

Taula de continguts

1	INTRODUCCIÓ	1
2	PROTOCOLS IOT.....	2
2.1	LoRa.....	2
2.1.1	Paràmetres LoRa	2
2.2	LoraWAN	3
2.2.1	Classes de nodes de LoRaWAN	4
2.2.2	Modes d'activació	4
2.3	Duty Cycle vs Listen Before Talk.....	5
3	MÈTODES I MATERIAL UTILITZAT.....	8
3.1	Software i hardware utilitzat.....	8
3.1.1	Hardware utilitzat	8
3.1.2	Software Utilitzat	10
3.2	Setup inicial.....	10
3.2.1	Instal·lació de l'entorn	11
3.2.2	Instal·lació de la placa.....	11
3.2.3	Instal·lació de llibreries necessàries	12
3.3	Estudi llibreries LoRa.....	13
3.3.1	Unofficial Heltec	13
3.3.2	Arduino LoRa	14
3.3.3	RadioLib	14
3.3.4	Comparativa entre llibreries LoRa	15
3.4	Estudi llibreries LoRaWAN.....	16
3.4.1	Beelan-LoRaWAN.....	16
3.4.2	Arduino LoRaWAN.....	16
3.4.3	Comparativa entre llibreries LoRaWAN.....	16
3.5	Estalvi d'energia i modes Sleep.....	17
3.5.1	Modes d'energia	17
3.6	Guardar dades en memòria no-volàtil i memòria RTC	20
3.6.1	Memòria no-volàtil.....	20
3.6.2	Memòria RTC.....	20
3.7	Configuració del Wi-Fi mitjançant BLE	21
3.7.1	Utilització de BLE	21
3.7.2	Seguretat en la transmissió BLE	21
3.8	Accés al medi.....	21
3.8.1	CSMA.....	22
3.8.2	TDMA.....	22

3.9	Càlcul del SF òptim	22
3.10	Càlcul Time on Air i Duty Cycle	23
4	RESULTATS	24
4.1	Utilització llibreria LoRa - RadioLib	24
4.1.1	Paràmetres utilitzats	24
4.1.2	Estudi funcionament CAD	25
4.1.3	Obtenció SNR i RSSI d'un missatge	29
4.2	Implementació LoRaWAN – Arduino LoRaWAN	30
4.3	Configuració Wi-Fi mitjançant BLE	32
4.4	Guardar dades en memòria no volàtil	35
4.4.1	Proves realitzades amb la llibreria Preferences	35
4.5	Estalvi d'energia i modes sleep	37
4.5.1	Mode light_sleep	37
4.5.2	Mode deep_sleep	38
4.6	Càlcul SF òptim	40
4.7	Disseny i implementació del protocol d'accés al medi	40
4.7.1	Escenari	40
4.7.2	Tipus de missatges	40
4.7.3	Càlcul ToA de cada missatge	43
4.7.4	Disseny protocol	44
4.7.5	Estats del node en memòria RTC.....	50
4.8	Proves de validació del protocol	51
4.8.1	Anàlisi temps entre missatges	51
4.8.2	Anàlisi temps en deep_sleep dels nodes	53
4.8.3	Anàlisi de funcionament amb varis nodes	53
5	DISCUSSIÓ	58
5.1	Validació funcionament protocol	58
5.1.1	Temps entre missatges.....	58
5.1.2	Temps en mode deep_sleep	58
5.1.3	Funcionament amb varis nodes.....	58
5.2	Comparació utilització Duty Cycle vs Listen Before Talk	58
5.3	Comparativa amb altres propostes realitzades	58
6	CONCLUSIONS I TREBALL FUTUR	60
6.1	Conclusions	60
6.2	Línies futures	60
6.2.1	Col·lisió missatges Request.....	60
6.2.2	ID dinàmic dels nodes.....	61
6.2.3	Un codi unificat per els nodes i gateway	61

6.2.4 Implementació del suport per varis gateways	61
6.2.5 Confirmació de la recepció de missatges	61
6.3 Problemàtica configuració Wi-Fi mitjançant BLE amb el node	61
6.4 Probabilitat d'error del CAD	62
6.5 Implementació servidor	62
6.6 Aplicació mòbil	62
6.7 Continuació del projecte.....	62
6.8 Estudi Sostenibilitat	62
6.8.1 Desenvolupament del treball final d'estudis	62
6.8.2 Execució del projecte.....	63
6.8.3 Riscos i limitacions.....	64
BIBLIOGRAFIA.....	65

Índex de figures

Figura 2.1 - Exemple de Coding Rate	3
Figura 2.2 - Trama LoRaWAN.....	4
Figura 2.3 - Esquema d'una xarxa LoRaWAN	5
Figura 2.4 - Il·lustració Duty Cycle	6
Figura 2.5 - Representació AFA.....	7
Figura 3.1 - Exemples de dispositius IoT basats en ESP32-S3	8
Figura 3.2 - SoC ESP32.....	9
Figura 3.3 - Instal·lació suport plaques ESP32	11
Figura 3.4 - Comprovació instal·lació ESP32.....	12
Figura 3.5 - Instal·lació llibreries.....	13
Figura 3.6 - Mode Actiu	18
Figura 3.7 - Mode Light Sleep	18
Figura 3.8 - Mode Deep Sleep	19
Figura 3.9 - Mode Hibernació	19
Figura 4.1 - Configuració paràmetres LoRa	24
Figura 4.2 - Utilització CAD	25
Figura 4.3 - Mesures temps detecció CAD	27
Figura 4.4 - Mesura temps reinici receptor LoRa	28
Figura 4.5 - Taxa d'error CAD	29
Figura 4.6 - Obtenció SNR i RSSI d'un missatge	30
Figura 4.7 - Missatge d'error LoRaWAN	31
Figura 4.8 - Imic_project_config.h	31
Figura 4.9 - Configuració TTN.....	32
Figura 4.10 - Connexió LoRaWAN.....	32
Figura 4.11 - Connexió BLE mitjançant nRF Connect	34
Figura 4.12 - Enviament de credencials Wi-Fi mitjançant BLE	35
Figura 4.13 - Emmagatzemant de credencials Wi-Fi amb Preferences i configuració Wi-Fi mitjançant BLE	36
Figura 4.14 - Reutilització dades NVP	36
Figura 4.15 - Comparativa temps esperat en <i>light_sleep</i> vs temps obtingut	38
Figura 4.16 - Comparativa temps esperat en <i>deep_sleep</i> vs temps obtingut	39
Figura 4.17 - Funció càlcul SF òptim.....	40
Figura 4.18 - Capçalera missatges	41

Figura 4.19 - Estructura missatge Beacon	41
Figura 4.20 - Estructura missatge Request.....	41
Figura 4.21 - Estructura missatge Schedule	42
Figura 4.22 - Estructura missatge Data	42
Figura 4.23 - Etapa de descobriment.....	45
Figura 4.24 - Etapa d'organització	46
Figura 4.25 - Etapa enviament de dades amb 2 nodes ubicats en els slots 0 i 1	47
Figura 4.26 - Diagrama de flux protocol.....	49
Figura 4.27 - Funcionament estats node	50
Figura 4.28 - Implementació protocol – Gateway i dos nodes	54
Figura 4.29 - Implementació protocol - Node 1.....	55
Figura 4.30 - Implementació protocol – Node 2	56
Figura 4.31 - Solució col·lisions missatges Request.....	57

Índex de taules

Taula 2.1 - Paràmetres editables LoRa	2
Taula 3.1 - Comparativa ESP32 - ESP32S3.....	9
Taula 3.2 - Llistat de mòduls compatibles amb la llibreria unofficial Heltec.....	13
Taula 3.3 - Llistat de mòduls compatibles amb la llibreria Arduino LoRa	14
Taula 3.4 - Llistat de mòduls compatibles amb la llibreria RadioLib	14
Taula 3.5 - Taula comparativa llibreries LoRa	15
Taula 3.6 - Paràmetres extra a tenir en compte en llibreries LoRa	16
Taula 3.7 - Llistat de mòduls compatibles amb la llibreria Beelan	16
Taula 3.8 - Llistat de mòduls compatibles amb la llibreria Arduino LoRaWAN.....	16
Taula 3.9 - Taula comparativa llibreries LoRaWAN	17
Taula 3.10 - Paràmetres extra a tenir en compte en llibreries LoRaWAN	17
Taula 3.11 - SF òptim en funció del SNR.....	23
Taula 3.12 - SF òptim en funció del RSSI.....	23
Taula 4.1 - Taula comparació mesures temps de detecció del CAD	28
Taula 4.2 - ToA Beacon	43
Taula 4.3 - ToA Request	43
Taula 4.4 - ToA Schedule.....	44
Taula 4.5 - ToA Data	44
Taula 4.6 - Resultats mesures temps entre missatges	51
Taula 4.7 - Comparació valors teòrics amb valors mesurats	52
Taula 4.8 - Comparació deep_sleep entre missatges Schedule i Data	53
Taula 4.9 - Comparació deep_sleep entre missatges Data i Beacon.....	53

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	Industrial, Scientific and Medical
LBT	Listen Before Talk
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
MHz	Mega Hertz
RSSI	Received Signal Strength Indicator
SF	Spreading Factor
SNR	Singal-to-Noise Ratio
SoC	System on Chip
ToA	Time on Air
TTN	The Things Network

1 Introducció

El motiu d'aquest treball és desenvolupar una eina per a dur a terme projectes de ciència ciutadana.

Els projectes de 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.

Concretament, en aquest treball es vol desenvolupar un dispositiu IoT (Internet of Things) que, atenent al seu ús, sigui de baix cost, fàcil d'utilitzar, apte per a funcionar en entorns amb possibilitat de connectivitat diferents, se li puguin acoblar diferents tipus de sensors i que pugui ser fàcilment programable, per adaptar el dispositiu al projecte en el que s'està emprant.

Aquest treball es centra en la part de connectivitat.

L'objectiu final, més enllà d'aquest treball, és el desenvolupament d'una solució sòlida que proporcioni diverses possibilitats de connectivitat, dintre d'un entorn robust, que permeti comunicar-se de diverses maneres amb, per ex., una estació meteorològica. Per tant, en aquest treball, com a primer pas, 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 objectius parcials dissenyar i desenvolupar una prova de concepte d'un protocol IoT que permeti la connectivitat entre diversos nodes. Aquesta necessitat ve donada per la utilització de LoRa (Long Range) en escenaris en què no hi hagi cobertura Wi-Fi. 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.

Atenent a l'ús que es vol donar a aquesta eina, la ciència ciutadana, i que es tracta del primer pas cap a una solució que tingui les característiques d'una solució comercial (robusta, de fàcil ús), el hardware que s'utilitzarà es de baix cost, totes les eines de software utilitzades són de codi lliure, compten amb una bona documentació i una gran comunitat al darrere, i el codi generat s'ha publicat per a facilitar el seu reutilització

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.

2 Protocols IoT

En aquest capítol, es defineixen els protocols IoT que s'han fet servir en aquest treball juntament amb els paràmetres que utilitzen.

2.1 LoRa

LoRa (Long Range) és una tecnologia de comunicacions sense fils que funciona mitjançant ràdio-freqüència, i que permet establir comunicació entre nodes amb un abast de fins a 4 km en zones urbanes amb una potència de transmissió molt baixa (+22 dBm). Actualment, LoRa és una tecnologia propietària i mantinguda per l'empresa Semtech®.

LoRa treballa a la capa física i utilitza una banda sense llicència ISM. Aquestes bandes són: 433 MHz per Àsia i Europa, 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. [1]

2.1.1 Paràmetres LoRa

A l'hora de fer servir LoRa es poden configurar alguns paràmetres per tal de garantir una òptima utilització. Aquests paràmetres, que s'explicaran en els següents subapartats, es mostren a la Taula 2.1:

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

2.1.1.1 Ampla de Banda (BW)

LoRa permet configurar una gran varietat d'amplades de banda. 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 necessària per transmetre la informació. Tot això, permet suportar el soroll molt millor i fer servir potències de transmissió molt baixes.

Concretament, LoRa pot ser configurat amb una ampla de banda que va 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.

2.1.1.2 Spread Factor (SF)

El Spread Factor (SF) és el factor d'eixamplament, 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, gràcies a un augment del SNR, i a costa d'obtenir un ToA (Time on Air) més elevat.

2.1.1.3 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. La taxa de codificació descriu la relació entre les dades reals (bits informació), i les dades de correcció d'errors afegits (bits correcció).

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

La Figura 2.1 representa una taxa de codificació 4/8. 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. 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.

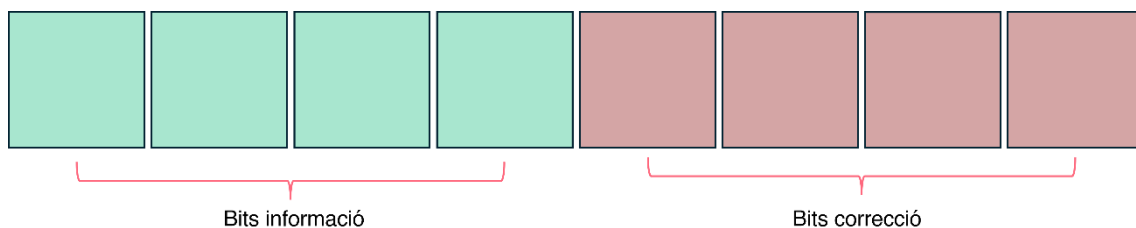


Figura 2.1 - Exemple de Coding Rate

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 utilitzant com a base LoRa, però afegint un protocol de capa d'enllaç, que defineixi missatges, amb les seves capçaleres, mètodes d'accés al medi o la comprovació de la rebuda de missatges.

2.2 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, a la capa d'enllaç, i té l'objectiu d'implementar millores sobre LoRa. En la figura 2.2, podem veure la jerarquia de les capes de LoRaWAN.

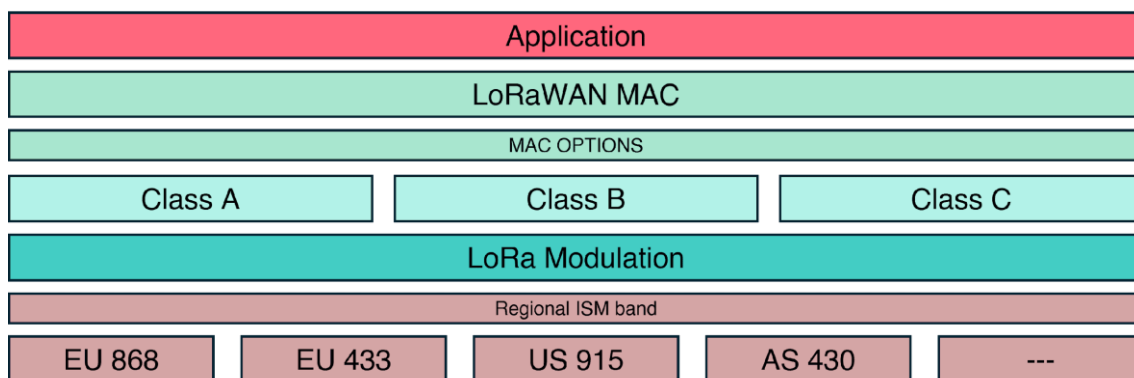


Figura 2.2 - Trama LoRaWAN

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.

2.2.1 Classes de nodes de LoRaWAN

2.2.1.1 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 habitualment estan alimentats per una bateria.

2.2.1.2 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.

2.2.1.3 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.

2.2.2 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). [2]

2.2.2.1 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. Aquest paràmetre s'ha canviat per JoinEUI en les especificacions de LoRaWAN més recents.
- AppKey: és una clau secreta de xifrat d' AES de 128 bits compartida entre el node i el l'aplicació. Es fa servir per a determinar les claus de la sessió.

2.2.2.2 Activation-By-Personalization (ABP)

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

- DevAddress: 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 (network server). 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'aplicació, 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 network server.
2. El network server valida que les dades corresponguin a la sessió.
3. Si la sessió és correcta, les dades es processen.

Aquest tipus permet una connexió més ràpida a la xarxa, ja que no requereix l'intercanvi de les claus de sessió, però no aplica mecanismes de seguretat tan bons com l'OTAA.

En la següent figura, es pot visualitzar un esquema d'una xarxa LoRaWAN:

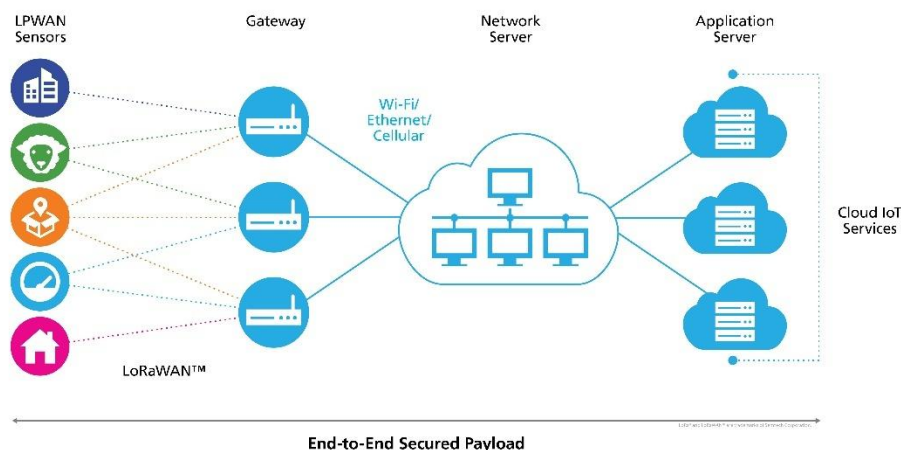


Figura 2.3 - Esquema d'una xarxa LoRaWAN

2.3 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 o la suma dels ToAs poden 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). [3]

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 ToA, el Duty Cycle i el Polite Spectrum Access.

El ToA és el temps que triga un missatge a ser transmès des del transmissor. El ToA depèn de la configuració de LoRa aplicada. Alguns dels 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 ToA.
- Mida Payload: Quants més bits d'informació s'enviïn, més temps de ToA hi haurà.

El Duty Cycle és el temps entre transmissions per a un determinat node. Defineix la relació de temps entre el ToA i el temps entre missatges. En la següent figura, es pot visualitzar una representació del Duty Cycle.

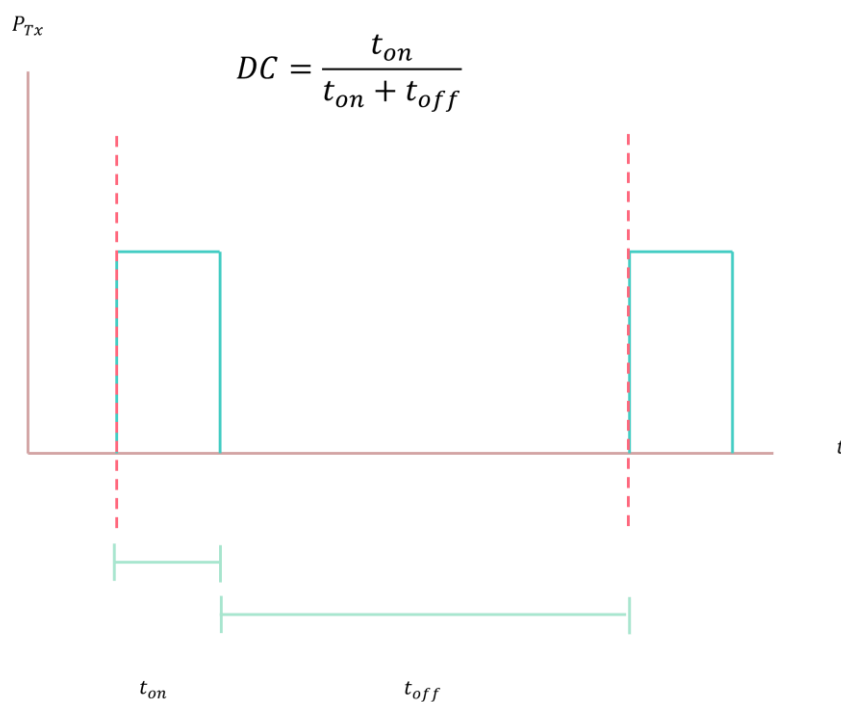


Figura 2.4 - Il·lustració Duty Cycle

El Polite Spectrum Access es una tècnica dissenyada per optimitzar la utilització de l'espectre en xarxes LoRa. Aquesta tècnica permet utilitzar dues funcions diferents:

- Listen Before Talk (LBT): El node transmissor abans d'enviar un missatge, ha d'escollar el canal durant almenys 160 μ s, si el soroll detectat en el canal no supera els -82 dBm, es considera que el canal està lliure.
- Adaptive Frequency Agility (AFA): El node fa un escaneig dels canals disponibles, i "marca" els canals que tenen més soroll. Aquests canals, queden fora el llistat de canals disponibles per a que el node pugui enviar les dades. D'aquesta manera, el node sempre envia per els canals amb menys soroll. [4]

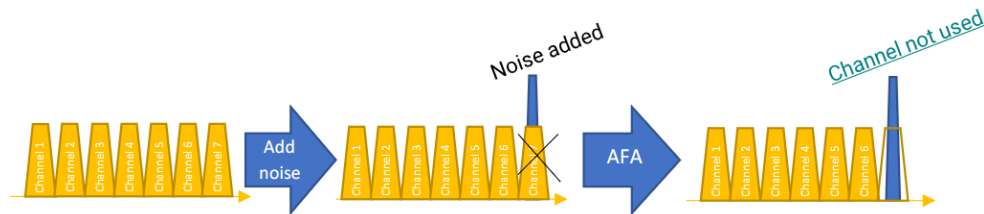


Figura 2.5 - Representació AFA

Per altra banda, en l'annex B de la normativa vigent definida per l'ETSI, podem observar com, quan s'utilitza el Polite Spectrum Access, no es requereix complir el 1% del Duty Cycle per comunicacions LoRa.

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 [3]

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. [5]

3 Mètodes i material utilitzat

3.1 Software i hardware utilitzat

3.1.1 Hardware utilitzat

L'eina principal que s'ha utilitzat en el desenvolupament d'aquest treball, ha estat el microcontrolador ESP32S3, en concret integrat en els dispositius IoT Heltec Wi-Fi LoRa v3 i XIAO ESP32S3, els quals ja porten integrat també un transceptor LoRa del model SX1262 de Semtech ®. Aquest tipus de transceptor, està dissenyat per dispositius de baix consum, com el ESP32, ja que tenen un consum de 4,2 mA en estat actiu i poden transmetre amb potències de fins a +22 dBm. [6]

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.

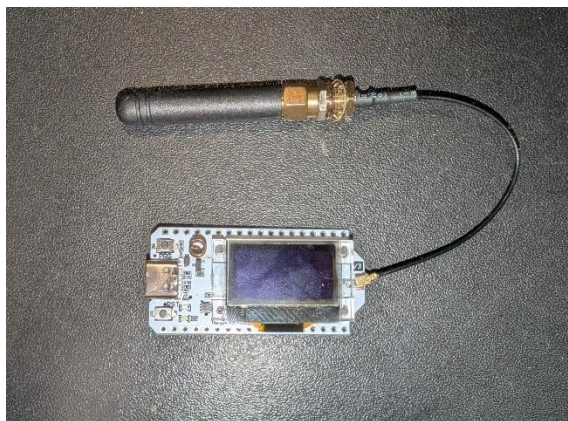


Figura 3.1.a - Heltec Wi-Fi LoRa V3

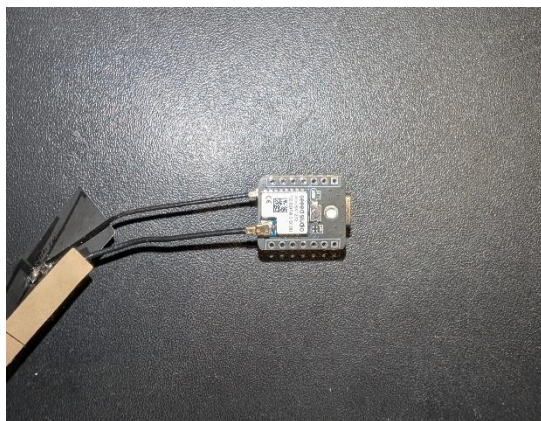


Figura 3.1.a - XIAO ESP32S3

Figura 3.1 - Exemples de dispositius IoT basats en ESP32-S3

3.1.1.1 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.

Els ESP32 són una sèrie de SoC (System on Chip). En la figura 3.2, podem observar com són els blocs definits en el SoC del ESP32.

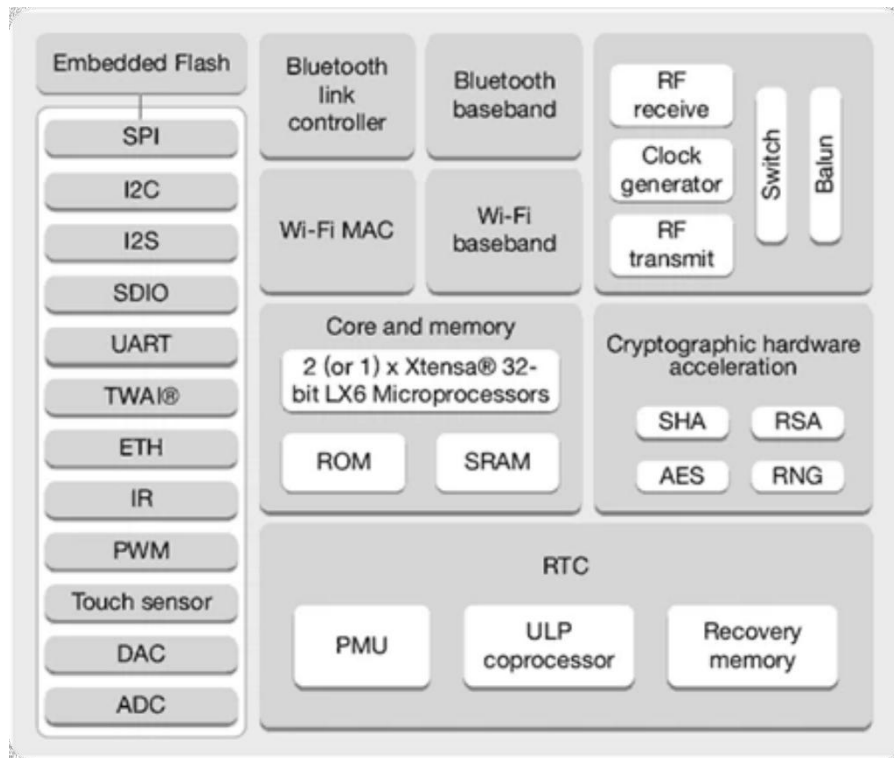


Figura 3.2 - SoC ESP32

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.

3.1.1.2 ESP32S3

La sèrie de microcontroladors ESP32S3 són una versió millorada de l'original ESP32. Respecte al ESP32, la versió S3 millora el processador incorporat i duplica l'espai de memòria Flash fins a 8 MB. Per altra banda, els ESP32S3 augmenten el suport de Bluetooth fins a la versió 5.0.

La major diferència entre ambdues plaques és el suport per acceleració d'IA que incorpora la versió S3, proporcionant acceleració per computació de xarxes neuronals i processament del senyal. [7]

En la següent taula es mostra una comparativa entre les dues versions dels microcontroladors:

	ESP32	ESP32S3
Processador	Xtensa dual-core 32bit LX6	Xtensa dual-core 32bit LX7
Memòria ROM	448 KB	384 KB
Memòria Flash	Fins a 4 MB	Fins a 8 MB
Wi-Fi	802.11 b/g/n, 2.4 GHz	802.11 b/g/n, 2.4 GHz
Bluetooth	BLE 4.2	BLE 5.0
Consum en deep_sleep	100 μ A	7 μ A

Taula 3.1 - Comparativa ESP32 - ESP32S3

3.1.2 Software Utilitzat

3.1.2.1 Arduino

Pel que fa al software, s'ha decidit fer servir el llenguatge de programació d'Arduino juntament amb l'IDE Arduino IDE en la versió 2.3.4, 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 futur del projecte.

3.1.2.2 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'ESP32, i a partir dels resultats obtinguts, posteriorment extreure conclusions.

3.1.2.3 Llibreries utilitzades

A continuació s'enumeren les llibreries utilitzades per el desenvolupament del projecte:

- Arduino:
 - o RadioLib: llibreria pel suport de LoRa (Més endavant, es justificarà la utilització d'aquesta llibreria)
 - o ArduinoBLE: llibreria pel suport de BLE
 - o Preferences: llibreria per guardar dades en memòria no volàtil
 - o WiFi: llibreria pel suport WiFi
- Python:
 - o Pandas: llibreria per la creació dels datasets
 - o Matplotlib: llibreria per la creació de diagrames
 - o Os: llibreria per accedir al sistema de fitxers de l'ordinador
 - o Serial: llibreria per llegir el port sèrie
 - o Csv: llibreria per crear arxius .CSV
 - o Time: llibreria per mesurar el temps
 - o Thread: llibreria per executar dos tasques simultàniament

3.1.2.4 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. Es pot trobar l'enllaç del repositori en el següent enllaç:

<https://github.com/paugarcia32/TFG>

3.1.2.5 nRF Connect

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

3.2 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.

3.2.1 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](#).

3.2.2 Instal·lació de la 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

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

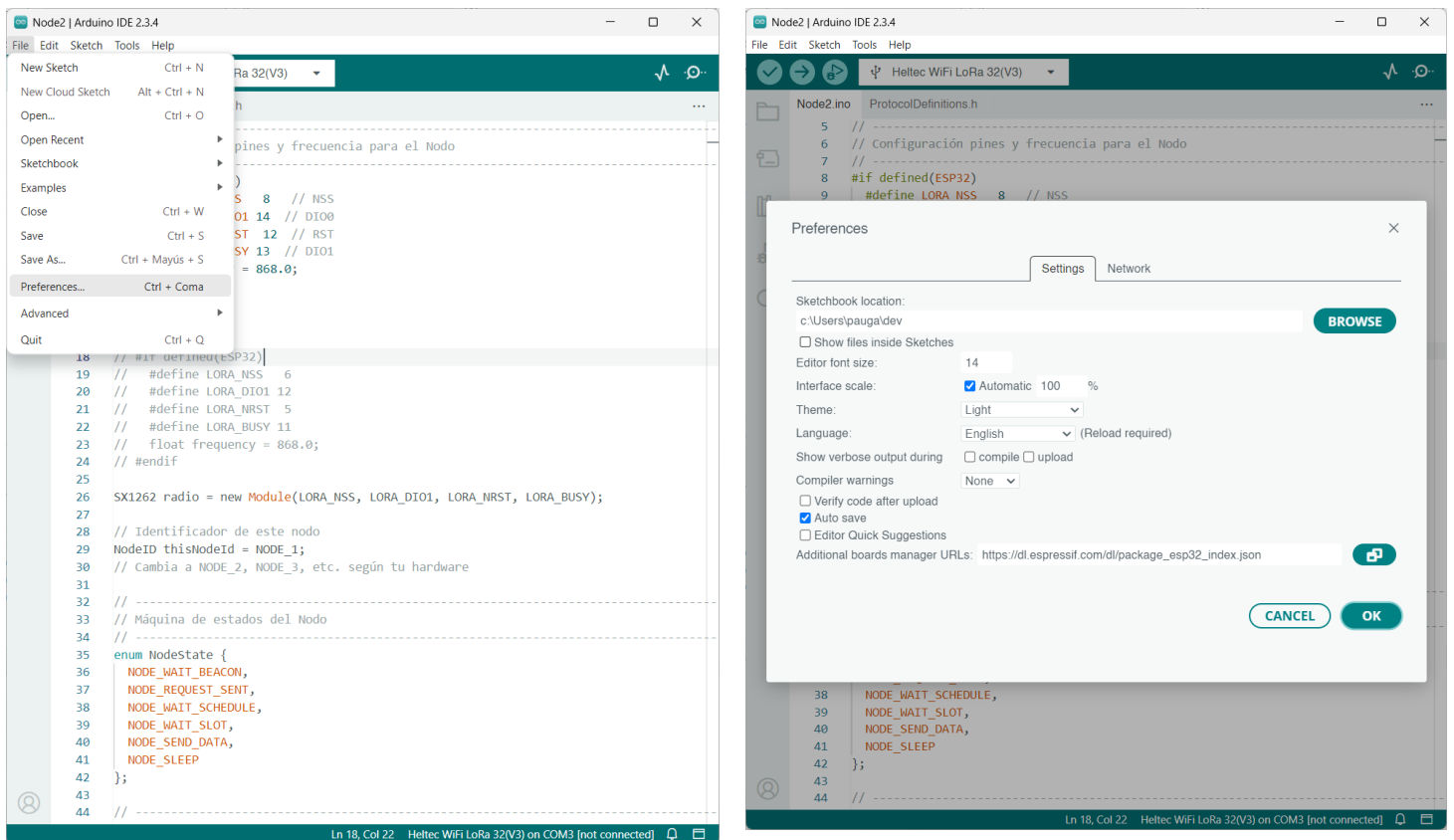


Figura 3.3 - Instal·lació suport plaques ESP32

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.

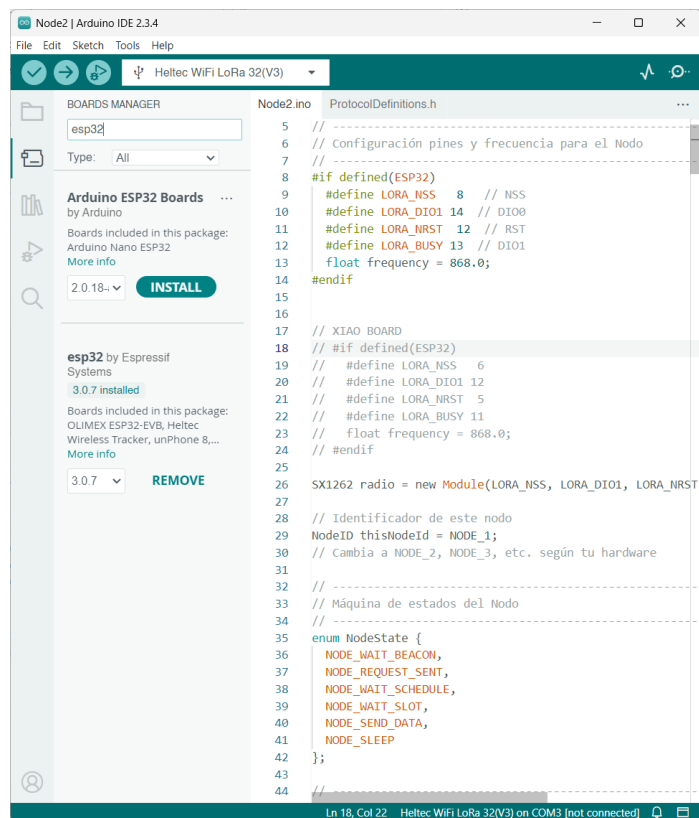


Figura 3.4 - Comprovació instal·lació ESP32

3.2.3 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.

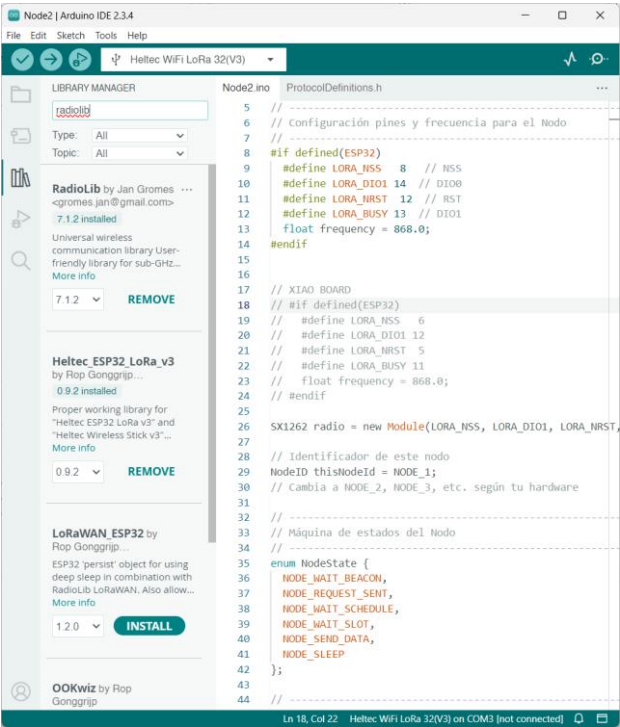


Figura 3.5 - Instal·lació llibreries

3.3 Estudi llibreries LoRa

Donat que Arduino és un llenguatge àmpliament 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 aquest estudi, i en el capítol de ‘Resultats’ es justificarà l’elecció final.

3.3.1 Unofficial Heltec

La llibreria Unofficial Heltec és una llibreria que implementa tot tipus de funcions relacionades amb les plaques del fabricant 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 3.2 - Llistat de mòduls compatibles amb la llibreria unofficial Heltec

3.3.2 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 3.3 - Llistat de mòduls compatibles amb la llibreria Arduino LoRa

3.3.3 RadioLib

Aquesta llibreria permet als usuaris integrar tot tipus de 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 3.4 - Llistat de mòduls compatibles amb la llibreria RadioLib

3.3.4 Comparativa entre llibreries LoRa

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

En les següents taules, quan un camp està marcat amb una 'X' indica que la llibreria permet configurar el paràmetre en qüestió.

3.3.4.1 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
P_{tx}	X	X	X
Tipus de Bus	SPI	SPI	SPI
CAD	X	X	X
Capçalera implícita / explícita		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 3.5 - Taula comparativa llibreries LoRa

3.3.4.2 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.

	Unnofficial Heltec	Arduino LoRa
Llibreria actualitzada	X	X
Bona documentació	X	X

Taula 3.6 - Paràmetres extra a tenir en compte en llibreries LoRa

3.4 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.

3.4.1 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 3.7 - Llistat de mòduls compatibles amb la llibreria Beelan

3.4.2 Arduino LoRaWAN

La llibreria MCCI Arduino LoRaWAN proporciona una forma estructurada d'utilitzar la llibreria Arduino-Imic 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 3.8 - Llistat de mòduls compatibles amb la llibreria Arduino LoRaWAN

3.4.3 Comparativa entre llibreries LoRaWAN

Comparativa de característiques suportades.

	Beelan	Arduino LoRaWAN
--	--------	-----------------

Authentication Keys	X	X
Activació OTAA	X	X
Mode Sleep	X	
Classes LoRaWAN	A i C	A

Taula 3.9 - Taula comparativa llibreries LoRaWAN

3.4.3.1 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 3.10 - Paràmetres extra a tenir en compte en llibreries LoRaWAN

3.5 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 diverses possibilitats de configuració.

Per l'eventual utilització d'una bateria connectada als ESP32, és crucial entendre el funcionament de tots els modes que ofereix l'ESP32 per optimitzar el consum d'aquests nodes.

3.5.1 Modes d'energia

3.5.1.1 Mode Actiu

Aquest mode és el mode que s'utilitza per defecte en els ESP32. 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.

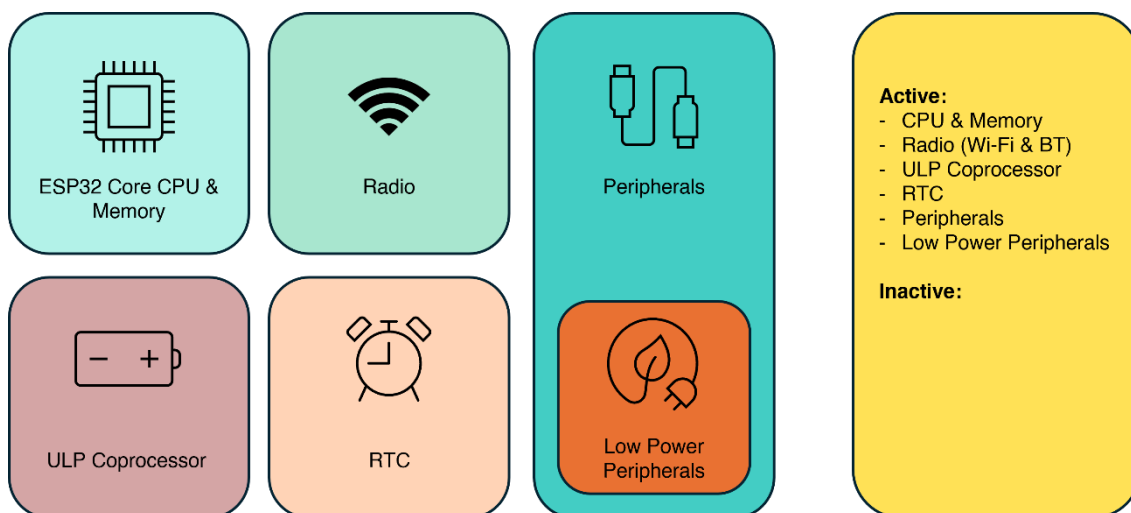


Figura 3.6 - Mode Actiu

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

3.5.1.2 Mode Light Sleep

Aquest mode és molt utilitzat quan no es requereix de comunicacions ràdio quan el dispositiu es posa a dormir. [8]

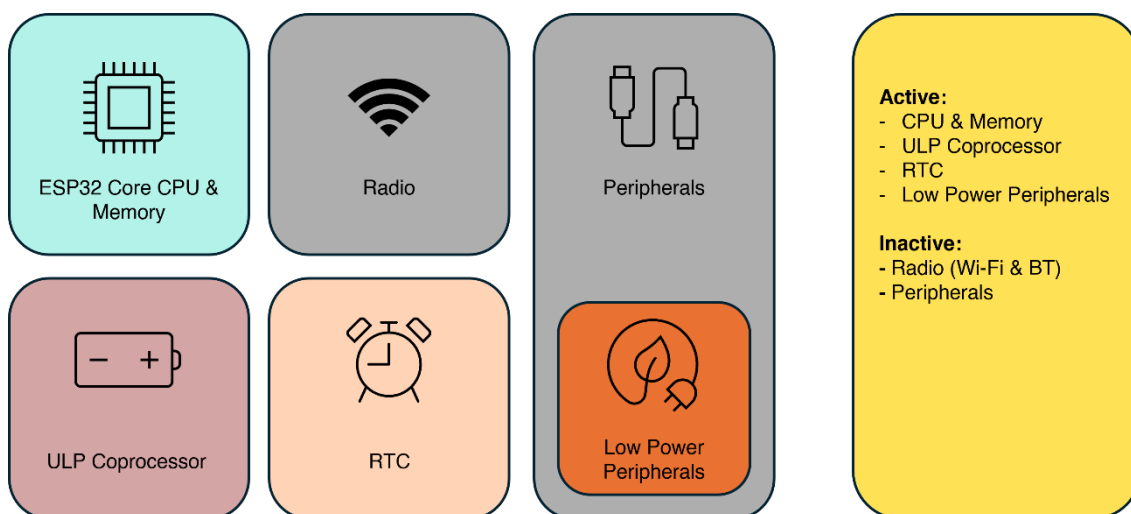


Figura 3.7 - Mode Light Sleep

Aquest mode deshabilita els mòduls radio que porta l'ESP32 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 ESP32 està en aquest mode, té un consum de 240 μ A.

3.5.1.3 Mode Deep Sleep

En comparació als modes explicats anteriorment, el Mode Deep Sleep deshabilita el nucli principal del ESP32, el qual inclou el processador principal i la memòria, obtenint així un consum mínim. [8]

Com el nucli del ESP32 està deshabilitat, el co-processador ULP que té el ESP32 es manté actiu a l'espera d'un esdeveniment extern o a un comptador per a despertar-se. En aquest mode d'energia, el ESP32 té un consum de 7 μ A.

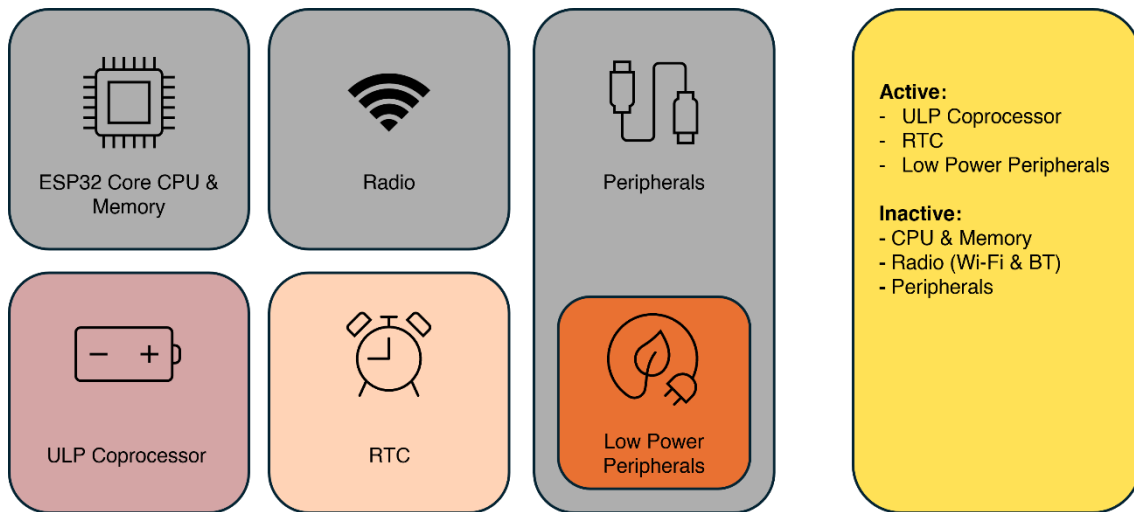


Figura 3.8 - Mode Deep Sleep

Un factor molt important a entendre per utilitzar correctament aquest mode d'estalvi d'energia, està relacionat amb la pèrdua de memòria del estat de l'ESP32 abans de posar-se a dormir. Quan l'ESP32 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, l'ESP32 deshabilita la memòria i no és capaç de recordar en quin punt del codi s'havia anat a dormir. Per aconseguir que l'ESP recordi per on continuar un cop s'ha despertat, es pot fer servir la memòria RTC. En l'apartat 3.6.2 s'explica en més detall aquest tipus de memòria.

3.5.1.4 Mode hibernació

Aquest mode és el mode amb menys consum. S'aconsegueix deshabilitant quasi tots els circuits interns, excepte l'RTC. En aquest mode, l'ESP32 únicament consumeix 5 μ A. [8]

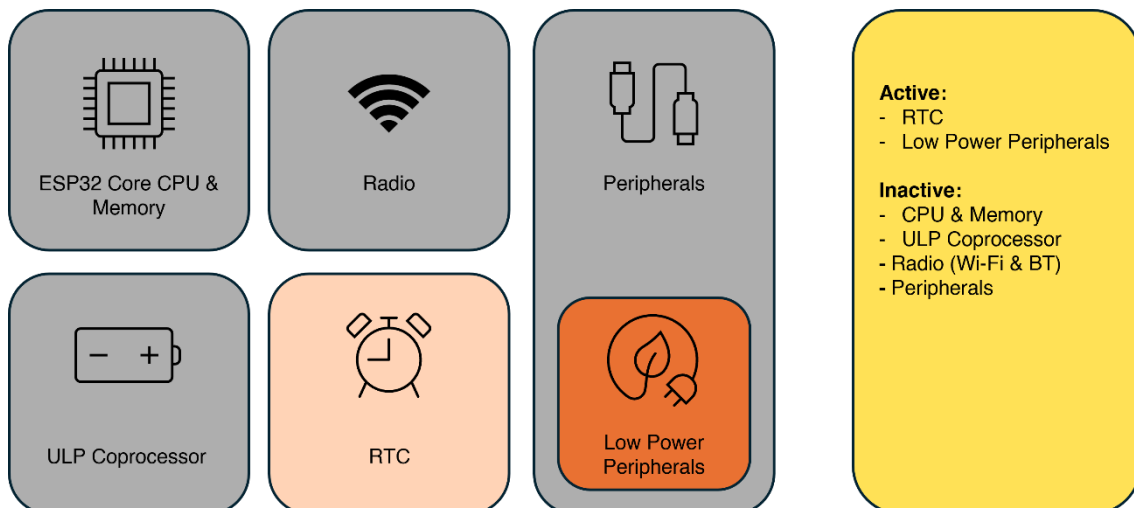


Figura 3.9 - Mode Hibernació

3.6 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 després d'una pèrdua de corrent o un reinici.

3.6.1 Memòria no-volàtil

Els ESP32 permeten guardar dades en memòria flash utilitzant dues 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'Espressif es recomana utilitzar la llibreria de Preferences per davant de EEPROM. Per tant, ens centrarem en l'explicació i utilització de Preferences [9].

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 ESP32 (NVP) per poder emmagatzemar dades. Les dades guardades en la memòria NVP, es conserven davant possibles reinicis 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.

3.6.2 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 específiques, mentre NVP pot servir per guardar credencials.

En relació amb els modes d'estalvi d'energia, com ja s'ha esmentat anteriorment, la memòria RTC no s'elimina després d'utilitzar qualsevol dels 3 nodes. Més endavant, s'ensenyarà com s'ha utilitzat la memòria RTC per guardar en quin estat estava l'ESP32, i continuar en el punt on estava abans de posar-se a dormir.

3.6.2.1 Similituds i diferències entre ambdós modes

En ambdós modes es permet guardar les dades mentre l'ESP32 està en mode de son profund o si l'ESP32 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 utilitza per emmagatzemar memòria que ha de ser accedida pel codi immediatament després d'un reinici.

3.7 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 que operin a la banda de 2,4GHz.

3.7.1 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 mitjançant 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 4.0 (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 ESP32, s'ha plantejat la possibilitat de guardar aquestes dades en memòria NVP. En l'apartat de resultats, es pot visualitzar l'execució d'aquesta configuració.

3.7.2 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 el xifrat per bloc que proporciona AES-128, assegurant així la confidencialitat de les dades. Aquesta encriptació es realitzada al payload del missatge un cop s'ha establert una connexió segura entre 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.

3.8 Accés al medi

Per aquells escenaris en què no es disposi de cobertura d'una xarxa LoRaWAN, s'ha plantejat la possibilitat de què un dispositiu amb connexió Wi-Fi a Internet, actuï com a gateway LoRa. 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.

3.8.1 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 eventuais 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.

3.8.2 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 canvi, si es col·loca el Polite Spectrum Access definit per l'ETSI, s'haurà d'escoltar el canal abans de d'enviar.

En termes de consum d'energia, TDMA és un mecanisme molt eficient: com el node ja sap quan ha d'enviar les dades, pot posar-se a dormir, i despertar-se només per enviar-les.

3.9 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 l'SF òptim en funció, s'ha utilitzat com a referència els valors trobats en un estudi, el qual relaciona un SF òptim amb el SNR i amb el RSSI. Aquestes relacions, es poden trobar en les següents taules. [10]

Posteriorment, es definirà una manera de poder complir amb les dues restriccions.

SF	SNR Limit (dB)
7	-7,5
8	-10
9	-12,5

10	-15
11	-17,5
12	-20

Taula 3.11 - SF òptim en funció del SNR

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

Taula 3.12 - SF òptim en funció del RSSI

3.10 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 ToA i del DC, s'ha fet servir la calculadora que proporciona Semtech [11], on es poden definir tots els valors utilitzats en la transmissió, així com el transceptor que s'ha fet servir.

4 Resultats

4.1 Utilització 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. 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, Unnofficial Heltec no ofereix cap avantatge respecte a RadioLib ja que totes les funcions que ofereix Unnofficial Heltec també les ofereix RadioLib.

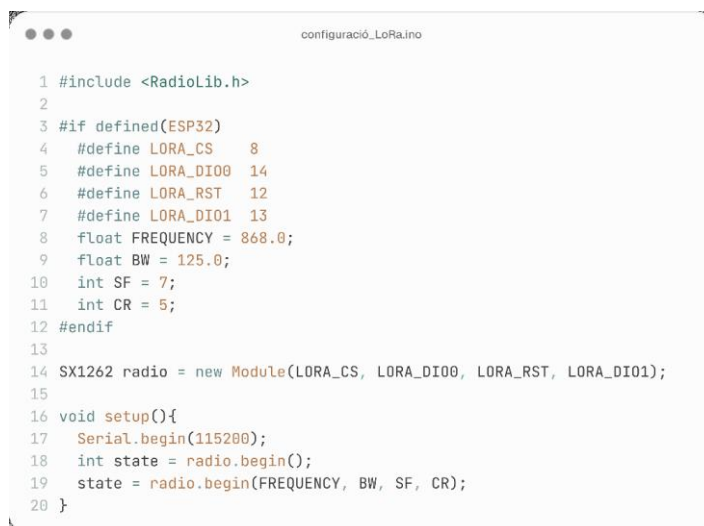
La implementació d'aquesta llibreria ha estat molt satisfactòria, ja que ha permès treballar amb la tecnologia LoRa sense problemes. Per altra banda, la documentació oficial de la llibreria és molt extensa, explica detalladament totes les funcions que permet, a més d'un gran nombre d'exemples per entendre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

4.1.1 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:



```
1 #include <RadioLib.h>
2
3 #if defined(ESP32)
4   #define LORA_CS    8
5   #define LORA_DIO0  14
6   #define LORA_RST   12
7   #define LORA_DIO1  13
8   float FREQUENCY = 868.0;
9   float BW = 125.0;
10  int SF = 7;
11  int CR = 5;
12 #endif
13
14 SX1262 radio = new Module(LORA_CS, LORA_DIO0, LORA_RST, LORA_DIO1);
15
16 void setup(){
17   Serial.begin(115200);
18   int state = radio.begin();
19   state = radio.begin(FREQUENCY, BW, SF, CR);
20 }
```

Figura 4.1 - Configuració paràmetres LoRa

Es pot trobar un exemple més extens de la configuració dels paràmetres LoRa en el següent enllaç:

https://github.com/jgromes/RadioLib/blob/master/examples/SX126x/SX126x_Settings/SX126x_Settings.ino

4.1.2 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()`.

El correcte funcionament del CAD es molt important degut a que es vol utilitzar CSMA com a control d'accés al medi. Si el CAD no funciona correctament, tampoc es podrà fer servir el LBT, i s'haurà de respectar el DC en comptes del Polite Spectrum Access.

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.



```
1 #include <RadioLib.h>
2 SX1262 radio = new Module(10, 2, 3, 9);
3
4 volatile bool scanFlag = false;
5 #if defined(ESP8266) || defined(ESP32)
6   ICACHE_RAM_ATTR
7 #endif
8 void setFlag(void)
9   scanFlag = true;
10 }
11
12 void setup() {
13   Serial.begin(115200);
14   Serial.print(F("[SX1262] Initializing ... "));
15   int state = radio.begin();
16   if (state == RADIOLIB_ERR_NONE) {
17     Serial.println(F("success!"));
18   } else {
19     Serial.print(F("failed, code "));
20     Serial.println(state);
21     while (true) { delay(10); }
22   }
23   radio.setDio1Action(setFlag);
24
25   Serial.print(F("[SX1262] Starting scan for LoRa preamble ... "));
26   state = radio.startChannelScan();
27   if (state == RADIOLIB_ERR_NONE) {
28     Serial.println(F("success!"));
29   } else {
30     Serial.print(F("failed, code "));
31     Serial.println(state);
32   }
33 }
34
35 void loop() {
36   if(scanFlag)
37     scanFlag = false;
38     int state = radio.getChannelScanResult();
39     if (state == RADIOLIB_LORA_DETECTED) {
40       Serial.println(F("[SX1262] Packet detected!"));
41     } else if (state == RADIOLIB_CHANNEL_FREE) {
42       Serial.println(F("[SX1262] Channel is free!"));
43     } else {
44       Serial.print(F("[SX1262] Failed, code "));
45       Serial.println(state);
46     }
47     Serial.print(F("[SX1262] Starting scan for LoRa preamble ... "));
48     state = radio.startChannelScan();
49     if (state == RADIOLIB_ERR_NONE) {
50       Serial.println(F("success!"));
51     } else {
52       Serial.print(F("failed, code "));
53       Serial.println(state);
54     }
55   }
56 }
```

Figura 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>

En totes les proves realitzades per el CAD, s'ha fet servir el dispositiu Heltec WiFi LoRa v3.

4.1.2.1 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 port 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'encarrega de guardar quin node ha enviat un missatge pel 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 en cada cas pot trigar en detectar els missatges el transceptor utilitzat mitjançant les funcions del CAD habilitades per la llibreria.

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

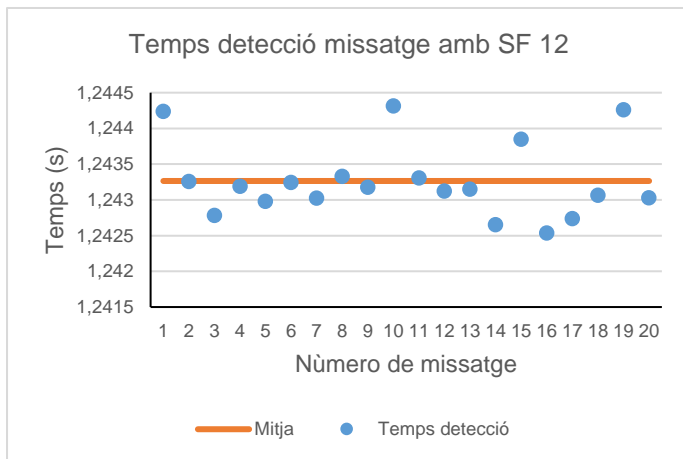


Figura 4.3.a – Mesura temps detecció CAD amb SF 12

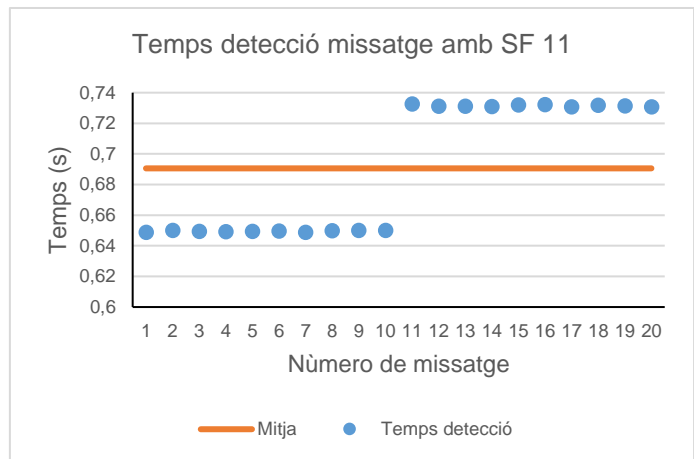


Figura 4.3.b – Mesura temps detecció CAD amb SF 11

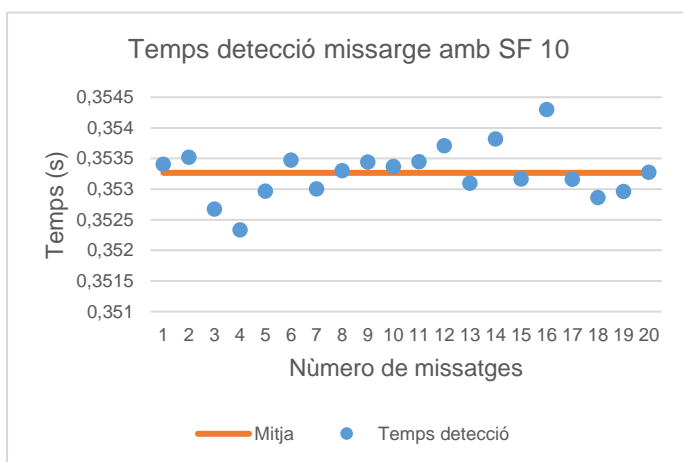


Figura 4.3.c – Mesura temps detecció CAD amb SF 10

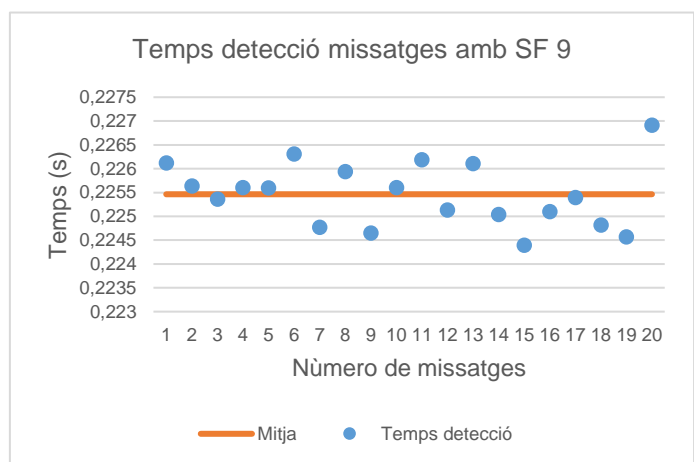


Figura 4.3.d – Mesura temps detecció CAD amb SF 9

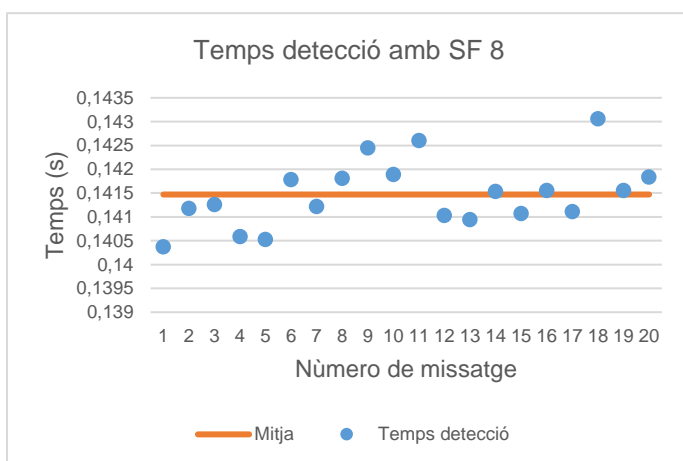


Figura 4.3.e – Mesura temps detecció CAD amb SF 8

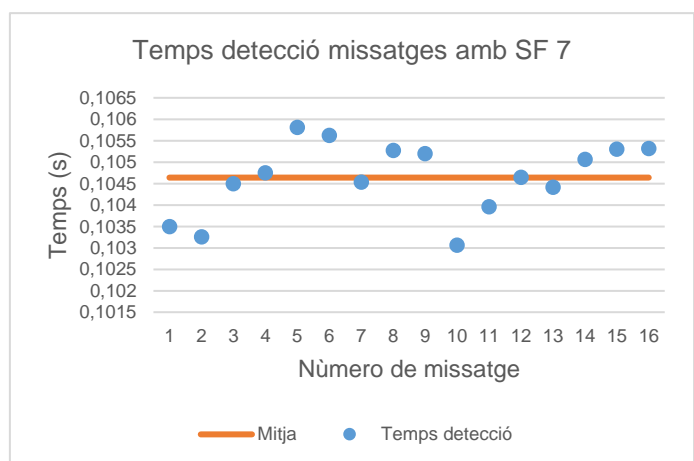


Figura 4.3.f – Mesura temps detecció CAD amb SF 7

Figura 4.3 - 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)	ToA 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

Taula 4.1 - Taula comparació mesures temps de detecció del CAD

4.1.2.2 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.

En la següent figura, es pot observar com el temps de reinici del receptor ha estat 8,2 ms de mitja. Per tant, es pot interpretar que el fet que el receptor no sigui capaç de detectar tots els missatges, no es degut a un temps de reinici elevat.

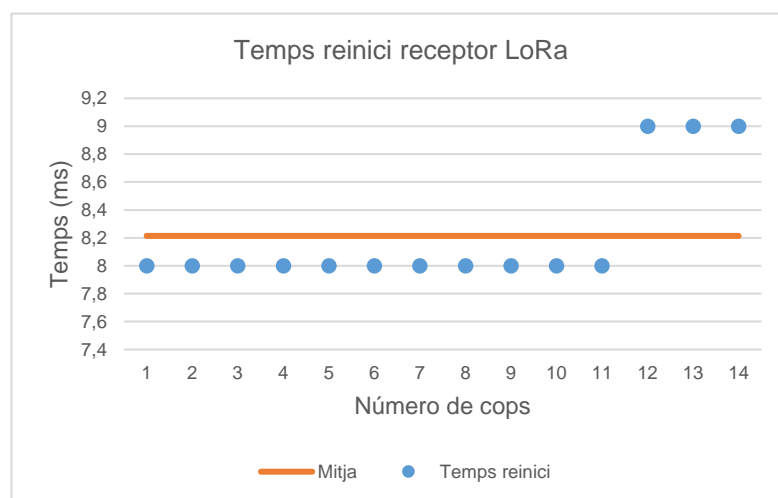


Figura 4.4 - Mesura temps reinici receptor LoRa

4.1.2.3 Taxa d'error del CAD

La última mesura realitzada, degut al fet de què el receptor només triga 9 ms en reiniciar-se, ha estat veure quina taxa d'error té el CAD. Aquesta taxa d'error es defineix de la següent manera:

$$\text{Taxa Error CAD} = \frac{n^{\circ} \text{ missatges detectats}}{n^{\circ} \text{ total missatges transmesos}}$$

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:

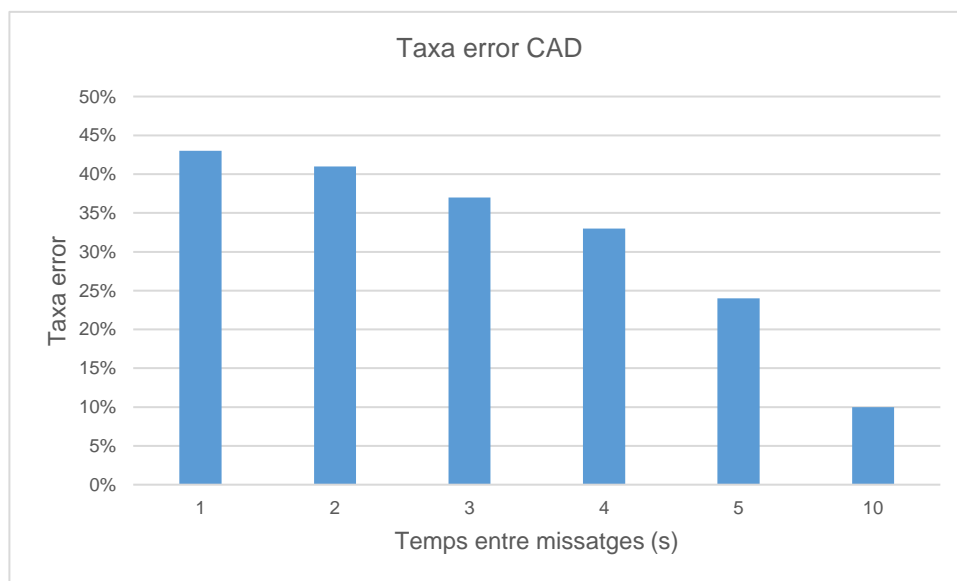



Figura 4.5 - Taxa d'error CAD

4.1.3 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.

Mitjançant l'obtenció del SNR i del RSSI, es poden fer servir les relacions amb l'SF que s'han explicat en l'apartat 3.9 per trobar l'SF òptim de cada missatge. Posteriorment, en l'apartat 4.6, es podrà veure aquesta implementació.

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



```
1 #include <RadioLib.h>
2 SX1262 radio = new Module(10, 2, 3, 9);
3
4 float rssi = radio.getRSSI();
5 float snr = radio.getSNR();
6
7 Serial.print(F("RSSI: "));
8 Serial.print(rssi);
9 Serial.print(F(" dBm"));
10
11 Serial.print(F("SNR: "));
12 Serial.print(snr);
13 Serial.print(F(" dB"));
```

Figura 4.6 - Obtenció SNR i RSSI d'un missatge

Per veure un exemple amb més detall, es pot visitar l'exemple proporcionar per RadioLib en el següent enllaç:

https://github.com/jgromes/RadioLib/blob/master/examples/SX126x/SX126x_Channel_Activity_Detection_Interrupt/SX126x_Channel_Activity_Detection_Interrupt.ino

4.2 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 ha estat més difícil fer servir LoRaWAN.

El principal problema, ha estat la poca compatibilitat que tenen les plaques Heltec amb LoRaWAN, ja que una gran part del desenvolupament d'aquest treball s'ha fet amb les plaques Heltec.

Tot i utilitzant el codi d'exemple que proporcionava la llibreria, amb les plaques Heltec no s'ha aconseguit establir connexió amb el gateway. En totes les proves realitzades per fer servir LoRaWAN, tant els nodes com el gateway han estat configurats per fer servir el mode d'activació OTAA juntament amb la xarxa de TTN (The Things Network). Per altre banda, la versió de LoRaWAN usada ha estat la LoRaWAN Specification 1.0.2.

El codi que s'ha fet servir es pot trobar en el següent enllaç:

<https://github.com/mcci-catena/arduino-lmic/blob/master/examples/ttn-otaa/ttn-otaa.ino>

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

```

ESP-ROM:esp32s3-20210327
Build:Mar 27 2021
rst:0x1 (POWERON),boot:0x9 (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:1
load:0x3fce3818,len:0x109c
load:0x403c9700,len:0x4
load:0x403c9704,len:0xb50
load:0x403cc700,len:0x2fe4
entry 0x403c98ac
Starting
Starting os_init
FAILURE
c:\Users\pauga\dev\libraries\MCCI_LoRaWAN_LMIC_library\src\hal\hal.cpp:35
Guru Meditation Error: Core 1 panic'ed (Interrupt wdt timeout on CPU1).

Core 1 register dump:
PC      : 0x4200258c  PS      : 0x00060a34  A0      : 0x820025cd  A1      : 0x3fcebda0
A2      : 0x3c030431  A3      : 0x00000023  A4      : 0x3fc94a60  A5      : 0x0000ff00
A6      : 0x00ff0000  A7      : 0xff000000  A8      : 0x8200258c  A9      : 0x3fcebda0
A10     : 0x3fc94a60  A11     : 0x00000002  A12     : 0x0000000a  A13     : 0xffffffff
A14     : 0x00000000  A15     : 0x00000000  SAR     : 0x0000001d  EXCCAUSE: 0x00000006
EXCVADDR: 0x00000000  LBEG    : 0x400556d5  LEND    : 0x400556e5  LCOUNT   : 0xffffffff

```

Figura 4.7 - Missatge d'error LoRaWAN

Una hipòtesi per la qual es creu que no ha funcionat correctament, és per una configuració errònia dels pins, tot 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-arduinosp32/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.

```

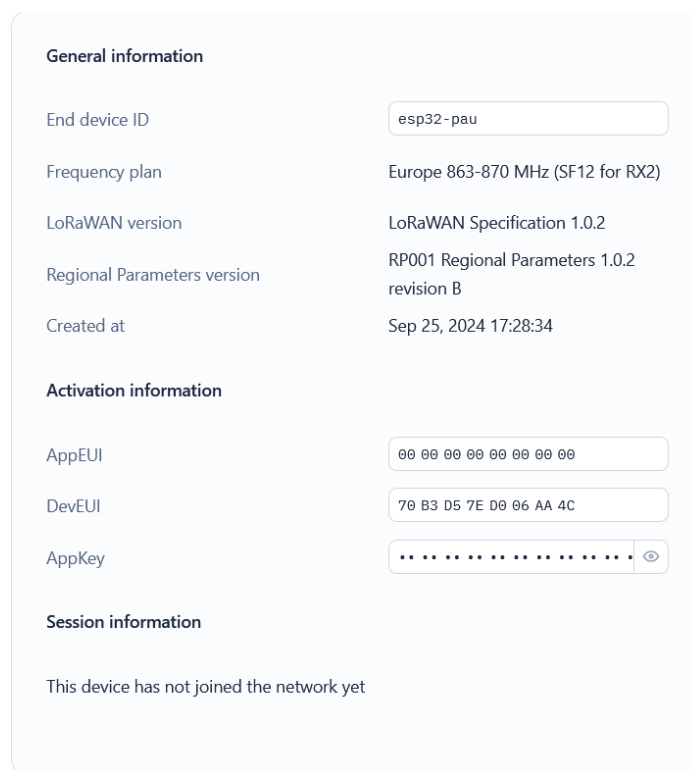
lmic_project_config.h

1 // project-specific definitions
2 #define CFG_eu868 1
3 //#define CFG_us915 1
4 //#define CFG_aus915 1
5 //#define CFG_as923 1
6 // #define LMIC_COUNTRY_CODE
7 //#define CFG_kr920 1
8 //#define CFG_in866 1
9 //#define CFG_sx1276_radio 1
10 //#define CFG_sx1261_radio 1
11 #define CFG_sx1262_radio 1
12 #define ARDUINO_heltec_wifi_lora_32_V3
13 //#define LMIC_USE_INTERRUPTS
14 #define DISABLE_PING

```

Figura 4.8 - 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.



General information

End device ID	esp32 - pau
Frequency plan	Europe 863-870 MHz (SF12 for RX2)
LoRaWAN version	LoRaWAN Specification 1.0.2
Regional Parameters version	RP001 Regional Parameters 1.0.2 revision B
Created at	Sep 25, 2024 17:28:34

Activation information

AppEUI	00 00 00 00 00 00 00 00
DevEUI	70 B3 D5 7E D0 06 AA 4C
AppKey

Session information

This device has not joined the network yet

Figura 4.9 - Configuració TTN

En canvi, amb els models de XIAO, si s'ha aconseguit establir una connexió amb TTN utilitzant el mateix codi d'exemple que proporciona la llibreria. En la següent figura, es pot visualitzar com l'ESP indica que ha estat capaç d'unir-se a la xarxa LoRaWAN.

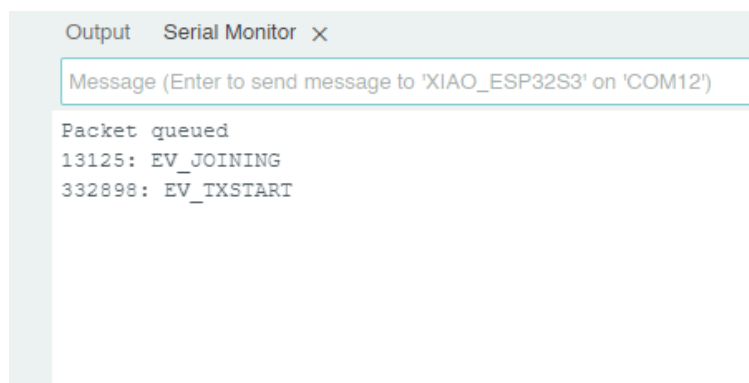


Figura 4.10 - Connexió LoRaWAN

4.3 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'ensenyà en l'apartat 4.4, justament amb el codi de com guardar dades en les Preferences.

Connexió amb l'ESP32

Per poder realitzar la connexió amb l'ESP32, s'ha fet servir l'aplicació nRF Connect. Un cop el servei BLE s'anuncia, es pot connectar el telèfon mòbil a l'ESP32. En les següents figures, es pot observar com es pot realitzar aquesta connexió.

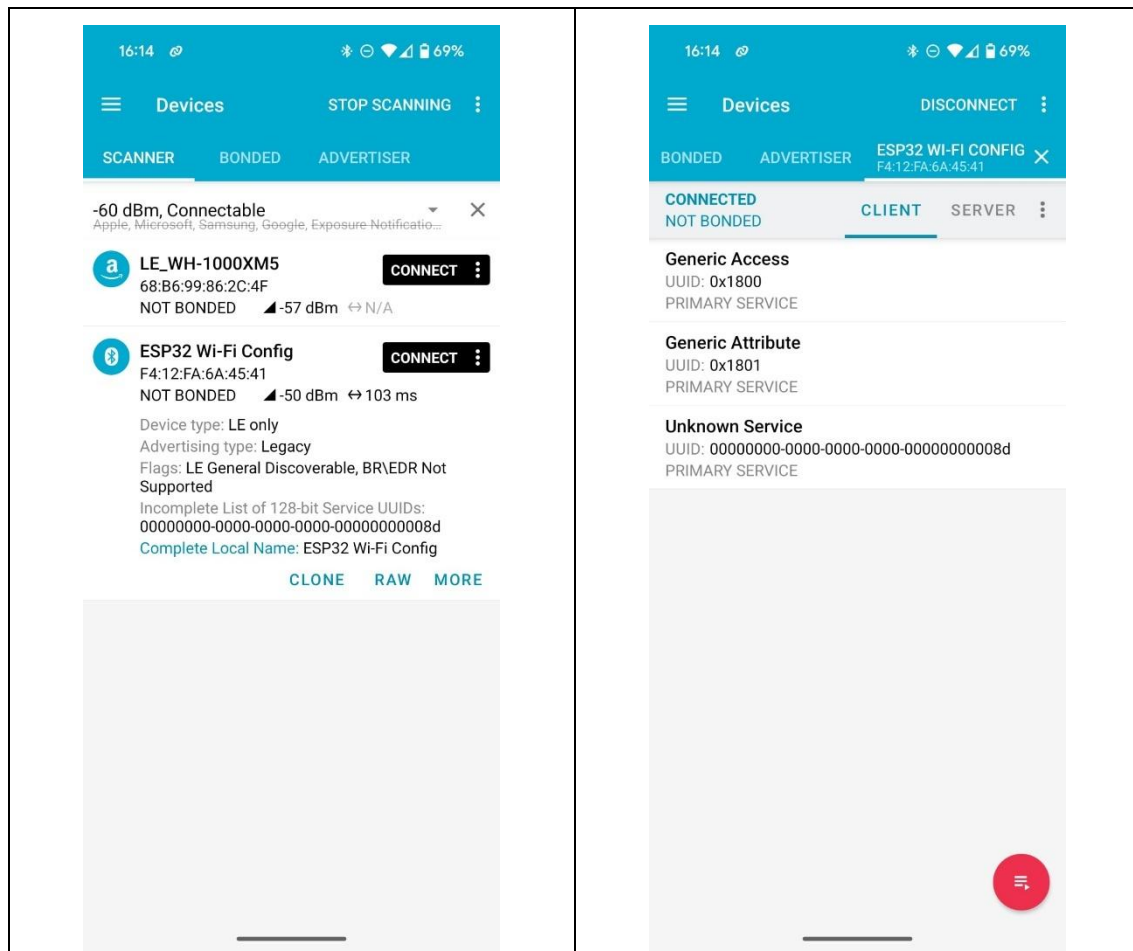


Figura 4.11 - Connexió BLE mitjançant nRF Connect

Un cop la connexió s'ha establert, l'ESP32 espera rebre el SSID i la Contrasenya d'un Wi-Fi. El missatge enviat per les credencials Wi-Fi, ha de ser en format TEXT (UFT-8) i l'estructura ha de ser l'SSID i la contrasenya escrits junts, únicament separats per el caràcter : (dos punts). En la següent figura es pot observar aquests mateixos passos:

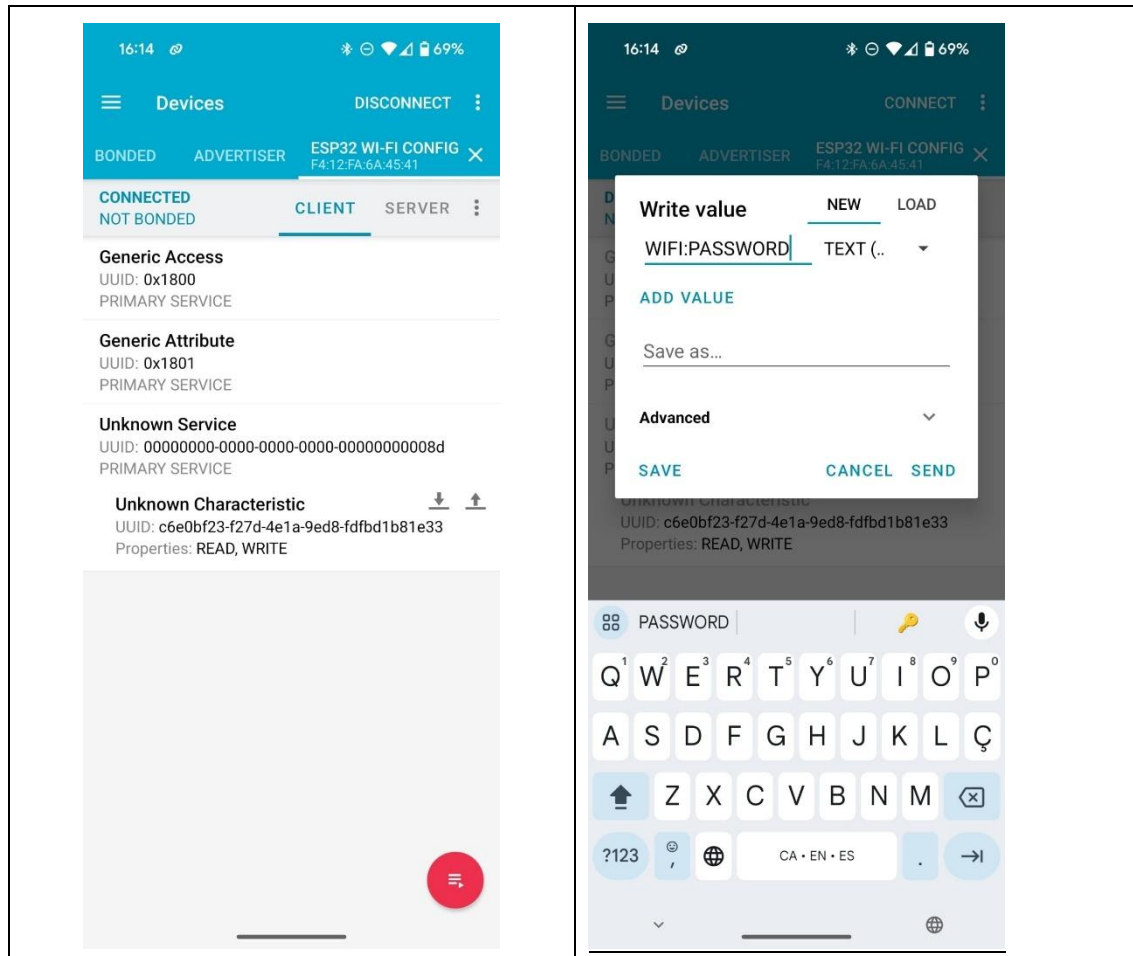


Figura 4.12 - Enviament de credencials Wi-Fi mitjançant BLE

4.4 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>

4.4.1 Proves realitzades amb la llibreria Preferences

Per guardar les credencials Wi-Fi rebudes per BLE, s'ha fet servir la llibreria Preferences. Inicialment, s'ha configurat el ESP32 per comprovar s'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'ESP32 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.

```

ESP-ROM:esp32s3-20210327
Build:Mar 27 2021
rst:0x1 (POWERON),boot:0x9 (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:1
load:0x3fce3818,len:0x109c
load:0x403c9700,len:0x4
load:0x403c9704,len:0xb50
load:0x403cc700,len:0x2fe4
entry 0x403c98ac
Serial connected
No stored Wi-Fi credentials.
BLE initialized
Advertising BLE service...
Connected to central: 56:3f:fb:39:fc:e1
Received data: SercommA788_2G:Q256UAUHX6X4BG
SSID: SercommA788_2G
Password: Q256UAUHX6X4BG
Connecting to Wi-Fi...
.....
Failed to connect to Wi-Fi.

```

Figura 4.13 - Emmagatzemant de credencials Wi-Fi amb Preferences i configuració Wi-Fi mitjançant BLE

En la figura anterior, es pot observar com al inici, l'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 l'ESP32 rep les credencials amb el format SSID:PASSWORD intenta connectar-se a la xarxa Wi-Fi.

En la següent figura, es pot observar com després d'un reinici, si l'ESP troba credencials Wi-Fi guardades en la memòria NVP, intenta connectar-se:

```

ESP-ROM:esp32s3-20210327
Build:Mar 27 2021
rst:0x1 (POWERON),boot:0x9 (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:1
load:0x3fce3818,len:0x109c
load:0x403c9700,len:0x4
load:0x403c9704,len:0xb50
load:0x403cc700,len:0x2fe4
entry 0x403c98ac
Serial connected
Found stored Wi-Fi credentials.
SSID: SercommA788_2G
Attempting to connect to Wi-Fi...
.....
Failed to connect to Wi-Fi.
BLE initialized
Advertising BLE service...

```

Figura 4.14 - Reutilització dades NVP

4.5 Estalvi d'energia i modes sleep

Per comprovar el correcte funcionament dels ESP32 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>

4.5.1 Mode light_sleep

Per comprovar el correcte funcionament d'aquest mode, s'han fet diversos experiments, 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 experiment; 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, l'ESP32 es desperta de mitja 1 segon abans. Aquests valors proporcionen força inexactitud per a una possible futura implementació.

A continuació, es poden trobar dos tipus de gràfiques diferents: A l'esquerra, les gràfiques indiquen el temps que l'ESP32 ha estat en mode light_sleep, a la dreta, les gràfiques indiquen la diferència de temps que hi havia entre el valor esperat i l'obtingut.

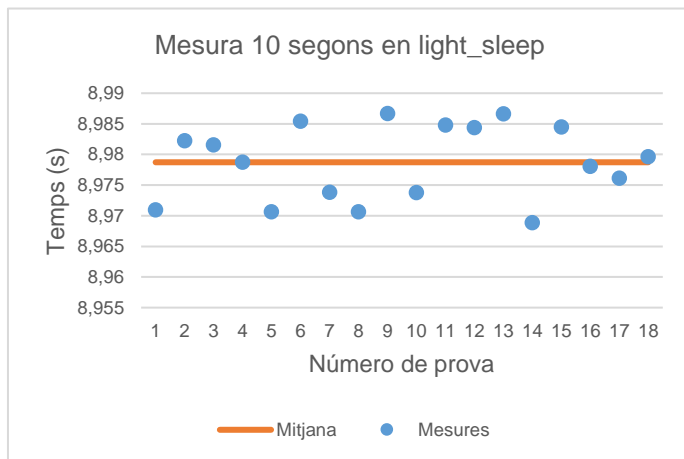


Figura 4.15.a – Mesura 10 segons en *light_sleep*

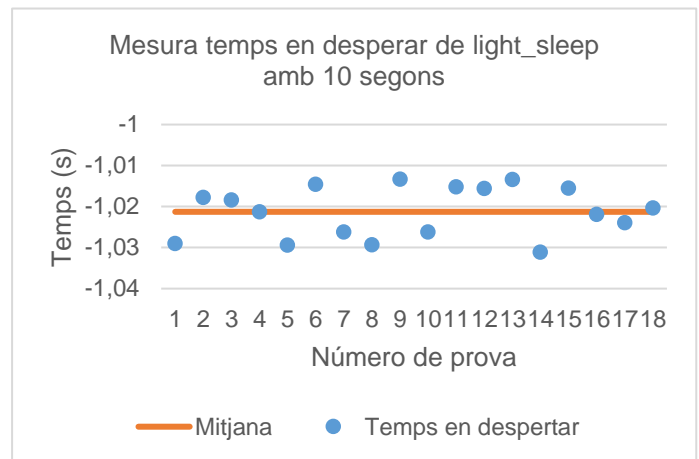
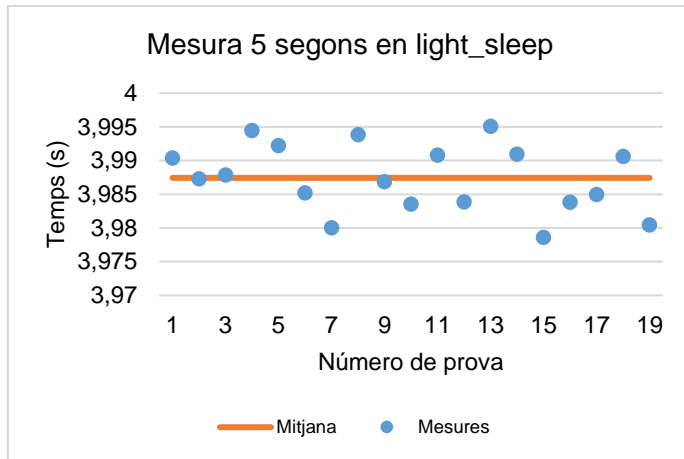
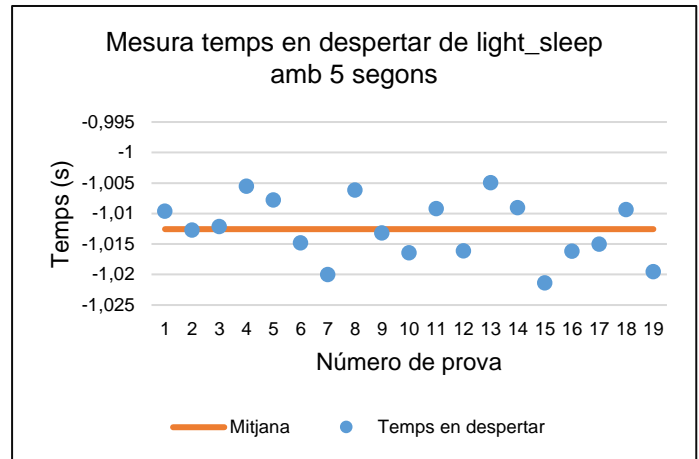
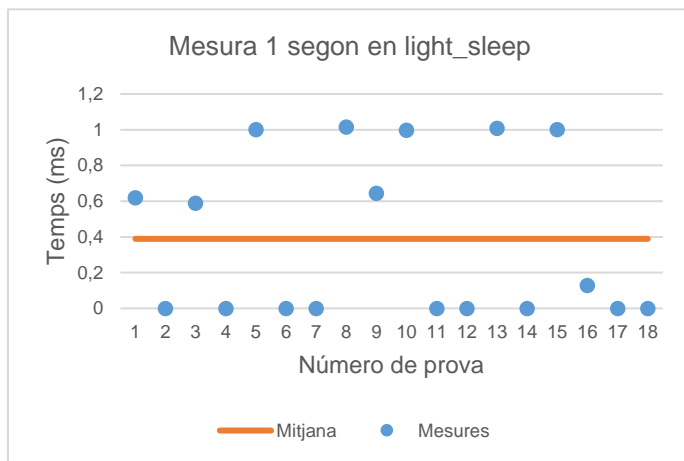
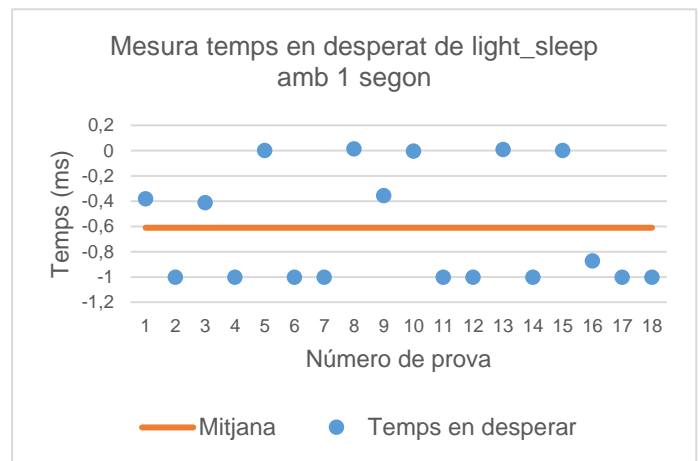
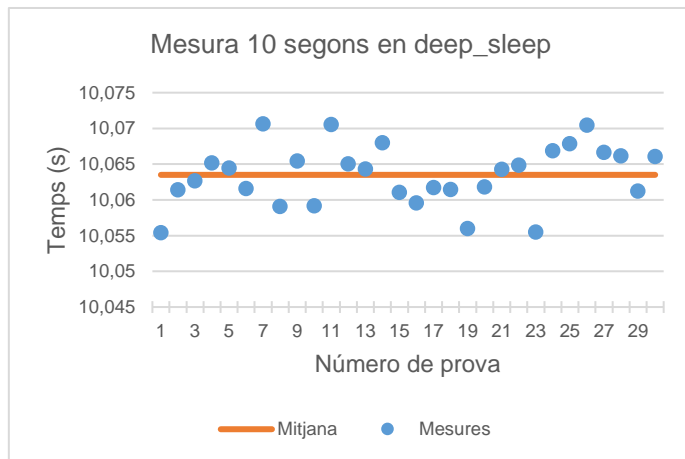
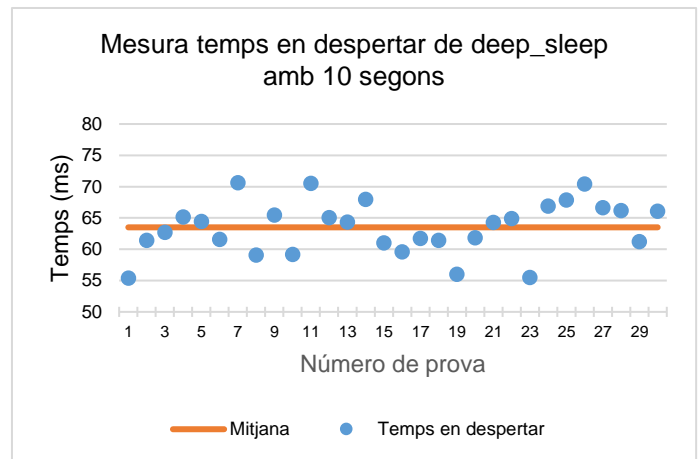
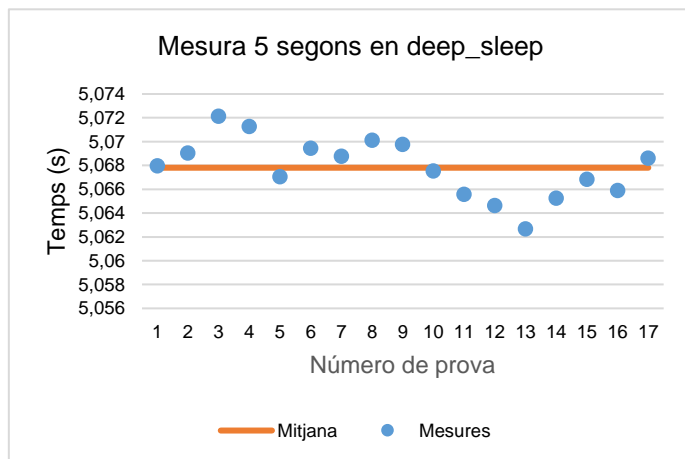
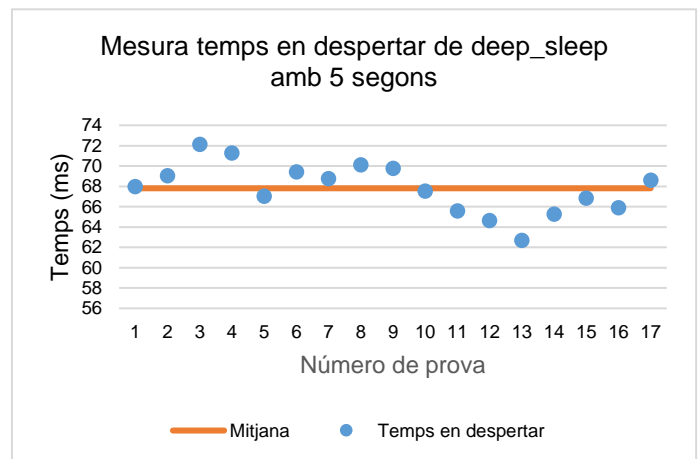
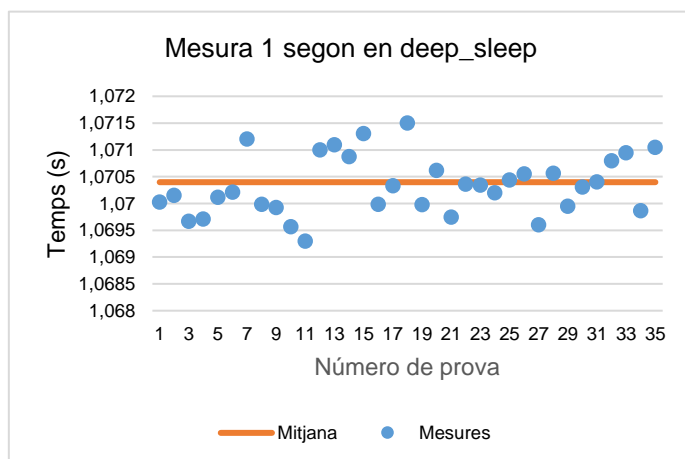
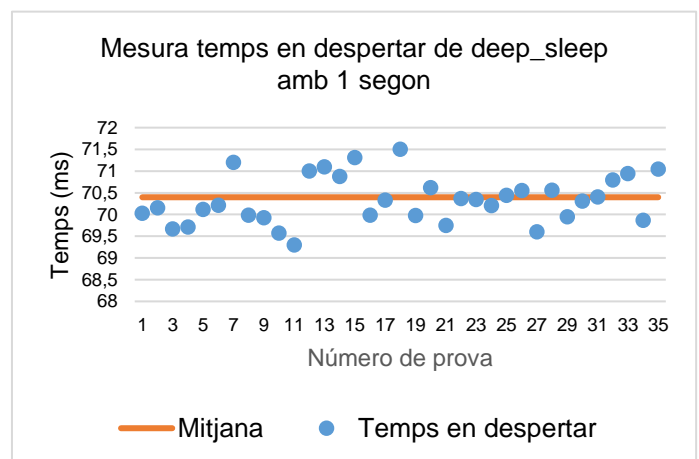


Figura 4.15.b – Diferència temps *deep_sleep* 10 segons i valor real

Figura 4.15.c – Mesura 5 segons en *light_sleep*Figura 4.15.d – Diferència temps *deep_sleep* 5 segons i valor realFigura 4.15.e – Mesura 1 segon en *light_sleep*Figura 4.15.f – Diferència temps *light_sleep* 1 segon i valor realFigura 4.15 - Comparativa temps esperat en *light_sleep* vs temps obtingut

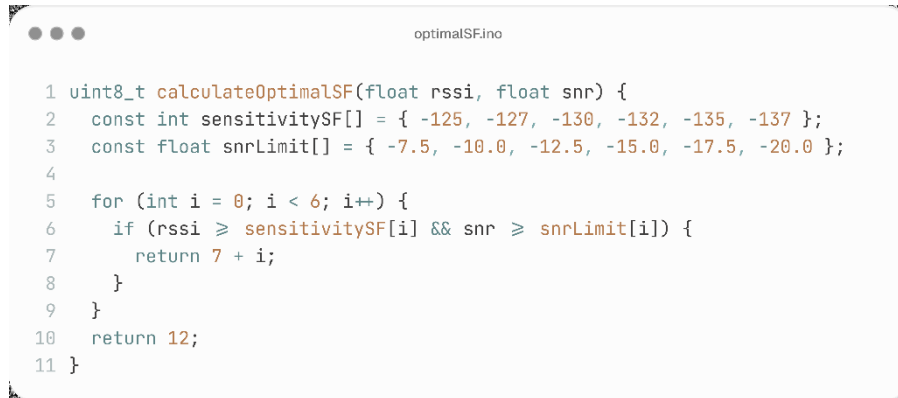
4.5.2 Mode *deep_sleep*

Les mesures realitzades en aquest mode, han estat idèntiques al mode anterior. En la figura 4.16 es mostren els resultats obtinguts. En aquest cas els valors obtinguts tenen molt sentit, i demostren un correcte funcionament d'aquest mode, ja que com es pot observar en les següents gràfiques, el node triga en despertar-se de mitja 70 ms.

Figura 4.16.a – Mesura de 10 segons en despertar *deep_sleep*Figura 4.16.b – Diferència temps *deep_sleep* 10 segons i valor realFigura 4.16.c – Mesura de 5 segons en despertar *deep_sleep*Figura 4.16.d – Diferència temps *deep_sleep* 5 segons i valor realFigura 4.16.e – Mesura de 1 segon en despertar *deep_sleep*Figura 4.16.f – Diferència temps *deep_sleep* 1 segon i valor realFigura 4.16 - Comparativa temps esperat en *deep_sleep* vs temps obtingut

4.6 Càlcul SF òptim

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



```
1 uint8_t calculateOptimalSF(float rssi, float snr) {
2     const int sensitivitySF[] = { -125, -127, -130, -132, -135, -137 };
3     const float snrLimit[] = { -7.5, -10.0, -12.5, -15.0, -17.5, -20.0 };
4
5     for (int i = 0; i < 6; i++) {
6         if (rssi ≥ sensitivitySF[i] && snr ≥ snrLimit[i]) {
7             return 7 + i;
8         }
9     }
10    return 12;
11 }
```

Figura 4.17 - Funció càlcul SF òptim

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

4.7 Disseny i implementació del protocol d'accés al medi

Un cop s'ha estudiat el comportament del ESP32 en els diferents àmbits en el qual es vol fer servir, es poden treure una sèrie de conclusions que permeten realitzar el disseny d'un protocol d'accés al medi sobre 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>

4.7.1 Escenari

Aquest protocol s'ha dissenyat com a prova de concepte, utilitzant un total de 4 nodes i un gateway.

4.7.2 Tipus de missatges

S'han definit 4 tipus de missatges en aquest protocol. Tots els missatges comencen sempre amb la mateixa capçalera, que està definida pels següents camps:

1. Tipus de missatge (2 bits)
 - a. 00 indica missatge Beacon
 - b. 01 indica missatge Request
 - c. 01 indica missatge Schedule
 - d. 11 indica missatge Data

2. Flag identificador de gateway (1 bits)
 - a. 0 indica que el dispositiu transmissor es un node
 - b. 1 indica que el dispositiu transmissor es un gateway
3. Flags no utilitzats, per a usos futurs (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 (la part superior indica el número de bit):

0	1	2	3	4	5	6	7
MSG Type		Flag	Payload				
Sender ID							
Receiver ID							

Figura 4.18 - Capçalera missatges

4.7.2.1 Beacon

El missatge Beacon, l'envia el gateway. Aquest missatge s'encarrega d'anunciar al gateway pel canal prèviament configurat.

L'estructura del paquet Beacon és la següent:

0	1	2	3	4	5	6	7
MSG Type: Beacon		Flag: Gateway	Payload				
Sender ID							
Receiver ID							

Figura 4.19 - Estructura missatge Beacon

4.7.2.2 Request

El missatge Request, és el missatge que envia cada node al gateway. La seva funció confirmar la recepció del Beacon i demanar recursos per poder enviar posteriorment.

0	1	2	3	4	5	6	7
MSG Type: Request		Flag: Node	MSG Type				
Sender ID							
Receiver ID							

Figura 4.20 - Estructura missatge Request

4.7.2.3 Schedule

El missatge Schedule, es el missatge que envia el gateway a tots els nodes que han respòs el seu missatge de Beacon amb un Request, 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:

0	1	2	3	4	5	6	7
MSG Type: Schedule		Flag: Gateway	Payload				
Sender ID							
Receiver ID							
Node ID 0							
SF Optim ID 0			Slot X ID0		Node ID 1		
...					SF Optim ID 1		
Slot X ID 1		Node ID 2					
...		SF Optim ID 2			Slot X ID 2		Node ID 3
Node ID 3							SF Optim ID 3
...		Slot X ID 3		...			

Figura 4.21 - Estructura missatge Schedule

Els paràmetres del missatge Schedule són:

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

En el pitjor dels casos (entenent que el pitjor dels casos seria quan hi ha 4 nodes disponibles, i el missatge a conseqüència d'això, el ToA augmenta), el missatge Schedule pot arribar a mesurar 76 bits, per tant 10 bytes.

4.7.2.4 Data

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

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

0	1	2	3	4	5	6	7
MSG Type: Data		Flag: Node	MSG Type				
Sender ID							
Receiver ID							
Data							

Figura 4.22 - Estructura missatge Data

4.7.3 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.

4.7.3.1 Beacon

Degut a que els missatges Beacon sempre s'envien amb els mateixos valors de configuració de LoRa, 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. L'SF és de 12 degut a poder maximitzar la cobertura.

En cas de respectar les restriccions vigents definides per l'ETSI, el Duty Cycle entre missatges Beacon hauria com a mínim de ser de 82,73 segons, per tant s'hauria d'esperar aquest temps abans de que el gateway tornés a enviar un altre Beacon.

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 Beacon

4.7.3.2 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 4.3 - ToA Request

4.7.3.3 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)
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 4.4 - ToA Schedule

4.7.3.4 Data

El ToA dels missatges Data varia en funció de l'SF utilitzat. 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. El missatge Data envia 1 byte de dades.

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 4.5 - ToA Data

4.7.4 Disseny protocol

Un cop s'han definit els tipus de missatges existents dintre del protocol, s'ha estimat el seu ToA i el duty cycle que caldria respectar, si no s'escolta abans d'enviar, s'explicarà el disseny d'aquest, juntament amb el motiu de les decisions realitzades.

El protocol es pot definir en 3 grans etapes:

- Descobriment
- Organització
- Enviament de dades

4.7.4.1 Etapa de descobriment

En l'etapa de descobriment, el gateway s'ha anunciant per un canal prèviament configurat mitjançant un missatge Beacon. Aquest missatge Beacon, sempre serà enviat amb un SF 12. Abans d'enviar el missatge, escota el canal per respectar el Polite Spectrum Access. 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.

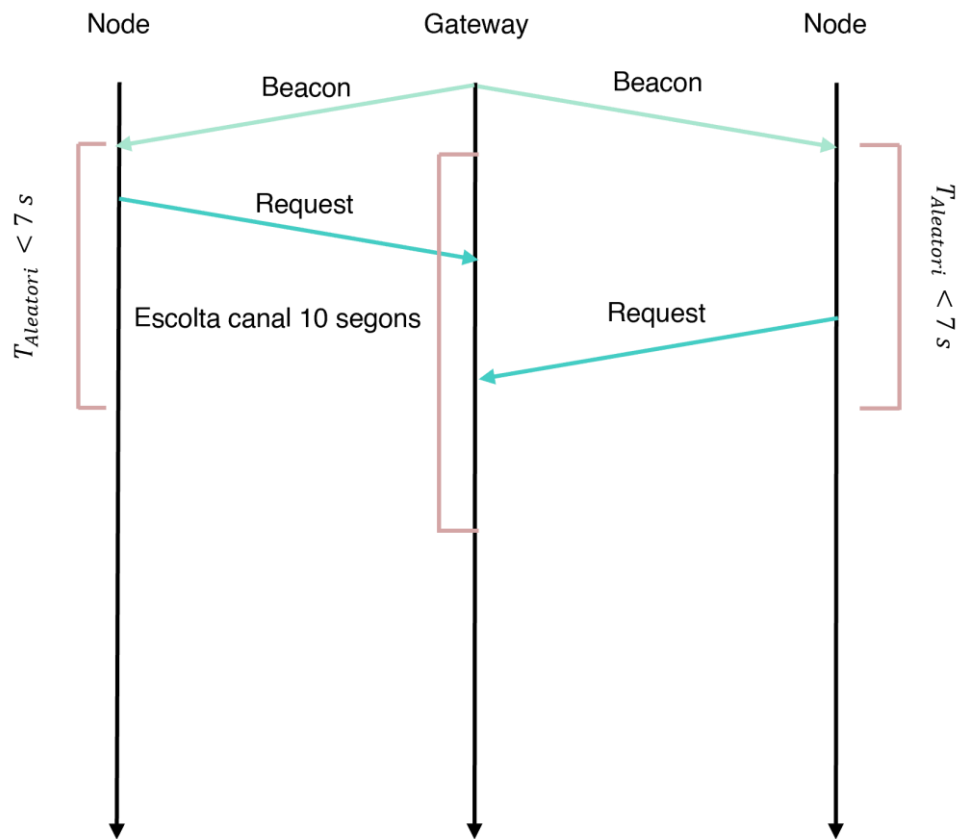


Figura 4.23 - Etapa de descobriment

4.7.4.1.1 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, és d'un interval aleatori entre 0 i 7 segons. Aquest interval s'ha definit així, ja que si els nodes envien les dades en el mateix instant, poden causar col·lisions i no establir connexió correctament amb el gateway. Per altra banda, els nodes abans d'enviar el missatge Request escolten el canal abans d'enviar, d'aquesta manera no cal respectar el DC.

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 altra banda, també s'ha de considerar el ToA del missatge Request. Per últim, s'ha tingut en compte que els nodes, s'esperen 1 segon més abans d'enviar, si el canal es troba ocupat. 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 motiu, i per deixar una mica de marge a error, s'ha definit un temps aleatori de 7 segons.

4.7.4.2 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 escolta el canal i envia un únic missatge Schedule per el canal, també amb SF 12, indicant als nodes en quina finestra de temps (slot) els hi toca enviar, i quin es l'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 40 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 decodificar el missatge Schedule, configurar l'SF, calcular el temps per dormir, i posar-se a dormir en mode deep_sleep.

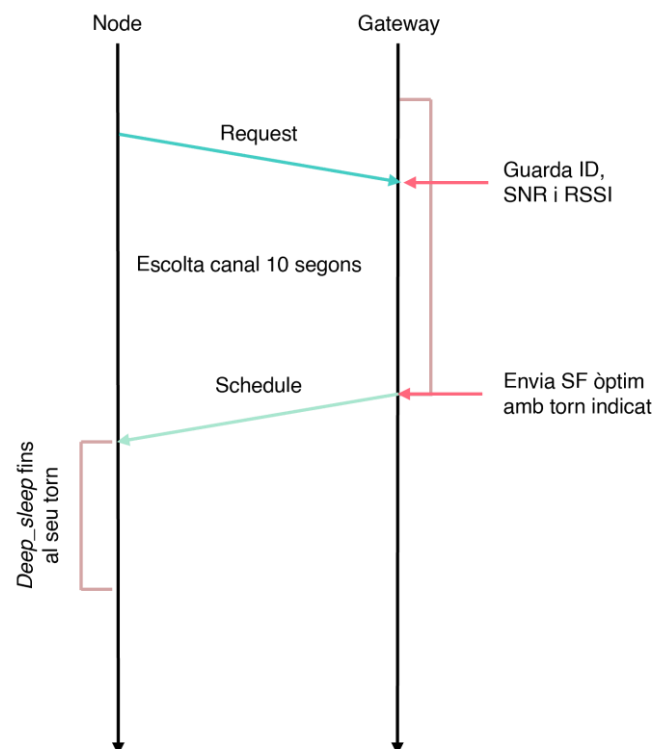


Figura 4.24 - Etapa d'organització

4.7.4.3 Etapa d'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 podrà transmetre el missatge Data sense risc de col·lisió, 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 un altre cop a l'etapa de descobriment de nodes.

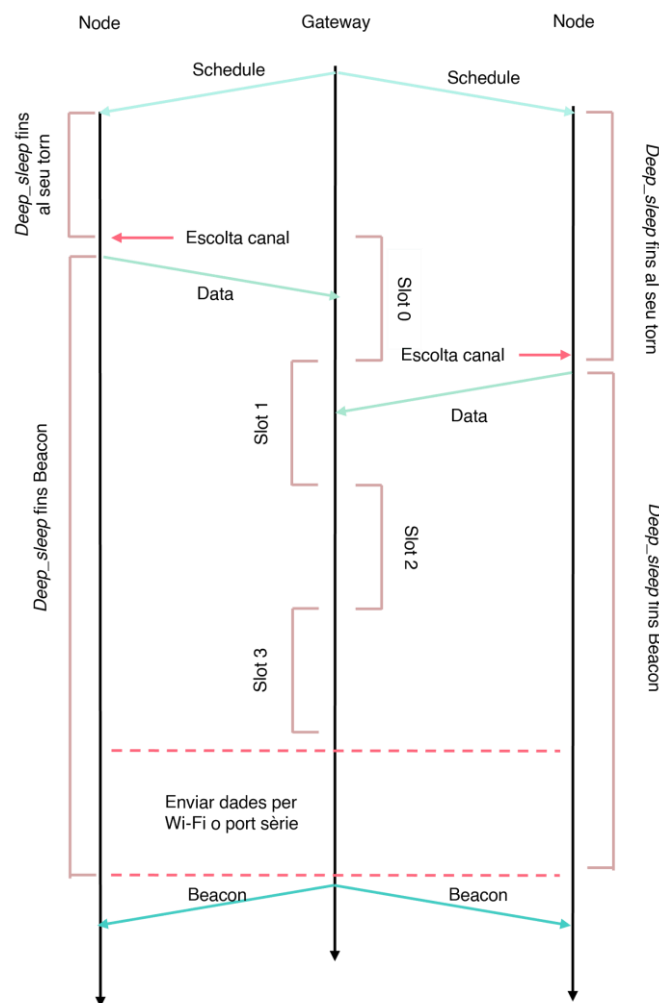


Figura 4.25 - Etapa enviament de dades amb 2 nodes ubicats en els slots 0 i 1

4.7.4.3.1 Justificació dels temps definits per cada slot

La ranura de temps de cada slot s'ha definit com a 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 pel càlcul del temps de cada slot, és 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, és 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. No és desitjable que el missatge `Data` s'hagués d'enviar amb l'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 prova 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 o un enviar més dades.

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

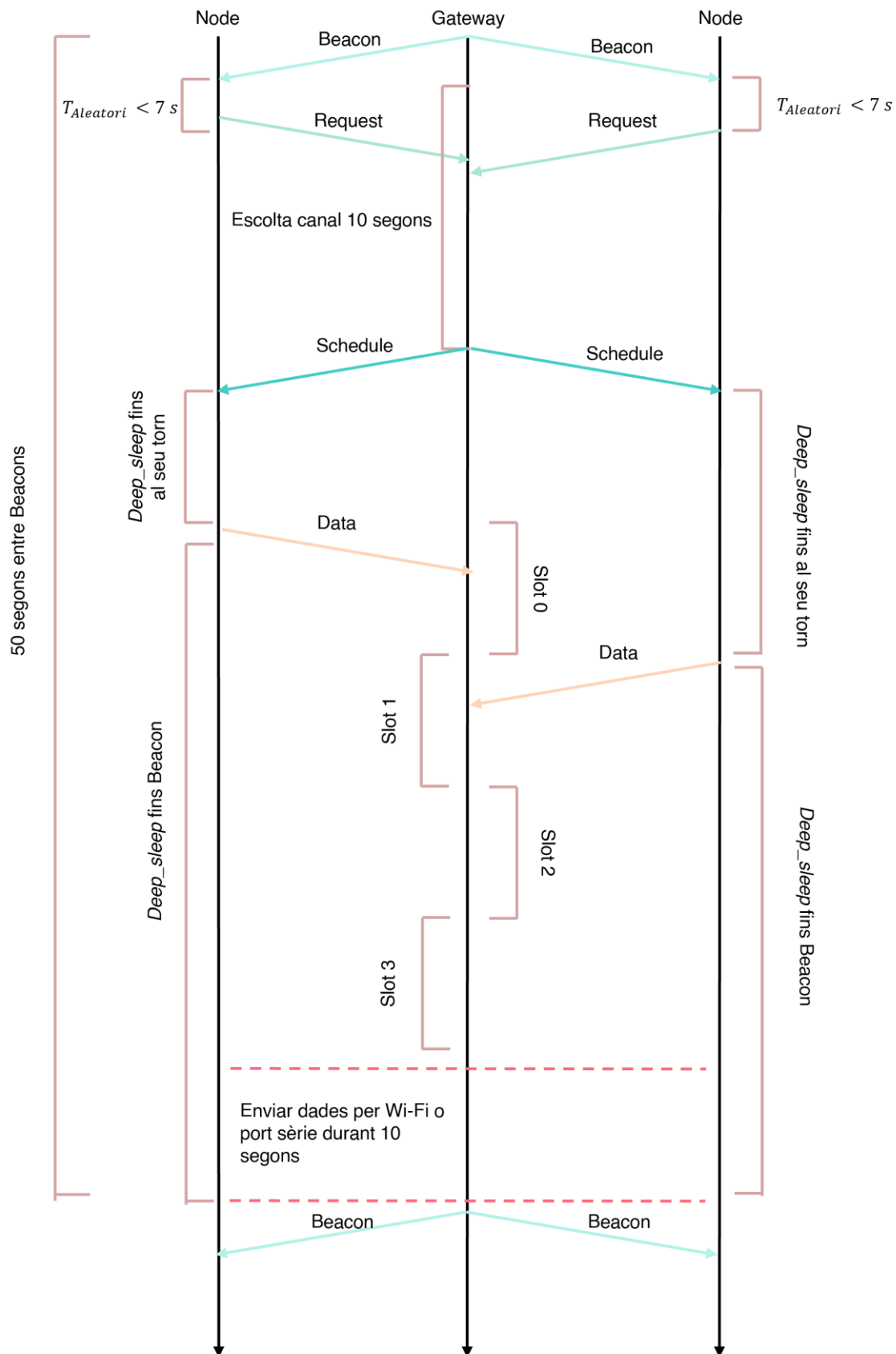


Figura 4.26 - Diagrama de flux protocol

4.7.5 Estats del node en memòria RTC

Per poder continuar l'execució normal del codi del node un cop s'ha despertat del 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:

```

1  enum NodeState {
2      NODE_WAIT_BEACON,
3      NODE_REQUEST_SENT,
4      NODE_WAIT_SCHEDULE,
5      NODE_WAIT_SLOT,
6      NODE_SEND_DATA,
7      NODE_SLEEP
8  };
9
10 RTC_DATA_ATTR NodeState savedState = NODE_WAIT_BEACON;
11
12 void resumeFromDeepSleep() {
13     Serial.println("[Node] ⇒ Reanudando tras deep sleep...");
14     Serial.println("U");
15
16     currentNodeState = savedState;
17     assignedSF = savedSF;
18     assignedSlot = savedSlot;
19
20     Serial.print("    savedState=");
21     Serial.println((int)savedState);
22     Serial.print("    assignedSF=");
23     Serial.println(assignedSF);
24     Serial.print("    assignedSlot=");
25     Serial.println(assignedSlot);
26     Serial.print("    sleepTimeMs=");
27     Serial.println(sleepTimeMs);
28
29     if(currentNodeState == NODE_WAIT_SLOT) {
30         currentNodeState = NODE_SEND_DATA;
31         radio.setSpreadingFactor(assignedSF);
32         sendData();
33         currentNodeState = NODE_SLEEP;
34         Serial.println("[Node] *** Data enviada tras deep sleep ⇒ FIN ***");
35     } else if(savedState == NODE_WAIT_BEACON) {
36         currentNodeState = NODE_WAIT_BEACON;
37         startReceive();
38     } else {
39         Serial.println("[Node] No era WAIT_SLOT ⇒ no transmitimos. Revisar lógica...");
40     }
41     wokeFromDeepSleep = false;
42 }

```

Figura 4.27 - Funcionament estats node

Com es pot observar en la figura anterior, s'ha definit la funció resumeFromDeepSleep() per tractar els possibles casos en els quals el node es desperta del deep_sleep. Aquests casos són dos, quan el node espera un slot per enviar dades, o quan el node espera un Beacon. Això és pel fet que el node només es posa a dormir en aquests dos estats. Tots els estats restants es van actualitzant a mesura que s'executa el codi però no tenen una funció real. Aquests estats es mantenen per si s'efectués una eventual modificació en un futur i així facilitar l'actualització del codi.

4.8 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/Analisis>

4.8.1 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 realitzat tot el procés del protocol varies vegades. Tant el node com el gateway, envien per el port sèrie quin tipus de missatge envien i indiquen 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 aspecte és important per entendre el comportament del dispositiu.

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

Temps entre missatges	Temps mínim (s)	Temps màxim (s)	Mitja (s)	Número 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 4.6 - Resultats mesures temps entre missatges

Per poder entendre correctament si aquests valors són els esperats o no, en la taula següent veuríem mostra 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 4.7 - Comparació valors teòrics amb valors mesurats

4.8.1.1 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 com a màxim de 7 segons abans d'enviar el missatge, per evitar possibles col·lisions amb altres nodes.

A la taula 4.7, podem comparar el valor màxim teòric i el valor mitjà obtingut. Es pot veure que el node compleix amb el valor esperat.

Per altre banda, en la taula 4.6, s'observa un valor màxim de 3,21 segons, per tant, no supera el temps màxim definit i es comporta adequadament.

4.8.1.2 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 4.6, 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.

4.8.1.3 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 envii el missatge Data, el gateway no espera rebre cap altre missatge.

En la taula 4.6, es pot observar com el gateway envia el següent Beacon una mitja de 30 segons després d'haver rebut el missatge Data. Aquest comportament és l'esperat, degut a que és 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.

4.8.1.4 Comparació de temps entre missatges Beacon

En aquesta mesura, es pot observar que el temps entre missatges Beacon es de 51,42. Com abans d'enviar un missatge s'escolta el canal, no es requereix complir la normativa del DC. En

cas d'haver de respectar el DC, com s'ha vist en la taula 4.2, el temps entre missatges Beacon hauria de ser de 82,73 s.

4.8.2 Anàlisis temps en deep_sleep dels nodes

Per tal d'efectuar 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 va a dormir o desperta. Imitant la prova anterior, s'ha utilitzat 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, és 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.

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

Com es pot visualitzar a la taula 4.8, el temps que està el node en mode deep_sleep, és el temps esperat. Aquest temps és el temps que el node està dormint 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 prova	Mitja (ms)	Valor Màxim (ms)	Valor Mínim (ms)	Valor teòric (ms)
10	10087,2	10089	10085	10000

Taula 4.8 - Comparació deep_sleep entre missatges Schedule i Data

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

Com es pot visualitzar a la taula 4.9, el temps que el node està en deep_sleep és el temps esperat. Aquest temps és el temps que el node es posa a dormir 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 prova	Mitja (ms)	Valor Màxim (ms)	Valor Mínim (ms)	Valor teòric (ms)
10	25073,6	25082	25061	25000

Taula 4.9 - Comparació deep_sleep entre missatges Data i Beacon

4.8.3 Anàlisi de 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.

4.8.3.1 Anàlisi funcionament del 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 òptima amb cada node.

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

```
[Gateway] Enviando Beacon en SF12...
[Gateway] Beacon enviadoBTX
RRX
[Gateway] REQUEST de nodo=0x2 => RSSI=-9.00 dBm, SNR=4.25 => SF=7
[Gateway] Nuevo nodo idx=0
RRX
[Gateway] REQUEST de nodo=0x3 => RSSI=-26.00 dBm, SNR=4.75 => SF=7
[Gateway] Nuevo nodo idx=1
[Gateway] Enviando SCHEDULE...
[Gateway] SCHEDULE enviado OK!
STX
[Gateway] Recibido un Mensaje type=2 no esperado aquí.
[Gateway] *** Enviando SCHEDULE completado ***
[Gateway] Esperamos 10s antes de empezar slots de Data...
[Gateway] Iniciando recepción de DATA en slots de 5s.
[Gateway] Slot #0 (5s de recepción)
[Gateway] SF=7
DRX
[Gateway] DATA de nodo 0x2: valor=255 => Temp=25.50°C, SF=7
[Gateway] DATA de nodo 0x2: dataValue=11111111 => temp=255.00 C, SF=7
[Gateway] Slot #1 (5s de recepción)
[Gateway] SF=7
DRX
[Gateway] DATA de nodo 0x3: valor=255 => Temp=25.50°C, SF=7
[Gateway] DATA de nodo 0x3: dataValue=11111111 => temp=255.00 C, SF=7
[Gateway] Slot #2 (5s de recepción)
[Gateway] Slot 2 sin nodo asignado, SF=12.
[Gateway] Slot #3 (5s de recepción)
[Gateway] Slot 3 sin nodo asignado, SF=12.
[Gateway] Terminó la recepción de Data de todos los slots.
[Gateway] Esperamos 10s para (simular) enviar datos por WiFi/serial...
[Gateway] RESUMEN DE DATOS RECIBIDOS:
- Nodo=0x2, dataValue=255 => 25.50 C, SF=7
- Nodo=0x3, dataValue=255 => 25.50 C, SF=7
```

Figura 4.28 - Implementació protocol – Gateway i dos nodes

4.8.3.2 Anàlisi funcionament dels nodes

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

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

Per altra banda, podem observar com després 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 després d'enviar els missatges Data, ambdós nodes se'n van a dormir 20 segons abans de rebre un altre cop els missatges Beacon.

```
[Nodo] Beacon recibido correctamente!
[Nodo] Escaneando canal, intento 1 ...
[Nodo] Canal LIBRE, esperando 4322 ms antes de enviar mensaje.
[Nodo] Enviando Request al Gateway...
[Nodo] Request enviado exitosamente!
RTX
[Nodo] Mensaje recibido, pero no es un Beacon esperado o estamos en otro estado.
[Nodo] Recibido SCHEDULE desde Gateway!
SRX
[Nodo] Mi SF asignado = 7, Slot asignado = 1
[Nodo] Entrando en deep sleep durante 15000ms hasta mi slot...
D
[Nodo] *** Entrando a deep sleep ***
ESP-ROM:esp32s3-20210327
Build:Mar 27 2021
rst:0x5 (DSLEEP),boot:0x9 (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:1
load:0x3fce3818,len:0x109c
load:0x403c9700,len:0x4
load:0x403c9704,len:0xb50
load:0x403cc700,len:0x2fe4
entry 0x403c98ac
[Nodo] Wakeup cause: 4
[Nodo] Radio inicializada correctamente!
[Nodo] => Reanudando tras deep sleep...
U
    savedState=3
    assignedSF=7
    assignedSlot=1
    sleepTimeMs=15000
[Nodo] Es mi turno (slot). Envío DATA y calculo tiempo de sueño...
[Nodo] Escaneando canal, intento 1 ...
[Nodo] Canal LIBRE, esperando 270 ms antes de enviar mensaje.
[Nodo] DATA enviado exitosamente!
DTX
[Nodo] Tiempo total de sueño calculado: 20000 ms
D
[Nodo] *** Entrando a deep sleep ***
ESP-ROM:esp32s3-20210327
Build:Mar 27 2021
rst:0x5 (DSLEEP),boot:0x9 (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:1
load:0x3fce3818,len:0x109c
load:0x403c9700,len:0x4
```

Figura 4.29 - Implementació protocol - Node 1

```

[Nodo] (DEBUG) Sigo en WAIT_BEACON, esperando Beacon...
BRX
[Nodo] Beacon recibido correctamente!
[Nodo] Escaneando canal, intento 1 ...
[Nodo] Canal LIBRE, esperando 1677 ms antes de enviar mensaje.
[Nodo] Enviando Request al Gateway...
[Nodo] Request enviado exitosamente!
RTX
[Nodo] Mensaje recibido, pero no es un Beacon esperado o estamos en otro estado.
[Nodo] Mensaje recibido, pero no es un Beacon esperado o estamos en otro estado.
[Nodo] Recibido SCHEDULE desde Gateway!
SRX
[Nodo] Mi SF asignado = 7, Slot asignado = 0
...
U
    savedState=3
    assignedSF=7
    assignedSlot=0
    sleepTimeMs=10000
[Nodo] Es mi turno (slot). Envio DATA y calculo tiempo de sueño...
[Nodo] Escaneando canal, intento 1 ...
[Nodo] Canal LIBRE, esperando 323 ms antes de enviar mensaje.
[Nodo] DATA enviado exitosamente!
DTX
[Nodo] Tiempo total de sueño calculado: 25000 ms
D
[Nodo] *** Entrando a deep sleep ***
[Nodo] Wakeup cause: 4
[Nodo] Radio inicializada correctamente!
[Nodo] => Reanudando tras deep sleep...
U
    savedState=0
    assignedSF=12
    assignedSlot=0
    sleepTimeMs=10000
[Nodo] (DEBUG) Sigo en WAIT_BEACON, esperando Beacon...
[Nodo] (DEBUG) Sigo en WAIT_BEACON, esperando Beacon...
BRX
[Nodo] Beacon recibido correctamente!

```

Figura 4.30 - Implementació protocol – Node 2

4.8.3.3 Anàlisi possible col·lisió amb missatges Request

Com ja s'ha esmentat anteriorment, una de les possibles conseqüències d'utilitzar CSMA com a protocol d'accés al medi és que hi hagin col·lisions en els missatges Request.

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


```

[Nodo] Beacon recibido correctamente!
[Nodo] Escaneando canal, intento 1 ...
[Nodo] Canal LIBRE, esperando 1785 ms antes de enviar mensaje.
[Nodo] Enviando Request al Gateway...
[Nodo] Request enviado exitosamente!
RTX
[Nodo] Mensaje recibido, pero no es un Beacon esperado o estamos en otro estado.
[Nodo] Recibido SCHEDULE desde Gateway!
SRX
[Nodo] No estoy en el SCHEDULE => Entraré en modo SLEEP hasta el próximo Beacon...
[Nodo] Entrando en deep sleep durante 40000 ms hasta el próximo Beacon...
D
[Nodo] *** Entrando a deep sleep ***
ESP-ROM:esp32s3-20210327
Build:Mar 27 2021
rst:0x5 (DSLEEP),boot:0x9 (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:1
load:0x3fce3818,len:0x109c
load:0x403c9700,len:0x4
load:0x403c9704,len:0xb50
load:0x403cc700,len:0x2fe4
entry 0x403c98ac
[Nodo] Wakeup cause: 4
[Nodo] Radio inicializada correctamente!
[Nodo] => Reanudando tras deep sleep...
U
    savedState=0
    assignedSF=12
    assignedSlot=255
    sleepTimeMs=40000
BRX
[Nodo] Beacon recibido correctamente!

```

Figura 4.31 - 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.

5 Discussió

5.1 Validació funcionament protocol

5.1.1 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.

5.1.2 Temps en mode deep_sleep

Els resultats de les mesures realitzades amb el mode deep_sleep, es corresponen amb 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.

5.1.3 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 és sempre el mateix, ha permès una correcta sincronització entre els nodes i el gateway.

S'ha pogut comprovar que el mecanisme per resoldre col·lisions de missatges Request funciona correctament evitant així que els nodes entrin en bucles infinits.

5.2 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 hagués canviat el resultat final del protocol, si s'hagués 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 tal de poder enviar un missatge han d'escollar el canal.

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

5.3 Comparativa amb altres propostes realitzades

Una de les propostes que s'havia plantejat com a protocol final, plantejava la possibilitat de definir en funció de l'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 de l'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>

6 Conclusions i treball futur

6.1 Conclusions

Aquest treball ha estat un primer pas pel desenvolupament d'una solució IoT per a projectes de ciència ciutadana amb característiques de producte comercial, ie. robustesa i facilitat d'ús, que aportí diferents solucions de connectivitat per a, per ex., 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 in auna altra placa XIAO ESP32S3. S'han realitzat diverses proves per poder entendre el funcionament d'aquestes plaques de desenvolupament en certs escenaris, per posteriorment dissenyar i implementar de la manera més òptima possible, una prova de concepte per la implementació d'un protocol de comunicació IoT basat en LoRa. Durant gran part del desenvolupament d'aquest treball, s'ha fet servir únicament les plaques Heltec, ja que no s'havia tingut accés a les plaques XIAO fins a l'etapa final del treball.

Aquestes proves 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 el disseny, implementació i test d'una primera prova de concepte 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.

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

6.2 Línies futures

Com ja s'ha esmentat anteriorment, aquest treball és un primer pas, i encara requereix un treball més extens per a la seva implementació final. A continuació s'expliquen possibles millores que es podrien realitzar.

6.2.1 Col·lisió missatges Request

La col·lisió dels missatges Request és conseqüència de 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, es podria assignar un slot a cada node en funció del seu ID.

Per altra 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 hagués d'esperar molt de temps per enviar les dades tot i que el canal no estigués utilitzat.

6.2.2 ID dinàmic dels nodes

Actualment, en aquesta prova de concepte l'ID dels nodes ja ve 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, seguint els exemples realitzats en aquest treball.

6.2.3 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.

6.2.4 Implementació del suport per varis gateways

Aquesta prova de concepte, no contempla la possibilitat d'utilitzar més d'un gateway. Una millora d'aquest protocol seria la possibilitat d'incorporar varis gateways 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 en funció de la cobertura.

6.2.5 Confirmació de la recepció de missatges

En la implementació actual no s'ha fet servir cap mecanisme de confirmació de recepció de missatges. Podria ser interessant utilitzar un mecanisme de confirmació de missatges pels missatges Data.

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

- Aprofitant 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.
- 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 o no ha rebut un missatge de Data de cada node. Donat aquest cas, es podria fer servir l'SF12 per sincronitzar els nodes i el gateway de manera que s'utilitzi el mateix SF, com ja es fa amb altres missatges.

6.3 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, enviés 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 executa el codi per la configuració del Wi-Fi mitjançant BLE.

6.4 Probabilitat d'error del CAD

A causa de l'elevada probabilitat d'error que s'ha obtingut en les proves realitzades al CAD, caldria dur a terme més proves però definir amb més exactitud el seu funcionament. Es podria realitzar, per exemple, la mateixa prova realitzada amb altres transceptors com ara el SX1272 en comptes del SX1262 com s'ha fet en aquest treball.

6.5 Implementació servidor

En aquest treball, no s'ha implementat cap servidor ni base de dades per recollir les dades enviades pels nodes, per aquest motiu seria interessant la implementació 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.

6.6 Aplicació mòbil

Podria ser interessant una eventual implementació d'una aplicació web o d'una aplicació mòbil en la qual, l'usuari sigui capaç de veure l'estat dels nodes, poder configurar-los o, fins i tot, poder visualitzar les dades recollides. Aquesta implementació facilitaria l'ús i monitoratge dels nodes.

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, dificulta la implementació d'aquest apart del codi per proporcionar tant el Wi-Fi, com el protocol LoRa, com les solucions de connectivitat en un sol dispositiu ESP32.

6.7 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. Es proposa la realització d'un Fork del projecte en comptes de continuar contribuint en el repositori original. D'aquesta manera 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.

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

6.8 Estudi Sostenibilitat

6.8.1 Desenvolupament del treball final d'estudis

6.8.1.1 Punt de vista ambiental

Aquest projecte utilitza un enfoc molt vinculat amb l'impacte ambiental que pot arribar a tenir.

En concret, en aquest projecte, com ja s'ha explicat en més detall en l'apartat 4.8.2, cada node està de mitja 35 segons dormint i 15 segons despert. Aquest 70% del temps, el node només consumeix 7 μ A. Aquest fet fa que el consum sigui ínfim i les bateries que es poden utilitzar en un futur podrien arribar a durar anys.

6.8.1.2 Punt de vista econòmic

En aquest projecte, el material utilitzat ha estat molt escàs i molt econòmic. Els dispositius IoT basats en microcontroladors ESP32 són força econòmics, tenint un cost d'uns 20€ com a màxim. Per altra banda, aquest projecte s'ha realitzat usant eines de desenvolupament lliure, com ara bé Arduino IDE i Python.

A més a més, per fer servir els dispositius ESP32 dintre de l'entorn d'aquest treball, es fa servir protocols de comunicacions com LoRa o BLE els quals no requereixen pagar cap operador extern pel seu ús.

6.8.1.3 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.

Per altra banda, com ja s'ha esmentat en la introducció, el motiu d'aquest treball és desenvolupar una eina per a dur a terme projectes de ciència ciutadana. Un primer pas per dur a terme un projecte de ciència ciutadana, implica que tant investigadors com la resta de societat que hi participa es beneficiï d'aquest projecte. [12]

6.8.2 Execució del projecte

6.8.2.1 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 a 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.

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.

6.8.2.2 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 fer servir les funcionalitats implementades només es requereix un node i un gateway, per tant, el cost mínim seria d'uns 40 €.

Per altra banda, si es vol fer servir els nodes com a dispositius portables, es requereix una bateria externa, la qual pot tenir un cost d'uns 10€ extra. També s'hauria d'afegir el cost dels sensors que farien servir els nodes. Agafant com a exemple un sensor de temperatura i pressió com el BME280 que té un preu d'11 €, s'hauria d'afegir al cost total uns altres 44 €. Finalment, s'haurà de tenir en compte la utilització de caixes per embolicar els ESP32 juntament amb l'antena LoRa i els sensors.

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.

6.8.2.3 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 de funcionament molt simple, ja que no requereix una configuració inicial. Únicament proporcionar 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.

6.8.3 Riscos i limitacions

6.8.3.1 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.

6.8.3.2 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.

6.8.3.3 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.

Bibliografía

- [1] «unsigned,» [En línea]. Available: <https://unsigned.io/website/understanding-lora-parameters/>. [Último acceso: 11 2024].
- [2] Sebas, 2 10 2017. [En línea]. Available: <https://medium.com/@Sabasacustico/haciendo-iot-con-lora-capitulo-2-tipos-y-clases-de-nodos-3856aba0e5be>. [Último acceso: 12 2024].
- [3] «etsi.org,» [En línea]. Available: https://www.etsi.org/deliver/etsi_en/300200_300299/30022002/03.02.01_60/en_30022002v030201p.pdf.
- [4] O. Khalil, 05 12 2020. [En línea]. Available: https://radiocrafts.com/uploads/AN051_High_Throughput_Networking_With_Polite_Spectrum_Access_in_RIIIM.pdf. [Último acceso: 02 2025].
- [5] J. H. A. S. i. E. D. P. Martijn Saelens, «Springer Open,» 02 Septiembre 2019. [En línea]. Available: <https://jwcn-urasipjournals.springeropen.com/articles/10.1186/s13638-019-1502-5>. [Último acceso: 01 2025].
- [6] J. G. Carmenate, «programafacil.com,» [En línea]. Available: <https://programafacil.com/esp8266/esp32/>. [Último acceso: 01 2025].
- [7] «espboards.dev,» 22 05 2023. [En línea]. Available: <https://www.espboards.dev/es/blog/esp32-soc-options/>.
- [8] «ESPBoards,» 16 10 2023. [En línea]. Available: <https://www.espboards.dev/blog/esp32-power-optimisation/>. [Último acceso: 11 2024].
- [9] «espressif.con,» [En línea]. Available: <https://docs.espressif.com/projects/arduino-esp32/en/latest/tutorials/preferences.html>. [Último acceso: 10 2024].
- [1 K. N. i. R. D. W. Gilang Hijrian Fahreja, «The Effect of Spreading Factor Value on the
- 0] Number of Gateways in the LoRaWAN Network at Bandung City,» 12 2023. [En línea]. Available: <https://www.jocm.us/2023/JCM-V18N12-768.pdf>. [Último acceso: 11 2024].
- [1 Semtech. [En línea]. Available: <https://www.semtech.com/design-support/lora-calculator>.
- 1] [Último acceso: 12 2024].
- [1 [En línea]. Available: <https://canviaelmon.upc.edu/ca/compromis-social/iniciatives/ciencia->
- 2] ciutadana. [Último acceso: 01 2025].