Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA, TITULACIONES DE INGENIERÍA INDUSTRIAL

MEMORIA TRABAJO DE FIN DE GRADO

Desarrollo de aplicación de Internet de las cosas usando protocolos LoRa y LoRaWAN

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Estudiante: Alejandro Santana Pérez

Tutor : D. Alberto Hamilton Castro

Fecha: 24/07/2023

Agradecimientos

Me gustaría agradecer a mi tutor Don Alberto Hamilton Castro por sus consejos y tutoría a la hora de desarrollar este trabajo de fin de grado.

A mi familia por el apoyo durante todo el transcurso del grado.
A Eduardo que me motivó a seguir adelante cuando más lo necesitaba.
A Lúa por acompañarme todos los días durante la mayoría del desarrollo.

Y por último a todos los profesores del grado y compañeros con los cuales no habría sido posible haber llegado hasta aquí.

"Alguna gente se lanzaba a la carga hacia su objetivo, corriendo con todas sus fuerzas.
Otros iban a trompicones.
Pero no era la velocidad lo que importaba.
Era la dirección en la que iban."
— Brandon Sanderson, El Ritmo de la Guerra

TABLA DE CONTENIDOS

TABLA DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. Resumen

El presente Trabajo de Fin de Grado se centra en el desarrollo de una aplicación IoT utilizando el protocolo LoRaWAN para la monitorización y riego automático de una planta. Para ello, se emplearon dos dispositivos TTGO LoRa32 como placas de desarrollo principales. La programación se basó en el ejemplo GitHub LMIC-node y se utilizó la plataforma IDE Platformio.

La aplicación logró recolectar datos de temperatura, humedad y nivel de agua, mediante la comunicación LoRa a través de The Things Stack. Estos datos se mostraron en una interfaz de usuario programada con Node-RED, un entorno de programación visual basado en flujos.

Para mejorar la precisión de las lecturas de los sensores, se aplicaron filtros. Además, se añadió un sistema de avisos a través de Telegram, permitiendo a los usuarios recibir alertas y actualizaciones.

Como resultado, la aplicación desarrollada permite la monitorización económica y efectiva del estado de la planta a través de Internet. Mediante la combinación de LoRaWAN, The Things Stack, Node-RED y la integración de Telegram, el proyecto logró una solución IoT eficaz y asequible. Además, la escalabilidad de esta aplicación favorece una fácil integración a un entorno industrial, adaptando las distintas necesidades del usuario o empresa con un surtido de sensores y dispositivos deseado.

Palabras clave: IoT, LoRaWAN, Riego Automático, TTGO LoRa32, LMIC-node, Platformio, The Things Stack, Node-RED

1.1 Abstract

The present Bachelor's thesis focuses on the development of an IoT application using the LoRaWAN protocol for the monitoring and automatic irrigation of a plant. The project used two TTGO LoRa32 development boards as the primary hardware components. Programming was based on the LMIC-node GitHub example, integrated with the Platformio IDE platform.

The application successfully collected data on temperature, humidity, and water level, using the LoRa communication provided by The Things Stack. The acquired data was displayed through a user interface created with Node-RED, a visual programming environment based on flow-based programming.

To enhance the accuracy of sensor readings, the implementation incorporated filter mechanisms. Additionally, a notification system was integrated using Telegram, allowing users to receive alerts and updates.

As a result, the developed application enables the cost-effective and efficient monitoring of the plant's status over the Internet. Through the combination of LoRaWAN, The Things Stack, Node-RED, and Telegram integration, the project achieved an effective and affordable IoT solution. Furthermore, the scalability of this application facilitates an easy integration into an industrial environment, adapting to the different needs of the user or company with a wide range of sensors and devices.

Key words: IoT, LoRaWAN, automatic irrigation, TTGO LoRa32, LMIC-node, Platformio, The Things Stack, Node-RED

2. Introducción

El Internet de las Cosas compone una red de dispositivos físicos capaces de transmitir información entre ellos y por lo tanto a usuarios a través de herramientas online. La disponibilidad de redes de Internet en la mayoría de la industria facilita la conexión de estos dispositivos entre ellos pero se ve necesario una forma de captar y transmitir esta información de forma eficiente y con un bajo coste de energía por lo tanto surgen protocolos como LoRaWAN. A partir de este tipo de soluciones se puede crear un enlace al Internet de las Cosas tanto en ambientes urbanos como rurales. Teniendo disponible en todo momento información de los distintos componentes se puede realizar una gestión de los recursos más sostenible y poder actuar ante adversidades con espontaneidad que sin embargo en una industria desconectada podría causar estragos irreparables. Para llevar a cabo esta conectividad constante se requiere de herramientas de desarrollo intuitivas que permitan implementarla no solo a las grandes industrias sino también a pequeños emprendedores.

2.1. Objetivos propuestos

El objetivo principal de este trabajo de fin de grado ha sido desarrollar una aplicación de Internet de las Cosas o *Internet of Things* (IoT) implementada a partir de protocolo LoRaWAN. Esta aplicación estará enfocada en la monitorización y control del riego de una planta teniendo principalmente atención en parámetros como la temperatura, humedad del suelo, volumen de agua disponible e intensidad de luz que serán mostrados a partir de una interfaz gráfica dinámica accesible desde un navegador web. A partir de esta interfaz el usuario podrá interactuar modificando distintos aspectos como el intervalo de tiempo entre la toma de datos de los distintos sensores implementados. Estos valores serán captados por una placa de desarrollo basada en ESP32, y como redundancia se tendrá otra placa midiendo tanto temperatura, humedad relativa como intensidad de luz. El segundo objetivo, intrínsicamente relacionado con el primero, ha sido familiarizarse con el protocolo de comunicación LoRaWAN y las herramientas relacionadas como The Things Stack, dispositivos nodo LoRa basados en ESP32 o, el protocolo de mensajería para el Internet de las Cosas MQTT. Por último pero no menos importante el tercer objetivo de este trabajo ha sido realizar un proyecto de elaboración propia relacionado con el Grado de Ingeniería Electrónica Industrial y Automática apoyándome en los conocimientos y herramientas adquiridas durante los estudios.

2.2. Alcance y limitaciones

El alcance de este trabajo es obtener un sistema robusto automatizado de bajo consumo que permita la implementación en la industria agricultora o en huertos urbanos, entornos en los que los formatos de transmisión inalámbrica más comunes como la telefonía móvil, Wi-Fi o Bluetooth muestran desventajas de coste energético y económico frente a la alternativa LoRaWAN. Asimismo se limita el proyecto a un ambiente experimental reducido a los recursos y tiempo disponibles.

2.3. Conocimientos previos

2.3.1 LoRa y LoRaWAN

LoRa, proveniente del inglés *Long Range*, es un tipo de modulación por radiofrecuencia caracterizado por su largo alcance y bajo consumo energético capaz de transmitir pequeños paquetes de datos en un entorno urbano hasta distancias de 5 km y 15 km en un área rural con línea visual directa de comunicación entre dispositivos. La modulación LoRa se basa en *Chirp Spread Spectrum (CSS),* en esta modulación los chirps, con una frecuencia central portadora, aumentan o disminuyen de forma constante su frecuencia haciendo uso de todo el ancho de banda disponible, en Europa este es restringido por estándares internacionales entre 125 kHz y 250 kHz sobre las bandas ISM EU433 y EU863.

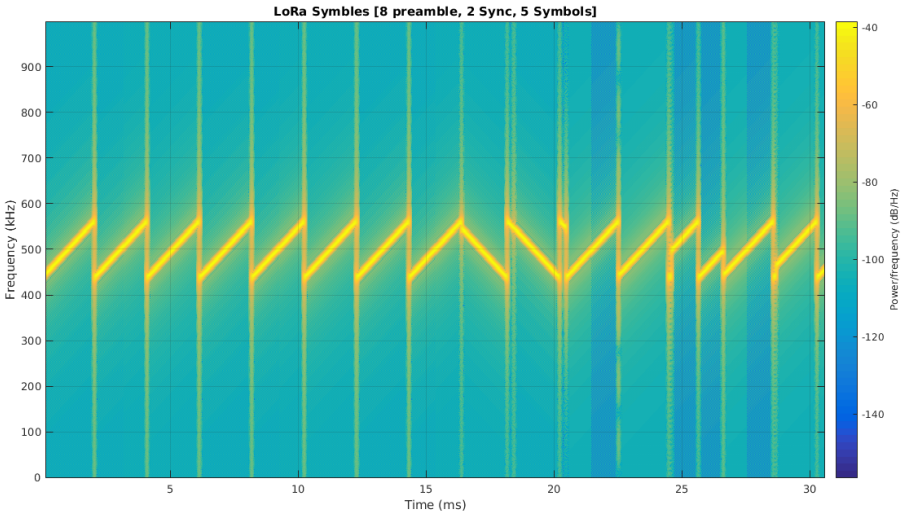

Ilustración 1 Visualización de espectrograma de chirps [3]

Para transmitir la información en modulación LoRa se emplea un conjunto de símbolos llamados up-chirps en los que aumenta la frecuencia y down-chirps en los que disminuye. A efectos de sincronización, los mensajes LoRa comenzarán siempre con un encabezado obligatorio de 8 símbolos

seguidos de un número de símbolos para indicar distintos aspectos de la señal según sea de tipo implícito o explícito. Tras este encabezado se transmitirá el mensaje llamado Payload. La tasa de envío de los chirps es definida por el factor de dispersión, un factor de dispersión menor indica chirps más rápidos y cortos y posibilita más información. En la modulación LoRa hay un total de 6 factores de dispersión posibles desde SF7 a SF12. Al afectar a la tasa de envío influyen también en la tasa de datos, el tiempo en aire, consumo energético y sensibilidad del recibidor. Como afecta a la sensibilidad del recibidor un factor mayor permitirá un radio de mayor alcance. Un aspecto importante de la modulación LoRa es que es capaz de demodular señales -20 dB por debajo del nivel de ruido permitiendo una alta penetración de la señal en entornos urbanos donde hay una gran cantidad de edificaciones.

LoRaWAN es un protocolo de red *Low Power, Wide Area* (LPWA) que establece una comunicación bidireccional entre dispositivos nodo LoRa y aplicaciones de IoT. Desarrollado principalmente por LoRa Alliance de forma abierta. Existen cinco elementos principales en una red LoRaWAN:

- Los nodos LoRa, generalmente dispuestos con sensores para captar y digitalizar distintos valores del ambiente de forma autónoma mediante baterías. Estos se encargan de generar los mensajes modulados mediante LoRa. Estos nodos pertenecen a la capa física del modelo OSI.
- Gateways, reciben los mensajes modulados enviados por los nodos y distribuyen los paquetes a los servidores de red LoRaWAN (LNS) correspondientes mediante Wi-Fi, cableado Ethernet o conexión celular. Los paquetes enviados por los nodos son recibidos por todos los dispositivos Gateway por lo tanto los servidores se encargan de hacer una eliminación de datos redundantes manteniendo solo una copia recibida por los distintos Gateway.

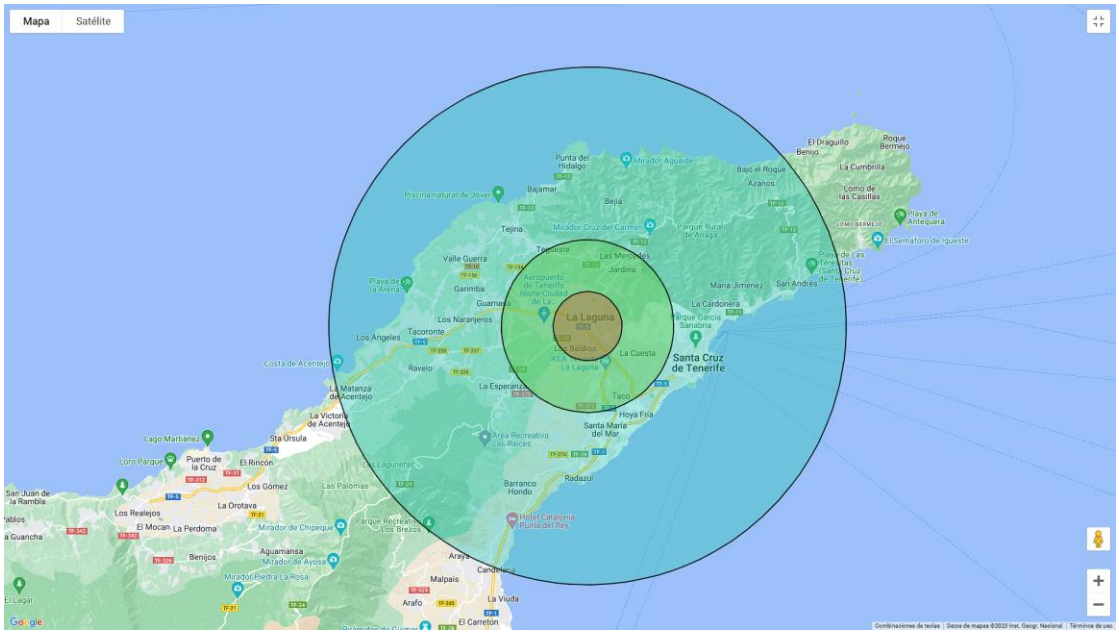

Ilustración 2 Radio teórico del Gateway en la Torre Profesor Agustín Arévalo

- Servidor Red LoraWAN, establece una conexión segura y dinámica para transportar los datos entre dispositivos LoRa y una aplicación de usuario en la nube. También controla el tráfico en esta red y asegura la integridad de los mensajes.
- Servidor de Aplicación, se encarga de interpretar los mensajes recibidos y generar mensajes de envío para los nodos.
- Servidor de Conexión, contiene la información para la activación de nodos LoRa mediante distintas credenciales y claves de acceso almacenadas tanto en los dispositivos finales como en el servidor. Durante el periodo de activación del nodo el servidor comunica tanto al servidor de Aplicación como al nodo un código de sesión para establecer la conexión correcta entre ambos.

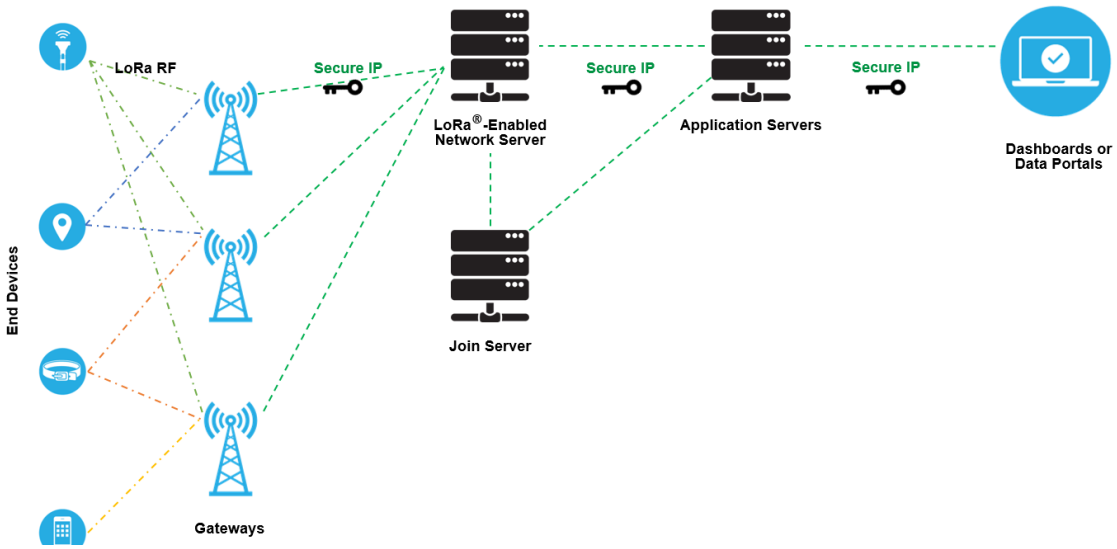

Ilustración 3 Esquema de un ejemplo de red LoRaWAN [2]

Los dispositivos nodo LoRa pueden operar en tres clases distintas dependiendo del tipo de comunicación que interese:

- Clase A: El dispositivo permanece en un estado de reposo hasta que se produce un cambio en el entorno o se haya programado tras un periodo para hacer una lectura de los sensores. Inician la comunicación LoRa y permiten una pequeña ventana de tiempo para recibir paquetes *downlink.* No es posible despertar los dispositivos de Clase A del reposo para recibir comunicación por lo tanto no son indicados para actuadores.
- Clase B: Este modo de funcionamiento ofrece ventanas programadas para recibir mensajes de la red para ello un proceso de sincronización por baliza, o *beacon* en inglés, tiene que ser transmitido mediante los Gateways de forma periódica a los nodos para alinear el tiempo interno de estos con la red.

- Clase C: Siempre encendidos, están constantemente pendientes de recibir un mensaje de la red y no dependen de baterías externas.

2.3.2 The Things Stack

The Things Stack es un LNS *open-source,* desarrollado y mantenido por The Things Industries, con distintas funcionalidades e integraciones para implementar una solución IoT. Establece estándares de uso de su red como la limitación del número de mensajes enviados y recibidos posibles por nodos según el factor de dispersión garantizando así un uso justo de esta. La red se forma gracias a más de 21.000 dispositivos gateway alrededor del mundo junto a los más de 190.000 miembros llegando a formar parte de 153 países.

Esta plataforma y conjunto de herramientas ideal para la gestión de redes LoRaWAN. Permite la creación de aplicaciones en los que se registran los dispositivos LoRa y la gestión de los datos recibidos. También permite el registro y gestión de Gateway que son los dispositivos encargados de comunicarse con los nodos y transmitir los datos a la red. Ofrece encriptado y desencriptado de mensajes y otras funcionalidades para tener una aplicación segura.

Además de ser una herramienta muy útil para implementar una solución de IoT a partir de LoRaWAN, incluye una gran cantidad de documentación y soporte técnico gratuito básico. Su versión Community es gratuita e incluye todas las funciones pero con limitaciones y está indicada para el desarrollo de soluciones LoRaWAN con objetivos educacionales. The Things Stack Cloud tiene un fin comercial el cual se puede comenzar a usar de forma gratuita y posteriormente obtener un plan de suscripción.

The Things Stack incorpora una amplia variedad de Integraciones y APIs que permiten la interacción con otras plataformas y sistemas, las más relevantes son MQTT, base de datos incorporada, AWS IoT, Azure IoT, LoRa Cloud y ThingSpeak.

2.3.3 MQTT

MQTT (Message Queuing Telemetry Transport) es un protocolo de mensajería para la comunicación de los paquetes entre dispositivos inteligentes y sensores o actuadores del Internet de las Cosas. A partir de un bróker instalado en una maquina o servidor en la nube que actúa como intermediario entre publicadores de mensajes y subscriptores, se distribuyen los mensajes de forma eficiente y ligera.

El modelo publicación/suscripción permite una comunicación liviana aprovechando los recursos limitados y minimizando la sobrecarga de la red con una comunicación eficiente. Los mensajes se publican en canales llamados tópicos permitiendo crear agrupaciones en los que juntar o separar para una mejor gestión. Admite tres niveles de calidad de servicio (QoS) dependiendo de los requisitos de la aplicación. Con QoS 0 no se garantiza la entrega de los mensajes ya que no hay intento de retransmisión ni confirmación de la entrega.

QoS 1 se envía el mensaje hasta obtener al menos una confirmación de entrega, esto puede llegar a causar duplicados. QoS 2 garantiza que el mensaje es entregado al mismo tiempo que no se produzcan duplicados llevando un seguimiento del estado de los mensajes pero congestiona más la red y puede ocasionar retrasos en la comunicación.

2.3.4 Node-RED

Node-RED es una herramienta gratuita para el desarrollo basado en JavaScript a partir de programación visual enfocada al enlace entre dispositivos *hardware,* APIs y servicios online. Node-RED habilita un editor de flujos a partir de un entorno en navegador web para realizar las conexiones entre dispositivos y aplicaciones. Node-RED se apoya en una interfaz basada en nodos y cables donde los nodos representan componentes y los cables el flujo de información por la que se comunican los nodos. Es una herramienta de gran utilidad para el IoT ya que permite conectar dispositivos entre ellos de una forma visual e intuitiva. Con Node-RED se puede automatizar tareas y sistemas de forma escalable. Se conforma de una interfaz con nodos agrupados en lo que se llaman paletas según el tipo de función. Estos nodos pueden tener una entrada, varias o ninguna y se seleccionan y arrastran al área de programación. Una vez en el área de programación se modifica o se configura según el objetivo deseado. Si es posible enlazar con otro nodo se crea una unión a partir de un punto visible a otro nodo.

3. Herramientas y componentes

Para diseñar y prototipar ambos nodos junto a sus sensores he requerido tanto de herramientas físicas como software para la programación.

Se ha utilizado el Multímetro Digital MUL0001 DT850L para comprobar el funcionamiento tanto de los sensores como las salidas y entradas de los pines. Un dispositivo Android para obtener información a través de puerto serial de los nodos mientras se ejecutaba su código conectado a partir de un cable USB.

En lo relativo al Software el entorno de programación Arduino IDE para la prueba de scripts sencillos y ejemplos de las distintas librerías para el testeo tanto de los sensores como funcionalidades de la comunicación LoRa. Por otro lado el entorno de programación Platformio para el diseño del programa final de ambos nodos a partir del repositorio GitHub LMIC-node.
Para la monitorización mediante puerto serial de los dispositivos LoRa en funcionamiento he utilizado la aplicación Android Serial USB Terminal.

3.1 Dispositivo nodo final para valores de temperatura, aire y luz medioambiental.

Para el primer nodo se ha utilizado la placa de desarrollo TTGO LoRa32 V2. Esta placa trabaja con la arquitectura ESP32 de Espressif que permite leer por entrada analógica desde 0 a 4095 valores de tensión entre 0 – 3.3 V aunque pierde la linealidad con valores por debajo de 0.1V y por encima de 3.1 V. A continuación se muestran especificaciones relevantes:

- Alimentación: 1.8-3.7V
- Frecuencia de operación (Europa): 868 MHz
- Memoria Flash: 4 MB
- Temperaturas de operación: -40ºC +85ºC
- Corriente pines: 12mA máx.
- Corriente en reposo: 1.5 µA
- SoC: ESP32 Pico D4
- Chip Modulación LoRa: SX1276
- Resolución analógica: 12 bits o 4095 puntos
- Valores de tensión GPIO: 0 – 3.3 V
- Puerto microUSB
- Botón de reinicio
- Antena

Esta placa dispone de 31 pines de los cuales 9 son utilizados para la comunicación LoRa, 4 son utilizados por defecto para comunicación I2C y

UART, 2 pines GND, 1 pin 3.3 V, 1 pin 5V, 9 pines GPIO con ADC, y 5 pines GPIO en los que solo se permite entrada. (Ilustración 4)

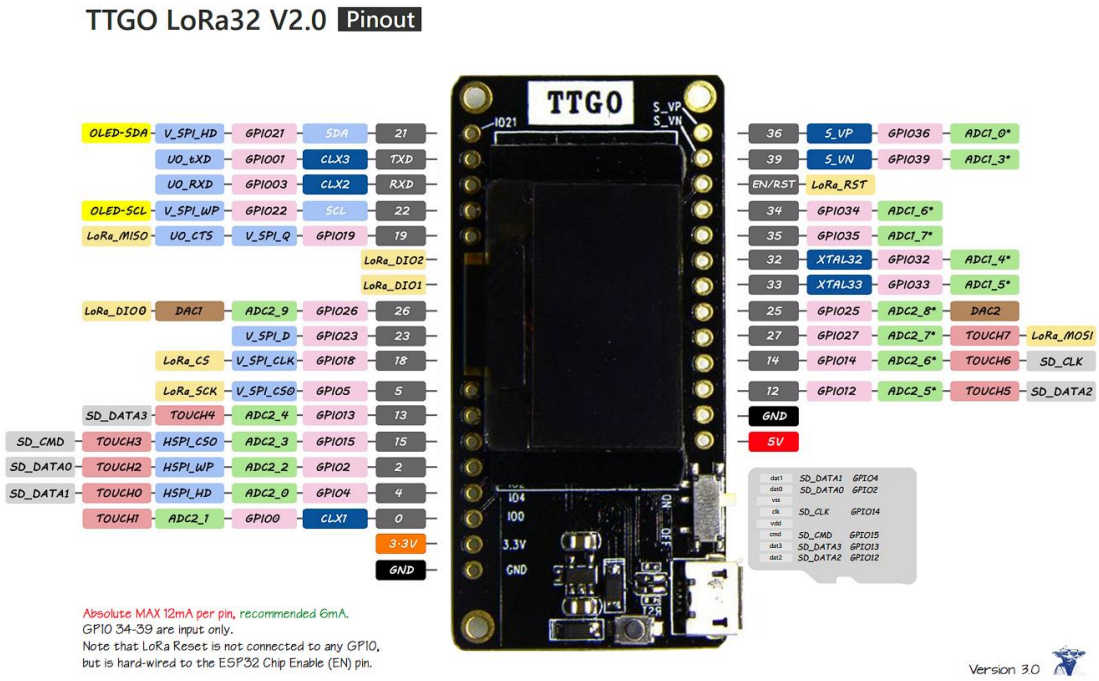

Ilustración 4 Esquema de pines TTGO LoRa32 V2.0 [5]

Para la lectura de la temperatura y humedad relativa se ha utilizado un sensor DHT11 conectado al pin GPIO4 y alimentado por el pin GPIO13. Este sensor digital precalibrado de fábrica tiene un rango de temperatura ambiente de entre 0 y 50ºC y de humedad relativa entre 20 a 90% y opera entre 3 a 5.5V DC. Es una opción bastante económica idónea para aplicaciones IoT.

Tabla 1 Carácterísticas DHT11 [9]

Rango de Temperatura	0 a 50 ºC ± 2 ºC
Rango de Humedad	20 a 90% ± 5%
Resolución	Humedad: 1% Temperatura: 1 ºC
Voltaje de operación	3 – 5.5 V DC
Corriente de alimentación	0.5 – 2.5 mA
Tiempo de muestreo	1 segundo
Precio	2 – 3 €

Este sensor viene incorporado en un módulo con una resistencia pull-up de 10 kΩ entre el pin de salida digital y la alimentación.

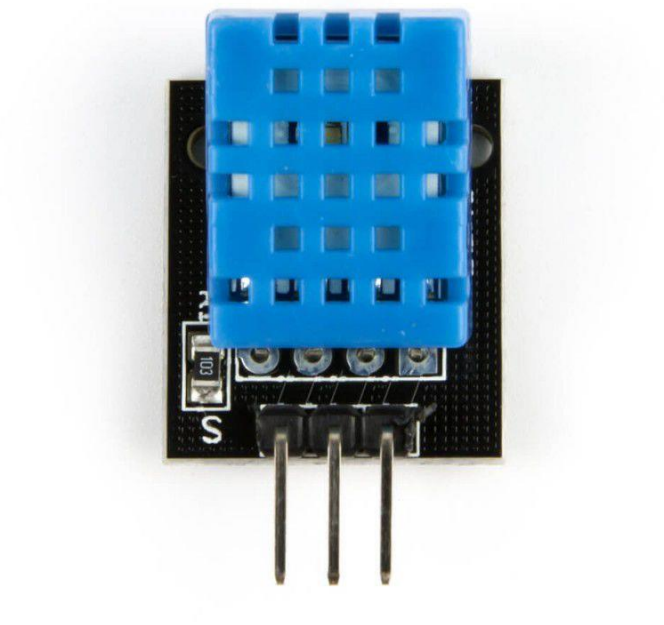

Ilustración 5 Módulo con sensor DHT11 [9]

La intensidad de luz se obtiene a partir del sensor BH1750. Este sensor digital utiliza el protocolo I2C para obtener los valores de luminancia en lux. Precalibrado de fábrica realiza la conversión de forma interna y se obtiene un valor entre 0 y 65535 lux con una velocidad de respuesta aproximada a la del ojo humano. El sensor permite tanto la lectura continua como instantánea ambas con 3 modos de resolución. La aplicación de este sensor permitirá tener un control de la cantidad de luz ambiental en distintos momentos del día.

Tabla 2 Características BH1750 [12]

Rango de luminancia	1 a 65535 lux ± 20%
Temperatura de operación	-40 +85 ºC
Resolución	Modo Baja 4 lux Modo Alta 1 lux Modo Alta 2 0.5 lux
Tiempo de lectura	Modo Baja 16 ms Modo Alta 120 ms Modo Alta 2 120 ms
Voltaje de operación	2 - 3.6 V
Corriente de alimentación	0.12 – 0.19 mA
Precio	3 – 6 €

La conexión de este sensor se ha hecho en los pines I2C 21 y 22 compartiendo alimentación del pin GPIO13 con el sensor DHT11.

Ilustración 6 Sensor BH1750 [10]

Para la conexión de estos sensores con la placa de desarrollo se ha utilizado cables Dupont y una protoboard de 400 pines. Además se ha alimentado con una batería de recargable Li-Po 3.7 V de 600 mAh.

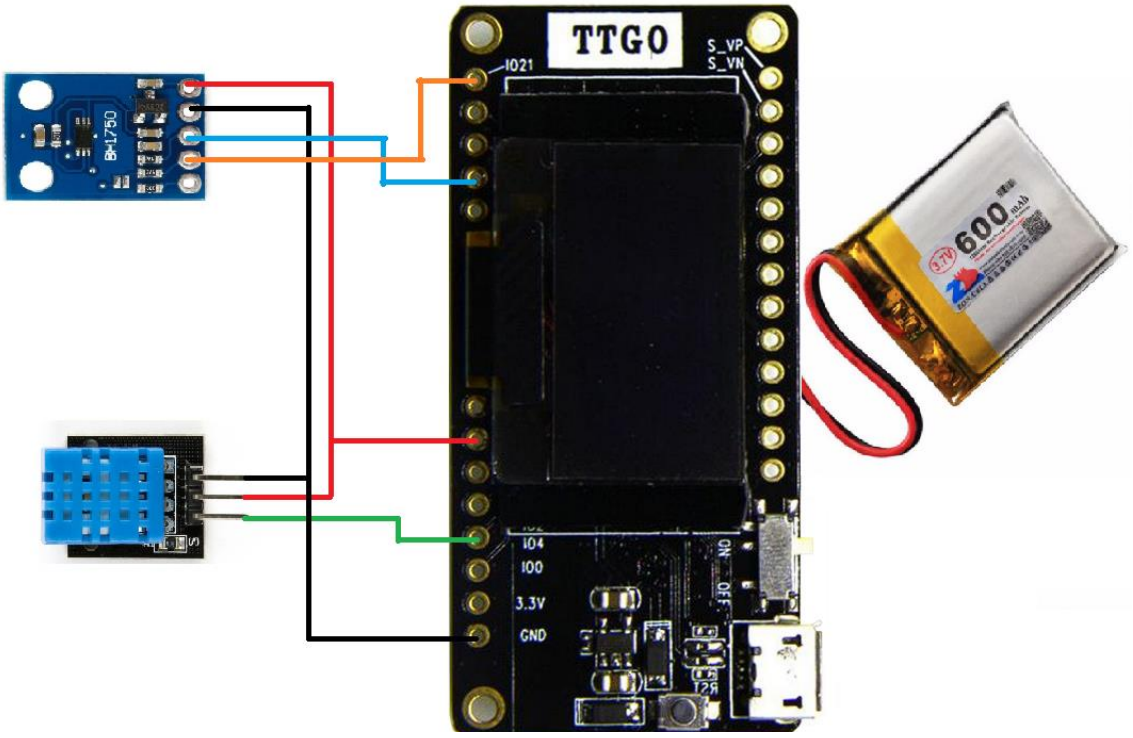

Ilustración 7 Diagrama de conexión Nodo 1

La integración de la placa para la puesta en marcha se ha realizado en un recipiente plástico con orificios en la parte superior, a través de los cuales se ha ubicado tanto ambos sensores como la antena incorporada. Esto permitirá proteger al sistema de elementos no deseados como polvo, tierra o incluso agua aunque no llega a ser del todo impermeable. Para la máxima protección

se debería diseñar una caseta para estación meteorológica o de funcionamiento similar que no abarca los objetivos de este proyecto. A partir de ahora se mencionará como Nodo 1.

3.2. Dispositivo nodo final para monitorizar y automatizar el riego de una planta.

La placa utilizada coincide con el Nodo 1 con el mismo conjunto de características. El objetivo de esta placa será tomar los valores analógicos de distintos sensores y procesarlos a partir de una parametrización para poder realizar el riego de una planta tras superar ciertos límites establecidos. El conjunto de sensores y actuadores junto a la placa se procederá a llamar Nodo 2 durante la longitud de este texto.

Se han incorporado a esta placa 4 sensores analógicos para medir las siguientes características:

Tabla 2 Sensores Nodo 2

Sensor	Mide	Objetivo
Módulo Fotorresistor	Cambios en luz ambiental	Captar los distintos niveles de luz a lo largo del tiempo y comparar con el sensor calibrado BH1750.
Módulo Termistor	Cambios en la temperatura	Captar los cambios de temperatura en el entorno cercano a la planta y comparar con el sensor digital DHT11.
Sensor de agua	Contacto con agua	Medir que el nivel de agua en un depósito no disminuye del límite permitido para poder activar una bomba de agua sin daños.
Sensor de humedad del suelo	Cambios en la conductividad del suelo	Monitorizar la humedad en el suelo activando un relé para alimentar una bomba de agua al superar el valor consignado.

Se ha decidido por añadir tanto un módulo fotorresistor como termistor para tener cierta redundancia con los valores captados por el Nodo 1, además de realizar una calibración y parametrización de estos sensores para poder interpretar los valores analógicos medidos en unidades del sistema internacional lux y grados ºC.

El termistor, incluido en un módulo con una resistencia pull-up de 10 kΩ, permite medir cambios de tensión en la patilla analógica entre 0 y 3.3 V con los que ha sido alimentado. Es de tipo NTC por lo tanto su resistencia interna es inversamente proporcional al aumento de la temperatura, proporcionando una mayor tensión y por lo tanto valor analógico a la salida mientras mayor sea la

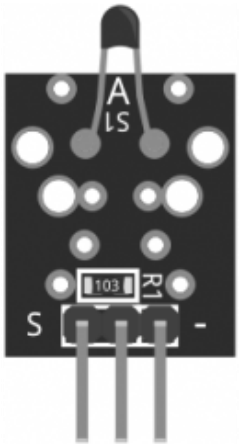

Ilustración 8 Módulo termistor KY-013 NTC [13]

temperatura. Permite un rango de medida entre -55 ºC y 125 ºC con una precisión de ±0.5 ºC.

El patillaje de este módulo está indicado erróneamente habiendo analizado a partir de un multímetro los distintos valores de resistencia y voltaje siendo S y la patilla central los pines para alimentación y tierra. La salida analógica se obtiene del pin etiquetado como "-"(Ilustr. 11). El pin analógico se conectará al GPIO15 de la placa.

De misma manera el módulo del fotorresistor LDR también incorpora una resistencia de 10 kΩ con el patillaje coincidiendo con el del termistor. El valor de la resistencia del fotorresistor disminuye con el aumento de intensidad de luz y también operará entre 0 y 3.3 V. El pin analógico se conectará al GPIO13 de la placa.

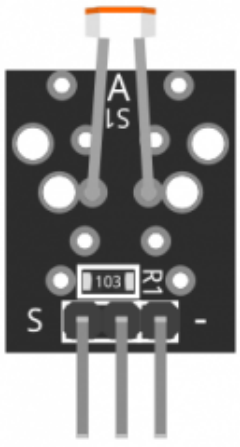

Ilustración 9 Módulo fotorresistor KY-018 [14]

El sensor de humedad del suelo se construyó a partir de dos varas de grafito conectadas a un módulo que incluye un comparador LM393 con entrada ajustable mediante potenciómetro, led verde para la alimentación, y led verde para la salida del comparador. El módulo al que se conectan las varas tiene 4 pines dos para la alimentación, 1 para la salida analógica y 1 para la salida digital del comparador. Se utilizan barras de grafito ya que originalmente se

producía corrosión por efecto galvánico y rápidamente perdía utilidad. El suelo húmedo presenta mayor conductividad lo que disminuye la resistividad y por lo tanto el valor analógico leído. El pin analógico se conectará al GPIO2 de la placa.

Ilustración 10 Varas de grafito para sensor de humedad del suelo

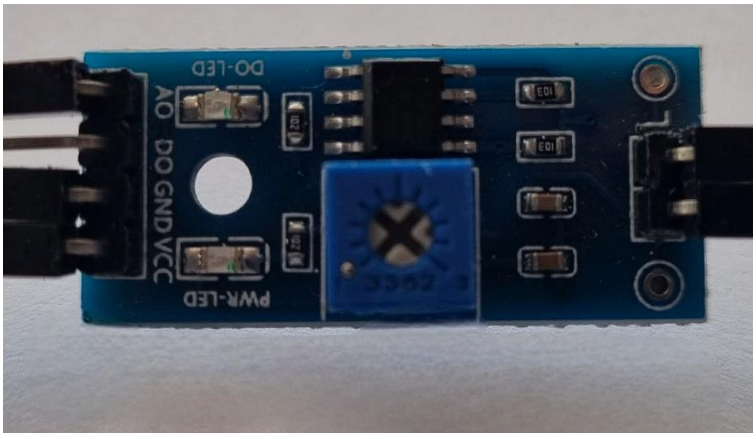

Ilustración 11 Módulo con comparador para medición de humedad del suelo

Por último en sensor de agua utilizado ha sido de tipo resistivo. Aunque es posible medir diferencias en el nivel del agua este tipo de sensor es susceptible a cambios en la temperatura, niveles de pH, cambios en la composición etc. Sigue un comportamiento similar a los anteriores, mientras mayor contacto tiene el agua con las trazas de cobre menor será su resistividad. Este sensor tiene la desventaja de necesitar un entorno acuático entonces para disminuir la corrosión solo se alimenta al sensor durante la toma de medida. Tiene 3 pines dos para la alimentación y el tercero para el valor analógico. Dispone de un led rojo activado por la alimentación. El pin analógico se conectará al GPIO4 de la placa.

Ilustración 10 Módulo sensor nivel de agua

Este sensor se utilizará para medir la ausencia de agua por encima de un nivel del depósito para indicar que es necesario rellenar de agua para poder proceder al riego.

El conjunto de los 4 sensores anteriores es alimentado por el pin GPIO12 durante el momento de la medida.

Como actuador se hará uso de un relé de 5V DC para poder alimentar a una bomba de agua con una pila de 4.5 V. La señal para activar este relé será enviada por el pin GPIO25 y la alimentación del relé será desde la salida de 5V de la placa. La bomba de agua opera entre 3 - 5 V DC con valores de corriente

entre 100 -200 mA capaz de ejercer un caudal mínimo de 72 l/h y 110 l/h máximo.

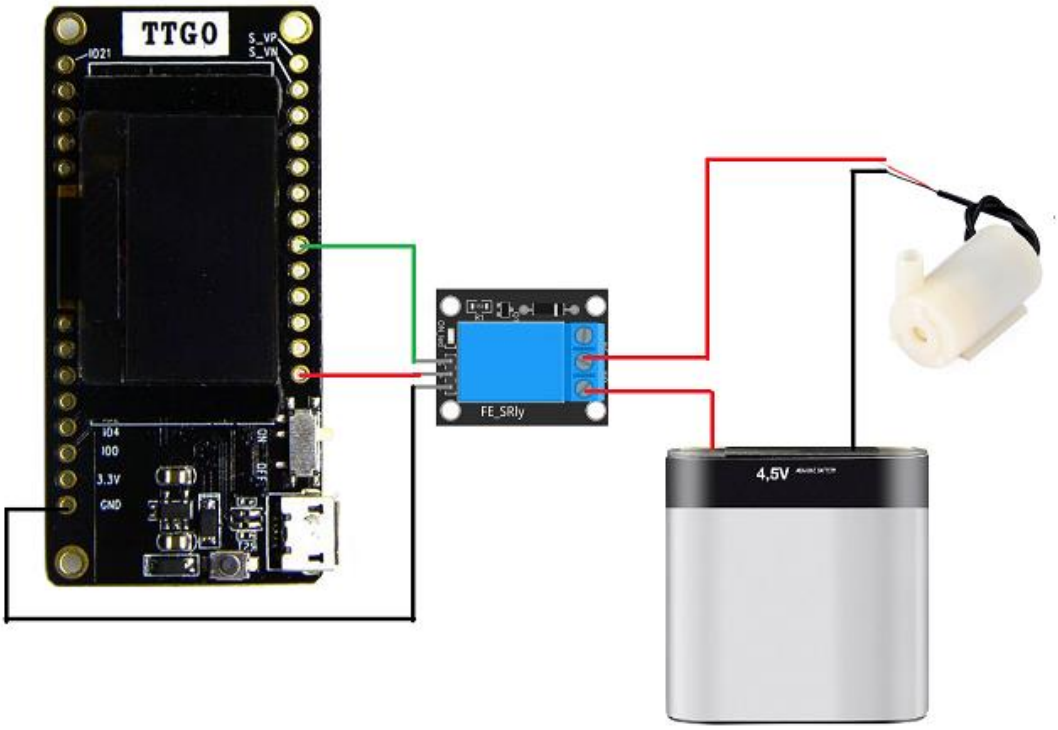

Ilustración 11 Esquema de conexión bomba de agua con relé, batería y placa

El Nodo 2 es alimentado por una batería Li-Po de 3.7V y 1200 mAh conectada a un módulo de recarga TP4056 con conexión micro USB y conectado a un panel solar de 11x8 cm 5V DC 1.25 W. El módulo TP4056 permite cambiar la corriente de carga de la batería y, por lo tanto, la capacidad de esta a partir de una resistencia Rprog que por defecto viene incluida de 1.2 kΩ. Incluye protecciones para cortocircuito, polaridad inversa, sobrecarga y sobre descarga, y leds de estado de carga. Aunque no es recomendado utilizar el cargador conectado a una carga constante, al tener periodos donde no recibe carga no surge ningún problema pero se aconseja el uso de un MOSFET, diodo y resistencia para evitar sobrecarga durante el uso.

A los terminales IN+ y IN- se conecta el panel solar, en B+ y B- los terminales de la batería y OUT+ y OUT- a la placa de desarrollo (Ilust. 12). Conectamos un interruptor en serie con la placa y la salida OUT+ para poder apagar y encender de forma cómoda la placa ya que estará encapsulada en un recipiente de plástico para su protección.

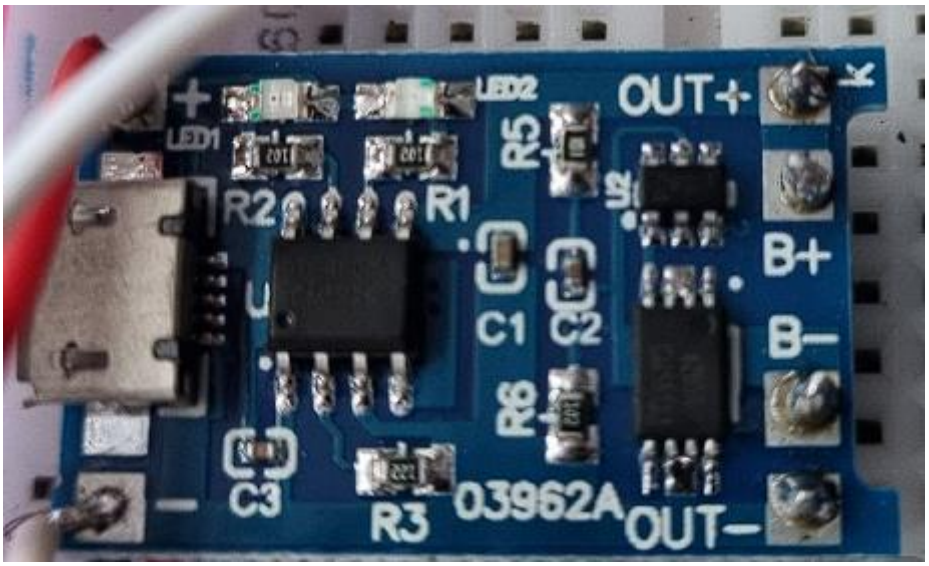

Ilustración 12 Módulo TP4056 para la recarga de baterías Li-Po

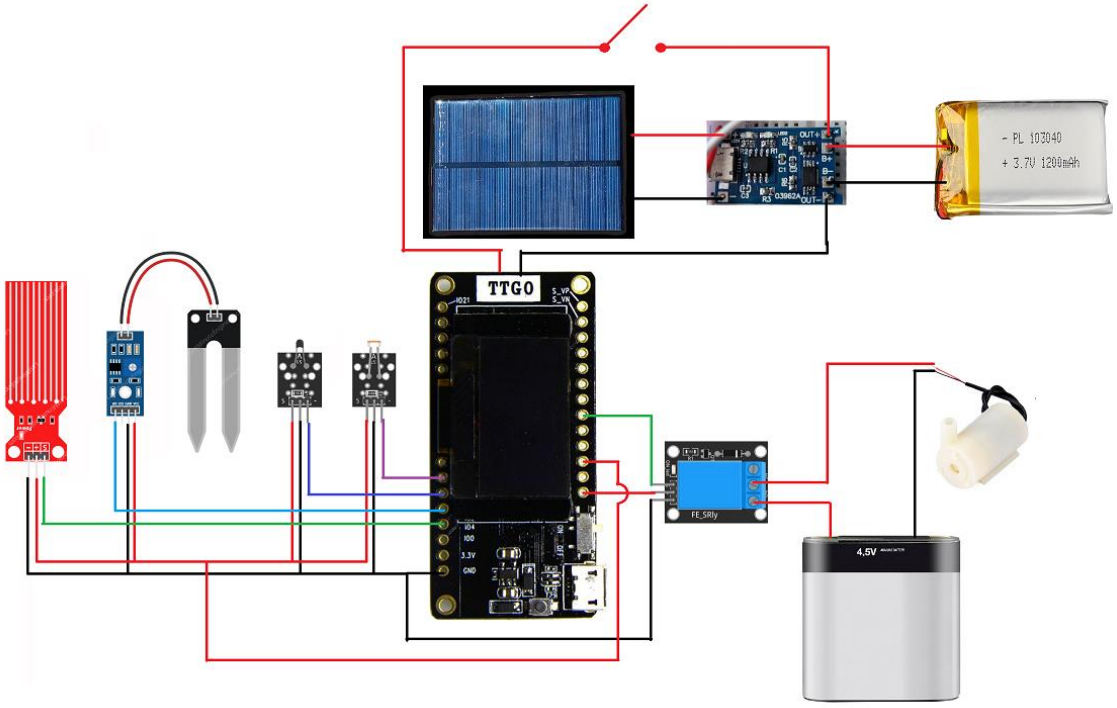

Ilustración 13 Esquema de conexión Nodo 2

El sensor de agua estará pegado a una de las paredes de un recipiente de 1.25 l donde se depositará la bomba de agua. Tanto la placa de desarrollo como el módulo fotorresistor, el módulo de recarga, el relé y la batería de litio estarán en un recipiente plástico para su protección de polvo y otros elementos. El termistor para poder hacer una lectura de la temperatura se situará fuera de un orificio de este recipiente por el que también saldrá la conexión para el módulo del sensor de humedad del suelo cuyas varas de grafito estarán enterradas en el suelo de una maceta. El panel solar se encontrará en la parte

superior del recipiente y la pila que alimenta a la batería entre el depósito de agua y el recipiente plástico (Ilustr. 14).

Ilustración 14 Nodo 2 en funcionamiento con depósito de agua y maceta para la medida de los parámetros.

3.3. Software utilizado

El desarrollo de la aplicación, la programación y diferentes pruebas se ha realizado a partir de un ordenador con el sistema operativo Windows 11 y varios dispositivos Android.

Arduino IDE es el entorno de programación multiplataforma utilizado en este trabajo para la parametrización de los distintos sensores analógicos y pruebas simples de funcionamiento de los sensores y comunicación LoRa. Se instala junto a los drivers de la placa ESP32 y librerías necesarias. Permite seleccionar la placa desarrollo de una base de datos y elegir ejemplos asociados a ella para familiarizarse con los distintos entornos tanto Arduino como ESP32. Incluye una visualización por serial de valores analógicos de forma gráfica para esto se debe tener en cuenta la nomenclatura de las distintas variables y cómo se expresan en el código. También permite comunicarse por serial con las distintas placas de desarrollo a través de un terminal. A partir del gestor de placas de desarrollo podemos instalar tanto drivers como librerías asociadas a las placas. También incluye un gestor de librerías para descargar e incluir en nuestros programas aquellas que necesitemos y estén incluidas en su base de datos. Dispone de una

herramienta para depurar el código y una opción para mantener en la nube los proyectos. Es una herramienta con bastante facilidad al uso e intuitiva, con bastante soporte por la comunidad siendo posible encontrar bastante documentación e información que permite desarrollar todo tipo de proyectos. El lenguaje de programación se enfoca principalmente en C++. En el momento de elegir una placa es necesario especificar el puerto al que está conectada, en este proyecto la conexión se ha hecho a partir de un puerto COM USB, después se seleccionan por defecto la velocidad de subida a la placa y la frecuencia de flash.

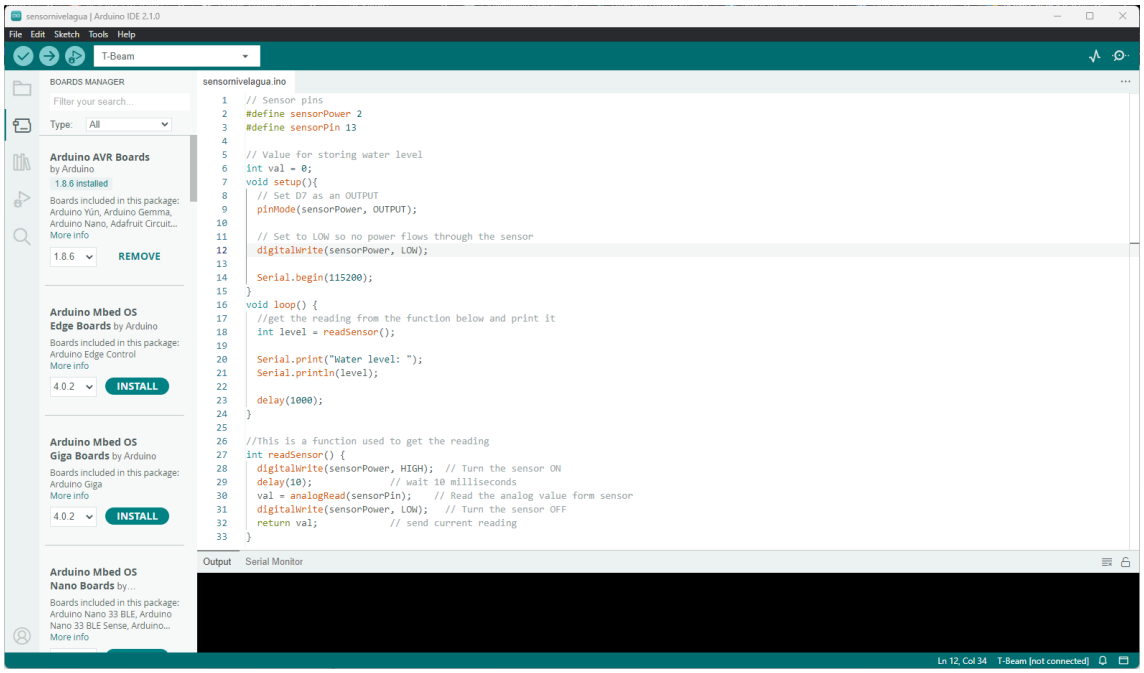

Ilustración 15 Arduino IDE con un ejemplo para mostrar el nivel de agua de un sensor analógico

El programa final que se ha incorporado a ambos nodos basado en el proyecto GitHub LMIC-node se ha realizado en Platformio desde Visual Studio Code. Platformio que también es un entorno de programación permite ser integrado junto a otros entornos de programación como Visual Studio Code, en este caso, lo cual capacita el aprovechamiento de todas las utilidades y herramientas que aportan ambos entornos en un solo lugar. Incluye un mayor rango de placas de desarrollo y una mejor gestión de las bibliotecas incluyendo la posibilidad de añadir bibliotecas de terceros. Otra cualidad es la integración con sistemas de control de versiones como puede ser GitHub para la colaboración en proyectos más complejos. A diferencia de Arduino IDE, Platformio soporta una mayor variedad de lenguajes aparte de C++, incluyendo Python, JavaScript, C# y más. Al ser un programa más extenso no cuenta con una comunidad tan enfocada como Arduino IDE pero no significa que tenga menos soporte. Platformio en Visual Studio Code permite tener un espacio de trabajo con una visualización de todos los archivos en árbol siendo más sencillo desplazarse entre documentos para un proyecto más complejo.

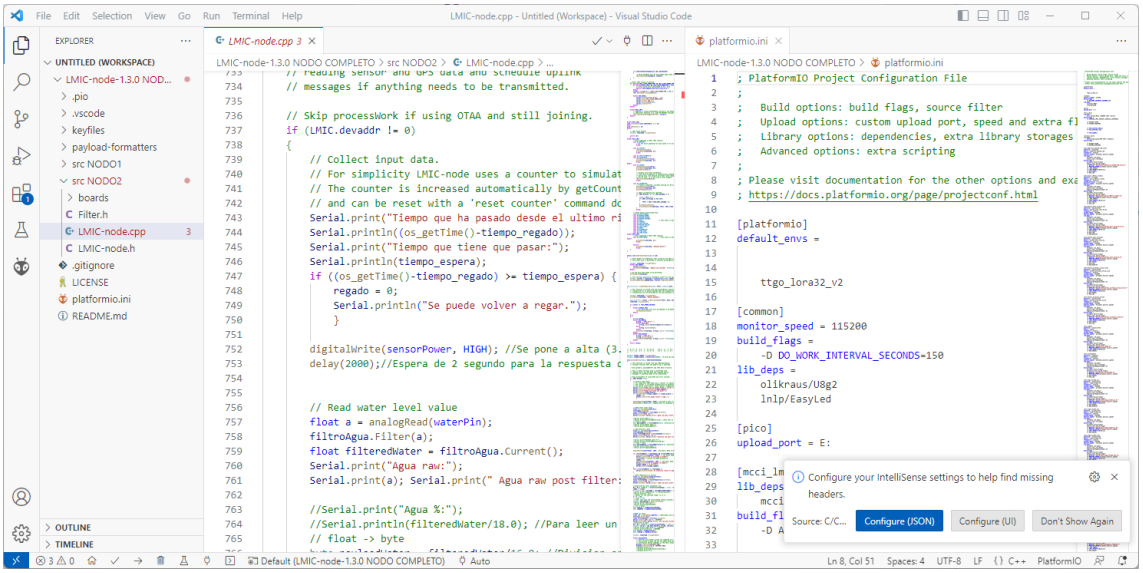

Ilustración 16 Interfaz de Platformio en Visual Studio Code con código del programa final del Nodo 2

La red LoRaWAN utilizada es la proporcionada por The Things Stack incluyendo sus integraciones con MQTT y Cayenne. Para la interfaz visual y gestión de los datos obtenidos por los nodos se ha instalado una instancia virtual de servidor Ubuntu 20.04 con dominio público. En este servidor se ha instalado la herramienta Node-RED para obtener los datos desde MQTT y mostrarlos en un dashboard.

La comunicación de los nodos con Node-RED se realiza a partir de una aplicación creada en el portal de The Thing Stack. En este portal también es posible registrar y modificar los Gateway instalados pero no ha sido el ámbito de este proyecto. Desde la pantalla principal se puede acceder al menú aplicación y observar datos relevantes de tus dispositivos registrados.

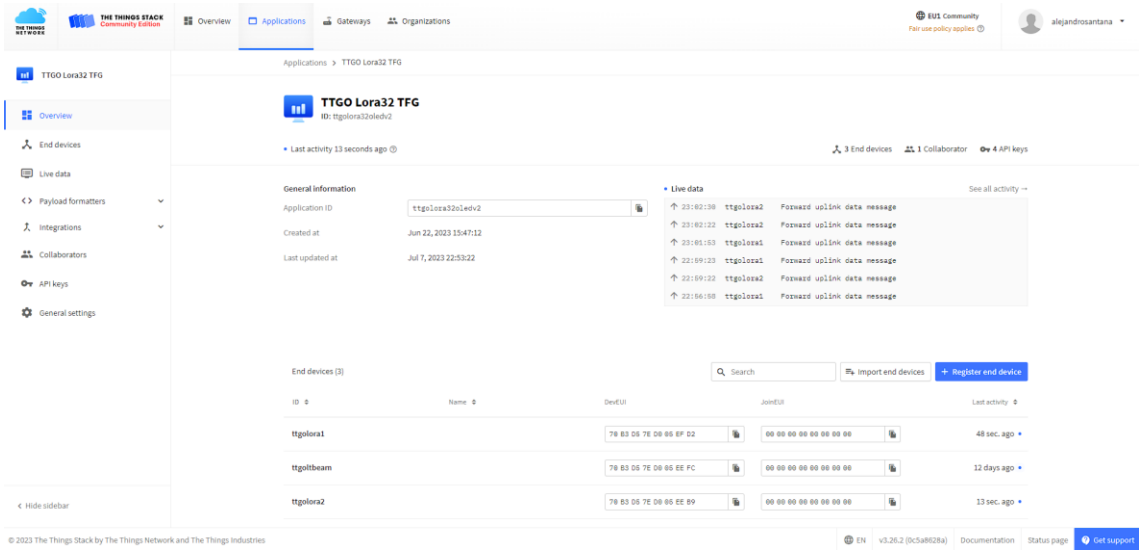

Ilustración 17 Página principal de la aplicación para este proyecto con información de los nodos registrados.

Con el menú se puede acceder a todas las integraciones incluidas o que queramos añadir, añadir, eliminar o modificar los nodos o dispositivos, mostrar la información recibida y enviada en tiempo real y otras opciones.

Recibiremos los mensajes de The Things Stack a partir del cliente de programación web Node-RED. Por defecto incluye varias paletas con nodos básicos, pero para este proyecto se ha añadido la paleta de *moment*, *dashboard* y *telegrambot*. Con la paleta básica podemos realizar la integración MQTT junto a The Things Stack y empezar a recibir mensajes del dispositivo o dispositivos que nos interese. Con la paleta *moment* podemos crear nodos para formatear y mostrar fechas y tiempo de una forma más sencilla. Con la paleta dashboard es posible crear una interfaz visual interactiva con distintos gráficos, información y entrada de usuario. Esta interfaz será accesible desde cualquier lado si se dispone de un dominio público como es el caso. Por último con la paleta telegrambot podemos crear avisos que llegarán al dispositivo móvil a través de la aplicación de mensajería Telegram. Node-RED es una herramienta muy versátil para implementar una solución de IoT que requiere una gran inversión de tiempo para poder explotar todo su potencial debido a la gran cantidad de opciones disponibles. Para poder empezar a programar en esta herramienta primero es necesario instalarla en un servidor o máquina. En la documentación hay información para poner en marcha el cliente tanto en un dispositivo local, Raspberry Pi, Android, Docker y entornos en la nube como AWS o Microsoft Azure. En este proyecto se ha decantado por utilizar Microsoft Azure ya que permite el uso gratuito de la plataforma y un crédito inicial durante los 30 primeros días. Una vez instalado y configurado se puede acceder de manera local o a través de internet desde un navegador Web donde encontraremos la pantalla principal para empezar a desarrollar el flujo.

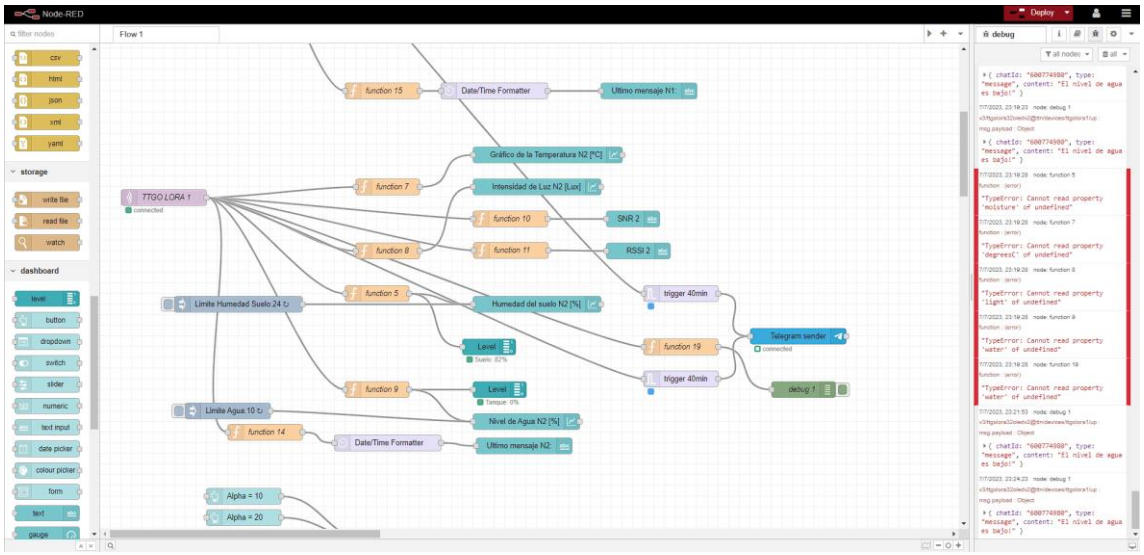

Ilustración 18 Ventana principal con parte del código utilizado para el proyecto.

A la izquierda de la pantalla principal se encuentran las paletas con los nodos disponibles, en el centro los nodos y los flujos programados, y a la derecha distintas pestañas con documentación del uso de los nodos, menú de depuración y otras opciones.

La aplicación Serial USB Terminal al conectar los nodos al dispositivo Android mediante cable USB y seleccionar la tasa de baudios correspondiente, permite mostrar los mensajes que hemos indicado en el código para confirmar el correcto funcionamiento o a modo informativo del código realizado. Un ejemplo del uso de esta aplicación se muestra en el capítulo 4.4.

3.3.1 Librerías

En la programación de los nodos se ha utilizado las siguientes librerías o repositorios:

- lnlp/LMIC-node es un repositorio con un ejemplo de aplicación LoRaWAN que incluye la función básica de envío de una cuenta a The Things Stack. Dispone de configuración para la mayoría de las placas de desarrollo LoRa incluyendo las distintas definiciones de pines requeridas para el correcto funcionamiento. Permitiendo mostrar por pantalla o a través de led su funcionamiento si fuera el caso de estar incluido. Para la puesta en marcha requiere de muy poca configuración pero su única función es enviar un *uplink* con el número de una cuenta. Se indica espacios claros para la modificación del código por parte del usuario ya sea para añadir otras librerías, declaración de variables globales, inicialización de sensores o funcionamientos específicos. En este repositorio viene incluido el soporte de las librerías MCCI LoRaWAN LMIC library, IBM LMIC framework para funcionalidades LoRaWAN, U8g2 y EasyLed para display y leds.

- Filter.h de MegunoLink Library para el filtrado de señales analógicas.

- DHT.h de adafruit/DHT-sensor-library para el uso del sensor digital para temperatura y humedad relativa DHT11.
- BH1750.h de claws/BH1750 para el uso del sensor digital de intensidad de luz BH1750.

4. Desarrollo

Durante el planteamiento de este trabajo se decide implementar dos nodos, uno como control del riego automático y otro para obtener valores calibrados del ambiente siendo apoyado por sensores redundantes en el nodo principal.

Desde los sensores se comunicará a los Gateway disponibles mediante modulación LoRa cumpliendo los estándares y políticas de uso justo de la red proporcionada por The Things Stack que junto a la integración de MQTT enviará los datos a la aplicación Node-RED donde se podrán observar mediante una interfaz web. (Ilustr. 19)

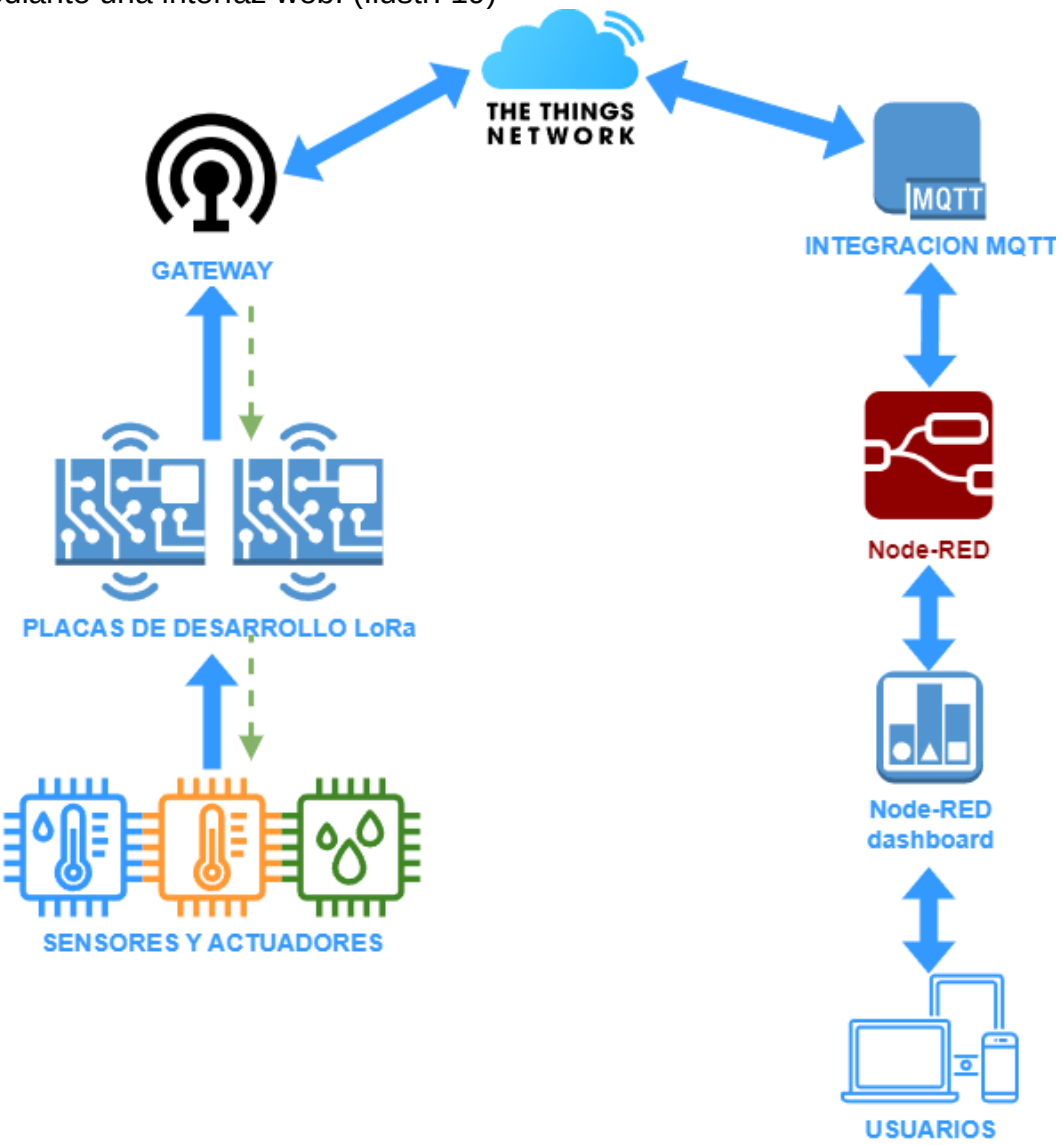

Ilustración 19 Esquema de implementación de la aplicación IoT

4.1. Desarrollo del código

Como se ha mencionado, el código para programar ambas placas de desarrollo se ha realizado a partir de ejemplos sencillos para la prueba de las distintas funcionalidades de los sensores, actuadores, comunicación LoRa y pantalla OLED de la placa de desarrollo. Una vez comprendido el funcionamiento este se ha incorporado al ejemplo LMIC-node modificando las partes necesarias para obtener un programa con el objetivo deseado.

En común ambos nodos se configuran con los siguientes aspectos en el archivo de configuración platformio.ini:

```
[platformio]
default_envs =

    ttgo_lora32_v2 ;Placa de desarrollo a utilizar

[common]
monitor_speed = 115200 ;Velocidad de monitorización por serial
build_flags =
    -D DO_WORK_INTERVAL_SECONDS=150 ;Tiempo entre lecturas
    ;Se puede modificar el tipo de activación del nodo añadiendo una etiqueta en este apartado
lib_deps =
    olikraus/U8g2 ;Libreria para pantalla OLED
    lnlp/EasyLed ;Libreria para uso de LED
    ;En el caso de querer añadir otras liberías como las de DHT se incluiria la dependencia aquí
[mcci_lmic]
lib_deps =
    mcci-catena/MCCI LoRaWAN LMIC library ;Libreria para la comunicación LoRaWAN
build_flags =
    -D ARDUINO_LMIC_PROJECT_CONFIG_H_SUPPRESS ; Para utilizar este archivo como configuración en vez de
                                              ;lmic_project_config.h.
    -D DISABLE_PING                           ; Se desabilita PING y BEACONS para dispositivos de Clase A
    -D DISABLE_BEACONS                        ;con el objetivo de ahorrar recursos de memoria
    -D CFG_sx1276_radio=1                     ; Radio que incorpora el chip
    -D USE_ORIGINAL_AES                       ; Tipo de encriptación utilizada
    -D CFG_eu868=1                            ; Para definir la configuración regional de Europa
    -D hal_init=LMICHAL_init                  ; Se añade porque hay un conflicto de librería entre las placas ESP32
                                              ;y la librería MCCI LoRaWAN
```

Ilustración 20 Configuración general del archivo platformio.ini

Se muestran solo las etiquetas utilizadas para la configuración del proyecto, el repositorio incluye opciones para añadir o modificar las etiquetas necesarias según nuestra región, placa de desarrollo u otros parámetros a modificar y se indica en tal caso cómo hacerlo. Entre los aspectos más destacables es indicar la placa de desarrollo, el tiempo que permanecerá en reposo en este caso es el tiempo entre lecturas de los datos de los sensores y configuración regional para la comunicación LoRaWAN.

```
[env:ttgo_lora32_v2]
; TTGO LoRa32 v2.0 (ESP32).
; Onboard OLED display SSD1306 0.96" 128x64.
; Requires manual wiring of DIO1 to GPIO33.
; GPIO22 is used for both onboard LED and I2C SCL
; therefore USE_LED and USE_DISPLAY cannot be used together.
platform = espressif32
board = ttgo-lora32-v2
framework = arduino
upload_speed = 921600
monitor_speed = ${common.monitor_speed}
lib_deps =
    ${common.lib_deps}
    ${mcci_lmic.lib_deps}
build_flags =
    ${common.build_flags}
    ${mcci_lmic.build_flags}
    -D BSFILE=\"boards/bsf_ttgo_lora32_v2.h\"
    -D MONITOR_SPEED=${common.monitor_speed}
    -D LMIC_PRINTF_TO=Serial
    -D USE_SERIAL
    ; -D USE_LED                ; Cannot be used together with USE_DISPLAY
    -D USE_DISPLAY              ; Cannot be used together with USE_LED
```

Ilustración 21 Configuración relacionada con la placa de desarrollo

En el mismo archivo se encuentran configuraciones y datos relevantes a la placa de desarrollo especificada. Para esta placa se especifica que es necesario hacer una conexión del pin DIO1 a GPIO33 para la comunicación LoRa y que no es posible hacer uso de la información LED y la pantalla al mismo tiempo. Para ambas aplicaciones se deshabilita tanto la pantalla como el LED ya que no le daremos uso. Se indica también el archivo donde encontraremos la asignación de pines de la placa.

En la subcarpeta llamada "*keyfiles*" se encuentra un archivo donde indicaremos las credenciales de aplicación, dispositivo y llave de aplicación que obtendremos desde el apartado de registro de dispositivos en nuestro portal de aplicación de The Things Stack. En el momento de programar ambas placas es necesario cortocircuitar el pin GPIO0 hasta que se instale en la placa el programa realizado.

4.1.1 Código para el desarrollo del Nodo 1 para la captación por sensores pre-calibrados de temperatura, humedad relativa e intensidad de luz.

Se incluyen las librerías necesarias para el sensor DHT11 y BH1750 así como el filtrado de los valores analógicos. Seguidamente se hace una definición de los pines en los que se conecta y se instancia tanto ambos sensores como los filtros para los 3 tipos de datos a leer, humedad relativa, temperatura e intensidad de luz. También se crea la variable global para el tamaño del buffer donde se cargarán los datos y el valor del factor de peso para el filtro por defecto. Los filtros aplicados emplean un peso, indicado como valoralfa, para darle más importancia a los datos más antiguos frente a los más recientes, el funcionamiento de estos filtros se explicará en el apartado 4.1.2.4. El segundo parámetro indicado en el filtro es el valor inicial analógico que inicialmente suponemos 0. (Ilustr. 22)

```
#include "DHT.h"
#include "BH1750.h"
#include "Filter.h"

#define DHTPIN 4 //conectado el sensor a GPIO4
#define DHTTYPE DHT11 //tipo de sensor DHT
#define PIN_SALIDA 13 //Pin para alimentar los sensores

BH1750 luxmeter; //Sensor de luz
DHT dht(DHTPIN, DHTTYPE); //inicializacion del sensor
const uint8_t payloadBufferLength = 6;    // Adjust to fit max payload length

//Filtros para las señales analogicas (valor alfa, valor inicial = 0)
float valoralfa = 10; //Por defecto el valor para el filtro es 10
ExponentialFilter<float> filtroHumedad(valoralfa, 0);
ExponentialFilter<float> filtroTemperatura(valoralfa, 0);
ExponentialFilter<float> filtroLuz(valoralfa, 0);
```

Ilustración 22 Librerías y declaración de variables Nodo 1.

Dentro de la función *"setup()" se* inicializa los sensores y los filtros con un primer valor analógico, además del pin 13 como salida. Esta función solo es llamada durante el primer ciclo del programa. (Ilustr. 23)

```
// Place code for initializing sensors etc. here.
dht.begin();
Wire.begin();
luxmeter.begin();
pinMode(PIN_SALIDA,OUTPUT); //Inicializamos el pin 13 como salida para alimentar los sensores

filtroHumedad.SetCurrent(dht.readHumidity());
filtroTemperatura.SetCurrent(dht.readTemperature());
filtroLuz.SetCurrent(luxmeter.readLightLevel());
```

Ilustración 23 Inicialización de sensores y filtros Nodo 1.

La función con la lectura de los datos analógicos se llama "*processWork*()" que será ejecutada cada 150s por defecto. Se hace una comprobación al inicio de la función para saber si el dispositivo ya se ha unido a la red LoRaWAN en caso contrario no se ejecuta el código y se procede a reintentar la conexión. Antes de la lectura de los sensores se pone en alta el pin 13 para alimentarlos, incluyendo un delay de 1 segundo para dar tiempo a la respuesta de los sensores. La lectura de los sensores se guarda en variables de tipo float y se operan para que ocupen un solo byte. Para este Nodo se ha querido utilizar un mensaje de 8 bytes con 2 bytes representando cada tipo de variable ambiental con y sin filtrado excepto para la intensidad de luz que cada valor ocupará 2 bytes.

Tabla 3 Distribución de bytes en el mensaje Payload

Payload								
Byte	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Variable	Humedad	Humedad Filtrada	Temperatura	Temperatura Filtrada	Luz byte bajo	Luz byte alto	Luz filtrada byte bajo	Luz filtrada byte alto
Tipo	float [0-100]	float [0-100]	float[0-50]	float[0-50]	int[0-54612]		int[0-54612]	
Conversión a byte	x2.55	x2.55	x5.1	x5.1	lowByte()	highByte()	lowByte()	highByte()

El valor de humedad relativa leído por el sensor es devuelto por la función de la librería dht.h y toma valores entre 0 y 100, para ocupar un byte entero multiplicamos por 2.55 y hacemos el cambio de tipo para la carga en un buffer.
Aunque la temperatura puede tomar valores negativos por el entorno en el que vamos a colocar el dispositivo podemos suponer que el rango real estará entre 0 y 50. El valor que devuelve la función del sensor dht se obtiene por defecto en grados centígrados y se hace la conversión a byte multiplicando por 255/(50-0).
La intensidad de luz devuelta por la función del sensor BH1750 se indica que puede tener valores entre 0 y 54612 en unidades de lux por lo tanto es necesario hacer uso de dos bytes sin ningún tipo de conversión.
Tras la lectura se pone a baja el pin 13 para dejar de alimentar a los sensores. (Ilustr. 24)

```
digitalWrite(PIN_SALIDA, HIGH); //Se pone a alta (3.3V) la salida para alimentar los sensores
delay(1000); //Se añade un segundo de delay para la respuesta de los sensores

// Read humidity value
float h = dht.readHumidity();
filtroHumedad.Filter(h);
float filteredMoisture = filtroHumedad.Current();
Serial.print("%RH ");
Serial.println(h);
byte humedad_raw = h * 2.55;
byte humedad_filtered = filteredMoisture * 2.55;

// adjust for the f2sflt16 range (-1 to 1)
//h = h / 100;
// Read temperature as Celsius
float t = dht.readTemperature();
filtroTemperatura.Filter(t);
float filteredTemperature = filtroTemperatura.Current();
Serial.print("Temperature: "); Serial.print(t);
Serial.println(" *C");
byte temp_raw = t * 5.1 ;
byte temp_filtered = filteredTemperature * 5.1;

// adjust for the f2sflt16 range (-1 to 1)
//t = t / 100;

//Read lux value
int lux = luxmeter.readLightLevel(); //Realizamos una lectura del sensor
filtroLuz.Filter(luxmeter.readLightLevel());
int filteredlux = filtroLuz.Current();
Serial.print("Light: "); Serial.print(lux);
Serial.println(" Lux");
byte luxLow = lowByte(lux);
byte luxHigh = highByte(lux);
byte luxLow_filtered = lowByte(filteredlux);
byte luxHigh_filtered = highByte(filteredlux);

digitalWrite(PIN_SALIDA, LOW); //Se pone a baja (0.0) la salida despues de la lectura
```

Ilustración 24 Lectura de los sensores Nodo 1.

Tras la lectura se prepara el envío del mensaje de los bytes anteriormente creados en un buffer. Se declara el puerto del *uplink* y el tamaño del buffer para la función “scheduleUplink” que intentará enviar el paquete de datos mediante LoRa en el próximo hueco disponible. (Ilustr. 25)

```
{
    // Prepare uplink payload.
    uint8_t fPort = 10;
    // place the bytes of temp into the payload
    payloadBuffer[0] = humedad_raw;
    payloadBuffer[1] = humedad_filtered;
    // place the bytes of humid into the payload
    payloadBuffer[2] = temp_raw;
    payloadBuffer[3] = temp_filtered;
    // place the bytes of lux into the payload
    payloadBuffer[4] = luxLow;
    payloadBuffer[5] = luxHigh;
    //bytes
    payloadBuffer[6] = luxLow_filtered;
    payloadBuffer[7] = luxHigh_filtered;

    uint8_t payloadLength = 8;

    scheduleUplink(fPort, payloadBuffer, payloadLength);
```

Ilustración 25 Carga en buffer y preparación del mensaje *uplink* para ser enviado por LoRa

Por último al finalizar la transmisión del mensaje se hace una espera para huecos Rx, donde se recibirá un *downlink* con un paquete de dato en esta ventana de tiempo, si se recibe algún paquete se llamará a la función “processDownlink()” con los parámetros del paquete, como el puerto, los datos y la longitud de estos. En esta función tendremos un switch que contemplará el tipo de dato y puerto recibido para modificar el parámetro alfa o el tiempo entre lecturas. Este switch es idéntico en ambos nodos. (Ilustr. 26)

```
const uint8_t cmdPort1 = 100; //Si el puerto es 100 se modifica el tiempo y el valor alfa.
const uint8_t cmdPort2 = 99; //Si el puerto es 99 se modifica solamente el valor alfa.

if (fPort == cmdPort1 && dataLength == 1)
{
    #ifdef USE_SERIAL
        printSpaces(serial, MESSAGE_INDENT);
        serial.println(F("Change time command received"));
    #endif
    ostime_t timestamp = os_getTime();
    switch(data[0]) {
        case 0:
            doWorkIntervalSeconds = 180;
            valoralfa = 10;
        break;
        case 1:
            doWorkIntervalSeconds = 360;
            valoralfa = 20;
        break;
        case 2:
            doWorkIntervalSeconds = 720;
            valoralfa = 40;
        break;
        case 3:
            doWorkIntervalSeconds = 1200;
            valoralfa = 80;
        break;
        case 4:
            doWorkIntervalSeconds = 1800;
            valoralfa = 90;
        break;
        default:
        break;
    }
    printEvent(timestamp, "Time between readings and alpha changed", PrintTarget::All, false);
}
```

Ilustración 26 Modificación de parámetros al recibir un downlink.

Para más información sobre el código se encontrará en el anexo un enlace al repositorio GitHub con ambos proyectos.

4.1.2 Código del desarrollo del Nodo 2 para el control del riego de una planta mediante sensores analógicos de temperatura, humedad del suelo, nivel de agua e intensidad de luz.

Antes de proceder a la explicación de los aspectos más importantes del código del segundo nodo es necesario tener en cuenta que los sensores utilizados en este nodo son completamente analógicos y por lo tanto por sí solos y sin calibración alguna solo representarán un valor de voltaje que aumentará o disminuirá en función de la resistencia o salida analógica que incluya este. Por lo tanto se ha realizado una calibración previa y parametrización de estos además de medido los niveles mínimos para la actuación en el caso del sensor de agua y de humedad del suelo. Además al obtener un valor analógico son más susceptibles al posible ruido e interferencias por lo tanto es de mayor relevancia el uso de algún tipo de filtrado ya que en caso contrario se interpretarían cambios en los valores leídos que no representan la realidad por lo tanto se decide implementar un filtrado de media en movimiento ponderada.

4.1.2.1 *Calibración y parametrización del sensor de temperatura termistor de tipo NTC.*

La calibración del termistor se ha realizado aplicando el modelo Steinhart-Hart. A partir de herramientas calibradas, en este caso apoyándose en el sensor DHT11 para comparar la temperatura se obtienen los valores de resistencia del termistor en 3 puntos distintos, dos en puntos extremos de temperatura a 4ºC y a 50ºC y otro a temperatura ambiente a 28.4 ºC. A partir de estos tres pares de valores se puede calcular los coeficientes de la ecuación y obtener una relación entre la salida analógica del sensor y la temperatura a la que se encuentra. Estos coeficientes se han calculado utilizando la aplicación NTCcalculator [14] (Ilustr. 27). Junto al valor de la resistencia pull-up del termistor se introducirán los coeficientes en el código para poder realizar una lectura en ºC del valor analógico.

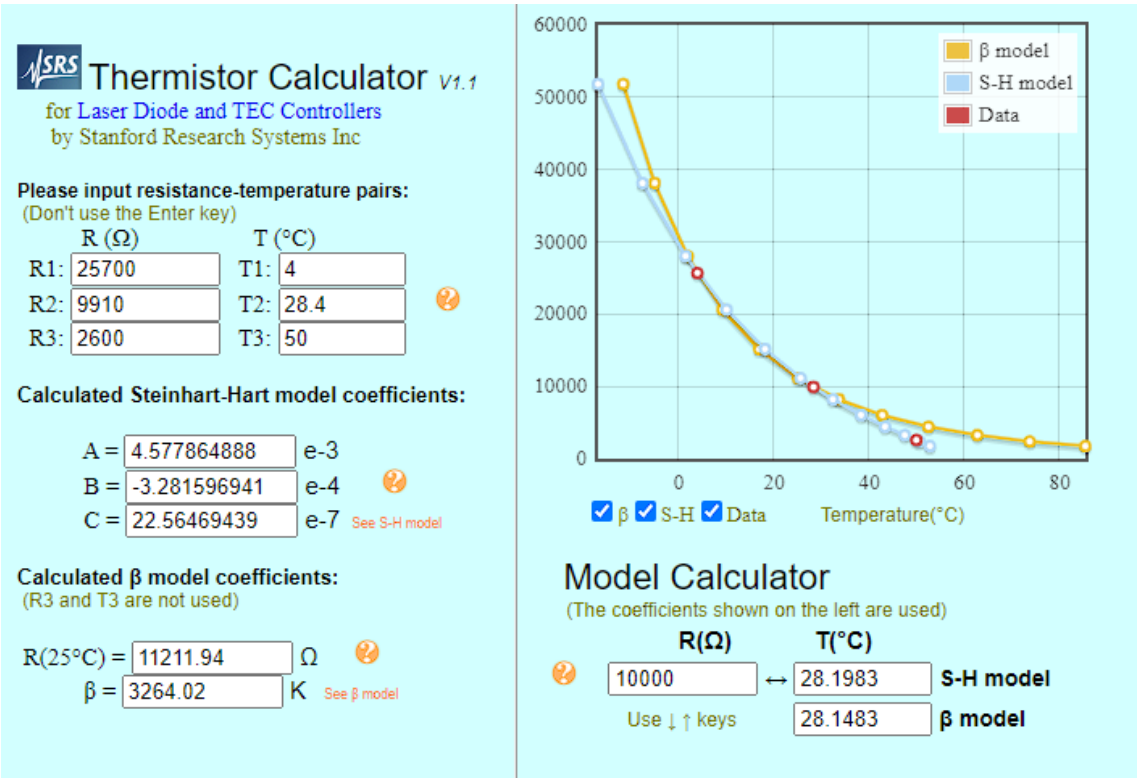

Ilustración 27 Aplicación con los 3 pares de datos tomados de forma experimental para la obtención de coeficientes del modelo S-H [16]

4.1.2.2 *Calibración y parametrización del sensor de intensidad de luz fotorresistor.*

La calibración y parametrización del fotorresistor se ha realizado utilizando una hoja de cálculo, sabiendo que la resistencia del fotorresistor decrece de forma exponencial frente a la intensidad de luz, al aplicar una escala logarítmica se pueden relacionar estas dos variables mediante una recta de pendiente m y cruce en b.

$$\log_{10}(\text{lux}) = m \cdot \log_{10}(R) + b \qquad (1)$$

Modificando la ecuación para despejar lux respecto a R se obtiene:

$$\text{lux} = R^{m} \cdot 10^{b} \qquad (2)$$

o

$$\text{lux} = A \cdot R^{B}\text{, con } A = 10^{b} \text{ y } B{=}m \qquad (3)$$

Para poder obtener una recta se miden 19 valores de R e intensidad de luz desde 1 lux con 130000 Ω hasta 3340 lux con una medición de 324 Ω. Estos valores se reflejan en la hoja de cálculo obtenida de la página web allaboutcircuits.com [17] e incluida en el anexo.

4.1.2.3 Parametrización de sensor de nivel de agua y humedad del suelo

El sensor de nivel de agua se humedece hasta que cubre un milímetro de su base y con un ejemplo básico programado en el nodo a modo de prueba se obtienen 10 medidas y se redondea. El valor analógico obtenido será el que añadiremos al código como límite.

Con el sensor de humedad del suelo se preparan 3 recipientes con la misma cantidad de tierra, el primero tierra completamente seca, el segundo se incorpora a la tierra un 10 % del volumen en agua y al tercero un 25% de agua a modo de simular el riego y nivel de conductividad del suelo deseada. A partir de un programa de ejemplo similar al del nivel de agua se obtienen valores por serial analógicos de las 3 muestras de tierra limpiando el sensor de residuos entre muestras. Se vuelve a preparar 3 recipientes de la misma manera y se toman 3 de nuevo valores analógicos. A partir de estos valores se puede separar en 3 conjuntos los datos analógicos, tierra húmeda, semi-humeda y seca. El valor para tierra seca que indicaremos en nuestro código como límite estará en torno al rango entre semi-humeda y seca según el interés.

4.1.2.4 Filtro de media móvil exponencial

Este filtro se utiliza para suavizar los valores analógicos obtenidos y eliminar ruidos posibles de interferencias o cambios drásticos en el entorno que puede que no representen la realidad. El funcionamiento es similar a un filtro pasa bajas. Se decide usar este filtro porque al aplicar una tendencia no es necesario la toma de muchos valores en un periodo de tiempo y realizar una media en ese intervalo, lo cual consumiría bastantes más recursos energéticos y de memoria en la placa de desarrollo. El funcionamiento de este filtro tiene menos en cuenta los valores más recientes frente a los anteriores dándoles un peso indicado como valor alfa para obtener el resultado. Sigue la siguiente ecuación:

$$y_n = w \cdot x_n + (1 - w) \cdot y_{n-1} \qquad (4)$$

Donde y_n representa el valor a la salida del filtro, w el peso alfa entre 0 y 1, x_n el valor actual medido, e y_{n-1} el último valor obtenido del filtro. Valores cercanos a la unidad darán mayor importancia a los datos nuevos mientras que valores bajos favorecerán a los datos más antiguos obteniendo un mayor suavizado.

4.1.2.5 Diseño del funcionamiento del control de la bomba de agua

Tras obtener los valores límites y coeficientes para parametrizar los sensores se realiza el diseño del código principal para controlar la bomba de agua. En el siguiente diagrama de flujo se puede observar el funcionamiento del código:

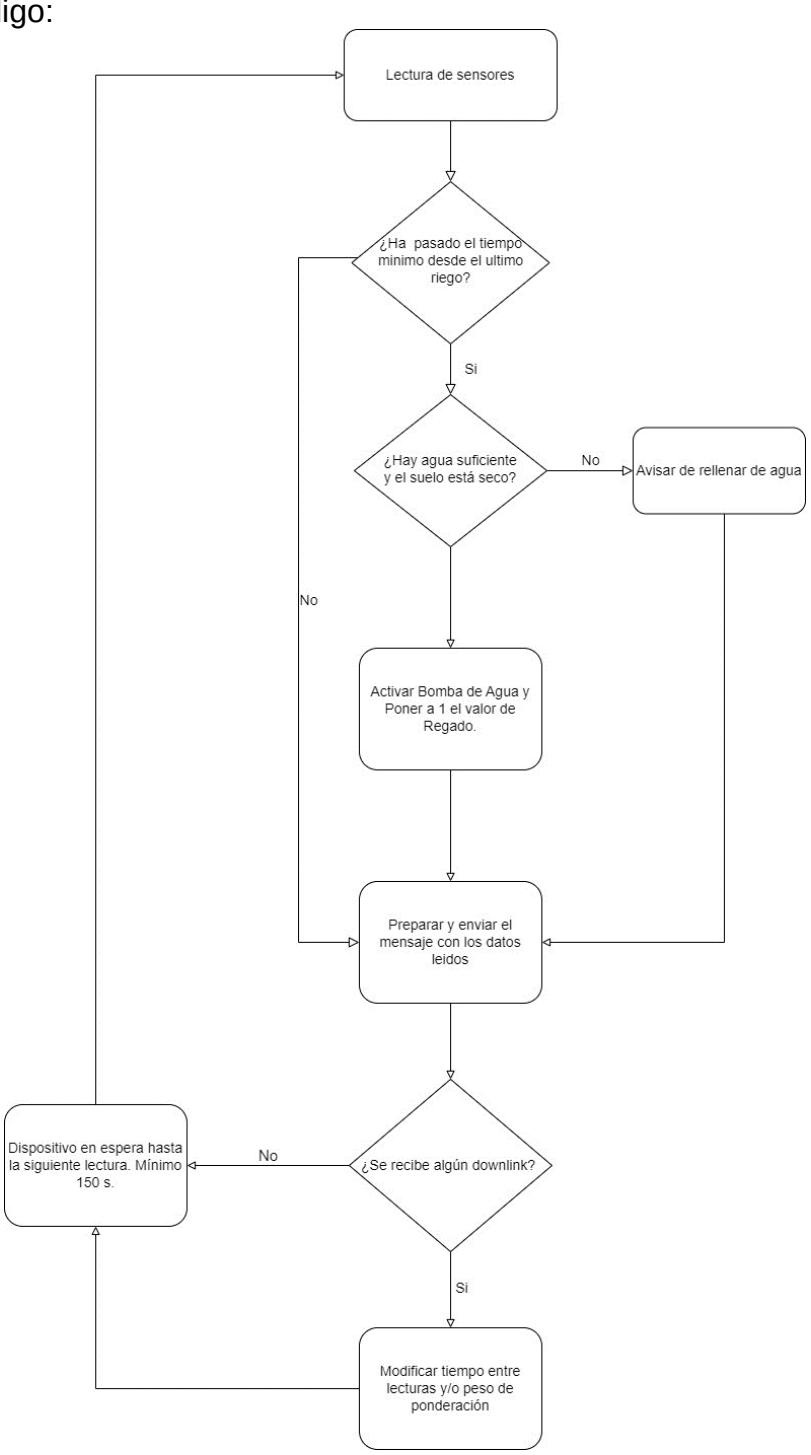

Ilustración 28 Diagrama de flujo del funcionamiento principal del Nodo 2

Este código al igual que el del Nodo 1 está implementado en la función "processWork()". Se incluye la librería "Filter.h" y se definen los pines a los que estarán conectados los 4 sensores. Tanto el relé como los sensores estarán alimentados por pines de la placa que se definirán al principio. Se inicializan las variables que contendrán los límites medidos de agua y humedad del suelo. Una variable para indicar el estado del riego y un valor alfa inicial para el peso del filtro. Se establece el tamaño del buffer como 6. Se declaran los coeficientes para el cálculo de la temperatura y la intensidad lumínica y se declaran los filtros para las señales. Se inicializa un contador desde el inicio del programa y un valor tiempo_espera que equivale a 8 horas para no realizar un riego seguido de otro y permitir que el sensor de humedad se adapte tras el regado. (Ilustr. 29)

```
#include <Filter.h>

#define waterPin 4 //Pin sensor nivel de agua
#define moisturePin 2 //Pin sensor humedad del agua
#define temperaturePin 15 //Pin sensor temperatura (termistor)
#define lightPin 13 //Pin sensor luz (fotoresistor)

//Pin que alimentara a los sensores analogicos para el sensor de nivel de agua
//y el sensor de humedad del suelo
#define sensorPower 12
#define relayPower 25 //Pin que alimentara al relay

//Valores limite para la automatizacion
float limiteAgua = 100; //Valor analogico minimo
float limiteSuelo = 3100; //Por encima es suelo seco 1100 es suelo humedo
bool regado = 0; //Se guarda aqui el valor de regado
float valoralfa = 10; //Por defecto el valor para el filtro es 10

const uint8_t payloadBufferLength = 6;    // Adjust to fit max payload length

//Coeficientes para calculo de la temperatura Steinhart-Hart
float logR2, R2, T;
float c1 = 4.577864888e-03; //coeficient A
float c2 = -3.281596941e-04; //coeficient B
float c3 = 22.56469439e-07; //coeficient C
float R1 = 10000; //Valor resistencia pull-up

//Coeficientes para pasar de R a Lumen
float Rp, ldrLux;
float LUX_CALC_EXPONENT = -0.813813917;
float LUX_CALC_SCALAR = 152821.3158;

//Filtros para las señales analogicas (valor alfa, valor inicial = 0)
ExponentialFilter<float> filtroAgua(valoralfa, 0);
ExponentialFilter<float> filtroHumedad(valoralfa, 0);
ExponentialFilter<float> filtroTemperatura(valoralfa, 0);
ExponentialFilter<float> filtroLuz(valoralfa, 0);

ostime_t tiempo_regado = os_getTime();
ostime_t tiempo_espera = sec2osticks(28800); //8 horas de espera para volver a regar
```

Ilustración 29 Librería y declaración de variables Nodo 2

Inicializamos los filtros y salidas de pines de la misma manera que el nodo anterior. Dentro de la función principal se comienza comprobando que el tiempo que ha pasado desde el ultimo riego no supere las 8 horas, si fuera el caso se pondría la variable booleana "regado" a 0 indicando que no se ha regado recientemente. A continuación se activa el pin para la lectura de los sensores y tras dos segundos se lee tanto el valor de humedad del suelo como nivel de agua. Para la conversión de float a un solo byte del valor analógico dividimos ambos números entre 16 ya que el rango analógico es entre 0 y 4095. Seguidamente se desactiva el pin para estos sensores. (Ilustr. 30)

```
Serial.print("Tiempo que ha pasado desde el ultimo riego");
Serial.println((os_getTime()-tiempo_regado));
Serial.print("Tiempo que tiene que pasar:");
Serial.println(tiempo_espera);
if ((os_getTime()-tiempo_regado) >= tiempo_espera) {
    regado = 0;
    Serial.println("Se puede volver a regar.");
    }

digitalWrite(sensorPower, HIGH); //Se pone a alta (3.3V) la salida para alimentar los sensores
delay(2000);//Espera de 2 segundo para la respuesta de los sensores


// Read water level value
float a = analogRead(waterPin);
filtroAgua.Filter(a);
float filteredWater = filtroAgua.Current();
Serial.print("Agua raw:");
Serial.print(a); Serial.print(" Agua raw post filter:"); Serial.println(filteredWater);

//Serial.print("Agua %:");
// float -> byte
byte payloadWater = filteredWater/16.0; //Division entre 16 para pasar de 4095 a 255
//Serial.print("Water byte:");
//Serial.println(payloadWater);

// Read moisture value
float h = analogRead(moisturePin);
filtroHumedad.Filter(h);
float filteredMoisture = filtroHumedad.Current();
Serial.print("Moisture raw:");
Serial.print(h); Serial.print(" Moisture raw post filter:"); Serial.println(filteredMoisture);

//Serial.print("Humedad suelo %:");
// float -> byte
byte payloadHumid = filteredMoisture/16.0; //Division entre 16 para pasar de 4095 a 255
//Serial.print("Humid byte:");
//Serial.println(payloadHumid);

digitalWrite(sensorPower, LOW); //Se pone a baja (0.0) la salida despues de la lectura
```

Ilustración 30 Lectura de humedad del suelo y nivel de agua

Con los valores analógicos filtrados del nivel de agua y humedad del suelo se compara si superan los límites establecidos y si no se ha regado recientemente. En este caso se pone a alta el pin para activar la bomba durante 5 segundos, se cambia el estado a 1 del valor de regado y tras los 5 segundos se apaga la bomba poniendo el pin a baja. Se actualiza el valor del temporizador de regado. En otro caso de que no se cumplan estas condiciones pero sí que el límite de agua esté por debajo del necesario se imprimirá por serial una advertencia. (Ilustr. 31)

```
if( (filteredWater > limiteAgua) && (filteredMoisture > limiteSuelo) && (regado == 0)){
    digitalWrite(relayPower, HIGH); // the water pump fills the bottle
    Serial.println("Se ha activado la bomba");
    delay(5000); //Activa la bomba durante 5s
    regado = 1;
    digitalWrite(relayPower, LOW); // the water pump stops
    Serial.println("Se ha apagado la bomba");
    tiempo_regado = os_getTime();
    Serial.println(tiempo_regado);
}
else if(filteredWater <= limiteAgua ){
    Serial.println("Aviso: Es necesario llenar el deposito de agua");
}
```

Ilustración 31 Activación de la bomba de agua en caso de cumplir las condiciones

A continuación se realizará una medida de los valores analógicos de los sensores de temperatura e intensidad de luz. Con estos valores se realizará los cálculos a partir de las ecuaciones con coeficientes mencionadas en los apartados 4.1.2.2 y 4.1.2.1. Aunque no se ha hecho para la temperatura en el Nodo 1, se ajusta el valor calculado dividiendo entre 100 para que esté en el rango -1 y 1 permitiendo transformar a partir de la función LMIC_f2sflt16() en un entero de 16 bits que posteriormente será interpretado con un decodificador. Se realizan los cálculos necesarios para obtener la intensidad de luz y se transforma para ocupar dos bytes. Estos valores se envían de la misma manera que el Nodo 1. (Ilustr. 32)

```
// Read temperature as Celsius
float t = analogRead(temperaturePin);
filtroTemperatura.Filter(t);
float filteredTemp = filtroTemperatura.Current();
Serial.print("Temp raw:");
Serial.print(t); Serial.print(" Temp raw post filter:"); Serial.println(filteredTemp);

//Conversión de valor analogico a ºC
R2 = R1 * ((4095.0 / (float)filteredTemp) - 1.0); //Valor de la resistencia termistor
logR2 = log(R2);
T = (1.0 / (c1 + c2*logR2 + c3*logR2*logR2*logR2))- 273.15;
//Serial.print("Temperature: "); Serial.print(T);
//Serial.println(" ºC");
// adjust for the f2sflt16 range (-1 to 1)
T = T / 100;
// float -> int
// note: this uses the sflt16 datum (https://github.com/mcci-catena/arduino-lmic#sflt16)
uint16_t payloadTemp = LMIC_f2sflt16(T);
//Serial.print("Temp byte:");
//Serial.println(payloadTemp);
// int -> bytes
byte tempLow = lowByte(payloadTemp);
byte tempHigh = highByte(payloadTemp);


//Read lux value
float lux = analogRead(lightPin); //Realizamos una lectura del sensor
filtroLuz.Filter(lux);
float filteredLight = filtroLuz.Current();
Serial.print("Light raw:"); //DEPURAR
Serial.print(lux); Serial.print(" Lux raw post filter:"); Serial.println(filteredLight); //DEPURAR
//Conversion a lumen
Rp = R1 * ((4095.0 / (float)filteredLight) - 1.0); //Valor de la resistencia photoresistor
uint16_t ldrLux = LUX_CALC_SCALAR * pow(Rp, LUX_CALC_EXPONENT); // ecuacion con exponentes de hoja de excel
//Serial.print("Light byte:"); //DEPURAR
//Serial.println(ldrLux); //DEPURAR
byte luxLow = lowByte(ldrLux);
byte luxHigh = highByte(ldrLux);
```

Ilustración 32 Lectura y formato de temperatura e intensidad de luz Nodo 2

4.2 Creación de aplicación en The Things Stack y registro de nodos

Para empezar a diseñar la aplicación de IoT crearemos una cuenta en thethingsnetwork.org donde elegiremos la versión community que nos aportará acceso a The Things Stack de forma gratuita. Accedemos al apartado “console” desde nuestro perfil y nos pedirá elegir el clúster o grupo regional al que unirnos que en este caso es Europa. Desde la pestaña de aplicaciones creamos una aplicación con un identificador en minúscula, números y guiones. Para este TFG el identificador es ttgolora32oledv2. Desde la pestaña principal de la aplicación seleccionamos “Register end device” para registrar un dispositivo. Como las placas de desarrollo no están registradas introduciremos los datos manualmente, indicamos el plan de frecuencia europeo “Europe 863-870 MHz (SF9 for RX2 – recommended)”, la versión LoRaWAN de las placas TTGO “LoRaWAN Specification 1.0.2” y la versión Paramétrica regional “RP001 Regional Parameters 1.0.2 revision B”.

Como nuestro dispositivo no tiene un identificador “JoinEUI” se usará el que se indica por defecto “00 00 00 00 00 00 00 00”. A continuación se generará el identificador único del dispositivo y su llave de encriptación única. Por último se le asignará un nombre siguiendo los mismos requisitos que el nombre de la aplicación. Estos últimos tres identificadores son los que se indicarán en el archivo de llaves para los dos programas de ambos nodos.

Dentro del apartado del nodo registrado se puede consultar esta información por si es necesaria, además de ser posible ver los datos recibidos y enviados en los últimos minutos, enviar mensajes o simular el recibo de mensajes para depurar y por último con bastante importancia el apartado “Payload formatters” que nos permitirá escribir un pequeño código en JavaScript en este caso para poder decodificar los mensajes recibidos del nodo. (Ilustr. 33)

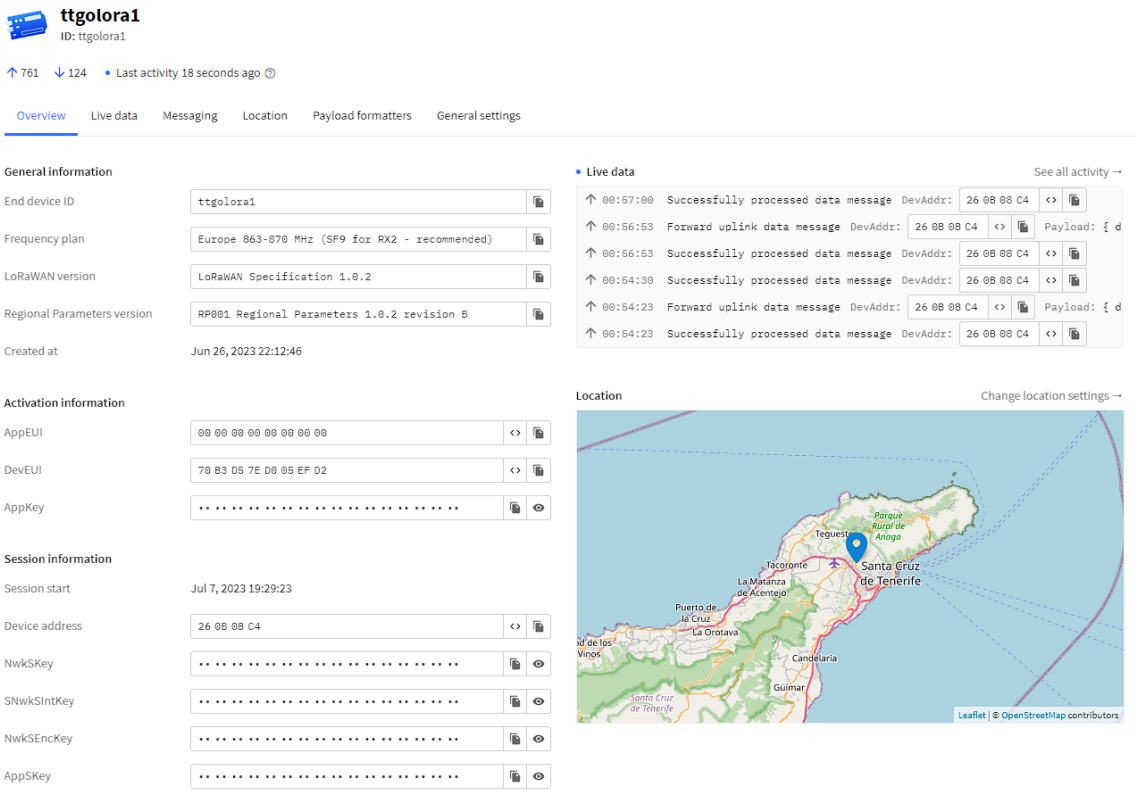

Ilustración 33 Datos y apartados visibles del Nodo 2

El código de este formateador almacena en un array de bytes el mensaje y transforma los bytes según indiquemos. Para la temperatura del Nodo 2 se indica también la función sflt162f para obtener el valor en grados centígrados multiplicando el resultado por 100 ya que previamente en el nodo se había dividido por este número. Para representar la humedad del suelo como un porcentaje se divide el valor almacenado en el byte entre 2.55 y con el nivel del agua se hace lo mismo. El valor de intensidad de luz se obtiene desplazando bit a bit 8 veces el valor alto del número lo que es igual que multiplicar el byte alto por 256 y sumarle el byte bajo. Los valores transformados se guardan en la variable decoded, decoded.degreesC será la temperatura, decoded.moisture la humedad del suelo, decoded.water el nivel de agua y decoded.light la intensidad de luz. La función devuelve esta variable que permitirá obtener un mensaje de tipo JSON con los valores decodificados. Se puede probar este formateador introduciendo un mensaje Payload válido y observando los resultados decodificados. (Ilustr. 34)

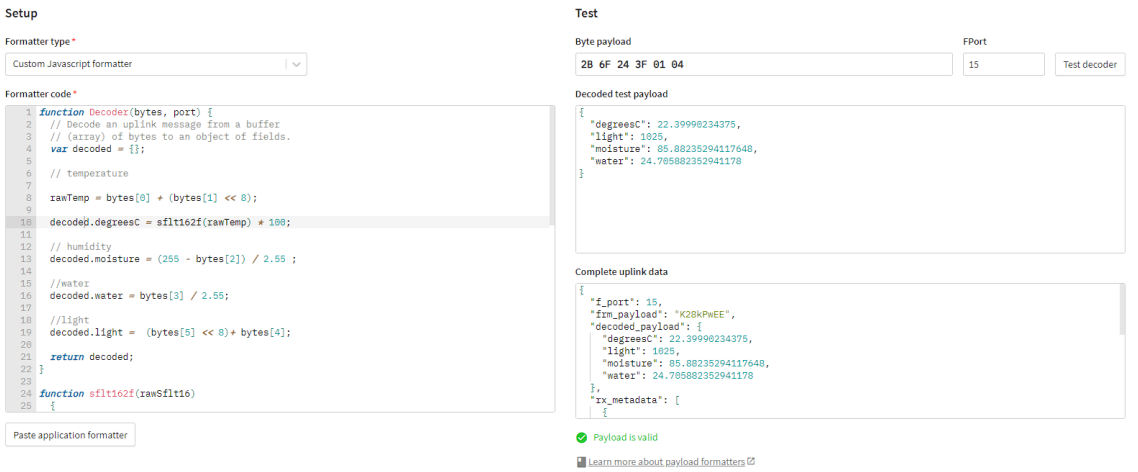

Ilustración 34 Formateador de Payload para Nodo 2

Existen formateadores específicos de las distintas integraciones incluidas. Se probó a utilizar CayeneLPP ya que existe una aplicación web de myDevices con una interfaz visual intuitiva para mostrar los datos y almacenarlos durante 30 días pero como inconveniente exigía el uso de dos bytes para cada tipo de medida como identificador del tipo de dato y del canal. Es decir un valor de temperatura que con el formato original ocupa 2 bytes con CayenneLPP ocuparía 4.

Durante el registro del primer dispositivo TTGO LoRa32 V2.0 en The Things Stack surgía un error con la creación de las credenciales por lo tanto fue necesario instalar el cliente de The Things Stack para acceder a la aplicación desde un entorno de comandos. Para el Nodo 2 esto no fue necesario.

4.2.1 Integración con MQTT y Node-RED

La integración con MQTT es muy sencilla, desde la pestaña de integraciones a la izquierda del menú principal de The Things Stack se accede a la parte de MQTT donde se visualizará los datos de conexión, se generará una llave API y un usuario que servirá para realizar la conexión con Node-RED.

Creamos una cuenta en la plataforma Azure de Microsoft para poder crear un entorno virtual que hará de servidor para Node-RED. La documentación aportada por Node-RED explica los pasos para crear la máquina virtual con las propiedades necesarias. Una vez la maquina virtual se está ejecutando desde la consola de comandos de Ubuntu instalamos tanto node.js como Node-RED. Abrimos la instancia con el comando node-red y accedemos a ella a partir de la dirección IP proporcionada por Azure. La dirección para el desarrollo de este TFG es http://13.95.108.244:1880/ , se asigna un nombre dns tfglora.westeurope.cloudapp.azure.com:1880. Desde estos enlaces se puede acceder desde cualquier navegador web al editor de Node-Red donde se creará la aplicación. Como cualquier persona puede acceder se asigna un usuario y contraseña.

Dentro del editor de Node-Red empezaremos recibiendo mensajes arrastrando el nodo “mqtt in” al espacio de trabajo. Configuramos el nodo con las propiedades del servidor que obtenemos desde The Things Stack introduciendo el usuario y contraseña o llave API generada anteriormente.

Podemos elegir la suscripción a todos los *topics* que permiten filtrar las distintas categorías de mensaje desde el servidor MQTT o podemos indicar un *topic* en concreto como es en este caso “v3/ttgolora32oledv2@ttn/devices/ttgolora2/up”. Con este topic recibimos solo los *uplink* del dispositivo ttgolora2 de la aplicación ttgolora32oledv2. (Ilustr. 35)

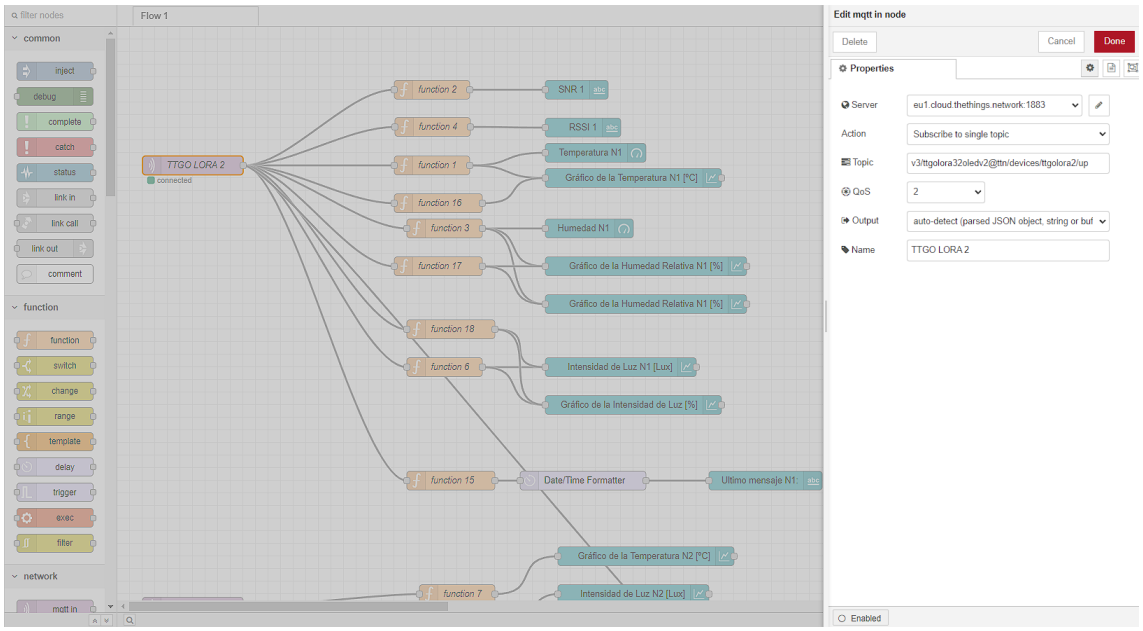

Ilustración 35 Parámetros del nodo mqtt in

A continuación se añade el nodo *function* para diseñar una función en JavaScript y procesar los paquetes por el nodo como se desee. Los mensajes son recibidos y devueltos por la función en un objeto JavaScript llamado msg con el mensaje en msg.payload por defecto. Una vez tenemos el Nodo 1 operativo, queremos mostrar los 3 valores medidos, para ello crearemos 3 funciones. Como ejemplo se utilizará la función para la temperatura. El objeto recibido por el nodo *mqtt in* tiene formato JSON por lo tanto si añadimos un nodo debug y copiamos la dirección en la que se encuentra la temperatura tendremos la siguiente dirección. (Ilustr. 36)

```
msg.payload.uplink_message.decoded_payload.degreesC
```

Redondeamos a dos cifras significativas y modificamos el tópico para diferenciar el mensaje. Por último se devuelve el mensaje modificado para procesarlo en el siguiente nodo a través de un flujo.

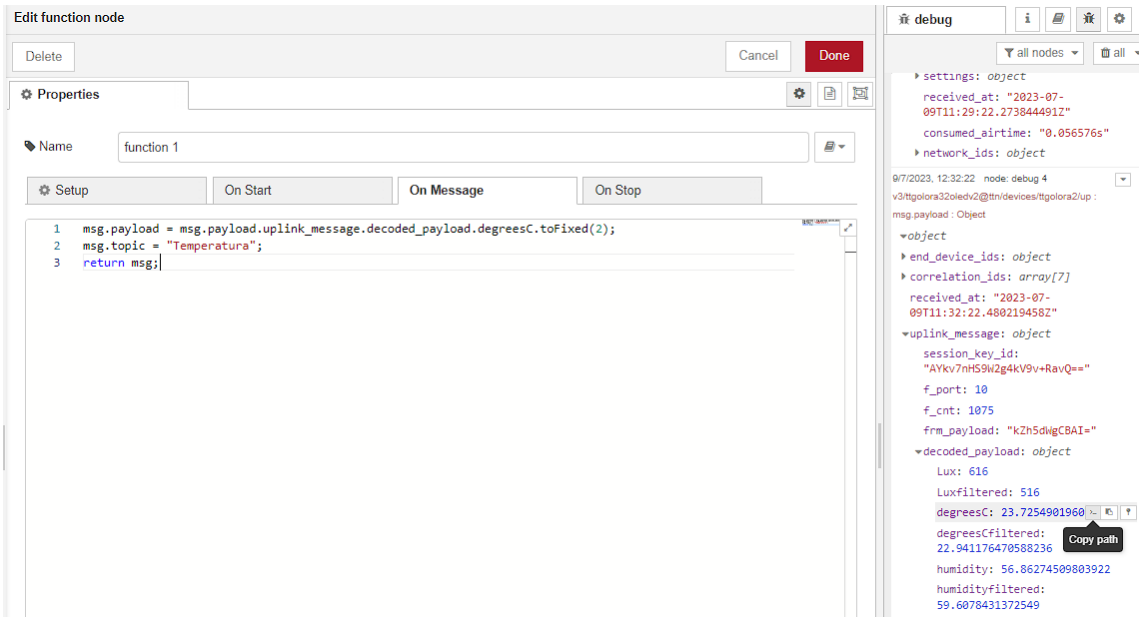

Ilustración 36 Función 1 para obtener la temperatura sin filtrar del Nodo 1

Se procede de la misma manera con el resto de los valores. En el nodo 1 tendremos un total de 9 funciones: 2 para las características de ruido de la señal, 6 para los 3 pares de valores no filtrados y filtrados de temperatura, humedad relativa y intensidad de luz, y 1 para la fecha del último mensaje recibido. Se puede observar estas 9 funciones como los nodos naranjas en la ilustración 35.

El Nodo 2 cuenta con 8 funciones, 2 para las características de ruido, 1 para la fecha y hora del último mensaje, 4 para los valores medidos por los sensores y 1 para crear un aviso cuando el nivel de agua es demasiado bajo.

4.2.1.1 Interpretación de los mensajes mediante la paleta dashboard de Node-RED y bot de avisos de Telegram

Instalando la paleta *dashboard* desde el menú de Node-RED podemos acceder a un nuevo grupo de nodos que nos permitirá procesar los mensajes a la salida de las funciones y visualizarlos en una interfaz en la dirección http://tfglora.westeurope.cloudapp.azure.com:1880/ui . Estos nodos tienen un color azul turquesa y permiten tanto mostrar información a un usuario como añadir interacciones por medio de botones, entradas de texto etc. Los nodos que se utilizaron son:

- *Chart,* para mostrar una gráfica en el tiempo de los distintos valores.
- *Text,* para mostrar en formato de texto tanto información de la señal como la fecha del último mensaje recibido.
- *Gauge,* para mostrar un indicador con el valor de temperatura o humedad último.
- *Level,* para mostrar una barra de nivel tanto para el depósito de agua como la humedad del suelo.

Las opciones disponibles al añadir un nodo tipo *chart* comienzan con el grupo al que pertenecerá separándose visualmente un grupo de otro en columnas. Se especificará una etiqueta si es necesario el tipo de gráfico, valores mínimos y máximos de los ejes y sus etiquetas. Se puede mostrar varios flujos o datos por gráfico hasta un máximo de 9 cada uno con un color distinto. (Ilustr. 37).

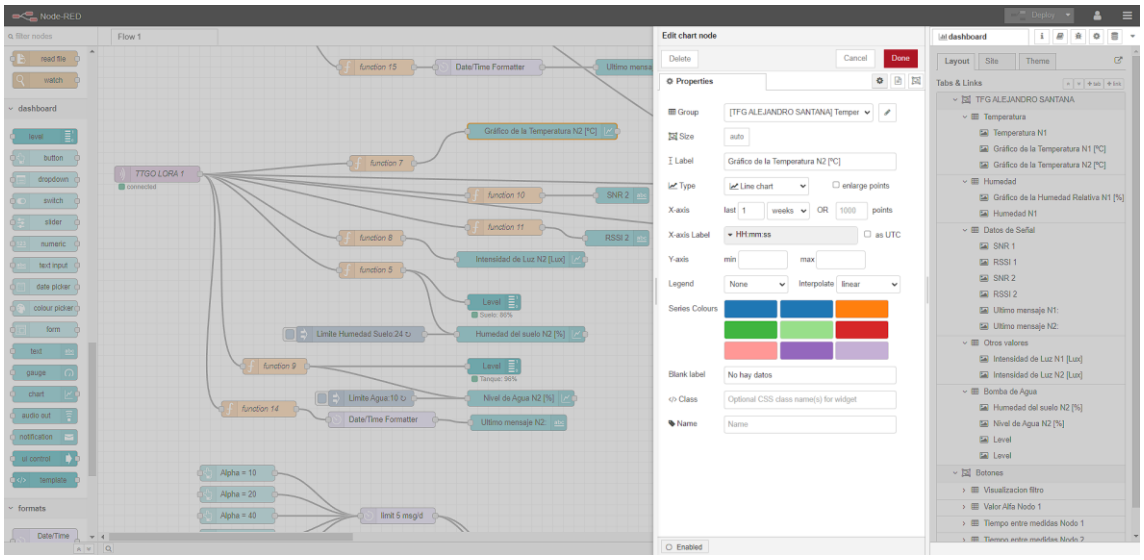

Ilustración 37 Interfaz donde se muestra la paleta dashboard, la configuración de la interfaz a la derecha y la configuración del nodo chart para la temperatura del NODO 2

A la hora de visualizar los límites de nivel de agua y humedad del suelo se añaden dos nodos de tipo *inject* para enviar un mensaje con el valor límite constante cada 5 minutos que se mostrará gráficamente junto al valor medido.

Instalamos la paleta Telegram para poder realizar la comunicación de avisos pertinentes. Para ello también es necesario crear un bot de Telegram, este se llamará AvisoLoraWanbot. El usuario tiene que empezar un chat con este bot con el comando /start lo cual le permitirá a Node-RED enviar los avisos. Añadimos el nodo *Telegram sender* de color azul y en las propiedades del nodo indicamos el nombre del bot y un token de acceso que nos fue proporcionado al crear el bot. En la función 19 se lee la variable del nivel de agua y se comprueba si es menor a 10, en ese caso se crea una variable msg.payload nueva con el chatId de la conversación de Telegram del Usuario, el tipo de mensaje y el contenido de este. Para avisar de la desconexión de alguno de los dispositivos LoRa se añade un nodo función *trigger*, este nodo comienza una cuenta atrás de 40 minutos tras recibir un objeto de tipo msg, si tras los 40 minutos no se ha recibido otro se crea un objeto msg.payload con el aviso al usuario de que se ha desconectado.

Por último a partir de la paleta dashboard se añade el nodo *button*, se mostrará una serie de botones en la interfaz para modificar el tiempo entre lecturas y el valor alfa para el filtrado. Al pulsar el botón se genera un mensaje de tipo downlink con el formato mostrado en la ilustración 38.

Ilustración 38 Formato del mensaje JSON que se envía al pulsar el botón alfa = 100

El byte se representa en formato base64 que en este caso para cambiar al valor alfa 100 en cualquiera de los nodos es necesario utilizar el puerto 99 con un byte de valor 4 en decimal. Se limita el número de veces que el usuario puede pulsar el botón a 10 por nodo por día para seguir las reglas de uso justo de The Things Stack. Este mensaje se publica mediante MQTT con el nodo *mqtt out.*

4.3. Otras implementaciones

Durante el desarrollo de este proyecto se optó por otras alternativas para la gestión de los datos recibidos por The Things Stack, al ofrecer una gran cantidad de integraciones primero se realizó la conexión con MQTT mediante alternativas más sencillas. La primera fue el cliente Mosquitto como bróker, este se ejecuta desde un entorno de comandos y se muestra por consola los mensajes recibidos a los que se ha suscrito y también permite publicar. Se acabó descartando esta opción ya que para almacenar los datos y mostrarlos se necesitaban más herramientas como una base de datos y un programa con interfaz gráfica externa.

Otra opción fue Cayenne de myDevices mencionado en el apartado 4.2. La razón por la que se descartó esta forma de visualizar los datos a pesar de ser bastante intuitiva no solo es el formato de Payload restringido sino el poco control que se tiene sobre los datos.

4.4. Resultados y discusión

Ambos nodos fueron colocados en una ventana de un edificio durante un periodo de 3 días conectados a la red eléctrica para poder ver una evolución continua de los datos en la interfaz. Se realizaron pruebas colocando el dispositivo Nodo 2 en una azotea para aprovechar el módulo de recarga con panel solar y se logró un uso continuo de varias horas pero no fue posible mantenerlo de forma autosuficiente ya que la corriente de consumo de la placa de desarrollo incluso en reposo era mayor de lo esperado entorno a 60mA entre medidas. Durante la desconexión de algunos de los sensores para comprobar su consumo se observan picos en los datos ya que los pines de la placa se encontraban en punto flotante y podrían indicar su valor máximo 3.3V o 0 V. Junto a la aplicación Serial USB Terminal se conectaron los dispositivos para obtener datos del funcionamiento mediante serial. (Ilustr. 39)

```
00:01:29.592 000029700881:  Event: EV_JOIN_TXCOMPLETE
00:03:28.533 000037135135:  Event: EV_TXSTART
00:03:34.024 000037478082:  Event: EV_JOINED
00:03:34.024            Network Id: 19
00:03:34.029            Device Address: 260BE832
00:03:34.029            Application Session Key: 21-F6-9B-D3-DE-94-68-1A-38-23-63-C9-01-
68-B7-7A
00:03:34.038            Network Session Key:    8E-39-63-CC-58-31-40-6F-B3-84-0A-B2-F3-
B7-06-04
00:03:34.826
00:03:34.826 000037530351:  doWork job started
00:03:34.854 %RH 72.00
00:03:34.854 Temperature: 23.30 *C
00:03:34.854 Intensity:553.00
00:03:34.854 Moisture:1658.00
00:03:34.870 000037532013:  Input data collected
00:03:34.870            Temp value: 28533
00:03:34.898 000037532955:  Packet queued
00:03:34.929 000037534951:  Event: EV_TXSTART
00:03:41.297 000037932725:  Event: EV_TXCOMPLETE
00:03:41.307            Up: 1,  Down: 1
00:03:41.334            Downlink received
00:03:41.334            RSSI: -116 dBm,  SNR: 5.7 dB
00:03:41.336            Port: 0
00:03:53.454 000038692703:  Event: EV_TXSTART
00:03:59.680 000039081640:  Event: EV_TXCOMPLETE
00:03:59.690            Up: 2,  Down: 1
00:06:04.828
00:06:04.828 000046905351:  doWork job started
00:06:04.855 %RH 72.00
00:06:04.855 Temperature: 23.30 *C
00:06:04.855 Intensity:371.00
00:06:04.855 Moisture:1467.00
00:06:04.870 000046906991:  Input data collected
00:06:04.870            Temp value: 28533
00:06:04.898 000046907929:  Packet queued
00:06:04.929 000046909918:  Event: EV_TXSTART
00:06:11.177 000047300141:  Event: EV_TXCOMPLETE
00:06:11.186            Up: 3,  Down: 1
00:08:34.829
00:08:34.829 000056280351:  doWork job started
00:08:34.855 %RH 72.00
00:08:34.855 Temperature: 23.30 *C
00:08:34.855 Intensity:371.00
00:08:34.855 Moisture:1475.00
00:08:34.871 000056281988:  Input data collected
00:08:34.871            Temp value: 28533
00:08:34.899 000056282923:  Packet queued
00:08:34.930 000056284914:  Event: EV_TXSTART
00:08:41.177 000056675152:  Event: EV_TXCOMPLETE
00:08:41.187            Up: 4,  Down: 1
00:11:04.829
00:11:04.829 000065655351:  doWork job started
00:11:04.856 %RH 72.00
00:11:04.856 Temperature: 23.20 *C
00:11:04.856 Intensity:375.00
00:11:04.856 Moisture:1485.00
00:11:04.871 000065656985:  Input data collected
00:11:04.871            Temp value: 28525
00:11:04.899 000065657920:  Packet queued
00:11:04.930 000065659907:  Event: EV_TXSTART
00:11:11.178 000066050131:  Event: EV_TXCOMPLETE
```

Ilustración 39 Salida por serial del Nodo 1 con unión y 5 uplinks generados. Pantalla de la aplicación Serial USB Terminal para Android

Haciendo el cálculo entre tiempo de envío de los mensajes se obtiene un número aproximado a los 150 segundos indicados en el programa. Aunque los valores del sensor DHT11 de humedad relativa y temperatura no varían mucho entre lecturas se observa que la intensidad de luz sí. Durante esta prueba al Nodo 1 también estaba conectado el sensor de humedad del suelo y vemos como lee un valor entorno a 1475 que indica suelo semi húmedo.

Analizando un *uplink* recibido del Nodo 2 con la siguiente *Payload*:

6C 74 21 3C FE 00

Bytes 0 y 1 [6C 74] equivalen a la temperatura y con la transformación del decodificador a partir de la función sflt162f():

degreesC: 27.63671875

El byte 2 [0x21 -> 00100001 -> 33] representa la humedad del suelo. Para obtener un valor entre 0 y 100 como porcentaje:

moisture: (255 – 33) / 2.55 = 87.06 %

El byte 3 [0x3C -> 00111100 -> 60] representa el nivel de agua. Para obtener un valor entre 0 y 100 como porcentaje sabiendo que para un valor de 62.5 el sensor está totalmente cubierto:

water: 60 * 1.6 = 96 %

Los bytes 4 y 5 [FE 00 -> 11111110 00000000 -> 254 0] representan la intensidad de luz. Se realiza el acarreo de bits y se obtiene el valor en lux.

light: 254 + 0 * 256 = 254 lux

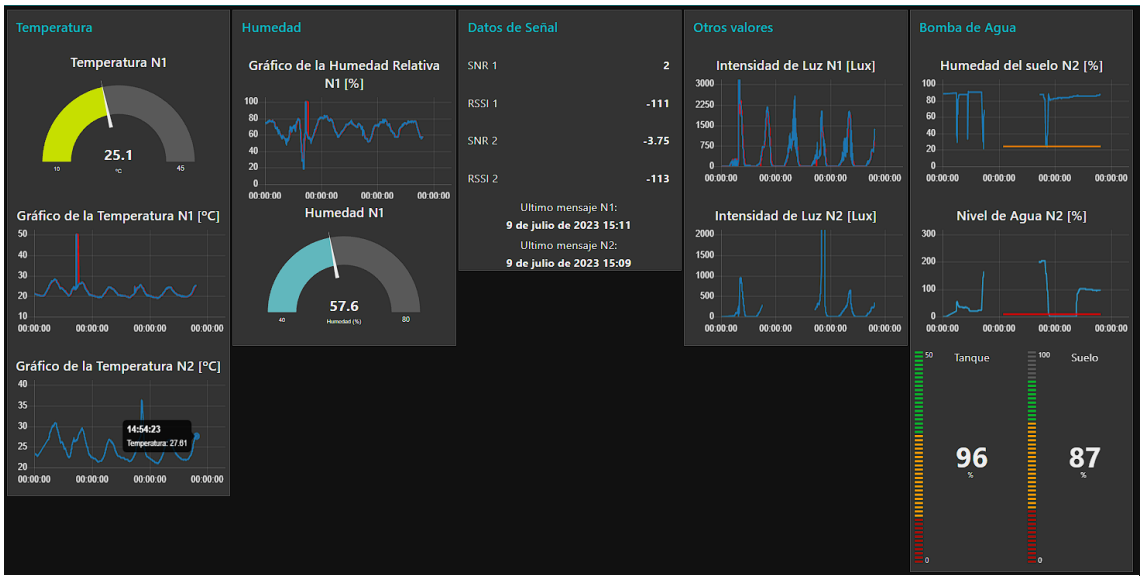

Ilustración 40 Interfaz visual Node-RED con datos entre 06/07/2023 y 09/07/2023

Comprobamos que estos son los datos recibidos en la interfaz de Node-RED en la ilustración 40.

Analizando los datos visibles en el *dashboard* se pueden extraer distinta información como los valores máximos y mínimos de cada unidad medida niveles del depósito y humedad. En particular observando unos picos de bajada en la humedad del suelo del Nodo 2 sin conocimiento externo podría dar a entender errores en la medida pero realmente lo que se estaba realizando eran pruebas para comprobar el funcionamiento de la bomba de agua cuando el nivel estaba por debajo de la línea naranja que indica el límite de sequedad del suelo. Con el depósito de agua surge algo similar, hay un periodo que se mantiene vacío ya que el agua se estaba filtrando por un orificio y se decide tapar con sellador que tiene que permanecer seco un tiempo, luego tras rellenar el depósito se visualiza un aumento. Hay segmentos sin datos en 3 gráficos del Nodo 2 porque se modificó el tópico del mensaje que recibía el nodo y luego se eliminó la función correspondiente perdiendo la información. La información coincide con la realidad habiendo diferencias muy leves entre los valores de temperatura e intensidad que marcan ambos nodos por la disposición de los sensores.

Para ilustrar el funcionamiento del filtro se ha graficado tanto la humedad relativa del Nodo 1 como la intensidad de luz. En rojo se observa el valor filtrado durante las últimas 5 horas con un peso del 10% y se aprecia un mayor suavizado en ambas curvas (Ilustr. 41)

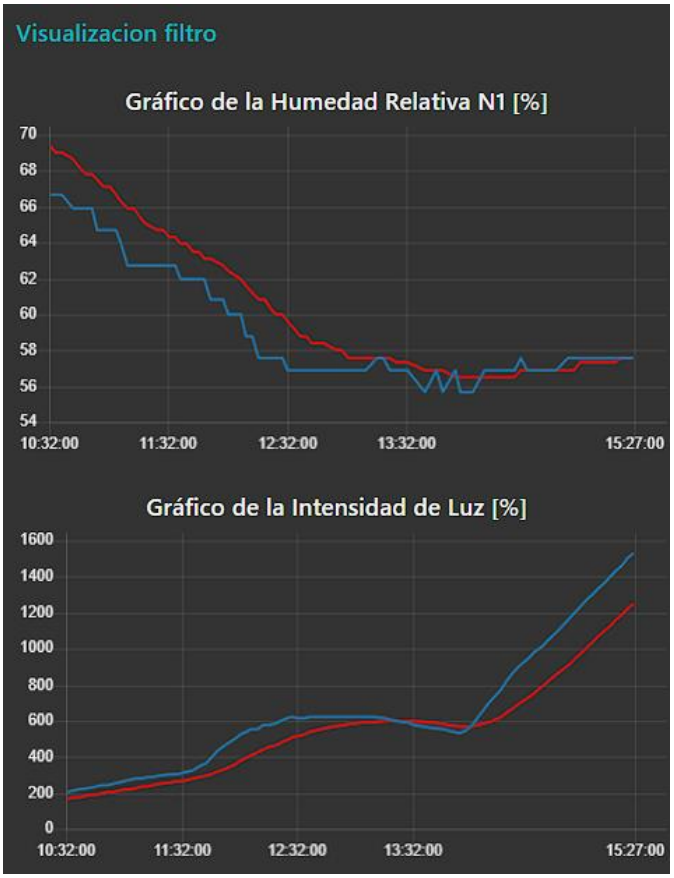

Ilustración 41 Curvas con Humedad e Intensidad de Luz con y sin filtrado

Si presionamos un botón para modificar el tiempo entre lecturas del Nodo 1 para que se realicen cada 12 minutos en vez de cada 2.5 minutos podremos ver el paquete a enviar desde la consola de comando de The Things Stack. Este será recibido por el nodo la próxima vez que se comunique con la red. (Ilustr. 42) Aunque lo más lógico sería obtener el mayor número de datos posible pueden surgir casos en los que se prefiera conservar energía haciendo una lectura y manteniendo el dispositivo en reposo durante un mayor tiempo o cambiar simplemente el valor del peso alfa para poder tener una lectura más próxima a la del sensor.

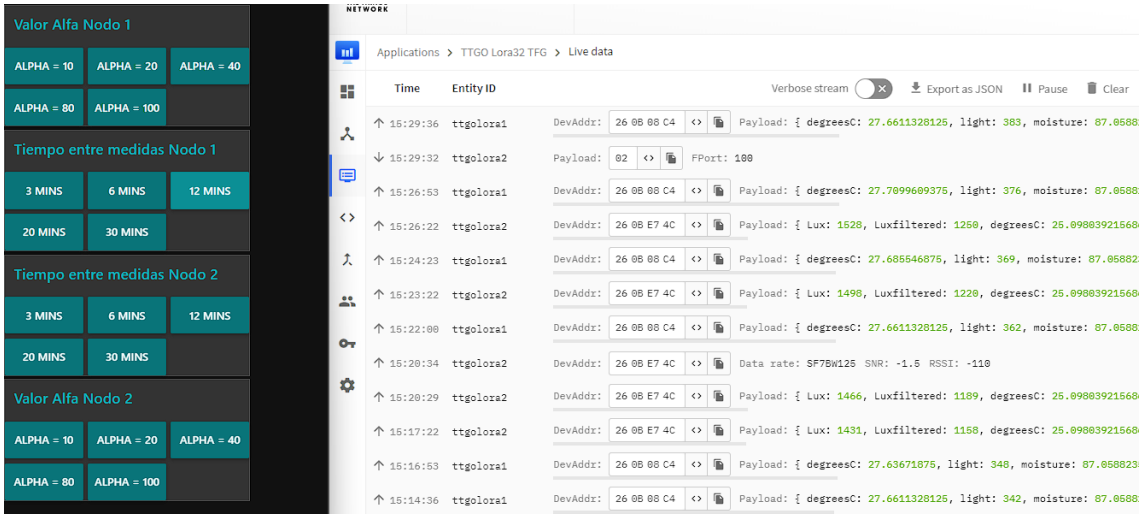

Ilustración 42 Downlink tras presionar boto en Nodo 1

Durante las distintas pruebas de vaciado del tanque y desconexión de los dispositivos para realizar mejoras se comprueba que la comunicación con Telegram está operando adecuadamente, aunque inicialmente el tiempo que se esperaba tras la desconexión era de 10 minutos, como el usuario puede cambiar el valor a través de la botonera para una lectura de 40 minutos se tuvo que cambiar pero los mensajes llegaban confirmando el correcto funcionamiento. (Ilustr. 43)

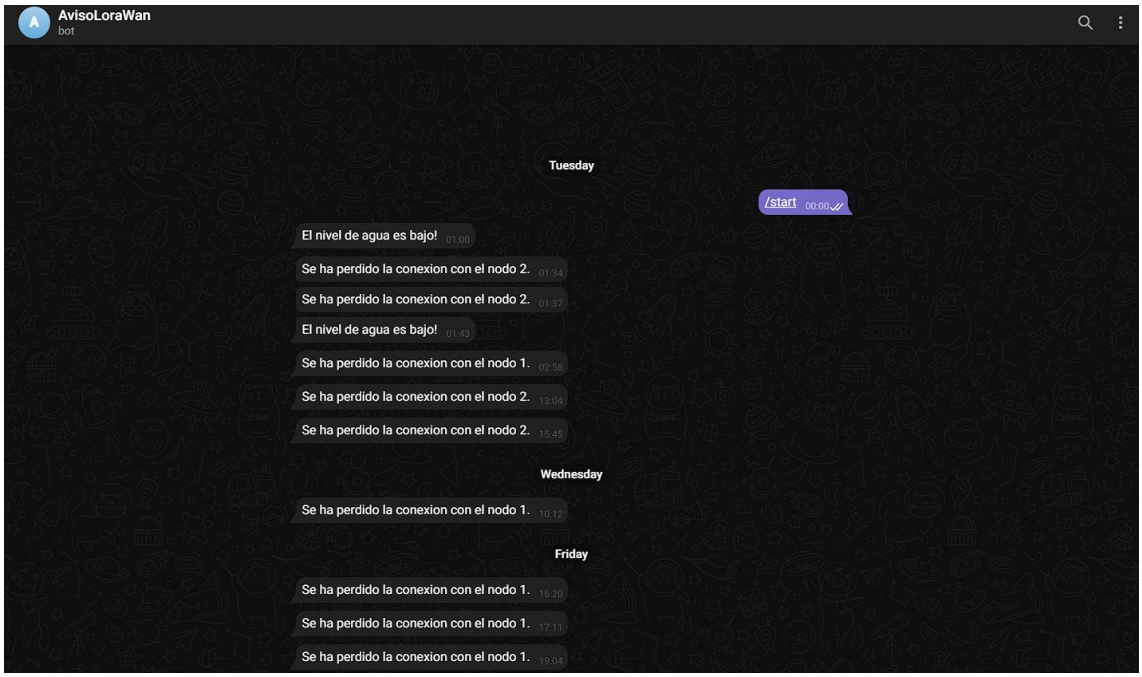

Ilustración 43 Chat con bot de Telegram enlazado con Node-RED

El funcionamiento de ambos nodos ha sido el esperado durante las distintas pruebas, si bien se esperaba que la autonomía en reposo fuera mayor de la esperada ya que con distintas baterías de Litio recargables tanto de 600 mAh, 1200mAh y 500 mAh no se lograba un uso continuo de más de un día

5. Conclusión

Tras finalizar el desarrollo principal de este proyecto es necesario valorar si los objetivos establecidos en un principio fueron suficientes, si se han llegado a cumplir y el uso de las herramientas y metodologías adoptadas han sido las adecuadas, además de tener en cuenta mejoras posibles a implementar.

5.1. Conclusiones

La implementación de un proyecto de IoT utilizando redes LoRaWAN y The Things Stack se ha demostrado factible además de económica, la mayoría de las herramientas asociadas son de código abierto y de uso gratuito existiendo la posibilidad de escalar la implementación a un entorno industrial de manera muy sencilla. El objetivo de este proyecto no solo era familiarizarse con las distintas capacidades de la comunicación LoRa y el Internet de las Cosas, sino también conseguir poner en marcha un sistema automatizado de riego de una planta lo cual se ha conseguido.

Esta solución una vez explorada se puede incrementar para aplicar en mayor escala siendo posible el control de un cultivo más grande como puede ser en un campo o en huertos urbanos, esto se ve apoyado por el bajo coste de los distintos componentes electrónicos. Con un sistema automatizado y con monitorización accesible desde cualquier red con Internet se pueden evitar la pérdida de cultivos o tener un asesoramiento de expertos de forma remota.

En conclusión, este proyecto muestra que el uso de LoRaWAN y The Things Stack puede resultar altamente beneficioso, no solo desde una perspectiva económica sino también funcional. Además de asentar la base para aplicaciones más amplias con un mayor aprovechamiento que ofrecen estas tecnologías para el Internet de las cosas.

5.2 *Conclusions*

The implementation of an IoT project using LoRaWAN and The Things Stack networks has proven to be possible and cost-effective. Most associated tools are open-source and freely available, offering the possibility of scaling the implementation to an industrial environment with ease. The aim of this project was not only to familiarize oneself with the different capabilities of LoRa communication and the Internet of Things but also to successfully launch an automated plant irrigation system.

Once explored, this solution can be expanded for larger-scale applications, such as controlling larger crops in fields or urban gardens. This is supported by the low cost of various electronic components. With an automated system and monitoring accessible from any internet-connected network, crop losses can be avoided, and remote expert advice can be obtained.

In conclusion, this project proves that the use of LoRaWAN and The Things Stack can be highly beneficial, not only from an economic perspective but also

from a functional one. Moreover, it lays the groundwork for broader applications, leveraging the potential offered by these technologies for the Internet of Things.

5.3. Líneas abiertas

Las herramientas utilizadas para el proyecto no son necesariamente las más óptimas por lo tanto está claro que existen alternativas y opciones de mejora. Se optó por Azure para la creación del servidor donde se albergaría el entorno Node-RED por la familiaridad de la plataforma, pero una opción que se barajó fue un servidor Raspberry-Pi ya que con una máquina física se tiene un mayor control y coste nulo mas allá del propio hardware.

El consumo de los nodos TTGO LoRa32 V2.0 era bastante mayor de lo esperado durante el reposo por lo tanto si se requiere autonomía estricta se debería utilizar otras placas de desarrollo como la disponible T-Beam que incluye un chip integrado de control de energía.

A partir de los datos obtenidos se puede realizar distintos análisis estadísticos y crear una base de datos con la evolución de las variables a lo largo del tiempo con lo que se llama hoy en día *Big Data*.

Las pantallas OLED de ambos nodos se utilizaron durante las distintas pruebas para mostrar valores simples de los sensores y de la señal pero en el programa final se deshabilitó ya que no se iba a consultar ningún tipo de dato ya que el objetivo era trabajar en un entorno alejado de los sensores. Esto se podría modificar con la programación de activación mediante botón para mostrar los valores durante un tiempo.

Por último los recipientes en los que se disponen los nodos son recipientes genéricos de plástico, la posibilidad de crear un prototipo en una carcasa a partir de una impresora 3D habilitaría un uso más compacto y portátil de los nodos.

5.4. Presupuesto

El coste total del desarrollo del proyecto se realizó teniendo en cuenta los materiales y software utilizado y el tiempo necesario para elaborarlo según la guía docente de la asignatura de Trabajo de Fin de Grado.

Tabla 4 Cálculo de costo total

Concepto	Coste Unitario (€)	Cantidad	Total (€)
Ingeniero Industrial	14	90	1260
Cableado	0,1	44	4,4
Dispositivos TTGO LoRa32 V2.0	17,17	2	34,34
Sensor DHT11	4,85	1	4,85
Sensor BH1750	4,5	1	4,5
Protoboards	2,95	4	11,8
Baterías externas	12	2	24
Termistor	0,95	1	0,95
Fotorresistor	1	1	1
Sensor Humedad del Suelo	2,95	1	2,95
Sensor Nivel de Agua	1,25	1	1,25
Recipientes	2,75	3	8,25
Kit panel solar	26,85	1	26,85
Servidor Azure	8	1	8
Bomba de agua y relé	6,25	1	6,25
		TOTAL	**1399,39**

5.5. Valoración personal

Escogí este proyecto porque siempre he tenido curiosidad por el mundo de la domótica que está altamente relacionado con el Internet de las Cosas. Partiendo de una base muy restringida y con poco conocimiento en The Things Network, modulación LoRa y Node-RED puedo afirmar que he terminado con una experiencia bastante positiva con ganas de seguir explorando las posibilidades que estas tecnologías brindan en un futuro no solo en mi vida laboral sino también intentando aplicar los conocimientos a mi vida diaria. Creo que este proyecto no lo habría podido realizar sin los conocimientos adquiridos en el grado ya que me he encontrado en situaciones similares a las experimentadas en las distintas asignaturas y he podido resolverlas mediante las herramientas y habilidades adquiridas. En general ha sido un proyecto muy enriquecedor que espero poder aprovechar en el futuro.

6. Referencias bibliográficas

[1] What is the Internet of Things (IoT)? oracle.com. https://www.oracle.com/internet-of-things/what-is-iot/. Accedido julio 6, 2023.

[2] LoRa and LoRaWAN: Technical overview | DEVELOPER PORTAL. semtech.com. https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan. Accedido junio 19, 2023.

[3] LoRa: Symbol Generation. http://www.sghoslya.com/p/lora-is-chirp-spread-spectrum.html. Accedido julio 8, 2023.

[4] LoRaWAN®. The Things Network. https://www.thethingsnetwork.org/docs/lorawan/. Accedido junio 19, 2023.

[5] Team H. Introducing the MQTT Protocol - MQTT Essentials: Part 1. https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/. Accedido junio 22, 2023.

[6] About : Node-RED. https://nodered.org/about/. Accedido junio 24, 2023.

[7] Lnlp. GitHub - lnlp/LMIC-node: LMIC-node. GitHub. https://github.com/lnlp/LMIC-node. Accedido junio 22, 2023.

[8] LILYGO®TTGO LORA32 V2.0. http://web.archive.org/web/20230322231435/http://www.lilygo.cn/prod_view.aspx?TypeId=50003&Id=1319&FId=t3:50003:3. Accedido junio 19, 2023.

[9] Santos S, Santos S. ESP32 with DHT11/DHT22 Temperature and Humidity Sensor using Arduino IDE | Random Nerd Tutorials. *Random Nerd Tutorials*. mayo 2019. https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-sensor-arduino-ide/.

[10] Santos S, Santos S. ESP32 with BH1750 Ambient Light Sensor | Random Nerd Tutorials. *Random Nerd Tutorials*. marzo 2022. https://randomnerdtutorials.com/esp32-bh1750-ambient-light-sensor/.

[11] DHT11 Temperature and Humidity Sensor Module Breakout. Cytron Technologies Malaysia. https://my.cytron.io/c-sensor/c-temperature-humidity-sensor/p-dht11-sensor-module-breakout. Accedido julio 7, 2023.

[12] Tutorial módulo sensor de luz BH1750. Naylamp Mechatronics - Perú. https://naylampmechatronics.com/blog/44_tutorial-modulo-sensor-de-luz-bh1750.html. Accedido junio 26, 2023.

[13] ArduinoModules. KY-013 Analog Temperature Sensor Module. *ArduinoModulesInfo*. diciembre 2021. https://arduinomodules.info/ky-013-analog-temperature-sensor-module/.

[14] ArduinoModules. KY-018 Photoresistor Module. *ArduinoModulesInfo*. diciembre 2021. https://arduinomodules.info/ky-018-photoresistor-module/.

[15] TP4056: Your Essential guide to the LiPo Battery Charger IC. *Best Microcontroller Projects*. https://www.best-microcontroller-projects.com/tp4056.html. Accedido junio 27, 2023.

[16] SRS Thermistor Calculator. https://www.thinksrs.com/downloads/programs/Therm%20Calc/NTCCalibrator/NTCcalculator.htm. Accedido junio 28, 2023.

[17] Design a Luxmeter Using a Light Dependent Resistor - Projects. https://www.allaboutcircuits.com/projects/design-a-luxmeter-using-a-light-dependent-resistor/. Published 28 de junio de 2021.

[18] Martinsen P. Exponential Filter. *MegunoLink*. septiembre 2020. https://www.megunolink.com/documentation/arduino-libraries/exponential-filter/.

[19] Cayenne Low Power Payload | myDevices Documentation. https://docs.mydevices.com/docs/lorawan/cayenne-lpp. Accedido junio 24, 2023.

[20] Running on Microsoft Azure : Node-RED. https://nodered.org/docs/getting-started/azure. Accedido junio 24, 2023.

[21] Node-RED. The Things Stack for LoRaWAN. https://www.thethingsindustries.com/docs/integrations/node-red. Accedido junio 25, 2023.

[22] Working with messages : Node-RED. https://nodered.org/docs/user-guide/messages. Accedido junio 25, 2023.

[23] node-red-contrib-telegrambot. https://flows.nodered.org/node/node-red-contrib-telegrambot. Accedido julio 3, 2023.

7. Anexos

https://github.com/AlejandroSantanaPerez/Lora32-TFG Enlace a GitHub con el código tanto de los nodos como del flujo Node-RED
http://tfglora.westeurope.cloudapp.azure.com:1880/ui/ Enlace a la página web con la interfaz visual
https://drive.google.com/drive/folders/19AQYylMP34euomQj2c0GQkEDluyN9z6Y?usp=sharing Enlace al archivo Excel utilizado para la parametrización del fotorresistor y archivo con el flujo principal utilizado en la programación de Node-RED