Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

NODE-RED COMO HERRAMIENTA VISUAL DE DISPOSITIVOS IoT

Alumno: **Miguel Ángel Moral Pérez**

Tutor 1: Prof. D. Ildefonso Ruano Ruano
Depto.: Ingeniería de Telecomunicación

Tutor 2: Prof. D. Elísabet Estévez Estévez
Depto.: Ingeniería Electrónica y Automática

Noviembre, 2021

Índice general

Índice de ilustraciones

Índice de tablas

1. Resumen

1.1 Resumen

En este Trabajo de Fin de Grado (TFG), se estudiará la herramienta de programación visual Node-RED. Su funcionamiento interno y la relación que tiene con dispositivos IoT. Además del estudio de esta herramienta y la creación de una guía de uso, se implementará un caso práctico para aplicar todos los conocimientos adquiridos. Además, se establecerá una comparativa con herramientas similares.

Se detallará la forma de captación de los datos que nos proporcionen los sensores, la forma de enviar estos datos a Node-RED, para posteriormente en esta herramienta trabajar con ellos.

A partir de esto, una vez que se reciban los datos, se pondrán en marcha una serie de actuaciones, en este caso, el control automático del riego de tierra. Además de la función de riego automático, el usuario también podrá controlarlo de manera manual.

Al tratarse de una herramienta IoT, los datos generados son de gran importancia, por eso todos los datos generados serán almacenados en un historial, utilizando distintos gestores de almacenamiento de datos.

1.2 Abstract

In this Thesis, the Node-RED visual programming tool will be studied. Its internal functioning and the relationship it has with IoT devices. In addition to studying this tool, a practical case will be implemented to apply all the knowledge acquired.

It will detail the way of capturing the data provided by the sensors, how to send this data to Node-RED, and later in this tool work with them.

From this, once the data is received, a series of actions will be launched, in this case, the automatic control of land irrigation. In addition to the automatic watering function, the user can also control this watering.

Being an IoT tool, the data generated is of great importance, so all the data generated will be stored in a history, using different data storage managers.

2. Introducción

En la actualidad, cada vez más se tiende a la utilización de IoT (Internet of Things), ya que nos permite conectar objetos cotidianos, como puede ser un electrodoméstico de cocina, a Internet, pudiendo controlarlos desde cualquier parte del mundo en la que se tenga una conexión a la red. Los usuarios cada vez tienden más a la utilización de estas tecnologías, por la facilidad de uso que tienen, ya que con su propio smarthpone, pueden bajar la persiana de su casa, desde cualquier lugar del mundo que se tenga conexión a Internet.

Además del uso de IoT en las Smart Homes, unos de los usos más importantes, es en el ámbito de la industria, ya que permite a través de los sensores y actuadores, una gestión y monitorización automática de la maquinaria. Por otro lado, con el almacenamiento masivo de todos estos datos, se podría hacer algunas predicciones y tomar las medidas necesarias de forma remota.

En este TFG, la herramienta para gestionar y trabajar con los datos es Node-RED. La cuál, nos permite de una forma visual e intuitiva, recibir los datos y realizar las acciones necesarias a partir de ellos.

Una vez se han explicado los conceptos clave y se ha desarollado la guía para esta herramienta, en concreto, se establecerá un escenario en el cual se tienen sensores de temperatura, humedad en el aire, presión y humedad de suelo. Los sensores de humedad de suelo estarán tomando medidas en la tierra, para cuando esta esté seca, automáticamente se activen bombas de agua, que procederán a regar ese sector de tierra hasta que se encuentre húmedo de nuevo. A parte del regado automático de la tierra, el usuario podrá controlar en todo momento si la bomba está funcionando o no, además de ver los datos a tiempo real y almacenarlos en una base de datos.

Para la captación de datos se utilizará un microcontrolador Arduino Uno Wifi REV2, que, captará los datos de los sensores comentados anteriormente, y los enviará a un servidor Mosquitto mediante el protocolo MQTT. La herramienta Node-RED accederá

a los datos del servidor Mosquitto publicados por el microcontrolador, para trabajar con ellos.

Además de Arduino, Node-RED también publicará datos en este servidor Mosquitto, que serán recogidos por la placa Arduino, como por ejemplo la activación de la bomba de agua cuando la tierra se encuentre seca.

El usuario además de observar los datos en el momento, controlará todo el sistema mediante un bot de Telegram creado para ello. Por otro lado, estos datos serán almacenados en ThingSpeak.

2.1 IoT (Internet of Things)

El término “Internet de las Cosas” (IoT) fue empleado por primera vez en 1999 por el británico Kevin Ashton para describir un sistema en el cual los objetos del mundo físico se podían conectar a Internet por medio de sensores [1].

No existe ninguna definición única y universal de qué es o qué significa IoT, pero según Samuel Greengard, Internet de las cosas significa literalmente "cosas" u "objetos" que se conectan a Internet, y entre sí. Esto podría ser casi cualquier cosa: una computadora, tableta o teléfono inteligente, dispositivo de ejercicios, bombilla, cerradura de puerta, motor de avión, zapatos o casco de fútbol, por nombrar algunos. Cada uno de estos dispositivos o cosas tiene un número de identificación único (UID) y una dirección de Protocolo de Internet (IP). Estos objetos se conectan a través de cables, o tecnologías inalámbricas, incluidos satélites, redes celulares, Wi-Fi y Bluetooth [2].

2.2 Comunicaciones IoT

La forma más común de comunicación entre los dispositivos IoT es la llamada M2M (Machine to Machine). Este modelo de comunicación, como su nombre indica, permite que los dispositivos se comuniquen entre sí directamente, sin intervenir ninguna persona en el intercambio de información.

Los principales condicionantes que presenta este tipo de comunicación son, entre otros, que se tiene una gran cantidad de dispositivos, tanto actuadores, como sensores, además de un gran número de comunicaciones simultáneas. Por otro lado, se tiene que permitir la retirada e inclusión dinámica de los dispositivos, para tener una buena escalabilidad.

Otro gran condicionante, es permitir la interoperabilidad y el acceso fácil de los dispositivos, para que el funcionamiento sea efectivo en la mayoría de dispositivos, lenguajes de programación y sistemas operativos. Por otro lado, uno de los principales inconvenientes es que estos dispositivos están expuestos a Internet, un lugar poco seguro, ya que se está transmitiendo información privada, e incluso se están controlando sistemas físicos [3].

Una vez sabemos todos los requisitos de las comunicaciones M2M para que sean óptimas, la solución consiste en establecer un servidor central que tenga la función de recibir los mensajes de todos los dispositivos que generen los datos, y distribuir estos datos, a los receptores que deseen obtenerlos, este servidor recibe el nombre de “broker”.

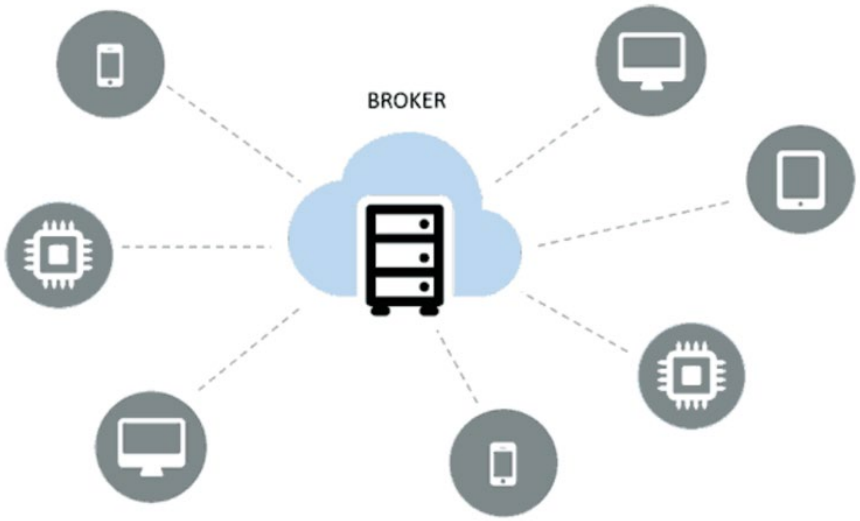

Ilustración 1. Esquema general de conexión IoT

Fuente: https://www.luisllamas.es/protocolos-de-comunicacion-para-iot/

Este servidor, si que tiene una dirección fija (no como los dispositivos que se conectan a él), permitiendo así una fácil accesibilidad a todos los dispositivos que quieran acceder o publicar información.

2.3 Metodologías IoT

Existen 2 metodologías a la hora de utilizar IoT, Router Remoder Procedure Calls (RRPC), en la cual un agente "Callee" proporciona un procedimiento, el cuál puede ser llamado por otro agente "Caller". El Router invoca el procedimiento en el "Callee", recoge el resultado del proceso, y lo comunica al "Caller" que lo ha invocado [4].

Por otro lado, el más usado es el Publish/Subscribe (PUBSUB), en el cual un agente "Subscriber", informa al broker que quiere obtener algún tipo de información que anteriormente haya sido publicada por el "Publisher".

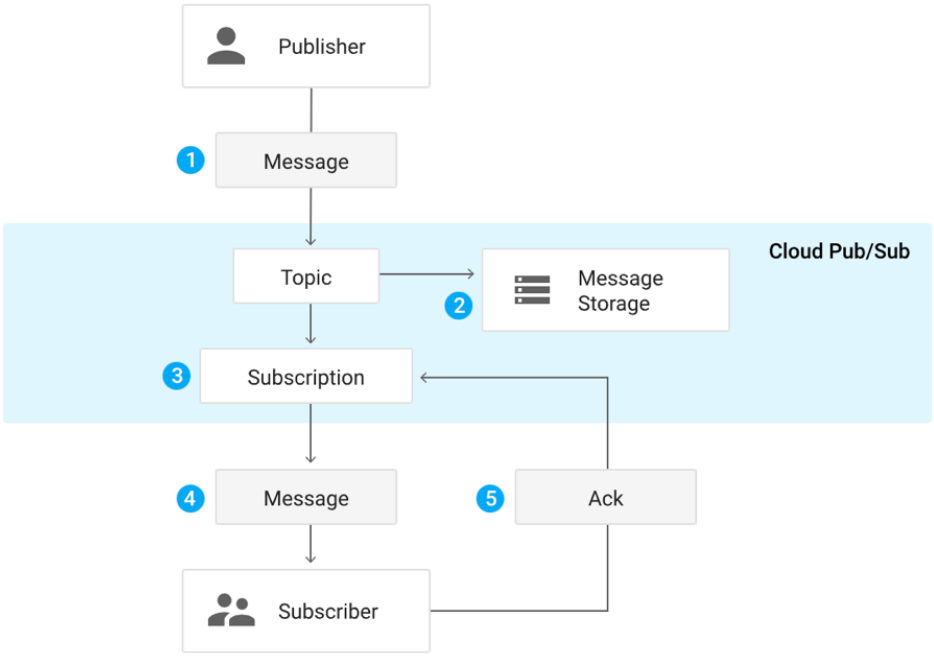

Ilustración 2. Esquema general Publicador/Subscriptor

Fuente: https://cloud.google.com/pubsub/docs/overview

2.4 Protocolos IoT

Hay muchos protocolos que se utilizan en la actualidad, dependiendo siempre del uso que se le vaya a dar. Por ejemplo, si es para un uso doméstico, en el cual la cantidad de datos que se van a intercambiar no va a ser muy grande, se usa un protocolo, o por el contrario si es para un uso industrial se usaría otro.

Pero en general, actualmente se destaca el uso de cuatro protocolos [5]:

- **AMQP (Advanced Message Queuing Protocol):** Es un protocolo que soporta las comunicaciones M2M y es del tipo comentado anteriormente, PUBSUB. Proviene y se utiliza sobretodo en el sector financiero, ya que tiene seguridad a través de cifrado y autenticación mediante Transport Layer Security (TLS) y Simple Authentication and Security Layer (SASL).
- **HTTP (Hypertext Transfer Protocol):** Su modo de funcionamiento es de tipo petición/respuesta y cliente/servidor. Su característica principal es que tiene una gran accesibilidad, ya que es un protocolo de código abierto. Es de gran utilidad usarlo en situaciones en las que se requiera enviar grandes cantidades de información, por ejemplo, la lectura de un sensor durante una hora, pero no es adecuado para un uso en el cual se requiera obtener información actualizada al segundo.
- **MQTT (Message Queuing Telemetry Transport):** Es uno de los más usados junto HTTP. Surgió para la comunicación entre máquinas y su funcionamiento es a través de PUBSUB. Se utiliza sobre todo para el envío de pequeños mensajes en poco tiempo. Este protocolo es explicado con más profundidad a continuación, en el apartado 4.1.
- **OPC-UA (Open Protocol Comunication - Unified Architecture):** De todos los nombrados anteriormente, es el que está mas orientado a la industria. Es un protocolo del tipo cliente/servidor, que permite a los usuarios conectarse, leer y escribir sobre la maquinaria industrial, obteniendo y publicando información al instante. Unas de las características diferenciales de este protocolo es su alto nivel de seguridad propia, es decir sin utilizar protocolos externos, como en el caso de HTTP que utiliza sobre

él SSL/TLS. OPC UA ofrece autenticación, cifrado y autorización de los datos mediante firmas. Actualmente, es el que más se utiliza en las grandes industrias.

3. Objetivos

El objetivo principal de este TFG, es el estudio y la obtención de información sobre la herramienta Node-RED, además de implementar un ejemplo de uso, en el cual se utilice para un caso práctico, en un escenario IoT.

La consecución de este objetivo principal está sujeta a los siguientes objetivos secundarios:

Se debe estudiar Node-RED desde el punto de vista de su funcionamiento, como surge y como funciona internamente. Para posteriormente realizar una guía y establecer comparaciones con herramientas similares.

Posteriormente una vez las bases son claras, se verá el uso de los nodos, tantos los incluidos por la propia herramienta, como el uso de nodos externos. Al tratarse de una herramienta enfocada a IoT, se entrará en este concepto en profundidad, explicando como funcionan estas comunicaciones y los protocolos y tecnologías que la forman.

Se explicará qué datos y de qué forma se han obtenido, para trabajar con ellos posteriormente. En este caso se utilizará el microcontrolador Arduino para esta función, acompañado de los respectivos sensores y actuadores. El agente intermediario entre Arduino y Node-RED es Mosquitto, el cual se comunicará con ambos mediante el protocolo MQTT. Por lo tanto, se explicará la estructura, las ventajas e inconvenientes de la utilización de este protocolo especializado en IoT.

Para recibir la información en Node-RED se explicará el uso de los nodos dedicados a MQTT, el uso de las variables en esta herramienta y como enviar y recibir información. Además del funcionamiento con Telegram, el programa encargado de que el usuario pueda tomar las decisiones que desee. Node-RED recibirá estas ordenes mediante comandos y ejecutará los nodos y herramientas necesarias para que se hagan efectivas.

Para finalizar, también se utilizará la herramienta ThingSpeak para almacenar un histórico de los datos obtenidos.

4. Materiales y métodos

4.1 MQTT

Una vez explicados los protocolos IoT más usados en la actualidad, el protocolo que se ha escogido para la comunicación en este TFG, es MQTT. Uno de los motivos principales, es que se trata de un protocolo abierto. Además, al ser uno de los protocolos IoT más usados, tiene su propia librería para la comunicación en Arduino y su propio bloque de base en Node-RED.

MQTT fue creado en el 1999 por el Dr. Andy Stanford-Clark de IBM y Arlen Nipper de Cirrus Link. La idea inicial era crear un protocolo para monitorear un oleoducto que atravesaba todo el desierto, que permitiera una comunicación eficiente, es decir que tuviera un consumo de ancho de banda bajo y que tuviera un consumo energético bajo.

Al principio fue un protocolo muy costoso, pero con el tiempo se ha convertido en un protocolo barato y abierto, ya que en 2010 IBM lo liberaría.

Las principales ventajas del uso de MQTT en la actualidad son las siguientes [6]:

- Alta escalabilidad, con capacidad de conectar un elevado número de clientes.
- Clientes independientes entre ellos, para que a la hora de tomar decisiones no repercuta la decisión de uno de ellos con los otros.
- Asincronismo.
- Es sencillo y ligero para que el consumo de recursos no sea demasiado elevado, aunque con el uso de seguridad TLS/SSL, aumenta.
- Eficiencia energética para que los dispositivos que funcionan con batería y están funcionando continuamente, no necesiten un gran ancho de banda.
- Buena seguridad y calidad, proporcionando así, una mayor fiabilidad en las comunicaciones.

4.1.1 Funcionamiento de la conexión

El protocolo MQTT usa una especie de filtro, para gestionar los mensajes que son enviados y recibidos por cada cliente. Se basa en topics que se organizan de forma jerárquica. De esta forma, un cliente puede publicar en el broker un mensaje, en un topic en concreto, y así, todos los clientes que estén conectado a ese broker y se suscriban a ese topic, recibirán el mensaje.

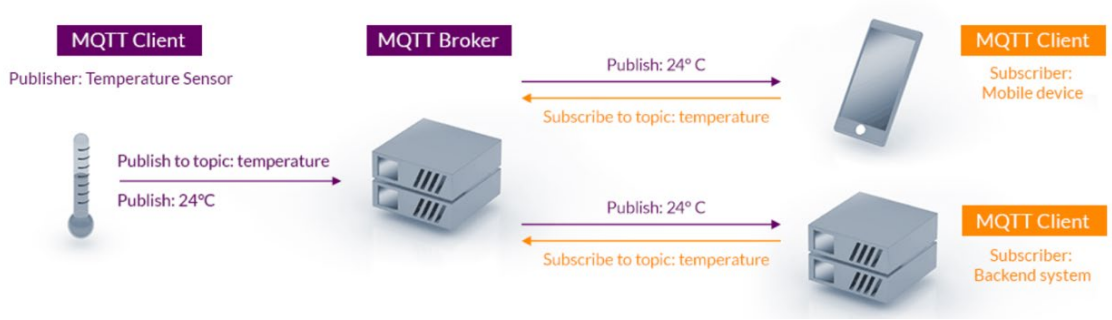

Ilustración 3. Arquitectura comunicación MQTT

Fuente: https://mqtt.org

Estos datos se mantendrán en cola y no se perderán hasta que el cliente no haya recibido el mensaje, ya que es un servicio de mensajería de tipo Message Qeue y se generará para cada uno de los clientes una cola independiente de mensajes. De este modo, si el cliente no está conectado y recibe un mensaje el broker lo almacena, y se lo entregará cuando se conecte el cliente.

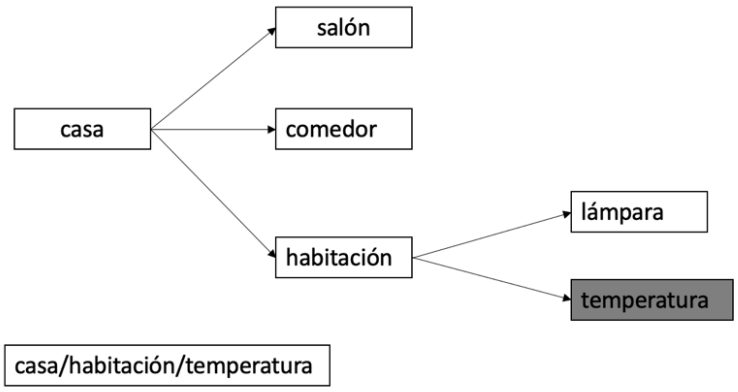

Ilustración 4. Uso de topics

La base para obtener y publicar la información en un broker MQTT es el uso de topics. Mediante este sistema se separa la información en niveles. Siguiendo el ejemplo de la ilustración 4, el nivel más alto es el denominado “casa” y a partir de este aparecen otros tres subniveles. Para la separación entre cada uno de los niveles se utiliza “/”.

Por otro lado, también se nos da la opción de acceder a los datos de varios topics a la vez, por ejemplo, si se quiere acceder al estado de la lámpara de la casa y la temperatura que hace se utilizaría “#”. Quedando de la siguiente forma: *casa/habitación/#.*

4.1.2 Formato del paquete

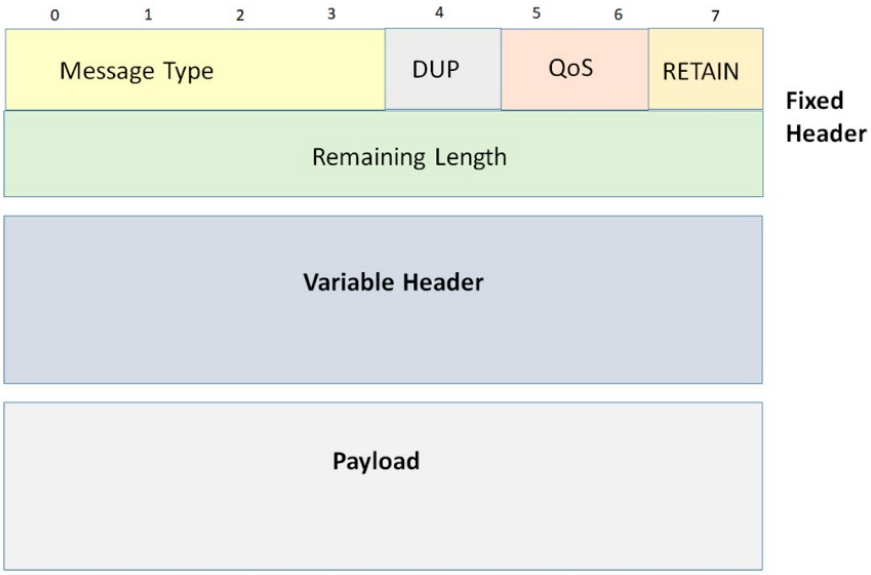

Ilustración 5. Estructura de un mensaje MQTT

Fuente: MQTT-AUTH: A TOKEN-BASED SOLUTION TO ENDOW MQTT, M.Calabretta

El formato de un mensaje MQTT es el representado en la ilustración 5. Se diferencia en tres grandes bloques, encabezado fijo, encabezado variable y carga útil.[7]

- **Encabezado fijo**: Es obligatorio y contiene los siguientes campos:
  - **Message Type**: Identifica el tipo de paquete. Algunos ejemplos de este mensaje pueden ser CONNECT, mediante el cuál se pone en marcha la conexión entre el publicador o subscriptor y el broker o PUBLISH, mensaje que se utiliza para recibir datos o almacenarlos en el broker. Por otro lado, mediante el mensaje SUBSCRIBE, se

realiza una petición al broker para acceder al contenido de el topic al que se quiere acceder.

- **DUPLICATION:** Es un bit el cual puede estar activado o desactivado. Indica que un mensaje ya ha sido recibido.
- **QoS:** Son 2 bits por los cuales se codifican tres tipos diferentes de calidades de servicio. Serán explicadas en el apartado 4.1.3.
- **RETAIN**: Es un bit el cual puede estar activado o desactivado. Indica al broker si se desea mantener el mensaje y la QoS indicada para futuros subscriptores.
- **Remaining Length:** Indica el número de bytes que faltan en el mensaje.

- **Encabezado variable:** Normalmente se utiliza para añadir información de control. Por ejemplo, en un mensaje del tipo CONNECT, este encabezado contendrá la versión del protocolo MQTT, el usuario y contraseña, etc.
- **Carga útil:** Contiene la información que se quiere transmitir o recibir de un topic a través de un broker.

4.1.3 Calidad y seguridad

En todo sistema de comunicación, es importante asegurar una buena calidad de servicio, con el fin de que sea un sistema estable y sin fallos.

En cuanto a la calidad de servicio (QoS) del protocolo MQTT, se diferencian 3 niveles:

- **QoS 0 (unacknowledge):** Se usa sobre todo cuando el servicio no es crítico. El mensaje se enviaría una sola vez, y en el caso que ocurra algún fallo, no se entregaría de nuevo.
- **QoS 1 (acknowledge):** El mensaje será enviado todas las veces que sean necesarias, hasta que se confirme que se ha entregado al cliente. El inconveniente que supone esta calidad de servicio, es que el cliente puede recibir un mismo mensaje varias veces.

- **QoS 2 (assured):** Se usa en sistemas críticos, donde es necesario un mayor nivel de fiabilidad. Es similar a QoS 1, la mejora que supone, es que se garantiza que el mensaje se reciba una sola vez.

En cuanto a la seguridad, MQTT utiliza el estándar SSL/TLS. Lo cual proporciona que a nivel de transporte se encripten los datos. Por lo tanto, cuando los datos se están transmitiendo, estos no pueden ser leídos. Solo serán leídos, cuando lleguen al usuario y proporcione sus datos de autentificación correspondientes, verificando así la identidad de ambas partes. [8]

4.1.4 Brokers

En los anteriores apartados, se ha explicado qué es y como funciona un broker en general. MQTT en su página [9], nos indica una larga lista de los brokers que hay disponibles que soporten este protocolo. Entre los que destacan Ably MQTT broker, Akiro o Apache ActiveMQ. Brokers muy potentes que actualmente son usados por la industria y todo tipo de escenarios.

En el caso de este TFG, se ha optado por la utilización del broker Eclipse Mosquitto. Un servidor de código abierto y uno de los más usado debido a su ligereza, ya que nos permite incluirlo en cualquier situación y escenario, aunque esta esté dotada de pocos recursos. Es uno de los más adecuado para el IoT de mensajería, como es el caso de sensores de baja potencia y microcontroladores como Arduino [10].

El proceso de instalación está basado sobre todo para plataformas Linux, pero también está disponible para el resto, como pueden ser Windows o MacOS.

4.2 Arduino

En los apartados anteriores, cuando se habla de envío y recepción de datos a través de un broker, se refieren a datos que se generan a partir de las medidas de los distintos sensores. El hardware y software encargado de la captación de los datos y el envío al broker, es el microcontrolador Arduino.

Arduino [11], es una plataforma de desarrollo que está basada en una placa electrónica PCB (Printed Circuit Board), las cuál, es la forma más compacta y estable de construir un circuito electrónico. Es de hardware libre e incorpora un microcontrolador re-programable y una serie de pines hembra, para la conexión de los comentados sensores o actuadores.

No es la única placa capaz de realizar estas acciones, otros microcontroladores, a parte de Arduno son el Teensy 3.6, el Netduino N3, SparkFun Thing Plus o Adafruit Feather Huzzah. Los principales motivos que han llevado a la realización de este TFG con un microcontrolador Arduino, son los siguientes:

- **Alcance:** Al ser uno de los microcontroladores más utilizados, se ha generado una gran comunidad que trabaja con esta plataforma, generando así bastante documentación.
- **Multiplataforma:** Su entorno de programación es multiplataforma.
- **Lenguaje de programación sencillo:** Su lenguaje de programación es de fácil comprensión, basado en C++. Haciendo posible que sea utilizado por nuevos programadores.
- **Versatilidad:** Se trata de una placa re-utilizable, ya que cuando se ha terminado un proyecto, se quieren hacer modificaciones o ha ocurrido algún error, se pueden desmontar los componentes conectados a los pines de la placa y empezar de nuevo o modificar alguno de ellos.

4.2.1 Modelos

Cuando se habla del hardware, hay muchos tipos de placas Arduino, pero en general se distinguen en 3 grandes niveles. Las placas de nivel de iniciación, las cuales son las ideales para comenzar con proyectos más sencillos y familiarizarse con el entorno. En este grupo se encuentra la placa más utilizada y conocida, la Arduino UNO. En segundo lugar, se encuentran las placas de nivel avanzado. Las cuales, nos permiten crear proyectos con un mayor alcance, ya que añaden funcionalidades extra a las placas de

iniciación como puede ser, un número mayor de conexiones para los sensores y actuadores u otro tipo de conexiones, como tomas RJ-45 para poder conectar la placa a la red.

En tercer lugar, se encuentran las placas para IoT. En este grupo se encuentra la placa que vamos a utilizar para este proyecto, la Arduino UNO WiFi REV2. Es similar a la comentada anteriormente Arduino Uno, pero con las funcionalidades recomendadas por Arduino para la creación de proyectos de IoT como este.

El módulo WiFi, en concreto se utilizará para los datos que capten los sensores, enviarlos al servidor Mosquitto.

4.2.2 Estructura

El hardware de la placa Arduino UNO WiFi REV2, está formado por los siguientes componentes [12]:

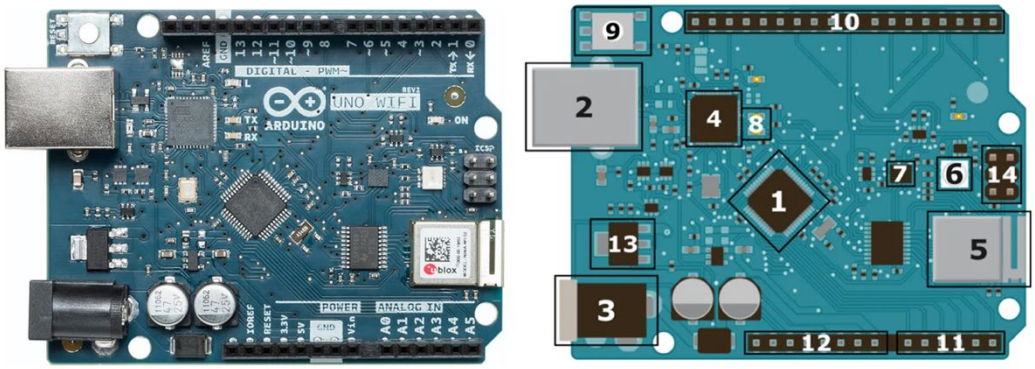

Ilustración 6. Estructura de una placa Arduino UNO WiFi REV2

Fuente: https://iot-guider.com/arduino/hardware-basics-of-arduino-uno-wifi-rev2/

1. **Micro-controlador**: En concreto, se utiliza el ATMEGA4809. El cuál nos presta una memoria flash programable de 48 KB, 6114 Bytes de SRAM (Static Random Acces Memory), consiguiendo que una vez que los datos se almacenen se mantengan hasta que se deje de alimentar la placa, 256 Bytes EEPROM (ROM programable y borrable eléctricamente) no volátil.
2. **Conector USB:** Es un conector USB del tipo B. Tiene dos funciones, una es conectar la placa con el ordenador para poder programarla mediante el Arduino IDE y otra es alimentar la placa.

3. **Puerto de Alimentación**: Se utiliza para alimentar a la placa desde una fuente externa.
4. **ISP Flash y Controlador USB:** Permite que la placa sea programada mientras está instalado el sistema de sensores completo, es decir, no es necesario que el chip sea programado antes de que se incluya en el sistema.
5. **Módulo WiFi**: En concreto utiliza el NINA-W102, el cuál utiliza el protocolo WiFi 802.11b/g/n y permite el modo dual con Bluetooth 4.2.
6. **RGB LED**: Esta placa tiene incluido un LED RGB con una resistencia que puede ser programado como si se tratase de un LED externo. Lo usaremos posteriormente como ejemplo.
7. **IMU (Inertial Measurement Unit):** Se utiliza para acelerómetros, giroscopios, sensores de temperatura y sensores de 3 ejes.
8. **RT/TX LED:** Estos dos LEDs, se encienden cuando se transmite información desde el Arduino IDE del ordenador, hasta la placa.
9. **Botón de Reseteo**: Resetea por completo todos los ajustes de la placa. estableciendo todos sus atributos de fábrica.
10. **Pines Digitales:** Esta placa tienen un total de 14 pines digitales, especificados, desde D0 a D13. 6 de ellos, en concreto los número 3, 5, 6, 9, 10 y 11 tienen el símbolo "~" lo que significa que son pines PWM. PWM (Modulación por ancho de pulso) nos permite emular una señal analógica en la salida digital. Su uso principal es para las entradas y salidas digitales.
11. **Pines Analógicos:** La placa tiene un total de 6 pines analógicos de entrada, especificados desde A0 a A5. Se utilizan para captar las señales de los pines analógicos y los conviertre a un valor digital.
12. **Pines de Alimentación:**
    a. **Vin:** Se utiliza para alimentar a la placa desde una fuente externa.
    b. **5V:** Proporciona una salida regulada de 5V, que alimentará a los sensores.
    c. **3V3:** Es de las mismas características que la anterior, pero de 3,3V.
    d. **GND:** Toma a tierra de los sensores.
    e. **IOREF:** Proporciona la referencia de tensión con la que funciona el microcontrolador.

13. **Regulador de Voltaje**: Tiene la función de regular la entrada y salida del voltaje de la placa, además de regular el voltaje para 5V y 3,3V de los pines.
14. **Pines de Cabecera ICSP**: ICSP(In Circuit serial Programming), se utiliza para progamar el BootLader del microcontrolador para poder cargar los programas o recuperar un programa sin conectar una fuente de programación externa.

4.2.3 Arduino IDE

La placa Arduino y todo su hardware, siempre va a acompañada de su software Arduino IDE (Integrated Development Enviroment). Es el entorno de programación el cuál va a editar y compilar el código que posteriormente se subirá a la placa.

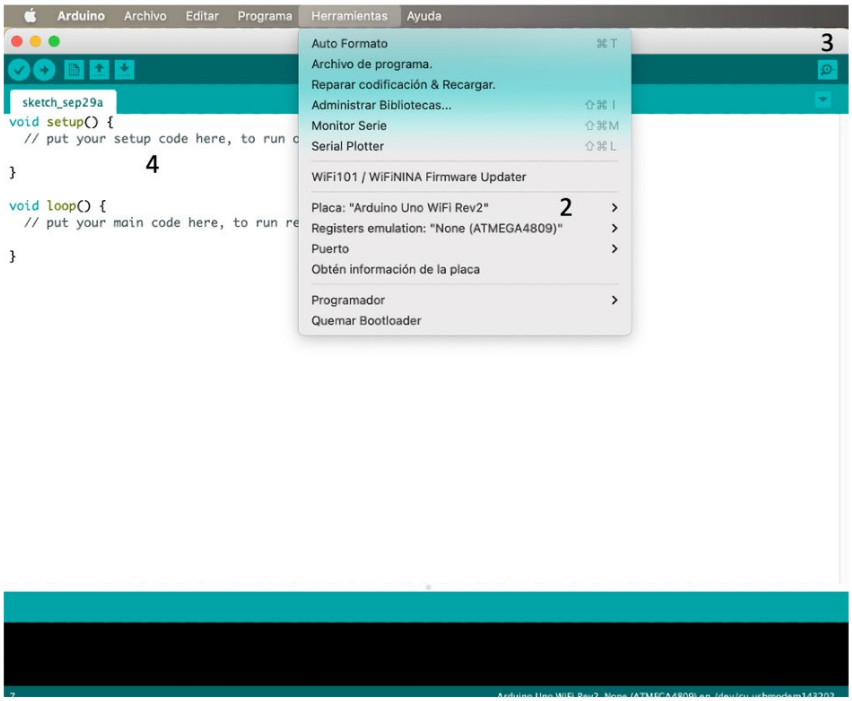

Ilustración 7. Interfaz Arduino IDE

1. En primer lugar, en la esquina superior izquierda, tenemos los botones para comprobar que el código funciona correctamente y el más importante, la flecha que actúa como compilador y envía las sentencias al microcontrolador.

2. Por otro lado, en la parte superior está el menú, en el cuál, se pueden instalar librerías externas, seleccionar el puerto en el que se encuentra la placa y seleccionar el modelo concreto del que se trata.
3. En la esquina superior derecha, se encuentra el botón para desplegar la respuesta de la consola.
4. Para finalizar, en el centro se encuentra el procesador de texto para escribir el código que haga funcionar el programa. Por defecto se generan las funciones setup, para establecer las variables principales y todo lo que se quiera que se ejecute nada más enviar el programa, y loop, la cual ejecutará el código que hay en su interior hasta que se ordene lo contrario. El periodo de ejecución viene por el comando delay, y a continuación los milisegundos.

4.3 Node-RED

En este escenario, el microcontrolador actuará principalmente como publicador en el servidor Mosquitto. La herramienta que hará de subscriptor y que recogerá los datos publicados es Node-RED.

Node-RED es una herramienta de programación visual mediante la cual se le permite al usuario programar sin escribir código, ya que muestra visualmente las relaciones y las funciones. Está basado en navegador y permite añadir, eliminar y conectar entre sí nodos con el fin de que se comuniquen entre ellos.

Surgió a principios de 2013, a partir de un proyecto en paralelo entre Nick O'Leary y Dave Conway-Jones del Grupo de Servicios de Tecnologías Emergentes de IBM. Todo comenzó con la prueba de un concepto para visualizar y manipular topics de MQTT que poco a poco fue mejorando y se convirtió en un proyecto que se extendió para varias direcciones. Para que oficialmente en septiembre de 2013 se convirtiera oficialmente en una herramienta de código abierto [13].

A día de hoy es una de las herramientas que más se usan en el ámbito de IoT por los siguientes motivos:

- Es de código abierto y sigue estando mantenido por IBM.
- Se puede incluir en cualquier sistema operativo ya que funciona con el navegador.
- No hay que profundizar en lenguajes de programación, ya que es muy intuitivo, lo que hacen que se puedan crear prototipos rápidamente.
- Evita las tareas repetitivas, ya que se pueden realizar simultáneamente varios procesos.

4.3.1 Fundamentos

Node-RED está creado a partir de NodeJS y la librería de JavaScript D3.js [14]. NodeJS es un software muy potente que hace posible la programación del lado del servidor en JavaScript, lo que proporciona la potencia necesaria para que sea fiable y sobre todo escalable. Es un entorno que se controla mediante eventos, que se ha diseñado para crear aplicaciones escalables. Una de sus principales ventajas es que está creado de una forma óptima, para poder soportar múltiples conexiones simultáneas.

Node.js se ha creado sobre el motor de JavaScript V8 de Google Chrome, lo que proporciona una gran velocidad a la hora de ejecutar el código

Por otro lado, D3.js se encarga de proporcionar la interfaz gráfica de la web.

4.3.2 Funcionamiento

Node-RED se basa en el concepto de FBP (Flow-Bassed Programming). La programación basada en flujos es una forma de programación a partir de una red de nodos. Se tratan de una especie de cajas negras en las que se ocultan código de Node.js y tienen un propósito diferenciado. Por ejemplo, un nodo captura los datos y los envía a otro nodo para que seleccione algunos de ellos mediante un filtro.

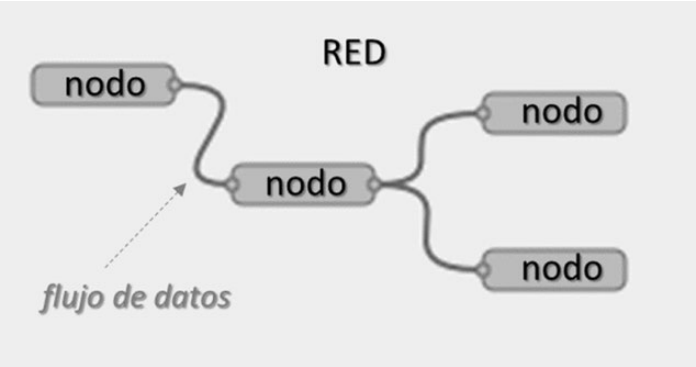

Ilustración 8. Red y flujo de datos entre nodos

Fuente: https://www.techedgegroup.com/es/blog/fundamenos-Node-RED

Según los tipos de entradas y salidas que tengan, se pueden clasificar tres tipos de nodos [15]:

- **Iniciadores:** Son nodos que inician un flujo de datos, únicamente poseen un puerto de salida. Pueden ser de dos tipos:
  - Activos: Se activan de una forma periódica bajo la supervisión de un evento determinado. Por ejemplo, que se envíe la fecha actual cada cinco segundos.
  - Pasivos: Están a la escucha de algún evento. Es el caso de MQTT. Este nodo está a la espera de que se actualice un topic en el broker, y cuando lo hace recibe este dato actualizado.
- **Intermedios:** Son nodos que reciben un flujo de datos, realizan las funciones pertinentes dependiendo del nodo que se trate y generan una salida con el resultado de esa operación. Por lo tanto, tienen un puerto de entrada y otro de salida. En este tipo de nodos, cabe hacer referencia al "fuction node", el cual nos permite incorporar directamente código propio de JavaScript para implementar nuestra propia función.
- **Terminales:** Se sitúan al final de las ramas del flujo de datos comentado anteriormente, por lo tanto, posee únicamente una salida. Se utiliza, sobre todo, para ejecutar otros flujos de la aplicación y para generar eventos, por ejemplo, el envío de datos al broker.

Por defecto, en Node-RED vienen incluidos los siguientes nodos:

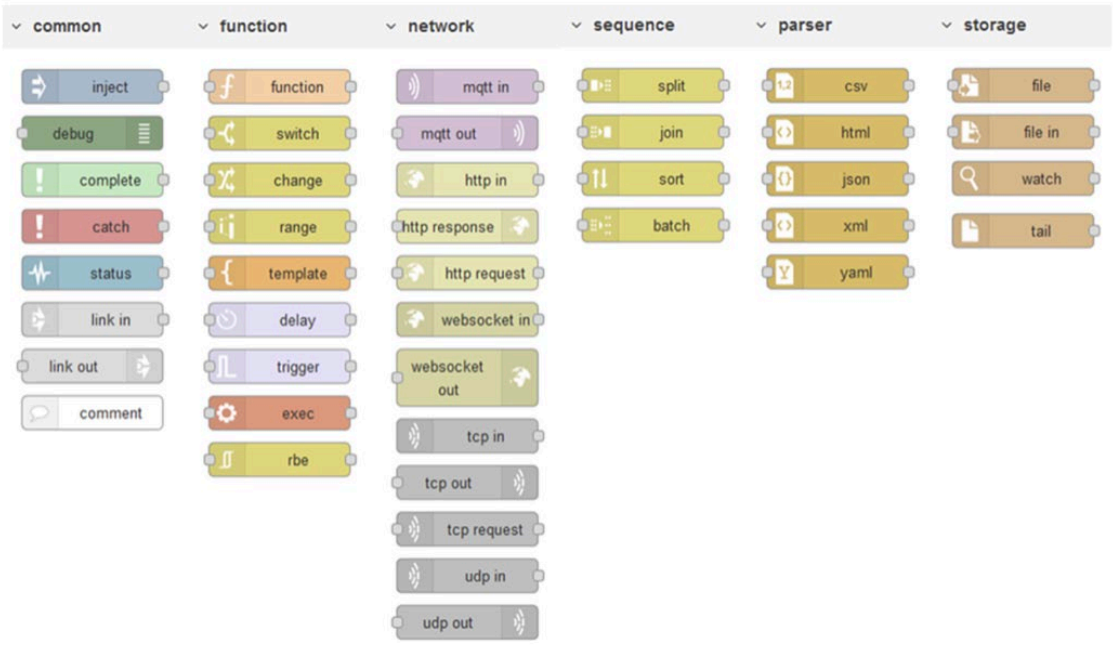

Ilustración 9. Nodos por defecto en Node-RED

A parte de la utilización de los nodos que vienen por defecto, unas de las mayores ventajas que presenta Node-RED, es su herramienta para incluir nodos externos creados tanto por la comunidad, como por aplicaciones externas. Actualmente hay disponibles, como se ve en la ilustración 10, 3486 módulos.

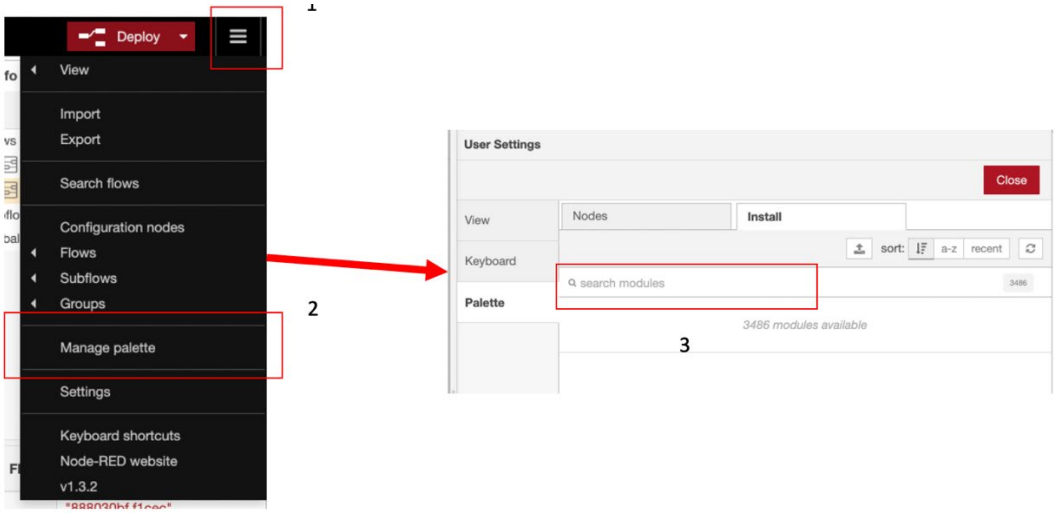

Ilustración 10. Instalación de nodos externos

4.3.3 Entorno de uso

El entorno de uso se puede dividir en las partes mostradas en la siguiente ilustración:

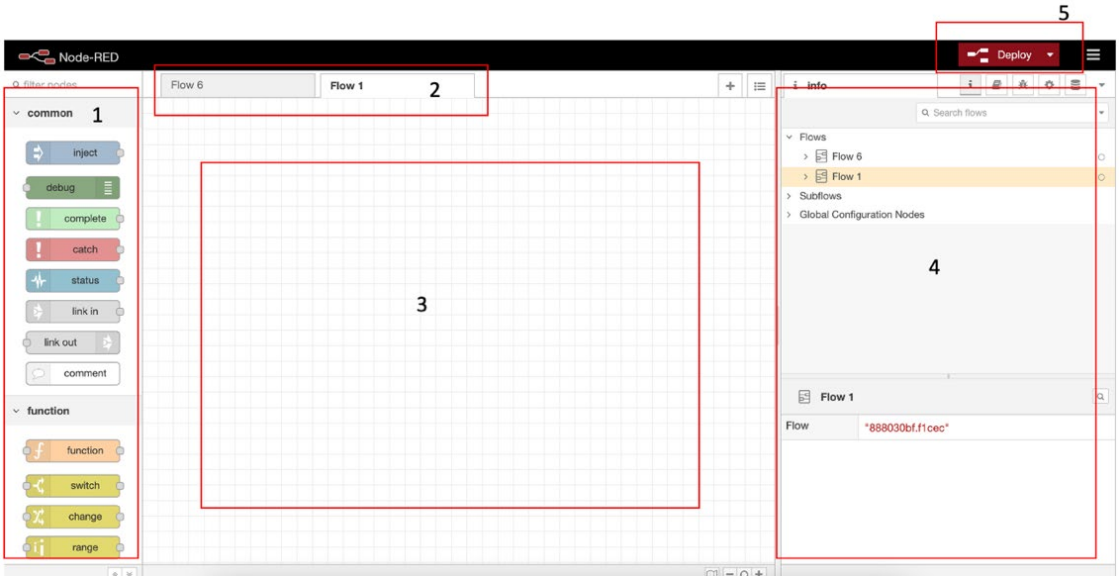

Ilustración 11. Entorno de uso de Node-RED

1. En el lado izquierdo de la aplicación se encuentran todos los nodos que se pueden utilizar, tanto los propios incluidos por Node-RED, como los que se han añadido de forma externa, organizados por tipos.
2. En la parte superior se encuentran las pestañas para acceder a los distintos flujos. Se pueden crear el número de flujos que quiera el usuario y pueden actuar conjuntamente o de forma independiente.
3. En la parte central, se encuentra el espacio de trabajo, hasta el cual se van a arrastrar los nodos que se vayan a utilizar.
4. En la parte derecha se encuentra el apartado de las herramientas. En este apartado se puede ver una pequeña guía de cada uno de los nodos, se muestra la respuesta que se obtiene por consola y las variables almacenadas junto a su valor.
5. El botón "deploy" hace la función de compilador. Despliega un menú, en el cual se da la opción de compilar todos los flujos, compilar solo el flujo actual que se está mostrando o por otro lado reiniciar todos los flujos y variables.

4.3.4 Guía

Una vez explicado el funcionamiento, los fundamentos y el entorno de uso, para cumplir con los objetivos de este TFG, se desarrollará una guía introductoria al uso de esta herramienta.

El programa más sencillo que se puede crear con esta herramienta, es imprimir por la ventana de depuración la fecha actual. Para ello, en primer lugar, se debe seleccionar y arrastrar el nodo “inject”, incluido dentro del paquete de nodos “common”. Este nodo, podrá inyectar los tipos de datos que se muestran en la ilustración 12, en este caso, como se quiere mostrar la fecha actual, se seleccionará “timestamp”.

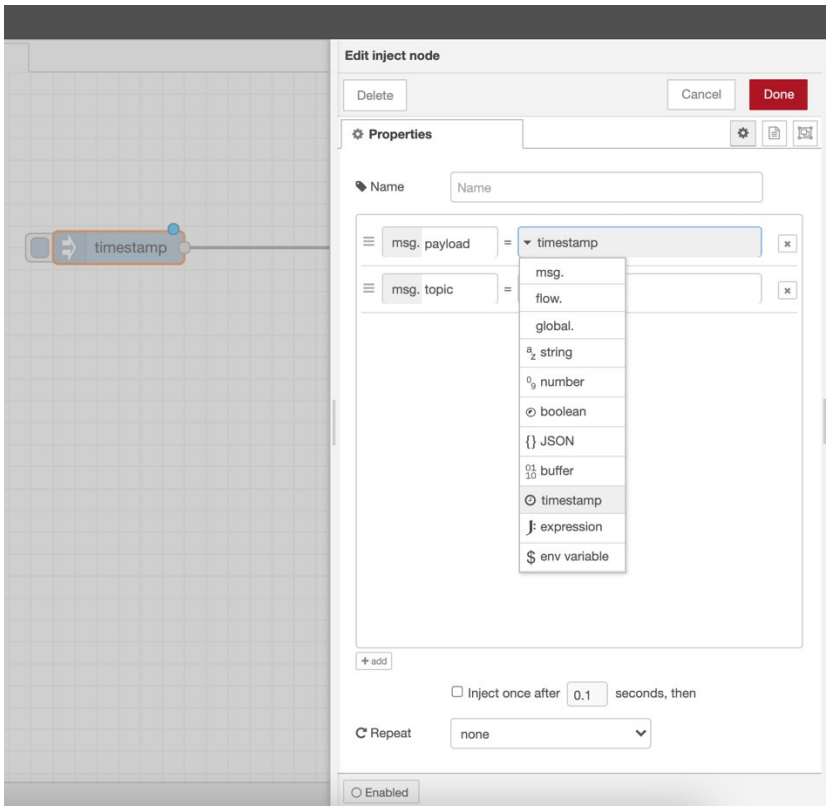

Ilustración 12. Uso Timestamp en Node-RED

El nombre de la variable “msg.payload”, es el que se genera por defecto a la salida de cualquier nodo, y, por lo tanto, el que recibirá el nodo “debug” para mostrar la información. Quedando el esquema de la siguiente forma:

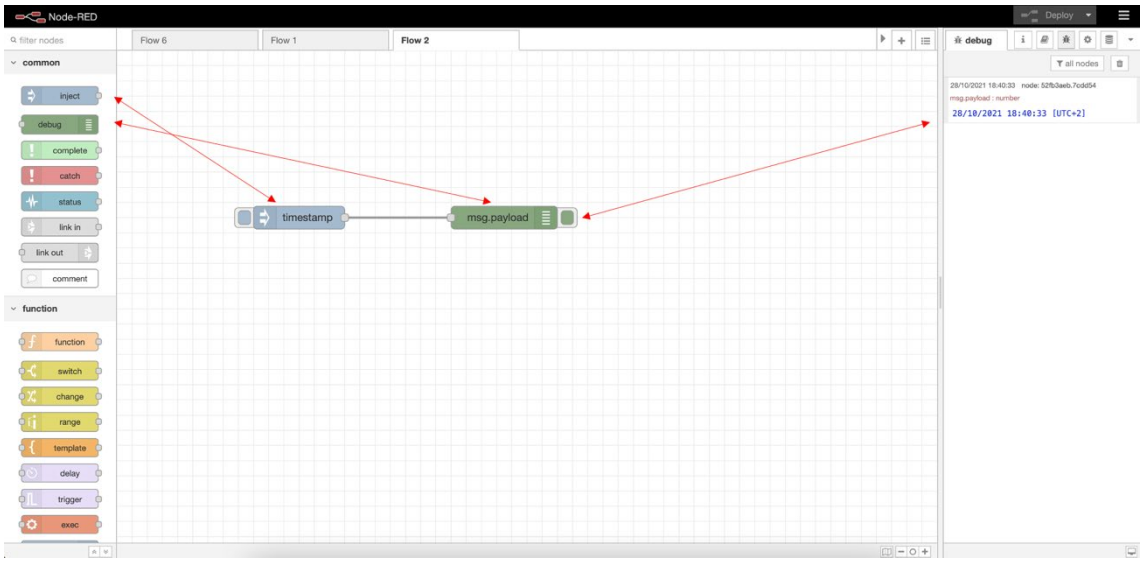

Ilustración 13. Uso ventana de depuración en Node-RED

4.3.4.1 Eliminar el valor de una variable

Por otro lado, en el caso de que queremos eliminar esta fecha generada, se utilizaría el nodo "change", incluido dentro del paquete de nodos "fuction". Este nodo, nos permitirá cambiarle el nombre a una variable, copiar o establecer su valor o cambiar su contexto, pero esto se explicará a continuación de forma detallada en el apartado 5.4 . Por ahora se utilizará solo la función de eliminar.

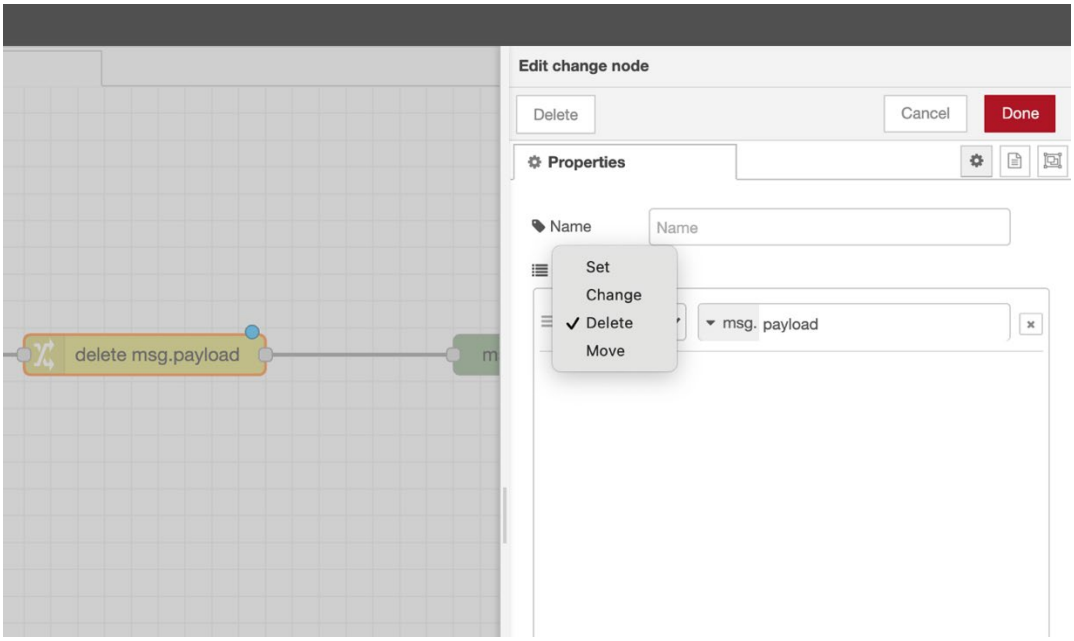

Ilustración 14. Menú nodo "change" de Node-RED

Por lo tanto, ahora, cuando se introduce el valor de la fecha, el nodo “change” elimina este valor y al nodo “debug” no le llega nada. Como se ve en la ilustración 15, el valor es “undefined”.

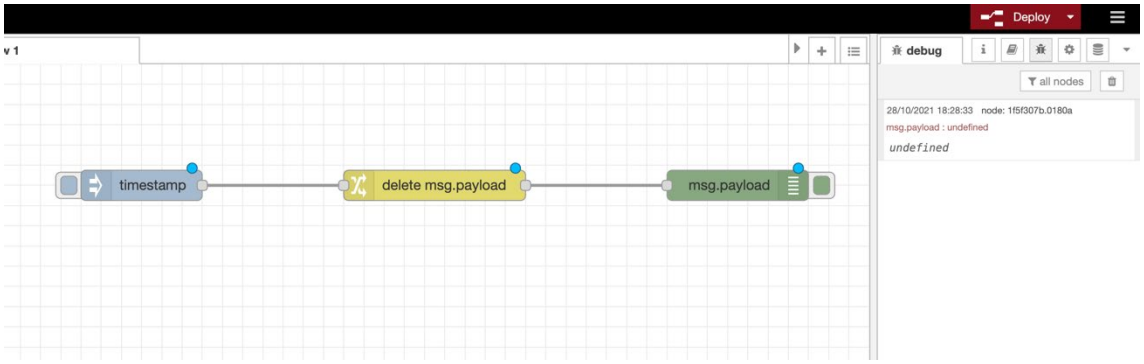

Ilustración 15. Eliminar un valor de una variable de Node-RED

4.3.4.2 Rangos

Otro nodo de gran utilidad es el denominado “range”, también incluido en el paquete “fuction”. Este nodo nos permitirá establecer un escalado de los números, es decir, por ejemplo, si tengo un rango entre 0 y 1023, se escale a un rango entre 0 y 5.

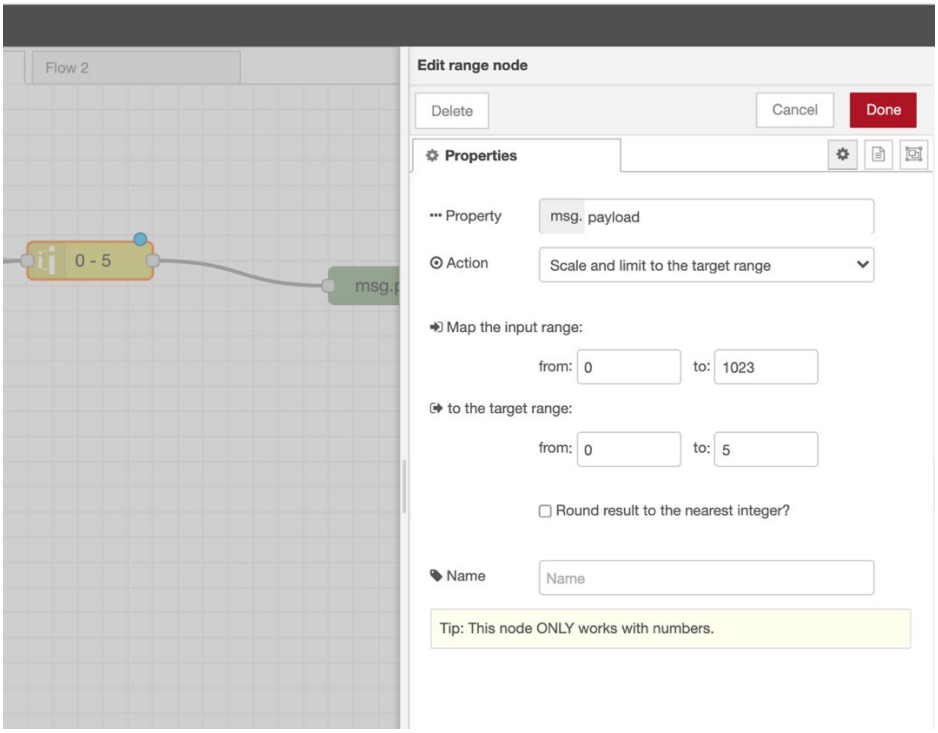

Ilustración 16. Menú nodo “range” de Node-RED

Una vez se ha rellenado el bloque con los valores deseados, por ejemplo, en el caso de que se introduzca un 0 se devolverá también 0. Pero en el caso de que se introduzca la mitad de estos valores, es decir 512, también se devolverá la mitad del rango al que se quiere escalar, en este caso 2,5. Por otro lado, si se introduce el máximo valor de un rango, también se devolverá el máximo valor del otro. Como se muestra a continuación:

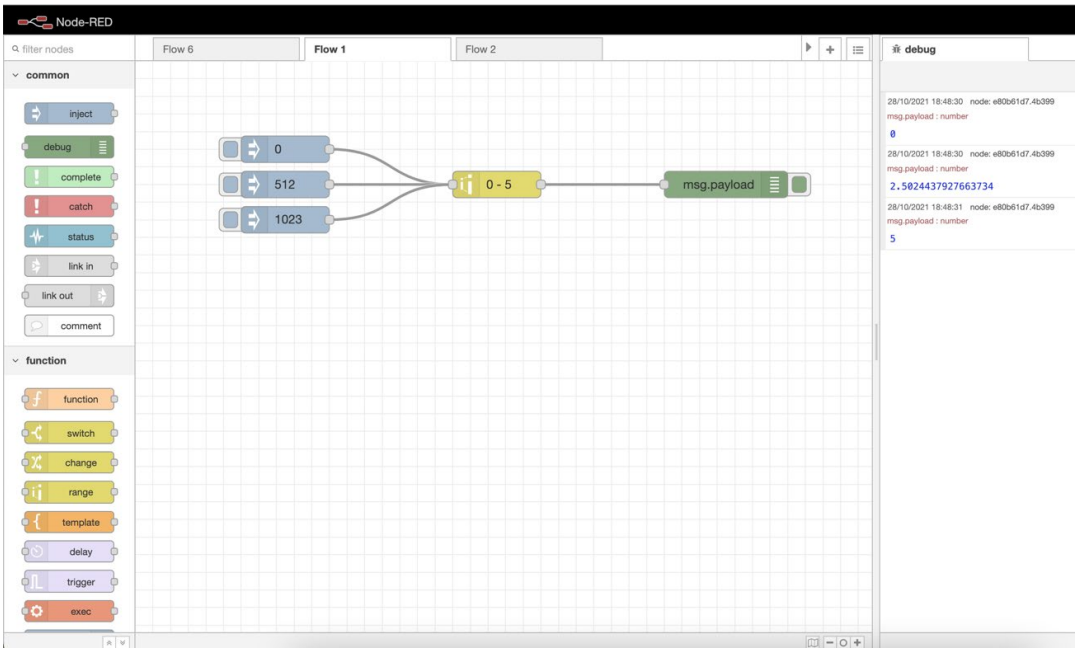

Ilustración 17. Cambiar rango de números en Node-RED

4.3.4.3 Arrays

Por otro lado, también cabe destacar el uso de arrays en Node-RED. En este caso, se plantea la situación en la que se introduce un array de números decimales y se quieren aproximar todos ellos al entero más próximo.

En primer lugar, se introducirá el array, para que posteriormente, este array sea recibido por el nodo "split" del paquete de nodos "secuence". Este nodo lo que hará es separar el mensaje en varios mensajes con cada uno de los números.

Una vez se han separado en diferentes mensajes los números del array, como se muestra en la ilustración 18, serán recibidos por el nodo comentado anteriormente, el nodo "range", y se seleccionará la función que tiene de redondear al entero próximo, como se ve en la ilustración 19.

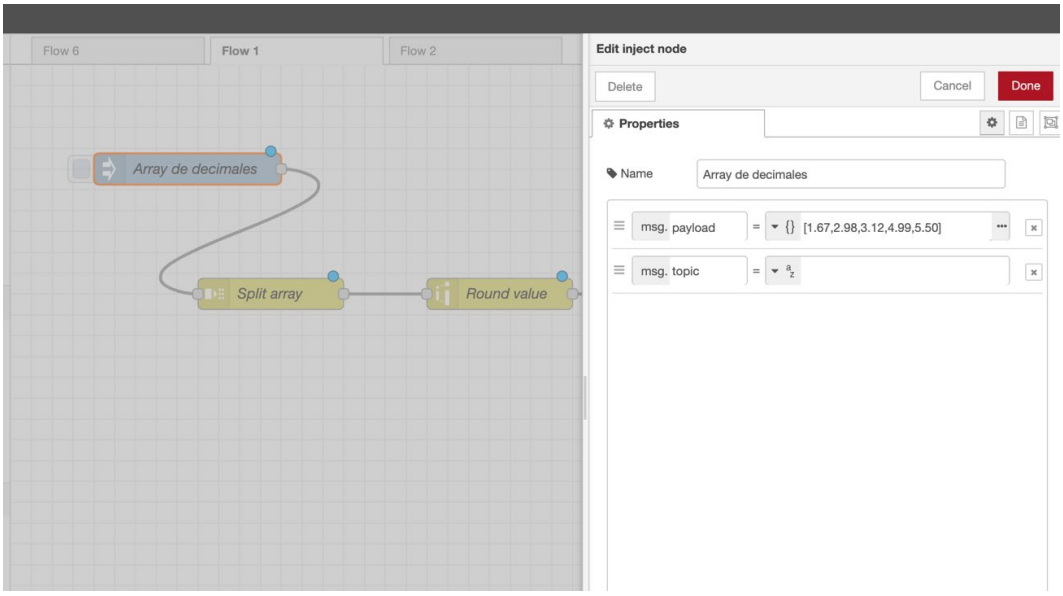

Ilustración 18. Envío de array con número decimales en Node-RED

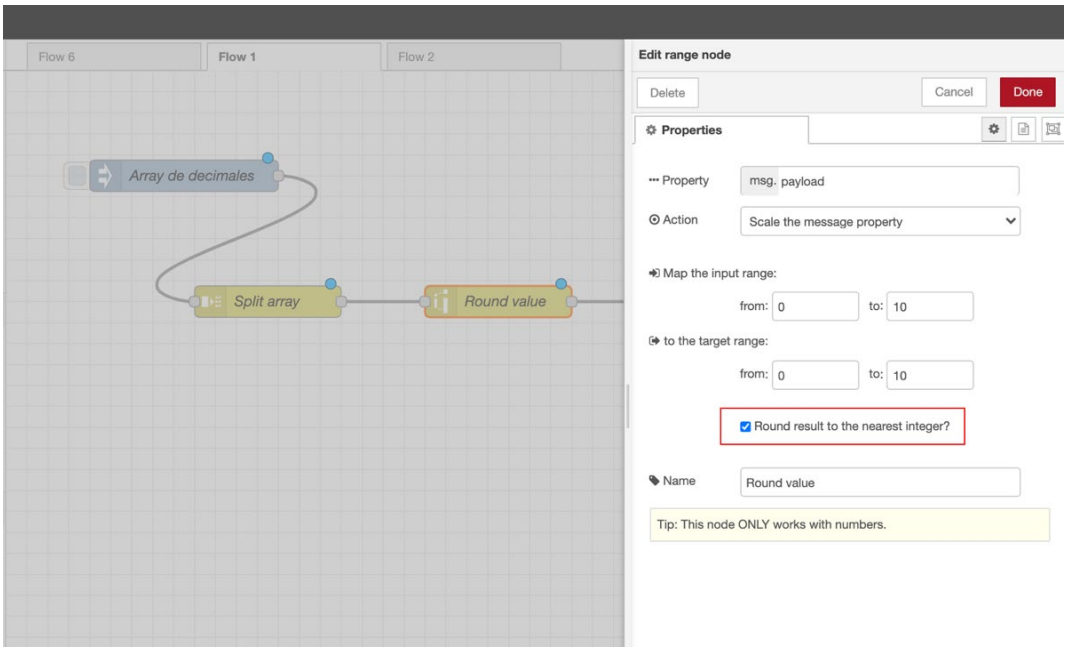

Ilustración 19. Establecer el redondeo al entero próximo en Node-RED

Una vez se han realizado estas acciones, para juntar de nuevo todos los números que habían sido separados en mensajes independientes, en un mismo array, se utilizará el nodo "join" del paquete de nodos "sequence". Quedando de la siguiente forma:

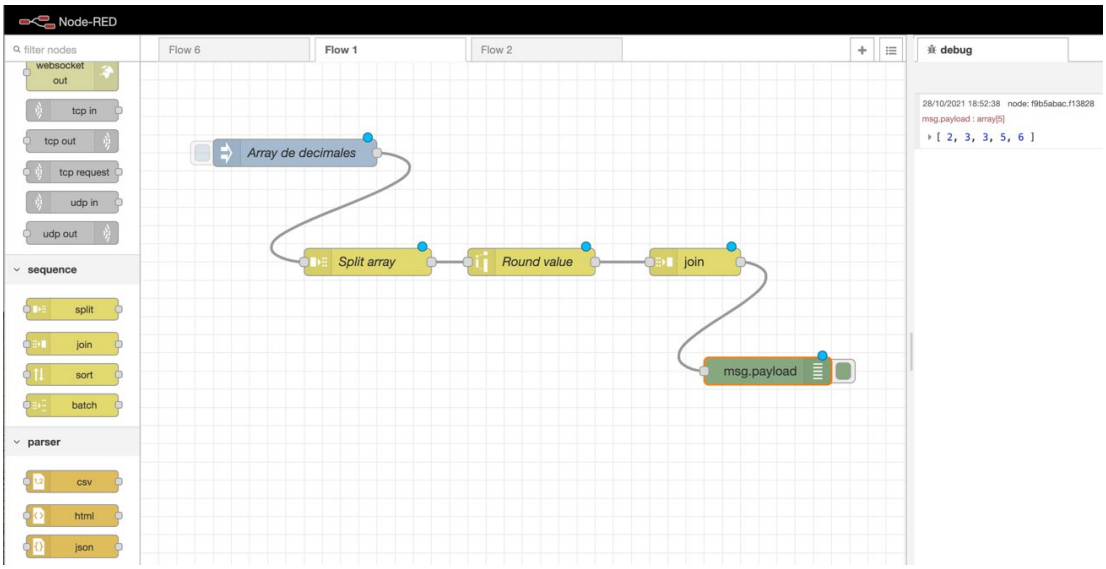

Ilustración 20. Redondeo array de decimales en Node-RED

4.3.4.4 Detección de cambios

También existe un nodo denominado "report-by-exception", el cual nos permite detectar si se produce algún cambio en la entrada. Por ejemplo, si se está introduciendo continuamente el número 0, la salida no cambiará hasta que no se produzca un cambio, y solo se mostrará el primer 0 introducido, hasta que no se introduzca otro. Cuando se introduzca cualquier otro número, ocurrirá lo mismo, solo se cambiará la salida si cambia de número.

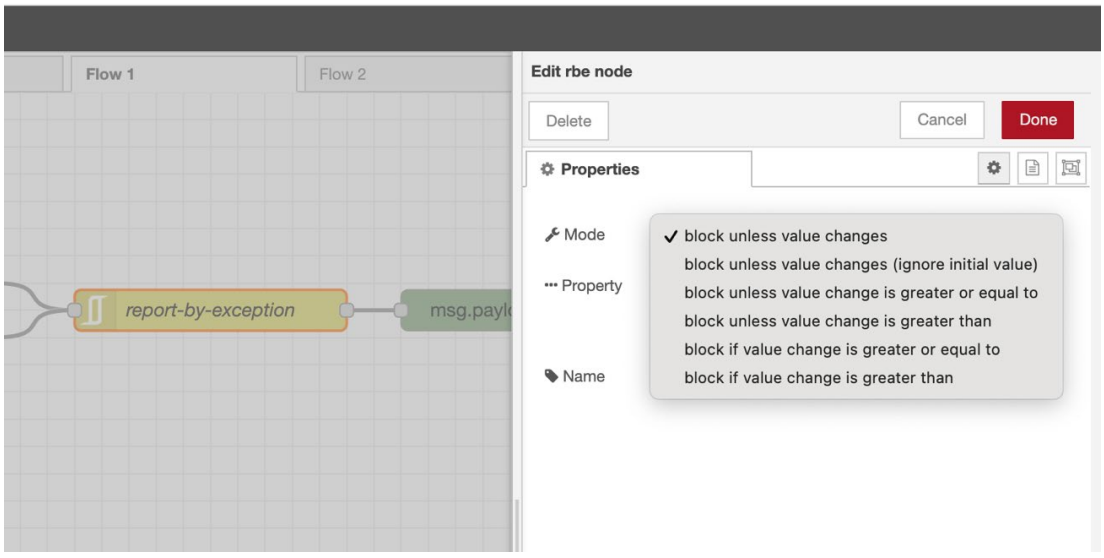

Ilustración 21. Menú del nodo "rbe" de Node-RED

Este nodo puede funcionar de diferentes modos, pero el que vamos a utilizar, es el primero, el cuál realizará la acción que se ha comentado anteriormente. Solo enviará una salida cuando se cambie de número. Si se envía el mismo número se bloqueará la salida. En la ilustración 22, se muestra un ejemplo con el número 0 y 1.

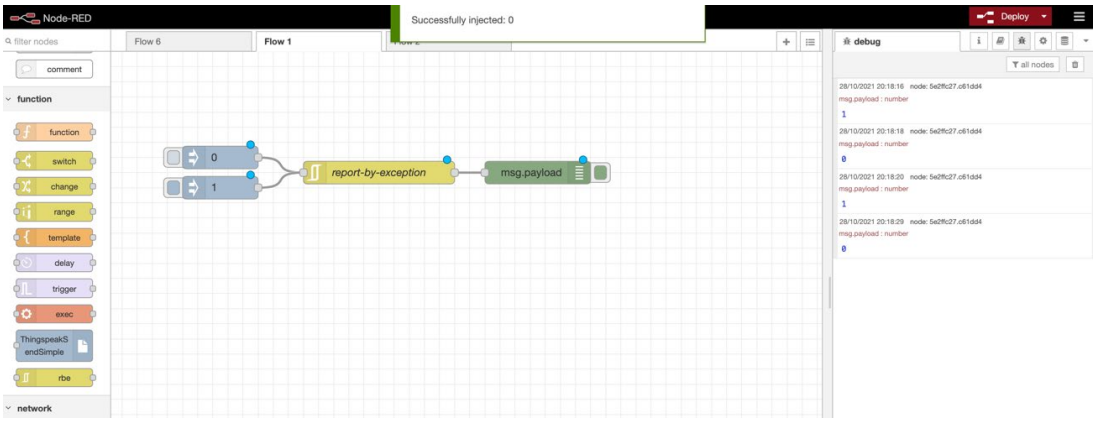

Ilustración 22. Detección de cambios de parametros en Node-RED

En este apartado se ha desarrollado una guía para comenzar con los primeros pasos en Node-RED y entender como funciona esta herramienta. A continuación, en el punto 5, cuando se explique la implementación del caso práctico, se explicarán elementos más específicos, como son, el uso de variables y su contexto, el envío y recepción de datos con el servidor Mosquitto, además de la implementación de nodos externos como es el de Telegram y el uso de peticiones HTTP, para publicar datos en una base de datos.

4.3.5 Importar y exportar

Node-RED proporciona un mecanismo para que la exportación e importación de flujos sea muy sencilla. Cuando se tiene un flow y se selecciona la función de exportar "Export" en la ilustración 10, se genera un código en el formato JSON, como se ve en la ilustración 23. Para importar este flujo, se copia el código generado en el anterior flujo y se pega en el nuevo, en la pestaña de "Import nodes".

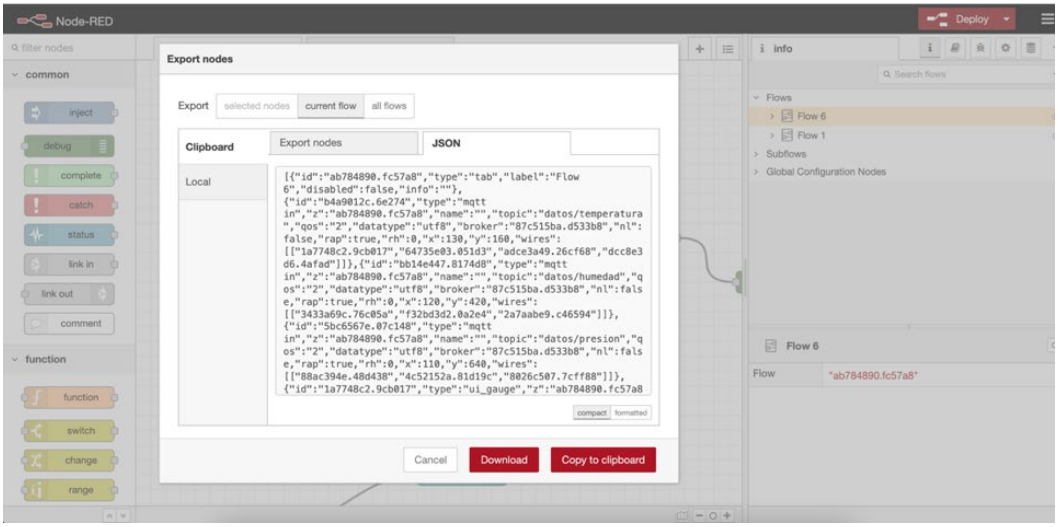

Ilustración 23. Exportación de flujo en Node-RED

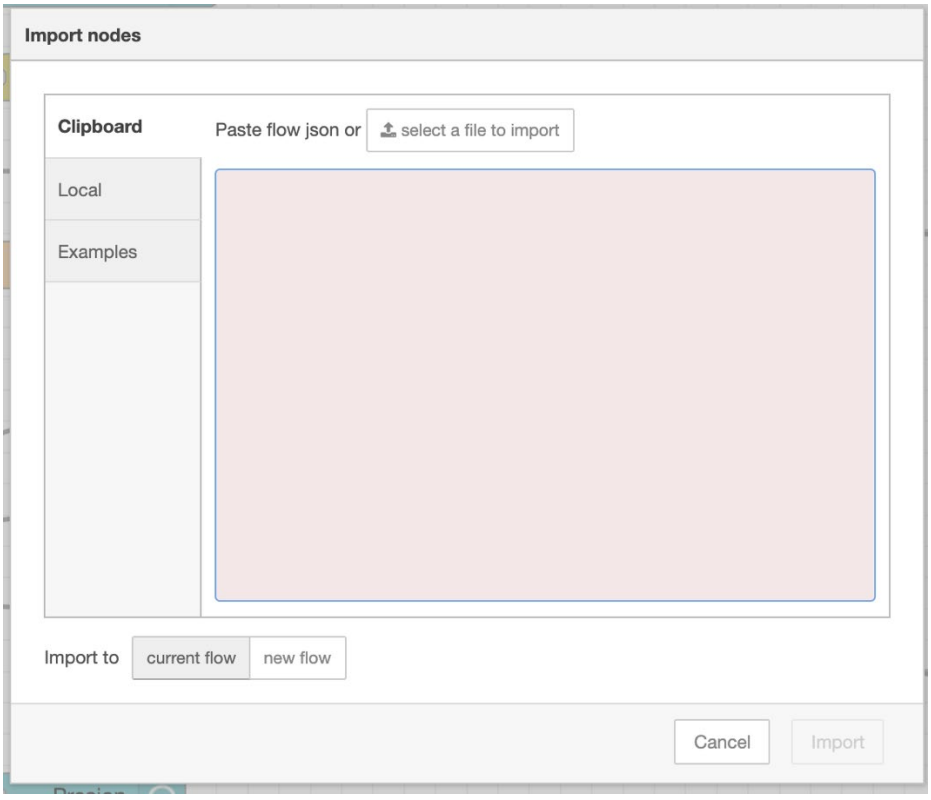

Ilustración 24. Importación de flujo en Node-RED

4.3.6 Herramientas y plataformas similares

En la actualidad existen muchas herramientas, que pueden realizar funciones similares a la de Node-RED, como pueden ser n8n.io, Havoc Shield, iffft o Microsoft

Flow [16], pero la herramienta que más se asemeja, aunque tenga algunas diferencias, es Zapier.

Zapier [17], es una herramienta que nos permite establecer conexiones entre miles de aplicaciones para que nos permita la automatización de flujos de trabajo. En el caso de Node-RED las relaciones se crean entre nodos que realizan funciones concretas, pero en el caso de Zapier las relaciones se crean entre aplicaciones ya definidas.

Concretamente, Zapier posee más de 2000 aplicaciones de todo tipo, entre las que destacan Facebook, Quickbooks o todas las aplicaciones de Google, como Google Sheets y Docs.

Normalmente se utiliza para conectar entre sí herramientas web, y como ocurría en Node-RED sin necesidad de tener grandes conocimientos de programación. La gran diferencia entre esta ultima, es que está más orientada a escenarios IoT, mientras que Zapier, como ellos mismos indican en su página, es una mejor herramienta para la automatización de Marketing [18].

Esta herramienta posee un plan gratuito, en el cual se pueden hacer un máximo de dos conexiones a la vez, denominado el plan “Free Forever”. La mayor desventaja de usar ese plan, es que además de solo poder hacer dos conexiones simultáneas no están todos los bloques disponibles. Para ello se tiene que acceder al plan “Premium” el cuál costaría 20$ al mes. Este plan su que permite integraciones con todos los pasos que se quiera y tener todos los bloques disponibles.

5. Implementación

Una vez explicado el concepto de Node-RED, se realizará a continuación, un caso práctico para poner en uso los conceptos aprendidos. El caso práctico tendrá la siguiente estructura:

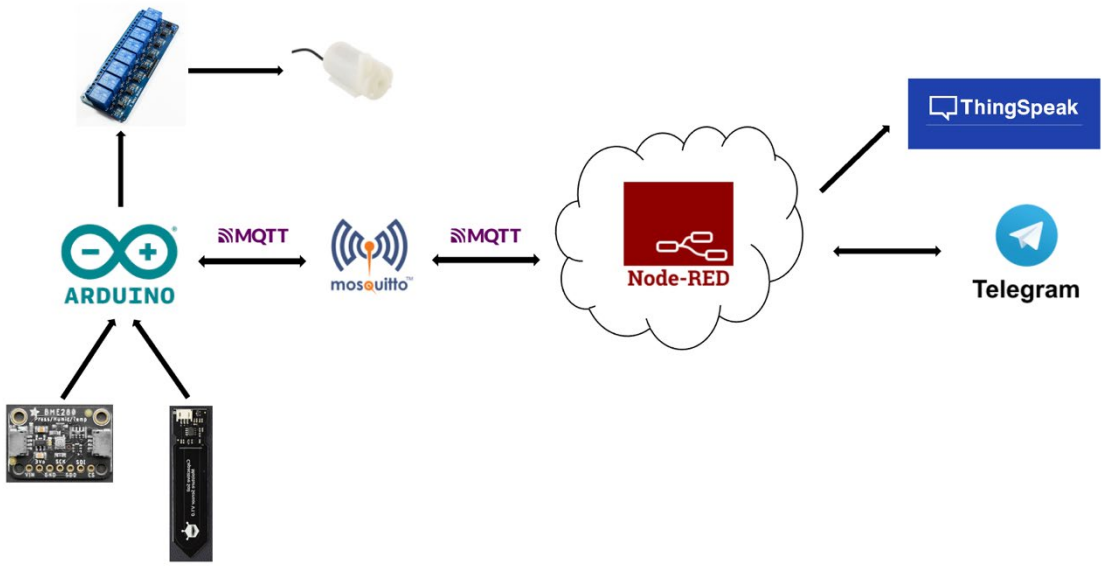

Ilustración 25. Esquema práctico general

5.1 Captación de datos

5.1.1 Sensores

La captación de los datos con los que se van a trabajar se realiza mediante los siguientes sensores:

5.1.1.1 Temperatura, humedad y presión

El sensor BME 280, es un sensor del fabricante Bosch Sensortech[19], que combina, termómetro, barómetro e higrómetro. Su predecesor es el BMP280, pero a esta nueva versión se la ha añadido la capacidad de medir la humedad del aire. En cuanto a su alimentación, será de 5V, aunque también soporta 3,3V.

La medición de la temperatura tiene un rango de -40ºC a +85ºC con una precisión de ±1ºC. Para la presión tiene un rango entre 300 y 1100 hPa con una precisión de ±1P. Con respecto a su funcionamiento como higrómetro tiene un rango de medición de humedad de 0% a 100% con una precisión de ±3%.

La comunicación con la placa será mediante I2C o SPI. En este caso utilizaremos I2C, por recomendación del fabricante y, sobre todo, para ahorrar las conexiones con el pin SDO (Serial Data Out) y CS (Chip Selected) ya que con la comunicación I2C la utilización de estos pines no son necesarias [20].

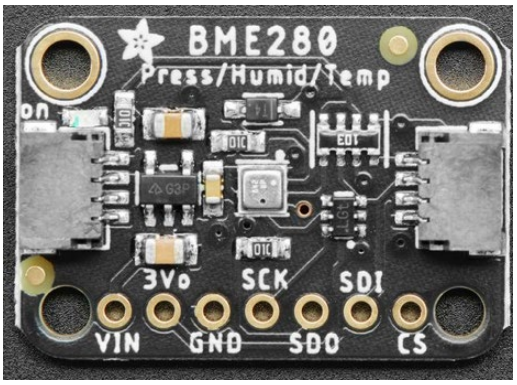

Ilustración 26. Sensor de temperatura, humedad y presión BME280

La función que tiene cada pin que se va a utilizar es la siguiente:

1. **VIN:** Es el pin por el cual se alimenta el sensor.
2. **GND:** Toma a tierra del sensor.
3. **SCK:** Es la señal de reloj del bus. Esta señal determina la velocidad a la que transmite cada Bit.
4. **SDI:** Se utiliza para la transmisión de datos.

5.1.1.2 Humedad de suelo

El sensor SKU:SEN0193, para captar la humedad de la tierra, está hecho con materiales resistentes a la corrosión y su funcionamiento es con un voltaje de 5V, aunque también soporta 3,3V. Captará la humedad en la tierra de forma capacitiva.

En este esquema se indica la profundidad la cuál se debe introducir en la tierra:

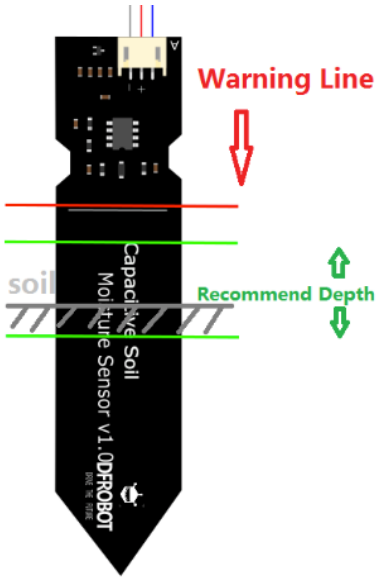

Ilustración 27. Sensor de humedad de suelo SKU:SEN0193

Los valores de respuesta que se obtiene de las medidas de este sensor van de 260 a 520. Considerando que entre 260 y 349 estaría sumergido en agua, entre 350 y 429 estaría la tierra húmeda y entre 430 y 520 la tierra estaría seca.

En cuanto a las conexiones, se trata de un sensor analógico normal, el cual necesitará una toma de corriente, una toma de tierra y otra toma de datos.

5.1.2 Datos en el microcontrolador

Para utilizar este sensor es necesario instalar dos librerías propias del fabricante la Adafruit BME280 Library y la Adafruid Unified Sensor.

Como se ha especificado anteriormente se utilizará la comunicación I2C entre la placa y el sensor. El bus I2C requiere únicamente dos cables, uno para la señal de reloj y otro para el envío de datos de forma síncrona. Cada dispositivo dispondrá de una dirección única, en este el caso de este sensor, la dirección por defecto es la 0x77 [21].

```
datosconexion
#include <Wire.h>                    // incluye libreria de bus I2C
#include <Adafruit_Sensor.h>         // incluye librerias para sensor BMP280
#include <Adafruit_BME280.h>
```

Ilustración 28. Librerias incluidas para la utilización del sensor BME280 en Arduino

Para conectarse con el sensor, en la función setup, se incluirá el siguiente código:

```
void setup() {

  Serial.begin(9600);                          // inicializa comunicacion serie a 9600 bps
  Serial.println("Iniciando:");                // texto de inicio
  if ( !bme.begin(0x77) ) {                    // si falla la comunicacion con el sensor mostrar,
                                               // se pone 0x77 porque es la dirección
    Serial.println("BME280 no encontrado !");  // texto y detener flujo del programa
    while (1);                                 // mediante bucle infinito
  }
```

Ilustración 29. Código para establecer conexión con el sensor BME280 en Arduino

Por otro lado, para el sensor de humedad en la tierra SKU:SEN0193, como se ha comentado anteriormente, es un sensor analógico normal, por lo que las 3 conexiones serán 1 a tierra, otra a 5V y como serán 2 sensores, irán a las entradas analógicas A0 y A1.

Al final, se crearán las variables y se almacenarán de la siguiente forma:

```
TEMPERATURA = bme.readTemperature();         // almacena en variable el valor de temperatura
PRESION = bme.readPressure();                // almacena en variable el valor de presion
HUMEDAD = bme.readHumidity();                // almacena en variable el valor de humedad

humetierra1 = analogRead(0);                 // almacena la humedad de tierra del sensor 1
humetierra2 = analogRead(1);
```

Ilustración 30. Almacenamiento de los datos en variables en Arduino

5.2 Envío de datos al servidor Mosquitto

Para poder hacer aplicaciones utilizando el protocolo MQTT, lo primero que se necesita es un servidor de MQTT, que hay diferentes en el mercado. En nuestro caso vamos a utilizar Eclipse Mosquitto.

Una vez se ha instalado correctamente el broker Mosquitto, para enviar los datos desde Arduino, se va a utilizar la librería que más se utiliza actualmente para MQTT, la PubSubClient. Además de la librería para la conexión WiFi, ya que será el medio de comunicación entre el broker y el microcontrolador.

En este caso MQTT funcionará mediante WiFi, aunque también lo puede hacer por Bluetooth u otras tecnologías.

```
include <WiFiNINA.h>            // incluye las librerias para la conexion WiFi (Propia de Arduino)
include <PubSubClient.h>        // incluye las librerias para la conexion MQTT con el servidor Mosquitto
```

Ilustración 31. Librerías para el envío de datos al servidor Mosquitto en Arduino

5.2.1 Conexión con red WiFi

Lo primero para establecer conexión con el broker, es conectarse a la red mediante WiFi, introduciendo las credenciales del router por el cual se va a acceder y relacionándolo con la librería PubSubClient, la propia de MQTT.

```
//DATOS PARA LA CONEXION WiFi
const char* ssid = "nombre_router";
const char* password = "contraseña_router";

int status = WL_IDLE_STATUS; //Variable de estado que se utilizará para la conexión WiFi

WiFiClient wifi;   // Se crea la variable wifi
PubSubClient client(wifi); // Se relaciona la variable wifi con MQTT


//CONEXION WiFi
  Serial.print("CONECTANDO...");
  while (status != WL_CONNECTED) { //Si no esta conectado
    status = WiFi.begin(ssid, password); //Que inicie la conexion con el usuario y la contraseña introducidos anteriormente
    Serial.print(".");
    delay(1000);
  }
  Serial.println("CONEXION ESTABLECIDA\n");
```

Ilustración 32. Acceso a la red mediante WiFi en Arduino

5.2.2 Conexión con broker Mosquitto

Antes de establecer la conexión con el servidor Mosquitto, en los ajustes de este, se tiene que crear un usuario, con su respectiva contraseña, además de por seguridad, se debe crear también para permitir las conexiones entrantes.

Este usuario y contraseña, se utilizará en la función que viene definida por la librería PubSubClient y posteriormente en Node-RED.

En este TFG, el usuario y contraseña son “usuario”, “usuario”, respectivamente y como forma de prueba.

En primer lugar, se deberán de definir las siguientes variables para la conexión MQTT:

```
//DATOS PARA LA CONEXION CON MOSQUITTO
const char* server = "miIP "; //Se utilizará la IP propia del dispositivo, ya que el servidor Mosquitto se ha creado en este.
const char* mqttUsername = "usuario";  //Usuario Mosquitto
const char* mqttPassword = "usuario";  //Clave Mosquitto
```

Ilustración 33. Variables para la conexión con Mosquitto en Arduino

Para posteriormente establecer la conexión:

```
// CONEXION CON EL SERVIDOR MQTT
  client.setServer(server, 1883); // Establecer conexion con el servidor (server) en el puerto 1883
  client.setCallback(callback); //Establecer el callback para recibir la información


  if (client.connect("arduinosub", mqttUsername, mqttPassword)) { //Conectarse con el usuario y la contraseña indicados
    Serial.println("MQTT CONECTADO");
    client.subscribe("motor/1"); //Subscribirse al topic del que se desea obtener la información
  } else {
    Serial.println("MQTT NO CONECTADO");
    Serial.print("failed, rc=");
    Serial.println(client.state()); //En el caso de que no se conecte mostrara el error
  }
```

Ilustración 34. Conexión con Mosquitto en Arduino

5.2.3 Envío de datos

Para el envío de datos, lo primero que se debe hacer es pasar los datos obtenidos de los sensores a cadena de caracteres, para que puedan ser enviados correctamente y puedan ser tratados con facilidad. Para ello se utiliza la función de Arduino “dtostrf”.

Una vez que se ha pasado el dato a una cadena de caracteres, se publicará indicando el topic en el cual se va a ubicar ese dato.

Así los datos se subirán al broker de la siguiente forma:

```
//ENVIO DE LA INFORMACION A EL SERVIDOR MQTT

long now = millis();
  if (now - lastMsg > 2000) { //la informacion se enviara cada 2s (Como el tiempo de refresco de los sensores).

    lastMsg = now;
    char tempString[8];
    dtostrf(TEMPERATURA, 2, 2, tempString); // Pasar la temperatura a una cadena (numero, tamaño, decimales, buffer donde se alamcena)
    client.publish("datos/temperatura", tempString);

    char humeString[8];
    dtostrf(HUMEDAD, 2, 2, humeString); // Pasar la humedad a una cadena (numero, tamaño, decimales, buffer donde se alamcena)
    client.publish("datos/humedad", humeString);
```

Ilustración 35. Envío de datos a Mosquitto en Arduino

5.3 Recepción de datos en Node-RED

Como se ha explicado anteriormente, Node-RED no accederá a los datos directamente desde el microcontrolador, si no que accederá a través del servidor Mosquitto, en el se encontrarán todos ellos.

El nodo en concreto, se encuentra en el paquete "network", como se muestra en la ilustración 36:

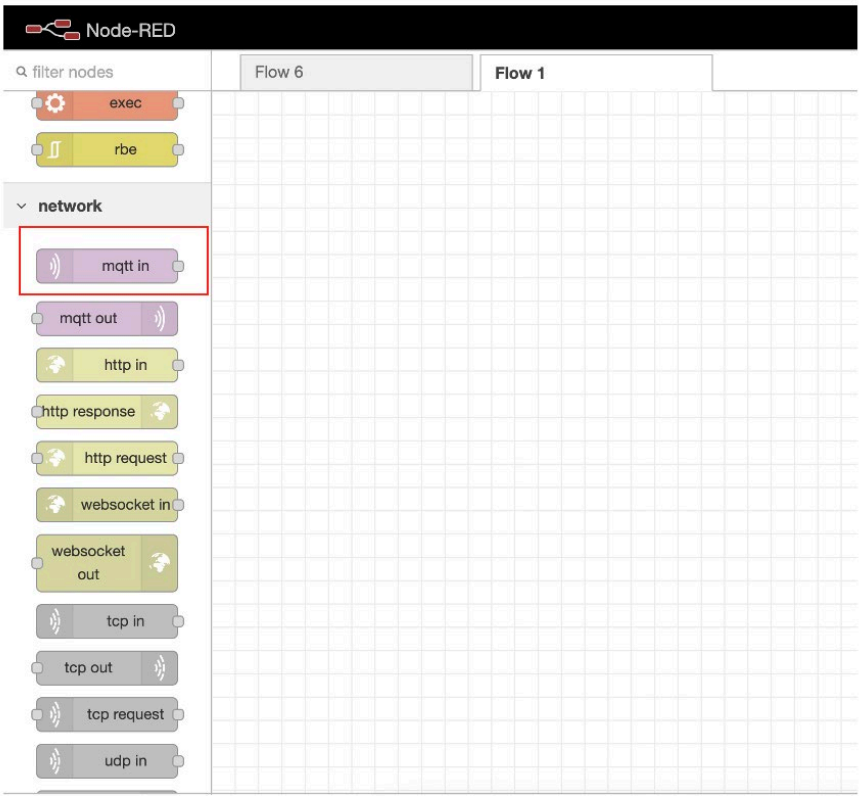

Ilustración 36. Localización nodo MQTT en Node-RED

Siguiendo su propia guía incluida en Node-RED, este nodo se encargará de conectarse a un broker MQTT y subscribirse a los mensajes de un topic en concreto. Se trata de un nodo iniciador pasivo, ya que se mantiene a la escucha de que cambie algún dato y cuando este cambie, es cuando lo recibe actualizado. Además, al ser iniciador, solo tendrá una salida, aunque en esta salida se incluyan varios datos:

- **payload:** Es la cadena de texto que contiene el dato en concreto. La variable que almacena este dato tendrá el nombre "msg.payload". Se trata de una variable de tipo "msg", porque es de contexto, es decir solo se almacenará para los nodos que estén conectada en el mismo flujo. Los nodos que no se encuentren en distinto flujo, no podrán a acceder a los objetos del tipo "msg".
- **topic:** Es la cadena de texto que contiene el topic al que está subscrito. La variable en la que se almacenará este dato tendrá el nombre "msg.topic". Para separar la jerarquía de niveles en los topics se usará "/".
- **qos:** Contendrá el nivel de la calidad de servicio en este caso, es 2, la máxima. La variable en la que se almacenará será "msg.qos".
- **retain:** Devolverá true o false, en el caso de que el mensaje que se ha recibido haya sido retenido, o haya sido generado hace mucho tiempo.

5.3.1 Configuración nodo MQTT

Para utilizar el nodo comentado en la ilustración 36, se debe de arrastrar al área central de trabajo, hacer doble click y se podrá editar, desplegando esta ventana:

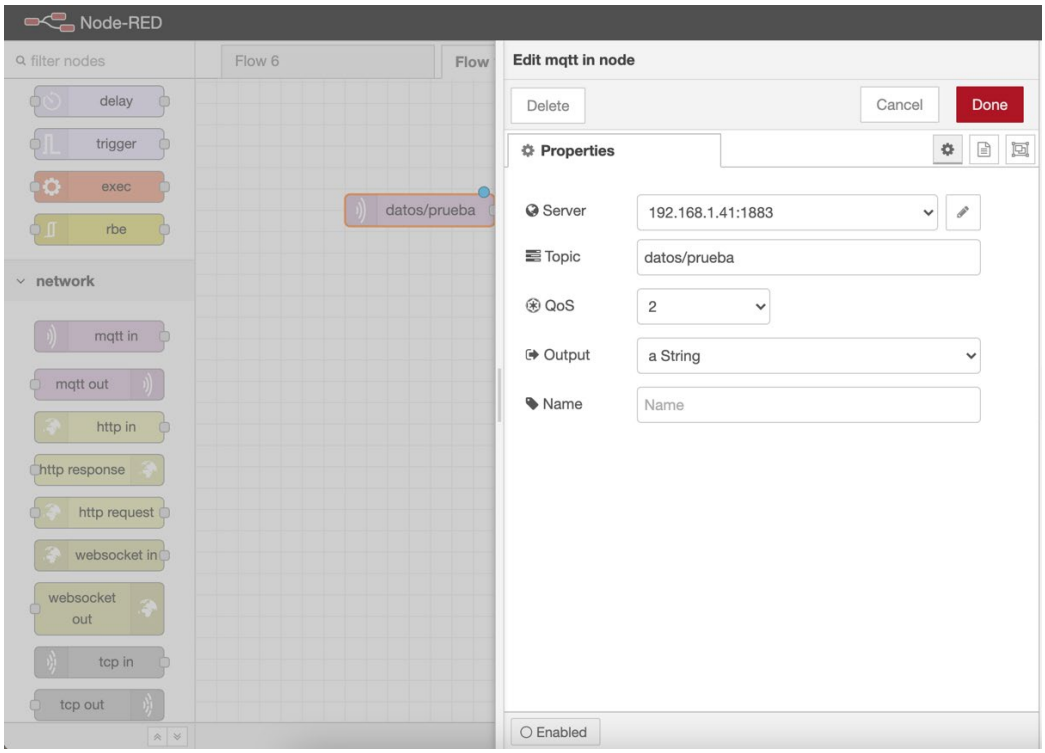

Ilustración 37. Menú del nodo "mqtt in" en Node-RED

En primer lugar, se encuentra el apartado de “Server”, el cuál se editará como se muestra en la ilustración 38:

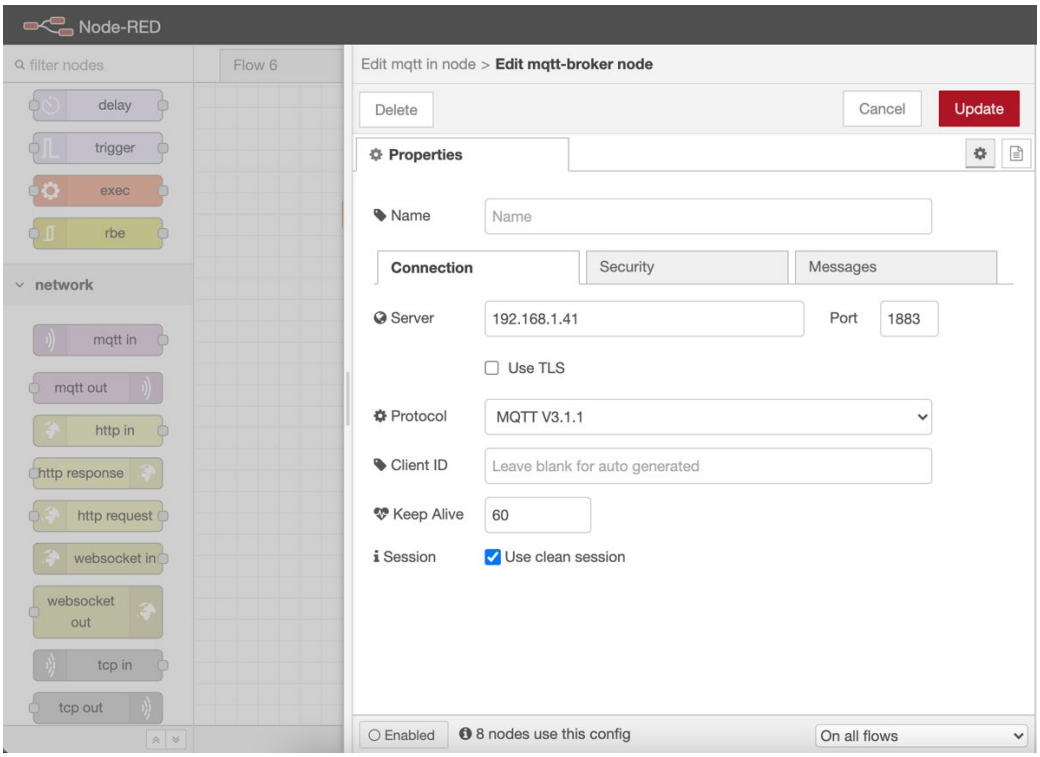

Ilustración 38. Configuración de la conexión del nodo "mqtt in" en Node-RED

En esta pestaña, se configurarán los datos necesarios para acceder al broker Mosquitto. En este caso, el servidor está instalado y corriendo en el mismo ordenador que se está utilizando, por lo tanto, la dirección del broker será la misma que la del propio ordenador. En el caso de este TFG se ha usado el sistema operativo MacOS, por lo tanto, para saber la dirección, se escribirá en el terminal el comando "ifconfig" y se pondrá esa dirección.

Por otro lado, el puerto es el 1883, el puerto por defecto en las comunicaciones MQTT.

Para concluir la versión del protocolo, en este caso es la 3.1.1.

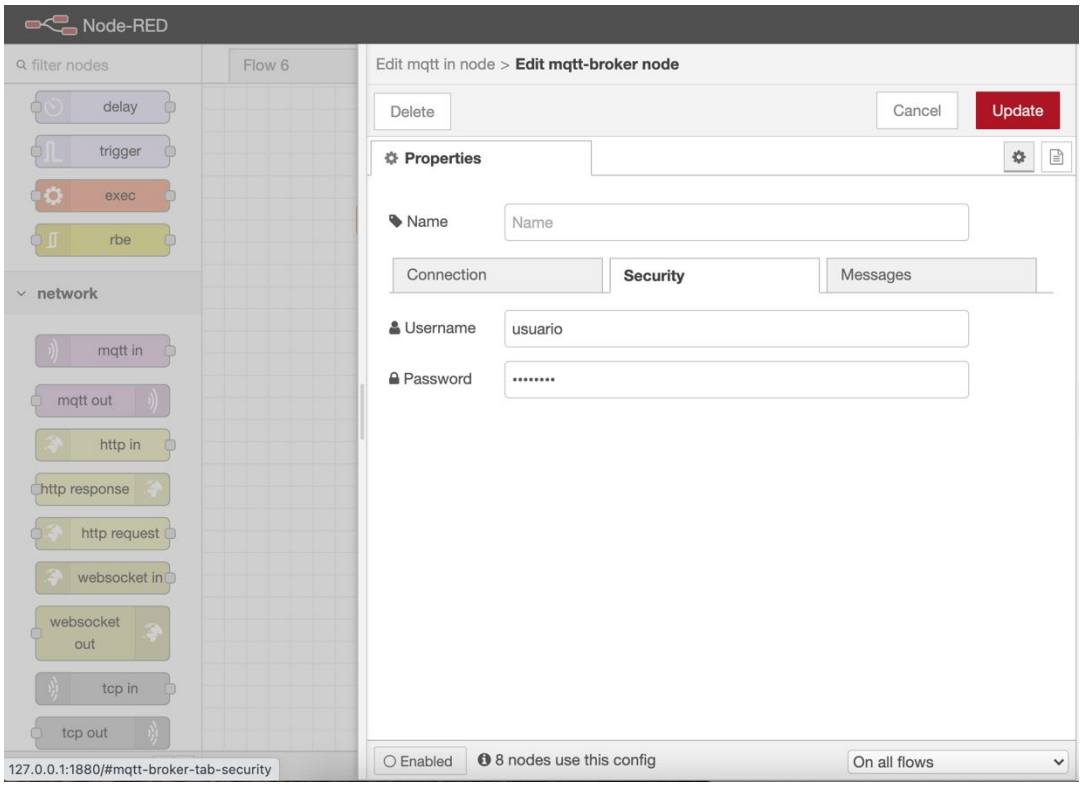

Ilustración 39. Seguridad en el nodo "mqtt in" de Node-RED

Continuando con la configuración del servidor, nos encontramos con el apartado de seguridad, en el cual se incluirán las credenciales del usuario que se haya creado previamente en el servidor, en este caso "usuario", "usuario".

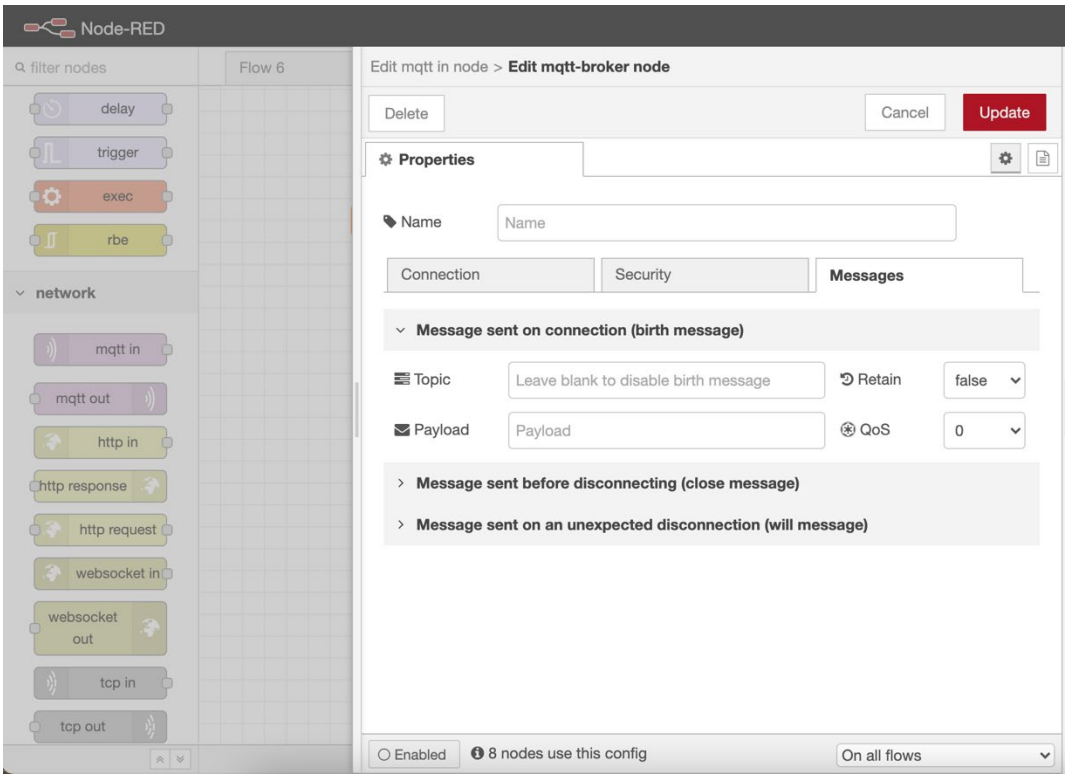

Ilustración 40. Tipos de mensajes en el nodo "mqtt in" de Node-RED

Para concluir con la configuración del servidor, se encuentra la pestaña “Messages”, la cual nos proporciona la funcionalidad, de opcionalmente, enviar un mensaje cuando Node-RED acceda al servidor. Indicando el topic en el cual se va a publicar el mensaje y el mensaje como tal. Además de la publicación de este mensaje cuando se conecte, existe también la posibilidad de enviar un mensaje cuando se desconecte o de cuando ocurra algún otro error.

5.3.2 Visualización de la información

En el apartado anterior, se ha explicado que valores devuelve el nodo de MQTT, pero para acceder a estos valores necesitamos otro nodo distinto.

Este nodo se encuentra dentro del paquete incluido por defecto por Node-RED, denominado “common”.

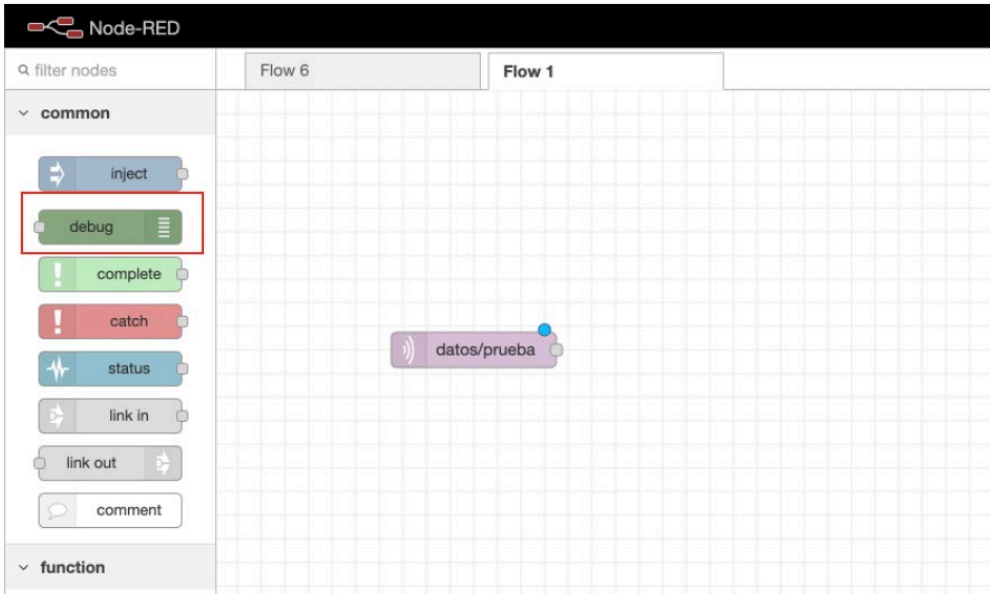

Ilustración 41. Nodo para la representación de la información en Node-RED

El bloque que se está señalando en la ilustración 41, se arrastra hasta el área de trabajo y se une con el nodo de recepción MQTT. Así la información recibida por este nodo, podrá ser compartida con el nodo “debug”.

El nodo “debug” nos proporciona la capacidad de mostrar en la pestaña “debug” la información que se le especifique, como se ve en la ilustración 42.

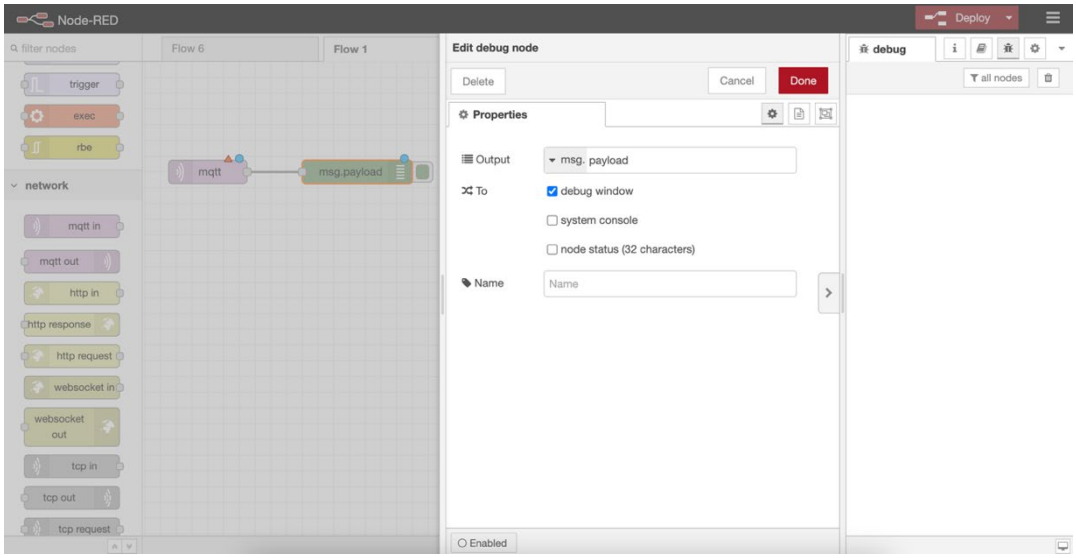

Ilustración 42. Configuración del nodo "debug" en Node-RED

Como se ve en el menú desplegado, a la hora de hacer doble click en el nodo debug, se podrá seleccionar el dato que se quiera mostrar en la ventana de depuración. En el ejemplo mostrado en la ilustración 42, se mostrará el dato en concreto. Aunque, como se ha explicado en el apartado 4.3, también se podría acceder al topic cambiando "msg.payload" por "msg.topic".

Además, se podrá decidir si aparte de mostrarse por la ventana de depuración, también se pueda hacer por consola o, por otro lado, establecer ese dato como estado del nodo.

5.4 Datos y variables en Node-RED

Cuando se genera un dato en Node-RED, solo podrán acceder a él, los nodos que se encuentren conectados, es decir, que pertenezcan al mismo flujo de datos. Por defecto, la variable se generará en el contexto del nodo, aunque existen tres tipos de contextos.

5.4.1 Contexto de las variables

Node-RED, proporciona una forma de almacenar la información, para que esta sea compartida, sin necesidad de que los nodos se encuentren en el mismo flujo de datos. Esto es el "contexto" de las variables [22].

El alcance que tenga un valor va a determinar con quién se comparte y por lo tanto su contexto. Hay tres niveles de alcance según su contexto:

- **Nodo:** Solo será visible para el nodo que estableció el valor y los nodos que se encuentren en su mismo flujo de datos.
- **Flujo (Flow):** Será visible para todos los nodos que se encuentren en la misma pestaña del flujo.
- **Global:** Será visible para todos los nodos que se encuentren en la herramienta Node-RED, independientemente si se encuentran conectados o en el mismo flujo de datos.

5.4.2 Cambio de nombre y contexto

Todos estos cambios de contexto comentados en el apartado anterior, el cambio del nombre de las variables o su eliminación, es gracias a un nodo específico denominado "change" dentro del paquete "function" incluido por defecto en Node-RED.

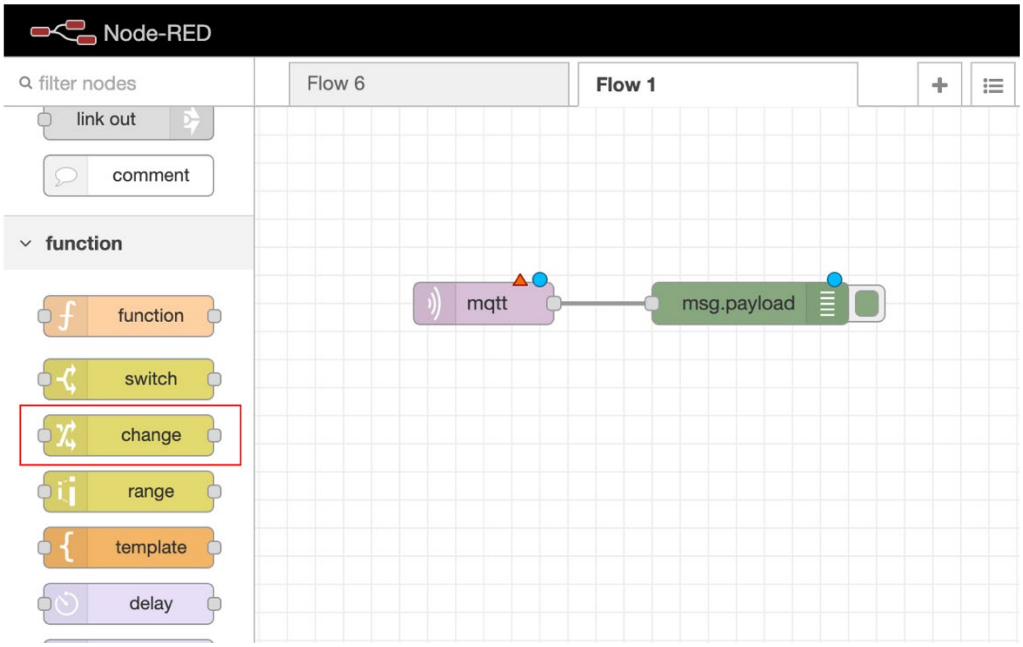

Ilustración 43. Localización del nodo "change" en Node-RED

Este nodo nos proporciona las funcionalidades de establecer, cambiar, eliminar o mover las propiedades de un mensaje y su contexto. Se podrán establecer varias reglas y serán aplicadas en el orden en el que se han definido. [23]

Concretamente, las operaciones disponibles son:

- **Set:** Establece una variable, normalmente a partir del valor al que está conectado, es decir, el "msg.payload" aunque también se podrá establecer propiedades a otros valores existentes.
- **Change:** Se encarga de buscar y reemplazar el valor de una variable, por ejemplo, si la variable coche es igual a azul, podrá cambiar este valor a rojo.
- **Delete:** Eliminará el valor de una variable.
- **Move:** Moverá o renombrará el valor de una variable.

En este caso, lo que se busca es cambiar el nombre y el contexto del dato recibido, por lo tanto, se utilizará la operación set, para cambiar el nombre de la variable a “dato” y cambiar su contexto a “flow”, para que este dato sea recibido y pueda ser utilizado por todos los nodos del programa.

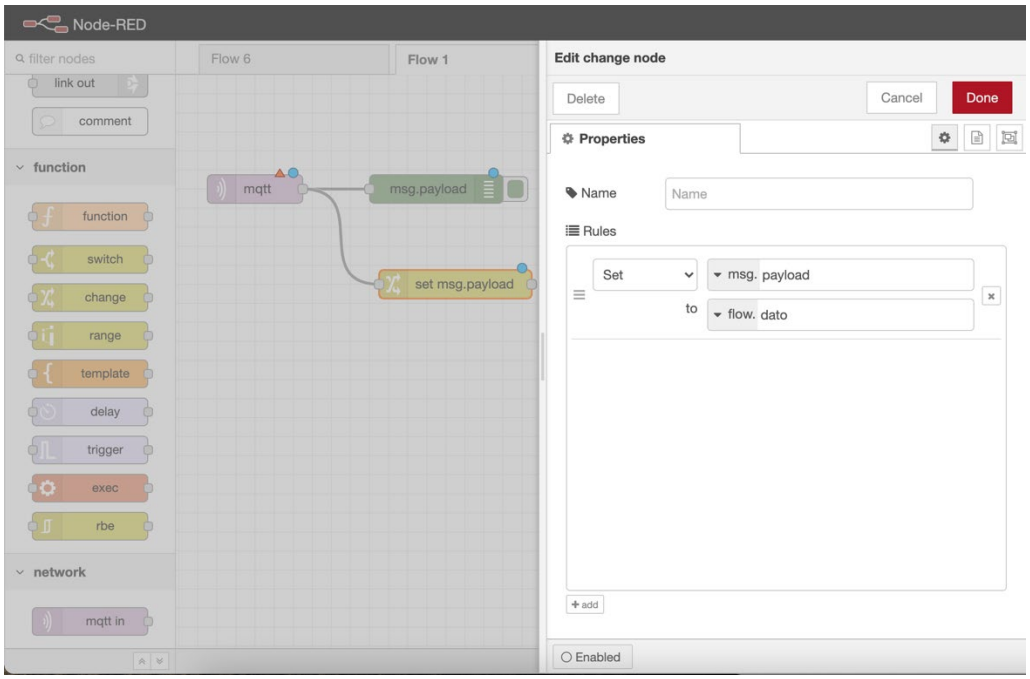

Ilustración 44. Cambio del contexto de las variables en Node-RED

Esta operación se realizará con todos los datos de los sensores, es decir, el de temperatura, el de humedad en el aire, el de presión y los dos que está conectados a la humedad de la tierra.

Configurando el nodo de MQTT correctamente, es decir, siguiendo los topics que se han generado desde el microcontrolador, introduciendo correctamente la dirección del Borker y los datos de autenticación. Además de establecer todas las variables en el contexto del flujo con su respectivo nombre.

Una vez realizado lo anterior quedará el siguiente esquema:

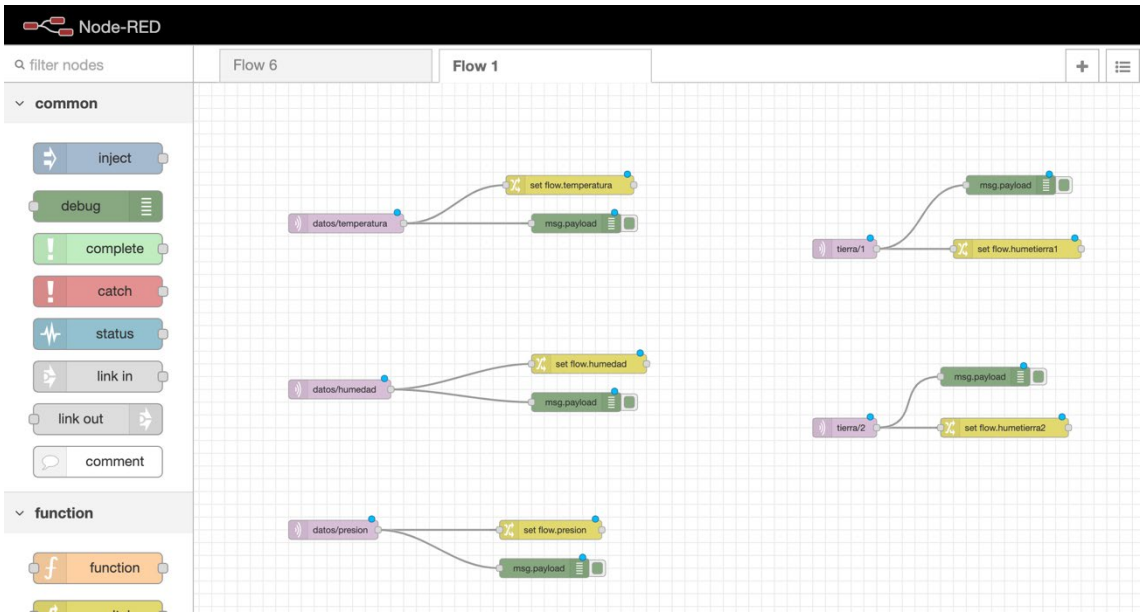

Ilustración 45. Esquema general de la recepción de datos y el cambio de contexto en Node-RED

5.4.3 Dashboard

Un dashboard es un tablero o cuadro de mandos que nos proporciona una forma gráfica de visualizar los datos en directo.[24]

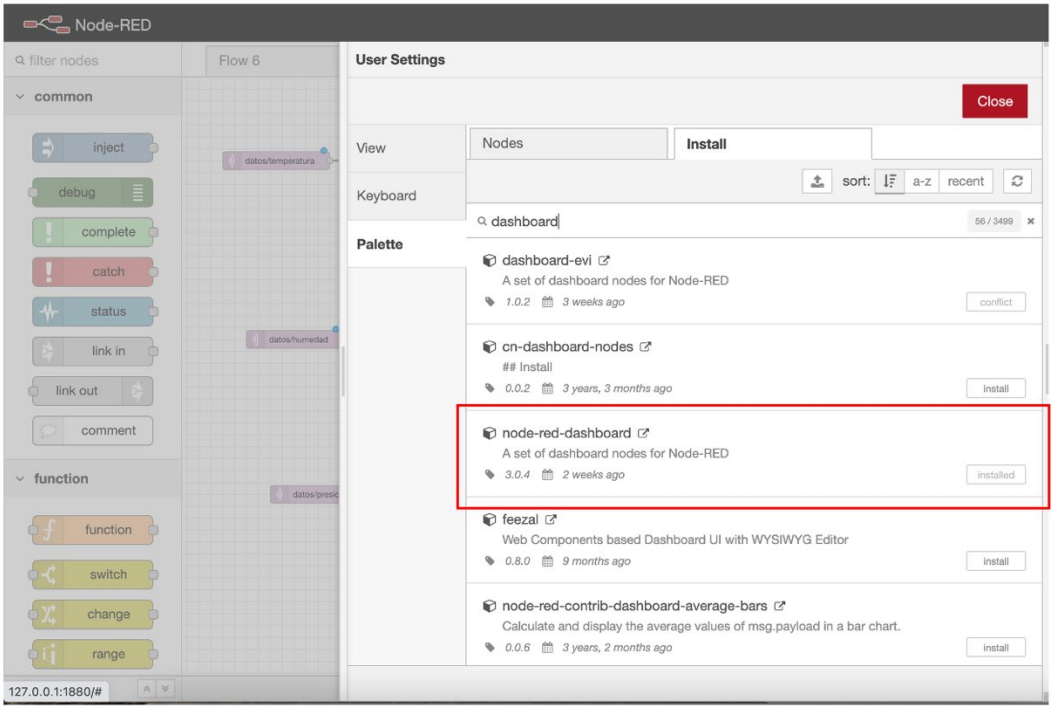

Ilustración 46. Instalación nodo "dashboard" en Node-RED

En este caso, Node-RED no proporciona una funcionalidad nativa para esto, si no que hay que hacer uso de la herramienta para instalar librerías externas. Para ello, hay que seguir los pasos explicados en el punto 2.4.2. Se introducirá en la barra de búsqueda "dashboard" y se seleccionará el indicado en la ilustración 47.

Se generará así un paquete nuevo de nodos. En concreto, como modo de prueba se utilizará el indicado en la ilustración 48. Se añadirá a las variables temperatura, humedad y presión para ver un registro en directo de forma gráfica.

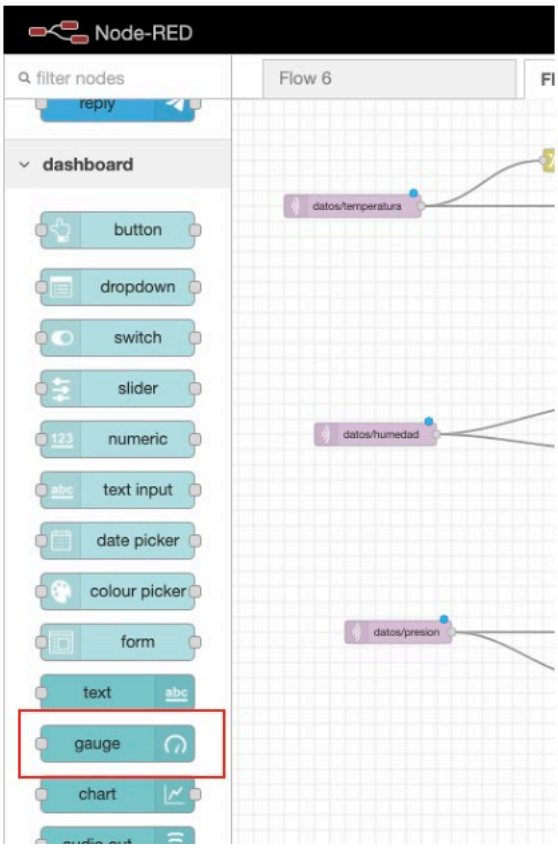

Ilustración 47. Localización nodo "dashboard" en Node-RED

Una vez se ha conectado este nodo al nodo que genera los datos, se modificarán las características del gráfico que generará. Por ejemplo, en este caso se utilizará un gráfico de aguja y en el caso de la temperatura, como se muestra en la ilustración 48, se establecerá un mínimo de 0 y un máximo de 50, utilizando los colores desde el verde al rojo pasando por el amarillo.

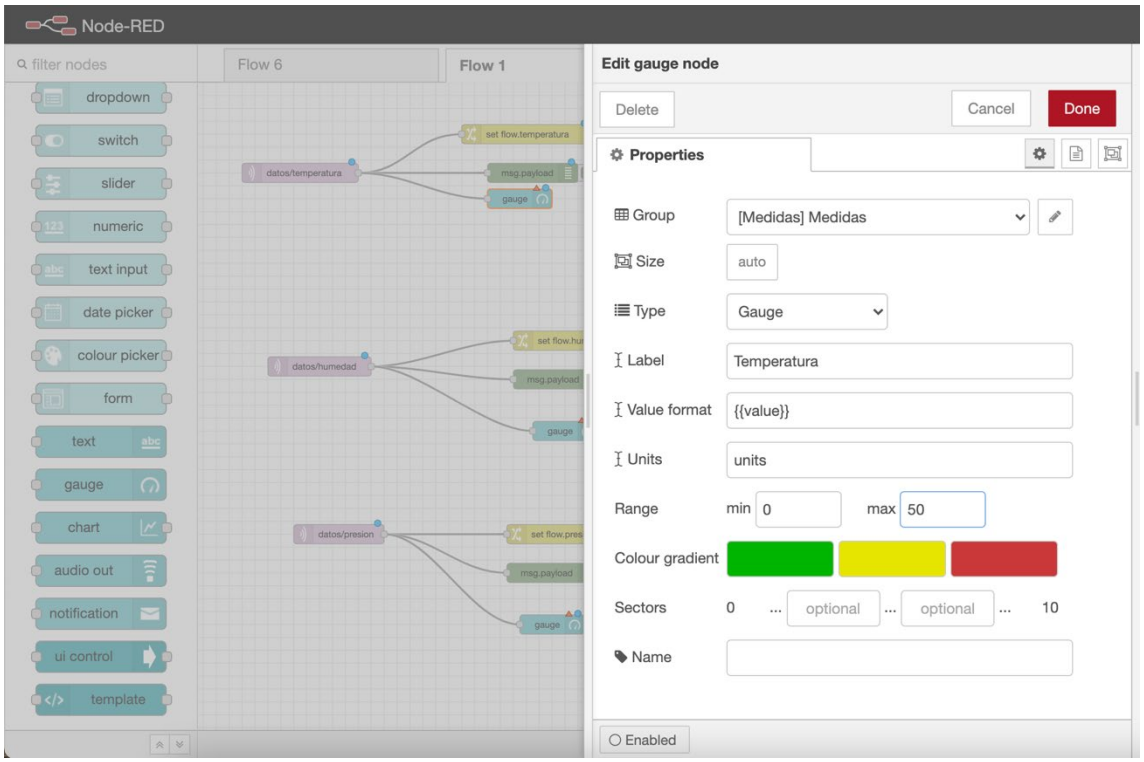

Ilustración 48. Configuración nodo "dashboard" en Node-RED

Una vez se han configurado las gráficas de las variables correctamente, se podrán abrir una nueva pestaña en el navegador con toda la información configurada.

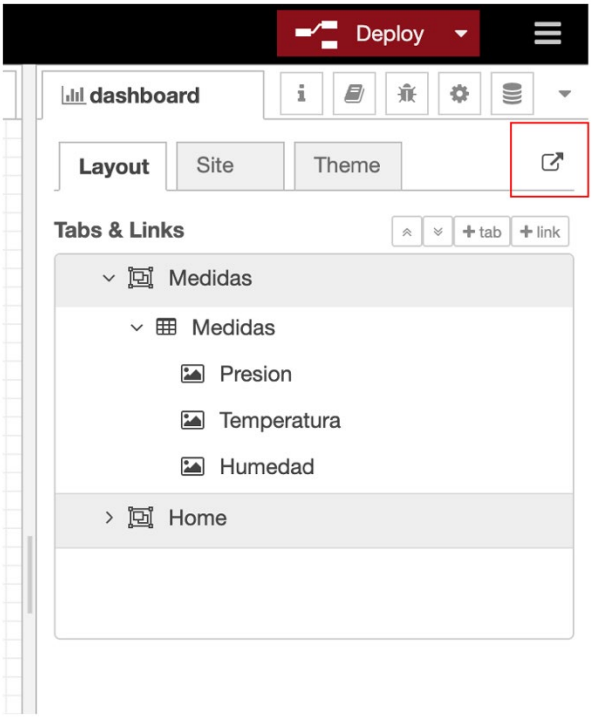

Ilustración 49. Localización de la ventana dashboard en Node-RED

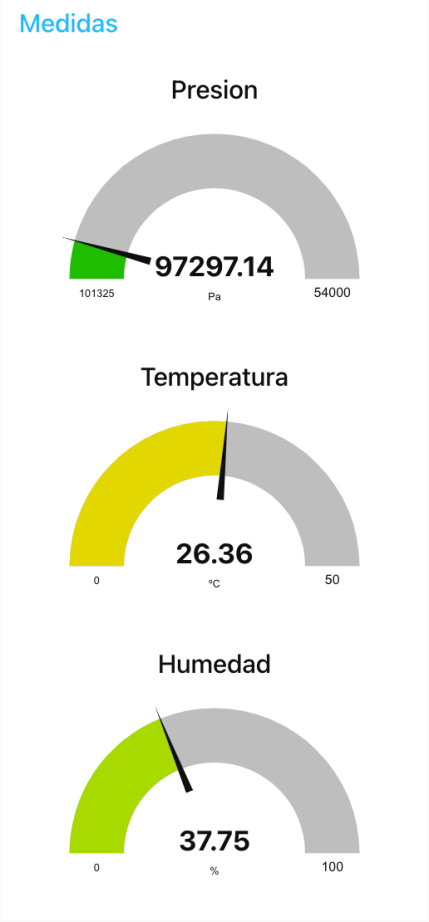

Ilustración 50. Visualización de la información en dashboard

5.5 Interacción con el usuario

Una vez la información se encuentra en la herramienta Node-RED, el usuario accederá a esta información e interactuará con el sistema mediante la aplicación Telegram [25]. Telegram nos permite la creación de un bot propio, en el cual se introducen una serie de comandos para crear una comunicación entre el usuario y nuestro sistema en concreto.

Se ha decidido usar en este TFG esta forma de interacción entre el usuario, Node-RED y el sistema porque por la parte del usuario, es muy sencillo acceder a la aplicación Telegram, ya que es gratuita en todo tipo de smartphones, tiene acceso mediante navegador web y una aplicación de escritorio. Por otro lado, por parte de Node-RED, aunque no lo incluye de forma nativa, se puede instalar de forma externa los nodos necesarios para completar la comunicación.

5.5.1 Creación del bot

Para la creación de nuestro propio bot en Telegram se hará uso del bot ya creado llamado "@BotFather".

Una vez se ha encontrado con la herramienta de búsqueda de Telegram, se introducirá el comando "/start" y el bot se activará y mostrará todos los comandos que reconoce, con todas las funcionalidades que tiene.

Ilustración 51. Creación de un bot en Telegram

Como se puede ver, para crear un nuevo bot, se debe introducir el comando "/newbot", y a continuación, nos pedirá que introduzcamos el nombre del bot, en este caso el nombre que se le ha puesto es "@node_red_epsl_bot" y nos generará un mensaje con el enlace del nuevo canal que se ha creado, que hará la función de nuestro bot y además un token, para, entre otras cosas, introducirlo cuando se configure la conexión con Node-RED.

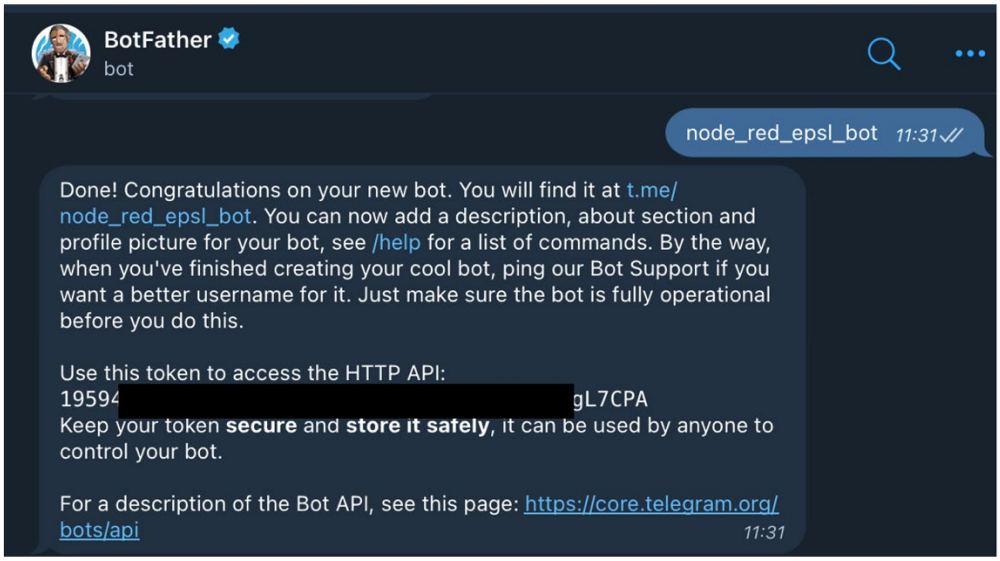

Ilustración 52. Establecimiento del nombre del bot en Telegram

Este token es de uso privado exclusivamente para el creador del bot, ya que, de otro modo, cualquiera podría controlar las características y funcionalidades del bot.

5.5.2 Comunicación con Node-RED

Para la comunicación de Telegram y Node-RED, antes de todo hay que instalar el paquete de nodos necesarios, ya que no vienen incluidos por defecto.

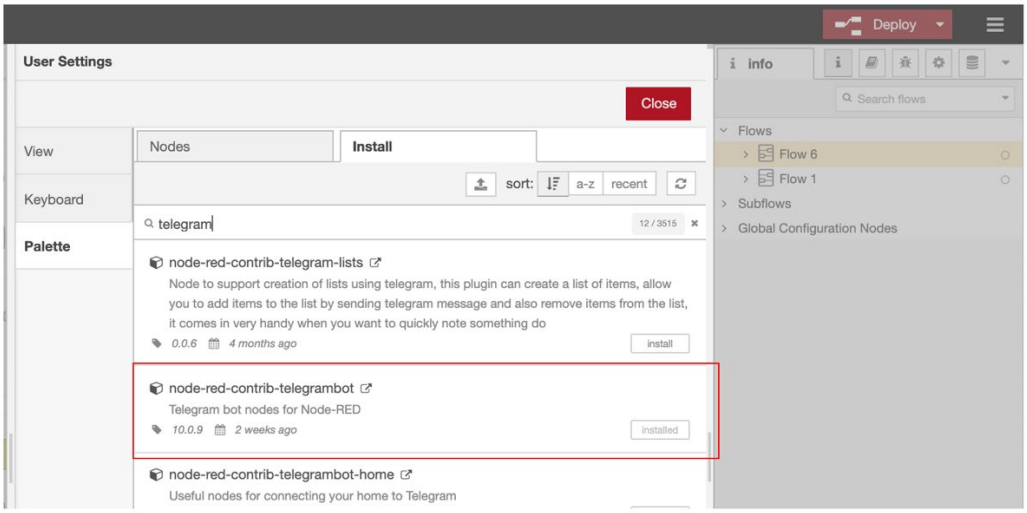

Ilustración 53. Instalación de los nodos de Telegram en Node-RED

Generando así los siguientes nuevos nodos:

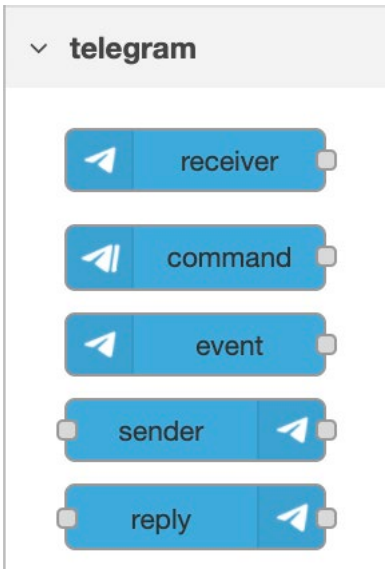

Ilustración 54. Nodos de Telegram en Node-RED

Para recibir un mensaje en Node-RED procedente del bot que hemos creado se hará uso del nodo “receiver”. Este nodo se tendrá que configurar con los datos que nos ha generado el “BotFather”, mostrados en la ilustración 52 y los datos que nos pida este nodo.

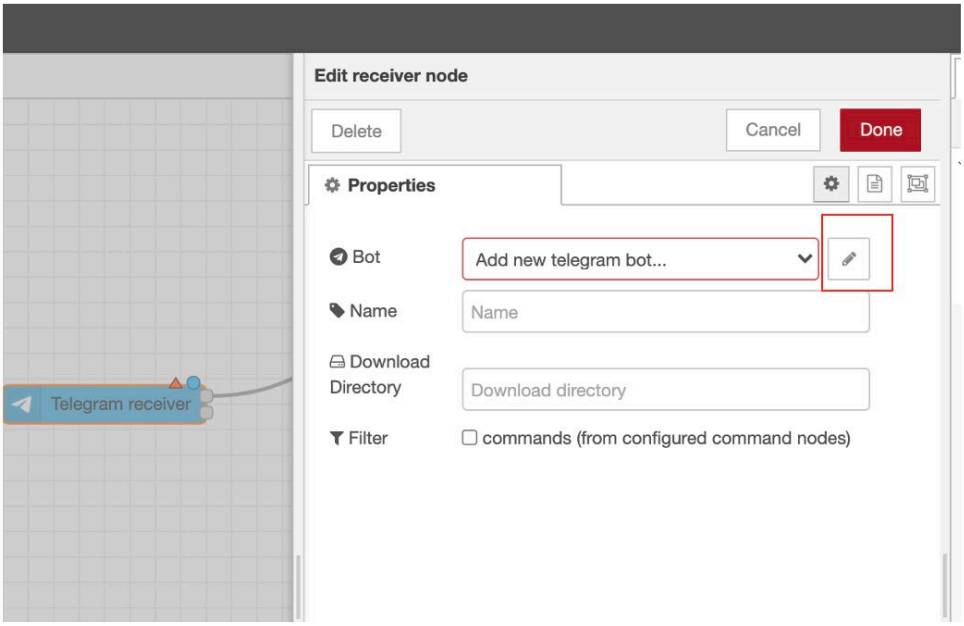

Ilustración 55. Menú del nodo "Telegram receiver" en Node-RED

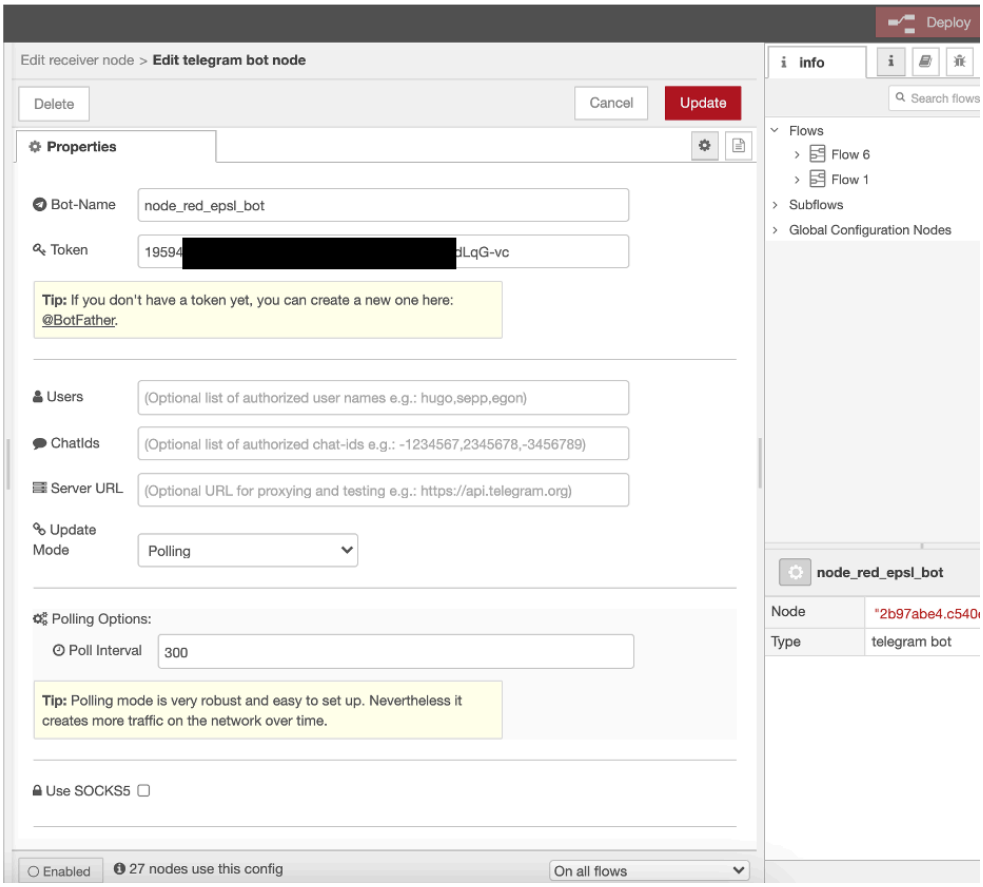

Ilustración 56. Configuración de las propiedades del nodo "Telegram receiver" en Node-RED

Como se muestra en la ilustración 56, se tendrá que introducir el nombre del bot que hemos creado y el token que se nos ha generado para este bot. Ahora, cuando se escriba un mensaje en el bot, este será recibido por Node-RED.

Cuando se recibe un mensaje, el nodo genera 4 salidas:

- **Content:** La cual es el contenido del mensaje.
- **Type:** El tipo de dato del que se trata el mensaje recibido.
- **MessageID:** El ID del mensaje.
- **ChatId:** Es el ID del chat desde el cual se ha escrito. Es uno de los datos más importantes, ya que nos indica quien ha enviado el mensaje y para saber posteriormente a quién debe responder Node-RED. Este ID es único para cada bot que se haya creado, es decir, en otro bot distinto, el ID también es distinto

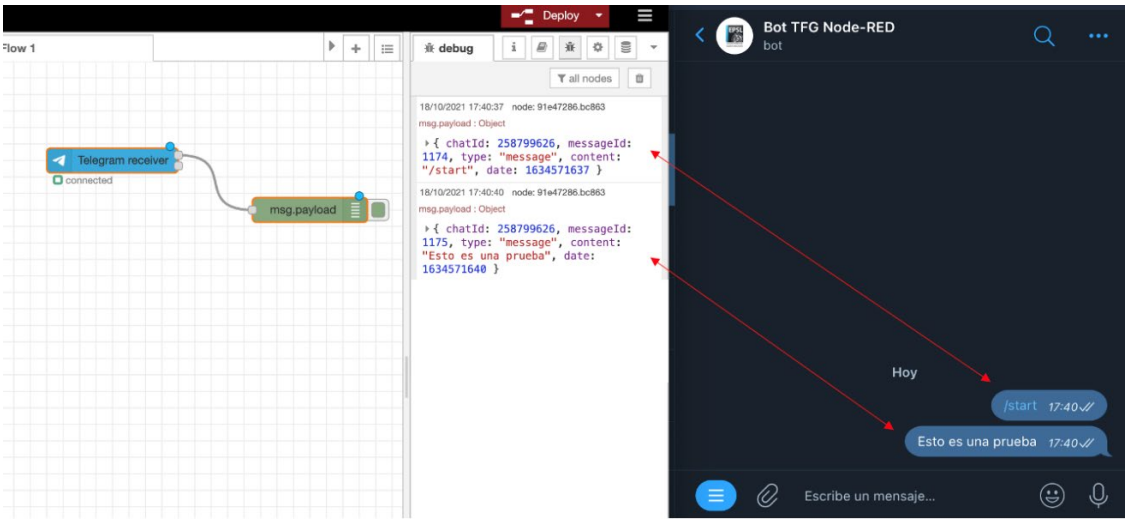

Ilustración 57. Envío de datos de Telegram a Node-RED

Por otro lado, para el envío de información de Node-RED al bot, se utilizará el nodo llamado "sender". A este nodo tendremos que introducirle la información con los mismos atributos que se reciben, es decir, content, type y chatID.

Para ello se utilizará el nodo "fuction" incluido por defecto en Node-RED. Este nodo nos proporciona la capacidad de escribir cualquier tipo de código en JavaScript. Realmente todos los nodos que están presentes en Node-RED, se pueden crear a partir de este nodo "fuction".

La función en concreto tendrá la siguiente forma:

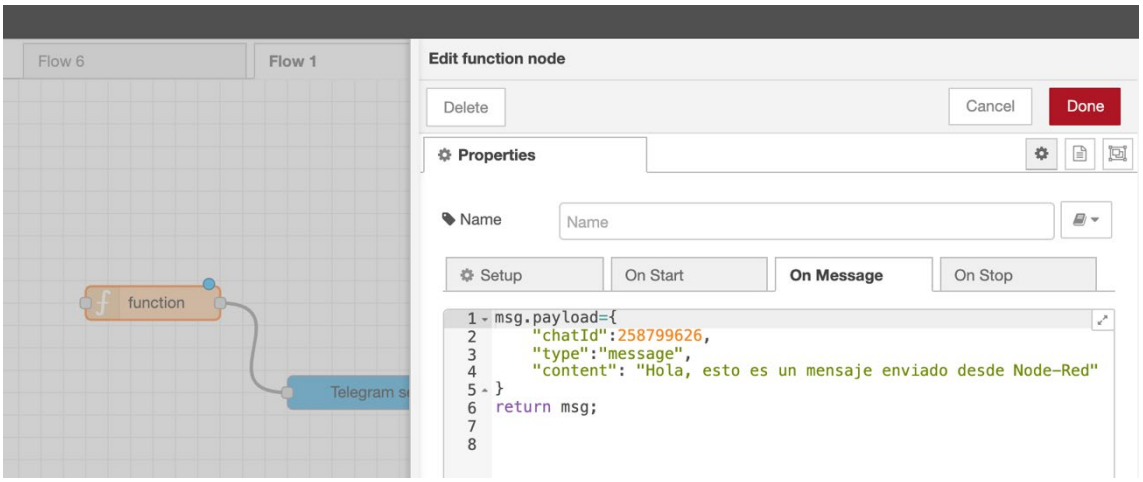

Ilustración 58. Función para el envío de datos de Node-RED a Telegram

Se creará un objeto payload con los distintos atributos dentro, y se enviarán al nodo "Telegram Sender", que ha de ser configurado de la misma forma que el "Telegram Receiver". Esto es una tarea sencilla, ya que el bot solo hay que incluirlo una sola vez en los nodos y posteriormente solo hay que seleccionarlo.

El "chatID", en este caso será el mismo que el que se ha obtenido en el nodo "Receiver", ya que se quiere enviar la información al mismo usuario.

A continuación, en la ilustración 59 se ve como ha llegado correctamente el mensaje que se ha generado a partir de la función de la ilustración 58.

Ilustración 59. Recepción de la información en Telegram procedente de Node-RED

5.5.3 Comandos

El verdadero potencial del bot, comienza cuando se utilizan los comandos. El nodo "command" nos permite la funcionalidad, de cuando se escriba en el bot un comando, se ejecuten una serie de acciones indicadas en Node-RED.

En nuestro caso, los comandos servirán tanto para obtener información como también para actuar. Simplemente con poner "/temperatura" obtendremos la temperatura captada por el sensor. Para ello, en primer lugar, se deben configurar los nodos

"command", a fin de que cuando se introduzca un comando en Telegram, este pertenezca a la lista de comandos reconocidos por Node-RED que se hayan introducido con anterioridad.

Cuando se introduzca un comando que esté configurado en Node-RED, se activarán los nodos conectados en el mismo flujo. En la ilustración 60, en el caso de que se introduzca en Telegram el comando "/temperatura". Se activará el nodo "fuction" y posteriormente el "Telegram Sender". Con el fin de que cuando se introduzca dicho comando, se envíe al usuario el valor de la temperatura.

Todos los valores obtenidos del servidor Mosquitto mediante el nodo MQTT, se han cambiado al contexto del "flow" completo, como se ha mostrado en el apartado 5.4.2. Por lo tanto, cualquier nodo del flujo podrá acceder a estos datos, como es en este caso. Mediante el nodo "Telegram Sender" y la información incluida en el nodo "fuction" se enviará el valor de la temperatura que se ha captado del nodo MQTT.

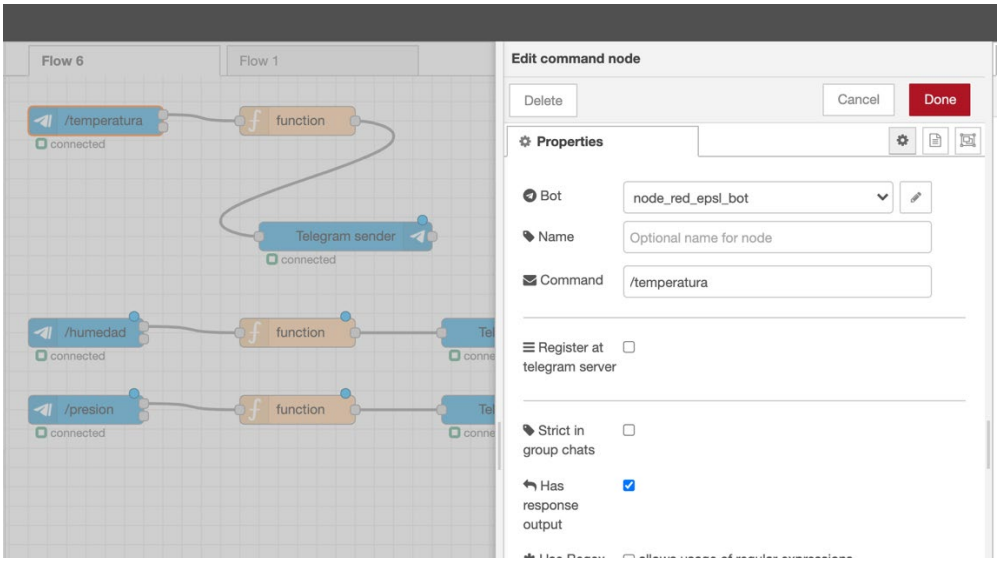

Ilustración 60. Configuración de los comandos de Telegram en Node-RED

Al valor de la temperatura se accederá mediante el código *flow.get("temperatura")*. Obteniendo así el valor de la variable. Quedando el nodo "fuction" de la siguiente forma:

```
msg.payload={
    "chatId":flow.get("idtelegram"),
    "type":"message",
    "content": "Actualmente hace una temperatura de " + flow.get("temperatura") +" "+"ºC"
}
return msg;

```

Ilustración 61. Función de respuesta a un comando de Telegram en Node-RED

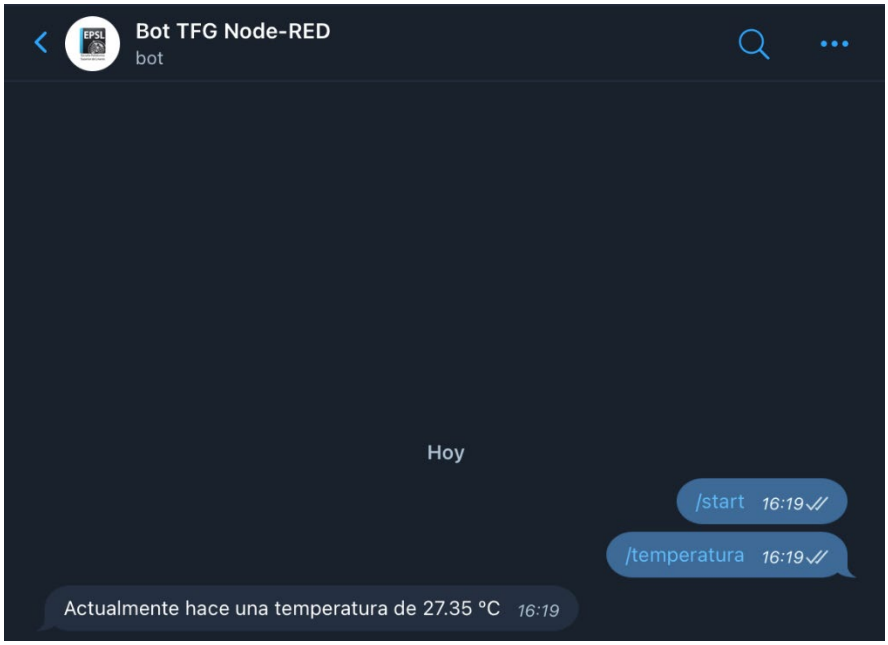

Ilustración 62. Respuesta a un comando en Telegram

El "chatID" al cuál se le enviará la información, será al que haya escrito el comando, en forma de respuesta. Esto es posible, debido a que cuando se escribe un comando, a parte de reconocerlo y actuar, también se guarda el contenido y el "chatID" del que proviene el último mensaje. Para ello, se usarán los siguientes nodos:

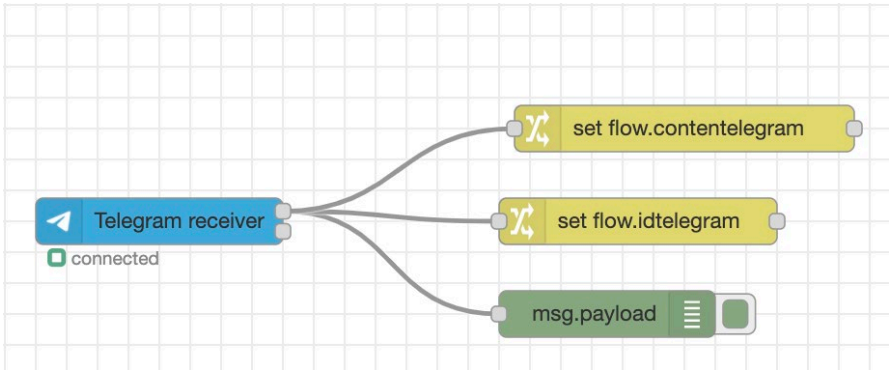

Ilustración 63. Esquema de la Información almacenada de un mensaje de Telegram en Node-RED

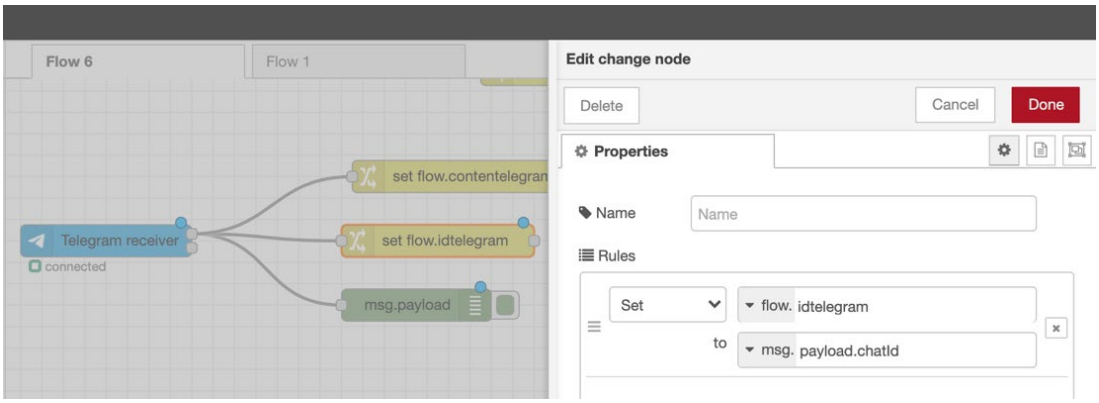

Ilustración 64. Cambio de nombre y contexto de las variables de Telegram en Node-RED

De esta forma, se almacena el contenido y el Id del chat del que proviene el último comando recibido por Node-RED, almacenándolo con un nombre y cambiándole el contexto para que puedan acceder todos los nodos, como es el caso del nodo "fuction" mostrado en la ilustración 61, que accede a la variable temperatura.

Esta operación se realizará con los demás comandos que se quieran añadir, para que los reconozca Node-RED y con los datos que se quiera responder al usuario. En este caso, con los que se muestra a continuación:

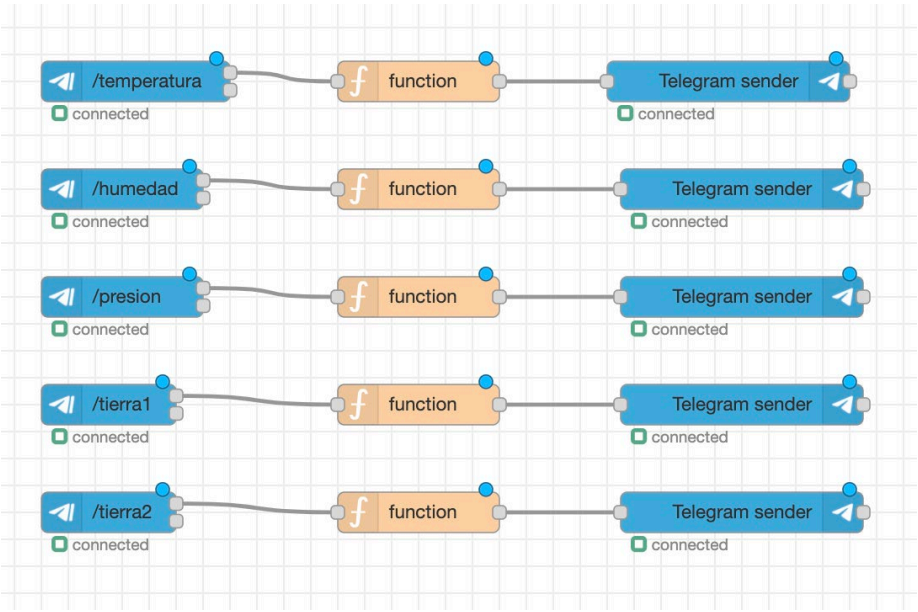

Ilustración 65. Lista de comandos 1 de Telegram en Node-RED

Por otro lado, la principal desventaja que presenta el uso de un bot con comandos, es la seguridad. Cualquier persona que conozca el nombre del bot, puede conocer los

comandos. Para ello, se ha creado un sistema para asegurar que nadie puede conocer los comandos disponibles.

Cuando se accede al bot, no se mostrará nada. Para saber los comandos que hay disponibles se tendrá que escribir en el bot */start(código)*. En el caso de este TFG, como forma de ejemplo se ha utilizado el código 0001. Por lo tanto, para ver los comandos que hay disponibles para que lo use el usuario, es */start0001*. En el caso de que se quieran añadir más usuarios o de que se quiera añadir más seguridad, se podría cambiar el código a cualquier otro.

Esto es posible, porque se ha añadido el comando */start0001,* a los conocidos por Node-RED, y la respuesta será enviar una cadena de texto con los comandos disponibles al usuario que haya escrito el comando. Esta cadena de texto, tiene la facilidad a la hora de ser representada en Telegram, de poder seleccionar directamente cualquier comando sin necesidad de escribirlo, como se ve en la ilustración 66.

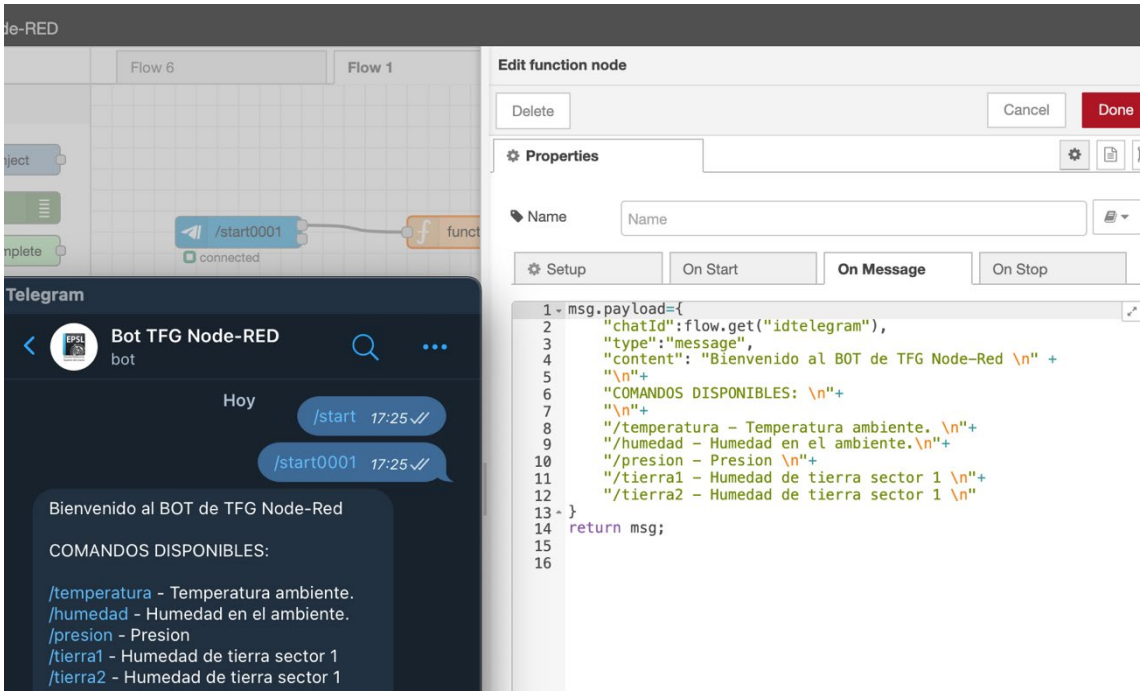

Ilustración 66. Visualización de los comandos disponibles en Node-RED

5.6 Control de los actuadores

El objetivo principal de este ejemplo práctico, es tener la capacidad de, a través de los datos recibidos por los sensores de humedad en la tierra, tener la capacidad de activar o desactivar las bombas de agua. En concreto, el modo de uso más funcional, es el automático. Este modo activa automáticamente la bomba de riego cuando el sensor detecte que la tierra está seca y se desactivará automáticamente, cuando detecte que esté mojada.

Además, se permitirá un modo manual, el cual permite activar o desactivar el motor, independientemente de que esté la tierra seca o mojada. En el caso de que esté el motor en modo automático, y se seleccione apagar o encender el motor, se desactivará el modo automático y se pasará al modo manual.

Independientemente del modo en el que se encuentre, se podrá consultar en todo momento el estado en el que se encuentre el motor, es decir, si está apagado o encendido.

En concreto, este caso práctico, contará con dos motores y dos sensores de humedad de tierra independientes, excepto en el caso de que se active el modo automático, que en este caso si se activarán los dos en modo automático. Para activarlos o desactivarlos si que se podrá hacer de forma independiente. Por ejemplo, si se activa el modo automático, ambos estarán en modo automático, pero si se ordena que se desactive el motor dos, solo se desactivará este. Si el motor 2 estaba en funcionamiento se desactivará y si ya estaba desactivado, así permanecerá. Pero en ambos casos se cambiaría a modo manual y ya no funcionará de forma automática el motor 2. Por otro lado, el motor 1 seguiría en modo automático.

5.6.1 Función de control

El control de los motores, se hará desde Node-RED, concretamente a partir del valor de la humedad de tierra. El propio fabricante, como se muestra en el apartado 5.1.1.1, dice que los valores de respuesta que se obtiene de las medidas del sensor van de 260 a

520. Considerando que entre 260 y 349 estaría sumergido en agua, entre 350 y 429 estaría la tierra húmeda y entre 430 y 520 la tierra estaría seca.

Por lo tanto, cuando Node-RED reciba el valor del servidor Mosquitto, a través de MQTT y este sea inferior a 345, se tomará que la tierra ya está lo suficientemente mojada y se detendrán las bombas de agua, mientras tanto, permanecerán activas.

Para llevar esto a cabo, se crearán cinco nuevos comandos que reconocerá Node-RED:

- */automatico*: Permitirá activar el modo automático del sistema, el cual cuando ambos sensores detecten, por separado, que la tierra está seca, se activará la bomba de cada uno de los sensores de forma independiente.
- */on1*: Activará el motor 1. En el caso de que esté apagado, lo encenderá y en el caso de que ya este activado, seguirá activo, pero de ambas formas, este comando desactivará el modo automático del motor 1.
- */off1*: Desactivará el motor 1. En el caso de que esté activado, lo desactivará y en caso de que ya este desactivado, permanecerá desactivo, pero de ambas formas, este comando desactivará el modo automático del motor 1.
- */on2*: La misma función que el */on1*, pero para el motor 2.
- */off2*: La misma función que el */off1*, pero para el motor 2.

Una vez se han creado los comandos, se añadirán a la función del comando */start0001* para que aparezcan disponibles en Telegram, en el menú de comandos disponibles.

Como se muestra en la ilustración 64, cuando se escriba algún mensaje, este se almacena en la variable "contentelegram", en el contexto del flujo completo. Por lo tanto, cuando se escriba */automatico,* o cualquier otro, se almacenará en esta variable.

La función que realice estas acciones, como se ha comentado al principio de este apartado, será a partir del valor de la humedad de tierra. Esta función, lo que hará es

analizar el contenido de la variable “contentelegram”, es decir la variable que contiene el mensaje que ha enviado el usuario. Con el fin de esperar a los comandos de actuación.

Como se ve en la ilustración 67, si el mensaje recibido es */automatico,* se activará el modo automático, es decir que cuando el valor de humedad sea menos de 345 el motor esté “OFF” y si el valor es mayor el motor esté “ON”. Además, la variable modo pasará a ser “automatico”.

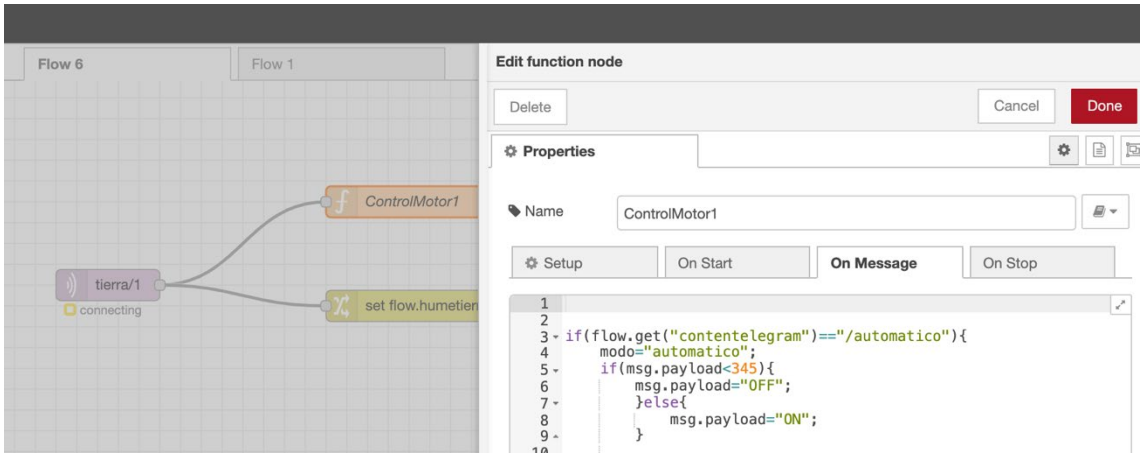

Ilustración 67. Primera condición de automatización de la función de control

El estado del motor, “ON” u “OFF”, se almacenará en la variable de salida, es decir “msg.payload”, para que el nodo de publicación de MQTT (ilustración 68), coja este mensaje y lo publique en el servidor Mosquitto, con el nuevo topic, motor/(número del motor), para que posteriormente pueda acceder a esta información el microcontrolador Arduino, y active o desactive el motor.

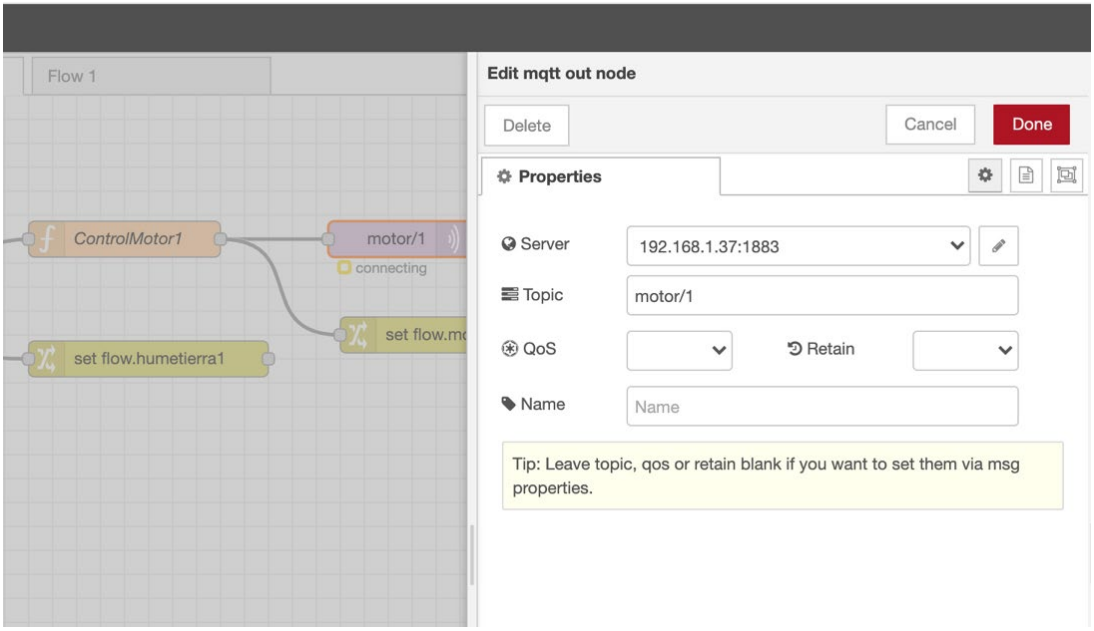

Ilustración 68. Servidor y topic donde se va a publicar la información de la función de control

Por otro lado, si se introduce el comando */on* o */off*, se activará o desactivará el motor y se pondrá en modo "manual".

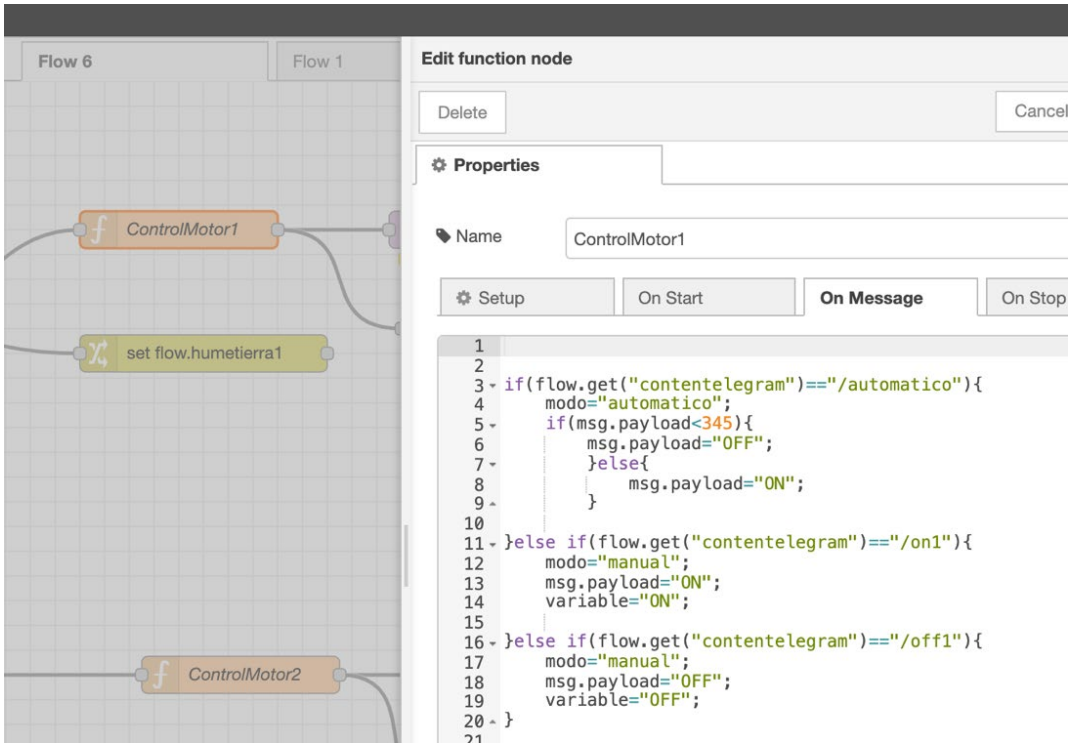

Ilustración 69. Condiciones para el establecimiento manual de la función de control

Uno de los problemas que se ha encontrado al desarrollar esta función, es que, si se activa el modo automático, utilizando el comando */automatico,* cuando posteriormente se quería consultar, por ejemplo, la temperatura, con el comando */temperatura*, se desactivaba el modo automático, ya que el comando que espera la función para que se active el modo automático es */automatico,* si recibe cualquier otro, se desactivará el modo automático.

La solución a este problema fue la creación de la variable "modo". Cuando se introduce el comando */automatico* el modo pasa a ser también automático. De esta forma, si el modo está en automático, cuando se introduzca cualquier otro comando, el motor va a seguir funcionando en modo automático.

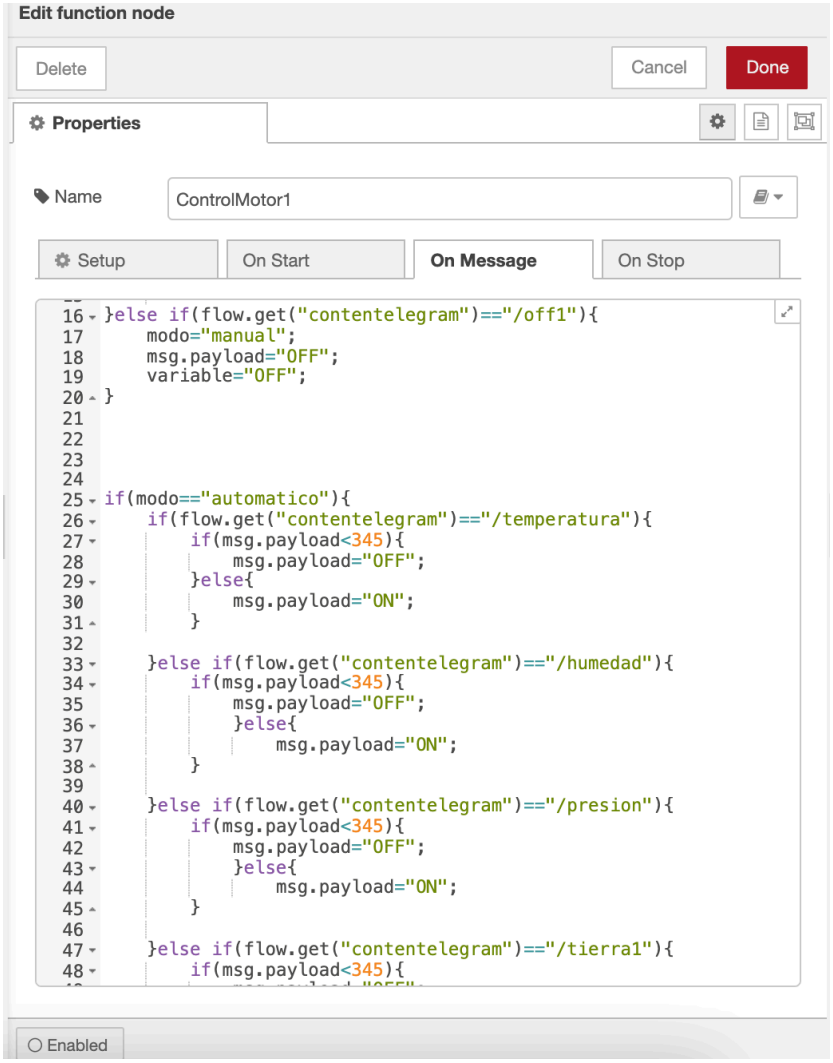

Ilustración 70. Comandos disponibles en el modo automático

De esta forma, si está en modo automático, y se recibe cualquier otro comando que no sea, */on* o */off* de su propio motor, seguirá estando en automático y mostrará la información respectiva del comando que se haya introducido.

Por otro lado, en el caso de que se escriba */on* o */off*, se desactivará el modo automático, en el caso de que esté activado, y entrará en modo manual.

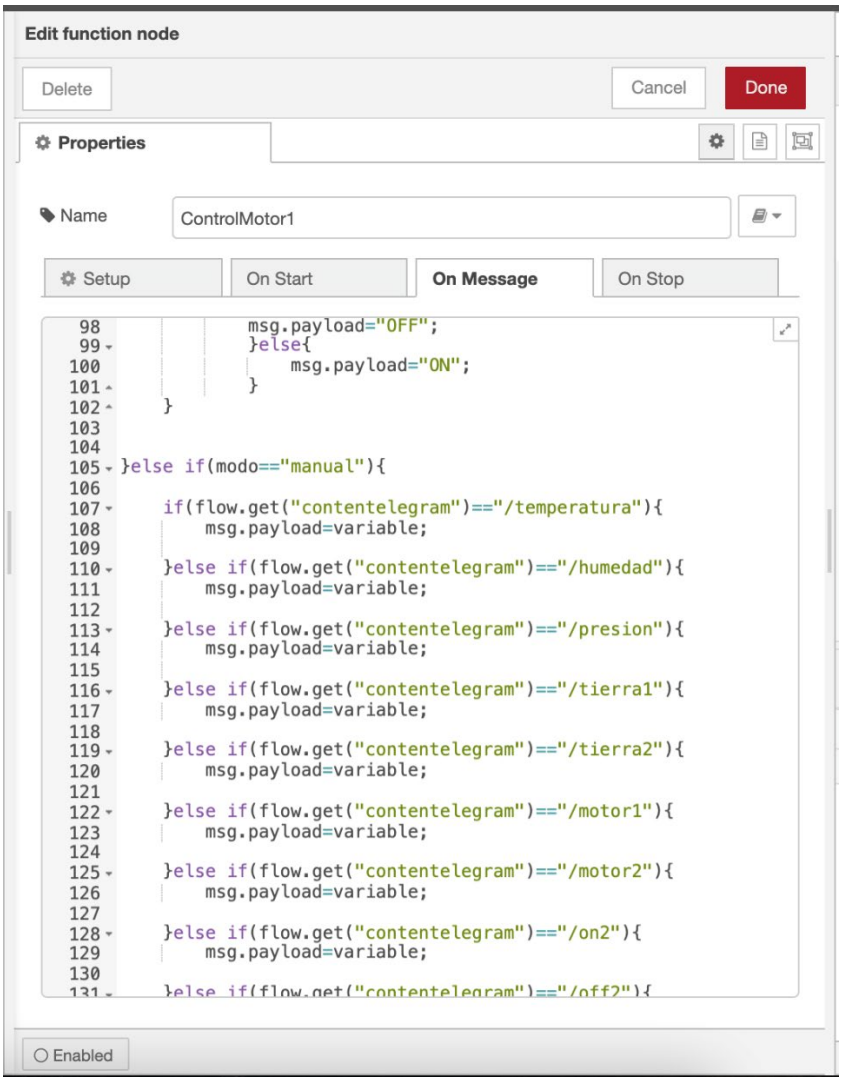

Ilustración 71. Comandos disponibles en el modo manual

En este caso si está en modo manual, y se escribe cualquier otro comando, el motor seguirá en el último estado que haya estado. Es decir, en el caso de que se escriba el comando */on,* se cambiará a modo manual, se enviará al servidor Mosquitto que el motor está "ON" y la variable del motor también estará "ON", con el fin de almacenar el estado

del motor. Para que posteriormente, cuando se escriba, por ejemplo, */temperatura,* se envíe al servidor, el último estado, almacenado en "variable".

Este proceso, se hará para los dos motores. Además, se añadirán a los comandos ya establecidos, los nuevos comandos */motor1* y */motor2*, con el fin de mostrar en que estado se encuentra cada motor. Quedando finalmente el esquema de comandos de la siguiente forma:

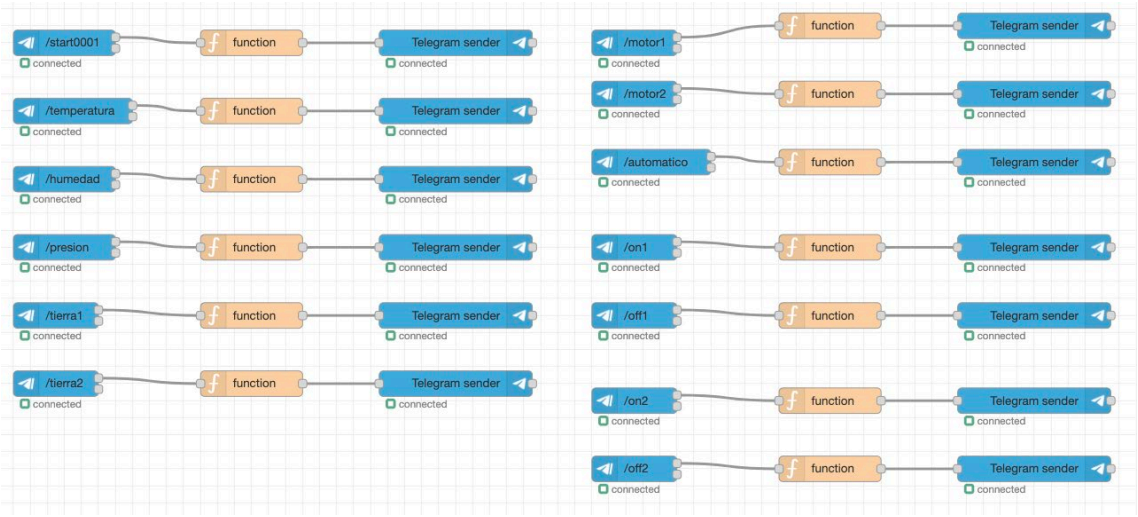

Ilustración 72. Esquema general de los comandos disponibles de Telegram

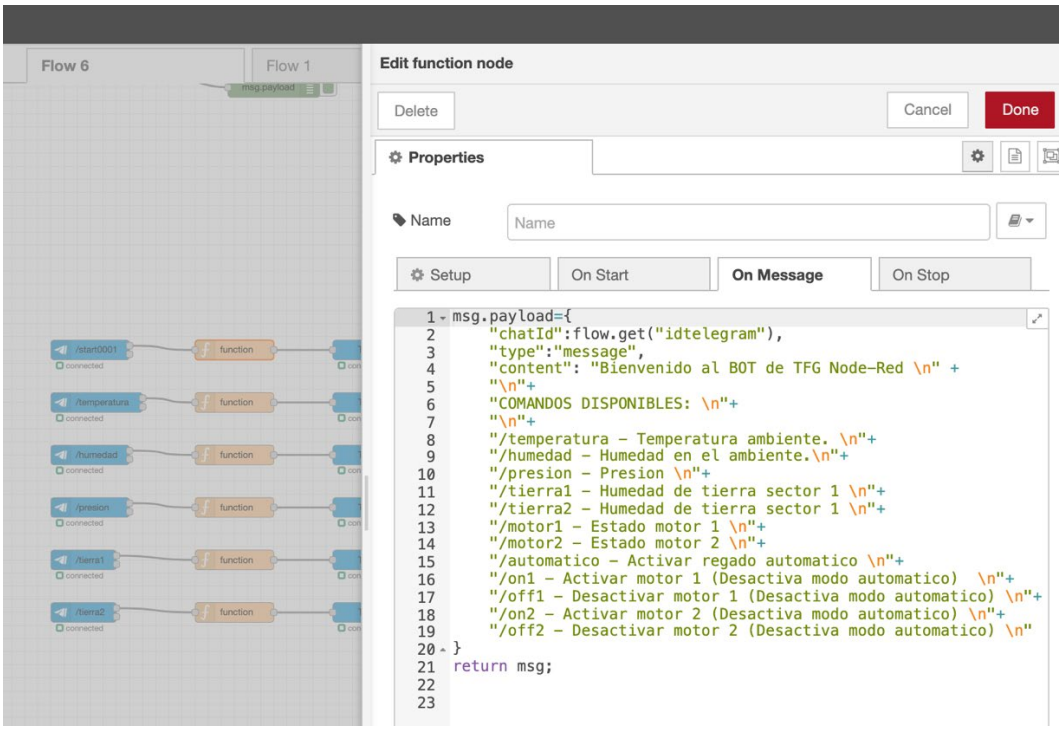

Ilustración 73. Envío a Telegram de los comandos disponibles desde Node-RED

5.6.2 Recepción en el microcontrolador

Una vez se ha publicado el estado de los motores en el servidor Mosquitto, el microcontrolador Arduino accederá a ellos, para activar o desactivar el motor. Accederá a la información que se encuentre en los topics "motor/1" y "motor/2". En ellos estará el estado de cada motor, "ON" u "OFF".

Para ello se utilizarán las mismas librerías que en los apartados 5.1 y 5.2, es decir las necesarias para acceder a la información del servidor Mosquitto, pero en este caso, en lugar de publicar la información, Arduino tendrá la función de subscritor. Cada vez que se reciba algún dato, en alguno de los 2 topics comentados, serán recibidos por el microcontrolador.

Esto es posible a la función incluida por la librería PubSubClient, denominada "callback".

```
void callback(char* topic, byte* payload, unsigned int length) {

  payload[length] = '\0'; //Establece a 0 para poder recibir varios mensajes

  if(strcmp(topic, "motor/1") == 0){ //Si el mensaje recibido tiene el topic motor/1
    PAYLOADSUB1 = (char*)payload;  //Se guarda este mensaje en la variable general
    Serial.println("Mensaje en "); //se almacena en esta variable para que todas las
    Serial.println(topic);         //funciones, en especifico loop(), puedan acceder
    Serial.println(PAYLOADSUB1);   //a este dato.

  }

  if(strcmp(topic, "motor/2") == 0){
   PAYLOADSUB2 = (char*)payload;
    Serial.println("Mensaje en ");
    Serial.println(topic);
    Serial.println(PAYLOADSUB2);
  }

}
```

Ilustración 74. Función para la subscripción al servidor Mosquitto en Arduino

Esta función cuando se actualice algún valor de los topics a los que se está subscrito, recibirá este valor. En este caso como se muestra en la ilustración 75, el topic al que está suscrito es "motor/#", es decir a "motor/1" y "motor/2", porque son los dos que hay.

```
// CONEXION CON EL SERVIDOR MQTT
  client.setServer(server, 1883); // Establecer conexion con el servidor (server) en el puerto 1883
  client.setCallback(callback);


  if (client.connect("arduinosub", mqttUsername, mqttPassword)) { //Conectarse con el usuario y la contraseña indicados
    Serial.println("MQTT CONECTADO");
    client.subscribe("motor/#");

  } else {
    Serial.println("MQTT NO CONECTADO");
    Serial.print("failed, rc=");
    Serial.println(client.state()); //En el caso de que no se conecte mostrara el error
  }
```

Ilustración 75. Topics subscritos en Arduino

Una vez se haya recibido este valor, se almacenará en una variable global, para que pueda acceder a ella todo el programa. Esto es así porque la función que se encargará de activar o desactivar el motor se encuentra en la función loop(). Como se muestra en la ilustración 76, que se almacenan en las variables PAYLOADSUB 1 y 2.

```
client.loop(); //Se utiliza para refrescar la conexión y poder recibir los datos como subscriptor

if(PAYLOADSUB1.equals("ON")){
    digitalWrite(motor1,LOW);
}

if(PAYLOADSUB1.equals("OFF")){
    digitalWrite(motor1,HIGH);
  }
if(PAYLOADSUB2.equals("ON")){
    digitalWrite(motor2,LOW);
}

if(PAYLOADSUB2.equals("OFF")){
    digitalWrite(motor2,HIGH);
  }
```

Ilustración 76. Activación y desactivación de los motores en Arduino

Por otro lado, con estos condicionales, lo que se quiere hacer es que si el dato recibido equivale a “ON” el motor se active, y si se recibe “OFF” el motor se desactive. Para ello, las variables motor1 y motor2 están definidas con el PIN digital del microcontrolador a la cual están conectados. En este caso, el motor1, se ha asociado al pin 2 y el motor2 se ha asociado al pin 3. Además, se tendrá que especificar que estos pines tendrán la función de salida.

```
const int motor1 = 2;  pinMode(motor1,OUTPUT);
const int motor2 = 3;  pinMode(motor2,OUTPUT);
```

Ilustración 77. Establecimiento de los pines de salida en Arduino

La conexión entre el microcontrolador y los motores, no será directa, si no que hará de intermediario un relé el cuál posee la capacidad de activar y desactivar las bombas según las ordenes que se le indique.

El relé hará de intermediario entre el microcontrolador y los motores. Su función es activar el motor cuando los pines estén “HIGH” y apagarlos cuando los pines estén “LOW”. Un relé es un interruptor mecánico operado eléctricamente que se puede encender o apagar dejando pasar la corriente o no, pudiéndose controlar tanto con voltajes bajos, como en este caso, como voltajes altos.[26].

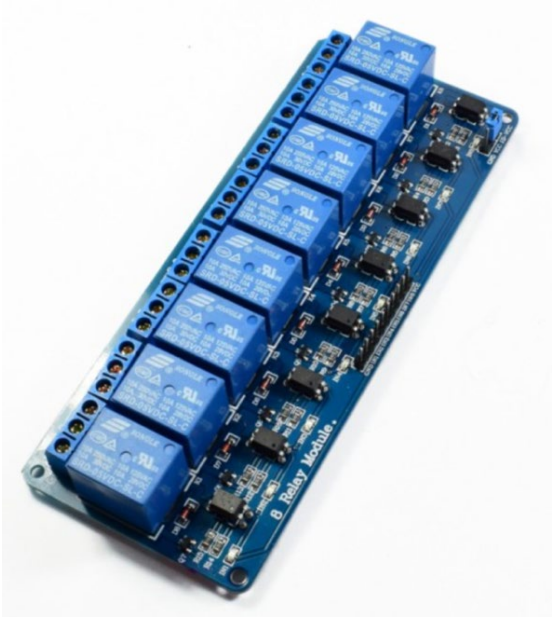

Ilustración 78. Relé utilizado

Fuente: https://tienda.bricogeek.com/interruptores/1376-modulo-8-reles-5v.html?vt_product=1376&from=UpSell&utm_campaign=Upsell%252520recommendation%252520for%252520prod&utm_medium=email&vt_campaign=1003027&utm_content=tpl&vt_content=10003986&utm_source=merchandising

El esquema interno de un relé tiene la siguiente forma:

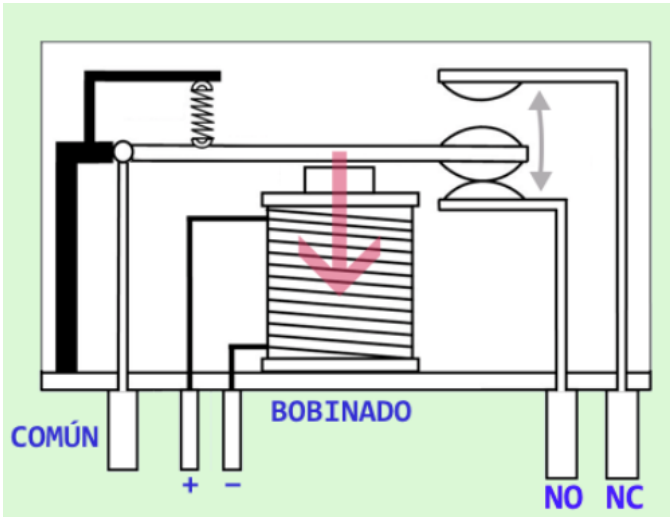

Ilustración 79. Esquema interno de un relé

Fuente: http://robots-argentina.com.ar/didactica/modulos-de-rele-y-arduino-domotica-1/

"NO" corresponde al inglés Normally Open (normalmente abierto) y "NC" a Normally Closed (normalmente cerrado). Un relé desactivado tiene unidos entre sí los contactos COMÚN y NC, y cuando se lo activa aplicando corriente a su bobina, quedan unidos entre sí los contactos COMÚN y NO.

En cuanto a las conexiones con Arduino y los motores, hay que seguir el siguiente esquema:

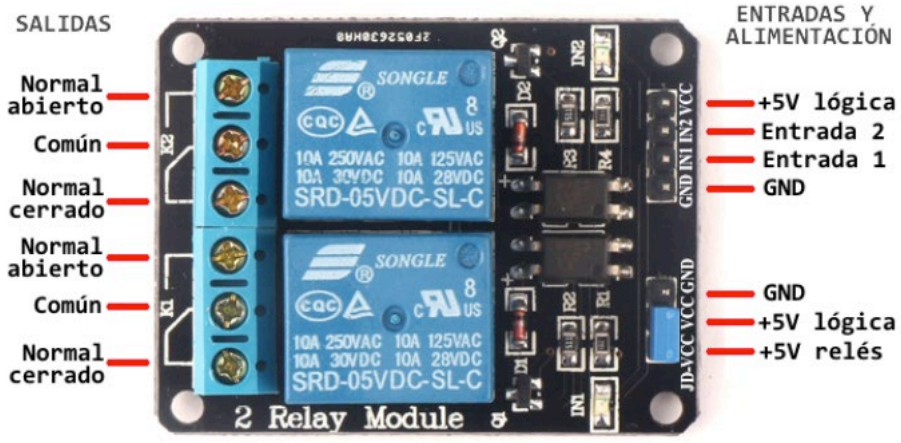

Ilustración 80. Conexiones de un relé

Fuente: http://robots-argentina.com.ar/didactica/modulos-de-rele-y-arduino-domotica-1/

Primero, se alimentará el relé, conectando la entrada Vcc a la toma de 5V de la placa y ambas GNDs. Posteriormente, se establecerán las entradas, conectando las

entradas 1 y 8 del relé con los pines digitales de Arduino 2 y 3. Esto hará que cada bomba funcione de manera independiente, para más tarde, por ejemplo, si se activa el pin digital 2, solo se active la bomba que se encuentra en la entrada del relé 1.

Para conectar las bombas de agua al relé, lo primero será alimentar esta bomba de 3V. Esto se conseguirá conectando la toma de alimentación de 3,3V de la placa Arduino con la salida común del relé.

Posteriormente, se conectará el positivo de la bomba a la salida NO (Normalmente abierto), para que el contacto con el relé siempre esté abierto y el circuito no conduzca, a menos que se envíe la señal de entrada para que se active el circuito.

Para concluir, se conectará el negativo de la bomba con la GND de la placa Arduino.

De esta forma, cuando en Arduino se ponga el pin digital 2 “HIGH”, se activará el relé 1, que es el que está conectado al motor 1. Por otro lado, si se pone el pin digital 3 “HIGH”, se activará el relé 8, que es el que está conectado al motor 2.

5.7 Almacenamiento de datos

La utilización de Telegram, para obtener todos los datos en el momento escribiendo el comando en el bot, es muy útil como una solución rápida, ya que podemos obtener la temperatura en menos de un segundo. Por otro lado, como se ha comentado en la introducción de este TFG, en IoT los datos son muy importantes, y a demás de obtener los datos actuales generados, es de alta utilidad, almacenar un histórico de todos los datos generados.

En el caso de que se quiera, por ejemplo, consultar el estado del sensor de humedad en un determinado momento, o establecer la media de la temperatura de la semana, es necesario acceder a un histórico que almacene todos los datos de forma continua. Para llevar a cabo este proceso, se hará uso de la herramienta ThingSpeak [27]. Ya que además de almacenar todos los datos, nos permitirá visualizarlos en tiempo real en forma de gráficas.

5.7.1 Creación base de datos

En primer lugar, hay que crear una cuenta en la página oficial, crear un canal y añadirle todas las variables que queramos almacenar. En mi caso:

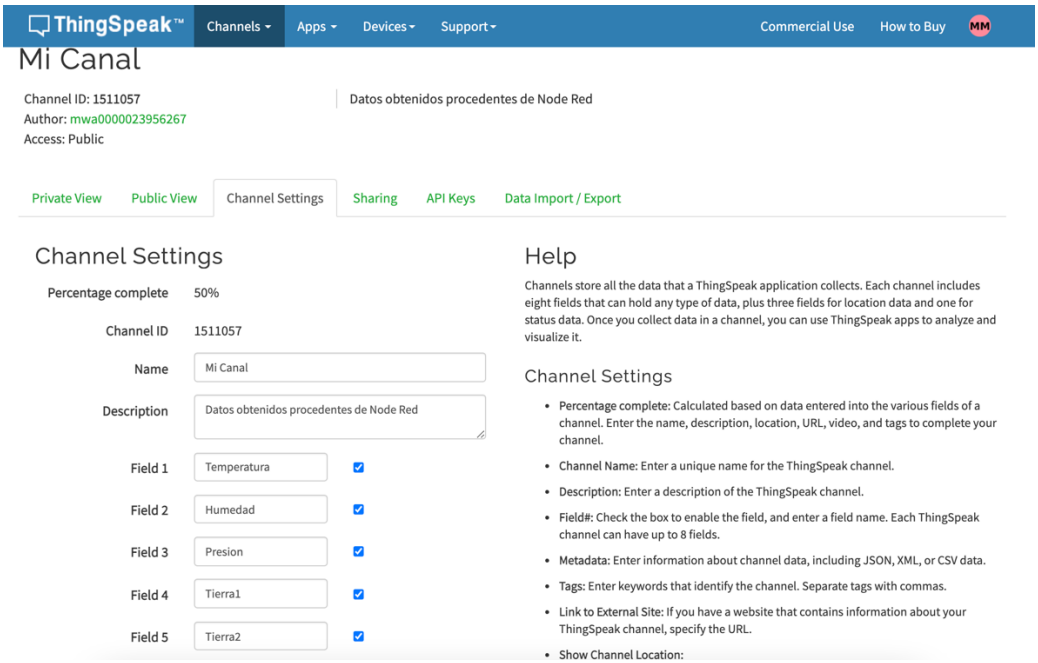

Ilustración 81. Implementación de ThingSpeak

Para comprobar que se ha realizado correctamente, se deberá de ir a la pestaña "Private View", en la cual aparecen los gráficos de las 5 variables. Como se ve en la ilustración 81.

La pestaña "Public View", por defecto viene desactivada, pero en este caso se ha activado, para que cualquier persona que tenga el enlace pueda acceder a la información que se está almacenando. Se verá lo mismo que en la pestaña "Private View".

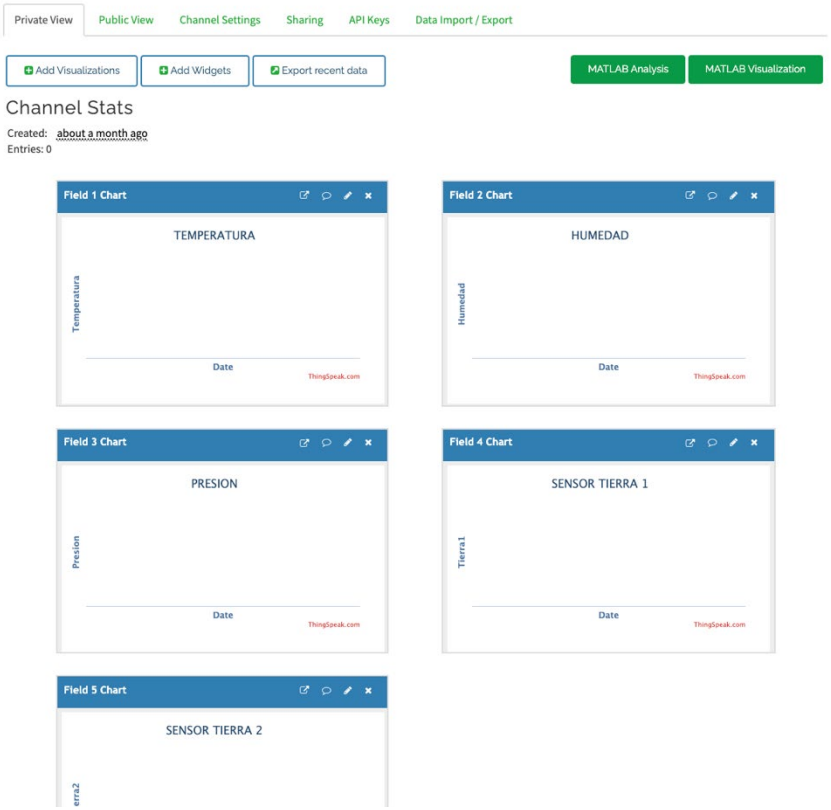

Ilustración 82. Variables introducidas en ThingSpeak

5.7.2 Implementación

Una vez se ha creado correctamente la base de datos, se procederá a implementarla desde Node-RED, ya que será la fuente de los datos que se van a almacenar.

En un primer lugar, se utilizó para esta acción, un bloque disponible de ThingSpeak para Node-RED, qué, aunque no estuviera de forma nativa, si podía añadir a nuestra paleta de nodos.

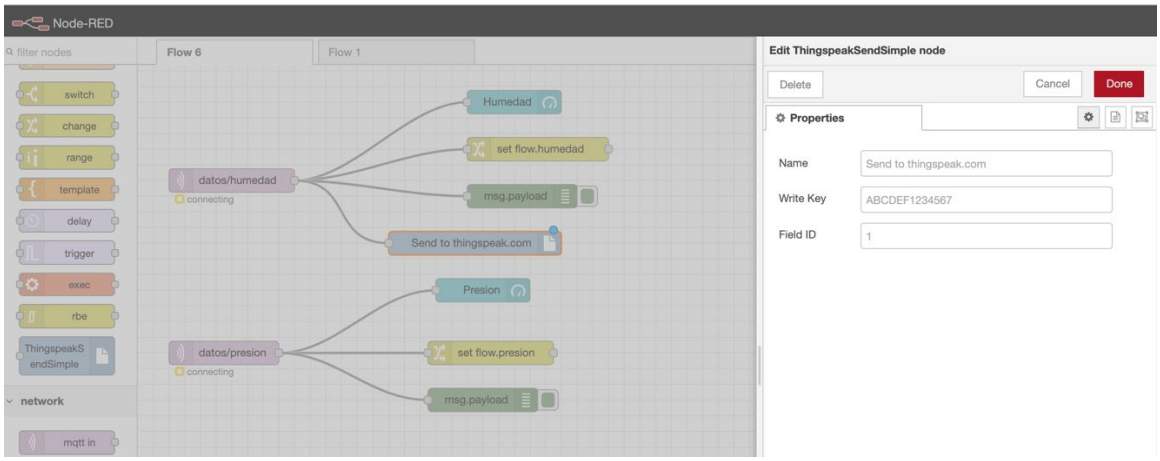

Ilustración 83. Uso del nodo "thingspeak" para el envío de datos en Node-RED

Los datos necesarios para el envío de la información se encuentran en la pestaña "API Keys", la cual nos proporciona las claves necesarias para poder publicar los datos, en este caso es desde Node-RED, pero también se puede utilizar desde otros programas, como puede ser el caso de Matlab.

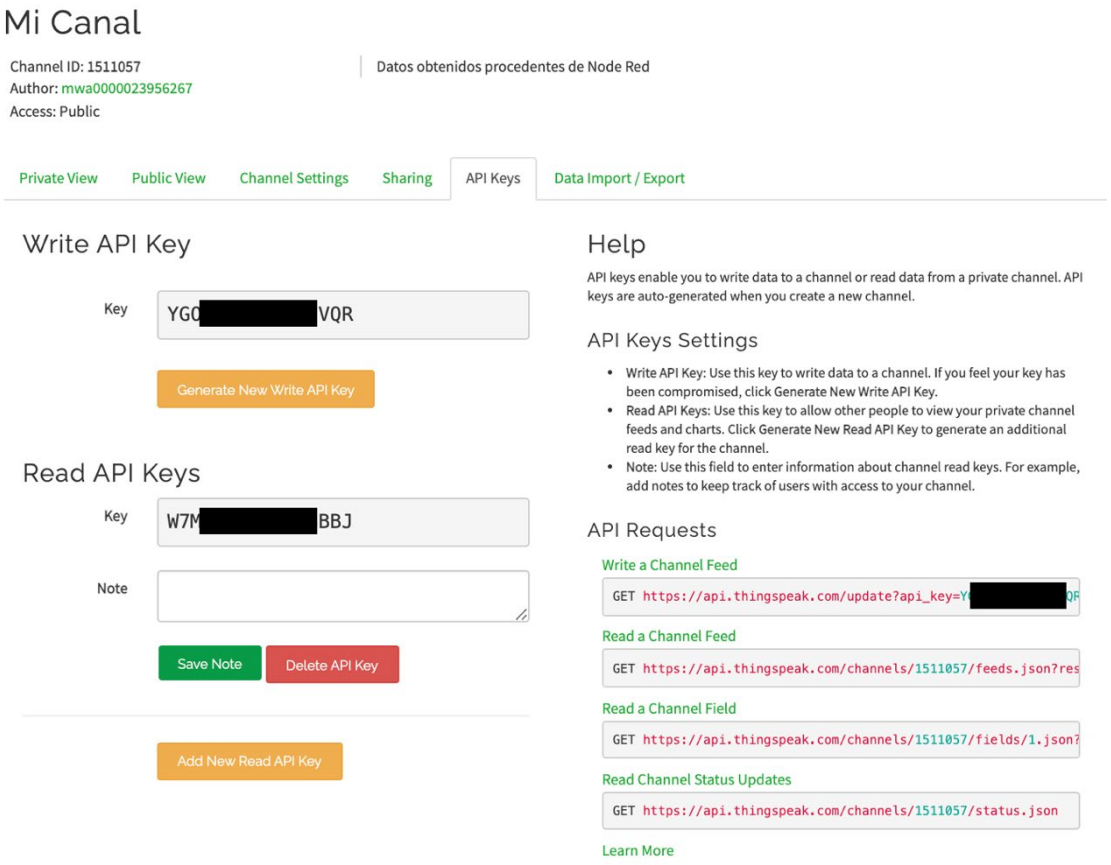

Ilustración 84. API Keys en ThingSpeak

Nos proporcionará las claves necesarias para acceder a los datos y lo más importante para escribir los datos en la base de datos, que es la que vamos a utilizar.

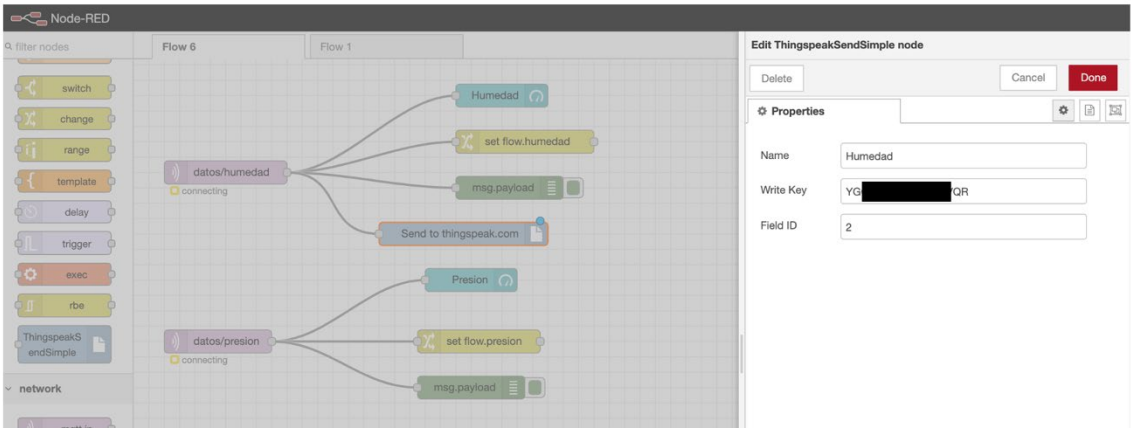

Ilustración 85. Utilización del nodo "thingspeak" en Node-RED

Una vez se han introducido las credenciales correctamente, en este caso como se ve en la ilustración 85, será el dato de la humedad, se introduce la "Write Key" que se encuentra en la pestaña "API Keys" y el numero de gráfico en el que se mostrará esta información.

Una vez se ha comprobado que está todo correcto, la información por algún motivo no se publica, de ninguno de los cinco datos que se pretenden almacenar.

Tras investigar la fuente del problema, se llega a la conclusión de que el problema es el nodo de Node-RED, ya que se ha quedado obsoleto y no se ha actualizado, como se muestra en la ilustración 86, lleva sin actualizarse cinco años. Además, no se trata de un nodo oficial de la página ThingSpeak, si no que es un nodo creado por un usuario. Por lo tanto, cabe destacar que la herramienta de añadir nodos es muy potente, ya que muchas de las páginas oficiales, posee un paquete de nodos específico, como es el caso, por ejemplo, del también usado Telegram. Pero también hay que tener en cuenta de que todas las páginas no lo poseen, y los nodos pueden ser creados por cualquier usuario, y en el caso de que no se esté actualizando continuamente, se quedará el nodo obsoleto e inservible.

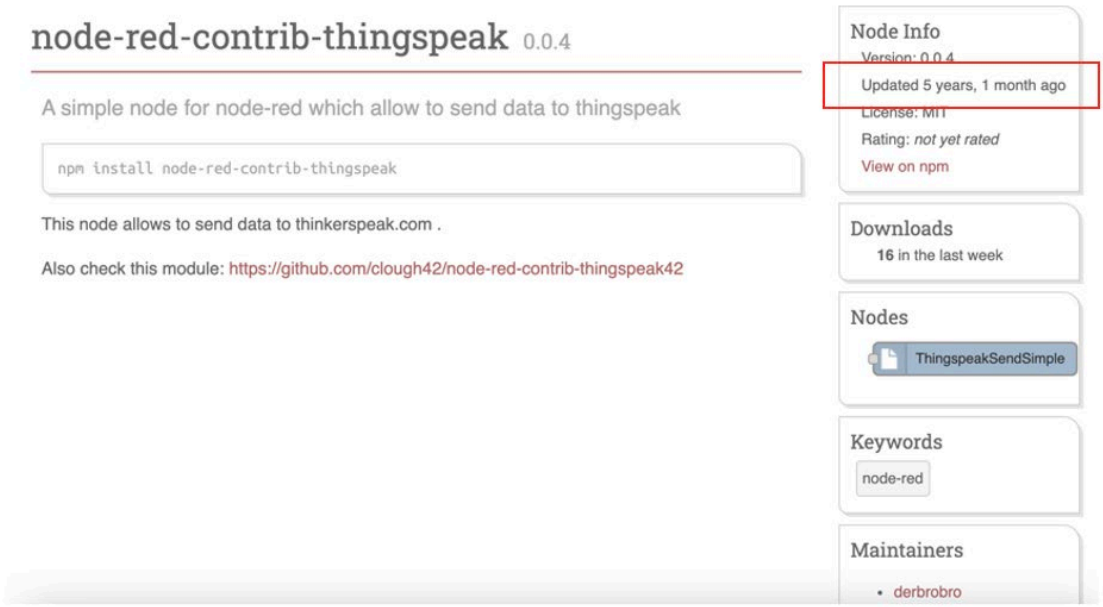

Ilustración 86. Información del nodo "thingspeak"

Una vez conocido este problema, hay que buscar una solución, y esta solución se encuentra en la pestaña "API Keys", mostrada en la ilustración 84. ThingSpeak, cuando crea un canal, además de generar una serie de claves, para utilizarlas para publicar la información, también nos genera una serie de URLs, los cuales actuarán como peticiones HTTP, para acceder a la información de la base de datos o para escribir en ella.

En este caso, se utilizará el enlace "Write a Channel Feed", el cuál nos genera una URL, que tenemos que completar con el dato que se quiere enviar y el número de gráfica al que está dirigido ese dato. La principal ventaja de utilizar este método es que el propio Node-RED, posee un nodo de forma nativa para realizar peticiones HTTP.

Toda la información se enviará en la misma petición HTTP, que estará formada como se muestra en la ilustración 87. A partir de la URL de base generada por ThingSpeak, se le añadirá a la URL, en número de gráfica en la que se quiere almacenar el valor, y el valor en sí.

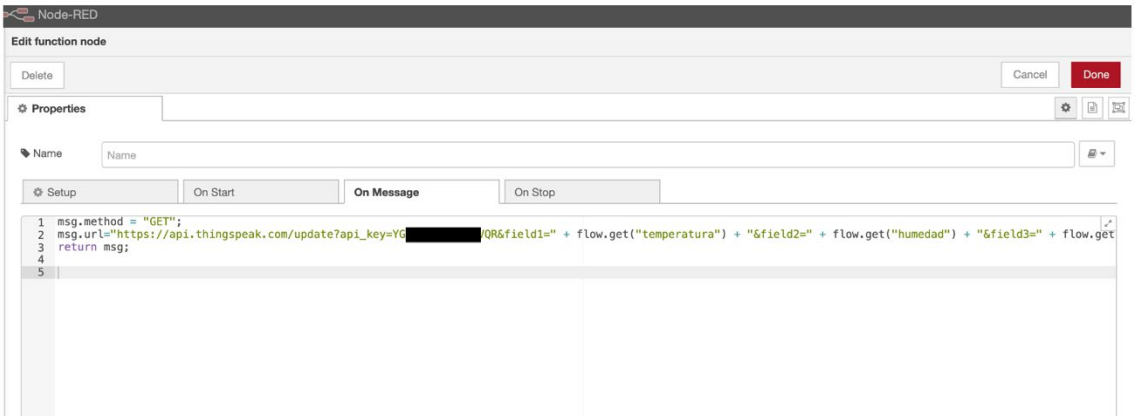

Ilustración 87. Envío de una petición HTTP con los datos de Node-RED a ThingSpeak

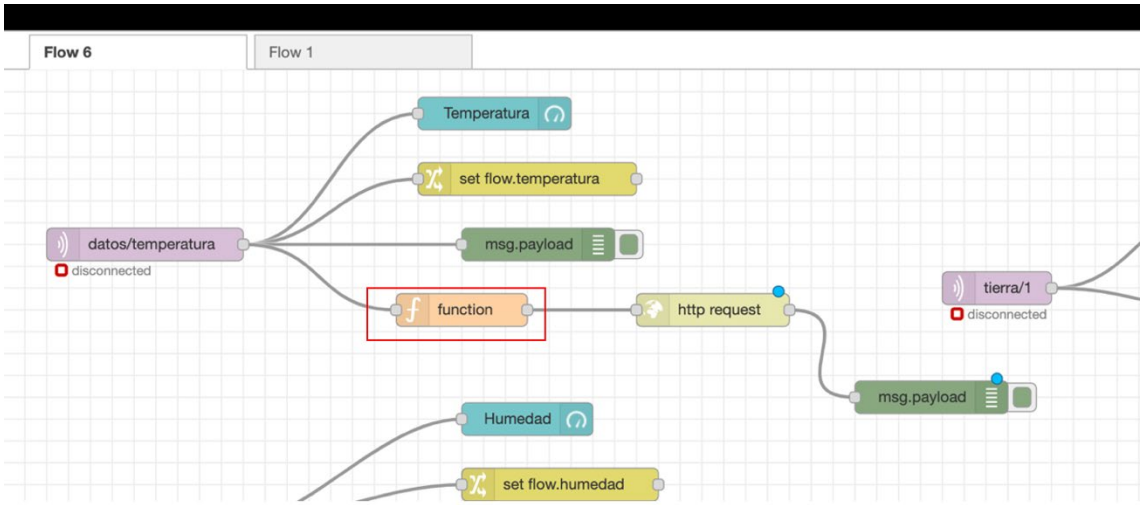

Ilustración 88. Esquema general del envío de la información a ThingSpeak

La petición HTTP contendrá todos los datos, y se enviará cada vez que se reciba un nuevo dato de temperatura, aunque en esta petición se encuentren los datos de las cinco variables que se pretenden almacenar.

En el nodo de "http request", no hay que configurar nada, ya que toda la configuración necesaria y la URL se encuentra en el nodo de la función.

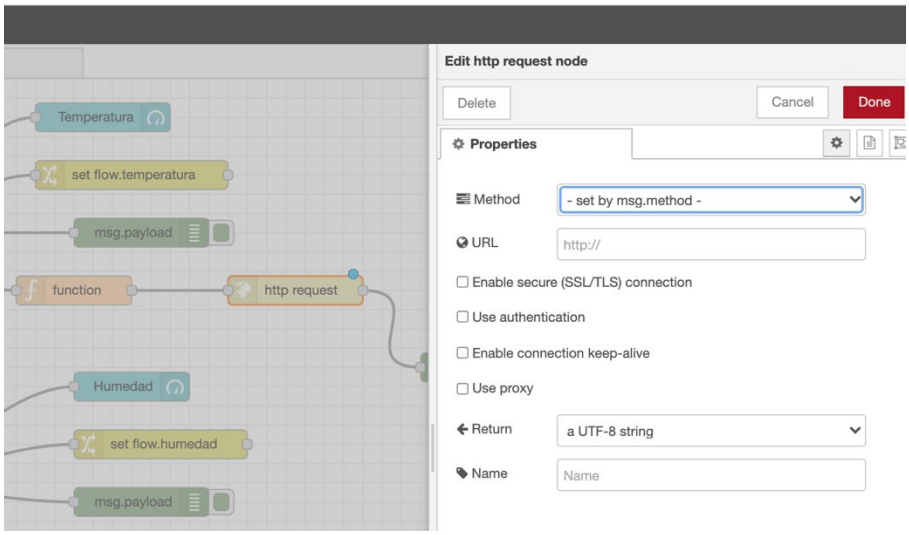

Ilustración 89. Configuración del nodo "http request" en Node-RED

5.7.3 Representación

Una vez se ha realizado el proceso anterior correctamente, se comenzarán a publicar los datos en las gráficas. Como se ve a continuación:

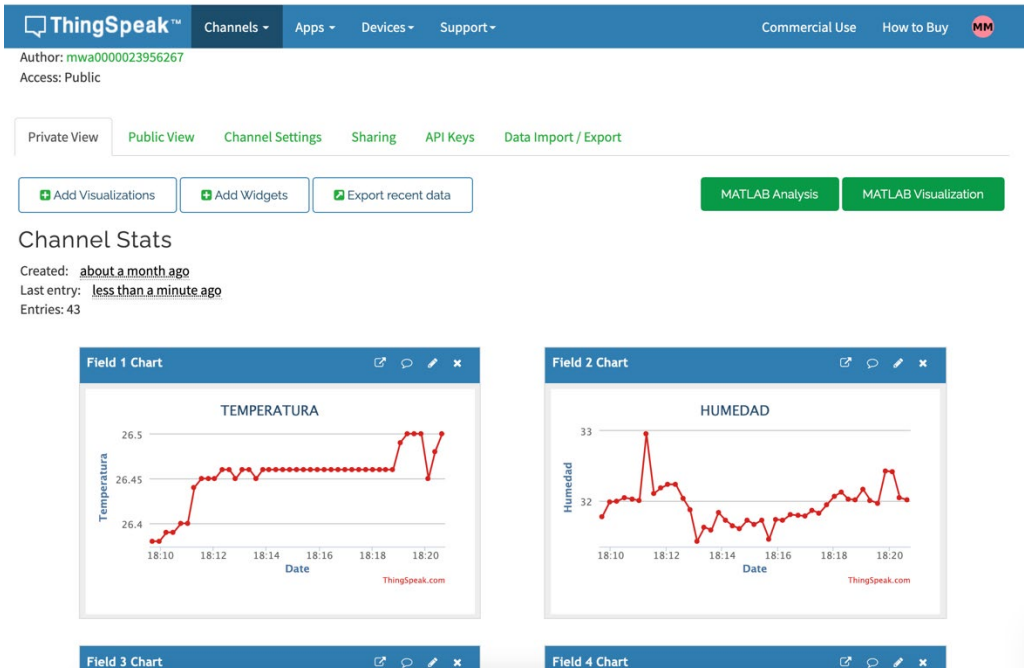

Ilustración 90. Representación de la información en ThingSpeak

Este ejemplo, es producto del funcionamiento durante 10 minutos.

Un gráfico más representativo es el de la humedad de tierra. El cual, se muestra casi todo el tiempo seco, con valores superiores a 500. Pero en algunos valores se producen picos por debajo de 400 e incluso menos. Esto es debido, a que, como forma de prueba, se han humedecido.

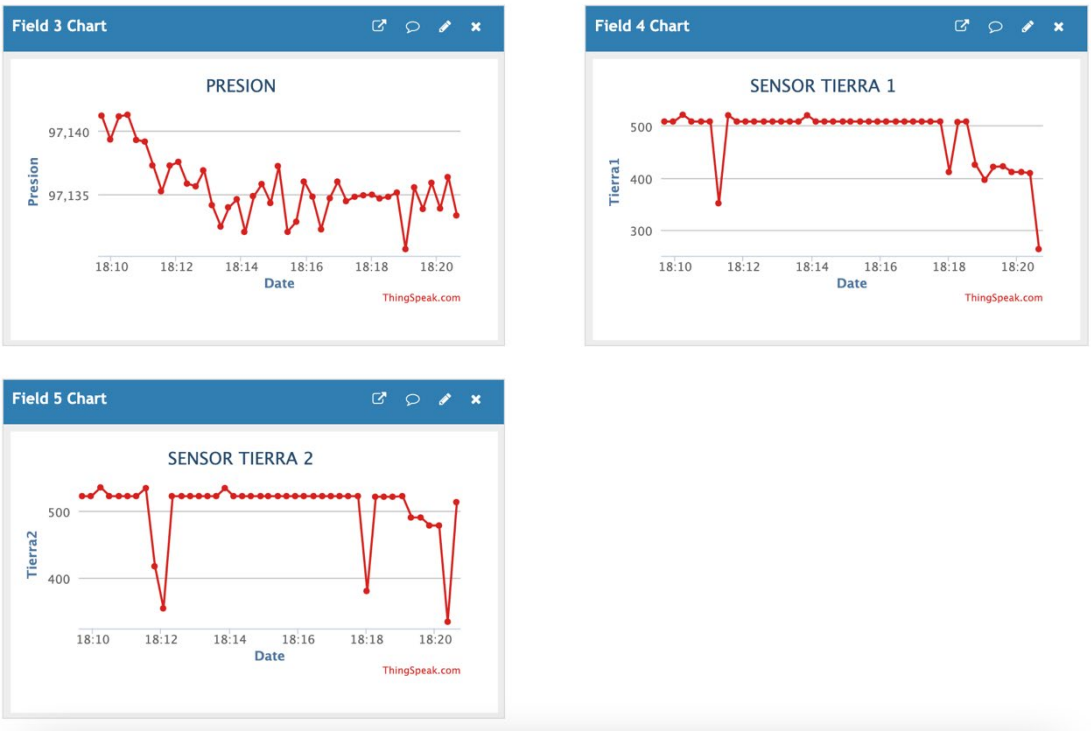

Ilustración 91. Cambio en los datos recibidos en ThingSpeak

Todos estos datos obtenidos y almacenados se pueden exportar en formato CSV.

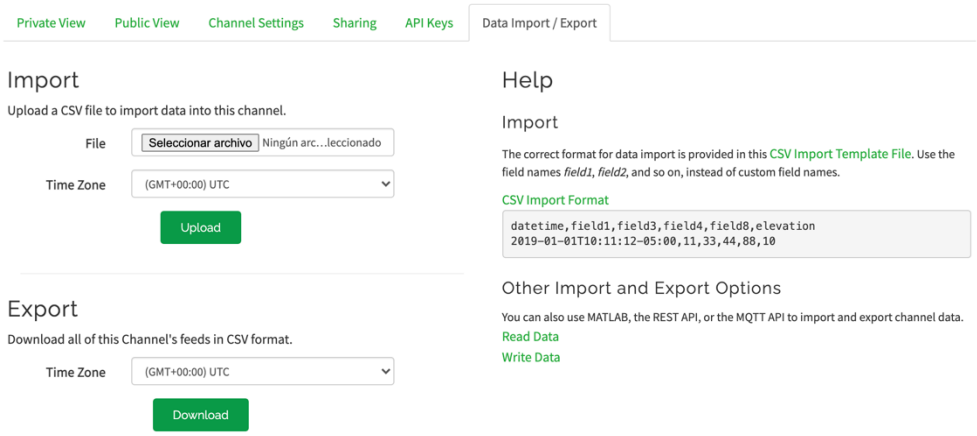

Ilustración 92. Exportación de los datos de ThingSpeak

5.8 Presupuesto

En este apartado se detallará el presupuesto para llevar a cabo la parte práctica de este TFG.

5.8.1 Materiales

Ud.	Concepto	P. Unitario	Subtotal
1	Cables DuPont Macho - Macho (20 cm / 40 unidades)	1,90 €	1,90 €
1	Cables DuPont Macho - Hembra (20 cm / 40 unidades)	1,90 €	1,90 €
1	Módulo 8 relés 5V	4,10 €	4,10 €
1	Sensor de temperatura, humedad y presión BME280	19,95 €	19,95 €
1	Arduino Uno WiFi Rev2	41,30 €	41,30 €
1	Placa de prototipo 8x5cm - 400 puntos	1,95 €	1,95 €
1	Cable USB (A/B)	3,70 €	3,70 €
2	Sensor de humedad de suelo anticorrosión	6,90 €	13,80 €
1	Batería externa ADDTOP de 26800mAh, USB C con PD 18W	29,95 €	29,95 €
1	Mini bomba de Agua DC 3V + 3M Tubería de PVC (pack 3 piezas)	21,50 €	21,50 €
1	Tablero de contrachapado de pino 250x125x15 mm	40,00 €	40,00 €
1	Tablero aglomerado 244x122x10 mm	7,99 €	7,99 €
	TOTAL		**188,04 €**

Tabla 1. Presupuesto materiales

El coste total de la compra de todos los materiales necesarios asciende a 188,04€ sin IVA.

5.8.2 Mano de obra

Horas	Concepto	Precio/Hora	Total
2	Especifiación de los requisitos	30,00 €	60 €
6	Diseño	30,00 €	180 €
12	Implementación	30,00 €	360 €
2	Pruebas	30,00 €	60 €
	TOTAL		**660 €**

Tabla 2. Presupuesto mano de obra

El coste de la mano de obra asciende a 660€ sin IVA.

5.8.3 Presupuesto total

CAPÍTULO	IMPORTE
Materiales	188 €
Mano de obra	660 €
TOTAL SIN IVA	848 €
IVA (21%)	178,08 €
TOTAL CON IVA	**1.026,08 €**

Tabla 3. Presupuesto total

Asciende el presente presupuesto de la parte de diseño e implementación de este TFG a la cantidad de MIL VEINTISÉIS EUROS CON OCHO CÉNTIMOS DE EURO.

6. Resultado y discusión

El resultado de este TFG ha sido una guía completa de la herramienta Node-RED, además de la incorporación de un caso práctico para implementar lo aprendido.

En primer lugar, se ha estudiado de forma general en que consiste IoT, en que ámbito se aplica y de que forma lo hace. Además de las tecnologías y protocolos que se utilizan en la actualidad. Una vez desarrolladas todas las tecnologías y protocolos se decidió utilizar MQTT como protocolo IoT. Por otro lado, como microcontrolador se utilizará Arduino.

Una vez se han establecido todos los medios relacionados con IoT, como pueden ser el protocolo de uso y el microcontrolador que proporcionará los datos, se profundiza en Node-RED. En cuanto a esta herramienta objeto de este TFG, en primer lugar, se hace una introducción a sus fundamentos, es decir, como está formado y sus principales características, para posteriormente entrar a su entorno de uso y como se utiliza básicamente.

A continuación, se realiza una guía, qué mediante ejemplos y casos prácticos, se explica el verdadero funcionamiento de Node-RED. Explicando en todo momento como funciona su interfaz, el uso de los nodos y las demás herramientas que nos proporciona.

En segundo lugar, se implementa un caso práctico para poner en práctica los conocimientos adquiridos. En concreto, se explica la captación de datos desde el microcontrolador, el envío de la información a un broker y por supuesto, el uso de estos datos en Node-RED, con todo tipo de nodos y funciones, generando así la lógica del programa.

El caso práctico costa de una maqueta que simula un sistema de riego automático, el cuál permite su control desde la app Telegram.

7. Conclusiones y líneas futuras

Como conclusión final, cabe destacar que Node-RED es una herramienta muy potente y relativamente fácil de usar, incluso para las personas que no tengan muchos conocimientos de programación. Si se adquieren los conocimientos básicos, como son el uso de los nodos, las variables y los datos, se pueden crear aplicaciones y servicios de gran tamaño. Siendo una gran ventaja que se trate de un software libre y con una gran comunidad activa.

Esta ventaja de que haya una gran comunidad activa, se puede volver en nuestra contra, ya que cualquier persona puede crear un nodo y en el caso de que se utilice, se tiene que tener precaución de que esté actualizado y que funcione correctamente.

Como líneas futuras, se propone:

- Uso de otro protocolo IoT para las comunicaciones. En este caso, se ha escogido por las razones justificadas, MQTT. Pero se podrían usar otros protocolos como son OPC-UA.
- Creación de un nodo propio para un uso específico, ya que, en el desarrollo de este TFG, en todo momento se han utilizado nodos incluidos por el propio Node-RED o nodos que hayan sido creados por otros usuarios, pero no nodos de creación propia.
- Implementación de otro tipo de comunicación con el usuario, como podría ser la creación de una aplicación, en lugar de los comandos de Telegram.
- Estudiar el uso de Node-RED en otra plataforma, como Raspberry Pi.
- Usando el mismo ejemplo de uso, añadir otro tipo de sensores, como podría ser un sensor de imagen, para captar el crecimiento de una planta o su color, para detectar cualquier perturbación, manejando así otro tipo de datos.
- Incluir la capacidad de que se puedan añadir mas usuarios automáticamente.
- Incluir otro ejemplo de uso, como podría ser, un sistema de alarma o la creación de una SmartHome.

8. Bibliografía

[1] K. ’Eldridge, S. ’Chapin, L. ’Rose, *La Internet de las Cosas - Una breve reseña*. Internet Society, 2015.

[2] S. Greengard, *The Internet of Things*. 2015.

[3] L. Llamas, “Protocolos de comunicación para IoT,” *https://www.luisllamas.es/protocolos-de-comunicacion-para-iot/*, 2019.

[4] M. Massimi, “Protocolos: La comunicación para IoT,” *https://www.murkyrobot.com/review/domotica/protocolos-comunicacion-iot*, 2019.

[5] I. Porro Sáez, “IoT: protocolos de comunicación, ataques y recomendaciones,” *https://www.incibe-cert.es/blog/iot-protocolos-comunicacion-ataques-y-recomendaciones*, Feb. 07, 2019.

[6] Isaac, “MQTT: un protocolo abierto de red y su importancia en IoT,” *https://www.hwlibre.com/mqtt/*, May 15, 2020.

[7] M. Calabretta, “MQTT-AUTH: A TOKEN-BASED SOLUTION TO ENDOW MQTT,” *JOUNAL OF COMMUNICATIONS SOFTWARE AND SYSTEMS* , vol. 12, Dec. 2018.

[8] B. Canet, “Fundamentos de seguridad de MQTT en la automatización industrial ,” Jul. 14, 2020.

[9] “MQTT Sotware - Servers/Brokers,” *https://mqtt.org/software/*.

[10] E. Crespo, “Aprendiendo Arduino - Mosquitto,” *https://aprendiendoarduino.wordpress.com/2018/11/19/mosquitto/*, Oct. 25, 2019.

[11] “¿Qué es Arduino?,” *https://arduino.cl/que-es-arduino/*.

[12] iotguider, “Hardware Basics of Arduino UNO WiFi REV2,” *https://iot-guider.com/arduino/hardware-basics-of-arduino-uno-wifi-rev2/*, Dec. 06, 2018.

[13] “Node-Red, About,” *https://nodered.org/about/*.

[14] L. del Valle Fernández, “Introducción a Node-Red,” *https://programarfacil.com/blog/raspberry-pi/introduccion-node-red-raspberry-pi/*, 2019.

[15] P. Sancho, “Fundamentos de Node-Red,” *https://www.techedgegroup.com/es/blog/fundamenos-node-red*, Apr. 20, 2020.

[16] "Node-RED Alternatives," *https://www.saashub.com/node-red-alternatives*, 2021.

[17] "Zapier, connect your apps and automate workflows," *https://zapier.com*.

[18] C. Borges, "Zapier: aprende a utilizar el mensajero del Marketing para integrar tus procesos," *https://rockcontent.com/es/blog/zapier/*, Jan. 2020.

[19] Adafruit, "Adafruit BME280 I2C or SPI Temperature Humidity Pressure Sensor - STEMMA QT," *https://www.adafruit.com/product/2652*.

[20] L. Llamas, "Sensor ambiental con Arduino y BME280," *https://www.luisllamas.es/sensor-ambiental-arduino-bme280/*, Apr. 15, 2020.

[21] L. Llamas, "El bus I2C en Arduino," *https://www.luisllamas.es/arduino-i2c/*, May 16, 2016.

[22] Node-Red, "Working with context," *https://nodered.org/docs/user-guide/context*.

[23] Node-Red, "The Core Nodes," *https://nodered.org/docs/user-guide/nodes#function*.

[24] Node-Red, "node-red-dashboard," *https://flows.nodered.org/node/node-red-dashboard*.

[25] "Telegram Web Page," *https://telegram.org*.

[26] L. Llamas, "Manejar salida con relé," *https://www.luisllamas.es/arduino-salida-rele/*, 2016.

[27] "ThingSpeak for IoT Projects," *https://thingspeak.com/*.

[28] "Node.js," *https://nodejs.org/es/*.

[29] "Running Node-Red locally," *https://nodered.org/docs/getting-started/local*.

[30] X. Cosmed, "Qué es Homebrew y para qué sirve," *https://lamanzanamordida.net/homebrew/*, 2017.

9. Anexos

9.1 Instalación Node-RED

Como se ha comentado en el apartado 4.3, Node-RED basa su funcionamiento en Node.js, por lo tanto, será lo primero en instalarse, desde su página oficial [28].

En el caso del sistema operativo MacOS, una vez instalado se hará uso de la herramienta que incluye Node.js, NPM (Node Package Management), la cual es un gestor de paquetes que se ha desarrollado bajo el lenguaje JavaScript. Esta herramienta nos permite obtener cualquier librería con un simple comando. En este caso, para instalar Node-RED, el comando es el siguiente [29]:

sudo npm install -g --unsafe-perm Node-RED

En el caso de que se esté usando Windows, ***sudo*** no será necesario. Este comando, instalará Node-RED como un módulo global con todas sus dependencias.

Una vez se ha instalado, se puede abrir la interfaz web, que por defecto corre en el puerto 1880.

9.2 Instalación Eclipse Mosquitto

En el caso de MacOS, se instalará en el terminal la herramienta Homebrew, la cuál nos permite instalar desde el terminal, herramientas que no vienen de serie en el Mac, como es el caso de Eclipse Mosquitto [30].

Una vez se ha instalado Homebrew, se procederá a instalar Eclipse Mosquitto, mediante la siguiente línea en la consola:

brew install mosquitto

Una vez instalado, para iniciarlo se utilizará la siguiente línea en consola:

brew services start mosquitto

Para ver la configuración del servidor y en que puertos está funcionando se utilizará:

/usr/local/sbin/mosquitto -c /usr/local/etc/mosquitto/mosquitto.conf

Para comprobar que se ha instalado de forma correcta, como cliente, se puede hacer la función de publicador o de subscriptor. El publicador añadirá la información que desee al servidor acompañada siempre de un topic, mientras que el subscriptor accederá a dicha información.

Publicar:

mosquitto_pub -d -t "test" -m "hola_mundo" -u "usuario" -P "usuario"

En el cual "test" será el topic, "hola_mundo" el mensaje y "usuario" el usuario y la contraseña que en este caso es la misma.

9.2.1 Creación de credenciales

Desde el sistema operativo MacOS, se debe realizar todo desde el terminal. La creación de un usuario y contraseña se debe realizar, ya que, si no se crea, el servidor Mosquitto interpreta que solo funcionará el localhost (127.0.0.1) y no permitirá conexiones entrantes.

Para ello se utilizará el comando:

sudo mosquitto_passwd -c /usr/local/etc/mosquitto/passwd usuario

Mediante el comando mosquitto_passwd se acceden a las opciones de contraseñas de Mosquitto. En este caso -c creará un nuevo fichero de contraseña con el nombre "passwd", en el caso de que ya existiera se sobrescribiría.

Por otro lado, si lo que se quiere hacer es eliminar un usuario en concreto en lugar de -c, se escribiría -D.

Después de haber introducido el comando, se creará un usuario, en este caso con el nombre "usuario" y se pedirá posteriormente la contraseña por dos veces que se le quiere asignar a ese usuario. Una vez finalizado, se habrá creado el fichero con la ruta indicada.

9.2.2 Habilitar autenticación

Para habilitar la autenticación correctamente del servidor Mosquitto, se debe modificar el archivo mosquitto.conf.

En mi caso se ha presentado el problema, que desde el sistema operativo MacOS, no se puede acceder directamente al directorio del archivo mosquitto.conf, para ello se ha descargado el programa Sublime Text 2, el cual, posee una funcionalidad que me permite abrir cualquier archivo desde consola.

Dentro de la carpeta de instalación de Sublime Text 2 posee un ejecutable que recibe el nombre de "subl", que es el que nos permitirá abrir todos los archivos desde la consola.

Para abrir el comentado mosquitto.conf, primero en el terminal cambiamos el directorio en el cual se encuentra el archivo:

cd /usr/local/etc/mosquitto/

Para asegurarnos de que están todos los archivos escribiremos ls y así también nos aseguramos de que se encuentra el nuevo archivo "passwd" creado.

Posteriormente utilizaremos para abrirlo la herramienta de Sublime Text 2, para la cual se ha de poner el directorio en el que se encuentra el ejecutable "subl" seguido del archivo que se quiere abrir, en este caso mosquitto.conf. Así quedaría el comando:

/Applications/Sublime\ Text\ 2.app/Contents/SharedSupport/bin/subl mosquitto.conf

Y se ejecutará el archivo dentro del programa. Una vez se ha ejecutado hay que dejar visibles algunas líneas de este fichero como son:

```
493  # =================================================================
494  # Security
495  # =================================================================
496
497  # If set, only clients that have a matching prefix on their
498  # clientid will be allowed to connect to the broker. By default,
499  # all clients may connect.
500  # For example, setting "secure-" here would mean a client "secure-
501  # client" could connect but another with clientid "mqtt" couldn't.
502  #clientid_prefixes
503
504  # Boolean value that determines whether clients that connect
505  # without providing a username are allowed to connect. If set to
506  # false then a password file should be created (see the
507  # password_file option) to control authenticated client access.
508  #
509  # Defaults to false, unless there are no listeners defined in the configuration
510  # file, in which case it is set to true, but connections are only allowed from
511  # the local machine.
512  allow_anonymous false
513
514  # -----------------------------------------------------------------
515  # Default authentication and topic access control
516  # -----------------------------------------------------------------
517
518  # Control access to the broker using a password file. This file can be
519  # generated using the mosquitto_passwd utility. If TLS support is not compiled
520  # into mosquitto (it is recommended that TLS support should be included) then
521  # plain text passwords are used, in which case the file should be a text file
522  # with lines in the format:
523  # username:password
524  # The password (and colon) may be omitted if desired, although this
525  # offers very little in the way of security.
526  #
527  # See the TLS client require_certificate and use_identity_as_username options
528  # for alternative authentication options. If an auth_plugin is used as well as
529  # password_file, the auth_plugin check will be made first.
530  password_file /usr/local/etc/mosquitto/passwd
```

Ilustración 93. Configuración del servidor Mosquitto 1

Mediante el “allow_anonymous false”, lo que se está consiguiendo es que la autenticación sea obligatoria, es decir, que un usuario no pueda acceder si no se ha registrado correctamente.

Por otro lado, mediante el “password_file” se está incluyendo el directorio del archivo creado anteriormente que contiene los usuarios y las contraseñas creadas. Todo usuario y contraseña que no se encuentre en este archivo no serán correctos.

Para concluir y los más importante, es habilitar esta línea, que por defecto en la versión Mosquitto 2.0 viene deshabilitada.

Ilustración 94. Configuración del servidor Mosquitto 2

La cual nos permite aplicar los cambios que se han realizado.