**DOMOTICA**

Comunicació entre dispositius mitjançant interfície web.

Gerard Flequé Bernadó

2n ASIX - 2017

INDEX

[AGRAÏMENTS 4](#_Toc483946710)

[INTRODUCCIÓ 5](#_Toc483946711)

[NOTES SOBRE CODI FONT I COMANDES. 5](#_Toc483946712)

[PLANTEJAMENT DEL PROJECTE 7](#_Toc483946713)

[CONCEPTE DE DOMÒTICA 8](#_Toc483946714)

[HARDWARE 10](#_Toc483946715)

[LLISTA DE REQUISITS INICIALS 11](#_Toc483946716)

[CARACTERISTIQUES IMPRESCINDIBLES 11](#_Toc483946717)

[CARACTERISTIQUES OPCIONALS 11](#_Toc483946718)

[SOFTWARE 12](#_Toc483946719)

[LLISTA DE REQUISITS INICIALS 13](#_Toc483946720)

[TIPUS DE COMUNICACIÓ ENTRE DISPOSITIUS 14](#_Toc483946721)

[Wi-Fi 14](#_Toc483946722)

[Bluetooth 14](#_Toc483946723)

[ZigBee 14](#_Toc483946724)

[SOFTWARE I TECNOLOGIES EMPRADES 16](#_Toc483946725)

[PHP 16](#_Toc483946726)

[BOOTSTRAP 16](#_Toc483946727)

[ARDUINO IDE 16](#_Toc483946728)

[HTML 16](#_Toc483946729)

[JAVASCRIPT 17](#_Toc483946730)

[JQUERY 17](#_Toc483946731)

[RATCHET 17](#_Toc483946732)

[JSON 17](#_Toc483946733)

[NODEJS 18](#_Toc483946734)

[EXPLICACIÓ SOFTWARE 19](#_Toc483946735)

[Script d’instal·lació 19](#_Toc483946736)

[Codi principal WebSockets 23](#_Toc483946737)

[JavaScript plana web 26](#_Toc483946738)

[Notes importants 28](#_Toc483946739)

[CONFIGURACIÓ 30](#_Toc483946740)

[SERVIDOR 31](#_Toc483946741)

[/opt/websocket/src/webservergerard/Chat.bin 31](#_Toc483946742)

[/var/www/html/updater.php 32](#_Toc483946743)

[/var/www/html/index.html 33](#_Toc483946744)

[CLIENTS 36](#_Toc483946745)

[CONCLUSIONS 40](#_Toc483946746)

[BIBLIOGRAFIA 41](#_Toc483946747)

# AGRAÏMENTS

En primer lloc, m’agradaria agrair a tots els professors de l’INS Sa Palomera, per la seva ajuda en la realització d’aquest projecte. En especial en Josep Catà i en Jose Moreno per motivar-me a fer aquest projecte.

També voldria agrair el suport de totes les persones que m’han ajudat, directament o indirectament a completar aquest projecte.

Per últim, vull agrair a totes les persones que comparteixen els seus treballs i projectes de DIY, ja que el meu treball té molta sinergia amb qualsevol d’aquests projectes.

# INTRODUCCIÓ

El meu treball es basa en donar la base per poder automatitzar i gestionar qualsevol casa, de forma molt econòmica, adaptable i fàcil d’amagar.

La comunicació es basa en una comunicació on tant el client com el servidor actuen com a servei i client, fent una comunicació unidireccional amb capacitat bidireccional on el client processa les ordres del servidor i envia les dades que facin falta com sensors o estat dels actuadors, i el servidor processa les dades i les guarda pel seu posterior ús i envia les accions corresponents al client.

L’usuari interactua amb aquest sistema mitjançant una interfície web, utilitzant php per enviar accions i Javascript i JQuery per modificar la web segons les accions.

### NOTES SOBRE CODI FONT I COMANDES.

Aquest treball de recerca, ateses les seves característiques, està ple de petits paràgrafs amb

aquest format que

$ podeu apreciar en

# aquest exemple.

Això indica que formen part d’un arxiu de configuració, de codi o bé que es tracta d’una seqüència de comandes. Les seqüències de comandes sempre van introduïdes pels símbols $ o #. El primer ($) indica que la comanda és executada com a usuari sense privilegis i el segon (#), que l’ordre és executada com a superusuari (root).

La paraula DIY (*do it yourself*), és una forma d’auto-producció sense esperar la voluntat d’altres per realitzar tasques personals. Aquest projecte es basa sobretot en aquest mètode, ja que té tantes opcions com imaginació tinguis.

La paraula o frases que es veuen [com aquesta frase](#a), són enllaços que corresponen a una web relacionada amb la paraula o frase.

## PLANTEJAMENT DEL PROJECTE

La idea de la qual sorgeix aquest treball és principalment en la curiositat de cóm podria funcionar un sistema de comunicació per gestionar diverses funcions en llocs diferents, i els elevats costos de les solucions actuals de domòtica. Per aquest motiu m’he plantejat fer un sistema escalable, de baix cost i sense necessitat de dependre de marques específiques.

## CONCEPTE DE DOMÒTICA

Per respondre la pregunta em valdré de diverses definicions del mateix concepte i, ja que no totes elles coincidiran exactament, intentaré redactar-ne una de satisfactòria incorporant idees de la resta.

El 1988 l'enciclopèdia Larousse defineix el terme domòtica com el següent: "el concepte de vivenda que integra tots els automatismes en matèria de seguretat, gestió de l’energia, comunicacions, etc."

Com veurem més endavant, aquesta explicació és poc específica, però reflexa perfectament el pensament de la seva època. Ara donem un cop d'ull a la descripció (més actual i concreta) que fa l'Institut Cerdà, una fundació privada que es dedica a assessorar diverses empreses per al disseny i construcció d'edificis intel·ligents.

Sota el punt de vista de l'usuari es denomina vivenda domòtica aquella que permet una major qualitat de vida a través de la tecnologia oferint una reducció del treball domèstic, un augment del benestar i de la seguretat dels seus habitants i una racionalització dels diferents consums.

Si analitzem la definició, en primer lloc se situa com a màxim valor de l'automatització la millora de qualitat de vida, un terme vague i ambigu que no deixa de ser desitjat per tots. Aquesta qualitat de vida, segons l'Institut Cerdà, s'aconsegueix aplicant el progrés tecnològic a l'habitatge per tal de suavitzar les tasques que ens veiem obligats a realitzar-hi, augmentar el benestar ( i per benestar entenem confort) i la seguretat dels habitants. Pel que fa al consum energètic, és cert que la nostra societat ha anat prenent consciència paulatinament del problema de l'esgotament de les fonts d'energia convencionals i de la seva importància en la millora de la qualitat de vida. L'arquitectura reflexa aquesta inquietud i es planteja la optimització dels recursos com un dels objectius d'una vivenda intel·ligent.

Sota el punt de vista tecnològic s'anomena vivenda domòtica aquella en la que existeixen agrupacions automatitzades d'equips normalment associats per funcions, que disposen de la capacitat de comunicar-se interactivament entre elles utilitzant un sistema tècnic.

En una línia molt més tècnica, s'entén per instal·lació domòtica un conjunt d'aparells electrònics que es comuniquen entre sí i realitzen algunes funcions. Aquesta explicació no té en compte que a més de comunicar-se entre sí, aquests dispositius han de ser capaços de comunicar-se amb les persones a qui serveixen, i no tan sols això sinó que ho han de fer de manera fàcil per tal que als usuaris no els suposi un esforç aprendre a controlar-los. Aquesta comunicació és el que s'anomena interfície home-màquina i engloba tant la capacitat d'un aparell de ser programat i configurat com la d'informar del que fa i com ho fa a cada moment.

Continuem amb la definició d' Ingrid Kirschning, llicenciada en enginyeria de sistemes computacionals, a la seva tesis sobre edificis intel·ligents, assessorada per la Dra. Ofelia Cervantes Villagómez.

Un edifici intel·ligent es defineix com una estructura que ofereix als seus usuaris i administradors un conjunt coherent d'eines i facilitats. Està dissenyat per poder cobrir tots els possibles avenços tecnològics, sempre tenint en compte les necessitats reals dels usuaris i administradors de l'edifici. La finalitat d'un edifici intel·ligent és la de proporcionar un ambient de confort i seguretat, per maximitzar la productivitat i la creativitat així com fer que la gent se senti a gust en el seu lloc de treball. A més aquest tipus d'edificis ha de proporcionar medis per a un manteniment eficient i oportú, tot això minimitzant els costos.

Amb aquesta explicació queden clares les finalitats d'un edifici intel·ligent i el fet que, evidentment, en una empresa el control integrat afavoreix tant els treballadors com la producció i, per tant, l'empresari. Tot i això, no es parla dels medis mitjançant els quals s'aconsegueixen les millores, és a dir que es podria completar força a un nivell més tècnic.

# HARDWARE

Aquesta secció esta dedicada a tot el tema relacionat amb la tria de del hardware adequat per a la realització del projecte. Aquí es parlarà de les necessitats i de la llista d’objectius inicials.

## LLISTA DE REQUISITS INICIALS

Després d’haver introduït el concepte del projecte i haver-ne presentat els seus aspectes més generals, cal definir les característiques de les màquines que volem construir. En la llista següent dividirem el que són característiques essencials del que serien afegits interessants, però en cap cas imprescindibles.

### CARACTERISTIQUES IMPRESCINDIBLES

Pel cas del servidor, gairebé qualsevol ordinador pot servir, però jo recomano la [Raspberry Pi 3](https://www.raspberrypi.org/products/raspberry-pi-3-model-b/), un ordinador complet, compacte i sorprenentment potent i barat, amb un consum increïblement baix. Això proporciona un consum molt inferior al d’un ordinador de casa comú, ja que es molt recomanable que estigui encès sempre.

Pels clients, jo utilitzaré el model [LinkNode D1](http://linksprite.com/wiki/index.php5?title=LinkNode_D1) i [NodeMCU ESP12E Dev Kit v2](https://www.amazon.com/s/ref=nb_sb_noss_1?url=search-alias%3Daps&field-keywords=nodemcu+v2) un chip programable basat en ESP8266, un chip Wifi de baix cost. Però es pot fer amb qualsevol dispositiu amb Wifi , però el chip que utilitzaré jo, ja te Wifi integrat i t’estalvia molts problemes. També es pot fer amb un altre Raspberry o similars!

### CARACTERISTIQUES OPCIONALS

Pels clients, vaig intentar aconseguir la [Raspberry Pi Zero W](https://www.raspberrypi.org/products/pi-zero-w/), un ordinador complet per 10€ amb Wifi i Bluetooth, però no ho vaig aconseguir.

# SOFTWARE

En aquesta secció del treball es parlarà dels objectius que es pretenen assolir pel que fa al software i es parlarà del software i les tecnologies de software utilitzades en aquest treball de recerca.

## LLISTA DE REQUISITS INICIALS

El servidor hauria de disposar de les següents característiques:

* [Raspbian](https://www.raspberrypi.org/downloads/raspbian/) (S.O. de Raspberry)
* PHP5
* Apache2
* [Ratchet](http://socketo.me/) (WebSockets)
* [Bootstrap](http://getbootstrap.com/) (Per disseny de la plana web)
* [Nodejs](https://nodejs.org/es/)

En el cas dels clients, només necessites un ordinador amb el programari Arduino IDE i les llibreries necessàries per el codi.IJIJWDIGIUG

* #include <Arduino.h>
* #include <[ESP8266WiFi.h](https://github.com/esp8266/Arduino)>

// ESP8266Wifi.h ens permet moltes funcions del chip ESP8266, entre elles, client NTP, punt d’accés o peticions HTTPS.

* #include <DNSServer.h>
* #include <[Adafruit\_Sensor.h](https://github.com/adafruit/Adafruit_Sensor)>

//Adafruit\_Sensor.h necessària per sensors DHT

* #include <ESP8266WebServer.h>

//ESP8266WebServer.h ens permet crear un servidor web.

* #include <[WiFiManager.h](https://github.com/tzapu/WiFiManager)>

//WifiManager.h ens permet configurar la connexió Wifi.

Com es pot veure, el servidor és la principal font de processament

## TIPUS DE COMUNICACIÓ ENTRE DISPOSITIUS

IoT, Internet of Things, és el futur en què tot tipus de dispositius i sensors es comuniquen entre si -- i amb els ordinadors distants i altres sistemes -- per operar d'una manera transparent i transformar el nostre món.

Tota la comunicació ha de succeir en algun tipus de xarxa. Alguns dispositius podrien ser afegits en una xarxa existent, però altres hauran de comunicar-se sense fils.

No hi ha escassetat de protocols sense fil, Wi-Fi, Bluetooth i ZigBee són totes les solucions. Si bé cada un es comunica a través d'ones de ràdio, ho fan de diferents maneres.

Aquí un ràpid resum de cada tipus de solució sense fils i els seus avantatges i desavantatges:

### Wi-Fi

Aquesta és una tecnologia sense fils d'àrea local que utilitza una freqüència de 2,4 GHz o 5 GHz. Aquesta tecnologia és ideal per a l'enviament de grans quantitats de dades sense fils entre dispositius. Però també requereix una gran quantitat d'energia per funcionar.

### Bluetooth

Introduït per Ericsson tornada a la dècada de 1990, la tecnologia Bluetooth permet que les xarxes de dades personals. Transmet dades a través de la banda de freqüència entre 2,4 i 2.485 GHz. Opera a través de distàncies més curtes que WiFi i requereix menys energia per funcionar. Pot aparellar dispositius com telèfons, smartwatches, auriculars, altaveus, ordinadors i més junts. Fins i tot amb l'abast limitat, les primeres implementacions de Bluetooth eren una gran sagnia de durada de la bateria.

### ZigBee

Aquest estàndard és molt adequat per a xarxes de malla. L'estàndard ZigBee permet que els dispositius de baixa potència enviar dades al llarg d'una xarxa, amb cada dispositiu capaç de retransmetre les dades cap al seu destí previst. Això li permet configurar una xarxa molt eficaç –no t’has de preocupar per les distàncies com amb els dispositius Bluetooth. Mentre el seu dispositiu ZigBee pugui xatejar amb un altre dispositiu a la xarxa, ja està llest. L’inconvenient principal és que no esta del tot estandarditzat, causant problemes entre dispositius.

## SOFTWARE I TECNOLOGIES EMPRADES

### PHP

PHP (Hypertext Preprocessor) és un llenguatge de codi obert molt popular especialment adequat per al desenvolupament web i que pot ser incrustat en HTML.

Per el nostre cas, l’utilitzarem per a processar totes les peticions dels clients, ja siguin els Arduino o navegadors web. S’encarrega de la comunicació entre els dispositius.

### BOOTSTRAP

Bootstrap és el framework més popular per HTML, CSS i JS per al desenvolupament de planes responsives, aptes tant per un mòbil com per una gran pantalla.

Bootstrap l’utilitzarem per donar un format agradable per la nostra plana web.

### ARDUINO IDE

L'Arduino IDE (Integrated Development Environment), en poques paraules, és l'entorn en el qual es pot escriure codi d’Arduino, compilar i carregar a l’Arduino o a la targeta compatible amb Arduino.

Utilitzarem aquest software per crear i compilar el codi que utilitzi el nostre arduino.

### HTML

HTML (Hypertext Markup Language) és el conjunt de símbols de marques o codis inserits en un arxiu destinat a la visualització en una plana web. El marcatge indica al navegador web com mostrar paraules i les imatges d'una pàgina web per a l'usuari. Cada codi de marcat individu es coneix com un element (però moltes persones també es refereixen a ella com una etiqueta).

L’estructura de la nostra web està en HTML.

### JAVASCRIPT

Javascript és un llenguatge de programació utilitzat per fer pàgines web interactives. S'executa en l'ordinador del seu visitant i no requereix descàrregues constants del seu lloc web. JavaScript s'utilitza sovint per crear enquestes i qüestionaris.

Faig servir Javascript per tindre interaccions amb l’usuari, com per exemple canviar els botons de color.

### JQUERY

jQuery és una biblioteca ràpida, petita i rica en funcions de JavaScript. Permet fer coses com recórrer i manipular HTML, maneig d'esdeveniments, animació i Ajax molt més simple amb un API fàcil d'utilitzar que funciona a través d'una multitud de navegadors.

L’utilitzaré per poder escriure Javascript millor.

### RATCHET

Ratchet és una llibreria PHP que proporciona als desenvolupadors eines per crear en temps real, aplicacions bidireccionals entre clients i servidors a través d'WebSockets.

Aplicació que ens permet tindre una comunicació bidireccional amb els clients web, enviant-li les dades actualitzades a tots els clients. (WebSocket)

### JSON

JSON (acrònim de JavaScript Object Notation) és un estàndard obert basat en text dissenyat per a intercanvi de dades llegible per humans. Deriva del llenguatge script JavaScript, per a representar estructures de dades simples i llistes associatives, anomenades objectes.

És a dir, és una forma d’enviar dades de forma simple i fàcil d’entendre.

### NODEJS

Node.js és un entorn de programació dissenyat per escriure aplicacions d'Internet escalables, els programes estan escrits en JavaScript.

L’utilitzarem com a intermediari dels missatges dels clients Arduino i el servidor.

## EXPLICACIÓ SOFTWARE

Per aquesta secció, trossejaré part del codi i comentar per sobre la seva funcionalitat.

Tot el codi aquí mencionat, el podeu trobar als annexos o a [GitHub](https://github.com/Enjoyed/RatchetInstallation).

### Script d’instal·lació

El podeu trobar complet a l’annex < Script instal·lació servidor (install.sh) >.

Com podeu veure, aquest primer paràgraf no te gaire sentit, però té sentit, ja que instal·lem (mes tard) una petita llibreria per fer la instal·lació una miqueta gràfica, aquesta llibreria s’instal·la a /etc/init.d/funcions.sh disponible a [GitHub](https://github.com/marcusatbang/efunctions.git) , la qual primer comprovem que existeix, sinó l’instal·larà més endavant. Per inicialitzar aquestes llibreries noves instal·lades, fa falta reiniciar l’script, per això li donem permisos d’execució. També creem el fitxer de log per comprovar si n’hi ha hagut errors durant la instal·lació.

#!/bin/bash

if [ -f "/etc/init.d/functions.sh" ]; then

. /etc/init.d/functions.sh

else

ScriptLoc=$(readlink -f "$0")

chmod ugo+x "$ScriptLoc"

fi

touch /var/log/gerardscript.log

El següent codi és simplement un comprovant, de que tenim connexió a internet i s’està executant amb permisos de root. Per comprovar-ho, comprovem l’UID de l’usuari, si és 0, significa que es root, en cas contrari, és qualsevol altre usuari. Per comprovar la connexió, fem un wget de google.com, amb l’opció spider per no descarregar res, en cas que no hagi pogut comunicar-se amb google.com, sortirà de l’script.

printf 'CHECKS... '

if [ "$(id -u)" != "0" ]; then

echo "This script must be run as root" 1>&2

exit 1

fi

wget -q --tries=10 --timeout=20 --spider http://google.com

if [[ $? -eq 0 ]]; then

true

else

echo "Need internet access to run this scrint" 1>&2

exit 1

fi

printf '[\e[1;32mDONE\e[0m]\n'

El següent tros de codi és la instal·lació de les llibreries anteriorment mencionades, un cop instal·lades torna a executar l’script per a poder utilitzar les noves llibreries instal·lades. També faig un clear per no deixar lleig el codi repetit (check)

if [ ! -f "/etc/init.d/functions.sh" ]; then

git clone https://github.com/marcusatbang/efunctions.git /opt/efunctions &>> /var/log/gerardscript.log

cd /opt/efunctions

./install.sh &>> /var/log/gerardscript.log

exec "$ScriptLoc"

fi

clear

Imprimim per pantalla una petit text per comunicar que s’està iniciant, i esperem 3 segons per continuar, en cas de que l’usuari hagi executat l’script per error, que pugui cancel·lar-ho sense afectar en gran mesura al sistema.

Com podeu veure, tot l’output es redirigeix a /var/log/gerardscript.log, en cas d’error s’hauria de mirar aquest log per veure on ha petat.

Per el codi futur, amagaré les comandes de <ebegin> i <eend>, ja que només serveixen per la part gràfica.

printf "Initializing Gerard Script (domotica project) by Gerard Fleque"

printf "\nLog -> /var/log/gerardscript.log\n\n"

sleep 3

echo "Executant apt-get update..." >> /var/log/gerardscript.log

ebegin "Updating your system..."

apt-get update &>> /var/log/gerardscript.log

eend $?

Ara instal·larà tots els paquets necessaris per al correcte funcionament del nostre sistema.

apt-get -qq install -y apache2 apache2-utils &>> /var/log/gerardscript.log

apt-get -qq install -y php5 php5-json php5-dev php5-curl &>> /var/log/gerardscript.log

apt-get -qq install -y nodejs npm &>> /var/log/gerardscript.log

Ara creem la nostra estructura de carpetes, per utilitats del servidor, com per exemple el servei de WebSockets i els actualitzadors.

mkdir -p /opt/websocket

mkdir -p /opt/websocket/src/websocketgerard

mkdir /opt/websocket/bin

mkdir -p /var/www/html/assets/css

mkdir /var/www/html/assets/js

mkdir /var/www/html/assets/fonts

mkdir /tmp/RatchetInstallation

Ara ens descarreguem tots els fitxers a través de git, els guardem en una carpeta temporal i més endavant els copiarem a les seves rutes corresponents.

git clone https://github.com/Enjoyed/RatchetInstallation /tmp/RatchetInstallation &>> /var/log/gerardscript.log

Necessitem instal·lar Composer, un gestor de dependències per PHP. Per instal·lar-ho, hem d’anar a la carpeta on ens interessa tenir-lo i seguir uns passos de la seva pàgina oficial.

cd /opt/websocket

php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');" &>> /var/log/gerardscript.log

php -r "if (hash\_file('SHA384', 'composer-setup.php') === '669656bab3166a7aff8a7506b8cb2d1c292f042046c5a994c43155c0be6190fa0355160742ab2e1c88d40d5be660b410') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP\_EOL;" &>> /var/log/gerardscript.log

php composer-setup.php &>> /var/log/gerardscript.log

php -r "unlink('composer-setup.php');" &>> /var/log/gerardscript.log

Ara instal·larem Ratchet, el servei de WebSockets, es tan simple com utilitzar Composer.

php composer.phar require cboden/ratchet &>> /var/log/gerardscript.log

A continuació començarem a moure tots els fitxers descarregats a les seves ubicacions corresponents.

yes | cp -rf /tmp/RatchetInstallation/Chat.php /opt/websocket/src/websocketgerard/Chat.php

yes | cp -rf /tmp/RatchetInstallation/composer.json /opt/websocket/composer.json

yes | cp -rf /tmp/RatchetInstallation/server.php /opt/websocket/bin/server.php

yes | cp -rf /tmp/RatchetInstallation/updater.js /opt/websocket/updater.js

yes | cp -rf /tmp/RatchetInstallation/refresh.js /opt/websocket/refresh.js

yes | cp -rf /tmp/RatchetInstallation/check.sh /opt/websocket/check.sh

yes | cp -rf /tmp/RatchetInstallation/index.html /var/www/html/index.html

yes | cp -rf /tmp/RatchetInstallation/custom.css /var/www/html/assets/css/custom.css

yes | cp -rf /tmp/RatchetInstallation/updater.php /var/www/html/updater.php

yes | cp -rf /tmp/RatchetInstallation/index.html /var/www/html/index.html

També aprofito i per comoditat canvio la IP al fitxer de la plana web principal.

NEWIP=$(ip route get 1 | awk '{print $NF;exit}')

sed -i.bak "s/10.19.250.1/$NEWIP/g" /var/www/html/index.html

Per a poder utilitzar Ratcher amb la classe que he fet, fa falta que actualitzem Composer amb la nova configuració

php composer.phar update &>> /var/log/gerardscript.log

Degut a la natura del propi Arduino, no pot executar JavaScript, per això necessitem una llibreria per a poder utilitzar el nostre servidor com a intermediari.

npm install --silent -g node-gyp &>> /var/log/gerardscript.log

npm install --silent websocket &>> /var/log/gerardscript.log

Descarreguem els components necessaris per la plana web, Bootstrap i jQuery al seu lloc corresponent.

cd /var/www/html/assets/css

wget https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css &>> /var/log/gerardscript.log

cd /var/www/html/assets/js

wget https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js &>> /var/log/gerardscript.log

cd /var/www/html/assets/js

wget https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js &>> /var/log/gerardscript.log

Per pura comoditat, faig que s’executi el servei quan inicia l’Apache2.

sed -i.bak "/ start)/!{p;d;};n;n;a php -q /opt/websocket/bin/server.php &" /etc/init.d/apache2

Creem una tasca programada per a actualitzar automàticament les dades dels nostres dispositius, per evitar dades incorrectes. En demana dades cada 30 segons.

crontab -l > mycron

echo "\* \* \* \* \* bash /opt/websocket/check.sh" >> mycron

echo "\* \* \* \* \* (sleep 30; bash /opt/websocket/check.sh)" >> mycron

crontab mycron

rm mycron

Finalment, modifiquem els permisos per evitar diversos problemes

chmod 775 -R /opt/websocket

chmod 775 -R /var/www/html/

### Codi principal WebSockets

El podeu trobar sencer a l’annex < Codi WebSocket servidor (Chat.php) >

Aquest codi és el principal procés del servei de WebSockets, processa tota l’entrada i sortida. Si un client web fa una acció, com per exemple un botó, després de processar-lo amb el propi Javascript de la pàgina, ho envia a aquest codi per processar i distribuir. S’encarrega d’enviar les noves dades als Arduino i de repartir les dades actualitzades a tots els clients, principalment.

El primer que veiem es una llista associativa, que conté tots els sensors/actuadors de tots els dispositius, aquí és on es guarden totes les dades dels sensors/actuadors. La seva sintaxis és:

< id\_dispositiu > \_ < sensor/actuador > => < valor >

$GLOBALS["values"] = array(

"d1\_temperatura" => 0, "d1\_led1" => 0,

"d2\_temperatura" => 0, "d2\_led1" => 0,

"d3\_temperatura" => 0, "d3\_led1" => 0,

"d4\_temperatura" => 0, "d4\_led1" => 0)

Just després, tenim un altre llista associativa, però aquesta guarda les adreces que fan referencia al dispositiu.

GLOBALS["dispositius"] = array(

"d1" => "10.19.250.50",

"d2" => "10.19.250.20",

"d3" => "10.19.250.30")

Ja que treballem molt amb formats JSON, vaig veure convenient tindre una funció per poder comprovar si té format JSON.

function isJson($string) {

json\_decode($string);

return (json\_last\_error() == JSON\_ERROR\_NONE); }

A continuació tenim la classe Chat, on succeeix la màgia. Hi han quatre funcions, equivalents a <Nova Connexió>, <Missatge rebut>, <Connexió tancada> i <Error>.

Quan un usuari nou es connecta, se li envia tota la llista de sensors/actuadors per a que s’actualitzi amb les noves dades. No té més misteri.

public function onOpen(ConnectionInterface $conn) {

// We add it to the list.

$this->clients->attach($conn);

// Encode our array with ALL sensors/actuators

$send = json\_encode($GLOBALS["values"]);

// And send it to the new client connected.

foreach ($this->clients as $client) {

if ($conn == $client) {

$client->send($send);

}

}

// A simple echo to debug

echo "New connection! ({$conn->resourceId})\n";

}

La part de comunicació ja es més complexa i consta de 3 parts principals, si és un missatge d’un client web, un missatge per renovar les dades preguntant als arduino o un missatge d’actualització, és a dir, un missatge que ve d’un arduino.

Quan el missatge arriba des d’un client web, preparem l’URL per a fer una petició GET, el nostre Arduino ja està preparat per processar aquestes peticions. En cas de que la petició no hagi funcionat correctament, s’envia el valor original (teòricament el correcte) al client que ha enviat la petició. Si ha arribat bé, s’afegeix el nou valor a la llista oficial, i s’actualitza a tots els clients.

foreach($array as $sensor => $valor) {

$sendValor = explode("\_",$sensor);

$url\_final = $GLOBALS["dispositius"][$sendValor[0]] . "/?" . $sensor . "=" . $valor;

$curl = curl\_init();

curl\_reset($curl);

curl\_setopt($curl, CURLOPT\_URL, $url\_final);

curl\_setopt($curl, CURLOPT\_TIMEOUT, 2);

curl\_setopt($curl, CURLOPT\_CONNECTTIMEOUT, 2);

$data = curl\_exec($curl);

if ($data === FALSE) {

echo "Curl failed: " . curl\_error($curl) . "\n";

$correct = "{\"" . $sensor . "\":" . $GLOBALS["values"][$sensor] . "}";

foreach ($this->clients as $client) {

if ($conn == $client) {

$client->send($correct);

}

}

}

else {

$GLOBALS["values"][$sensor] = $valor;

$send = "{\"" . $sensor . "\":" . $valor . "}";

foreach ($this->clients as $client) {

if ($conn !== $client) {

$client->send($send);

}

}

}

curl\_close($curl);

}

En cas de que el missatge sigui de renovació de dades, prepararem la URL cap a totes les adreces dels nostres dispositius (la llista associativa de dispositius), però processarem la resposta de l’Arduino, que conté les dades reals, i ho actualitzem a la nostre llista oficial, ja que aquestes dades son els valors reals. Un cop actualitzades s’envien les dades actualitzades per evitar confusions.

elseif(isset($array["refresh"]))

{

foreach($GLOBALS["dispositius"] as $dispositiu => $ip){

$curl = curl\_init();

curl\_reset($curl);

$url\_final = $ip . "/json";

curl\_setopt($curl, CURLOPT\_URL, $url\_final);

curl\_setopt($curl, CURLOPT\_TIMEOUT, 2);

curl\_setopt($curl, CURLOPT\_CONNECTTIMEOUT, 2);

$data = curl\_exec($curl);

if ($data === FALSE) {

echo "Could not get ::$ip:: Curl failed: " . curl\_error($curl) . "\n";

}

else {

$json = json\_decode($data,true);

foreach($json as $sensor => $valor){

if($sensor !== "machine\_id"){

$GLOBALS["dispositius"][$sensor] = $valor;

}

}

echo "Updated $sensor\n";

}

}

$send = json\_encode($GLOBALS["values"]);

foreach ($this->clients as $client)

{

$client->send($send);

}

}

Quan són missatges d’actualització, com que són de confiança, directament s’afegeixen valors actualitzats a la taula oficial i un cop afegits, s’actualitzen a tots els clients.

else{

foreach($array as $sensor => $valor){

if($sensor !== "incoming"){

$GLOBALS["values"][$sensor] = $valor;

}

}

$send = json\_encode($GLOBALS["values"]);

foreach ($this->clients as $client) {

if ($conn !== $client) {

$client->send($send);

}

}

}

### JavaScript plana web

El podeu trobar sencer a la part final de l’annex < Plana web principal (plantilla)(index.html) >

Aquest codi processa tot el que te que veure amb la interacció entre l’usuari i la interfície web, des de canviar els botons de color fins a enviar dades al nostre servei WebSockets.

El primer codi que es presència és la connexió al servei WebSockets, continuat de una comprovació. Si no aconsegueix connectar-se al servei en 2 segons, dona una alerta informant a l’usuari de tota acció que faci serà invàlida, que contacti amb un administrador.

var conn = new WebSocket("ws://10.19.250.1:8080");

setTimeout(function () {

if (conn.readyState != 1) {

alert("Cannot connect to WebSocket Service. Call the system admin.\nNOTE: Nothing you do in this webpage will work.");

}

}, 2000);

Ara tenim la funció de <OnMessage>, on ocurreix gran part de la màgia.

En aquesta funció és on arriban totes les dades del servei de WebSockets, és a dir, és on es processen les noves dades. Però no es res mes que fer un bucle de les noves dades i actualitzar en funció de si és un botó o un sensor.

conn.onmessage = function(e) {

console.log("Updating...");

Data = JSON.parse(e.data);

for (var prop in Data) {

var element = document.getElementById(prop);

var type = $(element).attr("tipo");

if(type == "onoff") {

switch(Data[prop]) {

case 0:

element.innerHTML = "OFF";

element.classList.remove("btn-success");

element.classList.add("btn-danger");

break;

case 1:

element.innerHTML = "ON";

element.classList.remove("btn-danger");

element.classList.add("btn-success");

break;

case "0":

element.innerHTML = "OFF";

element.classList.remove("btn-success");

element.classList.add("btn-danger");

break;

case "1":

element.innerHTML = "ON";

element.classList.remove("btn-danger");

element.classList.add("btn-success");

break;

}

}

else if(type == "sensor")

{

element.innerHTML = Data[prop];

}

}

}

Per evitar problemes, faig que els botons siguin clicables quan la pàgina hagi estat ben carregada.

window.onload = function()

{

$("a").each( function(index, element){

$(this).bind("click",changeClass);

});

}

També he afegit una funció per enviar les dades en format json; una jsonizadora!

function sendMessage(text, valor) {

var json = "{\"" + text + "\":" + valor + "}";

conn.send(json);

console.log(json);

}

### Notes importants

Com pots haver notat, hi han molts més fitxers, però aquests tres són els principals.

Els principals fitxers on hauries de configurar almenys un parell de coses, les quals les explico en l’apartat de [configuració](#_CONFIGURACIÓ):

#### /var/www/html/updater.php

En aquest fitxer hauries de modificar la llista <$accepted> per acceptar les adreces dels teus Arduino.

#### /opt/websocket/src/webservergerard/Chat.bin

En aquest fitxer has de modificar les dos llistes de <$GLOBALS>, a ‘values’ hauries de modificar-lo amb els teus sensors/actuadors. A ‘dispositius’ has de modificar amb l’adreça dels teus dispositius i el seu codi d’identificació.

És important que els dispositius estiguin disponibles (l’adreça de xarxa), del contrari, es podria saturar el sistema, perdent molt temps valuós esperant una resposta que no arribarà, això cada 30 segons!

#### /var/www/html/index.html

Aquí hem de modificar, afegir o eliminar els botons i sensors de cada dispositiu i en cas de modificar l’adreça de quan es va instal·lar, també l’adreça del servidor de WebSockets.

# CONFIGURACIÓ

En aquest apartat, explicaré com configurar tot el necessari per a configurar aquest sistema a la vostra manera.

## SERVIDOR

Començarem per tindre una bona organització dels nostres dispositius, dóna-li nom!, pel meu exemple, tindre tres dispositius, r1, que controla vuit relés, és a dir vuit botons, anomenats r1 fins r8, després tindrem a d1, amb un sensor de temperatura i finalment d2 amb un sensor de llum.

### /opt/websocket/src/webservergerard/Chat.bin

El primer arxiu que haurem de tocar és el codi principal del nostre servidor, /opt/websocket/src/webservergerard/Chat.bin, modificant les dues llistes, en aquest cas us mostro com ho canvio als meus requisits.

ORIGINAL

$GLOBALS["values"] = array(

"d1\_temperatura" => 20,

"d1\_led1" => 0,

"d2\_temperatura" => 20,

"d2\_led1" => 0,

"d3\_temperatura" => 20,

"d3\_led1" => 0,

"d4\_temperatura" => 20,

"d4\_led1" => 0,

"d5\_temperatura" => 20,

"d5\_led1" => 0,

"d6\_temperatura" => 20,

"d6\_led1" => 0,);

$GLOBALS["dispositius"] = array(

"d1" => "10.19.250.50",

"d2" => "10.19.250.20",

"d3" => "10.19.250.30",

"d4" => "10.19.250.40",

"d5" => "10.19.250.10"

);

MODIFICAT

$GLOBALS["values"] = array(

/\* Dispositiu d1 \*/

"d1\_luminositat" => 0,

/\* Dispositiu d2 \*/

"d1\_temperatura" => 0,

/\* Dispositiu R1 \*/

"r1\_r1" => 0,

"r1\_r2" => 0,

"r1\_r3" => 0,

"r1\_r4" => 0,

"r1\_r5" => 0,

"r1\_r6" => 0,

"r1\_r7" => 0,

"r1\_r8" => 0);

$GLOBALS["dispositius"] = array(

"r1" => "10.19.250.100", /\* IP r1 \*/

"d1" => "10.19.250.110", /\* IP d1 \*/

"d2" => "10.19.250.120", /\* IP d2 \*/

);

Com es pot apreciar, ho he modificat per tal que encaixi amb els dispositius i sensors que tinc. El nom dels sensors han de ser els mateixos que hagis configurat a l’Arduino.

### /var/www/html/updater.php

El següent arxiu que modificarem, és /var/www/html/updater.php, on modificarem la llista <$accepted>.

ORIGINAL

$accepted = array(

"::1",

"127.0.0.1",

"192.168.1.102",

"10.19.250.20",

"10.19.250.30",

"10.19.250.40",

"10.19.250.50");

MODIFICAT

$accepted = array(

"::1",

"127.0.0.1",

"10.19.250.10",

"10.19.250.110",

"10.19.250.120",

"10.19.250.130");

En el meu cas, tinc el següent:

* R1 té l’adreça 10.19.250.100
* D1 té l’adreça 10.19.250.110
* D2 té l’adreça 10.19.250.120
* D3 té l’adreça 10.19.250.130

Les adreces <127.0.0.1> i <::1> són adreces locals (la mateixa màquina), es poden treure, son opcionals.

### /var/www/html/index.html

El següent fitxer és el més extens i customitzable, on intentaré explicar com afegir o eliminar blocs, com un botó o un sensor.

<div class="col-md-2">

<h3>

Dispositiu 1

</h3>

<div class="row">

<div class="col-md-8">

<p>

<span>Temperatura:<span>

</p>

</div>

<div class="col-md-4">

<p id="d1\_temperatura" tipo="sensor">

0

</p>

</div>

</div>

<div class="row">

<div class="col-md-7">

<p>

LED1

</p>

</div>

<div class="col-md-5">

<a class="btn btn-danger" id="d1\_led1" tipo="onoff">

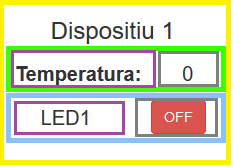
OFF

</a>

</div>

</div>

</div>

Com us podeu fixar amb els colors, és una estructura molt simple, si vols afegir un botó, només has de copiar el blau entre el blau i el groc, si vols un altre verd, només has de copiar-lo entre el blau i el groc. Pots posar tants com vulguis o necessitis.

#### COPIANT BOTONS

A l’hora de copiar un botó, has de tindre en compte, que has de modificar un parell de coses.

<div class="row">

<div class="col-md-7">

<p>

LED1

</p>

</div>

<div class="col-md-5" align="center">

<a class="btn btn-danger" id="d1\_led1" tipo="onoff">

OFF

</a>

</div>

</div>

Només hauries de modificar 2 coses, a part d’haver afegit el sensor a /opt/websocket/src/webservergerard/Chat.bin.

LED1: El nom del nou dispositiu, aquí es molt opcional, pots ficar-li tant led blau com <ultraled 3000>

d1\_led1: Aquí hauràs de vigilar amb majúscules i minúscules! Has de posar el nom del sensor, amb la sintaxis: <nom\_dispositiu>\_<sensor>.

#### COPIANT SENSORS

Pels sensors és bàsicament el mateix.

<div class="row">

<div class="col-md-8">

<p>

<span>Temperatura:<span>

</p>

</div>

<div class="col-md-4">

<p id="d1\_temperatura" tipo="sensor">

0

</p>

</div>

</div>

Els codis de color son els mateixos que els dels botons.

#### DESHABILITANT BOTONS

Per desactivar botons, en cas de que no vulguis eliminar tots els dispositius, és tan senzill com afegir la paraula <disabled>.

<div class="col-md-5" align="center">

<a class="btn btn-danger" id="d3\_led1" tipo="onoff" disabled>

OFF

</a>

</div>

#### ELIMINANT DISPOSITIUS

Per eliminar dispositius, només has de eliminar el <<div> que el conté, per exemple, si vols esborrar el dispositiu 8, només has d’eliminar tota la part verd.

<div class="col-md-2">

<h3>

Dispositiu 7

</h3>

</div>

<div class="col-md-2">

<h3>

Dispositiu 8

</h3>

</div>

#### MODIFICANT TAMANY DEL DISPOSITIU

Per fer més gran la columna d’un dispositiu, és tan simple com modificar un número. Tingues en compte, que l’amplada màxima són 12. Si tens dos dispositius, i vols que els dos ocupin mitja plana, hauries de modificar el número en color verd, per a que la suma de tots dos siguin 6. Pots fer el primer dispositiu més gran (posar 4) i el segon més petit (posar 2) per exemple.

<div class="col-md-2">

<h3>

Dispositiu 7

</h3>

</div>

<div class="col-md-2">

<h3>

Dispositiu 8

</h3>

</div>

## CLIENTS

En aquesta part, explicaré el codi principal dels Arduino. Començant per les llibreries, necessàries per realitzar diferents tasques, com llegir algun sensor específic o realitzar alguna tasca.

Els genèrics que utilitzaré jo, ens permeten tindre Wifi, un servei web, un client http i altres funcions.

#include <Arduino.h>

#include <ESP8266WiFi.h> //https://github.com/esp8266/Arduino

#include <DNSServer.h>

#include <Adafruit\_Sensor.h> //https://github.com/adafruit/Adafruit\_Sensor

#include <ESP8266WebServer.h>

#include <WiFiManager.h> //https://github.com/tzapu/WiFiManager

#include <ESP8266HTTPClient.h>

A continuació declarem les variables que utilitzarem.

ESP8266WebServer webserver(80);

WiFiClient client;

int timer = 0;

String last\_value = "";

String new\_value = "";

String \_id = "RELE1";

int r1 = 1;

int r2 = 1;

etc...

#### SETUP

Dintre del bloc de Setup, inicialitzarem tot el que necessitem.

void setup() {

// El codi aquí dins!

}

Primer, inicialitzem els pins que utilitzarem.

pinMode(pr1, OUTPUT);

pinMode(pr2, OUTPUT);

pinMode(pr3, OUTPUT);

pinMode(pr4, OUTPUT);

pinMode(pr5, OUTPUT);

pinMode(pr6, OUTPUT);

pinMode(pr7, OUTPUT);

pinMode(pr8, OUTPUT);

digitalWrite(pr1, HIGH);

digitalWrite(pr2, HIGH);

digitalWrite(pr3, HIGH);

digitalWrite(pr4, HIGH);

digitalWrite(pr5, HIGH);

digitalWrite(pr6, HIGH);

digitalWrite(pr7, HIGH);

digitalWrite(pr8, HIGH);

Després inicialitzem el Wifi i el Serial.

Serial.begin(115200);

WiFiManager wifiManager;

//wifiManager.resetSettings();

wifiManager.setAPStaticIPConfig(IPAddress(10,19,250,100), IPAddress(10,19,0,1), IPAddress(255,255,0,0));

wifiManager.setSTAStaticIPConfig(IPAddress(10,19,250,100),IPAddress(10,19,0,1),IPAddress(255,255,0,0));

wifiManager.autoConnect("Access RELE1");

I a continuació el servidor web.

webserver.on("/", [](){

webserver.send(200, "text/html", "");

});

webserver.on("/json", [](){

webserver.send(200, "text/html", Buildjson());

});

webserver.begin();

Serial.println("EVERYTHING DONE! Working...");

#### LOOP

Dintre del bloc de loop, escriurem el codi que s’executarà en bucle, és a dir, constantment.

void loop() {

// El codi aquí dins!

}

La primera part que veiem, és on es processa les peticions HTTP, per on li enviem instruccions.

El blau és el nom del dispositiu i el nom del sensor/actuador, la seva sintaxis és: <nom\_dispositiu>\_<nom\_sensor>.

El verd és la variable corresponent al sensor

El groc és la variable relacionada amb el pin.

webserver.handleClient();

if(webserver.args() != 0)

{

for (int i = 0; i < webserver.args(); i++) {

nom = webserver.argName(i);

valor = webserver.arg(i);

new\_value = nom + ":" + valor;

if(last\_value != new\_value){

if(nom == "r1\_r1"){

r1 = valor.toInt();

if(r1){

digitalWrite(pr1,HIGH);

}

else{

digitalWrite(pr1,LOW);

}

Serial.println("Modificat r1:" + valor);

}

Per enviar valors dels sensors al nostre servidor, utilitzarem el client http.

El groc ens permet modificar cada quan s’envia les dades del sensor.

if(timer > 50)

{

Serial.println("Starting send update");

HTTPClient http;

llum = analogRead(pResistor);

string = "http://10.19.250.1/updater.php?d1\_luminositat=" + String(llum) + "";

Serial.println(string);

http.begin(string);

int httpCode = http.GET();

if(httpCode > 0) {

if(httpCode == HTTP\_CODE\_OK) {

Serial.println("OK");

}

}

else

{

Serial.println("Could not connect to the updater");

}

http.end();

timer = 0;

}

timer++;

delay(10);

Per als que demanen la plana web de json, tinc una funció per preparar-ho.

String Buildjson()

{

Serial.println("Peticio JSON");

String json = "{\"\_id\":\"" + \_id + "\",\"r1\":\"" + String(r1) + "\",\"r2\":\"" + String(r2) + "\",\"r3\":\"" + String(r3) + "\",\"r4\":\"" + String(r4) + "\",\"r5\":\"" + String(r5) + "\",\"r6\":\"" + String(r6) + "\",\"r7\":\"" + String(r7) + "\",\"r8\":\"" + String(r8) + "\"}";

return json;

delay(1);

}

# CONCLUSIONS

Aquest projecte ha sigut tant una experiència com una introducció a altres branques de la tecnologia, com l’electrònica

# BIBLIOGRAFIA

Aquesta és una recopilació dels enllaços més rellevants que he utilitzat per a la realització d’aquest treball de recerca.

Alguns enllaços de les seccions de software i de les aclaracions no estan inclosos a la bibliografia per evitar-ne la repetició i fer més diﬁcultosa la lectura d’aquesta secció.

[google.com](http://www.google.com) - Cercador

[codemade.io](http://www.codemade.io) - Pagina recopiladora de projectes DIY

[raspberrypi.org](http://www.raspberrypi.org) - Web oficial Raspberry

[arduino.cc/](http://www.arduino.cc/) - Web oficial Arduino

[prometec.net](http://www.prometec.net/servidor-web-esp8266) - Creació servidor web amb arduino

[github.com](https://github.com/esp8266/Arduino/tree/master/libraries) - Llibreries chip ESP8266

[linksprite.com](http://linksprite.com/wiki/index.php5?title=LinkNode_D1) - Introducció a LinkNode D1

[w3schools.com](https://www.w3schools.com/js/default.asp) - Javascript

[w3schools.com](https://www.w3schools.com/php/default.asp) - PHP

[xtec.cat](http://www.xtec.cat/~jfresnil/domotica/Domoweb/id19.htm) - Què és la domòtica?

[fwthinking.com](http://www.fwthinking.com/blogs/how-will-the-internet-of-things-communicate.htm) - Com es comunicaran els dispositius IoT?

[stackexchange.com](https://raspberrypi.stackexchange.com/questions/8734/execute-script-on-start-up) - Executar script al botejar (Raspbian)

[xtec.cat](http://www.xtec.cat/~jfresnil/domotica/Domoweb/id19.htm) - Què es la domòtica?

[nodejs](https://nodejs.org/es/) - Executant Javascript

[npm](https://www.npmjs.com/) - Un gestor de paquets de Javascript

[websocket](https://www.npmjs.com/package/websocket) - client WebSocket per Nodejs

[ratchet](http://socketo.me/) - Servidor WebSocket

[stackoverflow](https://stackoverflow.com/questions/13322485/how-to-i-get-the-primary-ip-address-of-the-local-machine-on-linux-and-os-x) - Agafar IP amb bash

[raspberry](https://www.raspberrypi.org/products/pi-zero-w/) - Raspberry pi zero W