
UOC
Universitat Oberta
de Catalunya

Domotització total d'un habitatge per millorar l'eficiència energètica, el confort i la seguretat

Nom Estudiant: Octavi Pavon Montoliu

Pla d'estudis de l'estudiant: Grau en Enginyeria Informàtica.
Itinerari d'Enginyeria de computadors

Àrea de treball final: 05.663 -Arduino

Nom Consultor/a: Antoni Morell Pérez

Nom Professor/a responsable de l'assignatura: Pere Tuset Peiró

Data Lliurament

07/01/2019

FITXA DEL TREBALL FINAL

Títol del treball:	*Domotització total d'un habitatge per millorar l'eficiència energètica, el confort i la seguretat*
Nom de l'autor:	*Octavi Pavon Montoliu*
Nom del consultor/a:	*Antoni Morell Pérez*
Nom del PRA:	*Pere Tuset Peiró*
Data de lliurament (mm/aaaa):	*01/2019*
Titulació o programa:	*Grau en Enginyeria Informàtica*
Àrea del Treball Final:	*05.663 -Arduino*
Idioma del treball:	*Català*
Paraules clau	*domotic, mqtt, nodemcu*

Resum del Treball (màxim 250 paraules):

El treball busca crear un sistema domòtic de baix cost i altes prestacions per entorns domèstics. Es pretén automatitzar tot els elements possibles d'un habitatge limitant l'adquisició de productes comercials per aconseguir un habitatge autònom, energèticament eficient i segur. Paral·lelament, fer l'adquisició de dades d'una estació meteorològica semi-professional i utilitzar les dades tant per la presa de decisions com per publicar-les en una web pública.

Per aconseguir aquest objectiu es combinen dispositius de control i sensors comercials, en estat original o modificats, així com una sèrie de microcontroladors compatibles amb Arduino que executen un codi compatible amb el protocol MQTT. Aquests microcontroladors estan basats en el xip ESP8266, i tenen connectats diferents tipus de sensors i actuadors.

Tots els elements són gestionats i governats des de dos servidors basats en miniordinadors Raspberry Pi, utilitzant com a plataforma de gestió Home Assistant. En aquests servidors s'executen un conjunt de serveis de suport a nivell de xarxes i seguretat.

El resultat és un ecosistema administrable remotament, que també pot prendre decisions de forma autònoma sense intervenció humana.

Abstract (in English, 250 words or less):

The project objective is to create a home automation system with low cost and high performance for home environments. It tries to automate all the possible elements of a house, limiting the acquisition of commercial products, to obtain a system that be autonomous, energy efficient and secure. Also will do the acquisition of data from a semi-professional meteorological station where the data will be published in a public web page.

To achieve this goal, commercial devices are used in their original state or with firmware modifications, combined with different microcontrollers compatible with Arduino that execute a code that is compatible with the MQTT protocol. These microcontrollers are based on the ESP8266 chip, and this microcontrollers has connected different types of sensors and actuators.

All the elements are managed and governed from two servers based on mini-computers Raspberry Pi, using as Home Assistant as management platform. A set of network and security level support services are run on these servers.

The result is a remotely manageable ecosystem, which can also make decisions autonomously without human intervention.

Índex

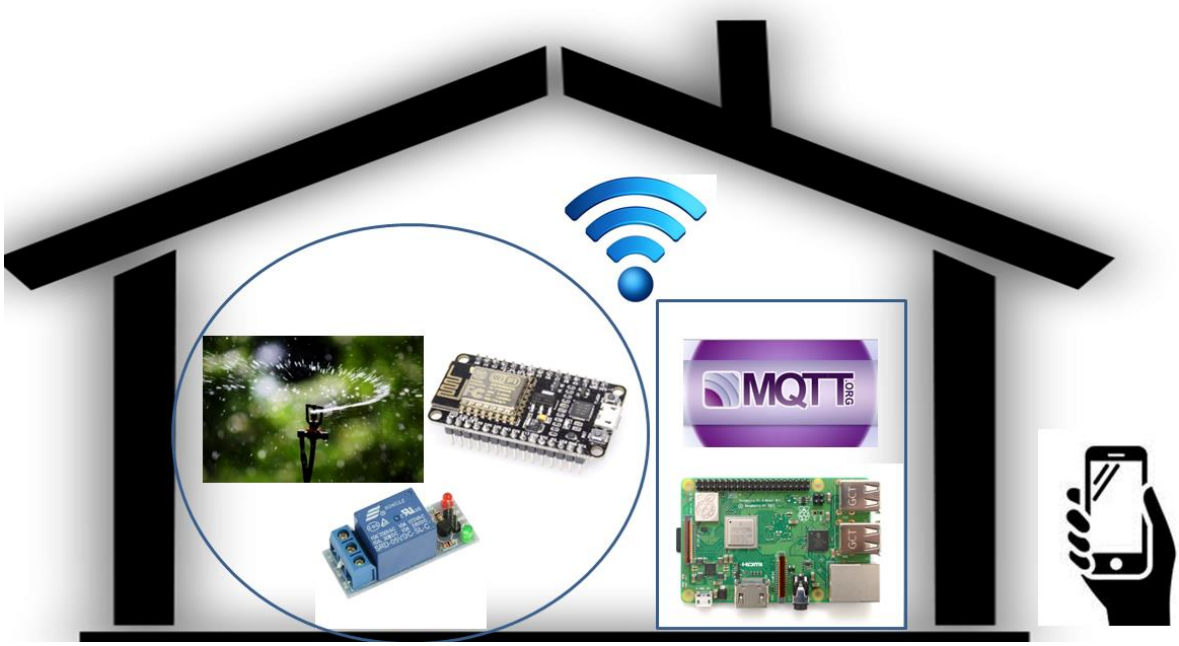
1. Introducció i motivació

1.1. Introducció

La idea d'aquest projecte sorgeix de professionalitzar, ampliar i millorar algunes petites domotitzacions domèstiques realitzades durant els últims cinc anys, com per exemple alguns relés controlats per una Rasperry Pi[1] per activar les electrovàlvules del reg automàtic, lectures de temperatura i humitat mitjançant plaques Arudino[2] o algunes petites automatització a partir de sensors de moviment i endolls WiFi[3]. Els diferents muntatges realitzats eren funcionals però independents, cadascun fet amb un hardware diferent i diferents codis, per aquest motiu es va decidir integrar-ho tot en un ecosistema tancat i autònom.

De forma abstracta es pretén automatitzar tot els elements possibles d'un habitatge amb un cost molt baix, comparant amb el preu de les opcions comercials disponible al mercat, i fer-ho limitant l'adquisició de productes comercials "clau en mà", amb l'objectiu d'aconseguir una habitatge autònom, energèticament eficient i segur. Bàsicament es controlarà la calefacció, l'aire condicionat, les persianes, el reg automàtic, les càmeres de vigilància, els tendals, l'aigua calenta, els consums elèctrics, alguns endolls i part de la il·luminació. Paral·lelament, es farà l'adquisició de dades d'una estació meteorològica semi-professional i s'utilitzaran les dades tant per la presa de decisions com per publicar-les en una web pública com a servei al municipi de Viladecavalls.

Per aconseguir aquest objectiu es combinen diferents sensors i dispositius de control, a partir de dispositius comercials i dispositius de creació pròpia mitjançant microcontroladors compatibles amb Arduino. Tots els elements són gestionats i governats des de dos servidors basats en Raspberry Pi i s'utilitza com a plataforma de gestió i control Home Assistant.



Il·lustració 1: Esquema introductori del conjunt

[1] **Raspberry Pi**: Mini ordinadors de baix cost amb prestacions bàsiques. Disposen de ports d'E/S lliures
[2] **Arduino**: Conjunt de microcontroladors i microprocessadors per crear projectes de desenvolupament.
[3] **WiFi:** Es refereix a la comunicació sense fils en xarxes TCP/IP

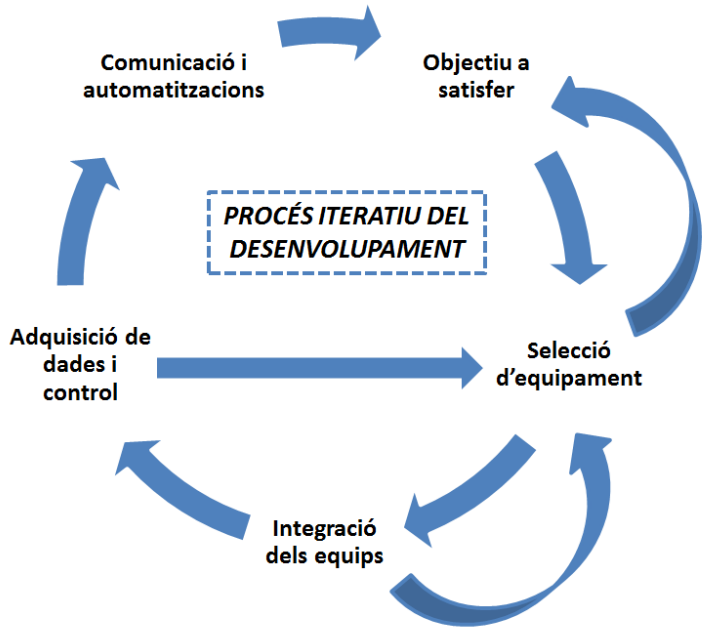
Com a esquema introductori, en la imatge anterior es poden veure els principals elements que integren una part del sistema a mode d'exemple. Sense entrar en el detall, bàsicament es mostra un habitatge on hi ha un miniordinador (Raspberry Pi) que treballa autònomament o a partir de les ordres rebudes per comunicacions externes. Aquest miniordinadors envia notificacions per un protocol determinat a través de la xarxa WiFi i aquestes notificacions les rep un microcontrolador, que segons el contingut activa un relé que inicia el reg automàtic.

1.2. Procés iteratiu

Per passar de la necessitat a satisfer, o el problema a solucionar, a un producte final funcionant de forma automàtica o a un servei acabat, s'ha seguit un procés iteratiu i interrelacionat de creació pròpia de 5 fases, que coincideixen amb els punts 2 a 6 de la present memòria. Aquest procés de desenvolupament està basat en certa manera en el model de desenvolupament de software anomenat metodologia Àgil[4] adaptat a les necessitats d'aquest projecte, ja que s'hi inclouen fases de desenvolupament de productes que no es contemplen en l'Àgil, però amb el mateix objectiu de ser un procés iteratiu i orientat a la reutilització del codi i el treball fet. Les fases de tests i avaluació del model Àgil s'han integrat en la d'integració i adquisició.

- **Objectiu a satisfer**: On es defineix que es vol automatitzar o quin servei es vol suplir.
- **Selecció d'equipament:** On es busca quin tipus d'equip pot donar solucions al problema a satisfer. Habitualment ens trobem que hi ha molts dispositius diferents en el mercat que ofereixen una mateixa funció, amb una gran disparitat de preus. Durant aquesta fase es habitual trobar nous equips que donen noves idees, pel que molt sovint es torna al primer punt amb nous objectius a satisfer.
- **Integració dels equips:** Un cop escollit l'equip a utilitzar, s'ha de fer disseny i muntatge dels circuits que facin falta per cobrir les necessitats. En molts casos, durant aquest procés s'arriba a la conclusió que els equips escollits requereixen un circuit fora de l'abast, per la dificultat o pel nivell d'integració de components que requereixen, motiu que fa replantejar els equips escollits.
- **Adquisició de dades i control:** Amb la part mecànica i electrònica funcionant s'ha de buscar la manera d'integrar-ho amb el sistema, ja sigui per fer lectures o per realitzar accions, ja sigui mitjançant codis propis, modificant el funcionament natiu o sense modificacions. També pot succeir que el dispositiu escollit no sigui possible fer-lo integrable al sistema i per tant s'ha de retornar al segon punt i escollir una nova opció.
- **Comunicació i automatització:** Com a última fase, s'integra el nou dispositiu a la plataforma de control i se li dona connectivitat. Un cop integrat s'hauran de definir les automatitzacions i/o tasques que haurà de dur a terme el nou dispositiu, habitualment prenen decisions de forma autònoma a partir dels demés elements del sistema. Aquest punt normalment ens portarà a haver complert l'objectiu inicial i acabar el procés però molt sovint ens portarà de nou a generar noves necessitats, perquè si tenim més informació disponible i tractada, poden sorgir noves idees per aprofitar aquesta informació en bé de noves metes.

[4] **Àgil**: Metodologia de desenvolupament de software que busca la reutilització de codi i la col·laboració.



Esquema 1: Procés iteratiu del desenvolupament

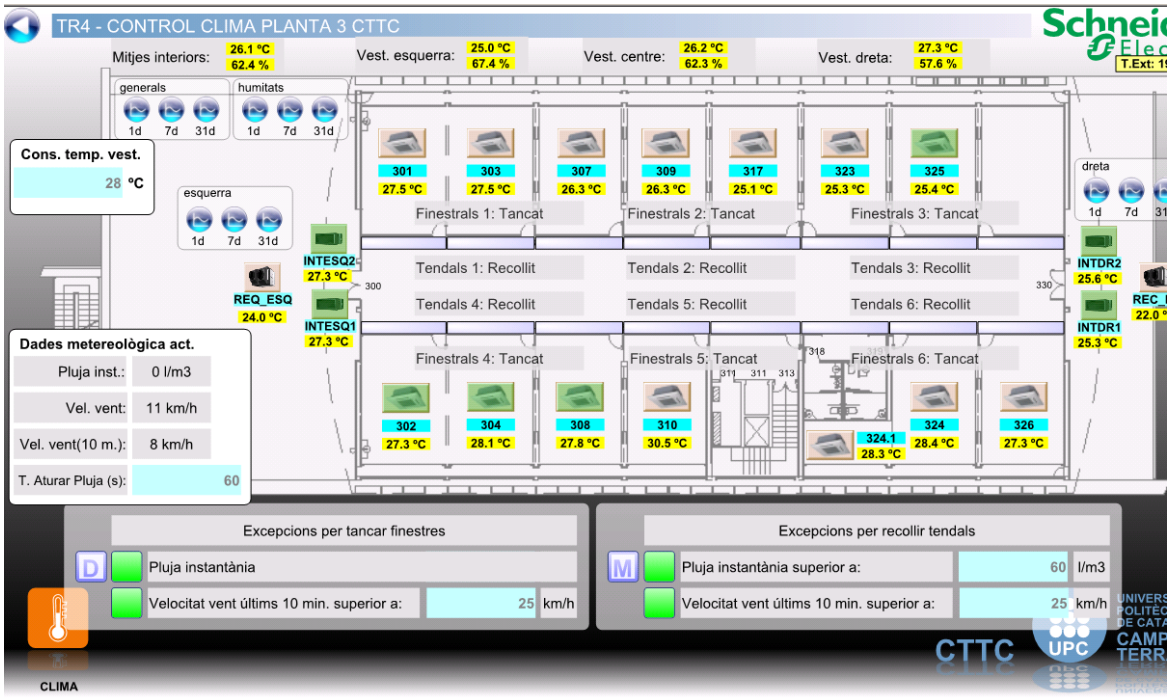
1.3.Motivació

La motivació de fer aquest projecte va venir donada pel fet que fa 6 anys vàrem inaugurar un nou edifici pel centre de recerca on treballo, el Centre Tecnològic de Transferència de Calor (CTTC)[5] de la Universitat Politècnica de Catalunya (UPC)[6], el qual pretenia ser un edifici bioclimàtic i totalment automatitzat. L'edifici que és pràcticament tot de vidre, integra uns finestrals superiors junt amb uns tendals per tapar el sol i al centre del edifici hi ha un jardí interior. També integra una façana bioclimàtica de doble pell, amb plantes de fulla caduca per controlar l'entrada de llum i radiació solar segons l'època de l'any.

Il·lustració 2: Edifici del CTTC

[5] **CTTC**: Centre Tecnològic de Transferència de Calor: www.cttc.upc.edu
[6] **UPC**: Universitat Politècnica de Catalunya: www.upc.edu

Per aconseguir que l'edifici fos autònom es va contractar una empresa especialista en automatització industrial a la qual se li va encarregar tot el projecte i com a resultat se'ns va donar un producte comercial tancat, plenament operatiu però d'un cost molt elevat. En definitiva el sistema es basa en un SCADA[7] que controla les finestres, els tendals, la climatització i la il·luminació mitjançant diferents PLCs[8].



Il·lustració 3: Part del SCADA del CTTC

El sistema resultant funciona correctament en general, però es totalment monolític. Cada modificació que s'ha volgut fer ha hagut de ser a través de l'empresa que ho va instal·lar i sempre hi ha hagut moltes dificultats per modificar el funcionament general. Arrel de viure en primera persona aquest muntatge i ser usuari i administrador alhora del sistema des de que es va posar en marxa, vaig veure que per arribar a l'objectiu de fer una casa totalment domòtica amb els elements comercials que hi ha al mercat s'ha de fer una inversió realment alta. També vaig fer la reflexió que per molt alta que sigui la inversió, sempre s'hi troben mancances. Per aquests motius es va escollir dissenyar un sistema propi, de baix cost i que fos adaptable a les noves tecnologies que van apareixent.

2. Objectius

L'objectiu global del projecte, bàsicament, és aconseguir que un sistema automàtic prengui decisions en cada moment a partir de les lectures que realitzi o la interacció amb l'usuari, amb tres objectius clars; confort, eficiència energètica i seguretat. Aquest objectiu global serà l'objectiu de satisfacció màxima o compliment total del objectiu, però probablement no tots els objectius seran satisfets completament per tant es definiran *sub* objectius segons el grau de compliment i el nivell de maduració de la tecnologia i el conjunt.

[7] **Supervisory Control And Data Adquisition:** Sistema de control complet que utilitza PLCs i GUIs.
[8] **Programmable Logic Controller:** Microcontroladors programables utilitzats sobretot en la industria.

2.1. Objectius globals

2.1.1. Confort

L'objectiu de confort busca que el sistema sigui autònom per realitzar les tasques per les que està programat. Per una banda, per comoditat, evitant als habitants fer tasques simples i repetitives i per l'altra, obtenir més confort a la llar sense la necessitat d'intervenir.

Els objectius de confort que busca solucionar aquest projecte són:

- **Climatització**: Mantenir les temperatures correctes a l'estiu i a l'hivern, tenint en compte la presencia d'habitants, la inèrcia tèrmica dels sistemes de climatització i les condicions climatològiques.
- **Il·luminació:** Gestionar intel·ligentment la il·luminació dels espais en funció de la presencia, la radiació solar externa i l'hora del dia.
- **Reg:** Regar de forma intel·ligent el jardí, tenint en compte la radiació solar, la temperatura i humitat del sòl, la pluja caiguda en els últims dies o fins i tot la previsió.
- **Evitar incidents:** Mitjançant els sensors de portes i les previsions meteorològiques el sistema haurà de notificar en cas de que hi hagi una finestra o porta oberta en quant s'abandona l'habitatge si hi ha previsions meteorològiques adverses.

2.1.2. Eficiència energètica

Per altra banda, l'objectiu del projecte també és aconseguir l'eficiència energètica de l'habitatge. Mitjançant tota la regulació que es farà es busca aconseguir un major estalvi energètic sense reduir el confort.

Els objectius d'eficiència energètica que busca cobrir aquest projecte són:

- **Persianes:** Pujar o baixar les persianes en funció de l'hora del dia i la radiació solar en funció de l'època de l'any. Per exemple, si és estiu i fa molt sol contra una habitació baixar la persiana per aïllar. Aquest objectiu també forma part de confort, perquè evita la intervenció humana.
- **Tendals:** D'igual manera que les persianes, es controlarà un tendal motoritzat per tal de controlar l'impacte del sol en els espais diürns de l'habitatge. També mitjançant els sensors de pluja instantània es recolliria el tendal en cas de pluja.
- **Control de consum:** També es monitoritzarà i es registrarà el consum elèctric de les diferents línies, mitjançant mesuradors de consum continu, per tal de saber el consum dels diferents elements de la casa, climatització, aigua calenta, cuina, etc... de forma que amb aquestes dades en un futur pròxim que es farà la instal·lació de plaques solars fotovoltaiques es podrà distribuir els consums en funció de la radiació solar, per exemple, només escalfar el dipòsit d'aigua calenta sanitària en les hores de sol o escalfar o refredar la casa si hi ha un excedent energètic encara que no es compleixin les condicions establertes de climatització, per exemple, escalfar la casa encara que no hi hagi ningú per tal d'aprofitar l'excedent.

2.1.3. Seguretat

L'objectiu de seguretat busca ampliar les funcions que faria una alarma connectada sense el cost associat. Mitjançant els sensors hauria de detectar la presència no desitjada dins i fora de l'habitatge en cas que hi hagués algú i enviar notificacions de tot el que passi. Els objectius de seguretat que hauran de cobrir en aquest projecte són:

- **Videovigilància**: Mitjançant càmeres IP en el exterior de l'habitatge, haurà de notificar als habitants de la presència de persones externes.
- **Control d'accessos**: D'igual manera s'haurà d'avisar en el cas que alguna porta sigui oberta sense la presencia dels habitants.
- **Control de moviment**: Igual que amb el control d'accessos s'haurà de notificar en cas de que es detecti moviment en el algun punt.

2.2. Quantificació dels objectius

El conjunt d'objectius definits tindran diferents graus de compliment en funció del resultat obtingut. Aquests graus de compliment vindran definits segons si s'ha aconseguit l'objectiu global, es a dir, compleix l'objectiu de treballar de forma autònoma i totalment integrada al sistema o s'ha arribat a objectius inferiors. Els objectius inferiors vindran definits per si s'ha trobat o dissenyat un equip físic que compleixi, si aquest equip s'ha pogut integrar a nivell físic amb l'habitatge, si aquest equip es comunicable des de la plataforma i per últim si es autònom. A partir de la taula següent, en el punt d'avaluació dels objectius s'avaluarà el grau de compliment dels objectius definits originalment.

Objectiu	Dispositiu seleccionat	Dispositiu integrat	Dispositiu comunicable	Dispositiu autònom
Grau de compliment	No suficient	Suficient	Notable	Excel·lent

Taula 1: Taula del grau de satisfacció

3. Selecció d'equips

Els equips físics utilitzats pel desenvolupament d'aquest projecte es divideixen en quatre categories:

1) Equips comercials d'adquisició o control. Alguns modificats a nivell de firmware[9] o amb alguna modificació i altres en estat original.
2) Equips comercials amb microcontroladors compatibles amb Arduino IDE[10].
3) Sistemes basats en microcontroladors compatibles amb Arduino IDE de disseny i muntatge propi.
4) Hardware de control

[9] **Firmware**: Programa informàtic que defineix la lògica de baix nivell per controlar qualsevol tipus de dispositiu electrònic. Té accés directa al hardware i es específic per cada hardware diferent.
[10] **Arduino IDE:** Llenguatge de programació basat en C++ que s'acostuma a utilitzar en sistemes encastats de la família Arduino o plaques similars.

3.1.Equips comercials d'adquisició o control

Com a equipaments que he optat per utilitzar sense cap mena de modificació, principalment tenim una estació meteorològica Oregon Scientific WMR180A amb els següents sensors: Temperatura i humitat en dos ubicacions exteriors i una interior, pluviòmetre, anemòmetre, penell i radiació solar UV[11]. Aquesta estació disposa d'una pantalla interior sense fils per llegir les dades i una antena USB per comunicar amb un ordinador Windows. Cal destacar que tots els sensors són sense fils i funcionen amb bateries i energia solar. El motiu d'escollir aquesta estació és perquè la interfície USB em permet accedir a les dades de forma automàtica alhora que està suportada per una llibreria Python[12], anomenada Weewx, que s'utilitza per fer l'adquisició en un sistema Linux. D'altra banda, el cost de la estació amb els sensors addicionals que he inclòs no supera els 200€, per tant es pot considerar que és assequible.

Il·lustració 4 i 5: Part de l'estació meteorològica utilitzada

D'altra banda, també he optat per utilitzar sense cap modificació uns sensors de sòl de la marca Xiaomi que em permeten llegir la temperatura, la humitat, la EC[13] del sòl o la intensitat de llum, des d'un petit dispositiu sense fils, que disposa d'una bateria superior al any de durada i té comunicació pel protocol Bluetooth[14]. Té un cost de 12€.

També de la mateixa marca Xiaomi, he utilitzat un conjunt d'elements per interactuar o rebre informació de la casa. Aquests elements són diferents sensors d'obertura de portes i finestres, sensors de moviment i interruptors sense fils. Tots ells treballen sota un protocol de comunicació anomenat Zigbee[15], similar al WiFi, de menor ample de banda però amb un

[11] **UV:** Tipus de radiació electromagnètica d'ona curta produïda entre altres pel sol.
[12] **Python**: És un llenguatge de programació d'alt nivell i propòsit general molt generalitzat
[13] **Soil electrical conductivity:** És una mesura que es correlaciona amb les propietats del sòl que afecten la productivitat dels cultius
[14] **Bluetooth:** Es un especificació per xarxes sense fils personals (WPAN), està dissenyat per dispositius de baix consum elèctric i que estiguin a poca distància.
[15] **Zigbee:** És una especificació basada en IEEE 802.15.4 per a un conjunt de protocols de comunicació d'alt nivell utilitzats per crear xarxes LAN amb ràdios digitals petites i de baixa potència

consum energètic molt inferior. Per comunicar entre aquests elements i el servidor s'utilitza una porta d'enllaç que fa de passarel·la Zigbee – WiFi.

Incrementant les modificacions, el següent element és una càmera IP de la marca Yi, on s'hi ha carregat un firmware modificat per tal de poder-hi accedir via SFTP[16] i via RTSP[17], per tal de poder descarregar els fitxers d'imatge així com poder reproduir en temps real la imatge de la càmera a la interfície de control del sistema domòtic. El cost d'aquesta càmera es de 25€

Recentment s'ha inclòs en el ecosistema domòtic un altaveu intel·ligent de la marca Amazon, amb l'objectiu de poder configurar ordres per interactuar amb el sistema domòtic. A demès de que aquest altaveu també serveix de passarel·la per comunicar amb un conjunt de bombetes de la marca Philips que treballen també amb el protocol Zigbee; ja que aquest altaveu té el hardware necessari per fer de passarel·la Zigbee – WiFi. Desgraciadament, la passarel·la que brinda l'altaveu no es compatible amb els sensors Xiaomi i viceversa amb les bombetes Philips que treballen amb Zigbee, per tant s'han de mantenir les dos passarel·les.

A nivell d'il·luminació, a part de les bombetes Philips que treballen amb Zigbee també hi ha integrades en el sistema algunes bombetes i alguns *downlights* de la Marca YeeLight que treballen directament amb WiFi i tenen un cost inferior a les Philips.

3.2.Equips comercials compatibles amb Arduino IDE

El procés de recerca de nous productes amb més capacitats i menys necessitats d'integració de circuits complexos a nivell elèctric, ha permès arribar a una línia nova de productes d'un fabricant Xinès anomenat SonOff, que comercialitza equips de domòtica domèstica de baix cost i altes prestacions. La majoria de productes d'aquest fabricant utilitzen el microcontrolador ESP8266 o el ESP8255, que són els mateixos microcontroladors que s'han utilitzat en el present projecte com a base de tots els sistemes encastats de disseny propi però en el cas d'aquests productes estan integrats en una circuit amb tots els components soldats, és a dir, en una mateixa placa hi ha els relés, resistències, convertidors de tensió i el microcontrolador. D'aquesta manera, s'ha pogut accedir a la placa electrònica dels dispositius i soldar-hi uns pins de comunicació, mitjançant els quals s'ha pogut comunicar a través d'un convertidor de USB a Sèrie RS232 TTL i s'hi ha pogut carregar els codis desenvolupats pel projecte a la memòria Flash dels dispositius, i utilitzar-los integrats en el sistema, en comptes de l'aplicació comercial pròpia.

Gràcies als equipaments d'aquesta marca, s'ha pogut centrar el desenvolupament més en el codi que en la part elèctrica. Dins aquesta categoria tenim:

- **Relés Wi-Fi**: S'ha utilitzat dos tipus d'equips, per una banda una dispositiu amb un sol relé i per l'altra, una dispositiu amb 4 relés. Les plaques d'un sol relé s'utilitzen per il·luminació ja que venen integrades amb una carcassa molt petita i van auto-alimentats, de forma que es poden amagar dins els registres elèctrics de l'habitatge amb seguretat de que no hi hauran curtcircuits. Les plaques de 4 relés són per actuar

[16] **SFTP:** Secure FTP, protocol de transferència de fitxers xifrat
[17] **RTSP:** Real Time Streaming Protocol. Estableix i controla un o diversos flixes de dades sincronitzats.

contra 2 persianes cada placa, quatre maniobres. Funcionen amb el codi que s'ha desenvolupat, tal com es detalla a la secció corresponent del codi. Aquests dispositius tenen un cost d'entre 3 i 20€

Il·lustració 6: Dos relés Sonoff d'un canal (desmuntats) + convertidor USB – RS232 TTL

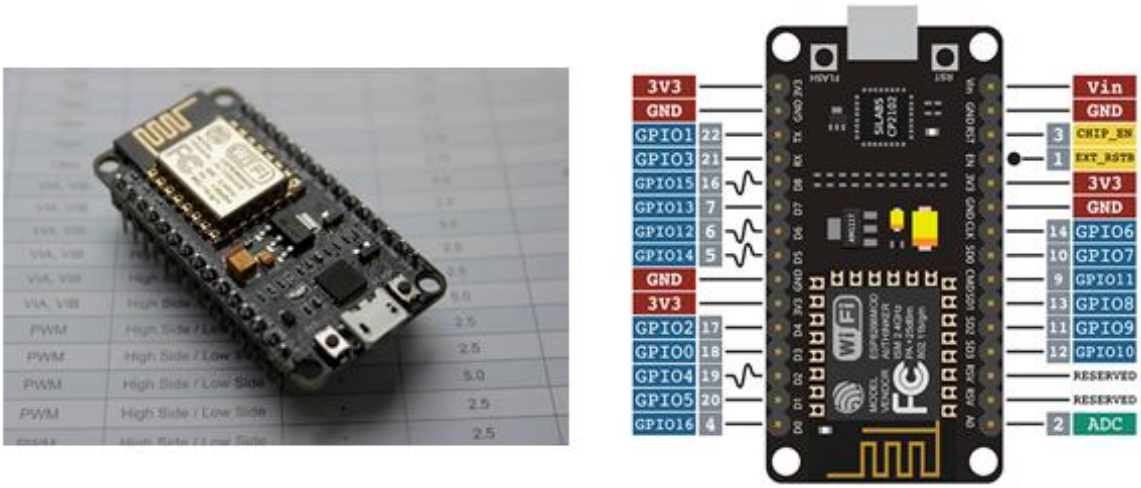
- **Interruptor de paret**: De la mateixa marca s'ha utilitzat un interruptor de paret tàctil a l'entrada de l'habitatge per indicar si s'arriba o es surt per tal de canviar les escenes. A nivell intern, únicament és un relé que s'activa o no. Aquest té un cost de 10€.
- **Mesurador de consums**: La última incorporació han estat 5 dispositius de la mateixa marca Sonoff per mesurar el consum, la potència i el voltatge en temps real. El model concret és Sonoff Pow R2. S'instal·len en el quadre elèctric i fan passar el tràfic elèctric per dins d'un relé per tal de fer les mesures. Aquests dispositius en comptes d'emprar el codi desenvolupat, s'hi ha carregat un codi GLP disponible a internet anomenat Espurna[18]; per la complexitat dels dispositius i el temps d'execució del projecte, no es pretén integrar-ho al codi desenvolupat. Tenen un cost de 12€ aproximadament, i per controlar tot el consum de l'habitatge n'hi hauran 5.
  La integració d'aquests dispositius ha generat la necessitat de dissenyar un quadre elèctric nou per encabir-los.

Il·lustració 7: Sonoff Pow R2

[18] **Espurna**: Conjunt de firmwares per tots els dispositius fabricats per Sonoff. Fet per Xose Pérez

3.3. Sistemes encastats de disseny i muntatge propi

Els gruix de sensors i actuadors del projecte es controlen mitjançant microcontroladors NodeMCU[19] V3.0 basats en el xip ESP8266. Són uns sistemes encastats compatibles amb el software Arduino IDE però molt més petits i de baix cost; aproximadament tenen un cost d'entre els 2 i 3€ cadascun. A demès tenen integrat un xip de WiFi per comunicar-hi, van alimentats a 5V, tenen una memòria de 4Mb, 10 E/S digitals i 1 E/S analògica. Mitjançant aquests E/S es faran totes les accions; com activar un relé per activar cada actuador o llegir una entrada per fer lectures, com per exemple, llegir la temperatura i humitat mitjançant un sensor digital DHT22 o un sensor analògic de pluja instantània.



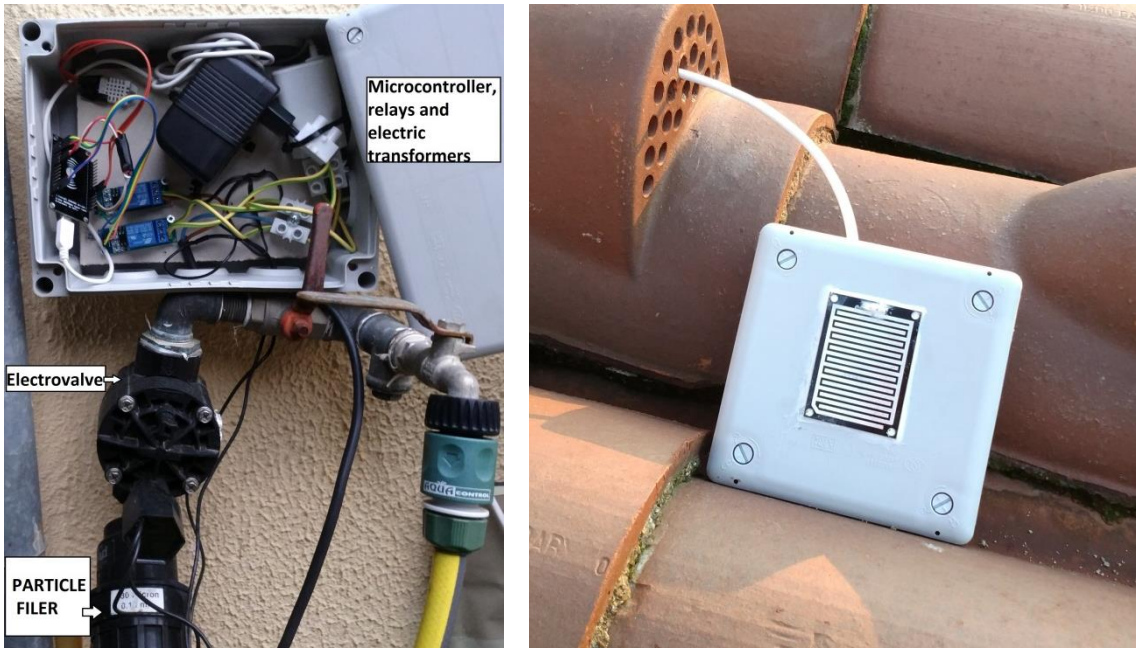
Il·lustració 8: NodeMCU V3 i esquema de pins

Principalment hi hauran els següents equips, tots ells amb el circuit electrònic i elèctric desenvolupat pel projecte i en tots els casos el microcontrolador és un NodeMCU V3:

- **Sensors:**
  - **Sensor de pluja instantània**: Circuit amb pistes que detecta diferencial de corrent, encapsulat en una caixa estanca mecanitzada per integrar el circuit i el microcontrolador. Està instal·lat a la teulada aprofitant la inclinació de les teules pel drenatge d'aigua de la placa. El circuit va connectat a un amplificador de senyal amb un offset. Te un cost de 3-4€.
  - **Sensors de temperatura i humitat interior**: Mitjançant sensors digitals DHT22 es llegeix la temperatura i humitat en varis punts de l'habitatge. Tenen una sensibilitat de 0,5ºC en la temperatura i un 2-5% en la humitat, i el seu cost és inferior a 3€. També s'utilitzen per controlar la temperatura on hi ha microcontroladors en punts solejats.
- **Actuadors:**
  - **2x electrovàlvules:** Que gestionen el reg automàtic mitjançant dos relés en una caixa estanca, també es prenen mesures de temperatura dins la caixa ja que està *sotmès directament a la radiació solar*.

[19] **NodeMCU**: és una plataforma de desenvolupament de codi obert. Inclou el firmware que s'executa al SoC ESP8266 d'Espressif Systems i el maquinari està basat en el mòdul ESP-12

- **Calefacció central**: Mitjançant un relé per engegar la caldera i un sensor DHT22 per mesurar la temperatura en la sala de la caldera.
- **AACC:** Mitjançant un relé d'estat sòlid SSR[20] de 40A i un sensor de temperatura per mesurar la temperatura del garatge, on hi ha la connexió.
- **Tendals:** Mitjançant un relé per activar les maniobres del motor.
- **Il·luminació:** Mitjançant relés s'activarà la il·luminació que es vulgui controlar. Principalment la il·luminació del garatge i el taller.



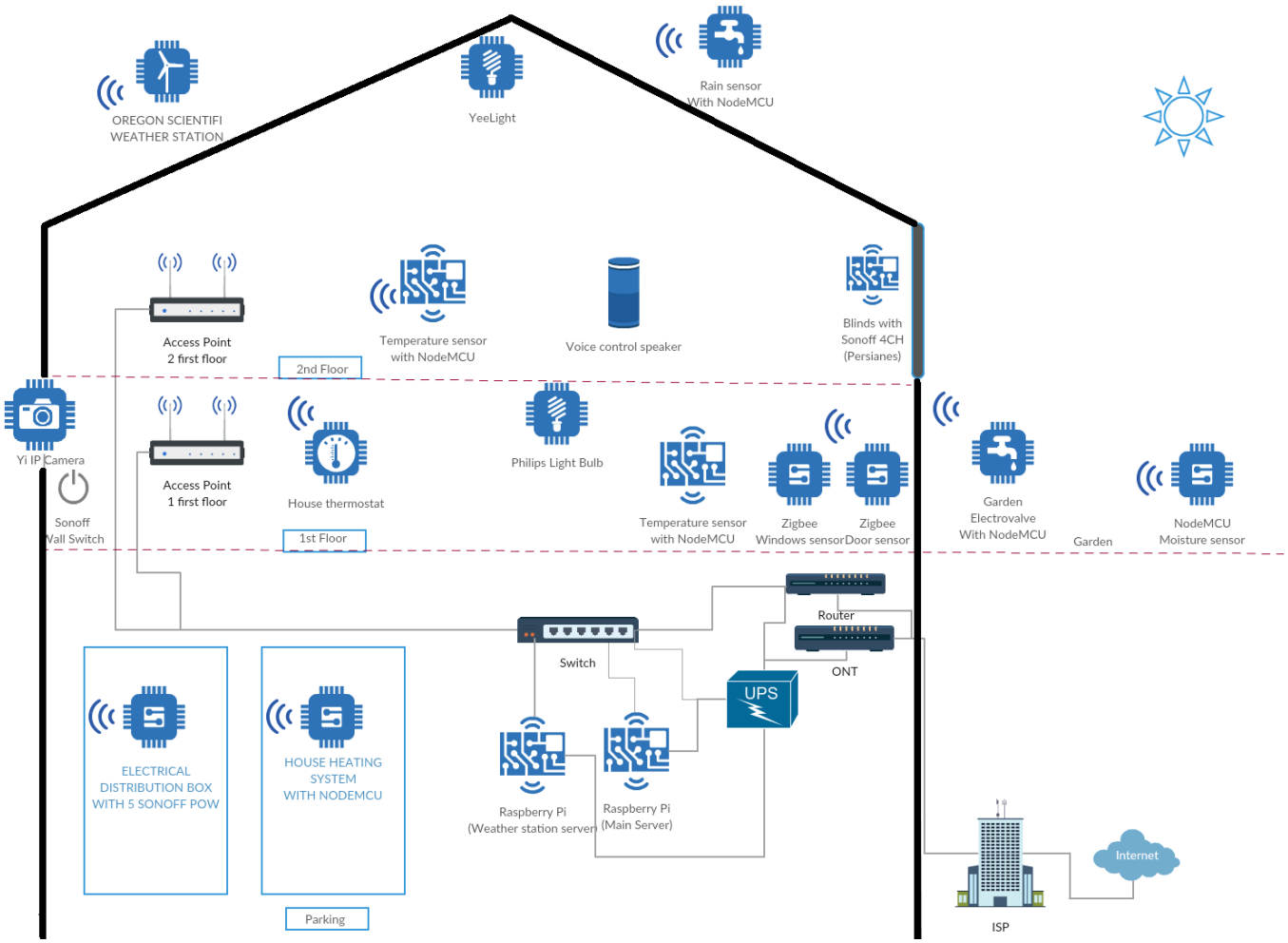
Il·lustració 9: Controlador de reg automàtic (Esquerra) i sensor de pluja instantània (Dreta)

3.4.Hardware de control

Dins del hardware de control bàsicament hi ha els dos servidors que fan la gestió de tot el conjunt. Els dos servidors estan basats en mini-ordenadors Raspberry Pi 3b i 3b+ i corren la distribució Open Source[21] Raspbian[22]. Originalment s'utilitzava un servidor DELL senzill, però el consum energètic i el cost no estaven alineats amb els objectius d'eficiència energètica i baix cost del projecte. Les plaques Raspberry Pi 3 tenen un cost de 35€ cada una.

El servidor principal és una RaspberryPi 3b+ i és on s'executa el servei de MQTT, les bases de dades, la pàgina web, etc. Bàsicament és on corren tots els serveis, per tant és força crítica. Per aquest motiu, està alimentada juntament amb el router i el punt d'accés amb un SAI. A part del servidor principal, hi ha una altre RaspberryPi 3b que és el servidor on correrà el codi d'accés a l'estació meteorològica (*Weewx*), que farà de passarel·la entre la estació i el servidor. Un cop definits tots els elements que integraran el projecte, es planteja un esquema on podem observar els elements distribuïts dins l'habitatge, ja siguin equips de control, d'adquisició o actuadors

[20] **SSR**: Es un tipus de relé més modern i més costos, amb un temps de vida superior i una capacitat de tall molt superior, els utilitzats SSR són de 40A versus els mecànics que són de 16A.
[21] **Open Source**: Codi amb llicencia oberta, disponible per ser editat per la comunitat.
[22] **Raspbian**: Distribució Linux basada en Debian, pensada per plaques Raspberry Pi.



Esquema 2: Esquema simplificat de la majoria dels elements distribuïts dins l'habitatge (Sense elements repetits).

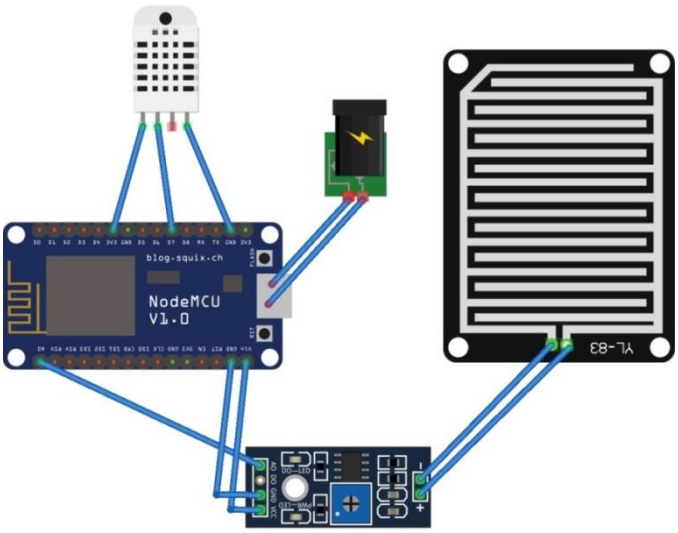
4 Integració d'equips

A l'etapa d'integració és on s'han dissenyat els circuits que s'utilitzaran com a sensors o actuadors en el conjunt del sistema. Es pot distingir entre diferents circuits que es repetiran en diferents entorns del sistema, ja que per exemple, el circuit per activar el reg automàtic serà el mateix tipus de circuit que engegarà la caldera, ja que els dos treballaran activant un relé . Així, per una banda tindrem els circuits que activen relés, i dins d'aquests els que activen relés integrats en una placa comercial, d'un canal o de quatre i els relés activats des del sistema encastat NodeMCU. D'altra banda, hi haurà els circuits que controlen sensors de temperatura i humitat i sensors de pluja instantània a través d'aquesta mateixa placa.

Els següent circuit mostra el cablejat entre un microcontrolador NodeMCU i el sensor de temperatura i humitat DHT22, i el sensor de pluja en un mateix disseny. Bàsicament el sensor DHT22 va alimentat a 3V i GND i te un port connectat a una E/S digital, mitjançant el qual transmet les lectures de la temperatura i la humitat.

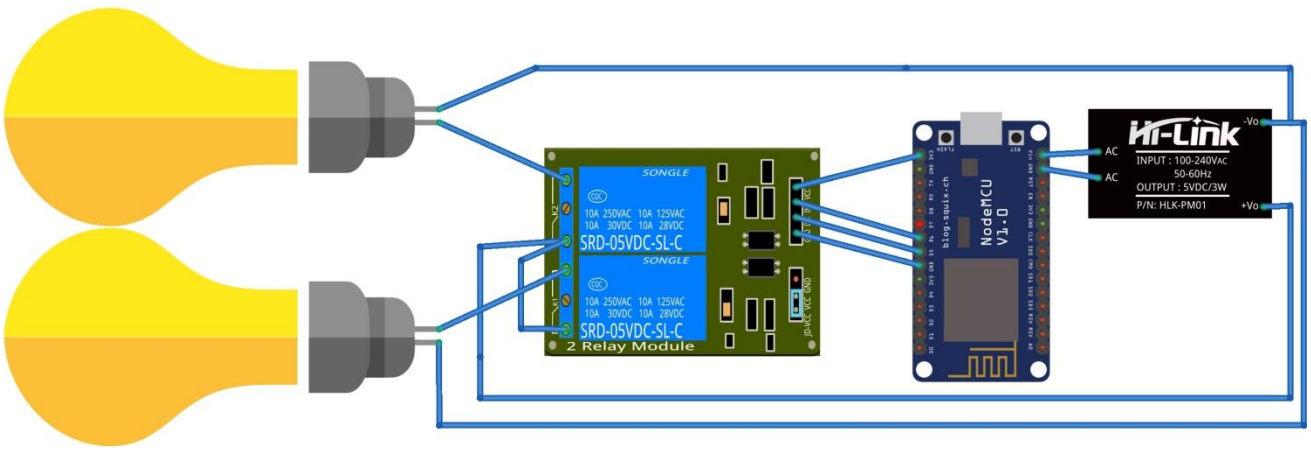
D'altra banda, en l'anterior circuit també tenim el sensor de pluja instantània, que va connectat a l'amplificador de senyal que a la vegada fa de circuit de control. A nivell de connexions va alimentat a 3V i GND i connectat a l'única E/S analògica, tot i que la placa de

control té un port digital no és operatiu, sembla ser que la placa de control és genèrica per altres dispositius.



Esquema 3: Circuit amb sensors de temperatura i gotes de pluja

Els següent circuit mostra el cablejat entre un microcontrolador NodeMCU i un circuit de dos relés mecànics. En els diferents muntatges he utilitzat plaques de 1, 2 i 4 relés de diferents tipus, alguns mecànics i alguns SSR. Per il·lustrar el circuit hi ha connectats dos bombetes als relés, però en els circuits reals també hi haurà altres dispositius com electrovàlvules, endolls, una caldera, etc. En aquest cas el circuit va alimentat mitjançant els ports Vin i GND connectats a la placa directament, en comptes d'utilitzar el port MicroUSB integrat a la placa i un carregador de USB. En alguns muntatges per reduir espai s'ha utilitzat l'electrònica interna d'un transformador de 5V sense el cable USB. S'ha obviat la alimentació a 220V dels dos circuits.



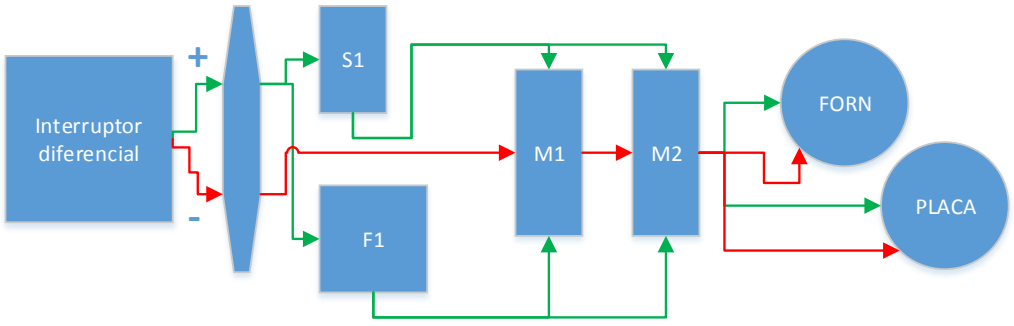
Esquema 4: Circuit per controlar dos relés i dos bombetes

En les últimes etapes del projecte, s'ha estat desenvolupant un sistema per mesurar els consums domèstics mitjançant els dispositius Sonoff Pow R2 explicats en la secció 3.2. Aquest desenvolupament ha sigut fins el moment el més complex de disseny i d'integració dels

equips, ja que amb el pretext de mesurar el consum, s'ha dissenyat i construït des de zero un quadre elèctric nou per tot l'habitatge, val a dir, que encara no està en producció.

Il·lustració 10: Quadre elèctric creat durant el desenvolupament

El quadre està compost per un interruptor diferencial, 5 mesuradors de consum i 15 magneto-tèrmics. El disseny està pensat per a que es pugi treballar en dos estats diferents; mesurant el consum o no, per tal de tenir una alternativa de funcionament en cas que algun dels dispositius mesuradors tingui problemes o arribi al final de la seva vida útil. Per fer-ho s'ha establert un camí normal de treball, on la corrent passa pel diferencial, i es reparteix en un distribuïdor, des d'on s'alimenten els 5 mesuradors (S1-S5) i els 5 magneto-tèrmics (F1-F5). Cada parella de mesurador i magneto-tèrmic, per exemple S1+F1, connecten amb 2 línies de l'habitatge, per exemple S1+F1 alimenten M1 i M2, que podrien ser dos línies de cuina per exemple. L'objectiu d'aquest muntatge, és que si el magneto-tèrmic està baixat, la corrent escollirà el camí que passa pel mesurador i a continuació a les línies, però en canvi, si el magneto-tèrmic està pujat, la corrent tindrà dos camins a escollir, mesurador o magneto-tèrmic, en aquest cas com que el mesurador actuarà com una resistència la corrent tendirà a passar pel magneto-tèrmic i evitarà el mesurador, d'aquesta manera, amb només amb 5 magneto-tèrmics es pot definir dos camins alternatius amb selector i estar protegits contra possibles fallades del mesurador.



Esquema 5: Esquema simplificat d'una part del quadre elèctric. (Color verd Fase, Vermell neutre)

Junt amb el quadre elèctric hi ha una pantalla LCD on es mostren els consums de les diferents línies. Aquesta pantalla es controla també des d'una placa NodeMCU i amb l'ajut d'una passarel·la I2C[23] per tal simplificar les connexions. En l'apartat de 5.3 s'explica més en detall el codi que s'utilitzarà per mostrar la informació per pantalla.

En la present memòria no s'ha detallat circuits més complexos com per exemple el que controla les persianes, on per dos persianes intervenen 4 relés, 4 polsadors manuals i dos motors amb diferents maniobres, ja que es considera que és més a nivell elèctric que electrònic. D'igual manera amb el circuits del quadre elèctric, on no s'ha detallat la gestió de les fases i els neutres que s'ha fet per minimitzar el cablejat, així com amb els diàmetres del cablejat i les potències suportades per cadascun dels dispositius.

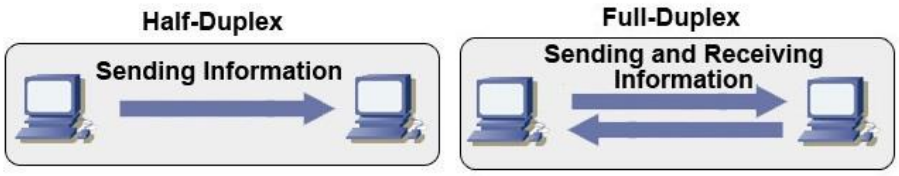
5 Adquisició de dades i control

5.1 Antecedents i evolució

Inicialment només es controlaven dos relés; un pel reg automàtic i l'altre per la calefacció, mitjançant scripts de Bash que s'executaven directament en dos Raspberry Pi 1 en que es treballava anteriorment. Amb el temps es va començar a emprar diferents Arduinos per llegir temperatura de sensors mitjançant simples codis, però no era possible comunicar-s'hi per obtenir les dades. Arrel de la necessitat de comunicació es va arribar a la conclusió d'emprar les plaques de desenvolupament NodeMCU i es va començar a treballar en un codi que feia la comunicació entre el servidor i les plaques NodeMCU mitjançant un codi propi d'Arduino IDE i sockets[24] TCP/IP. Aquest codi, tot i que era funcional, tenia limitacions i una de les més importants era que no era possible que el microcontrolador notifiqués el servidor sense ser preguntat; només es podia treballar per Pooling[25], ja que encara que es programés una ISR quant es detectes una condició determinada, no es podia notificar al servidor fins que no fos preguntat de nou.

Per suplir la necessitat de comunicació bidireccional es va actualitzar la part de comunicació feta en sockets TCP/IP al protocol MQTT[26], que està dissenyat per treballar amb el concepte de publicacions i subscripcions[27], que permeten obviar la direcció IP de cada NodeMCU i comunicar-s'hi fent publicacions i rebre notificacions. És a dir, el codi original fet amb sockets era unidireccional (half-duplex) i gràcies a la integració amb MQTT va passar a ser bidireccional (full dúplex).

[23] **I2C:** Protocol síncron que utilitza dos cables per comunicar (Rellotge i dades). A diferència de la UART en aquest protocol els dos cables s'utilitzen bidireccionalment, per enviar i rebre.
[24] **Sockets**: Es refereix a la comunicació entre dos equips en una xarxa TCP/IP, on la comunicació s'estableix per un port TCP o UPD i una direcció IP
[25] **Pooling**: En sistemes distribuïts es coneix com pooling el procediment de fer peticions recurrents.
[26] **MQTT**: Message Queuing Telemetry Transport https://wikipedia.org/wiki/MQTT
[27] **Publish–subscribe pattern:** https://wikipedia.org/wiki/Publish-subscribe_pattern



Esquema 6: Comparativa Half Duplex / Full duplex

Més en detall, el MQTT és un protocol on intervé un servidor anomenat Broker i un numero "n" de clients, que poden ser publicadors o subscriptors. Els missatges que s'envien s'anomenen tòpics i bàsicament són una cadena de caràcters separats per un caràcter "/". Aquests tòpics poden ser completats amb el que s'anomena un Payload, que és un valor numèric associat al tòpic, que serveix per completar el tòpic, per exemple, en una lectura de temperatura el tòpic serà l'identificador del sensor i el Payload el valor de la lectura.

En el cas d'aquest projecte els clients són cadascun dels microcontroladors però també és un client el usuari (o un servei deamon[28]) que sol·licita informació o dóna ordres a les plaques. Per tant, el flux comunicació podrà ser de dos tipus:

- El client publicarà un tòpic per realitzar una determinada acció, per exemple activar un relé. En funció del tòpic enviat, un NodeMCU determinat realitzarà l'acció.
- El client (o el *daemon* que gestiona el sistema) estarà subscrit a un tòpic, que per exemple indica la temperatura del jardí, cada cop que el NodeMCU fa una lectura publicarà la temperatura i el client la rebrà.

5.2 Desenvolupament del codi

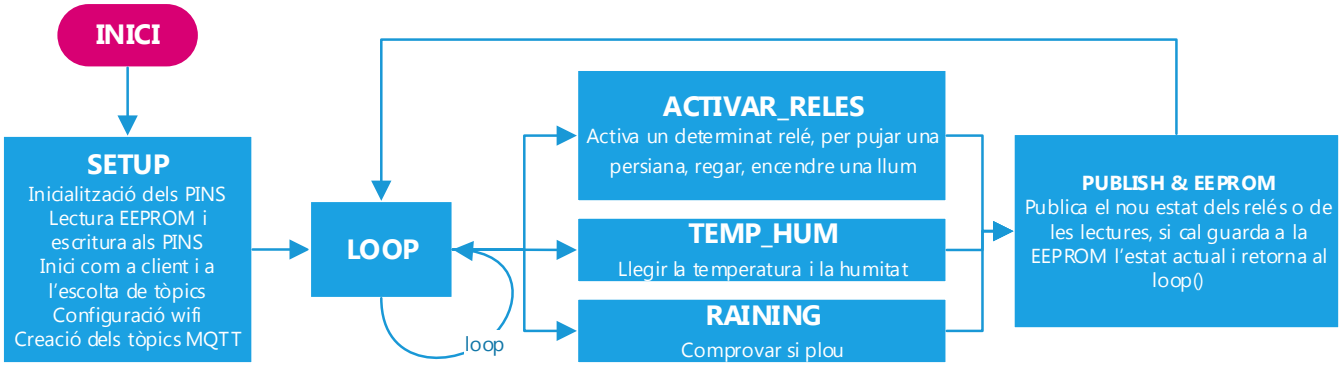
El codi actual, fet en llenguatge Arduino IDE, s'ha realitzat utilitzant el model anomenat SuperLoop, que és el model més basic de desenvolupament en sistemes encastats, on s'executen les funcions seqüencialment i no es poden definir interrupcions. Aquest model és més bàsic que els models RTOS, que també són compatibles amb Arduino IDE mitjançant FreeRTOS per exemple, però pel present codi no s'ha considerat necessari, ja que no es requereix d'una alta precisió i a més les interrupcions necessàries son limitades i es poden solucionar mitjançant *pooling*, és a dir, comprovant si s'ha activat una notificació cada determinat temps. També cal destacar, que la carrega de treball pels diferents sistemes encastats és molt baixa tot i estar basat en SuperLoop, per tant tampoc a nivell de recursos ha sigut necessari treballar amb un model RTOS.

Aquest codi, orientat a treballar amb el protocol MQTT, està desenvolupat per ser un únic codi per totes les plaques que treballen amb el microcontrolador ESP8266, ja siguin circuits propis amb les plaques NodeMCU o amb equips comercials basats en aquest en el microprocessador ESP8266. Per aconseguir aquest objectiu el codi llegeix els últims 6 dígits de l'adreça MAC[29] del microcontrolador i s'utilitzen per generar el tòpic arrel de MQTT, de manera que les diferents plaques estaran identificades per la seva MAC. Gràcies a això, es pot substituir qualsevol placa

[28] **Daemon**: Un programa informàtic que s'executa i corre en segon pla sense intervenció de l'usuari.
[29] **MAC**: Adreça de 48 bits que defineix el fabricant del xip de WiFi i és única.

únicament anotant la nova MAC, indiferentment de la tasca que realitzés el microcontrolador anteriorment.

El codi, de menys de 400 línies en la seva versió final optimitzada, es pot consultar en els annexes de la memòria. Abans d'entrar més en el detall, en el següent esquema es pot observar de forma molt simplificada què realitzarà el codi desenvolupat. Bàsicament el microcontrolador s'inicia fent les definicions i configuracions inicials i a continuació es quedarà en bucle esperant rebre ordres. Les ordres que rebrà bàsicament seran per activar relés, llegir temperatures o comprovar si està plovent. Un cop l'ordre s'ha realitzat, publicarà el nou estat o les dades relatives a la lectura i retornarà al bucle.



Esquema 7: Funcionament simplificat del codi desenvolupat

A continuació s'explica de forma més detallada el seu funcionament, juntament amb un diagrama de blocs complert de les operacions que realitza:

A l'iniciar-se el microcontrolador NodeMCU es defineixen les variables globals, entre les que cal destacar les de configuració de la xarxa i del protocol MQTT, així com les assignacions de noms als diferents ports d'entrada i sortida que governa el microprocessador ESP8266 que disposa la placa. L'execució continua amb la funció pròpia d'Arduino IDE "*setup()*" on es definirà l'estat inicial dels diferents ports, en funció de si són entrades o sortides, es llegirà i aplicarà l'estat dels ports d'entrada i sortida emmagatzemats en la memòria EEPROM[30] durant l'última execució i es configurarà per una banda el servei de MQTT, la connexió WiFi, un servei per imprimir notificacions per sèrie en cas de depuració d'errors i un servei per rebre actualitzacions del codi per OTA[31], de forma que es pugi actualitzar el codi remotament, sense accedir per un port USB sèrie al NodeMCU o al dispositiu SonoOff.
Per últim, dins el "*setup()*" també es cridarà a la funció "*init_id()*" que és la responsable de la creació de tots els tòpics. Aquesta funció llegirà la adreça MAC del microcontrolador i la utilitzarà com a primera part del tòpic, tant publicacions com per subscripcions, juntament amb un nom. Exemples de tòpics complerts serien: "81CDE1/out/raining", "81CDE1/out/relay6" o "81CDE1/in".

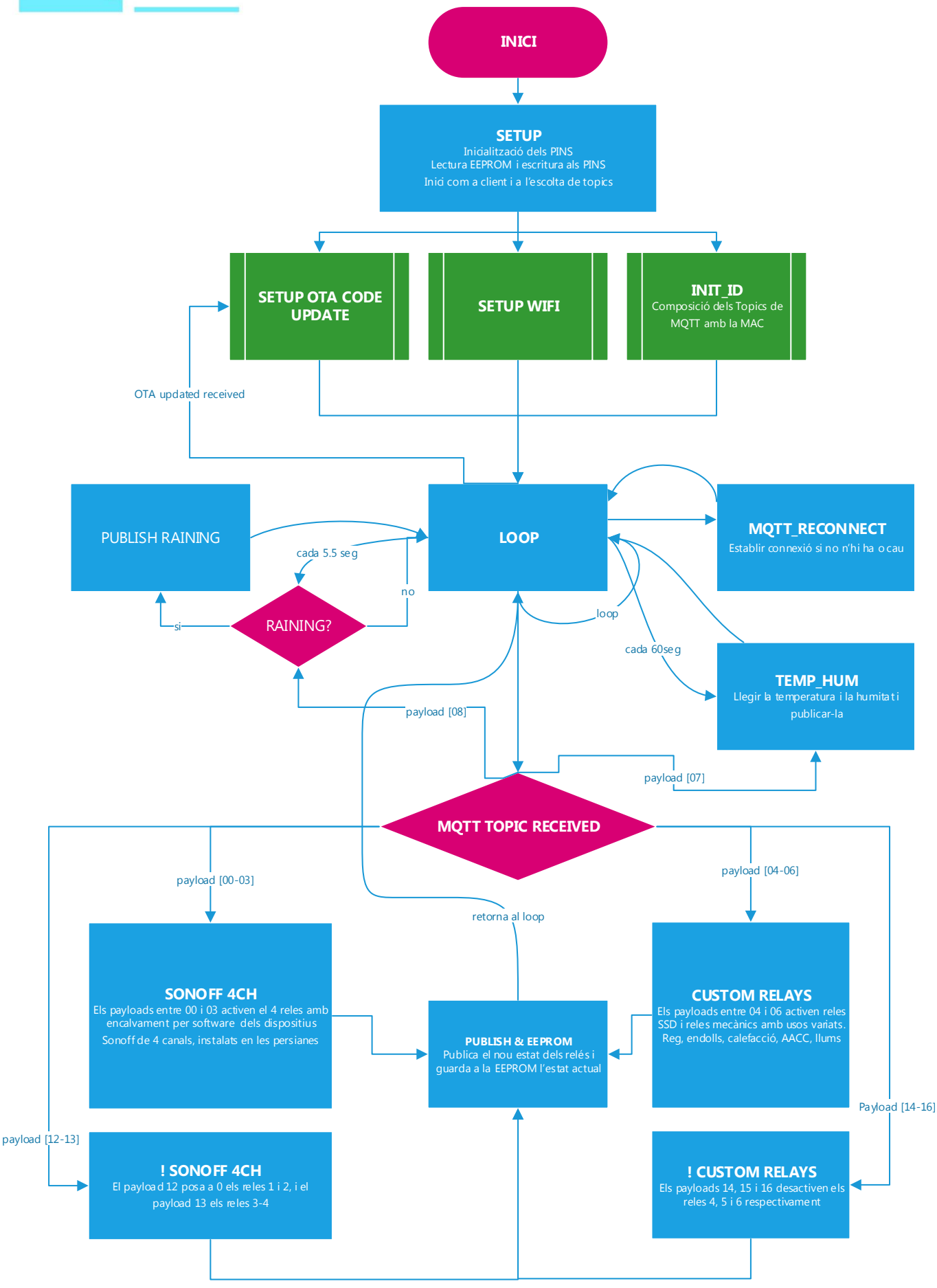
[30] **EEPROM**: Electrically Erasable Programmable Read-Only Memory. Memòria programable y esborrable elèctricament

[31] **OTA**: Over-The-Air. Sistema que permet distribuir actualitzacions de forma remota.

El codi continuarà amb la següent funció típica d'Arduino IDE, la funció *loop().* Dins d'aquesta funció es realitzaran les següents tasques:

- Es comprovarà si hi ha actualitzacions del codi disponibles per OTA.
- En cas que es perdi la connexió amb el Broker de MQTT, la restablirà.
- Cada minut es farà una publicació de la temperatura i la humitat actual. Per fer-ho es llegirà el valor d'un sensor de temperatura digital DHT22 i el valor llegit es publicarà com a Payload del tòpic corresponent.
- Cada 5 segons es comprovarà si plou. Per fer-ho es llegirà la única entrada analògica que disposa el NodeMCU i si el valor de la lectura és superior a 10 (que ens indicarà que hi ha un sensor connectat) i és inferior a 950, publicarà que està plovent.
  Les lectures van de 0 a 1024, hi després de molts tests, s'ha arribat a la conclusió que aquest rang és l'òptim.
- Cada 5 segons publicarà un tòpic indicant que el NodeMCU està operatiu.
- Per últim i com a tasca més important, s'estarà escoltant si es rep un tòpic que el destí coincideixi amb la MAC del NodeMCU. Si és el cas, es comprovarà el Payload rebut associat al tòpic i segons el valor realitzarà les següents accions:
  - Payload del [**00-03**]: Els valors del 00 a 03 serviran per activar els 4 relés integrats en les plaques modificades de Sonoff de 4 canals. Com que l'ús que se'ls hi dona és pujar i baixar persianes, es realitza un enclavament[32] per software, de forma que si arriba el Payload 00 s'activa el relé 0 i es desactiva el relé 1. D'aquesta manera s'eviten curtcircuits. Un cop activats els relés es publica el canvi d'estat en el relé. El motiu de utilitzar un rang específicament pel circuits SonOff de 4 canals és perquè els ports que utilitza són ports GPIO amb diferent numeració als que utilitza el NodeMCU, i com l'objectiu és tenir un codi únic per tots els dispositius, per això s'han codificat accions diferents.
  - Payload del [**12,13**]: Els valors 12 i 13 s'utilitzen per posar els relés 0-1 i 1-2 a 0 respectivament. Són els complementaris dels Payloads [00-03]
  - Payload del [**04-06**]: Els valors del 04 a 06 serviran per activar relés independents, ja siguin en les plaques modificades de Sonoff de 1 canal o en els circuits propis amb relés mecànics o sòlids. Un cop fet el canvi d'estat, es publicarà el canvi i s'emmagatzemarà a la memòria EEPROM.
  - Payload del [**14-16**]: Els valors del 14 a 16 serviran per desactivar relés, són els complementaris dels Payloads [04-06] Un cop fet el canvi d'estat també es publicarà i s'emmagatzemarà a la memòria EEPROM.
  - Payload del [**07**]: El valor 07 farà una lectura de temperatura i humitat sota demanda i la publicarà.
  - Payload del [**08**]: El valor 08 farà una comprovació de si plou sota demanda i publicarà que plou si és el cas.

[32] **Enclavament**: En un dispositiu elèctric, normalment un relé, s'utilitza per bloquejar una maniobra en activar una contraria. Normalment es realitza a nivell de circuit però també es pot fer per codi.



Esquema 8: Funcionament del codi

5.3 Codi mesurador de consum

Per tal de mostrar en una pantalla les lectures dels diferents consums, s'està desenvolupant un codi que per una banda, es subscrigui a les publicacions dels 5 mesuradors de consums i guardi les lectures. D'altra banda, a partir de les lectures, composar unes cadenes de caràcters per tal de mostrar-los a una pantalla I2C, a través de la llibreria *LiquidCrystal*. Aquest codi no està en producció ja que el desenvolupament d'un codi que estigui subscrit a múltiples tòpics de MQTT no està contemplat a les llibreries de MQTT utilitzades, i encara s'hi està treballant. Encara que l'objectiu és fer-ho com s'ha documentat, també s'ha valorat alternativament fer que un equip Unix, com el servidor per exemple mitjançant un script de Bash o de Python, es subscrigui a totes les publicacions dels diferents mesuradors, les processi i publiqui solament una que ja sigui la composició de totes, de forma que el microcontrolador NodeMCU només es subscrigui al tòpic que conté tota la informació i la imprimeixi per pantalla cada determinat temps, i així simplificar aquest codi addicional.

5.4 Resta de dades i accions

La lectura de dades i les activacions dels sensors amb la versió del codi que funcionava amb *sockets* es realitzaven mitjançant scripts de *bash* i aplicacions de *smartphone* que permetien executar-les de forma fàcil, amb aplicacions com Termux[33], així com bots[34] de T*elegram*[35]. D'altra banda els dispositius de tercers que funcionaven amb codi propi, cadascun treballava de forma independent amb la seva aplicació, però tot aquest sistema era molt poc escalable i poc pràctic. Per aquest motiu i gràcies a la inclusió del protocol MQTT en els circuits propis i la estació meteorològica es va poder migrar tot el sistema a un programa Open Source que s'anomena Home Assistant (*www.home-assistant.io*) que permet la integració de tots els elements en una mateixa pàgina web; ja siguin dispositius compatibles amb el protocol MQTT, les Gateways Zigbee amb tot el que tenen connectat, les càmeres de vigilància i bàsicament qualsevol dispositiu que tingui IP i estigui suportat. Aquest servei més endavant es detalla tota la part tècnica en l'apartat de Sistemes i serveis i la de control en el apartat d'automatitzacions, ja que Home Assistant permet definir les automatitzacions. També en el punt de Sistemes i serveis es detallarà la transmissió de les dades meteorològiques, bàsiques per la presa de decisions en les automatitzacions.

6 Comunicacions i seguretat

Per tal de gestionar el conjunt de dispositius es requereix de diferents xarxes que comuniquin tots els dispositius amb els servidors, així com formes d'actuar des de dins i des de fora de l'habitatge. A nivell de comunicacions es distingeixen entre les internes i les externes, ja siguin cablejades o sense fils, així com segons la seguretat definida.

A nivell d'internes hi ha quatre tipus de comunicacions:

[33] **Termux**: Aplicació que permet emular un terminal Linux i fins i tot executar scripts.
[34] **Bots:** Programes autònoms que realitzen accions de forma desatesa.
[35] **Telegram**: Aplicació principalment de smartphone per parlar amb altres usuaris. Té grans capacitats addicionals com per exemple integrar-ho amb sistemes Unix i crear configuracions autònomes i segures.

- Xarxa Ethernet: Mitjançant un cablejat de categoria 6 instal·lat en diferents punts de l'habitatge dóna connectivitat TCP/IP als tres punts d'accés WiFi així com als dos servidors basats en Raspberry Pi. La seguretat d'aquesta xarxa ve donada per la necessitat d'accés físic i el filtratge de MACs.
- Xarxa WiFi: Mitjançant la xarxa sense fils es comunicaran tots els elements de control; ja siguin microcontroladors amb xip WiFi integrat, càmeres, passarel·les de comunicació etc. També és la plataforma d'accés per interactuar amb el sistema. La xarxa WiFi està configurada amb una encriptació WPA2[36] i filtratge de adreces MAC.
- Xarxa Bluetooth: Alguns dels dispositius ja definits es comunicaran mitjançant una comunicació Bluetooth amb la passarel·les Bluetooth – WiFi. La seguretat en la connexió ve donada per una banda perquè no poden comunicar dispositius externs prèviament no enllaçats i per altra banda perquè la comunicació va xifrada.
- Xarxa Zigbee: Igual que la xarxa Bluetooth alguns dels dispositius ja definits es comunicaran mitjançant Zigbee amb les passarel·les Zigbee – WiFi. Igual que el Bluetooth les comunicacions Zigbee estan xifrades per defecte, utilitzen una encriptació AES[37].

En canvi, les connexions externes es faran sobre una línia de fibra òptica segura mitjançant una VPN[38]. S'explica en detall en punt de sistemes i serveis.

7 Sistemes i serveis

A nivell de sistemes i serveis tots estan distribuïts entre els dos servidors, un principal i un que fa de passarel·la del altre per determinats serveis, tots dos són Raspberry Pi 3B+. Sense entrar en la configuració pròpia dels serveis, es comenten a continuació les funcionalitats dels més destacables així com el paper que tenen en el entorn.

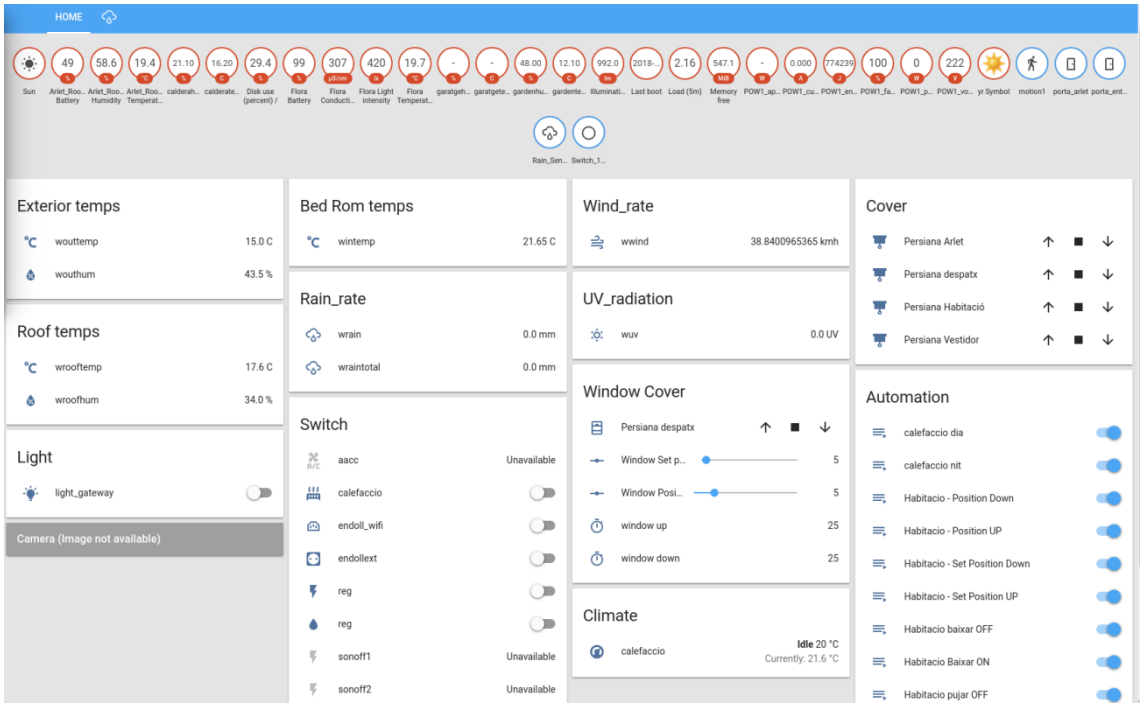
7.1 Raspbian

Raspbian és el sistema operatiu escollit com a host dels dos servidors, és un fork[39] de Debian àmpliament utilitzat per el seu grau d'optimització en aquests equips de baix consum i limitada potència. Aporta la disponibilitat de paquets que brinda el gestor de paquets Aptitude juntament amb la disponibilitat dels drivers[40] i binaris compilats per l'arquitectura ARM que utilitzen les plaques Raspberry Pi, de forma que es poden instal·lar llibreries que d'altra forma no serien compatibles amb l'arquitectura encara que es compilessin in situ.

[36] **WPA2:** Seguretat definida actualment en els dispositius Wi-Fi. Integra la autentificació i la encriptació.
[37] **AES:** Tipus d'encriptació basada en el xifrat mitjançant blocs.
[38] **VPN:** Xarxa privada punt a punt, utilitza un xifratge SSL i una autentificació client servidor.
[39] **Fork:** Concepte que s'utilitza en programació per referenciar que un codi descendeix d'un altre.
[40] **Driver:** També conegut com controlador. Conjunt de programari que permet al sistema operatiu comunicar amb un dispositiu o perifèric mitjançant una abstracció del hardware per utilitzar el dispositiu

7.2 Home Assistant

Home Assistant es una plataforma de codi obert per integrar dispositius de control i dispositius del Internet De Les Coses (IOT)[41], desenvolupada en Python3 i compatible amb tots els sistemes operatius ja que únicament necessita Python3, tot i que on més àmpliament s'utilitza és en mini ordinadors Raspberry Pi. El gran avantatge que té es que és compatible amb pràcticament qualsevol dispositiu pensat per automatitzar, monitoritzar o gestionar en entorns domèstics, ja que té documentat com utilitzar una gran quantitat de dispositius comercials sense grans dificultats.



Il·lustració 11: Pestanya principal de la interfície de Home Assistant.

En el marc d'aquest projecte, tot i que la majoria d'elements utilitzats funcionen amb un codi propi, es poden integrar perfectament al sistema perquè comuniquen amb el protocol MQTT. Home Assistant disposa de configuracions especifiques per relés MQTT, sensors MQTT o fins i tot mòduls desenvolupats per enllaçar diferents dispositius MQTT per gestionar per exemple la climatització on hi intervenen sensors i actuadors al mateix temps.

El funcionament de Home Assistant es basa en un servei corrent sobre el sistema operatiu Raspbian. Aquest servei quant s'inicia llegeix els fitxers de configuració, en sintaxis YAML[42], on estan definits tots els dispositius que ha de controlar, alhora que totes configuracions, ja siguin gràfiques o de funcionament. Dins la mateixa plataforma i mitjançant el mateix llenguatge es programen les automatitzacions que es realitzaran segons els diferents esdeveniments que succeeixin. Més endavant en el punt 8 es detalla com funcionen les automatitzacions.

[41] **IOT:** "Internet Of Things" es refereix a tots els dispositius que històricament no tenien connexió i que actualment s'hi pot accedir remotament, rentadores, neveres, càmeres.
[42] **YAML:** Llenguatge per crear fitxers de text *serialitzats*. Similar al XML i inspirat en Python i Perl

7.3 Weewx

El servidor secundari del sistema, també en un una Raspberry, s'encarrega de comunicar mitjançant una llibreria anomenada *Wewwx* feta en *Python* i llicència GPL amb l'estació meteorològica Oregon Scientific. Aquesta llibreria, que s'executa en forma de *deamon*, serveix de *driver* per comunicar la estació amb la Raspberry, que rep les dades de l'estació, les processa i les escriu en una base de dades MySQL[43] dins el servidor principal. Aquesta llibreria també s'encarrega de fer la composició de fitxers HTML i gràfiques que composaran la pàgina web on es podran visualitzar totes les dades meteorològiques de forma oberta, i les copiarà mitjançant *RSYNC*[44] al servidor principal que és el que té sortida a internet.

En aquesta base de dades en el servidor principal es recullen les dades de la temperatura i humitat exterior en dos punts, així com la velocitat i sentit del vent, la radiació solar i la quantitat de pluja. Totes aquestes dades, es mostren a una pàgina web pública com a servei al municipi, en funcionament des de fa tres anys. La web és https://www.meteovila.cat. Totes aquestes tasques es fan des de la Raspberry secundaria, que únicament fa de passarel·la entre la estació i la Raspberry primària, de forma que en cas de caiguda del servei es segueix publicant la web amb les últimes dades meteorològiques obtingudes.

Després de la inclusió de Home Assistant, es va adaptar el funcionament del servei Weewx per tal de que també publiques les dades meteorològiques amb el protocol MQTT, de forma que el servidor secundari publica cada 5 minuts totes les lectures que han fet els sensors i les inclou en la interfície de Home Assistant, així com les utilitza per la presa de decisions en les automatitzacions.

7.4 Servidor web

El servidor principal disposa d'un servei de publicació de pàgines web mitjançant Apache2 per dos tasques principals. Per una banda, per la publicació de la pàgina web amb la informació meteorològica i per altra banda per la publicació de la interfície de control de Home Assistant.

Home Assistant integra el seu propi gestor web, AIOHTTP, que és un client/servidor de pàgines web desenvolupat amb Python, però vista la necessitat de publicar més planes web des del mateix servidor i les limitacions del servei, es va optar per configurar un servei Apache2. Cal destacar que mitjançant la configuració d'Apache2, es dona servei a les dos planes web mitjançant connexions xifrades i certificats SSL vàlids. També mitjançant un proxy[45] invers es redirigeix el tràfic extern HTTPS del subdomini de control al port que treballa Home Assistant.

[43] **MySQL:** Sistema de base de dades relacional àmpliament utilitzat.
[44] **RSYNC:** Aplicació de Unix per tal de crear còpies de dades incrementals i diferencials de forma eficient. Pot funcionar sobre SSH, qui li proporciona la seguretat i la comunicació.
[45] **Proxy:** Un servidor proxy s'encarrega de fer d'intermediari entre les peticions d'un client a un servidor, en aquest cas, les peticions que arriben al servidor per un port, les redirigeix a un altre.

7.5 Bases de dades

Hi ha bases de dades de dos tipus, només en el servidor principal. Per una banda, la que emmagatzema les dades meteorològiques, que funciona amb un servei de MySQL. Aquesta base de dades és abocada a un arxiu de backup diàriament. L'altre base de dades del sistema, és interna del servei de Home Assistant i funciona mitjançant SQlite 3[46]. Aquesta base de dades és únicament de servei, ja que les dades emmagatzemades només són de les accions i lectures dels últims dies, per tant no se'n fa copies. En cas de falta de la base de dades, es crea una nova automàticament en iniciar el servei

7.6 Copies de seguretat

Es realitzen mitjançant el servei de Unix Cron[47] copies de seguretat diàries de les diferents configuracions dels serveis a un directori. També es sincronitzen les imatges de les càmeres de seguretat mitjançant un script que comunica per sFTP, alhora que es fa un bolcat de les bases de dades també en aquest directori. Un cop realitzades les diferents copies en el directori de copies locals, es sincronitza amb un disc de copies extern mitjançant RSYNC per tal de redundar la seguretat.

7.7 DHCP i DNS

Arribat el punt de que hi ha més de 30 dispositius amb una direcció IP és realment necessari disposar d'un servei de traducció de noms (DNS) i un servei d'assignació d'adreces (DHCP). Aquesta necessitat és a nivell d'usuari i de gestió i no a nivell de requeriment tècnic, ja que la majoria de comunicacions són a través del protocol MQTT que no requereix de noms ni IPs per funcionar; només s'ha de definir la configuració del Broker (el servidor que gestiona els missatges). La resta de dispositius són configurats en Home Assistant, habitualment sense definir la IP del dispositiu; únicament es defineix la MAC, però en canvi per temes de control i monitorització és més pràctic accedir mitjançant un nom DNS a un dispositiu.

Per simplificar la gestió de IPs s'ha configurat el servei de Unix BIND per subministrar resolució de noms als diferents clients de la xarxa. La configuració s'ha realitzat per tal de que el servidor principal sigui la Raspberry principal i el servidor secundari sigui la Raspberry d'adquisició de meteorologia, de forma que en cas de no disponibilitat d'una, l'altre la substitueix. També s'ha desenvolupat un script de Bash que s'encarrega de sincronitzar les modificacions dels fitxers de noms d'un servidor a l'altre.

Paral·lelament al servidor de resolució de noms s'ha configurat un servei d'entrega dinàmica de IP (DHCP) que permet donar una direcció IP determinada a cada equip de la xarxa en funció de la seva adreça MAC. Aquest servei aporta dos avantatges: per una banda permet tenir controlats tots els dispositius connectats a la xarxa, limitant l'ús d'adreces no registrades per

[46] **SQlite 3**: Sistema de gestió de base de dades relacional, que es basa en un sol arxiu.
[47] **CRON**: Programa habitual de Unix que serveix per programar i automatitzar tasques i execucions.

mantenir un control, i per l'altra banda permet disposar de la taula amb les direccions MAC de cadascun dels dispositius informació necessària per tal de tractar les diferents publicacions i subscripcions realitzades des de tots els microcontroladors, ja que el tòpic arrel ve definit per la MAC que té gravada cada placa.

8 Automatitzacions

8.1 Introducció

El conjunt d'automatitzacions que es pretenen portar a terme en aquest projecte es fonamenten en la gestió de automatitzacions que realitza Home Assistant, mitjançant una API[48] per tal de programar-les.

Abans d'entrar en el detall de les automatitzacions cal definir com gestiona els estats dels dispositius Home Assistant. Cada dispositiu, ja sigui una llum, un sensor de humitat o fins i tot la ubicació geogràfica d'un habitant de la casa, té un estat associat i es considera una entitat. Cada entitat està composada per tres elements bàsics:

- Un identificador únic, anomenat ***entity_ID***. Aquest identificador alfanumèric s'utilitza per identificar i referenciar l'entitat, ja sigui per definir regles d'automatització o per ubicar l'entitat a la plataforma amb una icona determinada.
- L'estat, anomenat ***state*** que pot ser binari, com obert o tancat, una dada numèrica, com la temperatura o la ubicació d'una persona mitjançant les coordenades o altres com la posició del sol o la previsió meteorològica (plujós, solejat, ennuvolat).
- Els atributs, anomenats ***attributes*** són tota la informació addicional relativa a l'entitat. Poden ser atributs la unitat de mesurament, la icona que es mostrarà a la interfície, l'estat per defecte i tot tipus d'informació rellevant per tal de definir l'entitat.

Un cop definides les entitats i les seves propietats es pot definir què són els grups; que bàsicament són unes entitats que agrupen altres entitats. Són molt útils per simplificar les automatitzacions, ja que una regla es pot aplicar a totes les entitats que formen un grup. Un exemple en seria el grup de il·luminació, que englobaria totes les entitats que controlessin una llum.

A continuació es detalla com funcionen les automatitzacions, on la programació de les regles segueix una estructura de tres parts. Dins de cada una d'aquestes parts es poden utilitzar condicionals (if / else), comparadors i tot tipus d'operadors.

- **Trigger:** La primera part és el *trigger* (disparador), on es defineixen les regles que desencadenaran o activaran l'automatització. Es defineixen una o més condicions que en cas que es compleixin s'activarà la rutina. Aquestes regles poden ser un canvi d'estat d'una entitat o d'algun dels atributs d'una entitat.

[48] **API:** Sigles de *Application Programming Interface*. Són un conjunt de subrutines, funcions i mòduls que serveixen com a capa d'abstracció per programar.

- **Condition:** La segona part són les condicions, que bàsicament són un conjunt de regles per condicionar en quins casos s'ha de produir la rutina. Són opcionals i busquen ampliar els *triggers*, ja que permeten afinar més les condicions necessàries per tal de que s'iniciï la rutina. Un exemple serien les condicions disjuntives, on volem que es compleixi una regla d'un mateix tipus amb estat diferent a la del *trigger*, per exemple, si el *trigger* és que l'habitant A estigui dins l'habitatge la condició pot ser que l'habitant B estigui a més de 10km.
- **Action:** Per últim tenim la secció d'accions, on hi haurà les tasques que es realitzaran en el cas de que la regla sigui llençada (*trigger*) i les condicions es compleixin. Aquestes accions podran ser de molts tipus, com per exemple canvis d'estat d'entitats per a que realitzin una acció determinada, o fins i tot l'activació o desactivació d'una altra regla d'automatització.

A partir de l'estructura definida, únicament cal crear un fitxer d'automatitzacions, i crear cada una de les automatitzacions que es vulguin definir, afegint un camp d'identificació per identificar-les i separar-les. A continuació es mostren a mode d'exemple tres automatitzacions. Una que canvia la temperatura del termòstat, una altra que encén una llum si detecta moviment i és de nit, i una altra que la para als deu minuts.

```
- alias: 'calefaccio dia'
  trigger:
    platform: time
    at: '08:00:00'
  action:
    - service: climate.set_temperature
      data:
        entity_id: climate.calefaccio
        temperature: '20'

- alias: 'Encendre llum escala si detecta moviment i es de nit'
  trigger:
    platform: state
    entity_id: sensor.motion_sensor
    to: 'on'
  condition:
    platform: sun
    event: sunset
  action:
    service: homeassistant.turn_on
    entity_id: light.light_escala

- alias: "Apaga llum escala si detecta moviment als deu minuts'
  trigger:
    platform: state
    entity_id: sensor.motion_sensor
    to: 'off'
    for:
      minutes: 10
  action:
    service: homeassistant.turn_off
    entity_id: light.light_escala
```

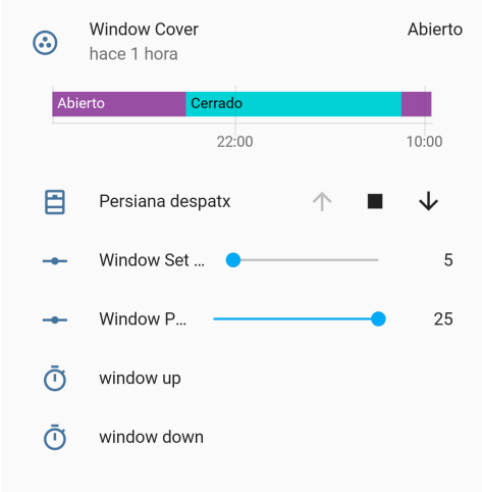
Esquema 9: Exemple de codi de dos automatitzacions

8.2 Automatitzacions intrínseques

Una de les virtuts de Home Assistant és que ja inclou un seguit de funcions software *buit-in*[49] per realitzar determinats conjunt d'accions típiques en un habitatge. D'entre les disponibles s'han utilitzat les següents:

- **Timmers:** Tot i no ser una funcionalitat específica, indirectament mitjançant els temporitzadors es poden realitzar les accions en espais de temps determinats.
- **Covers***:* Mitjançant el que Home Assistant anomena *Covers,* es poden gestionar les maniobres d'una persiana o una porta basculant. Mitjançant la definició d'aquest element, es poden gestionar tres maniobres, pujar, baixar i parar. Tenen una gran utilitat perquè mitjançant un sol element es poden gestionar les tres maniobres, i a demès, es poden crear fàcilment grups que permeten pujar o baixar totes les persianes alhora. A continuació es mostra l'aspecte d'un *cover* complet, n'hi ha 4, un per a cada persiana.

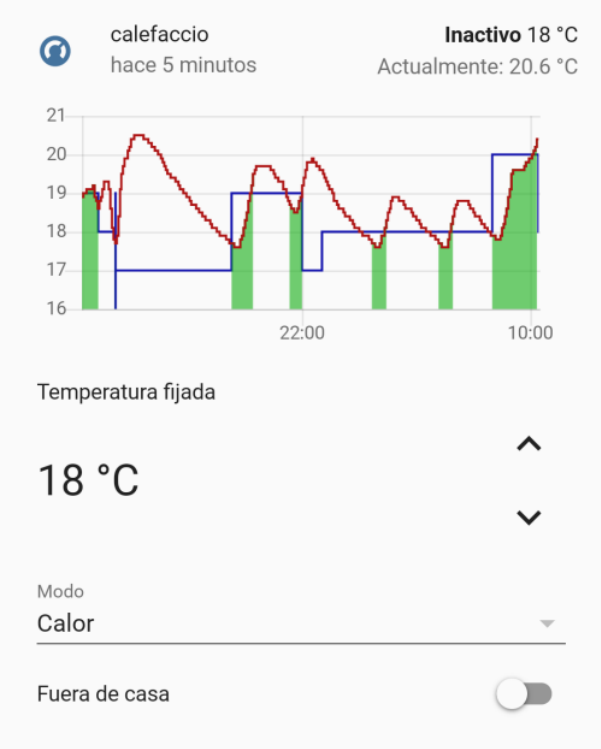
  A partir d'un conjunt de regles d'automatització més complexes i diferents temporitzadors s'ha desenvolupat la barra de la part inferior per poder definir la posició de la persiana; és a dir per no fer una maniobra completa. Aquesta part només es pot utilitzar de forma manual.



Il·lustració 12: Control disponible per una persiana.

- **Templates:** Igual que amb els temporitzadors, mitjançant la funcionalitat de plantilles, ens permet crear funcions que seran utilitzades per diferents automatitzacions. Un exemple aplicat a aquest projecte ha estat en la gestió de les persianes; on s'han utilitzat temporitzadors pels temps de maniobres i plantilles per definir el conjunt d'automatitzacions necessàries per fer maniobres parcials.
- **Climate:** Mitjançant l'element *Climate* es pot crear un termòstat virtual, que dóna la possibilitat d'integrar un o més sensors de temperatura i un relé connectat al sistema de calefacció per tal de crear un equip virtual tipus NEST[50]. Un cop definit aquest element, es podrà definir la temperatura objectiu o establir el mode de funcionament, normal, *"away mode"* o parada.

[49] **Built-In software:** Funcions ja desenvolupades en el programari que només cal instanciar o utilitzar.
[50] **NEST:** Companyia que desenvolupa termòstats intel·ligents i connectats, actualment són la referencia.



Il·lustració 13: Termòstat virtual (*Climate*)

(Color verd, encès. Color Vermell, temperatura. Color Blau, objectiu)

8.3 Esdeveniments

Una part determinant de les automatitzacions de Home Assistant ve determinada pels diferents esdeveniments que podem utilitzar. D'entre tots els disponibles n'hi ha alguns de destacables. Un d'ells és un sistema que mitjançant un programa instal·lat en els telèfons intel·ligents dels habitants de l'habitatge el servei els té geolocalitzats, a partir d'aquesta informació i d'unes àrees definides es pot determinar si un habitant està en una ubicació determinada, com per exemple el lloc de treball. Gràcies a això es poden definir accions en funció dels canvis d'ubicació, com per exemple, surt del lloc de treball. Altres esdeveniments molt útils en Home Assistant són els que ens donen la informació solar i meteorològica: aquests esdeveniments ens permeten realitzar accions com per exemple en la sortida del sol o la posta del sol, o en funció dels canvis meteorològics com per exemple si comença a ploure. Per últim, també són molt útils els esdeveniments basats en l'hora del dia, on el *trigger* és una hora determinada.

8.4 Llista d'automatitzacions

A continuació es defineixen un llistat d'automatitzacions per complir amb els requisits inicialment definits. Per simplificar la llista i evitar duplicitats s'utilitza l'operador || per

determinar més automatitzacions tot i que a la pràctica seran diferents. Es disposen amb l'estructura *Trigger – Condition – Action* de que requereix Home Assistant.

- **Canviar a estat absent:**
  - *Trigger*: Prémer el botó de l'entrada del habitatge.
  - *Condition*: Estat absent desactivat
  - *Action*: Canviar d'estat a absent l'esdeveniment Absent . Posar el termòstat en "Away mode". Apagar el grup il·luminació. Activar les notificacions del grup de seguretat. Si hi ha finestres o portes obertes i hi ha previsió de pluges notificar.
- **Canviar a estat no absent:**
  - *Trigger*: Prémer el botó de l'entrada del habitatge.
  - *Condition*: Estat absent activat.
  - *Action*: Canviar d'estat a no absent l'esdeveniment Absent. Posar el termòstat en "Heat". Apagar el grup il·luminació. Desactivar les notificacions del grup de seguretat.
- **Pujar persianes planta baixa / primera planta:**
  - *Trigger*: "Sortida del sol" o "Les 08.30h"
  - *Condition*: Ha d'haver-hi hagut moviment a la planta baixa / a la primera.
  - *Action*: Activar relés de pujada de les persianes planta baixa / primera planta durant 25s
- **Baixar persianes planta baixa / primera planta:**
  - *Trigger*: Posta del sol || Comença a ploure
  - *Condition*:
  - *Action*: Activar relés de baixada de les persianes planta baixa / primera planta durant 25s

    **Mitjançant un IF*: Si la radiació solar és superior a 3UV i l'hora del dia està entre les 9:00 i les 13:00, baixar les persianes de la planta baixa, si són entre les 13:00 i les 19:00 les de la primera planta. (Unes estan a la façana nord i les altres a la sud).
- **Augmentar la temperatura calefacció:**
  - *Trigger*: Les 08:00h
  - *Condition*: Estat absent desactivat i ser un mes d'hivern.
  - *Action*: Posar el termòstat a 21ºC
- **Reduir la temperatura calefacció:**
  - *Trigger*: Les 23:30h
  - *Condition*: Ser un mes d'hivern
  - *Action*: Posar el termòstat a 18ºC
- **Reduir la temperatura del AACC:**
  - *Trigger*: Les 08:00h
  - *Condition*: Estat absent desactivat i ser un mes d'estiu.
  - *Action*: Posar el termòstat a 24ºC
- **Augmentar la temperatura del AACC:**
  - *Trigger*: Les 23:30h
  - *Condition*: Ser un mes d'estiu
  - *Action*: Posar el termòstat a 27ºC

- **Regar:**
  - *Trigger*: Les 20.00h
  - *Condition*: Humitat terra > 40% + No previsió de pluja en les pròximes 24h.
  - *Action*: Activar reg 15 minuts.
- **Estendre el tendal:**
  - *Trigger*: "Sortida del sol" + "Ser estiu"
  - *Condition*: Radiació solar superior a 4UV + Temperatura interior > Temperatura exterior
  - *Action*: Activar relés de estendre el tendal 40 segons.
- **Estendre el tendal:**
  - *Trigger*: Posta del sol
  - *Condition*: Ser estiu
  - *Action*: Activar relé de recollida del tendal 40 segons.
- **Detecció de moviment i/o portes:**
  - *Trigger*: Detecció de moviment sensors exteriors o interiors i/o obertura de portes.
  - *Condition*: Estat absent activat.
  - *Action*: Notificació push.
- **Encendre llum escala:**
  - *Trigger*: Detecció de moviment sensor moviment escala.
  - *Condition*: Ser de nit.
  - *Action*: Encendre la llum de l'escala.
- **Apagar llum escala**
  - *Trigger*: Detecció de moviment sensor moviment escala..
  - *Condition*:
  - *Action*: Passats deu minuts apagar la llum de l'escala.

Aquest és un recull de les automatitzacions definides inicialment i que compleixen la majoria dels requisits definits en els objectius. Les automatitzacions d'il·luminació es limiten a tancar les que quedin obertes i a encendre la de l'escala així com les automatitzacions relacionades amb els controls de consum no es plantegen inicialment. En general es requereix d'un temps de funcionament del conjunt per optimitzar i determinar les noves funcionalitats que es requereixen al sistema.

9 Pressupost

En el següent pressupost, s'han inclòs tots els elements utilitzat al llarg dels últims anys per arribar al conjunt que actualment conforma el sistema. Només s'han inclòs els elements directament relacionats amb l'adquisició o el control i no elements addicionals com cablejat, motors o equips elèctrics. Només es pressuposten els elements adquirits i no el temps.

Concepte	Unitats	Preu Unitari	Total
Hardware de control			
Raspberry Pi 3b+	2	37.00 €	74.00 €
Carregador, Micro SD 32GB i caixa	2	18.00 €	36.00 €
Acces Point WiFi AC	2	30.00 €	60.00 €
Sistemes encastats de disseny i muntatge propi			
Node MCU V3 - ESP8266	10	3.60 €	36.00 €
Sensor de pluja instantània	1	2.50 €	2.50 €
Sensor temperatura i humitat - DHT22	4	3.00 €	12.00 €
Electrovàlvules	2	34.00 €	68.00 €
Transofrmadors 12v	2	8.00 €	16.00 €
Transofrmadors 5v	2	6.00 €	12.00 €
Rele 1 canal	3	2.40 €	7.20 €
Rele 2 canals	5	3.30 €	16.50 €
Relé sòlid SSR	1	6.70 €	6.70 €
Equips comercials compatibles amb Arduino IDE			
Sonoff 4 canals. Rele wifi	2	19.00 €	38.00 €
Interruptor de tàctil de paret amb wifi	1	9.00 €	9.00 €
Sonoff 1 canal. Rele Wifi	2	5.50 €	11.00 €
Sonoff Pow R2 - Mesurador de consum continuo	5	9.40 €	47.00 €
Equips comercials d'adquisició o control			
Oregon Scientific WMR180A	1	150.00 €	150.00 €
Sensor UV Oregon Scientific	1	40.00 €	40.00 €
Termohigrometre exterior Oregon Scientific	1	35.00 €	35.00 €
Mi Flora. Sensor sòl	1	12.00 €	12.00 €
Xiaomi Security Kit. (Passarel·la zibgee més sensors)	1	45.00 €	45.00 €
Sensors extres, 2 de porta, dos interruptors i un moviment	1	35.00 €	35.00 €
Endoll WiFi Xiaomi	1	12.00 €	12.00 €
Càmera YI Home. Càmera IP	1	28.00 €	28.00 €
Amazon Echo Plus	1	120.00 €	120.00 €
Bombetes Phipils HUE	1	15.00 €	15.00 €
Bombetes YeeLight	7	9.00 €	63.00 €
TOTAL			**1,006.90 €**

Taula 2: Pressupost global

10 Resultats i viabilitat

Els resultats del projecte venen determinats pel grau de maduració del conjunt, que ha estat en general bo però ha mancat temps d'execució. Tot i que no s'han arribat a complir uns objectius potser massa optimistes, el plantejament inicial no contemplava les successives ampliacions que s'han realitzat, ni tampoc contemplava la dificultat d'integrar tots els elements i automatitzar-los. Aquest últim punt és el que principalment no ha complert amb els objectius marcats, ja que en la majoria d'objectius marcats no s'ha arribat disposar d'un element autònom.

Per tal de quantificar el compliment dels objectius marcats i detectar la viabilitat de cadascun d'ells, s'analitzarà l'estat actual de cada objectiu en el moment de la redacció del present document. La valoració del grau de compliment serà mitjançant la taula següent:

Objectiu	Dispositiu seleccionat	Dispositiu integrat	Dispositiu comunicable	Dispositiu autònom
Grau de compliment	No suficient	Suficient	Notable	Excel·lent

Taula 3: Taula del grau de satisfacció

10.1 Resultats individuals

Tipus i objectiu	Dispositiu seleccionat	Dispositiu integrat	Dispositiu comunicable	Dispositiu autònom	Comentari
Confort: Climatització	Si, per AACC i calefacció	Si, disseny, muntatge i codi propi	Si, integrat totalment en Home Assistant.	Si, en ambdós casos.	Termòstat virtual funciona complint els objectius, amb una millora d'eficiència.
Confort: Il·luminació	Si, de diferents tipus i marques	Si, dispositiu comercial integrat sense modificacions	Si, integrat totalment en Home Assistant.	Parcialment, s'apaguen totes i es pot encendre una.	Pràcticament no s'utilitzen sensors de moviment ni interruptors sense fils per encendre i apagar.
Confort: Reg	Si	Si, disseny, muntatge i codi propi	Si, integrat totalment en Home Assistant.	No, s'ha de activar de forma manual.	No compleix els requisits per treballar de forma autònoma en funció de les condicions.
Confort: Evitar incidents	Si, varis	Si, dispositius comercials integrats sense modificacions	Si, integrat totalment en Home Assistant.	No, no hi ha cap regla definida que compleixi l'objectiu.	No s'utilitzen per enviar notificacions de cap mena, no compleixen l'objectiu.
Eficiència energètica: Persianes	Si	Si, dispositius comercials modificats amb codi propi	Si, integrat totalment en Home Assistant.	No, no hi ha cap regla definida que compleixi l'objectiu.	No compleix les funcions d'eficiència energètica ni tampoc les de confort treballant autònomes.
Eficiència energètica: Tendals	Si	No. No es disposa de motor en els tendals	No.	No.	L'objectiu estava condicionat a motoritzar els tendals i no s'ha realitzat.
Eficiència	Si	Si, dispositius	Si, integrat	Parcialment,	Tot i que està dissenyat

energètica: Control de consums		comercials modificats amb codi propi	totalment en Home Assistant.	ja que únicament és informació.	i integrat en el sistema, no està en producció, encara està en desenvolupament.
Seguretat: Videovigilància	Si	Si, dispositius comercials integrats amb modificació de firmware.	Si, integrat en Home Assistant.	No, notificacions a través de la aplicació pròpia.	Tot i que es poden visualitzar les càmeres des de Home Assistant, no hi ha cap regla que les utilitzi.
Seguretat: Control d'accessos i control de moviment	Si, varis	Si, dispositius comercials integrats sense modificacions	Si, integrat totalment en Home Assistant.	No, no hi ha cap regla definida que compleixi l'objectiu.	No s'utilitzen per enviar notificacions de cap mena, no compleixen l'objectiu.

Taula 4: Taula d'avaluació dels resultats

11 Conclusions

El projecte iniciat com a tal fa més d'un any, ha patit moltes modificacions i canvis de sentit, ja que l'idea original, molt més simple i com ja s'ha comentat, s'ha anat ampliant a mesura que s'ha realitzat la recerca de noves opcions alhora que s'han hagut d'adaptar a les limitacions trobades.

Sens dubte, els factors que més han revolucionat i han permès portar aquest projecte a un nivell de prestacions molt superiors a les esperades en començar-hi a treballar inicialment han estat Home Assistant i el protocol MQTT, ja que ha solucionat moltes de les incògnites inicials a les que es buscaven solució; com la integració de tot el conjunt o la comunicació bidireccional on hi hagués la possibilitat de que els dispositius notifiquessin al servidor i no només a la inversa. Per altra banda, aquests nous factors també han derivat en la impossibilitat de finalitzar tots els objectius dins del termini establert, motiu pel qual s'han definit els punts 11.1 i 11.2 on s'estructuren les diferents tasques per continuar amb el desenvolupament del projecte.

La selecció de l'equipament ha estat incremental durant tot el projecte, però la decisió inicial de basar gran part del sistema en NodeMCU va ser un factor clau per simplificar i reduir econòmicament tot el conjunt. Entre l'equipament que s'ha integrat, sens dubte la creació dels circuits on intervenien altres elements que els propis de control, ha estat un dels punts que més dedicació ha requerit tot i ser un projecte del àmbit informàtic. L'aprenentatge de coneixements relacionats amb l'electrònica de control i de potencia han estat necessaris per molts dels dissenys.

A nivell de programació també hi ha hagut una gran evolució i aprenentatge; des del codi inicial de *sockets* TCP/IP fins a l'actual codi que treballa en MQTT. Tots els problemes plantejats inicialment s'han pogut solucionar en les successives versions del codi. Els principals esculls que s'han trobat han sigut relacionats amb la comunicació entre el servidor i els microcontroladors, en disposar d'un únic codi per tots els microcontroladors i en les notificacions originades en els microcontroladors. Totes aquestes dificultats han pogut

solucionar-se gràcies a l'actual codi, on les notificacions i la comunicació mitjançant el protocol MQTT facilita el flux de dades i la lectura de l'adreça MAC que permet diferenciar els microcontroladors. El resultat ha estat que amb un codi de poc més de 400 línies, es poden realitzar diferents tipus de lectures automàtiques (o sota demanda), així com l'activació de diferents E/S per tal de gestionar relés en diferents tipus de sistemes encastats.

En els últims mesos de desenvolupament es va fer la inclusió de Home Assistant, i certament va permetre disposar d'una plataforma que integrés la gestió, el control i l'automatització, però com s'ha definit en el punt 10, ha mancat temps d'execució, ja que el grau d'autonomia buscat no s'ha aconseguit. Serà necessari un temps més per a poder disposar de totes les automatitzacions en producció.

El conjunt aconseguit fins el moment es considera satisfactori, ja que encara que no sigui plenament autònom, brinda una plataforma de gestió de l'habitatge que permet realitzar totes les tasques desitjables manualment.

11.1 Tasques pendents

La principal tasca pendent a realitzar en les pròximes setmanes i mesos, és la creació de tot el conjunt d'automatitzacions que no estan operatives actualment per aconseguir l'autonomia total de l'habitatge. A mesura que es compleixin els objectius definits sorgiran noves automatitzacions que es complementaran fins a disposar d'un sistema plenament autònom. a part, també es consideren importants per tancar el nucli del projecte les següents tasques:

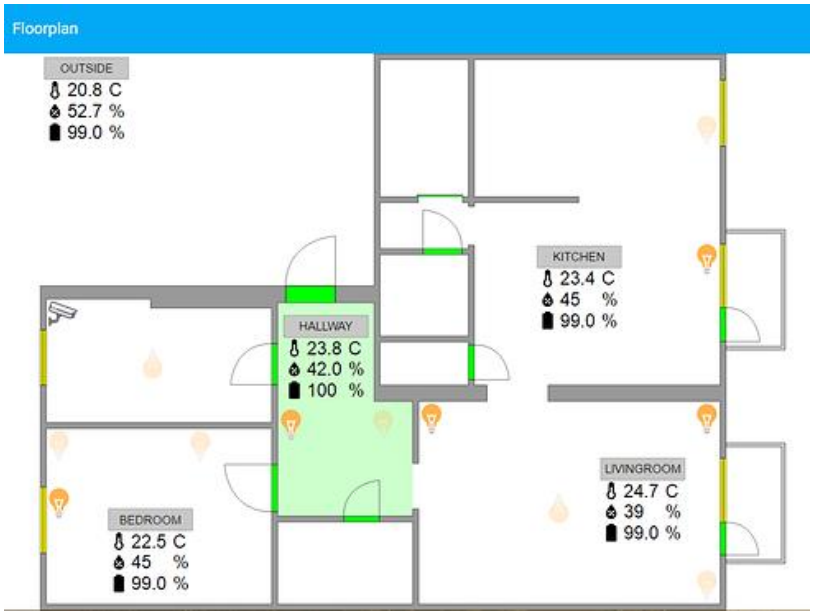
- **Containers:** Ha quedat pendent de realitzar la migració de tot el conjunt de serveis a containers de *docker*[51]. Home Assistant i MQTT, així com la majoria de serveis utilitzats estan orientats ens les seves últimes versions a treballar amb containers; quelcom que simplifica enormement la gestió del serveis, de les copies i de la seguretat.
- **Quadre elèctric:** El desenvolupament del quadre elèctric per monitoritzar les diferents línies requerirà d'una important feina de cablejat que està pendent, així com finalitzar el desenvolupament del codi que mostrarà per la pantalla els consums en temps real de les diferents línies.
- **DHCP i DNS:** Els serveis estan operatius en els dos servidors, un de primari i l'altre secundari, tot i que no s'ha realitzat tota la llista de dispositius amb la seva adreça MAC i una distribució de IP's per tota la xarxa. Actualment hi ha resolució de noms dels dispositius principals, servidors i xarxa.
- **Comunicació per veu:** La interacció entre Home Assistant i els altaveus intel·ligents és possible gràcies als desenvolupadors del nucli de Home Assistant que ho han inclòs. Per aquest motiu, s'han de desenvolupar el conjunt d'automatitzacions necessàries per així fer que les comandes de veu a l'altaveu intel·ligent siguin convertides en accions determinades. L'objectiu és que mitjançant ordres de veu es pugin realitzar totes les accions que es realitzen des de la interfície de control. Fins al moment s'han realitzat proves i el resultat és excel·lent.

[51] **Docker**: Principal projecte de codi obert per la gestió de containers de software.

11.2 Tasques futures

Es plantegen com a futures línies de treball per l'ecosistema creat els següents objectius.

- **Millora de la GUI:** Mitjançant llibreries de tercers es pot utilitzar planells de l'habitatge per interactuar amb el sistema, en comptes del diferents selectors i polsadors que permet Home Assistant originalment. Per exemple, es poden activar les bombetes clicant sobre la icona de la bombeta dins el planell, o visualitzar il·luminada una porta oberta.



Il·lustració 14: Exemple de la funció Floorplan.

- **Plaques fotovoltaiques:** Amb l'objectiu d'instal·lar 3kW d'energia fotovoltaica en els pròxims anys, mitjanant una instal·lació que seria sense bateries, on l'energia no consumida es perdria (actualment el plantejament més senzill i econòmic) es planteja fer una ampliació amb dos objectius; d'una banda l'anàlisi detallat del rendiment de les plaques i d'altra banda emprar aquesta energia de forma òptima, concentrant el consum dels dispositius a les hores de major radiació de forma autònoma. Per exemple, si hi ha excedent d'electricitat, activar la climatització encara que no es compleixin les condicions habituals, de forma que es guanyaria en confort, o limitar el temps de treball del escalfador d'aigua a les hores amb excedent de sol.
- **IoT:** Altres possibles ampliacions, inclourien noves idees per afegir "l'Internet de les Coses" a l'habitatge. A aquesta part es podrien incloure moltes idees; un acceleròmetre per avisar de quant acaba una rentadora, un endoll intel·ligent per encendre la cafetera quant detecti moviment al matí, controlar el sistema de purificació de la piscina segons l'època de l'any, obertura de les portes del garatge mitjançant sensors de proximitat per obrir-la en arribar amb el cotxe, etc.

12 Bibliografia

Tota la documentació del present projecte, s'ha originat principalment amb el material lectiu de l'assignatura de Sistemes Encastats, Editorial UOC, així com la informació recollida de diferents pàgines webs que ha nodrit d'exemples i idees el projecte. Les més destacades, classificades segons la temàtica són les següents:

Sistemes encastats i arduino:

http://www.nodemcu.com/

https://www.hackster.io/techiesms/mqtt-esp8266-12e-nodemcu-157e8b

https://www.instructables.com/id/NodeMCU-MQTT-Basic-Example/

https://github.com/xoseperez/espurna

https://www.luisllamas.es/

https://www.hadevices.com

Protocol MQTT:

https://github.com/weewx/weewx/wiki/mqtt

http://mqtt.org/

Home assistant:

https://www.home-assistant.io/

https://community.home-assistant.io

https://domology.es/

https://domoticaencasa.es/

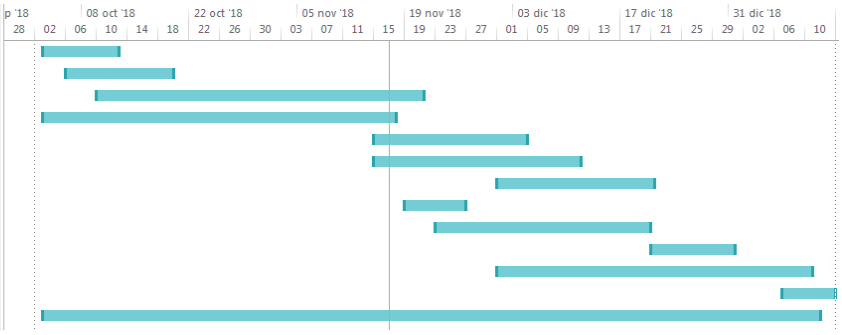
A banda de les mencionades pàgines, també s'ha fet ús de les webs de desenvolupament col·laboratiu de software https://github.com/ i https://stackoverflow.com/ per poder resoldre dubtes i obtenir exemples de programació.

13 Planificació del treball

La planificació del treball inicial es defineix en l'esquema següent. Originalment s'havien definit uns terminis més curts pel desenvolupament del codi i per la integració dels circuits, però al llarg del procés s'han hagut d'ampliar, ja que han sigut les tasques que més dedicació han requerit. Cal destacar, que els períodes definits sovint s'han estès en períodes més llargs, en modificacions secundaries, no en principals.

D'aquesta manera, les tasques de més transcendència, han estat les relacionades amb el desenvolupament de circuits i del codi.

	Nombre de tarea	Duración	Comienzo	Fin
1	Definició d'objectius	8 días	mié 03/10/18	vie 12/10/18
2	Cerca de component	11 días	sáb 06/10/18	sáb 20/10/18
3	Integració components	30,5 días	mié 10/10/18	mié 21/11/18
4	Desenvolupament del codi	34 días	mié 03/10/18	sáb 17/11/18
5	Optimització del codi	14 días	jue 15/11/18	mar 04/12/18
6	Desenvolupament interfície	19 días	jue 15/11/18	mar 11/12/18
7	Programació de automatitzacions	15,5 días	sáb 01/12/18	vie 21/12/18
8	Instalació de sistemes en els servidors	6 días	lun 19/11/18	lun 26/11/18
9	Serveis adicionals i seguretat	20 días	vie 23/11/18	jue 20/12/18
10	Ampliacions	7 días	vie 21/12/18	lun 31/12/18
11	Escritura memòria	30 días	sáb 01/12/18	jue 10/01/19
12	Preparació presentació	6 días	lun 07/01/19	dom 13/01/19
13	Contingencies	73 días	mié 03/10/18	vie 11/01/19



14 Annexes

14.1 Codi

A continuació es pot llegir el codi comentat. Per més detall accedir al punt 5.2 on està detallat el seu funcionament:

```
////////////////////////////////////////////////////////////
/////////// Copyright - OCTAVI PAVON MONTOLIU //////////////
///////////        All Rights Reserved        //////////////
///////////        pavon87@gmail.com          //////////////
///////////           -2016-2018-             //////////////
////////////////////////////////////////////////////////////

#include <ESP8266WiFi.h>
#include <ESP8266mDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <PubSubClient.h>
#include <EEPROM.h>
#include <DHT.h>

////////////////////////////////////////////////////////////
////////////// VARIABLES  DEFINITIONS //////////////////////
////////////////////////////////////////////////////////////

//WIFI
#define ssid "nom-del-wifi"
#define password "password-del-wifi"
WiFiClient espClient;
PubSubClient client(espClient);

//MQTT Server
#define mqtt_server "192.168.1.200"
#define mqtt_port "1883"
#define mqtt_user "nom-del-servidor"
#define mqtt_password "password-del-mqtt "

//DHT SENSORS
#define DHTPIN 13       //REFERS TO GPIO PIN, GPIO13 is D7
#define DHTTYPE DHT22

//GLOBAL VARIABLES AND MQTT TOPICS
long lastAlive = 0;
long lastTemp = 0;
char id[6];
char outTopic[30];
char inTopic[30];
char temperature_topic [20];
char humidity_topic[20];
char drops_topic[20];
char mqtt_status[20];

//RELAY TOPICS DEFINITIONS
// 0-3 SONOFF-4CH, 4-5 CUSTOM RELAYS, 6 SONOFF-1CH
char relay0_topic[20];
char relay1_topic[20];
char relay2_topic[20];
char relay3_topic[20];
char relay4_topic[20];
char relay5_topic[20];
char relay6_topic[20];

//RELAYS GPIO PINS
int relay0 = 15;   //SONOFF 4CH RELAY1
int relay1 = 4;    //SONOFF 4CH RELAY2
int relay2 = 5;    //SONOFF 4CH RELAY3
```

```
int relay3 = 12;   //SONOFF 4CH RELAY4
int relay4 = 2;    //CUSTOM RELAY PIN D4 (GPIO2)
int relay5 = 14;   //CUSTOM RELAY PIN D5 (GPIO14)
int relay6 = 12;   //SONOFF BASIC | SOLID RELAY 40A

//RELAY EEPROM SAVES
bool relay4_state = LOW; //MECHANICAL RELAYS NEGATE OUTPUT
bool relay5_state = LOW;
bool relay6_state = LOW;

////////////////////////////////////////////////////////////
//////////////////// MAIN PROGRAM //////////////////////////
////////////////////////////////////////////////////////////

void setup() {
  Serial.begin(115200);
  EEPROM.begin(512);

  //DEFINE OUTPUT GPIO PORTS
  pinMode(relay0, OUTPUT);
  pinMode(relay1, OUTPUT);
  pinMode(relay2, OUTPUT);
  pinMode(relay3, OUTPUT);
  pinMode(relay4, OUTPUT);
  pinMode(relay5, OUTPUT);
  pinMode(relay6, OUTPUT);

  // LED SONOFF 4CH
  pinMode(13, OUTPUT);
  digitalWrite(13, 1);

  //SET DEFAULT STATE TO RELAYS
  digitalWrite(relay0, 0); //SONOFF 4CH
  digitalWrite(relay1, 0);
  digitalWrite(relay2, 0);
  digitalWrite(relay3, 0);
  digitalWrite(relay4, 0); //MECHANICAL RELAYS NEGATE OUTPUT!
  digitalWrite(relay5, 0);
  digitalWrite(relay6, 0); //SONOFF BASIC || SOLID RELAY 40A


  //READ AND SET LAST RELAYS STATES
  relay4_state = EEPROM.read(4);
  relay5_state = EEPROM.read(5);
  relay6_state = EEPROM.read(6);
  digitalWrite(relay4, relay4_state);
  digitalWrite(relay5, relay5_state);
  digitalWrite(relay6, relay6_state);

  setup_wifi();
  init_id();
  ota_update();

  client.setServer(mqtt_server, 1883);
  client.setCallback(mqtt_callback);
  client.publish(mqtt_status, "1");
}

void loop() {
  ArduinoOTA.handle();

  if (!client.connected()) {
    mqtt_reconnect();
    delay(5000);
  }
  client.loop();

  //PUBLISH STATUS AND CHECK IF IS RAINING EVERY 5.5sec
```

```
  if (millis() - lastAlive > 5500) {
    client.publish(mqtt_status, "1");
    lastAlive = millis();
    raining();
  }
  if (millis() - lastTemp > 60000) {
    lastTemp = millis();
    temphum();
  }
}


// RAIN DETECTOR WITH DROPS SENSOR
void raining() {
  int drops = analogRead(0);
  if ( drops < 950 && drops > 10) {
    client.publish(drops_topic, "1");
    Serial.println(drops);
  } else {
    client.publish(drops_topic, "0");
    Serial.println(drops);
  }
}

// TEMPERATURE AND HUMIDITY WITH DHT22
void temphum() {
  char humidity[10];
  char temperature [10];
  DHT dht(DHTPIN, DHTTYPE);
  sprintf (humidity, "%.2f", dht.readHumidity());
  sprintf (temperature, "%.2f", dht.readTemperature());
  client.publish(humidity_topic, humidity);
  client.publish(temperature_topic, temperature);
  Serial.println(temperature);
  Serial.println(humidity);
}

////////////////////////////////////////////////////////////
////////////////// MQTT FUNCTIONS //////////////////////////
////////////////////////////////////////////////////////////

//MQTT RECEIVER AND CASE SELECT
void mqtt_callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");

  int ncase = ((((char)payload[0]) - '0') * 10) + (((char)payload[1]) - '0');
  switch (ncase) {
    case 0:
      digitalWrite(relay1, 0);
      digitalWrite(relay0, 1);
      client.publish(relay0_topic, "1");
      client.publish(relay1_topic, "0");
      break;
    case 1:
      digitalWrite(relay0, 0);
      digitalWrite(relay1, 1);
      client.publish(relay0_topic, "0");
      client.publish(relay1_topic, "1");
      break;
    case 2:
      digitalWrite(relay3, 0);
      digitalWrite(relay2, 1);
      client.publish(relay2_topic, "1");
      client.publish(relay3_topic, "0");
      break;
    case 3:
```

```
      digitalWrite(relay2, 0);
      digitalWrite(relay3, 1);
      client.publish(relay3_topic, "1");
      client.publish(relay2_topic, "0");
      break;
    case 4:
      digitalWrite(relay4, 0);
      client.publish(relay4_topic, "1");
      EEPROM.write(4, 0);
      break;
    case 5:
      digitalWrite(relay5, 0);
      client.publish(relay5_topic, "1");
      EEPROM.write(5, 0);
      break;
    case 6:
      digitalWrite(relay6, 0);
      client.publish(relay6_topic, "1");
      EEPROM.write(6, 0);
      break;
    case 7:
      temphum();
      break;
    case 8:
      raining();
      break;
    case 9:
      break;
    case 10:
      break;
    case 11:
      break;
    case 12:
      digitalWrite(relay0, 0);
      digitalWrite(relay1, 0);
      client.publish(relay0_topic, "0");
      client.publish(relay1_topic, "0");
      break;
    case 13:
      digitalWrite(relay2, 0);
      digitalWrite(relay3, 0);
      client.publish(relay2_topic, "0");
      client.publish(relay3_topic, "0");
      break;
    case 14:
      digitalWrite(relay4, 1);
      client.publish(relay4_topic, "0");
      EEPROM.write(4, 1);
      break;
    case 15:
      digitalWrite(relay5, 1);
      client.publish(relay5_topic, "0");
      EEPROM.write(5, 1);
      break;
    case 16:
      digitalWrite(relay6, 1);
      client.publish(relay6_topic, "0");
      EEPROM.write(6, 1);
      break;
  }
  EEPROM.commit();
}

//MQTT TOPIC COMPOSITION WITH MAC ADDRESS
void init_id() {
  byte mac[6];
  WiFi.macAddress(mac);
  sprintf(id, "%02X%02X%02X\0", mac[3], mac[4], mac[5]);
```

```
  strcpy(outTopic, id);
  strcat(outTopic, "/out");
  strcpy(inTopic, id);
  strcat(inTopic, "/in");
  strcpy(mqtt_status, id);
  strcat(mqtt_status, "/status");
  strcpy(relay0_topic, outTopic);
  strcat(relay0_topic, "/relay0");
  strcpy(relay1_topic, outTopic);
  strcat(relay1_topic, "/relay1");
  strcpy(relay2_topic, outTopic);
  strcat(relay2_topic, "/relay2");
  strcpy(relay3_topic, outTopic);
  strcat(relay3_topic, "/relay3");
  strcpy(relay4_topic, outTopic);
  strcat(relay4_topic, "/relay4");
  strcpy(relay5_topic, outTopic);
  strcat(relay5_topic, "/relay5");
  strcpy(relay6_topic, outTopic);
  strcat(relay6_topic, "/relay6");
  strcpy(drops_topic, outTopic);
  strcat(drops_topic, "/raining");
  strcpy(humidity_topic, outTopic);
  strcat(humidity_topic, "/hum");
  strcpy(temperature_topic, outTopic);
  strcat(temperature_topic, "/temp");

  Serial.print("ID: ");
  Serial.println(id);
  Serial.print("Topic In: ");
  Serial.println(inTopic);
  Serial.print("Topic Out: ");
  Serial.println(outTopic);
}

//MQTT RECONNECTION IN CASE OF FAULT
void mqtt_reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect(id, mqtt_user, mqtt_password )) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish(outTopic, "Board booted");
      // ... and resubscribe
      client.subscribe(inTopic);
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      for (int i = 0; i < 5000; i++) {
        delay(1);
      }
    }
  }
}

/////////////////////////////////////////////////////////
////////////////////// WIFI CONFIG //////////////////////
/////////////////////////////////////////////////////////

void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
```

```
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    for (int i = 0; i < 500; i++) {
      delay(1);
    }
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.print("WiFi Mode = ");
  Serial.println(WiFi.getMode());
  Serial.print("WiFi PhyMode = ");
  Serial.println(WiFi.getPhyMode());
  Serial.print("STA MAC: ");
  Serial.println(WiFi.macAddress());
  Serial.print("STA Status: ");
  Serial.println(WiFi.status());
  Serial.print("STA RSSI: ");
  Serial.println(WiFi.RSSI());
  Serial.print("STA psk: ");
  Serial.println(WiFi.psk());
  Serial.print("STA BSSID: ");
  Serial.println(WiFi.BSSIDstr());
  Serial.print("STA Chanel: ");
  Serial.println(WiFi.channel());
  Serial.println(WiFi.localIP());
  Serial.println("");
}

//OTA UPDATES OF CODE VIA WIFI
void ota_update() {
  ArduinoOTA.setHostname(id);
  ArduinoOTA.setPassword("admin");
  ArduinoOTA.onStart([]() {
  });
  ArduinoOTA.onEnd([]() {
    Serial.println("\nEnd");
  });
  ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {
    Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
  });
  ArduinoOTA.onError([](ota_error_t error) {
    Serial.printf("Error[%u]: ", error);
    if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
    else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed");
    else if (error == OTA_CONNECT_ERROR) Serial.println("Connect Failed");
    else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive Failed");
    else if (error == OTA_END_ERROR) Serial.println("End Failed");
  });
  ArduinoOTA.begin();
}
```

14.2 Índex de figures, taules i esquemes