Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politècnica de València

Trabajo Fin de Grado

Ingeniería en Electrónica Industrial y Automática

Desarrollo de una pasarela y nodos LoRaWAN como solución de bajo coste para implantaciones IoT

Documentos:

1 Memoria

2 Planos

3 Pliego de condiciones

4 Presupuesto

5-Anexos:

5.A1: Documentación código

5.A2: Hojas de características

Autor:

D. Alejandro Ortega Pérez

Tutor:

D. Ángel Perles Ivars

Valencia, diciembre de 2019

Gracias a mi tutor por su ayuda durante todo el desarrollo. A mi família por su apoyo desde la distancia y a mi pareja, Ángela, por su apoyo incondicional durante toda la carrera.

Resumen

Desde la aparición de Internet en 1969, el cambio cultural y social que ha generado ha sido más que evidente. El avance tecnológico en las últimas décadas ha sido más que notable, gracias en gran medida a la creación de una red de conocimientos global. Con el paso de los años hemos ido conectando más equipos con el fin de ampliar esta gran red, primero fueron los ordenadores, después aparecieron los móviles inteligentes, y en la actualidad estamos conectando casi cualquier cosa, esta es la próxima revolución tecnológica que estamos viviendo en la actualidad, lo que llamamos internet de las cosas, o IoT por sus siglas en ingles.

Existen en la actualidad diversas tecnologías para conectar estos dispositivos a la red, como es el caso de LoRa, una tecnología de modulación de frecuencia que permite la transmisión de datos a largas distancias y a un bajo coste. Existe un problema en la actualidad con esta tecnología en España, y es la falta de zonas de cobertura. La primera parte de nuestro trabajo consiste en dar solución a este problema, y mostrar la posibilidad de desarrollar una pasarela que incremente la cobertura LoRa gracias a la plataforma TTN.

Las ventajas de crear esta red, es la posibilidad de hacer uso de ella para la conexión de dispositivos inteligentes, o como les vamos a denominar a partir de ahora nodos. Estos nodos construidos en base a LoRa tienen la ventaja principal de bajo consumo energético, instalaciones económicas gracias a los alcances de transmisión elevados y conexión gratuita al usar la banda ISM que no requiere de licencia para su utilización.

Aunque nuestro interés actual es en la tecnología, no hay mejor muestra de los beneficios que una aplicación real. Por ello la segunda parte de nuestro trabajo consiste en monitorizar una comunidad de vecinos, con el fin de poder controlar el estado de distintas zonas como son la piscina, un parque infantil y el garaje.

Palabras clave: LoRa ,LoRaWAN, IoT, TTN, bajo consumo, monitorización, Pycom, github, sensores remotos.

Summary

Since the aparition of Internet in 1969, the cultural and social change it has generated has been more than evident. Technological advancement in recent decades has been more than remarkable, thanks in large parts to the creation of a global knowledge network. Over the years we have been connecting more devices in order to expand this great network, first were computers, then smartphones appeared, and today we are connecting almost anything, this is the next technological revolution that we are living today, what we call the Internet of Things, or IoT.

There are currently various technologies to connect these devices to the network, such as LoRa, a frequency modulation technology that allows the transmission of data over long distances and at a low cost. There is a problem today with this technology in Spain, and it is the lack of coverage areas. The first part of our project is to solve this problem, and show the possibility of developing a gateway that increases LoRa coverage thanks to the TTN platform.

The advantages of creating this network, is the possibility to make use of it for the connection of smart devices, or as we are going to call them from now on, nodes. These LoRa-based nodes have the main advantage of low power consumption, economical installations thanks to high transmission ranges, and free connection thanks to the use of non-licensed ISM band.

Although our current interest is in technology, there is no better example of the benefits than a real application. Therefore, the second part of our project is to monitor a community of neighbors, in order to be able to control the status of different areas such as the swimming pool, a playground and the garage.

Keywords: LoRa, LoRaWAN, IoT, TTN, low power, monitoring, Pycom, github, remote sensors.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Desarrollo de una pasarela y nodos LoRaWAN como solución de bajo coste para implantaciones IoT.

1. MEMORIA

Autor:

D. Alejandro Ortega Pérez

Tutor:

D. Ángel Perles Ivars

Valencia, diciembre de 2019

ÍNDICE DEL PROYECTO

Memoria

BOM Bill of materials. Lista de materiales

FIFO: First in First Out

FSK: Frequency Shift Keying

GND Ground (0V)

HAL Capa de abstracción de hardware

I2C Circuito inter-integrado´

IOT Internet de las cosas

IP Protocolo de Internet

LORAWAN Long Range Wide Area Network

LPWAN: Low-Power Wide Area Network

MAKER: Persona o grupo que desarrollan soluciones tecnológicas de bajo coste mediante código abierto.

MISO: Master Input Slave Output

MOSI: Maste Output Slave Input

MQTT Message Queuing Telemetry Transport

PAYLOAD: Carga útil,

PLUG AND PLAY: Dispositivo preparado para funcionar sin instalación previa.

SOC Sistema en chip

SPI Interfaz de periféricos serie

STM STMicroelectronics

TTN: The Things Network

UART Transmisor-Receptor Asincrono Universal

USB Bus Serie Universa

VCC Voltaje común de colector (alimentación positiva IC)

VSS Alimentación negativa IC

WAN: Wide Area Network

Memoria

1. OBJETO DEL PROYECTO

Mediante el presente proyecto se pretende crear una red de nodos sensores con tecnología Lora, que se comuniquen mediante protocolo LoRaWAN a través de una pasarela con la plataforma THE THINGS NETWORK

El proyecto se enfoca en el desarrollo y aprovechamiento de esta tecnología, siendo secundario el uso final que se le dé. La programación y puesta en marcha de la pasarela va a permitir la conexión de cientos de nodos, tanto lo desarrollados en este documento, como futuros desarrollos o nodos de terceros. No obstante, y con el fin de mostrar el potencial de este tipo de desarrollos en el entorno cotidiano, se va a implementar en una comunidad de vecinos con el fin de crear una Comunidad 4.0.

Son objeto de este proyecto, tanto la programación y desarrollo de los nodos sensores como del Gateway de comunicaciones que ejecute el protocolo LoRaWAN, como su implementación funcional.

Entran en el alcance así, el desarrollo de una pasarela totalmente operativa que comunique cualquier nodo dentro del área de influencia elegida, con la plataforma TTN. El desarrollo de un nodo sensor que monitorice las condiciones de un parque infantil y la creación de un panel web que muestre de manera sencilla la información obtenida. Así mismo se demostrará la

2. ANTECEDENTES

Desde la aparición de internet en 1969, el cambio que ha generado en las sociedades y en la tecnología son evidentes. Gracias a internet hemos conseguido mantener el contacto con todo el mundo, generando una red de conocimientos como jamás ha existido en la historia de la humanidad.

En los últimos años, se ha dado un paso más en esta dirección, y se han empezado a conectar dispositivos diversos a internet. A esta nueva rama tecnológica de internet se le ha denominado "Internet of things", o internet de las cosas en castellano. Y consiste en añadir a cualquier dispositivo la capacidad de compartir su estado a través de la Red hacia cualquier punto.

Actualmente existen diversas iniciativas para sacar el máximo partido de esta nueva revolución tecnológica, y una de ellas es la plataforma Open Source, The Things Network. Esta plataforma se basa en el protocolo LoRaWAN para dar acceso a cientos de sensores y actuadores a internet.

LoRaWAN a su vez usa tecnología LORA (Low-Power Wide-Area Network), la cual utiliza la modulación de frecuencia para transmitir paquetes de datos sencillos, a una distancia de más de 10 Km, con alta fidelidad y con un consumo mínimo, aproximadamente una batería cada 10 años.

Gracias a esto, The Things Network está intentando crear una red mundial de comunicaciones para IoT con el apoyo de usuarios independientes. Estos usuarios, instalan un Gateway que da cobertura a una amplia área. Dentro de esta, los usuarios pueden instalar los nodos que consideren necesarios y oportunos, y mediante Lora se comunicarán con los Gateways que servirán de pasarela para publicar los datos necesarios.

Actualmente la red creada por TTN es bastante amplia y cuenta en Valencia con varios puntos de acceso, sin embargo, la zona de Burjassot no cuenta con cobertura en esta red, y se requiere de un Gateway para el área que permita la creación de nuevos nodos en esta zona.

3. ESTUDIO DE NECESIDADES, FACTORES A CONSIDERAR: LIMITACIONES Y CONDICIONANTES

Para comenzar cualquier proyecto de esta índole lo primero es establecer los requisitos y necesidades, que delimiten el área de trabajo, marcando los objetivos y pautas a seguir en la realización del proyecto. Esto evitará desviarse del objetivo central del mismo.

El proyecto se divide en dos partes, por un lado, el desarrollo e implementación de la pasarela, y por otro el de los nodos. A continuación, se detallan los requisitos de ambas partes.

3.1 Conectividad y Red.

El punto de ubicación final del Gateway permite la conexión cableada mediante cable tipo ethernet, no obstante, debe existir la posibilidad de conexión remota mediante tecnología inalámbrica WiFi. Se plantea además la posibilidad de utilizar conexión 3G como solución alternativa, un método de conexión que suele generar menos inconvenientes que los que da el uso de WiFi

Respecto a la antena debe trabajar dentro de la Frecuencia Europea 868 MHz y tener un alcance mínimo de 10 km en condiciones óptimas.

En el caso de los sensores, requieren una antena adaptada a LORA que trabaje en frecuencia europea. Esto va a permitir conectar los sensores a largas distancias del Gateway sin necesidad de que existan redes inalámbricas o dispositivos bluetooth intermedios. Gracias a esto podremos instalar los nodos en garajes, zonas exteriores y espacios con baja cobertura. En nuestro caso la ubicación final será un parque infantil sin acceso a red WiFi.

3.2 Alimentación

Para el Gateway se van a implementar dos soluciones. Por defecto está habilitada la alimentación tradicional mediante adaptador que suministre 5V a 2.5A. Por otro lado, queda implementado dentro del Gateway la alimentación mediante Ethernet denominada PoE, usando el mismo cable para alimentación y transmisión de datos.

La solución adoptada buscará acometer a la envolvente con un único cable, por ello solo se alimentará de manera tradicional cuando el dispositivo se conecte mediante Wifi o 3G, en el resto de los casos la alimentación será proporcionada siempre por el cable de red.

Respecto a los nodos, la principal ventaja de usar tecnología LoRa es el bajo consumo en la transmisión de datos, de esta manera, tanto el microcontrolador y los sensores elegidos deben cumplir con los mismos requisitos de bajo consumo. La alimentación será por batería y deberá garantizar el funcionamiento autónomo durante un periodo de tiempo suficiente según su uso.

3.3 Condiciones de Funcionamiento

Tanto el nodo del parque infantil como el Gateway se ubicarán en exteriores, por ello es necesario en ambos casos una envolvente que garantice una IP65 como requisito mínimo. No obstante, y debido a la inclusión de un sensor de calidad del aire, que requiere un flujo constante de aire externo, la envolvente destinada al parque infantil se desarrollará sin cumplimiento de IP inicialmente. Se deberá garantizar aun así la máxima estanqueidad en las placas de control. Posteriormente se valorará el rediseño de una envolvente doble, permitiendo la circulación de aire, y garantizando la IP necesaria en las partes de críticas.

Los requisitos de funcionamiento de los nodos no son muy elevados, y son fácilmente reparables y remplazables, por lo que no se considera necesario ninguna monitorización ni requisito extra.

En el caso del Gateway, debido a su criticidad y condiciones más exigentes de funcionamiento para el microprocesador y el concentrador Lora,

como medida preventiva la envolvente deberá contar con una válvula para igualar la presión con el exterior y un inhibidor de corrosión para posibles condensaciones y filtraciones en la envolvente.

3.4 Versatilidad

El principal objetivo de este proyecto es la versatilidad. El proyecto tiene como fin ampliar la cobertura en la zona de Burjassot con el fin de poder desarrollar las soluciones que se consideren oportunas de manera sencilla y económica.

El Gateway debe ser versátil en cuanto a su instalación. Debe ser posible su montaje tanto en espacios cerrados como en exteriores; así como su conexión en zonas con punto de acceso a internet mediante cable de red, o en aquellas donde sola sea posible la conexión Wifi o en 3G en su defecto.

Los nodos deben ser totalmente versátiles en funcionalidad y capacidad. La envolvente debe poder alojar múltiples soluciones, el hardware estará diseñado con el fin de poder variar sensores y actuadores según se necesite. La programación debe seguir el mismo principio, siendo clara y modular, permitiendo el intercambio de periféricos solo modificando unas pocas líneas de código.

3.5 Magnitudes Físicas a medir

En el parque se van a controlar, la temperatura, la incidencia ultravioleta, la humedad y la calidad del aire. No se requiere alta precisión en las medidas, ya que será utilizadas como datos informativos para los vecinos, y no como datos representativos de un estudio.

Al tratarse de un parque infantil, el dispositivo solo transmitirá valores durante los momentos de utilización, acotamos por tanto el horario de funcionamiento entre las 9:00 de la mañana y las 20:00 de la tarde. Esto alargará la vida útil de las baterías, además de reducir el ruido generado en la transmisión de datos. Dentro de esta franja se publicarán los estados en periodos de 30 minutos.

Por tanto, será necesario un sensor de temperatura, sensor ultravioleta, sensor de humedad, sensor de partículas en suspensión.

4. PLANTEAMIENTO DE SOLUCIONES ALTERNATIVAS Y JUSTIFICACIÓN DE LA SOLUCIÓN ADOPTADA

4.1 GATEWAY:

Antes de entrar a la elección de la solución adecuada para generar nuestro Gateway, es importante analizar lo que nos ofrecen las soluciones profesionales que hay actualmente en el mercado. Entre estas soluciones hemos elegido LORIX One, un Gateway profesional para exteriores, el cual garantiza IP65 y trabaja en la frecuencia 868Mhz.

Lo primero que encontramos de este dispositivo son sus características:

- CPU ARM Cortex-A5 600MHz
- RAM 128 MBytes DDR2 200MHz
- NAND 512 MBytes flash with PMECC
- Linux Yocto 4.4
- Passive PoE 24V
- Ethernet 10/100
- SD-Card slot
- mini USB serial interface
- SX1301 gateway chip SPI based
- 8 channels, 49 demodulators @ 868 MHz
- RX Sensitivity -140 dbm
- TX power 27 dbm
- Gateway size 198 x 45 mm
- Packing size 530 x 60 x 140 mm (l x h x d)
- IP65 waterproof
- -30° to +55°C
- Hardware version 1.0d2

Como se ve, las especificaciones en lo que refiere al procesador no son muy exigentes.

La pasarela se monta en una SD, donde se ubica el sistema operativo, y se soporta en una distribución Linux denominada Yocto. Esta distribución es un proyecto basado en código abierto colaborativo, el cual permite a desarrolladores crear sistemas personalizados independientemente de la arquitectura del hardware.

En la alimentación del dispositivo nos encontramos con PoE, Power on Ethernet, que permite el suministro a 24 V desde un cable de red. Esta es una solución práctica, y que facilita garantizar el grado de IP al ser solo necesario un cable de acometida tanto para datos como para suministro de corriente.

En conectividad, echamos de menos la posibilidad de conexión 3G que, aunque sea más inestable siempre que el cable de Red facilita el montaje en ciertos entornos donde no es sencillo alcanzar con conexión cableada. Por otro lado, que la alimentación sea exclusiva sobre ethernet también nos limita en cierta medida la posibilidad de evitar el cable de red.

Dentro de la parte especifica de Lora, nos encontramos un Chip SX1301 de Semtech. Este es el corazón del Gateway, un procesador de señales que opera en las bandas ISM a nivel mundial. Más adelante se detallan las características de estos Chips. En este caso, y centrándonos en la funcionalidad del Gateway profesional que estamos tratando, este chip otorga 8 canales con un total de 49 demoduladores en una frecuencia de 828MHz. Esto permite escuchar simultáneamente a 1000 nodos transmitiendo datos cada 30 min.

La envolvente tiene un diseño atractivo y compacto, está desarrollada para su montaje en pared. Cuenta con una protección IP65 que garantiza su protección ante lluvias fuertes y grandes concentraciones de polvo. Cuenta con una antena de largo alcance desmontable.

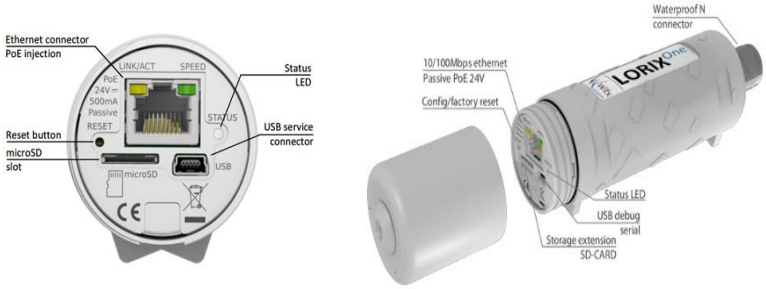


FIGURA 1 GATEWAY LORIX ONE

En conjunto vemos un dispositivo bastante interesante, cumple todos los requisitos para generar una pasarela LoRaWAN que nos permita conectar nuestros nodos a la Red. Pero en un análisis más detallado, vemos un producto caro unos 400 euros al cambio, con muy poca versatilidad debido a su diseño, y con un Hardware bastante limitado, que, si bien cumple su función, va a ser un impedimento en otras implementaciones o posibles mejoras y adaptaciones de este dispositivo.

Por ello vemos más positivo, tanto a nivel formativo, como a nivel usuario final, desarrollar nuestro propio dispositivo basado en un Chip de Semtech en conjunto con un microprocesador con características superiores a las que ofrece el dispositivo de Lorix. Esto nos va a permitir ir incorporando funcionalidades y mejoras a nuestro Gateway sin estar limitados en Hardware. Cualquier Raspberry Pi monta un núcleo ARM más potente e incluso con mayor RAM.

4.1.1 Microprocesador:

Para el Gateway no es suficiente con un microcontrolador como Arduino, es necesario una potencio que solo ofrece un microprocesador. En este apartado existen diversas soluciones a valorar.

- *Beaglebone:*

Beaglebone es la propuesta de Texas Instrument a la Raspberry Pi. Se desarrollo como un proyecto Open Source de bajo consumo destinada al uso con fines educativo. No obstante, y al ser un proyecto de Texas Instruments, puede ser más aceptado su uso a nivel industrial, que el de una Raspberry PI.

Memoria

Actualmente en el mercado encontramos 4 placas, se puede ver su detalle en la Tabla 1.

	BeagleBoard.org BeagleBone Black	SeeedStudio BeagleBone Green	BeagleBoard.org BeagleBone Blue	SanCloud BeagleBone Enhanced
Processor	AM3358 ARM Cortex-A8	AM3358 ARM Cortex-A8	AM3358 ARM Cortex-A8	AM3358 ARM Cortex-A8
Maximum Processor Speed	1GHz	1GHz	1GHz	1GHz
Analog Pins	7 (1.8V)	7 (1.8V)	4 (1.8V)	7 (1.8V)
Digital Pins	65 (3.3V)	65 (3.3V)	24 (3.3V)	65 (3.3V)
Memory	512MB DDR3 (800MHz x 16), 2GB (4GB on Rev C) on-board storage using eMMC, microSD card slot	512MB DDR3 (800MHz x 16), 4GB on-board storage using eMMC, microSD card slot	512MB DDR3 (800MHz x 16), 4GB on-board storage using eMMC, microSD card slot	1GB DDR3 (800MHz x 16), 4GB on-board storage using eMMC, microSD card slot
USB	miniUSB 2.0 client port, USB 2.0 host port	microUSB 2.0 client port, USB 2.0 host port	microUSB 2.0 client port, USB 2.0 host port	miniUSB 2.0 client port, 4 USB 2.0 Ports (2 A-type connectors, 2 on pin headers)
Video	microHDMI, cape add-ons	cape add-ons	SPI displays	microHDMI, cape add-ons
Audio	microHDMI, cape add-ons	cape add-ons	add-ons, Bluetooth	microHDMI, cape add-ons
Supported Interfaces	4x UART, 12x PWM/Timers, LCD, GPMC, MMC1, 2x SPI, 2x I2C, A/D Converter, 2x CAN Bus (w/o PHY)	4x UART, 12x PWM/Timers, LCD, GPMC, MMC1, 2x SPI, 2x I2C, A/D Converter, 2x CAN Bus (w/o PHY) 2 Grove (I2C, UART)	4x UART, 2-cell LiPo, 2x SPI, I2C, 4x A/D converter, CAN bus (with PHY), 8x 6V servo motor, 4x DC motor, 4x quadrature encoder	4x UART, 12x PWM/Timers, LCD, GPMC, MMC1, 2x SPI, 2x I2C, A/D Converter, 2x CAN Bus (w/o PHY)
Sensors	n/a	n/a	Barometer, Accelerometer, Gyro, Temperature	Barometer, Accelerometer, Gyro, Temperature
MSRP	$49	$39	$79	$69

Todas montan el mismo procesador, y se diferencian en funcionalidades extras, más pines, o incluso más RAM. En nuestro caso vamos a analizar la Beaglebone Black.

TABLA 1 COMPARATIVA MODELOS BEAGLEBONE

Como se observa en la tabla 1, la opción Black está disponible por un precio muy reducido, teniendo en cuenta lo que ofrece. La BeagleBone black cuenta con un procesador ARM Cortex-A8 con 1GHz de reloj, algo menor que una Raspberry Pi 3B, pero suficiente para tareas de computo medias. En memoria queda algo justa con 512 Mb de RAM, pero en compensación ofrece una gran ventaja frente a otros microprocesadores, y es la inclusión de una memoria interna flash de 2 Gb además del soporte microSD.

En conectividad contamos con cliente miniUSB 2.0 y un solo puerto USB 2.0 que nos limita ligeramente en periféricos Plug and Play. No obstante, es bastante versátil respecto a interfaces, contando con 4 puertos serie UART, 12 pines de potencia modulada, pantalla LCD, MMC1, GPMC, 2 puertos SPI, un

convertidor analógico digital, e incluso 2 puertos para CAN. Aunque a nivel GPIO lo más destacable son los 65 pines digitales a 3.3V con los que cuenta.

En general es un dispositivo a tener en cuenta, algo más reciente que la Raspberry Pi, y por tanto con una comunidad aun en desarrollo, es una solución muy potente y respaldada por una empresa tan consolidada en la electrónica como es Texas Instruments.

- *Raspberry Pi:*

Hablar de Raspberry Pi ya no es hablar de un microprocesador, al igual que Arduino en su área, Raspberry se ha convertido en un referente del desarrollo Open Source. La comunidad generada en torno a este pequeño ordenador es inmensa, existiendo miles de proyectos y millones de usuarios dando soporte y apoyo a los nuevos desarrolladores.

En nuestro caso vamos a hablar de un producto concreto de Raspberry, la Raspberry Pi 3B. Es la tercera generación de tarjetas producidas por la marca, en la cual se mantiene las mismas dimensiones que sus predecesoras, pero con importantes cambios en hardware.

Esta placa monta un procesador Broadcom de cuatro núcleos ARM Cortex-A53 y memoria RAM de 1GB LPDDR2. Junto a esto se incorpora un doble núcleo de procesamiento de video que dota a la placa de soporte Open GL ES2.0 y aceleración por Hardware OpenVG.

En lo que respecta a conectividad, la placa cuenta con conexión Ethernet, Wifi 802.11 b/g/n, bluetooth 4.1. Tiene conexiones para 27 pines GPIO de hasta 5V, soporte para I2C, SPI, UART. Como interfaces físicas cuenta también con salida de audio tipo Jack, conexión HDMI, 4 puertos USB 2.0, conexión MIPI de 15 pines para cámara y DSI de 15 pistas para monitores. En la Figura 2 Placa Raspberry Pi 3BFigura 2 se detallan estos periféricos.

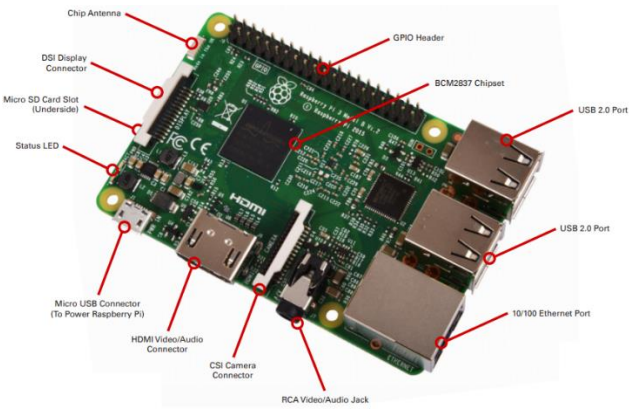


FIGURA 2 PLACA RASPBERRY PI 3B

Esta solución destaca por su bajo coste, su capacidad de procesamiento en un tamaño tan reducido y la versatilidad de su conectividad. Así, es válida para soluciones tan diversas como domótica, centros multimedia, un servidor, un PC completo y funcional o aplicaciones IoT.

Como cualquier ordenador, es necesario un sistema operativo para poder operar nuestra placa, Raspberry ha desarrollado para esto una versión propia en base a Linux denominada Raspbian, que distribuye de forma totalmente gratuita. En caso de buscar otras opciones Raspberry puede operar con muchos sistemas operativos entre ellos la reciente versión de Windows para dispositivos de pocos recursos, Windows 10 IoT.

En lo que respecta a la programación, Raspberry foundation aconseja el uso de Python, pero se puede programar en C, C++, Java o Ruby entre otros.

Se ha decidido finalmente el uso de este microcontrolador frente a BeagleBone por la comunidad que existe y el número de usuarios alrededor de Raspberry Pi. Como se ha mencionado, BeagleBone es un proyecto relativamente reciente, y por tanto cuenta con un soporte y una documentación más limitada frente a Raspberry.

4.1.2 Concentrador Lora

Antes de entrar a la elección del concentrado más adecuado para nuestra pasarela, es importante describir el corazón de estas tarjetas, el chip desarrollado por Semtech que se encarga de procesar las señales recibidas de cada nodo.

Actualmente en el mercado se encuentran principalmente dos modelos, el SX1301 y el SX1308. Ambos son bastante similares, su principal diferencia reside en la sensibilidad, el primero cuenta con una sensibilidad de –142.5 dBm, mientras que el segundo se queda en los –139 dBm.

Atendiendo al mercado, nos encontramos que existen muy pocas soluciones en las que se aplique el SX1308, por lo que por sencillez y soporte hemos decidido usar tarjetas que monten el Chip SX1301.

Este chip es un procesador masivo de señales digitales de largo alcance para comunicación ISM. El receptor del chip acepta flujo de bits digitalizados desde uno o dos receptores, posteriormente des modulariza la señal y almacena los datos en paquetes que cumplen regla FIFO. Estos paquetes quedan a disposición del MCU.

En el sentido inverso, los paquetes se modulan mediante el modulador programable (G)FSK/Lora. y se envían a uno de los transmisores.

El control del Chip mediante el MCU se realiza usando una capa de abstracción de Hardware (HAL), la cual facilita Semtech para su rápida implementación. En la Figura 3 se puede observar el modelo de bloques que detalla esta funcionalidad.

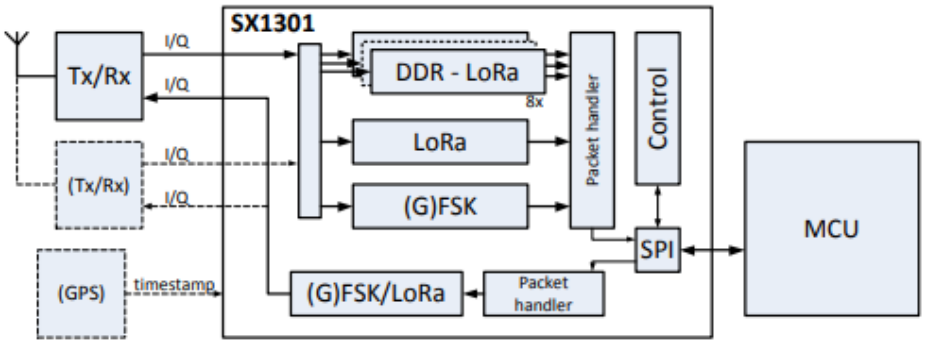


FIGURA 3 DIAGRAMA DE BLOQUES CHIP SX1301 FACILITADO POR SEMTECH

En el mercado existen diversas placas que montan este chip que, aunque podría ser usado directamente con unos condensadores de desacople, algunas resistencias y un par de receptores de Semtech, facilitan mucho su implementación junto a un microprocesador para levantar una pasarela.

A continuación, hablamos de 4 posibles soluciones para nuestro desarrollo.

- Multitech mCard

Esta tarjeta ha sido desarrollada por Multitech, una empresa dedicada al desarrollo de equipos de comunicaciones para el internet industrial de las cosas.

En su misión de desarrollar equipos innovadores, integrales y flexibles, han creado una serie de tarjeta, denominadas MultiConnect mCard. Estas tarjetas otorgan flexibilidad a diversas infraestructuras dotándolas de un amplio espectro de funcionalidades y protocolos según las tarjetas escogidas.

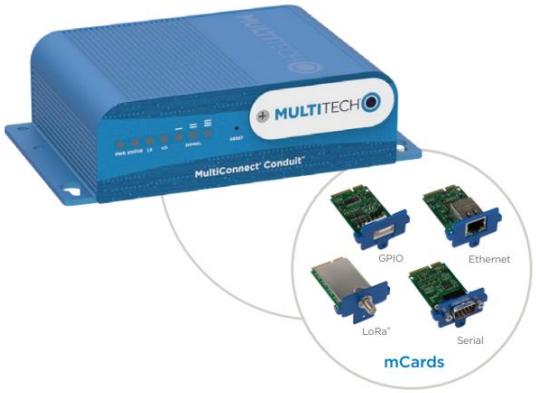


FIGURA 4 TARJETAS MCARDS OARA GATWAY MULTITECH

En nuestro desarrollo, la tarjeta adecuada seria la MTAC-LORA-H-868. Es una tarjeta de muy bajo consumo y conectividad de largo alcance. Posee Escaneo de banda ISM como todos los

concentradores basados en SX1301 y un rango de administración de activos LoRa de hasta 15km en línea de visión, o 2km a través de edificios. Además, cuenta con capacidad LBT, escuchar antes de hablar.

El software se basa en una distribución Linux, mLinux y se facilita por el fabricante el firmware necesario para su implementación.

Como se puede ver en la figura 4, esta tarjeta está diseñada para añadirse al enrutador modular de Multitech. Así pues, para nuestro desarrollo será necesario adquirir además un convertidor de USB a Mini PCIe para montar la mCard, este deberá ser diseñado o adaptado desde uno comercial para asegurar su funcionamiento. Las dimensiones además son algo elevadas (92mm x 35mm)

Esto complicaría bastante la implementación, además va a generar un ruido extra en la comunicación y posibles fallos posteriores.

A esta complicación, hay que añadir un precio elevado, 168€ en Digi-Key que se incrementa más aun al requerir de un adaptador PCIe .

Descartamos esta solución al no ser adecuada ni en diseño ni en precio a nuestro desarrollo.

- *RAK831*

Esta tarjeta ha sido desarrollada por la empresa China RAK Wireles, y al contrario que la desarrollada por Multitech, no está enfocada a usarla en un producto concreto, sino que es mucho más versátil y modular.

Las características principales son similares, ya que monta un Chip SX1301 como la anterior. Se pueden ver dimensión reducidas (80mm x 50mm), sensibilidad de hasta –142,5 dBm a una velocidad de transferencia de 293bps, interfaz de control mediante SPI, firmware de Semtech con Gateway HAL y alcance de hasta 15km.

La tarjeta permite como se ha dicho la comunicación mediante SPI o USB con el controlador, esta empresa ofrece además una serie de Kits de desarrollo para generar una pasarela de manera bastante sencilla y rápida. Entre estos kits destaca una placa adaptadora que permite montar el RAK831 directamente sobre una Raspberry Pi.

En internet es fácil localizar bastante información y soporte para esta tarjeta de usuarios independientes, lo que junto a estos kits que facilita la propia compañía, es bastante rápido generar una pasarela funcional.

FIGURA 5 RAK 831 JUNTO A SHIELD DE CONEXIÓN PARA RASPBERRY PI

Esta tarjeta cuenta con un precio en principio inferior a la tarjeta de Multitech, entorno a los 120€. Sin embargo, tiene un gran problema de distribución, y solo puede ser adquirida actualmente en tiendas chinas a través de plataformas como Ali Express, lo que implican tiempos y precios a través de aduanas bastante elevados.

Se descarta el uso de esta tarjeta por los inconvenientes de su importación y su bajo soporte a través del fabricante oficial al no contar con sede en Europa.

- *RHF0M301*

Nos encontramos ante una tarjeta de tamaño reducido (63mm x 40mm) y con el mismo chip que las anteriores.

En este caso la tarjeta nos ofrece múltiples opciones de banda, desde 434MHz hasta 915Mhz, pasando por los 868MHz que marca la normativa europea.

Cuenta con 10 caminos de demodulación paralelos, lo que permite el tratamiento de múltiples nodos simultaneaos de manera eficiente a través de sus 8 canales activos.

Cuenta además con factor de dispersión adaptativo desde SF12 a SF7 en cada uno de sus canales, consiguiendo también una sensibilidad de –142.5 dBm a una velocidad de 300 bps, algo mayor que la ofrecida por RAK831.

Soporta protocolos LoRaWAN de clase A, B y C. Sin embargo, una búsqueda en la web no nos aporta demasiado soporte ni una comunidad desarrollada que trabaje con este producto. No es sencillo encontrar un dataSheet de la placa, y además nos encontramos con que solo es posible comprarla a través de China, y no se encuentra fácilmente. No hemos sido capaces de localizar un precio al buscar información de ella.

Estamos por tanto ante una placa muy potente y en dimensiones muy reducidas, siendo de las más pequeñas en estas características, un factor a veces muy decisivo en la elección de un componente electrónico. Pero que tiene una logística compleja, y una comunidad de desarrolladores reducida.

FIGURA 6 PLACA RHF0M301

- IC880A

Esta tarjeta es la propuesta de la empresa alemana IMST para adaptar el Chip SX1301.

Al igual que las demás tarjetas nos ofrece todas las ventajas del Chip SX1301, aunque es cierto que en un tamaño algo mayor (79.8 x 97.3 mm)

También nos encontramos con la limitación, que no nos afecta en nuestro desarrollo al trabajar em la frecuencia europea, que solo soporta la banda 868 MHz.

En el resto de las características tenemos una placa muy potente que cumple todos los requisitos para desarrollar una pasarela funcional y adecuada para nuestro desarrollo. Cuenta con largo alcance, hasta 15km en línea de visión, factores de dispersión independientes en cada uno de los 8 canales, 10 líneas de demodulación en paralelo, dos receptores Sx1257, soporte SPI y USB y alimentación independiente.

En lo que respecta al Firmware, contamos con una colaboración de Semtech, IBM y Actility en su desarrollo. Junto a esto hay un repositorio en Git con todo el software y Firmware necesario para un proyecto con Lora-net. En este proyecto se encuentran todas las dependencias necesarias para nodos, el cliente del Gateway soportado por los drivers HAL de Semtech y la implementación de un servidor.

Aunque es cierto que estamos ante una placa algo más grande, y menos potente que las versiones de Multitech o las opciones chinas, estamos ante un producto más económico unos 119€, y de fácil importación al ser producto fabricado en Europa.

También contamos con una comunidad de desarrolladores amplia y con soporte suficiente para poder solventar pequeños problemas surgidos durante la implementación.

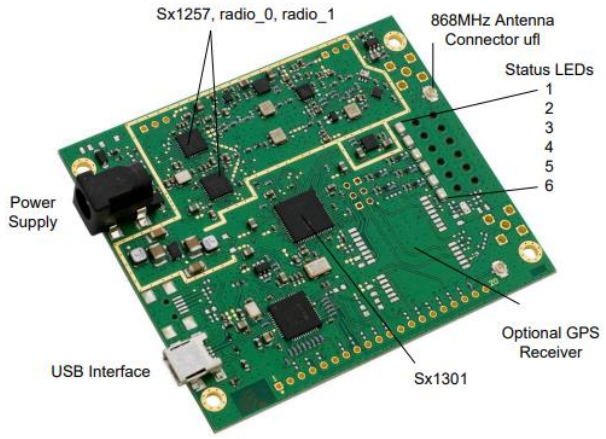


FIGURA 7 PLACA IC880A DE IMST

Como son productos muy similares, la decisión para elegir la tarjeta se ha basado fundamentalmente en el soporte y en la fácil adquisición de esta tarjeta frente a sus competidoras.

4.2 NODOS.

Para el desarrollo del nodo hay tres elementos a analizar, el microcontrolador, el módulo LoRaWAN y los sensores, en nuestro caso estos últimos serán de temperatura, humedad, luz ultravioleta y partículas en suspensión.

Existen dos soluciones posibles a la hora de escoger los componentes para un nodo, por un lado, está la opción de usar un microcontrolador tradicional y conectar mediante SPI un transceptor LoRa. Por otro lado, se pueden conseguir placas que llevan incorporado ya este transceptor y están totalmente preparadas para generar un nodo IoT con LoRaWAN de manera simple. Por simplicidad, y fiabilidad se ha decidido contar con soluciones que monten directamente los transceptores LoRa,

4.2.1 PLACAS DE DESARROLLO CON TRANSCEPTOR:

Esta opción es muy interesante ya que podemos empezar a trabajar de manera inmediata con nuestro nodo. Estas soluciones suelen facilitar todo el firmware y las librerías para hacer funcionar el nodo de manera rápida. El inconveniente que encontramos es que perdemos versatilidad y a veces pagamos por elementos que realmente no necesitamos. Aun así, las consideramos la mejor alternativa para el enfoque de este proyecto.

- TTN-UN-868

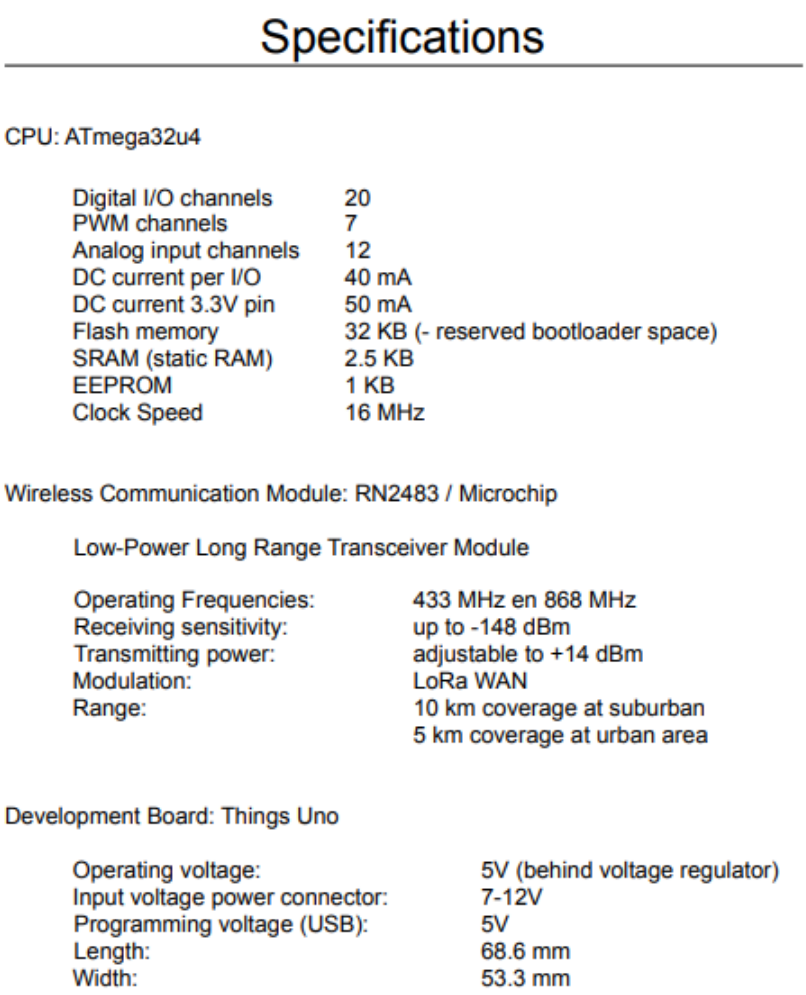
Esta placa se basa en la solución Leonardo de Arduino a la que se le ha incorporado un módulo de procesamiento LoRaWAN RN2483 de la empresa Microchip. Viene por defecto con una antena interna, aunque está preparada para añadirle una externa mediante conector SMA soldándolo a la placa.

Está equipada, tal y como se recoge sus especificaciones de la 8, con 12 canales analógicos de entrada, lo que sería más que suficiente para los sensores que necesitamos conectar, 20 canales digitales y 7 de modulación de pulsos, aunque no vamos a requerir de ellos en la solución que queremos desarrollar.

El firmware y la librería las proporciona directamente Microchip para el protocolo de LoRaWAN. Por otro lado, la placa está preparada para trabajar junto al IDE de Arduino y es compatible con todos los Shields y desarrollos compatibles con los dispositivos Arduino UNO.

Esta placa se encuentra disponible en FARNELL actualmente por 48 € un precio bastante razonable si tenemos en cuenta que el Arduino Leonardo "Oficial" se encuentra actualmente por unos 20€ y un transceptor ronda igualmente esa cantidad.

FIGURA 8 PLACA DE DESARROLLO TTN-UN-868



Specifications

CPU: ATmega32u4

Digital I/O channels	20
PWM channels	7
Analog input channels	12
DC current per I/O	40 mA
DC current 3.3V pin	50 mA
Flash memory	32 KB (- reserved bootloader space)
SRAM (static RAM)	2.5 KB
EEPROM	1 KB
Clock Speed	16 MHz

Wireless Communication Module: RN2483 / Microchip

Low-Power Long Range Transceiver Module

Operating Frequencies:	433 MHz en 868 MHz
Receiving sensitivity:	up to -148 dBm
Transmitting power:	adjustable to +14 dBm
Modulation:	LoRa WAN
Range:	10 km coverage at suburban 5 km coverage at urban area

Development Board: Things Uno

Operating voltage:	5V (behind voltage regulator)
Input voltage power connector:	7-12V
Programming voltage (USB):	5V
Length:	68.6 mm
Width:	53.3 mm

FIGURA 9ESPECIFICACIONES TTN-UN-868 RECUPERADAS DE LA PAGINA OFICIAL DE TTN

- TTN-ND-868

Este dispositivo se puede considerar un Nodo Plug and Play. El nodo incorpora el microcontrolador, el módulo LoRa, una antena de largo alcance, sensores de temperatura, luminosidad y un acelerómetro.

Es el dispositivo perfecto para iniciarse y realizar pruebas del Protocolo con algunos sensores sin tener que preocuparte del desarrollo del Nodo. El dispositivo incluye todas las bibliotecas necesarias y en menos de una hora se puede tener el dispositivo operativo y transmitiendo.

El precio de este dispositivo, aunque nos ofreces diversos sensores, es algo elevado, unos 55€, si tenemos en cuenta que además estamos sacrificando versatilidad y limitándonos a los que nos ofrece este nodo.

Para nuestro proyecto queda totalmente descartado ya que no nos permite medir todos los parámetros físicos que queremos monitorizar con nuestro Nodo. No obstante, es una opción muy interesante y a tener en cuenta para proyectos concretos o si simplemente se busca

iniciarse en el protocolo rápidamente, ya que nos ahorra bastante tiempo de programación y montaje.

FIGURA 10 TTN-ND-868

- ARDUINO MKR WAN 1300.

Lo primero antes de comenzar a hablar de Arduino, es entender que no es un microcontrolador como tal, sino que es una placa de desarrollo que monta un microcontrolador, la placa más común de esta plataforma Arduino UNO se basa por ejemplo en el chip ATMega328 de Microchip.

Arduino nació como un pequeño proyecto para el instituto de diseño interactivo IVRA en Italia y actualmente ha vendido millones de unidades, así como ha desarrollado placas específicas para usos muy diversos.

La grandeza de Arduino reside sobre todo en ser un proyecto de código abierto, no solo en Software sino en Hardware, permitiendo a cualquiera crear y desarrollar placas basadas en Arduino sin incurrir en delito de plagio. Esto ha permitido crear una comunidad de creadores y diseñadores a su alrededor, la cual ha propiciado aún más el desarrollo de esta tecnología.

La placa de desarrollo por excelencia de Arduino es la Arduino UNO, siendo la placa más vendida y replicada. Pese a no ser la más potente de la familia, monta un chip CMOS como se ha mencionado anteriormente, con una RAM muy reducida, una EEPROM limitada y una memoria flash de poca capacidad. Pese a estas limitaciones, es más que suficiente para el desarrollo de sistemas embebidos simples y soluciones IoT como la que planteamos. Contamos con interfaces de salida y entrada suficientes, y varias salidas de pulso modulado, además de soporte para comunicación tanto serial como SPI.

Dentro de las soluciones que propone Arduino, encontramos una familia completa dedicada a la mencionada comunidad MAKER, y que lleva este nombre como distintivo, son las placas Arduino MKR.

Estas placas montan microcontroladores ATMEL SAMD21 y mantienen el formato reducido de las Arduino Nano. Dentro de esta familia vamos a analizar la solución específica para desarrollos LoRa, es la Arduino MKR WAN 1300.

Como se ha mencionado, contamos con un ATMEL SAMD21, en este caso un Cortex-M0 que trabaja junto a un MCU de bajo consumo. Como añadido esta placa cuenta con un módulo de

radio para LoRa CMWX1ZZABZ de Murata. Un punto negativo en esta placa podría tomarse su baja disponibilidad de interfaces de conexión, disponemos de 12 pines PWM en los que se soporta 1 conexión UART, 1 SPI y 1 I2C. No obstante, estas pueden ser suficiente para la mayoría de desarrollos. En caso de no ser así, Arduino cuenta con un amplio rango de dispositivos de expansión o Shields, los cuales nos aportan mayor versatilidad.

En rendimiento contamos con una velocidad de reloj de 48 Mhz, una memoria flash limitada de 256 kB, y una SRAM de tan solo 32kB. Aunque suficiente para proyectos sencillo, y de bajo consumo como los protocolos LoRa, esta placa podría estar muy limitada para desarrollos más complejos.

FIGURA 11 ARDUINO MKR WAN 1300[OI]

- PYCOM LoPy4

Este dispositivo viene de la mano de la empresa Pycom. Esta empresa se dedica a crear soluciones completas ofreciendo desde el Hardware, hasta su propia plataforma e IDE. Las placas se programan en mycropython un desarrollo en C a través de Python3, optimizado para su implementación en microcontroladores.

Python es considerado actualmente uno de los más potentes lenguajes para IoT debido a su sencillez y su gran comunidad que se ha generado gracias al entorno MAKER.

La opción que propone PYCOM para LoRa es LoPy4 un microcontrolador construido en 4 tipos de conexiones diferentes LoRa, Bluetooth, WiFi y Sigfox. Aunque en nuestro caso solo requerimos de LoRa, es muy interesante que nos de todas las opciones en un único dispositivo, dotándolo así de una gran versatilidad y abriendo puertas a futuros desarrollos y ampliaciones funcionales.

Como podemos ver en sus características, estamos con un dispositivo potente, increíblemente versátil con sus cuatro modos de conexión, entradas analógicas y digitales, así como salidas PWM suficientes para cualquier desarrollo IoT mediante LoRa. Y todo esto con un consumo y dimensiones muy reducidas. También contamos con interfaces 2 x UART, SPI, 2 x I2C, I2S y tarjeta microSD

Esta placa nos brinda una RAM elevada para un microcontrolador, hasta 4MB, y una Flash externa de 8Mb. Lo que más destaca de esta placa es sus reducidas dimensiones 55mm x 20mm x 3.5mm y su bajo consumo, llegando a 15-uA con el modo“deep_sleep”

El precio además es muy adecuado si lo comparamos con otras soluciones, unos 35€ en la página oficial.

Si es necesario añadir una pega a este producto es la necesidad de adquirir una placa de expansión para conectarlo al PC y alimentarlo de manera adecuada a través de batería. Actualmente tiene varias soluciones para este problema, la más sencilla es la EXPANSION BOARD 3.0 que viene con los controladores necesarios e incluye un circuito para el control y carga de batería LiPo.

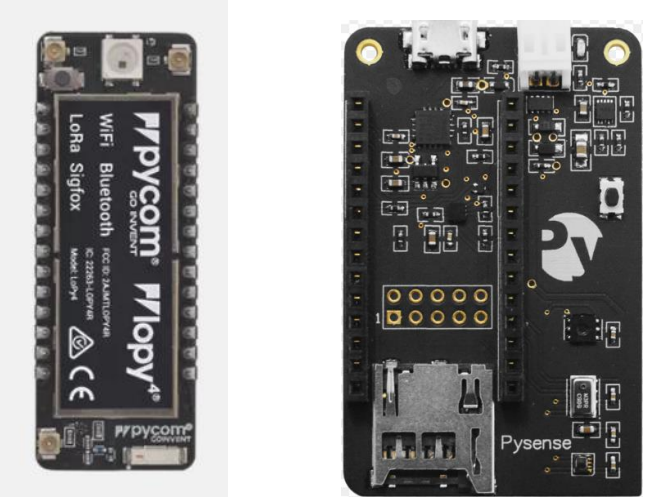


FIGURA 12 LOPY 4 (IZQUIERDA) Y TARJETA DE EXPANSIÓN PYSENSE (DERECHA)

Debido al enfoque MAKER que se le pretende dar a este proyecto, nos hemos decidido por esta placa sobre la solución más profesional que nos ofrece St. También la preferimos a la solución de TTN por su versatilidad, menor consumo y dimensiones reducidas. Respecto a la Arduino MKR, hemos elegido la solución de PYCOM por el interés en el lenguaje Python3, y en este caso su versión para microcontroladores Micropython.

En lo que respecta a la placa de expansión para esta tarjeta, hemos decidido contar con el modelo PYSENSE, que cuenta con sensores de presión barométrica, temperatura, humedad y luz, lo que nos va a servir para el prototipo de nuestro Nodo sensor.

- ST B-L072Z-LRWAN1

Desde St se ofrece una solución más profesional que las propuestas hasta el momento, aunque en una versión destinada al testeo y desarrollo de soluciones mediante LoRaWAN. St es una empresa que busca dar el máximo soporte a estudiantes y creadores mediante sus placas

de desarrollo que combinan diferentes tecnologías con interfaces de conexión y elementos de testeo dentro de una misma placa.

En este caso vamos a hablar de la placa B-L072Z-LRWAN, una herramienta de desarrollo perfecta para iniciarse en el protocolo LoRa. Monta un chip de Murata CMWX1ZZABZ, controlado por un núcleo STM32 y gracias a un transpondedor LoRa SX1276 permite la modulación de frecuencias de largo alcance.

Con todo, entre sus características destacamos su núcleo ya mencionado, el cual se basa en un M0+ de ARM, una RAM de 20Kbytes y una flash de 192Kbytes. Todo esto construido sobre el principio de ultra bajo consumo.

Respecto a interfaces de conexión nos encontramos ante un dispositivo muy completo enfocado al desarrollo versátil de múltiples soluciones. Así, nos encontramos con 4 canales, 12-bit ADC, 2 DAC, 6-bit timers, LP-UART y SPI, 7 LEDs integrados, 2 pulsadores e incluso una interfaz para interconectar una placa Arduino Uno V3 sobre él.

En lo que respecta a la programación, encontramos también una gran versatilidad como se podía esperar de una placa de desarrollo enfocada a prototipado. El lenguaje por defecto es C, y contamos con 4 toolchains diferentes contra los que programar. Keil con MDK-ARM, cuenta con versión de estudiantes, es una herramienta de desarrollo potente, aunque algo compleja en sus inicios. IAR EWARM, entorno profesional para programación de microcontroladores basados en núcleos ARM, no cuenta con versión de estudiantes y tiene un precio algo elevado. ARM Mbed, un entorno de desarrollo para microcontroladores web, una herramienta muy sencilla e intuitiva, falla en herramientas de compilación más avanzadas para el testeo de aplicaciones. Además de esto podemos usar cualquier IDE que se base en GCC.

En general nos encontramos ante una placa versátil y potente, aunque algo más compleja tanto en programación como en configuraciones. No obstante, esta placa puede ser algo compleja para desarrolladores independientes o sin conocimientos avanzadas de programación. Como el proyecto pretende poner de manifiesto la posibilidad de desarrollo particulares sencillos dentro del IoT, se ha descartado el uso de esta placa.

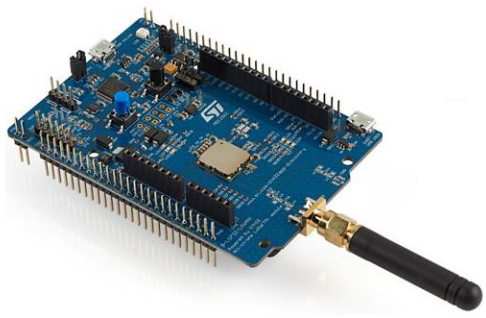


FIGURA 13 B-L072Z-LRWAN1 DE ST

5. DESCRIPCIÓN DETALLADA DE LA SOLUCIÓN ADOPTADA.

En el diagrama de la figura 14 se puede observar el funcionamiento general de la solución adoptada. Aunque a partir de ahora nos vamos a centrar en el desarrollo del prototipo para el parque infantil, en las siguientes líneas desarrollamos como seria una solución completa con los componentes escogidos.

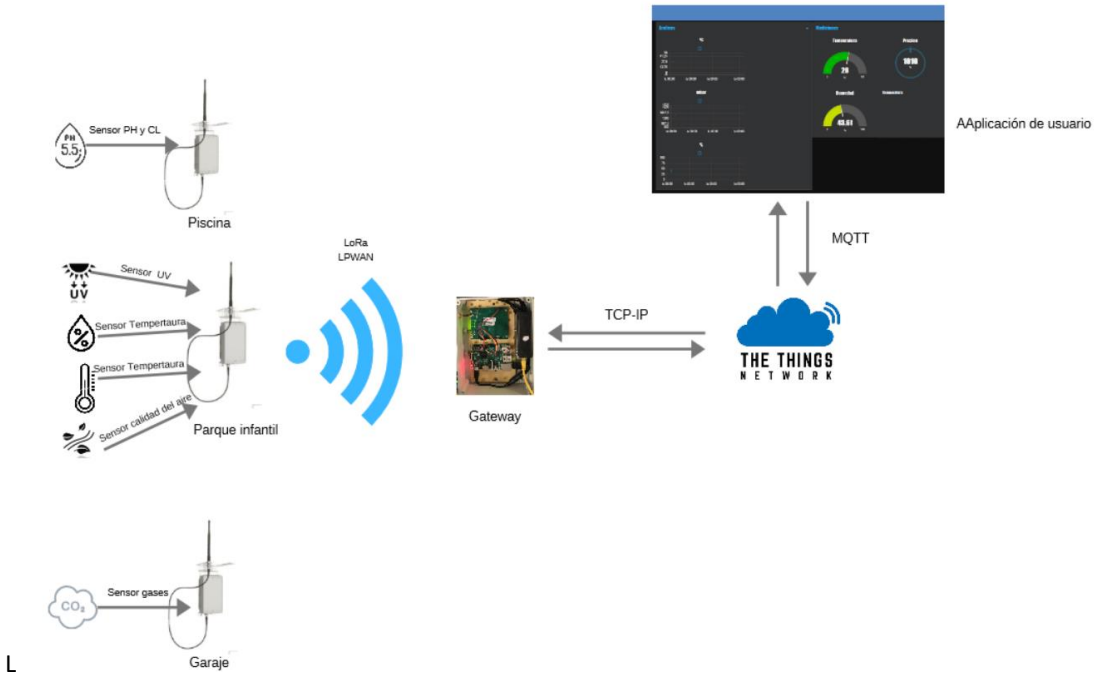
La instalación de una solución completa en una comunidad de vecinos consta de 3 nodos completos, un Gateway y la aplicación web alojada en un servidor.

Cada Nodo es individual y se destina a un objetivo concreto. Para el desarrollo de estos nodos finalmente se ha decidido hacer uso de la placa LoPy, es una opción económica, potente y con la versatilidad que buscamos para poder usarse en cualquiera de los nodos que implantemos. Esta tarjeta va a ser la encargada de solicitar y recibir el estado de cada sensor, convertirlo en una trama binaria y transmitirla al Gateway mediante LoRa LPWAN.

El Gateway ha sido desarrollado por nosotros descartando la opción comercial ofrecida por Multitech. Esta elección nos permite ahorrar en precio ganando en versatilidad y capacidad. Para desarrollarlo finalmente hemos decidido contar con la tarjeta LoRa IC880a de IMTS, como ya mencionamos antes, el principal motivo ha sido la sencillez de suministro al ser un proveedor europeo. Para controlar esta tarjeta se ha optado finalmente por la Raspberry pi3, su elección sobre la Beaglebone se ha debido principalmente al soporte y la gran comunidad de desarrolladores con la que cuente Raspberry.

El Gateway por tanto será el encargado de recibir los mensajes de cada uno de los nodos. Mediante protocolo TCP/IP irá pasando estos mensajes al servidor europeo de la plataforma TTN.

En este punto, se hace uso de una de las herramientas que nos ofrece dicha plataforma, permitiéndonos descifrar el mensaje binario recibido y darle formato adecuado para pasarlo a través de MQTT. Haciendo uso así de este protocolo, pasamos los datos recibidos a un panel WEB desarrollado mediante node-red. En este punto la información es mostrada de manera visual con una interfaz limpia y sencilla que recoja los datos obtenidos por cada uno de los nodos.



L

FIGURA 14 DIAGRAMA GENERAL DEL SISTEMA

5.1 Protocolo de Comunicaciones

Nuestro proyecto se basa en establecer la comunicación entre un nodo sensor, y una aplicación final de usuario. Para ello se va a pasar a través de 3 protocolos de comunicación diferentes. LoRaWAN desde el sensor al Gateway, TCP/IP desde el Gateway hacia la plataforma TTN, y finalmente MQTT para el paso de los datos de TTN a la aplicación final.

El protocolo TCP/IP es un protocolo muy establecido y de sobra conocido, por lo que, aunque se hace uso de el en este proyecto y es muy importante, no vamos a entrar en detalle en este proyecto.

Si que vamos a hablar de MQTT como uno de los protocolos que más se está usando en la IoT, y sobre todo de LoRaWAN, base de este proyecto.

5.1.1 LoRaWAN

Para hablar del protocolo LoRaWAN, lo primero es definir que es LoRa. LoRa es el tipo de modulación en frecuencia utilizada para establecer comunicaciones inalámbricas a larga distancia.

La mayoría de sistema de comunicación inalámbricos se basan en la modulación por desplazamiento de frecuencia, o FSK por sus siglas en inglés. Esta técnica de modulación se basa una señal moduladora digital, en la cual cada valor binario se representa con frecuencias diferentes f1 y f2 próximas ambas a la señal portadora:

$$v(t) = \begin{cases} V_p \ sen(2\pi \ f_1 \ t) \text{ para un "1" binario} \\ V_p \ sen(2\pi \ f_2 \ t) \text{ para un "0" binario} \end{cases}$$

Esta técnica de modulación es altamente eficiente en consumo de recursos. No obstante, LoRa se basa en una técnica diferente y optimizada para el largo alcance a la que se denomina "Espectro de propagación de chirrido" o CSS por sus siglas en inglés. Esta técnica utiliza pulsos de chirrido para codificar la información, se denomina chirrido a una señal sinusoidal que varía su frecuencia en función del tiempo. Este tipo de modulación mantiene la misma eficiencia que la FSK, pero otorga un rango de alcance mucho mayor, por ello ha sido usada en la comunicación militar y espacial durante años, LoRa es el primer uso comercial que se le da a esta técnica.

La ventaja principal de LoRa reside en su largo alcance frente a su bajo consumo, con un único Gateway es posible cubrir ciudades o zonas de varios kilómetros cuadrados. En la Figura 15 se observa un mapa de la cobertura elaborado por la compañía Proximus. En la actualidad cuentan con cobertura en todo el país, algo sorprendente que han desarrollado en muy pocos años

FIGURA 15 MAPA DE COBERTURA LORA EN BELIGICA OBTENIDO DE LA PAGINA WEB DE LA COMPAÑÍA PROXIMUS.

LoRa, por tanto, es la tecnología de modulación a usar, y es una de las usadas para el transporte de datos mediante LPWAN, o redes de área amplia y baja potencia en castellano.

En la actualidad podemos agrupar las tecnologías de transporte de datos inalámbricas en tres grandes grupos:

LAN o redes de área local. Son redes de corto alcance, generalmente de unos pocos metros. Como ventaja destacan su estándares claros y bien definidos, además de permitir grandes transferencias de datos simultáneamente. En contraposición, este tipo de tecnologías requieren un suministro de energía elevado, así como unas dependencias y costes de red muy elevados. En este grupo encontramos WiFi o bluetooth.

Redes Móviles, donde ubicamos los protocolos GSM, 3G, 4G o el actual en desarrollo 5G. Destacan por su gran cobertura, al ser el método de transmisión tradicional, se cuenta con enormes infraestructuras alrededor del mundo para cubrir casi todas las áreas del planeta, además de la gran transferencia de datos que permiten aun siendo móviles. Sin embargo, esta tecnología falla de nuevo en los recursos energéticos que consume para mantener el flujo de datos, así como su coste para el propietario.

Por último, encontramos la tecnología LPWAN. Como LoRa, se garantiza un consumo mínimo, un bajo coste, y un gran alcance. En sacrificio, no obstante, de un bajo flujo de datos simultaneo, y con estándares que está aún en desarrollo debido a su relativa novedad. Este tipo de tecnología será muy adecuado por tanto para todas aquellas conexiones que no requieran un flujo de datos elevados, pero se quieran instalar en zonas de baja cobertura LAN y alimentados con baterías.

Es por tanto la tecnología perfecta para construir dispositivos que nos comuniquen su estado de manera rápida y eficiente, en cualquier ubicación.

Como cualquier tecnología de comunicación, es necesario un protocolo que fije las reglas para que la comunicación pueda establecerse entre dos puntos. Es aquí donde aparece LoRaWAN, como uno de los protocolos de comunicaciones que hace uso de esta tecnología.

5.1.1.1 Arquitectura de Red.

En la mayoría de los protocolos de comunicaciones se establece una topología de malla con el fin de incrementar el alcance. Así, la información se pasa no solo entre el nodo y el Gateway, sino entre nodos. A nivel de alcance esto supone una ventaja, no obstante, a nivel de consumo de batería, es un desperdicio, ya que se está obligando a nodos a transmitir datos que no son relevantes para ellos.

En el protocolo LoRaWAN por otro lado, los nodos no se asocian a ningún Gateway en particular, sino que sus datos son recibidos por varios de ellos al mismo tiempo. Una vez los paquetes de datos alcanzan el Gateway, los paquetes son mandados al servidor red que se desee, en nuestro caso el servidor europeo de TTN, mediante cualquier tipo red de retorno (WiFi, ethernet, satélite o móvil). La complejidad del protocolo se le delega al servidor, el cual será el encargado de realizar las comprobaciones de seguridad, eliminar los paquetes redundantes, escoger las tasas de envió adecuadas según la circunstancia, etc.

Una gran ventaja de esta topología es la creación de nodos móviles, si existe una cobertura total dentro del área de movimiento, no habrá que reiniciar la conexión cada vez que se pasa de un punto de acceso a otro, algo muy importante en las aplicaciones que requieran trazabilidad.

5.1.1.2 Vida de la batería

La principal ventaja de este protocolo es el bajo consumo energético que se requiere en los nodos para la transmisión de datos.

En los protocolos tradicionales con distribución en malla, se requiere despertar constantemente a los nodos con el fin de sincronizarse con la red y ver si hay mensajes nuevos.

Por otro lado, en LoRaWAN los nodos son asíncronos, y solo transmiten cuando tienen datos preparados para enviar.

Gracias a estas características convierten a LoRaWAN en uno de los protocoloc mas eficientes en el consumo de recursos, lo que lo hace especialmente interesante para su implementación en dispositivos que no tienen acceso fácil a la red de suministro eléctrica.

5.1.1.3 Capacidad de red.

Para que una red en estrella a larga distancia sea viable, y teniendo en cuenta que, dentro del protocolo, cada nodo envía su información a todos los Gateway dentro de su alcance, estos deben ser capaces de aceptar la conexión de un número muy elevado de nodos.

Para que esto sea posible, se hace uso de flujos de datos adaptativos y de múltiples canales a través de transceptores multi-modem. En estas conexiones, la capacidad de la red se verá afectada por la cantidad de canales disponibles, el flujo de datos, la longitud de los “payload”, y cuantas veces se transmite.

Una de las ventajas de usar LoRa, es la posibilidad de usar diferentes espectros de dispersión. Los Gateway aprovechan esta ventaja pudiendo recibir múltiples flujos de datos diferentes en el mismo canal y al mismo tiempo. Mediante los factores de dispersión adecuados, cada nodo adapta la velocidad del flujo de datos a la distancia al Gateway, por tanto, cuanto más cerca esté, el flujo será mayor y se reducirá el tiempo en el aire permitiendo nuevas conexiones entrantes.

El uso de estos métodos de comunicación no solo dota de una gran capacidad a las redes LoRaWAN, sino que las hacen altamente escalables. La red inicial puede ser desplegada con una infraestructura mínima, e ir incrementándose poco a poco mediante la adición de más puntos de acceso según vaya siendo necesario.

En nuestro caso, hemos iniciado la red con un solo Gateway. Gracias a lo citado anteriormente, nuestra pasarela será capaz de recibir datos de cientos de nodos. De este modo, con un solo Gateway, se puede dar servicio a varias comunidades de vecinos en un área de unos 2 km aproximadamente. Mientras cada comunidad haga un uso adecuado de la red, no debería ser necesario un nuevo punto de acceso, por lo que se plantea una solución realmente económica y sencilla.

5.1.1.4 Clases de Dispositivos.

No todos los nodos son iguales, cada dispositivo puede tener una función muy diferente, se pueden requerir así distintas capacidades según el uso final del dispositivo. LoRaWAN distingue tres clases de dispositivos. Clase A, B y C

Los dispositivos de Clase A son los más sencillos y a la vez eficientes, cualquier dispositivo que soporte LoRaWAN debe soportar esta clase. El propio nodo será el encargado de realizar la transmisión una vez tenga los datos listos para transmitir. La decisión de transmitir puede ser programada o mediante eventos. Esta es la clase usada en sensores, recogen los datos, los pasan al Gateway, y vuelven a un estado de reposo hasta que tienen que transmitir de nuevo.

Los dispositivos denominados clase B, tienen un consumo algo mayor que los primeros. Se diferencian de estos en que además de transmitir al Gateway de manera programada, realizan escuchas periódicas, pudiendo así comunicarse el Gateway con ellos en cada uno de esos momentos. Según la duración y frecuencia de las escuchas el consumo de batería será más elevado. Este tipo de dispositivos son adecuados como actuadores.

Por último, los dispositivos de Clase C, se mantienen una comunicación abierta con el Gateway constantemente. El consumo de estos dispositivos es muy alto, y por tanto solo se aconsejará su uso en condiciones de alimentación a través de la red. Se usará en caso de que se requieran actuadores de activación inmediata, o se busque obtener datos de sensores en momentos determinados.

En nuestro caso se ha optado por un dispositivo de Clase A. No se requiere actualmente ningún actuador remoto en el diseño, y no se considera necesario aumentar el consumo a cambio de poder recibir actualizaciones de estado no programadas.

5.1.1.5 Seguridad.

La ciberseguridad es actualmente la mayor debilidad en los dispositivos IoT, es muy difícil garantizar la robustez y fiabilidad de las comunicaciones ante posibles ataques cuando se despliegan puntos de acceso nuevos susceptibles a vulnerabilidades. Para minimizar los riesgos, LoRaWAN hace uso de dos capas de seguridad: una autentifica el nodo en la red, mientras que la segunda asegura la autentificación de la aplicación, impidiendo que el operador de red pueda tener acceso a los datos de la aplicación.

Esta seguridad se consigue a través de una encriptación AES, con intercambio de claves mediante un identificador IEEE EUI64.

En el presente documento se indica cómo obtener estas claves, tanto de nuestra red, como de nuestro nodo, como de nuestra aplicación final.

5.1.2 MQTT

MQTT es el protocolo por excelencia para IoT. Nacido en 1999 de la mano de Andy Standford y Arlen Nipper. El objetivo era un protocolo ligero y que consumiera muy pocos recursos, tanto en ancho de banda como en batería.

El protocolo se basa en una arquitectura de suscripción y publicación en el que el punto central de la comunicación es el "Bróker". Este elemento del protocolo es el encargado de transmitir los mensajes entre los receptores adecuados.

Cada cliente que publica un mensaje al broker y este la añade un topic. Cada cliente que quiere recibir ese mensaje se tiene que subscribir al topic, y el broker se lo recibe. No es necesario por tanto que se conozcan los clientes, con subscribirse al topic es suficiente para entablar la comunicación.

Ese protocolo evita que el cliente tenga que solicitar constantemente la información que necesita, se mantiene por tanto una comunicación TCP constante con el broker. En caso de que se interrumpiera en algún momento esta comunicación, este puede almacenarla en cola y transmitirla cuando el cliente se vuelva a conectar.

El elemento principal de los mensajes transmitidos en MQTT es el Topic, que no es más que una cadena que puede poseer niveles jerarquizados separados por barras en el ejemplo de nuestro Gateway, por tanto, un Topic para pasar la temperatura de la CPU sería Gateway/Cpu/Temperature.

En nuestro desarrollo, Node-red cuenta con un paquete desarrollado para la interacción con TTN. Este paquete controla las suscripciones y publicaciones MQTT entre Node-Red y TTN, esto facilita enormemente

5.2 Desarrollo del Gateway.

A continuación, se detalla como generar una pasarela abierta hacia TTN mediante el uso del microprocesador, una Raspberry Pi, y el concentrador LoRa IC880A, escogidos.

Los siguientes puntos desarrollan así la implementación y puesta en marcha de un Gateway totalmente funcional y operativo, basándose en las herramientas y repositorios disponibles en la plataforma GitHub, así como en Raspberry Foundation e IMST.

5.2.1 Preparar la tarjeta SD.

Para este desarrollo se ha decidido utilizar "Raspbian", una distribución de Linux basada en Debian Stretch como marco de trabajo para generar nuestra pasarela.

Raspbian contiene todos los controladores y software necesario para hacer funcionar correctamente nuestra Raspberry Pi, y solo necesita ser montado en una tarjeta SD e introducirlo en el slot destinado a ello del microcontrolador.

Raspbian es gratuito y puede descargarse de manera sencilla del sitio web oficial de Raspberry en el siguiente enlace https://www.raspberrypi.org/downloads/raspbian/. Una vez descomprimido lo montaremos sobre nuestra SD, en nuestro caso hemos utilizado el terminal de Linux con las herramientas que ofrece de forma nativa este sistema.

Comenzamos por localizar nuestra tarjeta mediante el comando lsblk que nos muestra todos los dispositivos que tenemos conectados.

```
casa:~$ sudo lsblk
```

En nuestro caso la hemos localizado, y se le ha asignado el nombre de /mmcblk0. Nuestra SD cuenta además con una partición /mmcblk0p1.

Como nuestra tarjeta es reciclada, y la partición ha sido usada, vamos a desmontar la partición para evitar problemas. Para esto usamos el comando umount.

```
casa:~$ sudo umount /dev/mmcblk0p1
```

A continuación, vamos a montar la imagen en la tarjeta SD, para ello hacemos uso de la herramienta "dd". Esta es increíblemente poderosa y es muy importante saber bien lo que se hace con ella, una dirección equivocada en el comando puede inutilizar el equipo de trabajo.

El comando nos solicita unos parámetros para montar la imagen, es necesario que le pasemos, el tamaño de cada bloque "bs=", cuanto más alto más rápida la instalación, pero puede generar fallos y problemas si el tamaño es demasiado elevado, nosotros lo hemos fijado en 4. La ruta al archivo que queremos montar como argumento para "if=". El nombre del dispositivo sobre el que queremos montar nuestra imagen como argumento para "of=". Ejecutamos entonces el siguiente comando:

```
casa:~/Descargas$ sudo dd bs=4M if=2018-11-13-raspbian-stretch-lite.img of=/dev/mmcblk0
conv=fsync
```

Con estas sencillas operaciones tenemos nuestro entorno, para comenzar a levantar la pasarela, ya montado en la SD. A continuación, introducimos la SD en la Raspberry, la conectamos a internet mediante cable de ethernet y nos conectamos mediante ssh a traves de nuestro equipo.

```
casa:~/Descargas$ ssh pi@192.168.1.37
```

5.2.2 Configurar la Raspberry

Nos solicitará la contraseña, por defecto esta es Raspberry, es importante por seguridad modificarla.

Para poder usar la Raspberry como solución en nuestro desarrollo es necesario realizar unos ajustes que evitarán futuros problemas en la implementación. Para ello usamos el comando "raspi-config" el cual nos abre una venta con ajustes del sistema. Una vez dentro realizamos dos acciones, primero debemos habilitar la comunicación SPI que permitirá interconectar la placa con el IC880a y por otro lado habilitamos la expansión del sistema de archivos, asegurando que toda la SD se utilice por el sistema. De esta misma forma ajustamos los locales, seleccionando

la opción “ES-ES UTF-8” y ajustamos la zona horaria a la correspondiente, en este caso seleccionamos la zona horaria de Madrid.

Para continuar creamos nuestro propio usuario y le damos una nueva contraseña, además de incluirlo al grupo sudo. Es importante primero crear el usuario y después añadirlo al grupo añadiendo sudo al comando de creación, si se hace directamente nos devolverá error.

```
raspberrypi:~ $ sudo adduser ttngate
raspberrypi:~ $ sudo adduser ttngate sudo
```

5.2.3 Instalar los controladores del Gateway

5.2.3.1 GitHUB

A lo largo de este proyecto se usará GitHub con dos objetivos, el primero se enfoca dentro del punto presente con el fin de clonar un repositorio con el firmware necesario para hacer funcionar nuestra pasarela. La otra será como repositorio remoto para control de versiones de nuestro desarrollo.

GitHUB es una plataforma de desarrollo colaborativo de software que aloja proyecto mediante el sistema de control de versiones GIT. Esta solución es una de las principales fuentes de código libre para iniciar cualquier proyecto.

A través de GitHub podremos tanto aprovechar proyectos y soluciones ya creadas por otras personas, como contribuir aportando nuevos desarrollos.

En este caso se ha hecho uso del proyecto creado por Gonzalo Casas, en el que se encuentran las librerías, las capas de abstracción HAL, y los archivos de configuración necesarios para levantar una pasarela de manera rápida y sencilla.

Para conseguir este propósito solo necesitamos clonar el repositorio en la tarjeta SD de nuestro dispositivo. Ejecutamos a continuación los siguientes comandos.

Primero aseguramos contar con la versión más estable de Git:

```
raspberrypi:~ $ sudo apt-get install git
```

A continuación, clonamos el repositorio a la tarjeta:

```
raspberrypi:~ $ git clone -b spi https://github.com/ttn-zh/ic880a-gateway.g
```

A lo largo de este proyecto, se hará uso de GitHub también como repositorio remoto. Se ha decidido usar este mismo debido a la gran colaboración que brinda esta plataforma al entorno MAKER. Se pretende además que este sea punto de partida para nuevos desarrollos de nodos para comunidades de vecinos.

5.2.3.2 Instalación y configuración

Una vez clonado, el repositorio cuenta con un instalador que se encargará de generar todas las dependencias necesarias para que nuestro hardware funcione. Solo es necesario ejecutar el script de Shell –install.

```
raspberrypi:~/ic880a-gateway $ sudo ./install.sh spi
```

En este momento se nos da un elemento clave en la configuración del Gateway, es la clave EUI mencionada en el protocolo LoRaWAN punto 5.1.1.4 :

```
The Things Network Gateway installer

Version spi

Updating installer files...

Already up-to-date.

Gateway configuration:

Detected EUI **********7EEAC from eth0
```

servidor europeo de TTN

```
gateway_conf": {

"gateway_ID": "**********7EEAC",

"servers": [

{

"server_address": "router.eu.thethings.network",

"serv_port_up": 1700,

"serv_port_down": 1700,

"serv_enabled": true

}

],

"ref_latitude": 39.51919554,

"ref_longitude": -0.42374309,
```

```
"ref_altitude": 59,
"contact_email": "email de contacto",
"description": "casas_verdes_ttn"
}
```

Con esto quedaría el software para el control de la pasarela implementado y listo para funcionar.

5.2.4 Librería -libloragw

Semtech proporciona la capa de abstracción HAL a través de una biblioteca C completa, con los controladores necesarios para utilizar su chip mediante las funciones de alto nivel mínimas que configuran el hardware, así como permiten enviar y recibir paquetes.

La biblioteca se compone de un total de 6 módulos, loragw_hal, loragw_reg, lora_spi, loragw_aux, loragw_gps y loragw_radio. Junto a estos se incluyen además programas de prueba básicos para demostrar el uso del código y verificar su correcto funcionamiento.

Del módulo loragw_gps no hablaremos, ya que no se ha hecho uso de él. El concentrador IC880A en su versión estándar no contiene modulo GPS y por tanto no será objeto de este proyecto.

5.2.4.1 Loragw_hal

Aquí encontramos la librería principal, contiene todas las funciones de alto nivel que configuran y hacen uso del concentrador LoRa. Las funciones más destacadas de esta librería son:

*lgw_board_setconf, establece la configuración del concentrador

* lgw_rxrf_setconf, configura los canales de radio

* lgw_rxif_setconf, para configurar los canales IF

* lgw_txgain_setconf, permite ajustar la tabla de ganancias.

* lgw_start, aplica la configuración establecida e inicia la placa.

* lgw_stop, detiene las operaciones

* lgw_receive, permite reocger los paquetes que hayan sido enviados

* lgw_send, envia un solo paquete

* lgw_status, permite confirmar que un paquete ha sido enviado correctamente

En un primero momento es suficiente con esta librería para hacer funcionar el concentrador, no obstante, para un Gateway funcional, se requiere el uso de alguna más.

5.2.4.2 Loragw_reg

Mediante esta librería podemos acceder a los archivos de paginación, a los registros de solo lectura, modificar y escribir rutinas para registros sub-byte. Así como todas las funciones que nos permitirán hacer el código mucho más legible y fácil de depurar.

* lgw_connect, inicializa y testea la conexión con el hardware

* lgw_disconnect, desconecta el hardware

* lgw_soft_reset, reinicia el hardware reinicializando el registro completo.

* lgw_reg_check, permite cruzar los registros contra los valores por defecto y lo muestra.

* lgw_reg_r, lee un registro

* lgw_reg_w, permite escribir un registro

5.2.4.3 Loragw_spi

Este módulo es necesario para poder gestionar la conexión entre el concentrador y la Raspberry Pi. Esta librería contiene las funciones que gestionan la comunicación SPI entra ambas placas, son funciones sencillas de escritura y lectura de bits.

* lgw_spi_r para leer un byte

* lgw_spi_w escribe un byte

* lgw_spi_rb para leer 2 o más bytes

* lgw_spi_wb permite escribir 2 o más bytes.

5.2.4.4 Loragw_aux

Esta librería contiene una sola función “wait_ms”, la cual además es dependiente del anfitrión. Esta misma permite detener las operaciones durante un tiempo definido.

Los protocolos de operaciones en las comunicaciones a través del concentrador requieren de ciertas esperas a determinados momentos. Estas esperas suelen permitir el ajuste de relojes o voltajes tras un cambio.

En caso de que los tiempos de espera mínimos no se garanticen durante los procedimientos de arranque y paro, el hardware probablemente no funcione adecuadamente y podría terminar dañándose.

5.2.5 Uso de las librerias

El uso de las librerías queda perfectamente explicado junto a loas librerías facilitadas por Semtech. A continuación, se describe el uso básico de las mismas para el control del concentrador.

Lo primero es incluir el archivo de cabecera loragw_hal.h dentro del programa y linkarlo estáticamente a libloragw.a durante la compilación. Incluiremos también el archivo loragw_reg.h para poder acceder al registro.

Con esto podremos hacer uso del HAL de semtech, siempre y cuando sigamos las reglas que se mencionan a continuación.

* Siempre se configura la ruta de las radios y el IF+modem antes de habilitar la radio.

* la configuración solo pasará al concentrador cuando se llame a la función “start”

* No se recibirán ni enviarán paquetes hasta que se cumpla: la radio está habilitada, uno o más IF-modems está activo y el concentrado se ha iniciado.

*Siempre se debe detener el concentrado antes de modificar la configuración.

Un ejemplo básico de uso del concentrador se describe a continuación:

```
<configurar las radios y el IF+modems>
<iniciar el concentrador>
loop {
        <recuperar los paquetes recibidos por el concentrador>
        <procesar, almacenar o hacer uso de “pkt_forwarder” con los paquetes>
        <enviar los paquetes>
}
<detener el concentrador>
```

sa.
Se debe considerar así mismo el tiempo que tomará en enviarse un paquete antes de enviar uno nuevo, en caso contrario se bloqueará el envío generando un error.

5.2.6 Interconexión mediante Protocolo SPI

Para la interconexión entre el microprocesador y la Raspberry Pi se hace uso del protocolo SPI, o bus de interfaz de periféricos en castellano.

Este protocolo fue desarrollado por Motorola en 1980 para comunicar diferentes partes de sus sistemas electrónicos. Es un bus síncrono de 4 hilos que emplea un reloj para sincronizar las transmisiones de datos. La comunicación es bidireccional, y serial, se transmite un bit en cada ciclo.

Para establecer comunicación, el maestro debe seleccionar primero el esclavo fijando a valor bajo el selector de chip que corresponde al esclavo. A continuación, se activa el reloj (SCLK) generando una señal simétrica, y se usará cada ciclo del mismo para que el maestro escriba un bit en la linea MISO, y lea de la línea MOSI en el flanco de reloj inverso al de escritura. Simultáneamente a este proceso, el esclavo escribirá un bit en la MISO y leerá de la línea MOSI en cada flanco de bajada.

Cuando la trama de datos ha sido completamente transmitida entre ambos circuitos, el maestro deberá desactivar la señal de reloj y volverá a poner el bit de la línea de selección a nivel alto.

En la figura 17 se muestra un ejemplo de este tipo de comunicación

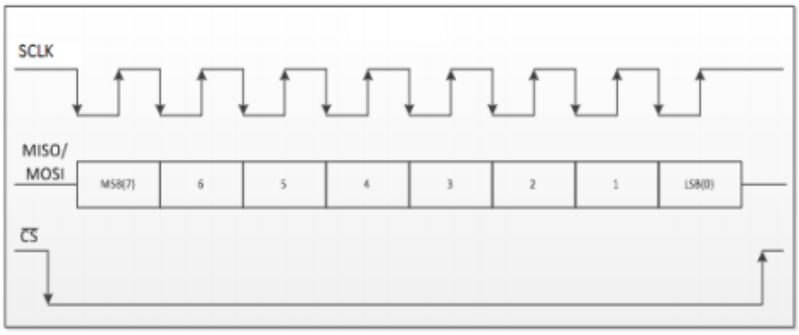


FIGURA 16 EJEMPLO DE TRANSMISIÓN DE DATOS MEDIANTE SPI

Para realizar esta conexión entre la Raspberry y la placa de desarrollo IC880A, se debe conocer el “Pinout” de cada tarjeta, para ello es necesario recurrir a los datasheet del fabricante. En la tabla 2 se puede consultar el pin correspondiente a cada señal como lo hemos conectado nosotros.

iC880a pin	Description	RPi physical pin
21	Supply 5V	2
22	GND	6
13	Reset	22
14	SPI CLK	23
15	MISO	21
16	MOSI	19
17	NSS	24

TABLA 2 CONEXIÓN DE PINES RASPBERRY PI E IC880A. VER DATASHEET EN ANEXO

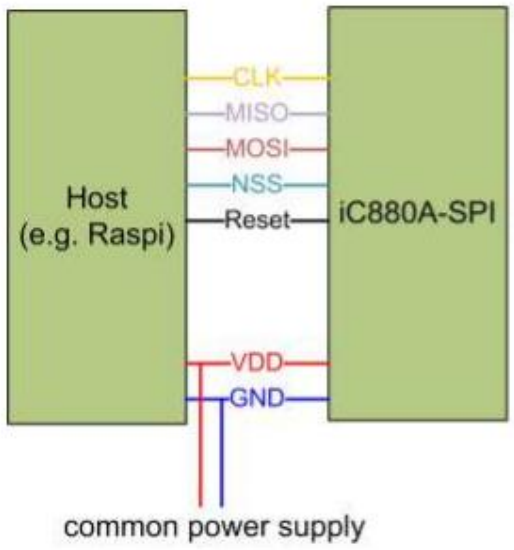


FIGURA 17 DIAGRAMA DE BLOQUES CONEXIONADO IC880A RECUPAREDO DE HTTP://WWW.IMST.COM/

En la Figura 17 se observa un diagrama de bloques sencillo de la conexión final entre la Raspberry y el concentrado IC880A. Como se observa la interconexión es muy sencilla y son solo necesarios 7 hilos para hacer funcionar el concentrador LoRa. A continuación, detallamos el uso de cada uno de ellos.

CLK: Este hilo controla le señal de reloj por la que se rige la transferencia de datos como se ha indicado anteriormente. Será iniciada por la Raspberry Pi.

MOSI: Línea empleada para que la Raspberry Pi transmita sus datos al concentrador. Durante los periodos de no comunicación, el pin 19 de la Raspberry se situará en modo de alta impedancia.

MISO: Esta línea será la encargada de transmitir los datos del IC880A hacia la Raspberry. En nuestro diseño solo hemos utilizado un esclavo, en caso de requerirse más, se conectarán a esta misma línea, y por tanto se deberá poner el pin 19 de la tarjeta IC880A en modo alta impedancia para permitir nuevas conexiones.

NSS: Es la línea de selección de esclavo. Cuando la Raspberry quiera comenzar a transmitir hacia la IC880a deberá fijar este pin a nivel bajo para que pueda comenzar la transferencia de datos.

RESET: Esta conexión queda fuera del protocolo SPI. Como recomendación del fabricante, este pin debe encontrarse a nivel alto durante 100ns cada vez que se vaya a iniciar el sistema, con el fin de asegurar un arranque estable. Se conectará a un pin GPIO de la Raspberry que enviará una señal de 3,3V durante 100ns cada vez que se requiera el inicio del sistema. Este pin permite el reinicio manual en caso de necesidad a través de consola, solo es necesario poner el pin de la Raspberry a nivel alto.

VDD/GND: Alimentará tanto a la Raspberry como al concentrador. Se detalla esta alimentación en el punto siguiente.

Todo el cableado se ha hecho utilizando cables tipo jumper de 10 cm de largo. El problema del protocolo SPI, es que debido a las frecuencias en las que trabaja, así como los rangos de corriente, es un protocolo de proximidad. En inicio está orientado a conexión soldadas dentro de la misma placa. Se ha decidido usar este tipo de conexión como prueba de concepto, el desarrollo final del diseño deberá contar con una backhaul de unión mediante PCB entre ambas placas para evitar problemas en la transmisión de paquetes.

5.2.4 Alimentación y conexión a la red mediante PoE.

Los requisitos de consumo de nuestro Gateway vienen marcados por el consumo combinado de la Raspberry y el concentrador LoRa. La tensión necesaria será siempre 5V.

En el datasheet del fabricante se indica como alimentación mínima para el IC880a una corriente de 700mA.

Por otro lado, la Raspberry Pi cuenta con un consumo medio de 1200 mA sin periféricos, es decir, mantener la CPU, la RAM, y los procesos básicos, requieren mantener la citada corriente, en caso de ser insuficiente, la Raspberry entra en modo baja energía limitando sus procesos y

bajando la velocidad de la CPU. En nuestro caso, además, vamos a incluir un pequeño ventilador para controlar la temperatura de la CPU con un consumo medio de 150mA.

$$I_{min} = I_{pi} + I_{vent} + I_{IC88A} = 2050\,mA.$$

Teniendo en cuenta momentos de arranque, procesos pesados, o la necesidad futura de algún pequeño periférico extra, hemos decidido usar un sistema de alimentación que garantice una corriente de 2400 mA a 5V.

Con el fin de garantizar una comunicación fiable y de alta velocidad, se ha descartado el WiFi, y aunque conexión GSM es un posible planteamiento a futuro, se ha decidido descarta por la diferencia de coste con respecto a un cable ethernet desde un router.

Así, la alimentación y la conexión a la Red se realizarán mediante un único cable de ethernet, mediante la denominada conexión PoE, (Power over Ethernet).

La alimentación a través de Ethernet se define en los estándares IEE802.3af a través del cual se garantiza una alimentación mínima de 12.95W por puerto. De nuevo, mediante matemáticas sencilla, podemos ver que esta alimentación es más que suficiente para nuestro desarrollo:

$$I = \frac{P}{V}\,;\quad I = \frac{12.95}{5} = 2.59A$$

La mayor ventaja de elegir esta alimentación es que reducimos la infraestructura necesaria para ubicar nuestro Gateway, será por tanto solo necesario acometer a la envolvente, un único cable Ethernet.

Para utilizar este tipo de alimentación se ha decidido hacer uso de un dispositivo comercial a la que se acometerá con el cable de ethernet, y nos dará una nueva conexión tipo RJ45 para conectar al puerto LAN de nuestra Raspberry, un conector micro-USB para alimentar nuestra solución. Para mayor referencia del dispositivo utilizado ver Figura 18

FIGURA 18 DSLRKIT GIGABIT ETHERNET ACTIVE POE SPLITTER 5V MICRO USB

ADVERTENCIA Es muy importante que nunca se alimente un concentrador si la antena no está conectada. Al no poder disipar la energía, el exceso que se genere quemaría la placa.

En nuestro caso hemos usado un adaptador u.ff a SMA y una antena con ganancia de 6 dB.

5.3 TTN

The Things Network es una plataforma Open Source fundada por Wienke Giezman y Johan Stokking. Nace en Ámsterdam en 2015, donde cubrieron todo el centro de la ciudad con cobertura LoRaWAN en unas pocas semanas.

El mayor aporte de TTN es el desarrollo en el lado del servidor, en el cual se ha construido el backend de la red para dar soporte a todos los Gateways conectados a través de ella. Mediante él se tratan los mensajes duplicados, se gestionan las bajadas de mensajes, la integración con plataformas, etc.

Otra de las ventajas que ofrece de TTN es la integración por HTTP y MQTT, así como APIs en GO, Java, Node-RED o Node.js, lo que facilita la creación de soluciones completas desde el nodo sensor hasta la aplicación final de usuario.

En definitiva, TTN es una red global abierta que tiene como objetivo principal facilitar la creación y desarrollo de soluciones IoT completas mediante tecnología LoRaWAN. Su objetivo principal es crear una red de cobertura mundial mediante la implantación de Gateways por parte de usuarios particulares.

TTN nos ofrece una consola de control para gestionar tanto nuestros Gateways, como nuestras aplicaciones de manera sencilla e intuitiva, una vez nos registremos en la plataforma.

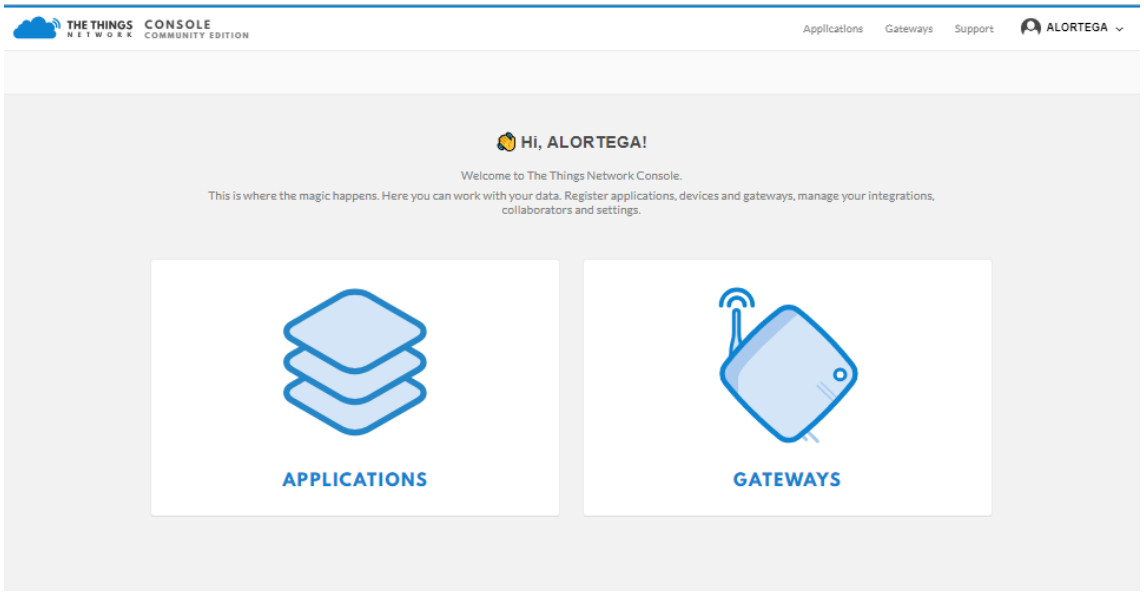


FIGURA 19 CONSOLA DE CONTROL DE TTN

5.3.1 Gateway TTN

Los Gateways son el puente entre los dispositivos y el servidor The Things Network. Este enlace se hace mediante la ya mencionada tecnología LoRaWAN entre dispositivo y Gateway, para posteriormente pasar la información, mediante una tecnología con mayor ancho de Banda como WiFi, ethernet o 3G a TTN.

El funcionamiento de los gateways se basa en la incorporación de un concentrado LoRa, un chip especial desarrollado por Semtech, el cual permite la demodulación de las transmisiones en frecuencia.

Los Gateways pueden ser generalmente de dos clases. Los que se desarrollan bajo una capa mínima de Firmware, de bajo coste y fácil desarrollo, y los que corren sobre un sistema operativo completo, este es el caso de nuestro Gateway.

Nuestro dispositivo se ha construido sobre el sistema operativo Raspbian, lo que aparte de usar el software de reenvio de paquetes, dota al administrador de un control y personalización mucho mayor del Gateway. Esto dota a nuestro desarrollo de gran versatilidad, la cual pretendemos hacer mayor uso en un futuro.

5.3.1.1 Registrar nuevo Gateway

El Gateway desarrollado hace uso del Forwarder de paquetes mediante UDP facilitado por Semtech, aunque actualmente se aconseja el uso de reenviador de paquetes de TTN, el de Semtech está aún soportado y es más sencillo como configuración inicial.

Para comenzar le configuración lo primero que se necesita es la EUI mencionada en puntos anteriores, y se obtiene durante la instalación del Gateway. Una vez la tenemos, accedemos a la parte de Gateway dentro de la consola de TTN y seleccionamos la opción de crear un nuevo Gateway.

Se abrirá ahora una ventana de registro como la que se muestra en la figura 17. Lo primero es seleccionar el uso del reenviador de paquetes heredado. Una vez hecho nos solicitará la EUI del Gateway, la introducimos junto a una descripción del dispositivo, la frecuencia, y el enrutador a usar. La frecuencia será 868MHz y el servidor el europeo de TTN para nuestro desarrollo.

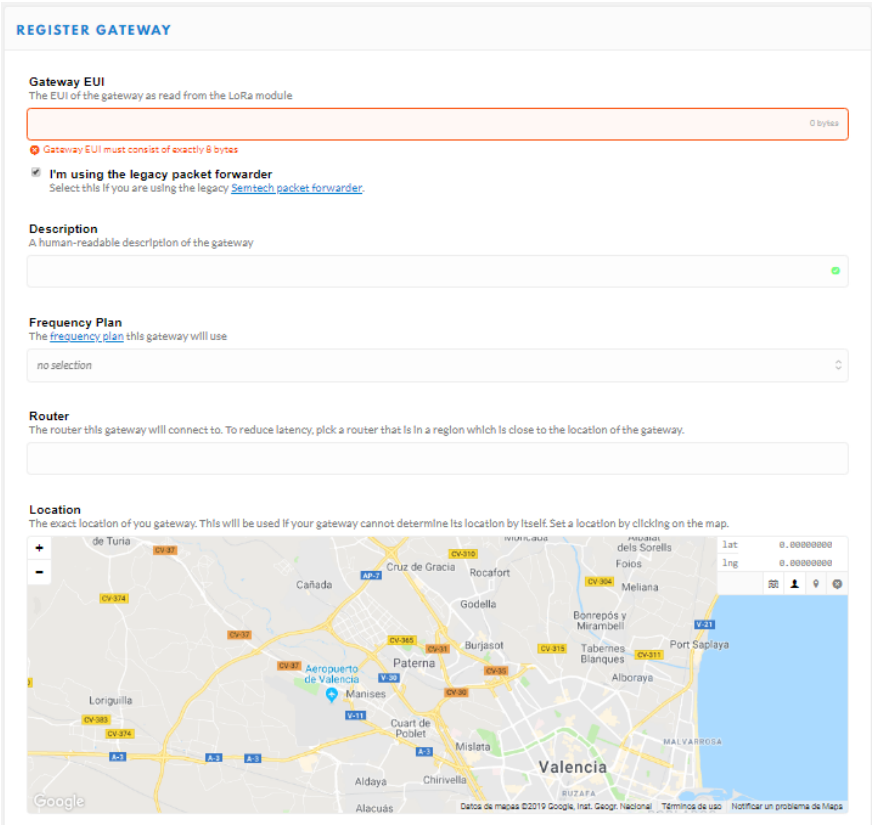


FIGURA 20 VENTANA DE REGISTRO PARA GATEWAYS TTN

Una vez terminada la configuración podremos ver nuestro Gateway registrado en la plataforma, y su estado. Si lo seleccionamos podemos entrar en una pantalla de visualización en la que se nos muestra una visión general de nuestro dispositivo. Se pueden ver ambas pantallas en la figura 22

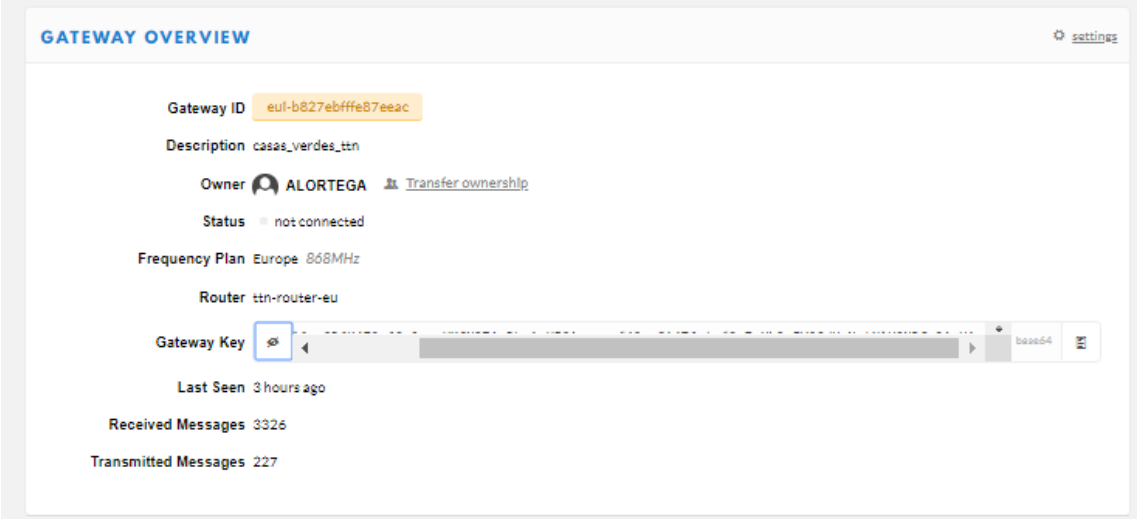


FIGURA 21 VISIÓN DEL GATEWAY EN EL PANEL DE CONTROL Y PANTALLA DE INFORMACIÓN DE ESTADO.

5.3.1.2 Semtech UDP Packet Forwarder

Como se ha mencionado anteriormente, el gestor de paquetes del que se ha hecho uso en este desarrollo heredado de Semtech. La función del gestor, por su definición en inglés "Paket Forwader" es adaptar los paquetes recibidos mediante LoRa, al protocolo requerido para su transmisión al servidor. Véase figura 22 para mayor referencia sobre su arquitectura.

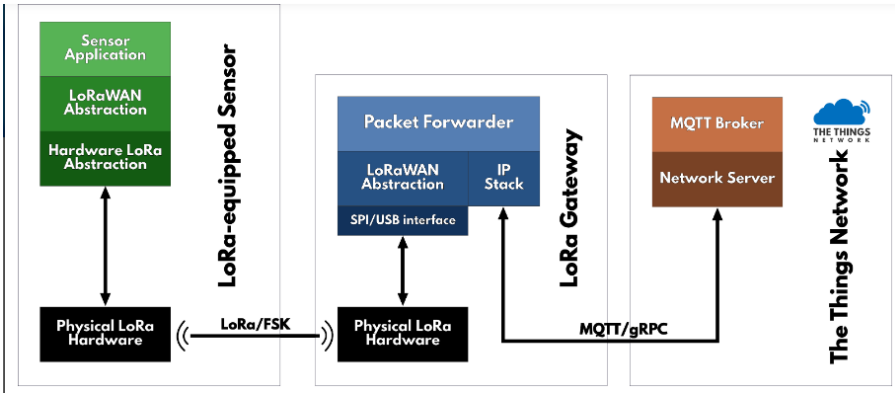


FIGURA 22 DIAGRAMA DE FUNCIONAMIENTO ENVIÓ DE DATOS MEDIANTE LORAWAN AL SERVIDOR REMOTO DEL NODO SENSOR.

El packet forwarder se ha instalado en nuestro Gateway como se ha visto en pasos anteriores, para su configuración solo es necesario ajustar los parámetros del servidor mediante el JSON de configuración global. En este caso solo es necesario indicar la dirección del servidor, en nuestro caso la europea, *router.eu.thethings.network*, y poner los puertos por defecto en 1700.

```
{
  [...]
          "gateway_conf": {
                  "server_address":
"router.eu.thethings.network",
                  "serv_port_up": 1700,
                  "serv_port_down": 1700,
    [...]
  }
}
```

5.3.2 Aplicación TTN

Con esto hacemos referencia a cualquier comunicación que realicen nuestros dispositivos con internet, desde una sencilla acción de encendido mediante, hasta un complejo panel web mediante Node-RED. Para poder habilitar esta comunicación es necesario registrar una aplicación en la consola de TTN y asociar dispositivos a ella.

La creación de aplicaciones es una tarea sencilla y rápida. Volvemos al panel de control de TTN y accedemos esta vez a la sección de aplicaciones, pulsamos en crear nueva aplicación y veremos una pantalla como la que se muestra en la figura 24.

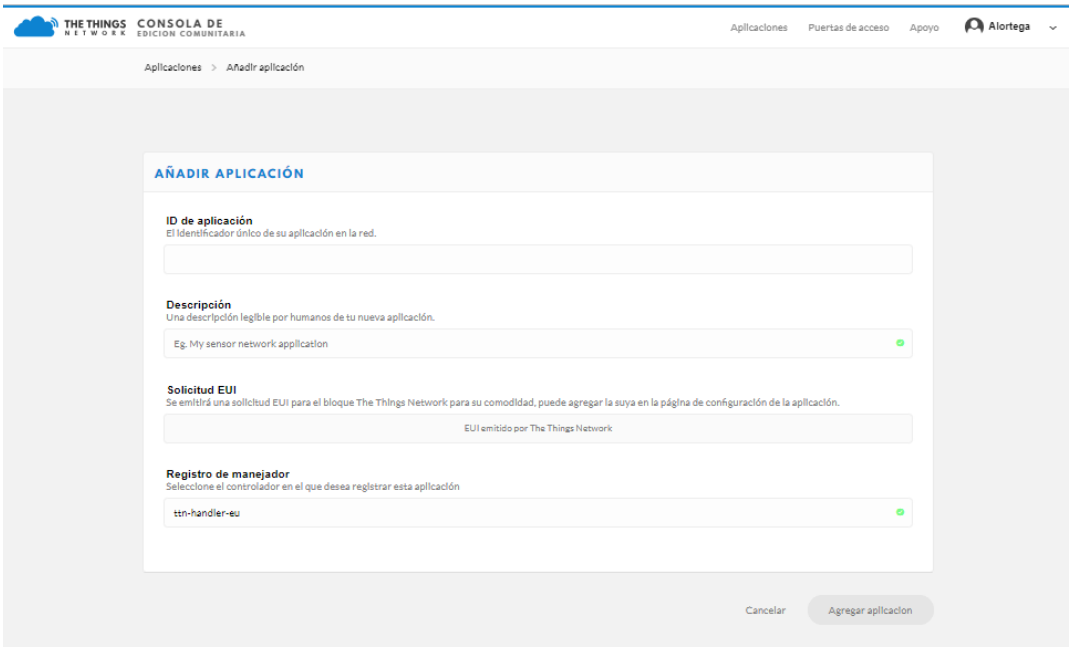


FIGURA 23 PANEL DE CREACIÓN DE APLICACIONES EN TTN

Para la ID introducimos un nombre único mediante minúsculas y caracteres alfanuméricos. Añadimos además una descripción que indique de manera clara y concisa el uso de esta.

Finalmente marcamos la casilla de registrar le aplicación por defecto en nuestra región y pulsamos finalizar.

Una vez generada la aplicación tendremos acceso a la pantalla que se observa en la figura 25. En ella se nos muestra la EUI de la aplicación, su ID, así como la clave de acceso que será importante en el desarrollo de cualquier comunicación como se verá más adelante.

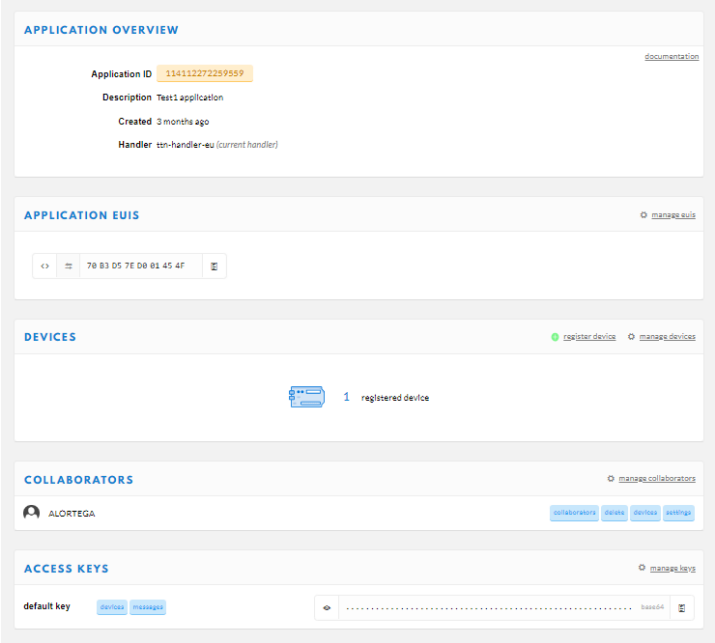


FIGURA 24 PANEL DE GESTIÓN DE APLICACIONES GENERADAS EN TTN

En esta pantalla podemos gestionar también los dispositivos registrados dentro de nuestra aplicación, así como añadir más. Para ello solo pulsamos en añadir nuevo dispositivo, y se abrirá una venta como la mostrada en la figura 26.

En esta nueva ventana añadimos una ID para nuestro dispositivo e introducimos la dirección EUI, más adelante se muestra cómo obtener esta dirección en el microcontrolador de Pycom.

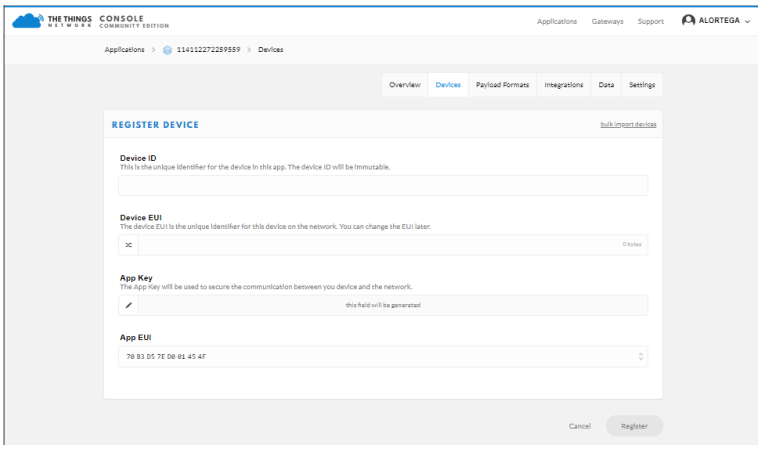


FIGURA 25 PANTALLA DE REGISTRO DE DISPOSITIVOS EN TTN

5.3.3 TTN MAPPER

Este desarrollo, aunque independiente de The Things Network, está perfectamente integrado con esta plataforma, siendo una herramienta fundamental para realizar mapas de cobertura. Esta iniciativa se basa en la colaboración de los desarrolladores para medir el rendimiento de los Gateway a los que se conectan sus dispositivos. La herramienta hay que agradecérsela a JPMeijers.

Existen dos formas de realizar el mapeo, mediante un nodo con GPS integrado o mediante un smartphone y cualquier nodo. Al no requerirse GPS en nuestro desarrollo, y no incluirse este de manera nativa en nuestro dispositivo, hemos descartado incrementar el coste de nuestro proyecto incluyéndolo, y por tanto se hará uso del segundo método.

Contribuir al mapa de cobertura mediante nodo y smartphone, se realiza en dos pasos.

Primero configuramos el nodo y lo damos de alta en TTN. Hemos hecho uso del ejemplo de LoRa que viene con la librería que facilita Pycom por defecto. Más adelante se muestra como se ha llevado a cabo un desarrollo más complejo a partir de esta librería, y como se realiza todo el proceso para incluir librerías a nuestro dispositivo. De momento es suficiente con enviar un mensaje a TTN. El registro de la aplicación y el nodo se realiza según los pasos descritos anteriormente.

Para poder realizar la conexión correctamente debemos dar de alta la integración en nuestra aplicación, para ello solo es necesario entran en la pestaña de integraciones dentro de la aplicación y seleccionar TTN Mapper, damos nombre e ID a nuestra integración y quedaría configurado para su uso.

Ahora instalamos el cliente móvil desde la tienda oficial de aplicaciones de Android. Esta aplicación solo transmite la ubicación mediante el GPS de nuestro móvil a la aplicación en TTN, aquí se combina con el paquete enviado por nuestro nodo. TTN Mapper recibe la información combinada de posición, tiempo de envió y calidad del paquete. Con estos datos va realizando un mapa de la cobertura en función de cada Gateway al que se conecte nuestro Nodo.

5.4 Nodo LoRa

En este punto entramos en la segunda parte de nuestro desarrollo. Hasta ahora se ha explicado el funcionamiento del protocolo LoRaWAN, se ha indicado el desarrollo de un Gateway mediante un microprocesador y una tarjeta con un concentrador LoRa, y se ha dado todo de alta en la plataforma open source The Thing Network.

Como se mencionaba al inicio del documento, el principal objetivo de este proyecto es mostrar las capacidades y la versatilidad de esta tecnología, por ello se ha decidido hacer un uso práctico real de la misma. Consideramos esto la mejor forma de mostrar las bondades de estos desarrollos.

Para mostrar las ventajas de este protocolo, así como de la plataforma TTN, no solo es necesario montar el Gateway, es necesario nodos que hagan uso de esta tecnología. En nuestro caso, como

ya se ha mencionado, el nodo sensor desarrollado monitorizará el estado de un parque infantil en una comunidad de vecinos. Esto abre la puerta a un desarrollo mucho mayor que permita la creación de un nuevo concepto de comunidad de vecinos, una comunidad 4.0.

Nuestro Nodo se basa en un microcontrolador adaptado para trabajar con LoRa. Al tratarse de una prueba de concepto y no de un desarrollo final, se ha hecho uso de una tarjeta de expansión que ya contiene sensores de humedad y temperatura facilitando el desarrollo inicial.

5.4.1 Desarrollo Electrónico

El desarrollo actual a nivel electrónico se realiza mediante cables aprovechando sensores con comunicación I2C o serial ya montados en placas, con el resto de los elementos electrónicos necesarios. No son objeto de este trabajo tareas como la adaptación de señales, el desarrollo de sensores o implementaciones más complejas que alejen del objetivo principal de este proyecto.

Todo nuestro desarrollo se basa en el microcontrolador LoPy 4. Como ya se ha mencionado anteriormente, es un dispositivo construido en 4 tipos de conexión diferentes WiFi, Sigfox, Bluetooth y LoRa. En la figura 23 se puede observar el diagrama de bloques por el que se rige este dispositivo. Todo se centra en el chip ESP32 por el que se rigen todas las comunicaciones.

El dispositivo usa SPI, más un pin GPIO para la comunicación con el chip de Semtech y su control. Esto habilita el uso de LoRa. Por otro lado, usa SPI para la comunicación con la Flash y la RAM. Quedan libres conexiones UART, GPIO, ADC, DAC, SPI, I2C PWM para el desarrollo de nuestras aplicaciones.

La alimentación del dispositivo es a 3,3V y para ello cuenta con un regulador lineal incorporado que permite rango de entre 3,5V y 5,5V en la entrada.

Cualquier pin es capaz de servir 3,3V, lo que implica que para algunos sensores puede ser necesario un incrementador de tensión para funcionar correctamente.

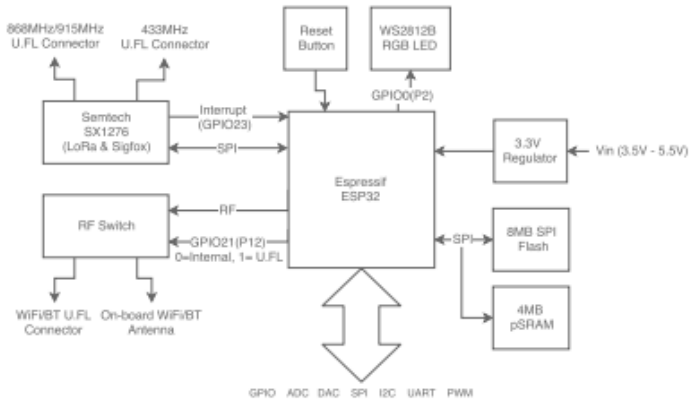


FIGURA 26 DIAGRAMA DE BLOQUES TARJETA LOPY4 FACILITADO POR PYCOM.

La comunicación serie mediante USB no está disponible en esta placa, para ello es necesario montarla sobre une tarjeta de expansión. Como alternativa se puede utilizar el protocolo FTP para programar la placa mediante WiFi, pero solo se recomienda su uso para usuarios avanzados, ya que es fácil bloquear este modo de comunicación al subir nuestros desarrollos, cerrando la comunicación con la tarjeta.

5.4.1.1 Tarjeta PYSENSE

Como ya se ha mencionado, la solución de Pycom para comunicación LoRa, no se distribuye con comunicación serie mediante USB por defecto. Por tanto, si se requiere conectarlo a un equipo para el desarrollo y carga de archivos en la memoria de la placa, es necesario una tarjeta de expansión facilitada por la misma compañía.

La tarjeta Pysense es la versión con sensores de la "Expansion Board" básica, y puede ser usada para aplicaciones ambientales de manera directa.

Entre el diverso hardware con el que cuenta la placa vamos a hacer uso del USB serie y el sensor de temperatura y humedad SI7006-A20.

Todos los sensores se comunican mediante interfaz I2C con la placa LoPy4 a través de los pines GPI09(SDA) y GPIO8(SCL).

La placa es por tanto de uso directo y la conexión se hace pinchando la Lopy4 en las conexiones hembra disponibles en la tarjeta Pysense con el Led de la LoPy en la parte del puerto serie USB. El único problema de este shield es que ocupamos los pines de la placa al montarlos sobre la tarjeta de expansión. Para solventar esto, la tarjeta cuenta con una cabecera de entradas y salidas, a la que hemos tenido que soldar pines macho para poder hacer la interconexión de los sensores externos.

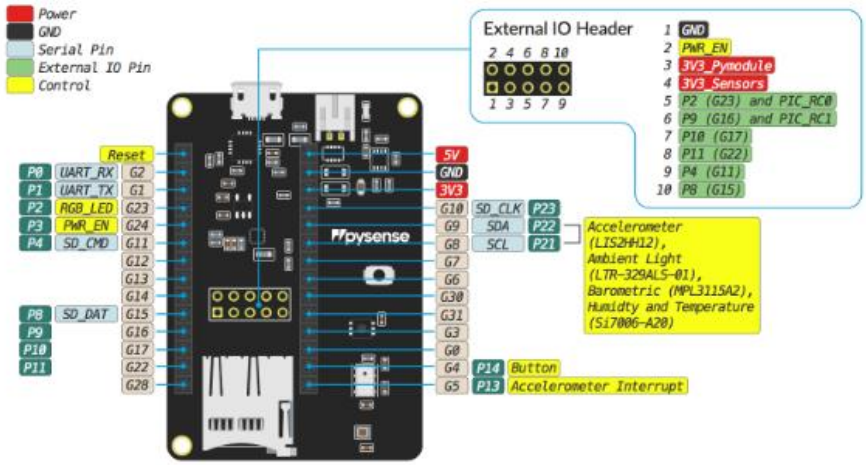


FIGURA 27 PINOUT Y REPRESENTACIÓN DE LA PLACA PYSENSE

5.4.1.2 Sensor SI7006-A20

El sensor SI7006-A20 es un circuito integrado creado por Silicon Labs, en el que se integran sensores de humedad, temperatura, un conversor ADC, calibración de datos, y una interfaz I2C.

El uso de este sensor dentro de la tarjeta Pysense, permite alcanzar los modos de bajo consumo que indica el fabricante. Su consumo en standby no supera los 60 nA, mientras que en tiempos de medida no alcanza los 200 µA. Todo esto con un rango de alimentación variable entre 1.9 a 3.6V.

El sensor es robusto y permite su uso en una amplitud de temperatura de -40 a +125°C y una humedad relativa de hasta el 100%. Como el sensor se encuentra montado junto a componentes que no pueden operar en esas condiciones, no es realmente aprovechable en nuestro desarrollo. No obstante, nos ofrece una medida más precisa en los valores reales de uso, ya que la incertidumbre es siempre mayor en los límites de medida, con este sensor no tendremos problema para monitorizar temperaturas hasta los 45°C que podamos llegar a obtener en pleno verano bajo el sol. La precisión se sitúa en torno a ±1°C y un ±5% de humedad relativa, más que suficiente para lo que se requiere en nuestra aplicación.

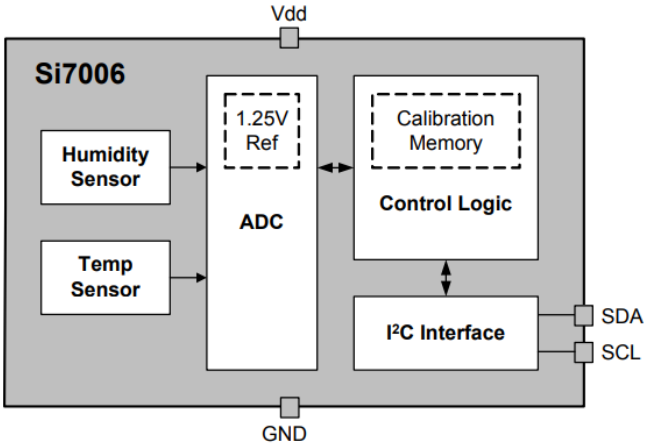


FIGURA 28 DIAGRAMA DE BLOQUES DEL SENSOR SI7006

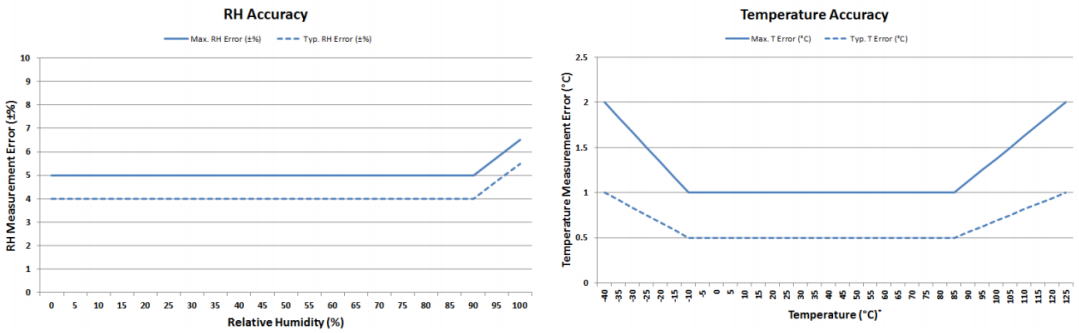


FIGURA 29 GRAFICAS DE PRECISIÓN EN TEMPERATURA Y HUMEDAD

5.4.2 Programación.

El desarrollo de este proyecto pretende seguir la cultura Maker, y para ello se ha decidido utilizar un dispositivo orientado a esta filosofía como se ha defendido en el análisis de alternativas. LoPy4 cuanta con una comunidad en crecimiento, y una documentación que hace bastante sencilla la implementación de desarrollo que encajen dentro de la cultura Maker.

La programación de esta placa se realiza en microPython, una versión de Python para microcontroladores, que difiere muy poco de la original y no va a ser ningún problema en nuestro desarrollo. Se ha decidido este lenguaje por su sencillez y simplicidad frente a lenguajes como C. Esto va a permitir que gente sin mucha experiencia en programación pueda colaborar fácilmente a la implementación de nuevas soluciones y nodos dentro de la comunidad. La filosofía de este lenguaje es que la sintaxis permita crear un código legible y fácilmente interpretable. Esto se recojo muy bien en los siguientes puntos que se denominan el ZEN de Python:

1. Lindo es mejor que feo.
2. Explícito es mejor que implícito.
3. Simple es mejor que complejo.
4. Complejo es mejor que complicado.
5. Plano es mejor que anidado.

6. Espaciado es mejor que denso.
7. La legibilidad es importante.
8. Los casos especiales no son lo suficientemente especiales como para romper las reglas.
9. Sin embargo, la practicidad le gana a la pureza.
10. Los errores nunca deberían pasar silenciosamente.
11. A menos que se silencien explícitamente.
12. Frente a la ambigüedad, evita la tentación de adivinar.
13. Debería haber una, y solamente una, manera obvia de hacerlo.
14. A pesar de que no sea obvio a menos que seas Holandés (como GvR)
15. Ahora es mejor que nunca.
16. A pesar de que nunca es muchas veces mejor que *justo ahora*.
17. Si la implementación es difícil de explicar, es una mala idea.
18. Si la implementación es fácil de explicar, quizás sea una buena idea.
19. Los espacios de nombres son una gran idea, ¡tengamos más de esas!

Esta filosofía ha hecho que Python sea considerado un leguaje de referencia en el mundo Maker y para el desarrollo del IoT. Actualmente está convirtiéndose en el lenguaje más usado en enseñanza, y se prevé que en poco tiempo pueda superar en uso a lenguajes más establecidos como C.

La arquitectura de software de nuestro programa se muestra en la figura 35. Se ha decidido optar por una programación modular que facilite la legibilidad de nuestro código, así como incrementar su versatilidad ante futuras versiones. Hay que tener en cuenta que el firmware de arranque que lleva cargado por defecto la placa LoPy4, y que se ha decidido mantener ya que no es objeto de este proyecto el acceso a funciones de bajo nivel o ensamblador para modificar el firmware con el que viene ya la placa, reconoce dos scripts al iniciarse, main.Py y boot.Py. El primero es el hilo de ejecución principal, y se mantendrá mientras este activa la placa, el segundo son las funciones que se ejecutan al arranque del dispositivo

A continuación, se detallan los módulos y funciones más destacadas de la programación realizada, para consultar el código completo referirse al ANEXO X del presente documento.

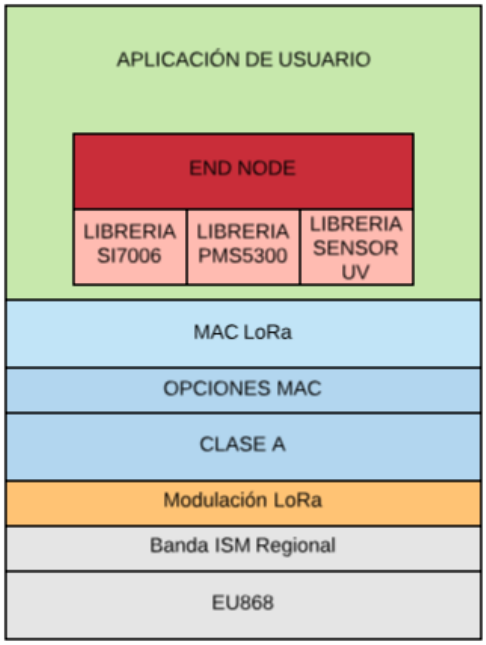


FIGURA 30 ARQUITECTURA DE SOFTWARE DEL SISTEMA

5.4.2.1 Entorno de desarrollo ATOM con modulo PYMAKR

Debido de nuevo al interés en código abierto, así como en la plataforma GitHub, se ha decidió hacer uso del IDE creado para esta plataforma, ATOM.

ATOM ha sido durante años un editor de texto plano, potente y versátil, totalmente integrado con la plataforma GitHUB. Debido a la aparición en los últimos años de múltiples editores similares, y que ofrecían nuevas funciones como Visual Studio Code, o Brackets, ATOM se ha visto obligado a evolucionar y ha decidido generar un entorno de desarrollo completo integrado, o IDE por sus siglas en ingles.

Este proyecto es un ambicioso intento de crear una plataforma de desarrollo para casi cualquier tipo de desarrollador. A través de paquetes fácilmente instalables, permite desarrollar en múltiples lenguajes, entre los que se encuentra Python.

Además de paquetes de lenguajes completos, con las funciones de autocompletar, corrección sintáctica o formateo de código. ATOM nos ofrece la opción de implementar paquetes extras como es el caso de PYMAKR, un paquete desarrollado por PYCOM con el fin de programar sus dispositivos.

Pymakr es un módulo que permite comunicar con cualquier tarjeta Pycom a través de una línea de comandos REPL. Esto permite cargar archivos a nuestra tarjeta, ejecutarlos o incluso escribir comandos que ejecutará directamente el microcontrolador.

Una vez instalado el plugin de ATOM, la línea de comandos se abrirá de forma predeterminada. Mediante el botón de ajustes en la esquina superior derecha se puede ajustar la configuración,

esto generará un fichero que se cargará en nuestra placa y guardará la configuración predeterminada que marquemos.

Mediante el botón de connect se iniciará la conexión con la placa que se encuentre en el puerto serie. Es importante en Linux habilitar los permisos de escritura y lectura del puerto donde se encuentre el dispositivo, esto no será necesario en Windows.

Una vez iniciada la conexión podemos comenzar a trabajar con nuestro desarrollo. Pymakr nos permite ir probando el código que vamos desarrollando sin necesidad de cargarlo en la placa mediante en el botón "run". Es importante destacar que esta opción solo arrancará el archivo sobre el que estemos trabajando, por lo que de existir dependencias hacia librerías en nuestro archivo a ejecutar, será necesario que estas se carguen previamente en la placa, de otra forma se generará un error en la ejecución.

Algunos comandos útiles durante el desarrollo se muestran a continuación:

- ctrl-opt/alt-c: (Re) conectar
- ctrl-opt/alt-d: Desconectar
- ctrl-opt/alt-t: Terminal de palanca
- ctrl-opt/alt-s: Subir proyecto
- ctrl-shift-s : Subir solo el archivo actual
- ctrl-opt/alt-r: Ejecutar archivo actual
- ctrl-shift-enter: Ejecutar selección actual o línea
- ctrl-r: Borrar terminal

5.4.2.2 Biblioteca Network y configuración LoRa

La placa Pycom cuenta con varias librerías ya cargadas por defecto, entre la que se encuentra Network, una librería en la que tenemos todas las funciones de red y configuraciones necesarias para que se pueda hacer uso de los cuatro tipos de conexión con los que cuenta nuestra placa.

En nuestro desarrollo no es necesario incorporar toda la librería con lo que solo importaremos a nuestro proyecto la clase LoRa que nos permitirá hacer uso de esta conectividad. En nuestro desarrollo hemos decidido crear un módulo aparte en el que prepara la configuración de nuestro dispositivo a la frecuencia europea. Le hemos denominado EU868 y le devuelve dos funciones de las que haremos uso en nuestro archivo principal, prepare_chanels, y lora_cb, función de callback para paquetes RX.

Lo primero para hacer uso de las funciones LoRa es crear un objeto tipo LoRa mediante el constructor network.LoRa (). De esta manera generamos nuestro objeto pasándole los parámetros que consideremos necesarios. En nuestro caso hemos definido el modo de funcionamiento, la región y el tipo de dispositivo.

```
lora = LoRa ( modo = LoRa. LORAWAN , region = LoRa. EU868 , device_class = LoRa. CLASS_A )
```

Comenzamos preparando los canales, y para eso hacemos uso de nuestra función ya creada prepara_chanels. A esta función le pasamos el objeto LoRa creado, el canal que queremos y el flujo de datos. Mediante los siguientes métodos se agregan y eliminan los canales en los que transmitiremos;

- lora.add_channel(índice, * , frecuencia, dr_min, dr_max). A traves de ella se añade un canal LoRaWAN en el índice que le especifiquemos.
- Lora.remove_channel(indice) que nos permitirá eliminar el canal especificado. En el caso de la banda europea, lo canales del 0 al 2 no puede eliminarse, por lo que si se quiere usar un solo canal el mejor método es agregar el mismo canal en los índices 0, 1 y 2.

A continuación, podemos iniciar la conexión y comenzar a transmitir mensajes mediante LoRaWAN. Para ello emos hecho uso del método:

lora join (activación , auth , * , timeout , dr).

Este es el método para conectarse a la red Lora. Para usarlo es necesario pasarle el método de activación escogido OTA o ABP, los parámetros de autentificación, el tiempo de espera para recibir el mensaje de aceptación, y el flujo de datos inicial.

En nuestro caso hemos elegido activación OTA, la tupla de activación será por tanto el EUI del dispositivo, clave de la aplicación y clave de la red. Estas dos últimas se obtienen, como ya se mencionó anteriormente, de la plataforma TTN, mientras que la primera la podemos sacar de nuestro dispositivo de diversas maneras, una de ellas es el método lora.MAC(). El timeout lo hemos fijado en cero, lo que nos permite hacer uso del método lora.has_joined(), que nos devolverá True en caso de que se establezca la conexión.

Una vez iniciada la conexión, se pueden establecer los sockets que nos permitirán pasar información a través de UTF-8 hacia el servidor.

Para hacer uso de ellos lo primero es importar la librería socket y crear un objeto del tipo socket a través del siguiente constructor

s = socket.socket(socket.AF_LORA, socket.SOCK_RAW).

Los sockets creados en LoRa son compatibles con diversos métodos de la librería socket, no obstante, en este proyecto solo se ha hecho uso de tres:

socket.setsockopt (nivel, nombre de operación, valor) mediante este método podemos escoger la operación y fijar un valor. Hemos hecho uso de esta función con el fin de establecer el flujo de datos y confirmar la recepción de mensajes.

- Socket.blocking (Flag) la cual permite hacer que el socket este bloqueado o no por defecto.

- Socket.send(Bytes) que pasa la información a través del socket. Solo transmite datos en forma de bytes, por lo que es importante realizar la conversión antes de enviar ningún mensaje.

Con estos métodos se puede tener un dispositivo que transmita mensajes a través de LoRaWAN de una manera sencilla y eficiente, en unas pocas líneas tenemos configurada la conexión y transmisión de datos desde el nodo hacia la pasarela TTN pasando por un Gateway.

De todas las clases que contiene la librería LoRaWAN, y de las que no se han hecho uso en este proyecto, hemos decidido mencionar tres de ellas que, aunque no se han implementado, deberán serlo en próximas revisiones.

- lora.set_battery_level (nivel) mediante la cual se pasa el estado de la batería a través de un nivel variante entre 1 y 254, siendo el 0 alimentación por batería y 255 un fallo en la lectura de la batería.
- Lora.nvram_save() Permite guardar la información de la conexión realizada en la memoria no volátil del dispositivo
- Lora.nvram_restore () Permitirá transmitir mensajes de manera inmediata tras salir del modo deep-sleep, sin necesidad de iniciar de nuevo la conexión.

Como se ve son funciones especialmente útiles para el ahorro y la gestión de la batería, uno de los pilares principales del protocolo LoRaWAN.

5.4.2.3 Biblioteca SI7006A20

Los controladores necesarios para el sensor de humedad y temperatura de Silicon Labs vienen incluidos en la librería SI7006A20. Esta clase cuenta con una clase para crear un objeto capaz de devolver los estados de temperatura y humedad. Además de esto cuenta con diversas funciones preestablecidas que permiten devolver valores como el punto de rocío, o una humedad ajustada a la temperatura.

El constructor que genera el objeto es el siguiente:

SI7006A20(pysense = None, sda = 'P22', scl = 'P21')

Para que este constructor sea valido será necesario pasarle un objeto I2C, o en este caso un objeto que permite establecer las comunicaciones I2C de la placa sobre la que se monta el chip, es el objeto Pysense.

Los métodos principales de esta clase son dos, y nos permiten obtener la temperatura y humedad en tipo float:

SI7006A20.humidity() a través del cual podemos obtener la humedad

SI7006A20.temperature() para leer la temperatura.

Los datos devueltos por la clase son de tipo float, por tanto, será necesaria su conversión a bytes con el fin de poder transmitirlos a través de mensajes LoRaWAN. Python contiene múltiples métodos para transformar datos a bytes. Antes de iniciar cualquier conversión, por sencillez, y

entendiendo que no es necesario múltiples decimales para el tratamiento que realizaremos de los datos, vamos a almacenar la temperatura en tipo INT. Una vez en este tipo de dato, podemos usar la función int.to_bytes() que transformará la temperatura y humedad en bytes para su transmisión.

5.4.2.6 Aplicación final

La aplicación final simplemente llama a cada una de las funciones creadas y es muy sencilla.

El hilo de ejecución seria; tras la activación, aprovechando la biblioteca Lora, buscamos el Gateway más cercano disponible, y mediante protocolo OTA nos conectamos a él. El hilo de ejecución esperará hasta que hayamos conseguido comunicar. Una vez realizado, se toman los valores de temperatura, humedad y presión y se pasan a binario para su transmisión a través del protocolo LoRaWAN.

Una vez conectados a la red, y los valores han sido tomados, simplemente mandamos la trama datos a la aplicación creada en TTN. En este punto hemos dejado una espera de 5 segundos para asegurar tiempo suficiente para terminar la transmisión en curso.

Para finalizar el nodo quedaría en suspensión mediante la función machine.deepsleep(), lo que nos permitirá ahorra batería dejando la placa en modo mínimo consumo. En la función citada solo tenemos que definir el tiempo de suspensión, y una vez alcanzado este tiempo se reiniciará y comenzará un nuevo ciclo de toma y transmisión de datos. Por otra parte, con la información pasada a TTN, tenemos un PAYLOAD en el que se encuentra la temperatura, la humedad, la presión y la huella temporal de nuestro dispositivo. Con el fin de obtener datos que puedan ser pasados a nuestro panel web, es necesario añadir unas líneas de código adicionales a nuestra aplicación TTN . Para nuestro caso con las líneas siguientes es suficiente para obtener de nuevo los datos en forma legible.

```
function bin2String(array) {
  var result = "";
  for (var i = 0; i < array.length; i++) {
    result += String.fromCharCode(parseInt(array[i], 10));
  }
  return result;
}
function Decoder(bytes, port) {
  var decoded = {};
  decoded.temperature = Math.round((((bytes[1]<<8) + bytes[0])/100- 273.15), -1);
  decoded.humidity = ((bytes[3]<<8) + bytes[2])/100;
  decoded.pressure = ((bytes[5]<<8) + bytes[4]);
  return decoded;
}
```

Para mayor referencia del código usado y la aplicación final ver el anexo adjunto. En el se puede observar claramente que con las librerías facilitadas por PYCOM y unas pocas líneas de código es muy sencillo generar una aplicación funcional y con una utilidad real en muy poco tiempo.

5.5 Panel web

5.5.1 EC2 AWS

Como prueba de concepto, y con el fin de testear y aprovechar las principales tecnologías relacionadas con el IoT y la nube, se ha decidido hacer uso de la plataforma Amazon Web Service, AWS de ahora en adelante.

Aunque AWS cuenta con su propio entorno para desarrollo de aplicaciones IoT, nosotros hemos decidido hacer uso de esta plataforma solamente como servidor para alojar nuestro desarrollo en Node-Red.

Amazon Web Service ofrece una suite completa de computación y desarrollo en la nube, en ella es posible desde crear máquinas virtuales sencilla y servicios web simples, hasta desarrollar aplicaciones de inteligencia artificial o hacer uso de Alexa, la IA de Amazon. En nuestro caso hemos hecho uso de la plataforma E2C de Amazon, sobre la que construiremos una máquina virtual sencilla que aloje nuestra aplicación.

AWS nos ofrece una serie de pruebas gratuitas durante 1 año, entre ellas se encuentra el uso de 750h de instancias EC2 al mes. Lo que es más que suficiente para validar y mostrar nuestra aplicación durante un tiempo limitado. En nuestro desarrollo vamos a implementar una imagen de Ubuntu en un servidor sobre el que posteriormente lanzaremos la aplicación desarrollada en Node-Red.

Para ello entramos en la consola de AWS EC2, se puede ver en la figura 36 y pulsamos en ‘Launch Instance’. Una vez dentro nos da diversas opciones, elegimos Ubuntu Server, y el tipo de instancia t2.micro que entra dentro de la prueba gratuita de Amazon. Esta instancia cuenta con un procesador básico y 1GB de RAM, suficiente para nuestro proyecto. Ahora se nos da una serie de parámetros que podemos configurar a nuestra elección, algunos de ellos implican un coste adicional. Nosotros solo hemos modificado la confutación del grupo de seguridad, en donde hemos añadido una nueva regla TCP personalizada para el puerto 1880. Finalmente, pulsamos en ‘Launch’, y nos aparecerá una ventana para configurar las claves de SSH. Seleccionamos ‘Download Key pair’ y generaremos un archivo .pem con los cifrados para conectarnos desde cualquier terminal mediante SSH.

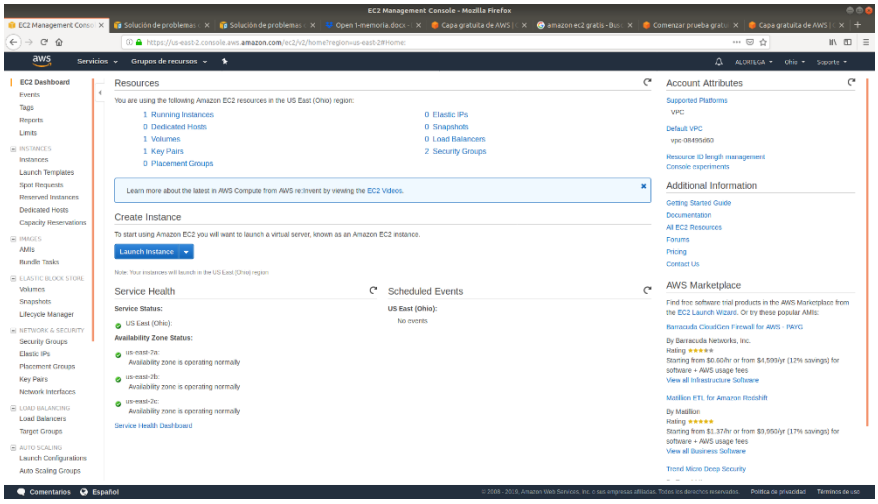


FIGURA 31 CONSOLA AWS

Una vez hemos generado una instancia, solo es necesario conectarse a ella mediante SSH. Aprovecharemos el software nativo instalado en Linux para comunicación SSH. Antes de comenzar, debemos asegurar que los permisos del archivo .pem descargado sean 400 y no 0777. Para ello usaremos la siguiente línea:

```
$ chmod 400 Comunidad40.pem
```

Una vez configurado, nos conectamos al servidor mediante SSH, debemos pasar las claves de certificación mediante el archivo .pem, y la DNS privada que podemos obtener de AWS. El comando en nuestro caso sería el siguiente:

```
$ ssh -i Comunidad40.pem ubuntu@ec2-18-219-194-1.us-east-2.compute.amazonaws.com
```

Esto nos dará acceso remoto a nuestro servidor y podremos trabajar en el igual que se hizo previamente con la Raspberry Pi. Ya podremos comenzar a instalar Node-Red.

5.5.2 NODE_RED

Una vez desplegado Ubuntu en AWS, instalaremos Node-Red. Clonamos el directorio de instalación de Node.js en nuestro sistema mediante la siguiente linea.

```
curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash –
```

A continuación, instalamos Nodejs:

```
$ sudo apt-get install -y nodejs build-essential
```

Una vez terminada, tendremos acceso al gesto de paquetes npm que nos permitirá lanzar la instalación de Node-Red.

```
$ sudo npm install node-red.
```

5.5.2.1 Configuraciones de Seguridad e inicio.

Node-Red, como ya se ha mencionado, corre sobre un navegador web, y se genera en el puerto local del equipo. En este desarrollo se ha dejado la IP de nuestro servidor accesible, así como el puerto, para poder mostrar nuestra aplicación web. Esto implica que el panel de administración de Node-Red es igualmente accesible, y requiere de una configuración adicional para protegerlo.

Para ello se debe acceder al archivo settings.js dentro de la carpeta de node-red. Una vez en él, mediante un editor de texto, des comentamos las siguientes líneas de código:

```
adminAuth:
 {    type: "credentials",
   users: [{        username: "admin",
      password:
"$2a$08$zZWtXTja0fB1pzD4sHCMyOCMYz2Z6dNbM6tl8sJogENOMcx
WV9DN.",        permissions: "*"    }]}
```

Cambiamos “admin” por nuestro usuario, y la contraseña por la que queramos. Para generar el nuevo password, debemos hacer uso del encriptador facilitado por node-red, el cual nos generará un ‘Hash’ de la contraseña elegida para sustituir la de defecto.

Como ajuste adicional, habilitaremos el cifrado https para asegurar la privacidad de la página. Usaremos un certificado generado mediante “openssl”, una herramienta gratuita que nos permite generar nuestros propios certificados de manera rápida y sencilla. Estos certificados son igual o más seguros que los comerciales, no tienen soporte en la mayoría de los navegadores, lo que nos devolverá en ocasiones un mensaje de sitio web inseguro.

Para incluir este protocolo seguro, eliminamos las marcas de comentario de las siguientes líneas del archivo settings.js, y añadimos la ruta a los archivos que contienen las claves y los certificados, node-key.pem y node-cert.pem en nuestro desarrollo.

```
var fs = require("fs");
…....
https: {
                key: fs.readFileSync('/home/ubuntu/.node-red/node-key.pem'),
                cert: fs.readFileSync('/home/ubuntu/.node-red/node-cert.pem')
   },
…....
requireHttps: true,
```

Aunque la ciberseguridad es un pilar fundamental, así como la principal fuente de riesgo del IoT, debido a su complejidad y extensión, se ha decidido no incluir mayores referencias a la misma en este proyecto. Se omiten por tanto explicaciones y detalles sobre cifrado, así como elementos de autentificación usados durante el presente proyecto.

Node-Red tiene una complicación más que debemos solventar con la configuración, y es que por defecto almacena la información en que se le pase en la memoria volátil de la aplicación, por lo que una vez reiniciado el proceso este se pierde automáticamente. Con el fin de solventar esto, debemos habilitar la función contextStorage dentro de settings.js. Para ello introducimos el siguiente código dentro del archivo.

```
contextStorage: {
        default:"memoryOnly",
            memoryOnly: {module: 'memory'},
            file: {module:'localfilesystem'}
        },
```

Con el comando default le indicamos que por defecto almacene los datos en la memoria, posteriormente le indicamos la posibilidad de guardar los datos en el sistema de archivos locales. Esta configuración nos permitirá indicarle que almacene exclusivamente los datos de nuestro interés, dejando al resto en la memoria del proceso. Con esta configuración, podremos elegir en cada Nodo donde queremos que se almacene la información.

Como paso final de la configuración, se hará uso del gestor de procesos pm2. Mediante el mismo se pretende mantener activa la aplicación node-red, así como reiniciarla cuando lo haga la instancia de AWS. Esto solventará la necesidad de acceder a la instancia mediante ssh cada vez que se reinicie.

PM2 se encuentra dentro del repositorio estándar y puede ser instalado directamente mediante el gestor de paquetes npm.

```
pm2 start `node-red` -- -v
  pm2 save
  pm2 startup
sudo npm install -g pm2
```

Tras la última línea de comandos, nos pedirá que copiemos un último comando, lo hacemos y lo lanzamos. Reiniciamos finalmente el sistema, y si la configuración ha sido correcta, node-red se lanzará de manera automática.

Name	id	mode	status	↺	cpu	memory	
node-red	1	N/A	fork	online	0	0%	15.3 MB
which node-red	0	N/A	fork	online	0	0%	3.1 MB

FIGURA 32 GESTOR DE ARRANQUE PM2

5.5.2.1 Panel

El panel de usuario pretende ser sencillo e intuitivo. En el menú desplegable nos permite escoger entre los distintos nodos que están transmitiendo su estado, como ejemplos hemos incluido el garaje, la piscina y nuestro desarrollo actual, el parque infantil.

En la pantalla se nos muestra el estado actual y una gráfica con el histórico de estados. En el desarrollo actual se nos muestra la temperatura, la humedad y la presión barométrica, en futuras implementaciones se pretende añadir la calidad del aire la incidencia de luz ultravioleta.

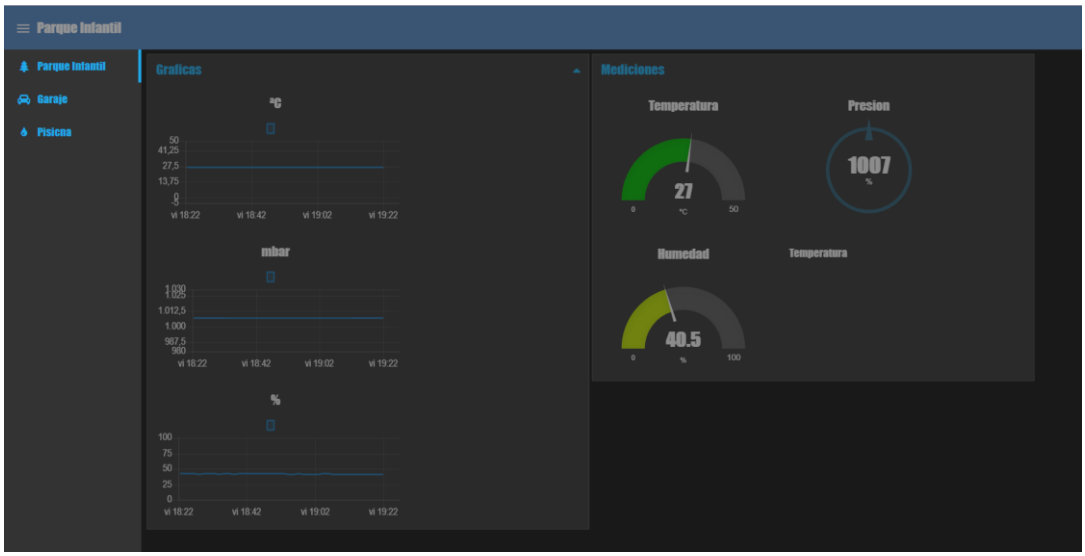


FIGURA 33 PANEL WEB

5.6 Dimensionado de la batería

Por sencillez, precio y disponibilidad del mercado se ha decidió hacer uso de baterías de polímero litio o LiPo por sus siglas en inglés. Este tipo de baterías se basan en sustancias semisólidas tipo gel, lo que permite reducir drásticamente su tamaño y adaptarlas a casi cualquier forma. Este tipo de baterías son las más usadas en la actualidad debido a su capacidad y reducido tamaño. Como mayor ventaja se encuentra su precio, podemos localizar este tipo de baterías en rangos de capacidad de 1000 mAh por menos de 2€.

A continuación, se va a calcular el dimensionado de nuestra batería, así como proponer su vida útil en condiciones normales de funcionamiento. Para ello lo primero es definir los diferentes estados, su consumo y el tiempo que se encuentra en cada estado. Para el dimensionado de la batería hemos tenido en cuenta el consumo de los sensores ultravioleta y de calidad del aire que se pretenden incluir en una próxima actualización.

Los modos de funcionamiento acorde a la estructura del sistema son 3.

1. Medida de los sensores: En este momento se toman datos de todos los sensores, debiendo estar activo por tanto cada sensor junto al MCU, generando consumo así a través del bus I2C y del puerto serie.
2. Transmisión LoRaWAN: Durante este periodo se inicia la conexión y se transmiten los paquetes de datos. Para ello es necesario tener activos transceptor LoRA y el microcontrolador.
3. Modo reposo: En este momento se hará uso de la función deep_sleep ya mencionada en este proyecto. Se dejará a todos los sistemas en un modo de sueño profundo de mínimo consumo.

En las siguientes tablas se puede observar los tiempos en cada estado, así como los consumos en función de las hojas de características facilitadas por los proveedores de cada componente:

Modulo	Consumo activo	Consumo Desactivado
MCU	35.4 mA	18.5 µA
SI7006-A20	60 µA	1 µA
PMS5003	100 mA	200 µA
Sensor UV	100 µA	6 µA
LoRa TX	108mA	-

Estado	MCU	SI7006-A20	PMS5003	Sensor UV	LoRa Tx	Consumo
Sensores	1	1	1	1	0	101.95 mA
LoRaWAN	1	0	0	0	1	108.35 mA
Deep_sleep	0	0	0	0	0	0.22 mA

Según el planteamiento del diseño, las medidas se toman en tramos de 30 min, se dedicará un tiempo para las mediadas de 20 segundos para que dé tiempo suficiente al sensor de partículas a tomar las muestras necesarias, y 30 segundos a la transmisión de paquetes y conexión.

Acorde a estos tiempos podemos realizar el consumo según estados, siendo así:

ESTADO 1:

$$101{,}95mA * 20.000ms = 2039000\ mAms * \frac{1h}{3600000ms} = 0{,}57\ mAh$$

ESTADO 2:

$$108{,}35mA * 30.000ms = 2039000\ mAms * \frac{1h}{3600000ms} = 0{,}90\ mAh$$

ESTADO 3 :

$$0{,}22mA * 1.800.000ms = 2039000\ mAms * \frac{1h}{3600000ms} = 0{,}11\ mAh$$

El consumo total de un ciclo será, por tanto:

$$0{,}57\ mAh + 0{,}90mAh + 0{,}11\ mAh = 1{,}58mAh$$

A partir de este punto podemos escoger la batería más adecuada en función del tiempo que queremos mantener nuestro dispositivo sin necesidad de volver a cargarlo. Hemos decidido fijar este periodo en 6 meses. Tal y como se indicó en este proyecto, el tiempo de funcionamiento del nodo se ha fijado entre las 9:00 y las 20:00 . Así, en el periodo de 1 año se realizarán aproximadamente 1.000 ciclos, lo que implicará contar con una batería de aproximadamente 1.500 mAh. Teniendo en cuenta consumos en los tiempos de inactividad, y la descarga con el tiempo de la batería, vamos a incorporar una batería LiPo de 2.000 mAh. Estas baterías son pequeñas y con un coste de aproximadamente 10€.

5.7 Pruebas de funcionamiento

5.7.1 Gateway

Para comprobar el correcto funcionamiento del Gateway la primera prueba ha sido vincularlo a nuestra cuenta de TTN. En este punto nos muestra el estado de nuestro dispositivo y si ha sido correctamente configurado.

Una vez realizada esta comprobación hemos realizado un mapeo de la señal a través de TTN Maper y nuestro nodo configurado para conectarse y mandar su huella temporal cada vez que se reiniciaba mediante el botón incluido en la placa. Esto nos ha mostrado un claro problema en nuestro desarrollo. La gran ventaja de este desarrollo, su gran alcance, se ve gravemente limitado por la calidad de las antenas y los obstáculos físicos en el camino de la señal. En nuestro caso no nos hemos podido alejar más de 500 m de la ubicación de nuestro Gateway ya que se encuentra actualmente en el interior de la vivienda y con una antena de bajo coste. No obstante, este alcance es más que suficiente y hemos sido capaces de enviar y recibir paquetes desde la ubicación del parque infantil, la piscina y el garaje con una calidad y velocidad bastante altas.

Validado el alcance hemos hecho una prueba intensiva de funcionamiento manteniendo el Gateway activo durante un par de meses, en este tiempo hemos ido realizando envíos de

paquetes y algunas pruebas con el nodo. El mayor problema que hemos detectado es que debido a la configuración de seguridad, en caso de pérdida de alimentación, es necesario el inicio manual mediante la introducción de la contraseña.

En la figura 34 podemos observar el Gateway en su versión más básica, para las pruebas de funcionamiento. En la figura 35 podemos observar la versión completa del Gateway en su emplazamiento final.

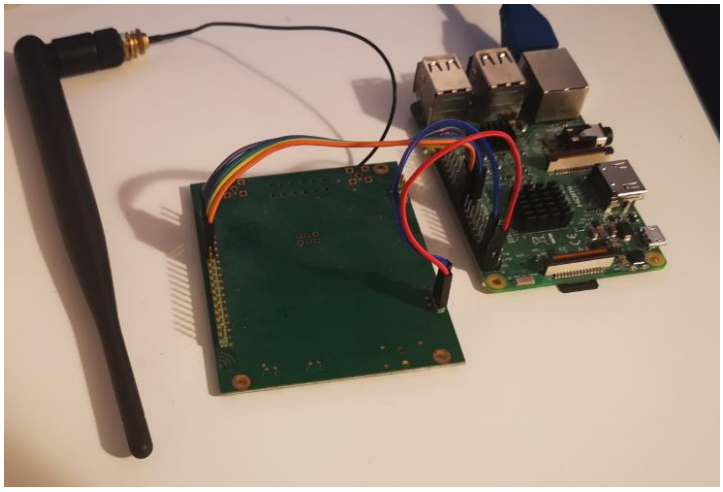


FIGURA 34 DISPOSITIVO EN PRUEBA DE FUNCIONAMIENTO

FIGURA 35 UBICACIÓN DEL GATEWAY

5.7.2 Nodo

El testeo del nodo se ha realizado en condiciones de alimentación continua. Para simular un uso más intensivo hemos tomado mediciones durante varios días cada 3 min. No podemos usar una frecuencia de envío mayor, ya que estaríamos contraviniendo la normativa de uso de la banda ISM.

En este aspecto no hemos encontrado mayores problemas de uso, es cierto que se ha observado un retraso en algunos envíos de paquetes, y que al configurar 3 minutos usando deepsleep, el tiempo no es real, ya que la demora en conectar de nuevo y volver a levantar la pasarela es de entorno 1min.

La prueba de funcionamiento se pude considerar adecuada.

En la figura 36 podemos ver el nodo en su versión básica utilizada para las pruebas de validación. La batería es inferior a la versión final, pero suficiente para mantener el dispositivo varias horas funcionando sin mayores problemas. En la figura 37 se puede observar el nodo ubicado en su posición final dentro del parque infantil.

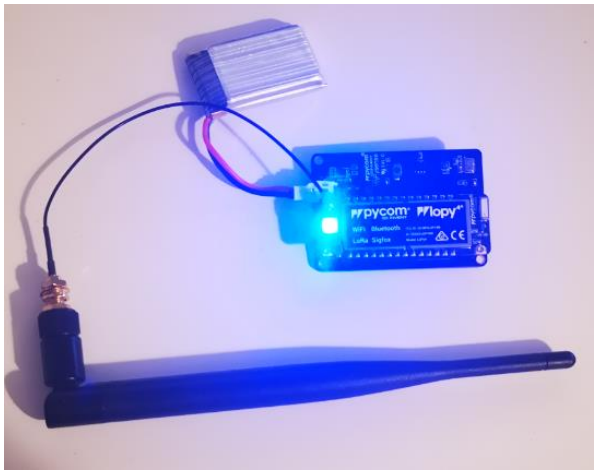


FIGURA 36 NODO VERSIÓN MÍNIMA PARA PRUEBA DE FUNCIONAMIENTO

FIGURA 37 NODO EN SU UBICACIÓN FINAL EN EL PARQUE INFANTIL

6. CONCLUSIONES

Durante el desarrollo de este proyecto se ha pretendido hacer uso del máximo número de disciplinas trabajadas durante el grado, ampliando y fortaleciendo lo adquirido. Así, hemos realizado el diseño tanto de sistemas digitales como analógicos.

La versatilidad y escalabilidad de este desarrollo queda limitada al hardware, lo que es fácilmente ampliable. Al hacer uso de tecnologías tan amplias como LoRa, AWS, la plataforma TTN, Node-red, o el entorno de desarrollo ATOM, no parece que vaya a existir una limitación a la hora de añadir nuevas funcionalidades o ampliar la red iniciada en este proyecto.

A nivel académico he conseguido un amplio conocimiento en el protocolo de red LoRaWAN y en consiguiente la tecnología LoRa. Esto amplia los conocimientos transversales de la titulación, abriendo nuevas áreas que dentro del plan formativo no ha sido posible alcanzar.

Otro aspecto positivo que ha aportado este proyecto a mi formación es el uso constante y la familiarización con el entorno Linux, hasta entonces desconocido. La posibilidad de trabajar dentro de este ecosistema me ha mostrado una nueva forma de entender los sistemas operativos y sus procesos de una manera más profunda. El uso de Linux como base de la mayoría de los sistemas embebidos como en la Raspberry Pi es innegable, y ser capaz de entenderlo y trabajar con el de manera fluida otorga una herramienta muy importante para cualquier futuro desarrollo.

Siguiendo con el software agradezco a este proyecto la posibilidad de haber conocido herramientas tan potentes como ATOM, Node_red, así como haber podido trabajar dentro de la nube de Amazon, un punto que sin duda seguiré trabajando en próximos desarrollos orientados a IoT. La creación sencilla de máquinas virtuales gratuitas, o sus diversas aplicaciones para la creación de desarrollos IoT son elementos muy destacables de AWS.

En lo que respecta al uso final de este desarrollo, considero que tiene un potencial enorme. Con un bajo coste se puede ofrecer un servicio de monitorización a un área muy amplia, este proyecto pretende ser un punto de partida y creo que queda patente a lo largo de esto documento la facilidad con la que podríamos levantar esta red y dar servicio a cientos de comunidades de vecinos. Creo que es importante seguir apoyando esta tecnología y entre todos hacer crecer a propuestas y plataformas tan interesantes como TTN.

7. BIBLIOGRAFÍA

- Alimentación POE https://kb.netgear.com/es/209/Qu%C3%A9-es-PoE-alimentaci%C3%B3n-a-trav%C3%A9s-de-Ethernet
- Github, TTN-ZH. 2017.Recuperado de:https://github.com/ttn-zh/ic880a-gateway/wiki
- IMST GmbH, Datasheet IC880A. Germany, 2016.
- https://www.ibm.com/developerworks/ssa/library/
- MQTT Version 3.1.1. Recuperado de http://mqtt.org/documentation
- Protocolo SPI TFG_Salvador_Cristina_Garcia_2014.pdf. Recuperado de https://e-archivo.uc3m.es/
- Semtech, What is LoRa. 2017. Recuperado de: http://www.semtech.com/wireless-rf/internet-ofthings/what-is-lora/

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Desarrollo de una pasarela y nodos LoRaWAN como solución de bajo coste para implantaciones IoT.

2. PLANOS

Autor:

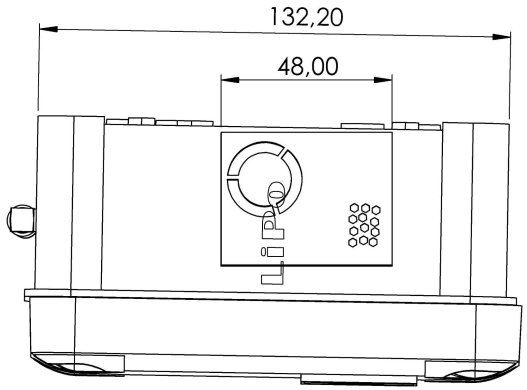
D. Alejandro Ortega Pérez

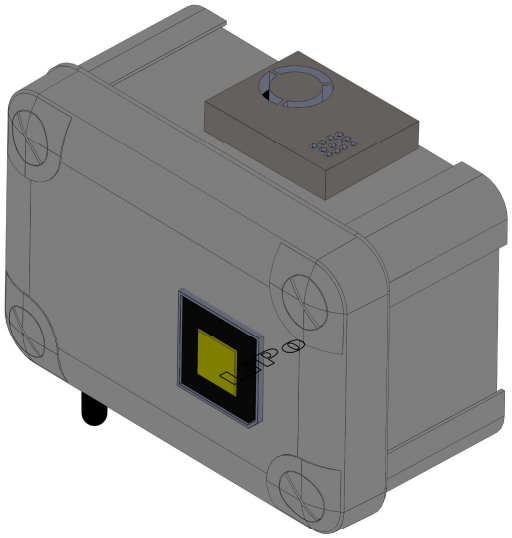
Tutor:

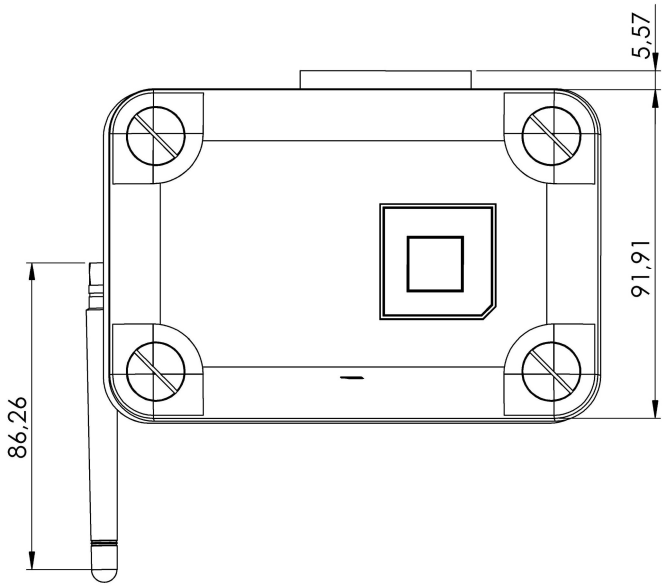
D. Ángel Perles Ivars

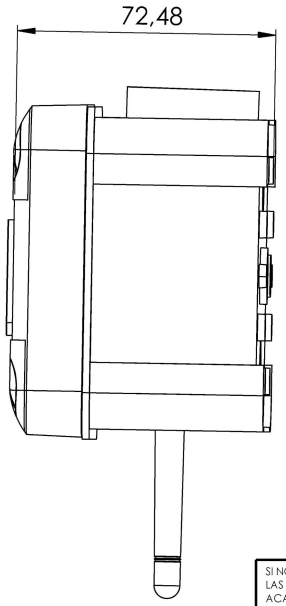
Valencia, diciembre de 2019

Índice de planos

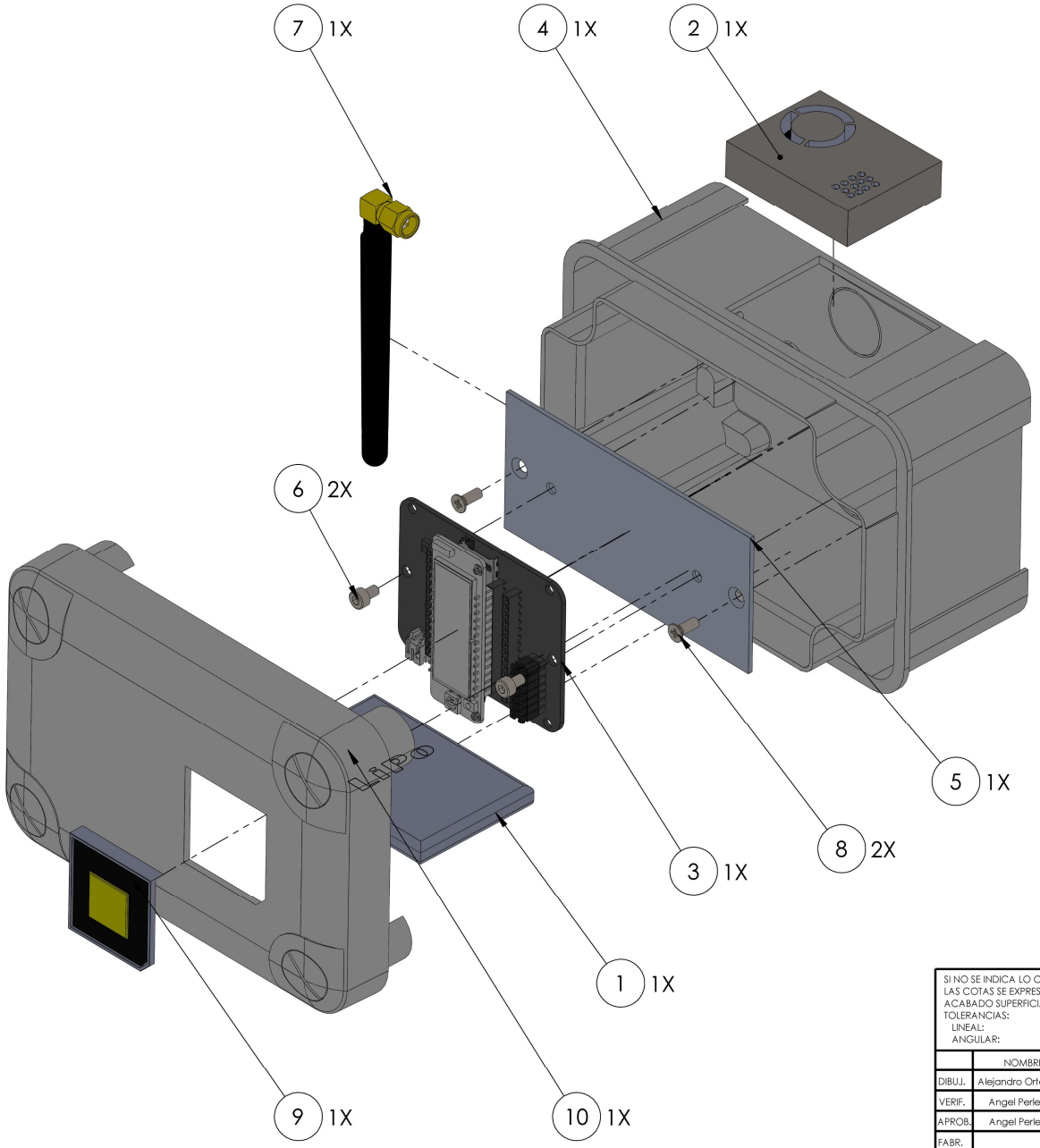








SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		Escala 2:3	REVISIÓN
						UPV	
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.	Alejandro Ortega		02/11/2019			Nodo 1 Dimensiones	
VERIF.	Angel Perles		02/11/2019				
APROB.	Angel Perles		02/11/2019				
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO	A3
						NODO0001_DIM	
				PESO:		ESCALA:1:2	HOJA 1 DE 1



N.º DE ELEMENTO	N.º DE PIEZA	DESCRIPCIÓN	CANT.
1	BAT0001	BATERIA LIPO 2000 MAH	1
2	ES0001	LASER DUST SENSOR	1
3	EB0001	EXPANSION SHEILD MULTI-SENSOR	1
4	ENV0001_1	NSYTBP1397	1
5	CH0001	CHAPA DE MONTAJE	1
6	TOR0001	DIN 912 M3 L6	2
7	TR0001	ANTENA RF SMA 868MHZ 3,5DBI	1
8	TOR0001	DIN 7991 M3 L6	2
9	ES0002	UVA UVB AND UV INDEX SENSOR	1
10	ENV0001_2	NSYTBS775	1

SI NO SE INDICA LO CONTRARIO:
LAS COTAS SE EXPRESAN EN MM
ACABADO SUPERFICIAL:
TOLERANCIAS:
LINEAL:
ANGULAR:

ACABADO:

REBARBAR Y ROMPER ARISTAS VIVAS

	NOMBRE	FIRMA	FECHA
DIBUJ.	Alejandro Ortega		02/11/2019
VERIF.	Angel Perles		02/11/2019
APROB.	Angel Perles		02/11/2019
FABR.			
CALID.			

MATERIAL:

PESO:

Escala 2:3

REVISIÓN

UPV

TÍTULO:

Vista de explosión Nodo

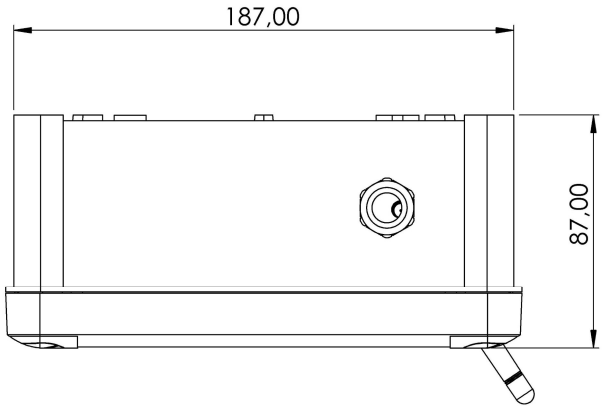
N.º DE DIBUJO

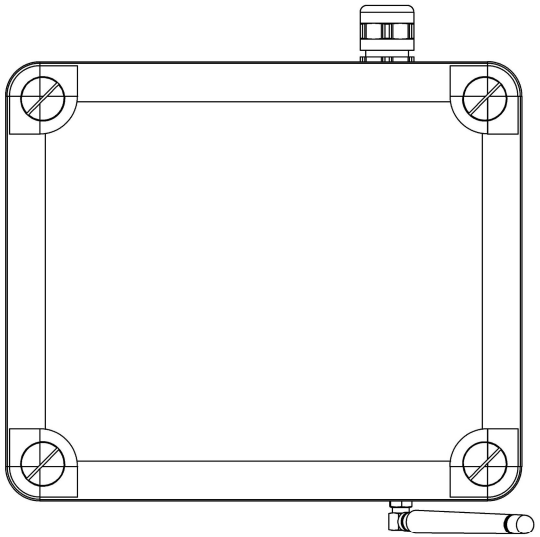
NODO0001

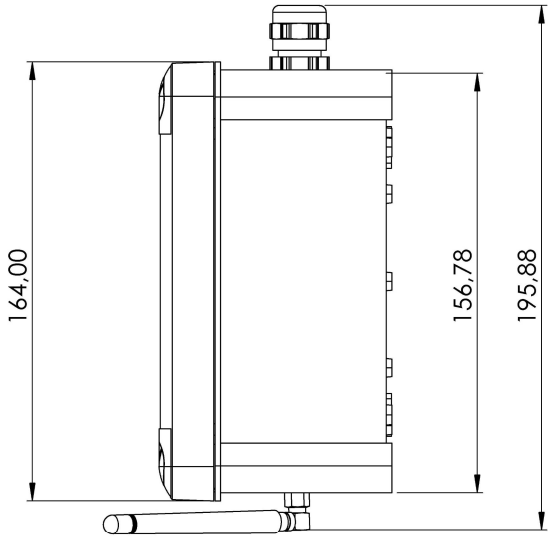
A3

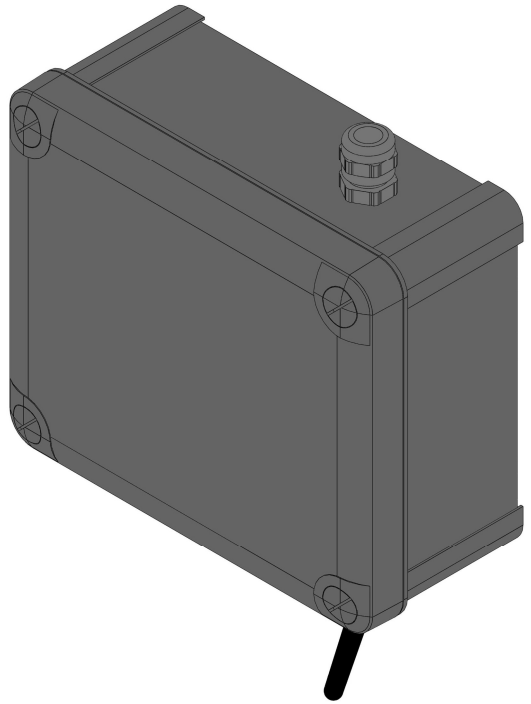
ESCALA:1:2

HOJA 1 DE 1









SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:				REBARBAR Y ROMPER ARISTAS VIVAS	
	NOMBRE	FIRMA	FECHA				
DIBUJ.	Alejandro Ortega		15/09/2019				
VERIF.	Ángel Perles						
APROB.	Ángel Perles		12/10/2019				
FABR.							
CALID.				MATERIAL:			
				PESO:			

ESCALA: 2 : 3 | REVISIÓN

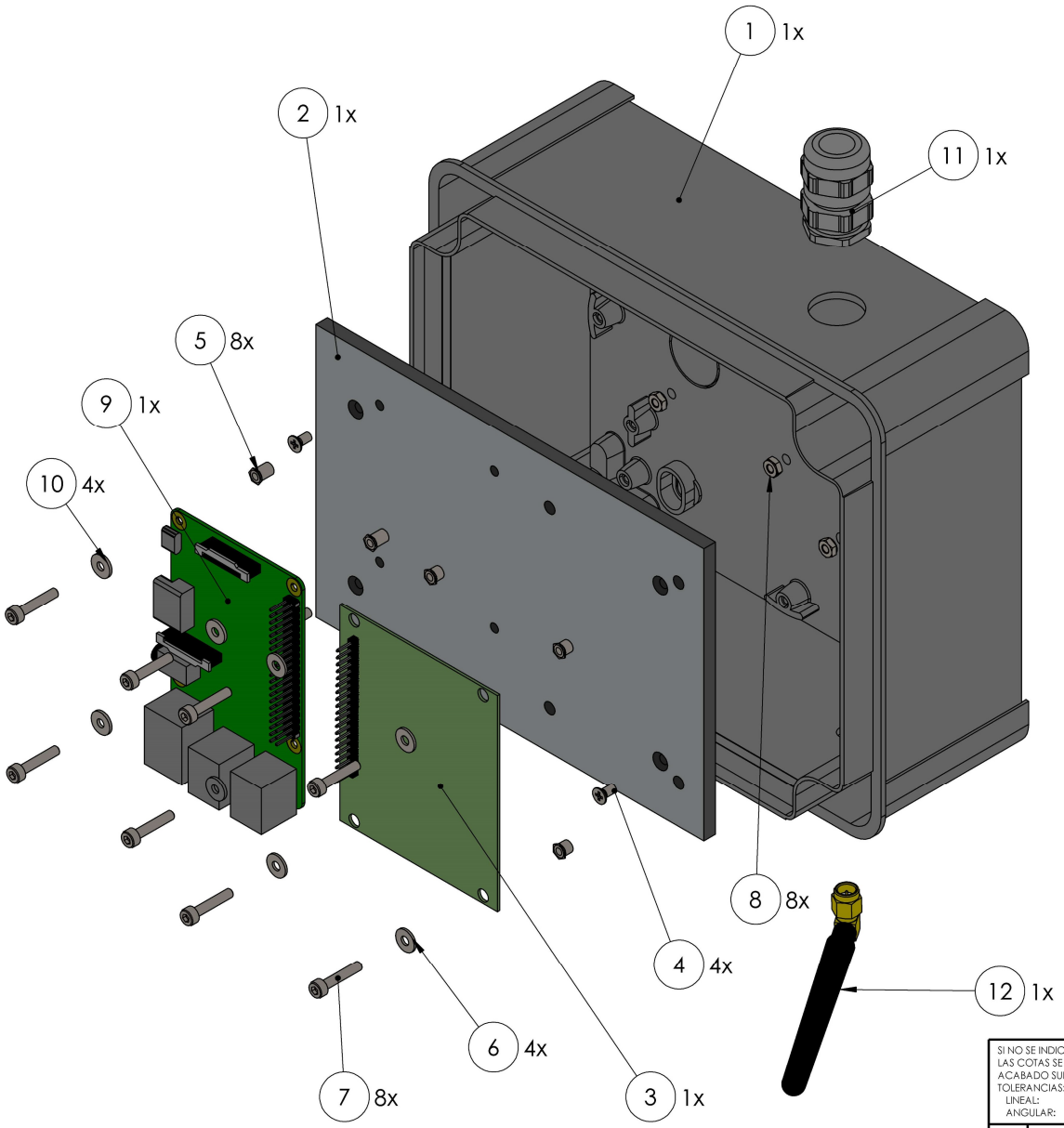
UPV

TÍTULO:

PLANO GENERAL

N.º DE DIBUJO: Gateway_Dimensiones | A3

ESCALA:1:2 | HOJA 1 DE 1



N.º DE ELEMENTO	N.º DE PIEZA	DESCRIPCIÓN	CANT.
1	ENV002	BOX_NSYTBS19168.STP	1
2	PL0001	SOPORTE	1
3	PCB0001	IC880A	1
4	TOR0010	ISO 7046-1 - M3 X 6 - Z - 6N	4
5	TOR0011	PEM SON-M3-4 ZI--N	8
6	TOR0012	DIN 6902-A3.2	4
7	TOR0013	DIN 912 M3 X 16 --- 16N	8
8	TOR0014	ISO 4032 - M3 - W - N	8
9	MP0001	RASPBERRY PI 3.STEP	1
10	TOR0015	10669 TYPE NL_ISO	4
11	VAR0001	KABELVERSCHRAUBUNG M16.STP	1
12	TR0002	ANTENNA 868МГЦ	1

SI NO SE INDICA LO CONTRARIO:
LAS COTAS SE EXPRESAN EN MM
ACABADO SUPERFICIAL:
TOLERANCIAS:
LINEAL:
ANGULAR:

ACABADO:

REBARBAR Y ROMPER ARISTAS VIVAS

	NOMBRE	FIRMA	FECHA
DIBUJ.	Alejandro Ortega		15/09/2019
VERIF.	Ángel Perles		
APROB.	Ángel Perles		12/10/2019
FABR.			
CALID.			

MATERIAL:

PESO:

ESCALA: 2:3

REVISIÓN

UPV

TÍTULO:

PLANO EXPLOSIÓN

N.º DE DIBUJO

Gateway

A3

ESCALA:1:2

HOJA 1 DE 1

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Desarrollo de una pasarela y nodos LoRaWAN

como solución de bajo coste para implantaciones IoT.

3. PLIEGO DE CONDICIONES

Autor:

D. Alejandro Ortega Pérez

Tutor:

D. Ángel Perles Ivars

Valencia, diciembre de 2019

Alejandro Ortega Pérez
Trabajo fin de grado

Pliego de condiciones

Contenido del Pliego

Tablas

1. Objeto

El actual documento recoge las condiciones de fabricación e instalación de una solución completa, mediante la cual se deberán transmitir datos mediante nodos sensores a través de una pasarela LoRaWAN para su posterior publicación en una web a la que tendrán acceso exclusivo los clientes del servicio.

Son objeto de este pliego, tanto la pasarela como los nodos sensores necesarios para la solución adoptada. Aunque el presente pliego recoge las condiciones reales de una instalación completa, el principal interés de este trabajo es mostrar los beneficios y las posibilidades que aporta esta solución a las crecientes iniciativas IoT que existen en la actualidad. Algunos puntos han sido simplificados u omitidos con el fin de no distraerse del principal objetivo del trabajo.

2. Condiciones Generales

2.1 Normativa aplicable

REAL DECRETO 842/2002 Por el que se regula cualquier instalación eléctrica.

Real Decreto 186/2016 por el que se regula la compatibilidad electromagnética de equipos eléctricos y electrónicos

RD 208/2005 mediante el cual se establece la correcta gestión de los residuos derivados de aparatos eléctricos y electrónicos

IET/787/2013 Por el que se regula el cuadro nacional de atribución de frecuencias

RoHS establecida en la directiva 2011/65/UE. Por la que se regulan las sustancias peligrosas presentes en las instalaciones eléctricas y electrónicas.

3. Condiciones Técnicas

3.1 Condiciones de los materiales

En los siguientes puntos se detallan las condiciones particulares de los materiales empleados en el desarrollo a un nivel básico, para mayor detalle de cualquiera de ellos acudir al anexo 2 del presente documento.

3.1.1 Concentrador LoRa IC880A

Trasmisor y receptor multicanal diseñado para recibir múltiples paquetes LoRa en distintos factores de dispersión. Para su correcto funcionamiento requiere de conexión a un sistema de control mediante interfaz SPI.

Es importante remarcar que no se deberá nunca alimentar la placa si no existe una antena conectada. El exceso de energía generado no podrá ser disipado y dañará de manera irreparable la placa.

El componente es distribuido en Europa a través del fabricante IMST. Los controladores se encuentran bajo licencia Open Source y puede ser recuperados directamente de la plataforma Github.

Cumple con todas las directivas y regulaciones europeas para dispositivos de su clase, entre ellas aquellas que son de aplicación para este desarrollo.

3.1.2 Microprocesador Raspberry Pi 3B+

Es el microcontrolador escogido para nuestro desarrollo. Es uno de los microprocesadores más potentes y versátiles del mercado para desarrollos independientes no industriales. Entre sus muchas características cuenta con soporte para periféricos mediante SPI, requisito mínimo indispensable para la comunicación con el concentrador LoRa, así como conector ethernet.

Raspberry está desarrollado por la Raspberry foundation con sede en reino unido, por ello cuenta con el cumplimento de las directivas europeas.

3.1.3 Alimentador POE29U-1AT (D)

Utilizado para acometer con un único cable a la envolvente mejorando las condiciones de estanqueidad. Este dispositivo cumple con los estándares IEEE802.3 siendo a su vez capaz de proporcionar una alimentación de hasta 56V con una carga máxima de 536mA a una tensión nominal de entrada de 100 a 240 VAC.

Este dispositivo cuenta con marcado CE, cumpliendo así con las regulaciones mencionadas en este documento.

3.1.4 Cables H05Z1-K

En cumplimiento con la normativa actual de emisiones y RoHs para la instalación de cableado en locales públicos, se ha optado por este tipo de cables para el desarrollo actual.

Entre sus características destacan ser libres de halógenos, no propagadores de llamas, con aislamiento flexible y temperaturas de conducción de hasta 70º.

Aunque no es demasiado el uso de cableado en este proyecto, es importante cumplir la normativa vigente y usar los materiales más adecuados para la protección de las personas y el medio.

3.1.5 Sensores

La mayoría de los sensores usados en este proyecto son importados, pero cumplen con todas las regulaciones vigentes, contando todos ellos con marcado CE.

El uso de cualquier nuevo modelo o fabricante de sensor debe ser analizada y estudiado para asegurar el cumplimento de la normativa y los máximos estándares de calidad. El mercado de sensores digitales es muy amplio, y en el existen soluciones de muy dudosa calidad que deben ser descartados de inmediato.

3.1.7 Microcontrolador LoPy 4.0

Tarjeta fabricada y distribuida por la iniciativa colaborativa PYCOM. En torno a su desarrollo existe una comunidad que brindará soporte y asistencia en todo momento.

La tarjeta está construida para comunicar en 4 protocolos diferentes, WiFi, Bluetooth, Sigfox y LoRa.

La tarjeta cuenta con marcado CE y cumplen todos los requisitos de certificación y normativas que exige nuestro proyecto.

Su manipulación como componente sensible a la estática debe ser siempre adecuado y siguiendo las indicaciones del fabricante.

Es importante no hacer uso de alguno de los protocolos de comunicación sin haber conectado antes una antena adecuada, el exceso de energía no disipada puede dañar seriamente el dispositivo.

Las condiciones de funcionamiento y mas datos sobre el dispositivo pueden ser consultados en el datasheet anexo a este documento.

3.1.9 Baterías

Las baterías utilizadas serán de polímero de litio, debiendo cumplir en todo momento con las normativas vigentes tanto como para su transporte como para su mantenimiento y desechado.

La capacidad de cada batería será calculada en función del tiempo de uso y el consumo del nodo destinado a alimentar. En cualquier caso, la vida útil de la batería no debe ser inferior a 6 meses. El cambio de baterías, o carga, será realizada siempre por técnicos cualificados. El sistema está desarrollado para auto reiniciarse una vez reestablecida la alimentación, no obstante, ningún usuario no conocedor del equipo debería manipularlo para asegurar el correcto funcionamiento futuro del equipo.

3.2 Condiciones de fabricación

La fabricación de nuestros dispositivos es sencilla y manual. No se hace necesarios equipos especiales ni unos costes de producción elevados.

En todo caso se deberá respetar la normativa de seguridad e higiene vigente en el momento de la fabricación, sin importar los costes o las perdidas derivadas de su aplicación.

El proceso de fabricación se divide en 4 grupos principales: interconexión, montaje y programación.

1. Interconexión:

Durante esta fase se preparan las tarjetas y se cablean, las siguientes pautas serán seguidas durante el proceso:

- Uso de guantes electroestáticos en la manipulación de componentes electrónicos
- Conexión de descarga electroestática a tierra durante la manipulación de componentes electrónicos
- Comprobación de continuidad punto a punto en las uniones cableadas.
- Trabajos siempre sin alimentación eléctrica.
- Temperatura y humedad controladas en el proceso.

El proceso de interconexión del Gateway y la mayoría de los nodos es similar y solo requiere de cableado y disposición correcta de componentes, se seguirá en todo

momento las pautas marcadas anteriormente y se ofrecerán los máximos estándares de calidad posibles.

El proceso de montaje del nodo del parque infantil requiere de soldadura para acceder a los puntos de conexión requeridos para la fabricación.

La soldadura se realizará en estaño adecuado a la normativa vigente y cumpliendo siempre las pautas estrictas del fabricante. El proceso de soldadura no debe durar más de 5 segundos en ningún caso, y la calidad se da con una soldadura en la que el contorno sea visible, con una superficie suave y brillante, sin áreas porosas. En la tabla 1 se puede observar las temperaturas adecuadas de soldado.

Stage	Duration/Rate	Temperature
Ramp to soak	2°C/s	Ambient – 185°C
Soak	60s	185°C
Ramp to peak	1°C/s	240°C
Reflow	45s	>225°C
Cool down	2°C/s	

TABLA 1 TEMPERATURA DE SOLDADURA Y TIEMPOS DE SOLDADO.

2. Programación

La parte principal de nuestro desarrollo es el software. Sin él nuestro dispositivo no tiene funcionalidad alguna.

El software de la pasarela es único para todos los dispositivos y no requiere de un desarrollo particular por cada solución, solo es necesario cargarlo en la memoria del controlador y unos pequeños ajustes para indicar posición e identidad de nuestro Gateway.

En nuestros dispositivos el software se carga sobre una Raspberry Pi sobre la capa del sistema Raspbian en su versión ligera. El controlador de la tarjeta LoRa está desarrollado en código abierto y puede ser recuperado de la plataforma Github, con los ajustes necesarios el código está listo para funcionar en nuestros dispositivos.

Los nodos hacen usos de softwares adaptados a cada solución concreta. Por lo general el software es totalmente reutilizable y solo requiere unas pequeñas variaciones entre

modelos de Nodo. De esta manera hacemos más sencillo tanto el mantenimiento que la fabricación de nuevas unidades.

El proceso de programación de nuestras tarjetas se hará siempre por personal cualificado y que corroborará el correcto de funcionamiento de cada dispositivo.

3. Montaje

El proceso de montaje se basa en la disposición final de componentes en la envolvente mediante uniones mecánicas.

La tornillería para usar será siempre en cumplimiento de la normativa vigente, no se utilizarán tratamientos considerados contaminantes en ningún caso. Así mismo, al ubicarse en una zona de alta humedad y en relativa cercanía al mar, se utilizará tornillería adecuada para alta corrosión del tipo A2-70

Para este proceso no se hará uso de ningún producto que requiera mención especial por peligrosidad o nocividad para el medioambiente.

Al final del proceso de montaje la máxima calidad es siempre exigible, se corregirá en todo momento cualquier defecto proveniente de la interconexión o generado en el proceso de montaje. El aspecto visual de nuestros productos es tan importante como el funcional, la calidad percibida es un pilar en cualquier dispositivo.

3.3 Condiciones de Instalación

Nuestros servicios incluyen desde la fabricación hasta la instalación en cliente. Nuestros expertos escogerán las ubicaciones ideales tanto para la pasarela como para cada nodo individual.

La fijación en los emplazamientos finales se realizará mediante elementos de unión mecánica que aseguren la durabilidad.

La pasarela se deberá ubicar siempre en lugares elevados, con el mayor ángulo de visión posible sobre los nodos a comunicar, lejos de la luz solar directa y de fuentes de calor como equipos acondicionados. Se hará un estudio adecuado de las bandas de frecuencia a usar garantizando la ausencia de interferencias que impidan las comunicaciones. El punto final de ubicación deberá contar con un punto de conexión ethernet a una distancia no muy elevada.

Los nodos se estudiarán particularmente dependiendo de su uso final. Como norma general se vitarán siempre incidencia de luz directa (salvo los nodos equipados con detector de niveles ultravioletas) y lejos de fuentes de calor extremos. En la medida de

lo posible se intentarán ubicar en lugares mas protegidos de la intemperie con el fin de alargar al máximo su vida útil.

Al tratarse de transmisión por frecuencia en la banda ISM no se hará necesario ningún tipo de licencia para su instalación.

3.4 Condiciones de entrega

Los equipos se entregan en perfecto funcionamiento y ubicados en su posición final ofreciendo servicio desde el primer momento.

El plazo de entrega desde la firma del contrato será de 4 semanas naturales a contar desde la firma de este. Se considerará el equipo entregado una vez el sistema esté operativo y haya superado todas las pruebas de entrega.

Si durante los plazos de entrega, el cliente por algún motivo no fuera accesible o no facilitará el acceso en los momentos solicitados para la instalación del producto, será responsabilidad exclusiva del cliente y asumirá cualquier retraso derivado de está causa.

Al ser un producto en desarrollo se ofrece un servicio de mantenimiento y actualizaciones de 2 años sin coste alguno para el cliente.

3.4 Prueba de servicio

Todos y cada uno de los dispositivos instalados deberá ser verificado y el funcionamiento general del sistema validado. A continuación, se definen las pruebas a realizar por los técnicos cualificados.

3.4.1 Pasarela

- Comprobar conexión a red de la pasarela.
- Realizar prueba intensiva de reenvío de paquetes durante 1 hora mínimo.
- Forzar varios reinicios de la red y asegurar la correcta recuperación del sistema tras las caídas.
- Generar mapa de cobertura mediante TTN map.
- Asegurar la no interferencia con otras señales de radiofrecuencia.

3.4.2 Nodos Sensores.

- Tomar medidas cada 5 segundos y contrastarlas con un equipo certificado para confirmar la correcta calibración de los sensores.
- Probar el envió correcto de paquetes.
- Asegurar la cobertura de red es óptima.
- Forzar un envío intensivo de datos durante al menos 1h.
- Asegurar la no existencia de interferencias magnéticas.

3.4.3 Sistema

- Confirmar la accesibilidad a la aplicación de TTN y a la web donde se alojará el panel de usuario.
- Confirmar la recepción de datos durante al menos 1h de manera intensiva.
- Simular caídas de red del sistema y del servidor para asegurar su correcta recuperación.
- Medir la latencia entre la recepción y el envío de paquetes entre pasarela y nodo.
- Asegurar la no perdida de ningún paquete durante el periodo de prueba.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Desarrollo de una pasarela y nodos LoRaWAN como solución de bajo coste para implantaciones IoT

4. PRESUPUESTO

Autor:
D. Alejandro Ortega Pérez
Tutor:
D. Ángel Perles Ivars

Valencia, diciembre de 2019

Presupuesto

Contenido del presupuesto

Glosario de términos

EA	Each (por unidad)
IVA	Impuesto sobre el Valor Añadido
PVP	Precio Venta Público
UNS	Unidades
UOM	*Unit of measure* (unidad de medida)

Presupuesto

1. Sobre el presupuesto

El presente presupuesto se ha realizado con el fin de demostrar el reducido coste de nuestra implementación, y como una parte más del desarrollo del presente trabajo. Es por ello que las consideraciones tomadas para su elaboración tienen en cuenta la implementación de la solución real con varios nodos, y no solo el desarrollado a modo de prototipo en los presentes documentos.

El presupuesto se encuentra dividido en varias partes con el fin de aportar información suficiente sobre los costes reales de una implementación completa de la solución, así como su escalabilidad a proyectos más ambiciosos. De esta manera las partidas quedan separadas por coste del Gateway, coste de los nodos y mano de obra.

Con el fin de demostrar el bajo coste de esta solución de la manera más real posible, se incluye el coste de desarrollo del software como un apartado independiente sin repercutir el precio al cliente, ya que es reutilizable en su totalidad. El coste de desarrollo se cobrará en forma de royalties bajo el concepto firmware.

Para los costes de materiales, al ser productos de un mercado cambiante, el presupuesto puede ser modificado sin necesidad de aviso previo en lo que respecta a su coste y modelo.

En todo caso el presente presupuesto no es vinculante sin ir adjunto a la memoria de desarrollo. Cualquier parte no contemplada en ninguno de los documentos debe ser negociada por separado.

El alquiler de un servidor dedicado para alojar la aplicación final no está incluido en el presupuesto. Su elección y contratación serán a través del cliente final de acuerdo con sus necesidades. Nuestro equipo dará soporte en esta tarea con el fin de encontrar la solución que más se adapte a nuestro cliente.

2. Materiales

2.1. Nodos

2.1.1. Parque infantil

Referencia	Cantidad	Description	Un	Fabricante	Ref Fabricante	Precio	Total
MC0001	1	MicroPython-enabled development board	EA	Pycom	LoPy 4.0	25,36 €	**25,36 €**
EB0001	1	Expansion Sheild Multi-Sensor	EA	Pycom	Pysense	15,56 €	**15,56 €**
ES0001	1	Laser dust sensor	EA	TELAIRE	SM-UART-04L	8,43 €	**8,43 €**
ES0002	1	UVA UVB and UV Index Sensor Breakout	EA	Adafruit	VEML6075	5,36 €	**5,36 €**
ENV0002	1	Envolvente externa	EA	Schneider	NSYTBP1397	12,50 €	**12,50 €**
CH0001	1	Chapa de montaje para nodo	EA	Varios		1,00 €	**1,00 €**
TOR001	2	DIN 7991 M3 L6	EA	Varios		0,05 €	**0,10 €**
TOR002	2	DIN 912 M3 L6	EA	Varios		0,05 €	**0,10 €**
FW001	1	Firmware para nodo parque infantil	EA	UPV	FW001-PI	20,00 €	**20,00 €**
CBL001	0,5	Cable violeta 0,5 mm	m	LAPP	H05Z1-K	0,12 €	**0,06 €**
CBL002	0,1	Cable blanco 0,5 mm	m	LAPP	H05Z1-K	0,12 €	**0,01 €**
CBL003	0,1	Cable rojo 0,5mm	m	LAPP	H05Z1-K	0,12 €	**0,01 €**
BAT001	1	Bateria Lipo 2000 mAh	EA	MIKROE	MIKROE-1120	12,25 €	**12,25 €**
TR0001	1	Antena RF SMA 868MHz 3,5dBi	EA	Siretta Ltd	DELTA1A/X/SMAM/S/S/11	7,00 €	**7,00 €**
Total							**107,74 €**

Tabla 1 Desglose materiales Parque Infantil

Presupuesto

2.1.2. Garaje

Referencia	Cantidad	Description	Un	Fabricante	Ref Fabricante	Precio	Total
MC0001	1	MicroPython-enabled development board	EA	Pycom	LoPy 4.0	25,360 €	**25,36 €**
ES0005	1	Sensor gases inflamables	EA	Adafruit	MQ2	1,250 €	**1,25 €**
ES0006	1	Sensor CO	EA	Adafruit	MQ7	1,500 €	**1,50 €**
ES0007	1	Sensor de calidad del aire	EA	Adafruit	MQ135	1,350 €	**1,35 €**
ENV0001	1	Envolvente interna para nodo	EA	Schneider	NSYTBS775	10,35 €	**10,35 €**
FW001	1	Firmware	EA		FW001-G	20,000 €	**20,00 €**
CBL001	1	Cable violeta 0,25 mm	m	LAPP	H05Z1-K	0,12 €	**0,12 €**
CBL002	0,1	Cable blanco 0,5 mm	m	LAPP	H05Z1-K	0,12 €	**0,01 €**
CBL003	0,1	Cable rojo 0,5mm	m	LAPP	H05Z1-K	0,12 €	**0,01 €**
BAT001	1	Bateria Lipo 2000 mAh	EA	MIKROE	MIKROE-1120	12,250 €	**12,25 €**
TR0001	1	Antena RF SMA 868MHz 3,5dBi	EA	Siretta Ltd	DELTA1A/X/SMAM/S/S/11	7,000 €	**7,00 €**
Total							**79,20 €**

Tabla 2 Desglose materiales Garaje

2.1.3. Piscina

Referencia	Cantidad	Description	Un	Fabricante	Ref Fabricante	Precio	Total
MC0001	1	MicroPython-enabled development board	EA	Pycom	LoPy 4.0	25,36 €	**25,36 €**
ES0003	1	Sensor Temperautra sumergible	EA	SONOFF	DS18B20	2,83 €	**2,83 €**
ES0004	1	Sensor de PH	EA	DM	PH-4502C	10,15 €	**10,15 €**
ENV0001	1	Envolvente interna para nodo	EA	Schneider	NSYTBS775	10,35 €	**10,35 €**
FW001	1	Firmware	EA	UPV	FW001-P	20,00 €	**20,00 €**
CBL001	0,5	Cable violeta 0,25 mm	m	LAPP	H05Z1-K	0,12 €	**0,06 €**
CBL002	0,1	Cable blanco 0,5 mm	m	LAPP	H05Z1-K	0,12 €	**0,01 €**
CBL003	0,1	Cable rojo 0,5mm	m	LAPP	H05Z1-K	0,12 €	**0,01 €**
BAT001	1	Bateria Lipo 2000 mAh	EA	MIKROE	MIKROE-1120	12,25 €	**12,25 €**
TR0001	1	Antena RF SMA 868MHz 3,5dBi	EA	Siretta Ltd	DELTA1A/X/SMAM/S/S/11	7,00 €	**7,00 €**
Total							**88,02 €**

Tabla 3Desglose materiales Piscina

2.2. Gateway

En este caso se han incluido exclusivamente los gastos asociados a la implantación y puesta en marcha del dispositivo para una única comunidad de vecinos y 3 nodos asociados. No obstante, al ser un servicio soportado por la plataforma TTN y la tecnología LoRaWAN, no existe exclusividad en el uso de este Gateway y puede ser aprovechado por cualquier entidad dentro del área de alcance. Esta partida se incluye al no existir cobertura en el área de implantación.

Referencia	Cantidad	Description	Un	Fabricante	Ref Fabricante	Precio	Total
PCB0001	1	Concentrador LoRa	EA	IMST	IC880A	115,00 €	**115,00 €**
MP0001	1	Microcontrolador	EA	Raspberry	Pi 3B	25,00 €	**25,00 €**
ENV0002	1	Envolvente	EA	Schneider	NSYTBP1176	16,05 €	**16,05 €**
PL0001	1	Chapa de montaje	EA	Varios		8,00 €	**8,00 €**
PS0001	1	Alimentador sobre ethernet	EA	Phihong	POE29U-1AT(PL)D	25,00 €	**25,00 €**
TOR0014	8	IS0 4032 M3 Tuerca Hexagonal	EA	Varios		0,02 €	**0,16 €**
TOR0013	8	DIN 912 M3 L16 Tornillo Hexagonal	EA	Varios		0,03 €	**0,24 €**
TOR0012	4	DIN 6902 A3.2 Arandela Plana	EA	Varios		0,01 €	**0,04 €**
TOR0011	8	PEM SON-M3-W-N	EA	Varios		0,10 €	**0,80 €**
TOR0010	4	ISO 7046-1 -M3 L6	EA	Varios		0,02 €	**0,08 €**
CBL001	0,5	Cable violeta 0,25 mm	m	LAPP	H05Z1-K	0,12 €	**0,06 €**
CBL002	0,1	Cable blanco 0,5 mm	m	LAPP	H05Z1-K	0,08 €	**0,01 €**
CBL003	0,1	Cable rojo 0,5mm	m	LAPP	H05Z1-K	0,08 €	**0,01 €**
CBL004	10	Cable Ethernet	m	LAPP	H05Z1-K	0,08 €	**0,80 €**
TR0002	1	Antena largo alcance	EA	Linx	ANT-868-ID	15,24 €	**15,24 €**
FW002	1	Firmware	EA	UPV	FW002	50,000 €	**50,00 €**
Total							**256,49 €**

Tabla 4 Desglose materiales Gateway

Presupuesto

3. Mano de obra

Para la mano de obra se ha tenido en cuenta la instalación de una solución completa como se ha mencionado, el coste de horas de diseño, tanto de firmware como electrónico, no serán considerados, tal y como se indica al inicio del presente documento.

Los costes se dividen según la actividad realizada en función de la cualificación requerida para su desarrollo. Los precios de la hora de mano de obra se han obtenido de las tablas oficiales del IVE.

Unidad	Description	Parcial	Cantidad	Precio	Total
h	MOOE12a Peón		1,5	14,71 €	22,07 €
	1 - Montaje Gateway en ubicación final	0,5			
	2 - Montaje Nodo Parque en ubicación final	0,5			
	3 - Montaje Nodo Garaje en ubicación final	0,5			
	4- Montaje Nodo Piscina en ubicación final	0,5			
	MOOE.8a Oficial 1ª		2,5	18,49 €	20,99 €
	1 - Ensamblaje y carga de firmware Gateway	1			
	2 - Ensamblaje y carga de firmware Nodo Parque	0,5			
	3 - Ensamblaje y carga de firmware Nodo Garaje	0,5			
	4 - Ensamblaje y carga de firmware Piscina	0,5			
h	MOOE.8a Ingeniero Técnico		17	22,57 €	383,69 €
	1 - Redacción de planos y presupuestos	10			
	2 - Supervisión de la instalación	5			
	3 - Puesta en marcha	2			
Total					**426,75 €**

Tabla 5 - Desglose presupuesto mano de obra

4. Desarrollo del Software

Tal y como se indica en el resumen del presupuesto, este desglose se incluye a modo informativo para mostrar el coste asociado del desarrollo del proyecto. No obstante, no se incluirá en la partida con el fin de demostrar el coste reducido que tiene esta solución para el cliente final.

Unidad	Description	Parcial	Cantidad	Precio	Total
h	MOOE.8a Ingeniero Técnico		49	22,57 €	1.105,93 €
	1 - Preparación de entorno de desarrollo	2			
	2 - Modificaciones en codigo fuente de la pasarela y raspbian	5			
	3 - Programación en TTN	3			
	4 - Creación de librerias Nodo	10			
	5 - Creación del codigo fuente de la aplicación final	15			
	6 - Preparación del servidor AWS	4			
	7 - Programación de Panel web en Node-Red	10			
Total					**1.105,93 €**

Unidad	Descripción	Coste Equipo	Vida Util	Cantidad	Precio	Total
h	Ordenador Personal	599,00 €	15000	60	0,04 €	2,40 €
h	LoPy 4	25,36 €	10000	10	0,003 €	0,03 €
h	Raspberry Pi 3B	25,00 €	10000	10	0,003 €	0,03 €
h	Pantalla BenQ	149,00 €	9000	80	0,02 €	1,32 €
Total					**3,77 €**	**3,77 €**

Presupuesto

5. Resumen

5.1 Resumen sin gastos de desarrollo

A continuación, se resumen los costes totales de la implantación de la solución completa.

Sección	Description	Parcial	Total
2	Materiales		**531,46 €**
2.1	Nodo Parque infantil	79,20 €	
2.2	Nodo Piscina	88,02 €	
2.3	Nodo Garaje	107,74 €	
2.4	Gateway	256,49 €	
3	Mano de obra		**426,75 €**
Total			**958,20 €**

IVA	21%	201,22 €
Gastos adicionales	10%	95,82 €
TOTAL COSTE PROYECTO		**1.255,25 €**

Tabla 6 - Resumen gastos totales impuestos y gastos adicionales incluidos

La implantación tiene un coste total asociado de MIL DOSCIENTOS CINCUENTA Y CINCO EUROS CON VEINTICINCO CENTIMOS

5.2 Resumen con gastos de desarrollo

Sección	Description	Parcial	Total
2	Materiales		**531,46 €**
2.1	Nodo Parque infantil	79,20 €	
2.2	Nodo Piscina	88,02 €	
2.3	Nodo Garaje	107,74 €	
2.4	Gateway	256,49 €	
3	Mano de obra		**426,75 €**
4	Desarrollo		**1109,70 €**
Total			**2067,90 €**

IVA	21%	434,26 €
Gastos adicionales	10%	206,79 €
TOTAL COSTE PROYECTO		**2708,95 €**

Anexo 2. Hojas de características

Desarrollo de una pasarela y nodos LoRaWAN como solución de bajo coste para implantaciones IoT

5. ANEXOS

Autor:

D. Alejandro Ortega Pérez

Tutor:

D. Ángel Perles Ivars

Valencia, diciembre de 2019

Anexo 1. Código

Contenido Anexo 1. Código

A1.1 Librerías

1.1 LIS2HH12.py

```
import math
import time
import struct
from machine import Pin


FULL_SCALE_2G = const(0)
FULL_SCALE_4G = const(2)
FULL_SCALE_8G = const(3)

ODR_POWER_DOWN = const(0)
ODR_10_HZ = const(1)
ODR_50_HZ = const(2)
ODR_100_HZ = const(3)
ODR_200_HZ = const(4)
ODR_400_HZ = const(5)
ODR_800_HZ = const(6)

ACC_G_DIV = 1000 * 65536


class LIS2HH12:

    ACC_I2CADDR = const(30)

    PRODUCTID_REG = const(0x0F)
    CTRL1_REG = const(0x20)
    CTRL2_REG = const(0x21)
    CTRL3_REG = const(0x22)
    CTRL4_REG = const(0x23)
    CTRL5_REG = const(0x24)
    ACC_X_L_REG = const(0x28)
    ACC_X_H_REG = const(0x29)
    ACC_Y_L_REG = const(0x2A)
    ACC_Y_H_REG = const(0x2B)
    ACC_Z_L_REG = const(0x2C)
    ACC_Z_H_REG = const(0x2D)
    ACT_THS = const(0x1E)
    ACT_DUR = const(0x1F)

    SCALES = {FULL_SCALE_2G: 4000, FULL_SCALE_4G: 8000, FULL_SCALE_8G: 16000}
    ODRS = [0, 10, 50, 100, 200, 400, 800]

    def __init__(self, pysense = None, sda = 'P22', scl = 'P21'):
        if pysense is not None:
            self.i2c = pysense.i2c
        else:
            from machine import I2C
            self.i2c = I2C(0, mode=I2C.MASTER, pins=(sda, scl))

        self.odr = 0
        self.full_scale = 0
        self.x = 0
        self.y = 0
        self.z = 0
        self.int_pin = None
        self.act_dur = 0
        self.debounced = False

        whoami = self.i2c.readfrom_mem(ACC_I2CADDR , PRODUCTID_REG, 1)
        if (whoami[0] != 0x41):
            raise ValueError("LIS2HH12 not found")
```

Anexo 1. Código

```
   63
   64         # enable acceleration readings at 50Hz
   65         self.set_odr(ODR_50_HZ)
   66
   67         # change the full-scale to 4g
   68         self.set_full_scale(FULL_SCALE_4G)
   69
   70         # set the interrupt pin as active low and open drain
   71         self.set_register(CTRL5_REG, 3, 0, 3)
   72
   73         # make a first read
   74         self.acceleration()
   75
   76     def acceleration(self):
   77         x = self.i2c.readfrom_mem(ACC_I2CADDR , ACC_X_L_REG, 2)
   78         self.x = struct.unpack('<h', x)
   79         y = self.i2c.readfrom_mem(ACC_I2CADDR , ACC_Y_L_REG, 2)
   80         self.y = struct.unpack('<h', y)
   81         z = self.i2c.readfrom_mem(ACC_I2CADDR , ACC_Z_L_REG, 2)
   82         self.z = struct.unpack('<h', z)
   83         _mult = self.SCALES[self.full_scale] / ACC_G_DIV
   84         return (self.x[0] * _mult, self.y[0] * _mult, self.z[0] * _mult)
   85
   86     def roll(self):
   87         x,y,z = self.acceleration()
   88         rad = math.atan2(-x, z)
   89         return (180 / math.pi) * rad
   90
   91     def pitch(self):
   92         x,y,z = self.acceleration()
   93         rad = -math.atan2(y, (math.sqrt(x*x + z*z)))
   94         return (180 / math.pi) * rad
   95
   96     def set_register(self, register, value, offset, mask):
   97         reg = bytearray(self.i2c.readfrom_mem(ACC_I2CADDR, register, 1))
   98         reg[0] &= ~(mask << offset)
   99         reg[0] |= ((value & mask) << offset)
  100         self.i2c.writeto_mem(ACC_I2CADDR, register, reg)
  101
  102     def set_full_scale(self, scale):
  103         self.set_register(CTRL4_REG, scale, 4, 3)
  104         self.full_scale = scale
  105
  106     def set_odr(self, odr):
  107         self.set_register(CTRL1_REG, odr, 4, 7)
  108         self.odr = odr
  109
  110     def set_high_pass(self, hp):
  111         self.set_register(CTRL2_REG, 1 if hp else 0, 2, 1)
  112
  113     def enable_activity_interrupt(self, threshold, duration, handler=None):
  114         # Threshold is in mg, duration is ms
  115         self.act_dur = duration
  116
  117         if threshold > self.SCALES[self.full_scale]:
  118             error = "threshold %d exceeds full scale %d" % (threshold,
self.SCALES[self.full_scale])
  119             print(error)
  120             raise ValueError(error)
  121
  122         if threshold < self.SCALES[self.full_scale] / 128:
  123             error = "threshold %d below resolution %d" % (threshold,
self.SCALES[self.full scale]/128)
  124             print(error)
  125             raise ValueError(error)
  126
  127         if duration > 255 * 1000 * 8 / self.ODRS[self.odr]:
  128             error = "duration %d exceeds max possible value %d" % (duration, 255 * 1000 * 8 /
self.ODRS[self.odr])
  129             print(error)
  130             raise ValueError(error)
  131
```

```
  132         if duration < 1000 * 8 / self.ODRS[self.odr]:
  133             error = "duration %d below resolution %d" % (duration, 1000 * 8 /
self.ODRS[self.odr])
  134             print(error)
  135             raise ValueError(error)
  136
  137         _ths = int(127 * threshold / self.SCALES[self.full_scale]) & 0x7F
  138         _dur = int((duration * self.ODRS[self.odr]) / 1000 / 8)
  139
  140         self.i2c.writeto_mem(ACC_I2CADDR, ACT_THS, _ths)
  141         self.i2c.writeto_mem(ACC_I2CADDR, ACT_DUR, _dur)
  142
  143         # enable the activity/inactivity interrupt
  144         self.set_register(CTRL3_REG, 1, 5, 1)
  145
  146         self._user_handler = handler
  147         self.int_pin = Pin('P13', mode=Pin.IN)
  148         self.int_pin.callback(trigger=Pin.IRQ_FALLING | Pin.IRQ_RISING,
handler=self._int_handler)
  149
  150         # return actual used threshold and duration
  151         return (_ths * self.SCALES[self.full_scale] / 128, _dur * 8 * 1000 /
self.ODRS[self.odr])
  152
  153     def activity(self):
  154         if not self.debounced:
  155             time.sleep_ms(self.act_dur)
  156             self.debounced = True
  157         if self.int_pin():
  158             return True
  159         return False
  160
  161     def _int_handler(self, pin_o):
  162         if self. user handler is not None:
  163             self._user_handler(pin_o)
  164         else:
  165             if pin_o():
  166                 print('Activity interrupt')
  167             else:
  168                  print('Inactivity’)
```

Anexo 1. Código

1.1 LTR329ALS01.py

```
import time
from machine import I2C

class LTR329ALS01:
    ALS_I2CADDR = const(0x29) # The device's I2C address

    ALS_CONTR_REG = const(0x80)
    ALS_MEAS_RATE_REG = const(0x85)

    ALS_DATA_CH1_LOW = const(0x88)
    ALS_DATA_CH1_HIGH = const(0x89)
    ALS_DATA_CH0_LOW = const(0x8A)
    ALS_DATA_CH0_HIGH = const(0x8B)

    ALS_GAIN_1X = const(0x00)
    ALS_GAIN_2X = const(0x01)
    ALS_GAIN_4X = const(0x02)
    ALS_GAIN_8X = const(0x03)
    ALS_GAIN_48X = const(0x06)
    ALS_GAIN_96X = const(0x07)

    ALS_INT_50 = const(0x01)
    ALS_INT_100 = const(0x00)
    ALS_INT_150 = const(0x04)
    ALS_INT_200 = const(0x02)
    ALS_INT_250 = const(0x05)
    ALS_INT_300 = const(0x06)
    ALS_INT_350 = const(0x07)
    ALS_INT_400 = const(0x03)

    ALS_RATE_50 = const(0x00)
    ALS_RATE_100 = const(0x01)
    ALS_RATE_200 = const(0x02)
    ALS_RATE_500 = const(0x03)
    ALS_RATE_1000 = const(0x04)
    ALS_RATE_2000 = const(0x05)

    def __init__(self, pysense = None, sda = 'P22', scl = 'P21', gain = ALS_GAIN_1X,
integration = ALS_INT_100, rate = ALS_RATE_500):
        if pysense is not None:
            self.i2c = pysense.i2c
        else:
            self.i2c = I2C(0, mode=I2C.MASTER, pins=(sda, scl))

        contr = self._getContr(gain)
        self.i2c.writeto_mem(ALS_I2CADDR, ALS_CONTR_REG, bytearray([contr]))

        measrate = self._getMeasRate(integration, rate)
        self.i2c.writeto_mem(ALS_I2CADDR, ALS_MEAS_RATE_REG, bytearray([measrate]))

        time.sleep(0.01)

    def _getContr(self, gain):
        return ((gain & 0x07) << 2) + 0x01

    def _getMeasRate(self, integration, rate):
        return ((integration & 0x07) << 3) + (rate & 0x07)

    def _getWord(self, high, low):
        return ((high & 0xFF) << 8) + (low & 0xFF)

    def light(self):
        ch1low = self.i2c.readfrom_mem(ALS_I2CADDR , ALS_DATA_CH1_LOW, 1)
        ch1high = self.i2c.readfrom_mem(ALS_I2CADDR , ALS_DATA_CH1_HIGH, 1)
        data1 = int(self._getWord(ch1high[0], ch1low[0]))

        ch0low = self.i2c.readfrom_mem(ALS_I2CADDR , ALS_DATA_CH0_LOW, 1)
```

```
67          ch0high = self.i2c.readfrom_mem(ALS_I2CADDR , ALS_DATA_CH0_HIGH, 1)
68          data0 = int(self._getWord(ch0high[0], ch0low[0]))
69
70          return (data0, data1)
```

Anexo 1. Código

1.2 MPL3315A2.py

```
import time
from machine import I2C

ALTITUDE = const(0)
PRESSURE = const(1)

class MPL3115A2exception(Exception):
    pass

class MPL3115A2:
    MPL3115_I2CADDR = const(0x60)
    MPL3115_STATUS = const(0x00)
    MPL3115_PRESSURE_DATA_MSB = const(0x01)
    MPL3115_PRESSURE_DATA_CSB = const(0x02)
    MPL3115_PRESSURE_DATA_LSB = const(0x03)
    MPL3115_TEMP_DATA_MSB = const(0x04)
    MPL3115_TEMP_DATA_LSB = const(0x05)
    MPL3115_DR_STATUS = const(0x06)
    MPL3115_DELTA_DATA = const(0x07)
    MPL3115_WHO_AM_I = const(0x0c)
    MPL3115_FIFO_STATUS = const(0x0d)
    MPL3115_FIFO_DATA = const(0x0e)
    MPL3115_FIFO_SETUP = const(0x0e)
    MPL3115_TIME_DELAY = const(0x10)
    MPL3115_SYS_MODE = const(0x11)
    MPL3115_INT_SORCE = const(0x12)
    MPL3115_PT_DATA_CFG = const(0x13)
    MPL3115_BAR_IN_MSB = const(0x14)
    MPL3115_P_ARLARM_MSB = const(0x16)
    MPL3115_T_ARLARM = const(0x18)
    MPL3115_P_ARLARM_WND_MSB = const(0x19)
    MPL3115_T_ARLARM_WND = const(0x1b)
    MPL3115_P_MIN_DATA = const(0x1c)
    MPL3115_T_MIN_DATA = const(0x1f)
    MPL3115_P_MAX_DATA = const(0x21)
    MPL3115_T_MAX_DATA = const(0x24)
    MPL3115_CTRL_REG1 = const(0x26)
    MPL3115_CTRL_REG2 = const(0x27)
    MPL3115_CTRL_REG3 = const(0x28)
    MPL3115_CTRL_REG4 = const(0x29)
    MPL3115_CTRL_REG5 = const(0x2a)
    MPL3115_OFFSET_P = const(0x2b)
    MPL3115_OFFSET_T = const(0x2c)
    MPL3115_OFFSET_H = const(0x2d)

    def __init__(self, pysense = None, sda = 'P22', scl = 'P21', mode = PRESSURE):
        if pysense is not None:
            self.i2c = pysense.i2c
        else:
            self.i2c = I2C(0, mode=I2C.MASTER, pins=(sda, scl))

        self.STA_reg = bytearray(1)
        self.mode = mode

        if self.mode is PRESSURE:
            self.i2c.writeto_mem(MPL3115_I2CADDR, MPL3115_CTRL_REG1, bytes([0x38])) # barometer mode, not raw, oversampling 128, minimum time 512 ms
            self.i2c.writeto_mem(MPL3115_I2CADDR, MPL3115_PT_DATA_CFG, bytes([0x07])) # no events detected
            self.i2c.writeto_mem(MPL3115_I2CADDR, MPL3115_CTRL_REG1, bytes([0x39])) # active
        elif self.mode is ALTITUDE:
            self.i2c.writeto_mem(MPL3115_I2CADDR, MPL3115_CTRL_REG1, bytes([0xB8])) # altitude mode, not raw, oversampling 128, minimum time 512 ms
            self.i2c.writeto_mem(MPL3115_I2CADDR, MPL3115_PT_DATA_CFG, bytes([0x07])) # no events detected
            self.i2c.writeto_mem(MPL3115_I2CADDR, MPL3115_CTRL_REG1, bytes([0xB9])) # active
        else:
```

```
   64             raise MPL3115A2exception("Invalid Mode MPL3115A2")
   65
   66         if self._read_status():
   67             pass
   68         else:
   69             raise MPL3115A2exception("Error with MPL3115A2")
   70
   71     def _read_status(self):
   72         while True:
   73             self.i2c.readfrom_mem_into(MPL3115_I2CADDR, MPL3115_STATUS, self.STA_reg)
   74
   75             if(self.STA_reg[0] == 0):
   76                 time.sleep(0.01)
   77                 pass
   78             elif(self.STA_reg[0] & 0x04) == 4:
   79                 return True
   80             else:
   81                 return False
   82
   83     def pressure(self):
   84         if self.mode == ALTITUDE:
   85             raise MPL3115A2exception("Incorrect Measurement Mode MPL3115A2")
   86
   87         OUT_P_MSB = self.i2c.readfrom_mem(MPL3115_I2CADDR, MPL3115_PRESSURE_DATA_MSB,1)
   88         OUT_P_CSB = self.i2c.readfrom_mem(MPL3115_I2CADDR, MPL3115_PRESSURE_DATA_CSB,1)
   89         OUT_P_LSB = self.i2c.readfrom_mem(MPL3115_I2CADDR, MPL3115_PRESSURE_DATA_LSB,1)
   90
   91         return float((OUT_P_MSB[0] << 10) + (OUT_P_CSB[0] << 2) + ((OUT_P_LSB[0] >> 6) & 0x03)
+ ((OUT_P_LSB[0] >> 4) & 0x03) / 4.0)
   92
   93     def altitude(self):
   94         if self.mode == PRESSURE:
   95             raise MPL3115A2exception("Incorrect Measurement Mode MPL3115A2")
   96
   97         OUT_P_MSB = self.i2c.readfrom_mem(MPL3115_I2CADDR, MPL3115_PRESSURE_DATA_MSB,1)
   98         OUT_P_CSB = self.i2c.readfrom_mem(MPL3115_I2CADDR, MPL3115_PRESSURE_DATA_CSB,1)
   99         OUT_P_LSB = self.i2c.readfrom_mem(MPL3115_I2CADDR, MPL3115_PRESSURE_DATA_LSB,1)
  100
  101         alt_int = (OUT_P_MSB[0] << 8) + (OUT_P_CSB[0])
  102         alt_frac = ((OUT_P_LSB[0] >> 4) & 0x0F)
  103
  104         if alt_int > 32767:
  105             alt_int -= 65536
  106
  107         return float(alt_int + alt_frac / 16.0)
  108
  109     def temperature(self):
  110         OUT_T_MSB = self.i2c.readfrom_mem(MPL3115_I2CADDR, MPL3115_TEMP_DATA_MSB,1)
  111         OUT_T_LSB = self.i2c.readfrom_mem(MPL3115_I2CADDR, MPL3115_TEMP_DATA_LSB,1)
  112
  113         temp_int = OUT_T_MSB[0]
  114         temp_frac = OUT_T_LSB[0]
  115
  116         if temp_int > 127:
  117             temp_int -= 256
  118
  119         return float(temp_int + temp_frac / 256.0)
```

Anexo 1. Código

1.3 pycoproc.py

```
from machine import Pin
from machine import I2C
import time
import pycom

__version__ = '0.0.2'

""" PIC MCU wakeup reason types """
WAKE_REASON_ACCELEROMETER = 1
WAKE_REASON_PUSH_BUTTON = 2
WAKE_REASON_TIMER = 4
WAKE_REASON_INT_PIN = 8

class Pycoproc:
    """ class for handling the interaction with PIC MCU """

    I2C_SLAVE_ADDR = const(8)

    CMD_PEEK = const(0x0)
    CMD_POKE = const(0x01)
    CMD_MAGIC = const(0x02)
    CMD_HW_VER = const(0x10)
    CMD_FW_VER = const(0x11)
    CMD_PROD_ID = const(0x12)
    CMD_SETUP_SLEEP = const(0x20)
    CMD_GO_SLEEP = const(0x21)
    CMD_CALIBRATE = const(0x22)
    CMD_BAUD_CHANGE = const(0x30)
    CMD_DFU = const(0x31)

    REG_CMD = const(0)
    REG_ADDRL = const(1)
    REG_ADDRH = const(2)
    REG_AND = const(3)
    REG_OR = const(4)
    REG_XOR = const(5)

    ANSELA_ADDR = const(0x18C)
    ANSELB_ADDR = const(0x18D)
    ANSELC_ADDR = const(0x18E)

    ADCON0_ADDR = const(0x9D)
    ADCON1_ADDR = const(0x9E)

    IOCAP_ADDR = const(0x391)
    IOCAN_ADDR = const(0x392)

    INTCON_ADDR = const(0x0B)
    OPTION_REG_ADDR = const(0x95)

    _ADCON0_CHS_POSN = const(0x02)
    _ADCON0_ADON_MASK = const(0x01)
    _ADCON1_ADCS_POSN = const(0x04)
    _ADCON0_GO_nDONE_MASK = const(0x02)

    ADRESL_ADDR = const(0x09B)
    ADRESH_ADDR = const(0x09C)

    TRISC_ADDR = const(0x08E)

    PORTA_ADDR = const(0x00C)
    PORTC_ADDR = const(0x00E)

    WPUA_ADDR = const(0x20C)

    WAKE_REASON_ADDR = const(0x064C)
    MEMORY_BANK_ADDR = const(0x0620)
```

```
 68
 69      PCON_ADDR = const(0x096)
 70      STATUS_ADDR = const(0x083)
 71
 72      EXP_RTC_PERIOD = const(7000)
 73
 74      def __init__(self, i2c=None, sda='P22', scl='P21'):
 75          if i2c is not None:
 76              self.i2c = i2c
 77          else:
 78              self.i2c = I2C(0, mode=I2C.MASTER, pins=(sda, scl))
 79
 80          self.sda = sda
 81          self.scl = scl
 82          self.clk_cal_factor = 1
 83          self.reg = bytearray(6)
 84          self.wake_int = False
 85          self.wake_int_pin = False
 86          self.wake_int_pin_rising_edge = True
 87
 88          # Make sure we are inserted into the
 89          # correct board and can talk to the PIC
 90          try:
 91              self.read_fw_version()
 92          except Exception as e:
 93              raise Exception('Board not detected: {}'.format(e))
 94
 95          # init the ADC for the battery measurements
 96          self.poke_memory(ANSELC_ADDR, 1 << 2)
 97          self.poke_memory(ADCON0_ADDR, (0x06 << _ADCON0_CHS_POSN) | _ADCON0_ADON_MASK)
 98          self.poke_memory(ADCON1_ADDR, (0x06 << _ADCON1_ADCS_POSN))
 99          # enable the pull-up on RA3
100          self.poke_memory(WPUA_ADDR, (1 << 3))
101          # make RC5 an input
102          self.set_bits_in_memory(TRISC_ADDR, 1 << 5)
103          # set RC6 and RC7 as outputs and enable power to the sensors and the GPS
104          self.mask_bits_in_memory(TRISC_ADDR, ~(1 << 6))
105          self.mask_bits_in_memory(TRISC_ADDR, ~(1 << 7))
106
107          if self.read_fw_version() < 6:
108              raise ValueError('Firmware out of date')
109
110
111      def _write(self, data, wait=True):
112          self.i2c.writeto(I2C_SLAVE_ADDR, data)
113          if wait:
114              self._wait()
115
116      def _read(self, size):
117          return self.i2c.readfrom(I2C_SLAVE_ADDR, size + 1)[1:(size + 1)]
118
119      def _wait(self):
120          count = 0
121          time.sleep_us(10)
122          while self.i2c.readfrom(I2C_SLAVE_ADDR, 1)[0] != 0xFF:
123              time.sleep_us(100)
124              count += 1
125              if (count > 500):  # timeout after 50ms
126                  raise Exception('Board timeout')
127
128      def _send_cmd(self, cmd):
129          self._write(bytes([cmd]))
130
131      def read_hw_version(self):
132          self._send_cmd(CMD_HW_VER)
133          d = self._read(2)
134          return (d[1] << 8) + d[0]
135
136      def read_fw_version(self):
```

Anexo 1. Código

```
  137         self._send_cmd(CMD_FW_VER)
  138         d = self. read(2)
  139         return (d[1] << 8) + d[0]
  140
  141     def read_product_id(self):
  142         self._send_cmd(CMD_PROD_ID)
  143         d = self._read(2)
  144         return (d[1] << 8) + d[0]
  145
  146     def peek_memory(self, addr):
  147         self._write(bytes([CMD_PEEK, addr & 0xFF, (addr >> 8) & 0xFF]))
  148         return self._read(1)[0]
  149
  150     def poke_memory(self, addr, value):
  151         self._write(bytes([CMD_POKE, addr & 0xFF, (addr >> 8) & 0xFF, value & 0xFF]))
  152
  153     def magic_write_read(self, addr, _and=0xFF, _or=0, _xor=0):
  154         self._write(bytes([CMD_MAGIC, addr & 0xFF, (addr >> 8) & 0xFF, _and & 0xFF, _or &
0xFF, _xor & 0xFF]))
  155         return self._read(1)[0]
  156
  157     def toggle_bits_in_memory(self, addr, bits):
  158         self.magic_write_read(addr, _xor=bits)
  159
  160     def mask_bits_in_memory(self, addr, mask):
  161         self.magic_write_read(addr, _and=mask)
  162
  163     def set_bits_in_memory(self, addr, bits):
  164         self.magic_write_read(addr, _or=bits)
  165
  166     def get_wake_reason(self):
  167         """ returns the wakeup reason, a value out of constants WAKE_REASON_* """
  168         return self.peek_memory(WAKE_REASON_ADDR)
  169
  170     def get_sleep_remaining(self):
  171         """ returns the remaining time from sleep, as an interrupt (wakeup source) might have
triggered """
  172         c3 = self.peek_memory(WAKE_REASON_ADDR + 3)
  173         c2 = self.peek_memory(WAKE_REASON_ADDR + 2)
  174         c1 = self.peek_memory(WAKE_REASON_ADDR + 1)
  175         time_device_s = (c3 << 16) + (c2 << 8) + c1
  176         # this time is from PIC internal oscilator, so it needs to be adjusted with the
calibration value
  177         try:
  178             self.calibrate_rtc()
  179         except Exception:
  180             pass
  181         time_s = int((time_device_s / self.clk_cal_factor) + 0.5) # 0.5 used for round
  182         return time_s
  183
  184     def setup_sleep(self, time_s):
  185         try:
  186             self.calibrate_rtc()
  187         except Exception:
  188             pass
  189         time_s = int((time_s * self.clk_cal_factor) + 0.5)  # round to the nearest integer
  190         if time_s >= 2**(8*3):
  191             time_s = 2**(8*3)-1
  192         self._write(bytes([CMD_SETUP_SLEEP, time_s & 0xFF, (time_s >> 8) & 0xFF, (time_s >>
16) & 0xFF]))
  193
  194     def go_to_sleep(self, gps=True):
  195         # enable or disable back-up power to the GPS receiver
  196         if gps:
  197             self.set_bits_in_memory(PORTC_ADDR, 1 << 7)
  198         else:
  199             self.mask_bits_in_memory(PORTC_ADDR, ~(1 << 7))
  200         # disable the ADC
  201         self.poke memory(ADCON0 ADDR, 0)
  202
  203         if self.wake_int:
  204             # Don't touch RA3, RA5 or RC1 so that interrupt wake-up works
```

```
  205             self.poke_memory(ANSELA_ADDR, ~((1 << 3) | (1 << 5)))
  206             self.poke_memory(ANSELC_ADDR, ~((1 << 6) | (1 << 7) | (1 << 1)))
  207         else:
  208             # disable power to the accelerometer, and don't touch RA3 so that button wake-up
works
  209             self.poke_memory(ANSELA_ADDR, ~(1 << 3))
  210             self.poke_memory(ANSELC_ADDR, ~(1 << 7))
  211
  212         self.poke_memory(ANSELB_ADDR, 0xFF)
  213
  214         # check if INT pin (PIC RC1), should be used for wakeup
  215         if self.wake_int_pin:
  216             if self.wake_int_pin_rising_edge:
  217                 self.set_bits_in_memory(OPTION_REG_ADDR, 1 << 6) # rising edge of INT pin
  218             else:
  219                 self.mask_bits_in_memory(OPTION_REG_ADDR, ~(1 << 6)) # falling edge of INT pin
  220             self.mask_bits_in_memory(ANSELC_ADDR, ~(1 << 1)) # disable analog function for RC1
pin
  221             self.set_bits_in_memory(TRISC_ADDR, 1 << 1) # make RC1 input pin
  222             self.mask_bits_in_memory(INTCON_ADDR, ~(1 << 1)) # clear INTF
  223             self.set_bits_in_memory(INTCON_ADDR, 1 << 4) # enable interrupt; set INTE)
  224
  225         self._write(bytes([CMD_GO_SLEEP]), wait=False)
  226         # kill the run pin
  227         Pin('P3', mode=Pin.OUT, value=0)
  228
  229     def calibrate_rtc(self):
  230         # the 1.024 factor is because the PIC LF operates at 31 KHz
  231         # WDT has a frequency divider to generate 1 ms
  232         # and then there is a binary prescaler, e.g., 1, 2, 4 ... 512, 1024 ms
  233         # hence the need for the constant
  234         self._write(bytes([CMD_CALIBRATE]), wait=False)
  235         self.i2c.deinit()
  236         Pin('P21', mode=Pin.IN)
  237         pulses = pycom.pulses_get('P21', 100)
  238         self.i2c.init(mode=I2C.MASTER, pins=(self.sda, self.scl))
  239         idx = 0
  240         for i in range(len(pulses)):
  241             if pulses[i][1] > EXP_RTC_PERIOD:
  242                 idx = i
  243                 break
  244         try:
  245             period = pulses[idx][1] - pulses[(idx - 1)][1]
  246         except:
  247             period = 0
  248         if period > 0:
  249             self.clk_cal_factor = (EXP_RTC_PERIOD / period) * (1000 / 1024)
  250         if self.clk_cal_factor > 1.25 or self.clk_cal_factor < 0.75:
  251             self.clk_cal_factor = 1
  252
  253     def button_pressed(self):
  254         button = self.peek_memory(PORTA_ADDR) & (1 << 3)
  255         return not button
  256
  257     def read_battery_voltage(self):
  258         self.set_bits_in_memory(ADCON0_ADDR, _ADCON0_GO_nDONE_MASK)
  259         time.sleep_us(50)
  260         while self.peek_memory(ADCON0_ADDR) & _ADCON0_GO_nDONE_MASK:
  261             time.sleep_us(100)
  262         adc_val = (self.peek_memory(ADRESH_ADDR) << 2) + (self.peek_memory(ADRESL_ADDR) >> 6)
  263         return (((adc_val * 3.3 * 280) / 1023) / 180) + 0.01    # add 10mV to compensate for
the drop in the FET
  264
  265     def setup_int_wake_up(self, rising, falling):
  266         """ rising is for activity detection, falling for inactivity """
  267         wake_int = False
  268         if rising:
  269             self.set_bits_in_memory(IOCAP_ADDR, 1 << 5)
  270             wake_int = True
```

Anexo 1. Código

```
271         else:
272             self.mask bits in memory(IOCAP ADDR, ~(1 << 5))
273 
274         if falling:
275             self.set_bits_in_memory(IOCAN_ADDR, 1 << 5)
276             wake_int = True
277         else:
278             self.mask_bits_in_memory(IOCAN_ADDR, ~(1 << 5))
279         self.wake_int = wake_int
280 
281     def setup_int_pin_wake_up(self, rising_edge = True):
282         """ allows wakeup to be made by the INT pin (PIC -RC1) """
283         self.wake int pin = True
284         self.wake_int_pin_rising_edge = rising_edge
```

1.4 pysense.py

```
1 from pycoproc import Pycoproc
2
3 __version__ = '1.4.0'
4
5 class Pysense(Pycoproc):
6
7     def __init__(self, i2c=None, sda='P22', scl='P21'):
8         Pycoproc.__init__(self, i2c, sda, scl)
```

Anexo 1. Código

1.5 SI7006A20.py

```
import time
from machine import I2C
import math

__version__ = '0.0.2'

class SI7006A20:
    """ class for handling the temperature sensor SI7006-A20
        +/- 1 deg C error for temperature
        +/- 5% error for relative humidity
        datasheet available at https://www.silabs.com/documents/public/data-sheets/Si7006-A20.pdf """

    SI7006A20_I2C_ADDR = const(0x40)

    TEMP_NOHOLDMASTER = const(0xF3)
    HUMD_NOHOLDMASTER = const(0xF5)

    def __init__(self, pysense = None, sda = 'P22', scl = 'P21'):
        if pysense is not None:
            self.i2c = pysense.i2c
        else:
            self.i2c = I2C(0, mode=I2C.MASTER, pins=(sda, scl))

    def _getWord(self, high, low):
        return ((high & 0xFF) << 8) + (low & 0xFF)

    def temperature(self):
        """ obtaining the temperature(degrees Celsius) measured by sensor """
        self.i2c.writeto(SI7006A20_I2C_ADDR, bytearray([0xF3]))
        time.sleep(0.5)
        data = self.i2c.readfrom(SI7006A20_I2C_ADDR, 3)
        #print("CRC Raw temp data: " + hex(data[0]*65536 + data[1]*256 + data[2]))
        data = self._getWord(data[0], data[1])
        temp = ((175.72 * data) / 65536.0) - 46.85
        return temp

    def humidity(self):
        """ obtaining the relative humidity(%) measured by sensor """
        self.i2c.writeto(SI7006A20_I2C_ADDR, bytearray([0xF5]))
        time.sleep(0.5)
        data = self.i2c.readfrom(SI7006A20_I2C_ADDR, 2)
        data = self._getWord(data[0], data[1])
        humidity = ((125.0 * data) / 65536.0) - 6.0
        return humidity

    def read_user_reg(self):
        """ reading the user configuration register """
        self.i2c.writeto(SI7006A20_I2C_ADDR, bytearray([0xE7]))
        time.sleep(0.5)
        data = self.i2c.readfrom(SI7006A20_I2C_ADDR, 1)
        return data[0]

    def read_heater_reg(self):
        """ reading the heater configuration register """
        self.i2c.writeto(SI7006A20_I2C_ADDR, bytearray([0x11]))
        time.sleep(0.5)
        data = self.i2c.readfrom(SI7006A20_I2C_ADDR, 1)
        return data[0]

    def read_electronic_id(self):
        """ reading electronic identifier """
```

```
   62          self.i2c.writeto(SI7006A20_I2C_ADDR, bytearray([0xFA]) + bytearray([0x0F]))
   63          time.sleep(0.5)
   64          sna = self.i2c.readfrom(SI7006A20_I2C_ADDR, 4)
   65          time.sleep(0.1)
   66          self.i2c.writeto(SI7006A20_I2C_ADDR, bytearray([0xFC]) + bytearray([0xC9]))
   67          time.sleep(0.5)
   68          snb = self.i2c.readfrom(SI7006A20_I2C_ADDR, 4)
   69          return [sna[0], sna[1], sna[2], sna[3], snb[0], snb[1], snb[2], snb[3]]
   70
   71      def read_firmware(self):
   72          """ reading firmware version """
   73          self.i2c.writeto(SI7006A20_I2C_ADDR, bytearray([0x84])+ bytearray([0xB8]))
   74          time.sleep(0.5)
   75          fw = self.i2c.readfrom(SI7006A20_I2C_ADDR, 1)
   76          return fw[0]
   77
   78      def read_reg(self, reg_addr):
   79          """ reading a register """
   80          self.i2c.writeto(SI7006A20_I2C_ADDR, bytearray([reg_addr]))
   81          time.sleep(0.5)
   82          data = self.i2c.readfrom(SI7006A20_I2C_ADDR, 1)
   83          return data[0]
   84
   85      def write_reg(self, reg_addr, value):
   86          """ writing a register """
   87          self.i2c.writeto(SI7006A20_I2C_ADDR, bytearray([reg_addr])+bytearray([value]))
   88          time.sleep(0.1)
   89
   90      def dew_point(self):
   91          """ computing the dew pointe temperature (deg C) for the current Temperature and
Humidity measured pair
   92              at dew-point temperature the relative humidity is 100% """
   93          temp = self.temperature()
   94          humid = self.humidity()
   95          h = (math.log(humid, 10) - 2) / 0.4343 + (17.62 * temp) / (243.12 + temp)
   96          dew_p = 243.12 * h / (17.62 - h)
   97          return dew_p
   98
   99      def humid_ambient(self, t_ambient, dew_p = None):
  100          """ returns the relative humidity compensated for the current Ambient temperature
  101              for ex: T-Ambient is 24.4 degC, but sensor indicates Temperature = 31.65 degC and
Humidity = 47.3%
  102                      -> then the actual Relative Humidity is 72.2%
  103              this is computed because the dew-point should be the same """
  104          if dew_p is None:
  105              dew_p = self.dew_point()
  106          h = 17.62 * dew_p / (243.12 + dew_p)
  107          h_ambient = math.pow(10, (h - (17.62 * t_ambient) / (243.12 + t_ambient)) * 0.4343 +
2)
  108          return h_ambient
```

Anexo 1. Código

A1.2 Funciones

2.1 boot.py

```
"""from machine import UART
import machine
import os

uart = UART(0, baudrate=115200)
os.dupterm(uart)

machine.main('main.py')
"""
```

2.2 EU868.py

```
 1
 2
 3 from network import LoRa
 4 import socket
 5 import binascii
 6 import struct
 7 import time
 8
 9 LORA_CHANNEL = 0 # zero = random
10 LORA_NODE_DR = 4
11 '''
12     utility function to setup the lora channels
13 '''
14 print("utility function to setup the lora channels")
15 def prepare_channels(lora, channel, data_rate):
16     EU868_FREQUENCIES = [
17         { "chan": 1, "fq": "868100000" },
18         { "chan": 2, "fq": "868300000" },
19         { "chan": 3, "fq": "868500000" },
20         { "chan": 4, "fq": "867100000" },
21         { "chan": 5, "fq": "867300000" },
22         { "chan": 6, "fq": "867500000" },
23         { "chan": 7, "fq": "867700000" },
24         { "chan": 8, "fq": "867900000" },
25     ]
26     if not channel in range(0, 9):
27         raise RuntimeError("channels should be in 0-8 for EU868")
28
29     if channel == 0:
30         import  uos
31         channel = (struct.unpack('B',uos.urandom(1))[0] % 7) + 1
32
33     upstream = (item for item in EU868_FREQUENCIES if item["chan"] == channel).__next__()
34
35     # set the 3 default channels to the same frequency
36     lora.add_channel(0, frequency=int(upstream.get('fq')), dr_min=0, dr_max=5)
37     lora.add_channel(1, frequency=int(upstream.get('fq')), dr_min=0, dr_max=5)
38     lora.add_channel(2, frequency=int(upstream.get('fq')), dr_min=0, dr_max=5)
39
40     for i in range(3, 16):
41         lora.remove_channel(i)
42
43     return lora
44
45 '''
46     call back for handling RX packets
47 '''
48 print("call back for handling RX packets")
49 def lora_cb(lora):
50     events = lora.events()
51     if events & LoRa.RX_PACKET_EVENT:
52         if lora_socket is not None:
53             frame, port = lora_socket.recvfrom(512) # longuest frame is +-220
54             print(port, frame)
55     if events & LoRa.TX_PACKET_EVENT:
56         print("tx_time_on_air: {} ms @dr {}", lora.stats().tx_time_on_air, lora.stats().sftx)
```

Anexo 1. Código

A1.3 Main

```
from network import LoRa
import socket
import binascii
import struct
import time
from EU868 import prepare_channels, lora_cb
from pysense import Pysense
from SI7006A20 import SI7006A20
from MPL3115A2 import MPL3115A2, PRESSURE
import pycom
import machine

LORA_CHANNEL = 0 # zero = random
LORA_NODE_DR = 4


sTH = SI7006A20(Pysense())
sP = MPL3115A2(Pysense(),mode=PRESSURE)


def get_press():
    pres= sP.pressure()
    pres= pres/100 #convert to mb
    bpres=int(pres).to_bytes(2, "little")
    return bpres


def get_temp():
    temp= (sTH.temperature() - 5) + 273.15
    btemp=int(temp*100).to_bytes(2, "little")
    return btemp

def get_hum():
    hum= sTH.humidity()
    bhum=int(hum*100).to_bytes(2, "little")
    return bhum


#  authentification parameters
dev_eui = binascii.unhexlify('70B3D549952B3385') # get value from LoPy when connect
app_key = binascii.unhexlify('70B3D57ED001454F') # not used leave empty loraserver.io
nwk_key = binascii.unhexlify('E349FBFC2374A8F6C0AD353CE6A987CD') #from ttn web


print('Main operations: this is sample code for LoRaWAN on EU868')
lora = LoRa(mode=LoRa.LORAWAN, region=LoRa.EU868, device class=LoRa.CLASS C)
prepare_channels(lora, 1, LORA_NODE_DR)
# join a network using OTAA
lora.join(activation=LoRa.OTAA, auth=(dev_eui, app_key, nwk_key), timeout=0,  dr=LORA_NODE_DR)
# DR is 2 in v1.1rb but 0 worked for ne
# wait until the module has joined the network
print('Over the air network activation ... ', end='')
while not lora.has_joined():
    time.sleep(2.5)
    print('.', end='')
    print('')
# create a LoRa socket
lora_socket = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
# set the LoRaWAN data rate
lora_socket.setsockopt(socket.SOL_LORA, socket.SO_DR, LORA_NODE_DR)
# msg are confirmed at the FMS level
lora_socket.setsockopt(socket.SOL_LORA, socket.SO_CONFIRMED, 0)
# make the socket non blocking y default
```

```
lora_socket.setblocking(False)
lora.callback(trigger=( LoRa.RX_PACKET_EVENT |
                        LoRa.TX_PACKET_EVENT |
                        LoRa.TX_FAILED_EVENT  ), handler=lora_cb)
time.sleep(4) # this timer is important to give enought time process to finish

pkt = get_temp() + get_hum() + get_press()
print('Sending:', pkt)
lora_socket.send(pkt)
time.sleep(10)
machine.deepsleep(60000)
```

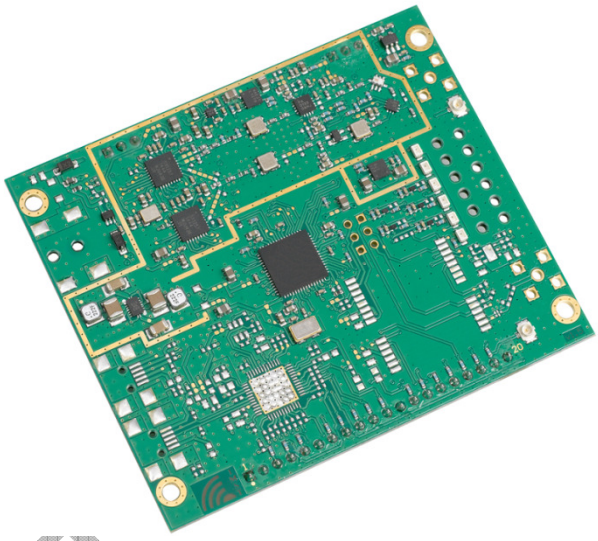
Anexo 2. Hojas de características

Contenido Anexo 2. Hojas de características

1. IC880a
2. Raspberry Py 3B+
3. ANT-868-ID
4. LoPy4
5. Pysense
6. PoE29U

WiMOD iC880A

DATASHEET



Document ID: 4000/40140/0074

IMST GmbH
Carl-Friedrich-Gauß-Str. 2-4
47475 KAMP-LINTFORT
GERMANY

Document Information

File name	iC880A_Datasheet.docx
Created	2018-06-04
Total pages	26

Revision History

Version	Note
0.9	Created
0.10	DC Jack polarity added
0.11	Reference antenna added, current consumption updated (Table 5-2)
0.12	PA settings updated (Table 5 5), SPI electrical characteristics added
0.13	PA settings updated (Table 5 5) RSSI offset information added to chapter 5.4.2
0.14	Added new picture of iC880A
0.15	Please note that USB driver support for iC880A-USB isn't provided anymore on https://github.com/Lora-net/lora_gateway from version 3.2."
0.16	Electrical IO specification update
0.17	Update chapter 6.2
0.50	chapter 5.4.1 updated , Table 5-1 updated, Table 2-1 removed
0.51	chapter 6.1 added MISO low impedance note chapter 5.4.1 tx settings update chapter 5.1 temperature range update
1.0	iC880A-USB removed and complete update of datasheet Table 5-1 updated
1.1	Table 5-1 updated

Aim of this Document

The aim of this document is to give a product description including interfaces, features and performance of the concentrator module iC880A-SPI.

Important Note

Caution: Operating the iC880A outside the given specification may harm the device.

Table of Contents

1. Introduction

The concentrator module iC880A is targeted for a huge variety of applications like Smart Metering, IoT and M2M applications. It is a multi-channel high performance transmitter/receiver module designed to receive several LoRa packets simultaneously using different spreading factors on multiple channels. The concentrator module iC880A can be integrated into a gateway as a complete RF front end of this gateway. It provides the possibility to enable robust communication between a LoRa gateway and a huge amount of LoRa end-nodes spread over a wide range of distance. The iC880A needs a host system for proper operation. This host system can be a PC or MCU that will be connected to iC880A via SPI-Interface.

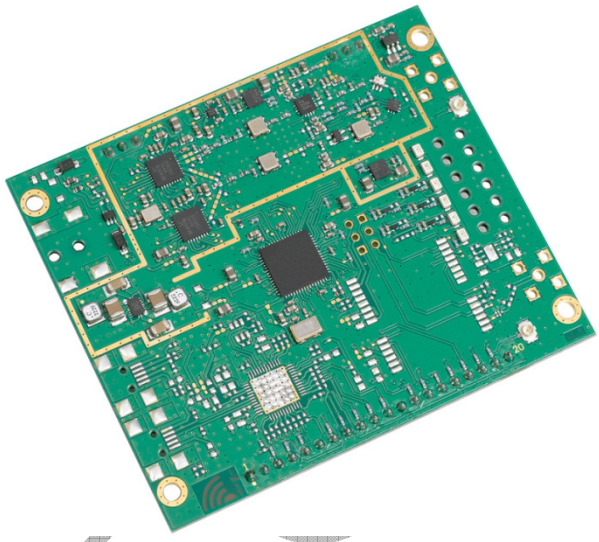


Figure 1-1: Picture of iC880A-SPI

The iC880A is able to receive up to 8 LoRa® packets simultaneously sent with different spreading factors and also on different channels. This unique capability allows to implement innovative network architectures advantageous over other short range systems:

- End-point nodes (e.g. sensor nodes) can change frequency with each transmission in a random pattern. This provides vast improvement of the system robustness in terms of interferer immunity and radio channel diversity.
- End-point nodes can dynamically perform link rate adaptation based (by adapting their spreading factors) on their link margin without adding complexity to the protocol. There is no need to maintain a table of which end point uses which data rate, because all data is demodulated in parallel.
- The capacity of the air interface can be increased due to orthogonal spreading factors.
- Due to the high range a star topology can be used. This results in simple implementation avoiding complex network layers, wireless routers and additional network protocol traffic.

1.1 Key Features

- Compact size 79.8 x 67.3 mm
- LoRa® modulation technology
- Frequency band 868 MHz
- Orthogonal spreading factors
- Sensitivity down to -137 dBm
- SPI interface
- SX1301 base band processor
- Emulates up to 49 x LoRa demodulators
- 10 parallel demodulation paths
- 1 (G)FSK demodulator
- 2 x SX1257 Tx/Rx front-ends
- Supply voltage 5 V
- RF interface optimized to 50 Ω
- Output power level up to 20 dBm
- GPS receiver (optional)
- Range up to 15 km (Line of Sight)
- Range of several km in urban environment[1]
- Status LEDs
- HAL is available from https://github.com/Lora-net/lora_gateway

1.2 Applications

- Smart Metering
- Wireless Star Networks
- Home-, Building-, Industrial automation
- Remote Control
- Wireless Sensors
- M2M, IoT
- Wireless Alarm and Security Systems
- LoRaWAN™
- …

Please visit our web site www.wireless-solutions.de for further information.

[1] Depending on the environment

2. LoRa Modulation Technique

The iC880A uses Semtech's LoRa® spread spectrum modulation technique. This modulation, in contrast to conventional modulation techniques, permits an increase in link budget and increased immunity to in-band interference.
LoRa also provides significant advantages in both blocking and selectivity, solving the traditional design compromise between range, interference immunity and energy consumption, please refer to [1].
Semtech's LoRa® technology transceivers support several bandwidth options and spreading factors ranging from 7 to 12. The spread spectrum LoRa® modulation is performed by representing each bit of payload information by multiple chips of information. The rate at which the payload information is sent is referred to as the nominal symbol rate (Rs), the ratio between the nominal symbol rate and chip rate is the spreading factor and represents the number of modulation symbols sent per bit of information. Note that the spreading factor must be normally known in advance on both transmit and receive sides of the radio link as different spreading factors are orthogonal to each other. Note also the resulting signal to noise ratio (SNR) required at the receiver input. It is the capability to receive signals with negative SNR that increases the sensitivity, so link budget and range, of the LoRa receiver.
For further information on LoRa® please refer to [2].

3. Module Overview

The Concentrator Module is currently available as "iC880A-SPI". An overview about designation of the key components is given by the following picture

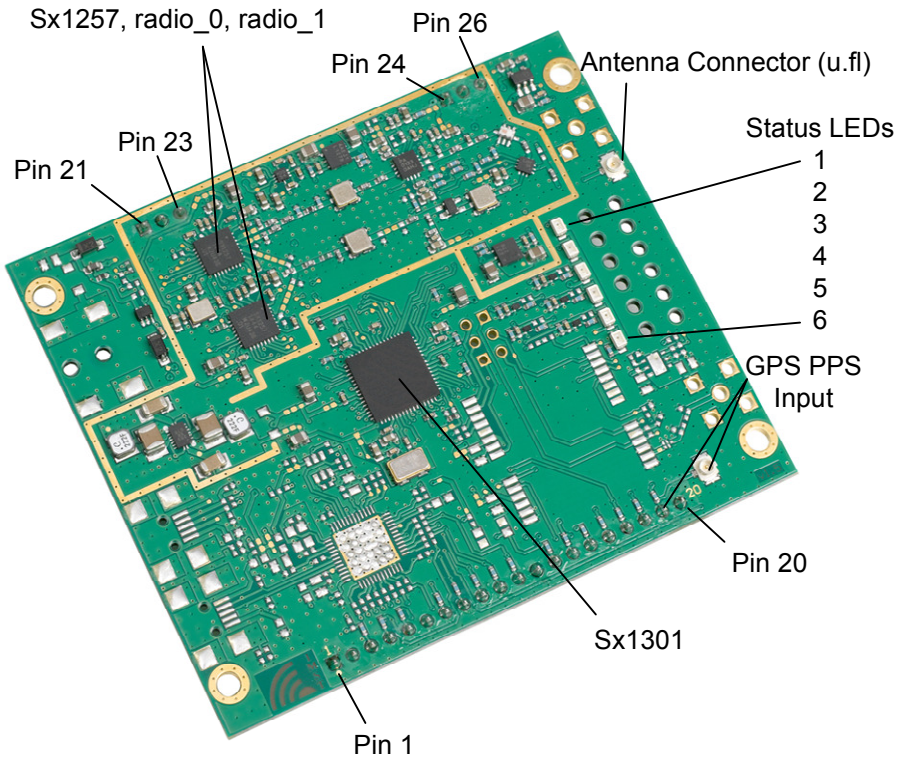


Figure 3-1: Component Overview iC880A-SPI

3.1 SX1301

The iC880A includes Semtech's SX1301 which is a digital baseband chip including a massive digital signal processing engine specifically designed to offer breakthrough gateway capabilities in the ISM bands worldwide. It integrates the LoRa concentrator IP.

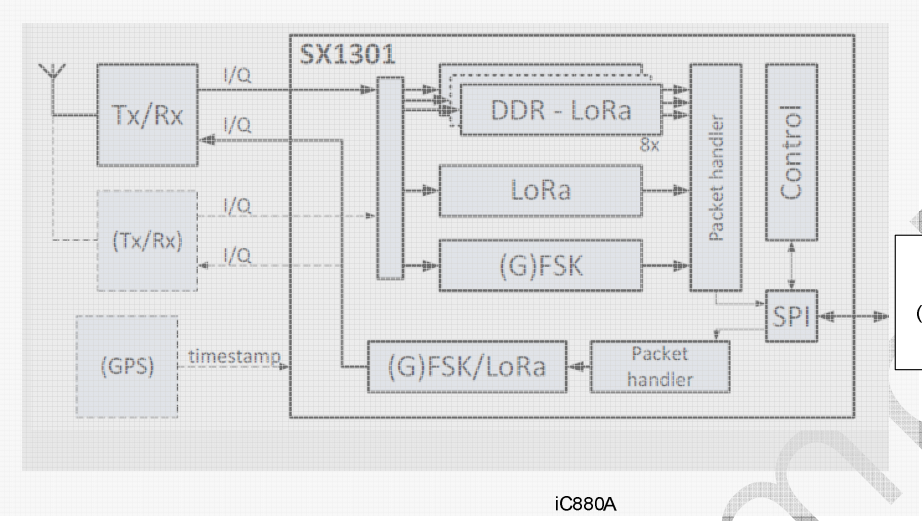


Figure 3-2: Block Diagram of iC880A with SX1301 Base Band Processor.

The SX1301 is a smart baseband processor for long range ISM communication. In the receiver part, it receives I and Q digitized bit stream for one or two receivers (SX1257), demodulates these signals using several demodulators, adapting the demodulators settings to the received signal and stores the received demodulated packets in a FIFO to be retrieved from a host system (PC, MCU). In the transmitter part, the packets are modulated using a programmable (G)FSK/LoRa modulator and sent to one transmitter (SX1257). Received packets can be time-stamped using a GPS PPS input.

The SX1301 has an internal control block that receives microcode from the host system (PC, MCU). The microcode is provided by Semtech as a binary file to load into the SX1301 at power-on (see Semtech application support for more information).
The control of the SX1301 by the host system (PC, MCU) is made using a Hardware Abstraction Layer (HAL). The Hardware Abstraction Layer source code is provided by Semtech and can be adapted by the host system developers.

It is highly recommended to fully re-use the latest HAL as provided by Semtech on https://github.com/Lora-net.

3.1.1 Block Diagram

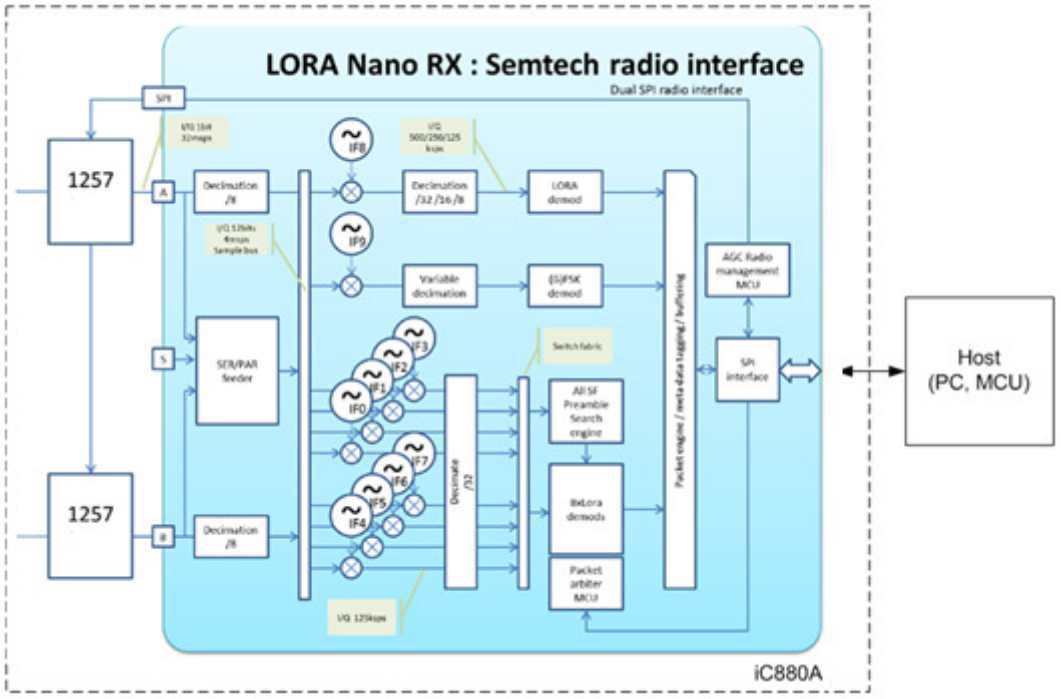


Figure 3-3: Detailed Block Diagram of SX1301 taken from [5].

The SX1301 digital baseband chip contains 10 programmable reception paths. Those paths have differentiated levels of programmability and allow different use cases. It is important to understand the differences between those demodulation paths to make the best possible use from the system.

3.1.2 IF8 LORA channel

This channel is connected to one SX1257 using any arbitrary intermediate frequency within the allowed range. This channel is LoRa only. The demodulation bandwidth can be configured to be 125, 250 or 500 kHz. The data rate can be configured to any of the LoRa available data rates (SF7 to SF12) but, as opposed to IF0 to IF7, ONLY the configured data rate will be demodulated. This channel is intended to serve as a high speed backhaul link to other gateways or infrastructure equipment. This demodulation path is compatible with the signal transmitted by the SX1272 (iM880B, iM881A) and SX1276 chip family.

3.1.3 IF9 (G) FSK channel

The IF9 channel is connected to a GFSK demodulator. The channel bandwidth and bit rate can be adjusted. This demodulator offers a very high level of configurability, going well beyond the scope of this document. The demodulator characteristics are essentially the same than the GFSK demodulator implemented on the SX1232 and SX1272 (iM880B, iM881A) Semtech chips. This demodulation path can demodulate any legacy FSK or GFSK formatted signal.

3.1.4 IF0 to IF7 LORA channels

Those channels are connected to one SX1257. The channel bandwidth is 125 kHz and cannot be modified or configured. Each channel IF frequency can be individually configured. On each of those channels any data rate can be received without prior configuration.

Several packets using different data rates (different spreading factors) may be demodulated simultaneously even on the same channel. Those channels are intended to be used for a massive asynchronous star network of 10000's of sensor nodes. Each sensor may use a random channel (amongst IF0 to IF7) and a different data rate for any transmission.

Sensors located near the gateway will typically use the highest possible data rate in the fixed 125 kHz channel bandwidth (e.g. 6 kbit/s) while sensors located far away will use a lower data rate down to 300 bit/s (minimum LoRa data rate in a 125 kHz channel).

The SX1301 digital baseband chip scans the 8 channels (IF0 to IF7) for preambles of all data rates at all times.

The chip is able to demodulate simultaneously up to 8 packets. Any combination of spreading factor and intermediate frequency for up to 8 packets is possible (e.g. one SF7 packet on IF0, one SF12 packet on IF7 and one SF9 packet on IF1 simultaneously).

The SX1301 can detect simultaneously preambles corresponding to all data rates on all IF0 to IF7 channels. However, it cannot demodulate more than 8 packets simultaneously. This is because the SX1301 architecture separates the preamble detection and signal acquisition task from the demodulation process. The number of simultaneously demodulated packets (in this case 8) is an arbitrary system parameter and may be set to other values for a customer specific circuit.

The unique multi data-rate multi-channel demodulation capacity SF7 to SF12 and of channels IF0 to IF7 allows innovative network architectures to be implemented.

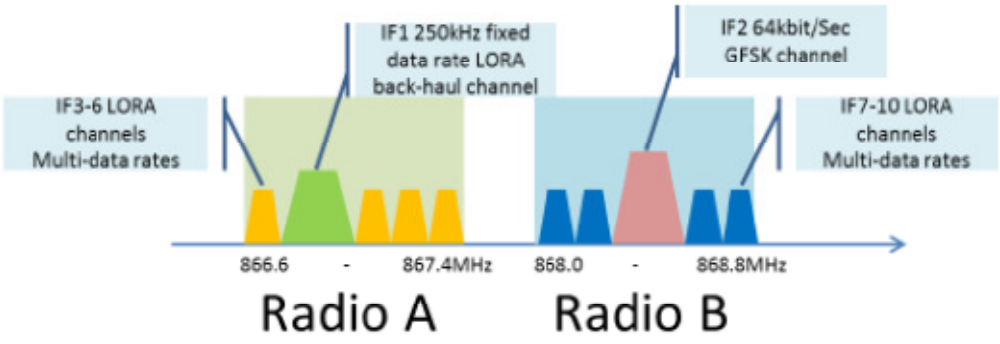


Figure 3-4: Possible use of radio spectrum taken from [5].

3.2 RF Interface

The iC880A supports an RF interface for the 868 MHz frequency band. By connecting an appropriate antenna[1] to the antenna connector, the iC880A is fully ready for communication.

3.3 External Module Connector

For easy integration into a target system and mounting of the iC880A on a carrier board, the headers on the module's bottom side can be used for these purposes (refer to Table 6-1 for the pin description).

3.3.1 SPI

The connector on the bottom side provides an SPI connection, which allows direct access to the Sx1301 SPI interface. This gives the target system the possibility to use existing SPI interfaces to communicate to the iC880A.

After powering up the iC880A it is required to reset SX1301 via PIN 13, refer to Table 6-1.

3.3.2 GPS PPS

In case of available PPS signals in the target system, it is possible to connect this available signal to the appropriate pin at the connector.

3.3.3 UART

The bottom connector provides a UART interface. This interface is for future use.

3.3.4 Digital IOs

There are five GPIOs of the Sx1301 available, which gives the user some possibilities to get information about the system status. These pins are the same, as they are used for the LEDs on the iC880A.

The functions of the LED are depending on the software used for operating the concentrator. The intentional use of the LEDs is as follows:

1) Backhaul packet
2) TX packet
3) RX Sensor packet
4) RX FSK packet
5) RX buffer not empty
6) Power

[1] Recommended antenna is CTA868/2/DR/SM/S2, available at CompoTEK GmbH, Germany

4. LoRa Systems, Network Approach

The use of LoRa® technology can be distinguished in "Public" and "Private" networks. In both cases the usage of a concentrator module can be reasonable. Public networks are operator (e.g. telecom) managed networks whereas private networks are individually managed networks.

LoRa networks are typically star or multiple star networks where a gateway relays the packets between the end-nodes and a central network server, see Figure 4-1. For private network approaches the server can also be implemented on the gateway host.

Due to the possible high range the connection between end-nodes and the concentrator iC880A is always a direct link. There are no repeaters or routers within a LoRa network.

Depending on the used spreading factor and signal bandwidth different data rates[1] (0.3 kbps to ~22 kbps) and sensitivities down to -137 dBm are possible. Spreading factor and signal bandwidth are a trade-off between data rate and communication range.

4.1 Overview

The iC880A is able to receive on different frequency channels at the same time and is able to demodulate the LoRa signal without knowledge of the used spreading factor of the sending node.

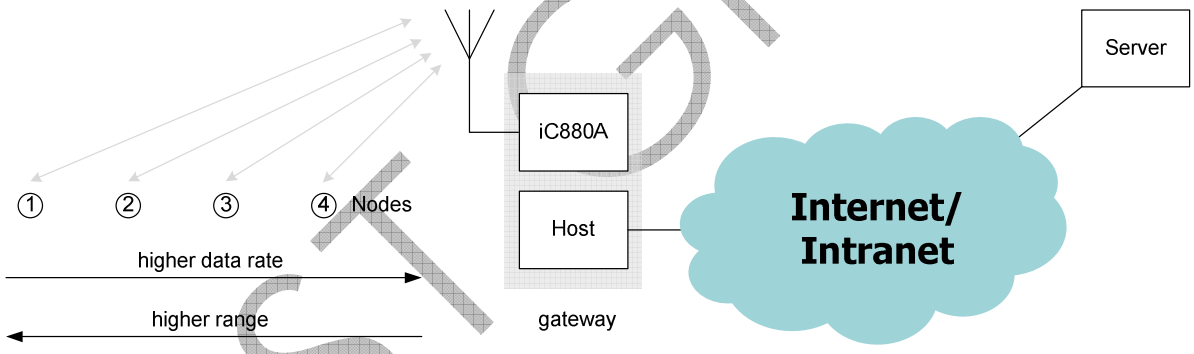


Figure 4-1: Public LoRa Network Approach

Due to the fact that the combination of spreading factors and signal bandwidths results in different data rates the use of "Dynamic Data-Rate Adaption" becomes possible. That means that LoRa nodes with high distances from the iC880A must use higher spreading factors and therefore have a lower data rate. LoRa nodes which are closer to the concentrator can use lower spreading factors and therefore can increase their data rate.

Due to the fact that spreading factors are orthogonal and iC880A supports up to 10 demodulations paths the channel capacity of a LoRa cell can be increased using iC880A compared to conventional modulation techniques.

[1] Equivalent bit rate.

4.2 Firmware

The LoRaWAN specification is currently driven by the LoRa Alliance™. Currently all available software, firmware and documentation can be found and downloaded from the open source project LoRa-net hosted on https://github.com/Lora-net

This project considers all parts that are needed to run a network based on LoRa technology. It includes the node firmware (several hardware platforms are supported), the gateway host software (HAL driver for SX1301, packet forwarder) and a server implementation.

It is highly recommended to re-use the latest HAL as provided. The "iC880A_QuickStartGuide.pdf" is available on request.

5. Electrical Characteristics & Timing specifications

In the following different electrical characteristics of the iC880A are listed. Furthermore details and other parameter ranges are available on request.

Note: Stress exceeding of one or more of the limiting values listed under "Absolute Maximum Ratings" may cause permanent damage to the radio module.

5.1 Absolute Maximum Ratings

Parameter	Condition	Min	Typ.	Max	Unit
Supply Voltage (VDD)		-0.3	5.0	5.5	V
Operating Temperature		-5		+55	°C
Operating Temperature (extended)		-40		+85°C	°C
RF Input Power				-15	dBm
Max Pin on Hot-Switching RF-Switch				+20	dBm
VSWR on any RF connector				3:1	
Notes:					

Table 5-1: Absolute Maximum Ratings

Note: With RF output power level above +15 dBm a minimum distance to a transmitter should be 1 m for avoiding too large input level.

5.2 Global Electrical Characteristics

T = 25°C, VDD = 5 V (typ.) if nothing else stated

Parameter	Condition	Min	Typ.	Max	Unit
Supply Voltage (VDD)		4.5	5.0	5.5	V
Receiver Current Consumption	medium activity (2 radios, 4 active paths)		288		mA
	high activity (2radios, 10 active paths)		428		

Table 5-2: General Characteristics

T = 25°C, VDD = 5 V (typ.) if nothing else stated

Parameter	Condition	Min	Typ.	Max	Unit
Logic low input threashold (VIL)	"0" logic input			0.4	V
Logic high input threashold (VIH)	"1" logic input	2.9		3.3	V
Logic low ouput level (VOL)	"0" logic output, 2 mA sink			0.4	V
Logic high output level (VOH)	"1" logic output, 2 mA source	2.9		3.3	V
Notes:					

Table 5-3: Electrical characteristics of IOs

5.3 SPI Interface Characteristics

T = 25°C, VDD = 5 V (typ.) if nothing else stated

Parameter	Condition	Min	Typ.	Max	Unit
SCK frequency				10	MHz
SCK high time		50			ns
SCK low time		50			ns
SCK rise time			5		ns
SCK fall time			5		ns
MOSI setup time	From MOSI change to SCK rising edge	10			ns
MOSI hold time	From SCK rising edge to MOSI change	20			ns
NSS setup time	From NSS falling edge to SCK rising edge	40			ns
NSS hold time	From SCK falling edge to NSS rising edge, normal mode	40			ns
NSS high time between SPI accesses		40			ns

Table 5-4: Timing characteristics of SPI Interface

5.4 RF Characteristics

5.4.1 Transmitter RF Characteristics

The iC880A has an excellent transmitter performance, which generally give a lot of possible settings for the power amplifier of the iC880A. It is highly recommended, to use an optimized configuration for the power level configuration. An application note called AN22 "iC880A Power Amplifier Settings" [6] is available, which describes optimized settings for the transmitter configuration. It is available on request.

The iC880A is specified for a max. RF output power of +20 dBm. Long-term operating of the iC880A with more than +20 dBm can destroy the internal power amplifier of iC880A. Especially in case of operating the iC880A with the github software it need to be ensured, that the maximum RF output power of +20 dBm is not exceeded. Therefore the settings of the global_conf.json might need to be changed accordingly.

T = 25°C, VDD = 5 V (typ.), 866.5 MHz if nothing else stated

Parameter	Condition	Min	Typ.	Max	Unit
Frequency Range		863	-	870	MHz
Modulation Techniques	FSK / LoRa™				
TX Frequency Variation vs. Temperature	-5°C to +55°C	-	+/- 3	-	kHz
TX Power Variation vs. Temperature[1]	Max. power level, -5°C to +55°C	-	+/- 2.4	-	dB
	Max. power level, -40°C to +85°C		+/-4.9	-	dB
TX Power Variation vs. Frequency	Max. power level	-	+/- 2	-	dB
TX Power Variation (initial)	Max. power level	-	+/- 1.5	-	dB
TX Current Consumption	Gain setting for nom. +14 dBm -5°C to +55°C PA=2; DAC=3; Mix=10; Dig=2	-	260	-	mA
	Gain setting for nom. +20 dBm -5°C to +55°C PA=2; DAC=3; Mix=12; Dig=0	-	300	-	mA
Notes: All values are based on settings given by Application Note 22 [6] which is available on request					

Table 5-5: Transmitter RF Characteristics

[1] Operational temperature range can be basically extended to -40°C to +85°C, but a larger power level drift vs. temperature need to be expected. In addition it is recommended, to use optimized TX settings which are also individually adapted to the temperature.

5.4.2 Receiver RF Characteristics

It is highly recommended, to use optimized RSSI calibration values. For both, Radio 1 and 2, the RSSI-Offset should be set to -169.

The following table gives typically sensitivity level of the iC880A:

Signal Bandwidth/[kHz]	Spreading Factor	Sensitivity/[dBm]
125	12	-137
125	7	-126
250	12	-136
250	7	-123
500	12	-134
500	7	-120

Table 5-6: Typically Radio Performance of iC880A

5.4.3 Certification and Compliancy Restrictions

This component has been designed to comply with the European Union's RE-Directive (Radio Equipment Directive) 2014/53/EU. As the product is a component only, the assessment is done on EMC and ERM (EN 300 220 v3.1.1) only. A declaration of conformity for this component will be available from IMST GmbH on request. National laws and regulations, as well as their interpretation can vary with the country. In case of uncertainty, it is recommended to contact either IMST's accredited Test Center or to consult the local authorities of the relevant countries.

RED pre-certification was done based on the LoRa Lite Gateway with an antenna giving 2.15 dBi gain. Due to special requirements of the RED certification the current output power may need to be limited by software depending on operational frequency and with respect to specified extreme conditions.

6. Module Package

In the following the iC880A module package is described. This description includes the iC880A pinout as well as the modules dimensions.

6.1 Pinout Description

The iC880A provides headers at the bottom side, which have a pitch of 2.54 mm. The description of the pins is given by Table 6-1. An additional overview gives Figure 6-1.

PIN	PIN Name	PIN Type	Description
1	GND	Power	
2	NC	NC	Reserved
3	nGPS_Reset	Input	GPS Module Reset (low active)
4	SPValid	Input	Sx1301 Radio C Sample Valid (don't connect)
5	EN_GPS_Supply	Input	GPS Module LDO: Enable Pin
6	NC	NC	Reserved
7	GPIO0	I/O	Sx1301 GPIO 0
8	GPIO1	I/O	Sx1301 GPIO 1
9	GPIO3	I/O	Sx1301 GPIO 3
10	GPIO2	I/O	Sx1301 GPIO 2
11	GPIO4	I/O	Sx1301 GPIO 4
12	GND	Power	
13	Reset	Reset	Sx1301 Reset, for a stable start-up Reset should be at high-level for 100 ns (min), once the supply voltage is stable . Internally pulled-down with 100 kΩ.
14	CLK	Input	Sx1301 SPI-Clock
15	MISO	Output	Sx1301 SPI-MISO
16	MOSI	Input	Sx1301 SPI-MOSI
17	NSS	Input	Sx1301 SPI-NSS
18	ScanMode	Input	Sx1301 ScanMode Signal
19	PPS	Input	GPS PPS Input Signal
20	GND	Power	
21	VDD	Power	+5 V Supply Voltage
22	GND	Power	
23	VDDB	Power	GPS backup supply voltage
24	GND	Power	
25	GPS_TX	Output	GPS UART TxD
26	GPS_RX	Input	GPS UART RxD

Table 6-1: iC880A Pinout Table

Note: MISO-signal is always low impedance. Do not share with other MISO-signal by direct connection.

6.2 Module Dimensions

The outer dimensions of the iC880A are given by 79.8 x 67.3 mm ± 0.2 mm. The iC880A provide four drills for screwing the PCB to another unit each with a drill diameter of 3 mm.

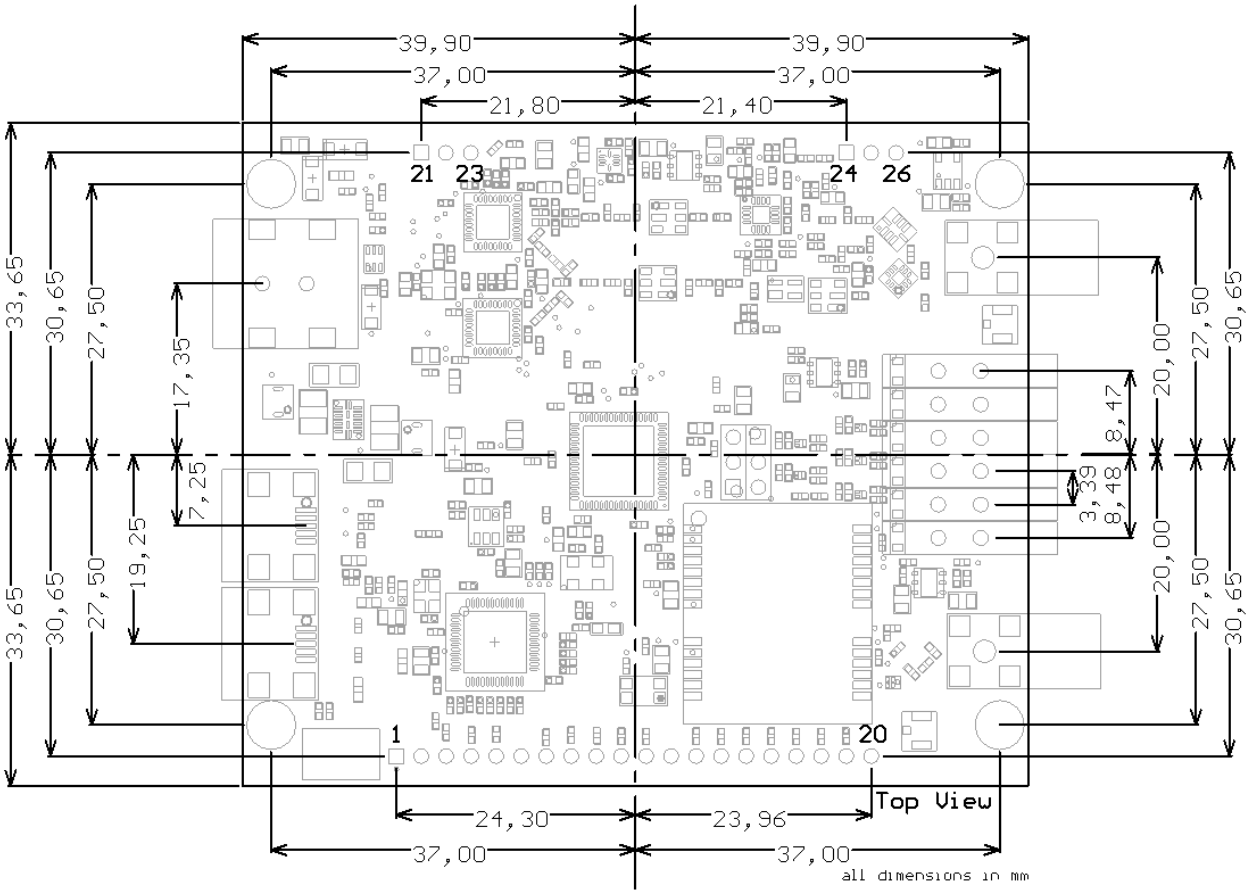


Figure 6-1: iC880A outlines and pins of bottom connector in top view

7. Ordering Information

Ordering Part Number	Description	Distributor
iC880A-SPI	Concentrator Module with SPI interface	

Table 7-1: Ordering Information

8. Restrictions and Limitations

8.1 Hardware Restrictions and Limitations

The characteristic values given by the present document are typically obtained by measurements based on evaluation kits of the entitled device. Using other carrier boards or connected equipment might lead to different characteristics. Subject to given measurement results the characteristic values might show the best performance of the entitled device, independent from any compliancy restriction of final operation purposes.

8.2 Software Restrictions and Limitations

The present document is a datasheet of the entitled device which intentional use is to provide information about basic characteristics related to the device hardware. Typically all described characteristic values require software for obtaining them accordingly. All features of the available software are subject to changes without claim to be complete at any time. Characteristically values might also be provided based on datasheets of the appropriate key components unless there are test results available based on the available software. For more information regarding current supported features of the available software refer to www.wireless-solutions.de.

8.3 Compliancy Restrictions and Limitations

The entitled device has been designed to comply with the standards namely given in the present document. The intentional operation shall be in so called ISM bands, which can be used free of charge within the European Union and typically licences free all over the world. Nevertheless, restrictions such as maximum allowed radiated RF power or duty cycle may apply which might result in a reduction of these parameters accordingly.

In addition, the use of radio frequencies might be limited by national regulations which requirements also need to be met.

In case the entitled device will be embedded into other products (referred as “final products”), the manufacturer for this final product is responsible to declare the conformity to required standards accordingly. A proof of conformity for the entitled device is available from IMST GmbH on request. Beside the entitled device the conformity also considers software as well as supporting hardware characteristics which might also have an impact accordingly.

The applicable regulation requirements are subject to change. IMST GmbH does not take any responsibility for the correctness and accuracy of the aforementioned information. National laws and regulations, as well as their interpretation can vary with the country. In case of uncertainty, it is recommended to contact either IMST’s accredited Test Center or to consult the local authorities of the relevant countries.

8.4 Disclaimer

IMST GmbH points out that all information in this document are given on an "as is" basis. No guarantee, neither explicit nor implicit is given for the correctness at the time of publication. IMST GmbH reserves all rights to make corrections, modifications, enhancements, and other changes to its products and services at any time and to discontinue any product or service without prior notice. It is recommended for customers to refer to the latest relevant information before placing orders and to verify that such information is current and complete. All products are sold and delivered subject to "General Terms and Conditions" of IMST GmbH, supplied at the time of order acknowledgment. IMST GmbH assumes no liability for the use of its products and does not grant any licenses for its patent rights or for any other of its intellectual property rights or third-party rights. It is the customer's duty to bear responsibility for compliance of systems or units in which products from IMST GmbH are integrated with applicable legal regulations. Customers should provide adequate design and operating safeguards to minimize the risks associated with customer products and applications. The products are not approved for use in life supporting systems or other systems whose malfunction could result in personal injury to the user. Customers using the products within such applications do so at their own risk.

Any reproduction of information in datasheets of IMST GmbH is permissible only if reproduction is without alteration and is accompanied by all given associated warranties, conditions, limitations, and notices. Any resale of IMST GmbH products or services with statements different from or beyond the parameters stated by IMST GmbH for that product/solution or service is not allowed and voids all express and any implied warranties. The limitations on liability in favor of IMST GmbH shall also affect its employees, executive personnel and bodies in the same way. IMST GmbH is not responsible or liable for any such wrong statements.

9. Appendix

9.1 List of Abbreviations

AFA	Adaptive Frequency Agility
BER	Bit Error Rate
BSC	Basic Spacing between Centers
GND	Ground
GPIO	General Purpose Input/Output
GPS	Global Positioning System
HAL	Hardware Abstraction Layer
IF	Intermediate Frequency
IoT	Internet of Things
ISM	Industrial, Scientific and Medical
LBT	Listen Before Talk
M2M	Machine to Machine
MAC	Medium Access Control
MCU	Microcontroller Unit
MPSSE	Multi-Protocol Synchronous Serial Engine (FTDI)
PCB	Printed Circuit Board
PPS	Pulse Per Second
RAM	Random Access Memory
RF	Radio Frequency
SMT	Surface Mounted Technology
SNR	Signal to Noise Ratio
SPI	Serial Peripheral Interface
TRX	Transceiver
USB	Universal Serial Bus

9.2 List of Figures

9.3 List of Tables

9.4 References

[1] IMST, iM880B-L Data sheet from www.wireless-solutions.de

[2] Semtech, White Paper LoRa Modulation from www.semtech.com

[3] ERC Recommendation 70-03 "Relating to the use of Short Range Devices (SRD)", Tromsø 1997, CEPT ECC subsequent amendments 13 October 2017

[4] IMST, iM880B_AN016_RFSettings from www.wireless-solutions.de

[5] Semtech, SX1301 Data Sheet from www.semtech.com

[6] IMST, AN022 iC880A Power Amplifier Settings

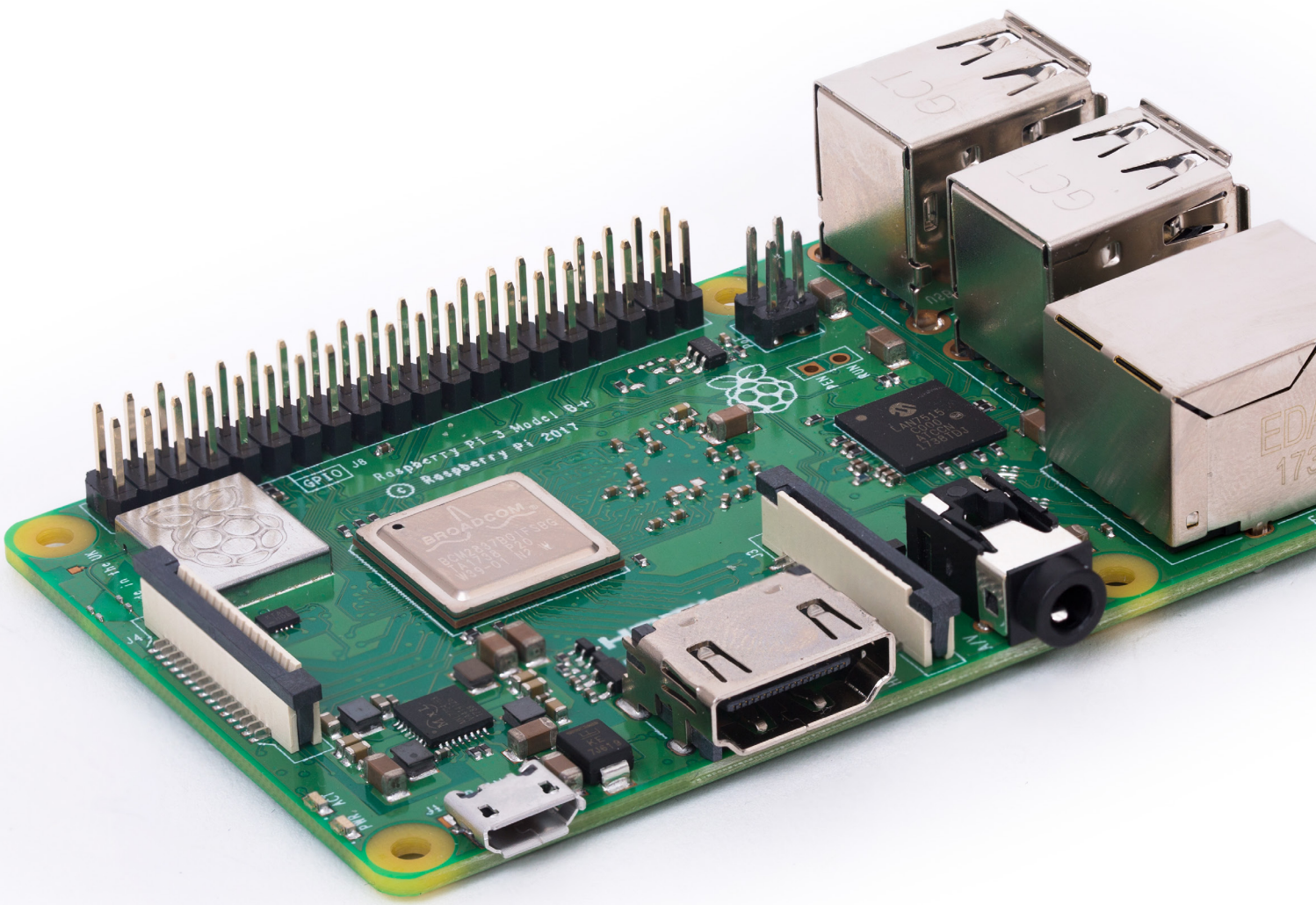
10. Contact Information

IMST GmbH

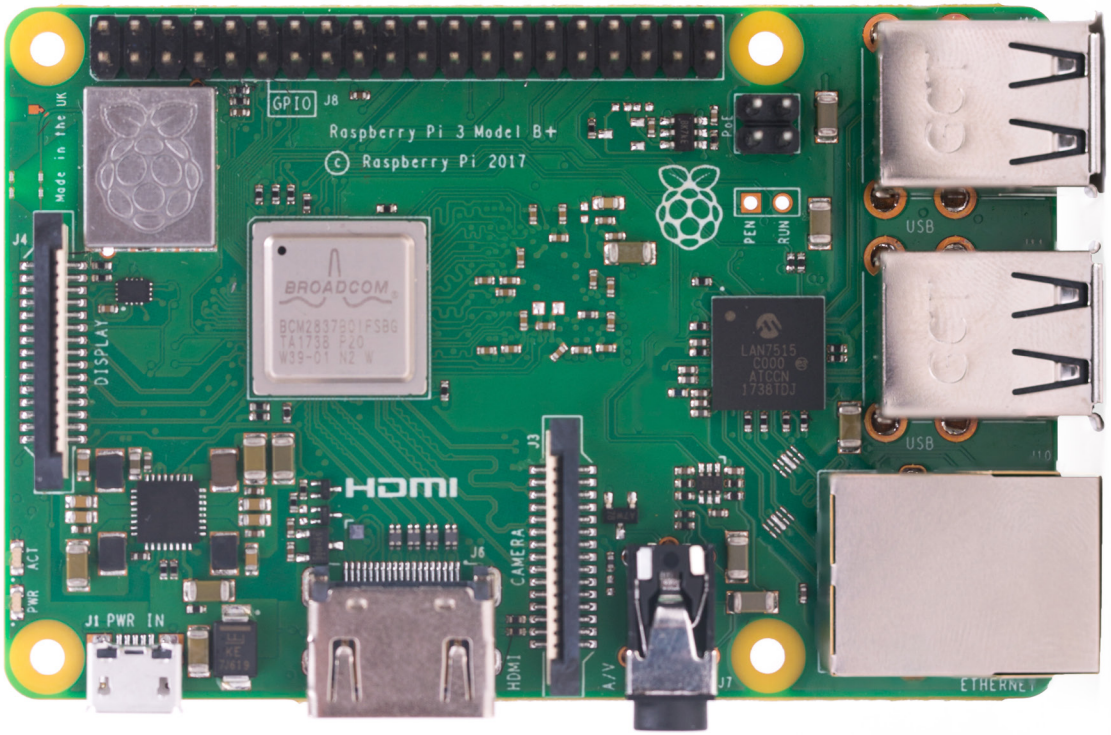
Carl-Friedrich-Gauss-Str. 2-4
47475 Kamp-Lintfort
Germany

T +49 2842 981 0
F +49 2842 981 299
E wimod@imst.de
I www.wireless-solutions.de

Raspberry Pi 3 Model B+



Overview



The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting a 64-bit quad core processor running at 1.4 GHz, dual-band 2.4 GHz and 5 GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT

The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market.

The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

Specifications

Processor:
Broadcom BCM2837B0, Cortex-A53
64-bit SoC @ 1.4 GHz

Memory:
1GB LPDDR2 SDRAM

Connectivity:
- 2.4 GHz and 5 GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
- 4 × USB 2.0 ports

Access:
Extended 40-pin GPIO header

Video & sound:
- 1 × full size HDMI
- MIPI DSI display port
- MIPI CSI camera port
- 4 pole stereo output and composite video port

Multimedia:
H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics

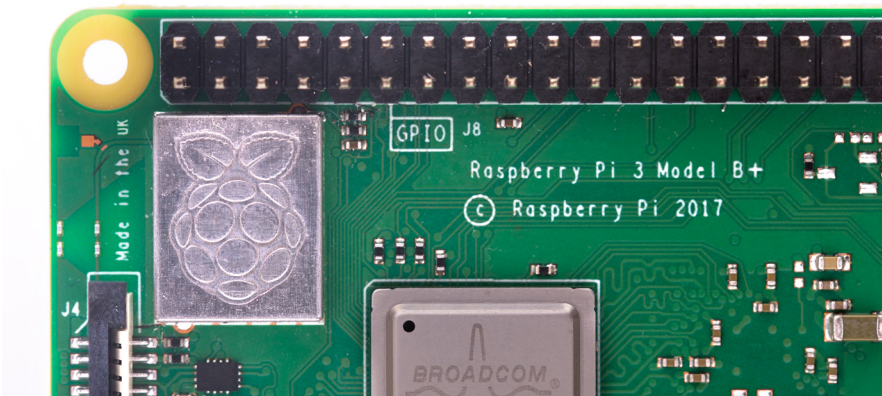
SD card support:
Micro SD format for loading operating system and data storage

Input power:
- 5 V/2.5 A DC via micro USB connector
- 5 V DC via GPIO header
- Power over Ethernet (PoE)–enabled (requires separate PoE HAT)

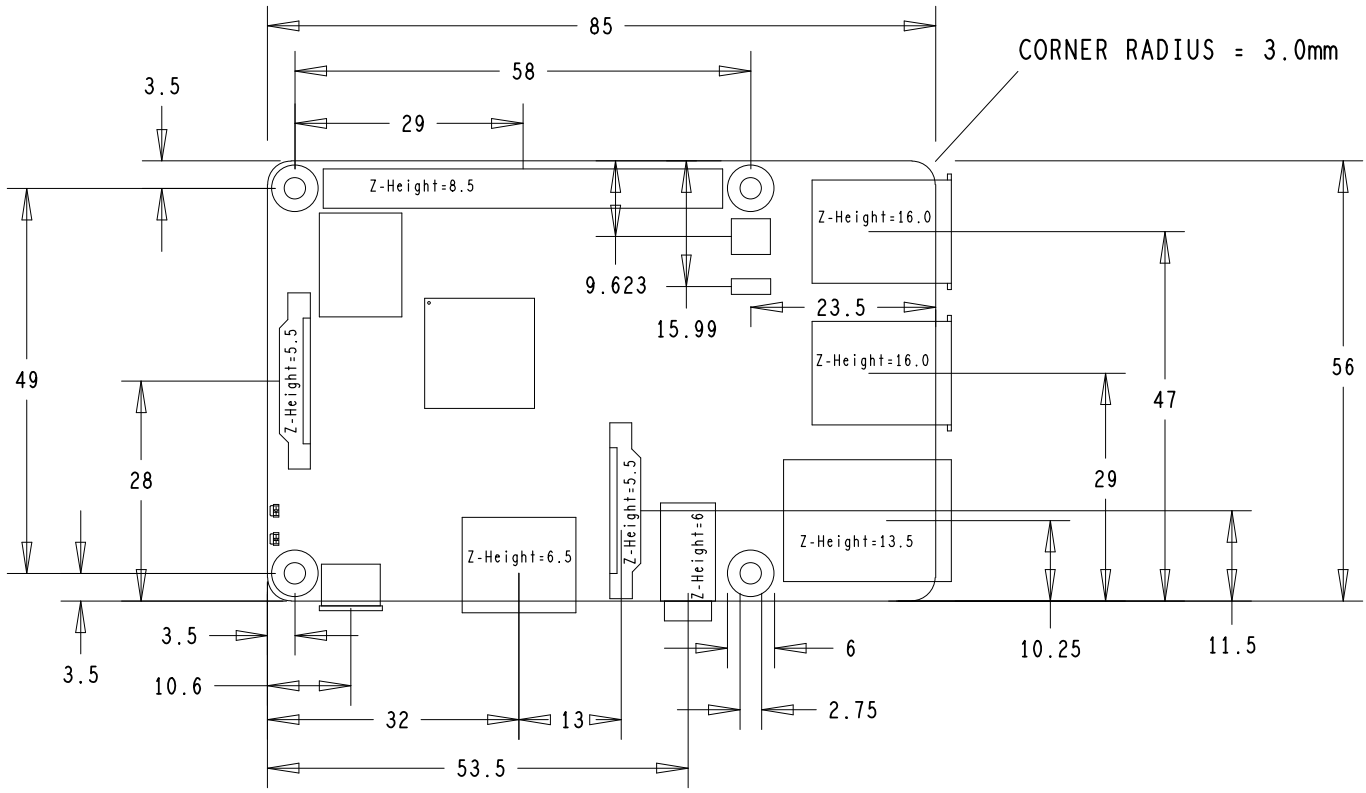
Environment:
Operating temperature, 0–50 °C

Compliance:
For a full list of local and regional product approvals, please visit www.raspberrypi.org/products/raspberry-pi-3-model-b+

Production lifetime:
The Raspberry Pi 3 Model B+ will remain in production until at least January 2023.



Physical specifications



Warnings

- This product should only be connected to an external power supply rated at 5 V/2.5 A DC. Any external power supply used with the Raspberry Pi 3 Model B+ shall comply with relevant regulations and standards applicable in the country of intended use.
- This product should be operated in a well-ventilated environment and, if used inside a case, the case should not be covered.
- Whilst in use, this product should be placed on a stable, flat, non-conductive surface and should not be contacted by conductive items.
- The connection of incompatible devices to the GPIO connection may affect compliance, result in damage to the unit, and invalidate the warranty.
- All peripherals used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met. These articles include but are not limited to keyboards, monitors, and mice when used in conjunction with the Raspberry Pi.
- The cables and connectors of all peripherals used with this product must have adequate insulation so that relevant safety requirements are met.

Safety instructions

To avoid malfunction of or damage to this product, please observe the following:

- Do not expose to water or moisture, or place on a conductive surface whilst in operation.
- Do not expose to heat from any source; the Raspberry Pi 3 Model B+ is designed for reliable operation at normal ambient temperatures.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Whilst it is powered, avoid handling the printed circuit board, or only handle it by the edges to minimise the risk of electrostatic discharge damage.

ANT-868-ID

Data Sheet

Product Description

The ID Series, ½-wave center-fed Industrial Dipole antenna is a weatherized design with industrial and outdoor applications in mind, such as outdoor meters, solar panel controls and other sensor monitoring and control systems requiring a low cost - but rugged - solution. The plastic and cable are rated for a wide temperature range and UV exposure for long term reliability.

Two standard cable lengths of 1m and 2m allow the antenna to be remote mounted in a position to achieve the best RF performance. An integrated flange makes mounting the antenna simple.

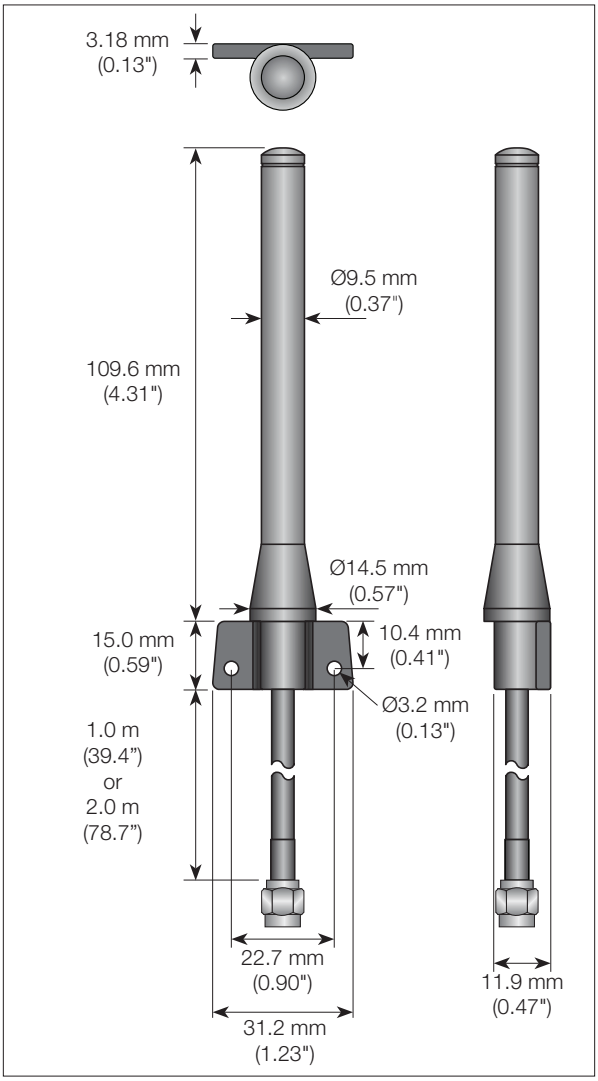
ID Series antennas attach via a standard SMA connector. Custom colors, cable lengths and connectors are available for volume OEM customers.

Features

- Weatherized assembly for outdoor installation
- UV protection
- IP-67 rating*
- Wide temperature range
- Small size
- Outstanding VSWR
- Integrated mounting flange
- Standard SMA connector

Electrical Specifications

Center Frequency:	868MHz
Recom. Freq. Range:	848–898MHz
Bandwidth:	50MHz
Wavelength:	½-wave
VSWR:	≤ 1.9 typical
Peak Gain:	0.6dBi max
Impedance:	50-ohms
Cable:	1 or 2m of RG-58/U
Connection:	SMA
Oper. Temp. Range:	–40°C to +80°C
UV Resistance:	UL 2556 section 4.2.8.5 or equivalent



Ordering Information

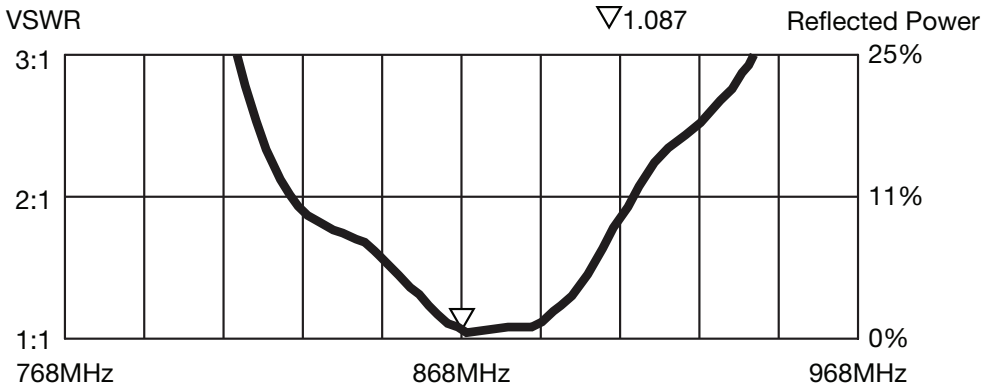
ANT-868-ID-1000-SMA (1m with SMA connector)
ANT-868-ID-2000-SMA (2m with SMA connector)
MEC-PSA-ID (Optional PSA adhesive patch)

Contact Linx for custom cable lengths.

*The IP rating applies to the antenna body only. IP ratings on the product's enclosure depend on how the mating connector is implemented.

Revised 9/13/2017

VSWR Graph



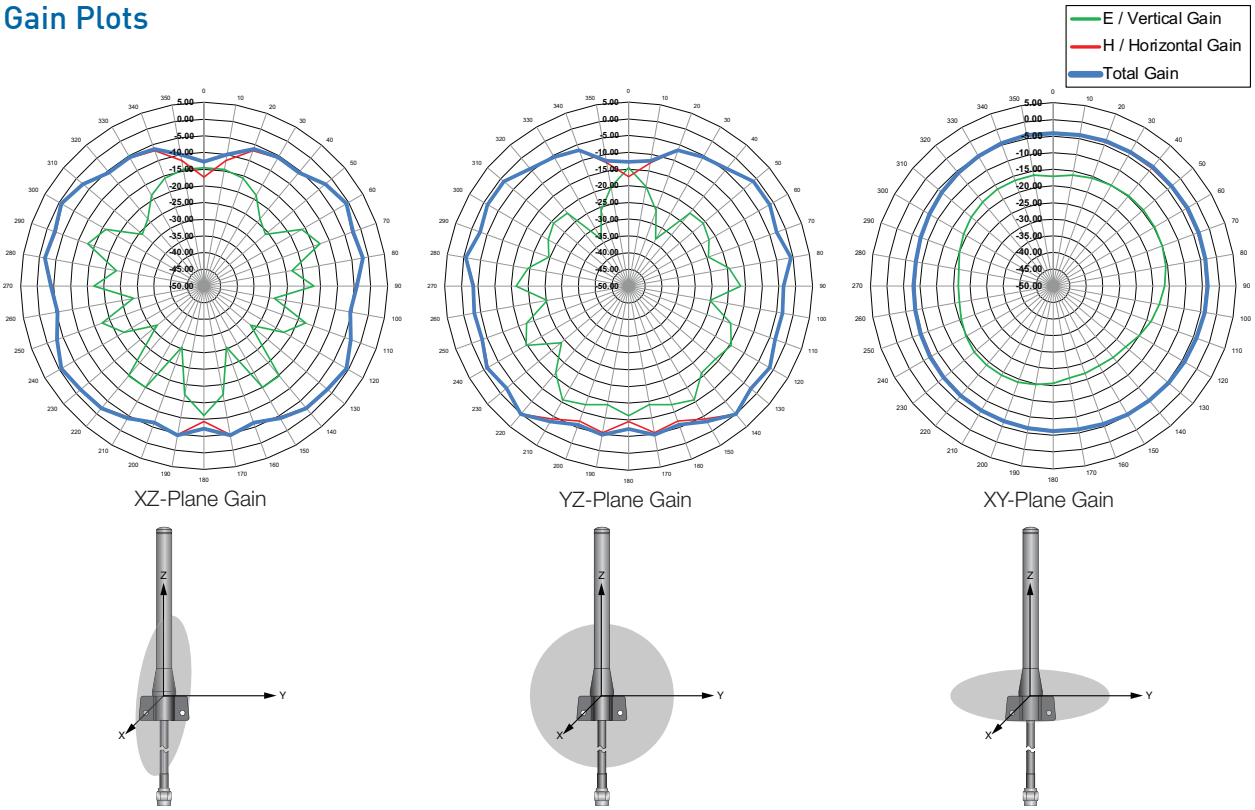
What is VSWR?

The Voltage Standing Wave Ratio (VSWR) is a measurement of how well an antenna is matched to a source impedance, typically 50-ohms. It is calculated by measuring the voltage wave that is headed toward the load versus the voltage wave that is reflected back from the load. A perfect match has a VSWR of 1:1. The higher the first number, the worse the match, and the more inefficient the system. Since a perfect match cannot ever be obtained, some benchmark for performance needs to be set. In the case of antenna VSWR, this is usually 2:1. At this point, 88.9% of the energy sent to the antenna by the transmitter is radiated into free space and 11.1% is either reflected back into the source or lost as heat on the structure of the antenna. In the other direction, 88.9% of the energy recovered by the antenna is transferred into the receiver. As a side note, since the ":1" is always implied, many data sheets will remove it and just display the first number.

How to Read a VSWR Graph

VSWR is usually displayed graphically versus frequency. The lowest point on the graph is the antenna's operational center frequency. In most cases, this is different than the designed center frequency due to fabrication tolerances. The VSWR at that point denotes how close to 50-ohms the antenna gets. Linx specifies the recommended bandwidth as the range where the typical antenna VSWR is less than 2:1.

Gain Plots

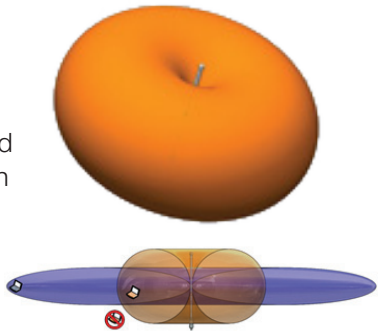


About Gain Plots

The true measure of the effectiveness of an antenna in any given application is determined by the gain and radiation pattern measurement. For antennas gain is typically measured relative to a perfect (isotropic) radiator having the same source power as the antenna under test, the units of gain in this case will be decibels isotropic (dBi). The radiation pattern is a graphical representation of signal strength measured at fixed distance from the antenna.

Gain when applied to antennas is a measure of how the antenna radiates and focuses energy into free space. Much like a flashlight focuses light from a bulb in a specific direction, antennas focus RF energy into specific directions. Gain in this sense refers to an increase in energy in one direction over others.

It should also be understood that gain is not “free”, gain above 0dBi in one direction means that there must be less gain in another direction. Pictorially this can be pictured as shown in the figures to the right. The orange pattern represents the radiation pattern for a perfect dipole antenna, which is shaped like a donut. The pattern for an omnidirectional antenna with gain is shown in blue. The gain antenna is able to work with a device located further from the center along the axis of the pattern, but not with devices closer to the center when they are off the axis – the donut has been squished.



Gain is also related to the overall physical size of the antenna, as well as surrounding materials. As the geometry of the antenna is reduced below the effective wavelength (considered an electrically small antenna) the gain decreases. Also, the relative distance between an electrically small antenna and its associated ground impacts antenna gain.

159 Ort Lane, Merlin, OR 97532
Phone: +1 541 471 6256
Fax: +1 541 471 6251
www.linxtechnologies.com

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Linx Technologies:

ANT-868-ID-2000-SMA ANT-868-ID-1000-SMA

LoPy4

The LoPy4 is a quadruple bearer MicroPython enabled development board (LoRa, Sigfox, WiFi, Bluetooth) – perfect enterprise grade IoT platform for your connected Things. With the latest Espressif chipset the LoPy4 offers a perfect combination of power, friendliness and flexibility.

Create and connect you things everywhere. Fast.

LoPy4 Features

- Powerful CPU, BLE and state of the art WiFi radio. 1KM Wifi Range
- Can also double up as a Nano LoRa gateway
- MicroPython enabled
- Fits in a standard breadboard (with headers)
- Ultra-low power usage: a fraction compared to other connected micro controllers

Processing

- Espressif ESP32 chipset
- Dual processor + WiFi radio System on Chip.
- Network processor handles the WiFi connectivity and the IPv6 stack.
- Main processor is entirely free to run the user application.
- An extra ULP-coprocessor that can monitor GPIOs, the ADC channels and control most of the internal peripherals during deep-sleep mode while only consuming 25uA.

LoRa Operating Frequencies

- 868 MHz (Europe) at +14dBm maximum
- 915 MHz (North and South America, Australia and New Zealand) at +20dBm maximum
- 433 MHz (Europe) at +10dBm maximum
- 470 - 510 MHz (China) at +14dBm maximum

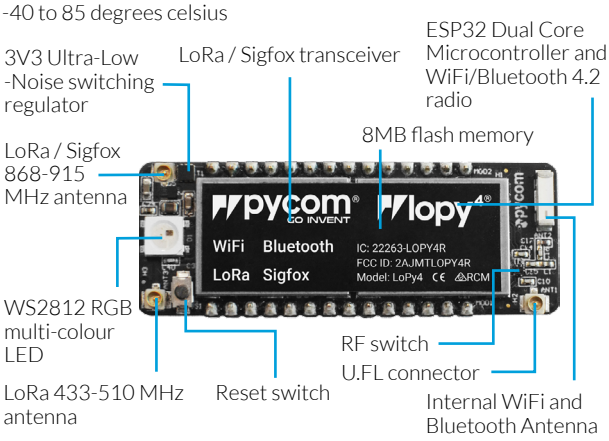
LoRa Specifiction

- Semtech LoRa transceiver SX1276
- LoRaWAN stack
- Class A and C devices

Mechanical

Size: 55mm x 20mm x 3.5mm

Operating temperature:
-40 to 85 degrees celsius



Sigfox Operating Frequencies

- RCZ1 - 868MHz (Europe)
- RCZ2 - 902MHz (US, Canada and Mexico)
- RCZ3 - (Japan and Korea)
- RCZ4 - 920 - 922 MHz (ANZ, Latin America and S-E Asia)

Sigfox Specification

- Class 0 device. Maximum Tx power:
+14dBm (Europe)
+20dBm (America)
+20dBm (Australia and New Zealand)
- Node range: Up to 50km
- Sigfox certified

Lora Range Specifiction

- Node range: Up to 40km
- Nano-Gateway: Up to 22km
- Nano-Gateway Capacity: Up to 100 nodes

Distributed and manufactured by Pycom Ltd.
Registered office:
High Point, 9 Sydenham Road, Guildford, Surrey GU1 3RX, UK

To order contact **sales@pycom.io**

pycom GO INVENT

LoPy4®

Interfaces

- 2 x UART, SPI, 2 x I2C, I2S, micro SD card
- Analog channels: 8x12 bit ADCs
- Timers: 4x16 bit with PWM and input capture
- DMA on all peripherals
- GPIO: Up to 24

Power

– Input: 3.3V – 5.5V
– 3v3 output capable of sourcing up to 400mA
– WiFi: 12mA in active mode, 5uA in standby
– LoRa: 15mA in active mode, 1-uA in standby
– Sigfox (Europe): 12mA in Rx mode, 42mA in Tx mode and 0.5uA in standby
– Sigfox (Australia, New Zealand and South America): 12mA in Rx mode, 120 mA in Tx mode and 0.5uA in standby

Security & Certifications

- SSL/TLS support
- WPA Enterprise security
- FCC - 2AJMTLOPY4R
- CE 0700

Use the Pymakr Plugins

Plugins for popular code editors to write your MicroPython scripts

Quick Verification

For easy and fast debugging use the interactive shell that is accessible through telnet or one of the serial ports

Easy Upload

Upload your scripts, and any other files you want to the LoPy4 via the FTP server

Locally or remotely

Reset the LoPy4 (you can do it locally, or remotely via Telnet)

Hash / encryption

SHA, MD5, DES, AES

WiFi

802.11b/g/n 16mbps

Bluetooth

Low energy and classic

RTC

Running at 32KHz

Memory

- RAM: 4MB
- External flash 8MB
- Hardware floating point acceleration
- Python multi-threading

With dozens of ready to use templates and libraries soon to be available on the Pycom Exchange, developing a new IoT solution is now easier and faster.

Distributed and manufactured by Pycom Ltd.
Registered office:
High Point, 9 Sydenham Road, Guildford, Surrey GU1 3RX, UK

To order contact **sales@pycom.io**

EU Regulatory Conformance

Hereby, Pycom Ltd declares that this device is in compliance with the essential requirements and other relevant provisions of Directive 1999/5/EC.

Federal Communication Commission Interference Statement

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) This device may not cause harmful interference.
(2) This device must accept any interference received, including interference that may cause undesired operation.

CAUTION: Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

NOTE: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

RF Warning Statement

To comply with FCC RF exposure compliance requirements, the antennas used for this transmitter must be installed to provide a separation distance of at least 20 cm from all persons and must not be co-located or operating in conjunction with any other antenna or transmitter.
This device is intended only for OEM integrators under the following conditions:
1) The antenna must be installed such that 20 cm is maintained between the antenna and users, and
2) The transmitter module may not be co-located with any other transmitter or antenna.

As long as two conditions above are met, further transmitter test will not be required.

However, the OEM integrator is still responsible for testing their end-product for any additional compliance requirements required with this module installed. To ensure compliance with all non-transmitter functions the host manufacturer is responsible for ensuring compliance with the module(s) installed and fully operational. For example, if a host was previously authorized as an unintentional radiator under the Declaration of Conformity procedure without a transmitter certified module and a module is added, the host manufacturer is responsible for ensuring that the after the module is installed and operational the host continues to be compliant with the Part 15B unintentional radiator requirements

The module is limited to OEM installation ONLY. The module is limited to installation in mobile or fixed application. We hereby acknowledge our responsibility to provide guidance to the host manufacturer in the event that they require assistance for ensuring compliance with the Part 15 Subpart B requirements.

IMPORTANT NOTE: In the event that these conditions cannot be met (for example certain laptop configurations or co-location with another transmitter), then the FCC authorization is no longer considered valid and the FCC ID cannot be used on the final product. In these circumstances, the OEM integrator will be responsible for reevaluating the end product(including the transmitter) and obtaining a separate FCC authorization.

End Product Labeling

This transmitter module is authorized only for use in device where the antenna may be installed such that 20 cm may be maintained between the antenna and users. The final end product must be labeled in a visible area with the following: “Contains FCC ID: 2AJMTLOPY4R”. The grantee's FCC ID can be used only when all FCC compliance requirements are met.

The following FCC part 15.19 statement has to also be available on the label:

This device complies with Part 15 of FCC rules. Operation is subject to the following two conditions:
(1) this device may not cause harmful interference and
(2) this device must accept any interference received, including interference that may cause undesired operation.

Manual Information to the End User

The OEM integrator has to be aware not to provide information to the end user regarding how to install or remove this RF module in the user’s manual of the end product which integrates this module.

In the user manual of the end product, the end user has to be informed that the equipment complies with FCC radio-frequency exposure guidelines set forth for an uncontrolled environment.

The end user has to also be informed that any changes or modifications not expressly approved by the manufacturer could void the user's authority to operate this equipment.

The end user manual shall include all required regulatory information/warning as show in this manual.

The maximum operating ambient temperature of the equipment declared by the manufacturer is
-20~+85C

Receiver category 3

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Pycom:

LoPy4 LoPy 4.0

Pysense

Imagine a shield which you can use with any of your Pycom multi-network modules. This is what we give you with Pysense and Pytrack – a leg up to get you off to a great start really quickly. These little fellows fit all Pycom boards and each contain a number of sensors.

Pysense Features

- Ambient light sensor
- Barometric pressure sensor
- Humidity
- 3 axis 12-bit accelerometer
- Temperature sensor
- A USB port with serial access
- LiPo battery charger
- MicroSD card compatibility
- Ultra low power operation (~1uA in deep sleep)

Location services supported

- Galileo
- QZSS

Use with Pybytes

Crafted by a group of Pycom back-end experts the Pybytes platform will be everything there needs to be in place to gather, organise and display your data FREE OF CHARGE in the cloud. Regardless of whether you are a company, IoT system architect, or an ambitious tech hobbyist we'll soon get you connected.

With dozens of ready to use templates and libraries soon to be available on the Pycom Exchange, developing a new IoT solution is now easier and faster.

Use the Pymakr IDE

Super easy code editor to write your Python scripts

Easy Upload

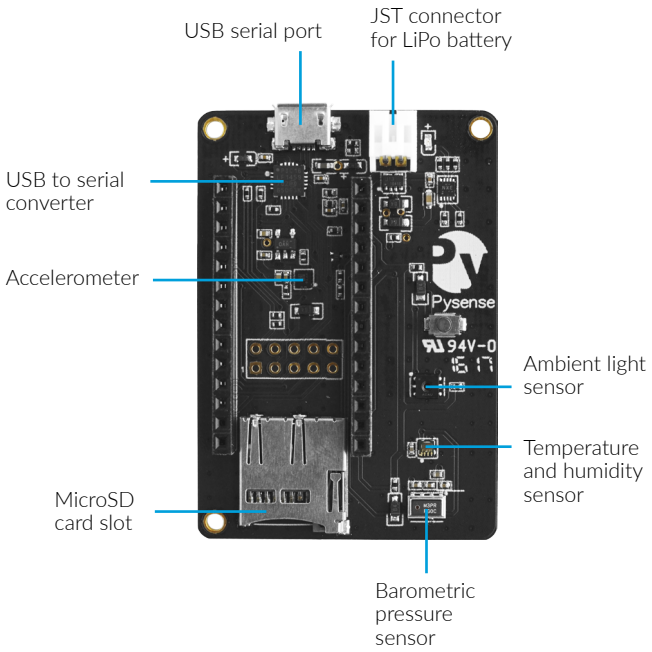
Upload your scripts, and any other files you want to the GPy via the FTP server

Locally or remotely

Reset the GPy (you can do it locally, or remotely via Telnet)

Mechanical

Size: 55mm x 35mm x 10mm
Operating Temperature: -40 to +85 Celsius
Weight: 50g



Distributed and manufactured by Pycom Ltd.
Registered office:
High Point, 9 Sydenham Road, Guildford, Surrey GU1 3RX, UK

To order contact **sales@pycom.io**

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Pycom:
Pysense

WWW.PHIHONG.COM

30W Single Port Power over Ethernet Midspan IEEE802.3at Compliant Power Injector

Features

- Compliant with IEEE802.3at Standard
- Non-vented Case
- Full Protection OVP, OCP
- 1 year warranty
- Limited Power Source
- Gigabit Compatible
- Shielded RJ45

Applications

- IP Telephones
- Wireless Access Points
- Bluetooth® Access Points
- Security Cameras
- IP Print Servers
- WiMAX® Access Points

Safety Approvals

- cUL/UL(60950)/(62368-1)
- CB
- CCC
- BSMI
- CU(EAC)
- IRAM(TUV-s)
- CE(IEC60950-1)/(IEC62368-1)
- RCM
- NOM/NYCE
- PSB
- KC+KCC
- EAC

Mechanical Characteristics

- Length: 140mm (5.51in.)
- Width: 65mm (2.55in.)
- Height: 36mm (1.42in.)
- Weight: 0.22Kg (7.76oz)

Output Specifications

Model	AC Input	DC Output Voltage	Load		Regulation	
			Min.	Max.	Line	Load
POE29U-1AT(PL)D-R	Three Wire	56V	10mA	536mA	54-57V	

INPUT:

AC Input Voltage Rating
100 to 240VAC

AC Input Voltage Range
90 to 264VAC

AC Input Current
0.8A (RMS) maximum at 120VAC
0.55A (RMS) maximum at 240VAC

AC Input Frequency
47 to 63Hz

Leakage Current
3.5mA max

Max In-rush Current
60A max at 240VAC and max load (Cold Start at ambient 25°C)

OUTPUT:

Output Power
30W

Hold up Time
10mSec min at max load at 120VAC, 60Hz

Efficiency
80% Typical at max load at 120VAC, 60Hz

ENVIRONMENTAL:

Temperature

Operating	-20°C to +50°C
Non-operating	-20°C to +70°C
Relative Humidity	5 to 90%

EMI
Complies with EN55032 Class B
Complies with FCC part 15 Class B

EMC

ESD:	IEC61000-4-2 , Level 3
RS:	IEC61000-4-3, Level 3
EFT/Burst:	IEC61000-4-4, Level 2
Surge:	IEC61000-4-5, Level 3
CS:	IEC61000-4-6 , Level 3
Voltage Dips:	IEC61000-4-11
Harmonic:	IEC61000-3-2, Class A

Isolation (HI-POT)
Primary to secondary: 1500VAC for 1 minute, 10mA

Insulation Resistance
Primary to secondary: 10M Ohm 500VDC

FEATURES:

OCP/OVP/Short Circuit Protection
OCP: ≤750mA Max
OVP: Latching
Short Circuit: The output can be shorted permanently without damage. Non-latching auto recovery.

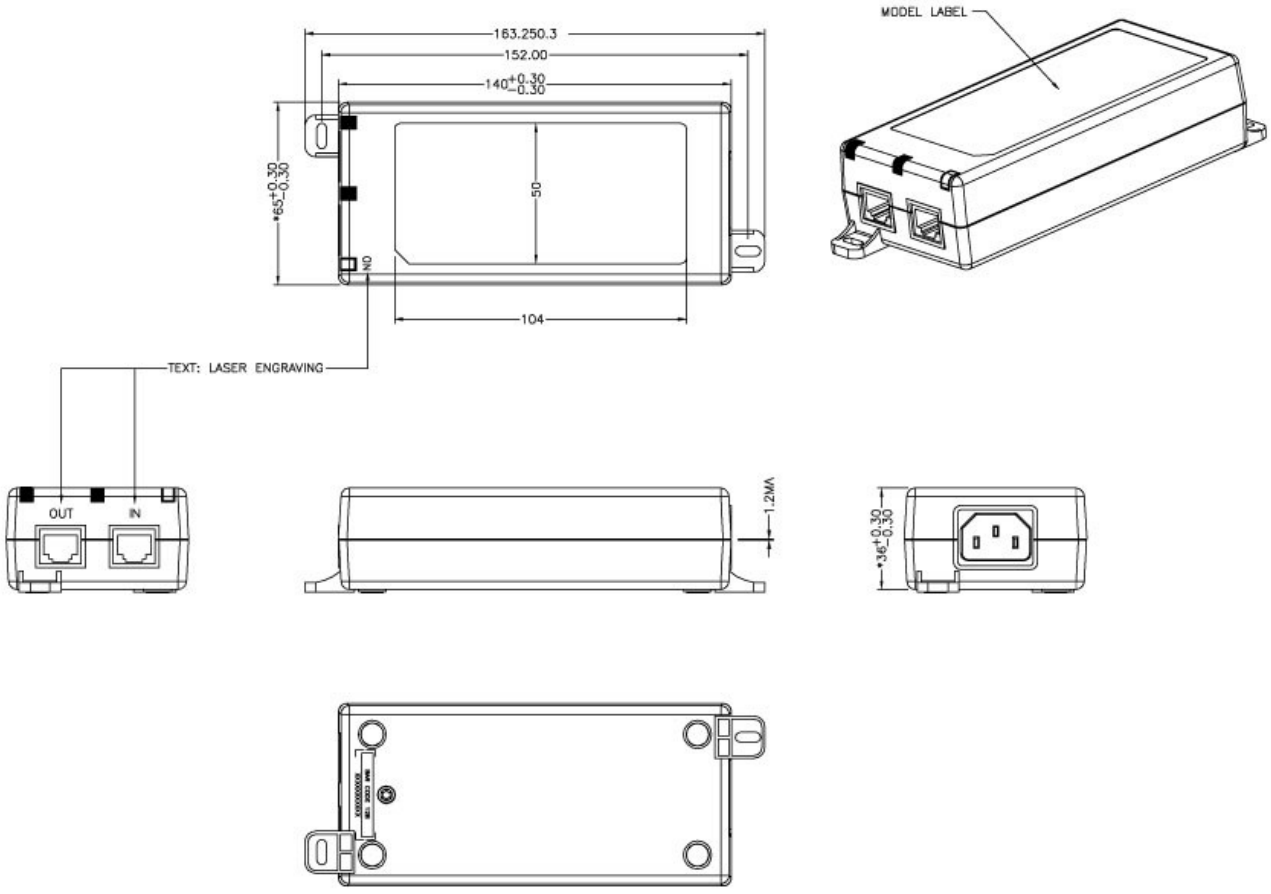
Indicator
Solid Green: unit has detected a IEEE802.3at or af load/PD
Blinking Green: Power "ON" ready for connection
Fast Blinking Amber/Green: Invalid load connected
Slow Blinking Amber/Green: fault

AC Input Connector
IEC320 C14 - three wire

Output Connection
+pins 3,6 / -pins 1,2

Data IN/POE OUT Connector
Shielded RJ45

Dimensional Diagram Unit: mm WWW.PHIHONG.COM



Supplier's Declaration of Conformity
47 CFR § 2.1077 Compliance Information

Phihong USA Corporation
47800 Fremont Boulevard
Fremont, CA 94538
Telephone: (510) 445-0100
www.phihong.com

NOTE: This model has/The models in this products series have been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

—Reorient or relocate the receiving antenna.
—Increase the separation between the equipment and receiver.
—Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
—Consult the dealer or an experienced radio/TV technician for help.

Changes or modifications to equipment not expressly approved by PHIHONG could void the user's authority to operate the equipment.

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Phihong:
POE29U-1AT(PL)D