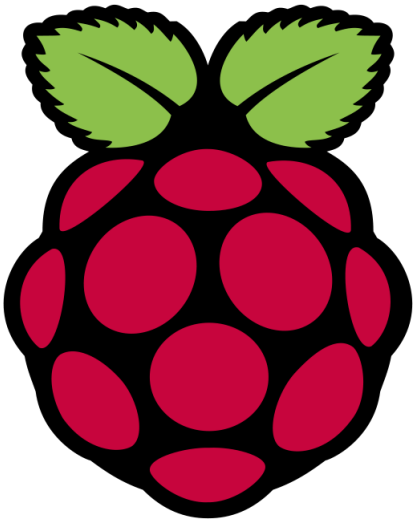
RaspiSecureSystem

Projecte de Tecnologies de la Informació

20 de desembre de 2016



Marlen Àvila | Eric Garcia | Toni Miquel Llull | Brian Martinez

Abstract

Aquest és un projecte per a l'especialitat d'IT de la carrera de *Grau en Enginyeria Informàtica* de la FIB, que consisteix en crear un petit sistema de seguretat utilitzant una Raspberry Pi (a partir d'ara, Raspi), una càmera IP, un Arduino i varis sensors que ens permetrà veure què passa a la nostra casa o habitació, i gràcies a un sensor de movient podrem fer que la càmera faci una foto i ens l'envii per correu o ens avisi amb una notificació al mòbil.

De la mateixa manera farem servir un sensor de temperatura que ens podrà servir com a "detector d'incendis", i podrem fer també que si passa d'una certa temperatura ens avisi d'alguna manera. Afegirem també l'opció de poder encendre o apagar alguns llums remotament, per si arribem tard a casa poder "fer veure"que hi ha algú.

Es pot trobat tot el codi del projecte al repositori de github

Introducció

Actualment l'*Internet de les coses*, també conegut com **IoT** per les seves sigles en anglès, està prenent molta importància en el nostre dia a dia. D'altra banda, les cases intel·ligents o domòtiques ja fa molts anys que es parla d'elles, i fins i tot a pel·lícules dels anys 80 com "Regreso al futuro"ja se'n feia alguna referència; encara així, sembla que no ha arribat a tenir una importància massa rellevant fins avui en dia, potser degut a aquest nou fenòmen de l'IoT.

Aprofitant això han sorgit varies empreses que es dediquen a connectar-te la casa amb varis sensors, càmeres, aparells, etc., però segurament degut a la poca competència que hi ha al mercat els preus són una mica elevats. És per això que hem pensat en fer el nostre petit sistema de seguretat, però a un preu molt més reduït.

Aquest sistema ens permetrà veure què passa a la nostra casa o habitació mitjançant una càmera IP, i gràcies a un sensor de movient podrem fer que es faci una foto i ens l'envii per correu o ens avisi amb una notificació al mòbil.

De la mateixa manera farem servir un sensor de temperatura que ens podrà servir com a "detector d'incendis", i podrem fer també que si passa d'una certa temperatura ens avisi d'alguna manera. Afegirem també l'opció de poder encendre o apagar alguns llums remotament, per si arribem tard a casa poder "fer veure"que hi ha algú.

Tant l'accés a la càmera com controlar els llums es podrà fer des d'una aplicació mòbil o vía web.

Per dur a terme el projecte hem fet servir els següents components:

- **Raspberry Pi (model 1 B)**: És un petit ordinador de baix cost que farem servir com a servidor, tant per la pàgina web com per tractar les senyals que ens enviaran els diferents sensors.
- **Càmera IP (Dlink DCS-932L)**: Farem servir una càmera IP que ja teniem, i gràcies a la seva connexió WiFi ens permetrà poder-la posar on vulguem, sense tenir la limitació d'haver-la de connectar directament a la Raspi.
- **Arduino (Mega)**: Aquest és el dispositiu que s'encarrega de gestionar els diferents sensors com el de moviment o temperatura, així com un relé que farem servir per encendre una bombeta. Al igual que la Raspi, és una placa de baix cost que ens permet fer una gran quantitat de coses.
- **Targeta de memòria SD de 32GB classe 10**: Aquesta targeta ens farà alhora de "disc dur"de la Raspi, i serà on s'hi enmagatzemarà tota la informació pel seu funcionament,així com les fotos que es capturin de la càmera.

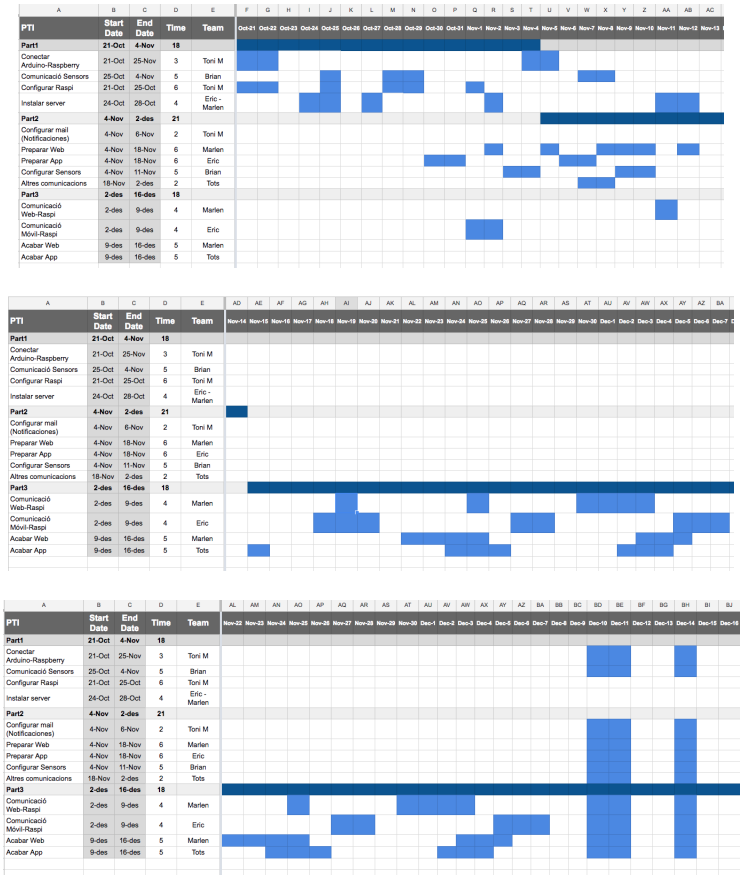
- **Sensor de temperatura**: Aquest sensor es connecta a l'Arduino i ens mesura cada 5 minuts la temperatura de l'habitació.
- **Sensor de moviment**: Serà l'encarregat de detectar si hi ha hagut moviment en la sala que vulguem controlar, i estarà connectat també directament a l'Arduino.
- **Relé**: Aquest relé ens serveix per, juntament amb l'Arduino, poder controlar una bombeta de 220v.

Metodologia

A continuació es descriuran els passos seguits per dur a terme aquest projecte, on detallarem cada un d'ells. Degut a l'extensió d'algunes de les seves parts, ho hem separat en diferents seccions per tal d'agrupar tot el que està relacionat.

D'aquesta manera tenim 6 grans seccions que són Raspberry, Càmera IP, Arduino, Aplicació Android, Web i Scripts més rellevants.

El fluxe de treball aproximat d'aquest projecte es pot representar amb el següent diagrama de gantt:



PTI	Start Date	End Date	Time	Team
Part1	**21-Oct**	**4-Nov**	**18**	
Conectar Arduino-Raspberry	21-Oct	25-Nov	3	Toni M
Comunicació Sensors	25-Oct	4-Nov	5	Brian
Configurar Raspi	21-Oct	25-Oct	6	Toni M
Instalar server	24-Oct	28-Oct	4	Eric - Marlen
Part2	**4-Nov**	**2-des**	**21**	
Configurar mail (Notificaciones)	4-Nov	6-Nov	2	Toni M
Preparar Web	4-Nov	18-Nov	6	Marlen
Preparar App	4-Nov	18-Nov	6	Eric
Configurar Sensors	4-Nov	11-Nov	5	Brian
Altres comunicacions	18-Nov	2-des	2	Tots
Part3	**2-des**	**16-des**	**18**	
Comunicació Web-Raspi	2-des	9-des	4	Marlen
Comunicació Móvil-Raspi	2-des	9-des	4	Eric
Acabar Web	9-des	16-des	5	Marlen
Acabar App	9-des	16-des	5	Tots

PTI	Start Date	End Date	Time	Team
Part1	**21-Oct**	**4-Nov**	**18**	
Conectar Arduino-Raspberry	21-Oct	25-Nov	3	Toni M
Comunicació Sensors	25-Oct	4-Nov	5	Brian
Configurar Raspi	21-Oct	25-Oct	6	Toni M
Instalar server	24-Oct	28-Oct	4	Eric - Marlen
Part2	**4-Nov**	**2-des**	**21**	
Configurar mail (Notificaciones)	4-Nov	6-Nov	2	Toni M
Preparar Web	4-Nov	18-Nov	6	Marlen
Preparar App	4-Nov	18-Nov	6	Eric
Configurar Sensors	4-Nov	11-Nov	5	Brian
Altres comunicacions	18-Nov	2-des	2	Tots
Part3	**2-des**	**16-des**	**18**	
Comunicació Web-Raspi	2-des	9-des	4	Marlen
Comunicació Móvil-Raspi	2-des	9-des	4	Eric
Acabar Web	9-des	16-des	5	Marlen
Acabar App	9-des	16-des	5	Tots

PTI	Start Date	End Date	Time	Team
Part1	**21-Oct**	**4-Nov**	**18**	
Conectar Arduino-Raspberry	21-Oct	25-Nov	3	Toni M
Comunicació Sensors	25-Oct	4-Nov	5	Brian
Configurar Raspi	21-Oct	25-Oct	6	Toni M
Instalar server	24-Oct	28-Oct	4	Eric - Marlen
Part2	**4-Nov**	**2-des**	**21**	
Configurar mail (Notificaciones)	4-Nov	6-Nov	2	Toni M
Preparar Web	4-Nov	18-Nov	6	Marlen
Preparar App	4-Nov	18-Nov	6	Eric
Configurar Sensors	4-Nov	11-Nov	5	Brian
Altres comunicacions	18-Nov	2-des	2	Tots
Part3	**2-des**	**16-des**	**18**	
Comunicació Web-Raspi	2-des	9-des	4	Marlen
Comunicació Móvil-Raspi	2-des	9-des	4	Eric
Acabar Web	9-des	16-des	5	Marlen
Acabar App	9-des	16-des	5	Tots

Índex

1 Raspberry

La Raspi és el nucli del nostre sistema, pel que és la que més treball requereix. Per això gran part d'aquest document fa referència a la seva instal·lació i configuració, i els passos a seguir es detallen tot seguit.

1.1 Instal·lar i preparar el SO

El SO escollit per aquest projecte ha estat **Raspbian** versió ***Pixel***, una distribució Linux basada en Debian-Jessie i adaptada pel chip ARM de la Raspi. Per instal·lar el SO ens hem servit de l'eina que ens proporciona la comunitat de Raspberry, anomenada **NOOBS**, el que ens permet instal·lar el sistema d'una forma senzilla.

El primer que hem de fer és baixar NOOBS de la pàgina de Raspberry Pi i descomprimir-lo. A continuació hem de donar format a la targeta; per això podem fer servir l'eina SDFormatter, oficial de SD Association. Una vegada tenim la targeta formatejada i NOOBS descomprimit, copiem tot el contingut de la carpeta NOOBS dins de la targeta.

COMPTE: NO copiar la carpeta com a únic arxiu, sino tots els arxius que trobem dins de la mateixa.

Per fer la instal·lació ens ajudarem d'una pantalla amb entrada HDMI, un teclat i un ratolí. Connectem tots els perifèrics a la Raspi i la connectem a la corrent. Després d'uns moments ens apareixerà una pantalla on ens demana quin sistema volem instal·lar. Seleccionem **Raspbian** i cliquem *Install*. Aquest procés pot tardar entre 20 i 30 minuts.

Quan la instal·lació hagi acabat, reiniciem el sistema i ja podrem arrencar Raspbian normalment.

Des del menú d'aplicacions anirem a *Preferences* → *Mouse and Keyboard Settings*, i posarem el teclat en Espanyol (Català). A continuació anirem a *Preferences* → *Raspberry Pi Configuration*, modificarem el **password**, i marcarem l'opció de **Boot** que diu *To CLI*. Això farà que quan arrenqui la Raspi no carregui l'entorn gràfic, ja que normalment no el farem servir i així tenim més recursos disponibles.

En cas que vulguem accedir a l'entorn gràfic (si tenim la Raspi conectada a una tele o pantalla, per exemple), ho podrem fer amb la comanda *startx*

```
startx
```

De moment encara mantenim la pantalla, teclat i ratolí, ja que ens queda fer algunes configuracions.

1.2 Assignar una IP estàtica

Una part important i que dóna sentit a una Raspi és el fet de poder accedir a ella des de qualsevol lloc a través d'un terminal. Això ho podem fer sempre i quan sapiguem la seva adreça IP però, com sabem, aquesta pot canviar si el router es reinicia o si reiniciem la Raspi. Per evitar-ho li assignarem una IP estàtica i així ens asegurarem que sempre té la mateixa.
Per fer això tenim varies opcions:

1.2.1 dhcpcd.conf

Abans de res ens connectem a la nostra WiFi; d'aquesta manera ja tindrem registrat el SSID i Password, i a continuació obrim un terminal amb la combinació de tecles **Ctrl+Alt+t**. Podem fer-ho també des del menú d'aplicacions.

A continuació hem de modificar l'arxiu *dhcpcd.conf*

```
sudo nano /etc/dhcpcd.conf
```

Afegint al final el següent codi si volem fer la connexió per cable:

```
interface eth0

static ip_address=192.168.1.XX/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

O aquest si la volem fer per WiFi:

```
interface wlan0

static ip_address=192.168.1.XX/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

Sortim amb **Ctrl+x**, acceptem els canvis amb **y**, i premem **enter**.

NOTA: Substituim el valor XX per l'adreça que vulguem, tenint en compte de posar un valor que estigui fora del rang DHCP. Normalment es comencen a donar adreces a partir del número 33 (tot i que pot variar), però dificilment comença per adreces baixes. Això ens permet assignar sense cap problema adreces a partir de la 2 o la 3. Igualment, hem de posar l'adreça del router i DNS que correspongui amb el nostre router, i normalment sol ser 192.168.1.1 en els routers domèstics, però assegureu-vos abans per si de cas.

1.2.2 interfaces

Una altra opció és modificar directament l'arxiu interfaces

```
sudo nano /etc/network/interfaces
```

I modifiquem les dades de wlan0 per les següents si farem servir el WiFi:

```
auto wlan0
allow-hotplug wlan0
iface wlan0 inet static
address 192.168.1.XX
netmask 255.255.255.0
gateway 192.168.1.1
wpa-passphrase wifi-password
wpa-ssid my-ssid
```

Canviant **wifi-password** pel password de la nostra xarxa, **my-ssid** pel nom de la nostra xarxa, i modificant el valor XX per l'adreça que volem.

O les dades de eth0 si ens connectarem per cable:

```
auto eth0
iface eth0 inet static
address 192.168.1.XX
netmask 255.255.255.0
gateway 192.168.1.1
```

Sortim amb **Ctrl+x**, acceptem els canvis amb **y**, i premem **enter**.

1.2.3 interfaces + wpa_supplicant.conf

Ens queda encara una alternativa, que és fer servir l'arxiu **wpa_supplicant.conf**

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Aquest arxiu és l'encarregat de guardar els noms i contrassenyes de les xarxes WiFi, i per tant haurem d'afegir al final la nostra xarxa si no ens hem connectat amb anterioritat a la xarxa WiFi. Si ja tenim una xarxa amb el nom i password correctes, no cal fer res:

```
network={
ssid="WiFi_name"
psk="Wifi_password"
}
```

Sortim amb **Ctrl+x**, acceptem els canvis amb **y**, i premem **enter**.

Ara anem a modificar l'arxiu interfaces

```
sudo nano /etc/network/interfaces
```

I canviem la part de wlan0 pel següent si farem servir la connexió WiFi:

```
allow-hotplug wlan0
iface wlan0 inet static
address 192.168.1.XX
netmask 255.255.255.0
gateway 192.168.1.1
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

O la part de eth0 si ens connectarem per cable:

```
auto eth0
iface eth0 inet static
address 192.168.1.XX
netmask 255.255.255.0
gateway 192.168.1.1
```

Com veiem, els canvis d'aquest arxiu son molt semblants a l'opció 1.2.2, però aquí no posem explícitament el nom i password de la nostra xarxa, sino que ens servim de l'arxiu wpa_supplicant.conf. Pensar a canviar el valor XX de l'adreça IP.

Una vegada modificada la configuració de la IP reiniciarem la Raspi:

```
sudo reboot now
```

I quan hagi tornat a arrencar, comprovarem que ens ha assignat l'adreça que li hem dit amb ifconfig:

```
ifcongif
```

En cas que no ho hagi fet, revisem els arxius que hem modificat per si ens hem errat en alguna cosa.

En aquest moment ens podem desfer de la pantalla, teclat i ratolí, ja que totes les comunicacions les farem per ssh. La manera d'accedir a la Raspi és, des d'un terminal de Linux/Mac teclejar el següent:

```
ssh pi@192.168.1.XX
```

Ens demanarà la contrassenya que hem modificat al punt 1.1 *(mentre s'escriu la contrassenya no es veurà res per pantalla)* i ja estarem connectats a la Raspi.

Nota: Si fem servir Windows, hi podrem accedir mitjançant l'aplicació PuTTY.

1.3 Canviar el port SSH

Addicionalment podriem voler emprar un port SSH diferent de l'standart (que per defecte és el 22). Això ens pot ser útil si volem accedir a la Raspi des de fora de casa i el port 22 del router ja el tenim assignat a un altre dispositiu, o per intentar tenir una mica més de seguretat evitant valors per defecte.

Per fer-ho modifiquem l'arxiu **sshd_config**

```
sudo nano /etc/ssh/sshd_config
```

Busquem la linia on indica el port i el canviem pel que ens interessi, ja sigui modificant la linia en qüestió o comentant aquesta i afegint una nova

```
#port 22
port 2234
```

Per comoditat, lo més pràctic és afegir 2 valors més al 22 inicial, per exemple 2234, així ens serà més fàcil d'associar el port a SSH, però podem posar el que vulguem (sempre i quan no estigui ja en ús).

Finalment reiniciem el servidor ssh

```
sudo service ssh restart
```

Una vegada fet el canvi, la manera d'accedir a la Raspi per terminal serà

```
ssh -p 2234 pi@192.168.1.XX
```

1.4 Generar una clau SSH a un dispositiu client

Una opció interessant de connectar-nos a la nostra Raspi per SSH és mitjançant claus. Això ens permet una connexió més ràpida ja que no haurem d'introduir la nostra contrassenya cada vegada i, per extensió, més segura, i és necessària per realitzar certes accions (com còpies de seguretat en dispositius remots, com és el nostre cas, i que s'explicarà a l'apartat d'scripts).

Per aconseguir aquesta connexió, primer necessitem generar les claus tant públiques com privades des del dispositiu client, que en aquest cas serà el nostre portàtil (ja que la Raspi farà les funcions de servidor). Per això, des d'un terminal executem la següent comanda

```
ssh-keygen -t rsa
```

Una vegada executada ens farà algunes preguntes, però ens bastarà amb premer enter fins que ens mostri un missatge semblant al següent

```
ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
4a:dd:0a:c6:35:4e:3f:ed:27:38:8c:74:44:4d:93:67 user@a
The key's randomart image is:
+--[ RSA 2048]----+
|          .oo.   |
|         .  o.E  |
|        + .  o   |
|     . = = .     |
|      = S = .    |
|     o + = +     |
|      . o + o .  |
|           . o   |
|                 |
+-----------------+
```

Això genera les claus públiques i privades del nostre dispositiu, i les guarda a */home/user/.sshd/id_rsa.pub* i */home/user/.sshd/id_rsa* respectivament.

Una vegada tenim les claus, executarem la següent comanda

```
ssh-copy-id pi@192.168.1.XX
```

Segurament ens sortirà un missatge semblant al següent, ons ens demanarà acceptar la connexió i introduir la contrassenya

```
The authenticity of host '12.34.56.78 (12.34.56.78)' can't be established.
RSA key fingerprint is b1:2d:33:67:ce:35:4d:5f:f3:a8:cd:c0:c4:48:86:12.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '12.34.56.78' (RSA) to the list of known hosts.
user@12.34.56.78's password:
Now try logging into the machine, with "ssh 'user@12.34.56.78'", and check in

  ~/.ssh/authorized_key

to make sure we haven't added extra keys that you weren't expecting.
```

A partir d'ara, les connexions que es facin des del nostre dispositiu cap a la Raspi no requeriran de contrassenya.

NOTA: Si el que volem és guardar la clau per fer connexions des de fora de la xarxa, farem

```
ssh-copy-id pi@elmeudomini.ddns.net
```

Però com crear dominis i accedir des de fora de la xarxa local s'explica al següent punt.

1.4.1 Esborrar una clau SSH guardada

Si el que volem fer és eliminar la clau que acabem de guardar a la nostra Raspi (per revocar l'accés automàtic, per exemple), el que farem serà obrir el següent fitxer

```
sudo nano ~/.ssh/known_hosts
```

Cercar la línia que correspon al nostre dispositiu (generalment serà la darrera línia del fitxer si no s'han afegit més claus) i l'esborrem.

A partir d'ara, ens tornarà a demanar contrassenya a cada connexió.

1.5 Instal·lar i configurar no-ip

De la mateixa manera que les direccions IP locals poden canviar, també ho fan les direccions IP públiques, pel que potser que ara en tiguem una i demà una altra. Això és un problems si intentem accedir a la Raspi des de fora de la xarxa local, ja que no sabrem quina direcció tenim assignada. Per evitar aquest problema ens ajudarem del servei no-ip.

No-ip és un servidor DNS que el que fa és traduir una direcció web (http://www.google.com) a la seva IP pública (http://74.125.224.72, per exemple). Així, el que farem ara serà accedir a www.noip.com, crear un compte (gratuit) i registrar un domini, per exemple **elmeudomini.ddns.net**.

Quan ja tenim el nostre domini registrat, toca instal·lar el client a la Raspi. Hi accedim per ssh amb:

```
ssh pi@192.168.1.XX
```

Abans ens assegurarem de tenir el sistema actualitzat, pel que farem un update i un upgrade. Això pot tardar fins a 20 o 30 minuts, depenent del que s'hagi d'actualitzar.

```
sudo apt-get update && sudo apt-get upgrade
```

És recomanable fer aquest procés cada cert temps, ja que hi pot haver actualitzacions de seguretat del sistema o d'alguna altra aplicació.

A continuació descarreguem el client de no-ip

```
wget http://www.no-ip.com/client/linux/noip-duc-linux.tar.gz
```

El descomprimim

```
tar -zxvf noip-duc-linux.tar.gz
```

Accedim a la carpeta que s'ha creat

```
cd noip-2.1.9-1
```

I l'instal·lem

```
sudo make
sudo make install
```

En aquest punt ens demanarà el nom d'usuari i la contrassenya del nostre compte de no-ip, i degut a que només tindrem un domini registrat, agafarà aquest per defecte. El temps de refresc el podem deixar per defecte, i a la següent pregunta, respondrem que NO (n).

Ara creem un nou fitxer que li direm noip2

```
sudo nano /etc/init.d/noip2
```

I hi copiarem la següent comanda

```
sudo /usr/local/bin/noip2
```

Guardem el fitxer amb **Ctrl+x**, acceptem els canvis amb **y**, premem **enter**, i li donem permissos d'execució

```
sudo chmod +x /etc/init.d/noip2
```

Actualitzem el fitxer d'inici perquè arrenqui cada vegada que engeguem la Raspi

```
sudo update-rc.d noip2 defaults
```

I posem el servei en marxa

```
sudo /usr/local/bin/noip2
```

Ens falta un darrer punt molt important, i que justifica la importància del punt 1.2. Per poder accedir des de fora de la xarxa necessitem saber l'adreça pública (problema que hem solucionat amb no-ip), però també necessitem saber a quina adreça privada volem anar. Això se soluciona fent "port-forwarding"al router i indicant que tot el que vingui des de fora que vulgui anar al port 22 (o el que haguem configurat si hem fet el punt 1.3), vagi a la nostra Raspi. Aquí veiem la importància de tenir una IP estàtica, i en el següent punt veurem com fer això.

1.6 Obrir els ports del router

Ara que ja tenim no-ip instal·lat i configurat, anirem a obrir els ports del router i indicar cap a on redirigir el tràfic.

Per aconseguir això necessitem accedir al router teclejant la seva adreça a qualsevol navegador (normalment sol ser 192.168.1.1), i posem el nom d'usuari i contrassenya. Els nous routers de fibra òptica de Movistar només demanen una contrassenya que es troba davall del router, i els més antics normalment tenen *admin* com a usuari, i *admin* o *1234* com a contrassenya. En cas que no vagi bé, s'haurien de cercar les credencials d'accés pel nostre router a Internet.

Una vegada hi hem accedit cerquem alguna opció que es digui "Ports", "Puertos"o similar (depèn molt de cada fabricant), i creem una nova regla que indiqui el següent:

```
Port: 22 (o 2234)
Tipus de protocol: TCP/UDP
Adreça destinació: 192.168.1.XX
```

Aquesta és la informació rellevant, sent XX l'adreça que hem assignat en el punt 1.2. Potser ens demana també un nom, que li podem posar SSH, per exemple.

Amb això ja tenim el port del router obert i redirigint el tràfic cap a la nostra Raspi, pel que hauriem de poder accedir-hi des de qualsevol lloc fora de la nostra xarxa local. No ho podrem provar si estem connectats a la nostra xarxa local, però ho podrem provar des del mòbil (amb una aplicació com JuiceSSH per Android) o demanant a algú que estigui en una altra xarxa que ens ho miri.

Des del terminal teclegem

```
ssh pi@elmeudomini.ddns.net
```

O si hem canviat el port ssh

```
ssh -p 2234 pi@elmeudomini.ddns.net
```

Si ens demana contrassenya (o accedeix directament si hem afegit claus SSH), vol dir que tot ha sortit bé, i ja podrem accedir a la nostra Raspi des de qualsevol lloc.

Ja que estem, aprofitarem per obrir tots els port que ens puguin fer falta i així no ho haurem de fer després un per un cada vegada que instal·lem un nou servei.

En el nostre cas, a banda del port SSH, obrirem els següents apuntant també a l'adreça de la nostra Raspi:

```
Ports: 80 (servidor web per defecte)
       21 (servidor FTP per defecte)
Protocols: TCP/UDP
IP destí: 192.168.1.XX
```

A més a més, ja que també farem servir una càmera IP, podem obrir el port que li correspondrà:

```
Port: 8081 (per exemple)
Protocols: TCP/UDP
IP destí: 192.168.1.XY
```

Canviarem XY per l'adreça que li vulguem donar a la nostra càmera, que més endavant li assignarem.

1.7 Instal·lar i configurar un servidor FTP

Un servidor ftp ens pot servir per agafar dades de la Raspi d'una forma senzilla. Per això, tot i que les captures i vídeo de la càmera es faran directament sobre la IP de la pròpia càmera, inicialment vam fer servir el servidor ftp per enviar les fotos, pel que creiem convenient explicar el seu procés de configuració.

Farem servir el servidor vsftpd, pel que procedim a la seva instal·lació

```
sudo apt-get install vsftpd
```

I un cop instal·lat, obrim l'arxiu de configuració per realitzar alguns canvis

```
sudo nano /etc/vsftpd.conf
```

Modificarem les linies que es mostren a continuació:

```
anonymous_enable=NO
....
local_enable=YES
...
write_enable=YES
...
```

Amb això permetrem poder interactuar amb els arxius, i no permetem l'accés anònim.

Tot i així, si no fem cap més modificació, tots els usuaris podrien accedir a tots els arxius. Si volem restringir només l'accés a les carpetes de cada usuari, modifiquem la següent linia:

```
chroot_local_user=YES
```

1.7.1 Connexió per SSL

Si volem afegir un certificat SSL a la nostra conexió, realitzarem els següents passos.

Primer creem el certificat amb la següent comanda:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:1024 \
-keyout /etc/ssl/private/vsftpd.pem -out /etc/ssl/private/vsftpd.pem
```

Això ens crea varis arxius, la ruta dels quals haurem d'afegir al fitxer de configuració del servidor. Per tant, el tornem a obrir

```
sudo nano /etc/vsftpd.conf
```

I al final de l'arxiu cerquem la línia que habilita l'accés per ssl i la descomentem. Afegim també, si cal, les rutes dels certificats

```
rsa_cert_file=/etc/ssl/private/vsftpd.pem
rsa_private_key_file=/etc/ssl/private/vsftpd.pem
ssl_enable=YES
```

I afegim les següents linies al final del fitxer

```
allow_anon_ssl=NO
force_local_data_ssl=YES
force_local_logins_ssl=YES

ssl_tlsv1=YES
ssl_sslv2=NO
ssl_sslv3=NO

require_ssl_reuse=NO
ssl_ciphers=HIGH
```

Un cop hem guardat l'arxiu (**Ctrl+x**, **y**, **enter**), ens queda reiniciar el servidor:

```
sudo service vsftpd restart
```

1.7.2 Canviar el port ftp

Igual que hem fet amb el port ssh, podem tenir la necessitat de modificar el port ftp. Per això, obrim el fitxer de configuració

```
sudo nano /etc/vsftpd.conf
```

I realitzaem els següents canvis a les linies corresponents

```
#connect_from_port_20=YES
ftp_data_port=2121
listen_port=2121
```

Aquí, com amb ssh, podem posar el port que vulguem sempre i quan estigui disponible, però la millor opció és mantenir el 21 inicial (per fer referència a ftp), i afegim dos digits més.

I per acabar, reiniciem el servidor

```
sudo service vsftpd restart
```

1.8 Instal·lar un servidor Web

De servidors web n'hi ha varis, i potser un dels més coneguts sigui Apache. Tot i que la Raspi no té problemes per treballar amb Apache, potser amb un de més lleuger com nginx o lighttph ja en tenim prou. Aquí es detalla la instal·lació d'Apache i nginx juntament amb php, però només cal instal·lar-ne un.

1.8.1 Apache

La instal·lació d'Apache és relativament senzilla, però requereix d'alguns preparatius. Primer crearem un nou grup *www-data*

```
sudo addgroup www-data
```

I afegim els usuaris al grup

```
sudo usermod -a -G www-data www-data
```

Ara, per instal·lar el servidor, ho farem amb

```
sudo apt-get install apache2 apache2-utils php5 libapache2-mod-php5
```

Iniciem el servidor

```
sudo /etc/init.d/apache2 restart
```

I provem que funciona accedint a la direcció IP de la Raspi si estem a la mateixa xarxa

```
http://IP_ADDRESS
```

O al nom del domini que hem creat a no-ip si estem a una xarxa externa

```
http://elmeudomini.ddns.net
```

Per provar php, creem el següent fitxer

```
sudo nano /var/www/html/info.php
```

Amb el següent contingut

```
<?php
phpinfo();
?>
```

I provem que funciona accedint a

```
http://ID_ADDRESS/info.php
```

1.8.2 nginx

Per instal·lar nginx, només hem de executar la comanda

```
sudo apt-get install nginx php5 libapache2-mod-php5
```

Una vegada instal·lat, creem la carpeta */var/www/html* en cas que no existeixi

```
sudo mkdir /var/www/html
```

I modifiquem l'arxiu de configuració de nginx

```
sudo nano /etc/nginx/sites-available/default
```

Deixant-lo de la següent manera

```
server {
    listen 80;
    server_name $domain_name;
    root /var/www/html;
    index index.html index.htm;
    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;
}
```

Reiniciem el servidor

```
sudo service nginx restart
```

I comprovem que funciona escrivint a qualsevol navegador l'adreça IP de la Raspi si estem connectats a la mateixa xarxa

```
http://IP_ADDRESS
```

o el nom del nostre domini si estem a una xarxa diferent.

```
http://elmeudomini.ddns.net
```

Per instal·lar php executem

```
sudo apt-get install php5 libapache2-mod-php5
```

Obrim l'arxiu de configuració

```
sudo nano /etc/nginx/sites-available/default
```

I el deixem de la següent manera

```
server {
    listen 80;
    server_name $domain_name;
    root /var/www/html;
    index index.html index.htm;
    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    location ~\.php$ {
        fastcgi_pass unix:/var/run/php5-fpm.sock;
        fastcgi_split_path_info ^(.+\.php)(/.*)$;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param HTTPS off;
        try_files $uri =404;
        include fastcgi_params;
    }
}
```

Ara només ens falta reiniciar el servidor

```
sudo service nginx restart
```

Crear un fitxer php

```
sudo nano /var/www/test.php
```

Amb el següent codi de prova

```
<?php
phpinfo();
?>
```

I provar que el php funciona correctament accedint a

```
http://IP_ADDRESS/test.php
```

O des de fora de la xarxa

```
http://elmeudomini.ddns.net/test.php
```

1.9 Instal·lar MySQL y phpMyAdmin

Per poder tenir una BD i treballar amb ella en cas de ser necessari, instal·larem MySQL

```
sudo apt-get install mysql-server mysql-client php5-mysql
```

Durant la instal·lació ens demanarà certa informació que anirem emplenant, i una vegada acabada iniciem el servei amb

```
sudo service mysql start
```

I provem si funciona

```
mysql -u root -p
```

Introduïm el password que hem escollit durant la instal·lació, i si carrega correctament la instal·lació ha sortit bé. Premem **Ctrl+c** per sortir.

Per instal·lar phpMyAdmin fem

```
sudo apt-get install libapache2-mod-auth-mysql php5-mysql phpmyadmin
```

Durant aquesta instal·lació li direm quin servidor fem anar, quan ens demani si volem fer servir MySQL li direm que sí, i posarem una contrassenya per phpMyAdmin.

A continuació obrirem el següent fitxer

```
sudo nano /etc/php5/apache2/php.ini
```

I afegirem la següent linia al principi del fitxer

```
extension=mysql.so
```

Després executem la següent comanda

```
sudo ln -s /etc/phpmyadmin/apache.conf /etc/apache2/conf.d/phpmyadmin.conf
```

I fem un reload del servidor

```
sudo /etc/init.d/apache2 reload
```

Provem a veure si podem accedir a phpMyAdmin amb

```
http://IP_ADDRESS/phpmyadmin
```

Nota: Tots els fitxers html, php, etc. que farem servir es trobarem a la carpeta "web"del repositori de github.

1.10 Configurar notificacions amb l'API de Pushover

Ja que les notificacions push natives a la nostra pròpia aplicació requereixen de certs passos una mica farragosos (registrar l'app a Google, tenir compte de Google Developer, etc.), hem optat per servir-nos d'una aplicació que ens proporciona aquesta funcionalitat: **Pushover**.

El que necessitem primer de tot és crear un compte a la web de Pushover (la qual cosa ens generarà una clau única d'usuari) i posteriorment crearem una "aplicació"a la mateixa web. Posteriorment baixarem l'app de Pushover de Google Play o Apple Store.

Una vegada tenim la nostra clau d'usuari, el token de la nostra "aplicaciói l'aplicació mòbil instal·lada, anem a la Raspi i creem un nou arxiu que li direm, per exemple, *notificacions.sh*.

Una recomanació és crear una carpeta al nostre home on guardar els nostres scripts, així ho tenim tot més ordenat.

```
mkdir scripts
cd scripts
sudo nano notifications.sh
```

I dins d'aquest fitxer copiem el següent codi:

```
#!/bin/bash

MSG="message=Escriure aquí el missatge desitjat"

curl -s \
    --form-string "token=APPTOKEN" \
    --form-string "user=USERTOKEN" \
    --form-string "$MSG" \
    https://api.pushover.net/1/messages.json
exit 0
```

Aquí haurem de substituir *APPTOKEN* i *USERTOKEN* pel token de la nostra app i la clau d'usuari, respectivament.

I finalment li hem de donar permisos d'execució

```
sudo chmod +x notificacions.sh
```

Aquest seria l'script base per enviar notificacions des de la Raspi al nostre mòbil, i ho podem provar simplement executant l'script

```
./notificacions.sh
```

A partir d'aquí, ens podem crear tants scripts com vulguem per enviar notificacions depenent del que ens interessi. Inclús podem cridar aquest script des d'un altre script, que de fet serà lo més habitual.

1.11 Configurar notificacions per Telegram

Aclaració: En un principi ens havia semblat una bona idea fer servir Telegram com a mitjà de notificacions però ens vam adonar de que, tot i que permet enviar-te missatges a tu mateix, aquests missatges no es reben com una notificació (ja que se suposa que ets tu qui està enviant aquest missatge), pel que no ens serveix en el nostre proposit. Encara així, expliquem el seu procés d'instal·lació i configuració.

Telegram és una aplicació de missatgeria molt versàtil i gràcies a que és de programari lliure ens permet treballar amb ella més enllà d'enviar missatges als nostres amics.

Per instal·lar Telegram, primer necessitem les següents llibreries

```
sudo apt-get install libreadline-dev libconfig-dev libssl-dev \
lua5.2 liblua5.2-dev libevent-dev libjansson-dev libpython-dev make
```

I després baixar el fork de Telegram per Linux

```
git clone --recursive https://github.com/vysheng/tg.git && cd tg
```

I fem

```
./configure
make
```

Una vegada instal·lat l'executem amb

```
bin/telegram-cli -k tg-server.pub -W
```

NOTA: Potser en aquest punt ens dóna un error com el següent

```
pi@raspberrypi:~/tg $ bin/telegram-cli -k tg-server.pub -W
Telegram-cli version 1.4.1, Copyright (C) 2013-2015 Vitaly Valtman
Telegram-cli comes with ABSOLUTELY NO WARRANTY; for details typ
'show_license'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show_license' for details.
Telegram-cli uses libtgl version 2.1.0
Telegram-cli includes software developed by the OpenSSL Project
for use in the OpenSSL Toolkit. (http://www.openssl.org/)
I: config dir=[/home/pi/.telegram-cli]
> telegram-cli: tgl/mtproto-utils.c:101: BN2ull: Assertion '0' failed.
SIGNAL received
```

Si és així, haurem de modificar el fitxer *tgl/mtproto-utils.c*, comentar les linies **101** i **105**, i tornar a compilar.

Quan el tenim correctament instal·lat, la primera vegada que l'executem ens demanarà el nostre número de telèfon, al qual haurem d'incloure el codi de país (+34 en el cas d'Espanya), i introduir el codi que rebrem via Telegram al nostre mòbil.

Ara, per enviar missatges ho farem amb la següent comanda

```
msg Name_Lastname message
```

NOTA: Name és el nom del nostre contacte i Lastname el cognom tal i com ens el mostra Telegram a l'aplicació, i s'han de separar amb la barra baixa (_)

1.11.1 Script per enviar missatges més fàcilment

Podem crear un script per enviar missatges més còmodament sense haver de tenir el client en marxa continuament. Per això, crearem un script amb el nom que vulguem

```
sudo nano /home/pi/scripts/tg.sh
```

i hi afegirem el següent codi

```
#!/bin/bash
to=$1
msg=$2
tgpath=/path/to/telegram/tg
cd ${tgpath}
cmd="bin/telegram-cli -W -k server.pub -e \"msg $to $msg\""
eval $cmd
```

NOTA: Tenir en compte que la variable **tgpath** ha de contenir la ruta on hem ubicat la carpeta tg que ens hem baixat del reposiroti.

I li donem els permisos necessaris

```
sudo chmod 0655 /home/pi/scripts/tg.sh
```

Per executar-lo, fem

```
tg.sh Name_Lastname Message
```

1.12 Enviar notificacions quan algú fa login

Una funcionalitat interessant és la d'enviar una notificació quan algun usuari accedeix a la Raspi. Per això farem servir un scrip molt similar al del punt 1.10.

```
sudo nano loginNotification.sh
```

I hi afegim el següent codi

```
#!/bin/bash

#Using PAM
MSG="message=Login from user $PAM_USER"

if [ "$PAM_TYPE" != "close_session" ]; then
curl -s \
    --form-string "token=APPTOKEN" \
    --form-string "user=USERTOKEN" \
    --form-string "$MSG" \
    https://api.pushover.net/1/messages.json
fi
exit 0
```

El que fa aquest codi és executar el curl només quan es fa login gràcies a la condició, ja que en principi no ens interessa saber quan es desconnecta. En cas que també ho vulguem saber, n'hi ha prou amb treure el condicional o afegir un else si volem enviar un missatge diferent en cada cas.

Li hem de donar també els permisos d'execució

```
sudo chmod +x loginNotification.sh
```

I per fer que aquest script funcioni, ens queda modificar un arxiu, i és **important fer-ho amb cura ja que un error ens pot bloquejar l'accés a la Raspi**.

Obrim el següent fitxer

```
sudo nano /etc/pam.d/sshd
```

Busquem la part de codi següent, i afegim les linies corresponents al nostre script, tenint cura de no modificar la resta de codi

```
.............
# SELinux needs to be the first session rule.  This ensures that any
# lingering context has been cleared.  Without this it is possible that a
# module could execute code in the wrong domain.
session [success=ok ignore=ignore module_unknown=ignore default=bad
pam_selinux.so close

# El meu script
session required pam_exec.so seteuid /home/pi/scripts/loginNotification.sh
# Fi del meu script

# Set the loginuid process attribute.
session    required     pam_loginuid.so

# Create a new session keyring.
session    optional     pam_keyinit.so force revoke
............
```

Sortim amb **Ctrl+x**, acceptem els canvis amb **y**, i premem **enter**.

Ara, per provar que tot ha sortit bé, obrirem un nou terminal amb Ctrl+alt+t, i provarem a fer login amb el nostre usuari

```
ssh pi@192.168.1.XX
```

Si després d'introduir la contrassenya aconseguim entrar, és que tot ha sortit com esperàvem. En cas que una vegada introduïda la contrassenya ens tregui fora de la sessió, hem de revisar el que hem fet i buscar algun error. Per això és important fer aquesta prova des d'un altre terminal, ja que si hem fet alguna cosa malament i sortim de la sessió actual no podriem tornar a accedir a la Raspi.

2 Càmera IP

En quant a la càmera IP, normalment tenen de per sí una interficie web amb la que podem configurar ja algunes opcions, però en el nostre cas ens interessa dependre lo mínim possible d'aquest sistema i interactuar directament amb ella a través de la seva IP.

Això ens dóna la llibertat de poder crear la nostra pròpia web o aplicació mòbil i accedir a la imatge de la càmera o fer captures gràcies als sensors que muntarem a l'Arduino. Tot i així, per configurar la càmera farem servir la pròpia interficie web, ja que al ser un entorn gràfic és més còmode.

2.1 Configuració

Segons les instruccions del fabricant, les càmeres IP tenen una adreça establerta a la qual s'hi pot accedir des d'un navegador si la connectem a un ordinador per cable ethernet, pel que la configuració inicial la farem des d'un ordinador. Una vegada sabem quina adreça té, i després d'haver introduït el nom d'usuari i contrassenya per defecte, farem les següents configuracions:

1. Assignar un port diferent del port 80 que ve per defecte, que serà el que hem obert al router abans, destinat a la càmera.
2. Canviar la contrassenya d'accés
3. Configurar la connexió FTP per comunicar-se amb la Raspi (Opcional)
4. Anotar l'adreça MAC de la càmera
5. Configurar la connexió WiFi afegint el SSID i contrassenya de la nostra xarxa

Nota: Cada càmera té la seva interfície, pel que on està cada opció pot variar.

Pel que fa referència a la IP, com podem imaginar també li hem d'assignar una IP estàtica, però en aquest cas hem optat per fer-ho des del router utilitzant l'adreça mac de la càmera, ja que no podem editar cap fitxer a la càmera (o no tan fàcilment) com sí podiem fer amb la Raspi.

Accedim al router i anem a l'opció de control MAC (novament, cada router és diferent). Afegim la MAC de la càmera i li assignem la mateixa IP que hem posat al obrir el port corresponent en el punt 5.

Després d'això, ja podem desconnectar la càmera de l'ordinador, apagar-la i tornar-la a encendre. Si tot ha sortit com esperavem, no hi hauria d'haver problema en accedir a la càmera des de qualsevol navegador web, simplement posant l'adreça IP seguida de : i el port, si estem dins de la xarxa local

```
http://192.168.1.XY:8081
```

O fent servir el nostre domini de no-ip, i que gràcies a la Raspi sempre estarà actualitzat (requereix haver obert els ports del router previament i haver redirigit les peticions a la IP de la càmera)

```
http://elmeudomini.ddns.net:8081
```

Amb això accedim al menú de la pròpia càmera i podrem realitzar les configuracions pertinents en cas de ser necessari, però com hem dit abans, no serà aquesta la forma en que agafarem les fotos, sino que ens servirem de les peticions per http que ens ofereix (i que podem consultar als forums de Dlikn. Si es fa servir una càmera diferent, s'haura de consultar el manual d'usuari i les comandes necessaries per configurar la càmera i addecir a ella.

Ja podem col·locar la càmera on vulguem, sempre que estigui dins de la cobertura WiFi.

2.2 Comandes bàsiques

Com hem dit, la forma amb la que interactuarem amb la càmera serà fent servir les pròpies peticions web que ens permet fer, i no haver de dependre de la seva interficie web. En concret, tenim 2 crides bàsiqes:

Agafar una foto

```
http://user:pass@IPADDRESS:PORT_NUMBER/image.jpg
```

I agafar el vídeo

```
http://user:pass@IPADDRES:PORT_NUMBER/video.cgi
```

Tant per una comanda com per l'altre es requereix un nom d'usuari i contrassenya; des de l'app es guarden dins de les opcions quan l'obrim per primera vegada, i des de la web s'han d'introduir manualment amb un procés de login.

Per les fotos que agafarà la Raspi quan rebi un avís de moviment de l'Arduino farem servir directament la IP de la càmera ja que ambdos dispositius estaran a la meteixa xarxa local.

```
wget http://user:pass@LOCAL_IP_CAM:PORT_CAM/image.jpg
```

3 Arduino

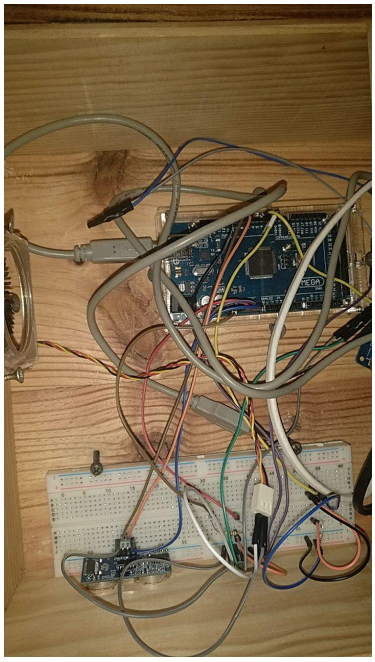
Un dels aspectes més importants d'aquest projecte recau en aquesta part. Bàsicament defineix la capa física de tot el projecte que es relacionarà directament amb el servidor per tal de oferir la informació pertinent a la web i a la aplicació Android.

S'ha escollit treballar amb Arduino per la facilitat que dona a l'hora de obtenir dades de diferents sensors, per el poc consum energètic que implica tenir-lo en funcionament molta estona i per el poc cost que implica haver de comprar-la. Hem decidit posar tot el contingut elèctric dins d'una caixa per tal de ocultat tot aquest contingut a l'usuari final i fer-lo més atractiu per aquest.

3.1 Material utilitzat

3.1.1 Placa Arduino Mega i ProtoBoard

Com s'ha indicat a la introducció, en el nostre cas, hem fet servir una placa Arduino Mega quer permetia treballar amb diversos sensors a l'hora. Aquest, es comunicava directament amb els sensors per una banda, i amb el servidor per l'altre.



Per tal de recollir les dades dels sensors, hem fet ús de una ProtoBoard que permet fer les connexions en serie i en paralel de tots els sensors donat que l'Arduino no disposa de infinits ports VCC de 5V o ports GROUND. Això permet tenir cada sensor distribuit per la ProtoBoard amb les seves propies connexions, aprofitant un sol port de l'Arduino per donat potència o terra a tots els sensors.

El fet de utilitzar una protoBoard ens donava major seguretat, ja que ens donava la possibilitat d'incorporar una resisència de 150Ω per tal de que el voltatge no es sobrepassi. Com s'ha comentat més a dalt, l'Arduino es comunica també amb el servidor. Això ho fa mitjançant el port serie per cable directament. El fet de utilitzar una comunicació per cable i no per inalàmbrica (Wifi) permet donar més fiabilitat a la comunicació.

3.1.2 Sensor de temperatura

Una de les funcions que ofereix el nostre projecte és el fet de donar-te la temperatura que hi ha a la sala on està posat el sensor en ºC. Aquest valor es mesura cada 60 segons, per tant, sempre es mostra el valor més recent. Això permet que la lectura sigui fiable i que es pogui saber amb poc temps, quina és la temperatura just en aquell moment.

El sensor que hem utilitzat és el DHT11, que també et dona la possibilitat de prendre els valors de humitat. Cal dir que aquest sensor no és 100% fiable i per tant els valors resultats poder ser un arrodoniment del valor real. És per això que el valor final pot tenir algunes dècimes d'error.

Els valors es mesuraran cada 1 minut i s'enviaran al servidor, que s'encarregarà de guardar tots aquests valors en un fitxer de text, que podrà ser llegit per l'aplicació Android.

3.1.3 Sensor de moviment

Un dels elements protagonistes és el sensor de moviment. Aquest és el que ens permetrà directament, detectar el moviment i enviar al servidor les dades necessàries. Inicialment vam treballar amb un sensor de moviment que funcionava per infrarojos. Per problemes tècnics vam haver de cercar una alternativa i vam escollir un sensor de moviment per ultrasons.

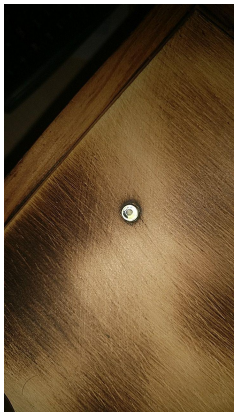
El sensor de moviment per ultrasons té dos cilindres. Un que actúa de emisor de l'ona i l'altre de receptor. El sensor està continuament enviant ones i rebent-les, calculant en cada cas el temps que s'ha trigat des de l'enviament fins la recepció. Donat que la velocitat del so és constant (343 m/s) i el temps el calcula el sensor, podem saber quina és la distància a la que es troba el primer objecte que hi ha davant:

$$Vso = D/tßD = (Vso * t)/2ßD = (343m/s * t)/2$$

Cal dividir per 2 per tal de calcular la distància de l'objecte real, ja que t és el doble (anar i tornar).

Actualment el sensor de moviment detecta moviment quan algun objecte es troba a menys de 20 cm. Hem escollit aquest valor per tal d'evitar interferències amb l'entorn.

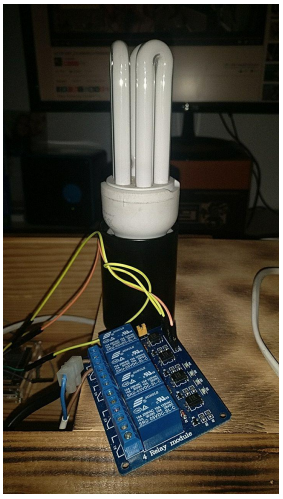
Una vegada s'ha detectat moviment, s'envia les dades al servidor, que s'encarregarà de fer una serie d'accions pertinents. A més a més, s'encendrà un led (Figura 3) durant 5 segons que indicarà la presencia de moviment.



En el moment en que es detecti moviment, el sistema estarà 10 segons sense controlar el moviment, per tal d'evitar enviar continuament dades al servidor en el cas de que una persona o objecte es quedi molta estona davant del sensor.

3.1.4 Llum i Shield Relé

Una altra funció molt important del projecte és el fet de controlar les llums de la vivenda. En el nostre cas disposem d'una bombeta que faria la funció de llum a una habitació. Donat que el llum funciona a 220V i el Arduino com a màxim a 5V, ha estat necessari acoplar un relé que permeti per una banda fer de interruptor del llum i per una altra, aconseguir que funcioni la bombeta amb l'Arduino sense que es cremi la placa. Podem veure a la Figura 4 un exemple del muntatge entre el llum i el relé:



Per tal de fer funcionar la bombeta cal que un dels extrems es connecti a la corrent directament.

3.2 Script Arduino

Anteriorment hem vist detalladament informació de cada component i hem explicat que l'Arduino es comunicaba amb el servidor a través del port serie. Per tal de comunicar-se i d'obtenir les dades dels sensors, ha estat necessari l'implementació d'un script fet amb el llengutge d'Arduino (basat en C++) amb l'ajuda de l'Arduino IDE.

```
#include "DHT.h"
int temp; //temperatura
long distancia; //distancia del objeto al sensor
long tiempo; // tiempo de ultrasonidos
int pin=2; //pin de temperatura
DHT dht(pin,DHT11); //sensor tipo DHT11
int cont=28; //cada 10 segundos aprox cambia
int cont_temp = 122; // contador de temperatura. cada 1 min cambia
boolean trobat=false; //cada 10 segundos cambia movement sensor
boolean trobat_temp = false; // cada 1 min cambia temperature sensor
int relay = 12; //pin del relé
char val = '0'; // 1=encender luz, 0= apagar luz
int led = 5; //pin del led
void encender_luz(){
  if(Serial.available()>0){
    val = Serial.read();
    if(val == '1'){
      digitalWrite(relay,LOW); //ENCENDER
    }
    else{
      digitalWrite(relay,HIGH); //APAGAR
    }
  }
}

void movement(){
  digitalWrite(9,LOW); /* Por cuestión de estabilización del sensor*/
  delayMicroseconds(10);
  digitalWrite(9, HIGH);
  delayMicroseconds(10);
  tiempo=pulseIn(8, HIGH);
  distancia= int(0.017*tiempo);
  if (distancia < 20 and cont == 28 and distancia != 0){
   digitalWrite(led,HIGH);
   Serial.println("dd"); //enviamos una d por el puerto serie
   Serial.flush();
    trobat=true;
  }
```

```
  if(cont == 14) digitalWrite(led,LOW);
  if(trobat) --cont;
  if(cont==0) {
    cont=28;
    trobat=false;

    }
}

void temperature(){
  temp = dht.readTemperature();
  if(!isnan(temp)){
    if (cont_temp == 122){
    Serial.println(temp - 3); // enviamos la temperatura por el puerto serie
    Serial.flush();
    trobat_temp = true;
    }
  }
  if(trobat_temp) --cont_temp;
  if(cont_temp == 0) {
    cont_temp = 122;
    trobat_temp = false;
  }
}

void setup(){
  Serial.begin(9600);
  pinMode(9, OUTPUT); /*activación del pin 9 como salida: para el pulso ultrasónico*/
  pinMode(8, INPUT); /*activación del pin 8 como entrada: tiempo del rebote del ultraso
  pinMode(relay,OUTPUT); // pin 12 de salida para la luz
  digitalWrite(relay,HIGH); //DEFAULT APAGADO
  dht.begin(); //init del sensor de temperatura
}


void loop() {
  encender_luz();
  movement();
  temperature();
  delay(100);
}
```

3.3 Scripts python a la Raspberry

Abans hem comentat l'script que permet comunicar l'Arduino amb la Rapberry. Però cal que la Raspberry faci alguna acció al rebre les dades. Per això s'ha fet tres scripts en Python. Dos d'ells que controlen el llum (encendre i apagar) i un tercer que s'encarrega de la temperatura i el moviment.

3.3.1 llums

Amb la finalitat de donar més granularitat, independència i fiabilitat a les capes superiors com l'aplicació móvil i la web, s'ha separat en dos scripts diferents el fet d'encendre o apagar el llum. D'aquesta manera es podien cridar de manera indipendendent des de fora.

Ambdos scripts bàsicament envien una senyal a l'Arduino que interpretarà i farà el que sigui corresponent, encendre o apagar el llum.

```
import serial
arduino = serial.Serial('/dev/ttyACM0',9600)
arduino.flushInput()
arduino.write("1")
arduino.close()
```

```
import serial
arduino = serial.Serial('/dev/ttyACM0',9600)
arduino.flushInput()
arduino.write("0")
arduino.close()
```

3.3.2 Moviment i temperatura

L'script del servidor que s'encarrega de rebre les dades de moviment i temperatura seria com el de la Figura 8. El que fa bàsicament, es anar consultant el port serie de forma activa per tal de cercar si han arribat dades. En cas de detectar moviment, s'enviará "dd" al port serie. L'script ho detectarà, netejarà el bus serie i farà una fotografia que s'enviarà per correu.

```
import serial
import os
import time

arduino=serial.Serial('/dev/ttyACM0',9600)
arduino.flushInput()
arduino.flushOutput()
while True:
    x = arduino.read(2).rstrip('\n')
    arduino.flushInput()
    arduino.flushOutput()
    if x == ("dd"):
        os.system("/home/demo/getImage.sh > /dev/null")
    elif x.isdigit():
        file = open("/home/demo/temperatura.txt","a+")
        file.write(x + '\n')
        file.close()
    arduino.flushInput()
    arduino.flushOutput()
arduino.close()
```

En cas de detectar el valor de la nova temperatura, la guardarà a un fitxer de text a la mateixa carpeta de l'script i netejarà el bus serie.

3.4 Limitacions o problemes trobats

El fet d'utilitzar aquests components ha tingut les seves limitacions o problemes, encara que finalment ha funcionat tot com es volia.

3.4.1 Comunicació Serie amb el servidor

Donat que el servidor fa una espera activa, intentant llegir dades del bus serie continuament, poden haver-hi dades errònies que cal descartar. En aquest sentit, s'ha hagut d'estudiar molt estrictament els casos on la lectura o escritura sobre el bus no era la correcta. Per tal de solventar això, hem hagut d'escollir bé quines dades s'enviaven al servidor i com s'enviaven i, a més a més, hem hagut de netejar el bus cada vegada que es rebía informació útil.

3.4.2 Ones del sensor de moviment

Donat que les ones es mouen per l'espai de l'habitació on tenim el sensor, podia passar que apareguin interferències donat al rebot de les ones amb l'espai. Això podia donar resultats incorrectes a l'hora de determinar si s'ha detectat moviment o no. Per solventar això hem hagut de disminuir la distància de detecció a 20 cm. on inicialment treballàvem amb 50cm. Amb aquesta disminució no hem trobat cap tipus d'error.

4 Aplicació Android

4.1 Instal·lació d'Android Studio i resolució de dependencies

Per tal de poder desenvolupar correctament una aplicació Android, hem optat per la utilització d'Android Studio. Eina que proporciona Google per a aquest fi. Hi han altres alternatives com Eclipse, però les hem descartat ja que avui en dia l'IDE proporcionat per Google està dissenyat per a facilitar la feina al desenvolupador, a part que conté un emulador propi per tal de provar l'aplicació en diferents smartphones.

L'instal·lació l'hem fet seguint les recomanacions de Google, en concret em seguit el següent enllaç.

Típicament, en un entorn GNU/Linux o MacOS, la instal·lació sol donar problemes si hi ha més d'una versió de Java instal·lada. La solució és simple, indicar el PATH del directori de Java que volem utilitzar.

4.2 Activities i Fragments

Amb l'IDE instal·lat, ja podem determinar quin serà a grans trets el funcionament de la nostra aplicació, i així, escollir una interfície adecuada i còmoda per a l'usuari. Per aquest fi hem utilitzat Fragments dintre d'Activities, això ens permet més versatilitat dintre de l'aplicació, ja que no haurem de canviar de vistes tan sovint, maximitzant l'eficiència de l'aplicació i oferint un entorn més agradable.

La utilització de Fragments no és trivial, ja que requereix tindre certs conceptes clars a l'hora de treballar amb ells. Això és degut al seu funcionament, i és que a l'hora d'utilitzar un Fragment, estàs utilitzant també l'Activity pare d'aquest Fragment. A més, els diferents Fragments utilitzats, poden disposar de layouts diferents, fet que complica la situació quan es vol canviar d'una vista a una altra. En el nostre cas, la solució ha estat utilitzar una pila, de manera que al crear un nou Fragment aquest s'introdueix a la pila i així després pot ser tractat sense més complicació. Si no es pren aquest tipus de mesures, al crear un Fragment, deixes de tenir accés directe a ell. Per tant, si es volgués "matar" aquell Fragment no es podria.

4.3 Comunicació amb càmera IP

Com hem comentat prèviament, farem ús d'una càmera IP per tal de visualitzar el video en streaming. Aquesta opció està implementada en l'aplicació com a un Fragment.

En les primeres versions, es va optar per un Activity amb dues opcions, càmera i video, es a dir, l'usuari tenia l'opció d'escollir entre veure el video en streaming, o simplement veure una foto estàtica. Ja treballant en la segona versió de la app, ens vam donar compte que això no tenia sentit, ja que la utilitat real per a l'usuari és veure en temps real el que transmet la càmera. Així, hem optat per un accés directe al video en streaming amb opció de fer una captura de la foto en cas que es vulgui.

Per a visualitzar el video transmés per la càmera IP s'ha fet ús de la classe Webview que proporciona Android. Aquesta classe ens permet simular el funcionament d'un navegador web a l'aplicació. Simplement hem hagut d'indicar quin és l'URL a mostrar, i la mateixa classe s'encarrega de tot.

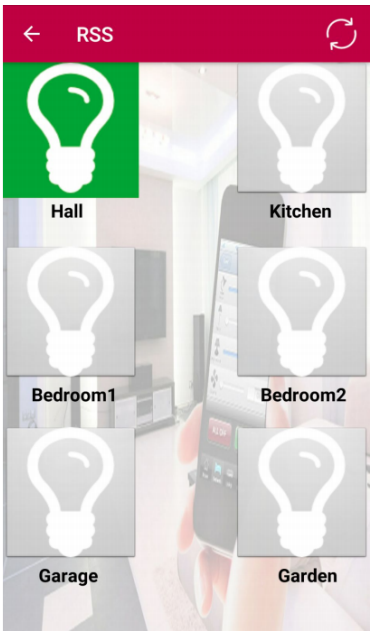
La complicació en aquest apartat ha estat l'utilització d'autenticació per part del client, ja que s'ha hagut de modificar l'objecte Webview per tal d'indicar-li com ha d'autenticar-se. A part, s'han definit una série d'opcions al objecte per a que mostri el video com nosaltres volem, ja que sinó la imatge no es visualitza correctament en segons quines pantalles d'smartphone.

Ja que la visualització de la càmera ha estat considerada per part nostra com a una funcionalitat de vital importància per a l'usuari, hem decidit afegir un floating button a la pantalla principal de l'aplicació per tal de tindre un accés ràpid i directe a la càmera.

4.4 Gestió del enllumenat

Una funcionalitat bàsica de la nostra aplicació de domótica és l'iluminació. Per tant, hem dedicat una Activity per al funcionament de les llums. L'implementació d'aquest apartat ha sigut relativament senzilla, ja que el funcionament bàsic és detectar quan un botó s'ha premut, i amb l'identificador d'aquell botó, encendre un llum o un altre. Fent ús del métode setOnClickListener (View v), hem pogut determinar l'acció de premer el botó i així, a través d'una funció per a la connexió amb el servidor, executar l'script pertinent que farà encendre o apagar el llum.

Per tal de fer connexions a un servidor remot, hem hagut de gestionar el funcionament de l'aplicació d'una manera diferent. Això és degut al funcionament de les aplicacions d'Android en sí. A l'hora de fer una petició que requereix connexió a l'exterior, Android empra el mateix procés que executa l'Activity per a aquest fí, això comporta que si la connexió falla, l'Activity "mor", per tant l'aplicació falla degut a un error.



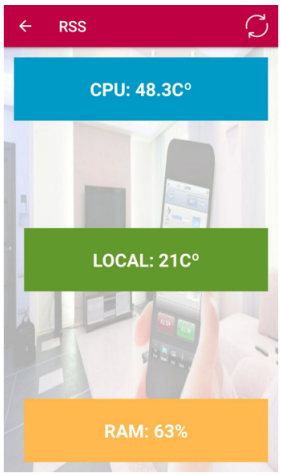
Això ens ha donat certs problemes ja que no sabíem que succeïa quan l'aplicació es tancava inesperadament. Després d'una cerca exhaustiva, ens vam adonar que l'aplicació només fallava quan la connexió també ho feia. Així, vam comprendre el funcionament real d'Android en quan a gestió de processos i vam poder determinar el que feia falta per a fer la connexió amb garanties de fiabilitat, un thread per a generar la connexió.

A partir d'aquest moment, qualsevol petició que requereixi connexió a internet la farem en un thread apart.

Afegint consistencia al projecte, cada cop que es fa un accés a l'Activity de l'enllumenat, es pregunta al servidor si hi ha algun llum encès, ja que hem de tindre en compte que des de la web també es poden encendre els llums.

Per determinar l'estat actual de les llums, es fa una petició al servidor i aquest ens indica si la llum està encesa o no. Si ho està canviem el fons del botó a verd, sinó a gris. En el cas de voler actualitzar la vista manualment, hi ha l'opció d'actualitzar a partir d'un botó per a tal fi.

4.5 Informació d'estat (Raspberry)



Com a funcionalitat addicional, hem implementat un apartat d'informació sobre l'estat del nostre servidor. Simplement és una Activity que mostra tres informacions: la quantitat de RAM utilitzada en aquell moment, la temperatura de la CPU, i extraordinàriament la temperatura local. El funcionament és similar als casos anteriors, es fa una connexió SSH a la Raspberry Pi que ens torna els valors demanats. L'string retornat es parseja i s'envía directament a l'Activity.

En un futur es podríen utilitzar aquestes dades per tal de prevenir funcionaments inesperats del sistema, enviant un mail o una notificació a l'usuari en cas de rebre uns valors inadecuats. En el cas de voler actualitzar la vista manualment, hi ha l'opció d'actualitzar a partir d'un botó per a tal fi.

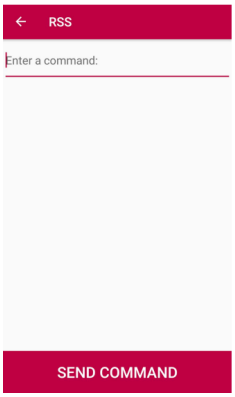
4.6 Gestor d'horaris

Per tal de poder escollir l'horari de funcionament de l'alarma de moviment, l'usuari ha de poder seleccionar dos franges horàries, la primera determina l'hora en que el sensor serà activat, y la segona determina quan aquest es para. Tot aquest funcionament queda reflexat en una sola Activity, amb opció també, d'activar o desactivar l'alarma manualment.

El funcionament del widget l'hem implementat utilitzant un TimePicker d'Android, aquest permet seleccionar horaris que a nosaltres ens serviràn per determinar el funcionament del sistema de seguretat. La manera de funcionar és simple: quan l'usuari escull un horari, depenent de la opció que triï, si franja inicial o final, enviem una comanda al servidor que fa executar un script o un altre depenent dels paràmetres passats. Això es veu reflectit en el cron de l'usuari, que és el que controla el sistema més internament.

4.7 Enviar comandes

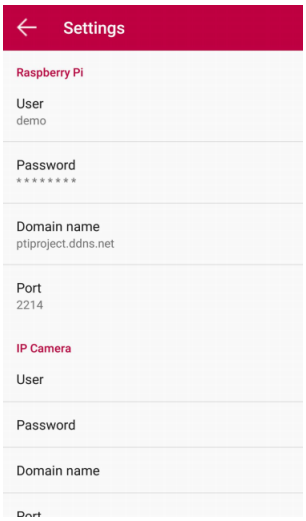
Opcionalment, per a aquest projecte, hem optat per introduir una Activity en la qual podrem enviar comandes directament a la Raspberry Pi. Per a un producte final, si l'usuari ho desitja tindrà aquesta opció, però tampoc és recomanable ja que usuaris inexperts podrien desconfigurar el servidor.



El funcionament és exactament el mencionat anteriorment per a les altres vistes, l'usuari/developer escriu una comanda, i la aplicació fa que s'executi en el servidor mitjançant el protocol d'SSH.

4.8 Preferències

La primera vegada que s'inicia l'aplicació, es mostra un login en el qual l'usuari s'identifica per tal d'accedir al servidor, si per alguna raó l'usuari volgués modificar la contrasenya o el seu nom d'usuari ho podria fer a través d'aquesta Activity. En el nostre cas, com hem de canviar la direcció IP depenent d'on està situada la Raspberry Pi, hem optat per incrementar el nombre d'opcions que l'usuari pot modificar, com el port i el domini. Aquesta funcionalitat de preferències la aconseguim amb la classe SharedPreferences d'Android.



5 Web

La web del nostre projecte està montada directament en el servidor (en aquest cas, la pròpia Raspberry), pel que totes les peticions que es facin seran bastant més ràpides que les que es facin des de l'aplicació d'Android, que ha d'accedir-hi remotament amb SSH.

Per altre banda, puntualitzar també que el fet de que la web estigui feta en el propi servidor que està montat en la mateixa Raspberry (es el punt central de tot el nostre sistema), fa que la nostra informació no viatgi a terceres parts, lo qual ho fa tot més segur i independent. Tots els scripts executats a la web (i a la app també) estàn locals a la Raspberry també.

Per últim cal recordar que tenim instal·lats tres servidors diferents: Apache, Nginx i Lighttpd. A l'hora de configurar-los ens vam encarregar de que tots reconeguessin com a pàgina inicial la que estigui a /var/www/ i es digui index.html, index.htm o index.php (ens vam encarregar també de que cadascún dels servers suporti php), per lo qual la web es veurà i funcionarà correctament en qualsevol dels tres servidors.

5.1 Llenguatges utilitzats

Els principals llenguatges utilitzats per implementar la web són HTML, CSS i PHP per la part "dinàmica". Totes les parts visuals estàtiques com el header, el footer, la pàgina d'inici,etc. són purament HTML+CSS. En alguns casos ens hem ajudat de la utilització de Bootstrap per facilitar el disseny i el fet que certes parts siguin "responsive".

Per utilitzar Bootstrap n'hi ha prou amb incloure'l amb una CDN en comptes de descarregar-lo, ja que així ens assegurem de que quan fem alguna petició, ens respongui el servidor que estigui més aprop nostre fent així el procés més ràpid. Quan usem bootstrap és necessari incloure també JQuery.

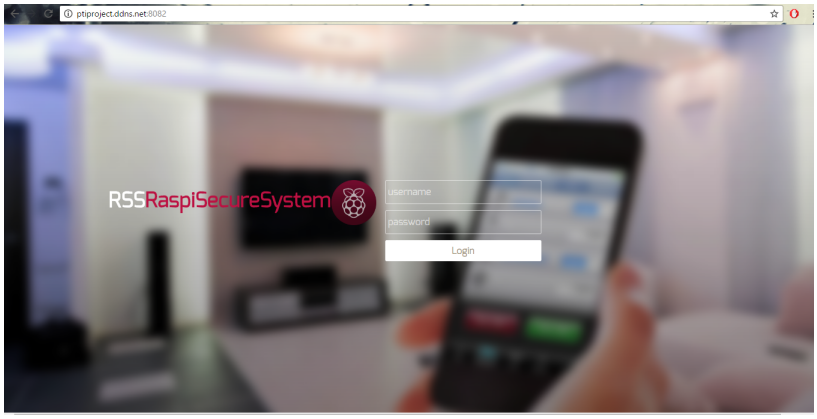
```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
```

Afegint les 3 línies de la imatge dins el ¡head¿ ja podem fer total ús de Bootstrap.

Pel que és més back-end i canvis dinàmics utilitzem PHP (el control de login, sessions, execució d'scripts, contingut dinàmic, etc.).

5.2 Parts de la web

5.2.1 Login



En la imatge podem veure un screenshot de com es veu el login de la nostra web. Aquesta pàgina es veurà la primera sempre i quan no hi hagi cap sessió definida de cap usuari: en el moment que algú hagi fet login, mentre no faci logout o tanqui el navegador, accedirà directament al home de la web. En les següents imatges es pot veure el tros de codi on es crea la variable de sessió i el tros de codi que fa el redireccionament depenent de si aquesta existeix o no.

```php
<?php
    session_start();
    if(isset($_SESSION['user'])){
        include "inici.php";
    }
    else{
        include "indexhtml.php";
    }
?>
```

```php
session_start();
if($_SERVER["REQUEST_METHOD"] == "POST"){
    if($_POST['user'] == $user && $_POST['password'] == $pass){
        $_SESSION['user'] = "currentUser";
        header("Location: inici.php");
    }
```

Evidentment hi ha control de camps requerits i control d'errors en cas que l'usuari o el password siguin incorrectes.

5.2.2 Home

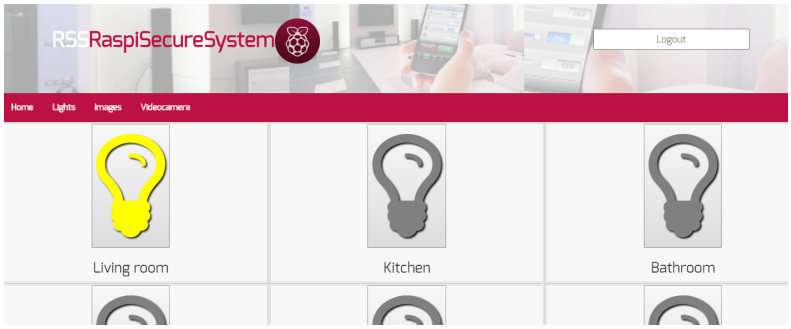
En la imatge podem veure la pinta que fa el nostre home. Veiem que consta d'un header amb la mateixa imatge i logo del login i una barra de navegació. També hi ha un footer amb el nostre logo. Aquests dos elements apareixeràn per igual en totes les pantalles de la nostra web (gràcies a php).

El text del body és simplement un Lorem Ipsum (generador de text aleatori), però en la versió "oficial" hi hauria una espècie de carta de presentació o indicacions bàsiques del nostre sistema. No dóna cap servei en sí més que informació la pàgina inicial.

5.2.3 Lights

A la imatge podem veure la pantalla de l'apartat "Lights" de la nostra web i des d'on podrem apagar i encendre llums. Em fet un grid amb varies icones de bombetes, on cadascuna representa una estància diferent de la casa. En el nostre cas només és funcional la primera icona d'adalt a l'esquerra (Living Room), però de forma aplicada podriem apagar i encendre qualsevol llum de qualsevol de les estàncies mostrades en aquesta pàgina.

Inicialment, totes apareixen en color gris ja que representa que estan tots els llums apagats, però si cliquem la icona de Living Room, veurem que canvia de color groc. Significa que el llum està encés (a la pràctica, el llum que tenim enganxat a la nostra caixa s'encén).



Hem de recordar que des de l'aplicació d'Android també és possible encendre i apagar llums, per lo qual pot haver inconsistències en aquest sistema si no es controlen.

En el nostre cas, si estem a la web i des de l'app s'han encès els llums, si fem un refresh de la pàgina veurem que l'icona ara surt de color groc, i el mateix en el cas d'apagar-les però en color gris. Per tant només obrir la pàgina veurem quines llums estan enceses i quines no.

Per fer tot aquest procés, executem el següent codi:

```
<form class="col-sm-4"  action="script.php" method="post">
    <?php
        $path = "/home/demo/lights.state";
        $fp = fopen($path, "r");
        $state =  fread($fp, 2);
        fclose($fp);
        if($state == "ON"){ ?>
            <button name="hola" id="bulb" class="fa fa-lightbulb-o" style="font-size:200px;color:yellow;text-shadow:2px 2px 4px #000000;"></button>
    <?php }
        else{ ?>
        <button name="hola" id="bulb" class="fa fa-lightbulb-o" style="font-size:200px;color:grey;text-shadow:2px 2px 4px #000000;"></button>
    <?php } ?>
```

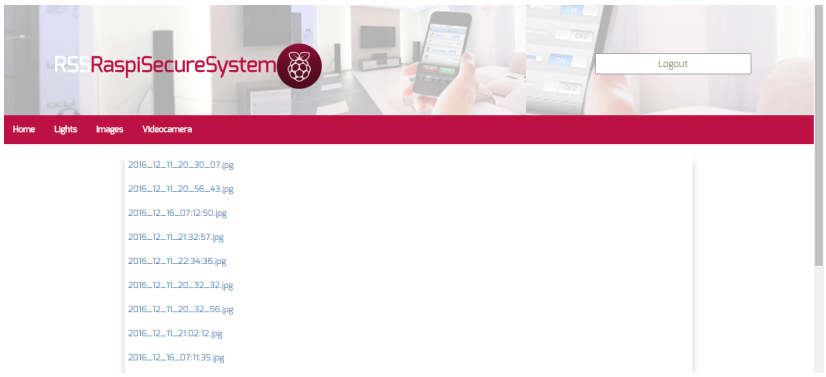
Com podem veure és una barreja entre html i php, ja que és un contingut dinàmic que ha d'anar canviant depenent de certes condicions. La icona es generarà a la pàgina d'un color o altre depenent de si dins del fitxer lights.state del nostre sistema hi trobem un "ON" o un "OFF". Com que això es fa cada cop al refrescar la pàgina, sempre tindrem el color de la icona que toca.

L'script que encèn les llums com a tal i li dóna valor al fitxer light.state s'executa només en quan cliquem la icona, i que com podem veure a l'atribut action del form es tracta de "script.php". El contingut rellevant de script.php és el següent:

```
if(isset($_POST['hola'])){
        exec("/home/demo/lights.sh");
  header("Location: llums.php");
}
```

Bàsicament executem amb la comanda l'script real del sistema ("lights.sh") que s'encarrega de tot el gestionament de les llums.

5.2.4 Images

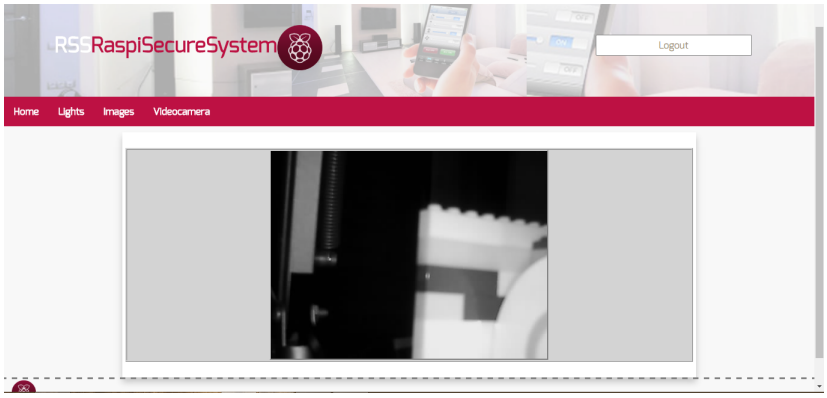


Això és el que veiem quan entrem a l'apartat de Images. Aquí és on podrem veure un llistat de totes les fotos que la videocàmera ha fet en quan ha captat algún moviment des del sensor de moviment. El contingut també és dinàmic, ja que en quan es fa una foto nova aquesta s'afegeix a la llista.

És important remarcar el detall del format del nom de les imatges, i és que per a que a l'usuari li sigui fàcil distingir quina és la foto més recent aquestes estan nombrades amb la data i l'hora en les que s'han fet. Evidentment en quan es clica qualsevol dels elements de la llista, en podrem veure la foto. En la següent imatge podem veure la part rellevant de codi que s'encarrega de fer aquesta feina:

```
<?php   $directorio = opendir("./images");
    while ($archivo = readdir($directorio)){
        if(!is_dir($archivo)){
            $path = "./images/";
            $path .= $archivo; ?>
            <a href=" <?php echo $path; ?>"> <?php echo $archivo ?></a><br><br>
        <?php }
    }
```

5.2.5 Videocamera



Per últim a la web, trobem l'apartat de la videocamera. Bàsicament mostra un <iframe>amb el contingut en streaming del que està veient la videocàmera que tenim connectada a la Raspberry. Aquesta càmera com a tal és una càmera IP que està configurada tal i com es descriu als primers apartats d'aquesta memòria. Per poder mostrar-ne el contingut basta amb fer la peticio corresponent al seu domini i port directament des d'html dins del ¡iframe¿.

Referències

[1] Pasos bàsics per preparar la Raspi,
http://www.electroensaimada.com/

[2] Configurar l'enviament de mails,
http://www.raspberry-projects.com/pi/category/software_utilities

[3] Configurar Google Drive,
https://www.howtoforge.com/tutorial/how-to-access-google-drive-from-linux-gdrive/

[4] Trebalalr amb els PID's de processos,
http://serverfault.com/questions/205498/how-to-get-pid-of-just-started-process

[5] Comandes càmera IP,
http://forums.dlink.com/index.php?topic=59172.0

[6] Treballar amb cron,
http://stackoverflow.com/questions/878600/how-to-create-cronjob-using-bash

[7] Esquema sensor de moviment,
http://www.luisllamas.es

[8] Esquema sensor de temperatura,
http://www.prometec.net/sensores-dht11/

[9] Esquema relé,
https://www.hell-desk.com/controla-bombilla-android-arduino-yun/

[10] Instal·lar un servidor lightweight,
http://xmodulo.com/lightweight-web-server-raspberry-pi.html

[11] W3 Schools,
http://www.w3schools.com/

[12] Instal·lar un servidor Apache,
https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md