

Projecte Residència

integració Sistemes

ADRIÀ SALVANS

DANIEL TRIAS

OSCAR VENDRELL

ABEL BALDELOMAR

January 23, 2019



Contents

1	Introducció	2
2	Estructura del projecte	3
3	Infraestructura	3
3.1	Xarxa de comunicació	3
3.1.1	Mode debug	5
3.1.2	Mode producte final	5
3.2	Dispositiu Detector de Caigudes	5
3.2.1	Prototipatge	5
3.2.2	Detecció de potències	8
3.2.3	Algorisme d'escaneig WIFI	10
3.3	Webserver	18
3.3.1	Web	18
3.3.2	Model-Vista-Controlador	19
3.3.3	Eloquent ORM	19
3.3.4	Middleware	19
3.3.5	Seguretat	19
3.3.6	Algorisme Localitzador	20
3.4	Websockets	24
3.4.1	Gestió de caigudes	24
3.5	Telegram	25
3.6	Base de Dades	26
4	Conclusions	28
5	Bibliografia	29
6	Annex Codi font	29

1 Introducció

En aquest projecte ens plantegem la gestió, detecció i localització de caigudes en una residència d'avís.

En el projecte anterior ja vam poder detectar la caiguda satisfactoriament, i vam poder enviar una alarma convenientment. Actualment ens trobem amb que la plataforma Arduino Uno la qual fèiem servir en el desenvolupament del primer algoritme ja no ens serveix, ja que s'ha efectuat una evolució del projecte, aquesta evolució ha esdevingut amb un primer prototip, placa de circuit imprès, la qual ja té una estructura més compacte i més adient a les nostres necessitats. A partir d'ara aquesta placa serà la nostra plataforma per desenvolupar l'algoritme de detecció de persones.

En aquest moment, podem deixar la caiguda d'avís per superada, i ens centrarem en la detecció del posicionament de les persones un cop s'hagi detectat una caiguda, o un cop aquesta persona hagi premut el boto de pànic.

Per la correcte simulació de l'algoritme ens plantegem la universitat com si sigues una simulació de residència.

En primera instància, ens plantegem la detecció de la persona aprofitant el mateix dispositiu de comunicació Wi-fi del dispositiu detector de caigudes, ja que tractarem aquest com a dispositiu personal, i per tant podem detectar la persona que tingui la urgència pertinent.

Com que el dispositiu és personal, en primera instància també ens plantegem la creació de una base de dades la qual ens permeti identificar quin dispositiu porta cada persona de la residència.

La web està orientada a la gestió de la residència (treballadors, residents...) i a l'avís i gestió de possibles caigudes de residents.

2 Estructura del projecte

En la següent taula si detallen els diferents fitxers rellevants i esmentats en la memòria i on estan organitzats dins del codi font.

Recurs	Directori
Models	App/
Controladors	App/Http/Controllers/
Vistes	Resources/views/
Middlewares	App/Http/Middleware/
Gates	App/Providers/
Sockets	App/Socket i comanda: app/Console/Commands
Taules de la BD	Database/migrations/
Biblioteca BD ssids_potències	Database/migrations/seeds/Position_Ssid TableSeeder.php
Rutes i API	Routes/

Figure 1

Pel que fa als fitxers d'arduino, hem utilitzat els següents:

- ADXL_ESP01.ino: Es el programa principal d'arduino. Dins hi ha tot el funcionament del ADXL i l'algoritme de detecció de wifis juntament amb la connexió al server.
- Splitter.cpp: Es un modul que hem creat, on introduim un línia del ESP01 i ens ailla el nom de la wifi i la seva potència. Retorna a una estructura de dades on tenim la potència i el nom en diferents variables.
- FSM_Posicio.cpp: Aquí tenim el algoritme del projecte de l'any passat, el algoritme de detecció de caigudes.
- tmr0.cpp: Es un timer que es necessita per el algoritme de detecció de caigudes.

3 Infraestructura

En aquesta imatge es representa l'esquema de funcionament del projecte. La placa (collaret) quan detecta una caiguda fa un mostreig dels diferents wifi's amb les potències que rep i les envia amb un post al webserver fet amb Laravel. En el webserver s'executa l'algoritme de localització i un cop trobada la posició on el resident ha caigut s'envia l'avís amb la localització al canal de Telegram. Cada treballador de la residència té accés a la web de gestió. Al connectar-si obren connexió de websocket amb el servidor. Un cop el webserver ha detectat la localització de la caiguda, a part d'enviar-la per telegram, l'envia a tots els clients web via websockets.

3.1 Xarxa de comunicació

Per el desenvolupament i tasteig del projecte s'ha configurat una xarxa en mode debug, però també s'ha plantejat la configuració de la xarxa pel producte final, un cop implementat el projecte en la residència.

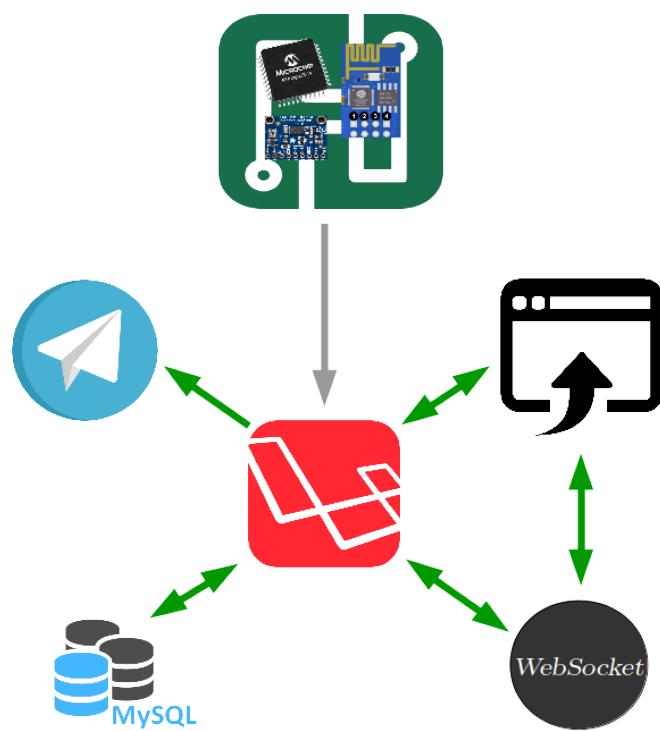


Figure 2

Apart de la pròpia xarxa de comunicació entre els dispositius, en els dos modes es reparteixen punts d'accés per tot l'edifici amb ssids pactats (AP_1,...,AP_N) amb la finalitat de poder fer la geolocalització.

3.1.1 Mode debug

Es crea una xarxa privada (local) de la qual s'hi connecta el collaret, el webserver i els websockets. No hi ha sortida a internet.

3.1.2 Mode producte final

A la residència es crea una xarxa privada on s'hi connecten tots els collarets i la màquina encarregada d'allotjar el webserver i els websockets. A més, aquesta màquina té una interfície amb sortida a internet. D'aquesta manera la web pot ser consultada sense estar connectat a la xarxa privada.

3.2 Dispositiu Detector de Caigudes

3.2.1 Prototipatge

Per fer el nostre aparell més funcional, hem trobat necessari fer un disseny de un recipient per contenir-lo i que aquest el pugui dur la persona corresponent.

D'aquesta forma hem ideat un penjoll en forma de pinça, el qual està pensat per anar penjat de una butxaca a la vista o amagat sota la roba, o bé per portar-lo a la butxaca.

Amb el disseny de la sivella ens volem assegurar que el dispositiu pugui estar ancorat adequadament en la roba del portador i no pugui saltar ni pendular de forma aleatòria. D'aquesta forma es busca una màxima subjecció entre element detector i element a detectar.

El disseny resultant del nostre dispositiu és el següent:

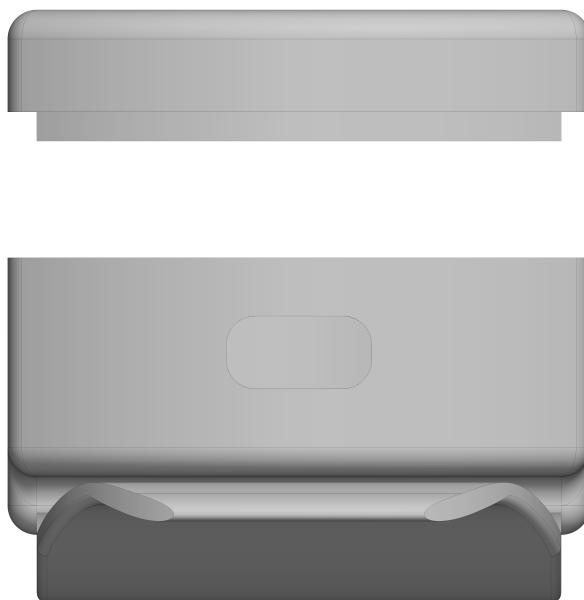


Figure 3: Alçat 1

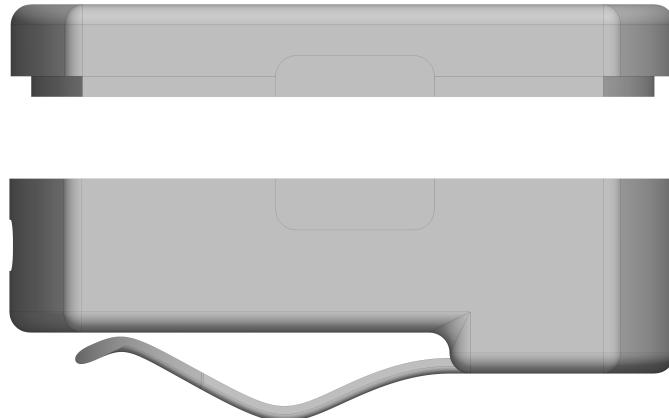


Figure 4: Alçat 2



Figure 5: Axonometria 1

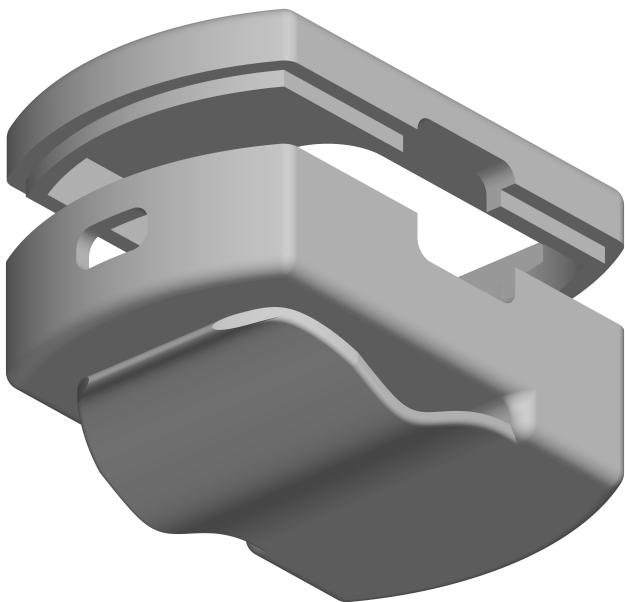


Figure 6: Axonometria 2

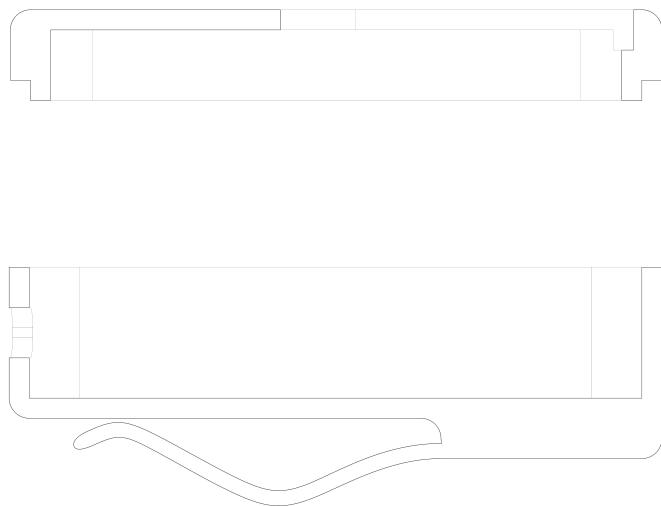


Figure 7: Secció

3.2.2 Detecció de potències

En aquest apartat exposem l'estudi sobre la detecció de potències que hem efectuat. Per fer-ho hem fet un mostreig de les potències, en concret 10 mostrejos en un mateix punt, i comparem el valor que ens dona un respecte l'altre per poder extreure una mitjana d'aquestes i una desviació estàndar.

Les potències rebudes son les següents:

Escanejos	Potencia1	Potencia2	Potencia3	Potencia4	Potencia5	Potencia6	Potencia7
1	0	70	75	89	86	74	50
2	0	69	68	86	77	75	50
3	79	72	70	92	80	70	54
4	74	64	62	84	82	73	51
5	0	74	64	88	82	75	52
6	78	66	65	86	84	73	51
7	78	68	66	87	84	75	52
8	77	66	64	86	88	75	53
9	76	67	67	86	87	73	52
10	76	71	67	85	88	76	54

Escanejos	Potencia1	Potencia2	Potencia3	Potencia4	Potencia5	Potencia6	Potencia7
Mitjanes de 10	53,8	68,7	66,8	86,9	83,8	73,9	51,9
n.º detecció potènica	7	10	10	10	10	10	10
Mitjanes Individuals	76,857142857	68,7	66,8	86,9	83,8	73,9	51,9

Figure 8: Taula de potències detectades

A partir de l'anàlisi d'aquestes potències, podem veure que els valors de lectura de potències, no són precisos de un escaneig a un altre, d'aquesta forma, per tal d'estabilitzar les dades, busquem fer una mitjana per poder-les estabilitzar en un valor que varii el menys possible.

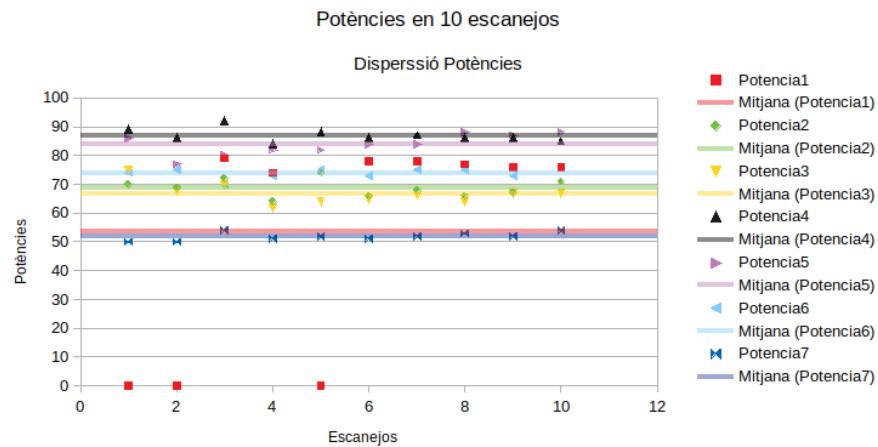


Figure 9: Grafica amb les potències detectades i la mitjana teòrica amb 10 escanejos

També podem veure en la taula de potències, hi ha vegades que la potència no és detectada, i per tant ens quedem amb un valor de 0. Aquest és un valor molt dolent per nosaltres, ja que ens augmenta la variància i per tant el valor de la desviació estàndar. A més a més, si fem una mitjana amb tres 0 detectats, podem apreciar que aquesta es queda baixa. El que hauríem de fer per poder detectar bé la potència és calcular una mitjana per cada Potència separadament. Per tant si la potència 1 només és detectada 7 cops, la mitjana que s'ha de fer per aquesta potència no és sobre 10, sinó sobre 7. Ja que d'aquesta forma ens assegurem fer la mitjana amb potències existents.

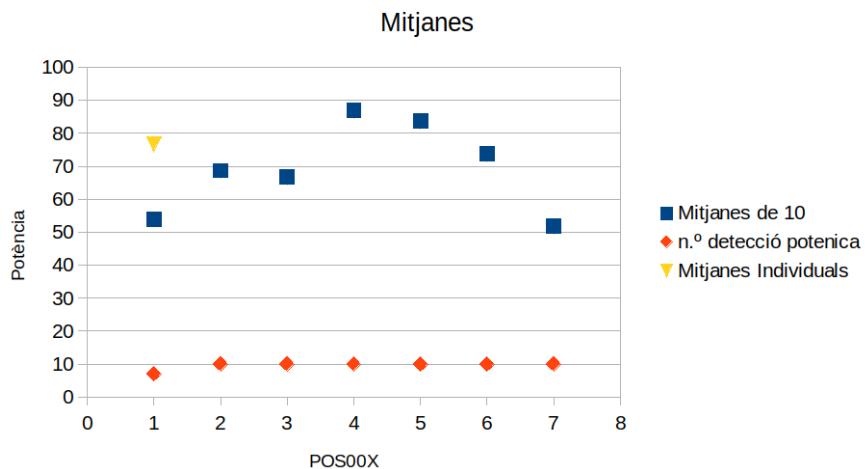


Figure 10: Gràfica de les mitjanes calculades per cada potència

3.2.3 Algorisme d'escaneig WIFI

Ens plantegem fer un algoritme que permeti escanejar totes les Wi-fi de una zona. El que ens interessa d'aquestes Wi-fi és la potència amb que les rebem. Però no ens a serveix qualsevol Wi-fi, han de ser punts d'accés que tinguem controlats amb ubicació i que tinguem un control sobre aquests. En concret les Wi-fi que nosaltres volem detectar son del tipus POS00X, on X és un nombre el qual ens marca el punt d'accés.

En si aquests punts d'accés es troben distribuïts en l'edifici de la següent forma:

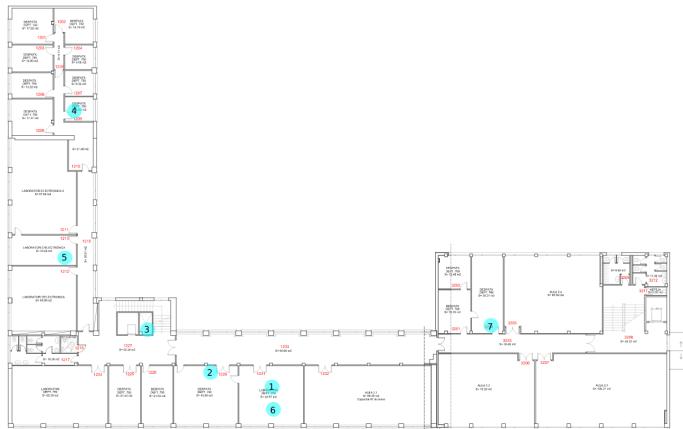


Figure 11: Localització dels punt d'accés en l'edifici

3.2.3.1 Primeres proves.

La potència de les Wi-fi les detectem a partir d'enviar la instrucció AT+CWLAP per el Serial1, on es troba el modul ESP. Un cop hem enviat la comanda, el mòdol ESP comença a bolcar les dades en el mateix Serial1. A partir d'aquest punt el serial ens retorna el seguent:

```

1 +CWLAP:(3,"POS001",-71,"f8:e4:fb:5b:a9:5a",1)
2 +CWLAP:(3,"POS006",-90,"04:f0:21:0f:1f:61",1)
3 +CWLAP:(3,"CLDRM",-69,"22:c9:d0:1a:f6:54",1)
4 +CWLAP:(2,"AllSaints",-88,"c4:01:7c:3b:08:48",1)
5 +CWLAP:(0,"POS005",-83,"c4:01:7c:7b:08:48",1)
6 +CWLAP:(0,"AllSaints-Guest",-83,"c4:01:7c:7b:05:08",6)
7 +CWLAP:(4,"POS002",-27,"e8:94:f6:90:f9:d7",6)
8 +CWLAP:(2,"AllSaints",-82,"c4:01:7c:3b:05:08",6)
9 +CWLAP:(3,"QGJTL",-87,"f8:e4:fb:b5:6b:b4",6)
10 +CWLAP:(4,"50EFA8",-78,"74:44:01:50:ef:a7",6)
11 +CWLAP:(0,"POS004",-78,"76:44:01:50:ef:a8",6)
12 +CWLAP:(3,"POS003",-95,"18:1b:eb:1a:af:5b",6)
13 +CWLAP:(3,"NETGEAR49",-86,"84:1b:5e:e0:28:03",7)
14 +CWLAP:(3,"POS007",-56,"20:e5:2a:79:b1:2f",11)
15 +CWLAP:(3,"BFZR4",-73,"18:1b:eb:1d:c3:91",11)
16 +CWLAP:(1,"5FFVL",-82,"00:26:b8:b5:c0:f2",11)
17 +CWLAP:(3,"POS004",-77,"00:7f:28:6d:91:7b",11)
18 +CWLAP:(3,"N16FU",-53,"20:cf:30:ce:60:fe",11)
19 +CWLAP:(3,"ITS",-82,"90:72:40:21:5f:76",11)
20 +CWLAP:(3,"ITS",-79,"24:a2:e1:f0:04:e4",11)
21
22 OK

```

Listing 1: "mostrawifis.c"

El que intentem fer és detectar el símbol OK per acabar el While de detecció i per tant poder emmagatzemar totes les dades en una array per posteriorment tractar-les.

3.2.3.2 Versió 1

Al principi per poder crear la biblioteca de posicions Wi-fi, vam estar pensant com poder fer varis escanejos per calcular la mitjana i fer-ho de la manera mes optima i rapida possible. En la primera versió d'aquest algoritme, se'n va ocórrer la idea de emegatzemar totes les dades en una mateixa array, per poder-la analitzar tota posteriorment amb una nova funció anomenada splitter. Aquesta funció, aïlla el nom de la wifi i la potència i ho emmagatzemava en una variable. Un cop teníem el nom de la Wi-fi ho filtràvem per nom, ja que a nosaltres només ens interessen les Wi-fi del tipus "POS00X". Un cop teníem el filtre fet, creàvem un array de ssids i un array de potències. Amb aquests dos arrays anàvem emmagatzemant el nom i la potència per finalment enviar-la al servidor. Quan una wifi es detectava un segon o tercer i x cop, automàticament amb una funció que vam implementar, mirava si la Wi-fi estava dins, i si estava dins només sumava la potència a la llista de potències, i sinó ho trobava afegia el nom a la llista de noms i la potència la llista de potències. Ho vam intentar de totes les maneres per poder fer-ho així ja que el algorisme hagues estat molt mes ràpid, però ens vam topar amb molts problemes amb la placa i finalment vam canviar el punt de vista creant la versió 2 que no és tant eficient però funciona correctament i aconsegueix el nostre objectiu.

```

1 void loop() {
2     //si globals no funciona
3     char recep_potencies[100]{};
4     bool comanda=false;
5     int i=0;
6     int j=0;
7     char ssid[500]{};
8     int pot[100]{};
9     int caracter_wifi=0;
10    bool trobada=false;
11
12    //enviament de codi per llegir potència wifis

```

```

13
14 if (Serial.available())
15 {
16   if (Serial.read()=='S'){
17     for (j=0;j<=9;j++)//Fem 10 escanejos de la mateixa posicio per calcular una
18     mitjana correcte
19     {
20       delay(100);
21       digitalWrite(led,true);
22       Serial1.write("AT+CWLAP\r\n"); //escrivim la comanda per buscar les wifis
23       comanda=false;
24       while(!flag_final){//mentre no truem un OK no sortim del while
25         if(Serial1.available()){
26           char caracter_actual=Serial1.read(); //anem llegint els caracters del ESP
27           maquina_estats(caracter_actual); //es una maquina de estats que esper el OK
28           per canviar el "flag_final" i indicar que hem trobat el "OK"
29           if (caracter_actual>=40 && caracter_actual<=122){//Filtrem els caracters que
30             no ens interesa i els deixem fora
31             if(caracter_actual=='+'{
32
33               if (!comanda)//com que entrem aqui quan trobem un "+" sabem que el
34               primer "+" es de la comanda AT+CWJAP
35               {
36                 comanda=true;
37               }
38             else
39             {
40
41               auto data = split(recep_potencies); //agafem la linia del ESP
42               "+CWLAP:(3,"POS001",-71,"f8:e4:fb:5b:a9:5a",1)" i aillem el nom i la potencia i
43               la posem a una estructura de dades. D'aquesta manera podem tractar el nom i la
44               potencia per separat
45               if(data.nom[0]=='P' && data.nom[1]=='0' && data.nom[2]=='S')//filtrem
46               per nom
47               {
48                 int
49                 potencia=potencia=((int)data.potencia[1]-48)*10+((int)data.potencia[2]-48); //convertim
50                 la potencia rebuda de char a int
51                 if(j==0)//primer escaneig
52                 {
53                   afegeix(data.nom,ssid); //afegeim la potencia detectada a la llista
54                   ssid
55                   afegeixpot(potencia,pot); //afegeim la potencia detectada a la
56                   mateixa posicio que hem collocat el ssid per a la llista pot
57                 }
58                 else//seguent escaneigos
59                 {
60                   trovada=hi_ets(data.nom,ssid); //mirem si la wifi la hem detectat
61                   anteriorment
62                   if (trovada)
63                   {
64                     suma(ssid,pot,data.nom,potencia); //si l'hem trobat sumem la
65                     potencia que teniem amb la potencia que hem rebut ara
66                   }
67                   else
68                   {
69                     afegeix(data.nom,ssid); //sino hi era afegeim el nom de la wifi
70                     a la llista de ssid
71                     afegeixpot(potencia,pot); //afegeim la potencia a la llista de
72                     pot
73                   }
74                 }
75               }
76               buida_cadena(recep_potencies); //buidem la variable que fem servir per
77               emmagatzemar cada linia que rebem del ESP
78               i=0;
79             }
80           }
81         }
82       }

```

```

64         i++;
65         recep_potencies[i]=caracter_actual;
66     }
67   }
68 }
69 flag_final=false;
70 }
71 }
72 }
73 else
74 {
75   digitalWrite(led, false);
76 }

```

Listing 2: "wifiver1.c"

3.2.3.3 Versió 2

Per a poder aconseguir una biblioteca de posicions de caigudes ben completa i fiable, hem decidit crear un programa on faci 10 escanejos per posició i envii a traves d'un missatge POST al servidor les potencies detectades i el nom de les Wi-fi detectades. Aquests 10 escanejos, els fem per que hem detectat amb proves anteriors que les potencies tendeixen a tindre una certa desviació a l'hora de detectar-les. Aquest punt es tractarà millor en l'apartat "detecció de potencies". La potencia de les Wi-fi les detectem a partir d'enviar la instrucció AT+CWLAP per el Serial1, on es troba el modul ESP. Un cop hem enviat la comanda, el mòdol ESP comença a bolcar les dades en el mateix Serial1. Nosaltres recollim aquestes dades amb el nostre algoritme i les emmagatzemam en un array. Aquesta vegada hem redut aquest array significativament, a només 80 posicions, ja que la informació que necessitem emmagatzemar aquest cop és molt menor ja que ens plantegem l'algoritme de forma diferent. El que farem serà anar acumulant dades en l'array, les dades que arriben per el port son del tipus:

- +CWLAP:(3,"CVBJB",-71,"f8:e4:fb:5b:a9:5a",1)

Com podem observar cada línia comença amb un símbol +. Per tant podem detectar que l'array s'ha omplert quant detectem el següent simbol +. En aquest punt és quant processem l'array anterior. A partir de la funció feta en la versió 1 split, agafem la SSID de la Wi-fi i la potencia i les posem en 2 variables diferents. En aquest punt buidarem l'array per que es pugui tornar a omplir en la següent volta de l'algoritme.

Un cop hem buidat, seguim fent una comparació del nom de la SSID, si la SSID és una POS00X, filtrem com em dit abans amb unes comparacions del nom. Per a cada Wi-fi detectada fem una variable per emmagatzemar la potencia detectada. D'aquesta manera per cada escaneig si troba un altre cop la Wi-fi, sumarem la potencia corresponent a la variable de potencia. Per poder calcular la mitjana de cada variable, se li ha afegit un comptador a cada condicional per saber quants cops s'ha detectat aquella Wi-fi.

Un cop han acabat els 10 escanejos, enviarem per un missatge "POST" al servidor el nom de punt d'accés i la seva potència d'aquesta manera: AP_X,POT; on X= numero de punt d'accés i POT= la potència mitjana d'aquell punt d'accés. En el missatge, a més a més enviarem l'identificador d'aquell detector de placa per saber quién és el collar que ha enviat l'alarma. Un cop escanejat i enviat el missatge el servidor ho interpretara i aplicara el algorisme per saber on s'ha detectat l'alarma.

```

1 while(vegades<10){//fem 10 esanegos per tenir una millor mitjana
2   vegades++;
3   Serial.print(F("veg: "));
4   Serial.println(vegades);
5   lectura_wifi_simple();
6   int posicio=0;
7   while(!flag_final){
8     if(Serial1.available()){
9       char caracter_actual=Serial1.read();
10      flag_final=check_ok(caracter_actual,flag_final); //esperem un "OK" per quan
hagi acabat el escaneig

```



```

61         Serial.println(Potencia_total);
62         Serial.println(potencia6);
63         potencia6=potencia6+Potencia_total;
64         Serial.print("pot6: ");
65         Serial.println(potencia6);
66         contador6++;
67     }
68     else if(data.nom[0]==wifi7[0] && data.nom[1]==wifi7[1] &&
69     data.nom[2]==wifi7[2] && data.nom[3]==wifi7[3] && data.nom[4]==wifi7[4]&&
70     data.nom[5]==wifi7[5]){
71         Serial.println(Potencia_total);
72         Serial.println(potencia7);
73         potencia7=potencia7+Potencia_total;
74         Serial.print("pot7: ");
75         Serial.println(potencia7);
76         contador7++;
77     }
78     else{
79         posicio++;
80         recep_potencies[posicio]=caracter_actual;
81     }
82 }
83 }
84 }
85
86 flag_final=false;
87 }
88 potencia1=potencia1/contador1;//Aqui calculem la potencia mitjana dels 10 escanegos
89 per cada wifi
90 potencia2=potencia2/contador2;
91 potencia3=potencia3/contador3;
92 potencia4=potencia4/contador4;
93 potencia5=potencia5/contador5;
94 potencia6=potencia6/contador6;
95 potencia7=potencia7/contador7;
96 if (potencia1== -1){//si les wifis no han sigut detectades, hi podem un 95
97     potencia1=95;
98 }
99 if (potencia2== -1){
100     potencia2=95;
101 }
102 if (potencia3== -1){
103     potencia3=95;
104 }
105 if (potencia4== -1){
106     potencia4=95;
107 }
108 if (potencia5== -1){
109     potencia5=95;
110 }
111 if (potencia6== -1){
112     potencia6=95;
113 }
114 if (potencia7== -1){
115     potencia7=95;
116 }
117 String AP="";
118 for (int i=0;i<7;i++) //anem concatenant totes les potencie mitjanes per poder fer
119 el POST al servidor
120 {
121     if (i==6)
122     {
123         String c=("AP_"+(String)(i+1)+","++(String)potencies_totals[i]);
124         AP=AP+c;
125         break;

```

```
125     }
126     else
127     {
128         String c=("AP_"+(String)(i+1)+"+"+(String)potencies_totals[i]+";");
129         AP=AP+c;
130     }
131 }
132 send_string("ssids_rssis="+AP+"&idPlaca=7"); //Funcio que fa el post amb les claus
i valor que li pasem
133 }
```

Listing 3: "wifiver2.c"

3.2.3.4 Versió final

En la versió final de l'algoritme hem ajuntat el programa detector de caigudes amb el programa de localització i gestió de potències. Hem tingut algun que altre problema a l'hora de unir els dos programes, però hem aconseguit un resultat satisfactori ja que el dispositiu envia una alarma tant sigui prement el boto com per una caiguda.

Els principals problemes que ens hem trobat, han sigut a l'hora de aconseguir enviar el POST, ja que aquest ocupa molta memòria dinàmica, i amb la junció dels dos programes aquesta s'ha vist saturada. El que hem fet per arreglar-ho ha sigut eliminar tot el que no necessitavem o posar alguns debugs a la memòria Flash.

Aquest algoritme va mirant en quina posició es troba el dispositiu en tot moment, i si el boto ha estat premut. En cas que no passi res, va fent voltes i va mirant la posició. En el cas que detecti una acceleració forta, utilitza la maquina de posició per detectar si es tracta de una caiguda o de una falsa alarma, en el cas que sigui una caiguda s'executa l'algoritme de posicionament, el qual captura les potències Wi-fi POS00X del lloc on es troba la persona i les envia mitjançant un POST a la web de gestió. En el cas de que no haguem detectat cap caiguda, però l'usuari hagi premut el boto, també es crida l'algoritme de detecció de potències Wi-fi, i fa el mateix procés que en la caiguda.

En els dos casos que s'activi la alarma, s'encendrà el led del boto. Per poder detectar visualment que aquell collaret ha petit una alarma.

3.3 Webserver

El SO del servidor és un Linux Ubuntu i com a webserver s'hi ha instal·lat i configurat un Apache 2.4 ja que proporciona un bon rendiment i versatilitat. Aquest web server allotja el projecte web dels quals ha estat desenvolupat amb PHP 7.2 utilitzant Laravel, una framework basada amb php. A nivell de frontend s'utilitza Html, Css, Javascript i Bootstrap (framework d'html, css i JS).



Figure 12: Eines emprades

3.3.1 Web

La web consta d'una pàgina d'inici on hi ha l'enllaç a la pàgina d'iniciar sessió i a la pàgina de registre, en les quals s'hi troba el corresponent formulari. Un cop s'ha iniciat sessió, es redirecciona a la pàgina de gestió d'usuaris, per defecte, i a la part superior hi ha el menú principal amb les diferents funcionalitats de la pàgina web com la gestió d'horaris, de sectors, de residents...

A screenshot of a web-based application's dashboard. The top navigation bar includes links for 'Users', 'Horari', 'Sector', 'Devices', and 'Clients'. On the far right, there is a dropdown menu for 'superadmin'. Below the navigation, a section titled 'Gestió d'usuaris' shows a table of users. The table has columns for 'Dni', 'Name', 'Email', 'Rol', 'Horari', 'Sectors', 'Editar Usuari', and 'Eliminar Usuari'. There are three rows of data: 1. Dni: 434334343, Name: adriàsT, Email: adriasalvans@gmail.com, Rol: admin, Horari: green, Sectors: green, Editar Usuari: blue, Eliminar Usuari: red. 2. Dni: 48048074q, Name: dani, Email: dani@gmail.com, Rol: consumer, Horari: green, Sectors: green, Editar Usuari: blue, Eliminar Usuari: red. 3. Dni: 48048074T, Name: superadmin, Email: superadmin@gmail.com, Rol: superadmin, Horari: green, Sectors: green, Editar Usuari: blue, Eliminar Usuari: red. A 'Agregar usuari' button is visible at the top left of the table area.

Figure 13: Pàgina principal

La web consta de les següents funcionalitats:

- Gestió de caigudes: Explicat a l'apartat de web sockets.
- Gestió d'horaris: Permet crear, editar i eliminar qualsevol franja horària per tal d'organitzar els horaris dels treballadors.
- Gestió de sectors: Permet crear, editar i eliminar sectors per tal de restringir les diferents àrees de la residència.
- Gestió d'usuaris: Permet crear, editar i eliminar usuaris, afegir/eliminar franges horàries i afegir/eliminar sectors en els diferents usuaris
- Gestió dels collarets (devices): Permet crear, editar i eliminar collarets.
- Gestió dels avis: Permet crear, editar i eliminar avis.

3.3.2 Model-Vista-Controlador

S'ha treballat seguint l'esquema de MVC (Model-Vista-Controlador), dels quals s'utilitzen els models per definir les "entitats" de l'aplicació, les vistes per mostrar la interfície de l'aplicació, mostrant la informació obtinguda a partir d'un model i el controlador per tal de programar-hi tota la lògica. La Caiguda, el Client, el Collaret (Device), l' Horari, el Sector i l'Usuari són les entitats de l'aplicació que segueixen aquest esquema i que per tant tenen el seu model, el seu controlador i el seu grup de vistes.

3.3.3 Eloquent ORM

L'Eloquent ORM és una funcionalitat que s'ha aprofitat de Laravel que consisteix en que cada taula de la base de dades correspon a un Model. D'aquesta manera utilitzant les propietats dels Models es pot treballar directament amb les taules de la base de dades. Amb aquesta funcionalitat s'aprofita tota la potència de les relacions i es facilita l'ús de cerques, modificacions... de la Base de dades ja que no s'utilitza SQL.

3.3.4 Middleware

Els middleware's s'utilitzen per filtrar les peticions HTTP que entren en l'aplicació per les rutes definides tant per URI com per API. S'utilitzen middleware's per verificar que l'usuari de l'aplicació està autenticat. Si l'usuari no està autenticat el middleware redirecciona l'usuari a la pàgina d'inici de sessió i també per la protecció de CSRF (explicat a Seguretat).

3.3.5 Seguretat

Tots els formularis de la pàgina web estan protegits sota els atacs tipus CSRF (Cross-Site request forgery), tipus d'exploit maliciós en la qual un usuari "autoritzat" transmet unes comandes no autoritzades. Es verifica, que el token enviat en la "request" és igual que el token guardat en la sessió. Les contrasenyes dels usuaris es guarden totes encriptades a la BD utilitzant l'algoritme de hash.

- Autenticació: Mitjançant Guards, funcions que defineixen com els usuaris s'autentifiquen per cada petició, es fa el control de sessió, mantenint l'estat amb l'emmagatzematge de sessió i cookies.
- Autorització: Utilitzant les Gates (funcions definides a App/Providers), es defineix si un usuari està autoritzat o no a dur a terme un acció.

3.3.6 Algorisme Localitzador

La implementació de l'algorisme detector de la localització d'una caiguda s'implementa dintre del webserver.

Tot seguit, s'explica el fonament teòric de l'algorisme. L'algorisme es basa en la idea de mapejament sobre un espai a partir de potències de senyals rebudes dels punts d'accés wifi. Donat el plànol d'una planta, aquest es distribueix en sectors. En el nostre cas un sector podria correspondre a l'àrea d'una habitació. Seguidament, es mostra un exemple d'una probable planta distribuïda en sectors.

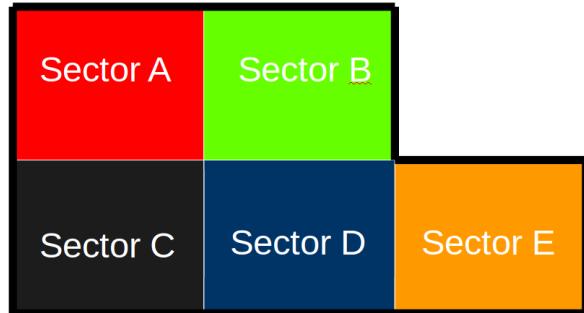


Figure 14: Estructura d'un plànol en sectors

Així mateix, cada sector està distribuït en posicions o punts mapejats.

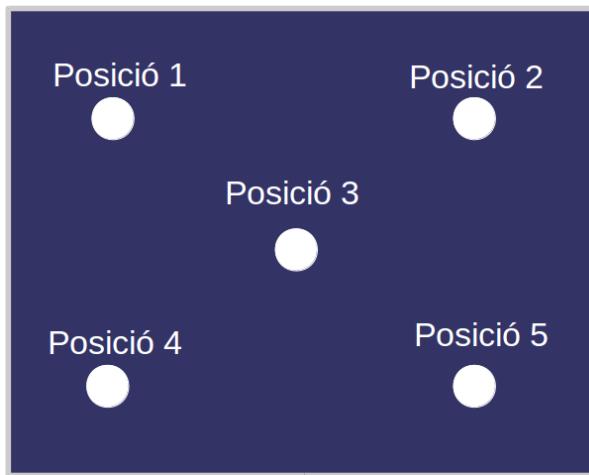


Figure 15: Sector distribuït en punts mapejats

D'aquesta forma, cada posició o punt mapejat consisteix en un llistat de punts d'accés, es a dir, un **llistat de ssids** amb la potència de cada senyal (**rssi**) corresponent al lloc.

Llista: (ssid1,rssi1), (ssid2,rssi2), ..., (ssidN,rssiN)

Figure 16: Punt mapejat

Quan una caiguda es detecta, el dispositiu transmet a l'algorisme un mostreig de les potències de cada senyal sobre el lloc de la caiguda. D'aquesta forma s'obté un llistat de (ssid,rssi) dels senyals detectats, es a dir, la mateixa estructura establerta per les posicions d'un sector. Un cop rebut la llista de (ssid,rssi) de la caiguda, aquesta es compararà amb les posicions de cada sector, amb el fi de determinar la localització del dispositiu.

La comparació consisteix en els següents punts:

1. **Grau de coincidència:** es compara el nombre de ssids comuns entre la llista de (ssid,rssi) de la caiguda i la llista de (ssid,rssi) de cada posició sobre cada sector. El grau de coincidència ha de ser igual o superior al 70%.
2. **Grau de desviació:** és la suma total del resultat de restar les **potències (rssi)** entre els ssid comuns entre la llista de caiguda i la llista de posició que s'analitza.

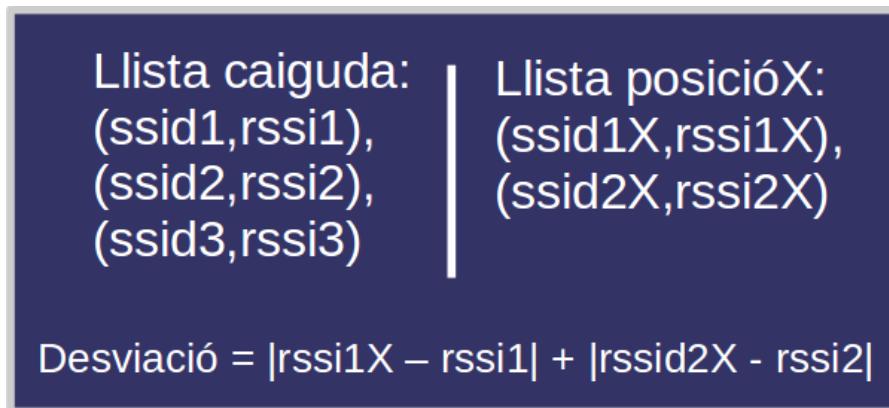
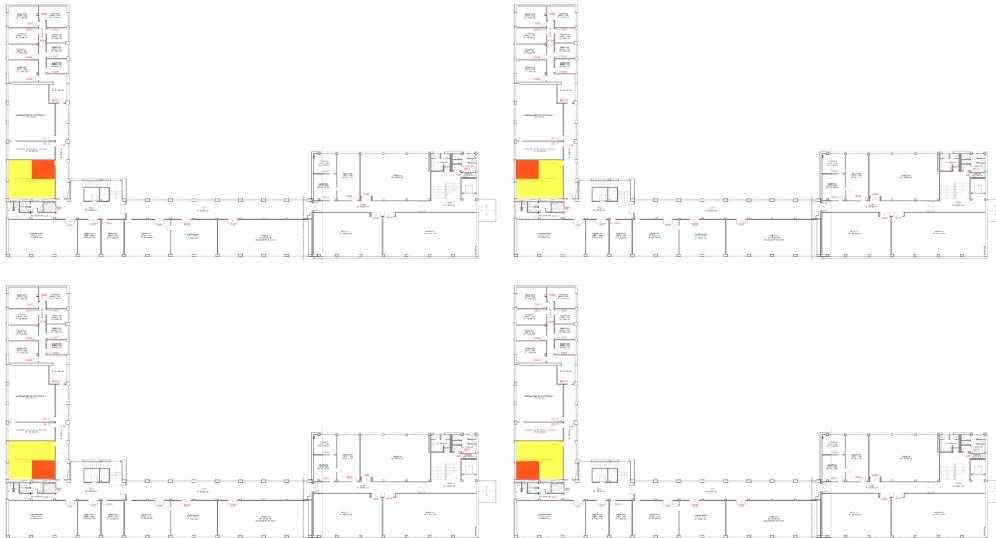


Figure 17: Càcul desviació

3. **Rànquing de sectors:** la localització del punt de caiguda es basa sobre un rànquing de sectors, aquell sector que tingui més punts determina la posició de localització. Per determinar els punts de cada sector es té en compte la desviació de les posicions que conformen un sector. Si la desviació és inferior a un **llindar establert (Threshold)**, es suma un punt al rànquing del sector corresponent. Cal recordar, que cada posició ha de superar en primer lloc el grau de coincidència i tot seguit el grau de desviació per poder ser tingut en compte en el mapeig.

3.3.6.1 Distribució sectors universitat

Per poder fer les proves pertinents de localització, hem agafat com a model la nostre universitat, aquesta la hem dividit en diferents sectors, aprofitant la distribució de l'edifici.



Com podem apreciar en les imatges superiors, hem aprofitat el que serien els mateixos noms de les habitacions per poder identificar cada sector. Podem apreciar cada sector en groc en les imatges. En aquest cas només s'emmarca amb groc una habitació per que totes les imatges pertanyen a la habitació 1214. Un cop estem dins la habitació, hem dividit aquesta en subsectors, els quals hem aprofitat l'arquitectura de columnes de l'edifici per poder-los delimitat aproximadament. Així doncs podem veure amb color vermell el subsector de l'habitació. Aquestes imatges les aprofitarem a posteriori per poder-les enviar per el telegram, d'aquesta forma podem identificar on s'ha produït la alarma amb una pressió aproximada de uns 5 metres si el subsector encerta, i amb una precisió de 10 metres aproximadament si el sector encerta.

Amb les proves que hem pogut efectuar, normalment no falla mai el sector, en aquest cas podem confirmar que el nostre algoritme detector de potències va bé per poder identificar una alarma en una habitació de l'edifici. En quant el subsector, depèn de si estàs molt a prop del del costat pot portar confusions, tot i així té una taxa d'encerts alts.

3.3.6.2 Resultat d'eficiència de l'algorisme

Tot seguit, s'observa un diagrama de l'eficiència de l'algorisme sobre les localitzacions efectuades en diversos punts dins l'àrea compressa a la planta dos de l'EPSEM. Concretament, sobre els sectors assignats com **1214, 1215, 1217, 1227, 1233**.

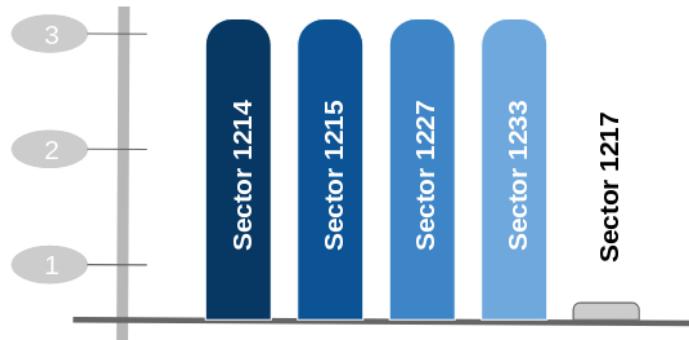


Figure 18: Diagrama de tres mostres per sector

Com s'observa en el diagrama, l'eficiència de l'algorisme és molt elevada gairabé sobre tots els sectors, exceptuant el sector **1217**. Tot així, en la següent imatge es pot observar que la determinació de la posició és inexacte pel sector **1217**, encara que, la localització resultant és molt propera a la desitjada. Aquest últim ha estat causat probablement pel soroll en la detecció de potències.



Figure 19: Localització propera per 1217

3.3.6.3 Demostració dels resultats obtinguts

En les següents enllaços es poden observar els resultats i comprobacions de l'algorisme de localització.

- [Video 1](#)
- [Video 2](#)
- [Video 3](#)

3.4 Websockets

La gestió de les caigudes s'han fet mitjançant web sockets, per tal de donar concorrència a la gestió dels avisos de les caigudes. S'ha utilitzat la llibreria Ratchet. Consta de 2 scripts; la inicialització i configuració del servidor de websockets i la pròpia aplicació de websockets dels quals hi ha les diferents funcions, per enviar i rebre dades via els websockets per exemple.

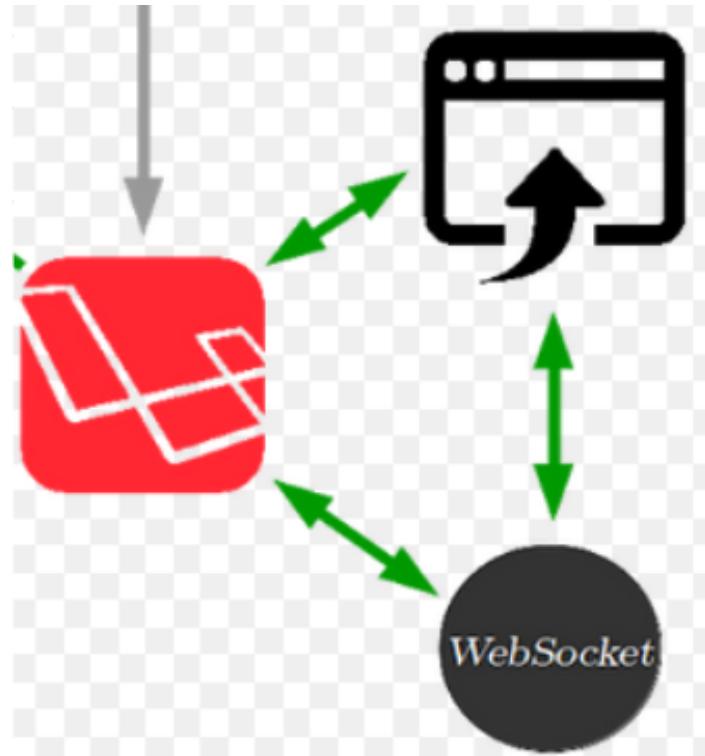


Figure 20: Estructura webserver-websocket-client

3.4.1 Gestió de caigudes

- Per part del servidor: S'inicialitza i es configura el websocket perquè estigui escoltant totes les connexions.
- Per part del client: En cada client a través de Javascript s'inicialitza una instància de websocket i es subscriu el client al tòpic de caigudes (D'aquesta manera pels websockets deixem la llibertat d'ampliació de funcionalitats).
- Execució: Quan el collaret envia un POST a l'API del servidor, després d'haver-hi hagut una caiguda, des del servidor s'envia l'alerta pel websocket a tots els clients oberts que s'han registrat al tòpic caigudes (En el nostre cas tots).

Es contempla la gestió d'1 a N caigudes, i es diferencien les caigudes en: d'atenció obligatòria i informatives. Els usuaris que estiguin treballant en la franja horària que s'ha produït la caiguda i que són responsables del sector en què s'ha efectuat la caiguda, és obligatori l'atenció de la caiguda, en canvi els usuaris que no compleixen les 2 condicions, els hi surt l'avís com a informatiu. Un cop assistida la caiguda apareix del panell d'alertes, i en cas de no haver-hi cap altre caiguda (situació corrent), la web redirecciona a l'apartat en que l'usuari estava abans de la caiguda.

3.5 Telegram

S'ha creat un bot de Telegram per enviar missatges des del webserver a un canal de Telegram privat destinat a tots els treballadors de la residència.

Un cop el servidor ha rebut una caiguda i l'ha tractat geolocalitzant la posició... A part d'enviar l'alerta de caiguda via els websockets per tots els clients web, s'envia un missatge de Telegram on hi consta el resident que ha caigut i un mapa on es mostra el sector i posició extacte on hi ha hagut la caiguda.

```
protected function sendTelegram($param1,$param2)
{
    $client = new Client(); //GuzzleHttp\Client
    $result = $client->post('https://api.telegram.org/bot615582162:AAGgt4fbRwCtNCzEltixeg4r1-WXay2AI/sendMessage', [
        'form_params' => [
            'chat_id' => '-1001271064871',
            'text' => "Client: ".$param1."\n"
        ]
    ]);

    $result = $client->post('https://api.telegram.org/bot615582162:AAGgt4fbRwCtNCzEltixeg4r1-WXay2AI/sendPhoto', [
        'form_params' => [
            'chat_id' => '-1001271064871',
            'photo' => "https://raw.githubusercontent.com/AbelSantiagoCode/caiguda/master/public/images/".$param2.".png"
        ]
    ]);

}
```

Figure 21

En aquesta imatge hi ha la funció encarregada d'enviar els missatges de Telegram que es crida des de l'algoritme un cop detectada i geolocalitzada una caiguda. Mitjançant l'API de Telegram s'envia un post. En el primer post s'hi envia les dades del resident que ha caigut i en el segon post s'hi envia limatge de la localització de la caiguda.

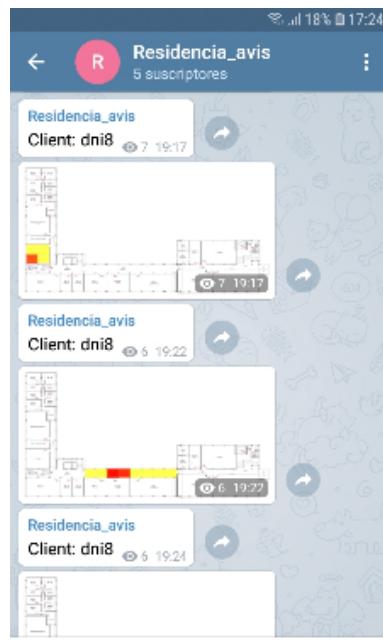


Figure 22: Grup telegram - residència

3.6 Base de Dades

A continuació, s'exposa el disseny del model relacional establert. En el següent model, podeu observar les taules i relacions existent.

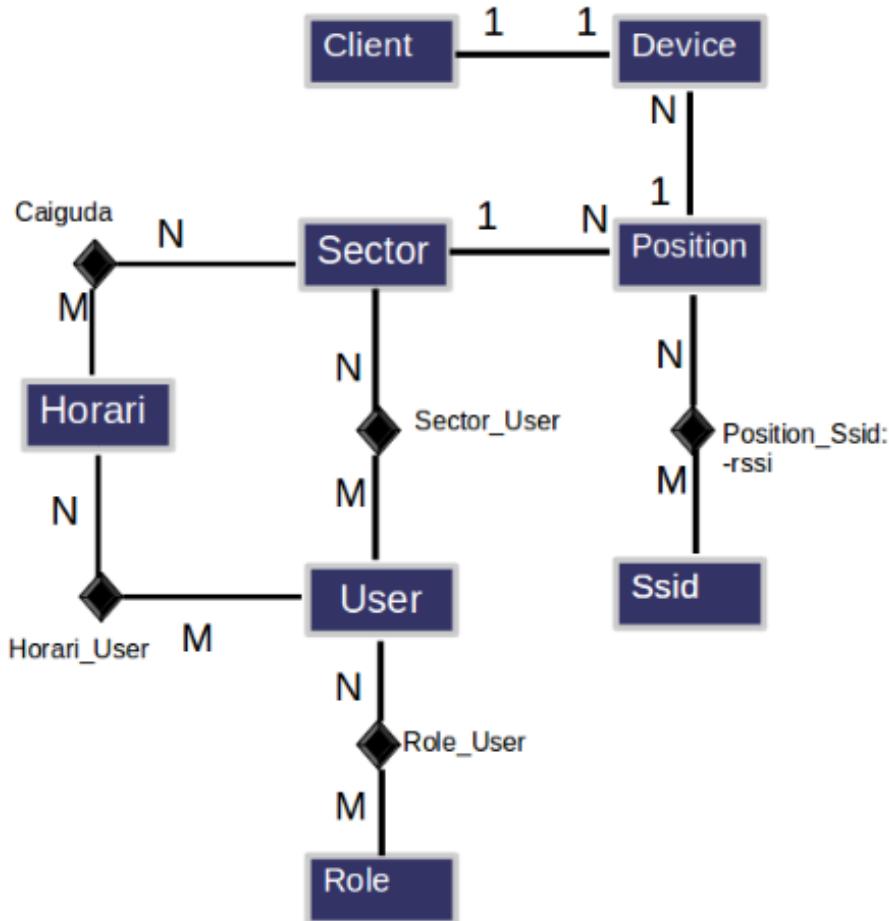


Figure 23: Model relacional de DB del projecte

Tot seguit, es descriu les taules existents:

- **Client:** taula que emmagatzema les dades relacionades als clients de la residència, es a dir, els avis i àvies.
- **Device:** taula que emmagatzema les dades relacionades als dispositius electrònics que portaren els avis i àvies (clients) de la residència. Aquesta taula té assignada una columna per l'identificador d'un client a un dispositiu. De la mateixa forma, també té assignat una columna per la posició en la qual es trobar.
- **Position:** taula que emmagatzema les dades relacionades amb la posició, com ara al sector al qual pertany.
- **Ssid:** taula que emmagatzema la llista de punts d'accés wifi disponibles per dur a terme l'algorisme de localització.
- **Position_Ssid:** taula que emmagatzema la llista de punts mapejats conformada pel conjunt (ssid,rssi).
- **Sector:** taula que emmagatzema la llista de sectors establerts.
- **User:** taula que emmagatzema l'informació relacionada al personal de la residència, es a dir, els infermers.
- **Sector_User:** taula que emmagatzema els sectors de treball assignats al personal de la residència.
- **Role:** taula que emmagatzema els roles i permisos que es fan servir per establir el sistema d'accés als serveis que ofereix la web.
- **Role_User:** taula que relació un role a un usuari.
- **Horari:** taula que emmagatzema les franges horàries de treballs establerts en la residència.
- **Horari_User:** taula que emmagatzema les franges horàries de treball dels users, es a dir, els infermers.
- **Caiguda:** taula que emmagatzema les caigudes detectades a un hora determinada i la seva posició, es a dir, el sector on es troba.

4 Conclusions

S'ha utilitzat Laravel com a framework per desenvolupar la web per varis motius; Perquè està basada en PHP, un llenguatge dedicat a la web i òptim pel Backend, per l'alt nivell d'extracció, pels diferents paquets que incorpora orientats a la seguretat, a la gestió de MVC, per l'Eloquent ORM, per l'escalabilitat i versatilitat de tots els components web en un sol projecte i al ser una de les frameworks més competents actualment, per tota la documentació i recursos que es troben per internet. Per a fer la gestió de websockets s'utilitza Ratchet per tal d'unificar al llenguatge amb què s'ha programat l'aplicació (PHP)

PHP és un llenguatge seqüencial. Quan un client fa una petició pel navegador, el servidor de PHP tracta aquesta petició i executa les línies de PHP convenientes dels quals en resulta una resposta. Per tant, al moment d'haver-hi una caiguda s'ha de mostrar l'avís a tots els clients que hagin accedit a l'aplicació web sense que aquests hagin de dur a terme cap acció. A més el servidor tampoc pot fer enquestes de caigudes, ja que és poc precís i en baixaria el rendiment. Per aquest motiu s'utilitzen websockets, per donar concorrència. A més s'utilitza la llibreria de Ratchet ja que ofereix els websockets amb PHP i d'aquesta manera s'unifiquen els llenguatges de programació de l'aplicació.

L'avís de les caigudes s'ha acabat implementant per 2 canals; canal privat de Telegram i pels websockets. El webserver s'ha optimitzat i s'han centrat els esforços per implementar-hi els websockets per fer l'avís de caiguda, ja que a través de la web és allà on hi consta tota la informació de la residència, es poden assistir les caigudes ... I és l'eina de gestió de la residència. Però a més, per tal de tenir un doble factor d'avís s'ha creat el bot de Telegram i canal privat, ja que enviant un missatge per Telegram, s'envia una notificació al telèfon mòbil dels treballadors i d'aquesta manera ens assegurem que la caiguda ha estat identificada pel màxim nombre de treballadors.

Pel què fa a l'algorisme de localització els resultats obtinguts són fructífer. Tot i que, el punt crític sobre el bon funcionament de l'algorisme recau en el fet d'un bon mostreig de les potències rebudes dels punts d'accisos en un punt determinat. Així mateix, el problema més rellevant pel mal funcionament es dona quan per un mateix punt el mostreig de les potències varia de forma rellevant en dos temps determinats diferents, probablement a causa del soroll capturat pel l'ESP.

Pel que fa al disseny del penjoll, creiem que portar-lo enganxat amb la sivella és una molt bona conclusió, ja que s'eviten molts problemes de comoditat i de falses alarmes per oscil·lacions que pogués tenir un disseny amb cordill i penjat del coll. A més a més, la pinça li dona múltiples opcions a l'hora d'escollar el lloc on portar-lo.

5 Bibliografia

- Laravel: <https://laravel.com/docs/5.7>
- Bootstrap: <https://getbootstrap.com/docs/4.1/getting-started/introduction/>
- Websockets: <http://socketo.me/docs/>
- Bot Telegram: <https://core.telegram.org/>
- Mysql: <https://dev.mysql.com/doc/>
- Instruccions ESP: <https://room-15.github.io/blog/2015/03/26/esp8266-at-command-reference/#AT+CWLAP>
- ADXL: <https://www.analog.com/en/search.html?q=ADXL345>
- Microcontrolador: <https://www.arduino.cc/>
- ESP01: <https://www.luisllamas.es/arduino-wifi-esp8266-esp01/>

6 Annex Codi font

- Codi font Weberser: <https://github.com/AbelSantiagoCode/caiguda>
- Codi font Websocket: <https://github.com/AbelSantiagoCode/caiguda/tree/master/app/Socket> and <https://github.com/AbelSantiagoCode/caiguda/tree/master/app/Console/Commands>
- Codi Base de Dades Model Relacional: <https://github.com/AbelSantiagoCode/caiguda/tree/master/database/migrations>
- Codi Base de Dades biblioteca: <https://github.com/AbelSantiagoCode/caiguda/tree/master/database/seeds>
- Codi font Dispositiu detector de caigudes: https://github.com/adriasalvans/codi_placa_caigudes