UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Monitorización de Parámetros Ambientales del Patrimonio Cultural Mediante Dispositivos IoT y Conexión por Satélite

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Gotor Ramos, Miguel

Tutor/a: Perles Ivars, Ángel Francisco

Cotutor/a externo: LABORDA MACARIO, JAIME

CURSO ACADÉMICO: 2021/2022

Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politécnica de València

Monitorización de parámetros ambientales del patrimonio cultural mediante dispositivos IoT y conexión por satélite

Trabajo de Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

Curso Académico: 2021/2022

Valencia, Julio de 2022

Documentos:

I. Memoria

II. Planos

III. Pliego de condiciones

IV. Presupuesto

V. Anexos

V.1. Código

V.2. Hojas de características

Autor: D. Miguel Gotor Ramos

Tutores: D. Ángel Francisco Perles Ivars

D. Jaime Laborda Macario

Gracias a mis tutores Ángel Perles Ivars y Jaime Laborda Macario por su gran ayuda y cercanía durante el desarrollo de este proyecto.

A mis padres, por su apoyo incondicional.

Resumen

El presente proyecto estudia la idea de conectar a internet aquellos bienes inmuebles históricos de interés cultural con especial problemática por su entorno, como son las pinturas rupestres. Por ello, se presenta la creación de un sistema de monitorización de parámetros ambientales basado en dispositivos de ultrabajo consumo y tecnologías IoT, en zonas del medio natural carentes de conexión a internet. Tampoco se prevé la existencia de una preinstalación eléctrica que permita alimentarlo desde una red pública, puesto que, a menudo, el bien cultural a proteger se encuentra en un medio técnicamente de difícil acceso y por lo general, poco viable para una instalación exclusiva.

Concretamente, se ha realizado el despliegue de una red LoRaWAN basada en sensores digitales inalámbricos, cuya pasarela se enlaza mediante una conexión puente MQTT a un kit de conexión por satélite, que integra todos los dispositivos necesarios para establecer la comunicación y finalmente conectarse a internet, habilitando el acceso privado a las medidas de los sensores. Además, se ha desarrollado un panel de visualización que permite a los expertos conservadores tomar decisiones en tiempo real, y efectuar las intervenciones necesarias.

Cada uno de los dispositivos integrados en el sistema posee un papel específico y complementario al resto, pero todos se han seleccionado con el fin de dotar al mencionado sistema de la máxima autonomía, eficiencia energética y protección frente a los agentes atmosféricos como el viento, los cambios de temperatura y las precipitaciones.

Palabras clave: *Conservación preventiva, IoT, LoRaWAN, Patrimonio cultural, Satélites.*

Abstract

This project studies the idea of connecting to the internet those historic immovable assets of cultural interest with special problems due to their environment, such as cave paintings. Therefore, it presents the creation of a system for monitoring environmental parameters based on ultra-low power devices and IoT technologies, in areas of the natural environment lacking internet connection. Nor is the existence of an electrical pre-installation that would allow it to be powered from a public network foreseen, given that the cultural asset to be protected is often located in a technically difficult to access environment and, in general, not feasible for an exclusive installation.

Specifically, a LoRaWAN network based on wireless digital sensors has been deployed, whose gateway is linked via an MQTT bridge connection to a satellite connection kit, which integrates all the necessary devices to establish communication and finally connect to the internet, enabling private access to the sensors measurements. In addition, a visualisation panel has been developed to allow the curatorial experts to make decisions in real time, and to carry out the necessary interventions.

Each of the devices integrated in the system has a specific and complementary role to the others, but all have been selected to provide the system with maximum autonomy, energy efficiency and protection against atmospheric agents such as wind, temperature changes and precipitation.

Keywords: *Preventive conservation, IoT, LoRaWAN, Cultural Hertitage, Satellites.*

Resum

El present projecte estudia la idea de connectar a internet aquells béns immobles històrics d'interés cultural amb especial problemàtica pel seu entorn, com són les pintures rupestres. Per això, es presenta la creació d'un sistema de monitoratge de paràmetres ambientals basat en dispositius d'ultra baix consum i tecnologies IoT, en zones del medi natural mancats de connexió a internet. Tampoc es preveu l'existència d'una preinstal·lació elèctrica que permeta alimentar-ho des d'una xarxa pública, ja que, sovint, el bé cultural a protegir es troba en un mitjà tècnicament de difícil accés i en general, poc viable per a una instal·lació exclusiva.

Concretament, s'ha realitzat el desplegament d'una xarxa LoRaWAN basada en sensors digitals sense fils, la passarel·la dels quals s'enllaça mitjançant una connexió pont MQTT al kit de conexió per satèl·lit, que integra tots els dispositius necessaris per a establir la comunicació i finalment connectar-se a internet, habilitant l'accés privat a les mesures dels sensors. A més, s'ha desenvolupat un panell de visualització que permet als experts conservadors prendre decisions a temps real, i efectuar les intervencions necessàries.

Cadascun dels dispositius integrats en el sistema posseeix un paper específic i complementari a la resta, però tots s'han seleccionat amb la finalitat de dotar a l'esmentat sistema de la màxima autonomia, eficiència energètica i protecció enfront dels agents atmosfèrics com el vent, els canvis de temperatura i les precipitacions.

Paraules clau: *Conservació preventiva, IoT, LoRaWAN, Patrimoni cultural, Satèl·lits.*

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politécnica de València

Monitorización de parámetros ambientales del patrimonio cultural mediante dispositivos IoT y conexión por satélite

I. MEMORIA

Autor: D. Miguel Gotor Ramos

Tutores: D. Ángel Francisco Perles Ivars

D. Jaime Laborda Macario

Índice general

Índice de Figuras

Índice de tablas

Glosario de siglas y abreviaturas

Abreviatura	Significado	Pág.
3GPP	Third Generation Partnership Project	28
ABP	Authentication By Personalization	51
ACK	Acknowledgement	68
ADC	Analog-to-Digital Converter	10
ADR	Adaptative Data Rate	50
AES	Advanced Encryption Standard	32
AP	Access Point	48
API	Application Programming Interface	44
ARTES	Advanced Research in Telecommunications Systems	37
ACII	American Standard Code for Information Exchange	60
ASHRAE	American Society of Heating, Air-conditioning and Refrigerating Engineers	68
BTS	Base Transceiver Station	27
CC	CollectionCare	45
CMOS	Complementary Metal-Oxide Semiconductor	41
CSS	Chirp Spreach Spectrum	32
DBPSK	Differential Binary Phase Shift Keying	35
DC	Direct Current	9
ESSID	Extended Service Set Identifier	48
ETSI	European Telecommunications Standards Institute	26
EUI	Extended Unique Identifier	55
FCC	Federal Communications Commission	22
FHSS	Frequency Hopping Spread Spectrum	22
GFSK	Gaussian Frequency Shift Keying	35
GPIO	General Purpose Input/Output	41
GPRS	General Packet Radio Service	27
GPS	Global Positioning System	28
GSM	Global System for Mobile Communications	26
HR	Humedad Relativa	3
HTTP	HyperText Transfer Protocol	55
I2C	Inter-Integrated Circuit	13
IANA	Internet Assigned Numbers Authority	54

IoT	Internet Of Things	2
IP	Internet Protocol	29
IPxx	Ingress Protection	14
IR	Radiación Infrarroja	4
ISL	Intersatellite Link	39
ISM	Industrial Scientific & Medical	17
ISS	Integrated Sensor Suite	22
ITU	International Telecommunication Union	26
KA	Keep Alive	55
L	Luz	3
LEO	Low Earth Orbit	36
LNS	LoRaWAN Network Server	47
LPWAN	Low-Power Wide-Area Network	7
LTD	Limited	19
LTE	Long Term Evolution	29
LWT	Last Will and Testament	55
M2M	Machine-to-machine	44
MAC	Media Access Control	48
MCU	Microcontroller Unit	15
MQTT	Message Queuing Telemetry Transport	47
NB-IoT	Narrowband IoT	35
NMEA	National Marine Electronic Association	60
NMEA	National Marine Electronic Association	60
ODM	Original Design Manufacturer	47
OEM	Original Equipment Manufacturer	47
OFDM	Orthogonal Frequency Division Multiplexing	29
OLED	Organic Light-Emitting Diode	42
OSI	Open Systems Interconnection	31
OTAA	Over The Air Activation	51
OTP	One Time Programmable	12
PCB	Printed Circuit Board	12
PTFE	Politetrafluoroetileno	12
QoS	Quality Of Service	55
RF	Radio Frecuencia	17
RSSI	Received Signal Strength Indicator	63
Rx	Receive	55

SF	Spreading Factor	32
SIM	Suscriber Identity Module	36
STA	Station (Mode)	58
T	Temperatura	3
TCP	Transmission Control Protocol	27
Tx	Transmit	55
U	Unit	38
UART	Universal Asynchronous Receiver-Transmitter	41
UMTS	Universal Mobile Telecommunications Service	28
UNB	Ultra-Narrow Band	35
URL	Uniform Resource Locator	68
USD	United States Dollar	40
UTF	Unicode Transformation Format	52
UV	Radiación Ultravioleta	4
VHF	Very High Frequency	42
WCDMA	Wideband Code Division Multiple Access	29
WLAN	Wide Local Area Network	17

Lista de símbolos

b	—	Bit
B	—	Byte
dBm	—	Decibelio-milivatio ($10log_{10}\frac{P}{1\ mW}$)
h	—	Hora
Hz	—	Hercio
Lx	—	Lux – Unidad de iluminancia ($^{lm}/_{m^2}$)
m	—	Metro
Pa	—	Pascal
s	—	Segundo
W	—	Vatio
º	—	Grado sexagesimal
ºC	—	Grado Celsius

1. Objeto del proyecto

El objetivo del presente proyecto consiste en la realización de un sistema que permita monitorizar los parámetros ambientales en lugares cuyo valor patrimonial justifiquen la instalación del mencionado sistema, y estén localizados en zonas rurales que no dispongan de una conexión de calidad, sin acceso a internet, o incluso sin cobertura telefónica.

Se pretende medir las distintas variables ambientales en el interior y aledaños de abrigos rocosos, (temperatura, humedad, presión, luminancia, presencia de agentes contaminantes…) mediante una serie de sensores, y compartir las lecturas recogidas mediante una conexión satelital a internet de baja velocidad.

Ello permitirá a los técnicos mantener una vigilancia constante sobre los mencionados parámetros que afectan a las pinturas ubicadas en el abrigo y efectuar las intervenciones necesarias para que se mantengan en óptimo estado de conservación, todo ello en tiempo real.

2. Antecedentes

El presente proyecto tiene su origen en la necesidad del cuidado preventivo del patrimonio histórico y cultural, dotando al personal competente de la información necesaria para su intervención antes de que los parámetros medioambientales puedan dañar un bien, evitando los desperfectos y la intervención "a posteriori", cuando el daño ya ha sido verificado.

La noción básica de prevenir el deterioro de los bienes culturales ha existido desde siempre, pudiendo encontrar ejemplos de ello desde el comienzo de la historia del arte: La utilización de cortinas para cubrir las pinturas, protegiéndolas de la luz y del polvo, es un ejemplo simple de tratar de minimizar el daño debido a agentes del medio al que se expone el bien cultural.

Por otra parte, el concepto de **conservación preventiva** es más reciente; se trata de una disciplina que data de finales del siglo pasado: El término “Conservación Preventiva” se utilizó por primera vez a principios de la década de los 70, aunque su uso comenzó a popularizarse a finales de siglo, entre los años 80 y los 90, conforme iba adquiriendo reconocimiento en el ámbito científico y cultural, al demostrar suponer una mejor alternativa a otros métodos de conservación más tradicionales como la “Conservación Curativa” o la restauración. Desde entonces, la progresiva

integración de la ciencia en el mundo de los museos, así como el fortalecimiento de la colaboración entre restauradores y científicos no ha hecho sino aumentar. Como resultado, surge su implementación a través de las nuevas tecnologías como es el caso de la **tecnología IoT** (*Internet Of Things*), o Internet de las Cosas, un campo de estudio en auge desde la última década y que está experimentando un crecimiento exponencial, basado en la integración de sensores, capacidad de procesamiento, y sistemas embebidos de toda índole en objetos físicos, y en su interconexión a través de internet u otras formas de comunicación con el fin de permitir una difusión de gran alcance de cualquiera que sea la información captada.

El empleo de dispositivos IoT permite extender las posibilidades de la conservación preventiva, a artefactos culturales en entornos de interior (museos, almacenes) o exterior, típico de bienes inmuebles, como es el caso que nos ocupa.

3. Estudio de necesidades, factores a considerar: limitaciones y condicionantes

Con el fin de garantizar la consecución de los objetivos expresados en el apartado anterior, es necesario determinar las necesidades del proyecto, establecidas por la experiencia del personal especializado en la temática, así como por recopilaciones procedentes de otros proyectos y estudios que se han realizado al respecto.

3.1. Factores a considerar

A modo de recordatorio, señalar que el presente proyecto tiene por finalidad la realización de un sistema que permita efectuar una monitorización de los distintos parámetros ambientales que puedan contribuir al deterioro de bienes culturales.

Concretamente, se pretende llevar a cabo un seguimiento de las pinturas rupestres ubicadas en el interior de abrigos rocosos. Tratándose de refugios naturales de poca profundidad, las manifestaciones de arte rupestre ubicadas en su interior son fácilmente accesibles a la pequeña fauna local (actualmente están protegidas por cerramientos metálicos de tipo "jaula"), quedando expuestas a factores extrínsecos de deterioro de toda índole:

- De origen **antrópico o antropogénico**: La manipulación de cualquier tipo, ya sea mero contacto, roces, impactos… Cualquier interacción física con sustrato sobre el que descansa la pintura comporta un serio perjuicio sobre la misma.

- **Biológicos**: Si bien son más susceptibles los bienes culturales en soporte de origen vegetal como papel, madera o lienzo, las pinturas no están a salvo del ataque de insectos o microorganismos (formación de colonias bacterianas, hongos, y otras comunidades microbianas)
- E incluso **catastróficos**: Inundaciones, incendios… Lógicamente, las pinturas se encuentran en un medio natural, donde por ejemplo el terreno agreste presenta gran cantidad de combustibles vegetales, susceptibles de propagar rápidamente el fuego en caso de incendio forestal.

Y, finalmente, aquellos en los que se va a centrar el sistema cuyo diseño se detallará en los subsiguientes apartados:

- **Ambientales**: Luz natural (L), humedad relativa (HR), temperatura (T), contaminantes atmosféricos, vibraciones, el viento… La combinación de los efectos del clima genera un desgaste constante en la roca; por ello, es preciso evitar variaciones bruscas o excesivas de estas magnitudes, que alteran las condiciones óptimas del entorno inmediato del abrigo.

Estas últimas están directamente relacionadas con circunstancias exteriores, imprevisibles o inevitables como el ciclo diurno y nocturno, o las precipitaciones y demás condiciones meteorológicas. En definitiva, no es sencillo intervenir en la evolución natural de dichos parámetros. Tanto es así, que la erosión producida por las condiciones ambientales constituye una de las más difíciles de controlar, especialmente tratándose de un bien cultural situado a la intemperie, por lo que presenta una problemática de gran interés, foco de este proyecto.

Sin embargo, mediante un conocimiento detallado y preciso de las magnitudes, así como de su progreso a lo largo de determinados periodos de tiempo, se logra minimizar el impacto que puedan tener sobre el bien cultural mediante la aplicación de las medidas correctivas correspondientes.

Por ello, conviene profundizar en las distintas magnitudes de interés.

3.2. Magnitudes de interés

3.2.1. Temperatura

La temperatura es un catalizador de reacciones químicas; en rangos ambientales, su aumento propicia el crecimiento microbiano, y valores extremos producen todo tipo de efectos adversos como fisuras y agrietamientos sobre la roca; en los pigmentos

son comunes la desecación, decoloración y oscurecimiento del tono, siendo este último uno de los efectos más observados a altas temperaturas.

3.2.2. Humedad Relativa

Al igual que la temperatura, favorece la proliferación de microorganismos.

Ambas magnitudes, combinadas con la presión del aire, determinan **el punto de rocío**. A partir de dicho punto, se produce la condensación del vapor de agua disuelto en el aire, que potencialmente contiene bacterias, sobre las pinturas, provocando cristalizaciones y descamaciones e incluso, **lixiviación**, un proceso químico que sucede en las rocas, afectando a su composición al reducir la concentración de minerales. Nuevamente, ocasiona descamaciones y crecimiento biológico.

3.2.3. Luz visible, radiación ultravioleta y radiación Infrarroja

Algunas de las pinturas soportan una exposición continuada a la luz natural, que contiene el espectro total de radiación electromagnética proveniente del sol. Es decir, tanto la luz visible como la radiación infrarroja (IR) y ultravioleta (UV).

La interacción entre cualquier tipo de luz y las moléculas de los materiales orgánicos produce un deterioro fotoquímico capaz de afectar a la tonalidad y a la composición físico-química de dichos materiales. No obstante, se recomienda dividirla en estas tres categorías ya que la cantidad de energía electromagnética es inversamente proporcional a la longitud de onda, lo que significa que la radiación UV causa un daño mayor que la luz visible, siendo el principal desencadenante de las reacciones químicas en colorantes, pigmentos y tintes. Por otra parte, la radiación IR o radiación térmica se transforma en calor al incidir sobre los objetos, por lo que sus efectos se ven reflejados en la medición de T.

Finalmente, se ha comprobado que la luz, en las condiciones adecuadas, contribuye a la mencionada lixiviación. Por todo ello, se deberá obtener los datos del nivel de incidencia lumínica y radiación UV sobre las pinturas para su adecuada conservación.

3.3. Requisitos de las mediciones

Cada una de estas variables posee una cierta capacidad erosiva, pero como se ha visto, determinadas combinaciones de las tres dan lugar a procesos específicos, que dañan seriamente las manifestaciones de arte rupestre; por consiguiente, para

anticiparse a estas situaciones perjudiciales, es necesario disponer de la información de sus valores, **de forma simultánea**, y **en tiempo real**.

3.3.1. Frecuencia de las mediciones. Requerimientos de los sensores

Al tratarse de variables ambientales, lo esperable es que su evolución sea relativamente lenta y esté acotada entre unos límites contenidos en el rango de funcionamiento de la mayoría de sensores comercialmente disponibles, cuyas características se estudiarán más adelante. Es además probable que la calibración de fábrica sea suficiente para los niveles de exactitud requeridos en la zona de trabajo del sensor.

Existen varios estándares en lo referente a la conservación de patrimonio cultural, y la monitorización de variables ambientales con dicho fin. En España, destacan:

- UNE-EN 15757:2011 – Conservación del patrimonio cultural. Especificaciones de temperatura y humedad relativa para limitar los daños mecánicos causados por el clima a los materiales orgánicos higroscópicos.
- UNE-EN 15758:2011 – Conservación del patrimonio cultural. Procedimientos e instrumentos para la medición de las temperaturas del aire y de las superficies de los objetos
- UNE-EN 16242:2014 – Conservación del patrimonio cultural. Procedimientos e instrumentos para la medición de la humedad del aire y los intercambios de humedad entre el aire y el patrimonio cultural.

Idénticas a sus equivalentes europeas. Tratan acerca de los valores de T y HR, pues son los principales parámetros que afectan a los artefactos culturales. Por otra parte, las Normas y directrices de la ASHRAE (*American Society of Heating, Air-Conditioning and Refrigerating Engineers*) (EEUU) y la UNI 10829:1999 – Bienes de interés histórico-artístico. Condiciones ambientales de conservación. Medición y análisis. (Italia) aportan recomendaciones adicionales, pudiendo servir para complementar o contrastar la información.

De acuerdo con la UNE-EN 15758, el sensor de T deberá poseer una exactitud mínima de ± 0,5 ºC, en el rango de –20 ºC a 60 ºC.

Por su parte, la UNE-EN 16242 establece varios niveles de precisión para las medidas de HR. Optando por el nivel '2' (nivel alto), la exactitud será del ±2 %, en el rango de 5 % a 95 %.

En la Tabla 1 quedan recogidas el resto de características estáticas que el sensor deberá reunir:

Parámetro	**Temperatura**	**Humedad Relativa**
Unidades	Grados Celsius (ºC)	Porcentaje (%)
Norma Europea	EN 15758	EN 16242
Rango de operación	-20 ºC a 60 ºC	5 – 95 %
Incertidumbre	0,5 ºC	2 %
Repetibilidad	0,1 ºC	1 %
Resolución	0,1 ºC	1 %
Periodo de muestreo	1 h	1 h

Tabla 1: Características estáticas de los sensores de temperatura y humedad

3.4. Requisitos del sistema de sensores

La necesidad principal consiste en tomar medidas sobre la superficie rocosa del abrigo, pero resulta conveniente medir asimismo otras variables climáticas en las proximidades de la cueva, por lo que será necesario emplear dispositivos IoT de distinta clase, cuya selección se especificará en el apartado 4 de la memoria.

El sistema de sensores queda dividido en dos partes diferenciadas:

- Para las medidas in situ, se requieren sensores comerciales individuales.
- Para las medidas de en los aledaños del abrigo, se precisa de otro tipo de instrumento de medida, capaz de captar información diferente, como la velocidad del viento y las precipitaciones. Se necesitará una estación climatológica.

3.4.1. Consumo energético. Autonomía

Como resulta habitual en este tipo de aplicaciones dentro del paradigma IoT, el sistema se hallará en un ambiente campestre sin acceso inmediato a la red eléctrica, por lo que las baterías constituirán la principal fuente de alimentación.

Dado que una de las necesidades básicas del proyecto reside en maximizar su vida útil sin necesidad de efectuar ningún tipo de mantenimiento, con el fin de que los expertos centren sus esfuerzos en la conservación del artefacto cultural, estas baterías

deberán durar grandes periodos de tiempo, incluso años. Por tanto, se deberá diseñar un sistema con un gasto energético ínfimo, mediante la adecuada selección de componentes de ultrabajo consumo.

3.4.2. Comunicación inalámbrica. Cobertura

- Conexión entre los nodos finales y el *gateway*

Es necesario alcanzar un compromiso entre velocidad, consumo y alcance de la red de comunicación. Teniendo en cuenta que se desea cubrir una gran distancia, y dados los requisitos de consumo explicados en el apartado anterior, se optará por una red de tipo LPWAN (*Low-Power, Wide-Area Network*).

El resto de especificaciones técnicas (Banda de frecuencia, velocidad de transferencia de datos, tamaño de paquete, número de paquetes diarios…) quedarán determinadas por el protocolo de comunicación, que, siguiendo la misma filosofía, deberá permitir crear nuestra propia red y conectar el dispositivo a internet con el menor coste posible.

- Conexión del sistema en el enclave a la nube

De nuevo, al tratarse de una zona rural, no todos los sistemas de comunicación ofrecen cobertura para conectar los dispositivos a la red. Se investigarán las distintas alternativas de módulos satélite y comunicaciones GSM convencionales.

Además, ambos niveles de comunicación deberán presentar algún mecanismo de seguridad para hacer el contenido legible solo al destinatario. Al fin y al cabo, los dispositivos se van a conectar a través de una pasarela a otras tecnologías, y conectar un dispositivo no seguro a una red segura implica introducir un eslabón débil.

4. Planteamiento de soluciones alternativas y justificación de la solución adoptada

Una vez identificadas las necesidades del proyecto, y con ello, los obstáculos que se nos plantean, es necesario emprender un estudio de alternativas.

Ello permitirá identificar los medios posibles para alcanzar los objetivos principales, y empleando métodos cuantitativos y cualitativos para determinar la adecuación de las diferentes alternativas, seleccionar aquellas que resulten más apropiadas desde los puntos de vista técnico y económico.

A continuación se realiza una descripción detallada de los distintos enfoques y soluciones que se han considerado para cada uno de los apartados del proyecto. Primeramente se estudiarán los elementos físicos del sistema, empezando por los sensores.

4.1. Elección de sensores

Se abordará la elección de sensores desde el punto de vista de la instrumentación electrónica; la rama científico-técnica de las medidas. Así pues, se repasarán brevemente los conceptos de señal, sistema de medida (bucle abierto y cerrado, dominio analógico y digital), y sus características técnicas principales (estáticas y dinámicas)

4.1.1. Diferencias entre sensor y transductor

A menudo ambos términos se aceptan como sinónimos; es común utilizarlos indistintamente, pero para el diseño del sistema de medida conviene puntualizar las diferencias entre estos dos tipos de dispositivos, fundamentales en el ámbito de la ingeniería electrónica.

Un **sensor** es todo objeto con una propiedad sensible al medio. Es decir, alguna de sus características físicas varía cuando lo hace la magnitud física (mecánica, térmica, magnética, eléctrica…). De esta forma, este dispositivo es capaz de convertir una señal física de entrada en otra de salida, que puede ser eléctrica o no. Por ello, en un sistema de medida, se trata del elemento dispuesto expresamente para obtener la información del medio, y se encuentra en contacto directo con la magnitud a medir.

Por otra parte, un **transductor** es un dispositivo generalmente más complejo, cuya finalidad es convertir o transformar la señal original en otra de naturaleza eléctrica (corriente o tensión), lo que la hace más útil y permite su transmisión al resto del sistema de medida.

Para ello, el transductor a menudo incluye un sensor y el resto de etapas necesarias para el acondicionamiento de la señal (conversión, amplificación, filtrado, linealización …etc. (Fig. 1)

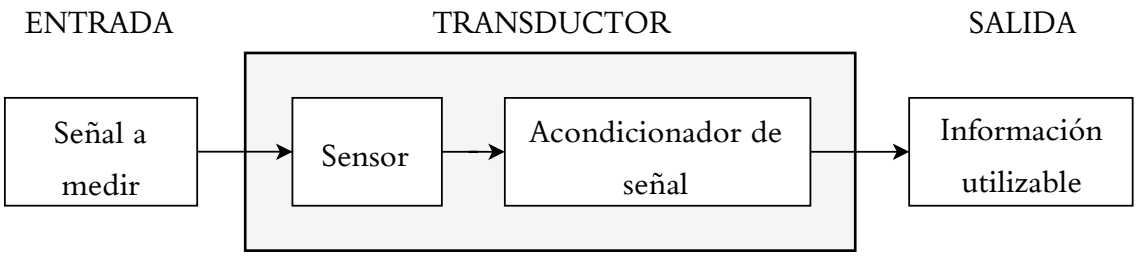


Fig. 1: Diagrama de bloques de un transductor

Pero para la obtención de un transductor correcto resulta imprescindible seleccionar aquel sensor que reúna las características técnicas mencionadas en la Tabla 1.

4.1.2. Características técnicas

Características dinámicas

Son aquellas que describen el comportamiento del sensor en el régimen transitorio de la medida. Algunas de ellas son:

- Ancho de banda (BW)
- Retardo
- Tiempo de establecimiento
- Sobreoscilación
- Distorsión armónica
- Constante de tiempo

En el caso que nos ocupa, resulta interesante procesar, no solo las variaciones de las señales en el tiempo, también su valor cuando permanecen constantes, lo que significa que el BW de la señal estará comprendido entre una frecuencia mínima de 0 Hz (corriente continua), y una máxima fmax. Como se ha explicado, las magnitudes a medir son extremadamente lentas; en el caso de la temperatura, por ejemplo, la fmax suele ser varios órdenes de magnitud inferior al hercio.

Teniendo esto en cuenta, a efectos prácticos, podemos considerar que la señal eléctrica obtenida será corriente continua (DC), por lo que no será necesario analizar las características dinámicas de los sensores que veremos a continuación.

Características estáticas

Representan la actuación del sensor en el régimen permanente. Resultan de especial interés:

- Exactitud: La capacidad de ofrecer un resultado cercano al auténtico valor de la medida.
- Incertidumbre: Margen probable de error, que contiene el valor real.
- Resolución: A veces denominada sensibilidad, es la mínima variación de la magnitud de medida que da lugar a una variación cuantificable de la salida.
- Precisión: Grado de concordancia entre sucesivas medidas.
- Sensibilidad o Ganancia: Es la relación entre el incremento de la magnitud de entrada y la variable de salida.
- Rango de medida: Intervalo de valores admisibles de la magnitud de entrada

4.1.3. Tipos de sensores

Entre los criterios de clasificación frecuentes de sensores electrónicos se encuentran aquellos que los dividen según la magnitud o variable física o química a detectar, según el tipo de propiedad sensible, según el aporte de energía y, muy especialmente, según el tipo de señal de salida. Distinguimos dos categorías:

- Sensores **Analógicos**: Serán aquellos que proporcionen señal continua y con un valor definido para cada instante de tiempo, que es función de la señal de entrada. Puede adoptar infinitos valores.
- Sensores **Digitales**: Generan una señal discreta en el tiempo, es decir, tiene un número finito de valores. Los componentes electrónicos digitales ofrecen una salida binaria, pudiendo tomar dos valores: nivel alto (1) y nivel bajo (0). Mediante la adecuada codificación, las combinaciones de ceros y unos de transforman en datos representativos de la señal de entrada.

En este caso, se pretende digitalizar las medidas de los sensores, para su posterior difusión, por lo que la obtención de una señal digital es fundamental. No obstante, todas las señales físicas son de naturaleza analógica, por lo que, será necesario realizar una conversión. Aquí es donde entra en juego el conversor Analógico a Digital o ADC (*Analog-to-Digital Converter*), un componente electrónico que, mediante una serie de procesos matemáticos, transforma la señal analógica en digital.

Para ello, elimina las frecuencias indeseadas mediante un filtro *anti-aliasing*, muestrea y discretiza la señal analógica a partir de un determinado periodo de muestreo, para finalmente codificar la señal cuantizada.

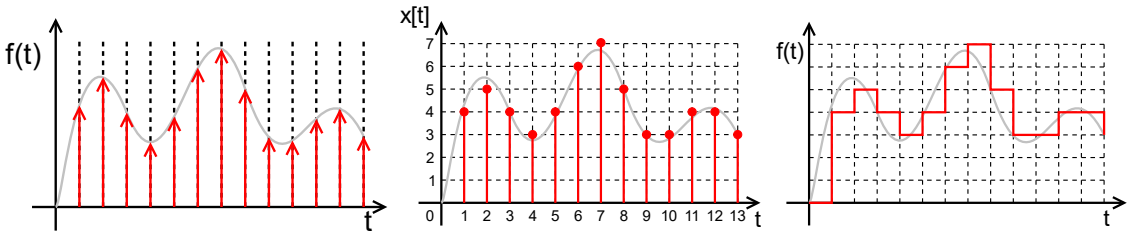


Fig. 2: Discretización de una señal analógica (www.wikimedia.org)

En la medida de lo posible, se emplearán sensores digitales comerciales ya existentes. Mediante el uso de sensores digitales se reduce la complejidad de la instrumentación y se acotan la imprecisión y exactitud. El registro electrónico de señales ambientales es muy habitual, por lo que existe una gran variedad de sensores comerciales aptos para esta aplicación.

4.1.4. Sensores digitales de temperatura y humedad

Algunos tipos de sensores de T son de tipo semiconductor como el LM35, termistores como el MF52, o completamente digital, como el sensor DHT11, que integra en un mismo componente un termistor sensible a la T y un sensor capacitivo sensible a la HR. Esto último sería lo que mejor se adaptaría a nuestro proyecto, ya que aúna la medición de dos parámetros interesantes con la salida digital. Siguiendo esa misma línea, analizaremos algunos termohigrómetros digitales en circuitos integrados de bajo coste. Para asegurar unas prestaciones mínimas, se han considerado únicamente sensores capaces de medir el rango completo de HR (0 – 100 %).

4.1.4.1. Aosong DHT22

El DHT22, también conocido como RHT03, o AM2302 en su versión cableada, es un popular sensor digital de T y HR, con unas especificaciones que superan ampliamente a las del DHT11; el rango de medida de T va desde -40 ºC hasta los 80 ºC con una precisión de ± 0,5 ºC y una resolución de 0,1 ºC, mientras que la medida de HR presenta una precisión del 2 %, que puede llegar al 5 % en los valores límite (< 10 % y > 90 %).

La transmisión de datos se realiza a través de una interfaz digital de un solo hilo, (*Single Bus Protocol*) proporcionando una señal digital calibrada que contiene los

valores de ambas medidas. Para ello, cada sensor se calibra en la fábrica, donde los coeficientes de calibración se graban permanentemente en la memoria OTP (*One Time Programmable).*

Al precisar de un único pin para la comunicación, la implementación del hardware resulta muy sencilla en plataformas como Arduino o Raspberry Pi, empleando las librerías de Adafruit. Su principal desventaja es su periodo de muestreo, no inferior a 2 s. Es posible superar dicha frecuencia de 0,5 Hz indicada por el fabricante, pero ello produciría un calentamiento del sensor, con lo que, dada la gran dependencia de la HR con la T, ambas medidas dejarían de ser representativas del entorno.

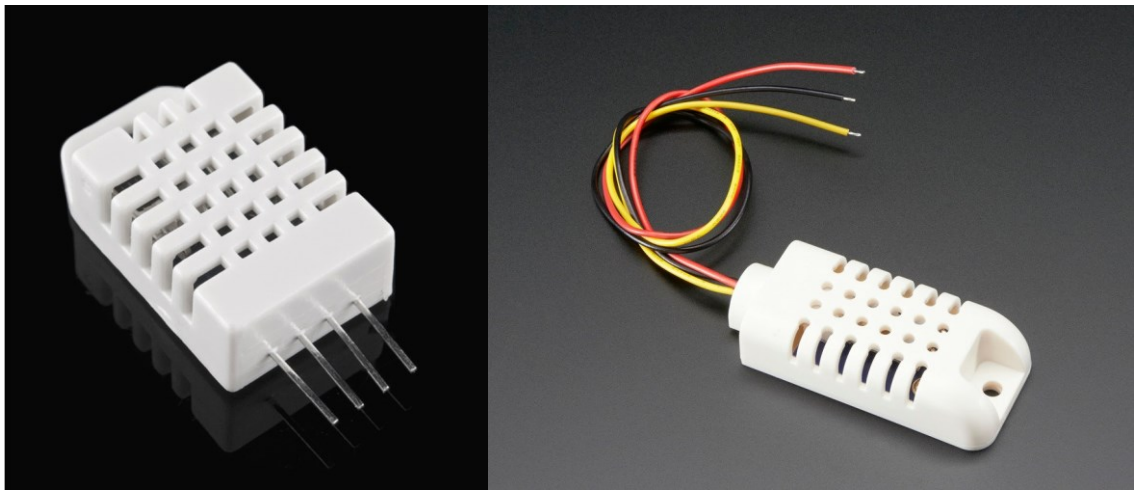


Fig. 3: DHT22 (izqda.) y AM2302 (dcha.) (www.adafruit.com)

4.1.4.2. Measurement Specialities HTU21D

De la misma manera que el DHT22, los sensores HTU21 son probados y calibrados individualmente por el fabricante.

La versión HTU21DF incluye una membrana protectora de Politetrafluoroetileno (PTFE) sobre el elemento sensible, que puede resguardarlo de la contaminación ambiental, aunque no tiene ningún impacto en la calidad de las medidas. Además, existen diversos módulos digitales de terceros (breakout board / DiY board) que conectan los pines del sensor a una PCB (*Printed Circuit Board*) que dispone de su propio patillaje y otra circuitería para su uso sin necesidad de soldar.

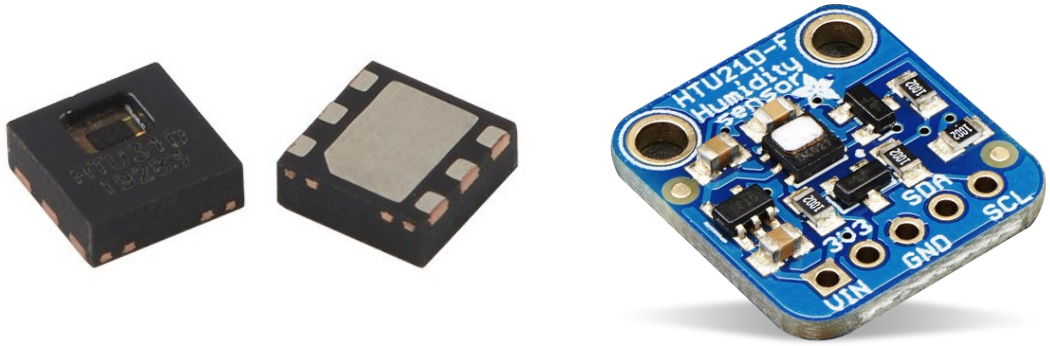


Fig. 4: HTU21D (izqda.) y HTU21DF (dcha.) sobre placa de evaluación Adafruit®

La precisión típica en las medidas de T es de ± 0,3 ºC entre -40 y 90 ºC, mientras que las medidas de HR presentan una precisión del ± 2 %.

En cuanto a la resolución, por defecto, este sensor posee una resolución de 12 bits para la medida de humedad y 14 bits para la de temperatura, pero estos valores se pueden configurar para reducir el tiempo de cada medición y por ende el consumo. No obstante, no es posible ajustar la resolución de las dos magnitudes de forma independiente: dado que se utilizan 2 bits para la configuración de la resolución, solo son posibles 4 combinaciones.

La transmisión de realiza a través del protocolo I²C (*Inter-Integrated Circuit*), uno de los estándares de comunicaciones más extendidos.

4.1.4.3. Sensirion SHT31

La serie de sensores SHT3x de Sensirion destacan por su extraordinario rendimiento en relación a su precio. Con un amplio rango de temperatura que va desde los -40 ºC hasta los 125 ºC, resultan aptos para cualquier escenario posible en este proyecto.

Existen tres versiones del sensor: SHT30, SHT31 y SHT35, diferenciándose ligeramente en la precisión de ambas medidas, pero con idénticas características en el resto de especificaciones. Al igual que el HTU21, se pueden adquirir independientemente o montados sobre una placa, presentan una interfaz digital basada en I²C – con la ventaja de poder seleccionar entre 2 direcciones – y pueden incluir el mencionado revestimiento de teflón con un grado de protección IP67 (*Ingress Protection*), pero resulta innecesario para la aplicación que nos ocupa.

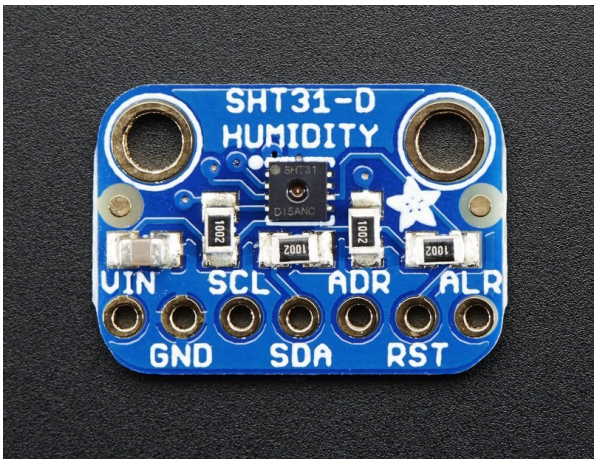


Fig. 5: SHT31 en Breakout Board de Adafruit®

Funcionan entre 3 y 5 V, por lo que pueden utilizarse con prácticamente cualquier microcontrolador. Además, incluyen un pequeño elemento calefactor (25 mW) que puede resultar útil para probar su funcionamiento, y permiten configurar una salida de alarma, que puede utilizarse para generar interrupciones al superar un determinado valor umbral. A modo de resumen, las tablas 2 y 3 recogen las especificaciones principales de los sensores mencionados:

Especificaciones del fabricante - HUMEDAD

Sensor	DHT22	SHT31	HTU21D
Rango de operación (%)	0 a 100	0 a 100	0 a 100
Precisión (%RH, 25°C)	± 2 (10 a 90 %)	± 2 (0 a 90 %)	± 2 (20 a 80 %)
Resolución (%)	0,1	0,01	0,04 a 0,7
Repetibilidad (%)	± 1	± 0,1	-
Estabilidad (% por año)	< 0,5	< 0,25	< 0,5
Tiempo de respuesta (s)	5	8	5

Tabla 2: Especificaciones técnicas de la medida de HR de los sensores DHT22, SHT31 y HTU21D

Especificaciones del fabricante - TEMPERATURA

Sensor	DHT22	SHT31	HTU21D
Rango de operación (°C)	-40 a 80	-40 a 125	-40 a 125
Precisión (°C)	± 0,5	± 0,3	± 0,3
Resolución (°C)	0,1	0,015	0,01 a 0,04
Repetibilidad (%)	0,2	0,06	-
Estabilidad (°C / año)	-	< 0,03	< 0,04
Tiempo de respuesta (s)	2	2	10

Tabla 3: Especificaciones técnicas de la medida de T de los sensores DHT22, SHT31 y HTU21D

4.1.5. Sensores digitales de luminosidad

De forma similar a la selección de termohigrómetros, se han escogido sensores de L digitales, asequibles y de dimensiones reducidas, capaces de comunicarse directamente con un MCU (*Microcontroller Unit*).

4.1.5.1. AMS TSL2561

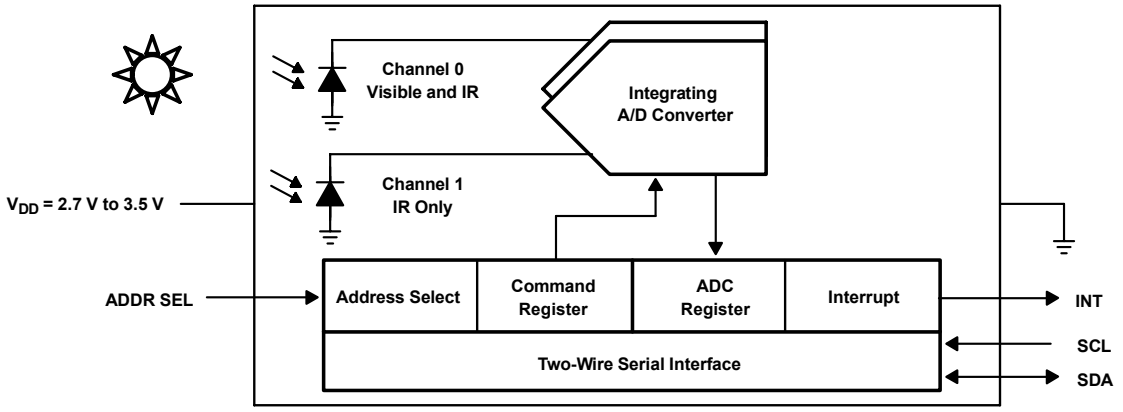


Fig. 6: Diagrama de bloques funcional del TSL2561 (www.ams.com)

Este módulo de sensor de luz ambiental dispone de dos fotodiodos que convierten la señal de L en una señal de corriente eléctrica. Como se observa en el diagrama de bloques, los fotodiodos alimentan a sendos conversores ADC, que a su vez integran la señal de forma sincrónica para generar una representación digital de la intensidad lumínica. Contiene un diodo exclusivo para las medidas de IR, y otro capaz de medir el espectro completo, lo que significa que podemos obtener la medida de L visible por diferencia entre ambas señales en un rango de 0,1 hasta 40.000 lx.

Es posible seleccionar entre 3 direcciones para la transmisión de datos mediante I^2C. También permite configurar los tiempos de integración, la sensibilidad y , al igual que el SHT31, un umbral para elevar una interrupción.

4.1.5.2. ROHM Semiconductor BH1750

El BHT1750 es un sensor diseñado apto para medir en un amplio rango de iluminancia, entre 1 y 6555,5 lx con una resolución máxima de 1 lx. Nuevamente presenta una interfaz I²C para la transmisión de las medidas, con 2 posibles direcciones para el modo esclavo. A diferencia del modelo anterior, incluye un único fotodiodo de banda ancha conectado a un ADC de 16 bits, así que no dispone de un mecanismo específico para separar la luz IR, si bien el fabricante asegura que su influencia es mínima. También cuenta con su librería de Arduino, creada y mantenida por la comunidad, cuyos autores afirman que es simple, robusta y compatible con diversas arquitecturas.

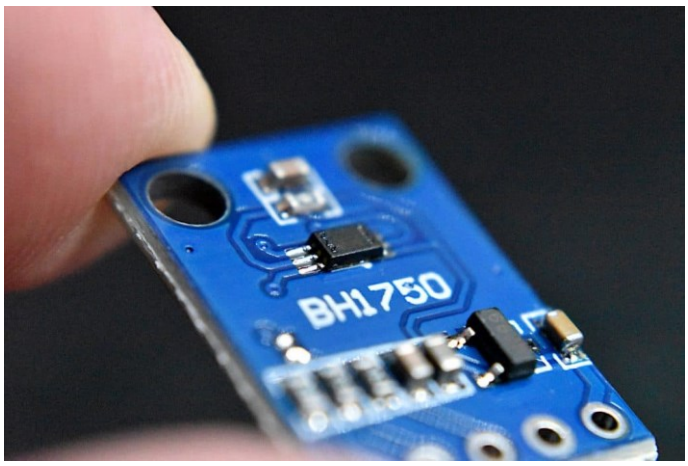


Fig. 7: BH1750 en Breakout Board

4.1.5.3. Vishay Semiconductors VEML6075

Este sensor óptico de luz ultravioleta mide la radiación entre 280 y 400 nm, incluyendo en este rango la a UV-B (280 a 315) y la UV-A (315 a 400). Los rayos UVA alcanzan en su totalidad la superficie terrestre y representan el 95 % de la luz UV, mientras que los UVB son absorbidos en su mayoría por las nubes y la capa de ozono, y suponen el 5 % restante. Son los dos tipos de radiación más peligrosa ya que, a diferencia de la UVC, atraviesan con facilidad la troposfera, y no pueden bloquearse con vidrios normales o telas demasiado finas.

Dispone de un canal individual para cada medida, con una resolución de 16 bits. Proporciona compensación de temperatura para una señal de salida estable, entre -40 ºC y +85 ºC, e incorpora el circuito necesario para transmitir sus mediciones a través del bus I²C.

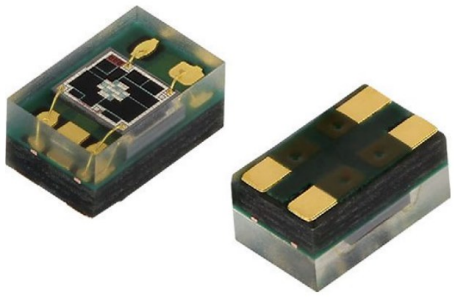


Fig. 8: Sensor VEML6075 en su encapsulamiento OLGA de 4 pines (mouser.es)

4.2. Elección de una estación meteorológica inalámbrica

La selección de la estación meteorológica a emplear supone un aspecto crítico del proyecto y, junto a la comunicación, presenta una de las novedades más destacables respecto a trabajos previos en materia de monitorización de parámetros ambientales.

Para sopesar las distintas opciones se valorará la facilidad de integración de la estación con el resto del sistema de sensores. Asimismo, el aspecto económico cobra particular importancia; por sí sola, la estación constituye un sistema de medida independiente, lo que queda reflejado en el coste.

Con el fin de resumir los apartados que siguen y evitar la redundancia, se resumen algunas características fundamentales que compartirán todas las estaciones que se incluyen en esta memoria, salvo que se indique lo contrario.

4.2.1. Características generales de una estación inalámbrica

A diferencia de la estación cableada convencional, la consola de la estación meteorológica cuenta con dos niveles de conectividad inalámbrica. Permite conectarse a Wi-Fi (2,4 GHz) mediante su chip WLAN (*Wide Local Area Network*) integrado y configurar dicha conexión para subir los datos a redes de estaciones meteorológicas como:

- Weather Underground
- Weathercloud
- Meteoclimatic

Por su parte, los sensores inalámbricos también se comunican con la consola a través de radiofrecuencia (RF). Por lo general, operan en la banda ISM (*Industrial, Scientific & Medical*) europea de 868 MHz, aunque algunos modelos antiguos

funcionan a 433 MHz. De forma esquemática, la comunicación queda resumida de la siguiente forma:

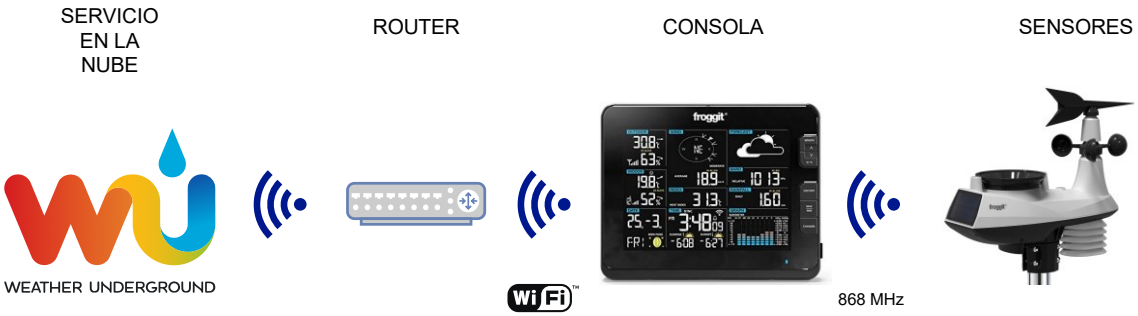


Fig. 9: Esquema de la conexión a internet de una estación meteorológica inalámbrica

En cuanto al hardware, se distinguen dos partes fundamentales en la estación climática:

- **El bloque integrado de sensores**. Unidad exterior.

Es el conjunto de sensores destinado a la medida de parámetros meteorológicos en el exterior. Al conjunto de sensores que incluye, se le conoce como una *matriz de sensores Osprey*. La matriz básica cuenta con los siguientes componentes:

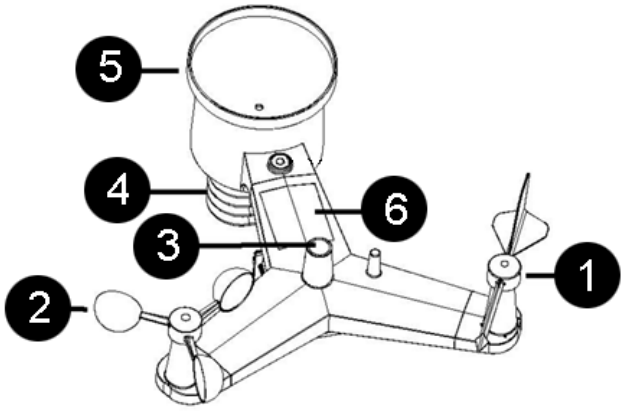


Fig. 10: Matriz de sensores Osprey de la estación Fine Offset WH65

1. Veleta de viento, para la medida de la dirección del viento
2. Anemómetro de molinete, para la medida de la velocidad del viento
3. Sensor de radiación solar y luminosidad (UV y L)
4. Termo-higrómetro digital, para las medidas de T y HR, con un escudo de radiación de pagoda que protege los sensores del sol y de la lluvia

5. Colector de lluvia o pluviómetro, para la medida de las precipitaciones
6. Colector solar, para prolongar la autonomía.

Por lo tanto, se asume que las estaciones incluyen la electrónica necesaria para captar las principales Magnitudes de interés (T, HR y UV), con la principal limitación de que sólo pueden hacerlo en un punto; recordemos que interesa hacer medidas multipunto para detectar gradientes de T y puntos calientes sobre el soporte rocoso del arte rupestre.

- **La consola**

A través de una pantalla típicamente LCD, la consola permite, por un lado, llevar a cabo la configuración de todos los aspectos de la estación, y por otro, visualizar las medidas principales, junto a otra información secundaria, basada en cálculos e interpretaciones de las mismas, como el índice de calor, el punto de rocío, o un pronóstico meteorológico.

Adicionalmente, muchas de las estaciones incluyen un sensor termo-higrométrico para interiores, igualmente inalámbrico, alimentado por baterías.

Tras esta corta introducción, de ahora en adelante se detallarán directamente las especificaciones técnicas en materia de comunicación inalámbrica y precisión y frecuencia de medición de los sensores, mencionando aquellas características distintivas o particulares.

Para comprender mejor las opciones reales de que disponemos, se estudian los fabricantes de la electrónica de estaciones meteorológicas.

4.2.2. Fine Offset

La marca Fine Offset pertenece a la empresa china Fine Offset Electronic, Co. Ltd. (*Limited Company*, equivalente anglosajón a Sociedad Limitada). con sede en la ciudad de Shenzen, que desde hace varios años fabrica las estaciones meteorológicas más conocidas del mercado, con gran cantidad de modelos entre los que destaca la WS1080 por su gran éxito de ventas. La empresa vende sus productos a través de terceras empresas, que son las que dan nombre al producto y lo comercializan. Por tanto, no se venden bajo el nombre comercial de Fine Offset, sino bajo marcas como:

- Ambient Weather
- Ecowitt
- Froggit
- Elecsa
- Maplin
- National Geographic

- PCE
- Waldbeck
- Sainlogic
- Watson

Entre otros vendedores que utilizan el mismo hardware. Por ello, existen multitud de modelos distintos de marcas diferentes, que en realidad son el mismo producto con otra denominación, hallándose a menudo la única diferencia en la consola que acompaña a la estación, y por tanto, en el software de control de la estación y de envío de datos.

4.2.2.1. Fine Offset WH65 / WH69

Fig. 11: Bloques de sensores integrados WH65 (izqda.) y WH69 (dcha.)

Ambos modelos fueron lanzados en 2018 y comparten todos los sensores diferenciándose en la forma del bloque.

Además de todos los sensores detallados en la lista anterior, poseen un indicador de nivel de burbuja y barómetro. En cuanto a la alimentación, incluyen el mencionado panel solar, que alimenta a un supercondensador para energizar el sistema, y dos pilas AA de respaldo no recargables. El envío de datos de la consola a la nube se efectúa cada 16 s como mínimo. Según el fabricante, la propagación en la línea de visión alcanza los 100 m, pero la mayoría de instalaciones reales presentan obstáculos que impiden la visibilidad directa entre emisor y receptor, con lo que normalmente se obtendrá un máximo de 30 m.

La WH65 se puede hallar en el mercado con los siguientes nombres:

- Fine Offset WH2900
- Froggit WH3000
- Misol Wireless
- Sainlogic WS3500
- Ambient Weather WS-2902
- Instrument Choice IC0370

Los modelos presentan variaciones estéticas, como la combinación de colores, y anemómetro y veleta en ocasiones aparecen intercambiados, pero las características técnicas, mostradas en la Tabla 4, son idénticas.

Variable	**Resolución**	**Rango**	**Precisión**
Humedad	1 %	10 % a 99 %	± 5 %
Temperatura interior	0,1 °C	– 30 a 65 °C	± 1°C
Temperatura exterior	0,1 °C	– 40 a 60°C	± 1°C
Precipitaciones	0,1 mm	0 a 6000 mm	± 10%
Dirección del viento	-	0 a 360°	-
Velocidad del viento	0,1 km/h	0 a 180 km/h	3,6 km/h
Luminosidad	-	0 a 200k lx	± 15 %
Presión barométrica	0,1 hPa	700 a 1100 hPa	± 3 hPa

Tabla 4: Especificaciones técnicas de los bloque de sensores Fine Offset WH65 y WH69

4.2.2.2. Fine Offset WH80

Fig. 12: Fine Offset WH80 sobre mástil

Este modelo sustituye la veleta y el anemómetro mecánicos por sensores ultrasónicos. Resulta un cambio interesante ya que la eliminación de partes móviles suele prolongar la vida útil del sistema. Los datos se envían con una frecuencia mayor, cada 4,75 s, con un alcance también superior, de hasta 300 m. No obstante, no incluye pluviómetro, por lo que sería necesario adquirirlo por separado.

Variable	Resolución	Rango	Precisión
Humedad	1 %	10 % a 99 %	± 5 %
Temperatura interior	0,1 °C	0 a 50 °C	± 1 °C
Temperatura exterior	0,1 °C	– 40 a 60°C	± 1 °C
Dirección del viento	-	0 a 360°	-
Velocidad del viento	0,1 km/h	0 a 160 km/h	3,6 km/h
Luminosidad	-	0 a 200k lx	± 15 %
Presión barométrica	0,1 hPa	300 a 1100 hPa	± 5 hPa

Tabla 5: Especificaciones técnicas del bloque de sensores Fine Offset WH80

4.2.3. Davis Instruments®

Fig. 13: Logotipo de Davis Instruments®

Esta empresa Norteamericana se dedica a la venta y fabricación de productos relacionados con la meteorología, termometría, medición y control electrónico. Comercializa estaciones meteorológicas profesionales de alta gama, repuestos de las mismas y accesorios adicionales. De nuevo, los sensores van instalados en bloques compactos todo-en-uno, a los que el fabricante hace referencia como ISS (*Integrated Sensor Suite*). Dicha suite de sensores se comunica con la consola a través de un método de transmisión de señales de radio denominado espectro ensanchado por salto de frecuencia (del inglés *Frequency Hopping Spread Spectrum* o FHSS). Se trata de una tecnología certificada por el FCC (*Federal Communications Commission*) que transmite en una banda sin licencia.

Este método de transmisión de señales presenta un alcance muy superior al de los modelos anteriores, convirtiéndolo en una de características distintivas de las estaciones inalámbricas Davis. Destacan tres modelos:

- Davis Vantage Vue
- Davis Vantage Pro
- Davis Vantage Pro2

La Davis Vantage Pro se encuentra actualmente descatalogada, así que a continuación veremos las otras dos alternativas.

4.2.3.1. Davis Vantage Vue 6250

Fig. 14: Davis Vantage Vue 6250 sobre mástil

A diferencia de las estaciones de Fine Offset, la Vantage Vue solo se puede acoplar al extremo final de un mástil, que no va incluido. Este modelo posee nivel de burbuja integrado, pero no incluye sensor de L de ningún tipo. Por otra parte, su carcasa es especialmente resistente a la humedad y a la corrosión, soportando mejor la lluvia ácida.

Además, no permite la integración de más sensores a parte de los incluidos por fabricación, por lo que solamente existe un modelo: el 6250,

Variable	**Resolución**	**Rango**	**Precisión**
Humedad	1 %	1 % a 100 %	± 5 %
Temperatura interior	0,1 °C	0 a 60 °C	± 1 °C
Temperatura exterior	0,1 °C	– 40 a 65 °C	± 1 °C
Precipitaciones	0,2 mm	0 a 6553 mm	± 10 %
Dirección del viento	-	0 a 360°	-
Velocidad del viento	1 km/h	3 a 290 km/h	-
Presión barométrica	0,1 hPa	540 a 1100 hPa	± 1 hPa

Tabla 6: Especificaciones técnicas de la estación Davis Vantage Vue 6250

4.2.3.2. Davis Vantage Pro2 / Pro2 Plus

Fig. 15: Davis Vantage Pro2 (izqda.) y Pro2 Plus (dcha.)

Esta estación presenta la ventaja de ser modular, es decir, se permite añadir múltiples sensores al bloque básico. Por otra parte, destaca respecto al resto de estaciones por varias cuestiones:

- Presenta una frecuencia de actualización de las medidas muy superior: hasta 2,5 s.
- El alcance de transmisión inalámbrica llega a 300 m entre la estación y la consola.
- Como se puede apreciar en la fotografía, anemómetro y veleta son desmontables, y se pueden colocar en una posición superior al resto de sensores para una medición más precisa.
- El colector de lluvia está rodeado de agujas para protegerlo de las aves.

Versiones disponibles:

La 6152 es la versión más popular. Se alimenta mediante un panel solar de 0,5 W y una pila de litio de 3 V y dispone del pluviómetro, anemómetro, veleta, termómetro e higrómetro.. El resto de modelos poseen añadidos respecto a esta versión básica.

- Las versiones 61XXC son las cableadas, que quedan descartadas.

- Las versiones 61X3X incluyen un sistema de ventilación de la pantalla protectora de radiación solar, y otro panel solar para alimentar unas baterías de níquel recargables.
- Las versiones 616XX incorporan un sensor de UV y de radiación solar. Se comercializan como **Vantage Pro2 Plus**, y es con diferencia la de mayor precio, encontrándose en torno a los mil euros.

Las especificaciones del ISS con todos los sensores, correspondiente a la Vantage Pro2 Plus, se detallan en la siguiente tabla:

Variable	**Resolución**	**Rango**	**Precisión**
Humedad	1 %	1 % a 100 %	± 2 %
Radiación solar	1 W/m²	0 a 1800 W/m²	± 5 %
Radiación UV	0,1 MEDs	0 a 199 MEDs	± 5 %
Temperatura interior	0,1 °C	0 a 60 °C	± 0,3 °C
Temperatura exterior	0,1 °C	– 40 a 65 °C	± 0,3 °C
Precipitaciones	0,2 mm	0 a 6553 mm	± 3 %
Dirección del viento	1°	1 a 360°	± 3 %
Velocidad del viento	1 km/h	0 a 322 km/h	± 3,2 km/h
Presión barométrica	0,1 hPa	540 a 1100 hPa	± 1 hPa

Tabla 7: Especificaciones técnicas de la estación Davis Vantage Pro2 Plus

4.3. Elección de la tecnología inalámbrica

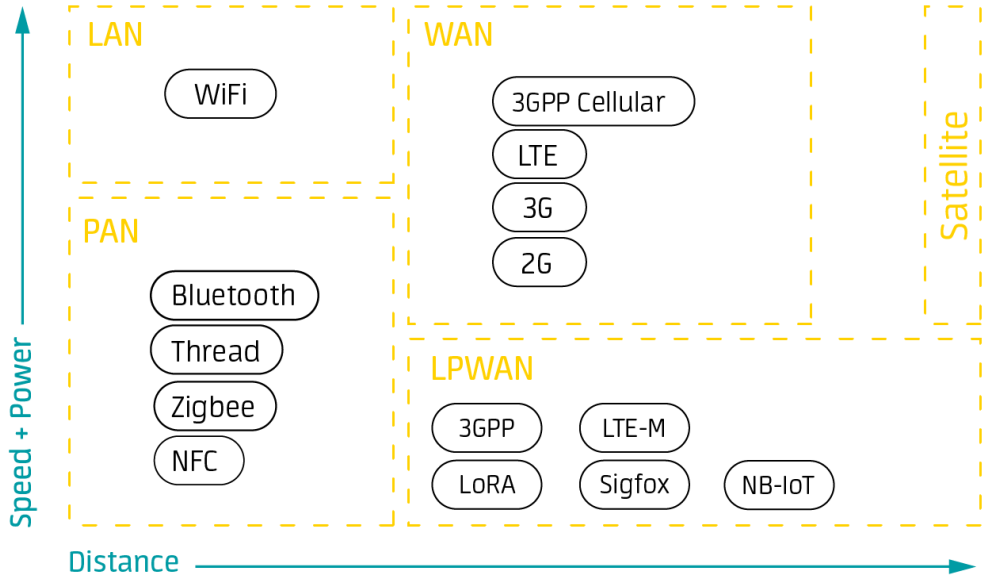


Fig. 16: Alcance y velocidad de diferentes tecnologías de conectividad inalámbrica (Avsystem.com)

Como se ha explicado, el presente trabajo consiste, entre otras cosas, en un proyecto exploratorio para investigar las distintas formas de telecomunicación disponibles en zonas de difícil acceso a internet.

Dadas las necesidades discutidas en la sección 3, una de las principales problemáticas a resolver surge de la necesidad de reunir y procesar información para finalmente presentarla de forma que resulte útil para la toma de decisiones, todo ello de modo eficiente desde el punto de vista energético, y económicamente asequible.

Para orientar la labor de estudio de las opciones disponibles, se debe partir de un enfoque claro, así que el análisis de alternativas de tecnologías de comunicación se encauzará dentro del paradigma IoT, descartando toda opción alámbrica como Ethernet. En su lugar, se optará por medios de transmisión no guiados (radio, microondas, infrarrojos...).

4.3.1. Comunicaciones móviles

4.3.1.1. GSM

Fig. 17: Logotipo de la tecnología GSM

El sistema de comunicaciones GSM (Originalmente acrónimo de *Groupe Spécial Mobile*, ahora oficialmente *Global System for Mobile Communications*) es una especificación desarrollada por el ETSI (*European Telecommunications Standards Institute*) e introducida en 1992, y abarca la segunda generación de tecnologías de comunicaciones móviles, conocida como **2G**, orientadas a la transmisión de servicios móviles de voz y datos encriptados digitalmente.

GSM utiliza las bandas designadas por el ITU (*International Telecommunication Union*), de 900 MHz y 1800 MHz en Europa, Oriente Medio, África y Oceanía, y 850 y 1900 MHz en EE.UU. Dichas bandas son denominadas GSM-900, GSM-1800...etc.

Con el fin de repartir el BW entre las distintas compañías operadoras, GSM implementa diversos métodos de acceso múltiple por división (TDMA , SDMA, FDMA, CDMA...)

La norma GSM autoriza un flujo máximo de 9,6 kbps, que permite la transferencia de voz y mensajes de texto, aunque requiere una conexión permanente entre emisor y receptor, lo que genera una serie de inconvenientes.

La comunicación se basa en células de tamaño variable en función del entorno (rural o urbano). Cada una de ellas posee un transceptor central llamado estación de base o BTS (*Base Transciever Station*), y colinda con otras 6 de frecuencia distinta.

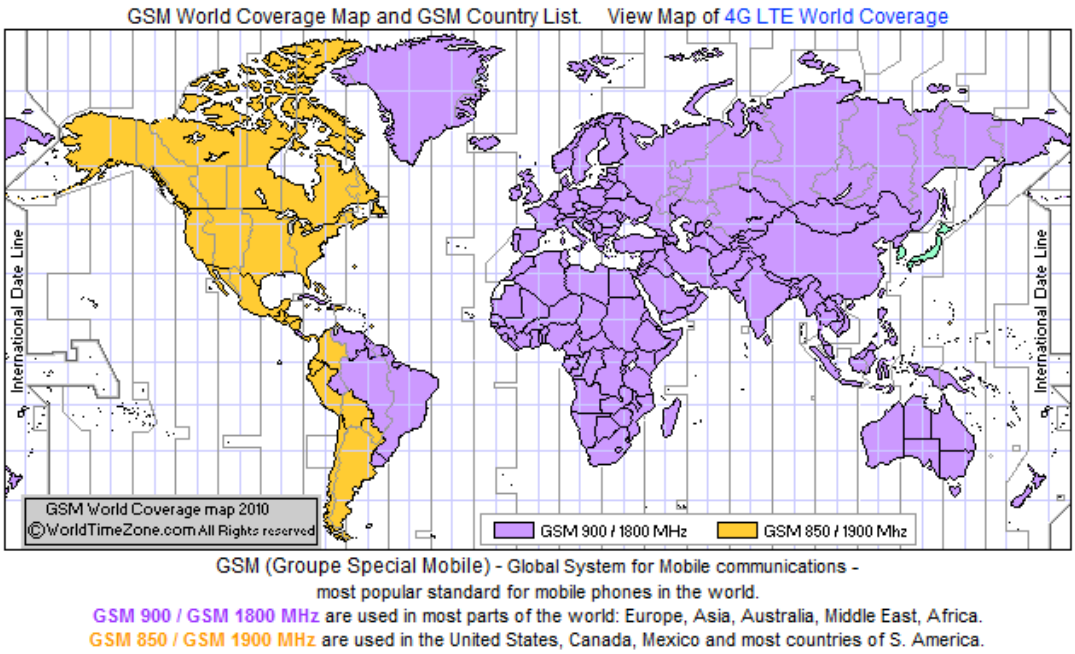


Fig. 18: Mapa de cobertura global de la tecnología GSM, con las bandas de frecuencia diferenciadas (www.WorldTimeZone.com)

4.3.1.2. GPRS

Fig. 19: Logotipo de la tecnología GPRS

La especificación GPRS (*General Packet Radio Services*) fue desarrollada en la década de los 80 por el ETSI sobre la plataforma GSM y se basa en el protocolo TCP/IP, por lo que también se conoce como GSM-IP. Actualmente es mantenida por el grupo

3GPP, (*Third Generation Partnership Project*), una asociación que reúne a organismos de estandarización nacionales (como la propia ETSI), con el objeto de desarrollar las especificaciones técnicas de la tercera generación de telecomunicaciones móviles celulares.

Llamado informalmente 2.5G, este estándar supone un mejora respecto a la tecnología GSM, aunque técnicamente sigue perteneciendo a la segunda generación. Al estar orientado a la transferencia de datos móviles a través de paquetes, utiliza los recursos de la red solo cuando existen datos listos para enviar, lo que presenta una serie de ventajas para el tratamiento de la información, y suprime la necesidad disponer de una conexión constante entre ambos extremos de la comunicación.

Por otra parte, la velocidad aumenta a un máximo de 114 kbps; este incremento en el *baud rate* hace la tecnología especialmente útil para aplicaciones basadas en transferencia de datos como el internet móvil clásico y la consulta de páginas web, y desbloquea nuevas funcionalidades como la obtención de cartografía on-line.

Además, GPRS introduce una serie de servicios:

- Aplicaciones en red a través del protocolo WAP.
- Posibilidad de utilizar el dispositivo como módem USB.
- Servicio de mensajes cortos (SMS).
- Servicio de mensajería multimedia (MMS).
- Servicios Peer-to-peer (P2P) utilizando el protocolo IP
- DGPS (*Differential Global Positioning Sysem*), que presenta una precisión superior al GPS

Todas estas mejoras y actualizaciones permitieron a GPRS acelerar la transición a las redes de banda ancha móvil 3G, que vemos a continuación.

4.3.1.3. UMTS

La tercera generación de redes de telefonía móvil o 3G pertenece el estándar global IMT-2000, definido por el ITU. Este estándar proporciona un marco para el acceso inalámbrico universal combinando los diversos sistemas de redes terrestres y satelitales, y engloba una serie de especificaciones, entre las que destaca UMTS (*Universal Mobile Telecommunications Service*), desarrollada y mantenida por 3GPP.

De nuevo, la adopción de esta tecnología implica un salto cuantitativo en todos los aspectos técnicos: Presenta un funcionamiento similar al GPRS, pero con un mayor flujo de datos, mejor calidad de voz, y cobertura potencialmente mundial, por lo que durante mucho tiempo se convirtió en la tecnología prioritaria para los desarrolladores de software y aplicaciones.

Como interfaz principal, la comunicación emplea un nuevo método de acceso múltiple por división denominado WCDMA (*Wideband Code Division Multiple Access*), a través de dos canales de 5 MHz dentro de un BW total de 2 GHz, pudiendo alcanzar 144 kbps en entornos rurales, y un máximo teórico de 2 Mbps en los urbanos.

4.3.1.4. LTE

Fig. 20: Logotipo oficial de la tecnología LTE

El estándar LTE (*Long Term Evolution*) es llamado a veces 3.9G o 3.95G por no cumplir los requisitos definidos por la IMT-Advanced en características de velocidades pico de transmisión y eficiencia espectral para clasificar como 4G (a diferencia de su versión mejorada, LTE-Advanced), pero ha sido ampliamente comercializado como tal y se trata de uno de los estándares más extendidos.

La telefonía móvil 4G se basa completamente en el protocolo IP (*Internet Protocol*) gracias a la convergencia entre las redes cableadas e inalámbricas, por lo que puede ser utilizada por toda clase de dispositivos móviles. Es una versión moderna y renovada del 3G con mejoras fundamentales en cobertura y velocidad, empleando una técnica avanzada de acceso al medio denominada OFDM (*Orthogonal Frequency Division Multiplexing*) para el *downlink*, manteniendo la retrocompatibilidad con 3G para asegurar la cobertura en zonas rurales.

A diferencia de las redes anteriores, basadas parcial o totalmente en la conmutación de circuitos, LTE se utiliza únicamente la **conmutación de paquetes**, con la

consiguiente optimización de la infraestructura. Como consecuencia, presenta un red de gran cobertura, fácil de desplegar y mantener.

Por otra parte, las franjas de frecuencia difieren en función del país, por lo que un dispositivo no funcionará en diferentes regiones a menos que sea multibanda.

También se caracteriza por un consumo mayor en los dispositivos, y resultar más cara que su antecesora, ambos factores esenciales en el criterio de selección de la tecnología inalámbrica explicado previamente.

4.3.2. Redes LPWAN

Las redes LPWAN son una subcategoría de tecnologías de conectividad IoT; como indican sus siglas, logran cubrir un área extensa, manteniendo un consumo energético bajo, tanto desde el punto de vista de la potencia de la transmisión (energía irradiada), como del consumo eléctrico del *hardware*. Además, presentan la ventaja de que los costes asociados al diseño, implementación, y mantenimiento de este tipo de redes son muy inferiores al del resto de alternativas.

Al centrarse en el alcance y la eficiencia energética, las principales limitaciones son la elevada latencia de transmisión, que se traduce en velocidades de transferencia bajas (e.g. 100 bps) y el reducido BW – además de otras restricciones adicionales que dependen de la banda de frecuencia de la transmisión – lo que restringe su uso a determinadas aplicaciones: No resultaría factible para la retransmisión de video o audio, navegar por internet o realizar llamadas telefónicas, pero sí para el envío periódico de datos sencillos, como las medidas codificadas de unos sensores digitales.

Sus características la convierten en la tecnología ideal para aplicaciones IoT, donde las necesidades a satisfacer son completamente distintas a las de la conectividad convencional; en lugar de manejar cantidades ingentes de información a velocidades igualmente elevadas, el objetivo a menudo es lograr la máxima conectividad entre dispositivos físicos de lo más variopintos, formando una red heterogénea de sensores, actuadores y otros sistemas embebidos, cuya capacidad de reunir, procesar y transmitir datos a grandes distancias constituye la clave de ecosistemas como las *Smart Cities*.

Tal es el auge de este tipo de redes, que los principales operadores de redes globales han visto su potencial, y están desarrollando sus propias soluciones propietarias y open-source. A continuación vemos algunas de las más extendidas.

4.3.2.1. LoRaWAN

Fig. 21: Logotipo de LoRaWAN®

- LoRa / LoRaWAN

Resulta conveniente distinguir entre LoRa y LoRaWAN, que conjuntamente definen esta especificación de red LPWAN.

LoRa® (*Long Range*) representa la parte privativa de esta tecnología, y constituye la capa física de acceso a la red, o nivel 1 del modelo OSI (*Open System Interconnection*), esto es, los chips de radiofrecuencia capaces de utilizar esta especificación. Fue desarrollada originalmente en Francia por Cycleo, pero fue adquirida por Semtech en el año 2012, actual dueño de la patente y fabricante del *hardware* para LoRaWAN.

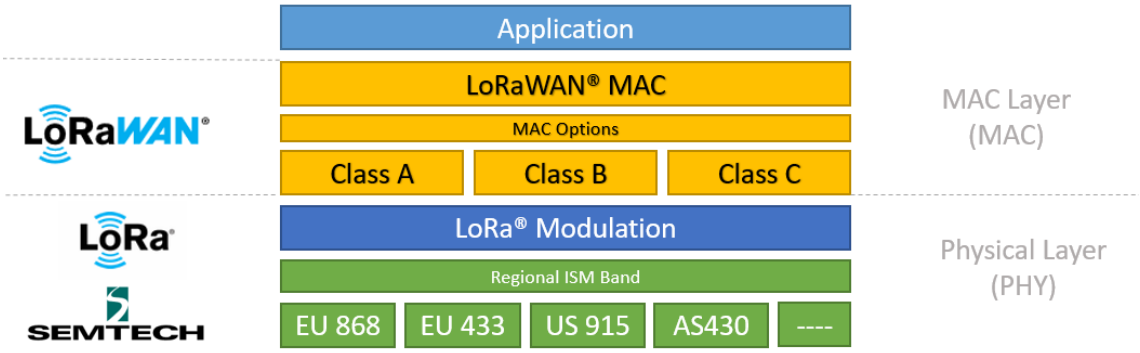


Fig. 22: Analogía entre LoRa/LoRaWAN y las capas OSI

Por su parte, LoRaWAN es un complemento a LoRa. Equivale a las capas 2 y 3 del estándar OSI, y define el protocolo de acceso a la red y la arquitectura del sistema. Se trata de un **estándar de código abierto**, que además opera las bandas de radio ISM, bandas de uso libre para propósitos industriales, científicos y médicos. En otras palabras, están habilitadas en todo el mundo para su uso no comercial y no necesitan licencia, con la condición de cumplir una serie de regulaciones.

Resulta especialmente interesante, pues recordemos que se pretende crear una red con un coste mínimo, aunque dicha red deberá estar en conformidad con la reglamentación nacional impuesta por la ETSI, que fija el *duty cycle* máximo (la fracción de tiempo que un dispositivo está transmitiendo) al 1 %.

Por las características de penetración de las ondas electromagnéticas, se trabaja en el espectro subgigahercio (Sub-GHz), pues se ha comprobado que estos márgenes de frecuencia ofrecen la mejor cobertura, especialmente en presencia de obstáculos de gran grosor, como muros. Dentro de dicho rango, las frecuencias varían según la región: normalmente, en Europa se utiliza la frecuencia de 868 MHz en ambientes urbanos con edificaciones, y menos frecuentemente 433 MHz en espacios abiertos, donde la línea de visión es mayor.

Especificaciones técnicas

Este protocolo ofrece conectividad segura mediante doble encriptación AES (*Advanced Encryption Standard*) de 128 bits en las capas de servidor y de aplicación, para el intercambio de claves entre el servidor y el nodo.

- La capa de seguridad de la red certifica la autenticidad del nodo en la red.
- La seguridad de la capa de aplicación maneja el cifrado de datos entre los nodos y el servidor de la aplicación para que los mensajes no puedan leerse o interferirse en tránsito.

La comunicación entre nodos LoRa y gateway utiliza una modulación basada en un *Chirp Spread Spectrum* (CSS), un protocolo de origen militar y de gran robustez frente a interferencias, que modula los datos sobre diferentes canales de frecuencia y velocidad. De esta forma, el gateway se adapta a las condiciones de transmisión de los nodos finales.

Por otra parte, la velocidad está directamente relacionada con otro factor denominado *Spreading Factor* (SF), que determina la cantidad de datos redundantes que se envían en la transmisión. Un valor SF elevado indica gran redundancia en los datos, y por ende máximo alcance, pero menor velocidad.

LoRa no emplea la técnica *Listen Before Talk*, mediante la cual los dispositivos analizan la red en busca de los canales menos saturados, pero las transmisiones se distribuyen en tres canales comunes de 125 kHz, lo que, añadido al reducido ciclo de trabajo impuesto por las restricciones de las bandas ISM (1 %), asegura que las colisiones sean muy inusuales.

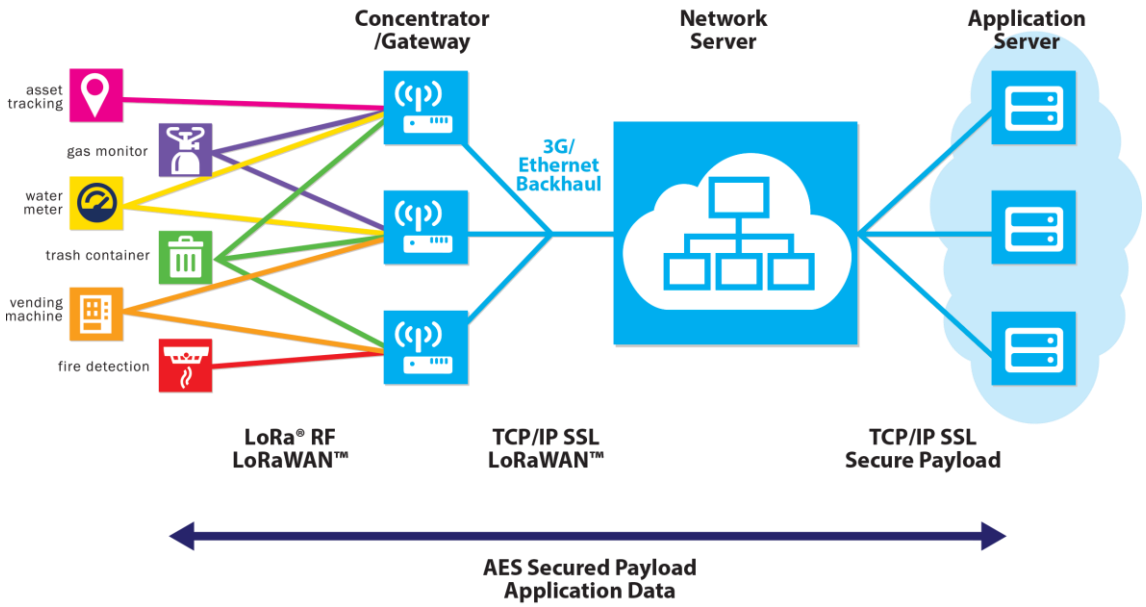


Fig. 23: Topología de red LoRaWAN (www.semtech.com)

La mayoría de redes LoRaWAN presentan una topología de doble estrella como la indicada en la imagen, ofreciendo comunicación bidireccional y multidireccional entre 868 y 870 MHz a velocidades entre 0,3 kpbs y 50 kbps. Permite hasta 84 mensajes *uplink* y 12 *downlink*, con un *payload* máximo de 256 bytes (B), dependiendo del proveedor de la red.

- En primer lugar tenemos los *End devices* o nodos finales, sensores de dimensiones reducidas e inalámbricos alimentados por baterías. A menudo envían la misma información a diferentes gateways.
- Los *gateways* o pasarelas se pueden considerar convertidores o puntos de acceso. Reciben la comunicación de los nodos finales y redirige la información al servidor de red.
- El *Network Server* maneja la deduplicación del mensaje, y lo reenvía al servidor de aplicación correspondiente.

La especificación LoRaWAN queda recogida por LoRa Alliance, una asociación creada en 2015, abierta y sin ánimo de lucro, con la misión de estandarizar el protocolo para garantizar la interoperabilidad entre sus productos y tecnologías, y certificar a cualquier fabricante de hardware que incorpore este protocolo. Actualmente tiene más de 500 miembros, entre ellos, grandes empresas del mundo de las telecomunicaciones como Cisco, IBM, Orange, Microchip, y por supuesto Semtech.

4.3.2.2. SigFox

Fig. 24: Logotipo de la compañía Sigfox

Es una solución LPWAN desarrollada en Francia en 2009 por la compañía homónima Sigfox. Se trata de un protocolo muy extendido: opera a nivel global, con cobertura en 75 países y prácticamente total en Europa.

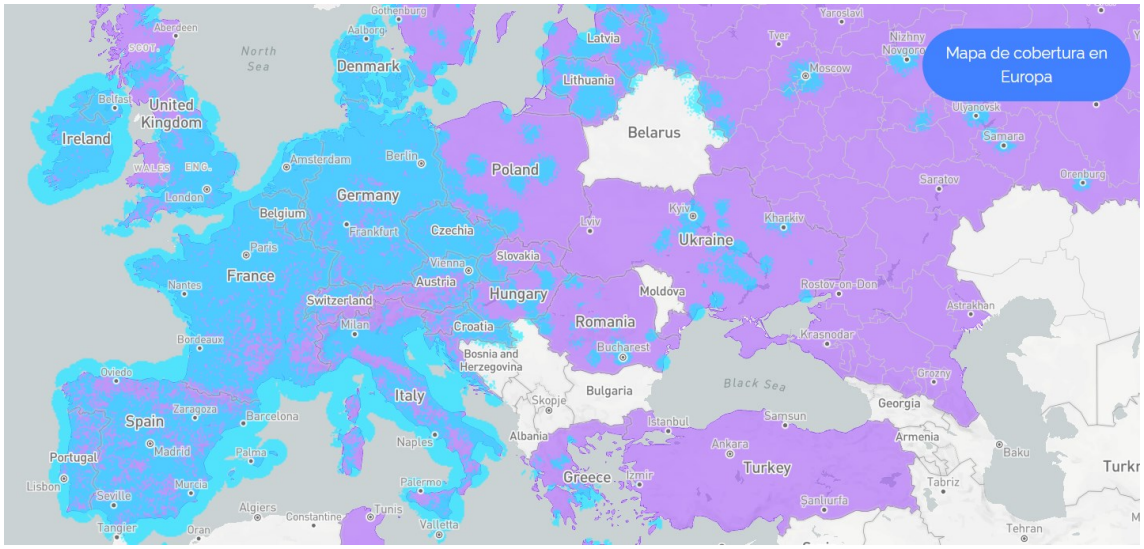


Fig. 25: Mapa de cobertura en Europa (www.sigfox.com)

Del mismo modo que LoRaWAN, opera en las bandas de frecuencia ISM, cuenta con encriptación AES128 y presenta una topología en estrella. La principal diferencia se encuentra en que tanto el hardware como el software es propietario, es decir, se trata de una **red de servicios privada**. De forma similar a otras compañías de telefonía móvil, Sigfox cobra una tarifa de suscripción (en torno a 2€ anuales por *End device*) para conectar nuestros dispositivos; a cambio, ofrece una infraestructura de telecomunicación independiente, basada en estaciones base: antenas locales de SigFox que reciben mensajes de los dispositivos emisores y los reenvían a la nube de dicha compañía. Están distribuidas estratégicamente para lograr la mayor cobertura, y escanean el espectro completo de 192 kHz en busca de señales a demodular.

Teóricamente, el alcance varía entre 30 y 50 km en ambientes rurales, y es de alrededor de 10 km en urbanos.

Especificaciones técnicas

La transmisión radio utiliza la tecnología UNB (*Ultra Narrow Band*) combinada con modulación DBPSK (*Differential Binary Phase Shift Keying*) para el u*plink*, y GFSK (*Gaussian Frequency Shift Keying*) para el *downlink*. Esto se debe a que el protocolo fue concebido como unidireccional, y tras su conversión a bidireccional, presenta una gran asimetría de enlace, con lo que el problema de diferencia entre *uplink* y *downlink*, ya presente en las redes LPWAN, se acentúa en el caso de Sigfox:

- El *uplink* permite un máximo de 140 mensajes diarios de 12 B de carga útil cada uno, con una velocidad fija de 100 bps.
- El *downlink* es de 4 mensajes de 8 B al día, a 600 bps.

Es decir, un mensaje cada 10 minutos. Conviene señalar que tanto tamaño de los mensajes como la velocidad son fijos.

4.3.2.3. NB-IoT

Fig. 26: Logotipo de la tecnología NB-IoT

NB-IoT (*Narrow Band IoT*) es la tecnología LPWAN más reciente; surgió en 2013, desarrollada por el consorcio 3GPP, pero su especificación no fue consolidada hasta 2016.

Es una especificación para servicios de alta gama enfocados a la cobertura de dispositivos móviles en espacios cerrados y altas densidades de conexión. Su protocolo de comunicación se basa en el de LTE, reduciendo su complejidad y funcionalidades para centrarse en las características de cobertura y eficiencia, por lo que se puede considerar un *subset* de dicho estándar.

De esta forma, puede desplegarse tanto en redes 2G (GSM) como 4G (LTE), haciendo uso de ondas portadoras GSM existentes, o a través de bandas de protección

libres entre canales de comunicación LTE, ocupando un BW de 180 kHz, equivalente a un bloque de recursos físicos.

La comunicación es bidireccional half-dúplex, y emplea la técnica de modulación QPSK (*Quadrature PSK*) para el *downlink* y BPSK o QPSK para el uplink. A diferencia de las dos soluciones anteriores, NB-IoT hace uso de **bandas licenciadas** como 700, 800 y 900 MHz, por lo que habrá que pagar una cuota y disponer de una tarjeta SIM (*Subscriber Identity Module*) o eSIM (*embedded SIM*).

A cambio, presenta mejores prestaciones en velocidad (200 kbps) y *payload* (1600 B), notablemente mayores al de los protocolos vistos previamente. No obstante, el *hardware* es más caro, su consumo es mayor, y la comunicación posee un alcance relativamente bajo, de 1 km en entornos urbanos y 10 km en rurales.

4.4. Elección de la comunicación por satélite

En los últimos años, ha surgido un modelo de negocio fruto del *NewSpace* (emergencia de la industria privada en el ámbito espacial) enfocado a ofrecer servicios de conexión a internet mediante comunicación satelital basada en el despliegue de nanosatélites. Según la NASA, "en términos de masa, un nanosatélite (*nanosat* o nano satélite) es aquel con un peso entre 1 y 10 kilos". Debido a su reducida masa y dimensiones, presentan una serie de ventajas respecto a los satélites artificiales convencionales, como precios más económicos y tiempos de desarrollo más cortos (menos de 8 meses), lo que lo hace accesible a empresas de todo tipo y tamaño.

Estos satélites se mueven en la órbita terrestre baja o LEO (*Low Earth Orbit*), situada entre los 160 y los 2.000 km de altura; más adelante comprobaremos que, dentro de esta amplia franja, la altura del nanosatélite no suele exceder los pocos cientos de kilómetros. A diferencia de los satélites geoestacionarios, los objetos situados en esta órbita se desplazan a gran velocidad respecto de la superficie terrestre, cubriendo una órbita completa en cuestión de horas, con lo que se logra una cobertura global casi constante. Además, las constelaciones de nanosatélites ofrecen redundancia y robustez, y se van renovando periódicamente para asegurar que la tecnología que emplean no queda obsoleta.

Como consecuencia, la vertiente satelital de la enorme infraestructura que sustenta el Internet de las Cosas ha experimentado un crecimiento explosivo. Tanto es así, que durante el periodo de redacción de la memoria no han dejado de surgir nuevas candidatas para el estudio de alternativas que se presenta a continuación.

En definitiva, las comunicaciones por satélite orientadas al IoT mediante su integración con redes LPWAN constituyen una tecnología incipiente, y es de esperar que sigan apareciendo soluciones capaces de satisfacer las necesidades del proyecto.

4.4.1. Lacuna Space

Fig. 27: Logotipo de la compañía Lacuna Space

Lacuna Space Ltd. es una *startup* inglesa establecida en 2017 en el campus de Harwell, en Oxfordshire, Reino Unido. Trabaja bajo la supervisión y el apoyo de la agencia espacial del Reino Unido y del programa ARTES (*Advanced Research in Telecommunications Systems*), un plan a gran escala y largo plazo de la Agencia Espacial Europea para promover el desarrollo de productos y servicios avanzados de comunicaciones satelitales.

A diferencia de sus competidores, Lacuna Space tiene por objetivo operar satélites con receptores lo suficientemente sensibles para captar señales emitidas directamente por sensores LoRa certificados, prescindiendo así de cualquier etapa intermedia, típicamente encargada de amplificar dicha señal y reenviarla al espacio.

De esta forma, los nodos transmiten su mensajes a un satélite en tránsito, que los almacena por un corto periodo de tiempo hasta pasar sobre una estación terrestre. Una vez llegan a la estación, la información se trasfiere automáticamente a internet, donde puede ser accedida a través de la aplicación web de Lacuna o reenviada a cualquier otra elegida por el usuario.

Para conseguirlo, Lacuna planea desplegar una constelación de 32 nanosatélites entorno a los 500 km de altura. Según el cofundador, Rob Spurret, dichos satélites contarán con propulsión y se basarán en el estándar de diseño CubeSat, que define un factor de forma escalable en cubos de 10 cm de arista y menos de 1,3 kg.

Por tanto, uno de los fines de esta compañía impulsar la tecnología LoRa y lograr que su protocolo alcance una cobertura global. El primer dispositivo funcional compatible con los chips de Semtech fue producido en 2020 (Fig. 28).

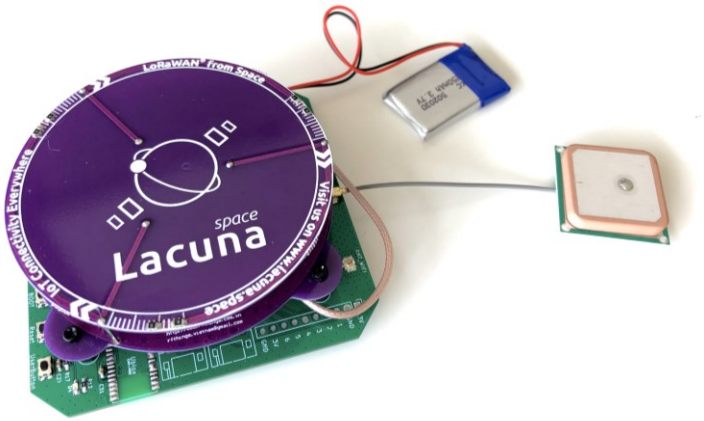


Fig. 28: Prototipo de Gateway con el transceptor LoRaEdge LR1110 de Semtech

Desde entonces, Lacuna ha enviado con éxito varios gateways al espacio y un satélite *CubeSat* 3U.

Fig. 29: Satélite CubeSat 3U con antena helicoidal

Los progresos realizados y su objetivo común han llevado a Semtech a involucrarse en el proyecto. Spurret espera que dicha cooperación, recientemente iniciada, les permita "comenzar los servicios comerciales en la primera mitad de 2022". No obstante, aunque prometedora, esta alternativa está todavía en proceso de investigación y desarrollo en el momento de redacción de la memoria, y no se encuentra comercialmente disponible.

4.4.2. Iridium

Fig. 30: Logotipo de la compañía Iridium®

Iridium Communications es una empresa comunicaciones por satélite, proveedora de servicios de voz en la Banda L y datos en todo el mundo. Afirma que "su arquitectura de constelación impar la convierte en la única red que cubre el 100 % del planeta".

Dicha constelación está formada por 6 órbitas de 11 satélites activos, junto a varios adicionales para suplir al resto en caso de fallo o avería. Se sitúan alrededor de los 780 km de altura, y se desplazan a unos 27 km/h describiendo una órbita polar con un inclinación de 86.4º. Además, la constelación posee enlaces cruzado de datos por satélite, o enlace *intersatélite* (ISL), también conocido como *crosslinking*. Permite que los satélites se conecten e intercambien información entre sí, lo que presenta una serie de ventajas de cobertura y robustez. En este caso, cada uno de los satélites puede enlazarse con 4 vecinos.

Propietaria y operadora de la constelación, Iridium Communications oferta servicios pero también equipos para acceder a los mismos, ya que diseña y manufactura gran variedad de productos IoT, que vende junto a los de sus socios comerciales. En su página web encontramos más de 200 referencias.

Entre ellas, algunos módulos transceptores diseñados para su integración en soluciones inalámbricas, como el 9602 o su versión mejorada, el 9603, cuatro veces más pequeño.

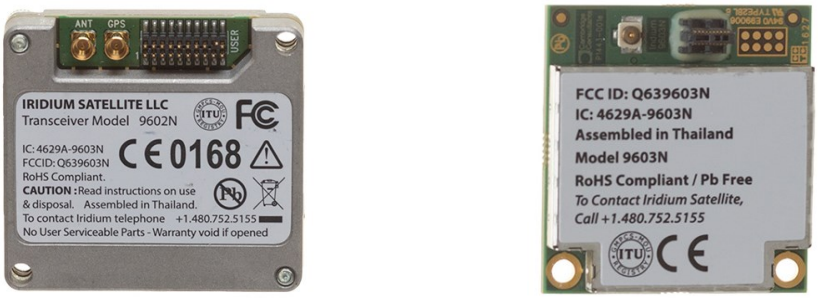


Fig. 31: Módulos 9602 (izqda.) y 9603 (dcha.)

Este último se encuentra comercialmente disponible, integrado el módem RockBLOCK 9603 de Rock Seven, que incluye una antena y satisface los requisitos de alimentación del transceptor de Iridium.

Fig. 32: Módem Rock7 RockBLOCK 9603 (www.adafruit.com)

No obstante, esta opción de ha descartado debido a los elevados costes de suscripción a los planes de datos, uno de los factores decisivos, como se ha explicado en el estudio de necesidades.

4.4.3. Swarm

Fig. 33: Logotipo de la compañía Swarm Technologies

Swarm Technologies Inc. es una compañía privada dedicada al sector de las telecomunicaciones por satélite, fundada en 2016 por la Dra. Sara Spangelo (actual CEO) y el Dr. Ben Longmier en Los Altos, California, con la finalidad de brindar conectividad global y accesible para toda clase de dispositivos IoT, especialmente en el ámbito de la industria agropecuaria y marítima.

Esta opción destaca por el bajo coste de suscripción; el plan de datos estándar consiste en un contrato anual de 5 USD mensuales, y permite enviar 750 mensajes de cada dispositivo cada mes, con una *payload* de hasta 192 B. No obstante, la cobertura no es constante; las horas de paso de los satélites deberán consultarse en una página oficial de la compañía: Swarm Pass Checker.

Dicha cobertura es proporcionada por una red de 150 picosatélites de 250 g y del menor tamaño del formato CubeSat: ¼ U, (11 × 11 × 2.8 cm) denominados **SpaceBEE**. Orbitan entre los 450 y 550 km de altura, siguiendo una órbita heliosíncrona, de modo que pasan sobre un determinado punto siempre al mismo tiempo solar local. Estos satélites ofrecen comunicación bidireccional y pueden comunicarse con las estaciones terrestres pero no entre sí, es decir, no cuentan con ISL.

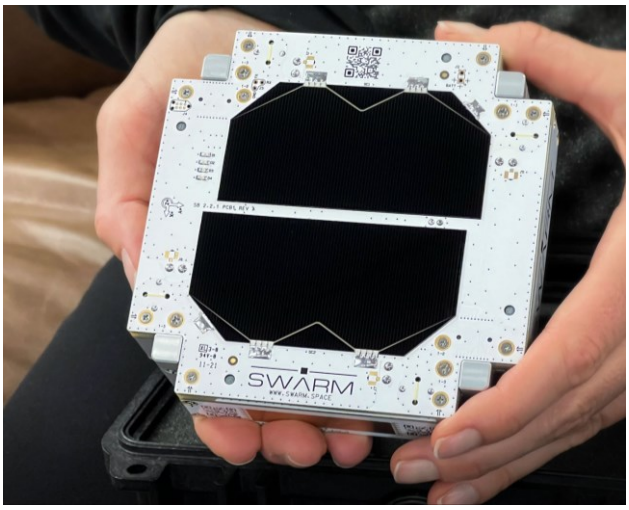


Fig. 34: Satélite SpaceBEE 0.25U (www.swarm.space)

Al igual que Iridium, Swarm cuenta con una amplia integración vertical, ya que diseña el software y fabrica todo el hardware necesario para establecer comunicación con sus satélites, y enviar mensajes a internet.

A través de su tienda *on-line*, se ofertan dos productos:

Módem M138

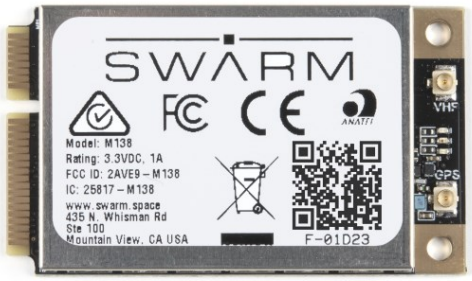


Fig. 35: Módem M138 (www.sparkfun.com)

Con un peso de menos de 10 g, el módem está diseñado para ser embebido en un dispositivo externo mediante su interfaz digital UART con tecnología CMOS y GPIO a 3,3 V.

- Proporciona comunicación bidireccional a 1 kbps en las frecuencias de 137-138 MHz para el *downlink* y 148-150 MHz para el *uplink*, es decir, trabaja en el espectro VHF (*Very High Frequency*)
- Está basado en un procesador ARM Cortex-M4
- Posee GPS integrado e incluye conectores UFL para antenas GPS y VHF

El proyecto se centra en la selección e integración de los distintos componentes, con un nivel de abstracción que permita centrarse principalmente en la cuestión de la conectividad. Además, la compra del Módem M138 solo se puede realizar en grupos de 25 Uds. Por estos motivos, se ha desechado esta opción.

El *kit* de evaluación – Swarm Eval Kit

Fig. 36: Swarm Eval Kit

Esta solución IoT sacrifica la comunicación bidireccional – prescindible en nuestro caso – permitiendo únicamente el envío de mensajes o la posición GPS. En cambio, el *kit* incluye todos los componentes necesarios para un despliegue inmediato:

- El propio modem M138
- Antena de ¼ de onda VHF
- Panel solar de 9W + 3 baterías recargables 18650
- Trípode con nivel de burbuja
- Placa ESP32 FeatherS2 para la conectividad inalámbrica
- Pantalla OLED FeatherWing de Adafruit

De modo que constituye un sistema completamente independiente.

El *kit* tiene una autonomía de 24 h sin el respaldo del panel solar, y presenta el grado máximo de protección frente al polvo y humedad: IP68.

Su adecuación a los requisitos expuestos en el estudio de necesidades ha llevado a seleccionar el **Swarm Eval Kit** para su utilización como *gateway* en el enclave.

5. Descripción detallada de la solución adoptada

Finalmente, se procede a exponer en detalle la solución a la problemática de este trabajo. A fin de ilustrar la ejecución del proyecto lo mejor posible, este apartado de la memoria se organizará siguiendo el diagrama de bloques de la Fig. 37, que presenta los distintos elementos hardware que conforman el sistema en su totalidad conforme a lo explicado anteriormente:

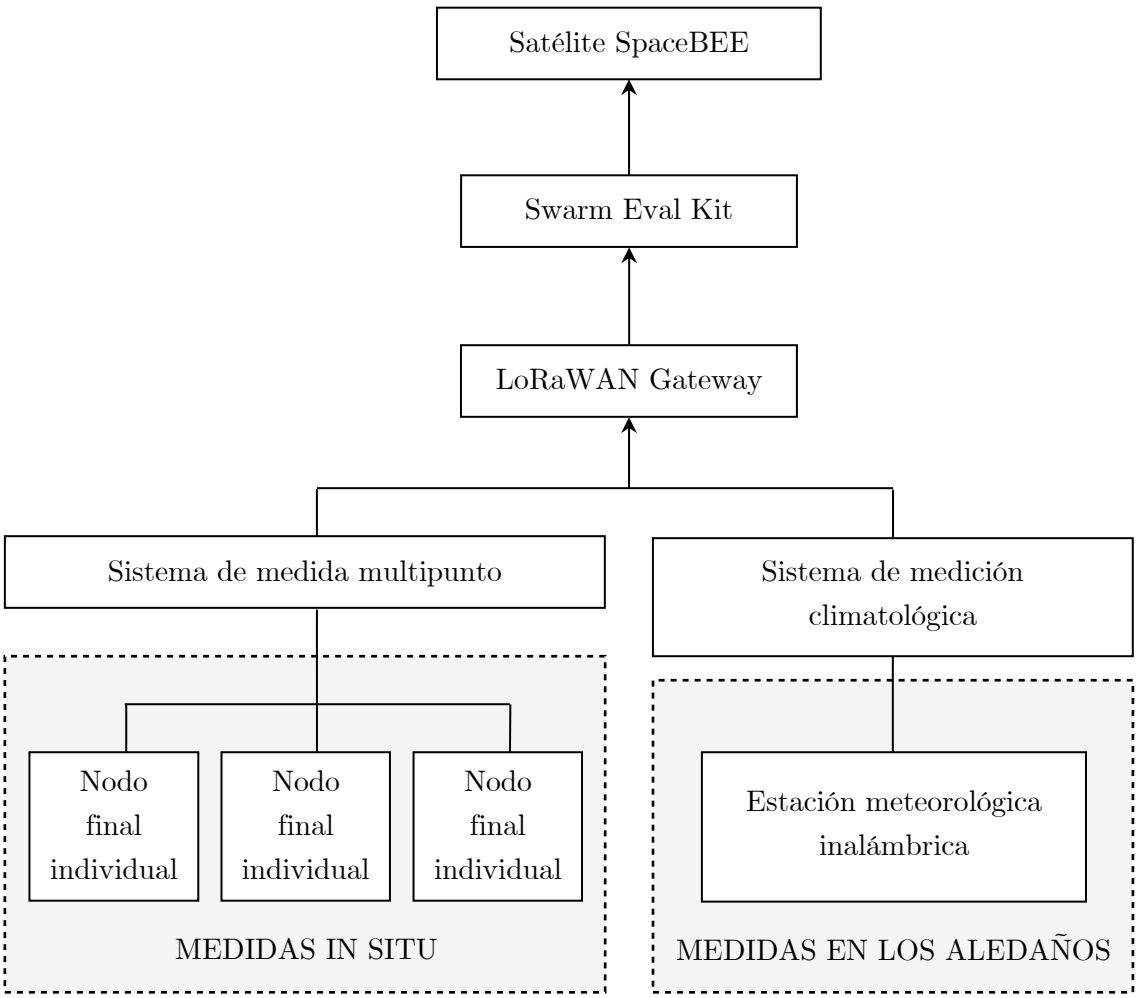


Fig. 37: Diagrama de bloques del sistema de monitorización

A continuación, a modo de introducción, se resume brevemente el funcionamiento del esquema:

Todos los nodos finales o *end devices*, ya sean individuales o integrados en una matriz de sensores en el caso de las estaciones, serán dispositivos LoRaWAN y se comunicarán inalámbricamente con el *gateway* de exterior empleando dicho protocolo. Como se explicó en el estudio de necesidades, no se prevé la disponibilidad de ningún tipo de cobertura telefónica ni conexión a internet (salvo la del *kit*), por lo que el sistema debe ser completamente autónomo. Para lograrlo, será utilizará un *gateway* cuyo software permita lanzar aplicaciones *out-of-the-box*, es decir, directamente y sin necesidad de ningún otro intermediario.

Por otra parte, la puerta de enlace no posee una forma de comunicarse directamente con el *kit* de Swarm, así que para enviar la lectura de los sensores al módem M138 se utilizará como puente un protocolo de telecomunicación digital ligero, específicamente orientado a las transmisión de datos M2M (*machine-to-machine)* entre dispositivos IoT. Además, para hacer un uso efectivo de dicha tecnología, será necesario combinarla con alguna implementación de *message broker* o bróker de mensajes, el *software* encargado de gestionar el flujo de datos y ponerlo a disposición del *kit*, que será el último eslabón de la cadena de comunicaciones de nuestro sistema en lo que a *hardware* se refiere.

Por último, el módem obtendrá los mensajes del bróker y los pondrá en cola para transmitirlos periódicamente a Swarm Hive, cuyo *BackEnd* permite emplear métodos Rest API (*Restful Application Programming Interface*) o *Webhooks* para entregar los datos a un servicio en la nube.

Ese servicio se encargará de comunicar los datos con la parte visible, que interactúa con los usuarios. Su función consistirá, por tanto, en solucionar todo lo relativo a la interfaz con el usuario; tipografía, efectos visuales, *widgets*, y otros elementos necesarios para la consulta de la información. En definitiva, el lado del cliente o *FrontEnd* de esta aplicación.

En él se implementará un panel de visualización o *dashboard* inteligente para el análisis y la monitorización de los datos en tiempo real. Debe ser compatible con el *message broker* y permitir crear gráficas personalizadas de forma sencilla.

Más adelante se detallará la selección de los componentes utilizados, así como los procesos e interacciones que llevan a cabo entre sí. Asimismo, se pretende abarcar los

pasos que permiten llevar a cabo la puesta en marcha y funcionamiento del sistema, quedando reservadas las instrucciones de uso y mantenimiento para el pliego de condiciones.

5.1. Red LoRaWAN

5.1.1. End Devices

Sistema de medida multipunto

Las medidas multipunto permitirán analizar posibles discrepancias en las mediciones efectuadas por los distintos nodos y detectar variaciones significativas de las magnitudes ambientales en el espacio, como gradientes de temperatura en la superficie – capaces de originar pequeñas corrientes de aire portadoras de polvo o microorganismos – u otras anomalías. Para obtener dicha información, se distribuirán varias unidades a lo largo de las paredes rocosas del abrigo.

Para el seguimiento de las condiciones microclimáticas en el entorno inmediato del bien cultural, se utilizará el sensor de **CollectionCare** (CC). Este nodo ha sido diseñado precisamente para satisfacer las necesidades de monitorización de artefactos culturales, por lo que se ajusta perfectamente a los requerimientos discutidos en el apartado 3.3

Fig. 38: Nodo sensor de CollectionCare

CollectionCare

COLL ECTION CARE

CollectionCare es un proyecto Europeo coordinado por la Universitat Politècnica de València destinado al desarrollo de un sistema de conservación preventiva para el cuidado de artefactos culturales individuales durante su transporte, almacenaje, y exhibición en escenarios de todo tipo.

Sistema de medición climatológica

Como se ha explicado, es preciso que las estaciones meteorológicas dispongan de un mecanismo para conectarse a la red LoRaWAN. Para ello, deberán incorporar un módulo transceptor inalámbrico de Semtech® – como los de la serie SX127X (SX1276/77/78/79) – y un microcontrolador que integre la comunicación inalámbrica, como el ESP32.

Es inusual encontrar sistemas de medición comerciales con el nivel de complejidad de una estación climatológica que incorporen la tecnología LoRa de forma nativa. No obstante, existen varios fabricantes del equipamiento o OEM (*Original Equipment Manufacturer*), que, como clientes del fabricante del diseño original o ODM (*Original Design Manufacturer*), realizan modificaciones sobre el producto, incorporando el *hardware* necesario para convertirlo en *end device* compatible con LoRaWAN.

Para el presente proyecto, se han planteado las siguientes alternativas:

- CER-ST-01 CERER – Axatel
- Wireless Weather Station Set – Barani Design
- IC-SNiP-AWS5-SL – Instrument Choice
- LoRaWAN Weather Station – Auroras.S.R.L
- LoRaWAN Base Model WS100LRW – UBIQ-IoT

Tras contactar con los distintos fabricantes y recabar la información pertinente, se ha seleccionado el modelo **WS100LRW** atendiendo a razones económicas y de disponibilidad.

Fig. 39: Estación meteorológica WS100LRW de UBIQ-IoT

Se trata de una versión derivada de la Davis Vantage Pro2 – cuyas especificaciones pueden consultarse en la Tabla 7 – gobernada por un *chipset* ESP32, con una autonomía no inferior a los 2 años a una transmisión por hora (requiere una batería SAFT LSH20 3.6V [Li-SOCl2] que habrá de adquirirse por separado), y un rango de hasta 20 km. Se instalará una unidad en las inmediaciones del abrigo.

5.1.2. Pasarela LoRaWAN

Continuando la similitud con el modelo OSI, una puerta de enlace, convertidor, pasarela… se trata de un dispositivo capaz de realizar las funciones de un *router* (capa 3 o capa de red) y de un *gateway* (capa 7 o capa de aplicación) simultáneamente, interconectando redes diferentes entre sí y controlando el tráfico de datos entre ellas. Enruta los paquetes IP entre redes, y proporciona acceso a aquellos paquetes que se encuentran dentro o fuera de la red local. En definitiva, es un nodo de red que actúa como punto de entrada a otra red empleando diversos métodos de conectividad (Ethernet, Wi-Fi, celular…).

En el sistema del proyecto, el *gateway* es el elemento central de interconexión, y un punto de parada clave para los datos en su camino desde la red LoRaWAN hacia el resto del sistema.

Lógicamente, el coste del *gateway* dependerá de la cantidad de nodos que pueda manejar y por ende, del número de canales. No obstante, este despliegue exige que una única puerta de enlace pueda proporcionar cobertura LoRaWAN a diversos *end devices* en un radio de varios kilómetros, por lo que se empleará una pasarela de 16 canales. Además, deberá presentar un elevado grado de protección frente a la humedad y al polvo dado que se hallará a la intemperie en todo momento. Encontraremos este nivel de aislamiento en un *gateway* de tipo *outdoor* o de exterior.

La pasarela LoRaWAN que se utilizará en el enclave será el **RAK7289 WisGate Edge Pro** del fabricante RAKWireless®. Los modelos de RAK son los indicados para esta aplicación, ya que incorporan la funcionalidad de servidor de red. Al no necesitar ningún otro servicio externo, es posible crear una aplicación *standalone* dentro del propio *gateway*, que gestione la red LoRaWAN de forma autónoma. En otras palabras, el *gateway* desempeñará la función de LNS (*Lora Network Server*). Además, ese mismo servidor de red integrado también permite al *gateway* hacer de bróker de mensajes MQTT (*Message Queuing Telemetry Transport*), solventando dos niveles de la comunicación simultáneamente, y facilitando notablemente la labor de integración.

5.1.3. Conexión de los nodos y configuración del gateway

A continuación se detalla paso a paso el proceso de configuración del *gateway*.

Es posible acceder al *gateway* mediante Ethernet o por Wi-Fi; la única diferencia se encuentra en la dirección IP a introducir en el navegador. Dado que la pasarela viene configurada por defecto en modo de punto de acceso o AP (*Access Point*) Wi-Fi, y que este método no requiere ningún hardware adicional, es el que se ha utilizado para llevar a cabo la configuración.

En la lista de redes Wi-Fi disponibles, encontraremos un ESSID (*Extended Service Set Identifier*) llamado RAK7289_XXXX, donde el valor de XXXX corresponde a los dos últimos bytes de la dirección MAC (*Media Access Control*) del *gateway*. Se trata de una red abierta, y no requiere ninguna contraseña para establecer conexión.

Para acceder a la página de inicio de sesión, abrimos la IPv4 192.168.230.1 en cualquier navegador, e introducimos las credenciales predeterminadas:

- Usuario: *root*
- Contraseña: *root*

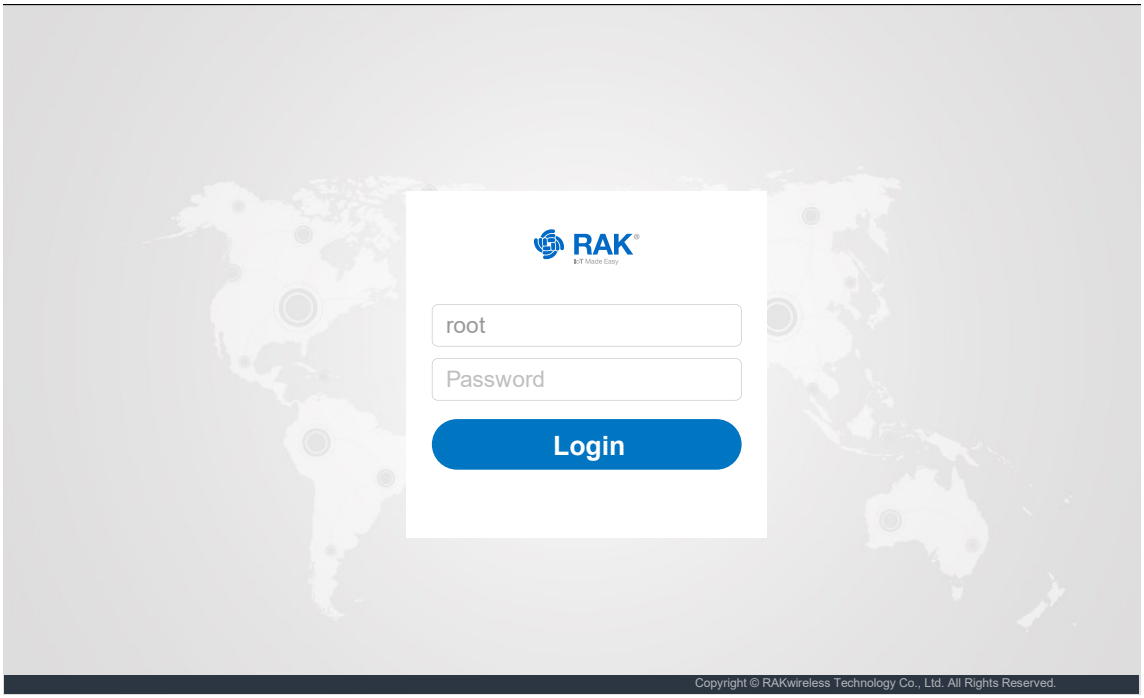


Fig. 40: Página de inicio de sesión del gateway

Lo que nos llevará a la interfaz gráfica de usuario del *firmware* instalado en el *gateway*, basado en OpenWRT. Aquí encontramos un menú principal, donde tenemos acceso a las distintas opciones de configuración, y un panel de visualización con los

datos más relevantes de la comunicación, que resultará de utilidad a la hora de comprobar el funcionamiento de los nodos.

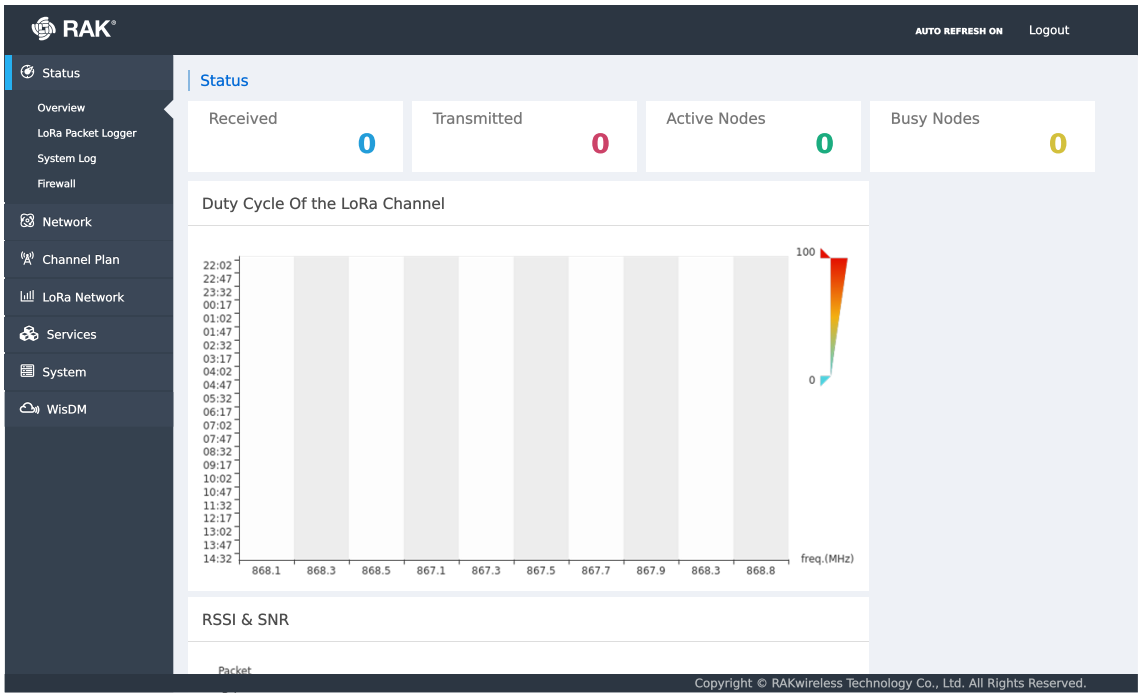


Fig. 41: Sitio web del gateway

En primer lugar, es necesario asegurarse de que la banda de frecuencia que utiliza la transmisión es la adecuada. Para ello, en la sección "Channel Plan" se selecciona el margen de frecuencias correspondiente a la región de Europa: 863 – 870 MHz

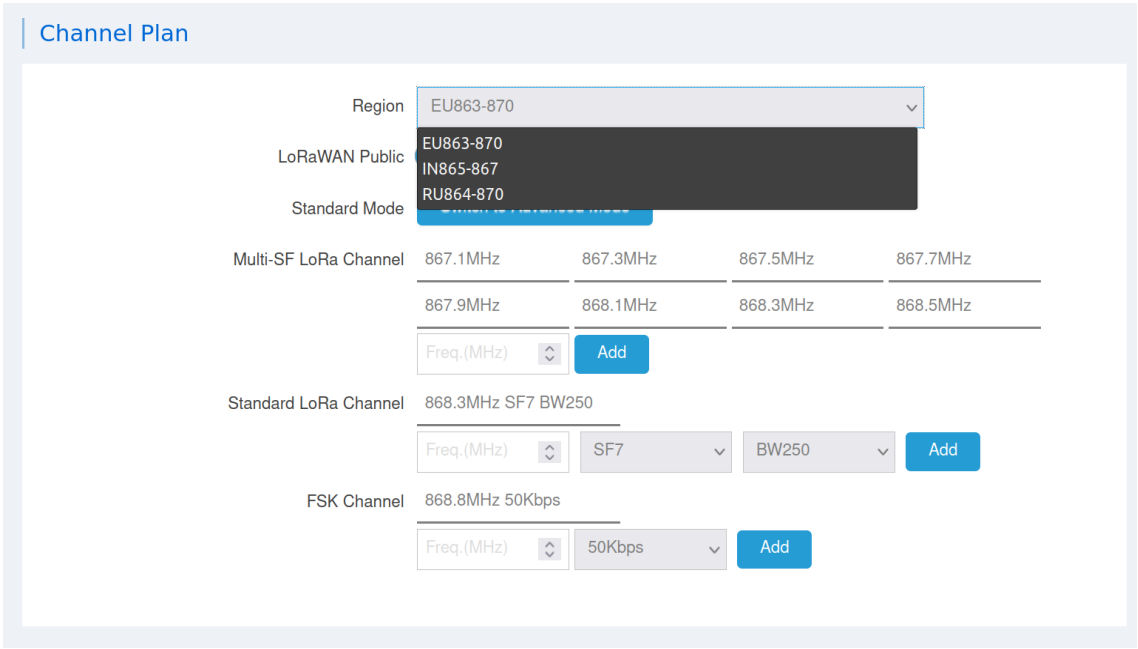


Fig. 42: Selección de la banda de frecuencia de transmisión del gateway

A continuación, en el apartado de ajustes de la red LoRaWAN, se establece el modo de funcionamiento. Este gateway presenta 3:

- *Packet Forwarder*
- *Basic Station*
- *Network server*

De nuevo, el necesario para esta aplicación es el de **servidor de red**.

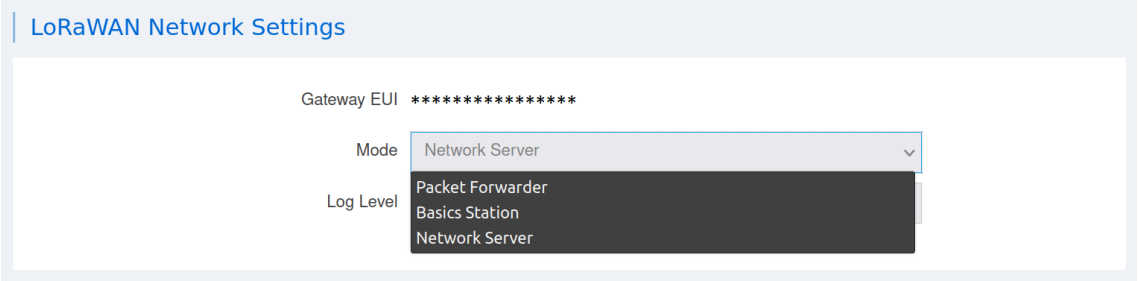


Fig. 43: Selección del modo de funcionamiento en los Ajustes de Red LoRaWAN

En la zona inferior encontramos una serie de ajustes del LNS interno, que permiten llevar a cabo una configuración avanzada. Teniendo en cuenta que, una vez instalados, los end nodes permanecerán fijos en su ubicación, **activaremos el ADR** (*Adaptative Data Rate*) con el fin de optimizar la transmisión automáticamente en la zona de despliegue, mediante el ajuste de una serie de parámetros (SF, BW, potencia de transmisión…) directamente relacionados con el consumo energético y el *data rate.* El resto de opciones se dejarán configuradas por defecto, incluidas las relativas al *beacon* del gateway.

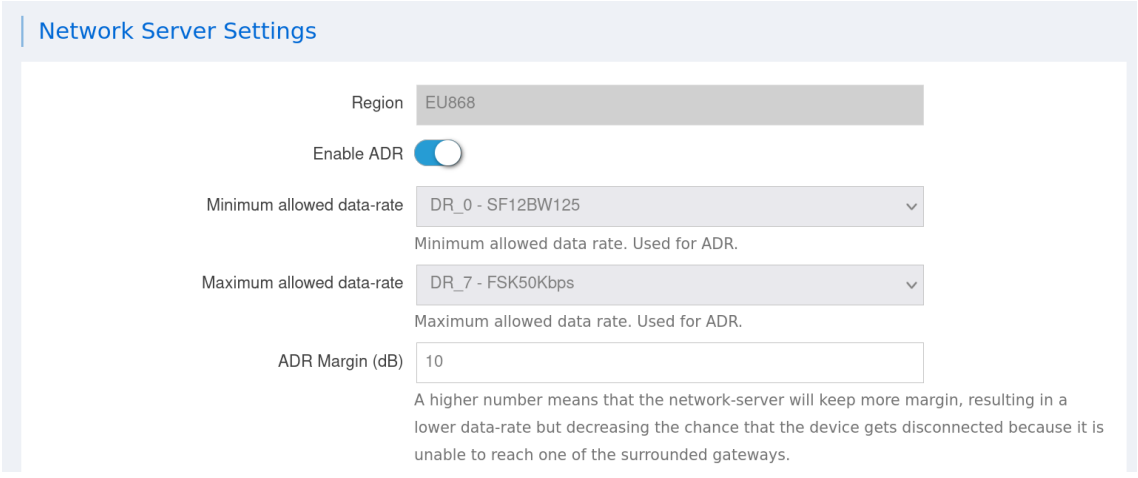


Fig. 44: Ajustes del servidor de red integrado

Finalmente podemos comenzar con la creación de la aplicación. Para ello abrimos la sección “Application”, donde aparece el listado de aplicaciones creadas, y la opción

de añadir una nueva. Seleccionando esta opción pasamos a la configuración de la aplicación.

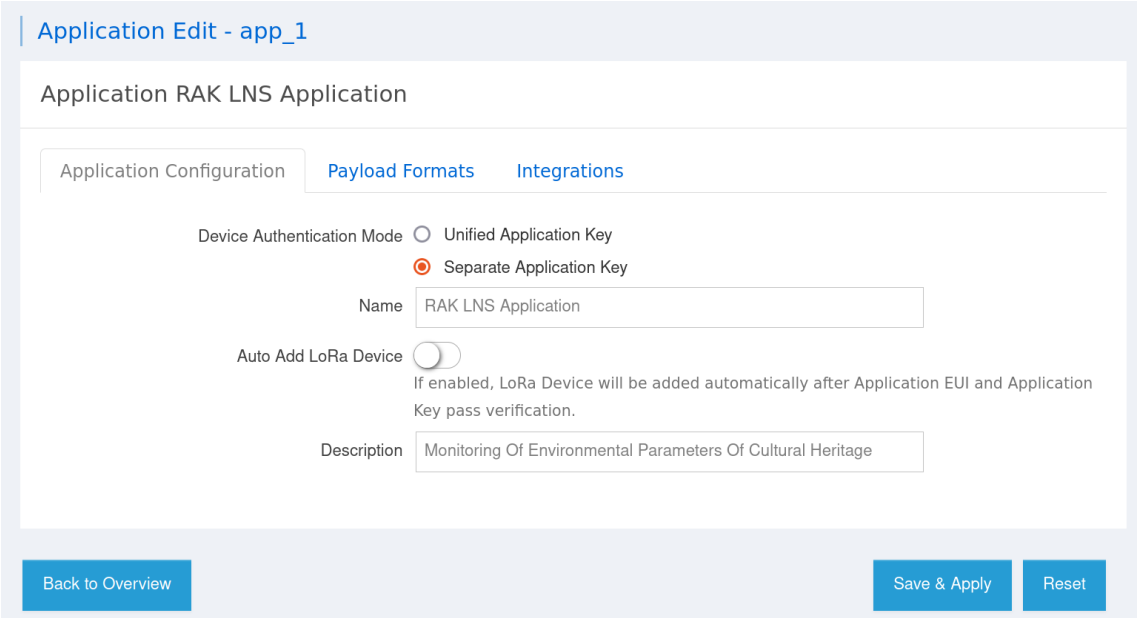


Fig. 45: Configuración de la aplicación

La *Application Key* o AppKey es un una clave AES128 se utiliza para derivar las claves de sesión para la encriptación, y puede ser específica al *end node* o no, como vemos a continuación.

Por defecto, viene seleccionada la modalidad "Unified Application Key", que ofrece la posibilidad de generar aleatoriamente un único AppKey para la aplicación, permitiendo que todos los *end devices* compartan la misma clave.

Este método resulta conveniente para despliegues de flotas de decenas o centenas de sensores, típicos de entornos industriales o museos. No obstante, en este caso, el número de nodos a manejar es relativamente bajo, por lo que se optará por la alternativa de claves individuales. De esta manera, se procede a añadir cada uno de los dispositivos, indicando sus especificaciones técnicas, como muestra la Fig. 46

Los nodos a utilizar serán de clase A, la implementación más soportada en los dispositivos LoRaWAN, y que ofrece un mayor ahorro energético por hacer un uso económico de las ventanas de recepción, ya que entra en modo escucha únicamente en dos ocasiones tras realizar la transmisión. Se conectarán mediante OTAA (*Over The Air Activation*). A diferencia del ABP (*Authentication By Personalization*), que emplea

claves estáticas, los dispositivos OTAA renegocian los *framecounters* y las claves de sesión al establecer cada nueva sesión, por lo que son considerados más seguros.

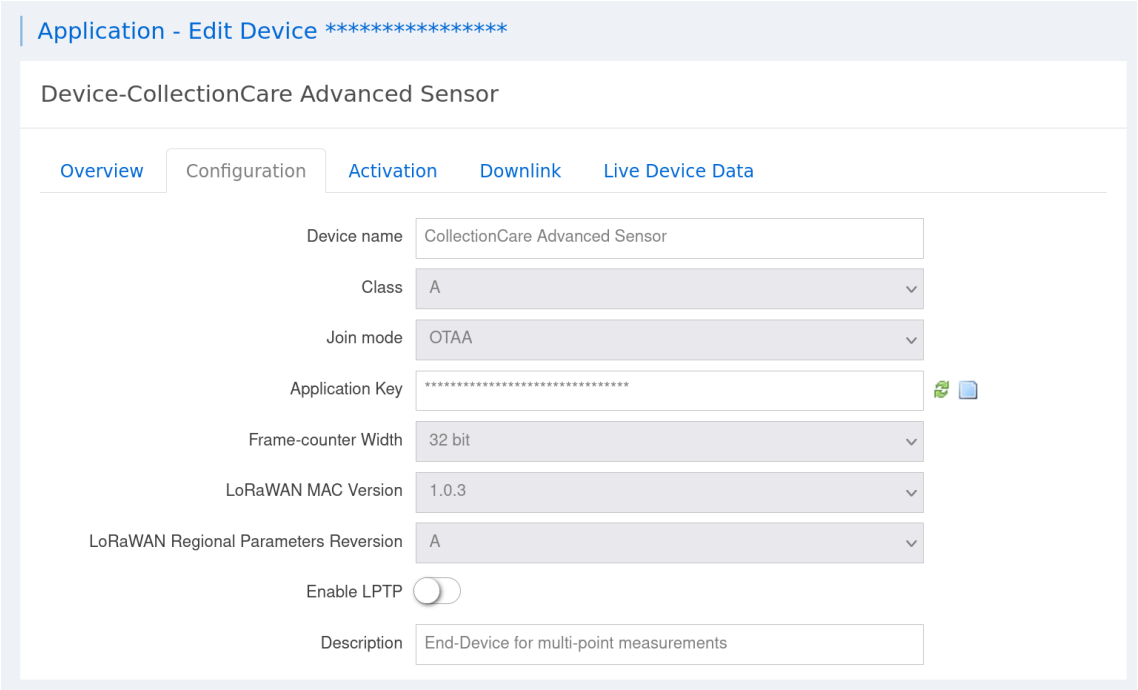
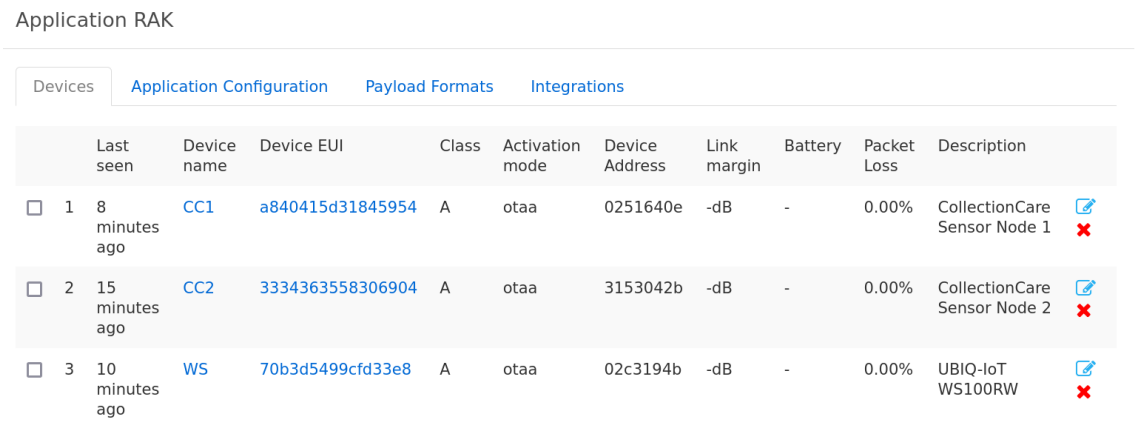


Fig. 46: Configuración del End device

El procedimiento para registrar el resto de nodos es idéntico, a excepción del correspondiente a la estación meteorológica, que se podrá consultar en las hojas de características del fabricante, adjuntas en el documento de anexos. Antes de proceder al siguiente paso, se comprueba la actividad de la transmisión:



	Last seen	Device name	Device EUI	Class	Activation mode	Device Address	Link margin	Battery	Packet Loss	Description
1	8 minutes ago	CC1	a840415d31845954	A	otaa	0251640e	-dB	-	0.00%	CollectionCare Sensor Node 1
2	15 minutes ago	CC2	3334363558306904	A	otaa	3153042b	-dB	-	0.00%	CollectionCare Sensor Node 2
3	10 minutes ago	WS	70b3d5499cfd33e8	A	otaa	02c3194b	-dB	-	0.00%	UBIQ-IoT WS100RW

Fig. 47: Lista de dispositivos registrados en la aplicación LoRaWAN

Y el estado global de la aplicación:

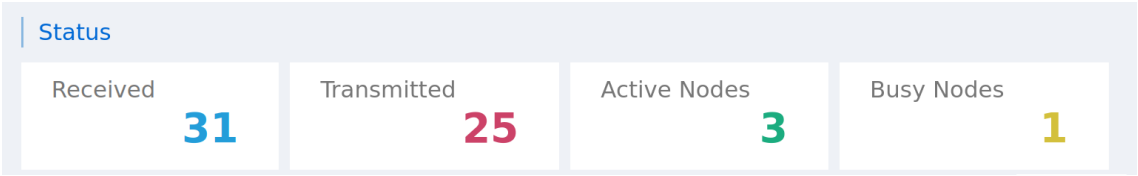


Fig. 48: Información general de la aplicación

5.2. Conexión mediante MQTT

Fig. 49: Logotipo del protocolo MQTT

A continuación se realizará toda la configuración concerniente a la interconexión de los dispositivos utilizando el **protocolo MQTT**, una especificación de red M2M liviana, enfocada a la transmisión de datos a nivel de byte, y basada en el mecanismo asíncrono de publicación/suscripción (*pub/sub model*) a determinados "tópicos", cadenas de texto en formato UTF-8 que el bróker utiliza para filtrar los mensajes para cada uno de los clientes conectados.

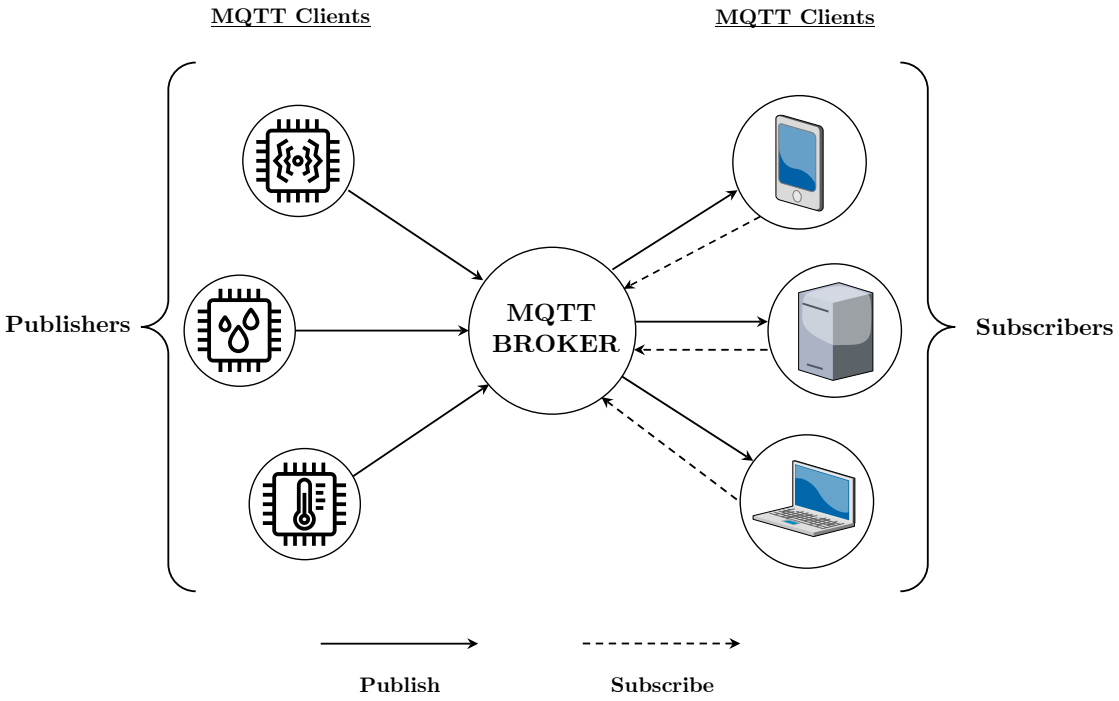


Fig. 50: Arquitectura MQTT (Imagen propia. Gráfico vectorial ampliable)

Por lo tanto, el *message broker* posee un papel central en la red MQTT, pues se trata del agente que gestiona las publicaciones y suscripciones, y es también el *software* que permite a los clientes (sistemas, aplicaciones, servicios...) comunicarse entre sí través de una conexión full-dúplex, como sugiere la Fig. 50.

5.2.1. Configuración de bróker MQTT

En primer lugar, abriremos la configuración "Global Integration" y seleccionamos el modo de integración de la aplicación: MQTT Genérico.

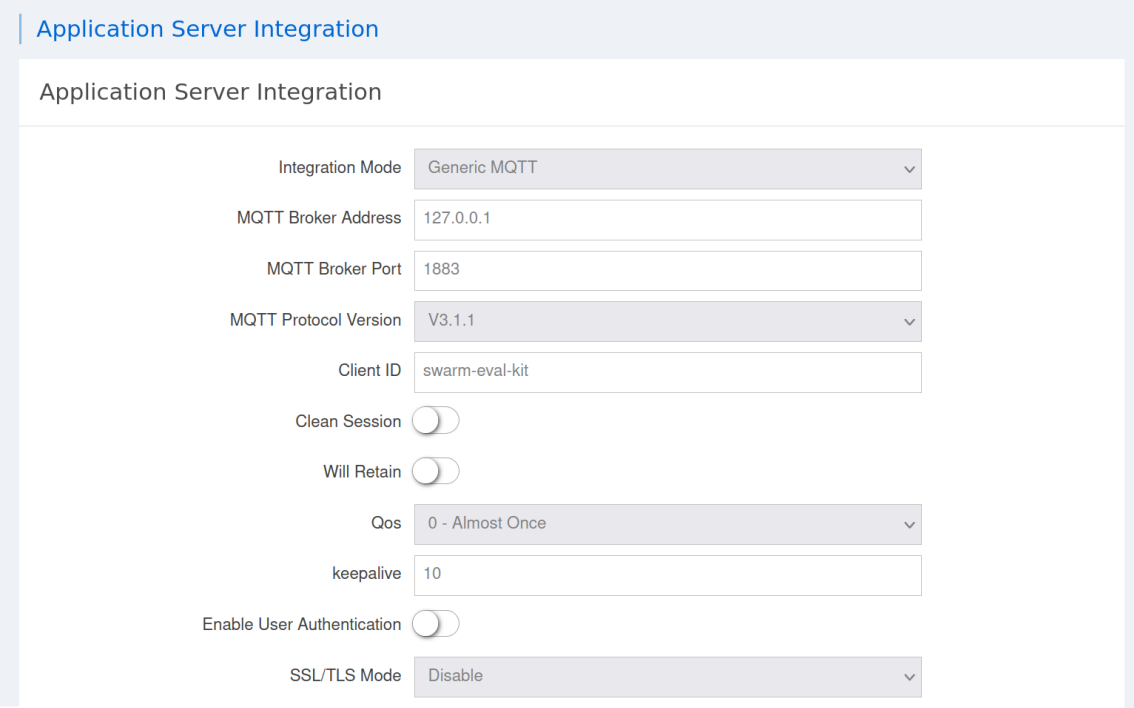


Fig. 51: Configuración de la integración de la aplicación con MQTT

Como se aprecia en la Fig. 51, el gateway permite conectarse a un bróker externo; existen varias implementaciones de bróker, y normalmente se elige en función de la aplicación (de microservicios, orientadas a eventos, de ingeniería de datos...), pero como se ha explicado, es imperativo que el sistema sea completamente independiente, por lo que hará las veces de bróker y de cliente.

Por ello, se dejará la dirección del *localhost* en la red local: 127.0.0.1. Se trata de una IP *loopback*, lo que significa que el *host* accederá a sus propios servicios independientemente de la configuración de red en que se encuentre.

El standard IANA (*Internet Assigned Numbers Authority*) asigna a MQTT el puerto TCP 1883, así que nuevamente se mantendrá el ajuste preestablecido.

Además, MQTT ofrece 3 niveles de calidad de servicio o QoS (*Quality Of Service*), aunque para ello, de forma análoga a HTTP, debe funcionar sobre un protocolo de transporte (Capa 4 del modelo OSI) que asegure la comunicación bidireccional y una transmisión ordenada y sin pérdidas de los mensajes, como en este caso TCP/IP.

Los clientes MQTT pueden permanecer conectados indefinidamente aunque no publiquen ni reciban ningún mensaje, pero el bróker necesita hacer un seguimiento de los clientes que siguen conectados para saber cuándo enviarles el mensaje de última voluntad o LWT (*Last Will & Testament*). Para ello se utiliza el tiempo de *KeepAlive* (KA).

Durante este periodo, el cliente puede permanecer completamente inactivo, mientras sigue conectado desde el punto de vista del bróker. Transcurrido este tiempo, para evitar que los clientes con un *data rate* bajo se desconecten de forma inadvertida, enviarán un paquete `PINGREQ` que el bróker deberá responder con el correspondiente `PINGRESP`, reiniciando el *timer* KA a 0. Es posible modificar el intervalo KA para por ejemplo cambiarlo a 0, desactivando dicho mecanismo si fuese necesario.

Por último, dado que la magnitud del *payload* supone un aspecto crítico, se desactiva la información de transmisión LoRaWAN (rxInfo /txInfo) para acortar la longitud del mensaje.

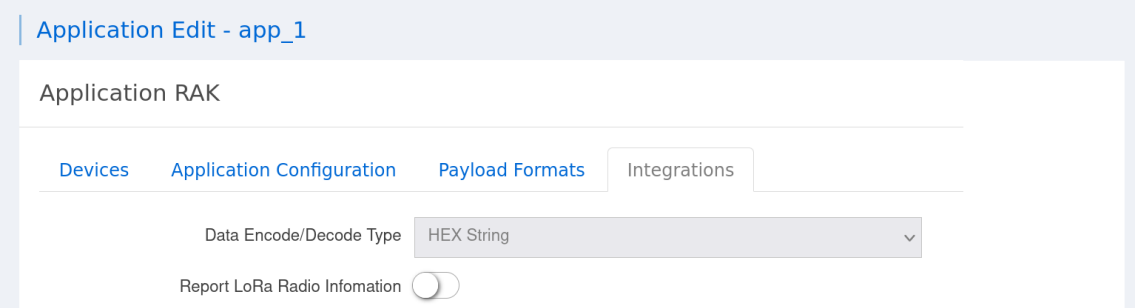


Fig. 52: Desactivación de la información de transmisión LoRaWAN

El bróker organiza los tópicos siguiendo la sintaxis de MQTT, en que cada nivel está separado por una barra de la siguiente forma:

Join Topic	`application/{{application_ID}}/device/{{device_EUI}}/join`
Uplink Topic	`application/{{application_ID}}/device/{{device_EUI}}/rx`
Downlink Topic	`application/{{application_ID}}/device/{{device_EUI}}/tx`
Ack Topic	`application/{{application_ID}}/device/{{device_EUI}}/ack`

Status Topic	`application/{{application_ID}}/device/{{device_EUI}}/status`

Tabla 8: Lista de tópicos del bróker MQTT del gateway

Para suscribirse a **todos** los nodos de la aplicación, utilizaremos el siguiente tópico: `application/{{application_ID}}/device/+/rx`

5.2.2. Programación del FeatherS2 ESP32. Configuración del kit

Finalmente se va a implementar una conexión puente o *bridge* para enlazar el broker de MQTT con un cliente externo: el kit de evaluación de Swarm.

El *hardware* del kit no incorpora soporte para MQTT de forma nativa, así que para efectuar dicha conexión será necesario modificar los archivos de código fuente guardados en la memoria flash del MCU: un Adafruit FeatherS2 ESP32.

Esta placa de desarrollo viene con **CircuitPython 7.0.0 Alpha 3** instalado, una versión derivada de MicroPython – a su vez basado en el lenguaje de *scripting* Python – creada por Adafruit para facilitar la labor de programación de sus placas. Presenta algunas peculiaridades que deberán considerarse a la hora de abordar la programación, en especial en cuanto al orden de ejecución de los archivos, a la compartición de estados entre los mismos, y a la nomenclatura de sus nombres, a saber:

- A diferencia de MicroPython, CircuitPython no permite que varios ficheros se ejecuten simultáneamente y compartan el mismo estado, circunstancia propia del multiprocesamiento.
- Por ello, el cuerpo del código principal con el correspondiente *superloop* debe almacenarse enteramente en code.py
- Los ficheros de configuración deben denominarse boot.py o settings.py, y solo se ejecutarán una vez durante el arranque de la placa.

Además, CircuitPython tiene *auto-reload* activado por defecto. Se desactivará para evitar que entorpezca la programación: `supervisor.disable_autoreload()`

Por otra parte, el FeatherS2 debe permanecer conectado en todo momento a la PCB del kit durante la ejecución del código para verificar su correcto funcionamiento. De lo contrario, no detectará los componentes electrónicos que inicializa durante el arranque – el sensor de tensión / corriente INA3221 y el módem M138 – y el programa se detendrá.

No obstante, solo es posible acceder a la memoria Flash mediante una conexión serie a través del puerto USB-C, únicamente accesible al separar la board de kit, lo que significa que no es posible probar el código directamente; en lugar de ello, sería necesario conectar y desconectar constantemente el FeatherS2 de su zócalo, algo nada recomendable ya que puede terminar cogiendo holgura.

Este detalle ha supuesto la principal dificultad en este apartado del sistema. Para afrontar este obstáculo, y con la idea de minimizar la manipulación de la placa, se ha partido de un "prototipo" sencillo para conectarse al bróker de MQTT sin necesidad de ningún hardware adicional, empleando las librerías de Adafruit `adafruit_minimqtt.py` y `matcher.py` disponibles en el repositorio de GitHub de la compañía: https://github.com/adafruit/Adafruit_CircuitPython_MiniMQTT/

Estos módulos de MQTT para CircuitPython permiten al ESP32 conectarse a un bróker de forma directa y sencilla mediante una serie de funciones preestablecidas:

- `mqtt_client.on_connect`
- `mqtt_client.on_disconnect`
- `mqtt_client.on_message`
- `mqtt_client.loop()`
- `mqtt_client.connect()`

Se implementan los cuerpos de las funciones de forma que proporcionen un *output* por el monitor serie. Empleando una implementación de cliente genérico MQTT (Fig. 53) como MQTT Explorer, simulamos un *uplink*, y a través de un terminal virtual como PuTTY o Tera Term observamos el resultado (Fig. 54)

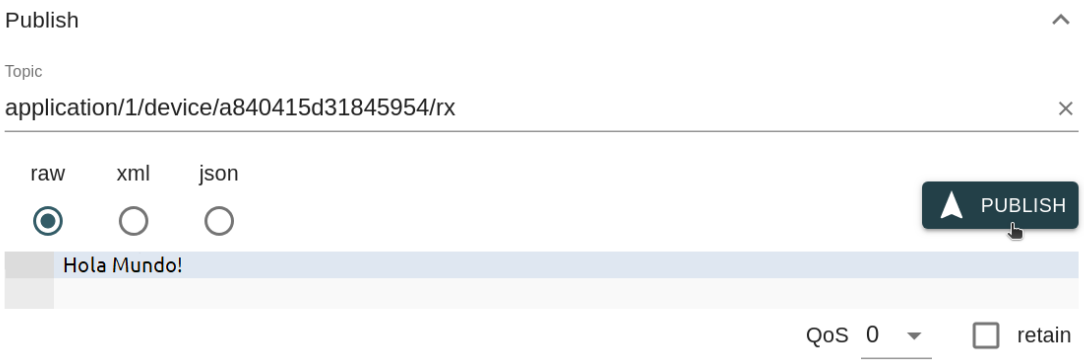


Fig. 53: Publicación al tópico de uplink desde MQTT Explorer

```
Resetting…

WIFI CONNECTION
========================================================================

Connecting via Wi-Fi to SSID: RAK7268C_9494…
Connected to: RAK7268C_9494!

MQTT CONNECTION
========================================================================

Connecting to MQTT Broker 192.168.230.1 on port 1883…
Connected to MQTT Broker!

MQTT LOOP
========================================================================

Listening for topic changes on application/1/device/+/rx

New message on topic application/1/device/a840415d31845954/rx:

Hola Mundo!
```

Fig. 54: Salida de la función por el terminal de Tera Term a través de conexión serial con el puerto COM

El siguiente paso consiste en integrar la conexión a MQTT en las funciones predefinidas del kit. Su funcionalidad Wi-Fi presenta dos configuraciones:

- Access Point Mode (AP)

En este modo, es posible conectarse al punto de acceso del kit y comunicarse con la dirección 192.168.4.1 con las credenciales indicadas en el manual para poner mensajes en cola y enviar otros comandos. Es la configuración predeterminada.

- Station Mode (STA)

En el modo STA, el usuario puede especificar a qué red conectar el kit. Una vez establecida dicha conexión, puede comunicarse con el kit utilizando la nueva dirección IP asignada, que aparecerá en pantalla.

Esta es la configuración en que se hallará en el despliegue final. El kit se conectará al AP del gateway, que es también la dirección del bróker MQTT, y se suscribirá al tópico de *uplink* de la aplicación.

Se configura el kit utilizando los comandos `@set` del Feather:

```
@set wifi enabled
Successfully enabled wifi.
Resetting…
@set mode sta
Successfully set mode to sta.
@set ssid RAK_7268C_9494
Successfully set ssid to RAK7268C_9494.
@set pw
Successfully set password to .
```

Fig. 55: Configuración del kit mediante comandos @set

Por otra parte, las especificaciones del bróker como la dirección IP y el puerto, se incluirán en un fichero aparte, denominado `secrets.py`, ubicado en el directorio raíz de la unidad `CIRCUITPY`, correspondiente al Feather.

Como se observa en la Fig. 56, el kit configurado como STA entra en un bucle para escuchar nuevos mensajes del bróker y los muestra en el display, mientras conserva el resto de funcionalidades originales.

Fig. 56: Pantalla OLED del FeatherS2 configurado como cliente MQTT

5.3. Conexión a Swarm Hive. Envío del mensaje.

5.3.1.1. NMEA Checksum Calculator

El Feather se ha conectado a MQTT con éxito, pero los mensajes no se almacenan de ninguna forma ni son accesibles desde internet. Para solucionarlo, el microcontrolador deberá **poner en cola** dichos mensajes para transmitirlos desde el módem M138 a Swarm Hive, la página web de Swarm que actúa como interfaz de usuario, y donde se puede consultar el contenido de los mensajes.

Para ello, es necesario tener en cuenta que *tile* utiliza una comunicación de tipo **NMEA** (*National Marine Electronics Association*) – una especificación vastamente presente en la instrumentación electrónica marina, como la ecosonda, los sonares, el anemómetro, el girocompás, el piloto automático, o los receptores GPS entre otros – lo que significa que todos los mensajes recibidos y enviados serán sentencias con el formato NMEA. Su sintaxis se resume a continuación:

- Las sentencias terminan con un solo carácter de nueva línea **\n**.
- Cada sentencia comienza con un **$** y termina con ***xx**, donde **xx** es un hexadecimal de dos dígitos con el *checksum* de los caracteres contenidos entre los delimitadores.
- El carácter **$** nunca podrá situarse dentro del comando, sólo puede aparecer como delimitador.
- Un ***** sí puede aparecer dentro del comando; la máquina verificará que los tres últimos caracteres son ***xx** tras recibir el **\n**, y antes de calcular el *checksum*.
- Cada **x** puede ser cualquier carácter ASCII en el rango 0..9, A..F o a..f.

El *checksum* es una combinación de caracteres que se utiliza para detectar errores surgidos durante la transmisión y verificar la integridad de los datos. Se calcula con la operación bit a bit XOR de todos los bytes exceptuando los delimitadores.

Dicho esto, se edita la función `message()` para incluir el encabezado necesario: `$TD` (*Transmit Data*) y la suma de comprobación.

```
def message(client, topic, message):
  displayLine(5, message)
  h = b'$TD ' + message
  cs = 0
  for c in h[1:]:
    cs = cs ^ c
  h = h + b'*%02X\n'%cs
  tile.write(h)
```

Fig. 57: Función message() modificada para dar el formato NMEA al mensaje

5.3.1.2. Payload Formatter

Con la modificación anterior, se logra habilitar el kit para transmitir mensajes publicados por el bróker.

Sin embargo, hasta ahora se ha tratado con cadenas de caracteres o *strings* escuetas como "`Hola Mundo`", pero aquellas provenientes de los sensores presentan una sintaxis completamente distinta (en formato JSON), y superan los 200 B, excediendo la limitación de 192 B por mensaje establecida en el plan de datos de Swarm:


```
{
    "applicationID":"1",
    "applicationName":"RAK",
    "devEUI":"3334363558306904",
    "deviceName":"CC2",
    "timestamp":1656863144,
    "fCnt":5,
    "fPort":2,
    "data":"BQOqbDYAAAA=",
    "data_encode":"base64",
    "adr":true
}
```


Fig. 58: Payload en crudo del sensor CC en formato JSON

Es decir, el *payload* presenta un problema de tamaño y de formato. Ambos deberán ser ajustados mediante el *Payload Formatter* adecuado. Resumidamente, el *Payload Formatter* realizará una serie de conversiones sobre los datos para adecuarlos al *uplink* del módem, y eliminará aquellos campos que no resultan estrictamente necesarios para la monitorización, manteniendo los siguientes:

Clave	Función
`devEUI`	Permite identificar el dispositivo con exactitud
`data`	Contiene la medida codificada del sensor en cuestión
`deviceName`	Se añade un identificador adicional para reconocer el tipo de dispositivo
`timestamp`	Indispensable para ordenar cronológicamente las medidas, dado que la marca de tiempo ofrecida for Swarm únicamente hace referencia al momento de recepción del mensaje en la plataforma, dependiente de la hora de paso de los satélites.

Tabla 9: Claves necesarias en el objeto JSON

Su implementación se observa en el código de la Fig 59.

```
1.
2.  def message(client, topic, message_string):
3.
4.  if is_json(message_string):
5.
6.  message_dictionary = json.loads(message_string)
7.  msg = message_dictionary
8.  checksum = 0
9.
10. msg.pop('applicationName', None)
11. msg.pop('applicationID', None)
12. msg.pop('data_encode', None)
13. msg.pop('fPort', None)
14. msg.pop('fCnt', None)
15.
16. jsonObj = json.dumps(msg)
17. jsonObj = jsonObj.replace(' ','')
18.
19. payload =  len(jsonObj)
20.
21. if payload > 192:
22. tcpconn.send("Message too long!")
23. return
24.
25. command = b'$TD ' + hexlify(jsonObj.encode())
26. else:
27. payload =  len(message_string)
28. command = b'$TD ' + hexlify(message_string.encode())
29.
30. for i in command[1:]:
31. checksum = checksum ^ i
32.
33. command = command + b'*%02X\n'%checksum
34. tile.write(command)
35.
36. if TCPSTATE == TCPSTATE_CONNECTED:
37. tcpconn.send("\nNew message on topic:" + str (topic) + "\n")
38. tcpconn.send("\nOriginal string:\n")
39. tcpconn.send(message_string)
40. tcpconn.send("\nPayload:" + str (payload) + " Bytes\n")
41. tcpconn.send("\nM138 Transmit Data Command ($TD):\n")
42. tcpconn.send(command)
43.
44. displayLine(4, "New message! (" + str(payload) + "Bytes)")
45. displayLine(5, message_string)
46.
```

Fig 59: Código del Payload Formatter

A partir de la línea 35, el código proporciona una serie de mensajes por conexión TCP y a través del *display*. Resultarán de utilidad durante el ensayo práctico.

Activación del kit de evaluación Swarm

Para habilitar la transmisión mensajes a internet, es necesario que el GPS adquiera una referencia temporal válida para establecer los valores internos de fecha y hora. De lo contrario el *tile* detectará un estado no permitido y toda comunicación quedará anulada; los intentos de poner mensajes en cola serán respondidos con: `$TD ERR,GPS_TIMENOTSET*xx`

Esto se logrará cuando el kit se encuentre en un entorno libre de ruido e interferencias electromagnéticas, donde el RSSI (*Received Signal Strength Indicator*) baje de los -97 dBm, habitualmente a unos 5 km de la ciudad.

Durante el transcurso del proyecto se consideraron varios puntos para la realización de las pruebas de transmisión a cielo abierto, escogiéndose finalmente una zona con un RSSI de -100 dBm (2ª imagen).

Fig. 60: Primer intento de transmisión con un valor de RSSI de -95 dBm.

Fig. 61: Segundo intento de transmisión con un valor de RSSI de -100 dBm.

A priori la comunicación está garantizada bajo estas condiciones, siempre y cuando se programe a la hora de paso de un satélite SpaceBEE, indicada por el **Swarm Pass Checker.**

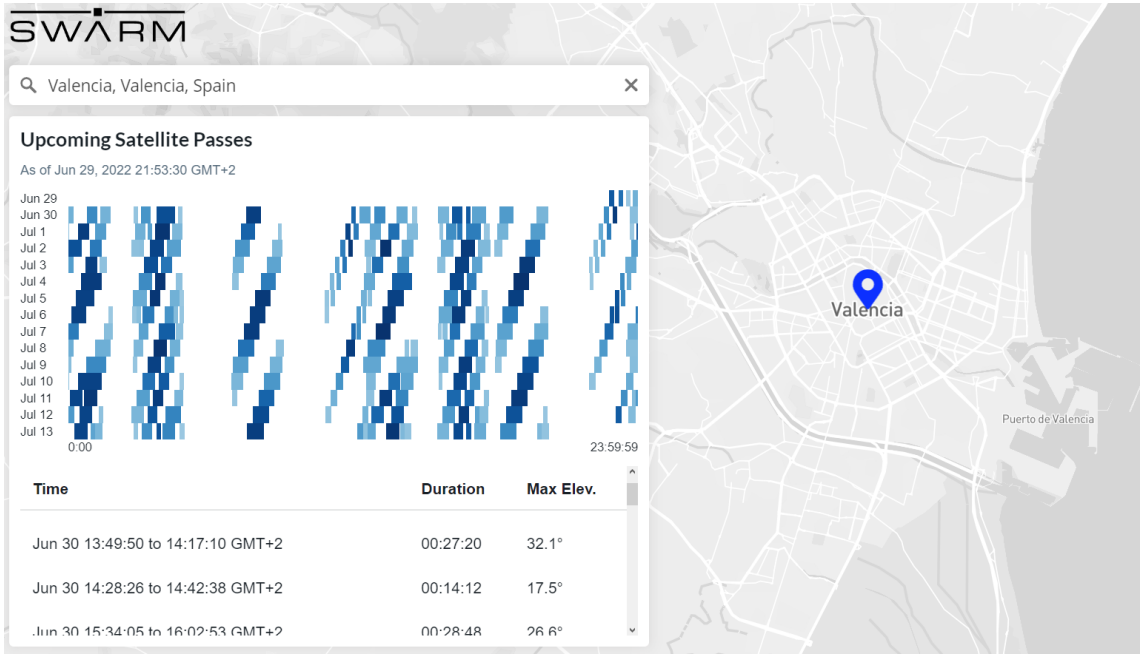


Fig. 62: Swarm Satellite Pass Checker mostrando las horas de paso de satélites por Valencia

Una vez en el emplazamiento, se establece una conexión TCP con la IP asignada al kit – visible en el *display* – y se confirma el funcionamiento del *Payload Formatter*.

```
Itaca@itaca-VPCSB3Q9E:~$ telnet 192.168.230.220
Trying 192.168.230.220...
Connected to 192.168.230.220.
Escape character is '^]'.

$M138 DATETIME*56
$DT 20220705095545,V*42
$RT RSSI=-86*23

New message on topic:application/1/device/+/rx

Original string:
{"applicationID":"1","applicationName":"RAK”,"devEUI":"a840415d31845954","dev
iceName":"CC1","timestamp":1656871541,"fCnt":1,"fPort":2,"data":"09010001000C
18","data_encode":"hexstring"}

Payload:95 Bytes

M138 Transmit Data Command ($TD):
$TD
7B226465766963654E616D65223A22434331222C2264617461223A223039303130303031303030
0433138222C2274696D657374616D70223A313635363837313534312C226465764555549223A22
6138343034313564333138343539353422 7D*4A
$TD OK,4521130262558*26
```

Fig. 63: Comunicación TCP/IP con el Kit a través de Telnet

El mensaje `$M138 DATETIME*56` indica que el GPS ha adquirido correctamente la referencia temporal. Además, la petición de transmisión ha sido respondida con `$TD OK`, lo que significa que en breve aparecerá en la plataforma:

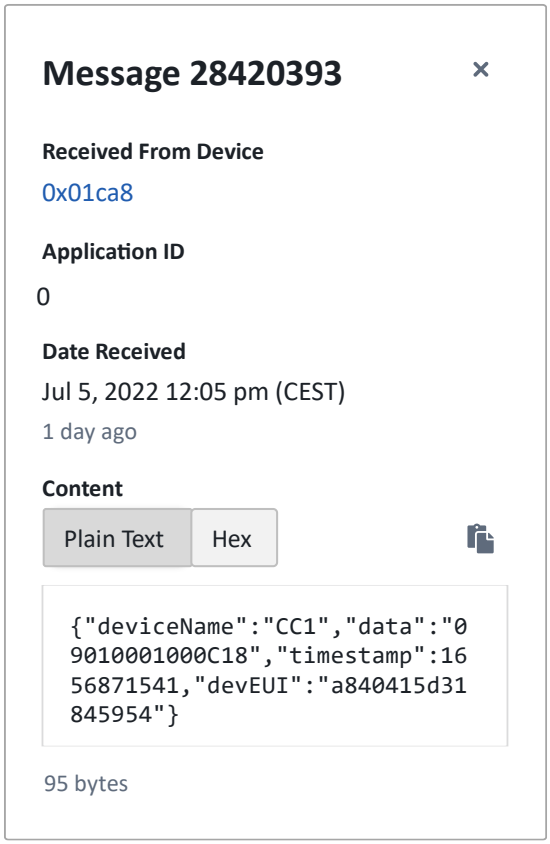


Fig. 64: Ventana de nuevo mensaje en Swarm Hive. Contenido en texto simple (ASCII)

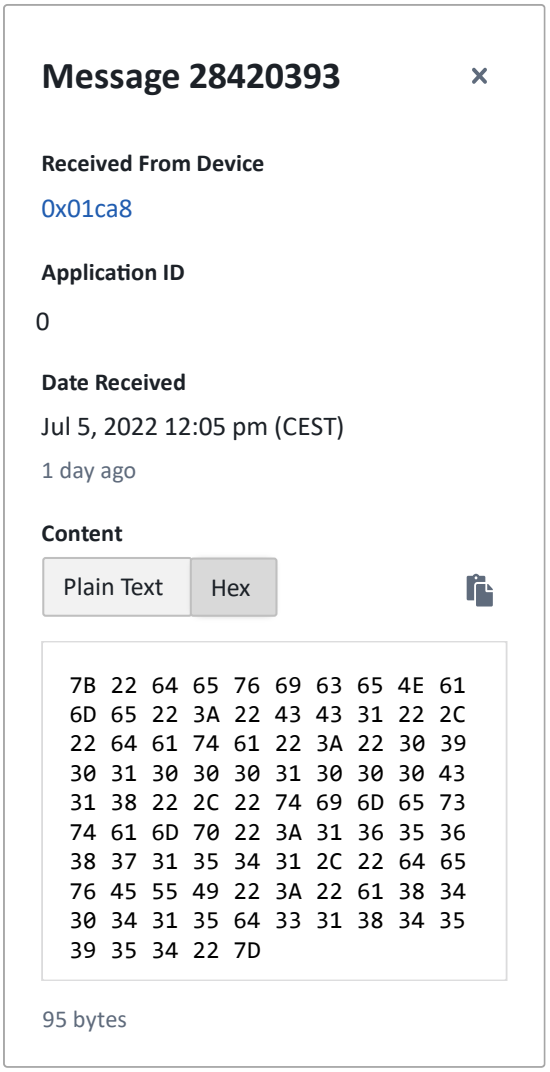


Fig. 65: Ventana de nuevo mensaje en Swarm Hive. Contenido en hexadecimal.

De esta forma se da por concluida toda la labor de programación y manipulación del software del material del despliegue.

5.4. Creación del panel de visualización

Para abordar la creación del panel de visualización, se utilizarán una serie de utilidades multiplataforma de código abierto:

- **Node-RED**: Se trata de una herramienta de programación visual basada en funciones JavaScript, para la interconexión entre dispositivos *hardware*, aplicaciones web, APIs y servicios en línea mediante flujos de datos. Será el encargado de conectar Swarm Hive a la base de datos.
- **InfluxDB**: Es un sistema de gestión de bases datos de lectura / escritura, y de alto rendimiento. No tiene dependencias externas y acepta datos mediante HTTP, TCP y UDP, lo que permite su acceso mediante Node-RED.
- **Grafana**: Consiste en una herramienta para la visualización interactiva y dinámica de datos de series temporales, como es el caso que nos ocupa. Se utilizará para crear una interfaz intuitiva formada por gráficas y tablas que presenten las medidas de los sensores de forma clara y organizada.

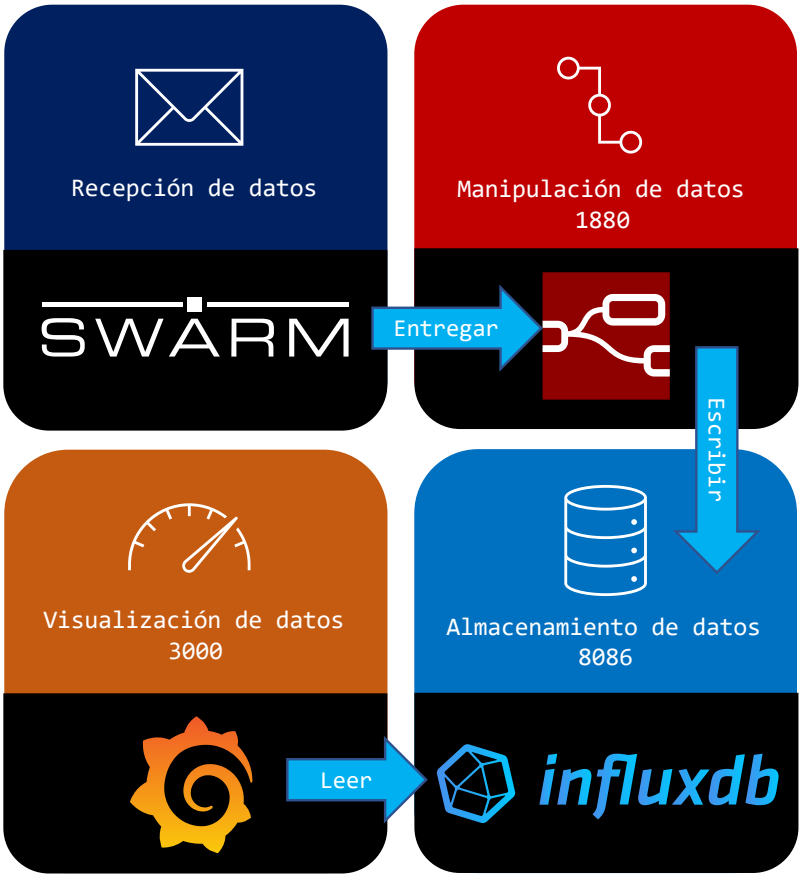


Fig. 66: Esquema del recorrido del mensaje desde la recepción en Swarm Hive hasta su representación en Grafana

La combinación Node-RED / InfluxDB / Grafana constituye una práctica común en escenarios IoT, ampliamente documentada. No es objeto de la memoria detallar el funcionamiento individual de cada herramienta, ni la forma enlazarlas. Por otra parte, si se mostrará cómo se ha llevado a cabo el vínculo con Swarm Hive por ser la plataforma exclusiva de este proyecto.

En la sección *Delivery*, añadiremos un nuevo método de entrega basado en *Webhooks*. De forma inversa a la dinámica típica en las APIs – en la que el cliente hace una petición al servidor – este mecanismo permite a la aplicación servidora notificar a la aplicación cliente mediante el envío automatizado de mensajes a la dirección especificada, haciendo uso de métodos HTTP cuando sucede un determinado evento desencadenante.

En este caso, el uplink del satélite provocará una petición POST en la URL del flujo de trabajo de Node-RED correspondiente al proyecto, incluyendo el payload en el cuerpo de la petición y los *tokens* de autenticación.

De esta forma se consigue establecer una comunicación unidireccional y en tiempo real desde el origen (Swarm Hive) hacia el destino (Node-RED).

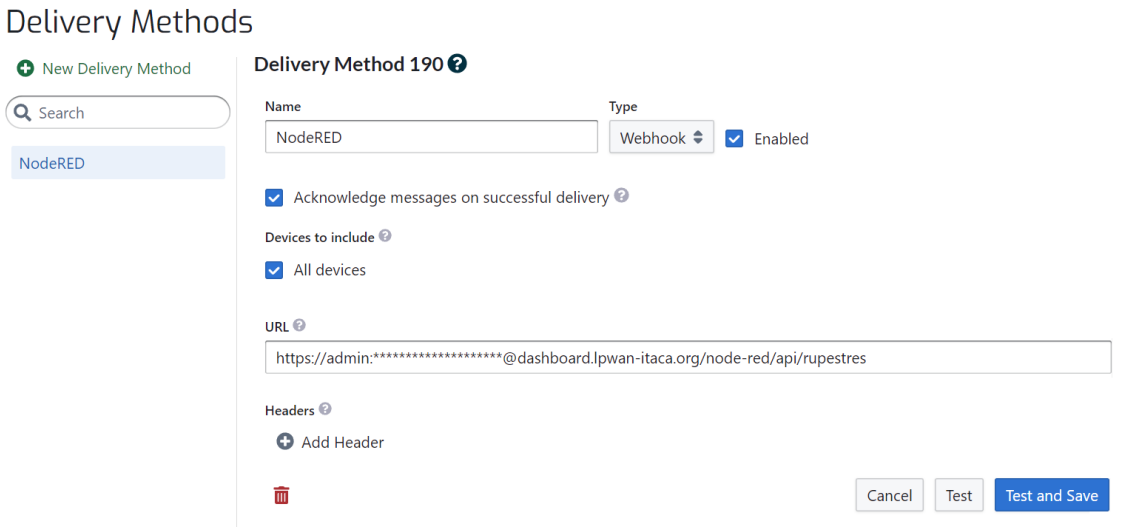


Fig. 67: Panel de Métodos de entrega de Swarm Hive

Se activará la primera opción para habilitar el envío de mensajes de reconocimiento o *acknowledge / acknowledgement* (ACK), que confirman la recepción del mensaje emitido por Swarm Hive.

Mientras tanto, se desarrolla la estructura de flujos en Node-RED. La Fig. 68 presenta un modelo de esquema básico para cargar datos a InfluxDB.

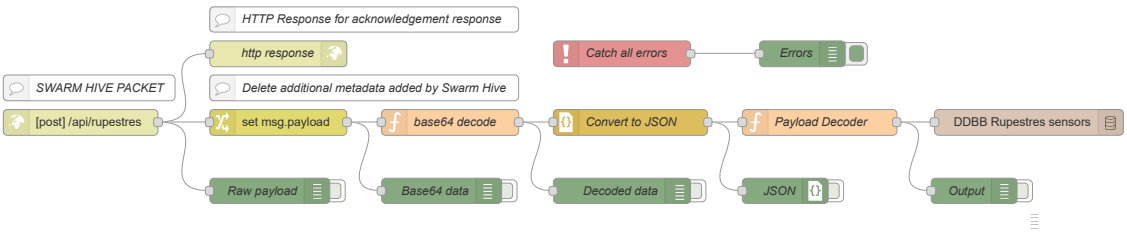


Fig. 68: Flujo en Node-RED (Gráfico vectorial ampliable)

Con todas las conexiones establecidas, es hora de realizar las pruebas de funcionamiento en la zona de transmisión habitual: un apartadero en la CV-401, la carretera de Alfafar al Saler.

5.5. Pruebas de funcionamiento

A continuación se adjunta una serie de fotografías descriptivas del despliegue. Para este efecto, se ha utilizado el gateway de interior RAK7268, completamente análogo al RAK7289 en cuanto al *firmware*.

Fig. 69: Estación meteorológica UBIQ-IoT WS100LRW en su mástil

Fig. 70: Sensores de CC utilizados

Fig. 71: Gateway conectado a powerbank

Fig. 72: Despliegue completo

Al encender el Gateway, los mensajes comienzan a llegar al Hive. El mensaje de *acknowledgement* indica que Node-RED recibe la información correctamente.

Messages

2022-06-11 00:00 | End date | CEST | Send Message

Way	Hive Packet ID	Device ID	Date (CEST)	Size (bytes)
	28605745 Acknowledged	0x01ca8	Jul 7, 2022 7:45 pm	103
	28605743 Acknowledged	0x01ca8	Jul 7, 2022 7:45 pm	97
	28605741 Acknowledged	0x01ca8	Jul 7, 2022 7:45 pm	97
	28605739 Acknowledged	0x01ca8	Jul 7, 2022 7:45 pm	97
	28605737 Acknowledged	0x01ca8	Jul 7, 2022 7:45 pm	96
	28605735 Acknowledged	0x01ca8	Jul 7, 2022 7:45 pm	96
	28605537 Acknowledged	0x01ca8	Jul 7, 2022 7:41 pm	96
	28605534 Acknowledged	0x01ca8	Jul 7, 2022 7:41 pm	138
	28605530 Acknowledged	0x01ca8	Jul 7, 2022 7:41 pm	96
	28602383 Acknowledged	0x01ca8	Jul 7, 2022 6:41 pm	95
	28602382 Acknowledged	0x01ca8	Jul 7, 2022 6:41 pm	103
	28602380 Acknowledged	0x01ca8	Jul 7, 2022 6:41 pm	95
	28602377 Acknowledged	0x01ca8	Jul 7, 2022 6:41 pm	103

Fig. 73: Historial de mensajes en Swarm Hive tras la realización de las pruebas

Y así lo confirma su consola de depuración:

```
7/7/2022, 6:41:01 PM node: 734132db.05437c msg.payload : Object
{ deviceName: "CC2", timestamp: 1657212061, devEUI: "3334363558306904", tempe
rature: 30.23, humidity: 51.23 … }

7/7/2022, 6:46:04 PMnode: 734132db.05437c msg.payload : Object
{ deviceName: "CC1", timestamp: 1657212362, devEUI: "a840415d31845954", tempe
rature: 30.11, humidity: 51.40 … }

7/7/2022, 6:51:05 PMnode: 734132db.05437c msg.payload : Object
{ deviceName: "CC2", timestamp: 1657212663, devEUI: "3334363558306904", tempe
rature: 30.01, humidity: 53.00 … }

7/7/2022, 6:56:07 PMnode: 734132db.05437c msg.payload : Object
{ deviceName: "CC1", timestamp: 1657212964, devEUI: "a840415d31845954", tempe
rature: 30.64, humidity: 52.92 … }

7/7/2022, 7:01:08 PMnode: 734132db.05437c msg.payload : Object
{ deviceName: "CC2", timestamp: 1657213263, devEUI: "3334363558306904", tempe
rature: 30.01, humidity: 53.81 … }

7/7/2022, 7:06:07 PMnode: 734132db.05437c msg.payload : Object
{ deviceName: "CC1", timestamp: 1657213564, devEUI: "a840415d31845954", tempe
rature: 30.05, humidity: 53.24 … }
```

Fig. 74: Mensajes de depuración de Node-RED

En el panel de Grafana, se comprueba que los valores quedan representados en sus gráficas.

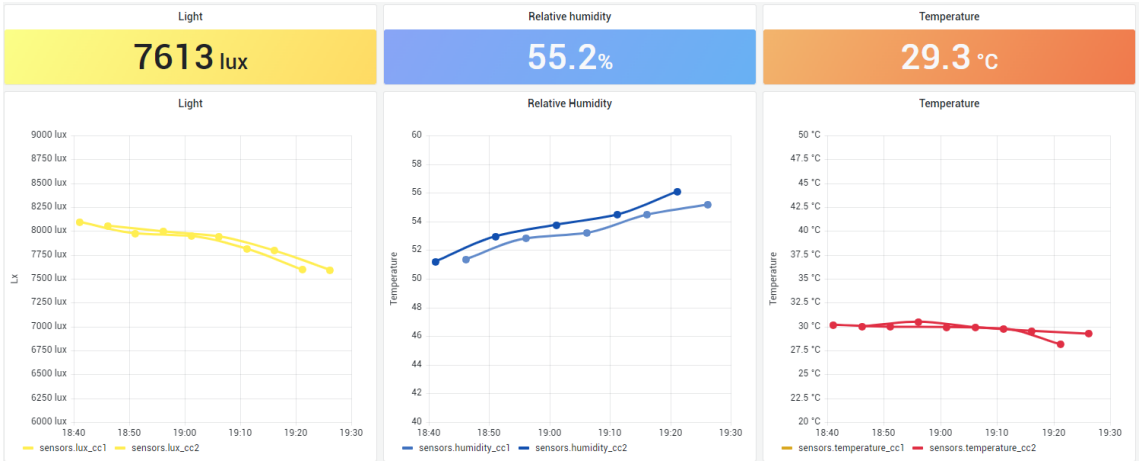


Fig. 75: Panel de visualización en Grafana

Si se desea consultar los datos de la Fig. 75 con más detalle, en el momento de redacción de la memoria se puede acceder al panel a través del siguiente enlace:

https://dashboard.lpwan-itaca.org/grafana/dashboard/snapshot/G37Cvlif5RYuGMlirJ8cPzW5jLh59Yto

6. Conclusiones

6.1. Sobre el trabajo realizado

Según se ha demostrado en el último apartado, el presente proyecto posee un carácter eminentemente práctico, y abarca y combina diferentes áreas de estudio, algunas de ellas tradicionales como la conservación del patrimonio cultural, con otras particularmente novedosas y actuales como el Internet de las Cosas o el *NewSpace*, lo que ha requerido una labor de investigación desde el inicio del trabajo que permitiese adquirir la bases necesarias.

Por otra parte, se han aplicado conocimientos de multitud de disciplinas impartidas a lo largo del grado, muy especialmente de informática y electrónica digital para el desarrollo del *software* del proyecto, y de redes para la interconexión de los dispositivos.

Como se ha podido ver en el estudio de alternativas, parte de la dificultad en este trabajo surge a la hora de identificar la instrumentación necesaria para confeccionar el sistema. No obstante, la complejidad principal radica en la cantidad de equipos a integrar en el despliegue final. Dispositivos de diversa índole que, combinados, conforman un sistema de monitorización de gran envergadura capaz de solventar un problema de comunicación en una localización a priori desprovista de conectividad inalámbrica.

Dicho sistema encuentra su aplicación directa en el ámbito de conservación del patrimonio cultural, para el que ha sido concebido, pero su carácter flexible y escalable permite adaptarse a prácticamente cualquier situación que exija la monitorización de parámetros ambientales. En definitiva, se ha alcanzado el objetivo del proyecto, satisfaciendo todos los requisitos expuestos.

6.2. Futuras mejoras

Este proyecto se presenta como la implementación de un prototipo prueba de concepto, resumido y limitado en algunos aspectos.

Una de las principales mejoras que se podría llevar a cabo en un futuro perfeccionamiento del sistema, se encuentra en el **manejo del payload** de los sensores.

Actualmente, los mensajes son transmitidos de un eslabón a otro en formato JSON, que presenta una estructura de datos sumamente legible y cómoda para enviar por TCP/IP (MQTT, páginas web…). Ello ha permitido verificar en todo momento que el contenido de los mensajes era correcto y ha facilitado su decodificación, agilizando las últimas etapas del desarrollo, en que el tiempo invertido era crítico.

No obstante, este formato resulta ineficiente en cuanto al tamaño, al contener caracteres de formato como comillas, llaves, puntos y espacios (aunque estos últimos sí se han suprimido).

En este caso, la magnitud del mensaje que contiene las medidas ronda los 100 B; se podría reducir a casi una cuarta parte enviando al módem una ristra de bytes íntegramente en hexadecimal o base64, incluida la marca de tiempo. La compactación de los datos resultaría en una reducción del tiempo de transmisión y del consumo energético, recomendable en un despliegue posterior.

7. Bibliografía

Agha, K. A., Pujolle, G., & Yahiya, T. A. (2016). Specific propietary solution: SIGFOX example. En *Mobile and Wireless Networks* (pág. 241). Wiley. Obtenido de https://books.google.com/books?id=_BzfDAAAQBAJ&pg=PA241

Akintade, O. O., Yesufu, T. K., & Kehinde, L. O. (2019). Development of an MQTT-based IoT Architecture for Energy-Efficient and Low-Cost Applications. *International Journal of Internet of Things, 8*(2), 27-35. doi: 10.5923/j.ijit.20190802.01

Angel Perles, E. P.-M.-D. (2018). An energy-efficient internet of things (IoT) architecture for preventive conservation of cultural heritage. *Future Generation Computer Systems*, 566-581. doi: http://dx.doi.org/10.1016/j.future.2017.06.030

Atkinson, J. K. (20 de septiembre de 2015). *History of Preventive Conservation.* Obtenido de The International Institute for Conservation of Historic and Artistic Works: https://www.iiconservation.org/sites/default/files/news/attachments/6661-iic-itcc_2015_ppt_history_of_preventive_conservation_jo_atkinson.pdf

Botello, J. B. (18 de diciembre de 2017). *Causas de deterioro.* Obtenido de Artículos Especializados: https://www.adabi.org.mx/publicaciones/artEsp/ccre/causasDeterioroPatrimonioDocumental.pdf

Brown, I. (s.f.). A Detailed Breakdown of LPWAN technologies and providers. *Lux Research.* Obtenido de http://web.luxresearchinc.com/hubfs/Insight_Breakdown_of_LPWAN_Technologies.pdf

Dahlman, E., Parkvall, S., Sköld, J., & Beming, P. (2008). *3G Evolution HSPA and LTE for Mobile Broadband. Second Edition.* Oxford: Elsevier Ltd. Obtenido de https://ptabdata.blob.core.windows.net/files/2017/IPR2017-01471/v13_1013%20-%203G%20Evolution.pdf

Davis Instruments. (2019). *Davis.* Obtenido de Precision Weather Instruments. Global Catalogue: https://www.e-wetter.eu/downloads/pr57_2019_catalog_global_nopricing.pdf

Ericsson. (junio de 2009). LTE – An introduction. *White Paper, Uen Rev B, 284 23-3124*. Obtenido de https://telecoms.com/files/2009/03/lte_overview.pdf

Joffroy, T. (2012). Preventive Conservation: A concept suited to the conservation of earthen-architectural heritage? *CRAterre-ENSAG*. Obtenido de https://hal.archives-ouvertes.fr/hal-01861818/document

Kulu, E. (2022). *SpaceBEE 0.24U*. Obtenido de Nanosats Database: https://www.nanosats.eu/sat/spacebee-1-4

L.Diffey, B. (2002). Sources and measurement of ultraviolet radiation. *Methods, 28*(1), 4-13. doi: https://doi.org/10.1016/S1046-2023(02)00204-9

Laboratorio de Procesado de Imagen. (3 de marzo de 2014). *Sistemas Lineales. Tema1. Introducción. Señales y sistemas.* Obtenido de Laboratorio de Procesado de Imagen. E.T.S.I. Telecomunicación. Universidad de Valladolid: http://www.lpi.tel.uva.es/lineales/apuntes/tema1.pdf

Lambert, S. (13 de enero de 2014). The Early History of Preventive Conservation in Great Britain and the United States (1850–1950). *CeROArt*. doi: https://doi.org/10.4000/ceroart.3765

Mekki, K., Bajic, E., Chaxel, F., & Meyer, F. (2019). A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express, 5*(1), 1-7. doi: https://doi.org/10.1016/j.icte.2017.12.005

Rubell, B. (23 de Julio de 2019). *MQTT in CircuitPython*. Obtenido de Adafruit: https://learn.adafruit.com/mqtt-in-circuitpython

Saiz-Jiménez, C. (2009). Los microorganismos y las cuevas con pinturas rupestres. *Sautuola: Revista del Instituto de Prehistoria y Arqueología*, págs. 453-460. Obtenido de http://hdl.handle.net/10261/63215

Schertz, J. (19 de January de 2019). *Swarm Preparing for 150 Cubesat Constellation.* Obtenido de The Space Resource: https://www.thespaceresource.com/news/2019/1/swarm-preparing-for-150-cubesat-constellation

Smith, D. K. (24 de marzo de 2017). *How fast are AM23xx(DHT22), SHT71 and BME280?* Obtenido de Kimberly & Robert's Home Page: https://www.kandrsmith.org/

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politécnica de València

Monitorización de parámetros ambientales del patrimonio cultural mediante dispositivos IoT y conexión por satélite

II. PLANOS

Autor: D. Miguel Gotor Ramos

Tutores: D. Ángel Francisco Perles Ivars

D. Jaime Laborda Macario

Índice de planos

Índice de tablas

1. Historial de revisiones

Historial de revisiones			
Versión	Fecha	Autor	Descripción
0.1	13/07/2022	M. Gotor	Primera versión del prototipo de prueba de concepto.
0.2	14/07/2022	M. Gotor	Sustitución de algunos de los bloques por iconos representativos.
0.3	19/07/2022	M. Gotor	Cambio en la disposición de los bloques.
0.4	23/07/2022	M. Gotor	Corrección errata protocolo. Ajustes estéticos menores.

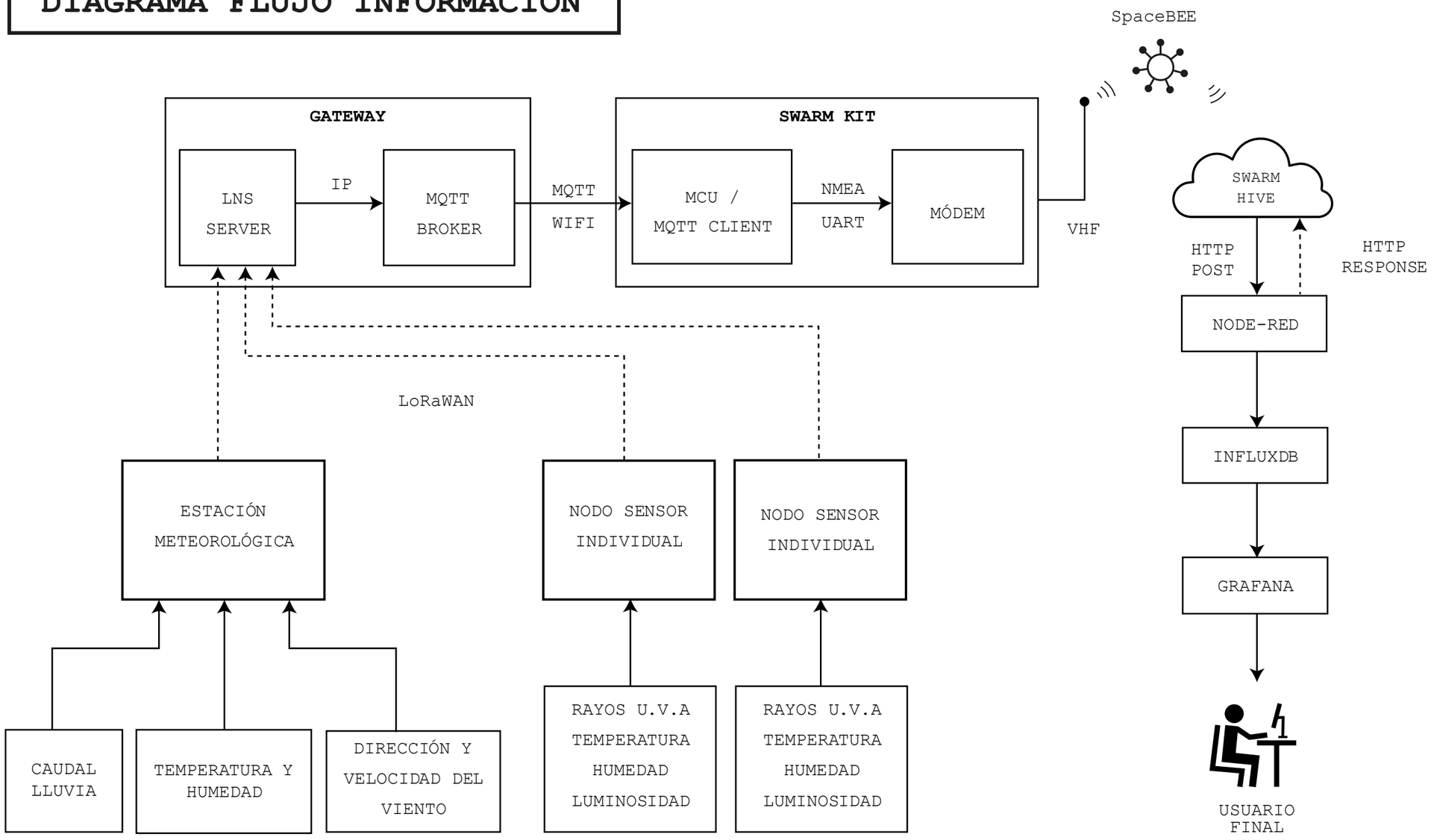
Tabla 1: Historial de revisiones

2. Sobre los planos

A continuación se incluye, no un plano convencional como exigiría una instalación física, sino un diagrama del flujo de la información, que muestra el recorrido completo de los datos desde su captación por los nodos sensores, hasta su llegada al usuario final.

Asimismo, el plano constituye un esquema de la comunicación, indicando todas las conexiones entre los elementos físicos y lógicos de la red.

DIAGRAMA FLUJO INFORMACION



	NOMBRE	FECHA	ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DEL DISEÑO
PROYECTADO:	Miguel Gotor Ramos	14/7/22	TÍTULO: SISTEMA DE MONITORIZACIÓN DE PARÁMETROS AMBIENTALES DEL PATRIMONIO CULTURAL MEDIANTE DISPOSITIVOS IOT. Y COMUNICACIÓN POR SATÉLITE
DIBUJADO:	Miguel Gotor Ramos	23/7/22	
CONFORMADO:	Angel Fco. Perles	23/7/22	

ESCALA:	DENOMINACIÓN del PLANO:	Nº de PLANO:
No procede	DIAGRAMA FLUJO INFORMACIÓN	001

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politécnica de València

Monitorización de parámetros ambientales del patrimonio cultural mediante dispositivos IoT y conexión por satélite

III. PLIEGO DE CONDICIONES

Autor: D. Miguel Gotor Ramos

Tutores: D. Ángel Francisco Perles Ivars

D. Jaime Laborda Macario

Índice del pliego de condiciones

Índice de tablas

Glosario de siglas y abreviaturas

Abreviatura	Significado	Pág.
ETSI	European Telecommunications Standards Institute	3
ISM	Industrial Scientific & Medical	3
RSSI	Received Signal Strength Indicator	2

1. Definición y alcance del pliego

Esta especificación se refiere a la implementación de un sistema de monitorización de parámetros ambientales y climatológicos. Se trata de un sistema electrónico de bajo consumo, capaz de operar a la intemperie. Funciona de forma autónoma en ausencia de suministro eléctrico y alejado de la cobertura de sistemas de comunicación móvil convencionales y soporta condiciones ambientales extremas de calor o humedad. Está formado por diversos dispositivos individuales: utilizando una red LoRaWAN, los nodos sensores inalámbricos realizarán lecturas periódicamente en la zona del despliegue, y compartirán sus medidas con la constelación de satélites de Swarm, haciéndolas accesibles desde internet.

El presente pliego de condiciones expone las condiciones generales y técnicas que se exigen durante la ejecución del proyecto, y bajo las cuales deberá componerse, reproducirse y utilizarse el sistema desarrollado en el documento de la memoria.

El sistema objeto del pliego no requiere ningún proceso de diseño ni fabricación de hardware. Ninguno de los elementos físicos o materiales constituyentes del sistema son propiedad del autor del proyecto. La originalidad del proyecto yace en la labor de configuración de estos elementos, y en la programación del software. Por ello, las directrices que se incluyen a continuación harán referencia exclusivamente a la instalación, uso y mantenimiento del mencionado conjunto, así como a las circunstancias que deben darse en el despliegue final para asegurar la comunicación entre los distintos dispositivos y una recolección de datos satisfactoria.

2. Condiciones y normas de carácter general

2.1. Instalación

Al tratarse de un diseño en fase de pruebas, el despliegue deberá realizarse por personal cualificado que conozca los equipos. Resulta necesario ya que existen unas condiciones específicas que deberán cumplirse con el fin de que el sistema funcione de forma adecuada.

Los nodos sensores no deben ubicarse en puntos próximos a fuentes artificiales de calor o frío que puedan alterar las mediciones de temperatura o deteriorar los componentes electrónicos.

Antes de colocar cada nodo, se deberá realizar un estudio del impacto visual en la obra de arte rupestre. En caso de duda se deberá consultar al autor del proyecto o a un experto conservador.

Los nodos finales se situarán al alcance de la antena del gateway, que a su vez deberá hallarse al alcance del kit. Se realizarán las comprobaciones pertinentes que verifiquen la comunicación entre todos los elementos del sistema.

La presencia de objetos ajenos a la instalación puede causar efectos de difracción que interrumpen las transmisiones de radio, por lo que el alcance de los dispositivos dependerá las circunstancias del escenario. Por ello:

- Previo a la instalación de la red de sensores, se llevará a cabo un estudio de utilización del espectro de banda con las pruebas pertinentes que descarte la existencia de problemas en el espectro de radiofrecuencias que impidan o alteren la comunicación de los dispositivos.
- Los elementos de interconexión de red se instalarán en ausencia de objetos metálicos que puedan perturbar la propagación de las ondas electromagnéticas.

Dada la naturaleza de la tecnología de comunicación utilizada, no se exige línea de visión directa entre los dispositivos LoRaWAN.

Por otra parte, el kit satélite deberá encontrarse a cielo descubierto, sin techo ni otro reparo alguno, en una zona alejada de fuentes de ruido e interferencias electromagnéticas, con un RSSI inferior a -95 dBm.

Con el fin de maximizar la irradiación solar y la autonomía del kit, su panel solar deberá orientarse hacia el sur geográfico (ángulo azimutal de 180º).

Al trabajar con bandas del espectro ISM, no se exige licenciar la instalación, pero sí cumplir las condiciones estipuladas por la ETSI en la banda EU863-870MHz, alias EU868:

Sub-banda	**Banda de frecuencia (MHz)**	**Potencia de transmisión máxima (dBm)**	**Ciclo de trabajo máximo**
G	863,00 – 868,00	14	1%
G1	868,00 – 868,60	14	1%
G2	868,70 – 869,20	14	0,1%
G3	869,40 – 869,65	27	10%
G4	868,70 – 870,00	14	1%

Tabla 1: Limitaciones en la banda EU868

Los sensores individuales que conforman el sistema de medida multipunto se instalarán de manera que queden al resguardo de la lluvia, en el interior del abrigo rocoso.

Deberán quedar fijos en la pared, empleando los accesorios con el mecanismo de fijación correspondiente como se indica en el manual de usuario.

La estación meteorológica irá instalada en el extremo final de un mástil, mientras que el módulo satélite se montará sobre el trípode incluido en dicho kit.

Tras el montaje, se verificará la verticalidad de ambos soportes, utilizando el nivel de burbuja integrado. No deberá apreciarse inclinación aparente.

Asimismo, se realizarán las correspondientes pruebas para comprobar su robustez. Se aplicará una fuerza axial no superior a 5 kg. No deberá apreciarse movimiento aparente alguno. De lo contrario, se emplearán los mecanismos de sujeción adicionales que se consideren oportunos.

No se garantiza el correcto funcionamiento de este sistema empleando software, hardware, o cualquier configuración diferente a los indicados en la memoria del proyecto, por lo que se desaconseja modificar el código fuente del microcontrolador.

En caso de modificación, la responsabilidad del correcto funcionamiento del sistema recaerá sobre la persona u organismo que realice la modificación.

2.2. Utilización

Una vez desplegado el sistema y configurados los nodos y los gateways, se conectará automáticamente a internet a la hora de paso de los satélites. Desde ese instante, puede llegar a transcurrir entre 5 y 10 minutos hasta la recepción de los datos en las diversas plataformas.

El sistema está diseñado para su utilización por el usuario sin asistencia ulterior importante del proveedor. No obstante, si los parámetros de la aplicación varían, puede ser necesario una reconfiguración de alguna de las partes, incluyendo la posibilidad de tener que hacerlo en el lugar físico donde se encuentra el sistema.

Para más información sobre la reconfiguración de los dispositivos, y la modificación del intervalo de sus medidas, consultar las hojas de especificaciones adjuntas.

Al tratase de un prototipo experimental, no se garantiza que funcione correctamente de forma indefinida, pudiendo detenerse en cualquier momento sin previo aviso, interrumpiendo el servicio.

Se recomienda, por tanto, que no se utilice como substituto a un método de monitorización tradicional hasta la finalización de la fase de pruebas.

Para más detalles sobre el funcionamiento y utilización del equipo, póngase en contacto con el autor del proyecto.

2.3. Mantenimiento

El mantenimiento requerido por el sistema incluye el recambio – o recarga, según su caso – de las baterías, la inspección visual periódica y la comprobación de los servicios en la nube.

El periodo de sustitución / recarga de las baterías es diferente para cada uno de los dispositivos, y depende de la frecuencia de las mediciones, a determinar en función de las necesidades del usuario final.

Al final de la vida útil de las baterías no recargables, deberán ser desechadas según la normativa 2006/66/CE, correspondiente a pilas y acumuladores y la gestión ambiental de sus residuos.

No se asegurará un correcto funcionamiento del sistema sin una inspección visual anual que no muestre ningún defecto aparente. Deberá ser realizada por personal cualificado que conozca el sistema.

Al ser un proyecto en fase de pruebas, existe la posibilidad de tener que realizar cambios en la configuración para corregir anomalías, o en caso de reemplazo de alguno de los componentes por avería.

Este procedimiento deberá realizarse in-situ, por lo que se desaconseja la instalación en lugares de difícil acceso antes de terminar el periodo de pruebas.

3. Condiciones técnicas particulares

3.1. Hardware

Todo el hardware empleado para el sistema debe ser exactamente el indicado en la memoria del proyecto. A continuación se detallan las especificaciones por referencia a fabricante:

- La estación meteorológica será el modelo UBIQ-IoT WS100LRW.
- El nodo individual será el sensor CollectionCare.
- El gateway LoRaWAN de exterior será el RAK7289 WisGate Edge Pro de RAK Wireless
- El terminal satélite será el kit de evaluación de Swarm (*Swarm Eval Kit)*.

La utilización de componentes diferentes a los indicados requerirá la aprobación del autor del proyecto. En caso contrario, no existirá garantía de correcto funcionamiento del sistema.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politécnica de València

Monitorización de parámetros ambientales del patrimonio cultural mediante dispositivos IoT y conexión por satélite

IV. PRESUPUESTO

Autor: D. Miguel Gotor Ramos

Tutores: D. Ángel Francisco Perles Ivars

D. Jaime Laborda Macario

Índice del presupuesto

Índice de tablas

Glosario de siglas y abreviaturas

Abreviatura	Significado	Pág.
EA	Each	1
ECTS	European Credit Transfer and Accumulation System	2
IVA	Impuesto de Valor Añadido	3
USB	Universal Serial Bus	1

1. Sobre el presupuesto

Para la elaboración del presupuesto, se han tomado los precios ofertados en las tiendas online de los distribuidores autorizados y de las solicitudes de presupuesto a los fabricantes del producto original, obteniendo de esta manera el precio de adquisición para el consumidor final de todos los equipos a fecha de julio de 2022. El presente presupuesto tiene una validez y garantía de un año.

Se ha añadido un porcentaje de costes generales del 13 % contemplando de esta manera los gastos generales en que se pueda incurrir durante la realización del proyecto, y un 6 % de beneficio industrial, que incrementan el presupuesto de ejecución material de la obra.

El presupuesto deberá ir acompañado de los documentos de la memoria y del pliego de condiciones del proyecto, y tan solo contempla lo descrito en ellos. Cualquier requisito que difiera de dichas descripciones puede tener costes adicionales no reflejados en el presente presupuesto.

2. Materiales y componentes

Materiales y componentes				
Uds.	Denominación	Cantidad	Precio	Total (€)
EA	Swarm Eval Kit	1	499,00 €	499,00 €
EA	CollectionCare Sensor Node	2	48,18 €	96,36 €
EA	RAK7289 Wireless WisGate Edge Pro	1	357,02 €	357,02 €
EA	Cable USB-C genérico	1	6,00 €	6,00 €
EA	UBIQ-IoT WS1000LRW	1	1.080,00 €	1.080,00 €
Subtotal materiales y componentes				**2.038,38 €**

Tabla 1: Coste de materiales y componentes

El coste total de los materiales asciende a 2.038,38 €

3. Mano de obra

El coste de la mano de obra se ha calculado a basándose en la estimación del salario horario de un graduado en ingeniería técnica industrial (26 e). Para ello se ha tenido en cuenta la dedicación estimada obtenida conforme a la carga de 12 ECTS (European Credit Transfer and Accumulation System). Dado que cada crédito corresponde a 25 h, la dedicación total es de aproximadamente 300 h. Por otra parte, se ha considerado un extra de 20 h de trabajo correspondientes a la realización de algunas tareas adicionales que exceden el ámbito habitual de trabajo, como los desplazamientos para realizar pruebas de transmisión. Por ello, el total asciende a 320 h. En la siguiente tabla se encuentre el desglose de las horas dedicadas a las distintas actividades.

Uds.	Descripción	
h	**Desarrollo despliegue LoRaWAN**	15
	1. Configuración gateway	6
	2. Configuración nodos	6
	3. Pruebas de funcionamiento	3
h	**Desarrollo conexión MQTT y transmisión a satélite**	190
	1. Configuración gateway	5
	2. Configuración kit satélite	10
	3. Desarrollo de código fuente prototipo	30
	4. Desarrollo de código fuente aplicación final e integración en el kit	80
	5. Depuración aplicación final	50
	6. Pruebas de funcionamiento	15
h	**Desarrollo interfaz gráfica**	15
	1. Creación diagrama de flujo en Node-RED	6
	2. Confeccionamiento del panel de visualización en Grafana	6
	3. Pruebas de funcionamiento	3
h	**Redacción y generación de documentación**	100
	1. Redacción de la memoria del proyecto	90
	2. Elaboración de planos	5
	3. Elaboración de presupuestos	5
	TOTAL	320

Tabla 2: Desglose mano de obra

Mano de obra				
Uds.	Denominación	Cantidad	Precio	Total
h	Investigación, diseño y desarrollo	320	26,00 €	8.320,00 €
Graduado en Ingeniería Electrónica Industrial y Automática				
Subtotal mano de obra				**8.320,00 €**

Tabla 3: Coste de mano de obra

El precio final en concepto de mano de obra asciende a 8.320,00 €

4. Coste del proyecto

Coste del proyecto	
Materiales y componentes	2.038,38 €
Mano de obra	8.320,00 €
TOTAL	**10.358,38 €**
13 % Gastos generales	1.346,59 €
6 % Beneficio industrial	621,50 €
Suma	**12.326,47 €**
21 % IVA	2.588,56 €
COSTE TOTAL DEL PROYECTO	**14.915,03 €**

Tabla 4: Coste total del proyecto

La elaboración del proyecto tiene un coste total asociado de 14.915,03 €.- CATORCE MIL NOVECIENTOS QUINCE EUROS CON TRES CÉNTIMOS (IVA INCLUIDO).

Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politécnica de València

Monitorización de parámetros ambientales del patrimonio cultural mediante dispositivos IoT y conexión por satélite

V. ANEXOS

Autor: D. Miguel Gotor Ramos

Tutores: D. Ángel Francisco Perles Ivars

D. Jaime Laborda Macario

Índice del Anexo 1 – Código.

Índice de figuras

1. Código del microcontrolador

A continuación de adjunta el listado en lenguaje CircuitPython, correspondiente al fichero `code.py`, cuyo contenido ha sido editado para añadir soporte a MQTT. De esta forma se logra establecer una conexión segura a un bróker configurable para recibir mensajes y transmitirlos a un satélite en la constelación Swarm, como se ha descrito en la memoria.

```
1.  # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
2.  # Copyright (C) 2021, Swarm Technologies, Inc.  All rights reserved.  #
3.  # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
4.  # MQTT support added by Miguel Gotor (migora@etsid.upv.es)            #
5.  # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
6.
7.  VERSION = '1.4'
8.  import board
9.  import displayio
10. import digitalio
11. import terminalio
12. import busio
13. import time
14. import neopixel
15. from adafruit_display_text import label
16. import adafruit_displayio_sh1107
17. from barbudor_ina3221 import *
18. import supervisor
19. import sys
20. import microcontroller
21. import json
22. from binascii import hexlify
23. from microcontroller import watchdog as w
24. from watchdog import WatchDogMode
25. from adafruit_debouncer import Debouncer
26. import gc
27.
28. supervisor.disable_autoreload()
29.
30. # MQTT global variables
31.
32. application_feed = None
33. mqttStarted = False
34. mqtt_client = None
35.
36. ina3221 = None
37. tile = None
```

```
38.  tileLine = bytearray(800)
39.  tilePtr = 0
40.
41.  global DEVTAG
42.  DEVTAG = "TILE"
43.
44.  TILE_STATE_UNKNOWN = 0
45.  TILE_STATE_REBOOTING = 1
46.  TILE_STATE_2 = 2
47.  TILE_STATE_3 = 3
48.  TILE_STATE_4 = 4
49.  TILE_STATE_5 = 5
50.  TILE_STATE_6 = 6  # M138 or TILE
51.  TILE_STATE_CONFIGURED = 7
52.
53.  tileStateTable = [('$FV',   '$FV 20',              4, TILE_STATE_2,
     TILE_STATE_REBOOTING),  # 0 state
54.                    ('$RS',   f'${DEVTAG} BOOT,RUNNING', 30, TILE_STATE_2,
     TILE_STATE_REBOOTING),  # 1 state
55.                    ('$DT 5', '$DT OK',              4, TILE_STATE_3,
     TILE_STATE_REBOOTING),  # 2 state
56.                    ('$GS 5', '$GS OK',              4, TILE_STATE_4,
     TILE_STATE_REBOOTING),  # 3 state
57.                    ('$GN 5', '$GN OK',              4, TILE_STATE_5,
     TILE_STATE_REBOOTING),  # 4 state
58.                    ('$RT 5', '$RT OK',              4, TILE_STATE_6,
     TILE_STATE_REBOOTING),  # 5 state
59.                    ('$CS', '$CS DI=',               4,
     TILE_STATE_CONFIGURED, TILE_STATE_REBOOTING),  # 5 state
60.                    (None,     None,                 0,
     TILE_STATE_CONFIGURED, TILE_STATE_CONFIGURED)]  # 6 state
61.  tileTimeout = 0.0
62.  tileState = TILE_STATE_UNKNOWN
63.
64.  tcpLine = bytearray(800)
65.  tcpPtr = 0
66.  i2c = None
67.
68.  TCPHOST = ""
69.  TCPPORT = 23
70.  TIMEOUT = None
71.  BACKLOG = 2
72.  MAXBUF = 256
73.  TCPSTATE_LISTENING = 1
74.  TCPSTATE_CONNECTED = 2
75.  TCPSTATE = TCPSTATE_LISTENING
76.  tcplistener = None
```

```
77.  tcpconn = None
78.  pool = None
79.
80.  HTTPHOST = None
81.  HTTPPORT = 80
82.  web_app = None
83.  wsgiServer = None
84.
85.  config = None
86.  display = None
87.  displayLines = []
88.  inaChannel = 1
89.  inaConnected = False
90.  inaData = {1: (None, None), 2: (None, None), 3: (None, None)}
91.
92.  switchA = None
93.  switchC = None
94.
95.  accumulate = ""
96.  inaTime = 0
97.
98.  # Received Signal Strength Indicator threshold values
99.  global RSSI_RED, RSSI_GREEN
100. # These values are default for DN=TILE
101. RSSI_RED = -91
102. RSSI_GREEN = -95
103. pixels = neopixel.NeoPixel(board.IO38, 2, bpp=4, pixel_order=neopixel.GRBW)
104. mdata = []
105.
106. lastGN = None
107. lastDT = None
108. lastRSSI = None
109. nextGPSTime = 0
110. gpsCount = 0
111.
112. helpMessage = '''
113. @set mode <ap, sta>
114. Set wifi mode to access point or station mode.
115.
116. @set wifi <enabled, disabled>
117. Enable or disable wifi functionality.
118.
119. @set ssid <ssid>
120. Set the ssid to connect to in station mode or to create when in ap mode.
121.
122. @set pw <password>
```

```
123. Set the password to connect to in station mode or to create when in ap
     mode.
124.
125. @set interval <minutes>
126. Set the interval for gps location updating. 0 is never. 15-720 minutes.
127.
128. @show
129. Print the wifi mode, ssid, password, and interval to be commited.
130.
131. @show <battery, 3v3, solar>
132. Print the battery, 3v3, and solar voltage and current.
133.
134. @reset
135. Restart the feather.
136.
137. You must @reset for changes to take effect.\n\nVersion ''' + VERSION
138.
139. # Adafruit FeatherWing 128x64 OLED graphic display for the Feather board
140. def displayInit():
141.   displayio.release_displays()
142.   display_bus = displayio.I2CDisplay(i2c, device_address=0x3C)
143.
144.   WIDTH = 128
145.   HEIGHT = 64
146.   BORDER = 0
147.
148.   display = adafruit_displayio_sh1107.SH1107(display_bus, width=WIDTH,
     height=HEIGHT)
149.
150.   # SWARM LOGO
151.   splash = displayio.Group(max_size=10)
152.   splash.y = 16
153.   display.show(splash)
154.   color_palette = displayio.Palette(1)
155.   color_palette[0] = 0xFFFFFF
156.
157.   image_file = open("swarm.bmp", "rb")
158.   image = displayio.OnDiskBitmap(image_file)
159.   image_sprite = displayio.TileGrid(image, pixel_shader=image.pixel_shader)
160.   splash.append(image_sprite)
161.   time.sleep(3)
162.   splash.pop()
163.   STRING = "      Swarm  Eval\n" + VERSION.center(24)
164.   text_area2 = label.Label( terminalio.FONT, text=STRING, scale=1,
     color=0xFFFFFF, x=0, y=3)
165.   splash.append(text_area2)
166.   time.sleep(3)
```

```
167.   splash.pop()
168.   # SWARM LOGO
169.
170.   splash = displayio.Group(max_size=10)
171.   display.show(splash)
172.
173.   color_bitmap = displayio.Bitmap(WIDTH, HEIGHT, 1)
174.   color_palette = displayio.Palette(1)
175.   color_palette[0] = 0xFFFFFF  # White
176.
177.   bg_sprite = displayio.TileGrid(color_bitmap, pixel_shader=color_palette,
      x=0, y=0)
178.   splash.append(bg_sprite)
179.
180.   inner_bitmap = displayio.Bitmap(WIDTH - BORDER * 2, HEIGHT - BORDER * 2,
      1)
181.   inner_palette = displayio.Palette(1)
182.   inner_palette[0] = 0x000000  # Black
183.   inner_sprite = displayio.TileGrid(inner_bitmap,
      pixel_shader=inner_palette, x=BORDER, y=BORDER)
184.   splash.append(inner_sprite)
185.
186.   LINEHEIGHT = 11
187.   LINESTART = 4
188.   for line in range(0, 6):
189.     text_area = label.Label(terminalio.FONT, text=20*" ", color=0xFFFFFF,
      x=0, y=LINESTART + line*LINEHEIGHT)
190.     displayLines.append(text_area)
191.     splash.append(text_area)
192.
193. def displayLine(line, text):
194.   displayLines[line].text = text
195.
196. def makeTileCmd(cmd):
197.   cbytes = cmd.encode()
198.   cs = 0
199.   for c in cbytes[1:]:
200.     cs = cs ^ c
201.   return cbytes + b'*%02X\n'%cs
202.
203. # WIFI CONFIGURATION
204. # The WiFi module provides necessary low-level functionality for managing
205. # wifi connections. "wifi.radio" manages both STA and AP modes.
206.
207. def wifiInit():
208.   if config['wifi'] == 'disabled':
209.     displayLine(0, "Wifi Disabled")
```

```
210.     return
211.   global pool, TCPHOST, mqttStarted
212.   try:
213.     if config['mode'] == 'sta':
214.       displayLine(0, "Connecting to AP...")
215.       wifi.radio.connect(config["ssid"], config["password"])
216.       print("Self IP", wifi.radio.ipv4_address)
217.       displayLine(0, "ST: " + str(wifi.radio.ipv4_address))
218.       pool = socketpool.SocketPool(wifi.radio)
219.       TCPHOST = str(wifi.radio.ipv4_address)
220.
221.       # CONNECT TO MQTT BROKER
222.       mqttInit()
223.
224.     else:
225.       displayLine(0, "Starting AP...")
226.       if(config['ssid'] == 'swarm'): config['ssid'] = 'swarm-' +
    '%02x%02x'%(wifi.radio.mac_address[4], wifi.radio.mac_address[5]) # The
    last two bytes of the ESP32 MAC address
227.       wifi.radio.start_ap(config["ssid"], config["password"])
228.       displayLine(0, "AP: " + str(wifi.radio.ipv4_address_ap))
229.       TCPHOST = str(wifi.radio.ipv4_address_ap)
230.       pool = socketpool.SocketPool(wifi.radio)
231.   except Exception as e:
232.     displayLine(0, "Can't Connect")
233.
234. def tileCheck(line):
235.   global tileTimeout
236.   if  tileStateTable[tileState][1] in line:
237.     tileTimeout = -1.0
238.
239. def tileStart():
240.   global tileState, tileTimeout
241.   displayLine(0, "Connecting to Modem...")
242.   tileState = TILE_STATE_UNKNOWN
243.   while tileState != TILE_STATE_CONFIGURED:
244.     tile.write(b'\n' + makeTileCmd(tileStateTable[tileState][0]))
245.     tileTimeout = time.monotonic() + tileStateTable[tileState][2]
246.     while (tileTimeout > 0.0) and (tileTimeout > time.monotonic()):
247.       tilePoll()
248.       serialPoll()
249.       w.feed()
250.     if tileTimeout  < 0.0:
251.       # Success from state table?
252.       tileState = tileStateTable[tileState][3]
253.     else:
254.       tileState = tileStateTable[tileState][4]
```

```
255.
256. def tileInit():
257.   global tile
258.   tile =
     busio.UART(board.TX,board.RX,baudrate=115200,receiver_buffer_size=8192,time
     out=0.0)
259.   tileStart()
260.
261. def tileParseLine(line):
262.   print(line)
263.   global lastDT, lastGN, lastRSSI
264.   if len(line) < 4:
265.     return
266.   if line[len(line) - 3] != '*':
267.     return
268.   cksum1 = 0
269.   cksum2 = int(line[-2:], 16)
270.   for c in line[1:-3]:
271.     cksum1 = cksum1 ^ ord(c)
272.   if cksum1 != cksum2:
273.     return
274.   if tileState != TILE_STATE_CONFIGURED:
275.     tileCheck(line)
276.     #return
277.   if line[0:3] == "$TD":
278.     if len(mdata) > 10:
279.       mdata.pop(0)
280.     mdata.append(line)
281.   if line[0:3] == "$DT":
282.     if line == "$DT OK*34":
283.       lastDT = None
284.     else:
285.       lastDT = line
286.   if line[0:3] == "$GN":
287.     if line == "$GN OK*2d":
288.         lastGN = None
289.     else:
290.         lastGN = line
291.   parse = line[:-3].split(' ')
292.   if parse[0] == "$CS":
293.     if ',' in parse[1]:
294.       cs_params = parse[1].split(',')
295.       for param in cs_params:
296.           k, v = param.split('=')
297.           if k == "DN":
298.               global DEVTAG, RSSI_RED, RSSI_GREEN
299.               DEVTAG = v
```

```
300.                # Here is where M138 vs Tile params can be set
301.   if parse[0] == '$RT':
302.     if 'RSSI' in parse[1]:
303.       if ',' in parse[1]:
304.         rdata = line[4:-3].split(',')
305.         rtdata = []
306.         for r in rdata:
307.           rtdata.append(r.split('='))
308.         rtdata = dict(rtdata)
309.         if 'T' in rtdata['TS']:
310.             d, t = rtdata['TS'].split('T')
311.         else:
312.             d, t = rtdata['TS'].split(' ')
313.         d = d.split('-')
314.         t = t.split(':')
315.         dtString = d[0][2:]+d[1]+d[2]+'T'+t[0]+t[1]+t[2]
316.         print(rtdata)
317.         displayLine(4, dtString + ' S' + rtdata['DI'][2:])
318.         displayLine(5, 'R:' + rtdata['RSSI'] + ' S:' + rtdata['SNR'] + '
    F:' + rtdata['FDEV'])
319.       else:
320.         rssi = parse[1].split('=')
321.         displayLine(2, "RSSI: " + rssi[1])
322.
323.         irssi = int(rssi[1])
324.         lastRSSI = irssi
325.
326.         if config['wifi'] == 'enabled':
327.           if irssi > RSSI_RED: # -91
328.             pixels[0] = (16, 0, 0, 0)
329.           elif irssi < RSSI_GREEN:  # -95
330.             pixels[0] = (0, 16, 0, 0)
331.           else:
332.             pixels[0] = (16, 16, 0, 0)
333.           pixels.write()
334.
335. def tilePoll():
336.   global tilePtr
337.   chars = tile.read(20)
338.   if chars == None:
339.     return
340.   if tcpconn != None:
341.     try:
342.       x = tcpconn.send(chars)
343.     except:
344.       pass
345.
```

```
346.   for c in chars:
347.     if c == 0x0A:
348.       tileParseLine(tileLine[:tilePtr].decode())
349.       tilePtr = 0
350.     elif c == 0x08 and tilePtr != 0:
351.       tilePtr = tilePtr - 1
352.     elif c >= 0x20 and c <= 0x7f and tilePtr < len(tileLine):
353.       tileLine[tilePtr] = c
354.       tilePtr = tilePtr + 1
355.   pass
356.
357. # INA3221 3-channel DC current sensor
358.
359. def inaInit():
360.   global ina3221, inaConnected
361.   try:
362.     ina3221 = INA3221(i2c, shunt_resistor = (0.01, 0.01, 0.01))
363.     ina3221.update(reg=C_REG_CONFIG, mask=C_AVERAGING_MASK |
    C_VBUS_CONV_TIME_MASK | C_SHUNT_CONV_TIME_MASK | C_MODE_MASK,
364.                                         value=C_AVERAGING_128_SAMPLES |
    C_VBUS_CONV_TIME_8MS | C_SHUNT_CONV_TIME_8MS |
    C_MODE_SHUNT_AND_BUS_CONTINOUS)
365.     # Setup the 3 channels
366.     ina3221.enable_channel(1) # Battery
367.     ina3221.enable_channel(2) # Solar Panel
368.     ina3221.enable_channel(3) # GPIO
369.     inaConnected = True
370.   except:
371.     displayLine(1, "ina disconnected")
372.     inaConnected = False
373.
374. def inaPoll():
375.   global inaChannel, inaTime, inaConnected, inaData
376.   if not inaConnected:
377.     inaInit()
378.     return
379.   if time.time() - inaTime > 5: # Updated each 5 s.
380.     try:
381.       inaChans = {1:'BAT:', 2:'SOL:', 3:'3V3:'}
382.       bus_voltage = ina3221.bus_voltage(inaChannel)
383.       current = ina3221.current(inaChannel)
384.
385.       displayLine(1, "%s %6.3fV %6.3fA"%(inaChans[inaChannel], bus_voltage,
    current))
386.       inaData[inaChannel] = (bus_voltage, current)
387.       # Channel looping
388.       inaChannel = inaChannel + 1
```

```
389.       if inaChannel == 4:
390.         inaChannel = 1
391.     except:
392.       inaConnected = False
393.     inaTime = time.time()
394.
395. def tcpInit():
396.   if config['wifi'] == 'disabled':
397.     return
398.   if wifi.radio.ipv4_address_ap is None and wifi.radio.ipv4_address is
     None:
399.     return
400.   global TCPSTATE, TCPHOST, tcplistener, tcpconn
401.   print("Create TCP Server socket", (TCPHOST, TCPPORT))
402.   tcplistener = pool.socket(pool.AF_INET, pool.SOCK_STREAM)
403.   tcplistener.settimeout(TIMEOUT)
404.   tcplistener.setblocking(False)
405.   tcplistener.bind((TCPHOST, TCPPORT))
406.   tcplistener.listen(BACKLOG)
407.   print("Listening")
408.
409. # Read FeatherS2 commands via TCP connection
410.
411. def tcpPoll():
412.   if config['wifi'] == 'disabled' or (wifi.radio.ipv4_address_ap is None
     and wifi.radio.ipv4_address is None):
413.     return
414.   global TCPSTATE, tcplistener, tcpconn, tcpPtr
415.   if TCPSTATE == TCPSTATE_LISTENING:
416.     try:
417.       tcpconn, addr = tcplistener.accept()
418.       tcpconn.settimeout(0)
419.       print("Accepted from", addr)
420.       TCPSTATE = TCPSTATE_CONNECTED
421.     except:
422.       pass
423.   elif TCPSTATE == TCPSTATE_CONNECTED:
424.     buf = bytearray(MAXBUF)
425.     try:
426.       size = tcpconn.recv_into(buf, MAXBUF)
427.       if size == 0:
428.         tcpconn.close()
429.         tcpconn = None
430.         print("Accepting connections")
431.         TCPSTATE = TCPSTATE_LISTENING
432.       else:
433.         print("Received", buf[:size], size, "bytes")
```

```
434.         for i in range(size):
435.           if buf[i] == 0x0A:
436.             if tcpLine[0] == 0x40:
437.               command = tcpLine[:tcpPtr].decode()
438.               params = command.split(' ')
439.               if params[0] == '@reset':
440.                 tcpconn.send("Resetting...")
441.                 microcontroller.reset()
442.               elif params[0] == '@color':
443.                 if len(params) ==  5:
444.                   if config['wifi'] == 'enabled':
445.                     pixels[1] =
    (int(params[1]),int(params[2]),int(params[3]),int(params[4]))
446.                     pixels.write()
447.               elif params[0] == '@set':
448.                 if params[1] == 'mode':
449.                   if params[2] in ['ap', 'sta']:
450.                     config['mode'] = params[2]
451.                     tcpconn.send(f"Successfully set mode to {params[2]}.")
452.                     writePreferences()
453.                 if params[1] == 'wifi':
454.                   if params[2] in ['enabled', 'disabled']:
455.                     config['wifi'] = params[2]
456.                     if config['wifi'] == 'disabled':
457.                       pixels[0] = (0,0,0,0)
458.                       pixels[1] = (0,0,0,0)
459.                       pixels.write()
460.                     tcpconn.send(f"Successfully {params[2]} wifi.")
461.                     tcpconn.send("Resetting...")
462.                     microcontroller.reset()
463.                     writePreferences()
464.                 if params[1] == 'ssid':
465.                   config['ssid'] = command[10:].strip()
466.                   tcpconn.send(f"Successfully set ssid to
    {config['ssid']}.")
467.                   writePreferences()
468.                 if params[1] == 'pw':
469.                   config['password'] = command[8:].strip()
470.                   tcpconn.send(f"Successfully set password to
    {config['password']}.")
471.                   writePreferences()
472.                 if params[1] == 'interval':
473.                   if int(params[2]) == 0 or (int(params[2]) >= 15 and
    int(params[2]) <= 720):
474.                     if int(params[2]) == 0 and config['interval'] > 0:
475.                       config['interval'] = config['interval'] * -1
476.                       tcpconn.send(f"Successfully set interval to off.")
```

```
477.                     else:
478.                       config['interval'] = int(params[2])
479.                       tcpconn.send(f"Successfully set interval to
    {config['interval']}.")
480.                     gpsInit()
481.                     writePreferences()
482.                   else:
483.                     tcpconn.send("Interval can only be 0 or 15-720
    minutes.")
484.               elif params[0] == '@show':
485.                 if len(params) == 2:
486.                   if params[1] == 'battery':
487.                     tcpconn.send('BAT: ' + str(inaData[1][0]) + 'V ' +
    str(inaData[1][1]) + 'A')
488.                   if params[1] == '3v3':
489.                     tcpconn.send('3V3: ' + str(inaData[3][0]) + 'V ' +
    str(inaData[3][1]) + 'A')
490.                   if params[1] == 'solar':
491.                     tcpconn.send('SOL: ' + str(inaData[2][0]) + 'V ' +
    str(inaData[2][1]) + 'A')
492.                 else:
493.                   tcpconn.send('wifi mode:' + config['mode'] + '\n')
494.                   tcpconn.send('wifi:' + config['wifi'] + '\n')
495.                   tcpconn.send('wifi ssid:' + config['ssid'] + '\n')
496.                   tcpconn.send('wifi pw:  ' + config['password'] + '\n')
497.                   tcpconn.send('gps interval: ' + (str(config['interval']),
    'OFF')[config['interval'] <= 0] + '\n')
498.               elif params[0] == '@factory':
499.                 microcontroller.nvm[0] = 0
500.                 tcpconn.send("Cleared NVM and Resetting...")
501.                 microcontroller.reset()
502.               elif params[0] == '@help':
503.                 tcpconn.send(helpMessage)
504.               else:
505.                 tcpconn.send("Invalid command. Type @help for help.")
506.               print("", end='')
507.             tile.write(tcpLine[:tcpPtr])
508.             tile.write(bytearray([0x0a]))
509.             tcpPtr = 0
510.           elif buf[i] == 0x08 and tcpPtr != 0:
511.             tcpPtr = tcpPtr - 1
512.           elif buf[i] >= 0x20 and buf[i] <= 0x7f and tcpPtr < len(tcpLine):
513.             tcpLine[tcpPtr] = buf[i]
514.             tcpPtr = tcpPtr + 1
515.     except Exception as e:
516.       pass
517.
```

```
518. def serialInit():
519.   print("", end='')
520.
521. # Read FeatherS2 commands via serial port
522.
523. def serialPoll():
524.   global accumulate
525.   if supervisor.runtime.serial_bytes_available:
526.     accumulate += sys.stdin.read(1)
527.   if "\n" in accumulate:
528.     accumulate = accumulate[:-1]
529.     params = accumulate.split(' ')
530.     if params[0] == '@reset':
531.       print("Resetting...")
532.       microcontroller.reset()
533.     elif params[0] == '@color':
534.       if len(params) ==  5:
535.         if config['wifi'] == 'enabled':
536.           pixels[1] =
    (int(params[1]),int(params[2]),int(params[3]),int(params[4]))
537.           pixels.write()
538.     elif params[0] == '@set':
539.       if params[1] == 'mode':
540.         if params[2] in ['ap', 'sta']:
541.           config['mode'] = params[2]
542.           print(f"Successfully set mode to {params[2]}.")
543.           writePreferences()
544.       if params[1] == 'wifi':
545.         if params[2] in ['enabled', 'disabled']:
546.           config['wifi'] = params[2]
547.           if config['wifi'] == 'disabled':
548.             pixels[0] = (0,0,0,0)
549.             pixels[1] = (0,0,0,0)
550.             pixels.write()
551.           print(f"Successfully {params[2]} wifi.")
552.           writePreferences()
553.           print("Resetting...")
554.           microcontroller.reset()
555.       if params[1] == 'ssid':
556.         config['ssid'] = accumulate[10:].strip()
557.         print(f"Successfully set ssid to {config['ssid']}.")
558.         writePreferences()
559.       if params[1] == 'pw':
560.         config['password'] = accumulate[8:].strip()
561.         print(f"Successfully set password to {config['password']}.")
562.         writePreferences()
563.       if params[1] == 'interval':
```

```
564.         if int(params[2]) == 0 or (int(params[2]) >= 15 and int(params[2])
     <= 720):
565.           if int(params[2]) == 0 and config['interval'] > 0:
566.             config['interval'] = config['interval'] * -1
567.             print(f"Successfully set interval to off.")
568.           else:
569.             config['interval'] = int(params[2])
570.             print(f"Successfully set interval to {config['interval']}.")
571.           gpsInit()
572.           writePreferences()
573.         else:
574.           print("Interval can only be 0 or 15-720 minutes.")
575.     elif params[0] == '@show':
576.       if len(params) == 2:
577.         if params[1] == 'battery':
578.           print('BAT: ' + str(inaData[1][0]) + 'V ' + str(inaData[1][1]) +
     'A')
579.         if params[1] == '3v3':
580.           print('3V3: ' + str(inaData[3][0]) + 'V ' + str(inaData[3][1]) +
     'A')
581.         if params[1] == 'solar':
582.           print('SOL: ' + str(inaData[2][0]) + 'V ' + str(inaData[2][1]) +
     'A')
583.       else:
584.         print('wifi mode:', config['mode'])
585.         print('wifi:', config['wifi'])
586.         print('wifi ssid:', config['ssid'])
587.         print('wifi pw:  ', config['password'])
588.         print('gps interval: ' + (str(config['interval']),
     'OFF')[config['interval'] <= 0] + '\n')
589.     elif params[0] == '@factory':
590.       microcontroller.nvm[0] = 0
591.       print("Cleared NVM and Resetting...")
592.       microcontroller.reset()
593.     elif params[0] == '@test':
594.       tileParseLine(' '.join(params[1:]))
595.     elif params[0] == '@help':
596.       print(helpMessage)
597.
598.     else:
599.       print("Invalid command. Type @help for help.")
600.     print("", end='')
601.     accumulate = ""
602.
603. def httpInit():
604.   if config['wifi'] == 'disabled':
605.     return
```

```
606.   if wifi.radio.ipv4_address_ap is None and wifi.radio.ipv4_address is
     None:
607.     return
608.   global web_app, wsgiServer, HTTPHOST
609.   if config['mode'] is 'ap':
610.     HTTPHOST = repr(wifi.radio.ipv4_address_ap)
611.   else:
612.     HTTPHOST = repr(wifi.radio.ipv4_address)
613.   web_app = WSGIApp()
614.
615.   # Open email web app .html file
616.   @web_app.route("/")
617.   def on_connect(request):
618.     f = open("index.html", "r")
619.     return ("200 OK", ["Content-Type: text/html"], f.read())
620.
621.   # Open Swarm logo file
622.   @web_app.route("/logo.png")
623.   def on_connect(request):
624.     f = open("logo.png", "r")
625.     return ("200 OK", ["Content-Type: image/png"], f.read())
626.
627.   @web_app.route("/msgsend")
628.   def on_msg(request):
629.     print(request)
630.     qs = request.query_params
631.     d = {"i": 1, "t": urlDecode(qs["user_to"]), "f":
     urlDecode(qs["user_from"]), "s": urlDecode(qs["user_subject"]), "m":
     urlDecode(qs["user_message"])}
632.     s = json.dumps(d)
633.     h = b'$TD AI=65000,' + hexlify(s.encode())
634.     cs = 0
635.     for c in h[1:]:
636.       cs = cs ^ c
637.     h = h + b'*%02X\n'%cs
638.     print(h)
639.     print(d)
640.     tile.write(h)
641.     return ("204 NO CONTENT", [], [])
642.
643.   @web_app.route("/mdata")
644.   def on_msg(request):
645.     global mdata
646.     mdata.insert(0, '...' if lastRSSI is None else str(lastRSSI))
647.     response = "\n".join(mdata)
648.     mdata = []
649.     return ("200 OK", [], response)
```

```
650.
651.   print(f"open this IP in your browser: http://{HTTPHOST}:{HTTPPORT}/")
652.   wsgiServer = server.WSGIServer(80, application=web_app)
653.   wsgiServer.start()
654.
655. def httppoll():
656.   if config['wifi'] == 'disabled':
657.     return
658.   if wifi.radio.ipv4_address_ap is None and wifi.radio.ipv4_address is
     None:
659.     return
660.   wsgiServer.update_poll()
661.
662. # Decode percent-encoding. Add reserved characters after percent-encoding
663. def urlDecode(s):
664.   i = 0
665.   r = ''
666.   while i < len(s):
667.     if s[i] == '+':
668.       r = r + ' '
669.     elif s[i] == '%':
670.       r = r + chr(int(s[i+1:i+3], 16))
671.       i = i + 2
672.     else:
673.       r = r + s[i]
674.     i = i + 1
675.   return r
676.
677. def gpsInit():
678.   global sentQuery
679.   sentQuery = False
680.   displayLine(3, 'GPS Ping: ' + (str(config['interval']) + 'min',
     'OFF')[config['interval'] <= 0])
681.
682. def gpspoll():
683.   global nextGPSTime,gpsCount,sentQuery
684.   global lastGN, lastDT
685.   if config['interval'] > 0:
686.       if time.time() > nextGPSTime:
687.           if lastGN is not None and lastDT is not None:
688.               gpsObj = {}
689.               gn = lastGN[4:-3].split(',')
690.               s = lastDT
691.               gpsObj['d'] = 946684800 + time.mktime((int(s[4:8]),
     int(s[8:10]), int(s[10:12]), int(s[12:14]), int(s[14:16]), int(s[16:18]), -
     1, -1, -1))
692.               gpsObj['lt'] = float(gn[0]) # Latitude
```

```
693.                gpsObj['ln'] = float(gn[1]) # Longitude
694.                gpsObj['a']  = float(gn[2]) # Altitude
695.                gpsObj['c']  = float(gn[3]) # Course (degrees)
696.                gpsObj['s']  = float(gn[4]) # Speed (km/h)
697.                gpsObj['n']  = gpsCount     # Number of messages sent
698.                gpsObj['si'] = inaData[2][1]# Solar panel current (A)
699.                gpsObj['sv'] = inaData[2][0]# Solar panel voltage (V)
700.                gpsObj['bi'] = inaData[1][1]# Battery current (A)
701.                gpsObj['bv'] = inaData[1][0]# Battery voltage (V)
702.                gpsObj['ti'] = inaData[3][1]# Modem current
703.                gpsObj['r'] = lastRSSI
704.                gpsCount += 1
705.                s = json.dumps(gpsObj)
706.                s = s.replace(' ', '')
707.                h = b'$TD AI=65535,' + hexlify(s.encode())
708.                cs = 0
709.                for c in h[1:]:
710.                  cs = cs ^ c
711.                h = h + b'*%02X\n'%cs
712.                tile.write(h)
713.                nextGPSTime = config['interval'] * 60 + time.time()
714.
715.                lastGN = None
716.                lastDT = None
717.                sentQuery = False  # reset query for next ping
718.            else:
719.                # Force send of $DT @ to update lastDT
720.                if not sentQuery:
721.                    tile.write(makeTileCmd("$DT @"))
722.                    tile.write(makeTileCmd("$GN @"))
723.                    sentQuery = True
724.
725. def writePreferences():
726.   configString = json.dumps(config)
727.   ba = bytearray(configString, 'utf-8')
728.   microcontroller.nvm[0:len(ba)] = ba
729.   microcontroller.nvm[len(ba)] = 0
730.
731. def readPreferences():
732.   global config
733.   try:
734.     x = microcontroller.nvm[0]
735.   except:
736.     microcontroller.nvm[0] = 0
737.   i = 0
738.   configString = ""
739.   while microcontroller.nvm[i] is not 0:
```

```
740.     configString += chr(microcontroller.nvm[i])
741.     i = i + 1
742.   if configString == "":
743.     configString = "{}"
744.   config = json.loads(configString)
745.   if not 'mode' in config:
746.     config['mode'] = 'ap'
747.   if not 'ssid' in config:
748.     config['ssid'] = 'swarm'
749.   if not 'password' in config:
750.     config['password'] = '12345678'
751.   if not 'interval' in config:
752.     config['interval'] = 60
753.   if not 'wifi' in config:
754.     config['wifi'] = "enabled"
755.
756. def watchDogInit():
757.   w.timeout = 60
758.   w.mode = WatchDogMode.RESET
759.   w.feed()
760.
761. def buttonInit():
762.   global switchA, switchC
763.
764.   pinA = digitalio.DigitalInOut(board.D5)
765.   pinA.direction = digitalio.Direction.INPUT
766.   pinA.pull = digitalio.Pull.UP
767.   switchA = Debouncer(pinA)
768.
769.   pinC = digitalio.DigitalInOut(board.D20)
770.   pinC.direction = digitalio.Direction.INPUT
771.   pinC.pull = digitalio.Pull.UP
772.   switchC = Debouncer(pinC)
773.
774.
775. def buttonPoll():
776.   switchA.update()
777.   if switchA.rose: # just released
778.     if config['wifi'] == "enabled":
779.       config['wifi'] = "disabled"
780.       pixels[0] = (0,0,0,0)
781.       pixels[1] = (0,0,0,0)
782.       pixels.write()
783.     else:
784.       config['wifi'] = "enabled"
785.     writePreferences()
786.     print(f"Successfully {config['wifi']} wifi.")
```

```
787.     print("Resetting...")
788.     microcontroller.reset()
789.
790.   switchC.update()
791.   if switchC.rose:
792.     config['interval'] = config['interval'] * -1
793.     writePreferences()
794.     gpsInit()
795.
796.   # Update wifi RSSI LED and oled
797.   if config['wifi'] == "enabled":
798.     if config['mode'] == "sta" and hasattr(wifi.radio.ap_info, "rssi"):
799.       rssi = wifi.radio.ap_info.rssi
800.       displayLine(0, "ST: " + str(wifi.radio.ipv4_address) + f" ({rssi})")
801.       if rssi > -45:
802.         pixels[1] = (0, 16, 0, 0)
803.       elif rssi < -67:
804.         pixels[1] = (16, 0, 0, 0)
805.       else:
806.         pixels[1] = (16, 16, 0, 0)
807.       pixels.write()
808.
809. def factoryResetCheck():
810.   switchA.update()
811.   if not switchA.value:
812.     microcontroller.nvm[0] = 0
813.     while not switchA.value:
814.       switchA.update()
815.     print("Cleared NVM and Resetting...")
816.     microcontroller.reset()
817.
818. watchDogInit()
819. buttonInit()
820. factoryResetCheck()
821. i2c = busio.I2C(board.SCL, board.SDA, frequency=400000)
822. # i2c = board.I2C()
823. displayInit()
824. readPreferences()
825. if config['wifi'] == 'enabled':
826.   import wifi
827.   import socketpool
828.   import ipaddress
829.   import wsgiserver as server
830.   from adafruit_wsgi.wsgi_app import WSGIApp
831.
832.   # MQTT LIBRARIES
833.   import ssl
```

```
834.   import adafruit_minimqtt as MQTT
835.   from secrets import secrets
836.
837.   application_feed = "application/1/device/+/rx" # Suscribe to all devices
     in application 1
838.
839.   def connected(client, userdata, flags, rc):
840.     global mqttStarted
841.     displayLine(3,"Connected to Broker!")
842.     client.subscribe(application_feed)
843.     displayLine(4,"Listening...")
844.     mqttStarted = True
845.
846.   def disconnected(client, userdata, rc):
847.     displayLine(3,"Disconnected from broker!")
848.
849.   def new_message(client, topic, message_string):
850.
851.     # 0 - Check if the string is valid JSON
852.
853.     if is_json(message_string):
854.
855.       # 1 - Convert data type from string to dictionary
856.
857.       message_dictionary = json.loads(message_string)
858.       msg = message_dictionary
859.
860.       # 2 - Shorten message
861.       # To avoid raising a KeyError, delete the following keys form the
     dictionary only if they exist, using the two-argument form of
     dictonary.pop():
862.
863.       msg.pop('applicationName', None)
864.       msg.pop('applicationID', None)
865.       msg.pop('data_encode', None)
866.       msg.pop('fPort', None)
867.       msg.pop('fCnt', None)
868.
869.       # 3 - Convert data type from dictionary to JSON
870.
871.       jsonObj = json.dumps(msg)
872.
873.       # 4 - Delete whitespace characters
874.
875.       jsonObj = jsonObj.replace(' ','') # *
876.
877.       # 5 - Check message length
```

```
878.
879.       payload =  len(jsonObj)
880.
881.       if payload > 192:
882.         tcpconn.send("Message too long!")
883.         return
884.
885.       # 6 - Add header and checksum
886.
887.       command = b'$TD ' + hexlify(jsonObj.encode())
888.     else:
889.       payload =  len(message_string)
890.
891.       command = b'$TD ' + hexlify(message_string.encode())
892.
893.     checksum = 0
894.
895.     for i in command[1:]:
896.         checksum = checksum ^ i
897.
898.     command = command + b'*%02X\n'%checksum
899.     tile.write(command)
900.
901.     if TCPSTATE == TCPSTATE_CONNECTED:
902.       tcpconn.send("\nNew message on topic:" + str (topic) + "\n")
903.       tcpconn.send("\nOriginal string:\n")
904.       tcpconn.send(message_string)
905.       tcpconn.send("\nPayload:" + str (payload) + " Bytes\n")
906.       tcpconn.send("\nM138 Transmit Data Command ($TD):\n")
907.       tcpconn.send(command)
908.
909.     displayLine(4, "New message! (" + str(payload) + "Bytes)")
910.     displayLine(5, message_string)
911.
912.   def mqttInit():
913.     displayLine(3, "MQTT CONNECTION")
914.     global mqtt_client
915.     mqtt_client = MQTT.MQTT(
916.       broker=secrets["broker"],
917.       port=secrets["port"],
918.       socket_pool=pool,
919.       ssl_context=ssl.create_default_context(),
920.     )
921.
922.     mqtt_client.on_connect = connected
923.     mqtt_client.on_disconnect = disconnected
924.     mqtt_client.on_message = new_message
```

```
925.
926.     displayLine(4, "Connecting to Broker...")
927.     mqtt_client.connect()
928.
929.   def mqttPoll():
930.     if mqttStarted == True:
931.       global mqtt_client
932.       mqtt_client.loop()
933.
934.   def is_json(myjson):
935.     try:
936.       json.loads(myjson)
937.     except ValueError as e:
938.       return False
939.     return True
940.
941. inaInit()
942. serialInit()
943. tileInit()
944. wifiInit()
945. tcpInit()
946. httpInit()
947. gpsInit()
948.
949. try:
950.   while True: # Superloop
951.     tilePoll()
952.     inaPoll()
953.     gpspoll()
954.     serialPoll()
955.     tcpPoll()
956.     httppoll()
957.     buttonPoll()
958.     mqttPoll()
959.     w.feed()
960.     gc.collect()
961.
962. except Exception as e:
963.   print(e)
964.   print("Resetting...")
965.   microcontroller.reset()
```

2. Decodificador de los sensores CollectionCare

Seguidamente se adjunta la función utilizada para decodificar el *payload* de los sensores CollectionCare en el flujo de Node-RED. Se ha implementado en un nodo función (*Function Node*) que permite ejecutar código JavaScript sobre los mensajes de entrada.

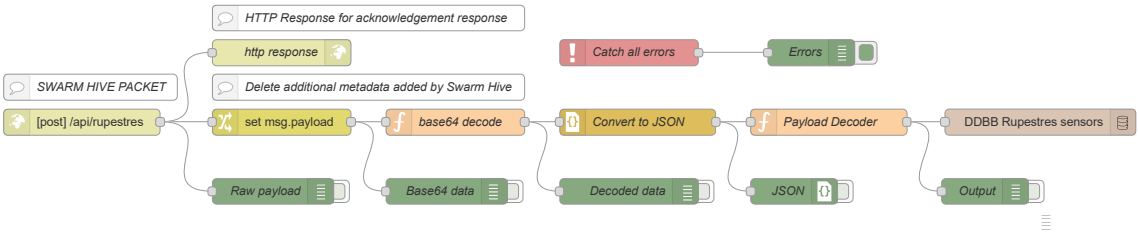
```
1.  //Parsing bytes from payload
2.  var datajson = msg.payload;
3.  var data = JSON.stringify(datajson.data).replace(/"/g, '');
4.  console.log(typeof(data) + ' ' + data);
5.
6.  var bytes = Buffer.from(data, 'hex');
7.  console.log(bytes);
8.
9.  if (datajson.deviceName == 'CC1') {
10.     // Dragino
11.     var decoded_payload = DecodeDragino(2, bytes);
12. } else if (datajson.deviceName == 'CC2') {
13.     var decoded_payload = DecodeCCNode(2, bytes);
14. }
15.
16. msg.payload = {
17.     ...msg.payload,
18.     ...decoded_payload
19. };
20.
21. return msg;
22.
23. //-----------------------///
24.
25. // CCNode //
26. function DecodeCCNode(fPort, bytes) {
27.     var decoded = {};
28.     var offset = 0;
29.     var message_type = 0;
30.
31.     if (offset + 1 <= bytes.length) {
32.         // Sequence (1 bytes)
33.         decoded['sequence'] = bytes[offset++];
34.
35.         // Message Type (1 bytes)
36.         message_type = bytes[offset++];
37.         decoded['message_type'] = message_type;
38.
39.         // Compressed message_type
40.         if (message_type == 0x03) {
41.             // Message type 3 -> Compressed (Advanced CCNode Sigfox)
42.
43.             // Temperature
```

```
44.             var temperature = ((bytes[offset++] << 5) + ((bytes[offset] >>
    3) & 0x1F)) / 100.0 - 25.0;
45.             decoded['temperature'] = temperature;
46.
47.             // Relative Humidity
48.             var humidity = (((bytes[offset++] & 0x07) << (10 - 3)) +
    ((bytes[offset] >> 1) & 0x7F)) / 10.0;
49.             decoded['humidity'] = humidity;
50.
51.             //Lux
52.             if (bytes[offset++] & 0x01) {
53.                 // Lux > 8000
54.                 var lux = ((bytes[offset++] << 5) + ((bytes[offset] >> 3) &
    0x1F)) * 10;
55.             } else {
56.                 // Lux < 8000
57.                 var lux = (bytes[offset++] << 5) + ((bytes[offset] >> 3) &
    0x1F);
58.             }
59.             decoded['lux'] = lux;
60.
61.
62.             //UVA
63.             var uva = ((bytes[offset++] & 0x07) << (10 - 3)) +
    ((bytes[offset] >> 1) & 0x7F);
64.             decoded['uva'] = uva;
65.         }
66.     }
67.     return decoded;
68. }
69.
70. // DRAGINO//
71.
72. function str_pad(byte) {
73.     var zero = '0';
74.     var hex = byte.toString(16);
75.     var tmp = 2 - hex.length;
76.     return zero.substr(0, tmp) + hex;
77. }
78.
79. function DecodeDragino(port, bytes) {
80.     var decode = {};
81.     if (port == 2) {
82.         if (bytes.length == 11) {
83.             decode.TempC_SHT = parseFloat(((bytes[0] << 24 >> 16 |
    bytes[1]) / 100).toFixed(2));
84.             decode.Hum_SHT = parseFloat(((bytes[2] << 24 >> 16 | bytes[3])
    / 10).toFixed(1));
85.             decode.TempC_DS = parseFloat(((bytes[4] << 24 >> 16 | bytes[5])
    / 100).toFixed(2));
86.
87.             decode.Ext = bytes[6];
```

```
88.             decode.Systimestamp = (bytes[7] << 24 | bytes[8] << 16 |
    bytes[9] << 8 | bytes[10]);
89.
90.             return decode;
91.         } else {
92.             decode.Status = "RPL data or sensor reset";
93.
94.             return decode;
95.         }
96.     }
97.
98.     if (port == 3) {
99.         decode.Status = "Data retrieved, your need to parse it by the
    application server";
100.
101.        return decode;
102.    }
103.
104.    if (port == 4) {
105.        decode.DS18B20_ID = str_pad(bytes[0]) + str_pad(bytes[1]) +
    str_pad(bytes[2]) + str_pad(bytes[3]) + str_pad(bytes[4]) +
    str_pad(bytes[5]) + str_pad(bytes[6]) + str_pad(bytes[7]);
106.
107.        return decode;
108.    }
109.
110.    if (port == 5) {
111.        decode.Sensor_Model = bytes[0];
112.        decode.Firmware_Version = str_pad((bytes[1] << 8) | bytes[2]);
113.        decode.Freq_Band = bytes[3];
114.        decode.Sub_Band = bytes[4];
115.        decode.Bat_mV = bytes[5] << 8 | bytes[6];
116.
117.        return decode;
118.    }
119. }
120.
```

3. Listado de código de Node-RED

El flujo creado en el editor de Node-RED se ha exportado a formato JSON. Nótese que incluye una estructura condicional para el tratamiento de mensajes provenientes de un nodo sensor Dragino LHT52 utilizado en los ensayos prácticos previos al despliegue.



Fig, 1: Flujo en Node-RED (Gráfico vectorial ampliable)

```
[
  {
    "id": "c79b78c8.a3c098",
    "type": "tab",
    "label": "Pinturas Rupestres",
    "disabled": false,
    "info": ""
  },
  {
    "id": "52e62626.a45768",
    "type": "http in",
    "z": "c79b78c8.a3c098",
    "name": "",
    "url": "/api/rupestres",
    "method": "post",
    "upload": false,
    "swaggerDoc": "",
    "x": 290,
    "y": 440,
    "wires": [
      [
        "dc72c3b0.d695d",
        "ef3a70f1.bbe3d"
      ]
    ]
  },
  {
    "id": "ef3a70f1.bbe3d",
    "type": "http response",
    "z": "c79b78c8.a3c098",
    "name": "http response",
    "statusCode": "",
    "headers": {},
    "x": 540,
    "y": 360,
```

```
36.       "wires": []
37.     },
38.     {
39.       "id": "6bddbaf3.6781b4",
40.       "type": "inject",
41.       "z": "c79b78c8.a3c098",
42.       "name": "Swarm Hive Test Packet",
43.       "topic": "",
44.       "payload":
    "{\"packetId\":0,\"deviceType\":1,\"deviceId\":0,\"userApplicationId\":0,\"
    organizationId\":65507,\"data\":\"VGhpcyBpcyBhIHRlc3QgbWVzc2FnZS4gVGhlIHBhY
    2tldElkIGFuZCBkZXZpY2VJZCBhcmUgbm90IHBvcHVsYXRlZCwgYnV0IHdpbGwgYmUgZm9yIGEg
    cmVhbCBtZXNzYWdlLg==\",\"len\":100,\"status\":0,\"hiveRxTime\":\"2022-07-
    06T08:37:08\"}",
45.       "payloadType": "json",
46.       "repeat": "",
47.       "crontab": "",
48.       "once": false,
49.       "onceDelay": 0.1,
50.       "x": 290,
51.       "y": 340,
52.       "wires": [
53.         [
54.           "dc72c3b0.d695d"
55.         ]
56.       ]
57.     },
58.     {
59.       "id": "64712a45.d3a514",
60.       "type": "debug",
61.       "z": "c79b78c8.a3c098",
62.       "name": "raw payload",
63.       "active": false,
64.       "tosidebar": true,
65.       "console": false,
66.       "tostatus": false,
67.       "complete": "payload",
68.       "targetType": "msg",
69.       "x": 490,
70.       "y": 640,
71.       "wires": []
72.     },
73.     {
74.       "id": "9f8767a7.13f158",
75.       "type": "function",
76.       "z": "c79b78c8.a3c098",
77.       "name": "base64 decode",
78.       "func": "msg.payload = Buffer.from(msg.payload,
    'base64').toString()\n\nreturn msg\n\n",
79.       "outputs": 1,
80.       "noerr": 0,
81.       "x": 820,
82.       "y": 440,
```

```
83.       "wires": [
84.         [
85.           "87b723ea.66a5c",
86.           "ec2cd494.c9d468"
87.         ]
88.       ]
89.     },
90.     {
91.       "id": "dc72c3b0.d695d",
92.       "type": "change",
93.       "z": "c79b78c8.a3c098",
94.       "name": "",
95.       "rules": [
96.         {
97.           "t": "set",
98.           "p": "payload",
99.           "pt": "msg",
100.          "to": "payload.data",
101.          "tot": "msg"
102.        }
103.      ],
104.      "action": "",
105.      "property": "",
106.      "from": "",
107.      "to": "",
108.      "reg": false,
109.      "x": 540,
110.      "y": 440,
111.      "wires": [
112.        [
113.          "a16b8b9a.0cb578",
114.          "9f8767a7.13f158"
115.        ]
116.      ]
117.    },
118.    {
119.      "id": "a16b8b9a.0cb578",
120.      "type": "debug",
121.      "z": "c79b78c8.a3c098",
122.      "name": "base64 encoded data",
123.      "active": false,
124.      "tosidebar": true,
125.      "console": false,
126.      "tostatus": false,
127.      "complete": "payload",
128.      "targetType": "msg",
129.      "x": 760,
130.      "y": 640,
131.      "wires": [],
132.      "icon": "node-red-dashboard/ui_numeric.png"
133.    },
134.    {
135.      "id": "cf551b29.26c5c8",
```

```
136.     "type": "inject",
137.     "z": "c79b78c8.a3c098",
138.     "name": "Dragino LHT52 Node",
139.     "topic": "",
140.     "payload":
"{\"packetId\":0,\"deviceType\":1,\"deviceId\":0,\"userApplicationId\":0,\"
organizationId\":65507,\"data\":\"eyJkZXZpY2VOYW1lIjoiQ0MxIiwiZGF0YSI6IjBDQ
jAwMUY0N0ZGRjAxNjJDNkVFREMiLCJ0aW1lc3RhbXAiOjE2NTcyMDQ0NDYsImRldkVVSSI6ImE4
NDA0MTVkMzE4NDU5NTQifQ==\",\"len\":100,\"status\":0,\"hiveRxTime\":\"2022-
07-06T08:37:08\"}",
141.     "payloadType": "json",
142.     "repeat": "",
143.     "crontab": "",
144.     "once": false,
145.     "onceDelay": 0.1,
146.     "x": 300,
147.     "y": 300,
148.     "wires": [
149.       []
150.     ]
151.   },
152.   {
153.     "id": "91e0c5c1.aca4d8",
154.     "type": "debug",
155.     "z": "c79b78c8.a3c098",
156.     "name": "JSON",
157.     "active": false,
158.     "tosidebar": true,
159.     "console": false,
160.     "tostatus": false,
161.     "complete": "payload",
162.     "targetType": "msg",
163.     "x": 1210,
164.     "y": 640,
165.     "wires": [],
166.     "icon": "node-red/parser-json.svg"
167.   },
168.   {
169.     "id": "8f18124e.53bf1",
170.     "type": "comment",
171.     "z": "c79b78c8.a3c098",
172.     "name": "SIMULATE SWARM HIVE PACKETS",
173.     "info": "",
174.     "x": 1650,
175.     "y": 160,
176.     "wires": []
177.   },
178.   {
179.     "id": "7263e5a0.c7fcbc",
180.     "type": "comment",
181.     "z": "c79b78c8.a3c098",
182.     "name": "DELETE ADDITIONAL FIELDS ADDED BY SWARM",
183.     "info": "",
```

```
    "x": 650,
    "y": 400,
    "wires": []
  },
  {
    "id": "19ac4be1.0ffee4",
    "type": "comment",
    "z": "c79b78c8.a3c098",
    "name": "SWARM HIVE PACKET",
    "info": "",
    "x": 300,
    "y": 400,
    "wires": []
  },
  {
    "id": "135d8c28.5d1634",
    "type": "catch",
    "z": "c79b78c8.a3c098",
    "name": "Catch all errors",
    "scope": null,
    "uncaught": true,
    "x": 820,
    "y": 360,
    "wires": [
      [
        "b72a49f1.ad8978"
      ]
    ]
  },
  {
    "id": "b72a49f1.ad8978",
    "type": "debug",
    "z": "c79b78c8.a3c098",
    "name": "Errors",
    "active": false,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "x": 1070,
    "y": 360,
    "wires": []
  },
  {
    "id": "87b723ea.66a5c",
    "type": "json",
    "z": "c79b78c8.a3c098",
    "name": "",
    "property": "payload",
    "action": "",
    "pretty": false,
    "x": 1070,
```

```
237.     "y": 440,
238.     "wires": [
239.       [
240.         "91e0c5c1.aca4d8",
241.         "db1c03c0.dca65"
242.       ]
243.     ]
244.   },
245.   {
246.     "id": "c3d9aa61.a2d998",
247.     "type": "influxdb out",
248.     "z": "c79b78c8.a3c098",
249.     "influxdb": "b03b7112.ca771",
250.     "name": "",
251.     "measurement": "correct_timestamps",
252.     "precision": "",
253.     "retentionPolicy": "",
254.     "x": 1570,
255.     "y": 440,
256.     "wires": []
257.   },
258.   {
259.     "id": "ec2cd494.c9d468",
260.     "type": "debug",
261.     "z": "c79b78c8.a3c098",
262.     "name": "decoded data",
263.     "active": false,
264.     "tosidebar": true,
265.     "console": false,
266.     "tostatus": false,
267.     "complete": "payload",
268.     "targetType": "msg",
269.     "x": 1020,
270.     "y": 640,
271.     "wires": [],
272.     "icon": "node-red-dashboard/ui_text.png"
273.   },
274.   {
275.     "id": "db1c03c0.dca65",
276.     "type": "function",
277.     "z": "c79b78c8.a3c098",
278.     "name": "Payload Decoder",
279.     "func": "//Parsing bytes from payload\nvar datajson = msg.payload;\nvar
    data =
    JSON.stringify(datajson.data).replace(/\"/g,'');\nconsole.log(typeof(data)+
    ' '+data);\n\nvar bytes = Buffer.from(data,
    'hex');\nconsole.log(bytes);\n\nif(datajson.deviceName == 'CC1'){\n  //
    Dragino\n  var decoded_payload = DecodeDragino(2, bytes);\n}else
    if(datajson.deviceName == 'CC2'){\n  var decoded_payload = DecodeCCNode(2,
    bytes);\n}\n\n\nmsg.payload = {...msg.payload,
    ...decoded_payload};\n\nreturn msg;\n\n\n\n\n\n//----------------------
    ///\n\n// CCNode //\nfunction DecodeCCNode(fPort, bytes) {\n var decoded =
    {};\n var offset = 0;\n var message_type = 0;\n\n if (offset + 1 <=
```

```
bytes.length) {\n  // Sequence (1 bytes)\n  decoded['sequence'] =
bytes[offset++];\n\n  // Message Type (1 bytes)\n  message_type =
bytes[offset++];\n  decoded['message_type'] = message_type;\n\n  //
Compressed message_type\n  if(message_type == 0x03){\n   // Message type 3
-> Compressed (Advanced CCNode Sigfox)\n\n   // Temperature\n   var
temperature = ((bytes[offset++] << 5) + ((bytes[offset] >> 3) &
0x1F))/100.0 - 25.0;\n   decoded['temperature'] = temperature;\n\n   //
Relative Humidity\n   var humidity = (((bytes[offset++] & 0x07) << (10-3))
+ ((bytes[offset] >> 1) & 0x7F))/10.0;\n   decoded['humidity'] =
humidity;\n\n   //Lux\n   if(bytes[offset++] & 0x01){\n    // Lux > 8000\n
var lux = ((bytes[offset++] << 5) + ((bytes[offset] >> 3) & 0x1F)) * 10;\n
}else{\n    // Lux < 8000\n    var lux = (bytes[offset++] << 5) +
((bytes[offset] >> 3) & 0x1F);\n   }\n   decoded['lux'] = lux;\n\n\n
//UVA\n   var uva = ((bytes[offset++] & 0x07) << (10-3)) + ((bytes[offset]
>> 1) & 0x7F);\n   decoded['uva'] = uva;\n  }    \n }    \n return
decoded;\n}\n\n\n// DRAGINO//\n\nfunction str_pad(byte){\n  var zero =
'0';\n  var hex= byte.toString(16);  \n  var tmp = 2-hex.length;\n  return
zero.substr(0,tmp) + hex;\n}\n\nfunction DecodeDragino(port, bytes)
{\n\nvar decode = {};\nif(port==2)\n{\n if(bytes.length==11)\n {\n
decode.TempC_SHT=parseFloat(((bytes[0]<<24>>16 |
bytes[1])/100).toFixed(2));\n  decode.Hum_SHT=parseFloat(((bytes[2]<<24>>16
| bytes[3])/10).toFixed(1));\n
decode.TempC_DS=parseFloat(((bytes[4]<<24>>16 |
bytes[5])/100).toFixed(2));\n\n  decode.Ext=bytes[6];\n
decode.Systimestamp=(bytes[7]<<24 | bytes[8]<<16 | bytes[9]<<8 | bytes[10]
);\n  \n  return decode;\n }\n else\n {\n  decode.Status=\"RPL data or
sensor reset\";\n \n  return decode;\n }\n}\n\nif(port==3)\n{\n
decode.Status=\"Data retrieved, your need to parse it by the application
server\";\n \n return decode;\n}\n\nif(port==4)\n{\n
decode.DS18B20_ID=str_pad(bytes[0])+str_pad(bytes[1])+str_pad(bytes[2])+str
_pad(bytes[3])+str_pad(bytes[4])+str_pad(bytes[5])+str_pad(bytes[6])+str_pa
d(bytes[7]);\n\n return decode;\n}\n\nif(port==5)\n{\n
decode.Sensor_Model=bytes[0];\n
decode.Firmware_Version=str_pad((bytes[1]<<8)|bytes[2]);\n
decode.Freq_Band=bytes[3];\n decode.Sub_Band=bytes[4];\n
decode.Bat_mV=bytes[5]<<8|bytes[6];\n \n return decode;\n}\n\n}",
    "outputs": 1,
    "noerr": 0,
    "x": 1270,
    "y": 440,
    "wires": [
      [
        "d9f55c41.b7a15",
        "c3d9aa61.a2d998"
      ]
    ]
  },
  {
    "id": "d9f55c41.b7a15",
    "type": "debug",
    "z": "c79b78c8.a3c098",
    "name": "",
    "active": false,
```

```
297.     "tosidebar": true,
298.     "console": false,
299.     "tostatus": false,
300.     "complete": "payload",
301.     "targetType": "msg",
302.     "x": 1470,
303.     "y": 640,
304.     "wires": []
305.   },
306.   {
307.     "id": "23fae666.1f70aa",
308.     "type": "inject",
309.     "z": "c79b78c8.a3c098",
310.     "name": "CC Node",
311.     "topic": "",
312.     "payload":
    "{\"packetId\":0,\"deviceType\":1,\"deviceId\":0,\"userApplicationId\":0,\"
    organizationId\":65507,\"data\":\"eyJkZXZpY2VOYW1lIjoiQ0MyIiwiZGF0YSI6IjAzM
    DNBRDUONEMwMDAwMDAiLCJ0aW1lc3RhbXAiOjE2NTcyMTI4NTgsImRldkVVSSI6IjMzMzQzNjM1
    NTgzMDY5MDQifQ==\",\"len\":100,\"status\":0,\"hiveRxTime\":\"2022-07-
    06T08:37:08\"}",
313.     "payloadType": "json",
314.     "repeat": "",
315.     "crontab": "",
316.     "once": false,
317.     "onceDelay": 0.1,
318.     "x": 340,
319.     "y": 260,
320.     "wires": [
321.       [
322.         "dc72c3b0.d695d",
323.         "64712a45.d3a514"
324.       ]
325.     ]
326.   },
327.   {
328.     "id": "b03b7112.ca771",
329.     "type": "influxdb",
330.     "z": "",
331.     "hostname": "influxdb",
332.     "port": "8086",
333.     "protocol": "http",
334.     "database": "rupestres",
335.     "name": "DDBB Rupestres",
336.     "usetls": false,
337.     "tls": ""
338.   }
339. ]
340.
```

Índice del Anexo 2 – Hojas de características

1. Hoja de características del kit de evaluación Swarm

En la siguiente página se encuentra la hoja de características del *Swarm Eval Kit*, descargada de la página web del fabricante: Swarm Technologies.

El datasheet presenta de forma resumida las especificaciones técnicas del producto junto con otras características definitorias, pero existe un una guía de inicio rápido que complementa dicha información con descripciones detalladas de los modos de funcionamiento, órdenes de usuario, métodos de conexión...etc.

Para consultar la documentación completa, se recomienda acceder al portal web del fabricante: *www.swarm.space*

SWARM

PRODUCT OVERVIEW

Swarm Eval Kit

The Swarm Eval Kit transmits and receives data to and from Swarm's satellite network to provide connectivity anywhere on Earth. The Eval Kit is designed to provide the developer with an easy-to-use platform, with the included FeatherS2 - ESP32 board + OLED, a USB-C port, and I2C port for sensors. FeatherWing add-on modules can provide a suite of additional capabilities.

READY OUT-OF-THE-BOX

The Eval Kit is a turn-key hardware platform for satellite data connectivity anywhere on Earth. It immediately starts beaconing your GPS location, battery, and solar charge information once turned on. You can easily send messages with 1-click using the Swarm messaging app, and receive data by email or the in Swarm user web UI.

INCLUDES EVERYTHING YOU NEED

The Eval Kit includes a tripod, solar panel, batteries, and integrated VHF and GPS antenna. A live readout of RF background noise helps you achieve the best possible link quality. Connect your devices via WiFi (AP or STA mode), USB, or serial interfaces, and easily retrieve and manage your data via the Swarm Cloud and REST API.

CONTACT

Website: www.swarm.space
Email: info@swarm.space

KEY FEATURES

- Easy remote data transfer from anywhere on Earth via the Swarm constellation
- Comes ready out-of-the-box
- Includes everything needed for developers to build IoT products

SENSORS	Onboard GPS (lat/lon/alt), CPU Temp
DIMENSIONS	32x26x8 cm (not including tripod or antenna)
MASS	2.6 kg
POWER	9 W solar panel, USB-C charger 30 Whr battery (3x 18650 cells) Unregulated 4.2 V battery output for external sensors or devices I2C with optional 3V3 pull-ups and a 3V3 regulated output 24hr lifetime on batteries only (wif on)
ENVIRONMENT	Operational: -40 C to +60 C Waterproof enclosure, IP67 rating
INTERFACE	Power: Solar + battery Data: WiFi, USB-serial, Feather-serial, I2C Antennas: Sat + GPS (U.FL or SMA)
BIT RATE	1 kbps (Max packet size 192 bytes) AES256 GCM encryption
FREQUENCY	137-138 MHz (downlink) 148-150 MHz (uplink)
COMMANDING	Two-letter NMEA formatted

2. Manual de usuario del gateway RAK7289

RAK® Documentation Center

RAK7289 WisGate Edge Pro Datasheet

Overview

Description

RAK7289 WisGate Edge Pro is an ideal product for IoT commercial deployment. With its industrial-grade components, it achieves a high standard of reliability.

Supports up to 16 LoRa channels, multi backhaul with Ethernet, Wi-Fi, and Cellular connectivity. Optionally there is a dedicated port for different power options, solar panels, and batteries. With its new enclosure design, it allows the LTE, Wi-Fi, and GPS antennas to be inside the enclosure.

The gateway provides for a solid out-of-the-box experience for quick deployment. Additionally, since its software and UI sits on top of OpenWRT it is perfect for the development of custom applications (via the open SDK).

Thus, the RAK7289 is suited for any use case scenario, be it rapid deployment or customization with regards to UI and functionality.

Product Features

Hardware

- **IP67/NEMA-6** industrial grade enclosure with cable glands
- **PoE (802.3af)** + Surge Protection
- Dual LoRa Concentrators for up to **16 channels**
- **Backhaul:** Wi-Fi, LTE and Ethernet
- GPS
- Supports DC 12V or Solar power supply with Electricity monitoring (Solar Kit optional)
- Internal antenna for Wi-Fi, GPS and LTE, External antenna for LoRa
- Dying-Gasp(optional)

Software

- Built-in Network Server
- OpenVPN
- Software and UI sit on top of **OpenWRT**
- LoRaWAN 1.0.3
- **LoRa Frame filtering** (node whitelisting)
- **MQTT v3.1** Bridging with **TLS** encryption
- **Buffering of LoRa frames in Packet Forwarder mode** in case of NS outage (no data loss)
- **Full duplex (optional)**
- **Listen Before Talk (optional)**
- **Fine timestamping (optional)**

Specifications

Overview

The overview presents the block diagram for the RAK7289 that shows the internal architecture of the board.

Block Diagram

RAK® Documentation Center

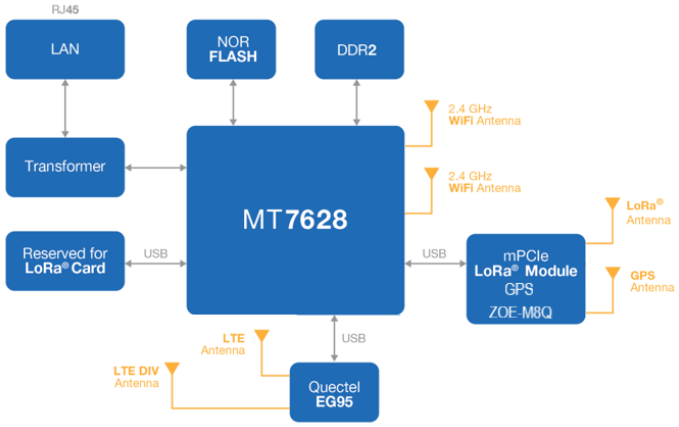


Figure 1: RAK7289 WisGate Edge Pro Block Diagram

Main Specifications

RAK® Documentation Center

Feature	Specifications
Computing	MT7628, DDR2 RAM 128 MB
Wi-Fi feature	Frequency: 2.4 GHz (802.11 b/g/b/)
	2x2 MIMO
	RX Sensitivity: -95 dBm (Min)
	TX Power: 20 dBm (Max)
	Operation channels: 2.4 GHz: 1-13
LoRa feature	SX1303 mPCIe card (connects maximum of two)
	8 Channels (16 channels optional)
	RX Sensitivity: -139 dBm (Min)
	TX Power: 27 dBm (Max)
	Listen Before Talk
Frequency	EU433/CN470/EU868/US915/AS923/AU915/IN865/KR920
Cellular feature	Supports Quectel EG95-E/EG95-NA (IoT/M2M -optimized LTE Cat 4 Module)
	EG95-E for EMEA Region
	- LTE FDD: B1/B3/B7/B8/B20/B28A
	- WCDMA: B1/B8
	- GSM/EDGE: B3/B8
	EG95-NA for North America Region
	- LTE FDD: B2/B4/B5/B12/B13
	- WCDMA: B2/B4/B5
Power supply	PoE (IEEE 802.3 af), 37~57 VDC
ETH	RJ45 (10/100 Mbps)
Antenna	LoRa: 1 or 2 N-Type connectors
	LTE: Internal antenna

RAK® Documentation Center

Feature	Specifications
	Wi-Fi: Internal antenna
Ingress protection	IP67
Enclosure material	Aluminum and plastic
Operating temperature	-30 ˚C to +55 ˚C
Installation method	Pole or wall mounting

Hardware

The hardware specification covers the interfacing of the RAK7289 and its corresponding functionalities. It also presents the parameters and the standard values of the board.

RF Specifications

Wi-Fi Radio Specifications

RAK® Documentation Center

Feature	Specifications
Wireless Standard	IEEE 802.11 b/g/n
Operating Frequency	ISM band: 2.412~2.472 (GHz)
Operation Channels	2.4 GHz: 1-13
Transmit Power (The max power maybe different depending on local regulations) - per chain	802.11b
	-19 dBm @1 Mbps
	-19 dBm @11 Mbps
	802.11g
	-18 dBm @6 Mbps
	-16 dBm @54 Mbps
	802.11n (2.4G)
	-18 dBm @MCS0 (HT20)
	-16 dBm @MCS7 (HT20)
	-17 dBm @MCS0 (HT40)
	-15 dBm @MCS7 (HT40)
Receiver Sensitivity (Typical)	802.11b
	-95 dBm @1 Mbps
	-88 dBm @11 Mbps
	802.11g
	-90 dBm @6 Mbps
	-75 dBm @54 Mbps
	802.11n (2.4G)
	-89 dBm @MCS0 (HT20)
	-72 dBm @MCS7 (HT20)
	-86 dBm @MCS0 (HT40)

RAK® Documentation Center

Feature	Specifications
	-68 dBm @MCS7 (HT40)

LoRa Radio Specifications

Feature	Specifications
Operating Frequency	EU433/CN470/EU868/US915/AS923/AU915/IN865/KR920
Transmit Power	27 dBm (Max)
Receiver Sensitivity	-139 dBm (Min)

Interfaces

Figure 2: RAK7289 WisGate Edge Pro Interfaces

- The function of the Reset key is as follows:
  - **Short press:** Restart the gateway.
  - **Long press** (5s and above): Restore factory settings.
- LEDs status description:

LEDs	Status Indication Description
LED 1 (PWR)	Power indicator - The LED is on when device power is on
LED 2 (ETH)	ON - Linkup
	OFF - Linkdown
	Flicker - Data transmitting and receiving
LED 3 (LoRa 1)	ON - LoRa 1 is working
	OFF - LoRa 1 is not working
	Flicker - Indicate LoRa 1 Packet receiving and sending
LED 4 (WLAN)	AP Mode:
	-ON - The AP is up
	-Flicker - Data receiving and sending
	STA Mode:
	-Slow flicker (1 Hz) - Disconnected
	-ON - Connected
	-Flicker - Data receiving and sending
LED 5 (LTE)	Slow Flicker (1800 ms High / 200 ms Low) - Network searching
	Slow flicker (200 ms High / 1800 ms Low) - Idle
	Fast flicker (125 ms High / 125 ms Low) - Ongoing data transfer
	ON - Voice is working
LED 6 (LoRa 2 for 16 channel)	ON - LoRa 2 is working
	OFF - LoRa 2 is not working
	Flicker - Indicate LoRa 2 Packet receiving and sending

Software

Firmware

The firmware sits on OpenWRT, which makes it possible to customize it. There is a Web UI for easy configuration and management of the device, as well as the possibility for SSH2 management.

RAK® Documentation Center

Software Features

LoRaWAN	Network	Management
Supports class A, B, C	Wi-Fi AP mode	WEB UI
LoRa package forward	Wi-Fi Client mode	SSH2, NTP
Frequency band setup	LTE APN setup	Firmware update
TX power setup	Uplink backup	LoRa packet forwarder
Data logger	Support 802.1q	Built-In Network Server
Statistic	DHCP Server/Client	OpenVPN, Ping Watch Dog
Location setup	Firewall	MQTT Bridge
Server address and port setup		

Models/Bundles

Part Number	8 Channel SX1303	16 Channel SX1303	Cat4 Cellular	GPS	Wi-Fi	Dying gasp
RAK7289-XYZ	√		√	√	√	
RAK7289-XYZ		√	√	√	√	
RAK7289-XYZ	√		√	√	√	√
RAK7289-XYZ		√	√	√	√	√
RAK7289-XYZ	√			√	√	
RAK7289-XYZ		√		√	√	
RAK7289-XYZ	√			√	√	√
RAK7289-XYZ		√		√	√	√

Last Updated: 11/29/2021, 5:12:23 AM

3. Manual de usuario del nodo sensor CollectionCare

A continuación se incluye un extracto del manual de usuario del sensor CollectionCare, con las características definitorias del nodo: dimensiones, peso, y especificaciones de las medidas.

Sensor node specifications & description

Sensor node specifications

The CollectionCare sensor node is the monitoring device of the CollectionCare system. It is developed to monitor environmental parameters that can have a direct influence on the degradation of cultural objects. Its specifications are as follows

Measurements	Temperature (T), Relative humidity (RH), Illuminance (L), Ultraviolet radiation (UV), Vibrations (V)
Sensor node sampling time	Default sampling time: 60 min (see FAQ's)
Operating temperature	-20°C up to +70°C
Power supply	Li-SOCL Battery, voltage 3.6V & nominal energy 9.36Wh[2]
Battery life	~10 years
Weight	36.7 g
Dimensions	57,50 x 52,15 x 15,40 mm
Included accessories	Instruction manual ..... x1 Slide part of the fixing system ..... x1 Plugs ..... x2 Screws ..... x2 Double-sided adhesive tape ..... x2 Reset tool x1
Optional accessories	Vibration sensor (accelerometer sensor) Air pollutants sensors (tVOCs sensor & NO2 sensor)

Technical data: environmental sensors

The CollectionCare sensor node is a simple and easy-to-operate wireless sensing device designed to simultaneously monitor and transmit to the cloud different environmental parameters. By default, this device comes with four onboard sensors which are temperature (T), relative humidity (RH), illuminance (L) and ultraviolet radiation (UV). As for the vibration (V) and air pollutants (AP) sensors, these can be added to the node on request.

Five of these sensors are located inside the node, such as T, RH, L, UV, and V, while AP sensors must be connected to the node through an external connector.
Also, these sensors have been selected in compliance with various European standards.

Temperature °C

Measuring range	-40 to 125 °C
Accuracy tolerance	± 0.5 °C
Resolution	0.01 °C
European standard	EN15757, EN 15758

Relative humidity %

Measuring range	0 to 100 %RH
Accuracy tolerance	± 2 %RH +8 at 25 %RH histeresis
Resolution	0.1 %RH
European Standard	EN15757, EN16242

LIGHT

Illuminance lx

Measuring range	0 to 120000 lx
Accuracy tolerance	±2 lx
Resolution	1 lx
European standard	EN16163

Ultraviolet radiation mW/m ^{2}

Measuring range	0 to 30000 mW/m^2
Accuracy tolerance	±2 mW/m^2
Resolution	1 mW/m^2
European standard	EN16163

Technical data: optional environmental sensors

Air pollutant and vibration sensors are added on request at extra cost.

AIR POLLUTANTS

NO2 ppm		**VOCs ppb**	
Measuring range	0 to 5 ppm	Output range	TVOC: 0 - 60'000 ppb CO2eq: 0 - 60'000 ppm
Lower detection limit	<20 ppb	Accuracy	15% of measured value
Resolution	<20 ppb		

Vibrations g

Measuring range	+ 8g
Resolution	1 mg
Sensitivity	3% mg/digit

Note: All sensors are calibrated. An optional calibration certificate may be required which involves an additional cost and periodic recalibrations.

4. Manual de usuario de la estación WS100 de UBIQ-IoT

Finalmente se añade el manual de la estación climatológica WS100LWR de UBIQ-IoT. En él se detalla el procedimiento para la puesta en marcha, y la configuración de los parámetros de la transmisión LoRaWAN y de la medidas. Además, en la página 35 (18 del manual) se encuentran enlaces a los *datasheets* del fabricante del equipo original, Davis Instruments.

UBIQ Weather Station

WS-100 Series

User Manual

Version 1.1

1/20

UBIQ-IoT s.r.l.
Viale Sondrio, 3
20124 – Milano (MI), Italy
Tel: +39 3277090124
info@ubiq-iot.com
www.ubiq-iot.com

Trademarks and Permissions
All trademarks and trade names mentioned in this document are the property of their respective holders.

Notice
The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Battery
The battery is NOT included with the product. The weather station has been tested and certified using one SAFT LSH20 3.6V lithium-thionyl chloride (Li-SOCl2) battery (see Technical details section).
The battery life time in operation mode is more than one year (tested in operation mode, transmitting on the EU868 frequency band every 20 minutes. More details about battery consumption are described in the Technical details section.

Disclaimer

All contents of this document are provided “AS IS”.

Except as required by applicable laws, no warranties of any kind, either express or implied, including but not limited to, the implied warranties of merchantability and fitness for a particular purpose, are made in relation to the accuracy, reliability or contents of this document.

To the maximum extent permitted by applicable law, under no circumstances shall UBIQ-IoT S.r.l. be liable for any special, incidental, indirect, or consequential damages, or loss of profits, business, revenue, data, goodwill savings or anticipated savings regardless of whether such losses are foreseeable or not.

The maximum liability of UBIQ-IoT S.r.l. arising from the use of the product described in this document shall be limited to the amount paid by the customer for the purchase of this product.

Contents

Introduction

UBIQ WS-100 is a LoRaWAN compliant weather station based on Davis Instruments Vantage Pro2 sensors array.

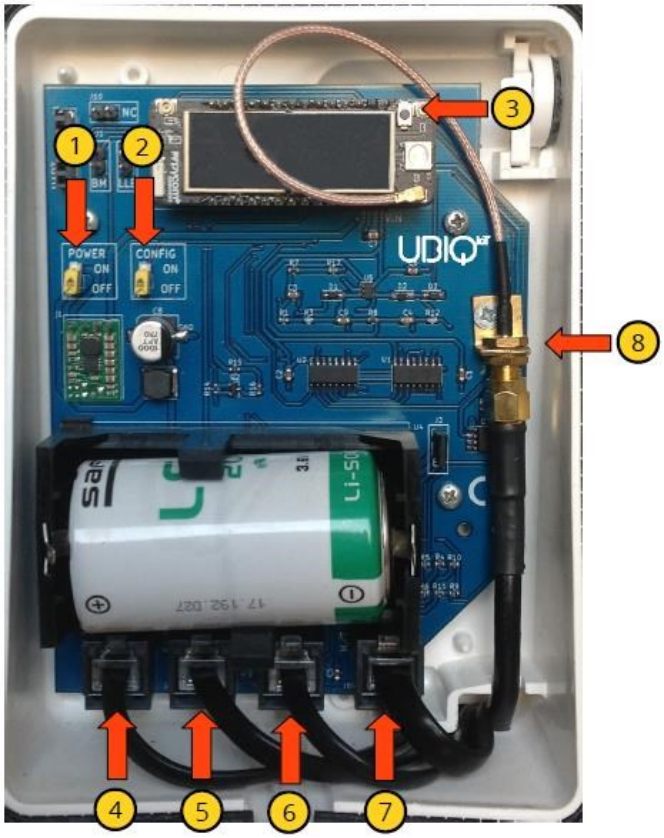


Figure 1 – Inside of the Weather Station

1- Power switch
2- Configuration switch
3- Reset button

5/20

4- Solar radiation sensor connector (RJ11)
5- Rain gauge sensor connector (RJ11)
6- Wind speed and direction sensor connector (RJ11)
7- Air Humidity and Temperature sensor connector (RJ11)
8- Antenna connector

The weather station can be fully configured to communicate with any LoRaWAN compatible gateway transmitting on one of the following ISM Band channel frequencies:

- AS923 (channel plan AS923)
- AU915 (channel plan AU915-928)
- EU868 (channel plan EU863-870)
- US915 (channel plan US902-928)

6/20

Device operation

The LoRaWAN transmission parameters MUST be configured on the weather station before its use (see Configuration Mode section). Read the documentation of your LoRaWAN gateway to find the correct configuration data.

The mounting guide for the installation of the weather station can be found at the following address:
https://www.davisinstruments.com/product_documents/weather/manuals/07395-298_IM_07717.pdf

WARNING!
Before any operation involving the antenna (attach/detach), the battery (insertion/removal) or any probe (insertion/removal) the weather station has to be turned off (Power switch in "OFF" position, see Figure 1).

Battery insertion and first power on of the device

After all mounting operations are completed, insert the battery in its socket, set the Configuration switch to the "ON" position and turn on the weather station setting the Power switch to the "ON" position. After a few seconds the Wi-Fi network of the device becomes available and it is possible to set-up the station (see the Configuration Mode section below).

Configuration Mode

To activate the Configuration Mode of the device the Configuration switch (see Figure 1) has to be in the "ON" position. If the weather station is already turned on, push the Reset button, otherwise turn it on using the Power switch.
Entering in the Configuration Mode, the device enables the internal Wi-Fi network, which could be accessed using the following credentials:

- SSID (network name) = UBIQ_WS100
- Password = configpw

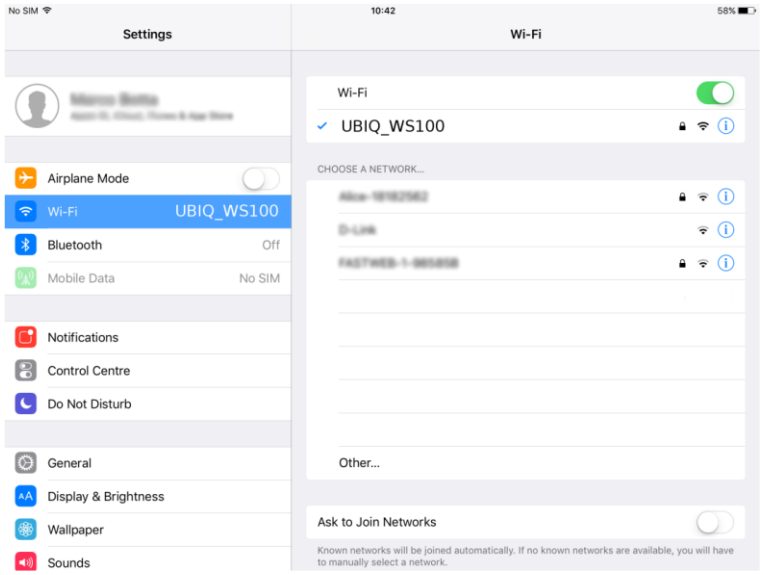


Figure 2 – Connection to the weather station Wi-Fi network from an iPad device

When your device is connected to the WiFi network, open the Internet browser and open the following address:

http://192.168.4.1/index

8/20

The web page represented in the next picture will be loaded.

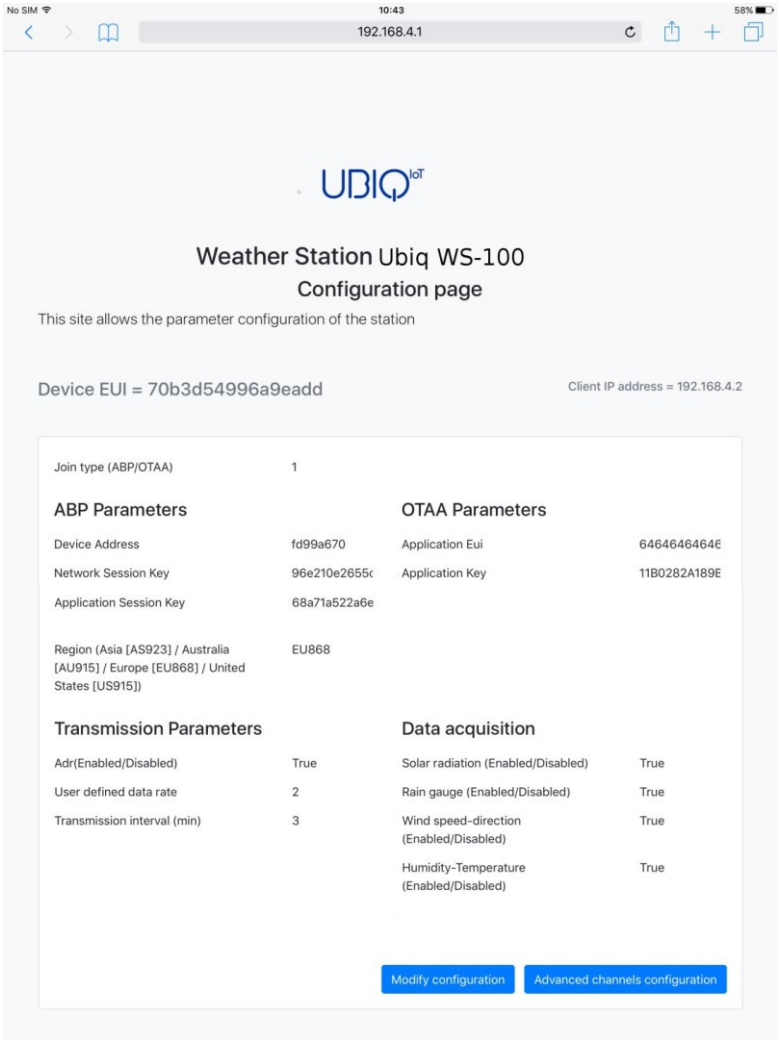


Figure 3 - Configuration page

9/20

The configuration page shows the device unique identifier (Device EUI) in bold and, on the right, the IP address of the client device connected to the Wi-Fi network of the weather station.

The "Data acquisition" section lists which probes are enabled. The weather station will read and transmit only the data acquired from the enabled probes.

The "Transmission interval" represents the interval (in minutes) between two probe readings and transmissions of the data. It can be set between 10 and 120 minutes.

All the other data presented are used to configure the weather station transmission parameters needed to communicate with the LoRaWAN gateway. These data have to be obtained from the organization that manages the network infrastructure:

- Join type: ABP (Activation By Personalization)
  OTAA (Over The Air Activation)
- ABP join type specific parameters
- OTAA join type specific parameters
- Transmission frequency band: AS923
  AU915
  EU868
  US915
- ADR (Adaptive Data Rate)
- User specified data rate (DR) [used only if ADR is not enabled]

10/20

Configuration data

The weather station configuration can be modified clicking on the “Modify configuration” button at the bottom of the page (Figure 3). The browser will open a new page where each parameter can be modified.

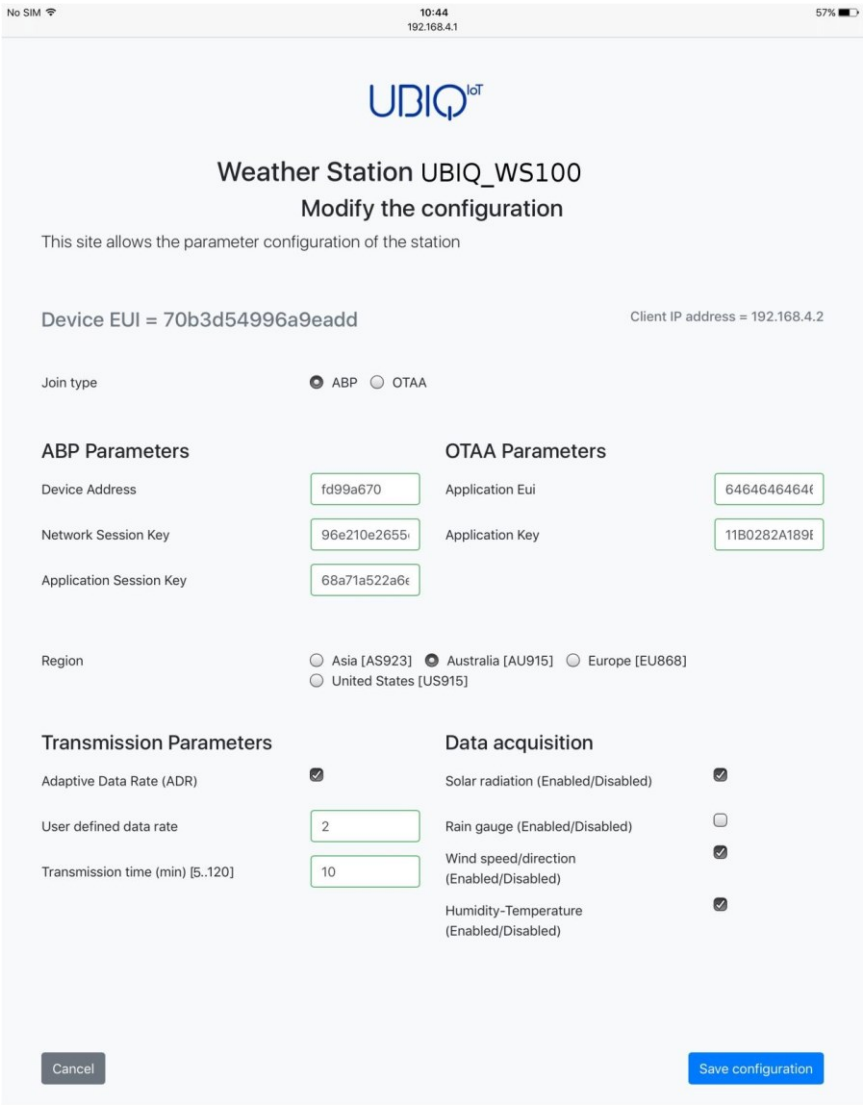


Figure 4 - Configuration editing page

12/20

To save the configuration data, click the "Save configuration" button at the bottom of the page. The browser will present the following page to confirm that the device has been correctly configured.

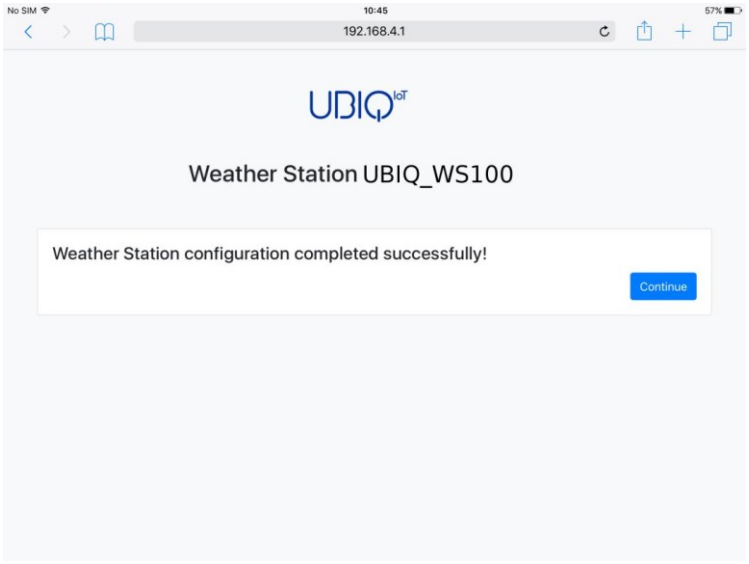


Figure 5 - Configuration modified successfully

Channels configuration

For the AU915 and US915 transmission bands the specific channels can be activated using the "Advanced channels configuration" button (see Figure 3) which enables the selection of the specific channels used by the LoRaWAN gateway.

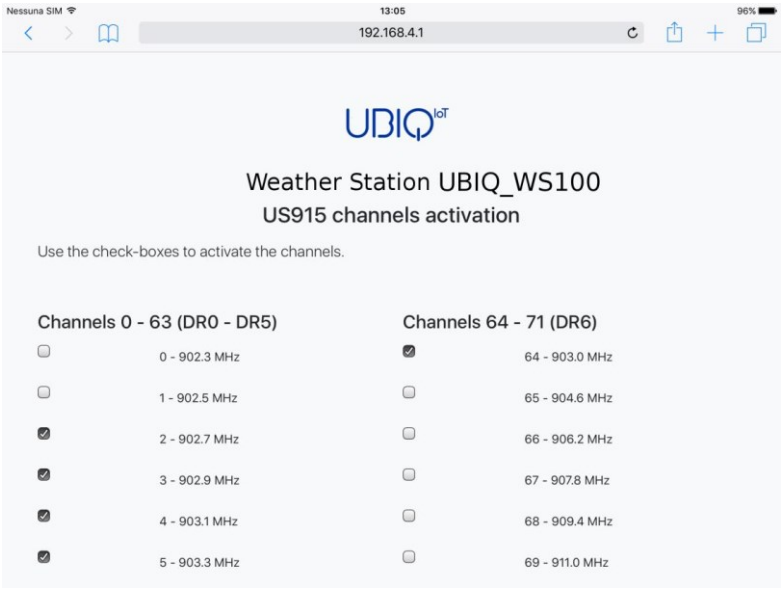


Figure 6 - Channels configuration page for AU915 and US915 bands

To save the configuration data, click the "Save configuration" button at the bottom of the page. The browser will present the following page to confirm that the device has been correctly configured.

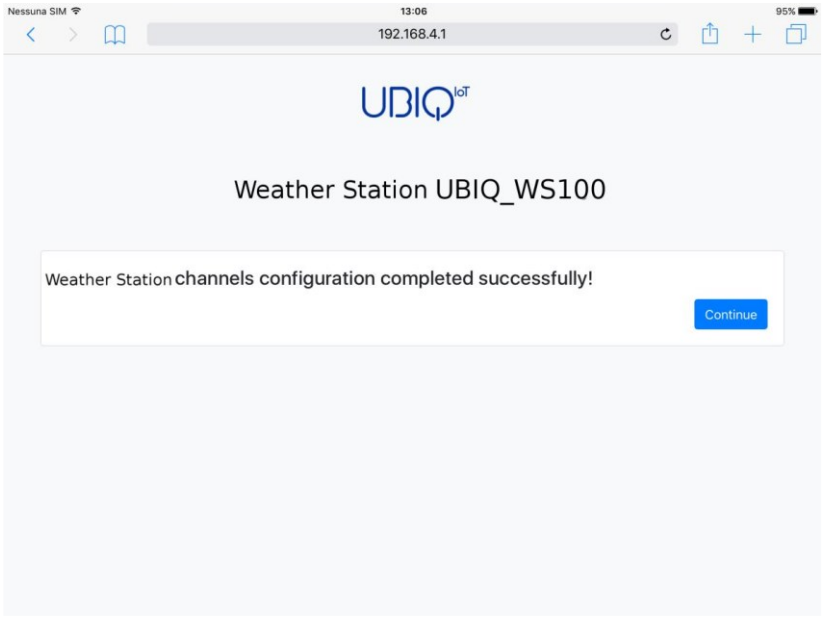


Figure 7 - Channel configuration completed

15/20

Exit the Configuration Mode

To exit the Configuration Mode, set the Configuration switch to the “OFF” position and push the Reset button (see Figure 1) to initiate the hand-shaking sequence.

Hand-shaking sequence

The weather station will start the hand-shaking sequence sending a series of three messages which have a fixed payload of 1 byte separated by 10 seconds each. These messages use the Data Rate set by the weather station configuration.
After the hand-shaking sequence is over, the device enters the Operation Mode and will start to transmit the sensor readings according to the specified configuration.

Operation Mode

The weather station works in Operation Mode when the Configuration switch is set in “OFF” position. When turned on setting Power switch in “ON” position, the weather station will immediately start to transmit according to the saved configuration. No actions are required by the user.

If case the device stops to send data push the Reset button (see Figure 1). The weather station should begin immediately to transmit again. Depending what was the cause of the problem, the device could restart from the hand-shaking sequence.

Technical details

Weather station dimensions:
75cm x 55cm x 25cm (L x H x P)

Operating temperature range:
-30 °C / +50 °C

Battery

Reference data sheet available at:
https://www.saftbatteries.com/products-solutions/products/ls-lsh

Solar Radiation

Range:
SR = 0 – 1800 Wm^2

Reference data sheet available at:
https://www.davisinstruments.com/product_documents/weather/spec_sheets/6450_SS.pdf

Rain

Range:
R = 0 – 10000

Conversion formula:
mm = (R / 100)
Range mm = from 0 mm to 100 mm

Reference data sheet available at:
https://www.davisinstruments.com/product_documents/weather/spec_sheets/6463_6465_SS.pdf

Wind Speed and Direction

Range:
WS = 0 – 322 Km/h
WD = 0° – 360°

Reference data sheet available at:
https://www.davisinstruments.com/product_documents/weather/spec_sheets/7911_SS.pdf

Temperature

Range:

T = 3300 – 13800

Range °C = from -40.15 °C to +64.85 °C

Conversion formula:

°C = [(T + 20000) / 100] – 273.15

Reference data sheet available at:

https://www.davisinstruments.com/product_documents/weather/spec_sheets/6830_6832_SS.pdf

Relative Humidity

Range:

H = 0 – 100 RH%

Reference data sheet available at:

https://www.davisinstruments.com/product_documents/weather/spec_sheets/6830_6832_SS.pdf

Energy consumption

Using a data rate of zero (DR 0), with the recommended battery model (SAFT LSH20), the life of a battery should be:

- at least 30 months with a transmission interval of one hour (1h)
- at least 16 months with a transmission interval of thirty minutes (30 min)

WARNING!

In Configuration Mode the unit requires a battery capable to provide a current of at least 1 A; in Operation Mode this requirement is lowered to at least 400 mA. Please, pay particular attention to these values for batteries different than the model specified in this manual.

Even if not recommended, it is possible to use a battery for the Operation Mode and use instead a power adapter to supply the device in Configuration Mode; the power adapter must be capable to provide a voltage between 3.5 V and 5.5 V. In any case DO NOT exceed 5.5 V or the device will be permanently damaged.

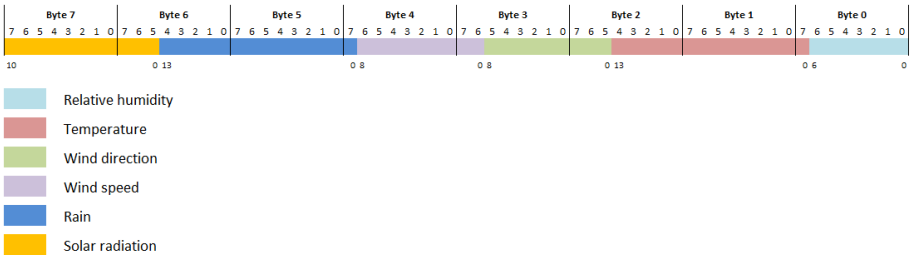
18/20

Payload specification

Operation mode payload format

Solar radiation: 11 bits
Rain: 14 bits
Wind speed: 9 bits
Wind direction: 9 bits
Temperature: 14 bits
Relative humidity: 7 bits

The bits are arranged into the 8 bytes long payload as shown in the following map:



Hand-shaking payload format

Byte 00 : Hand-shaking packet number

19/20

Troubleshooting

The Wi-Fi network in Configuration Mode is not available

Power off the unit and:

- make sure that the battery is firmly inserted into its socket
- make sure that the Configuration switch is in “ON” position
- the battery conforms to the specification

Switch on the device.

The Wi-Fi network in Configuration Mode could not be accessed

- make sure to insert the correct credentials
- make sure to access the specified address
- if your computer has installed a firewall, make sure that there are no rules blocking the configuration page address

The device messages are not received by the gateway/network server

Check the configuration parameters match those one specified by the service provider. If still no transmissions are received power off the unit and:

- make sure that the battery is firmly inserted into its socket
- the battery conforms to the specification
- make sure that the Configuration switch is in “OFF” position

Switch on the device.

The device stops to transmit

Push the Reset button and check if the handshake or normal operation messages start to be received again.

If still no transmission could be received, power off the unit, switch to Configuration Mode and check if the device configuration data are correct. Power off the unit, switch to Operation Mode and power on again the device.

20/20