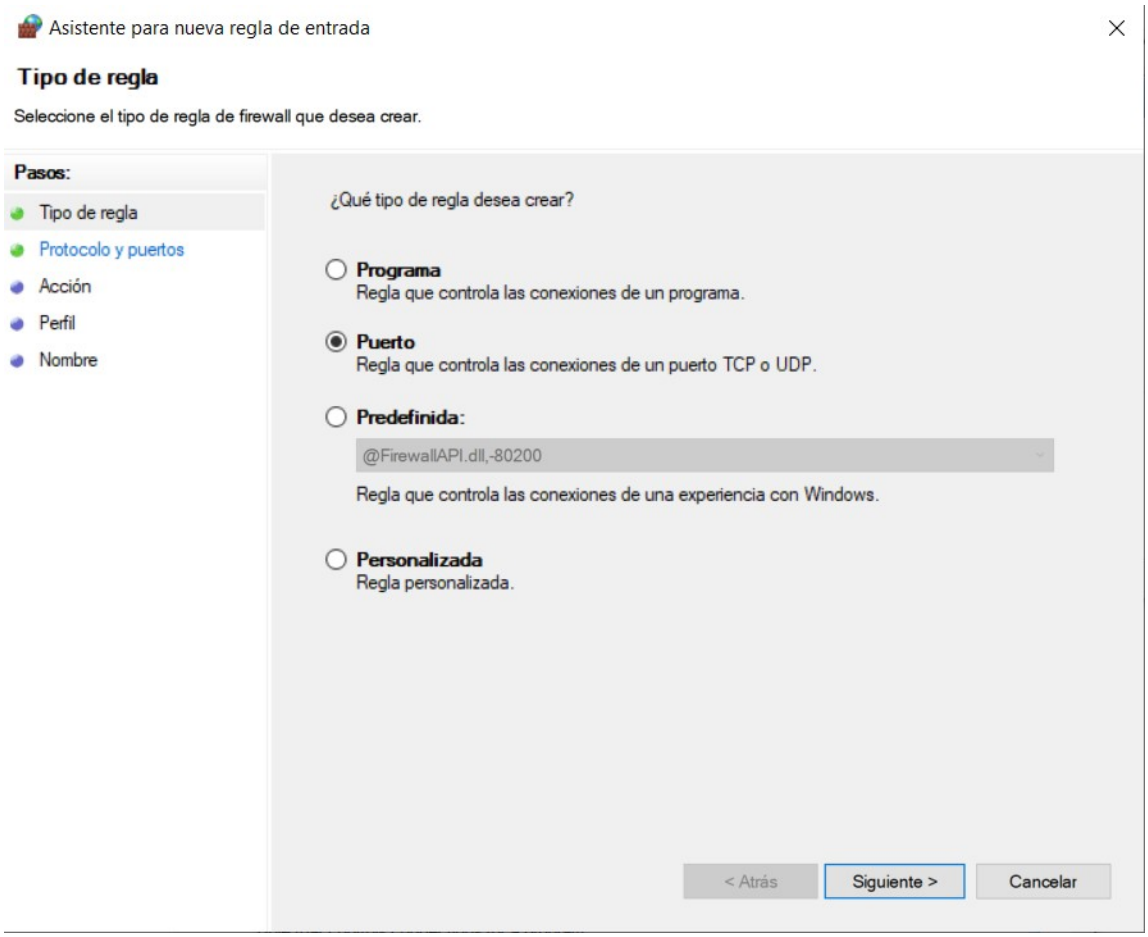


Figura 13.73 – Configuración del tipo de regla

Se abren los puertos 8080, 1883, 5683, que son los puertos recomendados por Thingsboard.

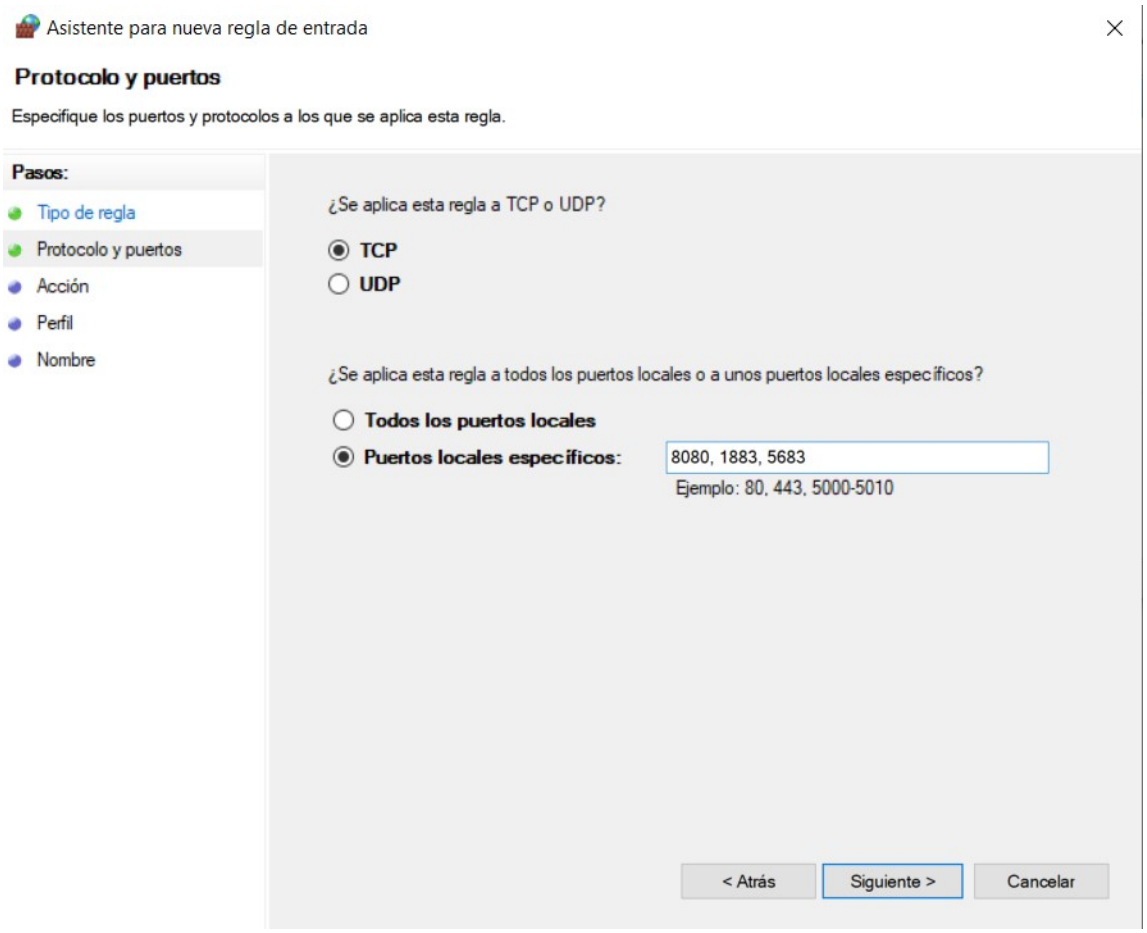


Figura 13.74 – Configuración de la visualización de los datos III

Ahora se define la accion como “Permitir la conexión”.

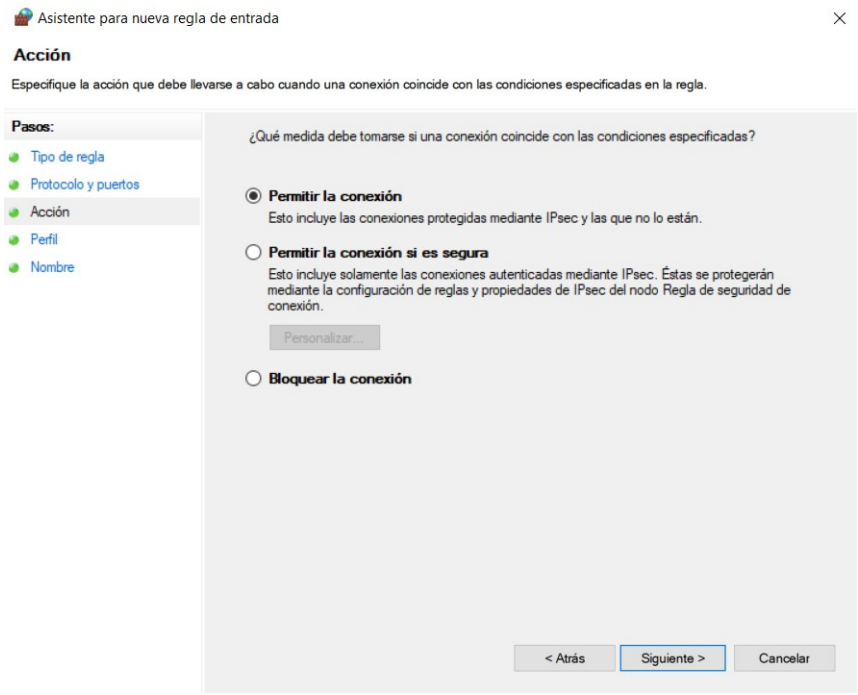


Figura 13.75 – Configuración del tipo de acción

La regla únicamente se define en las redes privadas, para que el servidor solo funcione en las redes que el usuario desee.

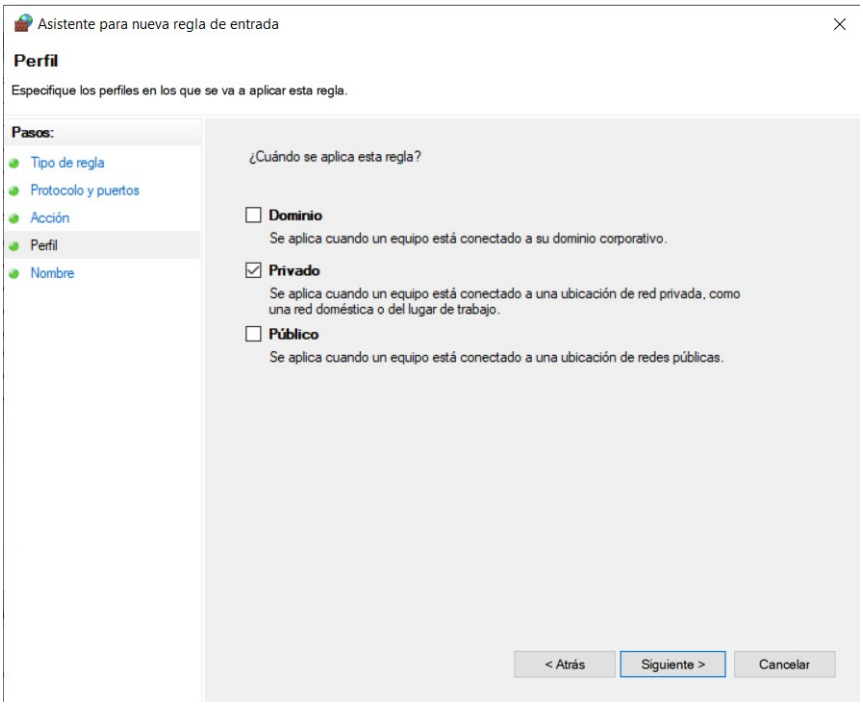


Figura 13.76 – Configuración de las redes en las que se aplica la regla

Por último, el nombre de la regla será “Thingsboard Service Networking”.

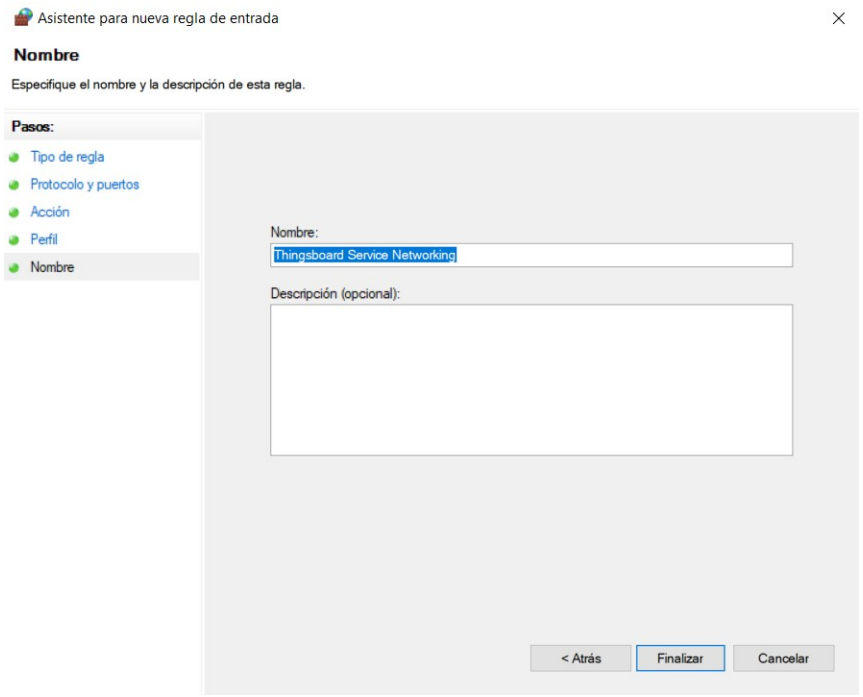


Figura 13.77 – Configuración del nombre de la regla

13.5.3. Compartir la base de datos en la red local

13.5.3.1. Habilitar el uso compartido en el servidor

Para compartir la base de datos en la red local, se va a la carpeta de PostgreSQL en “Archivos de programa” y a la carpeta “data”. En esta carpeta se busca el archivo “postgresql.config” y se abre con el bloc de notas, asegurandose de que en “listen_adress” tenga definido el asterisco.

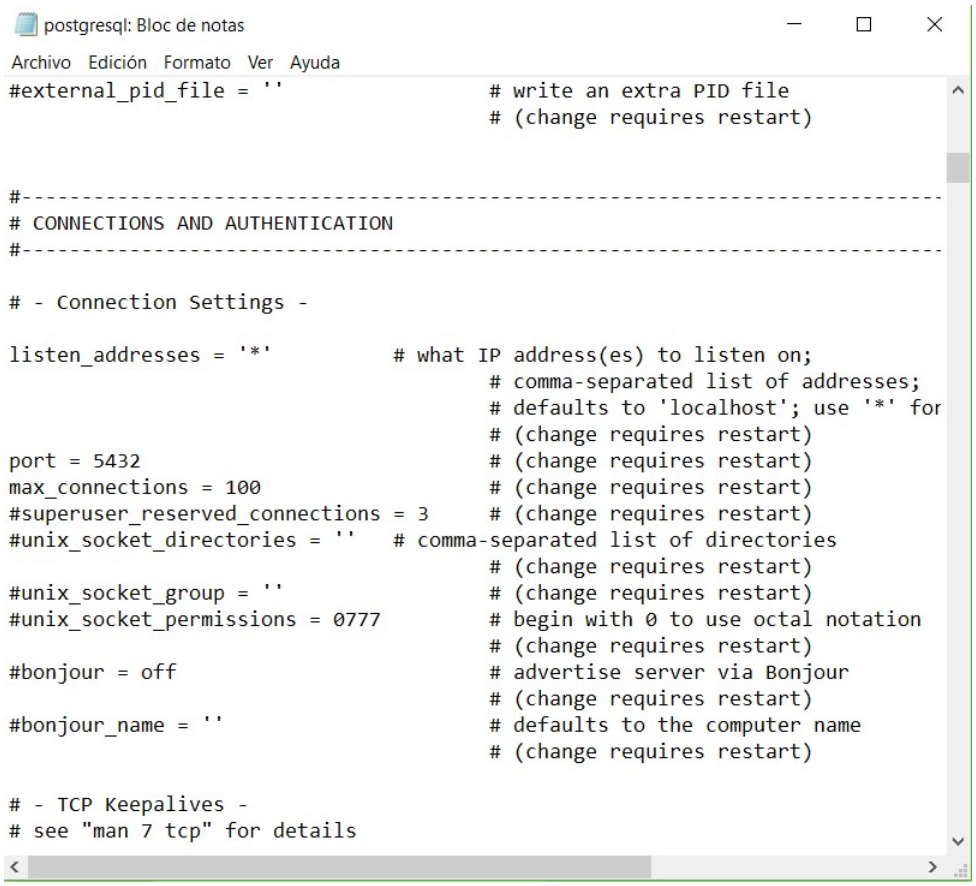


Figura 13.78 – Modificación de “postgresql.config”

Ahora, en el archivo que se encuentra en la misma carpeta, denominado “pg_hba.conf” y se introduce, en la parte inferior del documento, la siguiente línea de código:

$$host \quad all \quad all \quad 0{,}0{,}0{,}0/0 \quad md5$$

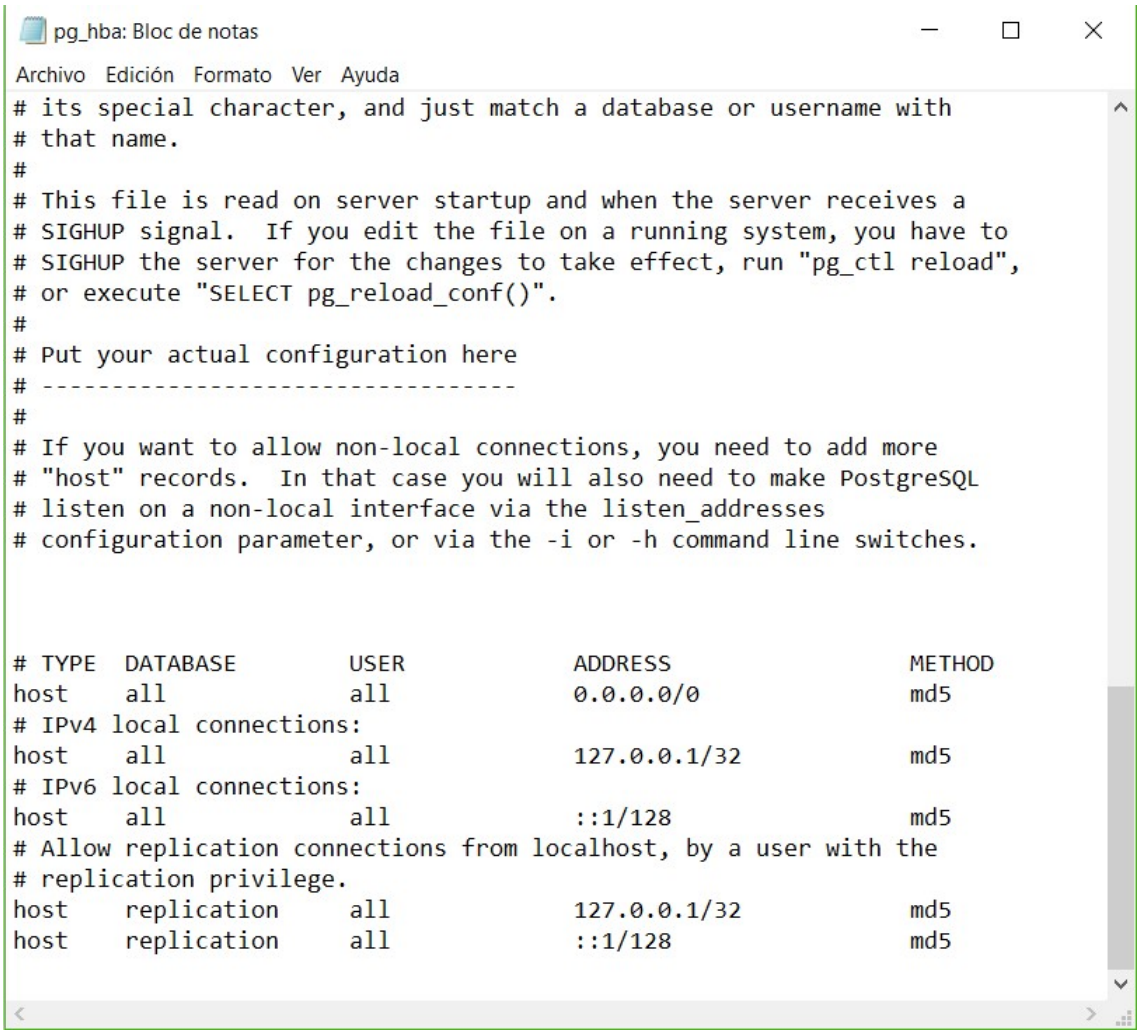


Figura 13.79 – Configuración de las direcciones IP que pueden acceder al servidor

Ahora se guarda el documento y se reinicia la base de datos.

13.5.3.2. Acceso a la base de datos desde el cliente

Para acceder a la base de datos desde otro dispositivo se debe abrir el programa PostgreSQL. Una vez abierto se hace clic derecho y se crea un nuevo servidor.

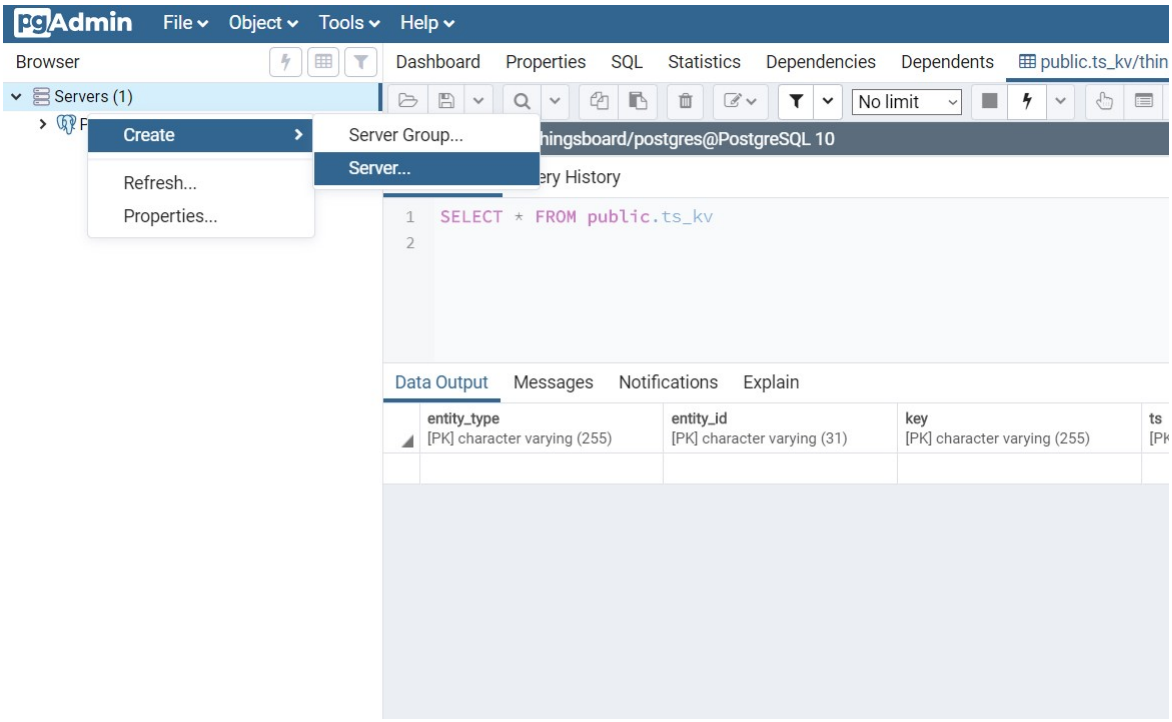


Figura 13.80 – Acceso a una base de datos externa I

Ahora se le otorga un nombre.

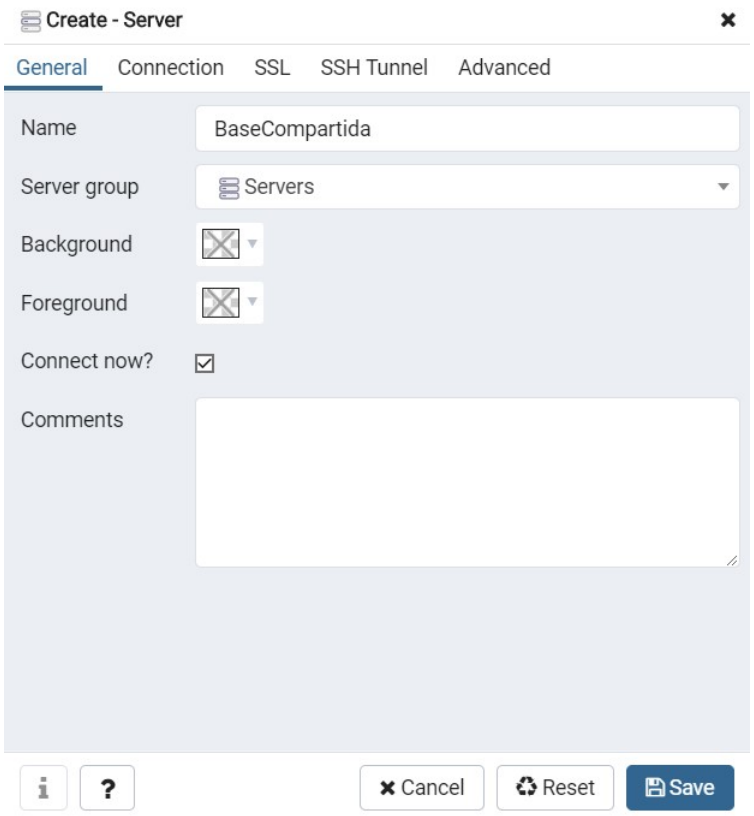


Figura 13.81 – Acceso a una base de datos externa II

En el apartado de “Connection”se introducen los siguientes parametros:

- Host name/address: Dirección IP del dispositivo que contiene la base de datos. Es la que aparece en el navegador cuando se abre el pgAdmin4.

- Port: Puerto del *host* asociado a la base de datos.

- Username: Nombre de usuario de la base de datos (Normalmente “postgres”).

- Password: Contraseña de la base de datos.

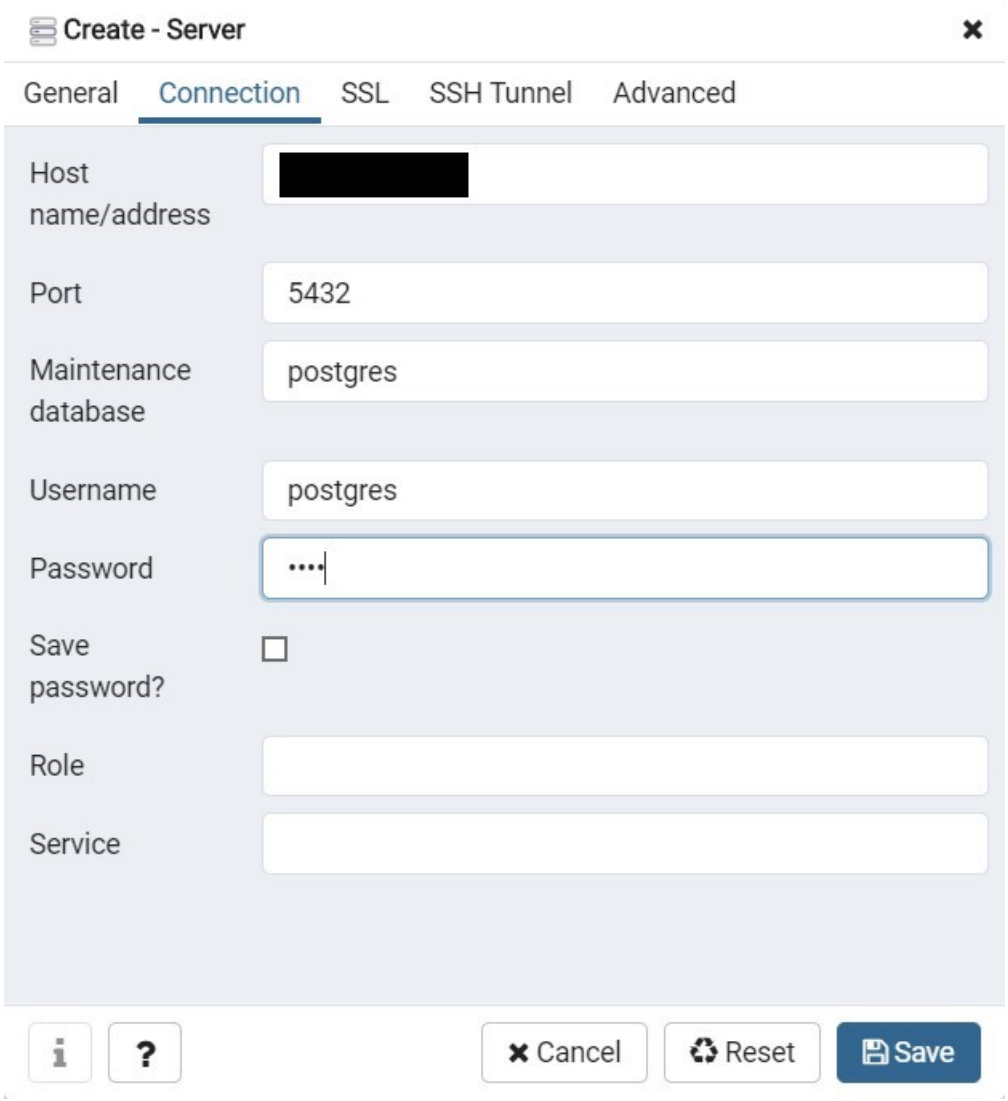


Figura 13.82 – Acceso a una base de datos externa III

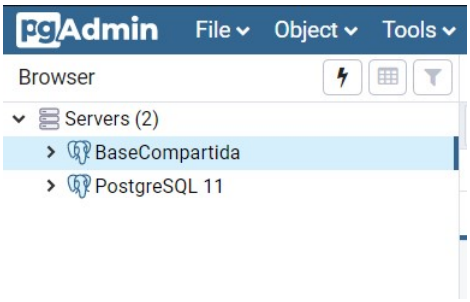


Figura 13.83 – Configuración de la visualización de los datos III

13.5.4. Configuración de Thingsboard

Por defecto el usuario administrador de Thingsboard es:

- usuario: sysadmin@thingsboard.org
- contraseña: sysadmin

Cuando se inicia sesión se carga la siguiente ventana. El usuario administrador no puede añadir dispositivos ni paneles. Para ello hay que crear una organización.

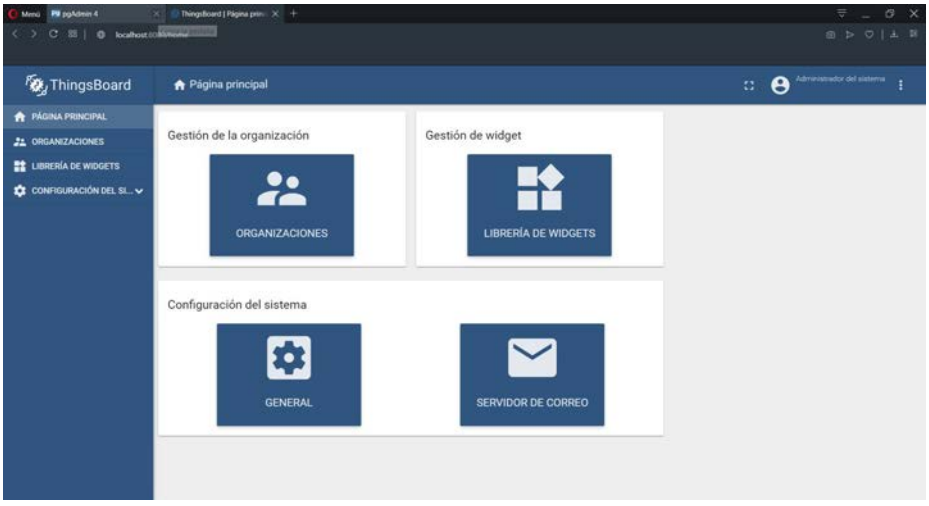


Figura 13.84 – Interfaz del usuario “sysadmin”

Para ello, se abre el apartado “Organizaciones” y se clica en el símbolo + de abajo a la derecha. Ahora se introduce el nombre de la organización y los datos que se quieran añadir a mayores.

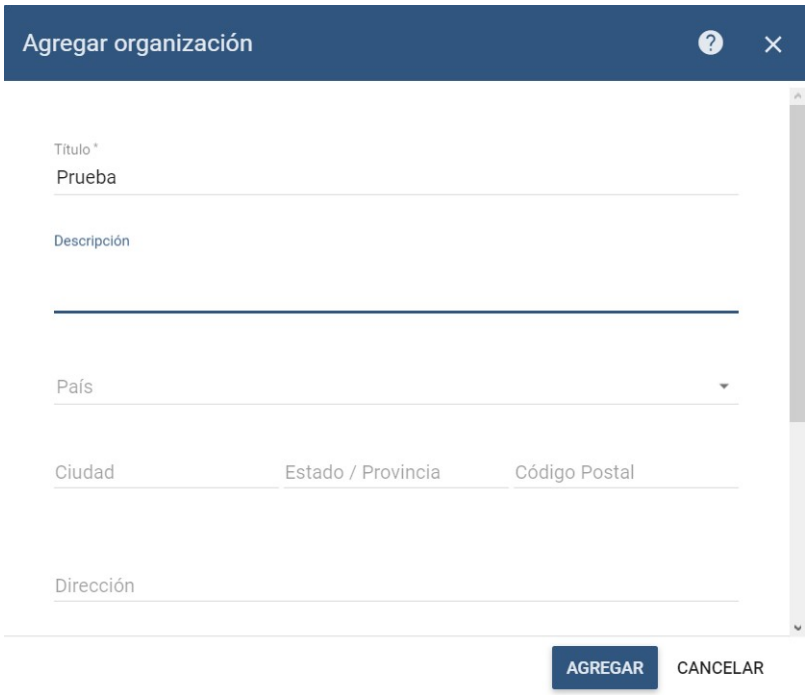


Figura 13.85 – Adición de un organicización

Ahora se agregan los administradores de la organización. Para ello se abre la organización y se clica en “Gestionar administradores de la organización.

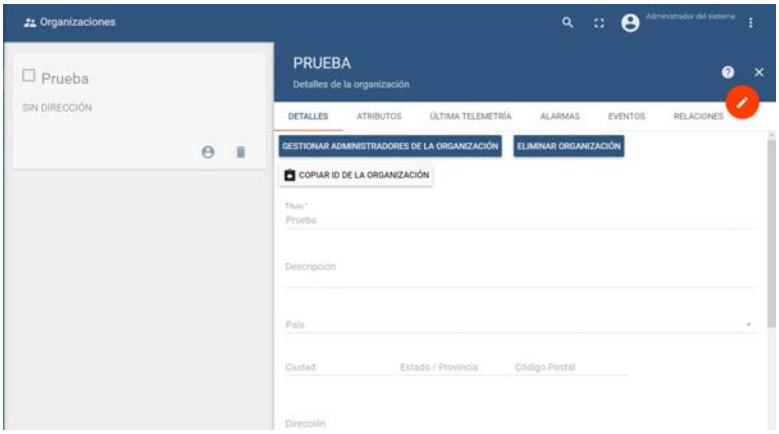


Figura 13.86 – Detalles de la organización

Ahora, se clica abajo a la derecha en el símbolo + y se introducen los datos correspondientes del administrador de la organización, en la siguiente ventana. Se deja el modo “Mostrar enlace de activación”.

Figura 13.87 – Agregar usuario

Figura 13.88 – Enlace de activación de usuario

Al abrir el enlace se puede introducir la contraseña de la cuenta.

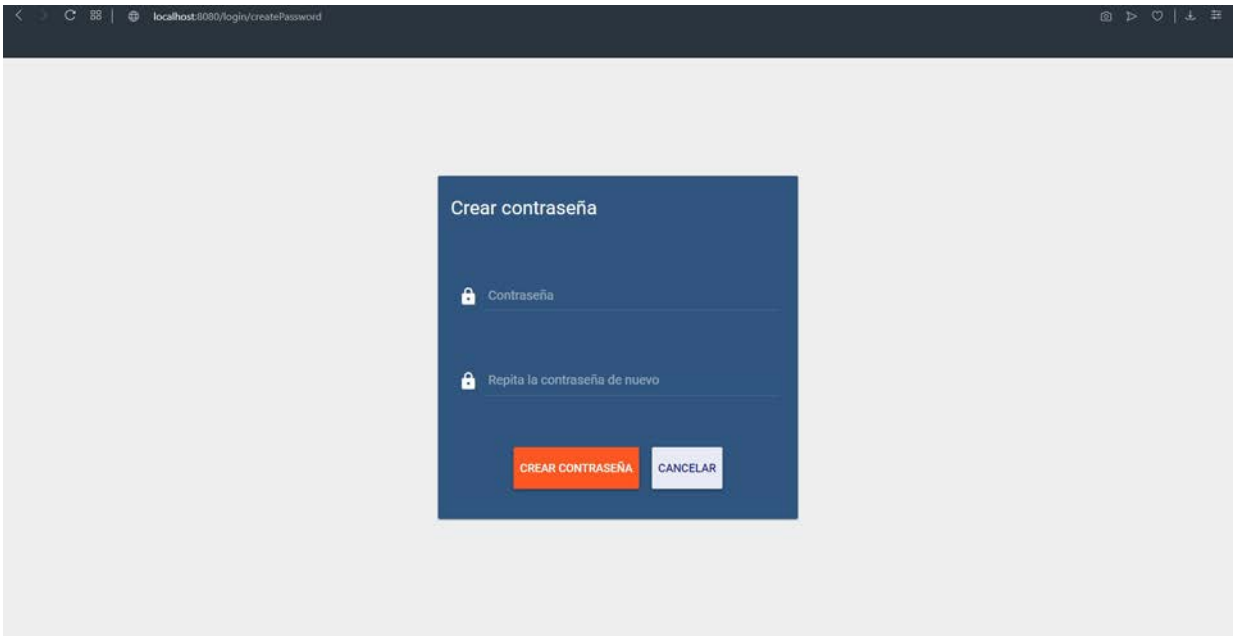


Figura 13.89 – Crear contraseña de usuario

Una vez introducido ya se puede acceder a todas las opciones que había en la versión Demo (Apartado 13.5.1). Y la configuración de los dispositivos y los paneles se haría del mismo modo.

Figura 13.90 – Interfaz del administrador de la organización

Ahora en el apartado de clientes se podrían configurar, de manera análoga a la configuración de la organización, clientes que podrían visualizar los paneles. Estos clientes verían la siguiente página principal:

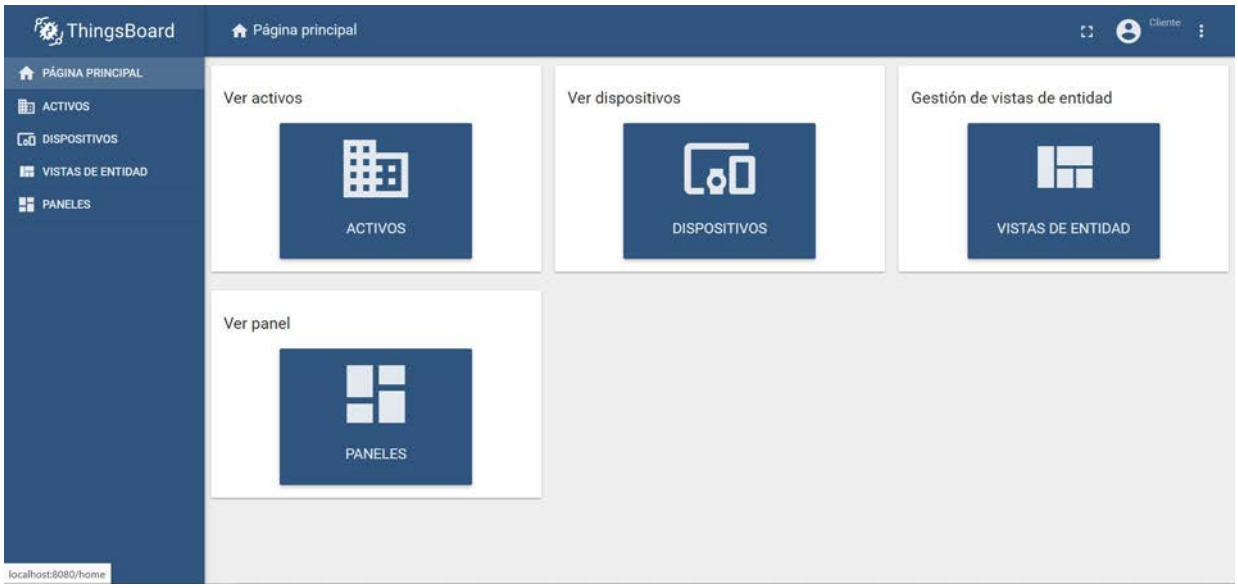


Figura 13.91 – Interaz de los clientes

Como se puede ver en la Figura 13.91, pueden ver los dispositivos y paneles, pero no tienen permisos para añadir nuevos o modificar los ya creados.

13.5.4.1. Exportado de los datos de la base de datos.

Para visualizar los datos almacenados en la base de datos se debe abrir el programa postgreSQL. Una vez abierto Para añadir la columna de tiempo a la tabla de la base de datos, se selecciona la base de datos “thingsboard” creada previamente y se clica en el simbolo marcado con el circulo rojo en la Figura 13.92

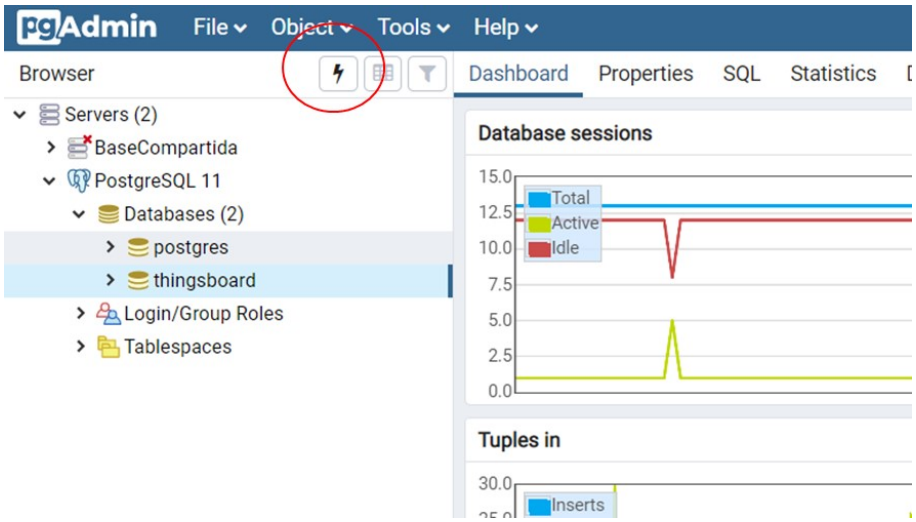


Figura 13.92 – Solicitud SQL

Para ver los datos se introduce en el siguiente código y se puelsa F5 para ejecutarlo:

```
SELECT * FROM public.ts_kv;
```

Sin embargo, la tabla que se crea por defecto no contien información del instante de tiempo en el que se reciben los datos. Para conocer este momento es necesario crear una nueva columna que lo contenga:

```
ALTER TABLE ts_kv ADD COLUMN created_at TIMESTAMP;
ALTER TABLE ts_kv ALTER COLUMN created_at SET DEFAULT now();
```

Ahora que la tabla contiene toda la información necesaria se puede proceder al exportado en formato .csv, pulsando F8 o clicando en el botón de descargar. Además es posible seleccionar el intervalo de tiempo cuyos datos son de interés para su visualización o exportado, mediante la siguiente solicitud:

```
SELECT * FROM public.ts_kv
WHERE (Created_at BETWEEN '2019-07-22 19:05:04' AND '2019-07-22 20:00:00')
ORDER BY Created_at;
```

En esta solicitud se substituirán las fechas de ejemplo, presentes en la solicitud, por las requeridas por el usuario.

13.6. Ubidots

En esta prueba se utilizará la versión de educación de la plataforma, ya que es la única que proporciona una licencia gratuita.

En primer lugar es necesario registrarse en la web: https://ubidots.com/education/

Para poner a funcionar el sistema únicamente hay que copiar el Token. Para ello se clica en el nombre de usuario, que aparece en la esquina superior derecha, y se abre la sección de "Credenciales del API".

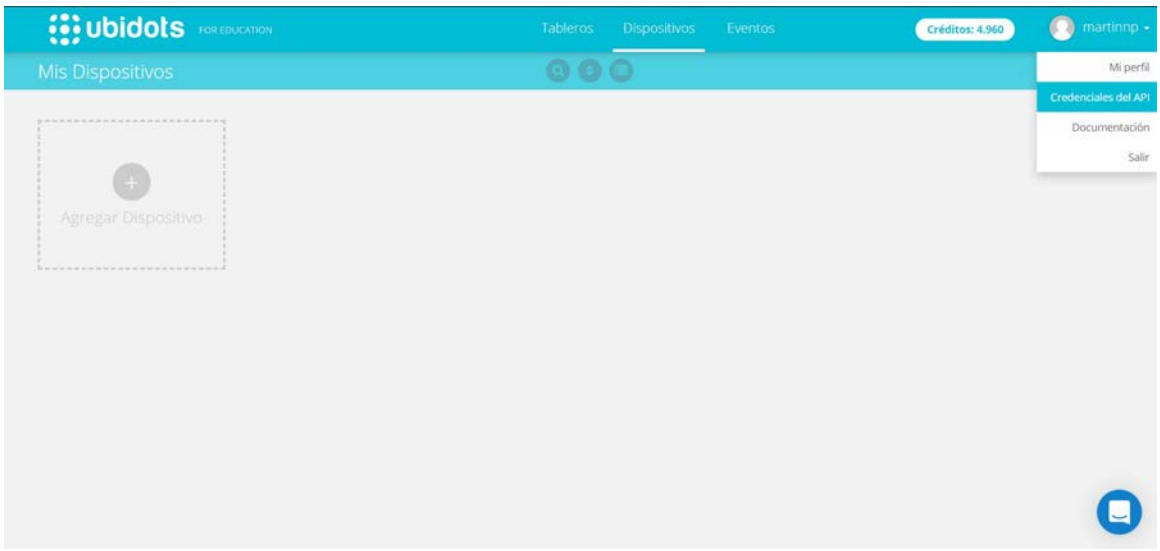


Figura 13.93 – Ubidots

Figura 13.94 – Credenciales del API

Este código se introducirá en la variable “TOKEN” del programa del Arduino y, al ser ejecutado, la plataforma reconoce el dispositivo automáticamente.

Figura 13.95 – Dispositivos

Clicando en el dispositivo podemos ver las variables.

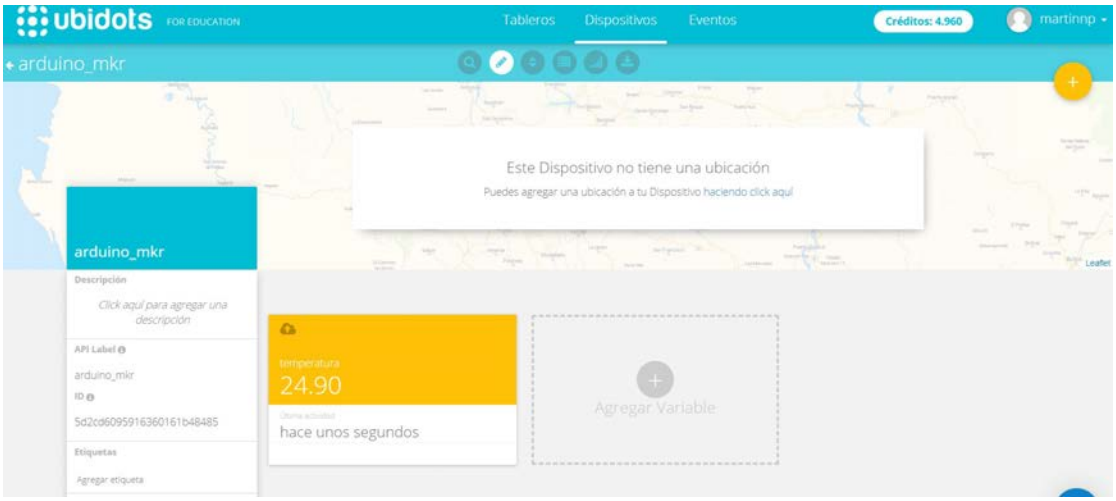


Figura 13.96 – Variables del dispositivo

Las graficas se crean en la pestaña de “Tableros”, clicando en “agregar un widget”.

Figura 13.97 – Agregar un widget

Ahora, se selecciona el widget entre las diferentes opciones.

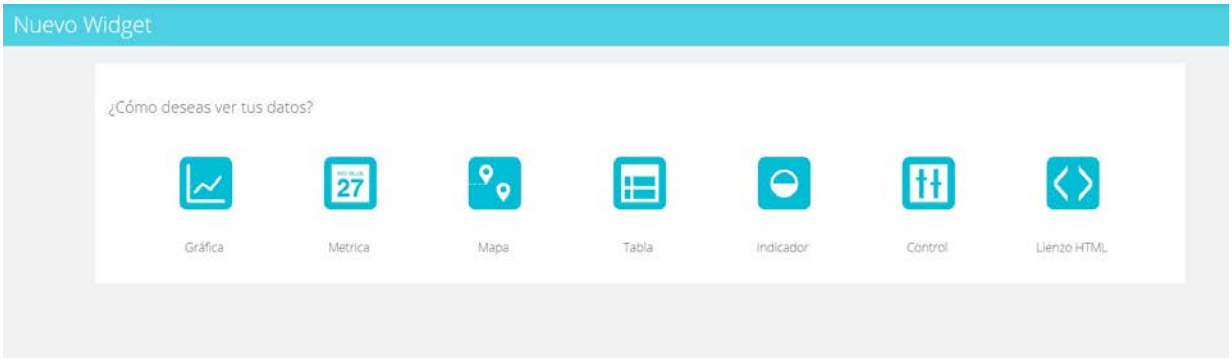


Figura 13.98 – Widgets

En este caso, se creará una gráfica de líneas.

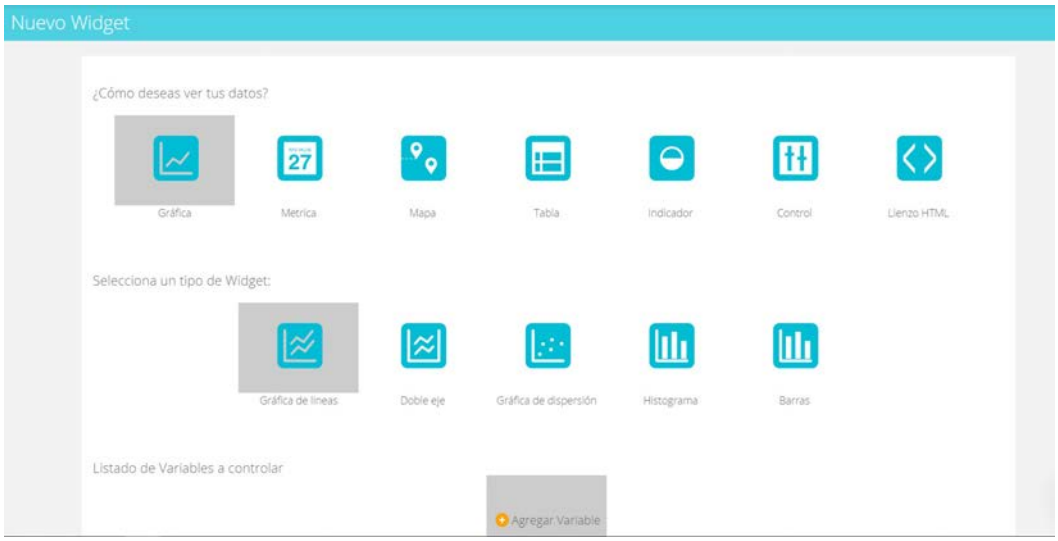


Figura 13.99 – Creación del widget

Clicando en “Agregar variable”, permite seleccionar entre las diferentes variables disponibles. En este caso, como únicamente se envía la temperatura, únicamente aparece esta.

Figura 13.100 – Selección de la variable

Como se puede ver en la Figura 13.101, se ha creado la gráfica correctamente.

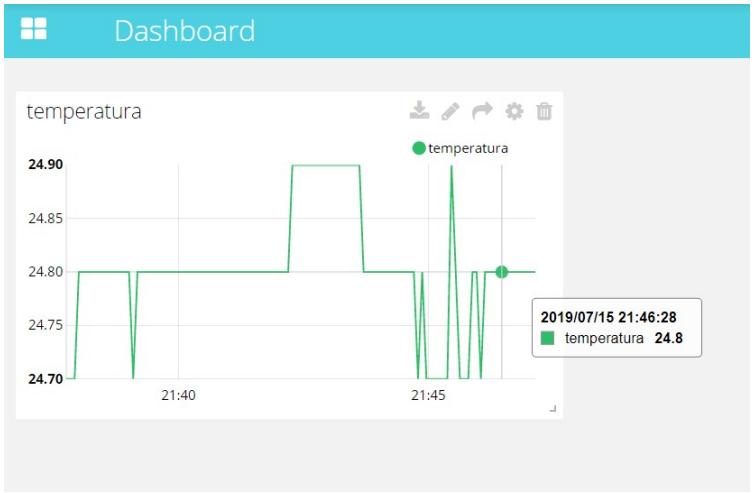


Figura 13.101 – Graficado Ubidots

Abriendo la configuración del widget se pueden configurar varias opciones de la gráfica, como el número de puntos a representar.

Figura 13.102 – Configuración de la gráfica

13.7. IBM

En primer lugar, es necesario registrarse en la web: `https://cloud.ibm.com`.

IBM funciona mediante lo que se denominan recursos. Estos, son las aplicaciones encargadas de realizar las diferentes tareas.

13.7.0.1. Internet of Things Platform

En primer lugar, se configurará la plataforma “Internet of Things Platform”, la cual se encarga de la comunicación con el dispositivo IoT. Para ello, se clica en “Crear recurso” y se selecciona en la categoria de “Internet de las cosas”.

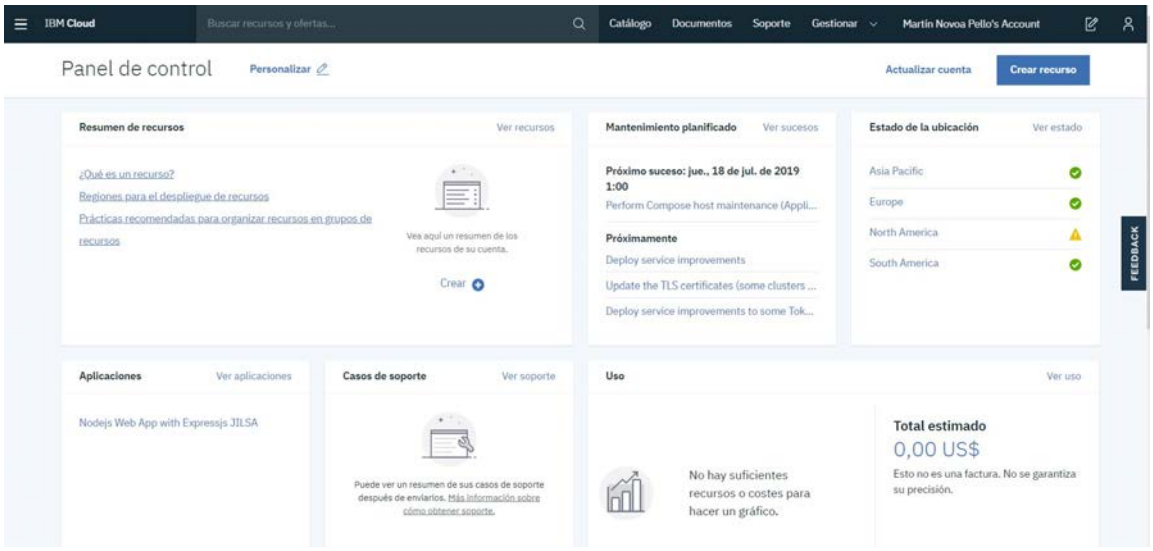


Figura 13.103 – Panel de control de IBM Cloud

Figura 13.104 – Selección del recurso “Internet of Thigs Platform”

Se le da nombre al servicio, dejando Londres como ubicación, ya que es la única disponible en la versión lite. El plan a elegir debe ser el lite.

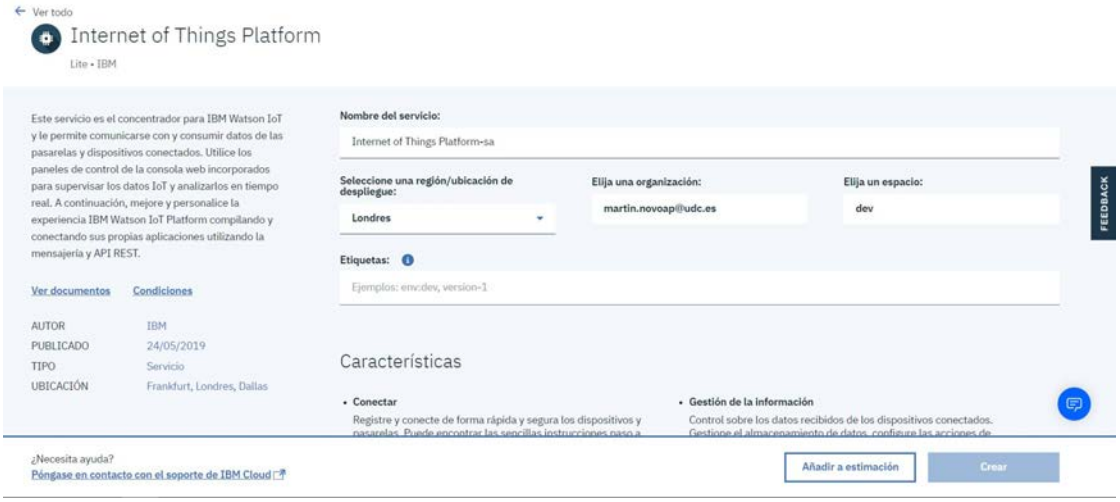


Figura 13.105 – Configuración del recurso “Internet of Thigs Platform”

En la ventana que se abre se clica en “Lanzar”.

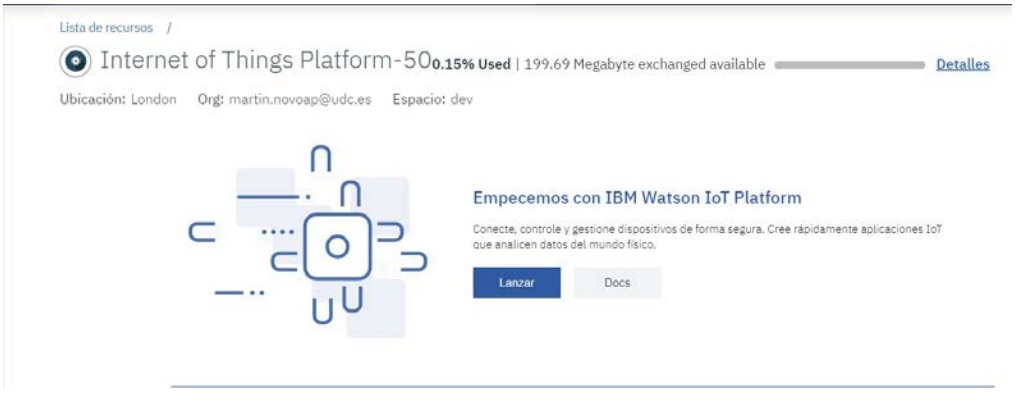


Figura 13.106 – Formulario de Google

De este modo, se abre el recurso. Para poder realizar la comunicación, es necesario configurar el dispositivo. Para ello, se clica en “Crear un dispositivo”

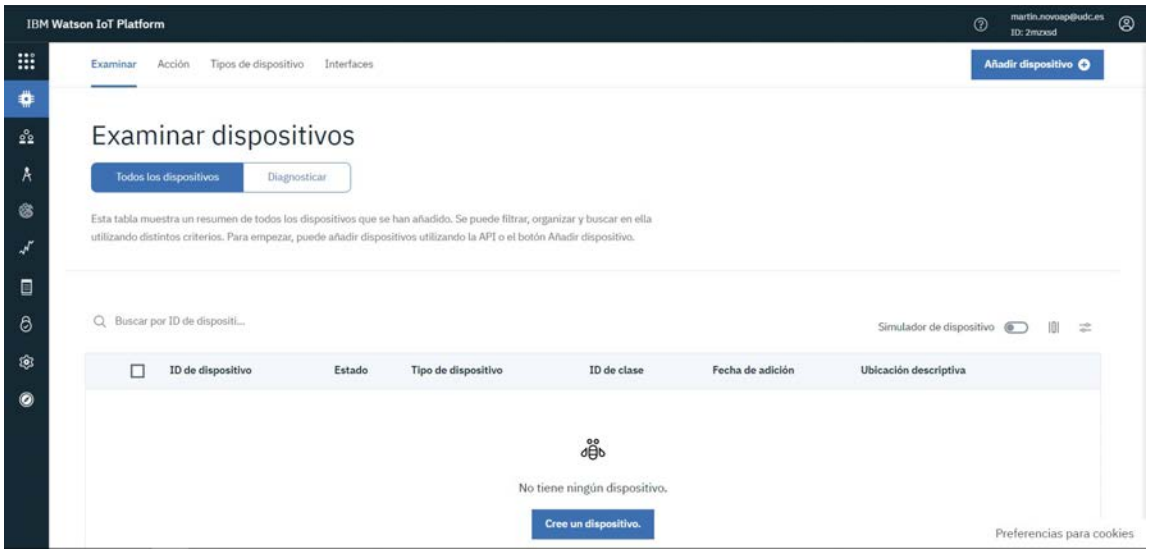


Figura 13.107 – IBM IoT Platform

Ahora, se le da nombre al tipo de dispositivo y un ID al dispositivo y se le da todo a siguiente.

Figura 13.108 – Configuración del dispositivo

Una vez finalizado aparece la siguiente información que se puede ver en la Figura 13.109. Es necesario copiar la señal de autenticación y el ID de Organización, ya que habrá que introducirla en el código del Arduino.

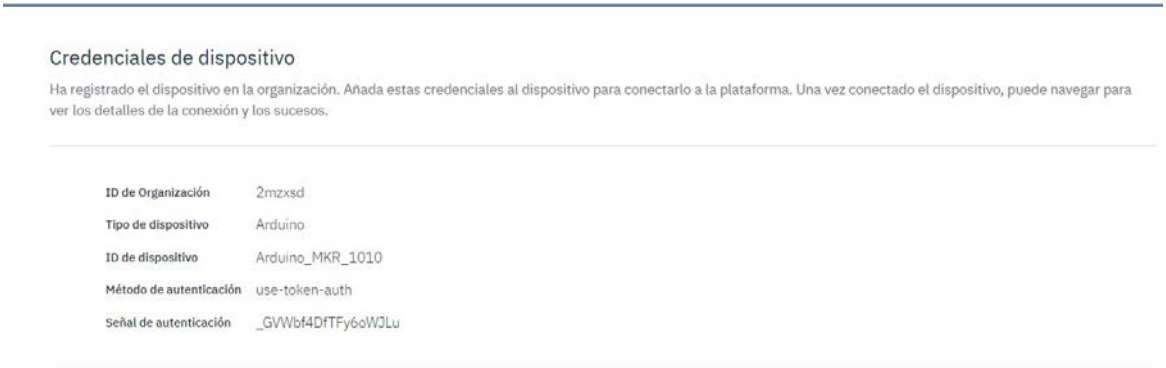


Figura 13.109 – Información del dispositivo

Ahora en el menú lateral se abre $Seguridad \rightarrow Seguridad\ \ de\ \ conexion$ para cambiar el nivel de seguridad a “TLS opcional”.

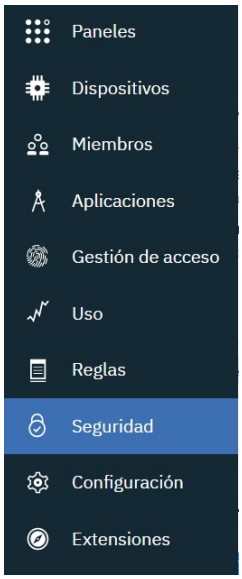


Figura 13.110 – Seguridad

Figura 13.111 – Configuración de la seguridad

Ahora, es necesario programar el Arduino con el código de Arduino del Apartado 12.1.9. Cuando el Arduino ejecuta el programa, la plataforma debe detectarlo y marcarlo como conectado. Como se puede ver en la Figura 13.112 aparece así definido.

Figura 13.112 – Estado del dispositivo

Ahora, se configurará la interfaz gráfica. Para ello se abre el apartado “Paneles” del menú lateral.

Figura 13.113 – Paneles

Se crea un nuevo panel y se le da un nombre.

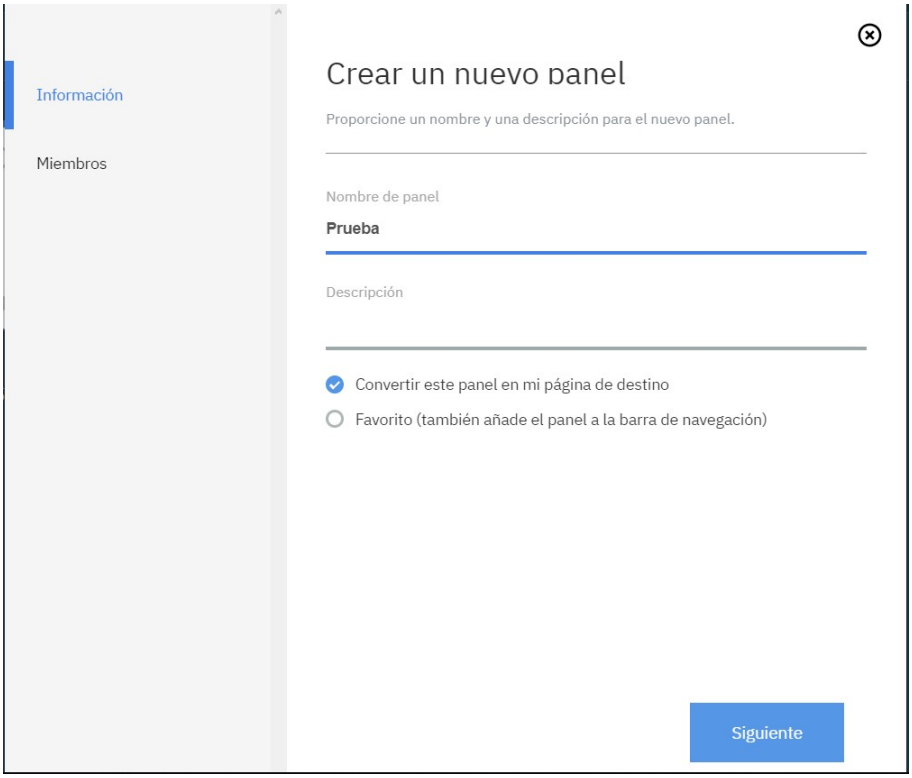


Figura 13.114 – Crear un nuevo panel

Para crear las gráficas se clica en “Añadir nueva tarjeta”.

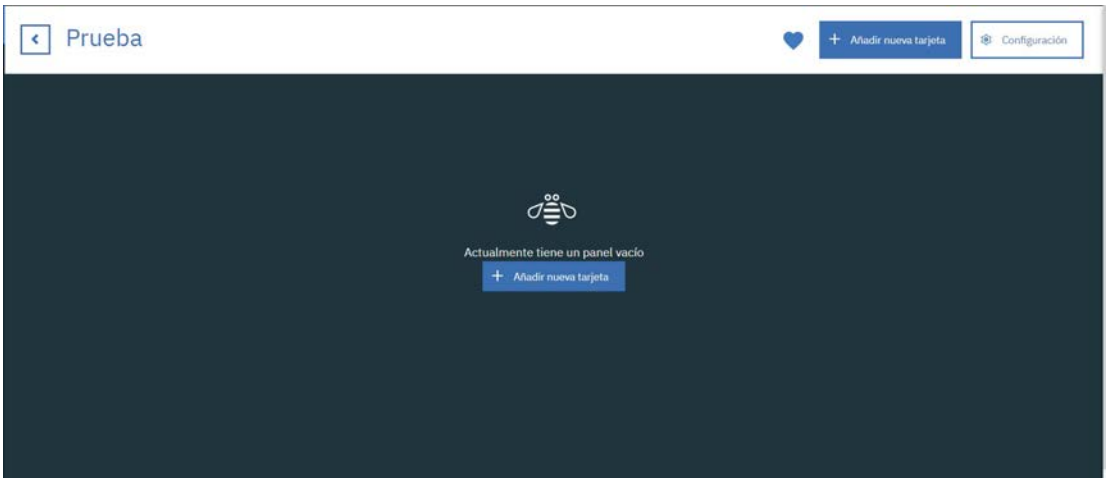


Figura 13.115 – Panel vacio

En este caso, se configura un gráfico de lineas.

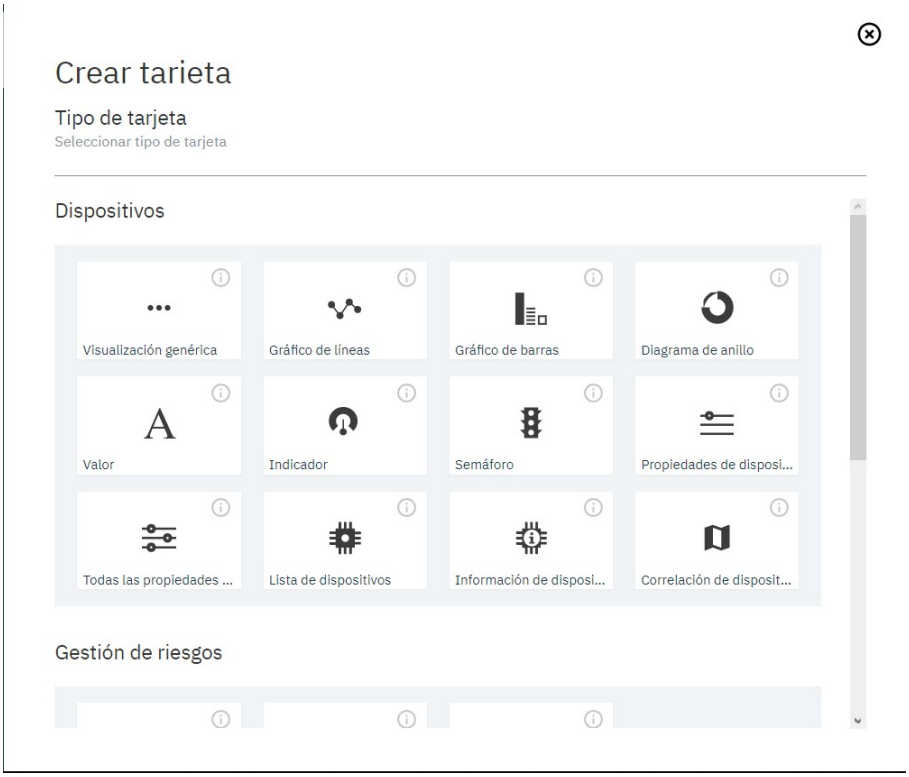


Figura 13.116 – Crear tarjeta

Ahora, se selecciona el Arduino como el origen del conjunto de datos.

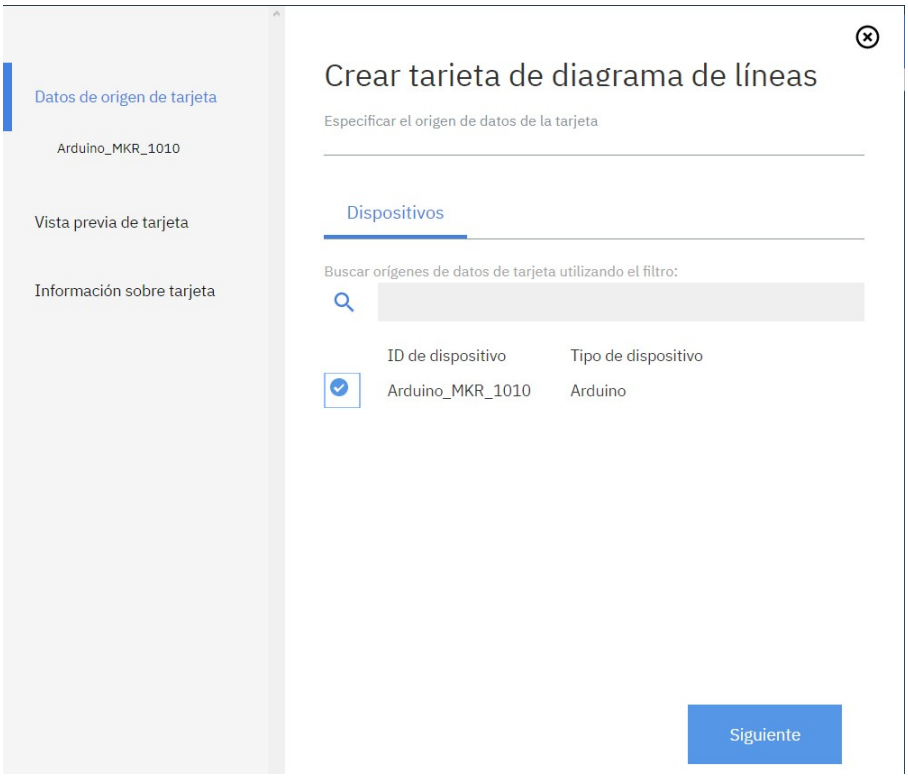


Figura 13.117 – Crear gráfico de lineas I

Ahora, en Suceso se selecciona status y en propiedad la variable que se desea representar. Por ultimo, se seleccionan el tipo de valor (numero, texto, etc) y la unidad. IMPORTANTE: para poder seleccionar las variables el Arduino las tuvo que haber mandado previamente.

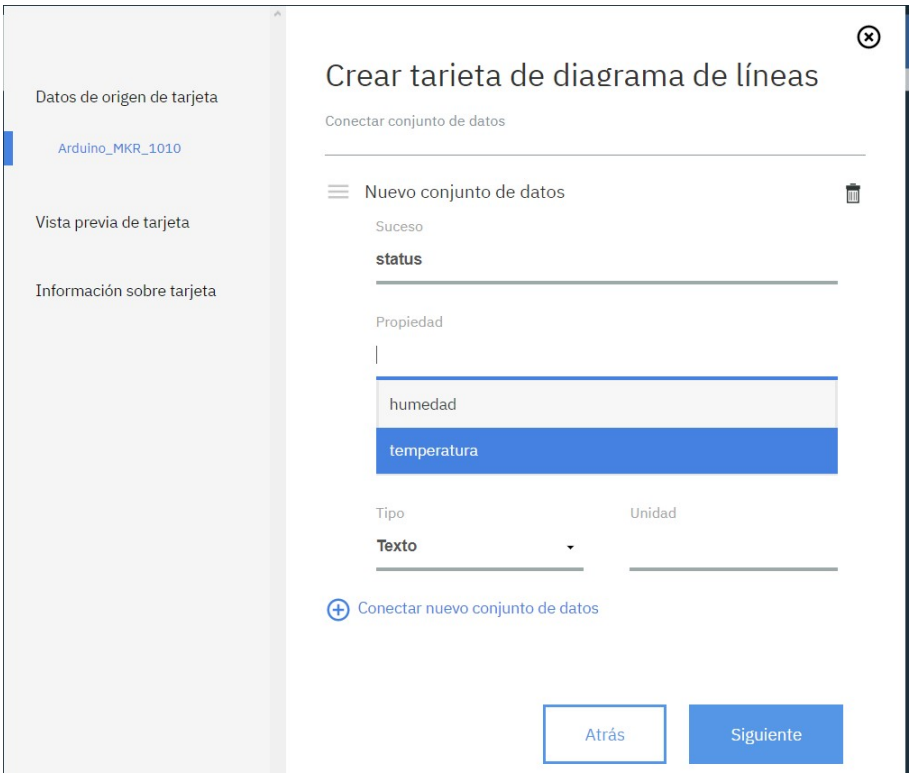


Figura 13.118 – Crear gráfico de lineas II

Como se puede ver en la Figura 13.119, se crean las gráficas y automáticamente se empiezan a representar los datos. Sin embargo, estos valores no se están almacenando en ningún lado. Para ello hay que crear una base de datos.

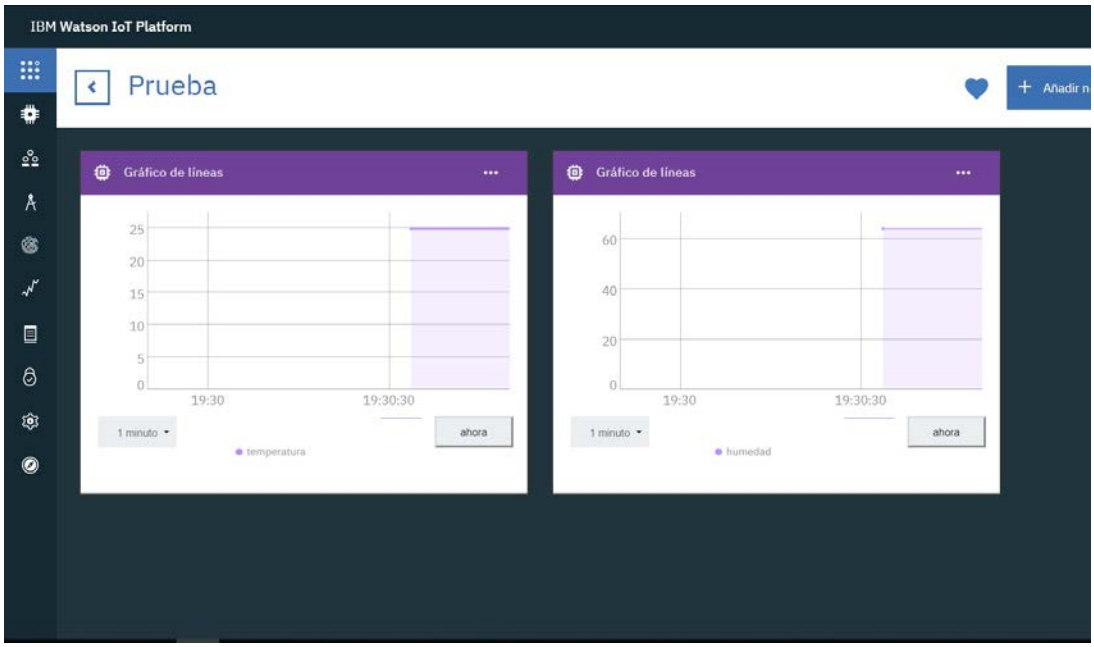


Figura 13.119 – Representación de las variables

13.7.0.2. Cloudant NoSQL

Para la base de datos se utilizará Cloudant, una base de datos NoSQL. Para crearla, es necesario ir a $Panel\ de\ control \rightarrow Crear recurso \rightarrow Bases\ de\ datos \rightarrow Cloudant$.

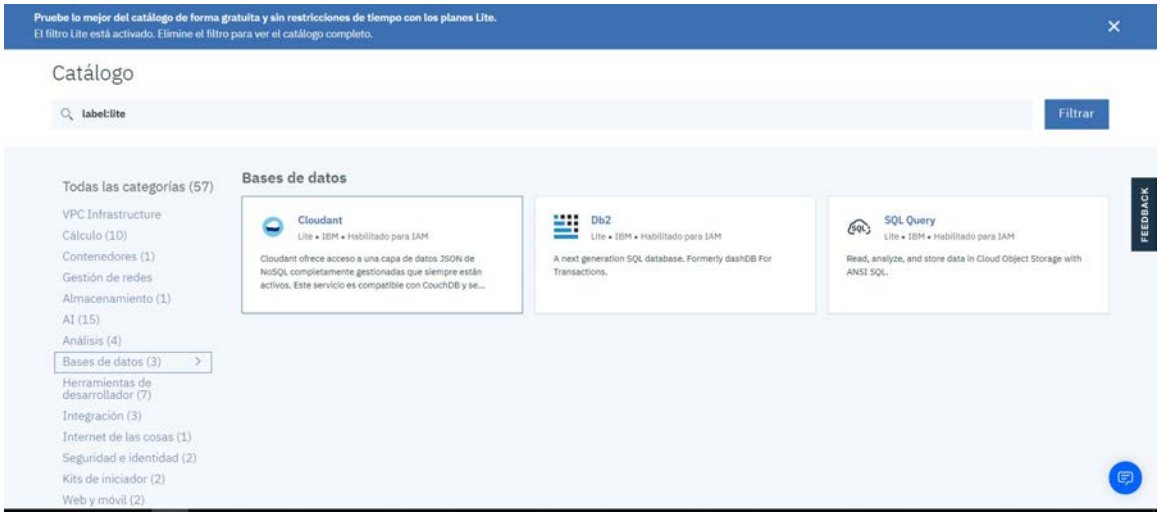


Figura 13.120 – Selección del recurso “Cloudant”

Ahora, se le da un nombre al recurso y se establece Londres como ubicación. En los modos de autentificación ponemos “Use both legacy credentials and IAM”.

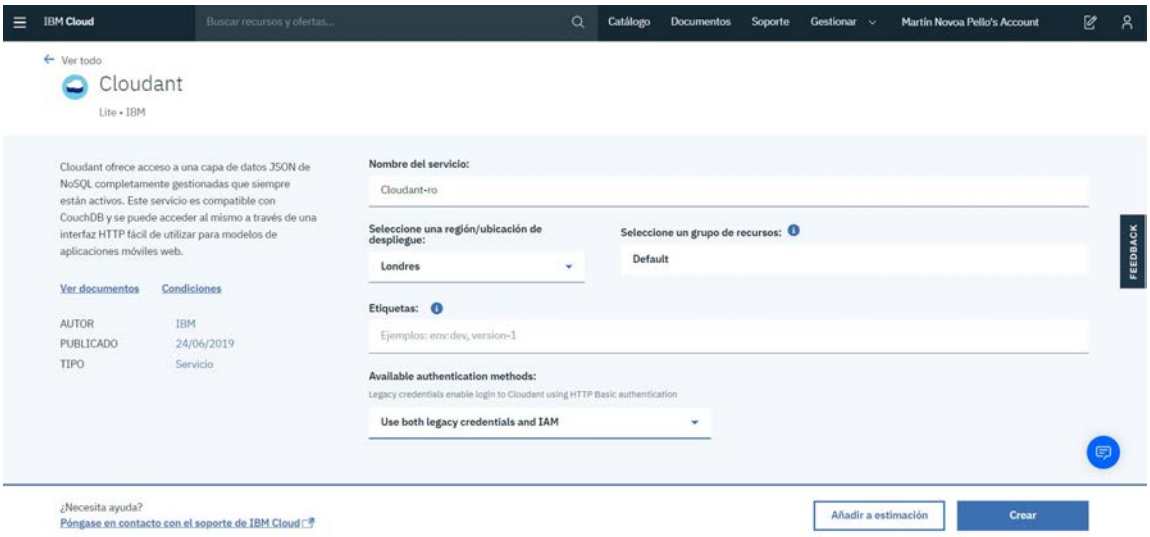


Figura 13.121 – Configuración del recurso “Cloudant

Una vez creado, se abre la configuración del recurso. En el menú lateral, se abre “Conexiones” y se clica en “Crear conexión”.

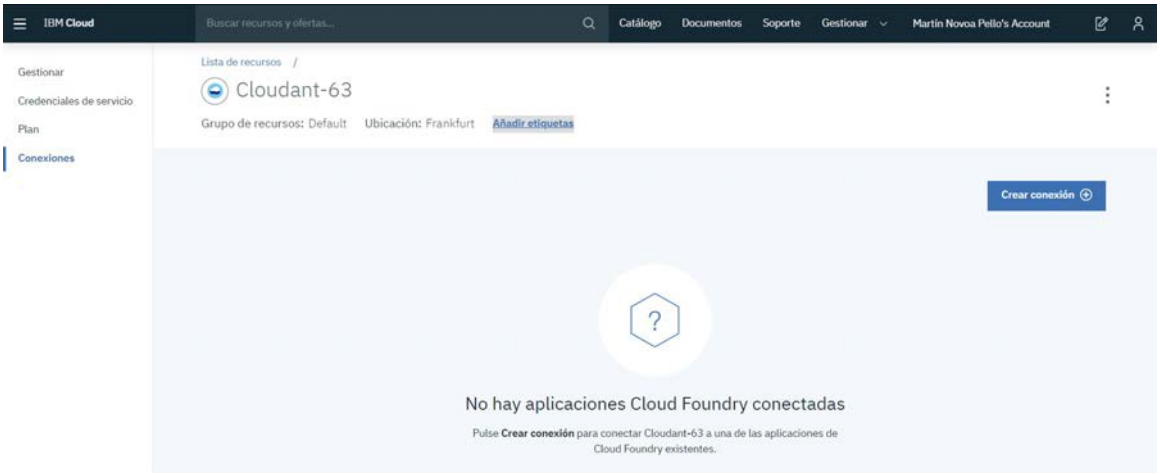


Figura 13.122 – Conexiones de Cloudant

Se selecciona la región de Londres y se clica en “Conectar”.

Figura 13.123 – Conectar a app existente

Ahora, se crea la aplicación de Bluemix que permitirá mandar los datos de Watson IoT a

Cloudant. Para ello, es necesario abrir el siguiente enlace, ya que es una API que no aparece en el menú de los recursos: http://ibm.biz/BdimYj

Se introduce el nombre y la ubicación. En el apartado “Select a source provider”, se deja el que está por defecto.

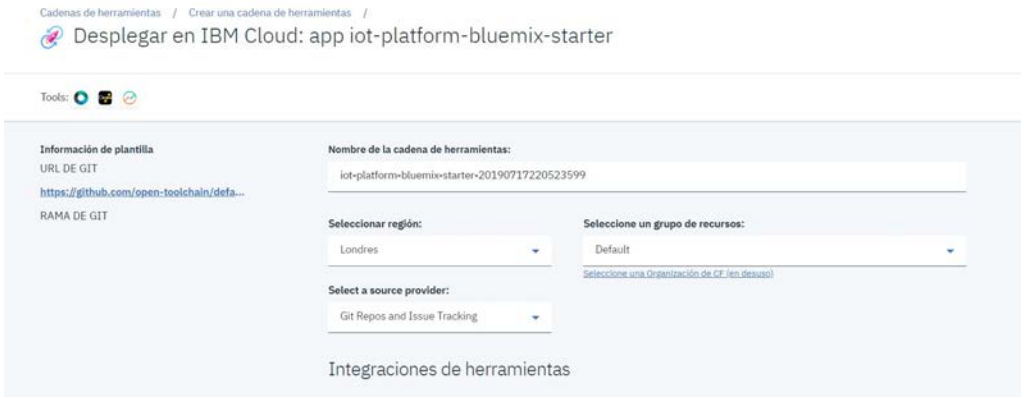


Figura 13.124 – Configuración del recurso “Desplegar en IBM Cloud”. I

Bajando la página, aparece la clave de API. Como no se dispone de ninguna, se clica en “Crear”

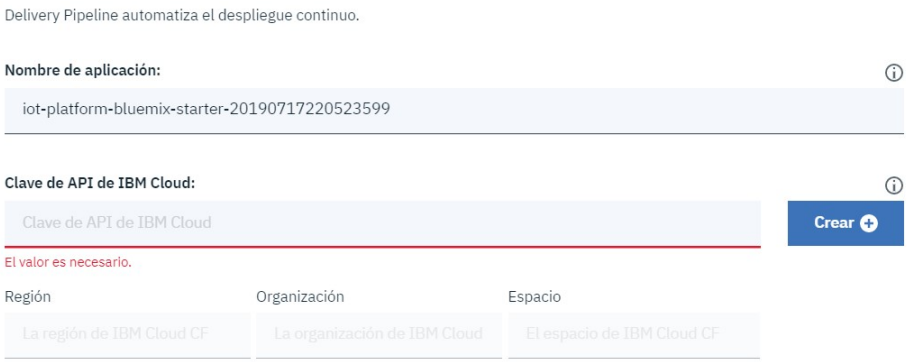


Figura 13.125 – Configuración del recurso “Desplegar en IBM Cloud”. II

Como se puede ver en la Figura 13.126, automáticamente se rellenan todos los apartados. Si no es así, se establece Londres como región.

Figura 13.126 – Configuración del recurso “Desplegar en IBM Cloud. III

Ahora, es necesario configurar la conexión en Watson IoT Platfonm. Para ello, se abre el recurso y en el panel de lateral se abre la sección de “Extensiones”.

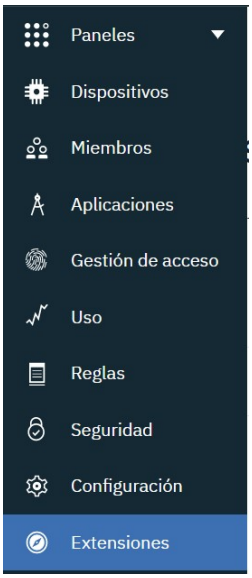


Figura 13.127 – Extensiones

Ahora, se clica en configurar el almacenamiento de datos históricos y luego a seleccionar en cloudant.

Figura 13.128 – Configuración del almacenamiento de datos históricos I

Se introduce la configuración pertinente y se clica en “Listo”.

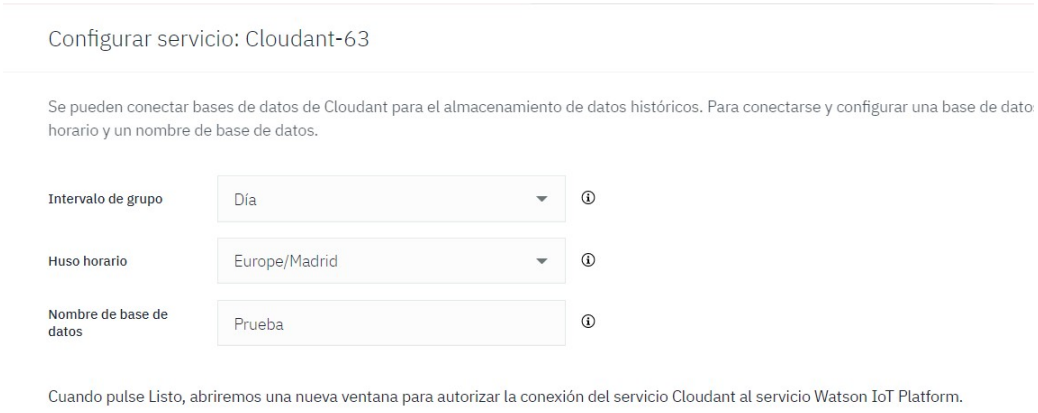


Figura 13.129 – Configuración del almacenamiento de datos históricos II

El intervalo del grupo se puede modificar posteriormente en este mismo apartado. Las posibilidades son: día, semana y mes.

Ahora, abriendo Cloudant se puede ver como se crearon las bases de datos, en las que se van a ir introduciendo los datos subidos a la nube.

Figura 13.130 – Bases de datos creadas en Cloudant

Al abrir la base de datos se puede ver que por defecto se representa la metadata.

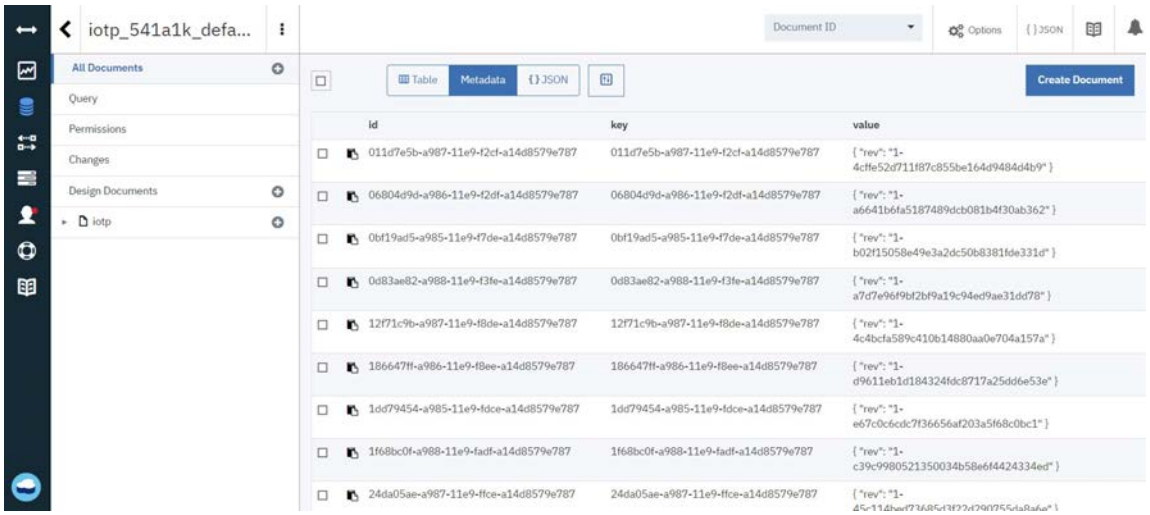


Figura 13.131 – Metadata de la base de datos

Para ver los datos correctamente se clica en “Table”.

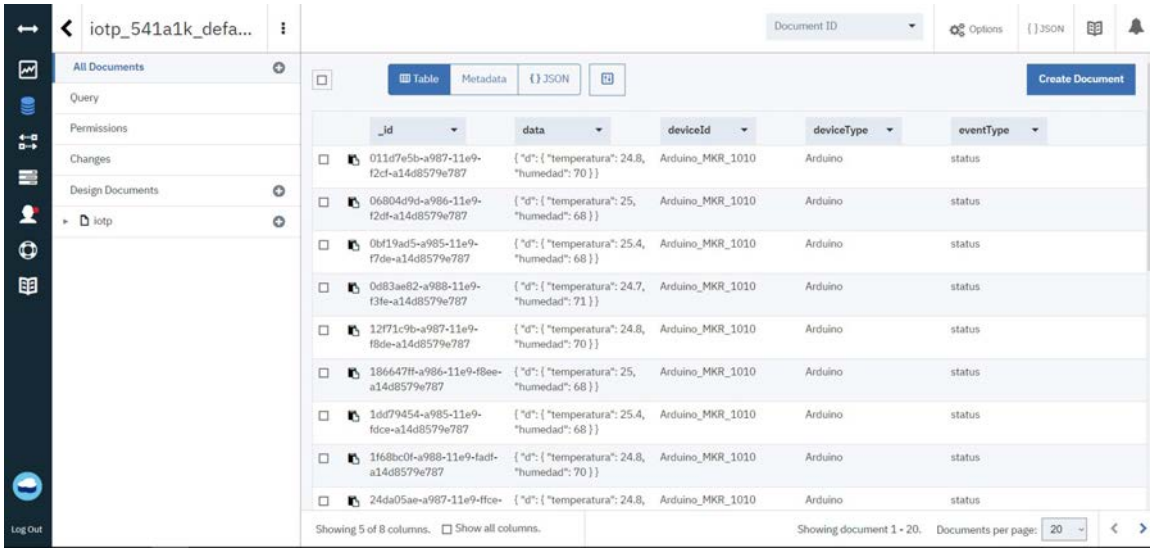


Figura 13.132 – Datos de la base de datos I

Ahora, se seleccionan los datos que se quieran visualizar.

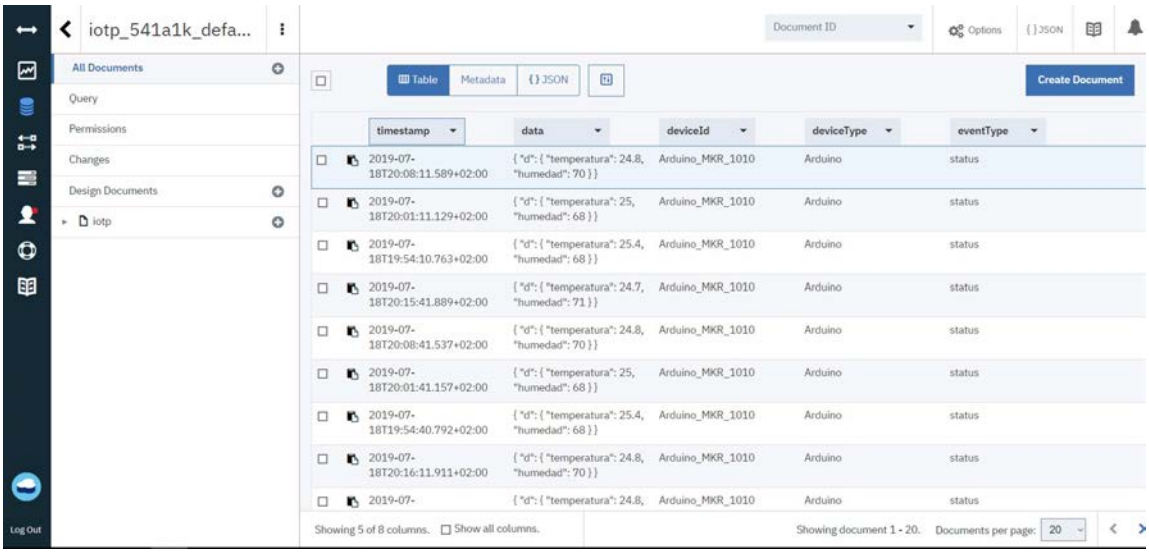


Figura 13.133 – Datos de la base de datos II

Como, se puede ver en la Figura 13.133 los datos están completamente desordenados. Para ordenarlos vamos a $iotp \rightarrow views \rightarrow by-date$. Esto hay que realizarlo para cada una de las bases de datos.

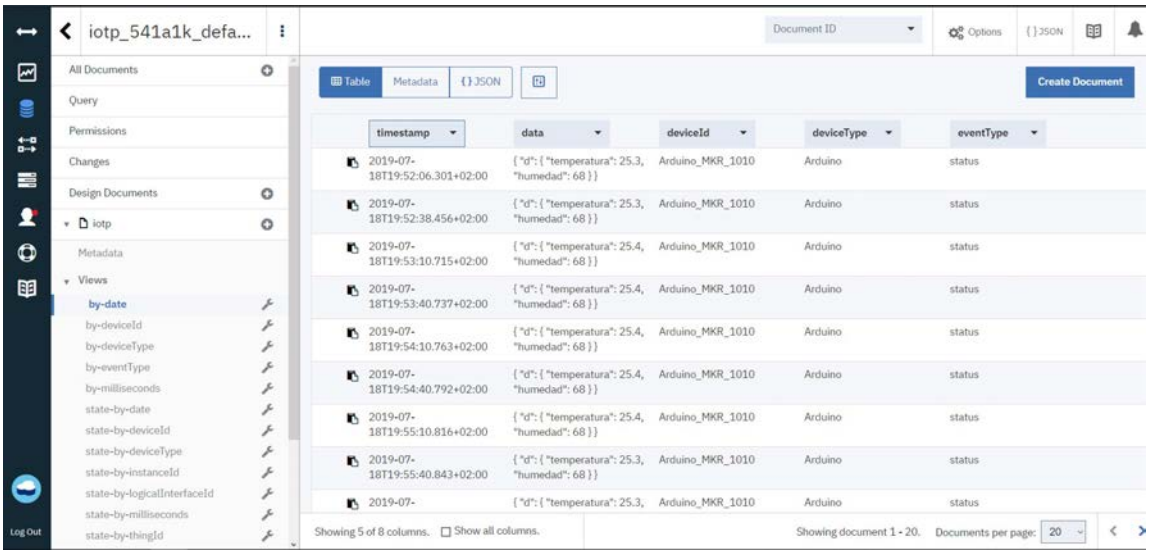


Figura 13.134 – Datos de la base de datos ordenados

13.7.1. Exportado de los datos

Los datos se descargan con la misma organización que le hayamos dado a la base de datos en el apartado anterior. Para descargarlos hay que clicar en “()JSON” y en la pestaña que se abre hacer clic derecho y guardar como.

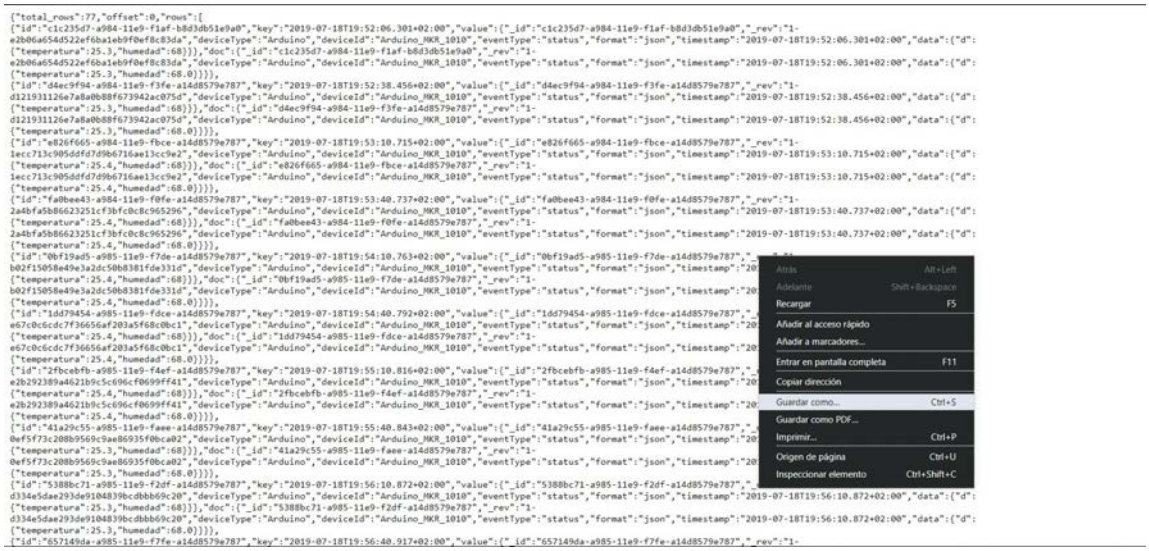


Figura 13.135 – JSON de la bse de datos

Ahora el archivo que hemos descargado hay que convertirlo a un formato que pueda ser leido por programas como Excel o MATLAB. Para transformarlo se sube la web https://json-csv.com transfórmalo a un formato .csv.

13.8. Cayenne

En primer lugar, es necesario registrarse en la web https://developers.mydevices.com/cayenne/features/ Una vez hecho esto, se abre la siguiente página web que se recoge en la Figura 13.136, en la que se pide que se seleccione el dispositivo que va a ser conectado al servicio.

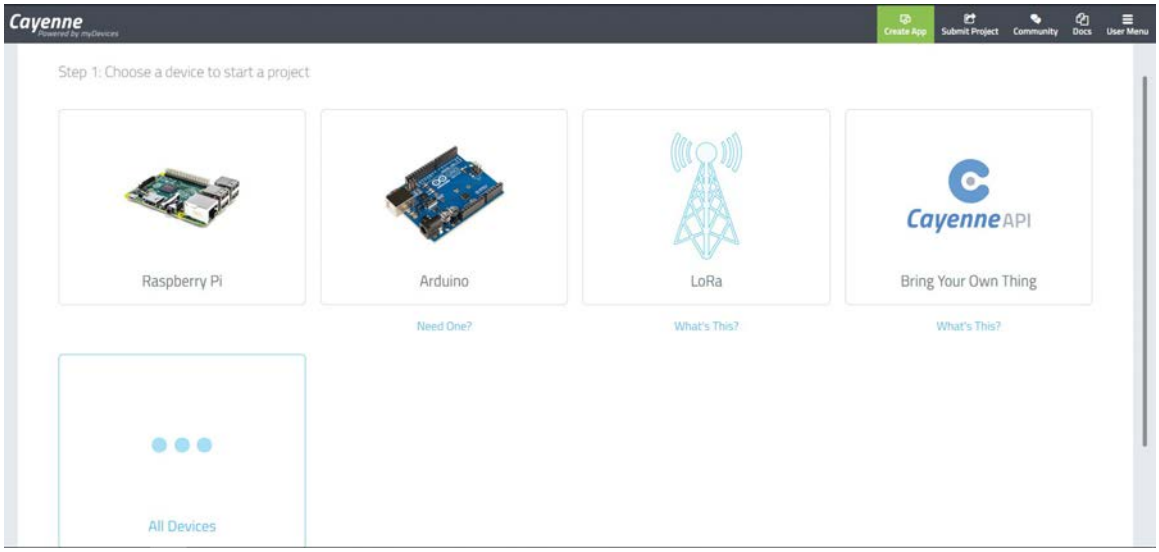


Figura 13.136 – Selección del dispositivo

Al seleccionar Arduino dice que se debe abrir el IDE de Arduino y configurar la placa.

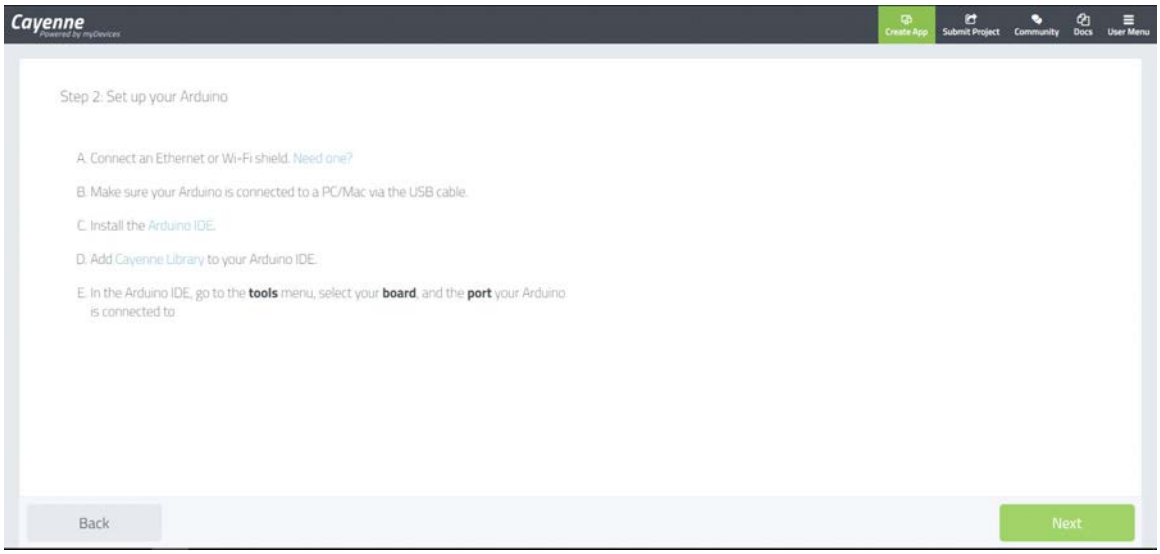


Figura 13.137 – Configurar el IDE de Arduino y la placa

Sin embargo, como se puede ver en la Figurar 13.138, no aparece el MKR 1010. Esto se debe a que, en este momento no existe la librería que permita utilizar con el MKR 1010

Figura 13.138 – Dispositivos Arduino disponibles

Por lo tanto, se puede seleccionar el Arduino MKR1000 o seleccionar “Bring your own thing” en la primera pantalla, como se hará en este caso. Como Cayenne no tiene soporte para el Arduino MKR 1010, es necesario crear las librerias necesarias. Esta librería se puede descargar de GitHub: https://github.com/martinnp12/TFG/blob/master/CayenneMQTT.zip.

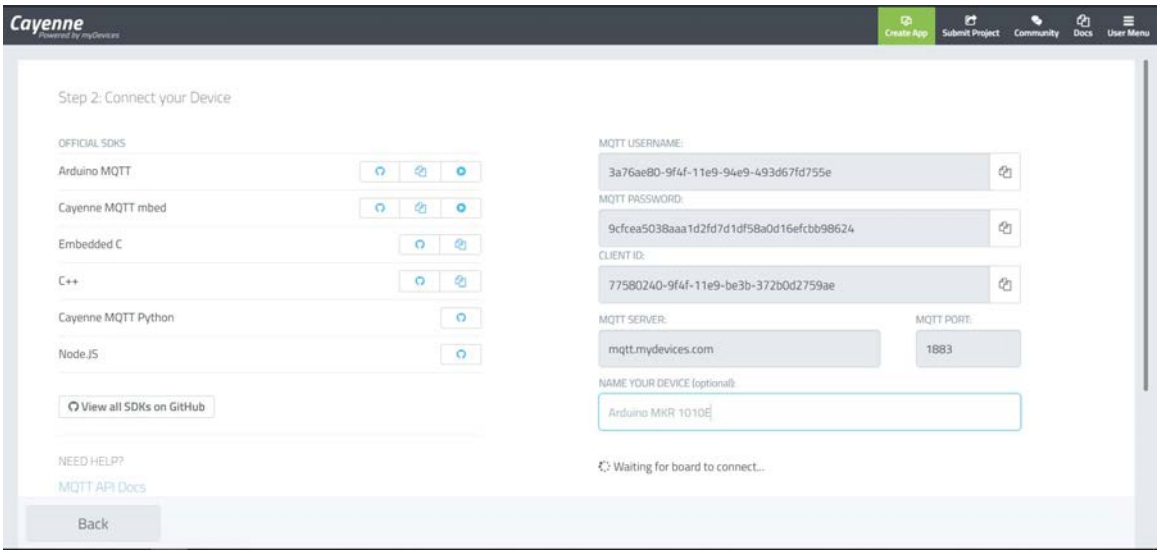


Figura 13.139 – Conectar el dispositivo

Ahora, se introduce el nombre del dispositivo, en este caso “Arduino MKR 1010” y se copian las claves “MQTT USERNAME”, “MQTT PASWORD”, “CLIENT ID” para pegarlas en las variables correspondientes del script de ejemplo “MKR1010”, que se encuentra en la librería de GitHub y lo ejecutamos.

En cuanto el Arduino se conecta el servidor se abre la siguiente página:

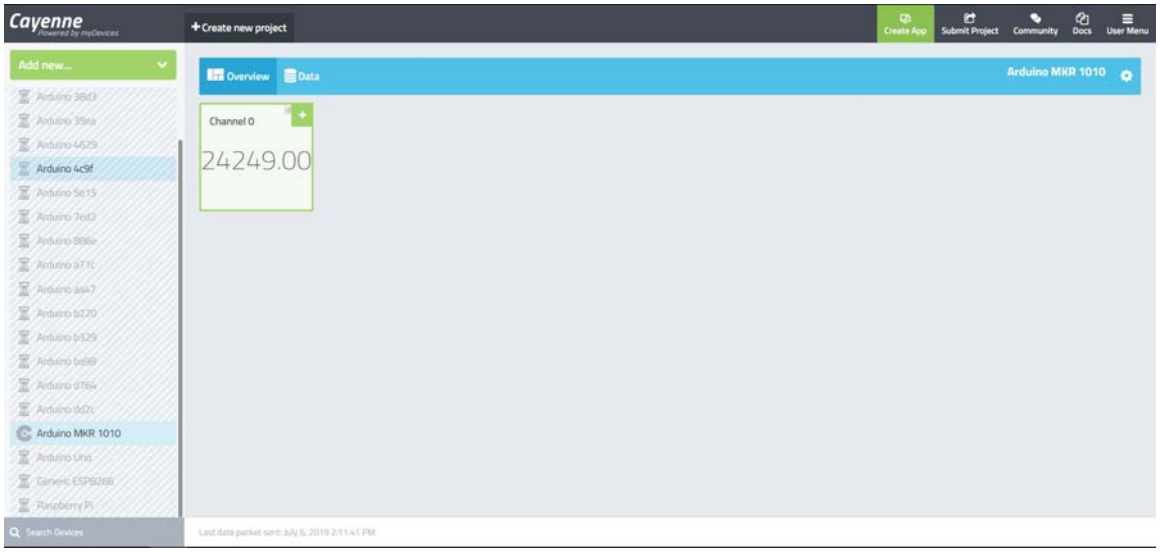


Figura 13.140 – Interfaz de la plataforma I

Como se puede ver en la Figura 13.140, ya aparece el dispositivo. Enviando diferentes parámetros, la plataforma los detecta automáticamente.

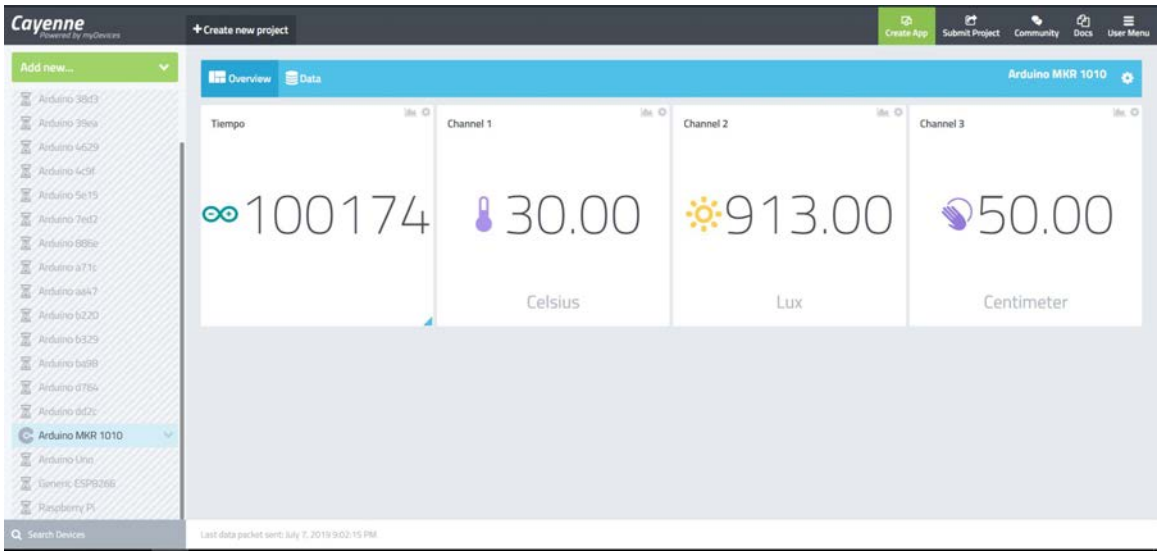


Figura 13.141 – Interfaz de la plataforma II

Clicando en el símbolo de la gráfica de cada uno de los widgets se pueden representar los datos.

Figura 13.142 – Representación de los datos

Como se puede ver en la Figura 13.142 se puede seleccionar el periodo de tiempo que se desee representar. Yendo al apartado “Data” se pueden ver los datos tabulados e incluso descargarlos en formato .csv

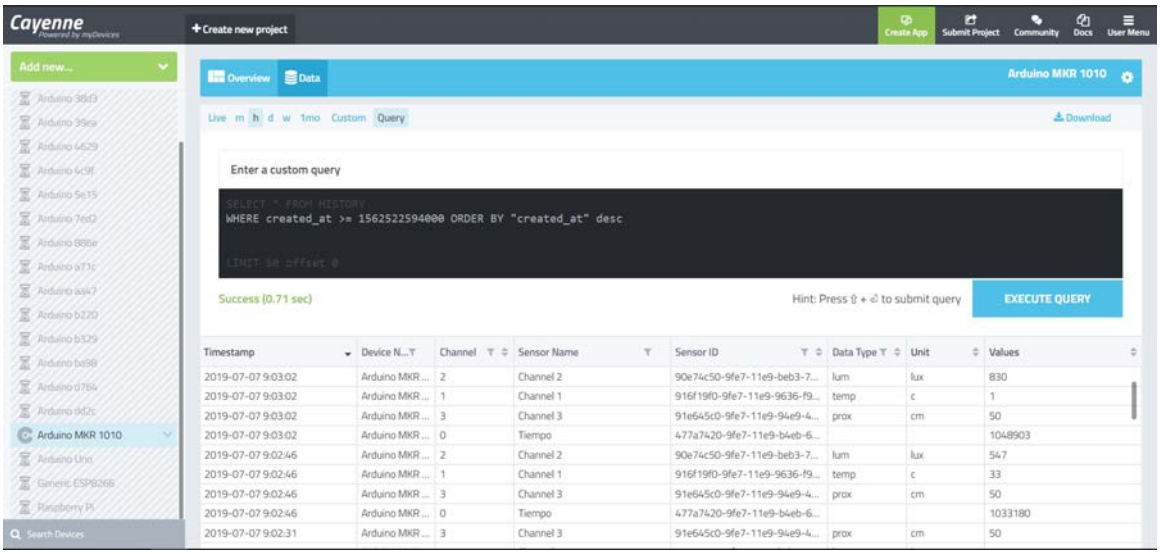


Figura 13.143 – Datos almacenados en la plataforma

TÍTULO: **DESARROLLO DE UN SISTEMA DE ADQUISICIÓN DE DATOS PARA PLANTA DE BOMBEO FOTOVOLTAICO.**

PLANOS

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

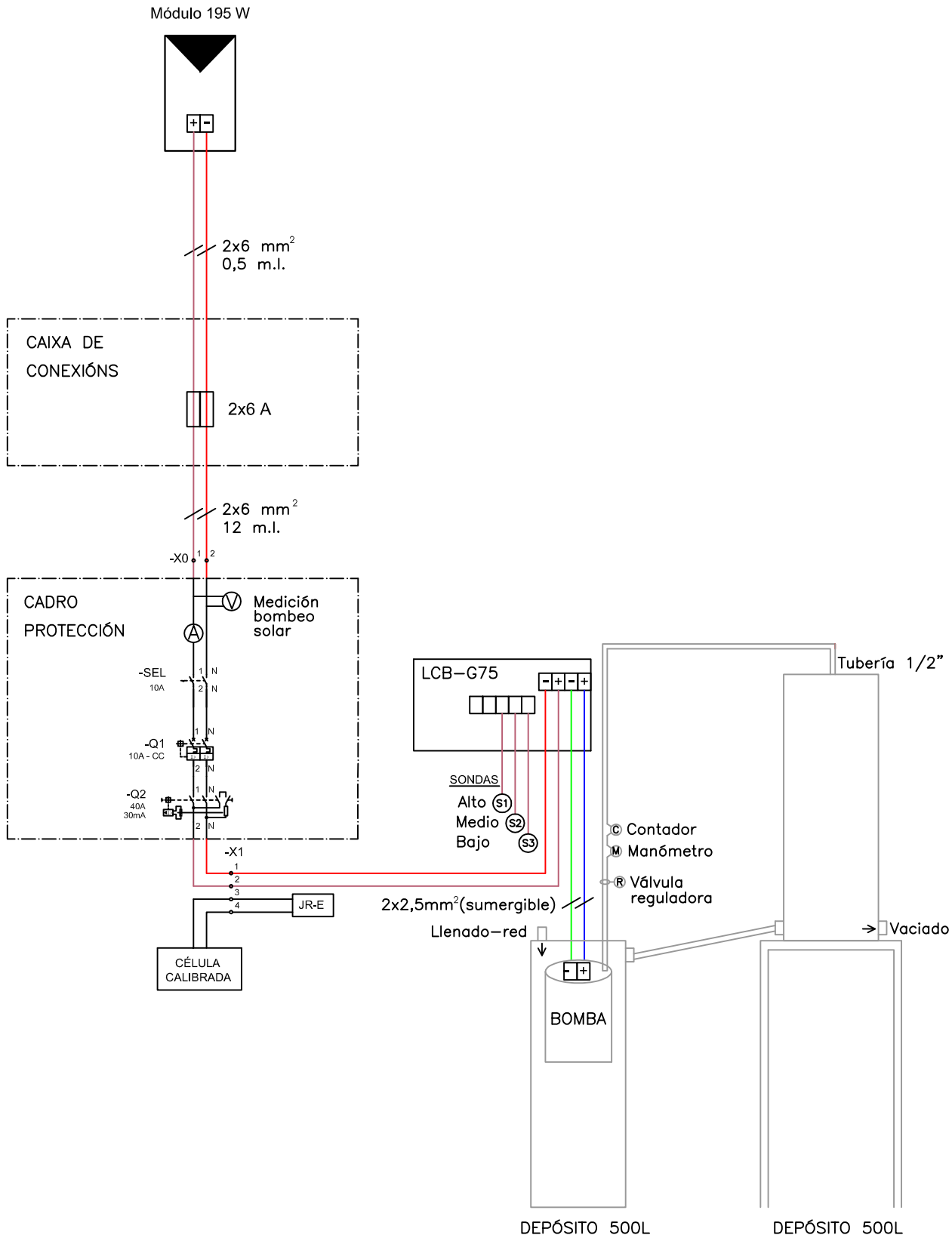
15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

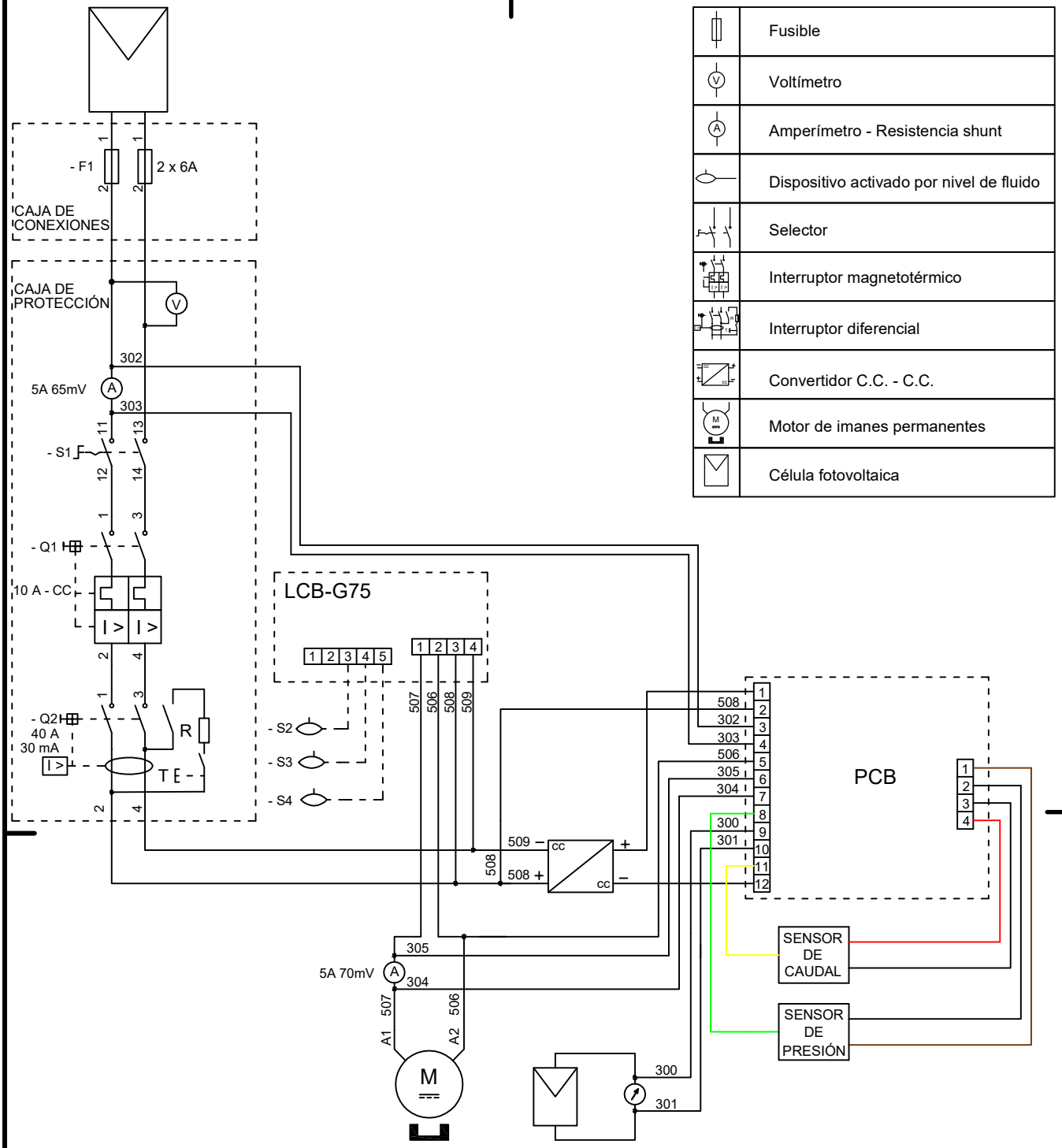
Fdo.: MARTÍN NOVOA PELLO

Índice de planos



ULTIMA REVISION		TITULO PLANO	KIT BOMBEO SOLAR FOTOVOLTAICO
DIBUJADO: VANESA VARELA PAIS	FECHA: 04 / 12 / 17	PLANO Nº: 1	ESCALA: S/E
VERIFICADO: ROBERTO TUIMIL PARAPAR	FECHA: 22 / 12 / 17	FORMATO: A4	HOJA 1 DE 1

NARONTEC GESTIÓN TECNOLÓGICA



UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECT. IND. Y AUTOMÁTICA

TFG Nº: 770G01A165

TÍTULO DEL TFG:

SISTEMA DE ADQUISICIÓN DE DATOS PARA PLANTA DE BOMBEO FOTOVOLTAICO

TÍTULO DEL PLANO:

ESQUEMA ELÉCTRICO DE LA INSTALACIÓN

FECHA: 26/08/2019

ESCALA: S/E

AUTOR:

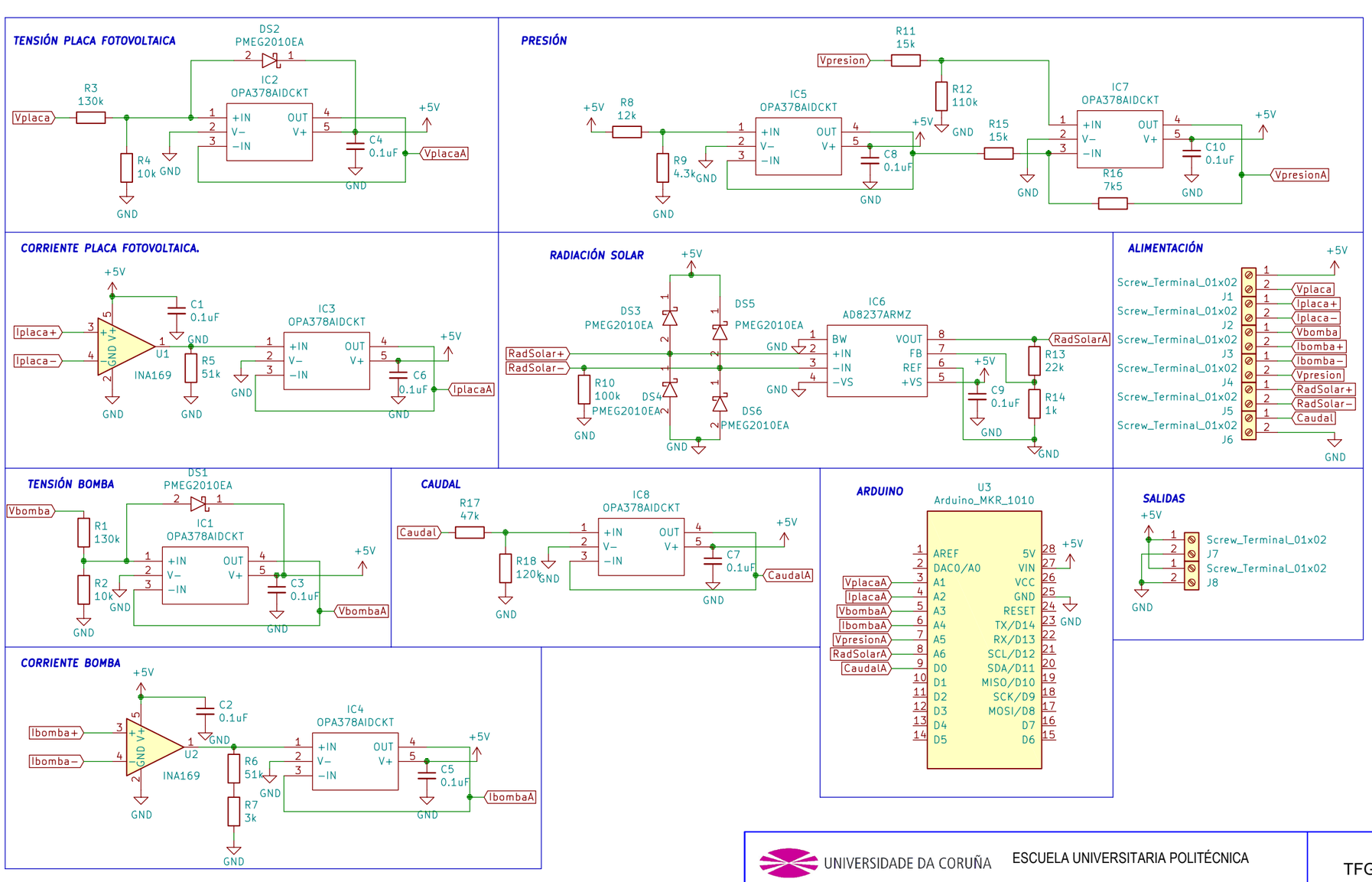
MARTÍN NOVOA PELLO

FIRMA:

PLANO Nº: 01

CREADO CON UNA VERSION PARA ESTUDIANTES DE AUTODESK

CREADO CON UNA VERSION PARA ESTUDIANTES DE AUTODESK



UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECT. IND. Y AUTOMÁTICA

TFG Nº: 770G01A165

TÍTULO DEL TFG:

SISTEMA DE ADQUISICIÓN DE DATOS PARA PLANTA DE BOMBEO FOTOVOLTAICO

TÍTULO DEL PLANO:

ESQUEMÁTICO PCB

FECHA: 26/08/2019

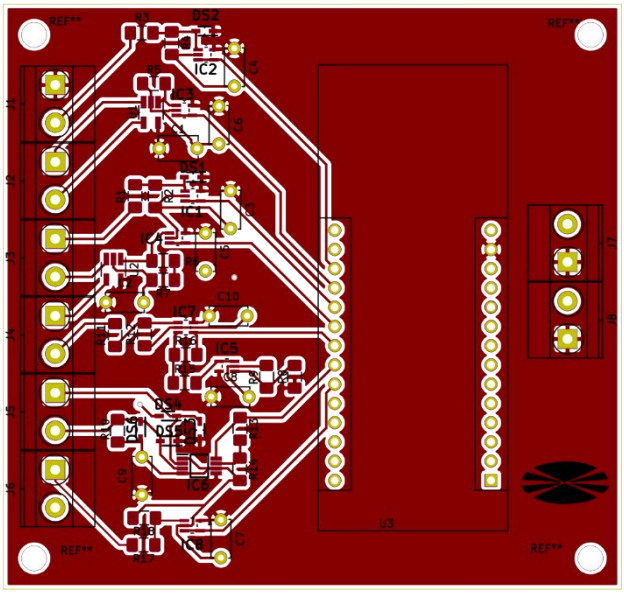
ESCALA: SE

AUTOR: MARTÍN NOVOA PELLO

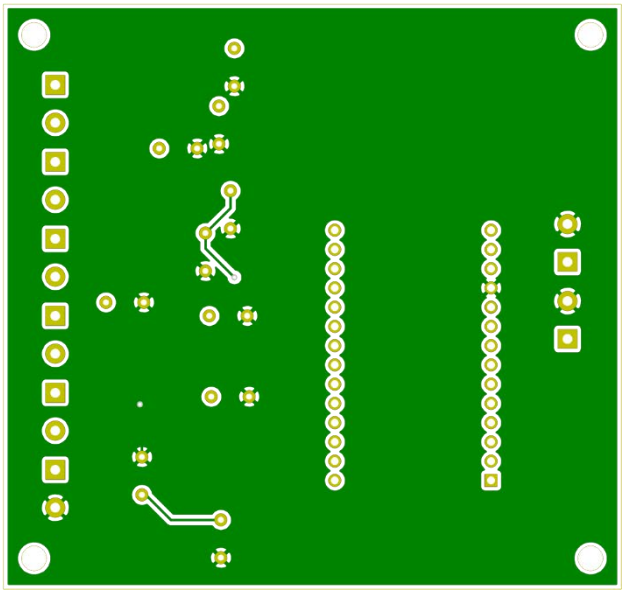
FIRMA:

PLANO Nº: 02

A3



CAPA SUPERIÓR



CAPA INFERIOR

UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECT. IND. Y AUTOMÁTICA

TFG Nº: 770G01A165

TÍTULO DEL TFG:

SISTEMA DE ADQUISICIÓN DE DATOS PARA PLANTA DE BOMBEO FOTOVOLTAICO

TÍTULO DEL PLANO:

PCB

FECHA: 26/08/2019

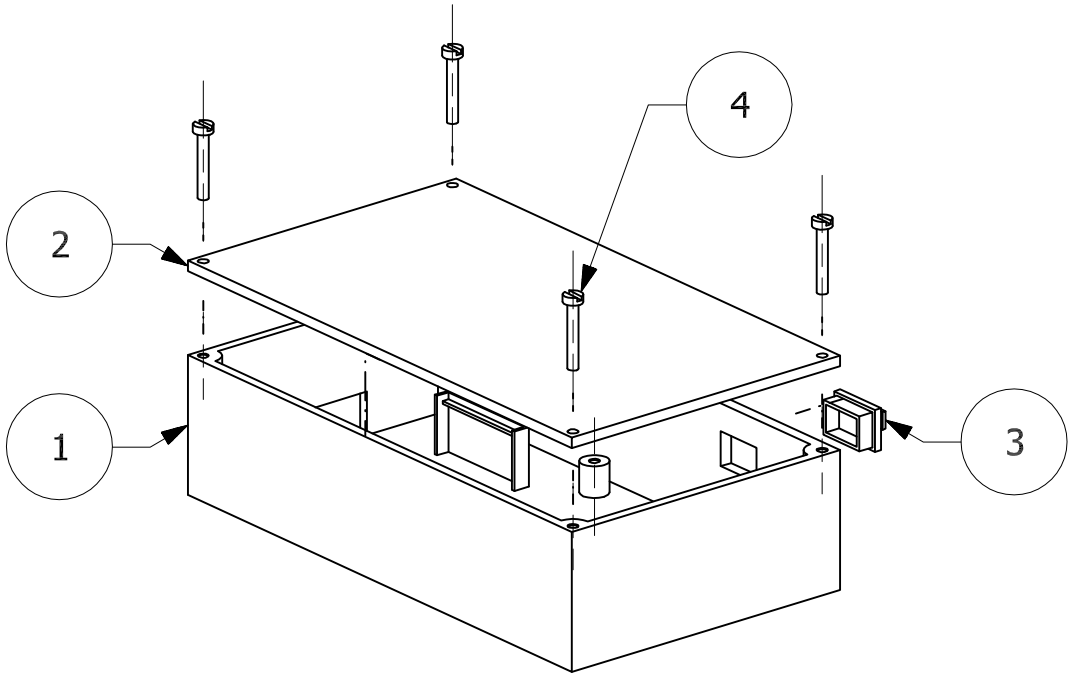
ESCALA: 1:1

AUTOR:

MARTÍN NOVOA PELLO

FIRMA:

PLANO Nº: 03



Nº PIEZA	NOMBRE DE LA PIEZA	CANTIDAD
1	CAJA	1
2	TAPA	1
3	TAPÓN	1
4	TORNILLO	4

UNIVERSIDADE DA CORUÑA ESCUELA UNIVERSITARIA POLITÉCNICA

GRADO EN INGENIERÍA ELECT. IND. Y AUTOMÁTICA

TFG Nº: 770G01A165

TÍTULO DEL TFG:

SISTEMA DE ADQUISICIÓN DE DATOS PARA PLANTA DE BOMBEO FOTOVOLTAICO

TÍTULO DEL PLANO:

EXPLOSIONADO CAJA

FECHA: 26/08/2019

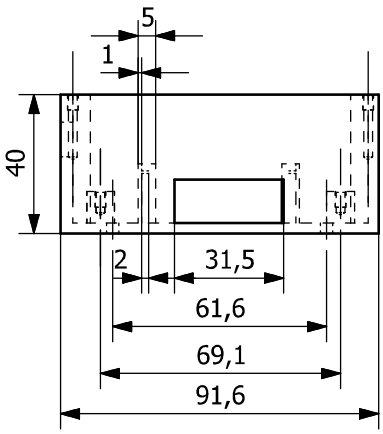
ESCALA: S/E

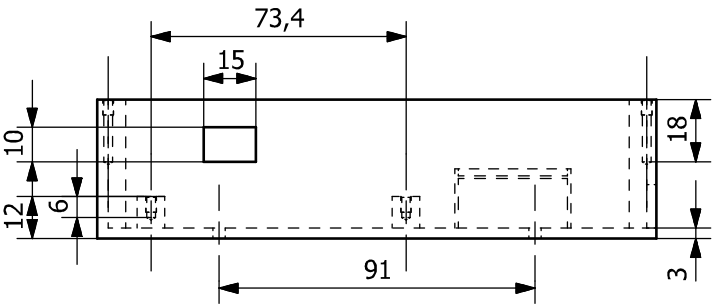
AUTOR:

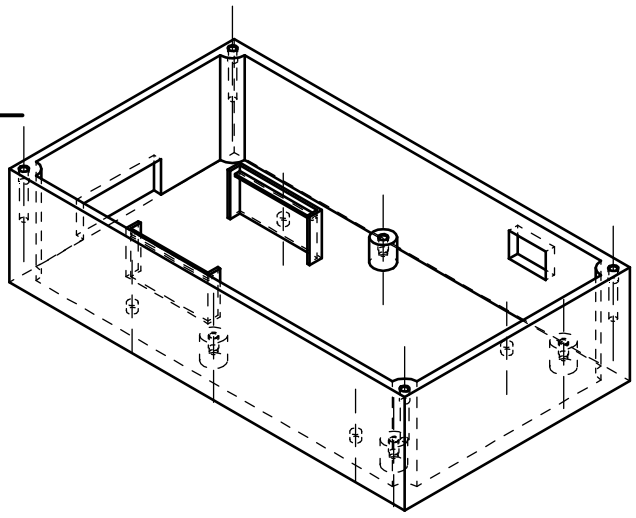
MARTÍN NOVOA PELLO

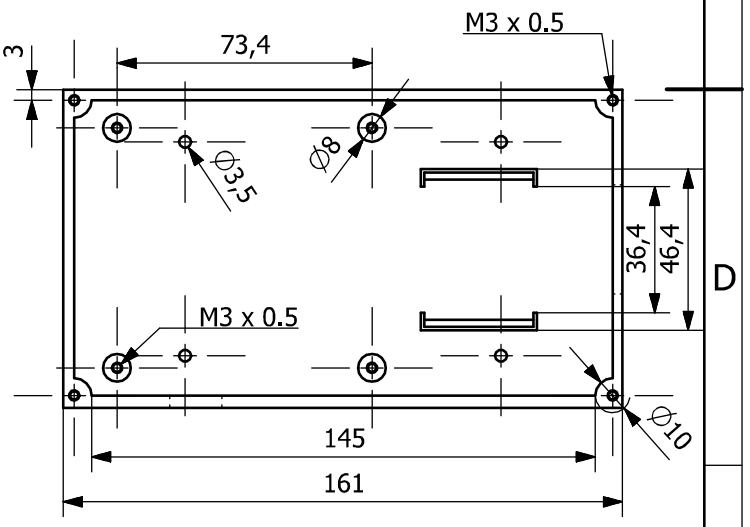
FIRMA:

PLANO Nº: 04









UNIVERSIDADE DA CORUÑA **ESCOLA UNIVERSITARIA POLITÉCNICA**

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG. Nº: 770G01A165

TÍTULO DEL PROYECTO:

SISTEMA DE ADQUISICIÓN DE DATOS PARA PLANTA DE BOMBEO FOTOVOLTAICO

TÍTULO DEL PLANO:

PARTE INFERIOR CAJA

FECHA: 26/08/2019

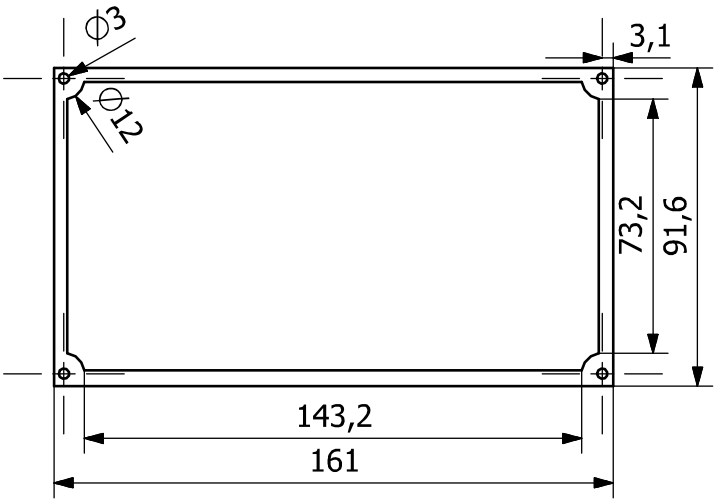
ESCALA: 1:2

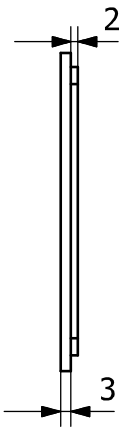
AUTOR:

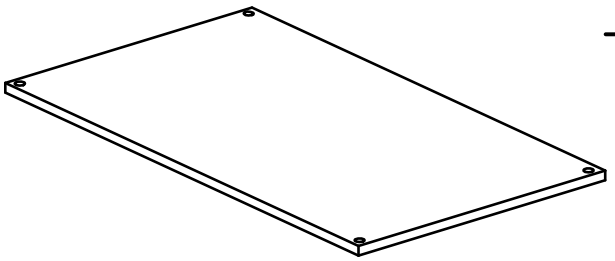
MARTÍN NOVOA PELLO

FIRMA:

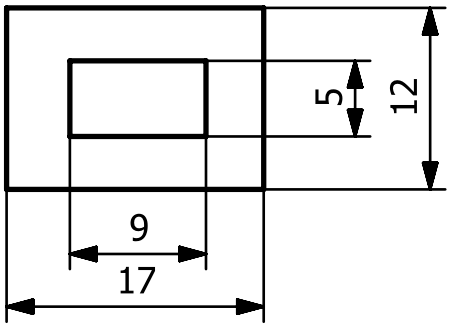
PLANO Nº: 05

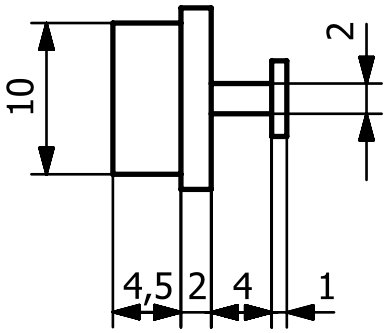


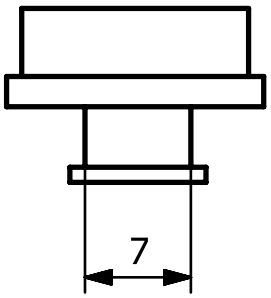


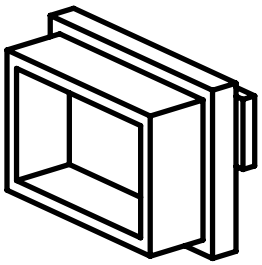


UNIVERSIDADE DA CORUÑA **ESCOLA UNIVERSITARIA POLITÉCNICA** **GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA**	**TFG Nº: 770G01A165**
TÍTULO DEL PROYECTO: **SISTEMA DE ADQUISICIÓN DE DATOS PARA PLANTA DE BOMBEO FOTOVOLTAICO**	
TÍTULO DEL PLANO: **TAPA CAJA**	FECHA: 26/08/2019
	ESCALA: 1:2
AUTOR: **MARTIN NOVOA PELLO** — FIRMA:	PLANO Nº: 06









UNIVERSIDADE DA CORUÑA **ESCOLA UNIVERSITARIA POLITÉCNICA**

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TFG Nº: 770G01A165

TÍTULO DEL PROYECTO:

SISTEMA DE ADQUISICIÓN DE DATOS PARA PLANTA DE BOMBEO FOTOVOLTAICO

TÍTULO DEL PLANO:

TAPÓN CAJA

FECHA: 26/08/2019

ESCALA: 2:1

AUTOR:

MARTÍN NOVOA PELLO

FIRMA:

PLANO Nº: 07

TÍTULO: **DESARROLLO DE UN SISTEMA DE ADQUISICIÓN DE DATOS PARA PLANTA DE BOMBEO FOTOVOLTAICO.**

PLIEGO DE CONDICIONES

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

Fdo.: MARTÍN NOVOA PELLO

Índice del documento PLIEGO DE CONDICIONES

14 ESPECIFICACIONES DE LOS MATERIALES

Los componentes empleados para la realización de este proyecto se han seleccionado de manera que no se superen, en ningún momento, los valores nominares de los componentes con el fin de prolongar su vida útil lo máximo posible.

14.1. Requisitos de los componentes

Los materiales y componentes seleccionados para la implementación del proyecto deben cumplir con los requisitos mínimos para el correcto funcionamiento del sistema, asegurandose de que ninguna magnitud pueda superar los valores nominales de los componentes. Para cada uno de los componentes seleccionados, el fabricante debe garantizar el cumplimiento de las especificaciones recogidas en las hojas de características.

La máquina en la que se instale el servidor y la base de datos debe tener, como mínimo, 2 GB de memoria RAM y contar con sistema operativo Windows 7/8/8.1/10. En caso de emplear una Raspberry Pi3 u ordenadores con otros sistemas operativos sera necesario descargar la versión de ThingsBoard y pgAdmin4 correspondiente.

14.2. Sustitución de componentes

En caso de que sea necesario la sustitución de alguno de los componentes sera necesario que tengan, como mínimo, los mismos rangos de funcionamiento que los componentes instalados previamente, asegurandose de que cuenten con las mismas dimensiones que los componentes previamente instalados, con el fin de poder emplear el mismo PCB. En caso contrario, será necesario diseñar y fabricar uno nuevo que permita la inclusión de estos.

En caso de que el Arduino MKR 1010 se descatalogue, cabe la posibilidad de que los códigos de programación no funcionen para el nuevo Arduino, ya que están diseñados específicamente para el SAMD21 y para el ESP32 de U-BLOX. Por lo tanto, será necesario modificar el programa para adaptarlo a la nueva placa.

15 SOLUCIÓN DE PROBLEMAS

15.1. Thingsboard y base de datos

Para el funcionamiento del almacenamiento de los datos es necesario que la aplicación de ThingsBoard esté encendida. Una vez iniciada permanecerá encendida, incluso después de

reiniciar el ordenador. Para encenderlo se ejecutará el programa “Simbolo del sistema” como administrador y entonces, se ejecutara la siguiente linea:

net star thingsboard

Si, por el contrario, se quiere apagar el servidor se debe ejecutar la siguiente linea:

net stop thingsboard

15.2. Programación del Arduino

En caso de que sea necesario reprogramar el Arduino, se debe cargar el programa en los primeros 10 segundos desde la conexión de la placa. Una vez pasado este tiempo el Arduino entrará en bajo consumo, estado en el que no es posible la comunicación con el ordenador. IMPORTANTE: La programación se debe realizar siempre con la planta desconectada.

TÍTULO: **DESARROLLO DE UN SISTEMA DE ADQUISICIÓN DE DATOS PARA PLANTA DE BOMBEO FOTOVOLTAICO.**

MEDICIONES

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

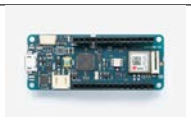
FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

Fdo.: MARTÍN NOVOA PELLO

Índice del documento MEDICIONES

16 MATERIALES

Componentes activos				
ID	**Imagen**	**Descripción**	**Referencia**	**Uds**
U1,U2		INA169 Amplificador para medición de corriente.	Farnell: 1234749	2
U3		Arduino con conexión WiFi.	Farnell: 2917569	1
IC1, IC2, IC3, IC4, IC5, IC7, IC8		OPA378 Amplificador operacional rail to rail.	Farnell: 2496308	7
IC6		AD8237 Amplificador de instrumentación.	Farnell: 2213571	1
DCDC		Convertidor DC-DC.	Sin ref.	1
SP1		Sensor de presión.	Sin ref.	1
SC1		Sensor de caudal.	Sin ref.	1

Conductores y elementos de conexión				
ID	**Imagen**	**Descripción**	**Referencia**	**Uds**
CMF1		Cable multifilar 1.5 mm	Farnell: 2528084	7
CMF2		Cable multifilar 2.5 mm	Farnell: 2528085	2
TPH		Terminal punta hueca	Farnell: 2218459	15
TA		Terminal de anillo	Farnell: 2295567	4
J1, J2, J3, J4, J5, J6 J7, J8		Borna circuito impreso	Cetronic. 999019660	8
CMH		Conector macho-hembra	Cetronic: 999334088	2
PCB		Placa de circuito impreso	Sin ref.	1

Otros materiales				
ID	**Imagen**	**Descripción**	**Referencia**	**Uds**
T1		Tornillo M3, 20 mm Cabeza alomada y ranura plana	Farnell: 1419295	4
T2		Tornillo M3, 6 mm Cabeza alomada y ranura tipo Pozidriv	Farnell: 1419986	4
PLA		Filamento PLA 1,75 mm, 1kg	BQ-PLA BLK	1

Componentes pasivos I				
ID	**Imagen**	**Descripción**	**Referencia**	**Uds**
DS1, DS2, DS3, DS4, DS5, DS6		NSR0530HT1G Diodo Schottky.	Farnell: 2508350	6
C1, C2, C3, C4, C5, C6, C7, C8, C9, C10		MCDTR10M35-1-RH Condensador de tantalio.	Farnell: 1186143	10
R1, R3		Resistencia SMD 150 k\Omega 250 mW	Farnell: 1534577	2
R2, R4, R8		Resistencia SMD 12 k\Omega 250 mW	Farnell: 1571705	3
R5, R6		Resistencia SMD 51 k\Omega 250 mW	Farnell: 1540039	2
R7		Resistencia SMD 3 k\Omega 250 mW	Farnell: 1570826	1
R9		Resistencai SMD 4.3 k\Omega 125 mW	Farnell: 2351720	1
R10		Resistencia SMD 100 k\Omega 250 mW	Farnell:1534576	1

Componentes pasivos II				
ID	**Imagen**	**Descripción**	**Referencia**	**Uds**
R11, R15		Resistencia SMD 15 k\Omega 250 mW	Farnell: 2413382	2
R12		Resistencia SMD 110 k\Omega 125 mW	Farnell: 2413382	1
R13		Resistencia SMD 22k\Omega 250 mW	Farnell: 1534581	1
R14		Resistencia SMD 1 k\Omega 250mW	Farnell: 1862221	1
R16		Resistencia SMD 7.5 k\Omega 250 mW	Farnell: 1631439	1
R17		Resistencia SMD 47 k\Omega 250 mW	Farnell: 1540033	1
R18		Resistencia SMD 120 k\Omega 250 mW	Farnell:2838420	1
SHUNT		Resistencia shunt Caida de tensión de 75 mV para 5 A.	Sin ref.	1

17 Mano de obra

Tarea	Personal	Horas
Documentación		
Análisis de los diferentes sistemas Arduino	Graduado en Ingeniería Electrónica Industrial y Automática	15
Analisis de los diferentes sensores	Graduado en Ingeniería Electrónica Industrial y Automática	5
Estudio de los diferentes métodos de alimentación	Graduado en Ingeniería Electrónica Industrial y Automática	15
Estudio de las plataformas IoT	Graduado en Ingeniería Electrónica Industrial y Automática	80
Ejecución		
Calibrado de la célula fotovoltaica	Graduado en Ingeniería Electrónica Industrial y Automática	10
Diseño y simulación de los circuitos de acondicionamiento	Graduado en Ingeniería Electrónica Industrial y Automática	60
Diseño del PCB	Graduado en Ingeniería Electrónica Industrial y Automática	20
Impresión 3D de la caja	Graduado en Ingeniería Electrónica Industrial y Automática	1
Modelado 3D de la caja	Graduado en Ingeniería Electrónica Industrial y Automática	10
Prueba de los sensores	Graduado en Ingeniería Electrónica Industrial y Automática	5
Programación del Arduino	Graduado en Ingeniería Electrónica Industrial y Automática	60
Soldadura del PCB y pruebas de su funcionamiento	Graduado en Ingeniería Electrónica Industrial y Automática	10
Redacción		
Redacción de la memoria	Graduado en Ingeniería Electrónica Industrial y Automática	60
	TOTAL:	341

TÍTULO: **DESARROLLO DE UN SISTEMA DE ADQUISICIÓN DE DATOS PARA PLANTA DE BOMBEO FOTOVOLTAICO.**

PRESUPUESTO

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: SEPTIEMBRE DE 2019

AUTOR: EL ALUMNO

Fdo.: MARTÍN NOVOA PELLO

Índice del documento PRESUPUESTO

18 MATERIALES

Componentes activos			
ID	**Descripción**	**Coste unitario (€/ud)**	**Coste total**
U1, U2	INA169 Amplificador para medición de corriente.	2.20	4.40
U3	Arduino con conexión WiFi.	28.07	28.07
IC1, IC2, IC3, IC4, IC5, IC7, IC8	OPA378 Amplificador operacional rail to rail.	1.81	12.67
IC6	AD8237 Amplificador de instrumentación.	2.65	2.65
DCDC	Convertidor DC-DC.	3.78	3.78
SP1	Sensor de presión.	7.81	7.81
SC1	Sensor de caudal.	5.57	5.57
		TOTAL	64.59

Conductores y elementos de conexión			
ID	**Descripción**	**Coste unitario (€/ud)**	**Coste total**
CMF1	Cable multifilar 1.5 mm	1	7
CMF2	Cable multifilar 2.5 mm	1.55	3.1
TPH	Terminal punta hueca	0.521	7.815
TA	Terminal de anillo	0.378	1.512
J1, J2, J3, J4, J5, J6 J7, J8,	Borna circuito impreso	0.23	0.23
CMH	Conector macho-hembra	1.74	3.48
PCB	Placa de circuito impreso	2.42	2.42
		TOTAL	25.327

Otros materiales			
ID	**Descripción**	**Coste unitario (€/ud)**	**Coste total**
T1	Tornillo M3, 20 mm Cabeza alomada y ranura plana	3.33	13.32
T2	Tornillo M3, 6 mm Cabeza alomada y ranura tipo Pozidriv	1.84	7.36
PLA	Filamento PLA 1,75 mm, 1kg	20.15	20.15
		TOTAL	40.83

Componentes pasivos I			
ID	**Descripción**	**Coste unitario (€/ud)**	**Coste total**
DS1, DS2, DS3, DS4, DS5, DS6	NSR0530HT1G Diodo Schottky.	0.139	0.834
C1, C2, C3, C4, C5, C6, C7, C8, C9, C10	MCDTR10M35-1-RH Condensador de tantalio.	0.9	9
R1, R3	Resistencia SMD $150k\Omega$ 250 mW	0.0261	0.0522
R2, R4, R8	Resistencia SMD $12k\Omega$ 250 mW	0.0261	0.0522
R5, R6	Resistencia SMD $51k\Omega$ 250 mW	0.0261	0.0522
R7	Resistencia SMD $3k\Omega$ 250 mW	0.0423	0.0423
R9	Resistencai SMD $4{,}3k\Omega$ 125 mW	0.0423	0.0423
R10	Resistencia SMD $100k\Omega$ 250 mW	0.0423	0.0423
		TOTAL	10.062

Componentes pasivos II			
ID	**Descripción**	**Coste unitario (€/ud)**	**Coste total**
R11, R15	Resistencia SMD $15k\Omega$ 250 mW	0.0261	0.0522
R12	Resistencia SMD $110k\Omega$ 250 mW	0.0261	0.0261
R13	Resistencia SMD $22k\Omega$ 250 mW	0.0261	0.0261
R14	Resistencia SMD $1k\Omega$ 250 mW	0.0261	0.0261
R16	Resistencai SMD $7{,}5k\Omega$ 125 mW	0.0261	0.0261
R17	Resistencia SMD $47k\Omega$ 250 mW	0.0261	0.0261
R18	Resistencia SMD $120k\Omega$ 250 mW	0.0261	0.0261
SHUNT	Resistencia shunt Caida de tensión de 75 mV para 5 A.	1.66	1.66
		TOTAL	1.843

19 MANO DE OBRA

Tipo de mano de obra	Unidades (h)	Honorarios (€/h)	Total (€)
Graduado en Ingeniería Electrónica Industrial y Automática	341	40	13640

20 PRESUPUESTO

Concepto	Coste (€)
Componentes activos	64.59
Componentes pasivos	11.905
Conductores y elementos de conexión	25.327
Otros materiales	40.83
Mano de obra	13640
SUBTOTAL	13780.232
IVA (21 %)	2893.85
TOTAL	**16676.5**