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

Antoni Cifre Vicens

Escola Politècnica Superior d'Enginyeria de Vilanova i la Geltrú

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 autentificacions. 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

1. Introducció

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.

2. Formulació del problema i objectius

El sistema complet d'una casa domòtica s'ha dividir en tres apartats diferents:

- **Dispositius IoT:** S'encarrega de la recol·lecció de les dades i interacciona amb l'ambient.
- **Interfície de l'usuari:** Representa els estats de forma visualment atractiva i permet als usuaris interactuar amb la resta del sistema.
- **Plataforma de control:** Unifica la comunicació entre els dispositius IoT i la interfície. A més, llança accions programades o per afecte d'altres accions i controla i persisteix l'estat de tot el sistema.

Aquests tres apartats són necessaris per a la creació d'un sistema domòtic complet.

Els dispositius IoT es limiten als dispositius que poden ser controlats per una single-board computer (SBC). Aquesta limitació ve donada per tal de generar un sistema tancat on s'utilitza un mateix protocol de comunicacions, d'autentificació i encriptació maximitzant així la seguretat i simplicitat del sistema.

En quant 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 però amb els desavantatges del navegador. L'elecció de crear una aplicació mòbil o per ordinador elimina les limitacions del navegador, però únicament es pot utilitzar per a un tipus de dispositiu.

L'aplicació central o plataforma de control és la secció més important, ja que connecta i manté els dispositius IoT i els clients. 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.

L'objectiu general és aconseguir implementar un sistema domòtic d'una casa amb la utilització de tecnologies obertes i amb la priorització 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.

- Els dispositius IoT han d'utilitzar tecnologies obertes on el firmware pugui ser crear i introduir dintre del dispositiu per un control específicament dissenyat pel sistema creat.També 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 d'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 s'ha d'encarregar 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.

3. Estudi tecnològic

S'ha realitzat un estudi tecnològic sobre les tecnologies que es necessiten per a dur a terme el projecte buscant les millors opcions en casa un dels apartats. En l'estudi s'ha abarcat les tecnologies sense fil per a la comunicació, els dispositius IoT, els protocols de dades a utilitzar i el millor servidor per allotjar el projecte.

Les tecnologies sense fil que s'ha elegit per a la comunicació dels dispositius IoT és el wifi [1], ja que aporta una major flexibilitat sobre els dispositius, obtenint una major versatilitat en el seu funcionament i reduint el cost de cada dispositiu.

Els dispositius IoT s'han format utilitzant un microcontrolador i un o diversos mòduls. Els principals microcontroladors escollits són de la família dels ESP8266 [2], un single-board computer (SBC) molt econòmic i amb potència suficient per a controlar un conjunt de mòduls. Els mòduls són el conjunt de sensors i actuadors connectats al controlador amb el qual es pot interactuar amb l'ambient. Per a la implementació del sistema s'han escollit un conjunt de mòduls en funció de les necessitats del client.

Per a la comunicació entre els diferents apartats s'han escollit dos protocols de dades, un per a la comunicació entre els dispositius IoT i la plataforma de control i un altre per a la comunicació entre la plataforma i la interfície de l'usuari. Entre els dispositius IoT i la plataforma s'ha seleccionat el protocol MQTT [3], un protocol de dades molt lleuger i amb una composició bàsica ideal dispositius petits amb una baixa potència de càlcul. MQTT implementa un sistema per a controlar la qualitat del servei (QoS) i permet l'encriptació i autentificació dels paquets amb TLS 1.3.

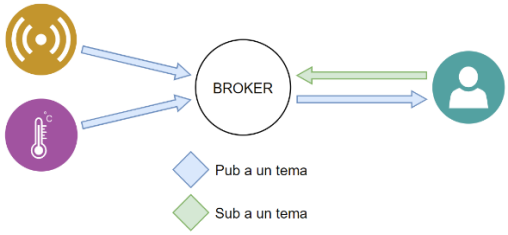

Fig. 1. *Sistema simple de comunicació amb MQTT*

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 usuaris se subscriuen als tòpics per obtenir els missatges i envien els missatges a un tòpic.

La comunicació entre la interfície i la plataforma s'ha elegit el protocol WebSocket [4]. La utilització de MQTT s'ha descartat, ja que el navegador no està optimitzat per a

treballar amb ell. El protocol WebSocket implementa una comunicació full-dúplex mantenint la seguretat de la web i no es veu influït pel nombre d'usuaris connectats a l'hora.

4. Arquitectura

L'arquitectura de la xarxa s'ha dissenyat per a facilitar el control de la seguretat, introduint els sensors en una xarxa interna mentre que el servidor s'ha connectat directament al encaminador extern per acceptar els paquets que provenen de l'exterior.

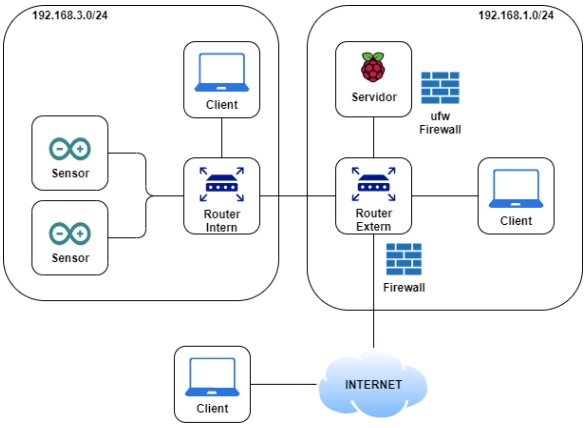

Fig. 2. *Arquitectura de la xarxa*

En la figura anterior 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.

Un cop s'ha dissenyat l'arquitectura de la xarxa s'ha creat el sistema de comunicacions entre els diferents apartats.

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 on únicament s'accepten els paquets dels dispositius que provenen de la xarxa local. La comunicació entre la interfície i el servidor s'ha dissenyat amb els protocols de dades HTTPS i WebSocket. Pel 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.

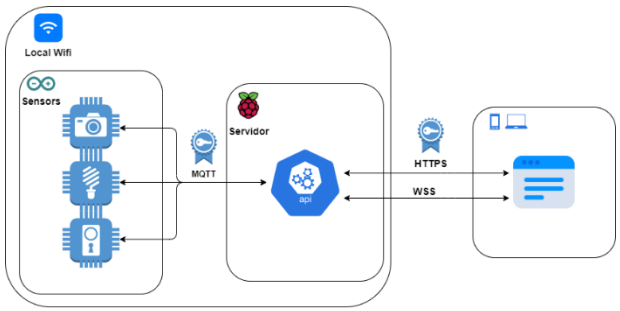

Fig. 3. *Arquitectura de comunicació*

5. 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 es pot accedir als recursos de l'API i ajuden a garantir el bon funcionament del programari incorporat.

Pel sistema MQTT s'ha considerat com EndPoint un tòpic on el dispositiu IoT envia i rep informació de manera establerta. En el cas del WebSocket, aquest EndPoint es defineix com els esdeveniments on es generen les dades.

Per a l'API s'ha utilitzat l'estructura establerta com a API REST [5]. Pel sistema MQTT i WebSocket no existeix un estàndard de comunicació que concordi amb un sistema com el que s'ha creat, per això s'ha generat un sistema d'EndPoint per organitzar les dades enviades a través d'aquests.

6. Servidor

El servidor manté un conjunt de serveis necessaris per al funcionament del sistema, tant per a servir la pàgina web de la interfície com per a mantenir el broker de MQTT.

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

Els serveis que s'han utilitzat i les seves característiques són:

- **Apache** [6] 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.

  Cada una de les peticions que són processades per Apache són encriptades abans de ser enviades cap a l'exterior. Per crear els certificats s'utilitza Let's Encrypt, una nova autoritat de certificació capaç d'emetre certificats signats creuadament per IdentTrust.

- 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 IoT. 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 1.3 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.

- Els proveïdors d'internet no donen una IP estàtica de forma predeterminada, per això s'ha implementat el servei **ddclient** [7]. 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.
- **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.

  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.

7. Dispositius IoT

Per a representar i controlar l'estat dels diferents dispositius IoT s'han creat 4 categories:

- **Càmera**: Captura i envia imatges a uns fotogrames per segon determinat en forma de bytes i analitza les imatges per detectar moviment. 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.
- **Actius**: Recol·lecta dades ambientals de forma activa a una freqüència determinada. 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'historial.
- **Passiu**: Captura esdeveniments de l'ambient generats per alguna causa externa de forma binària. L'usuari pot establir el temps que ha de passar per a tornar l'estat inicial.
- **Actuador**: Es queda a l'espera d'un senyal del sistema per actuar sobre l'ambient. Permet activar un temporitzador per aplicar el canvi d'estat.

Aquestes especialitzacions 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.

Muntatge

Els dispositius IoT s'han programat amb el llenguatge C++ amb un conjunt de llibreries per a realitzar les funcions específiques de cada mòdul.

Per a modificar el firmware dels microcontroladors es necessita un mòdul adaptador per a connectar aquest a l'ordinador i actualitzar el firmware amb el codi de cada dispositiu IoT.

Un cop el microcontrolador s'ha programat, s'ha d'unificar amb el mòdul o el conjunt de mòduls que controlarà. Per a la comunicació entre els mòduls i el microcontrolador s'utilitza els pins tipus GPIO per a transmetre les dades recol·lectades.

En la següent figura s'observa un exemple de connexió entre el mòdul PZEM-004t i un microcontrolador ESP8266 per a monitorar el consum elèctric.

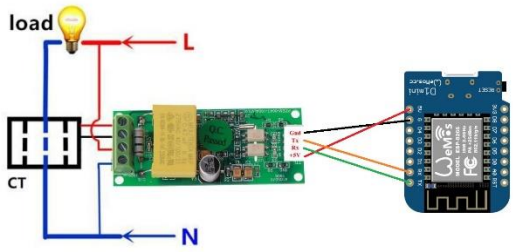

***Fig. 4**. Esquema elèctric entre el microcontrolador ESP8266 i el mòdul PZEM-004T*

Esdeveniments del sistema

A continuació es plasmen els esdeveniments del sistema generals que realitza cada un dels dispositius IoT per a mantenir el seu correcte funcionament.

- **Configuració inicial** - El primer cop que el dispositiu és connectat al corrent 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, les credencials del client MQTT, el nom del dispositiu i una llista amb els noms dels mòduls que conte el dispositiu.
- **Inici de sessió** - Un cop connectat amb el broker MQTT el dispositiu IoT sol·licita a la plataforma de control iniciar sessió perquè aquesta introdueixi el dispositiu dintre del sistema. 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 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. 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 mòduls.

- **Nou estat** - Cada cop que un sensor obté un estat diferent de l'anterior l'envia a través de MQTT cap a l'API.
- **Escolta del sistema**- Els sensors es mantenen a l'espera d'obtenir algun missatge nou que provingui del sistema. Els tòpics que escolten depén de l'especialització de cada sensor. Els missatges generals són: els paràmetres de configuració, el ping, l'acceptació de l'inici de sessió i el restabliment del dispositiu IoT. Els sensors Actuadors també es mantenen a l'escolta del canvi d'estat per part del sistema i l'establiment d'un temporitzador per aplicar el canvi d'estat.

8. BackEnd

El BackEnd manté l'estat i les comunicacions del sistema escoltant i tractant els missatges dels dispositius IoT i usuaris 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 especifica.

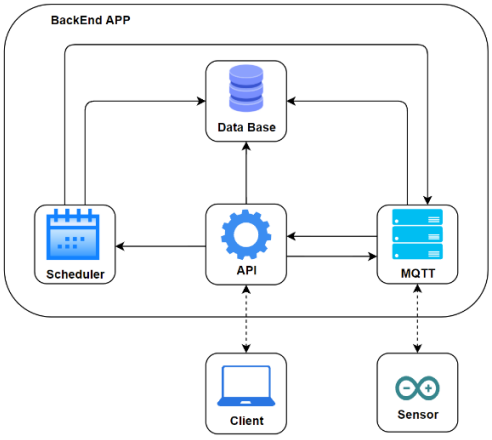

Fig. 5. *Visibilitat dels mòduls del BackEnd*

Base de dades

Per a mantenir la persistència de les dades s'han guardat en una base de dades SQLite [8]. Una base de dades lleugera i fiable mantinguda en un fitxer. El sistema domòtic local no genera un gran volum de dades fent aquest tipus de bases de dades una opció ideal.

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 utilitzant la filosofia d'operacions CRUD (Create, Read, Update, Delete) facilitant la comprensió i el disseny del codi.

API

Els EndPoints del BackEnd s'han creat utilitzant l'estil d'arquitectura REST a partir del framework Flask [9]. L'accés als EndPoints s'han limitat en funció del token JWT enviat pel client i el rol al qual pertany el client que autentificat.

A partir de l'extensió flask_socketio [10] s'ha implementat dos WebSocket per a crear les comunicacions bidireccionals a temps real entre la pàgina web i l'API. El WebSocket principal manté les dades generals dels dispositius IoT mentre que el segon socket únicament envia les imatges que capturen les càmeres per evitar una saturació del socket principal. El WebSocket també controla l'accés a partir del token que l'usuari envia al sol·licitar la connexió. A més s'ha mantingut un recompte dels usuaris connectats per evitar el reenviament de les dades si no hi ha cap usuari escoltant.

MQTT

El mòdul encarregat de controlar les comunicacions sobre MQTT fa ús de la llibreria paho.mqtt [11] per a mantenir el control sobre els missatges dels sensors.

En 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 IoT sol·liciten connectar-se amb la plataforma. En el cas que el BackEnd disposi d'algun sensor prèviament registrat s'inicia l'escolta cap als tòpics de l'estat del sensor.

Cada especialització es tracta de forma diferent, ja que el seu estat representa un valor distint.

Els **actuadors** i els **passius** envien un valor que generalment indica si el mòdul està encès o aturant. Quan es rep un nou estat es redirigeix cap al WebSocket per a la seva previsualització. A continuació s'actualitza l'estat del sensor en la base de dades i s'introdueix dintre de l'historial amb l'hora de la modificació. Finalment es revisa si existeix algun disparador on aquest canvi d'estat compleixi alguna condició.

Quan es rep un valor dels sensors **actius** i la diferencia de temps amb l'última mesura és major a la freqüència de mostreig, es crea una nova instància dintre de l'historial dels sensors Actius. A continuació s'informa a la interfície perquè actualitzi la gràfica del sensor. En el cas que el temps no sigui major a la freqüència, no es guardarà dintre de la base de dades, únicament s'enviarà l'estat a la interfície. Per últim i en els dos casos es comprovarà si existeix algun trigger on aquest canvi desencadeni una acció.

Les **càmeres** envien dades 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 de gravació i comprova si existeix algun trigger que es vegi afectat. Si la càmera està gravant quan rep el missatge, s'actualitzarà el temps de gravació.

Quan es rep un fotograma, el BackEnd comprova si es supera el màxim de 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 a la interfície. A continuació es comprova si s'ha de guardar el fotograma, en cas afirmatiu, s'afegirà el fotograma a l'arxiu de gravació. 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.

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 la plataforma. Per a crear aquesta lògica s'ha utilitzat la llibreria Schedule [12].

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.

9. FrontEnd

Els llenguatges que s'han utilitzat per a la creació completa de la pàgina web són JavaScript, HTML i CSS juntament amb la llibreria de codi obert *React* [13]. A partir de React s'ha creat una pàgina web amb la filosofia SPA (aplicacions d'una sola pàgina) i s'han afegit els components de *Material UI* per a desenvolupar la web baix un mateix estil visual.

L'obtenció de les dades s'ha realitzat via peticions HTTP amb la llibreria Axios [14] conjuntament amb el 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.

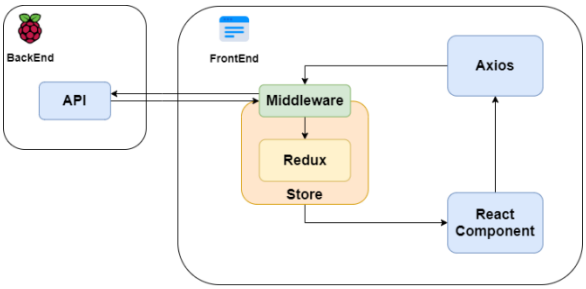

Fig. 6. *Estructura de la comunicació amb l'API*

Quan a través d'un component es realitza una petició utilitzant la llibreria Axios, la petició s'envia al middelware i d'aquest cap a l'API. En rebre la resposta, s'extreu la informació i s'envia al Redux perquè la guardi dintre del seu arbre d'estats. Quan s'actualitza un estat, els components que té visibilitat sobre aquest obté el nou valor actualitzat.

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.

Cada especialització dels sensors es representa com un component React. Quan el component és creat genera una nova escolta cap al EndPoint del socket que envia les dades del sensor en concret. Si el sensor es tracta d'una càmera, aquest crea una connexió amb el socket específic de les càmeres.

Cada component que s'ha utilitzat per a representar una especialització del sensor s'ha creat per a mostrar l'estat del sensor de forma simple i visual a través del qual es pot accedir a les funcions específiques de cada component.

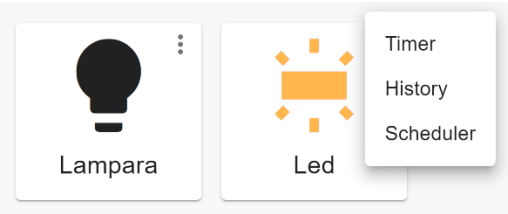

Fig. 7. *Components del sensor actuador*

En la figura anterior s'observa el component que representa als sensors actuadors, on de forma visual s'aprecia si l'actuador es troba actiu o no. Cada targeta disposa d'un mètode per accedir a les funcions específiques de cada especialització. En el component dels actuadors, l'accés a les funcions és a través d'un menú desplegable. De forma general, les opcions s'han col·locat al cantó superior de la dreta del component.

En el cas dels actuadors, l'usuari ha d'interactuar amb l'estat, podent canviar el seu valor de manera simple. Per aquesta interactuació s'ha decidit que el component sigui un botó on al ser clicar es modifica el valor de l'estat.

L'usuari pot modificar i accedir a totes les funcions del sistema des de la interfície, per això a continuació s'ha plasmat cada una de les pantalles junt amb les funcions del sistema que realitzen.

Inici de sessió

L'inici de sessió ataca a l'API amb les credencials introduïdes per a obtenir un token d'autentificació. 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.

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. Els components dels sensors s'agrupen pel factor elegit per l'usuari i dintre de cada agrupació es tornen a agrupar per especialització.

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 introduir el dispositiu.

Depèn de l'especificació de cada sensor, el formulari sol·licitarà els valors necessaris per al seu control.

Control de visibilitats

La pantalla per controlar els sensors visibles permet escollir els sensors que es visualitzen dintre de la pantalla principal. Únicament es podran escollir els sensors als quals l'usuari identificat i té accés.

Per a facilitar la navegació, els sensors s'agruparan en forma d'acordió.

Control dels disparadors

Dintre de la pantalla dels disparadors s'ha implementat una llista dels disparadors existents amb l'opció per eliminar el disparador.

A través d'aquesta pantalla es pot introduir un nou disparador. 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, ja que cada especialització controla un tipus de valor distint.

Gestió dels sensors

Dintre de la pantalla de gestió dels sensors es troben dues seccions diferenciables, 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 pàgina de control de visibilitats. La diferència és que s'ha introduït un desplegable per a seleccionar l'usuari sobre el qual es modificarà l'accés als sensors.

Per a modificar i eliminar els sensors s'han introduït dintre d'una taula on de manera senzilla en poden visualitzar tots els sensors amb els valors de configuració.

Gestió d'usuaris, definicions i habitacions

Les pantalles per a gestionar els usuaris, les definicions i les habitacions s'han creat utilitzar una llista per a la seva representació on les opcions de gestió de cada una es troben en forma d'icona a la seva dreta.

Les accions que es poden realitzar sobre un usuari son: la seva edició, la modificació de la contrasenya i l'eliminació de l'usuari. Sobre les definicions i les habitacions únicament es pot editar o eliminat. L'execució de cada una de les accions utilitza un modal per a la seva realització.

Per afegir un nou objecte es disposa d'una icona en la part superior dreta per accedir al formulari.

10. Seguretat

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 autentificacions. 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 1, 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). Des d'aquest router, es redirigiran totes les peticions únicament al servidor.

Un cop la petició accedeix al servidor, s'ha utilitzat el Firewall UFW per gestionar-les. Les peticions cap a l'API és tractada pel servei d'Apache, aquest servei s'encarrega de redirigit 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.

Si es desitja augmentar la seguretat, les llibreries Flask i Socket.io permeten aplicar una encriptació addicional utilitzant la llibreria WSGI [15], la qual es pot sincronitzar amb Apache.

Per últim, 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.

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 autentificació i encriptació.

Per a millorar aquesta seguretat cada un dels dispositius té un certificat diferent amb el qual, a més d'encriptar i desencriptar les comunicacions, s'autentificarà davant el servidor. El protocol criptogràfic que s'ha utilitzat per a aquesta comunicació el SSL/TLS1.3. Per tal d'obligar l'ús de l'última versió el broker MQTT permet indicar quina és la mínima versió de TLS i el del conjunt de xifratges que accepta.

En el cas del Firewall UFW aquest acceptarà el port 8883 (MQTT) únicament dels dispositius que es troben dintre de la xarxa interna.

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 al servidor.

11. Testing

Per a comprovar el bon funcionament i la carrega màxima suportada pel sistema s'han realitzat un conjunt d'anàlisis per a determinar si el sistema suportarà tota la carrega necessària en el seu funcionament diari.

Les proves s'han realitzat en les pitjors condicions de connectivitat on tant el servidor, el client i els dispositius IoT s'han connectat a través de la xarxa WiFi.

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

API

La comprovació de la velocitat en què respon a les peticions HTTP s'han realitzat utilitzant el software Postman [16] per a atacar de forma recursiva el conjunt de EndPoints que manté l'API.

Els resultats obtinguts són més que acceptables ja totes les respostes han rondat el temps ideal, els 0.1 segons. 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.

En el WebSocket de les càmeres es genera una limitació important. Depèn del dispositiu amb el qual s'accedeix a la pàgina web, aquest no és capaç de processar un nombre

elevat de fotogrames per segon. Per això l'usuari pot establir un màxim de fotogrames per segon que s'enviaran cap a la interfície per evitar un retard en la visualització de les càmeres. El límit recomanable per a funcionar amb tots els dispositius són els 60 fotogrames per segon.

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. La primera prova s'ha dut a terme sense cap usuari connectat al socket de les càmeres per evitar que l'API reenviï les imatges cap al socket. En la segona prova sí que s'ha introduït un usuari al socket.

Per a trobar el límit de càrrega que suporta el sistema s'han augmentat els fotogrames per segon de cada càmera 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 el funcionament.

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% de la CPU.

La limitació s'ha trobat als 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. L'error generat dintre del sistema domòtic en arribar al màxim de fotogrames ha estat la desconnexió i connexió d'algunes càmeres de forma aleatòria.

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 FPS la xarxa es comença a saturar.

12. Conclusions

L'objectiu general d'aquest projecte és l'estudi, desenvolupament i control d'un sistema de comunicacions per tal d'utilitzar 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 on es manté el control dels dispositius IoT i permet a l'usuari interactuar amb aquests correctament. El sistema de comunicacions implementat per a connectat cada un dels apartats funciona de forma fluida i segura, on cada un dels paquets és encriptat.

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 i una major versatilitat en els dispositius IoT.

La implementació del sistema descrit en aquest projecte per part de l'usuari requerirà uns mínims coneixements informàtics i electrònics. Fa unes dècades aquesta problemàtica hauria sigut decisiva per descartar aquesta metodologia. Això i tot, sabent que avui en dia, les noves generacions tenen més accés a aquests coneixements, molts dels quals han impregnat la cultura general. 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.

Referències

[1] T. Serrenho y P. Bertoldi, «Smart home and appliances: State of the art,» *JRC tecnical reports*, pp. 28-38, 2019.

[2] Espressif, «Espressif,» [En línea]. Available: https://www.espressif.com/en/products/socs/esp8266.

[3] Eclipse, «Mosquitto,» [En línea]. Available: https://mosquitto.org/.

[4] Socket.IO. [En línea]. Available: https://socket.io/docs/v4.

[5] OpenAPI, «OpenAPI Specification,» [En línea]. Available: https://spec.openapis.org/oas/v3.0.3.

[6] Apache. [En línea]. Available: https://www.apache.org/.

[7] Ubuntu, «Manpages ddclient,» [En línea]. Available: http://manpages.ubuntu.com/manpages/bionic/man8/ddclient.8.html.

[8] SQLite. [En línea]. Available: https://www.sqlite.org/index.html.

[9] Pallets Project, «Flask,» [En línea]. Available: https://flask.palletsprojects.com/en/2.0.x/.

[10] Read the Docs, «Flask-SocketIO,» [En línea]. Available: https://flask-socketio.readthedocs.io/en/latest/.

[11] Eclipse, «paho,» [En línea]. Available: https://www.eclipse.org/paho/index.php.

[12] Read the Docs, «Schedule,» [En línea]. Available: https://schedule.readthedocs.io/en/stable/.

[13] «React,» [En línea]. Available: https://reactjs.org/.

[14] J. Jakob y M. Zabriskie, «Axios-Http,» [En línea]. Available: https://axios-http.com/.

[15] Pallets Projects, «Werkzeug,» [En línea]. Available: https://werkzeug.palletsprojects.com/en/2.0.x/.

[16] Postman Inc. [En línea]. Available: https://www.postman.com/.