Treball de Fi de Grau

## Grau en Enginyeria en Tecnologies Industrials

# Disseny d'un sistema domòtic per a un habitatge unifamiliar

**MEMÒRIA**


**Autor:** Marc Borrell Roig
**Director:** Lluís Solano Albajes


**Convocatòria:** Gener de 2018




Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona


# Resum

Aquest projecte es basa a dissenyar, construir i validar un sistema domòtic que permeti als membres d'una casa familiar automatitzar i monitorar processos domèstics o condicions ambientals per millorar la qualitat de vida. Per tal de tenir una experiència completa, es desenvolupa tant hardware com software.

Per a fer-ho, s'utilitza l'entorn de programació de Node-RED, la llibreria Bootstrap 4, el protocol de comunicació MQTT, la base de dades InfluxDB, la llibreria de JavaScript Chart.js, el sistema operatiu específic Mongoose OS, el microcontrolador ESP8266 i finalment la plataforma en línia Dialogflow. Amb totes aquestes eines i mètodes, es dissenya el programa principal i els perifèrics.

El resultat d'aquest treball és la implementació real del sistema domòtic descrit anteriorment, en una habitació. Principalment es disposa d'un servidor, sensors, interruptors i diversos relés, tots testejats i validats fins a la final implementació.

La conclusió principal d'aquest projecte és l'assoliment dels objectius marcats a l'inici del període de treball, tant en l'àmbit teòric com en el pràctic. Es disposa d'una habitació completament domotitzada, amb control sobre aparells elèctrics i monitoratge de sensors utilitzant múltiples interfícies d'usuari.

# Sumari

**RESUM** 1

**SUMARI** 3

**SUMARI DE TAULES I FIGURES** 6

**1. GLOSSARI** 9

**2. PREFACI** 11

2.1. Origen del projecte 11
2.2. Motivació 11
2.3. Requeriments previs 11

**3. INTRODUCCIÓ** 13

3.1. Objectius del projecte 13
3.2. Abast del projecte 13

**4. ANÀLISI D'ANTECEDENTS** 15

**5. ESTUDI DE VIABILITAT** 16

5.1. Viabilitat tècnica 16
5.2. Viabilitat econòmica 16
5.3. Viabilitat ambiental 16

**6. ANÀLISI DEL PROBLEMA** 17

6.1. Accions a realitzar 17
6.2. Dades a monitorar 18

**7. FUNCIONALITATS** 19

**8. PROPOSTA DE SOLUCIÓ** 21

**9. ESTUDI DE MERCAT** 22

9.1. Controladors per a cases domòtiques 22
9.2. Estudi d'eines utilitzades 23
9.2.1. Frontend 23
9.2.2. Servidor 24
9.2.3. Comunicacions 26
9.2.4. Base de dades 27
9.2.5. Hardware servidor 27
9.2.6. Perifèrics 28

9.2.7. Convertir veu a text i viceversa ....29
9.2.8. Assistent virtual ....29

**10. EINES PRINCIPALS UTILITZADES 31**
10.1. Node-RED ....31
10.2. Bootstrap 4 ....31
10.3. MQTT ....31
10.4. InfluxDB ....32
10.5. Chart.js ....32
10.6. ESP8266 ....32
10.7. Mongoose OS ....33
10.8. Dialogflow ....33

**11. DISSENY DE LA SOLUCIÓ 35**
11.1. Disseny general ....35
11.2. Programa principal ....37
11.2.1. Servidor interfície web ....37
11.2.2. Comunicacions MQTT ....39
11.2.3. Comunicacions Websockets ....40
11.2.4. Base de dades ....43
11.2.5. Pins GPIO ....45
11.2.6. Node Propi ....52
11.2.7. Interfície de prototipatge ....53
11.2.8. Control per veu ....55
11.2.9. API Servidors de música ....57
11.2.10. Seguretat ....58
11.2.11. Actualitzacions ....59
11.3. Perifèrics ....60
11.3.1. Programa principal ....60
11.3.2. Comunicació ....61

**12. INTERFÍCIE D'USUARI 62**
12.1. Disseny general ....62
12.2. Base ....63
12.2.1. Descripció ....63
12.2.2. Codi ....64
12.3. Inici ....65
12.3.1. Descripció ....65
12.3.2. Codi ....65
12.4. Interruptors ....67

12.4.1. Descripció ..... 67
12.4.2. Codi ..... 67
12.5. Sensors ..... 69
12.5.1. Descripció ..... 69
12.5.2. Codi ..... 69
12.6. Comunicació amb Websockets ..... 70
12.7. Comunicació MQTT ..... 71
12.8. Optimització ..... 72
**13. IMPLEMENTACIÓ ..... 73**
13.1. Servidor ..... 73
13.2. Relés ..... 74
13.3. Sensors ..... 75
13.4. Interruptors ..... 76
**14. TEST I VALIDACIÓ ..... 78**
14.1. Errors i resolució ..... 78
14.2. Possibles millores ..... 79
**15. PLANIFICACIÓ ..... 80**
**16. PRESSUPOST ..... 81**
**17. IMPACTE AMBIENTAL ..... 83**
**CONCLUSIONS ..... 84**
**AGRAÏMENTS ..... 85**
**BIBLIOGRAFIA ..... 86**
Referències bibliogràfiques ..... 86

# Sumari de taules i figures

## TAULES

Taula 1. Glossari ..... 10

Taula 2. Accions o tasques ..... 17

Taula 3. Dades monitorades ..... 18

Taula 4.Servidors Python ..... 25

Taula 5. Servidors ..... 26

Taula 6. Opcions com a hardware a escollir ..... 28

Taula 7. Especificacions de la placa ESP8266 i MRK1000 ..... 28

Taula 8. Característiques de diferents plataformes existents al mercat ..... 30

Taula 9. Acció de les persianes ..... 48

Taula 10. Execució al rebre un “ON event” ..... 50

Taula 11. Execució al rebre un “Switches/llum (p/s) event” ..... 50

Taula 12. Errors i resolució ..... 78

Taula 13. Pressupost ..... 82

Taula 14. Càlcul de l’emissió equivalent de $CO_2$ ..... 83

## FIGURES

Figura 1. Previsió del nombre de dispositius connectats en cases intel·ligents, des de 2015 fins a 2018 (en milions) [4] ..... 22

Figura 2. Microcontrolador ESP8266. Font pròpia ..... 32

Figura 3. Sonoff T1 ..... 33

Figura 4. Placa ESP8266 DC Relay Board ..... 33

Figura 5. Esquema genèric de tot el sistema. Font pròpia..... 36

Figura 6. Flow de la interfície web. Font pròpia..... 38

Figura 7. Configuració del node"http in" i "file in". Font pròpia..... 38

Figura 8. Creació i descripció d'un subflow. Font pròpia..... 39

Figura 9. Configuració del broker mitjançant la llibreria Mosca..... 40

Figura 10. Communicació WebSockets. Font pròpia..... 40

Figura 11. Configuració del node "switch". Font pròpia..... 41

Figura 12. Configuració del node "change". Font pròpia..... 41

Figura 13. Flow de la base de dades. Font pròpia..... 43

Figura 14. Entrada a través de comunicació MQTT. Font pròpia..... 44

Figura 15. Entrada a través del node "link in". Font pròpia..... 44

Figura 16. Exemple format acceptat per a la base de dades. Font pròpia..... 45

Figura 17. Edició del node influxdb out..... 45

Figura 18. Flow dels pins GPIO. Font pròpia..... 46

Figura 19. Inversió del senyal amb un subflow. Font pròpia..... 47

Figura 20. Node rpi-gpio out. Font pròpia..... 47

Figura 21. Configuració dels relés de la persiana. Font pròpia..... 48

Figura 22. Configuració del mode automàtic. Font pròpia..... 49

Figura 23. Configuració del node rpi-dht22. Font pròpia..... 51

Figura 24. Funció pròpia de l'arrodoniment decimal de la temperatura..... 51

Figura 25. Pantalla de configuració del node propi (progress bar)..... 52

Figura 26. Flow de la interfície de prototipatge. Font pròpia..... 53

Figura 27. Botons de la interfície de prototipatge. Font pròpia..... 54

Figura 28. Configuració per a obtenir la interfície de temperatura i humitat. Font pròpia..... 54

Figura 29. Gràfic de la càrrega de processador del servidor, temperatura i humitat. Font pròpia ..... 55

Figura 30. Flow del control per veu. Font pròpia ..... 56

Figura 31. Flow del control del sistema de música. ..... 57

Figura 32. Exemple de creació d'un password hash. Font pròpia ..... 58

Figura 33. Mostra de l'accés amb usuari. Font pròpia ..... 58

Figura 34. A l'esquerre, funcionalitat del botó Deploy; a la dreta, vista de la configuració Manage Palette, on es veu l'estat de totes les llibreries. Font pròpia ..... 59

Figura 35. Procés dut a terme al segon bloc de l'algoritme dels interruptors. Font pròpia.... 61

Figura 36. Esquema general de la interfície d'usuari. Font pròpia..... 62

Figura 37. Detall de la barra de navegació. Font pròpia ..... 63

Figura 38. Visió dels elements <head>, <title> i <meta> del codi HTML. Font pròpia..... 64

Figura 39. Codi exemple d'una carta. Font pròpia..... 66

Figura 40. Muntatge del prototip: (1) placa de relés, (2) sensor de posició, (3) sensor de temperatura i humitat, (4) Raspberry Pi2. Font pròpia..... 73

Figura 41. Esquema elèctric de la connexió dels relés. Font pròpia..... 74

Figura 42. Placa de relés en funcionament en el sistema final. Font pròpia..... 75

Figura 43. Esquema de la configuració elèctrica dels sensors. Font pròpia ..... 75

Figura 44. Senors implementats en el sistema final. Font pròpia ..... 76

Figura 45. Muntatge de la placa ESP8266 DC Realy Board alimentada per una pila de 9V, per a implementar l'interruptor. Font pròpia..... 77

Figura 46. Esquema elèctric de la implementació de l'interruptor. ..... 77

Figura 47. Planificació del projecte. Font pròpia..... 80

# 1. Glossari

| | |
|---|---|
| API | Abreviació de *Application Programming Interface.* És una interfície que especifica com diferents components de programes informàtics haurien d'interaccionar. |
| Big Data | És un terme que fa referència a l'activitat de recollir moltes dades per extreure'n patrons o informació d'utilitat |
| broker MQTT | És un gestor de missatges que segueixen el protocol MQTT i els distribueix als clients connectats. |
| Chatbots | Programa informàtic amb el qual és possible mantenir una conversa, tant per si es vol demanar informació com si es vol realitzar una acció. |
| CoAP | Abreviació de *Constrained Application Protocol.* És un protocol de la capa d'aplicació d'internet per a dispositius amb recursos restringits. CoAP permet que aquests es puguin comunicar amb qualsevol node d'internet i està dirigit a l'*Internet of Things* (IoT). |
| CSS | Abreviació de *Cascading Style Sheets.* És un mecanisme que descriu com es mostrarà un document a la pantalla, com s'imprimirà o inclús com serà pronunciada la informació present en un document a través d'un dispositiu de lectura. Aquesta forma de descripció d'estils ofereix als desenvolupadors el control total sobre l'estil i el format dels seus documents. |
| Fase | Terminal d'un circuit de corrent elèctric altern. |
| Flows | En terminologia del software Node-RED, un flow engloba diferents nodes i les connexions que hi ha entre ells. |
| front-end | Terme utilitzat a la informàtica per descriure el conjunt de tecnologies que es fan servir per mostrar una pàgina web al cantó del navegador. |
| HTML | Abreviació de *HyperText Markup Language.* És un llenguatge informàtic dissenyat per a estructurar textos i relacionar-los en forma d'hipertext. Gràcies a Internet i als navegadors web, s'ha convertit en un dels formats més populars que existeixen per a la construcció de documents per a la web. |

| | |
|---|---|
| IoT | Abreviació de *Internet of Things* es refereix, en termes d'informàtica, a una xarxa d'objectes de la vida quotidiana interconnectats. |
| MQTT | Abreviació de *Message Queue Telemetry Transport.* És un protocol utilitzat per la comunicació *Machine-to-machine (M2M)* en el IoT. Està orientat a la comunicació de sensors pel fet que consumeix molt poc ample de banda i pot ser utilitzat en la majoria dels dispositius empitrats amb pocs recursos. |
| neutre | Terminal d'un circuit de corrent elèctric altern. En un sistema elèctric significa que no té càrrega elèctrica total, perquè la càrrega negativa es contrarestà amb la positiva. |
| Pins GPIO | Abreviació de pins General Purpose Input/Output. És un tipus de port que ofereix a una placa electrònica la possibilitat de comunicar-se amb altres circuits mitjançant senyals digitals o analògiques. |
| protocol TCP/IP | És un protocol utilitzat en xarxes, el qual descriu un conjunt de guies generals d'operació per a permetre que un equip pugui comunicar-se en una xarxa. Preveu connectivitat d'extrem a extrem, especificant com les dades haurien de ser formatejades, direccionades, transmeses, enrutades i rebudes pel destinatari. |

*Taula 1. Glossari*

# 2. Prefaci

## 2.1. Origen del projecte

Cada dia és més comú que maquines i robots assumeixin tasques que abans realitzaven les persones. Ja sigui en fabriques on es requereixen treballs manuals repetitius o en llars convencionals on els electrodomèstics cobreixen rutines diàries.

També és creixent la recollida de dades en molts sectors diferents, com per exemple els comentats prèviament entre d'altres, per a trobar patrons i optimitzar recursos. De fet, termes com el "big data" o "IoT" són termes que cada dia són més habituals en la societat i plasmen aquesta nova necessitat de controlar i automatitzar tots els sistemes que ens envolten [1].

## 2.2. Motivació

Dins d'aquest ecosistema creixent d'automatitzacions i monitoratges, s'ha pensat que els sistemes domèstics tot just estan començant a ser explorats per les empreses del sector. S'ha cregut doncs que hi ha molt marge per explorar i per desenvolupar nous sistemes.

Per altra banda, en l'última dècada s'han fet molt populars les creacions d'empreses d'emprenedors mitjançant el mètode de "crowdfunding". Aquest fet ha propiciat que, a l'hora d'orientar el projecte, s'hagin agafat com a referents empreses que van començar així però que a poc a poc s'han fet un lloc en el mercat internacional.

La passió per l'electrònica i la informàtica són els dos principals desencadenants d'aquest projecte, així com la creença que la complexitat que comporta és al mateix temps una font d'enriquiment personal com a enginyer.

## 2.3. Requeriments previs

Per dur a terme aquest projecte, són necessaris coneixements avançats de programació en algun llenguatge d'alt nivell (JavaScript, Python, C++), preferentment JavaScript. També cal que el desenvolupador se senti còmode treballant en un entorn Linux.

A més a més, també és important tenir noció de sistemes elèctrics i electrònics, ja que en el projecte es desenvolupa tant hardware com software. Pel que fa al hardware, es requereixen un seguit d'eines que s'aniran detallant a la memòria, les quals també són imprescindibles per a qualsevol empresa o individual involucrat en temes elèctrics.

# 3. Introducció

El projecte de disseny i la programació d'un sistema domòtic per a un habitatge unifamiliar s'ha desenvolupat durant el segon quadrimestre de 2017. En aquest, es recull i s'explica detalladament tot el que és necessari per crear un sistema domòtic i permet adquirir prou coneixement per continuar la tasca de desenvolupament.

El disseny d'aquest sistema ha sigut iteratiu. En alguns casos es pot observar que l'eina seleccionada no resulta adient i s'exposa quins problemes s'han trobat i com s'han resolt.

Al llarg de tota la memòria es mostren en molts casos només els resultats, per tal de presentar claredat i precisió. No obstant això, en alguns casos es desenvolupa més en detall el procés dut a terme i en els annexos s'inclou tota la informació addicional no exposada a la memòria.

## 3.1. Objectius del projecte

Els objectius del projecte són: dissenyar, construir i validar un sistema domòtic que permeti als membres d'una casa familiar automatitzar i monitorar processos domèstics o condicions ambientals per millorar la qualitat de vida.

Per altra banda, s'ha volgut crear una experiència completa, on diferents usuaris puguin provar realment el sistema a través de les seves interfícies i llegir-ne els resultats. Aquest és el motiu pel qual l'abast del projecte inclou el desenvolupament d'ambdós hardware i software.

Altres aspectes generals a assolir són els següents:

- L'usuari ha de poder interactuar amb el sistema a través de dispositius quotidians com mòbils, ordinadors o tauletes.

- El sistema s'ha de poder implementar en un ecosistema ja creat prèviament, per tant, implica que sigui modular, adaptatiu i configurable.

- Sistema econòmic, senzill i robust

## 3.2. Abast del projecte

Aquest projecte s'emmarca dins del quadrimestre de tardor de 2017-2018, és a dir, es desenvolupa en un període total d'aproximadament quatre mesos. L'abast d'aquest s'ha anat ampliant a mesura que s'ha anat desenvolupant, però sempre dins dels següents límits inicialment establerts:

- Pel que fa al software del sistema, s'ha fet un equilibri entre codi propi i llibreries externes. S'ha prioritzat el codi propi per no comprometre els objectius del projecte però, s'ha complementat amb llibreries externes que ja proporcionen les funcionalitats desitjades per complir les dates d'entrega.

- S'ha descartat fer un hardware propi a escala d'esquemes electrònics i disseny de PCB a causa de la limitació de temps. Malgrat això, s'ha decidit implementar el sistema en un habitatge real combinant i modificant convenientment sistemes comercials.

- Per l'optimització del sistema, tant en hardware com en software, s'ha establert un nivell bàsic. Això si, sempre el necessari per a no perjudicar l'experiència de l'usuari.

Aquests marges de treball han permès focalitzar els esforços i recursos, així com en cap moment han limitat els objectius inicials del projecte. S'ha volgut posar èmfasi en la *costumització* de tots els subsistemes perquè treballin com un de sol i per tal de complir tots els requeriments.

# 4. Anàlisi d'antecedents

La primera tecnologia domòtica va aparèixer l'any 1975 amb l'X10, un estàndard obert per a les comunicacions entre aparells electrònics, desenvolupat per *Pico Electronics* a Escòcia [2]. Des d'aleshores la domòtica ha evolucionat, apareixent un seguit de tecnologies noves que a poc a poc, ha anat prenent un lloc entre les cases.

En món de la domòtica, a més a més de grans empreses com Google, Apple o Amazon, recentment hi ha hagut una basant molt forta d'individus d'independents anomenats "Makers" que han desenvolupat i aplicat sistemes propis. Aquests darrers han permès fer que tota aquesta tecnologia sigui molt més accessible.

De totes maneres, en el moment d'escriptura d'aquest treball, no hi ha cap sistema integral que es vengui al mercat Espanyol per controlar habitatges com s'exposa en el punt 9, estudi de mercat. Aquest fet obre una finestra a nous projectes i empreses que siguin capaços d'aportar solucions noves.

# 5. Estudi de viabilitat

## 5.1. Viabilitat tècnica

Com s'ha comentat a l'anàlisi d'antecedents, per separat ja existeixen tecnologies que podrien fer algunes de les funcions desitjades. En aquest treball però, el que s'ha buscat és crear un paquet unificat i complet.

Per a fer-ho, tot i la necessitat de desenvolupar un codi propi i una part del hardware, en tots els casos s'ha vist que tècnicament el projecte és viable, ja que no requereix peces ni materials inexistents i hi ha suport en alguns àmbits.

## 5.2. Viabilitat econòmica

Com s'ha explicat en els objectius, un d'aquests és que el projecte sigui molt econòmic. Seguint aquesta línia, els programes, llibreries i altres softwares utilitzats són tots gratuïts i permeten l'ús lucratiu. El cost afegit que podria aparèixer en el projecte és el d'algunes *API's,* que poden arribar a ser de pagament depenent de l'ús que se'n faci. En els casos exposats en aquest treball però, tots els serveis són gratuïts.

Pel que fa a hardware, cal tenir en compte que sí que s'ha requerit comprar algunes plaques electròniques. Els detalls es poden trobar al punt 15, pressupost.

En total, es pot observar que el projecte és econòmicament viable dins l'abast esmentat prèviament.

## 5.3. Viabilitat ambiental

La viabilitat ambiental no és un aspecte restrictiu en aquest projecte, ja que es tracta de la domototzació d'un sol habitatge, la realització de la qual s'ha fet a través de codi propi i poques plaques comercials han estat utilitzades, les quals compleixen una normativa estricte en quant al seu ús i reciclatge.

A l'apartat d'impacte ambiental, punt 17, es detalla l'impacte ambiental final del projecte.

# 6. Anàlisi del problema

Per tal de conèixer a fons el problema a resoldre s'ha fet un estudi de les tasques que es poden automatitzar així com les dades que es poden monitorar en un habitatge. Estudiant aquests aspectes es podran especificar millor les funcions del sistema i el disseny de la interfície.

## 6.1. Accions a realitzar

En primer lloc, s'ha estudiat quin tipus d'accions i tasques són més comunes a la llar. Per a fer-ho, s'han pres dades de forma manual, en format d'una taula i al llarg de diversos dies diferents. S'ha buscat una mostra representativa de dies, en aquest cas, dies laborables i festius, 10 en ambdós casos. Els resultats es poden veure a la taula 2.

| Acció o tasca | Dies laborables (nº de vegades per usuari) | Dies festius (nº de vegades per usuari) |
|---|---|---|
| **Encendre la llum principal** | 110 | 200 |
| **Encendre la llum secundària** | 31 | 35 |
| **Carregar aparells electrònics** | 32 | 23 |
| **Encendre la llum d'escriptori** | 20 | 9 |
| **Posar música de fons** | 24 | 33 |
| **Obrir/ tancar la persiana** | 20 | 28 |
| **Posar una rentadora/secadora** | 2 | 2 |
| **Posar rentaplats (per habitatge)** | 2 | 2 |
| **Encendre aire condicionat** | 10 | 14 |

*Taula 2. Accions o tasques*

## 6.2. Dades a monitorar

S'ha buscat quines dades són interessants per l'usuari i aquestes han estat les escollides per a monitorar. Aquestes mesures poden ser, per exemple, dades ambientals de l'entorn o dades que el mateix sistema pot arribar a crear.

L'objectiu de recollir i mostrar aquestes dades és que els consumidors puguin prendre decisions encertades o més contrastades, pel que fa a l'entorn domèstic. D'aquestes s'espera que el mateix usuari en tregui conclusions i, si s'escau, li permeti optimitzar els seus patrons de conducta.

Les dades finalment monitorades són les que es mostren a la taula 3.

| Mesura | Unitat | Rang típic |
|---|---|---|
| **Temperatura** | Graus Celsius [ºC] | 0-35 |
| **Humitat** | [%] | 0-100 |
| **Activitat** | Nº de vegades | 0-100 |
| **So** | Binària | 0-1 |

*Taula 3. Dades monitorades*

# 7. Funcionalitats

D'acord amb els resultats que s'han obtingut a través de l'anàlisi del problema, s'han detallat totes les funcionalitats que abraça el projecte.

**La primera funcionalitat** i la més important de totes, és encendre i apagar llums i dispositius electrònics, donada l'alta puntuació que s'ha recollit a la taula d'accions. Aquesta funcionalitat s'ha de poder realitzar des dels interruptors que ja té la casa, per complir el requeriment imposat en el segon dels objectius del projecte, comentats en el punt 3.1.

També s'ha de poder modificar l'estat dels aparells des de qualsevol plataforma digital. Cal, per tant, que el sistema digital pugui saber en quin estat es troba cada aparell abans de ser actuat per tal de no crear confusió. Això implica que, en cas d'estar oberta més d'una interfície alhora, l'estat dels botons s'ha de mostrar en temps real.

Addicionalment s'ha d'oferir un control per veu senzill per fer el sistema més accessible a les persones amb discapacitats.

**La segona funcionalitat** és la de monitorar les dades més importants que s'han trobat a l'apartat 6.2. Aquestes s'han de poder veure en temps real o de forma gràfica en un determinat període de temps.

Per tal de crear una experiència més agradable per l'usuari, els gràfics han de poder ser ajustats a la mida de la pantalla, s'han de poder triar diferents escales de temps i diferents temps de refresc per cadascun. A més, s'ha de poder escollir quins gràfics es volen veure i quins no.

**La tercera funcionalitat,** ja que es tracta d'un sistema domòtic complet, és disposar d'una zona on s'organitzi i s'expliqui quines funcionalitats proporciona el sistema i on trobar-les. Ha de ser un punt d'inici o de nexe entre activitats diferents.

Aquest punt permetrà que, en un futur i a mesura que el sistema vagi creixent, es puguin trobar fàcilment les noves funcionalitats.

**La quarta funcionalitat** és de caràcter més general i segueix la línia de la tercera. En ser un sistema domòtic, s'espera que els màxims processos possibles siguin automatitzats. Aquesta funcionalitat deriva doncs d'una barreja entre les dues primeres, ja que en realitzar-se accions al llarg del temps, s'han de trobar patrons de conducta que es puguin substituir per automatismes.

Aquestes són totes les funcionalitats principals. De totes maneres, durant el desenvolupament del treball, s'han volgut afegir algunes funcionalitats addicionals que fan el sistema més

complet. Aquestes noves funcions s'aniran detallant al llarg del document però cal tenir present que en tot moment són secundàries amb l'únic objectiu de complementar l'experiència.

# 8. Proposta de solució

A partir de les funcionalitats, s'ha plantejat com abordar el problema i trobar una solució. En aquest i els següents apartats s'ha detallat el procés dut a terme. En aquest primer punt en concret, es fa un esquema tècnic detallat però genèric de la possible solució. Més endavant, amb l'estudi de mercat (punt 9) i finalment el disseny de la solució (punt 11), s'acabaran de concretar les tecnologies més adequades i com s'utilitzen.

Primer de tot, el sistema ha de ser dirigit per un servidor central on es recullen totes les dades i s'actua en conseqüència. Aquest servidor ha de funcionar dins d'un petit ordinador o microprocessador, el més econòmic possible però prou potent per mantenir el rendiment i la fluïdesa desitjada.

L'ordinador ha de tenir un sistema operatiu lleuger i que sigui fàcil d'implementar. Dins d'aquest sistema, s'hi ha d'instal·lar tots els paquets i llibreries necessàries perquè el codi que s'ha desenvolupat funcioni. El codi propi ha de ser l'encarregat de complir totes les funcionalitats requerides i que es compleixin tots els objectius marcats.

Per satisfer les necessitats addicionals dels usuaris, el servidor ha de ser capaç de connectar amb serveis externs i integrar-los en el mateix sistema, per proporcionar una experiència única.

Addicionalment, aquest ordinador ha de rebre senyals d'entrada de tots els perifèrics que hi hagi instal·lats per la casa mitjançant la comunicació adient. Aquesta comunicació entre perifèrics ha de ser lleugera, robusta i rapida.

Els perifèrics cal que estiguin basats en un hardware fàcil d'implementar, compacte i econòmic. No es requereix molta potència computacional, ja que és el servidor l'encarregat de fer càlculs pesats. A més a més, cal que tinguin possibilitat de comunicació sense fils per facilitar la instal·lació.

Per últim, tant els perifèrics com el servidor, s'han de poder programar remotament, sense causar aturades llargues al sistema i no perjudicar l'ús dels usuaris.

# 9. Estudi de mercat

## 9.1. Controladors per a cases domòtiques

Al mercat tecnològic hi ha una gran varietat de productes orientats a cases intel·ligents o domòtiques. És un sector que està en expansió vertiginosa [3]. A la figura 1 es mostra un gràfic mundial d'aquest creixement.

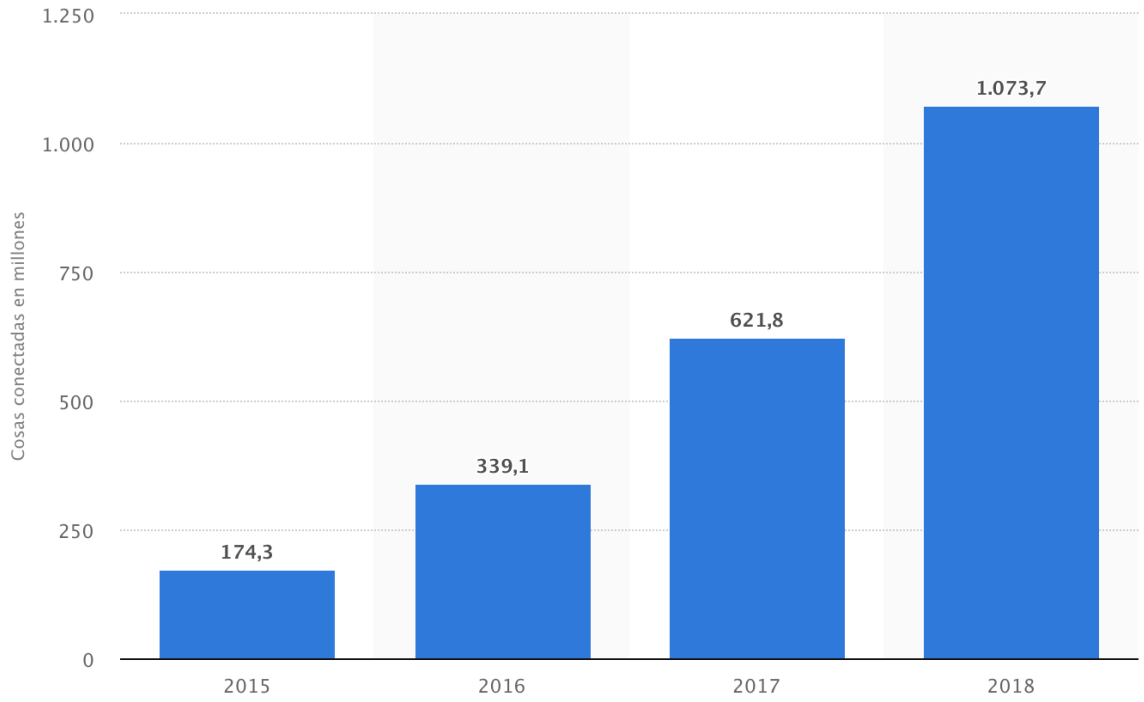



*Figura 1. Previsió del nombre de dispositius connectats en cases intel·ligents, des de 2015 fins a 2018 (en milions) [4]*

En concret, s'ha realitzat l'estudi de mercat a la regió d'Espanya, ja que és on es desenvolupa el producte. Els dispositius més comercialitzats a la regió, són els següents:

- **Philips HUE** [5] és un dels productes més venuts a l'estat. El kit està compost per un controlador central i un joc de bombetes que es poden controlar a distància. Això permet controlar les llums que tinguin aquesta bombeta des del mòbil. També es pot canviar el color de cada bombeta individualment.

Aquesta solució manca d'integració amb els sistemes de control ja existents a la casa, com per exemple els interruptors de paret. A més a més les funcionalitats només es limiten a activar o desactivar les llums manualment i no permet cap automatisme.

- **Belkin WeMo** [6] és un producte semblant al de Philips, en el sentit que permet controlar els llums que tinguin bombetes específiques de WeMo. A més a més també ofereixen endolls controlables des del mòbil.

  Les mancances però són les mateixes que en el cas anterior. El control només és possible des del seu aplicatiu i no et permet integrar-ho amb altres sistemes com sensors de moviment o de llum. Per tant, la seva funcionalitat queda limitada.

- **Trust AWST-8802** [7] també permet controlar els llums de la casa. Per fer-ho et proporcionen un interruptor sense fils amb bateria i un kit per posar a cada llum.

  El sistema no es pot controlar des del mòbil sinó que requereix botons o comandaments a distància. Tampoc es pot automatitzar.

A Estàts Units, s'estan popularitzant controladors com el *Google Home* de Google, l'*Amazon Echo* d'Amazon o *l'Apple HomePod* d'Apple. Aquests disposen de la capacitat de gestionar les llums amb control per veu, únicament en Angles. Per això encara no es venen Espanya.

Aquests controladors, tot i ser els més propers en termes de funcions i objectius que s'han proposat en aquest projecte, no acaben de complir amb totes les expectatives i requeriments plantejats. A més de tenir la interfície majoritàriament per veu, tampoc ofereixen una manera de controlar les llums des d'un interruptor convencional. Tampoc tenen la possibilitat de recollir dades de sensors externs ni graficar-les.

## 9.2. Estudi d'eines utilitzades

### 9.2.1. Frontend

S'ha fet un estudi i comparació de les tecnologies més utilitzades per crear interfícies web. L'objectiu és maximitzar la funcionalitat i l'adaptabilitat, considerant també la importància del disseny i l'estètica.

També s'han tingut en compte factors com el manteniment a llarg termini i la dificultat d'implementació.

Els llenguatges acceptats pels navegadors actuals són HyperText Markup Language (HTML), Cascading Style Sheets (CSS) i JavaScript (JS). Per desenvolupar el codi de manera més rapida, s'ha valorat l'opció d'utilitzar llibreries que facilitin el procés d'escriptura. Aquestes

llibreries ofereixen components que ja porten un codi CSS o JS associat.

L'avantatges d'usar llibreries de CSS i JS són:

- Fàcil manteniment del codi
- Codi més ven organitzat i més fàcil de crear
- Ja s'han pres decisions pel que fa a l'estructura del codi
- Elements precuinats adaptables a molts dispositius diferents
- La majoria són gratuïtes i de codi lliure
- Codi estable testejat per molts desenvolupadors
- Actualitzacions regulars per corregir errors o afegir noves funcionalitats

Els contres que té usar les llibreries:

- A vegades es necessita molts ajustos al codi per personalitzar-lo. Això pot costar més temps que fer-ho de nou
- Cal llegir una extensa documentació que es modifica quan hi ha actualitzacions
- Pot ser que no hi hagi la funcionalitat que es necessita
- Possiblement, el codi està sobredimensionat per certes pàgines web
- És més difícil utilitzar codi d'altres persones que no fan servir la llibreria

En aquest projecte s'ha optat per utilitzar llibreries. D'entre les principals llibreries que hi ha actualment al marcat, n'hi ha dos de principals, Bootstrap 4 i Foundation 6. Les dues llibreries són pràcticament iguals, tot i que segons la pàgina web BuiltWith [8], [9], Bootstrap 4 s'utilitza en un 16.3% de les cent mil pàgines més visitades d'Internet el 12-2017, mentre que Foundation 6 només es fa servir en un 3% d'aquestes.

Per tant degut a la popularitat més gran de Bootstrap 4 i ha l'extensa comunitat que té, s'usarà aquest.

### 9.2.2. Servidor

Estudi i comparació de tecnologies de servidors amb les següents qualitats: ràpid, eficient, fàcil d'implementar, robust, segur i amb capacitat d'expansió.

S'ha hagut de descartar l'ús de plataformes de domotització populars com OpenHAB [10], Home-assistant [11], Smartthings [12], Homeseer [13], Domoticz [14] o ifttt [15], perquè són servidors tancats que ofereixen una funcionalitat concreta i no es poden modificar. Així doncs, s'han hagut de buscar tecnologies que permetin crear un servidor des de zero.

Al mercat informàtic hi ha tants servidors disponibles que s'ha hagut de restringir la cerca imposant que el desenvolupament sigui amb Python, ja que és el llenguatge que s'ensenya a l'escola on es realitza aquest treball, l'ETSEIB. Hi ha dos tipus de servidors Python, els que són complets (Full-Stack Framework) i els que no ho són.

Els complets ofereixen un conjunt d'eines necessàries per crear el servidor, com per exemple: una base de dades, un gestor de plantilles, un gesto de Requests i un mòdul d'autentificació. En canvi, els que no ho són, només ofereixen el motor del servidor i la resta de components s'han de buscar per separat. Els principals servidors Python són els següents:

| Servidors Python complets | Servidors Python bàsics |
|---|---|
| - Django<br>- TurboGears<br>- Web2py | - Bottle<br>- CherryPy<br>- Flask<br>- Hug<br>- Pyramide |

*Taula 4.Servidors Python*

Els servidors Python bàsics permeten crear un servidor senzill més ràpidament, en canvi els servidors complets costen una mica més al començament però, quan el servidor creix en complexitat, ofereixen més comoditats.

Per això s'ha decidit fer servir un servidor complet. D'entre els més populars, Django és el més utilitzat, el més fàcil de fer servir i el que té més documentació i suport.

Es va començar a desenvolupar amb el servidor amb Django, però durant el desenvolupament, s'ha arribat a la conclusió que Django no és l'eina adequada per desenvolupar el projecte pels següents motius:

- No està pensat per interactuar amb perifèrics
- No està pensat per interactuar amb APIs i serveis externs
- Només està pensat per servir webs
- Per afegir totes les funcionalitats desitjades, s'ha d'executar un codi propi en paral·lel al servidor i el sistema s'alenteix
- Quan el codi creix el desenvolupament es torna molt complex i enrevessat

Tot el codi desenvolupat amb Django s'afegeix a l'annex.

Així doncs, s'ha hagut de buscar un servidor més adient. Per fer-ho, s'ha eliminat la restricció del llenguatge Python pel desenvolupament. Aquest cop s'han buscat servidors pensats pel

desenvolupament de plataformes IoT. Aquests tipus de servidors ofereixen les següents qualitats:

- Codi de lliure desenvolupament
- Connectivitat nativa amb dispositius IoT
- Interfícies gràfiques de desenvolupament
- Aptes per hardware senzill
- Serveis pensats per comunicar-se amb API externes

Els servidors que encaixen amb aquestes característiques són els següents:

| Servidor IoT | Empresa | Llenguatge de programació | Any de llançament | Espai d'instal·lació | Desenvolupadors |
|---|---|---|---|---|---|
| **Kura [16]** | Eclipse | Java | 2013 | 180 MB | 33 |
| **Node-RED [17]** | IBM | JavaScript | 2013 | 74 MB | 65 |
| **Flogo [18]** | TIBCO | Golang | 2016 | 3.3 MB | 13 |

*Taula 5. Servidors*

Després de comparar i fer recerca dels tres servidors, s'ha vist que Node-RED és l'ecosistema amb més documentació i amb una comunitat més gran de desenvolupadors. El fet que sigui un software madur, amb molta informació disponible i intuïtiu, ha fet que sigui finalment el servidor utilitzat.

### 9.2.3. Comunicacions

Per establir la comunicació entre servidor i perifèrics, s'ha buscat un protocol que sigui lleuger per poder funcionar en petits dispositius. S'han trobat dos protocols, típicament utilitzats en domòtica i en IoT.

El primer és **MQTT** [19]. Aquest protocol està basat en gestor o broker que distribueix missatges entre clients. Aquest sistema desacobla els clients generadors d'informació dels consumidors d'informació. Per fer-ho possible, se segueix una estructura on hi ha publicadors, que publiquen en uns tòpics concrets i uns subscriptors, que se subscriuen als tòpics que vulguin per rebre els canvis del valor emmagatzemat en aquell tòpic. Per tant, és un protocol que permet la comunicació entre molts clients diferents.

El segon és **CoAP** [20]. Aquest, és un protocol completament diferent. Està basat en comunicacions de màquina a màquina (M2M). CoAP fa ús dels mètodes RESTful típicament utilitzats per protocols com HTTP. Aquests mètodes permeten llegir, actualitzar, crear o esborrar informació entre els dos sistemes que s'estan comunicant.

Per desenvolupar el sistema domòtic, s'ha cregut convenient utilitzar MQTT, ja que és més adient en el cas d'un servidor i molts perifèrics.

### 9.2.4. Base de dades

La base de dades s'utilitzarà per enregistrar dades de sensors o altres dispositius al llarg del temps. Per això, s'ha fet una cerca de les millors bases de dades especialitzades en series temporals (Time Series Databases). S'ha utilitzat la web DB-Engine [21] per determinar les millors bases de dades. El seu mètode de puntuació està basat en diferents factors com: popularitat en els buscadors, en les xarxes socials, aparicions en blogs o en discussions en fòrums entre altres. Les 5 més populars ordenades de més a menys són les següents:

1. InfluxDB
2. RRDtool
3. Graphite
4. Kdb+
5. OpenTSDB

D'entre aquestes 5, s'ha comprovat que InfluxDB té una documentació excel·lent, molts exemples disponibles i una gran comunitat [22]. Per això s'ha triat per emmagatzemar totes les dades.

### 9.2.5. Hardware servidor

S'ha contemplat tots els ordinadors de baix cost que hi ha disponibles al mercat capaços de servir una petita pàgina web [23]. El microprocessador, ha de ser capaç de fer anar una versió lleugera de Linux ja que es on es desenvoluparà el codi principal.

Les opcions valorades són:

- Raspberry Pi 3
- Asus Tinker Board
- Banana Pi M64
- Odroid-C2
- BeagleBoard X15
- Parallela
- OrangePi Plus 2
- NanoPC-T3
- PineA64+
- NanoPi M3
- PixiePro
- Raspberry Pi 2

*Taula 6. Opcions com a hardware a escollir*

S'ha observat que no hi ha gran diferencia entre les diferents plaques. Per aquest motiu i com que es té a disposició una Raspberry Pi 2, s'ha fet servir aquesta per reduir els costos del prototip del projecte.

### 9.2.6. Perifèrics

Estudi i comparació de dispositius de fàcil implementació i baix cost per a controlar llums i electrodomèstics de la llar. Hi ha moltes plaques de prototipatge per escollir, entre elles hi ha les següents [24]:

- Arduino MRK1000
- ESP8266
- Particle Photon
- Teensy
- BeagleBone
- MSP430
- Discovery STM32

De totes aquestes, s'ha decidit escollir només les plaques que tinguessin un mòdul WIFI integrat amb el microcontrolador perquè s'han de fer servir en espais molt petits i és necessari poder establir una comunicació MQTT.

Així doncs, de tots els anteriors, només els dos primers, són aptes pel projecte. S'han comparat a la taula següent:

| Placa | Cost | Dimensions (mm) | Consum (transmissió) | Ports GPIO | CPU Clock |
|---|---|---|---|---|---|
| ESP8266 [25] | 2€ | 24.75 x 14.5 | 267 mW | 2 | 80MHz |
| MRK1000 [26] | 31€ | 61.5 x 25 | 515 mW | 18 | 48MHz |

*Taula 7. Especificacions de la placa ESP8266 i MRK1000*

Utilitzant la taula, es pot concloure que la placa més indicada per utilitzar és la ESP8266 pel seu baix cost i dimensions reduïdes.

### 9.2.7. Convertir veu a text i viceversa

En aquest cas no hi ha gaires opcions per triar. Les dues opcions que hi ha són: softwares que funcionen directament a l'ordinador o APIs externes.

S'ha optat per buscar utilitzar APIs externes perquè són serveis més sofisticats amb un gran poder de processament darrere.

L'únic servei que s'ha trobat en Català és el Google Cloud Speech API. Aquesta plataforma que ofereix Google, és gratuït per usuaris que enviïn menys d'una hora d'àudios per mes. Per més d'una hora, el cost és de 1,2 € l'hora.

En aquest treball, es limitarà l'ús a un màxim d'una hora per més.

### 9.2.8. Assistent virtual

Al mercat hi ha molts serveis per crear assistents virtuals o "chat-bots". Aquestes tecnologies estan evolucionant constantment, i n'apareixen de noves cada dia. Primer s'ha buscat alguna plataforma que tingués suport natiu en Català. Com que no hi ha cap, s'ha decidit obrir el ventall de possibilitats.

Per orientar l'estudi de mercat, s'ha fet servir una comparativa de *towardsdatascience [29]* actualitzada el desembre de 2017. En aquest article es recomanen 5 plataformes diferents. S'han estudiat el pros i contres que tenen cadascuna individualment a la taula 8:

| Assistent | Preu | Programar | Comunicació | Comentaris |
|---|---|---|---|---|
| ManyChat | Gratuït (Versió limitada) | Fàcil però amb possibilitats limitades | Facebook Messenger Chat | Pensat per empreses de màrqueting per difondre notícies o productes |
| Chatfuel | Gratuït (Versió limitada) | Mitja | HTML requests | Amb més de 17 milions d'usuaris globalment està posicionat com el més utilitzat |
| Conversable | Preu variable. Consultar vendes | Difícil | Plataformes de missatgeria | Destinat a grans empreses |

| | | | | |
|---|---|---|---|---|
| DialogFlow | Gratuït (Versió limitada) | Mitja | API Node-RED | Perfecte per desenvolupar assistents particulars per controlar dispositius |
| GupShup | Preu variable. Consultar vendes | Difícil | Plataformes de missatgeria | Està pensat per grans empreses de comunicació |

*Taula 8. Característiques de diferents plataformes existents al mercat*

D'aquestes 5 plataformes, la que s'adequa millor al projecte és DialogFlow, ja que hi ha una llibreria per comunicar-se des de Node-RED. A més a més la programació permet molta llibertat. Tanta que tot i no estar acceptat nativament el Català, es pot crear un assistent capaç d'interpretar aquest llenguatge.

# 10. Eines principals utilitzades

En aquest apartat s'explicarà les eines de més importància pel projecte que són poc conegudes. Les eines que no són gran importància pel projecte o que són habitualment conegudes no s'explicaran en detall aquí però s'aniran explicant on sigui convenint al llarg del document.

## 10.1. Node-RED

És un entorn de programació basat en el popular ecosistema Node.js® *[30]*. Node-RED fa que el codi pugui ser tractat en bocs gràfics on cada node són funcions de JavaScript i les unions entre ells són les dades que entren i surten d'aquestes.

Això permet un mètode de programació anomenat "Flow-Based Programming". Aquest editor es pot accedir des d'una interfície web i permet compilar, llançar i corre el nou codi des d'un altre ordinador.

Tot això fa que desenvolupar el programa sigui còmode de fer però alhora tenint tota la flexibilitat d'un llenguatge de programació tradicional.

## 10.2. Bootstrap 4

És una llibreria de components HTML, CSS i JS molt completa. Bootstrap 4 ajuda a crear el front-end de les pàgines que de manera còmoda i rapida, amb un disseny modern i estandarditzat. També és molt útil per crear interfícies adaptatives a diferents dispositius digitals. Per això, les seves funcionalitats i aplicacions s'explicaran a l'apartat 12.

Per utilitzar Bootstrap 4, calen coneixements avançats de HTML, CSS i JS, ja que per adaptar els components a les diferents pàgines pròpies requereix algunes modificacions del codi.

## 10.3. MQTT

És un protocol de comunicació pensat per dispositius IoT. Està dissenyat per ser lleuger i fàcil d'implementar. Segueix el model de comunicacions Publicar/Subscriure's basat en el protocol TCP/IP. Aquest model és una comunicació síncrona o asíncrona que crea una capa d'abstracció entre les diferents màquines del sistema i permet que molts dispositius diferents es comuniquin entre ells sense necessitat de coneixes.

Per fer anar aquest sistema de missatgeria, cal un gestor anomenat "broker". Aquest és l'encarregat de guardar les dades de cada "topic" i alerta als dispositius interessats quan hi ha

un canvi.

En aquest projecte, s'ha fet servir un broker de MQTT molt lleuger per a Node-RED anomenat Mosca.

## 10.4. InfluxDB

És una base de dades pensada per enregistrar sèries de dades temporals. Aquesta base de dades ús molt utilitzada per treballar amb alts volums de dades on el paràmetre distintiu és el temps.

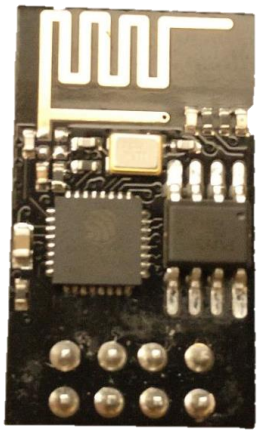
InfluxDB permet seleccionar dades d'un període de temps molt gran agrupades en intervals especificats de manera eficient. També gestiona les dades enregistrades i convenientment, les escala. Per exemple, si hi ha una dada que es guarda cada segon al llarg de molts anys, per reduir la mida, es mantindran les dades de cada segon de l'últim més però després ja només guardarà les dades de la mitjana de cada hora o dia depenent de la configuració.

## 10.5. Chart.js

És una llibreria de JavaScript que permet crear gràfics interactius per interfícies web. Aquesta llibreria destaca pel seu disseny minimalista i l'extensa documentació i exemples.

Cal destacar també que és un codi lliure desenvolupat per una àmplia comunitat de desenvolupadors.

## 10.6. ESP8266



És un microcontrolador amb un mòdul TCP/IP complet, connectivitat WIFI i ports GPIO disponibles. El cost d'un d'aquests dispositius oscil·la els 2€, el que els converteix en una plataforma ideal de prototipatge. Per aquest projecte en concret, s'han fet servir les prestacions WIFI per comunicar-se amb el servidor mitjançant MQTT i els pins GPIO per llegir l'estat de sensors i commutar interruptors.

*Figura 2. Microcontrolador ESP8266.*
*Font pròpia*

Cal destacar que s'han utilitzat plaques que, a més a més del chip ESP8266, incorporen reguladors DC/DC o AC/DC i relés entre altres perifèrics, per fer més fàcil l'implementació a l'habitatge. Aquestes plaques són la ESP8266 DC Relay Board i el Sonoff T1.

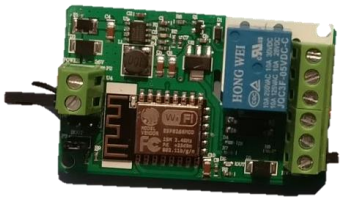


*Figura 4. Placa ESP8266 DC Relay Board*

*Font pròpia*

*Figura 3. Sonoff T1.*

*Font pròpia*

## 10.7. Mongoose OS

És un micro sistema operatiu per uns microcontroladors específics entre els quals hi ha l'ESP8266. Aquest sistema té 2 característiques essencials que són de gran importància per aquest projecte.

La primera, és que permet fer actualitzacions utilitzant la comunicació WIFI (en angles s'anomena *over the air updates* o OTA).

La segona, és que inclou una llibreria per comunicacions MQTT de sèrie.

A més a més del sistema operatiu, Mongoose OS proporciona l'editor des d'on es pot llençar el codi cap a la plataforma desitjada.

## 10.8. Dialogflow

És una plataforma en línia per desenvolupar "chatbots" de manera senzilla però amb molt potencial. Aquesta web permet crear una API perquè un servidor extern (en aquest projecte Node-RED) pugui resoldre demandes de text d'un usuari. El sistema que et permet crear, resol la demanda de text i n'extreu una resposta i una acció que es retornen al servidor en format JSÓN.

Aquest sistema de reconeixement està potenciat per un motor de *Machine learning* que facilita el desenvolupament. Per començar, s'han descrit un set de preguntes i respostes típiques,

però a mesura que s'utilitza el “chatbot”, el *Machine learning* és capaç de reconèixer noves combinacions.

L'únic desavantatge que té és que el Català no és un llenguatge que sigui reconegut per la documentació. Tot i no estar reconegut, s'ha decidit desenvolupar el chatbot en Català, ja que les prediccions de *Machine learning* poden adaptar-se amb el pas del temps.

# 11. Disseny de la solució

## 11.1. Disseny general

Per resoldre la problemàtica plantejada, s'ha desenvolupat tot un seguit de subsistemes, que en treballar conjuntament, resolen totes les funcionalitats requerides. De forma esquemàtica el sistema està compost per:

- Servidor

- Perifèrics

Per començar el cor del sistema està format per una Raspberry Pi 2 que actua com a servidor. El servidor basat en Node-RED és l'encarregat de gestionar tots els senyals que circulen pel sistema. El servidor té senyals d'entrada i sortida i s'encarrega de reencaminar les dades cap al lloc adient. Pot rebre dades de sensors extens o botons, també s'encarrega de servir les demandes dels clients web, s'encarrega de tractar amb les bases de dades, programa activitats o inclús interaccionar amb les APIs externes quan cal.

El servidor interactua amb els perifèrics. Aquests poden ser sensors o actuadors. Els sensors recopilen dades en uns intervals determinats de temps o en esdeveniments concrets, i publiquen les dades a través de MQTT en el tòpic que cadascun té assignat.

Els actuadors en canvi, estan subscrits a un tòpic en concret i quan algú canvia el valor d'aquest tòpic, l'actuador ho rep com a un esdeveniment i actua en conseqüència.

A la figura 5 es pot observar un esquema de tot el sistema.

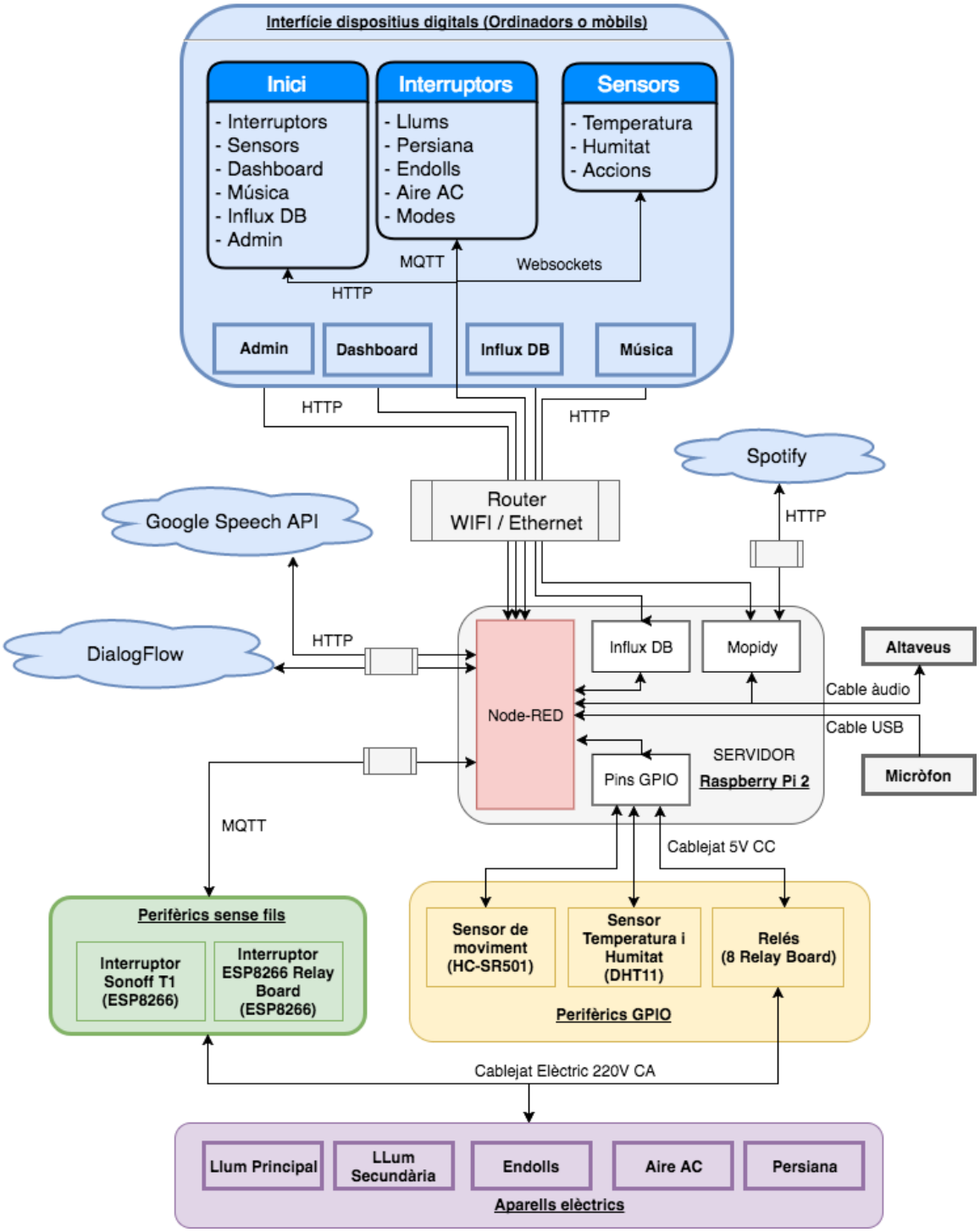



*Figura 5. Esquema genèric de tot el sistema. Font pròpia*

## 11.2. Programa principal

El sistema operatiu que s'ha fet servir és Raspbian. Dins d'aquest s'han d'instal·lar un conjunt de paquets i llibreries sobre el qual s'executa el codi.

Un d'aquests paquets és el Node-RED, que actua tant de servidor com d'editor. Els codis desenvolupats han tingut una evolució important al llarg del treball. Al començament, en ser poc codi estava tot posat en una mateixa pantalla, a mesura que ha anat creixent, s'han organitzat en "flows" per claredat però en l'àmbit pràctic aquesta distribució no aporta funcionalitat. Aquí només s'explica l'última versió del codi però a l'annex, s'adjunta fotos de l'evolució d'aquest.

Els nodes que s'han fet servir són una barreja de nodes integrals de Node-RED, nodes externs creats per la comunitat i nodes propis.

Els nodes integrals de Node-RED i els nodes externs s'han de configurar perquè facin la funció desitjada.

En els casos on una funció pròpia s'ha utilitzat més d'un cop, s'ha creat un subflow. Això permet fer modificacions fàcilment a la funció i que totes les que siguin dependents rebin el canvi.

Si la funció ja és complexa i hi ha variables pròpies de la funció que s'han d'ajustar sovint, es crea un node propi per aquesta funció.

Els nodes blancs són comentaris per fer el flows més entenedors i no tenen funcionalitat.

Els nodes verds són nodes per depurar el codi. Tots els altres nodes s'aniran explicant amb detall a continuació.

### 11.2.1. Servidor interfície web

El flow de la interfície web és senzill alhora que pràctic. Després de diverses maneres d'intentar resoldre el repte de servir una interfície, s'ha desenvolupat un sistema propi per complir totes les necessitats.

El funcionament és modular i permet tenir una plantilla única per a totes les pàgines amb la mateixa barra de navegació, logos o configuracions. Aquesta plantilla s'omple del contingut específic de cada pàgina en funció del que demani l'usuari.

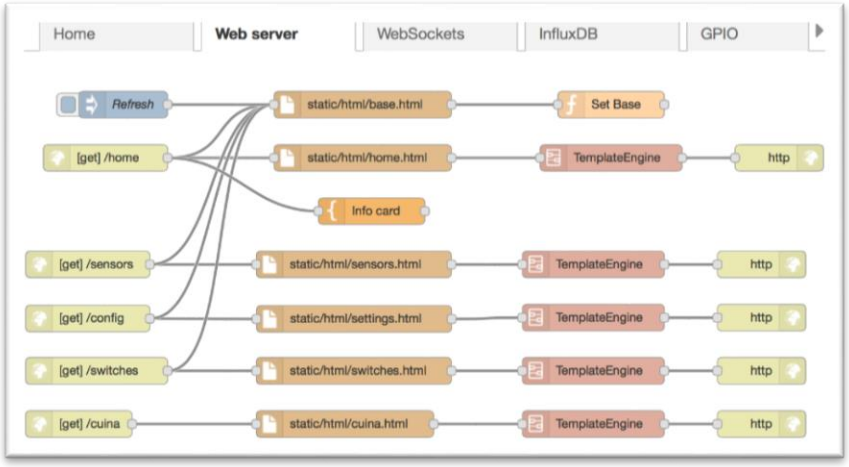



*Figura 6. Flow de la interfície web. Font pròpia*

El flow s'inicia amb l'entrada d'una *request* feta pel navegador de l'usuari contra la IP del servidor + "/URL". Aquesta request és recollida pel node "http in" que estigui configurat amb la mateixa adreça "/URL".

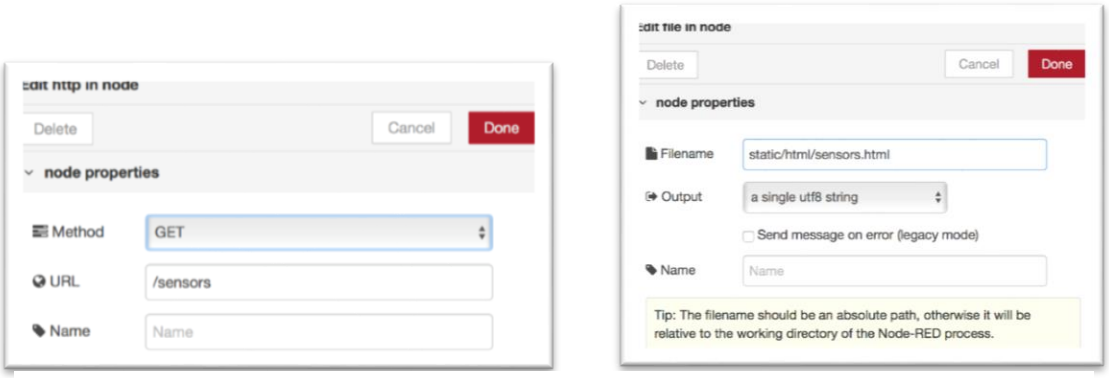



*Figura 7. Configuració del node"http in" i "file in". Font pròpia*

Seguidament, el node crida al següent connectat a la seva dreta en la cadena, en sentit d'esquerra a dreta. El següent node obre un fitxer ".html" especificat a la configuració, i l'envia al següent node. Aquest procés es mostra a la figura 7.

El següent node és un subflow. Dins d'aquest hi ha una funció de JavaScript pròpia que substitueix qualsevol paraula que hi hagi entre cotxets dins del fitxer html, pel valor que tingui la variable global amb el mateix nom. Es pot veure gràficament a la figura 8.

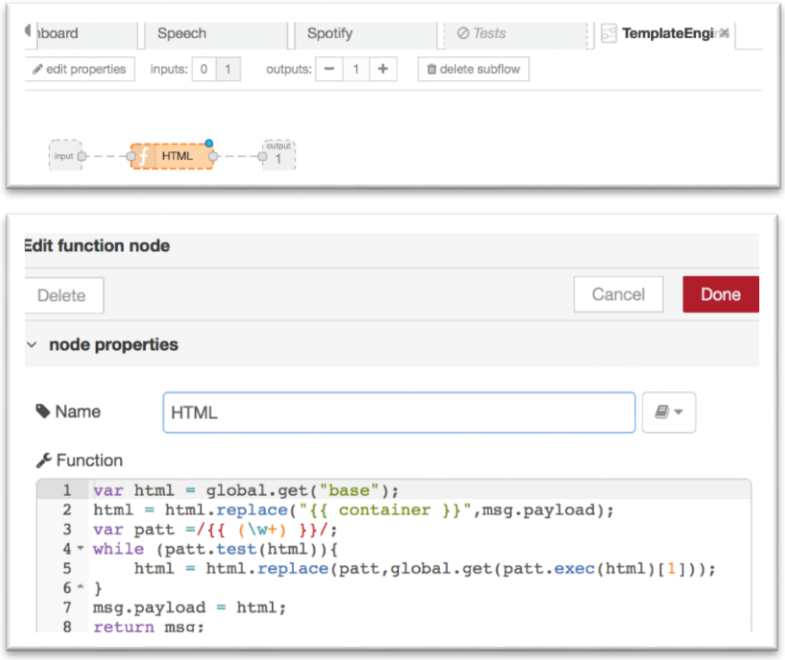



*Figura 8. Creació i descripció d'un subflow. Font pròpia*

Finalment, el missatge html que surt del TemplateEngine s'envia com a *response* cap al client.

Els avantatges que té aquest mètode són que els fitxers .html, .css i .js es poden editar des de qualsevol editor que no sigui Node-RED per treballar amb comoditat. A més a més, també permet afegir elements .html que es vulguin editar des de l'editor de node-RED.

### 11.2.2. Comunicacions MQTT

S'ha implementat la comunicació amb el protocol MQTT fent servir un node estàndard de la llibreria de Node-RED. Hi ha dos tipus de node: mqtt-in i mqtt out.

El node mqtt-in permet subscriure's a tòpics que hi hagi en un servidor en un broker especificat. Quan el valor d'aquest tòpic canviï o el node es connecti per primera vegada, enviarà un missatge amb aquest valor.

El node mqtt-out permet canviar el valor d'un tòpic pel valor que es rep en el missatge d'entrada del node. Aquest node, també ha d'estar configurat per comunicar-se amb un broker, si no, no pot funcionar correctament. A més a més, aquest node permet especificar les propietats "QoS" i "Retain" del missatge.

La propietat "QoS" especificat el nivell de confiança amb que s'ha de rebre el valor. Si val 0, el missatge s'envia només un cop sense confirmar que s'ha rebut. Si val 1, s'intenta enviar el

valor fins que es rep una confirmació que s'ha rebut. Per últim, si val 2, s'intenta enviar el missatge fins que es rep una confirmació que s'ha rebut i s'ha fet una confirmació d'aquesta primera confirmació. Aquest últim és la configuració més segura per no perdre missatges però per contra, és més lent.

La propietat "Retain" pot valer True o False. Si val True, el broker guarda l'últim valor del tòpic i l'envia a tots els clients que es connecten de nou. En l'altre cas, no ho fa.

Com a broker, s'ha utilitzat una llibreria externa que s'anomena Mosca. Per configurar el broker, només cal especificar els ports per on rebre missatges. Si es vol, es pot configurar una contrasenya per comunicar-se però no s'ha fet servir en aquest cas.

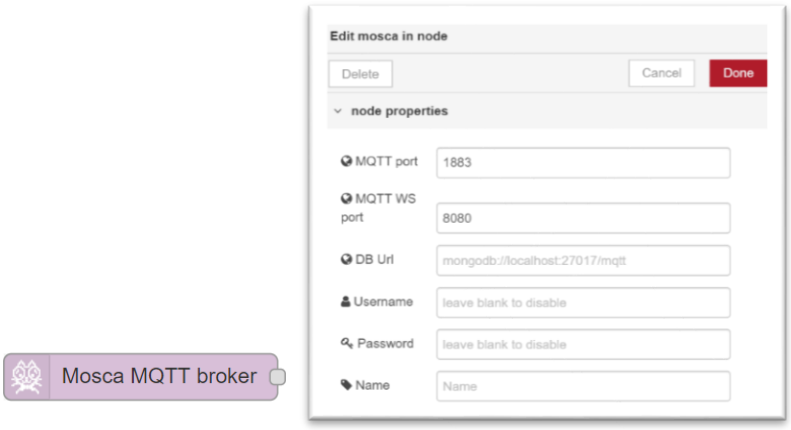



*Figura 9. Configuració del broker mitjançant la llibreria Mosca.*

*Font pròpia*

### 11.2.3. Comunicacions Websockets

Aquest és el mètode de comunicació entre servidor i client web, en els dos sentits. S'ha utilitzat en situacions on el protocol MQTT no era adequat. En concret, s'ha fet servir per actualitzar les dades de les gràfiques de la pàgina de sensors.

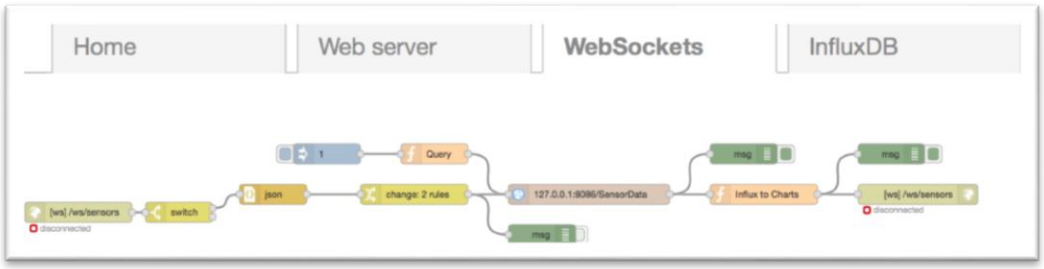



*Figura 10. Communicació WebSockets. Font pròpia*

La comunicació es rep a través del node "websockets in" de la llibreria integrada de nodes de Node-RED. El missatge es passa al node "switch" que està configurat per emetre pel port 1 tots els missatges que continguin la paraula "query".

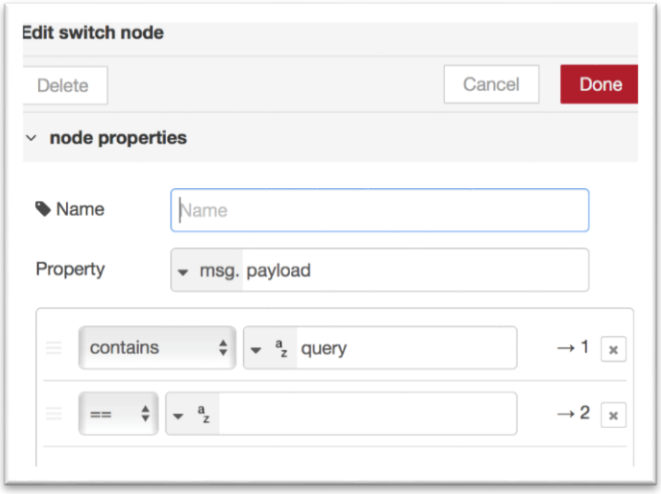



*Figura 11. Configuració del node "switch". Font pròpia*

Els missatges que compleixen aquesta condició es converteixen en format JSON amb el node "json" també de la llibreria de Node-RED.

Els elements "id" i "query" del missatge JSON passen a ser propietats de l'objecte msg utilitzant el node *change* amb la configuració de la figura 12.

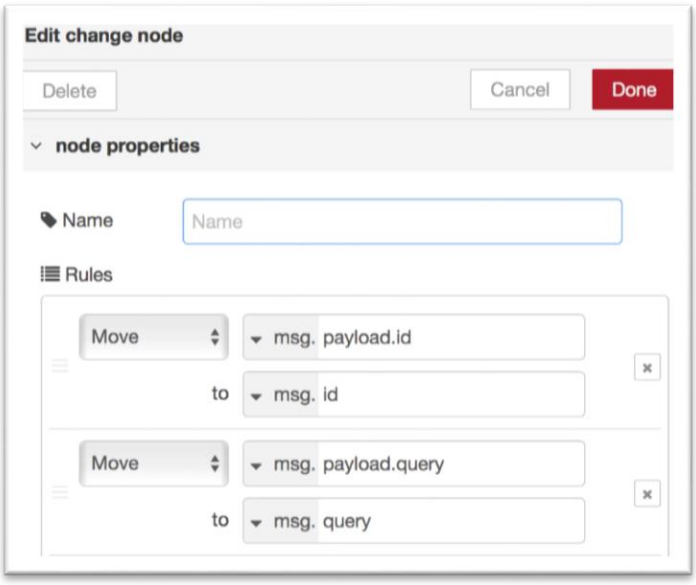



*Figura 12. Configuració del node "change". Font pròpia*

L'objecte msg.query, s'envia a la base de dades InfluxDB amb el node "influxdb in" de la llibreria "node-red-contrib-influxdb". Aquest node retorna una llista amb la següent forma dins de "msg.payload":

[{x: "2017-12-21T00:00:00Z", y: 0, Dispositiu: "ac"},{x:"...",...},...]

La funció que ve a continuació, afegeix l'id que tenia assignat l'objecte msg a msg.payload. La llista que venia dins de msg.payload s'assigna a msg.payload.data. Per tant, msg.payload queda de la seguent forma: {'id':id,'data':data}.

Tot això fa possible que l'últim node "websocket out" enviï a través de la comunicació oberta, un string que té el valor de msg.payload.

### 11.2.4. Base de dades

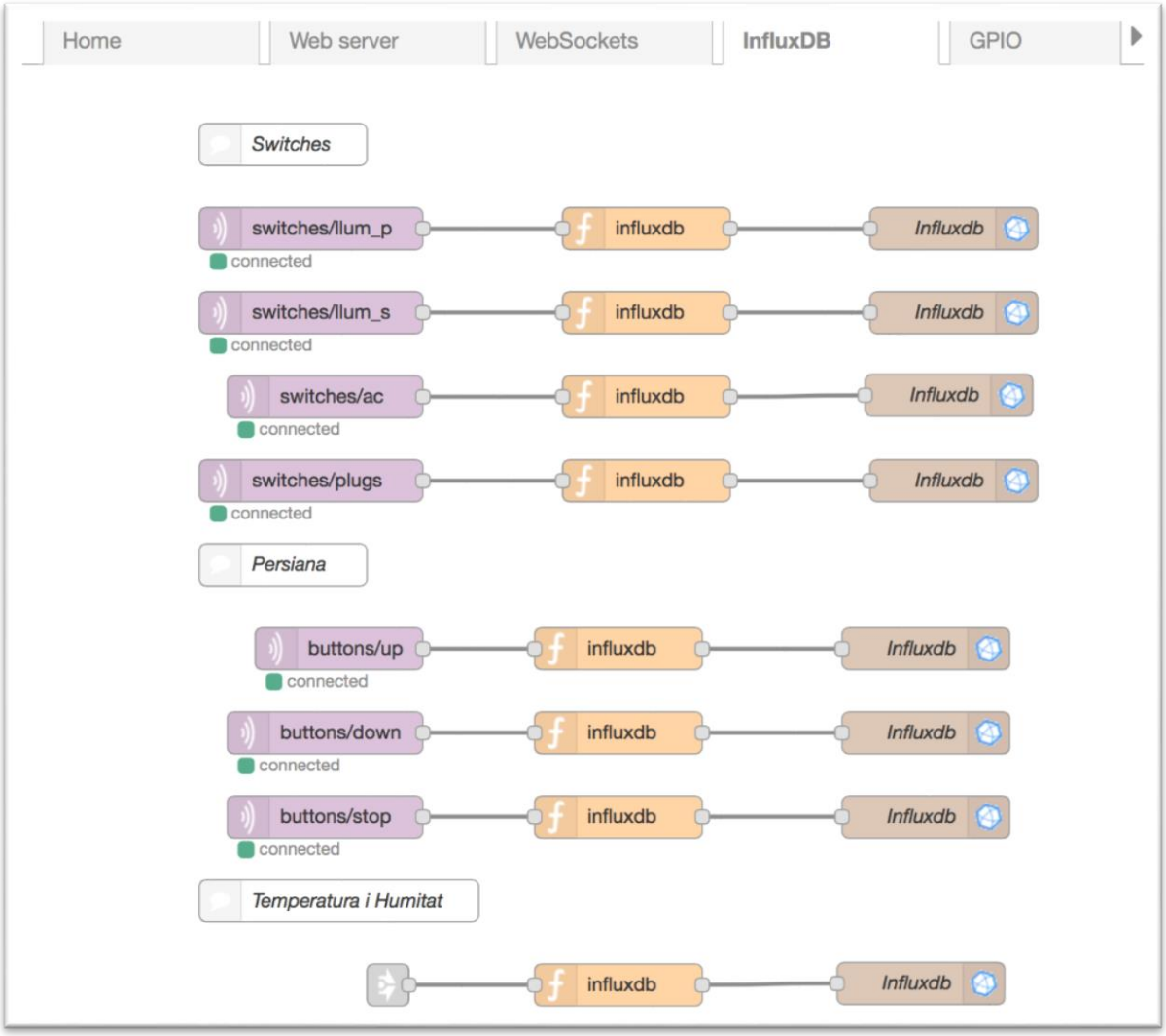



*Figura 13. Flow de la base de dades. Font pròpia*

El flow mostrat a la figura 13, és l'encarregat de registrar tots els esdeveniments a la base de dades InfluxDB.

Hi ha dos tipus d'entrades. La primera són missatges que es reben a determinades subscripcions a la comunicació MQTT.

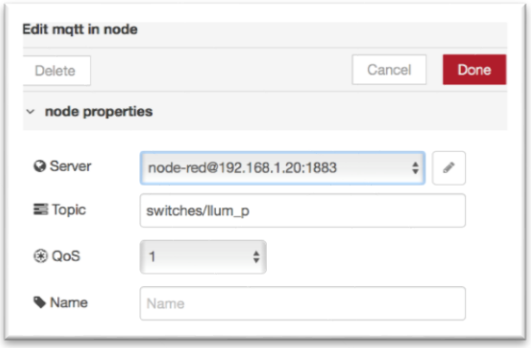



*Figura 14. Entrada a través de comunicació MQTT. Font pròpia*

El segon mètode d'entrada, és a través del node "link in". Aquest node inclòs a la llibreria estàndard de Node-RED captura esdeveniment que s'envien des del flow de GPIO quan s'enregistra una temperatura. El missatge que emet aquest node és la temperatura i la humitat del sensor.

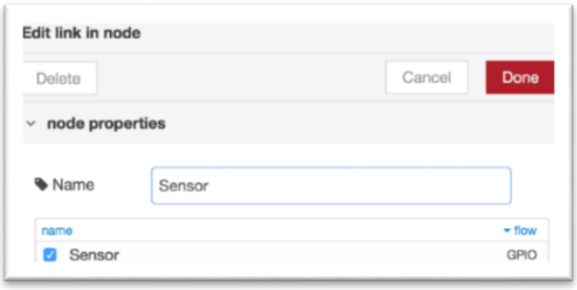



*Figura 15. Entrada a través del node "link in". Font pròpia*

Tant uns com els altres missatges, entren en una funció personalitzada per cadascun que proporciona el format correcte per enregistrar les dades dins de la base de dades temporal.

El format que accepta la base de dades és una llista amb dos objectes. El primer objecte té el nom de la dada i el valor que pren. El segon objecte conté etiquetes per després poder filtrar les dades amb facilitat. A la figura 16 es mostra un exemple.

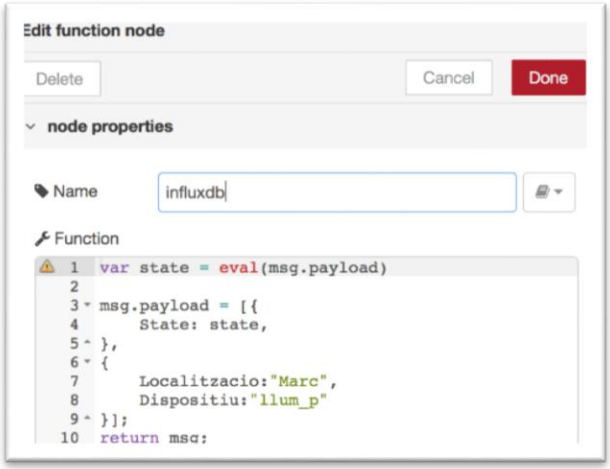



*Figura 16. Exemple format acceptat per a la base de dades.*

*Font pròpia*

Finalment, quan el missatge ja té el format adequat, s'envia a la base de dades a través del node "influxdb out". La taula on es vol guardar les dades s'especifica a l'apartat de measurement de la configuració del node.

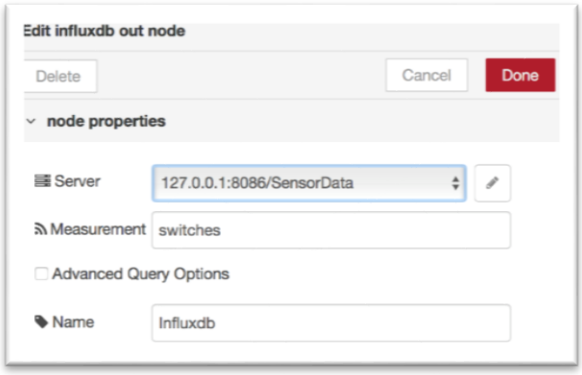



*Figura 17. Edició del node influxdb out.*

*Font pròpia*

### 11.2.5. Pins GPIO

Dins del flow de la figura 18, hi ha 4 blocs diferents distingibles pels comentaris blancs.

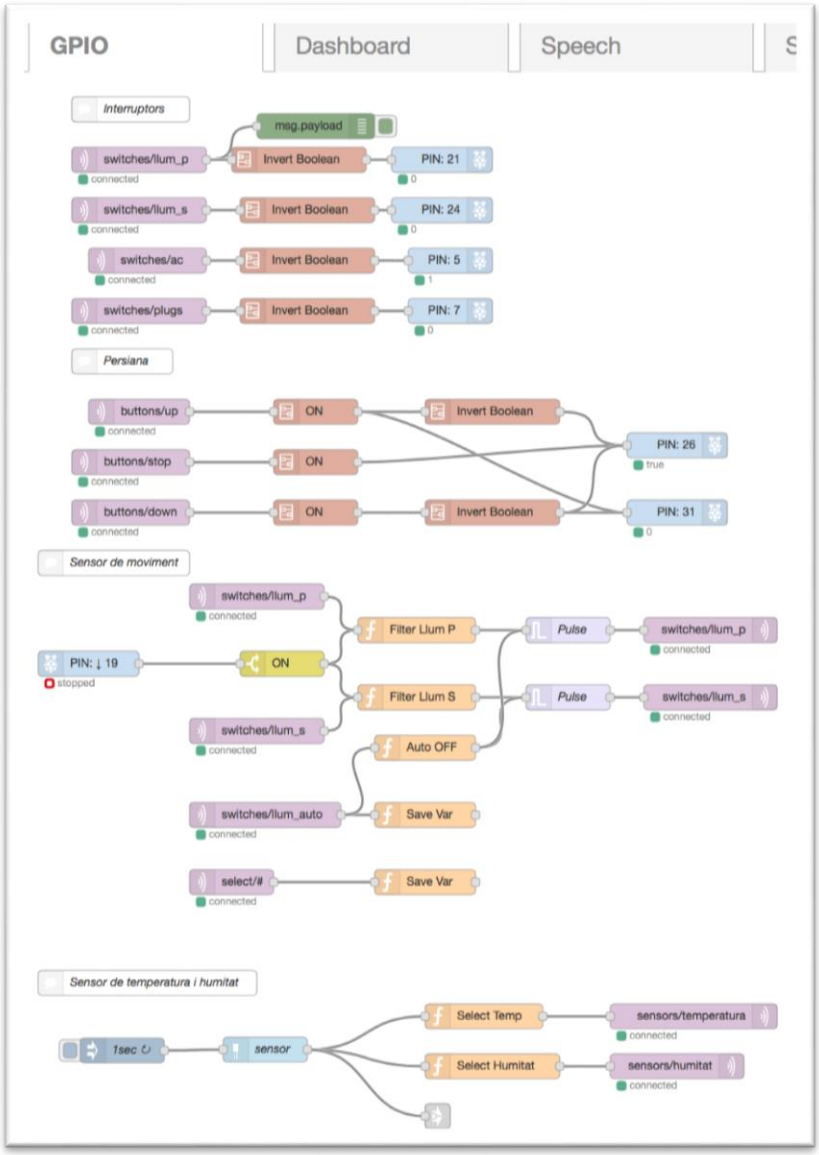



*Figura 18. Flow dels pins GPIO. Font pròpia*

**El primer bloc** d'interruptors és semblant a alguns flows que s'han vist abans. Els missatges que arriben per MQTT de cada subscripció especifica en aquest cas poden valer true o false. True en el cas de tancar el relé i false en el cas d'obrir-lo. Per ser coherents amb com està connectat a la realitat, cal invertir el senyal amb un subflow propi dedicat només a fer aquesta funció.

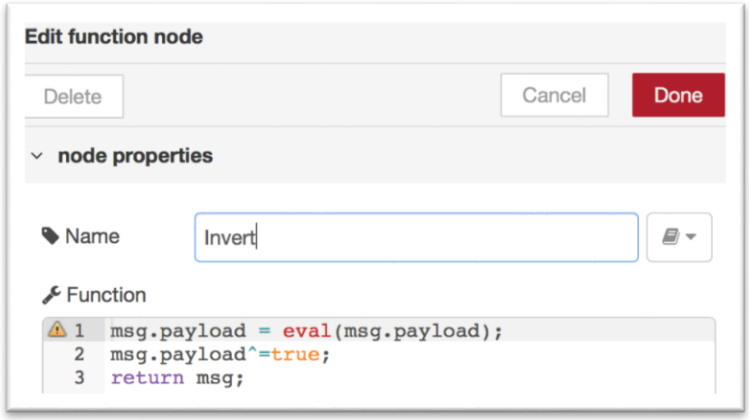



*Figura 19. Inversió del senyal amb un subflow. Font pròpia*

Un cop invertida, el missatge s'envia al node "rpi-gpio out" configurat per actuar sobre el pin que controla el relé desitjat.

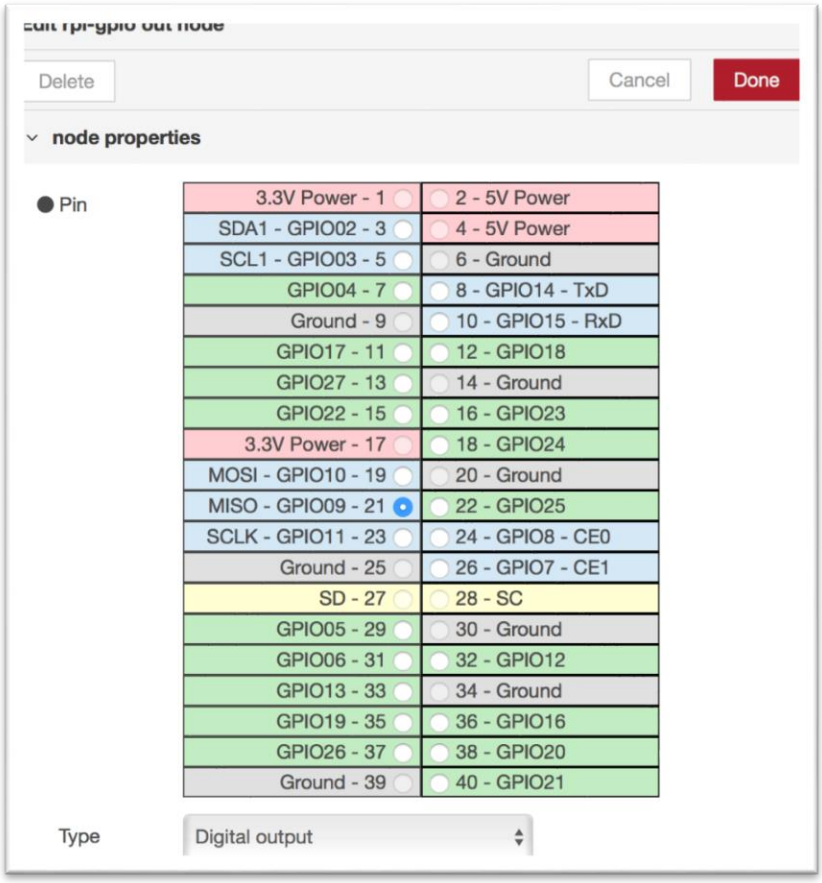



*Figura 20. Node rpi-gpio out. Font pròpia*

**El segon bloc** és una mica més complex. Per entendre'l cal conèixer com estan connectats els relés que controlen el motor de la persiana. El motor té una fase per pujar, una fase per baixar i un neutre. Per controlar l'alimentació de les fases, s'han utilitzat dos relés en la configuració de la figura 21.

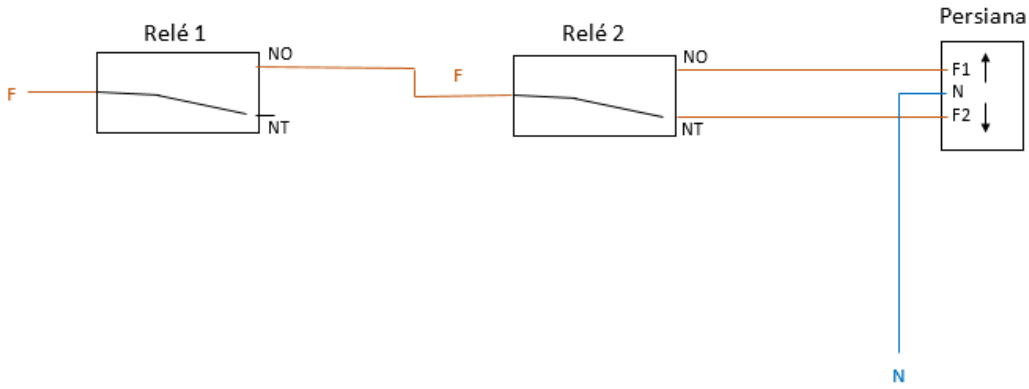



*Figura 21. Configuració dels relés de la persiana. Font pròpia*

| Relé 1 | Relé 2 | Acció persiana |
|---|---|---|
| 0 | 0 | Aturada |
| 0 | 1 | Aturada |
| 1 | 0 | Baixar |
| 1 | 1 | Pujar |

*Taula 9. Acció de les persianes*

En aquest cas els únics blocs que són nous són els subflows "ON". Com s'ha explicat abans, aquest emmascaren funcions que s'utilitzen repetidament. La funció ON envia un missatge amb valor True quan rep un missatge d'entrada. Això permet que qualsevol canvi que es faci en el tòpic buttons/up acabi amb l'acció d'apujar la persiana tingui el valor que tingui el tòpic.

S'ha tornat a utilitzar el subflow "invert boolean" per fer que els relés tinguin la mateixa resposta que la taula de la veritat anterior.

**El tercer bloc** s'encarrega de gestionar els llums com desitja l'usuari en funció del valor que

pren el sensor de moviment.

El primer node d'aquest bloc és el "rpi-gpio in" que efectua una lectura del pin connectat al sensor de moviment. En cas de detectar moviment, el sensor envia un 1. Al cap de 10s de no detectar moviment envia un 0. El node envia un missatge amb el valor que pren el sensor en canviar d'estat.

El següent node és un "switch" i serveix per deixar passar només els canvis d'estat per pujada de flanc, és a dir de 0 a 1. Aquest node sempre emet un 1 com a missatge.

A continuació hi ha un node "function" particular per cada llum que es vol controlar. En temes generals, la funció deixa passar el missatge només si les variables globals "llum_auto" i "llum_(p/s)_auto" (p/s indica que és tant per la llum principal com per la secundària) valen *True*.

Aquestes variables globals es modifiquen quan es reben els missatges adients a través dels canals de MQTT que hi ha configurats a la figura 22.

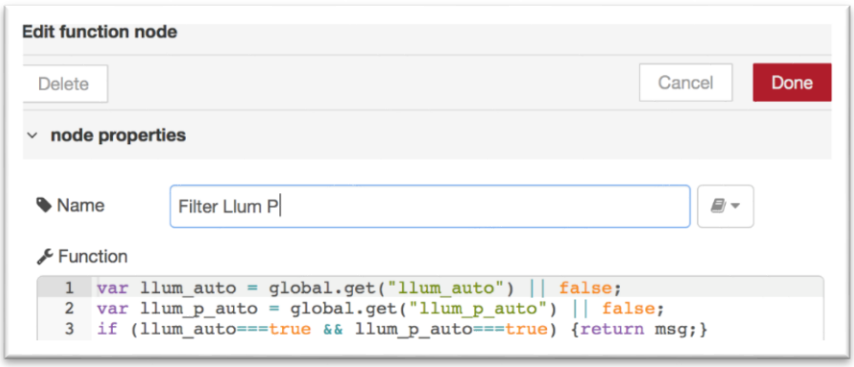



*Figura 22. Configuració del mode automàtic. Font pròpia*

ON event: en cas de rebre un esdeveniment d'aquest node, s'encenen les llums seleccionades per les variables globals com es mostra a la taula següent durant un període de temps establert pel node "trigger". Per conveniència està configurat a 1 min.

| Llum_auto | Llum_(p/s)_auto | Pulse (p/s) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 1 | 1 | 1 (durant 1 min després 0) |

*Taula 10. Execució al rebre un "ON event"*

Switches/llum (p/s) event: Aquest cas es dóna quan un client d'MQTT vol activar els llums, per exemple, un usuari i el mode automàtic (llum_auto) està activat. En aquest cas, es forçarà la llum desitjada a l'estat que es desitgi però en finalitzar el temps d'espera del node "trigger", es forçarà l'estat d'apagat. Aquesta seqüència es representa a la taula 10.

| Llum_auto | Llum_(p/s)_auto | Pulse (p/s) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 (durant 1 min després 0) |

*Taula 11. Execució al rebre un "Switches/llum (p/s) event"*

Switches/llum_auto event:

Aquest esdeveniment assegura que quan el mode automàtic es desactiva, en cas d'estar una llum oberta, segueixi així fins a canviar-se manualment. Per fer-ho cal resetejar el node "trigger" a 0.

El quart bloc recull les dades del sensor de temperatura i humitat cada segon. Per fer-ho es crea un esdeveniment cada segon amb el node "inject node". El valor del missatge de sortida d'aquest node és indiferent.

Aquest esdeveniment executa una lectura del sensor amb el node "rpi-dht22". A la configuració d'aquest node, s'ha d'especificar a quin pin està connectat el sensor, tal hi com es mostra a la figura 23.

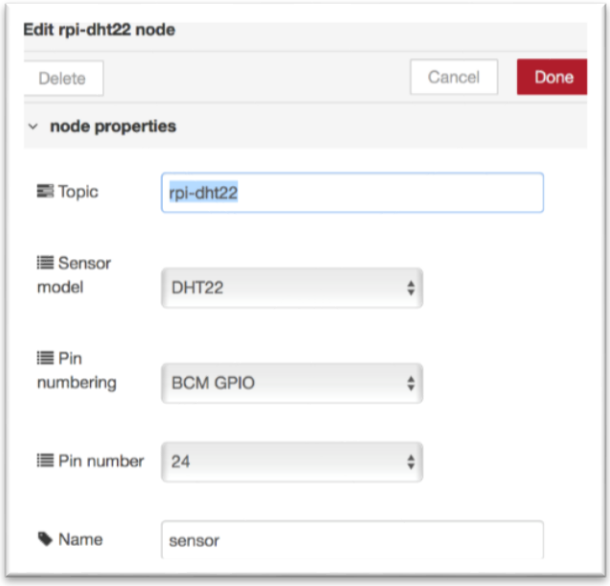



*Figura 23. Configuració del node rpi-dht22. Font pròpia*

Finalment, el missatge es descompon en dos, un de temperatura i un d'humitat. Aquest dos, es tracten amb unes funcions pròpies molt semblants que seleccionen la variable desitjada i l'arrodoneixen al primer decimal.


*Figura 24. Funció pròpia de l'arrodoniment decimal de la temperatura.*

*Font pròpia*

Per acabar, el missatge que surt d'aquestes dues funcions, es publica al seus tòpics corresponents a través del node "mqtt out".

### 11.2.6. Node Propi

Dins del flow de la persiana, hi ha el node "progressbar". Aquest és un bon exemple de la utilització de nodes propis. Cal crear un node propi quan el codi s'ha de reutilitzar en diferents llocs o s'ha d'editar sovint.

En un primer moment, el codi de JavaScript del node "progressbar" estava dins d'un node "function" però s'ha vist que depenent del tipus de persiana, s'han d'ajustar algunes variables del codi. Per emmascarar tot el codi i només exposar les variables desitjades, s'ha creat un node propi des de zero.

Per fer-ho, s'ha agut de crear els tres fitxers bàsics per definir un node en Node-RED:

- package.json: és un fitxer estàndard de Node.js per definir el contingut del paquet.

- progressbar.js: és el fitxer que conté la funcionalitat del node. El codi és molt semblant al que s'hauria escrit en un node "function" però amb la diferència les variables poden dependre d'una configuració externa.

- progressbar.html: defineix l'estil de la pestanya de configuració del node a l'editor de Node-RED. En aquest fitxer s'han d'especificar les variables que es volen editar.

Aquest node en concret modifica i emmagatzema l'última posició de la persiana en funció de quanta estona s'ha polsat el botó de pujar o baixar. Finalment serveix per mostrar una barra de progrés a la interfície d'interruptors que indica en quina posició es troba la persiana en temps real.

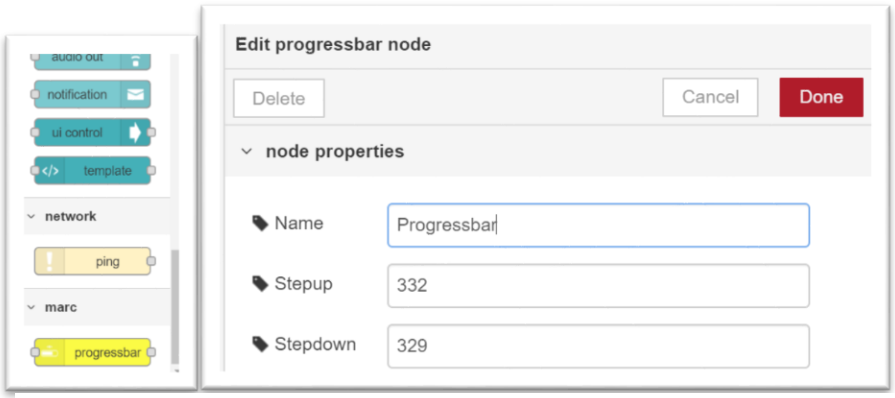



*Figura 25. Pantalla de configuració del node propi (progress bar).*

*Font pròpia*

### 11.2.7. Interfície de prototipatge

El flow de la figura 26 és una alternativa ràpida de crear una interfície d'usuari. Ha sigut la primera interfície que s'ha creat però, al no complir amb les especificacions de personalització que es demanaven, s'ha decidit crear una opció millor. De totes manes, aquesta interfície es conserva per testejar funcionalitats de manera rapida.

Els nodes utilitzats per crear-la pertanyen a una llibreria externa anomenada "node-red-dashboard".

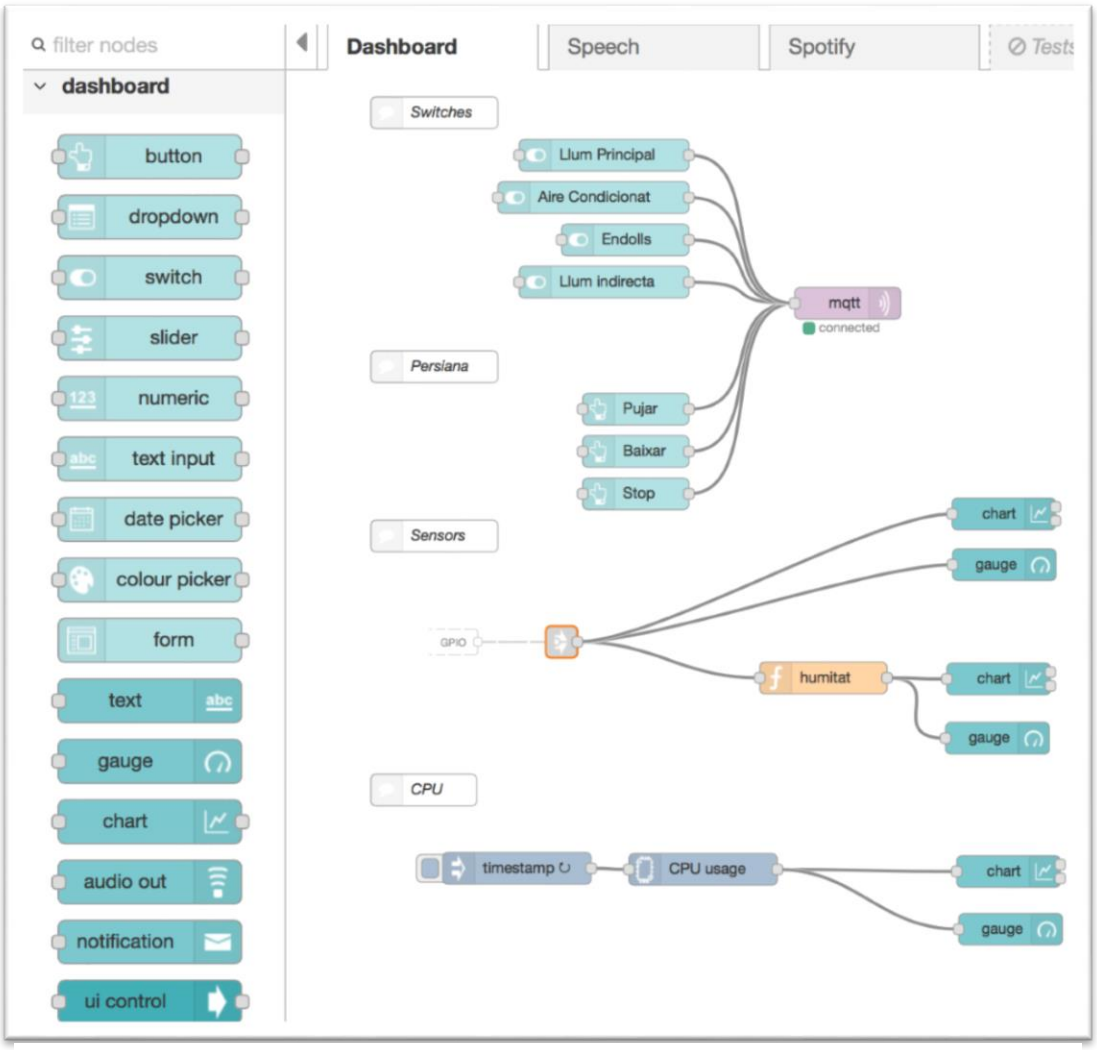



*Figura 26. Flow de la interfície de prototipatge. Font pròpia*

Cada node dels que es veuen a la figura 26 té un codi HTML assignat amb una funcionalitat específica. En utilitzar un node d'aquesta llibreria, automàticament apareix a l'adreça IP+"/ui".

Els botons utilitzats en aquest flow, creen la interfície de la figura 27 després de configurar-los.

L'estat dels botons s'envia directament al tòpic corresponent en cada cas. El tòpic s'ha especificat a la configuració de cada botó.

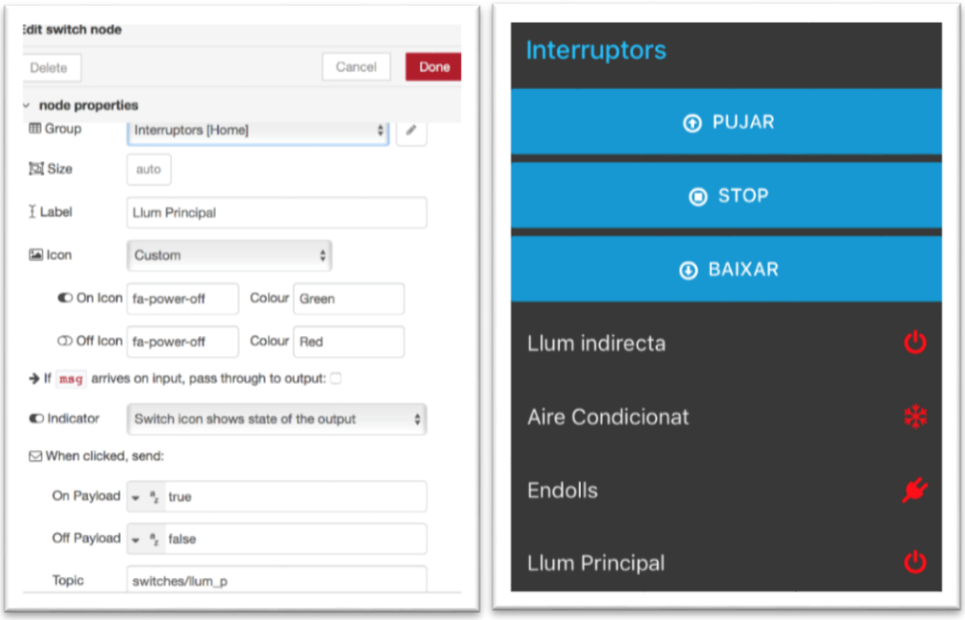



*Figura 27. Botons de la interfície de prototipatge. Font pròpia*

Aquesta llibreria també permet crear gràfics. En aquest flow, s'ha utilitzat les mateixes dades del sensor de temperatura i humitat recollides en el flow de GPIO utilitzant el node "link in".

En el cas de la humitat, s'han hagut d'extreure les dades del msg.humidity al msg.payload amb la funció pròpia "humitat".Utilitzant els nodes de "chart" i "graph" i configurant els rangs més adientsper cada mesura, s'ha obtingut la interfície mostrada a les figures 28 i 29.

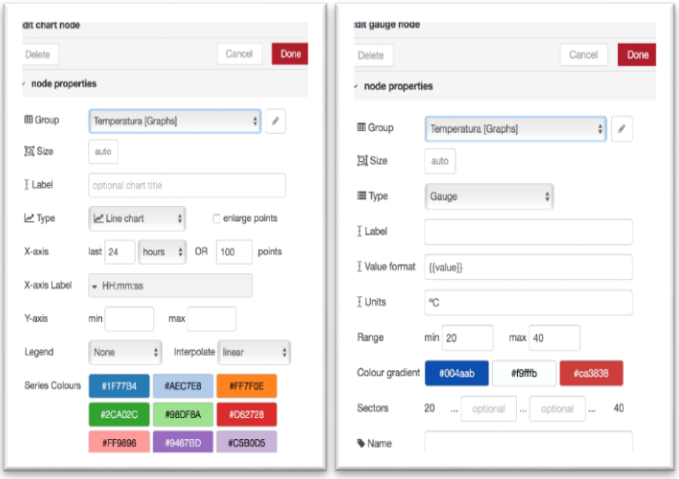



*Figura 28. Configuració per a obtenir la interfície de temperatura i humitat. Font pròpia*

Finalment també s'ha graficat la càrrega en temps real de processador del servidor (com es mostra a la figura 29). S'ha fet utilitzant el node "cpu" de la llibreria "node-red-contrib-cpu".


*Figura 29. Gràfic de la càrrega de processador del servidor, temperatura i humitat. Font pròpia*

### 11.2.8. Control per veu

Cal deixar clar que la manera principal d'interactuar amb el sistema és mitjançant els perifèrics físics o les interfícies web. De totes maneres, la manera més natural i intuïtiva d'interactuar amb el sistema domòtic és mitjançant la veu.

Per això s'ha implementat un sistema capaç d'entendre i parlar en Català fent ús de diverses APIs externes. S'han fet servir tres en total, una API de Google Speech per traduir les comandes de veu a text. Una segona, anomenada Dialogflow que serveix per determinar la intenció que té la comanda de l'usuari i donar una resposta per text. Finalment, la tercera converteix la resposta de text en un fitxer d'àudio.

El flow comença amb un missatge de MQTT. Aquest pot ser Start o Stop. Aquestes dues comandes controlen el següent node anomenat "MicroPi" que captura les comandes rebudes pel micròfon i les transmet en forma d'Strem. El següent node interactua amb l'API de Google utilitzant unes credencials prèviament configurades. El node GSpeech retorna el text dins del missatge msg.payload.transcript. Per passar el missatge a l'API de Dialogflow, cal una funció pròpia que extregui el missatge. L'API de Dialgoflow també s'ha de configurar des de la seva web que s'adjunta a la bibliografia. Aquest node retorna tot de dades de les quals s'extreuen les dues més importants fent servir dues funcions pròpies.

La primera funció, extreu la resposta en forma de text de la següent propietat del missatge: msg.payload.result.fulfillment.speech. La segona funció, extreu l'acció que s'ha de realitzar i l'envia al corresponent tòpic de MQTT.

La resposta del missatge estreta de la primera funció s'envia al següent node "Text-to speech" que conté l'API de VoiceRSS. Aquest node converteix el text en català en un àudio i el guarda en un fitxer MP3 dins la SD del servidor. S'ha configurat així perquè en cas de ser un text repetit, s'utilitza el fitxer ja realitzat anteriorment. Així s'estalvien crides a l'API i el sistema és més ràpid. Aquest node, també necessita una configuració externa que es proporciona a la bibliografia.

Finalment, es reprodueix l'àudio generat utilitzant el reproductor mpg321 a través dels altaveus connectats a la Raspberry.

El sistema pot interpretar les següents comandes entre altres:

- "Obre la llum principal", "Encendre la llum indirecta"
- "Tanca la llum", "Apagar la llum secundària"

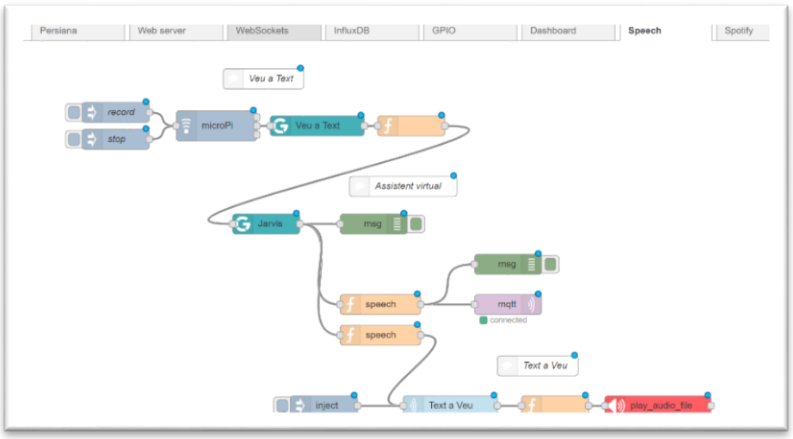



*Figura 30. Flow del control per veu. Font pròpia*

### 11.2.9. API Servidors de música

Una de les activitats més freqüents quan la gent està a casa, és escoltar música. Inclús en alguns casos, s'utilitzen melodies o cançons com a alarma per despertar-se. Per aquesta raó s'ha implementat un seguit de nodes per controlar els serveis d'streaming de música més popular com Google Music, Apple Music o Spotify i oferir les funcionalitats mencionades abans.

Per fer-ho s'ha instal·lat un servidor Mopidy. Aquest permet reproduir música pels altaveus de connectats a la Raspberry Pi 2 i controlar la música des del servidor de Node-RED o des d'una interfície local que proporciona al port 6680.

Dins del flow de música, s'ha utilitzat un conjunt de nodes per crear un despertador amb la música preferida de l'usuari.

El flow comença amb un missatge de MQTT que configura un node "bigtimer". El node "bigtimer" executa un esdeveniment quan l'hora coincideix amb l'hora configurada abans. Aquest esdeveniment executa un node "mopidy-out" que neteja la llista de reproducció del servei seleccionat.

Després, es crida un altre node de "mopidy-out" que afegeix la cançó prèviament seleccionada. La forma més senzilla de seleccionar la cançó és amb el títol de la cançó però la manera més fiable que la trobi, és utilitzant l'URL d'aquesta.

Finalment s'encadena un últim node de mopidy-out configurat per reproduir la cançó. En cas de necessitar-ho, també es pot afegir un node per controlar el volum o inclús incrementar-lo progressivament.

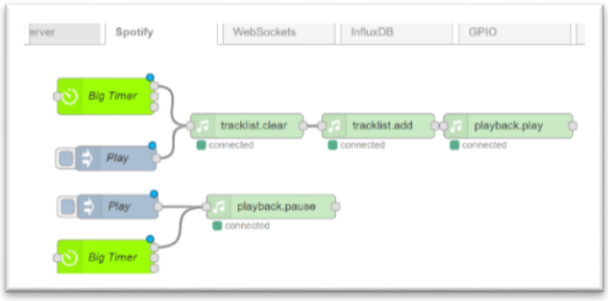



*Figura 31. Flow del control del sistema de música.*

*Font pròpia*

### 11.2.10.Seguretat

Per protegir el codi principal, s'ha decidit utilitzar el sistema de seguretat que facilita Node-RED. Per fer-ho, primer de tot, cal instal·lar la llibreria "node-red-admin". Amb aquesta llibreria, des de la terminal de la Raspberry Pi 2 s'ha de crear un password hash fent servir la següent comanda:

>>> node-red-admin hash-pw

A continuació, la terminal demana d'entrar un password i retorna un hash que s'ha de copiar en el fitxer settings.js de la carpeta d'instal·lació de Node-RED. Per exemple, com es mostra a la figura 32:

```
adminAuth: {
    type: "credentials",
    users: [{
        username: "admin",
        password: "$2a$08$zZWtXTja0fB1pzD4sHCMyOCMYz2Z6dNbM6tl8sJogENOMcxWV9DN.",
        permissions: "*"
    }]
}
```

*Figura 32. Exemple de creació d'un password hash. Font pròpia*


*Figura 33. Mostra de l'accés amb usuari. Font pròpia*

Aquesta configuració farà que en entrar a l'editor, es demani l'usuari i la contrasenya, com es pot veure a la figura 33. A les interfícies s'hi pot accedir sense usuari per comoditat, tot i que també es podrien protegir.

### 11.2.11. Actualitzacions

De cara a mantenir el codi del servidor sempre funcional o inclús augmentar la funcionalitat, hi ha unes quantes eines útils per fer manteniment.

La primera d'elles és el botó a la part superior dreta de l'editor que diu "Deploy". Aquest serveix per compilar l'última versió del codi i fer-la corre al servidor en un sol clic. Inclús permet actualitzar només la part del codi que s'ha canviat, deixant la resta de flow funcionant mentre s'actualitza en qüestió de segons. No cal reiniciar el servidor.

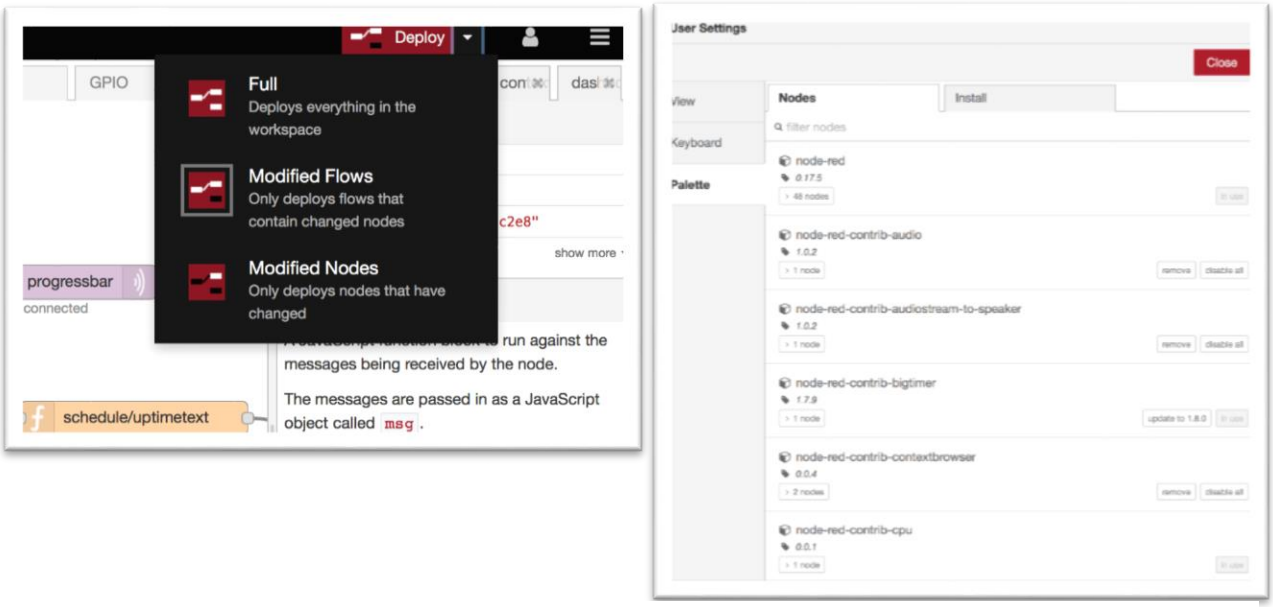



*Figura 34. A l'esquerre, funcionalitat del botó Deploy; a la dreta, vista de la configuració Manage Palette, on es veu l'estat de totes les llibreries. Font pròpia*

Per altra banda, també permet mantenir al dia totes les llibreries instal·lades. Per fer-ho, cal anar al botó de dalt a la dreta que té tres barres i clicar a "Manage Palette". Aquí es pot trobar l'estat de totes les llibreries, actualitzar-les, desinstal·lar-les o instal·lar-ne de noves. Després d'instal·lar-ne de noves cal reiniciar el servidor de Node-RED.

## 11.3. Perifèrics

Els perifèrics són els dispositius que controlen el sistema domòtic. En el sistema que s'ha desenvolupat en aquest treball, hi ha dos tipus. Els primers són els dispositius mòbils o ordinadors que fan servir la interfície web. Com el software d'aquests és tan complex, més endavant es dedica l'apartat 12, a explicar-los en detall.

El segon grup de perifèrics són sensors, botons o interruptors físics. Els sensors poden ser tant analògics com digitals. Els interruptors o botons, es poden ser els mateixos que ja hi ha instal·lats a la casa.

Per comunicar els sensors i els interruptors amb el servidor, s'utilitza un microcontrolador molt popular en el món del IoT, anomenat ESP8266. És un petit xip que porta un mòdul WIFI integrat que li permet comunicar-se amb el servidor sense cables utilitzant la comunicació MQTT.

Per programar aquests xips, s'ha decidit utilitzar una eina de programació que permet reprogramar els xips a través de la comunicació WIFI anomenada Mongoose OS. La primera instal·lació s'ha de realitzar per serial amb un ordinador.

### 11.3.1. Programa principal

S'ha realitzat un programa pels interruptors, ja que és el perifèric més complet. Inclou un sensor, el botó (l'interruptor) i un actuador (el relé). El codi del programa és minimalista perquè el microcontrolador que s'està fent servir és senzill. L'algoritme que regeix el programa està format per dos blocs importants.

El primer bloc s'executa a partir d'un esdeveniment, que és quan arriba un missatge del tòpic al qual està subscrit. Aquest missatge es guarda en una variable anomenada "estat". Aquesta variable guarda l'últim estat real al qual es troba, per exemple una llum. Després de guardar l'estat, canvia el relé a l'estat rebut.

El segon bloc, és un bucle infinit on s'executa el procés de la figura 35.

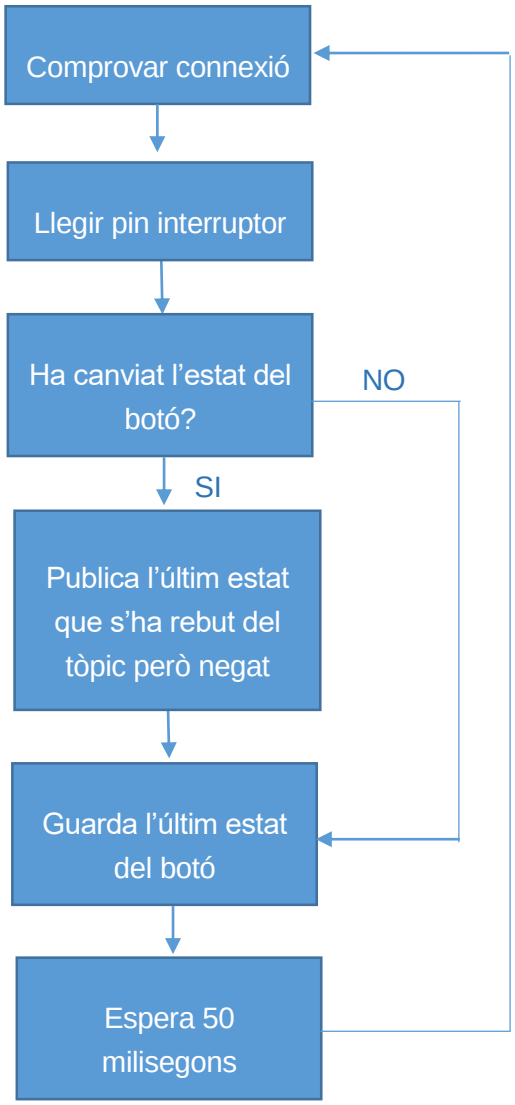



*Figura 35. Procés dut a terme al segon bloc de l'algoritme dels interruptors. Font pròpia*

Cal destacar que si els botons que ja venen amb la casa són interruptors, cal tractar-los com a esdeveniments i no com a estats. En cas contrari pot crear conflicte d'estats entre l'interruptor real i la interfície.

### 11.3.2. Comunicació

El servidor interactua amb els perifèrics a través de la comunicació MQTT. Aquesta comunicació, que es realitza a través de la xarxa local WIFI, utilitza la metodologia Publisher/Suscriber. En aquest paradigma, hi ha uns temes concrets que el servidor gestiona on els perifèrics poden publicar o subscriure. Això crea una capa abstracta entre la comanda i l'acció on el perifèric només li cal dir la comanda i el servidor ja s'ocupa de resoldre l'acció.

# 12. Interfície d'usuari

## 12.1. Disseny general

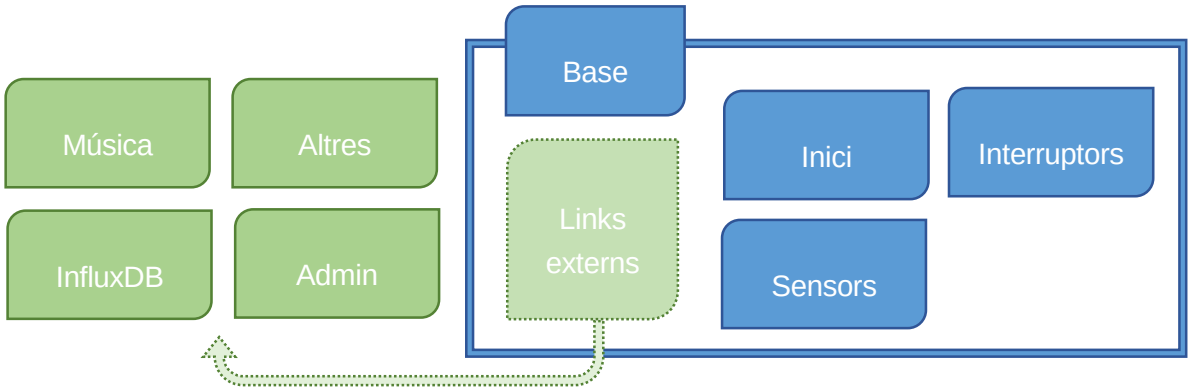



*Figura 36. Esquema general de la interfície d'usuari. Font pròpia*

L'objectiu de la interfície gràfica és tenir una manera senzilla, flexible i portable d'interactuar amb el sistema domòtic. En tot moment s'ha intentat que la interfície tingués una estructura coherent, entenedora i que seguís unes línies de disseny igual per a totes les pàgines.

La interfície està formada per pàgines pròpies i pàgines externes que els serveis utilitzats ofereixen. Les funcions principals per l'usuari, estan incloses dins les pàgines pròpies. Les pàgines externes aporten funcionalitats addicionals, de les quals un usuari inexpert pot prescindir. Algunes d'elles, només són necessàries per modificar configuracions avançades.

Les pàgines externes s'ofereixen des dels diferents servidors utilitzats. Això vol dir que són pàgines locals. Per tant, no hi ha risc que la pàgina canviï al llarg del temps o tanqui com podria passar amb pàgines web publiques. Aquestes pàgines, només les pot modificar el gestor del sistema. A més a més, les pàgines que poden modificar paràmetres importants del sistema, estan protegides per contrasenya.

Per altra banda, les pàgines pròpies són fetes a mida. Aquestes, també se serveixen en local i només les pot modificar el gestor del sistema. A diferència de les prefabricades, les pròpies són 100% costumitzables i totes tenen un disseny cohesionat.

Per mantenir la unicitat del disseny i facilitar el desenvolupament per fer les pàgines adaptatives a diferents dispositius s'ha utilitzat la llibreria Bootstrap 4. De totes maneres, la llibreria per si sola no dóna els resultats que es demanaven en aquesta interfície així que s'ha hagut de complementar amb codi propi.

Les pàgines pròpies segueixen l'estructura de plantilles que s'ha explicat a l'apartat 11.2.1. Totes, estan formades per un document base que proporciona el marc i l'estructura. Les altres pàgines, omplen el contingut del document base.

Aquesta estructura, permet fer canvis a la barra de navegació i que s'apliquin a totes les pàgines pròpies. Per exemple, quan es vol afegir una nova pàgina.

Els enllaços externs també s'inclouen dins de la barra de navegació però, al fer clic a l'enllaç, la pàgina nova s'obre en una altra pestanya. Aquest comportament està fet expressament per donar a entendre que s'està sortint de la navegació normal. Per avisar d'aquest comportament, s'ha posat una icona al costat dels noms de les pàgines prefabricades.

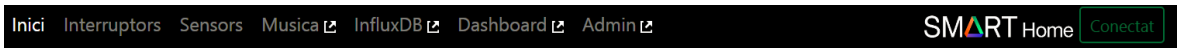



*Figura 37. Detall de la barra de navegació. Font pròpia*

Totes les pàgines pròpies segueixen una estètica unificada en forma de cartes. Aquestes estan organitzades en columnes, la mida i el nombre de les quals varia en funció de les dimensions de la pantalla on es mostri, de manera que s'adequa a tots els tipus de dispositius.

## 12.2. Base

### 12.2.1. Descripció

La funció del fitxer base és, com s'ha explicat a l'apartat anterior, facilitar la feina als desenvolupadors per no haver de tornar a escriure el codi que és igual en totes les pàgines pròpies.

Els elements què són iguals a totes les pàgines són els següents:

- Llibreries de Bootstrap 4
- Barra de navegació
- Títol de la pàgina
- Color de la barra del navegador Chrome per dispositius mòbils

### 12.2.2. Codi

Primer de tot, cal destacar que la barra de navegació té la funció de destacar el nom de la pàgina que s'està mostrant. En ser el mateix codi per totes les pàgines, la millor manera d'aconseguir aquest efecte és que el codi propi de cada pàgina destaqui el seu nom. Això s'aconsegueix, per exemple en el cas de la pàgina de sensors amb aquesta línia de codi:

>>> $('#sensors').addClass('active');

Tornant al codi HTML del document base, hi ha tres elements importants a destacar:

El primer element és <head> on s'importen totes les llibreries de CSS necessàries, on es defineix el títol de la pàgina i on es personalitza el color de la barra del navegador Chrome per dispositius mòbils.

El títol es defineix amb l'element <title>. El color de la barra del navegador, es pot definir utilitzant l'element <meta> i cal posar el codi del color a l'atribut content. Les llibreries s'importen amb l'element <link> i la referència relativa on està guardat el fitxer de CSS en el servidor.

```
<head>
  <!-- Configuració de bootstrap -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <!-- Defineix el color de la barra superior del navegador Chrome per mobils -->
  <meta name="theme-color" content="#000000">
  <!-- Per mostrar el titol de la pagina a la pestanya superor -->
  <title>SmartHome</title>
  <!-- Importar totes les llibreries css necessaries -->
  <link rel="stylesheet" href="/css/bootstrap.min.css">
  <link href="css/bootstrap-switch.css" rel="stylesheet">
  <link rel="stylesheet" type="text/css" href="css/bootstrap-clockpicker.min.css">
  <link rel="stylesheet" href="/css/base.css">
</head>
```

*Figura 38. Visió dels elements <head>, <title> i <meta> del codi HTML. Font pròpia*

Un altre element és el <nav>. Aquest és un element estàndard d'HTML modificat amb un estil de Bootstrap 4. Aquests estils s'han triat seguint les línies de disseny comunes a totes les pàgines. Les classes utilitzades són: navbar navbar-expand-lg navbar-dark sticky-top.

La primera és obligatòria per donar-li les propietats bàsiques de Bootstrap 4 a la barra de navegació. La segona fa la barra més gran. La tercera utilitza la paleta de colors foscos. Per últim, la quarta assegura que sempre estigui visible, inclús quan la pàgina es mou.

Dins de l'element <nav>, hi ha l'element <button>. Les configuracions que té aquest element, asseguren que només es vegi un botó quan la pantalla es fa petita.

En paral·lel al <button>, hi ha un . Aquest és un element molt flexible d'HTML que es pot utilitzar per a molts propòsits diferents. En aquest cas, està definint tots els elements que s'amagaran dins del botó anterior en cas que la pantalla es faci petita.

Dins d'aquest  s'hi troba una llista <ul> no numerada, on cada element de la llista és <li>. L'id de cada element, és la referència per destacar el nom en cas que sigui la pàgina activa com s'ha comentat al principi d'aquest apartat.

Per últim, cada element <li> té associat un element <a> que defineix un hiperlink cap a una pàgina en concret. Aquests hiperlinks, són relatius a la IP que s'està fent servir.

El codi acaba amb la importació de tots els codis de JavaScript que aporten funcionalitat a les pàgines. Això es fa amb l'element <script> i la direcció relativa on està guardat l'arxiu al servidor.

És important que els fitxers de CSS es cridin a l'inici de la pàgina i els fitxers de JavaScript al final perquè així la pàgina agafa la forma ràpidament i un cop està tot a lloc, s'incorpora la funcionalitat.

## 12.3. Inici

### 12.3.1. Descripció

La pàgina d'inici, està pensada per ser la primera pàgina que veu l'usuari. Aquí és on s'explica totes les funcionalitats que proporciona la interfície per interactuar amb el sistema domòtic.

És el punt de sortida cap a les altres pàgines. En aquesta, s'explica cada pàgina en detall i es mostra una miniatura de com serà la pàgina. També inclou una petita descripció de la funcionalitat que proporciona cadascuna i finalment un enllaç per anar-hi.

A més a més, també es pot mostrar algunes informacions destacades de cada pàgina en la mateixa descripció.

### 12.3.2. Codi

Com en tots els casos, cal importar els estils CSS i les funcions JavaScript pròpies d'aquesta pàgina. Per fer-ho, s'utilitzen els elements <link> i <script> respectivament ja explicats a l'apartat 12.2.2. La imatge del fons de pantalla, s'ha d'importar directament des del fitxer HTML en comptes del fitxer .css, ja que sinó, no troba el fitxer. Cal fer-ho amb l'element <style>.

```
<div class="container">
  <div class="card-columns">
    <div class="card d-inline-block">
      <img class="card-img-top" src="img/background_min.jpg" alt="Card image cap">
      <div class="card-body">
        <h4 class="card-title">Interruptors</h4>
        <p class="card-text">Des d'aquí podràs controlar tots els teus dispositius elèctrics de la teva habitació.
        Interruptors, persianes, endolls...</p>
        <a href="/switches" class="btn btn-primary">Anar-hi!</a>
      </div>
    </div>
```

*Figura 39. Codi exemple d'una carta. Font pròpia*

La resta d'elements, ja són tots part del contingut de la pàgina. El contingut està englobat per un  amb una classe de Bootstrap 4 anomenada "container" i un altre  amb la classe "card-columns". Aquests dos elements, asseguren que el contingut es mostri de manera ordenada dins del marc de la finestra web.

Dins d'aquests elements, hi han les cartes. Se'n podent posar tantes com es vulgui i però han de seguir la següent estructura:

Primer de tot, hi ha un  amb la classe "card" que defineix la carta. La classe "d-inline-block" garanteix que les cartes no quedin partides per la meitat en canviar de columna.

Després, hi ha la imatge de capçalera amb l'element <img> i la classe "card-img-top" que la defineix com a tal. És indispensable indicar la ubicació de la imatge al servidor.

A continuació hi ha un altre element  amb la classe "card-body" que defineix el contingut del cos de la carta.

Dins d'aquest contingut, hi ha l'element <h4> que defineix el títol, el <p> que conté l'explicació detallada de per què serveix la carta, i un element <a> que actua com a botó però que és un hiperlink a la pàgina que s'ha especificat a l'atribut obligatori "href". Com en altres casos, aquesta direcció és relativa.

Algunes cartes inclouen altres elements com per exemple l'element <small>. Aquest insereix un text més petit i al combinar-lo amb la classe "text-muted", en resulta un text petit i difuminat.

Per últim, podem posar un element del sistema de plantilles propi com per exemple, {{ info }}. El servidor serà l'encarregat de substituir aquest element per una carta adequada amb informació d'última hora, quan es carregui la pàgina.

## 12.4. Interruptors

### 12.4.1. Descripció

La pàgina d'interruptors està pensada per controlar tots els aparells elèctrics que hi ha connectats al sistema. Aquesta pàgina té un sistema de navegació propi que permet moure's per a les diferents habitacions que té la casa.

Després de la barra de navegació, hi ha les cartes que estan agrupades per funcionalitats. El títol de la carta orienta l'usuari cap a la funcionalitat que busca.

Dins de cada carta hi ha elements diferents, tots ells amb títols o noms que siguin fàcils d'entendre i d'utilitzar.

Per fer la interfície més pròxima a l'usuari s'ha incorporat dos mètodes senzills de customització. El primer és que les cartes es poden minimitzar fent clic al botó de "Amagar" que hi ha al costat del títol. El segon és el botó amb la icona d'unes escombraries que elimina l'element de la carta.

Aquestes modificacions es fan a nivell local de cada navegador. Per tant, mentre no es refresqui la pàgina, es mantindran les accions realitzades. Si es vol recuperar la interfície original, només cal refrescar la pàgina.

### 12.4.2. Codi

Com en el cas de l'inici, s'han importat totes les llibreries necessàries. També s'ha encapsulat tot el contingut dins d'un element  amb classe "container". Dins d'aquest, s'hi troben els següents components:

El primer element és <nav>, el qual genera la barra de navegació pròpia per identificar les habitacions. Dins hi ha una llista <ol> amb elements <li>. En posar la classe "breadcrumb" i "breadcrumb-item", la llibreria de Bootstrap 4 incorpora els separadors "/" automàticament mitjançant CSS. Cada element de la llista té un element <a> però com que només hi ha una habitació activa, els hipervincles estan desactivats.

El segon element és un . Altre cop, s'ha utilitzat la classe "card-columns" per organitzar i espaiar les cartes dins del contenidor. Cada carta, està definida per un  amb la classe "card". Els elements que hi ha dins d'aquestes són els següents:

- : La classe "card-header", assigna la propietat de títol de carta al text. "text-center" posiciona aquest títol al centre. L'element <a> fa que el text del títol sigui un botó per amagar o mostrar el contingut de la carta que

tingui el mateix identificador que el valor de l'atribut "href".

- : Aquest és l'element que delimita tot el contingut de la carta que s'ha d'amagar o mostrar en apretar el botó del títol.

- <ul><li class="list-group-item">: És una llista que conté elements. En utilitzar la classe "list-group-item", els elements de la llista queden dividits per una línia suau. Dins d'aquests elements <li> hi ha tres elements més.
  El primer  és un quadre de text estilitzat per descriure la funció d'interruptor. A continuació hi ha un <input>. Aquest input serà estilitzat en forma d'interruptor utilitzant la llibreria "Bootstrap 4-switch". A l'apartat de comunicació MQTT s'explicarà com canviar els estils d'aquest per fer-lo més entenedor.

  Per últim hi ha l'element <button> configurat amb una icona SVG. En ser clicat, aquest activa una funció de JavaScript que elimina tot l'element <li>.

- <button>: Aquest  amb la classe "btn-group" s'utilitza per englobar botons i donar l'aparença que estan junts. La classe "w-100" imposa que l'amplada d'aquests botons, ocupi el 100% de l'amplada de l'element que el conté, en aquest cas, l'element <li>.

  Dins d'aquest  hi ha l'element <button>. Per donar color blau al botó, s'ha utilitzat la classe "btn-primary" i per donar color vermell, la classe "btn-danger". Aquests colors són estàndards de la llibreria de Bootstrap 4.

- : La classe "progress", "progress-bar" i "progress-bar-striped" donen l'element  la forma d'una barra de progrés ratllada. La classe "mt-3", s'ha fet servir per deixar tres píxels de marge superior. La dimensió inicial de la barra de progrés està definida amb l'atribut style i width.

- : La classe "input-group" donà a l'element  l'aparença estàndard de Bootstrap 4 per elements d'entrada de text. Per altra banda, la classe "clockpicker" permet que s'executi la llibreria "Bootstrap 4-clockpicker" quan s'intenta entrar text.

  Aquesta llibreria desplega un menú amb un rellotge que permet seleccionar una hora i un minut. Aquests valors, són guardats a l'atribut "value" de l'element <input>.

Posteriorment, una funció de JavaScript recull aquest valor i l'envia al servidor.

- : En utilitzar la classe "dropdown", l'element es converteix en un botó desplegable. Aquest botó inclou una llista d'hipervincles <a> definits amb la classe "dropdown- item". En clicar un d'aquests hipervincles, s'envia el valor que conté l'hipervincle al servidor.
- : La classe "d-flex" hereta les propietats de visualització de l'element pare i fa que els botons no se sobreposin. La classe "justify-content-center" posiciona els botons en mig de l'element pare, en aquest cas, l'element <li>. La classe "pt-2" posa dos píxels de padding a la part superior.

L'element <input> crea el botó amb la propietat estàndard d'HTML del tipus "checkbox". Perquè prengui la forma desitjada, s'ha posat dins d'un element <label> on s'ha configurat el color del botó.

## 12.5. Sensors

### 12.5.1. Descripció

La pàgina s'ha dissenyat per graficar totes les dades que l'usuari necessiti veure. Per tant, un dels grans trets que té aquesta pàgina el la customització. Hi ha tres gràfiques que es poden ajustar. Totes tres tenen l'opció de triar el temps de refresc, la mida de la gràfica, l'espai de temps que es vol mostrar, la unitat de temps i la variable que es vol graficar.

Els gràfics són interactius i proporcionen dades en passar el ratolí per sobre o en clicar a la llegenda. Per fer-los, s'ha utilitzat la llibreria de Charts.js.

### 12.5.2. Codi

Com en els altres casos, primer de tot, s'importen les llibreries de CSS i JS especifiques d'aquesta pàgina. També s'ha utilitzat un  amb la classe "container" que engloba tot el contingut.

En aquesta pàgina, s'ha utilitzat el sistema de graelles de Bootstrap 4 per donar a l'usuari, la possibilitat de canviar la mida de les cartes. Per fer-ho, s'ha definit una fila amb el contingut centrat amb .

Dins d'aquesta fila, hi van totes les cartes. Aquestes cartes, són les columnes i estan configurades per tenir un mida diferent per diferents mides de pantalla. L'usuari també pot modificar aquesta variable. Un exemple de carta és .

Dins de cada carta s'hi troben els següents elements:

- : Aquest element, defineix el títol de la carta. A més a més, com en les altres pàgines, el títol pot minimitzar o ampliar el contingut de la carta.
- : En altres pàgines, també s'ha utilitzat el grup de botons, però en aquest cas, són "radio-buttons", un element estàndard de HTML que només permet seleccionar un valor a la vegada de tots els que s'ofereix.

  Les classes "btn-group" i "btn" aporten un disseny de Bootstrap 4 que segueix amb la línia de tota la pàgina i els agrupa.

  El botó seleccionat, a més a més, cal afegir-li la classe active amb JavaScript per donar l'aparença que està polsat.

- : Aquest element està format per blocs que intercalen títols amb caixes d'entrada de text. Els títols, es defineixen amb l'element  i les caixes de text amb l'element <input>. Aquests dos s'intercalen fins que el final s'afageix un botó <button>.

  Els elements <input> tenen un valor predefinit a l'atribut "value" que és el que pren, si l'usuari no escriu res.

  En aquest cas, el botó és de color gris perquè s'ha utilitzat la classe "btn-secondary".

- : Per últim, aquest element és el que conté el gràfic en si. Per definir l'alçada del gràfic, s'ha utilitzat un altre conjunt de files i columnes amb l'atribut "height".
  L'element que renderitza el gràfic és el <canvas>. El canvas, només és el contenidor però els gràfics s'han de realitzar des de JavaScript. Per fer-ho, s'ha utilitzat la llibreria de Chart.js
  La configuració d'aquesta llibreria, es realitza des de JavaScript i és molt extensa. Per això s'ha preferit adjuntar el codi comentat a l'annex en comptes d'explicar-lo aquí.

## 12.6. Comunicació amb Websockets

En aquest apartat, s'explica com establir la comunicació amb entre client i servidor utilitzant Websockets. Per fer-ho cal escriure codi JavaScript.

Primer de tot, cal crear un nou objecte “Websocket” a partir de la URL que vulguem fer servir.

Aquest objecte, se li han d’afegir un seguit de mètodes propis des de l’inici que carrega la pàgina. Per fer-ho, s’ha creat una funció wsConnect que es crida al començament.

Dins d’aquesta funció, es crea l’objecte i se li defineixen els tres mètodes següents:

- onmessage: aquest s’encarrega de convertir les dades rebudes al format JSON i afegir-les en el gràfic pertinent utilitzant la funció pròpia “rawData”.

- onopen: Aquest mètode es crida quan es realitza la connexió amb èxit. Canvia l’estat de la pàgina a connectat i crea els tres gràfics que s’han especificat al fitxer d’HTML dins del canvas.

- onclose: en cas de perdre la comunicació, es crida aquest mètode. Aquest canvia l’estat de la pàgina i intenta restablir la comunicació.

## 12.7. Comunicació MQTT

Igual que en el cas de la comunicació a través de Websockets, s’ha desenvolupat un codi propi en JavaScript que s’executa des del navegador del client.

Primer de tot, s’ha de crear un nou objecte “Paho.MQTT.Client” utilitzant la llibreria del projecte d’Eclipse Paho [27]. A aquest objecte se li han afegit un seguit de mètodes propis que gestiones els diferents estats de comunicació. Són els següents:

- onConnect: Aquest mètode canvia l’estat de la pàgina a connectat i es subscriu a tots els tòpics de MQTT que la pàgina necessita per funcionar amb “client.subscribe”.

- onConnectionLost: En cas de perdre la comunicació, el navegador crida aquest mètoque actualitza l’estat de la pàgina a desconnectat i estableix tots els interruptors pertinents com a indeterminats.

- onMessageArrived: En cas d’arribada d’un missatge, s’ha utilitzat l’expressió de JavaScript “switch-case” per classificar la intenció d’aquest en funció del tòpic. Per agafar l’últim element diferenciant del tòpic, s’ha utilitzat: “message.destinationName.split("/")[0]”

  Cada cas, treballa diferent amb els missatges que arriben. Com que el codi és molt específic i complex, s’ha preferit posar comentaris en el propi codi per explicar la seva funcionalitat i inclouré-ho a l’annex.

## 12.8. Optimització

Tot i no ser prioritari en aquest projecte, s'ha intentat optimitzar el codi al màxim per proporcionar una bona experiència al client.

Pel que fa a la interfície, s'ha reduït el consum de la pàgina principal reduint la qualitat de les fotografies que es mostren en petit. S'ha reduït la mida de la pàgina de 3.84MB a 935KB i el temps de càrrega de la pàgina també s'ha reduït de 3.31s a 442ms. Tot això fa que l'experiència per dispositius mòbils sigui més amena alhora que es redueix el consum de dades.

Per altra banda, les primeres versions del servidor feien ús exclusivament de la comunicació amb Websockets amb la interfície i les variables es guardaven com a variables globals de node-RED. Aquest mètode no era pràctic, ja que s'havien d'actualitzar les variables periòdicament. Per optimitzar el procés, s'ha implementat la comunicació MQTT a tots els llocs possibles. Ara és el mateix broker d'MQTT qui s'encarrega de manegar les dades més eficientment.

# 13. Implementació

Per tal de provar tot el codi descrit anteriorment i aplicar-lo en un cas real, s'ha implementat el sistema descrit en aquest projecte en una habitació d'una casa particular. A la figura 40 es mostra una imatge del prototip del muntatge utilitzat amb només els components bàsics.

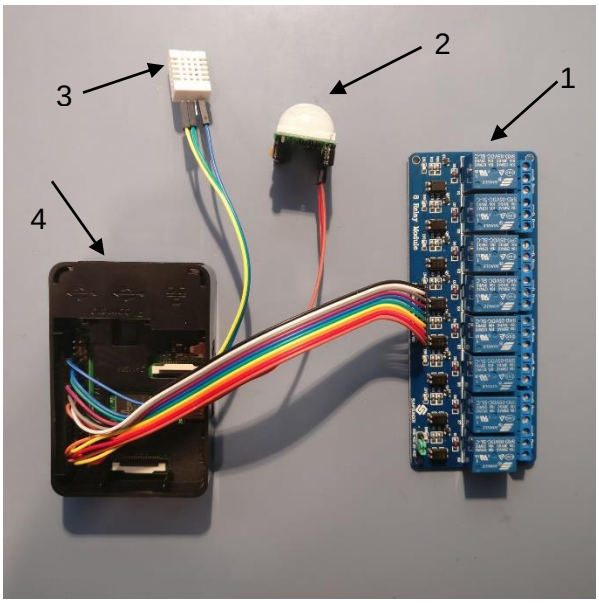



*Figura 40. Muntatge del prototip: (1) placa de relés, (2) sensor de posició, (3) sensor de temperatura i humitat, (4) Raspberry Pi2. Font pròpia*

En els punts següents es descriuen en més detall els principals elements instal·lats i la seva implementació.

## 13.1. Servidor

El programa principal està compilat en una imatge per facilitar la instal·lació. Només cal gravar aquesta imatge en una SD i introduir-la a la Raspberry Pi 2. Una SD amb la imatge s'adjunta amb el material de suport, on hi ha tot el codi que s'ha desenvolupat al llarg del treball. En encendre la Raspberry PI 2 ja es poden fer les configuracions adients des d'un altre ordinador a través d'internet.

El servidor ha d'estar en un lloc discret de la casa amb connexió a la corrent elèctrica i preferentment amb connexió a Internet via Ethernet. No cal buscar un lloc gaire accessible, ja que el sistema és de poc manteniment. Abans d'instal·lar la Raspberry Pi 2, s'ha de col·locar la SD. A més a més, si cal, es poden connectar perifèrics directament en aquesta mitjançant els pins de GPIO.

## 13.2. Relés

En aquesta instal·lació, s'ha fet servir una combinació de relés controlats des dels pins GPIO i relés independents del servidor controlats per MQTT. En aquest apartat s'expliquen els primers i a l'apartat d'interruptors, punt 13.4, els segons.

S'ha fet servir una placa comercial de 8 relés de 220v i 10A cadascun. Aquesta placa necessita una alimentació de 5V i una senyal binària per cadascun dels relés.

Aquests relés s'han connectat a les deferents fases de la caixa elèctrica d'una de les habitacions de la casa de proves. L'esquema elèctric de connexions que s'ha dut a terme és el mostrat a la figura 41, mentre que a la figura 42 es pot observar la implementació final de tota la instal·lació dels relés.

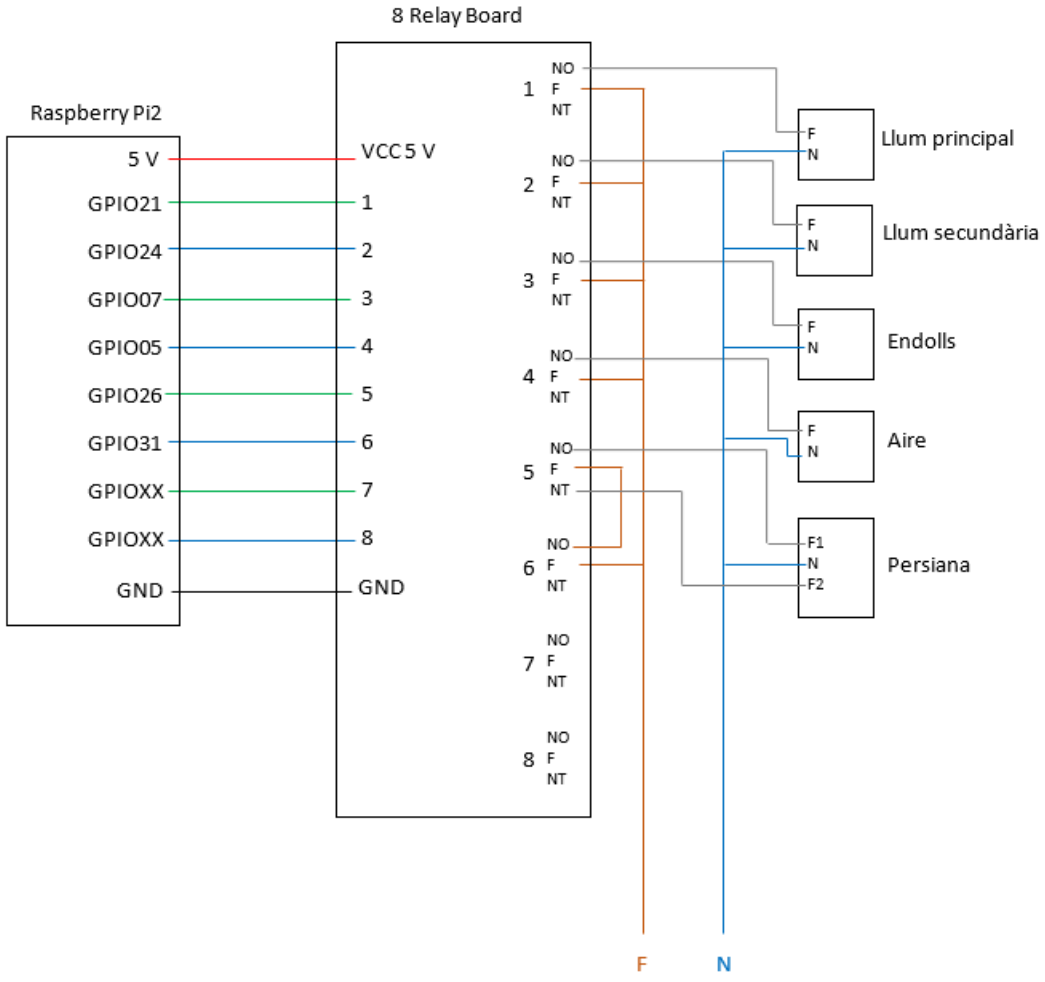



*Figura 41. Esquema elèctric de la connexió dels relés. Font pròpia*

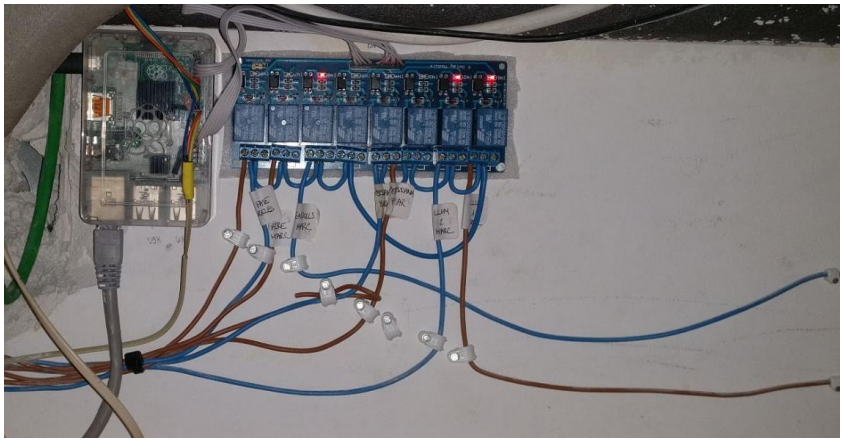


*Figura 42. Placa de relés en funcionament en el sistema final. Font pròpia*

## 13.3. Sensors

S'han fet servir dos tipus d'instal·lacions de sensors. Uns sensors connectats directament als pins GPIO de la Raspberry Pi 2 i uns altres sensors en els interruptors de paret. Els segons s'explicaran a l'apartat d'interruptors, al punt 13.4.

S'han connectat dos sensors als pins GPIO. Un sensor analògic d'humitat i temperatura i un altre digital de moviment. Per connectar-los al sistema, s'ha fet servir la configuració elèctrica mostrada a la figura 43. Els dos sensors es mostren implementats en el sistema final a la figura 44.

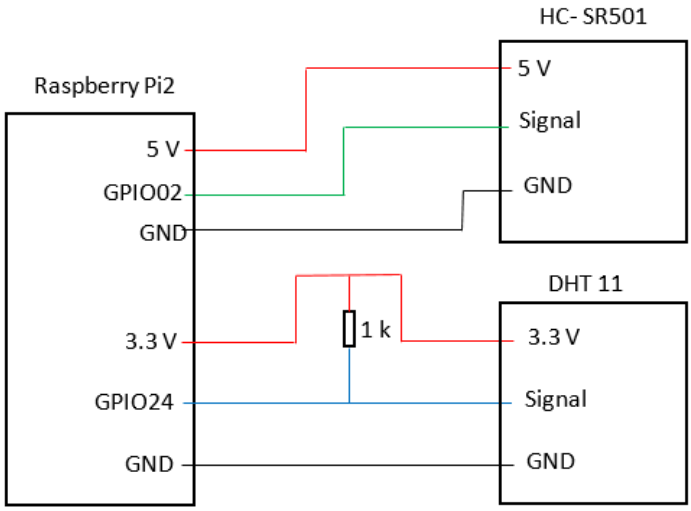



*Figura 43. Esquema de la configuració elèctrica dels sensors. Font pròpia*

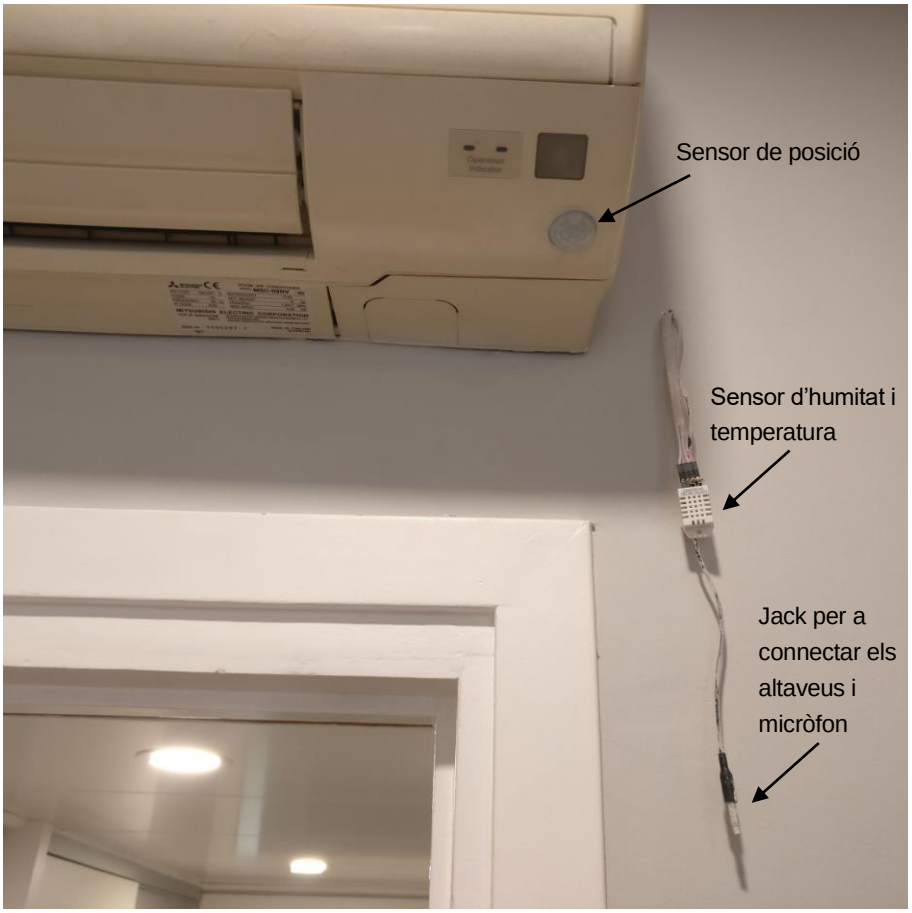



*Figura 44. Senors implementats en el sistema final. Font pròpia*

## 13.4. Interruptors

En aquest apartat s'explica tant el hardware per sensors com els actuadors inalàmbrics, ja que els chip és el mateix pels dos. El cas d'un interruptor de paret és ideal, ja que inclou els dos exemples.

Els interruptors de paret estan basats en el chip inalàmbric ESP8266. Aquest és un microcontrolador amb 2 pins GPIO i un mòdul WIFI. El microcontrolador s'ha programat un primer cop des d'un ordinador fent servir Moongose OS. Aquest editor permet fer actualitzacions del software a través de la comunicació WIFI.

Per implementar l'interruptor, s'ha fet servir dues plaques comercials que ofereixen en un sol paquet, un ESP8266, un convertidor per alimentar-lo i un relé de 220v i 10A. Aquestes són la

ESP8266 DC Relay Board  i el Sonoff T1. S'ha descrit el cas de la primera placa perquè és més il·lustratiu però el procediment és el mateix.

Per fer-la anar, s'ha fet servir una pila de 9V i un botó d'un interruptor de paret. A la figura 45 es mostra la implementació real utilitzada en les proves.

Finalment es mostra també l'esquema final del connexionat elèctric per implementar l'interruptor, a la figura 46. En els dos casos serà el mateix.

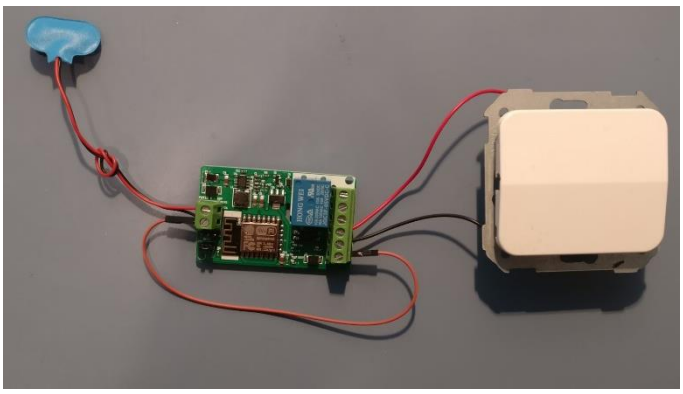


*Figura 45. Muntatge de la placa ESP8266 DC Realy Board alimentada per una pila de 9V, per a implementar l'interruptor. Font pròpia*

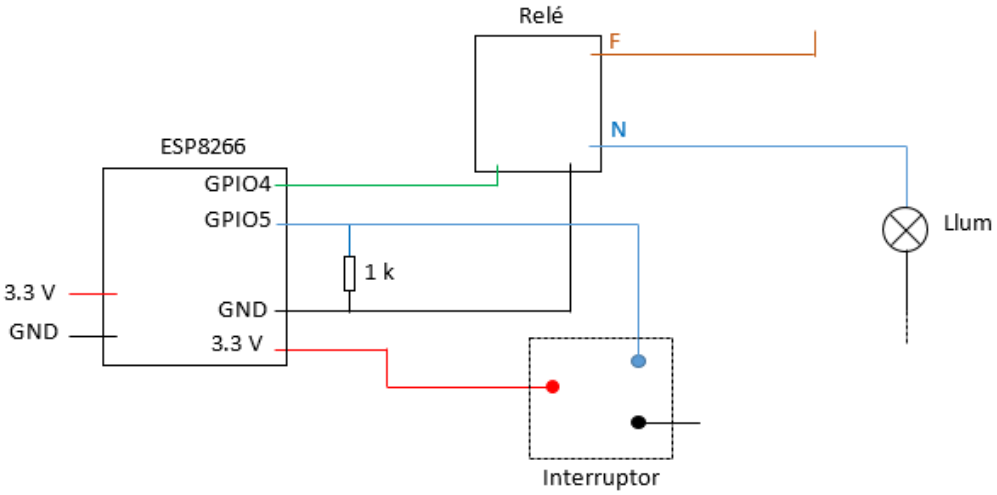



*Figura 46. Esquema elèctric de la implementació de l'interruptor.*

*Font pròpia*

# 14. Test i validació

En aquest projecte, s'ha decidit implementar el sistema en un cas real com s'ha explicat a l'apartat 3.2, abast del projecte. El sistema és utilitzat per una família de prova. S'ha fet un període de prova d'aproximadament un mes i els comentaris que s'han recollit s'han utilitzat per testejar i validar el funcionament del sistema. També s'han fet servir per valorar s'hi s'han complert els objectius proposats.

## 14.1. Errors i resolució

S'ha realitzat una taula d'errors que s'ha enregistrat al llarg del període de prova. Al costat hi han les resolucions que s'han implementat:

| Errors | Resolució |
|---|---|
| En obrir la llum manualment i estar el sensor de moviment en automàtic, les llums no s'apaguen mai. | S'ha refet l'algoritme que controla el node trigger del flow GPIO. Ara el senyal del llum també activa el trigger d'un minut. |
| La visualització en navegadors Chorme per Windows no ha sigut l'esperada. Hi ha una doble barra de posició vertical a la pàgina d'interruptors. | S'ha afegit l'atribut style="overflow:auto;" a l'element  que engloba tot el contingut. El codi final és:<br><br> |
| Els nous navegadors que es connecten a la pàgina d'interruptors no reben l'estat actual dels aparells elèctrics. | Dins del codi de JavaScript que s'encarrega de gestionar la comunicació MQTT s'ha utilitzat la propietat que té el servidor de guardar l'últim valor amb la següent línia de codi: payload.retained = true; |
| El client no pot saber si les accions que està realitzant, s'estan rebent o no en cas de perdre la comunicació a Internet. | S'ha afegit unes línies de codi a tots els fitxer HTML i als seus pertinents codis de JavaScript per implementar un indicador a dalt a la dreta de totes les pantalles que indica l'estat de la connexió. També es canvia l'estat dels botons a indeterminat. |

*Taula 12. Errors i resolució*

## 14.2. Possibles millores

A mesura que s'ha utilitzat el sistema s'ha pensat en possibles millores que es podrien implementar en un futur. Aquestes millores són possibles perquè el sistema és modular i permet afegir nous serveis i perifèrics sense interferir amb la resta d'elements.

En un futur pròxim, es podria millorar el control per veu. Aquest, ha demostrat ser un mètode de control amb molt potencial. Com a objectius de millora es pot intentar reduir el temps de resposta, ampliar la quantitat d'accions que pot realitzar o fer que el sistema estigui tota l'estona escoltant a l'espera d'una paraula clau que indiqui que ha d'actuar.

També es pot polir la interfície web, afegint un sistema d'usuaris que permeti personalitzar l'experiència per diferents persones. Per exemple, es podrien mostrar gràfics diferents depenent de l'usuari o limitar els aparells que poden actuar.

Una altra cosa fàcil d'ampliar és la quantitat de sensors que s'utilitzen en el sistema. En aquest treball s'han seleccionat els que s'ha cregut més útils però se'n podrien afegir més i de diferents tipus. Per exemple, sensors de llum, de fum o de distància.

En un futur llunyà es podria expandir molt més la funcionalitat. Al llarg del projecte s'ha aconseguit la funcionalitat plantejada al principi però a mesura que s'ha anat utilitzant, s'han pensat moltes més funcions que podria cobrir.

Per exemple, es podria fer una interfície per controlar l'estoc d'aliments. Aquest podria escanejar el codi de barres dels productes per introduir-los al sistema. Hi hauria un conjunt de receptes per fer i en triar-ne una, es restaria la quantitat gastada de cada producte. Finalment, el sistema realitzaria una compra setmanal a través d'un sistema de compra en línia a domicili.

Un sistema així ajudaria a consumir els aliments abans que es facin malbé i a tenir una dieta saludable amb ingredients naturals.

Per altra banda, també es podria desenvolupar un sistema de machine learning que aprengui amb el pas del temps les rutines de cada persona i optimitzes paràmetres com l'energia o s'anticipes en certes accions.

Finalment, també es podria estudiar la possibilitat d'incorporar una càmera al sistema per fer control de seguretat o controlar dispositius amb gests. Node-RED ja incorpora algunes llibreries per fer reconeixement d'imatge.

# 15. Planificació

La durada d'aquest projecte (projecte de fi de grau), correspon a un quadrimestre. Per aquest motiu, tota la planificació ha estat basada en aquest període de temps.

Per tal de mostrar de manera gràfica les diferents etapes que s'han seguit en l'elaboració del treball, a la figura 44 es mostra un diagrama de Gantt. A l'inici es va fer una cerca molt àmplia en l'àmbit de la domòtica, per tal de conèixer fins a quin punt i què és possible fer actualment. Les conclusions d'aquesta s'han exposat anteriorment a l'apartat 9. El següent pas constitueix l'anàlisi del problema on es defineix què es vol realitzar en detall i quines variables a monitorar són necessàries. Finalment s'ha elaborat la proposta de resolució i s'ha dut a terme la instal·lació final, tenint en compte també totes les proves realitzades per a la verificació del sistema. S'ha inclòs també en aquesta planificació, el temps de redacció d'aquesta memòria (més annexos).

| | Setembre | | | | Octubre | | | | | Novembre | | | | Desembre | | | | Gener | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 04 | 11 | 18 | 25 | 02 | 09 | 16 | 23 | 30 | 06 | 13 | 20 | 27 | 04 | 11 | 18 | 25 | 01 | 08 | 15 |
| **Objectius de projecte i funcionalitats/serveis** | | | | | | | | | | | | | | | | | | | | |
| **Estudi de mercat** | | | | | | | | | | | | | | | | | | | | |
| **Definició de tecnologies a utilitzar** | | | | | | | | | | | | | | | | | | | | |
| **Estructura del sistema i perifèrics** | | | | | | | | | | | | | | | | | | | | |
| **Desenvolupament servidor** | | | | | | | | | | | | | | | | | | | | |
| Frontend | | | | | | | | | | | | | | | | | | | | |
| Backend | | | | | | | | | | | | | | | | | | | | |
| Node-RED | | | | | | | | | | | | | | | | | | | | |
| **Desenvolupament perifèrics** | | | | | | | | | | | | | | | | | | | | |
| Hardware | | | | | | | | | | | | | | | | | | | | |
| Software | | | | | | | | | | | | | | | | | | | | |
| **Implementació del sistema** | | | | | | | | | | | | | | | | | | | | |
| **Validació del sistema** | | | | | | | | | | | | | | | | | | | | |
| **Redacció de la memòria** | | | | | | | | | | | | | | | | | | | | |

*Figura 47. Planificació del projecte. Font pròpia*

# 16. Pressupost

S'ha realitzat un estudi econòmic on s'han inclòs totes les despeses necessàries per a realitzar aquest projecte. L'anàlisi detallat es mostra a la taula 13. Cal tenir en compte que s'ha considerat des del punt de vista que un client particular contracta els serveis per a domotitzar la seva llar, de dimensions iguals que les exposades en aquest projecte. Els preus unitaris indicats són amb IVA inclòs.

| Concepte | Quantitat [hores; unitats] | Preu unitari [euros] | Amortització | Cost [euros] |
|---|---|---|---|---|
| Recursos humans | | | | |
| Enginyer industrial | 430 | 28 | - | 12040 |
| Electricista | 12 | 20 | - | 240 |
| Recursos informàtics | | | | |
| Ordinador per a la realització del codi i interfície gràfica | 1 | 1500 | 10 % | 150 |
| Llicència Microsoft Office | 1 | 110 | 25 % | 27.5 |
| Llicència Node-RED | 1 | 0 | - | 0 |
| Llicència Scart.js | 1 | 0 | - | 0 |
| Llicència Mopidy | 1 | 0 | - | 0 |
| Llicència Mongoose OS | 1 | 0 | - | 0 |
| Llicència Influxdb | 1 | 0 | - | 0 |

| | | | | |
|---|---|---|---|---|
| Llicència DialogFlow | 1 | 0 | - | 0 |
| Llicència GoogleSpeech Api | 1 | 0 | - | 0 |
| Llicència Spotify | 1 | 0 | - | 0 |
| Recursos electrònics | | | | |
| Raspberry Pi2 | 1 | 34 | - | 34 |
| HC-SR501 | 1 | 1.27 | - | 1.27 |
| DHT11 | 1 | 1.56 | - | 1.56 |
| 8 Relay Board | 1 | 4.39 | - | 4.39 |
| Sensor de Corrent | 1 | 6.59 | - | 6.59 |
| Micròfon USB | 1 | 24.95 | - | 24.95 |
| Altaveus | 1 | 39.9 | - | 39.9 |
| ESP8266 | 1 | 2 | - | 2 |
| ESP8266 DC Relay Board | 1 | 4.89 | - | 4.89 |
| Sonoff T1 eu | 1 | 12.91 | - | 12.91 |
| **Cost total després d'impostos [euros]** | | | | **12589.96** |

*Taula 13. Pressupost*

# 17. Impacte ambiental

Aquest projecte, tal com s'ha exposat anteriorment, es basa tant al desenvolupament de software com de hardware.

Pel que fa a la part de software, l'impacte ambiental causat és l'associat al consum elèctric de l'ordinador utilitzat per a tota la realització del codi. Coneixent aproximadament el valor d'aquest consum en watts, es pot calcular la petjada ecològica equivalent que s'ha generat en quilograms de $CO_2$. El valor total emès d'aquest projecte correspon a 21.973 quilograms de $CO_2$. A la taula 14 es mostra en detall el càlcul realitzat, tenint en compte que s'han utilitzat les dades proporcionades per WWF Espanya del 2016 [31],per a saber el percentatge de fonts renovables/no renovables i el càlcul de l'impacte.

| | Consum [W] | Temps d'ús [h] | Energia total [Wh] | Kg de $CO_2$ |
|---|---|---|---|---|
| Ús de l'ordinador | 350 | 430 | 150500 | 21.973 |

*Taula 14. Càlcul de l'emissió equivalent de $CO_2$*

D'altra banda, el hardware és el responsable de la major part de l'impacte ambiental. Les plaques electròniques necessàries per a realitzar la domotització, són fabricades amb productes que contaminen, com per exemple mercuri, cadmi o plom entre d'altres. Per aquest motiu, els components fets amb aquests materials han de ser reciclats en plantes especials degut a l'elevada toxicitat. La *Restriction of Hazardous Substances* (RoHS) [28] és una normativa adoptada al 2003, que restringeix l'ús d'aquestes substàncies. En aquest treball, totes les plaques comprades i emprades compleixen aquesta normativa.

A més a més, cal destacar que l'impacte positiu que aporta aquest projecte és molt major. El fet d'implementar la domòtica a les llars, permet controlar l'estat de totes les llums a través de la plataforma en línia exposada en aquest projecte, tant des d'ordinadors, com de mòbils com tauletes. Això implica el control absolut de la casa, de manera que estan fora, és possible apagar llums que en altres casos podrien quedar enceses un període de temps molt elevat. El mateix fet és aplicable al control de la calefacció i temperatura mitjançant els diversos sensors, o també a la implementació del mode "vacances". En tots els casos el resultat és un control molt elevat i precís de tots els espais, així com també, la reducció del consum elèctric de la llar. Aquest darrer fet implica indirectament la reducció d'emissions de C02, ja que la procedència de l'energia elèctrica consumida no és totalment renovable, sinó que procedeix de fons fòssils.

# Conclusions

Els principals objectius d'aquest projecte eren dissenyar, construir i testejar un sistema domòtic complet per una casa unifamiliar. S'ha aconseguit complir amb totes les funcionalitats desitjades.

El sistema pot actuar llums, endolls, aires condicionats, apujar i abaixar persianes o monitorar temperatures i humitats. Per interactuar amb el sistema es pot fer des d'una interfície web, des dels mateixos interruptors de la casa o inclús a treves de comandes per veu.

També s'ha aconseguit automatitzar les llums en funció d'un sensor de moviment o controlar la persiana automàticament en funció de la llum del sol i inclús integrar un sistema de música o alarma.

Per altra banda, s'ha vist que el mercat de la domòtica està tenint un fort creixement, cosa que reafirma que desenvolupar un projecte d'aquest estil té un gran interès econòmic.

Les eines que s'han utilitzat a l'etapa de desenvolupament del projecte, després de successives iteracions i canvis, s'ha vist que són les adequades per crear un sistema domòtic tan complex com es vulgui i donen la possibilitat d'expansió i de continuïtat en un futur.

A més a més, tant les eines per crear el hardware com el software, es pot afirmar que estan molt ben documentades i són accessibles per la majoria de la gent, així que no hi hauria d'haver cap problema per mantenir la continuïtat del projecte.

Per acabar, dir que és un treball molt ambiciós que té molt potencial per créixer i convertir-se en una part essencial d'un habitatge.

# Agraïments

En primer lloc vull donar les gràcies al professor i tutor d'aquest projecte, Lluís Solano, per haver acceptat portar aquest projecte, ajudar-me i donar-me la possibilitat de portar els meus coneixements al límit. També agrair la seva disponibilitat i consells al llarg de tota la realització del projecte.

D'altra banda, voldria agrair als familiars i amics, en especial els meus pares i el meu germà, pel suport moral i els ànims que m'han donat al llarg d'aquests mesos i de tota la carrera universitària. Gràcies a ells he pogut realitzar el treball i a més a més fer real la instal·lació domòtica a la meva habitació.

Finalment, un agraïment especial a la Laura Voltà pel seu suport incondicional al llarg de tot el treball.

# Bibliografia

## Referències bibliogràfiques

**[1]** DIARI DE GIRONA, *Big data*

[http://www.diaridegirona.cat/economia/2017/05/16/jornada-sobre-big-data-transformacio/846505.html, setembre de 2017]

**[2]** WIKIPEDIA, *Domòtica*.

[https://ca.wikipedia.org/wiki/Dom%C3%B2tica, 10 de setembre 2017].

**[3]** GIL PRESS, *22 Million Amazon Echo Smart Speakers To Be Sold In 2017, Driving US Smart Home Adoption.*

[https://www.forbes.com/sites/gilpress/2017/10/29/22-million-amazon-echo-smart-speakers-to-be-sold-in-2017-driving-us-smart-home-adoption/#35f462c4481a, 15 de setembre 2017].

**[4]** STATISTA, *Previsión del número de dispositivos conectados en casas inteligentes desde 2015 hasta 2018 (en miles de millones*).

[https://es.statista.com/estadisticas/599927/casas-inteligentes-cosas-conectadas--2018/, 10 de setembre 2017].

**[5]** AMAZON, *Philips Hue White and Color.*

[https://www.amazon.es/Philips-Hue-White-Color-controlable/dp/B015SX0UBM, 2 de octubre de 2017].

**[6]** BELKIN, *Domótica Wemo.*

[http://www.belkin.com/es/Products/home-automation/c/wemo-home-automation/, 2 de octubre de 2017].

**[7]** MEDIA MARKT, *Interruptor inalámbrico - Trust AWST-8802, Doble, Para control inalámbrico de luces, Blanco*.

[https://tiendas.mediamarkt.es/p/trust-awst-8802-interrup-pared-inalambr-1317841, 2 de octubre de 2017].

**[8]** BUILT WITH, *Twitter Bootstrap 4 Usage Statistics.*

[https://trends.builtwith.com/docinfo/Twitter-Bootstrap 4, octubre 2017].

**[9]** BUILT WITH, *Foundation Usage Statistics.*

[https://trends.builtwith.com/framework/Foundation, octubre 2017].

**[10]** OPEN HAB, *A vendor and technology agnostic open source automation software for your home.*

[https://www.openhab.org/, octubre 2017].

**[11]** HOME ASSISTANT, *Awaken your home.*

[https://home-assistant.io/, octubre de 2017].

**[12]** SMART THINGS, *Samsung SmartThings- Add a little smartness to your things*.

[https://www.smartthings.com/, octubre de 2017].

**[13]** HOME SEER, *Built a better Smart Home System.*

[https://homeseer.com/, octubre de 2017].

**[14]** DOMOTICZ, *Control at your finger tips.*

[http://domoticz.com/, octubre 2017].

**[15]** IFTTT, *A world that works for you.*

[https://ifttt.com/, octubre 2017].

**[16]** ECLIPSE, *Kura*

[https://www.eclipse.org/kura/, octubre 2017]

**[17]** IBM, *Node-Red*

[https://nodered.org/, octubre 2017]

**[18]** TIBCO, *Flogo*

[http://www.flogo.io/m octubre 2017]

**[19]** HIVE MQ, *Enterprise MQTT broker.*

[https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe, novembre 2017].

**[20]** CAOP, *RFC 7252 Constrained Application Protocol.*

[http://coap.technology/, novembre 2017].

**[21]** DB- ENGINES, *Knowledge Base of Relational and NoSQL Database Management Systems.*

[https://db-engines.com/en/, novembre 2017].

**[22]** INFLUX DATA, *Time Series Database (TSDB).*

[https://www.influxdata.com/time-series-database/, novembre 2017].

**[23]** BEEBOOM, *10 Best Raspberry Pi 3 Alternatives You Can Buy.*

[https://beebom.com/best-raspberry-pi-3-alternatives/, novembre 2017].

**[24]** ARDUINO STARTER KITS, *Top 6 Arduino Alternatives of 2017.*

[https://www.arduinostarterkits.com/resources/top-arduino-alternatives/, novembre 2017].

**[25]** AMAZON, *ESP8266 Thing - Dev Board.*

[https://www.amazon.com/SparkFun-ESP8266-Thing-Dev-Board/dp/B018011ZEK/ref=as_at?creativeASIN=B018011ZEK&linkCode=w61&imprToken=AsC5TfEYrAfhqSpbYnDnQQ&slotNum=1&tag=arduinostarterkits-20, novembre 2017].

**[26]** ADAFRUIT, *Arduino MKR1000.*

[https://www.adafruit.com/product/3156, novembre 2017].

**[27]** ECLIPSE PAHO, *Classes and Namespaces* file.

[http://www.eclipse.org/paho/files/jsdoc/Paho.MQTT.Client.html, desembre 2017].

**[28]** EUROPEAN COMISSION – ENVIRONMENT, *The RoHS Directive.*

[http://ec.europa.eu/environment/waste/rohs_eee/index_en.htm, 15 de desembre 2017].

**[29]** CARL DOMBROWSKI, *The top 5 best Chatbot and Natural Language Processing Tools to Build Ai for your Business.*

[https://towardsdatascience.com/the-top-5-best-chatbot-and-natural-language-processing-to-build-ai-for-your-business-3efea313d8db, desembre 2017].

**[30]** NODE.JS, *Entorno de ejecución para JavaScript.*

[https://nodejs.org/es/, novembre i desembre de 2017]

**[31]** WWF, *Observatorio de la Electricidad Enero 2016.*

[http://awsassets.wwf.es/downloads/oe_ene_2016.pdf, gener de 2017]

Treball de Fi de Grau

**Grau en Enginyeria en Tecnologies Industrials**

# Disseny d'un sistema domòtic per a un habitatge unifamiliar

**ANNEX A: CODI FINAL**
**ANNEX B: CODI DE LA PRIMERA VERSIÓ DEL SERVIDOR AMB DJANGO**
**ANNEX C: IMATGES DE L'EVOLUCIÓ DE LA INTERFÍCIE**
**ANNEX D: FLOWS DE NODE-RED SENSE UTILITZAR LA COMUNICACIÓ MQTT**
**ANNEX E: OPTIMITZACIÓ DE L'AMPLA DE BANDA**


**Autor:** Marc Borrell Roig
**Director:** Lluís Solano Albajes

**Convocatòria:** Gener de 2018


Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

# SUMARI

**SUMARI** 1

**A. ANNEX A. CODI FINAL** 4

A.1. Codi dels perifèrics: interruptors 5

A.2. Codi del node pròpi 9

A.2.1. Package.json 9

A.2.2. Progressbar.htm 10

A.2.3. Progressbar.js 11

A.3. Codi de la pàgina d'inici 13

A.3.1. HTML 13

A.3.2. CSS 16

A.3.3. JavaScript 17

A.4. Codi de la pàgina d'interruptors 19

A.4.1. HTML 19

A.4.2. CSS 25

A.4.3. JavaScript 27

A.5. Codi de la pàgina de sensors 33

A.5.1. HTM 33

A.5.2. CSS 39

A.5.3. JavaScript 40

A.6. Codi de la pàgina base 50

A.6.1. HTML 50

A.6.2. CSS 52

**B. ANNEX B. CODI DE LA PRIMERA VERSIÓ DEL SERVIDOR AMB DJANGO** 54

B.1. Main.py 55

B.2. Views.py 59

B.3. Urls.py 62

B.4. Models.py 63

**C. ANNEX C. IMATGES DE L'EVOLUCIÓ DE LA INTERFÍCIE** 64

C.1. Primera versió 65

C.2. Interfície final 68

C.3. Versions alternatives de la interfície d'interruptors 70

**D. ANNEX D. FLOWS DE NODE-RED SENSE UTILITZAR LA**

**COMUNICACIÓ MQTT** _____ **71**

**E. ANNEX D. OPTIMITZACIÓ DE L'AMPLA DE BANDA** _____ **72**

# A. ANNEX A. CODI FINAL

## A.1. Codi dels perifèrics: interruptors

```
#include <ESP8266WiFi.h>
#include "PubSubClient.h"

#define MQTT_SERVER "192.168.1.20"
const char* ssid = "ONO963335";
const char* password = "*******l";

const int switchPin = 5;
const int relayPin = 4;

int buttonState = 0;
int lastButtonState = 0;

char* lightTopic = "switches/llum_p";
bool state;

WiFiClient wifiClient;

void callback(char* topic, byte* payload, unsigned int length) {
   state = ((char)payload[0] == 't');
   Serial.println(state);
   if (state){
     digitalWrite(relayPin,HIGH);
    }
    else{
     digitalWrite(relayPin,LOW);
```

```
    }
}
PubSubClient client(MQTT_SERVER, 1883, callback, wifiClient);

void setup() {
  pinMode(switchPin, INPUT_PULLUP);
  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin,LOW);

  Serial.begin(115200);
  delay(100);

  WiFi.begin(ssid, password);
  reconnect();
     delay(2000);
}
void loop(){

  if (!client.connected() && WiFi.status() == 3) {reconnect();}
  client.loop();
  buttonState = digitalRead(switchPin);

  if (buttonState != lastButtonState) {
    state = !state;
    if (state){
      client.publish(lightTopic, "true");
      Serial.println("light on");
    }
```

```
     else{
      client.publish(lightTopic, "false");
      Serial.println("light off");
     }
    }
  delay(40);
  lastButtonState = buttonState;
  delay(10);
}
void reconnect() {
  if(WiFi.status() != WL_CONNECTED){
    Serial.print("Connecting to ");
    Serial.println(ssid);
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
  }
  if(WiFi.status() == WL_CONNECTED){
    while (!client.connected()) {
      Serial.print("Attempting MQTT connection...");

      String clientName;
      clientName += "esp8266-";
```

```
      uint8_t mac[6];
      WiFi.macAddress(mac);
      clientName += macToStr(mac);
            if (client.connect((char*) clientName.c_str())) {
        Serial.print("\tMTQQ Connected");
        client.subscribe(lightTopic);
      }
      else{Serial.println("\tFailed."); abort();}
    }
  }
}
String macToStr(const uint8_t* mac){
  String result;
  for (int i = 0; i < 6; ++i) {
    result += String(mac[i], 16);
    if (i < 5){
      result += ':';
    }
  }
  return result;
}
```

## A.2. Codi del node pròpi

### A.2.1. Package.json

```
{
  "name": "node-red-contrib-progressbar",
  "version": "1.1.0",
  "description": "De moment no hi ha cap descripció",
  "main": "progressbar.js",
  "directories": {
    "test": "tests"
  },
  "node-red" : {
        "nodes": {
            "progressbar": "progressbar.js"
        }
  },
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "progressbar"
  ],
  "author": "Marc Borrell Roig",
  "license": "ISC"
}
```

### A.2.2. Progressbar.htm

```
<script type="text/javascript">
    RED.nodes.registerType('progressbar',{
        category: 'marc',
        color: '#f9f940',
        defaults: {
            name: {value:""},
            stepup: {value:333,required:false,validate:RED.validators.number()},
            stepdown: {value:329,required:false,validate:RED.validators.number()}
        },
        inputs:1,
        outputs:1,
        icon: "progressbar.png",
        label: function() {
            return this.name||"progressbar";
        }
    });
</script>
<script type="text/x-red" data-template-name="progressbar">
    <div class="form-row">
        <label for="node-input-name"><i class="icon-tag"></i> Name</label>
        <input type="text" id="node-input-name" placeholder="Name">
    </div>
    <div class="form-row">
        <label for="node-input-stepup"><i class="icon-tag"></i> Stepup</label>
        <input type="text" id="node-input-stepup" placeholder="Name">
    </div>
    <div class="form-row">
```

```
        <label for="node-input-stepdown"><i class="icon-tag"></i> Stepdown</label>
        <input type="text" id="node-input-stepdown" placeholder="Name">
    </div>
</script>
<script type="text/x-red" data-help-name="progressbar">
    <p>A simple node that converts the message payloads into all lower-case characters</p>
</script>
```

### A.2.3. Progressbar.js

```
module.exports = function(RED) {
    function progressbar(config) {
        RED.nodes.createNode(this,config);
        var node = this;
        var globalContext = node.context().global;

        var progressbar = globalContext.get('progressbar') || {};
        progressbar.value = progressbar.value || 0;
        progressbar.stepup = config.stepup; // in milliseconds
        progressbar.stepdown = config.stepdown; // in milliseconds

        var timeoutid
        progressbar.refresh = function(msg){
            if (progressbar.value > 100 || progressbar.value < 0 || progressbar.action == "stop"){
            if (progressbar.value > 100) progressbar.value = 100;
            if (progressbar.value <0 ) progressbar.value = 0;
            clearTimeout(timeoutid);
            globalContext.set('progressbar', progressbar);
            node.status({});
```

```
        }else{
        if (progressbar.action == "up"){
          progressbar.value -= 1;
          clearTimeout(timeoutid);
          timeoutid = setTimeout(progressbar.refresh, progressbar.stepup, msg);
        }
        if (progressbar.action == "down"){
          progressbar.value += 1;
          clearTimeout(timeoutid);
          timeoutid = setTimeout(progressbar.refresh, progressbar.stepdown, msg);
        }
        }
        msg.payload = progressbar.value;
        node.send(msg);
    };
    var node = this;
    node.on('input', function(msg) {
        node.status({fill:"green",shape:"dot",text:"working"});
        progressbar.action = msg.topic.split("/").pop() || "stop";
        progressbar.refresh(msg);
    });
  }
  RED.nodes.registerType("progressbar",progressbar);
}
```

## A.3. Codi de la pàgina d'inici

### A.3.1. HTML

```
<link rel="stylesheet" href="css/home.css">

<style type="text/css">

  body {

    background: url("img/home.jpg") no-repeat center center fixed;

    background-color: #cccccc;

    -webkit-background-size: cover;

    -moz-background-size: cover;

    -o-background-size: cover;

    background-size: cover;

  }

</style>

<div class="container">

  <div class="card-columns">

    <div class="card d-inline-block">

      <img class="card-img-top" src="img/background_min.jpg" alt="Card image cap">

      <div class="card-body">

        <h4 class="card-title">Interruptors</h4>

        <p class="card-text">Des d'aquí podràs controlar tots els teus dispositius elèctrics de la teva habitació. Interruptors, persianes, endolls...</p>

        <a href="/switches" class="btn btn-primary">Anar-hi!</a>

      </div>

    </div>


    <div class="card d-inline-block">

      <img class="card-img-top" src="img/sensors_min.jpg" alt="Card image cap">

      <div class="card-body">
```

```
      <h4 class="card-title">Sensors</h4>

      <p class="card-text">Vols saber quin dia farà avui? En aquesta pàgina podràs trobar tota classe d'informació relacionada amb la meteorologia</p>

      <p class="card-text"><small id="temp_card" class="text-muted">Tint 25ºC Text 23ºC</small></p>

      <a href="/sensors" class="btn btn-primary">El Temps</a>

    </div>

  </div>


  <div class="card d-inline-block">

    <img class="card-img-top" src="img/food_min.jpg" alt="Card image cap">

    <div class="card-body">

      <h4 class="card-title">Cuina</h4>

      <p class="card-text">Tot el que et cal per ser un bon cuiner. Receptes, compres, i gestió d'aliments. Dóna un cop d'ull per saber-ne més</p>

      <a href="#" class="btn btn-primary">Anar a la cuina</a>

    </div>

  </div>


  <div class="card d-inline-block">

    <img class="card-img-top" src="img/settings_min.jpg" alt="Card image cap">

    <div class="card-body">

      <h4 class="card-title">Configuració</h4>

      <p class="card-text">Aquí hi ha tot el que necessites per gestionar la casa domòtica. Per exemple, es poden configurar els usuaris, les hores de pujar i baixar les persianes, etc</p>

      <a href="/config" class="btn btn-primary">Configurar</a>

    </div>

  </div>
```

```
<div class="card d-inline-block">
  <img class="card-img-top" src="img/ui_min.png" alt="Card image cap">
  <div class="card-body">
    <h4 class="card-title">Dashboard</h4>
    <p class="card-text">Aquesta és una interfície de prova antiga del dormitori. Controla tots els interruptors, persianes i gràfiques de temperatura. Aviat quedarà antiquada!</p>
    <a href="/sensors" class="btn btn-primary">Ves-hi igualment!</a>
  </div>
</div>
<div class="card d-inline-block">
  <img class="card-img-top" src="img/admin_min.png" alt="Card image cap">
  <div class="card-body">
    <h4 class="card-title">Admin</h4>
    <p class="card-text">Zona per desenvolupadors i usuaris experts! És el lloc on es poden fer ajustos del servidor i configurar els processos que s'executen. Gestionar amb precaució!</p>
    <a href="/sensors" class="btn btn-danger">Administrar</a>
  </div>
</div>
<div class="card d-inline-block">
  <img class="card-img-top" src="img/bank_min.jpg" alt="Card image cap">
  <div class="card-body">
    <h4 class="card-title">Influx DB</h4>
    <p class="card-text">Tingues un control total sobre les despeses de cada mes. Aquí i trobaràs gràfiques, ingressos i pagaments de cada trimestre</p>
    <p class="card-text"><small class="text-muted">Últim moviment fa 3 mins</small></p>
  </div>
</div>

{{ info }}
```

```
  </div>
</div>

<script type="text/javascript" src="js/home.js"></script>
```

### A.3.2. CSS

```
@media (min-width: 30em) {
    .card-columns {
        -webkit-column-count: 2;
        -moz-column-count: 2;
        column-count: 2;
    }
}
@media (min-width: 48em) {
    .card-columns {
        -webkit-column-count: 3;
        -moz-column-count: 3;
        column-count: 3;
    }
}
@media (min-width: 62em) {
    .card-columns {
        -webkit-column-count: 4;
        -moz-column-count: 4;
        column-count: 4;
    }
}
```

```
@media (min-width: 75em) {
    .card-columns {
        -webkit-column-count: 5;
        -moz-column-count: 5;
        column-count: 5;
    }
}
```

### A.3.3. JavaScript

```
$('#home').addClass('active');
var ws;
var wsUri = "ws:";
var loc = window.location;
console.log(loc);
if (loc.protocol === "https:") { wsUri = "wss:"; }
wsUri += "//" + loc.host + loc.pathname.replace("home","ws/home");

function wsConnect() {
    console.log("connect",wsUri);
    ws = new WebSocket(wsUri);
    ws.onmessage = function(event) {
        var data = JSON.parse(event.data);
        $('#temp_card').text("Tint "+data.temp+"ºC   Hint "+data.humidity+"%");
    }
    ws.onopen = function() {
        $('#disconnected').hide();
        $('#connected').show();
    }
```

```
    ws.onclose = function() {
        $('#connected').hide();
        $('#disconnected').show();

        setTimeout(wsConnect,3000);
    }
}
function doit(m) {
    if (ws) { ws.send(m); }
}
function sendstate(){
  var payload = {
  p:document.getElementById("1").checked,
  s:document.getElementById("2").checked,
  e:document.getElementById("3").checked,
  a:document.getElementById("4").checked
  }
  doit(JSON.stringify(payload));
  console.log(payload);
}
wsConnect();
```

# A.4. Codi de la pàgina d'interruptors

## A.4.1. HTML

```
<link rel="stylesheet" type="text/css" href="css/switches.css">
<style>
  body {
    background: url("img/background.jpg") no-repeat center center fixed;
    background-color: #cccccc;
    -webkit-background-size: cover;
    -moz-background-size: cover;
    -o-background-size: cover;
    background-size: cover;

  }
</style>
<div class="container" style="overflow:auto;">
  <nav aria-label="breadcrumb" role="navigation">
    <ol class="breadcrumb">
      <li class="breadcrumb-item"><a href="#">Totes</a></li>
      <li class="breadcrumb-item"><a href="#">Dormitoris</a></li>
      <li class="breadcrumb-item active" aria-current="page">Marc</li>
    </ol>
  </nav>
  <div class="card-columns">
    <div class="card">
      <div class="card-header text-center" role="tab">
        <h4 class="mb-0">
          <a data-toggle="collapse" href="#Interruptors">Interruptors</a>
```

```
      </h4>
    </div>
    <div class="collapse show" id="Interruptors">
    <ul class="list-group list-group-flush">
      <li class="list-group-item">
        <div class="input-group">
          <div class="input-group-addon">Llum Principal</div>
          <input class="input-group-addon" id="llum_p" type="checkbox" name="checkbox">
          <button name="trash" type="button" class="btn btn-danger input-group-addon"><img class="icon" src="svg/trash.svg"></button>
        </div>
      </li>
      <li class="list-group-item">
        <div class="input-group">
          <div class="input-group-addon">Llum secundaria</div>
          <input class="input-group-addon" id="llum_s" type="checkbox" name="checkbox">
          <button name="trash" type="button" class="btn btn-danger input-group-addon"><img class="icon" src="svg/trash.svg"></button>
        </div>
      </li>
      <li class="list-group-item">
        <div class="input-group">
          <div class="input-group-addon">Endolls</div>
          <input class="input-group-addon" id="plugs" type="checkbox" name="checkbox">
          <button name="trash" type="button" class="btn btn-danger input-group-addon"><img class="icon" src="svg/trash.svg"></button>
        </div>
      </li>
      <li class="list-group-item">
        <div class="input-group">
```

```
          <div class="input-group-addon">Aire condicionat</div>

          <input class="input-group-addon btn-success" id="ac" type="checkbox" name="checkbox">

          <button name="trash" type="button" class="btn btn-danger input-group-addon"><img class="icon" src="svg/trash.svg"></button>

        </div>

       </li>

     </ul>

     </div>

   </div>


   <div class="card">

    <div class="card-header text-center" role="tab" id="headingOne">

     <h4 class="mb-0">

      <a data-toggle="collapse" href="#Persiana">Persiana</a>

     </h4>

    </div>

    <div class="collapse show" id="Persiana">

     <ul class="list-group list-group-flush">

      <li class="list-group-item">

       <div class="btn-group w-100" role="group">

        <button type="button" name="button" id="up" class="btn btn-primary col">Pujar</button>

        <button type="button" name="button" id="stop" class="btn btn-danger col">Stop</button>

        <button type="button" name="button" id="down" class="btn btn-primary col">Baixar</button>

       </div>

      <div class="progress mt-3">

          <div id='progressbar' class="progress-bar progress-bar-striped" role="progressbar" aria-valuenow="75" aria-valuemin="0" aria-valuemax="100" style="width: 75%"></div>

      </div>

      </li>
```

```
        <li class="list-group-item">
          <div class="input-group clockpicker">
            <div class="input-group-addon">Pujar la persiana</div>
            <input id="uptime" type="button" class="form-control" value="7:35">
            <span class="input-group-addon">
            <img class="" src="svg/clock.svg">
            </span>
          </div>
          <div class="dropdown pt-2">
            <button class="btn btn-secondary dropdown-toggle" type="button" id="dropdownMenuButton"
data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
              Més opcions
            </button>
            <div class="dropdown-menu" aria-labelledby="dropdownMenuButton">
              <a class="dropdown-item" name="clockpicker" value="5006">Final de Nit</a>
              <a class="dropdown-item" name="clockpicker" value="5000">Alba</a>
              <a class="dropdown-item" name="clockpicker" value="5003">Matinada</a>
              <a class="dropdown-item" name="clockpicker" value="5002">Migdia</a>
              <a class="dropdown-item" name="clockpicker" value="5004">Capvespre</a>
              <a class="dropdown-item" name="clockpicker" value="5001">Posta</a>
              <a class="dropdown-item" name="clockpicker" value="5005">Nit</a>
            </div>
          </div>
        </li>
        <li class="list-group-item">
          <div class="input-group clockpicker">
            <div class="input-group-addon">Baixar la persiana</div>
            <input id="downtime" type="button" class="form-control" value="19:35">
            <span class="input-group-addon">
```

```
            <img class="" src="svg/clock.svg">
            </span>
          </div>
          <div class="dropdown pt-2">
            <button class="btn btn-secondary dropdown-toggle" type="button" id="dropdownMenuButton"
data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
              Més opcions
            </button>
            <div class="dropdown-menu" aria-labelledby="dropdownMenuButton">
              <a class="dropdown-item" name="clockpicker" value="5006">Final de Nit</a>
              <a class="dropdown-item" name="clockpicker" value="5000">Alba</a>
              <a class="dropdown-item" name="clockpicker" value="5003">Matinada</a>
              <a class="dropdown-item" name="clockpicker" value="5002">Migdia</a>
              <a class="dropdown-item" name="clockpicker" value="5004">Capvespre</a>
              <a class="dropdown-item" name="clockpicker" value="5001">Posta</a>
              <a class="dropdown-item" name="clockpicker" value="5005">Nit</a>
            </div>
          </div>
        </li>
      </ul>
    </div>
  </div>

  <div class="card">
    <div class="card-header text-center" role="tab" id="headingOne">
      <h4 class="mb-0">
        <a data-toggle="collapse" href="#modes">Modes</a>
      </h4>
    </div>
```

```
<div class="collapse show" id="modes">
 <ul class="list-group list-flush">
  <li class="list-group-item">
   <div class="input-group">
    <div class="input-group-addon">Sensor mov.</div>
    <input class="input-group-addon btn-success" id="llum_auto" type="checkbox" name="checkbox">
    <button name="trash" type="button" class="btn btn-danger input-group-addon"><img class="icon" src="svg/trash.svg"></button>
   </div>
   <div class="btn-group d-flex justify-content-center pt-2" data-toggle="buttons">
    <label class="btn btn-outline-dark btn-ht-xs">
     <input id="llum_p_auto" name="select" type="checkbox" autocomplete="off"> Principal
    </label>
    <label class="btn btn-secondary">
     <input id="llum_s_auto" name="select" type="checkbox" autocomplete="off"> Indirecte
    </label>
   </div>
  </li>
  <li class="list-group-item">
   <div class="input-group">
    <div class="input-group-addon">Mode Nit</div>
    <input class="input-group-addon btn-success" id="mode_nit" type="checkbox" name="checkbox">
    <button name="trash" type="button" class="btn btn-danger input-group-addon"><img class="icon" src="svg/trash.svg"></button>
   </div>
  </li>
  <li class="list-group-item">
   <div class="input-group">
```

```
            <div class="input-group-addon">Mode Vacances</div>

            <input class="input-group-addon btn-success" id="mode_vacances" type="checkbox" name="checkbox">

            <button name="trash" type="button" class="btn btn-danger input-group-addon"><img class="icon" src="svg/trash.svg"></button>

          </div>

        </li>

      </ul>

    </div>

  </div>

 </div>

</div>


<script src="js/bootstrap-switch.js"></script>

<script src="js/paho-mqtt.js"></script>

<script src="js/bootstrap-clockpicker.min.js"></script>

<script type="text/javascript" src="js/switches.js"></script>
```

### A.4.2. CSS

```
@media (min-width: 30em) {

   .card-columns {

      -webkit-column-count: 1;

      -moz-column-count: 1;

      column-count: 1;

   }

}

@media (min-width: 50em) {

   .card-columns {

      -webkit-column-count: 2;
```

```
        -moz-column-count: 2;
        column-count: 2;
    }
}
@media (min-width: 60em) {
    .card-columns {
        -webkit-column-count: 3;
        -moz-column-count: 3;
        column-count: 3;
    }
}
a{
  color: black;
  text-decoration: none;
}
.white{
  color:white;
}
.input-group{
  margin-top: 4px;
}
.icon {
  height: 10px;
  width: 10px;
  -webkit-filter: invert(100%);
  filter: invert(100%);
}
```

```
.bootstrap-switch{
  margin-top: 0px;
  border-radius: 0px;
}
.bootstrap-switch-container{
  height: 36px;
}
.mobileHoverFix:hover,
.mobileHoverFix.hover{
  outline: none !important;
  box-shadow: none;
}
```

### A.4.3. JavaScript

```
$('#switches').addClass('active');
$('[name="trash"]').on('click', function(event, state) {
  console.log($(this).parent().parent().remove());
});
$("button").on("touchstart", function(){
    $(this).removeClass("mobileHoverFix");
});
$("button").on("touchend", function(){
    $(this).addClass("mobileHoverFix");
});

// ------------------------- MQTT ----------------------------------------
function guid() {
  function s4() {return Math.floor((1 + Math.random()) * 0x10000).toString(16).substring(1);}
```

```
  return s4() + s4() + '-' + s4();
}
console.log(guid());
// Create a client instance
client = new Paho.MQTT.Client(location.hostname,1884,guid());

// set callback handlers
client.onConnectionLost = onConnectionLost;
client.onMessageArrived = onMessageArrived;

// connect the client
client.connect({onSuccess:onConnect,reconnect:true});

// called when the client connects
function onConnect() {
  // Once a connection has been made, make a subscription and send a message.
  console.log("connected")
  $('#disconnected').hide();
  $('#connected').show();
  $("[name='checkbox']").bootstrapSwitch('indeterminate',false);
  $("[name='checkbox']").bootstrapSwitch('readonly',false);
  $("[name='button']").removeClass('disabled');

  client.subscribe("switches/#");
  client.subscribe("progressbar");
  client.subscribe("schedule/#");
  client.subscribe("select/#");
}
```

```
// called when the client loses its connection
function onConnectionLost(responseObject) {
  if (responseObject.errorCode !== 0) {
    console.log("onConnectionLost:"+responseObject.errorMessage);
    $('#connected').hide();
    $('#disconnected').show();
    $("[name='checkbox']").bootstrapSwitch('indeterminate',true);
    $("[name='checkbox']").bootstrapSwitch('readonly',true);
    $("[name='button']").addClass('disabled');
  }
}
// called when a message arrives
function onMessageArrived(message) {
  switch (message.destinationName.split("/")[0]){
  case "switches":
    $('#'+message.destinationName.split("/").pop()).bootstrapSwitch('state',
eval(message.payloadString), true);
    break;
  case "progressbar":
    $('#progressbar').width(message.payloadString+'%');
    break;
  case "schedule":
    var id = message.destinationName.split("/").pop();
    if (id=="uptimetext" || id=="downtimetext"){
      id = id.replace('text','');
      hores = Math.floor(eval(message.payloadString)/60);
      min = (eval(message.payloadString)%60);
      //if (hores   < 10) {hores   = "0"+hores;}
      if (min < 10) {min = "0"+min;}
```

```
      console.log(hores+":"+min);
      $('#'+id).val(hores+":"+min);
    }
    break;
  case "select":
    console.log(message.payloadString=="true");
    if(message.payloadString=="true"){
      $('#'+message.destinationName.split("/").pop()).parent().addClass("active");
    }
    break;
  }
  $("#mode_nit").bootstrapSwitch('indeterminate',true);
  $("#mode_nit").bootstrapSwitch('readonly',true);
  $("#mode_vacances").bootstrapSwitch('indeterminate',true);
  $("#mode_vacances").bootstrapSwitch('readonly',true);
}
$('[name="select"]').on('change', function(event, state) {
  //console.log(this.checked);
  payload = new Paho.MQTT.Message(String(this.checked));
  payload.destinationName = "select/"+this.id;
  payload.qos = 1;
  payload.retained = true;
  client.publish(payload);
  console.log(payload);
});
$('[name="button"]').on('click', function(event, state) {
  //console.log(this.checked);
  client.publish("buttons/"+this.id,'1');
```

```
});
$('[name="checkbox"]').on('switchChange.bootstrapSwitch', function(event, state) {
  //console.log(this.checked);
  payload = new Paho.MQTT.Message(String(this.checked));
  payload.destinationName = "switches/"+this.id;
  payload.qos = 1;
  payload.retained = true;
  client.publish(payload);
});
// ------------------------------------ CONFIG SWITCH ----------------------------------------

$('#llum_auto').bootstrapSwitch('onText', 'AUTO');
$('#llum_auto').bootstrapSwitch('offText', 'MANUAL');
$('#llum_auto').bootstrapSwitch('labelWidth', 15);
$('#llum_auto').bootstrapSwitch('onColor', 'success');
$('#llum_auto').bootstrapSwitch('offColor', 'warning');

// ------------------------------------ CLOCKPICKER ----------------------------------------

$('.clockpicker').clockpicker({
  placement: 'bottom', // clock popover placement
  align: 'left',       // popover arrow align
  donetext: 'Done',     // done button text
  autoclose: true,    // auto close when minute is selected
  vibrate: true        // vibrate the device when dragging clock hand
  });
$('.clockpicker').clockpicker().find('input').change(function(){
  payload = new Paho.MQTT.Message("on_override "+this.value);
```

```
  payload.destinationName = "schedule/"+this.id;
  payload.qos = 1;
  payload.retained = true;
  client.publish(payload);
});
$('[name="clockpicker"]').on('click', function(event, state) {
//console.log($(this).parent().parent().prev().find('input').attr('id'));
//console.log($(this).attr('value'));
payload = new Paho.MQTT.Message("on_override "+$(this).attr('value'));
payload.destinationName = "schedule/"+$(this).parent().parent().prev().find('input').attr('id');
payload.qos = 1;
payload.retained = true;
client.publish(payload);
});
```

## A.5. Codi de la pàgina de sensors

### A.5.1. HTM

```
<link rel="stylesheet" type="text/css" href="css/sensors.css">
<style>
  body {
    background: url("img/sensors.jpg") no-repeat center center fixed;
    background-color: #fff;
    -webkit-background-size: cover;
    -moz-background-size: cover;
    -o-background-size: cover;
    background-size: cover;
  }
</style>
<div class="container">
  <div class="row justify-content-center">
    <div class="card m-1 col-lg-12 col-sm-12 col-12"  >
      <div class="card-header text-center" role="tab">
        <h4 class="mb-0">
          <a data-toggle="collapse" href="#temp">Temperatura</a>
        </h4>
      </div>
      <div class="card-body collapse show" id="temp">
        <div class="btn-group" data-toggle="buttons">
          <span class="input-group-addon">Temps de refresc: </span>
          <label class="btn btn-primary">
            <input type="radio" name="refresh" value='1000'> 1s
          </label>
```

```
    <label class="btn btn-primary active">
      <input type="radio" name="refresh" value='5000'> 5s
    </label>
    <label class="btn btn-primary">
      <input type="radio" name="refresh" value='30000'> 30s
    </label>
    <label class="btn btn-primary">
      <input type="radio" name="refresh" value='60000' checked> 1m
    </label>
    <label class="btn btn-primary">
      <input type="radio" name="refresh" value='600000'> 10m
    </label>
  </div>
  <div class="input-group">
    <span class="input-group-addon d-inline-block">SELECT</span>
    <input type="text" class="form-control" id='select' value='mean("Temperatura")'>
    <!-- <span class="input-group-addon d-inline-block">AS</span>
    <input type="text" class="form-control" id='as' value='"data"'>-->
    <span class="input-group-addon d-inline-block">FROM</span>
    <input type="text" class="form-control" id='from' value='"SensorData"."autogen"."temp&hum"'>
    <span class="input-group-addon d-inline-block">WHERE</span>
    <input type="text" class="form-control" id='where' value="time > now() - 1d">
    <span class="input-group-addon d-inline-block">GROUP BY</span>
    <input type="text" class="form-control" id='groupby' value="time(1h)">
    <!-- <span class="input-group-addon d-inline-block">FILL</span>
    <input type="text" class="form-control" id='fill' value="null"> -->
    <span class="input-group-btn">
      <button class="btn btn-secondary" type="button" name="query">Go!</button>
```

```
        </span>
       </div>
       <div class="row" style="height: 300px;">
        <div class="col">
         <canvas id="chart1"></canvas>
        </div>
       </div>
      </div>
     </div>

    <div class="card m-1 col-lg-12 col-sm-12 col-12">
     <div class="card-header text-center" role="tab">
      <h4 class="mb-0">
       <a data-toggle="collapse" href="#hum">Humitat</a>
       <button id="" type="button" class="btn btn-outline-secondary btn-sm float-right" disabled>Hide</button>
      </h4>

     </div>
     <div class="card-body collapse show" id="hum">
      <div class="btn-group" data-toggle="buttons">
       <span class="input-group-addon">Temps de refresc: </span>
       <label class="btn btn-primary">
        <input type="radio" name="refresh" value='1000'> 1s
       </label>
       <label class="btn btn-primary active">
        <input type="radio" name="refresh" value='5000'> 5s
       </label>
       <label class="btn btn-primary">
```

```
    <input type="radio" name="refresh" value='30000'> 30s
   </label>
   <label class="btn btn-primary">
    <input type="radio" name="refresh" value='60000' checked> 1m
   </label>
   <label class="btn btn-primary">
    <input type="radio" name="refresh" value='600000'> 10m
   </label>
  </div>
  <div class="input-group">
   <span class="input-group-addon d-inline-block">SELECT</span>
   <input type="text" class="form-control" id='select' value='mean("Humitat")'>
   <!-- <span class="input-group-addon d-inline-block">AS</span>
   <input type="text" class="form-control" id='as' value='"data"'>-->
   <span class="input-group-addon d-inline-block">FROM</span>
   <input type="text" class="form-control" id='from' value='"SensorData"."autogen"."temp&hum"'>
   <span class="input-group-addon d-inline-block">WHERE</span>
   <input type="text" class="form-control" id='where' value="time > now() - 1d">
   <span class="input-group-addon d-inline-block">GROUP BY</span>
   <input type="text" class="form-control" id='groupby' value="time(1h)">
   <!-- <span class="input-group-addon d-inline-block">FILL</span>
   <input type="text" class="form-control" id='fill' value="null"> -->
   <span class="input-group-btn">
    <button class="btn btn-secondary" type="button" name="query">Go!</button>
   </span>
  </div>
  <div class="row" style="height: 300px;">
   <div class="col">
```

```
        <canvas id="chart2"></canvas>
      </div>
    </div>
  </div>
</div>

<div class="card m-1 col-12">
  <div class="card-header text-center" role="tab">
    <h4 class="mb-0">
      <a data-toggle="collapse" href="#switch">Interruptors</a>
    </h4>
  </div>
  <div class="card-body collapse show" id="switch">
    <div class="btn-group" data-toggle="buttons">
      <span class="input-group-addon">Temps de refresc: </span>
      <label class="btn btn-primary">
        <input type="radio" name="refresh" value='1000'> 1s
      </label>
      <label class="btn btn-primary active">
        <input type="radio" name="refresh" value='5000' > 5s
      </label>
      <label class="btn btn-primary">
        <input type="radio" name="refresh" value='30000'> 30s
      </label>
      <label class="btn btn-primary">
        <input type="radio" name="refresh" value='60000' checked> 1m
      </label>
      <label class="btn btn-primary">
```

```
        <input type="radio" name="refresh" value='600000'> 10m
      </label>
    </div>
    <div class="input-group">
      <span class="input-group-addon d-inline-block">SELECT</span>
      <input type="text" class="form-control" id='select' value='mean("State")'>
      <!-- <span class="input-group-addon d-inline-block">AS</span>
      <input type="text" class="form-control" id='as' value='"data"'>-->
      <span class="input-group-addon d-inline-block">FROM</span>
      <input type="text" class="form-control" id='from' value='"SensorData"."autogen"."switches"'>
      <span class="input-group-addon d-inline-block">WHERE</span>
      <input type="text" class="form-control" id='where' value="time > now() - 1w">
      <span class="input-group-addon d-inline-block">GROUP BY</span>
      <input type="text" class="form-control" id='groupby' value='time(1d), "Dispositiu"'>
      <!-- <span class="input-group-addon d-inline-block">FILL</span>
      <input type="text" class="form-control" id='fill' value="null"> -->
      <span class="input-group-btn">
        <button class="btn btn-secondary" type="button" name="query">Go!</button>
      </span>
    </div>
    <div class="row" style="height: 200px;">
      <div class="col">
        <canvas id="chart3"></canvas>
      </div>
    </div>
  </div>
</div>
```

```
  </div>
</div>
<script src="js/moment.js"></script>
<script src="js/chart.js"></script>
<script src="js/sensors.js"></script>
```

### A.5.2. CSS

```
@media (min-width: 30em) {
    .card-columns {
        -webkit-column-count: 2;
        -moz-column-count: 2;
        column-count: 2;
    }
}
@media (min-width: 48em) {
    .card-columns {
        -webkit-column-count: 3;
        -moz-column-count: 3;
        column-count: 3;
    }
}
@media (min-width: 62em) {
    .card-columns {
        -webkit-column-count: 4;
        -moz-column-count: 4;
        column-count: 4;
    }
}
```

### A.5.3. JavaScript

```
$('#sensors').addClass('active');

// ------------------------------ CHARTS.JS ------------------------------------

var timeFormat = "YYYY-MM-DD'T'HH:mm:ss.SSS'Z'";


var chart1 = new Chart($('#chart1')[0].getContext('2d'), {
    // The type of chart we want to create
    type: 'line',
    // The data for our dataset
    data: {
        //labels: [],
        datasets: [{
            tag: "Temperatura",
            label: "Temperatura",
            backgroundColor: 'rgba(55, 163, 236,0.5)',
            borderColor: 'rgb(55, 163, 236)',
            borderWidth: 2,
            data: [],
            //pointRadius:0,
            tension: 0.2,
        }]
    },
    // Configuration options go here
    options: {
            maintainAspectRatio: false,
            layout: {
                    padding: {
                        left: 5,
```

```
                right: 0,
                top: 0,
                bottom: 0
            }
        },
    scales: {
      xAxes: [{
        type: "time",
        time: {
          format: timeFormat,
          distribution: 'linear',

          //round: 'day'
          //tooltipFormat: 'HH'
          //unit: 'hour',
        },
        scaleLabel: {
          display: false,
          labelString: 'Date'
        }
      }, ],
      yAxes: [{
        scaleLabel: {
          display: false,
          labelString: 'value'
        }
      }]
    },
```

```
    }
});
var chart2 = new Chart($('#chart2')[0].getContext('2d'), {
    // The type of chart we want to create
    type: 'line',

    // The data for our dataset
    data: {
        //labels: [],
        datasets: [{
            tag: "Humitat",
            label: "Humitat",
            backgroundColor: 'rgba(55, 163, 236,0.5)',
            borderColor: 'rgb(55, 163, 236)',
            data: [],
            borderWidth: 2,
            //pointRadius:0,
            tension: 0.1,
        }]
    },

    // Configuration options go here
    options: {
            maintainAspectRatio: false,
            layout: {
                    padding: {
                        left: 5,
```

```
                    right: 0,
                    top: 0,
                    bottom: 0
                }
            },
        scales: {
          xAxes: [{
            type: "time",
            time: {
              format: timeFormat,
              //round: 'day'
              //tooltipFormat: 'HH'
              // unit: 'hour'
            },
            scaleLabel: {
              display: false,
              labelString: 'Date'
            }
          }, ],
          yAxes: [{
            scaleLabel: {
              display: false,
              labelString: 'value'
            }
          }]
        },
    }
});
```

```
var chart3 = new Chart($('#chart3')[0].getContext('2d'), {
    // The type of chart we want to create
    type: 'bar',

    // The data for our dataset
    data: {
        //labels: [1,2],
        datasets: [{
            tag: "llum_p",
            label: "Principal",
            backgroundColor: 'rgba(123,202,187,0.5)',
            borderColor: 'rgb(123,202,187)',
            borderWidth: 1,
            data: [],
            tension: 0,
        },{
            tag: "llum_s",
            label: "Secundaria",
            backgroundColor: 'rgba(204,97,85,0.5)',
            borderColor: 'rgb(204,97,85)',
            borderWidth: 1,
            data: [],
            tension: 0,
        },{
            tag: "plugs",
            label: "Endolls",
            backgroundColor: 'rgba(245,203,98,0.5)',
            borderColor: 'rgb(245,203,98)',
```

```
        borderWidth: 1,
        data: [],
        tension: 0,
    },{
        tag: "ac",
        label: "Aire",
        backgroundColor: 'rgba(56,54,125,0.5)',
        borderWidth: 1,
        borderColor: 'rgb(56,54,125)',
        data: [],
        tension: 0,
    }
    ]
},
// Configuration options go here
options: {
        maintainAspectRatio: false,
        scales: {
          xAxes: [{
            type: "time",
            bounds:'ticks',
            gridLines: {
              offsetGridLines: true
            },
            time: {
              format: timeFormat,
              //round: 'day'
              //tooltipFormat: 'HH'
```

```
                unit: 'day'
              },
              scaleLabel: {
                display: false,
                labelString: 'Date'
              }
            }, ],
            yAxes: [{
              ticks: {
                beginAtZero:true,
              },
              scaleLabel: {
                display: false,
                labelString: 'value'
              }
            }]
          },
    }
});

chart1.query = 'SELECT mean("Temperatura") AS "y","time" AS "x"  FROM
"SensorData"."autogen"."temp&hum" WHERE time > now() - 1d GROUP BY time(1h) FILL(null)';

chart1.refreshstep = 60000;

chart1.timeoutid = null;


chart2.query = 'SELECT mean("Humitat") AS "y","time" AS "x" FROM
"SensorData"."autogen"."temp&hum" WHERE time > now() - 1d GROUP BY time(1h) FILL(null)';

chart2.refreshstep = 60000;

chart2.timeoutid = null;
```

```
chart3.query = 'SELECT count("State") AS "y","time" AS "x" FROM "SensorData"."autogen"."switches"
WHERE time > now() - 1w AND "State"=True GROUP BY time(1d), "Dispositiu"';

chart3.refreshstep = 60000;

chart3.timeoutid = null;

function addData(chart, label, data) {

    chart.data.labels.push(label);

    chart.data.datasets.forEach(function(dataset){

        dataset.data.push(data);

    });

    chart.update();

}

function rawData(chart, data) {

    var newdatasets = {};

    var obj;

    var tag;

    for (var i = data.length - 1; i >= 0; i--) {

      tag = data[i].Dispositiu || "default";

      (newdatasets[tag] = newdatasets[tag] || []).push(data[i]);

    }

    for (var i = chart.data.datasets.length - 1; i >= 0; i--) {

      chart.data.datasets[i].data = newdatasets[chart.data.datasets[i].tag] || newdatasets["default"];

    }

    chart.update();

}

// ------------------------------ WEBSOCKETS -----------------------------------

var dades = {};

var ws;

var wsUri = "ws:";

var loc = window.location;
```

```
console.log(loc);
if (loc.protocol === "https:") { wsUri = "wss:"; }
wsUri += "//" + loc.host + loc.pathname.replace("sensors","ws/sensors");

function wsConnect() {
    console.log("connect",wsUri);
    ws = new WebSocket(wsUri);
    ws.onmessage = function(event) {
        dades = JSON.parse(event.data);
        rawData(eval(dades.id),dades.data);
        //$('#temp').text(dades.temp+"ºC");
        //$('#hum').text(dades.humidity+"%");
    }
    ws.onopen = function() {
        $('#disconnected').hide();
        $('#connected').show();
        refresh_charts(chart1);
        refresh_charts(chart2);
        refresh_charts(chart3);
    }
    ws.onclose = function() {
        $('#connected').hide();
        $('#disconnected').show();
        setTimeout(wsConnect,3000);
    }
}

function refresh_charts(chart){
```

```
  console.log(chart.refreshstep);
    var payload = {
          id:chart.chart.canvas.id,
          query: chart.query
       };
    if (ws) ws.send(JSON.stringify(payload));
    chart.timeoutid = setTimeout(refresh_charts,chart.refreshstep,chart);
}
$('[name="refresh"]').on('change', function(event, state) {
  var chart = eval($(this).parent().parent().parent().parent().find("canvas").attr('id'));
  chart.refreshstep = parseFloat(this.value);
  clearTimeout(chart.timeoutid);
  refresh_charts(chart);
});
$('[name="query"]').on('click', function(event, state) {
  var chart = eval($(this).parent().parent().parent().parent().find("canvas").attr('id'));
  //console.log(chart.query);
  inputs = $(this).parent().parent();
  chart.query = 'SELECT '+ inputs.find('#select').val()
  +' AS "y","time" AS "x" FROM '+ inputs.find('#from').val() +' WHERE '+ inputs.find('#where').val()
  +' GROUP BY '+inputs.find('#groupby').val()
  +' FILL(null)';
  clearTimeout(chart.timeoutid);
  refresh_charts(chart);
});
wsConnect();
```

# A.6. Codi de la pàgina base

## A.6.1. HTML

```
<!DOCTYPE html>

<html>

 <head>

   <!-- Configuració de bootstrap -->

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <!-- Defineix el color de la barra superior del navegador Chrome per mòbils -->

   <meta name="theme-color" content="#000000">

    <!-- Per mostrar el titol de la pagina a la pestanya superor -->

   <title>SmartHome</title>

    <!-- Importar totes les llibreries css necessaries -->

   <link rel="stylesheet" href="/css/bootstrap.min.css">

   <link href="css/bootstrap-switch.css" rel="stylesheet">

   <link rel="stylesheet" type="text/css" href="css/bootstrap-clockpicker.min.css">

   <link rel="stylesheet" href="/css/base.css">

 </head>


 <body onunload="ws.disconnect();">

   <nav class="navbar navbar-expand-lg navbar-dark sticky-top">

      <!-- Brand and toggle get grouped for better mobile display -->


       <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNavDropdown" aria-controls="navbarNavDropdown" aria-expanded="false" aria-
label="Toggle navigation">

         <span class="navbar-toggler-icon"></span>

       </button>
```

```
<div class="collapse navbar-collapse" id="navbarNavDropdown">
  <ul class="navbar-nav">
    <li id="home" class="nav-item">
      <a class="nav-link" href="home">Inici</a>
    </li>
    <li id="switches" class="nav-item">
      <a class="nav-link" href="switches">Interruptors</a>
    </li>
    <li id="sensors" class="nav-item">
      <a class="nav-link" href="sensors">Sensors</a>
    </li>
    <li id="cuina" class="nav-item">
      <a class="nav-link" href="cuina">Cuina</a>
    </li>
    <li id="cuina" class="nav-item">
      <a class="nav-link" target="_blank" href="http://192.168.1.20:6680/iris/#/">Musica<img
class="ml-1 icon" src="svg/external-link.svg"></a>
    </li>
    <li id="config" class="nav-item">
      <a class="nav-link" href="config">Configuració<img class="ml-1 icon" src="svg/external-
link.svg"></a>
    </li>
    <li id="ui" class="nav-item">
      <a class="nav-link" target="_blank" href="/ui">Altres<img class="ml-1 icon"
src="svg/external-link.svg"></a>
    </li>
    <li id="admin" class="nav-item">
      <a class="nav-link" target="_blank" href="/">Admin<img class="ml-1 icon" src="svg/external-
link.svg"></a>
    </li>
```

```
          </ul>
        </div>
        <a class="navbar-brand" href="home">
          <img alt="Brand" height="20px" src="img/logo_white.png">
          <button id="connected" type="button" class="btn btn-outline-success btn-sm mr-2"
style="display:None" disabled>Conectat</button>
          <button id="disconnected" type="button" class="btn btn-outline-danger btn-sm mr-2"
disabled>Desconectat</button>
        </a>
    </nav>

    <script src="js/jquery-3.2.1.slim.min.js"></script>
    <script src="js/popper.min.js"></script>
    <script src="/js/bootstrap.min.js"></script>
    {{ container }}

  </body>
</html>
```

### A.6.2. CSS

```
.navbar{
   background: rgba(0,0,0,1);
   transition: background .5s;
   margin-bottom: 15px;
}
.navbar:hover{
   background: rgba(0,0,0,1);
   transition: background .5s;
}
```

```
.icon {
  height: 10px;
  width: 10px;
  -webkit-filter: invert(100%);
  filter: invert(100%);
}
```

# B. ANNEX B. CODI DE LA PRIMERA VERSIÓ DEL SERVIDOR AMB DJANGO

## B.1. Main.py

```
import RPi.GPIO as GPIO
import time
from pymongo import MongoClient
import Adafruit_DHT as dht
import datetime

# SETUP
# ---------------------------------
# Database setup
client = MongoClient()
db = client.db
outputs = db.family_outputs
inputs = db.family_inputs
settings = db.family_settings

# Raspberry Setup
GPIO.setmode(GPIO.BCM)
for pin in range(2, 10):
    GPIO.setup(pin, GPIO.OUT)
    GPIO.output(pin, GPIO.HIGH)

# Initialize Variables
motion_sensor_timer = settings.find_one({"name": "motion_sensor_timer"})['value']
motion_sensor_pin = settings.find_one({"name": "motion_sensor_pin"})['value']
GPIO.setup(motion_sensor_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
refresh_time = 0.1
```

```
morning = datetime.time(8, 30)

dark = datetime.time(18, 00)

night = datetime.time(23, 30)

sleep = datetime.time(00, 00)


# FUNCTIONS

# Update de la variable desitjada (outputs.update) amb registre de temps.

# Activar o desectivar pin (GPIO.output).


# LOOP

# ---------------------------------

count = 0

while True:

    # Motion Sensor

    try:

        if settings.find_one({"name": "motion_sensor_toggle"})['value']:

            if GPIO.input(motion_sensor_pin):

                # Fer que el temps d'apagat de la llum incrementi en funcio de la cantitat de cops que passes
per dabant

                count = 0

                timestamp = datetime.datetime.now().time()

                if datetime.time(22, 30) < timestamp < datetime.time(23, 00):

                    outputs.update({'name': 'Left light'}, {"$set": {'state': 1}}, upsert=False)

                else:

                    outputs.update({'name': 'Right light'}, {"$set": {'state': 1}}, upsert=False)

            if count == motion_sensor_timer:

                outputs.update({'name': 'Right light'}, {"$set": {'state': 0}}, upsert=False)

                outputs.update({'name': 'Left light'}, {"$set": {'state': 0}}, upsert=False)

            if count < motion_sensor_timer:
```

```
            count += 1
    except:
        pass


    # Time Trigger
    timestamp = datetime.datetime.now().time()
    if datetime.time(7, 30) < timestamp < datetime.time(7, 31):
        settings.update({'name': 'motion_sensor_toggle'}, {"$set": {'value': 1}}, upsert=False)
        outputs.update({'name': 'Endolls'}, {"$set": {'state': 0}}, upsert=False)
        outputs.update({'name': 'Baixar'}, {"$set": {'state': 0}}, upsert=False)
        outputs.update({'name': 'Pujar'}, {"$set": {'state': 1}}, upsert=False)
    if datetime.time(17, 00) < timestamp < datetime.time(17, 01):
        outputs.update({'name': 'Pujar'}, {"$set": {'state': 0}}, upsert=False)
        outputs.update({'name': 'Baixar'}, {"$set": {'state': 1}}, upsert=False)
    if datetime.time(23, 00) < timestamp < datetime.time(23, 01):
        settings.update({'name': 'motion_sensor_toggle'}, {"$set": {'value': 0}}, upsert=False)
        outputs.update({'name': 'Endolls'}, {"$set": {'state': 1}}, upsert=False)
        time.sleep(30)
        outputs.update({'name': 'Right light'}, {"$set": {'state': 0}}, upsert=False)
        outputs.update({'name': 'Left light'}, {"$set": {'state': 0}}, upsert=False)


    # Working
    if outputs.find_one({"name": "Endolls"})['state']:
        settings.update({'name': 'motion_sensor_toggle'}, {"$set": {'value': 0}}, upsert=False)
    else:
        settings.update({'name': 'motion_sensor_toggle'}, {"$set": {'value': 1}}, upsert=False)
```

```
# Update Pin Stauts
    for output in outputs.find():
        if output['state'] == 0:
            GPIO.output(output['pin'], GPIO.HIGH)
        else:
            GPIO.output(output['pin'], GPIO.LOW)
    # Update Sensor Values
    h, t = dht.read_retry(dht.DHT22, inputs.find_one({'name': 'temperature'})['pin'])
    inputs.update({'name': 'temperature'}, {"$set": {'metadata': t}}, upsert=False)
    inputs.update({'name': 'humidity'}, {"$set": {'metadata': h}}, upsert=False)
    time.sleep(refresh_time)
```

## B.2. Views.py

```
from django.shortcuts import render
from family.models import home_elements, outputs, inputs, settings
from django.utils import timezone


def index(request):
    context = {"home_elements": home_elements.objects.all()}
    return render(request, 'family/index.html', context)


def new_element(request):
    context = {}
    return render(request, 'family/new_element.html', context)
def add_element(request):
    home_elements.objects.get_or_create(name=request.POST['name'],
                                defaults={'date': timezone.now(),
                                        'img': "/static/family/img/%s.png" % request.POST['name']}
                                )
    return index(request)


def output(request):
    context = {"outputs": outputs.objects.all()}
    return render(request, 'family/outputs.html', context)


def new_output(request):
    context = {}
    return render(request, 'family/new_output.html', context)
```

```
def add_output(request):
    outputs.objects.get_or_create(name=request.POST['name'], pin=request.POST['pin'])
    return output(request)

def change_output(request):
    if request.POST['name'] == 'up' and outputs.objects.get(name='up').state == 0:
        outputs.objects.filter(name='down').update(state=0)
    if request.POST['name'] == 'down' and outputs.objects.get(name='down').state == 0:
        outputs.objects.filter(name='up').update(state=0)
    try:
        val = 1 - outputs.objects.get(name=request.POST['name']).state
    except:
        pass
    outputs.objects.filter(name=request.POST['name']).update(state=val)
    return output(request)

def temperature(request):
    context = {"inputs": inputs.objects.all()}
    return render(request, 'family/temperature.html', context)

def delete(request):
    home_elements.objects.filter(name=request.POST['name']).delete()
    inputs.objects.filter(name=request.POST['name']).delete()
    outputs.objects.filter(name=request.POST['name']).delete()
    settings.objects.filter(name=request.POST['name']).delete()
    return index(request)
```

```
def setting(request):
    context = {"settings": settings.objects.all()}
    return render(request, 'family/settings.html', context)


def add_settings(request):
    settings.objects.get_or_create(name=request.POST['name'], value=request.POST['value'])
    return setting(request)
```

## B.3. Urls.py

```
from django.conf.urls import url

from . import views


urlpatterns = [
    url(r'^$', views.index, name='index'),
    url(r'^delete', views.delete, name='delete'),
    url(r'^outputs', views.output, name='outputs'),
    url(r'^new_element', views.new_element, name='new_element'),
    url(r'^add_element', views.add_element, name='add_element'),
    url(r'^temperature', views.temperature, name='temperature'),
    url(r'^settings', views.setting, name='settings'),
    url(r'^new_output', views.new_output, name='new_output'),
    url(r'^add_output', views.add_output, name='add_output'),
    url(r'^add_settings', views.add_settings, name='add_settings'),
    url(r'^change_output', views.change_output, name='change_output'),
]
```

## B.4. Models.py

```
from django.db import models
from djangotoolbox.fields import ListField


class home_elements(models.Model):
    date = models.DateTimeField()
    name = models.TextField()
    metadata = models.IntegerField(blank=True, null=True, default=None)
    img = models.TextField()
    size = models.TextField(default='col-sm-2 col-xs-4')


class outputs(models.Model):
    name = models.TextField()
    pin = models.IntegerField()
    state = models.IntegerField(blank=True, null=True, default=0)
    time_log = ListField(blank=True, null=True, default=[])


class inputs(models.Model):
    name = models.TextField()
    pin = models.IntegerField()
    metadata = models.IntegerField(blank=True, null=True, default=0)
    log = ListField(blank=True, null=True, default=[])


class settings(models.Model):
    name = models.TextField()
    value = models.IntegerField()
```

# C. ANNEX C. IMATGES DE L'EVOLUCIÓ DE LA INTERFÍCIE

# C.1. Primera versió

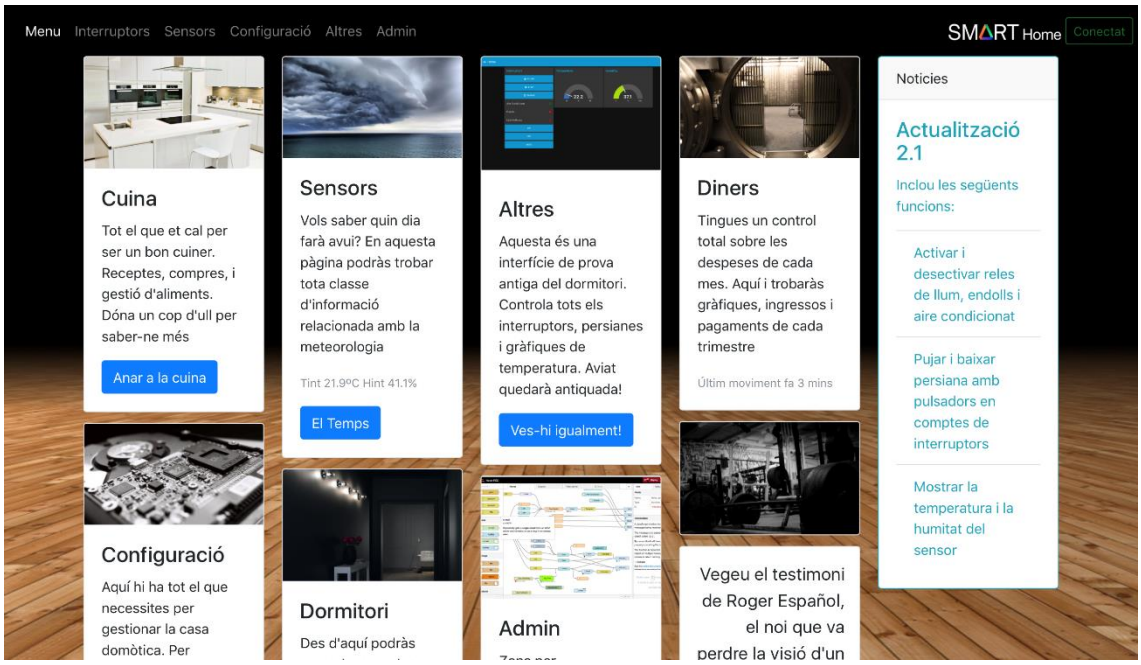



*Figura 1. Pàgina d'inici de la primera versió d'interfície. Font pròpia*

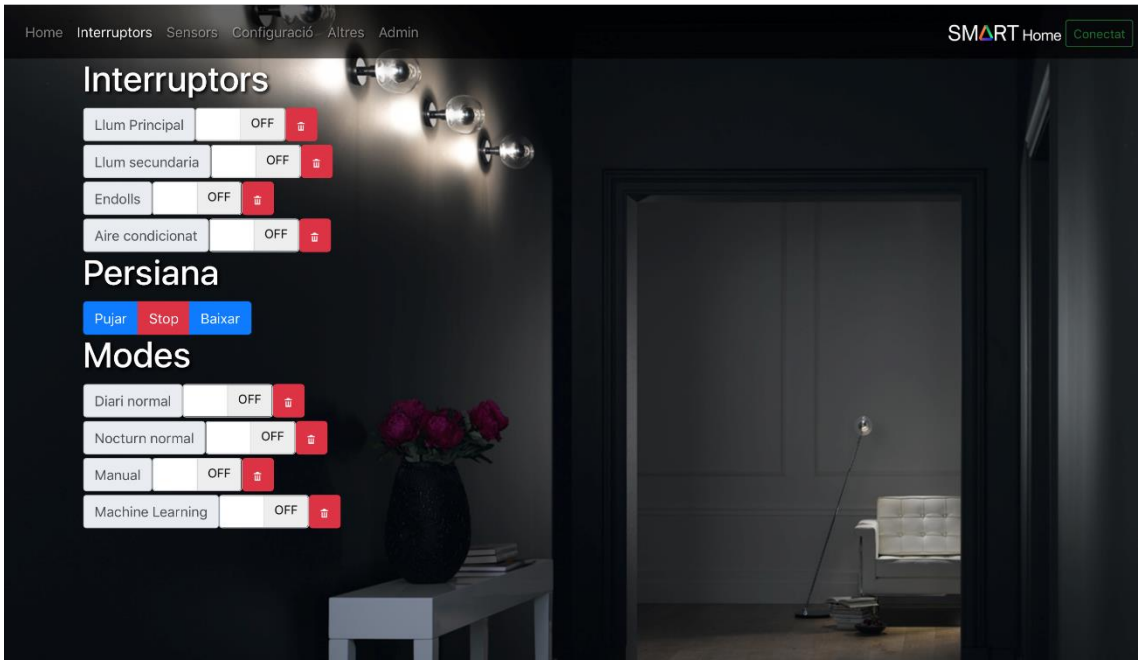



*Figura 2. Pàgina d'interruptors de la primera versió d'interfície. Font pròpia*

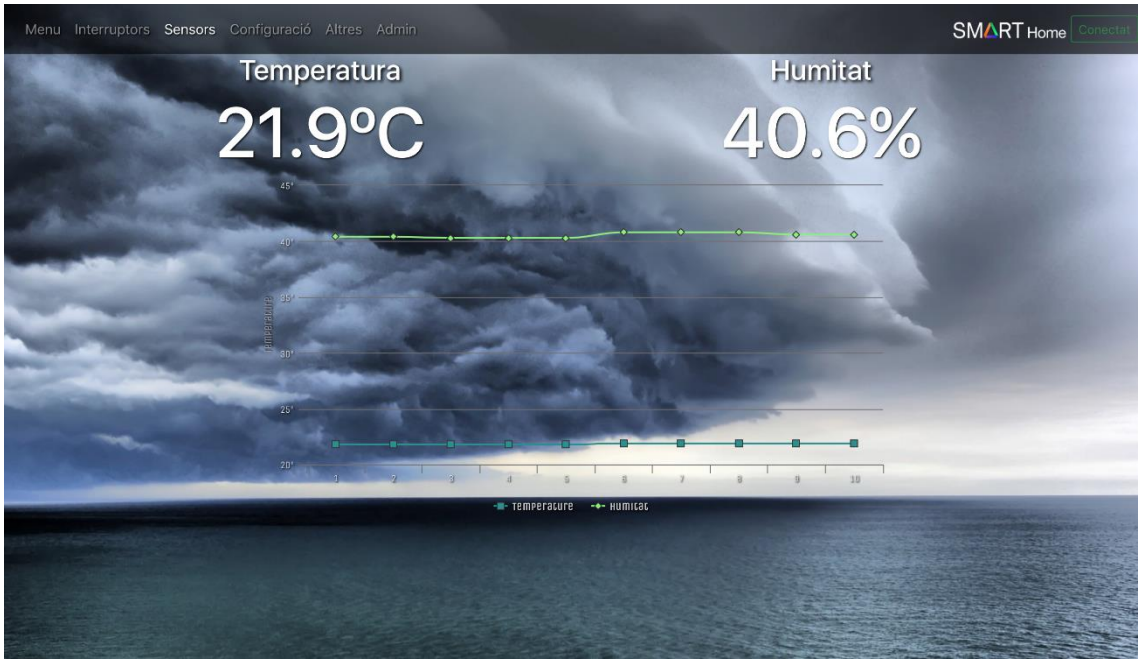



*Figura 4. Pàgina de sensors de la primera versió d'interfície. Font pròpia*

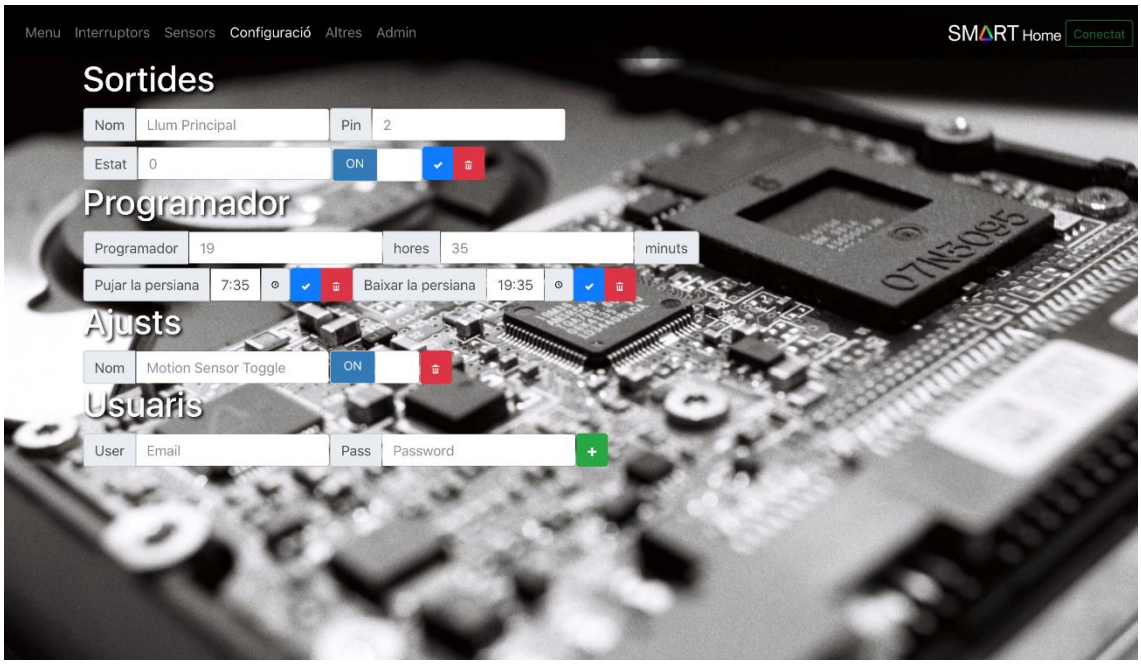



*Figura 3. Pàgina de configuració de la primera versió d'interfície. Font pròpia*

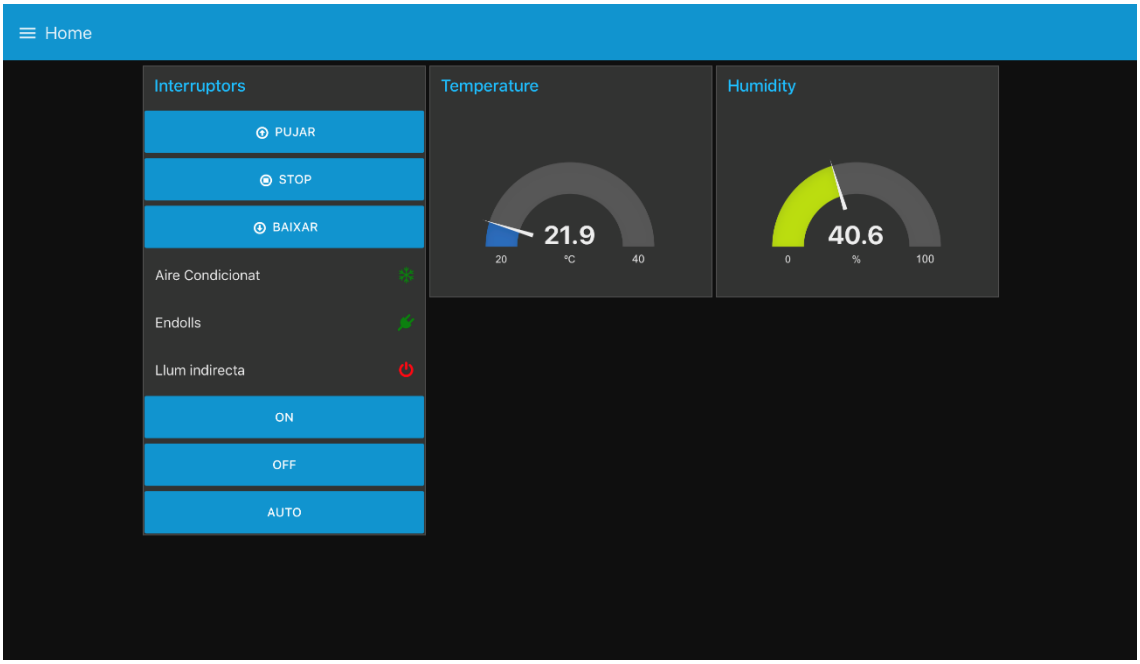



*Figura 5. Dashboard de la primera versió d'interfície. Font pròpia*

## C.2. Interfície final

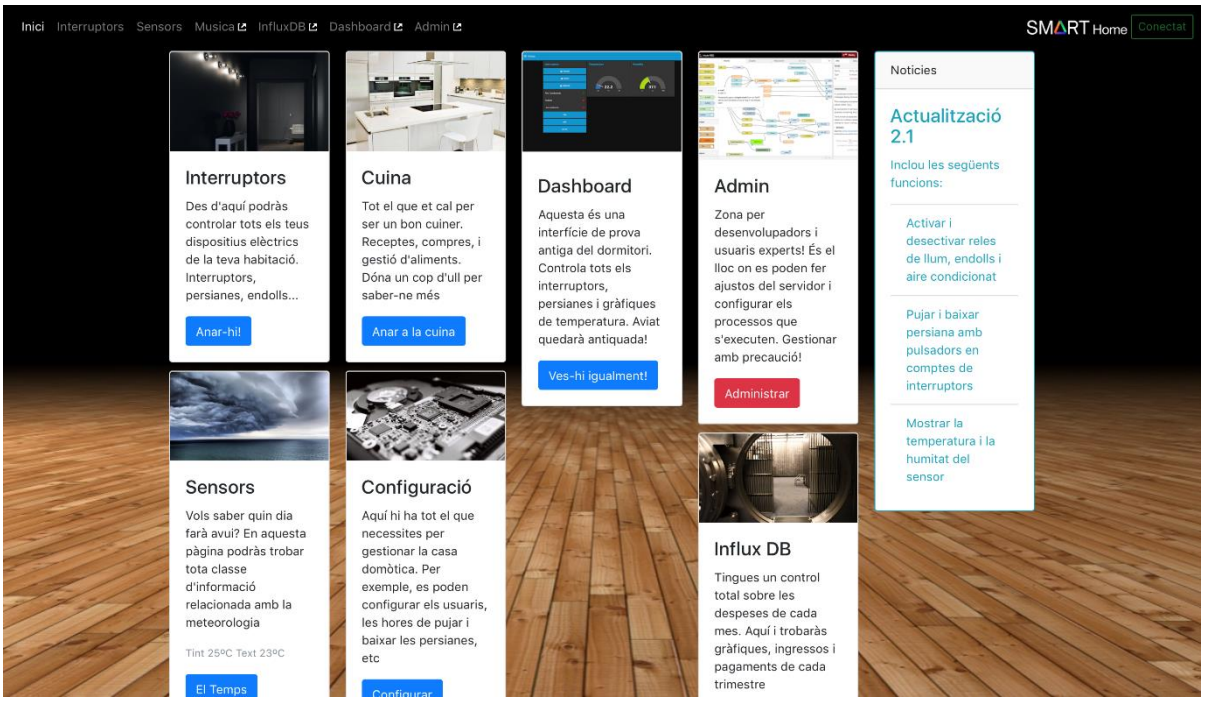



*Figura 7. Pàgina d'inici de la interfície final. Font pròpia*

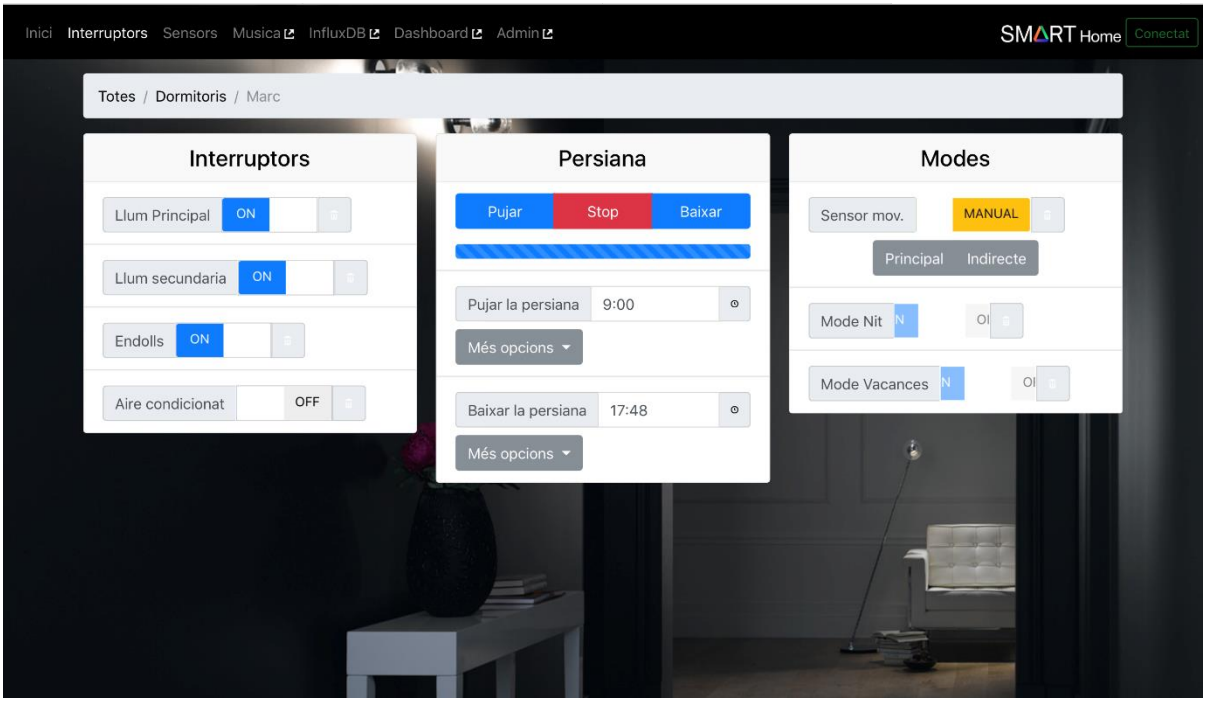



*Figura 6. Pàgina d'interruptors de la interfície final. Font pròpia*

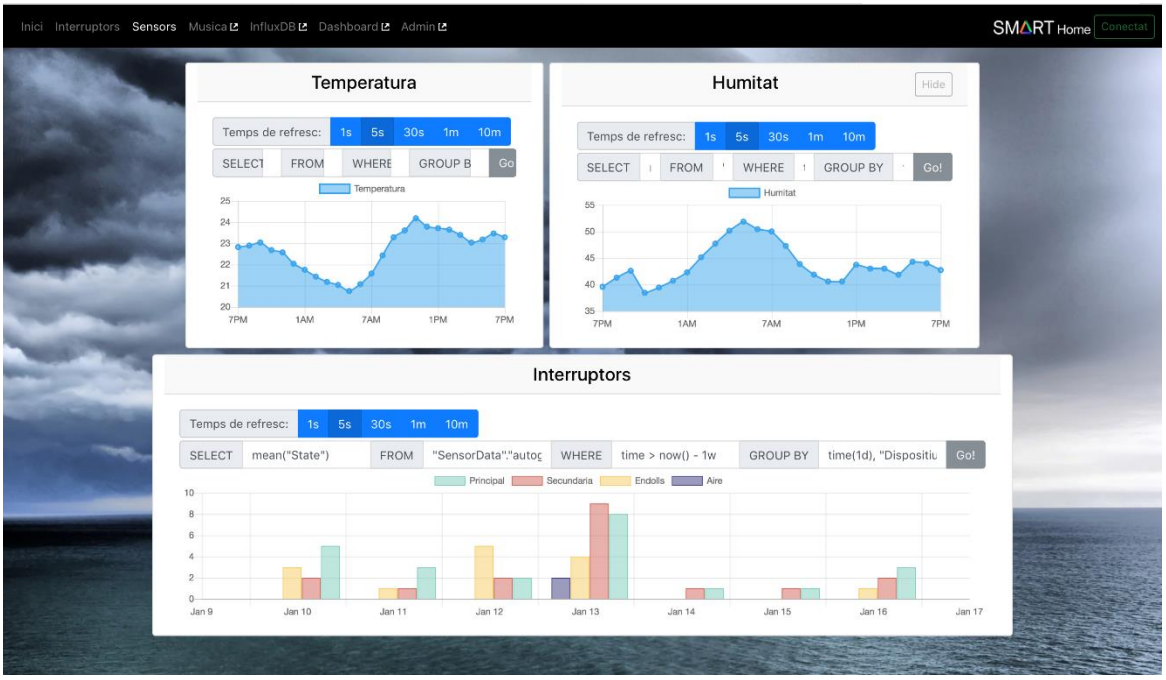



*Figura 8. Pàgina de sensors de la interfície final. Font pròpia*

## C.3. Versions alternatives de la interfície d'interruptors

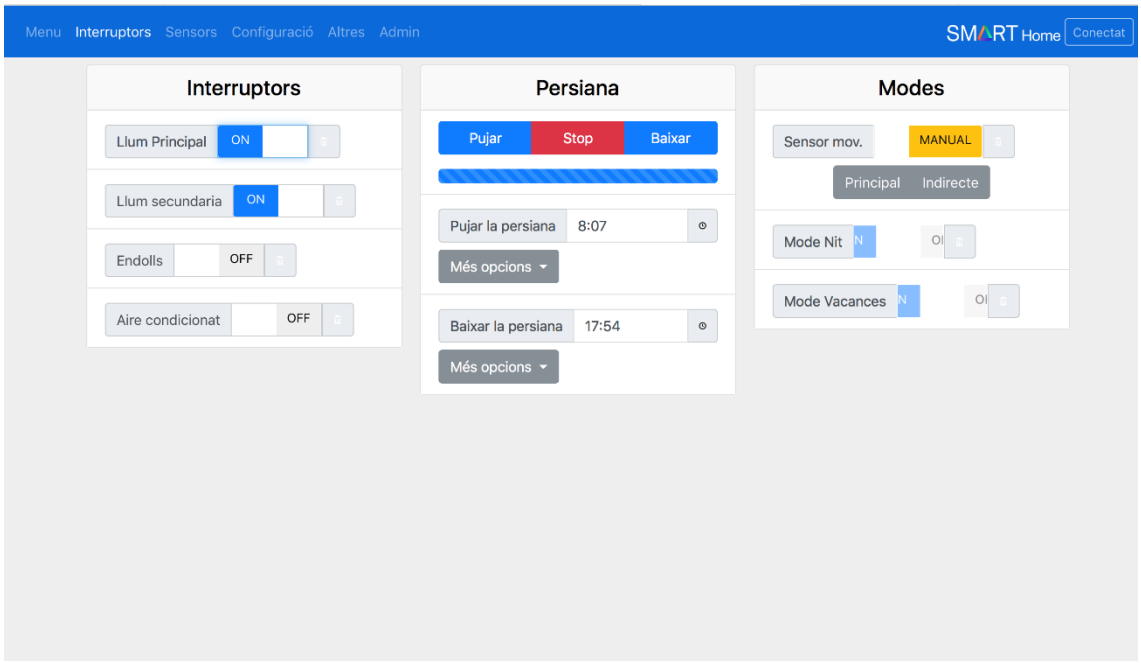



*Figura 9. Versió alternativa 1. Font pròpia*

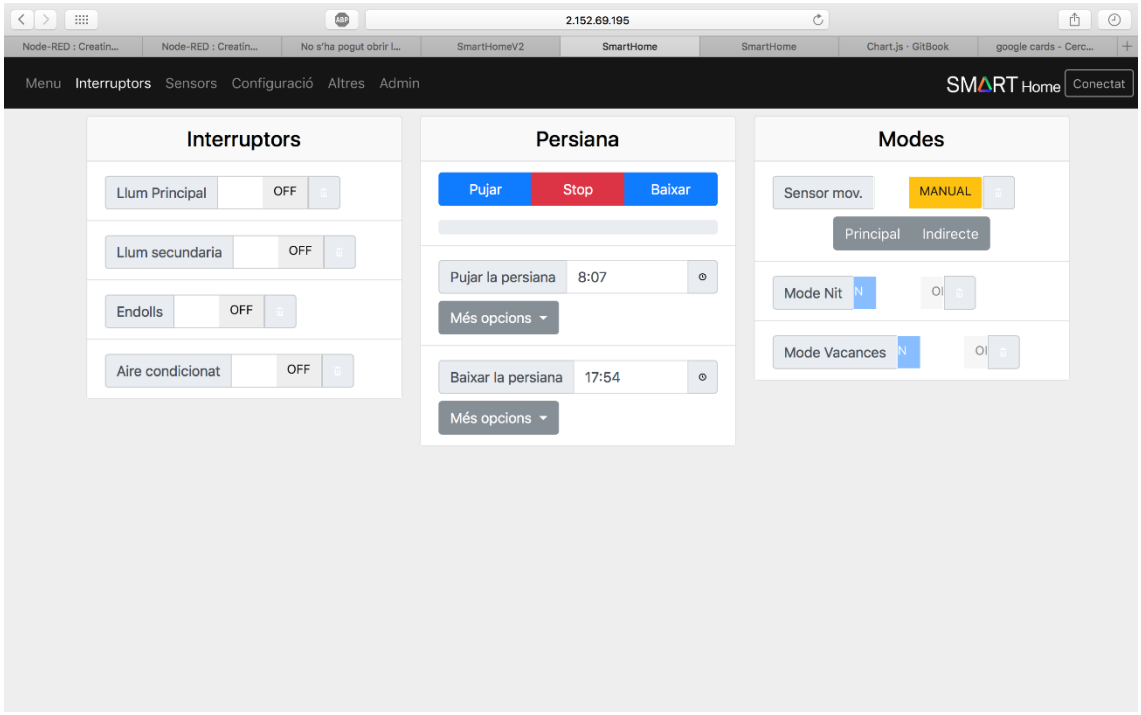



*Figura 10. Versió alternativa 2. Font pròpia*

# D. ANNEX D. FLOWS DE NODE-RED SENSE UTILITZAR LA COMUNICACIÓ MQTT

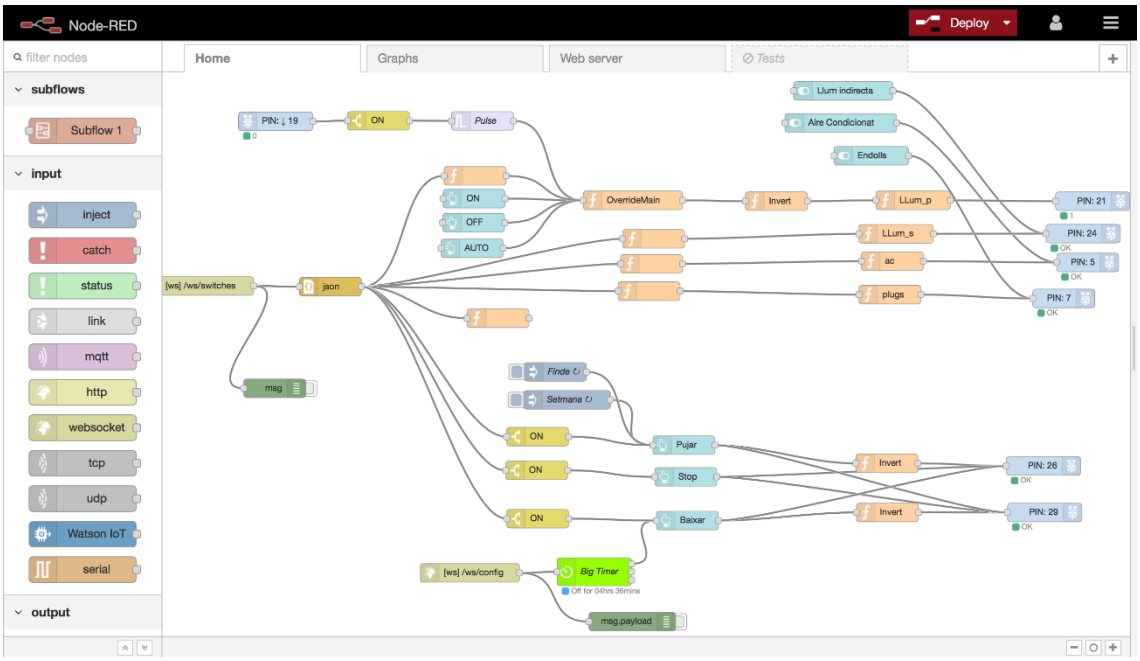



*Figura 12. Flow de la interfície d'usuari. Font pròpia*

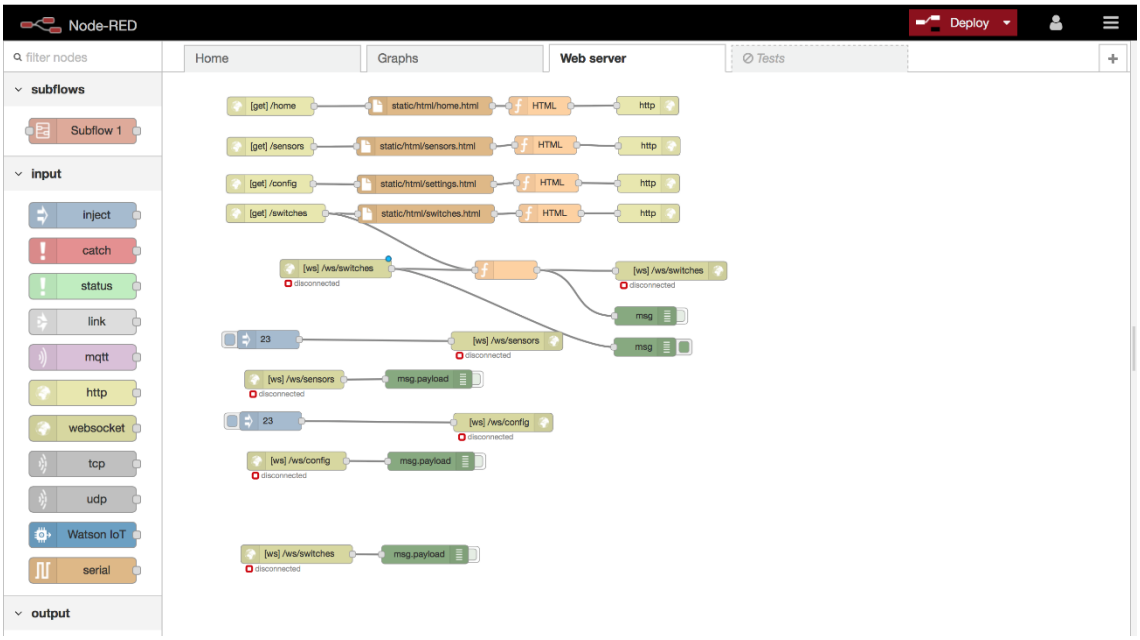



*Figura 11. Flow de la interfície web. Font pròpia*

# E. ANNEX D. OPTIMITZACIÓ DE L'AMPLA DE BANDA

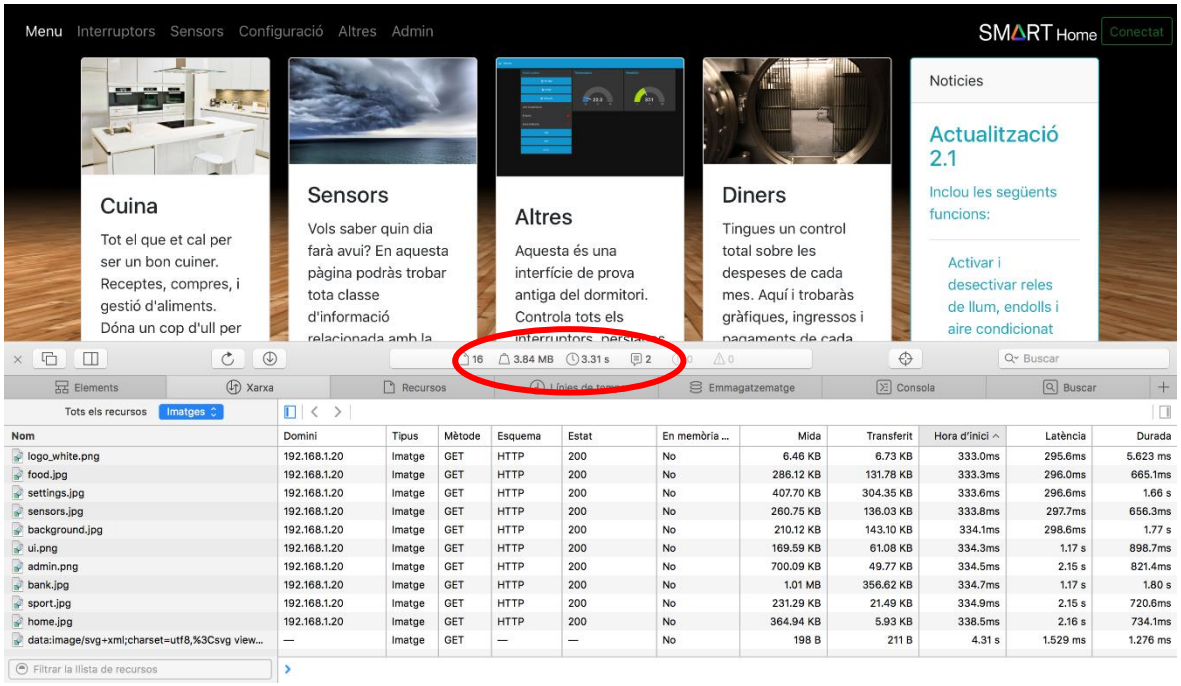



*Figura 133. Resultats sense optimitzar: 3.84 MB, 3.31s. Font pròpia*

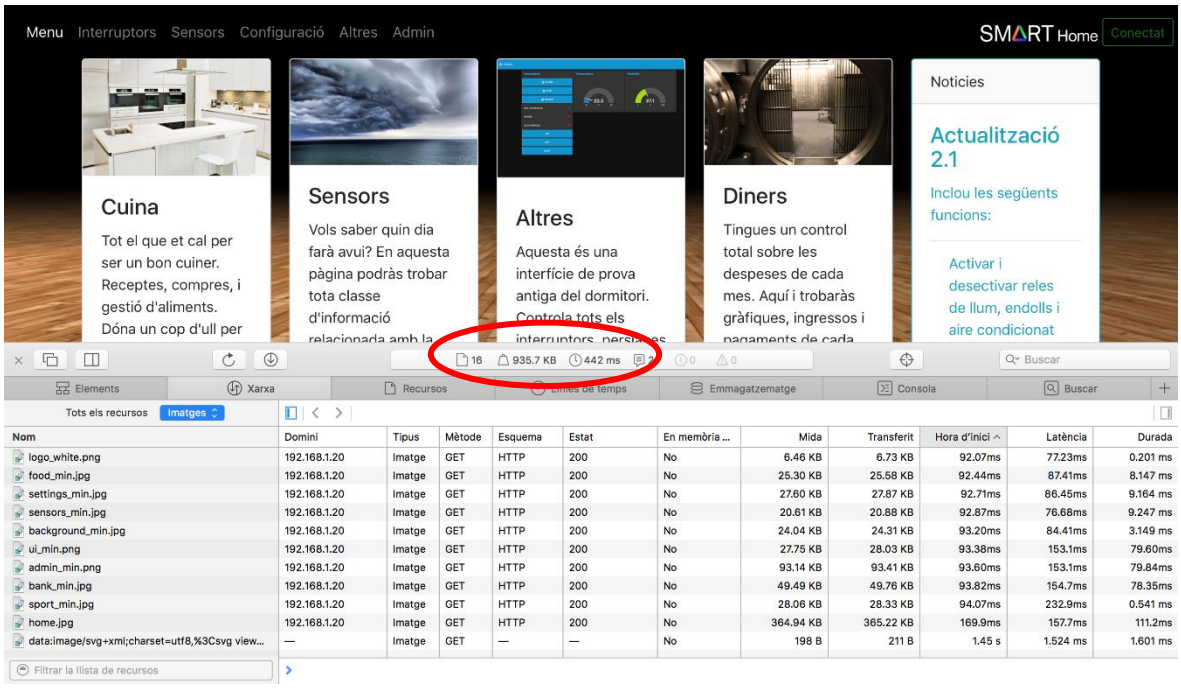



*Figura 14. Resultats amb optimització: 935.7 KB, 442 ms. Font pròpia*