Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Enginyeria Electrònica Industrial i Automàtica

Títol: Sistema de gestió d'aparcament basat amb el sistema LoRa

Document: 1. Memòria

Alumne: Ricard Casas i del Olmo

Tutor: Dr. Carles Pous Sabadi
Departament: Enginyeria Elèctrica, Electrònica i Automàtica
Àrea: Enginyeria de Sistemes i Automàtica

Convocatòria (mes/any): juny/2022

1. INTRODUCCIÓ

El projecte Sistema de gestió d'aparcaments basat amb LoRa tracta els aspectes i la informació necessària per dur a terme la correcte implementació i gestió als següents apartats.

1.1. Antecedents

La digitalització és cada cop més freqüent a tots els àmbits, des del privat al públic. Els sistemes IoT representen una característica fonamental d'aquesta digitalització, per això les ciutats veuen una gran oportunitat de millora i evolució. La gestió d'aparcaments n'és un exemple, millorar la mobilitat dins la ciutat o implementar estratègies de dinamització que ho motiven. Per aquest motiu l'Ajuntament de Girona ha estat desplegant i implementant diversos sistemes que permeten tenir un control relatiu dels aparcaments.

L'experiència extreta dels projectes precedents, ha fet que l'Ajuntament de Girona, a través de la Càtedra Girona Smart City, faci una proposta de gestió d'aparcaments temporals a través de la seva xarxa LoRa que motiva aquest TFG.

1.2. Objecte

L'objectiu és dissenyar un prototip que permeti detectar quan una plaça d'aparcament està ocupada, enviar la informació a través de la xarxa LoRa, portar un còmput del temps d'ocupació i generar alarmes en cas de sobrepassar un temps determinat que indiquin la retirada del vehicle.

L'alarma activarà un senyal lluminós instal·lat a la zona d'aparcament gràcies al fet que tot el sistema actua de manera autònoma a través del sistema de comunicació LoRa i aplicació corresponent.

1.3. Especificacions i abast

L'abast del projecte consta de tres fases, les quals es resumeixen en: Ampliació de la xarxa LoRa, desenvolupament i industrialització.

La primera fase s'inicia amb la dotació de l'equip i programari necessari per ampliar la cobertura de la xarxa LoRa al campus Montilivi. Per aquesta raó es busca el lloc i la manera més idònia d'instal·lar la "Gateway", especialitzada en comunicacions LoRa del fabricant MultiTech. La correcta integració amb la xarxa Ethernet i alimentació a través de PoE (Power over Ethernet) de la UdG és clau així com el correcte posicionament de l'antena per maximitzar la transmissió de les dades.

Durant la segona fase es realitza el prototip, es selecciona/dissenya el sensor necessari per a la detecció del vehicle i l'indicador lluminós segons els requisits. En el cas del sensor d'aparcament es selecciona un model de la marca IOTSens i l'indicador lluminós es dissenya un prototip a partir del microprocessador ESP32. Els requisits demanats per l'ajuntament de Girona són:

El sensor d'aparcament ha de ser totalment autònom (bateries) i comunicar la informació a través de les comunicacions LoRa.

L'actuador lumínic ha de ser capaç d'enviar i rebre dades a través de LoRa, realitzar estudis que prioritzin l'alimentació autònoma amb el màxim temps de vida i en cas de no tenir viabilitat, presentar alternatives.

Per últim tot el sistema ha de ser capaç de respondre en menys d'un minut des de que el sistema emet una ordre de retirada del vehicle.

Un cop resolt aquests requisits s'ha d'implementar la lògica, la qual es resumeix amb la detecció del vehicle, revisió del temps d'estacionament i activació de l'indicador lluminós. Per

dur a terme tal processament és imprescindible la creació i correcta gestió d'un servidor que albergui tots els elements indispensables, base de dades, aplicació, etc.

Finalment, es contempla futures implementacions, des de la posició del detector del vehicle i l'indicador lluminós, l'abast de les comunicacions, vida útil d'aquests i encapsulat pel seu ús intensiu i possible industrialització.

2. ARQUITECTURA I TECNOLOGIA

Aquest capítol i els seus apartats defineixen i identifiquen totes les comunicacions i protocols utilitzades pel sistema. La següent figura mostra l'estructura implementada de manera resumida i dóna una primera impressió del model final implementat.

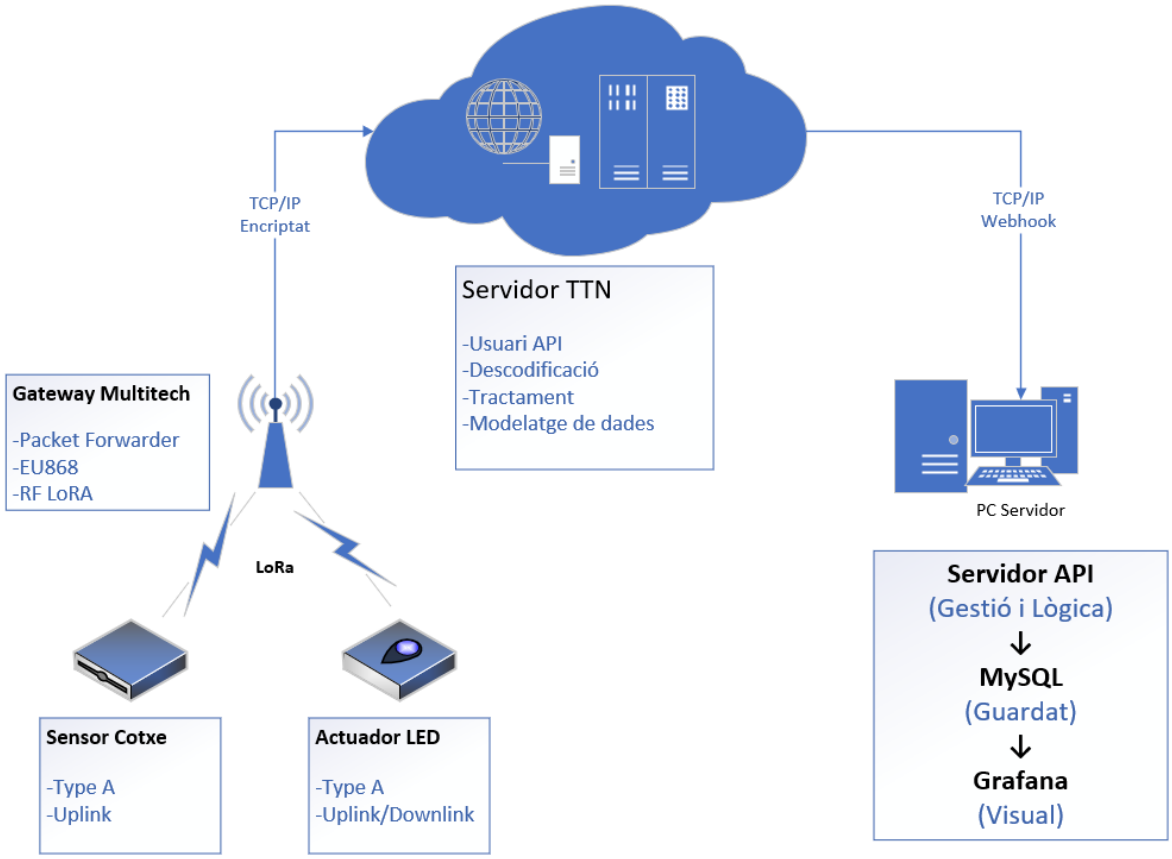

Figura 1 Arquitectura de xarxa

El sistema està constituït per diferents elements. A l'edge ens trobem el sensor de parking i el led indicador de si el vehicle aparcat està complint amb la franja horària. Seguidament tenim la gateway LoRa, que fa de passarel·la entre els dispositius sensor i actuador LED i el servidor TTN V3 (The Things Network), convertint la informació de protocol LoRa a TCP/IP, i finalment el servidor de l'usuari, on residiran les aplicacions que han de permetre a l'usuari configurar la xarxa, així com poder visualitzar i guardar les dades.

La lògica implementada segueix el següent transcurs: El sensor d'aparcament (o cotxe) envia el senyal per radiofreqüència LoRa la qual és rebuda per la Gateway Multitech. Aquesta està connectada a internet mitjançant un cable Ethernet, i redirigeix la informació als servidors The Things Network V3 (Stack Comunity) on es contrasta i modela. Un cop realitzat aquest pas es remet via Webhook al servidor JavaScript habilitat a la universitat.

En aquest servidor es tracten les dades i emmagatzematge a una base de dades MySQL a través del servidor WampServer. Es retorna la resposta en cas de sobrepassar el temps establert del parking, la qual ressegueix tota l'arquitectura de manera inversa fins a arribar a l'actuador lumínic (o LED) el qual, és l'únic programat per rebre i interpretar missatges a través de LoRa (Downlinks).

Als següents apartats s'expliquen els aspectes i característiques principals d'aquestes comunicacions utilitzades de manera més detallada així com els avantatges, inconvenients i els motiu final del seu ús.

2.1. Capa física LoRa

LoRa i LoRaWAN junts es defineixen com un protocol en xarxa de comunicacions de baix consum i àrea amplia o LPWA (Low Power Wide Area) dissenyat per dispositius sense fils que funcionin de manera autònoma, connectats a bateries i tenen una tassa baixa d'enviament de dades.Enfocada al món IoT constitueix avui dia una de les principals comunicacions utilitzades per sensors i actuadors dins el món Smart City. LoRa defineix el protocol a nivell de capa física, mentre que LoRaWAN ho fa a nivell de capa MAC.

El hardware LoRa en si, és una tècnica de modulació de radio que deriva de la tecnologia CSS (Chirp Spread Spectrum). Consisteix en un senyal d'amplitud constant que va variant la seva freqüència contínuament de forma lineal en el temps com es veu a la Figura 1.

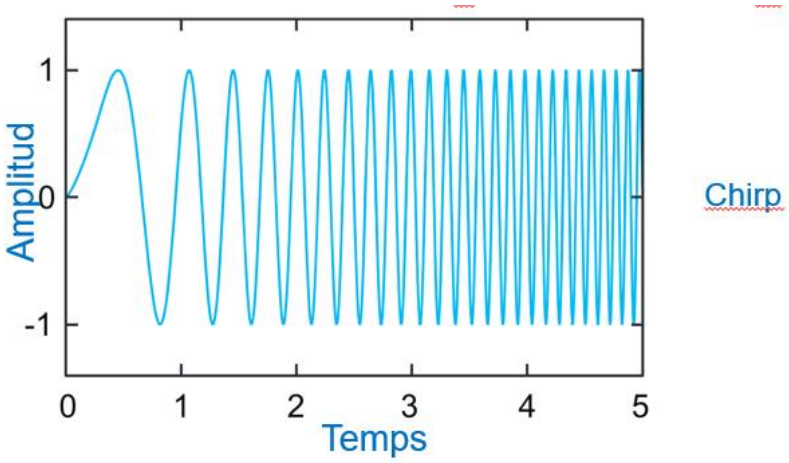

Figura 2 Amplitud CSS

Un rang de freqüències establert, freqüència baixa fL baixa i alta fH, estableix els límits de la variació, la següent figura mostra un gràfic de com es solen representar.

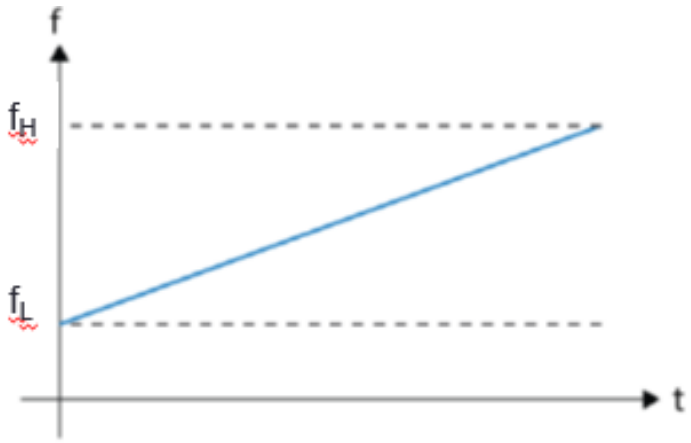

Figura 3 Rang freqüencial

A la realitat, però aquest escombrat lineal es realitza fent salts de freqüència discrets. El número de salts a fer dins de la durada de l'escombrat ve determinat pel paràmetre Spreading Factor (SF). Per exemple, si es té que SF = 7, vol dir que farà 2^7 = 128 salts de freqüència, una mostra g`rafica d'aquest comportament es mostra a la figura següent.

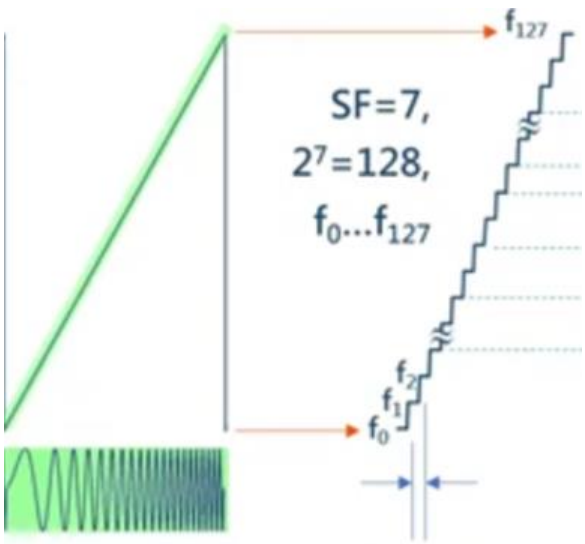

Figura 4 Mostreig SF7

Depenent de la freqüència a la que comenci a fer l'escombrat, representa un símbol diferent. Aquest símbol codifica tants bits com el SF que s'hagi seleccionat. En el cas anterior, cada símbol estaria representant un codi de 7 bits. La Figura 5 mostra un exemple teòric de les ones retallades.

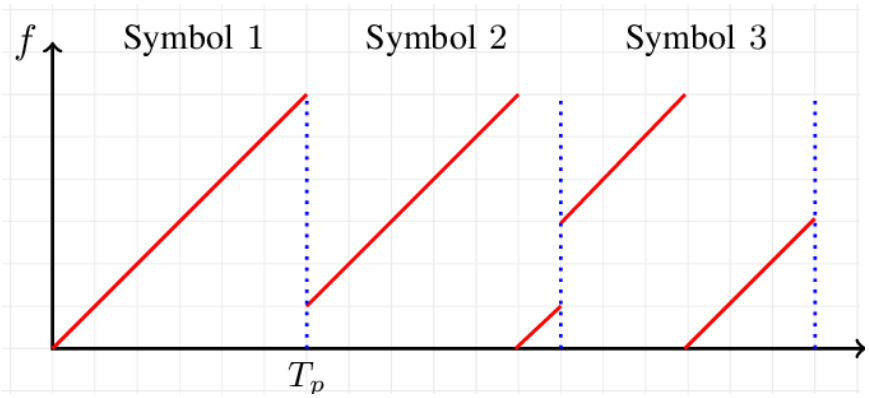

Figura 5 LoRa CSS

A nivell d'exemple i mantenint el cas de SF=7, si el símbol a transmetre ha començat per la freqüència número 67, f_{67}, vol dir que s'està transmetén el codi de 7 bits 100 0011.

Els resultats de la utilització d'aquesta modulació permeten que la senyal sigui molt robusta davant possibles distorsions, ja siguin per soroll ambiental o altres comunicacions LoRa sobreposades.

LoRa opera dins la banda ISM (Industrial, Scientific, Medial) una freqüència per sota dels GHz com per exemple 868 MHz Europa, 915 MHz USA o 433 MHz China. D'aquesta manera no requereix de llicència i pot ser usada de manera lliure per qualsevol usuari. Per altra banda pot ser operada dins els 2,4 GHz per millorar la velocitat de transmissió de dades a canvi d'una reducció en el rang.

La solució de compromís adoptada per aquesta tecnologia es resumeix en mantenir el consum al mínim i com a conseqüència l'ampla de banda disminueix, alhora que garanteix un major rang de transmissió. La següent figura il·lustra l'explicació donada comparant-la amb altres comunicacions.

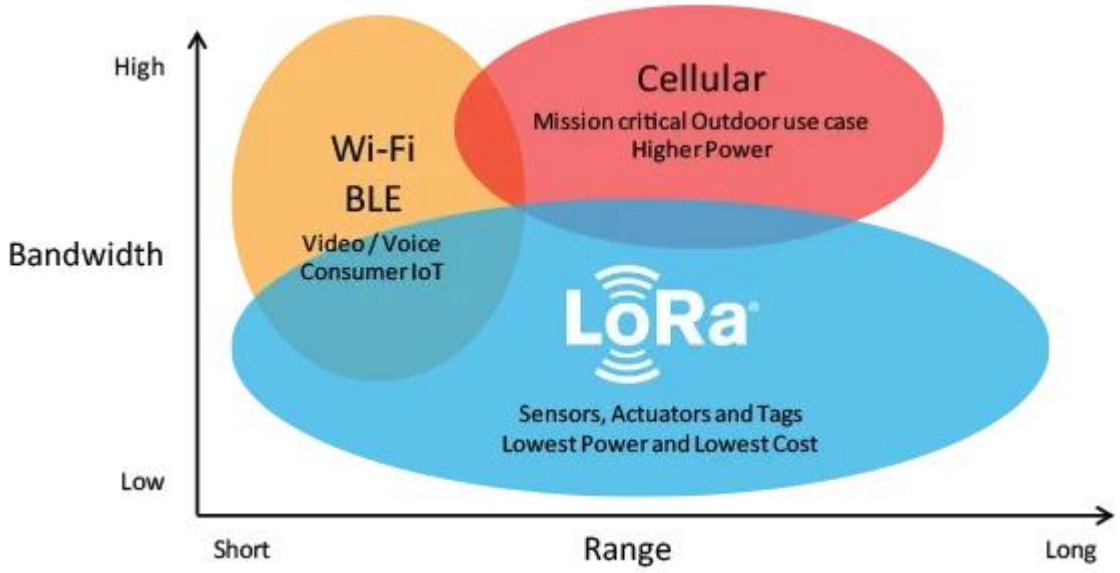

Figura 6 Retransmissió LoRa

Finalment, comentar breument que la tecnologia LoRa és exclusiva del fabricant Semtech el qual proveeix l'equipament (Hardware) necessari en tots els productes LoRa o aparells que

vulguin ser adaptats per aquestes comunicacions. Semtech és esmentat al llarg d'aquest projecte.

Models de transmissió similars com Sigfox i Xbee que enfoquen la seva tecnologia en aquests àmbit s'han descartat ja que són de pagament o no complicarien extraordinàriament l'instal·lació.

2.2. Protocol LoRaWAN

LoRaWAN és el protocol que implementa la capa Media Access Control (MAC). El programari defineix com s'utilitza LoRa, quan retransmetre, de quina manera, etc. Pertany a les capes 2 i 3 del model OSI, més concretament a la capa d'enllaç i xarxa respectivament. La Figura 7 mostra la totalitat de les capes i sentit.

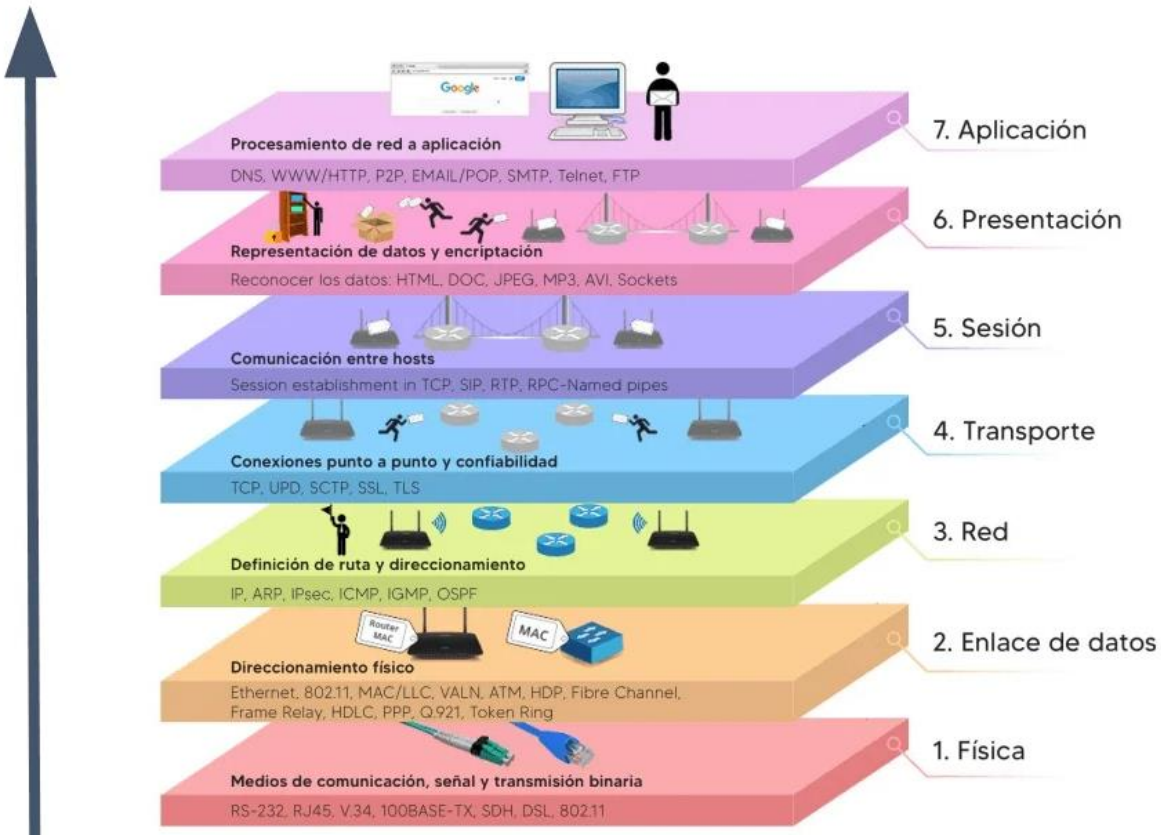

Figura 7 Model OSI

Actualment, LoRaWAN ha esdevingut un protocol estandarditzat gràcies a LoRa Alliance, una associació sense ànim de lucre promogut per les mateixes empreses involucrades des de les desenvolupadores fins les consumidores interessades.

2.2.1. Arquitectura

LoRaWAN es tracta d'una xarxa de topologia en estrella que es fonamenta en cinc elements principals: End Devices, Gateways, Network Server, Application Server i Join Server.

A continuació és fa una descripció de cadascun dels elements definits anteriorment i per donar una descripció més gràfica s'afegeix la Figura 8.

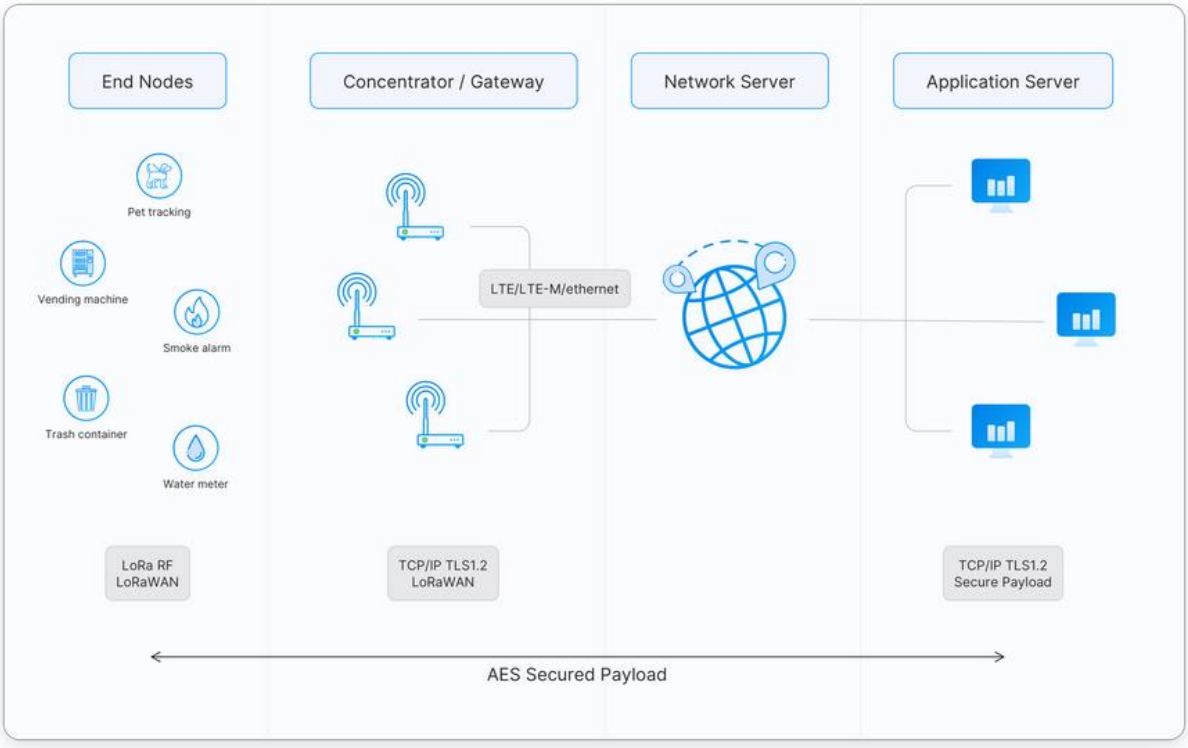

Figura 8 Arquitectura LoRaWAN

End Devices (Nodes), es refereix als sensors i actuadors que envien o reben missatges a través de la modulació LoRa des de les Gateways.

Gateways, dispositius encarregats de rebre els missatges dels Nodes i de encaminar aquesta informació al Network Server o a l'invers, interpretant les ordres del Network Server i redirigir-les al End Node corresponent.

A continuació es descriuen els programaris de la companyia TTI, el Network Server implementa el protocol LoRaWAN i gestiona la totalitat de la xarxa.Estableix la connexió segura AES (Advanced Encryption System) de 128 bits per transportar la informació entre els sensors i la Application Server. Valida la autenticitat dels sensors i garanteix la integritat dels missatges. Processa els missatges a través d'algoritmes que rebutgen els repetits, adreça aquest missatge a l'aplicatiu, la optimització, etc.

Application Server, és l'aplicació responsable de processar i gestionar de manera segura les dades i on es crea la xarxa de l'usuari. Genera totes les capes necessàries per enviar missatges direcció als actuadors o sensors. Pot interpretar i fer una primera gestió de la informació.

The Things Industries (TTI), és una empresa que ha implementat el protocol LoRaWAN oferint-lo als usuaris, tant en mode públic, gratuït, com en mode privat, de pagament. A la pila d'elements que constitueixen el network server l'anomenen The Things Stack (TTS). Dins de TTS, la versió gratuïta s'anomena The Things Stack Community Edition (anteriorment TTN) .

En el cas de TTS, el network server de la figura anterior el constitueixen diferents components. Aquests asseguren la connexió i pas d'informació de la manera correcta i amb la seguretat adequada en cada pas. Es tracten del Gateway Server, Join Server, Identity Server i Monitoring. La següent figura mostra amb més detall l'arquitectura. Cal esmentar que alguna de les integracions que s'hi mostren poden estar desactualitzades.

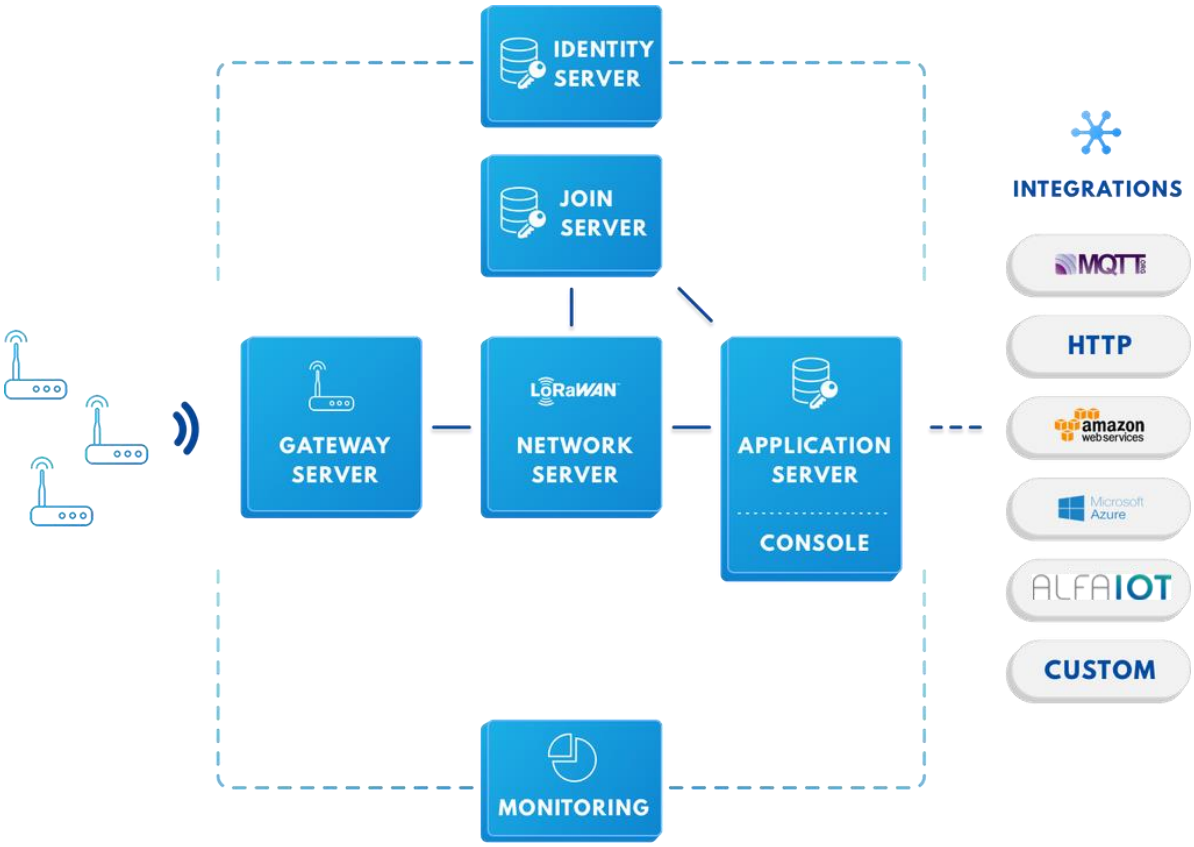

Figura 9 Servidor LoRa

Gateway Server, aglomera la totalitat de les gateways registrades a l'aplicatiu TTN (The Things Network) V3 i processa la informació d'aquestes en primer pla, categoritzant el temps de rebuda, la qualitat, etc. Tot seguit reenvia aquesta informació (missatges Uplink) direcció Network Server o viceversa, rep la informació del Network Server i la redirigeix a la Gateway corresponent. Gestiona les capes comunicatives necessàries per UDP, TCP, protocol MQTT, etc.

Join Server, es l'encarregat de processar les peticions dels missatges enviats pels Nodes (accedir a l'API de l'usuari corresponent) de manera segura. Activació dels sensors dins la xarxa, genera les claus necessàries, en el cas d'activació per OTAA (Over The Air Activation) o validar les de ABP (Activation By Personalization).

Identity Server, és l'encarregat d'atribuir i relacionar les entitats del la xarxa, relaciona l'aplicatiu amb els seus sensors i actuadors, gateways, usuaris, organitzacions, etc. Alhora

gestiona l'accés i control de la propietat (Gateways i aplicacions privades) i claus API (API Keys).

Finalment, Monitoring, l'empresa The Things Industries és l'encarregada de monitoritzar del seu correcte funcionament, manteniment i formalització de millores.

2.2.2. Tipus de sensors LoRaWAN

Els End Devices es classifiquen en tres classes (o tipus) segons el seu ús dins LoRaWAN. Es tracten dels tipus, A, B i C. Cal remarcar, tots els dispositius comparteixen en comú el fet d'haver d'enviar missatges (Uplink) per poder rebre missatges des de la Gateway (Downlink).

Tots els End Devices (Sensors) suporten el Tipus A, ja que és el més bàsic. Constitueixen els sensors de més baix consum i aquells que majoritàriament es dediquen a recaptar dades. Les comunicacions sempre s'activen a partir del Node, envia un missatge (Uplink) i espera un temps per si la Gateway ha d'enviar-li un missatge (Downlink). De fet, el protocol estableix dues finestres de temps d'espera, RX1 i RX2. La Figura 10 mostra el fil de la transmissió.

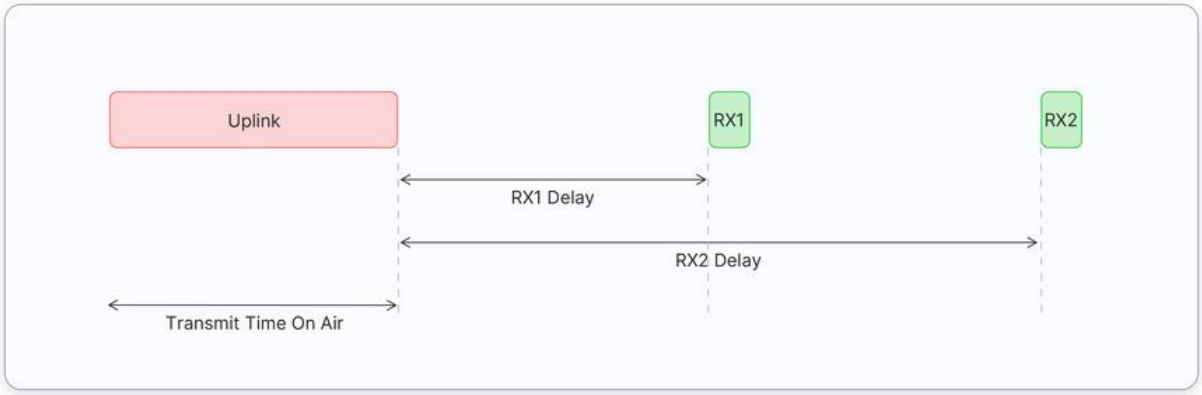

Figura 10 Transmissió Tipus A

Aquest tipus és l'implementat a tots els sensors i actuadors del prototip del present projecte, degut al seu baix consum i facilitat per implementar.

Tipus B, en addició al tipus A, obre finestres de Downlinks de manera programada en el temps anomenades balises (Beacon). Són senyals enviats periòdicament per la Gateway que marca l'inici i final del període de retransmissió i recepció. S'envia un Uplink amb informació del dispositiu, una petita càrrega d'informació extra o no (Payload) i espera a rebre el Downlink. La seva latència és inferior als de tipus A, però en mantenir més temps l'estat de recepció fa augmentar significativament el consum d'energia com es veu a la Figura 11.

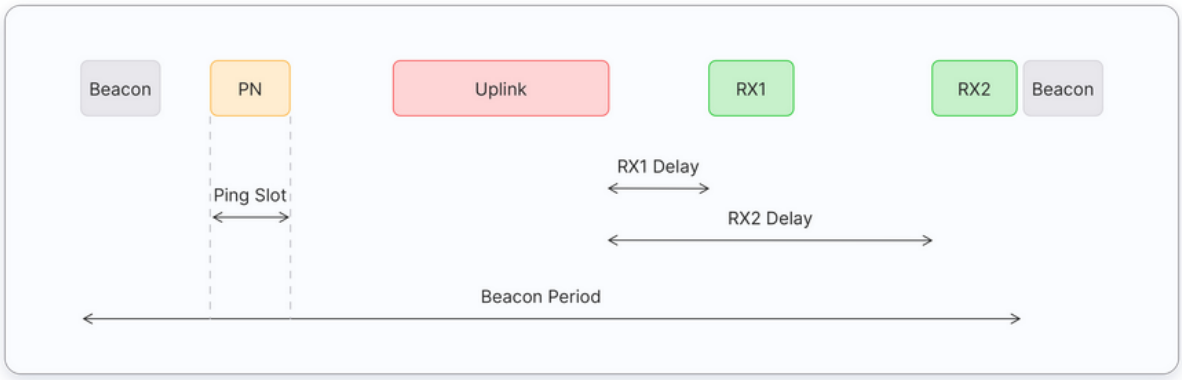

Figura 11 Transmissió Tipus B

Tipus C, és una extensió del Tipus A el qual, es manté a l'espera de Downlinks fins a la recepció d'aquest o l'enviament del següent Uplink. Clarament, no contempla cap mena d'estalvi d'energia i fa que l'alimentació d'aquests nodes sigui per connexió a la xarxa. La següent figura mostra el seu flux de transmissió.

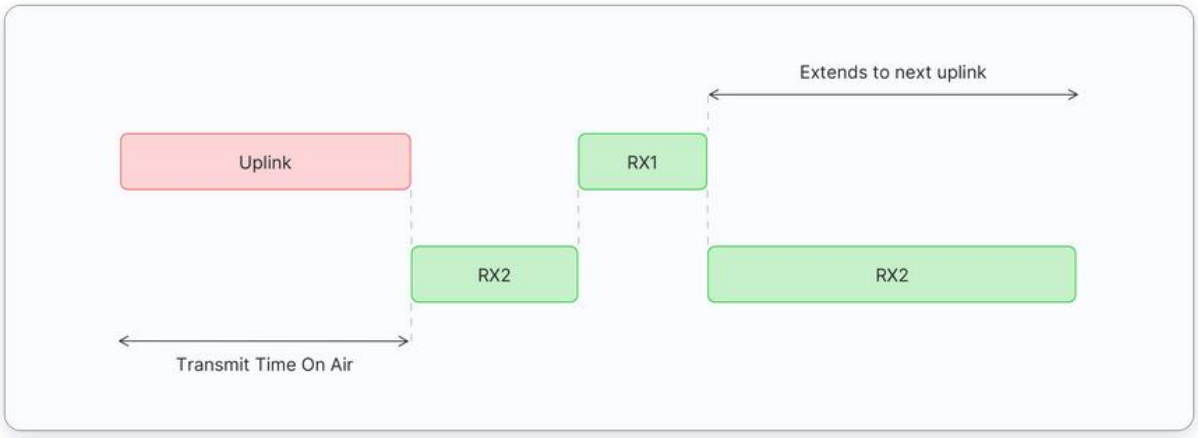

Figura 12 Transmissió Tipus C

Tots els nodes existents queden contemplats en un dels tres tipus esmentats i les possibles variacions i combinacions entre ells, tot i ser possibles, no són recomanables.

2.2.3. Activació sensors OTAA i ABP

Per integrar els sensors amb la xarxa LoRa és necessari l'anomenada activació, la qual contempla aspectes de seguretat i registre dins l'aplicació general (Application Server).

Existeixen dos mètodes d'activació, OTAA (Over The Air Activation) i ABP (Activation By Personalization.Els dos tipus d'activació utilitzen els mateixos paràmetres, però la forma en què són tractats marcarà la diferència. Aquests paràmetres són DevEUI i DevAddr.

DevEUI ID de 64 bits és un codi identificador del sensor únic per cada xarxa LoRa, en el cas de provenir d'un fabricant de dispositius (IOTSens) o es pot generar a través de l'API de TTN V3 quan es tracta d'un prototip, és pública. És fix tant per ABP com per OTAA.

DevAddr, és un codi de 32 bits que identifica a l'end Device dins la xarxa i és l'adreça utilitzada per totes les comunicacions després d'haver fet el join. Està constituit per la unió entre NwkAddr (adreça assignada al sensor dins la xarxa), i un prefix NwkID (identificador de xarxa).

ABP conté una DevAddr i contrasenyes de sessió fixes les quals han estat anteriorment seleccionades a l'aplicació TTN V3, esquivant el protocol de seguretat intern de TTN Join, fet que facilita el seu ús directe (proves, prototips, etc), però les seves funcions són poc segures i les limita.

OTAA, en canvi, conté contrasenyes provisionals seleccionades a l'aplicació TTN V3, les quals durant l'activació (funció Join) són assignades de manera dinàmica juntament amb el DevAddr. Així doncs, tant el DevAddr i contrasenyes canvien cada cop que s'inicien les comunicacions amb la xarxa, el que fa que aquests mètode d'activació sigui més segur comparat amb ABP.

Un cop executada l'activació fa falta mantenir la sessió oberta, d'aquesta manera no es repeteix l'activació cada cop que el dispositiu inicia comunicacions LoRa. Mantenir la sessió oberta significa que LoRaWAN espera rebre unes credencials úniques, que es tracten de les següents:

NwkSKey (Network Session Key), contrasenya usada pel Network Server i poder identificar el dispositiu, així com la integritat del missatge anomenat MIC (Message Integrity Code) , ja que valida els paràmetres d'activació DevEUI i DevAddr i finalment, l'aplicació corresponent, App EUI.

SNwkSIntKey (Serving Network session integrity key), clau de verificació MIC especial pels Downlinks, gestionada únicament pel servidor Join.

NwKSEncKey (Network session encryption key), serveix per encriptar i desencriptar els missatges Uplink i Downlink i relacionar-los amb el port de connexió amb la xarxa. També es compara amb la MAC del dispositiu. Gestionada únicament pel servidor Join

AppSKey (Application Session Key), serveix per encriptar i desencriptar les comunicacions del Payload.

Tots els paràmetres esmentats són únics i específics per cada dispositiu registrat a l'aplicació creada de LoRa. L'activació per OTAA s'encarrega automàticament de totes les claus, mentre que l'ABP requereix la creació i correcta gestió de les NwkSKey i AppSKey.

El projecte es basa en el sistema d'activació OTAA a causa de les següents raons:

La utilització de DevAddr fixes per part de l'activació ABP, dificulta la transmissió dels senyals, dins la xarxa LoRaWAN amb l'anomenat Packet Brocker. Aquest s'encarrega de traslladar les dades dels sensors fins al Network Server, alhora de gestionar tal volum de dades entre

clústers les dificulta i resulta en un deteriorament de l'optimització del sistema. OTAA canvia de DevAddr fent que aquests clústers puguin ser moguts segons calgui (tràfic, recursos, etc).

Les contrasenyes en el sistema ABP són fixes, si algú obté aquesta informació posa en risc la informació enviada pels sensors. Alhora aquesta informació està directament relacionada amb l'anomenada Frame Counter, el qual incrementa el seu valor amb cada transmissió. És a dir, un cop Frame Counter assoleix el seu valor límit, allà arriba la vida del sensor dins la xarxa LoRaWAN.

Amb OTAA tant la contrasenya com el Frame counter són negociats amb el servidor un cop activat, millorant la seguretat i la vida útil. En cas de quedar exposada tal informació només ho serà durant aquella transmissió.

Finalment, les gateways són capaces de redistribuir els temps d'enviament (TX) i rebuda (RX) de dades dels sensors perquè dos sensors no enviïn o rebin alhora. Per ABP aquests paràmetres són fixes, en canvi, amb OTAA són totalment reconfigurables i esperables que canviïn. D'aquesta forma l'escalabilitat del projecte es pot realitzar de manera més eficient, segura i fàcil.

2.3. Servidor The Things Network i Stack Comunity

A continuació es descriuen els diferents apartats i funcionalitat que ofereix aquesta plataforma, no només per crear la xarxa LoRa sinó també les integracions que ofereix per trametre la informació a altres aplicacions.

2.3.1. Aplicacions

Com el seu nom indica, les aplicacions de LoRaWAN ofereixen un entorn de registre de nodes, visualitzar les seves dades, descodificar i codificar els missatges i gestionar les integracions juntament amb les seves contrasenyes. En resum crear la xarxa de sensors LoRa.

L'aplicació de TTS compta amb una interfície d'usuari que permet la visualització de l'estat de totes les xarxes creades així com la informació de cada dispositiu, activació, sessió, missatges des dels sensors (Uplnik) o ordres de la xarxa en direcció a aquests (Downlink).Alhora permet gestionar l'enviament de dades a aplicacions externes per ser guardades i gestionades, anomenades integracions.

Les integracions són imprescindibles, la seva utilització permet fer el salt de la informació a l'aplicació principal a través de diferents mètodes i són variats. A continuació s'expliquen les disponibles a TTS a la data de creació d'aquest projecte.

MQTT, missatgeria M2M (Machine to Machine) per HTTP(S) que permet la subscripció del client al servidor MQTT principal de TTN V3 que envia les dades de manera segura i constant a canvi d'un consum de recursos moderat i de la constant revisió de la subscripció.

Webhooks, es tracta d'un protocol HTTP(S) que envia la informació als endpoints (punts de connexió final). Es poden escollir un seguit d'aplicacions relacionades o crear-ne una de pròpia. Tramet les dades amb la detecció d'esdeveniments, d'aquesta manera el client no ha de dedicar tants recursos al servidor principal HTTP.

Storage Integration, TTN V3 ofereix un servei d'emmagatzematge que permet registrar les dades a una base de dades al núvol i recuperar-les més tard.

AWS IoT, permet crear la unió entre el Network Server i AWS, publicant directament a aquesta aplicació.

LoRa Cloud, es tracta d'un servei que expandeix les opcions de l'aplicació TTN V3, permet gestionar funcionalitats que, d'altra banda, requeriria d'una API exterior (no la substitueix completament).

2.3.2. Gateways

L'apartat de gateways permet el registre d'aquests dispositius així com la parametrització de les comunicacions. Veure les dades que està rebent així com les que envia, permetre una primera visió en cas d'avaria i mantenir un control són els principals motius per al seu registre.

Els servidors de TTN V3 contempla la tecnologia "ALOHA" la qual permet la transmissió d'informació en qualsevol direcció a través de qualsevol Gateway. Sempre que un node sigui detectat per una Gateway pública, aquesta transportarà la informació al Network Server i redirigirà la informació al Aplication server corresponent tot i no pertànyer a la xarxa del compte d'usuari on es troba registrada la Gateway.

Cal tenir en compte que la tecnologia "ALOHA" queda anul·lada quan una Gateway es registri com a privada.

Les dades que es poden observar de la Gateway són totals, qualsevol node detectat mostrarà la seva informació, però encriptada. La possibilitat de visualitzar aquest tràfic no implica la potestat per redirigir o desxifrar-lo.

2.4. Servidor principal de la universitat

Les aplicacions descrites a continuació representen la gestió, emmagatzematge i visualització dels sistemes implementats en el servidor del que es disposa a la UdG per a aquest projecte. Per una banda tenim el servidor de JavaScript Node.js, per gestionar la lògica i el tractament de les dades.

Per l'altra el servidor de base de dades mySQL encarregat de garantir el correcte salvaguardat de les dades. I finalment Grafana per tal de poder visualitzar les dades de manera còmode. Seguidament es descriuen cadascuna d'elles amb més detall.

2.4.1. Node.js i servidor JavaScript

Node.js és un entorn de programació el qual, permet el disseny i maneig d'API, majoritàriament per aquelles dedicades a servidors. El seu llenguatge, JavaScript, utilitza una estructura orientada a esdeveniments asíncrons que minimitzen el temps de resposta i maximitza l'escalabilitat del sistema.

Entrant en més detall Node.js és executat sobre el motor Chrome V8 JavaScript. A diferència d'altres entorns de desenvolupament web com ASP.NET, JSP o Spring basades en processament multifil ("Multi-Threaded"), Node.js es basa en l'arquitectura d'un únic fil ("Single Threaded") de processament asíncron i "Non-blocking I/O". Dins un bucle principal, anomenat "Event Loop", aquesta característica permet associar una petició ("Request") d'un usuari de la llista d'entrada ("Event Queue") i associar-lo a un "Thread" de Kernel del servidor en qüestió. En tractar-se d'una ordre asíncrona l'"Event Loop" pot centrar-se a gestionar noves peticions i retornar els valors d'aquelles gestionades anteriorment, dirigir-los als usuaris determinats ("Execute Callback"). La següent figura mostra una visió general d'aquest procés amb les parts més importants definides.

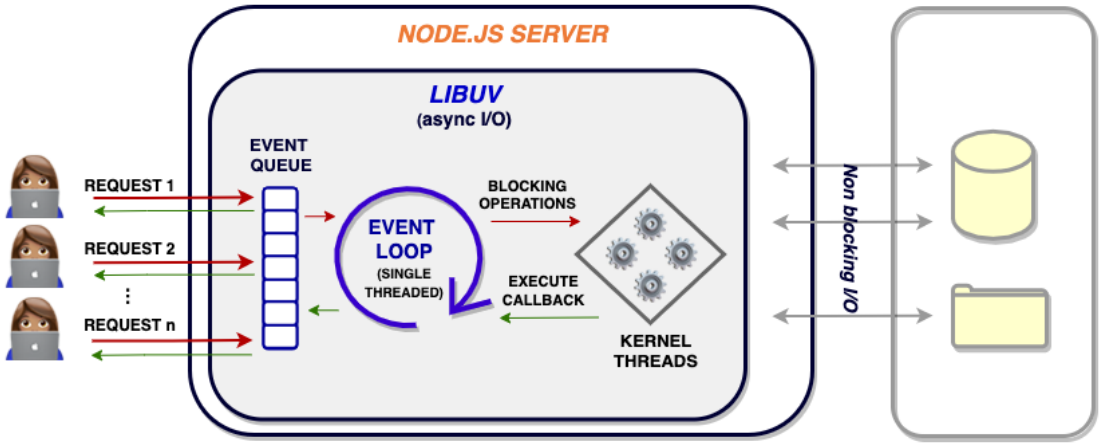

Figura 13 Arquitectura Node.js

Finalment, Libuv és la llibreria sobre la qual se sustenta Node.js per dur a terme les tasques i processos explicats, la qual es defineix com una llibreria multiplataforma, en llenguatge C i especialment dissenyada per donar suport a les funcions asíncrones i "Event Loop".

L'aplicació del servidor creat per aquest projecte està format pels llenguatges JavaScript, SQL i HTML, el llenguatge CSS s'ha deixat de banda, sobre Node.js. Utilitza el gestor de paquets npm (node package manager) integrat al Node.js per gestionar les diferents parts del servidor com la sustentació del propi, el pont d'unió entre aquest i la base de dades, etc.

Per suportar aquest servidor és possible utilitzar qualsevol OS, per les facilitats del llenguatge JavaScript, en ser un llenguatge interpretatiu i sempre que Node.js estigui disponible.

2.4.2. MySQL

Emmagatzemar les dades és indispensable per aquest projecte, mantenir una comunicació neta i fluida entre els diferents programes és clau. No s'utilitza la memòria cache de la mateixa aplicació JavaScrip, les dades es tracten a cada cicle d'execució i totes aquelles extretes de la base de dades no són utilitzades en segones execucions.

Per aquest projecte s'utilitza MySQL, és un sistema de gestió de bases de dades que utilitza el llenguatge SQL. Les bases de dades usades són relacionals multifil, això facilita i treu pes als recursos usats al servidor principal. Alhora permet la creació multiusuària, en aquest projecte només es contempla un únic usuari el qual és de proves.

En el cas del present projecte s'ha utilitzat la pila de software Wampserver, que porta servidor mySQL, servidor Apache i PHP. Per crear la base de dades MySQL, s'ha fet mitjançant l'entorn Workbench, que facilita tant la creació com la gestió de les bases de dades, usuaris, relació entre elles, etc. Permet crear i testejar les connexions creades entre el servidor propi (WAMP) i l'extern (Servidor JavaScript).

El gestor de bases de dades és seleccionat per la seva rapidesa en executar ordres i consultes gràcies a la forma en què ho fa, de forma nativa. Utilitza el PHP fins a la versió 4.X, aquí la versió és la 5, tot i això WAMPServer fa imprescindible l'ús de PHP per fer consultes de revisió i gestió independents del servidor JavaScript.

No s'ha escollit altres bases de dades com NoSQL, TimeRelated, etc. La raó, totes les dades estan perfectament estructurades, alhora fer consultes d'històrics i poder modificar línies de taules requeriria diversos tipus. Per aquest motiu MySQL ofereix una àmplia varietat a canvi d'un lleuger augment de la complexitat de programació.

2.4.3. Grafana

Grafana ofereix una aplicació web d'integració ràpida. Capaç d'analitzar i oferir una visualització interactiva, quadres de comandament, tot a partir de codi obert i multiplataforma. S'hi poden trobar gran varietat de connectors (plugins) que ofereixen gràfics de barres, llistes, historiogrames, etc. Són fàcils d'implementar i relacionar amb la base de dades. Ofereix un gran ventall de possibilitats de fonts de dades, com poden ser MySQL, InfluxDB o Cloud entre d'altres. En el nostre cas, es compte amb recursos més que suficients per MySQL. També permet la creació d'usuaris amb drets i permisos d'accés. S'interpreta com una altra manera de controlar l'accés i la seva visualització.

3. ELEMENTS

El següent capítol explica els aparells utilitzats per mitjà tècnic i la seva funcionalitat. Les especificacions, usos i altres recomanacions esmentades a cadascun dels apartats hi queden reflectits.

3.1. Sensor d'aparcament IOTSens

IOTSens proveeix la tecnologia de l'empresa Winext Technology. Es tracta d'un sensor geomagnètic superficial model AN-101D per a la detecció de vehicles estacionats. Porta integrat un radar, el qual detecta quan un vehicle accedeix o marxa del lloc d'estacionament. Un cop detecta aquest canvi, envia tota la informació a través de radiofreqüència LoRa. A la Figura 14 es mostra el seu aspecte.

Figura 14 Model IOTSens

Per utilitzar el sensor ha d'estar instal·lat correctament. Un cop complet aquest primer requisit es pot calibrar. Si la calibració ha estat satisfactòria (**¡Error! No se encuentra el origen de la referencia.**) el sensor es troba totalment operatiu i només caldrà tornar a realitzar la calibració en cas de canviar la ubicació o durant el recanvi de les bateries.

La funció principal del sensor un cop funcional, és enviar la informació cada cop que detecti l'entrada o sortida d'un vehicle a l'estacionament, el cicle és aproximadament 10 segons, des de la detecció fins l'enviament total de les dades. A part, transmet un missatge d'avís cada sis hores (Funció batec) i d'error si n'ha detectat algun. Finalment, tot i contemplar la recepció de dades (Downlink), per aquest projecte no s'utilitza. El flux que se segueix del funcionament del sensor d'aparcament, queda descrit al següent diagrama de la Figura 15.

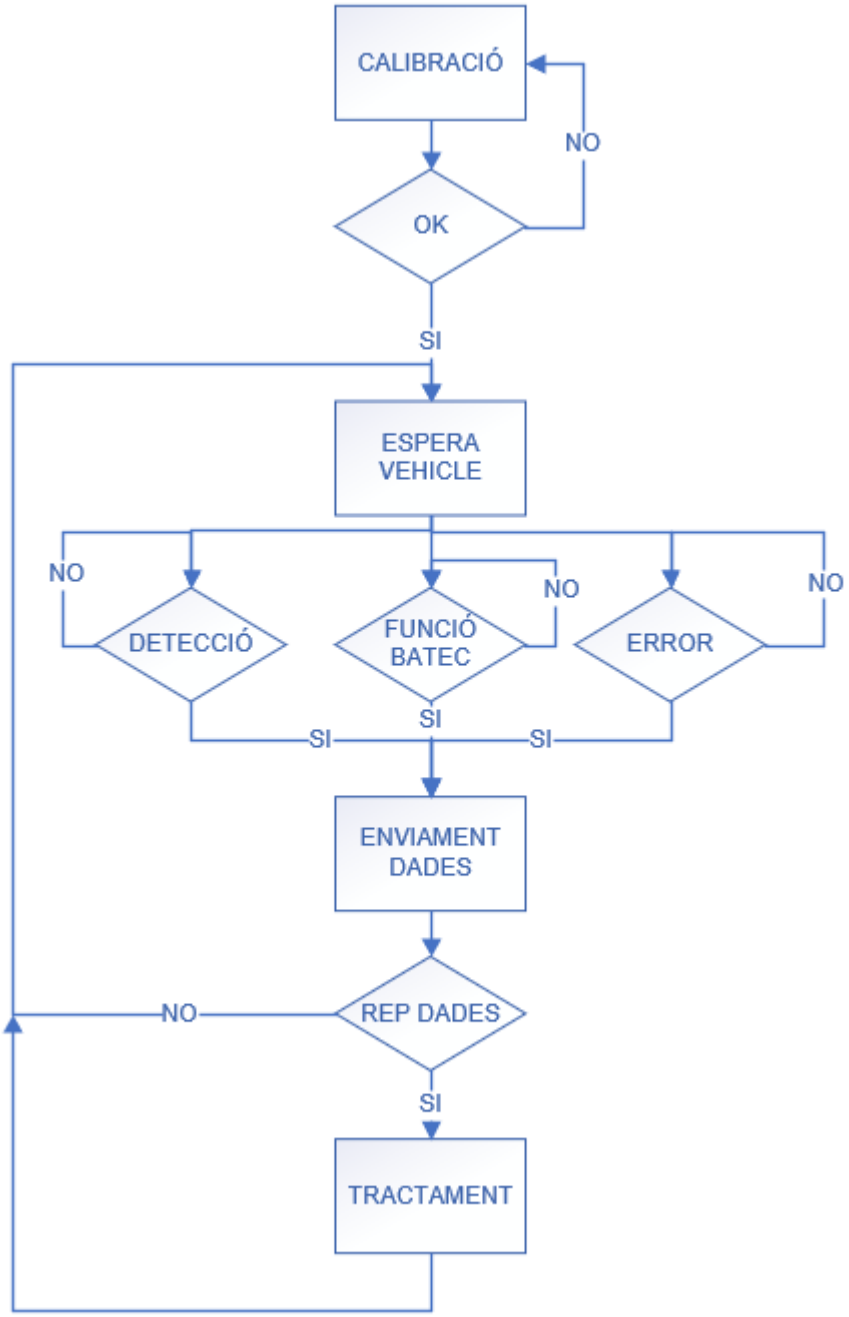

Figura 15 Diagrama de flux del sensor d'aparcament

3.1.1. Característiques principals

El sensor pertany a la Classe A LoRaWAN, pot ser programat per diverses freqüències, CN470 / AS923 / EU868 / US 915, com estem a EU és a EU868.

Les especificacions més importants de l'aparell, les quals motiven la seva raó d'ús són la seva protecció IP67, un rang de temperatura funcional de -40 ÷ 80 ºC, una bateria Li-SOCL2 de 3,6 V de 16.000 mAh i un rang de xarxa aproximat de 500 m i finalment una vida útil aproximada de 4 anys a 20 missatges per dia. Aquesta última dada es redueix considerablement ja que el seu ús serà més elevat, així i tot s'espera que amb un any de vida a 80 usos diaris sigui viable.

Altres especificacions menys rellevants, però igualment recomanables a tenir en compte, com el seu baix consum, una precisió mil·limètrica del radar, muntatge en superfície i programació per Bluetooth, la seva resistència de l'encapsulat, etc.

Finalment, les dades que transmet aquest sensor són diverses i poden ser utilitzades per una àmplia varietat de finalitats.

Per aquest sistema com a mínim es guarden a la base de dades les següents variables: "Frame type", "Direction", "X", "Y" i "Z axis", "Temperature", "Parking Status", "Battery Voltage" i "Sensor type".

La majoria de variables es poden interpretar pel seu nom, però per algunes és necessàri una petita explicació.

"Sensor type" fa referència al tipus (aparcament, botó SOS, gas, etc) de sensor del fabricant Winext.

"Frame type", expressa l'estat del sensor, indica si la informació rebuda és a causa de la calibració, error, detecció, etc.

"Direction", indica si el missatge ha estat enviat des del sensor cap a la xarxa LoRa o rebut en direcció contrària.

3.2. Indicador lumínic

L'indicador consta de tres parts: el seu microprocessador, del fabricant EZSBC, l'aparell encarregat de les comunicacions, xip RFM95W i un encapsulat mecanitzat a la universitat.

Aquest dispositiu és un prototip i fa que els únics requisits necessaris es concentrin en la seva integració dins el sistema LoRaWAN, alimentació a través de bateries i baix consum. També cal que pugui rebre i enviar dades a través d'aquestes comunicacions i poder interpretar i respondre a les ordres del servidor principal. La seva activació és OTAA i de classe A.

L'indicador lluminós envia les dades cada cicle d'activació, marcat per la periodicitat programada.Per aquest projecte es contempla un cicle de 58 segons que inclou l'enviament i gestió de les dades i el mode de molt baix consum ("Deep Sleep"). D'aquesta manera és capaç d'actuar en menys d'un minut.

La informació que s'envia és l'estat del led, si està activat o no. Aquestes dades són tractades en primer pla per l'API LoRaWAN i finalment pel servidor JavaScrip de la universitat.

Un cop enviades, espera rebre dades (Downlinks). Si no en rep cap torna a esperar per l'enviament. Si rep dades, aquest actuarà en conseqüència depenent de si cal activar o no el led.

Finalment, aquest procés queda resumit al diagrama de procés de la Figura 16, on es mostra totes les etapes esmentades anteriorment de forma gràfica i els possibles casos amb que es pot trobar el prototip.

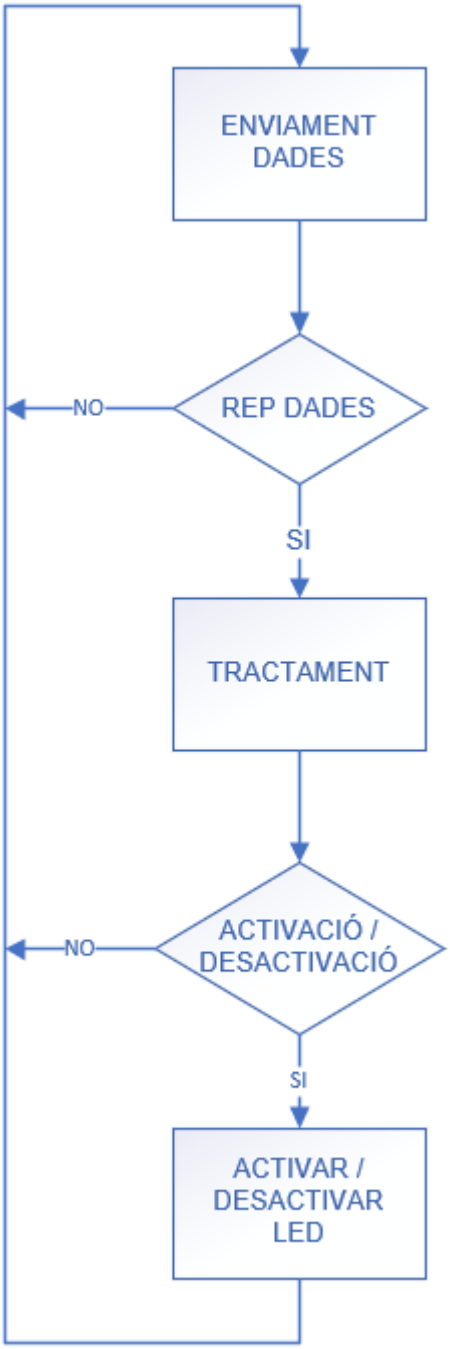

Figura 16 Diagrama de flux de l'indicador lluminós

3.2.1. Microprocessador ESP32 i EZSBC

El microprocessador ESP32 es troba dins el mòdul ESP32-WROOM-32E de la placa ESP32U-01 del fabricant EZSBC. El mòdul ESP32 és de la sèrie SoCs, Xtensa dual-core de 32-bit i

microprocessador LX6 amb 4 MB de memòria flash. Compta amb la integració de mòduls WiFi 802.11BGN HT40 de 2.4 GHz, Bluetooth i Bluetooth LE per possibles comunicacions sense fil. A disposició, 26 GPIO i tot una diversitat de busos de comunicació per controlar perifèrics (SPI, I2C, etc). Al trobar-se integrat dins la placa de EZSBC es resumeixen les següents característiques físiques.

La placa ESP32U-01 compta amb una totalitat de 40 pins, repartits entre d'altres amb 3 busos SPI, 2 per I2C, 2 DAC, 3 UARTs, etc. La placa funciona amb els rangs d'alimentació 3,0 i 3,6 V proveïts a través del regulador de l'USB o els pins Vin (bateria). Compta amb 4 MB de memòria flash, 520 KB de SRAM i finalment 16 KB SRAM dins la memòria RTC, utilitzada per guardar dades amb el mode "Deep Sleep". La Figura 17 mostra el seu aspecte.

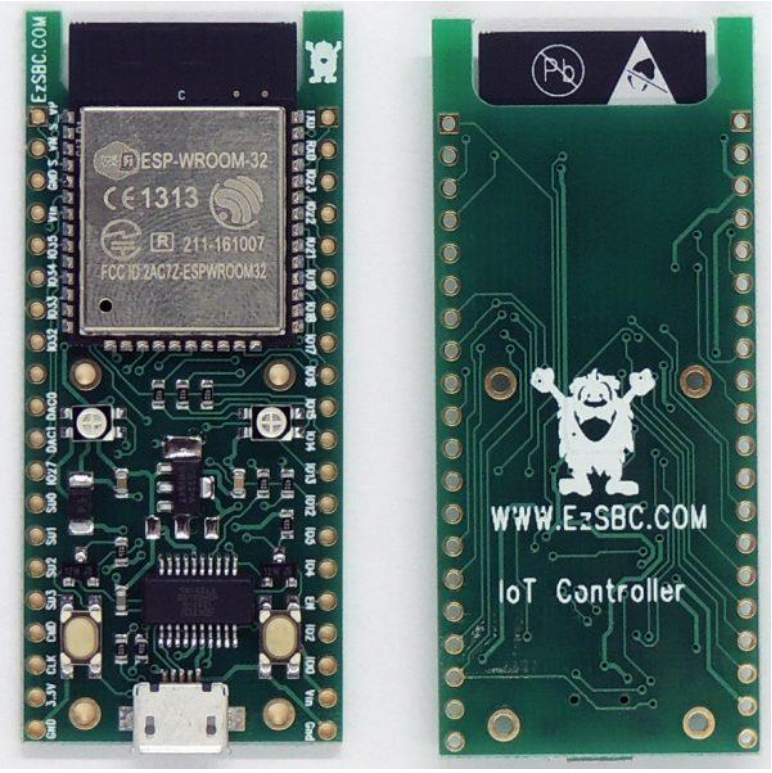

Figura 17 Placa ESP32U-01

La totalitat dels pins es poden observar a la Figura 18. No s'utilitzen tots, l'indicador LED va connectat al pin digital 33 com a sortida i 11 pins més són necessaris per dur a terme la gestió i alimentació del xip RFM95W.

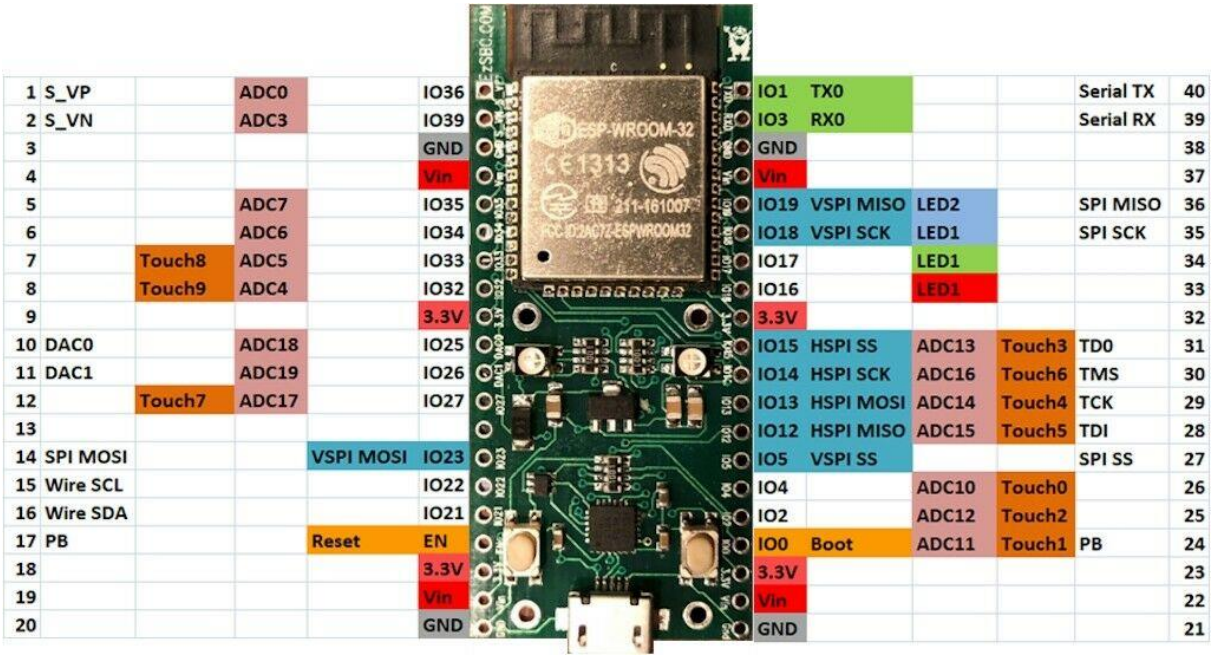

Figura 18 Pinout placa EZSBC

3.2.2. Xip Semtech RFM95W

Hoperf Electronic és una empresa que es dedica a la fabricació i industrialització de sistemes Embedded, ofereix una solució que adapta el xip SX1276 de Semtech en una placa per freqüències de la Unió Europea (EU868 & IN868). La Figura 19 mostra la seva forma.

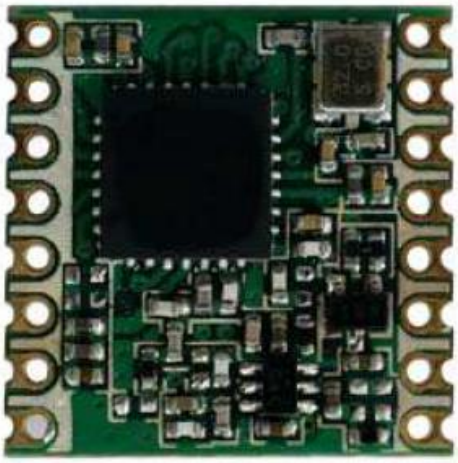

Figura 19 Placa RFM95W

El node RFM95W incorpora l'AFE SX1276 del fabricant Semtech, el rang d'alimentació va de 2,9 a 3,6 V, proporciona -148 dBm de sensibilitat i es combina amb un amplificador de potència integrat de +20 dBm fins a 100 mW, els quals permeten una millor recepció de dades. Altres especificacions importants són els seus 168 dB màxims de "Link Budget", paràmetre que indica la capacitat d'enviament de les dades a través d'obstacles i com a resultat més rang.

Un baix consum per la finestra de recepció de dades RX de 10,3 mA i 127 dB RSSI, que regulen la qualitat del missatge.

La llibreria amb la qual es programa el RFM95W és de codi obert, permet relacionar els pins necessaris, la seva de transmissió, entre 125 ÷ 300 kbps i activació per OTAA o ABP. El diagrama de pins del RFM95W es mostra a la següent figura.

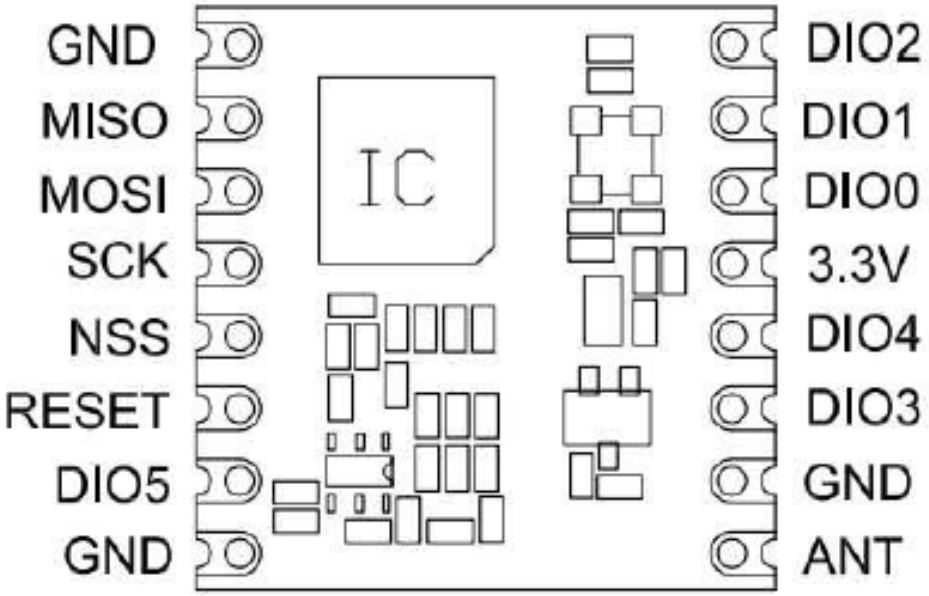

Figura 20 Diagrama de pins RFM95W

Per integrar de manera més satisfactòria aquest node s'ha dissenyat una petita placa (PCB) que integra el RFM95W amb un suport per l'antena i pins alineats de tal manera que es pugui realitzar les proves del prototip amb una "protoboard" i la placa EZSBC.

3.2.3. Encapsulat

L'encapsulat és proveït per la universitat. A causa de les modificacions generades per aquest projecte, ja no conserva les seves característiques inicials. S'ha mecanitzat l'encapsulat per la cabuda de les plaques ESP32 i RFM95W juntes, es contempla espai per posar-hi possibles bateries. Es mecanitzen forats per l'antena, l'indicador LED i una entrada d'alimentació externa en cas de necessitat.

3.3. Gateway MultiTech Conduit

MultiTech és una empresa que ofereix tant material com solucions per la Indústria 4.0. Una d'elles és les comunicacions LoRa. L'ajuntament de Girona compta actualment amb una Gateway MultiTech Conduit per la seva xarxa LoRa de la ciutat.

Com que el projecte és una prova pilot per a l'Ajuntament de Girona, s'ha optat per utilitzar el mateix model de gateway que utilitzen per a construir la xarxa LoRa que dóna cobertura a la ciutat. El model escollit, MTCDTIP-266A-868 contempla les comunicacions LoRa i protocols LoRaWAN de la Unió Europea per la seva correcta implementació. El seu ús com a transportador de dades (Package Fowarder) i FPGA interna permet una computació de les dades molt ràpida i adequada per l'escalabilitat del sistema.

El gateway compta amb 8 canals de ràdio, que es tradueix amb una capacitat de fins a 1.100 sensors diferents amb un SF9 (Spread Factor, recomanat amb LoRaWAN). També disposa d'una antena GPS que permet ubicar l'origen de la transmissió dels missatges per usos extres, tractament de dades, mapes de calor, etc.

El seu encapsulat amb IP67, mecanitzat i suports per la seva utilització en exteriors ofereixen una facilitat i seguretat que altres competidors no ofereixen o en tot cas de menor rang. Finalment, la seva alimentació PoE (Power over Ethernet), facilita la instal·lació elèctrica. La següent figura mostra l'aspecte de la gateway.

Figura 21 Gateway MultiTech

L'antena de la Gateway pot rebre o enviar dades amb un camp omnidireccional (forma de dònut) fet que afecta la sensibilitat i força del senyal rebut (o eviat), segons la provinença d'aquest, si un sensor es troba just a sota de l'antena, pot ser que la intensitat del senyal sigui menor que la d'un sensor posicionat de forma paral·lela amb l'antena. La següent figura mostra una comparativa entre una antena omnidireccional i una direccional.

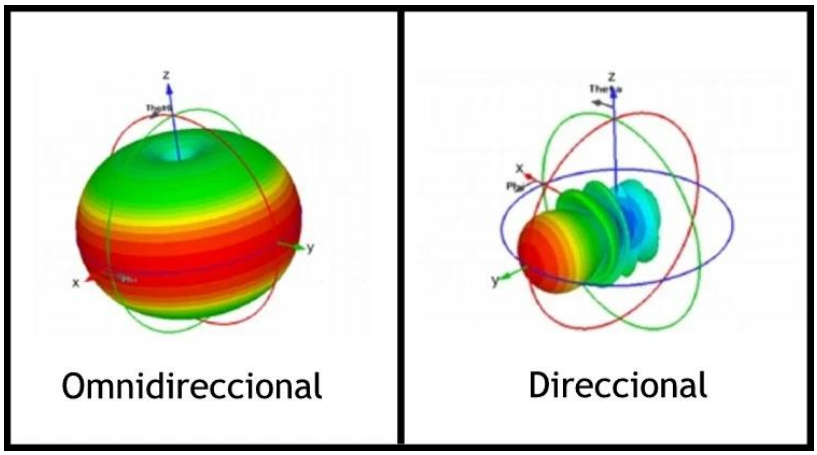

Figura 22 Lobular de l'antena

3.3.1. Alimentació PoE

La gateway queda alimentada a través de la tecnologia Power over Ethernet que permet utilitzar el cablejat de les comunicacions per alimentar a una quantitat limitada de dispositius. Aquest cablejat ha de ser UTP/STP i segons les especificacions de MultiTech, com a mínim de CAT 5.

PoE permet l'activació, apagat o reiniciï dels dispositius de manera remota usant els protocols que existeixen, per exemple SNMP (Simple Network Management Protocol). Per aquest sistema no serà usat.

La recomanació de potència d'alimentació PoE per la gateway demana ser igual o superior a 25 W, ha de complir els estàndards IEEE 802.3af, per aquest motiu s'utilitza el model TPE-115GI de TRENDnet. Aquest dispositiu PoE+ garanteix una alimentació de 30 W amb voltatge de 54 V de corrent continu de fins a 550 mA.

4. SERVIDORS I APLICACIONS

El següent capítol explica les raons tècniques i funcionalitat de les aplicacions i programes utilitzats per la realització del sistema de gestió.

4.1. API The Things Network V3

L'aplicació TTN permet tant la creació de la xarxa LoRa com l'expansió de les comunicacions a través de gateways públiques o privades. Per aquest sistema s'utilitza la gateway de forma pública.

Actualment altres formes de creació de xarxes LoRa són possibles com per exemple amb la marca ChirpStack, però l'ajuntament de Girona ha remarcat l'ús d'aquesta plataforma amb la TTN V3.

TTN ofereix aquesta API per facilitar la integració de la xarxa LoRa dins el seu sistema, Network Server, Aplication Server, ALOHA, etc. A continuació la Figura 23 mostra la pantalla d'inici de la plataforma.

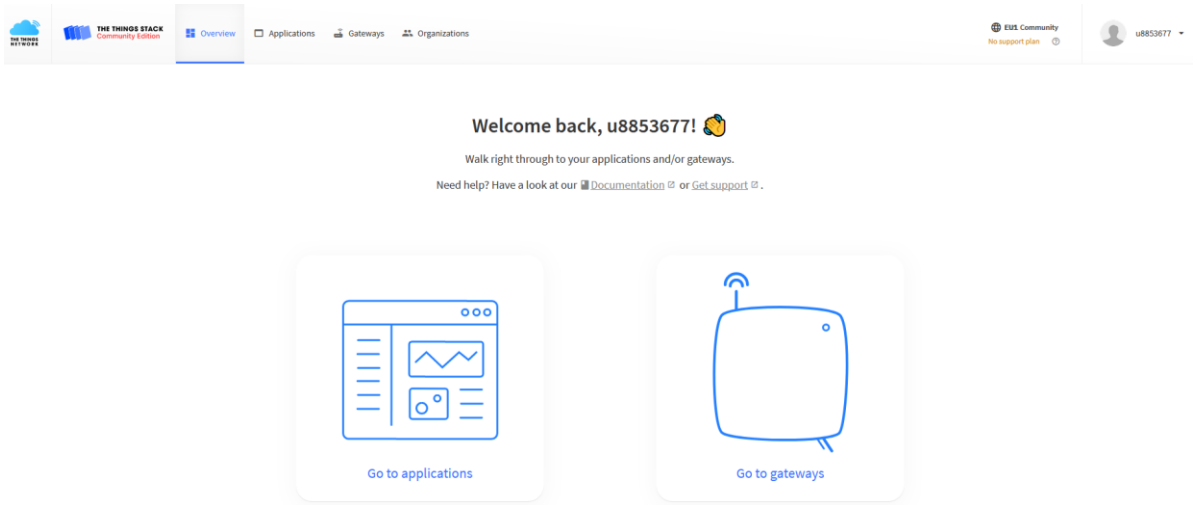

Figura 23 Menú inici usuari UdG

Característiques que afavoreixen el seu ús són la rapidesa i visibilitat dels aspectes més importants, tant dels sensors com de les gateways, així com una àmplia informació a l'abast que permet una entesa i resolució de problemes àgil.

Finalment, per mantenir l'estat de la xarxa, es disposa d'elements de visualització que informen de l'estat de cada sensor o actuador i la possible causa, igual que per les gateways.

Desavantatges, el mateix sistema de gestió no és propi de l'usuari i si es produeix algun error o caiguda de servidor, la companyia TTI ha d'oferir una solució.

Ara bé, des del punt de vista tècnic de sistemes, aporta més beneficis que la pròpia empresa creadora de LoRa, la mateixa TTI, es faci càrrec de tal gestió i del manteniment i hostatge del servidor LoRa.

4.1.1. Descripció de l'aplicació The Things Network V3

L'aplicació de la xarxa LoRa s'ha anomenat Aparcament Cotxe i conté el sensor d'aparcament i dos actuadors lumínics parametritzats amb el tipus A.

Els dos indicadors són utilitzats, un per relació directa amb el sensor d'aparcament i el segon per fer proves d'escalabilitat.

És essencial controlar el ID de l'aplicació com del sensor o actuador de cara a la rebuda o enviament de dades perquè qualsevol integració que ofereix l'API TTN V3 requereix aquests distintius. La següent figura mostra l'aspecte de la xarxa.

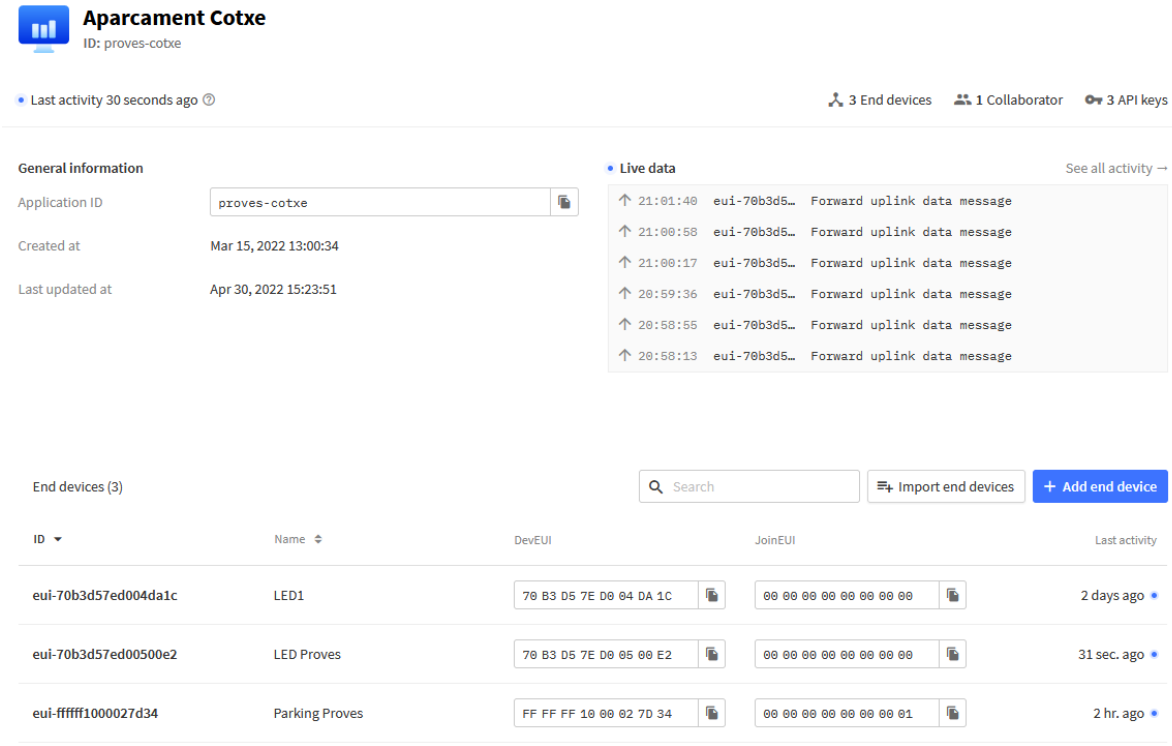

Figura 24 Xarxa aparcament cotxe

Esmentar que la informació que apareix a la figura anterior, tot i que és pública per a qualsevol usuari aliè, no és susceptible de perillositat. Amb la creació de l'aplicació i els nodes registrats correctament cal integrar una solució d'enllaç entre aquesta i el servidor de la universitat.

Abans d'enviar la informació a través de la integració utilitzada es fa un primer tractament de les dades, independent pel sensor i actuador i només per les dades útils. Aquesta informació s'anomena Payload i la direcció des del sensor o actuador cap a la xarxa s'anomena Uplink. Existeix un apartat que permet fer aquest primer tractament anomenat "Payload formatters". Aquí es descodifica el missatge i s'estructuren les dades segons es desitja. Per aquest projecte s'ha utilitzat el format JSON (JavaScript Object Notation). Es preparen les dades per ser transmesses dins les comunicacions de la integració, altre cop encriptada, però a través d'un protocol diferent al LoRa i amb seguretat extra, per exemple SSL, TLS, etc.

Finalment, cal esmentar que la gateway Multitech ha estat creada i registrada dins la part de Gateways de l'aplicació. Anomenada LoRa_Gateway_UdG, permet saber el seu estat i ús. Un apunt important, si es desitja tenir tècnica accés al servei tècnic per part de TTI és que s'ha

de registrar la gateway dins un repositori de Github, cosa que aquest projecte no contempla, la següent figura mostra la pantalla de la Gateway.

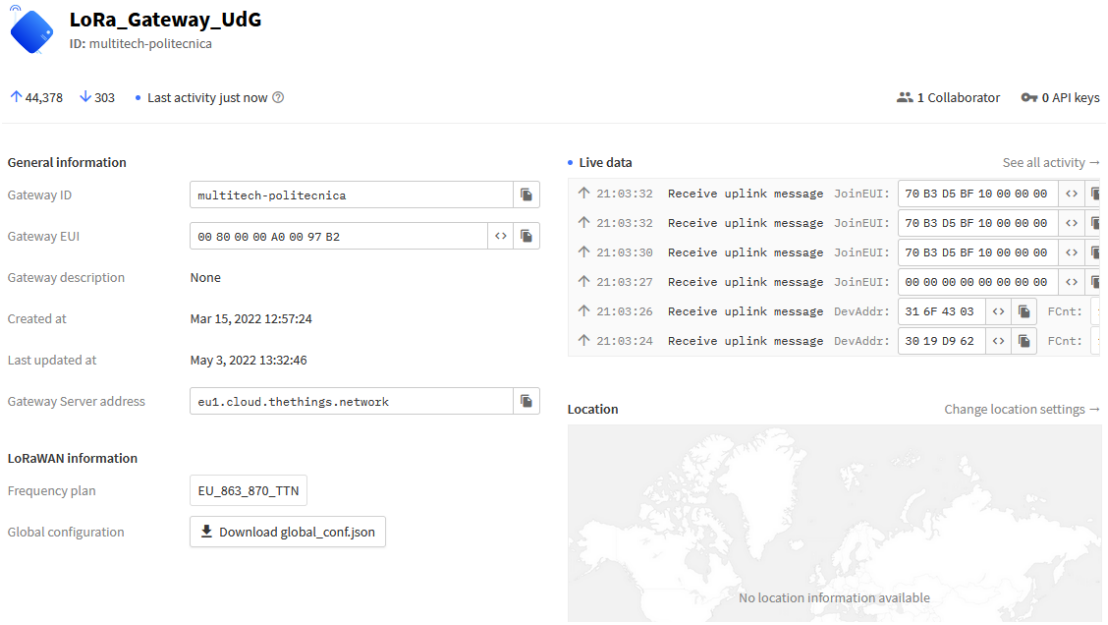

Figura 25 Gateway LoRa

4.1.2. Integració Webhook

L'ajuntament de Girona ha demanat que les dades siguin enviades a través de Webhook, motiu pel qual el sistema s'ha fet d'acord a aquesta integració. La integració Webhook fa d'enllaç entre la xarxa LoRa (TTN V3) i l'API del servidor de la Universitat. És senzilla de parametritzar, ja que es basa en el protocol HTTP(S).

4.1.3. Claus API

La creació d'una API key és necessària per a la recepció de les dades provinents del servidor de la universitat. L'API TTN V3 relaciona aquesta contrasenya que es troba dins la transmissió POST (enviament de dades a través d'internet), provinent del servidor de la universitat gràcies a la integració webhook amb l'aplicació destinatària (Aparcament Cotxe) de manera segura.

4.2. Conjunt servidor universitat

L'aplicació de gestió JavaScript, la base de dades i la plataforma de visualització de la universitat es troben instal·lades a un ordinador Windows que farà de servidor dins el departament eXiT de l'edifici P-IV. Els recursos necessaris per a l'execució i manteniment del servidor es resumeix amb el sistema propi utilitzat, OS Windows 10, processador AMD Ryzen 5 2600 de sis nuclis i 64 bits a 3,4 GHz, 16 GB de RAM i connexió a internet amb cable i velocitats iguals o superiors als 100 Mb.

4.2.1. API JavaScript de gestió

Aplicació principal del sistema, és el programa en llenguatge JavaScript, HTML i SQL, executat sobre Node.js encarregat de la rebuda, enviament i gestió de la informació i lògica.

S'ha optat per crear l'API personalitzada a causa de la complexitat a l'hora d'aplicar la lògica del sistema d'aparcament. Tot i ser executat sobre l'ordinador pot ser exportat a servidors externs al núvol com AWS, Google Cloud, Netlify, etc. D'aquesta manera l'escalabilitat del sistema podria créixer exponencialment, així i tot no s'ha optat per aquesta solució degut a la complexitat extra i possibles recàrrecs monetaris.

El servidor utilitza varis paquets de programació, com express o axios, per gestionar les peticions HTTP (POST, GET, etc) i mysql per la base de dades. Aquests paquets són imprescindibles i fàcils d'utilitzar causa de la seva ràpida implementació i comprensió. Al tenir oficialitat npm i de codi obert permet obtenir uns resultats que han estat provats anteriorment i obren la porta a modificacions i adaptacions del programa. Altres softwares com NodeRed han estat explorats, però tot i donar bons resultats inicials han suposat una dificultat extra a l'hora d'implementar la lògica, fent que s'optés per l'opció de crear l'aplicació personalitzada.

La funcionalitat principal d'aquesta aplicació és la gestió de la lògica implementada. Compta amb dos processos independents que revisen constantment, de forma periòdica, la informació de la base de dades i actuen en conseqüència.

El procés principal revisa la informació de la base de dades, en concret a la taula que registra l'estat dels sensors i actuadors, crea dues llistes i per tots aquells sensors i actuadors relacionats que: Primer, han avisat de l'estacionament d'un vehicle a la seva plaça, segon, que l'actuador lumínic relacionat no es trobi activat i tercer, que no s'hagi enviat prèviament una ordre d'activació pel seu actuador. Un cop s'han creat les llistes principals, es torna a consultar a la base de dades amb la informació de la llista dels sensors, però en aquest cas a l'històric, per saber el temps que porta el vehicle estacionat. Es crea un tercer llistat amb aquestes dates i es processa cadascuna d'elles. Si aquest temps és superior al desitjat es crea una ordre d'activació de l'actuador i es tramet a l'API LoRaWAN (Aparcament Cotxe), la qual ja és l'encarregada de fer arribar al actuador. Això es fa per tots els actuadors registrats a la llista creada. El diagrama de flux d'aquest procés queda esmentat a la següent figura.

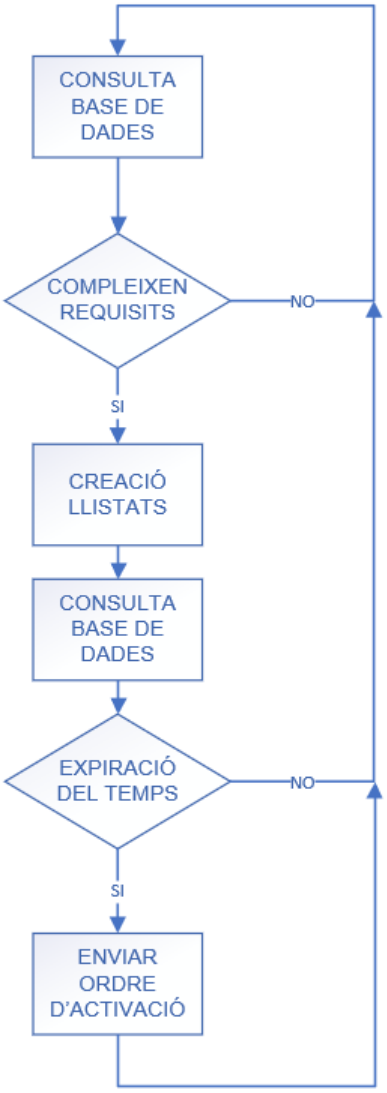

Figura 26 Diagrama de flux procés principal de gestió

El segon procés crea dos llistats iguals al primer, però amb les condicions inverses. D'aquesta manera pot saber que el vehicle ja no es troba estacionat i cal desactivar l'actuador. Per aquesta raó fa la consulta a la base de dades per crear els primers dos llistats i acte seguit envia una ordre a cadascun dels actuadors per desactivar la seva il·luminació.

Llavors, fa actualitzacions a la taula de la base de dades corresponent i d'aquesta manera el primer procés actua immediatament amb els sensors disponibles. La Figura 27 queda resumida l'explicació anterior.

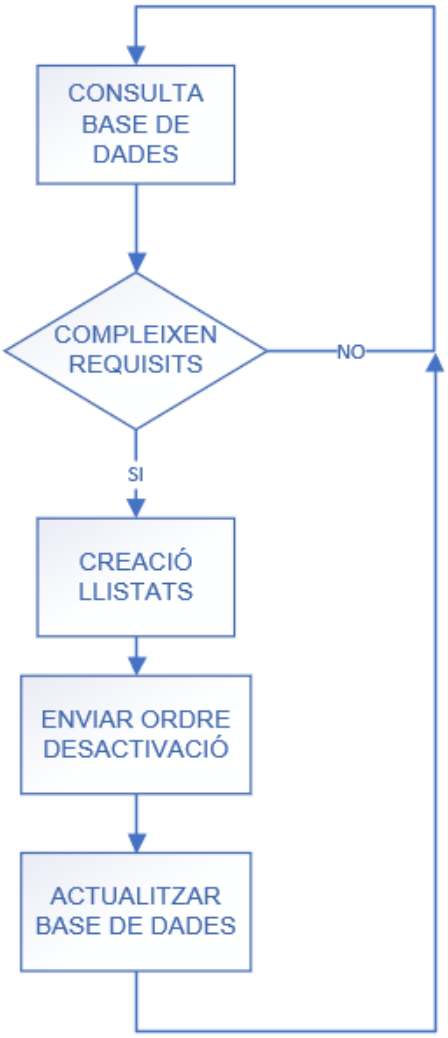

Figura 27 Diagrama de flux procés secundari de gestió

4.2.2. Base de dades MySQL i WampServer

L'ús del servidor WampServer i la base de dades MySQL s'escull gràcies a la seva facilitat d'integració, disseny i funcionalitat així com l'experiència prèvia amb aquesta. Ambdós compten amb llicència gratuïta GLP (General Public License).

El programa MySQL Workbench versió 8.0 CE permet la creació de les comunicacions amb el servidor Wamp versió 3.2.6 i la base de dades a través de comunicacions, que poden o no tenir seguretat extra, usuaris, comunicacions amb certificats, etc.

Per aquest sistema s'ha optat per la utilització d'usuari sense contrasenya i sense certificat SSL. Al trobar-se el servidor i la base de dades dins al mateix sistema operatiu tot queda a la mateixa xarxa WLAN sense haver d'accedir a Internet.

MySQL Workbench ha permès la creació de la base de dades que conté les tres taules utilitzades per aquest programa. La primera, anomenada "sensor_cotxe", que destaca per ser l'historial de tots els sensors d'aparcaments, conté tota la informació dels sensors d'aparcament. La seva lectura és realitzada pel servidor JavaScript i Grafana, mentre que la seva escriptura queda restringida solament pel servidor JavaScript. La Figura 28 mostra el seu aspecte gràcies al MySQL Workbench amb les seves columnes i tipus de dades.

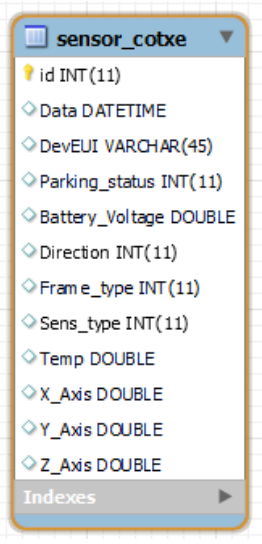

Figura 28 Taula DB sensor_cotxe

La segona taula, "gestió_cotxe", queda restringida per ús exclusiu de l'API servidor i el seu ús ha estat dissenyat per contenir un llistat que relaciona cada sensor d'aparcament amb el seu indicador lumínic. La resta de paràmetres són d'ús funcional per a la lògica de gestió com s'ha enviat una ordre i l'estat del LED. Seguidament es mostra una figura amb les seves columnes i dades.

Figura 29 Taula DB gestio_cotxe

Finalment, la tercera taula "sensor_cotxe1", és una còpia de la primera que permet fer proves de noves implementacions o canvis realitzats, ja sigui a l'API JavaScript o Grafana abans de configurar-los a la taula principal, i d'aquesta manera no malmetre informació valuosa. La Figura 30 mostra l'aspecte d'aquesta taula.

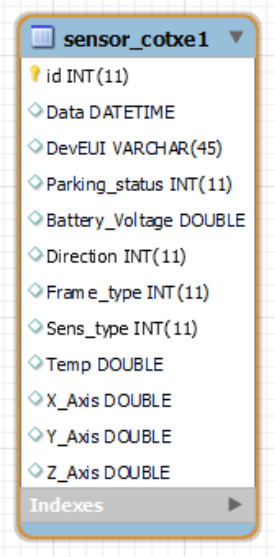

Figura 30 Taula DB sensor_cotxe1

Altres bases de dades com "InfluxDB" o "MongoDB" han estat estudiades i rebutjades per problemàtiques que dificultaven la gestió de la informació i implementació de la lògica i afegien més complexitat de la requerida.

4.2.3. API de visualització Grafana

S'ha utilitzat Grafana "OSS" versió 8.5 per Windows 10 i 64-bits sota la llicència AGPLv3 (Affero General Public License). Aquesta versió incorpora les funcions bàsiques i plugins estàndards amb la possibilitat d'expandir-los a través de la seva biblioteca en línia i permet l'escalabilitat a la versió per empreses si es desitja.

Grafana ha facilitat la creació d'un panell que permet veure la visualització de la informació més rellevant, com l'estat de l'aparcament o del sensor. A continuació s'exposen les diferents eines de visualització implementades (gràfics, taules, etc). Aquest panell anomenat "Aparcament_UdG" dins la carpeta General, està separat en dos desplegables anomenats Estat Aparcament i Estat Sensor, la següent figura mostra la visió general del panell.

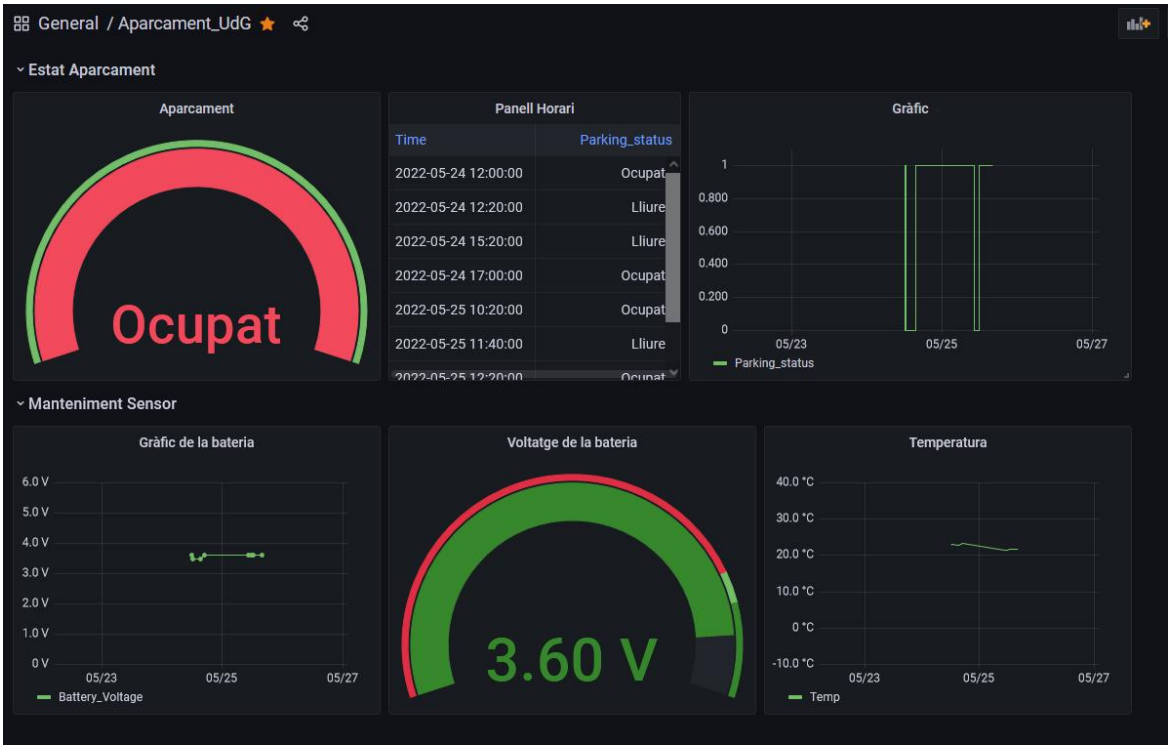

Figura 31 Panell Aparcament_UdG

El primer desplegable, Estat Aparcament, compta amb un calibrador, una taula i un gràfic. Tots ells mostren l'estat de l'aparcament i cada plugin vol respondre a peticions específiques, com per exemple visualitzar ràpidament l'estat (Gauge), saber l'hora precisa d'un estat (Taula) o freqüència i durada (Gràfic).

Gauge o calibrador, en aquest cas binari, anomenat "Aparcament" indica si es troba ocupat o lliure. Per saber aquest estat, Grafana fa una petició a la base de dades de l'última informació "Parking_status" del sensor específic. A la següent figura es veu l'estat de l'aparcament quan es lliure.

Figura 32 Gauge Aparcament Lliure

Quan es troba ocupat, el color canvia al vermell i mostra el text corresponent, a la figura següent es veu el seu aspecte canviat.

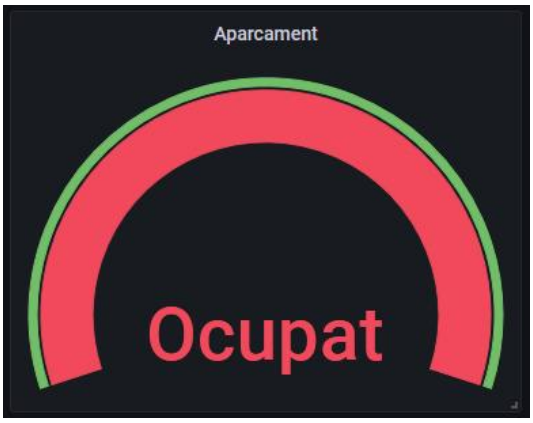

Figura 33 Gauge Aparcament Ocupat

La taula s'anomena "Panell Horari" i mostra la data i hora en què l'aparcament ha canviat d'estat, lliure o ocupat, a la Figura 34 es poden visualitzar aquestes referències. La precisió horària pot ser configurada per mostrar la data sencera o només l'hora. Per aquest projecte es deixa per defecte.

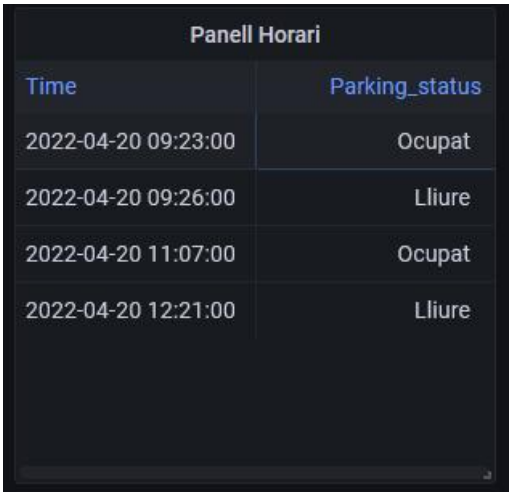

Figura 34 Panell Horari

Finalment, el gràfic. Aquest permet visualitzar l'estat al llarg del temps de manera més còmode. Cada punt mostrat són les dades recopilades de la base de dades. L'estil quadriculat de la gràfica ha estat parametritzat per concebre millor la informació mostrada. La Figura 35 mostra algunes dades d'exemple.

Figura 35 Gràfic d'ocupació

El segon desplegable, anomenat "Estat del Sensor" contempla dues gràfiques i un calibrador, per mostrar les dades més rellevants.

La primera gràfica amb nom "Temperatura", mostra els graus que registra el sensor cada vegada que envia un canvi en l'estat de l'aparcament, s'ha escalat el gràfic per mostrar valors en el rang de -10 a 40 ºC dins les últimes sis hores. La Figura 36 mostra unes mesures preses al laboratori.

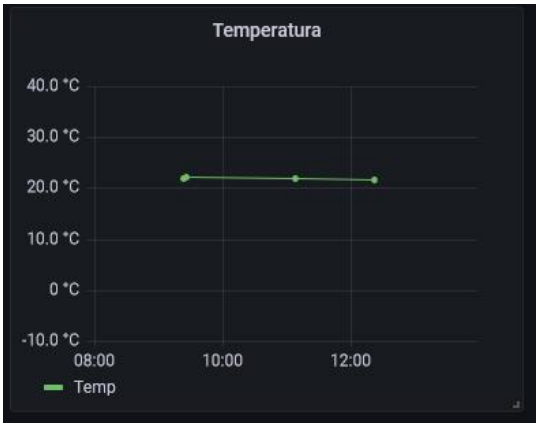

Figura 36 Gràfic Temperatura

La calibració, permet visualitzar el voltatge de la bateria (d'igual nom) amb el qual s'ha parametritzat per canviar de color segons el nivell de la bateria.

Els rangs són els següents: de 0 a 3,3 V el color canvia a vermell, valors entre 3,3 i 3,4 V es mostra de color verd pàl·lid i per sobre d'aquest últim valor es mostra en verd tal com s'observa a la següent figura. La calibració serveix per visualitzar ràpidament el valor exacte del voltatge de la bateria i donar un avís instantani a través del canvi amb el color.

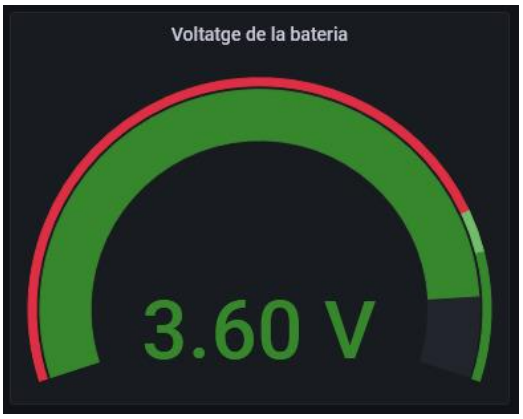

Figura 37 Gauge Voltatge de la bateria

Finalment, l'últim gràfic va relacionat amb l'anterior i es crea per representar les dades de la bateria al llarg del temps, dintre les últimes sis hores i de rang 0÷6 V. La Figura 38 mostra el seu aspecte.

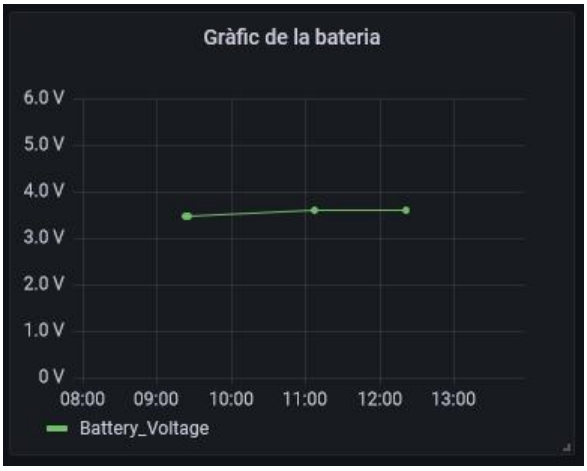

Figura 38 Gràfic de la bateria

5. CONFIGURACIÓ I DESPLEGAMENT

El següent capítol especifica els paràmetres i les funcions que fan de pont d'unió entre servidors, paràmetres de configuració de dispositius i la zona on s'ha desplegat aquest sistema després de les proves realitzades al departament. D'igual manera s'indica els passos a seguir per iniciar la totalitat del sistema. Cal seguir tots els aspectes d'aquest capítol per obtenir un resultat satisfactori amb el sistema de gestió d'aparcament.

5.1. Desplegament de cobertura al campus

Dotar de cobertura al campus i a una gran part de la ciutat de Girona (principalment Montilivi), és un punt fonamental. Per aquest motiu s'ha escollit el terrat de l'edifici P-IV de l'EPS en tractar-se d'un punt àlgid dins al campus i per poder tenir un control de les comunicacions.

El muntatge s'ha dut a terme cablejant l'Ethernet per les instal·lacions de l'edifici, des del laboratori del grup de recerca eXiT fins a la terrassa. L'alimentació PoE s'ha realitzat a la part més alta possible dins el quartet de manteniment que dóna accés a la terrassa, d'aquesta manera s'uneixen les comunicacions i l'alimentació el més a proper possible a la Gateway i també, evitant la utilització d'un quadre elèctric exterior. La gateway s'ha muntat al parallamps de la terrassa, ja que és l'estructura més rígida i que permet una altura superior de l'edifici, concretament s'ha instal·lat a 1,8 metres respecte al terraplè més proper i amb l'antena perpendicular al terra. A la següent imatge es pot observar la seva instal·lació.

Figura 39 Muntatge Gateway

5.2. Configuració Gateway

Per configurar la Gateway és recomanable llegir amb atenció el manual d'instruccions del model exacte de MultiTech utilitzat (MTCDTIP-266A-868). No obstant, aquest apartat resumeix les configuracions claus pel correcte funcionament i integració dins la xarxa LoRa de la universitat.

Si la Gateway es troba en configuració de fàbrica, sigui perquè és nova o ha estat decisió pròpia i es desitja accedir-hi, cal obrir un navegador web i introduir la IP següent: "192.168.2.1", un cop dins es demana l'usuari i contrasenya que es troba al manual. La configuració per un model nou o buit és diferent del que s'explica a continuació.

La gateway de la UdG ja ha estat configurada amb els paràmetres bàsics d'usuari i s'ha instal·lat a la rosseta número cinc del laboratori del grup de recerca eXiT. La seva IP és la "84.88.154.165". La contrasenya és confidencial i cal demanar-la al responsable del compte.

Un cop dins el menú principal cal accedir al menú LoRaWAN i NetworkSettings. A la següent figura es veu aquest directori, on es determina el mode de la gateway respecte LoRa.

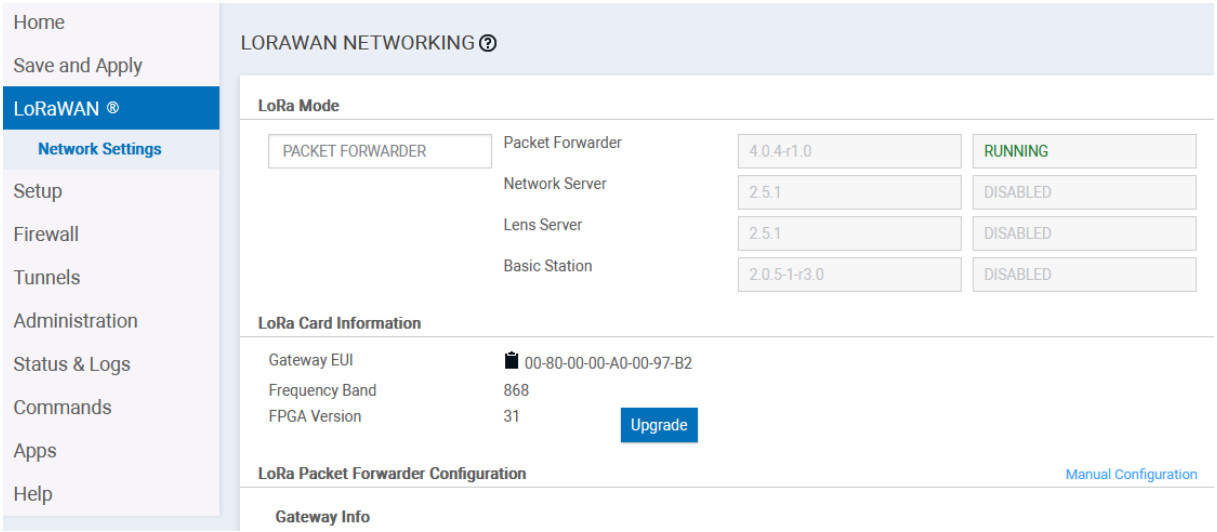

Figura 40 Gateway configuració LoRaWAN

Llavors, cal estar segurs que per aquest projecte s'ha seleccionat l'opció de "Packet Forwarder" com a mode de funcionament. Un cop es trobi connectat a la xarxa i plenament configurat, el seu estat ha de dir "RUNNING".

Aquesta configuració requereix la selecció de la freqüència d'ús. Com es Europa, se selecciona "EU868" dins el "Channel Plan".

La resta de paràmetres es deixen per defecte. Per revisar que la informació és correcte cal mirar la següent figura.

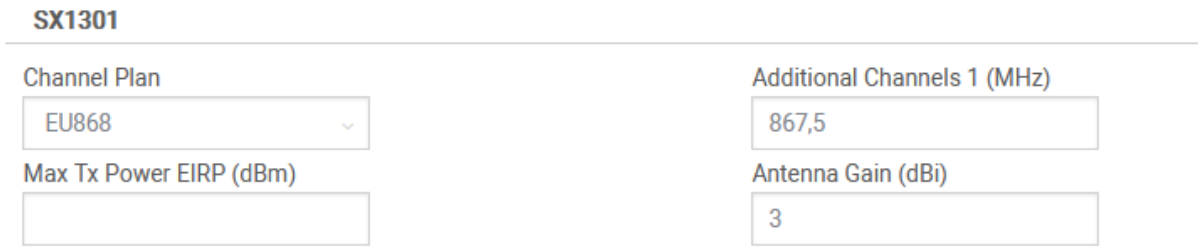

Figura 41 Gateway Channel Plan

Els paràmetres descrits a la Figura 42 són els necessaris i no s'han de tocar, sempre que no es vulgui canviar l'ús de la gateway dels servidors LoRaWAN a d'altres com per exemple ChirpStack, etc.

Dins l'apartat "Basics" és fonamental revisar que el paràmetre "gateway ID" sigui el correcte, alhora dins "Server" que el servidor siguin els de la companyia TTI, i a "Server Address" ha de constar "eu1.cloud.thethings.network".

També és convenient revisar que els paràmetres "Network" i "Upstream & Downstrem Port" siguin "Manual" i "1700" respectivament.

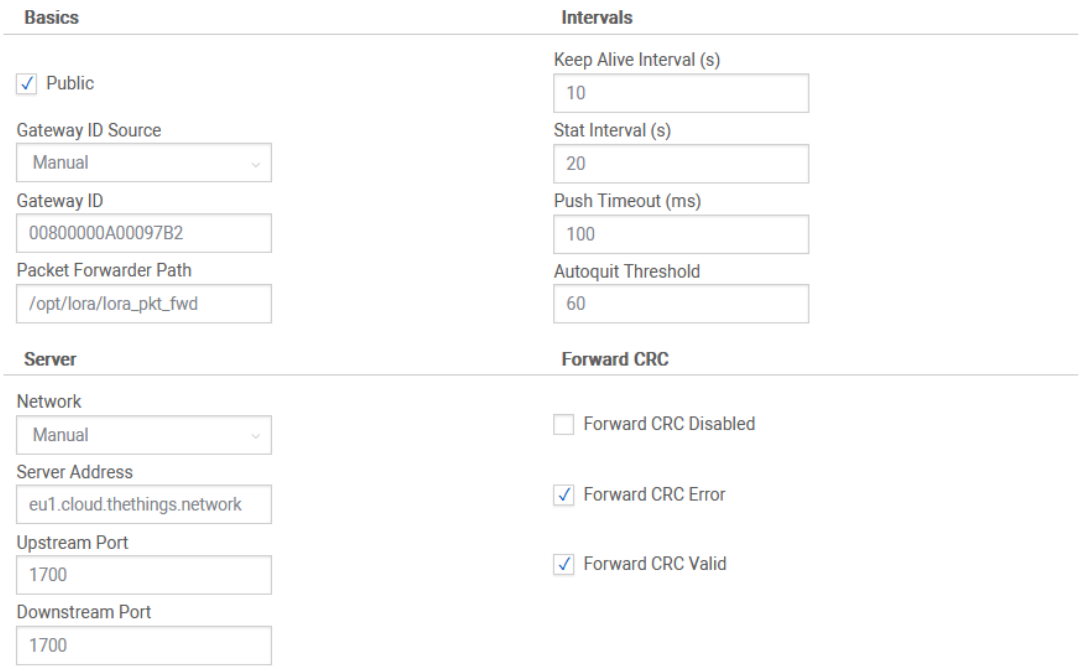

Figura 42 Gateway configuració bàsica

Finalment, per fer la connexió amb la xarxa d'internet de la UdG, s'ha d'accedir a l'apartat "Setup" i seleccionar la xarxa "eth0" de "Network Interfaces", i revisar que la IP i la versió són correctes. Si es desitja configurar una modificació cal editar aquesta xarxa amb el botó de la columna "Opcions". La Figura 43 Gateway Network Interface mostra aquesta xarxa.

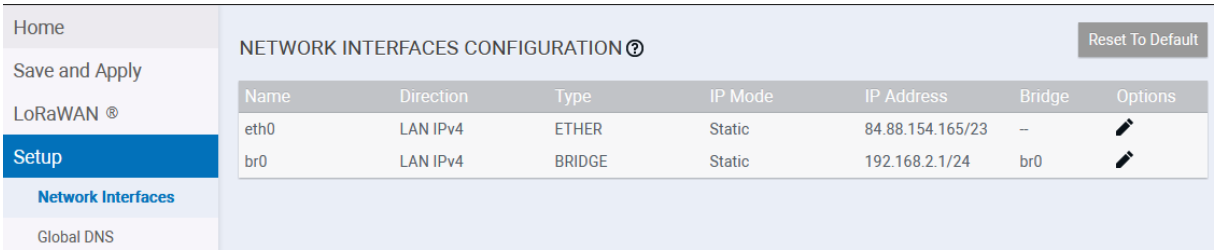

Figura 43 Gateway Network Interface

Segons la xarxa del laboratori eXiT fa falta una configuració com la mostrada a la Figura 44 Gateway configuració eth0. Qualsevol altre paràmetre modificat pot ser motiu del mal funcionament de la gateway.

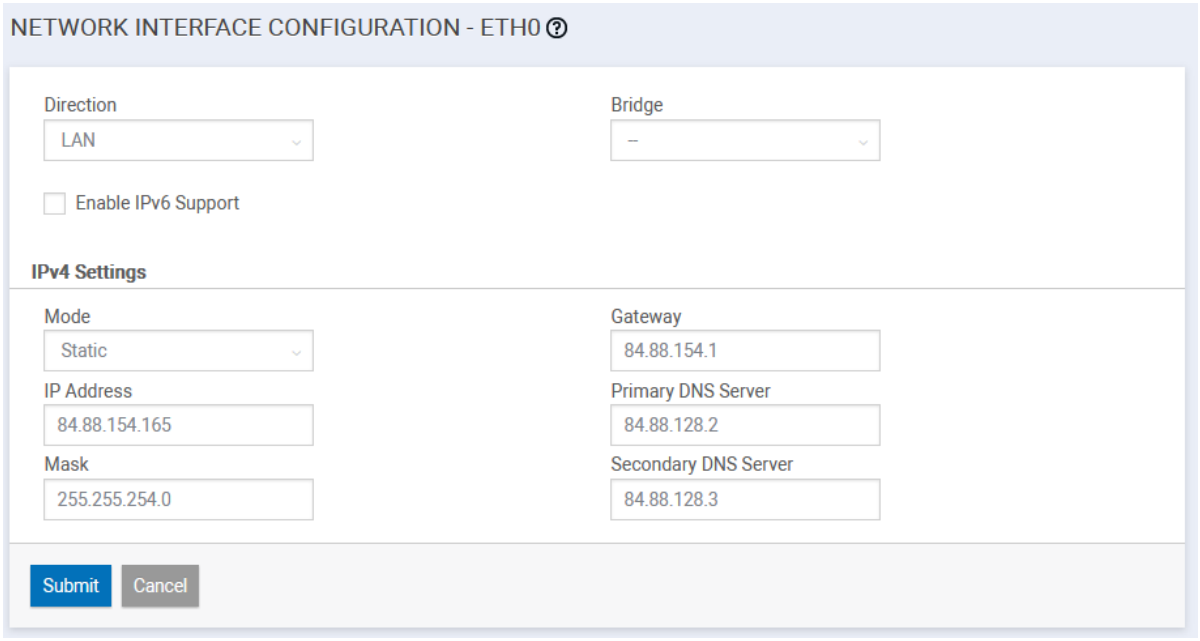

Figura 44 Gateway configuració eth0

5.3. Configuració i calibració actuador lumínic

L'actuador ha de configurar-se durant la programació del dispositiu, prèviament al seu muntatge dins l'encapsulat. La primera acció que s'ha de parametritzar és la llibreria "lmic" per adaptar-la a la placa EZSBC i establir el pla de freqüències.

Totes les llibreries i funcions usades es troben sota llicència MIT que permet el seu ús i modificació així com la seva industrialització.

Al directori i arxiu següent "\.pio\libdeps\esp32dev\MCCI LoRaWAN LMIC library\project_config\lmic_project_config.h" fa falta que el codi sigui idèntic al que es dóna a continuació.

```
// project-specific definitions
#define CFG_eu868 1
//#define CFG_us915 1
//#define CFG_au915 1
```

```
//#define CFG_as923 1

// #define LMIC_COUNTRY_CODE LMIC_COUNTRY_CODE_JP      /* for as923-JP; also define
CFG_as923 */

//#define CFG_kr920 1

//#define CFG_in866 1

#define CFG_sx1276_radio 1

//#define LMIC_USE_INTERRUPTS
```

És imprescindible que quedin definits els següents aspectes amb els valors corresponents: "CFG_eu868 1" defineix la freqüència de la Unió Europea, "CFG_sx1276_radio 1" defineix el model de chip Semtech.

Per tal d'integrar correctament el sensor dins la xarxa LoRa (Activació JOIN) calen tres paràmetres, APPEUI, DEVEUI i APPKEY. Al ser activació OTAA els únics paràmetres que quedaran fixes són l'APPEUI i DEVEUI.

Començant per l'APPEUI es tracta d'una cadena de vuit caràcters expressats en hexadecimal. Per l'actuador es deixa tots els camps a zero per tal de diferenciar-lo del sensor d'aparcament. Al següent fragment de programa es visualitza el resultat final. Tenir en compte el concepte "Little-endian", el qual indica que el número de més pes és l'últim, d'esquerra a dreta.

```
// APPEUI Little-endian

static const u1_t PROGMEM APPEUI[8]={ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}
```

El DEVEUI és la ID del dispositiu, que altra cop es tracta d'una cadena de vuit caràcters en hexadecimal. En aquest cas, però, s'ha d'inventar perquè el mòdul RFM95W no en proveeix cap. De nou el format és "Little-endian", tal com es veu al següent extracte de programa.

```
// DEVEUI Little-endian

static const u1_t PROGMEM DEVEUI[8]={ 0xE2, 0x00, 0x05, 0xD0, 0x7E, 0xD5, 0xB3, 0x70
};

void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}
```

Finalment, l'APPKEY és una cadena de setze caràcters en hexadecimal. En ser activació OTAA només servirà per a l'activació i llavors queda gestionada per part dels servidors corresponents. El següent codi mostra aquest paràmetre emplenat amb zeros per no revelar informació confidencial. Destacar la forma en "Big-endian" (invers a "Little-endian"), diferència dels dos paràmetres anteriors, la seva escriptura és inversa a la "Little-endian".

```
// APPKEY Big-endian

static const u1_t PROGMEM APPKEY[16] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

void os_getDevKey (u1_t* buf) {  memcpy_P(buf, APPKEY, 16);}
```

La relació entre els pins de l'embedded i el mòdul RFM95W de Semtech queda reflectida a la següent taula. No tots són configurats al programa a causa de la seva programació definida, com és el cas de les comunicacions SPI, en la que els pins usats són els mateixos que per defecte utilitza la llibreria SPI, sempre que no s'indiqui el contrari durant la seva inicialització.

PINS Usats	
ESP32U-01	RFM95W
GND	GND
3,3V	3,3V
IO2	DIO0
IO21	DIO1
IO22	DIO2
IO5	NSS
IO4	RESET
IO19	MISO
IO23	MOSI
IO18	SCK

Taula 1 Mapa pins ESP32U-01 i RFM95W

La secció del programa que s'ha de configurar pel correcte funcionament és la següent. És imprescindible definir els pins necessaris dins d'aquesta estructura. Els ".nss" com a "CS" o "SS" de les comunicacions SPI, el ".rst", reset del RFM95 i posar per ordre ascendent el ".dio".

```
// Pin mapping EZSBC i RFM95W
const lmic_pinmap lmic_pins = {
    .nss = 5,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 4,
    .dio = {2, 21, 22},
};
```

Finalment, aquest actuador no requereix cap calibració, més enllà de la correcta connexió de l'antena.

5.4. Configuració i calibració sensor aparcament

Per configurar el sensor són necessaris dispositius i programari específic. Aquest apartat dóna consells i recomanacions pel seu ús. Al manual de característiques es troben les instruccions completes i recomanades pel fabricant.

Cal destacar la importància d'haver instal·lat el dispositiu prèviament al lloc de desplegament desitjat perquè en tractar-se d'un sensor magnètic si es canvia de posició un cop calibrat, s'han experimentat errors de transmissió.

Activar el programa i detectar el dispositiu a través del dispositiu extra de programació connectat a l'ordinador.

Per allargar la vida del dispositiu cal ser ràpid i saber prèviament les accions a realitzar, les comunicacions es fan a través de Bluetooth i comporten un consum elevat durant el període d'utilització.

Un cop s'ha detectat el dispositiu cal executar la calibració amb el botó "Calibration", si tot és correcte el missatge es mostra a la següent figura.

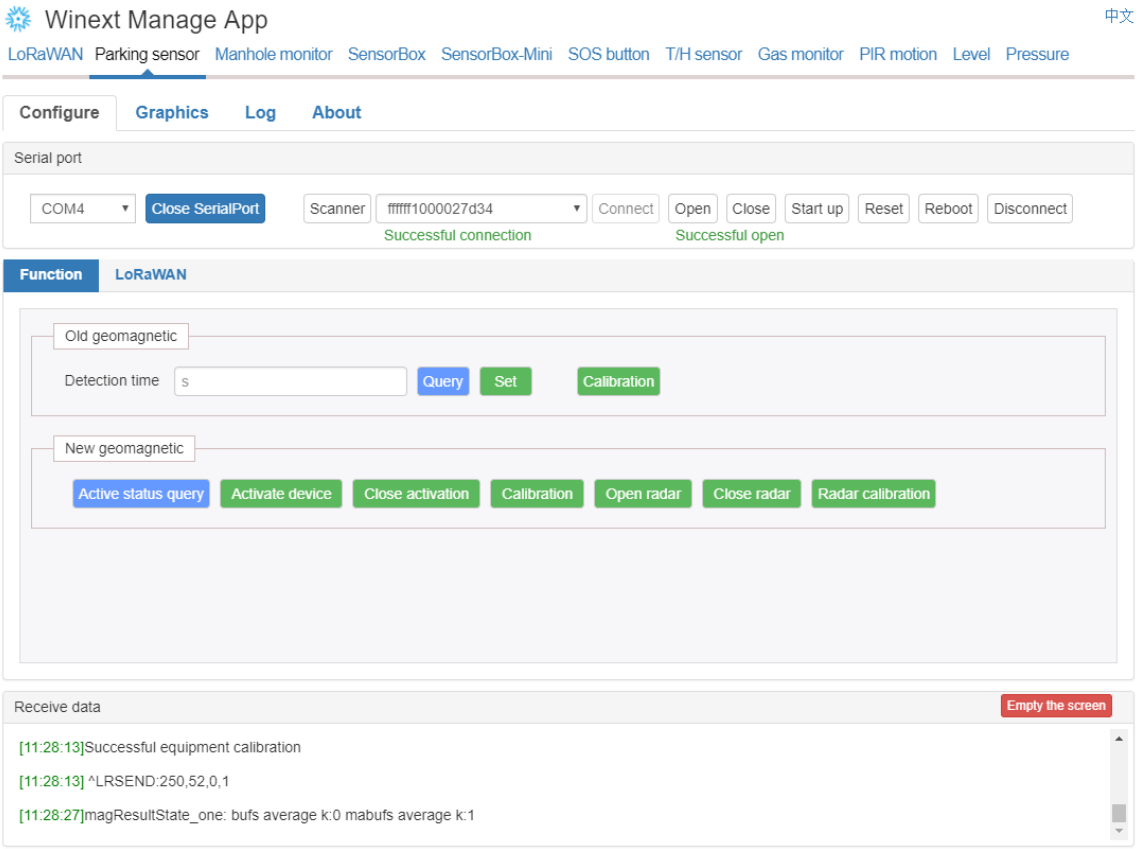

Figura 45 Calibració sensor aparcament

Qualsevol altra calibració és estàndard de fàbrica, i no s'ha modificat per aquest projecte, llavors no ha d'influir amb les funcions per les quals ha estat seleccionat. Més informació al manual d'ús del sensor.

Finalment, les claus d'activació del sensor són definides per IoTSens i només cal registrar el sensor dins la xarxa amb els següents valors amb activació OTAA.

DevEUI: FFFFFF1000027D34

AppKey: Reservat

AppEUI: 0000000000000010

5.5. Configuració aplicació sensorial TTN V3

La creació de l'aplicació no implica determinar el següent: restriccions de freqüència (SFx, Duty Cycle, versió LoRaWAN, etc.), activació dels sensors, integracions i contrasenyes API (API keys) tots aquests aspectes es tracten posteriorment. La xarxa es va configurant a mesura que es van integrant solucions com el nom "Aparcament Cotxe" i ID "proves-cotxe", aquest ID és important perquè és utilitzat per redirigir la informació. Una vegada creada l'aplicació i es vol integrar un sensor o actuador nou cal tenir en compte si ja existeix dins el registre de dispositius LoRaWAN, cap dels dos dispositius utilitzats es troba registrat i cal fer-ho de la manera manual. Si es trobessin actualment registrats en alguna altre aplicació, aquesta ens alertaria d'aquest fet i no deixaria continuar amb la configuració.

Abans de registrar manualment un dispositiu s'ha de mirar si existeix un model dins la TTN de LoRa, buscar l'empresa i el model. A continuació es veu una figura que mostra aquest menú de creació, es veu com el sensor d'aparcament no està disponible tot i que l'empresa IOTSens ja en te d'altres registrats.

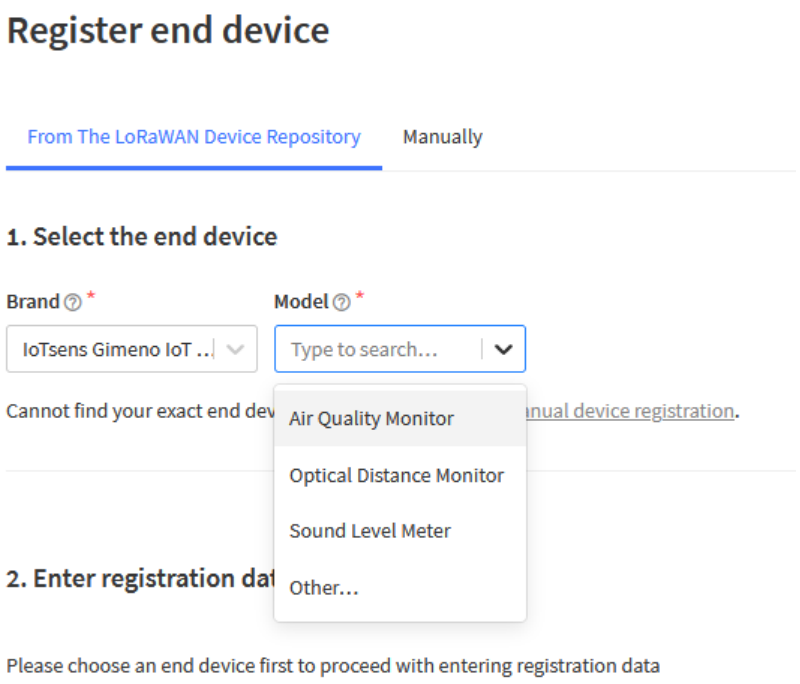

Figura 46 Registre dispositiu

Llavors, cal seguir els passos que es troben al desplegable "Manually" i especificar tota la seva configuració, com el "Spread Factor" (SF9, Europe863-870 MHz), aclarir que tot i seleccionar un "SF9" no indica que el dispositiu vagi a fer-ho amb aquest valor específic sinó com a manera genèrica, l'activació (OTAA), la versió LoRaWAN ("Specification 1.0.3") els paràmetres APPEUI, DEVEUI i APPKEY. Aquests últims paràmetres són Big-endian, tenir en compte segons la configuració de cada sensor. Un cop creat no es poden canviar, si es desitja cal eliminar el dispositiu de l'aplicació i crear-lo de nou.

Llavors, pel dispositiu d'aparcament IOTSens dota de la informació que s'han d'emplenar els paràmetres "APPEUI", "DEVEUI" i "APPKEY" per realitzar l'activació OTAA.

5.5.1. Configuració Webhook

El nom que s'ha donat al Webhook és el ID "api-webhook-udg", aquesta ID és el nom que respondrà en cas d'enviar dades des del servidor de la UdG per realitzar "Downlinks". En primer lloc, tot i ser informació susceptible de vulnerabilitat s'ha incorporat una API key conjunta que frena possibles atacs informàtics.

Pel que fa a l'estructura de la informació escollida es determina que sigui de tipus JSON. Llavors, cal una "Base URL", enllaç on es vol enviar la informació. Aquest URL de l'API creada, es pot tractar de dues formes, com a servidor a la universitat utilitza el domini "https://84.88.154.119:40300" o utilitzant un mètode de "Tunneling". Per la realització de les proves s'ha utilitzat aquest ñultim mètode amb el programa Ngrok (Proxy privat) que canvia d'URL cada cop que s'inicialitza, ja que la privacitat amb que compta la universitat de missatges entrants al servidor general és restrictiva.

Aquest programari contempla que el sistema sigui HTTPS per donar la seguretat necessària a cada transmissió. A continuació es mostra la figura de la configuració Webhook amb una URL desusada generada per Ngrok.

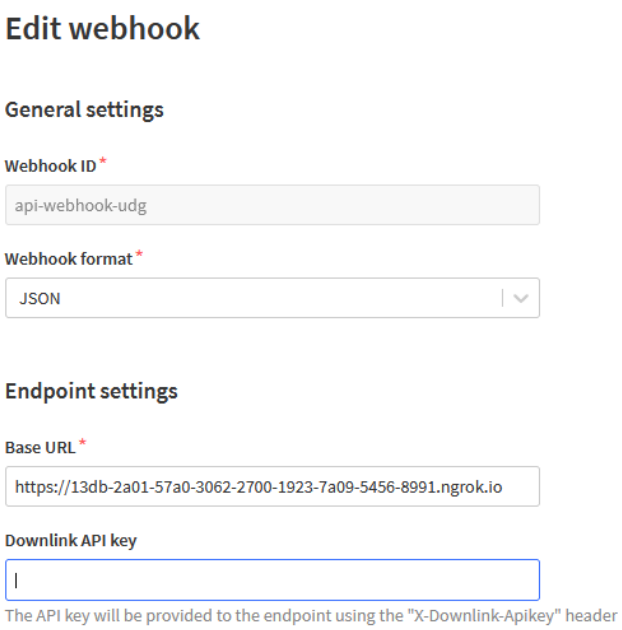

Figura 47 Configuració webhook part 1

L'apartat Downlink API key es deixa en blanc perquè es crea de manera independent i l'API TTN sap gestionar-la un cop s'envia des de l'API servidor de la universitat.

Finalment, s'habilita la classe de missatge que es vol enviar, en aquest sistema només cal "Uplink message", s'emplena el camp amb "/sensor" que marca la direcció final on es fa la transmissió HTTP amb el mètode POST via TCP/IP.

La direcció "Uplink" de "/sensor" respon tant pel sensor d'aparcament com per l'actuador lumínic. A la Figura 48 Configuració webhook part 2 es mostra el resultat final dins el camp corresponent.

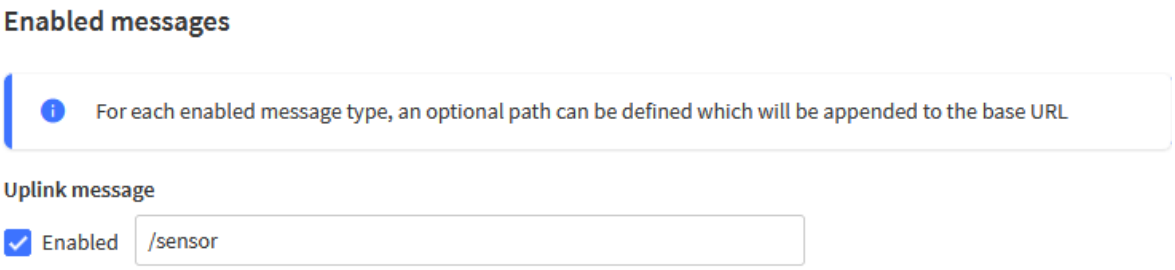

Figura 48 Configuració webhook part 2

5.6. Configuració i inicialització servidor JavaScript

Prèviament a l'activació del servidor cal repassar tres aspectes clau de configuració: Port d'activació, usuari de la base de dades i l'actualització i correcte instal·lació dels paquets "npm".

El port de l'aplicació s'ha escollit tenint en consideració els rangs d'utilització, el 40.300 ha estat l'escollit i si es vol instal·lar a un altre servidor que no sigui el de la UdG cal revisar que no sigui utilitzat per cap altre aplicació. Al següent extracte de codi del "servidor.js" (nom de l'arxiu principal) s'observa l'activació del servidor HTTP gràcies al paquet "express" i acte seguit la configuració del port.

```
//Executem el servidor i client
const app = express();
const port = 40300;
```

Un cop revisat el port s'ha d'introduir les credencials de les comunicacions creades amb la base de dades i especificar l'usuari i contrasenya (si escau), el host i la base de dades.

La connexió es troba configurada dins el mateix servidor així que el host és el "localhost", la base de dades "mydb" i només l'usuari "root", a continuació es veu la funció del programa implementat.

```
//Base de dades MySQL
var mysql      = require('mysql');

var connection = mysql.createConnection({
  host     : 'localhost',
  user     : 'root',
  password : '',
  database : 'mydb'
});
```

Es pot observar la crida del paquet "mysql" encarregat de gestionar aquesta connexió i posteriors funcions de crida a la base de dades.

Finalment, cal estar segurs de que tots els paquets es troben instal·lats i actualitzats per això cal executar la comanda "npm install" al directori on es troba l'arxiu arrel "server.js" a través de la línia de comandes CMD i esperar que retorni un missatge satisfactori.

Iniciar el servidor és senzill però és important haver iniciat el servidor WampServer prèviament i estar segurs de que ho ha fet correctament.

La comanda "npm start" a través de la CMD al directori on es troba l'arxiu "server.js" inicia el servidor, si tot s'activa segons l'esperat el servidor contesta amb les línies "Aplicació executada a la direcció http://localhost:40300" i "Database server running!".

Si es desitja establir les comunicacions pont entre el servidor i xarxa LoRa des d'una xarxa diferent a la utilitzada pel servidor de la UdG cal executar la comanda "ngrok http 40300" amb una CMD separada a la del servidor i amb la direcció arrel del programa "ngrok.exe".

Llavors, cal copiar l'URL de l'apartat HTTPS dins la integració webhook de l'aplicació LoRa (explicat a l'apartat anterior).

5.7. Configuració i inicialització WampServer MySQL

La configuració de les comunicacions entre l'aplicació principal i la base de dades es fa a través del programari MySQL Workbench. Al primer desplegable anomenat "MySQL Connections", cal especificar el nom de la connexió, "Sensors_Parking".

Dins el desplegable "Connection" cal seleccionar el protocol de connexió (TCP/IP) i a l'apartat "Parameters" emplenar correctament els camps "Hostname" (127.0.0.1 o localhost, és el mateix), el Port 3306 definit pel servidor WampServer i l'usuari "root", la contrasenya no és usada per aquest projecte.

A continuació es pot observar la Figura 49 on es resumeix l'explicació i la configuració descrita anteriorment.

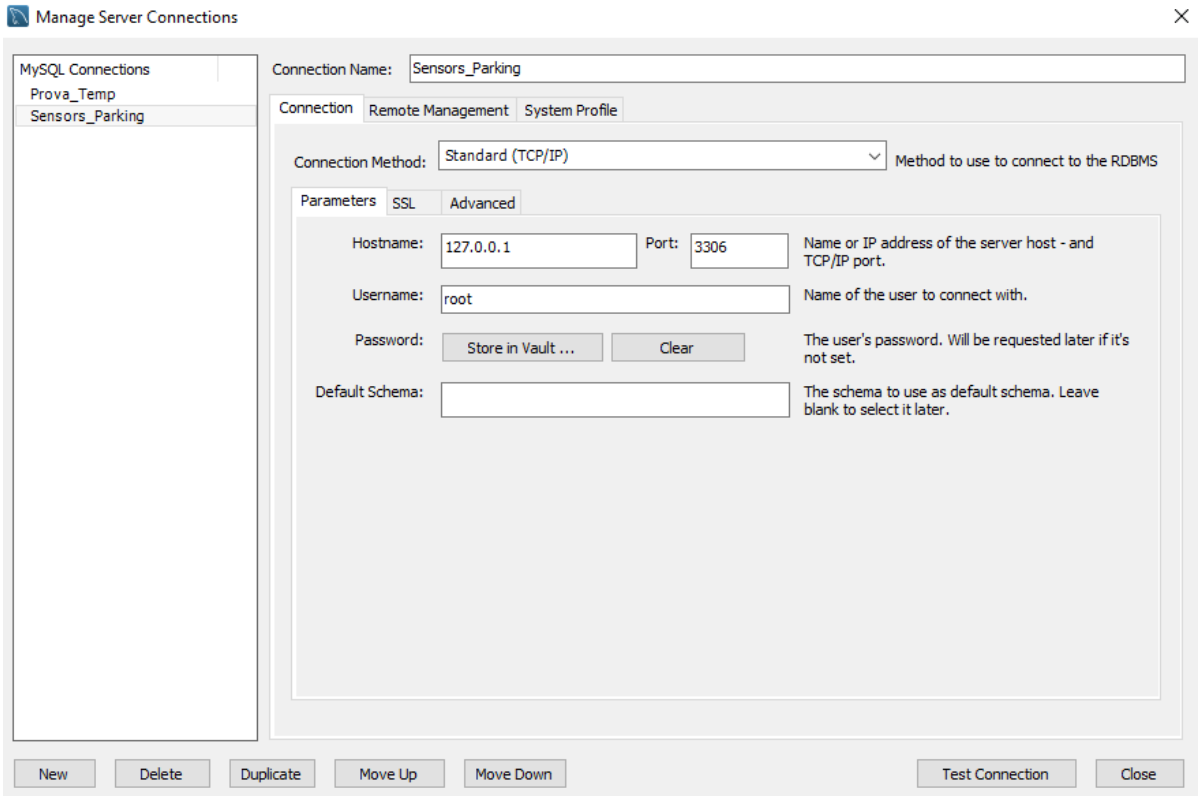

Figura 49 Configuració connexió MySQL i WampServer

La seguretat de la connexió TCP/IP és parametritzada al panell anomenat "SSL", per aquest projecte es deixa per defecte amb "If available", perquè al tractar-se de comunicacions dins el mateix servidor no s'ha trobat necessari la utilització de certificats SSL o TLS, la següent figura mostra aquest apartat.

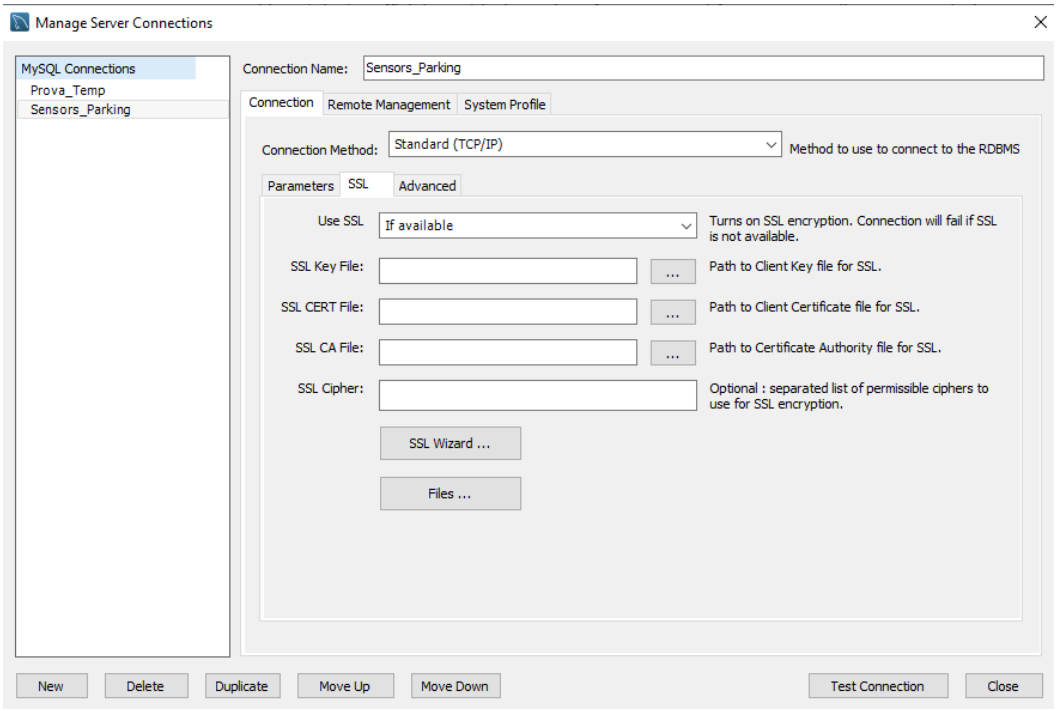

Figura 50 Configuració SSL MySQL

Un cop configurada la connexió i la base de dades creada, cal iniciar l'executable del WampServer, com el port utilitzat per aquest és el 3306, igual al configurat al Workbench, la connexió és automàtica. Per saber si el servidor Wamp funciona correctament la icona del programa surt de color verd. Cal dir, que compta amb dos estats més, de color taronja significa que la connexió s'està iniciant i en vermell que es troba desconnectat (per decisió de l'usuari o un problema). A la següent figura es pot visualitzar la icona del WampServer en correcte funcionament.

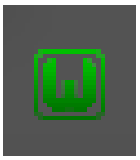

Figura 51 WampServer en funcionament

Per configurar les bases de dades i les seves taules és recomanable utilitzar el programa Workbench i deixar de banda les eines de modificació que ofereix WampServer com "phpMyAdmin", el qual permet la modificació de les bases de dades i els seus registres en funcionament. Per això és necessari anar a la segona opció del Workbench anomenat

"Models" i seleccionar l'arxiu "DB_Ajuntament" on es troben les taules descrites al subapartat 4.2.2.

A continuació apareixen les taules de la base de dades "mydb", fent doble clic es pot modificar, esborrar i crear els camps que es desitgin.

Si es realitzen canvis a les taules de la base de dades des del programari Workbench només fa falta utilitzar la funció "Forward Engineering" per actualitzar els valors dins el servidor WampServer, d'altra manera el servidor continua executant les bases de dades desactualitzades i els canvis no són aplicats. Aquesta opció es troba dins el menú "Database" com es mostra a la següent figura on tot seguit, el programa indica els passos a seguir.

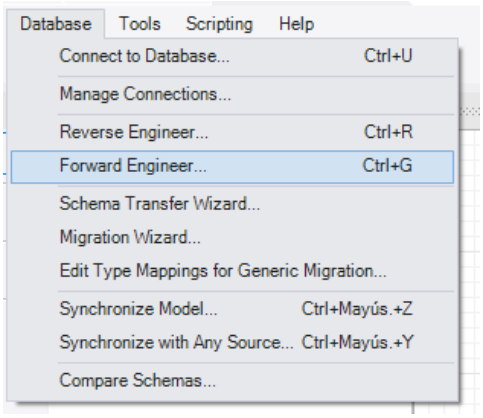

Figura 52 Forward Engineering

5.8. Configuració i inicialització Grafana

Un cop instal·lat el programari necessari per Grafana, aquesta queda en suspensió en segon pla fins que algun usuari requereix el seu ús. El port usat per Grafana és el 3000, i per accedir a l'aplicació creada cal obrir un buscador web (Google, firefox, etc) i posar la direcció "http://localhost:3000" al mateix servidor de la UdG, ja que no es compta amb cap mètode de comunicació amb l'exterior com Proxy, "Forward Tunneling", etc.

Per extreure les dades amb Grafana de MySQL es requereix la creació de l'anomenat "Data Source", per realitzar aquesta acció es va a l'opció "Configuration", se selecciona aquesta opció i es prem el botó per afegir la base de dades de tipus MySQL.

La següent figura mostra com accedir a aquesta opció a través de la configuració i la selecció de l'opció "Data Sources".

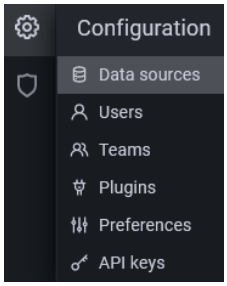

Figura 53 Grafana Data Source

Un cop creada, els seus paràmetres queden disponibles per a la seva modificació, aquests són molt similars als creats per la connexió del WampServer.

El nom del Data Source és "MySQL" i cal revisar que la configuració dels paràmetres "Host", "Database", "User" i "Password" siguin els correctes.

El "Host" ha de tenir la direcció del WampServer "localhost:3306" (o 127.0.0.1:3306), el nom de la base de dades amb "mydb", usuari "root" i la contrasenya en blanc.

La resta de paràmetres es deixa per defecte. A continuació es mostren els paràmetres d'aquesta connexió a la Figura 54 Data Source MySQL.

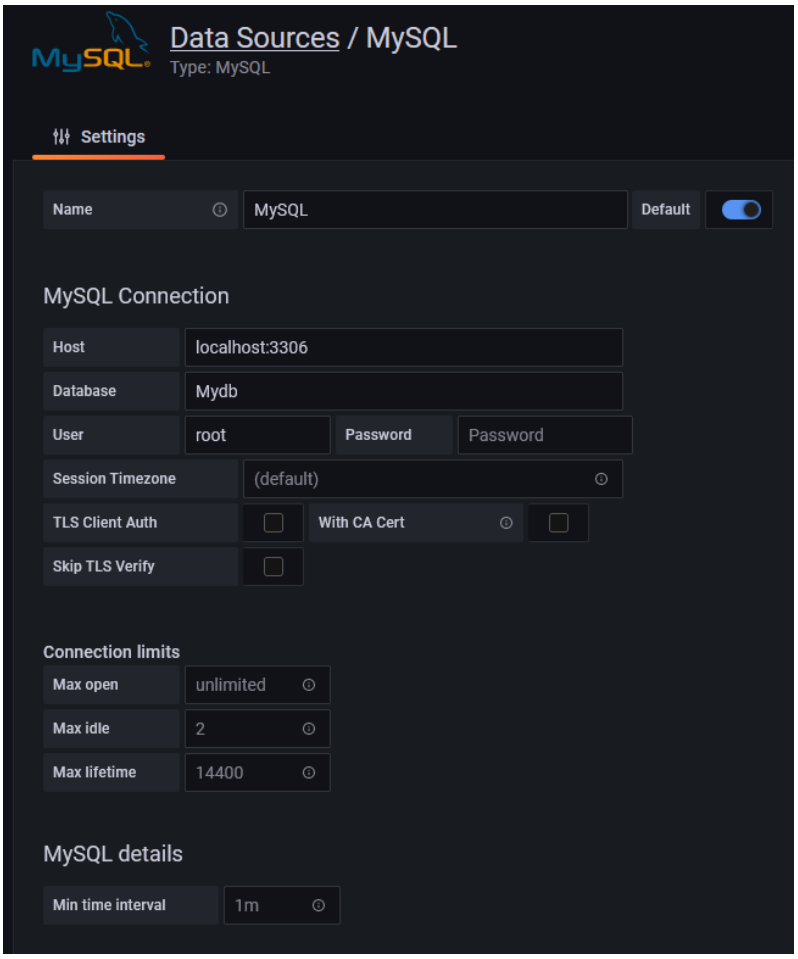

Figura 54 Data Source MySQL

Per saber si la connexió és funcional hi ha el botó "Save & test", com indica el seu nom, guarda la configuració i envia un missatge que informa de l'estat, que es troba a la part inferior de la pantalla de configuració Data Source MySQL.

Un cop configurada la connexió amb la base de dades i es desitja fer canvis dels taulers de visualització o "Dashboards" cal haver iniciat sessió amb un usuari amb drets d'administració i modificació registrat a l'aplicació Grafana del mateix servidor de la UdG.

Un cop s'ha iniciat sessió, cal seleccionar el panell que es desitgi modificar a través del desplegable "Dashboards" com es veu a la figura a continuació i seleccionar "Home", seguidament se selecciona el panell anomenat Aparcament_UdG.

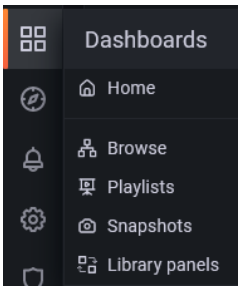

Figura 55 Dashboard Home

Llavors, només fa falta seleccionar l'opció "Edit" d'algun element, com per exemple el de la Figura 56 Editar Estat aparcament.

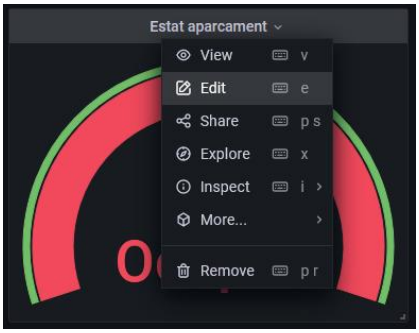

Figura 56 Editar Estat aparcament

Els aspectes més importants de la parametrització són la base de dades i el tipus de panell que es vol utilitzar com el gràfic, taula, etc. Tot seguit es veu la parametrització de la figura anterior, una vegada dins el menú d'edició s'ha d'observar la configuració de la part inferior on es parametritzen les variables necessàries per extreure la informació usant el "Data Source" creat.

El llenguatge és SQL, cal seleccionar la taula de la base de dades, la variable "Time" i a partir d'aquí seleccionar la informació que es vol conèixer.

A la Figura 57 Configuració Estat aparcament es mostren els paràmetres corresponents, als annexos consten tots els codis fets servir del panell Aparcament_UdG.

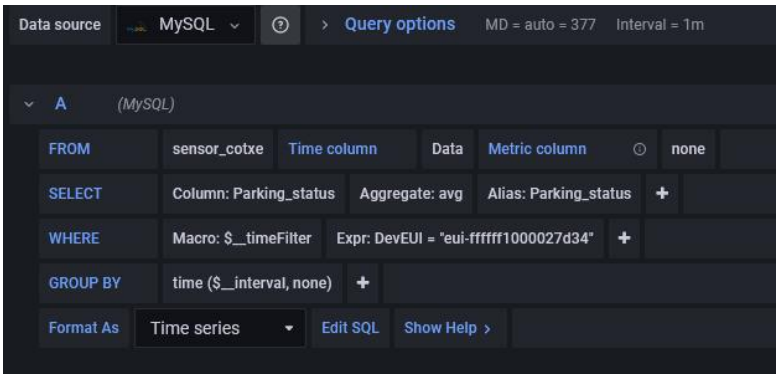

Figura 57 Configuració Estat aparcament

Si es desitja canviar el tipus de gràfica, estil, nombre de variables a mostrar, etc. Cal modificar els paràmetres de la part dreta del menú d'edició, a la següent figura es mostra un exemple.

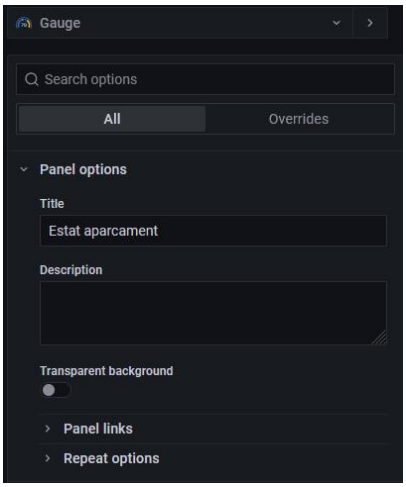

Figura 58 Edició Gauge

6. ESTUDIS I PROVES

El següent capítol mostra els estudis teòrics i pràctics que determinen els aspectes claus, com el consum o l'abast de les comunicacions, així com les solucions adoptades.

6.1. Limitacions Gateway MultiTech

La gateway seleccionada és de 8 canals, aquest fet comporta l'escolta simultània de fins a 8 dispositius (o enviament). És poc probable que aquest fet succeeixi, però s'ha de tenir en compte a partir d'una certa quantitat d'aparells i l'ús que es desitja com la classe, de cicle de retransmissió, "Srpead factor", etc.

MultiTech és coneixedora d'aquest succés i dota les seves gateways de FPGA's que garanteixen una rapidesa de processament excepcional, si això no és suficient el mateix programari de LoRaWAN (a través de les gateways) permet reestructurar els temps d'enviament del sensor, envia un Downlink al dispositiu que retarda un temps petit (mil·lisegons) la següent transmissió.

Les comunicacions i alimentació de la gateway es troben molt per sobre de les especificacions bàsiques necessàries, com són la potència d'alimentació i la velocitat d'internet disponible.

6.2. Limitacions sensor d'aparcament IOTSens

El sensor d'aparcament és seleccionat de l'empresa IOTSens, per aquest motiu i no perdre la garantia del dispositiu, les proves realitzades són purament pràctiques i dins un entorn totalment controlat (departament eXiT i diferents aparcament com l'edifici P.IV de l'EPS, etc).

Les proves realitzades durant la calibració del dispositiu han exposat una casuística amb la calibració que la mateixa empresa no sabia quan se la va informar. Tot i qualificar-lo com a casuístic, no afecta el seu funcionament final, aquest és el següent:

El manual del fabricant especifica detalladament els passos a seguir per calibrar el dispositiu, en concret demana muntar el dispositiu abans de la calibració, un cop calibrat si es decideix canviar de lloc el dispositiu, aquest deixa de transmetre la informació per LoRa, ni tan sols envia un missatge d'error, això si, cada sis hores continua enviant un missatge per saber que "funciona". Amb la informació que s'ha obtingut del sensor no hi ha motiu pel qual no hauria de deixar de transmetre, tot i que es pot intuir que depèn de la característica intrínseca del mateix sensor. Perquè torni a funcionar correctament obliga a realitzar una nova calibració.

Les proves realitzades han determinat que el rang el qual falla és de 30 centímetres a la rodona del punt de calibració inicial i sempre orientat sobre el mateix pla. Donar la volta al dispositiu (cas hipotètic de canvi de les bateries) pot arribar a comportar aquest problema. Finalment, tornar el dispositiu al lloc d'inici no ho soluciona, l'única manera és tornar a calibrar el dispositiu.

6.2.1. Detecció de vehicles

S'han realitzat proves de detecció amb aquest sensor, com el manual indica que el sensor ha d'estar instal·lat a dos cinquenes parts de la plaça d'aparcament es contemplen dos escenaris, quan el cotxe aparca amb el bloc motor o el maleter. La prova s'ha dut a terme amb els models de cotxes, Mazda 3, Suzuki Jimny i Toyota Avensis sobre un terreny pla.

Els dos escenaris han contemplat 10 cops l'entrada i retirada de cada vehicle, resultant amb 40 senyals totals per cada vehicle, 20 senyals indicant de la plaça ocupada i 20 de la retirada d'aquest.

La diferència amb l'altura de detecció entre els vehicles utilitzats ha creat un entorn idoni i realista. Llavors, les proves han mostrat la gran capacitat del sensor per detectar des d'utilitaris a cotxes de muntanya (alts), també s'ha fet aflorar la casuística de l'atenuació dràstica del senyal a causa del vehicle estacionat.

6.3. Estudi prototip indicador lluminós

L'indicador lluminós és un prototip dissenyat al departament i per això s'ha establert determinar la viabilitat amb els requisits de comunicació LoRa, determinar la millor alimentació (bateria o externa) i rang de recepció o enviament de les dades. Els següents apartats mostren els resultats del consum i el rang, finalment es determinen les possibles solucions a adoptar.

6.3.1. Consum

El consum de la placa ESP32U-01 de EZSBC i el RFM95W durant el mode "Deep Sleep" és de 0,012 mA i 0,001 mA respectivament. Quan el conjunt, transmet informació genera un pic de 35 mA just a l'inici i la resta del temps a 9 mA amb el led apagat.

Els temps que el dispositiu roman despert un cop s'ha unit a la xarxa (JOIN acceptat) és una mitjana de 6 segons i 52 segons que dorm. Llavors el cicle total de l'actuador és la suma dels dos temps i dóna un resultat de 58 segons. Recordar la necessitat de resposta en menys d'un minut.

Per calcular el consum mig teòric es reparteixen els consums segons la seva durada i es divideix pel total, l'equació resultant és la següent.

$$I_T = \frac{I_P * T_P + I_N * T_N + I_D * T_D}{T_T} \qquad \text{(Eq. 1)}$$

On: I_T és el consum mig, I_P el pic de consum durant la transmissió, I_N consum normal de la transmissió, I_D consum amb el dispositiu dormint "Deep Sleep". T_T és el cicle del dispositiu, T_P duració del pic de consum, T_N duració del consum normal durant la transmissió i T_D és el temps que es passa dormint. I tots els temps en segons i consums en mA.

Perquè el resultat sigui coherent amb els temps s'ha de complir l'equació següent de temps total de transmissió en aquest cas, $T_A = 6$ segons, mesurat prèviament.

$$T_A = T_P + T_N = 0{,}1 + 5{,}9 = 6 \quad \text{(Eq. 2)}$$

I el total dels temps és un sumatori de tots els temps utilitzats a la primera equació, $T_T = 58$ segons.

$$T_T = T_P + T_A = 6 + 52 = 58 \text{ s} \quad \text{(Eq. 3)}$$

Realitzant les proves les dades dels temps són, $T_P = 0{,}1$s i $T_N = 5{,}9$s donant un resultat de consum mig de $I_T = 0{,}85$ mA. Si es realitza el mateix exercici amb la il·luminació activada, afegeix 1 mA tot el cicle, el resultat final és de $I_T = 1{,}85$ mA. Es determina que el 25 % dels cicles el led es troba activat constantment, el sistema només ha de respondre 12 hores al dia (de 8 a 20 h per exemple), el 25 % de 12 són 3 hores, llavors la proporció és de 12,5 %, total del dia, per saber el consum mitjà d'aquests dos casos s'utilitza l'equació següent.

$$I_M = I_S \times a + I_L \times b = 1{,}85 \times 0{,}125 + 0{,}987 \times 0{,}875 = 1{,}09 \text{ mA} \quad \text{(Eq. 4)}$$

On: I_M és el consum mig dels dos casos, I_S és el consum amb led activat, I_L consum amb led desactivat, "a" és la proporció de temps (tant per 1) amb led activat i "b" la proporció amb led sense activar.

El resultat final és $I_M = 1{,}09$ mA, un cop tenim un model de consum només cal trobar el temps (en hores) que podrà operar, per aquest motiu s'utilitza la següent equació que divideix la capacitat de la bateria pel consum.

$$T_F = \frac{C}{I_M} \quad \text{(Eq. 5)}$$

On: T_F són les hores totals, C és la capacitat de la bateria en mAh i I_M el consum mitjà total.

Si la capacitat total està formada per dos bateries de 2.600 mAh, el temps operatiu dona T_F = 4.770,64 hores que en dies es tradueix a uns 198,77 dies d'autonomia des de la seva activació.

6.3.2. Rang teòric

El rang és un aspecte important que cal tenir en compte en comunicacions sense fils, aquest estudi es realitza en dues parts, un primer estudi teòric i finalment una prova pràctica.

L'estudi teòric es fa respecte a l'enviament de Downlinks per part de la gateway direcció l'actuador lumínic.

Primer de tot cal tenir present la següent equació, la qual ens permet calcular la propagació de les comunicacions. Sense objectes intermediaris i amb línia directa de visió.

$$P_R=P_T(\frac{\lambda}{4\pi d})^2 G_T G_R \qquad \text{(Eq. 6)}$$

On: P_R és la potència mínima necessària a rebre (Actuador, mW), relacionat amb la sensibilitat de l'antena. P_T és la potència de transmissió (Gateway, mW), λ és la longitud d'ona de la llum (metres), "d" la distància teòrica, G_T és el guany de l'antena transmissor (Gateway), expressada com a guany de potència de l'antena isotròpica en "dBi" i G_R igual però de l'antena receptora (Actuador).

Abans cal realitzar els càlculs previs per calcular els guanys i potències, Les següents fórmules realitzen el càlcul necessari, les primeres mostren els resultats dels guanys de cada antena.

Guany de l'antena de la Gateway, utilitzant 3,5 dBi de potència isotròpica (Omnidireccional).

$$3{,}5\ \text{dBi} = 10\log(G_T) \rightarrow G_T = 2{,}24 \qquad \text{(Eq. 7)}$$

Guany de l'antena de l'actuador, utilitzant 0 dBi, completament isotròpica.

$$0\ \text{dBi} = 10\log(G_R) \rightarrow G_R = 1 \qquad \text{(Eq. 8)}$$

Llavors, la potència de transmissió és de la Gateway a 20 mW i només fa falta saber la de l'actuador, la següent equació utilitza la sensibilitat de l'actuador per calcular la potència mínima necessària per rebre la transmissió. L'actuador té una sensibilitat de -148 dBm pròpia del xip SX1726 de Semtech del RFM95W. A la sensibilitat s'afegeix un coeficient de soroll que fa baixar la sensibilitat un 20 dBm, llavors la resultant queda -128 dBm.

$$-148\ \text{dBm} = 10\log\left(\frac{P_R}{1\ \text{mW}}\right) \rightarrow P_R = 0{,}158\ \text{pW} \qquad \text{(Eq. 9)}$$

Abans d'aplicar l'equació principal fa falta calcular la longitud d'ona de la llum, utilitzant la següent equació es mostra el seu resultat.

$$\lambda = \frac{c}{f} = \frac{3\times10^8}{2{,}4\times10^9} = 0{,}125\ \text{m} \qquad \text{(Eq. 10)}$$

Aplicant els valors a la primera equació i aïllant la distància resultant dona el resultat de la següent equació.

$$d = \frac{\lambda}{2\pi}\sqrt{\frac{P_T}{P_R}G_T G_R} = 334.479{,}23\ \text{m} \approx 334\ \text{km} \qquad \text{(Eq. 11)}$$

Aquest resultat pot variar dràsticament degut a totes les infraestructures i altres factors atenuen la potència transmesa. El resultat teòric mostra una de les raons per les quals actualment la tecnologia LoRa està realitzant proves amb satèl·lits.

Si s'executa el mateix estudi a la inversa, serveix per saber el rang teòric que pot enviar dades aquest actuador, com l'estat del led. El resultat es mostra a la següent equació, la potència (10 mA) que utilitza l'actuador, la resta no varien, senzillament els transmissors passen a ser receptors i viceversa.

$$d = \frac{\lambda}{2\pi}\sqrt{\frac{P_T}{P_R}G_T G_R} = 236.878{,}48\ \text{m} \approx 236\ \text{km} \qquad \text{(Eq. 12)}$$

Aquest resultat mostra una disminució important deguda principalment a la disminució de la potència transmesa. Aquesta reducció és la causa d'haver prioritzat el baix consum de l'actuador.

6.3.3. Rang pràctic

Contrastar els càlculs realitzats a l'apartat anterior per avaluar millor la seva industrialització és clau, per aquest motiu s'ha realitzat un petit estudi pràctic de rang a diferents punts del campus i la ciutat de Girona, realitzant mesures d'enviament de dades a dos radis d'actuació en un total d'11 zones, 5 properes i 6 llunyanes o amb molts obstacles.

Les zones d'adquisició de les dades de les 5 zones més properes són: EPS Edificis P.IV i P.I, la facultat de dret, l'institut de Montilivi i l'estadi municipal de Montilivi.

Les 6 zones més allunyades són: Barri residencial C/ Puig de Montilivi, complex esportiu de Palau-sacosta, residència universitària campus de Montilivi, pavelló municipal Palau 2, parc Científic i Tecnològic i la comandància de la Guàrdia Civil de Girona.

Totes les dades han estat adquirides a l'exterior, proper a zones d'estacionament de vehicles i a una alçada d'un metre, per garantir un mínim de semblança amb l'aplicació desitjada. La distància aproximada (en línia directa) des dels punts d'adquisició a la gateway es troben a la següent taula.

Senyals	
Ubicació	Dist. (m)
EPS Edifici IV	50
EPS Edifici I	125
Facultat de Dret	125
Institut Montilivi	200
Estadi Municipal de Montilivi	350
Residencial C/ Puig de Montilivi	500
Parc Científic i Tecnològic	700
Complex Esportiu de Palau-sacosta	850
Residència Universitària Campus de Montiliv	1000
Pavellò Municipal Palau 2	1100
Comandància de la Guàrdia Civil de Girona	1500

Taula 2 Distància del senyal respecte la gateway

Els paràmetres captats per aquest estudi són dos, el SNR (Signal to Noise Ratio) i RSSI (Received Signal Strength Indicator), aquests indicadors són avaluats independentment i finalment es realitza una comparativa general entre les diferents zones, aportants dades i resultats tècnics.

El RSSI és adquirit per determinar la força (o potència) del senyal rebut per la Gateway, aquest senyal es troba expressat en dBm. Com més proper a 0 dBm, major és el senyal rebut, on 0 dBm és un senyal ideal. D'aquesta manera es pot saber la cobertura de les zones on es produeixen les transmissions. Els rangs d'operació del LoRa es consideren entre -30 i -137 dBm teòrics.

El SNR és la relació entre la potència del senyal rebut i la potència del soroll captat per la gateway, aquests valors es troben expressats en decibels (dB). Indiquen principalment si el senyal es troba per sobre o sota de la potència del soroll captat, on un nombre positiu indica que la potència del senyal rebut és superior a la del soroll i un nombre negatiu a l'invers. Els

rangs teòrics de LoRa permeten valors negatius i el rang total està establert com el següent, -20 ÷ +10 dB. S'utilitza principalment per saber la qualitat del senyal o la capacitat de la tecnologia (LoRa) per condicionar-lo i aconseguir les dades transmeses.

La condició del senyal de LoRa programat i usat és amb un SF7 i un BW125 kbps, on el "Spread Factor" seleccionat per LoRa, comporta una SNR límit de -7,5 dB teòric i juntament amb l'amplada de banda seleccionada queda una sensibilitat reflectida a la següent equació, que es pot interpretar com el RSSI límit teòric.

$$S = -174 + 10 \times \log_{10}(BW) + NF + SNR_{limit} \quad \text{(Eq. 13)}$$

On: S és la sensibilitat del receptor, la gateway, -174 dB és estàndard de les comunicacions LoRa, l'amplada de banda (BW) es troba en Hz, NF és l'anomenada "Noise Figure", depèn del dispositiu utilitzat, en aquest cap el RFM95 utilitza el model SX1276 de Semtech, això comporta un valor de 6 dB, finalment el límit SNR, amb valor de -7,5 dB.

El resultat dóna la següent sensibilitat, S = -125 dBm teòrics, llavors es pot interpretar que un RSSI amb valors inferiors a aquesta sensibilitat no són possibles i valors propers indiquen una cobertura molt dolenta.

La següent taula mostra el criteri utilitzat per aquest estudi segons el rang de valors acceptats per les comunicacions LoRa i més pràctics segons el RSSI.

RSSI (dBm)	Criteri RSSI
> -70	Excel·lent
-70 > -90	Molt bona
-90 > -100	Bona
-100 > -110	Mediocre
< -110	Dolenta

Taula 3 Criteri RSSI

La Taula 4 mostra el discerniment dels senyals rebuts segons el seu valor pràctic.

SNR (dB)	Criteri SNR
> 10	Excel·lent
10 > 6	Molt bona
6 > 2	Bona
2 > -4	Mediocre
< -4	Dolenta

Taula 4 Criteri SNR

A cada zona s'ha adquirit un mínim de 10 dades per garantir un mínim de coherència amb els resultats, però algunes zones tenen més informació.

Abans de comentar els resultats cal aclarir que l'antena de la Gateway es troba perpendicular al terra tal com s'explica a l'apartat 3.3, ja que influeix directament amb els resultats obtinguts.

Per l'estudi de les dades s'han utilitzat diagrames de caixa, ja que permeten visualitzar la dispersió i la simetria dels senyals rebuts. Afavoreix una major visualització i comprensió dels resultats, a cada diagrama es pot visualitzar els valors mínims, màxims, quartils 1, 2 (mediana) i 3 i finalment, la mitjana (marcat per una "x").

A continuació s'expressen els resultats obtinguts de l'indicador RSSI de les zones més properes que es mostren a la Figura 59. Es fa una avaluació conjunta de la zona i s'extreuen els resultats més característics. Cal remarcar que tots els diagrames mostren els resultats de forma ascendent (direcció d'esquerra a dreta) amb la distància respecte a la Gateway (més propers a més llunyans).

Totes les dades mostren una tendència a empitjorar la cobertura a mesura que les dades enviades són cada cop més lluny de l'antena, ja que l'edifici P.IV de l'EPS és la zona més propera i indica una mediana de -63,5 dBm del senyal (molt bó) i la pitjor és la més llunyana, estadi municipal de Montilivi amb una mediana del senyal mediocre amb -101 dBm.

Així i tot, es pot observar una petita variació entre els resultats obtinguts de les zones de facultat de Dret i l'institut de Montilivi, aquest resultat indica que la facultat de dret, tot i trobar-se més propera té més obstacles que atenuen el senyal i empitjoren la cobertura. També es pot interpretar que no hi ha relació directa la distància i la variància d'aquest indicador, tot i que a l'estadi mostra una variància màxima de -19 dBm com a la pitjor i més llunyana, les dades amb menys variància han estat assolides a l'edifici P.I de l'EPS i l'institut de Montilivi, -3 i -4 dBm respectivament. Aquestes dues zones representen la segona i quarta posició respecta la distància màxima fins a la gateway.

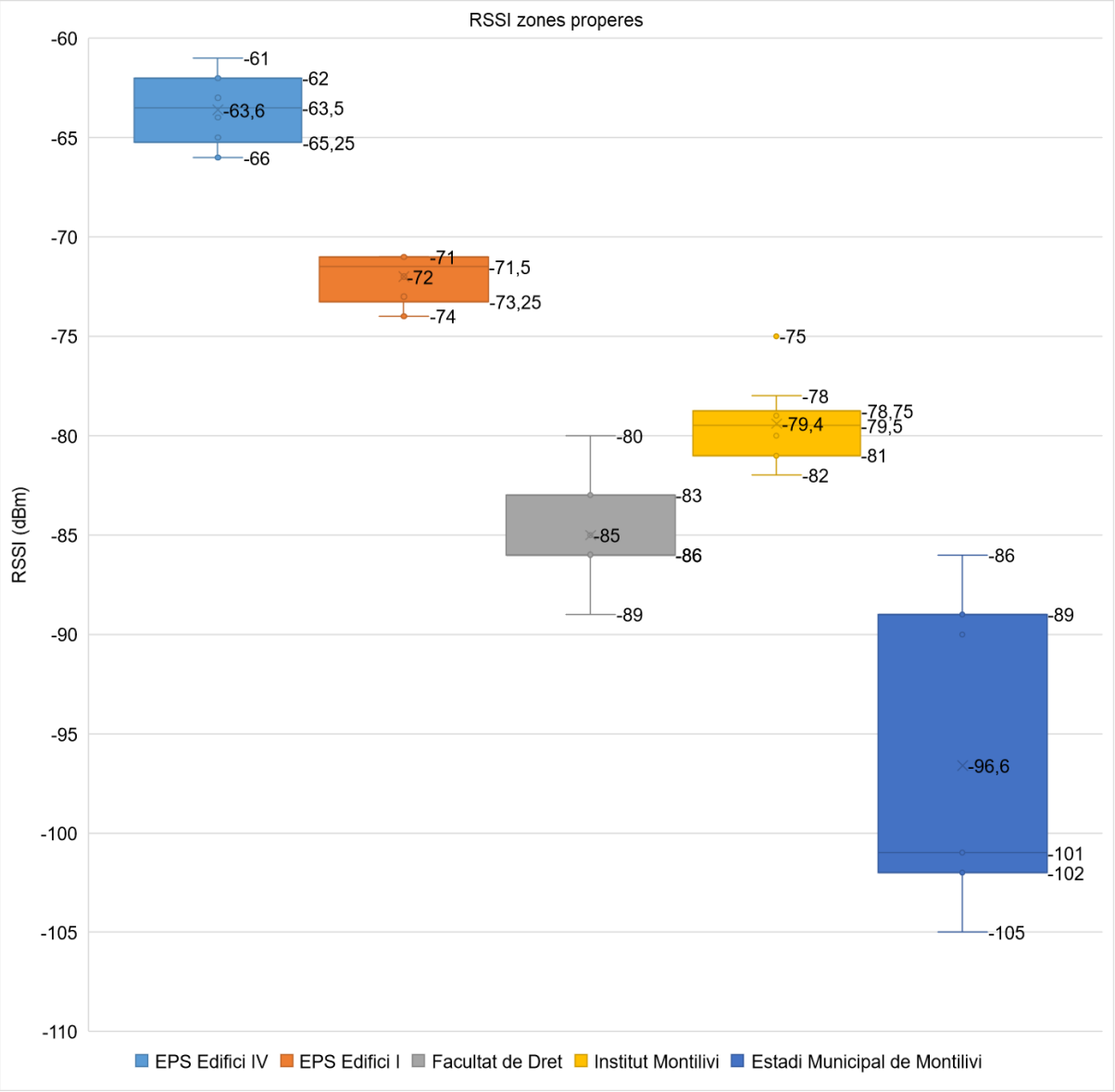

Figura 59 RSSI Proper

A les zones més allunyades que es veuen a la Figura 60 es pot observar el canvi de tendència respecte a la força del senyal, la distància ja no té un paper molt rellevant i els obstacles passen a dominar el RSSI. Una excepció apareix amb els resultats del parc científic, tot i tenir una muntanya, vegetació i edificis entremig, el RSSI es considera com a molt bo o bo amb una mediana de -67 dBm.

Entre les zones del complex esportiu de Palau-sacosta, el pavelló municipal Palau 2 i la comandància de la Guàrdia Civil destaca la mateixa força del senyal, mediocre o dolenta, tot i que cada ubicació es troba més lluny que l'anterior. Això indica una vegada més que els obstacles representen el principal factor d'empitjorament del senyal. El pitjor resultat quan a variància és la residència universitària del campus de Montilivi aquesta pot ser deguda als rebots del senyal provocat pels obstacles empitjorant la cobertura.

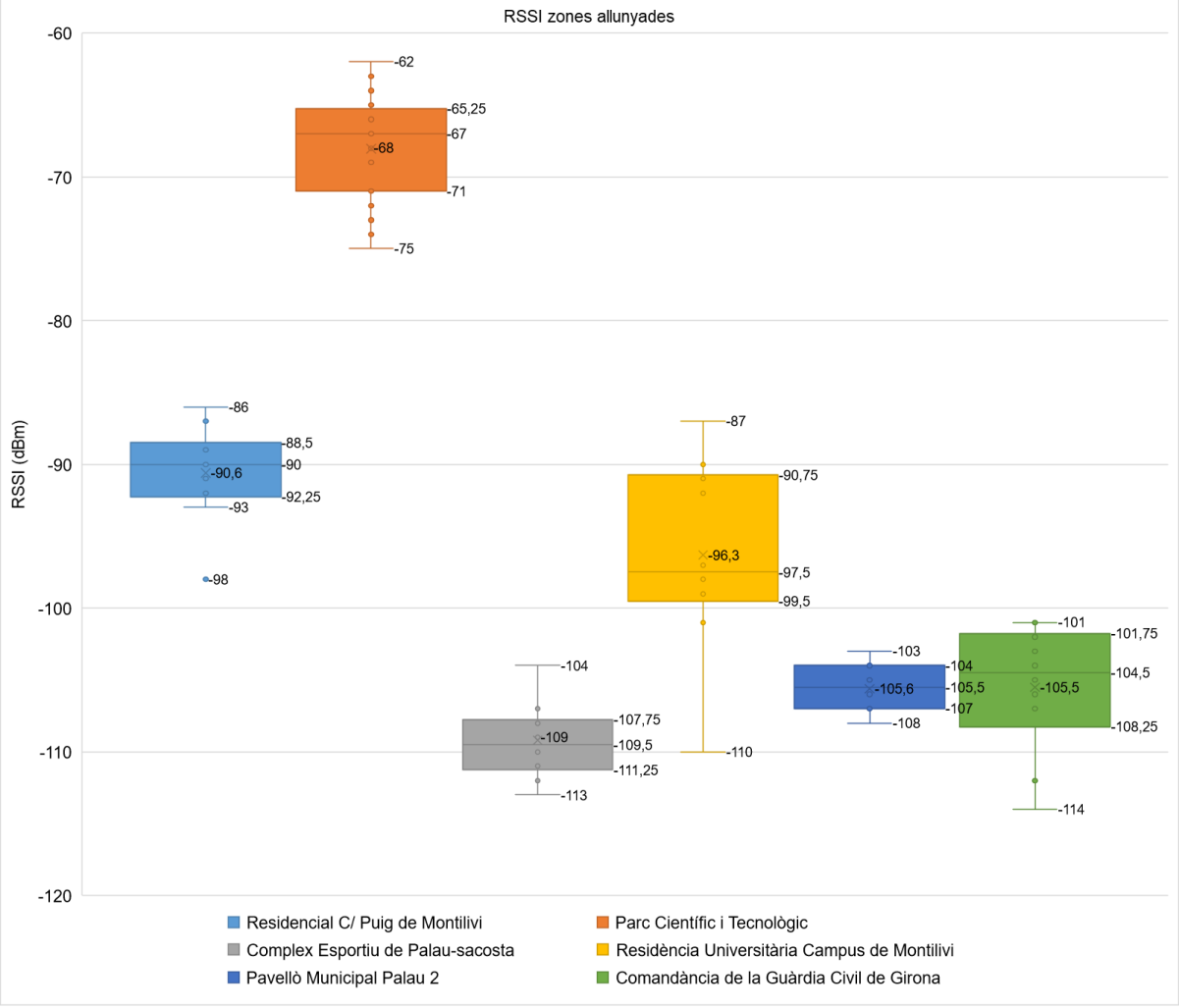

Figura 60 RSSI Llunyà

Un cop avaluats els resultats del RSSI s'assenyalen els resultats del SNR, que regulen la qualitat del senyal, a la Figura 61 es poden observar els valors. Els resultats mostren que tots els senyals són aptes i no se n'han perdut, on la millor qualitat del senyal és la mesurada a l'edifici P.I de l'EPS, no només pels seus valors alts, també per la seva poca variabilitat entre ells. La pitjor degut a la seva gran variància entre dades és a l'estadi municipal de Montilivi.

La resta de les dades obtingudes a les zones de l'edifici P.I, facultat de dret i l'institut de Montilivi són semblants, demostren que la distància per sota dels 400 metres no malmeten dràsticament la qualitat del senyal, sempre que no es trobin amb molts d'obstacles, especialment els metàl·lics, tal com es pot observar a l'estadi on la gran quantitat d'estructures metàl·liques generen un impacte molt significatiu amb la variabilitat de les dades.

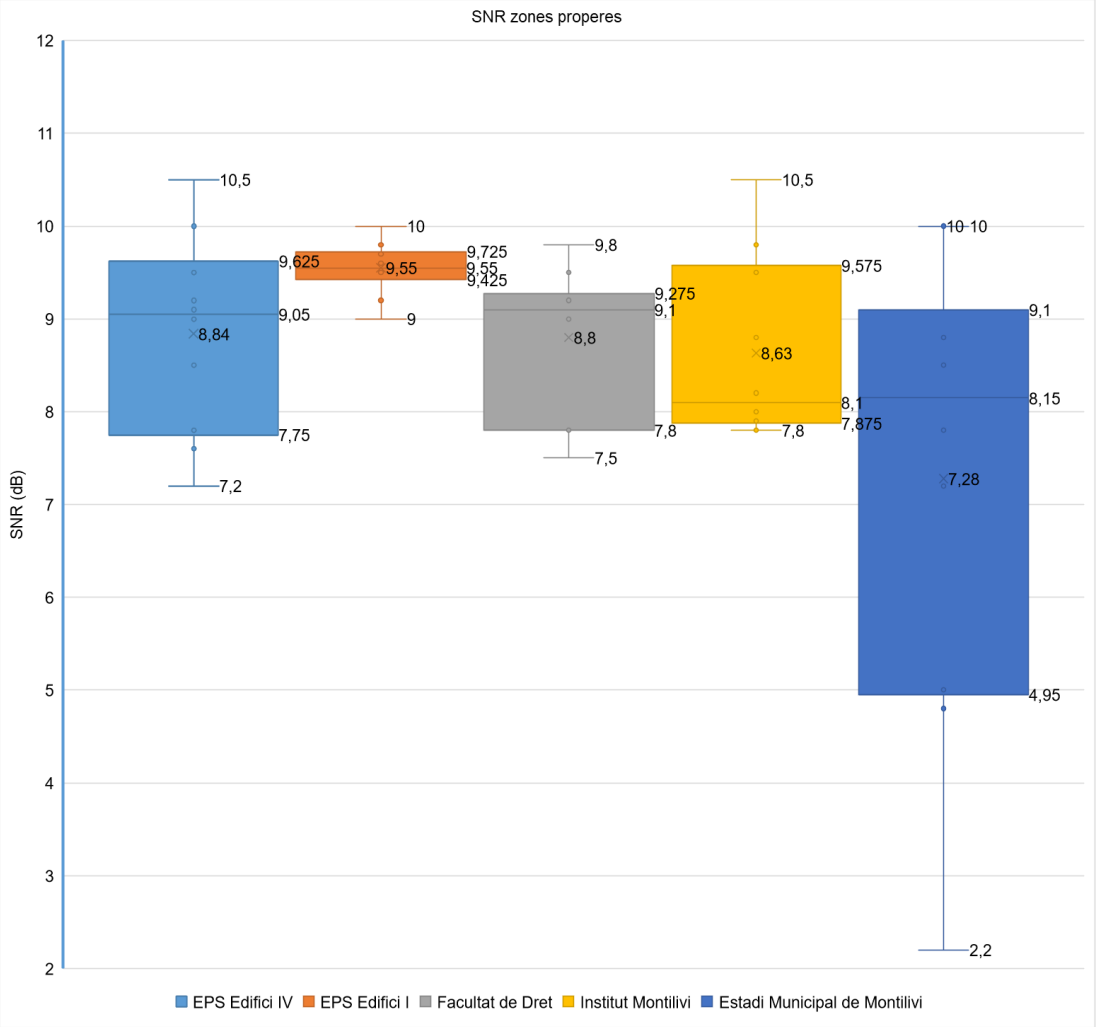

Figura 61 SNR Proper

Els valors obtinguts a la Figura 62 mostren uns resultats increïbles a la zona del parc científic on la mediana és de 12,875 dB. Aquests valors poden ser causats per la inclinació del terreny i l'àrea omnidireccional de l'antena de la gateway.

Per altra banda, es mostra una increïble variància a les zones més llunyanes, Palau-sacosta, la residència universitària, Palau 2 i la comandància de la Guardia Civil, on aquests valors oscil·len entre els 4,3 dB i 11,3 dB. A la residència cal destacar un valor atípic de -3,8 i fet que genera una desconfiança dels resultats obtinguts (valors SNR) a aquesta zona.

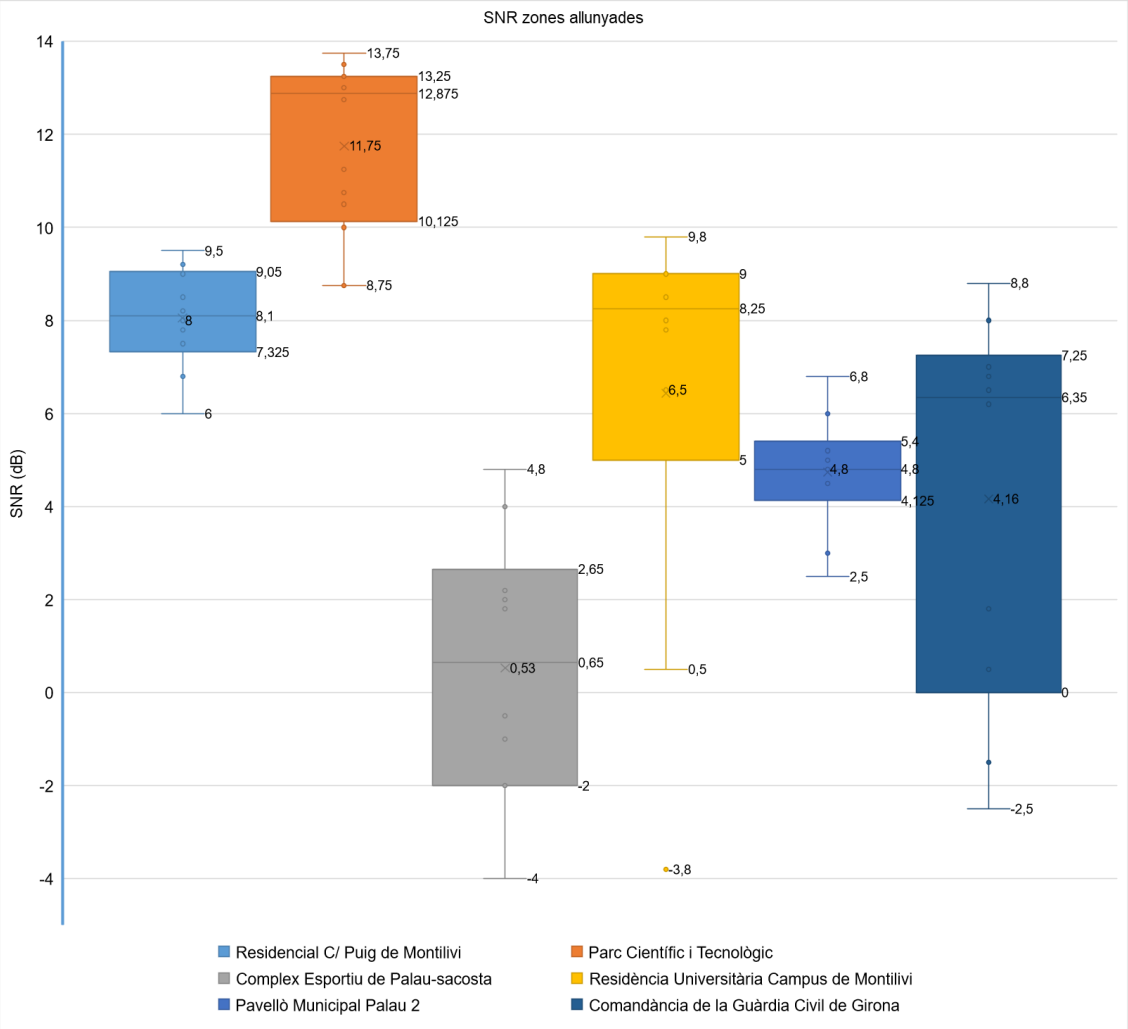

Figura 62 SNR Llunyà

6.3.4. Solució

La integració de les comunicacions LoRa no comporta cap problema, però el consum representa el principal obstacle per aquest dispositiu. Requerir bateries amb capacitats extremadament altes (+ 10 Ah) o gran quantitat d'aquestes és imprescindible, fet que incrementa el cost dràsticament.

Per aquest motiu el prototip compta amb dues possibilitats, la primera és acceptar el cost de les bateries, les quals es recomana una capacitat total d'entre 30 i 40 Ah per garantir un mínim de 3 a 4 anys de funcionament. Per altra banda, es pot separar l'alimentació de la il·luminació amb un nou prototip que contempli l'activació d'un circuit de potència alimentada per la xarxa d'electricitat de la ciutat de Girona, però no es contempla en aquest projecte.

Finalment, cal esmentar que la possibilitat d'alimentació externa a través de plaques solars, com "Solar harvesting", no es considera factible, ja que la instal·lació d'aquests dispositius són a una altura relativament baixa i qualsevol objecte proper pot fer baixar dràsticament el rendiment de la placa, com cotxes estacionats, persones, edificis, arbres, etc.

7. PROGRAMARI

Aquest capítol explica les funcions i característiques dels programes que engloben tot el sistema. L'explicació del programari tracta de donar resposta a les incògnites funcionals del projecte i la lògica implementada.

7.1. Indicador lluminós

El codi de l'indicador lluminós es descriu en aquest capítol per les diferents seccions de programa de les quals està format utilitzant el llenguatge C++. Primer de tot s'explica de manera breu el funcionament a través de les dues funcions principals "setup()" i "loop()" i llavors s'incideix en cadascuna de les funcions que les integren.

La funció "setup()", aquí es troben les parametritzacions generals del dispositiu, primer de tot es configura la velocitat del microprocessador a 10 MHz per reduir el consum amb la funció "setCpuFrequencyMhz(10)" i es deixa 10 ms perquè es duguin a terme els canvis necessaris. Llavors, s'inicialitza el bus "Serial", sempre que aconsegueixi connectar-se a un ordinador, a la velocitat de 115.200 bauds, llavors, es mostra el missatge "Starting". Per la configuració del LED com a pin RTC s'utilitzen les funcions "rtc_gpio_init(LED_VERMELL)" per iniciar el seu registre i "rtc_gpio_set_direcction(LED_VERMELL, RTC_GPIO_MODE_OUTPUT_ONLY)" que determina l'ús com a sortida del pin "LED_VERMELL" amb el mode "RTC_GPIO_MODE_OUTPUT_ONLY".

A partir d'aquest moment s'executen les funcions "os_init()", que inicia la radiofreqüència LoRa i la "LMIC_reset()", encarregada de realitzar una primera inspecció al RFM95W, ambdues funcions de la llibreria LMIC.

Saber si el dispositiu prové del "Deep Sleep" és imprescindible, per això es revisa la informació de la variable "RTC_LMIC.seqnoUp", explicada a la funció "LoRaWANDebug()". Si és diferent de zero, llavors s'executa la funció "LoadLMICFromRTC()" per recuperar la configuració de la

iteració anterior. Per saber l'estat, en cas de realització de proves, es crida la funció "LoRaWANDebug(LMIC)".

Finalment, s'executa la funció "do_send()" amb el punter "&sendjob" per determinar la informació que s'enviarà durant la següent transmissió.

```
void setup() {
    setCpuFrequencyMhz(10); // Millora de 80 mA a 30 mA
    delay(10);

    Serial.begin(115200);
    Serial.println(F("Starting"));

    //Configurar el pin LED a la memòria RTC
    rtc_gpio_init(LED_Vermell); //Inicialitzem el PIN GPIO RTC
    rtc_gpio_set_direction(LED_Vermell, RTC_GPIO_MODE_OUTPUT_ONLY); //Definim el mode
RTC del PIN

    // LMIC init
    os_init();
    // Reset MAC
    LMIC_reset();

    //Revisem si cal volcar la informació guardada dins la memòria RTC
    if (RTC_LMIC.seqnoUp != 0)
    {
        LoadLMICFromRTC();
    }
    //S'executa per la realització de proves i saber el seu estat
    LoraWANDebug(LMIC);

    // Start OTTA
    do_send(&sendjob);

}
```

A la funció "loop()", s'hi executa la funció "os_runloop_once()", aquesta inicia les comunicacions. Té en compte si fa falta començar l'activació del sensor dins la xarxa LoRa "JOINING" i executar l'enviament i recepció de les dades (la funció "onEvent"). A part es defineix la variable "lastPrintTime" com a estàtica, sense signe i "long" i servirà per imprimir la duració dels cicles de la funció "loop()". També es declara la constant booleana "timeCriticalJobs" com a resultat de la funció "os_queryTimerCrtitcalJobs", aquesta funció de la llibreria LMIC retorna un u ("true") si hi ha tasques a realitzar dins del temps predeterminat de "TX_INTERVAL".

Finalment, s'utilitzen les variables "timeCriticalJobs", "GOTO_DEEPSLEEP", "LMIC.opmode" i "OP_TXRXPEND" pels següents propòsits, el primer és posar a dormir el dispositiu, per això el condicional "if" executa les funcions "LoraWANPrintLMICOpmode()", "SaveLMICToRTC()" i "GoDeepSleep()", sempre que es compleixi que no hi hagi cap tasca per executar "!timeCriticalJobs", que la variable "GOTO_DEEPSLEEP=true" i no quedi cap "Flag" activat i obertura "TX" i "RX" per enviar o escoltar "!(LMIC.opmode & OP_TXRXPEND)". Si no, el segon cas "else if" imprimeix el temps cada dos segons amb les funcions "LoraWANPrintLMICOpmode()" i "PrintRuntime()", acabant amb l'actualització de la variable "lastPrintTime" amb els "millis()" que mostra un missatge indicant que no pot anar a dormir.

```
void loop() {

    static unsigned long lastPrintTime = 0;

    //Funció LoRa
    os_runloop_once();

    //Funció Intermitent que revisa si cal anar a dormir
                        const           bool            timeCriticalJobs           =
os_queryTimeCriticalJobs(ms2osticksRound((TX_INTERVAL * 1000)));
    if (!timeCriticalJobs && GOTO_DEEPSLEEP == true && !(LMIC.opmode & OP_TXRXPEND))
    {
        Serial.print(F("Anem a dormir "));
        LoraWANPrintLMICOpmode();
        SaveLMICToRTC(TX_INTERVAL);
        GoDeepSleep();
    }
```

```
    else if (lastPrintTime + 2000 < millis())
    {
        Serial.print(F("No pot dormir "));
        Serial.print(F("TimeCriticalJobs: "));
        Serial.print(timeCriticalJobs);
        Serial.print(" ");

        LoraWANPrintLMICOpmode();
        PrintRuntime();
        lastPrintTime = millis();
    }
}
```

A continuació s'expliquen les llibreries usades, aquestes són del paquet "MCCI LoRaWAN LMIC library" d'IBM, la qual controla les comunicacions amb la xarxa LoRa i el dispositiu (lmic.h i hal.h). La comunicació SPI integrada (SPI.h), controla les comunicacions i missatges enviats entre la placa EZSBC i el RFM95W. A continuació es pot visualitzar el codi amb la llibreria que permet realitzar la programació amb entorn Platformio de VSCode, la "Arduino.h" i la crida "#include" de les llibreries usades d'IBM.

```
// Llibreria PlatformIO
#include <Arduino.h>

//Llibreries per LoRa
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>
```

Per últim, amb les llibreries usades, la "driver/rtc_io.h" permet la configuració dels pins que necessitem mantenir activats un cop el microprocessador es posa a "Deep Sleep". El seu ús principal és pel LED, a continuació es veu el codi.

```
//Llibreria RTC
#include <driver/rtc_io.h>
```

Com l'embedded es posa en mode "Deep Sleep" és necessari habilitar la memòria RTC on guardar la informació necessària de les comunicacions, d'aquesta manera s'evita executar la funció "JOIN" (Activació) de LoRa cada cop que es desperta. Amb la funció "RTC_DATA_ATTR" s'atribueix la memòria equivalent a la variable "RTC_LMIC". Es copia l'estructura "lmic_t" de la llibreria IBM dins aquesta variable per crear una imatge idèntica i poder-hi guardar la informació correctament. L'estructura "lmic_t" compta amb la informació necessària per al correcte funcionament un cop activat. Finalment, la definició de la variable booleana "GOTO_DEEPSLEEP" que permetrà l'execució de la funció per posar a dormir la placa.

```
//Configuració RTC i Deep Sleep

RTC_DATA_ATTR lmic_t RTC_LMIC;
bool GOTO_DEEPSLEEP = false;
```

La configuració del dispositiu per OTAA és extreta d'un programa d'exemple que proveeix la llibreria LMIC, aquí cal emplenar els paràmetres APPEUI, DEVEUI i APPKEY, aquests paràmetres són expressats amb hexadecimal i de vuit, vuit i setze, caràcters respectivament. Serveixen per a l'activació OTAA del dispositiu i la seva configuració es troba explicada a l'apartat Configuració i calibració actuador lumínic.

Els paràmetres APPEUI, DEVEUI i APPKEY s'introdueixen dins la llista amb atributs "PROGMEM" amb el tipus "u1_t" que significa enter simple (8 bits) sense signe. Tot seguit es volca dins la funció corresponent de la llibreria "os_getArtEUI", "os_getDevEUI" i "os_getDevKey" a través de punters i la variable "buf". Aquesta variable "buf" és l'índex amb què la funció agafa els valors de les variables APPEUI, DEVEUI i APPKEY corresponents.

```
// APPEUI Little-endian
static const u1_t PROGMEM APPEUI[8]={ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}

// DEVEUI Little-endian
static const u1_t PROGMEM DEVEUI[8]={ 0xE2, 0x00, 0x05, 0xD0, 0x7E, 0xD5, 0xB3, 0x70
};
```

```
void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}


// APPKEY Big-endian
static const u1_t PROGMEM APPKEY[16] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
void os_getDevKey (u1_t* buf) {  memcpy_P(buf, APPKEY, 16);}
```

Llavors, la secció "Data types" defineix la llista del "payload" de "uint8_t" dintre la memòria RTC amb "RTC_DATA_ATTR", quatre enters simples (8 bits) sense signe on guardar les dades que es volen enviar i la variable "sendjob", estàtica declarada com a objecte "osjob_t", la qual és s'utilitza com a punter per la llibreria i d'aquesta manera poder executar les funcions, el següent extracte de codi mostra aquesta descripció.

```
// Data types variables
RTC_DATA_ATTR uint8_t payload[4];
static osjob_t sendjob;
```

La següent definició del codi determina la duració del mode "Deep Slepp" en segons, la variable "TX_INTERVAL", és llegida per la funció principal ("void loop") i es troba registrada a la configuració inicial del programa. Seguidament, es pot observar la definició del pin usat per activar el LED, important definir-lo amb el tipus "GPIO_NUM_X" perquè atribueix el tipus de dada "gpio_num_t", necessària per la parametrització de l'estat RTC.

```
// Duty Cycle enviament TX Uplink
const unsigned TX_INTERVAL = 52;

// Variable LLUM
#define LED_Vermell GPIO_NUM_33
```

Cal tenir en compte els intervals de transmissió segons el Payload transmès i la configuració SFx per no incomplir la norma del Duty Cicle. Qualsevol altre valor és d'ús estricte per la realització de proves temporals.

A continuació es mostra el codi de l'objecte "lmic_pinmap" de la llibreria LMIC, el qual defineix els pins segons el seu ús. La variable ".nss" és la selecció del chip a través de la comunicació

SPI, el pin usat és el IO5. El ".rxtx" serveix per controlar el chip SX1706 del RFM95W i no és utilitzat per aquesta llibreria, ni projecte. El ".rst" és l'encarregat d'inicialitzar el chip, correspon al pin 4. Finalment, ".dio" és una llista amb els pins que fa ús els pins 2, 21, 22 i relaciona amb els pins DIO0, DIO1, DIO2 del RFM95W.

```
// Pin mapping EZSBC i RFM95W
const lmic_pinmap lmic_pins = {
    .nss = 5,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 4,
    .dio = {2, 21, 22},
};
```

El programa contempla una conversió de les variables APPEUI, DEVEUI i APPKEY un cop han estat enviades. Permet saber a través del "Serial" si les dades introduïdes han estat convertides i trameses correctament a la xarxa LoRa. Aquesta acció és duta a terme per la funció "printHex2()" representat al següent codi.

```
// Conversió dades
void printHex2(unsigned v) {
    v &= 0xff;
    if (v < 16)
        Serial.print('0');
    Serial.print(v, HEX);
}
```

Per enviar les dades fa falta una funció que sigui executada quan toca, per aquest motiu s'utilitza la funció de la llibreria LMIC "do_send". Si s'executa aquesta funció durant la fase de preparació de les comunicacions, les dades són tractades, en cas contrari mostra un missatge d'error.Empaqueta la variable "payload" dins la funció "LMIC_setTxData2" i la comprimeix, com es veu al següent codi. També remet el contingut de la llista i un missatge de paquet tramès, "Serial.ptinrln(Payload[X])".

```
// Enviament dades
void do_send(osjob_t* j){
    // Check if there is not a current TX/RX job running
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
    } else {

        Serial.println(payload[0]);
        Serial.println(payload[1]);
        Serial.println(payload[2]);
        Serial.println(payload[3]);

        // Preparem les dades per la següent transmisisó TX.
        LMIC_setTxData2(1, payload, sizeof(payload), 0);
        Serial.println(F("Packet queued"));
    }
}
```

La funció "LoraWANPrintLMICOpmode()" permet saber l'estat de les comunicacions durant la realització de les proves. No es mostra el contingut total de la funció, es pot trobar a l'annex corresponent. Utilitza la variable "LMIC.opmode" de la llibreria per saber l'estat dels "Flags" (banderes) de comunicació.

```
// Event que respon segons l'estat de les comunicacions
void LoraWANPrintLMICOpmode(void)
{
    Serial.print(F("LMIC.opmode: "));
    if (LMIC.opmode & OP_NONE)
    {
        Serial.print(F("OP_NONE "));
    }
```

El codi de la següent funció "LoRaWANDebug" queda retallada en diversos paràgrafs, dit això, la feina principal és estructurar una taula on es mostren tots els paràmetres més importants del protocol de comunicacions en què es troba l'Embedded.

La primera part de la funció consisteix en declarar una entrada, "lmic_chek" amb estructura "limc_t" de la llibreria IBM, una vegada creada es crida la funció "LoRaWANPrintLMICOpmode()" perquè mostri els "Flags" i imprimeix els valors resultants.

```
//Funció que avisa de l'estat LoRa
void LoraWANDebug(lmic_t lmic_check)
{
    Serial.println("");
    Serial.println("");

    LoraWANPrintLMICOpmode();
    Serial.println("");
```

Llavors es revisen i imprimeixen pel "Serial" les variables de l'estructura "lmic_check" següents: la "seqnoUp", nombre enter que compta els cicles d'activació de les comunicacions (enviament Uplinks i connexió conseqüent després d'haver dormit "Deep Sleep").

El "globalDutyRate" indica els "Ticks" (cicles de rellotge de l'Embedded) que tarda en enviar el missatge, "globalDutyAvail" anuncia els "Ticks" disponibles restants per complir amb el "Duty Cycle". Ambdós utilitzen la funció "osticks2ms" i són dividides per 1000, d'aquesta manera s'extreuen els "Ticks" de les variables.

```
    Serial.print(F("LMIC.seqnoUp = "));
    Serial.println(lmic_check.seqnoUp);

    Serial.print(F("LMIC.globalDutyRate = "));
    Serial.print(lmic_check.globalDutyRate);
    Serial.print(F(" osTicks, "));
    Serial.print(osticks2ms(lmic_check.globalDutyRate) / 1000);
    Serial.println(F(" sec"));

    Serial.print(F("LMIC.globalDutyAvail = "));
    Serial.print(lmic_check.globalDutyAvail);
    Serial.print(F(" osTicks, "));
```

```
Serial.print(osticks2ms(lmic_check.globalDutyAvail) / 1000);
Serial.println(F(" sec"));
```

La variable "LMICbandplan_nextTx" extreu els segons que ha estat transmetent la modulació LoRa, utilitza la funció "os_getTime" que extreu els microsegons i s'utilitza dins la "osticks2ms".

```
Serial.print(F("LMICbandplan_nextTx = "));
Serial.print(LMICbandplan_nextTx(os_getTime()));
Serial.print(F(" osTicks, "));
Serial.print(osticks2ms(LMICbandplan_nextTx(os_getTime())) / 1000);
Serial.println(F(" sec"));

Serial.print(F("os_getTime = "));
Serial.print(os_getTime());
Serial.print(F(" osTicks, "));
Serial.print(osticks2ms(os_getTime()) / 1000);
Serial.println(F(" sec"));
```

Les variables "lmic_check.txend" i "lmic_check.txChnl" mostren el "Tick" final de la transmissió (Uplink) i el canal de radiofreqüència usat respectivament, són impreses utilitzant el "Serial".

Finalment, es crea la taula final d'informació relativa a les comunicacions, fent servir la primera funció "Serial.println" es creen les columnes amb les variables "Band", franja de comunicació LoRa, "avail", "Ticks" que bloquegen l'ús de l'amplada de banda, "avail_sec", el mateix, però en segons, "lastchnl", últim canal freqüencial utilitzat i "txcap", índex que prioritza el canal usat, major nombre, major prioritat. Utilitzant el bucle "for" s'imprimeix cada línia segons el nombre màxim de bandes usades de l'actuador, registrada a la variable "MAX_BANDS".

```
Serial.print(F("LMIC.txend = "));
Serial.println(lmic_check.txend);
Serial.print(F("LMIC.txChnl = "));
Serial.println(lmic_check.txChnl);

Serial.println(F("Band \tavail \t\tavail_sec\tlastchnl \ttxcap"));
```

```
    for (u1_t bi = 0; bi < MAX_BANDS; bi++)
    {
        Serial.print(bi);
        Serial.print("\t");
        Serial.print(lmic_check.bands[bi].avail);
        Serial.print("\t\t");
        Serial.print(osticks2ms(lmic_check.bands[bi].avail) / 1000);
        Serial.print("\t\t");
        Serial.print(lmic_check.bands[bi].lastchnl);
        Serial.print("\t\t");
        Serial.println(lmic_check.bands[bi].txcap);
    }
    Serial.println("");
    Serial.println("");
}
```

Un possible resultat d'aquesta funció es troba reflectit a la següent figura, la qual mostra els continguts del "Serial", tant de la funció "LoRaWANPrintLMICOpmode()" com la pròpia esmentada.

```
LMIC.opmode: OP_RNDTX OP_NEXTCHNL
LMIC.seqnoUp = 4
LMIC.globalDutyRate = 0 osTicks, 0 sec
LMIC.globalDutyAvail = 0 osTicks, 0 sec
LMICbandplan_nextTx = 0 osTicks, 0 sec
os_getTime = 29389 osTicks, 0 sec
LMIC.txend = 34873
LMIC.txChnl = 2
Band    avail           avail_sec       lastchnl        txcap
0       0               0               3               1000
1       0               0               4               100
2       0               0               15              10
3       0               0               0               0
```

Figura 63 Impressió Serial Debugger

El codi següent mostra la implementació de la funció "PrintRuntime", la seva activitat es resumeix en imprimir els segons totals que ha estat despert l'actuador abans de tornar a l'estat "Deep Sleep". Utilitza la funció "millis()", per saber els mil·lisegons d'ençà que ha despertat i

es divideix per mil, resultant en segons, llavors es volca a la variable "segons" de tipus "long", finalment, s'estructura el missatge.

```
// Funció Per saber el temps d'execució
void PrintRuntime()
{
    long seconds = millis() / 1000;
    Serial.print("Runtime: ");
    Serial.print(seconds);
    Serial.println(" seconds");
}
```

La funció "SaveLMICToRTC()" és l'encarregada de guardar les dades a la memòria RTC i bloquejar les comunicacions durant el temps indicat. La funció contempla una entrada de nombre enter "deepsleep_sec", directament relacionat amb la variable global "TX_INTERVAL". Es guarda la informació de la configuració "LMIC" dins l'estructura "RTC_LMIC" global, llavors, es defineix la variable "now" com a "long" sense signe on guardar els mil·lisegons totals d'execució del programa i finalment es calculen els temps que bloquegen les comunicacions LoRa. Per portar a terme aquesta lògica s'utilitza el condicional "if" que determina, primer de tot si la freqüència programada dins l'arxiu "lmic_project_config.h" és possible per l'espai Europeu gràcies a la definició de "CFG_LMIC_EU_like" de la llibreria "LMIC" d'IBM. En cas contrari es mostra el missatge "No DutyCycle recalculation function!".

El bloqueig de les comunicacions utilitza el bucle "for" per totes les bandes usades, "MAX_BANDS", dins d'aquest es crea la variable "correctedAvail" de tipus "ostime_t", determinat per la llibreria i es calculen els "Ticks" que bloquejaran les bandes segons el següent resultat: La variable "avail" de la banda indexada ("RTC_LMIC-band[i].avail") són els tikcs restants de bloqueig calculats a l'anterior iteració, a aquesta se li resta el resultat dels "Ticks" resultants de la funció actual, variable "now" dividida per mil, més el temps que ha

estat dormint "deepsleep_sec" i convertida a "Ticks" amb la funció "OSTICKS_PER_SEC". Si aquest resultat és negatiu la variable "correctedAvail" és zero, en cas contrari es volca dins la variable "RTC_LMIC.bands[i].avail" per ser usat a la següent execució. Finalment, es fa el mateix per la variable "globalDutyAvail".

```
// Funció per guardar les dades correctes
void SaveLMICToRTC(int deepsleep_sec)
{
    Serial.println(F("Guardant LMIC to RTC"));
    RTC_LMIC = LMIC;

    // Reset DutyCyles
    unsigned long now = millis();

    // EU Like Bands
    #if defined(CFG_LMIC_EU_like)
        Serial.println(F("Reset CFG_LMIC_EU_like band avail"));
        for (int i = 0; i < MAX_BANDS; i++)
        {
             ostime_t correctedAvail = RTC_LMIC.bands[i].avail - ((now / 1000.0 +
deepsleep_sec) * OSTICKS_PER_SEC);
            if (correctedAvail < 0)
            {
                correctedAvail = 0;
            }
            RTC_LMIC.bands[i].avail = correctedAvail;
        }

          RTC_LMIC.globalDutyAvail = RTC_LMIC.globalDutyAvail - ((now / 1000.0 +
deepsleep_sec) * OSTICKS_PER_SEC);
        if (RTC_LMIC.globalDutyAvail < 0)
        {
            RTC_LMIC.globalDutyAvail = 0;
        }
    #else
        Serial.println(F("No DutyCycle recalculation function!"));
```

```
    #endif
}
```

La funció "LoadLMICFromRTC()" és utilitzada per cridar la informació de la memòria RTC, la variable "RTC_LMIC" dins la llibreria d'IBM, "LMIC". S'hi mostra un petit missatge per saber que s'ha carregat la informació.

```
// Funció per carregar la informació després de despertar
void LoadLMICFromRTC()
{
    Serial.println(F("Carregar LMIC des de RTC"));
    LMIC = RTC_LMIC;
}
```

La funció "GoDeepSleep()" es crida quan es desitja enviar l'embedded a dormir, primer de tot es mostren els missatges "Go DeepSleep" i el resultat de la funció "PrintRuntime()", llavors es finalitza les comunicacions del "Serial" amb la funció "Serial.flush()" i finalment s'envia a dormir amb la funció "ESP.deepSleep(TX_INTERVAL * 10000000)" usant la variable global "TX_INTERVAL" multiplicada per un milió perquè el camp es troba en microsegons.

```
// Funció per dormir el dispositiu
void GoDeepSleep()
{
    // DEEPSLEEP
    Serial.println(F("Go DeepSleep"));
    // Mirem quan ha durat la retrsansmissió
    PrintRuntime();
    //Apaguem el Serial
    Serial.flush();
    // Posem a dormir
    ESP.deepSleep(TX_INTERVAL * 1000000);
}
```

El codi que contempla la següent funció s'explica en diferents parts a causa de la seva complexitat i les diferents seccions que contempla. La primera part mostra la definició de la funció anomenada "onEvent", aquesta és executada per la llibreria d'IBM cada cop que ha transcorregut el temps inserit a la variable "TX_INTERVAL". Compte amb una sola variable configurada, "ev" de tipus "ev_t" es troba definida dins la llibreria LMIC i descriu tota la parametrització segons la classe LoRa del Node (A, B o C), com el senyal "BEACON", si cal activació "JOINING", etc.

Un cop s'executa mostra el temps en mil·lisegons des de l'execució del programa dins el node i comença una estructura "switch" segons els estats que es troben dins "ev".

L'estat més important, al tractar-se de la classe A, és el que conté "EV_JOINED", aquesta execució utilitza les variables comprimides de les funcions anteriorment esmentades per APPEUI, DEVEUI i APPKEY. Les estructura segons les capes de comunicació necessàries i les mostra pel Serial de la placa EZSBC.

```
// Funció que respon segons l'estat de les comunicacions
void onEvent (ev_t ev) {
    Serial.print(os_getTime());
    Serial.print(": ");
    switch(ev) {
        case EV_SCAN_TIMEOUT:
            Serial.println(F("EV_SCAN_TIMEOUT"));
            break;
        case EV_BEACON_FOUND:
            Serial.println(F("EV_BEACON_FOUND"));
            break;
        case EV_BEACON_MISSED:
            Serial.println(F("EV_BEACON_MISSED"));
            break;
        case EV_BEACON_TRACKED:
            Serial.println(F("EV_BEACON_TRACKED"));
            break;
        case EV_JOINING:
            Serial.println(F("EV_JOINING"));
```

```
        break;
    case EV_JOINED:
        Serial.println(F("EV_JOINED"));
        {
          u4_t netid = 0;
          devaddr_t devaddr = 0;
          u1_t nwkKey[16];
          u1_t artKey[16];
          LMIC_getSessionKeys(&netid, &devaddr, nwkKey, artKey);
          Serial.print("netid: ");
          Serial.println(netid, DEC);
          Serial.print("devaddr: ");
          Serial.println(devaddr, HEX);
          Serial.print("AppSKey: ");
          for (size_t i=0; i<sizeof(artKey); ++i) {
            if (i != 0)
              Serial.print("-");
            printHex2(artKey[i]);
          }
          Serial.println("");
          Serial.print("NwkSKey: ");
          for (size_t i=0; i<sizeof(nwkKey); ++i) {
                  if (i != 0)
                          Serial.print("-");
                  printHex2(nwkKey[i]);
          }
          Serial.println();
        }
        LMIC_setLinkCheckMode(0);
        break;

    //|| case EV_RFU1:
    //||     Serial.println(F("EV_RFU1"));
    //||     break;
    //
    case EV_JOIN_FAILED:
        Serial.println(F("EV_JOIN_FAILED"));
        break;
```

```
case EV_REJOIN_FAILED:
    Serial.println(F("EV_REJOIN_FAILED"));
    break;
```

La segona part de l'estructura "switch" contempla el cas "EV_TXCOMPLETE", el més important per l'actuador lumínic, ja que és l'encarregat de rebre les dades i actuar en conseqüència. La primera estructura condicional "if" espera la finestra R1 de recepció de dades a través de la Gateway. Per optimitzar el programa es mira la variable "LMIC.dataLEn" si compte amb dades, longitud de les dades rebudes, tot seguit es mira si aquesta longitud és d'un byte, en cas afirmatiu aquesta es tracta de la dada que s'ha enviat des del sistema de gestió. S'estructura dins la variable "result" a través de la llista "LMIC.frame" que permet seleccionar byte a byte de les variables rebudes. Tot seguit s'activa o desactiva la il·luminació i es guarda aquest estat dins el primer byte de la llista Payload, "Payload[0]", per ser tramesa durant la següent transmissió. Com es tracta d'un pin RTC primer s'ha de deshabilitar amb la funció "rtc_gpio_dis(LED_VERMELL)", llavors determinar l'estat "HIGH" o "LOW" (Activat o desactivat) amb la funció "rtc_gpio_set_level(LED_VERMELL, LOW)" i per últim es bloqueja el seu estat amb la funció "rtc_gpio_hold_en(LED_VERMELL)".

Finalment, s'activa la variable booleana "GOTO_DEEPSLEEP" i d'aquesta manera executar la funció per adormir el dispositiu quan sigui llegida.

```
case EV_TXCOMPLETE:
    Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
    if (LMIC.txrxFlags & TXRX_ACK)
      Serial.println(F("Received ack"));
    if (LMIC.dataLen) {
      Serial.print(F("Received "));
      Serial.print(LMIC.dataLen);
      Serial.println(F(" bytes of payload"));

      //------ Recepció Dades Downlink ----------------

      if (LMIC.dataLen == 1) {
        uint8_t result = LMIC.frame[LMIC.dataBeg + 0];
        if (result == 0)  {
```

```
            Serial.println("RESULT 0");
            rtc_gpio_hold_dis(LED_Vermell); //Deshabilitar el PIN RTC
            rtc_gpio_set_level(LED_Vermell, LOW); //set high/low
            rtc_gpio_hold_en(LED_Vermell); //Habilitar el pin RTC
            payload[0] = 0;
        }
        if (result == 1)  {
            Serial.println("RESULT 1");
            rtc_gpio_hold_dis(LED_Vermell); //Deshabilitar el PIN RTC
            rtc_gpio_set_level(LED_Vermell, HIGH); //set high/low
            rtc_gpio_hold_en(LED_Vermell); //Habilitar el pin RTC
            payload[0] = 1;
        }
      }
     Serial.println();
    }

    // Activem la variable per la següent iteració anar a dormir
    GOTO_DEEPSLEEP = true;

    break;
```

L'última part de l'estructura contempla possibles resultats d'error durant la transmissió de les comunicacions, com "EV_LOST_TSYNC" o "EC_TXCANCELED", l'execució correcta de la recepció de les dades "EV_RXCOMPLETE" i errors d'activació del node "EV_JOIN_TXCOMPLETE", que mostra un missatge indicant que el "JOIN" no ha funcionat correctament. Finalment, esmentar l'estat "EV_RXSTART", el qual no ha de comptar amb cap mena de programa al seu interior perquè si no els temps de transmissió fallen (segons la informació de la llibreria).

```
case EV_LOST_TSYNC:
    Serial.println(F("EV_LOST_TSYNC"));
    break;
case EV_RESET:
    Serial.println(F("EV_RESET"));
    break;
```

```
        case EV_RXCOMPLETE:
            // Dades rebudes
            Serial.println(F("EV_RXCOMPLETE"));
            break;
        case EV_LINK_DEAD:
            Serial.println(F("EV_LINK_DEAD"));
            break;
        case EV_LINK_ALIVE:
            Serial.println(F("EV_LINK_ALIVE"));
            break;

        //|| case EV_SCAN_FOUND:
        //||    Serial.println(F("EV_SCAN_FOUND"));
        //||    break;

        case EV_TXSTART:
            Serial.println(F("EV_TXSTART"));
            break;
        case EV_TXCANCELED:
            Serial.println(F("EV_TXCANCELED"));
            break;
        case EV_RXSTART:
            /* No pescriure res o el timing no anirà bé */
            break;
        case EV_JOIN_TXCOMPLETE:
            Serial.println(F("EV_JOIN_TXCOMPLETE: no JoinAccept"));
            break;

        default:
            Serial.print(F("Unknown event: "));
            Serial.println((unsigned) ev);
            break;
    }
}
```

7.2. Aplicació LoRa

Dins l'aplicació Aparcament cotxe existeix la possibilitat d'afegir un petit Script per fer un tractament de les dades rebudes o enviades, a escala general de tota l'aplicació o independent per cada node, aquest programa ha de ser escrit amb JavaScript. Per aquest projecte s'utilitza un script de tractament de les dades per cada node.

El sensor d'aparcament compta amb un programa dins l'apartat "Payload formatters" dins el menú "End Devices" i seleccionant el sensor en qüestió. L'estructura de la funció "decodeUplink" és predeterminada pel tipus descrit a la configuració, com es veu a la següent figura.

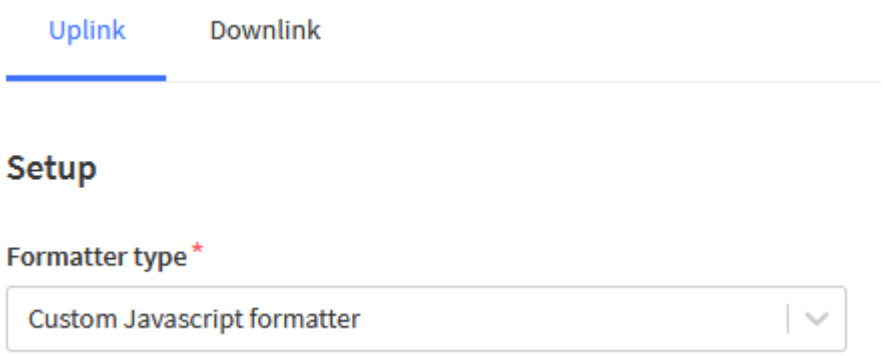

Figura 64 Formatter type

La variable "input" que usa la funció, és el nom que rep l'estructura de tipus JSON de la comunicació "Uplinks" rebut del sensor. De la seva estructura és necessària l'apartat "bytes", el qual empaqueta tota la informació enviada pel sensor. Aquesta funció s'ha programat per estructurar de manera correcta la informació del sensor d'aparcament i que es mostri correctament a l'aplicació a l'apartat "Live Data".

Es crea l'estructura JSON anomenada "data" i els noms de les variables enviades. "Sens_type" és el primer byte rebut i no cal processar-lo, "Frame_Type" és una variable de 4 bits dins els llocs de més pes del segon byte, com a tal s'utilitza la comparativa "& 0xF0" per triar les dades necessàries i es mouen els quatre bits de major pes als de menor amb ">> 4".

La variable "Direction" es troba al mateix byte que el "Frame_type" però al bit de menor pes, per aquest motiu només cal extraure la informació d'aquest byte amb la comparació "& 0x01".

Totes les variables "Axis" (X, Y i Z) són tractades igual, són valors enters amb el que estan compostos per dos bytes de la transmissió, per poder operar el valor abans cal estructurar la informació rebuda i invertir-la perquè els bytes de més pes arriben primer, el codi "input.bytes[2] << 8 + input.bytes[3]" mou el primer byte vuit posicions i concatena el següent a continuació (exemple "X_Axis"). Per acabar cal restar un valor de 32.768 i llavors, dividir 4096, d'aquesta manera es converteix en un valor real de tipus "float", segons indicació del manual de IOTSens.

La variable "Temp" que indica la temperatura, es rep com a enter dins el vuitè i novè byte i cal invertir-lo com les variables "Axis", llavors dividir el valor per 100, donant com a resultat un valor real de tipus "float".

El "Parking_status" és el bit de major pes del byte número 10 rebut, es compara amb el programa "& 0x80" i tot seguit es mou set bits a l'inici del byte amb ">> 7". La resta de valors són usats per determinar el valor de la bateria "Battery_Voltage", es compara amb el valor "& 0x7F" i es divideix per 10, convertint el valor simple d'un enter per un real de coma flotant "float".

La variable "Type_Sens" és creada específicament per diferenciar ràpidament entre el sensor d'aparcament i l'actuador lumínic.

Finalment, la funció retorna l'estructura "data" amb el nom "data" i possibles "warnings" i "errors", programats per l'aplicació LoRaWAN. A partir d'aquest punt, la integració webhook és l'encarregada de redirigir aquesta informació a l'URL del servidor de la UdG.

```
function decodeUplink(input) {
    var data = {};
    //data.bytes = input.bytes;
```

```
    data.Sens_type = input.bytes[0];
    data.Frame_type = (input.bytes[1] & 0xF0) >> 4;
    data.Direction =input.bytes[1] & 0x01;
    data.X_Axis = (((input.bytes[2] << 8) + input.bytes[3])-32768)/4096;
    data.Y_Axis = (((input.bytes[4] << 8) + input.bytes[5])-32768)/4096;
    data.Z_Axis = (((input.bytes[6] << 8) + input.bytes[7])-32768)/4096;
    data.Temp = ((input.bytes[8] << 8) + input.bytes[9])/100;
    data.Parking_status = (input.bytes[10] & 0x80) >> 7;
    data.Battery_Voltage = (input.bytes[10] & 0x7F)/10;
    data.Type_Sens = 1;

    return {
      data: data,
      warnings: [],
      errors: []
    };
  }
```

El sensor d'aparcament no compta amb Script per Downlinks.

L'actuador lumínic compta amb Script per Uplink i Downlink, tot seguit queden explicats. El programa per Uplinks és prou curt i es pot programar a la part "return" de la funció, s'estructura la resposta dins la variable "data" amb el valor de l'estat del led ("LED"), és l'únic byte rebut per l'aplicació. S'afegeix la variable "Type_Sens" amb un valor de 2 per distingir-lo del sensor d'aparcament. Finalment, les variables "warnings" i "errors", per depurar programa.

```
function decodeUplink(input) {
  return {
    data: {
      LED: input.bytes[0],
      Type_Sens: 2
    },
    warnings: [],
    errors: []
  };
```

```
}
```

Els missatges Downlink procedents de l'aplicació de gestió compta amb dues funcions, "encodeDownlink" codifica la informació enviada, només es tracta d'un byte d'activació o desactivació del LED a través del port 1 (variable "fPort").

```
function encodeDownlink(input) {
  return {
    bytes: [],
    fPort: 1,
    warnings: [],
    errors: []
  };
}
```

Tot seguit, dins el mateix Script es declara la funció predeterminada de descodificació de Downlink, "decodeDownlink" la qual mostra els bytes enviats correctament a l'apartat "Live Data" de l'aplicació.

```
function decodeDownlink(input) {
  return {
    data: {
      bytes: input.bytes
    },
    warnings: [],
    errors: []
  }
}
```

7.3. Aplicació servidor JavaScript

El programa gestor dels aparcaments és exposat a aquest apartat, exhibeix els paquets utilitzats, les funcions de petició HTTP i les que gestionen la lògica implementada.

L'explicació segueix el següent fil, enumerar els diferents paquets (npm) usats (a l'inici del programa), les funcions encarregades de la seva configuració (al final del programa), les funcions HTTP (POST, GET, etc.) i finalment les funcions que gestionen la lògica.

El primer paquet és "dotenv", s'utilitza gràcies a la funció "require()", la qual executa el paquet npm instal·lat a l'API (i igual per tots els paquets usats). El paquet "dotenv" és secundari i derivat de les proves realitzades amb la integració webhook, la seva funció és relacionar variables de configuració dipositades a un arxiu amb l'extensió ".env". Tot i no afectar a la gestió es deixa disponible per realització de proves en ambients que requereixin el seu ús com, gestor d'URL, contrasenyes, etc.

El segon paquet anomenat "express" és l'encarregat de gestionar les peticions que es fan a l'aplicació a través del protocol HTTP, és a dir, executar un servidor d'escolta, la seva configuració es volca dins la constant "express" i tracta totes les peticions entrants a l'API, ja siguin Webhooks, etc.

Finalment, el tercer "axios", paquet encarregat d'executar peticions a qualsevol servidor aliè, és utilitzat per realitzar peticions HTTP al servidor TTN. Al següent extracte de codi es poden observar.

```
require("dotenv").config();
const express = require("express");//Servidor HTTP
const axios = require("axios").default;//Client HTTP per enviar al TTN
```

Llavors, es volca la configuració a les constants "app" i "port", la primera és usada per accedir a les funcions del paquet "express()" i el port tal com indica el seu nom s'hi especifica el port d'ús de l'API.

```
//Executem el servidor i client
const app = express();
const port = 40300;
```

Llavors, el paquet npm "mysql" és una integració que permet realitzar les comunicacions entre l'API i el WampServer amb llenguatge SQL. Primer de tot és executat sota la variable "mysql" i finalment configurat dins la variable "connection" a través de la funció "mysql.createConnection". La seva configuració és la mateixa que a l'apartat Configuració i inicialització WampServer MySQL. El format JSON de l'estructura permet emplenar els paràmetres "host", "user", "password" i "database", que donen accés a la base de dades corresponent a la creada per aquest projecte.

Finalment, l'última funció d'aquest codi (a continuació), "connection.connect", com el seu nom indica si la connexió amb la base de dades a través del WampServer ha estat satisfactòria o no. En cas d'haver-hi un error el mostra per la CMD, però si ha estat satisfactori i no troba cap error, mostra el missatge "Database server running!!!" a través de la CMD amb la funció "console.log()". Per aquest motiu és imprescindible haver executat el servidor Wamp prèviament i estar segurs del seu correcte funcionament.

```
//Base de dades MySQL
var mysql      = require('mysql');

var connection = mysql.createConnection({
  host     : 'localhost',
  user     : 'root',
  password : '',
  database : 'mydb'
});

connection.connect(error => {
  if (error) throw error;
  console.log('Database server running!!!');
});
```

Per facilitar la gestió de peticions amb el paquet "express" es configura el seu ús com a JSON, a continuació es veu el codi utilitzant la funció "app.use", funció que permet accedir a parts

específiques dels arguments (path, body, callback, etc) d'un objecte (en aquest cas la configuració) i la configuració "express.json".

```
//Declara l'estructura de rebuda com a JSON
app.use(express.json());
```

El servidor ha de ser capaç de respondre qualsevol petició que li arribi, d'aquesta manera es configura una funció usant "app.use", en aquest cas accedint als arguments "req." de la petició HTTP i mira si existeix algun error, en cas afirmatiu avisa a la resposta amb el codi "res.status(500)", error intern del servidor, per donar a entendre que no ha estat capaç de gestionar la informació. Acte seguit envia aquest error com a resposta "res." a la crida formalitzada "res.send((error: error))" i mostra aquest valor per la CMD amb "console.error()".

```
// Errors no descrits a l'aplicació
app.use((error, req, res, next) => {
  res.status(500)
  res.send({error: error})
  console.error(error.stack)
  next(error)
})
```

La configuració del port es fa utilitzant la funció del codi que es veu a continuació a través de "app.listen()" amb l'entrada de la variable "port", respon amb el missatge adient i el port usat gràcies a la inserció d'aquesta variable dins el text amb el format "${port}".

```
// Missatge d'avís, per la connexió del port
app.listen(port, () =>
  console.log(`Aplicació executada a la direcció http://localhost:${port}`)
);
```

Un cop explicats tots els paquets usats i les seves configuracions es pot començar amb la definició i ús de les peticions HTTP.

Primer de tot la pàgina web principal de l'API, en tractar-se d'un buscador web ha de poder respondre a la petició GET i llegir el programa HTML que compte amb un títol "<head><title>", un text "<h1>" i una imatge "<img>" dins el cos de la pàgina web "<body>". Per això s'ha formalitzat una petita i molt simple pàgina que mostra un text de benvinguda i un GIF (Graphic Interchange Format).

La funció "app.get" respon a la direcció arrel "/" a la petició "req" amb la pàgina descrita amb la funció "res.send()". Aquesta pàgina ha estat usada principalment per identificar ràpidament si el servidor es troba en ús i no per la seva possible configuració.

```
//Pàgina principal


app.get("/", (req, res) => res.send(`
  <html>
    <head><title>Success!</title></head>
    <body>
      <h1>Benvingut al servidor de gestió </h1>
      <img src="https://media.giphy.com/media/XreQmk7ETCak0/giphy.gif" alt="Cool kid doing thumbs up" />
    </body>
  </html>
`
));
```

La següent funció és d'ús únic i exclusiu per a la realització de proves a la base de dades, concretament afecta la taula "Sensor_cotxe1". Per realitzar aquestes peticions s'ha utilitzat el programari POSTMAN, que permet la configuració de les peticions així com automatitzar-les per fer testos a l'aplicació, però es pot usar qualsevol altre mètode o programa per realitzar les proves. Aquesta funció també serveix per provar possibles enviaments de dades amb una estructura diferent i realitzar

El codi següent es visualitza com la funció "app.post" escolta les peticions dirigides a "/provadb", està configurada amb la funció "async" (utilitzada en altres funcions) que permet

gestionar la petició en segon pla i optimitzar temps de resposta en cas de múltiples peticions al mateix temps.

Defineix les dues variables "req" i "res" iguals per totes aquestes funcions i tot seguit es configura la constant local "sql" amb el text SQL per fer un "INSERT" a la taula "Sensor_Cotxe1" finalitzant amb "SET ?" perquè no cal especificar les columnes.

Llavors, es defineix la constant "cotxeObj" per estructurar les dades rebudes o també anomenat "body" de la petició, en format JSON. Un cop definides aquestes dues variables cal executar la funció "connection.query", la funció estructura les dades definides i les envia al WampServer, en tractar-se d'un "INSERT" no s'espera resposta més enllà d'un possible error. Si tot ha funcionat correctament, la funció mostra el missatge per la consola, "Sensor cotxe rebut".

La segona funció "connection.query", es defineix amb la constant "sql2" i variable JSON "variable". La constant "sql2" es defineix com a "SELECT" dels últims cinc valors segons la data "ORDED BY Data DESC LIMIT 5" de la columna "DevEUI", taula "Sensor_Cotxe1". Aquests valors són dipositiats dins la "variable" com a resultat "result" de la funció local i seguidament tramet la resposta HTTP amb la funció "res.status(200).send(variable)" amb el missatge 200 com a resposta "OK".

```
// Base de dades de Proves, Trastejar aquí
app.post("/provadb", async (req, res) =>{
  const sql = 'INSERT INTO Sensor_Cotxe1 SET ?';
  const cotxeObj = {
    //Data: req.body.uplink_message.received_at,
    Data: new Date(req.body.uplink_message.received_at),
    DevEUI: req.body.end_device_ids.device_id,
    Parking_status: req.body.uplink_message.decoded_payload.Parking_status,
    Battery_Voltage: req.body.uplink_message.decoded_payload.Battery_Voltage,
    Direction: req.body.uplink_message.decoded_payload.Direction,
    Frame_type: req.body.uplink_message.decoded_payload.Frame_type,
    Sens_type: req.body.uplink_message.decoded_payload.Sens_type,
```

```
    Temp: req.body.uplink_message.decoded_payload.Temp,
    X_Axis: req.body.uplink_message.decoded_payload.X_Axis,
    Y_Axis: req.body.uplink_message.decoded_payload.Y_Axis,
    Z_Axis: req.body.uplink_message.decoded_payload.Z_Axis,
  }
  //Guardem a la DB Sensor_Cotxe1
  connection.query(sql, cotxeObj, error =>{
    if (error) throw error;
    console.log('Sensor cotxe rebut');
    });

  const sql2 = 'SELECT DevEUI FROM Sensor_Cotxe1 ORDER BY Data DESC LIMIT 5';
  let variable = {};
  connection.query(sql2, (error, result) =>{
    if (error) throw error;
    variable = {result};
    console.log('Sensor cotxe rebut');
    res.status(200).send(variable);
    });
  console.log('------------------');

})
```

Els Uplink rebuts a través de la xarxa LoRa a través de l'API TTN són gestionades per aquesta funció, l'explicació es fa en diverses fases que retallen el codi de la funció per millorar la interpretació i visualització de codi. La funció principal d'aquesta funció és assegurar la recepció de dades útils, discernir entre sensor i actuador i finalment guardar i actualitzar les dades dins les taules corresponents de la base de dades.

La funció respon a una petició "POST" de la integració webhook LoRa a la direcció "/sensor" (tot i rebre sensors i actuadors), es defineix com a "async" (descrit a la funció anterior) i les variables típiques "req" i "res".

Primer de tot s'ha implementat el condicional "if", que detecta l'existència del paràmetre "decoded_payload" dins l'estructura "req" de la petició rebuda, d'aquesta manera s'identifica

amb seguretat si es tracta d'un "Uplink" amb informació útil. Per detectar aquesta propietat s'ha usat la funció "hasOwnProperty()" de JavaScript que identifica el camp descrit dins l'estructura anomenada.

Un cop superat aquest primer filtre es volca la informació dins la variable "sensor" i es torna a filtrar segons el paràmetre "Type_Sens", implementada durant la descodificació i primer tractament dels Uplink a l'API LoRa.

Quan la variable "Type_Sens" es guarda a la variable local "sensor", si aquesta val "1" es considera Uplink del sensor d'aparcament, llavors es prepara per registrar (inserir) la seva informació dins la taula d'històrics "Sensor_Cotxe" de la mateixa manera que s'ha descrit anteriorment, es defineixen les constants locals "sql" i "cotxeObj" i es genera una funció SQL "connection.query". Si ha estat executat correctament, mostra el missatge "Sensor cotxe (valor DevEUI) rebut".

```
//Post missatges rebuts de la API LoRa, TTN V3
app.post("/sensor", async (req, res) =>{

  //Mirem si és sensor Cotxe o actuador LED
  if (req.body.uplink_message.hasOwnProperty('decoded_payload')){
    let sensor = req.body.uplink_message.decoded_payload.Type_Sens;
    if (sensor == 1){
      //Preparem dades del sensor cotxe
      const sql = 'INSERT INTO Sensor_Cotxe SET ?';
      const cotxeObj = {
        Data: new Date(req.body.uplink_message.received_at),
        DevEUI: req.body.end_device_ids.device_id,
        Parking_status: req.body.uplink_message.decoded_payload.Parking_status,
        Battery_Voltage: req.body.uplink_message.decoded_payload.Battery_Voltage,
        Direction: req.body.uplink_message.decoded_payload.Direction,
        Frame_type: req.body.uplink_message.decoded_payload.Frame_type,
        Sens_type: req.body.uplink_message.decoded_payload.Sens_type,
        Temp: req.body.uplink_message.decoded_payload.Temp,
        X_Axis: req.body.uplink_message.decoded_payload.X_Axis,
        Y_Axis: req.body.uplink_message.decoded_payload.Y_Axis,
```

```
        Z_Axis: req.body.uplink_message.decoded_payload.Z_Axis,
      }
      //Guardem a la Taula Sensor_Cotxe
      connection.query(sql, cotxeObj, error =>{
      if (error) throw error;
      console.log('Sensor cotxe ' + cotxeObj.DevEUI + ' rebut');
      });
```

Dins el mateix condicional es parametritza una segona funció "connection.query", la qual usa una constant diferent "sql2", aquesta canvia la informació (UPDATE) de la variable "Parking_Status" de la taula "gestió_cotxe" on la columna "DevEUI_cotxe" coincideixi amb el valor de la variable "DevEUI". Gràcies al condicional SQL "WHERE" permet aquesta selecció de la taula i les funcions "mysql.escape" habiliten la inserció de variables dins el text. Si ha funcionat correctament mostra el missatge "Sensor cotxe (DevEUI) actualitzat".

```
      //Actualitzem la Taula Gestio_Cotxe
           const sql2 = 'UPDATE Gestio_Cotxe SET Parking_Status = ' +
mysql.escape(cotxeObj.Parking_status) + ' WHERE DevEUI_cotxe =' +
mysql.escape(cotxeObj.DevEUI);
      //Guardem a la DB Gestio_Cotxe
      connection.query(sql2, error =>{
      if (error) throw error;
      console.log('Sensor cotxe ' + cotxeObj.DevEUI + ' actualitzat');
      });

    }
```

Un cop finalitzada la part del sensor s'expliquen les accions realitzades per l'actuador. Primer de tot la variable "sensor" ha de valer 2, un cop duta a terme aquesta primera confirmació es tracta la informació rebuda pel "body" de la petició, per aquest cas més senzill, ja que només s'envia una sola variable, l'estat del led, anomenat "LED" i es guarda dins la constant "ledObj". Per aquest cas es defineix una variable anomenada "ledEUI" que guarda el valor "device_id" a part.

Les dades dels actuadors LED no tenen taula d'històrics, per aquest motiu només cal realitzar una petició "UPDATE" a la taula "gestió_cotxe", molt semblant a la realitzada pel sensor d'aparcament, però en aquest cas per la columna "Estat_led" quan la columna "DevEUI_led" coincideix amb la variable "ledEUI". Finalment, s'executa la funció "connect.query" que relaciona aquesta "sql" nova i mostra el missatge "Actuador (ledEUI) led actualitzat" si ha estat satisfactori.

L'última funció executada és la "res.status(200).send()" que retorna un missatge satisfactori a l'aplicació TTN, per assegurar una correcta implementació del protocol HTTP.

```
    else if (sensor == 2){

        let ledEUI =req.body.end_device_ids.device_id;
        const ledObj = { Estat_led: req.body.uplink_message.decoded_payload.LED }
        //console.log(ledObj);
        //Preparem dades del sensor cotxe
                const sql = 'UPDATE Gestio_Cotxe SET Estat_led =' +
mysql.escape(ledObj.Estat_led) + ' WHERE DevEUI_led =' + mysql.escape(ledEUI);

        //Guardem a la Taula Gestio_Cotxe
        connection.query(sql, error =>{
        if (error) throw error;
        console.log('Actuador ' + ledEUI + ' led actualitzat');
        });
    }
  }

  res.status(200).send();
})
```

La creació de la llista de registres sensorials que relacionen el sensor d'aparcament i l'actuador dins la taula "gestió_cotxe" a la base de dades "mydb" es fa a través de peticions formalitzades amb el programari POSTMAN. De nou, no és l'únic mètode que pot ser utilitzat.

La petició respon a la funció "POST" amb la direcció "/registre" i es tracta d'una funció "async". Primer de tot es passen les dades a les variables "cotxe" i "led" i se'n creen dues més, "c_cotxe" i "c_led" les quals són usades per comparar els resultats.

Llavors, es defineix la funció "Logica", però abans de descriure-la, fa falta explicar l'execució de la funció "connection.query" que es troba a la part final de la resposta "POST". Es defineix la constant local "sql" amb què es fa una selecció (SELECT) del valor coincident (WHERE) amb la variable "cotxe" a la columna "DevEUI_Cotxe" o del valor "led" afí a la columna "DevEUI_led" de la taula (FROM) "gestió_cotxe". Si s'obté alguna resposta aquests valors són copiats a les variables de còpia, "c_cotxe" i "c_led". Finalment, s'executa la funció "Logica".

La funció "Logica" no és "async" perquè fa falta la resposta de la base de dades, per aquest motiu es crida dins la mateixa petició a aquesta. Un cop s'executa, contempla quatre condicionals que per ordre queden:

El primer condicional compara els registres "cotxe" i "led" rebuts i la resposta de la base de dades, "c_cotxe" i "c_led" i espera que no en coincideixi cap d'ells. Un cop s'executa es formalitza de nou una funció "connection-query" amb les constants "sql2" i "cotxeObj", la primera serveix per inserir (INSERT) una nova línia a la taula "gestió_cotxe" i la segona constant per crear el cos de les dades, conté valors a totes les columnes de la taula destinatària. Si l'execució ha resultat satisfactòria mostra el missatge "REGISTRAT!" per la consola CMD.

La resta de condicionals sols informen de la nul·litat del registre dins la base de dades. El segon condicional filtra si el cotxe i el led es troben registrats i mostra el missatge "El sensor del cotxe i led ja estan registrats". El tercer espera que el cotxe coincideixi i el led discerneixi llavors, mostra el missatge "Sensor cotxe ja registrat". El quart, espera que el cotxe no coincideixi i el led si, en cas afirmatiu mostra el missatge "Actuador LED ja registrat". Finalment, qualsevol altra combinació mostra l'error "Error desconegut".

Finalment, es respon a la petició amb el missatge 200 com a resposta amb la funció "res.status(200).send()".

```
//Post Creació Postman dels sensors
app.post("/registre", async (req, res) =>{

  //Mirem si ja existeix
  let cotxe = req.body.EUICotxe;
  let c_cotxe = 'A';

  let led = req.body.EUILED;
  let c_led = 'A';

  //Al no ser async espera a recopilar l'informació i llavors decideix
  function Logica(c_cotxe, c_led){
    if (c_cotxe != cotxe & c_led != led) {
      //Preparem dades dela DB gestió cotxe
      const sql2 = `INSERT INTO Gestio_Cotxe SET ?`;
      const cotxeObj = {
        DevEUI_cotxe: req.body.EUICotxe,
        Parking_status: 0,
        Downlinks_sent: 0,
        DevEUI_led:req.body.EUILED,
        Estat_led:0,
      }
      //Guardem a la DB Sensor_Cotxe
      connection.query(sql2, cotxeObj, error =>{
      if (error) throw error;
      console.log('REGISTRAT!');
      });
    }
    else if ( c_cotxe == cotxe & c_led == led ) {
      console.log('El sensor del cotxe i led ja estan registrats!');
    }
    else if ( c_cotxe == cotxe & c_led != led ) {
      console.log('Sensor cotxe ja registrat!');
    }
    else if ( c_cotxe != cotxe & c_led == led ) {
      console.log('Actuador LED ja registrat!');
```

```
    }
    else {
      console.log('Error desconegut');
    }
  }

  //Preguntem a la base de dades si ja existeix algun camp
  const sql = 'SELECT * FROM Gestio_Cotxe WHERE DevEUI_cotxe = ' + mysql.escape(cotxe)
+ 'OR DevEUI_led  = ' + mysql.escape(led);
  connection.query(sql, (error, result) =>{
    if (error) throw error;
    if (result.length > 0){
    c_cotxe = result[0].DevEUI_cotxe;
    c_led = result[0].DevEUI_led;
    }
    Logica(c_cotxe,c_led);
  });

  res.status(200).send();
})
```

La següent funció serveix per activar o desactivar manualment l'actuador lumínic a través de peticions HTTP a través de POSTMAN, simula l'enviament d'un "Downlink" a l'aplicació TTN. El següent codi es troba configurat per apagar l'actuador. La funció respon a una petició "POST" a través de la direcció "/downlink" i executa la funció amb característica "async", defineix la constant "sended" amb l'estat de l'activació en formar hexadecimal i codificada en base64. Tot seguit es defineix i executa la funció "axios" perquè enviï una petició a l'aplicació TTN.

Per generar una petició a l'aplicació creada al TTN fa falta emplenar correctament els següents encapçalaments: "headers", és un objecte JSON que es troba format per "Authorization" omplert amb la fórmula "Bearer + 'APP key'", on la APP key demanada és la creada per la integració Webhook, "Content-Type" que ha d'indicar si és aplicació o gateway i el format de les dades, per aquest cas "Application/json", finalment, "User-Agent" que indica el nom de l'aplicació i la versió de TTN, "proves-cotxe/v3".

Llavors, cal indicar el mètode de petició HTTP dins "method", com es desitja realitzar un Downlink ha de ser un "post".

Per la "url" s'ha d'utilitzar la següent fórmula https://eu1.cloud.thethings.network/api/v3/as/applications/--Nom_Aplicació_TTN--/webhooks/--Nom_Webhook--/devices/--Nom_Device_EUI--/--Mode_HTTP--. El nom de l'aplicació és "proves-cotxe", el webhook és "api-webhook-udg", l'EUI del node destinatari, que és el "eui-70b3d57ed004da1c", finalment que es tracti d'una petició "post" comporta el mode "/down/push" perquè importi les dades transmeses.

Per últim, fa falta crear la família "downlinks" en format JSON dins "data", perquè el TTN tracti les dades com a tal i tres paràmetres, "frm_payload" on es volca la informació enviada (variable "sended"), el "f_port", el qual cal diferenciar-lo del "Uplinks" actuals del sensor perquè no xoquin i la prioritat dins "priority", per aquest cas "NORMAL".

Finalment, es respon amb el codi 200.

```
//Downlink LoRa Proves
// 1 = AQ== ------ 0 = AA==
app.post("/downlink", async (req, res) =>{
  const sended = 'AA==';
  axios({
    headers: {'Authorization': 'Bearer NNSXS.5SFW...',
    'Content-Type': 'application/json',
    'User-Agent': 'proves-cotxe/v3'},
    method: 'post',
         url:    'https://eu1.cloud.thethings.network/api/v3/as/applications/proves-
cotxe/webhooks/api-webhook-udg/devices/' + 'eui-70b3d57ed004da1c' + '/down/push',
    data: {"downlinks": [{
      "frm_payload": sended,
      "f_port": 15,
      "priority":"NORMAL"
    }]
    }
```

```
  });
  res.status(200).send();
})
```

Les funcions encarregades de gestionar la lògica són executades periòdicament a través de la funció "setInterval(funció,'temps en ms')", la qual genera un esdeveniment i executa la funció parametritzada amb els mil·lisegons programats.

El següent codi mostra la parametrització d'aquestes funcions, les quals han estat definides dins les constants "BuffRev" i "BuffRevDesac" que criden l'execució de les funcions "RevDevEUI" i "RevDevEUIdesac" cada 30 segons. La sola definició d'aquestes constants és suficient perquè el programa les executi cada 30 segons, si es vol fer per altres mètodes fa falta crida les funcions ("RevDevEUI" i "RevDevEUIdesac") a part.

```
// Aquesta funció mira la taula on hi han registrats tots el Sensors i envia l'ordre
per aquells necessaris d'activar el LED


const BuffRev = setInterval(RevDevEUI, 30000);//Cada 30 segons
const BuffRevDesac = setInterval(RevDevEUIdesac, 30000);//Cada 30 segons
```

Les funcions anteriorment esmentades s'expliquen a continuació, el primer codi és la definició de la funció "RevDevEUI" (Revision Device EUI), encarregada de crear la llista de tots aquells dispositius que es troben ocupats per un vehicle i encarregada de cridar la següent funció per determinar l'enviament del "Downlinks" als actuadors corresponents.

Primer de tot es crea la variable "indxbuff" (nombre total de dispositius) i la llista "arrsensor" (Array que compte amb els valors de la base de dades). Llavors, es fa una petició a la taula "gestió_cotxe" de tota la informació on les columnes de cada sensor complexin amb els valors següents: "Parking_Status = 1", "Downlink_sent = 0" i "Estat_led = 0". A continuació es posen aquests valors a les variables locals, sempre que la resposta compti amb valors, on "result.length > 0". Finalment, crida la funció "BufferSelecció" amb les llistes com a entrades, "indxbuff" i "arrsensor".

```
//Funció que revisa i  crea la llista que envia els downlinks d'ACTIVACIÓ
function RevDevEUI(){
  let indxbuff = 0;
  let arrsensor = [];
  const sql = 'SELECT * FROM Gestio_Cotxe WHERE Parking_Status = 1 AND Downlinks_sent
= 0 AND Estat_led = 0 ';
  connection.query(sql, (error, result) =>{
    if (error) throw error;
    if (result.length > 0){
      indxbuff = result.length;
      arrsensor = result;
      //console.log(arrsensor[0]);
      //console.log(indxbuff);
      //console.log(arrsensor);
    }
    BufferSeleccio(indxbuff, arrsensor);
  });

};
```

A l'anterior funció executa una segona anomenada "BufferSelecció", aquesta compara cada sensor i actuador de la llista creada i determina si ha superat el temps establert, en cas afirmatiu envia el Downlink a la xarxa TTN amb l'actuador corresponent. El codi queda separat per les explicacions de cada part.

La funció "BufferSeleccio" compte amb dues entrades, "index" i "llista", aquests valors són predeterminats a la funció anterior, "RevDevEUI". Llavors, es defineixen les noves variables locals "arrcotxe", "arrdwlnk" i "arrdate" com a llistes per incloure, primer, la llista de sensors d'aparcament, segon, la llista d'actuadors i finalment, la llista de les dates i temps de l'històric.

La primera part utilitza un bucle "for" que extrau la informació de la "llista" segons el nombre de la variable "index" incloses a la funció i se seleccionen les variables de les columnes "DevEUI_cotxe" i "DevEUI_led" per emplenar les llistes "arrcotxe" i "arrdwlnk".

```
//Funció que mira si ja s'ha enviat Downlink o si el led està ACTIVAT
function BufferSeleccio(index, llista){
```

```
let arrcotxe = []; // Array cotxes
let arrdwlnk = []; // Array leds
let arrdate = []; // Array dates dels cotxes
//Creem les llistes per buscar temps i preparar downlinks
for (let i = 0; i < index; i++) {
  arrcotxe[i] = llista[i].DevEUI_cotxe;
  arrdwlnk[i] = llista[i].DevEUI_led;
};

//console.log(arrcotxe);
```

Un cop generades les llistes, es descriu un segon bucle "for" amb el nombre "index", per determinar la necessitat d'enviar un "Downlink" per cada registre. La llista "arrcotxe[i]" és enumerada i definida dins la variable "eui", la qual és utilitzada per la definició de la següent petició a la base de dades. Aquesta petició selecciona l'últim valor (amb la funció "ORDER BY Data DESC LIMIT 1") de la columna "Data" dins la taula d'històrics "sensor_cotxe" i fent ús dels condicionals, primer que el valor "eui" correspongui amb la columna "DevEUI" i segon, que la columna "Parking_status" sigui 1.

Si es compleixen els requisits anteriors, la base de dades retorna un valor dins la variable "result" i es compleix que "result.length > 0". Llavors s'emplena la llista "arrdate[i] = result[0].Data", ja que ha de tractar-se com una llista amb format JSON.

Tot seguit s'executa la funció "new Date()" que es guarda dins la constant "act", d'aquesta manera es guarda la data i hora actual. S'utilitza la mateixa funció, però en aquest cas per convertir la variable "Data", la qual és "datatime" a "Date" dins la variable "t_sens". Aquest format "Date" és el valor numèric corresponent als mil·lisegons transcorreguts des de l'1 de gener del 1970 a les 00:00:00 UTC.

És desitjable operar els valors amb un nombre enter, d'aquesta manera es determina amb més facilitat el transcurs del temps, d'ençà que el sensor ha estat ocupat fins a l'actualitat. Per aquesta raó, la variable "minuts" equival a l'explicació anterior, amb el codi "(act – t_sens)/60000", la qual retorna els minuts de diferència.

```
// Un cop sabem la llista dels sensors busquem quins han sobrepassat el temps
```

```
  for (let i = 0; i < index; i++) {
    let eui = arrcotxe[i];
    const sql = 'SELECT Data FROM Sensor_Cotxe WHERE DevEUI = ' + mysql.escape(eui)
+ ' AND Parking_status = 1 ORDER BY Data DESC LIMIT 1';
    connection.query(sql, (error, result) =>{
      if (error) throw error;
      if (result.length > 0){
        //let index = result.length;
        arrdate[i] = result[0].Data;

        const act = new Date(); // Hora actual
        const t_sens = new Date(arrdate[i]); // Hora del sensor

        //console.log((act - t_sens));
        let minuts = (act - t_sens)/60000;
```

Llavors, s'utilitza el condicional "If" amb "minuts > 1" (1 minut per les proves, realment és de 15 minuts) i en cas afirmatiu envia la constant "sended = 'AQ=='", on "AQ==" és el format hexadecimal codificat amb "base64", requisit de l'aplciació TTN, a través de la funció "axios", explicat a la funció "app.post (/Downlink ..." (Downlink LoRa Proves), amb la inserció de la llista numerada "arrdwlnk[i]" dins la direcció webhook, perquè arribi a l'actuador corresponent.

```
        //Mirem si han passat els minuts
        if (minuts > 1){
          console.log('Downlink enviat');
          //console.log(arrcotxe[i]);
          //console.log(arrdwlnk[i]);
          const sended = 'AQ==';
          axios({
            headers: {'Authorization': 'Bearer NNSXS.5SFW...,
            'Content-Type': 'application/json',
            'User-Agent': 'proves-cotxe/v3'},
            method: 'post',
            url: 'https://eu1.cloud.thethings.network/api/v3/as/applications/proves-
cotxe/webhooks/api-webhook-udg/devices/' + arrdwlnk[i] + '/down/push',
            data: {"downlinks": [{
              "frm_payload": sended,
              "f_port": 15,
```

```
              "priority":"NORMAL"
            }]
            }

          });
```

Finalment, per saber que ha estat enviat un Downlink es fa una petició de modificació "UPDATE" del registre (columna) "Downlinks_sent", gràcies a la funció SQL "SET" on la columna "DevEUI_led" sigui el mateix a la llista "arrdwlnk[i]". La funció "mysql.escape()", permet integrar variables dins les peticions SQL.

```
          //Finalment actualitzem el valor Downlink
          const sql = 'UPDATE Gestio_Cotxe SET Downlinks_sent  = 1 WHERE DevEUI_led
=' + mysql.escape(arrdwlnk[i]);

          //Guardem a la DB Gestio_Cotxe
          connection.query(sql, error =>{
          if (error) throw error;
          console.log('Downlink pel actuador ' + arrdwlnk[i] + ' actualitzat');
          });

        };
      }
    });
  }

  /*const date = new Date();
  console.log(date -Date.now());*/

};
```

La funció "RevDevEUIdesac" (Revision Devise EUI desactivació) revisa constantment si l'aparcament ha estat desocupat i actua en conseqüència, enviant "Downlinks" als actuadors corresponents.

Igual que amb la funció "RevDevEUI", es crea una variable index, "indxbuff" i la llista "arrsensor", es configura la constant "sql" per recollir tota la informació de la taula "gestió_cotxe" on els valors de les següents columnes siguin inversos a la primera funció: "Parking_Status = 0", "Downlinks_sent = 1" i "Estat_led = 1". És a dir, l'actuador ha estat activat anteriorment, es troba encès actualment, però l'aparcament ha quedat lliure. La funció SQL "SELECT *" a través de "connection.query" espera retornar la fila dels valors corresponents, si compleix aquesta condició ("result.length > 0") guarda els resultats dins l'índex i la llista. Finalment, executa la funció "BufferSeleccioDesac" amb les entrades corresponents, "indbuff" i "arrsensor".

```
// Funció que revisa i crea els downlinks de DESACTIVACIÓ

//Aquesta funció no contempla el cas PS = 0 AND D_s = 1 AND E_l = 0 Espera que el LED
dongui resposta E_l = 1

function RevDevEUIdesac(){

  let indxbuff = 0;

  let arrsensor = [];

  const sql = 'SELECT * FROM Gestio_Cotxe WHERE Parking_Status = 0 AND Downlinks_sent
= 1 AND Estat_led = 1 ';

  connection.query(sql, (error, result) =>{

    if (error) throw error;

    if (result.length > 0){

      indxbuff = result.length;

      arrsensor = result;

      //console.log(arrsensor[0]);

      //console.log(indxbuff);

      //console.log(arrsensor);

    }

    BufferSeleccioDesac(indxbuff, arrsensor);

  });

};
```

L'última funció definida de l'aplicació és anomenada "BufferSelecciódesac" que compta amb dos camps d'entrada "index" i "llista", aquesta funció envia un Downlink a tots aquells actuadors que es troben a la llista "arrsensor" de la funció anterior "RevDevEUIdesac". Per realitzar aquestes peticions al servidor TTN, primer de tot es programa un bucle "for" que utilitza l'índex incorporat al camp d'entrada. Tot seguit guarda aquesta informació a les llistes

locals creades "arrcotxe" i "arrdwlnk", amb les variables "DevEUI_cotxe" i "DevEUI_led" respectivament.

Després de guardar les dades es defineix la variable "sended" amb el valor "AA==" que equival a un 0 en hexadecimal codificat amb "base64", per finalment executar la funció "axios". La funció "axios" es parametritza d'igual forma que l'esmentada a la funció "app.post("/downlink"..." (Downlink LoRa Proves). L'únic canvi es produeix dins la variable "url" on s'introdueix (concatena) la llista de l'actuador indexat "arrdwlnk[i]".

Finalment, es configura l'última funció "connectio.query" per realitzar una petició que modifica "UPDATE" el valor de la columna "SET Downlinks_sent = 0" de la taula "gestio_cotxe" on la columna "DevEUI_led" coincideixi amb el valor de "mysql.escape(arrdwlnk[i])". Si ha resultat satisfactòria, mostra el missatge "Downlink Desactivació per l'actuador (arrdwlnk[i]) actualitzat".

```
//Funció eliminatòria i downlink desactivació
function BufferSeleccioDesac(index, llista){
  let arrcotxe = []; // Array cotxes i leds assosiats
  let arrdwlnk = []; // Array leds
  //let arrdate = []; // Array dates dels cotxes
  //Creem les llistes per buscar temps i preparar downlinks
  for (let i = 0; i < index; i++) {
    arrcotxe[i] = llista[i].DevEUI_cotxe;
    arrdwlnk[i] = llista[i].DevEUI_led;
    const sended = 'AA==';
    axios({
      headers: {'Authorization': 'Bearer NNSXS.5SFW...',
      'Content-Type': 'application/json',
      'User-Agent': 'proves-cotxe/v3'},
      method: 'post',
          url:  'https://eu1.cloud.thethings.network/api/v3/as/applications/proves-
cotxe/webhooks/api-webhook-udg/devices/' + arrdwlnk[i] + '/down/push',
      data: {"downlinks": [{
        "frm_payload": sended,
        "f_port": 15,
```

```
        "priority":"NORMAL"
      }]
      }
    });
    //Finalment actualitzem el valor Downlink
    const sql = 'UPDATE Gestio_Cotxe SET Downlinks_sent  = 0 WHERE DevEUI_led =' +
mysql.escape(arrdwlnk[i]);
    //Guardem a la DB Gestio_Cotxe
    connection.query(sql, error =>{
    if (error) throw error;
      console.log('Downlink  Desactivació  per  l"actuador  '  +  arrdwlnk[i]  +  '
actualitzat');
    });
  };

};
```

Finalment, es criden les funcions "RevDevEUI()" i "RevDevEUIdesac()" cada vegada que es posa en marxa l'aplicació. D'aquesta manera executa una primera avaluació amb els últims registres que es van guardar a la base de dades abans de tancar l'API.

```
//Funció integrada a RevDevEUI que envia els downlinks d'activació i RevDevEUIdesac
per desactivacions
RevDevEUI();
RevDevEUIdesac();
```

8. RESUM PRESSUPOST

El quart document d'aquest projecte anomenat Estat d'Amidaments, documenta les hores i el material necessari per dur a terme la realització del Sistema de gestió d'aparcaments basat amb el sistema LoRa. El cinquè document, Pressupost, apareixen els preus de les hores de treball i el material utilitzat.

Un cop executades totes les tasques d'aquest projecte, el cost econòmic ascendeix a sis mil quatre-cents trenta-tres euros amb cinquanta-set cèntims, sense IVA.

9. CONCLUSIÓ

Durant la primera fase, la selecció de la gateway MultiTech model MTCDTIP-266A-868 mostra uns resultats excel·lents, tant en recepció com enviament del senyal. L'equipament, la instal·lació i la terrassa de l'edifici P.IV de l'EPS han demostrat dotar de molt bona cobertura el campus i una àmplia zona de la ciutat de Girona, quant a comunicacions LoRa, tal com indica l'estudi realitzat amb el prototip de l'actuador lumínic.

Amb la segona fase del projecte la detecció del cotxe es fonamenta amb el sensor AN-101D proveït per IOTSens, els resultats de les proves exercides i la posterior implementació al sistema ha sigut satisfactòria, tot i l'atenuació del rang.

L'actuador lumínic, prototip amb la placa ESP32U-01 del fabricant EZSBC i el xip RFM95W, respon a les ordres del servidor i la il·luminació es visualitza correctament, tot i ser un prototip dóna pas a la replicació de la tecnologia implementada per la seva industrialització. Els estudis realitzats ajuden a determinar l'ús i l'abast d'aquest sensor i actuador.

El servidor The Things Stack (TTN V3) ha permès la creació i integració de la xarxa LoRa amb protocol LoRaWAN entre els sensors i actuadors i el servidor creat a la universitat (LoRa + Webhook + API). Utilitzant les tècniques d'encriptació Lora (OTAA) garanteix una seguretat extra.

El servidor de la universitat gestiona l'emmagatzematge de les dades, la lògica implementada i les comunicacions entre aplicacions de l'aparcament, ho fa manera satisfactòria i segura gràcies al protocol HTTPS i el llenguatge JavaScript.

L'emmagatzematge que es fa sobre el servidor WampServer de MySQL, el qual reacciona de manera coherent i eficient. Grafana visualitza les dades emmagatzemades a la base de dades MySQL de manera versàtil i ràpida acord amb la finalitat del projecte i dona una experiència d'usuari atractiva.

Finalment, amb la tercera fase tot el sistema de gestió, TTN i API de servidors propis (JS, Wamp i Grafana) són de fàcil aplicació i escalabilitat gràcies a les funcions i lògica implementada en cadascun d'ells. El prototip realitzat per l'actuador lumínic Es pot donar pas a la possibilitat d'utilitzar la tecnologia implementada perquè pugui ser industrialitzada.

RICARD CASAS I DEL OLMO - DNI 41591773E

Firmado digitalmente por RICARD CASAS I DEL OLMO - DNI 41591773E
Fecha: 2022.06.07 17:16:21 +02'00'

Ricard Casas i del Olmo

Graduat en Enginyeria Electrònica i Automàtica

Amer, 7 de juny de 2022

10. RELACIÓ DE DOCUMENTS

El projecte consta i es regeix dels següents documents: Memòria, Plànols, Plec de condicions, Estat d'amidaments i Pressupost.

11. BIBLIOGRAFIA

Airtime Calculator, Calculadora Duty Cycle (https://avbentem.github.io/airtime-calculator/ttn/eu868/4 , 3 de maig de 2022)

EZSBC, Documentació placa (https://github.com/EzSBC/ESP32_Dev , 2 de maig de 2022)

EZSBC, Documentació programari (https://github.com/grillbaer/esp32-power-consumption-test , 2 de maig de 2022)

KYONG-HEE LEE, An Analsysis of an RF Link Budget and RSSI Ciruit Design for Long-Range Communications (https://ieeexplore.ieee.org/document/8436624 , 21 d'abril de 2022)

MCCI LoRaWAN LMIC, Llibreria comunicacions (https://forum.mcci.io/c/device-software/arduino-lmic/5 , 21 d'abril de 2022)

MESHARI ALJONHANI, STEPHAN OLARIU, ABRAR ALALI, SHUBHAN JAIN, A Survey of Parking Solutions for Smart Cities (https://ieeexplore.ieee.org/document/9552614/authors#authors , 25 de maig de 2022)

MUKUL KHANNA, Node.js Architecture and 12 Best Practices for Node.js Development (https://scoutapm.com/blog/nodejs-architecture-and-12-best-practices-for-nodejs-development , 3 de juny de 2022)

MySQL, Informació general (https://www.mysql.com/ , 21 d'abril de 2022)

NODE.JS, Informació general (https://nodejs.org/en/ , 13 d'abril de 2022)

NPM, Informació general (https://www.npmjs.com/ , 13 d'abril de 2022)

SEMTECH, Informació general (https://www.semtech.com , 11 d'abril de 2022)

SEMTECH, What are LoRa and LoRaWAN? (https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/ , 25 de maig de 2022)

THE THINGS NETWORK, LoRaWAN Documentació de les comunicacions (https://www.thethingsnetwork.org/docs/lorawan/ , 4 de maig de 2022)

12. GLOSSARI

AES: Advanced Encryption System

AFE: Analog Front End

API: Application Programming Interface

CSS: Chirp Spread Spectrum

I^2C: Inter-Integrated Circuit

LPWA: Low Power Wide Area

MAC: Media Access Control

MIC, Message Integrity Check

MQTT: Message Queuing Telemetry Transport

OS: Operating System

OSI: Open Sistem Interconect

PoE: Power Over Ethernet

RSSI: Recived Signal Strength indicator

RTC: Real Time Clock

SFx: Spread Factor x (Number)

SNMP: Simple Network Management Protocol

SPI: Serial Peripheral Interface

SSL: Secure Sockets Layer

TTI: The Things Industries

TTN: The Things Network

TTS: The Things Stack

WAMP: Windows Apache MySQL PHP

A. CODIS

Aquest annex encapsula la totalitat dels programes creats per aquest sistema, al servidor JavaScript, API de LoRa i actuador.

A.1 Codi servidor JavaScript

El següent codi realitza la funció de gestió i emmagatzematge del servidor principal.

```
////////////////////////////////////////////////////////////////////
//                                                                //
//                                                                //
//                  API Gestió dels aparcaments                   //
//                         Servidor UdG                           //
//                                                                //
////////////////////////////////////////////////////////////////////

require("dotenv").config();
const express = require("express");//Servidor HTTP
const axios = require("axios").default;//Client HTTP per enviar al TTN
//const schedule = require('node-schedule');

//Executem el servidor i client
const app = express();
const port = 40300;

//Base de dades MySQL
var mysql      = require('mysql');

var connection = mysql.createConnection({
  host     : 'localhost',
  user     : 'root',
  password : '',
  database : 'mydb'
});
```

```
connection.connect(error => {
  if (error) throw error;
  console.log('Database server running!!!');
});

//Declara l'estructura de rebuda com a JSON
app.use(express.json());

//Pàgina principal

app.get("/", (req, res) => res.send(`
  <html>
    <head><title>Success!</title></head>
    <body>
      <h1>Benvingut al servidor de gestió </h1>
      <img src="https://media.giphy.com/media/XreQmk7ETCak0/giphy.gif" alt="Cool
kid doing thumbs up" />
    </body>
  </html>
`
));

// Base de dades de Proves, Trastejar aquí
app.post("/provadb", async (req, res) =>{
  const sql = 'INSERT INTO Sensor_Cotxe1 SET ?';
  const cotxeObj = {
    //Data: req.body.uplink_message.received_at,
    Data: new Date(req.body.uplink_message.received_at),
    DevEUI: req.body.end_device_ids.device_id,
    Parking_status: req.body.uplink_message.decoded_payload.Parking_status,
    Battery_Voltage: req.body.uplink_message.decoded_payload.Battery_Voltage,
    Direction: req.body.uplink_message.decoded_payload.Direction,
    Frame_type: req.body.uplink_message.decoded_payload.Frame_type,
    Sens_type: req.body.uplink_message.decoded_payload.Sens_type,
    Temp: req.body.uplink_message.decoded_payload.Temp,
    X_Axis: req.body.uplink_message.decoded_payload.X_Axis,
    Y_Axis: req.body.uplink_message.decoded_payload.Y_Axis,
```

```
    Z_Axis: req.body.uplink_message.decoded_payload.Z_Axis,
  }
  //Guardem a la DB Sensor_Cotxe1
  connection.query(sql, cotxeObj, error =>{
    if (error) throw error;
    console.log('Sensor cotxe rebut');
    });

  const sql2 = 'SELECT DevEUI FROM Sensor_Cotxe1 ORDER BY Data DESC LIMIT 5';
  let variable = {};
  connection.query(sql2, (error, result) =>{
    if (error) throw error;
    variable = {result};
    console.log('Sensor cotxe rebut');
    res.status(200).send(variable);
    });
  console.log('------------------');

})

//Post missatges rebuts de la API LoRa, TTN V3
app.post("/sensor", async (req, res) =>{

  //Mirem si és sensor Cotxe o actuador LED
  if (req.body.uplink_message.hasOwnProperty('decoded_payload')){
    let sensor = req.body.uplink_message.decoded_payload.Type_Sens;
    if (sensor == 1){
      //Preparem dades del sensor cotxe
      const sql = 'INSERT INTO Sensor_Cotxe SET ?';
      const cotxeObj = {
        Data: new Date(req.body.uplink_message.received_at),
        DevEUI: req.body.end_device_ids.device_id,
        Parking_status: req.body.uplink_message.decoded_payload.Parking_status,
        Battery_Voltage: req.body.uplink_message.decoded_payload.Battery_Voltage,
        Direction: req.body.uplink_message.decoded_payload.Direction,
        Frame_type: req.body.uplink_message.decoded_payload.Frame_type,
        Sens_type: req.body.uplink_message.decoded_payload.Sens_type,
        Temp: req.body.uplink_message.decoded_payload.Temp,
```

```
        X_Axis: req.body.uplink_message.decoded_payload.X_Axis,
        Y_Axis: req.body.uplink_message.decoded_payload.Y_Axis,
        Z_Axis: req.body.uplink_message.decoded_payload.Z_Axis,
      }
      //Guardem a la Taula Sensor_Cotxe
      connection.query(sql, cotxeObj, error =>{
      if (error) throw error;
      console.log('Sensor cotxe ' + cotxeObj.DevEUI + ' rebut');
      });

      //Actualitzem la Taula Gestio_Cotxe
      const sql2 = 'UPDATE Gestio_Cotxe SET Parking_Status = ' +
mysql.escape(cotxeObj.Parking_status) + ' WHERE DevEUI_cotxe =' +
mysql.escape(cotxeObj.DevEUI);
      //Guardem a la DB Gestio_Cotxe
      connection.query(sql2, error =>{
      if (error) throw error;
      console.log('Sensor cotxe ' + cotxeObj.DevEUI + ' actualitzat');
      });

    }
    else if (sensor == 2){

        let ledEUI =req.body.end_device_ids.device_id;
        const ledObj = { Estat_led: req.body.uplink_message.decoded_payload.LED }
        //console.log(ledObj);
        //Preparem dades del sensor cotxe
        const sql = 'UPDATE Gestio_Cotxe SET Estat_led =' +
mysql.escape(ledObj.Estat_led) + ' WHERE DevEUI_led =' + mysql.escape(ledEUI);

        //Guardem a la Taula Gestio_Cotxe
        connection.query(sql, error =>{
        if (error) throw error;
        console.log('Actuador ' + ledEUI + ' led actualitzat');
        });
    }
  }

  res.status(200).send();
```

```
})

//Post Creació Postman dels sensors
app.post("/registre", async (req, res) =>{

  //Mirem si ja existeix
  let cotxe = req.body.EUICotxe;
  let c_cotxe = 'A';

  let led = req.body.EUILED;
  let c_led = 'A';

  //Al no ser async espera a recopilar l'informació i llavors decideix
  function Logica(c_cotxe, c_led){
    if (c_cotxe != cotxe & c_led != led) {
      //Preparem dades dela DB gestió cotxe
      const sql2 = `INSERT INTO Gestio_Cotxe SET ?`;
      const cotxeObj = {
        DevEUI_cotxe: req.body.EUICotxe,
        Parking_status: 0,
        Downlinks_sent: 0,
        DevEUI_led:req.body.EUILED,
        Estat_led:0,
      }
      //Guardem a la DB Sensor_Cotxe
      connection.query(sql2, cotxeObj, error =>{
      if (error) throw error;
      console.log('REGISTRAT!');
      });
    }
    else if ( c_cotxe == cotxe & c_led == led ) {
      console.log('El sensor del cotxe i led ja estan registrats!');
    }
    else if ( c_cotxe == cotxe & c_led != led ) {
      console.log('Sensor cotxe ja registrat!');
    }
    else if ( c_cotxe != cotxe & c_led == led ) {
      console.log('Actuador LED ja registrat!');
```

```
    }
    else {
      console.log('Error desconegut');
    }
  }

  //Preguntem a la base de dades si ja existeix algun camp
  const sql = 'SELECT * FROM Gestio_Cotxe WHERE DevEUI_cotxe = ' +
mysql.escape(cotxe) + 'OR DevEUI_led  = ' + mysql.escape(led);
  connection.query(sql, (error, result) =>{
    if (error) throw error;
    if (result.length > 0){
    c_cotxe = result[0].DevEUI_cotxe;
    c_led = result[0].DevEUI_led;
    }
    Logica(c_cotxe,c_led);
  });

  res.status(200).send();
})

//Downlink LoRa Proves
// 1 = AQ== ------ 0 = AA==
app.post("/downlink", async (req, res) =>{
  const sended = 'AA==';
  axios({
    headers: {'Authorization': 'Bearer NNSXS.5SFW...',
    'Content-Type': 'application/json',
    'User-Agent': 'proves-cotxe/v3'},
    method: 'post',
    url: 'https://eu1.cloud.thethings.network/api/v3/as/applications/proves-
cotxe/webhooks/api-webhook-udg/devices/' + 'eui-70b3d57ed004da1c' + '/down/push',
    data: {"downlinks": [{
      "frm_payload": sended,
      "f_port": 15,
      "priority":"NORMAL"
    }]
    }
  });
```

```
  res.status(200).send();
})


//----------Funcions repetitives de gestió lògica--------//


// Aquesta funció mira la taula on hi han registrats tots el Sensors i envia
l'ordre per aquells necessaris d'activar el LED


const BuffRev = setInterval(RevDevEUI, 30000);//Cada 30 segons
const BuffRevDesac = setInterval(RevDevEUIdesac, 30000);//Cada 30 segons


//Funció que revisa i  crea la llista que envia els downlinks d'ACTIVACIÓ
function RevDevEUI(){
  let indxbuff = 0;
  let arrsensor = [];
  const sql = 'SELECT * FROM Gestio_Cotxe WHERE Parking_Status = 1 AND
Downlinks_sent = 0 AND Estat_led = 0 ';
  connection.query(sql, (error, result) =>{
    if (error) throw error;
    if (result.length > 0){
      indxbuff = result.length;
      arrsensor = result;
      //console.log(arrsensor[0]);
      //console.log(indxbuff);
      //console.log(arrsensor);
    }
    BufferSeleccio(indxbuff, arrsensor);
  });


};


//Funció que mira si ja s'ha enviat Downlink o si el led està ACTIVAT
function BufferSeleccio(index, llista){
  let arrcotxe = []; // Array cotxes
  let arrdwlnk = []; // Array leds
  let arrdate = []; // Array dates dels cotxes
  //Creem les llistes per buscar temps i preparar downlinks
  for (let i = 0; i < index; i++) {
    arrcotxe[i] = llista[i].DevEUI_cotxe;
```

```
    arrdwlnk[i] = llista[i].DevEUI_led;
  };

  //console.log(arrcotxe);

  // Un cop sabem la llista dels sensors busquem quins han sobrepassat el temps
  for (let i = 0; i < index; i++) {
    let eui = arrcotxe[i];
    const sql = 'SELECT Data FROM Sensor_Cotxe WHERE DevEUI = ' + mysql.escape(eui)
+ ' AND Parking_status = 1 ORDER BY Data DESC LIMIT 1';
    connection.query(sql, (error, result) =>{
      if (error) throw error;
      if (result.length > 0){
        //let index = result.length;
        arrdate[i] = result[0].Data;

        const act = new Date(); // Hora actual
        const t_sens = new Date(arrdate[i]); // Hora del sensor

        //console.log((act - t_sens));
        let minuts = (act - t_sens)/60000;

        //Mirem si han passat els minuts
        if (minuts > 1){
          console.log('Downlink enviat');
          //console.log(arrcotxe[i]);
          //console.log(arrdwlnk[i]);
          const sended = 'AQ==';
          axios({
            headers: {'Authorization': 'Bearer NNSXS.5SFW...',
            'Content-Type': 'application/json',
            'User-Agent': 'proves-cotxe/v3'},
            method: 'post',
            url:
'https://eu1.cloud.thethings.network/api/v3/as/applications/proves-
cotxe/webhooks/api-webhook-udg/devices/' + arrdwlnk[i] + '/down/push',
            data: {"downlinks": [{
              "frm_payload": sended,
              "f_port": 15,
```

```
              "priority":"NORMAL"
            }]
            }

          });

          //Finalment actualitzem el valor Downlink
          const sql = 'UPDATE Gestio_Cotxe SET Downlinks_sent  = 1 WHERE DevEUI_led
=' + mysql.escape(arrdwlnk[i]);

          //Guardem a la DB Gestio_Cotxe
          connection.query(sql, error =>{
          if (error) throw error;
          console.log('Downlink pel actuador ' + arrdwlnk[i] + ' actualitzat');
          });

        };
      }
    });
  }

  /*const date = new Date();
  console.log(date -Date.now());*/

};

// Funció que revisa i crea els downlinks de DESACTIVACIÓ
//Aquesta funció no contempla el cas PS = 0 AND D_s = 1 AND E_l = 0 Espera que el
LED dongui resposta E_l = 1
function RevDevEUIdesac(){
  let indxbuff = 0;
  let arrsensor = [];
  const sql = 'SELECT * FROM Gestio_Cotxe WHERE Parking_Status = 0 AND
Downlinks_sent = 1 AND Estat_led = 1 ';
  connection.query(sql, (error, result) =>{
    if (error) throw error;
    if (result.length > 0){
      indxbuff = result.length;
      arrsensor = result;
```

```
      //console.log(arrsensor[0]);
      //console.log(indxbuff);
      //console.log(arrsensor);
    }
    BufferSeleccioDesac(indxbuff, arrsensor);
  });
};

//Funció eliminatòria i downlink desactivació
function BufferSeleccioDesac(index, llista){
  let arrcotxe = []; // Array cotxes i leds assosiats
  let arrdwlnk = []; // Array leds
  //let arrdate = []; // Array dates dels cotxes
  //Creem les llistes per buscar temps i preparar downlinks
  for (let i = 0; i < index; i++) {
    arrcotxe[i] = llista[i].DevEUI_cotxe;
    arrdwlnk[i] = llista[i].DevEUI_led;
    const sended = 'AA==';
    axios({
      headers: {'Authorization': 'Bearer NNSXS.5SFW...',
      'Content-Type': 'application/json',
      'User-Agent': 'proves-cotxe/v3'},
      method: 'post',
      url: 'https://eu1.cloud.thethings.network/api/v3/as/applications/proves-
cotxe/webhooks/api-webhook-udg/devices/' + arrdwlnk[i] + '/down/push',
      data: {"downlinks": [{
        "frm_payload": sended,
        "f_port": 15,
        "priority":"NORMAL"
      }]
      }
    });
    //Finalment actualitzem el valor Downlink
    const sql = 'UPDATE Gestio_Cotxe SET Downlinks_sent  = 0 WHERE DevEUI_led =' +
mysql.escape(arrdwlnk[i]);
    //Guardem a la DB Gestio_Cotxe
    connection.query(sql, error =>{
    if (error) throw error;
```

```
    console.log('Downlink Desactivació per l"actuador ' + arrdwlnk[i] + '
actualitzat');

    });

  };


};



//Funció integrada a RevDevEUI que envia els downlinks d'activació i RevDevEUIdesac
per desactivacions

RevDevEUI();

RevDevEUIdesac();


//-----------------------------------------------//


// Errors no descrits a l'aplicació

app.use((error, req, res, next) => {

  res.status(500)

  res.send({error: error})

  console.error(error.stack)

  next(error)

})


// Missatge d'avís, per la connexió del port

app.listen(port, () =>

  console.log(`Aplicació executada a la direcció http://localhost:${port}`)

);
```

A.2 Codi actuador lumínic

El següent codi és l'utilitzat per l'enviament i recepció de les dades per l'indicador lluminós.

```
//-------------------------------------------------------//
//                                                       //
//                  Actuador lumínic                     //
//                                                       //
//-------------------------------------------------------//


// Llibreria PlatformIO
#include <Arduino.h>


//Llibreries per LoRa
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>


//Llibreria RTC
#include <driver/rtc_io.h>


//Configuració RTC i Deep Sleep


RTC_DATA_ATTR lmic_t RTC_LMIC;
bool GOTO_DEEPSLEEP = false;


// APPEUI Little-endian
static const u1_t PROGMEM APPEUI[8]={ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}


// DEVEUI Little-endian
static const u1_t PROGMEM DEVEUI[8]={ 0xE2, 0x00, 0x05, 0xD0, 0x7E, 0xD5, 0xB3, 0x70
};
void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}


// APPKEY Big-endian
static const u1_t PROGMEM APPKEY[16] = { 0x02, 0xA4, 0x5C, 0x6A, 0xB7, 0x27, 0xE4,
0xD2, 0x92, 0x1C, 0x6E, 0x0F, 0x53, 0x07, 0x6D, 0xC1 };
```

```
void os_getDevKey (u1_t* buf) {  memcpy_P(buf, APPKEY, 16);}

// Data types variables
RTC_DATA_ATTR uint8_t payload[4];
static osjob_t sendjob;

// Duty Cycle enviament TX Uplink
const unsigned TX_INTERVAL = 52;

// Variable LLUM
#define LED_Vermell GPIO_NUM_33

// Pin mapping EZSBC i RFM95W
const lmic_pinmap lmic_pins = {
    .nss = 5,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 4,
    .dio = {2, 21, 22},
};

// Conversió dades
void printHex2(unsigned v) {
    v &= 0xff;
    if (v < 16)
        Serial.print('0');
    Serial.print(v, HEX);
}

// Enviament dades
void do_send(osjob_t* j){
    // Check if there is not a current TX/RX job running
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
    } else {

        Serial.println(payload[0]);
        Serial.println(payload[1]);
        Serial.println(payload[2]);
```

```
        Serial.println(payload[3]);

        // Preparem les dades per la següent transmisisó TX.
        LMIC_setTxData2(1, payload, sizeof(payload), 0);
        Serial.println(F("Packet queued"));
    }
}

// Event que respon segons l'estat de les comunicacions
void LoraWANPrintLMICOpmode(void)
{
    Serial.print(F("LMIC.opmode: "));
    if (LMIC.opmode & OP_NONE)
    {
        Serial.print(F("OP_NONE "));
    }
    if (LMIC.opmode & OP_SCAN)
    {
        Serial.print(F("OP_SCAN "));
    }
    if (LMIC.opmode & OP_TRACK)
    {
        Serial.print(F("OP_TRACK "));
    }
    if (LMIC.opmode & OP_JOINING)
    {
        Serial.print(F("OP_JOINING "));
    }
    if (LMIC.opmode & OP_TXDATA)
    {
        Serial.print(F("OP_TXDATA "));
    }
    if (LMIC.opmode & OP_POLL)
    {
        Serial.print(F("OP_POLL "));
    }
    if (LMIC.opmode & OP_REJOIN)
    {
```

```
        Serial.print(F("OP_REJOIN "));
    }
    if (LMIC.opmode & OP_SHUTDOWN)
    {
        Serial.print(F("OP_SHUTDOWN "));
    }
    if (LMIC.opmode & OP_TXRXPEND)
    {
        Serial.print(F("OP_TXRXPEND "));
    }
    if (LMIC.opmode & OP_RNDTX)
    {
        Serial.print(F("OP_RNDTX "));
    }
    if (LMIC.opmode & OP_PINGINI)
    {
        Serial.print(F("OP_PINGINI "));
    }
    if (LMIC.opmode & OP_PINGABLE)
    {
        Serial.print(F("OP_PINGABLE "));
    }
    if (LMIC.opmode & OP_NEXTCHNL)
    {
        Serial.print(F("OP_NEXTCHNL "));
    }
    if (LMIC.opmode & OP_LINKDEAD)
    {
        Serial.print(F("OP_LINKDEAD "));
    }
    if (LMIC.opmode & OP_LINKDEAD)
    {
        Serial.print(F("OP_LINKDEAD "));
    }
    if (LMIC.opmode & OP_TESTMODE)
    {
        Serial.print(F("OP_TESTMODE "));
    }
```

```
    if (LMIC.opmode & OP_UNJOIN)
    {
        Serial.print(F("OP_UNJOIN "));
    }
}

//Funció que avisa de l'estat LoRa
void LoraWANDebug(lmic_t lmic_check)
{
    Serial.println("");
    Serial.println("");

    LoraWANPrintLMICOpmode();
    Serial.println("");

    Serial.print(F("LMIC.seqnoUp = "));
    Serial.println(lmic_check.seqnoUp);

    Serial.print(F("LMIC.globalDutyRate = "));
    Serial.print(lmic_check.globalDutyRate);
    Serial.print(F(" osTicks, "));
    Serial.print(osticks2ms(lmic_check.globalDutyRate) / 1000);
    Serial.println(F(" sec"));

    Serial.print(F("LMIC.globalDutyAvail = "));
    Serial.print(lmic_check.globalDutyAvail);
    Serial.print(F(" osTicks, "));
    Serial.print(osticks2ms(lmic_check.globalDutyAvail) / 1000);
    Serial.println(F(" sec"));

    Serial.print(F("LMICbandplan_nextTx = "));
    Serial.print(LMICbandplan_nextTx(os_getTime()));
    Serial.print(F(" osTicks, "));
    Serial.print(osticks2ms(LMICbandplan_nextTx(os_getTime())) / 1000);
    Serial.println(F(" sec"));

    Serial.print(F("os_getTime = "));
    Serial.print(os_getTime());
```

```
    Serial.print(F(" osTicks, "));
    Serial.print(osticks2ms(os_getTime()) / 1000);
    Serial.println(F(" sec"));

    Serial.print(F("LMIC.txend = "));
    Serial.println(lmic_check.txend);
    Serial.print(F("LMIC.txChnl = "));
    Serial.println(lmic_check.txChnl);

    Serial.println(F("Band \tavail \t\tavail_sec\tlastchnl \ttxcap"));
    for (u1_t bi = 0; bi < MAX_BANDS; bi++)
    {
        Serial.print(bi);
        Serial.print("\t");
        Serial.print(lmic_check.bands[bi].avail);
        Serial.print("\t\t");
        Serial.print(osticks2ms(lmic_check.bands[bi].avail) / 1000);
        Serial.print("\t\t");
        Serial.print(lmic_check.bands[bi].lastchnl);
        Serial.print("\t\t");
        Serial.println(lmic_check.bands[bi].txcap);
    }
    Serial.println("");
    Serial.println("");
}

// Funció Per saber el temps d'execució
void PrintRuntime()
{
    long seconds = millis() / 1000;
    Serial.print("Runtime: ");
    Serial.print(seconds);
    Serial.println(" seconds");
}

// Funció per guardar les dades correctes
void SaveLMICToRTC(int deepsleep_sec)
{
```

```
    Serial.println(F("Guardant LMIC to RTC"));
    RTC_LMIC = LMIC;

    // Reset DutyCyles
    unsigned long now = millis();

    // EU Like Bands
    #if defined(CFG_LMIC_EU_like)
        Serial.println(F("Reset CFG_LMIC_EU_like band avail"));
        for (int i = 0; i < MAX_BANDS; i++)
        {
             ostime_t correctedAvail = RTC_LMIC.bands[i].avail - ((now / 1000.0 +
deepsleep_sec) * OSTICKS_PER_SEC);
            if (correctedAvail < 0)
            {
                correctedAvail = 0;
            }
            RTC_LMIC.bands[i].avail = correctedAvail;
        }

          RTC_LMIC.globalDutyAvail = RTC_LMIC.globalDutyAvail - ((now / 1000.0 +
deepsleep_sec) * OSTICKS_PER_SEC);
        if (RTC_LMIC.globalDutyAvail < 0)
        {
            RTC_LMIC.globalDutyAvail = 0;
        }
    #else
        Serial.println(F("No DutyCycle recalculation function!"));
    #endif
}

// Funció per carregar la informació després de despertar
void LoadLMICFromRTC()
{
    Serial.println(F("Carregar LMIC des de RTC"));
    LMIC = RTC_LMIC;
}

// Funció per dormir el dispositiu
```

```
void GoDeepSleep()
{
    // DEEPSLEEP
    Serial.println(F("Go DeepSleep"));
    // Mirem quan ha durat la retrsansmissió
    PrintRuntime();
    //Apaguem el Serial
    Serial.flush();
    // Posem a dormir
    ESP.deepSleep(TX_INTERVAL * 1000000);
}

// Funció que respon segons l'estat de les comunicacions
void onEvent (ev_t ev) {
    Serial.print(os_getTime());
    Serial.print(": ");
    switch(ev) {
        case EV_SCAN_TIMEOUT:
            Serial.println(F("EV_SCAN_TIMEOUT"));
            break;
        case EV_BEACON_FOUND:
            Serial.println(F("EV_BEACON_FOUND"));
            break;
        case EV_BEACON_MISSED:
            Serial.println(F("EV_BEACON_MISSED"));
            break;
        case EV_BEACON_TRACKED:
            Serial.println(F("EV_BEACON_TRACKED"));
            break;
        case EV_JOINING:
            Serial.println(F("EV_JOINING"));
            break;
        case EV_JOINED:
            Serial.println(F("EV_JOINED"));
            {
              u4_t netid = 0;
              devaddr_t devaddr = 0;
              u1_t nwkKey[16];
```

```
      u1_t artKey[16];
      LMIC_getSessionKeys(&netid, &devaddr, nwkKey, artKey);
      Serial.print("netid: ");
      Serial.println(netid, DEC);
      Serial.print("devaddr: ");
      Serial.println(devaddr, HEX);
      Serial.print("AppSKey: ");
      for (size_t i=0; i<sizeof(artKey); ++i) {
        if (i != 0)
          Serial.print("-");
        printHex2(artKey[i]);
      }
      Serial.println("");
      Serial.print("NwkSKey: ");
      for (size_t i=0; i<sizeof(nwkKey); ++i) {
              if (i != 0)
                      Serial.print("-");
              printHex2(nwkKey[i]);
      }
      Serial.println();
    }
    LMIC_setLinkCheckMode(0);
    break;

//|| case EV_RFU1:
//||     Serial.println(F("EV_RFU1"));
//||     break;
//
case EV_JOIN_FAILED:
    Serial.println(F("EV_JOIN_FAILED"));
    break;
case EV_REJOIN_FAILED:
    Serial.println(F("EV_REJOIN_FAILED"));
    break;
case EV_TXCOMPLETE:
    Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
    if (LMIC.txrxFlags & TXRX_ACK)
      Serial.println(F("Received ack"));
```

```
    if (LMIC.dataLen) {
      Serial.print(F("Received "));
      Serial.print(LMIC.dataLen);
      Serial.println(F(" bytes of payload"));

      //------ Recepció Dades Downlink ----------------

      if (LMIC.dataLen == 1) {
        uint8_t result = LMIC.frame[LMIC.dataBeg + 0];
        if (result == 0)  {
            Serial.println("RESULT 0");
            rtc_gpio_hold_dis(LED_Vermell); //Deshabilitar el PIN RTC
            rtc_gpio_set_level(LED_Vermell, LOW); //set high/low
            rtc_gpio_hold_en(LED_Vermell); //Habilitar el pin RTC
            payload[0] = 0;
        }
        if (result == 1)  {
            Serial.println("RESULT 1");
            rtc_gpio_hold_dis(LED_Vermell); //Deshabilitar el PIN RTC
            rtc_gpio_set_level(LED_Vermell, HIGH); //set high/low
            rtc_gpio_hold_en(LED_Vermell); //Habilitar el pin RTC
            payload[0] = 1;
        }
      }
     Serial.println();
    }

    // Activem la variable per la següent iteració anar a dormir
    GOTO_DEEPSLEEP = true;

    break;
case EV_LOST_TSYNC:
    Serial.println(F("EV_LOST_TSYNC"));
    break;
case EV_RESET:
    Serial.println(F("EV_RESET"));
    break;
case EV_RXCOMPLETE:
```

```
            // Dades rebudes
            Serial.println(F("EV_RXCOMPLETE"));
            break;
        case EV_LINK_DEAD:
            Serial.println(F("EV_LINK_DEAD"));
            break;
        case EV_LINK_ALIVE:
            Serial.println(F("EV_LINK_ALIVE"));
            break;

        //|| case EV_SCAN_FOUND:
        //||    Serial.println(F("EV_SCAN_FOUND"));
        //||    break;

        case EV_TXSTART:
            Serial.println(F("EV_TXSTART"));
            break;
        case EV_TXCANCELED:
            Serial.println(F("EV_TXCANCELED"));
            break;
        case EV_RXSTART:
            //No escriure res o el timing no anirà bé
            break;
        case EV_JOIN_TXCOMPLETE:
            Serial.println(F("EV_JOIN_TXCOMPLETE: no JoinAccept"));
            break;

        default:
            Serial.print(F("Unknown event: "));
            Serial.println((unsigned) ev);
            break;
    }
}

void setup() {
    setCpuFrequencyMhz(10); // Millora de 80 mA a 30 mA
    delay(10);
```

```
    Serial.begin(115200);
    Serial.println(F("Starting"));

    //Configurar el pin LED a la memòria RTC
    rtc_gpio_init(LED_Vermell); //Inicialitzem el PIN GPIO RTC
    rtc_gpio_set_direction(LED_Vermell, RTC_GPIO_MODE_OUTPUT_ONLY); //Definim el mode
RTC del PIN

    // LMIC init
    os_init();
    // Reset MAC
    LMIC_reset();

    //Revisem si cal volcar la informació guardada dins la memòria RTC
    if (RTC_LMIC.seqnoUp != 0)
    {
        LoadLMICFromRTC();
    }
    //S'executa per la realització de proves i saber el seu estat
    LoraWANDebug(LMIC);

    // Start OTTA
    do_send(&sendjob);

}

void loop() {

    static unsigned long lastPrintTime = 0;

    //Funció LoRa
    os_runloop_once();

    //Funció Intermitent que revisa si cal anar a dormir
                        const           bool           timeCriticalJobs           =
os_queryTimeCriticalJobs(ms2osticksRound((TX_INTERVAL * 1000)));
    if (!timeCriticalJobs && GOTO_DEEPSLEEP == true && !(LMIC.opmode & OP_TXRXPEND))
    {
```

```
        Serial.print(F("Anem a dormir "));
        LoraWANPrintLMICOpmode();
        SaveLMICToRTC(TX_INTERVAL);
        GoDeepSleep();
    }
    else if (lastPrintTime + 2000 < millis())
    {
        Serial.print(F("No pot dormir "));
        Serial.print(F("TimeCriticalJobs: "));
        Serial.print(timeCriticalJobs);
        Serial.print(" ");

        LoraWANPrintLMICOpmode();
        PrintRuntime();
        lastPrintTime = millis();
    }
}
```

A continuació es drecriu el codi de configuració de la xarxa dins la llibreria "lmic" del directori "project_config" document "lmic_project_config.h"

```
// project-specific definitions
#define CFG_eu868 1
//#define CFG_us915 1
//#define CFG_au915 1
//#define CFG_as923 1
// #define LMIC_COUNTRY_CODE LMIC_COUNTRY_CODE_JP      /* for as923-JP; also define
CFG_as923 */
//#define CFG_kr920 1
//#define CFG_in866 1

#define CFG_sx1276_radio 1
//#define LMIC_USE_INTERRUPTS
```

A.3 Codi servidor TTN V3 JavaScript

Els codis de codificació i descodificació de la xarxa LoRa queden esmentats a continuació.

Primer de tot el sensor d'aparcament i el seu codi de descodificació i tractament de les dades.

```
function decodeUplink(input) {
    var data = {};
    //data.bytes = input.bytes;
    data.Sens_type = input.bytes[0];
    data.Frame_type = (input.bytes[1] & 0xF0) >> 4;
    data.Direction =input.bytes[1] & 0x01;
    data.X_Axis = (((input.bytes[2] << 8) + input.bytes[3])-32768)/4096;
    data.Y_Axis = (((input.bytes[4] << 8) + input.bytes[5])-32768)/4096;
    data.Z_Axis = (((input.bytes[6] << 8) + input.bytes[7])-32768)/4096;
    data.Temp = ((input.bytes[8] << 8) + input.bytes[9])/100;
    data.Parking_status = (input.bytes[10] & 0x80) >> 7;
    data.Battery_Voltage = (input.bytes[10] & 0x7F)/10;
    data.Type_Sens = 1;

    return {
      data: data,
      warnings: [],
      errors: []
    };
  }
```

A continuació el codi de l'actuador lumínic.

```
function decodeUplink(input) {
  return {
    data: {
      LED: input.bytes[0],
      Type_Sens: 2
    },
```

```
    warnings: [],
    errors: []
  };
}
```

A.4 Codi taulell Grafana

Els codis aplicats a la configuració dels diferents gràfics i taules utilitzades queden reflectits a continuació i relacionats amb el nom de cadascun d'aquests. Primer el desplegable Estat Aparcament.

Estat Aparcament:

```
SELECT
  $__timeGroupAlias(Data,$__interval),
  avg(Parking_status) AS "Parking_status"
FROM sensor_cotxe
WHERE
  $__timeFilter(Data) AND
  DevEUI = "eui-ffffff1000027d34"
GROUP BY 1
ORDER BY $__timeGroup(Data,$__interval)
```

Panell Horari:

```
SELECT
  $__timeGroupAlias(Data,$__interval),
  avg(Parking_status) AS "Parking_status"
FROM sensor_cotxe
WHERE
  $__timeFilter(Data) AND
  DevEUI = "eui-ffffff1000027d34"
GROUP BY 1
ORDER BY $__timeGroup(Data,$__interval)
```

Gràfic:

```
SELECT
  $__timeGroupAlias(Data,$__interval),
  avg(Parking_status) AS "Parking_status"
FROM sensor_cotxe
WHERE
  $__timeFilter(Data) AND
  DevEUI = "eui-ffffff1000027d34"
GROUP BY 1
ORDER BY $__timeGroup(Data,$__interval)
```

A continuació el desplegable Manteniment Sensor.

Gràfic de la Bateria:

```
SELECT
  $__timeGroupAlias(Data,$__interval),
  avg(Battery_Voltage) AS "Battery_Voltage"
FROM sensor_cotxe
WHERE
  $__timeFilter(Data) AND
  DevEUI = "eui-ffffff1000027d34"
GROUP BY 1
ORDER BY $__timeGroup(Data,$__interval)
```

Voltatge de la bateria:

```
SELECT
  $__timeGroupAlias(Data,$__interval),
  avg(Battery_Voltage) AS "Battery_Voltage"
FROM sensor_cotxe
WHERE
  $__timeFilter(Data)
GROUP BY 1
ORDER BY $__timeGroup(Data,$__interval)
```

Temperatura:

```
SELECT
  $__timeGroupAlias(Data,$__interval),
  avg(Temp) AS "Temp"
FROM sensor_cotxe
WHERE
  $__timeFilter(Data) AND
  DevEUI = "eui-ffffff1000027d34"
GROUP BY 1
ORDER BY $__timeGroup(Data,$__interval)
```