Trabajo Fin de Máster

Máster en Ingeniería de Telecomunicación

Diseño e implantación de red telemática de sensores inalámbricos para optimización de riego

Autor: Jose Pablo Villén Macías

Tutor: Jose Manuel Fornes Rumbao

Dep. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Trabajo Fin de Máster
Máster en Ingeniería de Telecomunicación

Diseño e implantación de red telemática de sensores inalámbricos para optimización de riego

Autor:
Jose Pablo Villén Macías

Tutor:
Jose Manuel Fornes Rumbao

Dep. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Trabajo Fin de Máster: Diseño e implantación de red telemática de sensores inalámbricos para optimización de riego

Autor: Jose Pablo Villén Macías

Tutor: Jose Manuel Fornes Rumbao

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

Resumen

Al igual que Internet conecta a personas de todo el mundo, el Internet de las Cosas conecta cualquier objeto con la nube. Un ámbito en el que se puede implantar esta tecnología es el riego. El agua como recurso se está malgastando diariamente y está en nuestras manos arreglarlo. Invertir en el riego inteligente no solo aumenta notablemente la producción en el sector agrícola, sino que además cuida del medio ambiente.

Se desarrolla un sistema de riego inteligente, con una red de nodos que se comunican mediante el protocolo LoRaWAN enviando la humedad del suelo al servidor. Éste se encarga de procesar los datos en la nube y decide si el sistema de riego debe encenderse o apagarse, mandando dicha decisión de vuelta a los nodos.

En concreto, se realiza el diseño de la electrónica que compone a los nodos sensor y actuador y se implementa el código incrustado en ellos. De lado del servidor, se instalan la puerta de enlace y todos los servicios necesarios haciendo uso de una Raspberry pi3. Se implementa además un flujo de control en el servidor de aplicación para la toma de decisiones y se desarrolla un cuadro de mando para visualizar el sistema.

Palabras clave: IoT, LoRa, LoRaWAN, RAK, ChirpStack, Thingsboard, Raspberry pi3, sensor

Abstract

Just as the Internet connects people around the world, the Internet of Things connects any object to the cloud. One area where this technology can be implemented is in irrigation. Water as a resource is being wasted daily and it is in our hands to fix it. Investing in intelligent irrigation not only significantly increases production in the agricultural sector, but also takes care of the environment.

An intelligent irrigation system has been developed, with a network of nodes that communicate through the LoRaWAN protocol sending the soil moisture to the server. The server processes the data in the cloud and decides if the irrigation system should be turned on or off, sending the decision back to the nodes.

In particular, the design of the electronics that compose the sensor and actuator nodes is carried out and the code embedded in them is implemented. On the server side, the gateway and all the necessary services are installed using a Raspberry pi3. A control flow is also implemented in the application server for decision making and a dashboard is developed to visualize the system.

Key words: IoT, LoRa, LoRaWAN, RAK, ChirpStack, Thingsboard, Raspberry pi3, sensor

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

%	Porcentaje
ºC	Grados centígrados
€	Euro
Kbps	Kilobits por segundo
A	Amperios
mA	miliAmperios
µA	microAmperios
ms	milisegundos
mAh	miliAmperios hora
Ω	Ohmios
W	Vatios
µF	microFaradios
V	Voltios
~	Aproximadamente
kHz	kiloHercios
MkHz	megaHercios
GHz	gigaHercios
dBm	Decibelio-milivatio
B	Bytes
GB	gigaBytes
b	Bits
mm	Milímetros
Km	Kilómetros

1 INTRODUCCIÓN

Internet no se acaba, el agua sí

En este primer capítulo entenderemos el motivo por el cual se realiza el proyecto, así como el objetivo a llegar en el mismo. Esto conlleva a dividir el proceso en diferentes tareas específicas que en su conjunto conformarán un sistema de riego automático controlado desde la nube.

1.1. Motivación

A finales de los años 90, Internet se usaba principalmente como herramienta para buscar información. En los últimos 10 años se ha vivido una nueva forma de uso de Internet, donde todo se ha convertido en social, transaccional y móvil. Después de la red de redes (*World Wide Web*, WWW) y del Internet móvil, la sociedad está inmersa en una nueva y potencialmente más disruptiva fase, el llamado Internet de las Cosas (*Internet of Things*, IoT).

Un gran campo de acción en el que IoT puede ser aplicado es el sector agrícola, creando así la agricultura de precisión, que consiste en aplicar las herramientas que hoy en día brindan las tecnologías de la información y comunicaciones (TIC) en conjunto con diferentes sensores y herramientas de IoT que gestionan el terreno teniendo en cuenta las variables y propiedades que son importantes a fin de mejorar el rendimiento de los cultivos, la rentabilidad y la calidad del medio ambiente.

En los últimos 100 años la población mundial se ha triplicado, pero el uso del agua para fines humanos se ha sextuplicado [1]. En la actualidad existe una crisis del agua, pero ésta no radica en que sea insuficiente para satisfacer las necesidades humanas, sino al mal manejo que se le brinda, haciendo que el medio ambiente y miles de millones de personas alrededor del mundo sufran las consecuencias.

Según la Organización de las Naciones Unidas para la Alimentación y la Agricultura (ONUAA), la agricultura representa un 70% del uso de agua potable en el mundo, siendo la actividad que requiere mayor demanda de la misma. Además, gran parte del agua se malgasta o se desperdicia por fenómenos naturales o ambientales como el efecto del sol o la percolación profunda. Por lo que la innovación en tecnología en los sistemas de riego se ha convertido en una prioridad en los tiempos modernos, con el fin de volver más eficiente la producción agrícola y, al mismo tiempo, ahorrar nuestro más valioso recurso natural.

1.2. Objetivos

Se establece como objetivo general el diseño y montaje de un sistema de riego que sea capaz de comunicarse con un servidor a través de internet, tomando la decisión de encendido o apagado del sistema en base al nivel de humedad que haya en la superficie plantada, teniendo además en cuenta las condiciones climatológicas.

Para poder cumplir con dicho objetivo, se dividirá en tres secciones diferenciadas que a su vez se desglosan en tareas más específicas:

- **Base teórica**

    - Analizar las tecnologías inalámbricas usadas en IoT y elegir la apropiada para la red de nodos
    - Estudiar profundamente la tecnología LoRa en todos sus ámbitos
    - Analizar los servidores LoRa del mercado y elegir el apropiado para el proyecto
    - Elegir un servidor de aplicación para representar los datos
    - Ilustrar la topología a nivel de campo que pudiera tener el sistema
    - Analizar los componentes electrónicos necesarios y anotar sus especificaciones

- **Desarrollo**

    - Conectar la raspberry con el módulo RAK831
    - Configurar el gateway
    - Instalar los servicios necesarios, así como la integración entre ellos
    - Analizar, diseñar y construir la circuitería de cada nodo del sistema (sensor y sensor/actuador)
    - Crear el código basado en el framework RUI para los nodos sensor y sensor/actuador
    - Enviar un mensaje uplink de nodo a servidor
    - Crear un flujo de control en el servidor de aplicación
    - Realizar peticiones a las APIs de ChirpStack y openweather usando REST
    - Enviar un mensaje downlink de servidor a nodo
    - Simular un nodo LoRa clase B con uno de clase A y un mensaje dummy
    - Medir la humedad del suelo con el sensor y calibrarla con distintas superficies
    - Estabilizar la medida del sensor con un regulador
    - Medir el pocentaje de batería de la electroválvula
    - Encender o apagar la electroválvula en base al mensaje downlink
    - Utilizar un puente H para polarizar el impulso de la electroválvula
    - Minimizar el consumo del sistema
    - Crear un dashboard o cuadro de mando para visualizar el estado del sistema

- **Pruebas**

    - Probar cada funcionalidad de los nodos bajo demanda con el puerto serie y un ordenador
    - Examinar los mensajes uplink y downlink a nivel de trama (cruzar con conceptos teóricos)
    - Hacer debug al gateway por SSH y a los eventos del motor de reglas en el servidor
    - Medir el consumo y calcular la vida útil aproximada del sistema
    - Probar la electroválvula

2 ESTADO DEL ARTE

Solo evolucionamos cuando nos comunicamos

A lo largo de este capítulo se explicarán los conceptos teóricos necesarios para la realización del proyecto. Estudiaremos el concepto de internet de las cosas, así como las tecnologías vanguardistas que lo persiguen. Elegiremos la tecnología LoRa para abordar la tarea inalámbrica de la red de nodos y estudiaremos en detalle dicha tecnología, desde sus aspectos físicos hasta su nivel más alto de capa, con el objetivo de suavizar los problemas con los que nos podamos encontrar a la hora de su implementación.

A continuación, se compararán los dos servidores de red más comunes y nos decantaremos por ChirpStack, explicando sus características, así como los elementos que componen su arquitectura.

Por último, se explican las características de ThingsBoard, el servidor de aplicación que se usará en el proyecto. Desde aquí se tomará la decisión de encender o apagar el sistema de riego.

2.1. El Internet de las cosas

El término "Internet de las cosas", también conocido como IoT (Internet of Things), fue introducido por primera vez por *Kevin Ashton* en 1999 desde el Instituto Tecnológico de Massachusetts (MIT), investigando las tecnologías de identificación por radiofrecuencia (RFID).

"Internet de las cosas tiene el potencial de cambiar el mundo, como hizo Internet en su momento. Tal vez aún más." – Kevin Ashton [2].

Pero el nacimiento oficial de IoT, según Cisco System y su Grupo de Soluciones Empresariales Basadas en Internet (IBSG, *Internet Business Solutions Group*), fue en el momento del tiempo en el que se conectaron a Internet más cosas que personas habitaban el mundo. Así, se estableció la fecha en algún momento entre 2008 y 2009 [Fig. 1].

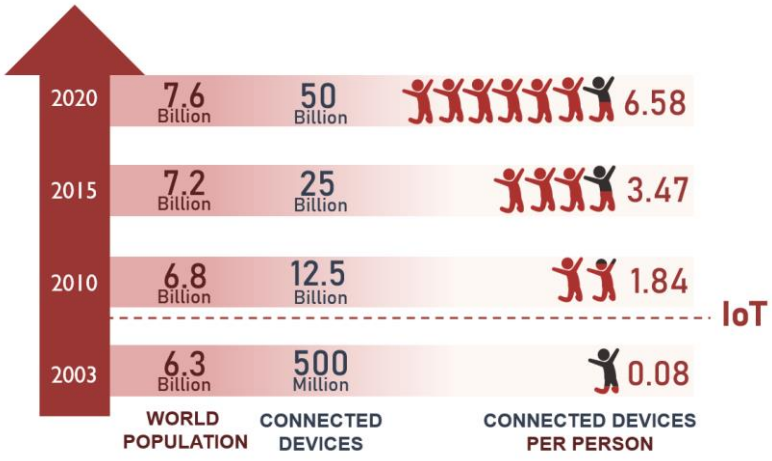


Fig. 1 - Nacimiento de IoT

En términos generales, el Internet de las cosas es la red de objetos físicos que se conectan a Internet usando diversas tecnologías y que tienen capacidades de conexión e interacción con el entorno, capacidades que les permiten tomar decisiones y comunicarse con el mundo.

Según Cisco Systems, la evolución de Internet se divide en cuatro fases distintas, teniendo cada una de ellas un efecto mayor que la anterior sobre los negocios y la sociedad en general:

1. **Conectividad**: empezó a principio de los años 90, y se democratizó y expandió gracias al uso masivo del correo electrónico, la navegación web y los motores de búsqueda.
2. **Transformación del proceso empresarial**: esta fase se inició a finales de los 90 e implicó la conexión digital de los sistemas logísticos, suponiendo el inicio del comercio electrónico y la forma en que las empresas acceden a nuevos mercados.
3. **Digitalización colaborativa**: la tercera fase empezó a principios de la década de 2000 y se caracterizó por el amplio uso de los medios sociales, la movilidad, los servicios de vídeo y audio en línea y el cloud computing (computación en la nube).
4. **Internet de las cosas**: es la cuarta fase, donde se conectan personas, procesos, datos y objetos, lo que permite transformar la información en decisiones y acciones que crean nuevas posibilidades y experiencias.

Adquiere pues una gran importancia ya que se trata de la fase actual de la evolución de internet, en la que las aplicaciones tienen el potencial de mejorar la manera en que las personas viven, aprenden, trabajan y se entretienen. IoT ha logrado que Internet sea sensorial (temperatura, presión, humedad, luz, vibración, estrés), lo que nos permite ser más proactivos y menos reactivos.

Existen tres tecnologías que han hecho posible el asentamiento del Internet de las cosas en la sociedad [3]:

- **Sensores**: Es el elemento hardware que interactúa entre nuestra tecnología y el entorno, capturando los datos que nosotros deseemos.
- **Nuevos procesadores**: Uno de los requisitos de IoT es que los dispositivos sean pequeños y de bajo consumo, primando más esto que su capacidad de cómputo. Hablamos de la familia Cortex de ARM, el procesador Quark de Intel o el propio Arduino, que permite realizar proyectos de IoT fácilmente con conocimientos básicos de electrónica y programación.
- **Comunicación de bajo consumo**: Se encarga de transportar por un canal de comunicación esas pequeñas cantidades de datos que hemos obtenido del medio, a otros dispositivos lejanos manteniendo un bajo consumo eléctrico. Estamos hablando de las redes LPWAN (Low Power Wide Area Network) [Fig. 2].

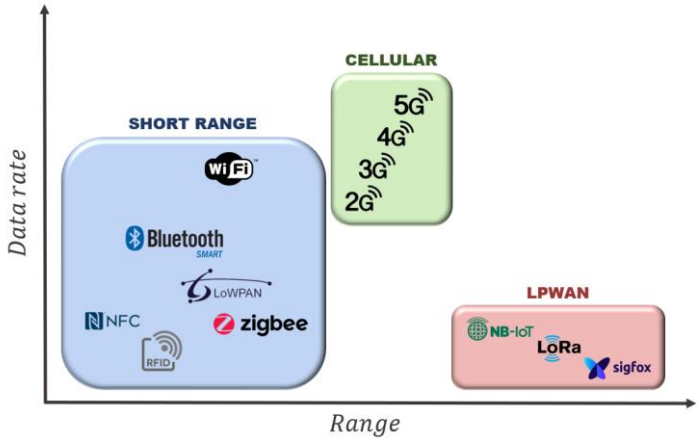


Fig. 2 – Protocolos de redes inalámbricas

2.2. Tecnologías de comunicación para IoT

El número de dispositivos conectados sigue aumentando (y se espera que alcance los 125.000 millones para 2030), por lo que las tecnologías inalámbricas maduras que los soportan también siguen recibiendo una gran atención en todo el mundo. NB-IOT (Narrowband IoT), LoRa y Sigfox son las puntas de lanza de todas las tecnologías de red de área amplia (LPWAN) de bajo consumo.

Cada una de estas tecnologías desempeñará un papel importante en el espacio de la IoT dependiendo del caso de uso, por lo que es fundamental comprender las características y diferencias de cada una [4].

2.2.1 NB-IoT

NB-IoT es la iniciativa del 3GPP, la organización que está detrás de la estandarización de los sistemas de tecnología celular (móviles), para abordar las necesidades de los dispositivos de muy baja velocidad de datos que necesitan conectarse a redes móviles, a menudo alimentadas por baterías. Como estándar celular, el objetivo de NB-IoT es estandarizar los dispositivos de IOT para que sean interoperables y más fiables.

Debido a que NB-IoT es una tecnología inalámbrica de grado móvil que utiliza modulación OFDM, los chips son más complejos, pero los presupuestos de enlace son mejores. Esto significa que los usuarios obtienen el alto nivel de rendimiento asociado a las conexiones celulares, pero a costa de una mayor complejidad y un mayor consumo de energía.

2.2.1.1 Ventajas

- Cobertura muy buena en entornos urbanos. Los dispositivos NB-IoT dependen de la cobertura 4G, por lo que funcionarían bien en interiores y en áreas urbanas densas.
- Tiene tiempos de respuesta más rápidos que LoRa y puede garantizar una mejor calidad de servicio.

2.2.1.2 Desventajas

- Algunas de las especificaciones de diseño de NB-IoT hacen que sea difícil enviar grandes cantidades de datos a un dispositivo.
- Es más adecuado para activos principalmente estáticos, como medidores y sensores en una ubicación fija, en lugar de activos en itinerancia.

2.2.2 LoRa

LoRa es una tecnología de modulación no celular para LoRaWAN. (Al igual que BPSK o QPSK es la modulación de NB-IoT.) Esos dos términos LoRa y LoRaWAN no son intercambiables: LoRaWAN es el protocolo estándar para las comunicaciones WAN y LoRa se utiliza como tecnología de red de área extendida.

LoRa representa una buena red radioeléctrica para las soluciones de IoT y tiene mejores presupuestos de enlace que otras tecnologías de radiocomunicaciones comparables. Pero fuera de unos pocos mercados en Europa, si desea conectarse a las redes LoRaWAN, se necesita desplegar *gateway* de red propio.

Esto puede parecer un inconveniente, pero en realidad hace de LoRa una buena alternativa a WiFi para los dispositivos de bajo consumo que necesitan conectarse en un área unificada, como un edificio o una superficie continuada. Además, la configuración de su propio gateway crea una red completamente separada y segura.

LoRaWAN utiliza espectro sin licencia. En Europa esto significa un ciclo de servicio del 1%, que limita el volumen y la frecuencia del tráfico, así como la capacidad de la estación base para controlar la red y enviar el tráfico hacia abajo.

2.2.2.1 Ventajas

- Es perfecto para aplicaciones que se encuentran dentro de un área unificada.
- Puede configurar y gestionar su propia red.
- Es una buena opción si necesita bidireccionalidad, por ejemplo, funcionalidad de mando y control, debido a la conexión simétrica.
- Los dispositivos LoRa funcionan bien también cuando están en movimiento, lo que los hace útiles para rastrear activos como los envíos.
- Tienen una mayor duración de la batería que los dispositivos NB-IoT.

2.2.2.2 Desventajas

- Tiene velocidades de datos más bajas que NB-IoT.
- Tiene un tiempo de latencia más largo que NB-IoT.
- Requiere una puerta de entrada (lo que también, en muchos casos, es una ventaja).

2.2.3 Sigfox

Sigfox es la compañía que despertó el potencial de los dispositivos IoT para usar conexiones de muy bajo ancho de banda. Sigfox es la más básica de las tres tecnologías, con las diferencias clave:

- Tiene los módulos de radio de menor costo (<€5, comparado con ~€10 para LoRa y €12 para NB-IOT).
- Es sólo un enlace ascendente. Aunque es posible un enlace descendente limitado, tiene un presupuesto de enlace diferente y es muy restringido.
- Es un reproductor de red y tecnología de extremo a extremo.

Su modelo de negocio es un tanto ambicioso, ya que la compañía ya ha gastado cientos de millones de dólares para desplegar una red a la que la gente está pagando para poder acceder. Sigfox piensa que mantener el costo de la aplicación a un nivel bajo es la manera de llevar a la gente a su mercado, pero esta línea de pensamiento ha hecho que sea difícil obtener suficientes ingresos. Al final, parece que Sigfox está teniendo dificultades en sus esfuerzos en establecerse como operador de una red celular alternativa para dispositivos de IoT.

2.2.3.1 Ventajas

- Consume una baja cantidad de energía.
- Funciona bien para dispositivos sencillos que transmiten con poca frecuencia, porque envía cantidades muy pequeñas de datos muy lentamente.
- Soporta una amplia zona de cobertura en las zonas donde se ubica.

2.2.3.2 Desventajas

- No está desplegado en todas partes, por lo que no funcionará en un gran número de casos de uso en la actualidad.
- La comunicación se dirige mejor desde el punto final a la estación base. Tiene funcionalidad bidireccional, pero su capacidad desde la estación base hasta el punto final es limitada.
- La movilidad es difícil con los dispositivos Sigfox.

Debido a su bajo consumo, bidireccionalidad, independencia con operadores de red, buen funcionamiento dentro un área unificada y posibilidad de disponer de una red aislada y segura, se utilizará la tecnología LoRa como protocolo de comunicación para la red de nodos del proyecto.

2.3. Riego inteligente

Consiste en la utilización de las Tecnologías de la Información y la Comunicación (TICs) para realizar una gestión optima del riego, utilizándose de forma más eficiente los recursos productivos (agua, energía y fertilizante) en las fincas de cultivo e incrementando las producciones por el uso más eficiente de estos recursos, es decir, se produce más con menos [5].

Las decisiones en el Riego Inteligente están basadas en la monitorización y adquisición de datos (datos climáticos, humedad del suelo, fertilización, consumos de agua, fertilizante y energía, imágenes), procesamiento de esos datos (modelización, simulación y predicción) y representación de la información.

Se realiza una programación óptima del riego para ahorrar agua, estableciendo el momento, la frecuencia y el tiempo de riego adecuados según las características del cultivo, la configuración de la red de riego, el clima y suelo de la finca, dando de esta forma el agua que necesita la planta en el momento adecuado.

La programación del riego debe implicar tanto el control de funcionamiento del sistema de riego como la distribución de la humedad en el suelo. Requiere una sectorización adecuada de la red de riego (unidades de riego homogéneas). Por otro lado, la variable clave en el manejo del riego, el tiempo, se debe controlar mediante la correspondiente automatización (programador electrónico y electroválvulas), que facilita la gestión y el ahorro en los costes de operación.

El futuro de la agricultura de regadío a nivel mundial depende, en buena parte, de la implantación de sistemas de riego inteligente en las fincas de cultivo, que permitan la utilización más eficiente de los recursos de agua, fertilizante y energía de manera que se aumenten los niveles de producción utilizando menos recursos productivos. Con el riego inteligente se incrementa la rentabilidad de las explotaciones y se minimiza el impacto ambiental de esta actividad al disminuir tanto el uso del agua como la aportación de elementos contaminantes al entorno. La implantación de sistemas de riego inteligente es fundamental para garantizar la sostenibilidad de la agricultura de regadío.

2.3.1 Tecnología LoRa en el riego inteligente

Los dispositivos LoRa y su tecnología de radiofrecuencia inalámbrica (LoRaWAN) están haciendo que sea fácil y asequible para los sistemas de gestión del riego transformar las superficies plantadas de todo el mundo [6].

Hasta ahora, los cultivadores no tenían una forma eficaz de medir la humedad del suelo, que no fuera la inspección visual. La capacidad de detectar los daños de las plantas afectadas y de hacer los cambios necesarios en el riego a menudo llegaba demasiado tarde y provocaba la pérdida de la cosecha.

La aparición del Internet de las cosas evita las consecuencias irreversibles de una mala gestión del riego al proporcionar a los agricultores una vigilancia en tiempo real de la humedad del suelo. El sistema de gestión del riego permite detectar los niveles de agua y automatizar los controles de las válvulas de riego.

La práctica de décadas de un ciclo de riego semanal y largo de los cultivos está siendo sustituida por ciclos de riego frecuentes y más cortos para alcanzar niveles óptimos.

La tecnología LoRa proporciona a los cultivadores datos procesables, permitiendo así una gestión más eficiente del agua, un crecimiento optimizado de las plantas, un aumento del rendimiento de los cultivos y un ahorro de hasta el 50% en el uso del agua de riego.

Una vez desplegados en un campo agrícola, los sensores de humedad del suelo y la plataforma de la estación de sensores utilizan el protocolo abierto LoRaWAN para transmitir datos de humedad del suelo a una aplicación de gestión basada en la nube. La aplicación transmite información procesable que es accesible para los agricultores a través de un cuadro de mando. El sistema calcula y muestra las necesidades de agua y puede activar los controladores de las válvulas de riego para liberar por intervalos el agua automáticamente y así satisfacer con precisión los niveles de humedad del suelo.

2.4. Tecnología LoRa

Cuando hablamos de LoRa en realidad debemos hacer referencia a dos tecnologías diferentes: LoRa y LoRaWAN.

La primera de ellas es una tecnología propietaria del fabricante Semtech que define la conectividad a nivel de capa 1 en el modelo OSI. Esta tecnología inalámbrica está diseñada para permitir la interconexión de dispositivos de recursos muy reducidos (energía, memoria y procesador) y a velocidades muy bajas (hasta 37.5 Kbps). El nombre LoRa deriva de las palabras "*Long Range*" (largo alcance).

Por otro lado, LoRaWAN es un protocolo de capa 2 promovido por LoRa Alliance, una asociación sin fines de lucro con la finalidad de desarrollar y promover la tecnología de manera estandarizada e interoperable. La asociación está compuesta por más de 500 empresas que desarrollan activamente soluciones y equipamiento relacionado.

2.4.1 LoRa

LoRa es una tecnología de comunicación inalámbrica para dispositivos que requieren bajo consumo de energía y admiten un bajo *bit rate*. Está diseñada como solución de infraestructura para IoT, en la que los nodos se comunican directamente a un *gateway*, el cual tiene un rol de "pasarela" entre los nodos y otros sistemas [7].

Utiliza un sistema de modulación propietario y típicamente tiene radios de cobertura del orden de 3 km en áreas urbanas y 10 km en áreas rurales [Tabla 1].

Entorno	Alcance (km)
Espacios urbanos	2-5
Espacios rurales	5-15
Línea de vision directa	>15

Tabla 1 – Alcance inalámbrico LoRa

LoRa no provee funcionalidades de seguridad, la tecnología sólo resuelve la transmisión, y no garantiza la confidencialidad, autenticidad ni integridad de la información por lo que estos mecanismos se deben implementar en las capas superiors (LoRaWAN).

2.4.1.1 Modulación

LoRa utiliza modulación de radio basada en Chirp Spread Spectrum (CSS) o esquema de modulación de espectro ensanchado. Dicha modulación utiliza pulsos chirp o "chirridos" modulados en frecuencia para codificar la información, ensanchando la señal deliberadamente en el dominio de la frecuencia. Esto se realiza transmitiendo la señal en ráfagas cortas, saltando entre las frecuencias siguiendo una secuencia pseudo-aleatoria, creando diferentes símbolos.

Cuando el *chirp* es un tono en el cual la frecuencia aumenta se denomina *up-chirp* y cuando disminuye se denomina *down-chirp* [Fig. 3].

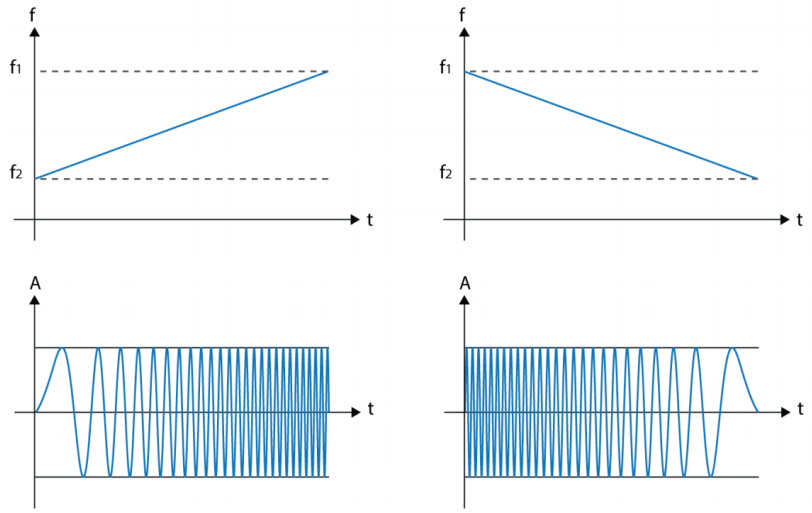


Fig. 3 – Upchirp (izquierda) y downchirp (derecha)

En la siguiente figura [Fig. 4], se aprecia cómo la frecuencia varía linealmente en el tiempo y realiza saltos de frecuencia para representar diferentes símbolos, vemos también cómo el chirrido cubre nominalmente todo el ancho de banda ($BW = f_{high} - f_{low}$) una vez durante el tiempo de un símbolo.

La cantidad de bits que se pueden codificar por símbolo viene dada por un parámetro ajustable llamado *Spreading Factor* (SF), por lo cual, si éste tiene un valor de N, el símbolo representa N bits y puede tener 2 elevado a la N posibles valores de frecuencia a lo que puede saltar.

Debido a que tiene un patrón muy distintivo, el receptor LoRa puede detectar chirridos incluso por debajo del nivel del ruido. Esta característica permite demodular muchas señales con diferentes factores de dispersión al mismo tiempo aprovechando su ortogonalidad.

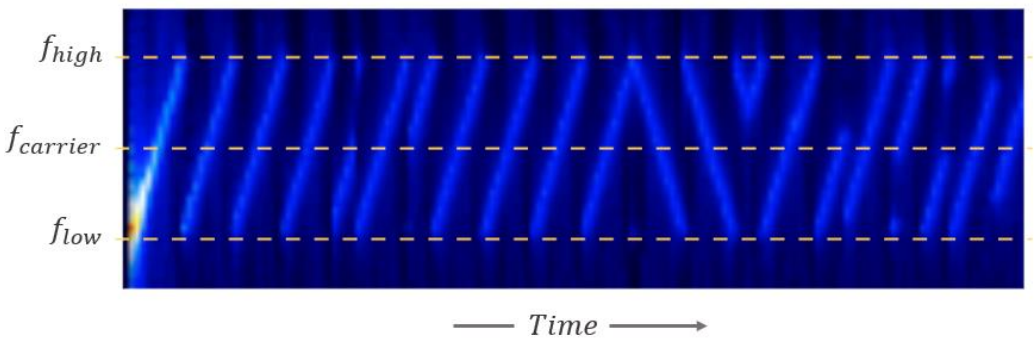


Fig. 4 – Señal modulada LoRa

Al igual que FSK, LoRa se clasifica como una modulación de envolvente constante, lo que significa que su demodulación es sencilla y se pueden reutilizar amplificadores con ganancia programable de bajo costo sin hacerles ninguna modificación.

2.4.1.2 Parámetros

Hay tres parámetros principales que pueden ser modificados en la modulación LoRa: ancho de banda (BW, *bandwidth*), factor de dispersión (SF, *spreading factor*) y ratio de codificación (CR, *code rate*). Es posible optimizar la modulación LoRa para una aplicación determinada variando estos tres parámetros, esta propiedad se conoce como *Adaptative Data Rate* (ADR). Estos parámetros determinan el *bitrate*, la inmunidad a la interferencia y el tiempo que se ocupa el canal en cada transmisión.

- **Factor de dispersión**: Se define como logaritmo en base 2 de la cantidad de "chirps" por símbolo. Cada bit de información (*payload*) es representado por múltiples "chips" de información. El ratio en el que se envía la información dispersa se refiere al ratio de símbolo (Rs), por lo tanto la relación entre el ratio de símbolo nominal y el ratio de chips es el factor de dispersión.

  Este factor incide en el *bitrate* efectivo y la inmunidad a la interferencia por ruido en la detección. Los valores posibles varían de 7 a 12. Un aumento del SF repercute positivamente en la sensibilidad del receptor, pero aumenta el tiempo del símbolo (en consecuencia, se ocupa más tiempo el canal) [Fig. 5].

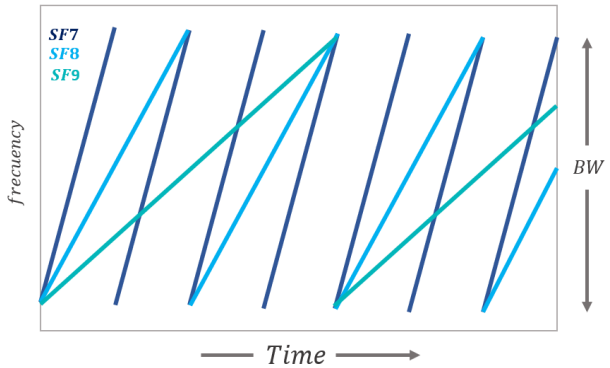


Fig. 5 – Aumentar el SF implica aumentar el Ts

  El Spreading Factor debe ser el mismo tanto para el emisor como por el receptor. Los distintos valores de SF son ortogonales entre ellos, por lo tanto, es posible reutilizar el mismo canal para distintos nodos de manera simultánea variando el SF.

- **Ancho de banda:** Los anchos de banda típicos utilizados son de 125, 250 y 500 kHz. Este es el ancho de banda que se ocupa en el espectro radioeléctrico durante la transmisión. Es posible utilizar anchos de bandas menores, pero en este caso Semtech recomienda utilizar un reloj externo (TXCO).

  Aumentar el *bandwith* hace la comunicación más rapida, ya que disminuye el tiempo de bit (aumenta la capacidad de canal *Sharon-Hartley*).

  Un aumento de uno en el *Spreading Factor* divide la frecuencia del *chirp* en 2. Dado que hay 2^{SF} "chips" en un símbolo, un símbolo puede codificar SF *bits* de información.

  Por lo tanto, el *symbol rate* (velocidad de símbolo) y el *bitrate* son dados por el SF, los cuales son proporcionales al BW. Queda expresado en la siguiente ecuación, siendo Ts el tiempo de duración de un símbolo para un determinado SF y BW:

$$T_s = \frac{2^{SF}}{BW}$$

- **Ratio de codificación:** LoRa incluye un FEC (*Forward Error Correction*), denominado CR (*Code Rate*) el cual puede tomar los siguientes valores: 4/5, 4/6, 4/7 y 4/8. El primer valor indica cuántos bits son de información (*payload*) y el segundo los bits enviados.

- **Bitrate:** Es la capacidad de la comunicación o tasa de datos. Utilizando la ecuación anterior y sabiendo que SF bits de información son enviados por cada símbolo se puede obtener la ecuación del *bitrate* efectivo de la modulación con los tres parámetros configurables.

$$R_b = SF \times \frac{BW}{2^{SF}} \times CR$$

Para un bandwith de 125KHz y un coding rate de 4/5, obtenemos las siguientes tasas de datos en el sistema [Tabla 2]:

SF	Rb (kbits/s)	SF	Rb (kbits/s)
7	5.5	10	0.98
8	3.13	11	0.54
9	1.76	12	0.29

Tabla 2 – Tasa de datos según el SF

- **Sensibilidad:** Es la mínima señal que un receptor puede decodificar. La modificación de los tres parámetros anteriores influye en la sensibilidad del decodificador, así como en el tiempo de transmisión que se utiliza para enviar la información. Por lo tanto, aumentar el ancho de banda empeora la sensibilidad del receptor (aumenta su valor). En cambio, aumentando el *Spreading Factor,* la sensibilidad del receptor mejora (disminuye su valor) [Fig. 6]. Por otro lado, disminuyendo el *Code Rate* se reduce el ratio de paquetes con error en presencia de pequeñas ráfagas de interferencia, pero el *bitrate* efectivo disminuye.

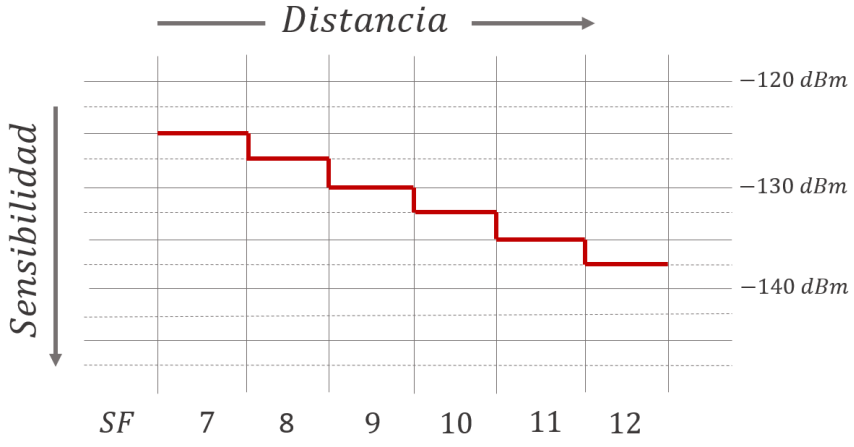


Fig. 6 – Aumentar SF mejora la Sensibilidad

2.4.1.3 Espectro

LoRa utiliza las bandas de radio industriales, científicas y médicas (ISM) por debajo de 1 GHz (433, 868 y 915 MHz según la versión del chip utilizado y la región donde nos ubicamos) [8]. En Europa, las frecuencias de enlace ascendente utilizadas según el canal son aquellas que se indican en la [Tabla 3] y las de enlace descendente en la [Tabla 4]. El nodo final cambia de canal de forma aleatoria para cada transmisión, haciendo así al sistema más robusto frente a interferencias.

Canal	Frec. uplink (MHz)	Spreading Factor & BW
0	868.1	SF7 BW125 - SF12 BW125
1	868.3	SF7 BW125 - SF12 BW125 / SF7 BW250
2	868.5	SF7 BW125 - SF12 BW125
3	867.1	SF7 BW125 - SF12 BW125
4	867.3	SF7 BW125 - SF12 BW125
5	867.5	SF7 BW125 - SF12 BW125
6	867.7	SF7 BW125 - SF12 BW125
7	867.9	SF7 BW125 - SF12 BW125
8	868.8	FSK

Tabla 3 – Frecuencias de enlace ascendente

Frec. downlink	Spreading Factor & BW
	Mismos canales que en uplink (RX1)
869.525	SF9 BW125 (RX2)

Tabla 4 - Frecuencias de enlace descendente

2.4.1.4 Estructura del paquete LoRa

La estructura del paquete LoRa es la siguiente [Fig. 7]:

- **Preámbulo**: Usado para detector el comienzo del paquete por el receptor (sincronizarse con el transmisor). En Europa consta de 8 símbolos lineales repetidos.
- **Cabecera**: Indica el modo de operación (longitud de trama en *bytes*, *code rate*, presencia de 16 bits opcionales en CRC o *Cyclic Redundancy Check*). La cabecera puede ser obviada siempre que dichos parámetros estén configurados en ambos extremos.
- **Carga**: Campo de longitud variable que contiene los datos codificados con FEC. Se pueden añadir hasta 16 bits de redundancia (CRC) opcionales.

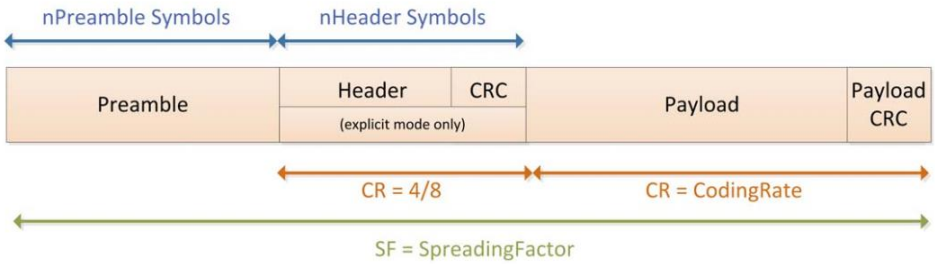


Fig. 7 – Estructura de paquete LoRa

2.4.1.5 Time on Air

El *Time on Air* (ToA) también llamado duración de paquete (T_{packet}) es el tiempo que tarda el transmisor en enviar el paquete, durante el que se ocupa el canal de radio (no es el tiempo desde el transmisor al receptor).

$$ToA = T_{packet} = T_{preamble} + T_{payload}$$

Es posible calcular el tiempo en el aire de nuestro sistema indicando el valor de los tres parámetros ajustables anteriores. El propio fabricante ofrece una herramienta en línea para dicho cálculo llamada *LoRa Modem Calculator tool* [9].

Para un ancho de banda de 125KHz, un *coding rate* de 4/5, un preámbulo de 8 *bytes* y una carga de 6 *bytes* (parametros que utilizaremos en el proyecto) y variando el *spreading factor,* obtenemos los siguientes resultados [Tabla 6]:

SF	Long. símbolo	Símbolos / trama	ToA	Duty cycle
7	1.02 ms	23	36.10 ms	Un mensaje cada 4s
8	2.05 ms	18	61.95 ms	Un mensaje cada 6s
9	4.10 ms	18	123.90 ms	Un mensaje cada 12s
10	8.19 ms	18	247.81 ms	Un mensaje cada 25s
11	16.38 ms	18	495.62 ms	Un mensaje cada 50s
12	32.77 ms	18	991.23 ms	Un mensaje cada 1:39m

Tabla 5 – Variación del Time on Air variando el SF

2.4.1.6 Duty cycle

Es un parámetro de regulación del espectro e indica la proporción del tiempo durante el cual un componente, dispositivo o sistema debe operar. Está relacionado con el ToA y puede ser expresado como ratio o como porcentaje. En Europa existe un 0.1% y 1% de *duty cycle* por día dependiendo del canal.

En la última columna de la tabla anterior [Tabla 6], se observa cómo ya ha sido calculado el *duty cycle* por la herramienta, multiplicando el ToA por 99 (*duty cycle* de 1%).

2.4.1.7 RSSI y SNR

En la parte del receptor, existen dos medidas ampliamente utilizadas para conocer el nivel de la señal recibida.

El *Received Signal Strength Indication* (RSSI) indica cómo de bien escucha el receptor una señal del transmisor. Es la potencia de señal recibida en miliwatios y se mide en dBm siendo un valor negativo (cuanto más cercano al 0 sea mejor). El mínimo RSSI típico en la comunicación LoRa es de -120dBm.

El *Signal-to-Noise Ratio* (SNR) es la relación entre la potencia de señal recibida y la potencia de suelo de ruido. El suelo de ruido contempla todas las interferencias no deseadas que corrompen la comunicación y fuerza retransmisiones. Normalmente, el nivel de ruido es el límite físico de la sensibilidad, en cambio, LoRa trabaja por debajo del nivel del ruido gracias a su modulación por CSS. Los valores típicos SNR en la comunicación LoRa están comprendidos entre -20dB y 10dB.

2.4.2 LoRaWAN

LoRaWAN es un protocolo de control de acceso al medio (capa 2 del modelo OSI), diseñado para ser implementado sobre redes LoRa [Fig. 8]. Se trata de una capa de acceso al medio compleja, debido a las posibilidades y funcionalidades que ofrece: soporte de numerosos tipos de dispositivo, activación y configuración de los dispositivos en la red e implementación de medidas de seguridad que aportan integridad y confidencialidad en el intercambio de mensajes [10].

Es un protocolo de red que esta optimizado para dispositivos alimentados por batería, que pueden estar ubicados en lugares fijos o pueden ser móviles.

Permite variar dinámicamente el *bitrate* de los equipos que están en la red, desde valores de 0,3 Kbps hasta 37,5 Kbps, dependiendo del chip utilizado y parámetros de modulación.

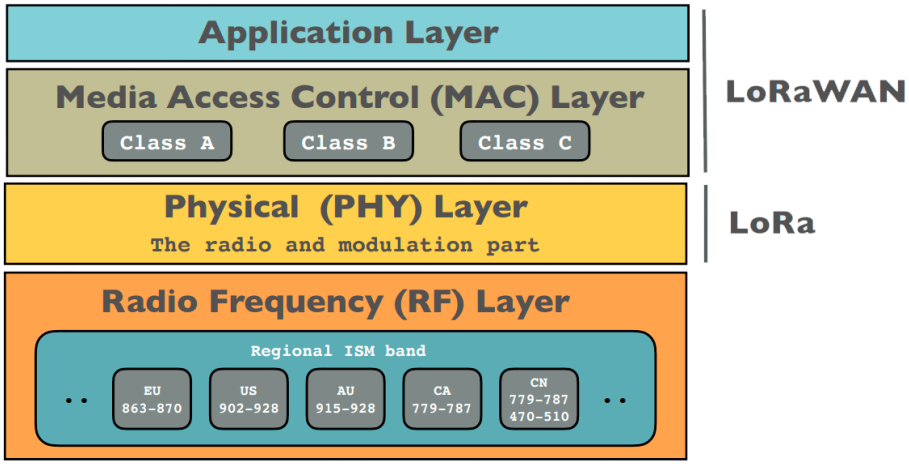


Fig. 8 – Modelo de capas del protocolo LoRa

2.4.2.1 Arquitectura

Su arquitectura está basada en topología estrella utilizando *gateways* como punto medio, que hacen de "pasarela" entre los nodos y el servidor de aplicación. Consta de 3 tipos de entidades: nodos, *gateways* "pasarelas" y servidor de aplicación. Los nodos se comunican con el gateway utilizando LoRa.

El *gateway* se comunica con el servidor de aplicación mediante IP. El *gateway* debe tener al menos 8 módulos de radio simultáneamente, para ser considerado un *Gateway* LoRa.

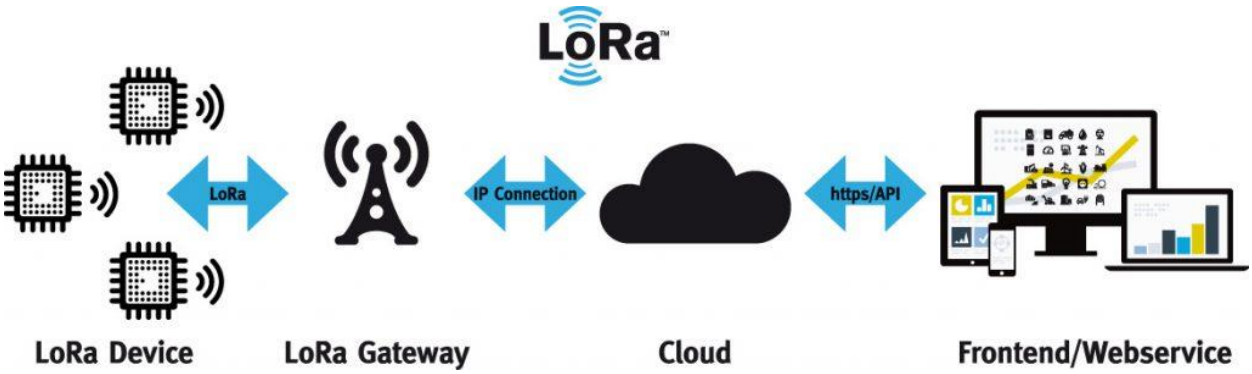


Fig. 9 – Arquitectura LoRa

Es posible implementar redes de tipo *mesh*, en la cual los nodos retransmiten mensajes de otros nodos, pero en este caso los nodos siempre deberían estar recibiendo y transmitiendo mensajes, lo que impacta en un alto consumo de energía de cada nodo. En este caso no estaríamos dentro del marco de IoT y LPWAN.

2.4.2.2 Clases de nodo

Se definen en LoRaWAN las siguientes clases de dispositivos:

- **Clase A (All)**: Los dispositivos de esta clase permiten realizar comunicaciones bidireccionales, donde el dispositivo después de enviar su información dispone de dos pequeñas ventanas de tiempo para recibir información. Los dispositivos envían información cuando ellos lo desean. Esta clase de operación es la que utilizan los dispositivos con menores recursos de energía.

  Se utiliza para aplicaciones que envían poca información desde el servidor hacia el nodo final y requieren poco consumo energético. En el caso de que el servidor necesite enviar mensajes al nodo, se debe esperar que éste envíe un mensaje al servidor.

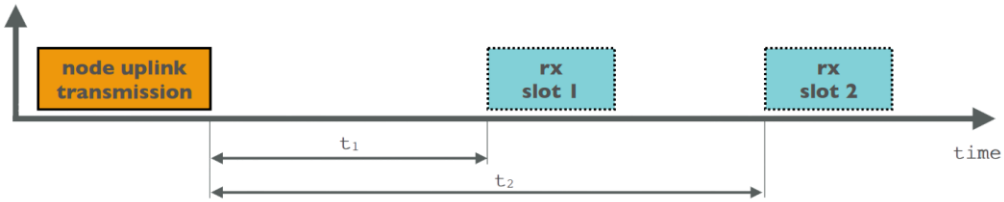


Fig. 10 - Clase A

- **Clase B (Beacon)**: En esta clase se pueden crear ventanas de recepción adicionales sin necesidad de que anteriormente haya existido una transmisión, aumentando así la capacidad de recibir datos por parte del dispositivo.

  La pasarela puede sincronizarse con el dispositivo final a través del envío de tramas *beacon* y, de esta forma, planificar el tiempo que éste debe mantener abierta la ventana de recepción. Este modo de funcionamiento implica un mayor consumo energético.

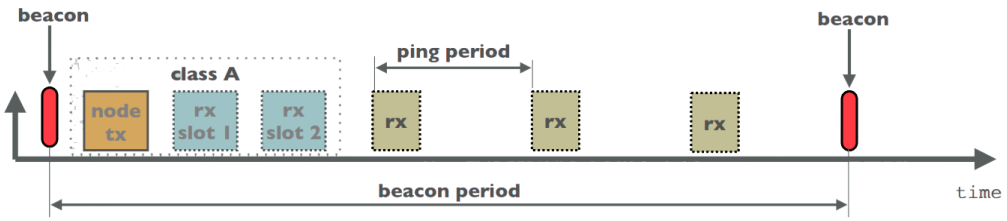


Fig. 11 - Clase B

- **Clase C (Continuous):** Los dispositivos se mantienen constantemente en estado de recepción, el cual solo se interrumpe si se produce una transmisión. Este modo de funcionamiento implica un gran consumo de energía.

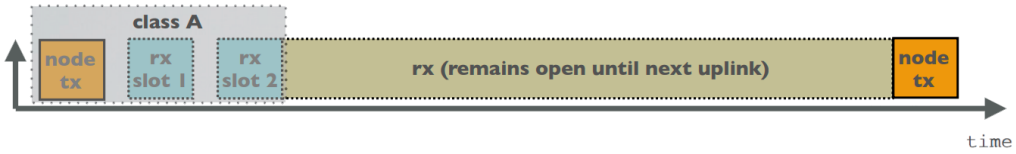


Fig. 12 – Clase C

Por un lado, nuestra aplicación requiere que los dispositivos tengan el mínimo consumo posible y, además, no supone un problema esperar a que el nodo final transmita para responder al nodo desde el servidor, por lo que los dispositivos del sistema serán de clase A.

2.4.2.3 Trama

En la siguiente figura [Fig. 13], se indica la estructura detallada de la trama LoRa desde su paquete físico (explicado en el apartado 2.4.1.4).

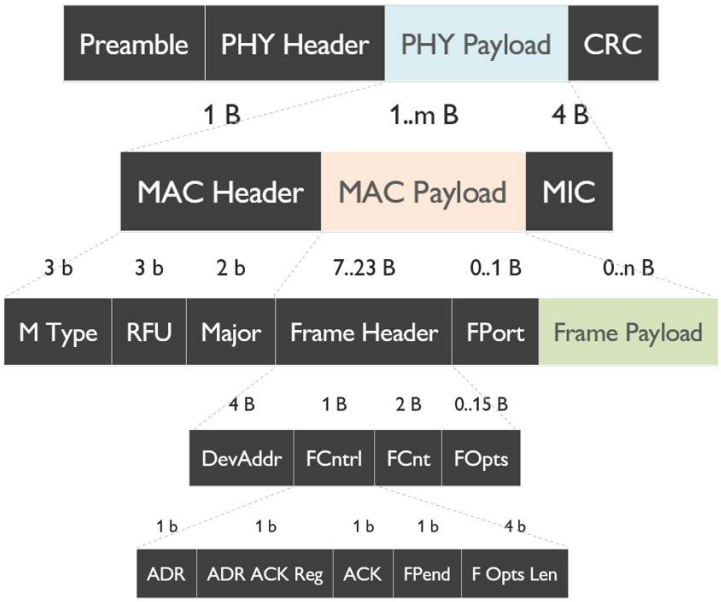


Fig. 13 – Estructura de la trama LoRa

Dentro de la carga del paquete físico (PHY *payload*), se distingue la cabecera MAC, que se encarga de especificar el tipo del mensaje en su campo *M Type* [Tabla 6], además, se reservan 3 bits en el campo RFU para uso futuro y se especifica el formato del procedimiento *join* en los 2 bits de *Major*. Junto con la cabecera, aparece la carga a nivel de enlace (MAC *payload*) y un *Message Integrity Check* (MIC) que verifica la no alteración del mensaje.

La carga del paquete de enlace (MAC *payload*) contiene la cabecera de la trama, un campo opcional de carga de trama o *frame payload* (este campo va a contener las medidas de los sensores) y un campo opcional de puerto (dependiente del campo carga de trama). Si el puerto es 0 indica que dentro de la carga de trama solo existen comandos MAC.

La cabecera de trama contiene la dirección del dispositivo (DevAddr), un campo de control (FCtrl), un contador de trama (FCnt) y hasta 15 *bytes* de opciones usadas para transportar comandos MAC. Los 7 bits más significativos del campo DevAddr sirven para identificar la red (NwkID), mientras que los 25 restantes se corresponden con la dirección de red (NwkAddr), la cual puede ser configurada y escogida por el administrador de red.

El campo de control contiene el campo *Adaptative Data Rate,* que cuando se activa (bit ADR), la red se optimiza buscando la tasa de datos más rápida. Este campo se activa por el nodo final o la propia red bajo demanda. Los campos de *Acknowledgement* (ACK) se usan para confirmar el recibo del mensaje si su tipo es de confirmación (tipos 100 y 101 de la Tabla 6). El bit de trama pendiente (FPend) se utiliza desde el *gateway* para indicar que faltan mensajes por enviar hacia el nodo. El campo longitud de opciones de trama indica el tamaño del campo FOpts.

Tipo	Descripción	Tipo	Descripción
000	Join Request	100	Confirmed Data Up
001	Join Accept	101	Confirmed Data Down
010	Unconfirmed Data Up	110	RFU
011	Unconfirmed Data Down	111	Propietary

Tabla 6 – Tipos de mensajes MAC

2.4.2.4 Métodos de activación

Para entrar en la red, así como para mantener la seguridad de la misma, se requieren una serie de claves y número de identificación por parte del nodo. Existen dos métodos de activación de estas claves.

- **OTAA** (*Over-The-Air Activation*): Es la manera más segura de conectarse a la red, debido a que las claves se renuevan cada vez que se pierde la conexión, se apaga o se reinicia el nodo.

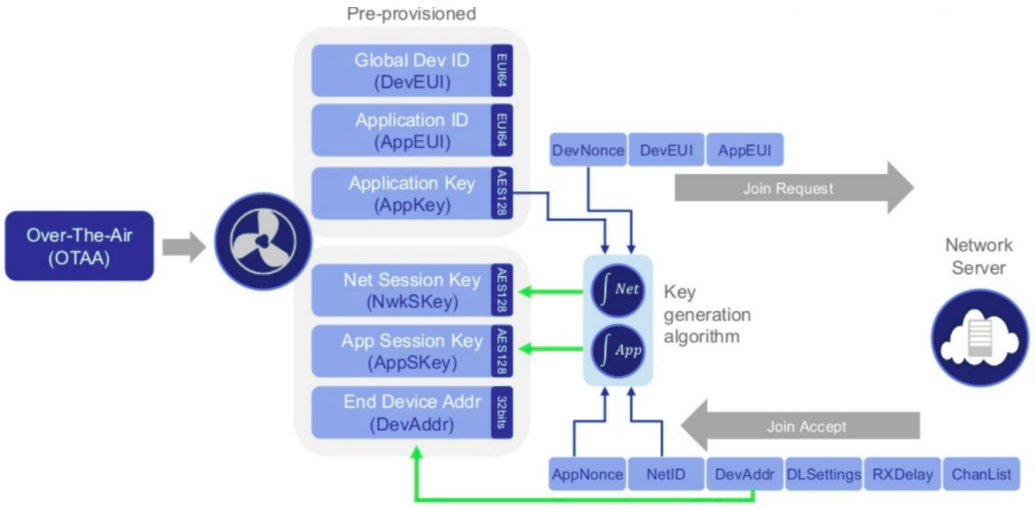


Fig. 14 – Método de activación OTAA

Los parámetros intrínsecos en el dispositivo son el identificador de fábrica propio del nodo (DevEUI) que hace cada dispositivo único, el identificador de aplicación (AppEUI) que es optativo en algunas aplicaciones y se utiliza para clasificar dispositivos y la clave secreta de aplicación (AppKey) con configuración de seguridad tipo AES-128b, compartida entre el nodo y la red que determinan las claves de sesión.

Con esta configuración, el nodo solicita un *join* a la red y abre la ventrana de recepción, el *gateway* recibe la solicitud y la envía al servidor, éste verifica que el nodo este dado de alta y su llave de encriptación sea correcta, si es así, asigna una sesión temporal al nodo por medio del *gateway* (ofreciéndole las claves de sesión y su dirección lógica DevAddr) pudiendo este último enviar datos a la red.

Las claves de sesión son NwkSKey que garantiza una conexión segura entre el nodo final y la red y AppSKey que garantiza dicha conexión segura hasta la aplicación.

- **ABP** (*Activation By Personalization*): Este es el modo más sencillo, ya que no requiere realizar un *join* a la red puesto que la sesión está manualmente asignada. Se utiliza en aplicaciones con dispositivos en movimiento o con mala recepción. Su desventaja es la brecha de seguridad que supone tener la llave de encriptación asignada desde el inicio.

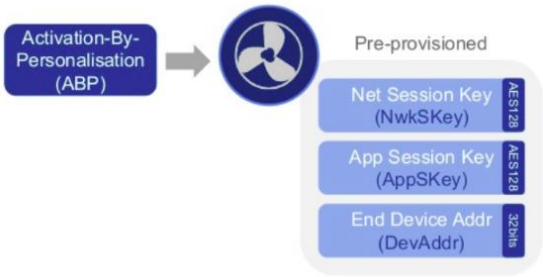


Fig. 15 – Método de activación ABP

2.5. Servidor de red LoRa

En el servidor de red tienen lugar los procesos más complejos de tratamiento de datos, siendo responsable de:

- Enrutamiento/reenvío de mensajes a la aplicación adecuada.
- Seleccionar la mejor puerta de enlace para el mensaje de enlace descendente, típicamente en función de una indicación de calidad de enlace, calculada a partir del RSSI (Indicación de la intensidad de la señal recibida) y SNR (Relación de señal a ruido) de los paquetes recibidos previamente.
- Eliminación de mensajes duplicados si se reciben por múltiples puertas de enlace.
- Descifrar los mensajes enviados desde los nodos finales y encriptar los mensajes que se envían a los nodos.

Existen multitud de servidores de red que soportan LoRaWAN, pero las dos soluciones *open-source* más utilizadas por la comunidad son The Things Network (TTN) y LoRaServer, recientemente renombrado como ChirpStack (nov 2019).

2.5.1 The Things Network

Es una red pública de amplio alcance y bajo consumo, a la que puede sumarse cualquier persona. Se trata de una iniciativa global, desplegada en múltiples ciudades del mundo, con el apoyo de grupos de voluntarios [11].

Su pila de servidor LoRaWAN en su versión 3 soporta todas las versiones de LoRaWAN, los modos de operación A, B y C y todos los parámetros regionales. Su arquitectura de microservicios permite obtener:

- **Seguridad**: Utiliza *join servers* que almacenan de forma segura las claves de LoRaWAN y emiten claves de sesión al servidor de red y al de aplicaciones.
- **Integraciones**: Permite integraciones con los protocolos gRPC, HTTP y MQTT.
- **Escalabilidad**: Permite la replicación de instancias de servicios distribuidos en *clusters* para crear redes globales escalables.
- **Monitorización**: Permite detectar incidencias, realizar diagnósticos y obtener alertas.

Cuenta también con una versión descargable que permite implementar una red propia LPWAN de forma local, *LoRaWAN Stack*, es un proyecto muy completo, pero aún prematuro en comparación con LoRaServer.

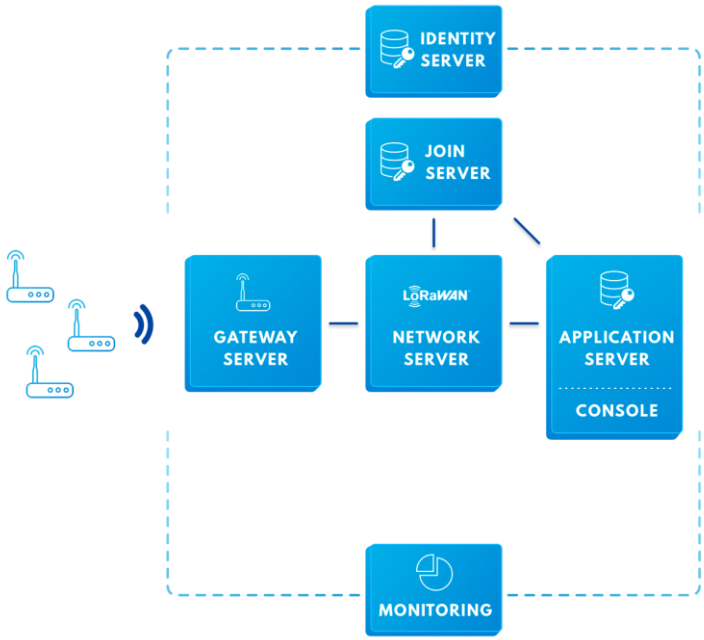


Fig. 16 – Estructura del servidor de red The Things Network

2.5.2 ChirpStack

ChirpStack (antiguo LoRaServer), se compone de una pila de servidores de red LoRaWAN de código abierto, que incluye una interfaz web para la administración de dispositivos y APIs para realizar integraciones.

ChirpStack soporta igualmente las clases A, B y C, la tasa de datos adaptativa (ADR), reconfiguración de canal, soporte multi-tenant, parámetros regionales y todas las versiones de LoRaWAN.

La arquitectura de ChirpStack es similar a la arquitectura de TTN, está basada en gateways y servidores que interconectan los nodos LoRaWAN con aplicaciones externas, pero en ChirpStack hay varios servicios que pueden o no estar en un mismo servidor y que se detallan en la siguiente figura [12]:

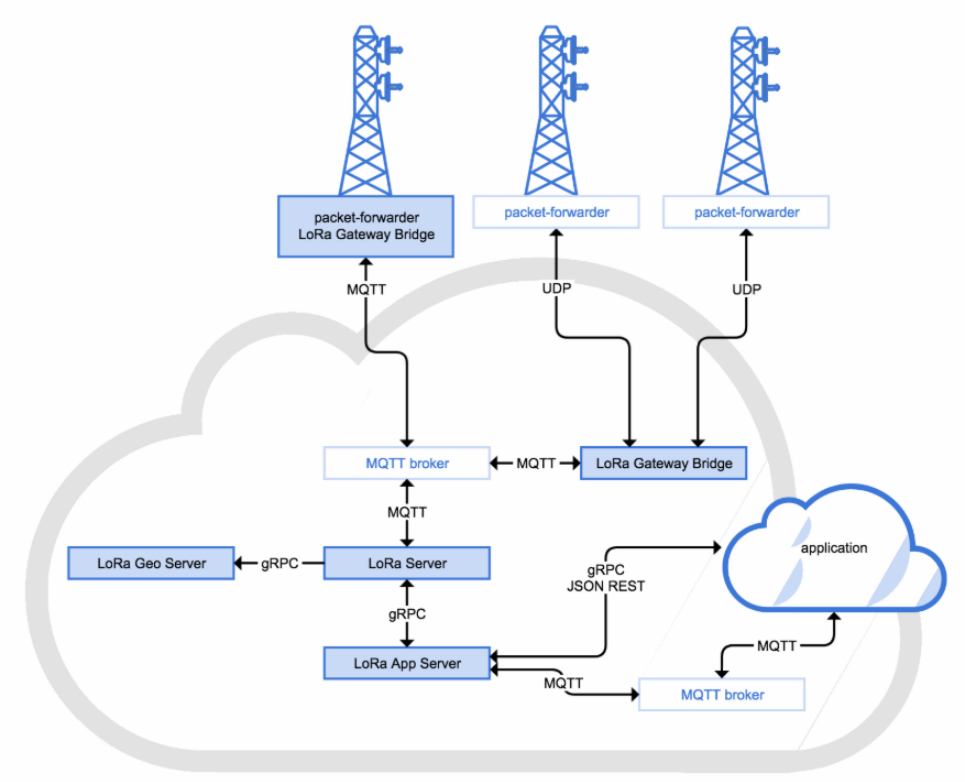


Fig. 17 - Estructura del servidor de red ChirpStack

Los elementos de esta arquitectura son:

- **LoRa Gateway Bridge**: Es un servicio que convierte los protocolos de reenvío de paquetes LoRa en un protocolo común de servidor LoRa (JSON y Protobuf).
- **LoRa Server**: Es el servidor de red LoRaWAN propiamente dicho, y es la parte principal del proyecto ChirpStack. La responsabilidad del componente servidor de red es la desduplicación y el manejo de las tramas de enlace ascendente recibidas por las puertas de enlace o gateways, el tratamiento de la capa de mac LoRaWAN y la programación de transmisiones de datos de enlace descendente.
- **LoRa App Server**: Es un servidor de aplicaciones LoRaWAN. Es responsable de la parte del inventario de dispositivos de una infraestructura LoRaWAN, del control de las solicitudes de unión y el tratamiento y cifrado de las cargas útiles de las aplicaciones. Ofrece una interfaz web donde se pueden administrar usuarios, organizaciones, aplicaciones y dispositivos. Para la integración con servicios externos, ofrece una API REST y gRPC.
- **LoRa Geo Server**: Es un servidor de geolocalización. Se puede usar para resolver la ubicación de dispositivos basados en metadatos TDoA (diferencia de tiempo de llegada) proporcionados por los gateways LoRaWAN.

Debido al mayor estado de madurez de Chirpstack con respecto a TTN a la hora de disponer de una red propia *full-stack* y así tener un control total de la misma, se utilizará ChirpsStack como servidor de red para el proyecto.

2.6. Servidor de aplicación

Existe una gran diversidad de aplicaciones *open-source* en IoT, entre las más destacadas se encuentran Thingsboard, DeviceHive o Mainflux, todas ellas tienen licencia Apache 2.0, lo que significa que se pueden ejecutar en un servidor propio y ofrecerlo comercialmente, con marca propia.

Thingsboard es la única que dispone de dashboards para visualizar la información recibida, con gráficas en función del tiempo, indicadores, mapas, etc. Esta razón hace que se haya elegido Thingsboard como servidor de aplicación para el proyecto.

2.6.1 Thingsboard

Como ya se ha indicado, Thingsboard es una plataforma de código abierto para la recopilación de datos, el procesamiento, la visualización y la gestión de dispositivos. Permite la conectividad de los dispositivos a través de los protocolos IoT estándar de la industria MQTT, CoAP y HTTP. ThingsBoard combina escalabilidad, tolerancia a fallos y rendimiento para asegurar la integridad de sus datos [13].

Entre sus características, se encuentran las indicadas a continuación:

- Es multi tenant, lo que permite servir a distintos clientes con configuraciones diferentes. A su vez, cada cliente puede tener múltiples usuarios y cada usuario gestionar distintos dispositivos.
- Dispone de *dashboards* en vivo para la visualización de datos y el control remoto de los dispositivos.
- El manejo de eventos se realiza de manera gráfica utilizando su lógica de reglas llamada *rule chain*, conectando bloques de código para enviar datos a otras aplicaciones, enviar notificaciones o disparar alarmas.
- Permite usar encriptación tanto en HTTPS o MQTT y admite el uso de credenciales en los dispositivos que se conectan a la plataforma.
- En cuanto a la gestión de los datos, permite utilizar tanto bases de datos SQL como NoSQL, con la capacidad de utilizar diferentes bases para las entidades principales y los datos de telemetría.
- Está desarrollada en Java 8.

3 ARQUITECTURA Y COMPONENTES

La inteligencia del riego no está en el suelo, sino en la nube

A lo largo de este capítulo se plantea la arquitectura del proyecto, mostrando su distribución o topología y distinguiendo cada nodo conformador de la solución. Se analizarán cada uno de los componentes electrónicos necesarios para hacer funcionar el sistema y anotaremos sus características técnicas que nos servirán de ayuda a la hora de su posterior integración.

3.1 Arquitectura de la solución

Teniendo en cuenta la base teórica recogida en el apartado anterior, la arquitectura del sistema propuesta se muestra en la siguiente ilustración [Fig. 18].

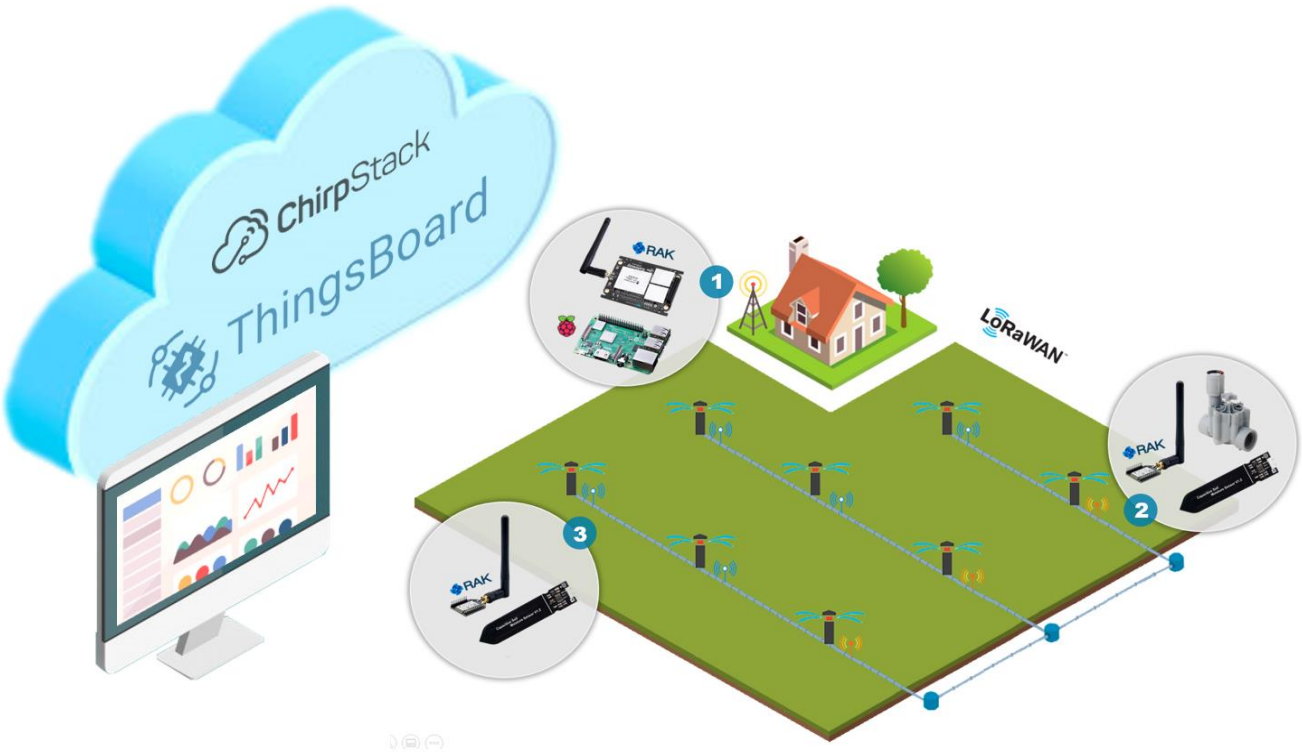


Fig. 18 – Arquitectura de la solución

En la figura anterior se ilustra una posible topología para el sistema de riego por aspersión, con una distribución de filas de aspersores, estando cada una de ellas controlada por un nodo inicial, responsable de dejar pasar el caudal que pasa por las mismas.

El sistema cuenta con tres tipos de dispositivos diferentes, enumerados en el propio diagrama:

1. **Nodo coordinador (*gateway*):** Es la puerta de enlace entre la red de nodos sensores y la nube. Está compuesto de la placa Raspberry Pi 3 que controla el módulo concentrador RAK831. Este módulo utiliza el chipset SX1301, especializado en el procesamiento de señales de banda ISM y con soporte de las funciones de pasarela LoRa. Este nodo dispone de toma de corriente y conexión a Internet.
2. **Nodo sensor/actuador**: Este nodo recoge la humedad del suelo a una frecuencia programada y la transmite al *gateway*, recibiendo a su vez la orden de encendido o apagado del riego aplicado en su canalización, actuando de esa forma sobre el medio.
   Está compuesto de un sensor de humedad del suelo, una electroválvula y un módulo transceptor RAK811. Este módulo utiliza el chipset SX1276, soportando el modo de trabajo LoRaWAN como nodo final. Solo existirá un nodo de este tipo en la entrada de cada canalización. Este nodo se alimenta mediante baterías.
3. **Nodo sensor:** Este nodo es similar al anterior, sin tener en cuenta la parte de actuación en el medio (electroválvula), recogiéndo la humedad del suelo y transmitiéndola a la pasarela.

El funcionamiento del sistema comienza recogiendo la humedad del suelo por parte de los sensores y transmitiéndola a la pasarela haciendo uso del protocolo LoRaWAN. La pasarela, reenvía dichos paquetes a través de internet (protocolo UDP) al servidor de red ChirpStack, que a su vez se encuentra integrado con la aplicación Thingsboard. Ésta última procesa los datos, realiza una toma de decisión automática basándose en una lógica preestablecida y transmite una petición *downlink* (de servidor a nodo) haciendo uso de la API REST de ChirpStack. Dicho servidor entrega el paquete a la pasarela que a su vez reenvía al nodo sensor/actuador, que enciende o apaga la electroválvula según la decisión que haya tomado el servidor.

Además, el servidor de aplicación Thingsboard en su interfaz web dispone de un cuadro de mando en el que se muestra el estado de la red, pudiendo ser monitorizada en todo momento y ofreciendo la posibilidad de modificar las reglas de toma de decisión en vivo.

3.2 Nodo coordinador

El nodo coordinador se compone del propio concentrador (puerta de enlace a internet) y un sistema anfitrión que lo controla. A continuación, se estudian las características de los módulos que van a ser utilizados para ello, con el objetivo de realizar de manera eficiente su implementación.

3.2.1 RAK831

Se trata de un módulo concentrador transceptor (transmisor y receptor) multicanal, diseñado para recibir numerosos paquetes de datos LoRa, utilizando distintos factores de dispersión (SF) en los 8 posibles canales. Esto proporciona una gran mejora de la robustez del sistema en términos de inmunidad a las interferencias y diversidad de canales radioeléctricos, permitiendo así una conexión estable entre la red de nodos que componen el sistema y la pasarela [14].

Fig. 19 – Módulo concentrador RAK831

El módulo RAK831 se basa en el chip de banda base digital SX1301, que incluye un motor de procesamiento de señal digital diseñado específicamente para ofrecer capacidades de pasarela en las bandas ISM de todo el mundo.

En la parte receptora, recibe el flujo de bits desde el receptor SX1257, demodula estas señales y almacena los paquetes en un FIFO para ser recuperados por el sistema anfitrión. En la parte del transmisor, los paquetes se modulan utilizando un modulador FSK LoRa y se envían al transmisor SX1257.

RAK831			
CPU	SX1301	TX/RX	SX1257
Supply Voltage	5 V	Channels	8
Sensibility	-142.5 dBm	LoRaWAN	1.0.2
Freq. Bands	433 / 470 / 868 / 915 MHz	Price	~ 120 €

Tabla 7 – Características técnicas RAK831

Para su correcto funcionamiento, este módulo necesita un sistema anfitrión (en nuestro caso la placa Raspberry pi), que se conecta al módulo a través del protocolo serie SPI.

3.2.2 Raspberry Pi 3

Raspberry es una placa computadora de bajo coste y código abierto, desarrollada en el Reino Unido por la Fundación Raspberry pi, con el objetivo de estimular la enseñanza de la informática en las escuelas. La placa Raspberry Pi 3 modelo B es la tercera generación de Raspberry Pi, aportando a sus sucesoras mayor velocidad de computación y añadiendo conectividad WLAN y Bluetooth [15].

Existen numerosas distribuciones Linux adaptadas a esta placa, pero cuenta además con un sistema operativo propio llamado Raspbian, siendo el que utilizaremos en el proyecto.

Fig. 20 – Raspberry Pi 3 Model B

Con la raspberry, el nodo coordinador será capaz de recibir y controlar los mensajes recibidos por el *gateway* y reenviarlos al servidor de red a traves de Internet. Asímismo, recibirá la respuesta del servidor y se la entregará al *gateway* para que lo reenvíe a los nodos finales.

Raspberry Pi 3 Model B			
SoC	BCM 2837	GPIO	40
CPU	ARM Cortex-A53	Ports	4 x USB 2.0
Proc. Power	1.2 GHz	WLAN	802.11 b/g/n
Memory RAM	1 GB	Price	~ 40 €

Tabla 8 – Características técnicas RPi3-B

3.3 Nodo sensor

El nodo sensor se encarga de recoger la humedad del suelo a una frecuencia temporal predeterminada y transmitirla a la pasarela siguiendo el protocolo LoRaWAN.

3.3.1 RAK811 Breakout

Se trata de un módulo LPWAN con soporte LoRa de largo alcance, campacto y de bajo consumo, con capacidades de transceptor inalámbrico. El nodo puede trabajar en varios modos, con diferentes velocidades de datos y leer los datos de los sensores conectados con la ayuda de sus comandos AT en serie [16].

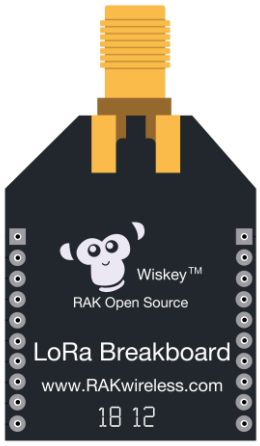


Fig. 21 – Módulo RAK811 Breakout

RAK811					
CPU	STM32L	TX/RX	SX1276	TX Consume	30 mA
Supply Voltage	3.3 V	TX Power	14 dBm	RX Consume	5.5 mA
LoRaWAN	1.0.2	RSSI	-130 dBm	Sleep Consume	7.2 uA
Freq. Bands	433 / 470 / 868 / 915 MHz	SNR	-15 dB	Price	~ 14 €

Tabla 9 - Características técnicas RAK811

El fabricante (RAK Wireless), también ofrece una API llamada *RAK Unified Interface* (RUI), que agiliza la programación del módulo sirviendo como capa de abstracción. En el proyecto, vamos a programar directamente sobre el propio STM32 del módulo basándonos en RUI, evitando así un sistema anfitrión (como podría ser una placa arduino) y optimizando al máximo su consumo.

3.3.2 Sensor de humedad del suelo

Existen diversos sensores para medir la humedad del suelo, siendo la mayoría de tipo resistivo o capacitivo. Los sensores resistivos son más inestables, presentando variaciones en las lecturas por factores externos como el fertilizante. Además, están compuestos de un material metálico en sus puntas que con el tiempo se oxida al estar en contacto con la humedad, disminuyendo así su vida útil.

En el proyecto, por tanto, vamos a utilizar sensores capacitivos, que miden el nivel de humedad del suelo por medio de la detección capacitiva, es decir, la capacitancia varía en función del contenido de agua presente en el suelo. Este valor de capacitancia se convierte a un valor de tensión entre 1.2 y 3V. La ventaja de estos sensores es su larga vida útil al ser resistentes a la corrosión por no tener las placas expuestas al aire libre [17].

Fig. 22 – Sensor de humedad del suelo

El sensor propiamente dicho, trata de un condensador compuesto por una placa positiva, una negativa y el espacio entre ellas, conocido como el dieléctrico. Midiendo los iones que se disuelven dentro del dieléctrico se obtiene un valor aproximado de la humedad en el suelo. Hay que tener en cuenta que la humedad del suelo también se ve afectada por la densidad del suelo, la cantidad de fertilizante o la temperatura, entre otros.

Internamente, dispone de un oscilador de frecuencia que genera una onda cuadrada que se aplica al capacitor, provocando una reactancia que, junto con otra resistencia, genera un divisor de tensión. Cuanto mayor sea la humedad del suelo, mayor será la capacitancia del sensor y menor su reactancia, haciendo que dismituya el voltage en la línea de señal (A0).

Capacitive Soil Moisture Sensor v2.0	
Operation Voltage	DC 3.3 – 5.5 V
Operation Current	5 mA
Output Voltage	DC 0 – 3 V
Pins	VCC. GND, Analog read
Price	~ 3 €

Tabla 10 – Características técnicas sensor humedad

El agua contenida en la humedad del suelo es la suma del agua libre, el agua capilar y el agua molecular. Desde un punto de vista agronómico, la fracción interesante es la capilar, ya que es la única capaz de ser retenida por el suelo y por tanto la que alimenta a las plantas.

Por otro lado, existen dos intervalos entre los cuales la planta se encuentra en los niveles adecuados de humedad disponible:

- **Capacidad de campo**: Es la cantidad máxima de agua que un suelo puede retener después del drenaje del agua libre. Si tenemos en cuenta lo descrito anteriormente, la fracción de agua correspondiente a la capacidad de campo será prácticamente equivalente al agua capilar.
- **Punto de marchitez**: Es el momento en que no existe agua disponible para las plantas y éstas comienzan, como indica el término a definir, a marchitarse.

Por tanto, trataremos de mantener la superficie plantada con una humedad establecida en un punto medio entre estos dos términos, ofreciendo unas condiciones idóneas para su nutrición.

3.3.3 Baterías

Los nodos deben alimentarse de baterías para que la aplicación sea implementable. Dentro de la infinidad de tipos de batería existentes, se han elegido para el proyecto las de tipo Li-ion 18650 recargables.

Fig. 23 – Baterías Li-ion 18650

Las baterías 18650 son pilas de litio recargables muy parecidas a las AA, pero con un tamaño más grande (18mm x 65mm). Estas baterías de litio también llamadas baterías Li-ion, son un dispositivo fabricado para la acumulación de energía eléctrica que utilizan como electrolito una sal de litio, que logra los iones precisos para la reacción electroquímica transformable, que tiene lugar entre el cátodo y el ánodo [18].

Son baterías muy ligeras, pero con una elevada capacidad energética y resistencia a la descarga, y con poco efecto memoria, aguantando un elevado número de ciclos.

Cuenta con circuitos que cortan la tensión de salida cuando la batería no debe descargarse más o cortando la tensión de entrada cuando ya se ha cargado suficiente.

Batería Li-ion 18650	
Nominal Capacity	9800 mAh
Nominal Voltage	3.7 V
Cut-off Voltage	3 V
Max Charge Voltage	4.2 V
Price	~ 1 €

Tabla 11 – Características técnicas batería Li-ion 18650

3.3.4 Regulador step-up MT3608

El regulador (Step Up o Boost) MT3608 es un dispositivo electrónico que permite la regulación o conversión de voltaje de corriente directa DC a otro voltaje DC con una alta eficiencia de conversión y excelente regulación de línea y bajo voltaje de rizado. Puede tomar voltajes de entrada tan bajos como 2V y aumentar la salida hasta 28V. Como se trata de un convertidor elevador, el voltaje de salida debe ser mayor que el voltaje de entrada suministrado [19].

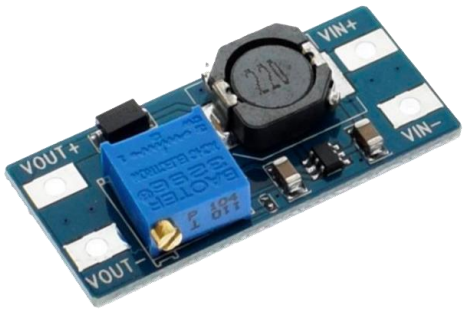


Fig. 24 – Regulador step-up MT3608

Este módulo tiene un potenciómetro para ajustar el voltaje de salida con el que se puede ajustar fácilmente la salida del módulo al voltaje necesario, pero la corriente de salida sera menor en comparación con la corriente de entrada. Incluye además bloqueo por bajo voltaje, limitación de corriente y protección contra la sobrecarga térmica para evitar posibles daños.

Regulador step-up MT3608			
Input Voltage	2 – 24 V	Output Voltage	5 – 28 V
Max. Output Current	2 A	Output Power	6 W
Efficiency	93 %	Price	~ 1 €

Tabla 12 – Características técnicas regulador step-up MT3608

3.3.5 Regulador lineal L7805CV

Se trata de un dispositivo capaz de modificar una señal de tensión que obtiene a su entrada y ofrecer una señal de 5V a su salida. Para hacer eso posible, el regulador de tensión cuenta con un circuito interno con una serie de resistencias y transistores bipolares conectados de tal forma que permiten afinar la señal del voltaje de una forma adecuada [20].

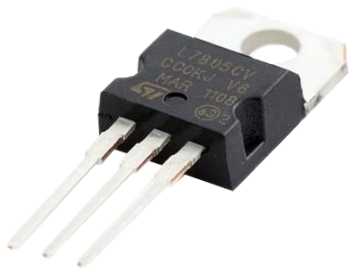


Fig. 25 – Regulador L7805CV

La funcionalidad principal de este dispositivo es ofrecer una señal estable a la salida ante una señal variable a su entrada.

Es recomendable utilizar condensadores a la entrada y a la salida del dispositivo para suavizar la ondulación en la entrada y desacoplar la salida.

3.4 Nodo sensor/actuador

El dispositivo restante es el nodo que además de las labores de sensor comentadas en el apartado anterior, actúa sobre el medio si el servidor así se lo indica. Para ello, este nodo requiere la adición de un nuevo componente, la electroválvula.

3.4.1 Electroválvula solenoide latch

Las válvulas de solenoide tipo Latch están normalmente desenergizadas y son activadas por un impulso eléctrico momentáneo, que coloca al solenoide en posición de apertura.

Se activa para liberar la presión de control de la válvula principal mediante un breve impulso eléctrico a la bobina Latch o "Pull", tras lo cual el actuador del solenoide se engancha en la última posición por medio de un imán de retención (*Holding Magnet*) permanente.

Para el rearme del sistema, debe aplicarse un impulso eléctrico a la bobina De-latch o "Push", creando así una fuerza contraria que neutraliza al imán de retención y permite que el actuador vuelva a su posición normal.

Fig. 26 – Valvula solenoide latch

Las válvulas de solenoide tipo Latch se utilizan en sistemas que funcionan con un limitado suministro de energía eléctrica. Requieren impulsos de tan solo 50ms para funcionar, con lo cual se reduce la demanda de energía y se incrementa la fiabilidad del sistema.

Para el proyecto, se ha dispuesto de la electroválvula tipo latch de 2 vías (2W) normalmente cerrado (NC) con resistencia de 4Ω. Teniendo en cuenta su ficha técnica o *datasheet* [21], obtenemos las siguientes características relevantes:

Electroválvula			
Switching Period	25 – 50 ms	Pressure	12 bar
Operating Voltage	6 – 9 V	Price	~ 15 €

Tabla 13 – Características técnicas válvula solenoide latch

Debido a que se trata de una válvula de dos hilos, necesita pulsos positivos y negativos para activar o desactivar el paso del flujo de agua. Necesitamos que este cambio de polaridad se realiza de manera automática siendo controlado por el nodo RAK811, por lo que necesitaremos un circuito adicional que se encargue de esto, como puede ser el puente H-L298.

3.4.2 Puente H L298N

El distintivo de este módulo es el gran chip negro con disipador de calor (L298N). Es un controlador de motor H-Bridge de doble canal capaz de conducir un par de cargas de continua. Se alimenta a través de una borna de tres pines, la fuente de alimentación de la carga (Vs), la fuente de alimentación lógica de 5V (Vss) y masa o tierra que debe ser la misma referencia para ambas alimentaciones [22].

Dispone de un jumper que da la posibilidad de alimentar la entrada lógica (Vss) con la propia entrada de alimentación de la carga (Vs), mediante un regulador de tensión 78M05.

Existe una caída de alrededor de 2V entre la tensión de alimentación de la carga y la salida a la misma, debido a los transistores de conmutación del circuito.

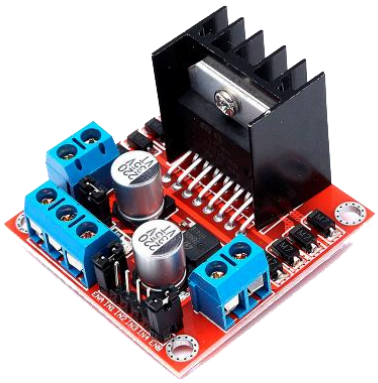


Fig. 27 – Puente H - L298N

La particularidad del puente H se encuentra en la manipulación de sus interruptores, pudiendo aplicarle una tensión positiva o negativa a la carga.

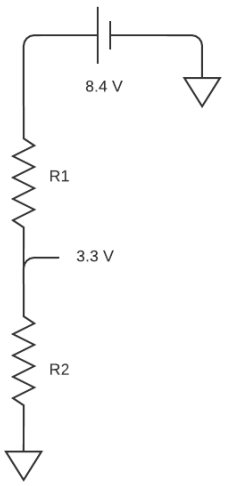
Puente H L298N			
Logic Voltage	5 V	Driving Voltage	5 – 35 V
Logic Current	0 – 36 mA	Driving Current	2 A
Max. Power	25 W	Price	~ 2 €
Positive Pulse	ENA ↑ IN1 ↑ IN2 ↓	Negative Pulse	ENA ↑ IN1 ↓ IN2 ↑

Tabla 14 – Características técnicas Puente H - L298N

3.4.3 Otros componentes

Este nodo tendrá dos baterías, una para alimentar el RAK811 (igual que en el nodo sensor) y una bateria adicional compuesta por dos pilas en serie para alimentar la electroválvula, que dispondrá entre 6 y 8.4V (la electroválvula trabaja entre 6-9V).

Esta segunda fuente de alimentación, que se consumirá antes que la primera, va a disponer de una monitorización de su nivel de batería. Para hacer esto, necesitaremos un divisor resistivo para que en su punto central tenga un máximo de 3.3V, ya que éste es el voltaje más elevado que el nodo RAK811 puede leer (su tensión de trabajo). Una vez medido, se realizarán unos cálculos para obtener el voltaje real.



La bateria compuesta por dos pilas en serie tendrá un máximo de 8.4V y el máximo voltaje que puede medir el nodo RAK811 es 3.3V, por lo que nos queda calcular el valor de las resistencias.

$$3.3 = 8.4 \times \frac{R_2}{R_1 + R_2}$$

$$3.3R_1 = 5.1R_2$$

$$\frac{R_1}{R_2} = 1.5 \cong \frac{33k}{22k}$$

Con estos valores, cuando midamos el punto medio del divisor resistivo con el nodo RAK811, simplemente tendremos que hacer el cálculo inverso para conocer la tensión que ofrecen las pilas y así sacar su porcentaje de batería.

$$Vcc = 2.5\,Vm$$

La desventaja de monitorizar la batería es que el divisor resistivo está consumiendo la batería y esto hace que la aplicación no tenga una vida útil adecuada. Para afrontar este problema, vamos a hacer uso de un módulo compuesto por un transistor mosfet que sencillamente dejará pasar el voltaje entre sus bornas a su salida cuando el nodo RAK811 se lo ordene (activando su pin SIG), imitando el comportamiento de un interruptor y garantizando así un consumo mínimo.

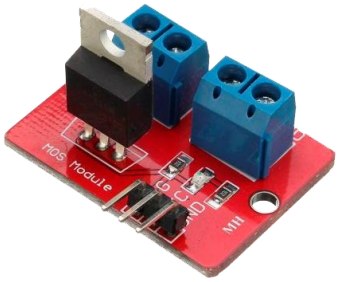


Fig. 28 – MOS Module

MOS Module			
Vin	Battery Positive Terminal	GND (2)	GND
V+	Resistor Divisor Input	V-	GND
SIG	Activator Pin	VCC	NC

Tabla 15 – Conexiones Mos Module

4 DESARROLLO HARDWARE

La conexión hace la fuerza

En este capítulo se abordan los aspectos electrónicos del desarrollo. Conectaremos todos los componentes analizados en el capítulo anterior con el objetivo de fabricar los tres nodos principales del sistema. Haremos uso de los datasheets de los componentes, indicaremos mediante tablas y esquemas las conexiones entre pines e ilustraremos el montaje final de cada nodo.

4.1 Nodo coordinador

Como ya sabemos del apartado anterior, el nodo coordinador o puerta de enlace se compone de un sistema anfitrión que será la Raspberry Pi 3 y un módulo concentrador RAK831. Para proceder a su conexión, hacemos uso del *pinout* facilitado por fabricante, donde se indica qué función realiza cada pin o terminal del circuito. A continuación, se muestra el *pinout* de la Raspberry Pi [23] y del nodo RAK831 [24].

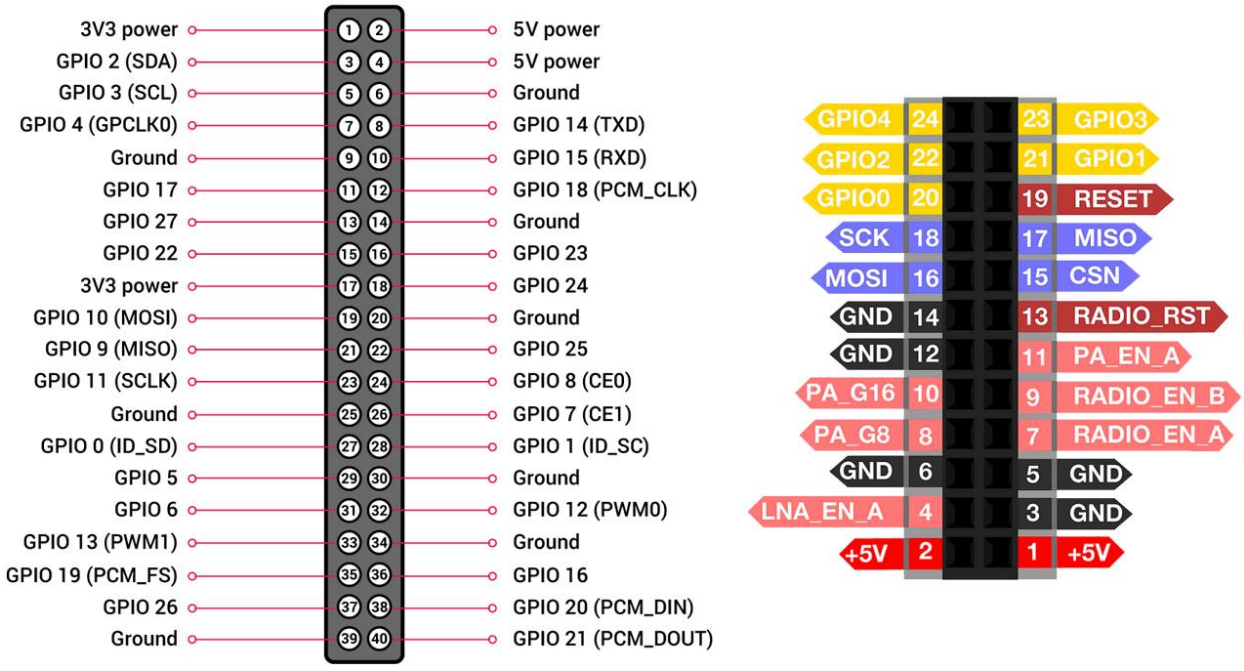


Fig. 29 – Pinout RPi3 (izquierda) y RAK831 (derecha)

Antes de conectarlos mediante SPI, debemos conocer algunas nociones de este protocolo serie. Sus siglas significan *Serial Peripheral Interface* y se trata de un protocolo síncrono (señal de reloj) que trabaja en modo *full duplex*, es decir, transmite y recibe a la vez utilizando canales distintos. Tiene una topología mastro – esclavo, en la cual el maestro transmite al esclavo utilizando la línea MOSI (*Master Out Slave In*) y a la inversa con la línea MISO (*Master In Slave Out*), siendo antes activado

el esclavo por medio de la línea SS (*Slave Select*) y utilizando la señal de reloj usando línea SCK (en el caso que nos concierne se dispondrá únicamente de un esclavo).

Conociendo esto, podemos relacionar los pines necesarios para conectar cada módulo, teniendo en cuenta los pines SPI (pines con fondo azul en el *pinout* del RAK831) y su alimentación (Vcc y GND).

RPi3	Pin Description	RAK831
2	Vcc	1
6	GND	3
24	CSN	15
19	MOSI	16
21	MISO	17
23	SCK	18
22	RST	19

Tabla 16 – Conexión entre RPi3 – RAK831

En la siguiente ilustración, se muestra su esquema de conexión, haciendo uso de la herramienta software *Fritzing.*

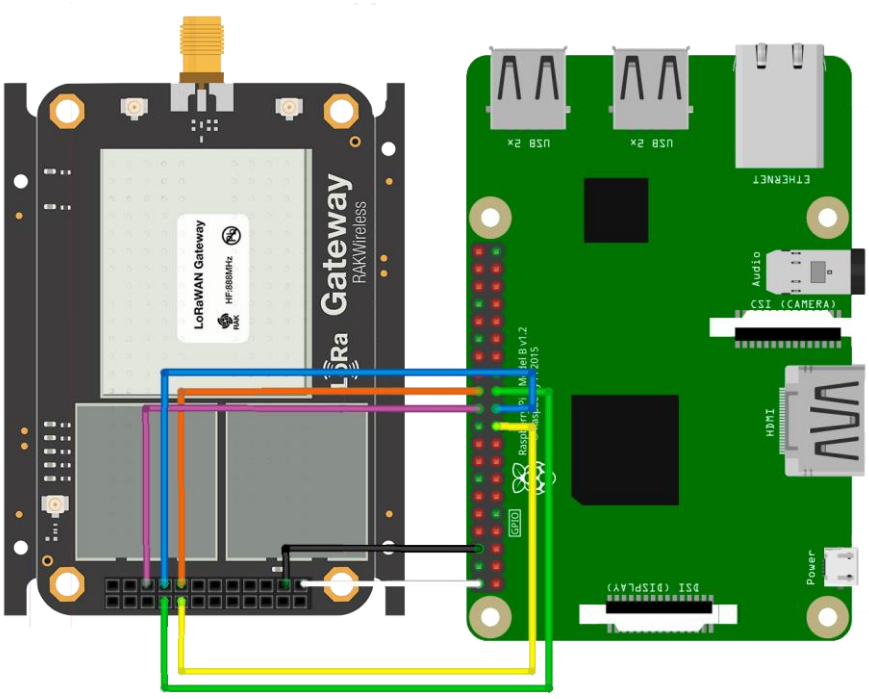


Fig. 30 – Esquema de conexión RPi3 – RAK831

Haciendo uso de una carcasa para la Raspberry y unos cables de protoboard, la conexión y montaje del nodo coordinador o *gateway* se muestra en la siguiente figura, quedando un dispositivo bastante compacto.

Fig. 31 – Montaje del nodo coordinador

Como ya vimos en la topología del sistema, este nodo se encuentra dentro de una casetilla con acceso a toma de corriente para alimentarse y acceso wifi para conectarse a la nube.

4.2 Nodo sensor

El dispositivo principal de este nodo es el chip RAK811 que contiene el código que se ejecuta de manera indefinida, obteniendo la medida de humedad del suelo y entregándosela al gateway mediante el protocolo LoRa. Este nodo se alimenta con una batería 18650.

El nodo se encuentra en estado *stand by* la mayor parte del tiempo. Cuando se consume el periodo de tiempo preestablecido, se despierta y activa el pin que alimenta el step-up, que aumenta el voltaje de 4V a 9V aproximadamente y que posteriormente se regula con el chip L7805CV a 5V, asegurando este voltaje a la entrada del sensor de humedad (que trabaja entre 3.3 y 5.5V) aunque la batería vaya decrementando su voltaje a medida que se consume.

Con los 5V ofreciéndose a la entrada del sensor, el chip RAK811 obtiene la medida de humedad y tras calcular su porcentaje, envía el dato al gateway haciendo uso de la API RUI. Cuando recibe el ACK del servidor, el nodo vuelve al estado *stand by* hasta que vuelva a consumirse el tiempo que dura el periodo.

A continuación, se muestran los pines del dispositivo RAK811, así como su pin digital equivalente (el que activamos/desactivamos en el código) [25] y una tabla de conexión entre los pines del RAK811 y los componentes externos utilizados.

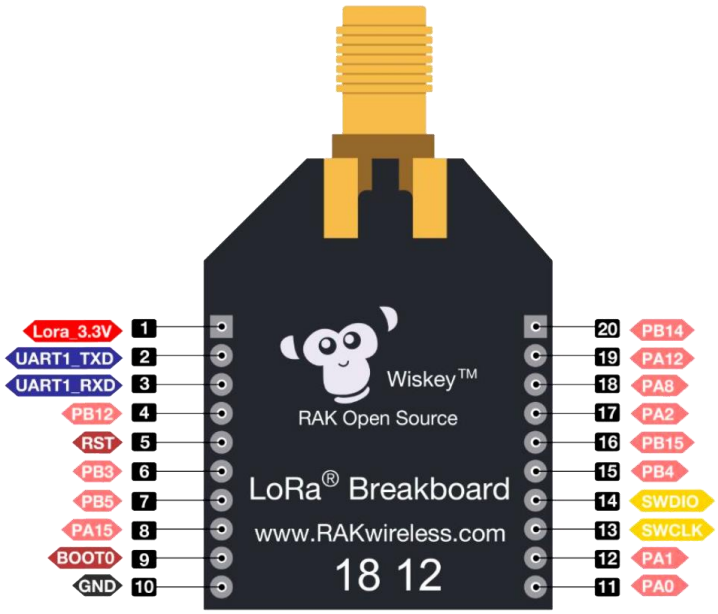


Fig. 32 – Pinout RAK811

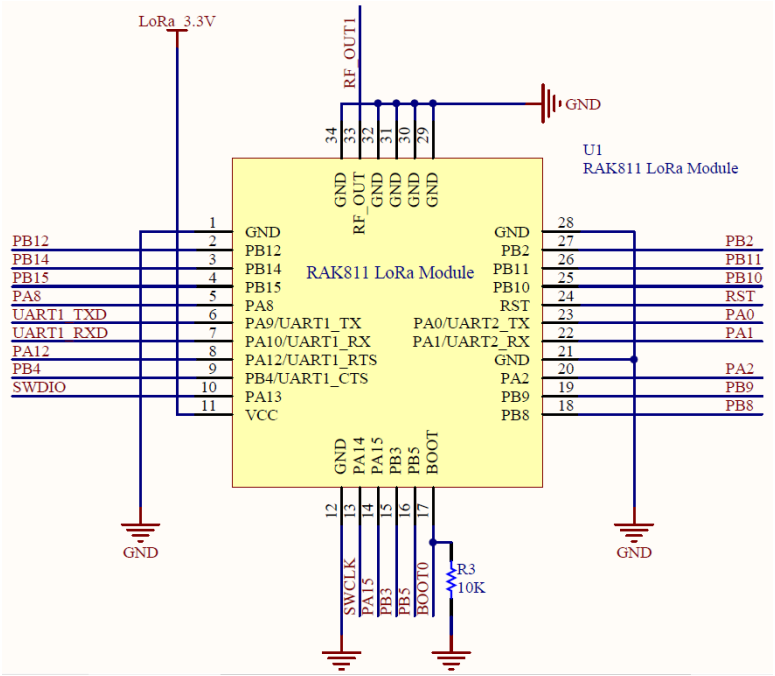


Fig. 33 – Esquema interno RAK811

RAK811	
Operating Voltage	Lora 3.3V
BOOT0 / GND	GND
Enable Sensor	PB15 (logic 4)
Get Sensor Data	PA1 (logic 22)

Tabla 17 – Conexión entre RAK811 y sus periféricos (sensor)

A continuación, se muestra el esquema de conexión del nodo sensor. Para el regulador de 5V se han utilizado condensadores electrolíticos de $10\mu F$ y $1\mu F$ a su entrada y salida para estabilizarlo.

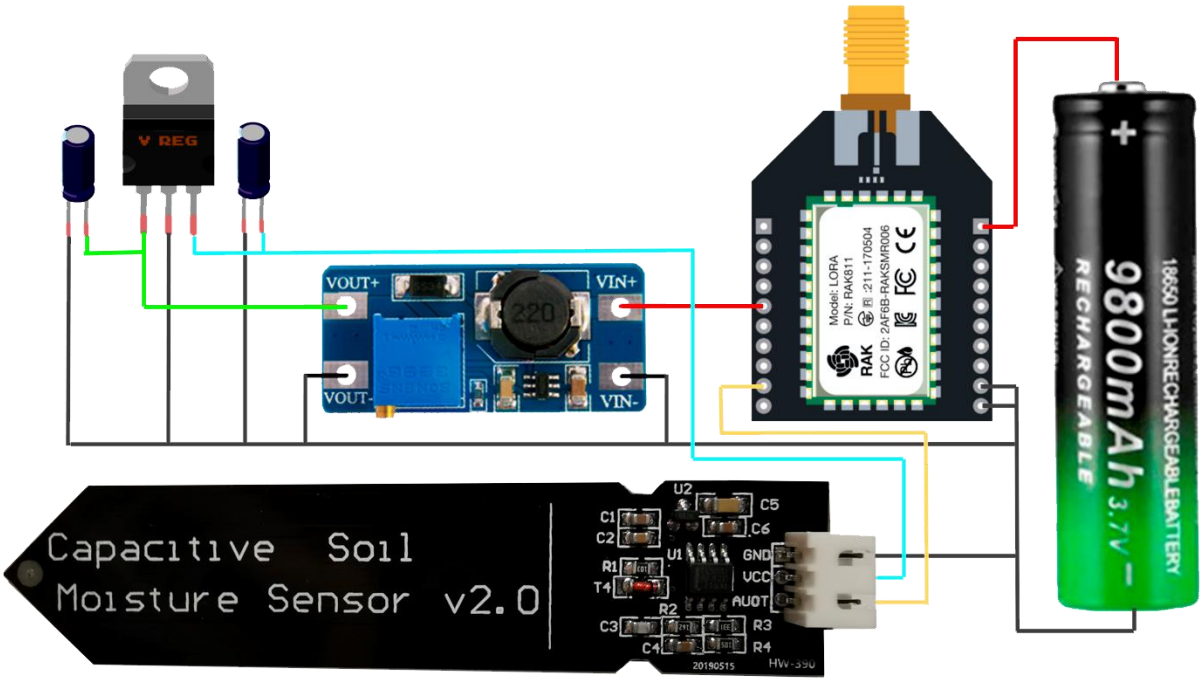


Fig. 34 – Esquema de conexión RAK811 – Periféricos (sensor)

Haciendo uso de una pequeña protoboard y conectando los elementos como se indica en la figura anterior (añadiéndole la antena al RAK811), nos queda el montaje del nodo sensor como se muestra en la siguiente figura [Fig. 35].

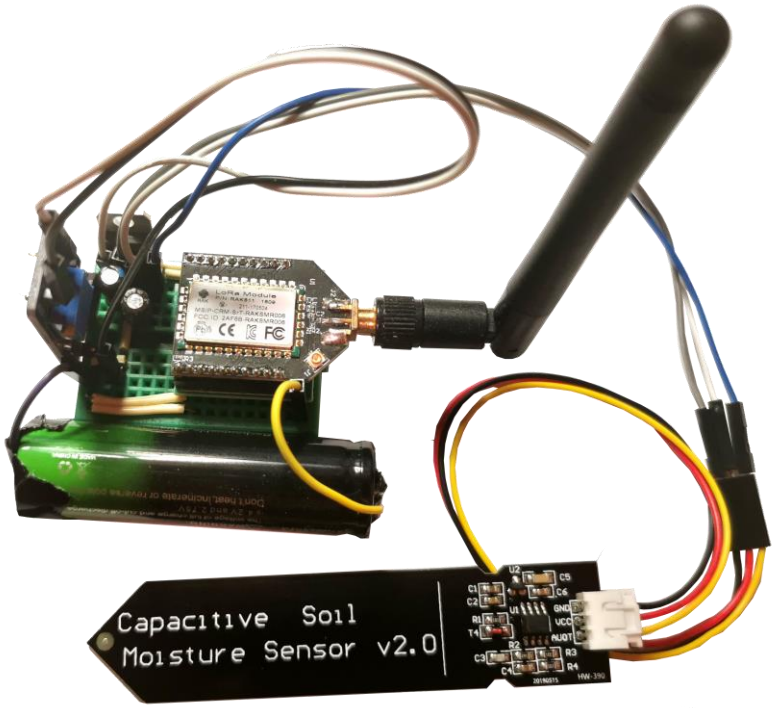


Fig. 35 – Montaje del nodo sensor

Cabe destacar que, para conectar el nodo al ordenador para transferir el código, hace falta un conversor TTL a USB. Para el proyecto se ha utilizado el dispositivo FTD1232 [Fig. 36].

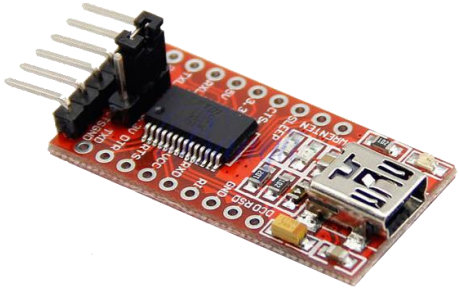


Fig. 36 – Conversor FTD1232

RAK811	FTD1232
UART1_TXD	RX
UART1_RXD	TX
Lora 3.3V	VCC
BOOT0	CTS
GND	GND

Tabla 18 – Conexión entre RAK811 – FTD1232

4.3 Nodo sensor/actuador

Este nodo, además de obtener la medida de humedad del suelo, también actúa sobre el medio activando o desactivando la electroválvula cuando sea necesario.

Al igual que el nodo sensor, se encuentra en modo *stand by* la mayor parte del tiempo, y se despierta cada periodo de tiempo establecido, toma la medida de humedad del suelo (haciendo uso del step-up y el regulador de 5V) y manda el mensaje *uplink* al servidor.

Se procesa el dato y el servidor aprovecha el mensaje ACK para indicarnos mediante una bandera si debemos encender o apagar la electroválvula. En cualquier caso, en ese momento, el nodo RAK811 activa el transistor mosfet que alimenta el divisor resistivo y toma el valor de tensión en el punto medio para obtener el valor de la batería que alimenta la electroválvula. Hecho esto, desactiva el mosfet y activa el puente H y dependiendo de la orden de encendido o apagado, le indicará al puente H que aplique la tensión de sus bornas a la salida con polaridad positiva o negativa (mediante dos pines) si se ha producido un cambio de estado.

A continuación, se detalla en una tabla la conexión entre los pines del nodo RAK811 y los componentes externos utilizados, sirviendo como guía para cuando se realice el código embebido en el nodo.

RAK811			
Operating Voltage	Lora 3.3V	Enable Mosfet	PB12 (logic 2)
BOOT0	GND	Get Battery Level	PA0 (logic 23)
GND	GND	Enable H Bridge	PB4 (logic 9)
Enable Sensor	PB15 (logic 4)	Enable IN1	PB3 (logic 15)
Get Sensor Data	PA1 (logic 22)	Enable IN2	PB5 (logic 16)

Tabla 19 - Conexión entre RAK811 y sus periféricos (sensor/actuador)

En la siguiente figura, se muestra el esquema de conexión del nodo sensor/actuador. Si trazamos una línea vertical imaginaria que pase por el centro del nodo RAK811, el espacio que queda en el lado derecho es la parte relativa al nodo sensor y en el lado izquierdo la parte de nodo actuador.

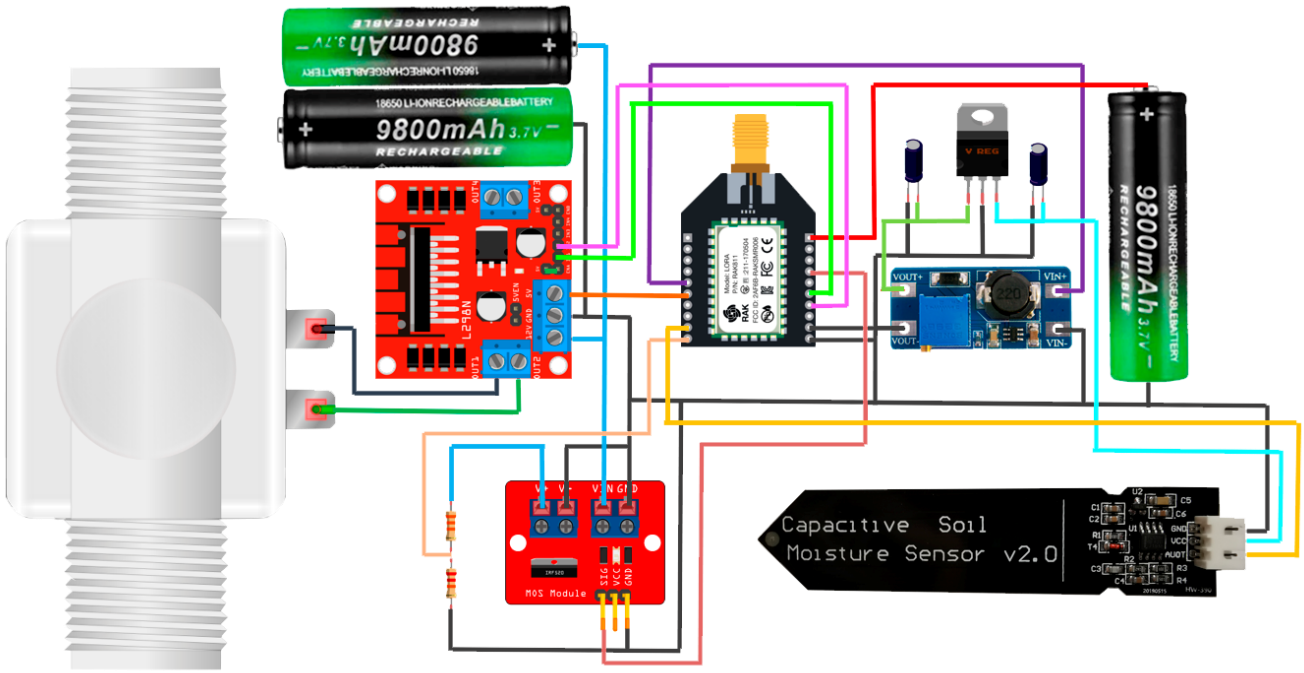


Fig. 37 - Esquema de conexión RAK811 – Periféricos (sensor/actuador)

Por último, procedemos al montaje físico del componente teniendo en cuenta el esquema de conexión y la tabla de conexión entre pines. Para ello, haremos uso de dos mini protoboards y conectaremos la antena al nodo RAK811.

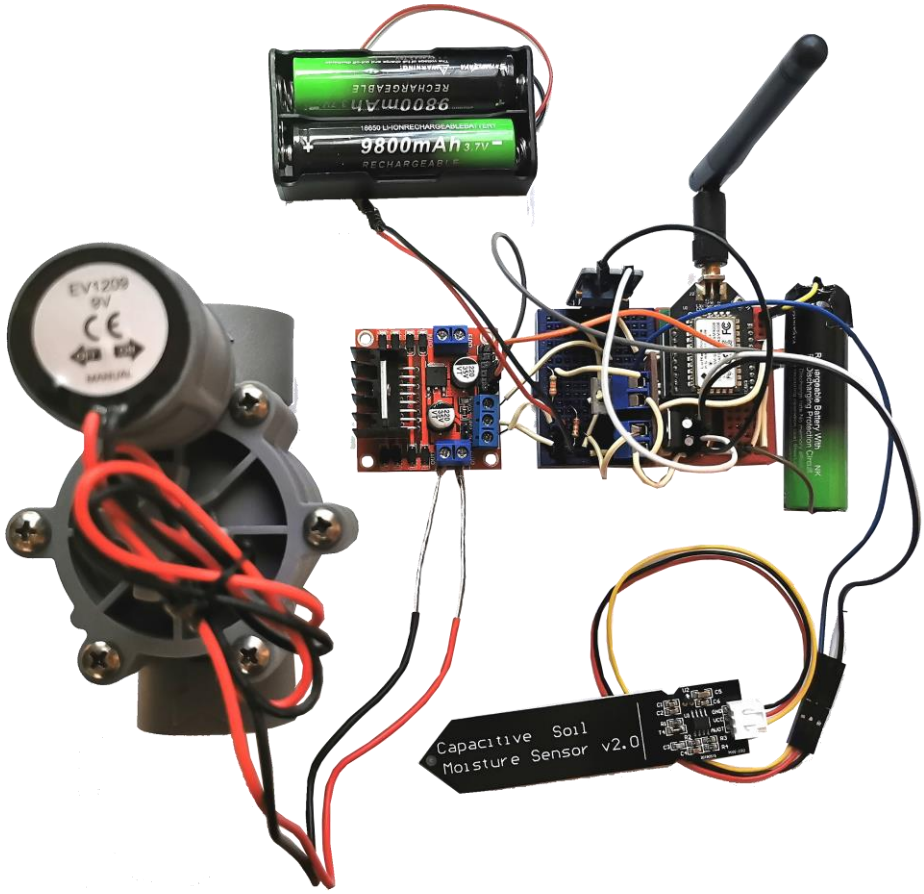


Fig. 38 – Montaje del nodo sensor/actuador

Llegados a este punto, disponemos de toda la infraestructura física necesaria para afrontar el proyecto. El siguiente paso consiste en la programación de los nodos y la configuración del servidor.

5 DESARROLLO SOFTWARE

No existe inteligencia sin lógica

En este capítulo se abordan los aspectos telemáticos del desarrollo. Por un lado, se detalla la configuración del nodo gateway, instalación e integración de los servicios necesarios, así cómo la lógica que ejecutará el servidor en el intercambio de mensajes con los nodos. Por otro lado, se explica el código de los nodos en sus dos versiones (sensor y sensor/actuador), así como su compilación e integración en el chip.

5.1 Nodo coordinador

En este apartado se va a implementar el *gateway*, desde la instalación del sistema operativo hasta la implantación de los servidores de red y aplicación, así como la configuración e integración entre ellos.

5.1.1 Configuración del sistema anfitrión

Como ya se ha comentado, se va a utilizar la distribución de Linux Raspbian como sistema operativo por estar optimizado para el hardware de la placa. Para instalar dicho sistema, haremos uso de una tarjeta microSD y del software *etcher* [26].

Descargamos el sistema operativo de la página web de Raspberry [27], insertamos la tarjeta en el ordenador y ejecutamos el programa *etcher*, siguiendo sus tres sencillos pasos:

1. Seleccionar la imagen que hemos descargado previamente.
2. Seleccionar la unidad que corresponde con nuestra tarjeta microSD.
3. Hacer click en flashear.

Una vez finalizado el proceso, tendremos lista la tarjeta microSD con Raspbian instalado. Insertamos dicha tarjeta a la Raspberry y conectamos un monitor, teclado y ratón para configurar el sistema.

Cuando se haya iniciado el sistema, procedemos a realizar las siguientes configuraciones:

- Para poder conectar la Raspberry con el concentrador, debemos habilitar el protocolo SPI (*Serial Peripheral Interface*). Además, para poder conectarnos a la Raspberry por remoto, necesitaremos habilitar también el protocolo SSH. Abrimos el menu de inicio, seleccionamos P*references* y *Raspberry Pi Configuration.* En la pestaña *Interfaces,* establecemos las opciones SSH y SPI a *enabled*.
- Para establecer la distribución del teclado en español, en la pestaña *Localisation,* seleccionamos el lenguaje español en las opciones que aparecen.
- Conectar la Raspberry a la red wifi de la que se disponga mediante el icono de wifi de la parte superior (si estamos en el interfaz gráfico) o agregando el siguiente bloque de código al final del fichero **wpa_supplicant.conf (**suponiendo seguridad WPA**)** y reiniciando el sistema.

```
$ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf

network={
  ssid="nombre-de-tu-wifi"
  psk="password-de-tu-wifi"
  key_mgmt=WPA-PSK
}
```

A partir de ahora, utilizaremos la Raspberry en remoto, haciendo uso del software *WinSCP*, con el que podemos acceder al sistema de ficheros mediante el protocolo SFTP y a la consola de comandos mediante el protocolo SSH. Para conocer la IP de la Raspberry, usamos el comando:

```
$ hostname -I
```

5.1.2 Instalación del gateway

Para que nuestro controlador tenga los conocimientos del protocolo LoRa y asuma las funciones de reenvío de paquetes, debemos instalar la librería LoRa (v5.0.1) y el *UDP packet forwarder* (v4.0.1).

El *packet forwarder* es un servicio ejecutándose en el host del gateway (nuestra RPi3) que se relaciona con el concentrador (RAK831) haciendo pull y push a los paquetes, miestras que interactúa a su vez con el servidor.

Antes de comenzar con la instalación, actualizamos todos los paquetes del sistema:

```
$ sudo apt-get update && sudo apt-get upgrade -y
```

Instalamos git para traernos los repositorios de los instalables:

```
$ sudo apt-get install git
```

Clonamos el siguiente repositorio, que instalará ambas funcionalidades y a su vez creará y habilitará en el inicio (systemd) el servicio **ttn-gateway** [28]:

```
$ git clone https://github.com/robertlie/RAK831-LoRaGateway-RPi
~/rak831-loragateway
$ cd ~/rak831-loragateway
$ sudo ./install.sh
```

Durante la instalación, se indica la dirección EUI del *gateway*, que deberemos guardar para registrar posteriormente dicha puerta de enlace en el servidor.

Asimismo, tenemos la opción de seleccionar un nombre para el sistema, de indicar la latitud, longitud y altitud de la ubicación en la que se encontrará el *gateway* y la región (indicamos EU).

Al acabar la instalación, podemos comprobar que el servicio del concentrador está levantado con el comando:

```
$ systemctl status ttn-gateway -l
```

```
pi@mairena-LoRa:~ $ systemctl status ttn-gateway.service -l
● ttn-gateway.service - The Things Network Gateway
   Loaded: loaded (/lib/systemd/system/ttn-gateway.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2020-06-20 20:27:19 CEST; 2 weeks 0 days ago
 Main PID: 1181 (start.sh)
    Tasks: 6 (limit: 4915)
   CGroup: /system.slice/ttn-gateway.service
           ├─1181 /bin/bash /opt/ttn-gateway/packet_forwarder/lora_pkt_fwd/start.sh
           └─1192 ./lora_pkt_fwd
```

Fig. 39 – Servicio ttn-gateway

La configuración del concentrador se encuentra en los ficheros **global_conf.json** y **local_conf.json**. Si algún objeto de la configuración coincide en ambos ficheros, prevalece el local. Contienen ajustes como la dirección del servidor de red, los puertos de enlace de subida y bajada o la frecuencia de operación, entre otros.

El fichero **global_conf.json** contiene un objeto "SX1301_conf" (el chip del concentrador), que contiene los parámetros de frecuencia de operación. Este fichero podemos descargarlo del github de TheThingsNetwork, estando ya configurado para la frecuencia europea (868MHz) [29].
El fichero **local_conf.json** contiene el objeto "gateway_conf", con los parámetros específicos del concentrador. Editamos el fichero:

```
$ cd /opt/ttn-gateway/packet_forwarder/lora_pkt_fwd/
$ sudo nano local_conf.json
```

Y ajustamos lo siguiente:

- **gateway_ID:** Dirección EUI guardada en la instalación.
- **server_address:** Dirección IP del servidor de red (en nuestro caso localhost).
- **serv_port_up:** 1700
- **serv_port_down:** 1700

Una vez realizados los cambios, para que se hagan efectivos, debemos reiniciar el concentrador:

```
$ sudo service ttn-gateway restart
```

5.1.3 Instalación de ChirpStack / LoRa Server

Teniendo en cuenta el estudio previo realizado en el apartado 2.5.2, ChirpStack se resume en tres componentes principales:

- **LoRa Gateway Bridge:** Convierte el dato que proviene del protocolo UDP por el *packet forwarder* en formato json y se lo transmite al LoRa Server mediante el protocolo MQTT. Esto se realiza mediante un MQTT broker, en el cual, el puente publica sus paquetes y el LoRa Server los recibe por estar suscrito a los mismos.
- **LoRa Server:** Es el servidor en sí mismo, que se encarga de manejar las tramas de subida y bajada de enlace haciendo uso de la capa LoRaWAN.
- **LoRa App Server:** Maneja las peticiones de join, la encriptación de paquetes y ofrece una API REST, gRPC y MQTT para interactuar con otros servicios. Tiene además una interfaz web donde poder ver los paquetes recibidos.

Existen diversas posibilidades de formar la pila de ChirpStack, en el proyecto vamos a utilizar la que se muestra en la siguiente figura [Fig. 40], con la pila completa dentro del servidor, que al ser éste la propia Raspberry, tendremos tanto el concentrador como el servidor LoRa en la misma máquina local.

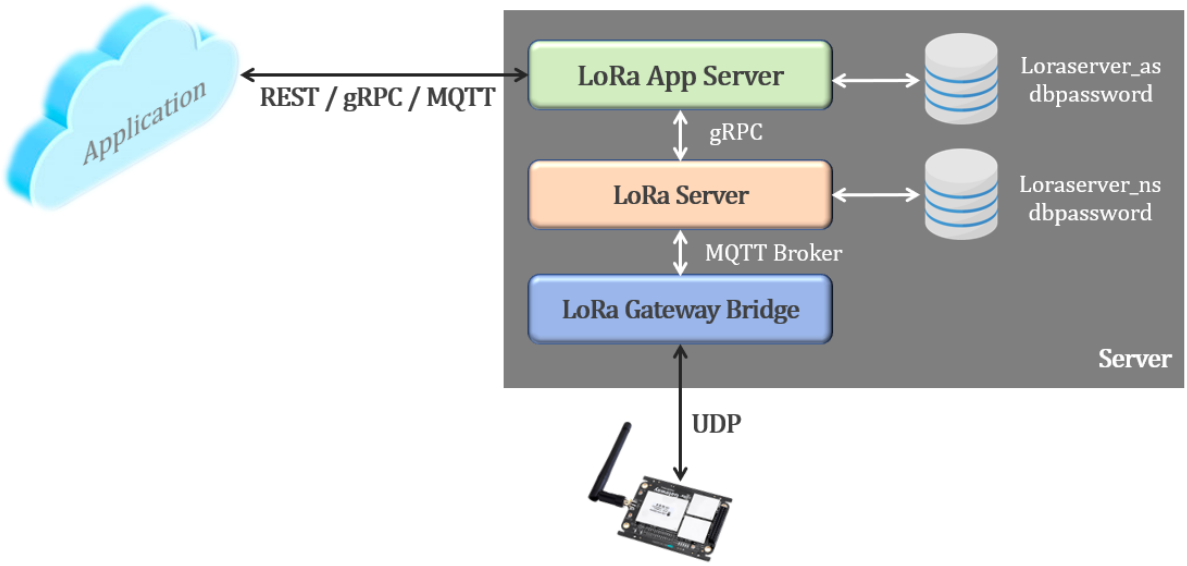


Fig. 40 – Pila ChirpStack utilizada

Procedemos a instalar la pila completa de ChirpStack haciendo uso del manual ofrecido por la misma entidad en su web [30].

Como ya sabemos, el gateway bridge se comunica con el componente loRa server a través de un MQTT broker. Lo instalamos en la RPi3 y configuramos su servicio en el inicio como sigue:

```
$ sudo apt-get install mosquitto
$ sudo systemctl start mosquitto
$ sudo systemctl enable mosquitto
$ sudo systemctl status mosquitto
```

```
pi@mairena-LoRa:~ $ sudo systemctl status mosquitto
● mosquitto.service - LSB: mosquitto MQTT v3.1 message broker
   Loaded: loaded (/etc/init.d/mosquitto; generated; vendor preset: enabled)
   Active: active (running) since Mon 2020-07-06 07:21:38 CEST; 1 day 10h ago
     Docs: man:systemd-sysv-generator(8)
  Process: 420 ExecStart=/etc/init.d/mosquitto start (code=exited, status=0/SUCCESS)
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/mosquitto.service
           └─493 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf
```

Fig. 41 – Servicio MQTT

Activamos el repositorio que contiene los paquetes binarios del proyecto LoRa Server:

```
$ sudo apt install apt-transport-https dirmngr
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
1CE2AFD36DBCCA00
$ sudo echo "deb https://artifacts.loraserver.io/packages/2.x/deb
stable main" | sudo tee /etc/apt/sources.list.d/loraserver.list

$ sudo apt-get update
```

Instalamos el gateway bridge, editamos su fichero de configuración **lora-gateway-bridge.toml** para indicar que el servidor MQTT se encontrará en localhost (127.0.0.1) y en su puerto por defecto (1883) e iniciamos su servicio:

```
$ sudo apt-get install lora-gateway-bridge
```

```
$ sudo nano /etc/lora-gateway-bridge/lora-gateway-bridge.toml

      server="tcp://127.0.0.1:1883"

$ sudo systemctl start lora-gateway-bridge
$ sudo systemctl enable lora-gateway-bridge

$ sudo systemctl status lora-gateway-bridge
```

```
pi@mairena-LoRa:~ $ sudo systemctl status lora-gateway-bridge
● chirpstack-gateway-bridge.service - ChirpStack Gateway Bridge
   Loaded: loaded (/lib/systemd/system/chirpstack-gateway-bridge.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2020-07-06 07:21:38 CEST; 1 day 10h ago
     Docs: https://www.chirpstack.io/
 Main PID: 498 (chirpstack-gate)
    Tasks: 12 (limit: 4915)
   CGroup: /system.slice/chirpstack-gateway-bridge.service
           └─498 /usr/bin/chirpstack-gateway-bridge
```

Fig. 42 – Servicio lora gateway bridge

El servidor LoRa requiere una base de datos en la que almacenar los paquetes recibidos. Para ello, se van a utilizar el sistema de base de datos **PostgreSQL** y la base de datos de **redis**. Una vez instalados, creamos un owner para la base de datos y un *database* en sí (loraserver_ns para ambos).

```
$ sudo apt-get install postgresql
$ sudo apt-get install redis-server
$ sudo -u postgres psql

      create role loraserver_ns with login password 'dbpassword';
      create database loraserver_ns with owner loraserver_ns;
      \q
      psql -h localhost -U loraserver_ns -W loraserver_ns
      \q
```

Instalamos el servidor LoRa, editamos su fichero de configuración **loraserver.toml** para indicar la conexión a la BBDD creada, frecuencia europea y conexión al MQTT broker e iniciamos su servicio:

```
$ sudo apt-get install loraserver
$ sudo nano /etc/loraserver/loraserver.toml

      dsn="postgres://loraserver_ns:dbpassword@localhost/loraserver_ns?
      sslmode=disable"
      automigrate=true
      name="EU_863_870"
      timezone="Local"
      server="tcp://127.0.0.1:1883"

$ sudo systemctl start loraserver
$ sudo systemctl enable loraserver

$ sudo systemctl status loraserver
```

```
pi@mairena-LoRa:~ $ sudo systemctl status loraserver
● chirpstack-network-server.service - ChirpStack Network Server
   Loaded: loaded (/lib/systemd/system/chirpstack-network-server.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2020-07-06 07:21:38 CEST; 1 day 11h ago
     Docs: https://www.chirpstack.io/
 Main PID: 499 (chirpstack-netw)
    Tasks: 13 (limit: 4915)
   CGroup: /system.slice/chirpstack-network-server.service
           └─499 /usr/bin/chirpstack-network-server
```

Fig. 43 – Servicio LoRa Server

El LoRa App Server también dispone de una base de datos propia, por lo que procedemos a realizar la misma operación (loraserver_as para owner y BBDD). En este caso, añadiremos la extensión pg_trgm (trigram) a la base de datos, que le aportará más velocidad en la comparativa de literales.

```
$ sudo -u postgres psql
      create role loraserver_as with login password 'dbpassword';
      create database loraserver_as with owner loraserver_as;
      \c loraserver_as
      create extension pg_trgm;
      \q
      psql -h localhost -U loraserver_as -W loraserver_as
      \q
```

Instalamos el servidor App LoRa, creamos un token (jwt – *java web token*) con el comando **openssl** y editamos su fichero de configuración **lora-app-server.toml**, indicando la conexión a la BBDD creada previamente, el token que acabamos de generar, la conexión con el MQTT server (si queremos usar este protocolo con una app externa), la conexión interna que comunica el lora server con el lora app server mediante el puerto 8001, la conexión con la interfaz web que levantaremos en el puerto 8081 (dejaremos el 8080 para la interfaz web de Thingsboard) e iniciamos su servicio:

```
$ sudo apt-get install lora-app-server

$ openssl rand -base64 32

$ sudo nano /etc/lora-app-server/lora-app-server.toml


dsn="postgres://loraserver_as:dbpassword@localhost/loraserver_as?
sslmode=disable"
jwt_secret="lGS2EvRnagmivQoV.....l43Xs85fYM1iNFNROEGkWvJzZEmo"
server="tcp://localhost:1883"
public_host="localhost:8001"
bind="0.0.0.0:8081"

$ sudo systemctl start lora-app-server
$ sudo systemctl enable lora-app-server

$ sudo systemctl status lora-app-server
```

```
● chirpstack-application-server.service - ChirpStack Application Server
   Loaded: loaded (/lib/systemd/system/chirpstack-application-server.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2020-07-06 07:21:38 CEST; 1 day 11h ago
     Docs: https://www.chirpstack.io/
 Main PID: 496 (chirpstack-appl)
    Tasks: 15 (limit: 4915)
   CGroup: /system.slice/chirpstack-application-server.service
           └─496 /usr/bin/chirpstack-application-server
```

Fig. 44 – Servicio LoRa App Server

5.1.4 Instalación de Thingsboard

Como ya se ha comentado en el apartado 2.6.1, vamos a utilizar el servidor de aplicación de Thingsboard. Para ello, vamos a instalar dicho software en la RPi3 como se indica en la propia web de la marca [31]:

Comprobamos que tenemos java 8 y descargamos los paquetes correspondientes:

```
$ java -version
$ wget https://github.com/thingsboard/thingsboard/releases/
      download/v3.0.1/thingsboard-3.0.1.deb
$ sudo dpkg -i thingsboard-3.0.1.deb
```

Creamos una base de datos propia para la aplicación en nuestro PostgreSQL:

```
$ psql -U postgres -d postgres -h 127.0.0.1 -W
      CREATE DATABASE thingsboard;
$ \l
$ \q
```

Añadimos al fichero de configuración **thingsboard.conf** la conexión con base de datos y restringimos el uso de memoria a 256MB:

```
$ sudo nano /etc/thingsboard/conf/thingsboard.conf

      export DATABASE_ENTITIES_TYPE=sql
      export DATABASE_TS_TYPE=sql
      export SPRING_JPA_DATABASE_PLATFORM=
            org.hibernate.dialect.PostgreSQLDialect
      export SPRING_DRIVER_CLASS_NAME=org.postgresql.Driver
      export SPRING_DATASOURCE_URL=
            jdbc:postgresql://localhost:5432/thingsboard
      export SPRING_DATASOURCE_USERNAME=postgres
      export SPRING_DATASOURCE_PASSWORD=dbpassword
      export JAVA_OPTS="$JAVA_OPTS -Xms256M -Xmx256M"
```

Nos aseguramos de que la web se lanza en el puerto 8080 y como Thingsboard también necesita su propio MQTT, cambiamos el puerto por defecto en el que se va a ejecutar al 1884 en su fichero de configuración **thingsboard.yml:**

```
$ sudo nano /etc/thingsboard/conf/thingsboard.yml

      server:
            port: "${HTTP_BIND_PORT:8080}"
      mqtt:
            bind_port: "${MQTT_BIND_PORT:1884}"
```

Por último, cargamos los datos por defecto de tenant, usuarios y dispositivos e iniciamos el servicio:

```
$ sudo /usr/share/thingsboard/bin/install/install.sh --loadDemo
$ sudo service thingsboard start
$ sudo service thingsboard status
```

```
pi@mairena-LoRa:~ $ sudo service thingsboard status
● thingsboard.service - thingsboard
   Loaded: loaded (/lib/systemd/system/thingsboard.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2020-07-07 19:06:51 CEST; 23h ago
 Main PID: 365 (thingsboard.jar)
    Tasks: 75 (limit: 4915)
   CGroup: /system.slice/thingsboard.service
           ├─365 /bin/bash /usr/share/thingsboard/bin/thingsboard.jar
           └─408 /usr/bin/java -Dsun.misc.URLClassPath.disableJarChecking=true -Dplatform=deb
```

Fig. 45 – Servicio Thingsboard

5.1.5 Configuración e integración de los servicios

Llegados a este punto, tenemos desplegados los servicios de ChirpStack y Thingsboard en nuestra RPi3. Si nos encontramos en la misma red que la RPi3, podemos entrar en las webs a través de la IP que haya asignado el router a la RPi3. Esta IP podemos averiguarla con el comando **hostname -I** en la propia RPi3 o accediendo a la página de configuración del router.

Una vez autenticado el usuario en la interfaz web de ChirpStack (puerto 8081), para configurar correctamente el Gateway, debemos seguir los siguientes pasos:

1. Pestaña **Network-servers**:
    a. Creamos una entrada nueva, indicando el nombre para el servidor LoRa que hemos instalado y su dirección (localhost:8000).
    b. Se deja sin marcar "Gateway discovery" ya que solo tenemos una puerta de enlace.
2. Pestaña **Organización:**
    a. Indicamos el nombre para la organización y establecemos la opción de que puede tener gateways.
3. Pestaña **Service-Profiles:**
    a. El perfil de servicio conecta la organización al servidor configurados previamente.
    b. Marcamos la opción "Add Gateway metadata", para que en las tramas enviadas al servidor de aplicación tengan en sus cabeceras el RSSI, SNR, etc.
    c. Desmarcamos "Enable network geolocation", ya que no disponemos de módulo GPS en los nodos ni la aplicación requiere geolocalización.
4. Pestaña **Device-Profiles:**
    a. El perfil de dispositivo define las capacidades y parámetros de inicio de un dispositivo.
    b. Lo asociamos a nuestro servidor.
    c. LoRaWAN MAC version: 1.0.2 (versión MAC soportada en el datasheet del nodo RAK811).
    d. Marcamos la opción "Device supports **OTAA**".

En "Payload codec" marcamos la opción "**Cayenne LPP**". Ésta será la codificación que vamos a seguir en las tramas por su sencillez y fidelidad con las restricciones de tamaño de trama (hasta 11 bytes) [32].

5. Pestaña **Gateways:**
    a. Indicamos un nombre para nuestra puerta de enlace (SmartHumidity_Gateway).
    b. En "Gateway ID", indicamos su código **EUI** (el que se obtiene al instalar ttn-gateway en la RPi3). En nuestro caso "b827ebfffeaf5659".
    c. Asociamos con el servidor y perfil de Gateway configurados previamente.
    d. En Gateway location, marcamos en el mapa el punto final en el que se encontrará nuestro sistema de riego.
6. Pestaña **Applications:**
    e. Indicamos un nombre para nuestra aplicación.

f. La asociamos con el perfil de servidor configurado previamente.
g. Seleccionamos como codificación "**Cayenne LPP**".
h. En "Devices", creamos uno nuevo (nodo1), asociamos al perfil de dispositivo configurado previamente y dejamos que se establezca un "**Device EUI**" y un "**Application Key**" automáticos.
i. Se repite el paso anterior para el número de nodos que se desee.

Nos conectamos ahora a la interfaz web de Thingsboard (puerto 8080) con el rol "tenant" (administrador de la organización):

7. Pestaña **Dispositivos**
    a. Añadimos un nuevo dispositivo (nodo1)
    b. Abrimos los detalles del dispositivo, y copiamos el "**Token de Acceso**"

Volvemos a la interfaz web de Chirpstack y configuramos el token:

8. Pestaña **Applications**
    c. En la sección "INTEGRATIONS", añadimos la conexión con Thingsboard:
        - Integration kind: ThingsBoard.io
        - Thisboard.io server: **http://localhost:8080**
    d. Dentro de la configuración del nodo a integrar con Thingsboard, en la sección "variables", añadimos:
        - Name: **ThingsBoardAccessToken**
        - Value: Token de acceso copiado en el paso 1.b

Se repiten los pasos 1 y 2b para el número de nodos que se desee.

5.1.6 Lógica del servidor

El verdadero valor añadido de la aplicación Thingsboard con respecto a las demás es su motor de cadena de reglas. Es un sistema personalizable y configurable para el procesamiento de eventos complejos. Ofrece tareas de filtrado y transformación de mensajes originados por dispositivos IoT y activos relacionados, además de facilitar la comunicación con sistemas externos.

En la siguiente figura [Fig. 46] se muestra la lógica del servidor de manera funcional. Tal y como funciona el sistema, a medida que llega cualquier mensaje de alguno de los nodos vinculados previamente, comienza el flujo de eventos del motor de reglas de Thingsboard.

A la llegada de cualquier mensaje, se comprueba si el contenido de dicho mensaje es el que nos interesa, es decir, la humedad del suelo. Si es así, se identifica el nodo origen con la información del metadata del mensaje. Para poder calcular la media aritmética con el resto de las medidas de humedad de los demás nodos, se hace uso de un bloque que abstrae el flujo al asset, es decir, podemos acceder tanto a la información del nodo en cuestión como a la de los demás nodos pertenecientes al asset [33]. Una vez recopiladas todas las medidas, se calcula la media aritmética y se almacena el resultado.

A partir de este momento, solo continúa el flujo el nodo sensor/actuador (bloques color naranja en el diagrama), ya que será éste el que reciba el mensaje hacia abajo (downlink) para encender/apagar la electroválvula.

Para interactuar con ChripStack, haremos uso de su API REST. Para que la conexión sea segura, primero debemos obtener un token de acceso (endpoint **internal/login**) [34] y posteriormente utilizarlo como cabecera para utilizar la función de encolar el mensaje downlink (endpoint **devices/${devEUI}/queue**) [35].

Como utilidad adicional al proceso, se realiza una consulta a la API externa **openweathermap** [36], haciendo uso de su endpoint **data/2.5/forecast** y pasándole como parámetros:

- **id:** 1686940 (Sevilla)
- **units**: metric (grado centígrado o Celsius – Cº)
- **cnt**: 2 (número de líneas devueltas)
- **appid**: 621c3752ab396fde703a14d7c29f9ab4 (hay que registrarse en la aplicación para obtenerlo)

Esta petición nos devolverá un objeto json, del cual nos centraremos en el weather.id y su rango de valores (500-532), perteneciente a condiciones climatológicas de lluvia [37].

Si está lloviendo o va a llover próximamente, mandamos un mensaje downlink de disable al nodo actuador. Si existen buenas condiciones sin lluvia aparente, comprobamos la media aritmética calculada de las medidas de humedad obtenidas de los sensores y la comparamos con un umbral (por ejemplo 30%), si dicha medida se encuentra por debajo del umbral, mandamos un enable (suelo seco) y si se encuentra por encima un disable (suelo muy húmedo).

Fig. 46 – Lógica funcional del servidor

En la siguiente figura [Fig. 47], se muestra la implementación del flujo en el motor de cadena de reglas de Thingsboard, haciendo uso de los distintos bloques de los que dispone.

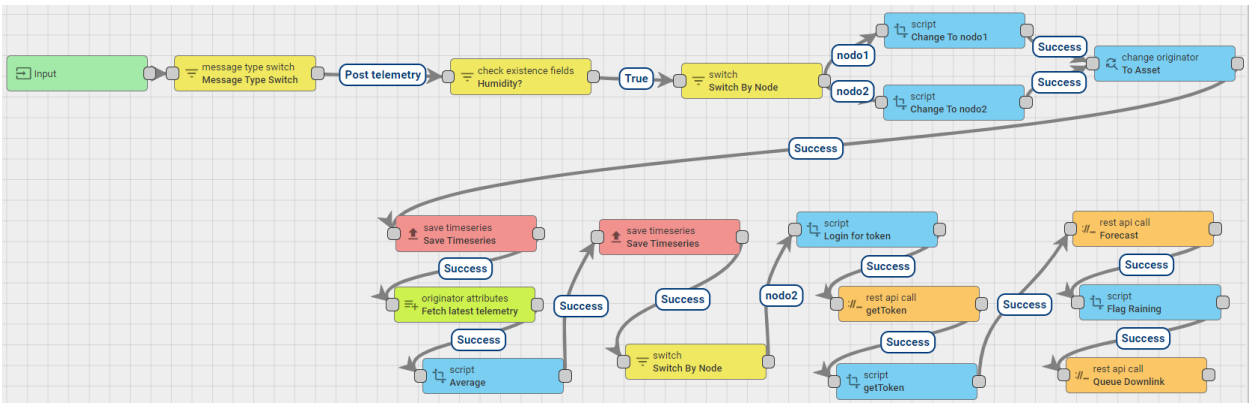


Fig. 47 – Lógica del servidor en el motor de cadena de reglas de Thingsboard

- **Input**: Bloque de inicio del flujo.
- **Message Type Switch**: Nos interesan los mensajes de tipo **Post Telemetry.**
- **Check existence fields – Humidity:** Comprueba si el Message data = "data_humidity_sensor_1" (valor por defecto resultante de usar 0x68 con decodificación Cayenne LPP).
- **Switch by node:** Diferencia el nodo origen:

```
function nextRelation(metadata, msg) {
      if (metadata.deviceName === 'nodo1') {
            return ['nodo1'];
      } else if (metadata.deviceName === 'nodo2')
            return ['nodo2'];
}

return nextRelation(metadata, msg);
```

- **Change To nodo1:** Añadimos al mensaje su nodo origen:

```
var newMsg = {};

newMsg.nodo1 = msg.data_humidity_sensor_1;

return {
   msg: newMsg,
   metadata: { "deviceName": metadata.deviceName },
   msgType: msgType

};
```

- **Change originator – To Asset:** Subimos el nivel de dispositivo a activo para tener acceso a las medidas del resto de nodos.
    - **Originator source:** Related
    - **Contains:** Activo
- **Save Timeseries:** Almacenamos valor en base de datos.
- **Fetch latest telemetry:** Obtenemos las últimas medidas almacenadas en base de datos de los distintos nodos del activo.
    - **Latest timeseries:** nodo1, nodo2
- **Average:** Calculamos la media aritmética entre las distintas medidas de los nodos:

```
var newMsg = {};

newMsg.Avg = (Math.abs(metadata.nodo1) +
Math.abs(metadata.nodo2))/2 ;

return {
    msg: newMsg,
    metadata: metadata,
    msgType: msgType
};
```

- **Switch By Node:** Continúa el flujo únicamente el nodo actuador.
- **Script - Login for token:** Preparamos el mensaje para hacer uso del endpoint de login de Chirpstack mediante su API REST:

```
msg.username = "username";
msg.password = "password";

if (msg.Avg !== 'undefined') {
    metadata.hum = msg.Avg;
}

return {
    msg: msg,
    metadata: metadata,
    msgType: msgType
};
```

- **Rest api call - getToken:** Realizamos la petición de login usando la API de Chirpstack.
    - **Endpoint URL pattern:** http://localhost:8080/api/internal/login
    - **Request method:** POST
    - **Header: Content-Type:** application/json
- **Script - getToken:** Almacenamos el token obtenido en el metadata del flujo:

```
metadata.jwt = msg.jwt;

return {
    msg: msg,
    metadata: metadata,
    msgType: msgType
};
```

- **Rest api call - Forecast:** Hacemos una petición a la API de openweathermap como se ha comentado previamente.
    - **Endpoint URL pattern:** http://api.openweathermap.org/data/2.5/forecast?id=6361014&units=metric&cnt=2&appid=621c3752ab396fde703a14d7c29f9ab4
    - **Request method:** POST
- **Script – Flag Raining:** Realizamos la lógica comentada según la respuesta de previsión de lluvia y la media aritmética obtenida por las medidas de los nodos:

```
metadata.devEUI = "3530353083377318";

if(msg.list[0].weather[0].id > 500 && msg.list[0].weather[0].id <
532){
    msg = {};
    msg.deviceQueueItem = {
        "fPort": 1,
        "data": "0000"
    };
} else {
    msg = {};
    if (metadata.hum !== 'undefined') {
        if (metadata.hum < 30) {
            msg.deviceQueueItem = {
                "fPort": 1,
                "data": "ffff"
            };
        } else {
            msg.deviceQueueItem = {
                "fPort": 1,
                "data": "0000"
            };
        }
    }
}

return {
    msg: msg,
    metadata: metadata,
    msgType: msgType

};
```

- **Rest api call – Queue Downlink:** Hacemos uso del endpoint de la API de ChirpStack para poner en cola el mensaje downlink.
    - **Endpoint URL pattern:** http://localhost:8080/api/devices/${devEUI}/queue
    - **Request method:** POST
    - **Header - Grpc-Metadata-Authorization:** Bearer ${jwt}
    - **Header – Content-Type:** application/json

5.2 Nodos sensor y sensor/actuador

La lógica del nodo podría situarse en un procesador externo (como un arduino) que controlara el RAK811 para enviar y recibir mensajes, pero esto tendría un consumo elevado ya que la batería tendría que alimentar dos componentes (además de la circuitería). En su lugar, la lógica estará integrada dentro del propio RAK811, programando directamente su controlador de ultra bajo consumo STM32L y ayudándonos de las herramientas que nos ofrece RAKWireless [38]:

- **RUI Online Compiler** [39]: Para compilar nuestro código escrito en C y basado en el framework RUI (RAK Unified Interface).
- **STM32 Cube Programmer**: Para resetear el nodo y embeber el firmware.
- **RAK LoRaButton Upgrade Tool**: Para actualizar el firmware del nodo con nuestro código compilado.
- **RAK Serial Port Tool**: Para realizar pruebas con el nodo usando UART y el puerto serie de un ordenador.

5.2.1 Código

Como se ha comentado previamente, el código estará basado en el framework RUI [40], que nos facilita la interacción (mediante SPI) del controlador con el transceptor LoRa y ofrece un conjunto de funciones de nivel alto para que el programador se centre en las necesidades de la aplicación.

La lógica del código se encuentra subida a github [41] (diferenciando nodo sensor y nodo sensor/actuador) y para tener una visión global del mismo, se indica su diagrama a continuación [Fig. 48], donde se observan tres funciones principales y las condiciones para que se ejecuten:

- **main**: Es la función principal del código que se ejecuta cuando se alimenta al nodo con al menos 3.3V. Se encarga de inicializar tanto las variables del código como el framework RUI, define los callbacks (funciones que se ejecutan cuando ocurren determinadas acciones) y muestra por consola el estado del nodo, que indica las direcciones asociadas al nodo (dev_eui, app_eui, app_key), si se encuentra enlazado con el gateway (joined), su modo de trabajo (lora / p2p), modo de activación (OTAA / ABP), clase (A / B / C), intervalo de envío (entre envíos el nodo está en ultra bajo consumo), si queremos confirmación (ACK) o la región, entre otros.

  En este punto, el código entra en un bucle infinito **while(1)** que comienza recuperando el estado del nodo, iniciando las tareas necesarias de RUI (**rui_running**) y comporbando la variable de control **sample_status** (por defecto a false), que determinará si existen datos en el buffer para mandar al gateway.

- **app_loop**: Comprueba si el nodo está enlazado al gateway (el servidor debe tener registrado el nodo con su dev_eui y app_key). Si no lo está, se enlaza con la función **rui_join**. Si estaba enlazado, se obtienen los datos a enviar (**get data**) y se activa la variable de control **sample_status.**

- **user_lora_send**: Se envían los datos que se hayan recogido del entorno mediante la función **rui_lora_send** y se desactiva la variable de control de envíos.

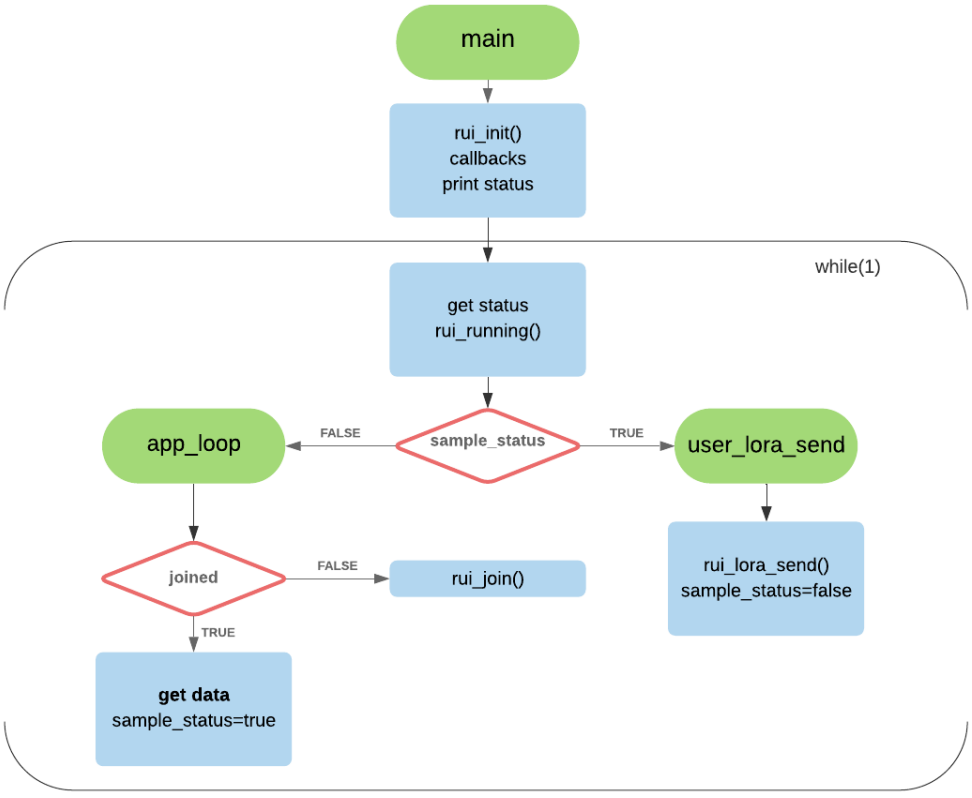


Fig. 48 – Código general del nodo

Los callbacks que se inicializan al principio de la función main son principalmente:

- **LoRaWANJoined_callback:** Se ejecuta cuando el nodo se encuentra en la fase de enlazado con el gateway. Comprueba si se ha enlazado correctamente, repitiendo las peticiones de join tantas veces como tenga establecido en su configuración.
- **LoRaWANSendsucceed_callback:** Se ejecuta cuando se envía un paquete al gateway, imprimiendo por pantalla un mensaje de éxito cuando se realiza de forma correcta.
- **LoRaReceive_callback:** Se ejecuta cuando se recibe el ACK del servidor. Muestra por consola los metadatos del paquete (puerto, RSSI, SNR y tamaño del buffer) y el contenido del mensaje en sí. El nodo sensor/actuador añade además un simulado de clase B con doble envío y el encendido o apagado de la electroválvula según la respuesta del servidor.

Las diferencias de código entre el nodo sensor y el nodo sensor/actuador se encuentran en:

- **get data**: En el siguiente diagrama [Fig. 49], se muestra la lógica combinando lo que ejecuta cada nodo (en azul lo ejecutan ambos y en naranja solo el nodo sensor/actuador).

  Ambos nodos activan la alimentación del sensor de humedad del suelo (pin 4), recuperan el valor de humedad en ese momento (pin 22), calculan el porcentaje según un umbral y desactivan la alimentación del sensor.

  El nodo sensor/actuador, activa el mosfet (pin 2) para establecer temporalmente un divisor de tensión y así poder calcular la tensión en su punto medio (pin 23), desactiva el divisor de tensión y recupera además el estado en el que se encuentra en dicho momento la electroválvula (variable watering).

  Ambos nodos escriben en el buffer los datos que hayan recogido haciendo uso de la codificación Cayenne LPP. El nodo sensor únicamente escribe la humedad (0x68 - Humidity), mientras que el nodo sensor/actuador escribe también el valor de la batería (0x66 – Presence Sensor) y el estado de la electroválvula (0x01 – Digital Output).

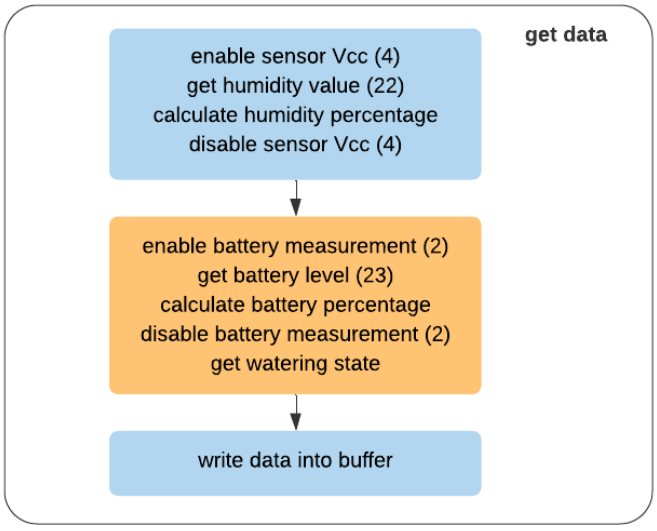


Fig. 49 – función get data

- **LoRaReceive_callback**: La función que se ejecuta al recibir un mensaje del servidor también difiere según el nodo. El nodo sensor únicamente imprime por consola los metadatos y el mensaje en sí. El nodo sensor/actuador, mediante una variable de control (first_send), simula un nodo clase B. Los nodos clase B reciben paquetes bajo demanda usando tramas beacon, pero eso requiere un consumo mayor ya que el nodo se encuentra a la espera de dichas tramas.

Siendo clase A (envía mensaje y se duerme cada cierto tiempo), al entrar por primera vez en el callback, espera 10 segundos en sleep mode, envía un mensaje sin datos (dummy) y desactiva la variable de control first_send. En ese tiempo, el servidor ha procesado el mensaje del nodo y ha tomado la decisión de encendido/apagado del sistema de riego, poniendo el paquete en cola (usando la API de ChirpStack) hasta el siguiente mensaje. Cuando vuelve a recibir el ACK del mensaje dummy, esta vez sí tenemos respuesta del servidor. Activamos la variable de control, imprimimos el mensaje y según el mismo (1 o 0) y el estado en ese momento del sistema (watering true o false), se enciende o se apaga el relé (switch_rele).

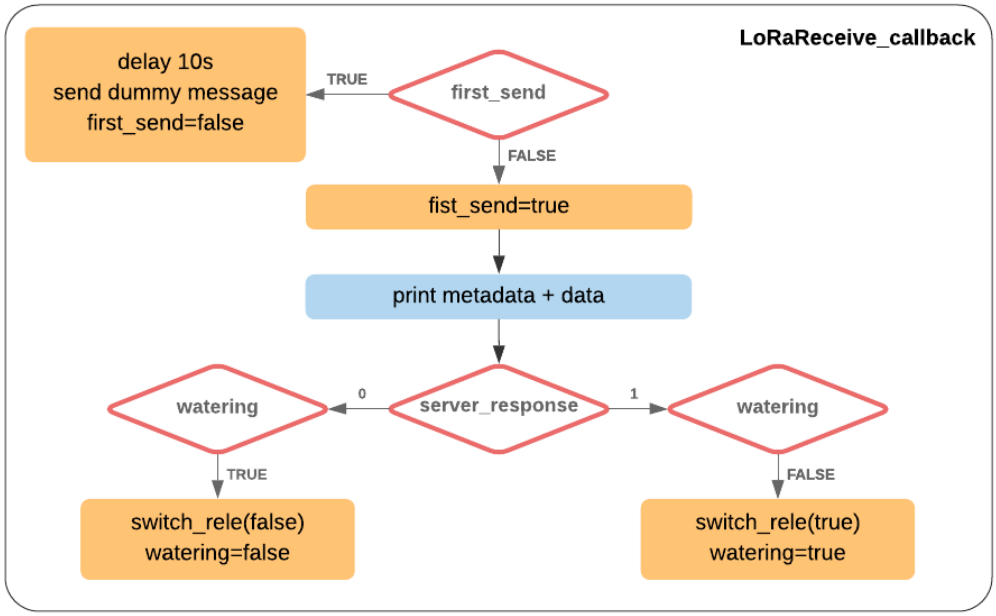


Fig. 50 – Función LoRaReceive_callback

Por último, la función **switch_rele** recibe por parámetros el estado que debe tener la electroválvula y según éste, se enciende un interruptor del puente H o el otro (pines 15 y 16). Para ahorrar en consumo, solo se activa la alimentación del puente (pin 9) cuando se necesita.

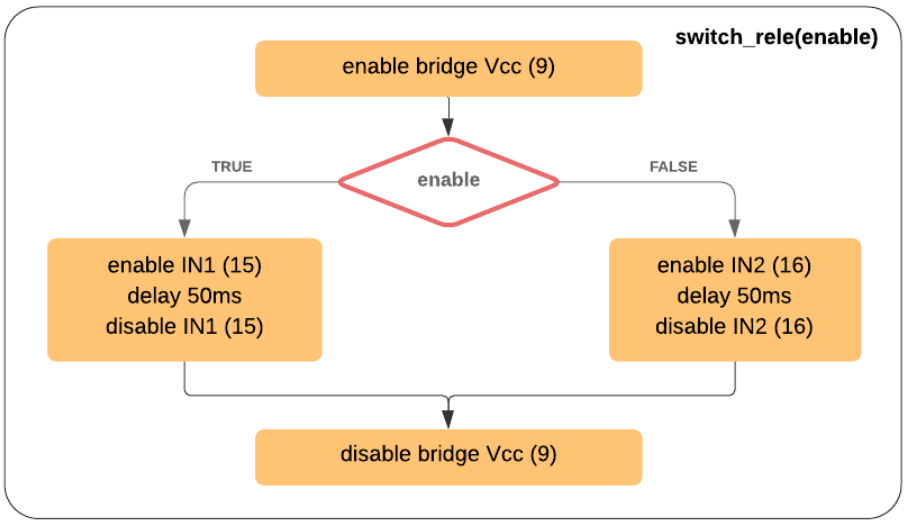


Fig. 51 – Función switch_rele

Para finalizar la sección relativa al código, cabe destacar que al hacer uso del framework RUI, nos estamos olvidando de la comunicación a nivel bajo del controlador con el transceptor LoRa. Internamente, se están utilizando los siguientes comandos AT:

- Get humidity (nodos sensor y sensor/actuador)

```
at+set_config=device:gpio:4:1
at+get_config=device:adc:22
at+set_config=device:gpio:4:0
```

- Get battery (nodo sensor/actuador)

```
at+set_config=device:gpio:2:1
at+get_config=device:adc:23
at+set_config=device:gpio:2:0
```

- Enable relé (nodo sensor/actuador)

```
at+set_config=device:gpio:9:1
at+set_config=device:gpio:15:1
at+set_config=device:gpio:15:0
at+set_config=device:gpio:9:0
```

- Disable relé (nodo sensor/actuador)

```
at+set_config=device:gpio:9:1
at+set_config=device:gpio:16:1
at+set_config=device:gpio:16:0
at+set_config=device:gpio:9:0
```

5.2.2 Integración del código en el nodo

Una vez tenemos el código con la lógica deseada, procedemos a integrarlo en el nodo RAK811. Para ello, lo primero será hacerle BOOT al nodo mediante el software **STM32 Cube Programmer** y con la imagen más actualizada que haya disponible en rakwireless (en este momento **RAK811_BOOT_V3.0.2**) [42].

Dentro del programa, en la sección de la derecha, configuramos la conexión serie con el nodo:

- **Port**: COM3 (el que aparezca por defecto)
- **Baudrate**: 115200 (tasa de fábrica en RAK811, aunque se puede cambiar)
- **Parity**: Even

Para que el nodo entre en modo BOOT, hay que poner su pin físico 9 **BOOT0** a nivel alto (por defecto lo tenemos a tierra). Hecho esto, ya podremos conectar el nodo con el programa y hacemos click en el icono de abajo a la izquierda con forma de goma de borrar para hacerle flash al chip. Estando vacío, hacemos click en Open file, seleccionamos el archivo BOOT descargado previamente y hacemos click en Read para escribir en el chip.

Para obtener el archivo binario que contiene la lógica del nodo, haremos uso del compilador en línea que ofrece el fabricante **RUI Online Compiler** (es necesario solicitar una cuenta de acceso a alguno de los administradores). Entramos con el user/passwd pertinente y seleccionamos RAK811-H (nuestro modelo y frecuencia). Para subir el código, debemos comprimirlos en un archivo zip. Éste archivo contiene:

- **app.c**: Contiene la lógica de la aplicación (los diagramas mostrados en el apartado 5.2.1)
- **at_cmd.c + at_cmd.h:** Contiene los comandos AT de nivel bajo (sobre lo que está montado RUI)
- **lora_config.c + lora_config.h**: Contiene la configuración de los estándares del protocolo LoRa.
- **rui.h:** Funciones del framework RUI.

Una vez tenemos el archivo comprimido en un zip, lo seleccionamos mediante la opción "Select file to upload" y lo subimos con "Upload". Hecho esto, hacemos "Compile" (aparece abajo un log en línea y si todo va bien, nos descarga el binario).

Para insertar dicho binario en el nodo, hacemos uso del programa RAK LoRaButton Upgrade Tool. Nos aseguramos que el pin físico 9 del nodo está a nivel bajo, seleccionamos el puerto serie en el que se encuentra y la tasa de 115200, seleccionamos el binario con "Choose File" y hacemos click en "Start".

Una vez se haya actualizado el nodo, ya tendremos todos los componentes de la aplicación instalados y configurados, listos para ser testeados.

6 PRUEBAS

Si no lo testeo no lo creo

En este capítulo vamos a comprobar todos los aspectos abarcados en los capítulos anteriores. Primero analizaremos la trama en sus enlaces ascendente y descendente para el nodo sensor, comprobando a su vez la calibración del sensor de humedad del suelo. Observaremos las medidas generadas por el envío inalámbrico haciendo uso del protocolo LoRa. Comprobaremos además que el dato llega a cada uno de los servidores. Por otra parte, nos centraremos en el nodo actuador, que añade las funciones de obtener batería y cambair el estado de la electroválvula. Haremos debug en los eventos del flujo de control del servidor y visualizaremos el sistema en un dashboard. Por último, calcularemos el consumo de los nodos con el objetivo de estimar una vida útil para la aplicación.

6.1 Nodo sensor

Con el gateway conectado a una red wifi y a la corriente, nos conectamos mediante ssh desde nuestro ordenador utilizando el software *Putty*. Nos aseguramos de que el concentrador se encuentra activo a la espera de recibir paquetes de los sensores y reenviarlos al servidor.

```
$ cd /opt/ttn-gateway/packet_forwarder/lora_pkt_fwd/
$ ./lora_pkt_fwd
```

Por otro lado, conectamos el RAK811 al ordenador haciendo uso del convertidor FTD1232 como se explica en el apartado 4.2. Al alimentar el nodo, se ejecuta su código interno de manera indefinida, tomando la humedad ofrecida por el sensor.

Introducimos el sensor de humedad en una superficie de tierra seca para observar su comportamiento.

Fig. 52 – Superficie de tierra seca

Haremos uso del software *RAK Serial Port Tool*, conectándonos al puerto serie que haya asignado el ordenador a su entrada USB y ajustando la frecuencia a 115200 baudios (tasa por defecto en el RAK).

El código embebido en el chip del RAK hace uso del framework RUI. Además de las funciones internas del código, también dispone de una API Serial para poder lanzar comandos AT en vivo a través de un puerto serie.

```
$ at+version

    Firmware Version: RUI v3.0.0.12.H.T4
```

A la función interna de get device status se le ha añadido la funcionalidad de ofrecer la última medida de humedad medida por el sensor.

```
$ at+get_config=device:status

    ===============Device Status List================
    Board Core: RAK811
    MCU: STM32L151CB_A
    LoRa chip: SX1276


    Humidity: 7percent
    ==================List End=====================
```

Desde el puerto serie, podemos inicializar cada nodo, estableciendo los códigos EUI y app key (los dos necesarios para la conexión con el servidor ChirpStack), modo de join OTAA (0), clase A (0), región y frecuencia europea (EU868), mensajes con confirmación y tiempo entre lectura de humedad y envío al gateway de 30 segundos (en producción se aumentará este tiempo para alargar la vida útil del sistema):

```
$ at+set_config=lora:dev_eui:AB2F4E9C91A55118
$ at+set_config=lora:app_key:631E77D7083523AF4EF0B3258F51E99D
$ at+set_config=lora:join_mode:0
$ at+set_config=lora:class:0
$ at+set_config=lora:region:EU868
$ at+set_config=lora:confirm:1
$ at+set_config=lora:send_interval:1:30
```

Podemos comprobar los ajustes comentados haciendo uso del siguiente comando:

```
$ at+get_config=lora:status
```

```
==============LoRaWAN Status List================
Work Mode: LoRaWAN
Region: EU868
Send_interval: 30s
Auto send status: true.
Send_interval work at sleep
Join_mode: OTAA
DevEui: AB2F4E9C91A55118
AppEui: 70B3D57ED0013E07
AppKey: 631E77D7083523AF4EF0B3258F51E99D
Class: A
Joined Network:true
IsConfirm: true
AdrEnable: true
EnableRepeaterSupport: false
RX2_CHANNEL_FREQUENCY: 869525000, RX2_CHANNEL_DR:0
RX_WINDOW_DURATION: 3000ms
RECEIVE_DELAY_1: 1000ms
RECEIVE_DELAY_2: 2000ms
JOIN_ACCEPT_DELAY_1: 5000ms
JOIN_ACCEPT_DELAY_2: 6000ms
Current Datarate: 0
Primeval Datarate: 0
ChannelsTxPower: 0
UpLinkCounter: 87
DownLinkCounter: 111
==================List End======================
```

Una vez que tengamos el nodo ajustado de manera correcta, podemos observar cómo se despierta de su estado normal (*stand by*), obtiene la humedad del suelo a través del sensor, imprime por pantalla el valor de tensión medido por el sensor (2.87V), así como su conversión a porcentaje (7%). Envía correctamente el paquete al servidor y recibe el ACK indicando el puerto (0), la relación señal a interferencia RSSI (-48), la relación señal a ruido SNR (8) y la longitud del dato (0 en ACKs). Por último, el nodo vuelve a su estado habitual, *stand by* de mínimo consumo.

```
Wake up

Humidity adc: 2867

Humidity decimal: 7

[LoRa]: send out
[LoRa]: RUI_MCPS_CONFIRMED send success
OK
at+recv=0,-48,8,0
Go to Sleep
```

Si observamos el log del gateway que tenemos capturando paquetes, podemos identificar algunos parámetros LoRa como la frecuencia (867.1MHz), spreading factor SF12, ancho de banda BW 125KHz, coding rate CR 4/5, relación señal a ruido SNR (7.2), relación señal a interferencia RSSI (-36), longitud del dato (18), así como el dato en bruto. También se indican estadísticas de rendimiento del concentrador enfrentando paquetes recibidos correctamente y paquetes erróneos.

```
INFO: Received pkt from mote: 070F6983 (fcnt=24)

JSON up: {"rxpk":[{"tmst":186457812,"chan":3,"rfch":0,"freq":867.100000,"stat":1,"modu":"LORA","dat
r":"SF12BW125","codr":"4/5","lsnr":7.2,"rssi":-36,"size":18,"data":"gINpDweCGAADBggoEWuDMM1X"}]}
INFO: [up] PUSH_ACK received in 44 ms
INFO: [down] PULL_ACK received in 44 ms
INFO: [down] PULL_ACK received in 47 ms

##### 2020-09-03 16:15:57 GMT #####
### [UPSTREAM] ###
# RF packets received by concentrator: 1
# CRC_OK: 100.00%, CRC_FAIL: 0.00%, NO_CRC: 0.00%
# RF packets forwarded: 1 (18 bytes)
# PUSH_DATA datagrams sent: 2 (311 bytes)
# PUSH_DATA acknowledged: 100.00%
### [DOWNSTREAM] ###
# PULL_DATA sent: 3 (100.00% acknowledged)
# PULL_RESP(onse) datagrams received: 0 (0 bytes)
# RF packets sent to concentrator: 0 (0 bytes)
# TX errors: 0
# TX rejected (collision packet): 0.00% (req:2, rej:0)
# TX rejected (collision beacon): 0.00% (req:2, rej:0)
# TX rejected (too late): 0.00% (req:2, rej:0)
# TX rejected (too early): 100.00% (req:2, rej:2)
# BEACON queued: 0
# BEACON sent so far: 0
# BEACON rejected: 0
### [JIT] ###
# SX1301 time (PPS): 189925399
src/jitqueue.c:448:jit_print_queue(): INFO: [jit] queue is empty
### [GPS] ###
# GPS sync is disabled
##### END #####
```

Fig. 53 – Log del concentrador en el gateway

En lo relativo a las medidas de calidad de la señal frente a ruido e interferencias, observamos que los valores se encuentran dentro del rango de dichas medidas que vimos en el apartado 2.4.1.7. Vimos que SNR suele estar comprendido entre -20 y 10dB y tenemos un SNR de 7.2. Por otro lado, el RSSI mínimo es de -120dBm y el máximo teórico 0, obteniendo el nodo un RSSI de -36dBm.

Estos valores tan optimistas se han obtenido debido a la cercanía del gateway con el nodo para realizar la prueba. En el campo se encontrarán a una distancia considerable que empeorará estas medidas.

Si entramos ahora en la interfaz web de ChipStack, en la sección Applications/SmartHumidity/Devices/nodo1, podemos examinar los paquetes recibidos por el servidor en las pestañas LoRaWAN Frames (a nivel de paquete) y Device Data (alto nivel).

Fig. 54 – Captura de paquetes en ChirpStack (LoRaWAN Frames)

Para ver la otra sección debemos esperar a que se envíe otro paquete desde el nodo.

```
JSON up: {"rxpk":[{"tmst":390119524,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","dat
r":"SF12BW125","codr":"4/5","lsnr":8.8,"rssi":-35,"size":18,"data":"gINpDweCHgADBghSFMSvlVvk"}]}
INFO: [up] PUSH_ACK received in 41 ms
```

Fig. 55 – Segundo paquete uplink desde el nodo sensor

La diferencia con la otra sección es que ahora se ha decodificado el dato en bruto usando el estándar de Cayenne LPP. Internamente, el nodo RAK811 ha utilizado el comando at+send=lora:1:016807, en el que indica puerto 1 (1), canal 1 (01), tipo de dato codificado “Humidity Sensor” (68) y el dato en sí (0x07 = 7).

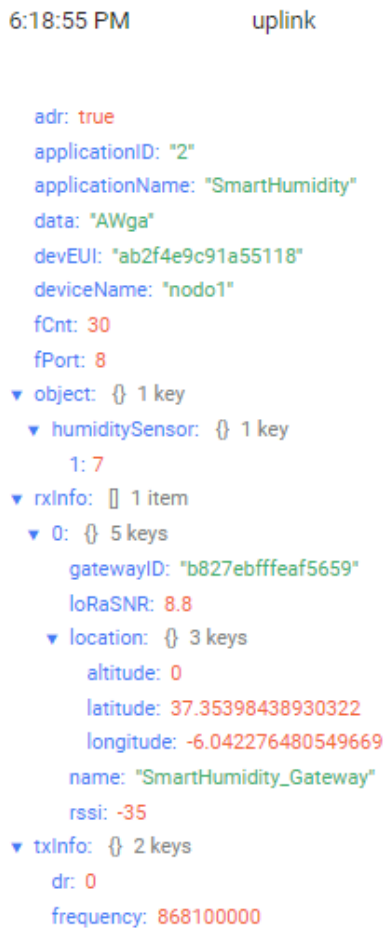


Fig. 56 - Captura de paquetes en ChirpStack (Device Data)

Para cerrar el circuito, comprobamos que el dato ha llegado al servidor de aplicación Thingsboard, entrando en la sección grupos de dispositivos/SmartHumidity/nodo1 y en la pestaña última telemetría.

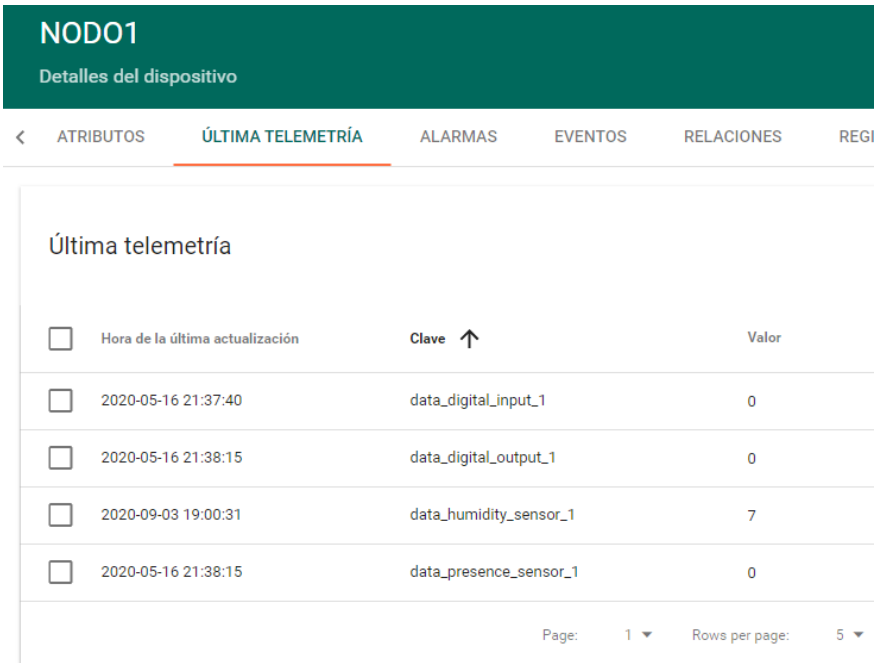


Fig. 57 – Dato de humedad baja en Thingsboard desde el nodo sensor

Si probamos ahora a regar la superficie de tierra, podemos observar cómo el sensor reconoce correctamente el porcentaje de humedad y dicho dato llega al servidor final.

Fig. 58 - Superficie de tierra húmeda

```
Wake up

Humidity adc: 1185

Humidity decimal: 91

[LoRa]: send out
[LoRa]: RUI_MCPS_CONFIRMED send success
OK
at+recv=0,-49,11,0
Go to Sleep
```

NODO1

Detalles del dispositivo

ATRIBUTOS | ÚLTIMA TELEMETRÍA | ALARMAS | EVENTOS | RELACIONES | REGI

Última telemetría

Hora de la última actualización	Clave	Valor
2020-05-16 21:37:40	data_digital_input_1	0
2020-05-16 21:38:15	data_digital_output_1	0
2020-09-03 19:04:35	data_humidity_sensor_1	91
2020-05-16 21:38:15	data_presence_sensor_1	0

Page: 1 Rows per page: 5

Fig. 59 - Dato de humedad alta en Thingsboard desde el nodo sensor

6.2 Nodo sensor/actuador

Para probar el nodo actuador, alimentamos por batería el nodo sensor y conectamos el nodo actuador a nuestro ordenador tal y como se hizo con el nodo sensor. Observamos el log que hemos personalizado en el código:

```
Wake up

Humidity adc: 1562

Humidity percent: 72

Battery adc: 3299

Battery percent: 93

Watering: false

[LoRa]: send out
[LoRa]: RUI_MCPS_CONFIRMED send success
OK
[LoRa]: waiting for downlink msg...
Go to Sleep
OK
Wake up
[LoRa]: RUI_MCPS_CONFIRMED send success
OK
at+recv=1,-49,10,3:d34d34
[LoRa]: disable rele
[LoRa]: already disabled
Go to Sleep
```

En este momento, medimos una humedad del 72%, tenemos la batería de la electroválvula al 93% y el sistema de riego se encuentra apagado. Se manda esta información al servidor, esperamos 10 segundos (en *stand by*) para darle tiempo al servidor a tomar la decisión de encendido o apagado y mandamos un mensaje *dummy* con el objetivo de obtener la respuesta del servidor (encolada en ChirpStack usando la API Rest). Como la humedad del suelo es elevada, el servidor ordena el apagado del sistema de riego, pero como se encontraba ya apagado, no hace nada y entra en *stand by* hasta que pase el periodo de tiempo establecido.

Última telemetría

Hora de la última actualización	Clave ↑	Valor
2020-09-05 18:36:38	data_digital_input_1	0
2020-09-05 18:36:24	data_digital_output_1	0
2020-09-05 18:36:24	data_humidity_sensor_1	72
2020-09-05 18:36:24	data_presence_sensor_1	88

Fig. 60 - Dato de humedad alta en Thingsboard desde el nodo sensor/actuador

En la figura anterior [Fig. 60], observamos cómo Cayenne LPP ha decodificado correctamente la trama, obteniendo los valores pertinentes:

- **data_digital_input_1**: Se utiliza para el mensaje *dummy*, simulando LoRa clase B (beacon).
- **data_digital_output_1**: Indica el estado del sistema de riego (encendido/apagado)
- **data_humidity_sensor_1**: Indica el porcentaje de humedad del suelo
- **data_presence_sensor_1**: Indica el porcentaje de batería de la electroválvula

El servidor Thingsboard, en su sección de cadena de reglas, permite hacer un *debug* a los eventos producidos en cada bloque del flujo.

- **Fetch latest telemetry**: Se recupera el último dato del sensor de base de datos (nodo1)

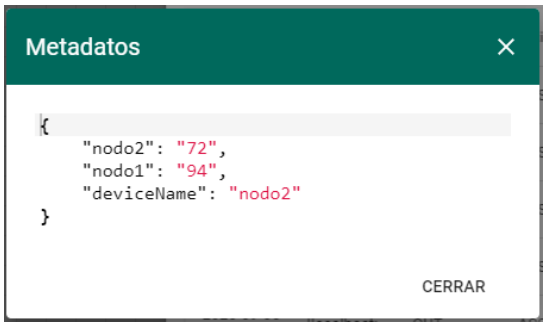


Fig. 61 – Salida del bloque fetch – Humedad elevada

- **Average**: Calcula la media aritmética de ambas medidas

Fig. 62 – Salida del bloque average – Humedad elevada

- **getToken**: Obtiene un token de acceso para poder hacer uso de la API REST de ChirpStack

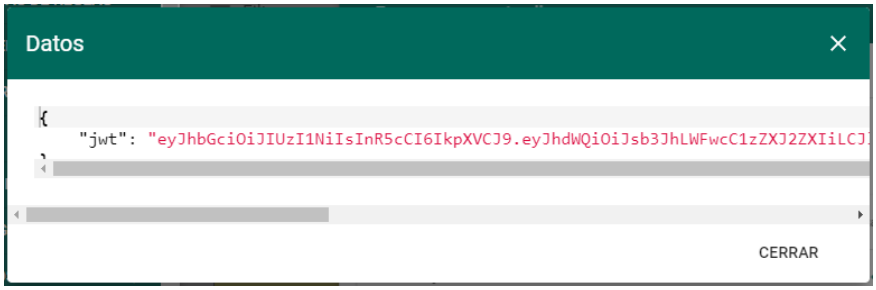


Fig. 63 – Salida del bloque getToken – Humedad elevada

- **Forecast**: Se obtiene la previsión de lluvia actual usando la API de openweather

Fig. 64 – Salida del bloque forecast – Humedad elevada

- **Queue Downlink**: Se trata de la decisión de encendido/apagado del servidor. En este caso, el id de la previsión del tiempo es 800 (no se encuentra dentro del rango de lluvia 500-532) y la humedad de la superficie es 83%. Si tenemos un umbral del 30% la decisión será apagar el riego.

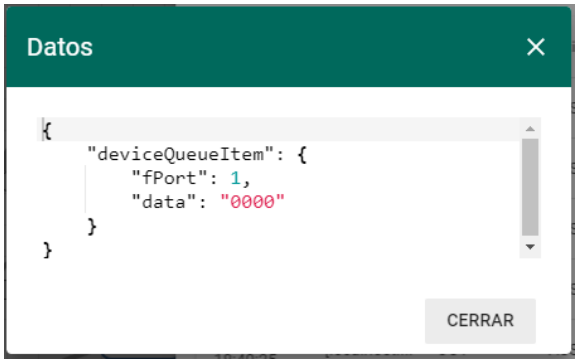


Fig. 65 – Salida del bloque Queue Downlink – Humedad elevada

Thingsboard posee además la opción de crear un dashboard o cuadro de mando para visualizar las medidas del sistema de manera centralizada. De los distintos widgets disponibles, se han elegido:

- **Simple card**: Para indicar la última medida de humedad de cada sensor
- **Vertical bar – justGage**: Para indicar la batería de la electroválvula
- **Watering**: Semáforo que indica si el sistema de riego está encendido (verde) o apagado (rojo)
- **Timeseries table**: Recoge las medidas enviadas por cada sensor en una tabla, humedad en ambos y batería y estado del sistema de riego en el actuador.

- **Timeseries bars**: Recoge las medidas de humedad de los nodos en el tiempo, indicando además los mínimos, máximos y valores promedio.

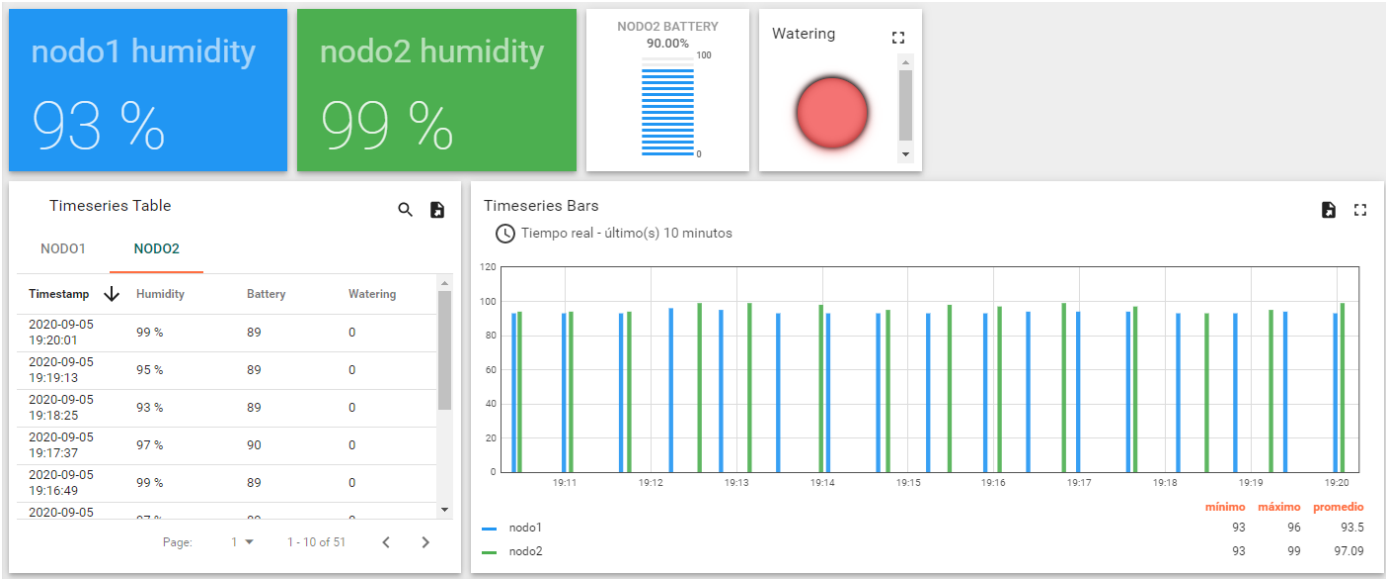


Fig. 66 – Dashboard con humedad elevada

Si probamos ahora con una superficie menos húmeda, lo suficiente como para que esté por debajo del umbral, el sistema se comporta de la manera esperada.

```
Wake up

Humidity adc: 2271

Humidity percent: 37


Battery adc: 3258

Battery percent: 89

Watering: false

[LoRa]: send out
[LoRa]: RUI_MCPS_CONFIRMED send success
OK
[LoRa]: waiting for downlink msg...
Go to Sleep
OK
Wake up
[LoRa]: RUI_MCPS_CONFIRMED send success
OK
at+recv=1,-56,8,3:7df7df
[LoRa]: enable rele
[LoRa]: enabling...
Go to Sleep
```

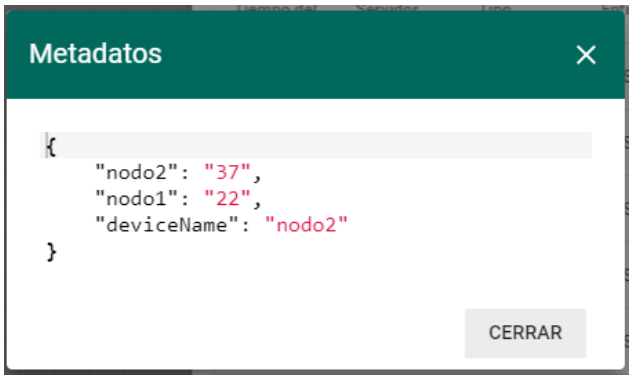


Fig. 67 – Salida del bloque fetch – Humedad baja

Observamos que, aunque la humedad del nodo2 es 37%, la media de ambas medidas es 29.5% (menor que 30%) y, por tanto, esto hará que la decisión del servidor sea de encendido.

Fig. 68 - Salida del bloque Queue Downlink – Humedad baja

Observamos en el dasboard cómo ha decrementado la altura de las barras debido a la bajada de humedad y cómo debido a la decisión del servidor, el sistema de riego se ha encendido (semáforo en verde). Además, en el momento de la orden de encendido, se ha escuchado un “click” en la electroválvula debido al cambio de estado de la membrana de su interior, dejando ahora pasar el agua de un lado hacia el otro.

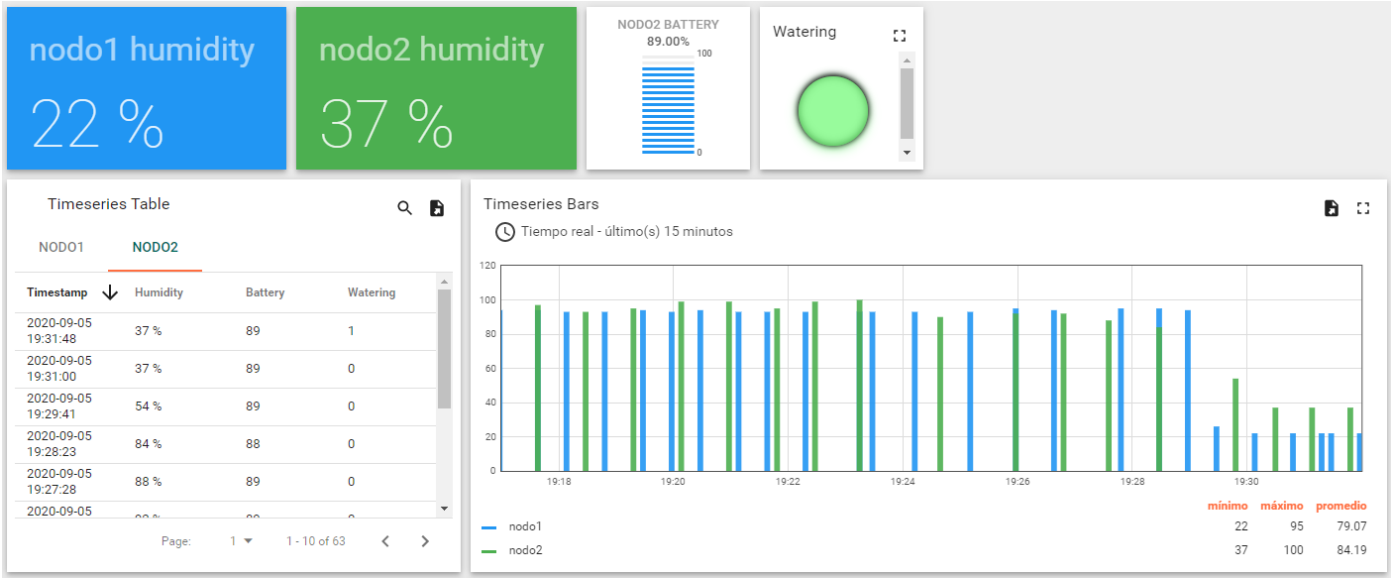


Fig. 69 - Dashboard con humedad baja

6.3 Consumo

A continuación, vamos a calcular el consumo de cada batería y a realizar una estimación teórica de la vida útil de las mismas.

6.3.1 Sensor

Para calcular el consumo en la parte de nodo sensor, vamos a utilizar un multímetro, que colocaremos en serie entre el polo positivo de la batería de una pila y la entrada de 3.3V del módulo RAK811.

RAK811	
Get Humidity	25mA
TX	30mA
RX	5.5mA
Sleep mode	11.5µA

Tabla 20 – Consumo de la batería del nodo

Los 25mA que se consumen al obtener la medida de humedad, se producen a raíz del step-up junto con el regulador de 5V y el sensor en sí. Este consumo lo controlamos con el pin 4, haciendo que no consuma nada hasta que nos haga falta el valor de humedad (tiempo casi despreciable).

El consumo en transmisión y recepción lo podemos obtener del propio datasheet del nodo RAK811, ya que son tiempos muy pequeños incapaces de ser leídos por el multímetro.

El consumo en *sleep mode* o *stand by* si lo podemos medir con el multimetro, obteniendo un valor del rango de microamperios, de vital importancia para que el sistema tenga una buena duración (ya que el nodo se encontrará en su mayor parte del tiempo en dicho modo).

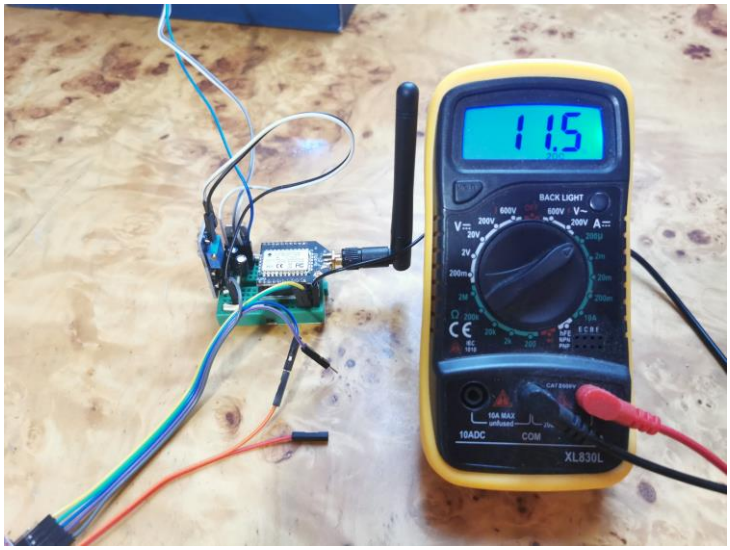


Fig. 70 – Medida del modo stand by con multímetro

Para realizar el cálculo aproximado, vamos a fijar una frecuencia de operación (tiempo que el nodo se encuentra en *stand by*) de 5 minutos. Por otro lado, sabemos que transmite y recibe dos veces en el nodo actuador, por lo que vamos a situarnos en el peor caso (aunque en el nodo sensor solo lo haga una vez). Entre las dos transmisitiones (2s), dos recepciones (2s) y la toma de medida de humedad (1s), vamos a estimar unos 5 segundos, que enfrentándolos a los otros 295s para llegar a la frecuencia de 5 minutos hacen un consumo de:

$$30mA \times 2 + 5.5mA \times 2 + 25mA \times 1 = 19.2mA\ (5s)$$

$$19.2mA \times 5 + 11.5\mu A \times 295 = 331\mu A$$

Tenemos el consumo aproximado por hora, teniendo en cuenta que las pilas que estamos utilizando producen 9800mAh, podemos obtener la duración en años:

$$\frac{9800mAh}{331\mu A} = 29.607h = 1.233d = 41meses = 3.43\ años$$

Para aplicarle mayor precisión al cálculo, vamos a suponer que de los 9800mAh que indica la pila, solo son reales en un 60%, es decir, 5880mAh. Repitiendo el cálculo anterior, obtenemos una duración del nodo de 2 años.

$$\frac{5880mAh}{331\mu A} = 17.764h = 740d = 24meses = 2\ años$$

Hay que tener en cuenta que se está suponiendo que el sistema estará alternando entre encendido y apagado cada 5 minutos, hecho que no será siempre así dependiendo del suelo y las condiciones climatológicas.

6.3.2 Actuador

Para calcular el consumo en la parte de nodo actuador, vamos a colocar el multímetro en serie entre el polo positivo de la batería de dos pilas y la entrada en la protoboard que conecta la batería con el pin Vcc del puente H, así como la entrada del mosfet. Por otro lado, sabemos que la electroválvula tiene una resistencia de 4Ω para los aproximados 8V que ofrece la batería, por lo que tiene un pico de corriente de 2A durante 50ms que dura el pulso que activa o desactiva la electroválvula.

RAK811	
Enable relé	2A
Enable bridge	2.2mA
Stand by	128µA

Tabla 21 - Consumo de la batería de la electroválvula

Para el cálculo, vamos a suponer que una vez activo, consume 2A en los 50ms que dura el impulso, y los 950ms restantes para llegar al segundo supondremos que consume los 2.2mA de activar el puente, ya que también tiene que medir la batería con el divisor resistivo.

$$2A \times 50 + 2.2mA \times 950 = 100mA\ (1s)$$

$$100mA \times 1 + 128\mu A \times 299 = 461\mu A$$

$$\frac{5880mAh}{461\mu A} = 12.754h = 531d = 18meses = 1.5\ años$$

La opción más recomendable es cambiar una vez al año las pilas recargables de todo el sistema, asegurando que las baterías se encuentren en un estado idóneo para su funcionamiento.

7 COMENTARIOS FINALES

En este último capítulo se exponen los objetivos alcanzados y el resultado obtenido, distinguiendo sus características más relevantes. Finalmente, se debaten posibles mejoras futuras que completarían aún más el producto.

7.1 Conclusiones

Se ha diseñado y fabricado un sistema de riego inteligente con tres nodos físicos compuestos por diversos componentes como raspberry, transceptores, sensor, actuador o baterías.

Se les ha dotado de la capacidad de comunicarse entre sí y elevar la computación a la nube, donde se tiene acceso a internet para consultar, por ejemplo, la previsión del tiempo en ese instante.

Se ha conseguido implementar un sistema de IoT para riego con las siguientes características:

- Eco-friendly: Cuida el medio ambiente manteniendo las condiciones idóneas para la plantación a la vez que se ahorra agua ayudando a mitigar la actual crisis de dicho recurso.
- Flexible: Con el trabajo implentado, agrandar la topología del sistema se basa en repetir los pasos de montaje realizados para los nodos sensor y sensor/actuador. De la misma manera con la parte del servidor, que bastaría con configurar los nuevos nodos y añadirlos al flujo.
- Económico: El presupuesto del simulador con los tres componentes del sistema es de aproximadamente 220€, a los que se le añadirían alrededor de 20€ por nodo sensor, 30€ por nodo sensor/actuador y 10€ por cada aspersor. Un sistema completo de estas características en el mercado para abarcar únicamente 100m se encuentra entorno a 1500€ [43].

7.2 Mejoras futuras

Este proyecto sirve como preámbulo para la implantación a nivel de campo de un sistema de riego automático. Si se realizara el paso de implantación, existen diversas mejoras que optimizarían el sistema.

Los nodos sensores y sensores/actuadores del sistema, deberían estar encapsulados en una carcasa impermeable para hacerlos más robustos y duraderos, ya que se encontrarán cerca de los aspersores.

Con vistas a producir la solución a nivel empresarial, sería interesante fabricar un circuito impreso que minimice el tamaño de los nodos, haciendo uso de algún software de diseño como Eagle, imprimiéndolo posteriormente en papel vegetal y pasándolo con un PCB por una insoladora, para después revelarlo y atacarlo con ácido. Finalmente, se perforaría y soldarían los componentes.

Por último, se podría dotar al sistema de inteligencia de negocio, integrando Thingsboard con Trendz Analytics. Esto haría posible la creación de patrones de comportamiento y modelos predictivos en base a los datos capturados por los sensores en el tiempo.

8 REFERENCIAS

[1] «World Water Council - Visión mundial del agua,» [En línea]. Available: https://www.worldwatercouncil.org/fileadmin/wwc/Library/Publications_and_reports/Visions/SpanishExSum.pdf. [Último acceso: Abril 2020].

[2] M. L. i. Seuba, Internet de las cosas. La transformación digital de la sociedad. Cap3. Historia, Ra-Ma Editorial, 2019.

[3] «Xataka - Las 3 tecnologías clave para el Internet de las cosas,» [En línea]. Available: https://www.xataka.com/internet-of-things/las-3-tecnologias-clave-para-el-internet-de-las-cosas. [Último acceso: Abril 2020].

[4] «alfaiot - NB-IoT vs LoRa vs SigFox,» [En línea]. Available: https://alfaiot.com/blog/ultimas-noticias-2/post/nb-iot-vs-lora-vs-sigfox-10. [Último acceso: Abril 2020].

[5] M. M. Arroyo, «El Riego Inteligente en la agricultura,» *iagua,* Septiembre 2017.

[6] «Planting Smart Irrigation Solutions,» *semtech,* 2018.

[7] «LoRa Alliance - A technical overview of LoRa and LoRaWAN,» [En línea]. Available: https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf. [Último acceso: Abril 2020].

[8] L. Alliance, «RP001-1.0.0 LoRaWAN Regional Parameters,» [En línea]. Available: https://lora-alliance.org/sites/default/files/2020-01/rp_2-1.0.0_final_release.pdf. [Último acceso: Abril 2020].

[9] «loratools.nl,» [En línea]. Available: https://www.loratools.nl/#/airtime. [Último acceso: Abril 2020].

[10] L. Alliance, «LoRaWAN 1.0.3 Specification,» 2018. [En línea]. Available: https://lora-alliance.org/sites/default/files/2018-07/lorawan1.0.3.pdf. [Último acceso: Abril 2020].

[11] T. T. Network, «LoRaWAN Network Server,» [En línea]. Available: https://www.thethingsnetwork.org/tech-stack. [Último acceso: Abril 2020].

[12] ChirpStack, «Arquitecture,» [En línea]. Available: https://www.chirpstack.io/overview/architecture/. [Último acceso: Abril 2020].

[13] «Thingsboard,» [En línea]. Available: https://thingsboard.io/. [Último acceso: Abril 2020].

[14] rakwireless, «RAK831 LPWAN Gateway Concentrator,» [En línea]. Available: https://doc.rakwireless.com/datasheet/rakproducts/rak831-lorawan-gateway-datasheet. [Último acceso: Abril 2020].

[15] rs-components, «Raspberry Pi 3 Model B,» [En línea]. Available: https://www.alliedelec.com/m/d/4252b1ecd92888dbb9d8a39b536e7bf2.pdf. [Último acceso: Abril 2020].

[16] rakwireless, «RAK811 LPWAN Breakout Module,» [En línea]. Available: https://doc.rakwireless.com/datasheet/rakproducts/rak811-lora-breakout-module-datasheet. [Último acceso: Abril 2020].

[17] digikey, «Capacitive Soil Moisture Sensor,» [En línea]. Available: https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/SEN0193_Web.pdf. [Último acceso: Abril 2020].

[18] PKCELL, «Polymer Li-ion Battery Technology Specification,» [En línea]. Available: https://datasheetspdf.com/pdf-file/1266160/PKCELL/ICR18650/1. [Último acceso: Abril 2020].

[19] Aerosemi, «MT3608 Datasheet,» [En línea]. Available: https://www.olimex.com/Products/Breadboarding/BB-PWR-3608/resources/MT3608.pdf. [Último acceso: Abril 2020].

[20] stmicroelectronics, «L7805CV - Datasheet,» [En línea]. Available: https://es.farnell.com/stmicroelectronics/l7805cv/reg-tensi-n-5-0v-7805-to-220-3/dp/9756078. [Último acceso: Agosto 2020].

[21] Baccara, «Ficha técnica electroválvula,» [En línea]. Available: https://mundoriego.es/producto/solenoide-g75-o-baccara-9-12v-dc-2-vias-2-hilos-latch-nc/?attachment_id=24582&download_file=5d5e632a39b03. [Último acceso: Abril 2020].

[22] ST, «L298 Dual Full-Bridge Driver,» [En línea]. Available: https://saber.patagoniatec.com/wp-content/uploads/2019/08/l298.pdf. [Último acceso: Abril 2020].

[23] raspberry, «GPIO,» [En línea]. Available: https://www.raspberrypi.org/documentation/usage/gpio/. [Último acceso: Abril 2020].

[24] rakwireless, «RAK831 Gateway Module pinout,» [En línea]. Available: https://doc.rakwireless.com/datasheet/rakproducts/pin-definition-rak831. [Último acceso: Abril 2020].

[25] rakwireless, «RAK811 - Pin Definition,» [En línea]. Available: https://doc.rakwireless.com/datasheet/rakproducts/pin-definition-rak811-lora-breakout-module. [Último acceso: Agosto 2020].

[26] etcher, «balena etcher,» [En línea]. Available: https://etcher.io/. [Último acceso: Abril 2020].

[27] Raspberrypi, «Raspbian OS,» [En línea]. Available: https://www.raspberrypi.org/downloads/raspbian/. [Último acceso: Abril 2020].

[28] robertlie, «github,» [En línea]. Available: https://github.com/robertlie/RAK831-LoRaGateway-RPi. [Último acceso: Julio 2020].

[29] TheThingsNetwork, «github,» [En línea]. Available: https://github.com/TheThingsNetwork/gateway-conf/blob/master/EU-global_conf.json. [Último acceso: Julio 2020].

[30] ChirpStack. [En línea]. Available: https://www.chirpstack.io/. [Último acceso: julio 2020].

[31] Thingsboard. [En línea]. Available: https://thingsboard.io/docs/user-guide/install/rpi/. [Último acceso: Julio 2020].

[32] C. Docs. [En línea]. Available: https://developers.mydevices.com/cayenne/docs/lora/#lora-cayenne-low-power-payload. [Último acceso: Julio 2020].

[33] Thingsboard, «Data function based on telemetry from 2 devices,» [En línea]. Available: https://thingsboard.io/docs/user-guide/rule-engine-2-0/tutorials/function-based-on-telemetry-from-two-devices/. [Último acceso: Agosto 2020].

[34] ChirpStack, «API - Authentication,» [En línea]. Available: https://www.chirpstack.io/application-server/api/#authentication_1. [Último acceso: Agosto 2020].

[35] ChirpStack, «API - Enqueue downlink,» [En línea]. Available: https://www.chirpstack.io/application-server/api/http-examples/. [Último acceso: Agosto 2020].

[36] openweathermap, «API Documentation,» [En línea]. Available: https://openweathermap.org/current. [Último acceso: Agosto 2020].

[37] openweathermap, «Weather Conditions,» [En línea]. Available: https://openweathermap.org/weather-conditions. [Último acceso: Agosto 2020].

[38] «RAKWireless,» [En línea]. Available: https://downloads.rakwireless.com/LoRa/RAK811/Tools/. [Último acceso: Julio 2020].

[39] «RUI Online Compiler,» [En línea]. Available: http://47.112.137.11:12090/#/user/login. [Último acceso: Julio 2020].

[40] rakwireless, «RUI: RAK IoT SDK,» [En línea]. Available: https://downloads.rakwireless.com/RUI/RUI_RAK_LoRaWAN_OpenMCU_Development_Guide-V4.0.pdf. [Último acceso: Julio 2020].

[41] P. Villén, «github,» [En línea]. Available: https://github.com/pablovilmac/SmartHumidity. [Último acceso: Julio 2020].

[42] rakwireless, «downloads.rakwireless.com,» [En línea]. Available: https://downloads.rakwireless.com/LoRa/RAK811/Firmware/Bootloader/. [Último acceso: Julio 2020].

[43] habitissimo, «Instalar sistemas de riego: Precio y Presupuestos,» [En línea]. Available: https://www.habitissimo.es/presupuestos/instalar-sistemas-de-riego#:~:text=De%20manera%20orientativa%2C%20podemos%20contar,%3A%20100%20%E2%82%AC%20%2D%20200%20%E2%82%AC.. [Último acceso: Septiembre 2020].

GLOSARIO

WWW	Worl Wide Web
IoT	Internet of Things
TIC	Tecnologías de Información y Comunicaciones
ONUAA	Organización de las Naciones Unidas para la Alimentación y la Agricultura
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
RUI	RAK Unified Interface
API	Application Programming Interfaces
REST	Representational State Transfer
SSH	Secure Shell
MIT	Massachusetts Institute of Technology
RFID	Internet Business Solutions Group
IBSG	Acorn RISC Machine
ARM	Low Power Wide Area Network
NB-IOT	Narrowband IoT
3GPP	3rd Generation Partnership Project
OFDM	Orthogonal Frequency Division Multiplexing
OSI	Open System Interconnection
CSS	Chirp Spread Spectrum
BW	Band Width
SF	Spreading Factor
FSK	Frequency Shift Keyin
BPSK	Binary phase-shift keying
QPSK	Quadrature phase-shift keying
CR	Code Rate
ADR	Adaptative Data Rate
TXCO	Temperature Compensated Crystal Oscillator
FEC	Forward Error Correction
ISM	Industrial, Scientific and Medical
CRC	Cyclic Redundancy Check
ToA	Time on Air
RSSI	Received Signal Strength Indication
SNR	Signal-to-Noise Ratio
MAC	Media Access Control
MIC	Message Integrity Check
ACK	Acknowledgement
OTAA	Over-The-Air Activation
ABP	Activation By Personalization
AES	Advanced Encryption Standard
TTN	The Things Network
gRPC	Remote Procedure Calls
HTTP	Hypertext Transfer Protocol
MQTT	Message Queing Telemetry Transport
JSON	Javascript Object Notation
TDoA	Time Difference of Arrival
CoAP	Constrained Application Protocol
SQL	Structured Query Language
UDP	User Datagram Protocol
FIFO	First In First Out
SPI	Serial Peripheral Interface
RPi	Raspberry Pi
WLAN	Wireless Local Area Network
USB	Universal Serial Bus
SoC	System on Chip
GPIO	General Purpose Input/Output
CPU	Central Processing Unit
RAM	Random Access Memory
TX	Transmisor
RX	Receptor

MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
MOSI	Master Out Slave In
MISO	Master In Slave Out
SS	Slave Select
SFTP	Secure File Transfer Protocol
JWT	Java Web Token
GPS	Global Positioning System
UART	Universal Asynchronous Receiver-Transmitter