ServerSnitch. A system-independent IoT solution for monitoring server/computer status.

Jose Antonio Ramos Andrades

Resum– Un dels majors problemes pels administradors de sistemes és no saber l'estat real dels seus ordinadors o servidors que tenen a llocs de treballs allunyats o a armaris a una sala de servidors. Tenen un coneixement limitat de la informació que el sistema és capaç de proveir. Quan les condicions no són les millors, com per exemple, quan la xarxa d'internet cau o quan el servidor està parat, és ja massa tard per actuar. Aquest projecte busca solucionar aquest problema nodrint d'informació una aplicació al núvol, informació que s'enviarà fent servir o bé la connexió habitual de l'ordinador, o bé una xarxa Iot si la xarxa habitual no funciona o l'ordinador/servidor està parat.

Paraules clau– Internet de les coses, monitoratge d'estat de servidors, cloud, control remot.

Abstract– One of the major problems for system administrators is not knowing the real status of their computers or servers that are located in remote workplaces or cabinets in a server room. They have limited knowledge of the information that the system is capable of providing. When conditions are not optimal, such as when the internet network goes down or when the server is offline, it is already too late to take action. This project aims to solve this problem by gathering information through a cloud-based application, which will be sent using either the computer's regular connection or an IoT network if the regular network is not functioning or the computer/server is offline.

Keywords– Internet of Things, server monitoring, cloud, remote control.

1 Introducció - Context del treball

ServerSnitch busca ser un dispositiu que interactua amb la placa base de l'ordinador/servidor -a partir d'ara *el sistema*- de manera que permet operar-lo a distància sense l'estricta necessitat que el sistema estigui encès o tingui una connexió a internet convencional. Això es busca aconseguir fent ús d'un dispositiu que es connecti als pins de control d'encès i apagat *del sistema* (vegeu figura 1). Això permet encendre, apagar i reiniciar *el sistema* remotament d'una manera física fent ús dels pins comentats anteriorment i a més a més, el dispositiu busca ser un intermediari per proveir d'informació al núvol. Informació que pot anar des de l'ús de CPU, RAM i disc al consum energètic, informació de serveis o processos, etc. En resum tot allò que *el sistema* operatiu sigui capaç de consultar. Mitjançant el port intern USB (vegeu figura 2) el dispositiu serà capaç d'interactuar amb el sistema operatiu per aconseguir aquesta informació i enviar-la al núvol fent ús de la connexió del propi *sistema* o la xarxa IoT en cas que hi hagi un error a la xarxa.

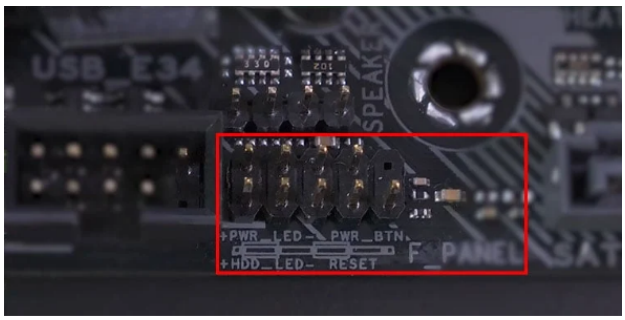


Fig. 1: Pins d'encès i re-inici de la placa base. [1]

• E-mail de contacte: info@ramosandrades.com
• Menció realitzada: Enginyeria del Software
• Treball tutoritzat per: Marc Codina Barberà (Departament de Microelectrònica i Sistemes Electrònics)
• Curs 2022/23

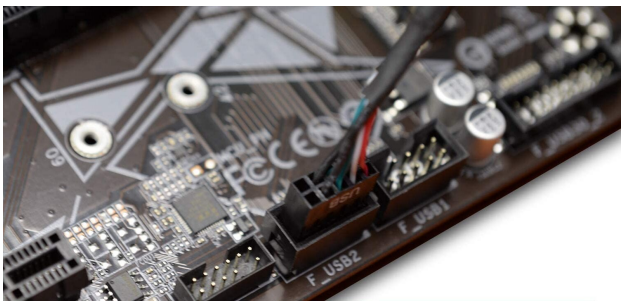


Fig. 2: Connexió per a port USB de la placa base. [2]

2 Estat de l'art

En aquest apartat es busca donar una visió de les tecnologies que existeixen i que s'utilitzen per a intentar assolir els requisits de monitoratge i control de servidors i ordinadors. Com és natural no posaré aquí totes les possibilitats, però si un conjunt d'eines que desenvolupen tasques semblants a les que busco assolir jo amb els meus objectius.

2.1 Nagios

Nagios [3] és una eina de monitoratge de serveis i dispositius de codi obert que basa la seva operació a detectar i avisar de problemes de rendiment de la xarxa o d'impediment d'arribada de paquets i transmissions entre dispositius. Aquesta eina s'ha d'instal·lar en un servidor i es comunicarà amb els altres servidors fent ús de protocols com SNMP. Per tant, és una solució de programari que depèn de la xarxa per funcionar i d'un accés a internet per a poder mostrar l'estat dels dispositius i dels serveis. En el cas que el servidor on hi hagi *Nagios* instal·lat no tingui accés a la xarxa o estigui parat el servei de monitoratge i notificacions es pararà. Per altra banda aquest programari de monitoratge és gratuït, molt utilitzat i compta amb una àmplia comunitat que desenvolupen *plugins* i s'ajuden entre si.

2.2 EnerMon

A l'article [4] es parla del desenvolupament d'un dispositiu capaç de mesurar el consum energètic de qualsevol aparell mitjançant diversos sensors de mesura de corrent no invasius. Si bé aquest dispositiu utilitza la xarxa *LoRa* [5] i no una xarxa d'internet convencional no compleix tots els altres objectius que busca complir *ServerSnitch*. El preu de la solució que es proposa a l'article és d'uns 80€.

2.3 Projecte per desenvolupar un interruptor virtual

Aquest projecte, que es pot trobar a *hackster* [6], és capaç d'encendre un ordinador fent ús de la mateixa connexió que tinc intenció d'utilitzar jo, amb una aplicació és capaç d'encendre un ordinador fent ús de la connexió WiFi de la casa i el protocol *MQTT* [7]. El projecte és barat de realitzar, però es limita únicament a encendre i parar l'ordinador, no té cap interacció amb el programari ni permet recopilar informació al núvol.

2.4 Avantatges de ServerSnitch

Com es pot veure hi ha diversos dispositius que permeten assolir alguns dels objectius que m'he proposat aconseguir, no obstant això, no hi ha cap solució al mercat que permeti gestionar i monitorar un *sistema* remotament amb les mateixes característiques que *ServerSnitch*. La qual cosa m'ha permès tirar endavant el projecte per tal de poder fer una solució pròpia des de zero.

Com a avantatges envers la competència que hi pugui haver es pot denotar que *ServerSnitch* és capaç de demanar dades al sistema operatiu, cosa que permet tenir un monitoratge més exhaustiu i, a més, permet establir una comunicació amb el núvol per a la transferència de dades sense la necessitat de fer servir la mateixa xarxa que l'aparell que s'està monitorant. S'espera que el cost de desenvolupament del dispositiu sigui baix, ja que no es requereix maquinari especialitzat, simplement una placa de desenvolupament que permeti una connexió amb la xarxa IoT i uns quants cables per a fer les connexions pertinents.

3 Objectiu

L'objectiu del projecte és dissenyar i desenvolupar un dispositiu i una arquitectura capaços d'assolir els següents punts.

1. El dispositiu ha de poder encendre i parar *el sistema* de manera remota, sigui amb connexió a internet o sense.
2. El dispositiu ha de poder establir una connexió directa amb el sistema operatiu fent ús d'un programa que servirà per demanar tota aquella informació que interessi enviar al núvol.
3. Per tal de mantenir aquesta comunicació directa i gestionar tota l'operativa caldrà desenvolupar un programa que correrà al dispositiu.
4. S'ha de desenvolupar una aplicació al núvol capaç de gestionar els diferents dispositius i veure la informació d'aquests.
5. L'aplicació al núvol ha de permetre fer ús de la capacitat del dispositiu per encendre i parar *el sistema*.

Per assolir aquests objectius caldrà identificar quina plataforma és la més adient per a dur a terme el projecte. Això inclou escollir un controlador, una xarxa IoT, un servidor per allotjar l'aplicació al núvol i els llenguatges de programació adients per cada part.

Per facilitat d'ús per part de l'usuari, el programa que s'executarà al sistema operatiu per tal de recopilar la informació *del sistema* s'escriurà en *Python* [8], d'aquesta manera també el faig *platform independent*. Tota la part que gestioni la comunicació sistema operatiu i dispositiu no haurà de ser programat per l'usuari, però per donar d'alta els serveis o informació a monitorar caldrà tenir un petit coneixement de *Python* per part de l'usuari final.

4 Metodologia i planificació

La metodologia triada per desenvolupar el projecte és *KanBan* [9]. *KanBan* es basa en l'ús de targetes sobre

un panell que s'utilitzen per visualitzar el flux de treball i gestionar el procés de desenvolupament i disseny d'una manera àgil i eficient. *KanBan* es divideix en tres fases: planificació, gestió i millora continua. En la fase de planificació, s'identifiquen els elements clau del procés i es dissenyen els fluxos de treball. En la fase de gestió, s'utilitza una sèrie de targetes per indicar l'estat de cada tasca durant el procés d'execució. En la fase de millora continua, s'analitzen les dades i s'ajusten els processos per millorar l'eficiència i la qualitat del producte, en aquest cas serà el dispositiu i el programari que es desenvoluparà.

Com tota eina es pot fer servir com està o personalitzar-la, en el meu cas he decidit fer les columnes que es poden veure a la figura 3, no només tenint les tasques a fer i els diferents estats sinó també una columna extra per a tenir a la vista els diferents artefactes que generaré durant el procés. Concretament, he creat el meu tauler *KanBan* a l'aplicació web *Trello* [10] (vegeu figura 3).

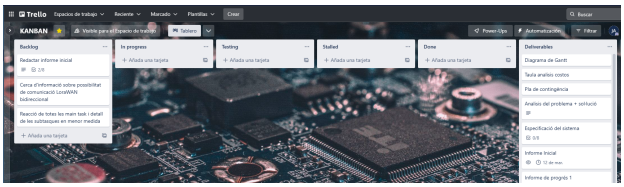


Fig. 3: Taulell Trello a la fase d'inici del projecte.

4.1 Planificació

Per tal de tenir una visió general a l'inici del projecte vaig identificar les tasques més grans i les vaig dividir en subtasques, d'aquesta manera he tingut un detall de tot el que s'ha de fer. A l'annex 1 es pot trobar una imatge on es poden veure les tasques i la seva duració així com una ordenació cronològica que he estimat per poder assolir el projecte. Aquesta primera planificació l'he fet amb *Microsoft Project* [11].

Com a punts a remarcar es van calcular unes 300 hores de treball sense tenir en compte les diferents reunions amb el tutor ni l'avaluació del tribunal. En termes de reunions només vaig tenir en compte la reunió inicial. Es va definir la data d'inici del projecte el dia 17/02/2023, que és la data de la primera reunió i s'espera que el projecte estigui acabat pel dia 02/07/2023. Finalment, aquesta planificació no s'ha pogut complir per diferents motius que s'aniran explicant a l'apartat Disseny i desenvolupament 5. Així doncs, la planificació resultant a l'acabar el projecte es pot veure a l'annex 2. Amb un total de 505 hores dedicades i diversos canvis en la planificació cronològica de les tasques.

Com a resultat de fer la planificació s'han identificat una sèrie de requisits funcionals que es poden trobar als annexos (vegeu annex 3). Les diferents fases identificades del projecte han estat les següents:

- Definició de requisits: Fase on es defineix la sèrie de punts que s'han de complir per tal de considerar que el projecte s'acaba satisfactòriament.
- Cerca de material i recursos: Fase utilitzada per buscar el material maquinari i programari necessari per al desenvolupament del projecte.
- Desenvolupament del dispositiu: Fase per dissenyar i desenvolupar el programa que s'executarà al dispositiu seleccionat per tal de dur a terme la seva operativa.
- Desenvolupament del backend: Fase per dissenyar i desenvolupar el programa que s'executarà al núvol per tal de poder actuar com a font de referència on enviar les dades des del dispositiu i demanar-les des de l'aplicació.
- Desenvolupament de l'aplicació: Fase per dissenyar i desenvolupar l'aplicació web per poder gestionar l'operativa de tota la solució.
- Integració de les parts: Fase per integrar les diferents parts de la solució de manera les relacions entre aquestes parts funcionin de manera correcta.
- Test del sistema: Fase per provar la solució sencera, de manera que tot funcioni correctament i com s'espera.
- Final del projecte: Fase on recollir tota la informació generada pel treball fet per tal de documentar i avaluar el tancament del projecte i els propers passos.

Després d'identificar les diferents fases vaig fer un anàlisi per comprovar quins requisits havia de complir el treball per tal de ser vàlid des del punt de vista funcional. La taula de requisits es pot veure a l'annex (vegeu annex 1).

5 Disseny i desenvolupament

Durant les diferents fases del desenvolupament del projecte s'han generat una sèrie d'artefactes que aniré mostrant ordenadament, així com els diferents problemes, solucions i conclusions que s'han generat per la implementació i disseny de cada una d'elles.

5.1 Cerca de materials i recursos

Pel fet que l'objectiu del projecte no és únicament el desenvolupament de programari, ha estat necessari seleccionar components físics per tal d'implementar la solució. En aquest cas són 2 els components bàsics que s'han identificat i fet servir: un cable de connexió USB a micro-USB i un mòdul de desenvolupament IoT. Si bé és cert que es menciona més amunt que es farà ús de la connexió USB directe de la placa, en ser un prototip no he cregut necessari fer-ho amb un cable especial que s'hi connecti, ja que la tecnologia i el protocol de comunicació USB serà el mateix tant fent servir directament un capçal USB a la placa base, com fent servir un port USB normal i corrent.

De les diferents opcions que hi ha al mercat de mòduls que permetin fer una comunicació IoT he triat la placa *PyCom FiPy* [12] (vegeu figura 4) pensada per a la microprogramació en *Python*, això implica que els programes que s'executen en aquestes plaques requereixen pocs recursos per a funcionar, ja que aquestes plaques es basen en un model de funcionament de baix consum energètic per durar amb una bateria el major temps possible. De totes maneres el consum de bateria en aquest projecte es pot ignorar,

ja que sempre tindrà una font d'alimentació. Concretament, aquesta placa té 22 pins d'entrada i sortida digitals, 1 pin analògic d'entrada i 3 pins de comunicació sèrie. La majoria dels pins digitals tenen capacitat PWM. Aquesta placa compta amb la possibilitat de connectar-se a la xarxa *LoRa* i a més té la capacitat de fer servir *WiFi*, *Bluetooth Low Energy* i *NB-IOT*, que bàsicament és semblant a la xarxa mòbil però pensada per a dispositius IoT. El motiu principal per escollir aquesta placa i no d'altres similars és a causa de la versatilitat que ofereix el mòdul. En ser un prototip el maquinari que utilitzi no ha de per què ser estrictament igual que el maquinari en cas que es tiri endavant el projecte en termes de producció, per tant, per no limitar l'abast ni posar-me traves a mi mateix he seleccionat aquesta placa.

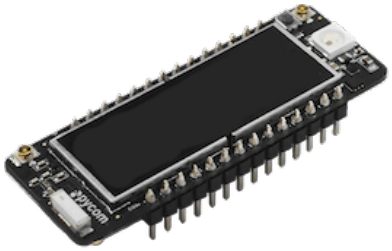


Fig. 4: Placa PyCom FiPy.

Pel que fa als recursos que es necessiten per a dur a terme el projecte, a banda del mateix mòdul s'han de tenir en compte 3 elements més: la xarxa o xarxes IoT per a comunicar, el servidor on estarà allotjat el backend i el dispositiu que és monitorarà (servidor o ordinador). Els dos últims tenen les seves pròpies seccions a l'article, però les xarxes IoT escollides les defineixo aquí sota.

5.1.1 Xarxes IoT escollides

Per poder enviar dades al backend és necessari seleccionar una xarxa de *LoRa* disponible, en el cas d'Espanya la més utilitzada i estesa és *The Things Network* [13]. *The Things Network* permet crear aplicacions on gestionar el flux d'informació provinent dels dispositius LoRa que s'hi registrin. Posteriorment per a poder fer ús d'aquestes dades hauran de ser enviades al servidor backend i emmagatzemades a una base de dades per poder realitzar les consultes que toquin amb l'objectiu de mostrar aquesta informació a l'aplicació. L'enviament d'aquestes dades al servidor es fa usant un *WebHook*, cada cop que una nova dada arriba a la xarxa *LoRa* s'envia un event prèviament definit a una API que s'escull a l'hora de configurar l'aplicació LoRa.

Inicialment, tenia intenció de fer ús de la xarxa *LoRa* exclusivament, però ja que a casa meva no tinc cobertura per aquesta xarxa he fet una ampliació al projecte afegint una altra opció de comunicació, *PyBytes* [14]. Pybytes és una plataforma d'Internet de les Coses (IoT) que utilitza la connexió WiFi per transferir informació. És especialment útil per evitar sobrecarregar la xarxa LoRa amb una gran quantitat de dades. Amb Pybytes, podem connectar dispositius IoT compatibles a una xarxa WiFi existent i transferir informació de manera eficient. Un altre motiu que m'ha impulsat en fer el canvi és que la tecnologia LoRa és ideal per a la comunicació de baix consum i llarg abast, però té una capacitat limitada per transmetre grans quantitats de dades. Amb l'ús de Pybytes i una connexió WiFi, podem aprofitar la velocitat i la capacitat de transmissió més àmplia d'aquesta xarxa per transferir la informació fent un ús mínim de la xarxa *LoRa*.

L'arquitectura de la solució que es descriu a aquest article es pot veure a continuació (vegeu figura 5)

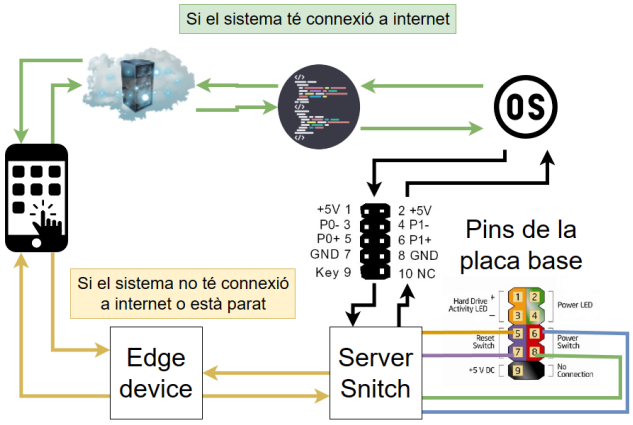


Fig. 5: Arquitectura de la solució

El camí verd indica la comunicació "normal"del sistema, quan hi ha una connexió a internet proporcionada per la xarxa d'on es troba el sistema. Per altra banda, el camí groc mostra com es comunicarà el sistema quan no hi hagi aquesta connexió a internet comentada anteriorment o quan el sistema estigui parat.

A més també es pot comprovar amb més detall les tres possibilitats per a la comunicació al següent diagrama (vegeu figura 6).

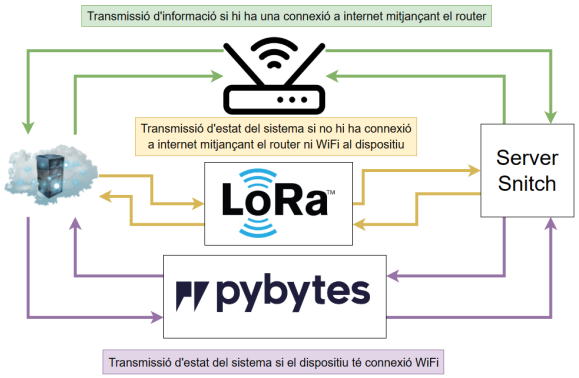


Fig. 6: Arquitectura de la solució amb detall de les xarxes IoT

A la següent taula (vegeu taula 2) es recullen els diferents modes de funcionament que té la solució de ServerSnitch des del punt de vista de la comunicació de dades de monitoratge.
O.S fa referència a sistema operatiu, mentre que S.S vol dir la informació que proporciona *ServerSnitch*

5.2 Desenvolupament del dispositiu

5.2.1 Software

A partir dels requisits que s'han de complir i des del punt de vista del que fa el dispositiu s'ha dissenyat i implementat el següent diagrama de casos (vegeu figura 7).

TAULA 1: MODES DE COMUNICACIÓ DE DADES

Sistema encés?	Sistema té connexió a internet?	Tipus d'info. disponible.	Mode d'enviament
SI	SI	O.S i S.S	Router
SI	NO	S.S	LoRa/PyBytes
NO	NO	S.S	LoRa/PyBytes

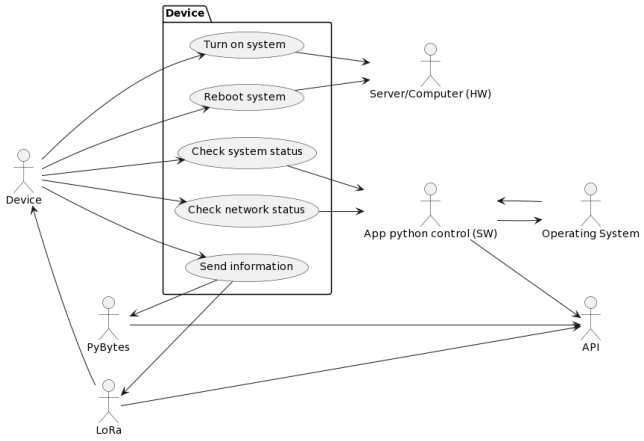


Fig. 7: Diagrama de casos d'ús del dispositiu.

Al diagrama es poden identificar diversos actors:

1. Device: Aquest actor fa referència al dispositiu FiPy pròpiament, el dispositiu estarà tota l'estona en un cicle infinit demanant l'estat de la xarxa per determinar si ha de ser ell el que ha d'enviar les dades que demanarà a l'App python de control o si pot delegar aquesta feina directament a l'App python.

2. LoRa/PyBytes: Aquests actors representen la possibilitat d'enviar les dades fent ús de la xarxa LoRa/PyBytes. A més en el cas de LoRa també representa la funcionalitat que permet fer ús de la xarxa LoRa per donar l'ordre al dispositiu d'arrancar o reiniciar el sistema.

3. Server/Computer: És l'abstracció física dels pins d'arrancada i reinici que es troben a la placa base (vegeu figura 1).

4. API: Actor que representa els mètodes que hi ha al backend per a enviar les dades des de l'App de control i a més, extreure les dades del cloud de LoRa per tal de tenir les dades encara que l'internet del sistema falli.

5. Operating System: Actor que representa el sistema operatiu del sistema, aquest ens proporciona la informació relativa als serveis que volem monitorar i dades com l'ús de CPU, memòria...

6. App python control: Aquest actor és el programa Python que actua al sistema per extreure la informació, a l'apartat *"Disseny del programa de monitoratge (App python control)"* hi ha més informació de com funciona.

Per comunicar amb *LoRa* hi ha dues configuracions possibles, OTAA i ABP [15]. Idealment i en l'entorn en el qual es troba el meu projecte la manera més correcta és OTAA, tot i que jo he hagut de decantar-me per ABP, ja que el mòdul *FiPy* no és capaç d'enviar ni rebre fent ús d'OTAA, aquesta decisió implica un enfocament més estàtic i menys flexible en termes de gestió de dispositius LoRa.

El motiu pel qual el dispositiu no és capaç de comunicar fent servir OTAA té indicis de ser causat per alguna incompatibilitat provocada per la combinació del maquinari LoRa que té el dispositiu, la versió de firmware i l'especificació de LoRa que es defineix al cloud de *LoRa*. Al diagrama de seqüència següent (vegeu figura 8) es pot comprovar com funciona el protocol d'autenticació ABP.

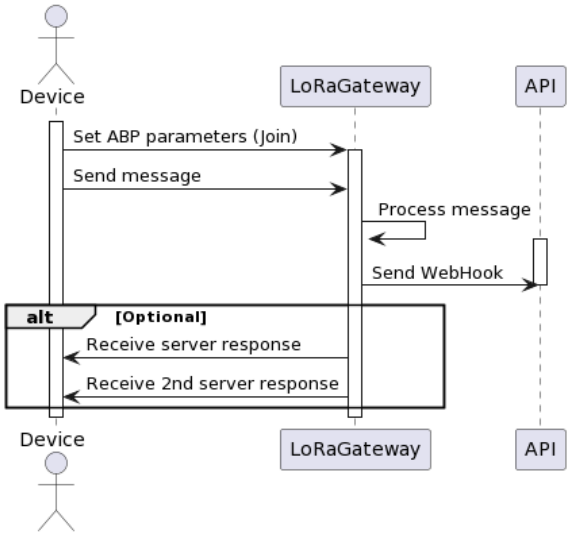


Fig. 8: Diagrama de seqüència del protocol d'autenticació ABP.

Pel que fa a les dades que es transmeten per la xarxa *LoRa* es representen per la següent taula, on 0x00 vol dir NO o Fals i 0x01 vol dir SÍ o Veritat:

TAULA 2: MODES DE COMUNICACIÓ DE DADES

Sistema encés?	Sistema té connexió a internet?	Sistema té. connexió local?
0x00/0x01	0x00/0x01	0x00/0x01

Les dades de retorn cap al dispositiu per a indicar si ha d'encendre o reiniciar el sistema és 0x00 per iniciar/parar i 0x01 per reiniciar.

5.2.2 Hardware

Per al disseny maquinari del dispositiu he fet servir una placa de proves, dos optoacobladors, dos diodes LED i 4 resistències.
Existeixen dos motius principals per usar optoacobladors:

- Aïllament galvànic: Els optoacobladors proporcionen un aïllament galvànic entre la part de control i la part controlada. Això vol dir que no hi ha connexió elèctrica directa entre els dos circuits, el que ofereix protecció contra sobretensions, soroll elèctric i possibles danys causats per defectes en una de les parts.

Aquest aïllament galvànic és especialment important quan es treballa amb components sensibles com ara plaques base.

- Seguretat: Els optoacobladors ajuden a millorar la seguretat del sistema. En emular els pins d'encès i reinici amb optoacobladors (des del punt de vista del dispositiu), es redueix el risc de sobrecarregar els circuits de la placa base. Això pot ser útil en situacions en què es requereixi controlar remotament l'encès o el reinici del sistema, ja que es poden utilitzar senyals de control segurs per evitar possibles accidents o danys.

El fet d'usar els dos díodes LED per emular els pins de l'ordinador té un raonament més simple: així es pot veure d'una manera immediata i visual si s'encén un dels díodes per indicar quina ordre a manat a executar l'aplicació i, per extensió el dispositiu.

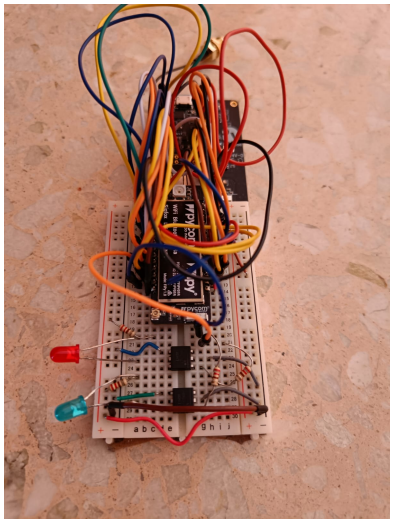


Fig. 9: Muntatge del circuit de la placa FiPy amb 2 LED per simular el funcionament.

5.2.3 Programa pel servidor/ordinador

El programa pel servidor/ordinador o "App Python de control"és el que s'encarrega de proveir al dispositiu amb la informació relativa a l'estat del sistema i de les comunicacions, sigui la xarxa LAN o l'accés a internet (WAN). També recol·lectarà la informació dels serveis deitjats i l'ús de *CPU*, *RAM* i *disc*. com el seu nom indica aquest programa s'executa al servidor i manté una comunicació amb el dispositiu fent ús de comunicació sèrie. Em referiré a aquest programa també com a "programa de monitoratge".

El diagrama de casos d'ús de l'App Python de control és el següent (vegeu figura 10).

Bàsicament, l'App Python de control espera una ordre del dispositiu FiPy, en funció del que li demani l'App Python de control haurà de retornar la informació corresponent (Si no hi ha internet o si es demana explícitament l'estat de l'internet), o bé recol·lectar i enviar les dades directament a l'API.

En el següent diagrama d'estats es pot veure millor com funciona el programa (vegeu figura 11).

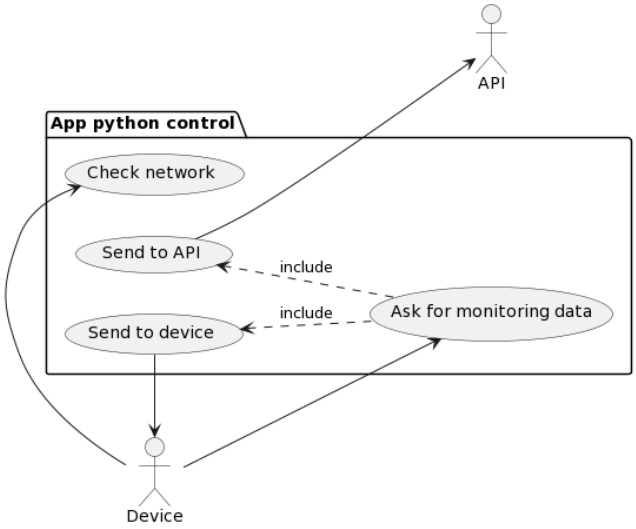


Fig. 10: Diagrama de casos d'ús del programa del servidor.

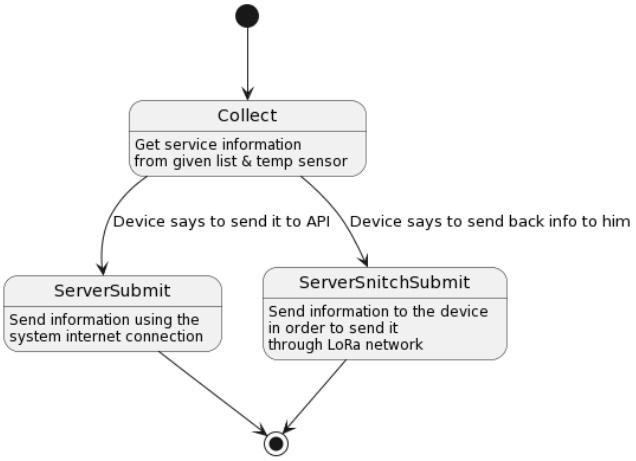


Fig. 11: Diagrama d'estat del programa del servidor.

5.3 Desenvolupament del backend

El desenvolupament del backend està separat en dues parts, la part de disseny i implementació de la base de dades i la part de disseny i implementació de la API per a consultar i enviar dades.

5.3.1 Disseny i implementació de la base de dades

La base de dades l'he implementat fent ús de MySQL [16] un producte lliure i molt provat. Al diagrama de la base de dades es pot veure l'estructura d'aquesta (vegeu figura 12), la descripció de les diferents taules la faig a continuació:

Taula "ActionLog": Aquesta taula guarda un registre de les ordres de parada i reinici que se li envien des de l'aplicació als dispositius,

- Camps: `UserId` (identificador de l'usuari), `DeviceId` (identificador del dispositiu), `TimeStamp` (data i hora de l'acció), `Action` (acció realitzada, pot ser "start"o "reboot").

- Relacions: La clau primària d'aquesta taula està composta per `UserId` i `DeviceId`, la qual cosa indica que un usuari pot tenir múltiples accions registrades per a diferents dispositius. A més, hi ha una

clau externa `DeviceAction` que es vincula al camp `DeviceId` de la taula "Device", la qual cosa significa que cada acció està associada a un dispositiu específic.

Taula "DataValue": Aquesta taula guarda els registres dels diferents serveis que és monitoren al sistema, conté el nom d'on prové el valor i el valor en si, juntament amb la data i hora d'aquest valor i el tipus de dada que és.

- Camps: `Id` (identificador del valor), `Name` (nom del valor), `Value` (valor), `Type` (tipus de valor, pot ser "bool", ïnt", "float"o "string"), `TimeStamp` (data i hora del valor), `DeviceId` (identificador del dispositiu).
- Relacions: La clau primària és el camp `Id`. També hi ha una clau externa `DeviceDataValue` que es relaciona amb el camp `DeviceId` de la taula "Device", la qual cosa indica que cada valor de dades està associat a un dispositiu específic.

Taula "Device": Aquesta taula guarda tots els dispositius del sistema.

- Camps: `EUI` (identificador únic del dispositiu), `MAC` (adreça MAC del dispositiu), `Alias` (àlies o nom del dispositiu), `Description` (descripció del dispositiu), `GroupId` (identificador del grup al qual pertany el dispositiu), `UserId` (identificador de l'usuari propietari del dispositiu).
- Relacions: La clau primària és el camp `EUI`, que és un identificador únic per a cada dispositiu. A més, hi ha claus úniques per a `Alias` i `MAC`, la qual cosa garanteix que no hi hagi duplicats. Hi ha dues claus externes, `DeviceGroup` i `DeviceCreator`, que es relacionen amb els camps `GroupId` i `UserId` de les taules "DeviceGroupï Üser", respectivament.

Taula "DeviceGroup": Aquesta taula relaciona els dispositius amb els diferents grups.

- Camps: `Id` (identificador del grup), `Alias` (àlies o nom del grup), `Description` (descripció del grup), `Location` (ubicació del grup).
- Relacions: La clau primària és el camp `Id`. A més, el camp `Alias` té una restricció de clau única per evitar duplicats, per tant, un dispositiu només pot pertanyer a un grup.

Taula "User": Aquesta taula guarda els usuaris del sistema.

- Camps: `Id` (identificador de l'usuari), `User` (nom d'usuari), `PwdHash` (hash de la contrasenya), `Name` (nom de l'usuari).
- Relacions: La clau primària és el camp `Id`. A més, el camp `User` té una restricció de clau única per garantir la unicitat dels noms d'usuari, també destaca que la contrasenya no s'emmagatzema directament, sinó que s'emmagatzema un hash que es calcula a l'aplicació web, per tant, en cap moment arriba la contrasenya de l'usuari a l'API.

Taula "UserDeviceGroup": Aquesta taula relaciona els usuaris amb els diferents grups.

- Camps: `DeviceGropuId` (identificador del grup de dispositius), `UserId` (identificador de l'usuari).
- Relacions: La clau primària està composta per `DeviceGropuId` i `UserId`, la qual cosa indica una relació de molts a molts entre usuaris i grups de dispositius.

Pel que fa a les restriccions i les claus foranies, s'han definit diverses per garantir la integritat referencial i la consistència de les dades.

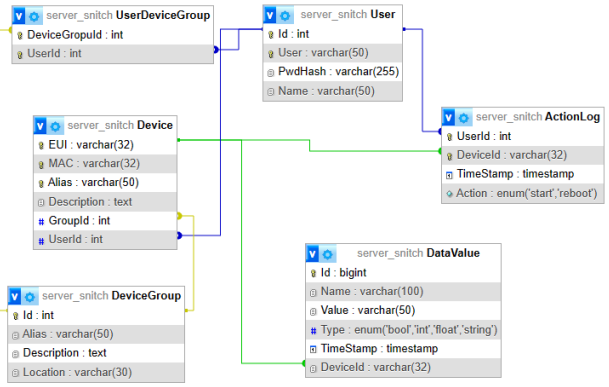


Fig. 12: Diagrama de la base de dades amb les relacions entre taules.

5.3.2 Implementació de la API

L'API agrupa els mètodes remots als quals es fan peticions per tal de fer les operacions de guardar, consultar i eliminar les dades de l'aplicació. S'han incorporat els mètodes d'emmagatzematge de la informació i també tots aquells mètodes que caldran per poder visualitzar la informació des de l'aplicació final de l'usuari. Pel fet que l'aplicació d'usuari fa ús directe de l'API per mostrar la informació he decidit representar-ho només amb un diagrama. Encara que l'API al diagrama no crida a cap mètode dono per entès que s'utilitzen els mateixos, però de manera interna i que es retornen les dades a l'aplicació quan aquesta fa les peticions. A més el mètode que fa servir l'App Python de control per a pujar les dades es veu definit per la relació entre el mètode "Send to API"del diagrama *"Diagrama de casos d'ús del programa del servidor"*. També es pot veure una relació reflectiva entre l'API i el cloud de LoRa, ja que l'API és capaç d'enviar les ordres d'arrancada i reinici mitjançant el cloud. El diagrama, per tant, queda així (vegeu figura 13):

L'estructura de dades amb la informació dels serveis i la informació relativa al sistema que s'enviarà a la API quan hi hagi internet segueix l'estàndard JSON i té la següent estructura:

```
{
  ''services": {
    ''msedge.exe": {
      ''name": ''msedge.exe",
      ''cpu_percent": 0.05,
      ''memory_rss": 50003968
    },
    ''pycharm64.exe": {
```

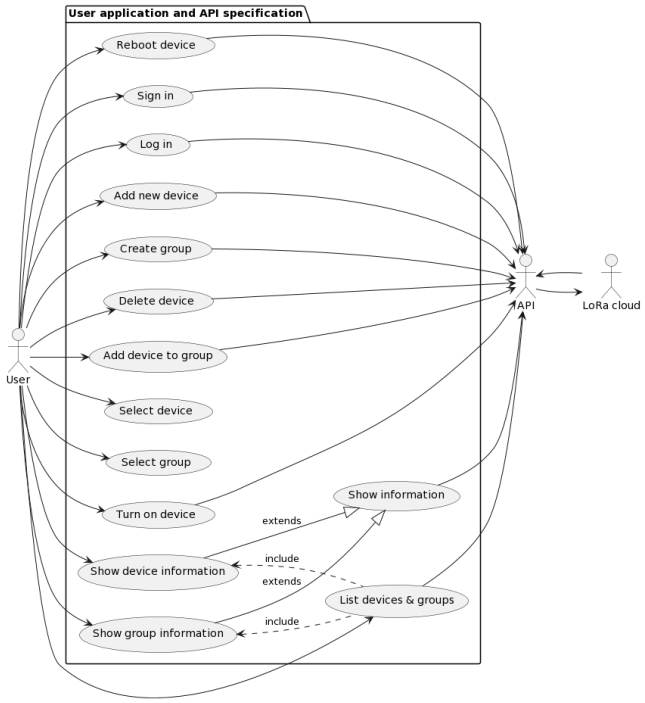


Fig. 13: Diagrama de casos d'ús de l'aplicació d'usuari i API.

```
        ``name": ``pycharm64.exe",
        ``cpu_percent": 0.20,
        ``memory_rss": 1508806656
      },
      ``python.exe": {
        ``name": ``python.exe",
        ``cpu_percent": 0.05,
        ``memory_rss": 23363584
      }
    },
    ``load_avg": 20.0,
    ``disk": 50.9,
    ``mem": 47.8,
    ``wan": true,
    ``lan": true,
    ``eui": ``123456789"
}
```

5.4 Desenvolupament de l'aplicació

Per desenvolupar l'aplicació web i l'API he fet servir el framework Flask [17] per a Python. He escollit Flask com a framework web pel meu projecte per diverses raons. En primer lloc, Flask és un framework lleuger i fàcil d'aprendre, el que em permet desenvolupar ràpidament la meva aplicació web. La seva sintaxi senzilla i el seu enfocament minimalista em permeten centrar-me en la lògica de la meva aplicació sense complicacions innecessàries. A més és fàcilment escalable, compta amb diversitat de plugins per ampliar-lo i una gran comunitat.

Inici de sessió i registre

Els formularis d'inici de sessió (vegeu figura 14) i de registre (vegeu figura 15). En aquest cas per al registre he comptat només amb els camps: Nom, nom d'usuari i contrasenya. Considero que ara mateix no calen més tot i que en el cas que el projecte continuï tirant endavant en un futur caldrà ampliar-lo.

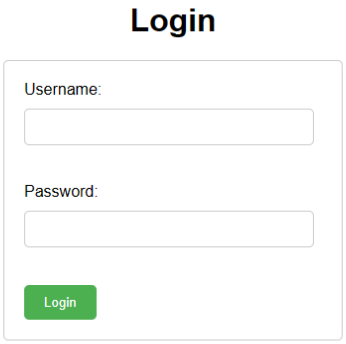


Fig. 14: Formulari d'inici de sessió.

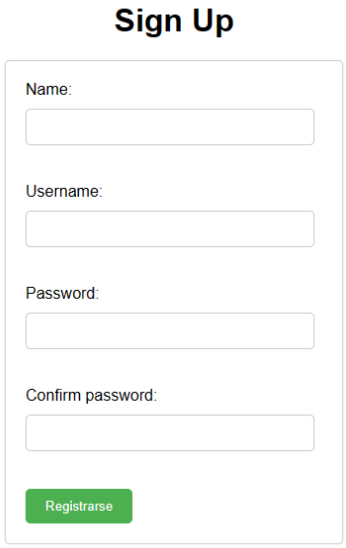


Fig. 15: Formulari de registre.

Creació i visualització de grups

La creació de grups (vegeu figura 16) conté també pocs camps, ja que per crear un grup només necessites donar-li un nom i posar una descripció si es considera necessari. A l'iniciar sessió l'usuari serà redirigit a una secció on se li llistaran tots els grups i on podrà fer clic a un per veure els dispositius que conté el grup.

La visualització de grups (vegeu figura 17) presenta els grups diferents columnes en funció del número de grups a mostrar, i tenen diferents codis de color, quan tots els dispositius del grup es troben en bon estat el grup surt en verd i si algun estan malament surt en vermell.

Creació i visualització de dispositius

La creació de dispositius (vegeu figura 18) té una mica més de treball que els altres formularis de l'aplicació, ja que és l'essència d'aquesta. Els camps en aquest cas són EUI (identificador únic de dispositius LoRa), MAC, Alies

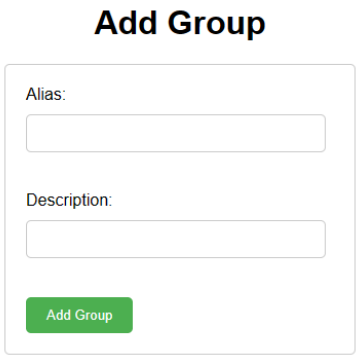


Fig. 16: Formulari de creació de grup.

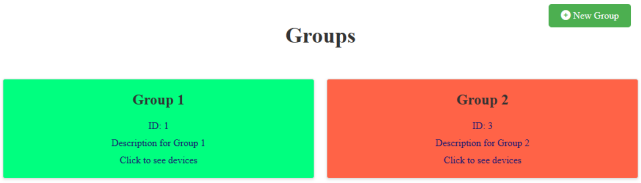


Fig. 17: Llista de grups de l'usuari.

i descripció. El camp "Alies"està pensat específicament per anomenar al dispositiu amb el nom del sistema que estarà monitorant, així és més fàcil d'identificar. És molt important que l'identificador EUI es defineixi correctament, ja que la comunicació LoRa depèn molt d'aquest identificador.

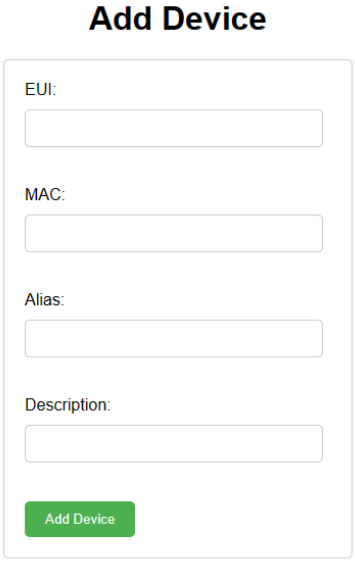


Fig. 18: Formulari de creació de dispositius.

Per a la visualització dels dispositius (vegeu figura 19) al clicar un grup he disposat els dispositius també en dues columnes i segueixen la mateixa configuració de colors que els grups. Si un dispositiu té tots els serveis arrancats es llistarà en verd, si no en vermell. A més cada dispositiu té tres botons, un per enviar el senyal d'encendre el servidor al dispositiu, un per enviar el senyal de reiniciar i un últim per mostrar les gràfiques i l'estat dels serveis monitorats.

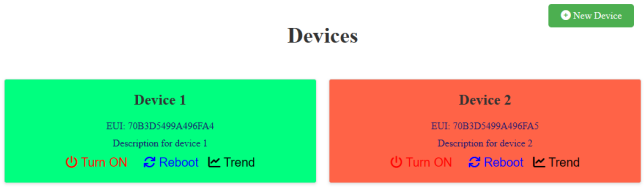


Fig. 19: Llista de dispositius d'un grup.

Visualització de serveis monitorats En el cas dels serveis monitorats (vegeu figura 20) he fet servir dos conceptes: per a indicar si els serveis estan correctes així com si el sistema està funcionant correctament he fet servir una llista de punts amb *ticks i creus*, els ticks indiquen que el servei o sistema estan bé i les creus que no; a més per cada servei monitorat es mostren 2 dades, el percentatge d'ús de CPU i la memòria RAM que utilitza el procés. Es mostren gràfiques amb l'evolució temporal dels serveis i estat dels recursos del sistema.

Fig. 20: Llista gràfiques amb l'estat dels processos.

5.5 Integració de les parts

Integració dispositiu i programa de monitoratge
Per integrar el dispositiu i el programa de monitoratge he fet servir comunicació sèrie. La comunicació sèrie és una forma eficient i fiable de transmetre dades entre el dispositiu i el programa mitjançant un cable sèrie. Aquesta opció és ideal per al meu cas, ja que em permet establir una connexió directa i bidireccional entre els dos components. Utilitzant un protocol de comunicació estàndard, puc enviar i rebre informació de manera estructurada, com ara dades de serveis o ordres de control.

Integració dispositiu amb xarxes LoRa i PyBytes
Com ja he explicat més amunt per a la integració de les dues xarxes IoT que faig servir he escollit *The Things Network* per LoRa i *PyCom PyBytes* per a la comunicació PyBytes pròpiament, tot i que no he hagut de configurar-ho aquesta última fa ús del protocol MQTT per a comunicar.

Integració de les xarxes LoRa i PyBytes amb l'API
Per a integrar les dades provinents de les xarxes LoRa i PyBytes he fet ús de què s'anomena *WebHooks*. Un "WebHook"és un mecanisme que permet la comunicació bidireccional entre una xarxa LoRa i una aplicació externa a través d'Internet. Un webhook en LoRa s'utilitza per enviar dades o notificacions en temps real des dels dispositius LoRa a una aplicació externa, o bé per rebre instruccions i comandes de l'aplicació externa cap als dispositius LoRa. D'aquesta forma cada cop que la xarxa LoRa o la xarxa PyBytes reben

un missatge el reencaminen fent ús d'una crida HTTP cap al seu corresponent destí a l'API per tal d'incorporar aquestes dades.

Integració de l'aplicació amb l'API

La integració de l'aplicació amb l'API es realitza de manera directa gràcies als mecanismes que proporciona l'API per crear les vistes en HTML a partir de plantilles. Aquesta integració és possible gràcies a les funcionalitats que ofereix l'API, com ara la capacitat de gestionar rutes, processar les peticions HTTP i generar les respostes en format HTML. Aquesta integració directa permet una interacció fluida entre l'aplicació i l'API, ja que l'API s'encarrega de la generació de les vistes en HTML basant-se en les plantilles i les dades proporcionades per l'aplicació. A més, utilitzant plantilles, puc mantenir una estructura i un disseny coherents en totes les pàgines de l'aplicació, facilitant la seva gestió i actualització.

5.6 Test del sistema

Per provar el sistema, he usat proves manuals d'integració de punt a punt, ja que la complexitat del sistema no m'ha permès implementar una estructura de proves adequada. En aquest sentit, he provat primer les parts independents del sistema per assegurar-me que funcionen correctament de manera individual. Això implica comprovar que cada component o mòdul realitza les seves funcionalitats de manera adequada.

Una vegada he provat les parts independents i he verificat que estan funcionant correctament, he procedit a provar el flux complet del sistema. Això implica dur a terme proves que simulin l'ús real del sistema, des de l'inici fins a la finalització del procés. En aquestes proves, m'he centrat a comprovar que els components s'integren de manera adequada, que les dades es transmeten correctament entre ells i que el resultat final és el desitjat.

Encara que les proves manuals de punt a punt poden ser més laborioses i requereixen més temps, en aquest cas han estat necessàries donada la complexitat del sistema. No obstant això, són una opció viable per provar el funcionament global del sistema quan no és factible o pràctic implementar una estructura de proves automatitzades. L'objectiu principal és garantir que el sistema funcioni de manera satisfactòria i que compleixi amb els requisits establerts.

Per la prova final de punt m'he ajudat del diagrama de seqüència complet del sistema que es pot veure seguidament (vegeu figura 21):

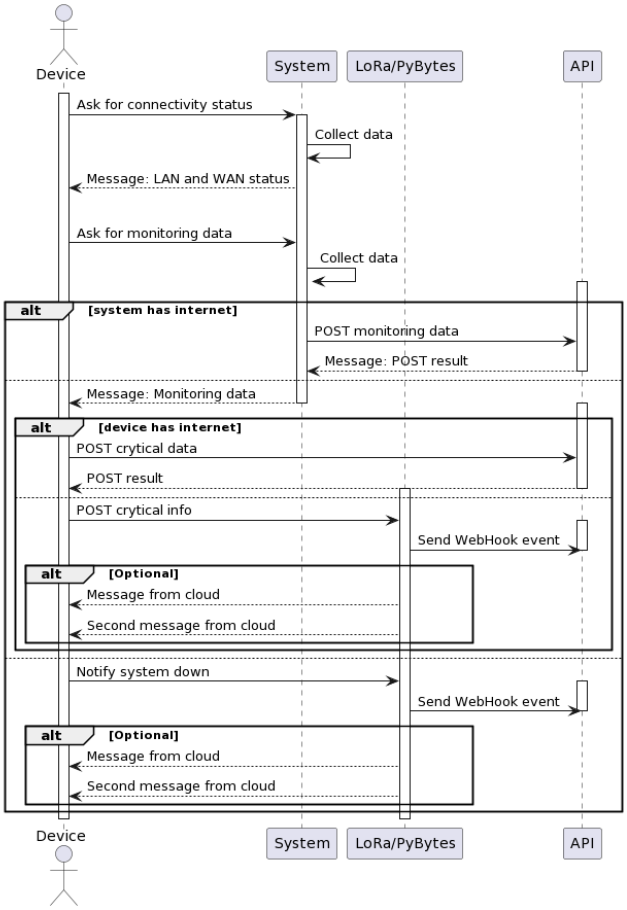


Fig. 21: Diagrama de seqüència del funcionament de tota la solució.

6 Resultats

Els resultats del desenvolupament del meu projecte han estat assolits de manera satisfactòria malgrat haver tingut multitud de problemes a l'hora de fer ús de la xarxa LoRa pel motiu que he comentat anteriorment. Amb aquest projecte he après el canvi de paradigma que comporta treballar amb dispositius IoT, ja que implica treballar més aviat amb una programació orientada a esdeveniments i seqüencial que no pas orientada a objectes com acostuma a ser més comú avui en dia. El resultat que sembla més evident és l'aplicació web que he desenvolupat (vegeu apartat 5.4), pel fet que és la porta d'entrada a tota la solució desenvolupada, tot i que el que ha comportat més feina i ha estat el nucli del projecte han estat les parts de desenvolupament del dispositiu (vegeu apartat 5.2) i el desenvolupament de l'aplicació de control (vegeu apartat 5.2.3). Considero que el projecte ha complert amb el que es buscava, fer una solució capaç de monitorar servidors/ordinadors fent ús de la xarxa IoT per enviar l'estat d'aquests sempre que no hi hagi connexió a internet. El que no s'ha complert ha estat la planificació inicial, ja que durant el transcurs del projecte han sorgit problemes que han requerit més temps del previst per solucionar-los, així doncs la planificació final es pot consultar als annexos (vegeu annex 2).

6.1 Tasques que han implicat més temps del previst

La tasca relacionada amb la configuració i desenvolupament del dispositiu (Desenv. del disp.) i la tasca relacionada amb la integració del dispositiu a la xarxa LoRa i posterior enviament de les dades a l'API han estat les més costoses en temps, i les que han provocat que el temps previst no s'hagi complit. A l'annex es pot trobar una taula amb el resum dels motius dels enderreriments (vegeu annex 4).

7 Conclusions

Per acabar considero que el projecte compleix de sobres els requisits amb els quals ha de comptar un treball de final de grau i que s'han assolit els objectius que he definit (vegeu

apartat 3). Els coneixements que m'ha aportat fer el treball em serviran de cara al futur per a poder executar projectes en l'àmbit de les tecnologies IoT, a més la disciplina que es necessita per a poder documentar el procés de desenvolupament em servirà per a aplicar-la en qualsevol altre àmbit del desenvolupament de programari.
Tot i que s'han assolit els objectius i s'han complert els requisits queden obertes possibles ampliacions que serien molt interessants de fer:

1. Ampliar la funcionalitat amb la xarxa nb-IoT o LTE.
2. Fer un sistema de redundància on diferents dispositius que estiguin a prop puguin comunicar-se per WiFi o Bluetooth i si un no acaba de funcionar bé delegar les dades a un altre.
3. Oferir la possibilitat de personalitzar de manera en línia quins processos s'han de monitorar i com mostrar-los.
4. Solucionar el problema amb OTAA per fer-lo servir com a protocol de comunicació LoRa.
5. Afegir una manera de comprovar l'estat d'un SAI associat al sistema.
6. Aprofitar la possibilitat del mòdul FiPy per afegir una bateria en cas de pèrdua d'energia.

Agraïments

Vull donar les gràcies a en Marc Codina Barberà la seva tutorització i per proporcionar-me el material necessari per desenvolupar el meu treball de final de grau. Marc, La teva ajuda ha estat essencial per a la seva realització i el teu coneixement en l'àmbit IoT m'ha servit per a resoldre alguns dels problemes que m'he trobat a l'hora d'integrar el dispositiu a la xarxa.

Estic molt agraït per la teva orientació experta i el teu suport constant. Sense la teva dedicació i coneixement, no hauria estat possible obtenir els resultats que he aconseguit.

Repositoris de codi

Aqui estàn els repositoris de codi on porto el control de versions:

1. https://github.com/kolibriconk/ServerSnitch-api
2. https://github.com/kolibriconk/ServerSnitch-monitor
3. https://github.com/kolibriconk/ServerSnitch-fipy

Referències

[1] *Motherboard pins*. TechNewsToday. [Online]. Available: https://www.technewstoday.com/wp-content/uploads/2022/03/f-panel.webp

[2] *Motherboard F_USB*. Amazon. [Online]. Available: https://m.media-amazon.com/images/I/71CMg4bP9WL._AC_SX679_.jpg

[3] *Nagios*. nagios.com. [Online]. Available: https://www.nagios.org/

[4] *EnerMon*. DiogoSantos and Joao C. Ferreira. [Online]. Available: https://www.mdpi.com/2071-1050/11/19/5355

[5] *LoRa Alliance*. LoRaWAN®. [Specification]. Available: https://lora-alliance.org/resource-hub/lorawan-specification-v1-0

[6] *WeMos ESP8266 Remote PC Switch*. [Online]. Available: https://www.hackster.io/zvonko-bockaj/wemos-esp8266-remote-pc-switch-062c7a

[7] *OASIS*. MQTT. OASIS Standard. Available: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

[8] *Python Software Foundation*. Python. [Software]. Available: https://www.python.org/

[9] *Code & Pepper*. Kanban Tool. [Software]. Available: https://kanbantool.com/es/metodologia-kanban

[10] *Trello inc.*. Trello. [Software]. Available: https://trello.com/

[11] *Microsoft Corporation.*. Microsoft Project. [Software]. Available: https://www.microsoft.com/en-us/microsoft-365/project/project-and-portfolio-management-software

[12] *PyCom FiPy. [Hardware]. Available: https://docs.pycom.io/datasheets/development/fipy/*

[13] *The Things Network. [Online]. Available: https://www.thethingsnetwork.org/*

[14] *PyCom*. PyBytes. [Software]. Available: https://docs.pycom.io/pybytes/

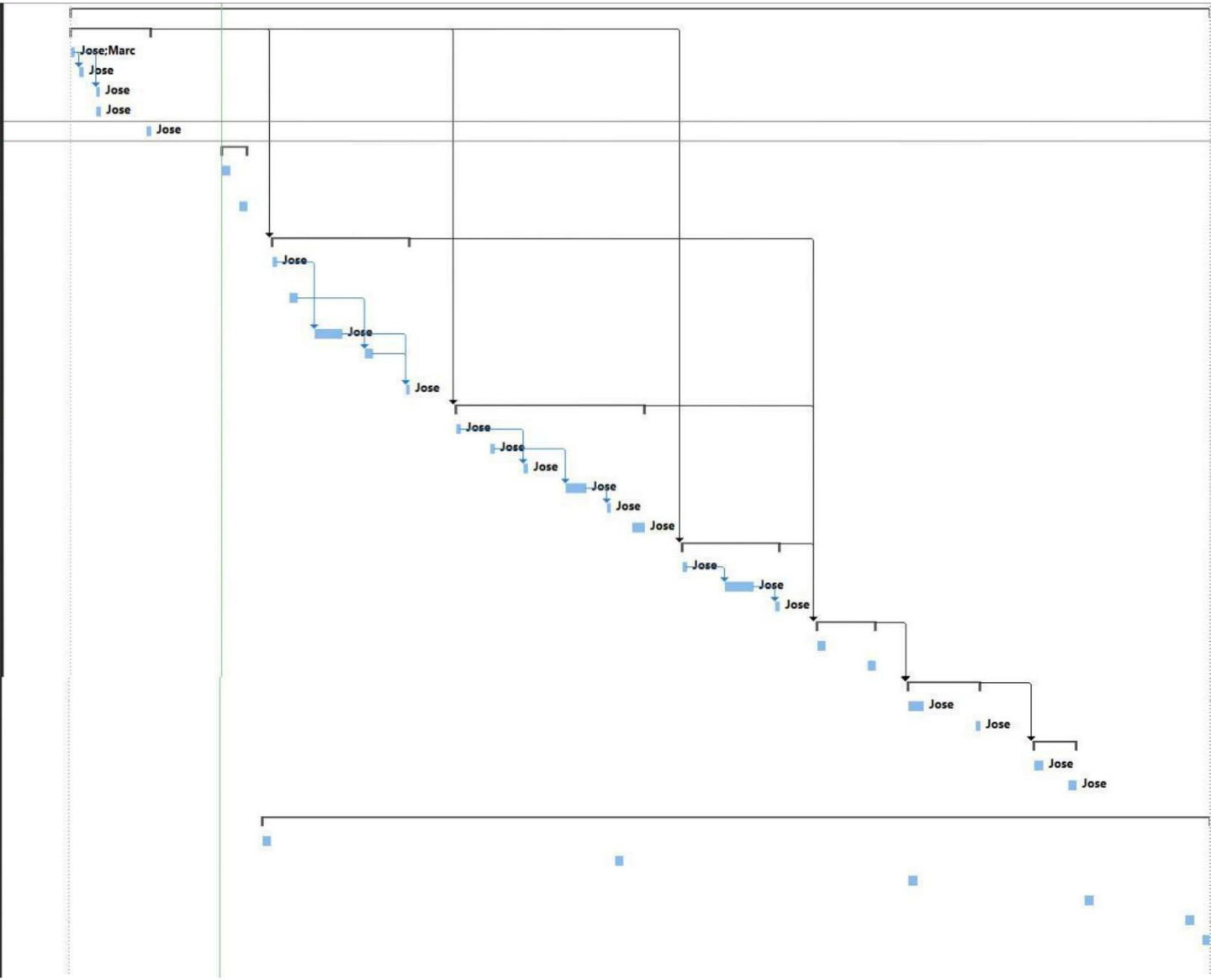
[15] *OTAA vs ABP. The things Network. [Online]. Available: https://www.thethingsindustries.com/docs/devices/abp-vs-otaa/*

[16] *MySQL. [Online]. Available: https://www.mysql.com*

[17] *Flask. The Pallets Projects. [Online]. Available: https://palletsprojects.com/p/flask/*

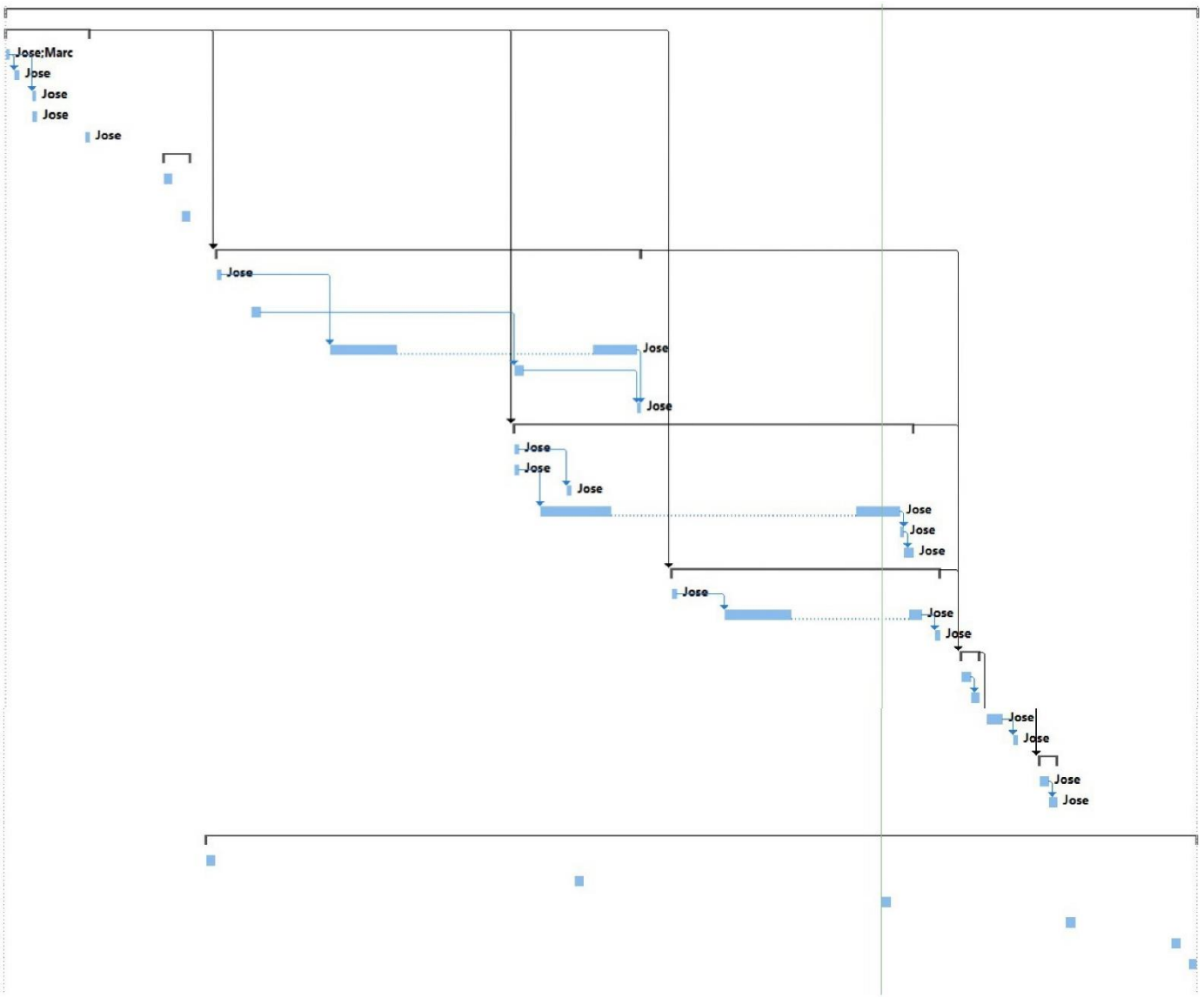
Annex 1 – Planificació inicial

Server Snitch development.	**300 horas**	**vie 17/02/23**
Definició de requeriments	**13 horas**	**vie 17/02/23**
Reunió kick off	1 hora	vie 17/02/23
Requisits funcionals disp.	3 horas	sáb 18/02/23
Requisits funcionals backend	2 horas	lun 20/02/23
Requisits funcionals app	3 horas	lun 20/02/23
Definició diagrama de tasques	4 horas	dom 26/02/23
Cerca de material i recursos	**10 horas**	**mar 07/03/23**
Cerca de plataforma per dispositiu (Arduino/RSPB/...)	5 horas	mar 07/03/23
Cerca de plataforma IoT per comunicar	5 horas	jue 09/03/23
Desenv. Dispositiu	**65 horas**	**lun 13/03/23**
Disseny d'esquemes i connexions	4 horas	lun 13/03/23
Disseny del programa pel servidor	4 horas	mié 15/03/23
Desenv. del disp.	25 horas	sáb 18/03/23
Desenv. Del programa pel servidor	30 horas	vie 24/03/23
Test del dispositiu	2 horas	mié 29/03/23
Backend development.	**44 horas**	**mar 04/04/23**
Disseny de BBDD	4 horas	mar 04/04/23
Disseny del backend	4 horas	sáb 08/04/23
Implementació BBDD	4 horas	mié 12/04/23
Desenv. backend	20 horas	lun 17/04/23
Test del backend	2 horas	sáb 22/04/23
Configuració del cloud	10 horas	mar 25/04/23
Desenv. aplicació	**33 horas**	**lun 01/05/23**
Disseny de la interfície	4 horas	lun 01/05/23
Desenv. aplicació	25 horas	sáb 06/05/23
Test de l'aplicació	4 horas	vie 12/05/23
Integració de les parts	**20 horas**	**mié 17/05/23**
Connexió de l'aplicació	10 horas	mié 17/05/23
Connexió del disp.	10 horas	mar 23/05/23
Test del sistema	**19 horas**	**dom 28/05/23**
Test "end to end"	15 horas	dom 28/05/23
Test d'arrancada/parada	4 horas	lun 05/06/23
Final del projecte	**16 horas**	**lun 12/06/23**
Evaluació del projecte	8 horas	lun 12/06/23
Definició de futures ampliacions	8 horas	vie 16/06/23
Redacció d'informes	**80 horas**	**dom 12/03/23**
Informe inicial	10 horas	dom 12/03/23
Informe de progrés I	10 horas	dom 23/04/23
Informe de progrés II	10 horas	dom 28/05/23
Informe final	20 horas	dom 18/06/23
Presentació del projecte	10 horas	vie 30/06/23
Dossier final	20 horas	dom 02/07/23



Annex 2 – Planificació final

Server Snitch development.	**505 horas**	**vie 17/02/23**
Definició de requeriments	**13 horas**	**vie 17/02/23**
Reunió kick off	1 hora	vie 17/02/23
Requisits funcionals disp.	3 horas	sáb 18/02/23
Requisits funcionals backend	2 horas	lun 20/02/23
Requisits funcionals app	3 horas	lun 20/02/23
Definició diagrama de tasques	4 horas	dom 26/02/23
Cerca de material i recursos	**10 horas**	**mar 07/03/23**
Cerca de plataforma per dispositiu (Arduino/RSPB/...)	5 horas	mar 07/03/23
Cerca de plataforma IoT per comunicar	5 horas	jue 09/03/23
Desenv. Dispositiu	**140 horas**	**lun 13/03/23**
Disseny d'esquemes i connexions	4 horas	lun 13/03/23
Disseny del programa pel servidor	4 horas	vie 17/03/23
Desenv. del disp.	100 horas	dom 26/03/23
Desenv. Del programa pel servidor	30 horas	dom 16/04/23
Test del dispositiu	2 horas	dom 30/04/23
Backend development.	**128 horas**	**dom 16/04/23**
Disseny de BBDD	4 horas	dom 16/04/23
Disseny del backend	4 horas	dom 16/04/23
Implementació BBDD	4 horas	sáb 22/04/23
Desenv. backend	104 horas	mié 19/04/23
Test del backend	2 horas	mar 30/05/23
Configuració del cloud	10 horas	mar 30/05/23
Desenv. aplicació	**79 horas**	**jue 04/05/23**
Disseny de la interfície	4 horas	jue 04/05/23
Desenv. aplicació	71 horas	mié 10/05/23
Test de l'aplicació	4 horas	sáb 03/06/23
Integració de les parts	**20 horas**	**mar 06/06/23**
Connexió de l'aplicació	10 horas	mar 06/06/23
Connexió del disp.	10 horas	mié 07/06/23
Test "end to end"	15 horas	vie 09/06/23
Test d'arrancada/parada	4 horas	lun 12/06/23
Final del projecte	**16 horas**	**jue 15/06/23**
Evaluació del projecte	8 horas	jue 15/06/23
Definició de futures ampliacions	8 horas	vie 16/06/23
Redacció d'informes	**80 horas**	**dom 12/03/23**
Informe inicial	10 horas	dom 12/03/23
Informe de progrés I	10 horas	dom 23/04/23
Informe de progrés II	10 horas	dom 28/05/23
Informe final	20 horas	dom 18/06/23
Presentació del projecte	10 horas	vie 30/06/23
Dossier final	20 horas	dom 02/07/23



Annex 3 - Definició de requisits de la solució

Requisits funcionals dispositiu FiPy

ID	Descripció	Prioritat	Objectius
RF_disp_iniciar	El dispositiu ha de ser capaç d'iniciar el sistema rebent l'ordre des de la xarxa LoRa.	4	1, 3
RF_disp _reiniciar	El dispositiu ha de ser capaç de reiniciar el sistema rebent l'ordre des de la xarxa LoRa.	4	1, 3
RF_disp _comprovar_xarxa	El dispositiu ha de ser capaç de demanar a l'App Python de control l'estat de la xarxa del sistema per determinar accions a realitzar.	4	2, 3
RF_disp _estat_sistema	El dispositiu ha de ser capaç de demanar a l'App Python de control les dades relacionades amb els serveis a monitoritzar prèviament definits i dades relacionades amb el sistema com ús de CPU mitja, RAM, etc.	4	2, 3
RF_disp _enviar_lora	El dispositiu ha de ser capaç d'enviar les dades que provenen de l'App Python de control fent ús de la xarxa LoRa.	4	2, 3
RF_disp _comunicar_app_py	El dispositiu ha de ser capaç de comunicar-se de manera bidireccional amb l'App Python de control.	4	2, 3

Requisits funcionals App Python de control

ID	Descripció	Prioritat	Objectius
RF_app_py_comunicar _dev	L'App Python ha de ser capaç de comunicar-se de manera bidireccional amb el dispositiu FiPy.	4	2
RF_app_py _estat_xarxa	L'App Python ha de ser capaç de consultar l'estat de la xarxa del sistema.	4	2
RF_app_py _consulta_serveis	L'App Python ha de ser capaç de consultar l'ús de CPU, memòria i estat dels serveis a monitoritzar.	4	2
RF_app_py _consulta_sistema	L'App Python ha de ser capaç de consultar l'ús de CPU mitjana, memòria del sistema en general.	4	2
RF_app_py _enviar_api	L'App Python ha de ser capaç d'enviar la informació dels serveis monitoritzats i la informació del sistema a la API mitjançant internet.	4	2
RF_app_py _enviar_dispositiu	L'App Python ha de ser capaç d'enviar la informació dels serveis monitoritzats i la informació del sistema al dispositiu fent ús de la comunicació del requisit "RF_comunicar_dev".	4	2

Requisits funcionals backend (API)

ID	Descripció	Prioritat	Objectius
RF_api_iniciar	L'API ha de definir un mètode GET per donar l'ordre a un dispositiu o grup de dispositius que encengui el sistema fent ús de la xarxa LoRa.	4	1, 5
RF_api_reiniciar	L'API ha de definir un mètode GET per donar l'ordre a un dispositiu o grup de dispositius que reinicïi el sistema fent ús de la xarxa LoRa.	4	1, 5
RF_api_login	L'API ha de definir un mètode POST per iniciar sessió al sistema.	2	4
RF_api_register	L'API ha de definir un mètode POST per registrar-se al sistema.	2	4
RF_api_alta_disp	L'API ha de definir un mètode POST per donar d'alta un dispositiu al sistema.	3	4

RF_api_alta_grup	L'API ha de definir un mètode POST per donar d'alta un grup al sistema.	1	4
RF_api_afegir_a_grup	L'API ha de definir un mètode POST per afegir un dispositiu a un grup.	2	4
RF_api_info_disp	L'API ha de definir un mètode GET per seleccionar un dispositiu del sistema i veure operacions a realitzar.	2	4
RF_api_info_grup	L'API ha de definir un mètode GET per seleccionar un grup del sistema i veure operacions a realitzar.	1	4
RF_api_llista_grups	L'API ha de definir un mètode GET per mostrar una llista de grups.	2	4
RF_api_llista_disp	L'API ha de definir un mètode GET per mostrar una llista de dispositius d'un grup.	2	4
RF_api_dades_monitor	L'API ha de definir un mètode POST per rebre les dades de l'App Python de control i emmagatzemar-les.	4	4
RF_base_dades	La solució ha de tenir una base de dades on poder emmagatzemar tota la informació que calgui.	4	4

Requisits funcionals aplicació d'usuari

ID	Descripció	Prioritat	Objectius
RF_usuari_iniciar	L'aplicació ha tenir una opció per iniciar un dispositiu remotament.	4	1, 5
RF_usuari_reiniciar	L'aplicació ha tenir una opció per reiniciar un dispositiu remotament.	4	1, 5
RF_usuari_login	L'aplicació ha tenir una opció per iniciar sessió al sistema i poder-la utilitzar.	2	4
RF_usuari_register	L'aplicació ha tenir una opció per poder registrar-se al sistema.	2	4
RF_usuari_alta_disp	L'aplicació ha tenir una opció per donar d'alta un dispositiu nou.	3	4
RF_usuari_alta_grup	L'aplicació ha tenir una opció per donar d'alta un grup.	1	4
RF_usuari_afegir_a_grup	L'aplicació ha tenir una opció per afegir un dispositiu a un grup.	2	4
RF_usuari_info_disp	L'aplicació ha tenir una opció per mostrar la informació d'un dispositiu.	2	4
RF_usuari_info_grup	L'aplicació ha tenir una opció per mostrar la informació d'un grup.	1	4
RF_usuari_llista_grups	L'aplicació ha tenir una opció per mostrar una llista de grups.	2	4
RF_usuari_llista_disp	L'aplicació ha tenir una opció per mostrar els dispositius d'un grup.	2	4

Annex 4 – Explicació dels endarreriments

Nom de la tasca	Hores previstes	Hores reals	Descripció
Desenv. del disp.	25h	100h	Primerament, el dispositiu no acabava de mostrar totes les funcionalitats que havia de tenir, després d'investigar vaig descobrir que no tenia la versió correcta del firmware instal·lada. Segonament, i després de fer moltes proves, vaig determinar que l'antena de LoRa que tinc relativament a prop de casa o no funcionava bé o no m'arribava realment la cobertura. Seguidament, vaig estar a la universitat fent proves per fer comunicar el dispositiu fent ús del protocol OTAA, però un altre cop vaig haver de desistir perquè sembla que la combinació de hardware, firmware i versió de LoRa no eren compatibles. Finalment, vaig aconseguir fer comunicar el dispositiu fent ús d'ABP i l'endemà no funcionava, després de fer moltes proves i de demanar ajuda a en Marc vam descobrir que el dispositiu estava perdent el comptador de missatges i deixava d'enviar en reiniciar-lo.
Desenv. Backend	20h	104h	Fer les proves d'integració i "parseig" de les dades que provenen de la xarxa LoRa o de la xarxa PyBytes ha requerit molta feina d'anar provant, i a causa de les diferents possibilitats de comunicació que té la solució vaig fer-ho exhaustivament per estar segur que tot funcionava com havia de fer-ho.