Detector Caigudes

Enginyeria de Sistemes

Dani Trias

Adria Salvans

Abel S. Baldelomar

Oscar Vendrell

June 19, 2018

Contents

1 Algoritme

1.1 Anàlisi

A l'hora de desenvolupar l'algoritme, primerament vam utilitzar l'integrat GY85 que incorporava un acceleròmetre, un giroscopi i un magnetòmetre. Vam descartar el magnetòmetre ràpidament ja que no acabavem de veure què podia aportar per detectar una caiguda. Després vam pensar de desenvolupar l'algoritme a partir de l'acceleròmetre i el giroscopi. L'acceleròmetre evidentemt ens donava resultats rellevants i útils, en canvi el giroscopi era dificíl d'estudiar i treure resultats rellevants i complementàris a l'acceleròmetre, per aquest motiu, al comprovar les possibilitats que ens donava l'acceleròmetre i el contrari per el què fa el giroscopi, vam decidir de desenvolupar l'algoritme basant-nos amb les dades obtingudes per l'acceleròmetre.

L'acceleròmetre ens dona dades d'acceleració dels eixos X, Y i Z. Per tant ràpidament vam veure que l'estudi s'hauria de basar en la comparació del comportament dels diferents eixos i alhora comparant-ho amb uns llindars. Per tant directament ens vam dedicar a recollir una biblioteca de dades per tal d'observar com canviaven els valors d'acceleració dels eixos.

1.2 Biblioteca

Per començar a fer les proves vam tenir en compte l'esquemàtic del datasheet de l'ADXL, de manera que puguem quantificar què passa ab cada eix per les diferents posicions.

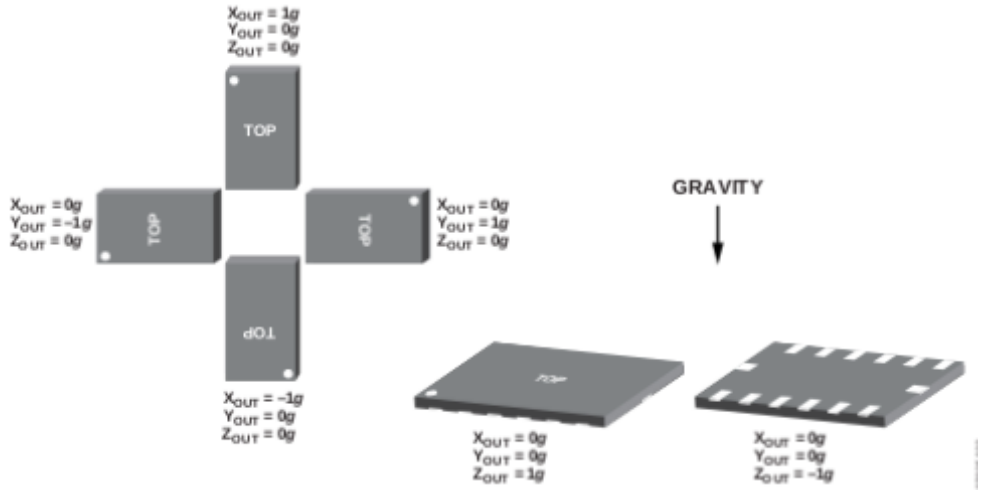

Figure 1: Imatge

A partir d'aquesta informació, vam fer les següents proves per veure si podiem observar com es plasmava en la pràctica:

- Simulacions previes per posar-nos en context i ajustar el tractament de les dades, per mostrar-les correctament... Les dades les mostrem amb G's.
- Caiguda de peu frontal

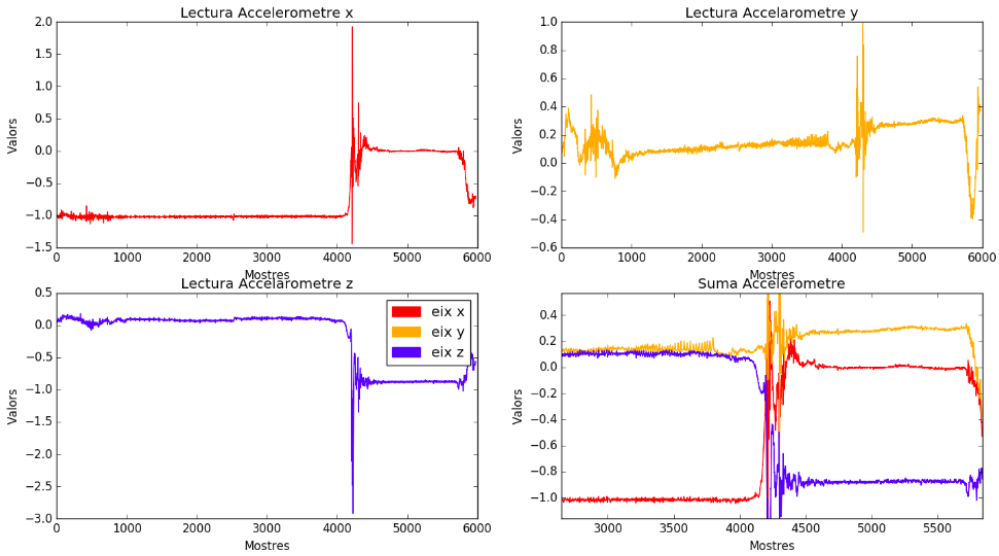

Figure 2: Imatge

Com es pot veure en la gràfica de les 3 acceleracions, a l'estar de peu, l'eix x=-1 i els altres eixos a 0. L'eix X és negatiu com a conseqüència de com posicionem l'acceleròmetre sobre la persona. Un cop hi ha hagut el threshold es pot observar com l'eix X i Y passen a ser 0 i l'eix Z a -1. Comprovant en el datasheet la disposicó X=0, Y=0 i Z=-1 observem que l'integrat està tumbat de cara cap avall. Per tant identifiquem perfectament la caiguda.

- Caiguda de peu de cul
- Caiguda de peu d'esquena

En aquestes proves el resultat és pràcticament igual al cas explicat anteriorment, amb la diferència que després del threshold vara l'eix Y o Z.

- Caiguda de peu d'esquena i costat dret

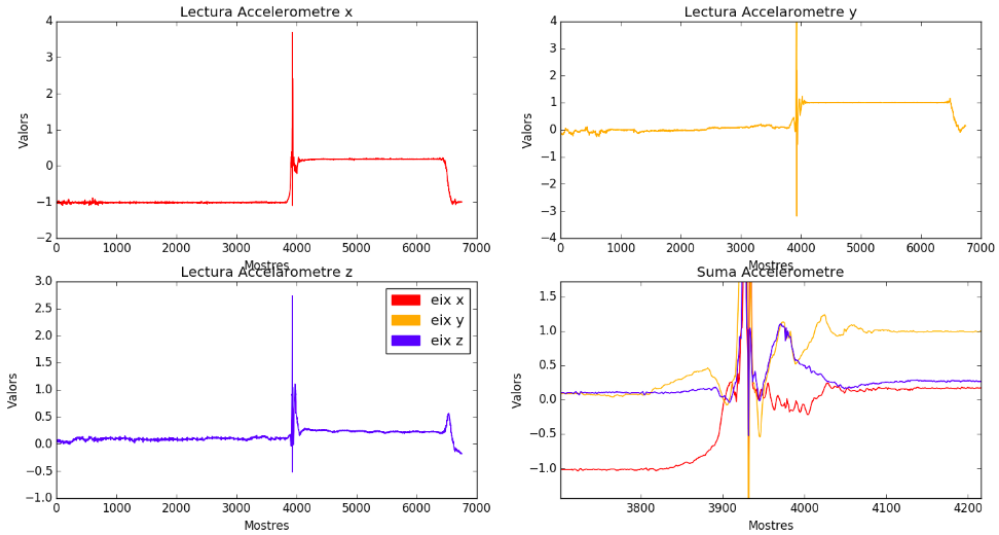

Figure 3: Imatge

Es pot identificar que l'individu està dret ja que X=-1, Y=0 i Z=0. Després del threshold, a causa de les

acceleracions produïdes, els eixos es troben : X=0, Z=1 i Y=0.5 corresponent a una posició d'esquena, i a mesura que passa el temps, Z va acostant-se a 0 i Y a 1, ja que està tumbat pel costat dret equival a la lectura d'eixos de X=0, Y=1 i Z=0.

- Caiguda de peu d'esquena i costat esquerra. Tenim els mateixos resultats que el cas anterior però amb Y=-1 després de la caiguda a casusa de tractar-se del cantó esquerra.
- Caiguda de peu assegut, i tornar-se a posar de peu:

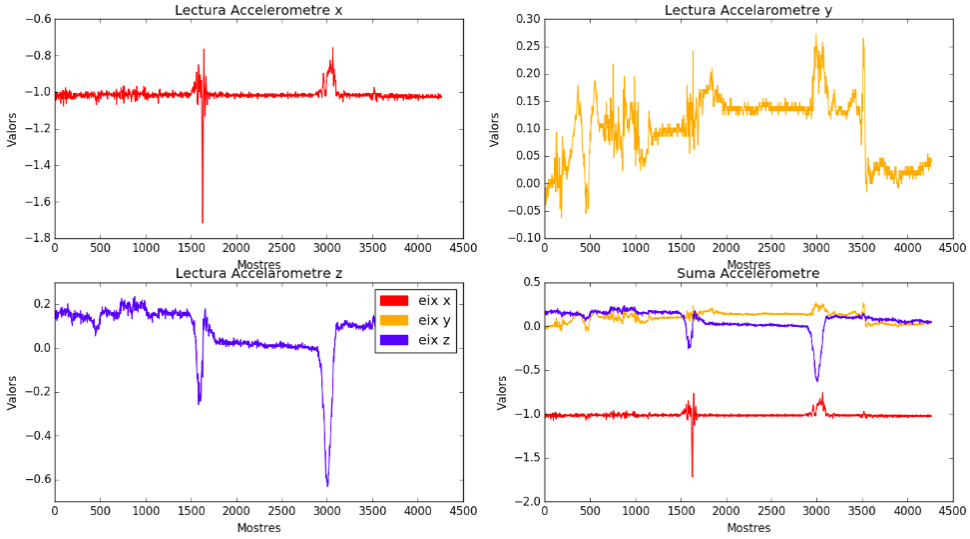

Figure 4: Imatge

Com es pot observar en el gràfic, la persona està de peu ja que Y i Z valen 0 i X=-1 (equival a 1 en l'esquemmàtic ja que el tenim posicionat del revés). En el moment de la caiguada hi ha un threshold important en l'eix X, i també coniderable en l'eix Z. Després de la caiguada, a l'estar assentat verticalment, els valors dels eixos continuen igual. Quan la persona s'aixeca, tampoc varien els eixos i s'observa el petit threhold en l'eix X i un altre és significatiu en l'eix Z.

- Salt molt intens

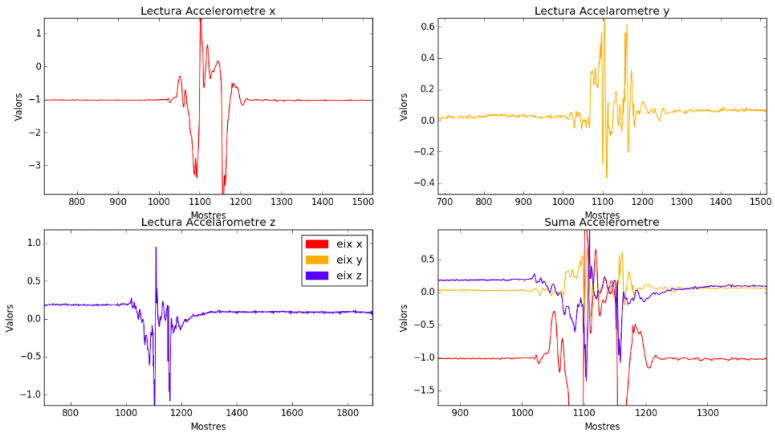

Figure 5: Imatge

A l'estar dret, els eixos X=-1 i Y=0 i Z=0. En el moment del salt salten molts thersholds en tots els eixos. Però la posició dels eixos després dels thresholds és la mateixa que abans. Per tant seria un tractament de les dades similar a la caiguada de peu.

- Salt poc intens. Té el mateix comportament que el salt molt intens però amb thresholds menys significatius.
- Tumbat i de peu

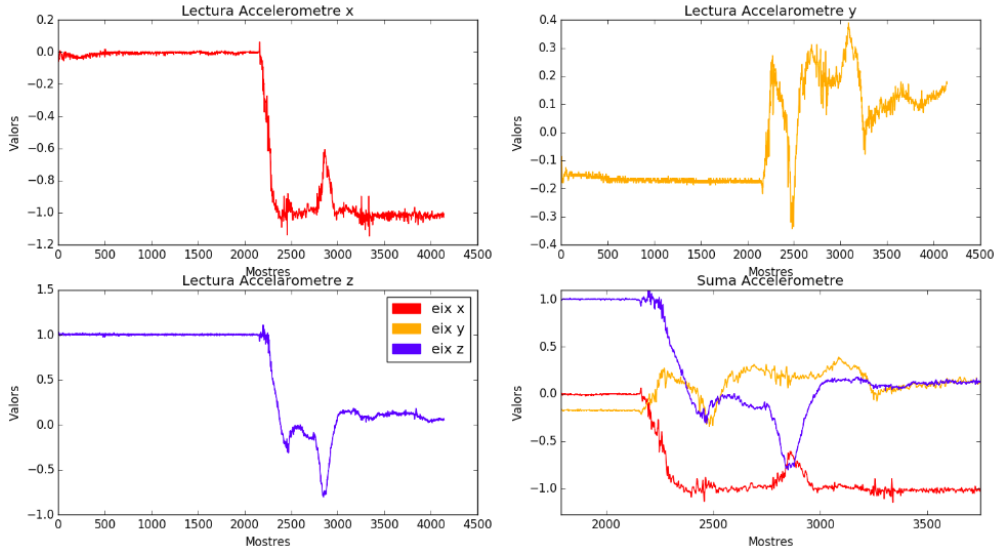

Figure 6: Imatge

Els eixos: X=0, Y=0 i Z=1, corresponent a estar tumbat segons l'esquemàtic del datasheet. Després d'axecar-se, amb una zona poc estable en quan a llindars d'acceleració, les lectures dels eixos s'acaben estabilitzant com es pot comprovar en la figura, de manera que X= -1, Y=0 i Z=0, corresponent a la posició d'estar dret. En aquest tenim en compte que no s'ha produït cap caiguda.

En el repositori del projecte hi ha la biblioteca complerta amb les diverses proves, les pertients lectures i els gràfics.

1.3 Conclusió de l'anàlisis

Com es pot comprvar en la biblioteca de dades recollides amb l'acceleròmetre, quan es produeix una caiguida sempre hi solen haver 2 eixos que han estat alterats després d'un brusc augment en l'acceleració d'aquests.

Per basant-nos amb aquesta premisa, hem de tenir en compte l'estat dels 3 eixos, detectar un augment considerable de l'acceleració, tenint en compte un llindar, seguidament d'un altre estudi de l'estat dels 3 eixos.

D'aquesta manera tenim guardada tota la seqüència i podem determinar que ha passat, basant-nos amb l'algorimte.

A més partim d'aquestes consideracions en les lectures dels eixos: Tenim en compte 3 posicions o estats; El dret, tumbat i desconegut (inclinació no classificable). Dins el tumbat podem precissar si ha set caiguda frontal, d'esquena o de costat, però a nivell de detectar la caiguda, si abans del threshold la posició era dret, i després del threshold la posició és tumbat, considerem que és una caiguda. Si entre un threshold la posició anterior és tumbat i la posició actual és tumbat, també considerem que s'ha produït una caiguda. Pel què fa les alertes, si entre un threshold la posició anterior és dret, i la posició actual també, considerem

que és una alerta de possible caiguda, ja que normalment en una caiguda la persona no queda dreta, i possiblement pot ser un petit salt, que la persona s'hagi assentat... També alertem quan la posició anterior és desconeguda i després del threshold la posició és tumbat, ja que al no tenir constància de la posició anterior no podem afirmar que s'ha produït una caiguda, però si alertar, i obviament si entre el threshold la posició anterior i la actual han set desconegudes, ja que possiblement pot ser un error de funcionament o un cas extrem que no som capaços de mesurar i per si de cas alertem.

1.4 Implementació de l'algoritme

Pel què fa a l'estudi dels 3 eixos, tant l'anterior com la posterior a l'impacte, vam tenir clar que haviem de definir una freqüència de mostreig i determinar unes finestres i guardar-ne les dades.

Per tant vam configurar el timer per obtenir una Fs= 100Hz, i vam considerar fer la primera finestra de 2s i la segona finestra de 0.2 s. Per guardar les dades obtingudes en les dues finestres, com s'explica a l'apartat del software, vam veure que era un punt crític ja que es poden gastar molts recursos i provocar que l'algoritme tingui deficiències o que fins i tot que no es pugui portar a terme. Per tant no acabem guardant les dades de les lectures directament, sinó que definim una llista amb 11 acceleracions llindar, i a partir de la comparació d'aquestes amb les dades llegides de l'ADXL, les classifiquem en el llindar on pertanyen, i per cada eix tenim una llista de contadors d'acceleració, dels quals indexem a la mateixa posició que la de la llista d'acceleracions llindar, la dada llegida, en la llista de l'eix corresponent, i aixis succesivament anar incrementant les diferents posicions de les llistes dels 3 eixos, segons a quina posició pertoca aquella lectura respecte les acceleracions llindar. Per tant acabem tenint 3 llistes amb 11 posicions cada una, que corresponen el nombre de cops que en aquella finestra hi han hagut accelaracions en aquells llindars.

Per tant hem dissenyat 2 màquines d'estats, una màquina de mostreig i una màquina de posició. En la màquina de mostreig tenim 2 estats, dels quals en el primer portem a terme el primer mostreig amb una finestra de 2 s, i estem pendents de si hi ha un threshold. En cas que no hi hagi hagut un threshold i s'hagi mostregat tota la finestra, s'envien les lectures a la màquina de posició. En cas de threshold enviem també les dades obtingudes a la màquina de posició i passem al segon estat. En el segon estat mostregem amb una finestra de 0.2 s i analitzem les dades que hem llegit de l'ADXL per tal d'enviar a la màquina de posició l'esdeveniment de si els eixos han estat alterats i per tant hi ha hagut una caiguda, com descriu el fonamement de l'algoritme.

La màquina de posició també té dos estats. En el primer estat hi accedim quan s'ha mostregat la primera finestra de 2 s en la màquina de mostreig. Allà comprovem a partir de les dades que s'obtenen a quina posició corresponen (dret, tumbat o desconegut). Si en el primer estat , la màquina de mostreig envia l'esdeveniment de threshold a la màquina de posició, es passa al següent estat de la màquina de posició. En el segon estat, quan s'ha fet el mostreig de la segona finestra de 0.2 segons a la màquina de mostreig, comprovem a quina posició corresponen les dades que s'han enviat a la màquina de posició i posteriorment analitzem si hi ha hagut una caiguda, alerta o res a partir de la posició analitzada a l'estat 1, i a l'analitzada en anteriorment a l'estat 2.

2 Comunicació

2.1 ICSP

Al començar aquest projecte primerament vam haver d'investigar sobre la connexió i programació de l'arduino via la connexió ICSP per així evitar utilitzar la connexió USB convencional de l'arduino. No hi van haver gaires problemes a l'hora de buscar informació per la connexió i programació ICSP, a més vam poder comprovar amb un arduino que no es podia programar via USB, com via ICSP si que es podia i aixi poder veure que fallava en aquest arduino; el controlador USB (va ser el cas) o bé el chip arduino.

2.2 I2C

Per poder comunicar entre els diferents sensors utilitats, utilitzem el bus I2C que ens proporcionen aquests mòduls. El bus I2C necessita dos resistències de 10k entre 3 V i les potes sda i scl. A la vegada aquestes potes han d'anar connectades a les potes A4, pel que fa a la sda i la scl connectada a la pota A5 de la placa Arduino.

2.3 Bluetooth

2.3.1 HC-06

Per començar a experimentar amb el pas de missatges per una comunicació inalambrica, vam agafar 2 moduls HC-06, els quals utilitzavem un de emisor i unaltre de receptor amb dues arduinos diferents. Al principi vam tenir alguns problemes a l'hora de configurar un com a emisor i l'altre com a receptor, pero vam descobrir que no era problema de configuració emisor/receptor, sino que teniem problemes amb les adreçes MAC per adreçarnos de un arduino a un altre. Un cop solucionat gràcies a les comandes AT vam poder fer els primers experiments, però no vam profunditzar mes, ja que vam decidir no utilitzar aquesta tecnologia principalment per tema de la potencia que tenen, ja que es molt poca i a més d'uns 5 metres ja perdiem la comunicació i no unteresava.

Apart del principal inconvenient de la distància, tambe vam veure que a l'hora de transmetre les dades, era mes lenta que no pas el WiFi.

Cal dir que no tot eren inconvenients, una de les principals avantatges era la facilitat a l'hora de transmetre els missatges d'un arduino a un altre.

El consum tambe era un gran avantatge ja que consumeix entre 30 i 40 mA i es pot alimentar a 3,3 V.

2.4 Wifi

2.4.1 ESP-01

Un cop fets els primers experiments amb els moduls bluetooth, vam començar a experimentar amb el modul wifi ESP-01, on nomes conectar i començar a intentar fer les comandes AT Vam tenir molts problemes, ja que ens retornava caracters no imprimibles o aleatoris i no ens donava el ready. Un cop descartat els problemes de conexió, vam profunitzar en les comandes AT, i el perque no ens retornava cap ready.

Al principi vam pensar que sera pel consum del ESP-01 ja que és mes elevat que el del bluetooth (200mA) i vam alimentar el ESP exteriorment i no amb la alimentació del arduino.

En aquest punt només podia ser un problema de software ja que ara ens retornava al ready i podiem fer comandes AT, pero a l'hora de buscar les xarxes wifi, no ens mostrava totes les xarxes disponibles o no ens mostrava cap.

Investigant vam descobrir la comanda AT+UART_DEF la qual podem configurar el baud rade del ESP-01 i ens va solucionar practicament tots els problemes a l'hora de fer les comandes AT i ja funcionava perfectament.

2.4.2 Servidor TCP

Un cop solucionats tots els problemes de configuració, creem un punt d'accés WiFi, on aprofitant el script que hem fet a l'assignatura de Aplicacions i Serveis d'Internet, creem un servidor on es rebrant els missatges d'alerta de caiguda.

Per tant finalment vam configurar el modul com a client, d'aquesta manera veiem tot el que envia al servidor.

2.4.3 Emulador Router

Per simular la connexió del ESP-01 a un router, hem creat un punt d'accés wifi amb el mobil on conectem el ESP-01 i el ordinador i ens comuniquem via sockets.

2.5 Modul Serial

En el començament del desenvolupament, com que hem utilitzat l' IDE d'Arduino, hem utilitzat el Serial de l'IDE per fer el debbug, i a mesura que hem anat desenvolupat versions definitives, aprofitant que tenim la llibreria libpbn, hem aprofitat el modul **serial_device** per poder fer debugging dels moduls que hem implementat i asegurar-nos que tot funciona correctament.

2.6 Bot Telegram

Hem creat un bot de Telegram per tal de tenir una comunicació ràpida i òptima amb els treballadors de la residència, ja que tenen incorporat en el mateix mòvil personal o de feina l'aplicatiu per l'avis de caigudes. A més aquest aplicatiu, Telegram, no només té un ús tancat a la nostre aplicació, ja que obviament el poden utilitzar per la comunicació en molts àmbits, i a més utilitzable en multi-plataforma (Móvil, Tablet, PC...), per tant és versàtil i no obliguem a tenir una aplicació tant tancada.

Per fer-ho hem creat un bot Telegram, dels quals l'hem configurat de manera òptima i agradable per la tasca a dur a terme. A més hem creat un canal privat pels treballadors de la residència on hi hem afegit el bot com a administrador per tal de gestionar la comunicaciò.

Pel què fa a l'enviament, en el servidor Python hi tenim les diverses configuracions del bot i el canal per dur a terme l'enviament, per part del bot, cap al canal, i d'aquesta manera quan s'ha enviat des del collaret (Arduino) una senyal de caiguda o d'alerta, aquesta és enviada pel servidor cap al canal de Telegram. D'aquesta manera els treballadors es podran organitzar òptimament per dur a terme l'execució protocolaria adequadament.

Oferim aquesta configuració en quan al bot i al canal de Telegram, però som flexibles a propòstes o re-organitzacions del canal de comunicacions de Telegram per part de la residència (exemple: canal per sector...).

3 Hardware

3.1 Servidor Python

El servidor python està hostetjat a un pc, implementable facilment a un pc-server de la residència, engegat les 24h del dia.

3.2 1r Prototip

Ens plantejem el disseny a partir de regletes ja que els components que utilitzarem són de montatge per insercció. A partir d'aqui, utilitzarem la placa DIPSE per muntar totes les regletes femella, i d'aquesta manera poder posar hi treure els components amb més versatilitat.

Basem el primer prototip sobre la placa Arduino UNO.

3.2.1 Sensors

Primerament vam utilitzar la placa GY-85 per poder fer proves amb més sensors complementaris a l'acceleròmetre, però finalment hem acabat utilitzant l'ADXL que només incorpora un acceleròmetre. Per comunicar entre els senors utilitzem la comunicació I2C.

3.2.2 Actuadors

A l'inici vam utilitzar dos mòduls Bluetooth HC-06, un configurat com a mestre i l'altre com esclau per poder-se comunicar entre ells. Tot i així vam tenir molts problemes de connectivitat i vam optar per utilitzar el mòdul wifi ESP_01. Ens hem trobat amb diverses dificultats alhora de posar en funcionament els dos sistemes. En el mòdul Bluetooth hem tingut problmes de traspàs de dades. En l'ESP_01 els problemes que ens hem trobat amb la palca Arduino UNO és que aquesta no està preparada per poder alimentar el mòdul a 200 mA amb una tensió de 3,3 V, ja que el regulador LDO de 3,3 V només pot proporcionar 50 mA. Per poder solucionar aquest inconveninet d'alimentació hem optat per muntar un condensador electrolític de 200 mirco-Farads i 10 V per estabilitzar la tensió d'alimentació del mòdul.

Utilitzem les potes RX i TX de l'ESP_01 per comunicar-nos amb l'Arduino.

3.2.3 Consums del sistema

El nostre sistema funcionant sempre i sense transmetre res, té un consum de 170mA. Aquí hem d'incloure el consum del Mc Atmega328p, el consum del convertidor usb que el podem considerar menyspreable, el consum de l'acceleròmetre, i el consum del ESP-01 sense transmetre.

$$\text{tiempo de descarga} = \frac{\text{carga electrica bateria}}{\text{consumo electrico dispositivo}}$$

Figure 7: Imatge

Si per el nostre dispositiu triem una bateria de 2000mAh, el nostre prototip 1, tindrà una autonomia de:

2000mAh/170mA=11h 45min 52s

El dispositiu dura tan poc, ja que no tenim implementat cap mode sleep, i a més a més tenim el mòdul ESP sempre activat, no transmeten, però si que es troba alimentat.

3.3 2n Prototip.

Com que veiem que amb el prototip nº 1 tenim bastants problemes amb el mòdul Esp-01, per mals contactes o falta de corrent i amb l'acceleròmetre per falta de subjecció, ens hem plantegat dissenyar i fabricar amb placa de prototipat un segon prototip, el qual ja no tingui aquestes falles i sigui a l'hora funcional per poder fer proves amb subjectes del client.

Per dissenyar-lo ens centrarem en:

- Anul · lar tot el que no sigui necessari de la placa Arduino.
- Proporcionar la suficient potència a la placa Esp-01.
- Fer la placa amb el mínim de components possibles, i fer el disseny el més petit possible.
- Incorporar els elements necessaris per la solució del problema.

3.3.1 Compliment dels Objectius de disseny.

Per anul · lar tot el que no és necessari de l'arduino, hem investigat una mica, i hem decidit només posar un regulador LDO NCP1826s amb encapsulat To-223 de 4 potes, per tal de proporcionar 3.3V a tota la placa. Aquest regulador serà l'encarregat d'alimentar tot el nostre collaret a 3.3V. Aquesta decisió tindrà uns efectes positius en el disseny i uns efectes negatius que comentarem a continuació. A la vegada hem suprimit totes les arrays de pins que no ens fan falta de l'arduino.

Per proporcionar la suficient potència a l'arduino hem triat expressament el regulador NCP1826s ja que és capaç d'aportar 1000mA a la placa. Aquest component ens dona marge suficient de potència per tal que l'ESP-01 no decaigui quant estigui transmetent una senyal.

Els components afegits que té la placa que són absolutament necessaris per el nostre disseny són: - Un botó d'alarma manual. - Un modul ESP-01 per transmetre les dades - Un acceleròmetre ADXL-345 per detectar els moviments. - Un botó de reset per si per alguna circumstància necessitéssim rebotegar el nostre sistema. - Un connector d'alimentació. - Un connector ICSP per poder carregar el programa al nostre sistema. - Un microcontrolador Atmega328p. - Un led per mostrar quant s'ha detectat una caiguda i per ajudar a localitzar la persona.

Per tal que tots els components ocupin el mínim d'espai, hem optat per muntar el màxim de components de muntatge superficial, i utilitzar el mínim de components de thrue-holl. A la vegada hem optat per encapsulats de la mida 0805, ja que tot i ser components petits, amb una mica de paciència es poden soldar a mà.

Per poder dissenyar les PCB més còmodament optem per pensar amb un disseny de doble cara, sent la top la capa de muntatge superficial, i la Buttom la capa dels forats. D'aquesta forma ens permet un traçat de pistes més còmode i un disseny més compacte.

3.3.2 Avantatges que ens suposa el disseny.

- Només hem d'utilitzar un regulador per tota la placa. Ocupem menys espai per fer l'alimentació.
- Els components de muntatge superficial ocupen poc espai i el disseny és compacte i poc voluminós.
- L'alimentació de 3.3V serveix per tots els components de la placa.
- El disseny està pensat per treballar amb bateries liPo de 3.7V.
- Els nivells lògics de l'arduino passen a ser 0 i 3.3V, per tant no hi ha problemes amb la comunicació I2C, ni amb la comunicació Tx Rx de l'ESP en quan a voltatges.

3.3.3 Problemes que ens suposa el disseny.

- Un problema que ens suposa el disseny amb quant al tipus de muntatge que hem decidit, és que les reparacions i substitucions d'elements son molt costoses, ja que dessoldar un component SMD o un

microxip SMD és difícil.

3.3.4 Solucions als problemes que ens presenta el disseny.

- Per comoditat en comptes d'utilitzar un altre Arduino UNO com a programador mitjançant el port ICSP, utilitzarem un mòdul adaptador de corrent i interfície serie/UART TTL, 3.3 V/5V, CP2102 usb Serial Converter.

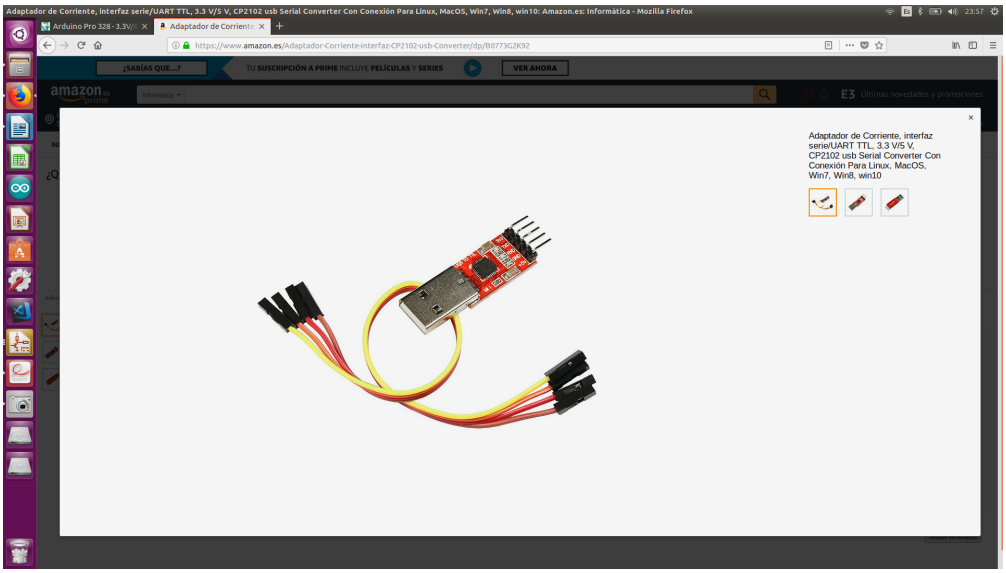

Figure 8: Imatge

El qual ens proporciona una manera senzilla de programar la nostre placa a partir del connector ICSP ja que disposa de alimentació 3.3V i no necesitem carregar cap programa a cap Arduino UNO perquè fagi de pont. Simplement a través de l'IDE o del Makefile configurar que es programa amb aquest adaptador.

A Continuació disposeu de l'esquema elèctric esmentat en tot l'apartat, i els dissenys PCB del Prototip nº2.

Hem de tenir en compte que la capa Top és la del muntatge superficial, i la Buttom és la capa inferior on arribaran els components amb forats.

3.3.5 Esquema.

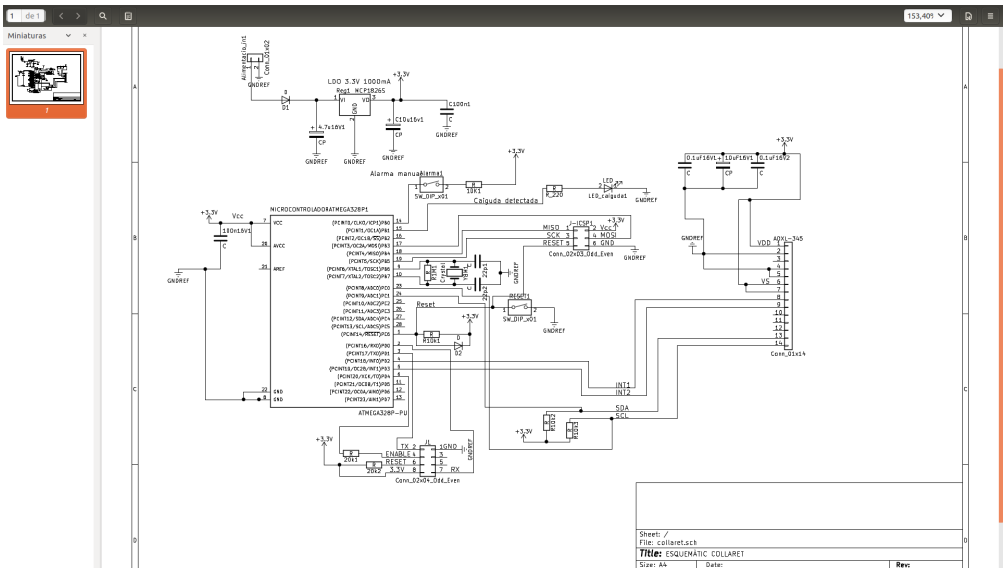

Figure 9: Imatge

3.3.6 PCB

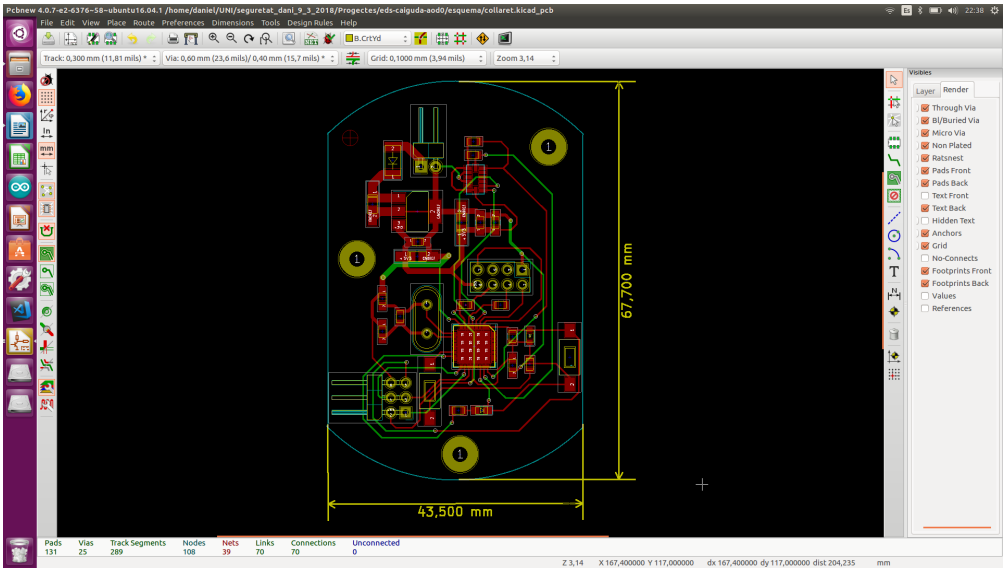

Figure 10: Imatge

3.3.6.1 Top

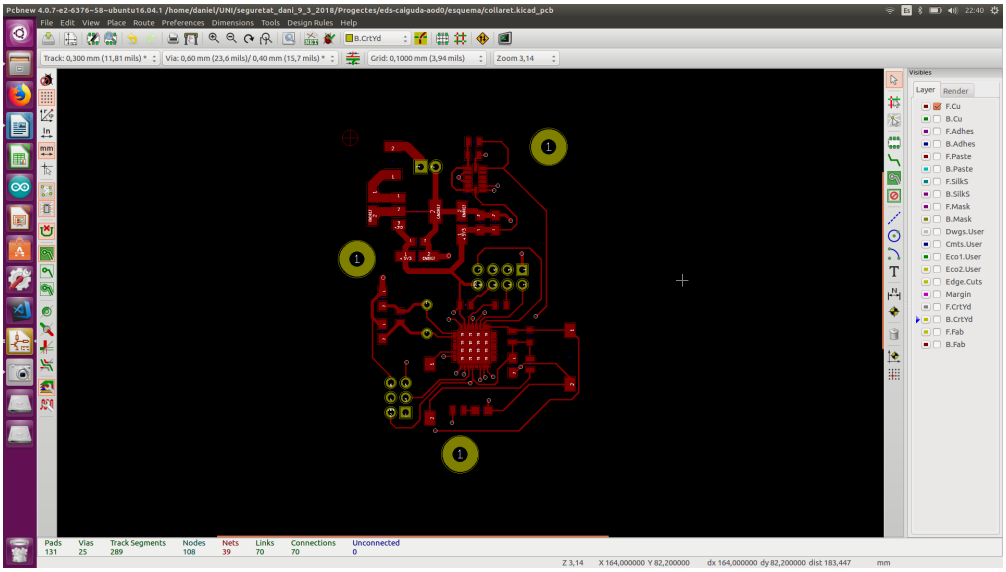

Figure 11: Imatge

3.3.6.2 Buttom

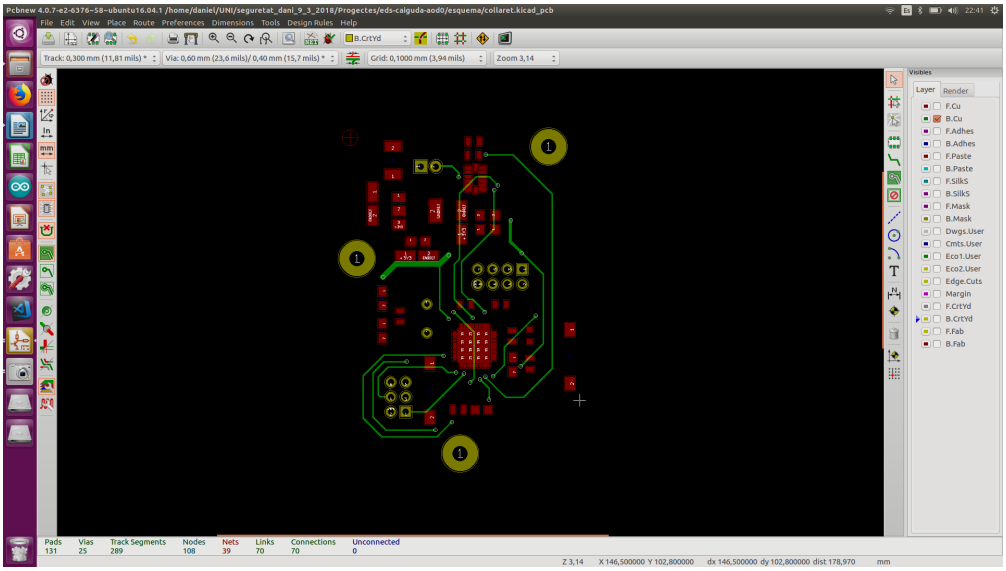

Figure 12: Imatge

3.3.6.3 Plans de massa

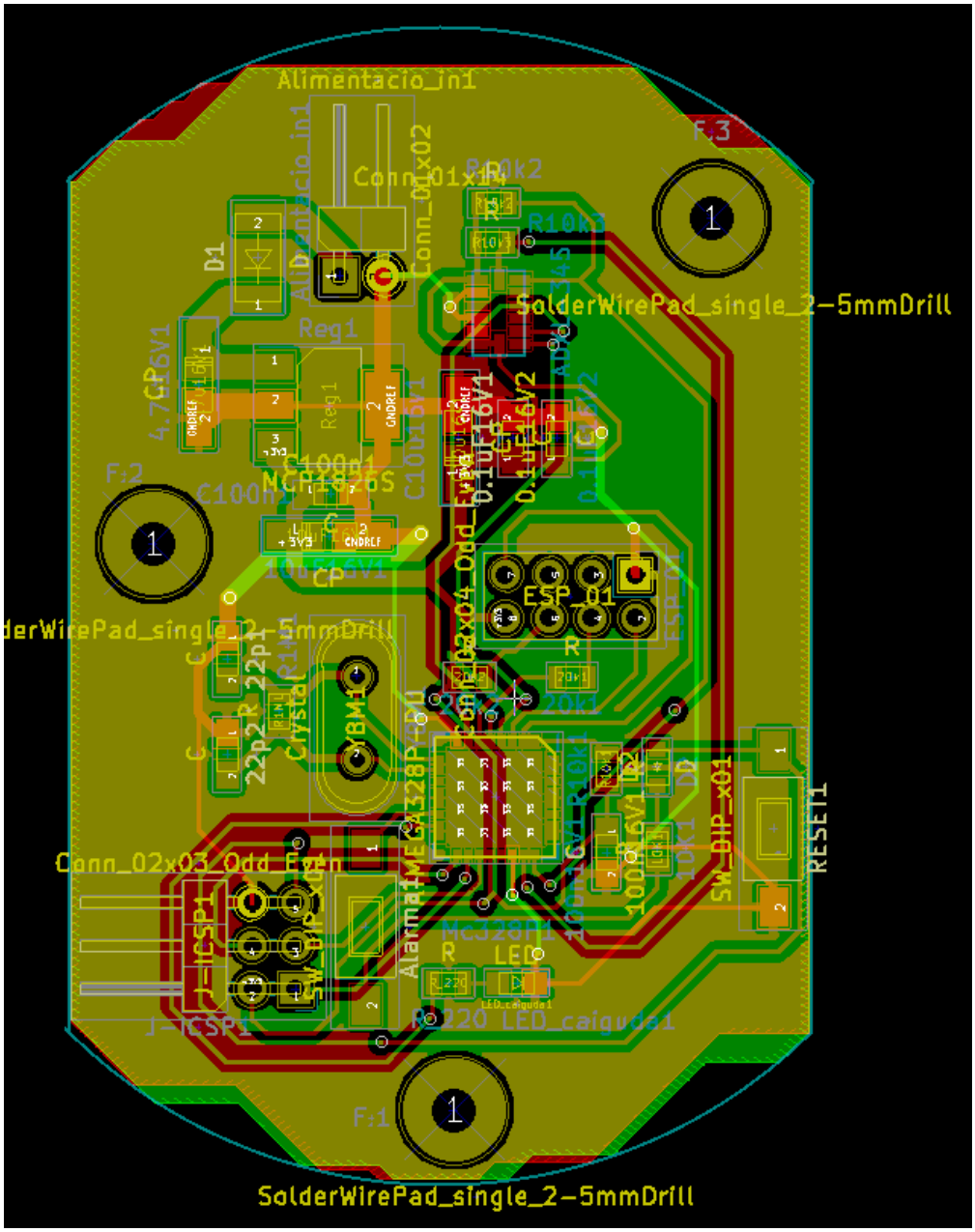

Figure 13: Imatge

3.3.6.4 3D

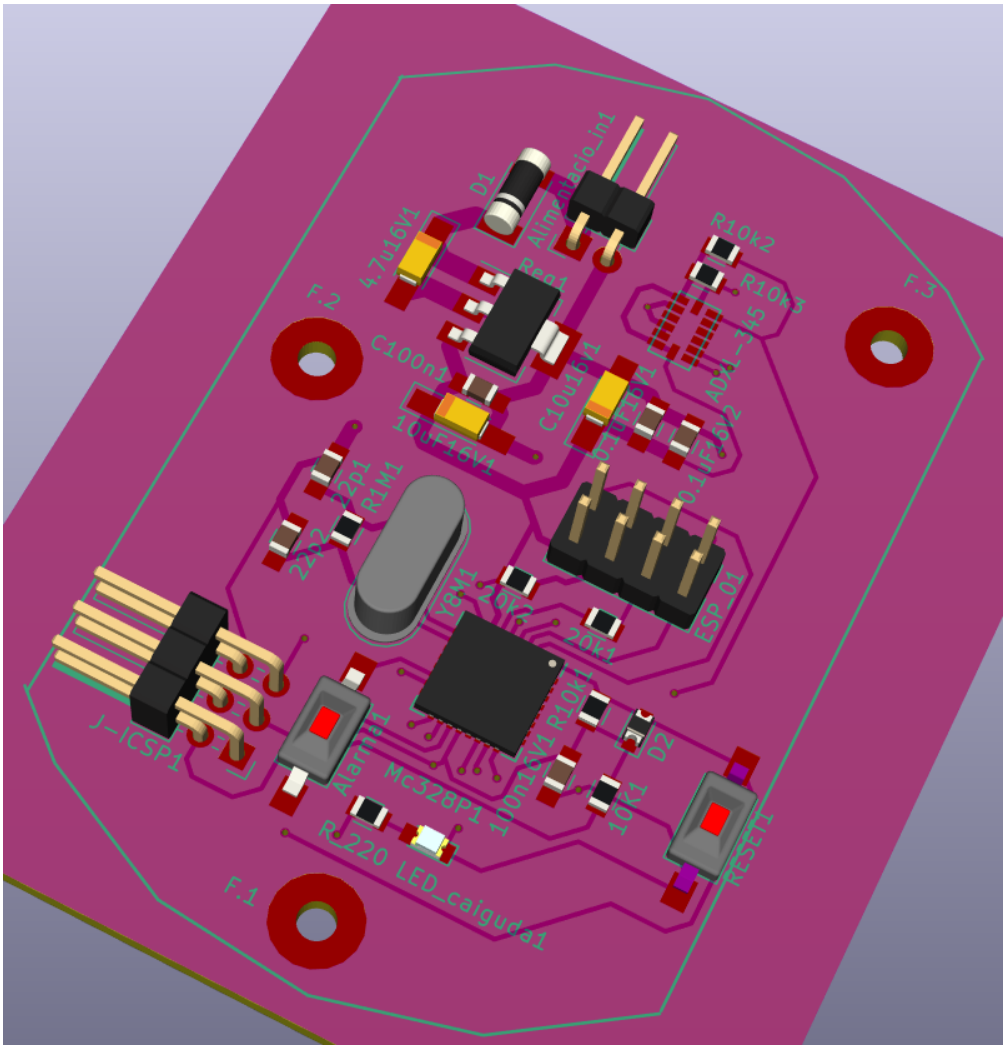

Figure 14: Imatge

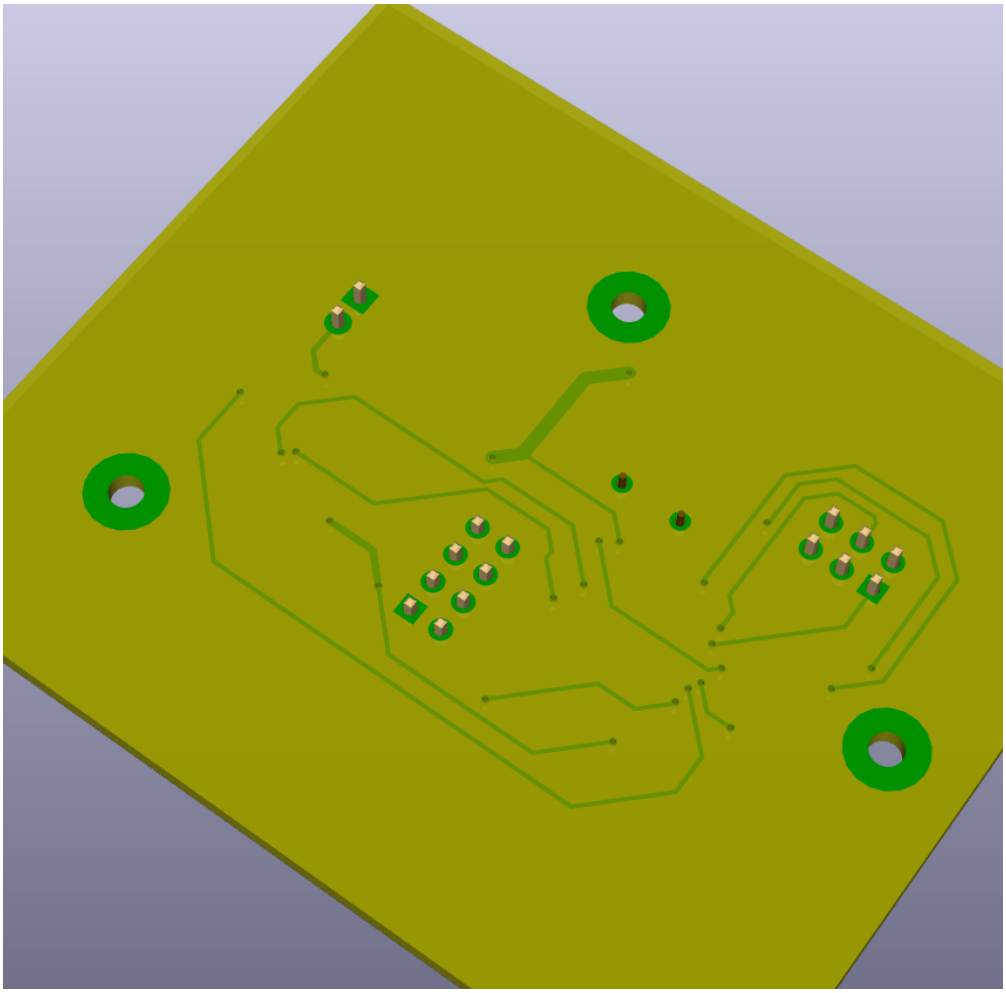

Figure 15: Imatge

3.3.6.4.1 Plans de massa

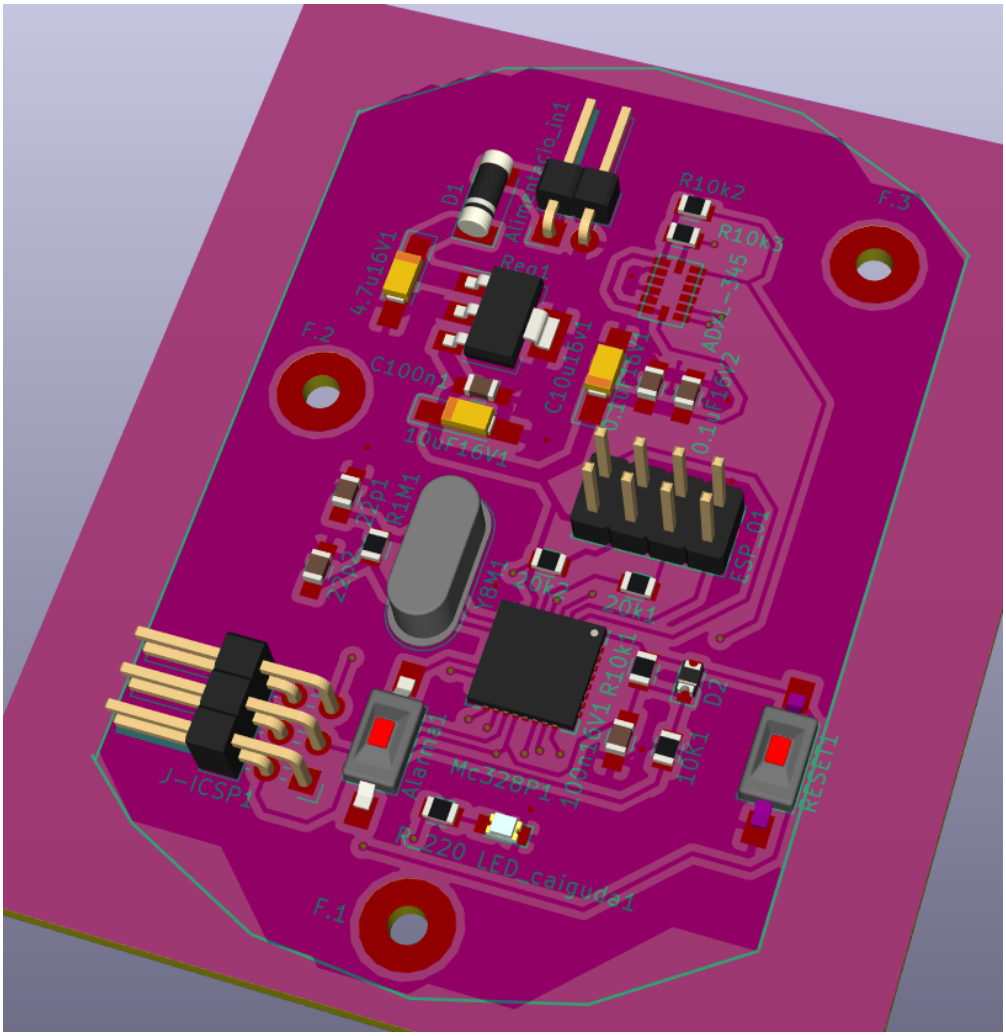

Figure 16: Imatge

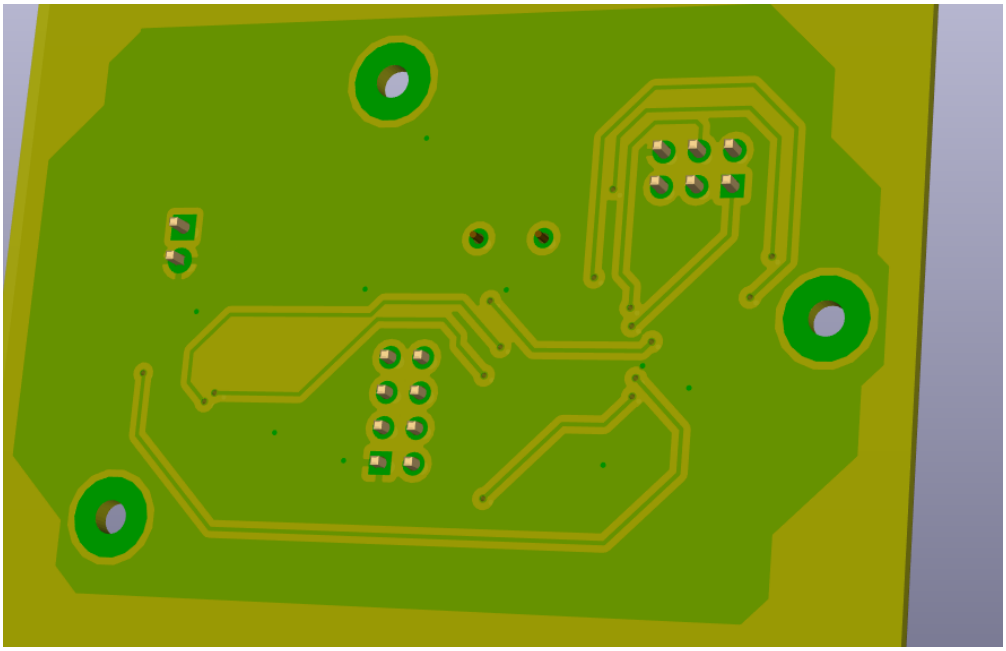

Figure 17: Imatge

4 Software

4.1 Entorn

Per desenvolupar la part software d'aquest projecte vam començar fent-ho amb l'Arduino IDE ja que voliem buscar optimització en la creació d'un projecte i l'interacció amb altres mòduls ja creats, amb extensió cpp i per tant, l'ús era més fàcil d'aquesta manera.

Tot hi així tot el codi que hem anat creant sempre s'ha fet amb C. Fins el punt que s'utilitza l'Arduino IDE només per el linkatge del mòdul SparkFun_ADXL345, ja que tot el projecte ha acabat sent amb C, d'aquesta manera tenim un bon control i optimització de recursos.

4.2 Estudi de l'algoritme

Per dur a terme l'algoritme i identificar com són les caigudes, igual que per dur a terme una biblioteca de dades, ho hem fet mitjançant Python, amb la llibreria matplot i la seva API que ofereix fer els subplots dels gràfics... I a més, com es comentarà més endavant per desenvolupar l'algoritme de detecció de caigudes.

4.3 Programa principal - Main

El programa principal és simple ja que simplement consisteix en les inicialitzacions de les màquines de mostreig i de posició, i l'arrencada de la màquina de Mostreig.

4.4 Unitats de mesura

4.5 Maquines d'estat

L'algoritme de la detecció de caigudes es basa en 2 màquines d'estat, la màquina de mostreig i la màquina de posició. Com diu el nom, la màquina de mostreig és amb la que fem el mostreig i la màquina de posició per, a partir del mostreig d'unes finestres, amb el "core" de l'algoritme de detecció de caigiudes, decidir si s'ha produït una caiguda, una alerta...

Primerament el desenvolupament de les màquines d'estat s'ha fet amb Python donada la complexitat de l'algoritme. Un cop passats els tests de qualitat, hem traslladat les màquines a C.

4.5.1 Màquina de Mostreig

El mostreig el fem de la següent manera; primerament una finestra de 2 segons,

La funció *run_FSM()* és la pròpia màquina d'estats, dels quals primerament fem un mostreig de 2 s, llegint de l'acceleròmetre *read_adxl()*, i classificant les lectures amb *classificar_rang()*. Tenim una llista on hi tenim 11 llindars d'acceleració, i segons el valor de la lectura que es fa, per cada eix tenim una llista, on hi ha un comptador de les diferents acceleracions indexades per la llista de les 11 acceleracions llindars. Després de la finestra de 2 s, enviem les 3 llistes corresponents als 3 eixos (X, Y ,Z) de contadors de les acceleracions llegides i classificades segons els llindars, cap a màquina d'estats de posició amb *setBuffers()* i amb el corresponent esdeveniment de final de mostreig *send_event(Time_out)*. Si durant el mostreig d'aquesta finestra hi ha hagut un threshold, que ho identifiquem amb *first_threshold(x,y,z)*, funció on comprovem per cada lectura de la finestra, si aquesta ha passat d'un valor determinat (en eix X i Y, eix X i Z o eix Y i Z), doncs passem al 2n estat, on tornem a fer un altre mostreig però d'una finestra més petita, de 0.2 s. Un cop passats els 0.2 s amb la funció *check_zone_threshold()*, comprovem si pels eixos

explicats anteriorment han passat d'un cert llindar, i amb *send_event(Threshold)* enviem a la màquina de posició que hi ha hagut una caiguda.

4.5.2 Màquina de Posició

A través dels *send_event()* que crida la màquina de mostreig, s'executa el run_FSM() de la màquina de posició amb el corresponent event passat per paràmetre. Amb la funció *update_posicio()*, per les 3 llistes de contadors d'acceleracions dels corresponents eixos X,Y i Z, busquem l'accelaració màxima que hi ha hagut en cada eix, i segons un llindar1 i un llindar 2 d'acceleració, aplicant l'algoritme que determina l'estudi d'eix X i Y, Y i Z o X i Z, Comprovem per les 3 combinacions si les acceleracions màximes dels 3 eixos estan dintre dels 2 llindars aquests, i així determinem si la persona està dreta, tumbada o posició desconeguda.

Per tant amb el primer estat comprovem amb *update_posicio()* amb quina posició està la persona. Dins d'aquest estat, si la màquina de mostreig envia un Threshold com a event, passem al següent estat de la màquina de posició, on la màquina de mostreig fa el mostreig de la segona finestra de 0.2 segons, ia la màquina de posició s'espera a que acabi i li envi l'esdeveniment Time_out. Quan aquest arriba es torna afer un *update_posicio()* i seguidament un *check_fall()* on es comprova la posició anterior (primer *update_posicio()* del primer estat) amb la posició actual (darrer *update_posicio()*) i segons l'algorimte (si la posició anterior = dret i la actual = tumbat => caiguda... I els diferents cassos explicats en l'apartat d'algoritme), s'envia per l'ESP al servidor de python el corresponent missatge de caiguda o d'alerta.

4.6 ADXL

Inicialment vam utilitzar l'integrat GY85, dels quals hi havia integrat un acceleròmetre, un giroscopi i un magnetòmetre. El magnetòmetre tenia un ús dubtós, per això no hi vam fer masses proves... I vam creure que podiem fer una convinacció d'ús amb l'acceleròmetre i el girsocopi. Al veure la complexitat del giroscopi, i que tampoc aportava dades massa rellevants per el què voliem nosaltres, vam optar per només utilitzat l'acceleròmetre, i per aquest motiu més tard vam adquirir l'integrat ADXL, i per aquest motiu vam crear una API feta a les nostres necessitats a partir del mòdul SparkFun_ADXL345.

4.6.1 Adxl api

Adxl és un mòdul que hem creat nosaltres, que ofereix una API amb les funcions *init_adxl()* i *read_adxl()*, que són les utilitats per part de l'integrat que necessitem. Estructuran-ho per capes, aquest mòdul seria la capa superior del mòdul SparkFun_ADXL345 on hi ha la API complerta de l'ADXL. En el *read_adxl()* llegim amb una freqüència de 100 Hz, configuració de timer que fem anteriorment en la funció *init_adxl()* , i en cada Fs, en l'interrupció de timer habilitem un flag que alhora està com a condicional en el *read_adxl()*, d'aquesta manera ens assegurem que la lectura només es fa quan toca.

4.6.2 SparkFun_ADXL345

Aquest mòdul és el dirver de l'ADXL. És allà on hi han tots els define's, types, funcions... de l'integrat.

4.7 Timer

Mòdul per configurar el timer adequadament, amb la funció **setup_tmr0**. Com diu el nom, utilitzem el timer 0 de l'Arduino.

4.8 ESP_01

Mòdul que ens hem creat per utilitzar com a dirver de l'ESP_01, el mòdul wifi. Consta de la funció *setup_ESP01()* i *sendstring_ESP01()*. A la *setup_ESP01()*, fem les pertinents configuracions perquè el mòdul wifi funcioni correctament i el connectem amb el servidor Python.

Amb la funció *sendstring_ESP01()* enviem cadenes de caràcters (strings). Per tal que aquestes funcions funcionin correctament, com que envien comandes a l'ESP_01, i aquest els respòn gerenalment amb un OK, i això significa que tarda un temps, després d'executar una linia de comandes cap a l'ESP_01 tenim una funció que es diu *check_ok()* que és bloquejant i espera la resposta OK de l'ESP_01 perquè les funcions del mòdul puguin continuar executant-se després d'haver rebut l' OK.

4.9 Serial

Primerament utilitzavem el Serial de l'Arduino IDE, però quan hem anat passant tot el codi a C, hem implementat el mòdul serial, que està format per; *serial_init()* per inicialitzar el canal, concretrament a una baud rate de 115200, *serial_get()* per fer lectura d'un byte pel canal, *serial_can_read()* per si es pot dur a terme una lectura, *serial_put()* per enviar un byte, *serial_print()* per enviar un string i *serial_println()* per enviar un string amb nova linia i retorn de carro.

Aquest mòdul ha set molt útil per anar fent debbug.

4.10 Server

El server l'hem fet amb Python. L'hem plantejat com un objecte per la versabilitat que dona (variables globals, instàncies...), i amb mètodes genèrics per poder-hi connectar varis clients, que el funcionament sigui "amigable" i entenedor...

4.10.1 Bot de Telegram

Per a dur a terme l'enviament de missatges cap al canal de Telegram mitjançant el bot de Telegram, hem creat un mètode de la classe Server anomenat **send_message_telegram** que tenint l'id del bot, el token i l'url, pots enviar l'string que vulguis cap al canal privat de Telegram.