



**Escola Politècnica Superior  
d'Enginyeria de Vilanova i la Geltrú**

UNIVERSITAT POLITÈCNICA DE CATALUNYA

# TREBALL FINAL DE GRAU

**TÍTOL: Anàlisi, desenvolupament i control d'un sistema de comunicació  
entre dispositius IOT domèstics**

**AUTOR: CIFRE VICENS, ANTONI**

**DATA DE PRESENTACIÓ: Octubre, 2021**

**COGNOMS: CIFRE VICENS**

**NOM: ANTONI**

**TITULACIÓ: Grau en Enginyeria Informàtica**

**PLA:**

**DIRECTOR:**

**DEPARTAMENT:**

**QUALIFICACIÓ DEL TFG**

**TRIBUNAL**

**PRESIDENT**

**SECRETARI**

**VOCAL**

**DATA DE LECTURA:**

**Aquest Projecte té en compte aspectes mediambientals: ☐ Sí ☒ No**

## RESUM

En els últims anys s'ha normalitzat la inclusió de les tecnologies en l'àmbit familiar, normalitzant la implementació de sistemes domòtics dintre de casa. El desenvolupament dels sistemes per part de les grans corporacions suposa deixar vulnerable la privadesa de l'usuari. Per aquest motiu s'ha optat per a la creació d'un sistema domòtic complet de codi obert i segur. Aquests sistemes es compon en tres apartats diferents: els dispositius Internet of Things (IoT), la plataforma de control i la interfície de l'usuari. Per a la unificació dels tres, s'ha implementat un sistema de comunicació per a mantenir, analitzar i retransmetre les dades. La privadesa i seguretat de les dades es basa a assegurar que les comunicacions no són visibles per a un individu extern al sistema. La seguretat en les comunicacions s'ha dut a terme utilitzant els últims protocols de xifratge a partir d'un sistema de certificats i autenticacions. Pel que fa al control de les dades que es generen dintre del sistema, aquestes s'han emmagatzemat únicament en el servidor i s'ha restringit l'accés per evitar l'extracció no permesa de dades. Un cop creats els tres apartats del sistema domòtic i implementada la comunicació entre ells s'ha obtingut una fluïdesa molt bona en totes les comunicacions, i a més una càrrega màxima suficient per a mantenir una gran quantitat de dispositius IoT. Finalment, es pot afirmar que s'ha aconseguit dur a terme un sistema domòtic òptim, que manté el control dels dispositius IoT de forma autònoma, permetent a l'usuari interactuar amb aquests de forma segura i fluida.

### Paraules clau:

Casa intel·ligent	Internet de les coses	API REST	MQTT
Python	React	Raspberry Pi	Arduino

## ABSTRACT

In recent years, the inclusion of technologies in society has been standardised, normalising home automation systems. The development of these systems by large private corporations results in exposing and leaving vulnerable the user privacy. Therefore, this project describes the creation and implementation of a completely open-source and secure home automation system. Home automation systems consist of three elements: Internet of Things (IoT) devices, the control platform and the user interface. The communication system has been implemented to unify the three elements to maintain, analyse, and retransmit the data. Data privacy and security ensure that communications are not visible to an individual outside the system. Security in communications has been carried out using the latest encryption protocols supported by a system of certificates and authentications. The control of the data generated within the system has been stored only on the server, restricting access to prevent unauthorised data extraction. As a result of creating the three sections of the home automation system and the implementation of communication between them, excellent fluidity has been obtained in all communications. Additionally, the resulting system supports more components than the required to implement a smart home. Finally, an optimal home automation system has been achieved, which maintains control of IoT devices autonomously, allowing the user to interact with them securely and smoothly.

### Keywords:

Smart home	Internet of things	API REST	MQTT
Python	React	Raspberry Pi	Arduino

## SUMARI

<b>INTRODUCCIÓ</b>	<b>8</b>
<b>1. CONTEXT I FORMULACIO DEL PROBLEMA</b>	<b>9</b>
1.1 OBJECTIUS	10
<b>2. ESTAT DE L'ART</b>	<b>11</b>
2.1 TIPUS DE XARXES	11
2.1.1 XARXA EN ESTRELLA	11
2.1.2 XARXA EN MALLA	12
2.2 TECNOLOGIES SENSE FILS	12
2.2.1 RADIOFREQUÈNCIA	13
2.2.2 WI-FI	13
2.2.3 Z-WAVE	13
2.2.4 ZIGBEE	13
2.3 TIPUS DE DISPOSITIUS	14
2.4 PLATAFORMES DE CONTROL DOMÒTIC	14
<b>3. TECNOLOGIES A UTILITZAR</b>	<b>16</b>
3.1 TECNOLOGIA SENSE FILS	16
3.2 DISPOSITIUS	16
3.2.1 CONTROLADORS	16
3.2.2 MÒDUL	18
3.3 PROTOCOL DE DADES	20
3.3.1 DISPOSITIU – SERVIDOR	20
3.3.2 SERVIDOR – USUARI	22
3.4 PLATAFORMA	22
3.4.1 LENGUATGE DE PROGRAMACIÓ	23
3.4.2 BASE DE DADES	23
3.5 SERVIDOR	24
<b>4. DISSENY</b>	<b>26</b>
4.1 ARQUITECTURA	26
4.1.1 ARQUITECTURA DE LA XARXA	26
4.1.2 ARQUITECTURA DE COMUNICACIÓ	27
4.2 DISPOSITIUS IoT	28
4.2.1 CLASSIFICACIÓ	28
4.2.2 CARACTERÍSTIQUES DE CADA DISPOSITIU	29
4.2.3 DIAGRAMA DE FLUX	29
4.3 MAPA CONCEPTUAL	35
4.4 ENDPOINT	36
4.4.1 RUTES MQTT	36
4.4.2 API	38
<b>5. IMPLEMENTACIÓ</b>	<b>41</b>
5.1 SERVIDOR	41
5.1.1 SERVEIS	41
5.2 DISPOSITIUS	43
5.2.1 LENGUATGE, EINES I LLIBRERIES	43
5.2.2 PROGRAMACIÓ DELS MICROCONTROLADORS	44
5.2.3 ESQUEMA ELÈCTRIC	46
5.2.4 CONFIGURACIÓ INICIAL	48
5.2.5 DIAGRAMA DE COMUNICACIONS	48
5.3 BACKEND	50
5.3.1 LENGUATGE, EINES I LLIBRERIES	50
5.3.2 MÒDULS	52
5.3.3 ARQUITECTURA	59
5.4 FRONTEND	60

5.4.1	LLENGUATGE, EINES I LLIBRERIES	60
5.4.2	COMUNICACIÓ	62
5.4.3	REPRESENTACIÓ I TRACTAMENT DELS SENSORS	63
5.4.4	PANTALLES	67
5.5	SEGURETAT	74
5.5.1	COMPROVACIÓ	76
5.6	ESTUDI ECONÒMIC	78
<b>6.</b>	<b>TESTING</b>	<b>80</b>
6.1	API	80
6.2	MQTT	81
<b>7.</b>	<b>DEPLOYMENT DEL SISTEMA</b>	<b>84</b>
7.1	FRONTEND	84
7.2	BACKEND	84
	<b>CONCLUSIONS</b>	<b>85</b>
	<b>BIBLIOGRAFÍA</b>	<b>86</b>

## SUMARI DE FIGURES

FIGURA 1. EXEMPLE DE XARXA EN ESTRELLA	12
FIGURA 2. EXEMPLE DE XARXA EN MALLA	12
FIGURA 3. CONTROLADOR ESP-01	17
FIGURA 4. CONTROLADOR ESP-12F	17
FIGURA 5. CONTROLADOR ESP32-S	18
FIGURA 6. MÒDULS LED, TEMPERATURA, HUMITAT I RELÉ PER ESP-01	18
FIGURA 7. MÒDUL PIR, D'HUMITAT, VOLTÍMETRE I DE PROXIMITAT PER A CONTROLADORS GENERALS	19
FIGURA 8. MÒDUL ESP32-CAM AMB EL XIP ESP32	19
FIGURA 9. SISTEMA SIMPLE DE COMUNICACIÓ AMB MQTT	21
FIGURA 10. RASPBERRY PI 4	24
FIGURA 11. ARQUITECTURA DE LA XARXA.	27
FIGURA 12. ARQUITECTURA DE COMUNICACIÓ.	28
FIGURA 13. DIAGRAMA DE FLUX DEL DISPOSITIU GENERAL	30
FIGURA 14. DIAGRAMA DE FLUX DEL DISPOSITIU ACTUADOR	31
FIGURA 15. DIAGRAMA DE FLUX DEL DISPOSITIU ACTIU	32
FIGURA 16. DIAGRAMA DE FLUX DEL DISPOSITIU PASSIU	33
FIGURA 17. DIAGRAMA DE FLUX DEL DISPOSITIU CÀMERA	34
FIGURA 18. MAPA CONCEPTUAL	35
FIGURA 19. APACHE FORWARD PROXY	41
FIGURA 20. MÒDUL ADAPTADOR A SÈRIE PER ARDUINO	45
FIGURA 21. DIAGRAMA DE CONNEXIÓ ENTRE ESP-01 I L'ADAPTADOR SÈRIE	45
FIGURA 22. DIAGRAMA DE CONNEXIÓ ENTRE UN ESP32 I L'ADAPTADOR SÈRIE	46
FIGURA 23. ESQUEMA ELÈCTRIC D'UN RELÉ	46
FIGURA 24. ESQUEMA ELÈCTRIC D'UN DISPOSITIU PIR	47
FIGURA 25. ESQUEMA ELÈCTRIC PER A MONITORAR UNA XARXA ELÈCTRICA.	47
FIGURA 26. PUNT D'ACCÉS PER A LA CONFIGURACIÓ INICIAL D'UN DISPOSITIU IOT.	48
FIGURA 27. DIAGRAMA DE COMUNICACIÓ DE L'INICI DE SESSIÓ DEL DISPOSITIU	49
FIGURA 28. DIAGRAMA DE COMUNICACIÓ DE L'ACTUALITZACIÓ DE L'ESTAT DEL DISPOSITIU	49
FIGURA 29. DIAGRAMA DE COMUNICACIÓ DEL CANVI D'ESTAT DE L'ACTUADOR	50
FIGURA 30. DIAGRAMA DE COMUNICACIÓ DELS PARÀMETRES DE CONFIGURACIÓ DEL DISPOSITIU	50
FIGURA 31. ARQUITECTURA DEL BACKEND	60
FIGURA 32. ESTRUCTURA DE LA COMUNICACIÓ AXIOS DINTRE DEL FRONTEND.	62
FIGURA 33. ESTRUCTURA DEL WEBSOCKET DINTRE DEL FRONTEND	63
FIGURA 34. TARGETA VISUAL DEL SENSOR ACTUADOR.	64
FIGURA 35. FORMULARIS EMERGENT DEL SENSOR ACTUADOR.	64
FIGURA 36. TARGETA VISUAL DEL SENSOR ACTIU.	65
FIGURA 37. TARGETA VISUAL DEL SENSOR PASSIU.	66
FIGURA 38. TARGETA VISUAL DEL SENSOR PASSIU.	66
FIGURA 39. PANTALLA LOGIN.	67
FIGURA 40. PANTALLA PRINCIPAL.	68
FIGURA 41. PANTALLA PER AFEGIR DISPOSITIUS IOT.	69
FIGURA 42. MENÚ DE NAVEGACIÓ ENTRE PANTALLES.	69
FIGURA 43. PANTALLA PER CONTROLAR LES VISIBILITATS.	70
FIGURA 44. PANTALLA DE CONFIGURACIÓ DELS DISPARADORS.	71
FIGURA 45. MODAL DE CREACIÓ D'UN DISPARADOR.	72
FIGURA 46. PANTALLA DE GESTIÓ DELS SENSOR.	73
FIGURA 47. PANTALLA DE GESTIÓ DELS USUARIS.	73
FIGURA 48. PANTALLA PER GESTIONAR LES DEFINICIONS.	74
FIGURA 49. ESCANEIG DE SEGURETAT AMB NESSUS.	77
FIGURA 50. CAPTURES DELS PAQUETS MQTT AMB WIRESHARK.	77
FIGURA 51. COL·LECCIÓ DE POSTMAN PER ANALITZAR LA API.	80
FIGURA 52. MITJANA ARITMÈTICA DEL TEMPS DE RESPOSTA EN EL TEST DE LA API.	81
FIGURA 53. VISUALITZACIÓ DEL CONSUM DELS RECURSOS AMB HTTP.	82
FIGURA 54. GRÀFICA DELS RECURSOS UTILITZANT EN FUNCIÓ DELS FOTOGRAFES PER SEGON SENSE REENVIAMENT.	83

FIGURA 55. GRÀFICA DELS RECURSOS UTILITZANT EN FUNCIO DELS FOTOGAMES PER SEGON AMB REENVIAMENT.



## **GLOSSARI DE SIGNES, SÍMBOLS, ABREVIATURES, ACRÒNIMS I TERMES**

**AES** - Advanced Encryption Standard  
**API** - Application Programming Interface  
**CRUD** - Create, Read, Update and Delete.  
**CSS** - Cascading Style Sheets  
**DDNS** - Dynamic DNS  
**DNS** - Domain Name System  
**DOM** - Document Object Model  
**DTLS** - Datagram Transport Layer Security  
**FPS** - Fotogrames per segon.  
**GPIO** - General-purpose Input/Output  
**HTML** - HyperText Markup Language  
**HTTPS**- Hypertext Transfer Protocol  
**IDE** - Integrated Development Environmen  
**IEEE** - Institute of Electrical and Electronics Engineers  
**IO** - Input/Output  
**IoT** - Internet of the Things.  
**IP** - Internet Protocol  
**JSON** - JavaScript Object Notation  
**JWT** - JSON Web Token  
**MQTT** - Message Queuing Telemetry Transport  
**OS** - Operating System  
**PIR** - Passive Infrared Sensor  
**QoS** - Quality of service  
**REST** - Representational State Transfer  
**RFC** - Request for Comments  
**SPA** - Single-Page Application  
**SQL** - Structured Query Language  
**SSH** - Secure Shell  
**SSL** - Secure Sockets Layer  
**SVG** - Scalable Vector Graphics  
**TCP** - Transmission Control Protocol  
**TLS** - Transport Layer Security  
**UML** - Unified Modeling Language  
**URL** - Uniform Resource Locator  
**VCC** - Voltage Common Collector  
**WSGI** - Web Server Gateway Interface  
**WSS** - WebSocket Security  
**XSRF** - Cross-site Request Forgery

## INTRODUCCIÓ

En els últims anys han aparegut una gran quantitat d'empreses dedicades a la creació de sistemes i aparells domòtics. Algunes d'elles se centren exclusivament a crear els dispositius amb Internet of Things (IoT), mentre que d'altres se centren a elaborar les aplicacions que unifiquen aquests dispositius.

Avui en dia els dispositius domòtics presenten principalment tres problemes. El primer és la seva vinculació a un conjunt de termes i condicions, per tal de poder ser utilitzats. Addicionalment els usuaris d'aquestes tecnologies freqüentment no disposen de coneixements reals sobre quin és seu funcionament real. Per tant, treballant amb dispositius vinculats a les grans empreses del sector tecnològic, deixen l'usuari exposat al desconeixement de com les empreses fan ús de les dades privades i personal, com són les que recullen els dispositius domòtics a l'interior de les nostres cases. El tercer obstacle va lligat a la dificultat de poder garantir que aquests sistemes són segurs. En altres paraules, va lligat a la capacitat que tenen els sistemes a defensar-se enfront atacs externs. En aquest context el terme seguretat no va enfocat exclusivament a evitar la difusió de les dades personals, sinó que també inclou la protecció de l'usuari enfront a l'obtenció de control sobre els dispositius que formen part de la casa domòtica per part d'un tercer no desitjat. En aquesta situació l'intrús seria capaç de controlar i monitorar els dispositius de la casa.

Tenint això en compte, aquest projecte descriu la creació d'un sistema domòtic tancat propi. El disseny del sistema, des dels dispositius IoT fins a la interfície de l'usuari, així com el seu allotjament a un servidor local garanteix l'obtenció d'un producte transparent, segur i lleuger.

S'ha de considerar que la reproducció i implementació d'aquest sistema no està a l'abast de qualsevol persona. Això és a causa del fet que per a implementar el sistema s'han de tenir uns coneixements mínims d'electrònica per a poder muntar els dispositius IoT i d'informàtica per a configurar i aixecar l'aplicació.

En aquest document s'exposen les possibles tecnologies que es poden utilitzar per a crear cada una de les parts d'una casa domòtica. Addicionalment es presenta el perquè de l'elecció de cada una de les tecnologies i protocols de comunicació escollits així com el seu funcionament. Finalment es descriu la implementació de tots els elements en un sistema local.

# 1. CONTEXT I FORMULACIO DEL PROBLEMA

Per poder entendre el problema que es vol resoldre amb aquest projecte, primer és necessari entendre els sistemes domòtics. És per això que en els següents apartats es presentarà el funcionament i les característiques que componen un sistema domòtic, aplicat específicament a una casa domòtica.

El sistema complet d'una casa domòtica es pot dividir clarament en tres apartats diferents:

- **Dispositius IoT:** recullen dades i interaccionen amb l'ambient. Alguns exemples serien una càmera o un interruptor.
- **Interfície de l'usuari:** representa i manipula els estats dels dispositius IoT.
- **L'aplicació central:** unifica la comunicació entre els dispositius IoT i la interfície. A més, llança accions programades o per afectar d'altres accions i controla i persisteix els estats dels dispositius.

Aquests tres apartats són necessaris per a la creació d'un sistema domòtic. Analitzant individualment cada apartat s'aprofundirà en les opcions que es disposen per a poder resoldre cada un d'aquests. A més a més, permetrà estudiar les característiques que aporta cada possible solució, per més endavant poder escollir quina ens és més viable per muntar una casa domòtica amb les característiques escollides.

L'elecció d'únicament utilitzar tecnologies lliures i pròpiament creades limita els dispositius IoT a la utilització de sensors que es controlin amb single-board computer (SBC). Aquesta limitació ve donada per tal de generar un sistema tancat on s'utilitza un mateix protocol de comunicacions, d'autenticació i encriptació maximitzant així la seguretat i simplicitat del sistema.

L'elecció d'utilitzar SBC no limita el nombre de dispositius IoT del sistema gràcies al fet que existeixen un gran conjunt de sensors i controladors creats per a funcionar amb les plaques SCB. La part negativa és que és necessària una petita inversió de temps addicional pel muntatge, però de forma general el cost econòmic de cada dispositiu IoT és menor amb relació a les altres opcions.

Pel que fa a la interfície de l'usuari, es disposa d'un seguit d'opcions entre les quals escollir. La creació d'una interfície web aporta un accés general per a tots els dispositius, des d'ordinadors fins a telèfons mòbils, amb una sola aplicació. El problema d'aquesta elecció és que genera limitacions en la velocitat de càlcul i en l'emmagatzematge de dades. L'elecció de crear una aplicació mòbil o per ordinador elimina les limitacions amb la velocitat, però únicament es pot utilitzar per a un tipus de dispositiu.

La selecció d'un tipus d'interfície no és essencial perquè no repercuteix en el funcionament del sistema. La part crítica es troba sobre l'aplicació central, ja que si aquesta s'ha generat de forma correcta, la interfície es pot modificar o implementar-ne una de nova sense la necessitat de canviar res del sistema.

L'aplicació central o plataforma de control compta un gran nombre d'opcions pel fet que ha de controlar i mantenir tot el sistema. Una elecció important que recau sobre aquesta secció és la selecció del protocol de comunicació entre els dispositius IoT i la interfície. L'elecció del protocol de comunicacions determina les limitacions i característica per a cada un dels apartats fent que una mala elecció del protocol generi un mal funcionament del sistema.

En l'elecció de les tecnologies de l'aplicació central s'ha d'anar amb molta cura, ja que la seva modificació pot repercutir de forma brusca sobre els altres apartats.

## 1.1 OBJECTIUS

L'objectiu general és aconseguir implementar un sistema domòtic d'una casa amb la utilització de tecnologies obertes i amb la prioritització de la seguretat i el control de les dades.

Per arribar a complir aquest objectiu és necessari dissenyar i implementar cada un dels apartats del sistema. Això implica la creació dels dispositius IoT, una interfície d'usuari i una plataforma per controlar les dues anteriors. A més de tot aquest conjunt s'ha d'utilitzar el mètode més segur i eficient per encriptar i autenticar les dades que es transmeten dintre del sistema.

Els dispositius IoT han d'utilitzar tecnologies obertes on el firmware pugui ser creat i introduir dintre del dispositiu per un control específicament dissenyat pel sistema creat. Els dispositius han de mantenir un control sobre l'autoritat de la qual poden obtenir les ordres per evitar l'escolta d'una autoritat impostora de fora del sistema.

La interfície de l'usuari ha de poder interactuar amb els dispositius IoT i el sistema en funció de l'autoritat de l'usuari identificat. La interpretació dels valors que recopilen els dispositius ha de ser senzilla i amigable per a permetre la utilització i interacció amb el sistema sense un previ coneixement d'ell.

La plataforma de control obté la responsabilitat més gran en encarregar-se de controlar i mantenir la informació de tot el sistema. Ha de gestionar les peticions dels usuaris procedents de la interfície en funció de la seva autoritat i ha de controlar els dispositius IoT en funció de les dades que rep.

Aquesta ha de mantenir un sistema d'encriptació i autenticació per restringir l'accés als paquets que no utilitzin els certificats seleccionats i evitar que un tercer extragui informació dels paquets que envia el sistema.

Un dels objectius en implementar una casa domòtica és l'obtenció de funcionalitats addicionals que automatitzen aspectes en la interacció amb la casa. Per això és necessari la creació de funcions que automatitzin accions sobre els dispositius IoT en funció de l'estat de la casa.

Un cop el sistema es troba complet es requereix una comunicació fluida entre tots els apartats que el formen. La seguretat i la privadesa és la prioritat del sistema, però si la interacció amb aquest no es realitza de forma fluida, no es pot considerar un sistema utilitzable.

## 2. ESTAT DE L'ART

Des dels inicis del segle XX ha existit una visió cap a l'existència de les cases intel·ligents. Començant amb la implementació de robots que ens netegen el menjador fins a l'automatització completa per tal de fer la vida més fàcil en ser controlada a través de dispositius automàtics amb la mínima interacció humana.

En l'any 1965, Gordon Moore va publicar el que es coneix com a la llei de Moore, aquesta llei és coneguda per preveure que la densitat dels circuits integrats en un sol xip és duplicada cada any durant la pròxima dècada. Però aquesta millora fins ara només afectava la millora d'ordinadors i en el cas dels aparells domèstics no tenia cap implicació fins aquests últims anys, on ja es disposa d'una potència adequada de càlcul en dispositius petits amb un cost molt baix. Aquesta millora permet introduir dispositius autònoms als aparells de la llar fent que en els últims anys, un aparell com pot ser una rentadora, que la seva millora ha estat mínima, ara es pot automatitzar i integrar en un sistema domòtic connectant-la amb altres dispositius amb un conjunt de funcionalitats més elevat.

Encara que la totalitat d'aquesta visió no s'hagi complert, ja existeixen una gran varietat de dispositius autònoms i intel·ligents que comencen a encaminar aquesta visió cap a la realitat.

D'aquesta manera, en aquest apartat revisarem les tecnologies i dispositius autònoms, principalment enfocats a la llar, que pretenen automatitzar i controlar alguna de les taques que es realitzen dintre de la llar.

### 2.1 TIPUS DE XARXES

Un sistema intel·ligent es basa en la comunicació entre els diferents elements que el formen, per això és important el tipus de la xarxa de comunicacions que es crea entre els diferents dispositius. Aquesta distribució de la xarxa tindrà un efecte dintre de l'ecosistema de la llar, millorant o empitjorant diverses funcionalitats d'aquest.

Per a comprendre aquestes diferències s'ha analitzat el conjunt d'estructures més utilitzades en l'actualitat.

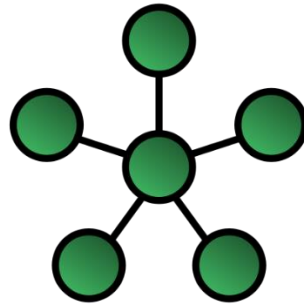
#### 2.1.1 XARXA EN ESTRELLA

El tipus de Xarxa en estrella és una topologia de xarxa típicament utilitzada en àrees locals on els nodes estan connectats a un node central el qual actua d'encaminador per rebre i transmetre els missatges entre nodes.

Aquest tipus de xarxa té l'avantatge de què si algun node perifèric falla no té repercussió sobre el funcionament de la resta de la xarxa fent un manteniment més senzill. El problema és que el sistema depèn d'un sol node a més el rendiment disminueix en afegir més nodes.

La xarxa en anell és útil en els sistemes domòtics locals, ja que la xarxa local d'una casa sol estar formada seguint aquesta distribució, formada per un encaminador central i un conjunt de dispositius connectats a ell. D'altra banda, en un sistema local no s'introdueixen una quantitat descomunal de dispositius, de tal forma que no tindrà

una gran repercussió sobre el rendiment.

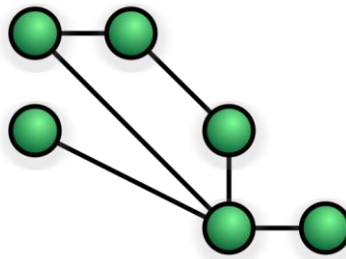


**Figura 1.** Exemple de xarxa en estrella

### 2.1.2 XARXA EN MALLA

La xarxa en malla és una topologia de xarxa formada per nodes distribuïts en estructura de malla on cada node està interconnectat amb un o més nodes. Tot i que aquesta estructura pot estar formada per cables, el seu potencial apareix quan els nodes es comuniquen sense fils, on es transforma en una xarxa d'encaminadors cercant en cada moment la manera més adient d'enviar un missatge entre cada node.

Aquest tipus de xarxa és el més prometedor per a la llar intel·ligent i s'espera que jugui un gran paper en el IoT gràcies a la forma en què es comuniquen entre els nodes i la connectivitat que en genera dintre d'una llar. El problema actualment és que existeixen pocs protocols que explotin aquest sistema i una quantitat molt reduïda i poc econòmica de dispositius que acceptin aquests protocols.



**Figura 2.** Exemple de xarxa en malla

## 2.2 TECNOLOGIES SENSE FILS

Gràcies a l'aparició de les tecnologies sense fil, ha estat possible el desenvolupament de la llar intel·ligent pel baix cos de la instal·lació i els dispositius permetent arribar a una major part de la població.

En els últims anys han aparegut una gran quantitat de tecnologies sense fil, algunes d'elles amb una gran aplicació en els sistemes domòtic mentre d'altres desapareixien de l'àmbit IoT. Per a poder comprendre diferents opcions, a continuació s'observa una visió general de les tecnologies sense fil més utilitzades en una casa intel·ligent i les seves característiques.

## 2.2.1 RADIOFREQUÈNCIA

Aquest és una de les primes tecnologies sense fil utilitzades per a transmetre informació entre dispositius intel·ligents tot i que en l'actualitat la seva utilització ha disminuït bruscament.

La versió estandarditzada és amb 433 MHz la qual permet la comunicació a una distància de 20 a 200 metres depenent del voltatge aplicat. El principal problema és la baixa amplada de banda que permet, deixant aquesta tecnologia en sistemes molt simples que generen un baix flux de dades.

## 2.2.2 WI-FI

El Wi-Fi és una de les tecnologies més comunes de comunicació en àrees domèstiques i amb una velocitat de transmissió de dades elevada, cosa que la fa una de les tecnologies clau en la implementació d'una casa intel·ligent.

Actualment és la tecnologia més comuna en els dispositius IoT que formes el sistema domòtic local i amb una gran quantitat de protocols que funcionen a sobre d'ella. A més, arran de la gran acollida d'aquesta tecnologia per part dels dispositius IoT, la Wi-Fi Alliance ha creat la versió Wi-Fi 6 pensant amb l'adopció dels dispositius IoT. Aquesta nova versió permet el control d'una major quantitat de dispositius simultàniament i un menor consum d'energia per als dispositius que funcionen amb bateries.

El Wi-Fi funciona sota la norma IEEE 802.11 i el protocol d'Internet IPv6.

## 2.2.3 Z-WAVE

Z-Wave forma una topologia de xarxa de malla amb un baix consum energètic i està bàsicament implementada per a sistemes domòtics gràcies a la seva fiabilitat en sistemes locals. Aquesta fiabilitat és gràcies a l'estructura d'ordres senzilles i una baixa interferència d'altres tecnologies. A més proporciona una baixa latència amb la transmissió de petits paquets de fins a 100 kbps i una distància de senyal de 30 metres i 100 m metres en interior o en exterior respectivament.

El problema és la baixa amplada de banda, ja que limita la utilització de dispositius IoT com les càmeres. Per altra banda, aquest protocol no és obert, el que genera que únicament existeix un únic proveïdor del maquinari i una competència molt limitada, per tant, el preu amb relació a altres és més elevat i existeix una menor disponibilitat.

## 2.2.4 ZIGBEE

És un protocol via radiofreqüència típicament utilitzat en sistemes domòtic funcionant segons l'estàndard IEEE 802.15.4. Igual que Z-Wave funciona com una xarxa de malla i és de baixa potència energètica, però amb l'avantatge que és un protocol obert i amb una major amplada de banda arribat als 250 kbps.

Els dispositius ZigBee de forma predeterminada disposen d'una encriptació AES 128 utilitzant en la capa d'aplicació un mòdul d'encriptació per a cada una de les

connexions entre els nodes.

## 2.3 TIPUS DE DISPOSITIUS

Els dispositius IoT són la part fundamental per a la creació d'una casa intel·ligent, encarregats de recopilar informació de l'ambient i poder actuar sobre el mateix.

Una de les principals funcions en l'ecosistema és mantenir la seguretat de la llar amb la vigilància i control de l'estat en tot moment. En aquest àmbit es troben un conjunt de dispositius enfocats en el seu monitoratge, com ara càmeres de seguretat, sensors d'obertura de portes i finestres o sensor de gasos perjudicials en l'ambient per tal de poder detectar qualsevol problema i actuar en funció de cada necessitat.

En primer lloc es troben els sensors visuals, com són les càmeres, a partir d'aquests sensors es pot detectar moviment, persones o objectes. A més, es poden guardar les imatges per a la seva posterior revisió.

També es poden trobar sensor ambiental com per exemple els sensors binaris, com els detectors de presència o absència d'un objecte o moviment. Dintre d'aquesta secció es troben els sensors de moviment, vibració, contacte o de pressió per tal de detectar quan una porta és oberta o es troba algú en una habitació.

Els sensors ambientals també inclouen els sensors meteorològics com és la temperatura, humitat o pressió o sensors que analitzen l'índex de contaminació en l'aire, la seva qualitat, nivell de pols o pol·len i fins i tot sensors de gas butà i altres gasos.

Per altra banda hi ha els sensors de monitoratge i control del consum elèctric de la casa, com podria ser un monitor elèctric dedicat a mesurar el voltatge, watts i amperatge que consumeix una casa o una part d'ella. Una altra part important són els relés usats per activar o desactivar una línia de la casa, com podria ser una llum, un calefactor o un endoll.

A més del conjunt de sensors, una part important és la personalització del comportament en funció de cada usuari i el seu ús típic sobre aquests, podent unificar els dispositius perquè actuïn en funció de l'estat dels altres sensors, com per exemple l'activació del relé del termòstat quan la temperatura es baixa i l'usuari estat tornant a casa.

Això i tot s'ha comprovat que per a l'automatització total d'una casa encara falta un llarg camí per a recórrer, ja que el nombre de variables que s'ha de tenir en compte per automatitzar completament les accions que una persona té amb la seva casa no són poques i sobretot molt canviants.

## 2.4 PLATAFORMES DE CONTROL DOMÒTIC

En disposar d'un gran nombre de dispositius intel·ligent, es necessita una plataforma que els englobi a tots i els orquestri per al seu funcionament en grup, d'aquesta



manera, han sorgit un conjunt de plataformes encarregades de la seva unificació i control.

Com és normal, les dues majors plataformes actuals pel control dels sistemes domòtics són Google Home i Alexa. A l'hora de parla d'usuaris totals, aquestes dues companyies tenen el domini del mercat, però actualment no es pot dir que disposin d'un sistema domòtic real, perquè el nombre de dispositius IoT amb els quals són compatible és molt baix i només permeten controlar aspectes reduïts de la llar. Per altra banda el seu nivell de comunicació entre dispositius és reduït.

Per altra banda, s'ha trobat Home Assistant, una plataforma de control de codi obert creada amb Python, la qual té una de les majors integracions del mercat, podent controlar a la majoria dels dispositius IoT gràcies a totes les seves integracions amb els diferents protocols de comunicació, des de ZigBee fins a la integració amb Alexa i Google. També permet la connexió amb dispositius IoT a través del protocol MQTT.

Aquesta plataforma de control s'ha d'allotjar en un servidor o en una Raspberry Pi local el que limita la seva utilització a usuaris amb un interès cap al camp de la domòtica i amb coneixements informàtics.

### **3. TECNOLOGIES A UTILITZAR**

En aquest apartat s'han plasmat les tecnologies disponibles, les seves característiques i l'elecció de cada una per a resoldre les necessitats a l'hora de la creació d'aquest projecte. S'ha de tenir en compte que en alguns casos no existeix una solució única, ja que aquesta no varia el resultat final o perquè cada objectiu es prioritza de forma diferent fent l'elecció d'aquesta purament personal.

#### **3.1 TECNOLOGIA SENSE FILS**

En primer lloc, i abans de l'elecció dels dispositius, s'ha d'establir la tecnologia sense fils que s'utilitzarà en el sistema domòtic, pel fet que depèn de l'elecció es disposarà d'un nombre diferent de dispositius.

Una de les tecnologies sense fil més utilitzades és la tecnologia Wi-Fi. Aquest protocol és el més generalitzat i amb una gran flexibilitat, el que el fa el candidat perfecte per a la implementació d'una casa intel·ligent. Tot i que ZigBee té un gran avantatge sobre el Wi-Fi a l'utilitzar una xarxa en malla, però dintre del panorama actual on el Wi-Fi sol arribar a tots els racons d'una casa, l'avantatge no és massa elevada.

Per altra banda, la tecnologia Wi-Fi pot abastar una gran velocitat permetent incloure dintre d'aquesta dispositius IoT que necessiten un gran consum d'amplada de banda com podrien ser les càmeres de seguretat, mentre que ZigBee o Z-Wave no ho podria suportar. A més, totes les cases actuals ja disposen d'una xarxa Wi-Fi el que suposa una major facilitat a l'hora de la implementació.

Finalment, com es veurà a continuació, la majoria dels controladors barats utilitzen Wi-Fi, menters que els que utilitzen altres tecnologies són més escassos i cars a causa de la baixa demanda pel públic general.

Per aquests motius s'ha decidit utilitzar la tecnologia sense fils Wi-Fi, ja que aquesta aporten una major flexibilitat sobre els dispositius IoT i les tecnologies de programació compatibles.

#### **3.2 DISPOSITIUS**

Els dispositius IoT que encaixen en un sistema domòtic com el que s'ha exposat estan formant per dues parts, el dispositiu en si, com seria un relé o el sensor de temperatura i per altra banda el microcontrolador, el qual es programa el firmware per tal de complir les funcionalitats de cada dispositiu i del sistema en el seu conjunt.

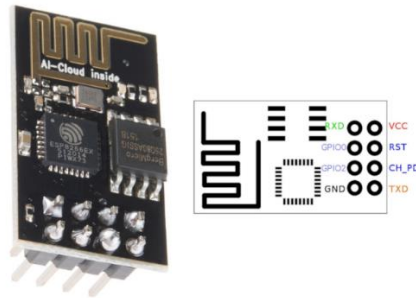
A continuació es veurà un petit conjunt dels controladors i sensors disponibles per a cobrir les funcions bàsiques d'una casa domòtica.

##### **3.2.1 CONTROLADORS**

Un dels microxips més utilitzats és el ESP8266, aquest és un microxip Wi-Fi de baix cost, amb una pila TCP/IP completa i molt econòmic per a utilitzar amb dispositius IoT. A partir del microxip ESP8266 es creen un conjunt de microcontrolador molt barat i que ofereixen una bona potència de càlcul per a cobrir la majoria de necessitats que es

tenen en una casa intel·ligent.

Dintre dels controladors que utilitzen el ESP8266 un dels més destacats per la seva mida i la versatilitat és el ESP-01.



**Figura 3. Controlador ESP-01**

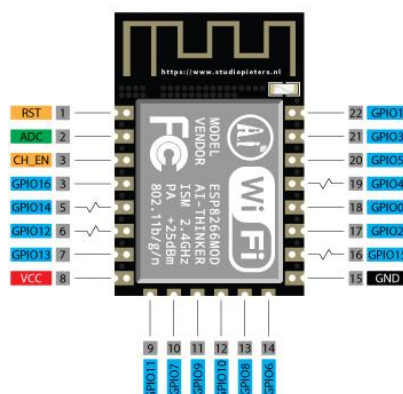
Aquest controlador és una de les millors eleccions per a controlar un sol dispositiu IoT gràcies al seu preu, el qual varia entre 0,75 € i 3 €, i el nombre de llibreries disponibles per a la lectura i control dels sensors. L'única limitació és el baix nombre de pins dels quals disposa, el que fa és que únicament es pugui controlar un o dos sensors a l'hora.

Les seves característiques tècniques principals són:

- Funciona amb una tensió d'alimentació de 3.3 V fins a 3.7 V.
- Suporta el protocol 802.11 b/g/n a 2,4 GHz.
- Potència de sortida de 0,15 W.
- Processador integrat de 32 bits.
- Mida de 24 mm x 14 mm.
- Memòria flash de 512kB.

Per a millora aquestes característiques existeix la versió ESP-01s, la qual augmenta la memòria flash a 1MB i elimina el LED que indica si està encès, disminuint el consum d'energia.

També, del costat del ESP8266, es troba el controlador ESP-12F, un mòdul un poc més car que l'anterior, arribant a costar fins a 5€, però amb un nombre major de pins, fent-lo ideal per a controlar un conjunt de sensors units gràcies als 16 pins. En relació amb el ESP-01 augmenta la seva memòria flash fins a 4MB, però la resta de característiques tècniques son igual.



**Figura 4. Controlador ESP-12F**

A partir d'aquests dos controladors es pot manejar a la majoria de sensors del mercat menys les càmeres, per a poder implementar-les s'ha utilitzat el controlador ESP32, ja que té una major potencia que els anteriors i una major connectivitat permetent suportar una gravació, encriptació i comunicació de les imatges en temps real. També permet la connexió d'una antena externa per augmentar el senyal Wi-Fi.

El ESP32-S té un sol processador funcionant a 240MHz, una memòria SRAM de 320 kiB i una rom de 128 kiB, a més de tenir uns 43 pins programables. Aquestes característiques la fan la placa més potent, el problema és que el seu preu ronda entre els 5 € i 10 € i el consum elèctric és més elevat, l'limitat la seva utilització únicament per als mòduls i sensors que ho requereixen.



**Figura 5.** Controlador ESP32-S

A partir d'aquests microcontroladors s'ha implementat el sistema domòtic, des dels dispositius més simples fins als dispositius més complexos gràcies a la varietat de la potència i nombre de pins.

### 3.2.2 MÒDUL

Els mòduls són el conjunt de sensors i actuadors connectats al controlador per tal de creat un dispositiu IoT que interactuï amb la casa domòtica. La selecció dels mòduls depenen de les necessitats que s'ha de cobrir per això en aquesta secció es plasma un conjunt de dispositius interessants que s'han utilitzat.

En primer lloc, en utilitzar el controlador ESP-01, es troben disponibles molts mòduls ja preparats per a una instal·lació senzilla al dur integrat en la placa un connector per al ESP-01 eliminant la necessitat de soldar els pins necessaris i unificant els connectors d'alimentació dels dos components.

S'han trobat controladors per a tires LED RGB, sensors de temperatura, d'humitat, de pressió i relés entre d'altres.



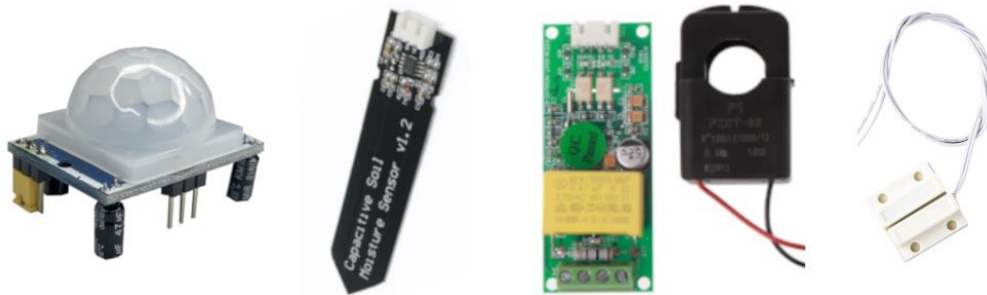
**Figura 6.** Mòduls LED, temperatura, humitat i relé per ESP-01

Com es veu en la figura 6, tots els mòduls tenen el connector de 8 pins per a inserta el

ESP-01 i així disminuir el cost d'instal·lació. El ESP-01 dintre del mercat IoT té una de les millors acollides, per això existeixen un gran nombre de mòduls disponibles a un baix cost econòmic preparats per funcionar amb aquest controlador.

Per altra banda, es troben els mòduls generals, els quals es poden controlar amb qualsevol dels controladors anteriors, sempre que aquest tingui el nombre de pins necessaris per a la comunicació de les dades. L'únic desavantatge respecte als anteriors mòduls és que la unió entre el controlador i el mòdul és més costosa, ja que s'han de connectar els pins de forma manual.

Per exemple dintre d'aquest conjunt s'han seleccionat sensors de moviment PIR, un d'humitat de sòl capacitiu, un sensor per medi el consum d'electricitat (pzem-004t) i un sensor magnètic per a detectar l'obertura de portes o finestres.



**Figura 7.** Mòdul PIR, d'humitat, voltímetre i de proximitat per a controladors generals

Cada un d'aquests mòduls estan dissenyats per a poder executar una tasca específica, d'aquesta manera, es pot utilitzar un sol controlador per a cada un dels mòduls o unificar un conjunt de mòduls i connectar-se a un únic controlador que disposi de suficient pins per a tots.

Per últim utilitzarem el mòdul ESP32-CAM, un mòdul de càmera molt petit amb el xip ESP32-S que costa entre 5 € i 10 €. A més de la càmera OV2640 i diversos pins per connectar perifèrics, també inclou una ranura per a targetes microSD que és utilitzada per emmagatzemar imatges i vídeos capturats per la càmera.



**Figura 8.** Mòdul ESP32-CAM amb el xip ESP32

Gràcies al fet que el dispositiu està format per un mòdul de càmera mes el controlador ESP32-S ens permet implementar un gran nombre de càmeres en el nostre sistema sense una inversió econòmica molt elevada, a més es pot configurar amb un gran nombre de funcions com el reconeixement de persones i controlar amb el mateix altres mòduls a l'hora.

### 3.3 PROTOCOL DE DADES

Una de les parts més importants és l'elecció del protocol de dades que s'ha utilitzat entre els dispositius, el servidor i el client, ja que depèn d'aquesta elecció les característiques i limitacions del sistema. Cada protocol proporciona un tipus d'encriptació, una velocitat de transmissió màxima o una mida dels paquets determinada.

Els protocols de dades en l'àmbit IoT s'utilitza per connectar, monitorar i controlar els dispositius IoT a través d'una aplicació i proporcionar una comunicació entre aquests i l'usuari.

En primer lloc s'han comparat els dos protocols més utilitzats a l'hora de comunicar els dispositius IoT amb la plataforma encarregada d'orquestrar els seus estats i a continuació els protocols més usats per a crear la comunicació entre la plataforma de control i el client per tal de veure els estats dels dispositius en temps real i poder actuar sobre ells.

#### 3.3.1 DISPOSITIU – SERVIDOR

El protocol actualment més utilitzat per a la retransmissió de dades per part dels dispositius IoT és **MQTT**, un protocol de dades molt lleuger i amb una composició bàsica transformant-lo en un protocol ideal per a ser usat amb dispositius petits i amb una baixa potència de càlcul.

El seu model de missatgeria es basa en editor-subscriptor (Pub/Sub), un flux de dades simple i una major fiabilitat. MQTT implementa un sistema per a controlar la qualitat del servei (QoS) creant 3 nivells possibles:

- QoS 0: Com a màxim s'entrega un cop – Adequat si es permet la pèrdua de dades. Requereix una menor quantitat de tràfic dintre de la xarxa.
- QoS 1: Com a mínim s'entrega un cop – S'assegura que el missatge és rebut pels consumidors del tema. S'utilitza únicament si és necessari garantir que el missatge és entregat permetent duplicats.
- QoS 2: S'entrega exactament un cop – Ideal per escenaris crítics on és essencial que el missatge s'entregui un sol cop. Genera una major quantitat de tràfic dintre de la xarxa i un major consum de recursos.

MQTT ha permès el creixement dels dispositius IoT petits, barats i de baixa potència, ja que aquest protocol va ser dissenyat específicament per a dispositius simples. Al principi, els dispositius IoT usaven el protocol HTTP per a comunicar-se. Aquest protocol té una capçalera de 8000 bytes per a cada un dels paquets que envia, mentre que MQTT únicament necessita 2 bytes.

MQTT va ser creat per IBM per ser utilitzat en condicions difícils, una de les característiques que necessitaven era que necessités un baix consum energètic pel seu funcionament. Si es compara amb el protocol HTTP, la diferència és sorprenent, en el cas de ser utilitzat amb una xarxa de 3G consumeix 170 vegades menys i 47

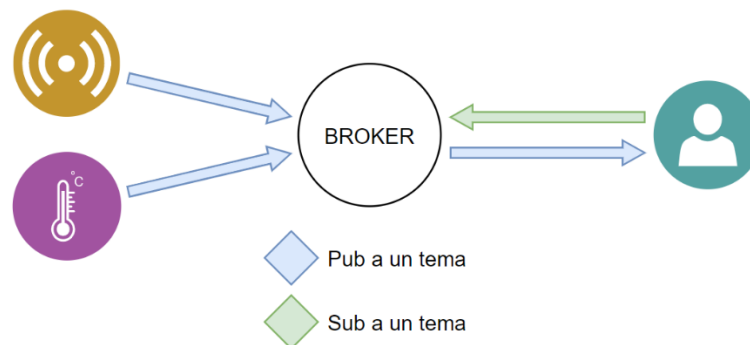
vegades menys amb una Wi-Fi.

A l'hora de parlar del protocol de dades en un sistema domòtic, la seguretat és un dels aspectes més importants, per això, MQTT suporta el protocol criptogràfic TLS/SSL dissenyat per proporcionar seguretat de comunicacions a través d'una xarxa informàtica.

El protocol de dades MQTT es divideix en dues bandes. El broker, posicionat a la banda del servidor, encarregat de recollir, controlar i repartir tots els missatges en funció del tòpic. Per la banda es troben els clients, els quals es subscriuen als tòpics per obtenir els missatges i envien missatges a un tòpic.

També s'ha de tenir en compte que TLS/SSL suporta l'autenticació, d'aquesta manera el broker pot controlar l'accés dels dispositius i limitar les seves accions en funció a la seva autenticació. MQTT utilitza un model Pub/Sub, aquest model utilitza els tòpics per organitzar els missatges de la mateixa manera que els directors organitzen els arxius (tema/tema/...). A partir d'aquesta estructura, el broker pot limitar l'accés i les accions sobre els temes en funció de l'autenticació de cada client.

MQTT també disposa d'una cua de missatges on es mantenen en el cas que els subscriptors estiguin desconnectats, permetent mantenir la coherència de les dades i entregar els missatges un cop el client es torni a subscriure.



**Figura 9. Sistema simple de comunicació amb MQTT**

En la Figura 9 s'observa un sistema molt simple de comunicació utilitzant MQTT on hi ha dos sensors que publiquen l'estat en un tòpic concret, enviant cada un d'aquests missatges al broquer, mentre que un consumidor està subscript al tòpic per tal que el broker li reenvii cada un dels missatges que els sensors publiquen.

Per altra part existeix **CoAP**, un protocol de transferència web per a la comunicació de sol·licitud/resposta entre un client i un servidor i dissenyat per a dispositius de baix recursos fent-lo ideal per IoT.

A diferència de MQTT, aquest protocol encara no està estandarditzat, el que provoca que a l'hora de dissenyar nous projectes poden faltar llibreries per la implementació dels dispositius IoT dintre del sistema domòtic.

CoAP funciona com una espècie de HTTP per a dispositius restringits. El protocol està dissenyat per a lidiar en un entorn amb un baix amplada de banda i una alta congestió a través del seu baix consum energètic i una baixa sobrecàrrega de la xarxa. Aquesta característica permet funcionar a CoAP en entorns on altres protocols basats en TCP com MQTT no es comuniquen de manera efectiva.

Pel que fa a l'encriptació de les comunicacions el protocol CoAP utilitza DTLS que és

la implementació del protocol SSL sobre el protocol UDP.

En definitiva s'ha observat que CoAP és un protocol més útil que MQTT en sistemes més extensos on la distància entre dispositius IoT és elevada i on les condicions de connectivitat són molt dèbils i amb alta càrrega de treball. Per exemple a l'hora d'utilitzar satèl·lits per a la comunicació dels paquets. Però si es parla d'un sistema local i controlat, com seria una casa intel·ligent, MQTT aporta majors beneficis, des d'un major control en la qualitat dels missatges, un sistema d'autenticacions més complex i una major facilitat en la seva implementació al disposar d'un nombre més elevat de llibreries i dispositius IoT per a la seva utilització.

### **3.3.2 SERVIDOR – USUARI**

Per les comunicacions entre el client i el servidor s'ha tingut en compte que el client ha d'obtenir els estats dels dispositius en temps real, per això a continuació s'observarà un conjunt de protocols en temps real per a utilitzar en diferents interfícies d'usuaris, des d'una pàgina web fins a una aplicació per ordinador.

Com s'ha vist en l'apartat anterior, MQTT disposa d'un gran nombre de llibreries, el problema és que no en tots els casos funcionen igual. Si es parla d'utilitzar el llenguatge de Python, el seu funcionament és ideal i aprofita tot el rendiment que el sistema permet, a més, existeixen més d'una llibreria, cada una especifica per a millorar diferents característiques. La limitació ve donada en la seva utilització amb pàgines web, ja que no és capaç de controlar un flux molt elevat de dades com per exemple el control de les càmeres.

També hi ha el problema de l'encryptació i autenticació per a poder connectar-se amb el broker MQTT perquè són necessaris un conjunt de certificats i el navegador no permet una gestió senzilla d'aquest.

Per altra banda, en el cas del navegador web hi ha el protocol WebSocket, un protocol de comunicació que crea una única connexió TCP. Aquest protocol permet la comunicació entre el servidor i el navegador sent compatible amb els ports de HTTP 443 i 80, cosa que el fa ideal per pàgines web i telèfons mòbils. El protocol WebSocket implementa una comunicació full-dúplex mantenint la seguretat de la web. El protocol està optimitzat per a tots els navegadors del mercat augmentant la velocitat de comunicació respecte a la llibreria MQTT-js.

Si es desitja crear una aplicació per l'ordinador, es recomana utilitzar les llibreries de MQTT mentre que si s'ha d'utilitzar en un navegador web, és més recomanable que la comunicació entre el servidor i el client es faci a través de WebSocket per tal de suportar un flux major i una fàcil l'encryptació.

Per últim i com és típic, la interfície d'usuari necessita els estats actuals, però també els estats anteriors que el servidor s'ha encarregat de guardar en una base de dades. Per aquest tipus de comunicacions unidireccionals amb el servidor s'utilitzarà el protocol HTTPS.

## **3.4 PLATAFORMA**

La plataforma de control és l'encarregada de mantenir i controlar els estats dels



dispositius, atendre a les peticions que realitza el client en funció dels permisos d'aquests i executar tasques en funció als estats dels dispositius o altres esdeveniments.

Per a poder realitzar una de les tasques més importants del sistema, primer de tot s'ha de determinar quin llenguatge de programació és més convenient per a dur a terme la seva funció de la forma més eficient.

### **3.4.1 LLENGUATGE DE PROGRAMACIÓ**

Existeixen un gran nombre de llenguatges de programació, però no tots són capaços de dur a terme aquestes tasques. Els 2 llenguatges que s'han considerat per a dur a terme la plataforma de control són Python i Java, ja que són un dels dos llenguatges més potent i amb una quantitat de suport i llibreries més ben optimitzades.

Tant Python com Java permeten una gran flexibilitat i facilita en controlar diferent mòdul en un sol programa, on cada mòdul és dedicat a controlar una secció de la plataforma. La diferència a l'hora de la implementació, és que Java consumeix una major quantitat de recursos en comparació a Python i la seva creació és més complexa. Si el sistema és molt complex, Java aporta més funcionalitats i permet utilitzant els micro-serveis per a dividir cada mòdul. Per aplicacions que no necessiten d'una quantitat enorme de serveis, Python facilita tant el disseny com la implementació i a més consumeix molt pocs recursos, permetin utilitzar un servidor més lleuger.

### **3.4.2 BASE DE DADES**

Les bases de dades mantindran l'estat dels dispositius, un historial d'aquests i un control d'usuaris i de les característiques extres de les quals disposi el sistema domòtic. Al parlar d'un sistema simple, com és una casa domòtica, una sola i no un conjunt d'elles, l'elecció de la base de dades no és un punt clau, ja que la quantitat de dades que ha de mantenir un sistema d'aquesta mida no és molt elevat. En una casa intel·ligent no és comú trobar més de 100 dispositius IoT, d'aquesta manera s'ha decidit escollir la base de dades que aportï una major flexibilitat al sistema.

D'aquesta manera, es pot escollir entre diverses bases de dades, en aquest cas compararem SQLite i MongoDB tot i que existeix un gran nombre de gestors de bases de dades.

En una casa intel·ligent, no es generen una quantitat de consultes (select) elevada, ja que típicament, el nombre d'usuaris que consulten les dades és baix, però en el cas de les insercions a bases de dades el nombre de peticions és més elevat, perquè si es guarda cada un dels valors que notifiquen els sensors i actuadors implicaria un nombre elevat d'insercions, això no implica que sigui necessari guardar cada un dels valors.

Un sistema d'aquestes característiques implica que el nombre d'insercions és més constant que el nombre de consultes. Tot i això, aquesta diferència no és essencial a l'hora de l'elecció del gestor perquè aquest nombre és mínim amb relació a la càrrega que suporten tant SQLite com MongoDB.

La diferència de SQLite sobre MongoDB, les quals impliquen avantatges, són la mida i els recursos que consumeix del servidor, ja que SQLite sobreposa l'eficiència i la confiança. Pel que fa a MongoDB, aquest té un major consum dels recursos del nostre

sistema i requereix un servei addicional dintre del servidor per a funcionar, però aporta una major velocitat d'insercions.

Per motius d'eficiència i simplicitat s'ha escollit SQLite, ja que aportar una major facilitat a l'hora de la instal·lació en el servidor i tractament d'aquesta. Tot i que MongoDB realitza les insercions de forma més ràpida, s'ha considerat que l'avantatge no és clau perquè el sistema no necessita aquesta velocitat pel seu bon funcionament.

### 3.5 SERVIDOR

El servidor és la màquina encarregada d'allotjar la plataforma de control, el broker MQTT i la pàgina web en el cas de la seva implementació. La màquina ha de disposar d'uns recursos mínims pel seu bon funcionament. S'ha calculat que els mínims necessaris no són molt elevats gràcies al fet que ni el broker MQTT ni Python consumeix una quantitat elevada de recursos i el sistema ha estat dissenyat pensant amb aquestes característiques.

Per evitar un cost econòmic elevat en la compra del servidor, a continuació es mostren dues opcions econòmiques per allotjar el sistema.

La primera opció és la utilització d'una Raspberry pi 4, un ordinador petit d'una sola placa amb un baix cost i les següents característiques tècniques:

- Processador de quatre nuclis Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- Opció d'elecció entre 2GB, 4GB o 8GB de SDRAM
- Connectivitat de 2.4 GHz i 5.0 GHz IEEE 802.11ac
- Port Ethernet de 1 Gigabit
- 40 pins programables
- 2 entrades micro-HDMI
- Una entrada d'alimentació 5V DC via USB-C amb un mínim amperatge de 3 A



**Figura 10.** Raspberry pi 4

En comparació a un ordinador té unes característiques tècniques molt baixes, però és ideal per allotjar un OS (sistema operatiu) amb un consum baix de recursos. S'ha de tenir en compte que l'arquitectura del seu processador es ARM el que pot provocar algun problema si es desitja implementar alguna funcionalitat addicional que no suporta aquesta arquitectura, però actualment està estesa i no suposa un gran problema.

D'altra banda, la Raspberry Pi per defecte utilitza el sistema operatiu Raspberry Pi OS, ús OS creat expressament per a ser utilitzat amb aquesta, però es poden instal·lar

altres sistemes operatius com per exemple l'última versió d'Ubuntu Server 20.04.3 LTS.

El preu de la Raspberry Pi 4 ronda entre els 50€ i els 120€ depenent del model i els components addicionals. S'ha calculat que per mantenir el sistema, la versió de 2 GB de ram és suficient per a un funcionament correcte.

Com a segona opció es pot utilitzar un ordinador vell, però és recomanable que es tracti d'un portàtil, perquè el consum elèctric i l'espai d'una torre és més elevat.

## 4. DISSENY

Abans de començar la implementació s'ha dissenyat l'estructura del sistema de comunicacions entre les aplicacions i els dispositius, els diagrames de flux de cada dispositiu i els diagrames de comunicació entre els dispositius, l'API (application programming interface) i la interfície d'usuari.

Per a la implementació inicial, només s'ha creat una pàgina web per a la interfície d'usuari perquè és el mode més fàcil d'arribar a tots el dispositiu d'una sola atacada, però el disseny s'ha plantejat per a poder afegir una aplicació mòbil o d'ordinador amb les mínimes modificacions.

### 4.1 ARQUITECTURA

#### 4.1.1 ARQUITECTURA DE LA XARXA

S'ha dissenyat la xarxa en funció del que es disposava en el sistema de proves.

En la següent figura s'han plasmat els següents components:

- **Servidor:** Raspberry Pi 4 que allotja l'aplicació , la base de dades, el broker MQTT i la pàgina web.
- **Client:** Aplicació web multiplataforma que es comunica amb el servidor a través de peticions HTTPS i una connexió WebSocket encriptada amb TLS 1.3.
- **Sensor:** Dispositius IoT format pel microcontrolador i un o més mòduls. Es comunica amb el servidor utilitzant el protocol MQTT i una encriptació TLS 1.3.
- **Encaminador extern:** Encaminador que tracta les peticions procedents de l'exterior i les redirigeix cap al servidor si passen el filtre.
- **Encaminador intern:** Encaminador amb Wi-Fi V6 i un major rang de cobertura on es connecten tots els dispositius.

Per a una major seguretat i cobertura s'ha optat per a la instal·lació d'un segon encaminador amb la tecnologia de Wi-Fi 6, tecnologia que com ja s'ha explicat anteriorment (pg. 13) permet la gestió d'un major nombre de dispositius connectats sense una pèrdua considerable de la potència. Aquesta estructura amb dos encaminadors permet un major control sobre els dispositius IoT aplicant un control i filtratge de paquet i ports entre els dos encaminadors.

En el encaminador extern es connecta el servidor per una major facilitat a l'hora de filtrar i redirigir les peticions que provenen de l'exterior.

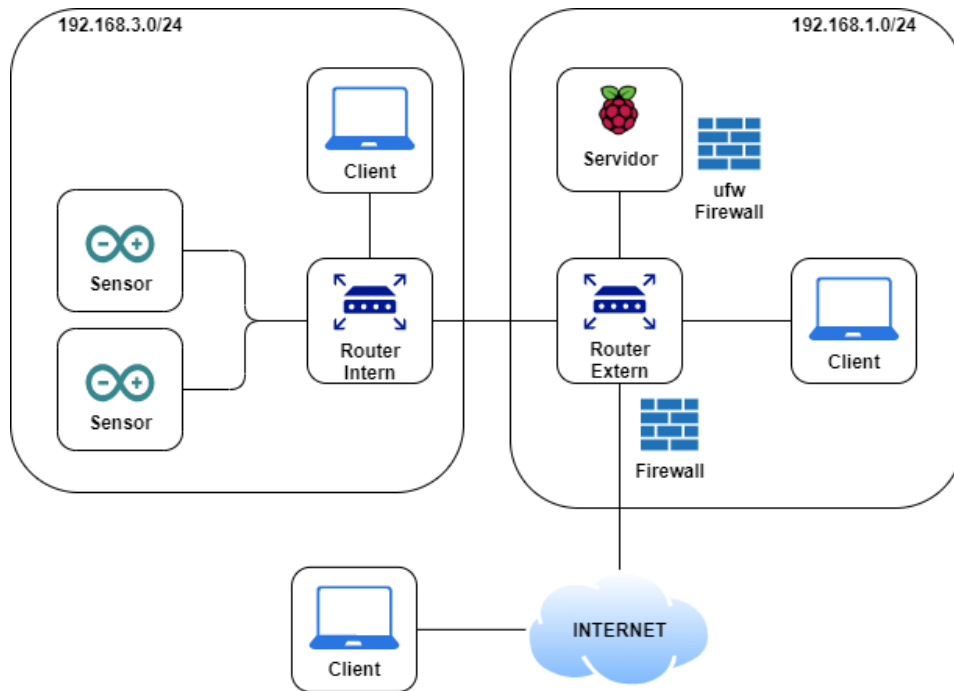


Figura 11. Arquitectura de la xarxa.

Pel que fa als dispositius IoT, aquest s'han connectat a l'encaminador intern per a poder mantenir un major control sobre ells, limitant els paquets que tenen accés a ells. Aquesta segona xarxa també permet que el servidor únicament accepti els paquets dels dispositius IoT que provenen de la xarxa interna.

#### 4.1.2 ARQUITECTURA DE COMUNICACIÓ

Un cop s'han analitzat i escollit les tecnologies de comunicació que s'utilitzen en el projecte, s'ha dissenyat l'estructura de comunicació que es crearà per a interconnectar tot el sistema.

Per accedir al sistema des de l'exterior s'ha decidit obrir l'accés al control de la casa intel·ligent fora de la xarxa local. Per altra banda, només es permetrà la connexió de sensors que es trobin dintre de la xarxa local per a una major seguretat. En el cas de voler afegir sensors de l'exterior, s'haurà de configurar l'encaminador per a redirigir les peticions cap al servidor.

En el següent diagrama s'observen els diferents protocols de dades que s'han utilitzat per a connectar cada un dels apartats que formen el sistema. La connexió entre els dispositius IoT i el servidor s'ha realitzat via MQTT acceptant únicament paquets locals i autènticats. La comunicació entre la interfície i el servidor s'ha dissenyat amb els protocols de dades HTTPS i WebSocket. El protocol HTTPS s'ha utilitzat per a la comunicació amb l'API utilitzant l'estàndard CRUD. En el cas del WebSocket, aquest s'ha dissenyat per realitzar la comunicació en temps real entre el servidor i la interfície.

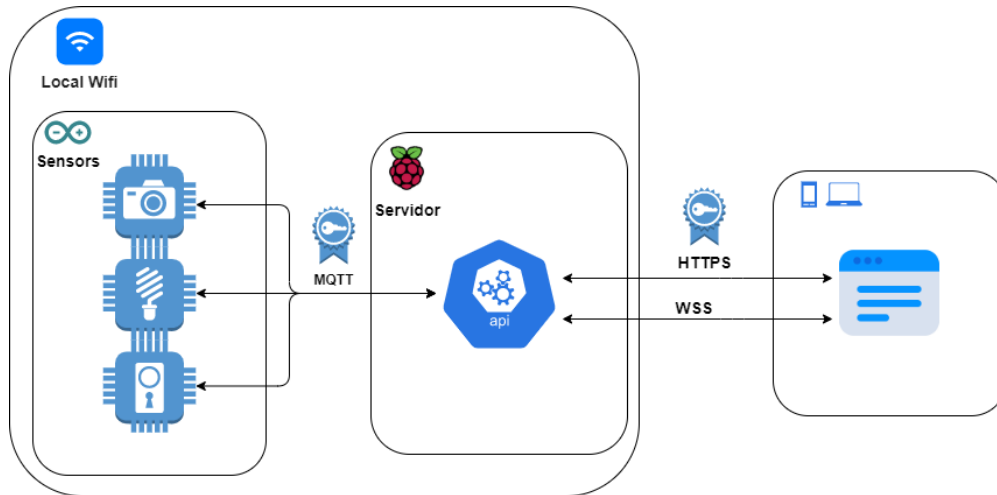


Figura 12. Arquitectura de comunicació.

Com s'observa en la Figura 12 totes les comunicacions entre els sensors i el servidor es realitzen a través de MQTT utilitzant l'enciptació TLS/SSL 1.3 que incorpora el protocol MQTT utilitzant certificats autos signats perquè cada un dels dispositius s'autentifiqui i encripti la comunicació. A més es requereix un usuari i contrasenya per a connectar amb el broker.

Un cop el servidor rep els missatges dels sensors s'encarrega d'executar la seva lògica i posteriorment reenviar el valor cap a la interfície de l'usuari utilitzant WebSocket on anteriorment l'usuari s'ha autenticat per a poder-se connectar. Aquest recorregut també es realitza en sentit contrari, per exemple quan l'usuari envia un missatge via WebSocket a l'api perquè notifiqui a un dispositiu de què realitzi una acció establerta.

Per últim, l'usuari mitjançant peticions HTTPS recupera la informació que l'API manté guardada per tal de poder representar-la. També es pot utilitzar aquest camí per a crear nous usuaris o modificar sensors prèviament creats entre moltes altres opcions que s'han dissenyat en la secció de l'API.

## 4.2 DISPOSITIUS IoT

### 4.2.1 CLASSIFICACIÓ

Per diferenciar els dispositius IoT i poder representar i controlar l'estat s'ha creat un conjunt d'especialitzacions per agrupar els diferents mòduls dels dispositius en funció a les seves característiques de funcionament, format de les dades i accions que realitzen dintre del sistema.

D'aquesta manera s'han creat 4 categories:

- **Càmera:** Aquest mòdul captura i envia imatges a uns fotogrames per segon determinat en forma de bytes i analitza les imatges per detectar moviment.
- **Actius:** Mòdul que recoll i envia dades ambientals de forma activa i envien aquests valors. Es podria entendre com Actiu a un sensor de temperatura que

captura i envia l'estat de l'ambient cada cert temps en nombres reals.

- **Passiu:** Es denomina com a mòdul Passiu aquell que captura esdeveniments de l'ambient generats per alguna causa externa de forma binària (esdeveniment detectat o no detectat). Per exemple un sensor de presència que envia un senyal de moviment quan algú passa per davant.
- **Actuador:** El mòdul Actuador es queda a l'espera d'un senyal numèric per tal de realitzar una acció, també permet activar un temporitzador per aplicar el canvi d'estat. Per exemple, un relé espera a rebre un senyal de l'usuari per encendre o aturar una llum.

Aquestes classificacions s'han creat per a facilitar el control i manteniment del sistema domòtic i poder representar de manera generalitzada els estats de cada mòdul en funció a la seva especialització.

## 4.2.2 CARACTERÍSTIQUES DE CADA DISPOSITIU

Cada dispositiu s'ha dissenyat perquè realitzi un conjunt de funcions diferents i tenir l'opció d'afegir més característiques en funció a les noves necessitats sense un gran canvi sobre el codi. En aquest estat inicial s'han plantejat les característiques essencials que l'usuari ha de poder configurar sobre cada un dels mòduls per especificar el seu funcionament.

**Càmera:** En el cas de la càmera l'usuari pot configurar la velocitat de fotogrames que la càmera envia al servidor per segon i si aquesta ha de detectar el moviment o no. El moviment serà detectat per cada un dels mòduls per alleujar la càrrega del servidor i en el cas de detectar moviment es guardarà una gravació.

**Actiu:** L'usuari pot configurar la freqüència en la qual el mòdul Actiu actualitza l'estat i la freqüència amb què el servidor guarda aquests valors en l'històric.

**Passiu:** Per aquest sensor s'ha configurat el temps que ha de passar per a tornar l'estat inicial. Un exemple seria quan en el sensor de moviment, aquest paràmetre especifica quant de temps sense detecció de moviment és necessari per indicar que ja no es detecta moviment.

**Actuador:** En els Actuadors es poden crear temporitzadors per executar el canvi d'estat en un temps determinat. Per a facilitar el canvi d'estat de manera física, s'han dissenyat interruptors per a interactuar amb el mòdul sense la utilització de la interfície d'usuari.

## 4.2.3 DIAGRAMA DE FLUX

Cada una de les categories anteriors es comportarà d'una manera diferent per a poder realitzar les seves funcions, per això s'han dissenyat els diagrames de flux dels controladors per a cada una de les especialitzacions dels mòduls.

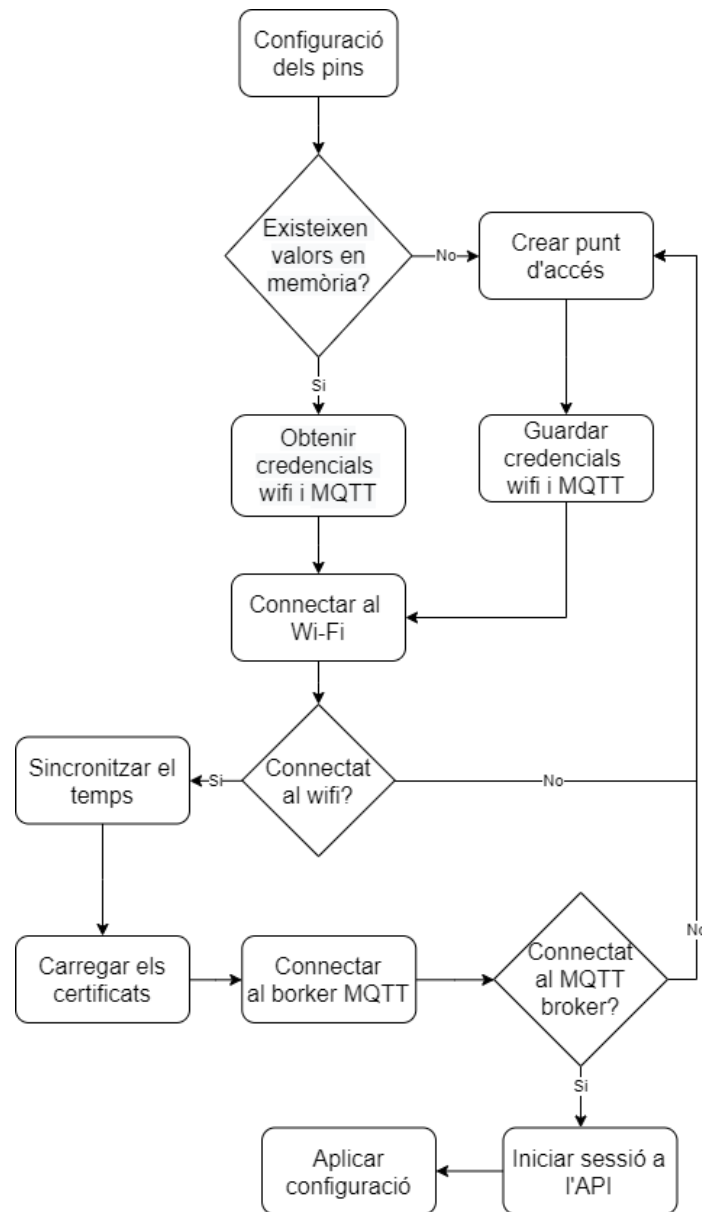
### Inicial

En aquesta secció s'ha creat el diagrama de flux que tots els controladors realitzen a

l'inici de la seva execució per tal d'establir connexió amb la xarxa Wi-Fi local i el broker MQTT.

Per a facilitar la configuració a l'usuari, la primera vegada i en cas d'error en la configuració inicial, s'ha creat un punt d'accés Wi-Fi. A partir del punt d'accés, l'usuari introdueixi les credencials per a connectar i configurar la connexió Wi-Fi i al broker MQTT. Una vegada introduïda les credencials, si els valor són correctes, es guarden en memòria perquè el pròxim cop es connecti automàticament.

A continuació, un cop connectat al broker, s'envia un missatge d'inici de sessió per a obtenir l'estat inicial i els valors de configuració per part de l'API. Per exemple, un relé sempre que es connecta s'inicia amb l'estat desactivat, però l'API manté el seu estat com actiu, un cop es connecti a l'API aquesta informarà el relé del seu estat inicial.



**Figura 13.** Diagrama de flux del dispositiu general

A partir d'aquest diagrama de flux es continuaran els següents.



## Actuador

Un cop s'ha realitzat la configuració general, s'arriba al funcionament que tindrà el mòdul de forma repetitiva. A cada un dels bucles com a primera instància es comprova que encara es manté connexió amb el Wi-Fi i amb el broker, en cas afirmatiu es mira si s'ha rebut algun missatge del broker per tal de poder tractar el missatge, en el cas que el missatge indiqui un canvi d'estat s'aplicarà aquest nou estat i es tornarà a començar.

En el cas que no es tingui connexió amb el Wi-Fi o el broker s'intentarà connectar i a continuació es comprovarà si l'usuari ha premut l'interruptor físic per tal de poder canviar l'estat.

S'ha dissenyat un botó físic que interactua amb el mòdul per a poder utilitzar-lo sense la interfície de l'usuari i a més per si es genera un error en el servidor o en internet. Per exemple, si un relé perd la connexió amb el servidor, però l'usuari vol utilitzar la llum, aquest pot canviar l'estat encara que el servidor no funcioni.

A continuació es pot observar el diagrama de flux dels mòduls Actuadors.

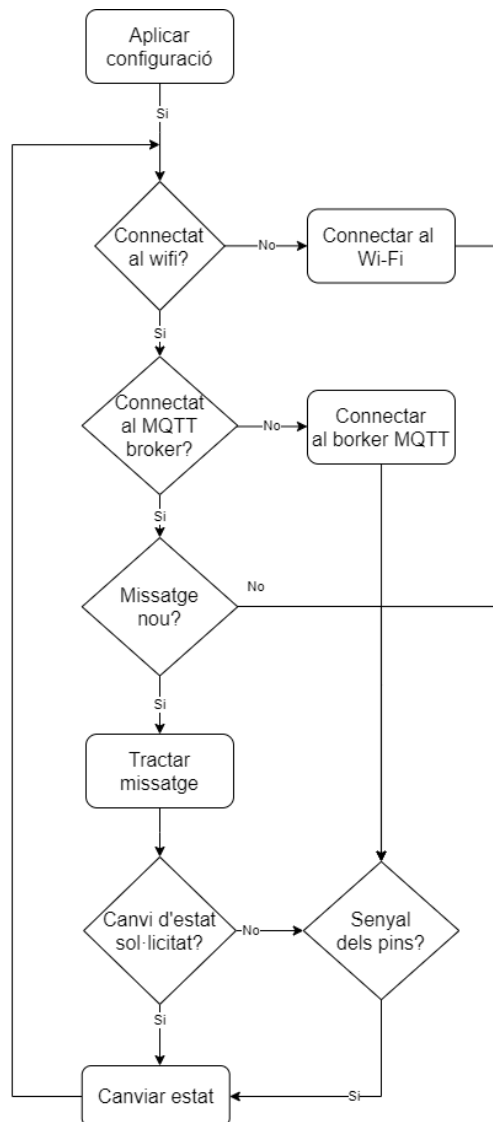
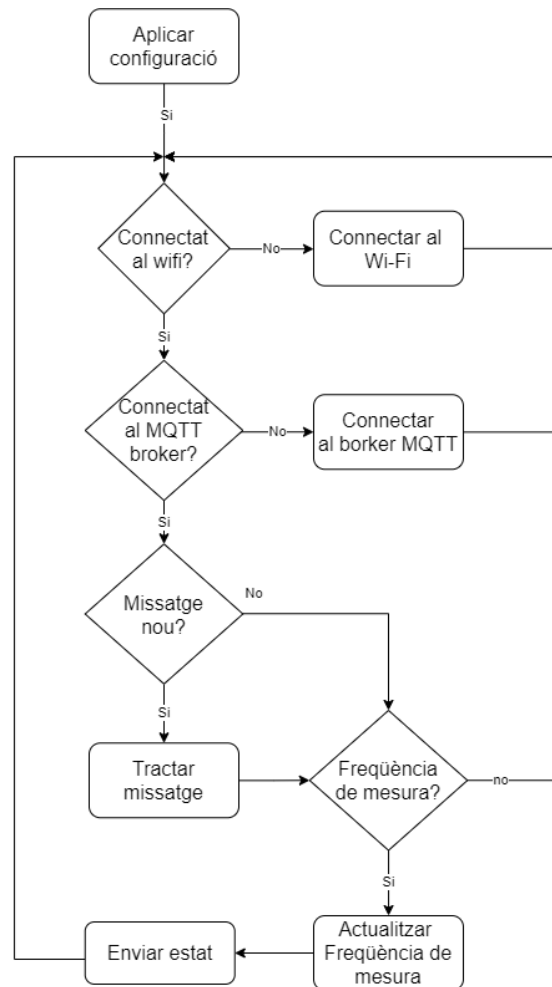


Figura 14. Diagrama de flux del dispositiu Actuador

## Actiu

Igual que amb el mòdul Actuator, es comprova la connexió amb el Wi-Fi i amb el broker, però en aquesta especialització si alguna de les connexions falla, s'espera un temps i es torna a començar.

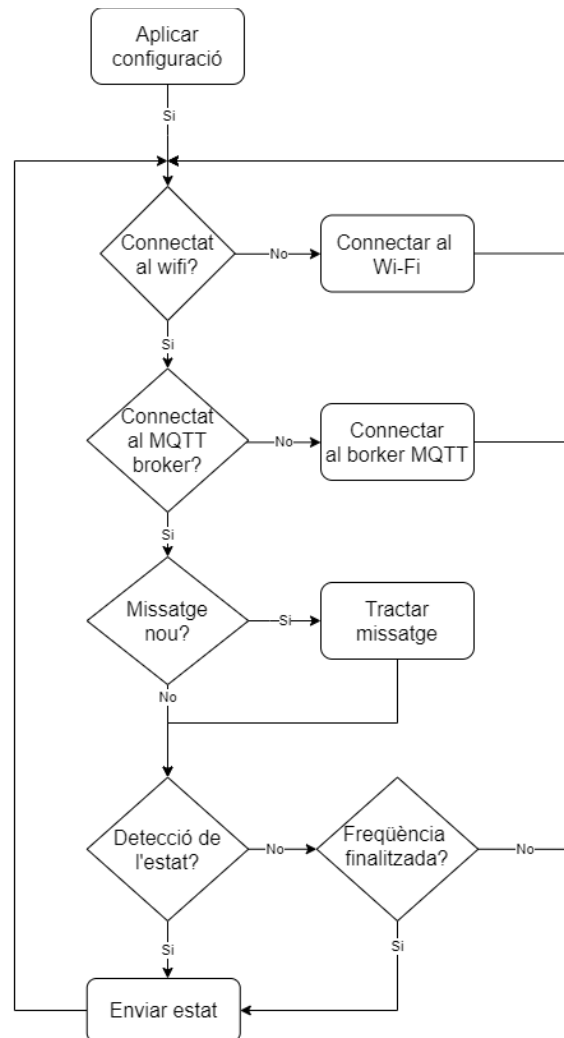
Si es mantenen les dues connexions actives, es revisa si existeix un nou missatge per part del broker, en cas afirmatiu es tracta i a continuació, si ha passat el temps de la freqüència establerta, s'analitza l'ambient i s'envien els valors.



**Figura 15.** Diagrama de flux del dispositiu Actiu

## Passiu

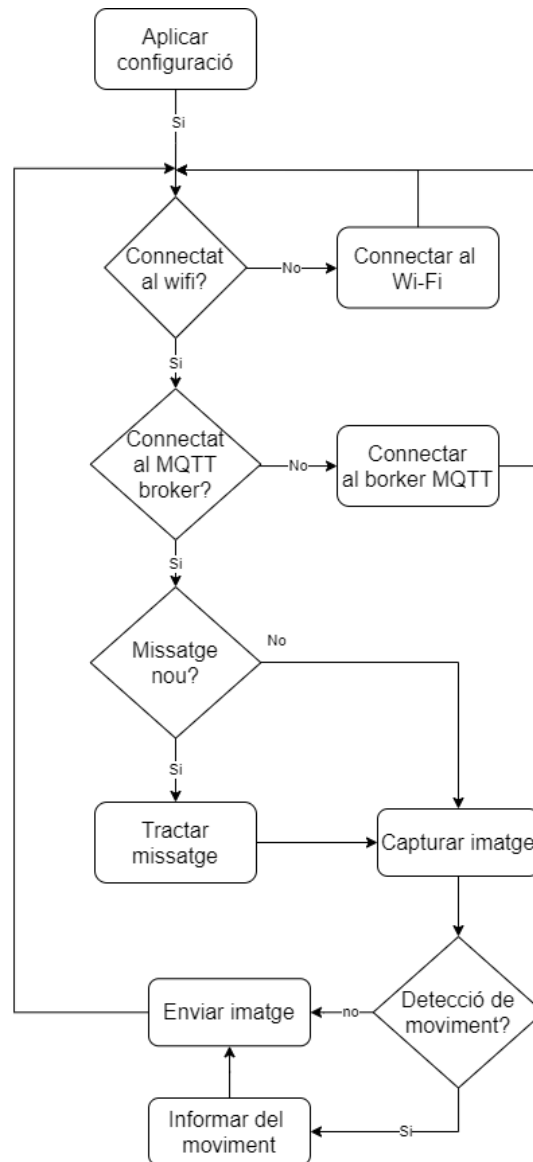
En el cas del mòdul Passiu, aquest actua de la mateixa forma que l'Actiu, però no espera un determinat temps per enviar l'estat, sinó que en detectar l'estat que busca l'envia directament. Quan ha passat la freqüència establerta sense detectar l'estat torna a la posició inicial.



**Figura 16.** Diagrama de flux del dispositiu Passiu

## Càmera

Per últim, el mòdul de la Càmera captura i envia una nova imatge a una freqüència determinada. Per a detectar el moviment, cada segon compara la imatge capturada amb l'última que ha comparat per comprovar si hi ha alguna diferència, en cas afirmatiu informa al servidor del moviment.



**Figura 17.** Diagrama de flux del dispositiu Càmera



els 30 °C es pot llançar una acció que activi l'aire condicionat i una altra acció que tanqui una finestra.

**Acció:** Acció que es desencadena sobre un sensor quan s'activa algun trigger o alguna rutina.

**Historial:** Historial dels estats de cada sensor guardats al llarg del temps. Cada especialització de sensor té el seu propi historial, en el cas de la càmera, guarda l'hora i el nom de l'arxiu un cop s'ha gravat. En el cas de l'Actiu es guarda el valor màxim i mínim de cada dia. Per últim en el cas de l'Actuador i del Passiu guardarà un historial de cada canvi d'estat.

**Historial diari:** Guarda els estats del sensor Actiu cada cert temps durant un període de 24 hores per tal que l'usuari visualitzi els valors de l'últim dia.

**Definició:** Enumerable que classifica les especialitzacions depenent de la seva funció. Per exemple en el cas dels sensors Actius, els classifica en sensors de temperatura i humitat entre d'altres per tal d'una millor representació a la interfície de l'usuari.

**Planificador:** Pla en funció del temps que s'aplica als Actuadors. Per exemple per poder activar cada dia la cafetera a les 8:00 del matí o per activar les llums de la terrassa a l'hora de la posta de sol.

**Rutina:** Rutina creada per un usuari que agrupa un conjunt d'accions que es realitzen a l'hora. Per exemple apagar tots els llums de la casa i activar les càmeres a l'hora d'anar a dormir.

## 4.4 ENDPOINT

Un punt final de l'API és la ubicació digital específica on un programa envia les sol·licituds d'informació per recuperar el recurs digital que hi ha. Els punts finals especifiquen on les API poden accedir als recursos i ajuden a garantir el bon funcionament del programari incorporat.

Pel sistema MQTT es considera un endpoint un tòpic on el dispositiu IoT envia i rep informació de manera establerta.

### 4.4.1 RUTES MQTT

Per tal de llegir i enviar els estats dels dispositius IoT o la configuració d'aquests juntament amb altres accions s'utilitza el protocol MQTT, que com ja hem vist anteriorment utilitza un sistema de Pub/Sub on els tòpics etiqueten i ordenen els missatges emesos, per això s'ha dissenyat el següent sistema de tòpics per a capturar totes les publicacions de manera simple i ordenada.

#### Emès pels Dispositius

##### General:

- <Dispositiu>/login  
Sol·licita iniciar sessió a l'API.

Valor: JSON amb l'ID del dispositiu i dels sensors que controla.

- ping/ack  
Respon la petició de què està connectat.  
Valor: 1
- <Dispositiu>/<Sensor>/state  
Nou estat del sensor.  
Valor: estat

#### Càmera:

- <Dispositiu>/<Sensor>/move  
Detecció de moviment.  
Valor: 1

### **Emès pel servidor**

#### General:

- <Dispositiu>/logged  
Acceptar l'inici de sessió.  
Valor: JSON amb els valors de Configuració inicials.
- ping  
Sol·licita als dispositius que confirmen la seva connexió.  
Valor: ---

#### Actuador:

- <Dispositiu>/<Sensor>/set  
Canvi d'estat de l'Actuador.  
Valor: estat
- <Dispositiu>/<Sensor>/timer  
Sol·licita la creació d'un temporitzador per a canviar l'estat de l'Actuador.  
Valor: JSON amb els segons per al canvi d'estat i en nou estat.

#### Passiu:

- <Dispositiu>/<Sensor>/period  
Envia la freqüència que ha de passar per tornar a l'estat inicial.  
Valor: Temps en segons.

#### Actiu:

- <Dispositiu>/<Sensor>/frequency  
Envia la freqüència de captura de l'estat.  
Valor: freqüència en segons.

#### Càmera:

- <Dispositiu>/<Sensor>/detectMotion  
Activar o desactivar la detecció de moviment.  
Valor: 1 per activar, 0 per desactivar
- <Dispositiu>/<Sensor>/frameRate  
Defineix els fotogrames per segon.

Valor: fotogrames per segon.

## 4.4.2 API

A continuació s'han dissenyat alguns dels endpoint més interessants per tenir al cap a l'hora d'implementar l'API per tal de poder servir les dades a la interfície d'usuari. A partir dels següents es pot reinterpretar com s'implementaria els endpoint d'altres entitats, ja que s'ha utilitzat la filosofia d'una API REST.

### Crides HTTP

L'API utilitzarà com a base del Path el valor `"/API/V1"` per tal d'indicar que s'està apuntant a la versió 1 de l'API.

#### GET:

- `/user/sensors/values/?group`  
Retorna una llista dels sensors de l'usuari sol·licitant amb el seu valor agrupat pel "group" indicat.
- `/sensors/<sensor>/history/`  
Retorna l'historial d'un sensor determinat.
- `/user/sensors/show/?group`  
Retorna una llista de sensors amb l'indicador de si es mostra dintre de la pàgina principal de l'usuari sol·licitant agrupat pel "group".
- `/users/<user>/sensors/<sensor>/?group`  
Retorna una llista de sensors amb un indicador de si l'usuari indicat té accés a ell o no agrupat pel valor "group".
- `/rooms`  
Retorna una llista de les habitacions.
- `/actuators`  
Retorna una llista dels sensors classificats com Actuadors amb els seus paràmetres.
- `/actives`  
Retorna una llista dels sensors classificats com Actius amb els seus paràmetres.
- `/users`  
Retorna una llista dels usuaris.
- `/definition/< especificacio>`  
Retorna una llista de les definicions de l'especificació determinada.

#### POST:

- `/auth/signup`  
Crea un nou usuari a partir dels paràmetres enviats.
- `/auth/login`



Inici de sessió amb els paràmetres enviats i retorna un token per a poder-se comunicar amb l'API.

- /device  
Introdueix un nou dispositiu juntament amb una llista de sensor que aquest conté.
- /room  
Introdueix una nova habitació.
- /definition/<especificació>  
Introdueix una nova definició a l'especificació donada.
- /trigger  
Introdueix un nou trigger amb els sensors desencadenants i les accions que s'executaran.
- /<sensor>/routine  
Introdueix una nova rutina pel sensor indicat.
- /users/<user>/sensors/<sensor>  
Crea la relació entre l'usuari i el sensor indicats.

#### PUT:

- /sensors/<sensor>  
Actualitza el sensor indicat amb els nous paràmetres enviats.
- /rooms/<room>  
Actualitza el sensor indicat amb els nous paràmetres enviats.
- /<especialització>/definitions/<definition>  
Actualitza la definició de l'especialització sensor indicat.
- /users/<user>/sensors/<sensor>/show  
Modifica el valor que indica si un sensor es mostra a la pàgina inicial de l'usuari

#### DELETE:

- /sensor/<sensor>  
Elimina el sensor indicat i en el cas que el dispositiu al qual pertany ja no tenen més sensors també s'eliminarà.
- /rooms/<room>  
Elimina l'habitació indicada.
- /users/<user>  
Elimina l'usuari indicat.
- /users/<user>/sensors/<sensor>  
Elimina la relació entre l'usuari i el sensor indicats.

## WebSocket

El WebSocket utilitza un sistema d'esdeveniments per a dirigir els missatges cap als receptors adequats. Per aprofitar la lògica del sistema MQTT s'ha utilitzat el sistema d'esdeveniment amb la lògica del al sistema MQTT vista en l'apartat anterior.

A continuació es defineix els endpoint del websockets utilitzant el sistema d'esdeveniments per part de la interfície d'usuari i l'API.

### Emès pel client:

- connect  
Envia una petició per a connectares al servidor amb un token que l'autentifica.
- <sensor>/set  
Envia l'estat i el tòpic el qual l'API emet cap al sistema MQTT.
- <sensor>/timer  
Envia el tòpic al qual l'API emet cap al sistema MQTT amb el temporitzador i el nou estat que s'envien cap a l'Actuador

### Emès pel servidor:

- new-device  
Missatge amb la sol·licitant registrar-se d'un dispositiu IoT nou.
- <sensor>/state  
Actualització de l'estat del sensor corresponent.
- <sensor>/add  
Afegeix un nou valor per a la grafica del sensor Actiu.
- <device>/connStatus  
Actualitza l'estat de la connexió del dispositiu.

## 5. IMPLEMENTACIÓ

### 5.1 SERVIDOR

En aquesta implementació s'ha escollit l'ús d'una Raspberry Pi 4 amb el sistema operatiu Ubuntu server 20.04 LTS. L'elecció del sistema operatiu Ubuntu davant del predeterminat de la Raspberry Pi assegura que aquesta implementació és funcional en qualsevol ordinador amb un Ubuntu server instal·lat.

En aquesta secció es mostren els serveis utilitzats per poder servir la pàgina web, redirigir les peticions cap a l'API i mantenir el broker MQTT.

#### 5.1.1 SERVEIS

A continuació es descriu el conjunt de serveis que s'han utilitzat amb les seves implicacions dintre del projecte.

La instal·lació i configuració dels següents serveis s'ha automatitzat utilitzant un script escrit en bash per tal de facilitar els passos inicials.

##### Apache

Apache és un servidor web HTTP de codi obert utilitzat per a crear un servidor segur, eficient i extensible que proporciona uns serveis HTTP sincronitzats amb els estàndards HTTP actuals.

Pel que fa a la pàgina web, Apache s'ha utilitzat per a servir aquesta als usuaris que la sol·liciten i redirigir les peticions que la web realitza cap a l'API. La pàgina web també estableix una comunicació via WSS (WebSocket segur) redirigida cap al BackEnd mitjançant l'ús de les eines Proxy que aporta Apache.

Aquestes redireccions s'han realitzat per obrir el menor nombre de ports possibles cap a l'exterior facilitant el control d'accés únicament amb l'obertura del port 443 i 80. Utilitzant el mòdul *mod\_proxy* i el *mod\_proxy\_http* d'Apache s'ha redirigit cada petició cap al port corresponent en funció a l'URL.

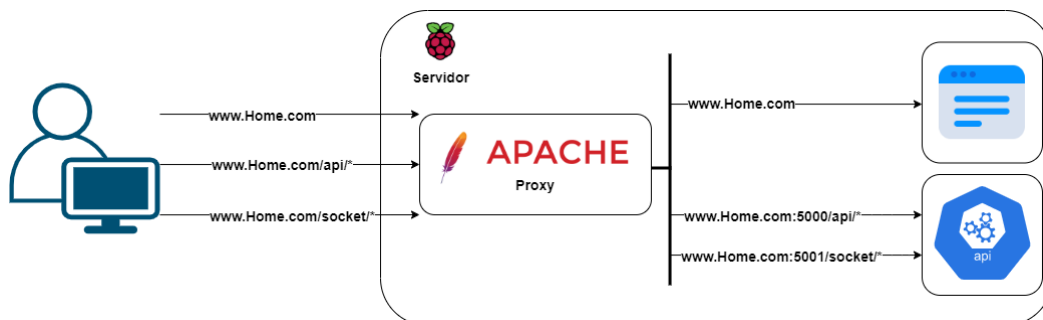


Figura 19. Apache forward Proxy

Si s'observa la figura anterior, en funció a l'URL de la petició, Apache modificarà el port per accedir al servei correcte, permetin arribar a més d'un recurs amb un sol port obert

cap a l'exterior.

A més, Apache permet encriptar les comunicacions que passen a través d'ell, centralitzant l'encriptació de les dades. Per crear els certificats s'utilitza Let's Encrypt, una nova autoritat de certificació capaç d'emetre certificats signats creuadament per *IdentTrust*, que permet que tots els navegadors principals acceptin els seus certificats finals.

Un cop creada la configuració de l'Apache s'ha introduït els certificats utilitzant l'aplicació certbot, un client automàtic i fàcil d'utilitzar que obté certificats SSL/TLS gratis per una adreça web proveïda per Let's Encrypt.

## **Mosquitto**

Eclipse Mosquitto és un broker de missatges de codi obert que implementa les versions 5.0, 3.1.1 i 3.1 del protocol MQTT.

Per a controlar l'accés al broker i mantenir diferents restriccions entre els dispositius IoT i el servidor s'han creat dos usuaris, un per a l'aplicació i l'altre pels dispositius. A partir dels dos usuaris MQTT s'ha creat una llista de control d'accés en funció de l'usuari i del nom donat a la connexió creada. Aquesta llista permet indicar els permisos d'escriptura i lectura de cada usuari sobre cada tòpic existent.

Finalment per implementar el xifratge amb TLS s'ha creat un certificat auto signat per al broker i a partir d'aquest un conjunt de certificats per a cada client que necessiti connectar amb el broker. Per a crear els certificats auto signats s'ha utilitzat la llibreria OpenSSL, la qual aporta les eines necessàries per a generar tots els certificats amb un alt grau de seguretat.

Aquests certificats encripten i autentifiquen les comunicacions entre els clients i el broker. L'autenticació que genera permet que els clients s'autentifiqui utilitzant únicament els certificats i elimina la necessitat d'utilitzar els usuaris i contrasenyes, però per a facilitar la creació de la llista de control d'accés s'han implementat els usuaris.

## **ddclient**

Els proveïdors d'internet no donen una IP estàtica de forma predeterminada, per això s'ha implementat el servei ddclient. Aquest és un servei centrat a actualitzar l'adreça IP del servidor DNS que apunta al domini creat quan la IP pública de casa canvia.

El servei ddclient per Ubuntu comprovarà periòdicament la IP pública del servidor i en el cas de detectar un canvi sol·licitarà al servidor DNS que modifica la IP. Per a configurar aquestes peticions és necessari que el proveïdor DNS accepti DDNS (dynamic Domain Name System), un mètode per actualitzar automàticament la IP del domini.

Per aquesta implementació s'ha utilitzat la plataforma de Google Domains per obtenir el domini i utilitzar el seu servei de DDNS. Per obtenir accés al servei s'ha sol·licitat l'usuari i contrasenya per a realitzar les modificacions i s'han afegit dintre de l'arxiu `"/etc/ddclient.conf"`

Un exemple de configuració és el següent:  
*protocol=dyndns2*

```
use=web
server=domains.google.com
ssl=yes

login=<USUARI>
password=<CONTRASENYA>
<URL>
```

Amb aquest simple arxiu el servei ddclient mantindrà actualitzada la IP pública a la qual apunta el domini.

## Fail2Ban

Fail2Ban és un framework de prevenció d'intrusos que protegeix el servidor d'atacs de força bruta. Aquest software s'ha utilitzat tant per assegurar les connexions via SSH com per evitar un atac de força bruta contra l'API per a descobrir una possible clau d'accés a la pàgina web.

Per a mantenir aquest control el framework permet gestionar el temps de bloqueig que se li aplica a l'adreça IP sol·licitant, el nombre de peticions consecutives per aplicar el bloqueig i el temps que ha de transcórrer entre aquest nombre de peticions.

## 5.2 DISPOSITIUS

En aquesta secció es mostra com s'han implementat els dispositius per a format part del sistema domòtic.

### 5.2.1 LLENGUATGE, EINES I LLIBRERIES

#### Llenguatge

El llenguatge utilitzat per a crear el codi dels microcontroladors és **C++** amb lleugeres modificacions per tal d'aplicar mètodes i funcions addicionals que permeten controlar els pins del controlador i característiques específiques d'aquests.

#### Eines

Per a facilitar la tasca de programar els dispositius s'ha utilitzat **PlatformIO**, una eina professional multiplataforma utilitzada per facilitar el control i instal·lació de les llibreries i el software de control dels microcontroladors. Aquesta eina permet pujar el codi creat al microcontrolador i monitorar el seu funcionament de manera fàcil i amb control d'errors.

Per a utilitzar PlatformIO es necessita una IDE (entorn integrat de desenvolupament) compatible amb el seu funcionament, entre aquestes es troben VS Code, Atom i Eclipse entre altres. Per a la implementació s'ha utilitzat **VS Code** gràcies al fet que disposa una gran quantitat d'ajudes a l'hora de programar com l'autocompletat, depuració del codi i detecció d'error entre moltes altres.

## Llibreries

Les llibreries que s'han utilitzat són necessàries per afegir característiques que permeten aplicar la funció dels diferents tipus de sensors. A continuació s'han plasmat les llibreries que s'han utilitzat per al funcionament dels dispositius IoT.

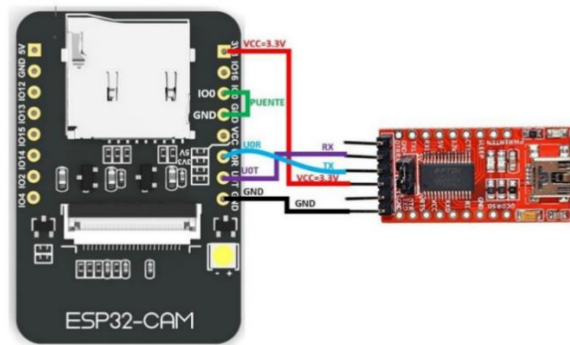
- **LittleFS:** És un petit sistema d'arxius a prova d'errors que treballa amb petites quantitats de memòria creada específicament per a microcontroladors. S'ha utilitzat per a guardar i llegir les credencials Wi-Fi i MQTT.
- **WiFiManager:** Administrador de connexió Wi-Fi per microcontroladors ESP8266 i ESP32 amb un portal de configuració web. S'utilitza per sol·licitar les credencials Wi-Fi i MQTT de forma senzilla per l'usuari.
- **WiFiClientSecure:** Implementa la compatibilitat amb la connexió segura TLS/SSL mitjançant la utilització de certificats. S'ha aplicat per encriptar i autenticar la connexió amb el broker MQTT.
- **PubSubClient:** Proporciona un client per a intercanviar missatge utilitzant el sistema Pub/Sub amb un servidor MQTT. S'ha implementat com a client del protocol MQTT per crear una connexió amb el broker.
- **DHT:** Llibreria enfocada al control de la sèrie de sensors DHT de temperatura i humitat. S'ha utilitzat per obtenir els estats de temperatura i humitat dels mòduls actius.
- **PZEM-004T:** És un Monitor d'energia per a la placa PZEM-004T que utilitza la ModBUS. S'ha implementat per llegir les mostres recollides pel mòdul de monitoratge elèctric.
- **Servo:** Control de posició i moviment d'un servo motor.
- **esp\_camera:** Controlador del sensor ESP32-Cam amb la implementació d'eines de conversió d'imatge. S'ha implementat en el mòdul de càmera per capturar imatges, detectar el moviment i controlar les funcions que aporta el dispositiu ESP32-CAM.
- **ArduinoJson:** Llibreria simple i eficient pel tractament de JSON. S'ha utilitzat per tractar els missatges en format JSON rebuts a través de MQTT.

## 5.2.2 PROGRAMACIÓ DELS MICROCONTROLADORS

Per a poder introduir el codi creat per a cada un dels microcontroladors en necessita un adaptador sèrie d'Arduino. Per a fer aquesta funció es pot utilitzar una placa Arduino com a intermediari entre l'ordinador i el microcontrolador. Però en aquest cas s'ha utilitzat un mòdul adaptador específicament creat per a realitzar aquesta funció com el de la Figura 20.



### ESP32:



**Figura 22.** Diagrama de connexió entre un ESP32 i l'adaptador sèrie

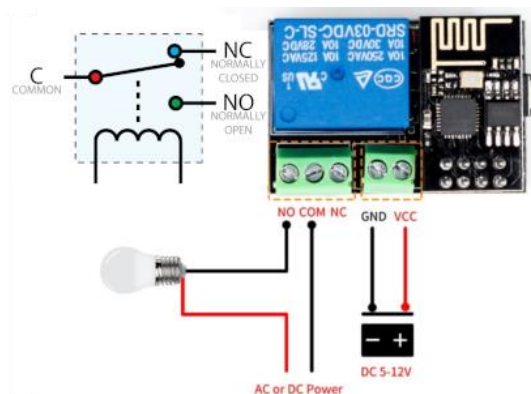
## 5.2.3 ESQUEMA ELÈCTRIC

Un cop el controlador s'ha programat, s'ha connectat a cada un dels mòduls que ha de controlar. D'aquesta manera, en aquesta secció es visualitza l'estructura de les connexions entre un sensor i un controlador creant així un dispositiu IoT funcional. Només es mostrarà un conjunt de dispositius on el seu desplegament varia entre si, ja que a partir dels 3 següents exemples es pot extrapolar l'esquema elèctric als altres dispositius.

També s'ha de tenir en compte que a un controlador se li poden connectar tants de mòduls com nombre de pins disposa pel seu control.

### **Relé:**

La creació d'un relé és de les més simples, ja que s'ha utilitzat un mòdul dissenyat específicament per funcionar amb el ESP-01. Pel seu funcionament únicament es necessita una font d'alimentació de 5V la qual es repartirà entre del mòdul i el controlador. A continuació s'ha connectat la línia elèctrica que es desitja controlar. D'aquesta manera, en rebre un senyal d'aturada des del controlador, el mòdul relé canvia la seva posició.



**Figura 23.** Esquema elèctric d'un relé

Aquest esquema es molt similar per els mòduls dissenyats per a funcionar amb un controlador en específic.



### Sensor PIR:

El sensor PIR (sensor d'infrarojos passiu) és un mòdul de detecció de moviment a través de la captació de la llum infraroja radiada pels objectes. En detectar un canvi en l'estat envia un senyal elèctric que es captura amb un pin GPIO del controlador. Per al seu desplegament s'ha utilitzat una font d'alimentació de 3.3 V pel controlador ESP-01 i el sensor PIR s'ha connectat al controlador per obtenir els 3.3 V i la sortida de les dades que aquest captura.

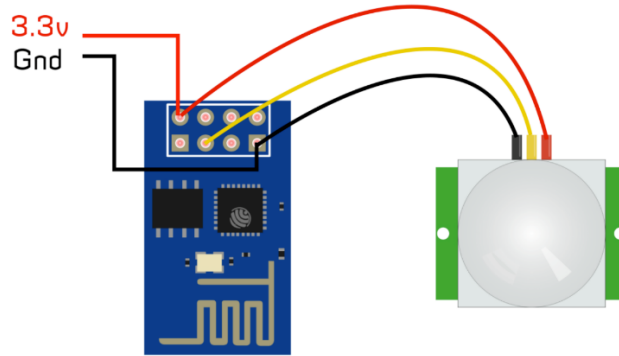


Figura 24. Esquema elèctric d'un dispositiu PIR

Típicament els sensors que únicament capturen una dada segueixen aquest esquema.

### Monitor d'energia PZEM-004T

Pel monitoratge de la xarxa elèctrica s'ha connectat la secció de la xarxa que es desitja controlar, és a dir, si es desitja controlar el consum d'una habitació, es connecta el dispositiu a l'inici de la lineal que entra a l'habitació, si es vol controlar el consum de tota la casa, s'ha de col·locar a l'entrada de la casa.

En aquest sistema, les dades entre el mòdul i el controlador s'envien utilitzant els pins TX i RX, però si es necessita es pot utilitzar una llibreria del controlador per convertir un pin tipus GPIO en un TX o RX. S'ha de tenir en compte que cada controlador estableix quins són els pins GPIO que es poden convertir en TX o RX.

El diagrama de la Figura 25 monitora una xarxa elèctrica de 220 V AC, si es vol monitorar una xarxa de 12 V el diagrama és diferent.

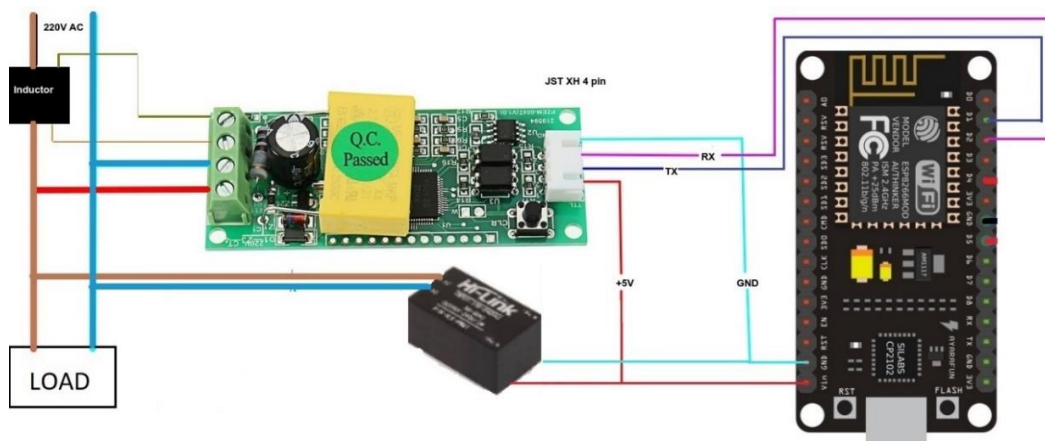


Figura 25. Esquema elèctric per a monitorar una xarxa elèctrica.

## 5.2.4 CONFIGURACIÓ INICIAL

El primer cop que el dispositiu és connectat al corrent, aquest gràcies a la llibreria WiFiManager crea un punt d'accés WiFi amb el nom del dispositiu enfocat a introduir les dades que el dispositiu necessita cada cop que es reinicia. Aquestes dades són les credencials WiFi on es connectarà el dispositiu IoT, el host i el port del broker MQTT, l'usuari i contrasenya del client MQTT, el nom del dispositiu i una llista amb els noms dels mòduls que conte el dispositiu.



Sign in to TempAndHumi  
192.168.4.1

WIFISANPAU_2G	72%
Hotel Universal	64%
MiFibra-7B66	50%
Barcelona WiFi	46%
AndroidAP5256	36%
MiFibra-382F-24G	36%
MiFibra-382E	28%
MiFibra-69F6	22%
DIGIFIBRA-SfxH	18%

DomoHome  
\*\*\*\*\*

192.168.3.3  
8883  
toni  
tonipass  
TAndH  
Temp  
Hum

save

**Figura 26.** Punt d'accés per a la configuració inicial d'un dispositiu IoT.

Com es pot veure en la Figura 26, el dispositiu mostra una llista dels senyals WiFi que detecta amb la seva intensitat juntament amb un camp on introduir cada una de les dades mencionades en el paràgraf anterior.

Un cop s'han guardat les dades, amb la llibreria LittleFS es crea un arxiu dintre de la memòria del controlador guardant cada un d'aquests paràmetres. D'aquesta manera el pròxim cop que el dispositiu s'iniciï no necessitarà que l'usuari torni a introduir les dades.

## 5.2.5 DIAGRAMA DE COMUNICACIONS

En aquest apartat es plasmaran els diagrames de comunicació entre els dispositius i l'aplicació de l'usuari per a cada un dels esdeveniments del sistema.

### Inici de sessió:

Les accions que es realitzen a l'hora d'iniciar sessió són diferents en funció de si el dispositiu ha iniciat sessió anteriorment o no.

Si el dispositiu mai ha iniciat sessió, un cop ho ha sol·licitat a l'API, aquesta envia la informació del nou dispositiu a la interfície de l'usuari on es mostrarà un missatge a l'usuari amb permisos d'administrador perquè introdueixi el nou dispositiu al sistema

juntament amb tots els sensors i valors de configuració que conté. Un cop l'usuari ha introduït el nou dispositiu, l'API envia els paràmetres de configuració amb l'aprovació de l'inici de sessió i inicia la seva escolta.

En el cas que l'API tingui el dispositiu guardat retorna l'aprovació amb els paràmetres de configuració directament, indicant a l'usuari que el dispositiu s'ha connectat i inicia l'escolta cap als estats dels sensors.

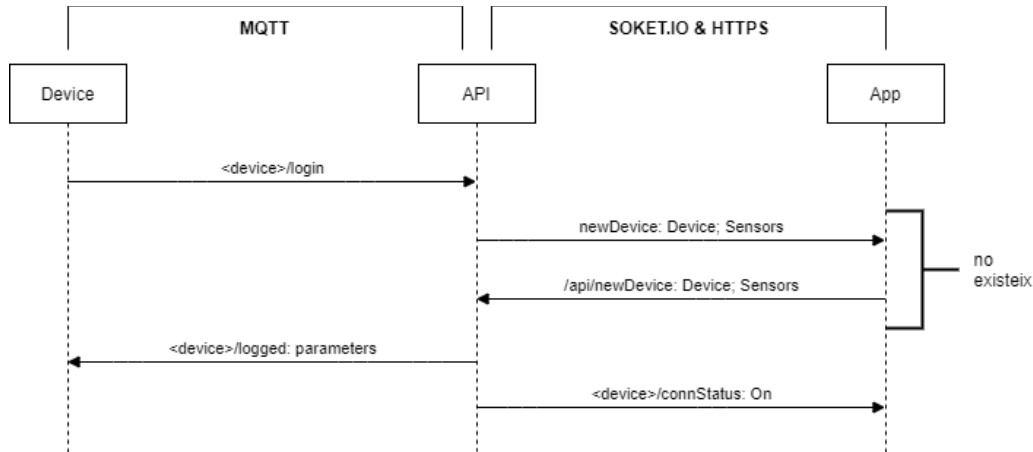


Figura 27. Diagrama de comunicació de l'inici de sessió del dispositiu

### Nou estat:

Cada cop que un sensor actualitza el seu estat envia un missatge a través de MQTT cap a l'API informant del nou estat. Si l'API té el sensor registrar, rebrà el nou estat, el tractarà i el retransmetrà cap a l'aplicació de l'usuari utilitzant el WebSocket sempre que existeixi algun usuari connectat.

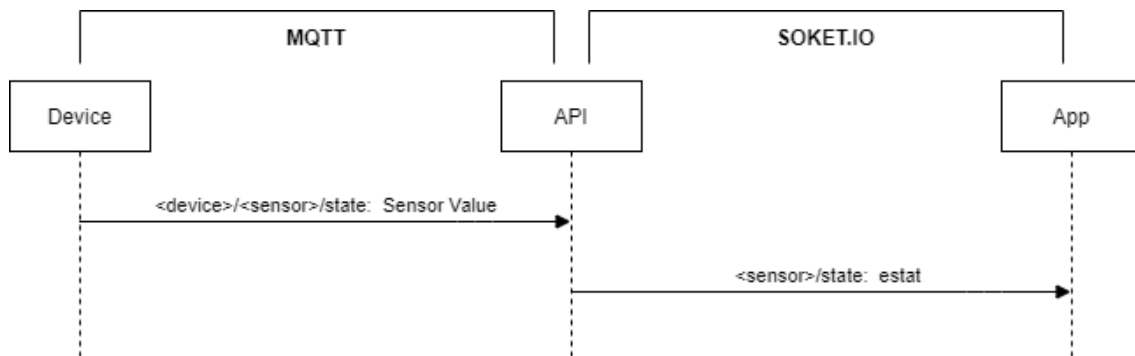
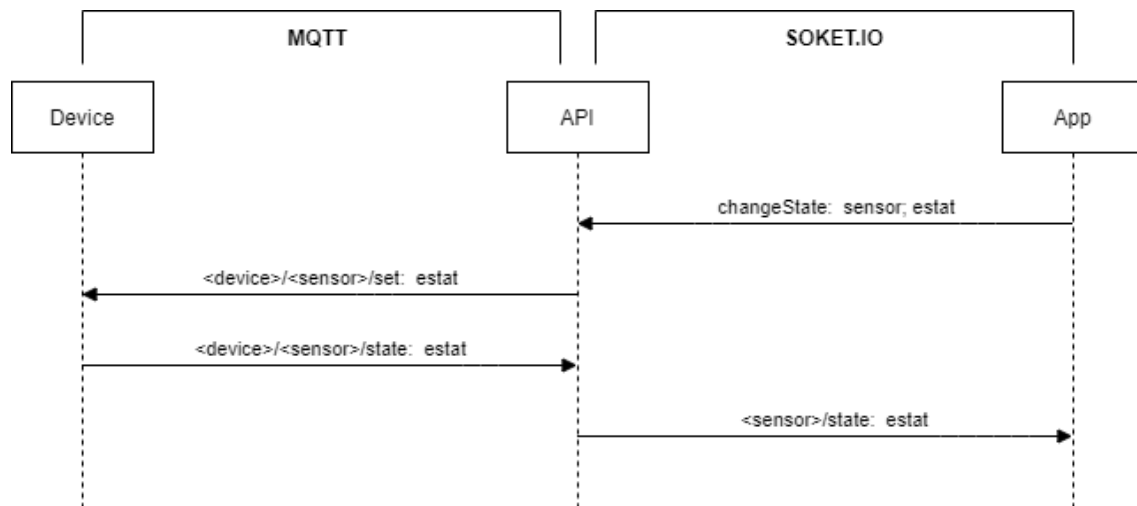


Figura 28. Diagrama de comunicació de l'actualització de l'estat del dispositiu

### Canviar estat de l'Actuador:

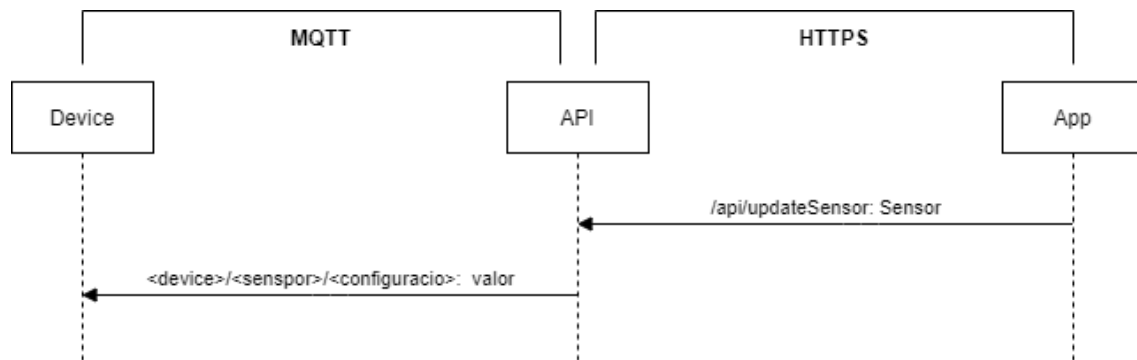
En el cas dels Actuadors, l'usuari pot modificar el seu estat i crear un temporitzador per aquest canvi. L'usuari envia una petició a través del WebSocket cap a l'API perquè aquesta sol·liciti el canvi de l'estat de l'Actuador. Igual que hem vist en el diagrama anterior, un cop s'ha modificat l'estat, el sensor notifica de la seva modificació i es reenvia cap a la interfície de l'usuari per a la seva confirmació. Fins que la interfície no rep la confirmació aquesta no representarà el nou estat sol·licitat.



**Figura 29.** Diagrama de comunicació del canvi d'estat de l'Actuador

### Canviar configuració:

Cada especialització dels sensors té un conjunt de configuracions diferents, quan l'usuari modifica un sensor i es canvia un dels paràmetres de configuració, l'API envia el nou valor cap al sensor. Per exemple parlant del sensor Actiu, si es modifica la freqüència de mesura de l'estat, aquesta serà enviada al sensor per a la seva modificació.



**Figura 30.** Diagrama de comunicació dels paràmetres de configuració del dispositiu

## 5.3 BACKEND

El BackEnd ha de poder resoldre un conjunt de diferents problemes, escoltant i tractant els missatges dels dispositius i clients a l'hora. Per a facilitar el tractament d'aquestes funcions s'ha optat per a crear una sola aplicació formada per diferents mòduls, cada mòdul especialitzat a resoldre una funció específica.

A continuació s'exposaran els utensilis usats conjuntament amb les funcions principals que resol cada mòdul.

### 5.3.1 LLENGUATGE, EINES I LLIBRERIES

## Llenguatge

El llenguatge pel qual s'ha optat per a crear la totalitat del backend es **Python**, aquest ens aporta una gran flexibilitat en poder unificar un conjunt molt gran de tecnologies en una sola aplicació facilitant el tractament i comunicació entre els mòduls que conte el backend.

## Eines

Per a millorar la fluïdesa a l'hora de programar el backend s'ha utilitzat el IDE de **PyCharm**, una IDE que proporciona eines per a facilitar una gran quantitat d'aspectes a l'hora de programar, com per exemple ajuda a mantenir el control de les llibreries de manera simple, indica els possibles errors en el codi, facilita la comprensió del mateix, introdueix una gran quantitat d'ajudes per depurar l'execució del codi en temps real entre moltes altres característiques.

D'altra banda, per a mantenir un control sobre les modificacions i versions del codi s'ha utilitzat **GitHub**, una plataforma per allotjar projectes utilitzant un sistema de control de versions Git.

## Llibreries

- **Dotenv**: És un mòdul que carrega les variables d'entorn d'un fitxer `.env` a un procés `env`. Utilitzat per a recuperar i manipular fàcilment variables de configuració del nostre sistema.
- **Gevent**: Llibreria de xarxa basada en corutines que utilitza `greenlet` per proporcionar una API síncrona d'alt nivell a sobre del bucle d'esdeveniments. Utilitzada per a sincronitzar el mòdul que controla MQTT i el que controla les peticions de l'API.
- **Logging**: Mòdul que defineix funcions i classes que implementen un sistema de registres d'esdeveniment per a mantenir un control sobre el funcionament de l'aplicació.
- **OpenCV2**: Proporciona una biblioteca, eines i maquinària de visió per ordinador optimitzat per a funcionar a temps real. També permet l'execució de models per a l'aprenentatge automàtic i la intel·ligència artificial. S'ha utilitzat per a tractar les imatges rebudes de les càmeres i guardar els fotogrames en forma de vídeo.
- **Sqlite3**: Gestor de bases de dades SQLite basada en disc lleugera que no requereix d'un procés de servidor separat.
- **Paho.mqtt**: Proporciona una classe client compatible amb MQTT v5.0, MQTT v3.1.1 i v3.1. També proporciona algunes funcions d'ajuda per fer que la publicació de missatges únics a un servidor MQTT sigui molt senzilla. S'ha utilitzat per comunicar-se i miniaturitzar els dispositius IoT.
- **PyJWT**: Biblioteca Python que permet codificar i decodificar JSON Web Tokens (JWT). JWT és un estàndard obert de la indústria (RFC 7519) per

representar reclamacions de forma segura entre dues parts. Utilitzat per la creació i tractament dels tokens que autentifiquen la comunicació entre el client i l'API.

- **Flask:** Framework minimalista escrit en Python que permet crear aplicacions web. Està basat en l'especificació WSGI de Werkzeug i el motor de templates Jinja2. Utilitzar per a crear i servir l'API.
- **Flask-httpauth:** És una extensió de Flask que simplifica l'ús de l'autenticació HTTP amb les rutes de Flask. Juntament amb PyJWT es controla l'accés als EndPoints de l'API.
- **Flask-SocketIO:** Proporciona a l'aplicació Flask accés a comunicacions bidireccionals de baixa latència entre els clients i el servidor. L'aplicació del costat del client pot utilitzar qualsevol de les biblioteques de client SocketIO per establir una connexió permanent amb el servidor.
- **Werkzeug:** Llibreria completa d'aplicacions web WSGI. Va començar com una simple col·lecció de diverses utilitats per a aplicacions WSGI i s'ha convertit en una de les biblioteques d'utilitat WSGI més avançades. Controla tota l'encriptació dels usuaris de l'API.
- **Schedule:** Programació de l'execució de funcions Python periòdicament mitjançant una sintaxi amigable. S'ha utilitzat per a realitzar tasques de manteniment dintre del sistema i mantenir les accions periòdiques dels dispositius IoT.

## 5.3.2 MODULS

### 5.3.2.1 BASES DE DADES

Per a mantenir la persistència de les dades s'han guardat en una base de dades SQLite. Una base de dades lleugera i mantinguda en un fitxer, ja que en tractar-se d'una aplicació lleugera i que no genera un gran volum de dades no és necessari fer el pas a una base de dades més potent perquè el que aporta no supera el que es consumeix de recursos. En el cas de desitjar-ho únicament s'hauria de modificar el mòdul de la base de dades seguint l'estructura.

La base de dades s'ha manejat a través d'un mòdul creat per a funcionar de forma conjunta amb tots els mòduls a l'hora utilitzant la llibreria sqlite3 de Python i consultes SQL. Aquest mòdul s'ha implementat mantenint una bona filosofia utilitzant les operacions CRUD (Create, Read, Update, Delete) per tal de facilitar la comprensió i la velocitat de dissenyar el codi.

#### Taules

Per a poder mantenir la lògica de l'aplicació s'han creat les següents taules on es pot observar les claus primàries (PK), les claus forànies (FK) i una descripció de les seves característiques.

- user

name:	varchar(25)	(PK)
password_hash:	varchar(128)	
role:	integer (0, 1)	

L'usuari compta el rol que indica si un usuari és administrador o no. S'ha introduït com un enter i no com un booleà per donar la possibilitat d'afegir més rols.

- home

id:	integer
name:	varchar(15)
status:	integer

En aquest document el sistema està creat perquè els dispositius IoT que es connecten al nostre servidor no provenen de l'exterior per a una major seguretat. Si es desitja controlar un conjunt de cases des del mateix sistema s'ha implementat la taula home i obrir l'accés dels dispositius des de l'exterior per a tindre aquesta opció.

- room

name:	varchar(25)	(PK)
home:	integer	(FK → Home)

Cada habitació pertany a una casa.

- device

device_id:	varchar(25)	(PK)
connect:	boolean	

Un *Device* representa a un dispositiu IoT on aquest controla un conjunt de sensors i es desconnecta del sistema els sensors que compta es deshabilitaran dintre de la interfície de l'usuari.

- sensor

sensor_id:	varchar(25)	(PK)
device_id:	varchar(25)	(FK → Device)
room:	varchar(25)	(FK → Room)
name:	varchar(25)	
type:	varchar(8) (actuador, passive, active, camera)	

Un *sensor* pertany a una *room* i a un sol *device*. És el sensor qui pertany a una habitació i no el dispositiu perquè aquest pot controlar dos sensors que es troben en dues habitacions diferents.

- user\_sensor

user_id:	varchar(25)	(FK → User)
sensor_id:	varchar(25)	(FK → Sensor)
show:	boolean	

La taula *user\_sensor* indica a quins *sensors* té accés un *user* i en el cas de tenir accés la columna show indica si el sensor es mostra dintre de la interfície de l'usuari.

- actuador\_definition

definition:	varchar(25)	(PK)
icon:	varchar(25)	

Definició d'un actuador amb la icona que el representa.

- actuador

- |             |             |                     |
|-------------|-------------|---------------------|
| sensor_id:  | varchar(25) | (PK, FK → Sensor)   |
| value:      | boolean     |                     |
| definition: | varchar(25) | (FK → A Definition) |
- Especialització del *Sensor* que conte el seu valor i la seva definició.
- **actuador\_history**

sensor_id:	varchar(25)	(FK → Actuator)
value:	boolean	
date:	integer	

Historial que guarda els canvis d'estats dels Actuadors.
  - **passive\_definition**

definition:	varchar(25)	(PK)
icon:	varchar(25)	

Definició d'un sensor Passiu amb la icona que el representa.
  - **passive**

sensor_id:	varchar(25)	(PK, FK → Sensor)
value:	boolean	
change_period:	integer	
definition:	varchar(25)	(FK → P Definition)

Especialització del *Sensor* que conte el seu estat i el temps que ha de passar per indicar que ja no es detecta moviment.
  - **passive\_history**

sensor_id:	varchar(25)	(FK → Passive)
value:	boolean	
date:	integer	

Historial que guarda els canvis d'estats dels sensors Passius.
  - **active\_definition**

definition:	varchar(25)	(PK)
icon:	varchar(25)	

Definició d'un sensor Actiu amb la icona que el representa.
  - **active**

sensor_id:	varchar(25)	(PK, FK → Sensor)
save_period:	integer	
take_period:	integer	
definition:	varchar(25)	(FK → Ac Definition)

Especialització del *Sensor* que conte la freqüència d'actualització de l'estat del sensor i la freqüència en la qual aquest estat es guarda dintre de l'historial.
  - **active\_history**

sensor_id:	varchar(25)	(FK → Active)
max:	real	
min:	real	
mean:	real	
samples:	int	
date:	varchar(15)	

Historial que guarda el màxim, mínim, mitjana i el nombre de mostres que es recopilen en un dia.
  - **active\_daily\_history**



sensor_id:	varchar(25)	(FK → Active)
value:	real	
date:	integer	

Historial que guarda els estats de les últimes 24 hores.

- camera

sensor_id:	varchar(25)	(PK, FK → Sensor)
film:	boolean	
min_framerate:	integer	
max_framerate:	integer	

Especialització del *Sensor* que conte els fotogrames per segon màxims i mínims i un booleà que indica si la càmera detecta moviment per a gravar.

- camera\_history

sensor_id:	varchar(25)	(FK → Camera)
name:	varchar(50)	
date:	integer	

Historial que guarda els noms i la data de les gravacions que la càmera ha realitzat.

- trigger

id:	integer	(PK)
active:	boolean	

Identificador d'un disparador amb el valor de si eta actiu o no que conte el conjunt de sensors que el disparen i les accions que es realitzen.

- trigger\_sensor

id:	integer	(PK)
trigger_id:	integer	(FK → Trigger)
sensor_id:	varchar(25)	(FK → Sensor)
trigger:	varchar(25)	

Sensor que pertany al conjunt de disparadors amb el condicional que activa aquest.

- trigger\_action

id	integer	(PK)
trigger_id:	integer	(FK → Trigger)
sensor_id:	varchar(25)	(FK → Sensor)
value:	integer	
timer:	integer	

Sensor que pertany al conjunt d'accions amb el nou valor que s'ha d'aplicar i un possible temporitzador.

- scheduler

id:	integer	(PK)
actuator:	varchar(25)	(FK → Actuator)
every:	varchar(25)	
interval:	integer	
at:	varchar(25)	
value:	boolean	

Planificador que actua sobre un *actuator* de forma periòdica en funció d'un determinat interval.

- routine

id:	integer	(PK)
user :	varchar(25)	(FK → User)
name:	varchar(25)	

Rutina que conte un conjunt d'accions que es realitzaran quan un usuari ho indica.

- routine\_action

id	integer	(PK)
routine_id:	integer	(FK → Trigger)
sensor_id:	varchar(25)	(FK → Sensor)
value:	integer	

Acció que es du a terme quan una rutina és cridada.

- info\_login

id	integer	(PK)
user	varchar(25)	(FK → User)
ip	varchar(40)	
browser	varchar(10)	
version	integer	
os	varchar(10)	
country	varchar(25)	
countryCode	varchar(3)	
regionName	varchar(55)	
city	varchar(55)	
zip	integer	
lat	real	
lon	real	
org	varchar(55)	
date	integer	

Conjunt d'informació que es guarda quan un usuari inicia sessió per a mantenir un control de seguretat.

### 5.3.2.2 API

La base de l'API s'ha creat amb el framework Flask, ja que és un sistema molt lleuger on es poden anar afegint mòduls en funció a les necessitats. En aquest cas s'ha utilitzat únicament el mòdul d'autenticació flask\_httpauth que introdueix les característiques per a mantenir un control dels usuaris que poden accedir a l'API permetin limitar l'accés a determinades rutes en funció a l'autenticació i al rol de l'usuari a través del tractament de tokens JWT. També s'ha utilitzat el mòdul flask\_socketio que permet implementar de manera simple l'API un WebSocket per a crear les comunicacions bidireccionals a temps real entre la pàgina web i l'API.

A partir d'aquest framework s'han creat els EndPoints del BackEnd utilitzant l'estil d'arquitectura REST. Aquesta filosofia s'ha utilitzat per facilitar la comprensió de les crides mantenint un bon estil en les adreces URL i unes operacions ben definides per a cada una de les accions que es poden realitzar (GET, POST, PUT i DELETE). Per a veure un exemple les següents crides controlen totes les accions que es poden fer amb una entitat, en aquest cas les habitacions.

- app.route('/api/v1/rooms', methods=['GET'])  
En el cas de les accions GET s'han utilitzat per a recuperar el conjunt d'una

entitat. Si només es vol recuperar una sola entitat, s'afegirà l'identificador de l'entitat que es desitja recuperar ('/api/v1/rooms/<RoomID>').

Per altra banda, també es pot obtenir un conjunt d'entitats que depenen d'una altra entitat ('/api/v1/users/<user>/sensor').

- `app.route('/api/v1/rooms', methods=['POST'])`  
Les operacions POST indiquen que es vol introduir una nova entitat dintre del seu conjunt passant els valor d'aquesta dintre del body de la petició.
- `app.route('/api/v1/rooms/<identificador>', methods=['PUT'])`  
Aquesta operació PUT indica l'actualització de l'entitat amb l'identificador dintre de l'URL i amb els nous valor dintre del body.
- `app.route('/api/v1/rooms/<identificador>', methods=['DELETE'])`  
Per últim l'operació DELETE sol·licita l'eliminació de l'entitat amb l'identificador indicat en l'URL.

A partir d'aquesta estructura s'han creat cada un dels EndPoints que conte l'API on les respostes es fan en forma de JSON.

Per a mantenir el control sobre l'accés a cada una de les crides s'ha utilitzat la llibreria *flask\_httpauth*, ja que permet aplicar uns requisits d'autenticació a cada un dels EndPoints a partir de la utilització de decoradors.

A l'hora d'implementar aquestes característiques, s'han creat dues funcions, una per a validar el token enviat pel client i una altra que retornarà el rol. A partir d'aquestes dues funcions marcades amb el decorador *auth.verify\_token* i *auth.get\_user\_roles* respectivament es podran indicar els EndPoints que es desitgen restringir l'accés amb el decorador *auth.login\_required* per indicar que abans de procedir a la resposta s'ha de validar la seva autenticació. També permet limitar l'accés per rol, per exemple el decorador *auth.login\_required(role=0)* exigeix que l'usuari sigui un administrador.

Pel que fa al WebSocket, aquest s'ha utilitzat principalment pel mòdul MQTT per actualitzar els estats dels sensors visibles pel client. També s'ha implementat per escoltar al client quan executa una acció sobre els sensors Actuadors.

Igual que els EndPoints de l'API, el WebSocket controla l'accés a partir del token de l'usuari sol·licitant-lo a l'hora de connectar-se a ell. A més s'ha mantingut un recompte dels clients connectats per evitar el reenviament de dades cap al client des del mòdul MQTT si no existeix cap usuari escoltant.

La latència d'una petició per WebSocket depèn de la connexió de l'usuari i la quantitat de dades que es transmeten a través d'aquest, per això, quan es manté una gran quantitat de càmeres que actualitzaven el seu estat afecta la latència de les peticions de canvi d'estat dels Actuadors. Per evitar aquesta latència quan es manté un conjunt elevat de càmeres i obtenir una bona latència d'actuació entre la sol·licitud d'una ordre i la seva execució, s'han implementat dos WebSockets diferents. El socket principal s'ha enfocat al tractament de les dades simples i les peticions cap a l'API mentre que el segon socket s'ha enfocat a mantenir les càmeres.

Amb aquesta estructura s'ha aconseguit aguantar un gran volum de fotogrames per segon sense un afecte sobre la latència dels Actuadors i l'actualització dels altres sensors.

A més, per a millorar la visualització de les càmeres en el FrontEnd, l'usuari pot escollir

una limitació en el nombre de fotogrames per segon que vol rebre a través del WebSocket. S'ha calculat el FrontEnd accepta entre 30 i 90 fotogrames per segon sense observar cap retard entre la captura del fotograma i la visualització. Aquest rang de fotogrames depenen de la connexió a internet i el dispositiu utilitzat. Si se supera el nombre de fotogrames per segon que el dispositiu aguanta es generarà un retard fins a arribar a un màxim d'un minut.

Quan s'arriba al màxim els paquets es comencen a descartar, però la fluïdesa es manté correcta. Aquesta elecció depèn de l'usuari, de si és important per ell observar les càmeres en temps real o prefereix obtenir els màxims de fotogrames per segon.

### 5.3.2.3 MQTT

El mòdul encarregat a controlar les comunicacions sobre MQTT fa ús de la llibreria paho.mqtt per a mantenir el control sobre els missatges dels sensors. S'ha utilitzat la llibreria per subscriure als tòpics del sensor enregistrats dintre del sistema i organitzar els missatges que es reben en funció de cada tòpic. Aquesta característica permet agrupar les diferents especialitzacions dels sensors de manera senzilla i tractar els missatges en funció de cada una.

Al connectar-se al broker MQTT es generés dues subscripcions, una al tòpic on els sensors responen al ping que genera el servidor i una segona on els dispositius sol·liciten connectar-se amb l'aplicació.

El servidor cada 10 segons envia un missatge al tòpic 'ping' on cada un dels dispositius IoT estan subscrits. Quan els dispositius reben aquest missatge responen al tema 'ping/ack' per tal que el backend pugui mantenir un control dels dispositius que estan connectats al broker MQTT. Si passen els 10 segons i un dispositiu no ha respost es considera que el dispositiu s'ha desconnectat del broker MQTT.

En el cas de l'inici de sessió dels dispositius, el servidor es manté a l'espera de rebre un missatge al tema '/login' amb els paràmetres especificats. En el cas d'estar prèviament enregistrats, el servidor respon amb els valors anteriorment establerts per l'usuari i s'informarà la interfície que el dispositiu està connectat. Si el dispositiu no està enregistrat dintre del sistema, s'informa a l'usuari que hi ha un nou dispositiu a l'espera de configuració i un cop l'usuari l'ha configurat, els valors es guardaran per futures ocasions.

A continuació s'exposa el tractament de les dades en funció de l'especialització dels sensors que les envia:

- **Actuador i Passius:** Els actuadors i els passius envien un valor que generalment indica si el mòdul està encès o aturant. El primer que s'ha implementat quan es rep un missatge és la redirecció cap a l'usuari per a la seva previsualització, a continuació es guarda dintre de la base de dades actualitzant el seu valor i a l'hora es genera una nova fila a la taula de l'historial indicant que l'estat s'ha modificat amb l'hora de la modificació. Finalment es revisa si existeix algun trigger on aquest canvi dispari les accions.
- **Actiu:** Quan es rep un valor dels sensors Actius i el temps es major a la freqüència que l'usuari ha escollit que es guardi dintre de la base de dades es crearà una nova instància dintre del historial dels sensors Actius i a continuació s'informa al client que actualitzi la gràfica del sensor. En el cas de que el temps

no sigui major a la freqüència no es guardarà dintre de la base de dades, únicament s'informarà a l'usuari del nou estat del sensor. Per últim i en els dos casos es comprovarà si existeix algun trigger on aquest canvi generi una acció.

- **Càmera:** Les càmeres emeten a dos tòpics diferents, un per a indicar que la càmera ha detectat moviment i un segon per enviar les imatges capturades. Quan la càmera indica que ha detectat moviment, el backend inicia una gravació per a la càmera que ho ha indicat, estableix un temps de gravació, envia un missatge a la càmera indicant els fotogrames per segon que ha d'enviar a partir d'aquest moment i comprova si existeix algun trigger que es vegi afectat. En el cas que ja s'entrigui gravant quan es detecta el moviment, s'actualitzarà el temps de gravació.

Per altra banda, quan la càmera envia un fotograma el backend comprova si se superen els màxims fotogrames per segon que es poden enviar al frontend, aquest màxim l'estableix l'usuari. En cas de no superar el màxim s'envia el fotograma cap al frontend. A continuació es comprova si s'ha de gravar la càmera que ha enviat el fotograma, en cas afirmatiu, s'afegirà el fotograma a l'arxiu que s'està utilitzant per a crear el vídeo. Si el temps de la gravació s'ha superat, es finalitzarà la gravació i es guarda el vídeo amb el nom de la càmera, el dia i l'hora. Per últim s'afegirà el nom de la gravació a l'historial de les càmeres.

### 5.3.2.4 PLANIFICADOR

El BackEnd ha d'executar un conjunt de tasques de forma periòdica creades per l'usuari i definides pel sistema per a mantenir el bon funcionament de l'aplicació. Per a crear aquesta lògica s'ha utilitzat la llibreria *Schedule*.

El planificador manté dues accions principals que es repeteixen de forma periòdica per a mantenir la lògica i el funcionament dels altres mòduls.

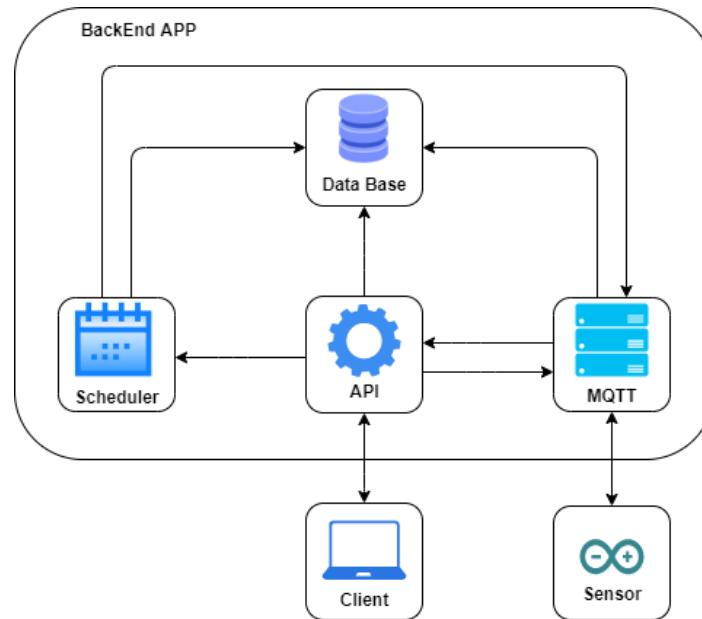
Una de les accions que es defineixen en primera instància és la d'enviar un missatge ping de forma periòdica als dispositius IoT a través del mòdul MQTT per a poder mantenir un registre dels dispositius que es mantenen connectats al broker.

La segona acció s'encarrega cada hora de tractar les mostres guardades més antigues dintre de l'historial diari dels sensors Actius i guardar-les dintre de l'historial general unificant els seus valors i eliminant les mostres utilitzades de l'historial diari.

Per altra banda, els usuaris poden crear tasques periòdiques sobre els sensors Actuadors definint canvis d'estat en funció de l'hora i el dia. Aquestes tasques són guardades dintre de la base de dades i mantingudes per aquest mòdul.

### 5.3.3 ARQUITECTURA

En la següent Figura 31 es poden observar les visibilitats entre cada un dels mòduls, permetin la comunicació entre ells en funció de les necessitats.



**Figura 31. Arquitectura del BackEnd**

Com es pot observar en la figura anterior, l'API té visibilitats sobre cada un dels mòduls, ja que l'usuari ha de poder realitzar totes les funcions, des d'enviar un missatge a un sensor utilitzant el mòdul de MQTT fins a introduir una nova tasca al planificador.

Per altra banda, El mòdul MQTT pot inserir i comunicar-se amb la base de dades i accedir a l'API per tal d'utilitzar el WebSocket i poder enviar els estats dels sensors cap al client.

Pel que fa a mòdul de la base de dades, aquest únicament és utilitzat pels altres mòduls per tal de persistir i accedir a les dades guardades dintre de la base de dades SQLite.

Finalment el planificador pot accedir a la base de dades per tal de mantenir l'historial del sensor Actius i per crear les tasques periòdiques dels sensors Actuadors a l'arrancar. També disposa d'accés al mòdul MQTT per a poder enviar els missatges ping i executar els canvis d'estat periòdics dels sensors Actuadors creats per l'usuari.

## 5.4 FRONTEND

A continuació s'exposen els aspectes més importants en el desenvolupament de la pàgina web.

### 5.4.1 LLENGUATGE, EINES I LLIBRERIES

#### Llenguatge

Els llenguatges que s'han utilitzat per a la creació completa de la pàgina web són JavaScript, HTML i CSS. **JavaScript** és el llenguatge principal del FrontEnd

encarregat de mantenir la lògica de la interfície de l'usuari. Les funcions principals de les quals s'encarrega JavaScript és la realització i tractament les crides a l'API, gestionar els components que s'han de renderitzar i controlar els valors dels Dispositius IoT que es visualitzen.

L'**HTML** estructura els continguts de la pàgina web. És a dir, s'encarrega d'organitzar i pintar els components dintre de la pàgina representa els valors rebuts des de l'API.

Per últim el **CSS** dona estil als components que l'HTML introdueix dintre de la interfície.

## Eines

Per a facilitar la creació de la pàgina web s'ha utilitzat el IDE **WebStorm** de JetBrains. Una IDE enfocada principalment a la programació amb JavaScript la qual aporta una gran quantitat d'eines per a facilitar el tractament d'error, depurar del codi en temps real, importar i controlar les llibreries de manera senzilla entre altres eines i extensions disponibles.

Una altra eina molt interessant a l'hora de dissenyar una pàgina web són les eines de desenvolupador que aporta el mateix navegador. Utilitzant aquestes eines (Normalment es poden obrir prement la combinació de tecles ctrl + shift + i) es pot observar l'estructura de la pàgina web i fer modificacions en temps real. També permet visualitzar les crides realitzades a l'API juntament amb les dades i el temps de cada resposta, veure la consola del navegador i els valor guardats a l'emmagatzemament local.

## Llibreries

Les llibreries amb més important que s'han utilitzat per a la creació de la pàgina web són les següents:

- **React:** Llibreria JavaScript de codi obert dedicada a construir interfícies d'usuari. React es pot utilitzar per a crear aplicacions mòbils o pàgines web amb la filosofia de SPA (aplicacions d'una sola pàgina), és a dir, una web on la interacció amb l'usuari reescriu dinàmicament l'estructura de la mateixa pàgina amb informació nova en lloc de sol·licitar una pàgina totalment nova per a cada interacció. Aquesta llibreria s'encarrega de l'administració i representació dels estats en el DOM, és a dir l'estructura de l'HTML, per això és necessari la utilització d'altres llibreries per a implementar més funcionalitats a la pàgina web.
- **Material ui:** Conjunt de components React per a un desenvolupament web més senzill amb el disseny de Material Design. S'ha utilitzat per a donar un disseny atractiu i uniforme a tota la interfície de l'usuari.
- **DX React chart:** Llibreria de tractament i representació de gràfiques híbrides que combina HTML i SVG. Permet la representació de l'historial dels sensors amb gràfiques molt variades i visualment atractives.
- **Notistack:** Gestor de visualització i tractament de notificacions per pàgines web altament personalitzable.

- **Redux:** Utilitzat per a la gestió de l'estat de l'aplicació web. Aquest manté l'estat de tota una aplicació en un sol arbre d'estats permetent compartir aquests entre els components de manera senzilla.
- **Socket.io:** Llibreria que permet la comunicació en temps real, bidireccional i basada en esdeveniments entre el navegador i el servidor amb la utilització de WebSockets. Aquesta llibreria s'ha utilitzat per a poder obtenir i modificar els estats dels sensors.
- **Axios:** Llibreria JavaScript que s'utilitza per a realitzar sol·licituds XMLHttpRequests des del navegador. Utilitzat per a executar i interceptar sol·licituds i respostes HTTP cap a l'API i permet la protecció contra XSRF.
- **Base-64:** És un codificador/descodificador base64 robust. S'ha utilitzat per transformar les fotografies de base binària a base64 per a la seva representació de manera senzilla i ràpida.

## 5.4.2 COMUNICACIÓ

La comunicació entre el FrontEnd i l'API per a obtenir les dades genèriques del sistema s'ha utilitzat la llibreria *AXIOS* conjuntament amb un middleware de *Redux*. El middleware és l'encarregat d'obtenir el token de la resposta en iniciar la sessió i inserta aquestes credencials dintre del cap de cada petició per tal que l'API l'accepti.

Quan el middleware obté el token, aquest el guarda dintre de l'emmagatzematge local per a poder accedir a ell cada cop que ho necessita. Per a mantenir el token actualitzat, juntament amb ell es rep el temps d'expiració, quan falta menys d'una hora perquè el token sigui invàlid, es realitza una petició automàtica al servidor sol·licitant nou token, el qual substituirà l'actual.

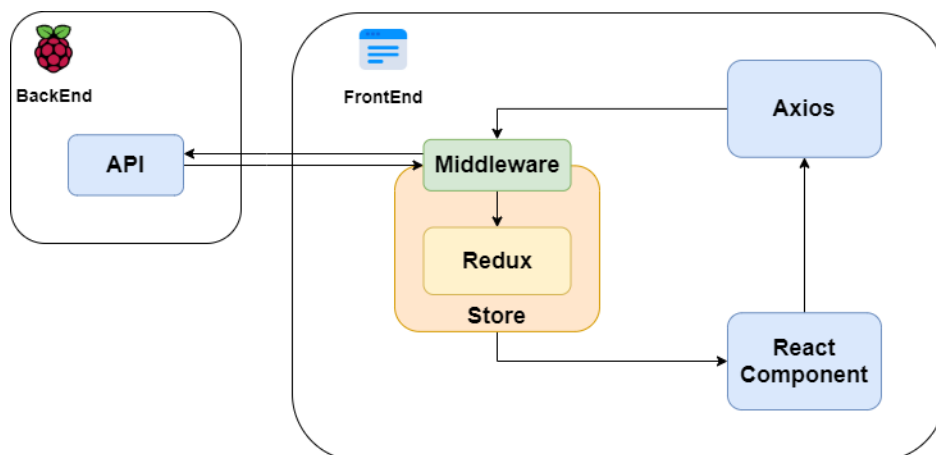


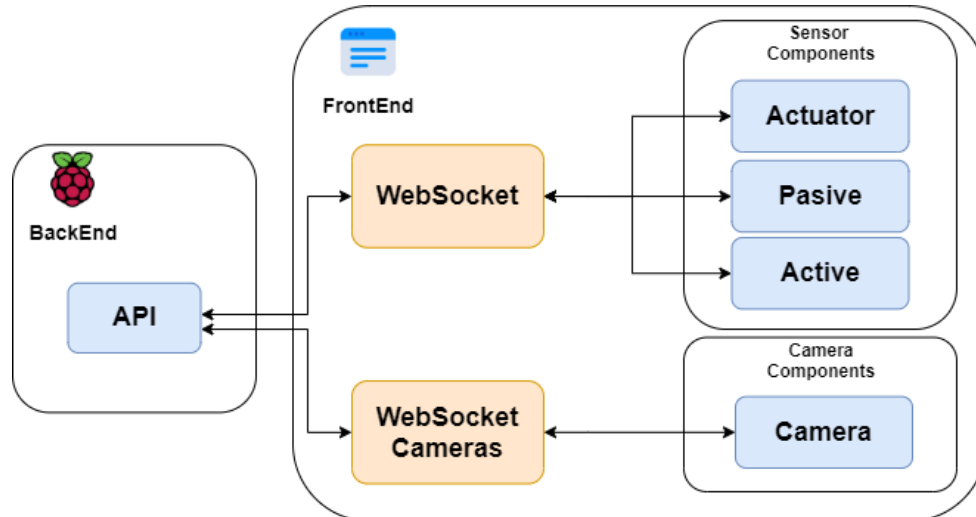
Figura 32. Estructura de la comunicació axios dintre del FrontEnd.

Quan a través d'un component es realitza una petició utilitzant la llibreria Axios, la petició s'envia al middleware, el qual s'ha utilitzat per a realitzar les cridades i encarregar-se de les respostes. En rebre la resposta, aquest extreu la informació que necessita i reenvia la resposta al Redux perquè la guardi dintre del seu arbre d'estats, per tal que el component que ha realitzat la petició pugui accedir al nou estat i



actualitzar la seva representació.

Pel que fa a les dades en temps real, el FrontEnd i l'API es comuniquen utilitzant dos WebSockets, un per als fotogrames de les càmeres i un segon per a totes les altres dades.



**Figura 33.** Estructura del WebSocket dintre del FrontEnd

En iniciar sessió i amb el token que es rep s'estableix una connexió amb el WebSocket de l'API. Un cop creada la connexió és cada component dels sensors qui s'encarrega d'escoltar els esdeveniments del WebSocket que a cada un li convé, a més, el component de l'actuador pot enviar missatges cap a l'API.

En el cas del WebSocket de les càmeres únicament es crearà la connexió si la interfície de l'usuari té alguna càmera per representar, si no no es crearà la connexió. Igual que amb el socket principal és cada un dels components de les càmeres qui s'encarrega d'utilitzar el socket per actualitzar l'estat.

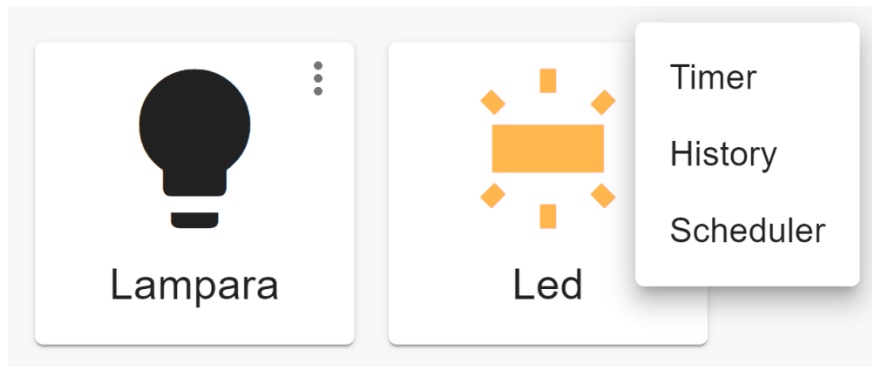
### 5.4.3 REPRESENTACIÓ I TRACTAMENT DELS SENSORS

A continuació es mostra el funcionament i la forma en la qual s'ha decidit implementar la representació de cada especialització dels sensors.

#### Actuador

Els Actuadors representen un enter, aquest enter indica l'estat del mòdul, per exemple en el cas de parla d'un relé, aquests possibles valors són 0 o 1, ja que el relé només té dues posicions. En el cas d'una tira LED RGB a més de mantenir el valor d'aturat i encès també es necessita indicar la intensitat de cada color.

Per tal de controlar les petites variacions entre una mateixa especificació s'ha utilitzat la *definició*, la qual indica aquestes petites diferències. La definició també estableix la icona que es mostrarà i les accions addicionals.



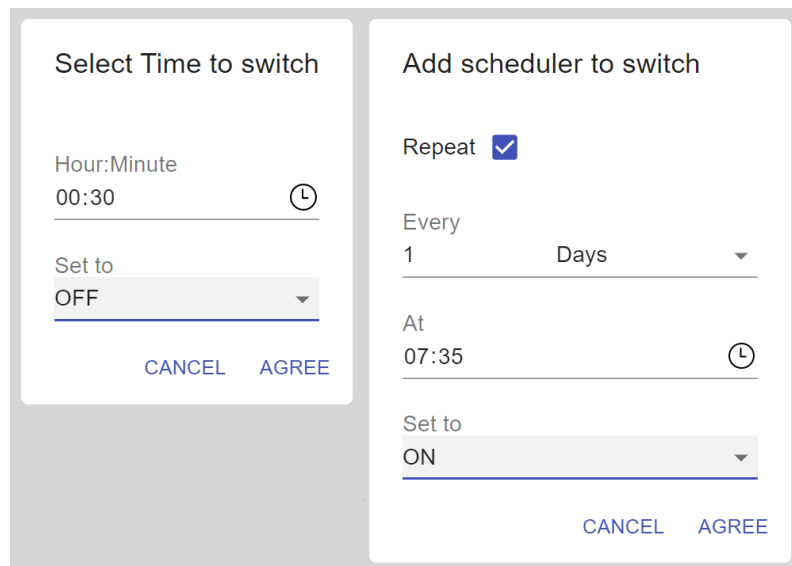
**Figura 34.** Targeta visual del sensor Actuador.

Com es pot veure en la Figura 34, la targeta dels Actuadors representa l'estat de forma visual a partir del color amb el qual es pinta la icona. En el cas que el sensor es trobi desconnectat, la targeta es mostrarà de color gris.

Per executar l'ordre de canviar el seu estat, s'ha implementat l'acció en clicar a sobre de la targeta. En clicar-se s'envia a través del WebSocket una petició al dispositiu IoT sol·licitant que modifiqui el seu estat entre encès i aturat. Un cop rep el nou estat modificarà el color de la icona en funció al valor rebut.

A la part superior dreta de cada un dels Actuadors apareix una icona per a desplegar el menú d'accions addicionals. Dintre del menú s'han inserit les accions addicionals, aquestes són: el temporitzador per a canviar l'estat, l'opció de visualitzar l'historial de canvi d'estats del sensor i un planificador per a poder crear rutines en funció de les hores, el dia o el dia de la setmana, per exemple que la llum de l'habitació s'encengui entre setmana a les 8:00 h.

En la següent Figura 35 es poden observar els formularis emergents que apareixen en clicar sobre les opcions del menú d'un Actuador per a crear un temporitzador i un planificador.



**Figura 35.** Formularis emergent del sensor Actuador.

Si la definició de l'Actuador té alguna funció especial, aquesta també s'ha inserit dintre

del menú. Per exemple, en el cas anterior mencionat d'una tira LED, apar de poder aturar o encendre els LED, s'ha de poder configurar el seu color, aquesta acció de seleccionar el color s'ha introduït dintre del menú.

## Actiu

Els sensors Actius representen un nombre real, però la visualització més important és l'historial de les últimes 24 hores. Per això s'ha decidit que la targeta mostrarà el valor més recent a baix del seu nom i una visualització minimalista de la gràfica de les últimes 24 hores.



Figura 36. Targeta visual del sensor Actiu.

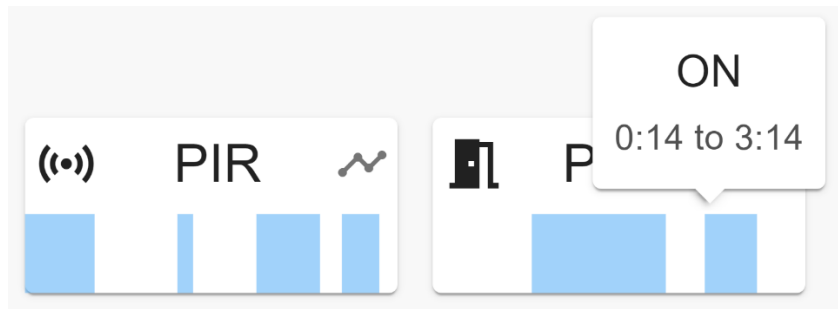
En la part superior de la targeta es mostra el nom del mòdul i el valor actual el qual es va actualitzant cada cop que rep un nou estat a través del WebSocket. A partir de la definició s'extreu la icona que es mostra a l'esquerra de la targeta i les unitats en què es representa el valor. A la part inferior es mostra la gràfica de les últimes 24 hores.

En passar el ratolí, o el dit en el cas de telèfon mòbil, sobre la gràfica es mostra un requadre on es visualitzà els valors de la gràfica, amb l'estat i l'hora en què es va recollir la mostra.

Per a poder visualitzar l'historial complet es pot clicar a la icona superior de la dreta per a poder obrir la finestra emergent on es mostra l'historial. Aquest serà representat en funció del que es necessiti. Es pot visualitzar el mínim, màxim, la mitjana i la suma dels valors de cada dia permetin crear una bona visualització independent-ment del tipus de dades que recopila el sensor.

## Passiu

La representació del sensor Passiu genera una gràfica de les últimes hores on es mostra en quin instant ha canviat d'estat i durant quant de temps s'ha mantingut. Igual que en la targeta dels sensors actius, en passar el cursor per a sobre de la gràfica es mostra l'estat i el rang d'hores que s'ha mantingut l'estat.



**Figura 37.** Targeta visual del sensor Passiu.

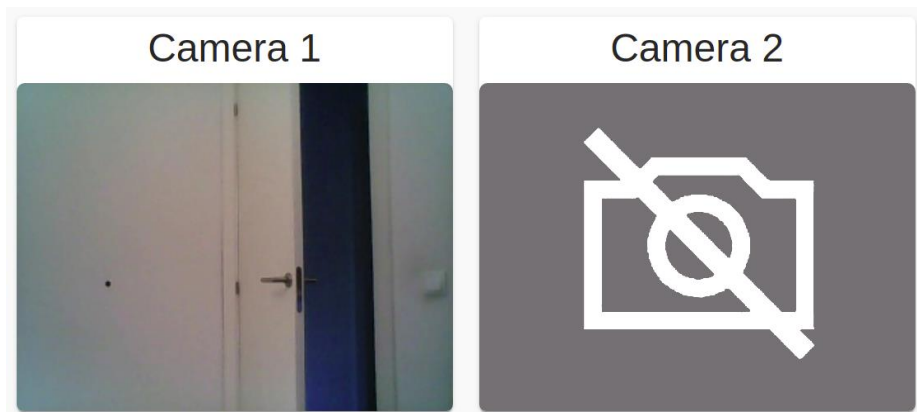
S'ha escollit aquest tipus de representació de l'estat perquè és la forma més senzilla d'observar el que ha captat un sensor passiu durant un període de temps.

Per a visualitzar tots els canvis d'estats es pot accedir a l'historial clicant a sobre de la icona de la part superior dreta.

## **Càmera**

La targeta creada per a representar l'estat de la càmera inicialment conté una imatge predeterminada, la qual es mostrarà com a indicatiu de què no s'ha rebut cap imatge o que s'ha perdut la connexió amb la càmera.

Un cop rep la primera imatge de la càmera, s'ha utilitzat la llibreria Base-64 per a convertir la imatge rebuda a través del WebSocket en bytes a base 64 perquè l'HTML pugui representar. Per a cada un dels estats que rep s'actualitza la imatge fer l'efecte d'estar reproduint una gravació.



**Figura 38.** Targeta visual del sensor Passiu.

Com s'observa en la figura, la targeta de la càmera mostra el nom del sensor en la part superior i si no s'agrupen per habitació, es mostra el nom just d'avall el nom del sensor. La imatge es pinta en la part inferior de forma responsiva per adaptar-se a cada resolució.

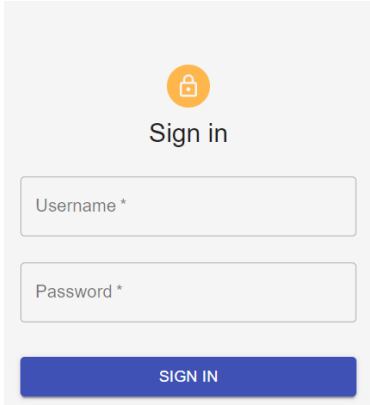
## 5.4.4 PANTALLES

A continuació es visualitzaran les diferents pantalles i finestres que formen la pàgina web i la navegació entre cada una d'elles juntament amb l'explicació del seu funcionament i utilitat.

### Inici de sessió

L'inici de sessió ataca a l'API amb les credencials introduïdes obtenint un token per a poder realitzar les consultes utilitzant el token. A l'hora d'accedir a la pàgina web, si el navegador no disposa d'un token anteriorment guardat o el que té ha caducat, la web redirigirà a l'usuari al formulari d'inici de sessió.

El formulari s'envia a l'API i espera la resposta, si aquesta resposta és afirmativa, es guarda el token de la resposta en l'emmagatzemament local del navegador i es redirigeix l'usuari a la pàgina inicial. Quan la resposta indica un error d'autenticació s'indicarà amb un missatge a sobre del formulari, en qualsevol altre cas es mostrarà el motiu amb una notificació emergent.



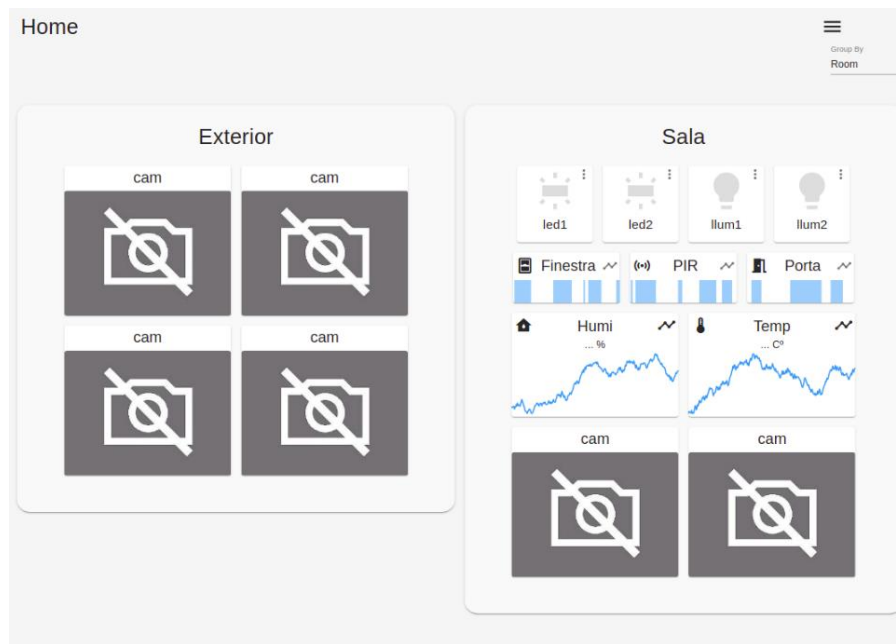
The image shows a login form with a light gray background. At the top center is an orange circular icon containing a white padlock. Below the icon is the text "Sign in". There are two input fields: the first is labeled "Username \*" and the second is labeled "Password \*". Below these fields is a blue button with the text "SIGN IN" in white capital letters.

**Figura 39. Pantalla Login.**

### Pagina principal

Dintre de la pantalla principal es pot controlar i visualitzar tots els sensors enllaçats a l'usuari utilitzant les targetes que els representen. Dintre de la Figura 40 s'observa com es visualitza la pàgina principal des d'una pantalla amb una resolució alta amb els sensors agrupats per l'habitació a la qual pertanyen. Els sensors es poden agrupar per habitació, especialització i pel dispositiu IoT. En aplicar l'agrupació s'estructuren en forma de quadrícula en funció de l'amplada de la pantalla.

Dintre de la pantalla principal únicament es visualitzaran els sensors que l'usuari tingui accés i la seva visualització es trobi activada. Per últim cal indicar que l'ordre dels sensors dintre d'una mateixa agrupació, com en aquest exemple es veu en el requadre de la "Sala", s'estructuren per l'especialització.



**Figura 40.** Pantalla principal.

### Afegir un dispositiu IoT

Quan un dispositiu IoT sol·licita el registre dintre del sistema, els usuaris amb rol d'administrador tindran accés al formulari per acceptar o denegar el dispositiu sol·licitant.

El dispositiu IoT envia el ID amb una llista dels sensors que controla. Depèn de l'especificació de cada sensor, en la pantalla es mostrarà el formulari pertinent per introduir tots els valors necessaris per al seu control.

En la Figura 41 s'observa com el dispositiu IoT amb el ID "TAndH" sol·licita ser registrat dintre del sistema amb els sensors "Temp" i "Hum" els quals són sensors actius.

Home

## Add Device

TAndH

Temp

Name \* Incorrect entry.

rooms type

Show in home

save period take period

Save in DB in min. Update value in seconds.

Hum

Name \* Incorrect entry.

rooms type

Show in home

save period take period

Save in DB in min. Update value in seconds.

SUBMIT

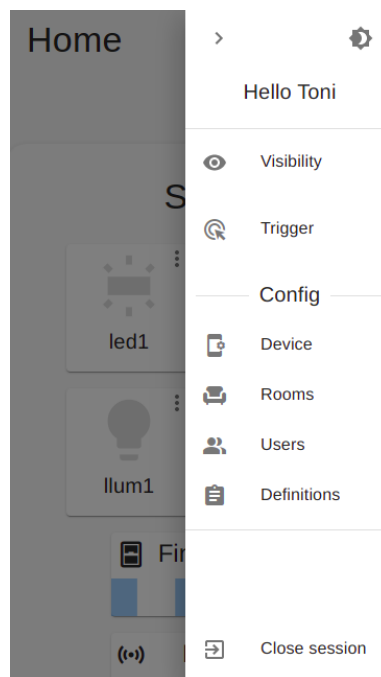
RESET DEVICE

**Figura 41.** Pantalla per afegir dispositius IoT.

Per a declinar la sol·licitud es pot cancel·lar o restablir el dispositiu amb els valors de fàbrica eliminant tots els paràmetres prèviament guardats dintre del dispositiu.

## Menú de navegació

A partir del menú lateral es pot navegar a totes les pantalles d'una forma simple i accessible a més de poder variar entre del tema clar i el tema fosc. En el menú únicament es mostrà les opcions de configuració si l'usuari registrat és un administrador.



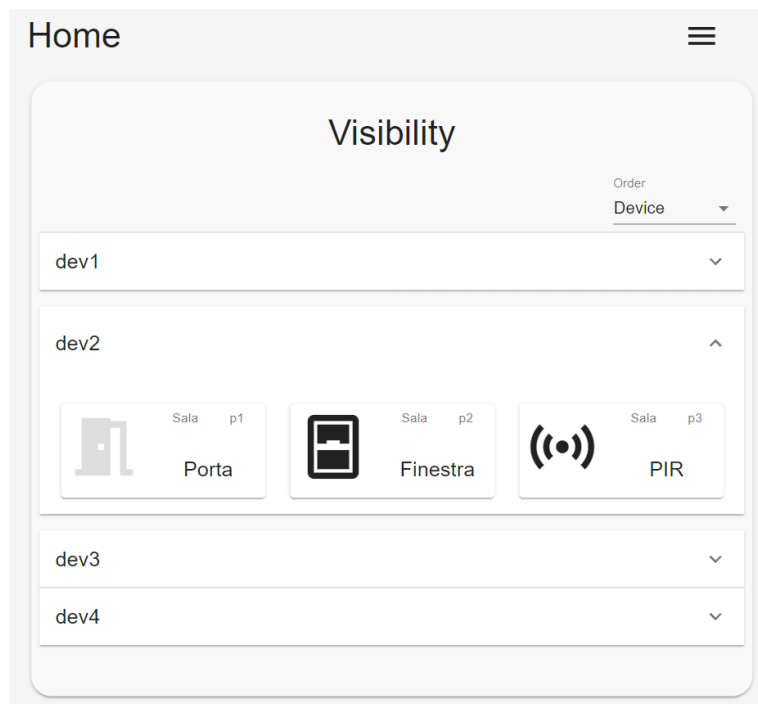
**Figura 42.** Menú de navegació entre pantalles.

## Control de visibilitats

La pantalla per controlar els sensors que són visibles dintre de la pantalla principal mostraran els sensors que l'usuari té accés agrupats per habitació, especialització o per dispositiu. Aquestes agrupacions s'han representat en forma d'acordió per facilitar la navegació.

Cada un dels sensors es representarà en forma de targeta amb el nom, l'identificador i l'habitació on estan ubicats. Les targetes són botons i en clicar a sobre s'envia la petició per canviar l'estat de la visibilitat. Aquesta visibilitat es veurà representada pel color de la icona de cada sensor.

En l'exemple de la Figura 43 s'observa com els sensors estan agrupats pel dispositiu IoT al qual pertanyen i dintre del dispositiu "dev2" la visibilitat de la porta es troba desactivada.



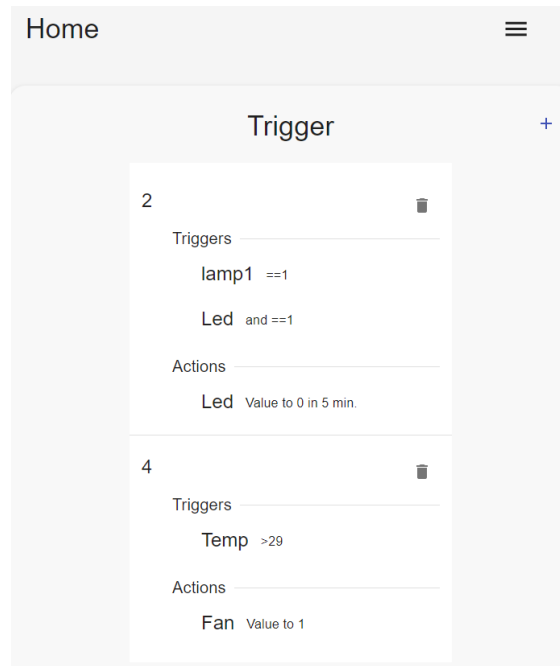
**Figura 43. Pantalla per controlar les visibilitats.**

## Control dels disparadors

Dintre de la pantalla dels disparadors s'ha implementat una llista dels disparadors creats amb les condicions que l'activen i finalment les accions que es desencadenen. Dintre de la llista apareix l'opció per eliminar cada un dels disparadors.

En aquesta pantalla també es troba l'opció per introduir un nou disparador. En clicar en la icona de la part superior dreta se sobreposa un modal amb el formulari per a introduir un nou disparador.





**Figura 44. Pantalla de configuració dels disparadors.**

En la Figura 45 es mostra el modal encarregat d'afegir un nou disparador al sistema. Per a la seva creació s'introduirà com a mínim una condició i una acció. El format de les condicions depenen de l'especialització de cada sensor, per exemple en els actuadors i passius es comprovarà si l'estat és 0 o 1 i en el cas dels actius es comprovarà el valor numèric del seu estat amb l'indicat a la condició.

En introduir una nova condició s'ha d'indicar si s'ha de complir la condició actual i l'anterior per a desencadenar les accions o únicament és necessari que es compleixi una de les dues condicions. A la part inferior de la secció es pot introduir una nova condició o eliminar l'última introduïda.

En la secció de les accions es poden seleccionar els sensors actuadors per a modificar l'estat si es compleixen les condicions o creat un temporitzador per executar la modificació. També permet indicar a una càmera que executi el mode de gravació. Igual que amb les condicions es poden controlar el nombre d'accions amb les icones de la part inferior.

The screenshot shows a web application interface for creating a trigger. The modal is titled "Add Trigger" and has a close button in the top right corner. It is divided into two main sections: "Sensor" and "Action".

In the "Sensor" section, there are three dropdown menus: "Sensor" (selected: p2 Finestra), "When" (selected: Closed), and "Value \*" (selected: 29). Below these are minus and plus buttons for adding or removing sensors.

In the "Action" section, there are two dropdown menus: "Sensor" (selected: l3 Ilum1) and "Action" (selected: ON). Below these are minus and plus buttons for adding or removing actions.

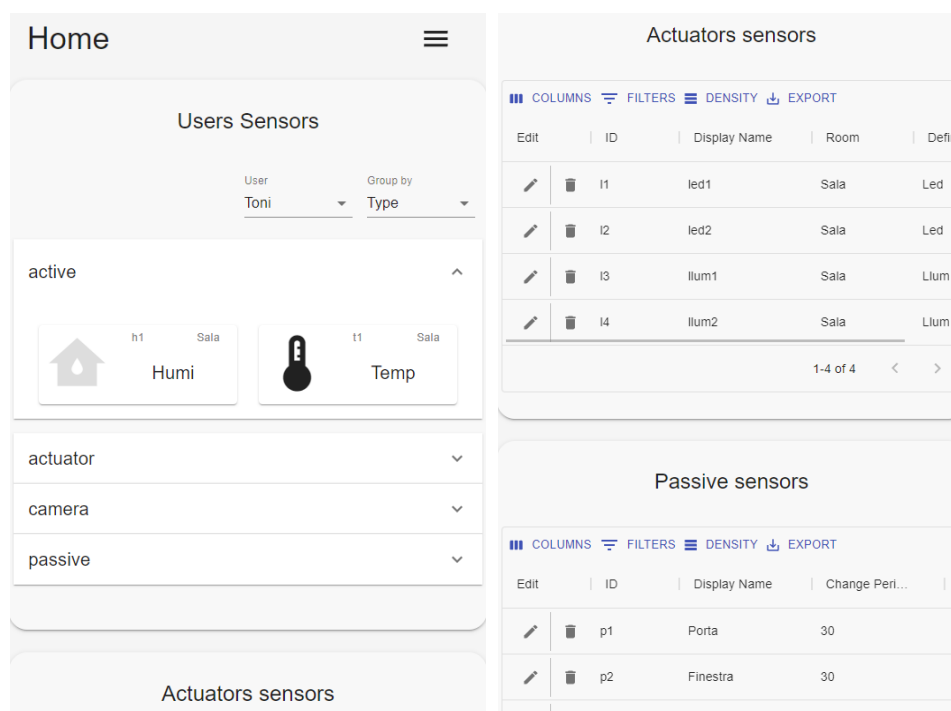
A "SAVE" button is located at the bottom of the modal.

**Figura 45.** Modal de creació d'un disparador.

## Gestió dels sensors i dispositius

Dintre de la pantalla de gestió dels sensors es troben dues seccions diferenciades, una per a controlar la llista de sensors al qual l'usuari hi té accés i una segona per visualitzar, modificar i eliminar cada tipus de sensor.

En el control d'accés als sensors per a cada usuari s'ha escollit una representació pareguda a la secció de visualització vista anteriorment (Figura 43). La diferència és que s'ha introduït un desplegable per a seleccionar l'usuari sobre el qual es modificarà l'accés als sensors.



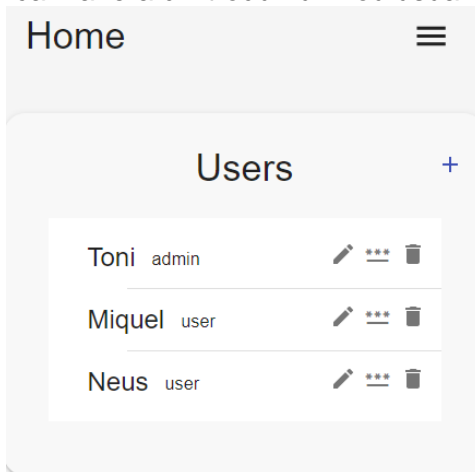
**Figura 46.** Pantalla de gestió dels sensor.

A continuació, com es pot observar en l'anterior Figura 46, la gestió dels sensors es divideixen en seccions en funció de la seva especialització. Aquesta gestió s'ha realitzat a través d'una taula on es pot editar i eliminar de manera senzilla cada un del sensor usant les icones de l'esquerra. Quan es clica una icona, se sobreposa un modal per a executar l'acció clicada.

## Gestió d'usuaris

La pantalla per a gestionar els usuaris s'ha escollit utilitzar una llista per a la seva representació amb les accions a l'esquerra de cada usuari. Les accions que es poden realitzar sobre un usuari són: la seva edició, la modificació de la contrasenya i l'eliminació de l'usuari. L'execució de cada una de les accions utilitza un modal per a la seva realització.

Per afegir un usuari es disposa d'una icona en la part superior dreta per accedir al formulari. Aquesta és l'única manera d'introduir un nou usuari.

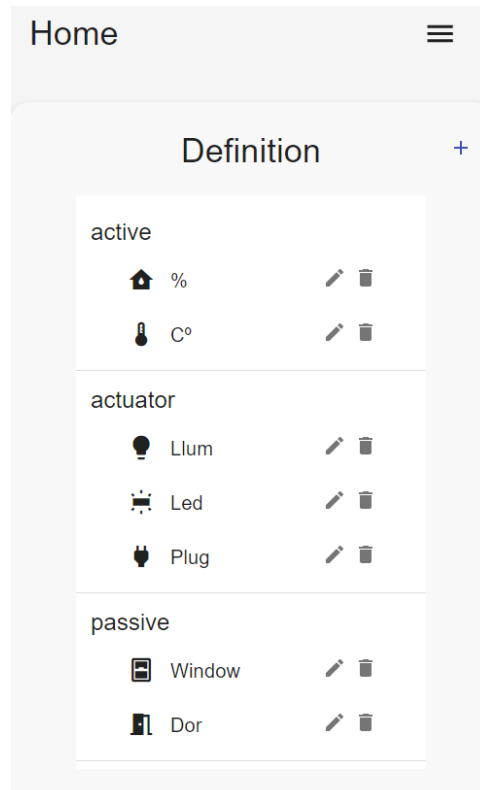


**Figura 47.** Pantalla de gestió dels usuaris.

## Gestió de definicions i habitacions

Tant les definicions com les habitacions són representades en forma de llista on les opcions de gestió de cada una es troben en forma d'icona a la seva dreta.

Les definicions són agrupades per l'especialització a la qual pertanyen. Per a la seva creació se selecciona l'especialització, el nom o unitats que representen i la icona que el representa utilitzant les icones que aporta Google Fonts.



**Figura 48.** Pantalla per gestionar les definicions.

## 5.5 SEGURETAT

Com ja s'ha comentat, una de les necessitats més importants en una casa domòtica és la seguretat davant els atacs de persones no autoritzades. Per això en aquesta secció s'han exposat les mesures usades per evitar la intrusió dintre del nostre sistema de persones no desitjades.

En un sistema domòtic és fonamental la seguretat de les comunicacions i les dades que es generen dintre del sistema. La seguretat en les comunicacions s'ha dut a terme utilitzant els últims protocols de xifratge suportats a partir d'un sistema de certificats i autenticacions. Pel que fa al control de les dades que es generen, aquestes s'han emmagatzemat únicament en el servidor i s'ha limitat l'accés evitant l'extracció de dades no permeses.

En primer lloc es veu com s'ha protegit el sistema d'atacs de l'exterior de la xarxa local

i a continuació als atacs des de l'interior la mateixa xarxa local.

## Xarxa exterior

Com es pot veure en la Figura 11. Arquitectura de la xarxa, la xarxa disposa de dos routers. Per a limitar l'accés des de l'exterior en el router extern s'ha implementat un Firewall on únicament s'han obert dos ports, el port 80 (HTTP) i el port 443 (HTTPS) per tal de poder accedir a la pàgina web. Des d'aquest router, es redirigiran totes les peticions a aquests dos ports cap al servidor.

Un cop la petició accedeix al servidor, s'ha utilitzat el Firewall **UFW** per gestionar-les. En aquest cas el servidor acceptarà únicament els ports 80 (HTTP) i 443 (HTTPS) per a les peticions que provenen de l'exterior. Un cop arriba la petició de la pàgina web és tractada pel servei d'Apache, aquest servei s'encarrega de redirigir totes les peticions del port 80 al 443, d'encriptar totes les comunicacions amb un certificat creat per let's encrypt i d'utilitzar un Proxy pass per a redirigir les peticions cap al port 5000 o 5001 si es tracta d'una petició dirigida al WebSocket de l'API.

A més, si volem augmentar la seguretat, les llibreries Flask i Socket.io permeten aplicar una encriptació addicional utilitzant la llibreria WSGI, la qual permet una sincronització amb Apache.

Per últim, quant a l'accés des de l'exterior de la xarxa local, el servidor també controla el nombre d'intents que una IP realitza per iniciar sessió mitjançant el framework **Fail2Ban** per evitar un atac de força bruta contra l'API per a descobrir les credencials d'un usuari.

## Xarxa local

Al parlar de la xarxa local, el més important és assegurar les comunicacions amb els dispositius IoT per tal d'evitar que un intrús suplanti la identitat d'un dispositiu o del mateix broker, per això s'han creat un conjunt de certificats per a la seva autenticació i encriptació. Les funcions de cada certificat són els següents:

- **CA.crt i CA.key**  
Aquests fitxers fan la funció d'una autoritat de certificació on aquesta determina si els certificats del client o del servidor són autèntics. A més és l'encarregada d'emetre nous certificats, sent aquest l'origen dels certificats dels clients i del servidor.
- **Server.crt i Server.key**  
Els certificats del servidor s'utilitzen per encriptar els missatges abans de ser enviats i de desencriptar els que provenen del client. En encriptar els paquets és generar una marca perquè el client validi que el missatge prové del servidor.
- **Client.cst i Client.key**  
Els certificats del client funcionen de la mateixa manera que els del servidor.

Per a millorar aquesta seguretat cada un dels dispositius tindrà un certificat diferent amb el qual, a més d'encriptar i desencriptar les comunicacions, s'autenticarà davant el servidor. El protocol criptogràfic que s'ha utilitzat per a aquesta comunicació el SSL/TLS1.3 per a poder obtenir la màxima seguretat possible. Per tal d'obligar l'ús de

L'última versió el broker MQTT permet indicar quina és la mínima versió de TLS que accepta i un del conjunt de xifratges, d'aquesta manera s'obliga que tot client de MQTT utilitzi l'última versió de TLS amb un sistema de xifratge determinat per a connectar-se.

En el cas del Firewall UFW aquest acceptarà el port 8883 (MQTT) únicament dels dispositius que es troben dintre de la xarxa interna, es a dir, la mantinguda per el router intern. En quan a la totalitat de la xarxa local s'acceptaran els ports 443 (HTTPS), 80 (HTTP) i 22 (SSH). Per últim i únicament de forma local es podrà accedir a la api a traves dels ports 5000 (API i socket de dades) i 5001 (Socket enfocat a les càmeres) on aquest ports son redirigits de forma local per Apache a partir del port 443.

Dintre de la xarxa local, Fail2Ban controla els intents d'inici de sessió sobre el servei de SSH evitant un atac des de l'interior de la xarxa local.

### 5.5.1 COMPROVACIÓ

Per a donar per bona la implementació de seguretat s'han utilitzat un conjunt de serveis i aplicacions per a comprovar que les mesures s'han aplicat de forma correcta.

La validació del bon funcionament dels certificats usats per Apache s'han analitzat utilitzant la web [www.ssllabs.com](http://www.ssllabs.com). Una pàgina web gratuïta especialitzada en l'anàlisi dels certificats la qual ens aporta un servei web per a poder observar les capacitats i característiques dels certificats creats per let's encrypt i utilitzats per Apache per encriptar la web i les comunicacions entre l'API i la interfície de l'usuari.

Un software més complex, però molt complet que s'ha utilitzat per a la validació de la seguretat és **Nessus**. Un escàner de vulnerabilitats generalitzat que es pot instal·lar en qualsevol sistema operatiu. Aquest servei s'utilitza tant per a l'escaneig de la xarxa local com per a trobar vulnerabilitats des de l'exterior de la xarxa.

Per a l'escaneig local, Nessus aporta un sistema d'escaneig per a descobrir tots els host i els ports amb els serveis que aquests mantenen i possibles vulnerabilitats dels serveis.

També aporta un escaneig centrat en un sol host enfocat a extreure tota la informació d'aquest per a trobar possibles errors i vulnerabilitat a partir dels paquets que li arriben i serveis del que manté.

Quant a l'escaneig des de l'exterior de la xarxa local únicament fa falta executar un anàlisi sobre la pàgina web, ja que és l'únic punt d'entrada. Per aquest tipus d'anàlisis Nessus disposa d'un test centrat en aplicacions web on recopila tota la informació que pot extreure d'una web i retorna un document en forma d'HTML o Làtex exposant tot el que ha trobat, des de missatges d'informació amb tot el que ha analitzat fins als errors juntament amb tota la informació relacionada.

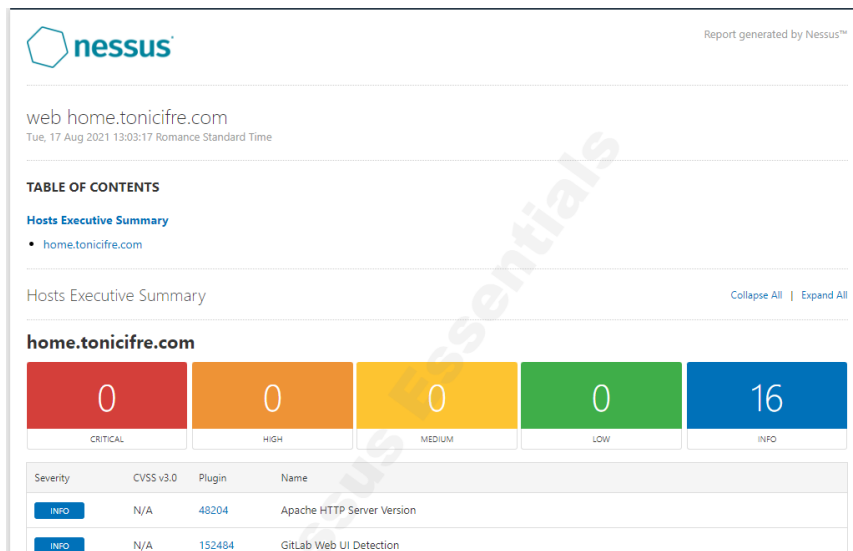


Figura 49. Escaneig de seguretat amb nessus.

En la figura anterior s'observa l'anàlisi sobre la pàgina web des de l'exterior de la xarxa. Aquest escàner i els altres s'han utilitzat per observar els possibles errors de seguretat dintre de la web i actualitzar aquest fins a aconseguir una execució sense errors.

Per últim i com a forma ràpida i senzilla de comprovar que tant MQTT com Apache estan encriptant els paquets de la forma correcta s'ha executat un anàlisi de la xarxa amb el software **Wireshark**, un analitzador de paquets gratuït i de codi obert. A partir de les captures que realitza s'ha determinat si s'estan encriptant els paquets de forma correcta i quina versió de TLS s'utilitza per a la seva encriptació.

Time	Source	Destination	Protocol	Length	Info
1 0.0000...	192.168.158.98	192.168.3.3	TCP	74	48769 → 8883 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSva...
2 0.0353...	192.168.3.3	192.168.158.98	TCP	74	8883 → 48769 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK...
3 0.0353...	192.168.158.98	192.168.3.3	TCP	66	48769 → 8883 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3570385947 TS...
4 0.0356...	192.168.158.98	192.168.3.3	TLSv1.3	583	Client Hello
5 0.0474...	192.168.3.3	192.168.158.98	TCP	66	8883 → 48769 [ACK] Seq=1 Ack=518 Win=64768 Len=0 TSval=1025079045 ...
6 0.0624...	192.168.3.3	192.168.158.98	TLSv1.3	15...	Server Hello, Change Cipher Spec, Application Data, Application Da...
7 0.0624...	192.168.158.98	192.168.3.3	TCP	66	48769 → 8883 [ACK] Seq=518 Ack=1449 Win=64128 Len=0 TSval=35703859...
8 0.0626...	192.168.3.3	192.168.158.98	TLSv1.3	10...	Application Data, Application Data, Application Data
9 0.0627...	192.168.158.98	192.168.3.3	TCP	66	48769 → 8883 [ACK] Seq=518 Ack=2453 Win=63232 Len=0 TSval=35703859...
10 0.0648...	192.168.158.98	192.168.3.3	TLSv1.3	15...	Change Cipher Spec
11 0.0648...	192.168.158.98	192.168.3.3	TLSv1.3	848	Application Data, Application Data, Application Data
12 0.0708...	192.168.3.3	192.168.158.98	TCP	66	8883 → 48769 [ACK] Seq=2453 Ack=1966 Win=64128 Len=0 TSval=1025079...
13 0.0709...	192.168.3.3	192.168.158.98	TCP	66	8883 → 48769 [ACK] Seq=2453 Ack=2748 Win=64128 Len=0 TSval=1025079...
14 0.0709...	192.168.158.98	192.168.3.3	TLSv1.3	129	Application Data
15 0.0761...	192.168.3.3	192.168.158.98	TLSv1.3	12...	Application Data
16 0.0761...	192.168.158.98	192.168.3.3	TCP	66	48769 → 8883 [ACK] Seq=2811 Ack=3588 Win=64128 Len=0 TSval=3570385...
17 0.0761...	192.168.3.3	192.168.158.98	TCP	66	8883 → 48769 [ACK] Seq=3588 Ack=2811 Win=64128 Len=0 TSval=1025079...
18 0.0811...	192.168.3.3	192.168.158.98	TLSv1.3	12...	Application Data, Application Data
19 0.0811...	192.168.158.98	192.168.3.3	TCP	66	48769 → 8883 [ACK] Seq=2811 Ack=4749 Win=64128 Len=0 TSval=3570385...
20 0.7964...	192.168.158.98	192.168.3.3	TCP	15...	48769 → 8883 [ACK] Seq=2811 Ack=4749 Win=64128 Len=1448 TSval=3570...
21 0.7965...	192.168.158.98	192.168.3.3	TCP	15...	48769 → 8883 [PSH, ACK] Seq=4259 Ack=4749 Win=64128 Len=1448 TSval...
22 0.7966...	192.168.158.98	192.168.3.3	TCP	15...	48769 → 8883 [ACK] Seq=5707 Ack=4749 Win=64128 Len=1448 TSval=3570...

Figura 50. Captures dels paquets MQTT amb WireShark.

En aquesta Figura 50 s'observen les captures dels paquets a l'hora de la connexió d'un dispositiu amb el broker MQTT. En la columna del protocol, es pot veure com el client MQTT (192.168.158.98) sol·licita una connexió al broker (192.168.3.3), un cop el servidor respon, envia una llista amb els cipher que accepta per a connectar-se, és a dir, el sistema de xifratge. Si el client accepta algun dels cipher proposat pel broker, s'establirà la connexió.

## 5.6 ESTUDI ECONÒMIC

Per a obtenir el cost econòmic del projecte s'ha calculat el cost de tot el material utilitzat i les hores de treball per a la implementació del sistema.

El cost del hardware s'ha basat en la compra de tots els materials a través de la web AliExpress. Aquesta web és una de les pàgines amb més varietat de dispositius electrònics i amb els preus més baixos.

- Raspberry Pi 4 4GB RAM amb carcassa, dissipadors de temperatura i adaptador de font d'alimentació:
  - Valor: 70 €
  - Consum elèctric: 5.5W
- ESP-01S
  - Valor: 0.75 €
  - Consum elèctric:  $70\text{mA} \cdot 3.3\text{V} = 0.23\text{ W}$   
El consum dels ~70mA es en un estat normal, si es troba en repòs el consum és menor.
  - Quantitat: 10
- Mòdul relé:
  - Valor: 1 €
  - Consum elèctric: 0.3 W com a màxim
  - Quantitat: 5
- Mòdul de font d'alimentació de transformador reductor a 3 V i 5 V:
  - Valor: 1.5 €
  - Consum elèctric: eficiència del 85%  
El consum elèctric depèn de l'eficiència del transformador i dels watts que els dispositius necessiten, es a dir que si un conjunt de dispositius connectats a l'hora consumeixen 3 W, la font d'alimentació perdrà el 15%, el que es tradueix en uns 0.45 W.
  - Quantitat: 5
- Mòdul de prova de corrent PZEM-004T:
  - Valor: 4€
  - Consum elèctric: una mitja de 0.75 W
  - Quantitat: 1
- Mòdul ESP32-CAM:
  - Valor: 4.2 €
  - Consum elèctric: 0.9 W – 1.25 W  
Aquest consum depèn de si el dispositiu es troba en mode de gravació i amb el flash encès.
  - Quantitat: 3
- Mòdul de Sensor de temperatura i humitat DHT11:
  - Valor: 1 €
  - Consum elèctric: 0.1 W  
Aquest consum depèn de la freqüència entre mostres.
  - Quantitat: 2
- Mòdul de Sensor PIR:



- Valor: 1 €
- Consum elèctric: <0.1 W  
Aquest consum depèn de la freqüència entre mostres.
- Quantitat: 1
  
- Mòdul de Sensor d'obertura:
  - Valor: 0.5 €
  - Consum: irrellevant, ja que el seu consum és únicament del controlador.
  - Quantitat: 1
  
- Mòdul de Sensor de qualitat de l'aire MQ135:
  - Valor: 1 €
  - Consum elèctric: 0.5 W  
Aquest consum depèn de la freqüència entre mostres.
  - Quantitat: 1

El cost total de la inversió inicial per a l'adquisició del hardware en el sistema implementat és de **109.1€**.

El consum generat al dia és de 13.75 W, creant un cost mensual de:

$$13.75 \text{ W} \times \frac{24\text{h}}{\text{dia}} = 0.33 \text{ kWh} \times \frac{\sim 0.20\text{€}}{\text{kWh}} * 30 \text{ dies} = \sim \mathbf{1.98 \text{ €}}$$

Totes aquestes dades variaran en funció del servidor utilitzat, del conjunt de dispositius IoT implementats i el cost de l'electricitat.

Quant al cost dels recursos humans, s'ha realitzat una aproximació del temps utilitzat per a la creació del projecte i s'ha extret una mitja del que cobra un programador junior a Barcelona.

$$500 \text{ hores} \times 15 \text{ €} = \mathbf{7500 \text{ €}}$$

Si la creació del projecte l'hagués realitzat una empresa, el cost dels recursos humans per a la creació del sistema domòtic seria d'uns 7.500 €.

## 6. TESTING

Per a comprovar el bon funcionament i la càrrega màxima suportada pel sistema s'han realitzat un conjunt d'anàlisis per a determinar si el sistema suportarà tota la càrrega necessària en el seu funcionament diari. A partir de la càrrega màxima suportada en un interval de temps, s'ha determinat l'escala màxima en què el sistema pot créixer sense un augment del hardware.

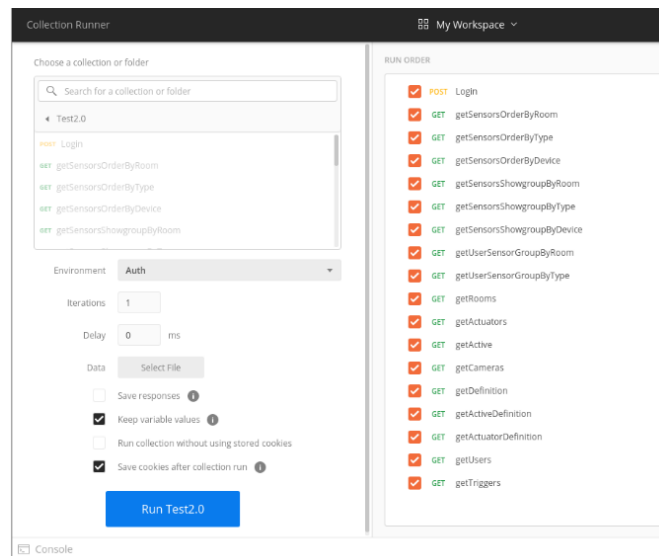
A continuació s'han analitzat els punts del sistema que han de suportar més càrrega, l'API i el sistema MQTT.

Les proves s'han realitzat en les pitjors condicions de connectivitat, és a dir, tant el servidor, el client i els dispositius IoT s'han connectat a través de la xarxa WiFi. A més les proves de la interfície d'usuari s'han realitzat des de l'exterior de la xarxa local. Per últim també s'ha de remarcar que el servidor, el FrontEnd i el BackEnd estan en funcionament durant les proves de rendiment, és a dir que no s'han provat els mòduls individualment, sinó amb tots els serveis actius per a recrear les condicions d'un cas real.

### 6.1 API

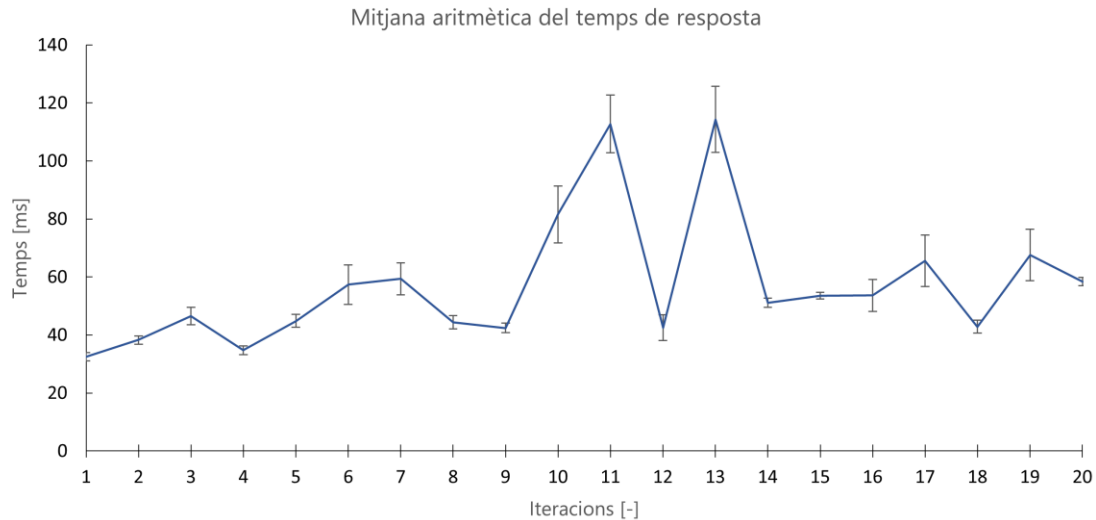
Pel que fa a l'API, aquesta respon a un conjunt de peticions HTTP i manté els WebSocket de les dades i de les càmeres.

La comprovació de la velocitat en què respon a les peticions HTTP s'han realitzat utilitzant el software Postman, a través d'aquest software s'ha analitzat cada un dels EndPoints que manté l'API de forma recursiva per fer una prova de càrrega i veure com varia el temps de resposta en funció del nombre de peticions rebudes.



**Figura 51.** Col·lecció de Postman per analitzar la API.

En la Figura 51 es mostra el sistema de col·leccions que Postman aporta per a testear un conjunt de Endpoints. A partir de l'execució de la col·lecció mostrada amb un nombre de 20 iteracions s'ha obtingut el següent resultat.



**Figura 52.** Mitjana aritmètica del temps de resposta en el test de la API.

En la gràfica de la Figura 52 s'observa com el temps mitjà del conjunt de peticions augmenta en les rondes 11 i 13, però inclús el temps més elevat obtingut és considerar un bon resultat. La prova s'ha realitzat des de l'exterior de la xarxa local i amb una base de dades sobredimensionada específicament per a realitzar les proves de càrrega.

Dintre de la gràfica s'ha calculat l'interval de confiança del conjunt de peticions. L'interval representa la diferència de temps que pot haver-hi entre les diferents peticions d'una ronda.

Si parlem del WebSocket de les càmeres, el problema no és causat per la banda de l'API sinó que la limitació prové del costat de la interfície de l'usuari. S'ha ajustat aquest socket per enviar únicament 60 FPS de forma predeterminada cap a la pàgina web. La limitació es deu al fet que un navegador típic consumeix molts de recursos a l'hora de manejar la rebuda i tractament d'una quantitat elevada de fotogrames. Si s'utilitza un ordinador potent, aquest suporta una major quantitat de fotogrames, però el problema ve donat quan s'utilitza un ordinador vell o un telèfon mòbil.

Per a permetre la utilització de la pàgina web des de qualsevol dispositiu s'ha optat per aplicar aquesta reducció de fotogrames per segon enviades cap al navegador, però aquest valor es pot modificar per l'usuari en funció de les seves preferències.

La limitació dels fotogrames no provoca cap problema per a la visualització de les càmeres, suposant que s'introdueixen 15 càmeres dintre del sistema domòtic, cada una d'aquestes es visualitzarà a 4 FPS. Tractant-se d'una càmera de seguretat és més que suficient. A més el servidor MQTT sí que és capaç de tractar un conjunt més elevat de fotogrames.

## 6.2 MQTT

El sistema MQTT engloba tant una part del BackEnd com el servei que controla el broker MQTT. Per a comprovar la càrrega que suporta s'han recreat un conjunt de 30 càmeres, ja que és el dispositiu IoT que genera un major consum de recursos en diferència dintre del sistema domòtic.

S'han realitzat 2 proves inserint 30 càmeres dintre del sistema MQTT. La primera

prova s'ha dut a terme sense cap usuari connectat al socket de les càmeres per evitar que l'API reenvii les imatges cap al socket. En la segona prova sí que s'ha introduït un usuari al socket, fent que reenvii les 60 imatges per segon cap al socket de l'usuari generant un major consum.

Per a trobar el límit de càrrega que suporta el sistema s'han augmentat el nombre de fotogrames per segon de forma progressiva. Els fotogrames per segon de cada càmera s'han augmentat de 5 en 5 fins a trobar el màxim suportat. Cada cop que s'augmenta el nombre de fotogrames s'ha esperat 15 minuts per validar que el servidor suporta la càrrega de forma estable.

Les 30 càmeres es connecten com a dispositius individuals al broker, un cop connectades començaran a enviar les imatges a una freqüència constant. Per evitar que l'ordinador encarregar a simular les càmeres generi alguna limitació a l'hora de realitzar les proves s'ha dividit aquesta feina entre dos, on cada ordinador simula 15 càmeres. D'aquesta manera s'aconsegueix disminuir la possibilitat que l'error sigui generat per l'ordinador de proves.

Per a controlar el consum dels recursos del servidor s'ha utilitzat el paquet **htop**, aquest paquet mostra a través del terminal el consum dels recursos que generen cada un dels serveis actius.

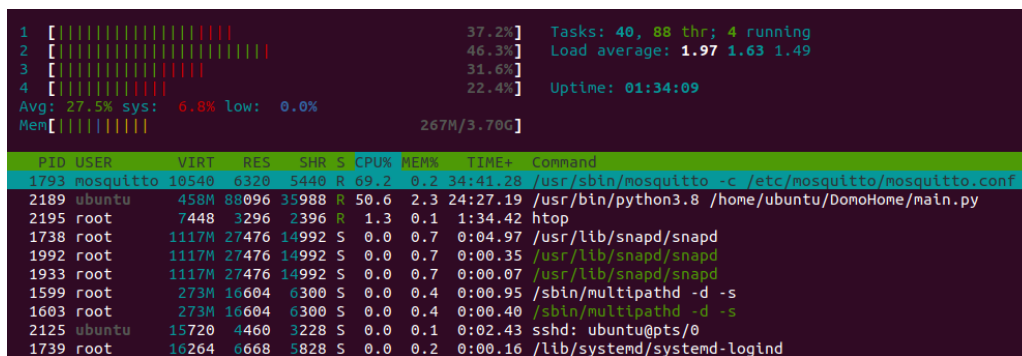
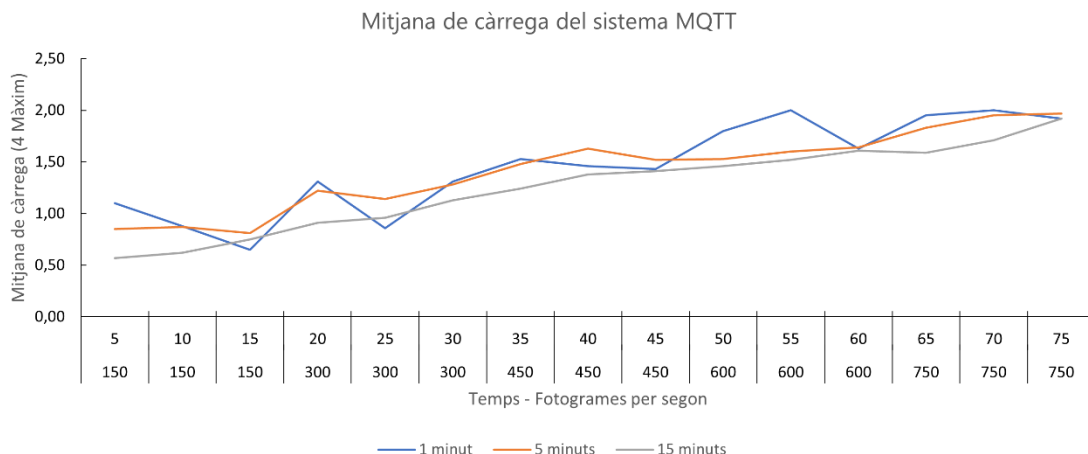


Figura 53. Visualització del consum dels recursos amb htop.

La Figura 53 és una captura del resultat que retorna el paquet htop, a la part superior es mostra una visualització general del consum dels recursos i en la part inferior es desglossa aquest consum per a cada servei en execució. La memòria RAM s'ha considerat irrellevant per l'anàlisi, ja que mai s'ha observat un consum major de 300 MB. Per a la realització d'aquestes proves s'ha observat el "load average", Aquest valor representa una mitjana del consum general dels d'últims 1, 5 i 15 minuts.

El límit del consum depèn del nombre de nuclis del processador, en aquest cas la Raspberry Pi 4 disposa de 4 nuclis, és a dir, que el valor màxim de la mitjana de la càrrega de la CPU és de 4.

La primera prova, on els fotogrames no es reenvien cap a la pàgina web, el servidor ha suportat 25 FPS de cada una de les càmeres enviant un total de 750 FPS amb un consum del 50% dels recursos disponibles de la CPU. Dintre d'aquest consum també s'inclouen els recursos dels altres serveis.

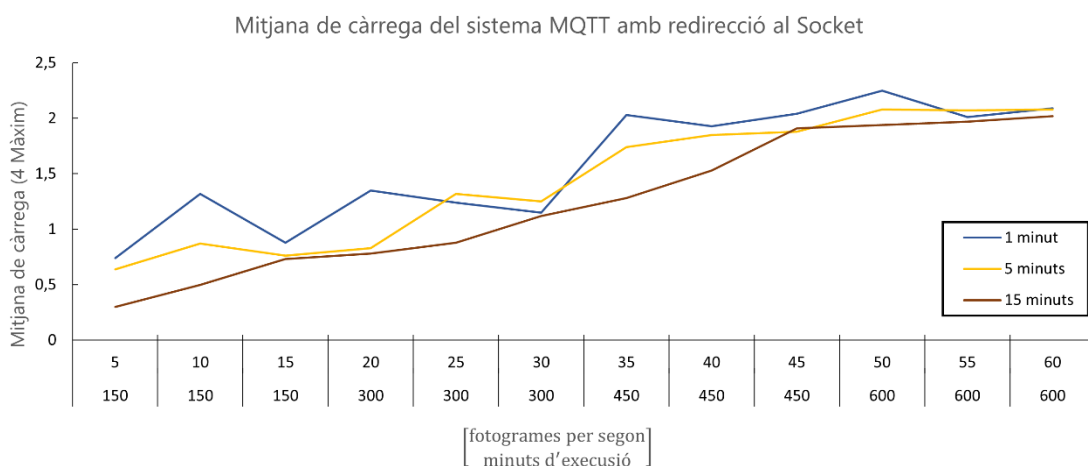


**Figura 54.** Gràfica dels recursos utilitzant en funció dels fotogrames per segon sense reenviament.

La limitació dels fotogrames per segon suportats s'ha trobat en enviar 30 fotogrames per càmera. Però aquesta limitació no és causada pel consum exerciu de la CPU ni de la RAM, sinó per una saturació de la connexió WiFi. No s'ha pogut trobar on es genera aquesta limitació, si és per culpa del servei MQTT, del BackEnd o de si es genera en el client MQTT que envia les imatges cap al servidor.

L'error generat dintre del sistema domòtic en arribar al màxim de fotogrames ha estat la desconexió i connexió d'algunes càmeres de forma aleatòria. Aquest error no bloqueja el funcionament del sistema domòtic, però genera una pèrdua d'imatges.

En la segona prova s'ha generat un consum més elevat per part del BackEnd perquè ha de reenviar les dades cap al WebSocket. El màxim que s'ha aconseguit sense arribar a la saturació han estat els 20 FPS. A partir dels 25 les càmeres perdre l'estabilitat de la connexió amb el servidor MQTT a causa de la saturació de la xarxa igual que en el cas anterior.



**Figura 55.** Gràfica dels recursos utilitzant en funció dels fotogrames per segon amb reenviament.

## 7. DEPLOYMENT DEL SISTEMA

Abans de procedir a la instal·lació i configuració del sistema, si es desitja obrir l'accés a la interfície de l'usuari des de l'exterior, s'ha de crear un domini amb el qual s'enllaçarà la web. És recomanable que el proveïdor del domini disposi del servei de DDNS.

Per instal·lar i configurar els serveis necessaris del servidor s'ha creat un script per automatitzar i facilitar el procediment. L'Script sol·licita els següents valors per a ser executat i configurar de forma adequada els serveis:

- Apache:
  - Adreça URL de la pàgina web.
  - Direcció del directori on s'ubica la web.
- MQTT:
  - Paràmetres per crear els certificats auto signats
    - IP estàtica del servidor
    - Nom de l'organització
  - Usuari i contrasenya per a iniciar sessió.
- DDClient:
  - Usuari del DNS
  - Contrasenya del DNS
  - Adreça URL de la pàgina web.

Un cop s'ha executat l'script els serveis necessaris per al funcionament del FrontEnd i BackEnd s'hauran instal·lat i configurat automàticament.

### 7.1 FRONTEND

Pel FrontEnd s'ha penjat la compilació del codi al GitHub on amb la utilització del paquet git s'ha descarregat dintre del servidor i col·locat els arxius obtinguts dintre del directori `"/var/www/"`.

Per a comprovar el funcionament únicament és necessari accedir a l'URL amb què s'ha configurat Apache i comprovar que retorna la web.

Si es desitja modificar de la pàgina web és necessari obtenir el codi de la web i compilar un cop s'han aplicat les modificacions.

### 7.2 BACKEND

En el primer pas s'han instal·lat totes les llibreries de Python necessàries per a l'execució de la plataforma. Aquestes dependències juntament amb la versió que s'ha utilitzat es troben dintre del fitxer `"requirements.txt"` a partir del qual es poden instal·lar amb una sola comanda al terminal.

Un cop instal·lades les dependències s'ha creat un servei per fer que s'executi de forma automàtica quan el servidor s'inicia i en cas d'error es torni a iniciar. El fitxer de configuració del servei s'ha introduït dintre de la carpeta `"/etc/systemd/System/<nom-del-servei>.service"`.

## CONCLUSIONS

En els darrers 5 anys s'ha observat que la inclusió de les tecnologies en l'àmbit familiar ha incrementat, fins al punt de normalitzar la implementació de sistemes domòtics. Això i tot, el desenvolupament d'aquests dispositius i sistemes per mans de les grans corporacions suposa deixar a l'usuari vulnerable a perdre de la seva privadesa. Sabent aquest creixent fenomen, l'objectiu general d'aquest projecte és l'anàlisi, el desenvolupament i el control d'un sistema per a manejar i monitorar un conjunt de dispositius IoT domèstics. Un cop implementat, es pot afirmar que s'ha aconseguit dur a terme un sistema domòtic òptim, que manté el control dels dispositius IoT i permet a l'usuari interactuar amb aquests. També s'ha de remarcar que s'ha aconseguit implementar un sistema de seguretat per encriptar cada una de les comunicacions que es generen dintre del sistema.

Comparant el projecte amb les opcions que es troben en el mercat s'ha aconseguit un control total sobre les dades personals recopilades en el sistema local, a més d'una major varietat de dispositius IoT en comparació al que accepten els sistemes típics del mercat.

Com s'havia explicat prèviament al desenvolupament de la metodologia i la creació del sistema, en un futur si un usuari vol implementar-ho requerirà uns mínims coneixements informàtics i electrònics. Fa unes dècades aquesta problemàtica hauria sigut decisiva per descartar aquesta metodologia. Això i tot, avui en dia, les noves generacions tenen més accés a aquests coneixements, molts dels quals han impregnat la cultura general. Això fa possible que aquest sistema estigui a l'abast d'un grossor de la població representativament més elevat. Addicionalment, si algun usuari potencial sense formació prèvia en informàtica desitgés implementar el sistema ho podria dur a terme gràcies a l'script que s'ha creat per a facilitar la instal·lació i a través d'una guia d'instal·lació. El que no podria realitzar és alguna modificació important del sistema ni el control de possibles error.

La creació d'un projecte obert i lliure sobre un tema que encara es troba en fase "early adopter", no sol crear cap efecte sobre el mercat de les empreses institucionalitzades. El nínxol de mercat que ataca un projecte d'aquest tipus és molt reduït i principalment està format per persones apassionades sobre les innovacions en l'àmbit del projecte. El mercat al qual ataquen les empreses és al grup que els agrada la tecnologia, però no disposen del temps o el coneixement necessaris per arribar un poc més enllà. Actualment, aquest projecte i els projectes més coneguts que es troben en el mercat no estan dirigits sobre el mateix gruix de la població.

Si aquest estudi es financés amb la intenció d'entrar en el mercat, s'eliminarien els requeriments que els usuaris necessiten per a la seva implementació creant un conjunt de dispositius IoT premuntats juntament amb un servidor central el qual no requereixi cap instal·lació pel seu funcionament. Amb aquestes modificacions s'atacaria a un conjunt major de la població fent competència a les empreses ja establertes.

## BIBLIOGRAFÍA

Apache. [En línia] <https://www.apache.org/>.

ArduinoJson. [En línia] <https://arduinojson.org/v6/doc/>.

Certbot. [En línia] <https://certbot.eff.org/docs/>.

Dan Abramov. *Redux*. [En línia] <https://redux.js.org/>.

DevExpress. *Dev Extreme React*. [En línia] <https://devexpress.github.io/devextreme-reactive/react/chart/>.

Eclipse. *Paho.MQTT*. [En línia] <https://www.eclipse.org/paho/index.php?page=documentation.php>.

Eclipse. *Mosquitto*. [En línia] <https://mosquitto.org/>.

Gevent. [En línia] <https://pypi.org/project/gevent/>.

GitHub. *WiFi Manager*. [En línia] <https://github.com/tzapu/WiFiManager>.

GitHub. *Esp32 Camera*. [En línia] <https://github.com/espressif/esp32-camera>.

GitHub. *DHT sensor library*. [En línia] <https://github.com/adafruit/DHT-sensor-library>.

Hossein Dehnokhalaji. *Notistack*. [En línia] <https://iamhosseindhv.com/notistack/api>.

John Jakob i Matt Zabriskie. *Axios*. [En línia] <https://axios-http.com/>.

Knolleary. *PubSubClient*. [En línia] <https://pubsubclient.knolleary.net/>.

Let's Encrypt. [En línia] <https://letsencrypt.org/docs/>.

Material UI. *MUI*. [En línia] <https://mui.com/>.

Numpy. [En línia] <https://numpy.org/doc/stable/>.

OpenCV. [En línia] <https://docs.opencv.org/master/>.

Pallets Projects. *Flask*. [En línia] <https://flask.palletsprojects.com/en/2.0.x/>.

Pallets Projects. *Werkzeug*. [En línia] <https://werkzeug.palletsprojects.com/en/2.0.x/>.

PlatformIO. [En línia] <https://docs.platformio.org/en/latest/>.

Postman Inc. *Postman*. [En línia] <https://www.postman.com/>.

Python. *Logging*. [En línia] <https://docs.python.org/3/library/logging.html>.

Python. *SQLite3*. [En línia] <https://docs.python.org/3/library/sqlite3.html>.

Python. *SSL*. [En línia] <https://docs.python.org/3/library/ssl.html>.



React. [En línea] <https://reactjs.org/>.

Read the Docs. *Filesystem*. [En línea] <https://arduino-esp8266.readthedocs.io/en/latest/>.

Read the Docs. *Bearssl client secure*. [En línea] <https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html#bearssl-client-secure-and-server-secure>.

Read the Docs. *Schedule*. [En línea] <https://schedule.readthedocs.io/en/stable/>.

Read the Docs. *Flask Httpauth*. [En línea] <https://flask-httpauth.readthedocs.io/en/latest/>.

Read the Docs. *Slask Socketio*. [En línea] <https://flask-socketio.readthedocs.io/en/latest/>.

Read the Docs. *PyJWT*. [En línea] <https://pyjwt.readthedocs.io/en/stable/>.

Read The Docs. <https://flask-cors.readthedocs.io/en/latest/>. [En línea] <https://flask-cors.readthedocs.io/en/latest/>.

*Smart home and appliances: State of the art*. **Serrenho, T. y Bertoldi, P. 2019**. s.l. : JRC Technical Reports, 2019, JRC technical reports, págs. 28-38.

Socket.IO. [En línea] <https://socket.io/docs/v4>.

Tenable. *Nessus*. [En línea] <https://es-la.tenable.com/products/nessus>.

Ubuntu. *DynamicDNS*. [En línea] <https://help.ubuntu.com/community/DynamicDNS>.

Ubuntu. *Fail2Ban*. [En línea] <https://help.ubuntu.com/community/Fail2Ban>.

Ubuntu. *Systemd Service*. [En línea] <http://manpages.ubuntu.com/manpages/cosmic/man5/systemd.service.5.html>.

Ubuntu. *UFW*. [En línea] <https://help.ubuntu.com/community/UFW>.

WireShark. [En línea] <https://www.wireshark.org/docs/>.