UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politècnica de València

Trabajo Fin de Grado

Ingeniería en Electrónica Industrial y Automática

MONITORIZACIÓN DEL PATRIMONIO CULTURAL MEDIANTE TECNOLOGÍA IoT IQRF

Documentos:

1. Memoria
2. Planos
3. Pliego de condiciones
4. Presupuesto
5. Anexos

Autor: D. Miguel Andrés Sánchez Galdón

Tutor: D. Ángel Perles Ivars

Fecha: Valencia, julio 2019

Gracias a mi tutor Ángel Perles Ivars

por su ayuda y cercanía en la elaboración de este proyecto.

A mi familia por su constante apoyo y comprensión.

RESUMEN

En una era en la que las nuevas tecnologías mejoran día tras día nuestras vidas, uno de los campos más prometedores es el del Internet de las Cosas. Este campo pretende cambiar, automatizar y simplicar muchisimas aplicaciones y actividades actuales, revolucionando el mundo que conocemos hoy en día, desde la seguridad del hogar y la domótica, hasta redes de sensores y actuadores que abarcan ciudades.

Este proyecto se centra en la monitorización ambiental de las obras de arte y otros bienes patrimoniales, permitiendo una adecuada conservación de las mismas, con el fin de garantizar su perdurabilidad en el tiempo.

Concretamente, se desarrolla una red de sensores basada en la tecnología de comunicación IQRF que recogerá datos de temperatura y los enviará a la Nube para su posterior procesamiento.

En este proyecto se diseñan los nodos y se configuran para poder trabajar en conjunto con los servicios web de IBM.

Palabras clave: Conservación, IoT, IQRF, Patrimonio cultural, Sensor inalámbrico.

ABSTRACT

In an era in which new technologies improve our lives day after day, one of the most promising fields is the Internet of Things one. This field aims to change, automate and simplify many current applications and activities, revolutionizing the world we know today, from home security and automation, to networks of sensors and actuators that cover cities.

This project focuses on the environmental monitoring of works of art and other heritage assets, allowing an adequate conservation of them, in order to ensure their durability over time.

Specifically, a sensor network based on IQRF communication technology has been developed, which collects temperature data and send it to the Cloud for further processing.

In this project the nodes are designed and configured to work in conjunction with the IBM web services.

Keywords: Conservation, IoT, IQRF, Cultural heritage, Wireless sensor.

RESUM

En una era en la que les noves tecnologies milloren dia a dia les nostres vides, un dels camps més prometedors es el del Internet de les Coses. Aquest camp pretén canviar, automatitzar i simplificar moltíssimes aplicacions i activitats actuals, revolucionant el món que coneixem avui, des de la seguretat en la llar i la domòtica, fins les xarxes de sensors i actuadors que engloben les ciutats.

Aquest projecte es centra en la monitorització ambiental de les obres d'art i altres bens patrimonials, permitint una adequada conservació de les mateixes, amb la finalitat de garantitzar la seua permanència en el temps.

Concretament, es desenvolupa una xarxa de sensors basada en la tecnología de comunicació IQRF que recollirà dades de temperatura i els enviarà al Núvol per al seu posterior processament.

En aquest projecte es disenyen els nodes i es configuren per a poder treballar en conjunt amb els serveis web d'IBM.

Paraules clau: Conservació, IoT, IQRF, Patrimoni cultural, Sensor Inalàmbric.

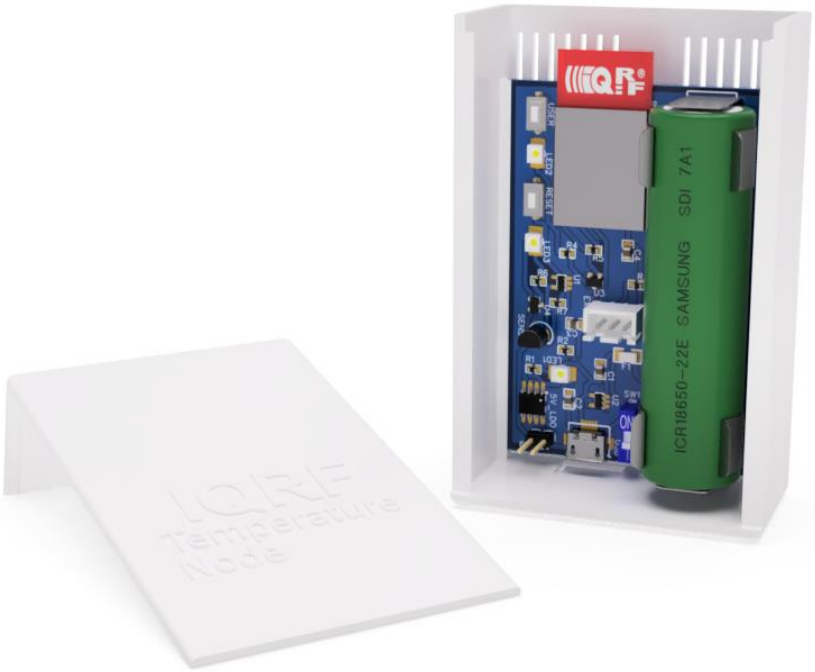
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

MONITORIZACIÓN DEL PATRIMONIO CULTURAL

MEDIANTE TECNOLOGÍA IOT IQRF

1. Memoria



Autor: D. Miguel Andrés Sánchez Galdón

Tutor: D. Ángel Perles Ivars

Valencia, julio de 2019

ÍNDICE DE LA MEMORIA

GLOSARIO DE SIGLAS Y ABREVIATURAS

ADC	Conversor analógico a digital
API	Interfaz de programación de aplicaciones
BLE	Bluetooth de baja energía
CSS	Chirp Spread Spectrum
CPI	Ciudad Politécnica de la Innovación
D-BPSK	Differential Binary Phase Shift Keying
D0	Pin digital 0
DPA	Direct Peripheral Access
FTDI	Future Technology Devices International
GND	Ground (0V)
HTTP	Hypertext Transfer Protocol
HWPID	Hardware Profile Identification
I2C	Circuito inter-integrado
IC	Circuito integrado
ID	Número de identificación
IDE	Entorno de desarrollo integrado
IEE	Institute of Electrical and Electronics Engineers
IOT	Internet de las cosas
IP	Protocolo de Internet
ISM	Industrial, Scientific and Medical bands
JSON	Notación para objetos JavaScript
LAN	Local Access Network
MCU	Microcontrolador
MQ	Message Queuing
MQTT	Message Queuing Telemetry Transport
NADR	Node Adress
PCB	Tarjeta de circuito impreso
PCMD	Profile Command
PNUM	Peripheral Number
RTD	Resistance Temperature Detector
RX	Pin de recepción UART
SIG	Special Interest Group
SD	Secure Digital
SMD	Tecnología de montaje superficial
SPI	Serial Peripheral Interface
SSL	Secure Sockets Layer
TLS	Transport Layer Security
TR	Radiotransmisor
TX	Pin de emisión UART
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Datagram Protocol
USB	Bus Serie Universal
VCC	Voltaje común de colector (alimentación positiva)
VPN	Virtual Private Network
WPAN	Wireless Personal Area Network

ÍNDICE DE FIGURAS

1. OBJETO DEL PROYECTO

Este proyecto tiene como finalidad monitorizar los valores de determinadas magnitudes ambientales (humedad, temperatura, etc.) de las zonas en las que se encuentran obras de arte y otros bienes considerados de valor patrimonial, que requieren unas condiciones determinadas para su adecuada conservación. Recogiendo todos estos datos en un solo lugar, se facilitará a los expertos su correcto mantenimiento y protección frente a los agentes externos y el paso del tiempo, evitando la degradación natural a la que se enfrentan.

2. ANTECEDENTES

El proyecto surge de la necesidad de controlar las variables ambientales de algunos edificios del campus de la UPV, en los cuales se recogen algunos bienes patrimoniales de importancia. También cabe recalcar la importancia de evitar el coste de las restauraciones, ya que "por cada euro que se destina a conservación, se ahorran entre 3 y 5 euros de restauración" [1].

El campo del mantenimiento del patrimonio puede verse muy favorecido gracias al desarrollo actual del IoT (Internet of Things) y otras tecnologías relacionadas, cuyo crecimiento está en continuo auge, y facilita las labores de conservación que anteriormente han sido más laboriosas y, por lo tanto, más costosas. El término Internet of Things se utiliza de manera pública por primera vez en 2009, por Kevin Ashton, profesor del MIT, aunque él mismo asegura que internamente se utilizaba el término desde 1999[2].

En el entorno de la conservación de elementos históricos, algunas empresas y fundaciones ya acogen al término "Smart Patrimonio", que pretende monitorizar los bienes de patrimonio cultural mediante nuevas tecnologías, apostando por la conservación preventiva[3].

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

Dos de los factores ambientales principales que afectan al patrimonio son la temperatura y la humedad del lugar en el que se encuentran. Estos condicionantes puede nser determinante para la conservación de estos objetos y, por lo tanto, deben ser factores a controlar. Aquí es donde entra la tecnología que, mediante un sistema de sensorización puede ayudar a evitar la pérdida de los elementos históricos.

La aplicación concreta se centra en dar visibilidad a nuevas tecnologías para este propósito, en el cual priman los entornos de interior donde se recogen distintas obras de arte y otros elementos históricos. Por ello, se estudia la implementación de un sistema modular que tome medidas de temperatura en puntos a corta distancia, asegurando que todas las zonas se encuentran en unas condiciones favorables.

Dicho sistema de sensorización debe tomar medidas en distintas zonas para asegurar que la temperatura no varíe en las distintas salas o plantas del edificio, con el fin de poder mantener los elementos almacenados en un entorno lo más adecuado posible para su apropiada conservación. Por este motivo, se necesitan diferentes dispositivos IoT interconectados que proporcionen valores de temperatura y los transmitan a un lugar común.

En este tipo de sistemas, el entorno arquitectónico cobra una gran importancia, en especial las distancias y los obstáculos. Para todo sistema inalámbrico de interior, un factor muy importante es la construcción del propio edificio, más incluso que las distancias, ya que puede impedir las comunicaciones, debido a las dimensiones de los muros y sus materiales.

3.1. Medición de temperatura

La temperatura es la principal magnitud a considerar, ya que es la variable a controlar en este proyecto. En este caso específico, intentamos mantener una temperatura adecuada en un nivel relativamente cercano al del entorno colindante, por lo que las variaciones de temperatura serán fácilmente previsibles y sin cambios abruptos. Al tratarse de un valor cotidiano, la mayoría de sensores comerciales abarcarán dicha temperatura en su rango de especificaciones, por lo que no será difícil encontrar un sensor apropiado ni se tendrá que recurrir a sensores específicos para una actividad determinada. Para la conservación del patrimonio, la práctica actual consiste en mantener las condiciones ambientales históricas[4].

3.2. Medición de la humedad

La humedad es el otro factor principal que afecta a la conservación del patrimonio. La humedad afecta a desarrollo de reacciones químicas que pueden degradar los objetos encontrados en las instalaciones, entorpeciendo así su adecuado mantenimiento. De la mano de la temperatura, la humedad puede reducir mucho el tiempo que se pueden conservar los elementos históricos, tanto en interior como en exterior. Como prueba de concepto, en este trabajo no se medirá la humedad, siendo siempre ampliable en un futuro.

3.3. Distancias y obstáculos

En este tipo de aplicaciones partimos del supuesto de utilizar comunicaciones inalámbricas, ya que el hecho de situar cableado por cada sensor aumentaría el coste y el impacto en el lugar designado para la instalación.

Partiendo de éste punto, debemos considerar como otro factor de importancia las distancias y obstáculos que se encuentran en el entorno y que pueden repercutir en la difusión de las señales inalámbricas emitidas por los dispositivos y sensores del sistema. Dadas las estructuras y su construcción, deberemos elegir una tecnología adecuada para que no haya interrupciones del servicio.

3.4. Consumo y autonomía

Otro factor a tener en cuenta es el consumo de los elementos de la red, ya que pueden ser representativos en los costes de mantenimiento. Para este tipo de redes siempre se tiende a evitar el cableado, por lo que entra en juego el concepto de autonomía. Destaca la importancia del consumo en cada uno de los elementos del sistema ya que su alimentación deberá ser mayoritariamente por medio de baterías, y según este consumo, la autonomía variará de manera proporcionalmente inversa.

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

Determinada el objetivo, se procede a estudiar las posibles soluciones que se pueden emplear para conseguir un sistema adecuado de monitorización de la temperatura.

A continuación, se detallan las diferentes alternativas planteadas, con sus puntos fuertes y sus debilidades. Se plantearán primero los posibles sensores a utilizar, seguido de los microcontroladores considerados, las tecnologías de comunicación posibles y la manera de visualizar los datos recogidos.

4.1. Sensores de temperatura

Para este proyecto se deben utilizar sensores de temperatura ya que el fin último es conseguir monitorizar esta magnitud física. Hay muchos sensores en el mercado que pueden utilizarse en este tipo de aplicaciones, por lo que se considerarán únicamente aquellos en cuyas especificaciones entren las necesidades del proyecto, como el rango de temperatura, la resolución de las medidas, el retardo de medición y el consumo energético. Éste último será el más importante. Caben destacar dos tipos de sensores de temperatura, analógicos y digitales, cuyas diferencias se explicarán en los apartados posteriores.

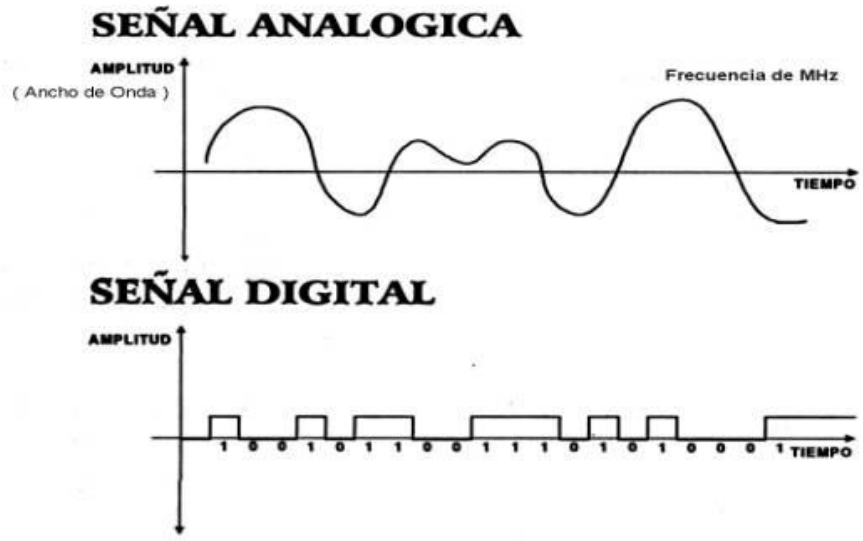


Ilustración 1 - Señal analógica y señal digital

4.1.1. Sensores analógicos

Los sensores de temperatura analógicos son aquellos que transforman un valor de temperatura en un nivel de voltaje y habitualmente de forma lineal. El voltaje proporcionado por este tipo de sensores suele ser del orden de mV por grado centígrado, con lo cual a veces es necesario amplificar la señal de salida para poder procesarla de una manera más precisa, además de requerir la eliminación del offset. Otro inconveniente que suelen presentar un consumo mayor que los digitales y en su uso en microcontroladores es necesario un conversor analógico-digital (ADC), aunque a menudo va incluido en el microcontrolador.

La conversión analógico-digital también supone un retardo que dependerá del propio microcontrolador. Existen diferentes subtipos, como termopares, termistores, RTD y los integrados en chip.

Los termopares son un tipo de sensores muy utilizados en actividades industriales, ya que resisten temperaturas muy altas. Se componen de dos metales de diferente naturaleza y que al aplicarle una temperatura crean una diferencia de potencial.

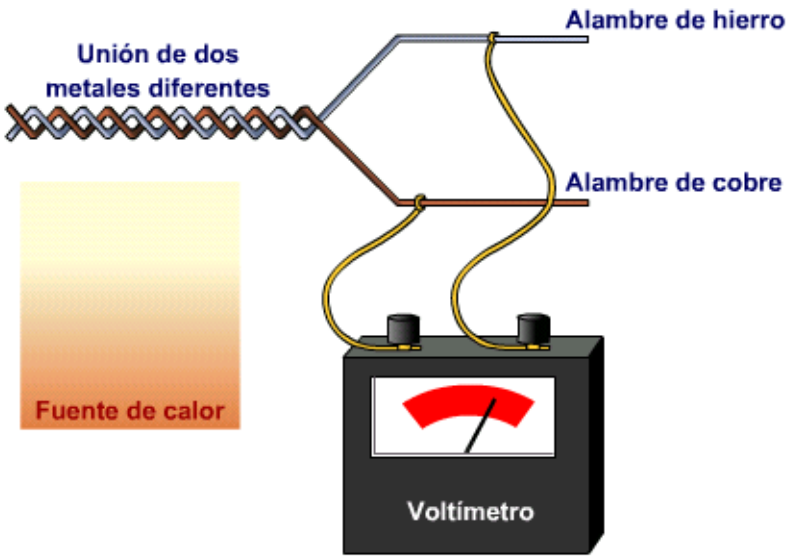


Ilustración 2 - Termopar

Proporcionan señales del orden de microvoltios, con lo cual su precisión es difícil de interpretar en un microcontrolador con ADC integrado.

Por otro lado, los termistores y los RTD poseen una resistencia interna que varía en función de la temperatura, por lo que pueden presentar consumos altos a determinadas temperaturas, lo cual es un factor que los discrimina.

Ilustración 3 - Termistores

Sin duda los más utilizados son los integrados en chip gracias a su rango de temperatura, su tamaño y la linealidad de su salida. El más usado de estos sensores es el sensor LM35.

4.1.1.1. Sensor LM35

Ilustración 4 - Sensor LM35

El LM35 es un sensor de temperatura analógico muy extendido. Tiene un precio relativamente bajo y linealidad en su salida en casi todo su rango de medición (de −55°C a 150°C). Su salida es de 10mV/ºC y tiene una precisión de ±0.2ºC. Por contra, su uso en temperaturas negativas requiere añadir una resistencia conectada a un voltaje negativo, además de requerirse más de 4V para alimentación, lo que impide su uso en circuitos de muy baja potencia. A esto se añade la necesidad de, como todo sensor analógico, utilizar un conversor analógico-digital a la salida.

4.1.2. Sensores digitales

Los avances en la integración de la tecnología han dado paso a que los nuevos sistemas electrónicos incorporen en circuitos integrados más funciones en tamaños más reducidos, reduciendo el volumen, así como su consumo. Así es como los sensores digitales se han abierto paso en el progreso de la tecnología. Este tipo de sensores hacen que no sea necesario convertir posteriormente la señal analógica a digital, ya que proporcionan directamente a su salida una cadena de bits que, mediante distintos protocolos, se pueden adquirir desde un microcontrolador[5]. Las cadenas de bits suelen

tener una longitud de entre 8 y 12 bits. Estos sensores no requieren ningún tipo de acondicionamiento de la señal.

4.1.2.1. Maxim DS18B20

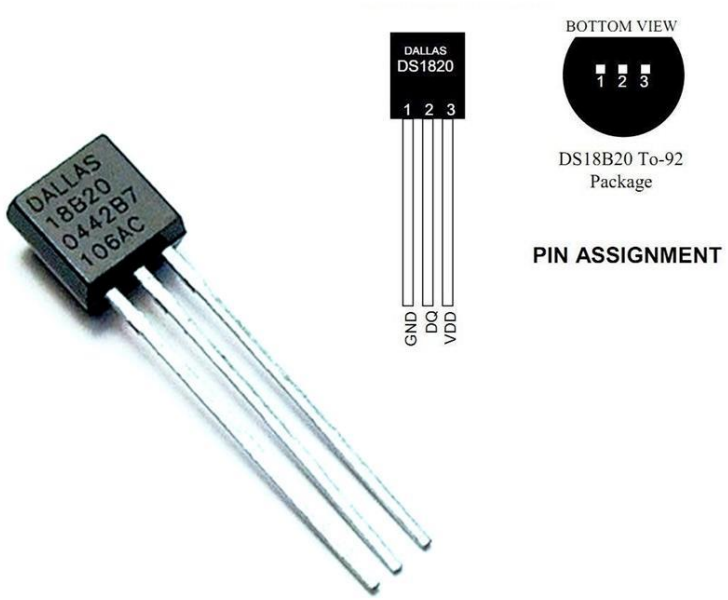


Ilustración 5 - Sensor DS18B20

El DS18B20 de Maxim Integrated es uno de los sensores digitales más usados en el mercado. Esto se debe a su bajo consumo, su fácil integración en un circuito reducido y su gran resolución, del orden de 0,05ºC. Su rango de temperatura va desde -55ºC hasta 125ºC, cubriendo sobradamente las necesidades de este proyecto. Este sensor puede configurarse para trabajar con diferentes resoluciones (9, 10 11 y 12 bits), lo que permite ajustar la resolución de las mediciones.

Una de las principales ventajas de este sensor de temperatura es la posibilidad de alimentarse desde 3V hasta 5,5V, lo que lo hace muy atractivo para su uso con placas que trabajan a 3,3V, como las basadas en ESP o algunas Arduino. El uso de un voltaje menor se traduce en un consumo energético más reducido, lo que es importante en este tipo de aplicaciones. Según el fabricante, se contemplan consumos de 750nA en modo Standby, es decir, cuando no se realiza una medición. Realizando mediciones los consumos rondan 1mA[6].

Además, el DS18B20 se proporciona en diferentes formatos, tanto para integración en placa PCB como el típico T0-92, que además puede comercializarse en formato sonda impermeable, permitiendo su uso en aplicaciones exteriores o sumergidas en fluidos.

La configuración con un microcontrolador se realiza mediante el protocolo 1-Wire, que puede ser ligeramente complicado en su programación. Sin embargo, el sensor está tan extendido en el mercado que existen librerías para la mayoría de microcontroladores y plataformas, lo que facilita mucho su uso en microcontroladores.

4.2. Microcontroladores

4.2.1. Arduino

Ilustración 6 - Placas Arduino UNO, MEGA y Nano

Arduino es la placa de prototipado basada en microcontrolador más extendida y utilizada en el mundo[7]. Ello proporciona mucho soporte por parte de la firma y de la comunidad, además de unos costes más reducidos y un gran catálogo de periféricos y sensores. Este soporte ha producido una extensión de su Entorno de desarrollo, Arduino IDE, a otras plataformas, manteniendo el estilo de programación. Sin duda es una solución muy sencilla que proporciona distintas maneras de utilizarse para este caso. Se basa en varios microcontroladores diferentes, aunque sin duda el más utilizado es el Atmel328p, cuyo consumo es muy competitivo y viene en diferentes formatos, además de existir la posibilidad en la mayoría de placas Arduino de alimentarse a 5V o a 3,3V, lo que puede reducir el consumo notablemente.

La inmensa mayoría de sensores se pueden utilizar con este microcontrolador de una manera sencilla, gracias a sus librerías públicas, por lo que el factor más importante de ésta solución será el tipo de tecnología que se utilizará.

Además, Arduino tiene la vista puesta en las tecnologías IoT, por lo que ya posee una gama amplia de placas de bajo consumo destinadas a este tipo de aplicaciones, implementando protocolos de comunicación como LoRa[8].

4.2.2. Espressif ESP

Ilustración 7 - Módulo ESP-12

Los microcontroladores ESP de Espressif se han abierto paso en el mercado cotidiano en los últimos años, en gran parte por su coste reducido y su facilidad a la hora de configurar con sensores. Además de estas características, destaca la inclusión de Wifi en el propio microcontrolador, lo que le aporta una importancia especial en los sistemas IoT, para los que existe una gran facilidad de configuración.

También posee una alta potencia de procesado, que permite procesar los datos en el propio microcontrolador y enviarlos directamente a la plataforma.

Esta alta potencia conlleva un alto consumo energético, el cual dificulta su uso en sistemas alimentados por baterías. Destacan los ESP8266, distribuidos en diversos formatos diferentes, tanto para prototipado como para implementación en PCBs, con las mismas facilidades de programación, y la serie ESP32, más integrados en los sistemas industriales.

Existen diversos entornos de programación para este tipo de controladores, que los convierten en una opción muy competitiva para este tipo de redes:

- Micropython es un entorno de programación para los microcontroladores ESP que destaca por su sencillez y facilidad para programar. La principal ventaja que aporta es la implementación de un lenguaje de programación muy completo como es Python en un microcontrolador, brindando la posibilidad de usar códigos más complejos y, por lo tanto, con más posibilidades. Esto añade a cada microcontrolador que posee un sensor la posibilidad de procesar los datos y, mediante Wifi, enviarlos directamente a la plataforma.
- ESPEasy es otro entorno para la serie ESP, especialmente dirigido a entornos IoT o sistemas domóticos. Destaca la rápida programación y configuración para estos entornos. Sin embargo, es uno de los sistemas operativos para microcontrolador que mayor consumo tiene y ofrece menos compatibilidades para modos de bajo consumo.
- NodeMCU Firmware: Se trata de un firmware basado en el lenguaje de programación Lua. Presenta la ventaja de la ejecución de scripts modo Python, sin necesidad de compilar, aunque eso conlleva un procesamiento más lento.
- Programación directa sin OS: Los microcontroladores ESP permiten la programación en C mediante distintas suites como Arduino IDE. Es el método más eficiente ya que trabaja directamente en lenguaje de bajo nivel. Dadas las numerosas librerías existentes, su programación se convierte en una tarea sencilla.

La combinación de un microcontrolador dotado de WiFi como los ESP y protocolos de comunicación tales como MQTT permiten que este tipo de microcontroladores se puedan utilizar en entornos IoT de diversos tipos, utilizando los estándares de comunicación WiFi como red para comunicarse.

El alto consumo de esta gama de dispositivos lo hace poco viable para una aplicación alimentada por baterías. Incluso con la mejor programación, la autonomía de los dispositivos ESP que consumen menos (ESP-03) se limita a unas horas frente a años que se mantienen otros módulos, como aquellos con tecnología ZigBee[9].

4.2.3. Otros microcontroladores

Para el estudio de este trabajo se han tenido en cuenta otros microcontroladores, tales como los PIC o STM. Estos microcontroladores destacan por su uso en aplicaciones industriales y otros sistemas embebidos. Aunque carecen de sistemas de comunicación inalámbricos de manera nativa, presentan grandes ventajas en cuanto a consumos. La empresa STM asegura un mínimo de consumo tan bajo como 170nA en el modo de menor consumo de una de sus mejores microcontroladores[10].

El proceso de adición de un sistema inalámbrico de comunicación conlleva un trabajo extra que puede no ser una solución para todas las empresas, dada su complejidad. A esto se suma una programación a más bajo nivel, que dificulta el trabajo a la hora de utilizar sensores digitales, ya que incorporan protocolos de comunicación más sofisticados.

También existen numerosas aplicaciones de microcontroladores en el ámbito de IoT que ya tienen implementados sistemas operativos base sobre los que programar, al igual que ocurre con los ESP. Un ejemplo de esto es el microcontrolador PIC que se utiliza en tecnologías como IQRF. Esta combinación permite trabajar a un lenguaje de programación de más alto nivel, además de incorporar determinados protocolos de comunicación que facilitan el uso de estos microcontroladores.

4.3. Sistemas de comunicación inalámbrica

4.3.1. WiFi

Ilustración 8 - Logo de tecnología WiFi

Cuando se habla de comunicaciones inalámbricas, la primera mencionada suele ser la conexión WiFi. Con su moderado alcance y su fácil configuración, su penetración en el mercado se ha vuelto indiscutible. Esta tecnología trabaja en la banda 2,4GHz IEEE 802.11 y recientemente se ha incorporado la banda de 5GHz, mucho menos utilizada, que consigue evitar la posibilidad de interferencias con otras señales[11].

Para el caso específico de Arduino, existen shields de apoyo con esta tecnología incorporada, a precios competitivos y en una gran cantidad de formatos. Los microcontroladores ESP incorporan de serie un módulo WiFi en su encapsulado.

Por contra, una placa Arduino con un shield WiFi o un microcontrolador ESP presentan unos consumos relativamente altos, ya que es el principal inconveniente de utilizar tecnología WiFi en dispositivos alimentados por baterías. Sin embargo, es una tecnología que encontrará un lugar en el campo de IoT, como afirma Silicon Labs, ya que se espera que en la versión WiFi 6 se implementen redes mesh[12].

Cabe destacar también la necesidad de una red WiFi extendida por toda la planta, y posiblemente de uso exclusivo para este sistema. Otro de los factores a considerar será el enrutado de cada uno de los dispositivos y su recopilación de datos, lo que puede requerir un trabajo más extenso.

4.3.2. Bluetooth

Ilustración 9 - Logo de la tecnología Bluetooth

La segunda tecnología más conocida en cuanto a lo que tecnologías inalámbricas se refiere es sin duda alguna el Bluetooth, la cual es una tecnología muy extendida, con un amplio abanico de formas y versiones e implementada en dispositivos desde teléfonos móviles hasta electrodomésticos, pasando por todo tipo de electrónica de consumo. Fue creada en 2003 por el grupo Bluetooth SIG.

Categorizada como red WPAN (Wireless Personal Area Network), utiliza la banda ISM (IEEE 802.15.1) de 2.45GHz, en la que convive con otras tecnologías. Puede evitar las interferencias gracias a un sistema que permite buscar una parte no utilizada del espectro (2.4GHz hasta 2.5GHz)[13].

Sin duda la única a tener en cuenta en este tipo de sistemas son las últimas versiones del Bluetooth, a partir de la 4.0, que incorporan Bluetooth Low Energy, ya que son las únicas que pueden considerarse en sistemas alimentados por baterías.

El BLE proporciona robustez y seguridad además de una velocidad alta (1Mbit/s) con una latencia baja(6 ms) [14], teniendo por contra las distancias de acción, que no suelen superar los 30 metros, y su alta sensibilidad a los obstáculos. Como el Wifi, se requiere un sistema de enrutado y recopilación de los datos recogidos por los sensores.

Además, su uso sin un protocolo más avanzado sólo permite conexiones en estrella, teniendo que estar conectado a un nodo central directamente. Cabe destacar la existencia de un sistema de red MESH (Bluetooth Mesh) desarrollado por Bluetooth SIG

y publicado en 2017, que permite otras topologías en la red bluetooth[15]. Sin embargo, por el momento no son muchos los dispositivos que lo incluyen.

4.3.3. Bandas ISM 916, 868 y 433 MHz

Otra posibilidad a considerar es un sistema basado en radiofrecuencia con una banda específica, como la de 433MHz. La principal ventaja de usar este tipo de comunicaciones radica en las distancias abarcables por ellas, que pueden ser mucho más altas. También cabe destacar el bajo coste de los componentes necesarios para la transmisión de datos por estas frecuencias, que se pueden utilizar sin necesidad de licencia, dentro del marco regulatorio de bandas ISM (Industrial, Scientific, and Medical).

Sin embargo, su uso exige utilizar un protocolo específico para la transmisión de datos y la autentificación de los sensores miembros de la red con el fin de mantener la seguridad de la red. Además, presenta el problema del alto uso de dispositivos en estas bandas. Otros aparatos, como mandos de garaje, utilizan estas bandas para realizar sus funciones, por lo que requerirán un sistema adecuado de modulación.

No se descarta el uso de este tipo de bandas, pero siempre en combinación con otros tipos de protocolos, con el fin de poder asegurar la red del sistema.

4.3.4. Redes Zigbee

Ilustración 10 - Logo de tecnología ZigBee

Las redes ZigBee se basan en una serie de protocolos de alto nivel de comunicación inalámbrica que se usa para la difusión de información mediante radiofrecuencia con un consumo bajo de energía. Se incluye en el grupo de trabajo IEEE 802.15.4 de redes inalámbricas de área personal (WPAN)[16].

Sus usos se destinan a aquellas aplicaciones que necesiten comunicaciones seguras con una tasa baja de envío de datos. Está considerado una de las mejores opciones para su uso con baterías. Al ser una tecnología moderna, se espera su crecimiento en el ámbito de la domótica. Las bandas de frecuencia que utiliza esta tecnología son aquellas consideradas bandas libres (ISM), lo que significa que no requieren una licencia para utilizarlas. Su velocidad de transmisión alcanza los 250Kb/s (banda de 2.4GHZ, 20Kb/s en 868MHz) con rangos de aproximadamente 70 metros[17]. Debido a esta distancia máxima de aplicación, se destina principalmente a entornos domóticos donde no priman los espacios de grandes dimensiones.

ZigBee puede crear estructuras de diferentes topologías, tales como malla, estrella o árbol. Precisa tres tipos de dispositivos para crear una red funcional:

- Coordinador ZigBee: Es el nodo principal. Envía las ordenes y se encarga de construir los caminos mediante los cuales la información va a discurrir. Este elemento es indispensable en cualquier tipo de aplicación con tecnología ZigBee
- Router ZigBee: Es el encargado de interconectar los elementos de la red para poder ejecutar un código de usuario.

- Dispositivo final o end-point ZigBee: Este dispositivo sólo se comunica con el nodo principal o coordinador y ejecuta las órdenes recibidas. Puede ser un dispositivo actuador o sensor y presenta la ventaja de poder entrar en "sleep-mode", reduciendo así su consumo energético.

Sin embargo, presenta una gran complejidad y dificultad en su uso, además de que no suele ser una opción muy económica para aplicaciones de pequeña y media escala.

4.3.5. LoRaWAN

Ilustración 11 - Logo de tecnología LoRaWAN

LoRa es una técnica de modulación patentada por Semtech Corporation que permite la transferencia de información en largas distancias con una tasa de transferencia muy baja. Usa una topología de estrella, por lo que solo un salto entre módulos es posible.

LoRaWAN es un protocolo que utiliza la tecnología LoRa y que constituye otra de las alternativas que se utilizan en el mundo IoT. Es la implementación de la tecnología LoRa en redes inalámbricas con módulos alimentados por baterías con comunicación bidireccional. Este sistema asegura una intercomunicación entre los diferentes dispositivos de una red IoT sin grandes y complejas implementaciones. Utiliza las bandas sin licencia ISM (868MHz en Europa).

Esta tecnología posee grandes ventajas respecto a las distancias abarcables y a la autonomía de dispositivos alimentado por baterías que utilizan el protocolo LoRaWAN, llegando hasta los 10 kilómetros[18]. Consigue velocidades de transferencia de entre 0,3Kb/s hasta 50kb/s[19]. La autonomía de esta tecnología llega a cifras de años con

baterías de cerca de 1000mAh, como se describe en algunos proyectos que incluyen LoraWAN[20].

También destaca por su gran seguridad, que está constantemente siendo comprobada. Posee una modulación CSS (Chirp Spread Spectrum), utilizada también en ámbitos como la investigación espacial y entornos militares, dado que consigue evitar efectivamente las interferencias. Es la primera implementación de la modulación CSS en entornos de consumidor y empresas, donde los costes son más reducidos[21].

Dado que no posee capacidades de enrutado, solo un salto es posible entre los módulos de una red LoRaWAN (one hop), lo que es un gran impedimento para la aplicación concreta del sistema de este proyecto.

4.3.6. Sigfox

Ilustración 12 - Logo de tecnología Sigfox

Dentro de las alternativas enfocadas a las redes IoT se incluye también la tecnología Sigfox. Se trata de una tecnología LPWAN, al igual que LoRa, que están destinadas a entornos tanto exteriores como interiores, y estás dos son las más destacables de este tipo de redes.

Al igual que otras tecnologías ya mencionadas, utiliza frecuencias sin licencia ISM (en Europa, la frecuencia de 868 MHz). Consigue velocidades de 100b/s y utiliza bandas ultra estrechas (100 Hz de salto) sub-GHz con 2000 canales. Por normativa europea, esta tecnología limita la emisión de paquetes a 140 al día, con una longitud de 8 bytes.

La red Sigfox utiliza una modulación D-BPSK (Differential Binary Phase Shift Keying) que aumenta su sensibilidad y hace frente al ruido de una manera efectiva[22]. Como Lora o ZigBee, necesita una estación base.

Sigfox destaca por su bajo consumo y su largo alcance[23]. El consumo de un módulo AX-SIGFOX ON es de 19mA sin emisión y de 49mA en emisión, por lo que usando 2 baterías de 1500mAh a 140 paquetes por día duraría 6,5 años[22].

Consigue unos alcances muy prometedores, de hasta 20 kilómetros entre nodo y base, y por ello se concibe para entornos exteriores con largas distancias[24]. Inicialmente se desarrolló como una comunicación de un solo sentido, pero más tarde evolucionó para conseguir transferencias de datos bidireccionales.

Posee una interfaz web y una API, que permite automatizar la administración de los dispositivos e implementar la integración de los datos de una manera más sencilla. Se pueden utilizar comandos HTTP GET o POST en esta API, con posibilidad de usar conexiones SSL[24].

Sigfox ofrece muchas facilidades a la hora de construir una red y unas grandes distancias abarcables. Sin embargo, su uso está más orientado para el exterior, mientras que este proyecto se centra en espacios interiores. Además, el requerimiento de una estación base es un factor importante a la hora de construir este tipo de sistemas. El factor negativo más importante es la restricción de las transmisiones.

4.3.7. IQRF

Ilustración 13 - Logo de tecnología IQRF

IQRF es una tecnología utilizada en IoT que destaca por su seguridad y su bajo consumo. Nace en 2004, desarrollada por MICRORISC y desde 2017 se desarrolla bajo una división de la propia compañía llamada IQRF Tech s.r.o. IQRF Tech ha conseguido año tras año desarrollar su tecnología desde su nacimiento, creando un sistema simple, seguro y de bajo consumo. Los actuales planes de esta empresa son estandarizar la tecnología para extender su uso[25].

Puede trabajar a distintas frecuencias. Dependiendo del módulo puede usar indistintamente la banda de 433MHz, 868MHz o 916MHz, lo que permite utilizarse en entornos con otras comunicaciones inalámbricos presentes sin problemas. A pesar de no ser muy conocida, su desarrollo se centra en casos específicos similares a éste.

Una de las principales ventajas de construir el sistema con esta tecnología radica la posibilidad de establecer una red en malla, utilizando nodos adyacentes para poder llegar a todos los nodos de la red, sin necesidad de conexión directa. La tecnología IQRF permite hasta 240 saltos entre nodos (hops) en la red de malla que constituyen. Esto permite evitar utilizar conexiones en estrella, pudiendo ampliar la zona de acción de la red y encontrar mejores vías de comunicación. Esta red en malla se denomina IQMESH, definida en 2005[26] como un sistema de organización mesh propio de la tecnología IQRF, que también posee, entre otros, un mecanismo que permite conectar dos o más redes (mallas) entre sí a través de un solo transceptor, que puede actuar como coordinador en una red y como nodo común en otra[27].

Además de esta singular ventaja, este sistema presenta otras como un consumo ultra bajo: 900nA en modo “standby” y 35µA máximos en recepción[28], lo que le permite trabajar con unas exigencias muy altas de consumo en dispositivos alimentados por baterías, en conjunto con potencia de radio programable (desde 1,3mW hasta 12,5mW). También permite distancias de hasta 170 metros según diversos estudios[29], aunque otros informes más recientes aseguran coberturas de hasta 500 metros[30].

Los dispositivos que utilizan esta tecnología poseen un sistema operativo propio llamado IQRF OS, y éstas son algunas de sus características:

- DPA

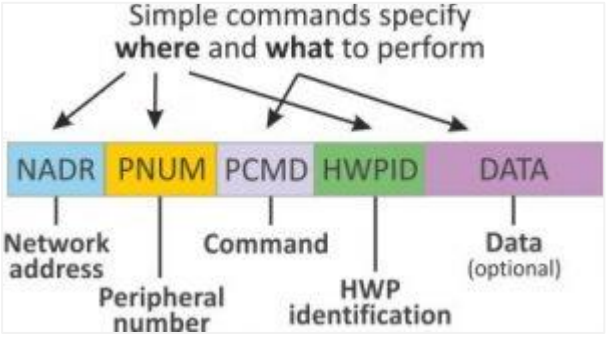


Ilustración 14 - Estructura de un paquete DPA

El DPA o Direct Peripheral Access es un protocolo perteneciente a IQRF OS[31]. Se trata de un sistema que permite mediante una serie de bytes concretos ejecutar una acción concreta en otro dispositivo de la red IQRF. Consta de cuatro partes, las cuales dirigen la acción hacia un nodo concreto, un periférico, una función y un perfil de hardware específico.

- FRC

EL FRC o Fast Response Command es un protocolo que permite obtener datos de todos los dispositivos de la red en un tiempo muy reducido. Dependiendo del número de nodos de la red IQRF, el tiempo será diferente, siempre por debajo de los 29 segundos (para 239 nodos)[32]. En colaboración con el DPA, se pueden dirigir órdenes hacia todos los nodos o a aquellos que tengan un perfil de hardware concreto. Este protocolo está integrado en el IQRF OS, el sistema operativo que se encuentra en el microcontrolador del módulo IQRF.

- IQRF y la nube

Un sistema basado en IQRF puede ser fácilmente configurado para conectarse a una plataforma en la nube, mediante el protocolo UDP, MQ o MQTT, con el fin de administrar y monitorizar los datos a distancia. Mediante una configuración sencilla con interfaz se pueden utilizar los servicios de Amazon Web Services, IBM Cloud o Microsoft Azure, que ya poseen soporte oficial en los sistemas IQRF[33].

4.4. Administración y servicios web

Una parte importante del proyecto es la administración del sistema y la monitorización de los datos obtenidos de los sensores. Para ello, es necesaria una interfaz de control y un protocolo de transmisión de datos adecuado a la velocidad de refresco. Será necesaria, como mínimo, una interfaz local, situada en el Gateway y es interesante considerar la posibilidad de tener acceso de manera remota.

4.4.1. Administración local

La interfaz local puede implementarse como un servidor que proyecte una página web alojada en el Gateway. Dicha interfaz será útil para resolución de problemas de sistema, observar los módulos incluidos en la red, hacer pruebas de diagnóstico y observar el flujo de datos de la red, además de incluir la posibilidad de modificar la configuración de la red o añadir o eliminar nodos.

Además, sin esta capa de mantenimiento, en caso de fallo de la instalación o caída de la red, no se puede conseguir un acceso al sistema con el fin de restaurarla.

4.4.2. Administración remota

La interfaz remota es opcional, pero sin duda un punto interesante a la hora de administrar la red y visualizar los datos, ya que permite realizar estas acciones sin necesidad de encontrarse en dentro de la misma red. Se pueden barajar dos opciones:

4.4.2.1. Acceso a interfaz local por túnel VPN

Ilustración 15 - Logo de OpenVPN

Una opción es acceder a la interfaz local creada anteriormente por medio de un túnel VPN. Es relativamente sencillo de configurar un servidor OpenVPN en el Gateway, creando las debidas credenciales y certificados de seguridad. Dado que se trata de un software abierto, está presente en la mayoría de plataformas. En general, los túneles VPN son muy seguros, ya que poseen un grado de protección muy alto[34], pero una mala configuración del servidor OpenVPN puede exponer la red a merced de piratas informáticos. Se recomienda siempre el uso de conexiones cifradas SSL/TLS.

Plantea el problema de mantener abierto un puerto (1194 UDP) a internet con la brecha de seguridad que eso conlleva, además de poder presentarse problemas debido a la complejidad de la red. Además, se requiere conectar al servidor VPN siempre que se requiera observar la actividad del sistema de sensores.

4.4.2.2. Interfaz en la nube

La segunda opción sería implementar una interfaz en un servidor alojado en la nube. Empresas como Amazon, IBM o Microsoft proporcionan servicios personalizados para cada aplicación. Estos servicios son muy versátiles y ofrecen muchas facilidades a la hora de programar y configurar la interfaz. Al igual que en una interfaz local, un servicio web puede comunicarse mediante diferentes protocolos para poder enviar órdenes directamente desde el mismo tales como MQTT o HTTP GET/POST.

5. DESCRIPCIÓN DETALLADA DE LA SOLUCIÓN ADOPTADA Y JUSTIFICACIÓN

En la siguiente tabla se recogen los datos de red de las tres tecnologías más apropiadas para este tipo de aplicaciones[35].

Tecnología	Bandas de frecuencia	Distancia máxima	Tasa de datos	Topología	Modulación	Longitud de datos	Potencia de transmisión
LoRa	868, 915, 433 MHz	15 - 20 km	0,25 - 50 kbit/s	Estrella	FSK - FEC	2 - 255 B	10 - 18 dBm
Sigfox	868, 915 MHz	3 - 10km	100 bit/s	Estrella	D-BPSK	Máx. 12B	0 - 14 dBm
IQRF	868, 915, 433 MHz	100 - 500 m	1.2 - 115 kbit/s	Mesh	FSK	Máx. 128B	Configurable, Max 11dBm

Ilustración 16 - Tabla de comparación LoRa, Sigfox e IQRF

Teniendo en cuenta que la aplicación se orienta a un entorno sin grandes distancias, al observar las características de la tabla y teniendo en cuenta las cifras de consumo y precio de módulos, IQRF es la tecnología que mejor se adapta a este proyecto. Otros factores refuerzan esta decisión, como la facilidad a la hora de configurar las redes y conectar con diferentes servicios web.

La solución elegida es, por lo tanto, un sistema basado en IQRF por su facilidad de configuración, especialización en este tipo de aplicaciones y conexión con los servicios web, cumpliendo con creces las exigencias de distancia y consumo.

El sistema se compondrá de una serie de nodos situados en las zonas donde se tomarán las medidas y las enviarán a un coordinador, que estará conectado por USB a un Gateway. Este Gateway emitirá por MQTT los datos de temperatura a la nube IBM Cloud, que los mostrará utilizando NodeRED.

El esquema de este sistema quedaría de la siguiente manera:

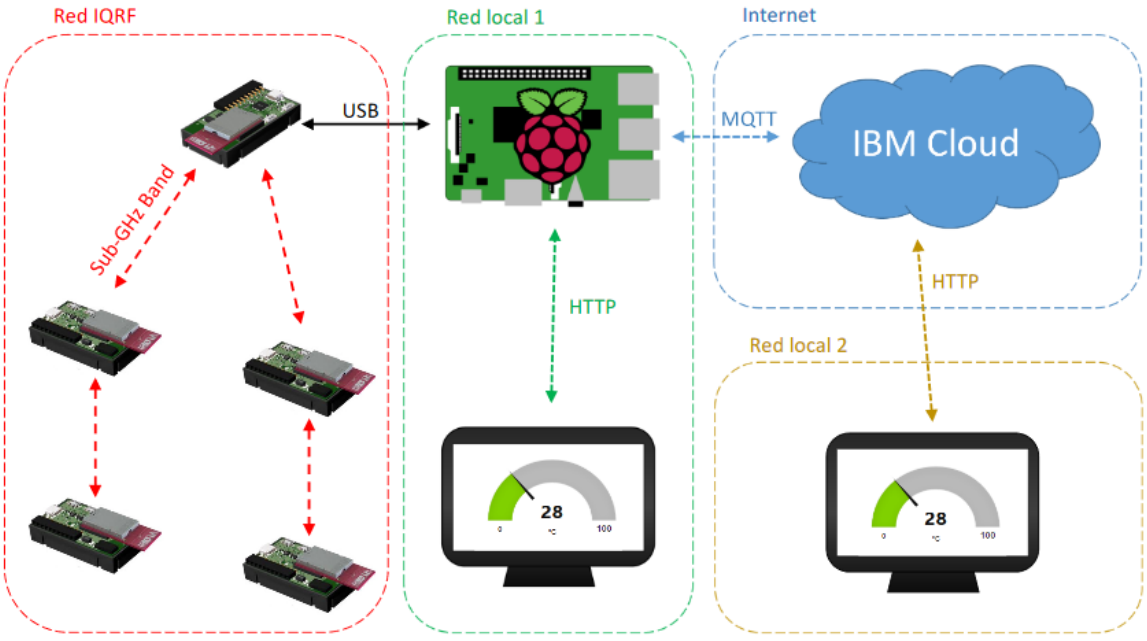


Ilustración 17 - Esquema de la solución adoptada

5.1. Sensores

Los sensores elegidos son los DS18B20, muy conocidos en el mercado, fiables y fáciles de configurar. Utilizan el protocolo de comunicación 1-wire. Se montarán estos sensores en formato T0-92, en una pequeña placa PCB con una resistencia de 4,7 kΩ entre el pin Vcc y el D0, tal y como se explica en el manual del fabricante.

La placa PCB contiene 3 pines de salida (Vcc, D0 y GND) que se conectarán a las entradas de los módulos Nodo. En medio quedarán dos pines vacíos para adaptarse exactamente al módulo al que se acoplará.

Se propone también una PCB alternativa para la parte del nodo, la cual posee el circuito de alimentación, en la que se integra el sensor de temperatura. Se expondrá posteriormente.

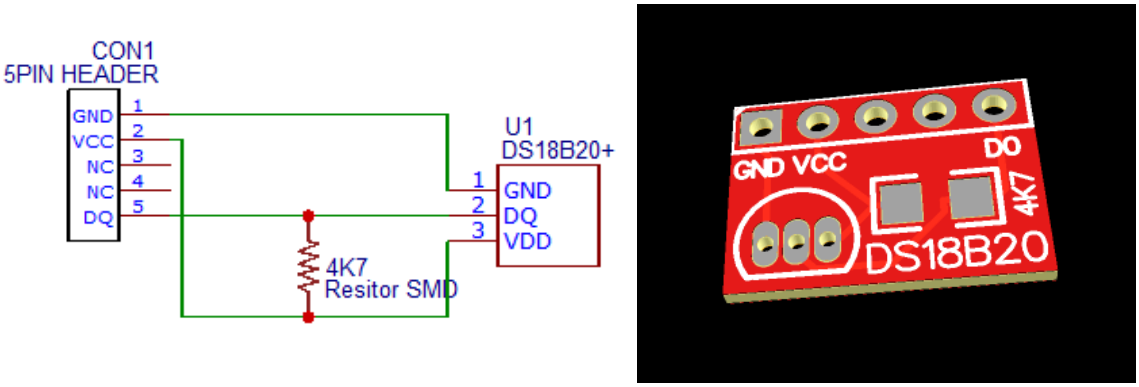


Ilustración 18 - Circuito del sensor

1-Wire es un protocolo de comunicación en serie desarrollado por Dallas Semiconductor. Se basa en un bus en el que se encuentran un dispositivo maestro y uno o más esclavos, a través del cual se realizan los intercambios de datos. Todos los dispositivos de la red deben estar referenciados a tierra.

El protocolo 1-Wire tiene ventajas como el uso de un solo cable para la comunicación bidireccional y la posibilidad de situar varios sensores en el mismo cable. Gracias a esto, se puede usar un solo pin del microcontrolador para utilizar varios sensores de temperatura. Para diferenciar los sensores, cada uno tiene un identificador único e

inalterable (64 bits) que el microcontrolador puede solicitar para diferenciar las mediciones.

El dispositivo maestro (en este caso, módulo TR), creará y controlará la comunicación del bus, mientras que los dispositivos esclavos (sensores), se sincronizan con el reloj del maestro. Esta comunicación se produce en Time Slots de 60 microsegundos y se puede estructurar en tres fases:

- Inicio y sincronización de los dispositivos. Los dispositivos esclavos se inician y sincronizan con el maestro.
- Comando ROM. Comando que se envía a los esclavos. Pueden ser enumeración, selección de dispositivo, etc. En este caso se puede utilizar el comando "Skip ROM", que se utiliza cuando solo hay un sensor en el bus, para direccionarlo.
- Comando memoria. Estos comandos son específicos de cada sensor, por lo que pueden ser diferentes de un tipo de sensor a otro. Leen o escriben la memoria interna de los dispositivos y sus registros. En el caso del DS18B20, un comando memoria sería la lectura de los 8 bits altos de temperatura.

5.2. Módulos TR

Los módulos se compondrán de:

- 11 módulos TR modelo DCTR-72DA, donde se encuentra los microcontroladores programados de la siguiente manera:
  - 10 módulos TR programados con perfil de nodo y funciones para el sensor DS18B20.
  - 1 módulo TR programado con el perfil de coordinador.

Ilustración 19 - Módulo TR-72DA

Este módulo TR posee una antena PCB integrada que permite alcances de 170m en condiciones óptimas de difusión. También posee diferentes periféricos como regulador LDO de 3V (entrada de hasta 5,1V), conversor AD y comparador, HW timer, memoria EEPROM de 256Kb, salida PWM, 2 LEDs configurables y puertos configurables I2C y SPI, con un consumo extra-bajo (desde 1,7µA en "Sleep mode", 2,8mA con "RF ready"). La capa de sistema operativo (IQRF OS) permite un control más sencillo de los periféricos y funciones del dispositivo.

Para aplicaciones con menor número de módulos se puede utilizar el módulo TR DCTR-72DC, el cual posee un conector U.FL, que permite la conexión de una antena externa, introduciendo la posibilidad de aumentar notablemente el rango de actuación. Sin embargo, para esta aplicación no se disponen de este tipo de módulos, por lo que no se podrá comprobar la mejoría de una antena externa.

5.3. Módulo Coordinador

- 1 placa CK-USB-04A, con 1 módulo TR con perfil de coordinador, que se conectará mediante USB al Gateway y servirá para enviar y recibir los datos.

Ilustración 20 - Placa CK-USB-04A

Esta placa sirve tanto para programar el módulo TR como para conectar el coordinador al Gateway, con el fin de emitir y recibir datos. No posee batería ni la requiere, ya que se alimenta a través del mismo puerto USB por el que circulan los datos. Este módulo puede configurarse en diferentes modos, a través del IQRF IDE:

- Modo Custom Device: Este modo es el que se utiliza para subir el código a los módulos TR, con las instrucciones que necesita. A su vez, el módulo TR con perfil de Coordinador utiliza este modo para enviar y recibir órdenes en un PC con Windows a través del IQRF IDE, con fines depurativos.
- Modo CDC IQRF: Este modo es el utilizado para comunicación entre el Gateway (más concretamente la herramienta iqrf-daemon) y el módulo TR coordinador. Este modo será el utilizado principalmente para las comunicaciones entre la red IQRF y el Gateway.
- Modo CDC SPI: Este modo no se utilizará en esta aplicación. Permite utilizar un adaptador universal de módulo TR a pines genéricos. Se puede utilizar con los mismos fines que el modo CDC IQRF con dispositivos que puedan realizar comunicaciones SPI.
- Modo CDC UART: Modo funcionalmente similar al CDC SPI. Se utilizará con dispositivos que puedan realizar comunicaciones a través de UART.

5.4. Módulos Nodo

Una de las opciones propuestas es usar placas DK-EVAL-04A con módulos TR programados con el perfil de nodo, batería incorporada y sin interfaz USB y una PCB con sensor por placa acoplada a su salida. El conector USB integrado en la placa sirve para cargar la batería.

Ilustración 21 - Placa DK-EVAL-04A

Los módulos se pueden configurarar con el modo FRC o de respuesta rápida, con el fin de que los datos de temperatura de todos los nodos se reciban en el mismo momento y en el mismo paquete de datos. De esta manera se facilita la discretización de éstos y se evitan grandes cambios a la hora de modificar la red IQRF, ya sea añadiendo, eliminando o sustituyendo módulos. No se recomienda dado que presenta unos consumos más altos, dado que para que la respuesta esté siempre disponible, se está constantemente tomando medidas de temperatura.

Para programarlos se utilizará el código de ejemplo para una placa de expansión complementaria que posee el sensor DS18b20 (0002_DDC_SE01) y se realizará la subida a cada dispositivo desde un ordenador con el IQRF IDE instalado.

Estos módulos poseen una batería integrada (LIP552240) de 400mAh con un voltaje nominal de 3,7V[36].

5.4.1. PCB Personalizada

Alternativamente a la solución compuesta por una placa DK-EVAL-04A, la cual sirve para alimentar el módulo TR y proporcionar el puerto para el sensor, se ha diseñado una placa PCB que puede realizar la misma función, además de incluir el sensor de temperatura. Con esto se consigue reducir los costes, además de añadir la posibilidad de instalar una batería de mayor capacidad, traduciéndose en una mayor autonomía. La siguiente imagen muestra el acabado final de la PCB.

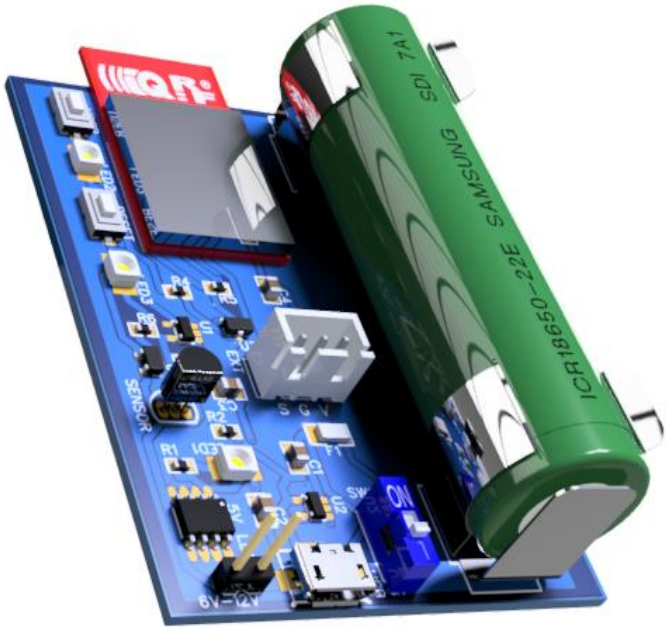


Ilustración 22 - PCB con batería y sensor

Este diseño sustituye la batería Li-ion de 400mAh integrada por una batería de formato 18650, que, dependiendo del fabricante, pueden tener una capacidad de 1600 hasta 3600 mAh. Este tipo de baterías poseen un formato extraíble que permite el reemplazo instantáneo de la batería, reduciendo el tiempo de interrupción a segundos.

También se consigue integrar el sensor de DS18B20 en dicha PCB, siendo únicamente necesario colocar el módulo TR en el zócalo.

En el diseño se han integrado las siguientes partes:

- Módulo de carga de la batería y alimentación de 12V a 5V. Posee un puerto micro USB a través del cual se puede alimentar el módulo y/o cargar la batería a una tensión de 5V, más que estandarizada en este tipo de aplicaciones. Además, incluye dos pines de alimentación, que admiten un voltaje de 6V a 12V, que posteriormente se regula para conseguir los 5V.

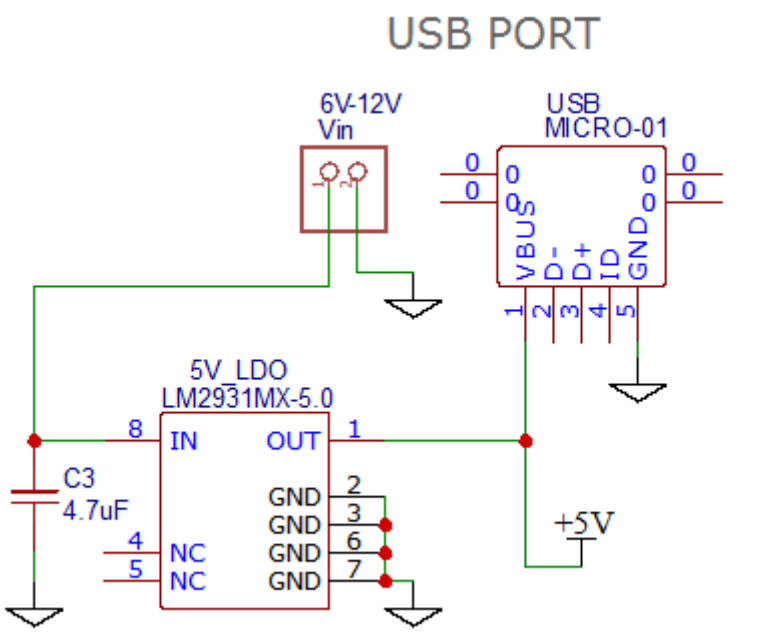


Ilustración 23 - Puerto USB y entrada 6-12V

El circuito siguiente posee un regulador de carga de batería MCP73831T alimentado con los 5V anteriores. Éste se ha configurado para una corriente de carga de 505mA mediante el uso de una resistencia de 2kΩ conectada al pin PROG, según indica el fabricante[37]. Se ha añadido un led que indica el estado de carga de la batería. La siguiente imagen incluye también la batería, con un fusible que se rearma automáticamente.

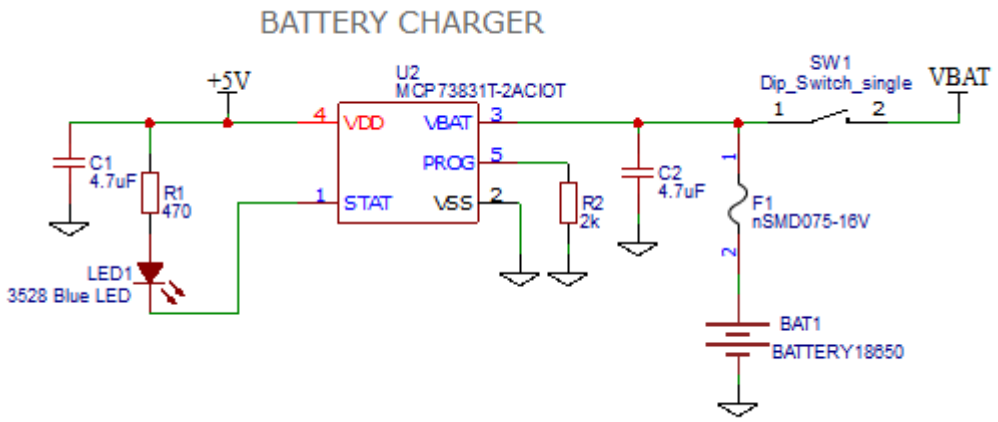


Ilustración 24 - Circuito de carga de batería

- Zócalo de Batería 18650. Permite baterías del formato 18650 con capacidades desde 1600 a 3600mAh, que proporcionan una autonomía de entre 4 y 9 veces la ofrecida por el módulo del fabricante.

Ilustración 25 - Zócalo de batería 18650

- Zócalo del Módulo TR. En dicho zócalo se situará el receptor TR, permitiendo su rápida y sencilla extracción, ya que posee un formato SIM. La propia empresa IQRF Tech s.r.o. suministra este componente[38]. El módulo TR irá alimentado a 3.3V y conectará sus dos pines C5 y C6 al botón de usuario programable y al sensor de temperatura, respectivamente.

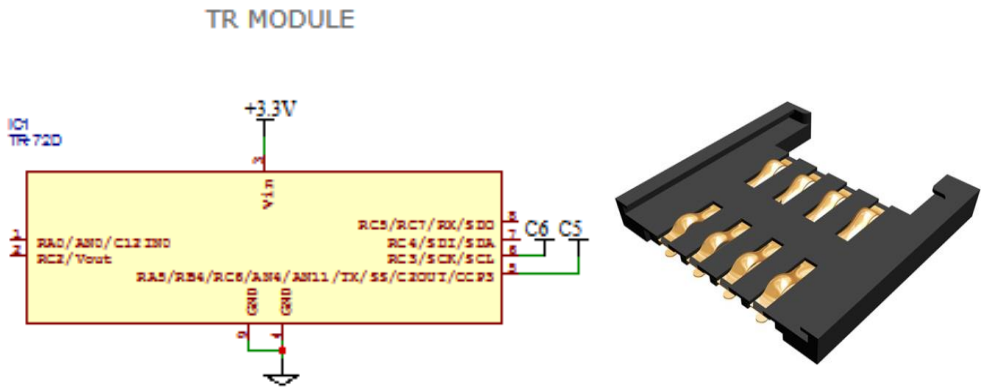


Ilustración 26 - Módulo TR y zócalo

- Circuito de sensor de temperatura DS18B20. Este circuito incluye el sensor DS18B20 en la PCB, junto a su resistencia de 4,7 KΩ, indicada por el fabricante. Además, incluye un conector TX11 para poder conectar una sonda externa en lugar del sensor interno[6].

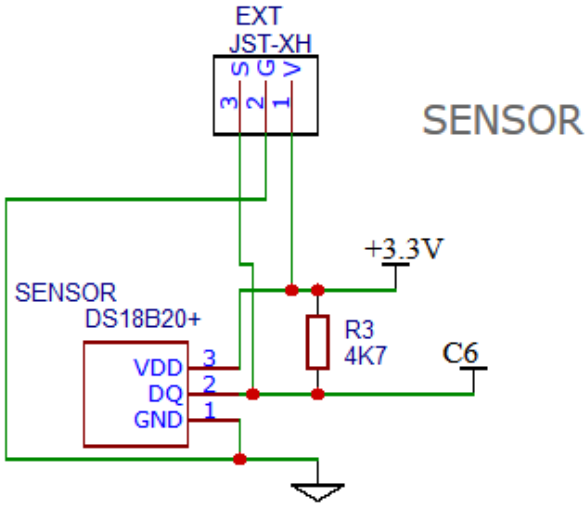


Ilustración 27 - Sensor DS18B20

- Pulsador de usuario con LED. Circuito mediante el cual se conecta un pulsador al pin C5 del módulo TR, que puede ser configurado como pin de entrada. Este botón se puede usar para realizar una función programada en el módulo TR y posee un LED que se enciende al pulsar el botón. En este proyecto no se utilizará, pero se ha incluido para posibles implementaciones en el futuro.

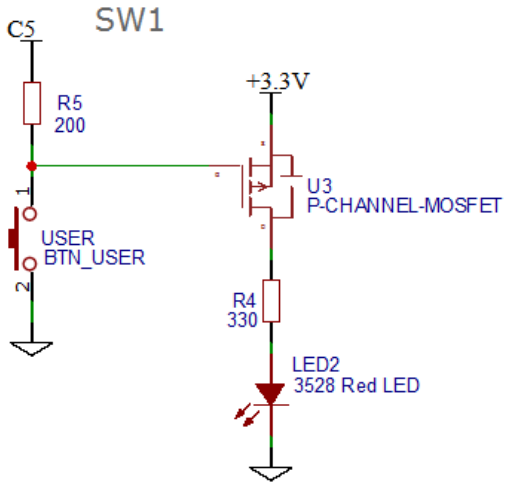


Ilustración 28 - Pulsador de usuario

- Pulsador de reinicio con LED. Este circuito posee un pulsador que al ser presionado enciende un led e interrumpe la corriente de 3.3V de la PCB, apagando el módulo TR y el sensor. Pulsándolo, se activa el paso de corriente por el transistor encendiendo el LED y se envía un nivel bajo al pin "Enable" del regulador de 3.3V.

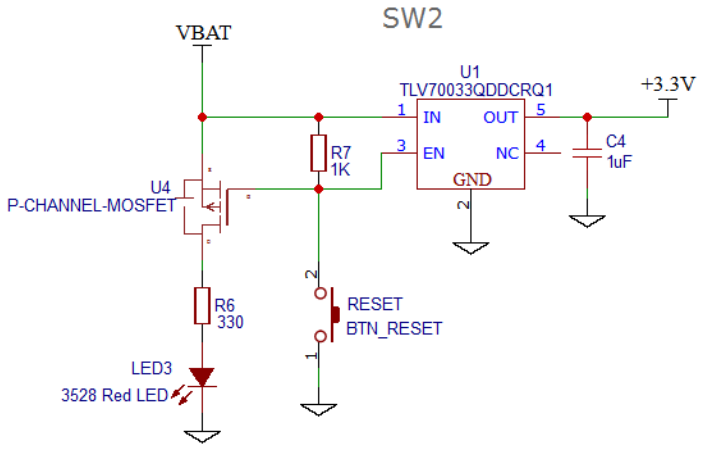


Ilustración 29 - Pulsador de reset

- Interruptor de alimentación. Frente al método de encendido y apagado del fabricante, que consiste en un jumper, se ha optado por un interruptor DIP. El uso de este tipo de interruptores elimina la posibilidad de pérdida de los jumpers. Aparece en el circuito correspondiente al cargador de la batería.

Ilustración 30 - DIP Switch

Se ha optado por un diseño con componentes SMD para reducir el tamaño de la placa. Las dimensiones de la PCB diseñada son de 50x70mm con un espesor de 1,6mm. La altura máxima incluyendo los componentes es de 19,6 mm, que se obtiene del espesor de la placa sumado a la altura del mayor componente, que es la batería 18650, que se monta horizontalmente y posee un diámetro de 18mm (18mm de diámetro, 65mm de longitud).

Por otro lado, no se han considerado efectos adversos de temperatura, ya que el circuito es de muy baja potencia y no generará un calor que pueda afectar a la funcionabilidad de la batería o a las medidas del sensor. En su colocación, se recomienda evitar situaciones de incidencia de luz directa o entornos muy adversos, como cualquier dispositivo alimentado con baterías.

La dimensión del grosor de las pistas de la placa se ha acercado a 1mm, quedándose a 0,8mm dada la imposibilidad de aumentar el ancho de pista debido a las dimensiones de la placa. Al ser una aplicación de ultra baja potencia no presentará ningún problema. Por este mismo motivo, no hay limitación por parte de la batería para suministrar carga, ya que este formato de baterías posee una corriente máxima de carga cercana a 2,41A[39], muy lejana a la corriente diseñada, que es de 505mA[37].

El esquema eléctrico completo se encuentra en el interior de la sección Planos de este TFG, así como el diseño de la PCB, ya que esta propuesta es la que se adoptará en este trabajo.

5.4.1.1. Autonomía

En cuando a la autonomía, se ha calculado teóricamente el consumo de los tres circuitos que están constantemente funcionando en la PCB diseñada. Posteriormente se ha calculado la dimensión de la batería con un valor máximo y un valor mínimo según la batería 18650 elegida y se ha comparado con la solución del fabricante. Los tres circuitos son los siguientes:

- Módulo TR-72DA: El consumo del módulo TR, el cual será el dispositivo que más afecte a la autonomía. Tiene dos estados: emisión y sleep mode.
- Net SW2: En este circuito se encuentra el interruptor de reset y el transformador de voltaje. Se encuentra siempre en estado activo.
- Net Sensor: Corresponde al consumo del circuito del sensor de temperatura DS18B20. Posee un tiempo de medición y un tiempo de modo Standby.

Los cálculos se estiman en un tiempo de 2 minutos, ya que el ciclo de medición del sensor tiene ese periodo.

Net	Consumo Activo (mW)	Tiempo Activo (s)	Consumo Standby (mW)	Tiempo Standby (s)	Consumo Total (mW)
TR-72DA	27,39	0,695	0,00759	119,305	0,16618
Net SW2	0,000231	120	0,1023	0	0,00023
Net Sensor	5,617021	0,495	0,002475	119,505	0,02564
TOTAL					**0,19205**

Ilustración 31 - Consumo circuitos PCB

Por lo tanto, el consumo del dispositivo nodo diseñado es de 0,192 mW. Según las capacidades de las baterías posibles, se han calculado las autonomías.

Batería	Batería original	Batería 18650 mín.	Batería 18650 máx.	Samsung INR 18650-25R
Capacidad (mAh)	400	1600	3600	2500
Autonomía (h)	6873,36	27493,42	61860,20	42958,47
Autonomía (años)	**0,78**	**3,14**	**7,06**	**4,90**

Ilustración 32 - Autonomía baterías

A la vista de los resultados, la autonomía del nodo se puede estimar entre 3,14 y 7,06 años dependiendo de la batería 18650 utilizada. La batería elegida, Samsung INR 18650-25R, proporciona una autonomía de **4,9 años**. Esto supone un aumento de 6,25 veces la autonomía del módulo del fabricante.

5.5. Carcasas de protección:

Para la protección y fijación de los nodos se ha diseñado la siguiente carcasa en 3D, impresa mediante impresora 3D y probado con material ABS. Se dispone su uso para los nodos IQRF dotados del sensor de temperatura.

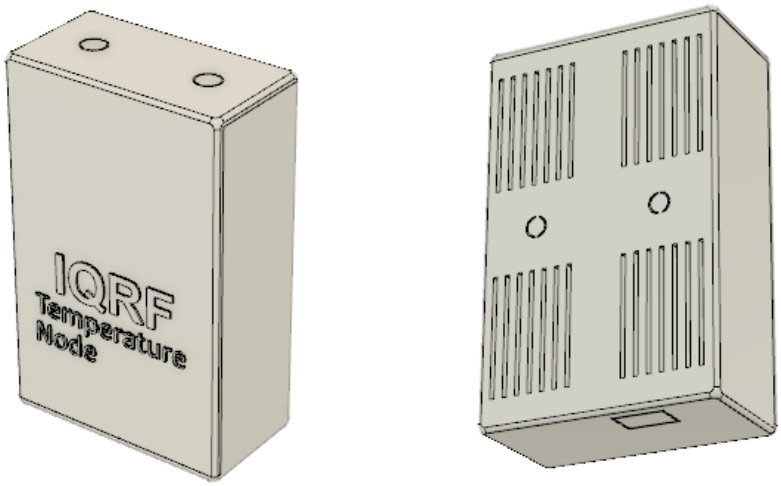


Ilustración 33 - Carcasa nodo

Con unas dimensiones de 56,20 x 89 x 30,3mm, su diseño está orientado a la PCB anteriormente diseñada, aunque podría usarse también junto con la solución del fabricante. Se han incluido perforaciones para situar antena externa (para su uso con un módulo TR con conector UF.L), sonda externa, para cargar la batería por puerto USB sin extraer la PCB y dos perforaciones más para situar tornillos de montaje. Además, en la parte posterior, posee ranuras que permiten el paso del aire, con el fin de conseguir unas medidas acordes con la temperatura ambiente.

Los esquemas correspondientes y medidas se encuentran en la sección Planos de este TFG.

5.6. Red

En el sistema desarrollado se deben diferenciar dos redes de distinta naturaleza: la red IQRF, que contendrá los módulos y sensores, y la red LAN con conexión a internet, en la que se situará el Gateway y su conexión a la nube.

5.6.1. Red IQRF

La red IQRF es aquella red mesh compuesta por los módulos IQRF, los cuales se comunican entre sí mediante radiofrecuencia y se basan en el sistema DPA. Este sistema permite de una manera simple, con un volumen de datos muy bajo, enviar información entre los diferentes módulos anclados a la red. Es una red segura que en la que solamente los dispositivos autorizados pueden enviar o recibir información.

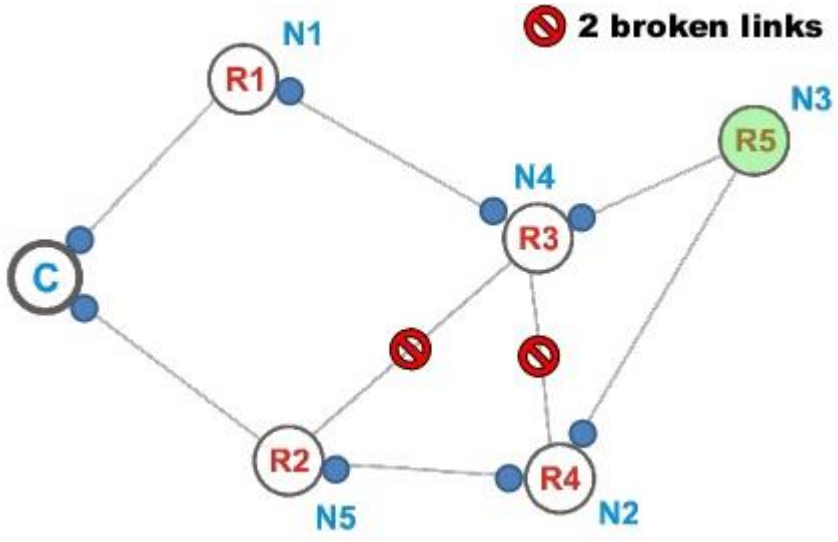


Ilustración 34 - Red IQRF MESH

Las redes mesh tienen como ventaja la posibilidad de comunicarse con módulos que no tienen una conexión directa con el coordinador, permitiendo extenderse más allá de su alcance, utilizando otros módulos como puente. Tal y como se muestra en la imagen anterior, si un módulo, como el R5, no tiene una conexión directa con el coordinador (nombrado como C), utilizará otros módulos de la red para llegar hasta él. En el caso de R5, encuentra dos caminos de conexión, R5->R4->R2->C y R5->R3->R1->C, por lo que el algoritmo de red mesh decidirá cuál será el mejor camino a seguir.

El sistema IQMESH de IQRF asigna una dirección del 0 al 239 de manera automática a todos los módulos autorizados, para que los paquetes de datos (órdenes y datos de temperatura) se envíen a módulos concretos de la red IQRF. A esto se le llama "autorouting" o enrutamiento automático.

El DPA (Direct Peripheral Access) proporciona una manera de ejecutar acciones concretas utilizando un volumen de datos muy pequeño. Se compone de cuatro partes:

NADR: Se trata de un valor hexadecimal que determina la dirección de cada dispositivo de la red IQRF. Gracias a esta dirección se puede enviar una orden específica a uno de los elementos de la red. Sus valores se pueden asignar automáticamente o de manera personalizada, yendo desde el valor 0x00 a 0xEF. El valor 00 siempre corresponderá al coordinador de la red. También es posible enviar órdenes a todos los dispositivos mediante la dirección de broadcasting, con valor FF.

PNUM: Corresponde al número del periférico. También es un valor hexadecimal que determina a qué periférico del módulo RF va dirigida la orden. Sus valores preestablecidos van de 0x00 al 0x1F para los periféricos integrados en el módulo (LEDs, memoria EEPROM, memoria RAM, puerto SPI, etc). Las direcciones del 0x20 al 0x3E se destinan a periféricos personalizados por el usuario.

PCMD: Determina la función a ejecutar. Éste valor hexadecimal se corresponde con cada una de las funciones integradas en cada periférico del módulo. Se pueden situar desde 0x00 a 0x7F, omitiendo la dirección 0x3F, la cual está reservada para el sistema.

HWPID: Perfil de hardware. Este valor, también hexadecimal, se usa para filtrar aquellos módulos incluidos en la red que no puedan ejecutar la orden emitida, porque se destinan a otros fines.

PData: Cadena de bytes opcional. Se usa para enviar información entre módulos. En el caso planteado se usará, por ejemplo, para devolver los datos de temperatura al coordinador.

5.6.2. Red LAN y conexión a Internet.

Con el fin de poder monitorizar los datos recogidos por los sensores, será necesario usar determinadas redes, asignar una dirección IP interna fija y tener acceso a internet.

La red local o LAN será un factor más a considerar, ya que, dependiendo de su estructura en cuanto a dominios y permisos de red, el sistema puede presentar problemas a la hora de acceder a la interfaz del Gateway, impidiendo el mantenimiento y/o las modificaciones del sistema IQRF, además de la configuración de la conexión a la nube.

Será determinante establecer una dirección IP interna fija para poder localizar el Gateway fácilmente dentro de la red local, además de situarlo en un dominio común al equipo que se usará para diagnóstico y solución de problemas.

5.7. Gateway

El Gateway se encontrará en las instalaciones donde se recogerán los datos de temperatura. Estará compuesto por una Raspberry Pi, en este caso la versión 3 modelo B, que lleva incorporado WiFi, así como interfaz Ethernet, por si existieran problemas a la hora de utilizar la red WiFi existente. No se han tenido en cuenta otras placas alternativas, ya que este proyecto no se centra en ello. Cualquier placa compatible con Linux y al menos un puerto USB, además de posibilidad de conectar a Internet, hubiera desempeñado de la misma manera el papel de Gateway. Se ha optado por la Raspberry Pi por su bajo coste y su facilidad de configuración, al ser la placa más extendida.

Contendrá un sistema operativo basado en Linux en su tarjeta micro SD, concretamente Raspbian, lo que aporta las posibilidades de un sistema operativo Linux en un microordenador de tamaño reducido. Una vez instalado el sistema Raspbian, actualizado a su última versión, se procederá a configurar el servidor OpenSSH, que proporcionará un terminal de comandos Linux (Bash), con el que se administra el sistema en la Raspberry Pi, sin necesidad de monitor (headless system). Mediante un túnel SSH se consigue a través de una red WAN acceder a la consola del Gateway, facilitando su configuración.

5.7.1. Instalación de servidor OpenSSH

De no estar ya instalado en la distribución de Raspbian, se puede instalar mediante los comandos:

```
$ sudo apt-get install -y ssh
$ sudo systemctl enable ssh.service && sudo systemctl start ssh.service
```

Los cuales instalan el paquete SSH y lo habilitan para iniciarse en el arranque. Para este caso se configura el usuario pi (usuario por defecto) con una contraseña, protegiéndolo de posibles modificaciones no autorizadas.

```
$ sudo passwd pi
```

Una vez configurado, se puede obtener un acceso SSH desde la misma red local mediante la dirección IP asignada a la Raspberry Pi en el puerto 22, mediante el programa adecuado según el sistema operativo del ordenador de administración.

En el caso de Windows se puede usar Hercules o Putty. En Putty, suponiendo una dirección IP interna 192.168.1.65, se conectaría al servidor SSH de la siguiente manera:

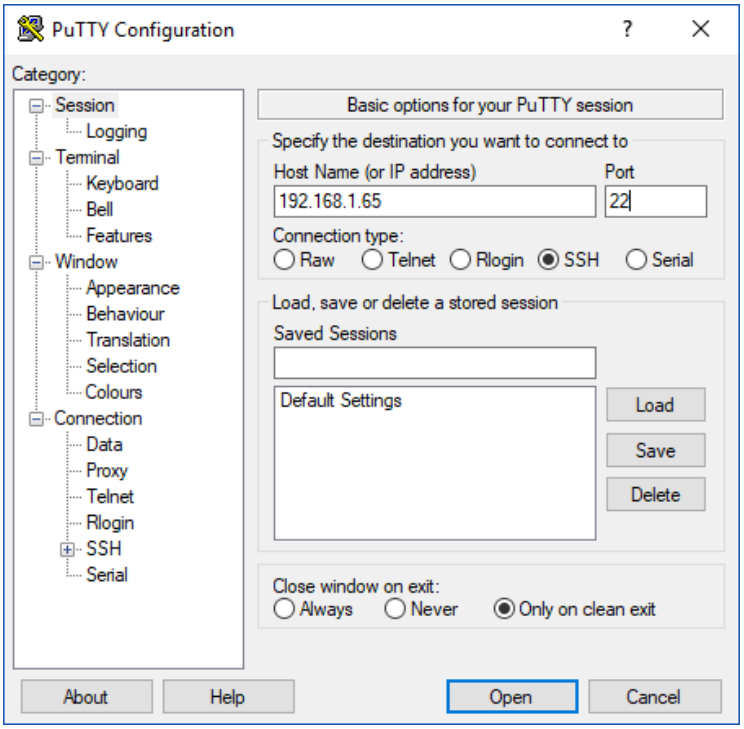


Ilustración 35 - Conexión con Putty

Introduciendose usuario “pi” y contraseña “raspberry”, se obtiene acceso a la consola del Gateway:

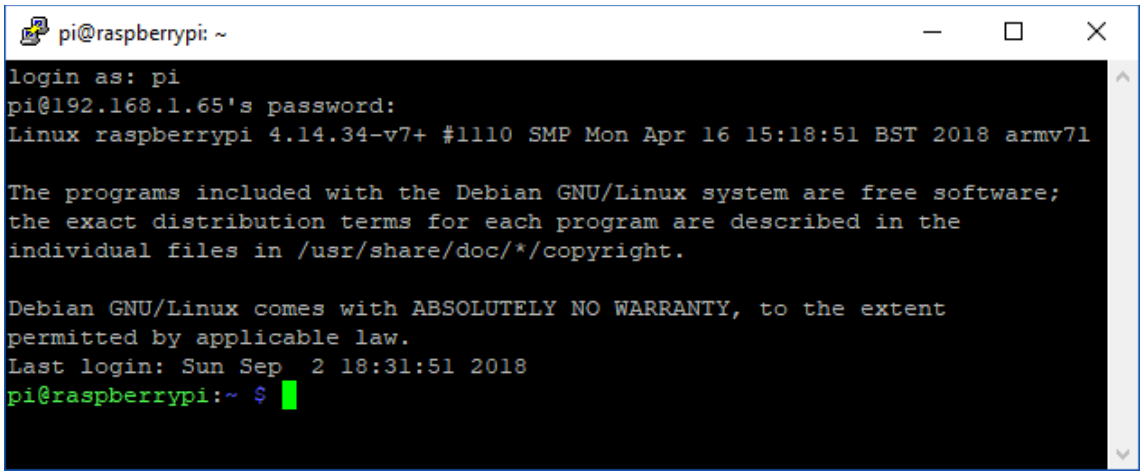


Ilustración 36 - Consola del Gateway

Para Linux se puede acceder mediante el Terminal integrado con el comando “**ssh pi@192.168.1.65:22**”. Introduciendo la contraseña por defecto “raspberry”, se llega al mismo punto de la imagen anterior.

Una vez configurado el acceso SSH, se obtiene una consola de comandos directa al Gateway y se puede dar paso a instalar las utilidades necesarias.

5.7.2. Habilitar UART como puerto serie.

En el caso de no poder gestionar la consola a través de la red local (por complejidad de la red, no tener acceso…), existe la posibilidad de habilitar determinados pines del Gateway como interfaz serie.

Estos pines son el GPIO14 (UART0_TXD) Y GPIO15(UART0_RXD) de la Raspberry Pi, correspondientes a los pines 8 y 10. Estos pines funcionan a 3.3V y no pueden ser conectados a dispositivos serie que trabajen a 5V (como Arduino o puerto serie de PC). Para poder utilizarlos, la opción más sencilla es utilizar un conversor UART-USB (o chip FTDI) que trabaje a 3.3V.

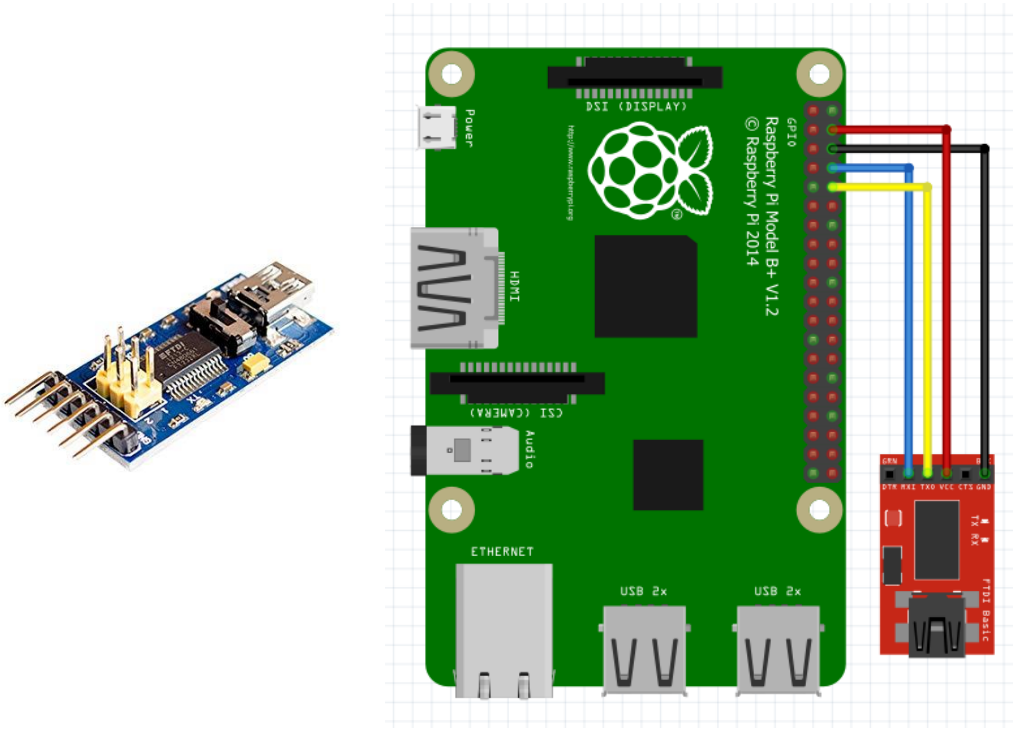


Ilustración 37 - Conexión FTDI y RPI

Se conectan los pines en configuración "null-modem", la cual consiste en conectar los pines de voltaje (VCC) y las masas (GND) y los pines de envío (TX) y recepción (RX) cruzados, es decir, RX del adaptador se conecta a TX de la Raspberry Pi (Pin 8, UART0_TXD), y TX del adaptador a RX de la Raspberry Pi (Pin 10, UART0_RXD).

Además, se debe configurar dichos pines en la consola para habilitarla como puerto UART. Esto se consigue añadiendo "`enable_uart=1`" al final del archivo config.txt situado en la partición boot de la tarjeta SD. Con un solo comando:

```
$ sudo echo "enable_uart=1" | sudo tee -a /boot/config.txt
```

Otro método, disponible en algunas distribuciones como Raspbian, es utilizar el comando "`raspi-config`" con privilegios sudo, que mostrará un menú. Seleccionando la opción "`5. Interfacing options`" y habilitando la opción "`Serial`", se consigue el mismo resultado.

Hecho esto, ya está disponible la comunicación UART hasta el ordenador. Para abrir la comunicación, se puede usar Putty en Windows escribiendo el puerto COM correspondiente al adaptador FTDI, seleccionando una velocidad (Baud Rate) de 115200. En Linux se puede usar el comando de minicom:

```
$ sudo minicom --device /dev/ttyUSB0
```

5.7.3. Instalación del IQRF Gateway Daemon y IQRF Gateway Daemon WebApp

El IQRF Daemon es un paquete de utilidades para la línea de comandos que permite la comunicación con la red IQRF. Se instala de la siguiente manera:

```
$ sudo apt-get install -y dirmngr
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 9C076FCC7AB8F2E43C2AB0E73241B9B7B4BD8F8E
$ echo "deb http://repos.iqrfsdk.org/raspbian stretch stable" | sudo tee -a /etc/apt/sources.list
$ sudo apt-get update && sudo apt-get install -y iqrf-daemon
```

Y se comprueba que esté activo:

```
$ sudo systemctl status iqrf-daemon.service
```

Que debería devolver el estado activo. En caso contrario:

```
$ sudo systemctl restart iqrf-daemon.service
```

Se procede a instalar IQRF Daemon WebApp, que servirá para administrar la red iqrf de una manera visual, mediante una interfaz web:

```
$ cd /home/pi
$ sudo apt-get install -y git && git clone https://github.com/iqrfsdk/iqrf-daemon-webapp.git
$ cd iqrf-daemon-webapp/install/
$ sudo python3 install.py -d debian -v 9
```

Se requiere Python 3 para la instalación, de no estar instalado:

```
$ sudo apt-get install python3
```

Para comprobar el correcto funcionamiento de la interfaz, se accede mediante navegador web a la dirección IP del Gateway: http://192.168.1.65/. Aparecerá una interfaz de inicio de sesión, con credenciales por defecto user=**admin** y pass=**iqrf**.

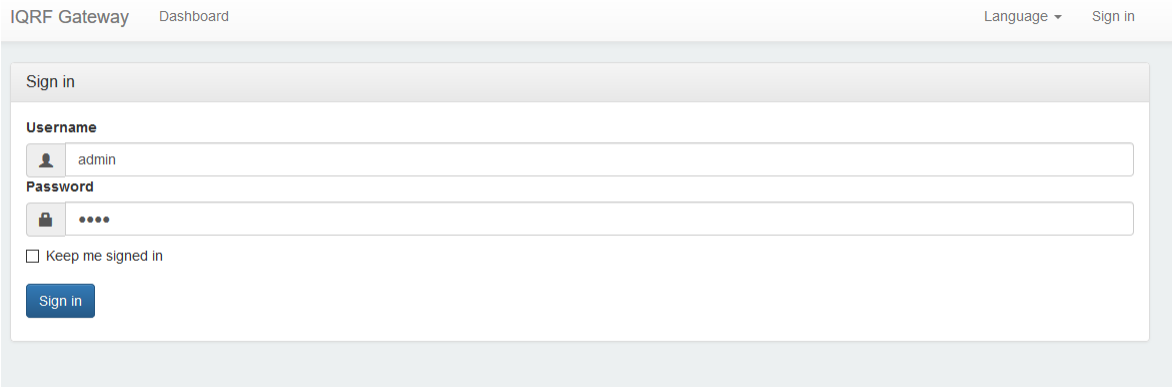


Ilustración 38 - Interfaz IQRF DAemon WebApp

5.7.4. Configurar el coordinador en el Gateway

Para establecer conexión con la red IQRF es necesario utilizar un módulo IQRF conectado por USB al Gateway, el coordinador. Para que el IQRF Daemon lo detecte y pueda interactuar con él, será necesario que esté configurado en el modo IQRF CDC. Se puede configurar desde el IQRF IDE, instalado en un ordenador con Windows. El programa IQRF IDE se puede conseguir gratuitamente desde la web oficial: https://www.iqrf.org/technology/iqrf-ide/iqrf-ide-gui .

Una vez instalado, se conecta el coordinador por usb y se activa **Tools -> USB Classes -> Switch to CDC IQRF mode**.

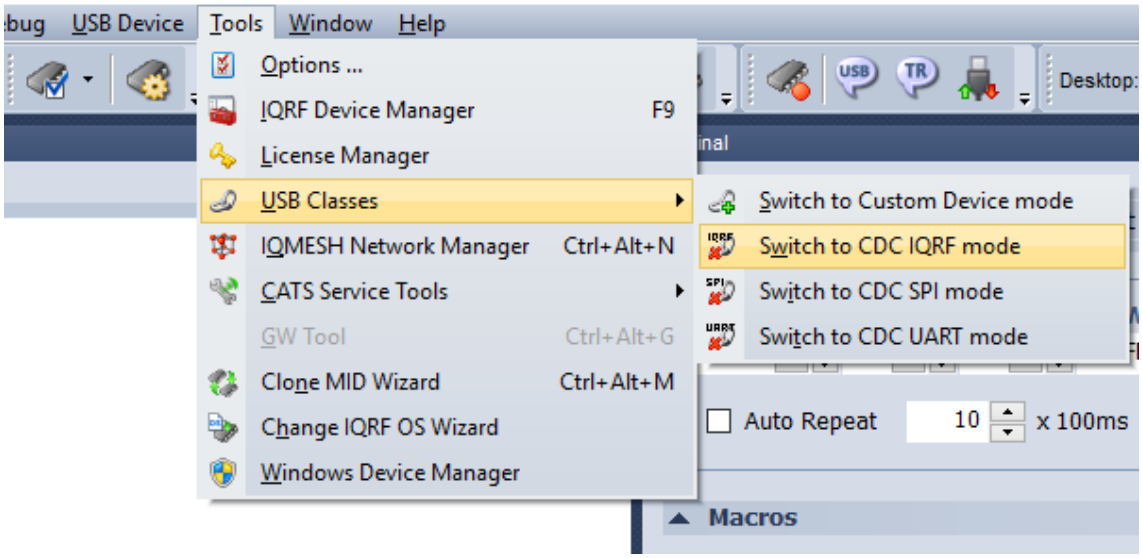


Ilustración 39 - Cambio a modo CDC IQRF

Se esperan unos segundos y se desconecta el coordinador para conectarlo en el Gateway. En la dirección http://192.168.1.65/en/config/iqrf, debe aparecer el coordinador como USB CDC.

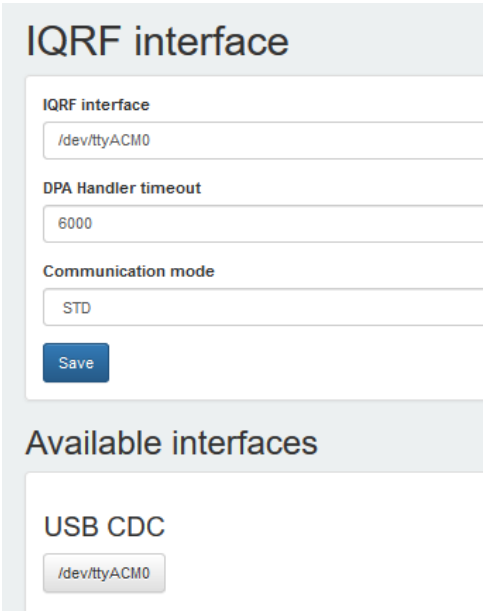


Ilustración 40 - Información del coordinador

Se guarda la configuración tal y como aparece en la imagen y se resetea el servicio iqrf-daemon para asegurar que detecta al coordinador usando el comando:

```
$ sudo systemctl restart iqrf-daemon.service
```

Llegados a este punto, el coordinador debe aparecer, con los datos del módulo, en la sección Status de la interfaz web http://192.168.1.65/en/gateway/info/ :

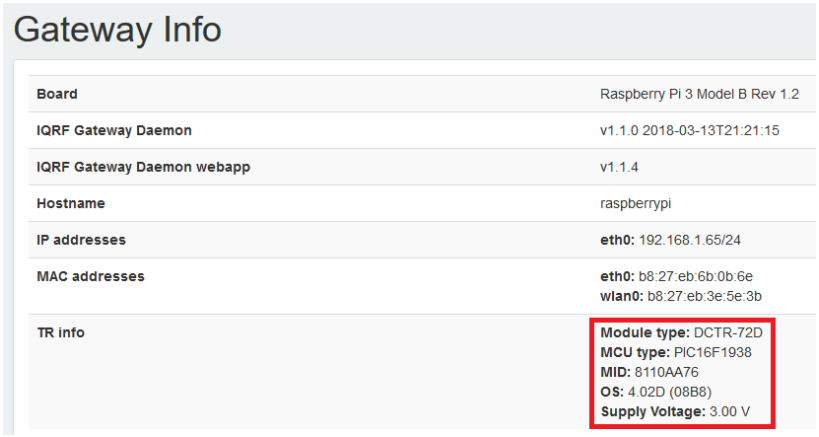


Ilustración 41 - Información del gateway

El coordinador ahora está correctamente configurado y puede usarse para enviar órdenes a la red IQRF desde la interfaz web.

5.7.5. Interfaz de control local

La interfaz de control local que se ha instalado previamente permite tener un control de diagnóstico y mantenimiento para la red IQRF. Proporciona diversas utilidades para el control de la red, entre las que destacan la modificación de los elementos de la red, añadir o eliminar módulos o enviar órdenes personalizadas, además de una pantalla donde se muestra el registro de acciones realizadas por el módulo coordinador en el servicio iqrf-daemon.

- Registro de iqrf-daemon (log)

  El registro proporciona un resumen de los eventos ocurridos en el servicio iqrf-daemon, permitiendo monitorearlo y diagnosticar posibles problemas. Se puede acceder a él mediante la conexión SSH al Gateway y ejecutando el comando

"sudo systemctl status iqrf-daemon.service", aunque desde la interfaz web del IQRF Daemon se puede visualizar también.

Esto se consigue accediendo desde la misma red local en la que se encuentra el Gateway y utilizando su dirección IP (192.168.1.65 en este caso) en la siguiente dirección: http://192.168.1.65/gateway/log/. Aquí aparecerán todas las órdenes emitidas por el coordinador, además del estado del servicio y los cambios de configuración hechos en la interfaz web o en la red IQRF.

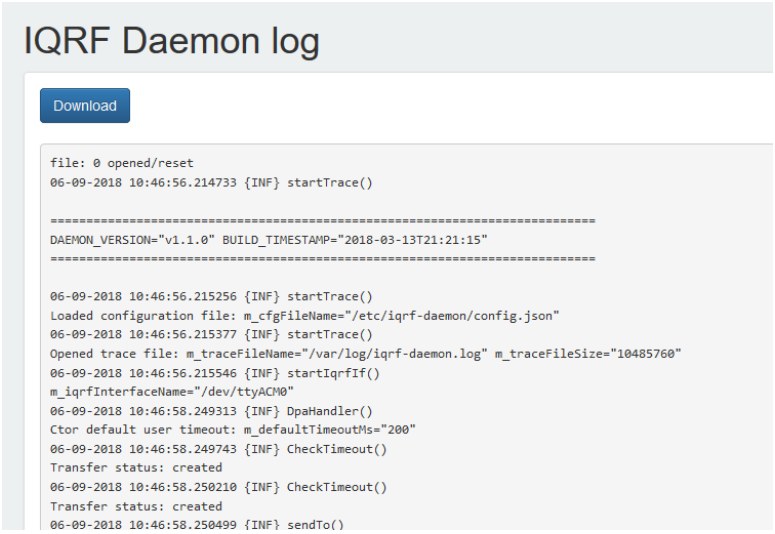


Ilustración 42 - Registro de iqrf-daemon

- Añadir o eliminar módulos

La interfaz web proporciona una pantalla para la modificación de la red IQRF llamada IQRF Network Manager con la finalidad de añadir, eliminar o sustituir módulos.

El método de autentificación en la red reside en el coordinador y consiste en un método similar al "pairing", solo que con una red con más de dos dispositivos. Este sistema se llama "Bonding" en los sistemas IQRF, y es posible modificarlo en la página: http://192.168.1.65/iqrfnet/network/.

Desde aquí, el apartado Bonding permite, mediante una serie de botones, añadir un nodo con una dirección personalizada o automática, volver a autentificar

un nodo que solo se ha eliminado en el coordinador, eliminar el nodo de la dirección elegida o incluso eliminar todos los módulos autentificados en la red.

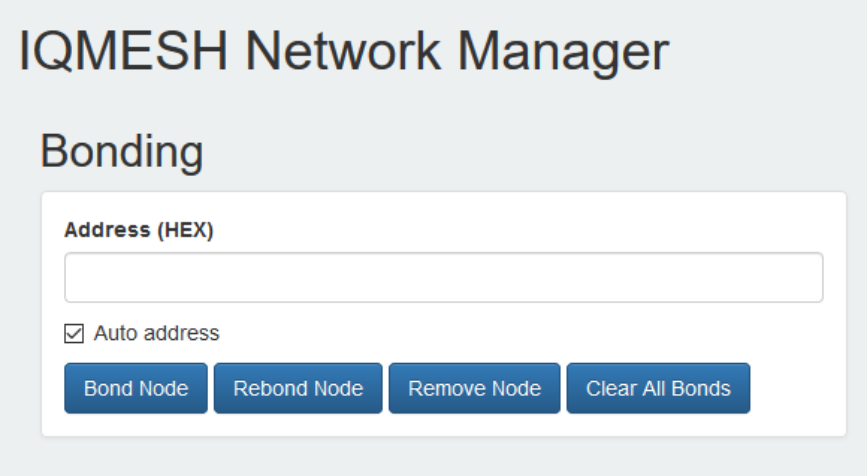


Ilustración 43 - Sección Bonding

Debajo de esta sección, se puede modificar la contraseña establecida en formato ASCII que proporciona una seguridad todavía mayor de la red IQRF.

- Enviar una orden (DPA packet)

Desde la interfaz también se pueden emitir órdenes y observar el correcto funcionamiento de la red IQRF. Esta función no decodifica los datos y sólo permite observar los datos sin procesar (RAW format), además de no poder programar la orden para que se envíe periódicamente. Esto hace que sólo pueda usarse con fines de diagnóstico o consultas esporádicas de la red, ya sea a todos los módulos o algunos concretos.

Se puede configurar un tiempo de espera en el que debe recibirse la respuesta, o de lo contrario se devolverá un error, cuyo valor por defecto es de 1000ms. Para órdenes relacionadas con el FRC, será recomendable ampliar este valor para asegurar que todos los dispositivos de la red IQRF tienen tiempo suficiente para enviar su respuesta.

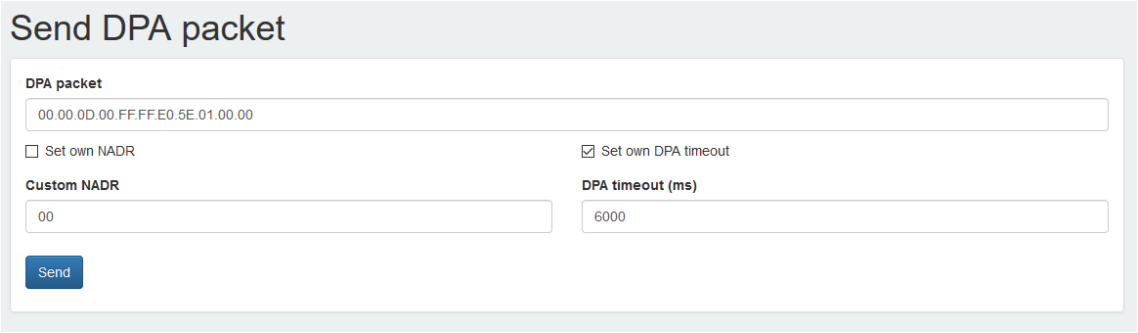


Ilustración 44 - Sección de envío de órdenes

- MQTT broker

Para enviar y recibir solicitudes hacia la red IQRF y recibir los datos más allá del servicio iqrf-daemon, será necesario que el programa que realice la solicitud se comunique con el servicio. Ello se puede conseguir, se emita la orden desde el propio Gateway o desde otro dispositivo, mediante el protocolo MQTT.

El protocolo MQTT (Message Queue Telemetry Transport) es un protocolo machine-to-machine muy usado en el mundo del Internet of Things, especialmente orientado a la transmisión de información de sensores. La tipología de este tipo de comunicación es de estrella, permitiendo hasta 10000 clientes. El nodo central, denominado bróker, es el que transmite los mensajes y se encarga de administrar la red MQTT. Los clientes envían de manera periódica un paquete de datos (PINGREQ) y esperan una respuesta del broker (PINGRESP), que mantiene activa la red. Esta comunicación puede ser cifrada para aumentar la seguridad. El puerto estándar para este protocolo es el 1883, donde se realiza la comunicación sin encriptar.

Esta comunicación se identifica mediante temas o "topics", que determinan la naturaleza de la comunicación. A su vez, estos temas pueden tener subcategorías, declarados de forma jerárquica separados por el símbolo "/" (ej. "Topic/Subtopic").

En este caso específico existirán dos temas: **Iqrf/DpaRequest** e **Iqrf/DpaResponse**, que corresponden al envío de solicitudes y las respuestas,

respectivamente. Más adelante se configurarán en bloques de emisión y recepción MQTT que la utilidad NodeRED posee para esta finalidad.

El bróker se situará en el Gateway, permitiendo recibir y enviar paquetes de información con el protocolo MQTT tanto desde el propio Gateway con el NodeRED, como desde dispositivos o servicios más allá de la red local.

Mosquitto es una utilidad disponible para sistemas linux (Armbian entre ellos), cuya función es crear un bróker de comunicación MQTT, permitiendo configurar salidas y entradas de paquetes de información. Se puede instalar desde la línea de comandos del Gateway ejecutando la siguiente orden:

```
$ sudo apt-get install -y mosquitto mosquitto-clients
```

Tras completarse la instalación, se comprueba que el servicio del bróker está activo:

```
$ systemctl status mosquitto.service
```

Y se pueden observar las comunicaciones realizadas ejecutando uno de los comandos

```
$ mosquitto_sub -t Iqrf/DpaRequest
$ mosquitto_sub -t Iqrf/DpaResponse
```

según sea se quieran ver solicitudes o respuestas, ya que el atributo "–t" corresponde al tema o "topic" de la comunicación.

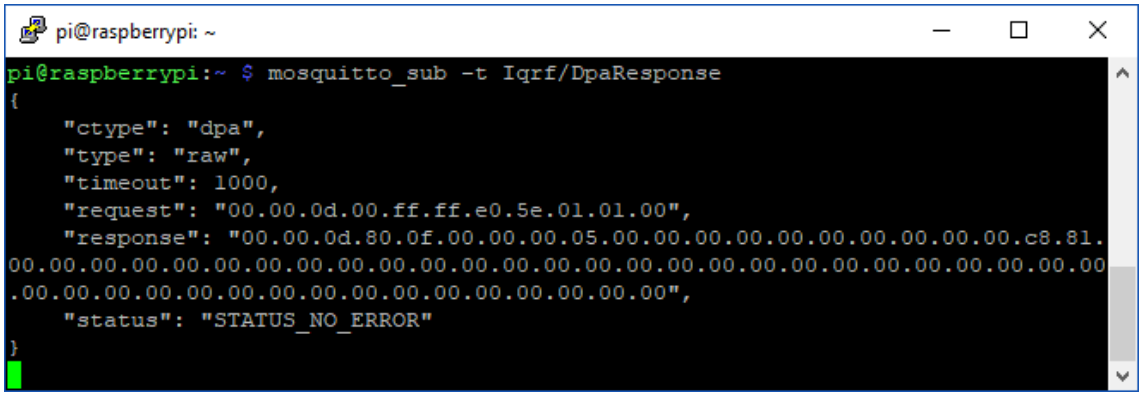


Ilustración 45 - Respuesta MQTT

- NodeRED

Una de las necesidades que se requieren para la monitorización de las variables del entorno, es que la solicitud de temperatura se realice periódicamente. NodeRED es una utilidad destinada a la programación de tareas de una manera completamente visual. Su método de programación basado en diagramas de flujo permite ahorrar mucho tiempo a la hora de programar este tipo de sistemas.

Primero, se requiere el entorno Node.js y el paquete npm para poder instalar el paquete correspondiente a NodeRED. Ejecutando los siguientes comandos se comprueba si ya están instalados:

```
$ node –v
$ npm -v
```

De no estarlo, se pueden instalar mediante:

```
$ sudo apt-get install -y curl
$ curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
$ sudo apt-get install nodejs
```

Repitiendo los comandos de comprobación, se mostrará en pantalla la versión de los paquetes instalados. El siguiente paso será instalar el conjunto de utilidades de NodeRED. Para ello se ejecutan los siguientes comandos en el terminal del Gateway:

```
$ sudo npm install –g –unsafe-perm node-red
$ sudo npm install -g pm2
```

Se sitúa el directorio en el directorio del usuario, en este caso “pi”, y se arranca el NodeRED.

```
$ cd /home/pi
$ pm2 start /usr/bin/node-red --node-args="--max-old-space-size=128"
```

Accediendo a la dirección IP del Gateway en el puerto 1880 (“http://192.168.1.65:1880”), se obtiene la interfaz de edición del NodeRED, donde podrán situarse los flujos que constituirán la programación de este servicio.

Ahora que el servicio está instalado, se configurará para iniciarse automáticamente al arrancar el Gateway. Para ello se ejecutan los siguientes

comandos, que incluyen el servicio en el arranque y añaden el directorio al PATH (variable del sistema):

```
$ pm2 save
$ pm2 startup
$ sudo env PATH=$PATH:/usr/bin /usr/lib/node_modules/pm2/bin/pm2 startup
systemd -u pi --hp /home/pi
```

Se procede a reiniciar el Gateway:

```
$ sudo reboot
```

La conexión SSH se interrumpirá y habrá que conectar nuevamente. Al volver a acceder al terminal del Gateway se ejecuta el siguiente comando para comprobar que el servicio ha arrancado correctamente y se encuentra activo:

```
$ sudo systemctl status pm2-pi
```

Una vez comprobado que funciona correctamente, se vuelve a la interfaz de edición de NodeRED para programar los flujos que compondrán las actividades periódicas del Gateway.

En la aplicación a la que se le va a dar uso, es posible configurarlo en el Gateway con la finalidad de que interactúe con la red IQRF, enviando órdenes, recibiendo datos y procesándolos. Se diferenciarán 3 flujos: Envío, Recepción y Visualización.

- Envío:

Este flujo emite la orden de solicitud de temperatura periódicamente a través del protocolo MQTT, donde lo recibe el iqrf-daemon.

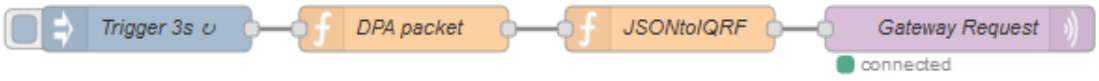


Ilustración 46 - Flujo de envío

En este flujo se sitúan cuatro bloques:

- Disparador: Se programa un disparador que se dispara cada 3 segundos, activando el flujo.

- DPA packet: Crea la orden de solicitud de temperatura en formato JSON.

```
var data={
  type: "raw",
  request: {
    nadr: "0x0000",          //Coordinador
    pnum: "0x0D",            //FRC
    pcmd: "0x00",            //Función 00 (Envío)
    hwpid: "0xFFFF",         //Todos los perfiles
    pdata: "0xE05E010100",   //Solicitud Temperatura
  },
  timeout: 1000  //Tiempo máximo
}
msg.payload=data;
return msg;
```

- JSONtoIQRF: Traduce de JSON a un formato que puede entender el iqrf-daemon. Su código es el siguiente:

```
out = ('000'+Number(msg.payload.request.nadr).toString(16)).slice(-2);
out += "."+(('000'+Number(msg.payload.request.nadr).toString(16)).slice(-4)).substr(0,2);
out += "."+('0'+Number(msg.payload.request.pnum).toString(16)).slice(-2);
out += "."+('0'+Number(msg.payload.request.pcmd).toString(16)).slice(-2);
out += "."+('000'+Number(msg.payload.request.hwpid).toString(16)).slice(-2);
out += "."+(('000'+Number(msg.payload.request.hwpid).toString(16)).slice(-4)).substr(0,2);
msg.payload={ctype:"dpa", type: msg.payload.type, request: out, timeout: msg.payload.timeout};
return msg;
```

- Gateway Request: El último bloque envía la orden al broker MQTT del iqrf-daemon en la dirección de “loopback”.

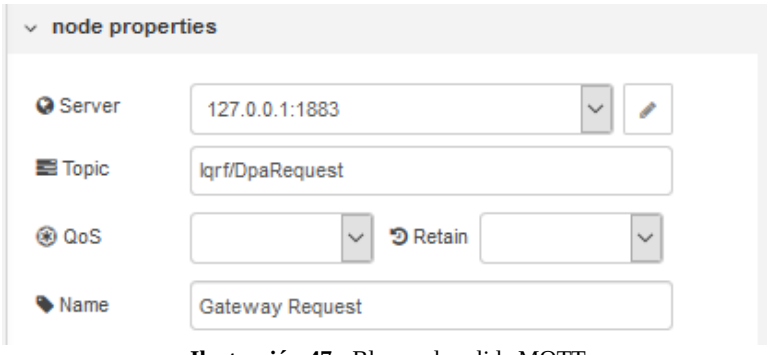


Ilustración 47 - Bloque de salida MQTT

- Recepción:

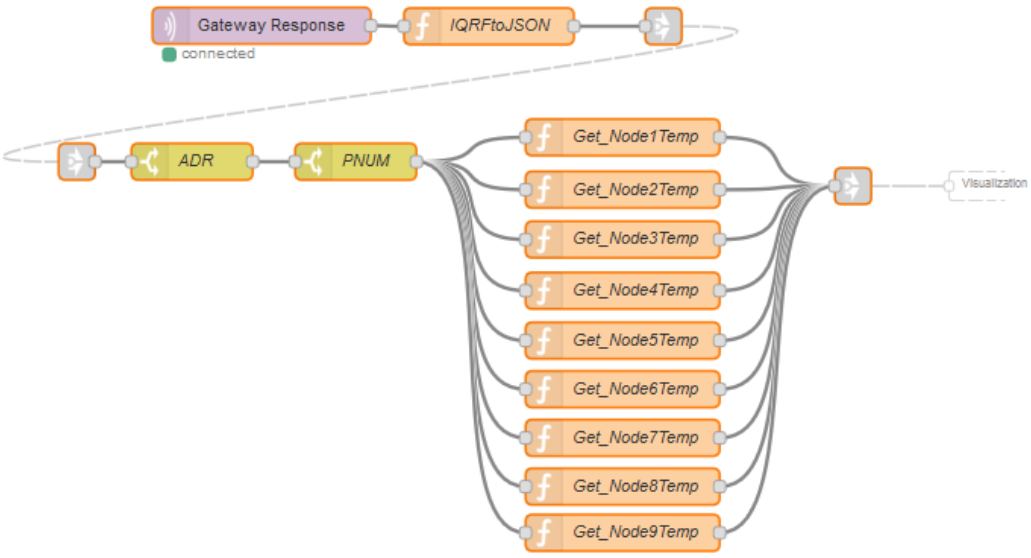


Ilustración 48 - Flujo de recepción

- Gateway Response: El primer bloque recibe la respuesta del puerto MQTT. Se configura igual que el bloque Gateway Request, solo que con un tema diferente.

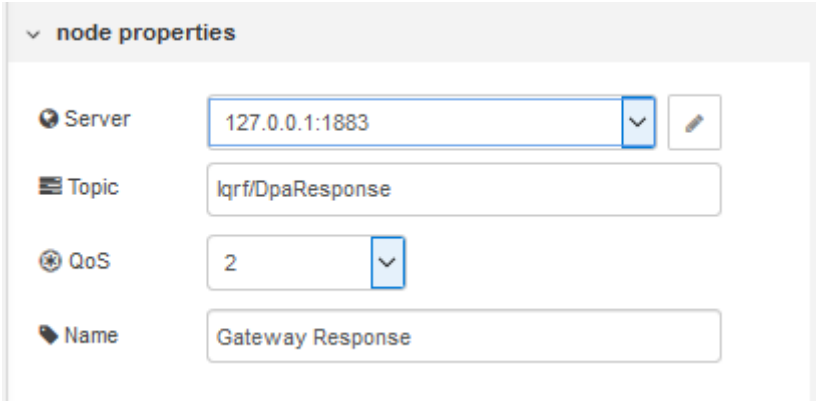


Ilustración 49 - Bloque de Entrada MQTT

- IQRFtoJSON: Esta función transforma la respuesta que devuelve el iqrf-daemon en formato JSON, con el fin de administrar mejor los datos. La función genera una plantilla con la solicitud y la respuesta y a continuación la rellena extrayendo los datos de la respuesta.

```
var response=msg.payload.response.split(".");
var request=msg.payload.request.split(".");
var data={
  type: "",
  request: {
    nadr: "",
    pnum: "",
    pcmd: "",
    hwpid: ""
  },
  response: {
    nadr: "",
    pnum: "",
    pcmd: "",
    hwpid: "",
    errn: "",
    dpa: "",
    pdata: "",
  },
  timeout: "",
  result: "",
  time: new Date()
}
data.type = msg.payload.type;
data.request.nadr=parseInt(request[0], 16) + (parseInt(request[1], 16)*256);
data.request.pnum=parseInt(request[2], 16);
data.request.pcmd=parseInt(request[3], 16);
data.request.hwpid=parseInt(request[4], 16) + (parseInt(request[5], 16)*256);
data.response.nadr=parseInt(response[0], 16) + (parseInt(response[1], 16)*256);
data.response.pnum=parseInt(response[2], 16);
data.response.pcmd=parseInt(response[3], 16);
data.response.hwpid=parseInt(response[4], 16) + (parseInt(response[5], 16)*256);
data.response.errn=parseInt(response[6], 16);
data.response.dpa=parseInt(response[7], 16);
data.response.pdata=msg.payload.response.slice(24).trim();
data.timeout=msg.payload.timeout;
data.result=msg.payload.result;
msg.payload=data;
return msg;
```

- ADR y PNUM: son dos bloques “switch” que filtran si la respuesta no se corresponde con la solicitud realizada.

- Get_NodeXTemp: Estas funciones separan el contenido de cada uno de los nodos para diferenciarlos mejor y convierten los bytes recibidos en el valor real de temperatura, acompañado de una etiqueta que diferencia el nodo al que se refiere.

```
var res=msg.payload.response.pdata.split(".");
parstemp1 = parseInt(res[4],16);  //ff
parstemp2 = parseInt(res[3],16);  //d0
temp1 = parstemp1;
temp2 = parstemp2;
temp1 &= 0x7F;
temp2 &= 0xF0;
temp = (temp1<<4) + (temp2>>4);
temp2 =  parstemp2;
temp2 &= 0x0F;
temp2 *= 0.0625;
temp += temp2;
if(((parstemp1 & 0x80)>>7) != 1)
{
   temp = 2048 - temp;
   temp = -temp;
}
if(temp == -2048)
{
   temp = "Desconectado";
}
msg.payload={"Node1Temp":temp};
return msg;
```

- ToVisualization: Este link envía los datos de cada uno de los nodos al apartado de visualización.

- <u>Visualización:</u>
  El último apartado se encarga de recibir los datos de temperatura y mostrarlas en la interfaz web.

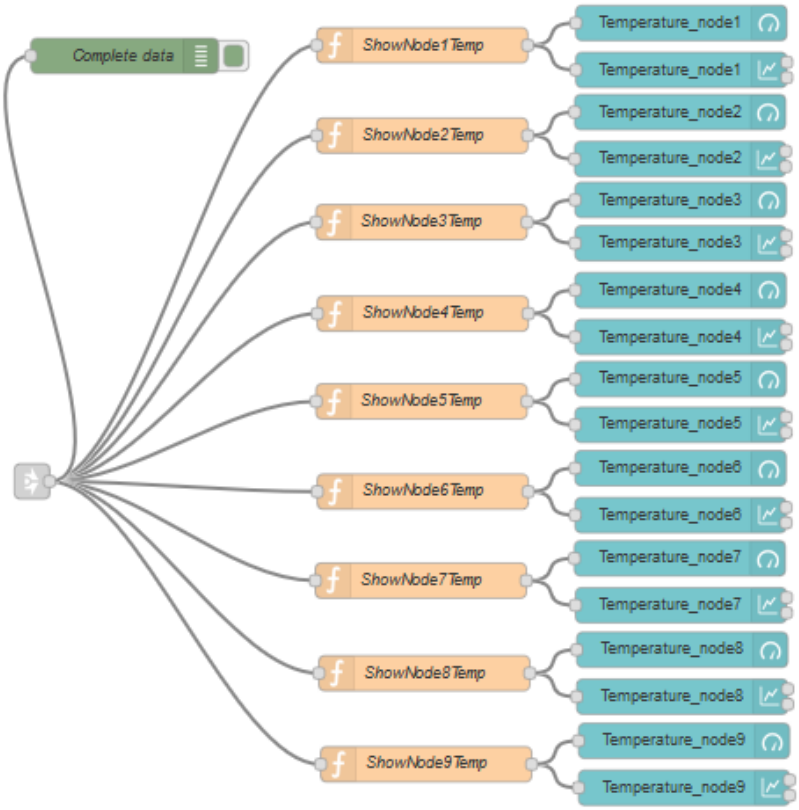


Ilustración 50 - Flujo de visualización

- Complete data: Este bloque sirve para diagnóstico. Muestra los datos en la ventana de debug del NodeRED.
- ShowNodeXTemp: Separa los datos por la etiqueta que contienen y devuelve el valor de temperatura limpio y redondeado a las décimas.

```
msg.payload=msg.payload.Node1Temp;
if(msg.payload > -2000 && msg.payload.Node1Temp !== "ERROR")
{
  msg.payload = Math.round(msg.payload * 10)/10;
  return msg;
}
else
        return msg;
return;
```

- Temperature_nodeX: Son dos bloques por nodo que muestran en la interfaz del NodeRED un indicador para el valor de temperatura y una

gráfica que muestra las últimas medidas. El indicador mostrará el valor de temperatura siempre que el módulo no esté desconectado.

Nota: Para encontrar los bloques correspondientes a la UI, en este caso los indicadores y las gráficas, será necesario añadir el paquete de bloques "node-red-dashboard". Esto se puede hacer desde la interfaz del editor del NodeRED:

☰ -> Settings -> Palette -> Install -> node-red-dashboard.

Añadido el nodo, aparecerán los bloques de visualización.

De esta manera, se ha creado en el Gateway, con la herramienta NodeRED, un sistema que envía solicitudes de temperatura a la red IQRF y recibe los datos, mostrándolos en una interfaz web accesible desde la dirección http://192.168.1.65:1880/ui .

5.8. Administración en la nube

Para facilitar la administración y monitorización de la temperatura de las instalaciones, se ha optado por la creación de una interfaz en la nube que permita visualizar los datos recogidos por los sensores a través de Internet, siendo ésta accesible desde el navegador de cualquier dispositivo. Además, será ésta la que envíe las solicitudes de temperatura y reciba directamente los datos.

Como prueba de concepto, se van a utilizar el conjunto de utilidades en la nube de IBM, llamados Bluemix. Estos servicios disponen de varias tarifas, entre la que se encuentra una versión Lite gratuita, que tiene un límite mensual de tráfico de datos de 256MB. Al comenzar el mes, este límite se reinicia. Además, gracias a las aplicaciones de Cloud Foundry incluidas en el plan, se puede utilizar la utilidad NodeRED. Para eliminar ese límite, se puede contratar una tarifa de pago-por-uso. Para esta aplicación, el coste anual es de aproximadamente 12,20 € mensuales.

Migrando los diagramas de flujo del NodeRED local del Gateway a los servicios en la nube, se consigue simplificar el proceso de programación.

5.8.1. Cuenta IBM

El primer paso será registrar en los servicios web de IBM una empresa, que se usará como cuenta en Bluemix. Este proceso es completamente gratuito seleccionando la tarifa Lite. Una vez creada la cuenta, se procede a registrar un producto. Accediendo a Catálogo, en el apartado Internet de las cosas, se selecciona el producto Internet of Things Platform Starter, se rellena el nombre del servicio y se acepta la versión Lite. En el menú de la izquierda se accede a la sección Panel de control, donde aparecerá la aplicación que se acaba de crear. Al seleccionarlo se acceden a los detalles de la aplicación, que ya estará en funcionamiento.

Seleccionando el servicio IoT de nuestra aplicación, aparecerá la ventana para arrancar el servicio Watson IoT Platform.

En la aplicación Watson IoT Platform se desactivará el protocolo TLS obligatorio, por si alguna de las comunicaciones no puede usar este protocolo. Para ello, en el apartado de Seguridad, se selecciona Seguridad de conexión y se activa TLS opcional.

5.8.2. Registro del Gateway

El siguiente paso será registrar la Raspberry Pi como dispositivo en la nube, para que pueda comunicarse con los servicios web. Para ello, en la aplicación Watson, apartado Dispositivos, se selecciona la pestaña Tipos de dispositivos y se crea uno nuevo. El tipo será dispositivo y el nombre iqrf_gw_device. Una vez creado el tipo, se procede a registrar un nuevo dispositivo. Para ello, en Examinar -> Añadir dispositivo, se elige el tipo de dispositivo “iqrf_gw_device” y se añade el ID de dispositivo “RPI_GW”.

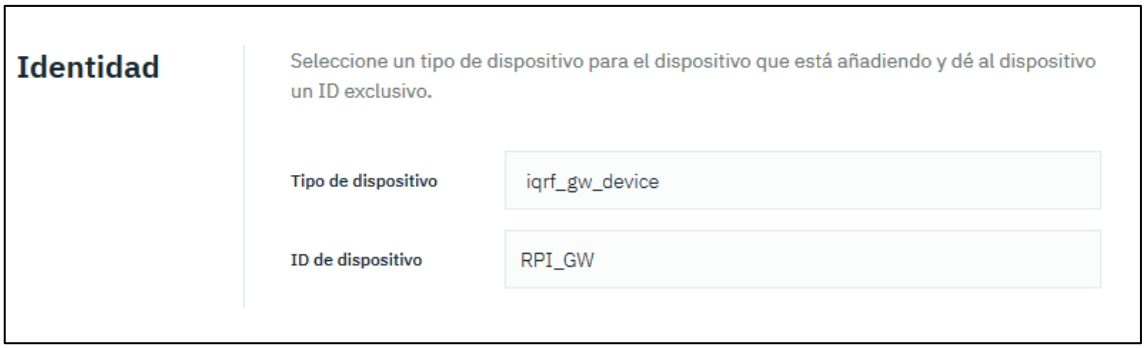


Ilustración 51 - Identidad de dispositivo

Se escribe una señal de autentificación (AUTH TOKEN), el cual es importante apuntar ya que no podrá ser consultado en el futuro y será necesario introducirlo en el Gateway. Se copian los datos y se guarda el dispositivo.

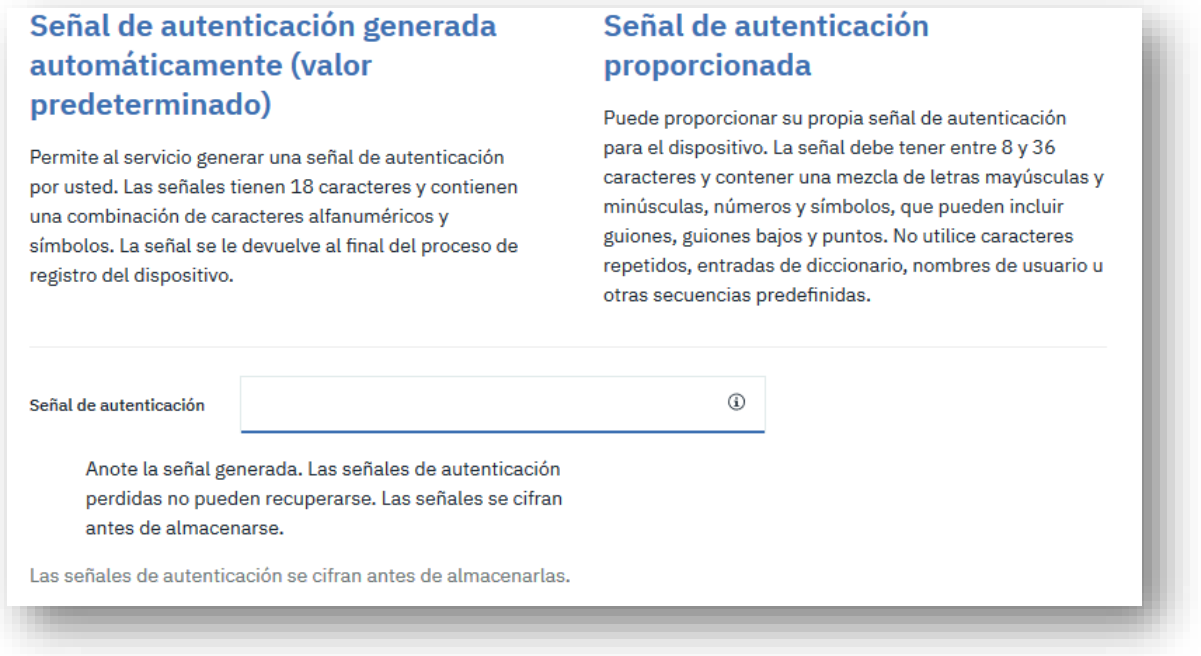


Ilustración 52 - Señal de autentificación

5.8.3. Configuración del Gateway

Para utilizar los servicios en la nube, se desactivará el servicio del Gateway correspondiente al NodeRED, realizando todas las tareas desde la nube a través del protocolo MQTT. Bastará con acceder al Gateway a través de SSH y ejecutar en el terminal los siguientes comandos:

```
$ pm2 unstart
$ sudo systemctl disable nodered.service
```

Para finalizar la comunicación, será necesario registrar en el Gateway las credenciales creadas anteriormente con el fin de conseguir que se autentifique correctamente en los servicios web. Esta tarea se puede realizar desde el propio iqrf-daemon utilizando la interfaz web IQRF Daemon WebApp. Una vez accedido desde el navegador a la dirección IP de la Raspberry Pi, con el usuario y contraseña, se selecciona al apartado Clouds subsección IBM Cloudmix. Una vez ahí, se elige la opción Add

MQTT interface. Aparece un formulario y se introducen los datos previamente generados en el registro del nuevo dispositivo, además del ID de la organización, generado al crear la cuenta de IBM Cloud y como evento se introducirá “iqrf”.

Se guardan los datos y ahora aparece la nueva interfaz IQRF que se acaba de crear. Para iniciar la nueva configuración, se reiniciará el servicio iqrf-daemon, bien ejecutando el comando

```
$ sudo systemctl restart iqrf-daemon.service
```

desde un terminal por SSH o en la interfaz web, seleccionando **Service -> Restart**.

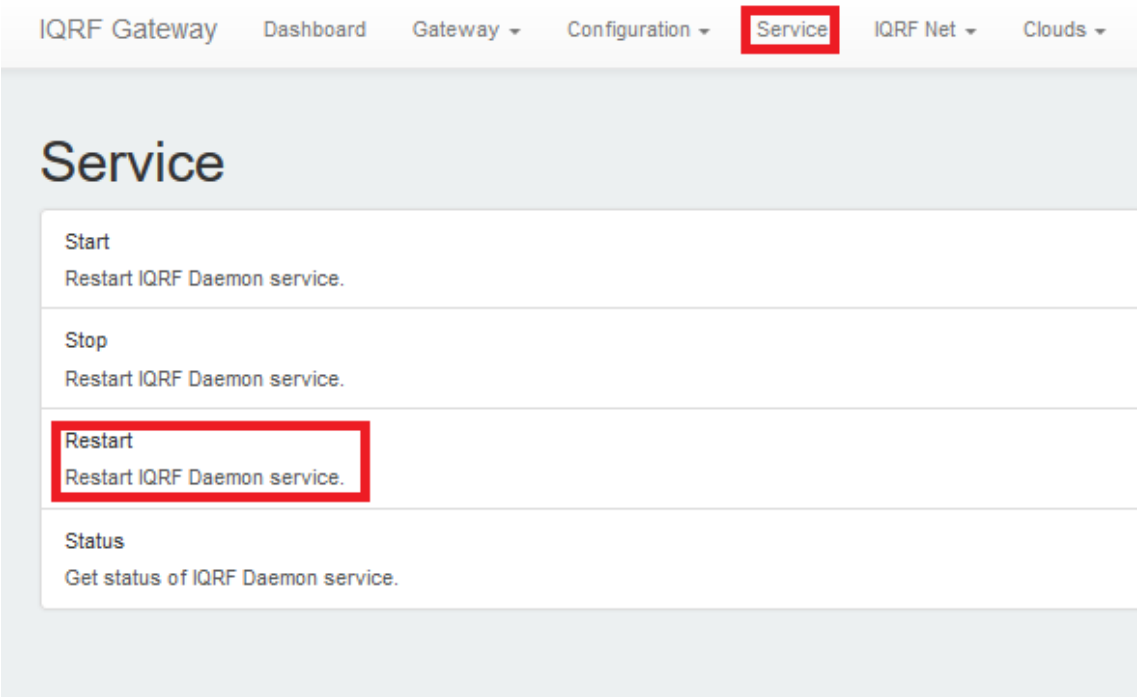


Ilustración 53 - Reinicio del servicio

Desde este momento, desde el Panel de Watson IoT Platform, el dispositivo creado anteriormente figurará como conectado.

Ilustración 54 - Dispositivo en Watson

5.8.4. NodeRED

El servicio NodeRED ahora estará en la nube y se ejecutará desde aquí. A la interfaz de programación del NodeRED se accede mediante la dirección URL correspondiente a la aplicación creada, en este caso https://iqrf-itaca.eu-gb.mybluemix.net/, la cual llevará a una página de presentación de NodeRED. En ella aparece un botón con el texto "Go to your Node-RED flow editor". Haciendo click en este botón, se accederá a la interfaz del editor del NodeRED situado en la nube, que será igual que el editor que se configuró en el Gateway. Previamente, se deberá establecer un usuario y contraseña de acceso al editor, con el fin de proteger los flujos que ejecutará el servicio.

Al igual que en el NodeRED configurado en el Gateway que ahora se ha desactivado, en el editor de la nube se situarán tres flujos diferentes: Envío, Recepción y Visualización.

- Envío: Este flujo se encargará de emitir el paquete DPA desde la nube, pasando por el Gateway y, mediante el servicio iqrf-daemon, llegando a los módulos IQRF. Este proceso se consigue con el siguiente flujo:

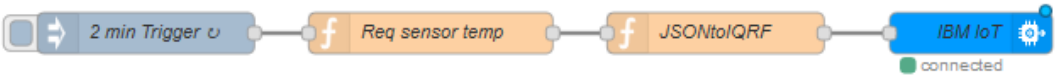


Ilustración 55 - Flujo de envío

  - 2min Trigger: Se trata de un disparador que activa el flujo al que está conectado cada 2 minutos. El hecho de incrementar el tiempo de muestreo se debe a la limitación mensual de 200MB de tráfico de la nube de IBM, la cual se superaría en 2-3 días si se configurara cada 3 segundos como en la interfaz local. Al ser una prueba de concepto y la temperatura una magnitud sin cambios bruscos, se puede utilizar este tiempo de muestreo.

  - Req sensor temp: Este bloque construye en formato JSON el paquete DPA. Su código es idéntico al bloque homónimo configurado en el servicio NodeRED local.
  - JSONtoIQRF: Como su propio nombre indica, este bloque convierte el paquete DPA en formato JSON a un formato adecuado para la interpretación del servicio iqrf-daemon del Gateway. Su código es idéntico al bloque homónimo configurado en el servicio NodeRED local.

- IBM IoT: Se trata de un bloque de salida destinado a las comunicaciones IoT de la nube de IBM. Este bloque es muy importante y debe configurarse de la siguiente forma:

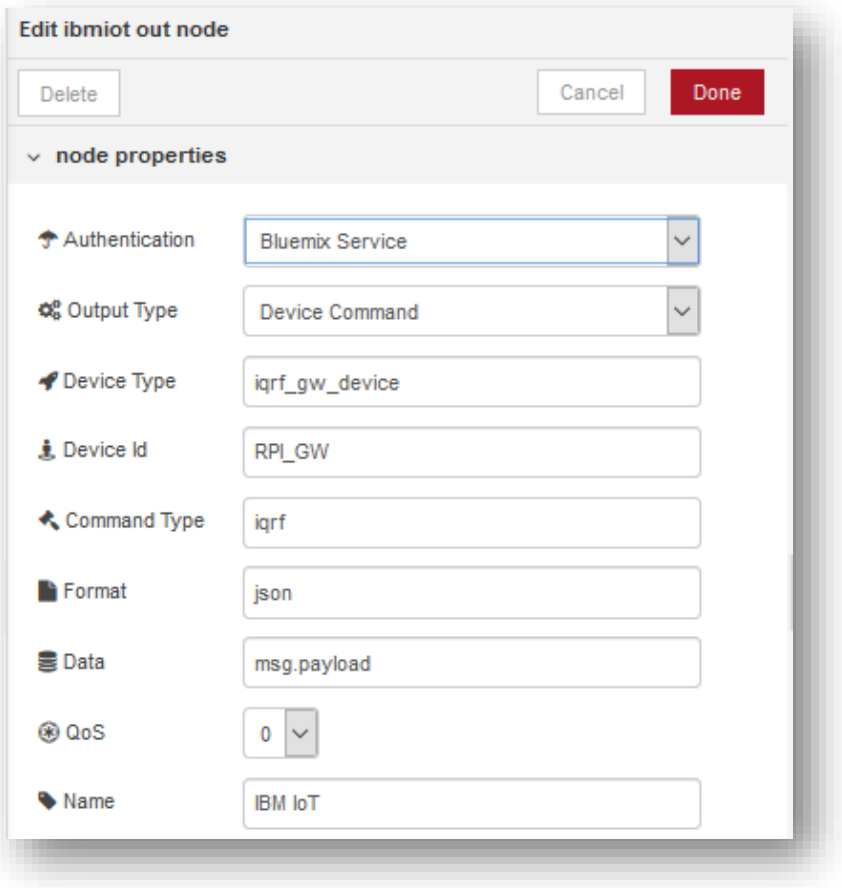


Ilustración 56 - Bloque salida IBM IoT

De esta manera se establece como comando de salida de un dispositivo, en concreto, el dispositivo que se creó anteriormente y que corresponde al Gateway.

- Recepción: Este flujo tiene como finalidad recibir la respuesta de la red IQRF y filtrar y separar los datos de temperatura.

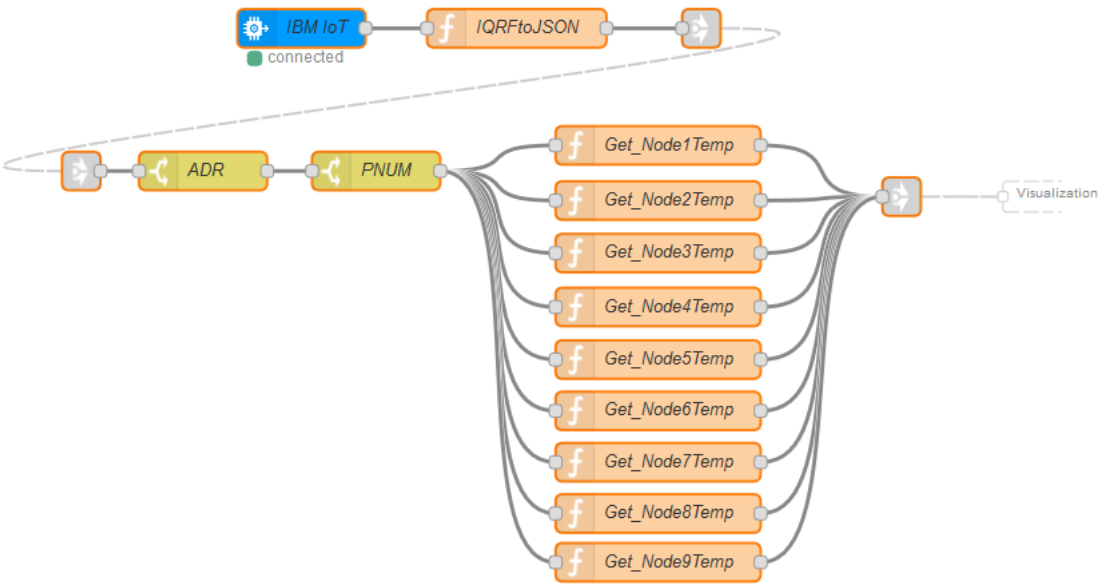


Ilustración 57 - Flujo de recepción

Consta de los siguientes bloques:

- IBM IoT: Se trata de un bloque de entrada destinado a las comunicaciones IoT de la nube de IBM y, en este caso, para conectar con el Gateway. Este bloque es muy importante y debe configurarse de la siguiente forma:

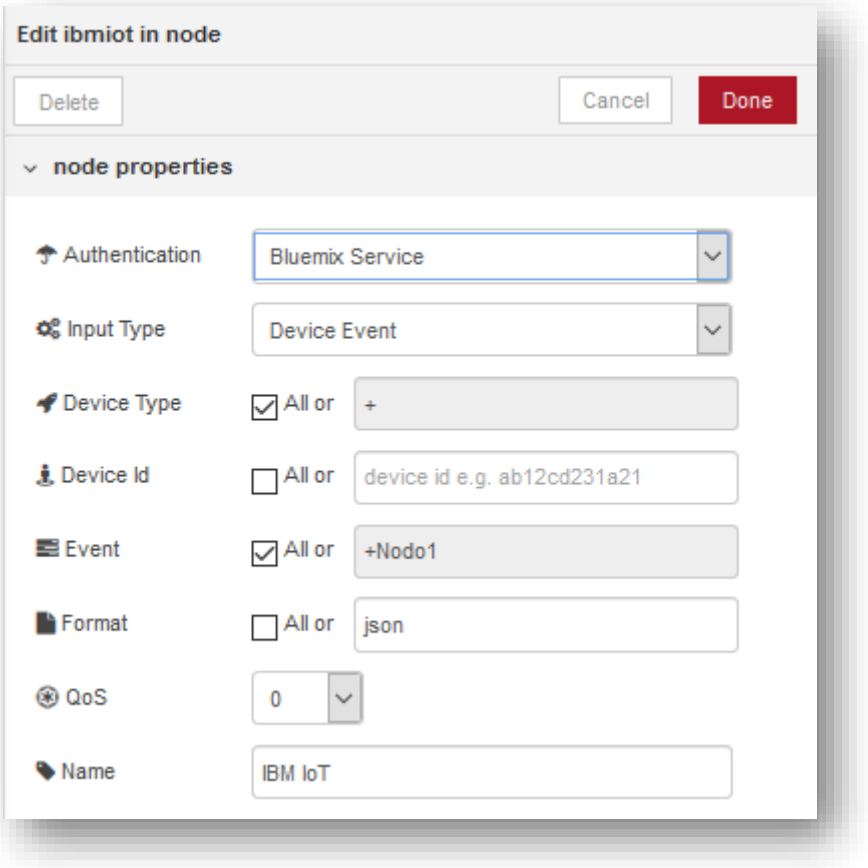


Ilustración 58 - Bloque entrada IBM IoT

En el caso de tener más dispositivos registrados, podría filtrarse la respuesta por el tipo de dispositivo, por el ID del dispositivo o incluso por el evento que tenga el paquete MQTT. Cabe destacar también el tipo de autentificación mediante clave API, la cual podría permitir a otros servicios comunicarse de esta forma con la nube de IBM.

- IQRFtoJSON: Este bloque convierte la respuesta recibida por el bloque anterior en formato IQRF a formato JSON, de manera que sea más manejable en los bloques siguientes.
- ADR y PNUM: Estos dos bloques simplemente filtran las respuestas por la dirección de la red IQRF y el periférico incluidos en la solicitud.

- Get_NodeXTemp: Estos bloques separan y adecúan los datos de temperatura, y le asignan una etiqueta en función del nodo al que corresponden. Su código es idéntico al bloque homónimo configurado en el servicio NodeRED local.
- ToVisualization: Enlace que envía la información del flujo de Recepción al flujo de Visualización.

- Visualización: Este flujo se encarga de mostrar los datos en la interfaz web.

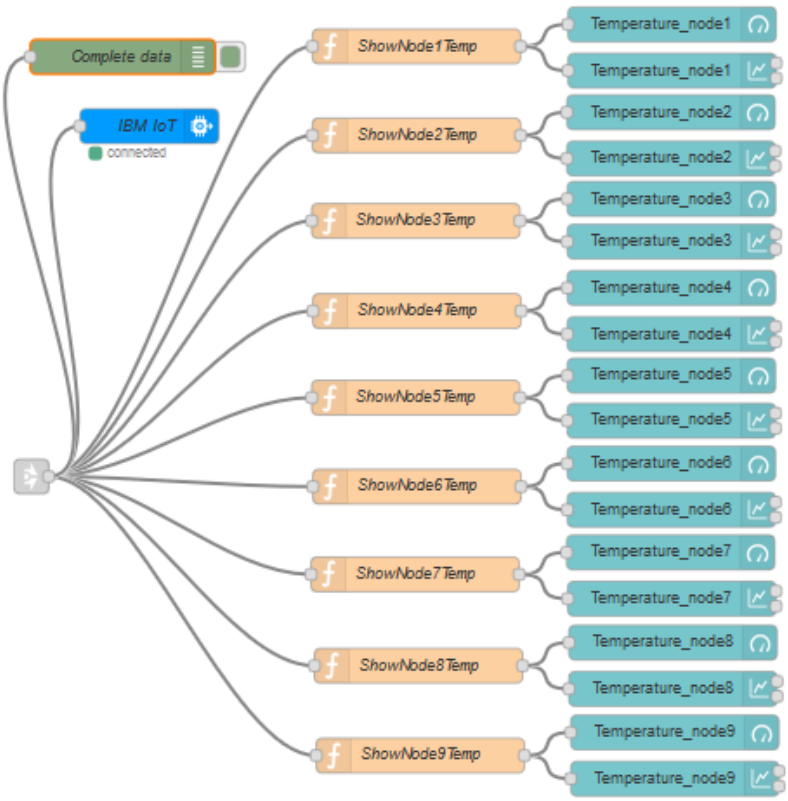


Ilustración 59 - Flujo de Visualización

La información llega desde el flujo de Recepción mediante el enlace, llegando a los siguientes bloques del flujo de Visualización:

- Complete data: Este bloque sirve para diagnóstico. Muestra los datos en la ventana de debug del NodeRED.
- ShowNodeXTemp: Separa los datos por la etiqueta que contienen y devuelve el valor de temperatura limpio y redondeado a las décimas.

```
msg.payload=msg.payload.Node1Temp;
if(msg.payload > -2000 && msg.payload.Node1Temp !== "ERROR")
{
   msg.payload = Math.round(msg.payload * 10)/10;
   return msg;
}
else
            return msg;
return;
```

- Temperature_nodeX: Son dos bloques por nodo que muestran en la interfaz del NodeRED un indicador para el valor de temperatura y una gráfica que muestra las últimas medidas. El indicador mostrará el valor de temperatura siempre que el módulo no esté desconectado.
- IBM Iot: Este bloque de salida servirá para comunicarse con la aplicación Watson, con el fin de transmitir datos y representarlos de una manera diferente. Se configurará de la siguiente forma:

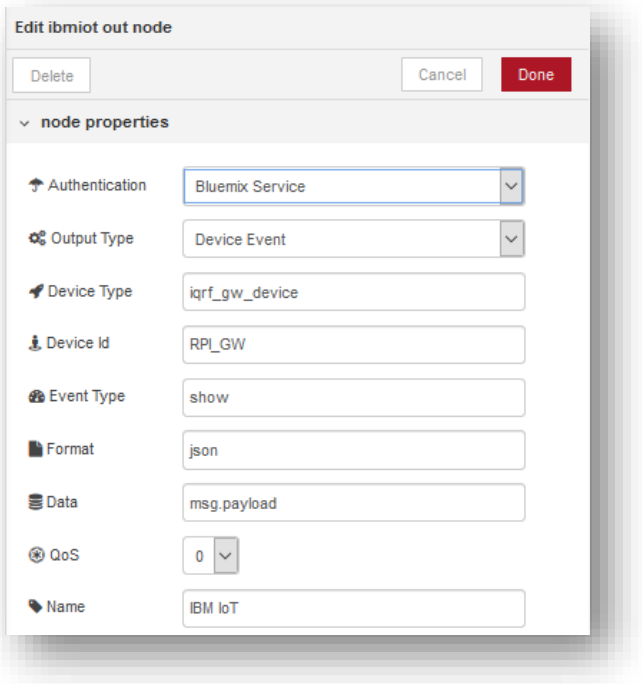


Ilustración 60 - Bloque IBM IoT de salida

El resultado de la interfaz configurada de la manera mencionada anteriormente será la siguiente.

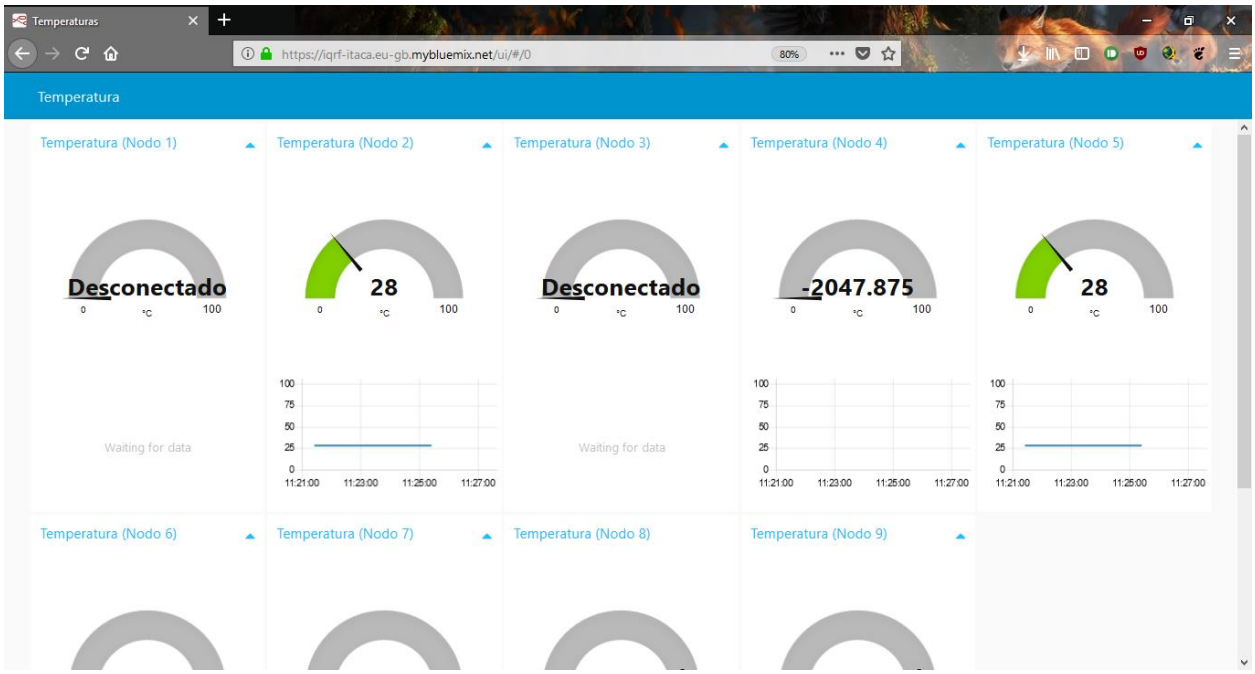


Ilustración 61 - Interfaz Gráfica NodeRED

La interfaz posee nueve indicadores y nueve gráficas que muestran la temperatura recogida por cada uno de los sensores. En este caso, varios módulos están desconectados, mientras que el nodo número cuatro se encuentra conectado a la red IQRF, pero no tiene un sensor acoplado.

5.8.5. Tarjetas Watson

Las tarjetas de la aplicación Watson son otra alternativa para visualizar los datos recogidos por los sensores. Además, con ellas se puede monitorizar el consumo de tráfico y el número de Gateways conectadas, entre otras cosas. Son un conjunto de tarjetas dedicadas a IoT que permiten crear y modificar una interfaz de una manera sencilla, aunque no tiene tanta versatilidad como otras.

Arrastrando una tarjeta de visualización genérica se crea la tarjeta donde se puede visualizar los datos de temperatura. Se selecciona el conjunto de datos con el valor "show" y los nodos que se quieran mostrar. De la misma forma se añadirá una tarjeta de gráfico mostrando esos valores de temperatura y una lista de dispositivos, donde aparecerá el Gateway.

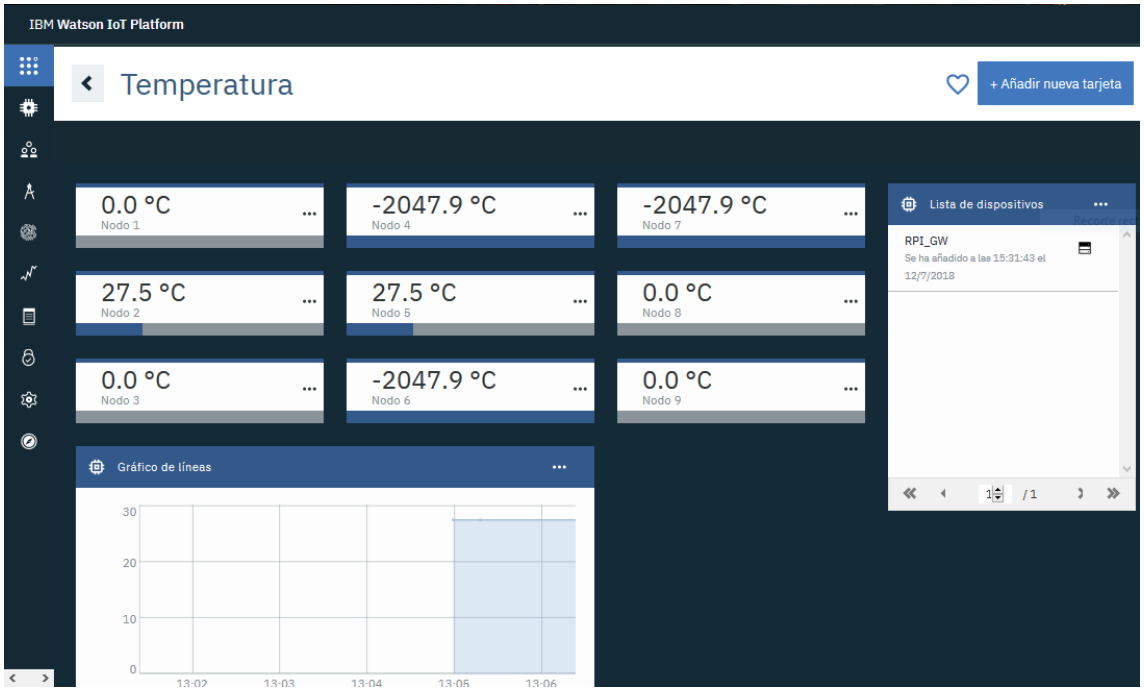


Ilustración 62 - Tarjetas de Watson IoT Platform

De esta manera se obtiene una interfaz fácilmente modificable alternativa a la interfaz del NodeRED, pero que puede coexistir con ella.

6. PRUEBAS DE FUNCIONAMIENTO EN EL LABORATORIO ITACA

Aparte del diseño de este sistema, en este trabajo se incluyen una serie de pruebas realizadas con el fin de asegurar y comprobar los alcances posibles con la tecnología IQRF. Estas pruebas se han realizado con el fin de comparar los datos de rendimiento que IQRF ha publicado con los valores que se consiguen de una aplicación real.

Para las pruebas de distancia, se han obtenido una serie de módulos IQRF de prueba de la empresa Monolitic.

Los dispositivos IQRF o módulos TR disponibles para este trabajo corresponden a la serie TR-72DA, que tienen la posibilidad de trabajar en tres bandas diferentes: 916MHz, 868MHz y 433MHz. Sólo las dos últimas se pueden usar en Europa según la ley, la primera es la banda usada en EEUU. A mayor frecuencia, más fácil es la absorción de la onda por parte de los elementos físicos del entorno que la rodea[40], por lo que se elegirá la banda de menor frecuencia, evitando la tan utilizada banda de 433MHz. Por lo tanto, la banda utilizada para estas pruebas será la de frecuencia de 868Mhz.

Dada la posibilidad de poder modificar la potencia de difusión de los módulos en una escala del 1 al 7, escogeremos la más potente, correspondiente al número 7, con el fin de que los módulos alcancen la mayor distancia posible. La única repercusión que tendrá aumentar la potencia de difusión será un consumo mayor de batería, la cual será mínima dado el bajo consumo de estos dispositivos. La potencia de difusión puede modificarse en la programación de los módulos a través del entorno IQRF IDE, en las opciones del DPA (DPA-config.xml), en el apartado HWP, editando el valor de la etiqueta “TX power”.

Cabe recordar que según el datasheet del módulo DK-EVAL-04A [36], los módulos TR-72DA no son los más adecuados para realizar test de distancia, ya que para ello recomienda otros módulos que integren conector de antena externa, como el TR-72DC o extensores de antena RNG-EXT-01.

Los módulos elegidos se distribuirán por el Laboratorio ITACA y otras zonas de la Ciudad Politécnica de la Innovación (en adelante CPI) de manera experimental, con el fin de evaluar las distancias abarcables por éstos dispositivos. Es importante destacar que este entorno es especialmente desfavorable para las comunicaciones, dada la construcción férrea del edificio y la presencia de numerosas redes de dispositivos en diferentes frecuencias.

Gracias a la colaboración del centro, se pueden distribuir por determinadas áreas los dispositivos y trazar en un mapa la situación de éstos, estimando así, de una manera aproximada y experimental, las distancias que pueden cubrir los módulos IQRF.

Las pruebas realizadas en la CPI comienzan en el bloque A o edificio 8G, situado en un lateral de la CPI, donde se encuentra el laboratorio ÍTACA, y abarcarán un total de 5 plantas. Debido a la limitación del número de módulos disponibles, no se ha podido extender la red a otro edificio de la CPI, aunque sí se llegará a la pasarela que los comunica. Se han realizado 6 pruebas diferentes:

- Prueba 1: Toma de contacto: Se sitúan 3 módulos en la misma sala, uno de ellos dentro de una caja metálica.
- Prueba 2: Máxima extensión en planta. Se distribuyen 6 módulos por toda la planta principal del ITACA y se observa la morfología de la red creada.
- Prueba 3: Respuesta ante obstáculos. Se sitúan 6 módulos en la misma planta, 1 de ellos se va desplazando hasta llegar a otro piso.
- Prueba 4: Máxima extensión por pisos. Se sitúan 6 módulos con el fin de alcanzar la máxima altura posible
- Prueba 5: Estabilidad de la red ante fallos. Estructura de la prueba 4, se elimina cada vez un módulo para observar la respuesta.
- Prueba 6: Extensión del interior al exterior. Se intenta extender la red al exterior, a través de la azotea de la CPI.

A continuación, se muestra el desarrollo extendido de las pruebas.

6.1. Prueba 1

Esta primera prueba se centrará en la comunicación de la tecnología IQRF, aislando uno de los módulos para observar su penetración en metales. Para ello, se sitúa el Nodo 4 en caja de metal a 7 metros del coordinador. A 3 metros del Nodo 4, el Nodo 7.

Ilustración 63 - Prueba 1, apartado 1

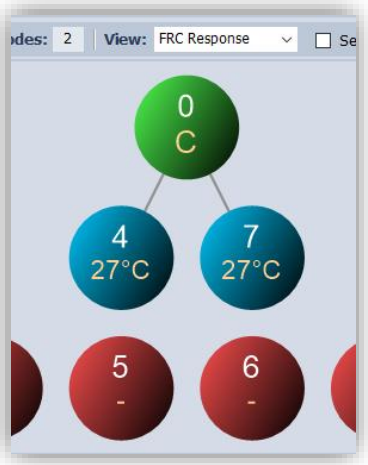


Ilustración 64 - Prueba 1, respuesta apartado 1

Resultado: Conexión directa del Nodo 4 y el Nodo 7 con el coordinador. La caja de metal no afecta apreciablemente en la comunicación.

6.2. Prueba 2

En la segunda prueba se hará un pequeño ensayo en la planta 0 del Ítaca como toma de contacto con el edificio A de la Ciudad Politécnica de la Innovación. Esta prueba es simplemente para tener una referencia de las distancias abarcables por la tecnología IQRF en el edificio.

1. Se sitúa el nodo 4 al otro lado de la pasarela cercana al laboratorio ITACA, al lado de la sala de reuniones. El nodo 7 en el otro extremo del pasillo, más lejos que el nodo 4 del coordinador. Entre el coordinador y los nodos 4 y 7, el nodo 6, colocado a mitad de la pasarela. Los nodos 2 y 5 se encuentran al lado del coordinador.

Ilustración 65 - Prueba 2, pasillo nodo 4

Ilustración 66 - Prueba 2, pasillo nodo 7

Ilustración 67 - Prueba 2, nodo 4

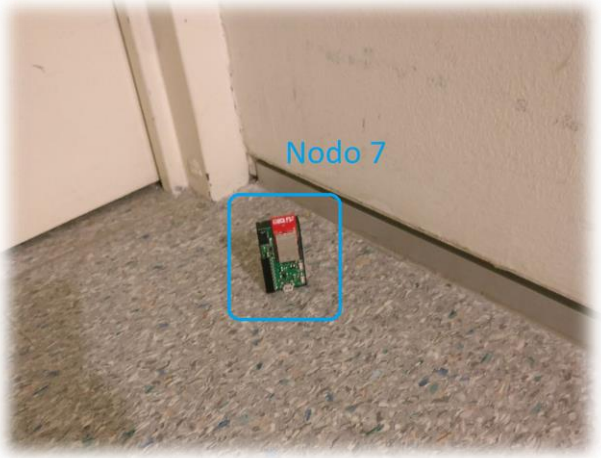


Ilustración 68 - Prueba 2, nodo 7

Ilustración 69 - Prueba 2, nodo 7

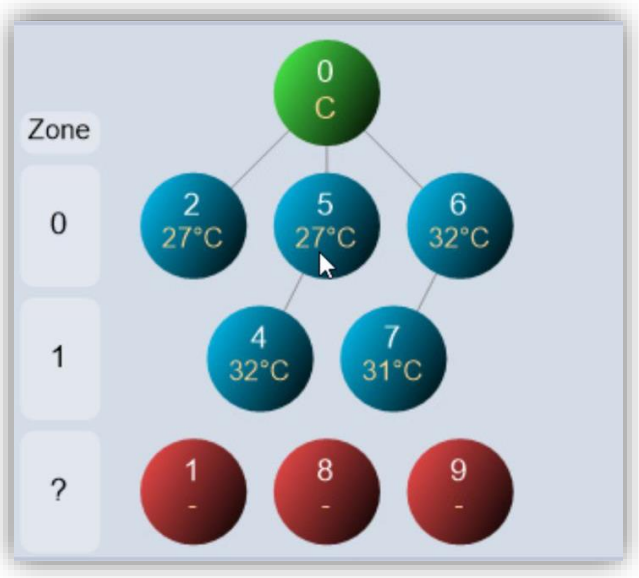


Ilustración 70 - Prueba 2, respuesta apartado 1

Resultado: El nodo 5 repite al 4 y el 6 al 7 sin problemas.

2. Se apaga el nodo 6.

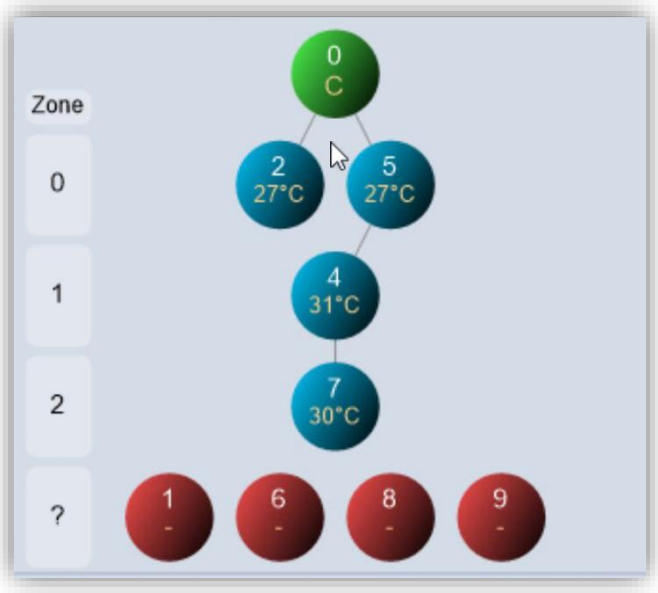


Ilustración 71 - Prueba 2, respuesta apartado 2

Resultado: El nodo 4 llega al 5, situado junto al coordinador y repite al 7.

3. Se apaga el nodo 4.

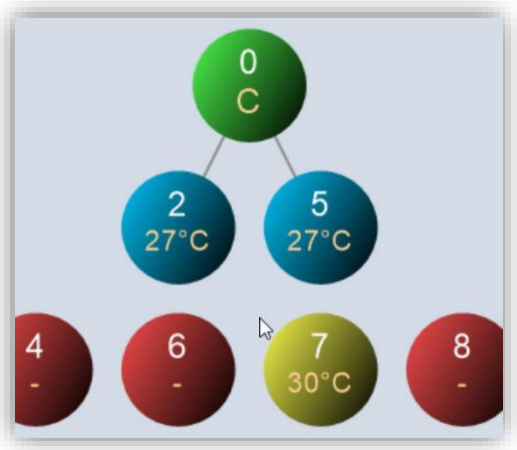


Ilustración 72 - Prueba 2, respuesta apartado 3

Resultado: Nodo 7 tiene problemas de conexión.

6.3. Prueba 3

1. Nodo 2 al lado del coordinador. Nodo 4 al lado de la sala de reuniones y módulo 7 al final del pasillo. Nodo 5 y Nodo 6 en la puerta de la planta.

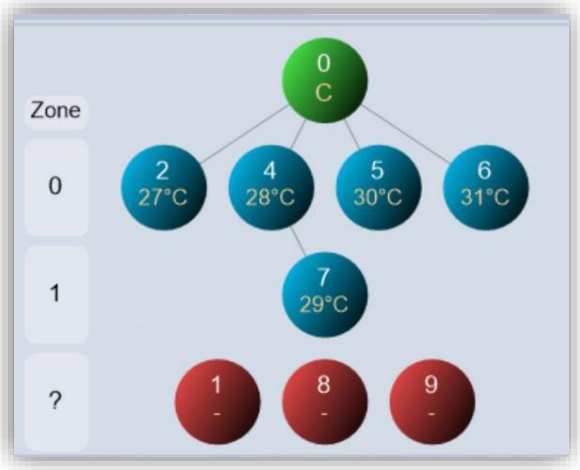


Ilustración 73 - Prueba 3, respuesta apartado 1

Resultado: Nodo 4 repite a nodo 7, el resto conexión directa con el coordinador.

2. Se sube un piso con el nodo 6 (pasillo ascensor).

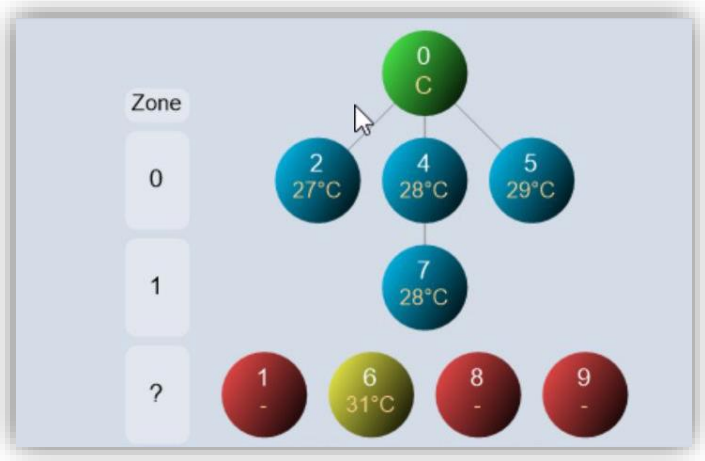


Ilustración 74 - Prueba 3, respuesta apartado 2

Resultado: Nodo 6 con problemas de conexión.

3. Se sube el nodo 6 dos pisos.

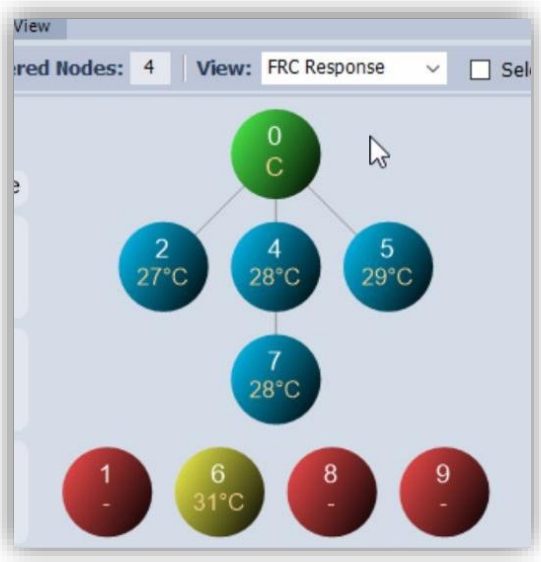


Ilustración 75 - Prueba 3, respuesta apartado 3

Resultado: Nodo 6 con problemas de conexión.

4. Se sube el nodo 6 tres pisos.

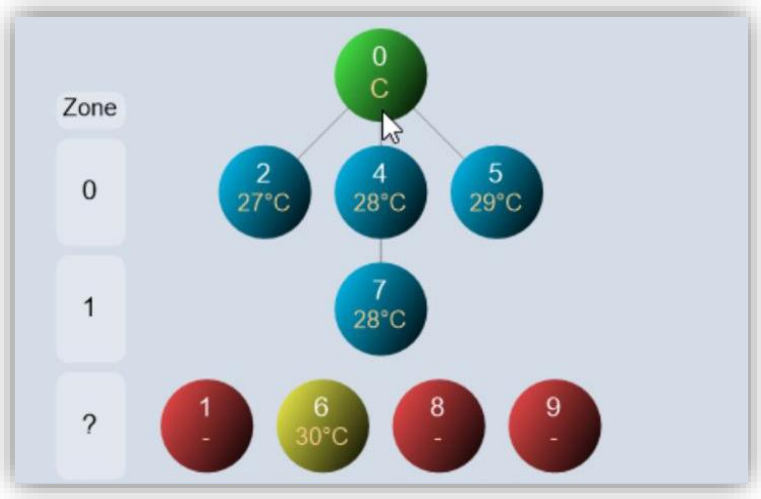


Ilustración 76 - Prueba 3, respuesta apartado 4

Resultado: Nodo 6 con problemas de conexión.

6.4. Prueba 4

1. Coordinador en el laboratorio, como en la prueba anterior. Nodo 2 en el pasillo del ascensor, planta 0.

Ilustración 77 - Prueba 4, nodo 2

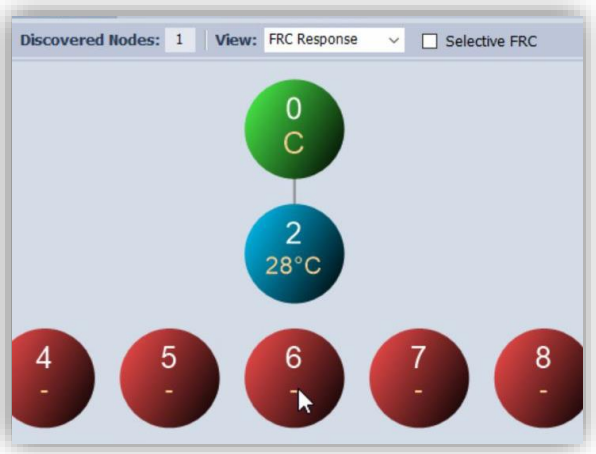


Ilustración 78 - Prueba 4, respuesta apartado 1

Resultado: El coordinador conecta con Nodo 2 (1 nivel, 1 piso).
Nota: el nodo 2 ha tenido que ponerse antes de la puerta de acceso ya que al otro lado se presentaba problemas de conexión.

2. Se añade Nodo 4 al otro lado de la puerta (planta 0).

Ilustración 79 - Prueba 4, nodo 4

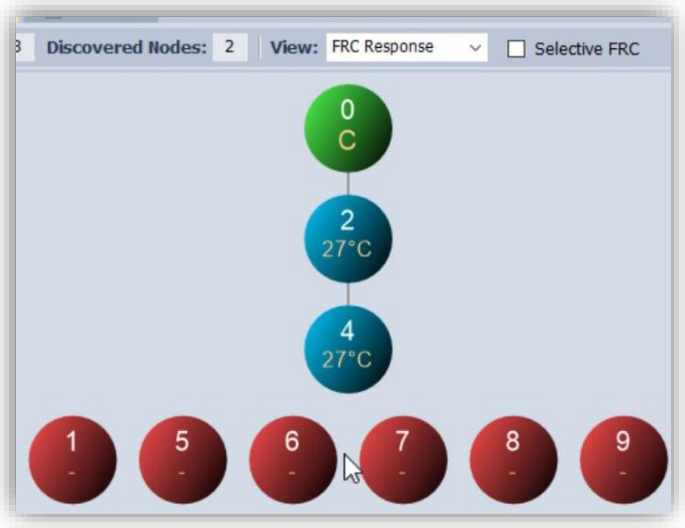


Ilustración 80 - Prueba 4, respuesta apartado 2

Resultado: El coordinador conecta con el Nodo 2 y éste repite al Nodo 4 (2 niveles, 1 pisos).

3. Se añade Nodo 5 sobre nodo 4 en planta 1.

Ilustración 81 - Prueba 4, nodo 5

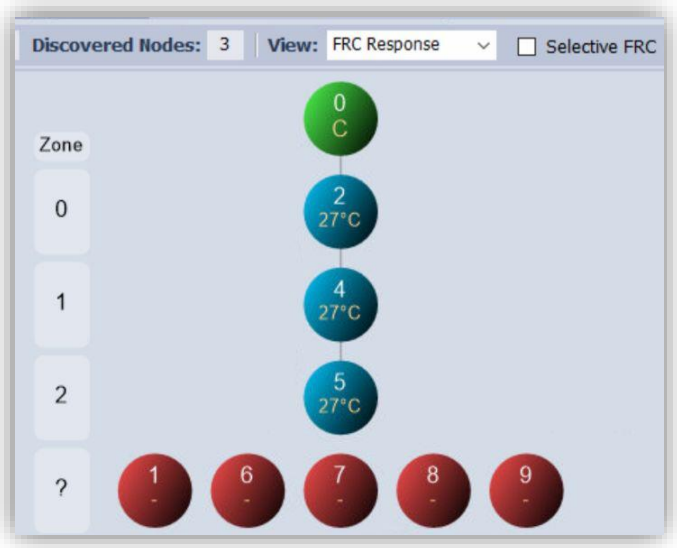


Ilustración 82 - Prueba 4, respuesta apartado 3

Resultado: El nodo 5 conecta con nodo 4, el cual conecta con nodo 2 y éste al coordinador (3 niveles, 2 pisos).

4. Se añade Nodo 6 sobre nodo 5 en planta 2.

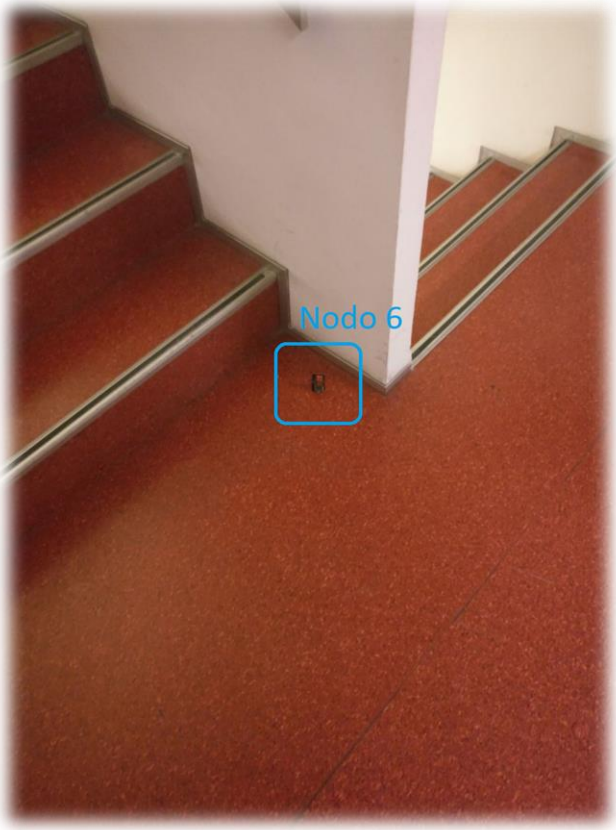


Ilustración 83 - Prueba 4, nodo 6

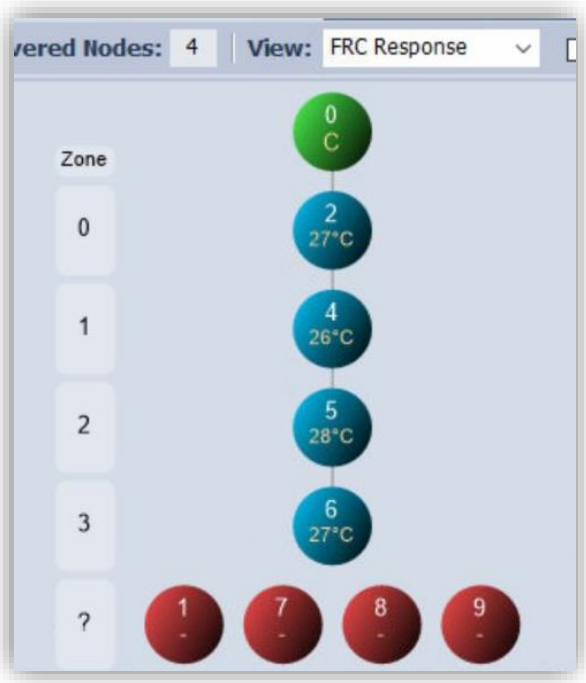


Ilustración 84 - Prueba 4, respuesta apartado 4

Resultado: El nodo 6 conecta con el nodo 5. El nodo 5 conecta con el nodo 4, el cual conecta con nodo 2 y éste al coordinador (4 niveles, 3 pisos).

5. Se añade Nodo 7 sobre nodo 6 en planta 3.

Ilustración 85 - Prueba 4, nodo 7

Ilustración 86 - Prueba 4, respuesta apartado 5

Resultado: El nodo 6 y el nodo 7 conectan con el nodo 5. El nodo 5 conecta con el nodo 4, el cual conecta con nodo 2 y éste al coordinador (4 niveles, 4 pisos).

6. Se recoloca el Nodo 7 en la planta 4 (un piso sin módulo).

Ilustración 87 - Prueba 4, nodo 7

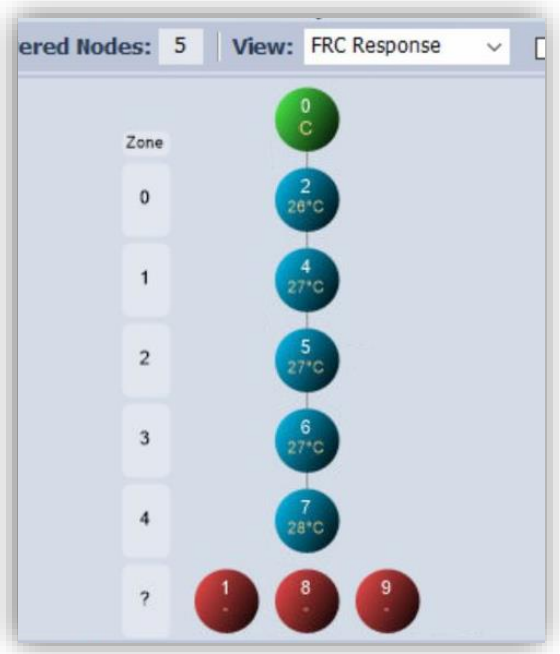


Ilustración 88 - Prueba 4, respuesta apartado 6

Resultado: Cadena de 5 niveles. Entre el nodo 6 y el nodo 7 hay un piso sin nodo (5 niveles, 5 pisos).

6.5. Prueba 5

1. En esta prueba se procederá a comprobar el alcance de los módulos eliminando cada vez uno de la cadena construida anteriormente. Con la cadena anterior, se retira nodo 5:

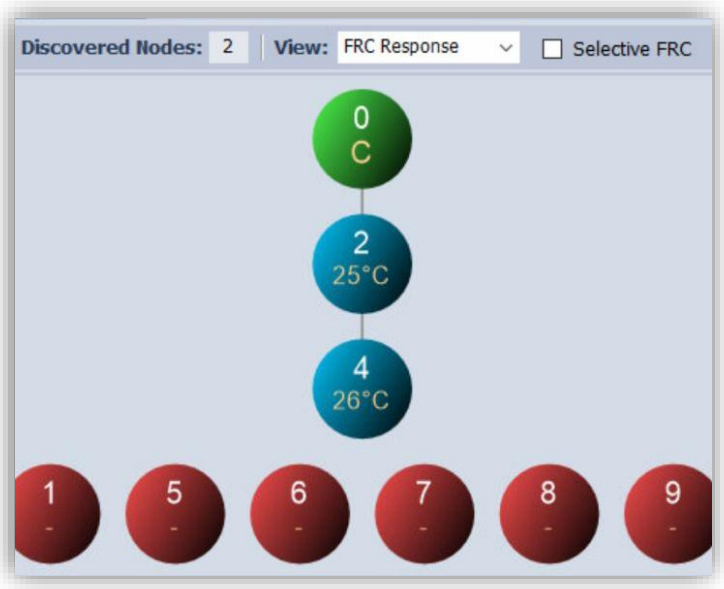


Ilustración 89 - Prueba 5, respuesta apartado 1

Resultado: La cadena se rompe y el nodo 4 no llega al nodo 6.

2. Se repone el nodo 5 y quito el nodo 4.

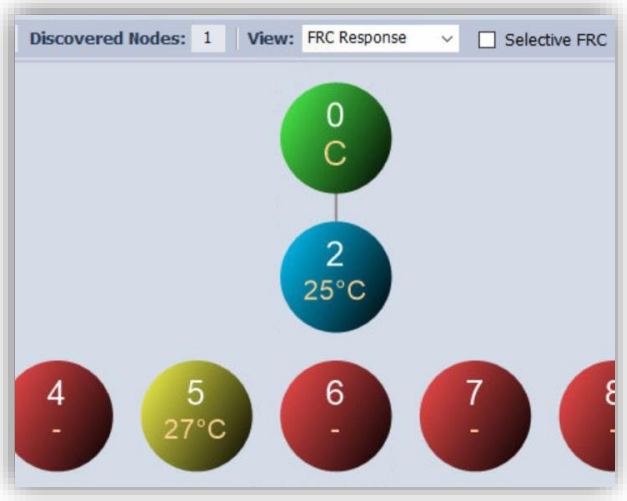


Ilustración 90 - Prueba 5, respuesta apartado 2

Resultado: El nodo 5 presenta problemas de conexión, por lo que se rompe la cadena.

3. Se repone el nodo 4 y se elimina el nodo 2.

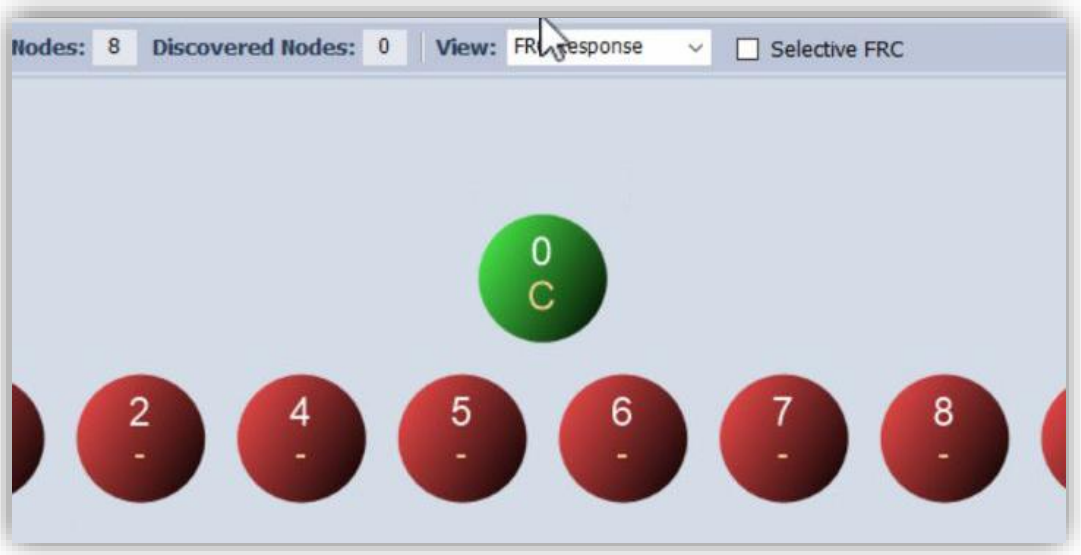


Ilustración 91 - Prueba 5, respuesta apartado 3

Resultado: El coordinador no llega al nodo 4, se rompe completamente la cadena.
Nota: Recordar que el coordinador se encuentra en el laboratorio.

6.6. Prueba 6

El fin de esta prueba es llegar a poner un nodo en la pasarela de la azotea de la ciudad de la innovación. Para ello se situará el nodo 2 en la misma planta que el coordinador, cerca del ascensor. Y se irán situando módulos planta por planta hasta llegar a la azotea.

Se coloca el nodo 2 al lado de la puerta del Ítaca, al lado del ascensor de la planta 0 (1 nivel, 1 piso).

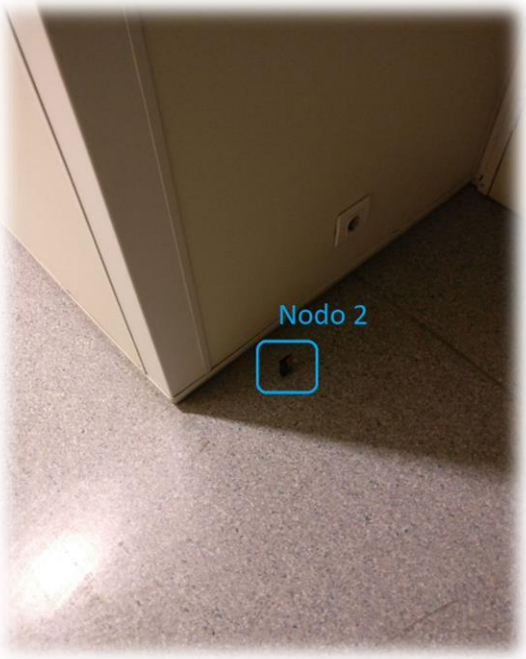


Ilustración 92 - Prueba 6, nodo 2

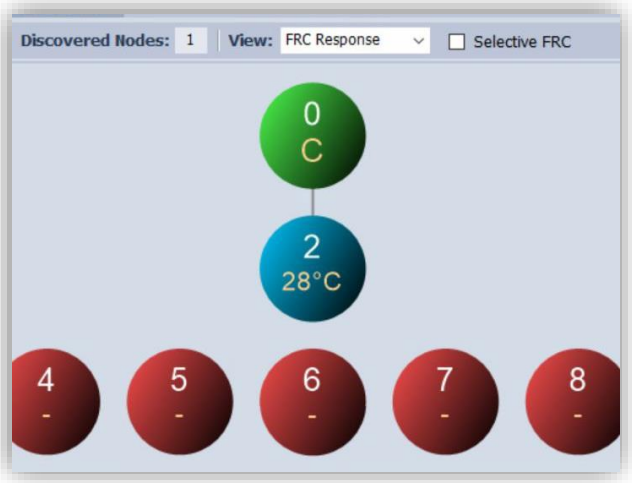


Ilustración 93 - Prueba 6, respuesta apartado 1

1. Se coloca el nodo 4 al otro lado de la puerta en planta 0 (2 niveles, 1 pisos).

Ilustración 94 - Prueba 6, nodo 4

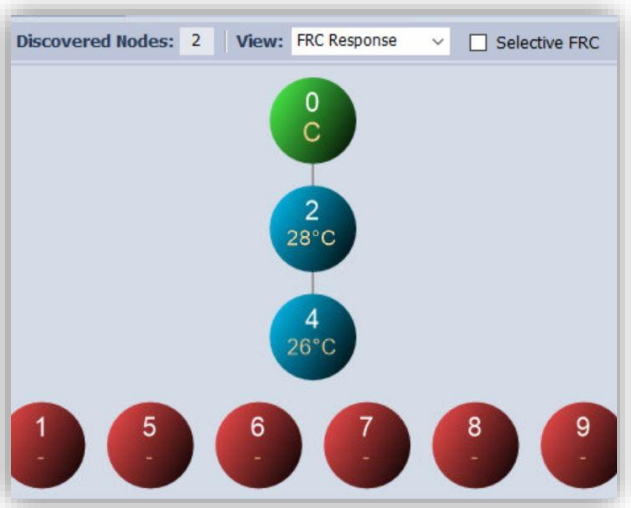


Ilustración 95 - Prueba 6, respuesta apartado 2

2. Se coloca el nodo 5 en el siguiente piso (planta 1), sobre el nodo 4 (3 niveles, 2 pisos).

Ilustración 96 - Prueba 6, nodo 5

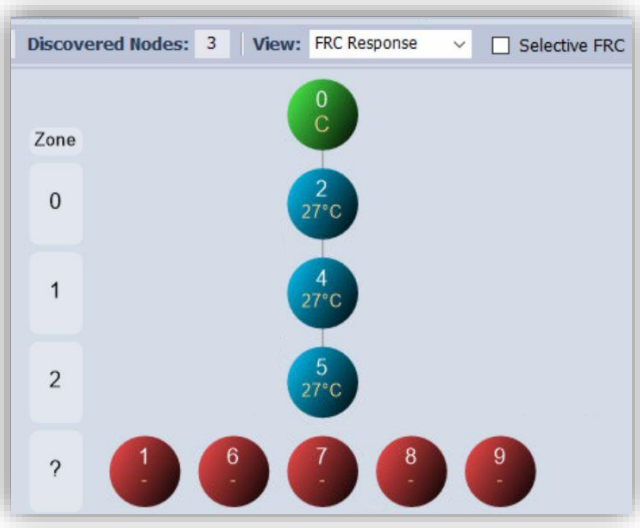


Ilustración 97 - Prueba 6, respuesta apartado 3

3. Se coloca el nodo 6 en el siguiente piso (planta 2), sobre el nodo 5 (4 niveles, 3 pisos).

Ilustración 98 - Prueba 6, nodo 6

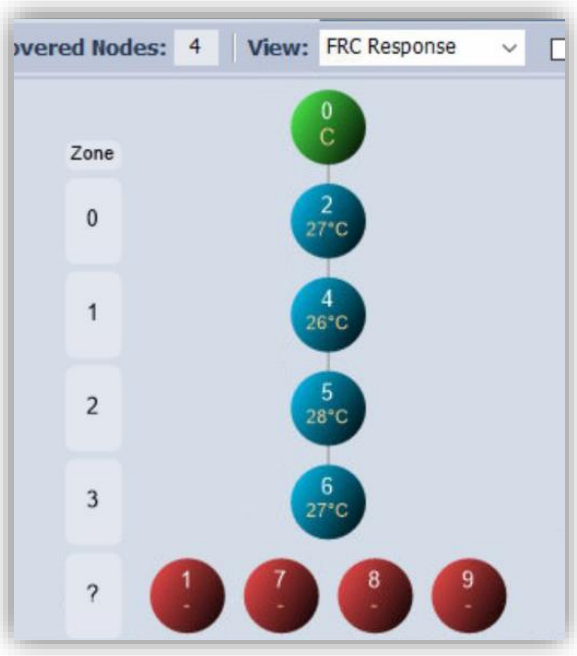


Ilustración 99 - Prueba 6, respuesta apartado 4

4. Se coloca el nodo 7 junto a la puerta de la azotea (planta 3), sobre el nodo 6 (4 niveles, 3 pisos).

Ilustración 100 - Prueba 6, nodo 7

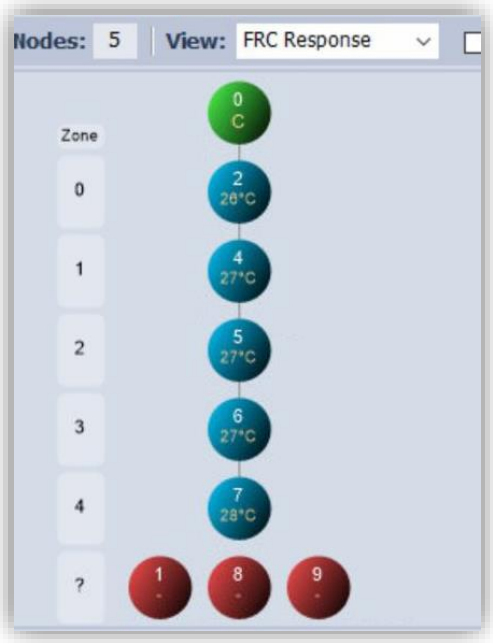


Ilustración 101 - Prueba 6, respuesta apartado 5

5. Se mueve el nodo 7 fuera del edificio.

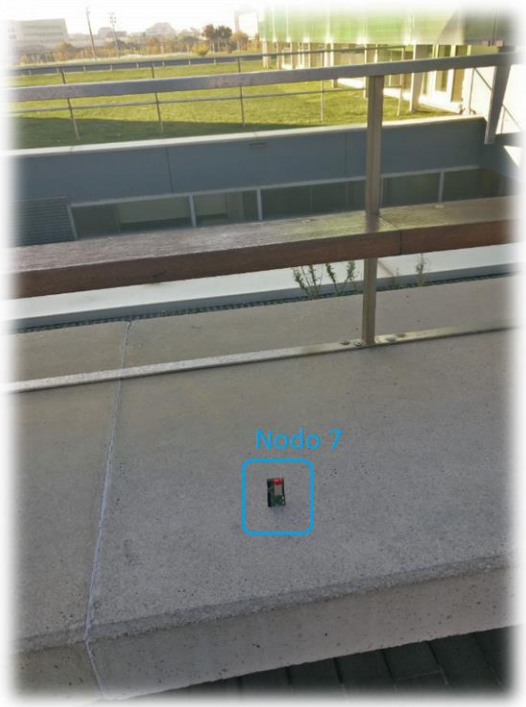


Ilustración 102 - Prueba 6, nodo 7 recolocado

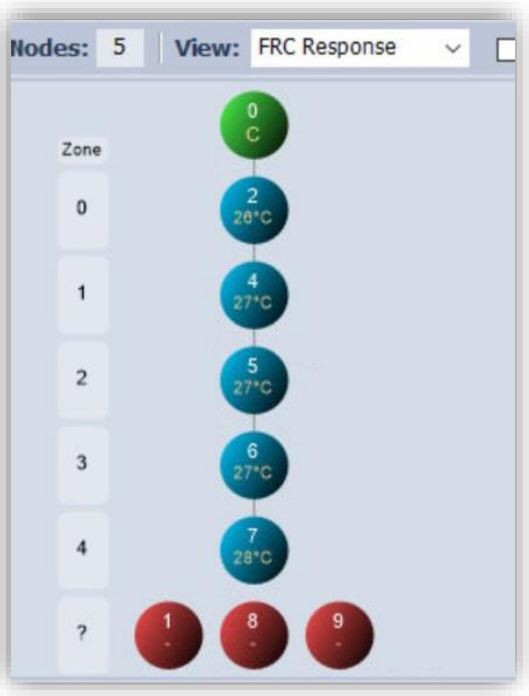


Ilustración 103 - Prueba 6, respuesta apartado 6

6. Se añaden nodos 8 y 9 (carentes de batería) alimentados con baterías externas.

   Debido a la protección antidescarga de las baterías disponibles y al bajo consumo de los módulos, éstas cortan la alimentación a los pocos segundos, haciendo imposible la prueba. Sin embargo, ha dado tiempo a comprobar que efectivamente se pueden usar estos módulos como repetidores.

Ilustración 104 - Prueba 6, nodos 4 y 9

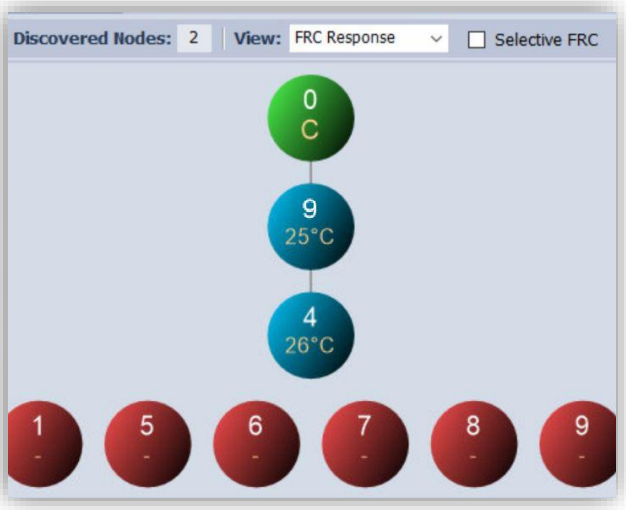


Ilustración 105 - Prueba 4, respuesta apartado 6

6.7. Resultados

De los resultados de las pruebas se han extraído datos que figuran en el anexo de este trabajo, pero se pueden resumir en la siguiente tabla:

Resumen						
Prueba	Nº nodos	Pot. de emisión	Nivel máx. de la red	Distancia máxima sin fallo (m)	Distancia mínima con fallo (m)	Nº máx plantas
1	3	7	2	4,2	-	1/1
2	6	7	4	30,0	35,3	1/1
3	6	7	3	30,7	8,4	1/2
4	6	7	6	14,2	-	5/5
5	6	7	3	14,2	8,4	1/5
6	6	7	6	18	-	4/4

Ilustración 106 - Tabla Pruebas

Al observar los datos pueden parecer paradójicos o incorrectos, pero ello puede explicarse a la existencia de obstáculos y diferentes zonas de la CPI, además de la posición de los módulos.

Como máximas distancias se han alcanzado distancias en planta de hasta 30,7 metros, que no han podido extenderse más allá por falta de espacio. Probablemente podrían haberse extendido más allá de haber sido posible.

Sin embargo, al cambiar de una planta a otra, la comunicación se ve realmente afectada. Esto puede deberse a la posición de los módulos, por lo que debe recalcarse que, para las aplicaciones de estos transceptores, los dispositivos deben colocarse de manera perpendicular a la línea que va desde el módulo al siguiente.

Además, se ha observado una peor comunicación en determinadas plantas, lo que puede concluirse que es debido a la presencia de otras redes inalámbricas en el entorno, las cuales tienen mayor presencia en esas zonas.

En otros experimentos independientes se han logrado distancias entre dos nodos en exterior de 116,6 metros y 15,1 metros en interior, manteniendo al mínimo la potencia de emisión[41].

7. CONCLUSIONES

En este proyecto se ha intentado realizar un producto completamente funcional partiendo desde cero. Se ha intentado también dar visibilidad a tecnologías menos reconocidas en el sector del Internet de las cosas, como la tecnología IQRF, con el fin de reafirmar sus grandes prestaciones y utilidad.

Para realizar este proyecto se han tenido en cuenta conocimientos previos, conocimientos aprendidos en el Grado de Electrónica Industrial y Automática y una parte muy grande de investigación. Este tipo de proyectos permiten al estudiante desarrollarse en campos muy innovadores y que tienen sin duda alguna un gran futuro por delante, como es el campo del IoT y, en especial, de la sensorización. Además, se adquiere formación de distintas áreas, desde la programación de un microcontrolador, la cual está relacionada con el grado, hasta diseño de redes de comunicaciones, programación de sistemas embebidos o tecnologías relacionadas con la nube.

Por otro lado, es un proyecto que puede mejorarse, ampliando el tamaño de la red IQRF, realizar cambios en la programación de las interfaces o reduciendo el consumo y precio del Gateway mediante el uso placas más reducidas y sistemas operativos más livianos. Sin embargo, este proyecto es un prototipo y se ha enfocado en la tecnología IQRF y la comunicación que puede desarrollar, demostrando las ventajas y la competitividad de esta tecnología.

8. BIBLIOGRAFÍA

[1] Alberto Iglesias Fraga, «Conservación del patrimonio a través de la tecnología», *Blog Think Big*, 24-oct-2014. .

[2] Bruno Cendón, «El origen del IoT», *Bruno Cendón: Pensamientos y tecnología*, 16-ene-2017. .

[3] Telefónica S.A., «Smart Patrimonio». [En línea]. Disponible en: http://www.movistar.es/grandes-empresas/soluciones/fichas/SmartPatrimonio.

[4] M. Zaccarini, A. Iannucci, M. Orlandi, M. Vandini, y S. Zambruno, «A multi-disciplinary approach to the preservation of cultural heritage: A case study on the Piazzetta degli Ariani, Ravenna», en *2013 Digital Heritage International Congress (DigitalHeritage)*, Marseille, France, 2013, pp. 337-340.

[5] Arrow Electronics, «Sensores digitales: el camino al futuro», 09-dic-2015.

[6] Maxim Integrated Products, «DS18B20 Datasheet». 2014-2005.

[7] Google, «Estadísticas de búsqueda de Arduino a nivel mundial», Google, a 2019 2004.

[8] Arduino Team, «Introducing the Arduino MKR WAN 1300 and MKR GSM 1400», *Blog Arduino*, 25-sep-2017. .

[9] D. T. Ross McPherson y J. I. GreigPaul, «Optimizing Power Consumption of Wi-Fi for IoT Devices: An MSP430 processor and an ESP-03 chip provide a power-efficient solution», oct. 2016.

[10] STMicroelectronics, «STM32 Ultra Low Power MCUs».

[11] M. R. Palattella *et al.*, «Internet of Things in the 5G Era: Enablers, Architecture, and Business Models», *IEEE J. Select. Areas Commun.*, vol. 34, n.º 3, pp. 510-527, mar. 2016.

[12] Chris DeMartino, «Keeping Watch Over the IoT World», mar. 2019.

[13] Claudio García, «Bluetooth», Ministerio de Educación, Cultura y Deporte, jun. 2005.

[14] Michael Wedd, «Bluetooth IoT Applications: From BLE to Mesh», 17-oct-2018.

[15] J. R. Jeroen Hoebeke y Adnan Shahid, «The Bluetooth Mesh Standard: An Overview and Experimental Evaluation», p. 24, jun. 2018.

[16] A. Kaur, J. Kaur, y G. Singh, «An efficient hybrid topology construction in Zigbee sensor network», en *International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)*, Jaipur, India, 2014, pp. 1-6.

[17] Radio-Electronics.com Team, «What is ZigBee Technology».

[18] L. H. Trinh, V. X. Bui, F. Ferrero, T. Q. K. Nguyen, y M. H. Le, «Signal propagation of LoRa technology using for smart building applications», en *2017 IEEE Conference on Antenna Measurements & Applications (CAMA)*, Tsukuba, Japan, 2017, pp. 381-384.

[19] A. Lavric y V. Popa, «Internet of Things and LoRa™ Low-Power Wide-Area Networks: A survey», en *2017 International Symposium on Signals, Circuits and Systems (ISSCS)*, Iasi, Romania, 2017, pp. 1-5.

[20] T. Bouguera, J.-F. Diouris, J.-J. Chaillout, R. Jaouadi, y G. Andrieux, «Energy Consumption Model for Sensor Nodes Based on LoRa and LoRaWAN», *Sensors*, vol. 18, n.º 7, p. 2104, jun. 2018.
[21] O. Khutsoane, B. Isong, y A. M. Abu-Mahfouz, «IoT devices and applications based on LoRa/LoRaWAN», en *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, Beijing, 2017, pp. 6107-6112.
[22] A. Lavric, A. I. Petrariu, y V. Popa, «Long Range SigFox Communication Protocol Scalability Analysis Under Large-Scale, High-Density Conditions», *IEEE Access*, vol. 7, pp. 35816-35825, 2019.
[23] N. I. Osman y E. B. Abbas, «Simulation and Modelling of LoRa and Sigfox Low Power Wide Area Network Technologies», en *2018 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, Khartoum, 2018, pp. 1-5.
[24] Y. Chung, J. Y. Ahn, y J. Du Huh, «Experiments of A LPWAN Tracking(TR) Platform Based on Sigfox Test Network», en *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju, 2018, pp. 1373-1376.
[25] IQRF Tech s.r.o., «www.iqrf.tech». .
[26] «www.iqrf.org», 15-mar-2012.
[27] IQRF Tech s.r.o, «Module for wireless communication between electric or electronic equipment or systems, method for its control and method for creating generic platforms for user applications in area of wireless communications with those modules.», Czech Republic PUV 18340.
[28] P. Seflova, V. Sulc, J. Pos, y R. Spinar, «IQRF wireless technology utilizing IQMESH protocol», en *2012 35th International Conference on Telecommunications and Signal Processing (TSP)*, Prague, Czech Republic, 2012, pp. 101-104.
[29] R. V. Radek Kuchta y Vladisc Sulc, «Smart wireless communication platform IQRF», University of Technology, Microrisc s.r.o, Czech Republic, ene. 2010.
[30] Francisco Alcalá, «Ventajas de IQRF® en diseños IoT», Monolitic S.A., feb. 2019.
[31] IQRF Tech s.r.o, «A method of accessing the peripherals of a communication device in a wireless network of those communication devices, a communication device to implement that method and a method of creating generic network communication platforms with communication devices.», Czech Republic PUV 18679.
[32] IQRF Tech s.r.o, «IQRF: Fast Response Command®», *IQRF Technology*. .
[33] IQRF Tech s.r.o, «IQRF: Connect to the Cloud», Github, 13-feb-2019.
[34] Igor Oskolkov, «Las implementaciones de VPN y sus peculiaridades», *Kapersky Lab Daily*, 10-mar-2016. .
[35] IEEE International Conference on Mechatronics *et al.*, *17th Mechatronika 2016: proceedings of the 2016 17th International Conference on Mechatronics - Mechatronika (ME) 2016 : Prague, Czech Republic, December 7-9, 2016*. 2016.
[36] IQRF Tech s.r.o., «DK-EVAL-04A Datasheet». 2017.
[37] Microchip Technology Inc., «MCP73831/2 Datasheet». 2014-2005.
[38] IQRF Tech s.r.o., «IQRF - KON-SIM-02». [En línea]. Disponible en: https://www.iqrf.org/products/accessories/connectors/kon-sim-02.

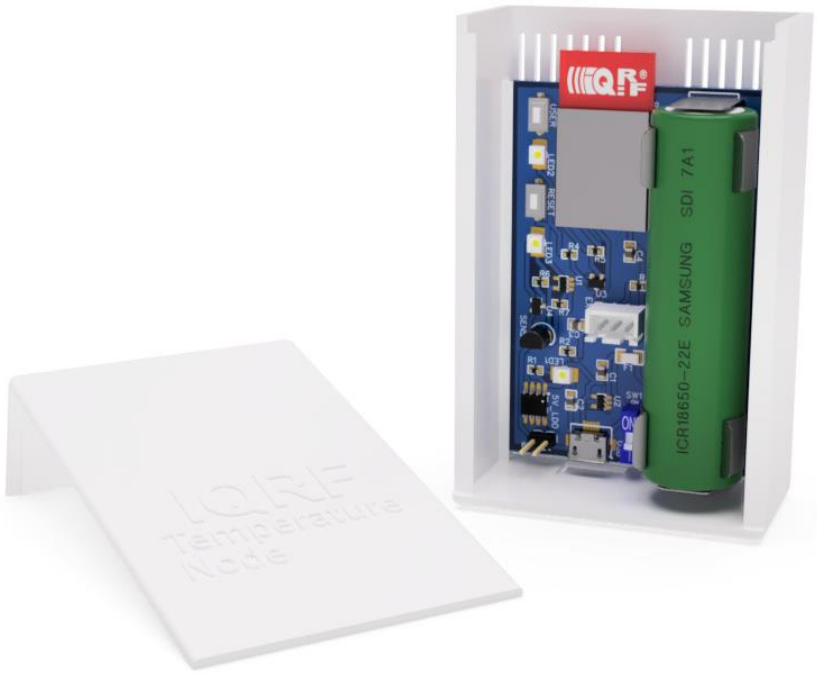
[39] G. Khan, M. Rajalingam, N. Chandrasekaran, M. Tholath, y V. Diwakar, «Thermal studies on battery packs with different geometric configuration of 18650 cells», en *2017 IEEE Transportation Electrification Conference (ITEC-India)*, Pune, 2017, pp. 1-6.
[40] Mariana Molina Matute, «Estudio de banda de frecuencias Sub-GHz para redes de sensores inalámbricas e implementación en plataforma modular», Proyecto Fin de Máster, Universidad Politécnica de Madrid, Madrid, 2011.
[41] V. Sulc, R. Kuchta, y R. Vrba, «IQRF Smart House - A Case Study», en *2010 Third International Conference on Advances in Mesh Networks*, Venice, Italy, 2010, pp. 103-108.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

MONITORIZACIÓN DEL PATRIMONIO CULTURAL MEDIANTE TECNOLOGÍA IOT IQRF

2. Planos



Autor: D. Miguel Andrés Sánchez Galdón

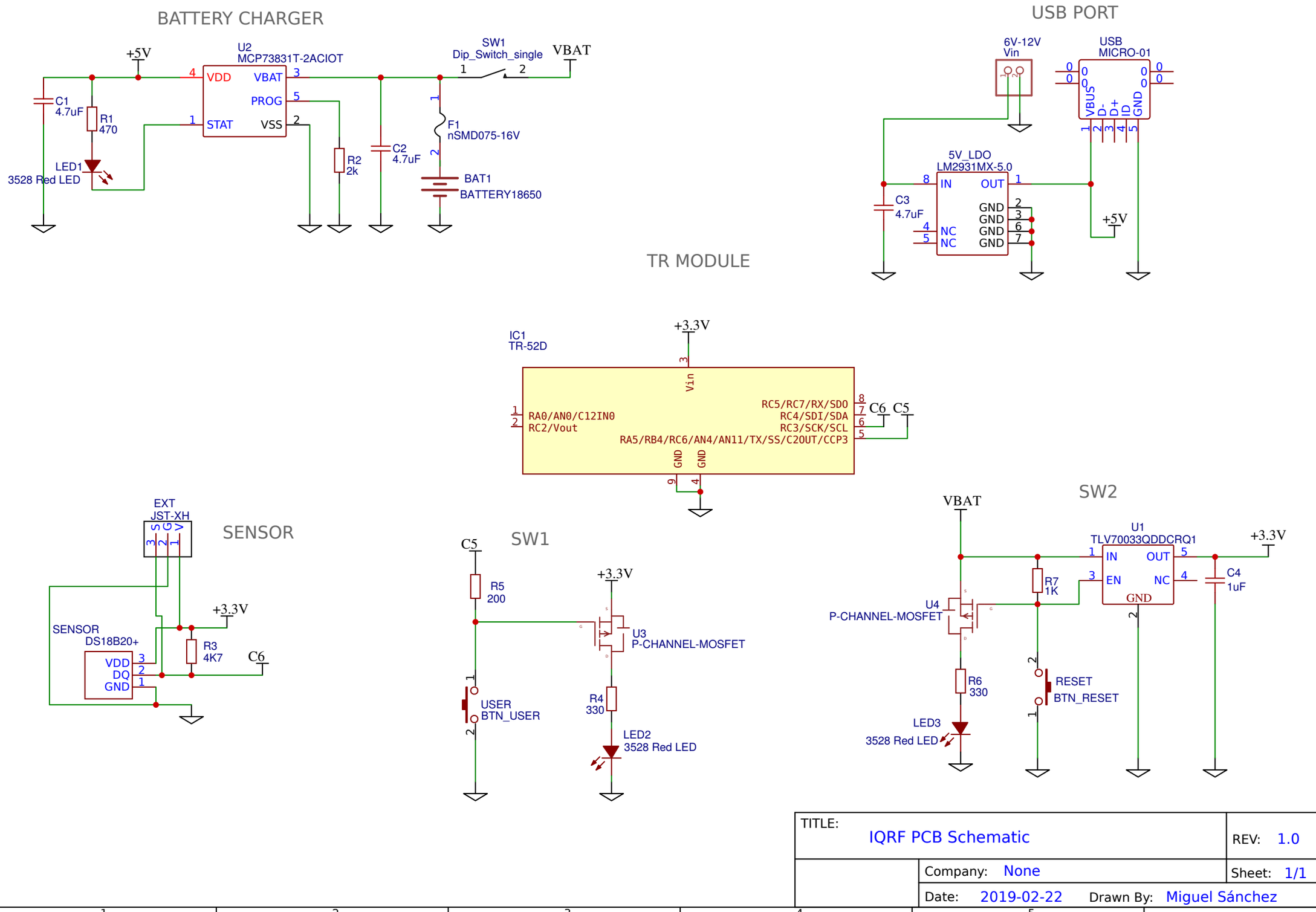
Tutor: D. Ángel Perles Ivars

Valencia, julio de 2019

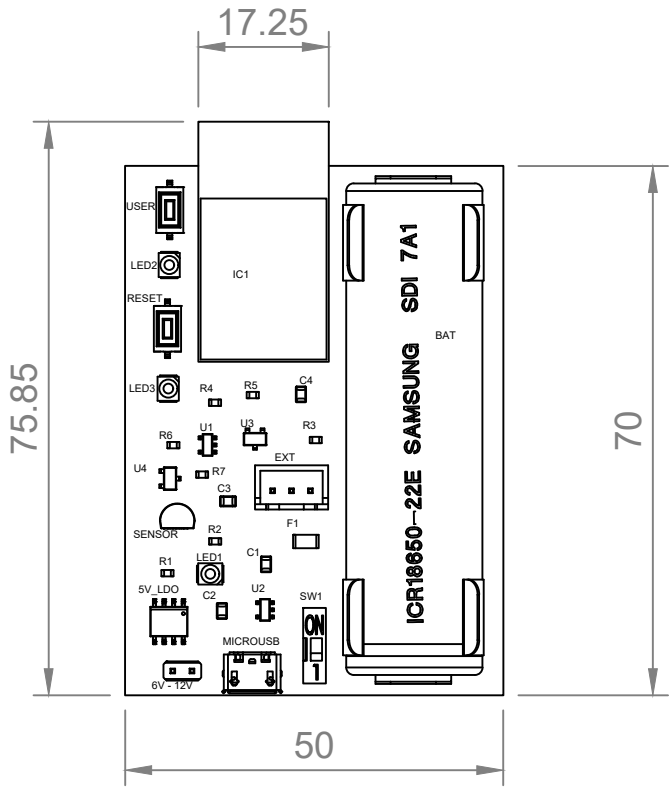
ÍNDICE DE PLANOS

1. REVISIONES

Versión	Fecha	Autor	Descripción
0,1	08/06/2019	M. Sánchez	Versión inicial



TITLE: IQRF PCB Schematic		REV: 1.0
	Company: None	Sheet: 1/1
	Date: 2019-02-22	Drawn By: Miguel Sánchez



IQRF Temp. sensor PCB

Assembly Top

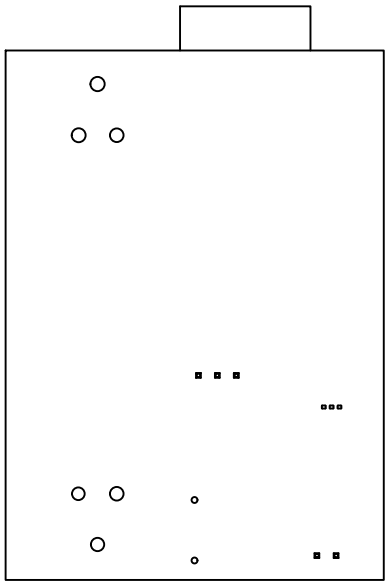
ESCALA	CARACT.	FORM.
1:1	Peso:	A4
	Material:	

	Nombre y firma	Fecha
DIBUJ	M. Sánchez	08/06/19
VERIF		
APROB		
FABR.		
CALID		

Las cotas se expresan en milímetros si no se indica lo contrario.
No cambiar la escala.

Nº Dibujo:

Nº Referencia:



IQRF Temp. sensor PCB

Assembly Bottom

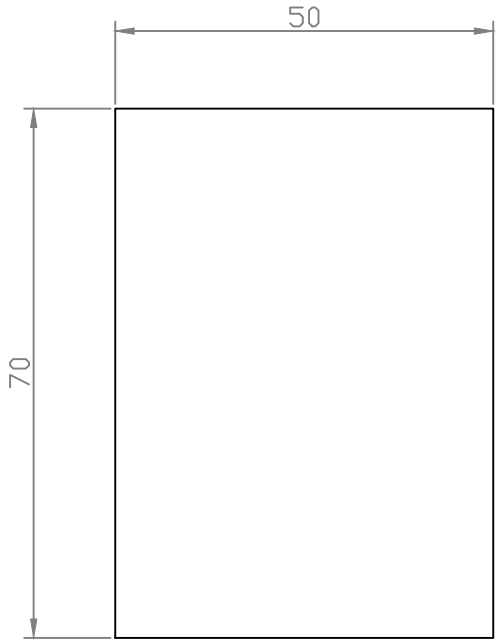
ESCALA	CARACT.	FORM.
1:1	Peso:	A4
	Material:	

	Nombre y firma	Fecha
DIBUJ	M. Sánchez	08/06/19
VERIF		
APROB		
FABR.		
CALID		

Las cotas se expresan en milímetros si no se indica lo contrario.
No cambiar la escala.

Nº Dibujo:

Nº Referencia:



IQRF Temp. sensor PCB

Board outline

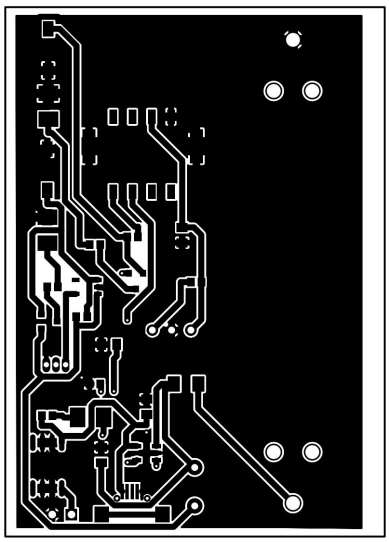
ESCALA	CARACT.	FORM.
1:1	Peso:	A4
	Material:	

	Nombre y firma	Fecha
DIBUJ	M. Sánchez	08/06/19
VERIF		
APRØB		
FABR.		
CALID		

Las cotas se expresan en milímetros si no se indica lo contrario.
No cambiar la escala.

Nº Dibujo:

Nº Referencia:



IQRF Temp. sensor PCB

Copper Top

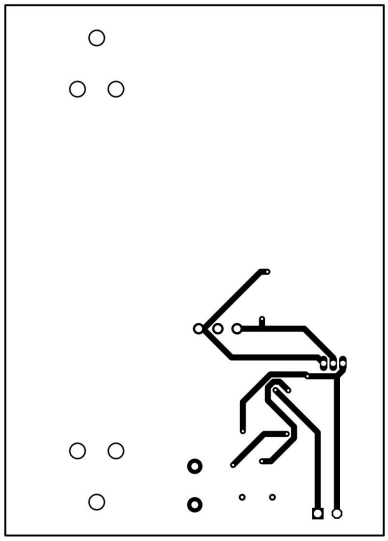
ESCALA	CARACT.	FORM.
1:1	Peso:	A4
	Material:	

	Nombre y firma	Fecha
DIBUJ	M. Sánchez	08/06/19
VERIF		
APROB		
FABR.		
CALID		

Las cotas se expresan en milímetros si no se indica lo contrario.
No cambiar la escala.

Nº Dibujo:

Nº Referencia:



IQRF Temp. sensor PCB

Copper Bottom

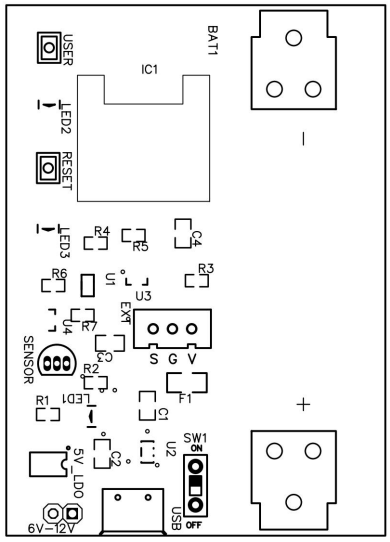
ESCALA	CARACT.	FORM.
1:1	Peso:	A4
	Material:	

	Nombre y firma	Fecha
DIBUJ	M. Sánchez	08/06/19
VERIF		
APROB		
FABR.		
CALID		

Las cotas se expresan en milímetros si no se indica lo contrario.
No cambiar la escala.

Nº Dibujo:

Nº Referencia:



IQRF Temp. sensor PCB

Silkscreen Top

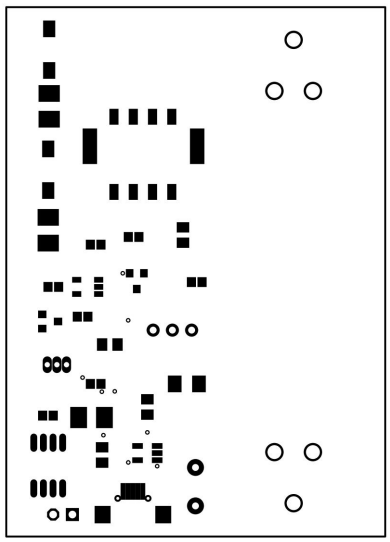
ESCALA	CARACT.	FORM.
1:1	Peso:	A4
	Material:	

	Nombre y firma	Fecha
DIBUJ	M. Sánchez	08/06/19
VERIF		
APROB		
FABR.		
CALID		

Las cotas se expresan en milímetros si no se indica lo contrario.
No cambiar la escala.

Nº Dibujo:

Nº Referencia:



IQRF Temp. sensor PCB

Soldermask Top

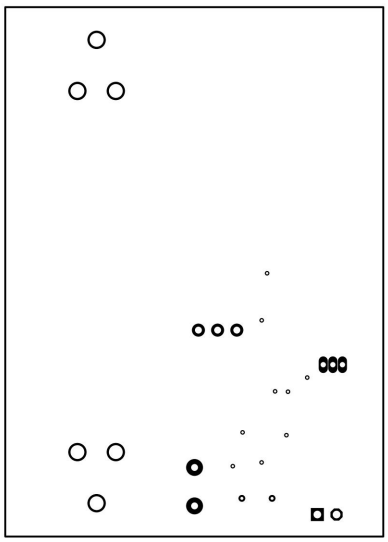
ESCALA	CARACT.	FORM.
1:1	Peso:	A4
	Material:	

	Nombre y firma	Fecha
DIBUJ	M. Sánchez	08/06/19
VERIF		
APROB		
FABR.		
CALID		

Las cotas se expresan en milímetros si no se indica lo contrario.
No cambiar la escala.

Nº Dibujo:

Nº Referencia:



IQRF Temp. sensor PCB

Soldermask Bottom

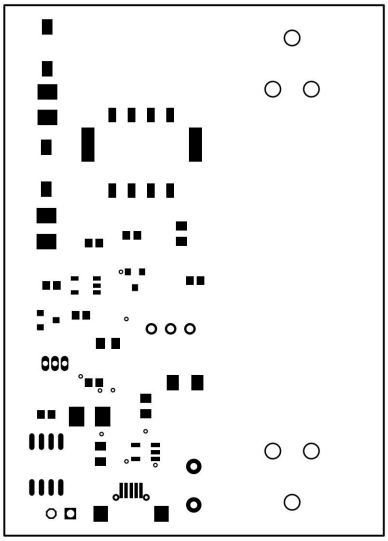
ESCALA	CARACT.	FORM.
1:1	Peso:	A4
	Material:	

	Nombre y firma	Fecha
DIBUJ	M. Sánchez	08/06/19
VERIF		
APROB		
FABR.		
CALID		

Las cotas se expresan en milímetros si no se indica lo contrario.
No cambiar la escala.

Nº Dibujo:

Nº Referencia:



IQRF Temp. sensor PCB

Solderpaste Top

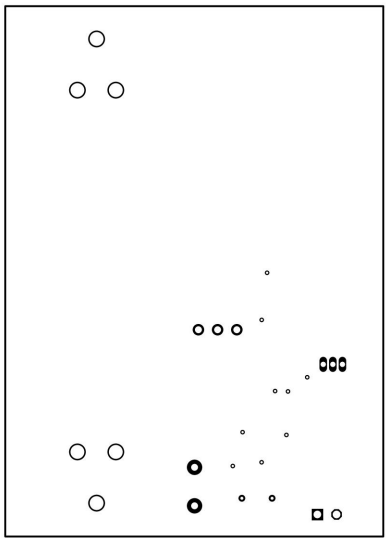
ESCALA	CARACT.	FORM.
1:1	Peso:	A4
	Material:	

	Nombre y firma	Fecha
DIBUJ	M. Sánchez	08/06/19
VERIF		
APROB		
FABR.		
CALID		

Las cotas se expresan en milímetros si no se indica lo contrario.
No cambiar la escala.

Nº Dibujo:

Nº Referencia:



IQRF Temp. sensor PCB

Solderpaste Bottom

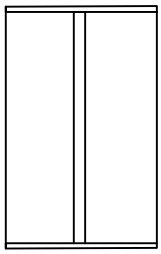
ESCALA	CARACT.	FORM.
1:1	Peso:	A4
	Material:	

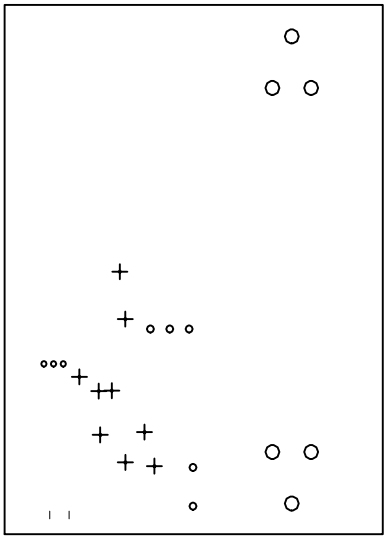
	Nombre y firma	Fecha
DIBUJ	M. Sánchez	08/06/19
VERIF		
APRÛB		
FABR.		
CALID		

Las cotas se expresan en milímetros si no se indica lo contrario.
No cambiar la escala.

Nº Dibujo:

Nº Referencia:





DRILL CHART: TOP TO BOTTOM				
FIGURE	SIZE	TOLERANCE	PLATED	QTY
o	1.829	±0.1829	PLATED	6
'	1.016	±0.1016	PLATED	2
°	0.915	±0.0915	PLATED	5
°	0.701	±0.0701	PLATED	3
+	0.310	±0.031	PLATED	9

IQRF Temp. sensor PCB

Hole Drilling

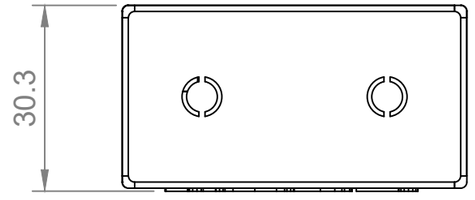
ESCALA	CARACT.	FORM.
1:1	Peso:	A4
	Material:	

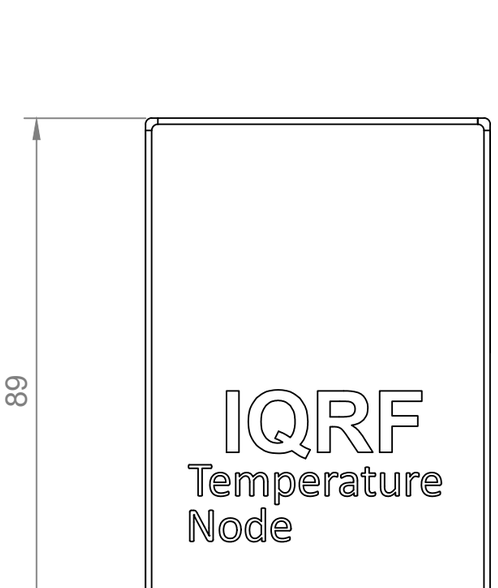
	Nombre y firma	Fecha
DIBUJ	M. Sánchez	08/06/19
VERIF		
APROB		
FABR.		
CALID		

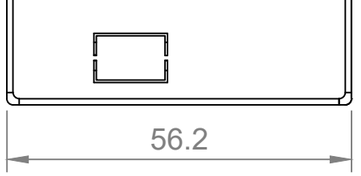
Las cotas se expresan en milímetros si no se indica lo contrario.
No cambiar la escala.

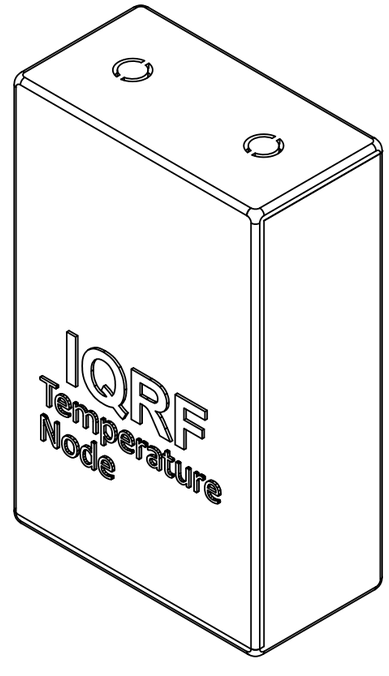
Nº Dibujo:

Nº Referencia:









IQRF Temp. sensor

Vista general del ensamblaje

	Nombre y firma	Fecha
DIBUJ	M. Sánchez	08/06/19
VERIF		
APROB		
FABR.		
CALID		

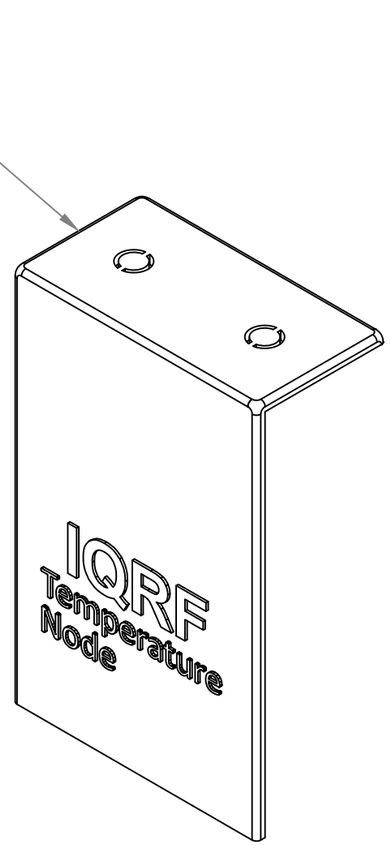
ESCALA	CARACT.	FORM.
1:1	Peso:	A3
	Material:	

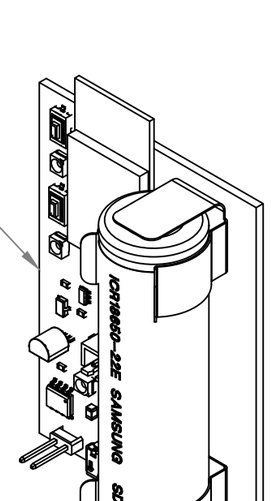
Las cotas se expresan en milímetros si no se indica lo contrario.
Los bordes redondeados poseen un radio de 1mm
No cambiar la escala.

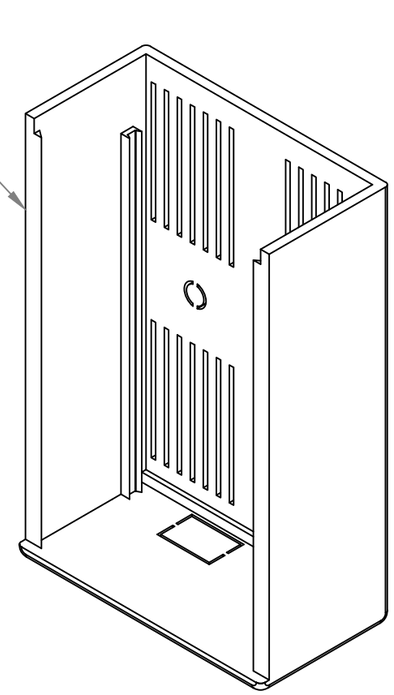
Nº Dibujo: A.Box IQRF_All

Nº Referencia:

CREADO CON UNA VERSIÓN PARA ESTUDIANTES DE AUTODESK







IQRF Temp. sensor

Plano de explosión

ESCALA	CARACT.	FORM.
1:1	Peso:	A3
	Material:	

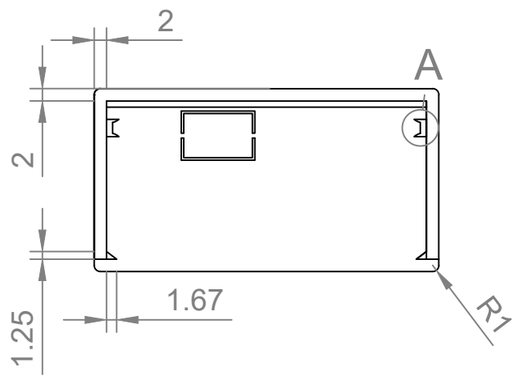
	Nombre y firma	Fecha
DIBUJ	M. Sánchez	08/06/19
VERIF		
APROB		
FABR.		
CALID		

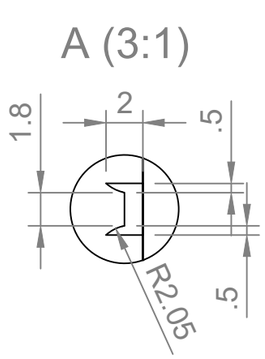
Las cotas se expresan en milímetros si no se indica lo contrario.
Los bordes redondeados poseen un radio de 1mm
No cambiar la escala.

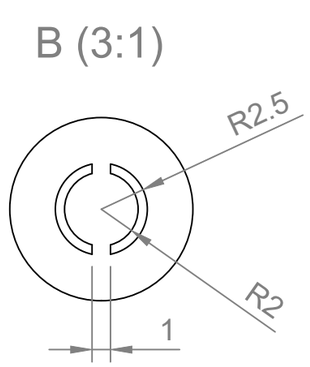
Nº Dibujo: A.Box IQRF_All

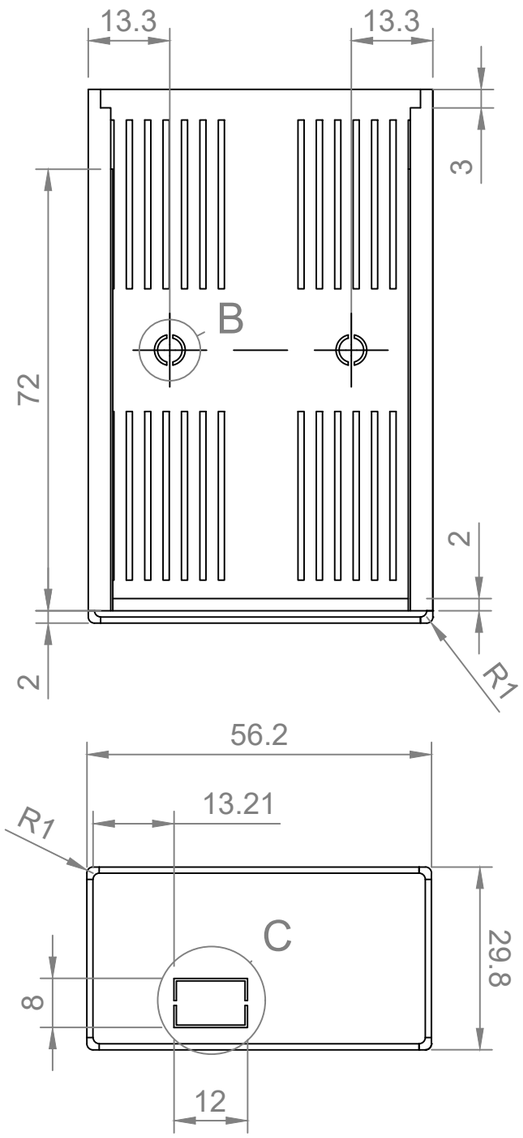
Nº Referencia:

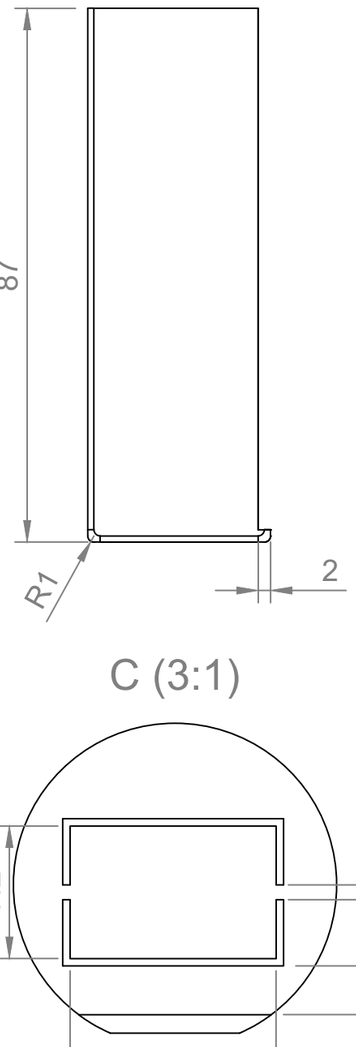
CREADO CON UNA VERSIÓN PARA ESTUDIANTES DE AUTODESK

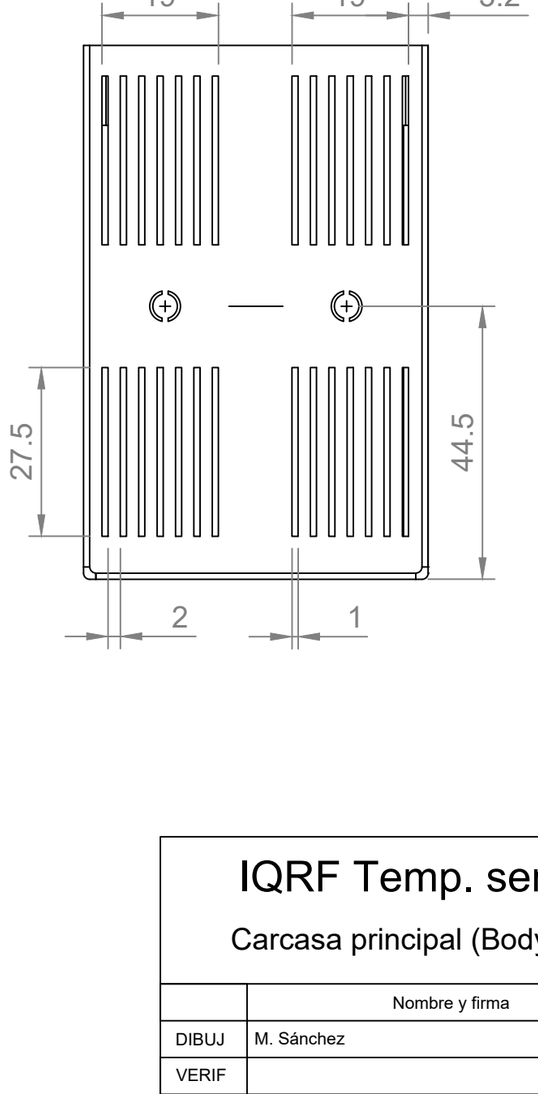


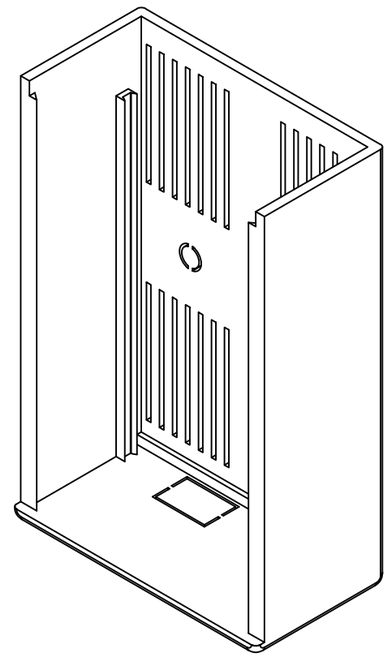












IQRF Temp. sensor Box

Carcasa principal (Body)

ESCALA	CARACT.	FORM.
1:1	Peso:	A3
	Material: ABS	

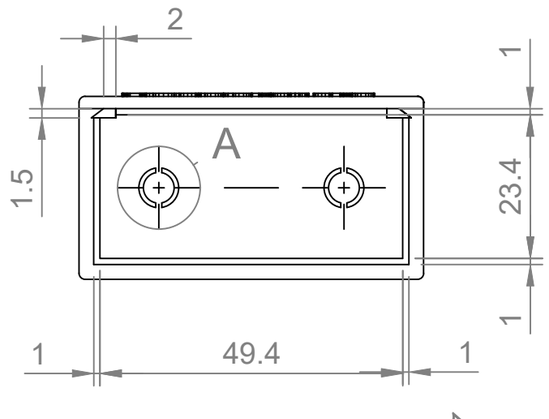
	Nombre y firma	Fecha
DIBUJ	M. Sánchez	08/06/19
VERIF		
APROB		
FABR.		
CALID		

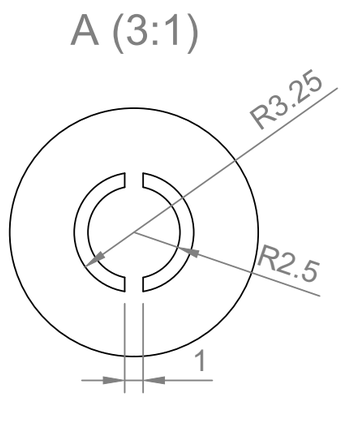
Las cotas se expresan en milímetros si no se indica lo contrario.
Los bordes redondeados poseen un radio de 1mm
No cambiar la escala.

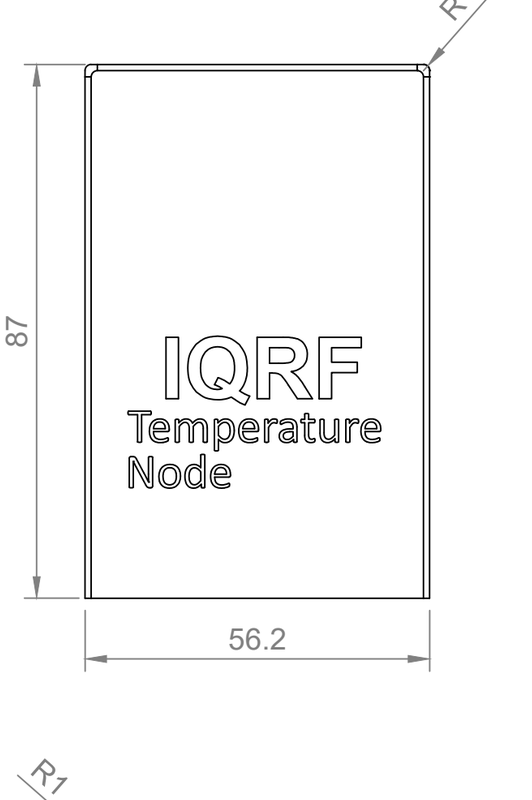
Nº Dibujo: C. Box IQRF_Body

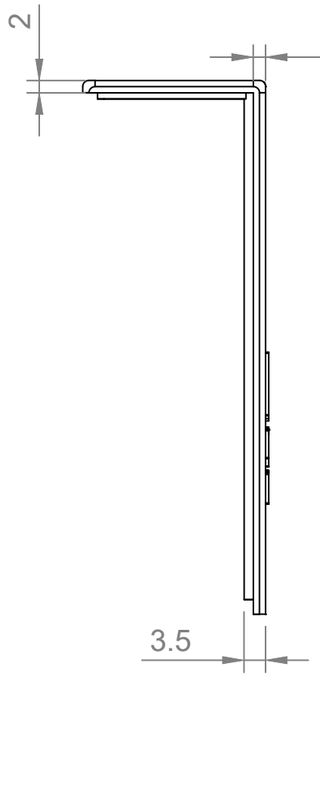
Nº Referencia: ITSM000

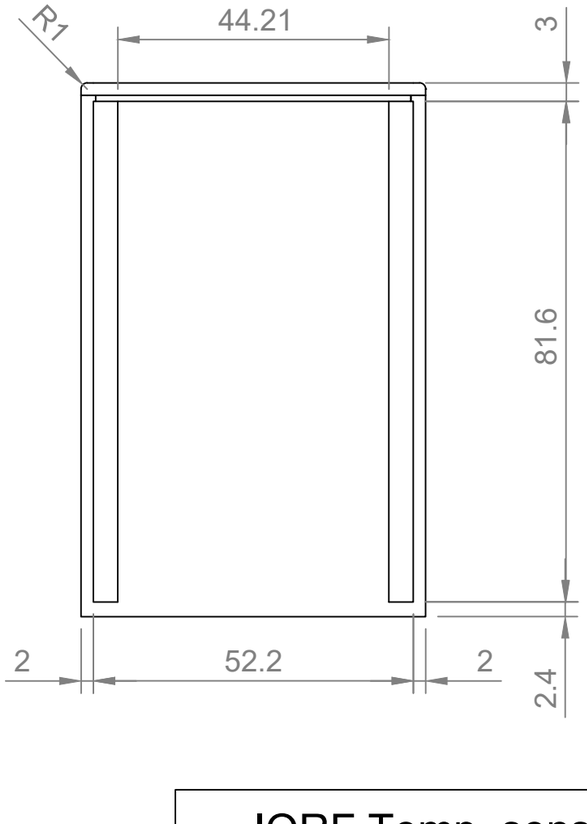
CREADO CON UNA VERSIÓN PARA ESTUDIANTES DE AUTODESK

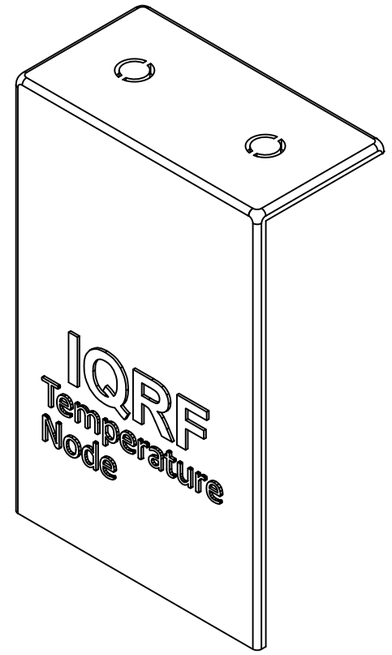


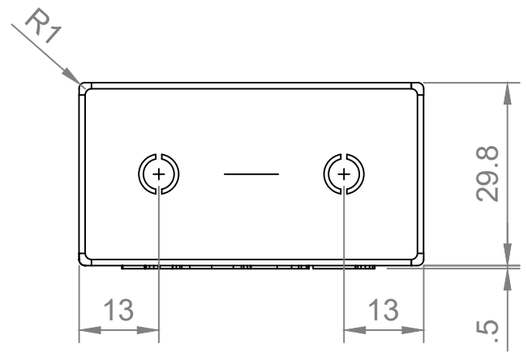












IQRF Temp. sensor Box

Tapa de carcasa (Front)

ESCALA	CARACT.	FORM.
1:1	Peso:	A3
	Material: ABS	

	Nombre y firma	Fecha
DIBUJ	M. Sánchez	08/06/19
VERIF		
APROB		
FABR.		
CALID		

Las cotas se expresan en milímetros si no se indica lo contrario.
Los bordes redondeados poseen un radio de 1mm
No cambiar la escala.

Nº Dibujo: D. Box IQRF_Front

Nº Referencia: ITSM001

CREADO CON UNA VERSIÓN PARA ESTUDIANTES DE AUTODESK

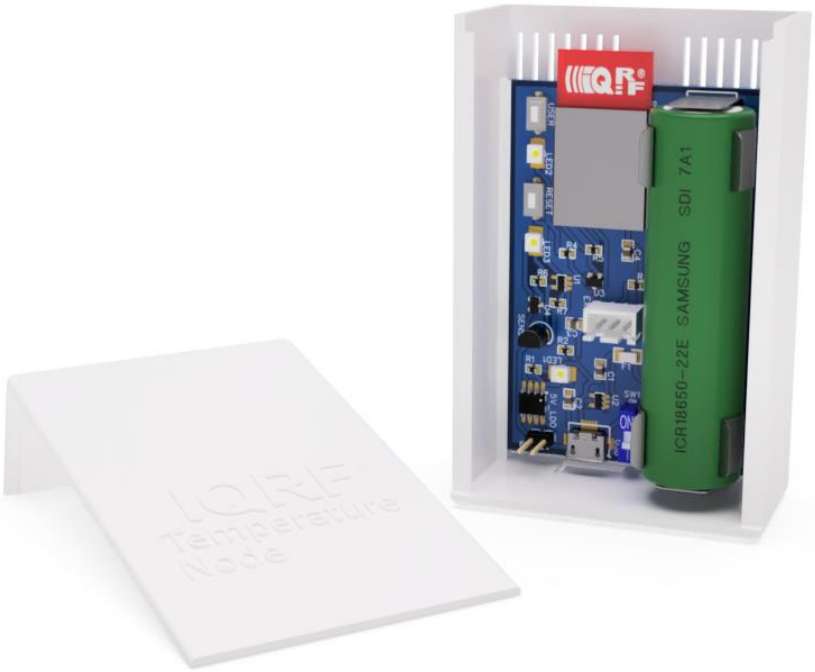
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

MONITORIZACIÓN DEL PATRIMONIO CULTURAL

MEDIANTE TECNOLOGÍA IOT IQRF

3. Pliego de condiciones



Autor: D. Miguel Andrés Sánchez Galdón

Tutor: D. Ángel Perles Ivars

Valencia, julio de 2019

ÍNDICE DEL PLIEGO

GLOSARIO DE SIGLAS Y ABREVIATURAS

CAD	Computer assisted design
CE	Conformidad Europea
Gerber	Formato de archivo para fabricación de PCB
IoT	Internet de las cosas
PCB	Tarjeta de circuito impreso
UE	Unión Europea
CE	Conformidad Europea
IPC	Asociación de estándares para la industria electrónica
RoHS	Restricción de sustancias peligrosas
JEDEC	Asoc. independiente de estandarización de circuitos integrados

1. DEFINICIÓN Y ALCANCE DEL PLIEGO

Este trabajo incluye el diseño y despliegue de un sistema basado en nodos inalámbricos que utilizan tecnologías IoT e IQRF con el fin de transmitir las medidas de temperatura recogidas en un lugar determinado.

En el presente pliego de condiciones se incluyen las directrices y condiciones con las que se debe fabricar, reproducir y utilizar el sistema desarrollado en el documento memoria en conjunto con los planos y la documentación técnica de cada elemento del proyecto.

Para utilizar la documentación de este proyecto de manera completa, será necesario disponer de los archivos CAD y Gerbers para la fabricación tanto de la placa como de la carcasa, así como el código de los módulos y diagramas de NodeRED.

2. CONDICIONES Y NORMAS DE CARÁCTER GENERAL

El diseño del nodo transmisor de este proyecto se ha realizado bajo unas condiciones de funcionamiento y su fabricación e instalación debe realizarse conforme a ellas.

2.1. Instalación

Al tratarse de un diseño en fase de pruebas, la instalación de los módulos deberá realizarse por un personal cualificado que conozca el producto. Esto es requerido ya que existen unas condiciones específicas que se deben cumplir con el fin de que el sistema funcione de manera correcta.

La ubicación de cada nodo deberá estudiarse en función de los siguientes condicionantes. De lo contrario, no se asegura el correcto funcionamiento de éstos.

Los módulos no deben situarse en ubicaciones cercanas a fuentes de calor y/o frío que puedan alterar las mediciones de temperatura, así como dañar los componentes del mismo. Por lo tanto, se deberá evitar situaciones en las que se encuentren cerca elementos como lámparas, bombillas, salidas de ventilación o aires acondicionados, así como equipos que desprendan gran cantidad de calor o aparatos de refrigeración.

En la ubicación de los módulos se debe evitar la incidencia directa de la luz, que podría alterar las medidas de temperatura y/o degradar los componentes.

No se debe emplear carcasas o cajas metálicas que puedan afectar a la difusión de las ondas de radio.

Antes de instalar una red de sensores como esta, se debe realizar un estudio de utilización del espectro de banda con las pruebas pertinentes que apruebe que no existan problemas en el espectro de radiofrecuencias que impidan o alteren la comunicación de los módulos. Al trabajar con bandas del espectro ISM, no requiere licenciar la instalación.

Antes de situar cada módulo, se deberá realizar un estudio de las características anteriormente mencionadas, así como de impacto visual en la obra de arte. En caso de duda se deberá consultar al autor del proyecto o a un experto conservador.

De la misma manera, el Gateway deberá situarse en una zona en la que exista conexión WiFi o en su defecto red LAN cableada. Tanto la instalación como la configuración descrita en la memoria deben ser realizadas por personal cualificado.

No se garantiza el correcto funcionamiento de este sistema empleando diferente software o hardware al descrito en la memoria del proyecto.

2.2. Seguridad

El nodo diseñado está catalogado como seguro, ya que su fuente de alimentación consiste en baterías 18650 no peligrosas de bajo voltaje (3,7V). Las baterías utilizadas en este proyecto cumplen con la normativa UL94 de inflamabilidad y la UN38.3 de transporte de baterías de litio. Cada nodo debe utilizar las baterías que se han establecido en su diseño y que se presentan en la documentación del proyecto, y no se garantiza ni el funcionamiento ni la seguridad con el uso de otras.

A este dispositivo se le aplica la directiva europea de marcado CE y en específico las siguientes normativas de obligado cumplimiento:

- 206/95/CE de baja tensión y seguridad general.

- 2004/108/CE y 2014/30/UE de compatibilidad electromagnética, donde se miden y ensayan las emisiones o perturbaciones electromagnéticas e inmunidad conducidas o radiadas según los materiales y el entorno de aplicación.
- 1995/CE de Radio y Telecomunicaciones.

Y en específico las normativas propias de aparatos radioeléctricos al tratarse de un sensor inalámbrico como la Directiva 2014/53/UE (NB nº. 0370) y la 2008/07/11/BOE sobre dispositivos de radiofrecuencia operando en la banda de 865-868MHz.

Todos los transceptores IQRF de la serie TR-7xxx poseen certificación CE y cumplen los estándares europeos recogidos en los apartados EN 300 y EN 301.

Todos los elementos electrónicos de este proyecto cumplen la directiva RoHS (Restriction of Hazardous Substances) 2011/65/EU.

2.3. Utilización

Una vez desplegado el sistema y configurados los nodos y el Gateway, el sistema se conecta automáticamente a la nube. Puede llegar a tardar 5 minutos en visualizarse los datos en la plataforma. Para la autentificación con el servicio en la nube, el dispositivo se suministra con las claves API preconfiguradas en este servicio. Si los parámetros de la aplicación varían, puede ser necesario una reconfiguración de alguna de las partes, incluyendo la posibilidad de tener que hacerlo en el lugar físico donde se encuentra el sistema.

El sistema utiliza una red IQRF, por lo que su uso no es propiedad de la empresa o autor del proyecto.

Al tratarse de un dispositivo en fase de pruebas, no se garantiza el correcto funcionamiento a largo plazo, pudiéndose producir bajas en el servicio, por lo que no se recomienda su uso exclusivo en aplicaciones de alta importancia.

Para más información sobre uso y funcionamiento, contactar al autor del proyecto.

2.4. Mantenimiento

El mantenimiento requerido por el sistema incluye el recambio de las baterías, la inspección visual periódica y la inspección del servicio en la nube.

Las baterías deberán ser sustituidas cada 3 años o en caso de desconexión de nodo. Las baterías podrán ser recargadas en una estación de carga homologada, según indique el fabricante. Al final de la vida útil de las baterías, deben ser desechadas según la normativa 2006/66/CE, correspondiente a pilas y acumuladores y la gestión ambiental de sus residuos.

No se asegurará un correcto funcionamiento del sistema sin una inspección visual periódica cada 2 años que no muestre ningún defecto aparente. Deberá ser realizada por personal cualificado que conozca el sistema.

Al ser un proyecto en fase de pruebas, se incluye la posibilidad de tener que realizar cambios en la programación de los nodos para corregir anomalías. Este procedimiento deberá realizarse in-situ, por lo que no se recomienda la instalación en lugares de difícil acceso antes de terminar el periodo de pruebas.

3. CONDICIONES PARTICULARES Y FABRICACIÓN

3.1. Diseño de PCB

El diseño de la placa de circuito impreso se ha llevado a cabo siguiendo la normativa IPC actual para el diseño de circuitos impresos:

- IPC-2221A de diseño general de tarjetas de circuito impreso.
- IPC-2222 específica para placas rígidas.
- IPC-7351B que especifica los requerimientos para componentes de montaje superficial y define el tamaño de huella en la placa de circuito impreso.

3.2. Fabricación de PCB

La fabricación de las placas de circuito impreso debe realizarse bajo las siguientes condiciones:

- El material dieléctrico utilizado para la fabricación de las placas debe ser FR4 de 1,6mm de espesor, tal y como se especifica en el resto de la documentación del proyecto. Las modificaciones de las condiciones de fabricación anteriormente mencionadas deben ser aprobadas por el autor del proyecto o, en su defecto, por un ingeniero autorizado.
- La placa fabricada debe seguir la normativa correspondiente a la especificación de placas rígidas IPC-6012B y IPC-600 o aquellas normas más actuales.
- Debe realizarse con al menos dos capas de cobre. En caso de añadirse más capas deberán ser aprobadas por el autor.
- La PCB debe tener un acabado superficial ENIG, según dicta la norma IPC-4552. El acabado podrá ser plateado o dorado.
- La máscara de soldadura debe ser azul y de tipo LPISM.
- La serigrafía debe ser de color blanco y únicamente capa superior. No se acepta serigrafía sobre los pads.
- La fabricación de dicha placa requiere un test eléctrico de continuidad con el fin de garantizar el correcto funcionamiento y aislamiento de la placa.
- Cualquier problema detectado en inspección deberá ser reportado al autor del proyecto y en ese caso deberá detenerse su producción hasta que se solucione el problema.
- La placa deberá llevar la marca UL en cobre o máscara de soldadura, nunca en serigrafía. No se permiten otras marcas añadidas. Tampoco se permite la eliminación de planos de cobre.
- La tarjeta debe cumplir la normativa RoHS correspondiente y la directiva europea 2011/65/EU o actual.
- Toda placa debe ser entregada con su Certificado de Conformidad.

El incumplimiento de las anteriores directrices no asegura el correcto funcionamiento de las placas. El cliente puede rechazar la placa si no se cumplen las directrices mencionadas.

3.3. Ensamblaje de PCB

El ensamblaje de la tarjeta deberá ser realizado de acuerdo a la normativa IPC-A-610E por un técnico certificado u una empresa ensambladora autorizada.

La soldadura de la placa deberá realizarse en horno de convección, empleando el perfil de temperatura más conservador con el fin de asegurar en correcto tratamiento de los componentes electrónicos, nunca superando los 250 ºC de temperatura. En caso de no poseer un proceso adecuado, se recomienda el de la Ilustración 1:

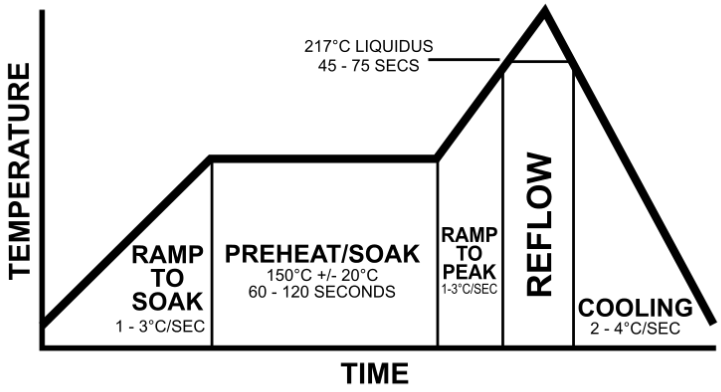


Ilustración 1 - Proceso de soldadura en horno de convección (Fuente: ReflowBerryPi)

Para la soldadura no se permite el uso de halógenos tales como mercurio, plomo o cadmio, según la normativa RoHS (2002/95/CE). La empresa ensambladora deberá acreditar que el proceso se realiza de acuerdo a dicha normativa. Así mismo, se exige el seguimiento de la especificación J-STD-0001F, la cual recoge los requerimientos para soldadura eléctrica y ensambles electrónicos.

3.4. Componentes electrónicos

Los componentes electrónicos empleados en este circuito deben ser los mismos indicados en el presente proyecto. En caso de uso de componentes diferentes a los indicados deberá ser aprobado por el autor del proyecto. En caso contrario, no existirá garantía de correcto funcionamiento del mismo.

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

Los componentes deberán cumplir la normativa europea sobre la Restricción de Sustancias Peligrosas en aparatos eléctricos y electrónicos (RoHS) correspondiente e incluir el certificado de conformidad a dicha ley que lo asegure.

Los componentes electrónicos no deberán ir expuestos a luz directa y deberán ir contenidos en bolsas herméticamente cerradas al vacío que garanticen la adecuada conservación del mismo, según se describe en la normativa IPC/JEDEC J-STD-020.

En caso de humedad durante el transporte (superior al 60%), según la normativa J-STD-033, los componentes deberán introducirse en un horno de secado durante 48 horas a 125 ºC, o lo que especifique el fabricante.

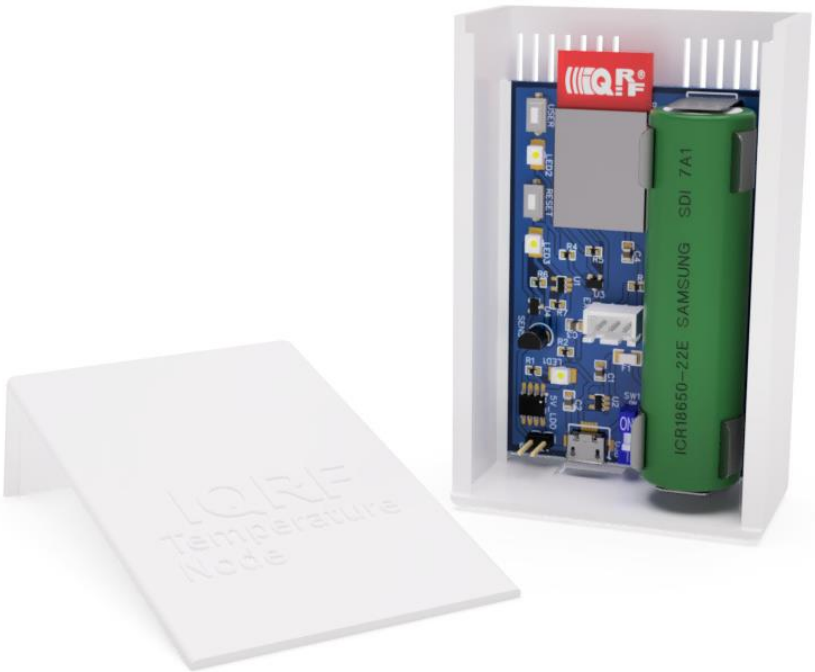
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

MONITORIZACIÓN DEL PATRIMONIO CULTURAL

MEDIANTE TECNOLOGÍA IOT IQRF

4. Presupuesto



Autor: D. Miguel Andrés Sánchez Galdón

Tutor: D. Ángel Perles Ivars

Valencia, julio de 2019

ÍNDICE DEL PRESUPUESTO

GLOSARIO DE SIGLAS Y ABREVIATURAS

EA	Each (por unidad)
IVA	Impuesto sobre el Valor Añadido
PCB	Printed Circuit Board
PVP	Precio de Venta al Público
UDS	Unidades
UOM	Unit of measure (unidad de medida)

1. Sobre el presupuesto

En este proyecto se han diferenciado dos partes referentes al presupuesto. Por un lado, se han tenido en cuenta todos los elementos relacionados con el diseño y producción del prototipo realizado para los nodos, así como del Gateway. En esta parte se recogen los gastos asociados a materiales empleados, componentes electrónicos y horas de trabajo empleadas, tanto en desarrollo como en montaje.

En la segunda parte se contemplan los gastos asociados al despliegue de la red IQRF y servicios web alojados en la nube, así como su configuración.

Todos los precios recogidos en el presente documento de componentes electrónicos y elementos diversos necesarios para la aplicación concreta datan del 28 de mayo de 2019. Los proveedores de dichos productos son distribuidores autorizados de equipos de electrónica como Mouser y LCSC. Dado que se encuentran en un mercado con muchas fluctuaciones en los que los precios (PVP) pueden cambiar relativamente rápido, este presupuesto se presenta con una garantía y validez de un año. Así mismo, se advierte que los componentes pueden quedar obsoletos a medida que pasa el tiempo, en cuyo caso será necesario un rediseño de los mismos.

2. Materiales

2.1 Componentes electrónicos

En la siguiente tabla se recogen todos los costes relacionados con los componentes electrónicos que se incluyen en cada uno de los diez dispositivos nodo del proyecto.

Referencia	Cantidad	Descripción	Uds.	Fabricante	Ref. Fabricante	Precio	Total
ITSEA000	10	Samsung Li-Ion Batería 18650 2.5Ah 20A Max	EA	Samsung	INR 18650-25R	5,45 €	54,50 €
ITSEB000	10	Dip switch Slide Type 1P	EA	TE Connectivity	NDS-01V	0,33 €	3,26 €
ITSEB001	20	PushButton 2P SMD	EA	XKB Enterprise	TS-1101-C-W	0,03 €	0,68 €
ITSEC000	30	Condensador de cerámica SMD 0805 4.7uF 10%	EA	Murata Electronics	GRM21BR61H475KE51L	0,22 €	6,72 €
ITSEC001	10	Condensador de cerámica SMD 0805 1uF 10%	EA	KEMET	C0805C105K4RAC7210	0,09 €	0,91 €
ITSED000	30	LED estándar - SMD Red 624nm	EA	VCC	VAOL-S12RP4	0,18 €	5,37 €
ITSEF000	10	Resetteable Fuse 16V Trip 1.5A	EA	TECHFUSE	nSMD075-16V	0,06 €	0,58 €
ITSEP000	10	KMMX-ABSMT5SG-30TR MicroUSB Connector	EA	Kycon	KMMX-ABSMT5SG-30TR	0,94 €	9,43 €
ITSEP001	20	Contacto Batería 18650 16-19mm	EA	Keystone Electronics	54	0,31 €	6,14 €
ITSEP002	10	XH Connector socket 3P 2.5mm	EA	JST Sales America	B3B-XH-A	0,06 €	0,57 €
ITSEP003	10	IQRF SIM connector for TR modules	EA	IQRF Tech	KON-SIM-02	1,90 €	19,00 €
ITSEP004	10	KK 254 Breakaway Header 2P	EA	Molex	22-28-4027	0,13 €	1,30 €
ITSEQ000	10	TLV70033QDDCRQ1 3.3V LDO	EA	Texas Instruments	TLV70033QDDCRQ1	0,45 €	4,54 €
ITSEQ001	10	MCP73831T-2ACI/OT Battery Charger	EA	Microchip Technology	MCP73831T-2ACI/OT	0,50 €	4,96 €
ITSEQ002	10	LM2931MX-5.0 5V LDO	EA	Texas Instruments	LM2931MX-5.0	0,77 €	7,70 €

Referencia	Cantidad	Descripción	Uds.	Fabricante	Ref. Fabricante	Precio	Total
ITSEQ003	20	P-Channel MOSFET SOT23-3	EA	Changjiang Electronics Tech	CJ2301	0,03 €	0,61 €
ITSER000	10	Resistor SMD 200Ohms ±0.1% 1/10W 0603	EA	KOA Speer Elec	RN731JTTD2000B25	0,11 €	1,05 €
ITSER001	20	Resistor SMD 330Ohms ±0.5% 1/10W 0603	EA	Yageo	RT0603DRE07330RL	0,07 €	1,30 €
ITSER002	10	Resistor SMD 470Ohms ±0.5% 1/10W 0603	EA	Yageo	RT0603DRE07470RL	0,07 €	0,65 €
ITSER003	10	Resistor SMD 1KOhms ±0.5% 1/10W 0603	EA	Yageo	RT0603DRE071KL	0,06 €	0,62 €
ITSER004	10	Resistor SMD 2KOhms ±0.5% 1/10W 0603	EA	Yageo	RT0603DRE072KL	0,06 €	0,62 €
ITSER005	10	Resistor SMD 4.7KOhms ±0.1% 1/10W 0603	EA	Viking Tech	ARG03BTC4701	0,03 €	0,29 €
ITSEU000	10	DS18B20+ Temperature Sensors TO-92	EA	Maxim Integrated	DS18B20+	0,69 €	6,91 €
ITSEU001	10	IQRF transceiver TR-72DA	EA	IQRF Tech	TR-72DC	11,95 €	119,50 €
TOTAL							**257,22 €**

Tabla 1 - Presupuesto componentes electrónicos

El coste total de los componentes electrónicos asciende a **257,22 €.**

2.2 Fabricación de circuito impreso

El circuito impreso es una de las partes no estandarizadas del proyecto, por lo que debe construirla un proveedor de que cumpla con la normativa correspondiente a la fabricación de circuitos IPC-600. El fabricante elegido es JLCPCB (https://jlcpcb.com), de origen chino.

El precio de las placas se ha determinado solicitando un presupuesto personalizado a la empresa, con las características siguientes.

Característica	Opción
Dimensiones	70x50mm
Cantidad	10
Tipo de dieléctrico	FR-4
Número de capas	2
Ancho final de la placa	1,6mm
Ancho de la capa de cobre	1oz (35µm)
Agujero de pasante mínimo	0,3mm
Máscara de soldadura	Sí, Azul
Seriegrafía	Sí, Blanca
Test eléctrico	Sí

Tabla 2 - Características PCB

El coste total de las diez PCB asciende a 9,25€ + 6,30€ de envío, lo que suma un total de **15,55€.** El coste por unidad asciende a 1,56€.

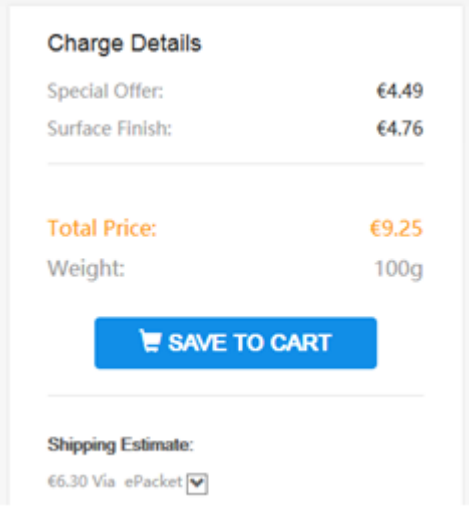


Ilustración 1 – Presupuesto JLCPCB

2.3 Carcasa impresa en 3D, material ABS

Las carcasas se han producido en impresoras 3D. Así se consigue reducir el tiempo de espera al no depender de terceros.

Se ha utilizado una impresora Anet A3S de deposición de material fundido tipo Prusa Mendel. Para determinar los costes se han tenido en cuenta el valor del material ABS, determinando la cantidad necesaria para la fabricación de las carcasas, además del coste energético de la propia máquina.

Unidad	Descripción	Parcial	Cantidad	Precio	Total
g	ABS filamento Dia. 1,75mm Den. 1,25g/cm3 1kg (24€/kg)		440	0,02 €	10,56 €
	ITSM000 - Carcasa Body ABS (10 uds.)	260			
	ITSM001 - Carcasa Tapa Frontal ABS (10 uds.)	180			
h	Coste impresión 3D (horas de impresora)		100,67	0,25 €	25,17 €
	ITSM000 - Carcasa Body ABS (10 uds.)	62,67			
	ITSM001 - Carcasa Tapa Frontal ABS (10 uds.)	38,00			
EA	Coste de preparación y calibración (por pieza)		1	3,00 €	3,00 €
TOTAL					**38,73 €**

Tabla 3 - Coste Impresión 3D

El gasto total de las carcasas es de 38,73€, por lo que cada unidad posee un coste de **3,87€.**

3. Punto de acceso y servicio web

Uno de los aspectos destacables de este proyecto es la conexión con los servicios en la nube. Para ello se han requerido elementos con comunicación a través de internet. El servicio web de IBM posee, a día 28 de mayo de 2019, una tarifa en la que se paga por consumo. La estimación se ha realizado para un año. Para más información, referirse al capítulo correspondiente a la memoria.

Cantidad	Descripción	Uds	Fabricante	Ref. Fabricante	Precio	Total
1	Raspberry Pi 3 Model B Rev. v1.2	EA	Raspberry Pi F.	896-8660	29,47 €	29,47 €
12	Servicio IBM Cloud, 500MB Lite	mes	IBM	-	12,30 €	147,64 €
1	CK-USB-04A	EA	IQRF Tech	CK-USB-04A	55,00 €	55,00 €
1	Sandisk Industrial MicroSD 16GB	EA	SanDisk	SDSDQAF3-016G-XI	12,31 €	12,31 €
1	IQRF TR-72DA	EA	IQRF Tech	IQRF TR-72DA	11,95 €	11,95 €
TOTAL						**256,36 €**

Tabla 4 - Coste punto de acceso y servidor

El coste total asciende a **256,36 €.**

4. Mano de obra

La mano de obra es el elemento de mayor coste en este proyecto, ya que los materiales utilizados son un bajo coste comparados con las horas de desarrollo de un nuevo producto electrónico.

Los costes de mano de obra incluyen el diseño de software y hardware, programación y fabricación. Se han calculado estimando la facturación relativa a un alumno recién graduado en la rama industrial de una ingeniería. Esta estimación se ha realizado teniendo en cuenta los créditos (12 ECTS) correspondientes al TFG que implican un total aproximado de 300 horas, teniendo en cuenta un salario de 40€/h. Sin embargo, la duración total del proyecto, dado que se requieren más horas de las previstas, se ha realizado la siguiente tabla con las cifras reales de trabajo. Para las tareas de producción y comprobación, al requerirse una cualificación inferior para su realización, se ha establecido un precio más bajo.

Unidad	Descripción	Parcial	Cantidad	Precio	Total
h	Diseño eléctrico		57	40 €	2.280 €
	1 - Diseño del esquema electrónico en EasyEDA	40			
	2 - Layout del circuito y diseño PCB	15			
	3 - Preparación documentación de fabricación	2			
h	Ensamblaje del prototipo		210	25 €	5.250 €
	1 - Montaje de la PCB. Soldadura manual (10 uds.)	12			
	2 - Inspección visual (10 uds.)	2			
	3 - Limpieza de restos de flux (10 uds.)	2			
	4 - Test eléctrico de continuidad y polaridad (10 uds.)	5			
h	Desarrollo de la programación y depuración		83	40 €	3.320 €
	1 - Desarrollo código fuente de test	13			
	2 - Desarrollo código fuente aplicación final	55			
	3 - Depuración aplicación final y programación prototipo	15			
h	Diseño mecánico de la carcasa		9	40 €	360 €
	1 - Diseño mecánico del cuerpo de la carcasa	4			
	2 - Diseño mecánico de la tapa	3			
	3 - Ensamblajes y simulaciones CAD	2			
h	Redacción y generación de documentación		60	40 €	2.400 €
	1 - Redacción memoria de proyecto	50			
	2 - Generación de planos	6			
	3 - Generación de presupuestos	4			
TOTAL					13.610 €

Tabla 5 - Coste mano de obra

El precio final correspondiente a la mano de obra es de **13.610,00€**.

5. Amortización de equipos

Para el diseño y desarrollo de este tipo de proyectos se requiere determinado equipo que en ocasiones puede ser de alto coste. Por ello, se requiere realizar un cálculo que incluya en la factura la amortización de dichos equipos en función de su coste y la reducción en su vida útil. El valor en horas se consigue multiplicando las horas diarias de uso por 365 días de un año y por los años de vida estimada que proporciona el fabricante.

$$\text{Vida útil (h)} = Utilización\ diaria(h) \times 365\ días\ \times Vida\ estimada(años)$$

Dividiendo el valor anterior entre el precio de cada producto, se obtiene el valor amortizado por hora de uso que, multiplicado por las horas de uso en el proyecto, proporciona la amortización del equipo.

Las horas de la tabla se han recogido en función de los estipulado en el apartado anterior 3. Mano de obra.

Unidad	Descripción	Precio Equipo	Vida útil	Cantidad	Precio	Total
h	Ordenador personal Asus X550	650,00 €	11680	419	0,06 €	23,32 €
h	Multímetro Genérico	50,00 €	5110	5	0,01 €	0,05 €
h	Impresora 3D Anet A3S	250,00 €	2920	101	0,09 €	8,62 €
h	Soldador Punta fina Genérico	50,00 €	3650	12	0,01 €	0,16 €
TOTAL						**32,15 €**

Tabla 6 – Coste amortización

El precio total de la utilización de los equipos es de **32,15€.**

6. Resumen

En este apartado se reúnen todos los gastos calculados en los apartados anteriores a modo de resumen.

Sección	Descripción	Parcial	Total
1	Materiales		567,86 €
1.1	Componentes electrónicos	257,22 €	
1.2	Fabricación circuito impreso	15,55 €	
1.3	Impresión carcasa 3D ABS	38,73 €	
1.4	Punto de acceso y servicio web	256,36 €	
2	Mano de obra		13.610,00 €
3	Equipos		32,31 €
TOTAL SIN IVA			**14.210,01 €**
Impuestos	IVA	21%	2.984,10 €
Añadidos	Gastos adicionales	10%	1.421,00 €
GASTO TOTAL DEL PROYECTO			**18.615,11 €**

Tabla 7 - Coste total

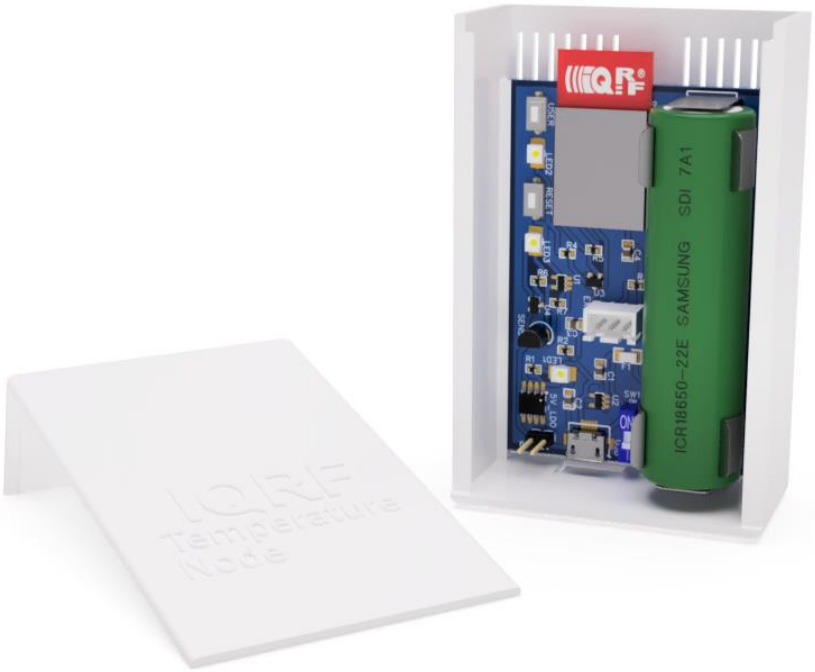
La elaboración del proyecto conlleva un gasto total de DIECIOCHO MIL SEISCIENTOS QUINCE EUROS CON ONCE CÉNTIMOS (**18.615,11 €**).

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

MONITORIZACIÓN DEL PATRIMONIO CULTURAL MEDIANTE TECNOLOGÍA IOT IQRF

5. Anexo



Autor: D. Miguel Andrés Sánchez Galdón

Tutor: D. Ángel Perles Ivars

Valencia, julio de 2019

ÍNDICE DEL ANEXO 1 - TABLAS DE PRUEBAS ITACA

1. TABLAS DE PRUEBAS DE FUNCIONAMIENTO

1.1. Prueba 1

Prueba 1								
Apartado	Enlace	Nodos	Distancia (m)	Nivel de la red	Obstáculos	Conexión	Repetitividad	Planta
1	1	0-4	3,42	0	SI	SI	SI	0
	2	0-7	4,18	0	NO	SI	SI	0

1.2. Prueba 2

Prueba 2								
Apartado	Enlace	Nodos	Distancia (m)	Nivel de la red	Obstáculos	Conexión	Repetitividad	Planta
1	1.1	0-2	0,50	0	NO	SI	SI	0
	1.2	0-5	0,94	0	NO	SI	SI	0
	1.3	0-6	23,05	0	SI	SI	SI	0
	1.4	5-4	30,01	1	SI	SI	SI	0
	1.5	6-7	15,83	1	SI	SI	SI	0
2	2.1	0-2	0,50	0	NO	SI	SI	0
	2.2	0-5	0,94	0	NO	SI	SI	0
	2.3	5-4	30,01	1	SI	SI	SI	0
	2.4	4-7	17,74	2	NO	SI	SI	0
3	3.1	0-2	0,50	0	NO	SI	SI	0
	3.2	0-5	0,94	0	NO	SI	SI	0
	3.3	5-7	35,34	1	SI	NO	NO	0

1.3. Prueba 3

Prueba 3								
Apartado	Enlace	Nodos	Distancia (m)	Nivel de la red	Obstáculos	Conexión	Repetitividad	Planta
1	1.1	0-2	0,50	0	NO	SI	SI	0
	1.2	0-4	30,69	0	SI	SI	SI	0
	1.3	0-5	12,28	0	SI	SI	SI	0
	1.4	0-6	11,65	0	SI	SI	SI	0
	1.5	4-7	17,74	1	NO	SI	SI	0
2	2.1	0-2	0,50	0	NO	SI	SI	0
	2.2	0-4	30,69	0	SI	SI	SI	0
	2.3	0-5	12,28	0	SI	SI	SI	0
	2.4	5-6	8,36	2	SI	NO	NO	1

1.4. Prueba 4

Prueba 4								
Apartado	Enlace	Nodos	Distancia (m)	Nivel de la red	Obstáculos	Conexión	Repetitividad	Planta
1	1.1	0-2	14,17	0	SI	SI	SI	0
2	2.1	0-2	14,17	0	SI	SI	SI	0
	2.2	2-4	6,21	1	SI	SI	SI	0
3	3.1	0-2	14,17	0	SI	SI	SI	0
	3.2	2-4	6,21	1	SI	SI	SI	0
	3.3	4-5	5,60	2	SI	SI	SI	1
4	4.1	0-2	14,17	0	SI	SI	SI	0
	4.2	2-4	6,21	1	SI	SI	SI	0
	4.3	4-5	5,60	2	SI	SI	SI	1
	4.4	5-6	5,60	3	SI	SI	SI	2
5	5.1	0-2	14,17	0	SI	SI	SI	0
	5.2	2-4	6,21	1	SI	SI	SI	0
	5.3	4-5	5,60	2	SI	SI	SI	1
	5.4	5-6	5,60	3	SI	SI	SI	2
	5.5	5-7	11,20	3	SI	SI	SI	3
6	6.1	0-2	14,17	0	SI	SI	SI	0
	6.2	2-4	6,21	1	SI	SI	SI	0
	6.3	4-5	5,60	2	SI	SI	SI	1
	6.4	5-6	5,60	3	SI	SI	SI	2
	6.5	6-7	11,20	4	SI	SI	SI	4

1.5. Prueba 5

Prueba 5								
Apartado	Enlace	Nodos	Distancia (m)	Nivel de la red	Obstáculos	Conexión	Repetitividad	Planta
1	1.1	0-2	14,17	0	SI	SI	SI	0
	1.2	2-4	6,21	1	SI	SI	SI	0
	1.3	4-6	11,20	2	SI	NO	NO	2
	1.4	6-7	11,20	3	SI	NO	NO	4
2	2.1	0-2	14,17	0	SI	SI	SI	0
	2.2	2-5	8,36	1	SI	NO	NO	1
	2.3	5-6	5,60	2	SI	NO	NO	2
	2.4	6-7	11,20	3	SI	NO	NO	4
3	3.1	0-4	15,48	0	SI	NO	NO	0
	3.2	4-5	5,60	1	SI	NO	NO	1
	3.3	5-6	5,60	2	SI	NO	NO	2
	3.4	6-7	11,20	3	SI	NO	NO	4

1.6. Prueba 6

Prueba 6								
Apartado	Enlace	Nodos	Distancia (m)	Nivel de la red	Obstáculos	Conexión	Repetitividad	Planta
1	1.1	0-2	14,17	0	SI	SI	SI	0
	1.2	2-4	6,21	1	SI	SI	SI	0
	1.3	4-5	5,60	2	SI	SI	SI	1
	1.4	5-6	5,60	3	SI	SI	SI	2
	1.5	6-7	9,70	4	SI	SI	SI	3
2	2.1	0-2	14,17	0	SI	SI	SI	0
	2.2	2-4	6,21	1	SI	SI	SI	0
	2.3	4-5	5,60	2	SI	SI	SI	1
	2.4	5-6	5,60	3	SI	SI	SI	2
	2.5	6-7	17,52	4	SI	SI	SI	3

ÍNDICE DEL ANEXO 2 - LISTADO CÓDIGOS

ANEXO 2.1

LISTADO CÓDIGO DEL NODO (FORMATO C PARA IQRF IDE)

```
//
******************************************************************
//   Custom DPA Handler code example - Standard Sensors - DDC-SE01
*
//
******************************************************************
// Copyright (c) IQRF Tech s.r.o.
//
// File:    $RCSfile: 0002_DDC-SE01.c,v $
// Version: $Revision: 1.8 $
// Date:    $Date: 2017/11/16 18:11:59 $
//
// Revision history:
//   2017/11/16  Release for DPA 3.02
//   2017/08/14  Release for DPA 3.01
//
//
******************************************************************

// Online DPA documentation http://www.iqrf.org/DpaTechGuide/

// This example implements 4 sensors according to the IQRF Sensors
standard
// 1st sensor is on-board TR temperature sensor.
// 2nd sensor is either Dallas 18B20 or MCP9802 temperature sensor at
DDC-SE01 board (according to the HW jumper position) chosen at the
runtime based on the SW detection.
// 3rd sensor is light intensity indicator at DDC-SE01 board (value
range is 0[max light]-127[max dark]).
// 4th sensor is potentiometer value at DDC-SE01 board (value range is
0[left stop]-127[right stop]).

// The example will not work at demo DPA version because demo DPA does
not support PCMD != 0

// Default IQRF include (modify the path according to your setup)
#include "../../Development/include/IQRF_OS/IQRF.h"

// Default DPA header (modify the path according to your setup)
#include "../../Development/include/DPA/DPA.h"
// Default Custom DPA Handler header (modify the path according to
your setup)
#include "../../Development/include/DPA/DPAcustomHandler.h"
// IQRF standards header (modify the path according to your setup)
#include "../../Development/include/DPA/IQRFstandard.h"
#include "../../Development/include/DPA/IQRF_HWPID.h"

//#################################################################
########################
```

```
// Number of implemented sensors
#define SENSORS_COUNT 4

// Variable to store sensor value at Get?_????() methods. This example
implements sensors returning maximum 2 bytes of data.
uns16 sensorValue @ param3;

// Reads sensor value to the sensorValue variable and to
responseFRCvalue(2B) variable
bit Get0_TemperatureTR();
bit Get1_Temperature();
bit Get2_BinaryData_Light();
bit Get3_BinaryData_Potentiometer();

// Temperature sensors read routine for both DDC-SE01 sensor types
void GetTemperature();
// Read preset PIC ADC for DDC-SE01
uns8 ReadAdc();

// Stores sensor value byte(s) to the FSR1[+1...], in case of
PCMD_STD_SENSORS_READ_TYPES_AND_VALUES sensor type is stored before
value byte(s)
void StoreValue( uns8 sensorType );

// Sensor connected to PORT C.3 (compatible with DDC-SE-01)
#define OneWire_TRIS         TRISC.3
#define OneWire_IO_IN        PORTC.3
#define OneWire_IO_OUT       LATC.3

// Writes sensor configuration (resolution)
bit Ds18B20WriteConfig( uns8 value );

// Resets OneWire
bit OneWireReset();
// Reads OneWire byte
uns8 OneWireReadByte();
// Writes OneWire byte
void OneWireWriteByte( uns8 byte );

// DS18B20 commands
#define CMD_READROM       0x33
#define CMD_CONVERTTEMP   0x44
#define CMD_CPYSCRATCHPAD 0x48
#define CMD_WSCRATCHPAD   0x4e
#define CMD_MATCHROM      0x55
#define CMD_RPWRSUPPLY    0xb4
#define CMD_RECEEPROM     0xb8
#define CMD_RSCRATCHPAD   0xbe
#define CMD_SKIPROM       0xcc
#define CMD_ALARMSEARCH   0xec
#define CMD_SEARCHROM     0xf0

// I2C routines
```

```
void i2c_init();
void i2c_shutdown();
void i2c_waitForIdle();
void i2c_start();
void i2c_repStart();
void i2c_stop();
uns8 i2c_read( bit ack );
void i2c_write( uns8 i2cWriteData );

// MCP9802 address
#define I2C_ADR             0b10010110
// Power pin
#define PWR_SENSOR_TRIS     TRISC.7
#define PWR_SENSOR_IO       LATC.7

// Special temperature value to indicate a sensor error, compatible
with IQRF sensor standard
#define ERROR_TEMPERATURE 0x8000

// TRUE if DS18B20 is enabled at runtime at startup, FALSE in case of
MCP9802
bit isDS18B20;
// Final DS18B20 temperature value read by state machine
uns16 temperature;

// Must be the 1st defined function in the source code in order to be
placed at the correct FLASH location!
//#####################################################################
########################
bit CustomDpaHandler()
//#####################################################################
########################
{
  // This forces CC5X to wisely use MOVLB instructions (doc says:  The
'default' bank is used by the compiler for loops and labels when the
algorithm gives up finding the optimal choice)
#pragma updateBank default = UserBank_01

  // Finite machine states
  typedef enum
  {
    S_ResetConvertT = 0,
    S_SkipRomConvertT,
    S_CmdConvertT,

    S_WaitConvertT,

    S_ResetReadTemp,
    S_SkipRomReadTemp,
    S_CmdReadTemp,
    S_Byte1ReadTemp,
    S_Byte2ReadTemp
  } TState;
```

```
  // Handler presence mark
  clrwdt();

  // Sleeping parameters, valid when Time != 0
  static TPerOSSleep_Request PerOSSleep_Request;
  // Finite machine state
  static uns8 state; // = S_ResetConvertT = 0
  // Pre-read lower temperature byte
  static uns8 temperatureByteLow;
  // Conversion timeout counter
  static uns8 timeout;

  // Detect DPA event to handle
  switch ( GetDpaEvent() )
  {
    // -------------------------------------------------
    case DpaEvent_Interrupt:
      // Do an extra quick background interrupt work
      // ! The time spent handling this event is critical.If there is
no interrupt to handle return immediately otherwise keep the code as
fast as possible.
      // ! Make sure the event is the 1st case in the main switch
statement at the handler routine.This ensures that the event is
handled as the 1st one.
      // ! It is desirable that this event is handled with immediate
return even if it is not used by the custom handler because the
Interrupt event is raised on every MCU interrupt and the empty return
handler ensures the shortest possible interrupt routine response time.
      // ! Only global variables or local ones marked by static
keyword can be used to allow reentrancy.
      // ! Make sure race condition does not occur when accessing
those variables at other places.
      // ! Make sure( inspect.lst file generated by C compiler )
compiler does not create any hidden temporary local variable( occurs
when using division, multiplication or bit shifts ) at the event
handler code.The name of such variable is usually Cnumbercnt.
      // ! Do not call any OS functions except setINDFx().
      // ! Do not use any OS variables especially for writing access.
      // ! All above rules apply also to any other function being
called from the event handler code, although calling any function from
Interrupt event is not recommended because of additional MCU stack
usage.

      // ms per TMR6 interrupt
#define TICKS_LEN  10

      //  If TMR6 interrupt occurred
      if ( TMR6IF )
      {
        // Unmask interrupt
        TMR6IF = 0;
        // Decrement count
        if ( timeout != 0 )
          timeout--;
```

```
      }
      return Carry;

      // -------------------------------------------------
    case DpaEvent_Idle:
      // Do a quick background work when RF packet is not received

      // Should go to sleep?
      if ( PerOSSleep_Request.Time != 0 )
      {
        // Copy sleep parameters to the DPA request
        _DpaMessage.PerOSSleep_Request.Time = PerOSSleep_Request.Time;
        _DpaMessage.PerOSSleep_Request.Control =
PerOSSleep_Request.Control;
        // Switch off sleeping time=flag
        PerOSSleep_Request.Time = 0;
        // Finalize OS Sleep DPA Request
        _DpaDataLength = sizeof( _DpaMessage.PerOSSleep_Request );
        _PNUM = PNUM_OS;
        _PCMD = CMD_OS_SLEEP;
        // Perform local DPA Request to go to sleep
        DpaApiLocalRequest();
      }

      // Run finite state machine to read temperature from DS18B20 at
background so the temperature value is immediately ready for FRC
      if ( !isDS18B20 )
        break;

      // Make sure 1Wire data pin at LATX.y is low as it might be set
by another PORTX.? pin manipulation
      OneWire_IO_OUT = 0;

      skip( state );
#pragma computedGoto 1
      goto _S_ResetConvertT;
      goto _S_SkipRomConvertT;
      goto _S_CmdConvertT;
      goto _S_WaitConvertT;
      goto _S_ResetReadTemp;
      goto _S_SkipRomReadTemp;
      goto _S_CmdReadTemp;
      goto _S_Byte1ReadTemp;
      goto _S_Byte2ReadTemp;
#pragma computedGoto 0
      ;
      // --------------
_S_Byte2ReadTemp:
      temperature.high8 = OneWireReadByte();
      temperature.low8 = temperatureByteLow;

ResetMachine:
      state = S_ResetConvertT;
      goto ExitMachine;
```

```
      // --------------
_S_ResetConvertT:
_S_ResetReadTemp:
      if ( !OneWireReset() )
      {
_S_Error_Reset:
        temperature = ERROR_TEMPERATURE;
        goto ResetMachine;
      }
      goto NextState;

      // --------------
_S_SkipRomConvertT:
_S_SkipRomReadTemp:
      // OneWire: Skip ROM
      OneWireWriteByte( CMD_SKIPROM );
      goto NextState;

      // --------------
_S_CmdConvertT:
      // OneWire: Convert temperature
      OneWireWriteByte( CMD_CONVERTTEMP );
      // Setup timeout for approx 750 ms (the longest conversion time)
      timeout = 2 + 750 / TICKS_LEN;
      goto NextState;

      // --------------
_S_WaitConvertT:
      if ( OneWireReadByte() == 0xff )
        goto NextState;

      // Timeout?
      if ( timeout == 0 )
        goto _S_Error_Reset;

      goto ExitMachine;

      // --------------
_S_CmdReadTemp:
      // OneWire: Read scratchpad
      OneWireWriteByte( CMD_RSCRATCHPAD );
      goto NextState;

      // --------------
_S_Byte1ReadTemp:
      temperatureByteLow = OneWireReadByte();
      goto NextState;

      // --------------
NextState:
      ++state;

ExitMachine:
```

```
      break;

      // -----------------------------------------------
    case DpaEvent_Init:
      // Do a one time initialization work before main loop starts

      // Initialize sensors
      // C5 (AN4) as input
      moduleInfo();
      // Connected TR pins?
      if ( !bufferINFO[5].7 )
      {
        TRISC.6 = 1;
        TRISB.4 = 1;
      }
      TRISA.5 = 1;

      // C1 (AN0) as input
      TRISA.0 = 1;

      // Setup TMR6 to generate ticks on the background (ticks every
10ms)
#if F_OSC == 16000000
      PR6 = 250 - 1;
      T6CON = 0b0.1001.1.10;    // Prescaler 16, Postscaler 10, 16 *
10 * 250 = 40000 = 4MHz * 10ms
#else
#error Unsupported oscillator frequency
#endif

      // Setup DS18B20 for 9bit precision, conversion takes 94ms (see
datasheet)
      if ( Ds18B20WriteConfig( 0b0.00.00000 ) )
        // DS18B20 is enabled
        isDS18B20 = TRUE;
      else
        // Expect MCP9802 is enabled
        i2c_init();

      break;

      // -----------------------------------------------
    case DpaEvent_AfterSleep:
      // Called after woken up after sleep
      if ( !isDS18B20 )
        i2c_init();

      // Called on wake-up from sleep
      TMR6IE = TRUE;
      TMR6ON = TRUE;
      break;

      // -----------------------------------------------
    case DpaEvent_BeforeSleep:
```

```
      // Called before going to sleep
      if ( !isDS18B20 )
        i2c_shutdown();

      // -------------------------------------------------
    case DpaEvent_DisableInterrupts:
      // Called when device needs all hardware interrupts to be
disabled (before Reset, Restart and RFPGM)
      // Must not use TMR6 any more
      TMR6ON = FALSE;
      TMR6IE = FALSE;
      break;

      // -------------------------------------------------
    case DpaEvent_DpaRequest:
      // Called to interpret DPA request for peripherals
      // -------------------------------------------------
      // Peripheral enumeration
      if ( IsDpaEnumPeripheralsRequest() )
      {
        // We implement 1 standard user peripheral
        _DpaMessage.EnumPeripheralsAnswer.UserPerNr = 1;
        FlagUserPer( _DpaMessage.EnumPeripheralsAnswer.UserPer,
PNUM_STD_SENSORS );
        _DpaMessage.EnumPeripheralsAnswer.HWPID =
HWPID_MICRORISC__DEMO_DDC_SE01;
        _DpaMessage.EnumPeripheralsAnswer.HWPIDver = 0x0000;

DpaHandleReturnTRUE:
        return TRUE;
      }
      // -------------------------------------------------
      // Get information about peripheral
      else if ( IsDpaPeripheralInfoRequest() )
      {
        if ( _PNUM == PNUM_STD_SENSORS )
        {
          _DpaMessage.PeripheralInfoAnswer.PerT =
PERIPHERAL_TYPE_STD_SENSORS;
          _DpaMessage.PeripheralInfoAnswer.PerTE =
PERIPHERAL_TYPE_EXTENDED_READ_WRITE;
          // Set standard version
          _DpaMessage.PeripheralInfoAnswer.Par1 = 12;
          goto DpaHandleReturnTRUE;
        }

        break;
      }
      // -------------------------------------------------
      else
      {
        // Handle peripheral command

        // Supported peripheral number?
```

```
        if ( _PNUM == PNUM_STD_SENSORS )
        {
          // Supported commands?
          switch ( _PCMD )
          {
            // Invalid command
            default:
              // Return error
              DpaApiReturnPeripheralError( ERROR_PCMD );

              // Sensor enumeration
            case PCMD_STD_ENUMERATE:
              if ( _DpaDataLength != 0 )
                goto _ERROR_DATA_LEN;

              // Then just enumerate their types
              _DpaMessage.Response.PData[0] =
STD_SENSOR_TYPE_TEMPERATURE;
              _DpaMessage.Response.PData[1] =
STD_SENSOR_TYPE_TEMPERATURE;
              _DpaMessage.Response.PData[2] =
STD_SENSOR_TYPE_BINARYDATA7;
              _DpaMessage.Response.PData[3] =
STD_SENSOR_TYPE_BINARYDATA7;
              W = SENSORS_COUNT;
              goto _W2_DpaDataLength;

              // Supported commands. They are handled the same way
except one "if" at StoreValue() method
            case PCMD_STD_SENSORS_READ_VALUES:
            case PCMD_STD_SENSORS_READ_TYPES_AND_VALUES:
            {
              // No sensor bitmap specified? W = _DpaDataLength. Note:
W is used to avoid MOVLB at next if
              W = _DpaDataLength;
              if ( W == 0 ) // Note: must not modify W
              {
                // Actually clears the bitmap
#if &_DpaMessage.Request.PData[0] != &bufferRF[0]
#error
#endif
                clearBufferRF();
                // Simulate 1st only sensor in the bitmap (states of
the other unimplemented sensors do not care)
                _DpaMessage.Request.PData[0].0 = 1;
                // Bitmap is 32 bits long = 4
                _DpaDataLength = W = 4;
            }

              // Invalid bitmap (data) length (W = _DpaDataLength)?
              if ( W != 4 )
              {
_ERROR_DATA_LEN:
                // Return error
```

```
                DpaApiReturnPeripheralError( ERROR_DATA_LEN );
              }

              // Now read the sensors

              // Prepare pointer (minus 1, see below) to store sensor
(types and) values to
              // Note: 4 sensors at this example cannot return more
than DPA_MAX_DATA_LENGTH bytes of data, so it does not have to be
checked...
              // ... If it would be the case, then ERROR_FAIL must be
returned
              FSR1 = &_DpaMessage.Response.PData[-1];

              // Store bitmap of sensors to get values from
              uns8  sensorsBitmap = FSR1[1];

              // 1st sensor (index 0) selected?
              if ( sensorsBitmap.0 )
              {
                Get0_TemperatureTR();
                StoreValue( STD_SENSOR_TYPE_TEMPERATURE );
              }

              // 2nd sensor (index 1) selected?
              if ( sensorsBitmap.1 )
              {
                Get1_Temperature();
                StoreValue( STD_SENSOR_TYPE_TEMPERATURE );
              }

              // 3rd sensor (index 2) selected?
              if ( sensorsBitmap.2 )
              {
                Get2_BinaryData_Light();
                StoreValue( STD_SENSOR_TYPE_BINARYDATA7 );
              }

              // 4th sensor (index 3) selected?
              if ( sensorsBitmap.3 )
              {
                Get3_BinaryData_Potentiometer();
                StoreValue( STD_SENSOR_TYPE_BINARYDATA7 );
              }

              // Compute returned data bytes count
              W = FSR1L - ( (uns16)&_DpaMessage.Response.PData[0] &
0xFF ) + 1;
              // Optimization: return W long block of bytes at
response
_W2_DpaDataLength:
              _DpaDataLength = W;
              goto DpaHandleReturnTRUE;
            }
```

```
        }
      }

        break;
  }

      // ------------------------------------------------
    case DpaEvent_FrcValue:
      // Called to get FRC value

      // FSR1 for optimization purposes (avoid MOVLB) will be used to
point to DataOutBeforeResponseFRC[0...]
      FSR1 = &DataOutBeforeResponseFRC[0];
      // Check for correct FRC user data
      if ( *FSR1++ /*DataOutBeforeResponseFRC[0]*/ == PNUM_STD_SENSORS
)
      {
        // Actually used sensor index
        uns8 sensorIndex = FSR1[1] /*DataOutBeforeResponseFRC[2]*/ &
0x1f;
        // Test sensor type
        switch ( *FSR1++ /*DataOutBeforeResponseFRC[1]*/ )
        {
          default:
            goto DpaHandleReturnFALSE;

            // No type specified, use specified index value
          case 0x00:
            goto _KeepSensorIndex;

            // For other types make the index value based on the
requested index value and sensor type
          case STD_SENSOR_TYPE_TEMPERATURE:
            if ( sensorIndex > 1 )
              goto DpaHandleReturnFALSE;
            W = 0 + sensorIndex;
            break;

          case STD_SENSOR_TYPE_BINARYDATA7:
            if ( sensorIndex > 1 )
              goto DpaHandleReturnFALSE;
            W = 2 + sensorIndex;
            break;
        }

        // New sensor index based on type and requested index
        sensorIndex = W;
_KeepSensorIndex:

        // Test for supported FRC commands
        switch ( _PCMD )
        {
          default:
            goto DpaHandleReturnFALSE;
```

```
          case FRC_STD_SENSORS_BIT:
          case FRC_STD_SENSORS_1B:
          case FRC_STD_SENSORS_2B:
            switch ( sensorIndex )
            {
              default:
                goto DpaHandleReturnFALSE;

              case 0:
                Carry = Get0_TemperatureTR();
                break;

              case 1:
                Carry = Get1_Temperature();
                break;

              case 2:
                Carry = Get2_BinaryData_Light();
                break;

              case 3:
                Carry = Get3_BinaryData_Potentiometer();
                break;
            }

            // This type of FRC is not valid for the specified sensor
            if ( !Carry )
              goto DpaHandleReturnFALSE;

            break;
        }

        // Some sensor was measured by FRC, check if there is a sleep
request
        FSR1++;
        if ( INDF1.7 ) // Note: same as DataOutBeforeResponseFRC[3].7
        {
          // Remember sleep parameters to go to sleep at the Idle
event later
          PerOSSleep_Request.Time.low8 = FSR1[4 - 3]; // Note: same as
DataOutBeforeResponseFRC[4];
          PerOSSleep_Request.Time.high8 = FSR1[5 - 3]; // Note: same
as DataOutBeforeResponseFRC[5];
          PerOSSleep_Request.Control = FSR1[6 - 3]; // Note: same as
DataOutBeforeResponseFRC[6];
        }
      }

      break;

      // -------------------------------------------------
    case DpaEvent_FrcResponseTime:
      // Called to get FRC response time
```

```
      // In this example the FRC commands are fast
      switch ( DataOutBeforeResponseFRC[0] )
      {
        case FRC_STD_SENSORS_BIT:
        case FRC_STD_SENSORS_1B:
        case FRC_STD_SENSORS_2B:
          responseFRCvalue = _FRC_RESPONSE_TIME_40_MS;
          break;
      }
      break;
}
DpaHandleReturnFALSE:
  return FALSE;
}

//#################################################################################
// Increases FSR1 and then stores the byte
void setPlusPlusINDF1( uns8 data @ W )
//#################################################################################
{
  FSR1++; // Note: must not modify W
  setINDF1( data );
}

//#################################################################################
// Stores measured sensor value byte(s) and optionally sensor type to
the FSR[+1...]
void StoreValue( uns8 sensorType )
//#################################################################################
{
  // Is the sensor type to be stored too?
  if ( _PCMD == PCMD_STD_SENSORS_READ_TYPES_AND_VALUES )
    setPlusPlusINDF1( sensorType );

  // Store lower value byte
  setPlusPlusINDF1( sensorValue.low8 );

  // No more value bytes to store?
  if ( sensorType.7 != 0 )
    return;

  // Store higher value byte
  setPlusPlusINDF1( sensorValue.high8 );

  // Note: this example implements sensors returning only 1 or 2 bytes
of data. If another data widths are returned, then it must be
implemented explicitly.
}
```

```
//#####################################################################
########################
bit setFRCerror1B()
//#####################################################################
########################
{
  responseFRCvalue = 2;
  return TRUE;
}

//#####################################################################
########################
bit setFRCerror2B()
//#####################################################################
########################
{
  responseFRCvalue2B = 2;
  return TRUE;
}

//#####################################################################
########################
bit sensorError;
bit AdjustFrcTemperature()
//#####################################################################
########################
{
  // Test for supported FRC commands
  switch ( _PCMD )
  {
    default:
      return FALSE;

    case FRC_STD_SENSORS_1B:
      // Return sensor FRC value 1B
      // Check for out of limits
      if ( sensorError || (int16)sensorValue > (int16)( 105.5 * 16 )
|| (int16)sensorValue < ( (int16)-20 * 16 ) )
        return setFRCerror1B();

      // Convert to the "F = ( T + 22 ) * 2 " from 1/16 resolution
      uns16 _sensorValue = sensorValue + 4; // Note: do rounding when
/8
      responseFRCvalue = (uns8)( _sensorValue / 8 ) + 44;
      break;

    case FRC_STD_SENSORS_2B:
      // Return sensor FRC value 2B
      if ( sensorError )
        return setFRCerror2B();

      responseFRCvalue2B = sensorValue ^ 0x8000;
      break;
  }
```

```
  return TRUE;
}

//#####################################################################
########################
// Sensor index 0: measure temperature using the TR sensor
bit Get0_TemperatureTR()
//#####################################################################
########################
{
  // Make sure FSR1 is not modified

  // When error, then adjust the standard error values
  sensorError = FALSE;
  // Measure temperature using TR sensor
  if ( getTemperature() == -128 )
  {
    sensorError = TRUE;
    sensorValue = 0x8000;
  }

  // Return sensor value
  sensorValue = param3;
  // Handler FRC
  return AdjustFrcTemperature();
}

//#####################################################################
########################
// Sensor index 1: measure temperature using one of the DDC-SE01
sensors
bit Get1_Temperature()
//#####################################################################
########################
{
  // Make sure FSR1 is not modified

  // Measure temperature using DDC-SE01 sensors
  sensorError = FALSE;
  // Read temperature and check for an error
  GetTemperature();
  // When error, return standard (FRC) error value(s)
  if ( sensorValue == ERROR_TEMPERATURE )
    sensorError = TRUE;

  // FrcValues
  return AdjustFrcTemperature();
}

//#####################################################################
########################
bit AdjustFrcBinaryData()
```

```
//#####################################################################
########################
{
  // Test for supported FRC commands
  switch ( _PCMD )
  {
    default:
      return FALSE;

    case FRC_STD_SENSORS_BIT:
      // If there is a sensor error, 2-bit FRC cannot indicate it, it
returns [01]

      // Number of shifts to get the bit out of the return value
      uns8 bitLoop = ( INDF1 >> 5 ) + 1;
      // Value to get the bit from
      W = sensorValue.low8;
      do
      {
        // Get the bit to Carry
        W = rr( W );
        // Next bit
      } while ( --bitLoop != 0 ); // Note: must not modify W and Carry
      // Current (prepared by DPA) FRC value is [01], change it to
[11] (means bit is set)
      responseFRCvalue.1 = 1; // Note: must not modify Carry
      // Is bit set?
      if ( !Carry )
        // Bit is NOT set, return [10]
        responseFRCvalue.0 = 0;
      break;

    case FRC_STD_SENSORS_1B:
      responseFRCvalue = sensorValue.low8 + 4;
      break;
  }
  return TRUE;
}

//#####################################################################
########################
// Sensor index 2: returns light intensity indicator value using DDC-
SE01
bit Get2_BinaryData_Light()
//#####################################################################
########################
{
  // Make sure FSR1 is not modified

  // ADC initialization (for more info see PIC datasheet) pin C1 (AN0)
as analog input
  ANSELA.0 = 1;
  // ADC setting (AN0 channel)
  ADCON0 = 0b0.00000.01;
```

```
  // Read ADC
  sensorValue.low8 = ReadAdc() / 2;
  // Return sensor FRC value
  return AdjustFrcBinaryData();
}

//######################################################################
########################
// Sensor index 3: returns potentiometer value using DDC-SE01
bit Get3_BinaryData_Potentiometer()
//######################################################################
########################
{
  // Make sure FSR1 is not modified

  // ADC initialization (for more info see PIC datasheet) pin C5 (AN4)
as analog input
  ANSELA.5 = 1;
  // ADC setting (AN4 channel)
  ADCON0 = 0b0.00100.01;
  // Read ADC
  sensorValue.low8 = ReadAdc() / 2;
  // Return sensor FRC value
  return AdjustFrcBinaryData();
}

//######################################################################
########################
// OneWire and Dallas 18B20 routines
//######################################################################
########################

//######################################################################
########################
void Delay5us( uns8 val @ W ) // Absolutely precise timing but val !=
0
//######################################################################
########################
{
  // 16 MHz
  // + 0.75us ( W=val, Call )
  for ( ;; )
  {         // loop time
    nop2(); // 0.50us
    nop2(); // 1.00us
    nop2(); // 1.50us
    nop2(); // 2.00us
    nop2(); // 2.50us
    nop2(); // 3.00us
    nop();  // 3.25us
    if ( --val == 0 ) // + 0.75us (W--, BTFS )
      return;         // + 0.25us
    nop2(); // 4.50us
  }         // 5.00us (Goto)
```

```
}
//#########################################################################
########################

#define OneWireData0()  { OneWire_TRIS = 0; }      // 1us [0.5us @
16MHz]
#define OneWireData1()  { OneWire_TRIS = 1; }      // 1us [0.5us @
16MHz]

//#########################################################################
########################
void OneWireWriteByte( uns8 byte )
//#########################################################################
########################
{
  uns8 bitLoop = 8;
  do
  {
    // Next sequence is time precision critical
    GIE = FALSE;

    OneWireData0();
    nop2();          // 1 us [0.5 us]
    nop2();          // [1.0 us]
    if ( byte.0 )    // 2.5 us [1.75us]
      OneWireData1();

    // End of time precision critical sequence
    GIE = TRUE;

    // 60us minimum in total, does not have to be precise
    Delay5us( ( 60 - 3 ) / 5 + 1 );

    OneWireData1();

    byte >>= 1;
  } while ( --bitLoop != 0 );
}

//#########################################################################
########################
uns8 OneWireReadByte()
//#########################################################################
########################
{
  uns8 result;
  uns8 bitLoop = 8;
  do
  {
    // Next sequence is time precision critical
    GIE = FALSE;

    OneWireData0();
    nop2();          // 1 us [0.5 us]
```

```
#if F_OSC == 16000000
    nop2();         // [1.0 us]
#endif
    OneWireData1();         // 2 us [1.5 us]
    Delay5us( 15 / 5 );     // 17 us [16.5 us]

    Carry = 0;              // 17.5 us [16.75 us]
    if ( OneWire_IO_IN )    // 18.5 us [ 17.25 us] (condition must not
modify Carry)
      Carry = 1;

    // End of time precision critical sequence
    GIE = TRUE;             // must not modify Carry
    result = rr( result );

    // 60us minimum in total, does not have to be precise
    Delay5us( ( 60 - 20 ) / 5 + 1 );
  } while ( --bitLoop != 0 );

  return result;
}

//#########################################################################
########################
bit OneWireReset()
//#########################################################################
########################
{
  // Setting the pin once to low is enough
  OneWire_IO_OUT = 0;
  // Reset pulse
  OneWireData0();
  Delay5us( 500 / 5 );
  // Reset pulse end
  OneWireData1();
  // Next sequence is time precision critical
  GIE = FALSE;
  // Wait for presence pulse
  Delay5us( 70 / 5 );
  // End of time precision critical sequence
  GIE = TRUE;
  // Presence pulse?
  if ( OneWire_IO_IN )
  {
    // No presence, finish initialization sequence
    Delay5us( ( 500 - 70 ) / 5 );
    return FALSE;
  }
  else
  {
    // Presence OK, finish initialization sequence
    Delay5us( ( 500 - 70 ) / 5 );
    return TRUE;
  }
```

```
}

//#########################################################################
########################
void OneWireCmd( uns8 cmd )
//#########################################################################
########################
{
  // OneWire: Skip ROM
  OneWireWriteByte( CMD_SKIPROM );
  // OneWire: Send command
  OneWireWriteByte( cmd );
}

//#########################################################################
########################
bit Ds18B20WriteConfig( uns8 value )
//#########################################################################
########################
{
  if ( OneWireReset() )
  {
    // Write Scratchpad
    OneWireCmd( CMD_WSCRATCHPAD );

    // Write TL = ? (we dot not care the value)
    OneWireWriteByte( W );
    // Write TH = ? (we dot not care the value)
    OneWireWriteByte( W );
    // Write Config byte
    OneWireWriteByte( value );

    if ( OneWireReset() )
    {
      //  Copy Scratchpad
      OneWireCmd( CMD_CPYSCRATCHPAD );
      return TRUE;
    }
  }
  return FALSE;
}

//#########################################################################
########################
void writeToSSPCON2( uns8 value @ param4.high8 )
//#########################################################################
########################
{
  writeToRAM( &SSPCON2, value );
}

//#########################################################################
########################
void writeOredToSSPCON2( uns8 value @ param4.high8 )
```

```
//#########################################################################
########################
{
  writeToSSPCON2( SSPCON2 | value );
}

//#########################################################################
########################
// I2C routines
//#########################################################################
########################

bit i2cTimeout;

//#########################################################################
########################
void i2c_init()
//#########################################################################
########################
{
  // SCL as input (SIM C6)
  TRISC.3 = 1;
  // SDA as input (SIM C7)
  TRISC.4 = 1;

  // I2C master mode      SSPCON = 0b00111000
  writeToRAM( &SSPCON1, 0x38 );
  writeToSSPCON2( 0x00 );

  // 50 kHz SCL frequency
  SSPADD = ( F_OSC / 50000 / 4 ) - 2;
  // Disable slew rate control
  SMP = 1;
}

//#########################################################################
########################
void i2c_shutdown()
//#########################################################################
########################
{
  // I2C master mode      SSPCON = 0
  writeToRAM( &SSPCON1, 0x00 );
}

//#########################################################################
########################
void i2c_waitForIdle()
//#########################################################################
########################
{
  i2cTimeout = FALSE;
  uns8 timeout;
  // Wait for idle and not writing
```

```
  timeout = 0;
  while ( ( SSPCON2 & 0b0001.1111 ) != 0 || RW_ )
    if ( ++timeout == 0 )
    {
      i2cTimeout = TRUE;
      break;
    }
}

//#########################################################################
########################
void i2c_start()
//#########################################################################
########################
{
  i2c_waitForIdle();
  // SEN = 1
  writeOredToSSPCON2( 0x01 );
}

//#########################################################################
########################
void i2c_repStart()
//#########################################################################
########################
{
  i2c_waitForIdle();
  // RSEN = 1
  writeOredToSSPCON2( 0x02 );
}

//#########################################################################
########################
void i2c_stop()
//#########################################################################
########################
{
  i2c_waitForIdle();
  // PEN = 1
  writeOredToSSPCON2( 0x04 );
}

//#########################################################################
########################
uns8 i2c_read( bit ack )
//#########################################################################
########################
{
  i2c_waitForIdle();
  // RCEN = 1
  writeOredToSSPCON2( 0x08 );

  i2c_waitForIdle();
```

```
  uns8 i2cReadData @ userReg0;
  i2cReadData = SSPBUF;

  i2c_waitForIdle();

  if ( ack )
    // Acknowledge, ACKDT = 0
    writeToSSPCON2( SSPCON2 & 0xDF );
  else
    // Not acknowledge, ACKDT = 1
    writeOredToSSPCON2( 0x20 );

  // Send acknowledge sequence, ACKEN = 1
  writeOredToSSPCON2( 0x10 );
  return i2cReadData;
}

//#####################################################################
########################
void i2c_write( uns8 i2cWriteData @ param2 )
//#####################################################################
########################
{
  i2c_waitForIdle();
  SSPBUF = i2cWriteData;
}

//#####################################################################
########################
void MCP9802GetTemp()
//#####################################################################
########################
{
  sensorValue = ERROR_TEMPERATURE;

  i2c_start();
  if ( i2cTimeout )
    return;

  // MCP9802 address
  i2c_write( I2C_ADR );
  // pointer: 1 = configuration register
  i2c_write( 0x01 );
  // configuration: 9-bit ADC
  i2c_write( 0x00 );
  i2c_stop();

  i2c_start();
  // MCP9802 address
  i2c_write( I2C_ADR );
  // pointer: 0 = temperature
  i2c_write( 0 );
  i2c_stop();
```

```
  i2c_start();
  // MCP9802 address + read
  i2c_write( I2C_ADR | 1 );
  // store the result
  sensorValue.high8 = i2c_read( TRUE );
  sensorValue.low8 = i2c_read( FALSE );
  i2c_stop();
}

//#########################################################################
########################
// Other routines
//#########################################################################
########################

//#########################################################################
########################
void GetTemperature()
//#########################################################################
########################
{
  // Reads temperature from an enabled sensor
  if ( isDS18B20 )
    // Temperature is ready at the background
    sensorValue = temperature;
  else
  {
    // Temperature value must be read from I2C sensor
    MCP9802GetTemp();
    if ( sensorValue != ERROR_TEMPERATURE )
    {
      sensorValue += 0x08;
      sensorValue /= 0x10;
    }
  }
}

//#########################################################################
########################
uns8 ReadAdc()
//#########################################################################
########################
{
  // ADC result - left justified, Fosc/8
  ADCON1 = 0b0001.0000;
  // Do a smallest delay for ADC ACQUISITION TIME
  waitMS( 1 );
  // start ADC
  GO = 1;
  // wait for ADC finish
  while ( GO );
  return ADRESH;
}
```

```
//#####################################################################
#######################
// Default Custom DPA Handler header; 2nd include to implement Code
bumper to detect too long code of the Custom DPA Handler (modify the
path according to your setup)
#include "../../Development/include/DPA/DPAcustomHandler.h"
//#####################################################################
#######################
```

ANEXO 2.2

LISTADO CÓDIGO NODERED (FORMATO JSON)


```
[
    {
        "id": "cf072c02.5646f8",
        "type": "tab",
        "label": "Request",
        "disabled": false,
        "info": ""
    },
    {
        "id": "b5b5a973.429a8",
        "type": "tab",
        "label": "Reception",
        "disabled": false,
        "info": ""
    },
    {
        "id": "86f75d07.20ec1",
        "type": "tab",
        "label": "Visualization",
        "disabled": false,
        "info": ""
    },
    {
        "id": "cd8e3a6.551bec8",
        "type": "ibmiot",
        "z": "",
        "name": "api key",
        "keepalive": "130",
        "serverName": "",
        "cleansession": true,
        "appId": "",
        "shared": false
    },
    {
        "id": "a250971c.5a32e8",
        "type": "ibmiot",
        "z": "",
        "name": "api key",
        "keepalive": "60",
        "serverName": "",
        "cleansession": true,
        "appId": "",
        "shared": false
    },
    {
        "id": "9407a4ee.3fae78",
        "type": "mqtt-broker",
        "z": "",
        "name": "",
        "broker": "127.0.0.1",
```

```
        "port": "1883",
        "clientid": "",
        "usetls": false,
        "compatmode": true,
        "keepalive": "60",
        "cleansession": true,
        "birthTopic": "",
        "birthQos": "0",
        "birthPayload": "",
        "closeTopic": "",
        "closeQos": "0",
        "closePayload": "",
        "willTopic": "",
        "willQos": "0",
        "willPayload": ""
    },
    {
        "id": "37c2fda.cdccb82",
        "type": "ui_base",
        "z": 0,
        "theme": {
            "name": "theme-light",
            "lightTheme": {
                "default": "#0094CE",
                "baseColor": "#0094CE",
                "baseFont": "-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif",
                "edited": true,
                "reset": false
            },
            "darkTheme": {
                "default": "#097479",
                "baseColor": "#097479",
                "baseFont": "-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif",
                "edited": false
            },
            "customTheme": {
                "name": "Untitled Theme 1",
                "default": "#4B7930",
                "baseColor": "#4B7930",
                "baseFont": "-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif"
            },
            "themeState": {
                "base-color": {
                    "default": "#0094CE",
                    "value": "#0094CE",
                    "edited": false
                },
                "page-titlebar-backgroundColor": {
                    "value": "#0094CE",
                    "edited": false
                },
```

```
                "page-backgroundColor": {
                    "value": "#fafafa",
                    "edited": false
                },
                "page-sidebar-backgroundColor": {
                    "value": "#ffffff",
                    "edited": false
                },
                "group-textColor": {
                    "value": "#1bbfff",
                    "edited": false
                },
                "group-borderColor": {
                    "value": "#ffffff",
                    "edited": false
                },
                "group-backgroundColor": {
                    "value": "#ffffff",
                    "edited": false
                },
                "widget-textColor": {
                    "value": "#111111",
                    "edited": false
                },
                "widget-backgroundColor": {
                    "value": "#0094ce",
                    "edited": false
                },
                "widget-borderColor": {
                    "value": "#ffffff",
                    "edited": false
                },
                "base-font": {
                    "value": "-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif"
                }
            },
            "angularTheme": {
                "primary": "indigo",
                "accents": "blue",
                "warn": "red",
                "background": "grey"
            }
        },
        "site": {
            "name": "Temperaturas",
            "hideToolbar": "false",
            "allowSwipe": "false",
            "allowTempTheme": "true",
            "dateFormat": "DD/MM/YYYY",
            "sizes": {
                "sx": 48,
                "sy": 48,
                "gx": 6,
```

```
                "gy": 6,
                "cx": 6,
                "cy": 6,
                "px": 0,
                "py": 0
            }
        }
    },
    {
        "id": "288c68c6.27362",
        "type": "ui_tab",
        "z": 0,
        "name": "Temperatura",
        "icon": "dashboard",
        "order": 1
    },
    {
        "id": "a9eedb24.b461f",
        "type": "ui_group",
        "z": 0,
        "name": "Nodo 1",
        "tab": "",
        "order": 1,
        "disp": true,
        "width": "6",
        "collapse": false
    },
    {
        "id": "7fa37e4d.d7bca",
        "type": "ui_group",
        "z": 0,
        "name": "Nodo 2",
        "tab": "",
        "order": 4,
        "disp": true,
        "width": "6",
        "collapse": false
    },
    {
        "id": "d2e33307.45399",
        "type": "ui_group",
        "z": 0,
        "name": "Default",
        "tab": "",
        "disp": true,
        "width": "6",
        "collapse": false
    },
    {
        "id": "c07d8bfd.ed1f5",
        "type": "ui_group",
        "z": 0,
        "name": "Group 1",
        "tab": "",
```

```
        "order": 1,
        "disp": true,
        "width": 6
    },
    {
        "id": "6ff7f81b.62e928",
        "type": "ui_group",
        "z": 0,
        "name": "TemperatureIQRF",
        "tab": "",
        "order": 1,
        "disp": false,
        "width": "6"
    },
    {
        "id": "b42c9eaf.67d018",
        "type": "ui_group",
        "z": 0,
        "name": "Light",
        "tab": "",
        "order": 4,
        "disp": false,
        "width": "6"
    },
    {
        "id": "504fba86.72204c",
        "type": "ui_group",
        "z": 0,
        "name": "Light",
        "tab": "",
        "order": 6,
        "disp": true,
        "width": "6"
    },
    {
        "id": "44ebafcc.1cf",
        "type": "ui_group",
        "z": 0,
        "name": "Air condition",
        "tab": "",
        "order": 7,
        "disp": true,
        "width": "6"
    },
    {
        "id": "36e30f3a.bcd328",
        "type": "ui_group",
        "z": 0,
        "name": "Temperatura (Nodo 2)",
        "tab": "288c68c6.27362",
        "order": 2,
        "disp": true,
        "width": "6",
        "collapse": true
```

```
},
{
    "id": "9b292989.f5d9b",
    "type": "ui_group",
    "z": 0,
    "name": "Temperatura (Nodo 1)",
    "tab": "288c68c6.27362",
    "order": 1,
    "disp": true,
    "width": "6",
    "collapse": true
},
{
    "id": "2eeb502.da9e83",
    "type": "ui_group",
    "z": 0,
    "name": "Temperatura (Nodo 2)",
    "tab": "",
    "order": 3,
    "disp": true,
    "width": "6",
    "collapse": true
},
{
    "id": "ab533106.2d18f8",
    "type": "ui_group",
    "z": 0,
    "name": "Temperatura (Nodo 1)",
    "tab": "",
    "order": 2,
    "disp": true,
    "width": "6",
    "collapse": true
},
{
    "id": "e7241457.f61438",
    "type": "ui_group",
    "z": 0,
    "name": "Temperatura (Nodo 3)",
    "tab": "288c68c6.27362",
    "order": 3,
    "disp": true,
    "width": "6",
    "collapse": true
},
{
    "id": "b53bdc19.c4b6e8",
    "type": "ui_group",
    "z": 0,
    "name": "Temperatura (Nodo 4)",
    "tab": "288c68c6.27362",
    "order": 4,
    "disp": true,
    "width": "6",
```

```
        "collapse": true
    },
    {
        "id": "e2871101.d4dbc8",
        "type": "ui_group",
        "z": 0,
        "name": "Temperatura (Nodo 5)",
        "tab": "288c68c6.27362",
        "order": 5,
        "disp": true,
        "width": "6",
        "collapse": true
    },
    {
        "id": "a6a0def4.5548a",
        "type": "ui_group",
        "z": 0,
        "name": "Temperatura (Nodo 6)",
        "tab": "288c68c6.27362",
        "order": 6,
        "disp": true,
        "width": "6",
        "collapse": true
    },
    {
        "id": "f4a086d8.64361",
        "type": "ui_group",
        "z": 0,
        "name": "Temperatura (Nodo 7)",
        "tab": "288c68c6.27362",
        "order": 7,
        "disp": true,
        "width": "6",
        "collapse": true
    },
    {
        "id": "5ddee758.2fa97",
        "type": "ui_group",
        "z": 0,
        "name": "Temperatura (Nodo 9)",
        "tab": "288c68c6.27362",
        "order": 9,
        "disp": true,
        "width": "6",
        "collapse": true
    },
    {
        "id": "95e99112.4af158",
        "type": "ui_group",
        "z": 0,
        "name": "Temperatura (Nodo 8)",
        "tab": "288c68c6.27362",
        "order": 8,
        "disp": true,
```

```
        "width": "6",
        "collapse": false
    },
    {
        "id": "8bf78c5.c0162f",
        "type": "inject",
        "z": "cf072c02.5646f8",
        "name": "",
        "topic": "",
        "payload": "{\"ctype\": \"dpa\",\"type\": \"raw\",\"msgid\":
\"1510754980\",\"request\":
\"00.00.0d.00.ff.ff.e0.5e.01.01.00\",\"request_ts\":
\"\",\"confirmation\": \"\",\"confirmation_ts\": \"\",\"response\":
\"\",\"response_ts\": \"\"}",
        "payloadType": "json",
        "repeat": "",
        "crontab": "",
        "once": false,
        "onceDelay": 0.1,
        "x": 355,
        "y": 301.66664123535156,
        "wires": [
            [
                "a507d9c6.07aaa"
            ]
        ]
    },
    {
        "id": "2a87df21.dfc56",
        "type": "ibmiot out",
        "z": "cf072c02.5646f8",
        "authentication": "boundService",
        "apiKey": "a250971c.5a32e8",
        "outputType": "cmd",
        "deviceId": "RPI_GW",
        "deviceType": "iqrf_gw_device",
        "eventCommandType": "iqrf",
        "format": "json",
        "data": "msg.payload",
        "qos": 0,
        "name": "IBM IoT",
        "service": "registered",
        "x": 778.3333740234375,
        "y": 59.99999237060547,
        "wires": []
    },
    {
        "id": "d13e6603.d56408",
        "type": "debug",
        "z": "cf072c02.5646f8",
        "name": "",
        "active": false,
        "tosidebar": true,
        "console": false,
```

```
        "tostatus": false,
        "complete": "false",
        "x": 995,
        "y": 421.66664123535156,
        "wires": []
    },
    {
        "id": "b95312a3.38c16",
        "type": "inject",
        "z": "cf072c02.5646f8",
        "name": "2 min Trigger",
        "topic": "",
        "payload": "1",
        "payloadType": "num",
        "repeat": "120",
        "crontab": "",
        "once": true,
        "onceDelay": 0.1,
        "x": 180,
        "y": 60,
        "wires": [
            [
                "e7e891dc.3fedf8"
            ]
        ]
    },
    {
        "id": "e7e891dc.3fedf8",
        "type": "function",
        "z": "cf072c02.5646f8",
        "name": "Req sensor temp",
        "func": "var data={\n    type: \"raw\",\n    request: {\n
nadr: \"0x0000\", \n        pnum: \"0x0D\",\n        pcmd: \"0x00\",\n
hwpid: \"0xFFFF\",\n        pdata: \"0xE05E010100\",\n    },\n
timeout: 1000\n}\nmsg.payload=data;\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 380,
        "y": 60,
        "wires": [
            [
                "a0c890fa.a8ce38"
            ]
        ]
    },
    {
        "id": "a0c890fa.a8ce38",
        "type": "function",
        "z": "cf072c02.5646f8",
        "name": "JSONtoIQRF",
        "func": "out =
('000'+Number(msg.payload.request.nadr).toString(16)).slice(-2);\nout
+=
\".\"+(('000'+Number(msg.payload.request.nadr).toString(16)).slice(-
```

```
4)).substr(0,2);\nout +=
\".\"+('0'+Number(msg.payload.request.pnum).toString(16)).slice(-
2);\nout +=
\".\"+('0'+Number(msg.payload.request.pcmd).toString(16)).slice(-
2);\nout +=
\".\"+('000'+Number(msg.payload.request.hwpid).toString(16)).slice(-
2);\nout +=
\".\"+(('000'+Number(msg.payload.request.hwpid).toString(16)).slice(-
4)).substr(0,2);\n\n\n\nif (typeof msg.payload.request.pdata !==
'undefined' && msg.payload.request.pdata !== null) \n{\n    // jsou
pro nas data pdata !!!\n    // potreba zjistit delku stringu.\n    \n
len = msg.payload.request.pdata.length; \n    count = -len;\n
for(i=len;i>2;i=i-2)\n    {\n        count = count + 2;\n        out
+=
\".\"+(('000'+Number(msg.payload.request.pdata).toString(16)).slice(co
unt)).substr(0,2);\n    }\n    //msg.payload = count;\n    //return
msg;\n}\n\nmsg.payload={ctype:\"dpa\", type: msg.payload.type,
request: out, timeout: msg.payload.timeout};\nreturn msg;\n",
        "outputs": 1,
        "noerr": 0,
        "x": 590,
        "y": 60,
        "wires": [
            [
                "2a87df21.dfc56",
                "b111da92.b292f",
                "cbad87fb.f1f188"
            ]
        ]
    },
    {
        "id": "cbad87fb.f1f188",
        "type": "debug",
        "z": "cf072c02.5646f8",
        "name": "",
        "active": false,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "false",
        "x": 805,
        "y": 130.83334350585938,
        "wires": []
    },
    {
        "id": "a507d9c6.07aaa",
        "type": "function",
        "z": "cf072c02.5646f8",
        "name": "Req sensor temp",
        "func": "var data={\n    type: \"raw\",\n    request: {\n
nadr: \"0x0000\",\n        pnum: \"0x0D\",\n        pcmd: \"0x00\",\n
hwpid: \"0xFFFF\",\n        pdata: \"0xE05E010100\",\n    },\n
timeout: 1000\n}\nmsg.payload=data;\nreturn msg;",
        "outputs": 1,
```

```
        "noerr": 0,
        "x": 555,
        "y": 301.66664123535156,
        "wires": [
            [
                "da7ede1e.0f59d8"
            ]
        ]
    },
    {
        "id": "da7ede1e.0f59d8",
        "type": "function",
        "z": "cf072c02.5646f8",
        "name": "JSONtoIQRF",
        "func": "out =
('000'+Number(msg.payload.request.nadr).toString(16)).slice(-2);\nout
+=
\".\"+(('000'+Number(msg.payload.request.nadr).toString(16)).slice(-
4)).substr(0,2);\nout +=
\".\"+('0'+Number(msg.payload.request.pnum).toString(16)).slice(-
2);\nout +=
\".\"+('0'+Number(msg.payload.request.pcmd).toString(16)).slice(-
2);\nout +=
\".\"+('000'+Number(msg.payload.request.hwpid).toString(16)).slice(-
2);\nout +=
\".\"+(('000'+Number(msg.payload.request.hwpid).toString(16)).slice(-
4)).substr(0,2);\n\n\n\n\nif (typeof msg.payload.request.pdata !==
'undefined' && msg.payload.request.pdata !== null) \n{\n    // jsou
pro nas data pdata !!!\n    // potreba zjistit delku stringu.\n    \n
len = msg.payload.request.pdata.length; \n    count = -len;\n
for(i=len;i>2;i=i-2)\n    {\n        count = count + 2;\n        out
+=
\".\"+(('000'+Number(msg.payload.request.pdata).toString(16)).slice(co
unt)).substr(0,2);\n    }\n    //msg.payload = count;\n    //return
msg;\n}\n\nmsg.payload= {ctype:\"dpa\", type: msg.payload.type,
request: out, timeout: msg.payload.timeout};\nreturn msg;\n",
        "outputs": 1,
        "noerr": 0,
        "x": 765,
        "y": 301.66664123535156,
        "wires": [
            [
                "d13e6603.d56408"
            ]
        ]
    },
    {
        "id": "5c6f2af8.0ad9e4",
        "type": "link in",
        "z": "b5b5a973.429a8",
        "name": "",
        "links": [
            "4a24c128.04cc98",
            "8ce2f796.994448",
```

```
            "9e9ad9b9.3c1148",
            "12894694.8c8991"
        ],
        "x": 140,
        "y": 220,
        "wires": [
            [
                "e6f2f07c.b72d4"
            ]
        ]
    },
    {
        "id": "e6f2f07c.b72d4",
        "type": "switch",
        "z": "b5b5a973.429a8",
        "name": "ADR",
        "property": "payload.response.nadr",
        "propertyType": "msg",
        "rules": [
            {
                "t": "eq",
                "v": "0",
                "vt": "str"
            }
        ],
        "checkall": "true",
        "repair": false,
        "outputs": 1,
        "x": 235,
        "y": 220,
        "wires": [
            [
                "13445d74.1d0eab"
            ]
        ]
    },
    {
        "id": "13445d74.1d0eab",
        "type": "switch",
        "z": "b5b5a973.429a8",
        "name": "PNUM",
        "property": "payload.response.pnum",
        "propertyType": "msg",
        "rules": [
            {
                "t": "eq",
                "v": "13",
                "vt": "str"
            }
        ],
        "checkall": "true",
        "repair": false,
        "outputs": 1,
        "x": 370,
```

```
        "y": 220,
        "wires": [
            [
                "ef2972ff.a3ec08",
                "b15024ff.848d2",
                "2d4eb214.45704e",
                "19988543.bfecab",
                "f193634b.a8a878",
                "32dfd8ef.d3b76",
                "a90de5c1.fab51",
                "defa193d.c7f74",
                "d0df92e1.92aa78"
            ]
        ]
    },
    {
        "id": "ef2972ff.a3ec08",
        "type": "function",
        "z": "b5b5a973.429a8",
        "name": "Get_Node1Temp",
        "func": "var
res=msg.payload.response.pdata.split(\".\");\n\nparstemp1 =
parseInt(res[4],16);   //ff\nparstemp2 =  parseInt(res[3],16);
//d0\n\ntemp1 = parstemp1;\ntemp2 = parstemp2;\n\ntemp1 &=
0x7F;\ntemp2 &= 0xF0;\n\ntemp = (temp1<<4) + (temp2>>4);\n\ntemp2 =
parstemp2;\ntemp2 &= 0x0F;\n\ntemp2 *= 0.0625;\n\ntemp += temp2;
\n\nif(((parstemp1 & 0x80)>>7) != 1)\n{\n    temp = 2048 - temp;\n
temp = -temp;\n}\nif(temp == -2048)\n{\n    temp =
\"Desconectado\";\n}\nmsg.payload={\"Node1Temp\":temp};\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 590,
        "y": 200,
        "wires": [
            [
                "221e5ad2.700cce",
                "e766de59.985f28"
            ]
        ]
    },
    {
        "id": "221e5ad2.700cce",
        "type": "link out",
        "z": "b5b5a973.429a8",
        "name": "",
        "links": [
            "b59983ca.5f531"
        ],
        "x": 780,
        "y": 240,
        "wires": []
    },
    {
        "id": "b15024ff.848d2",
```

```
        "type": "function",
        "z": "b5b5a973.429a8",
        "name": "Get_Node2Temp",
        "func": "var
res=msg.payload.response.pdata.split(\".\");\n\nparstemp1 =
parseInt(res[6],16);   //ff\nparstemp2 =  parseInt(res[5],16);
//d0\n\ntemp1 = parstemp1;\ntemp2 = parstemp2;\n\ntemp1 &=
0x7F;\ntemp2 &= 0xF0;\n\ntemp = (temp1<<4) + (temp2>>4);\n\ntemp2 =
parstemp2;\ntemp2 &= 0x0F;\n\ntemp2 *= 0.0625;\n\ntemp += temp2;
\n\nif(((parstemp1 & 0x80)>>7) != 1)\n{\n    temp = 2048 - temp;\n
temp = -temp;\n}\nif(temp == -2048)\n{\n    temp =
\"Desconectado\";\n}\n\nmsg.payload={\"Node2Temp\":temp};\nreturn
msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 590,
        "y": 243,
        "wires": [
            [
                "221e5ad2.700cce"
            ]
        ]
    },
    {
        "id": "e766de59.985f28",
        "type": "debug",
        "z": "b5b5a973.429a8",
        "name": "Temp",
        "active": false,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "x": 1014,
        "y": 125,
        "wires": []
    },
    {
        "id": "b59983ca.5f531",
        "type": "link in",
        "z": "86f75d07.20ec1",
        "name": "",
        "links": [
            "221e5ad2.700cce",
            "d1ad4317.d8c0e"
        ],
        "x": 314.9999694824219,
        "y": 591.0000076293945,
        "wires": [
            [
                "3f8231eb.ad0b16",
                "b28ba432.6dfe78",
                "35f0d8a.30564a8",
                "3e372c7c.810b6c",
```

```
                "f9d1499.e9a8e38",
                "659b97d1.9570d",
                "5e3d4fc4.111a3",
                "b59448db.491668",
                "fafe2333.284ed8",
                "13a81b9b.68d6cc",
                "2562d4a2.cff3cc",
                "81c04fd3.b5ab2"
            ]
        ]
    },
    {
        "id": "13a81b9b.68d6cc",
        "type": "debug",
        "z": "86f75d07.20ec1",
        "name": "Complete data",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "x": 393.99993896484375,
        "y": 228,
        "wires": []
    },
    {
        "id": "3f8231eb.ad0b16",
        "type": "function",
        "z": "86f75d07.20ec1",
        "name": "ShowNode2Temp",
        "func": "msg.payload=msg.payload.Node2Temp;\nif(msg.payload >
-2000 && msg.payload.Node2Temp !== \"ERROR\")\n{\n    msg.payload =
Math.round(msg.payload * 10)/10;\n    return msg;\n}\nelse return
msg;\nreturn;",
        "outputs": 1,
        "noerr": 0,
        "x": 647.9999694824219,
        "y": 296,
        "wires": [
            [
                "d640a5.3ba80758",
                "22bbf70f.fe3f78"
            ]
        ]
    },
    {
        "id": "b28ba432.6dfe78",
        "type": "function",
        "z": "86f75d07.20ec1",
        "name": "ShowNode1Temp",
        "func": "msg.payload=msg.payload.Node1Temp;\nif(msg.payload >
-2000 && msg.payload.Node1Temp !== \"ERROR\")\n{\n    msg.payload =
Math.round(msg.payload * 10)/10;\n    return msg;\n}\nelse\n    return
msg;\n\nreturn;",
```

```
        "outputs": 1,
        "noerr": 0,
        "x": 647.9999694824219,
        "y": 219,
        "wires": [
            [
                "9ad8a506.cc44e",
                "e2e3bd26.49064"
            ]
        ]
    },
    {
        "id": "2562d4a2.cff3cc",
        "type": "ibmiot out",
        "z": "86f75d07.20ec1",
        "authentication": "boundService",
        "apiKey": "cd8e3a6.551bec8",
        "outputType": "evt",
        "deviceId": "RPI_GW",
        "deviceType": "iqrf_gw_device",
        "eventCommandType": "show",
        "format": "json",
        "data": "msg.payload",
        "qos": 0,
        "name": "IBM IoT",
        "service": "registered",
        "x": 417,
        "y": 288.0000305175781,
        "wires": []
    },
    {
        "id": "2d4eb214.45704e",
        "type": "function",
        "z": "b5b5a973.429a8",
        "name": "Get_Node3Temp",
        "func": "var
res=msg.payload.response.pdata.split(\".\");\n\nparstemp1 =
parseInt(res[8],16);   //ff\nparstemp2 =  parseInt(res[7],16);
//d0\n\ntemp1 = parstemp1;\ntemp2 = parstemp2;\n\ntemp1 &=
0x7F;\ntemp2 &= 0xF0;\n\ntemp = (temp1<<4) + (temp2>>4);\n\ntemp2 =
parstemp2;\ntemp2 &= 0x0F;\n\ntemp2 *= 0.0625;\n\ntemp += temp2;
\n\nif(((parstemp1 & 0x80)>>7) != 1)\n{\n    temp = 2048 - temp;\n
temp = -temp;\n}\nif(temp == -2048)\n{\n    temp =
\"Desconectado\";\n}\n\nmsg.payload={\"Node3Temp\":temp};\nreturn
msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 590,
        "y": 284,
        "wires": [
            [
                "221e5ad2.700cce"
            ]
        ]
```

```
    },
    {
        "id": "19988543.bfecab",
        "type": "function",
        "z": "b5b5a973.429a8",
        "name": "Get_Node4Temp",
        "func": "var
res=msg.payload.response.pdata.split(\".\");\n\nparstemp1 =
parseInt(res[10],16);   //ff\nparstemp2 =  parseInt(res[9],16);
//d0\n\ntemp1 = parstemp1;\ntemp2 = parstemp2;\n\ntemp1 &=
0x7F;\ntemp2 &= 0xF0;\n\ntemp = (temp1<<4) + (temp2>>4);\n\ntemp2 =
parstemp2;\ntemp2 &= 0x0F;\n\ntemp2 *= 0.0625;\n\ntemp += temp2;
\n\nif(((parstemp1 & 0x80)>>7) != 1)\n{\n    temp = 2048 - temp;\n
temp = -temp;\n}\nif(temp == -2048)\n{\n    temp =
\"Desconectado\";\n}\n\nmsg.payload={\"Node4Temp\":temp};\nreturn
msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 590,
        "y": 326,
        "wires": [
            [
                "221e5ad2.700cce"
            ]
        ]
    },
    {
        "id": "f193634b.a8a878",
        "type": "function",
        "z": "b5b5a973.429a8",
        "name": "Get_Node5Temp",
        "func": "var
res=msg.payload.response.pdata.split(\".\");\n\nparstemp1 =
parseInt(res[12],16);   //ff\nparstemp2 =  parseInt(res[11],16);
//d0\n\ntemp1 = parstemp1;\ntemp2 = parstemp2;\n\ntemp1 &=
0x7F;\ntemp2 &= 0xF0;\n\ntemp = (temp1<<4) + (temp2>>4);\n\ntemp2 =
parstemp2;\ntemp2 &= 0x0F;\n\ntemp2 *= 0.0625;\n\ntemp += temp2;
\n\nif(((parstemp1 & 0x80)>>7) != 1)\n{\n    temp = 2048 - temp;\n
temp = -temp;\n}\nif(temp == -2048)\n{\n    temp =
\"Desconectado\";\n}\n\nmsg.payload={\"Node5Temp\":temp};\nreturn
msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 590,
        "y": 367,
        "wires": [
            [
                "221e5ad2.700cce"
            ]
        ]
    },
    {
        "id": "32dfd8ef.d3b76",
        "type": "function",
```

```
        "z": "b5b5a973.429a8",
        "name": "Get_Node6Temp",
        "func": "var
res=msg.payload.response.pdata.split(\".\");\n\nparstemp1 =
parseInt(res[14],16);   //ff\nparstemp2 =  parseInt(res[13],16);
//d0\n\ntemp1 = parstemp1;\ntemp2 = parstemp2;\n\ntemp1 &=
0x7F;\ntemp2 &= 0xF0;\n\ntemp = (temp1<<4) + (temp2>>4);\n\ntemp2 =
parstemp2;\ntemp2 &= 0x0F;\n\ntemp2 *= 0.0625;\n\ntemp += temp2;
\n\nif(((parstemp1 & 0x80)>>7) != 1)\n{\n    temp = 2048 - temp;\n
temp = -temp;\n}\nif(temp == -2048)\n{\n    temp =
\"Desconectado\";\n}\n\nmsg.payload={\"Node6Temp\":temp};\nreturn
msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 590,
        "y": 407,
        "wires": [
            [
                "221e5ad2.700cce"
            ]
        ]
    },
    {
        "id": "a90de5c1.fab51",
        "type": "function",
        "z": "b5b5a973.429a8",
        "name": "Get_Node7Temp",
        "func": "var
res=msg.payload.response.pdata.split(\".\");\n\nparstemp1 =
parseInt(res[16],16);   //ff\nparstemp2 =  parseInt(res[15],16);
//d0\n\ntemp1 = parstemp1;\ntemp2 = parstemp2;\n\ntemp1 &=
0x7F;\ntemp2 &= 0xF0;\n\ntemp = (temp1<<4) + (temp2>>4);\n\ntemp2 =
parstemp2;\ntemp2 &= 0x0F;\n\ntemp2 *= 0.0625;\n\ntemp += temp2;
\n\nif(((parstemp1 & 0x80)>>7) != 1)\n{\n    temp = 2048 - temp;\n
temp = -temp;\n}\nif(temp == -2048)\n{\n    temp =
\"Desconectado\";\n}\n\nmsg.payload={\"Node7Temp\":temp};\nreturn
msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 590,
        "y": 447,
        "wires": [
            [
                "221e5ad2.700cce"
            ]
        ]
    },
    {
        "id": "defa193d.c7f74",
        "type": "function",
        "z": "b5b5a973.429a8",
        "name": "Get_Node8Temp",
        "func": "var
res=msg.payload.response.pdata.split(\".\");\n\nparstemp1 =
```

```
parseInt(res[18],16);   //ff\nparstemp2 =  parseInt(res[17],16);
//d0\n\ntemp1 = parstemp1;\ntemp2 = parstemp2;\n\ntemp1 &=
0x7F;\ntemp2 &= 0xF0;\n\ntemp = (temp1<<4) + (temp2>>4);\n\ntemp2 =
parstemp2;\ntemp2 &= 0x0F;\n\ntemp2 *= 0.0625;\n\ntemp += temp2;
\n\nif(((parstemp1 & 0x80)>>7) != 1)\n{\n    temp = 2048 - temp;\n
temp = -temp;\n}\nif(temp == -2048)\n{\n    temp =
\"Desconectado\";\n}\n\nmsg.payload={\"Node8Temp\":temp};\nreturn
msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 590,
        "y": 487,
        "wires": [
            [
                "221e5ad2.700cce"
            ]
        ]
    },
    {
        "id": "13fb74da.940a13",
        "type": "function",
        "z": "cf072c02.5646f8",
        "name": "Disconected",
        "func": "if (msg.data === 0 && msg.payload.request.pnum !==
undefined && msg.payload.response === undefined) {\n
msg.payload={\n        type: \"raw\",\n        Node1Temp: \"Error\",\n
Node2Temp: \"Error\",\n        Node3Temp: \"Error\",\n
Node4Temp: \"Error\",\n        Node5Temp: \"Error\",\n
Node6Temp: \"Error\",\n        Node7Temp: \"Error\",\n
Node8Temp: \"Error\",\n    }\n    global.i = 0;\n    return msg;\n}",
        "outputs": 1,
        "noerr": 0,
        "x": 470,
        "y": 560,
        "wires": [
            [
                "d1ad4317.d8c0e",
                "2dd2de1a.695e4a"
            ]
        ]
    },
    {
        "id": "d1ad4317.d8c0e",
        "type": "link out",
        "z": "cf072c02.5646f8",
        "name": "",
        "links": [
            "b59983ca.5f531"
        ],
        "x": 655,
        "y": 560,
        "wires": []
    },
    {
```

```
        "id": "2dd2de1a.695e4a",
        "type": "debug",
        "z": "cf072c02.5646f8",
        "name": "DEbug",
        "active": false,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "x": 650,
        "y": 500,
        "wires": []
    },
    {
        "id": "b111da92.b292f",
        "type": "function",
        "z": "cf072c02.5646f8",
        "name": "",
        "func": "if (i === 'undefined'){\n    var i=0;\n}\nelse\n{\n
i++\n    if (i >= 3)\n    {\n        i=0\n        msg.payload=0\n
return msg;\n    }\n}\n",
        "outputs": 1,
        "noerr": 0,
        "x": 570,
        "y": 405,
        "wires": [
            [
                "13fb74da.940a13",
                "54be475b.b9cb8"
            ]
        ]
    },
    {
        "id": "54be475b.b9cb8",
        "type": "debug",
        "z": "cf072c02.5646f8",
        "name": "DEbug",
        "active": false,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "x": 775.8333740234375,
        "y": 441.6666259765625,
        "wires": []
    },
    {
        "id": "3e372c7c.810b6c",
        "type": "function",
        "z": "86f75d07.20ec1",
        "name": "ShowNode4Temp",
        "func": "msg.payload=msg.payload.Node4Temp;\nif(msg.payload >
-2000 && msg.payload.Node4Temp !== \"ERROR\")\n{\n    msg.payload =
```

```
Math.round(msg.payload * 10)/10;\n    return msg;\n}\nelse return
msg;\nreturn;",
        "outputs": 1,
        "noerr": 0,
        "x": 648.3832397460938,
        "y": 447.88330078125,
        "wires": [
            [
                "5f6530bf.0b30b",
                "4594e0be.f2fa38"
            ]
        ]
    },
    {
        "id": "35f0d8a.30564a8",
        "type": "function",
        "z": "86f75d07.20ec1",
        "name": "ShowNode3Temp",
        "func": "msg.payload=msg.payload.Node3Temp;\nif(msg.payload >
-2000 && msg.payload.Node3Temp !== \"ERROR\")\n{\n    msg.payload =
Math.round(msg.payload * 10)/10;\n    return msg;\n}\nelse\n    return
msg;\n\nreturn;",
        "outputs": 1,
        "noerr": 0,
        "x": 647.3832702636719,
        "y": 369.88330078125,
        "wires": [
            [
                "4505dee.c894e2",
                "2433d40c.b18a74"
            ]
        ]
    },
    {
        "id": "659b97d1.9570d",
        "type": "function",
        "z": "86f75d07.20ec1",
        "name": "ShowNode6Temp",
        "func": "msg.payload=msg.payload.Node6Temp;\nif(msg.payload >
-2000 && msg.payload.Node6Temp !== \"ERROR\")\n{\n    msg.payload =
Math.round(msg.payload * 10)/10;\n    return msg;\n}\nelse return
msg;\nreturn;",
        "outputs": 1,
        "noerr": 0,
        "x": 648.3831787109375,
        "y": 599.7666625976562,
        "wires": [
            [
                "92b3e3f9.3b10e8",
                "93d39b8b.0d36d8"
            ]
        ]
    },
    {
```

```
        "id": "f9d1499.e9a8e38",
        "type": "function",
        "z": "86f75d07.20ec1",
        "name": "ShowNode5Temp",
        "func": "msg.payload=msg.payload.Node5Temp;\nif(msg.payload >
-2000 && msg.payload.Node5Temp !== \"ERROR\")\n{\n    msg.payload =
Math.round(msg.payload * 10)/10;\n    return msg;\n}\nelse\n    return
msg;\n\nreturn;",
        "outputs": 1,
        "noerr": 0,
        "x": 647.3832092285156,
        "y": 522.7666625976562,
        "wires": [
            [
                "4fe1fbee.0876b4",
                "49da2e9e.6bccf8"
            ]
        ]
    },
    {
        "id": "b59448db.491668",
        "type": "function",
        "z": "86f75d07.20ec1",
        "name": "ShowNode8Temp",
        "func": "msg.payload=msg.payload.Node8Temp;\nif(msg.payload >
-2000 && msg.payload.Node8Temp !== \"ERROR\")\n{\n    msg.payload =
Math.round(msg.payload * 10)/10;\n    return msg;\n}\nelse return
msg;\nreturn;",
        "outputs": 1,
        "noerr": 0,
        "x": 649.7665100097656,
        "y": 754.6499633789062,
        "wires": [
            [
                "5d2dab06.15610c",
                "db021dc4.a6651"
            ]
        ]
    },
    {
        "id": "5e3d4fc4.111a3",
        "type": "function",
        "z": "86f75d07.20ec1",
        "name": "ShowNode7Temp",
        "func": "msg.payload=msg.payload.Node7Temp;\nif(msg.payload >
-2000 && msg.payload.Node7Temp !== \"ERROR\")\n{\n    msg.payload =
Math.round(msg.payload * 10)/10;\n    return msg;\n}\nelse\n    return
msg;\n\nreturn;",
        "outputs": 1,
        "noerr": 0,
        "x": 646.7665100097656,
        "y": 675.6499633789062,
        "wires": [
            [
```

```
                "52ed92be.4e0d54",
                "6a34533c.d2dadc"
            ]
        ]
    },
    {
        "id": "d0df92e1.92aa78",
        "type": "function",
        "z": "b5b5a973.429a8",
        "name": "Get_Node9Temp",
        "func": "var
res=msg.payload.response.pdata.split(\".\");\n\nparstemp1 =
parseInt(res[19],16);   //ff\nparstemp2 =  parseInt(res[20],16);
//d0\n\ntemp1 = parstemp1;\ntemp2 = parstemp2;\n\ntemp1 &=
0x7F;\ntemp2 &= 0xF0;\n\ntemp = (temp1<<4) + (temp2>>4);\n\ntemp2 =
parstemp2;\ntemp2 &= 0x0F;\n\ntemp2 *= 0.0625;\n\ntemp += temp2;
\n\nif(((parstemp1 & 0x80)>>7) != 1)\n{\n    temp = 2048 - temp;\n
temp = -temp;\n}\nif(temp == -2048)\n{\n    temp =
\"Desconectado\";\n}\n\nmsg.payload={\"Node9Temp\":temp};\nreturn
msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 590,
        "y": 525,
        "wires": [
            [
                "221e5ad2.700cce"
            ]
        ]
    },
    {
        "id": "fafe2333.284ed8",
        "type": "function",
        "z": "86f75d07.20ec1",
        "name": "ShowNode9Temp",
        "func": "msg.payload=msg.payload.Node9Temp;\nif(msg.payload >
-2000 && msg.payload.Node8Temp !== \"ERROR\")\n{\n    msg.payload =
Math.round(msg.payload * 10)/10;\n    return msg;\n}\nelse return
msg;\nreturn;",
        "outputs": 1,
        "noerr": 0,
        "x": 651.3331604003906,
        "y": 832.3333740234375,
        "wires": [
            [
                "3ff3b5d8.b5a8e2",
                "4ef51804.520448"
            ]
        ]
    },
    {
        "id": "fe5b7313.b54f1",
        "type": "ibmiot in",
        "z": "b5b5a973.429a8",
```

```
        "authentication": "boundService",
        "apiKey": "a250971c.5a32e8",
        "inputType": "evt",
        "logicalInterface": "",
        "ruleId": "",
        "deviceId": "",
        "applicationId": "",
        "deviceType": "+",
        "eventType": "+Nodo1",
        "commandType": "",
        "format": "json",
        "name": "IBM IoT",
        "service": "registered",
        "allDevices": "",
        "allApplications": "",
        "allDeviceTypes": true,
        "allLogicalInterfaces": "",
        "allEvents": true,
        "allCommands": "",
        "allFormats": "",
        "qos": 0,
        "x": 311.75,
        "y": 108.75,
        "wires": [
            [
                "5dfbb639.5df268"
            ]
        ]
    },
    {
        "id": "d90d7b55.ae57c",
        "type": "debug",
        "z": "b5b5a973.429a8",
        "name": "",
        "active": false,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "false",
        "x": 735.75,
        "y": 53.75,
        "wires": []
    },
    {
        "id": "5dfbb639.5df268",
        "type": "function",
        "z": "b5b5a973.429a8",
        "name": "IQRFtoJSON",
        "func": "if (msg.payload.response !== undefined) {\nvar
response=msg.payload.response.split(\".\");\nvar
request=msg.payload.request.split(\".\");\n\nvar data={\n    type:
\"\",\n    request: {\n        nadr: \"\",\n        pnum: \"\",\n
pcmd: \"\",\n        hwpid: \"\"\n    },\n    response: {\n
nadr: \"\",\n        pnum: \"\",\n        pcmd: \"\",\n        hwpid:
```

```
\"\",\n        errn: \"\",\n        dpa: \"\",\n        pdata: \"\",\n
},\n    timeout: \"\",\n    result: \"\",\n    time: new
Date()\n}\n\n\ndata.type =
msg.payload.type;\n\ndata.request.nadr=parseInt(request[0], 16) +
(parseInt(request[1],
16)*256);\ndata.request.pnum=parseInt(request[2],
16);\ndata.request.pcmd=parseInt(request[3],
16);\ndata.request.hwpid=parseInt(request[4], 16) +
(parseInt(request[5],
16)*256);\n\ndata.response.nadr=parseInt(response[0], 16) +
(parseInt(response[1],
16)*256);\ndata.response.pnum=parseInt(response[2],
16);\ndata.response.pcmd=parseInt(response[3],
16);\ndata.response.hwpid=parseInt(response[4], 16) +
(parseInt(response[5],
16)*256);\ndata.response.errn=parseInt(response[6],
16);\ndata.response.dpa=parseInt(response[7],
16);\ndata.response.pdata=msg.payload.response.slice(24).trim();\ndata
.timeout=msg.payload.timeout;\ndata.result=msg.payload.result;\nmsg.pa
yload=data;\nreturn msg;\n}",
        "outputs": 1,
        "noerr": 0,
        "x": 479.75,
        "y": 108.75,
        "wires": [
            [
                "9e9ad9b9.3c1148",
                "d90d7b55.ae57c"
            ]
        ]
    },
    {
        "id": "9e9ad9b9.3c1148",
        "type": "link out",
        "z": "b5b5a973.429a8",
        "name": "",
        "links": [
            "82f58b8a.7aa028",
            "5c6f2af8.0ad9e4",
            "513deff9.0a9748",
            "e1284715.74b12"
        ],
        "x": 623.75,
        "y": 108.75,
        "wires": []
    },
    {
        "id": "583b39db.99f3a",
        "type": "cloudant out",
        "z": "86f75d07.20ec1",
        "name": "",
        "cloudant": "",
        "database": "temperaturas",
        "service": "IQRF-ITACA-cloudantNoSQLDB",
```

```
        "payonly": true,
        "operation": "insert",
        "x": 702.36669921875,
        "y": 900,
        "wires": []
    },
    {
        "id": "81c04fd3.b5ab2",
        "type": "join",
        "z": "86f75d07.20ec1",
        "name": "",
        "mode": "custom",
        "build": "merged",
        "property": "payload",
        "propertyType": "msg",
        "key": "topic",
        "joiner": ",",
        "joinerType": "str",
        "accumulate": false,
        "timeout": "",
        "count": "9",
        "reduceRight": false,
        "reduceExp": "",
        "reduceInit": "",
        "reduceInitType": "num",
        "reduceFixup": "",
        "x": 533.36669921875,
        "y": 904.25,
        "wires": [
            [
                "583b39db.99f3a"
            ]
        ]
    },
    {
        "id": "22bbf70f.fe3f78",
        "type": "ui_gauge",
        "z": "86f75d07.20ec1",
        "name": "Temperature_node2",
        "group": "36e30f3a.bcd328",
        "order": 1,
        "width": "6",
        "height": "5",
        "gtype": "gage",
        "title": "",
        "label": "°C",
        "format": "{{value}}",
        "min": 0,
        "max": "100",
        "colors": [
            "#00b500",
            "#e6e600",
            "#ca3838"
        ],
```

```
        "seg1": "",
        "seg2": "",
        "x": 867.9999694824219,
        "y": 276,
        "wires": []
    },
    {
        "id": "9ad8a506.cc44e",
        "type": "ui_gauge",
        "z": "86f75d07.20ec1",
        "name": "Temperature_node1",
        "group": "9b292989.f5d9b",
        "order": 1,
        "width": "6",
        "height": "5",
        "gtype": "gage",
        "title": "",
        "label": "°C",
        "format": "{{value}}",
        "min": 0,
        "max": "100",
        "colors": [
            "#00b500",
            "#e6e600",
            "#ca3838"
        ],
        "seg1": "",
        "seg2": "",
        "x": 868.9999694824219,
        "y": 200,
        "wires": []
    },
    {
        "id": "4594e0be.f2fa38",
        "type": "ui_gauge",
        "z": "86f75d07.20ec1",
        "name": "Temperature_node4",
        "group": "b53bdc19.c4b6e8",
        "order": 1,
        "width": "6",
        "height": "5",
        "gtype": "gage",
        "title": "",
        "label": "°C",
        "format": "{{value}}",
        "min": 0,
        "max": "100",
        "colors": [
            "#00b500",
            "#e6e600",
            "#ca3838"
        ],
        "seg1": "",
        "seg2": "",
```

```
        "x": 868.3832397460938,
        "y": 427.88330078125,
        "wires": []
    },
    {
        "id": "4505dee.c894e2",
        "type": "ui_gauge",
        "z": "86f75d07.20ec1",
        "name": "Temperature_node3",
        "group": "e7241457.f61438",
        "order": 1,
        "width": "6",
        "height": "5",
        "gtype": "gage",
        "title": "",
        "label": "°C",
        "format": "{{value}}",
        "min": 0,
        "max": "100",
        "colors": [
            "#00b500",
            "#e6e600",
            "#ca3838"
        ],
        "seg1": "",
        "seg2": "",
        "x": 868.3832702636719,
        "y": 350.88330078125,
        "wires": []
    },
    {
        "id": "93d39b8b.0d36d8",
        "type": "ui_gauge",
        "z": "86f75d07.20ec1",
        "name": "Temperature_node6",
        "group": "a6a0def4.5548a",
        "order": 1,
        "width": "6",
        "height": "5",
        "gtype": "gage",
        "title": "",
        "label": "°C",
        "format": "{{value}}",
        "min": 0,
        "max": "100",
        "colors": [
            "#00b500",
            "#e6e600",
            "#ca3838"
        ],
        "seg1": "",
        "seg2": "",
        "x": 868.3831787109375,
        "y": 579.7666625976562,
```

```
        "wires": []
    },
    {
        "id": "4fe1fbee.0876b4",
        "type": "ui_gauge",
        "z": "86f75d07.20ec1",
        "name": "Temperature_node5",
        "group": "e2871101.d4dbc8",
        "order": 1,
        "width": "6",
        "height": "5",
        "gtype": "gage",
        "title": "",
        "label": "°C",
        "format": "{{value}}",
        "min": 0,
        "max": "100",
        "colors": [
            "#00b500",
            "#e6e600",
            "#ca3838"
        ],
        "seg1": "",
        "seg2": "",
        "x": 868.3832092285156,
        "y": 503.76666259765625,
        "wires": []
    },
    {
        "id": "db021dc4.a6651",
        "type": "ui_gauge",
        "z": "86f75d07.20ec1",
        "name": "Temperature_node8",
        "group": "95e99112.4af158",
        "order": 1,
        "width": "6",
        "height": "5",
        "gtype": "gage",
        "title": "",
        "label": "°C",
        "format": "{{value}}",
        "min": 0,
        "max": "100",
        "colors": [
            "#00b500",
            "#e6e600",
            "#ca3838"
        ],
        "seg1": "",
        "seg2": "",
        "x": 869.7665100097656,
        "y": 734.6499633789062,
        "wires": []
    },
```

```
{
    "id": "52ed92be.4e0d54",
    "type": "ui_gauge",
    "z": "86f75d07.20ec1",
    "name": "Temperature_node7",
    "group": "f4a086d8.64361",
    "order": 1,
    "width": "6",
    "height": "5",
    "gtype": "gage",
    "title": "",
    "label": "°C",
    "format": "{{value}}",
    "min": 0,
    "max": "100",
    "colors": [
        "#00b500",
        "#e6e600",
        "#ca3838"
    ],
    "seg1": "",
    "seg2": "",
    "x": 867.7665100097656,
    "y": 656.6499633789062,
    "wires": []
},
{
    "id": "4ef51804.520448",
    "type": "ui_gauge",
    "z": "86f75d07.20ec1",
    "name": "Temperature_node9",
    "group": "5ddee758.2fa97",
    "order": 1,
    "width": "6",
    "height": "5",
    "gtype": "gage",
    "title": "",
    "label": "°C",
    "format": "{{value}}",
    "min": 0,
    "max": "100",
    "colors": [
        "#00b500",
        "#e6e600",
        "#ca3838"
    ],
    "seg1": "",
    "seg2": "",
    "x": 871.3331604003906,
    "y": 812.3333740234375,
    "wires": []
},
{
    "id": "d640a5.3ba80758",
```

```
        "type": "ui_chart",
        "z": "86f75d07.20ec1",
        "name": "Temperature_node2",
        "group": "36e30f3a.bcd328",
        "order": 2,
        "width": 0,
        "height": 0,
        "label": "",
        "chartType": "line",
        "legend": "false",
        "xformat": "HH:mm:ss",
        "interpolate": "linear",
        "nodata": "Waiting for data",
        "dot": false,
        "ymin": "0",
        "ymax": "100",
        "removeOlder": "5",
        "removeOlderPoints": "",
        "removeOlderUnit": "60",
        "cutout": 0,
        "useOneColor": false,
        "colors": [
            "#1f77b4",
            "#aec7e8",
            "#ff7f0e",
            "#2ca02c",
            "#98df8a",
            "#d62728",
            "#ff9896",
            "#9467bd",
            "#c5b0d5"
        ],
        "useOldStyle": true,
        "outputs": 2,
        "x": 867.9999694824219,
        "y": 316,
        "wires": [
            [],
            []
        ]
    },
    {
        "id": "e2e3bd26.49064",
        "type": "ui_chart",
        "z": "86f75d07.20ec1",
        "name": "Temperature_node1",
        "group": "9b292989.f5d9b",
        "order": 2,
        "width": 0,
        "height": 0,
        "label": "",
        "chartType": "line",
        "legend": "false",
        "xformat": "HH:mm:ss",
```

```
        "interpolate": "linear",
        "nodata": "Waiting for data",
        "dot": false,
        "ymin": "0",
        "ymax": "100",
        "removeOlder": "5",
        "removeOlderPoints": "",
        "removeOlderUnit": "60",
        "cutout": 0,
        "useOneColor": false,
        "colors": [
            "#1f77b4",
            "#aec7e8",
            "#ff7f0e",
            "#2ca02c",
            "#98df8a",
            "#d62728",
            "#ff9896",
            "#9467bd",
            "#c5b0d5"
        ],
        "useOldStyle": true,
        "outputs": 2,
        "x": 868.9999694824219,
        "y": 240,
        "wires": [
            [],
            []
        ]
    },
    {
        "id": "5f6530bf.0b30b",
        "type": "ui_chart",
        "z": "86f75d07.20ec1",
        "name": "Temperature_node4",
        "group": "b53bdc19.c4b6e8",
        "order": 2,
        "width": 0,
        "height": 0,
        "label": "",
        "chartType": "line",
        "legend": "false",
        "xformat": "HH:mm:ss",
        "interpolate": "linear",
        "nodata": "Waiting for data",
        "dot": false,
        "ymin": "0",
        "ymax": "100",
        "removeOlder": "5",
        "removeOlderPoints": "",
        "removeOlderUnit": "60",
        "cutout": 0,
        "useOneColor": false,
        "colors": [
```

```
            "#1f77b4",
            "#aec7e8",
            "#ff7f0e",
            "#2ca02c",
            "#98df8a",
            "#d62728",
            "#ff9896",
            "#9467bd",
            "#c5b0d5"
        ],
        "useOldStyle": true,
        "outputs": 2,
        "x": 868.3832397460938,
        "y": 467.88330078125,
        "wires": [
            [],
            []
        ]
    },
    {
        "id": "2433d40c.b18a74",
        "type": "ui_chart",
        "z": "86f75d07.20ec1",
        "name": "Temperature_node3",
        "group": "e7241457.f61438",
        "order": 2,
        "width": 0,
        "height": 0,
        "label": "",
        "chartType": "line",
        "legend": "false",
        "xformat": "HH:mm:ss",
        "interpolate": "linear",
        "nodata": "Waiting for data",
        "dot": false,
        "ymin": "0",
        "ymax": "100",
        "removeOlder": "5",
        "removeOlderPoints": "",
        "removeOlderUnit": "60",
        "cutout": 0,
        "useOneColor": false,
        "colors": [
            "#1f77b4",
            "#aec7e8",
            "#ff7f0e",
            "#2ca02c",
            "#98df8a",
            "#d62728",
            "#ff9896",
            "#9467bd",
            "#c5b0d5"
        ],
        "useOldStyle": true,
```

```
        "outputs": 2,
        "x": 868.3832702636719,
        "y": 387.88330078125,
        "wires": [
            [],
            []
        ]
    },
    {
        "id": "92b3e3f9.3b10e8",
        "type": "ui_chart",
        "z": "86f75d07.20ec1",
        "name": "Temperature_node6",
        "group": "a6a0def4.5548a",
        "order": 2,
        "width": 0,
        "height": 0,
        "label": "",
        "chartType": "line",
        "legend": "false",
        "xformat": "HH:mm:ss",
        "interpolate": "linear",
        "nodata": "Waiting for data",
        "dot": false,
        "ymin": "0",
        "ymax": "100",
        "removeOlder": "5",
        "removeOlderPoints": "",
        "removeOlderUnit": "60",
        "cutout": 0,
        "useOneColor": false,
        "colors": [
            "#1f77b4",
            "#aec7e8",
            "#ff7f0e",
            "#2ca02c",
            "#98df8a",
            "#d62728",
            "#ff9896",
            "#9467bd",
            "#c5b0d5"
        ],
        "useOldStyle": true,
        "outputs": 2,
        "x": 868.3831787109375,
        "y": 619.7666625976562,
        "wires": [
            [],
            []
        ]
    },
    {
        "id": "49da2e9e.6bccf8",
        "type": "ui_chart",
```

```
        "z": "86f75d07.20ec1",
        "name": "Temperature_node5",
        "group": "e2871101.d4dbc8",
        "order": 2,
        "width": 0,
        "height": 0,
        "label": "",
        "chartType": "line",
        "legend": "false",
        "xformat": "HH:mm:ss",
        "interpolate": "linear",
        "nodata": "Waiting for data",
        "dot": false,
        "ymin": "0",
        "ymax": "100",
        "removeOlder": "5",
        "removeOlderPoints": "",
        "removeOlderUnit": "60",
        "cutout": 0,
        "useOneColor": false,
        "colors": [
            "#1f77b4",
            "#aec7e8",
            "#ff7f0e",
            "#2ca02c",
            "#98df8a",
            "#d62728",
            "#ff9896",
            "#9467bd",
            "#c5b0d5"
        ],
        "useOldStyle": true,
        "outputs": 2,
        "x": 868.3832092285156,
        "y": 542.7666625976562,
        "wires": [
            [],
            []
        ]
    },
    {
        "id": "5d2dab06.15610c",
        "type": "ui_chart",
        "z": "86f75d07.20ec1",
        "name": "Temperature_node8",
        "group": "95e99112.4af158",
        "order": 2,
        "width": 0,
        "height": 0,
        "label": "",
        "chartType": "line",
        "legend": "false",
        "xformat": "HH:mm:ss",
        "interpolate": "linear",
```

```
        "nodata": "Waiting for data",
        "dot": false,
        "ymin": "0",
        "ymax": "100",
        "removeOlder": "5",
        "removeOlderPoints": "",
        "removeOlderUnit": "60",
        "cutout": 0,
        "useOneColor": false,
        "colors": [
            "#1f77b4",
            "#aec7e8",
            "#ff7f0e",
            "#2ca02c",
            "#98df8a",
            "#d62728",
            "#ff9896",
            "#9467bd",
            "#c5b0d5"
        ],
        "useOldStyle": true,
        "outputs": 2,
        "x": 869.7665100097656,
        "y": 774.6499633789062,
        "wires": [
            [],
            []
        ]
    },
    {
        "id": "6a34533c.d2dadc",
        "type": "ui_chart",
        "z": "86f75d07.20ec1",
        "name": "Temperature_node7",
        "group": "f4a086d8.64361",
        "order": 2,
        "width": 0,
        "height": 0,
        "label": "",
        "chartType": "line",
        "legend": "false",
        "xformat": "HH:mm:ss",
        "interpolate": "linear",
        "nodata": "Waiting for data",
        "dot": false,
        "ymin": "0",
        "ymax": "100",
        "removeOlder": "5",
        "removeOlderPoints": "",
        "removeOlderUnit": "60",
        "cutout": 0,
        "useOneColor": false,
        "colors": [
            "#1f77b4",
```

```
            "#aec7e8",
            "#ff7f0e",
            "#2ca02c",
            "#98df8a",
            "#d62728",
            "#ff9896",
            "#9467bd",
            "#c5b0d5"
        ],
        "useOldStyle": true,
        "outputs": 2,
        "x": 868.7665100097656,
        "y": 696.6499633789062,
        "wires": [
            [],
            []
        ]
    },
    {
        "id": "3ff3b5d8.b5a8e2",
        "type": "ui_chart",
        "z": "86f75d07.20ec1",
        "name": "Temperature_node9",
        "group": "5ddee758.2fa97",
        "order": 2,
        "width": 0,
        "height": 0,
        "label": "",
        "chartType": "line",
        "legend": "false",
        "xformat": "HH:mm:ss",
        "interpolate": "linear",
        "nodata": "Waiting for data",
        "dot": false,
        "ymin": "0",
        "ymax": "100",
        "removeOlder": "5",
        "removeOlderPoints": "",
        "removeOlderUnit": "60",
        "cutout": 0,
        "useOneColor": false,
        "colors": [
            "#1f77b4",
            "#aec7e8",
            "#ff7f0e",
            "#2ca02c",
            "#98df8a",
            "#d62728",
            "#ff9896",
            "#9467bd",
            "#c5b0d5"
        ],
        "useOldStyle": true,
        "outputs": 2,
```

```
        "x": 871.3331604003906,
        "y": 852.3333740234375,
        "wires": [
            [],
            []
        ]
    }
]
```