**Taula de continguts**

[1 Introducció 1](#_Toc188189827)

[2 Marc teòric 2](#_Toc188189828)

[2.1 ESP32 2](#_Toc188189829)

[2.2 LoRa 2](#_Toc188189830)

[2.2.1 Paràmetres LoRa 3](#_Toc188189831)

[2.3 LoRaWAN 4](#_Toc188189832)

[2.3.1 Classes de LoRaWAN 5](#_Toc188189833)

[2.3.2 Modes d’activació 5](#_Toc188189834)

[2.4 Duty Cycle vs Listen Before Talk 6](#_Toc188189835)

[3 Mètodes i material utilitzat 7](#_Toc188189836)

[3.1 Software i hardware utilitzat 7](#_Toc188189837)

[3.1.1 Hardware utilitzat 7](#_Toc188189838)

[3.1.2 Software Utilitzat 8](#_Toc188189839)

[3.2 SetUp inicial 9](#_Toc188189840)

[3.2.1 Instal·lació de l’entorn 9](#_Toc188189841)

[3.2.2 Instal·lació placa 9](#_Toc188189842)

[3.2.3 Instal·lació de llibreries necessàries 11](#_Toc188189843)

[3.3 Estudi llibreries LoRa 12](#_Toc188189844)

[3.3.1 Unofficial Heltec 12](#_Toc188189845)

[3.3.2 Arduino LoRa 12](#_Toc188189846)

[3.3.3 RadioLib 13](#_Toc188189847)

[3.3.4 Comparativa entre llibreries LoRa 13](#_Toc188189848)

[3.4 Estudi llibreries LoRaWAN 15](#_Toc188189849)

[3.4.1 Beelan-LoRaWAN 15](#_Toc188189850)

[3.4.2 Arduino LoRaWAN 15](#_Toc188189851)

[3.4.3 Comparativa entre llibreries LoRaWAN 15](#_Toc188189852)

[3.5 Estalvi d’energia i modes Sleep 16](#_Toc188189853)

[3.5.1 Modes d’energia 16](#_Toc188189854)

[3.6 Guardar dades en memòria no-volàtil i memòria RTC 18](#_Toc188189855)

[3.6.1 Memòria no-volàtil 18](#_Toc188189856)

[3.6.2 Memòria RTC 18](#_Toc188189857)

[3.7 Configuració del Wi-Fi mitjançant BLE 19](#_Toc188189858)

[3.7.1 Utilització de BLE 19](#_Toc188189859)

[3.7.2 Seguretat en la transmissió BLE 20](#_Toc188189860)

[3.8 Accés al medi 20](#_Toc188189861)

[3.8.1 CSMA 20](#_Toc188189862)

[3.8.2 TDMA 20](#_Toc188189863)

[3.9 Càlcul del SF òptim 21](#_Toc188189864)

[3.10 Càlcul Time on Air i Duty Cycle 21](#_Toc188189865)

[4 Resultats 22](#_Toc188189866)

[4.1 Implementació llibreria LoRa - RadioLib 22](#_Toc188189867)

[4.1.1 Paràmetres utilitzats 22](#_Toc188189868)

[4.1.2 Estudi funcionament CAD 23](#_Toc188189869)

[4.1.3 Obtenció SNR i RSSI d’un missatge 28](#_Toc188189870)

[4.2 Implementació LoRaWAN – Arduino LoRaWAN 29](#_Toc188189871)

[4.3 Configuració Wi-Fi mitjançant BLE 31](#_Toc188189872)

[4.4 Guardar dades en memòria no volàtil 31](#_Toc188189873)

[*4.4.1* Proves realitzades amb la llibreria *Preferences* 31](#_Toc188189874)

[4.5 Estalvi d’energia i modes *sleep* 32](#_Toc188189875)

[*4.5.1* Mode *light\_sleep* 32](#_Toc188189876)

[*4.5.2* Mode *deep\_sleep* 34](#_Toc188189877)

[4.6 Càlcul SF òptim 36](#_Toc188189878)

[4.7 Implementació protocol final 36](#_Toc188189879)

[4.7.1 Escenari 36](#_Toc188189880)

[4.7.2 Tipus de missatges 36](#_Toc188189881)

[4.7.3 Càlcul ToA de cada missatge 38](#_Toc188189882)

[4.7.4 Disseny protocol 40](#_Toc188189883)

[4.8 Proves de validació del protocol 45](#_Toc188189884)

[4.8.1 Anàlisis temps entre missatges 45](#_Toc188189885)

[4.8.2 Anàlisis temps en *deep\_sleep* dels nodes 46](#_Toc188189886)

[4.8.3 Anàlisi funcionament amb varis nodes 47](#_Toc188189887)

[5 Discussió 50](#_Toc188189888)

[5.1 Validació funcionament protocol 50](#_Toc188189889)

[5.1.1 Temps entre missatges 50](#_Toc188189890)

[*5.1.2* Temps en mode *deep\_sleep* 51](#_Toc188189891)

[5.1.3 Funcionament amb varis nodes 51](#_Toc188189892)

[5.2 Possibles millores 51](#_Toc188189893)

[5.2.1 Col·lisió missatges Request 51](#_Toc188189894)

[5.2.2 ID dinàmic dels nodes 51](#_Toc188189895)

[5.2.3 Un codi unificat per els nodes i gateway 51](#_Toc188189896)

[5.2.4 Implementació del suport per varis gateway 51](#_Toc188189897)

[5.2.5 Confirmació recepció missatges 52](#_Toc188189898)

[5.3 Comparació utilització Duty Cycle vs Listen Before Talk 52](#_Toc188189899)

[5.4 Comparativa amb altres propostes realitzades 52](#_Toc188189900)

[5.5 Estudi Sostenibilitat 53](#_Toc188189901)

[5.5.1 Desenvolupament del treball final d’estudis 53](#_Toc188189902)

[5.5.2 Execució del projecte 53](#_Toc188189903)

[5.5.3 Riscos i limitacions 54](#_Toc188189904)

[6 Conclusions i treball futur 56](#_Toc188189905)

[7 Bibliografia 57](#_Toc188189906)

**Índex de figures**

[Figura 2.1 - SoC ESP32 2](#_Toc188121635)

[Figura 2.2 - Exemple de Coding Rate 4](#_Toc188121636)

[Figura 2.3 - Trama LoRaWAN 5](#_Toc188121637)

[Figura 3.1 - Instal·lació suport plaques ESP32 10](#_Toc188121638)

[Figura 3.2 - Comprovació instal·lació ESP32 11](#_Toc188121639)

[Figura 3.3 - Instal·lació llibreries 11](#_Toc188121640)

[Figura 3.4 - Mode Actiu 16](#_Toc188121641)

[Figura 3.5 - Mode Light Sleep 17](#_Toc188121642)

[Figura 3.6 - Mode Deep Sleep 17](#_Toc188121643)

[Figura 3.7 - Mode Hibernació 18](#_Toc188121644)

[Figura 4.1 - Mesures tems detecció CAD 26](#_Toc188121645)

[Figura 4.2 - Taxa d'error CAD 28](#_Toc188121646)

[Figura 4.3 - Missatge d'error LoRaWAN 29](#_Toc188121647)

[Figura 4.4 - Configuració TTN 30](#_Toc188121648)

[Figura 4.5 - Configuració Wi-Fi mitjançant BLE 31](#_Toc188121649)

[Figura 4.6 - Reutilització dades NVP 32](#_Toc188121650)

[Figura 4.7 - Mesures *light-sleep* 33](#_Toc188121651)

[Figura 4.8 - Mesures *deep\_sleep* 35](#_Toc188121652)

[Figura 4.9 - Capçalera missatges 37](#_Toc188121653)

[Figura 4.10 - Estructura Beacon 37](#_Toc188121654)

[Figura 4.11 - Estructura Request 37](#_Toc188121655)

[Figura 4.12 - Estructura Schedule 38](#_Toc188121656)

[Figura 4.13 - Estructura Data 38](#_Toc188121657)

[Figura 4.14 - Etapa de descobriment 41](#_Toc188121658)

[Figura 4.15 - Etapa d'organització 42](#_Toc188121659)

[Figura 4.16 - Etapa enviament de dades 43](#_Toc188121660)

[Figura 4.17 - Diagrama de flux protocol 44](#_Toc188121661)

[Figura 4.18 - Implementació protocol - Gateway 48](#_Toc188121662)

[Figura 4.19 - Implementació protocol - Node 1 49](#_Toc188121663)

[Figura 4.20 - Implementació protocol - Node 2 49](#_Toc188121664)

[Figura 4.21 - Solució col·lisions missatges Request 50](#_Toc188121665)

**Índex de taules**

[Taula 3.1 - Llistat de mòduls Heltec 12](#_Toc188121616)

[Taula 3.2 - Llistat de mòduls Arduino LoRa 12](#_Toc188121617)

[Taula 3.3 - Llistat de mòduls RadioLib 13](#_Toc188121618)

[Taula 3.4 - Taula comparativa llibreries LoRa 14](#_Toc188121619)

[Taula 3.5 - Paràmetres extra a tenir en compte en llibreries LoRa 14](#_Toc188121620)

[Taula 3.6 - Llistat de mòduls Beelan 15](#_Toc188121621)

[Taula 3.7 - Llistat de mòduls Arduino LoRaWAN 15](#_Toc188121622)

[Taula 3.8 - Taula comparativa llibreries LoRaWAN 15](#_Toc188121623)

[Taula 3.9 - Paràmetres extra a tenir en compte en llibreries LoRaWAN 16](#_Toc188121624)

[Taula 3.10 - Relació SF - SNR 21](#_Toc188121625)

[Taula 3.11 - Relació SF - RSSI 21](#_Toc188121626)

[Taula 4.1 - ToA Beacon 39](#_Toc188121627)

[Taula 4.2 - ToA Request 39](#_Toc188121628)

[Taula 4.3 - ToA Schedule 39](#_Toc188121629)

[Taula 4.4 - ToA Data 40](#_Toc188121630)

[Taula 4.5 - Resultats mesures temps entre missatges 45](#_Toc188121631)

[Taula 4.6 - Comparació valors teòrics amb valors mesurats 46](#_Toc188121632)

[Taula 4.7 - Comparació *deep\_sleep entre missatges Schedule i Data* 47](#_Toc188121633)

[Taula 4.8 - Comparació deep\_sleep entre missatges Data i Beacon 47](#_Toc188121634)

**Glossari**

AES Advanced Encryption Standard

AFA Adaptative Frequency Agility

BLE Bluetooth Low Energy

BW Bandwidth

CR Coding Rate

CRC Cyclic Redundancy Check

DC Duty Cycle

ETSI European Telecommunications Standard Institute

GHz Giga Hertz

IEEE Institute of Electrical and Electronics Engineers

IoT Internet of Things

ISM Sndustrial, Scientific and Medical

LBT Listen Before Talk

LoRa Long Range

LoRaWAN Long Range Wide Area Network

MHz Mega Hertz

SF Spreding Factor

SoC System on Chip

ToA Time on Air

# Introducció

El motiu inicial d’aquest treball, és desenvolupar una solució de ciència ciutadana mitjançant una plataforma IoT (Internet of Things) que proporcioni vàries solucions de connectivitat.

S’entén per ciència ciutadana tota aquella recerca científica conduïda per científics no professionals. Els projectes involucrats en la ciència ciutadana, s’orienten a la generació de nou coneixement amb la participació activa i imprescindible de la ciutadania en alguna etapa del procés de recerca del projecte.

Actualment, existeixen moltes solucions IoT, mitjançant protocols com CoAP, MQTT o Zigbee, però aquests queden fora de l’abast d’aquest treball. Gran part d’aquest treball, es centra en el desenvolupament d’un protocol basat en LoRa, per proporcionar una nova solució de connectivitat, mantenint els avantatges que proporciona, com la connectivitat a llarga distància i el poc espai que pesen els paquets, i afegint sincronisme entre nodes i controls d’accés al medi.

L’objectiu final d’aquest projecte, més enllà d’aquest treball, és el desenvolupament d’un producte comercial que proporcioni diverses possibilitats de connectivitat, dintre d’un entorn robust, que permeti comunicar-se de diverses maneres amb una estació meteorològica. Per tant, en aquest treball, com estat inicial d’aquest projecte, s’ha definit com a objectiu desenvolupar una plataforma que suporti diferents tecnologies de connectivitat de cara a facilitar la seva configuració i posterior connexió a Internet per recollir les dades generades. Per aconseguir aquest objectiu, s’han definit com objectiu dissenyar i desenvolupar una prova de concepte d’un protocol IoT que permeti la connectivitat entre diversos nodes. Per altra banda, també s’ha definit com a objectiu investigar i implementar solucions d’estalvi d’energia, solucions per una eventual configuració dels nodes de forma simple i la possibilitat de guardar dades en memòria no volàtil per evitar perdre dades de configuració en possibles pèrdues d’energia.

En aquesta memòria, s’explicaran els estudis i proves realitzades, s’aclariran les raons de la manera en que s’han procedit, i s’exposaran les limitacions i els problemes trobats durant el desenvolupament del treball.

# Marc teòric

## ESP32

Els ESP32 són una família de microcontroladors molt utilitzats en entorns IoT, creats per l’empresa Espressif Systems. Una de les característiques més importants dels ESP32 en comparació amb altres microcontroladors, és la incorporació pel suport de Wi-Fi i Bluetooth. Cal destacar la gran varietat d’opcions de connectivitat sense fils que ofereixen els ESP32, com per exemple, connectivitat Wi-Fi compatible amb l’estàndard IEEE 802.11 b/g/n en la banda de 2,4 GHz, i el suport de Bluetooth v4.2 i BLE (Bluetooth Low Energy).

Els ESP32 són una sèrie de SoC (System on Chip). En la següent figura, podem observar com són els blocs definits en el SoC del ESP32.

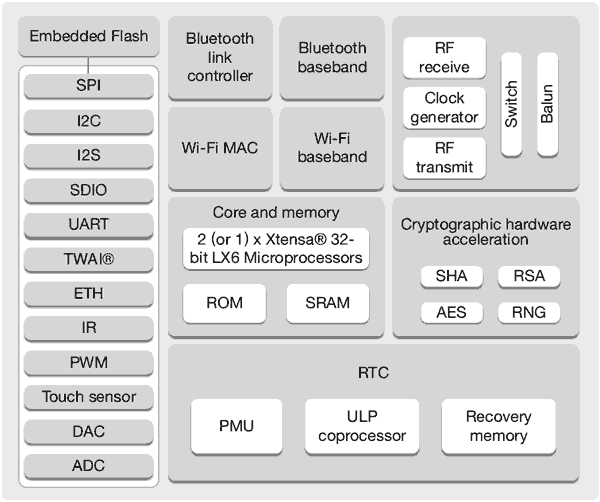


Figura 2.1 - SoC ESP32

Per altra banda, els ESP32 porten integrats dos microprocessadors de baix consum, a més d’un coprocessador d'ultra baix consum, utilitzat per realitzar funcions en modes d'estalvi d’energia.

## LoRa

LoRa (Long Range) és un protocol de comunicacions sense fils que funciona mitjançant ràdio-freqüència, i que permet establir comunicació entre nodes amb un abast de fins a 10 km en zones urbanes amb una potència de transmissió molt baixa. Actualment, LoRa és una tecnologia propietària i mantinguda per l’empresa Semtech ®.

LoRa treballa a la capa física i a la capa d’enllaç i utilitza una banda sense llicència ISM. Aquestes bandes són: 433 MHz per Àsia, 868 MHz per Europa, i 915 MHz per Amèrica. Per altra banda, fa servir una topologia de xarxa en forma d'estrella, la qual proporciona simplicitat en les comunicacions.

### Paràmetres LoRa

A l’hora de fer servir LoRa es poden configurar alguns paràmetres per tal de garantir una òptima utilització. Aquests paràmetres es mostren a la Taula 3.1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SF** | **BW (kHz)** | **CRC** | **Payload** | **Coding Rate** |
| 7 | 7,8 | Si | Fins a 255 Bytes | 4/5 |
| 20 |
| 8 | 41 | 4/6 |
| 9 | 62 | 4/7 |
| 10 | 125 | No | 4/8 |
| 11 | 250 |
| 12 | 500 |

Taula 2.1: Paràmetres editables LoRa

#### Ampla de Banda (BW)

LoRa permet configurar una gran varietat d’amplades de banda de radiofreqüència. Habitualment, una modulació ocupa una amplada de banda més o menys determinada, però donat que LoRa utilitza una modulació CSS (Chirp Spread Spectrum), distribueix el senyal en una amplada de banda més extensa que l’estrictament necessari per transmetre la informació. Tot això, permet suportar el soroll molt millor i fer servir potències de transmissió molt baixes.

Com s’ha esmentat anteriorment, LoRa pot ser configurat amb una varietat d’amplades de banda, des de 7,8 kHz, fins a 500 kHz. Utilitzar una amplada de banda més petita implicarà tenir una velocitat de transferència més lenta, però augmentarà la cobertura.

#### Spread Factor (SF)

El Spread Factor (SF) és el factor de propagació, és el paràmetre que controla la dispersió temporal de cada bit transmès. Si s’utilitza un SF més alt, cada bit es distribuirà en més símbols i els missatges LoRa seran més robustos davant d’interferències i tindran més cobertura, a costa d’obtenir un ToA (Time on Air) més elevat i un augment del SNR. Com es pot veure a la Taula 3.1, es poden fer servir diferents valors de SF.

#### Coding Rate (CR)

La Taxa de Codificació, o Coding Rate (CR) indica la quantitat de bits d’un flux de dades que realment són d’utilitat. És un mecanisme que s’empra per a la correcció d’errors en la transmissió. Tal i com es mostra a la Taula 3.1, es poden fer servir diferents valors de CR.

Una taxa de codificació més alta, no augmentarà la cobertura, però farà que l’enllaç entre dos nodes sigui més confiable i robust si hi ha interferències.

Per exemple, si es fa servir una taxa de codificació de 4/8, per cada 4 bits d'informació útil, hi hauran uns altres 4 que seran redundants.

La Figura 2.2 representa una taxa de codificació. La taxa de codificació descriu la relació entre les dades reals (information bits), i les dades de correcció d’errors afegits (error correction bits). A l’hora d’escollir una taxa de codificació, és important tenir en compte si és necessari utilitzar de manera permanent una taxa de codificació alta, amb la pèrdua de velocitat de transmissió que comporta, o si és més eficient permetre una pèrdua ocasional de paquets a causa de les interferències.

Imagen que contiene Forma

Descripción generada automáticamente

Figura 2.2 - Exemple de Coding Rate

Donat que la taxa de codificació no modifica els paràmetres físics de la modulació, dos transceptors LoRa que estiguin configurats amb diferents taxes poden descodificar els senyals de l’altre node. Això pot ser útil si un receptor està ubicat en una àrea amb moltes interferències. Per tant, en aquest cas, ambdós transceptors poden estar configurats amb taxes diferents i arribar a obtenir taxes de transferència òptimes.

Donades aquestes característiques, s’ha decidit dissenyar i implementar un protocol, utilitzat com a base LoRa, per a suplir algunes de les carències que té LoRa, com podrien ser la falta de capçalera en els missatges, mètodes d’accés al medi o la comprovació de la rebuda de missatges.

## LoRaWAN

LoRaWAN (Long Range Wide Area Network) és un protocol de comunicació sense fils de baix consum que utilitza LoRa. S’ubica una capa superior de la capa física de Lora, la capa d’enllaç, i té l’objectiu d’implementar millores sobre LoRa. En la següent figura, podem veure la jerarquia de les capes de LoRaWAN:

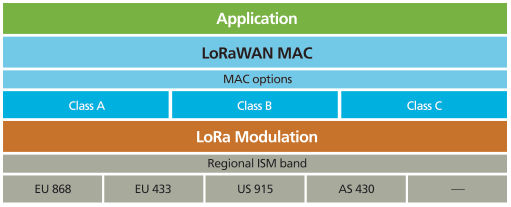


Figura 2.3 - Trama LoRaWAN

El disseny de LoRaWAN, implica millores respecte a LoRa, com per exemple augmentar la capacitat i la seguretat de la xarxa, afegir el suport de 3 classes de nodes diferents, l’administració de dispositius més eficients, i la implementació de xarxes públiques i privades.

### Classes de LoRaWAN

#### Classe A

La classe A és la classe més suportada, i proporciona l'estalvi més gran energia, ja que només entra en mode recepció després d’enviar dades cap al gateway. Els nodes que són classe A són dispositius que sempre estan alimentats per una bateria.

#### Classe B

Els dispositius de classe B són dispositius que tenen uns temps predeterminats per enviar i rebre missatges amb el gateway. Són dispositius que no necessàriament han d’estar alimentats mitjançant una bateria externa.

#### Classe C

Els dispositius de classe C, són dispositius que sempre estan escoltant el canal, i, per tant, ofereixen un estalvi d’energia menor a la resta de classes. Per aquest motiu, aquests nodes no acostumen a alimentar-se mitjançant una bateria.

### Modes d’activació

Dintre de LoRaWAN hi ha dos modes diferents per connectar-se a la xarxa: OTAA (Over-The-Air Activation) i ABP (Activation-By-Personalization)

#### Over-The-Air Activation (OTAA)

El mode OTAA és el mode més segur de connectar-se a una xarxa LoRaWAN. Per poder utilitzar aquest mode, es necessiten 3 paràmetres:

* DevEUI: És l’identificador de fàbrica i és únic per cada node.
* AppEUI: És un identificador únic de 64 bits, i fa servir per classificar els dispositius per aplicacions.
* AppKey: És una clau secreta xifrada amb AES de 128 bits compartida entre el node i el gateway. Es fa servir per a determinar les claus de la sessió.

Amb aquestes dades s’estableix la connexió del node a la xarxa de la següent manera:

1. El node sol·licita un inici de sessió a la xarxa amb les dades de configuració.
2. El gateway rep la sol·licitud, i l'envia al servidor.
3. El servidor verifica que el node estigui donat d’alta i la clau d’encriptació sigui correcta.
4. Si la clau és correcta, el servidor assigna una sessió temporal i l'envia mitjançant el gateway al node.
5. El node rep la sessió temporal i pot enviar dades a la xarxa.

Aquest mode aporta molta seguretat a la xarxa, ja que es crea una sessió temporal, i si es perd la connexió, o el node es reinicia, es torna a crear una sessió diferent, dificultant així robar la sessió de cada node.

#### Activation-By-Personalization (ABP)

El mode ABP és el mode més senzill de connexió que suporta LoRaWAN. Utilitza  tres paràmetres de configuració:

* DevAdress: És una direcció que farà servir per a comunicar-se posteriorment amb la xarxa.
* NetworkSessionKey: És una clau de xifratge entre el dispositiu i l’operador. Es fa servir per a les transmissions, i per validar la integritat dels missatges.
* ApplicationSessionKey: És una clau de xifratge entre el dispositiu i l’operador, feta servir per transmissions i per validar la integritat dels missatges.

Amb aquestes dades, s’estableix la connexió del node a la xarxa de la següent manera:

1. El node envia les dades al gateway.
2. El gateway valida que les dades corresponguin a la sessió.
3. Si la sessió és correcta, les dades es processen.

Aquest tipus de connexió permet una connexió més ràpida a la xarxa, ja que no requereix la confirmació del servidor però no aplica mecanismes de seguretat tan bons com l'OTAA.

## Duty Cycle vs Listen Before Talk

Un dels factors més determinants en el desenvolupament d’aquest treball, ha estat les limitacions del Duty Cycle definides per l’ETSI, les quals, en la banda de 868 MHz, limiten el Duty Cycle a un valor màxim del 1%. Aquesta limitació, implica que en un interval d’una hora, el ToA d’un missatge pot ser com a màxim de 36 segons.

En la normativa vigent, l’ETSI defineix el següent:

The Duty Cycle at the operating frequency shall not be greater than values in annex B or any NRI for the chosen operational frequency band(s). [1]

I a l’annex B, indica el següent:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operational Frequency** | **Maximun Effective radiated power, e.r.p** | **Channel Access and occupation rules (e.g. Duty cycle or LBT + AFA)** | **Maximum occupied bandwidth** |
| 868.0 MHz to 868.6 MHz | 25 mW e.r.p | ≤1% duty cycle or polite spectrum access | The whole band |

Per entendre correctament les implicacions d’aquesta regulació, s’explicarà a continuació a què es refereix el Duty Cycle i el ToA.

El Duty Cycle, és el temps que hi ha entre un missatge d’un tipus amb el següent missatge del mateix tipus. Es a dir, seria el temps que hi ha entre un missatge tipus A fins a la transmissió del següent missatge tipus A.

El ToA és el temps que triga un missatge a ser enviat des del transmissor fins al receptor. El ToA es pot veure afectat per la configuració del missatge LoRa. Alguns del paràmetres que poden afectar al ToA són:

* SF: Quant més elevat es el SF, més cobertura tindrà el missatge però el ToA també incrementa
* Amplada de Banda: Un ample de banda menor, augmenta la cobertura però incrementa el ToA
* Coding Rate: Un CR elevat incrementa la correcció de bits erronis però també incrementa el RoA.
* Mida Payload: Quants més bits d’informació s’enviïn, més temps de ToA hi haurà.

Per altra banda, en l’annex B de la normativa vigent, podem observar com, quan s’utilitza el *Polite Spectrum Access*, no es requereix complir el 1% del duty cylcle.

Respecte al *Polite Spectrum Access*, si s’utilitza el *Clear Channel Assessment Threshold,* l’ETSI indica el següent:

Clear channel Assessment clause applies to EUT with polite spectrum access instead of duty cycle where permitted by table B.1 in annex B, or table C.1 in annex C or any NRI [1]

Quan es fa servir el criteri del Polite Spectrum Access, en comptes d’emprar un 1% del Duty Cycle, es permet utilitzar el 2,7%, el qual implica augmentar de 36 a 100 segons de ToA disponibles per hora. [2]

# Mètodes i material utilitzat

## Software i hardware utilitzat

### Hardware utilitzat

L’eina principal que s’ha utilitzat en el desenvolupament d’aquest treball, ha estat el microcontrolador ESP32S3, en concret el model Heltec Wi-Fi LoRa v3, el qual ja porta integrat un transceptor LoRa.

En el transceptor incorporat s’ha fet servir elmodel SX1626 de Semtech ®. Aquest tipus d’antena, està dissenyada per dispositius de baix consum, com el ESP32, ja que tenen un consum de 4,2 mA en estat actiu i poden transmetre amb potencies de fins a +22 dBm.

Per altra banda, s’ha fet servir un ordinador portàtil per poder programar aquest microcontrolador i s’ha necessitat un telèfon mòbil amb suport a BLE (Bluetooth Low Energy) per la configuració sense fils dels microcontroladors.

Imagen que contiene circuito, electrónica

Descripción generada automáticamente

Il·lustració 6.1

### Software Utilitzat

#### Arduino

Pel que fa al software, s’ha decidit fer servir el llenguatge de programació d’Arduino juntament amb l’IDE Arduino IDE, pel fet que és un llenguatge de programació àmpliament usat, pel qual existeix molta documentació i llibreries de tota classe que poden facilitar el desenvolupament del projecte.

#### Python

També s’ha utilitzat el llenguatge de programació Python per generar arxius CSV amb mesures realitzades durant el projecte. Gràcies a la gran versatilitat que ofereix, ha permès la realització de scripts per efectuar mesures en temps real del codi que s’executa en l’ESP, i a partir dels resultats obtinguts, extrets conclusions posteriorment.

#### Llibreries utilitzades

A continuació es citen les llibreries utilitzades per el desenvolupament del projecte:

* Arduino:
  + RadioLib: Llibreria pel suport de LoRa
  + ArduinoBLE: Llibreria pel suport de BLE
  + Preferences: Llibreria per guardar dades en memòria no volàtil
  + WiFi: Llibreria pel suport WiFi
* Python:
  + Pandas: Llibreria per la creació dels datasets
  + Matplotlib: Llibreria per la creació de diagramas
  + Os: Llibreria per accedir al sistema de fitxers de l’ordinador
  + Serial: Llibreria per llegir el port sèrie
  + Csv: Llibreria per crear arxius .CSV
  + Time: Llibreria per mesurar el temps
  + Thread: Llibreria per executar dos tasques simultàniament

#### Control de versions

Pel fet que aquest projecte és de ciència ciutadana, el projecte sencer és de codi obert i s’ha utilitzat Git i GitHub tant com per eines de control de versions com a repositori obert per trobar tot el material usat pel desenvolupament del projecte, així com exemples per provar funcionalitats d’algunes llibreries, proves empíriques del funcionament del ESP32, i versions inicials del protocol que s’ha dissenyat.

#### nRF Connect

S’ha fet servir la aplicació mòbil *nRF Connect* per establir comunicació BLE amb el microcontrolador.

## SetUp inicial

Tant per utilitzar com per modificar alguna part del projecte es requereix una petita instal·lació de dependències, així com disposar del hardware esmentat anteriorment.

### Instal·lació de l’entorn

Inicialment, es necessita instal·lar tant Arduino-CLI com un IDE que permeti seleccionar ports, llegir el port serial i pujar el codi a la placa. Tot això es possible fer-ho de forma separada, però Arduino dona suport per el seu [IDE](https://www.arduino.cc/en/software).

### Instal·lació placa

Un cop s’ha descarregat l’IDE, és necessari descarregar plaques ESP32. Per realitzar això, s’ha d’obrir el menú a dalt a la dreta i entrar en les preferències.

Un cop ubicats en les preferències, s’ha d’enganxar la següent línia en l’apartat de *“Additional boards manager URLs”:* [*https://dl.espressif.com/dl/package\_esp32\_index.json*](https://dl.espressif.com/dl/package_esp32_index.json)

En la següent figura, es mostra com s’ha de fer.

|  |  |
| --- | --- |
| Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico  Descripción generada automáticamente | Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente |

Figura 3.1 - Instal·lació suport plaques ESP32

Un cop s’han instal·lat el suport per tots els models dels ESP32, és important comprovar que estiguin instal·lades, per això, en el menú de l’esquerra, en la segona icona, s’ha de buscar “*esp32 by espressif”* i comprovar que estigui instal·lat.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura 3.2 - Comprovació instal·lació ESP32

### Instal·lació de llibreries necessàries

Per poder instal·lar les llibreries necessàries dintre del IDE, s’ha d’estar ubicat en la tercera icona del menú de l’esquerra. Un cop dintre, podem instal·lar les llibreries abans esmentades.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Figura 3.3 - Instal·lació llibreries

## Estudi llibreries LoRa

Donat que Arduino és un llenguatge mundialment utilitzat hi ha una gran varietat de llibreries en relació a LoRa. Per aquest motiu, s’ha fet un estudi de les llibreries més utilitzades, comparant quins paràmetres permeten modificar i quins transceptors suporten. D’aquesta manera, s’ha pogut escollir la llibreria que millor s’adapta a les nostres circumstàncies. A continuació és presenta l’estudi de diverses llibreries LoRa, i en l’apartat de ‘Resultats’ es justificarà l’elecció final.

### Unofficial Heltec

La llibreria Unofficial Heltec és una llibreria que implementa tot tipus de funcions relacionades amb les plaques Heltec. No només permet utilitzar paràmetres LoRa, sinó que també incorpora tota mena de funcions relacionades amb aquest tipus de plaques. Això és degut al fet que Heltec proporciona molt poca documentació sobre les seves plaques, i pot arribar a ser difícil utilitzar-les.

En la documentació de la llibreria, podem trobar que per utilitzar les funcions LoRa, utilitza la llibreria RadioLib.

|  |  |
| --- | --- |
| **Mòdul** | **Descripció** |
| SX1262 | Semtech SX1262 sub-GHz transceiver |

Taula 3.1 - Llistat de mòduls Heltec

### Arduino LoRa

La llibreria Arduino Lora és una llibreria mantinguda per la comunitat, la qual, suporta transceptors SX 127x.

|  |  |
| --- | --- |
| **Mòdul** | **Descripció** |
| SX127x | Sèrie de mòduls LoRa SX1272, SX1273, SX1276, SX1277, SX1278, SX1279 |
| Dragino | LoRa Shield |
| HopeRF | RFM95W, RFM96W, RFM97W |
| Modtronix | inAir4, inAir9, inAir9B |

Taula 3.2 - Llistat de mòduls Arduino LoRa

### RadioLib

La llibreria RadioLib. Aquesta llibreria permet als usuaris integrar tot tipus de diferents mòduls de comunicació sense fils i protocols en un sol sistema. De forma nativa, RadioLib suporta Arduino, però també està dissenyada per executar-se en entorns no-Arduino.

|  |  |
| --- | --- |
| **Mòdul** | **Descripció** |
| CC1101 | Mòdul ràdio FSK |
| LLCC68 | Mòdul LoRa |
| LR11X0 | Sèrie LoRa/GFSF LR1110, LR1120, LR1121 |
| nRF24L01 | Mòdul 2,4GHz |
| RF69 | Mòdul FSK/OOK |
| RFM2X | Sèrie mòduls FSK RFM22, RFM23 |
| RFM9X | Sèrie mòduls LoRa RFM95, RFM96, RFM97, RFM98 |
| Si443X | Sèrie mòduls FSK Si4430, Si4431, Si4432 |
| STM32WL | Microcontrolador integrat / Mòdul LoRa |
| SX126X | Sèrie de mòduls LoRa (SX1261, SX1262, SX1268) |
| SX127X | Sèrie de mòduls LoRa (SX1272, SX1273, SX1276, SX1277, SX1278, SX1279) |
| SX128X | Sèrie de mòduls LoRa/GFSK/BLE/FLRC (SX1280, SX1281, SX1282) |
| SX123X | Mòduls ràdio FSK/OOK SX1232, SX1233 |

Taula 3.3 - Llistat de mòduls RadioLib

### Comparativa entre llibreries LoRa

Per poder fer una correcta elecció de quina seria la millor llibreria per el nostre ús, s’ha fet un anàlisis profund de les tres llibreries abans esmentades comparant totes les funcionalitats i característiques que ens podrien fer falta pel nostre projecte.

#### Comparativa de característiques

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Unnoficial Heltec** | **Arduino LoRa** | **RadioLib** |
| **Freqüència (MHz)** | X | X | X |
| **SF** | X | X | X |
| **Ample de Banda (KHz)** | X | X | X |
| **Coding rate** |  | X | X |
| **Preamble Length (bits)** |  | X | X |
|  | X | X | X |
| **Tipus de Bus** | SPI | SPI | SPI |
| **CAD** | X | X | X |
| **Capçalera implícita / explicita** |  | X | X |
| **Escriure dades en el paquet** | X | X | X |
| **RSSI** |  | X | X |
| **SNR** |  | X | X |
| **Modes ràdio** |  | Idle / sleep |  |
| **Sync Word** |  | X | X |
| **CRC** |  | X | X |

Taula 3.4 - Taula comparativa llibreries LoRa

#### Altres paràmetres a tenir en compte

Per la realització d’aquesta taula, s’ha tingut en compte el següent :

* La llibreria es considera actualitzada si ha s’ha pujat un commit en el repositori original en els últims 12 mesos.
* La documentació es considera bona, si es proporcionen exemples de codi de com der servir l’API de la llibreria que siguin d’utilitat.

|  |  |  |
| --- | --- | --- |
|  | **Unnoficial Heltec** | **Arduino LoRa** |
| **Llibreria actualitzada** | X | X |
| **Bona documentació** | X | X |

Taula 3.5 - Paràmetres extra a tenir en compte en llibreries LoRa

## Estudi llibreries LoRaWAN

Per l’estudi de llibreries de LoRaWAN, s’han trobat una gran varietat de llibreries. Un problema molt comú ha estat que moltes d’aquestes llibreries s’han provat amb hardware molt específic. Per aquest motiu, s’ha decidit fer la comparativa amb dues de les llibreries que suporten més transceptors.

### Beelan-LoRaWAN

Aquesta llibreria s’ha desenvolupat per ser utilitzada en plataformes genèriques, per tant, ens pot arribar a ser útil. En la següent taula podem visualitzar els transceptors que suporta.

|  |  |
| --- | --- |
| **Mòdul** | **Descripció** |
| SX127x | Sèrie de mòduls LoRa SX1272, SX1273, SX1276, SX1277, SX1278, SX1279 |

Taula 3.6 - Llistat de mòduls Beelan

### Arduino LoRaWAN

La llibreria MCCI Arduino LoRaWAN proporciona una forma estructurada d’utilitzar la llibreria Arduino-lmic per enviar dades del sensor mitjançant The Things Network o una xarxa similar basada en LoRaWAN.

|  |  |
| --- | --- |
| **Mòdul** | **Descripció** |
| SX127x | Sèrie de mòduls LoRa SX1272, SX1273, SX1276, SX1277, SX1278, SX1279 |

Taula 3.7 - Llistat de mòduls Arduino LoRaWAN

### Comparativa entre llibreries LoRaWAN

Comparativa de característiques

|  |  |  |
| --- | --- | --- |
|  | **Beelan** | **Arduino LoRaWAN** |
| **Authentication Keys** | X | X |
| **Activació OTAA** | X | X |
| **Mode Sleep** | X |  |
| **Classes LoRaWAN** | A i C | A |

Taula 3.8 - Taula comparativa llibreries LoRaWAN

#### Altres paràmetres a tenir en compte

Per a la realització d’aquesta taula s’ha tingut en compte el següent :

* La llibreria es considera actualitzada si ha s’ha pujat un commit en el repositori original en els últims 12 mesos.
* La documentació es considera bona, si es proporcionen exemples de codi de com fer servir l’API de la llibreria que siguin d’utilitat.

|  |  |  |
| --- | --- | --- |
|  | **Beelan** | **Arduino LoRaWAN** |
| **Llibreria actualitzada** | X | X |
| **Bona documentació** | X | X |

Taula 3.9 - Paràmetres extra a tenir en compte en llibreries LoRaWAN

## Estalvi d’energia i modes Sleep

Un dels punts mes importants en relació a projectes d’IoT, és el consum d’energia. És per aquest motiu que els ESP32 permeten la configuració de diferents modes d’energia, els quals ofereixen vàries possibilitats de configuració.

Com un dels objectius finals del projecte és poder utilitzar els nodes mitjançant una bateria com a principal font d’energia, és crucial entendre el funcionament de tots els modes que ofereix l’ESP per optimitzar el consum d’aquests nodes.

### Modes d’energia

#### Mode Actiu

Aquest mode és el mode que s’utilitza per defecte en els ESP. Tot i que aquest mode és el que permet la major capacitat de processador, també és el que té el consum d’energia més elevat.

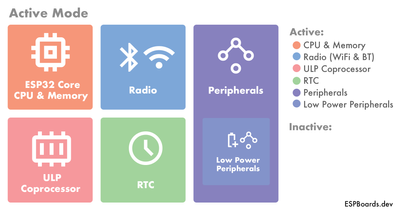


Figura 3.4 - Mode Actiu

Com podem observar en la figura, aquest mode utilitza totes les parts disponibles del SoC del ESP32, a diferència de la resta de modes.

#### Mode Light Sleep

Aquest mode és molt utilitzat quan no es requereix de comunicacions mitjançant ràdio quan el dispositiu es posa a dormir. [3]

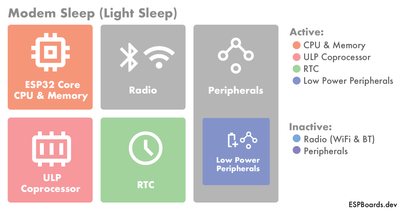


Figura 3.5 - Mode Light Sleep

Aquest mode deshabilita els mòduls radio que porta l’ESP així com els perifèrics que no són de baix consum. Amb aquest mode, es poden configurar mètodes per despertar el microcontrolador per fer algunes tasques que necessitin de més potència, o utilitzar la ràdio. Quan el ESP està en aquest mode, té un consum de 240 µA.

#### Mode Deep Sleep

En comparació als modes explicats anteriorment, el Mode Deep Sleep deshabilita el nucli principal del ESP, el qual inclou el processador principal i la memòria, obtenint així un consum mínim. [3]

Com el nucli del ESP està deshabilitat, el co-processador ULP que té el ESP es manté actiu a l’espera d’un esdeveniment extern o a un comptador per a despertar-se. En aquest mode d’energia, el ESP té un consum de 8 µA.

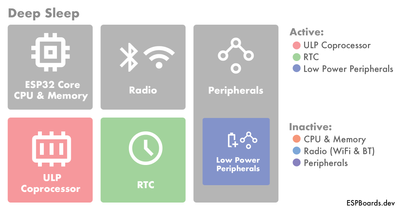


Figura 3.6 - Mode Deep Sleep

Un factor molt important a entendre per utilitzar correctament aquest mode d’estalvi d’energia, està relacionat amb la pèrdua de memòria del estat del ESP abans de posar-se a dormir. Quan el ESP es desperta del mode *deep\_sleep,* torna al inici del codi, en comptes de seguir executant el codi per on l’havia deixat. Això ve donat pel fet que en aquest mode, el ESP deshabilita la memòria i no és capaç de recordar en quin punt del codi s’havia anat a dormir, per aquest motiu el ESP realitza aquest reinici quan desperta del mode *deep\_sleep*. Per aconseguir que el ESP recordi per on continuar un cop s’ha despertat, es pot fer servir la memòria RTC.

#### Mode hibernació

Aquest mode és el mode amb menys consum. S’aconsegueix mitjançant la deshabilitació de quasi tots els circuits interns, excepte el RTC. En aquest mode, el ESP únicament consumeix 5 µA. [3]



Figura 3.7 - Mode Hibernació

## Guardar dades en memòria no-volàtil i memòria RTC

Guardar dades en memòria no volàtil és de força importància si es vol que els dispositius no hagin de configurar-se un altre cop desprès d’una pèrdua de corrent o un reinici.

### Memòria no-volàtil

Els ESP32 permeten guardar dades en memòria flash utilitzant diverses llibreries:

* EEPROM
* Preferences

EEPROM ha estat la llibreria més utilitzada en entorns Arduino per guardar dades en memòria no volàtil durant molt de temps, però a la documentació oficial d’espresiff, es recomana utilitzar la llibreria de Preferences per davant de EEPROM. Per tant, ens centrarem en l’explicació i utilització de Preferences [4].

La llibreria Preferences és exclusiva per plataformes Arduino-ESP32, i es recomana utilitzar-la com a reemplaçament d’EEPROM. El seu funcionament es basa en la utilització d’una part de la memòria no volàtil integrada del ESP (NVP) per poder emmagatzemar dades. Les dades guardades en la memòria NVP, es conserven davant possibles reiniciïs del dispositiu i/o d’esdeveniments de pèrdua d’energia del sistema.

La llibreria Preferences funciona millor per emmagatzemar molts valors petits, en comptes de poca quantitat de valors amb un pes elevat. En cas de necessitar emmagatzemar dades de gran capacitat, es recomana la utilització d’altres biblioteques o llibreries relacionades amb el sistema d’arxius, com pot ser LittleFS.

### Memòria RTC

La memòria RTC (Real-Time Clock) es un espai de memòria independent associada al circuit del rellotge en temps real de l’ESP32. La memòria RTC té dos tipus principals:

* Memòria lenta RTC
* Memòria ràpida RTC

Per comparació a la memòria NVP, la memòria RTC s’utilitza per emmagatzemar dades de menys capacitat i que requereixen una gran velocitat a l’hora de ser consultades. Com a exemple pràctic, RTC pot servir per emmagatzemar variables especifiques, mentre NVP pot servir per guardar credencials, utilitzant la del disseny de “clau-valor”.

En relació amb els modes d’estalvi d’energia, com ja s’ha esmentat anteriorment, la memòria RTC no s’elimina desprès d’utilitzar qualsevol dels 3 nodes. Més endavant, s’ensenyarà com s’ha utilitzat la memòria RTC per guardar en quin estat estava el ESP, i continuar en el punt on estava abans de posar-se a dormir.

#### Similituds i diferències entre ambdós modes

En ambdós modes es permeten guardar les dades mentre l’ESP està en mode de son profund o si l’ESP perd la font d’energia.

Per altre banda, mentre el mode lent permet als dos nuclis del microcontrolador accedir a les dades, el mode ràpid només permet accedir al nucli PRO CPU (core 0).

Per últim, com el seu nom ja ho indica, el mode ràpid es capaç d’accedir a la memòria de forma més ràpida, i es utilitzat per emmagatzemar memòria que ha de ser accedida pel codi immediatament després d’un reinici.

## Configuració del Wi-Fi mitjançant BLE

Per configurar el Wi-Fi en dispositius Arduino-ESP32, es pot utilitzar la llibreria que ja porten integrats aquests dispositius, anomenada “WiFi.h”. Aquesta llibreria permet definir quin SSID té la xarxa a la qual es vol connectar el microcontrolador, i definir quina contrasenya ha d’utilitzar.

Degut a que l’escenari en el qual es planteja utilitzar aquests ESP32, les xarxes Wi-Fi a les quals es poden connectar poden variar, s’ha optat per una solució dinàmica, que permeti a l’usuari canviar la configuració del Wi-Fi en funció de les seves necessitats.

És important recalcar la importància de que les xarxes Wi-Fi a les quals l’ESP32 es pot connectar, són únicament les xarxes de 2,4GHz, degut a que el sensor integrat dels ESP32, no suporta les xarxes de 5GHz.

### Utilització de BLE

Per aconseguir el dinamisme esmentat anteriorment per a la configuració del Wi-Fi, s’ha decidit fer la configuració del SSID i la contrasenya de les xarxes Wi-Fi mitjanant BLE.

L’entorn Arduino-ESP32, permet la utilització de la llibreria “ArduinoBLE.h”, la qual permet utilitzar el protocol BLE. D’aquesta manera, es pot definir el ESP32 com a central BLE, i mitjançant un mòbil amb suport a la versió Bluetooth 5.0 (els dispositius amb aquesta versió ja suporten la utilització de BLE), es pot establir una connexió entre ambdós dispositius i configurar els paràmetres Wi-Fi.

Per a que les credencials Wi-Fi romanguin en la memòria del dispositiu, i siguin persistents a possibles pèrdues d’energia del ESP, s’ha plantejat la possibilitat de guardar aquestes dades en memòria NVP. En l’apartat de resultats, es pot visualitzar els l’execució d’aquesta configuració.

### Seguretat en la transmissió BLE

Degut a que la intenció es transmetre les credencials Wi-Fi mitjançant BLE, és important conèixer si BLE ofereix alguna capa de seguretat, o en contraposició, envia les dades en text pla, i per tant, qualsevol persona que estigui en aquell moment escoltant el transit del canal podria llegir les credencials.

BLE implementa l’encriptació amb el xifrat per bloc que proporciona AES-128, assegurant així la confidencialitat de les dades. Aquesta encriptació es realitzada un al payload de BLE un cop s’ha establert una connexió segura entre els dos terminals.

Per altra banda, BLE també implementa MIC (Message Integrity Check). D’aquesta manera el node receptor és capaç de confirmar o descartar l’autenticitat del transmissor, i evitar així una possible suplantació d’identitat per part d’una tercera part.

Un altra mesura implementada a BLE, es la implementació de les claus IRK (Identity Resolving Key), les quals serveixen per generar i resoldre adreces BLE aleatòries. Aquestes adreces només poden ser resoltes per dispositius de confiança, mantenint d’aquesta manera la privacitat de cada dispositiu.

## Accés al medi

Per a que els nodes puguin comunicar-se de forma eficient i segura amb el gateway, i donat a que LoRa no suporta de manera nativa cap control d’accés al medi, s’ha plantejat la possibilitat d’implementar 2 opcions d’accés al medi diferents.

### CSMA

Degut a la dificultat de sincronitzar els nodes des d’un inici, ja que no hi ha un rellotge central que sincronitzi a tots els nodes, s’ha decidit implementar com a primera opció el mecanisme d’accés al medi basat en CSMA.

Aquest mecanisme, consisteix en escoltar el canal i enviar el missatge si no hi ha ningú transmetent. En cas de que hi hagi algun altre node utilitzant el canal, torna a escoltar el canal durant un altre segon, i el torna a enviar.

Es requereix d’aquest tipus d’accés al medi en l’inici de les comunicacions del protocol, degut a que no hi ha un rellotge central, i per tant, els nodes no poden enviar missatges en ordre. D’aquesta manera, es podem evitar eventuals col·lisions entre missatges i respectar la normativa vigent respecte al *Polite Spectrum Access*. Si en comptes de respectar el *Polite Spectrum Access* es decidís respectar el temps del Duty Cycle, amb un control d’accés al medi ALOHA hagués estat suficient, ja que no faria falta escoltar el canal abans d’enviar.

### TDMA

Un cop els nodes i el gateway ja han establert una comunicació inicial, i es sap quants nodes hi ha, es pot definir un sistema de slots temporals, en el qual cada node ja sap en quin moment ha d’enviar, evitant així possibles col·lisions a l’hora d’enviar un missatge.

Per simplificar la complexitat dels nodes, s’ha decidit que el gateway és l’encarregat de definir quin slot de temps correspon a cada node. Amb aquesta implementació, es simplifica la complexitat de cada node, ja que tenen una funció de “*slave*”, mentre que és el gateway que fa de *“master”.*

Amb un accés al medi de tipus TDMA, no seria necessari esperar a escoltar el canal, ja que els no s’espera que cap altre node estigui transmeten en el mateix instant. En cavi, si es col respectar el *Polite Spectrum Access* definit per l’ETSI, s’haurà d’escoltar el canal abans de d’enviar.

En termes de consum d’energia, com el node ja sap quan ha d’enviar les dades, pot posar-se a dormir, i despertar-se només per enviar les dades.

## Càlcul del SF òptim

Per optimitzar la comunicació LoRa entre el gateway i els nodes, s’ha decidit calcular el SF òptim per a cada node en funció del RSSI i del SNR. D’aquesta manera, el ToA dels missatges i el DC (Duty Cycle) a complir serà menor per als missatges que utilitzin aquest SF.

Per trobar el SF òptim en funció del SNR i del RSSI, s’ha utilitzat com a referència els valors calculats en un estudi del Institut Tecnològic d’Enginyeria de Telecomunicacions de Purwokerto, Indonèsia [5]. En les taules següents, podem trobar les relacions utilitzades.

|  |  |
| --- | --- |
| **SF** | **SNR Limit (dB)** |
| 7 | -7,5 |
| 8 | -10 |
| 9 | -12,5 |
| 10 | -15 |
| 11 | -17,5 |
| 12 | -20 |

Taula 3.10 - Relació SF - SNR

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SF** | 7 | 8 | 9 | 10 | 11 | 12 |
| **Sensitivitat (dBm)** | -125 | -127 | -130 | -132 | 135 | -137 |

Taula 3.11 - Relació SF - RSSI

## Càlcul Time on Air i Duty Cycle

El càlcul del ToA i del DC és de força importància, ja que implica el compliment de les normatives vigents, i la futura definició del protocol.

Per la realització del càlcul del Time on Air i del Duty Cycle, s’ha fer servir la calculadora que proporciona Semtech [6], on es poden definir tots els valors utilitzats en la transmissió, així com el transceptor que s’ha fet servir.

# Resultats

## Implementació llibreria LoRa - RadioLib

Per implementar la tecnologia LoRa en aquest projecte, s’ha decidit utilitzar la llibreria RadioLib, ja que és la llibreria que més opcions permet configurar. Per altra banda, tal i com es mostra a la taula 3.4, també dona suport a molts altres transceptors més enllà del que utilitzarem nosaltres, per tant, permetria una possible millora del hardware sense haver de buscar altres llibreries compatibles.

La implementació d’aquesta llibreria ha estat molt satisfactòria, ja que ha permès treballar amb la tecnologia LoRa sense problemes. Per altra banda, la documentació oficial de la llibreria és molt extensa, on explica detalladament totes les funcions que permet, a més d’un gran nombre d’exemples per entendre millor el seu funcionament.

S’ha decidit fer servir RadioLib per davant de Unofficial Heltec, ja que tot i que Unofficial Heltec també utilitza RadioLib per les transmissions LoRa, afegeix moltes opcions que realment no es requereixen per el projecte. Per altra banda, Unnoficial Heltec no ofereix cap aventatge respecte a RadioLib ja que totes les funcions que ofereix Unnoficial Heltec també les ofereix RadioLib.

### Paràmetres utilitzats

En el protocol basat en LoRa, s’han configurat una sèrie de paràmetres per establir connexió. Aquests paràmetres han estat:

* Freqüència
* SF
* Amplada de banda
* Coding Rate

Com ja s’ha explicat en el marc teòric, aquests paràmetres són els que LoRa permet configurar per establir connectivitat entre dos nodes. Per poder configurar-los mitjançant RadioLib, s’ha de fer de la següent manera:

Texto, Carta

Descripción generada automáticamente

Il·lustració 4.1 - Configuració paràmetres LoRa

### Estudi funcionament CAD

RadioLib, permet utilitzar el CAD (Channel Activity Detection) amb els models de transceptors SX1262. Per poder fer servir la funcionalitat del CAD, RadioLib dona suport a dues funcions: startChannelScan() i getChannelScanResult().

En la següent il·lustració, podem veure l’exemple simplificat que proporciona RadioLib en la seva documentació, per detectar si hi ha algun altre node transmetent.

Texto, Carta

Descripción generada automáticamente

Il·lustració 4.2 - Utilització CAD

És important saber si aquest CAD funciona correctament, per aquest motiu, s’han fet una sèrie de proves per mesurar si té un funcionament correcte. El codi utilitzat per les següents proves es pot trobar en el següent enllaç: https://github.com/paugarcia32/TFG/tree/main/03%20-%20Results%20of%20the%20measurements/CAD

#### Temps de detecció

La primera prova realitzada, ha estat el temps de detecció dels missatges en el canal. Per la realització d’aquesta prova, s’han configurat 2 nodes connectats mitjançant USB a un PC. La distància entre els nodes ha estat d’uns 50 cm.

S’ha configurat el node transmissor, per enviar el missatge “*Hola Mundo #X*” pel canal, sent la “*X*” el número de missatge. A l’hora, cada cop que transmet el missatge, envia un missatge per el port serial.

El node receptor, s’ha configurat per escoltar el canal, i en el moment en que detecta el missatge, deixa d’escoltar el canal i envia un altre missatge per el port serial. Un cop ha enviat el missatge per el por sèrie, torna a escoltar el canal.

Mentre ambdós nodes estan comunicant-se mitjançant missatges Lora, i transmetent per els ports sèrie, hi ha un script escrit en Python, escoltant els dos ports sèrie, que s’ha encarregat de guardar quin node ha enviat un missatge per el port sèrie, i a quina hora. Aquestes dades són guardades en un arxiu CSV, per ser analitzades posteriorment.

Aquesta prova s’ha fet amb els 6 SF possibles per poder analitzar el temps que pot trigar en detectar els missatges el transceptor utilitzat mitjançant les funcions del CAD avilitades per la llibreria.

En la següent figura, es pot visualitzar els resultats obtinguts per a cada SF.

|  |  |
| --- | --- |
| Figura 4.1.a – Mesura temps detecció CAD amb SF 12 | Figura 4.1.b – Mesura temps detecció CAD amb SF 11 |
| Figura 4.1.c – Mesura temps detecció CAD amb SF 10 | Figura 4.1.d – Mesura temps detecció CAD amb SF 9 |
| Figura 4.1.e – Mesura temps detecció CAD amb SF 8 | Figura 4.1.f – Mesura temps detecció CAD amb SF 7 |

Figura 4.1 - Mesures tems detecció CAD

Els resultats obtinguts en les mesures realitzades, tenen molt sentit, ja que si tenim en compte el ToA dels missatges, sabent que el payload es de 13 bytes (1 byte per cada caràcter del missatge), són resultats molt coherents.

|  |  |  |  |
| --- | --- | --- | --- |
| **SF** | **ToA teòric (s)** | **Valor mitjà mesurat (s)** | **Temps detecció CAD (ms)** |
| 12 | 1,16 | 1,244 | 84 |
| 11 | 0,5775 | 0,6905 | 113 |
| 10 | 0,2887 | 0,3232 | 34,5 |
| 9 | 0,1648 | 0,2254 | 60,6 |
| 8 | 0,0926 | 0,1414 | 21,5 |
| 7 | 0,0514 | 0,1046 | 53,2 |

#### Temps en reiniciar el receptor

La realització d’aquesta mesura ha estat pel fet de que en la prova anterior, s’ha detectat que el CAD no ha estat capaç de detectar tots els missatges enviats. Una primera hipòtesi ha estat que el receptor triga massa temps en reiniciar-se.

Per a la realització d’aquesta mesura, s’ha configurat el node receptor per enviar un missatge per el port sèrie just abans de reiniciar el transceptor LoRa, i enviar un altre missatge just quan ja l’ha reiniciat.

Per altra banda, s’ha configurat un script que escolta els missatges del port sèrie i guarda les dades en un CSV per poder analitzar-les.

#### Taxa d’error del CAD

La última mesura realitzada, degut al fet de que el CAD només triga 9 ms en reiniciar-se, ha estat veure quina taxa d’error té el CAD. Per fer aquesta mesura, s’han configurat dos nodes connectats mitjançant USB al mateix PC. En aquest cas, els nodes utilitzen SF 7.

Per a cada mesura, el node transmissor envia missatges en intervals de temps diferents: 1, 2, 3, 4, 5 i 10 segons de diferència entre els missatges. El node ha estat configurat per enviar 100 missatges en cada prova.

Els resultats obtinguts indiquen que la taxa d’error varia en funció del temps entre missatges de la següent manera: si el temps entre missatges -és menor, la taxa d’error augmenta, mentre que si el temps entre missatges és major, la taxa disminueix. Podem visualitzar els resultats en la figura següent:

Figura 4.2 - Taxa d'error CAD

### Obtenció SNR i RSSI d’un missatge

RadioLib, també permet mitjançant el transceptor SX1262, mesurar quin SNR i RSSI hi rep un node d’un missatge. Aquestes funcions són de gran utilitat per el càlcul del SF òptim.

Es pot obtenir el SNR i el RSSI de la següent manera:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Il·lustració 4.3 - Obtenció SNR i RSSI d’un missatge

## Implementació LoRaWAN – Arduino LoRaWAN

La implementació de LoRaWAN en aquest projecte, s’ha efectuat mitjançant la llibreria Arduino LoRaWAN. En contraposició a l’apartat relacionat amb la implementació de LoRa, en aquest cas no s’ha aconseguit fer servir LoRaWAN en el projecte. Tot i utilitzant el codi d’exemple que proporcionava la llibreria, no s’ha aconseguit establir connexió amb el gateway.

En la següent figura, adjunto el missatge d’error que més cops ha sortit en el desenvolupament d’aquesta part del projecte.

Tabla

Descripción generada automáticamente

Figura 4.3 - Missatge d'error LoRaWAN

Una hipòtesi per la qual es creu que no ha funcionat correctament, és per una configuració errònia dels pins, totot i haver utilitzat la definició de pins que ve assignada a la placa en el arxiu *pins\_arduino.h* ubicada en el directori *C://Users/[usuari]/.platformio/packages/framework-arduinoespressif32/variants/heltec\_wifi\_lora\_v3.*

Per altra banda, també s’ha configurat correctament l’arxiu *lmic\_project\_config.h* en el qual es configura els paràmetres de freqüència i mòdul ràdio que s’utilitza.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Il·lustració 4.4 - lmic\_project\_config.h

Un altra de les configuracions que s’han fet, ha estat la d’un gateway a TTN, per poder establir connectivitat entre l’ESP i el gateway. Com la compilació del codi a la placa no ha funcionat, no s’ha aconseguit enviar dades al gateway LoRaWAN de TTN.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 4.4 - Configuració TTN

## Configuració Wi-Fi mitjançant BLE

La configuració del Wi-Fi a l’ESP mitjançant BLE ha funcionat correctament. Com ja s’ha esmentat, s’ha fet servir la llibreria ArcuinoBLE per establir la connectivitat BLE mitjançant un telèfon mòbil i l’ESP.

Per l’aplicació mòbil, s’ha fet servir l’aplicació *nRF Connect*, la qual permet fer un escaneig del canal, i fer comunicacions BLE amb altres dispositius. El codi que s’ha fet servir, s’ensenya en l’apartat 4.4, justament amb el codi de com guardar dades en les *Preferences.*

## Guardar dades en memòria no volàtil

Emmagatzemar dades en memòria no volàtil també ha estat possible. S’ha implementat una funció per poder guardar les credencials Wi-Fi mitjançant la llibreria *Preferences* a la memòria no volàtil del ESP32. Per altra banda, també s’ha pogut guardar l’estat del dispositiu en memòria RTC, per poder continuar en el mateix punt del codi abans de que el microcontroladors es posés en mode *deep\_sleep*. El codi utilitzat per les següents proves es pot trobar en el següent enllaç: https://github.com/paugarcia32/TFG/tree/main/03%20-%20Results%20of%20the%20measurements/NVPStorageWithWiFiCredentials

### Proves realitzades amb la llibreria *Preferences*

Per comprovar el correcte funcionament de la llibreria Preferences, s’ha fet servir la mateixa prova que s’ha realitzar per BLE. Inicialment, s’ha configurat el ESP per comprovar si hi ha dades guardades a les Preferences. En cas de no haver-hi, s’inicialitza el BLE i s’espera a rebre dades.

Quan amb el telèfon mòbil, enviem les dades amb el format adequat, l’ESP guarda les dades en les Preferences, i intenta connectar-se al Wi-Fi.

En la següent figura, podem veure l’output del port sèrie, on es mostra el comportament del ESP.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Figura 4.5 - Configuració Wi-Fi mitjançant BLE

En la figura anterior, es pot observar com al inici, el ESP32 busca si té credencials Wi-Fi emmagatzemades : *“No sotred Wi-Fi credentials”.* Si no té credencials guardades, inicia el servei BLE, per posteriorment guardar les credencials. Quan el ESP rep les credencials amb el format *SSID:PASSWORD* intenta connectar-se a la xarxa Wi-Fi.

En la següent figura, es pot observar com desprès d’un reinici, si el ESP troba credencials Wi-Fi guardades en la memòria NVP*,* intenta connectar-se:

Texto

Descripción generada automáticamente

Figura 4.6 - Reutilització dades NVP

## Estalvi d’energia i modes *sleep*

Per comprovar el correcte funcionament dels ESP quan canvien de modes d’energia, s’han fet proves per comprovar quant de temps estan realment en mode *sleep*, i quant de temps triguen en despertar. El codi utilitzat per les següents proves es pot trobar en el següent enllaç: https://github.com/paugarcia32/TFG/tree/main/03%20-%20Results%20of%20the%20measurements/Sleep%20Modes%20ESP

### Mode *light\_sleep*

Per comprovar el correcte funcionament d’aquest mode, s’han fet vàries mesures, que consisteixen en mesurar quant de temps està dormint el dispositiu realment, en comparació al temps que se li ha indicat.

Aquesta prova ha consistit en definir un temporitzador de temps per cada mesura; 1, 5 i 10 segons. Cada cop que el ESP es va a dormir, envia un missatge per el port serial, i cada cop que es desperta, envia un altre missatge. Al mateix temps, hi ha un script escrit en Python que guarda les dades en un CSV, per poder analitzar-les més endavant. D’aquesta manera podem mesurar la diferència de temps entre cada missatge.

Els resultats obtinguts en aquestes mesures han estat molt sorprenents, ja que en aquest mode, el ESP sempre es desperta 1 segon abans del que s’ha indicat. Aquests valors proporcionen força inexactitud per a una possible futura implementació. Es poden observar els resultats de la mesura en la següent figura.

|  |  |
| --- | --- |
| Figura 4.3.a – Mesura 10 segons en *light\_sleep* | Figura 4.3.b – Diferència temps *deep\_sleep* 10 segons i valor real |
| Figura 4.3.c – Mesura 5 segons en *light\_sleep* | Figura 4.3.d – Diferència temps *deep\_sleep* 5 segons i valor real |
| Figura 4.3.e – Mesura 1 segon en *light\_sleep* | Figura 4.3.f – Diferència temps *light\_sleep* 1 segon i valor real |

Figura 4.7 - Mesures *light-sleep*

### Mode *deep\_sleep*

Les mesures realitzades en aquest mode, han estat idèntiques al mode anterior. A continuació es mostren els resultats.

En aquest cas els valors obtinguts tenen molt sentit, i demostren un correcte funcionament d’aquest mode. Es poden visualitzar els resultats en les següents figures.

|  |  |
| --- | --- |
| Figura 4.1.a – Mesura de 10 segons en despertar *deep\_sleep* | Figura 4.1.b – Diferència temps *deep\_sleep* 10 segons i valor real |
| Figura 4.1.c – Mesura de 5 segons en despertar *deep\_sleep* | Figura 4.1.d – Diferència temps *deep\_sleep* 5 segons i valor real |
| Figura 4.1.e – Mesura de 1 segon en despertar *deep\_sleep* | Figura 4.1.f – Diferència temps *deep\_sleep* 1 segon i valor real |

Figura 4.8 - Mesures *deep\_sleep*

## Càlcul SF òptim

Per el càlcul del SF òptim, s’ha fet server les funcions per obtenir el RSSI i el SNR. La funció utilitzada per el càlcul ha estat la següent:

Texto

Descripción generada automáticamente

Il·lustració 4.5 – Funció càlcul SF òptim

Els valors de referència utilitzats, han estat els ja esmentats en la l’apartat 3.9.

## Implementació protocol final

Un cop s’ha estudiat el comportament del ESP en els diferents àmbits en el qual es vol fer servir, es poden treure una sèrie de conclusions per realitzar el disseny més òptim per aquest protocol basat en LoRa. El codi de la implementació final del protocol es pot trobar en el següent enllaç: https://github.com/paugarcia32/TFG/tree/main/04%20-%20LoRa%20Based%20Protocol

### Escenari

Aquest protocol s’ha dissenyat com a prova de concepte, utilitzant un total de 4 nodes i un gateway. Cada node s’identifica amb un ID prèviament determinat de 8 bits. Si un missatge té un ID de tot 1s, es defineix com missatge Broadcast. El temps definit entre missatges Beacon es de 100 segons.

### Tipus de missatges

S’han definit 4 tipus de missatges en aquest protocol. Tots els missatges comencen sempre amb la mateixa capçalera.

La capçalera dels missatges està definida per varius camps:

1. Tipus de missatge (2 bits)
2. Flag identificador de gateway (1 bits)
3. Payload possibles millores (5 bits)
4. ID node transmissor (8 bits)
5. ID node receptor (8 bits)

Podem veure una representació de la capçalera a la següent figura:

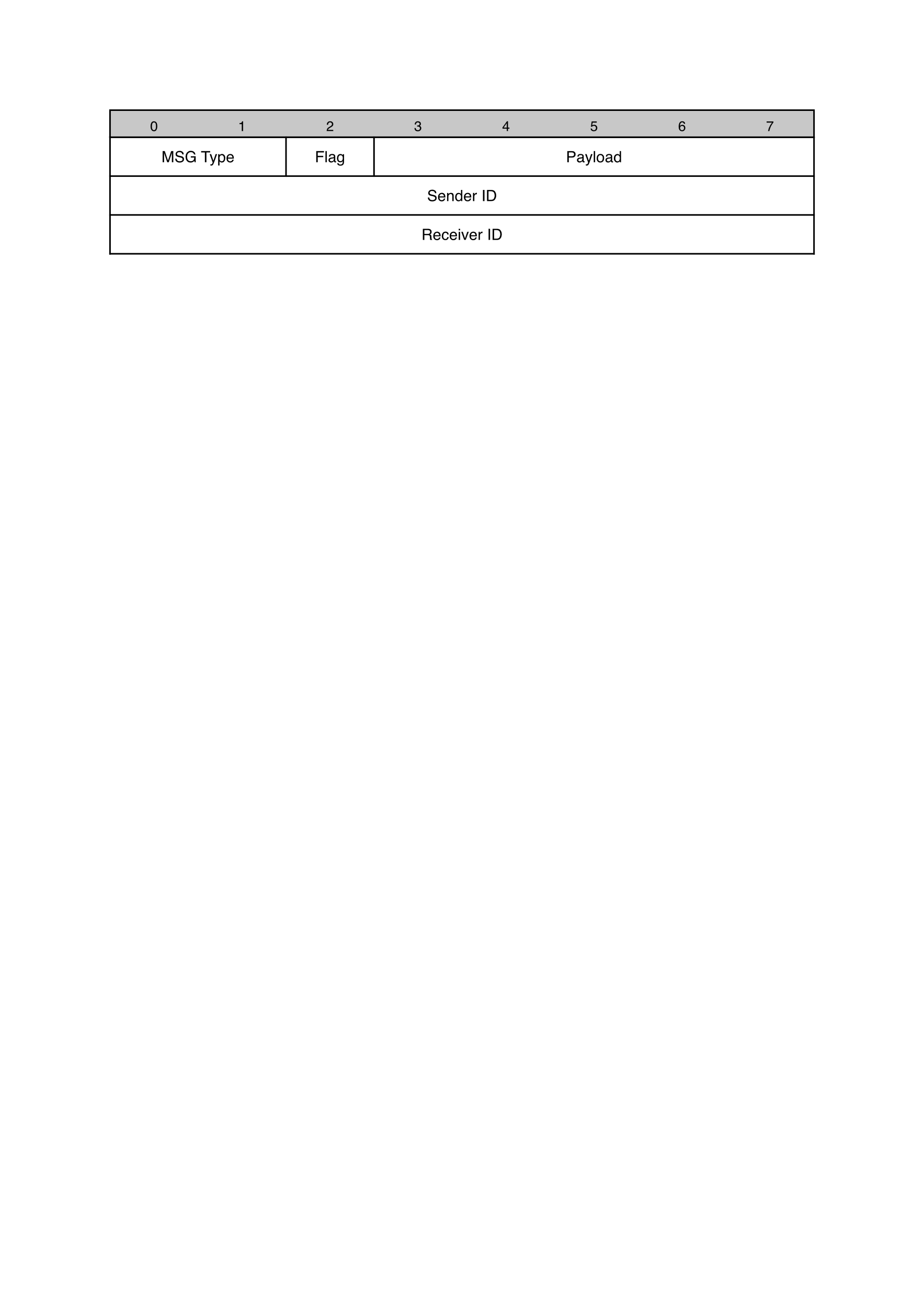


Figura 4.9 - Capçalera missatges

#### Beacon

El missatge *Beacon*, és exclusiu del gateway. Aquest missatge s’encarrega d’anunciar al gateway per el canal.

L’estructura del paquet Beacon és la següent:

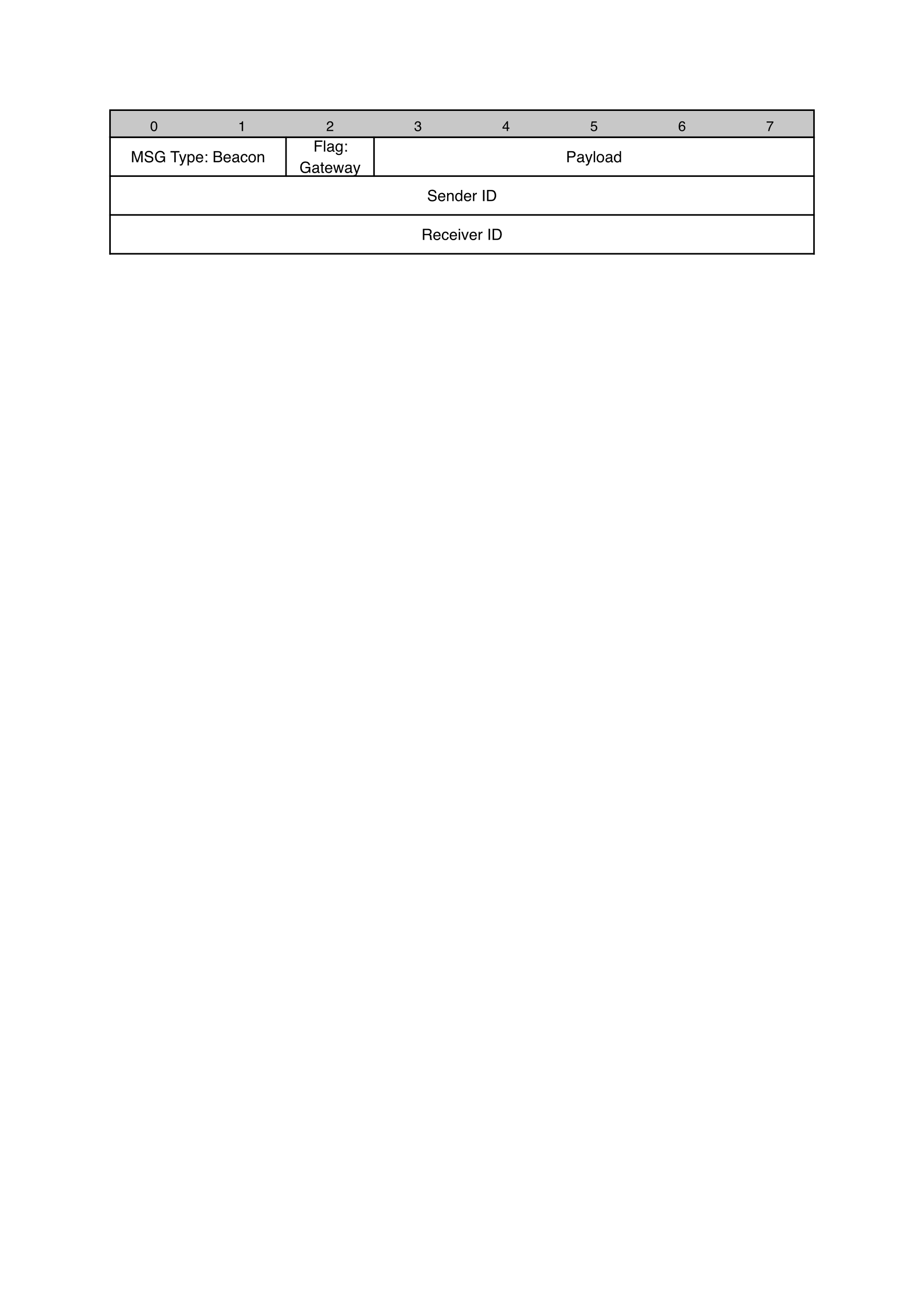


Figura 4.10 - Estructura Beacon

#### Request

El missatge Request, és el missatge que envia cada node al gateway, com a mode de confirmació per indicar que ha rebut el missatge Beacon.

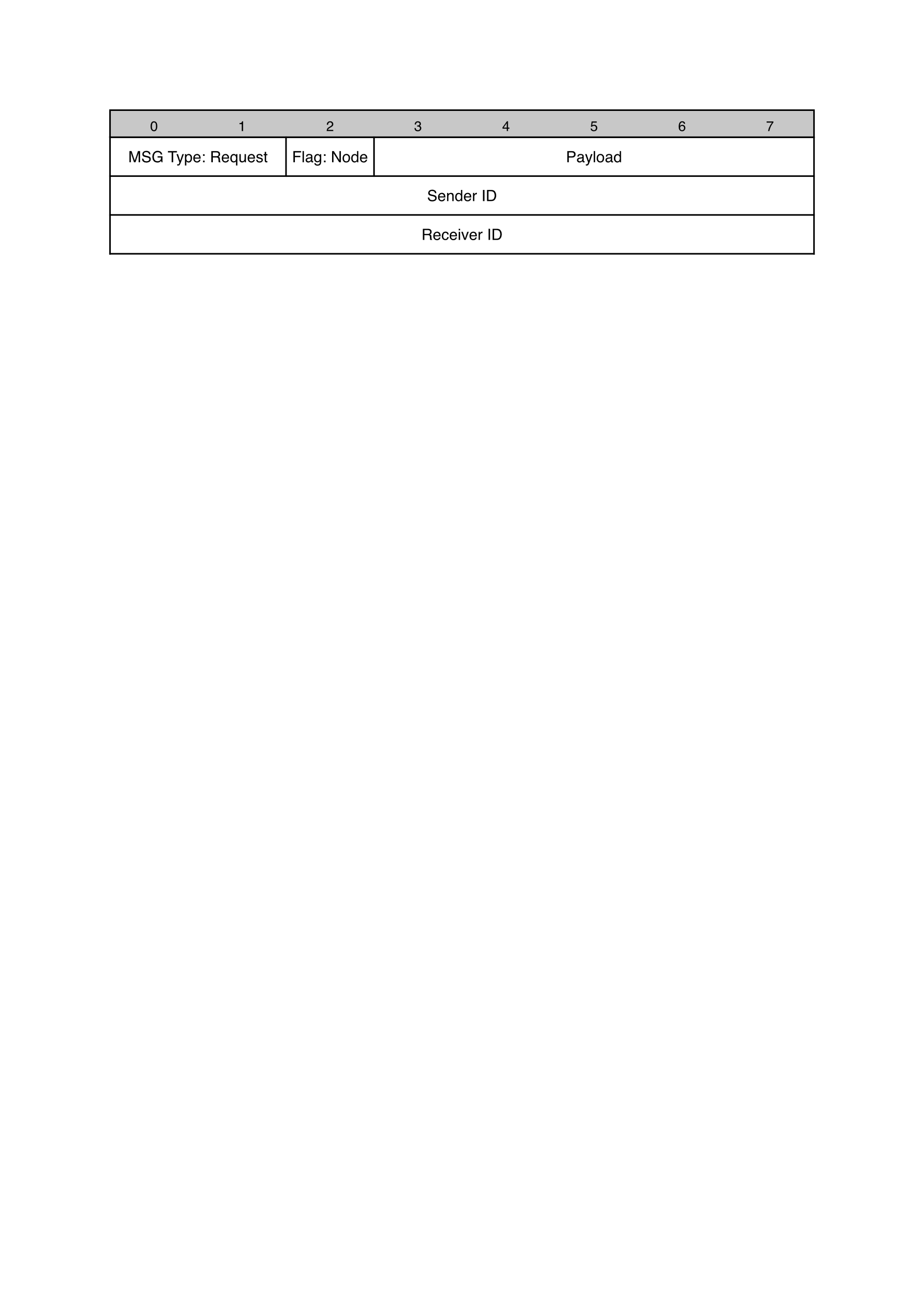


Figura 4.11 - Estructura Request

#### Schedule

El missatge Schedule, es el missatge que envia el gateway a tots els nodes que han respòs, indicant a cada node quin es el seu SF òptim i en quin torn l’han d’enviar.

L’estructura del missatge Schedule es la següent:

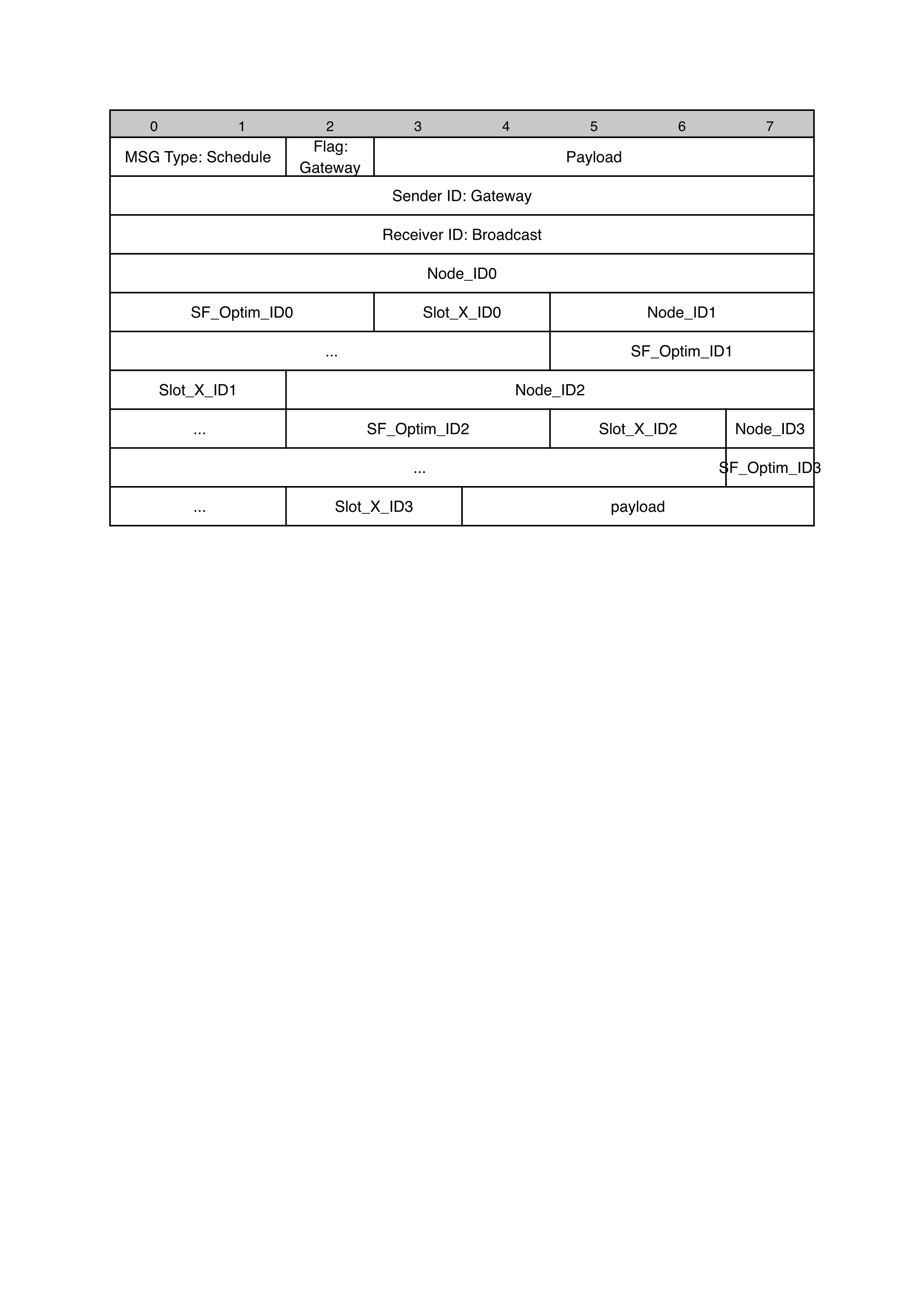


Figura 4.12 - Estructura Schedule

Els paràmetres del missatge Schedule són:

* ID node (8 bits)
* SF òptim (3 bits)
* Slot (2 bits)

En el pitjor dels casos, el missatge Schedule pot arribar a mesurar 76 bits, per tant 10 bytes.

#### Data

El missatge data, es el missatge que envien els nodes al gateway amb les dades que ha mesurat amb el sensor que porten.

En aquest cas, el missatge data porta 8 bits per transmetre temperatures. L’estructura de la trama Data es la següent:

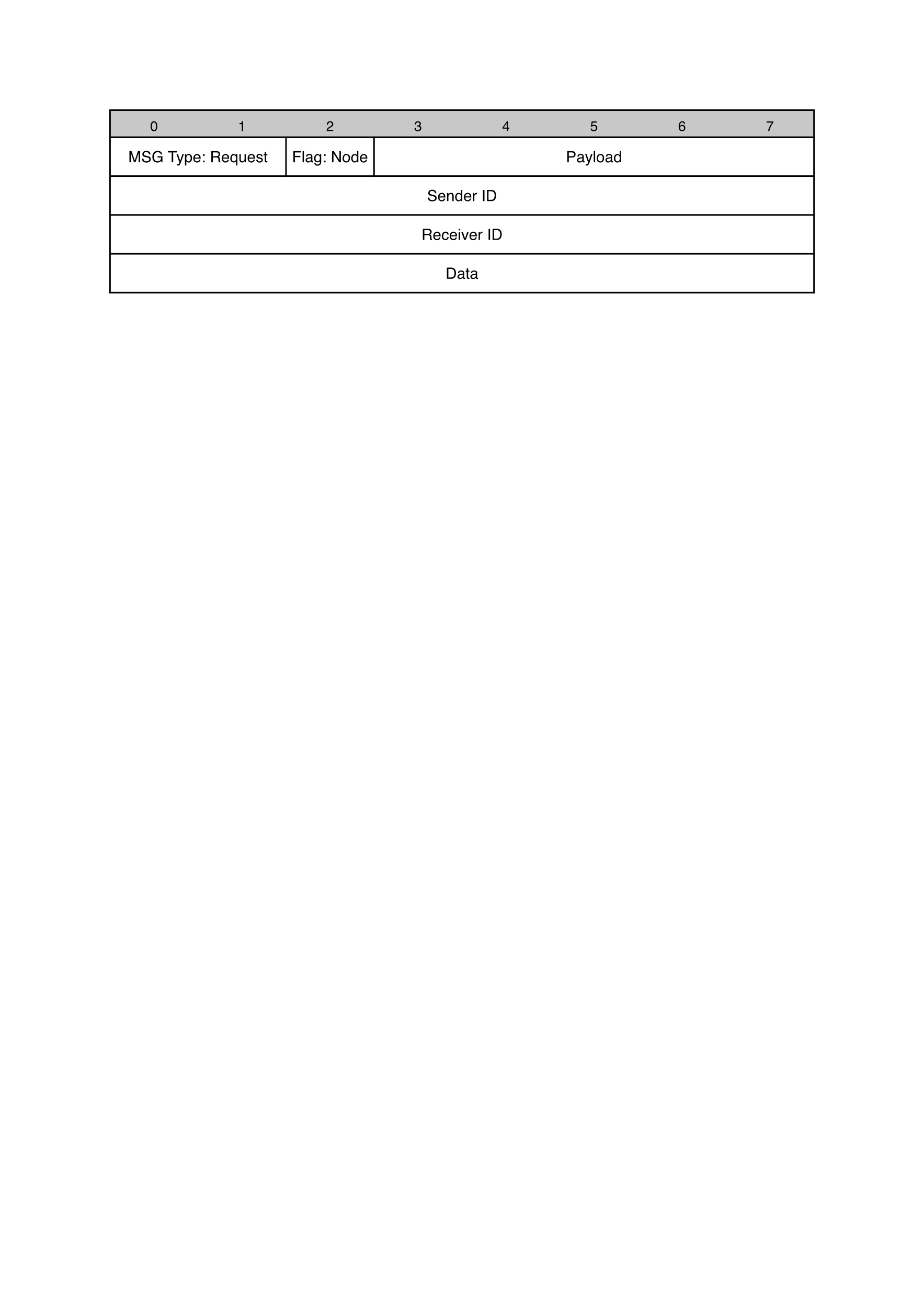


Figura 4.13 - Estructura Data

### Càlcul ToA de cada missatge

Un cop tots els tipus de missatges estan definits, podem calcular de forma precisa quins valors de ToA hi haurà per cada missatge. Com ja s’ha esmentat anteriorment, s’ha fet servir la calculadora que proporciona Semtech ® per realitzar aquest càlcul.

#### Beacon

Degut a que els missatges Beacon sempre s’envien amb els mateixos valors, només cal calcular un cop el ToA, ja que sempre serà el mateix. En aquest cas, com es pot veure en la següent taula, el valor del ToA es de 827,38 ms.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SF** | **Freqüència (MHz)** | **Amplada de banda (kHz)** | **Coding Rate** | **Payload (Bytes)** | **CRC** | **ToA (ms)** | **Duty Cycle (s)** |
| 12 | 868 | 125 | 4/5 | 3 | Si | 827,38 | 82,73 |

Taula 4.1 - ToA Beacon

#### Request

Els missatges Request, tenen exactament les mateixes característiques que els missatges Beacon, per tant, els resultats obtinguts són els mateixos. Es poden veure els resultats en la següent taula:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SF** | **Freqüència (MHz)** | **Amplada de banda (kHz)** | **Coding Rate** | **Payload (Bytes)** | **CRC** | **ToA (ms)** | **Duty Cycle (s)** |
| 12 | 868 | 125 | 4/5 | 3 | Si | 827,38 | 82,73 |

Taula 4.2 - ToA Request

#### Schedule

Per mesurar el ToA dels missatges Schedule, s’ha de tenir en compte que pot variar en funció del numero de nodes que hi hagin a l’abast i per tant, la quantitat de Bytes a enviar anirà augmentant en funció del número de nodes. Els resultats obtinguts han estat, en el millor dels casos, amb 1 node connectat, un ToA de 827 ms, i en el pitjor dels casos, amb els 4 nodes amb 991,22 ms de ToA.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nombre de Nodes** | **SF** | **Freqüència (MHz)** | **Amplada de banda (kHz)** | **Coding Rate** | **Payload (Bytes)** | **CRC** | **ToA (ms)** | **Duty Cycle (s)** |
| 0 | 12 | 868 | 125 | 4/5 | 3 | Si | 827,38 | 82,73 |
| 1 | 12 | 868 | 125 | 4/5 | 5 | Si | 827,38 | 82,73 |
| 2 | 12 | 868 | 125 | 4/5 | 7 | Si | 991,22 | 99,12 |
| 3 | 12 | 868 | 125 | 4/5 | 8 | Si | 991,22 | 99,12 |
| 4 | 12 | 868 | 125 | 4/5 | 10 | Si | 991,22 | 99,12 |

Taula 4.3 - ToA Schedule

#### Data

El ToA dels missatges Data, també varia bastant, ja que el valor que pot oscil·lar entre els missatges es el SF. En aquest cas, trobem el pitjor dels casos amb un ToA de 827,38 ms si el missatge Data ha d’utilitzar un SF de 12, i el millor dels casos, un ToA de 36,09 ms si el SF és de 7.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SF** | **Freqüència (MHz)** | **Amplada de banda (kHz)** | **Coding Rate** | **Payload (Bytes)** | **CRC** | **ToA (ms)** | **Duty Cycle (s)** |
| 12 | 868 | 125 | 4/5 | 4 | Si | 827,38 | 82,73 |
| 11 | 868 | 125 | 4/5 | 4 | Si | 413.69 | 41,36 |
| 10 | 868 | 125 | 4/5 | 4 | Si | 247,80 | 24,78 |
| 9 | 868 | 125 | 4/5 | 4 | Si | 123,90 | 12,39 |
| 8 | 868 | 125 | 4/5 | 4 | Si | 61,94 | 6,194 |
| 7 | 868 | 125 | 4/5 | 4 | Si | 36,09 | 3,609 |

Taula 4.4 - ToA Data

### Disseny protocol

Un cop s’han definit els tipus de missatges existents dintre del protocol, s’explicarà el disseny d’aquest, juntament amb el motiu de les decisions realitzades.

El protocol es pot definir en 3 grans etapes:

1. Descobriment
2. Organització
3. Enviament de dades

#### Etapa de descobriment

En l’etapa de descobriment, el gateway inicia anunciant-se per el canal mitjançant un missatge Beacon. Aquest missatge Beacon, sempre serà enviat amb un SF 12. A l’hora, els nodes estaran en mode recepció amb el SF configurat a 12 per rebre el missatge Beacon.

En el mateix instant en el que el gateway envia el missatge Beacon, es posa en mode recepció durant 10 segons per rebre els missatges Request.

En el instant en el que els nodes reben el missatge Beacon, esperen un temps aleatori de 7 segons per enviar el missatge Request. Abans d’enviar el missatge Request, els nodes escolten el canal per identificar si hi ha algun altre node utilitzant el canal en el mateix instant, en cas afirmatiu, el node que vol transmetre espera un altre segon a tornar a enviar.

En la següent figura, podem veure una representació de l’etapa de descobriment, suposant un escenari sense col·lisions entre els missatges d’ambdós nodes.

Imagen que contiene Forma

Descripción generada automáticamente

Figura 4.14 - Etapa de descobriment

##### Justificació del temps aleatori abans d’enviar un Beacon

El temps definit per enviar el missatge Request un cop el node ha rebut un missatge Beacon, es d’un interval aleatori entre 0 i 7 segons. Aquest interval s’ha definit amb aquest valor, ja que si els nodes envien les dades en el mateix instant, poden causar col·lisions, i no establir connexió correctament amb el gateway.

Degut a que el gateway, un cop que ha enviat el missatge Beacon comença el compte enrere de 10 segons per enviar el missatge Schedule, s’ha de tenir en compte el ToA del missatge Beacon per definir aquest temps. Per altre banda, també s’ha de considerar el ToA del missatge Request. Per últim, s’ha tingut en compte que els nodes, si el canal esta ocupat, s’esperen 1 segon més abans d’enviar. Tenint en compte que ambdós missatges tenen un ToA de 827 ms i hi ha 1 segon per repetir el missatge si el canal està ocupat, aquest interval de temps aleatori hauria de ser de 7,346 segons. Per aquest moriu, i per deixar una mica de marge a error, s’ha definit un temps aleatori de 7 segons.

#### Etapa d’organització

Mentre el gateway va rebent els missatges Request enviats per els nodes, guarda l’ID del node, juntament amb el SNR i el RSSI de cada missatge per calcular el SF òptim de cada node.

Un cop finalitza la finestra de recepció de missatges Request, el gateway envia un únic missatge Schedule per el canal, indicant als nodes en quina finestra de temps els hi toca enviar, i quin es el SF que han de fer servir per establir la comunicació.

Un cop els nodes reben els missatges Schedule, busquen dintre del missatge si el gateway els hi ha assignat una finestra de temps i amb quin SF han d’enviar.

Si el node troba el seu ID en el missatge Schedule, actualitza el seu SF, i, en funció de la finestra de temps que l’hi ha tocat, calcula quant de temps pot posar-se a dormir abans d’enviar i es posa a dormir.

En cas de que el node no hi sigui en el missatge Schedule, degut a una possible col·lisió del missatge Request amb un altre missatge, o a un eventual enviament del missatge Request fora del temps delimitat, el node es posarà a dormir x segons fins a posar-se en mode recepció per rebre el següent missatge Beacon, i tornar a començar de nou.

Entre el missatge Schedule i la recepció dels missatges Data per part dels nodes, el gateway deixa 10 segons als nodes per rebre el missatge Schedule, configurar el SF, calcular el temps per dormir, i posar-se a dormir.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Figura 4.15 - Etapa d'organització

#### Etapa enviament de dades

Un cop han passat els 10 segons, comencen les finestres temporals assignades a cada node. Cada finestra temporal té una durada de 5 segons i està assignada a un únic node, el qual, ha de ser capaç de transmetre el missatge Data sense problemes, degut a que cap altre node estarà utilitzant el canal. Per poder complir amb la normativa del *Polite Spectrum Access*, els nodes abans d’enviar escolten el canal.

Per altra banda, el gateway, per cada finestra temporal, va ajustant el seu SF en funció del node corresponent, per poder establir correctament la comunicació.

Un cop cada node ha enviat el seu missatge Data, reinicia el seu SF a 12 i es posa a dormir el temps restant fins a poder posar-se en mode recepció per rebre un altre missatge Beacon.

El gateway, per altre banda, deixa 10 segons per enviar les dades recollides dels nodes.

Quan passen els 10 segons, torna a un altre cop a l’etapa de descobriment de nodes.

Imagen en blanco y negro

Descripción generada automáticamente con confianza media

Figura 4.16 - Etapa enviament de dades

##### Justificació dels temps definits per cada slot

La ranura de temps de cada slot s’ha definit per ser de 5 segons. Dintre d’aquest interval de temps, els nodes han d’haver estat capaços d’enviar els missatges Data sense problemes.

Per entendre millor aquest valor de temps, ens hem d’ubicar en el moment en el que el Gateway envia el missatge Schedule. Un cop el gateway ha enviat el missatge Schedule, comença una compta enrere de 10 segons per posar-se en mode recepció, i començat el temps del primer slot.

Per aquest motiu, s’ha de tenir en compte el ToA del missatge Schedule en el càlcul del temps de cada slot. Un altre factor a tenir en compte per el càlcul del temps de cada slot, es el temps que triga el node en despertar-se del mode *deep\_sleep*, ja que un cop rebut el missatge Schedule, es posa a dormir durant 10 segons. Finalment, l’últim paràmetre a tenir en compte, es el ToA del missatge Data que envia el propi node.

En el pitjor dels casos, el missatge Schedule utilitzaria tot l’espai suportat en el seu tipus de missatge per enviar les dades als 4 nodes. Per altra banda, el pitjor des casos per el missatge Data seria haver d’enviar amb el SF 12.

Tenint en compte això, podem mesurar quant de temps seria el màxim temps de retard teòric entre un missatge Schedule i un Data: 1,888,6 segons (991,22ms (ToA Missatge Schedule) + 827,38ms (ToA missatge Data) + 70ms (Temps en despertar Node))

Aquest valor, és el valor teòric màxim que podria arribar a trigar en rebre el missatge Data el Gateway un cop ha començat el temps del slot. Degut a que aquest protocol és una proba de concepte, i està subjecte a possibles modificacions en un futur, s’ha deixat el temps del slot a 5 segons, per tenir un marge per incorporar nous paràmetres en els missatges.

En la següent figura, podem veure la representació del diagrama de flux del protocol:

Forma

Descripción generada automáticamente con confianza baja

Figura 4.17 - Diagrama de flux protocol

### Estats del node en memòria RTC

Per poder continuar l’execució normal del codi del node un cop s’ha despertat el mode *deep\_sleep*, s’ha decidit utilitzar la memòria RTC. Per fer això, s’han definit els estats 6 possibles estats en els quals el node es pot ubicar en el codi. En la següent figura es pot observar els estats del node:

Texto

Descripción generada automáticamente con confianza baja

Figura 4.18 - Funcionament estats node

Com es pot observar en la figura anterior, s’ha definit una funció per tractar els possibles casos en els quals el node es desperta del *deep\_sleep.* Aquesta funció només contempla els casos quan el node espera un slot per enviar dades, o quan el node espera un Beacon. Això es pel fet que el node només es posa a dormir en aquests dos estats. Tots els estats restant, es van actualitzant a mesura que s’executa el codi però no tenen una funció real. Aquests estats es mantenen degut a una eventual modificació en un futur, i facilitar l’actualització del codi.

## Proves de validació del protocol

Per validar el correcte funcionament del protocol, s’han fet diversos anàlisis del seu comportament, com per exemple, mesurar quant de temps hi ha entre tipus de missatges, o quant de temps està el node realment dormint. El codi de les proves realitzades, es pot trobar en el següent enllaç: https://github.com/paugarcia32/TFG/tree/main/04%20-%20LoRa%20Based%20Protocol/Analisys

### Anàlisis temps entre missatges

La primera prova realitzada per comprovar el funcionament del protocol, ha estat mesurar quant de temps hi ha entre tipus de missatges.

Aquesta mesura s’ha realitzat amb un gateway i un únic node. Ambdós han realitzar tot el procés del protocol vàries vegades. Tant el node com el gateway, envien per el port sèrie quin tipus de missatge envien, i si són el receptor o el transmissor.

La nomenclatura utilitzada en el port sèrie ha estat la següent:

* Transmissió d’un Beacon: *BTX*
* Recepció d’un Beacon: *BRX*
* Transmissió d’un Request: *RTX*
* Recepció d’un Request: *RRX*
* Transmissió d’un Schedule: *STX*
* Recepció d’un Schedule: *SRX*
* Transmissió d’un Data: *DTX*
* Recepció d’un Data: *DRX*

Un aspecte a tenir en compte abans d’analitzar els resultats obtinguts, és quin comportament tindrà el Node en relació al slot per enviar la trama Data. Com és l’únic node en l’escenari, sempre tindrà el slot 0. Aquest apartat es important per entendre el comportament de dispositiu.

En aquest anàlisi, es compararà els resultats esperats prèviament definits en la definició del protocol, i els resultats obtinguts en aquesta proba. Podem veure els resultats obtinguts en la següent taula.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Temps entre missatges** | **Temps mínim (s)** | **Temps màxim (s)** | **Mitja (s)** | **Numero de vegades** |
| BRX -> RTX (Node) | 2,307421 | 3,214218 | 2,810701 | 3 |
| SRX -> DTX (Node) | 10,1866 | 10,19472 | 10,18934 | 3 |
| DRX -> BTX (Gateway) | 30,79999 | 30,85989 | 30,83683 | 3 |
| BTX -> BTX (Gateway) | 51,4169 | 51,4243 | 51,42173 | 3 |

Taula 4.5 - Resultats mesures temps entre missatges

Per poder entendre correctament si aquests valors són els esperats o no, en la teula següent podem veure una comparació amb els valors que s’havien definit prèviament en l’apartat 4.7, tenint en compte que el node sempre s’ubica en el slot 0 per transmetre les dades.

|  |  |  |
| --- | --- | --- |
| **Temps entre missatges** | **Mitja (s)** | **Valor teòric (s)** |
| BRX -> RTX (Node) | 2,810701 | > 7 |
| SRX -> DTX (Node) | 10,18934 | 10 |
| DRX -> BTX (Gateway) | 30,83683 | 30 |
| BTX -> BTX (Gateway) | 51,42173 | 50 |

Taula 4.6 - Comparació valors teòrics amb valors mesurats

#### Comparació del temps que triga el node en respondre amb un Request

En primer lloc, podem veure el temps que triga el node entre que ha rebut el missatge Beacon i envia el missatge Request. En la implementació del protocol, s’havia definit que el node esperés un temps aleatori de 7 segons abans d’enviar el missatge, per evitar possibles col·lisions amb altres nodes.

En la taula 4.2, podem comparar el valor màxim teòric i el valor mitja obtingut. Es pot veure que el node compleix amb el valor esperat.

Per altre banda, en la taula 4.1, s’observa un valor màxim de 3,21 segons, per tant, no supera el temps màxim definit i es comporta adequadament.

#### Comparació del temps que triga el node en rebre el missatge Schedule i enviar les dades

En aquest apartat, tenint en compte que el node s’ubica en el slot 0, només ha d’esperar els 10 segons entre el Schedule rebut, i el missatge de Data que ha de rebre. Per altra banda, no pot superar els 15 segons, ja que estaria utilitzant un slot que no l’hi correspon.

En la taula 4.2, podem observar que els temps obtinguts són els esperats, ja que el node tant bon punt han passat els 10 segons, envia el missatge Schedule.

#### Comparació del temps que triga el Gateway en tornar a enviar un Beacon un cop rebut l’últim missatge Data

En aquest apartat, s’ha de tornar a considerar la ubicació del node en el slot 0, ja que després de que el node enviï el missatge Data, el gateway no espera rebre cap altre missatge.

En la taula 4.2, es pot observar com el gateway envia el següent Beacon una mitja de 30 segons després d’haver rebut el missatge Data. Aquest comportament es l’esperat, degut a que es el temps de 20 segons que duren els 4 slots (5 segons cadascun) més els 10 segons que es reserva el gateway per tractar les dades.

#### Comparació de temps entre missatges Beacon

Aquesta mesura és de vital importància per el sincronisme amb els nodes i per el compliment de la regulació del ToA i el Duty Cycle. Com es pot observar en la taula 4.2, el temps compleix amb l’expectativa. El Gateway triga 51,42 segons de mitja en tornar a enviar un missatge Beacon.

### Anàlisis temps en *deep\_sleep* dels nodes

Per a la realització d’aquesta mesura, s’ha utilitzat un únic node, degut a que la resta de nodes funcionen de la mateixa manera.

En aquesta prova, s’han afegit missatges per el port sèrie cada cop que el node es va a dormir o es desperta. Imitant la prova anterior, s’ha utilitzar un script en Python per guardar les mesures en un CSV i fer l’anàlisi dels resultats obtinguts.

El temps esperat que el node estigui en *deep\_sleep*, es de 35 segons en cada cicle, tenint en compte els 10 segons que ha de estar dormint entre el missatge Schedule y Data, i els 25 segons que ha d’estar dormint entre que envia el missatge Data i es torna a posar en mode recepció per rebre el nou Beacon.

#### Comparació temps entre temps teòric i temps mesurat entre missatges Schedule i Data

Com es pot visualitzar a la taula 4.3, el temps que està el node en mode *deep\_sleep*, es el temps esperat. Aquest temps és el temps que el node està dormit esperant al seu torn d’enviar el missatge de Data. Com el node s’ubica en el slot 0, aquest temps sempre haurà de ser de 10 segons.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Iteracions proba** | **Mitja (ms)** | **ValorMàxim (ms)** | **Valor Mínim (ms)** | **Valor teòric (ms)** |
| 10 | 10087,2 | 10089 | 10085 | 10000 |

Taula 4.7 - Comparació *deep\_sleep entre missatges Schedule i Data*

#### Comparació temps entre temps teòric i temps mesurat entre missatges Data i Beacon

Com es pot visualitzar a la taula 4.4, el temps que el node està en *deep\_sleep* és el temps esperat. Aquest temps és el temps que el node es posa a dormit un cop ja ha enviat el seu missatge Data, a l’espera de posar-se en mode recepció per rebre el següent Beacon. En aquest cas, com el node s’ubica en el slot 0, ha d’estar en mode sleep els 15 segons que duren els altres 3 slots, més els 60 segons que el gateway reserva per tractar les dades.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Iteracions proba** | **Mitja (ms)** | **ValorMàxim (ms)** | **Valor Mínim (ms)** | **Valor teòric (ms)** |
| 10 | 25073,6 | 25082 | 25061 | 25000 |

Taula 4.8 - Comparació deep\_sleep entre missatges Data i Beacon

### Anàlisi funcionament amb varis nodes

En aquesta prova, s’ha comprovat el correcte funcionament del protocol utilitzant 2 nodes i 1 gateway. En les següents figures, es mostren els missatges que apareixen en els dispositius per el port sèrie per poder analitzar que està passant.

#### Anàlisi funcionament Gateway

En la següent figura, podem veure els missatges que ha tret el gateway per el port sèrie. En la figura s’observa com per Node, guarda el seu ID i el seu RSSI i SNR. En ambdós casos, té el valor del SF calculat té sentit, ja que ambdós nodes estaven a menys d’un metre de distància.

També es pot visualitzar com per cada slot, canvia el seu SF per poder establir la comunicació més optima amb cada node.

Finalment, podem veure com en aquest cas, el gateway ensenya els resultats obtinguts per el port sèrie.

Texto

Descripción generada automáticamente

Figura 4.19 - Implementació protocol - Gateway

#### Anàlisi funcionament nodes

En les següents figures, es pot observar el port sèrie d’ambdós nodes.

Inicialment, podem veure com desprès de rebre el missatge Beacon, ambdós nodes esperen un temps aleatori de 4322 ms i 1677 ms respectivament.

Per altra banda, podem observar com desprès de rebre els missatges Schedule, els nodes es posen a dormir el temps que els hi toca en funció del seu slot, en aquest cas, 15 segons i 10 segons respectivament.

Finalment, es pot observar com desprès d’enviar els missatges Data, ambdós nodes se’n van a dormir 20 segons abans de rebre un altre cop els missatges Beacon.

Texto

Descripción generada automáticamente

Figura 4.20 - Implementació protocol - Node 1

Texto, Carta

Descripción generada automáticamente

Figura 4.21 - Implementació protocol - Node 2

#### Anàlisi possible col·lisió amb missatges Request

Com ja s’ha esmentat anteriorment, un dels efectes possibles pel fet d’ utilitzar CSMA com a protocol d’accés al medi, era que hi haguessin col·lisions en els missatges Request.

En la següent figura, podem veure el comportament del node en aquest cas:

Texto

Descripción generada automáticamente

Figura 4.22 - Solució col·lisions missatges Request

Com es pot visualitzar en la figura anterior, si el node no s’ubica dintre del missatge Schedule, es posa a dormir el temps restant fins al següent missatge Beacon.

# Discussió

## Validació funcionament protocol

### Temps entre missatges

El temps mesurat entre els missatges, és un bon indicador del correcte funcionament del protocol. Els resultats obtinguts, són pràcticament idèntics als esperats.

### Temps en mode *deep\_sleep*

Els resultats de les mesures realitzades amb el mode *deep\_sleep*, han obtingut els resultats teòrics esperats. Com a implementació de la primera prova realitzada en l’apartat 4.8.2, i mirant els resultats obtinguts, es pot dir que el mode *deep\_sleep* funciona correctament.

### Funcionament amb varis nodes

Els resultats obtinguts en aquesta prova, han estat molt satisfactoris. El protocol ha funcionat correctament.

L’escenari plantejat en aquesta mesura, ha intentat simular un escenari real, on els nodes s’encenen i es deixen funcionar durant un temps indefinit. Degut a que el temps entre missatges Beacon es sempre el mateix, ha permès una correcta sincronització entre els nodes i el gateway.

## Problemàtica configuració Wi-Fi mitjançant BLE amb el node

La idea inicial respecte al node, consistia que si el ESP32 trobava una configuració Wi-Fi, en comptes d’utilitzar el protocol LoRa, envies les dades directament al servidor, sense haver de passar pel gateway per enviar les dades.

La configuració del Wi-Fi mitjançant BLE s’havia dissenyat per ser implementada just quan el ESP32 es despertés d’un reinici. Aquest disseny s'ha pogut implementar en el codi del gateway, ja que no es posa a dormir en cap moment. En contraposició, el node, sí que es posa a dormir, i com ja s’ha explicat anteriorment, quan el ESP32 es desperta del mode *deep\_sleep*, perd la memòria del dispositiu, i torna a executar el codi de l'inici.

Per tant, cada cop que el node es desperta del mode *deep\_sleep*, en comptes de fer la tasca que hauria de fer (per exemple, enviar un missatge Data o posar-se en mode recepció d’un missatge Beacon), abans executava el codi per la configuració del Wi-Fi mitjançant BLE.

El mode d’actuació del *deep\_sleep*, juntament amb la implementació actual del node per recuperar l’estat anterior abans de posar-se a dormir, fa impossible la implementació d’aquest apart del codi per proporcionar tant el Wi-Fi com el protocol LoRa com a solucions de connectivitat en un sol dispositiu ESP32.

## Possibles millores

### Col·lisió missatges Request

La col·lisió dels missatges Request ve donada a la utilització de ALOHA com a mètode d’accés al medi. Una possible millora que permetria evitar aquesta circumstància, seria utilitzar també TDMA com a mètode d'accés al medi.

Per aconseguir aquesta hipotètica implementació sense la necessitat d’enviar un missatge Schedule anteriorment, podria donar-se per exemple, fent que cada node en funció del seu ID, tingui assignat un slot.

Per altre banda, aquesta implementació té un inconvenient, i és que si s’utilitza aquest mecanisme en el qual cada node ja sap quin slot li correspon en funció del seu ID, és perd el dinamisme i podria donar-se la circumstància en la qual un node hagés d’esperar molt de temps per enviar les dades tot i que el canal no estigués utilitzat.

### ID dinàmic dels nodes

Actualment, en aquesta proba de concepte, l’ID dels nodes ja bé pre-definit. Per tal de millorar la implementació, es podria fer que l’usuari pogués configurar l’ID del node utilitzant BLE i guardar el valor en memòria no volàtil, utilitzant les proves realitzades en aquest treball.

### Un codi unificat per els nodes i gateway

En aquesta implementació, hi ha 2 codis diferents, un primer per el gateway, i un segon per els nodes. Una millora seria unificar els codis en un de sol, i permetre a l’usuari escollir quina funció vol que faci el ESP32 que utilitza.

### Implementació del suport per varis gateway

Aquesta proba de concepte, no contempla la possibilitat d’utilitzar més d’un gateway. Una millora d’aquest protocol seria la possibilitat d’incorporar varis gateway en l’escenari, augmentant així la cobertura dels nodes, i optimitzant els ToA de molts missatges, ja que els nodes podrien escollir quin gateway utilitzar.

### Confirmació recepció missatges

En la implementació actual, no s’ha fet servir cap mecanisme de confirmació de recepció de missatges. Podria ser interesant utilitzar un mecanisme de confirmació de missatges pels missatges DATA.

Dues possibles implementacions d’aquest mecanisme, podrien fer-se de la següent manera:

Una primera forma de realitzar aquesta millora, podria venir donada si s’aprofités el temps sobrant que hi ha en els slots assignats a cada node. Aquesta implementació podria aprofitar el fet que cada el node que ha enviat el missatge i el gateway ja tenen el mateix SF, i per tant, poden establir comunicació sense haver de sincronitzar-se un altre cop.

Per altre banda, en comptes d’utilitzar els slots per enviar de manera personalitzada a cada node un missatge de confirmació, podria implementar-se un missatge de resum enviat pel gateway indicant si ha rebut un missatge de data o no de cada node. En aquesta casuística, per sincronitzar els nodes i el gateway per utilitzar el mateix SF, podria utilitzar-se el SF12 com ja es fa amb altres missatges.

## Comparació utilització Duty Cycle vs Listen Before Talk

Com ja s’ha pogut veure durant el desenvolupament del treball, el factor més determinant ha estat el compliment de les normatives vigents en relació a la utilització de l’espectre. Per aquest motiu, s’ha decidit comparar com hagés canviat el resultat final del protocol, si s’hagés canviat d’enfoc.

L’enfoc final del protocol, ha estat complint el *Polite Spectrum Access*, el qual ha permès acotar el Duty Cycle entre Beacon a 50 segons.

Això ha estat possible degut a que s’ha respectat el *Listen Before Talk*, ja que els nodes, per poder enviar un missatge, han d’escoltar el canal.

Per altra banda, si s’hagés escollit l’enfoc relacionat amb el compliment del 1% del Duty Cycle, el Duty Cycle mínim a complir hagés estat de 100 segons, el doble de l’actual. Això ve donat pel fet que el missatge Schedule, en e pitjor dels cassos, pot tenir un ToA d’un segon.

## Comparativa amb altres propostes realitzades

Una de les propostes que s’havia plantejat com a protocol final, plantejava la possibilitat de definir en funció del SF òptim, quants missatges Data podria enviar el node. Aquesta proposta venia donada tenint en compte la intenció de respectar el 1% de Duty Cycle.

Degut a que si el SF passa de 12 a 11, el ToA es redueix quasi a la mitat, i així successivament fins arribar al SF 7, s’havia plantejat definir quants cops podia enviar el node un missatge Data abans de tornar a rebre el missatge Beacon, i així optimitzar al màxim el DC.

Aquesta proposta s’ha descartat davant l’actual, degut a la complexitat que requeriria al gateway haver de sincronitzar-se tants cops amb els nodes, ja que hauria de calcular per cada node el nombre de missatges que hauria d’enviar, i en quin moment podria fer-ho, per poder sincronitzar-se amb el valor del SF.

Durant el procés de prova d’aquest prototip, s’havia aconseguit realitzar una implementació amb un únic node, per tant, els temps de cada finestra eren molt simples de calcular.

Donat a que s’ha decidit complir la normativa respecte el *Polite Spectrum Access,* el temps d’enviament de dades es pot optimitzar molt més, ja que el temps entre els missatges Beacon és molt menor.

Es pot trobar la implementació d’aquesta proposta del protocol en el següent enllaç: <https://github.com/paugarcia32/TFG/tree/main/03%20-%20Results%20of%20the%20measurements/Old%20Protocol%20Implementations/CSMA-LoRa-Bit>

## Continuació del projecte

En el cas d’una eventual continuació del projecte, més enllà d’aquest treball, es proposa una mesura per poder fer-ho.

Com que aquest projecte és de codi obert, està totalment disponible a GitHub en el següent enllaç: <https://github.com/paugarcia32/TFG> .

Es proposa la realització d’un Fork del projecte en comptes de continuar contribuint en el repositori original. D’aquest amanera es proporciona una solució simple per poder veure l’abast d’aquest treball, i separar-se de l’eventual treball futur. Alhora, es pot veure qui ha participat en cada etapa del projecte, sense treure protagonisme a l’autor de cada etapa.

## Estudi Sostenibilitat

### Desenvolupament del treball final d’estudis

#### Punt de vista ambiental

Aquest projecte utilitza un enfoc molt vinculat amb l’impacte ambiental que pot arribar a tenir. El fet de ser un projecte d’IoT, ja té un enfoc molt orientat a reduir al màxim el seu impacte.

En concret, en aquest projecte, com ja s’ha explicat en més detall en l’apartat 4.8.2, cada node està de mitja 85 segons dormint i 15 segons despert. Aquest 85% del temps, el node només consumeix 10 µA. Aquest fet, fa que el consum sigui ínfim, i les bateries que es poden utilitzar en un futur, podrien arribar a durar dècades.

#### Punt de vista econòmic

En aquest projecte, el material utilitzat ha estat molt escàs i molt econòmic. Els microcontroladors ESP32 són força econòmics, tenint un cost d’uns 20€ com a màxim. Per altra banda, es requereix d’un PC per programar-los, però aquest PC no requereix de grans especificacions per programar els ESP32, ja que tot el codi és executat en el mateix microcontrolador, i no des del PC utilitzat.

A més a més, aquest projecte requereix d’antenes LoRa. En el nostre cas, s’ha optat per utilitzar les plaques Heltec LoRa V3, que ja porten un transceptor incorporat, d’aquesta manera s’ha estalviat en haver de comprar el transceptor apart.

#### Punt de vista social

Avui en dia, el sector de l’IoT està augmentant molt i hi ha moltes implementacions relacionades amb LoRa i LoRaWAN. Malauradament, els distribuïdors dels microcontroladors utilitzats no proporcionen cap mena de codi de conducta per al seu ús.

### Execució del projecte

#### Punt de vista ambiental

L’impacte ambiental del projecte és molt escàs. El fet de ser un projecte IoT ja implica l’optimització del recursos per utilitzar-ne l’estrictament necessari per la implementació d’aquest. Com ja s’ha esmentat anteriorment, aquests dispositius estan pensats per estar dormint la majoria del temps, el qual implica una reducció dràstica del seu consum.

La vida útil del projecte, pot arribar a durar vàries dècades, ja que pel fet de tenir un consum tan reduït, les bateries no requereixen de ser canviades en molts anys. Un cop una bateria s’esgota, només cal canviar-la i seguir utilitzant el microcontrolador com abans.

El projecte podria fins i tot reduir encara més el seu consum. Com s’ha esmentat anteriorment, el dispositiu està dissenyat per dormir un 70% del temps. No s’ha ajustat més degut a que aquest protocol encara és una prova de concepte, però els temps poden ser modificats amb facilitat, per poder millorar l’impacte ambiental.

#### Punt de vista econòmic

El cost estimat d’aquest projecte, pot ser d’uns 100 € tenint en compte que es poden arribar a utilitzar 5 ESP32 (4 nodes i 1 gateway). Per utilitzar les funcionalitats implementades només es requereix d’un node i un gateway, per tant, el cost mínim seria d’uns 40€.

Per altra banda, si es vol utilitzar els nodes com a dispositius portables, es requereix d’una bateria externa, la qual pot tenir un cost d’uns 10€ extra.

Aquest projecte pot ser implementat en altres projectes relacionats amb la ciència ciutadana i el Iot. El propòsit inicial del projecte era una implementació d’un protocol per a una estació meteorològica, però els nodes poden canviar els tipus de sensors utilitzats i enviar altres dades que poden ser útils, com per exemple sensors de moviment o sensors de llum.

#### Punt de vista social

Aquest projecte beneficia principalment a la ciutadania. La correcta implementació del protocol afegeix una nova possibilitat de connectivitat a usuaris que no tinguin connectivitat Wi-Fi, i requereixin d’enviar dades a un altre punt.

La implementació d’aquesta solució de connectivitat proporciona una forma molt simple de ser utilitzada, ja que no requereix una configuració inicial. Únicament connectant proporcionant corrent elèctrica als dispositius ja permet als usuaris implementar aquesta solució de connectivitat.

Aquesta nova solució de connectivitat inicialment està implementada per una estació meteorologia, però pot ser implementada en diferents escenaris de ciutats intel·ligents i conduir l’enfoc del projecte a solucionar altres problemàtiques socials.

### Riscos i limitacions

#### Punt de vista ambiental

Aquest projecte pot incrementar la seva petjada ambiental amb possibles noves implementacions. Donat que aquest treball és únicament una fase inicial del projecte, pot variar molt l’enfoc ambiental del mateix, i veure afectada la petjada ambiental.

Aquest projecte s’ha implementat pràcticament en la seva totalitat utilitzant ESP32 que ja s’havien utilitzat en anteriors projectes. El fet de provar el correcte funcionament del protocol amb el sincronisme amb altres nodes, ha requerit de l’adquisició de nous dispositius. Per altra banda, aquest dispositius poden ser utilitzats per la persona que continuï el projecte, i reduir la petjada ambiental relacionada amb el desenvolupament del projecte.

#### Punt de vista econòmic

En aquest projecte no hi ha hagut limitacions econòmiques pel fet del baix cost del material utilitzat.

La continuïtat del projecte recau en la intenció de la ciutadania en la seva evolució, sent principalment els estudiants de les universitats els principals responsables de la continuació d’aquest projecte.

#### Punt de vista social

Aquest projecte no pot proporcionar cap resultat perjudicial per a cap segment de la població degut a que gran part del projecte, es centra únicament el solucions realitzades per software.

Com aquesta solució de connectivitat s’utilitza per projectes de IoT, podria donar-se el cas de ser utilitzat en països subdesenvolupats. Donada aquesta casuística, al tractar-se d’un país subdesenvolupat, podria generar algun tipus de dependència al correcte funcionament d’aquesta solució de connectivitat, i perjudicar algun tipus d’infraestructura que requereixi del correcte funcionament d’aquesta solució en cas de fallada.

# Conclusions i treball futur

Aquest treball, ha estat un primer pas pel desenvolupament d’un projecte comercial per proporcionar un producte que aporti diferents solucions de connectivitat per a una estació meteorològica.

Durant el desenvolupament del treball, s’ha efectuat una recerca per escollir el millor software donat un hardware en concret, en aquest cas, una placa ESP32S3 Heltec LoRa Wi-Fi V3. S’han realitzat vàries proves per poder entendre el funcionament d’aquesta placa de desenvolupament en certs escenaris, per posteriorment dissenyar i implementar de la manera més òptima possible, una proba de concepte per la implementació d’un protocol de IoT basat en LoRa.

Aquestes proves realitzades al ESP, han consistit en comprovar el comportament del dispositiu en els modes d’estalvi d’energia, i en l’emmagatzematge de dades en memòria no volàtil. Per altra banda, també s’han estudiat altres solucions de connectivitat, com BLE, per a una possible configuració dels dispositius.

El resultat final d’aquest treball, consisteix en una primera proba de concepte del funcionament d’un protocol basat en LoRa, que permet implementar alguns dels avantatges que ja proporciona LoRa de forma nativa, com podria ser la gran cobertura o la baixa potència de transmissió i millorar algunes de les carències que té, com podria ser la falta d’estructura en els tipus de missatges, o una correcta forma d’accés al medi per evitar possibles col·lisions.

Com ja s’ha esmentat anteriorment, aquest treball és l’inici d’un producte comercial, i encara requereix un treball més extens per a la seva implementació final. Alguns aspectes que es podrien implementar en un possible treball futur, podrien ser un petit servidor amb el qual el gateway pogués comunicar-se, o en cas de trobar una xarxa Wi-Fi, els nodes també podrien enviar les dades directament sense enviar primer al gateway. Per altra banda, també podria implementar-se una aplicació web o una aplicació mòbil per la configuració dels dispositius o fins i tot, per visualitzar les dades recollides amb els nodes.

Aquest treball finalitza amb la implementació del protocol dissenyat, i amb una sèrie de possibles millores per una eventual continuació del project

# Bibliografia

|  |  |
| --- | --- |
| [1] | «ETSI,» [En línia]. Available: https://objects.githubusercontent.com/github-production-repository-file-5c1aeb/151064439/12129485?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAVCODYLSA53PQK4ZA%2F20250118%2Fus-east-1%2Fs3%2Faws4\_request&X-Amz-Date=20250118T120653Z&X-Amz-Expires=3. [Últim accés: 1 2025]. |
| [2] | J. H. A. S. E. D. P. Martjin Saelens, «Impact of EU duty cycle and transmission power limitations for sub-GHz LPWAN SRDs: an overview and future challenges,» 2019. [En línia]. Available: https://jwcn-eurasipjournals.springeropen.com/articles/10.1186/s13638-019-1502-5. [Últim accés: 1 2025]. |
| [3] | «ESPBoards,» 16 10 2023. [En línia]. Available: https://www.espboards.dev/blog/esp32-power-optimisation/. [Últim accés: 12 2024]. |
| [4] | «espressif.com,» [En línia]. Available: https://docs.espressif.com/projects/arduino-esp32/en/latest/tutorials/preferences.html. [Últim accés: 11 2024]. |
| [5] | K. N. i. R. D. W. Gilang Hijrian Fahreja, «The Effect of Spreading Factor Value on the Number of Gateways in the LoRaWAN Network at Bandung City,» 13 12 2023. [En línia]. Available: https://www.jocm.us/2023/JCM-V18N12-768.pdf. [Últim accés: 10 2024]. |
| [6] | Semtech. [En línia]. Available: https://www.semtech.com/design-support/lora-calculator. [Últim accés: 12 2024]. |
| [7] | M. «unisgned,» [En línia]. Available: https://unsigned.io/website/understanding-lora-parameters/. [Últim accés: 11 2024]. |
| [8] | «TheThingsNetwork,» [En línia]. Available: https://www.thethingsnetwork.org/docs/lorawan/lora-phy-format/. [Últim accés: 22 2024]. |
| [9] | Sebas, «Medium,» 12 2024. [En línia]. Available: https://medium.com/@Sabasacustico/haciendo-iot-con-lora-capitulo-2-tipos-y-clases-de-nodos-3856aba0e5be. |
| [10] | «GitBooks,» [En línia]. Available: https://pedrobq.gitbooks.io/fundamentos-de-redes/content/protocolos\_de\_acceso\_aleatorio/aloha.html. [Últim accés: 1 2025]. |
| [11] | J. G. Carmenate, «ProgramarFacil,» [En línia]. Available: https://programarfacil.com/esp8266/esp32/. [Últim accés: 1 2025]. |
| [12] | L. Llamas, «luisllamas,» 22 09 2023. [En línia]. Available: https://www.luisllamas.es/esp32-consumo-energia/. [Últim accés: 01 2025]. |