UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Politècnica Superior d'Enginyeria de Manresa

Treball Final de Grau

Display activat per moviment fent servir persistència de visió

Grau en Enginyeria TIC

Curs 15/16

Autor: David Carrera Casado

Director: Pere Palà Schönwälder

Data: Octubre 2016

Localitat: Manresa

RESUM DEL PROJECTE

Aquest projecte té com a objectiu el disseny i implementació d'un display basat en persistència de visió, el qual utilitza moviment per tal d'activar-se i detectar el moment adequat per re-dibuixar el text mostrat. El display es bassa en una barra allargada, amb un acceleròmetre en un extrem, i una tira de leds en part de la seva llargada. Els components electrònics són muntats en una placa de circuit imprès i controlats per un microcontrolador tipus Arduino, el qual inclou un mòdul Bluetooth, permetent que el text es pugui canviar enviat missatges via Bluetooth.

Aquest document parla sobre les principals alternatives pel que fa a displays basats en persistència de visió, principalment els displays giratoris. A continuació, les diferències amb aquests s'enumeren. Finalment, les varies tecnologies utilitzades en el projecte són explicades, per tal d'introduir antecedents.

La següent part cobreix extensivament el disseny i implementació del dispositiu, començant amb una descripció general i continuant amb l'explicació tant del hardware com del software que composen el projecte

L'última part del document inclou tot el codi font escrit específicament pel projecte.

OVERVIEW OF THE PROJECT

This project is intended to design and implement a persistence of vision based display which uses movement to activate and to the detect the right moment to start redrawing the shown text. This display is based on a long bar, with an accelerometer on one end, and a strip of leds along part of its length. The electronic components are mounted on a printed circuit board, and driven by an Arduino based microcontroller, which includes a Bluetooth module. Allowing for the shown text to be changed by sending messages via Bluetooth.

This document goes over the main alternatives of persistence of vision based displays, mainly focusing on propeller displays. Then the differences from those are outlined. Finally, the various technologies used on the project are explained, in order to introduce some background.

The next part covers extensively the design and implementation of the device, beginning with an overview and moving on to a description of both the hardware and the software that compose the project.

The last part of the document includes all of the source code written specifically for the project.

ÍNDEX

1.INTRODUCCIÓ....3

2.ANTECEDENTS I COMPARATIVA....4

2.1.TECNOLOGIES UTILITZADES....4

2.1.1.Acceleròmetres....4

2.1.2.Bluetooth Low Energy....5

2.1.3.Microcontroladors....5

2.1.4.Plaques de circuit imprès....5

2.1.5.LED....6

2.1.6.Bus SPI....6

3.DISSENY I IMPLEMENTACIÓ....8

3.1.DESCRIPCIÓ GENERAL DEL DISSENY....8

3.2.HARDWARE....10

3.2.1.Acceleròmetre....10

3.2.2.Radino....11

3.2.3.Placa de circuit imprès....11

3.3.SOFTWARE....15

3.3.1.El mòdul accel....15

3.3.2.El mòdul comm....17

3.3.3.El mòdul leds....18

3.3.4.El mòdul timer....19

3.3.5.El programa principal....20

4.CONCLUSIONS....25

5.BIBLIOGRAFIA....26

6.ANNEXOS....28

6.1.TFG_ACCEL.CPP....28

6.2.TFG_ACCEL.H....29

6.3.PROVA_ACCEL.INO....29

6.4.PROVA_ACCEL.PY....31
6.5.TFG_COMM.CPP....32
6.6.TFG_COMM.H....45
6.7.PROVA_COMM.INO....46
6.8.MODIFICACIONS A HAL_ACI_TL.CPP....47
6.9.TFG_LEDS.CPP....48
6.10.TFG_LEDS.H....50
6.11.PROVA_LEDS.INO....51
6.12.TFG_TIMER.CPP....52
6.13.TFG_TIMER.H....53
6.14.PROVA_TIMER.INO....54
6.15.TFG.INO....54
6.16.CHARACTER_DEFS.H....59
6.17.PROVA_MAIN.PY....71

1. INTRODUCCIÓ

L'objectiu d'aquest projecte és la creació d'un display activat per moviment que, utilitzant persistència de visió, mostra a l'aire un text. El text mostrat s'ha de poder reprogramar fàcilment utilitzant una comunicació Bluetooth amb el dispositiu. El display consisteix d'una tira de leds que queden apagats fins que, al detectar moviment, s'encenen i s'apaguen per tal de mostrar un text per l'espai en que s'estan movent.

L'efecte ha de ser vistós i interessant a primer cop d'ull. La finalitat del dispositiu és simple, ser utilitzat com una demostració de les TIC, per exemple, com a exemple del tipus de coses interessants que es poden fer amb un microcontrolador i uns quants leds.

L'abast del projecte està limitat a fer un sol dispositiu funcional, però el mateix concepte en el que està basat es podria aplicar a un display molt més gran que podria reemplaçar un cartell lluminós quan s'ha de fer arribar un missatge a un nombre de persones des de una certa distància.

2. ANTECEDENTS I COMPARATIVA

L'idea d'un display que utilitza la persistència de visió per tal de mostrar una imatge o un text no és nova: hi ha més d'una versió feta per aficionats a l'electrònica, i fins i tot versions comercialitzades. La gran majoria són displays circulars, on una sola línia de leds es mou formant un cercle, a una velocitat constant, molt ràpidament. Aquests displays s'utilitzen moltes vegades com una mena de rellotge analògic a més de per mostrar textos. Normalment, s'utilitza un sensor per tal de detectar quan el display ha fet una volta complerta i començar a re-dibuixar la imatge. La tira de leds es mou utilitzant algun tipus de motor, a una velocitat constant.

Aquest projecte, però, no utilitza cap motor per moure's, i en el seu lloc el display s'activa quan un sensor de moviment, un acceleròmetre, detecta un moviment d'un costat a l'altre. Aquesta mesura d'acceleració s'utilitza, a més, per detectar els pics que corresponen amb el moment que l'aparell canvia de direcció. Cada cop que es detecta un pic, es re-dibuixa la imatge un sol cop. Cada pic dibuixa la imatge de forma invertida, de forma que tant quan s'està movent d'esquerra a dreta com de dreta a esquerra s'acaba dibuixant la mateixa imatge.

A més, al contrari que moltes altres aplicacions similars, es pot modificar molt fàcilment el missatge mostrat: És possible connectar-se al dispositiu mitjançant Bluetooth i, utilitzant una aplicació general disponible tant per iOS com per Android i altres plataformes, canviar el text que es mostra. Per tant, es pot canviar el text molt fàcilment, en qüestió de segons, utilitzant un telèfon mòbil.

2.1. Tecnologies utilitzades

En aquesta secció s'ofereix una explicació de cadascuna de les tecnologies implicades en el projecte.

2.1.1. Acceleròmetres

Un acceleròmetre és un aparell que mesura l'acceleració física. És a dir, l'acceleració experimentada per un objecte respecte a l'estat de caiguda lliure. Per tant, es mesura l'acceleració experimentada per la gravetat de la terra en tot moment.

L'acceleració es mesura a partir del desplaçament d'una massa respecte d'una posició neutral, que es converteix utilitzant un procés electromecànic microscòpic (MEMS, Microscopic ElectroMechanical System) en un senyal elèctric.

Aquest procés només percep l'acceleració en una sola direcció. Un acceleròmetre que mesuri l'acceleració en tres eixos de moviment, per tant, utilitzarà tres masses de les quals en mesurarà el desplaçament.

Un acceleròmetre sol tenir un ample de banda que indica la freqüència amb que es poden fer noves mesures i obtenir dades precises. [1][2]

2.1.2. Bluetooth Low Energy

Bluetooth Low Energy és una versió de Bluetooth orientada a dispositius que, com el seu nom indica, utilitzen poca energia. Les últimes versions de Bluetooth suporten BLE i, per tant, es pot utilitzar per comunicar-se amb gairebé qualsevol aparell que incorpori Bluetooth. En general, BLE està pensat per comunicar petits aparells entre ells i que intercanviïn dades i interactuïn sense cables. L'anomenat Internet de les coses. [3][4]

2.1.3. Microcontroladors

Un microcontrolador és un petit ordenador, en un sol circuit integrat que conté el processador, la memòria i perifèrics programables utilitzats per entrada i sortida. Els microcontrolador solen utilitzar-se en sistemes encastats, com comandaments a distància o joguines, per tal de controlar altres aparells.

Típicament, un microcontrolador utilitza com a dispositius d'entrada/sortida interruptors, leds, aparells de radiofreqüència, sensors de temperatura, acceleració... Normalment no tenen dispositius d'entrada que permetin comunicació directa amb humans. Per tal de comunicar-se amb aquests dispositius de forma ràpida i consistent es solen utilitzar interrupcions: Funcions que interrompen el programa principal per tal de gestionar comunicació amb perifèrics.

La memòria disponible en el xip per tal d'emmagatzemar programes és limitada, i per tant els programes solen compilar-se per tal d'ocupar la menor quantitat de memòria possible. Depenent en el tipus de memòria, diferents microcontroladors utilitzen diferents formes d'escriure i reescriure la memòria de programa per tal de ser programats. [5]

2.1.4. Plaques de circuit imprès

Una placa de circuit imprès o PCB (Printed Circuit Board) utilitza vies conductives per tal de suportar mecànicament components elèctrics en un circuit. S'utilitzen plaques de cobre laminades sobre un substrat no conductor. Cadascun dels components sol estar soldat sobre la placa. Una PCB pot tenir més d'una capa de vies conductores interconnectant cadascun dels components. Les plaques de circuit imprès són utilitzades per gairebé tots els productes electrònics.

Una PCB està formada per una capa central de substrat de fibra de vidre FR4. Aquest li dona rigidesa i gruix a la placa. A continuació s'aplica una capa de cobre. Es poden aplicar dues capes de cobre, una a sobre i una a sota de la capa rígids central, o fins i tot més capes intercalant-les amb més fibra de vidre. A continuació s'aplica una capa de màscara que aïlla el cobre de l'exterior i del contacte accidental amb altres conductors, deixant al descobert només el contactes on soldar. Finalment s'aplica una capa de serigrafia amb lletres, números i símbols per tal de veure més fàcilment la funcionalitat de cadascun dels pins de la placa. Una sola capa de cobre sobre un substrat és el mínim, i suficient per una placa implementant un circuit simple. [6][7]

2.1.5. LED

Un LED (Light Emitting Diode) és un díode que produeix llum quan un corrent passa a través seu. El díode és una junció P-N que, sota un cert voltatge entre els seus terminals, recombina la configuració dels seus electrons i causa una alliberació d'energia en forma de llum.

Els LEDs s'utilitzen molt sovint en aplicacions electròniques com a indicadors, ja que són petits i consumeixen poca energia. També han començat a utilitzar-se per il·luminació en general degut al menor consum i major esperança de vida en comparació amb els mètodes més tradicionals. [8]

2.1.6. Bus SPI

Un bus SPI (Serial Peripheral Interface) permet comunicació síncrona entre un mestre i un o més esclaus. Està pensat per a ser utilitzat en distàncies petites i en sistemes encastats. Tot i que el sistema va ser implementat per Motorola inicialment, s'ha convertit en un estàndard de facto, es a dir, que no té una implementació oficial. La Figura 1 mostra les connexions per tal de connectar un sol mestre i un sol esclau per SPI.

Figura 1: Comunicació SPI entre un sol mestre (SPI master) i un sol eslau (SPI slave).

Aquest bus utilitza tres línies per comunicar-se, a més d'un cable extra per a cada esclau. Per tant, un mínim de quatre línies, amb quatre per a cada esclau. El mestre controla la línia SCLK per marcar la temporització en les comunicacions. La línia MOSI (Master Out, Slave In) també està controlada pel mestre i s'utilitza per enviar dades a l'esclau. La línia MISO (Master In, Slave Out) és utilitzada per tots els esclaus per enviar dades al mestre. Per tal de fer això, aquesta línia es manté en alta impedància quan no es fa servir. Tan sols quan un

esclau és seleccionat, aquesta linia deixa l'estat d'alta impedància. Finalment, hi ha una línia SS (Slave Select) per cada esclau, que es manté a nivell alt. El mestre posa una sola línia SS a nivell baix quan vol comunicar-se amb aquell esclau.

Els detalls d'implementació, i fins i tot les abreviatures dels noms de les línies, canvien segons el dispositiu.

La transmissió de dades es fa de la següent forma: El mestre activa una línia SS amb un nivell lògic de 0 per tal de seleccionar un esclau, i inicia un senyal de clock a SCLK d'uns quants MHz. El mestre envia un byte de dades per la línia MOSI mentre que l'esclau envia un byte per la línia MISO. Dins d'això, però, hi ha diferències entre dispositius: Per exemple, el senyal de clock pot mirar l'estat de la línia MOSI/MISO quan hi ha un flanc de pujada o un flanc de baixada, i la línia de clock pot estar 'aturada' amb un 1 o amb un 0 lògic.

Un gran nombre de microcontroladors inclouen suport de hardware per a SPI i opcions per configurar aquesta interfície segons quina versió de l'estàndard segueix el dispositiu amb el que es vol comunicar. [9]

3. DISSENY I IMPLEMENTACIÓ

3.1. Descripció general del disseny

La part central del disseny és el Radino, que combina el microcontrolador ATmega32U4, el mateix que en l'Arduino Micro, amb un mòdul Bluetooth Low Energy. El Radino està connectat a un grup de leds que formen el display, i a un acceleròmetre que detecta el moviment de tot l'aparell.

Movent-ho tot d'un costat a l'altre, com es mostra en la Figura 2, es detecten màxims i mínims d'acceleració que senyalen que s'ha arribat a un extrem del moviment. Aquests inicien un procés que fa que es comenci a dibuixar la imatge a mostrar, encenent i apagant els leds estratègicament en els intervals de temps adequats. Així es crea l'efecte d'un text que es dibuixa a l'aire, com es pot veure en la Figura 3.

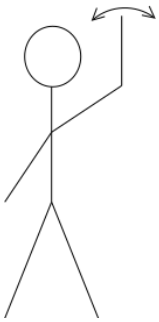


Figura 2: Moviment del display.

Per tant, l'acceleròmetre es situa a la punta per tal de detectar els canvis d'acceleració el millor possible. A continuació se situen els leds en línia, i tots els elements es connecten al Radino, que analitza les dades rebudes des de l'acceleròmetre i controla els leds.

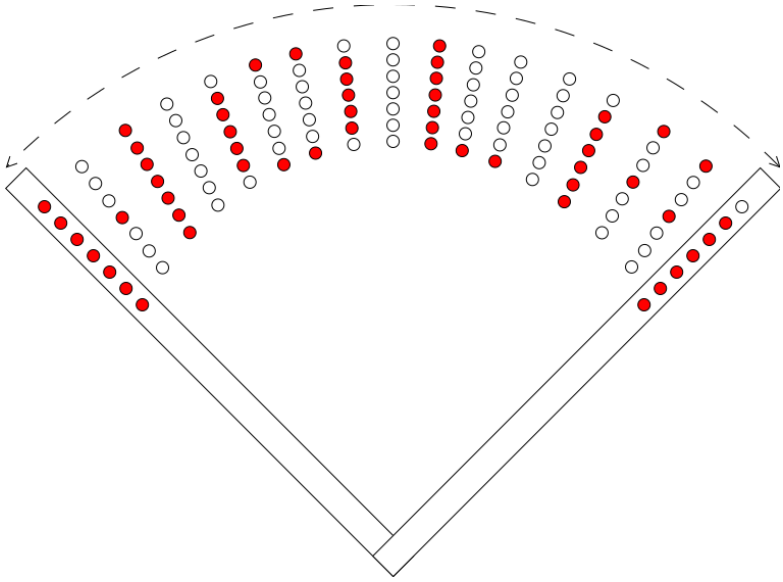


Figura 3: El moviment dels leds forma un text.

El text mostrat es guarda en la memòria del Radino com una imatge. Aquesta imatge es pot canviar utilitzant Bluetooth. També es pot utilitzar el port sèrie de forma idèntica a com s'utilitza el Bluetooth per canviar-la.

Quan es rep un caràcter, ja sigui des de la interfície Bluetooth com des de el port sèrie, es reescriu part de la imatge amb la representació assignada al caràcter, i es continua reescrivint segons es van rebent caràcters fins que s'acaba la transmissió (o en el cas del port sèrie, es rep un salt de línia).

Per tal de detectar extrems de moviment s'utilitza un acceleròmetre. Els extrems d'acceleració es poden considerar màxims i mínims de moviment. Si suposem que hi ha un

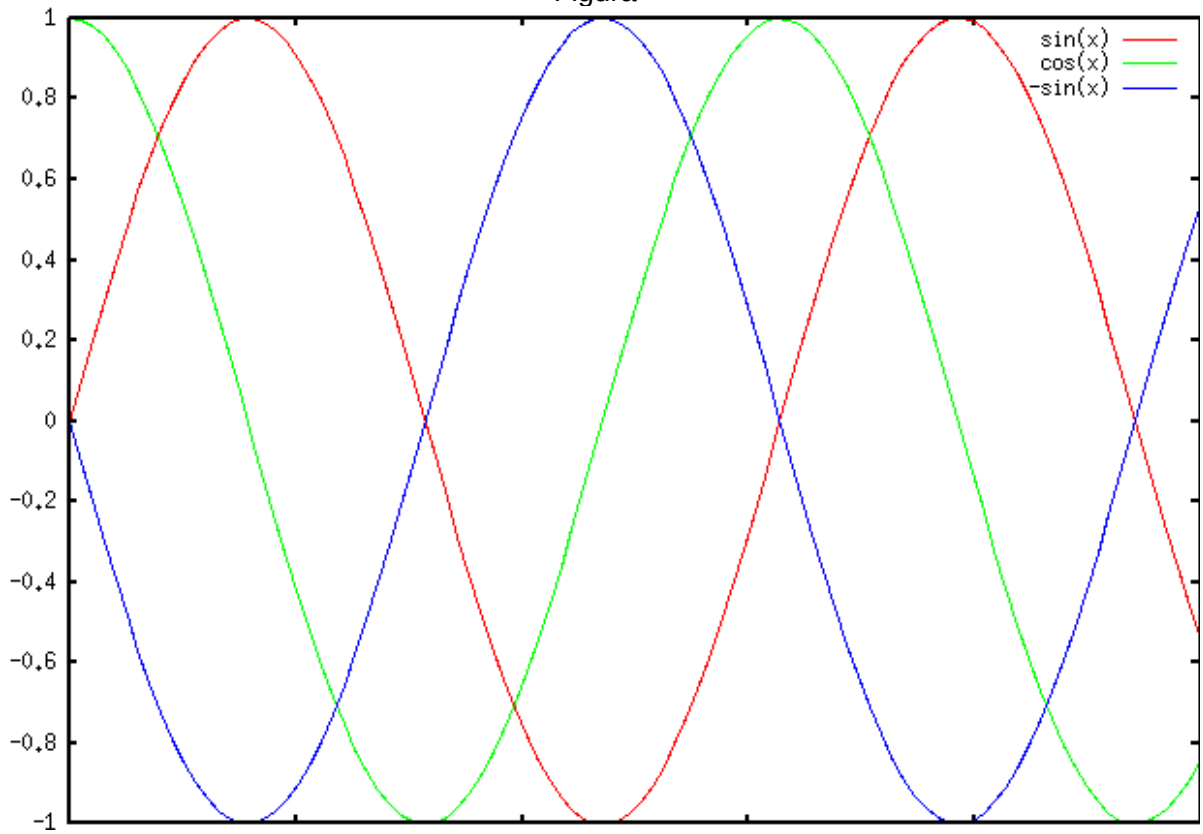
moviment més o menys sinusoïdal, l'acceleració és la doble derivada d'aquest moviment. Per tant:

$$x''(t) = v'(t) = a(t)$$

$$\sin(t) \rightarrow \cos(t) \rightarrow -\sin(t)$$

Els màxims i mínims d'acceleració es corresponen amb mínims i màxims de moviment, tal com es pot veure en la Figura 4.

Figura



4: Comparació entre espai/velocitat/acceleració sinusoidals

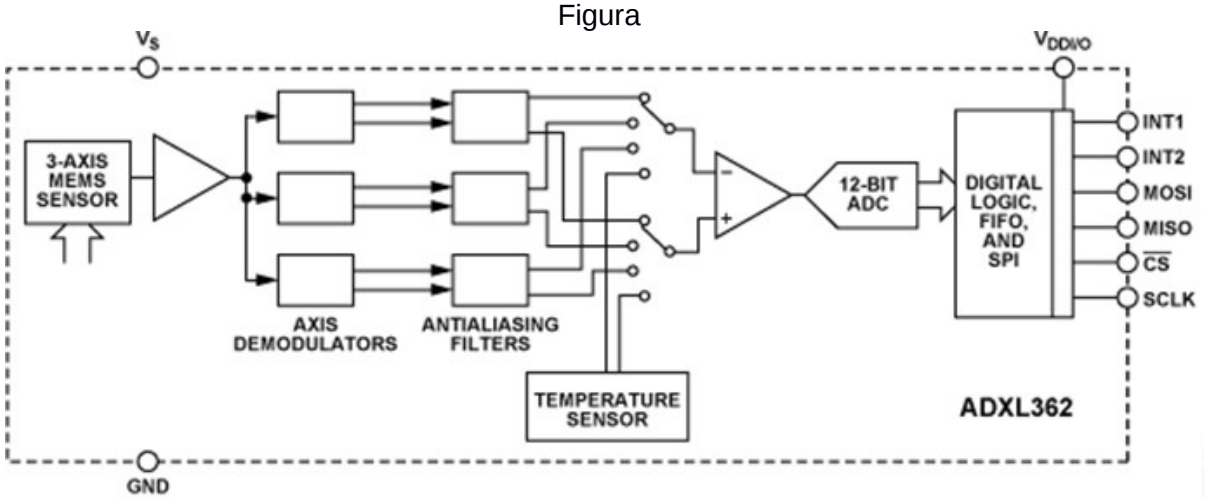
En la realitat, el moviment que es segueix no és del tot sinusoïdal, però és bastant similar. La Figura 9 i la Figura 12 en la secció 3.3 contenen una gràfica de les dades reals d'acceleració.

3.2. Hardware

3.2.1. Acceleròmetre

L'acceleròmetre utilitzat és l'ADXL362. Aquest, inclou acceleració en tres eixos: x, y i z i, a més, un sensor de temperatura. La mesura de temperatura, però, no s'arriba a utilitzar en el projecte. La Figura 5 mostra l'esquema de funcionament.

Figura



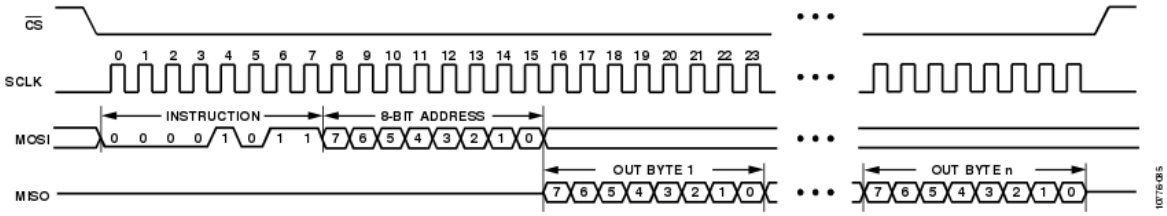
5: Esquema de funcionament de l'acceleròmetre

Aquest model d'acceleròmetre consumeix un corrent negligible de l'ordre de microamperes en el mode de mesura constant, i pot arribar a un corrent de nanoamperes (tot i que no s'arriba a optimitzar tant). L'ample de banda de l'acceleròmetre, la màxima freqüència de mostreig, és de 400 Hz. Es suporten rangs de mesura de ±2 g, ±4 g, i ±8 g. Es pot operar amb voltatges entre 1.6 i 3.5 V. Cada lectura té 12 bits de resolució.

S'inclou un FIFO i un seguit de registres en els que es pot escriure i dels que es pot llegir. La comunicació amb el dispositiu es fa a través d'SPI, actuant com a esclau, amb una velocitat de clock recomanada d'entre 1 i 5 MHz.

L'acceleròmetre rep una comanda del mestre, una adreça, i a continuació s'envien o es reben dades. Les tres comandes possibles són: escriure a un registre, llegir d'un registre o llegir el FIFO. Per tal de llegir dades d'acceleració tan sols cal fer unes escriptures inicials per configurar-lo i a continuació una lectura periòdica d'un registre concret que dona com a resultat l'enviament de 8 bytes corresponent a els tres eixos d'acceleració i una mesura de temperatura. Aquesta lectura es la recomanada per tal de que la lectura dels tres eixos x, y i z sigui concurrent i complerta. La Figura 6 mostra una d'aquestes lectures en 'burst' per SPI. [10][11]

Figura



6: Lectura en burst per SPI

3.2.2. Radino

El microcontrolador al centre del disseny és el Radino, un xip ATmega16U4 (el mateix que hi ha en un Arduino micro) amb un mòdul Bluetooth Low Energy, connectats mitjançant un bus SPI amb algunes línies extra que no formen part del protocol SPI més comú. Els detalls complexes de la comunicació en si venen donats en la forma d'una llibreria per l'IDE d'Arduino. El Radino és, per tant, bàsicament un Arduino connectat per SPI a un mòdul Bluetooth.

L'ATmega16U4 és un microcontrolador AVR de vuit bits, clock de 16 MHz amb un seguit de característiques i perifèrics propis d'un microcontrolador; com suport hardware per SPI, varis timers, una interfície USB, UART, interrupcions... A més de pins de sortida i entrada digital o analògica.

Tot el disseny es controla amb aquest microcontrolador, programat en C i C++ utilitzant l'IDE d'Arduino. S'utiliza aquesta degut a que la llibreria que controla el mòdul Bluetooth està pensada per utilitzar-se amb aquesta IDE i tot el seu entorn. A més de la llibreria també s'inclouen definicions del hardware per tal que l'IDE pugui programar el Radino.[12][13]

3.2.3. Placa de circuit imprès

L'acceleròmetre i els leds estan connectats en una placa de circuit imprès. Des de aquesta placa i a través de dos cables, les 13 línies utilitzades (7 leds + 4 línies SPI + GND + VCC) es troben amb el Radino.

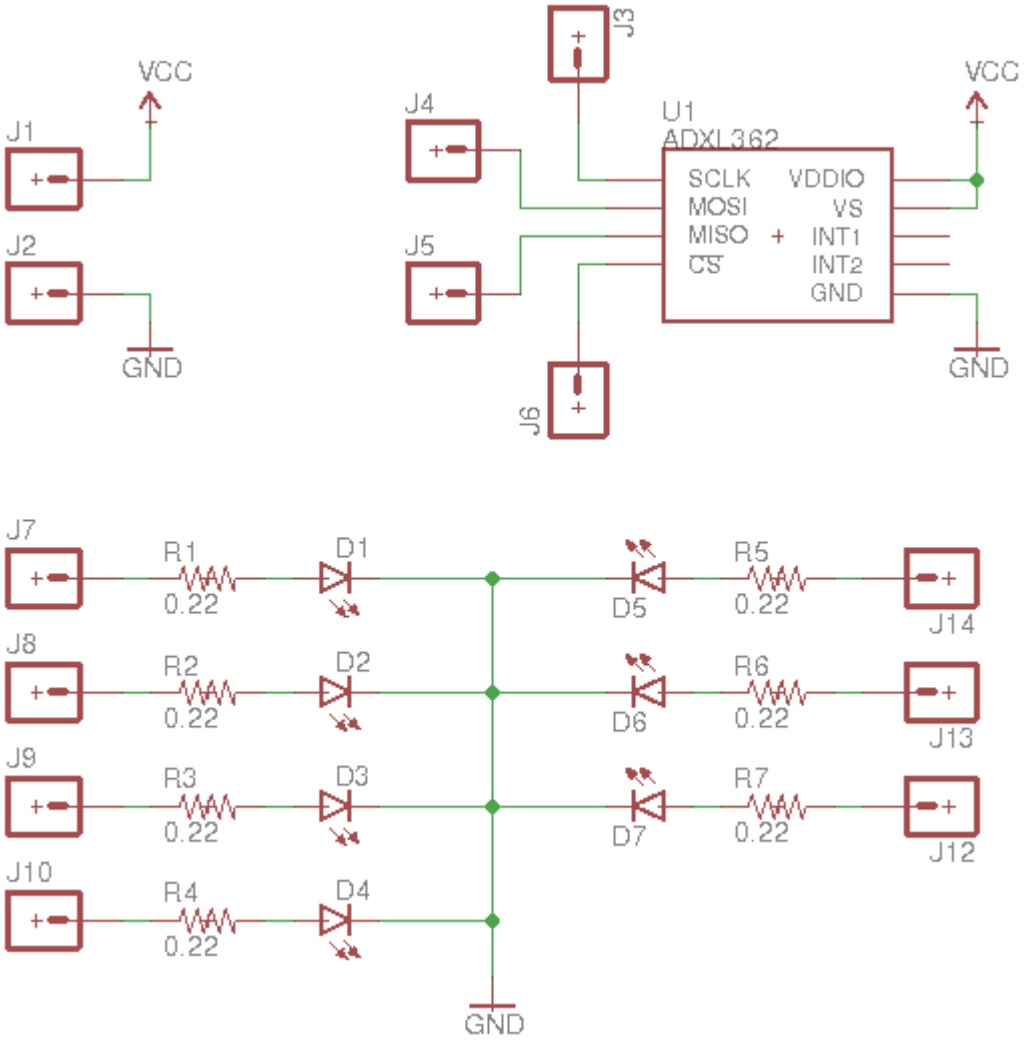
El disseny de la placa s'ha fet amb la versió gratuïta d'Eagle. Els leds estan situats en línia i centrats, amb l'acceleròmetre a la punta. L'esquema del circuit de la PCB es pot veure a la Figura 7.

Les línies SCLK, MOSI, MISO i CS de l'ADXL362 estan connectades directament a l'exterior, a un cable que acaba en el Radino. A més les línies VDDIO i VS, l'alimentació, també acaben connectades directament al Radino. GND es comparteix amb tots els leds i, també, es connecta al Radino.

Cadascun dels leds es connecta a una resistència per tal de que s'utilitzi un corrent especific i conegut suficient per que el led s'encengui, i connectat a un pin específic del Radino per un terminal i a GND per l'altre.

El disseny de la PCB es mostra a la Figura 8. Tot i que no és visible per claredat, la majoria de la placa és un pla de massa que es correspon amb el GND i, per tant, interconnecta els leds, l'acceleròmetre i el Radino. L'acceleròmetre es connecta mitjançant vies amb els forats on s'acaben connectant els cables que arribaran al Radino. Pel que fa als leds, cadascun d'ells està connectat a un espai per un cable a través d'un espai per la resistència, i a l'altre extrem a l'espai buit que es correspon al pla de massa. També es pot veure que hi ha un espai per dos forats. Aquest espai s'utilitzarà per assegurar el cable i que no es tibi de les soldadures dels cables a la PCB.

Figura



7: Esquema del circuit de la placa de circuit imprès

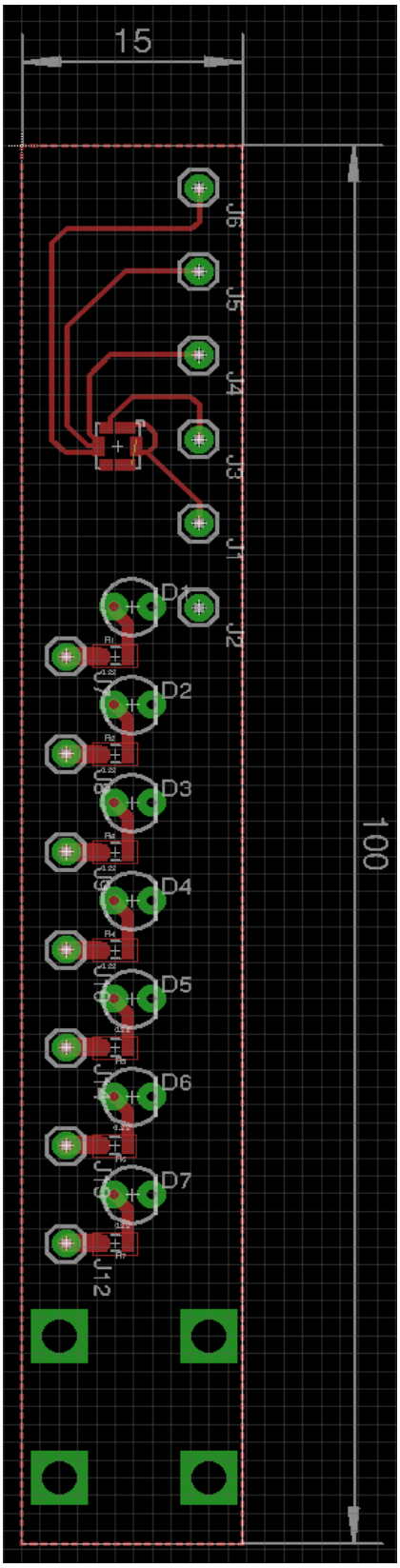


Figura 8: Disseny del circuit de la placa de circuit imprès

3.3. Software

El microcontrolador s'ha programat utilitzant l'IDE d'Arduino. Les definicions de hardware i un seguit de llibreries són necessaris per tal de programar i d'utilitzar les funcions de Bluetooth de la placa Radino. Aquestes es poden trobar al wiki d'In-Circuit [14].

Pel que fa a l'acceleròmetre ADXL362, s'ha utilitzat una simple llibreria que aprofita la llibreria SPI de l'IDE. La llibreria es pot trobar a GitHub [15].

Pel que fa a la resta del codi, aquest s'ha dividit en quatre mòduls: accel, comm, leds i timer; a més d'un programa principal que utilitza les funcions que cadascun d'aquests mòduls implementen. Cada mòdul inclou una petit sketch de prova.

A continuació es descriuen els mòduls i les funcions que implementen.

3.3.1. El mòdul accel

Aquest mòdul utilitza funcions de la llibreria de l'ADXL362 per tal d'implementar un seguit de funcions que automatitzen millor la tasca que l'acceleròmetre fa en el projecte.

La implementació de la llibreria original és orientada a objectes. El codi d'aquest mòdul inicialitza una sola instància de l'objecte que representa l'acceleròmetre, i ofereix les funcions que efectuen crides als mètodes d'aquesta instància.

Aquestes són les funcions incloses en el mòdul:

- accel_init: Inicialitza l'acceleròmetre. Efectua una crida als mètodes begin i beginMeasure. Aquestes funcions inicialitzen la interfície SPI i defineixen un pin a utilitzar com a Slave Select, després fan que l'acceleròmetre faci un reset i inicien el mode de mesura.

- accel_filter: Aquesta funció no interactua amb l'acceleròmetre. Implementa un filtre simple per tal de treure part del soroll de les mesures de l'acceleròmetre. La funció retorna el següent valor del filtre quan s'entra el nou valor a filtrar. El filtre segueix la formula $y_n = \alpha \cdot x + (1-\alpha) \cdot y_{n-1}$ on y és el valor calculat pel filtre, n és la mostra numero n del filtre, x és el valor entrat al filtre i α és un valor constant entre 0 i 1 que està definit en el header del mòdul. El primer valor de y és 0. El càlcul es fa amb tipus float. Utilitzar enters per representar decimals amb coma fixa és una possible optimització que no ha sigut necessària. Tant la entrada com el valor retornat per la funció són enters, però la variable interna on es guarda l'ultim valor calculat és de tipus float.

- accel_raw: Llegeix dades de l'acceleròmetre. Pren com a paràmetre un punter a un array d'enters, i guarda els valors d'acceleració pels eixos x, y i z en els index 0, 1 i 2 de l'array. Utilitza el mètode readXYZTData de l'acceleròmetre per tal de llegir en una sola petició dades sobre els tres eixos i de temperatura. La dada de temperatura es descarta.

- accel_next: Aquesta funció utilitza accel_filter i accel_raw. De fet, aquesta és la única funció que es crida en el programa principal, les altres dues existeixen per separat per veure el funcionament del filtre. Aquesta funció pren la següent mesura del filtre, obté un sol valor d'acceleració a partir dels tres eixos, i utilitza el filtre per retornar un sol valor d'acceleració. Per tal d'obtenir un sol valor d'acceleració es sumen els valors dels tres eixos. Això no resulta en un valor vàlid d'acceleració, però ja que només es necessita el valor per trobar màxims i mínims, sumar els tres eixos és suficient. Per tant, es podria dir que aquesta funció no retorna un valor d'acceleració, només la suma dels tres eixos filtrada.

Per assegurar que les comunicacions per SPI són atòmiques, quan es crida un dels mètodes de l'acceleròmetre, es crida noInterrupts i interrupts per tal de desactivar les interrupcions durant la crida i tornar-les a activar després.

El mòdul accel inclou un programa d'exemple per tal de comprovar el seu funcionament. Aquest programa de prova du a terme el mateix procés que la funcion accel_next: crida a accel_raw per tal d'obtenir dades d'acceleració, suma els valors dels tres eixos i aplica accel_filter per tal d'obtenir un valor filtrar.

La diferència és que aquest programa de prova envia pel port sèrie, junt amb un temps en milisegons, la suma dels eixos d'acceleració abans i després de ser filtrada. Això és útil per poder veure l'efecte del filtre en les mostres d'acceleració.

Es pot utilitzar un programa per fer gràfiques a partir de conjunts de valors o, com és el cas en aquest projecte, un script de Python que utilitza matplotlib i llegeix del port sèrie per fer una gràfica de les dades. Aquest script està disponible junt amb la resta del codi.

La Figura 9 mostra el resultat de fer una prova amb aquest programa d'exemple, i utilitzar el script de Python per tal de visualitzar les dades.

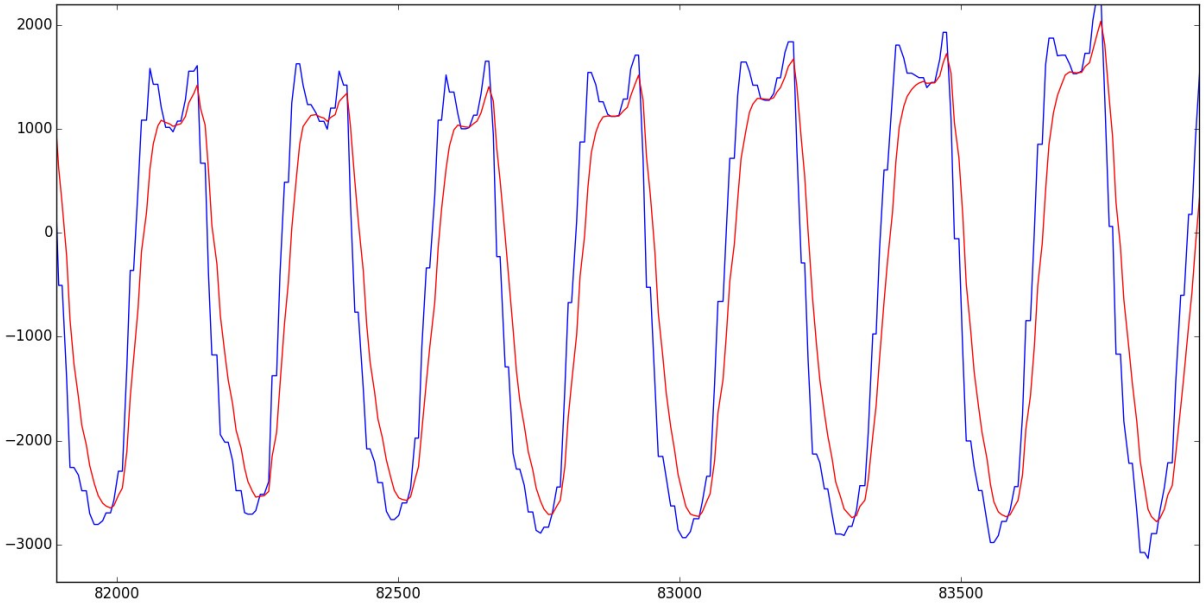


Figura 9: Gràfica de les dades d'acceleració, ampliada per veure-les en detall. La linia blava és l'acceleració sense filtrar, mentre que la vermella està filtrada.

Aquesta gràfica permet veure clarament com, tot i que hi ha un retard perceptible entre l'acceleració abans i després de ser filtrada, un cop ha passat pel filtre queda molt més llisa. Tot i el filtre, però, la forma d'U que es pot veure en els màxims persisteix. Això no és gaire important, ja que depèn de la forma en que es mou el display.

El filtrat fa més fàcil la detecció dels màxims i mínims. En la secció 3.3.5 es parla sobre la màquina d'estats que detecta els extrems d'acceleració a partir de les dades filtrades.

La primera opció que es va explorar alhora d'implementar el filtre va ser un filtre de mitjana mòbil: Aquest tipus de filtre manté en memòria un historial de dades i, segons s'afegeixen noves mostres, la mostra més antiga és substituïda amb la mostra més nova. Per tal de calcular el valor filtrat corresponent, es sumen totes les mostres acumulades i es divideix el resultat pel nombre de mostres.

El filtre de mitjana mòbil, però, introdueix un retard bastant gran, a més d'utilitzar bastanta memòria. En el seu lloc s'ha acabat utilitzant un filtrat exponencial [16] que acaba resultat en menys retard, menys ús de memòria, i la capacitat de parametritzar la quantitat de filtratge/retard amb una sola variable.

3.3.2. El mòdul comm

Aquest mòdul modifica l'exemple donat per la llibreria del Radino, per tal d'implementar recepció de missatges via Bluetooth i via port sèrie, ambdós de forma asíncrona mitjançant un sol callback.

El mòdul no distingeix entre bytes rebuts pel port sèrie o per Bluetooth. Aquesta distinció no és necessària, ja que la recepció de caràcters per qualsevol de les dues interfícies ha de tenir el mateix efecte. Tampoc ofereix funcions per enviar bytes pel port sèrie o per Bluetooth, ja que no hi ha cap necessitat d'enviar res per Bluetooth, i una funció per enviar caràcters pel port sèrie seria una versió menys flexible del mètode Serial.print d'Arduino.

Aquest mòdul implementa dues funcions:

- comm_init: Inicialitza la comunicació per Bluetooth. Aquesta funció pren com a paràmetre dues funcions: Un callback que es crida cada cop que es rep un caràcter per Bluetooth o pel port sèrie; i un callback que es crida quan s'han acabat de rebre caràcters. Aquesta última funció es crida quan es rep un salt de línia pel port sèrie i quan s'han acabat de rebre caràcters per Bluetooth.

- comm_loop: Executa una sèrie de funcions per tal de gestionar els events de la llibreria Bluetooth, cridant els callbacks si és necessari.

Per tal de que l'acceleròmetre i el hardware Bluetooth utilitzin una configuració compatible de l'SPI, també s'ha hagut de modificar una petita part de la llibreria Bluetooth: El mòdul Bluetooth envia els bits de cada byte per SPI començaran per el LSB, mentre que l'acceleròmetre comença pel MSB. És a dir, que mentre el Bluetooth rebria o enviaria '00110011', l'acceleròmetre rebria o enviaria '11001100'.

La llibreria Bluetooth, però, inclou funcions que permeten rebre les dades en el mateix ordre que l'acceleròmetre i un cop rebudes són invertides. I igualment inverteix les dades abans d'enviar-les. Per tant s'ha afegit i activat un flag de més i s'ha modificat el codi per que utilitzi aquestes funcions quan aquest flag s'activa, a més de quan detecta que s'està utilitzant un microcontrolador que no suporta el mode de SPI que espera comunicar-se començant a enviar el LSB.

El mòdul inclou un petit programa d'exemple i de prova que mostra pel port sèrie els caràcters rebuts per Bluetooth, i un salt de línia quan s'acaba la transmissió. Al contrari que el programa de prova del mòdul accel, aquest no té cap altra utilitat que comprovar el funcionament correcte del mòdul.

3.3.3. El mòdul leds

Aquest mòdul implementa funcions per gestionar la imatge que es guarda en memòria i per mostrar-la apagant i encenent els leds.

Quan es reinicia el microcontrolador es carrega una imatge per defecte, que es pot canviar en qualsevol moment. La imatge es guarda com un array d'enters de vuit bits. Cada enter representa una columna, és a dir, un estat dels leds segons van canviant amb el moviment.

El mòdul implementa les següents funcions:

- leds_init: Inicialitza el mòdul, configurant els pins que s'han de connectar als leds com a sortides i assegurant que estan apagats.
- leds_load_col: Canvia la columna de la imatge en memòria corresponent a l'índex donat pel byte donat.
- leds_show_col: Encén i apaga els leds per tal de que mostrin la columna corresponent a l'índex donat.
- leds_off: Apaga tots els leds.

Per tal d'encendre els leds a partir d'un byte indicant el seu estat, el codi comprova el valor de cada bit, des de el bit menys significatiu fins al bit setè bit. El nombre de bits utilitzats es pot canviar redefinint una constant en el header del mòdul. S'utilitzen set bits en comptes de vuit perquè aquest valor queda millor visualment alhora de dibuixar un caràcter.

Per tal de canviar l'estat dels pins, s'acaba cridant la funció interna set_led, que associa números del 1 al 7 amb els pins del hardware corresponents.

Al igual que el mòdul comm, el programa de test d'aquest mòdul no fa res especial, i és útil principalment per comprovar el funcionament correcte del mòdul i, si es fa alguna modificació, comprovar que no ha tingut efectes inesperats. Aquest programa de test canvia la imatge en memòria per tal que sigui un patró de leds fàcil de reconèixer, va mostrant cadascuna de les columnes cada poques dècimes de segon, per tal que es vegi el patró.

3.3.4. El mòdul timer

Aquest mòdul implementa un servei de timer sense utilitzar directament el timer de hardware, ja que no hi ha accés directe a aquest quan s'utilitza l'IDE d'Arduino. En el seu lloc s'utilitza la funció millis, que retorna el nombre de milisegons des de l'inici de la execució del programa. És a dir, és un servei de timer basat en polling.

El mòdul es centra en una funció pensada per que es cridi contínuament. Aquesta funció comprova el temps que ha passat des de l'última crida a millis i crida les funcions registrades si ha passat el temps necessari, i si no descompta el tems que ha passat del temps que falta per que s'hagin de cridar. Un cop es crida la funció s'elimina de la llista i el seu lloc pot ser reutilitzat per una nova funció registrada.

El mòdul utilitza dos arrays: Un per guardar cadascuna de les funcions registrades per cridar, i un altre per mantenir el nombre de milisegons que queden per cridar cada funció.

S'implementen les següents funcions:

- timer_init: Inicialitza els arrays utilitzats pel mòdul, assegurant que no hi ha cap funció registrada per ser cridada.

- timer_after: Registra una funció per que es cridi al cap del nombre donat de milisegons, guardant-la en el primer espai buit de l'array de funcions a cridar. I els milisegons que ha de trigar en cridar-se en el mateix index d'un altre array.

- timer_loop: Aquesta funció està pensada per cridar-se cada poc temps. Implementa el servei de timer mitjançant polling, utilitzant la funció millis de l'IDE d'Arduino. En primer lloc calcula el temps passat des de l'ultima crida a la funció. A continuació, per a cada funció registrada (és a dir, que no és NULL) per a ser cridada al cap d'un cert temps, comprova si ja ha passat prou temps per ser cridada. Si és així la funció es crida i s'elimina de l'array. Si no, es resta la diferencia de temps calculada del temps restant per tornar a cridar la funció.

El programa de test d'aquest mòdul defineix una funció que mostra un punt pel port sèrie i, a continuació, es registra a ella mateixa per que es torni a cridar al cap d'un segon. La funció es crida un sol cop i, gràcies al timer, va mostrant un punt pel port sèrie cada segon.

3.3.5. El programa principal

El programa principal bàsicament inicialitza cadascun dels mòduls, registrant una sèrie de callbacks, i a continuació passa la resta del temps cridant la funció timer_loop.

Es registren les dues funcions com a callback a comm_init: on_byte_received i on_msg_end. Aquestes dues funcions tenen una variable global en comú, que manté l'índex de la columna de la imatge que s'hauria de continuar canviant.

La primera funció, on_byte_received, canvia les següents sis columnes de la imatge per tal de que es corresponguin amb el caràcter rebut, modificant la variable global. La Figura 10 mostra el mapa de caràcters utilitzat.

Figura 10: Mapa de caràcters utilitzat per traduir caràcters a imatges. L'espai i tots els altres caràcters es tradueixen a un espai buit.

La segona funció, on_msg_end, escriu zeros a la resta de columnes i reinicia la variable global al índex 0.

Per tant, amb aquestes dues funcions combinades, els caràcters rebuts es van escrivint a la imatge que es manté en memòria i, un cop s'ha acabat de rebre el missatge, la resta de la imatge queda en blanc.

A més, es registren dues funcions per ser cridades al cap d'un cert temps, i al final d'ambdues funcions es tornen a registrar a elles mateixes. Aquestes són comm_loop_timer,

que simplement crida comm_loop abans de registrar-se a ella mateixa un altre cop, i sm_loop, que és la funció que implementa la màquina d'estats que detecta màxims i mínims d'acceleració i, per tant, que acaba fent que els leds que formen el display s'activin a temps.

La màquina d'estats que implementa sm_loop utilitza accel_next en cada crida per rebre la següent dada de l'acceleròmetre ('val' en la Figura 11) . A partir d'aquesta dada, detecta màxims i mínims d'acceleració per tal de detectar màxims i mínims de posició, és a dir, per detectar els extrems del moviment de balanceig que es fa al moure l'aparell. La Figura 11 mostra el funcionament de la màquina d'estats.

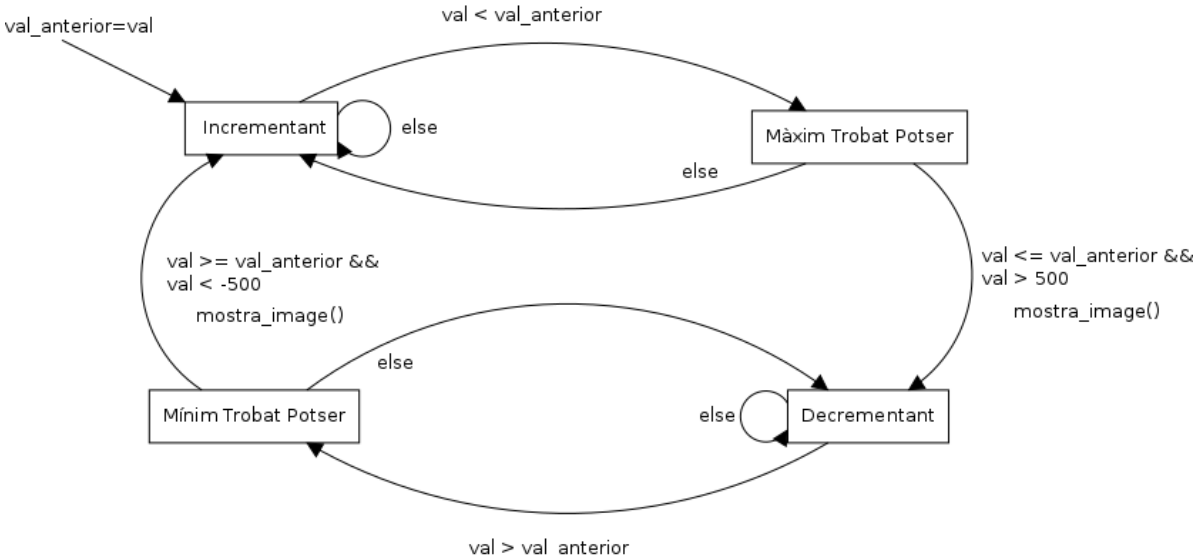


Figura 11: Màquina d'estats per detectar màxims i minims. Un màxim es detecta quan el valor donat comença a decrementar, sempre que sigui un valor prou alt. Un mínim es detecta de forma similar.

Quan es detecta un màxim o un mínim, es crida una funció que acaba mostrant la primera columna de la imatge guardada en memòria, i es registra en el timer per tal que es torni a cridar automàticament al cap d'uns pocs milisegons. La diferència entre un màxim i un mínim és l'ordre en el que es mostren les columnes, de esquerra a dreta o de dreta a esquerra, per tal que s'acabi mostrant la mateixa imatge en els dos casos. Aquesta funció es crida cada cinc milisegons, o amb una freqüència de de 200 Hz. La meitat de l'ample de banda de l'acceleròmetre.

Per tal de detectar extrems d'acceleració, la màquina d'estats comença esperant a trobar un valor anterior al valor passat. Així, assumint que havia començant incrementant, es determina que s'ha arribat a un màxim i es comença a disminuir. Per tal d'assegurar que és un màxim, es comprova que el valor en si sigui superior a una certa constant. Un cop s'ha trobat el màxim s'assumeix que a partir d'ara el valor d'acceleració començarà a decreixer, fins que es trobi un mínim i torni a incrementar. El mateix procés es segueix per detectar el mínim que hauria de venir a continuació.

Si no s'activa un flag de compilació per tal de desactivar-ho, la màquina d'estats mostra informació pel port sèrie per tal de veure que passa durant la seva execució: Aquesta informació és semblant a la del programa de prova del mòdul accel. Cada cop que s'executa un pas de la màquina d'estats es mostra una línia pel port sèrie. S'inclou: un nombre de milisegons (el temps des de que s'ha iniciat el programa), el valor d'acceleració passat a la màquina d'estats (ja filtrat) i un valor que dona informació sobre l'estat actual del sistema, i que pot tenir un d'aquests significats:

- Incrementant: La màquina d'estats està en l'estat Incrementant
- Decrementant: La màquina d'estats està en l'estat Decrementant
- Determinant Màxim: Estava en l'estat Màxim Trobat Potser, però no s'ha determinat com a màxim i s'ha tornat a l'estat Incrementant.
- Determinant Mínim: Estava en l'estat Mínim Trobat Potser, però no s'ha determinat com a mínim i s'ha tornat a l'estat Decrementant.
- Màxim: Des de l'estat Màxim Trobat Potser, s'ha trobat el màxim i s'ha començat a mostrar la imatge.
- Mínim: Des de l'estat Mínim Trobat Potser, s'ha trobat el mínim i s'ha començat a mostrar la imatge.

La Figura 12 mostra l'output del programa principal, passat per un script de Python per fer una gràfica a partir de les dades.

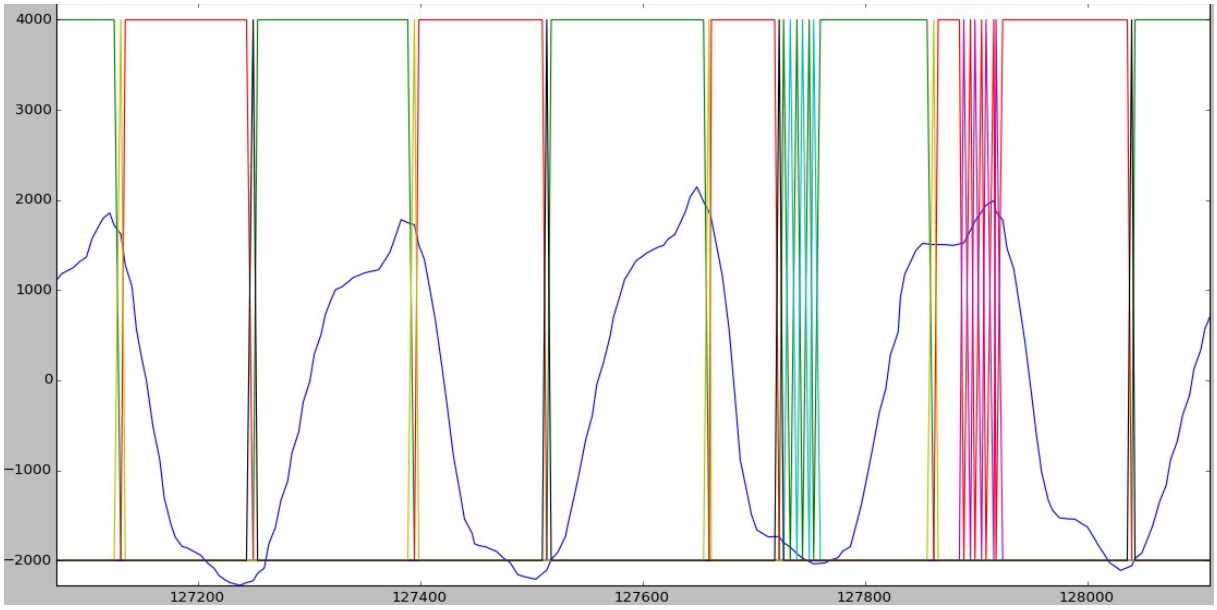


Figura 12: Gràfica feta a partir de l'output del programa principal, mostrant les transicions d'estats.

La línia blava mostra els canvis d'acceleració. Un sol pic groc representa un màxim, mentre que un sol pic negre representa un mínim. Les zones vermelles representen l'acceleració decreixent, i les verdes incrementant. Els pics blau clar i lila són falsos positius que s'han evitat, i sempre van acompanyats d'uns quants pics verds o vermells. Si es mira el primer cas d'un fals positiu evitat, es pot veure que es detecta un mínim però, en comptes d'incrementar, l'acceleració passa unes quantes mostres en que segueix decreixent. Com que la màquina d'estats assumeix que l'acceleració hauria d'estar incrementant, immediatament creu que s'ha arribat a un màxim. Però al seguir amb valors baixos d'acceleració, no comença a mostrar la imatge corresponent. Al cap d'unes mostres l'acceleració comença a incrementar un altre cop.

En el cas del màxim es pot veure com arriba a un màxim, disminueix durant un temps, i després continua incrementant, generant falsos positius, fins a començar a disminuir cap al mínim.

4. CONCLUSIONS

El funcionament del projecte és bastant satisfactori: La impressió que fa al ser vist per primer cop és el que es buscava inicialment: és vistós i interessant a primer cop d'ull. Per tant, pot ser útil per ensenyar-lo i provocar interès en les TIC en general.

Per tal de dissenyar i implementar el projecte, s'han hagut d'utilitzar molts dels conceptes apresos durant el grau. Per mencionar-ne només alguns: disseny de plaques de circuit imprès, programació en baix nivell de microcontroladors, electrònica... També hi ha hagut temes que s'han hagut de revisar, com el disseny de PCBs.

Hi ha coses que es podrien millorar: El disseny general és bastant simple, tot i que fa tot el que hauria de fer. A més, exteriorment tampoc és res extraordinari.

Una altra línia per la que es podria ampliar el projecte és aplicar el mateix disseny i el mateix concepte en diferents formes, per exemple en un cartell lluminós fàcil de desplaçar i que només necessita moure's per funcionar. Per exemple, en el cas en que es vulgui mostrar un missatge més o menys complexe sense haver de cridar-lo. Una versió més gran amb llums potents podria transmetre un missatge a qualsevol que el mirés, i si es podria canviar fàcilment per Bluetooth amb un smartphone (afegint un mecanisme de seguretat, com una contrasenya, per que no qualsevol el pugui canviar), els qual son omnipresents avui en dia.

5. BIBLIOGRAFIA

1) *Wikipedia. Accelerometer.* Octubre 2016.
https://en.wikipedia.org/wiki/Accelerometer

2) *DimensionEngineering. Beginner's guide to accelerometers.* Octubre 2016.
http://www.dimensionengineering.com/info/accelerometers

3) *Wikipedia. Bluetooth Low Energy.* Octubre 2016.
https://en.wikipedia.org/wiki/Bluetooth_low_energy

4) *Bluetooth technology website. Bluetooth Low Energy.* Octubre 2016.
https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics/low-energy

5) *Wikipedia. Microcontroller.* Octubre 2016.
https://en.wikipedia.org/wiki/Microcontroller

6) *Wikipedia. Printed Circuit Board.* Octubre 2016.
https://en.wikipedia.org/wiki/Printed_circuit_board

7) *Sparkfun. PCB Basics.* Octubre 2016.
https://learn.sparkfun.com/tutorials/pcb-basics

8) *Wikipedia. Light-Emitting Diode.* Octubre 2016.
https://en.wikipedia.org/wiki/Light-emitting_diode

9) *Wikipedia. Serial Peripheral Interface Bus*. Octubre 2016.
https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

10) *Analog Devices. ADXL362 Datasheet and Product Info.* Octubre 2016.
https://www.sparkfun.com/products/11446

11) *Analog Devices. ADXL362 Datasheet.* Octubre 2016.
http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL362.pdf

12) *In-Circuit Online Shop. Radino nRF8001.* Octubre 2016.
http://shop.in-circuit.de/product_info.php?products_id=31

13) *Atmel. Datasheet ATmega16U4.* Octubre 2016.
http://www.atmel.com/Images/Atmel-7766-8-bit-AVR-ATmega16U4-32U4_Datasheet.pdf

14) *In-Circuit Wiki. Radino Software.* Octubre 2016.
http://wiki.in-circuit.de/index.php5?title=radino_software

15) *GitHub. ADXL362 by anem.* Octubre 2016.
http://annem.github.io/ADXL362/

16) *MegunoLink,* Paul Martinsen*. Three Methods to Filter Noisy Arduino Measurements.* Octubre 2016.
http://www.megunolink.com/articles/3-methods-filter-noisy-arduino-measurements/

6. ANNEXOS

A continuació s'inclou el codi amb el que s'ha programat el microcontrolador. No s'inclouen llibreries de tercers, excepte seccions editades s'un sol fitxer d'una d'aquestes llibreries.

També hi ha dos scripts que automatitzen el procés de llegir dades del port sèrie i convertir-les en una gràfica.

6.1. tfg_accel.cpp

Codi font del mòdul accel.

```
1   #include <ADXL362.h>
2   #include "Arduino.h"
3
4   #include "tfg_accel.h"
5
6   static ADXL362 accel;
7
8   void accel_init(void) {
9     noInterrupts();
10    accel.begin(6);
11    accel.beginMeasure();
12    interrupts();
13  }
14
15  int16_t accel_filter(int16_t input) {
16    static float y_prev = 0;
17    y_prev = ((float)(ACCEL_WEIGHT))*((float)input) + (1-((float)
      (ACCEL_WEIGHT)))*y_prev;
18    return (int16_t)y_prev;
19  }
20
21  int16_t *accel_raw(int16_t *dades) {
22    int16_t x,y,z,t;
23
24    noInterrupts();
25    accel.readXYZTData(x,y,z,t);
26    interrupts();
27
28    dades[0] = x;
```

```
29                dades[1] = y;
30                dades[2] = z;
31                return dades;
32              }
33
34              int16_t accel_next(void) {
35                int16_t vals[3];
36                int16_t val;
37
38                accel_raw(vals);
39                val = vals[0]+vals[1]+vals[2];
40
41                return accel_filter(val);
42              }
```

6.2. tfg_accel.h

Header del mòdul accel.

```
1               #ifndef TFG_ACCEL_H
2               #define TFG_ACCEL_H
3
4               #define ACCEL_WEIGHT 0.3f
5
6               /* Inicialitza l'accelerometre */
7               void accel_init(void);
8
9               /* Aplica un filtre a la entrada. Retorna la sortida amb un cert retard. */
10              int16_t accel_filter(int16_t input);
11
12              /* Pren dades de l'accelerometre: Els tres eixos x,y,z */
13              int16_t *accel_raw(int16_t *dades);
14
15              /* Pren una dada de l'accelerometre, la filtra i la retorna. Tot en un pas. */
16              int16_t accel_next(void);
17
18              #endif
```

6.3. prova_accel.ino

Exemple/test del mòdul accel.

```
1  /*
2   * Versio antiga del IDE: si una llibreria utilitza d'altres, s'han
3   * d'incloure totes les llibreries en l'sketch principal.
4   */
5  #include <ADXL362.h>
6  #include <SPI.h>
7  /**********************/
8
9  #include <tfg_accel.h>
10
11 //#define NO_SERIAL_OUTPUT
12
13 void setup() {
14   pinMode(13, OUTPUT);
15   Serial.begin(57600);
16   accel_init();
17 }
18
19 void loop() {
20   int16_t vals[3];
21   int16_t val;
22
23   accel_raw(vals);
24   #ifndef NO_SERIAL_OUTPUT
25   Serial.print(millis());
26   Serial.print(' ');
27   #endif
28   val = vals[0] + vals[1] + vals[2];
29   if (val > -1000 && val < -500) {
30     digitalWrite(13, HIGH);
31   } else {
32     digitalWrite(13, LOW);
33   }
34   #ifndef NO_SERIAL_OUTPUT
35   Serial.print(val);
36   #endif
37
38   val = accel_filter(val);
39   #ifndef NO_SERIAL_OUTPUT
40   Serial.print(' ');
41   Serial.println(val);
42   #endif
43
44   delay(5);
```

```
45          }
```

6.4. prova_accel.py

Script per llegir l'output de prova_accel.ino i mostrar una gràfica.

```
1           import serial
2           import time
3           import matplotlib.pyplot as plt
4           from troba_tty import troba_tty
5
6           print "Fent proves..."
7
8           arduino = serial.Serial(port=troba_tty(),
9                                   baudrate=57600,#38400,
10                                  bytesize=8,
11                                  parity=serial.PARITY_NONE,
12                                  stopbits=1,
13                                  timeout=1)
14          time.sleep(2)
15
16          mostra_n = []
17          original = []
18          filtrat = []
19          while True:
20            try:
21              rcvd = arduino.readline()
22              try:
23                time,rcvd_orig,rcvd_filt = [int(n) for n in rcvd.strip().split()]
24              except:
25                print repr(rcvd)
26                continue
27
28              print time,rcvd_orig,rcvd_filt
29
30              mostra_n.append(time)
31              original.append(rcvd_orig)
32              filtrat.append(rcvd_filt)
33            except KeyboardInterrupt:
34              break
35
36          arduino.close()
37
```

```
38          plt.plot(mostra_n,original,'-b', mostra_n,filtrat,'-r')
39          plt.show()
```

6.5. tfg_comm.cpp

Codi font del mòdul comm. Basat en gran part en el codi d'exemple donat per la llibreria Bluetooth de Radino.

```
1           /* Copyright (c) 2014, Nordic Semiconductor ASA
2            *
3            * Permission is hereby granted, free of charge, to any person obtaining a copy
4            * of this software and associated documentation files (the "Software"), to deal
5            * in the Software without restriction, including without limitation the rights
6            * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
7            * copies of the Software, and to permit persons to whom the Software is
8            * furnished to do so, subject to the following conditions:
9            *
10           * The above copyright notice and this permission notice shall be included in all
11           * copies or substantial portions of the Software.
12           *
13           * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY
             KIND, EXPRESS OR
14           * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
             MERCHANTABILITY,
15           * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO
             EVENT SHALL THE
16           * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
             DAMAGES OR OTHER
17           * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
             OTHERWISE, ARISING FROM,
18           * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
             OTHER DEALINGS IN THE
19           * SOFTWARE.
20           */
21
22          #define TFGCOMM_DEBUG
23
24          #include "Arduino.h"
25
26          /*
27           * Aquest codi consisteix en parts del codi d'exemple de communicació amb
28           * Bluetooth/Serial.
29           */
```

```
30
31              #include <SPI.h>
32              #include <lib_aci.h>
33              #include <aci_setup.h>
34              #include "uart_over_ble.h"
35
36              /* Put the nRF8001 setup in the RAM of the nRF8001. */
37              #include "services.h"
38
39              #include "tfg_comm.h"
40
41              #ifdef SERVICES_PIPE_TYPE_MAPPING_CONTENT
42                 static services_pipe_type_mapping_t
43                    services_pipe_type_mapping[NUMBER_OF_PIPES] =
                SERVICES_PIPE_TYPE_MAPPING_CONTENT;
44              #else
45                 #define NUMBER_OF_PIPES 0
46                 static services_pipe_type_mapping_t * services_pipe_type_mapping =
                NULL;
47              #endif
48
49              /* Store the setup for the nRF8001 in the flash of the AVR to save on RAM */
50              static const hal_aci_data_t setup_msgs[NB_SETUP_MESSAGES] PROGMEM =
                SETUP_MESSAGES_CONTENT;
51
52              /*
53               * aci_struct that will contain
54               * total initial credits
55               * current credit
56               * current state of the aci (setup/standby/active/sleep)
57               * open remote pipe pending
58               * close remote pipe pending
59               * Current pipe available bitmap
60               * Current pipe closed bitmap
61               * Current connection interval, slave latency and link supervision timeout
62               * Current State of the the GATT client (Service Discovery)
63               * Status of the bond (R) Peer address
64               */
65              static struct aci_state_t aci_state;
66
67              /* Temporary buffers for sending ACI commands */
68              static hal_aci_evt_t  aci_data;
69              //static hal_aci_data_t aci_cmd;
70
```

```
71  /* Timing change state variable */
72  static bool timing_change_done = false;
73
74  /* Used to test the UART TX characteristic notification */
75  static uart_over_ble_t uart_over_ble;
76  static uint8_t         dummychar = 0;
77
78  /* Initialize the radio_ack. This is the ack received for every transmitted packet.
    */
79  //static bool radio_ack_pending = false;
80
81  /* Aquesta funció es cridarà cada cop que rebi un byte. */
82  void (*callback_rcvd)(byte) = NULL;
83  void (*callback_end)(void) = NULL;
84
85
86  /* Define how assert should function in the BLE library */
87  void __ble_assert(const char *file, uint16_t line)
88  {
89    Serial.print("ERROR ");
90    Serial.print(file);
91    Serial.print(": ");
92    Serial.print(line);
93    Serial.print("\n");
94    while(1);
95  }
96
97  /*
98   * Description:
99   *
100  * In this template we are using the BTLE as a UART and can send and receive
    packets.
101  *
102  * The maximum size of a packet is 20 bytes.
103  * When a command it received a response(s) are transmitted back.
104  * Since the response is done using a Notification the peer must have opened
    it(subscribed to it) before any packet is transmitted.
105  * The pipe for the UART_TX becomes available once the peer opens it.
106  * See section 20.4.1 -> Opening a Transmit pipe
107  * In the master control panel, clicking Enable Services will open all the pipes
    on the nRF8001.
108  *
109  * The ACI Evt Data Credit provides the radio level ack of a transmitted packet.
110  */
```

```
111
112             /* Inicialitza les communicacions per bluetooth/usart */
113             void comm_init(void (*on_rcvd)(byte), void (*on_end_rcvd)(void)) {
114               callback_rcvd = on_rcvd;
115               callback_end = on_end_rcvd;
116
117               Serial.begin(57600);
118
119               /**
120               Point ACI data structures to the the setup data that the nRFgo studio
                generated for the nRF8001
121               */
122               if (NULL != services_pipe_type_mapping)
123               {
124                 aci_state.aci_setup_info.services_pipe_type_mapping =
                &services_pipe_type_mapping[0];
125               }
126               else
127               {
128                 aci_state.aci_setup_info.services_pipe_type_mapping = NULL;
129               }
130               aci_state.aci_setup_info.number_of_pipes    = NUMBER_OF_PIPES;
131               aci_state.aci_setup_info.setup_msgs         = setup_msgs;
132               aci_state.aci_setup_info.num_setup_msgs     = NB_SETUP_MESSAGES;
133
134               /*
135               Tell the ACI library, the MCU to nRF8001 pin connections.
136               The Active pin is optional and can be marked UNUSED
137               */
138               aci_state.aci_pins.board_name = BOARD_DEFAULT; //See board.h for details
                REDBEARLAB_SHIELD_V1_1 or BOARD_DEFAULT
139               aci_state.aci_pins.reqn_pin   = 9; //SS for Nordic board, 9 for
                REDBEARLAB_SHIELD_V1_1
140               aci_state.aci_pins.rdyn_pin   = 8; //3 for Nordic board, 8 for
                REDBEARLAB_SHIELD_V1_1
141               aci_state.aci_pins.mosi_pin   = MOSI;
142               aci_state.aci_pins.miso_pin   = MISO;
143               aci_state.aci_pins.sck_pin    = SCK;
144
145               aci_state.aci_pins.spi_clock_divider      = SPI_CLOCK_DIV8;//SPI_CLOCK_DIV8
                = 2MHz SPI speed
146                                                         //SPI_CLOCK_DIV16 = 1MHz SPI
                speed
147
```

```
148   aci_state.aci_pins.reset_pin            = 4; //4 for Nordic board, UNUSED for
      REDBEARLAB_SHIELD_V1_1
149   aci_state.aci_pins.active_pin           = UNUSED;
150   aci_state.aci_pins.optional_chip_sel_pin  = UNUSED;
151
152   aci_state.aci_pins.interface_is_interrupt = false; //Interrupts still not
      available in Chipkit
153   aci_state.aci_pins.interrupt_number       = 1;
154
155   //We reset the nRF8001 here by toggling the RESET line connected to the
      nRF8001
156   //If the RESET line is not available we call the ACI Radio Reset to soft reset
      the nRF8001
157   //then we initialize the data structures required to setup the nRF8001
158   //The second parameter is for turning debug printing on for the ACI
      Commands and Events so they be printed on the Serial
159   lib_aci_init(&aci_state, false);
160  }
161
162  void uart_over_ble_init(void)
163  {
164    uart_over_ble.uart_rts_local = true;
165  }
166
167  bool uart_tx(uint8_t *buffer, uint8_t buffer_len)
168  {
169   bool status = false;
170
171   if (lib_aci_is_pipe_available(&aci_state, PIPE_UART_OVER_BTLE_UART_TX_TX)
      &&
172       (aci_state.data_credit_available >= 1))
173   {
174     status = lib_aci_send_data(PIPE_UART_OVER_BTLE_UART_TX_TX, buffer,
      buffer_len);
175     if (status)
176     {
177       aci_state.data_credit_available--;
178     }
179   }
180   return status;
181  }
182
183  bool uart_process_control_point_rx(uint8_t *byte, uint8_t length)
184  {
```

```
185             bool status = false;
186             aci_ll_conn_params_t *conn_params;
187
188             if (lib_aci_is_pipe_available(&aci_state,
                PIPE_UART_OVER_BTLE_UART_CONTROL_POINT_TX) )
189             {
190               #ifdef TFGCOMM_DEBUG
191               Serial.println(*byte, HEX);
192               #endif
193               switch(*byte)
194               {
195                 /*
196                 Queues a ACI Disconnect to the nRF8001 when this packet is received.
197                 May cause some of the UART packets being sent to be dropped
198                 */
199                 case UART_OVER_BLE_DISCONNECT:
200                   /*
201                   Parameters:
202                   None
203                   */
204                   lib_aci_disconnect(&aci_state, ACI_REASON_TERMINATE);
205                   status = true;
206                   break;
207
208                 /*
209                 Queues an ACI Change Timing to the nRF8001
210                 */
211                 case UART_OVER_BLE_LINK_TIMING_REQ:
212                   /*
213                   Parameters:
214                   Connection interval min: 2 bytes
215                   Connection interval max: 2 bytes
216                   Slave latency:          2 bytes
217                   Timeout:                2 bytes
218                   Same format as Peripheral Preferred Connection Parameters (See nRFgo
                  studio -> nRF8001 Configuration -> GAP Settings
219                   Refer to the ACI Change Timing Request in the nRF8001 Product
                  Specifications
220                   */
221                   conn_params = (aci_ll_conn_params_t *)(byte+1);
222                   lib_aci_change_timing( conn_params->min_conn_interval,
223                                          conn_params->max_conn_interval,
224                                          conn_params->slave_latency,
225                                          conn_params->timeout_mult);
```

```
226                     status = true;
227                     break;
228
229                   /*
230                   Clears the RTS of the UART over BLE
231                   */
232                   case UART_OVER_BLE_TRANSMIT_STOP:
233                     /*
234                     Parameters:
235                     None
236                     */
237                     uart_over_ble.uart_rts_local = false;
238                     status = true;
239                     break;
240
241                   /*
242                   Set the RTS of the UART over BLE
243                   */
244                   case UART_OVER_BLE_TRANSMIT_OK:
245                     /*
246                     Parameters:
247                     None
248                     */
249                     uart_over_ble.uart_rts_local = true;
250                     status = true;
251                     break;
252                 }
253               }
254               return status;
255             }
256
257             void aci_loop()
258             {
259               static bool setup_required = false;
260
261               // We enter the if statement only when there is a ACI event available to be
                processed
262               if (lib_aci_event_get(&aci_state, &aci_data))
263               {
264                 aci_evt_t * aci_evt;
265                 aci_evt = &aci_data.evt;
266
267                 switch(aci_evt->evt_opcode)
268                 {
```

```
269                 /**
270                 As soon as you reset the nRF8001 you will get an ACI Device Started Event
271                 */
272                 case ACI_EVT_DEVICE_STARTED:
273                 {
274                   aci_state.data_credit_total = aci_evt->params.device_started.credit_available;
275                 switch(aci_evt->params.device_started.device_mode)
276                 {
277                   case ACI_DEVICE_SETUP:
278                     /**
279                     When the device is in the setup mode
280                     */
281                     #ifdef TFGCOMM_DEBUG
282                     Serial.println(F("Evt Device Started: Setup"));
283                     #endif
284                     setup_required = true;
285                     break;
286
287                   case ACI_DEVICE_STANDBY:
288                     #ifdef TFGCOMM_DEBUG
289                     Serial.println(F("Evt Device Started: Standby"));
290                     #endif
291                     //Looking for an iPhone by sending radio advertisements
292                     //When an iPhone connects to us we will get an ACI_EVT_CONNECTED event from the nRF8001
293                     if (aci_evt->params.device_started.hw_error)
294                     {
295                       delay(20); //Magic number used to make sure the HW error event is handled correctly.
296                     }
297                     else
298                     {
299                     lib_aci_connect(180/* in seconds */, 0x0050 /* advertising interval 50ms*/);
300                     #ifdef TFGCOMM_DEBUG
301                     Serial.println(F("Advertising started"));
302                     #endif
303                     }
304                     break;
305                 }
306                 }
307                   break; //ACI Device Started Event
308
```

```
309             case ACI_EVT_CMD_RSP:
310               //If an ACI command response event comes with an error -> stop
311               if (ACI_STATUS_SUCCESS != aci_evt->params.cmd_rsp.cmd_status)
312               {
313                 //ACI ReadDynamicData and ACI WriteDynamicData will have status
                codes of
314                 //TRANSACTION_CONTINUE and TRANSACTION_COMPLETE
315                 //all other ACI commands will have status code of
                ACI_STATUS_SCUCCESS for a successful command
316                 #ifdef TFGCOMM_DEBUG
317                 Serial.print(F("ACI Command "));
318                 Serial.println(aci_evt->params.cmd_rsp.cmd_opcode, HEX);
319                 Serial.print(F("Evt Cmd respone: Status "));
320                 Serial.println(aci_evt->params.cmd_rsp.cmd_status, HEX);
321                 #endif
322               }
323               if (ACI_CMD_GET_DEVICE_VERSION == aci_evt-
                >params.cmd_rsp.cmd_opcode)
324               {
325                 //Store the version and configuration information of the nRF8001 in the
                Hardware Revision String Characteristic
326                 lib_aci_set_local_data(&aci_state,
                PIPE_DEVICE_INFORMATION_HARDWARE_REVISION_STRING_SET,
327                   (uint8_t *)&(aci_evt->params.cmd_rsp.params.get_device_version),
                sizeof(aci_evt_cmd_rsp_params_get_device_version_t));
328               }
329               break;
330
331             case ACI_EVT_CONNECTED:
332               #ifdef TFGCOMM_DEBUG
333               Serial.println(F("Evt Connected"));
334               #endif
335               uart_over_ble_init();
336               timing_change_done          = false;
337               aci_state.data_credit_available = aci_state.data_credit_total;
338
339               /*
340              Get the device version of the nRF8001 and store it in the Hardware
                Revision String
341              */
342               lib_aci_device_version();
343               break;
344
345             case ACI_EVT_PIPE_STATUS:
```

```
346                 #ifdef TFGCOMM_DEBUG
347                 Serial.println(F("Evt Pipe Status"));
348                 #endif
349                 if (lib_aci_is_pipe_available(&aci_state,
            PIPE_UART_OVER_BTLE_UART_TX_TX) && (false == timing_change_done))
350                 {
351                   lib_aci_change_timing_GAP_PPCP(); // change the timing on the link as
            specified in the nRFgo studio -> nRF8001 conf. -> GAP.
352                                                     // Used to increase or decrease bandwidth
353                   timing_change_done = true;
354                 }
355                 break;
356
357               case ACI_EVT_TIMING:
358                 #ifdef TFGCOMM_DEBUG
359                 Serial.println(F("Evt link connection interval changed"));
360                 #endif
361                 lib_aci_set_local_data(&aci_state,
362                                        PIPE_UART_OVER_BTLE_UART_LINK_TIMING_CURRENT_SET,
363                                        (uint8_t *)&(aci_evt->params.timing.conn_rf_interval),
            /* Byte aligned */
364
            PIPE_UART_OVER_BTLE_UART_LINK_TIMING_CURRENT_SET_MAX_SIZE);
365                 break;
366
367               case ACI_EVT_DISCONNECTED:
368                 #ifdef TFGCOMM_DEBUG
369                 Serial.println(F("Evt Disconnected/Advertising timed out"));
370                 #endif
371                 lib_aci_connect(180/* in seconds */, 0x0100 /* advertising interval
            100ms*/);
372                 #ifdef TFGCOMM_DEBUG
373                 Serial.println(F("Advertising started"));
374                 #endif
375                 break;
376
377               case ACI_EVT_DATA_RECEIVED:
378                 #ifdef TFGCOMM_DEBUG
379                 Serial.print(F("Pipe Number: "));
380                 Serial.println(aci_evt->params.data_received.rx_data.pipe_number,
            DEC);
381                 #endif
382                 if (PIPE_UART_OVER_BTLE_UART_RX_RX == aci_evt-
            >params.data_received.rx_data.pipe_number)
```

```
383                 {
384                   #ifdef TFGCOMM_DEBUG
385                   Serial.print(F(" Data(Hex) : "));
386                   #endif
387                   for(int i=0; i<aci_evt->len - 2; i++)
388                   {
389                     #ifdef TFGCOMM_DEBUG
390                     Serial.print((char)aci_evt-
                  >params.data_received.rx_data.aci_data[i]);
391                     #endif
392
393                     if (callback_rcvd != NULL) {
394                       //Callback a cridar en rebre un caracter
395                       callback_rcvd((byte)aci_evt-
                  >params.data_received.rx_data.aci_data[i]);
396                     }
397
398                     #ifdef TFGCOMM_DEBUG
399                     Serial.print(F(" "));
400                     #endif
401                   }
402
403                   if (callback_end != NULL) {
404                     callback_end();
405                   }
406
407                   #ifdef TFGCOMM_DEBUG
408                   Serial.println(F(""));
409                   #endif
410                   if (lib_aci_is_pipe_available(&aci_state,
                  PIPE_UART_OVER_BTLE_UART_TX_TX))
411                   {
412                     /*Do this to test the loopback otherwise comment it out
413                     */
414                     /*
415                     if (!uart_tx(&uart_buffer[0], aci_evt->len - 2))
416                     {
417                       Serial.println(F("UART loopback failed"));
418                     }
419                     else
420                     {
421                       Serial.println(F("UART loopback OK"));
422                     }
423                     */
```

```
424                 }
425               }
426               if (PIPE_UART_OVER_BTLE_UART_CONTROL_POINT_RX == aci_evt->params.data_received.rx_data.pipe_number)
427               {
428                 uart_process_control_point_rx(&aci_evt->params.data_received.rx_data.aci_data[0], aci_evt->len - 2); //Subtract for Opcode and Pipe number
429               }
430               break;
431
432             case ACI_EVT_DATA_CREDIT:
433               aci_state.data_credit_available = aci_state.data_credit_available + aci_evt->params.data_credit.credit;
434               break;
435
436             case ACI_EVT_PIPE_ERROR:
437               //See the appendix in the nRF8001 Product Specication for details on the error codes
438               #ifdef TFGCOMM_DEBUG
439               Serial.print(F("ACI Evt Pipe Error: Pipe #:"));
440               Serial.print(aci_evt->params.pipe_error.pipe_number, DEC);
441               Serial.print(F("  Pipe Error Code: 0x"));
442               Serial.println(aci_evt->params.pipe_error.error_code, HEX);
443               #endif
444
445               //Increment the credit available as the data packet was not sent.
446               //The pipe error also represents the Attribute protocol Error Response sent from the peer and that should not be counted
447               //for the credit.
448               if (ACI_STATUS_ERROR_PEER_ATT_ERROR != aci_evt->params.pipe_error.error_code)
449               {
450                 aci_state.data_credit_available++;
451               }
452               break;
453
454             case ACI_EVT_HW_ERROR:
455               #ifdef TFGCOMM_DEBUG
456               Serial.print(F("HW error: "));
457               Serial.println(aci_evt->params.hw_error.line_num, DEC);
458
459               for(uint8_t counter = 0; counter <= (aci_evt->len - 3); counter++)
460               {
```

```
461                 Serial.write(aci_evt->params.hw_error.file_name[counter]);
462               }
463               Serial.println();
464               #endif
465               lib_aci_connect(180/* in seconds */, 0x0050 /* advertising interval
          50ms*/);
466               #ifdef TFGCOMM_DEBUG
467               Serial.println(F("Advertising started"));
468               #endif
469               break;
470
471             }
472           }
473           else
474           {
475             //Serial.println(F("No ACI Events available"));
476             // No event in the ACI Event queue and if there is no event in the ACI
          command queue the arduino can go to sleep
477             // Arduino can go to sleep now
478             // Wakeup from sleep from the RDYN line
479           }
480
481           /* setup_required is set to true when the device starts up and enters setup
          mode.
482            * It indicates that do_aci_setup() should be called. The flag should be cleared
          if
483            * do_aci_setup() returns ACI_STATUS_TRANSACTION_COMPLETE.
484            */
485           if(setup_required)
486           {
487             if (SETUP_SUCCESS == do_aci_setup(&aci_state))
488             {
489               setup_required = false;
490             }
491           }
492         }
493
494         bool stringComplete = false;  // whether the string is complete
495         uint8_t stringIndex = 0;      //Initialize the index to store incoming chars
496
497         void serialEvent();
498         void comm_loop(void)
499         {
500           //Process any ACI commands or events
```

```
501                 aci_loop();
502                 //For ChipKit you have to call the function that reads from Serial
503                 if (Serial.available())
504                 {
505                   serialEvent();
506                 }
507               }
508
509               /*
510                COMMENT ONLY FOR ARDUINO
511                SerialEvent occurs whenever a new data comes in the
512                hardware serial RX.  This routine is run between each
513                time loop() runs, so using delay inside loop can delay
514                response.  Multiple bytes of data may be available.
515                Serial Event is NOT compatible with Leonardo, Micro, Esplora
516                */
517               void serialEvent() {
518
519                 while(Serial.available() > 0){
520                   // get the new byte:
521                   dummychar = (byte)Serial.read();
522                   if (dummychar == '\n') {
523                     if (callback_end != NULL) {
524                       callback_end();
525                     }
526                   } else {
527                     if (callback_rcvd != NULL) {
528                       callback_rcvd(dummychar); //La resta ho fa aquesta funciÃ³.
529                     }
530                   }
531                 }
532               }
```

6.6. tfg_comm.h

Header del mòdul comm.

```
1               #ifndef TFG_COMMS_H
2               #define TFG_COMMS_H
3
4               #include "Arduino.h"
5
6               /*
```

```
7                * Inicialitza uart/bluetooth. Quan es rebi un byte per bluetooth es
8                * passarà a on_rcvd, si no és NULL. Quan s'hagi rebut un missatge es
9                * cridarà on_end_rcvd si no és NULL.
10               */
11               void comm_init(void (*on_rcvd)(byte), void (*on_end_rcvd)(void));
12
13               /*
14                * S'ha de cridar cada poc temps per tal de que executi les
15                * funcions corresponents.
16                */
17               void comm_loop(void);
18
19               #endif
```

Nota: A més d'aquests fitxers, el mòdul comm utilitza els fitxers donats en l'exemple radino_nRF8001_IO_HF_USB_Test donat per la llibreria del Radino.

6.7. prova_comm.ino

Exemple/test del mòdul comm.

```
1                /*
2                 * Versio antiga del IDE: si una llibreria utilitza d'altres, s'han
3                 * d'incloure totes les llibreries en l'sketch principal.
4                 */
5                #include <SPI.h>
6                #include <lib_aci.h>
7                /**************************/
8
9                #include <tfg_comm.h>
10
11               void on_byte_blt(byte c) {
12                 Serial.print((char)c);
13               }
14
15               void print_linebreak(void) {
16                 Serial.println();
17               }
18
19               void setup() {
20                 Serial.begin(57600);
21                 delay(5000);
22                 comm_init(on_byte_blt, print_linebreak);
```

```
23          }
24
25          void loop() {
26            comm_loop();
27            delay(10);
28          }
29
```

6.8. Modificacions a hal_aci_tl.cpp

Nota: Només s'inclouen les seccions del codi modificades, junt amb els numeros de linia corresponents. Aquest fitxer està inclòs en la llibreria donada per Radino.

```
22          #define ADXL362_WANTS_MSBFIRST

41          #if !defined(ADXL362_WANTS_MSBFIRST) && defined (__AVR__)
42            //For Arduino add nothing
43          #elif defined(__PIC32MX__) || defined(ADXL362_WANTS_MSBFIRST)
44          //For ChipKit as the transmission has to be reversed, the next definitions have to
            be added
45            //L'accelerometre vol que el SPI estigui configurat amb MSBFIRST
46            #define REVERSE_BITS(byte) (((reverse_lookup[(byte & 0x0F)]) << 4)
            + reverse_lookup[((byte & 0xF0) >> 4)])
47            static const uint8_t reverse_lookup[] = { 0, 8,  4, 12, 2, 10, 6, 14,1, 9, 5,
            13,3, 11, 7, 15 };
48          #endif

354           #if !defined(ADXL362_WANTS_MSBFIRST) && defined (__AVR__)
355             //For Arduino use the LSB first
356             SPI.setBitOrder(LSBFIRST);
357           #elif defined(__PIC32MX__) || defined(ADXL362_WANTS_MSBFIRST)
358             //For ChipKit use MSBFIRST and REVERSE the bits on the SPI as LSBFIRST
            is not supported
359             //Per el ADXL362, s'ha de configurar l'SPI amb MSBFIRST o sigui que fem
            servir la mateixa estrategia
360             SPI.setBitOrder(MSBFIRST);
361           #endif

428         #if !defined(ADXL362_WANTS_MSBFIRST) && defined (__AVR__)
429           //For Arduino the transmission does not have to be reversed
430           return SPI.transfer(aci_byte);
431         #elif defined(__PIC32MX__) || defined(ADXL362_WANTS_MSBFIRST)
432           //For ChipKit the transmission has to be reversed
433           //Per l'ADXL362 també
```

```
434             uint8_t tmp_bits;
435             tmp_bits = SPI.transfer(REVERSE_BITS(aci_byte));
436             return REVERSE_BITS(tmp_bits);
437           #endif
```

6.9. tfg_leds.cpp

Codi font del mòdul leds.

```
1           #include "Arduino.h"
2           #include "tfg_leds.h"
3
4           byte imatge[N_COLUMNES] = {
5            0b1111111,
6            0b0001000,
7            0b0001000,
8            0b0001000,
9            0b1111111,
10           0b0000000,
11
12           0b1111111,
13           0b1001001,
14           0b1001001,
15           0b1001001,
16           0b1000001,
17           0b0000000,
18
19           0b1111111,
20           0b1000000,
21           0b1000000,
22           0b1000000,
23           0b1000000,
24           0b0000000,
25
26           0b1111111,
27           0b1000000,
28           0b1000000,
29           0b1000000,
30           0b1000000,
31           0b0000000,
32
33           0b0111110,
34           0b1000001,
```

```
35                0b1000001,
36                0b1000001,
37                0b0111110,
38                0b0000000
39               };
40
41               /* Encen/Apaga el led N (s=HIGH/LOW) */
42               static inline void set_led(byte n, byte s) {
43                byte m;
44
45                switch(n) {
46                case 1: m = A0; break;
47                case 2: m = A1; break;
48                case 3: m = A2; break;
49                case 4: m = A3; break;
50                case 5: m = 10; break;
51                case 6: m = 11; break;
52                case 7: m = 12; break;
53                }
54
55                digitalWrite(m,s);
56               }
57
58               void leds_init(void) {
59                //pinMode(6, OUTPUT);  //radino pin 4 = Arduino D6
60                pinMode(10, OUTPUT); //radino pin 5 = Arduino IO10
61                pinMode(11, OUTPUT); //radino pin 6 = Arduino IO11
62                //pinMode(13, OUTPUT); //radino pin 7 = LED = IO13
63                //pinMode(5, OUTPUT);  //radino pin 8 = Arduino D5
64                //pinMode(A5, OUTPUT); //radino pin 9 = Arduino A5
65                //pinMode(0, OUTPUT);  //radino pin 12= Arduino D0/RXD
66                //pinMode(1, OUTPUT);  //radino pin 13= Arduino D1/TXD
67                pinMode(A1, OUTPUT); //radino pin 19= Arduino A1
68                pinMode(A2, OUTPUT); //radino pin 20= Arduino A2
69                pinMode(A0, OUTPUT); //radino pin 21= Arduino A0
70                pinMode(A3, OUTPUT); //radino pin 22= Arduino A3
71                //pinMode(2, OUTPUT);  //radino pin 23= Arduino D2/SDA
72                //pinMode(3, OUTPUT);  //radino pin 24= Arduino D3/SCL
73                pinMode(12, OUTPUT); //radino pin 25= Arduino IO12
74
75                for (byte i=1; i<=7; i++) {
76                  set_led(i,LOW);
77                }
78               }
```

```
79
80              /* Encen els leds per mostrar la columna donada */
81              static inline void encen_leds(byte columna) {
82               for (uint8_t i=0; i<N_LEDS; i++) {
83                if (columna & (1<<i)) {
84                 set_led(i+1, HIGH);
85                } else {
86                 set_led(i+1, LOW);
87                }
88               }
89              }
90
91              void leds_load_col(byte columna, byte n) {
92               imatge[n] = columna;
93              }
94
95              void leds_show_col(byte n) {
96               encen_leds(imatge[n]);
97              }
98
99              void leds_off(void) {
100              encen_leds(0);
101             }
102
```

6.10. tfg_leds.h

Header del mòdul leds.

```
1               #ifndef TFG_LEDS_H
2               #define TFG_LEDS_H
3
4               #include "Arduino.h"
5
6               #define N_LEDS 7
7               #define N_COLUMNES 36
8
9               /* Posa els pins corresponents a cada led com a sortides, i els apaga. */
10              void leds_init(void);
11
12              /* Canvia la columna N de la imatge guardada en memoria. */
13              void leds_load_col(byte columna, byte n);
14
```

```
15              /* Mostra la columna N de la imatge (encen leds) */
16              void leds_show_col(byte n);
17
18              /* Apaga els leds */
19              void leds_off(void);
20
21              #endif
```

6.11. prova_leds.ino

Exemple/test del mòdul leds.

```
1               #include <tfg_leds.h>
2
3               void setup() {
4                 leds_load_col(0b1000001, 0);
5                 leds_load_col(0b1100011, 1);
6                 leds_load_col(0b1110111, 2);
7                 leds_load_col(0b1111111, 3);
8                 leds_load_col(0b1111111, 4);
9                 leds_load_col(0b1110111, 5);
10                leds_load_col(0b1100011, 6);
11                leds_load_col(0b1000001, 7);
12                leds_load_col(0b0000000, 8);
13                leds_load_col(0b0000001, 9);
14                leds_load_col(0b0000011, 10);
15                leds_load_col(0b0000111, 11);
16                leds_load_col(0b0001111, 12);
17                leds_load_col(0b0011111, 13);
18                leds_load_col(0b0111111, 14);
19                leds_load_col(0b1111111, 15);
20                leds_load_col(0b1111110, 16);
21                leds_load_col(0b1111100, 17);
22                leds_load_col(0b1111000, 18);
23                leds_load_col(0b1110000, 19);
24                leds_load_col(0b1100000, 20);
25                leds_load_col(0b1000000, 21);
26                leds_load_col(0b0000000, 22);
27                leds_load_col(0b1000001, 23);
28                leds_load_col(0b0100010, 24);
29                leds_load_col(0b0010100, 25);
30                leds_load_col(0b0001000, 26);
31                leds_load_col(0b0010100, 27);
```

```
32                leds_load_col(0b0100010, 28);
33                leds_load_col(0b1000001, 29);
34
35                leds_init();
36              }
37
38              void loop() {
39                for (byte i=0; i<30; i++) {
40                  leds_show_col(i);
41                  delay(100);
42                }
43                leds_off();
44                delay(100);
45              }
46
```

6.12. tfg_timer.cpp

Codi font del mòdul timer.

```
1               #include "Arduino.h"
2               #include <stdint.h>
3               #include <stdlib.h>
4               #include "tfg_timer.h"
5
6               void(*callbacks[TIMER_NENTRIES])(void);
7               unsigned long ms_left_cb[TIMER_NENTRIES];
8
9               void timer_init(void) {
10                for (uint8_t i=0; i<TIMER_NENTRIES; i++) {
11                  callbacks[i] = NULL;
12                }
13              }
14
15              void timer_after(void(*cb)(void), unsigned long ms) {
16                for (uint8_t i=0; i<TIMER_NENTRIES; i++) {
17                  if (callbacks[i] == NULL) {
18                    callbacks[i] = cb;
19                    ms_left_cb[i] = ms;
20                    return;
21                  }
22                }
23              }
```

```
24
25              void timer_loop(void) {
26                static unsigned long last_millis = 0;
27
28                unsigned long current_millis = millis();
29                unsigned long diff_millis = current_millis - last_millis;
30
31                if (diff_millis == 0) return;
32                last_millis = current_millis;
33
34                for (uint8_t i=0; i<TIMER_NENTRIES; i++) {
35                  if (callbacks[i] != NULL) {
36                    if (diff_millis >= ms_left_cb[i]) {
37                      (*callbacks[i])();
38                      callbacks[i] = NULL;
39                    } else {
40                      ms_left_cb[i] -= diff_millis;
41                    }
42                  }
43                }
44              }
```

6.13. tfg_timer.h

Header del mòdul timer.

```
1               #ifndef TFG_TIMER_H
2               #define TFG_TIMER_H
3
4               #define TIMER_NENTRIES 30
5
6               /* Inicialitza variables del modul */
7               void timer_init(void);
8
9               /* Crida la funcio donada al cap del temps donat. */
10              void timer_after(void(*cb)(void), unsigned long ms);
11
12              /*
13               * No hi ha acces al timer de HW, utilitza polling i la funcio millis().
14               * S'ha de cridar aquesta funcio en el loop()
15               */
16              void timer_loop(void);
17
```

```
18              #endif
```

6.14. prova_timer.ino

Exemple/test del mòdul timer.

```
1               #include <tfg_timer.h>
2
3               void segon(void) {
4                 static byte n=0;
5
6                 Serial.print('.');
7
8                 if (++n >= 60) {
9                   Serial.print('\n');
10                  n = 0;
11                }
12
13                timer_after(segon, 1000);
14              }
15
16              void setup() {
17                timer_init();
18                delay(5000);
19                Serial.println("Dibuixa un . per segon,\n"
20                               "       un \\n per minut,\n");
21                timer_after(segon, 1000);
22              }
23
24              void loop() {
25                timer_loop();
26              }
```

6.15. tfg.ino

Programa principal, utilitzant els mòduls anteriors.

```
1               /*
2                * Versio antiga del IDE: si una llibreria utilitza d'altres, s'han
3                * d'incloure totes les llibreries en l'sketch principal.
4                */
5               #include <ADXL362.h>
```

```
6   #include <SPI.h>
7   #include <lib_aci.h>
8   /*********************/
9
10  #include <tfg_accel.h>
11  #include <tfg_leds.h>
12  #include <tfg_timer.h>
13  #include <tfg_comm.h>
14
15  #include "character_defs.h"
16
17  #define COLUMN_PERIOD 3
18  #define SM_PERIOD 5
19  #define COMM_PERIOD 10
20
21  #define NO_SERIAL_OUTPUT
22
23  byte load_column_index = 0;
24  static void on_byte_received(byte rcvd) {
25    byte c[5];
26
27    if (load_column_index >= N_COLUMNES) return;
28    /* Format: 7x5 nomes majuscules i numeros (i alguns simbols) */
29    switch(rcvd) {
30      TFG_CHARACTER_SWITCH(c);
31    }
32
33    leds_load_col(c[0],load_column_index++);
34    leds_load_col(c[1],load_column_index++);
35    leds_load_col(c[2],load_column_index++);
36    leds_load_col(c[3],load_column_index++);
37    leds_load_col(c[4],load_column_index++);
38    leds_load_col(0b0000000,load_column_index++);
39  }
40
41  void on_msg_end(void) {
42    while (load_column_index < N_COLUMNES) {
43      leds_load_col(0b0000000,load_column_index++);
44    }
45    load_column_index = 0;
46  }
47
48  /*
49   * Les dues funcions mostren la seguent columna cada cop que es criden.
```

```
50                * Una en un ordre i l'altra en l'altre.
51                */
52               void mostra_columna(void) {
53                 static uint8_t i = 0;
54
55                 leds_show_col(i++);
56
57                 if (i >= N_COLUMNES) {
58                   i = 0;
59                   timer_after(leds_off, COLUMN_PERIOD);
60                 } else {
61                   timer_after(mostra_columna, COLUMN_PERIOD);
62                 }
63               }
64               void mostra_columna_backwards(void) {
65                 static uint8_t i = N_COLUMNES;
66
67                 leds_show_col(--i);
68
69                 if (i <= 0) {
70                   i = N_COLUMNES;
71                   timer_after(leds_off, COLUMN_PERIOD);
72                 } else {
73                   timer_after(mostra_columna_backwards, COLUMN_PERIOD);
74                 }
75               }
76
77               void mostra_imatge(bool backwards) {
78                 if (backwards) {
79                   mostra_columna_backwards();
80                 } else {
81                   mostra_columna();
82                 }
83               }
84
85               void sm_loop(void) {
86                 /* Estat de la maquina d'estats */
87                 static enum {
88                   INCREMENTANT,
89                   MAXIM_TROBAT_POTSER,
90                   DECREMENTANT,
91                   MINIM_TROBAT_POTSER
92                 } estat = INCREMENTANT;
93
```

```
94                  /* Valor de l'accelerÃ²metre anterior */
95                  static int16_t val_anterior;
96
97                  /* Per tal d'inicialitzar val_anterior */
98                  static bool once=true;
99
100                 /* Pren la segÃ¼ent mostra de l'accelerÃ²metre */
101                 int16_t val = accel_next();
102
103                 #ifndef NO_SERIAL_OUTPUT
104                 Serial.print(millis());
105                 Serial.print(' ');
106                 Serial.print(val);
107                 Serial.print(' ');
108                 /*
109                  * INCREMENTANT      -> 0b000 (0)
110                  * DECREMENTANT      -> 0b001 (1)
111                  * DETERMINANT MÃ€XIM -> 0b010 (2)
112                  * DETERMINANT MÃ½NIM -> 0b011 (3)
113                  * MÃ€XIM TROBAT      -> 0b100 (4)
114                  * MÃ½NIM TROBAT      -> 0b101 (5)
115                  */
116                 #endif
117
118                 /* Inicialitza val_anterior el primer cop */
119                 if (once) {
120                   once = false;
121                   val_anterior = val;
122                 }
123
124                 /* Maquina d'estats */
125                 switch (estat) {
126                 case INCREMENTANT:
127                   /* PodrÃem estar en un mÃ xim si el valor ha comenÃ§at a
128                      decrementar. */
129                   if (val < val_anterior) {
130                     estat = MAXIM_TROBAT_POTSER;
131                   }
132                   #ifndef NO_SERIAL_OUTPUT
133                    Serial.print('0');
134                   #endif
135                   break;
136
137                 case MAXIM_TROBAT_POTSER:
```

```
138                 /* Estarem en un mÃ xim si encara estem decrementant i el valor es
139                  positiu */
140                 if (val <= val_anterior && val > 500) {
141                  /* Estem en un mÃ xim */
142                  #ifndef NO_SERIAL_OUTPUT
143                   Serial.print('4');
144                  #endif
145                  mostra_imatge(true);
146                  estat = DECREMENTANT;
147                 } else {
148                  /* Encara estem incrementant */
149                  #ifndef NO_SERIAL_OUTPUT
150                   Serial.print('2');
151                  #endif
152                  estat = INCREMENTANT;
153                 }
154                 break;
155
156               case DECREMENTANT:
157                /* PodrÃem estar en un mÃnim si comencem a incrementar. */
158                if (val > val_anterior) {
159                 estat = MINIM_TROBAT_POTSER;
160                }
161                #ifndef NO_SERIAL_OUTPUT
162                 Serial.print('1');
163                #endif
164                break;
165               case MINIM_TROBAT_POTSER:
166
167                if (val >= val_anterior && val < -500) {
168                 /* Estem en un mÃnim */
169                 #ifndef NO_SERIAL_OUTPUT
170                  Serial.print('5');
171                 #endif
172                 mostra_imatge(false);
173                 estat = INCREMENTANT;
174                } else {
175                 /* Encara estem decrementant */
176                 #ifndef NO_SERIAL_OUTPUT
177                  Serial.print('3');
178                 #endif
179                 estat = DECREMENTANT;
180                }
181                break;
```

```
182
183             default:
184               #ifndef NO_SERIAL_OUTPUT
185                Serial.print('!');
186               #endif
187               estat = INCREMENTANT;
188             }
189
190             #ifndef NO_SERIAL_OUTPUT
191              Serial.print('\n');
192             #endif
193             val_anterior = val;
194
195             timer_after(sm_loop, SM_PERIOD);
196           }
197
198           void comm_loop_timer(void) {
199             comm_loop();
200             timer_after(comm_loop_timer, COMM_PERIOD);
201           }
202
203           void setup() {
204             Serial.begin(57600);
205             //delay(5000);
206             leds_init();
207             comm_init(on_byte_received, on_msg_end);
208             accel_init();
209             timer_init();
210
211             timer_after(comm_loop_timer, COMM_PERIOD);
212             timer_after(sm_loop, 10000);
213           }
214
215           void loop() {
216             timer_loop();
217           }
```

6.16. character_defs.h

Definicions de cadascun dels caràcters utilitzats. S'inclou com un fitxer separat per tal que el programa principal no sigui innecessàriament llarg, i per tal de separar-les de la resta del codi.

```
#ifndef TFG_CHARACTER_DEFS

#define TFG_CHAR_TO_CASE(c,var,n0,n1,n2,n3,n4)  \
 case c:                          \
 var[0] = 0b##n0;                 \
 var[1] = 0b##n1;                 \
 var[2] = 0b##n2;                 \
 var[3] = 0b##n3;                 \
 var[4] = 0b##n4;                 \
 break;

#define TFG_CHARACTER_SWITCH(var)          \
 TFG_CHAR_TO_CASE('0',var,             \
          0111110,             \
          1010001,             \
          1001001,             \
          1000101,             \
          0111110)             \
                                \
 TFG_CHAR_TO_CASE('1',var,             \
          0000000,             \
          1000010,             \
          1111111,             \
          1000000,             \
          0000000)             \
                                \
 TFG_CHAR_TO_CASE('2',var,             \
          1100010,             \
          1010001,             \
          1001001,             \
          1001001,             \
          1000110)             \
                                \
 TFG_CHAR_TO_CASE('3',var,             \
          0100001,             \
          1000001,             \
          1001001,             \
          1001101,             \
          0110011)             \
                                \
 TFG_CHAR_TO_CASE('4',var,             \
          0011000,             \
          0010100,             \
          0010010,             \
```

```
45                      1111111,        \
46                      0010000)        \
47                                      \
48              TFG_CHAR_TO_CASE('5',var,       \
49                      0100111,        \
50                      1000101,        \
51                      1000101,        \
52                      1000101,        \
53                      0111001)        \
54                                      \
55              TFG_CHAR_TO_CASE('6',var,       \
56                      0111100,        \
57                      1001010,        \
58                      1001001,        \
59                      1001001,        \
60                      0110001)        \
61                                      \
62              TFG_CHAR_TO_CASE('7',var,       \
63                      0000001,        \
64                      1110001,        \
65                      0001001,        \
66                      0000101,        \
67                      0000011)        \
68                                      \
69              TFG_CHAR_TO_CASE('8',var,       \
70                      0110110,        \
71                      1001001,        \
72                      1001001,        \
73                      1001001,        \
74                      0110110)        \
75                                      \
76              TFG_CHAR_TO_CASE('9',var,       \
77                      1000110,        \
78                      1001001,        \
79                      1001001,        \
80                      0101001,        \
81                      0011110)        \
82                                      \
83              case 'A':               \
84              TFG_CHAR_TO_CASE('a',var,       \
85                      1111100,        \
86                      0010010,        \
87                      0010001,        \
88                      0010010,        \
```

```
89                      1111100)            \
90                                          \
91              case 'B':                   \
92              TFG_CHAR_TO_CASE('b',var,           \
93                      1111111,            \
94                      1001001,            \
95                      1001001,            \
96                      1001001,            \
97                      0110110)            \
98                                          \
99              case 'C':                   \
100             TFG_CHAR_TO_CASE('c',var,           \
101                     0111110,            \
102                     1000001,            \
103                     1000001,            \
104                     1000001,            \
105                      0100010)           \
106                                         \
107             case 'D':                   \
108             TFG_CHAR_TO_CASE('d',var,           \
109                     1111111,            \
110                     1000001,            \
111                     1000001,            \
112                     1000001,            \
113                      0111110)           \
114                                         \
115             case 'E':                   \
116             TFG_CHAR_TO_CASE('e',var,           \
117                     1111111,            \
118                     1001001,            \
119                     1001001,            \
120                     1001001,            \
121                     1000001)            \
122                                         \
123             case 'F':                   \
124             TFG_CHAR_TO_CASE('f',var,           \
125                     1111111,            \
126                     0001001,            \
127                     0001001,            \
128                     0001001,            \
129                      0000001)           \
130                                         \
131             case 'G':                   \
132             TFG_CHAR_TO_CASE('g',var,           \
```

```
133                     0111110,            \
134                     1000001,            \
135                     1000001,            \
136                     1010001,            \
137                      1110001)           \
138                                         \
139             case 'H':                   \
140             TFG_CHAR_TO_CASE('h',var,           \
141                     1111111,            \
142                     0001000,            \
143                     0001000,            \
144                     0001000,            \
145                     1111111)            \
146                                         \
147             case 'I':                   \
148             TFG_CHAR_TO_CASE('i',var,           \
149                     0000000,            \
150                     1000001,            \
151                     1111111,            \
152                     1000001,            \
153                      0000000)           \
154                                         \
155             case 'J':                   \
156             TFG_CHAR_TO_CASE('j',var,           \
157                     0100000,            \
158                     1000000,            \
159                     1000000,            \
160                     1000000,            \
161                      0111111)           \
162                                         \
163             case 'K':                   \
164             TFG_CHAR_TO_CASE('k',var,           \
165                     1111111,            \
166                     0001000,            \
167                     0010100,            \
168                     0100010,            \
169                     1000001)            \
170                                         \
171             case 'L':                   \
172             TFG_CHAR_TO_CASE('l',var,           \
173                     1111111,            \
174                     1000000,            \
175                     1000000,            \
176                     1000000,            \
```

```
177                     1000000)          \
178                                       \
179             case 'M':                 \
180             TFG_CHAR_TO_CASE('m',var,          \
181                     1111111,          \
182                     0000010,          \
183                     0001100,          \
184                     0000010,          \
185                      1111111)         \
186                                       \
187             case 'N':                 \
188             TFG_CHAR_TO_CASE('n',var,          \
189                     1111111,          \
190                     0000100,          \
191                     0001000,          \
192                     0010000,          \
193                     1111111)          \
194                                       \
195             case 'O':                 \
196             TFG_CHAR_TO_CASE('o',var,          \
197                     0111110,          \
198                     1000001,          \
199                     1000001,          \
200                     1000001,          \
201                      0111110)         \
202                                       \
203             case 'P':                 \
204             TFG_CHAR_TO_CASE('p',var,          \
205                     1111111,          \
206                     0001001,          \
207                     0001001,          \
208                     0001001,          \
209                      0000110)         \
210                                       \
211             case 'Q':                 \
212             TFG_CHAR_TO_CASE('q',var,          \
213                     0111110,          \
214                     1000001,          \
215                     1000001,          \
216                     0100001,          \
217                     1011110)          \
218                                       \
219             case 'R':                 \
220             TFG_CHAR_TO_CASE('r',var,          \
```

```
                    1111111,            \
                    0001001,            \
                    0011001,            \
                    0101001,            \
                     1000110)           \
                                        \
        case 'S':                       \
        TFG_CHAR_TO_CASE('s',var,           \
                    0100110,            \
                    1001001,            \
                    1001001,            \
                    1001001,            \
                     0110010)           \
                                        \
        case 'T':                       \
        TFG_CHAR_TO_CASE('t',var,           \
                    0000001,            \
                    0000001,            \
                    1111111,            \
                    0000001,            \
                    0000001)            \
                                        \
        case 'U':                       \
        TFG_CHAR_TO_CASE('u',var,           \
                    0111111,            \
                    1000000,            \
                    1000000,            \
                    1000000,            \
                     0111111)           \
                                        \
        case 'V':                       \
        TFG_CHAR_TO_CASE('v',var,           \
                    0011111,            \
                    0100000,            \
                    1000000,            \
                    0100000,            \
                     0011111)           \
                                        \
        case 'W':                       \
        TFG_CHAR_TO_CASE('w',var,           \
                    1111111,            \
                    0100000,            \
                    0011000,            \
                    0100000,            \
```

```
265                     1111111)            \
266                                         \
267             case 'X':                   \
268             TFG_CHAR_TO_CASE('x',var,            \
269                     1100011,            \
270                     0010100,            \
271                     0001000,            \
272                     0010100,            \
273                      1100011)           \
274                                         \
275             case 'Y':                   \
276             TFG_CHAR_TO_CASE('y',var,            \
277                     0000011,            \
278                     0000100,            \
279                     1111000,            \
280                     0000100,            \
281                     0000011)            \
282                                         \
283             case 'Z':                   \
284             TFG_CHAR_TO_CASE('z',var,            \
285                     1100001,            \
286                     1010001,            \
287                     1001001,            \
288                     1000101,            \
289                     1000011)            \
290                                         \
291             TFG_CHAR_TO_CASE('!',var,            \
292                     0000000,            \
293                     0000000,            \
294                     1011111,            \
295                     0000000,            \
296                     0000000)            \
297                                         \
298             TFG_CHAR_TO_CASE('"',var,            \
299                     0000000,            \
300                     0000111,            \
301                     0000000,            \
302                     0000111,            \
303                      0000000)           \
304                                         \
305             TFG_CHAR_TO_CASE('#',var,             \
306                     0010100,            \
307                     1111111,            \
308                     0010100,            \
```

```
309                     1111111,            \
310                     0010100)            \
311                                         \
312             TFG_CHAR_TO_CASE('$',var,           \
313                     0100100,            \
314                     0101010,            \
315                     1111111,            \
316                     0101010,            \
317                      0010010)           \
318                                         \
319             TFG_CHAR_TO_CASE('%',var,           \
320                     0100011,            \
321                     0010011,            \
322                     0001000,            \
323                     1100100,            \
324                      1100010)           \
325                                         \
326             TFG_CHAR_TO_CASE('&',var,           \
327                     0110110,            \
328                     1001001,            \
329                     1010110,            \
330                     0100000,            \
331                     1010000)            \
332                                         \
333             TFG_CHAR_TO_CASE('\'',var,          \
334                     0000000,            \
335                     0000000,            \
336                     0000111,            \
337                     0000000,            \
338                      0000000)           \
339                                         \
340             TFG_CHAR_TO_CASE('(',var,           \
341                     0011100,            \
342                     0100010,            \
343                     1000001,            \
344                     0000000,            \
345                     0000000)            \
346                                         \
347             TFG_CHAR_TO_CASE(')',var,           \
348                     0000000,            \
349                     0000000,            \
350                     1000001,            \
351                     0100010,            \
352                     0011100)            \
```

```
353                                             \
354             TFG_CHAR_TO_CASE('*',var,               \
355                     0100010,            \
356                     0010100,            \
357                     1111111,            \
358                     0010100,            \
359                      0100010)           \
360                                             \
361             TFG_CHAR_TO_CASE('+',var,               \
362                     0001000,            \
363                     0001000,            \
364                     0111110,            \
365                     0001000,            \
366                     0001000)            \
367                                             \
368             TFG_CHAR_TO_CASE(',',var,               \
369                     0000000,            \
370                     1000000,            \
371                     0110000,            \
372                     0000000,            \
373                     0000000)            \
374                                             \
375             TFG_CHAR_TO_CASE('-',var,               \
376                     0001000,            \
377                     0001000,            \
378                     0001000,            \
379                     0001000,            \
380                      0001000)           \
381                                             \
382             TFG_CHAR_TO_CASE('.',var,               \
383                     0000000,            \
384                     0000000,            \
385                     1000000,            \
386                     0000000,            \
387                     0000000)            \
388                                             \
389             TFG_CHAR_TO_CASE('/',var,               \
390                     0100000,            \
391                     0010000,            \
392                     0001000,            \
393                     0000100,            \
394                     0000010)            \
395                                             \
396             TFG_CHAR_TO_CASE(':',var,               \
```

```
397                     0000000,            \
398                     0000000,            \
399                     0010100,            \
400                     0000000,            \
401                      0000000)           \
402                                         \
403             TFG_CHAR_TO_CASE(';',var,           \
404                     0000000,            \
405                     1000000,            \
406                     0110100,            \
407                     0000000,            \
408                     0000000)            \
409                                         \
410             TFG_CHAR_TO_CASE('<',var,           \
411                     0001000,            \
412                     0010100,            \
413                     0100010,            \
414                     1000001,            \
415                     0000000)            \
416                                         \
417             TFG_CHAR_TO_CASE('=',var,           \
418                     0010100,            \
419                     0010100,            \
420                     0010100,            \
421                     0010100,            \
422                      0010100)           \
423                                         \
424             TFG_CHAR_TO_CASE('>',var,           \
425                     0000000,            \
426                     1000001,            \
427                     0100010,            \
428                     0010100,            \
429                     0001000)            \
430                                         \
431             TFG_CHAR_TO_CASE('?',var,           \
432                     0000010,            \
433                     0000001,            \
434                     1011001,            \
435                     0000101,            \
436                     0000010)            \
437                                         \
438             TFG_CHAR_TO_CASE('@',var,           \
439                     0111110,            \
440                     1000001,            \
```

```
441                     1011101,            \
442                     1011001,            \
443                      1001110)           \
444                                         \
445             TFG_CHAR_TO_CASE('[',var,           \
446                     1111111,            \
447                     1111111,            \
448                     1000001,            \
449                     1000001,            \
450                     1000001)            \
451                                         \
452             TFG_CHAR_TO_CASE('\\',var,          \
453                     0000010,            \
454                     0000100,            \
455                     0001000,            \
456                     0010000,            \
457                     0100000)            \
458                                         \
459             TFG_CHAR_TO_CASE(']',var,           \
460                     1000001,            \
461                     1000001,            \
462                     1000001,            \
463                     1111111,            \
464                      1111111)           \
465                                         \
466             TFG_CHAR_TO_CASE('^',var,            \
467                     0010000,            \
468                     0001000,            \
469                     0000100,            \
470                     0001000,            \
471                     0010000)            \
472                                         \
473             TFG_CHAR_TO_CASE('_',var,           \
474                     1000000,            \
475                     1000000,            \
476                     1000000,            \
477                     1000000,            \
478                     1000000)            \
479                                         \
480             TFG_CHAR_TO_CASE('`',var,           \
481                     0000000,            \
482                     0000001,            \
483                     0000010,            \
484                     0000100,            \
```

```
485                          0000000)           \
486                                             \
487                 TFG_CHAR_TO_CASE('{',var,           \
488                          0001000,           \
489                          0111110,           \
490                          1110111,           \
491                          1000001,           \
492                          1000001)           \
493                                             \
494                 TFG_CHAR_TO_CASE('|',var,           \
495                          0000000,           \
496                          0000000,           \
497                          1111111,           \
498                          0000000,           \
499                          0000000)           \
500                                             \
501                 TFG_CHAR_TO_CASE('}',var,           \
502                          1000001,           \
503                          1000001,           \
504                          1110111,           \
505                          0111110,           \
506                           0001000)          \
507                                             \
508                 TFG_CHAR_TO_CASE('~',var,           \
509                          0000010,           \
510                          0000001,           \
511                          0000011,           \
512                          0000010,           \
513                          0000001)           \
514                 case ' ':                   \
515                 default:                     \
516                  var[0] = 0b0000000;              \
517                  var[1] = 0b0000000;              \
518                  var[2] = 0b0000000;              \
519                  var[3] = 0b0000000;              \
520                  var[4] = 0b0000000
521
522                 #endif
```

6.17. prova_main.py

Script per llegir l'output del programa principal i mostrar una gràfica.

```
1  import serial
2  import time
3  import matplotlib.pyplot as plt
4  from troba_tty import troba_tty
5
6  arduino = serial.Serial(port=troba_tty(),
7                          baudrate=38400,
8                          bytesize=8,
9                          parity=serial.PARITY_NONE,
10                         stopbits=1,
11                         timeout=1)
12 time.sleep(2)
13
14 mostra_n = []
15 valors = []
16 events = dict(zip(range(6),[[] for _ in range(6)]))
17 while True:
18   try:
19     rcvd = arduino.readline()
20     try:
21       time,val,event = [int(n) for n in rcvd.strip().split()]
22     except:
23       print repr(rcvd)
24       continue
25
26     print time,val,event
27
28     mostra_n.append(time)
29     valors.append(val)
30
31     for key in events:
32       events[key].append(-2000)
33     events[event].pop()
34     events[event].append(4000)
35   except KeyboardInterrupt:
36     break
37
38 arduino.close()
39
40 plt.plot(mostra_n,valors,'-b',
41        mostra_n,events[0b000],'-g', #Green,   Incrementant
42        mostra_n,events[0b001],'-r', #Red,     Decrementant
43        mostra_n,events[0b010],'-c', #Cyan,    Fals Maxim
44        mostra_n,events[0b011],'-m', #Magenta, Fals Minim
```

```
45                  mostra_n,events[0b100],'-y', #Yellow,  Maxim
46                  mostra_n,events[0b101],'-k', #Black,   Minim
47              )
48              plt.show()
```