Securització d'un sistema de telemetria IoT

Autor: Adrián Barredo Cordón

Tutor: Joan Caparrós Ramírez

Professor: Helena Rifà Pous

Màster Universitari en Seguretat de les Tecnologies de la Informació i de les Comunicacions

Àrea TFM-Seguridad en la Internet de las cosas

29/12/2020

Crèdits/Copyright

FITXA DEL TREBALL FINAL

Títol del treball:	*Securització d'un sistema de telemetria IoT*
Nom de l'autor:	*Adrián Barredo Cordón*
Nom del col·laborador/a docent:	*Joan Caparrós Ramírez*
Nom del PRA:	*Helena Rifà Pous*
Data de lliurament (mm/aaaa):	*12/2020*
Titulació o programa:	*Màster Universitari en Seguretat de les Tecnologies de la Informació i de les Comunicacions*
Àrea del Treball Final:	*Àrea TFM-Seguridad en la Internet de las cosas*
Idioma del treball:	*Català*
Paraules clau	*BLE, IoT, Telemetria*

Resum del Treball (màxim 250 paraules):

Avui dia els dispositius IoT estan molt estesos per diversos àmbits de la nostra vida quotidiana però molts fabricants descuiden o infravaloren la importància de les mesures de seguretat en aquests dispositius posant en risc la nostra privacitat. Tot i que hi ha fabricants que implementen mesures de seguretat adequades en els seus dispositius IoT, n'hi ha d'altres que no implementen cap tipus de mesura de seguretat.

En aquest treball es pretén implementar un prototip d'un sistema de telemetria basat en IoT usant BLE per tal de valorar la possibilitat i dificultat d'implementar mesures de seguretat en aquest tipus de sistemes. Per fer-ho s'usa Arduino i Raspberry Pi ja que són dos plaques de baix cost amb un bon rendiment i reduïdes dimensions.

Per dur a terme el projecte s'ha investigat els protocols que intervenen en les comunicacions BLE així com les llibreries disponibles per les plataformes seleccionades en l'àmbit de les comunicacions BLE, la criptografia asimètrica, etc.

Un cop implementat el prototip de sistema de telemetria s'ha arribat a la conclusió de que sí que és possible implementar mesures de seguretat en els dispositius IoT actuals tot i les seves limitacions de processament.

Abstract (in English, 250 words or less):

Nowadays IoT devices are widespread in various areas of our daily lives but many manufactures neglect or underestimate the importance of security measures in these devices putting our privacy at risk.

While some manufactures implement appropriate security measures in their IoT devices, there are others that do not implement any security measures.

This paper aims to implement a prototype of an IoT-based telemetry System using BLE in order to assess the possibility and difficulty of implementing security measures in such Systems. Arduino and Raspberry Pi are used to do this as they are two low-cost boards with a good performance and small dimensions.

To carry out the project, the protocols involved in BLE Communications as well as the libraries available for the selected platforms in the field of BLE communications, asymmetric cryptography, etc have been investigated.

Once the telemetry system prototype has been implemented, it has been concluded that it is possible to implement security measures on current IoT devices despite their processing limitations.

Abstract

Nowadays IoT devices are widespread in various areas of our daily lives but many manufactures neglect or underestimate the importance of security measures in these devices putting our privacy at risk.

While some manufactures implement appropriate security measures in their IoT devices, there are others that do not implement any security measures.

This paper aims to implement a prototype of an IoT-based telemetry System using BLE in order to assess the possibility and difficulty of implementing security measures in such Systems. Arduino and Raspberry Pi are used to do this as they are two low-cost boards with a good performance and small dimensions.

To carry out the project, the protocols involved in BLE Communications as well as the libraries available for the selected platforms in the field of BLE communications, asymmetric cryptography, etc have been investigated.

Once the telemetry system prototype has been implemented, it has been concluded that it is possible to implement security measures on current IoT devices despite their processing limitations.

Resum

Avui dia els dispositius IoT estan molt estesos per diversos àmbits de la nostra vida quotidiana però molts fabricants descuiden o infravaloren la importància de les mesures de seguretat en aquests dispositius posant en risc la nostra privacitat.

Tot i que hi ha fabricants que implementen mesures de seguretat adequades en els seus dispositius IoT, n'hi ha d'altres que no implementen cap tipus de mesura de seguretat.

En aquest treball es pretén implementar un prototip d'un sistema de telemetria basat en IoT usant BLE per tal de valorar la possibilitat i dificultat d'implementar mesures de seguretat en aquest tipus de sistemes. Per fer-ho s'usa Arduino i Raspberry Pi ja que són dos plaques de baix cost amb un bon rendiment i reduïdes dimensions.

Per dur a terme el projecte s'ha investigat els protocols que intervenen en les comunicacions BLE així com les llibreries disponibles per les plataformes seleccionades en l'àmbit de les comunicacions BLE, la criptografia asimètrica, etc.

Un cop implementat el prototip de sistema de telemetria s'ha arribat a la conclusió de que sí que és possible implementar mesures de seguretat en els dispositius IoT actuals tot i les seves limitacions de processament.

Índex

Figures i taules

Llista d'imatges, taules, gràfics, diagrames, etc., numerades, amb títols i les pàgines en les quals apareixen.

Índex de figures

Índex de taules

1. Introducció

1.1. Introducció/Prefacio

Als últims anys s'ha popularitzat el concepte de Internet of Things (IoT), és a dir, l'internet de les coses i cada dia hi ha més dispositius o objectes interconnectats ja sigui de forma directa, a través d'una xarxa privada o través d'una connexió a internet per tal d'oferir serveis o funcionalitats més complexes i automatitzades als seus usuaris.

Aquesta revolució de l'Internet of Things ha sigut possible gràcies a l'avenç en la tecnologia que ha permès fabricar sensors i dispositius amb preus i consums energètics més baixos així com de reduïdes dimensions.

Tot i la popularitat d'aquests dispositius els fabricants estan descuidant la seguretat i privacitat d'aquests, havent-hi en el mercat dispositius sense cap tipus de mesura de seguretat i altres que en tenen però s'han detectat vulnerabilitats.

Aquesta falta de seguretat en els sensors i dispositius poden suposar l'accés a informació de caràcter personal per part de persones no autoritzades si es tracta de dispositius d'àmbit domèstic o en el cas de dispositius industrials pot provocar la interrupció de la producció o l'accés a secrets industrials i per tant pèrdues econòmiques.

1.2. Descripció/Definició

Cada dia tenim més dispositius intel·ligents, *wearables*, *SmartHome*, *SmartCity*, *SmartFactory*, etc, tots aquests conceptes estan relacionats amb l'Internet of Things, és a dir, un conjunt de dispositius interconnectats que es poden comunicar entre ells, enviar dades i que poden realitzar accions en funció de les dades rebudes.

El fet de que en el nostre dia a dia estiguem envoltats d'aquests dispositius fa que si no estan correctament dissenyats i tenen falles de seguretat poden afectar a la nostra privacitat.

En un sistema IoT format per un conjunt de nodes interconnectats és important impedir que algú no autoritzat pugui extreure informació de la xarxa així com comprovar que la informació que circula per la xarxa prové d'un node autoritzat i que aquesta no ha estat manipulada. Ja que si no es podria produir un comportament no desitjat per part d'alguns dels nodes del sistema.

El problema a l'hora d'establir una comunicació segura en un sistema de telemetria és que els sensors i la resta de dispositius IoT del sistema no acostumen a tenir una gran capacitat de processament. Això fa impossible o dificulta usar alguns algoritmes d'encriptació que siguin computacionalment costosos.

Durant el projecte es desenvoluparà un sistema de telemetria basat en Iot format per tres balises construïdes amb una placa Arduino amb sensors d'orientació i acceleració i que podran enviar informació del seu estat així com la informació captada amb els seus sensors a un sistema central desenvolupat amb una Raspberry Pi.

Una balisa és un dispositiu que transmet un senyal de baixa energia amb un identificador únic als dispositius propers. Juntament amb el seu identificador la balisa transmet uns quants bytes que poden ser usats per transmetre informació útil.

Figura 1: Dos balises BLE

L'objectiu és que aquest sistema de telemetria sigui segur, de tal manera que les dades enviades per les balises no puguin ser interceptades. És a dir, cap persona no autoritzada ha de poder interpretar o modificar aquestes dades. A més el sistema només ha d'acceptar dades de dispositius autoritzats, rebutjant dades des d'orígens desconeguts.

1.3. Objectius generals

L'objectiu principal d'aquest TFM és el disseny i implementació d'un sistema IoT de telemetria amb balises que puguin enviar informació del seu estat a un sistema central. Per tal, de dur-ho a terme serà necessària la cerca de documentació i informació sobre els sistemes de telemetria IoT actuals així com de papers científics sobre les tecnologies usades.

Aquesta cerca d'informació, a part d'un requisit per poder realitzar correctament aquest TFM, també serveix per complir l'objectiu d'obtenir un coneixement ampli de l'estat de l'art, adquirir coneixements sobre els principals atacs que poden patir i els protocols i mecanismes de seguretats usats en les connexions entre dispositius IoT per tal d'evitar-los.

A més, com la implementació es realitzarà usant Raspberry Pi i Arduino, també es té l'objectiu d'adquirir coneixements sobre aquestes dues plaques, les seves característiques, funcionament, etc.

1.3.1. Objectius principals

Objectius de l'aplicació/producte/servei:

- Entendre el funcionament de les balises BLE i la connexió entre central i perifèric de BLE.
- Identificar els principals factors de risc en la seguretat de sistemes IoT, incloent vulnerabilitats típiques dels sistemes Iot i els diferents tipus d'atacs que es realitzen contra ells.
- Estudiar els mecanismes de seguretat, protocols i algoritmes usats en el sistemes IoT per tal d'establir connexions segures.
- Dissenyar i implementar un sistema de telemetria segur basat en IoT format per un node central i tres balises BLE.
- Dissenyar i implementar un quadre de comandament on visualitzar la informació de telemetria.

Objectius personals de l'autor del TF:

- Conèixer les diferents tecnologies, protocols i dispositius que s'usen als ecosistemes IoT.
- Conèixer els algoritmes i sistemes de criptografia actuals.
- Adquirir coneixements sobre Raspberry Pi i Arduino, els llenguatges utilitzats per programar-les i les possibilitats que ofereixen.

1.4. Metodologia i procés de treball

Per tal de dur a terme aquest TFM es realitzarà una recerca d'informació sobre IoT i les tecnologies relacionades amb aquest. Els protocols usats en la comunicació entre els nodes de les xarxes IoT, sistemes d'autenticació i autorització, exemples de xarxes Iot amb sensors, etc.
Aquesta cerca es realitzarà tant a la Internet general com en llocs especialitzats amb publicacions científiques, papers, actes de conferències, etc.

A més de la recerca sobre IoT també es buscarà documentació sobre Raspberry Pi i Arduino per tal de tenir els coneixements necessaris per tal de desenvolupar un sistema de telemetria amb sensors basat en IoT.

Per últim, es desenvoluparà un sistema de telemetria basat en Iot format per tres balises BLE construïdes amb una placa Arduino amb sensors d'orientació i acceleració i que podran enviar informació del seu estat així com la informació captada amb els seus sensors a un sistema central desenvolupat amb una Raspberry Pi.

A aquest sistema central es podrà accedir amb un mòbil o ordinador i es mostrarà un quadre de comandament amb la informació rebuda de les balises.

En el desenvolupament de tot aquest sistema de telemetria es tindrà en compte la seguretat del mateix, aplicant els coneixements adquirits durant la recerca.

1.4.1. Scrum

Scrum és un metodologia de treball molt usada en el desenvolupament de software que aporta un marc de treball per tal de gestionar un projecte.
La paraula més important en la metodologia Scrum és el sprint, un sprint és un període de temps fix, que normalment comprén un parell de setmanes, en el que es realitzen les tasques.

Per treballar amb aquesta metodologia el primer que es fa és crear una llista amb totes les tasques que s'han de realitzar, aquesta s'anomena backlog.

Després es defineix la durada dels sprints i s'assignen les tasques del primer sprint seleccionant-les de entre les del backlog.

Durant el sprint es van fent les tasques i un cop acaba el sprint es realitza un sprint review on s'avaluen totes les tasques realitzades al llarg del sprint.

D'aquesta avaluació s'obté un feedback d'on poden sorgir noves tasques per tal d'arreglar, modificar o millorar parts realitzades a les tasques del sprint. Aquestes seran afegides al backlog, juntament amb la resta de tasques no realitzades en aquest sprint.

Un cop arribat a aquest punt, es defineix un nou sprint assignant-li tasques des del backlog i es repeteix el mateix procediment que en el sprint anterior.

En el cas d'aquest TFM es seguirà una metodologia Scrum amb sprints de dos setmanes ja que això permetrà tenir feedback abans de les entregues que són cada quatre setmanes i així tenir temps per poder corregir-ho o millorar-ho.

Un sprint més llarg tindria l'inconvenient de tenir poc marge per corregir el treball abans de l'entrega i un sprint més curt donaria poca flexibilitat en la distribució de les hores dedicades.

Un cop rebut el feedback es fragmentarà en tasques i s'afegirà en el backlog abans de plantejar les tasques del següent sprint.

1.5. Planificació

La quantitat d'hores que es destinaran a la recerca i desenvolupament d'aquest TFM és de 2h o 3h els dies laborables i 5h els festius i caps de setmana, això fa un total de 20h – 25h setmanals.

Aquests càlculs són una estimació ja que la dedicació real i la distribució d'hores dependrà dels problemes i dificultats que vagin apareixent durant el seu desenvolupament.

La llista de tasques i el diagrama de Gantt del projecte és el següent:

Activity	Inicio	Final	Días	
Entrega 1: Elaboració pla de treball	**16-09-20**	**29-09-20**	**14.0**	**28.0**
Anàlisi de l'estat de l'art	16-09-20	19-09-20	4.0	
Contextualització	20-09-20	21-09-20	2.0	
Definició dels objectius	22-09-20	23-09-20	2.0	
Definició de la metodologia de treball	24-09-20	25-09-20	2.0	
Planificació i elaboració del diagrama de Gantt del projecte	26-09-20	29-09-20	4.0	
Entrega 2: Recerca	**30-09-20**	**27-10-20**	**28.0**	**56.0**
Recerca tecnologies IoT	30-09-20	02-10-20	3.0	
Recerca sobre balises BLE	03-10-20	05-10-20	3.0	
Recerca informació sobre Arduino	06-10-20	06-10-20	1.0	
Recerca sobre balises BLE amb Arduino	07-10-20	11-10-20	5.0	
Recerca sobre sensors d'Arduino	12-10-20	12-10-20	1.0	
Recerca sobre seguretat en BLE amb Arduino	13-10-20	17-10-20	5.0	
Recerca sobre RaspBerry Pi	18-10-20	18-10-20	1.0	
Recerca Raspberry Pi i balises BLE	19-10-20	23-10-20	5.0	
Recerca seguretat a Raspberry Pi	24-10-20	27-10-20	4.0	
Entrega 3: Implementació	**28-10-20**	**24-11-20**	**28.0**	**56.0**
Instal·lació dels sensors a les plaques Arduino	28-10-20	28-10-20	1.0	
Programació dels Arduino per obtenir dades dels sensors	29-10-20	30-10-20	2.0	
Desenvolupament de balisa BLE amb Arduino	31-10-20	05-11-20	6.0	
Instal·lació i configuració Raspberry	06-11-20	07-11-20	2.0	
Desenvolupament codi per llegir informació de balises BLE	08-11-20	13-11-20	6.0	
Desenvolupament del quadre de comandament	14-11-20	21-11-20	8.0	
Testeig i correcció d'errors	22-11-20	24-11-20	3.0	
Entrega 4	**25-11-20**	**29-12-20**	**35.0**	**70.0**
Revisió de seguretat del sistema	25-11-20	04-12-20	10.0	
Finalitzar memòria del TFM	05-12-20	29-12-20	25.0	
				210.0

Figura 2: Llista de tasques

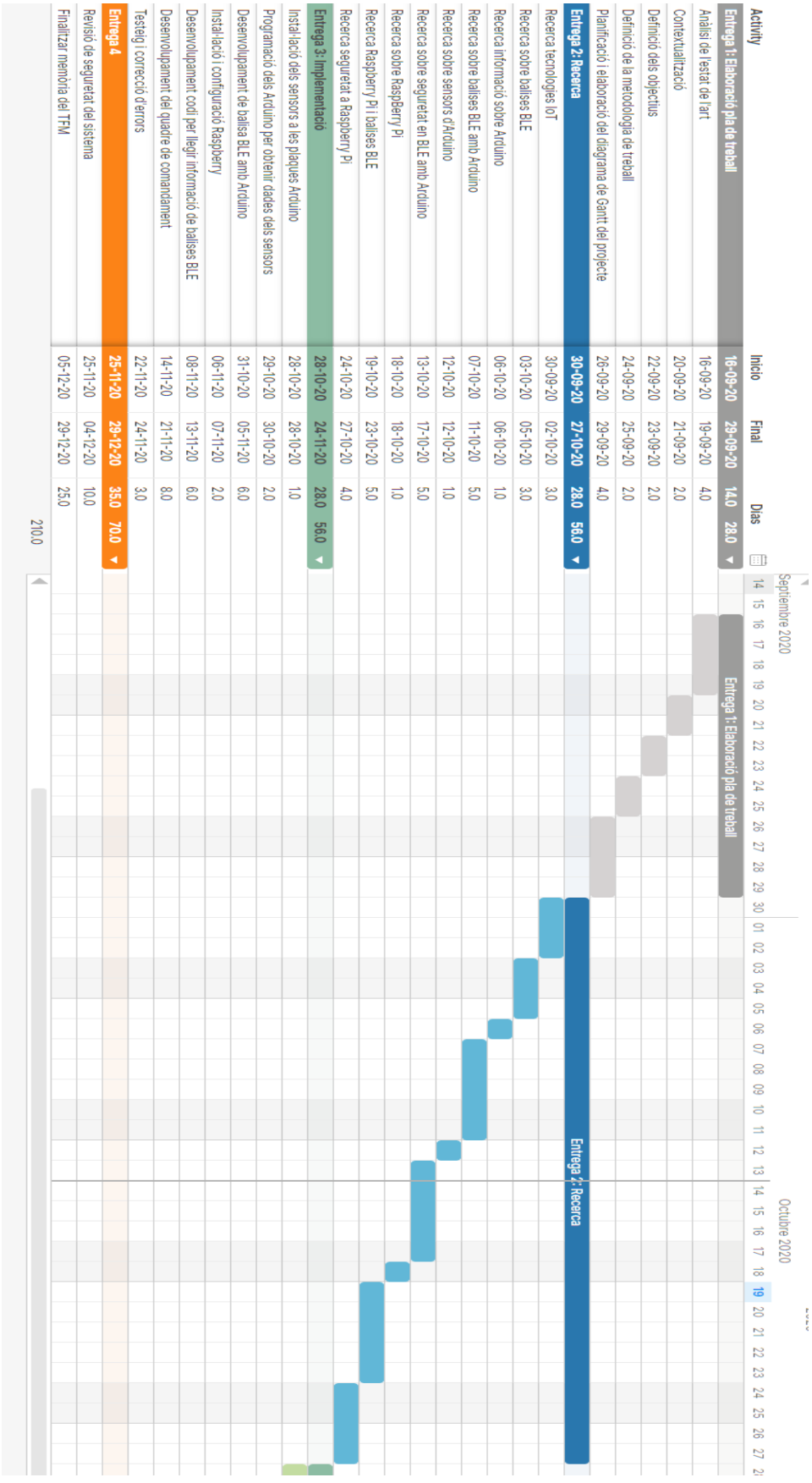

Figura 3: Gantt

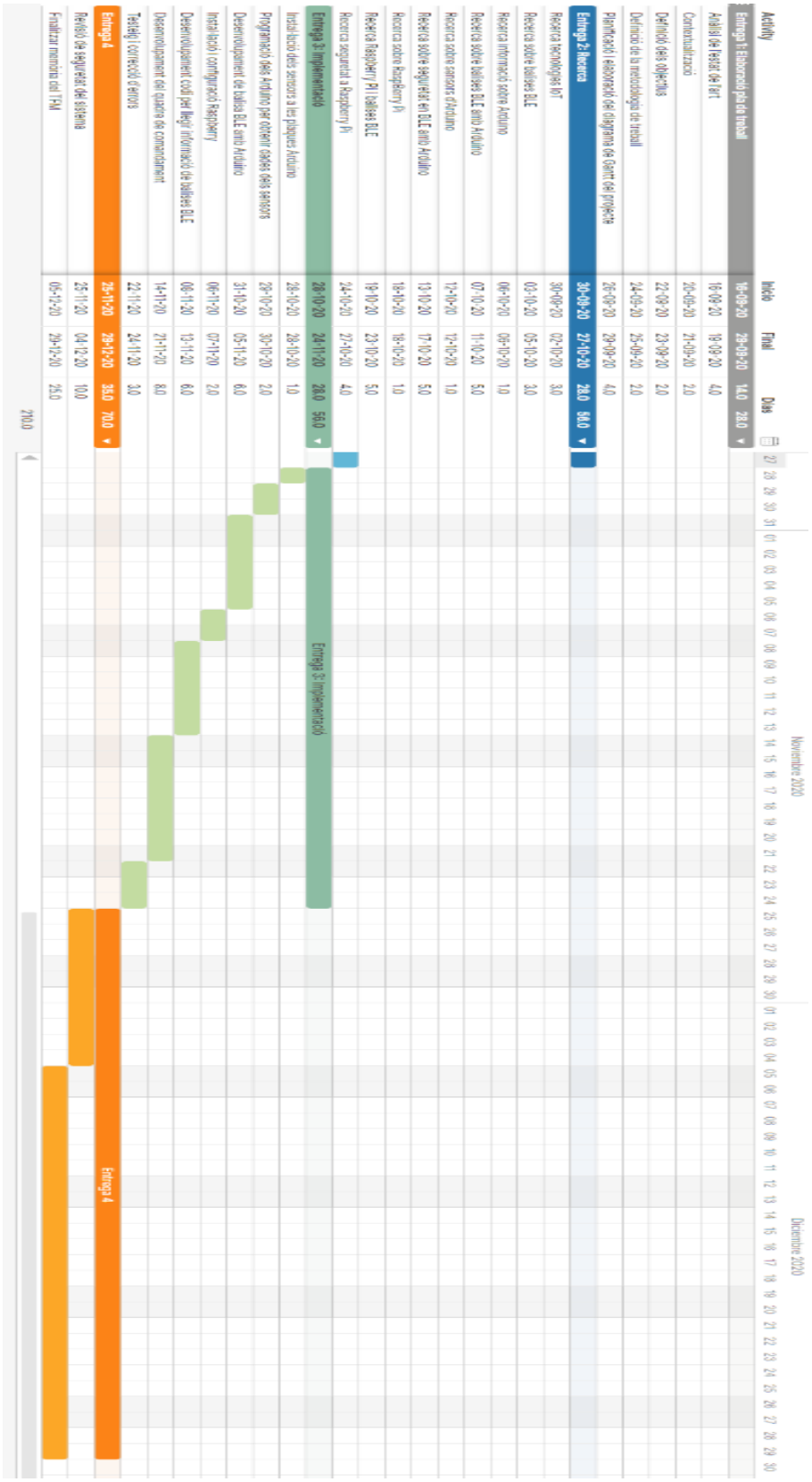

Figura 4: Gantt 2

En aquesta planificació s'ha tingut en compte els coneixements previs sobre alguns dels temes, per exemple, la recerca d'informació genèrica sobre Arduino i Raspberry Pi està planificada perquè duri només un dia per cadascun d'ells.

Tot i així s'ha considerat necessari incloure a la planificació la recerca d'informació sobre elles, ja que al llarg dels anys han patit molts canvis, han tret noves versions, etc. Per exemple, fa anys l'Arduino no tenia ni Bluetooth ni WiFi incorporats.
Per tot això, es requerirà temps de recerca especifica de les llibreries d'Arduino que hagin sorgit sobre BLE i de com usar-les per tal de fer-ho funcionar com a balisa BLE.

A més, un cop aprengui a fer una balisa BLE, hauré de fer que les connexió entre les balises i el node central sigui segura, per això també serà necessari invertir temps en la recerca d'informació sobre la seguretat en balises BLE i les seves comunicacions tant en general com en el cas concret de Arduino.

1.6. Anàlisi de riscos

En aquest apartat es descriuen alguns dels riscos que poden fer que el projecte es desviï de la planificació o no assoleixi algun dels objectius.

Problemes a l'hora de trobar informació
És possible que algun dels temes específics com balises BLE amb Arduino o la seguretat d'aquestes dos plataformes usades en el món del IoT no tingui prou documentació i costi trobar-la, dificultant la investigació i allargant el període de recerca més del que estava planejat.

Possible solució
Una possible solució és acotar la recerca dels diferents temes limitant-les temporalment al que esta planificat. Una altra solució és destinar el temps de recerca que no ha sigut necessari per investigar temes molt documentats o que finalment eren més senzills del que semblava a priori, i que per tant la seva planificació estava sobredimensionada, per investigar altres temes que requereixen més temps del planificat.

Problemes d'integració
En aquest projecte s'implementarà un sistema de telemetria usant diferents tecnologies i plataformes que s'han d'integrar i fer funcionar de forma conjunta, i a l'hora de fer-ho poden sorgir complicacions d'incompatibilitats o degut a no tenir els coneixements necessaris per interconnectar-les.

Possible solució
La forma d'evitar aquest risc és investigar bé tant les tecnologies com exemples d'interconnexió d'elles durant la fase de recerca.

Problemes amb el hardware

Per tal d'implementar el sistema de telemetria s'usaran plaques Arduino i Raspberry Pi i quan es treballa amb hardware poden aparèixer problemes relacionats amb aquest. Des de el mal funcionament de components electrònics per defectes de fàbrica a altres difícils de trobar explicació.

Possible solució

En el cas dels defectes de fàbrica, aquests no són molt comuns, però en el cas de que es donessin un cop detectat el motiu de l'error només caldria demanar una devolució. Per tal de poder seguir la planificació del projecte el més adient seria comprar de nou el component mentre encara es tramita la devolució del que falla.

1.7. Pressupost

El pressupost per tal de desenvolupar un prototip de sistema de telemetria IoT que consistirà en tres nodes capaços d'enviar la seva orientació, acceleració i inclinació, així com un node central que rebi totes aquestes dades és el següent:

Concepte	€/u	Quantitat	Total
Arduino MKR WiFi 1010	33,48 €	3	100,44 €
MPU-6050 Accelerometer + Gyro	2,996 €	3	8,99 €
Raspberry Pi 4 Modelo B / 4 GB SDRAM	58,44 €	1	58,44 €
Font d'alimentació USB-C 5.1V 3A	10,95 €	1	10,95 €
Tarjeta microSD 64GB	9,82 €	1	9,82 €
Batería lipo 3.7V 1100mAh	5,25€	3	15,75€
Total			**204,39 €**

Taula 1: Pressupost

En el pressupost s'inclouen tres Arduino MKR WiFi 1010 que són plaques Arduino de petites dimensions amb la possibilitat de connectar-les-hi una bateria lipo (inclosa en aquest pressupost) i que tenen WiFi i Bluetooth Low Energy incorporat el que les fa perfectes per fer-les servir com a balises BLE en aquest projecte. També es té en compte el preu de tres sensors MPU-6050 amb acceleròmetre i giroscopi per tal de que les balises puguin usar-los i transmetre la informació que obtenen.

Per últim, el pressupost inclou el preu d'una Raspberry Pi model 4 amb la seva font d'alimentació i tarjeta microSD per tal de fer-la servir com a node central. Aquest model és l'últim llançat i té WiFi i BLE incorporat. Aquest model de Raspberry Pi està disponible en tres versions amb diferent quantitat de memòria RAM, per aquest projecte s'ha triat la versió de 4GB de RAM per tal de que no es quedi curta de RAM però sense augmentar massa el pressupost.

1.8. Estat de l'art

A l'actualitat s'usen diferents tecnologies sense fils de baixa potencia dintre de l'ecosistema IoT com per exemple RFID, ZigBee, 6LoPan i Bluetooth Low Energy (BLE).

I d'entre aquestes tecnologies destacarem BLE ja que és de les més extensament usades amb milions de dispositius compatibles, per exemple tots els smartphones d'avui dia tenen BLE, a més a més, el seu baix consum energètic és perfecte per aplicacions Iot.

Tant el sector acadèmic com l'industrial han potenciat l'ús de BLE per aplicacions IoT usant balises BLE i existeixen molts casos d'us real d'aquestes en empreses.

Alguns casos d'ús de sistemes amb balises BLE són:

Un sistema de geomarketing, millorant l'experiència de dels compradors dintre d'un centre comercial, utilitzant les balises BLE per geolocalitzar els usuaris dintre del centre comercial i oferint-li promocions, descomptes i ofertes en funció de la seva ubicació per d'augmentar les vendes. Segons (Zaim & Bellafkih, 2016) la implementació d'aquest sistema usant BLE redueix el consum d'energia del sistema de monitorització a llarg termini.

Un sistema de museu intel·ligent que proveeix l'usuari d'informació addicional de les obres que està veient en el museu. Per aquest sistema s'usen balises BLE per obtenir la ubicació de l'usuari dintre del museu i un sistema de reconeixement d'imatges sobre les obres d'art per poder determinar quina informació proveir-li. Aquest cas d'us ha sigut analitzat per (Alletto, et al., 2016).

Un sistema de localització per interiors que et guiï per arribar des de la teva localització a un altre punt adaptat per cecs, persones amb discapacitat visual. La seva implementació usa balises BLE estratègicament col·locades i una aplicació que calcula la proximitat de l'usuari a cadascuna d'elles basant-se en la potencia de la senyal i així poder ubicar l'usuari.

Un sistema de gestió intel·ligent de l'energia per les oficines, que mitjançant balises BLE i una aplicació mòbil, pot detectar quan els usuaris entren i surten dels diferents espais de l'oficina, i així canviar el mode d'estalvi d'energia dels ordinadors, monitor i llums.

Un sistema de gestió de magatzems que usa un sistema de geolocalització usant iBeacons per triangular la posició d'objectes i així poder localitzar-los ràpidament dintre del magatzem. Això redueix els temps dels processos de preparació de comandes i de comprovació de stock.

A més algunes empreses han desenvolupat els seus propis estàndards com l'Eddystone de Google, l'iBeacon d'Apple, el LINE beacon de LINE, etc.

iBeacon és un protocol desenvolupat per Apple basat en BLE que va ser anunciat al 2013 i des de llavors diversos fabricants han desenvolupat balises que l'implementen. Aquests dispositius transmeten un identificador únic juntament amb uns quants bytes més usant BLE. Aquesta informació transmesa pot ser captada per dispositius compatibles.

Eddystone, llançat al 2015 per Google, és un altre protocol basat en BLE similar a iBeacon però en aquest cas de codi obert i multiplataforma, és capaç de fer tot el que permet fer iBeacon, i a més proporciona als desenvolupadores una sèrie d'APIs per que sigui més fàcil contextualitzar la informació enviada afegint-li informació del món real.

Figura 5: Balisa que implementa Eddystone

Si parlem dels aspectes relacionats amb la seguretat, tots els sistemes IoT han de procurar mantenir segures les comunicacions entre els seus dispositius, i actualment hi ha diferents propostes a l'hora de fer segura una connexió en una xarxa IoT.

Timothy Claeys et al. (Claeys, Rousseau, & Tourancheau, 2017) proposen usar un sistema d'autorització i autenticació basat en tokens, amb una infraestructura de clau pública per tal de crear una cadena de confiança entre els dispositius i així simplificar l'intercanvi dels tokens.

Swati Kinikar et al. (Kinikar & Terdal, 2016) proposen usar un token OAuth per autenticar els dispositius IoT.

Özlem Yerlikaya et al. (Yerlikaya & Dalkiliç, 2018) proposen usar contrasenyes d'un sol ús basades en HMAC (HOTP) per a l'autenticació.

Krishna Shingala (Shingala, 2019) proposa usar JSON Web Token (JWT) per autenticar els dispositius.

Aldar C. F. Chan et al. (Chan, Wong, Zhou, & Teo, 2016) proposen un esquema amb segon factor d'autenticació escalable, usant com a primer factor una clau secreta per tal d'identificar el dispositiu i l'històric de dades com a segon factor d'autenticació per tal de verificar l'autenticitat del dispositiu.

2. Recerca

2.1. Balises BLE

Un balisa BLE és un dispositiu BLE que transmet petites peces d'informació com per exemple la temperatura, pressió atmosfèrica, humitat, localització, orientació, acceleració, rotació, etc.
Degut a que usen la tecnologia de Bluetooth Low Energy tenen un molt baix consum energètic, hi ha balises dissenyades perquè les seves bateries arribin a durar anys.

2.1.1. Canals a les comunicaciones BLE

A les comunicacions entre dispositius BLE, els dispositius s'envien paquets d'informació a través de diferents tipus de canals, els canals d'advertising i els canals de dades.

Cada tipus de canal té la seva funció, per exemple, els canals d'advertising s'usen per descobrir els dispositius de l'entorn, establir connexions i per transmissions de broadcast.

Mentre que els canals de dades s'usen per comunicacions bidireccionals entre dispositius connectats i pel salt de freqüència adaptatiu.

2.1.2. Estructura dels paquets BLE

Els paquets del protocol BLE estan estructurats d'una manera determinada segons les especificacions del protocol.

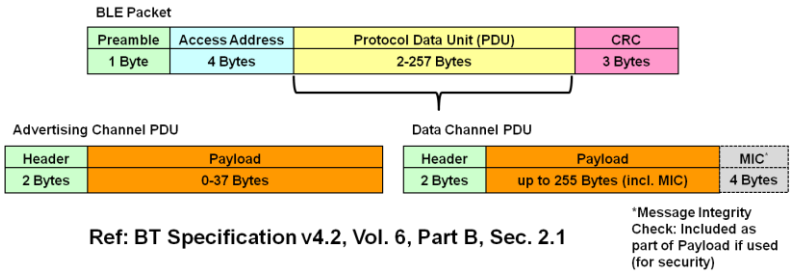

Figura 6: Estructura paquets BLE

Com es pot veure a l'esquema de la imatge anterior un paquet BLE comença amb un preàmbul, l'adreça d'accés i després va el Protocol Data Unit (PDU) que és la informació que es vol transmetre i per últim el CRC per verificar que no hi ha hagut errors durant la transmissió.
El camp PDU dels paquets BLE depèn del tipus de paquet, existeixen set tipus de paquet que poden ser enviats pels canals d'advertising i un tipus de paquet pels canals de dades.

L'estructura genèrica d'un paquet d'advertising és la següent:

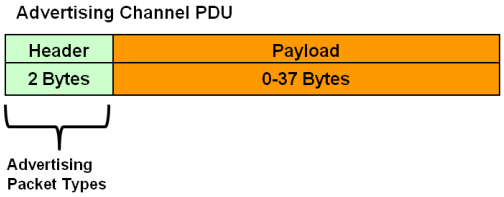

Figura 7: Estructura paquet d'advertising

Existeixen set tipus diferents de paquets d'advertising, cadascun amb una variació de l'estructura del payload.

En canvi, l'estructura d'un paquet de dades és la següent:

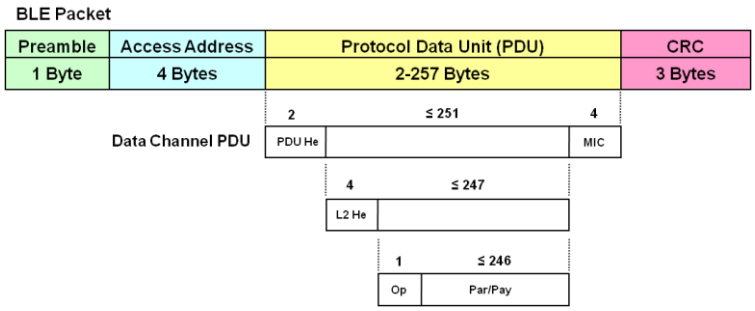

Figura 8: Estructura paquets de dades

Dintre del PDU conté la capçalera PDU, la capçalera L2CAP, el codi d'operació, paràmetres i payload. Pot incloure també el camp MIC (Messatge Integrity Check), com a part del payload, per verificar la integritat del missatge.
Aquest tipus de paquet s'usa per les comunicacions bidireccionals entre dispositius connectats.

2.1.3. Rols GAP

Els dispositius BLE poden tenir diferents comportaments segons el rol o rols amb el que estiguin funcionant. Existeixen quatre rols diferents a la capa GAP (Generic Access Profile): perifèric, central, broadcaster o observador. És important destacar que un mateix dispositius pot estar funcionant amb més d'un rol al mateix temps.

El parell de roles broadcaster/observador són exemples de comunicació d'una direcció, els dispositius amb el rol de broadcaster van enviant contínuament paquets d'advertising a intervals regulars amb la informació que volen transmetre per aquells dispositius que els vulguin escoltar.

A l'altre costat de la comunicació es troben els dispositius amb el rol d'observadors que estan pendents dels paquets que transmeten els broadcasters. Poden haver-hi un o més dispositius amb el rol d'observador escoltant un mateix broadcaster tal i com es pot veure a la imatge següent.
Així mateix, un observador pot rebre els paquets de més d'un broadcaster.

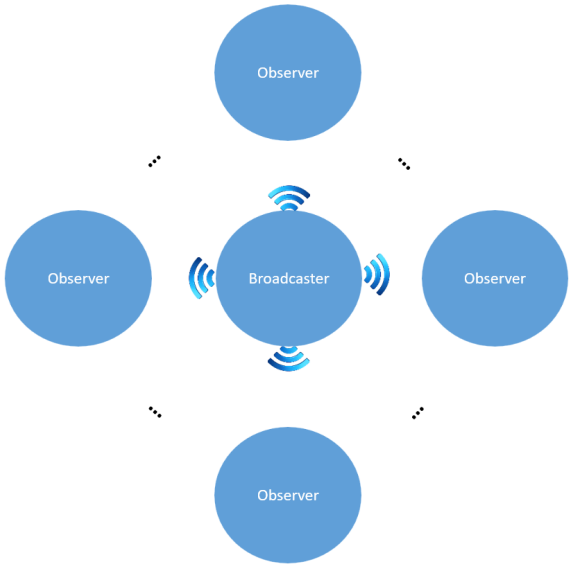

Figura 9: Comunicació unidireccional Broadcaster/Observador

Tant els dispositius amb el rol de broadcaster com els que tenen el rol d'observador no accepten connexions.

Els dispositius amb el rol de broadcaster poden arribar a tenir una duració de bateria molt llarga, ja que poden estar en un estat d'hibernació, despertar per transmetre la informació cada cert període de temps i tornar a hibernar.

Per altre banda, a BLE s'estableixen connexions bidireccionals de dades entre dispositius amb els rols de perifèric i central. En el cas del rol de perifèric, el dispositiu envia paquets d'advertising per tal de ser descobert pels dispositius amb el rol de central i accepta connexions en les que es comporta com a dispositiu esclau (de la capa d'enllaç de dades).

Els dispositius amb rol de central escanegen els seu entorn per descobrir els dispositius BLE amb el rol de perifèric a partir dels paquets d'advertising que envien aquests i un cop descoberts poden establir una connexió amb ells per tal de començar una comunicació bidireccional on es comporten com a dispositius master (de la capa d'enllaç de dades).

Figura 10: Comunicació bidireccional Central/Perifèric

Una mateixa central pot connectar-se a més d'un dispositius perifèric rebent informació de tots ells.

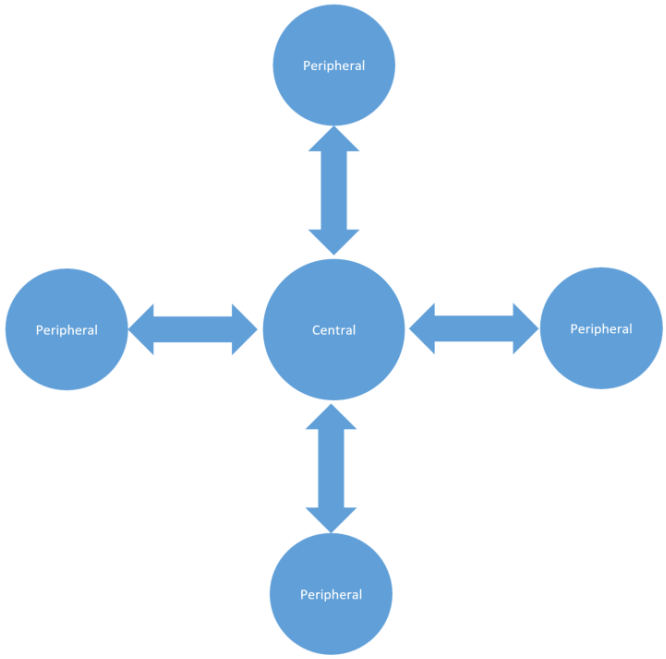

Figura 11: Comunicació bidireccional Central/Perifèrics

Després de veure els diferents rols que hi ha a GAP es pot veure com els que més s'adeqüen a una balisa BLE són els de perifèric i broadcaster. I en el cas d'aquest projecte el rol que assumiran les balises serà el de perifèric.

2.1.4. GATT

GATT (Generic Attribute Profile) permet la gestió d'informació organitzada en perfils, serveis i característiques.

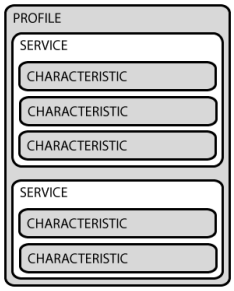

Figura 12: GATT

A la imatge anterior es pot veure com s'organitza la informació, els perfils contenen un o més serveis i alhora cada servei conté una o més característiques.

Existeixen perfils i serveis oficialment adoptats, que serien un estàndard, per certes funcionalitats però es poden definir perfils i serveis personalitzats amb les característiques que fossin necessàries per un projecte o funcionalitat especifica.

Un servei es pot distingir d'un altre a través d'un ID únic anomenat UUID que són de 16 bits pels oficials i 128 bits pels personalitzats.

Les característiques encapsulen dades i es pot distingir una característica d'una altra de la mateixa manera que amb els serveis, cadascuna té un UUID de 16 bits o 128 bits.

A més, les característiques poden ser només de lectura o també d'escriptura, permetent una comunicació bidireccional.
En el context d'aquest projecte les balises tindran la informació dels seus sensors, de la bateria, del temps que porten en funcionament i demés informació que volen servir organitzada d'aquesta manera per tal de que la central pugui obtenir-la.

2.2. Seguretat en BLE

Existeixen molts tipus d'atacs diferents que un dispositius BLE pot patir, i per tant una balisa BLE, en aquest apartat es mencionaran els més comuns i maneres de mitigar-los tenint en compte les mesures de seguretat de les que pot disposar un dispositius BLE.

2.2.1. Tipus d'atacs

Sniffing
En aquest tipus d'atac, l'atacant examina la informació enviada pels canals d'advertising usats pels dispositius que implementen BLE per tal de trobar-se entre ells.

Man in the middle
Aquests tipus d'atacs consisteix en que l'atacant es situa en mig de la comunicació manipulant els missatges entre les parts. Les dos parts es pensen que s'estan comunicant entre elles però realment estant comunicant-se amb l'atacant que reenvia els missatges rebuts a l'altra part modificant-los prèviament.

Spoofing
En aquests atacs l'atacant imita l'adreça MAC o el UUID d'un dispositius BLE per tal de suplantar-lo.

Denial of service

Aquests tipus d'atacs són bastant coneguts ja que es donen molt a l'àmbit web en la seva modalitat DDoS (Distributed Denial of Service), en el cas del BLE com que els dispositius BLE d'IoT normalment estan dissenyats per connectar-se només a un dispositius master alhora, l'atacant bombardeja el dispositius BLE amb sol·licituds de connexió evitant que usuaris legítims puguin connectar-se.

Hijacking

Aquest atac consisteix en el segrest d'una comunicació legitima per part d'algú no autoritzat. Per exemple, en el cas d'un telèfon intel·ligent en el que tenim una aplicació per connectar-nos a un perifèric BLE, l'aplicació establirà una comunicació segura amb el perifèric però un cop establerta, l'atac de hijacking es produiria quan una altra aplicació que també estigui instal·lada en el dispositiu, de manera fraudulenta, segresta les credencials, qualsevol tipus de token o aprofitant una vulnerabilitat aconsegueix apropiar-se o usar la connexió prèviament establerta amb el perifèric.

2.2.2. Mitigació

Per tal d'establir connexions segures en l'àmbit de les balises BLE podem prendre algunes mesures com per exemple enviar la informació encriptada de tal manera que només els dispositius autoritzats puguin accedir a la informació.

Enviant la informació encriptada podem mitigar els atacs dels tipus sniffing i man in the middle, ja que encara que la informació sigui escoltada per tercers no autoritzats no podran desencriptar-la sense la clau. Tot i així, s'ha de procurar fer l'intercanvi de claus de forma segura, ja que si es transmeten les claus de forma insegura pel mateix canal que l'atacant està escoltant, aquest podrà obtenir-la i trencar l'encriptació.

Tot i l'encriptació encara es podria patir un atac de spoofing, en aquest cas, es podria evitar fent que el UUID sigui dinàmic. D'aquesta manera un atacant no pot reutilitzar un UUID ja usat i s'evita que pugui fer-se passar per la balisa autentica.

Respecte al Hijacking, per evitar-ho cal tenir un control del software i configuracions del dispositiu central evitant instal·lar aplicacions o software no fiable.

2.3. Arduino

Com que en aquest projecte s'usaran plaques Arduino per tal de muntar balises BLE pel sistema de telemetria IoT, aquesta secció pretén ser una petita introducció al que és Arduino per tal de posar una mica en context la placa Arduino que finalment s'usarà.

Arduino és una plataforma d'electrònica de hardware obert i software obert fàcil d'usar. A l'actualitat s'usa en molts àmbits diferents, en l'àmbit educacional es utilitzada a les escoles per introduir en el món de l'electrònica als seus estudiants, també és usat per programadors i entusiastes de l'electrònica per desenvolupar els seus projectes personals de forma fàcil i ràpida, en l'àmbit professional ha sigut emprat com a cervell de multitud de projectes i instruments científics.

Algunes de les característiques que fan que sigui tant popular són:

- **El seu baix cost**: En el mercat hi ha més plaques similars però Arduino té un cost més baix. D'entre els diferents models d'Arduino que hi ha, el més barat és un que has d'acoblar tu mateix.
- **Multiplataforma**: L'IDE d'Arduino està disponible per Windows, MacOS i Linux, a diferencia de moltes de les altres plataformes que només estan disponibles per Windows.
- **Entorn de programació simple**: L'IDE d'Arduino és fàcil d'usar per persones principiants però alhora suficientment flexible per programadors avançats. Està basat en l'entorn de programació de Processing cosa que és molt convenient pels professors, ja que als estudiants que estan aprenent a programar amb aquest entorn els hi resultarà molt familiar.
- **Codi obert i extensible**: El software d'Arduino està publicat sota una llicencia de codi obert, el que permet a programadors amb experiència estendre'l mitjançat llibreries en C++ per tal de proporcionar-li noves funcionalitats.
- **Hardware obert:** Els plànols de les plaques Arduino es publiquen sota una llicencia de Creative Commons de tal manera que dissenyadors de circuits poden fer la seva pròpia versió, fent-hi modificacions per tal de millorar-ho.

2.3.1. Plaques genèriques

Hi ha multitud de plaques Arduino, algunes més genèriques i altres especialitzades per a coses més concretes.

Un exemple de placa genèrica seria l'Arduino UNO, que és una de les millors per introduir-se en el món de l'electrònica i la programació. Va per la revisió número 3, que té 14 pins digitals d'entrada/sortida, sis dels quals poden ser usats com a sortida PWM i 6 pins d'entrada analògics. Té una memòria flash de 32 KB dels quals 500 bytes són usats pel bootloader. Pesa 25 grams i mesura 68,6mm de llargada i 53.4mm d'amplada.

Figura 13: Arduino UNO Rev3

Una altra placa genèrica molt popular i més potent que l'Arduino UNO és l'Arduino Mega 2560 Rev3, aquesta placa compta amb 54 pins digitals d'entrada/sortida, 15 dels quals poden ser usats com a sortides PWM i 16 pins d'entrada analògics. Aquesta gran quantitat de pins juntament amb un millor processador el fan perfecte per projectes més ambiciosos, que siguin més complexes o requereixin més potencia. A més, t'he l'avantatge respecte altres plaques, que és compatible amb la majoria de shields dissenyats per l'Arduino UNO.

Figura 14: Arduino Mega 2560 Rev3

Existeix també una versió de l'Arduino UNO amb WiFi anomenada Arduino UNO WiFi que va per la seva segona versió. I que és molt similar a l'Arduino UNO normal però amb una millor connectivitat ja que disposa de WiFi i BLE tot i que no es poden fer servir tots dos alhora.

Figura 15: Arduino UNO WiFi Rev2

I encara que no es pretén fer un llistat de totes les plaques Arduino ja que n'hi ha moltes, cal mencionar l'Arduino Zero que destaca per sobre de les dos anteriors per tenir un processador de 32 bits.

Figura 16: Arduino Zero

Encara que per aquest projecte es podrien fer servir qualsevol placa genèrica, encara que afegint-hi algun shield de BLE a les que no ho tinguin incorporat hi ha plaques que s'adapten millor a la finalitat per la que es faran servir en aquest projecte que és crear una balisa BLE.

2.3.2. Shields

Els shields són plaques modulars que es munten a sobre de les plaques Arduino per les que han estat dissenyades i li afegeixen noves funcionalitats. Hi ha multitud de shields i cadascun té la seva documentació, ja que tot i que alguns estan dissenyats de tal manera que només hi ha una manera d'encaixar-ho sobre els altres shields o sobre l'Arduino, hi ha altres que són una mica més complexes de muntar o que tenen components o els pins sense soldar.

A més, a la documentació es pot trobar informació sobre quins pins de l'Arduino queden inutilitzats degut a que són usats pel shield o si usa un bus i quins requisits té per tal d'usar-ho, també a la documentació s'acostuma a trobar un exemple d'us o una llibreria per tal de facilitar el seu us.

Existeixen tot tipus de shields que afegeixen noves funcionalitats com: WiFi, BLE, GPS, GSM, sensors, pantalles LED, pantalles TFT, control de motors, control de relés, etc.
Alguns d'aquests són oficials fabricats per l'empresa d'Arduino i altres fabricats per altres empreses.
Per exemple, l'Arduino Ethernet Shield 2 pot proporcionar accés a internet a través d'un cable d'ethernet a una placa Arduino compatible.

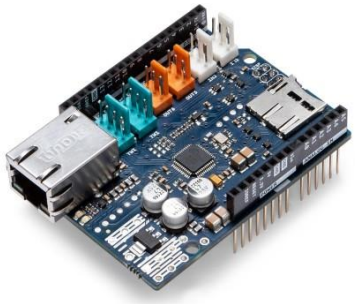

Figura 17: Arduino Ethernet Shield 2

El Arduino MKR ETH shield és un altra shield que proporciona una connexió a internet a través de cable d'ethernet però aquest cop de la família MKR que són molt més petits ja que estan pensats per projectes IoT.

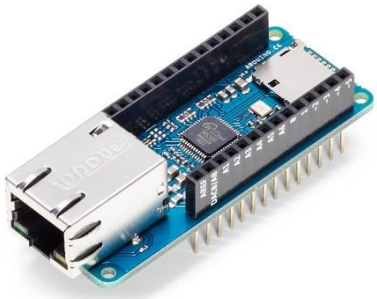

Figura 18: Arduino MKR ETH Shield

També existia l'opció d'afegir-li WiFi a un Arduino amb el Arduino WiFi Shield, però aquest va ser retirat del mercat, potser degut a que a l'actualitat ja existeixen moltes opcions de plaques amb WiFi incorporat i que per tant si aquesta connectivitat és un requeriment del projecte es pot usar una d'aquestes, evitant les que no ho tenen incorporat de sèrie.

Figura 19: Arduino WiFi Shield

Tot i així, encara existeixen multitud de shields fabricat per tercers que afegeixen la connectivitat WiFi a Arduino. Com per exemple la companyia de hardware obert Adafruit.

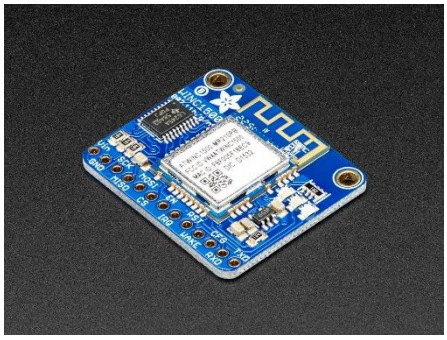

Figura 20: Adafruit WiFi

2.3.3. Plaques especialitzades

Hi ha plaques Arduino especialitzades per finalitats concretes com l'Arduino MKR GSM 1400 que ve preparat per projectes que requereixin una connexió GSM/3G. Les característiques que el fan perfecte per aquest tipus de projectes, a part del mòdul de GSM/3G, són les seves reduïdes dimensions, el seu processador de baix consum i la possibilitat de connectar-li una bateria lipo.

Existeixen plaques especialitzades en àmbits diferents, algunes pel control de motors, altres per ser programades en llenguatges d'alt nivell, altres per tractar amb sensors, IoT, etc. D'entre les quals pel cas que pertoca a aquest projecte les que interessen són les especialitzades en IoT.

L'Arduino MKR GSM 1400 mencionat en el primer paràgraf d'aquesta secció també seria perfecte per un projecte d'IoT o també ho serien plaques amb mòduls que proporcionen connectivitat Narrowband que funcionen a les bandes de IoT LTE o altres que tenen mòduls de connectivitat Lo-Ra, però pel cas concret d'aquest projecte dintre de la família de plaques Arduino especialitzades en IoT ens fixarem en les que tenen connectivitat BLE.

L'Arduino Nano 33 IoT pesa 5 grams, té unes dimensions de només 45mm de llargada per 18mm d'amplada, connexió WiFi i BLE, té un consum molt baix i funciona a 3.3V i menys de 20mA.

L'Arduino MKR1000 és una combinació de les funcionalitats de l'Arduino Zero i el WiFi Shield, és una mica més gran que l'Arduino Nano 33 IoT, pesa 32 grams i mesura 61,5mm de llarg per 25mm d'ample. Tot i tenir unes dimensions una mica més grans t'he l'avantatge de que se li pot connectar una bateria lipo, però com a desavantatge aquest no té BLE.

Una variant millorada de l'Arduino MKR1000 és l'Arduino MKR WiFi 1010, té les mateixes dimensions i pes que el seu predecessor però amb un nou chipset que d'entre altres millores li proporciona connectivitat BLE.

Aquestes característiques fan l'Arduino MKR WiFi 1010 perfecte per aquest projecte ja que té unes mides reduïdes, té connectivitat BLE i la possibilitat de connectar-li una bateria, cosa desitjable en una balisa BLE.

Figura 21: Arduino MKR WiFi 1010

2.3.4. Seguretat

La placa d'Arduino MKR WiFi 1010 compta amb un mòdul criptogràfic, l'ATECC508A que té un emmagatzematge de claus de forma segura basat en hardware, un generador de números aleatoris i pot executar algoritmes de clau pública com ECDSA i ECDH aquestes i altres característiques criptogràfiques permeten establir connexions de forma segura.

2.4. Raspberry Pi

La Raspberry Pi és un microordinador de placa única de baix cost desenvolupat per la fundació Raspberry Pi, va per la seva cinquena versió anomenada Raspberry Pi 4.
En aquest projecte s'usarà una Raspberry Pi 4 de 4GB de RAM com a node central del sistema de telemetria que servirà un panell d'administració a l'usuari amb les dades recol·lectades de les balises BLE.

Aquesta versió compte amb WiFi i BLE connexions que seran molt útils per aquest projecte ja que s'usarà el BLE per obtenir la informació de les tres balises BLE i el WiFi serà usat per servir una web amb un panell d'administració amb les dades recol·lectades de les balises.

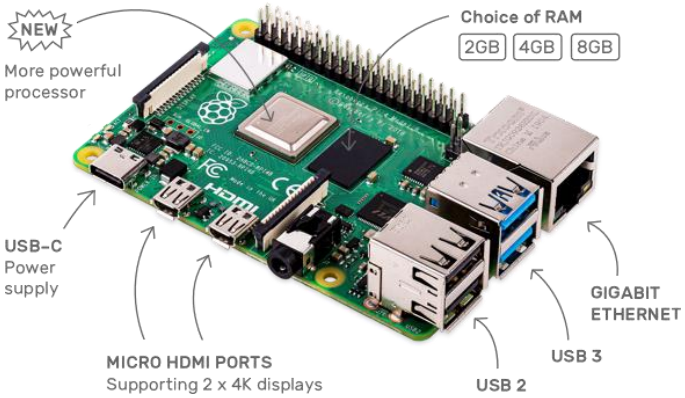

Figura 22: Raspberry Pi 4

3. Implementació

3.1. Implementació d'un sistema de telemetria

Com a part pràctica d'aquest TFM s'ha desenvolupat un prototip de sistema de telemetria amb balises BLE. El sistema consta de tres balises BLE implementades amb tres plaques Arduino MKR WiFi 1010 que tal i com s'ha exposat a la secció 2.3.3 té característiques que la fan perfecte per aquest fi, com per exemple: mides reduïdes, BLE incorporat i coprocessador criptogràfic.

Cadascuna de les plaques Arduino té connectat un mòdul GY-521 que incorpora una unitat de mesurament inercial MPU-6050, un sensor que compta amb un acceleròmetre i un giroscopi de tres eixos, i li proporciona la informació captada per aquests. Un cop la balisa obté la informació dels sensors l'envia a la central signada i encriptada.

Per altre banda, s'ha implementat la central amb una Raspberry Pi 4 que compta amb BLE incorporat de fàbrica amb el que pot escanejar el seu entorn en busca de dispositiu BLE, trobar les balises i connectar-se a elles per obtenir la informació que li enviïn.

Dintre de la Raspberry Pi s'ha instal·lat un servidor web apache que serveix un panell d'administració amb la informació obtinguda de les balises per tal de que pugui ser accedida des d'un ordinador o mòbil connectat a la Raspberry Pi a través d'una connexió WiFi.

3.1.1. Flux de dades

A la imatge següent es pot apreciar el flux de la informació des de que es captada pels sensors de les balises fins que arriba al panell d'administració que visualitza l'usuari.

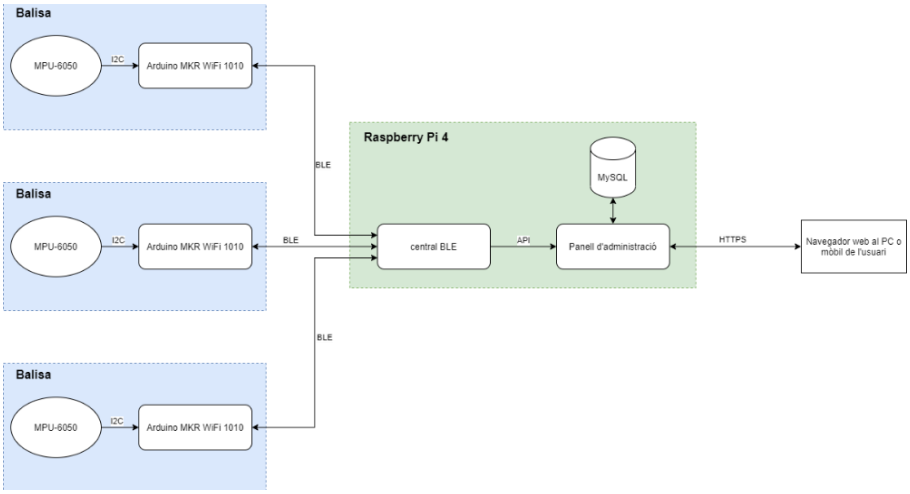

Figura 23: Flux de dades

Els sensors i l'Arduino que formen la balisa es comuniquen a través del protocol I2C un cop l'Arduino obté la informació del sensor el transmet a la central BLE a través d'una connexió que té establida amb aquesta mitjançant BLE.

La central BLE implementada dintre de la Raspberry Pi crida a una API del panell d'administració, també implementat dintre de la pròpia Raspberry Pi i aquesta API quan rep les dades de la central BLE les emmagatzema dintre d'una base de dades MySQL, també dintre la Raspberry Pi.

Quan l'usuari amb el navegador web del seu ordinador o mòbil accedeix al panell d'administració, servit per un servidor web Apache configurat dintre de la Raspberry Pi, aquest obté la informació de la base de dades i li serveix a l'usuari.

3.2. Especificacions dels diferents components

3.2.1. Arduino MKR WiFi 1010

Microcontroller	SAMD21 Cortex®-M0+ 32bit low power ARM MCU
Radio module	u-blox NINA-W102
Board Power Supply (USB/VIN)	5V
Secure Element	ATECC508
Supported Battery	Li-Po Single Cell, 3.7V, 1024mAh Minimum
Circuit Operating Voltage	3.3V
Digital I/O Pins	8
PWM Pins	13 (0 .. 8, 10, 12, 18 / A3, 19 / A4)
UART	1
SPI	1
I2C	1
Analog Input Pins	7 (ADC 8/10/12 bit)
Analog Output Pins	1 (DAC 10 bit)
External Interrupts	8 (0, 1, 4, 5, 6, 7, 8, 16 / A1, 17 / A2)
DC Current per I/O Pin	7 mA
CPU Flash Memory	256 KB (internal)
SRAM	32 KB
EEPROM	no
Clock Speed	32.768 kHz (RTC), 48 MHz
LED_BUILTIN	6
USB	Full-Speed USB Device and embedded Host
Length	61.5 mm
Width	25 mm
Weight	32 gr.

Taula 2: Especificacions tècniques Arduino MKR WiFi 1010

3.2.2. Mòdul GY-521

Chip	MPU-6050
Power supply	3-5v
Communication	standard IIC communication agreement
Gyroscope range	+ 250 500 1000 2000 ° /s
Acceleration range	± 2 ± 4 ± 8 ± 16 g
Size	21*15*1.2 mm(max)
Net weight	3 g

Taula 3: Especificacions tècniques mòdul GY-521

3.2.3. Raspberry Pi 4

Chip	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
GPU	VideoCore VI 500 MHz
RAM	4GB LPDDR4-3200 SDRAM
Wireles connectivity	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
Power supply	5V/3A USB-C, 5V GPIO
Storage	microSD
Other connectivity	Gigabit Ethernet; 2xUSB 3.0 ports; 2xUSB 2.0 ports; 2xmicro-HDMI ports (up to 4kp60 supported); 2-lane MIPI DSI display port; 2-lane MIPI CSI camera port; 4-pole stereo audio and composite video port

Taula 4: Especificacions tècniques Raspberry Pi 4

3.3. Muntatge de la balisa

L'Arduino MKR WiFi 1010 que s'ha adquirit ve ja completament muntat de fàbrica però en el cas del mòdul GY-521 MPU-6050 cal soldar-hi els pins a la placa per després poder connectar-ho a l'Arduino.

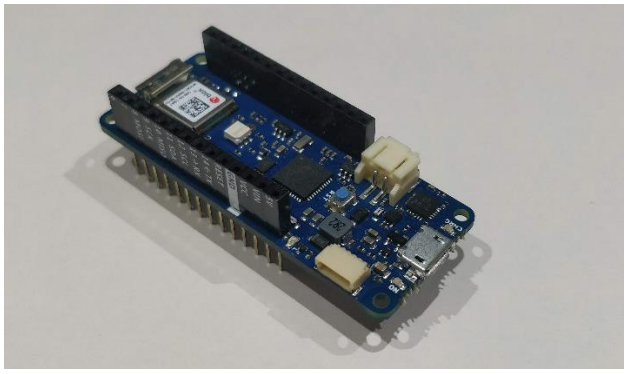

Figura 24: Arduino MKR WiFi 1010

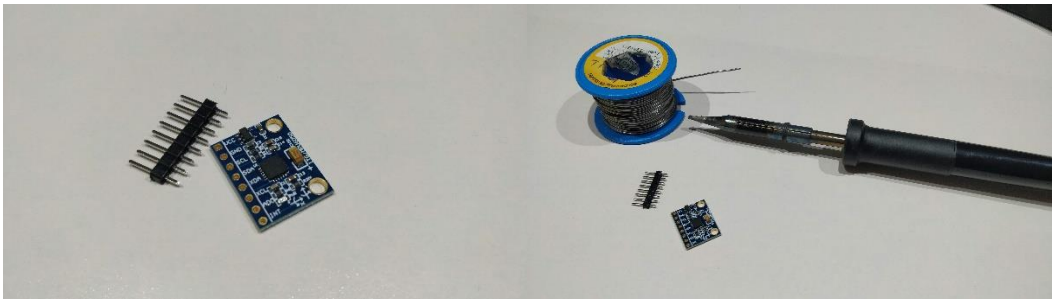

Figura 25: GY-521 MPU-6050 sense els pins soldats

Figura 26: Tres GY-521 MPU-6050 amb els pins soldats

Un cop el mòdul GY-521 MPU-6050 ja té els pins soldats al mòdul es poden realitzar les connexions amb l'Arduino, per fer-ho s'ha d'analitzar l'esquema de connexions de l'Arduino MKR WiFi 1010 i els pins que té el GY-521 per veure com s'han d'interconnectar.

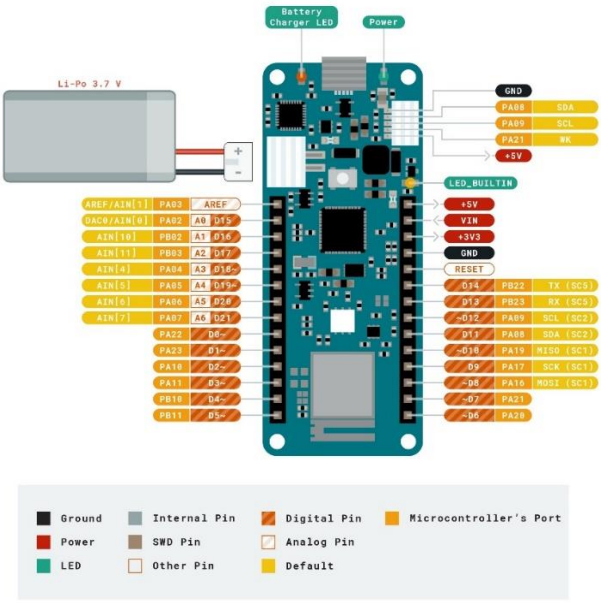

Figura 27: Esquema Arduino MKR WiFi 1010

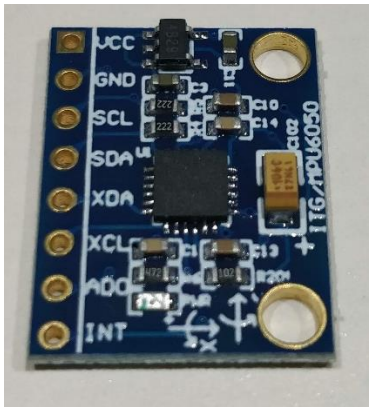

Figura 28: GY-521 MPU-6050

Observant els pins de tots dos components es pot apreciar com s'han de connectar, s'ha de connectar el pin SCL del mòdul amb el pin SCL de l'Arduino, també etiquetat com a pin PA09 o ~D12, el SCL és el pin de rellotge mitjançant el qual el mòdul i l'Arduino poden treballar amb els rellotges sincronitzats, després s'han d'interconnectar els pins SDA del mòdul i el pin SDA de l'Arduino, també etiquetat com a pin PA08 o D11, aquesta connexió és a través de la qual es passaran les dades i per últim, per tal d'alimentar el mòdul, s'interconnecten els pins GND de tots dos, que corresponen a la massa i es connecta el pin Vcc del mòdul al pin Vcc (+3V3) de l'Arduino o al pin 5V ja que el mòdul pot funcionar en un rang d'entre 3V i 5V.

El protocol que utilitzen l'Arduino i el mòdul per comunicar-se a través d'aquest esquema de connexions és I2C.

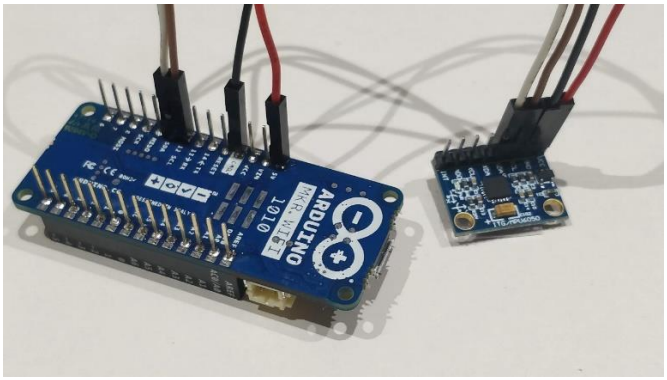

Figura 29: Balisa amb Arduino MKR WiFi 1010 i GY-521 MPU6050

3.4. Implementació balisa BLE

Per tal d'implementar una balisa BLE amb un Arduino s'ha fet servir la llibreria Arduino BLE, que és una llibreria desenvolupada per la pròpia companyia d'Arduino i permet usar fàcilment el mòdul BLE incorporat a l'Arduino.

La part de signatura/encriptació de la informació abans de ser enviada a la central es realitza usant la llibreria ArduinoECCX08 una altra llibreria desenvolupada per Arduino i que en aquest cas gestiona el mòdul criptogràfic i ens permet treballar amb certificats, algoritmes d'encriptació amb clau asimètrica, etc.

Per últim, per obtenir la informació dels sensors del mòdul GY-521 MPU-6050 s'ha usat la llibreria MPU6050, en aquest cas desenvolupada per Electronic Cat, i que permet obtenir la informació del l'acceleròmetre de tres eixos, el giroscopi de tres eixos i el sensor de temperatura sense la necessitat de treballar a baix nivell les comunicacions amb I2C.

3.4.1. Implementació balisa BLE amb Arduino

Usant les llibreries mencionades a l'apartat anterior s'ha desenvolupat el codi de la balisa BLE tal i com es pot veure a la secció 6.1. En ell s'han definit dos serveis, el primer un batteryService usant la definició estàndard amb el UUID "180F" i que contindrà una característica "2A19" que representa el nivell de bateria de la balisa, i un segon servei anomenat dataService que conté característiques amb la informació obtinguda dels sensors de la balisa.

La informació obtinguda per l'acceleròmetre de la balisa en els diferents eixos es guardada a la característica accelChar amb UUID "6666", aquesta característica és de només lectura i permet subscriure's per ser notificat quan el seu valor canviï.

La informació del giroscopi de la balisa es tractada de forma similar i guardada a la característica gyroChar amb UUID "5555" i també és de només lectura i permet subscriure's per rebre notificacions.

La última característica de dades que té aquest servei és statusChar, amb UUID "4444", que conté la informació del temps que porta en marxa la balisa expressat en mil·lisegons. El permisos d'accés d'aquesta característica són els mateixos que les dues anteriors, només lectura i notificacions.

A més de les tres característiques de dades aquest servei compta amb dues característiques usades per autenticar la central BLE de tal manera que només ella pugui accedir a la informació dels sensors.

Aquestes dues característiques són authChar i authStatusChar, amb UUID "0000" i "0001" respectivament, a la primera els dispositius només tenen permisos d'escriptura, no poden llegir el seu contingut, i es usada per la central BLE per escriure la contrasenya de la balisa i d'aquesta manera autenticar-se. La segona característica, authStatusChar és un boolean que indica si s'està ja autenticat o no i serveix perquè la central BLE pugui saber si en la connexió actual en la que es troba amb la balisa ja està autenticada o no.

S'estableixen dos handlers per tal de detectar la connexió i desconnexió de dispositius remotes a la balisa. El handler que controla la connexió d'un dispositius només és usat per a finalitats de debug però en el handler que s'executa en l'esdeveniment de desconnexió a part de informació de debug també es restauren els valors de les característiques als seus valors inicials.

Aquests valors inicials són false pel cas de authStatusChar, cero pel cas de batteryLevelChar i un guió per la resta de característiques.

Un cop establerts els handlers per aquests dos esdeveniments, es comprova que tant el coprocessador criptogràfic com el mòdul de BLE s'inicien i funcionen correctament, en cas contrari no continua l'execució del script.

Si l'execució continua correctament s'inicialitzen els sensors, s'estableix el nom de la balisa, s'afegeixen les diferents característiques als seus corresponents serveis i s'estableixen per ser anunciades als paquets d'advertising emesos per la balisa. Per últim, es crida a la funció de la llibreria arduinoBLE per tal de que comenci a emetre els paquets d'advertising alhora que s'estableixen els valors inicials de les diferents característiques.

Un cop acaba l'execució de tot el codi d'inicialització de la balisa es comença a executar el codi del loop del script que s'executarà de forma indefinida mentre la balisa estigui en marxa.

En el loop es pot apreciar com en el cas de tenir una connexió establerta amb una central es comprova si aquesta està autenticada. El procés d'autenticació consisteix en comprovar si el valor de la característica authChar, modificat per la central, correspon a la contrasenya de la balisa i si és així s'estableix el valor de la característica authStatusChar a true.

Després del procés d'autenticació en el cas de que la central s'hagi autenticat correctament es procedeix a actualitzar els valors de les característiques ja que mentre no s'autentica la central els valors d'aquestes és un guió.

En el procés d'actualització dels valors de les característiques accelChar i gyroChar s'obté les dades dels sensor i es construeix un JWS que conté la informació obtinguda d'aquests pels diferents eixos. A més aquest JWS també té una signatura usada per la central per tal de verificar el remitent d'aquest JWS així com la integritat de la informació i un header que conté informació sobre quin algoritme s'ha usat per la generació de la signatura i l'identificador de la balisa per tal de que la central sàpiga quina clau pública ha d'usar per tal de verificar la signatura.

En el cas de la característica statusChar, igual que en el cas de les dues anteriors també es genera un JWS que conté en aquest cas la informació del temps que porta en marxa la balisa.

El codi sencer de la balisa BLE es pot trobar al repositori de gitlab referenciat a la bibliografia (Barredo Cordón, Gitlab - balisaBLE, 2020).

3.5. Muntatge i instal·lació de la central BLE

Tot i que la Raspberry Pi en sí no necessita muntatge ja que ve muntada de fàbrica, s'ha optat per afegir-li dissipadors de calor de coure al processador i altres microxips així i un ventilador per tal de millorar la seva refrigeració. A més de ficar-la dintre d'una carcassa per protegir-la de cops i altres elements externs.

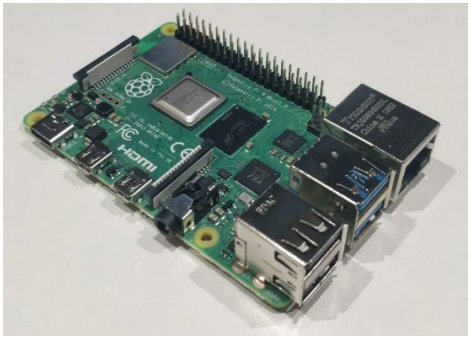

Figura 30: Raspberry Pi 4

A la imatge anterior es pot veure la Raspberry Pi 4 tal qual ve de fàbrica, mentre que a la imatge següent ja se li han instal·lat tres dissipadors de coure i la targeta microSD com a emmagatzematge.

Figura 31: Raspberry Pi 4 amb dissipadors de coure i microSD

Després s'ha introduït la Raspberry Pi a la carcassa que es munta simplement encaixant la part superior i inferior de la carcassa amb la Raspberry Pi al mig. I per últim es cargola el ventilador a la tapa la qual va encaixada a la part superior de la carcassa.

El ventilador té un cable vermell que es connecta al pin de 5V o 3V de la Raspberry Pi segons la velocitat desitjada i un altre cable negre que es connecta al pin de massa.

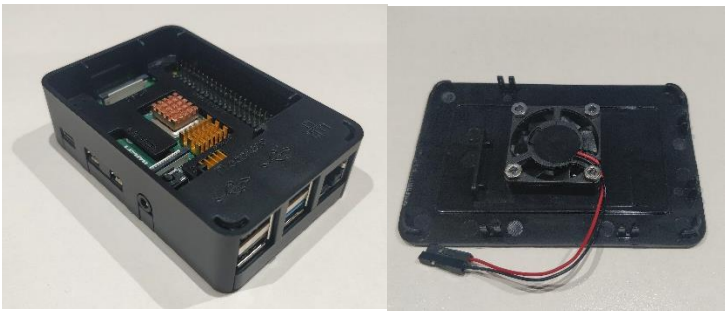

Figura 32: Raspberry Pi 4 amb carcassa i tapa de la carcassa amb el ventilador

Figura 33: Raspberry Pi 4 després de muntar la carcassa

Un cop muntada la Raspberry Pi cal instal·lar-hi el sistema operatiu, que en el cas d'aquest projecte és Raspberry Pi OS, i a més cal canviar la contrasenya per defecte per motius de seguretat.

Per últim, s'ha instal·lat php, mariadb, apache, les llibreries de python necessàries pel projecte i les seves dependències, etc.

En el cas d'apache també ha calgut configurar-ho per tal de que serveixi el panell d'administració pel port 443 amb un certificat autosignat generat expressament per aquest fi.

3.6. Implementació central BLE

Per la implementació d'una central BLE que obtingui la informació de les balises BLE s'ha usat una Raspberry Pi 4 i s'ha fet una recerca i un anàlisis de diferents llibreries que poguessin ser usades per aquest fi.

Algunes de les llibreries estudiades han sigut: bluepy, gatt-python, gattlib, pygatt, PyBluez, node-ble i noble, entre altres. Com es pot apreciar en el nom de les llibreries la majories d'elles són llibreries en python i algunes poques en nodejs tot i que el llenguatge de programació en el que estiguessin desenvolupades no era la característica més important a l'hora de triar una llibreria BLE per aquest projecte tot i que sí una més a tenir en compte.

Sobretot s'ha tingut en compte per triar una llibreria la seva facilitat d'us, que estigui ben documentada i que no hagués sigut abandonada pels desenvolupadors o la comunitat, sense oblidar que sigui capaç de realitzar totes les funciones que seran necessàries per la central que es vol implementar.

D'entre totes les llibreries que s'han tingut en compte s'ha triat bluepy, ja que tenia una bona relació entre documentació i facilitat d'us.

3.6.1. Implementació de la central BLE amb python

Tal i com s'indica a l'apartat anterior s'ha triat la llibreria bluepy per desenvolupar la central BLE que es comunica amb les balises i obté d'elles la informació dels sensors i l'estat.

A l'apartat 6.2 es pot veure el codi principal del script python que obté de la base de dades les balises habilitades i va iniciant un fil d'execució per cadascuna d'elles on intentarà establir connexió.
Va guardant en un array la referència a cadascun d'aquests fils i periòdicament anirà comprovant que els fils continuen vius ja que en cas contrari voldrà dir que la connexió amb la balisa s'ha trencat. Si algun dels fils ha mort s'inicia un fil nou per establir una nova connexió amb aquella balisa.

El que fan els diferents fils d'execució és cridar la funció connectBeacon, el codi de la qual es pot trobar a l'apartat 6.3, en ell es pot veure en que consisteix la connexió a una balisa.

Primer s'instancia un objecte Peripheral passant-li com a paràmetres al constructor l'adreça MAC de la balisa i el tipus d'adreça que pot ser pública o random. Quan s'instancia un objecte d'aquesta classe la llibreria bluepy intenta realitzar una connexió amb el dispositiu i en cas de que s'estableixi correctament podrem interactuar amb aquest mitjançant els seus mètodes.

Un cop establerta la connexió s'obté el servei a partir del seu UUID usant el mètode getServiceByUUID, en aquest cas el UUID del servei és 00007777-0000-1000-8000-00805f9b34fb, s'obté una instància del tipus Service, un objecte que encapsula la informació del servei.

Aquest objecte Service té els mètodes per tal de poder accedir a les característiques del servei, per fer-ho es crida el mètode getCharacteristics passant-li el UUID de la característica que es vol obtenir.

Un cop es té l'objecte del Service s'usa aquest mètode per obtenir totes les característiques del servei amb la informació dels sensors i l'estat.

A continuació s'activen les notificacions per cadascuna de les característiques de dades, aquestes notificacions seran rebudes pel mètode handleNotification de la classe MyDelegate.

A més es llegeix el valor de la característica authStatusChar que indica si s'està autenticat, aquesta característica és de només lectura. En cas de que encara no s'hagi autenticat la central amb la balisa, la central escriu la contrasenya a la característica authChar a la que només es té accés d'escriptura.

A partir d'aquí, la central ja està autenticada amb la balisa i ha activat les notificacions de les característiques de dades per tant només li queda restar a la espera de rebre notificacions amb les actualitzacions dels valors de les característiques.
El procés de processament de les notificacions rebudes es pot veure a l'apartat 6.4, en el codi del mètode handleNotification.

En aquest mètode s'usa la llibreria de python jwcrypto per tal de deserialitzar i verificar el jws token rebut des de la balisa. Per tal de verificar l'autenticitat del token s'identifica quina balisa ha emès el token a partir del camp beaconId del seu header i s'usa la clau pública de la balisa per tal de verificar la signatura del token.

Un cop s'ha deserialitzat i verificat el token es crida una api que forma part del panell d'administració per tal de que guardi la informació obtinguda del token a la base de dades.

El codi sencer de la central BLE es pot trobar al repositori de gitlab referenciat a la bibliografia (Barredo Cordón, Gitlab - CentralBLE , 2020).

3.7. Panell d'administració / Quadre de comandaments

S'ha desenvolupat un panell d'administració o quadre de comandaments on un usuari autoritzat pot accedir-hi per tal de consultar la informació enviada per les balises.

El panell ha estat desenvolupat amb symfony 5.1.8 (PHP 7.4.13) i MariaDB 10.0.28 per la part del backend i HTML, jquery 3.4.1 (javascript) i css pel frontend.

Per tal d'accedir al panell l'usuari haurà de connectar-se a la mateixa xarxa WiFi de la central, ja sigui una xarxa WiFi generada per la central en mode Acces Point o una xarxa WiFi qualsevol a la que estiguin tots dos dispositius connectats. Tot i que l'opció més segura seria la de que l'usuari es connecti a una xarxa WiFi generada per la central en mode Acces Point.

Un cop l'usuari està connectat a la mateixa xarxa que la central només cal que mitjançant el navegador accedeixi al panell. En el cas d'usar una xarxa WiFi generada per la pròpia central l'usuari pot ser redirigit automàticament al panell si aquest es configura com a portal captiu o en cas d'usar una altra xarxa WiFi l'usuari pot accedir-hi mitjançant la IP de la central.

El primer que veurà l'usuari en accedir al panell serà la pantalla de login on haurà d'iniciar sessió per tal d'accedir-hi ja que el seu accés està restringit.

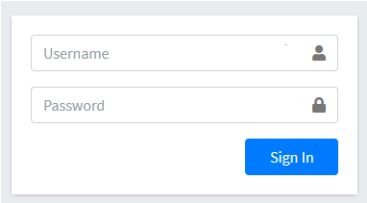

Figura 34: Pantalla de login

Un cop l'usuari inicia sessió correctament ja té accés al panell i és redirigit a la home d'aquest on pot veure l'última informació rebuda de les balises.

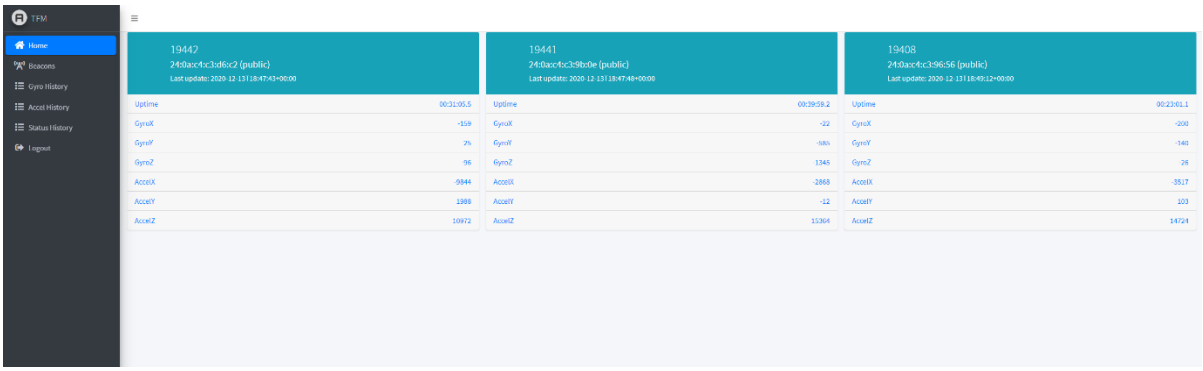

Figura 35: Pantalla home

Tal i com es pot veure a la imatge anterior a la pantalla home es mostra una targeta per cada balisa habilitada on es veu l'última informació rebuda. Aquesta informació es va actualitzant de forma periòdica a mesura que es van reben nous paquets de dades.

La informació mostrada de les balises inclou el seu identificador, l'adreça MAC, el tipus d'adreça (public o random), el timestamp de l'últim cop que s'han actualitzat les dades, el temps que porta en marxa la balisa i la informació obtinguda pel giroscopi i l'acceleròmetre de la balisa en els seus tres eixos.

A més es pot apreciar que a l'esquerra del panell hi ha un menú per tal de poder navegar per les diferents pantalles d'aquest així com per fer logout.

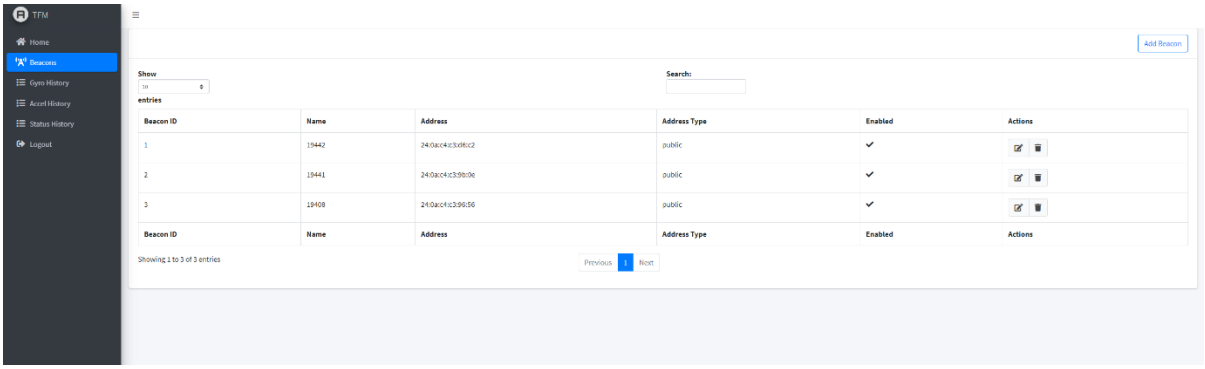

Figura 36: Pantalla beacons

A la pantalla de beacons es pot veure la llista de les balises, veure si estan habilitades o no així com també es poden crear noves balises, editar les existents o eliminar-les.

Quan es prem el botó "Add Beacon" es mostra un formulari on l'usuari pot introduir la informació de la balisa per tal de donar-la d'alta al panell, aquest formulari és el mateix que el d'editar la informació d'una balisa excepte que en el cas d'editar el formulari està preomplert amb la informació de la balisa que es vol editar.

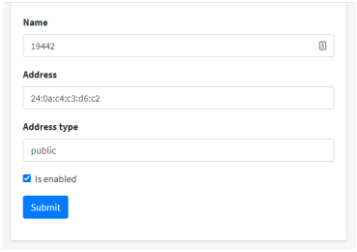

Figura 37: Formulari edició balises

A la pantalla de l'historial de dades del giroscopi es pot veure una taula amb el registre de tota la informació rebuda referent als giroscopis de les balises. En aquesta pantalla es permet a l'usuari filtrar pels diferents camps, com per exemple filtrar els registres de tal manera que només es vegin els d'una balisa en concret.

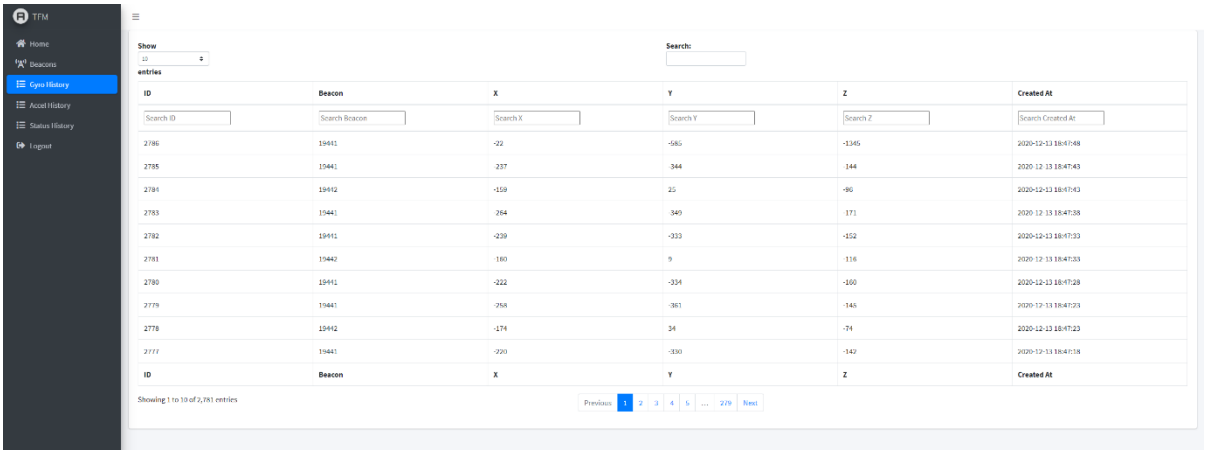

Figura 38: Pantalla d'historial del giroscopi

La pantalla amb l'historial del acceleròmetre és exactament igual que la del historial del giroscopi i amb les mateixes funcionalitats però mostrant la informació dels acceleròmetres de les balises.

Per últim, tenim la pantalla d'historial d'estatus on es pot veure les dades rebudes de les balises referent al seu estat, en aquest cas la informació que es mostra és el temps que porta en marxa la balisa però es podria fer que la balisa envies més informació rellevant sobre el seu estat i ser mostrada en aquesta pantalla.

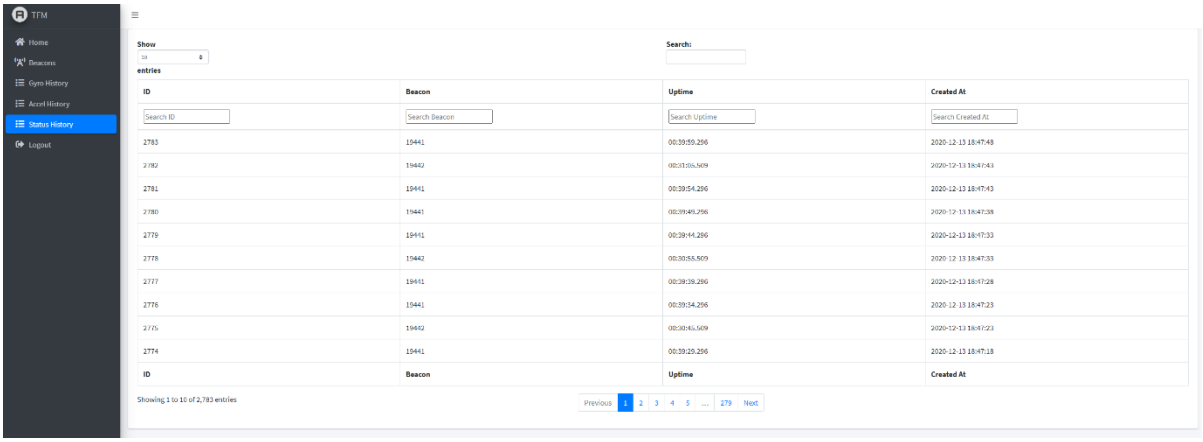

Figura 39: Pantalla d'historial d'estat

A més d'aquestes pantalles que es poden veure entrant al panell d'administració, aquest també consta d'una api que és la que s'encarrega de rebre les peticions del script python de la mateixa central i guardar a la base de dades la informació rebuda.

El codi sencer del panell d'administració es pot trobar al repositori de gitlab referenciat a la bibliografia (Barredo Cordón, Gitlab - PanellAdmin , 2020).

3.7.1. Estructura de la base de dades

A la implementació del panell d'administració ha calgut dissenyar una arquitectura de base de dades que permetés guardar la informació de les balises de forma estructurada així com gestionar els permisos dels usuaris autoritzats a accedir al panell d'administració. Com que s'ha usat Symfony per desenvolupar-ho l'estructura de la base de dades ha estat generada automàticament a partir de les entitats i relacions definides al projecte.

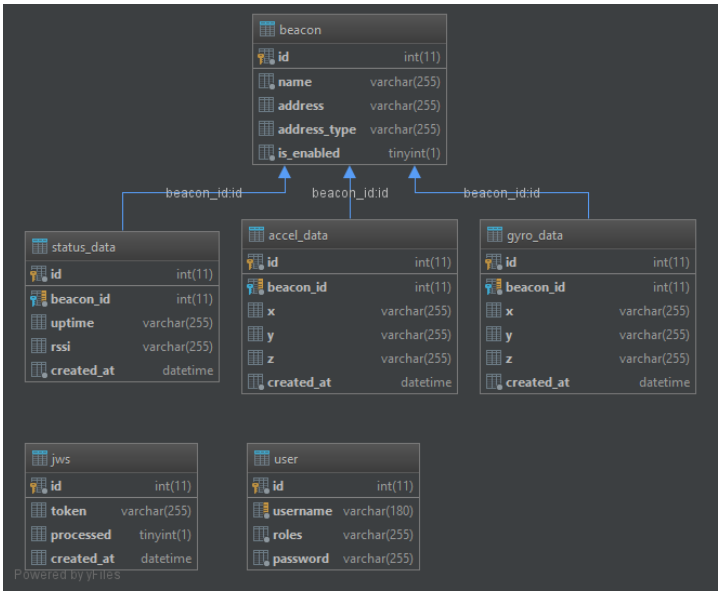

Figura 40: Diagrama base de dades

Tal i com es pot veure a la imatge anterior, s'ha creat una entitat balisa per tal de guardar tota la informació relativa a les balises que té el sistema de telemetria, això inclou, el nom, l'adreça MAC, el tipus d'adreça i si la balisa està habilitada o no.

També s'han creat tres entitats per tal de guardar la informació rebuda dels sensors de les balises, aquestes entitats són StatusData, AccelData i GyroData. Totes tres entitats tenen un identificador de balisa que relaciona la informació amb la balisa a la qual pertany així com un camp per tal de guardar la data en la que s'han obtingut aquests valors.

Les entitats AccelData i GyroData són molt similars, totes dues tenen un camp per cadascun dels eixos x, y i z per tal de guardar la informació obtinguda pel sensor en aquests eixos.

L'entitat StatusData per altra banda conté informació sobre l'estat de la balisa com és el temps que porta aquesta en marxa, així com un camp rssi que es va afegir en una de les proves del prototip per tal d'intentar guardar a més a més del uptime la potència de la senyal entre la balisa i la central.

A més en el projecte també existeix una entitat User que no està relacionada amb cap de les altres i que es usada per guarda la informació relativa als usuaris que poden accedir al panell. Aquesta entitat consta de tres camps, username, password i roles. El username conté el nom d'usuari amb el que l'usuari ha d'iniciar sessió, el password conté un hash de la contrasenya així com el nom de l'algoritme usat per calcular el hash i altres paràmetres com el salt. El camp de roles s'ha afegit degut a que era un requeriment del security de Symfony però en aquest projecte no és imprescindible ja que només existeix un tipus d'usuari.

Al diagrama també es pot apreciar una taula anomenada jws, això es degut a que durant l'elaboració del prototip s'han fet diverses proves de concepte i algunes d'elles s'han mantingut de forma opcional. En el cas d'aquesta taula era usada per tal de que les balises no haguessin de verificar els tokens rebuts a temps real, si no que podien inserta el token jws tal qual el rebien en aquest taula i més tard era verificat per un procés asíncron i guardada la informació que contenia a les taules corresponents.

A més tot i que no s'ha mencionat en els paràgrafs anteriors, totes les entitats consten d'un camp id usat per tal de identificar de forma única cada registre. No només és una bona pràctica tenir aquest identificador com a index autoincrementat si no que a més a més és obligatori quan es treballa amb Symfony ja que doctrine, el ORM usat per Symfony per defecte, l'usa per la seva gestió interna de la base de dades.

3.8. Problemes durant la implementació

A la fase d'implementació del prototip de sistema de telemetria basat en IoT usant BLE han sorgit diferents problemes que han dificultat la implementació i enrederit la planificació d'aquesta.

3.8.1. Port busy

Durant la implementació de la balisa BLE usant una placa Arduino s'usava el terminal del IDE d'Arduino per tal de poder testejar i obtenir informació de depuració durant les proves de concepte de les diferents parts de la balisa.

La balisa envia la informació a la terminal del IDE mitjançant el port USB que es mapejat com a port COM, però aquest port també es usat per tal de pujar una nova versió del codi a l'Arduino. Això va provocar que durant algunes proves de concepte el port COM quedés bloquejat per la informació que enviava la placa impedint actualitzar el codi i per tant deixant la placa inservible.

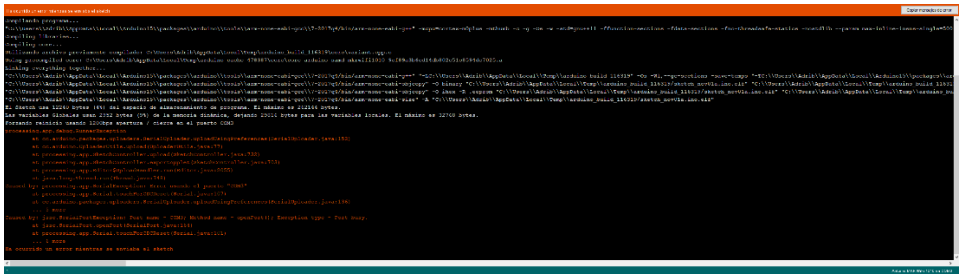

Figura 41: Diagrama base de dades

Buscant a internet a fòrums sobre el desenvolupament amb Arduino hi havia fils on s'indicava que si es manté pressionat el botó de reset integrat a la placa Arduino fins el moment just on l'IDE comença a pujar el codi després de compilar-ho s'evita aquest error.

Però tot i que això va arribar ha funcionar no era una manera viable d'evitar l'error ja que costava molt d'aconseguir fer-ho en el moment just i que funcionés i per tant es perdia molt de temps intentant aplicar aquesta solució cada cop que es produïa l'error.

Després de continuar investigant aquest error pels fòrums i contactar amb el venedor, es va trobar un altre fil on es mencionava que es podia evitar aquest problema prement dos cops el botó integrat de reset. Aquesta si que va ser la solució adequada al problema, prement dos cops els botó de reset la placa es desconnectava del port USB i es reconnectava, aquest cop mapejada amb un altre port COM que no estava ocupat i permetia carregar el nou codi sense problema.

3.8.2. Broken pipe

Un altre error que va sorgir quan s'estava desenvolupant la comunicació entre les balises i la central BLE és que es tancava sobtadament el procés d'execució del script de la central BLE mostrant l'error "broken pipe".

Aquest error no es produïa sempre i es podia solucionar simplement reiniciant el procés d'execució de la central BLE, tot i així per tal de donar més robustesa al sistema es va depurar codi i es van fer proves per tal d'esbrinar l'origen del error.

Aquest error es produïa durant la recepció del JWS degut a que la Raspberry Pi no era capaç de processar a temps la gran quantitat de dades rebudes. Algunes de les mesures que es van prendre va ser reduir la freqüència d'advertising de les balises a cinc segons ja que no era necessari rebre dades cada segon.

A més, s'ha implementat un sistema multithreading on el script de la central BLE inicia un fil d'execució independent per rebre les dades de cada balisa i va comprovant de forma periòdica si aquests fils continuen executant-se en cas de que s'hagi detingut la execució d'algun fil el reinicia.

4. Conclusions i treball futur

4.1. Conclusions

Durant la fase d'investigació d'aquest TFM s'han analitzat les comunicacions entre dispositius BLE així com els diferents protocols que intervenen per tal d'entendre el seu funcionament i les diferents vulnerabilitats i punts febles que poden tenir per tal de poder implementar un prototip de sistema de telemetria segur.

Després de la fase d'anàlisi i d'implementar un prototip de sistema de telemetria BLE basat en IoT segur s'ha arribat a la conclusió de que actualment existeix la tecnologia necessària per tal de poder realitzar encriptació i signatura amb algoritmes de clau asimètrica amb dispositius de baix cost, tot i que es continuen tenint certes limitacions.

Per exemple, en el cas de la central BLE implementada amb una Raspberry Pi 4 de 4GB de RAM, s'han trobat limitacions a la hora de poder verificar la signatura del JWS enviat per les balises ja que si aquest contenia molta informació la central no era capaç d'executar l'algoritme de verificació amb clau pública prou ràpid, produint errors i trencant-se la comunicació amb la balisa.

Això no vol dir que no sigui possible enviar molta més informació, si no que destaca el fet de que existeix una limitació de capacitat de processament del dispositiu i que per tant cal tenir molta cura a l'hora de programar-ho per tal de fer-ho de la manera més optima possible reduint el cost computacional el màxim possible. Ja que a més la central ha de realitzar aquestes operacions de forma simultània amb cadascuna de les balises.

En canvi, en el cas de les balises s'han pogut implementar usant una placa Arduino MKR WiFi 1010 de forma relativament senzilla i aquesta és capaç de generar i signar un JWS Token sense detectar problemes de rendiment gràcies al seu coprocessador criptogràfic.

A la fase d'implementació d'aquest prototip, a més de les limitacions de processament de la Raspberry Pi anteriorment mencionades, s'han trobat limitacions a les llibreries disponibles tant per l'Arduino com per la Raspberry Pi.

Les llibreries existents amb pocs mètodes limitaven la llibertat a l'hora de dissenyar un protocol de comunicació segur en el sistema de telemetria. Tot i així, això no hauria de ser un problema per les empreses que desenvolupen els seus dispositius IoT BLE, ja que si dissenyen el seu propi hardware i software poden adaptar-ho a les seves necessitats i per tant no és excusa per no implementar al mercat dels dispositius IoT mesures de seguretat adequades.

Igualment amb plataformes de codi obert i de hardware obert com Arduino es podria desenvolupar una llibreria pròpia adaptada a les necessitats concretes del projecte, però això requeriria de molt més temps i coneixements a baix nivell de la plataforma.

4.2. Treball futur

Com a possibles línies de treball futur caldria millorar o optimitzar algunes de les parts del projecte que degut a la limitació temporal han sigut acotades o simplificades.

Algunes de les possibles millores del projecte serien les següents:

- Afegir la informació de la potencia de senyal amb cadascuna de les balises al panell d'administració.
- Posar bateries a les balises i mostrar el valor real de la bateria al panell d'administració, ja que actualment al panell es mostra una simulació del nivell de bateria de la balisa degut a que durant la fase d'implementació hi ha hagut dificultats per tal de trobar un adaptador de les connexions que tenien les bateries comprades.
- Millorar el codi principal de la central per tal de solucionar l'error de "broken pipe" d'una manera més optima.
- Permetre afegir la clau d'una nova balisa des del panell d'administració ja que actualment cal afegir un fitxer amb la clau a una ruta concreta de la central.
- Desenvolupar una llibreria pròpia per gestionar el mòdul BLE d'Arduino i d'aquesta manera poder adaptar-la millor al projecte.
- Continuar revisant el sistema buscant possibles vulnerabilitats per tal de reforçar la seva seguretat.

5. Bibliografia

Alletto, S., Cucchiara, R., Del Fiore, G., Mainetti, L., Mighali, V., Patrono, L., & Serra, G. (Abril / 2016). An Indoor Location-aware System for an IoT-based Smart Museum. *IEEE Internet of Things Journal*, 244-253. doi:10.1109/JIOT.2015.2506258

Barredo Cordón, A. (12 / 2020). *Gitlab - balisaBLE.* Recollit de https://gitlab.com/abarredo-tfm/balisable

Barredo Cordón, A. (12 / 2020). *Gitlab - CentralBLE .* Recollit de https://gitlab.com/abarredo-tfm/centralBLE

Barredo Cordón, A. (12 / 2020). *Gitlab - PanellAdmin .* Recollit de https://gitlab.com/abarredo-tfm/panelladmin

Chan, A. C.-F., Wong, J. W., Zhou, J., & Teo, J. (2016). Scalable two-factor authentication using historical data. *European Symposium on Research in Computer Security*, (p. 91-110).

Claeys, T., Rousseau, F., & Tourancheau, B. (2017). Securing Complex IoT Platforms with Token Based Access Control and Authenticated Key Establishment. *International Workshop on Secure Internet of Things (SIOT).* Oslo.

content.arduino.cc. (2017). Consultat el 17 / 10 / 2020, a https://content.arduino.cc/assets/mkr-microchip_atecc508a_cryptoauthentication_device_summary_datasheet-20005927a.pdf

Harvey, I. (24 / 11 / 2020). *bluepy.* Recollit de https://ianharvey.github.io/bluepy-doc/

Kinikar, S., & Terdal, S. P. (2016). Implementation of open authentication protocol for IoT based application. *International Conference on Inventive Computation Technologies (ICICT)*, (p. 1-4). Coimbatore.

microchipdeveloper. (24 / 11 / 2020). Recollit de https://microchipdeveloper.com/wireless:ble-link-layer-channels

Shingala, K. (2019). JSON Web Token (JWT) based client authentication in Message Queuing Telemetry Transport (MQTT). *Cryptography and Security, Cornell University*.

Yerlikaya, Ö., & Dalkiliç, G. (2018). Authentication and Authorization Mechanism on Message Queue Telemetry Transport Protocol. *3rd International Conference on Computer Science and Engineering (UBMK)*, (p. 145-150). Sarajevo.

Zaim, D., & Bellafkih, M. (2016). Bluetooth Low Energy (BLE) Based Geomarketing System. *11th*, (p. 1-6).

6. Annex

6.1. Codi balisa BLE amb Arduino

```
#include <ArduinoBLE.h>

#include <ArduinoECCX08.h>
#include <utility/ECCX08JWS.h>
#include <ECCX08.h>

#include "I2Cdev.h"
#include "MPU6050.h"

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif

MPU6050 accelgyro;

int16_t ax, ay, az;
int16_t gx, gy, gz;

int advertisingTime = 5000;
String beaconPassword = "=W$(GeTjs9J;'x55Rp]Xr";

BLEService batteryService("180F");
BLEService dataService("7777");

BLEUnsignedCharCharacteristic batteryLevelChar("2A19", BLERead | BLENotify);
BLEStringCharacteristic accelChar("6666", BLERead | BLENotify, advertisingTime);
BLEStringCharacteristic gyroChar("5555", BLERead | BLENotify, advertisingTime);
BLEStringCharacteristic statusChar("4444", BLERead | BLENotify, advertisingTime);

BLEStringCharacteristic authChar("0000", BLEWrite, advertisingTime);
BLEBoolCharacteristic authStatusChar("0001", BLERead);

int oldBatteryLevel = 0;  // last battery level reading from analog input
long previousMillis = 0;  // last time the battery level was checked, in ms

String header = "{\"alg\":\"ES256\",\"typ\":\"JWS\",\"beaconId\":\"19441\"}";

void setup() {

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  Wire.begin();
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
```

```
  Fastwire::setup(400, true);
#endif

  Serial.begin(9600);    // initialize serial communication

  pinMode(LED_BUILTIN, OUTPUT); // initialize the built-in LED pin to indicate when a central is
connected

  BLE.setEventHandler(BLEConnected, blePeripheralConnectHandler);
  BLE.setEventHandler(BLEDisconnected, blePeripheralDisconnectHandler);

  if (!ECCX08.begin()) {
    Serial.println("No ECCX08 present!");
    while (1);
  }

  // begin initialization
  if (!BLE.begin()) {
    Serial.println("starting BLE failed!");

    while (1);
  }

  accelgyro.initialize();

  BLE.setDeviceName("Beacon 19441");
  BLE.setLocalName("Beacon 19441");

  BLE.setAdvertisedService(batteryService); // add the service UUID
  batteryService.addCharacteristic(batteryLevelChar); // add the battery level characteristic
  BLE.addService(batteryService); // Add the battery service
  batteryLevelChar.writeValue(oldBatteryLevel); // set initial value for this characteristic

  dataService.addCharacteristic(accelChar);
  dataService.addCharacteristic(gyroChar);
  dataService.addCharacteristic(statusChar);
  dataService.addCharacteristic(authChar);
  dataService.addCharacteristic(authStatusChar);
  BLE.addService(dataService);

  // start advertising
  BLE.advertise();

  resetChars();

  Serial.println("Bluetooth device active, waiting for connections...");
}
```

```
void loop() {
  // wait for a BLE central
  BLEDevice central = BLE.central();

  // if a central is connected to the peripheral:
  if (central) {
    if (authChar.value() == beaconPassword) {
      authStatusChar.writeValue(true);
    } else {
      authStatusChar.writeValue(false);
    }

    // while the central is connected:
    while (authStatusChar.value() && central.connected()) {
      long currentMillis = millis();
      int rssi = 0; //central.rssi();

      if (currentMillis - previousMillis >= advertisingTime) {
        previousMillis = currentMillis;
        updateBatteryLevel();
        updateAccel();
        updateGyro();
        updateStatus(rssi);
      }
    }
  }
}

void blePeripheralConnectHandler(BLEDevice central) {
  // central connected event handler
  digitalWrite(LED_BUILTIN, HIGH);
  Serial.print("Connected event, central: ");
  Serial.println(central.address());
}

void blePeripheralDisconnectHandler(BLEDevice central) {
  // central disconnected event handler
  resetChars();
  digitalWrite(LED_BUILTIN, LOW);
  Serial.print("Disconnected event, central: ");
  Serial.println(central.address());
}

void resetChars() {
  authChar.writeValue("-");
  batteryLevelChar.writeValue(0);
```

```
  accelChar.writeValue("-");
  gyroChar.writeValue("-");
  statusChar.writeValue("-");
  authStatusChar.writeValue(false);
}


void updateBatteryLevel() {
  int battery = analogRead(A0);
  int batteryLevel = map(battery, 0, 1023, 0, 100);

  if (batteryLevel != oldBatteryLevel) {      // if the battery level has changed
    batteryLevelChar.writeValue(batteryLevel);  // and update the battery level characteristic
    oldBatteryLevel = batteryLevel;           // save the level for next comparison
  }
}

void updateAccel() {
  accelgyro.getAcceleration(&ax, &ay, &az);

  char accelBuff[100];
  sprintf(accelBuff, "{\"accelX\":\"%i\",\"accelY\":\"%i\",\"accelZ\":\"%i\"}", ax, ay, az);
  String accel = String(accelBuff);

  String jws = ECCX08JWS.sign(0, header, accel);
  accelChar.writeValue(jws);
}

void updateGyro() {
  accelgyro.getRotation(&gx, &gy, &gz);

  char gyroBuff[100];
  sprintf(gyroBuff, "{\"gyroX\":\"%i\",\"gyroY\":\"%i\",\"gyroZ\":\"%i\"}", gx, gy, gz);
  String gyro = String(gyroBuff);

  String jws = ECCX08JWS.sign(0, header, gyro);
  gyroChar.writeValue(jws);
}

void updateStatus(int rssi) {
  char statusBuff[100];
  sprintf(statusBuff, "{\"uptime\":\"%i\", \"rssi\":\"%i\"}", millis(), rssi);
  String statusData = String(statusBuff);

  String jws = ECCX08JWS.sign(0, header, statusData);
  statusChar.writeValue(jws);
}
```

6.2. Codi principal de la central BLE

```
bd = Database()
beacons = bd.getBeacons()

if (beacons == -1):
    appendToLog("NO BEACONS ENABLED")
    quit()

threadsArray = []

while True:
    try:
        appendToLog("START")

        if (len(threadsArray) == 0):
            for beacon in beacons:
                t = threading.Thread(target=connectBeacon, args=(beacon,))
                t.setName(beacon[0])
                t.start()
                threadsArray.append(t)
        else:
            for index in range(len(threadsArray)):
                if threadsArray[index].isAlive():
                    print("Alive thread: " + threadsArray[index].getName())
                    continue
                print("Restarting thread: " +
threadsArray[index].getName())
                threadsArray[index] =
threading.Thread(target=connectBeacon, args=(beacons[index],))
                threadsArray[index].setName(beacons[index][0])
                threadsArray[index].start()

    except Exception as e:
        appendToLog(e)

    appendToLog("END")
    time.sleep(5.0)
```

6.3. Funció per establir connexió amb una balisa

```
def connectBeacon(beacon):
    appendToLog("Thread: " + beacon[0])

    p = Peripheral(beacon[0], beacon[1])

    p.getServices()
    dataService = p.getServiceByUUID("00007777-0000-1000-8000-
00805f9b34fb")

    # Auth Characteristics
    authChar = dataService.getCharacteristics(UUID("0000"))[0]
    authStatusChar = dataService.getCharacteristics(UUID("0001"))[0]

    # Data Characteristics
    accelChar = dataService.getCharacteristics(UUID("6666"))[0]
    gyroChar = dataService.getCharacteristics(UUID("5555"))[0]
```

```
    statusChar = dataService.getCharacteristics(UUID("4444"))[0]

    # Activant notificacions de les Characteristics
    p.setDelegate(MyDelegate(p))
    p.writeCharacteristic(accelChar.valHandle + 1, b"\x01\x00")
    p.writeCharacteristic(gyroChar.valHandle + 1, b"\x01\x00")
    p.writeCharacteristic(statusChar.valHandle + 1, b"\x01\x00")

    # Authenticating
    try:
        appendToLog(p.addr + " " + str(authStatusChar.read()))
        if authStatusChar.read() == b'\x00':
            appendToLog("Authenticating: " + p.addr)
            authChar.write(password, True)
    except Exception as e:
        appendToLog(e)

    # Esperant notificacions
    try:
        while True:
            if p.waitForNotifications(1.0):
                # handleNotification() was called
                appendToLog("Notification: " + p.addr)
                continue
            appendToLog("Waiting... " + p.addr)

    except Exception as e:
        appendToLog(e)
```

6.4. Classe MyDelegate

```
class MyDelegate(btle.DefaultDelegate):
    p = None

    def __init__(self, peripheral):
        btle.DefaultDelegate.__init__(self)
        self.p = peripheral

    def handleNotification(self, cHandle, data):

        char = self.p.readCharacteristic(cHandle)
        if (len(sys.argv) > 1 and sys.argv[1] == "async"):
            token = char.decode("ASCII")
            bd.insertJWS(token)
        else:
            jwstoken = decodeToken(char)
            verifyToken(jwstoken)
            appendToLog(jwstoken.payload)
            data = json.loads(jwstoken.payload)

            header = jwstoken.jose_header
            beaconId = header.get("beaconId")
            bd_beaconId = bd.getBeaconId(beaconId)

            if bd_beaconId > 0:
                if data.get("uptime") is not None:
                    insertStatusData(bd_beaconId, data)
```

```
                if data.get("gyroX") is not None:
                    insertGyroData(bd_beaconId, data)
                if data.get("accelX") is not None:
                    insertAccelData(bd_beaconId, data)
```

6.5. Funcions per verificar y deserialitzar el JWS token

```
def decodeToken(token):
    try:
        jwstoken = jws.JWS()

        if(type(token) != str):
            token = token.decode("ASCII")

        jwstoken.deserialize(token)
        return jwstoken
    except Exception as e:
        appendToLog(e)
        appendToLog("Formato invalido")


def verifyToken(jwstoken):
    try:
        header = jwstoken.jose_header
        beaconId = header.get("beaconId")
        with open("certs/" + beaconId + ".pem", "rb") as pemfile:
            key = jwk.JWK.from_pem(pemfile.read())
        jwstoken.verify(key)
        appendToLog("OK")
    except Exception as e:
        appendToLog("Firma Invalida")
```