Pràctiques de Sistemes Digitals i Microprocessadors

Curs 2016-2017

Pràctica 2 – Fase 2

El Walkie-Textie

Alumnes	Login	Nom
	Ls30759	Samuel Tavares da Silva
	Ls3062	Gabriel Cammany Ruiz

Entrega	Placa	Memòria	Nota

Data	21/05/2017

Portada de la memòria

Pràctiques de Sistemes Digitals i Microprocessadors

Curs 2016-2017

Pràctica 2 – Fase 2

El Walkie-Textie

	Login	Nom
Alumnes	Ls30759	Samuel Tavares da Silva
	Ls3062	Gabriel Cammany Ruiz

	Placa	Memòria	Nota
Entrega			

Data	21/05/2017

Portada de l'alumne

Índex

Síntesi de l'enunciat 4
Plantejament 5
UART 5
PWM 5
Radio Freqüència 6
LCD 7
A/D 8
Altaveu 9
Diagrama de mòduls 10
LCD 10
ADC 11
Altaveu 11
UART 11
PIC24 11
Disseny 12
Diagrama de TADs 12
Motors 15
Radio Freqüència 15
Propaganda 16
PWM 17
SIO 19
Àudio 19
LCD 20
Esquemes elèctrics 21
Problemes observats 23
Conclusions 24
Planificació 25
Annex 27

Síntesi de l'enunciat

Aquesta practica es basa en la implementació d'una xarxa de telecomunicació basada en enllaços de radiofreqüència controlats per microcontroladors. En el nostre cas sens proposa l'ús d'un emissor central i receptors a les sucursals, d'aquesta manera podrem enviar missatges sense fils, per tal de no suposar una molèstia a l'hora de testejar i implementar el projecte s'ha decidit enviar el missatges com a text, enlloc de missatges de veu com un sistema de Walkie-Talkie típic.

Per aquesta fase, com que en l'anterior vàrem implementar un emissor RF, realitzarem la part de recepció de les trames.

Tenint en compte aquest objectiu, s'ha dividit la fase en variïs mòduls, els quals es dedicaran a realitzar les diferents parts del projecte.

Per començar, el mes important, la recepció per Radio freqüència. Donat que ho hem separat per mòduls, només s'ha d'encarregar d'anar escoltant i desant el missatge en cas de que vagi dirigit a nosaltres, de manera independent als altres apartats. Aquest haurà de tenir un protocol de comunicació compartit amb l'anterior fase de manera que pugui rebre les trames que transmet.

A banda del RF, un altre part important que haurem de realitzar ha de ser la comunicació del usuari amb el nostre sistema. Ja que el nostre receptor podrà ser consultat sobre les dades que conte en qualsevol moment per l'usuari. Per tant, mitjançant una interfície gràfica, que en el nostre cas es un hyperterminal, mostrarem un menú que deixarà a l'usuari realitzar un conjunt d'accions. Tot això recolzat per el perifèric EUSART.

Juntament amb aquesta interfície gràfica, l'usuari també serà capaç de veure el estat del sistema mitjançant un LCD. Aquest anirà aportant informació al usuari de les dades que hi ha. Es a dir, contindrà el ID del usuari, el número de trames pròpies, trames totals i finalment un missatge dinàmic. Aquest últim haurà d'anar variant depenent del que estigui passant.

Després de les parts de comunicació amb l'usuari, també haurem de tenir tres ports del nostre microcontrolador que es dediquin a generar un pols de modulació d'amplada constant, que anirà variant mentre no l'usuari no introdueixi el ID. Un cop introduït , s'ha de mantenir amb una amplada determinada basant-se amb el dígit corresponent al port PWM. Altrament, hem de tenir connectat un altaveu que utilitzarem per generar una melodia que variarà respecte la longitud del missatge que hem rebut.

Ja per acabar, l'ultima mòdul que tenim es un ADC. Aquest es dedicarà a obtenir el valor analògic d'una entrada amb un potenciòmetre, que definirà el temps de refresc de la pantalla.

Plantejament

Un cop hem analitzat el objectiu que hem d'assolir, hem començat a plantejar de quina manera afrontarem aquesta segona fase.

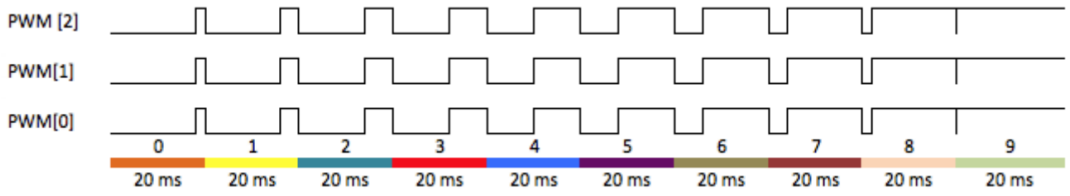
UART *Universal asynchronous receiver/transmitter*

Per dur a terme aquest bloc, utilitzarem el canal sèrie que te la PIC24 per tal de comunicar-nos amb el usuari. Aquesta comunicació es durà a terme mitjançant un menú amb les diferents opcions de selecció, que serà mostrat gràficament per el *Hyperterminal*.

1. Assignar el ID de la nostre placa
    - Per dur a terme aquesta part, simplement esperarem a que l'usuari ens introdueixi el ID i un cop introduït el desarem en una variable.
2. Consultar el ID que hem afegit
    - Retornarem la variable on hem desat el ID
3. Consultar número de trames identificades
    - Donat que durant tot el procés de RF ens anirem fent un recompte de les trames que hem rebut, simplement enviarem el valor d'aquest comptador.
4. Consultar número de trames totals
    - Com en l'opció anterior, retornarem la variable que porta el recompte de trames totals.
5. Consultar l'últim missatge rebut
    - Finalment, com en els anteriors casos, per retornar l'últim missatge rebut, enviarem la variable on ens hem desat el missatge quan ens ha entrat per RF.

PWM *Pulse-width modulation*

Aquest bloc s'encarrega de generar una senyal coneguda com PWM, modulació per amplada de polsos, que no deixa de ser una senyal, que en el nostre cas va a 50Hz, la qual va variant la seva amplada amb el temps.
Per tant, el PWM variarà la seva amplada en referencia al ID que tinguem en cada moment. Hem de generar tres PWM per tal de tenir un per cada dígit, i la seva figura tindrà la següent forma especificada per l'enunciat.



Així doncs, en el moment que estiguem esperant per el ID, l'amplada del pols que enviem per els ports específics, variaran depenent del dígit que estigui mostrant-se, per tant, donat que els dígits aniran rotant, la senyal anirà incrementant la seva amplada fins arribar a ser tot 1. A partir d'allà, com que el ID anirà rotant, tornarem al 0 i començarem de nou amb l'amplada del 0.

Radio Freqüència

La manera que afrontarem aquest bloc, serà la següent.

Donat que rebem el que la primera fase ens envia, necessitarem fer uns petits canvis a la primera fase per tal de que d'alguna manera sapiguem que el que estem rebent, prové de la nostra fase 1.

Per tant, plantejarem el nostre protocol de la manera que tinguem un caràcter especial en la part inicial de la nostra trama que ens defineixi que això es una trama que pot anar dirigida a nosaltres. Tot seguit, afegirem el nostre identificador de grup, ja que es possible que rebem trames que tinguin la mateixa estructura però no estiguin destinades a la nostra placa.

Tot i plantejar el contingut del missatge, la part on es necessitarà un protocol mes precís serà el que anomenem la part de sincronització. Definim la part de sincronització com aquell moment que la nostre placa ha de separar el soroll de una possible trama.

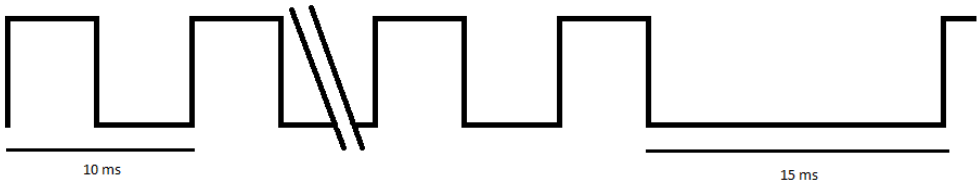
Aquesta, ha de tenir una estructura simple però que puguem confiar al màxim, ja que d'ella dependrà si ens perdem una trama dirigida a nosaltres o no. Es per aquest motiu que l'hem dissenyat d'una manera que ens aporta un equilibri entre simplicitat i a la vegada fiabilitat.

Per això hem decidit fer el protocol que es veu en la figura. Enviarem 10 polsos de 10 mseg, amb un duty cycle del 50% i tot seguit una espera de 10 mseg. Donat que no sabem quant el nostre receptor començarà a llegir aquesta trama, no ens podem basar en que haurà de rebre 10, sinó que serà en el moment que tinguem la espera de 15 mseg despres d'un pols la que ens indicarà que això es una trama amb la nostra estructura i per tant l'hem d'escoltar.

Un cop que han passat els 10 mseg, rebrem el nostre start byte, que consistirà del caràcter '$', tot seguir del nostre ID de placa, que hem adjuntat amb el missatge que s'envia de la primera fase.

Per tant, quan una trama compleixi aquesta estructura de sincronització i tingui el mateix ID, la considerem com a nostre, en cas contrari la declararem una trama no identificada.

Hem de tenir en compte, que com s'ha plantejat l'estructura del programa de manera cooperativa, es a dir, on no bloqueja'm el microcontrolador fent una cosa única sinó separem la carrega de treball en petites execucions, el plantejament del bloc RF, s'ha fet de tal manera que estarem escoltant en tot moment qualsevol trama que anem rebent. Així, no farà falta cap instrucció per part del usuari que comencem a escoltar.



LCD *Liquid Crystal Display*

Aquest bloc te diferents funcions assignades, podríem dir que es tracta d'una versió física del mateix menú que es transmet al PC. A diferencia d'aquest, l'usuari no pot decidir que mostrar, es mostrarà una cosa o altre depenent del moment.

Així doncs, per pantalla s'ha de mostrar la següent informació:

1. ID del usuari

Aquest apartat, tal com es comenta en l'enunciat, fins que l'usuari no ha introduït un ID, ha d'anar rotant cíclicament, des de 000 fins a 999.

2. Nombre de trames identificades

Tal com ho realitzem en l'apartat del menú, cada cop que fem un *refresh* del display, consultarem la variable de trames identificades que conte el recompte actual de trames que hem rebut cap a la nostre placa.

3. Nombre de trames totals

Com en el cas anterior, anirem constantment consultant la variable i en cas de que es modifiqui en el pròxim *refresh* de la pantalla s'actualitzarà automàticament.

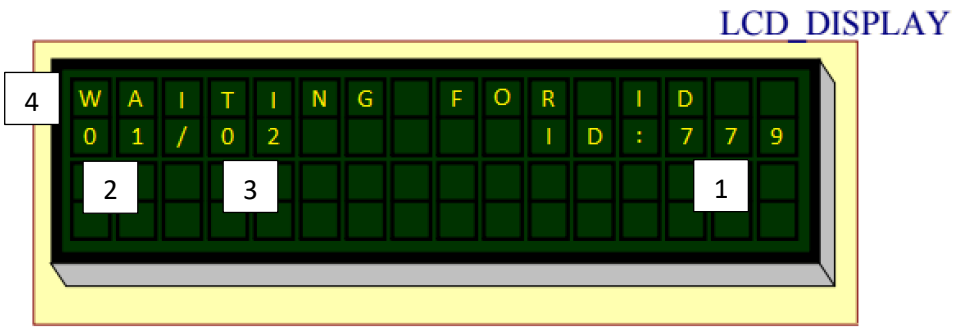
4. Missatge de status

Aquest apartat, es el mes complet, ja que ha de mostrar diferents missatges depenent del moment que estiguem, es a dir, en cas de rebre un missatge haurà d'anar mostrant el missatge rebut cíclicament, en els altres casos ha de mostrar tres missatges

- Waiting for ID
- ID set
- New Message

Així doncs, en el inici del nostre sistema, es mostrarà de manera estàtica el primer. Tot seguit, un cop introduït el ID, canviarà el missatge a "ID set".

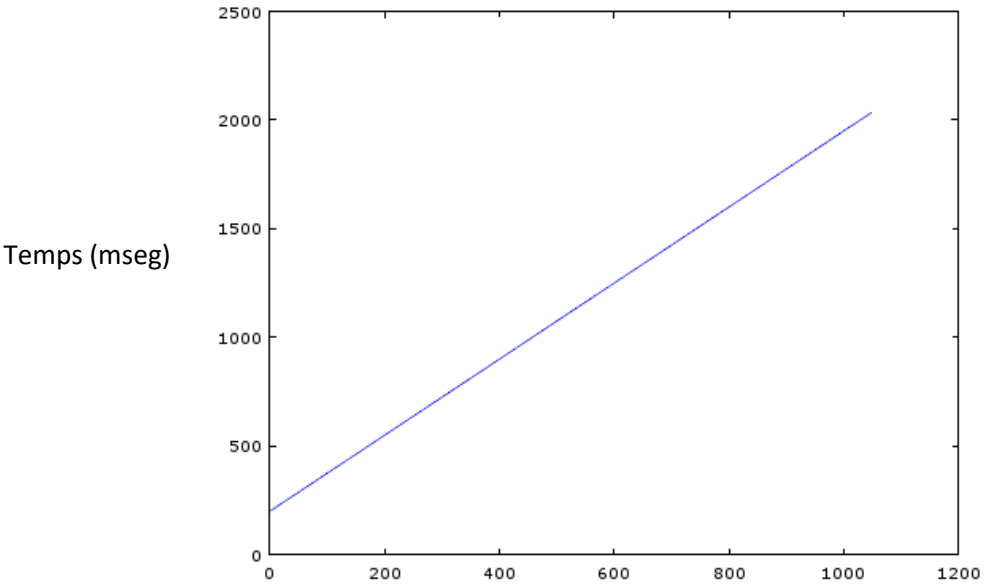
Finalment, quan rebem un missatge mostrarem l'últim com a avis de que s'ha rebut un nou missatge i es mostrarà a continuació.



A/D *Analog-to-Digital*

El funcionament d'aquest apartat es força senzill. A partir de la mostra que ens introdueixen per el conversor analògic digital, hem de establir el temps de refresh del LCD.

Per tant, dit d'un altre manera, no deixarem actualitzar les dades del LCD fins que no passi el temps extret de la conversió de l'entrada analògica a digital. La conversió del valor que ens introdueixen a temps, es pot veure en la següent figura:



Com es pot veure, es una simple recta. Per tant utilitzant matemàtiques bàsiques, es pot treure un valor que ens introdueixen per l'entrada analògica a temps d'actualització. L'equació es la següent:

$$y = 1.75 * x + 200$$

Altaveu

Finalment, per realitzar l'apartat del altaveu, realitzarem quelcom similar a l'anterior apartat.

Per establir la melodia a sonar per aquest, hem d'establir un període. El període es important, ja que depenent d'aquest l'altaveu sonarà duna manera o altre. Perquè si recordem electrònica bàsica, el període es igual a 1/f. Modificant el període variarem la freqüència i per tant el altaveu anirà modificant el so que genera.

A partir d'aquesta base i tenint en compte que hem de variar la melodia depenent de la longitud del missatge, realitzarem una simple regla de tres, com en l'anterior cas, però sense tenir un offset de base, per tant partirà de 0, sent 0 bytes 0 de període.

Tot i així, com que el període del altaveu mínim ha de ser de 1 com també el numero de bytes enviats, no fa falta que especifiquem un offset, ja que quan arribem a utilitzar aquesta equació segur que tenim mínim un byte de missatge.

$$y = \frac{numBytes}{30}$$

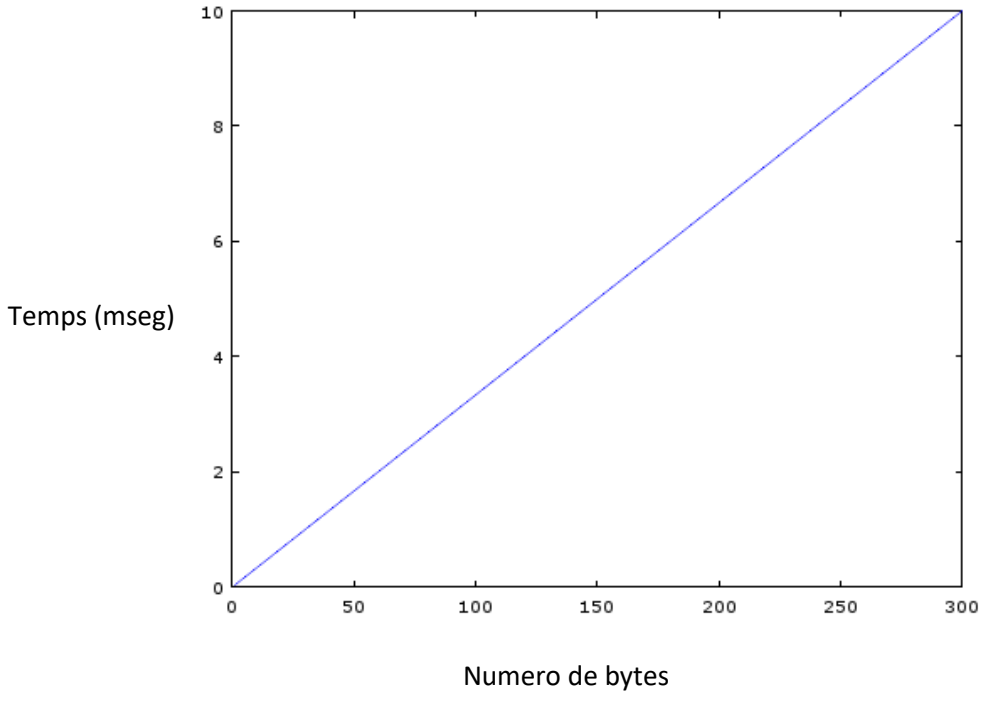
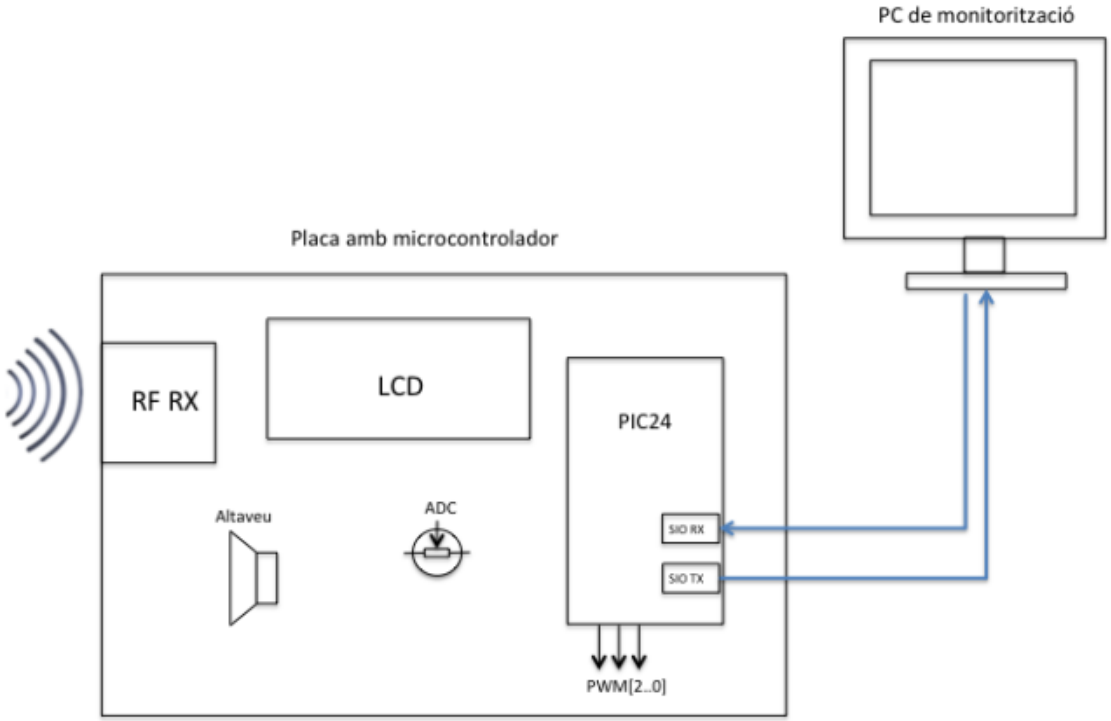


Diagrama de mòduls



Partint del diagrama de mòduls que se'ns ha plantejat a l'enunciat de la practica mostrarem el nostre plantejament respecte cada mòdul.

LCD

Aquest mòdul perquè funcioni correctament, haurà de rebre les següents variables

Entrades

- Missatge
- ID
- Trames Totals
- Trames identificades

Sortides

- Pantalla

ADC

Aquest es el bloc encarregat de gestionar un potenciòmetre, que utilitzem com entrada analògica al nostra sistema, permetent-nos regular la velocitat de desplaçament del missatge del display. Per requeriment de l´enunciat establim que el valor mínim del potenciòmetre serà de 200 mil·lisegons per lletra, mentre que el valor màxim serà de 2 segons per lletra.

Podríem dir per tant, que aquest bloc està integrat en el mateix bloc LCD.

Altaveu

Sent el bloc encarregat de gestionar l´altaveu ens permetrà sentir una petita melodia de freqüència proporcional a la llargada del missatge, sempre i quan es rebi una trama amb el mateix ID associat al nostre sistema. Per tant, per generar la melodia, es necessitarà com a entrada el període.

UART

Bloc encarregat de gestionar el port sèrie de la nostra pic amb el PC de monitorització, mitjançant l'ús del protocol RS-232. Cal destacar que el PC ens permetrà configurar l'identificador de la placa o bé consultar diferents estadístiques relacionades amb el funcionament de la placa.

PIC24

Com ja sabem, aquest mòdul es l'encarregat de fer tota la lògica de la practica. Per tant les entrades que tindrà aquest mòdul seran les següents:

- SioRX
- RFRx
- Potenciometre

Per la banda de les senyals de sortida:

- PWM[0..2]
- SioRx
- AudioOut
- D[7..4]
- Rs
- E
- RW

Disseny

Diagrama de TADs

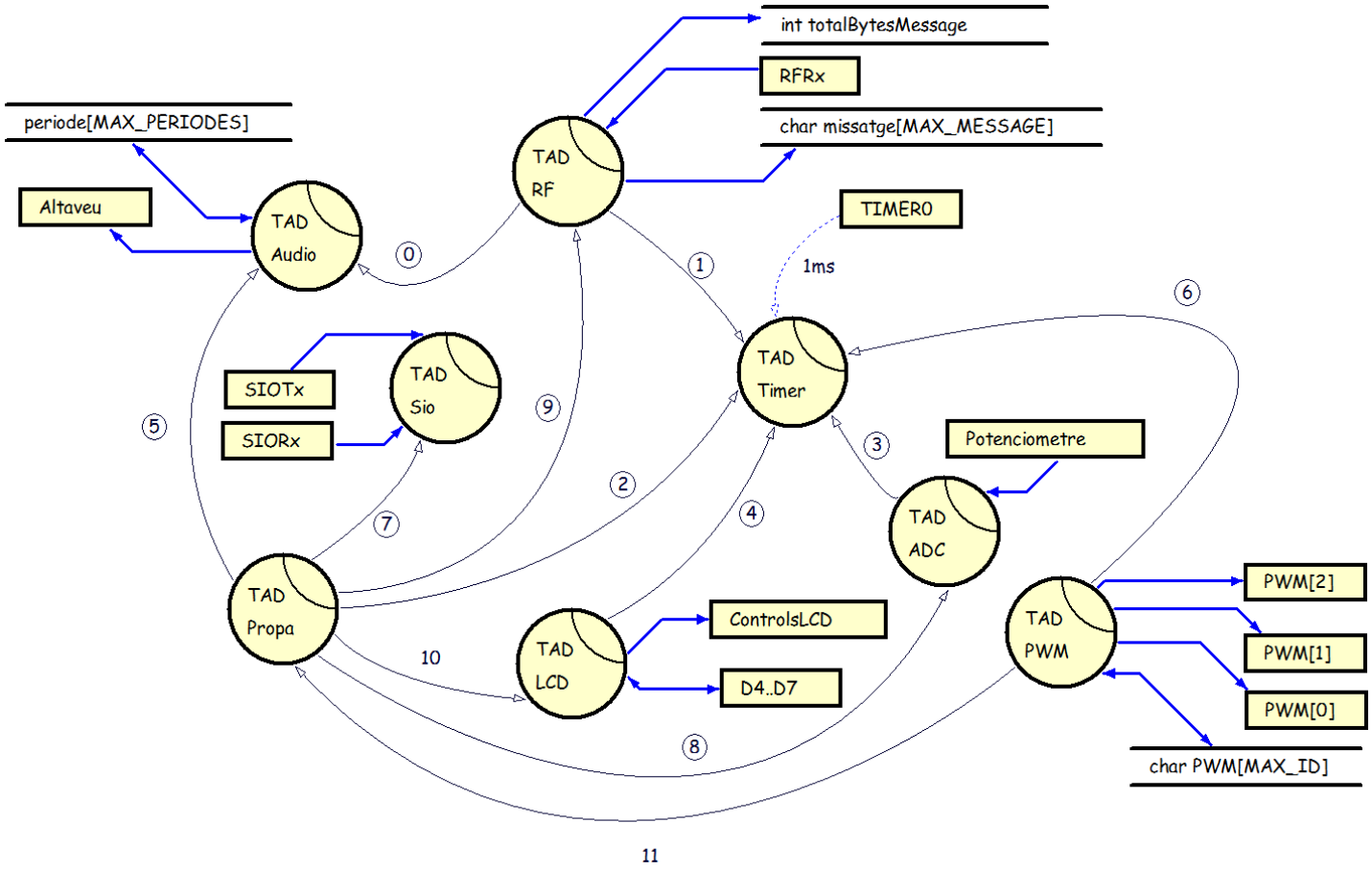
El diagrama de TADs d'aquesta fase es el següent.

Tots els perifèrics estan definits amb noms genèrics per no especificar massa en els ports.

El diagrama consta de 10 interfícies que comuniquen els diferents TADs.

Com podem veure, tots els TADs excepte el TSIO utilitzen el timer y podríem dir que el TPropaganda es qui gestiona la majoria de successos.

D'aquests 8 TADs, només hi han 6 motors. Els únics que no en tenen son el LCD, timer i el ADC. El propaganda però, conté el motor LCD.



Diccionari

Interfície 0

```
//Interficie 0

char changeAudioStatus();
    //Post Canvia l'estat d'audio

void setAudioPeriode(char nouPeriode);
// Pre: nouPeriode >= 1
```

Interfícies 1,2,3,4,6

```
int TiGetTimer (void);
    /*************************************************************************
    //Precondicions: Hi ha algun timer lliure. Maxim TI_NUMTIMERS
    //Postcondicions: Retorna un handle per usar les funcions TiGetTics i
    //TiResetTics. Retorna -1 si no hi ha cap timer disponible.
    \\************************************************************************/

void TiResetTics (unsigned char Handle);
    /*************************************************************************
    //Precondicions: Handle ha estat retornat per Ti_OpenTimer.
    //Postcondicions: Engega la temporitzaci$ associada a 'Handle'.
    //              i agafa la referencia temporal del sistema
    \\************************************************************************/

unsigned int  TiGetTics (unsigned char Handle);
    /*************************************************************************
    //Precondicions: Handle ha estat retornat per TiGetTimer.
    // La distància temmporal entre GetTics i ResetTics ha de ser menor
    // de TI_MAXTICS ms (actualment, 30 segons)
    //Postcondicions: Retorna els milisegons transcorreguts des de la crida
    //                a l'StartTimer del 'Handle'.
    \\************************************************************************/
```

Interfície 5

```
void turnOffAudio();
//Post: Apaga el Altaveu

void seguentFrequencia();
//Post: Canvia el periode del altaveu amb una desviació maxima del
//original de MAX_PERIODES
```

Interfície 8

```
int AdGetMostra(void);
//Post: Retorna la mostra convertida en 10 bits
```

Interfície 7

```
int SiCharAvail(void);
// Pre: retorna el nombre de car?cters rebuts que no s'han recollit
// amb la funciÛ GetChar encara

char SiGetChar(void);
// Pre: SiCharAvail() Ès major que zero
// Post: Treu i retorna el primer car?cter de la cua de recepciÛ.

void SiSendChar(char c);
// Post: espera que el car?cter anterior s'hagi enviat i envia aquest

void SiPutsCooperatiu(char *s);
    //Pre: La referència de char *s és o bé un const char o bé puc garantir que
    //     no es sobreescriurà fins que no l'hagi enviat...
    //Post: Encua *s a la cua de cadenes per enviar...
```

Interfície 9

```
int getLength();
//Pre: -
//Post: Retorna la mida del missatge tenim desat

char* getTramesTotals();
//Pre: -
//Post: Retorna el numero de trames totals

char* getTramesPropies();
//Pre: -
//Post: Retorna el numero de trames propies

unsigned char* getMessage(unsigned char offset);
//Pre: 0<= offset <= MAX_MESSAGE
//Post: Retorna el missatge que hi ha actualment amb el offset especificat
```

Interfície 11

```
char getIDPos(unsigned char pos);
//Pre: 0 <= pos < MAX_ID_STRING
//Post: Retorna el digit del ID que es demani
```

Interfície 10

```
void LcClear(void);
// Post: Esborra el display i posa el cursor a la posiciÛ zero en
// l'estat en el que estava.
// Post: La propera ordre pot trigar fins a 1.6ms

void LcGotoXY(char Columna, char Fila);
// Pre : Columna entre 0 i 39, Fila entre 0 i 3
// Post: Posiciona el cursor en aquestes coordenades
// Post: La propera ordre pot trigar fins a 40us

void LcPutChar(char c);
// Post: Pinta C en l'actual posciciÛ del cursor i incrementa
// la seva posiciÛ. Si la columna arriba a 39, salta a 0 tot
// incrementant la fila si el LCD Ès de dues files.
// Si es de 4 files, incrementa de fila en arribar a la columna 20
// Així mateix, la fila 4 passa a la zero.
// En els LCDs d'una fila, quan la columna arriba a 39, torna
// a zero. No s'incrementa mai la fila
```

Motors

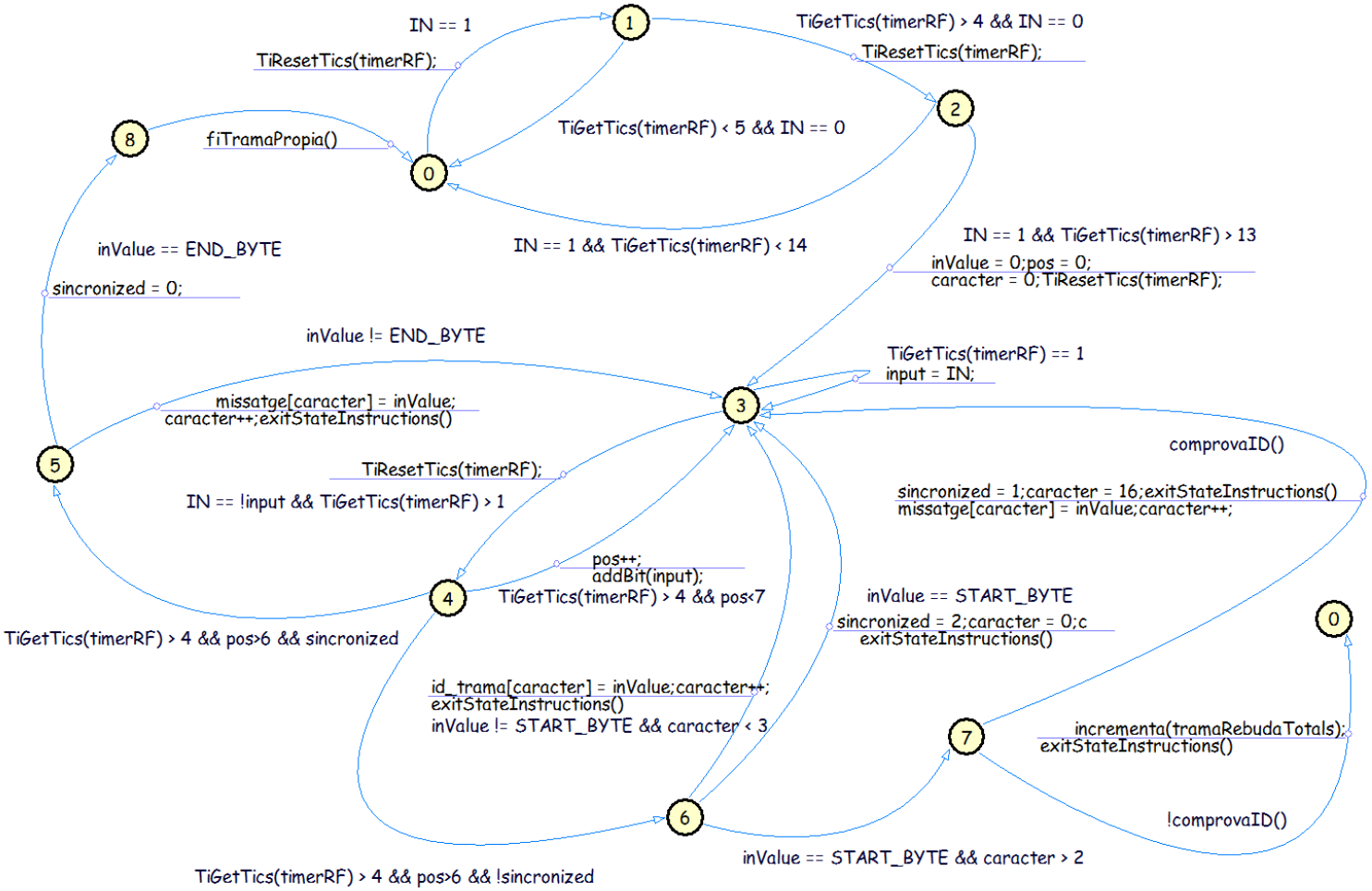
Radio Freqüència

Aquest motor esta pensat amb la màxima cooperativitat possible però a la vegada realitzar-ho en un nombre de estats reduït.

Podríem dir que esta dividit en dues parts principals.

Sincronització

La part de sincronització de la trama seran els estats 0,1 i 2. Per tant seran els estats on mes estona estigui el motor, donat que no sempre estarem reben missatges, aquests estats es dedicaran a "filtrar" el soroll del espai de transmissió per radio freqüència.



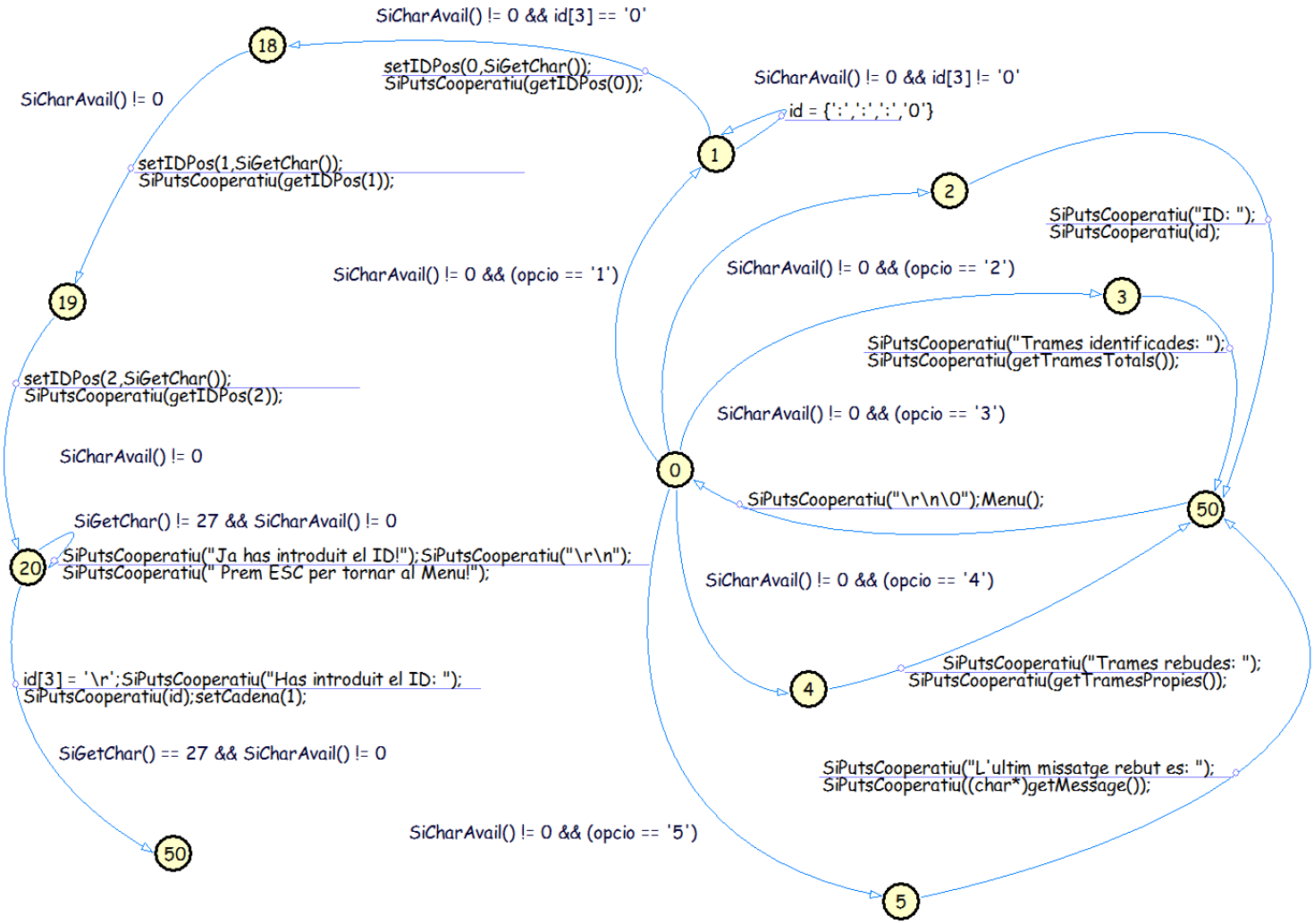
Anàlisi i processament del missatge

Un cop hem detectat una tram que conte la mateixa estructura que la nostre, son els estats a partir del 3 que realitzen les comprovacions pertinents a si la trama conte el mateix identificador que la nostra placa (estat 7) o si es el inici/final de la trama (estats 5 i 6). A part d'aquestes comprovacions, es l'estat 3 i 4, principalment, que es dediquen a extreure els bits de cada byte de al trama.

Propaganda

Aquest motor, te el objectiu de comunicar-se amb al usuari. Mitjançant les interfícies que te amb els diferents TADs, va executant les ordes que l'usuari introdueix per el terminal del ordinador.

La particularitat d'aquest motor es que la majoria de les seves operacions son realitzades en altres TADs i aquest nomes es dedica a cridar-les depenent del moment en que es trobi.



PWM

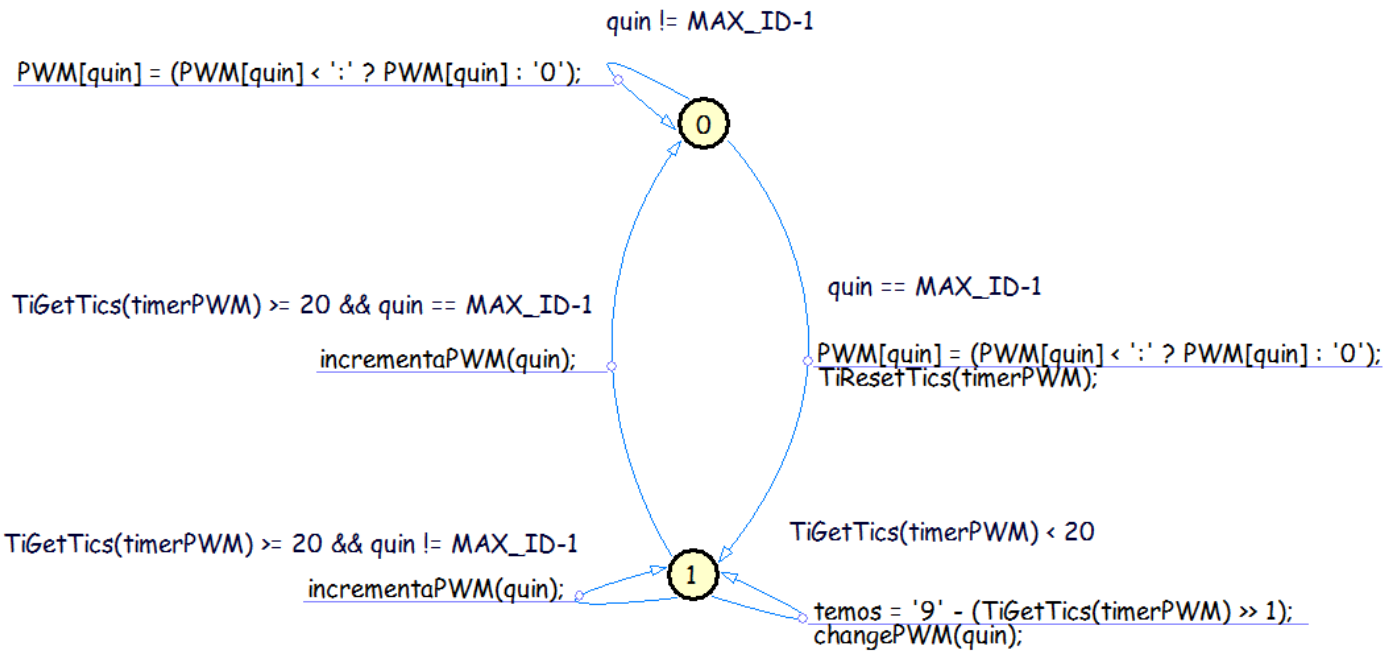
Com ja hem explicat amb anterioritat, el PWM es bàsicament una modulació per amplada de polsos, es a dir, una senyal que varia la seva amplada (temps a 1 normalment).

Donat que l'objectiu en si es bastant simple, el motor ha acabat sent bastant simple també.

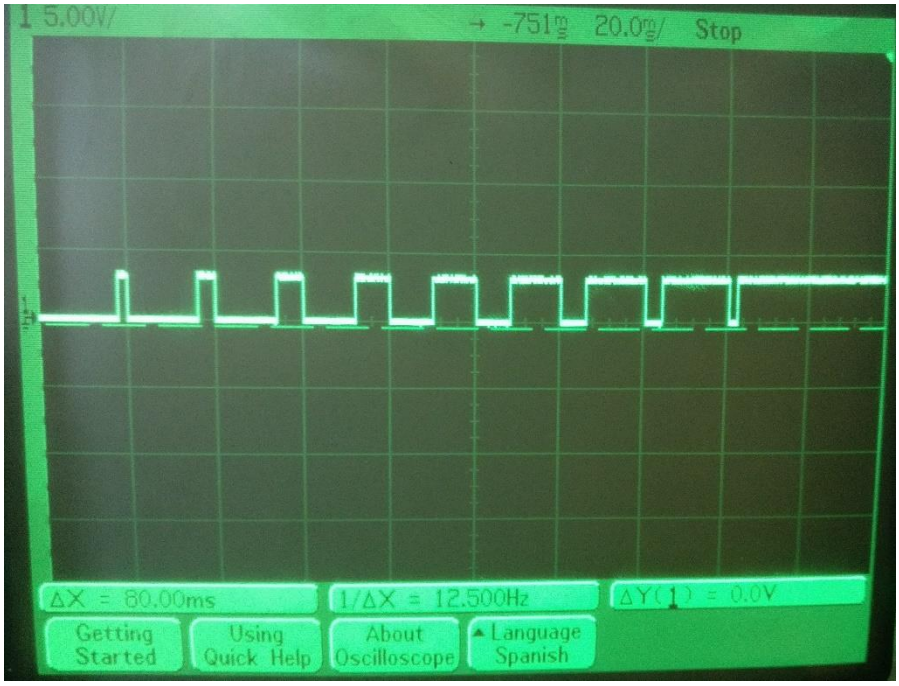
Estarà constantment generant aquests polsos així que no necessitarem en cap moment apagar-lo, per aquest motiu no tenim cap estat d'espera.

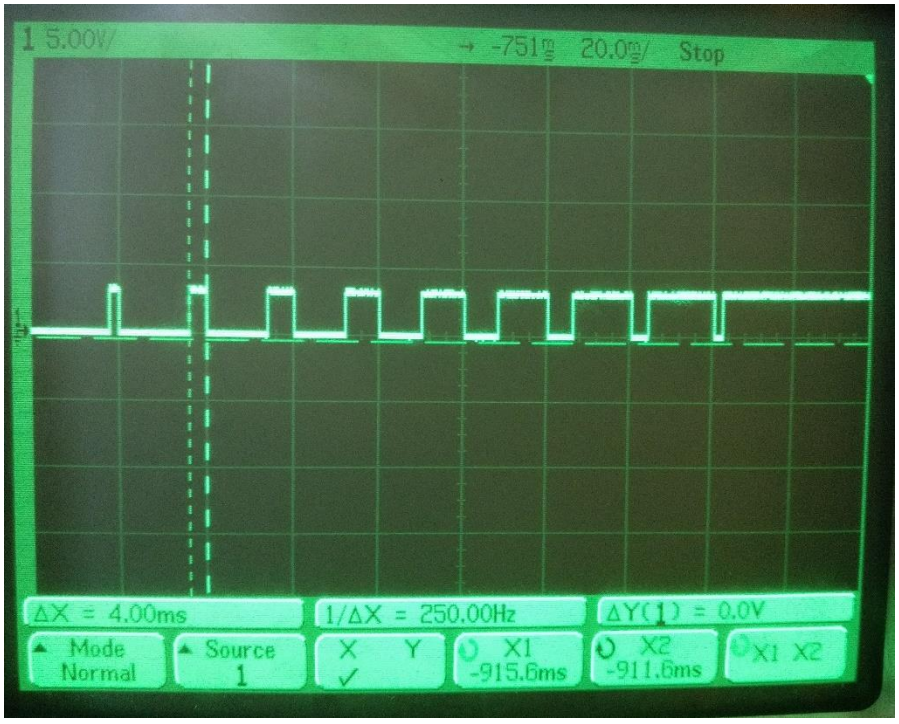
Esta fet perquè sigui el màxim genèric possible, d'aquesta manera pots cridat tants cops com vulguis aquest motor sempre tenint en compte que hauràs de tenir tants PWM com motors vulguis cridar.

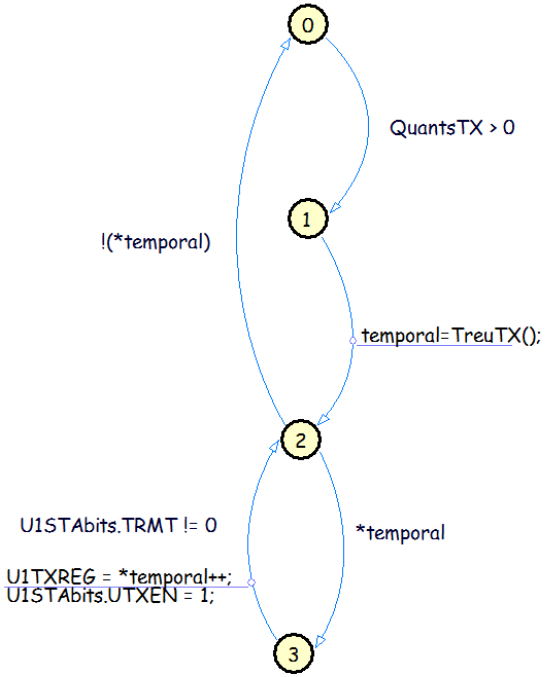
.



Captures PWM



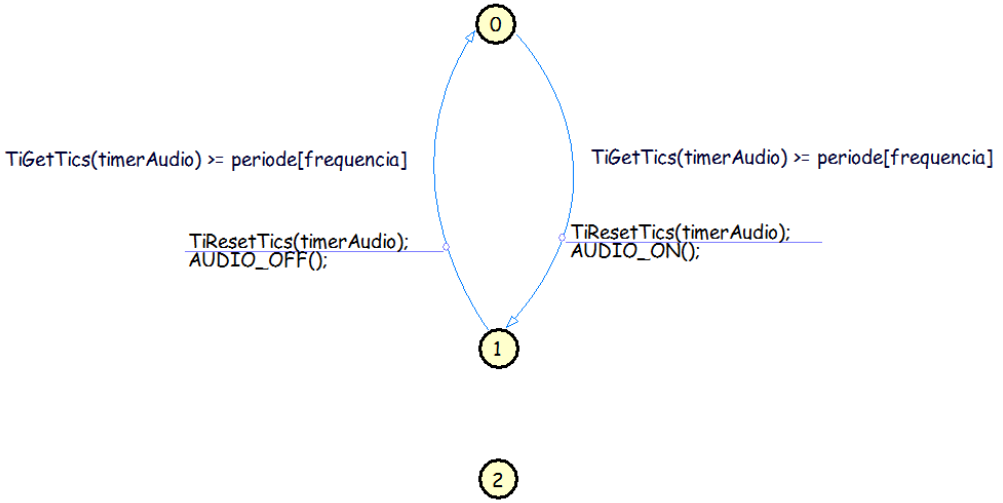




SIO

El funcionament del motor es de la següent manera; quan hi ha elements a enviar en la cua d'enviament, va traient per el port fins que ja no hi queden mes, indefinidament. Per tant no hi ha cap estat que apagui l'escolta dels inputs. A diferencia de per exemple el motor del Audio.

Àudio



Aquest motor te una estructura força simple. Quan es vol encendre el altaveu, s'estableix un període i es posa el motor al estat 0 tot fent un reset del timer.

Aquest, després d'un determinat temps s'activa, el mateix en el cas del estat 1, però en comptes d'activar-lo l'apaga. Així cíclicament fins que es vulgui.

Per desactivar-lo però, aquest motor te un estat d'espera que desactiva el motor fins que l'usuari canvi el estat manualment.

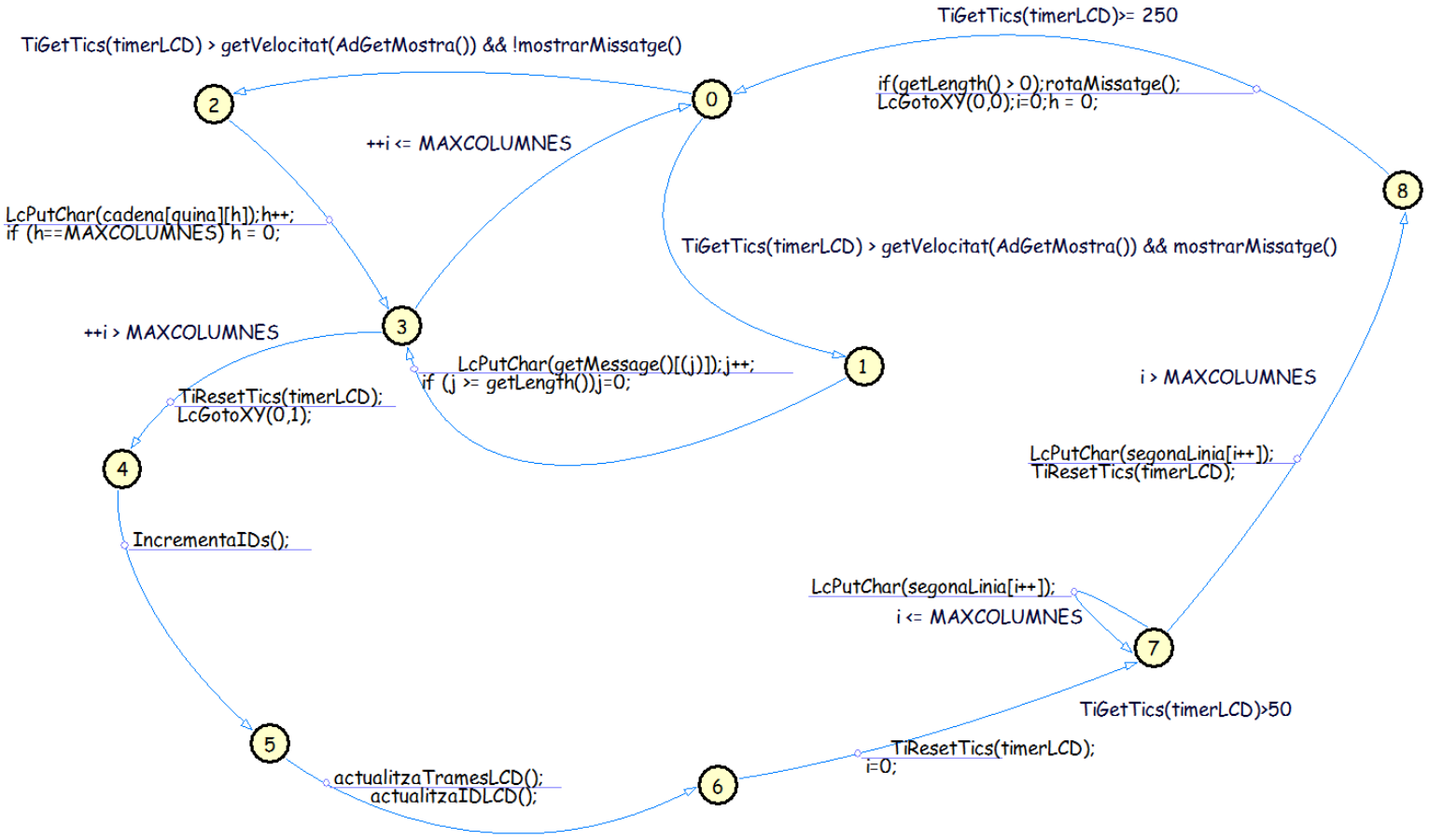
LCD

Aquest motor a diferencia d'altres, esta dintre del TADPropaganda. Ja que el TAD LCD es un TAD simple, sense una part lògica.

Aquest motor, no fa cap tipus de configuració del LCD, nomes es dedica a posar les dades que ha de mostrar. Ja que gracies al TAD LCD, la part de configuració queda oculta per aquest motor.

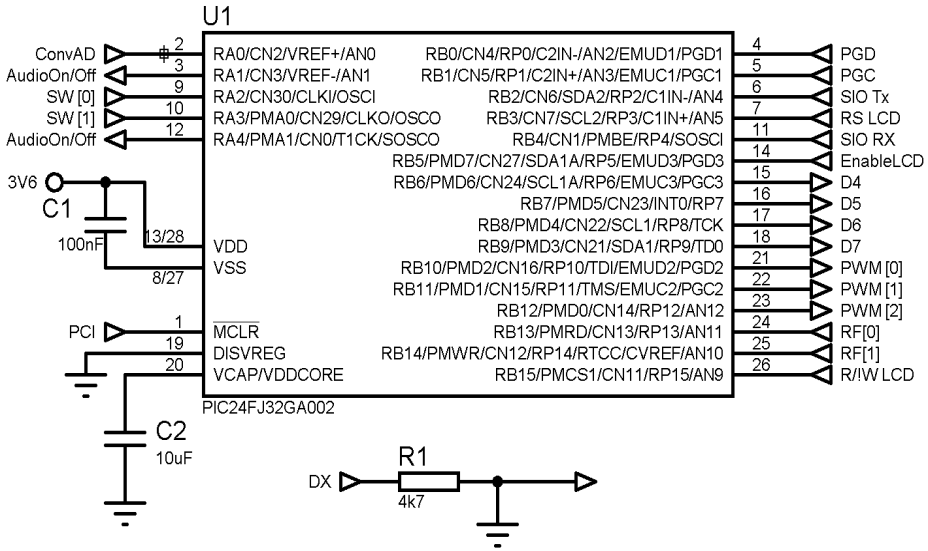
Per tal de modificar el missatge que es mostra en la primera línia del LCD, anirem del estat 0 al 2 o al 1.

Després els següents es dedicaran a afegir la informació de la segona línia, preguntant als TADs corresponents la informació que necessiten.

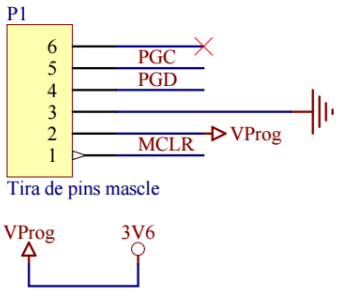


Esquemes elèctrics

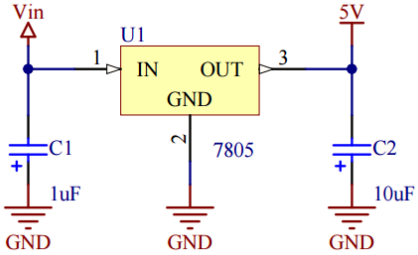
PIC24FJ64GA002

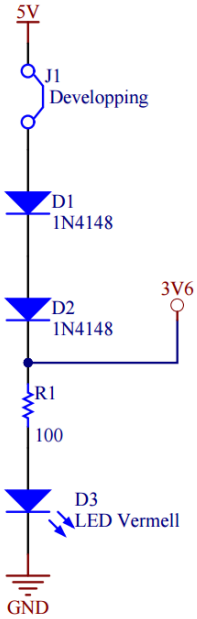


Connector PICKit

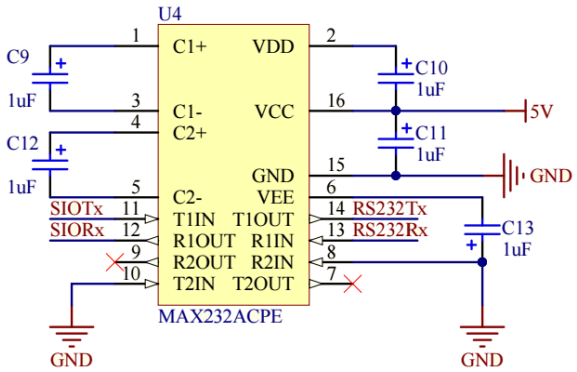


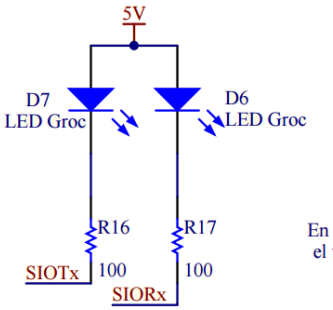
Alimentació

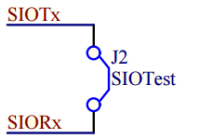




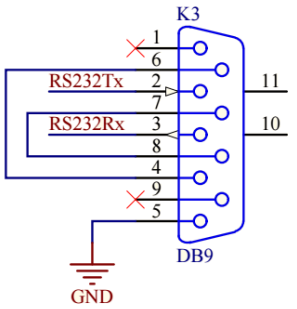
Canal sèrie



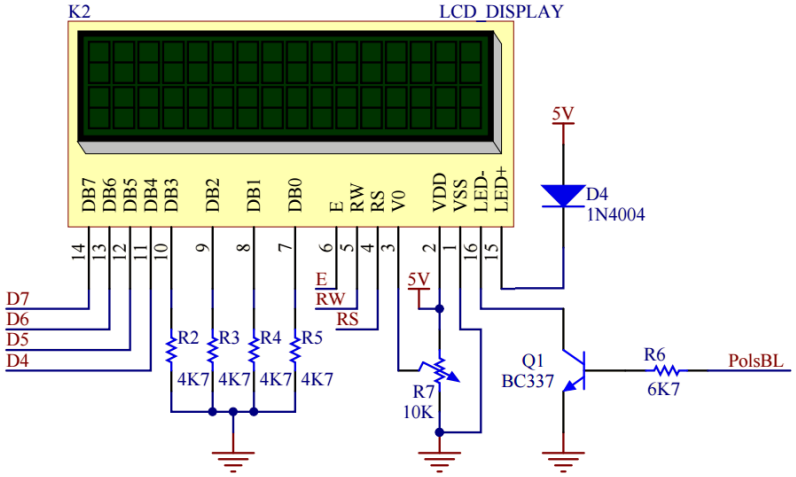




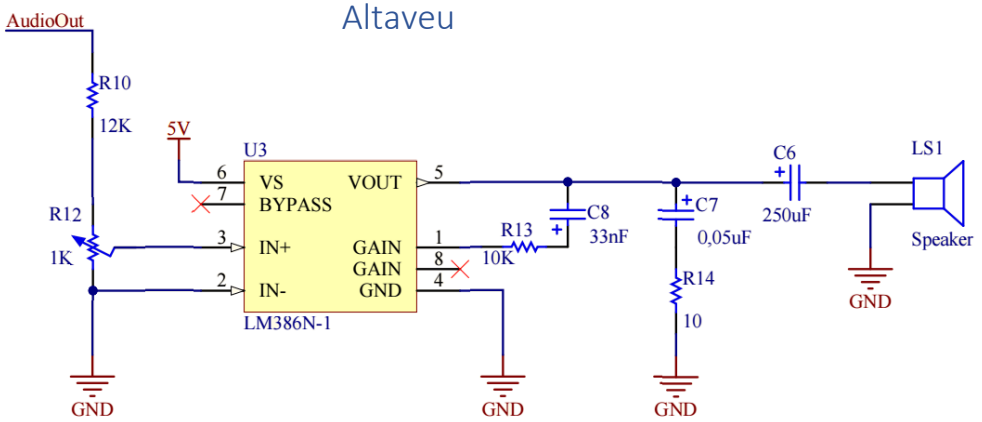
En operació normal estarà obert, el tancarem per comprovar que el MAX232 fa bé la feina



Display LCD



Altaveu



Problemes observats

Com a practiques anteriors, no sempre es tracta d'un camí recta i pla, sinó que sempre hi ha els seus alts i baixos que compliquen la realització de la practica.

En aquest cas, la fase en el moment de soldar a diferencia d'altres fases, es força senzilla. Principalment per la quantitat de xips i components a soldar. Tot i això, hem tingut alguns problemes amb components que no eren els correctes i components que no tenien el valor correcte, però això no es una nova historia, degut a que la solució es posar més atenció a petits detalls.

Així que podríem dir que on hem tingut els problemes mes importants ha estat en la realització dels nostres propis TADs. Perquè per molt que partim d'un projecte base, no impliqui que siguin de fàcil comprensió entendre TADs que no han sigut implementats per nosaltres, però un cop entès el seu funcionament hem tingut que dissenyar els nostres propis TADs per tal d'arribar a l'objectiu final, incrementant considerablement la dificultat.

Posant en banda la implementació i estructura dels TADs, l'apartat on ha suposat una complicació mes elevada ha estat en la recepció de trames per Radio freqüència.

Donat que no estem treballant per un connector que ens permeti tenir uns valors d'interferència mínims o garanteixi la recepció completa d'informació, la recepció i anàlisi de trames per radio freqüència, es més complicada que altres mètodes. Un altre factor que ens ha suposat un problema ha sigut no saber com diferenciar el soroll de la informació que et pot ser útil. Es per aquest motiu que per arribar a crear un protocol fiable i que funcioni en quasi el cent per cent dels casos ha estat a prova i error.

Un cop que vàrem trobar la manera d'assegurar la recepció de les trames i identificar-les correctament, el camí es va alleugerar força, degut a que la part mes complexa ha sigut l'enviament per radiofreqüència. Les parts realitzades a continuació, tot i tenir petits errors de variables, els vam poder fer en un temps relativament curt en comparació amb la recepció per RF.

Conclusions

Un cop hem arribat a la fase final d'aquesta practica ens hem trobat amb una sèrie de dificultats generades per la gran diferencia de planteja-ment i òbviament funciona-ment respecte la fase anterior, clarament degut a que la practica ha seguit la lineal de la teoria donada a classe.

Tot i així, gracies a aquest fet, realitzar una practica que s'apliqui el que estàs aprenent de manera teòrica, et dona la visió de la realitat del que estàs aprenent.

Per el que fa la aportació personal que ha suposat aquesta fase per nosaltres, hem de destacar la gran utilitat vista a l'hora de treballar cooperativament, ja que creiem que ens aportarà un gran valor laboral a l'hora que ens permetrà tenir una nova perspectiva del que pot arribar a fer un microcontrolador programat en c. També considerem que cal destacar que fins aleshores no havíem treballat a tant baix nivell amb C, de manera que aquesta fase ens ha permès ampliar coneixements d'un llenguatge que vam aprendre a més alt nivell.

Respecte la practica 2 en general cal destacar que ens ha fascinat com poden integrar-se dos sistemes tan diferents, sent l'únic requeriment l'ús d'una correcte configuració, obtenim la capacitat de poder treballar amb dos sistemes implementats en llenguatges diferents, de manera que ens ha canviat la perspectiva de treball en equip i la capacitat de realitzar projectes més grans, en que podem treballar amb costos més reduïts, gracies a l'ús d'una tecnologia econòmica i fàcil d'integrar amb altres sistemes.

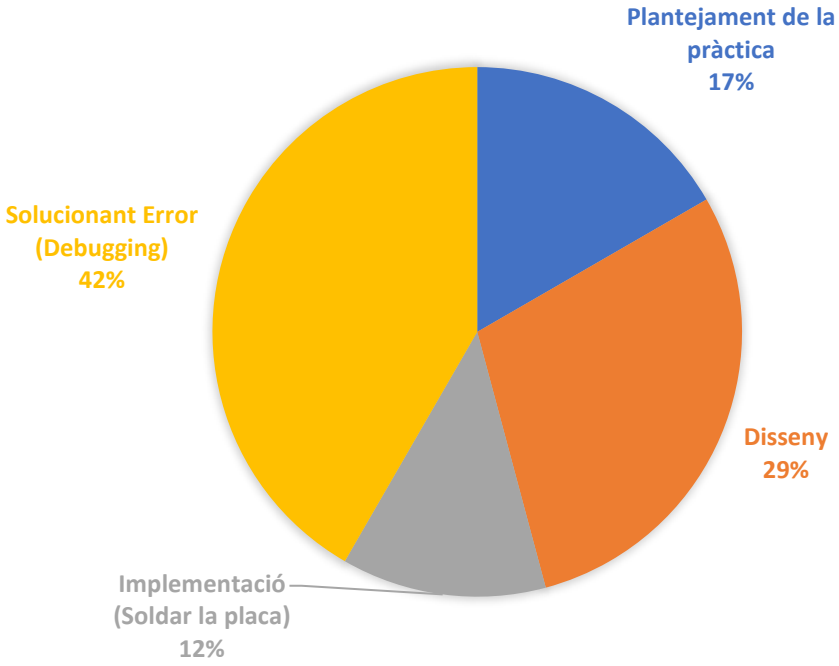
Finalment concloem que aquesta practica ha sigut de gran ajuda per aclarir i entendre molts conceptes vistos a teoria, a part de que ens ha ajudat a comprendre els avantatges i inconvenients de fer servir el llenguatge c com a llenguatge de programació de la PIC i el paradigma de la programació col·laborativa.

Planificació

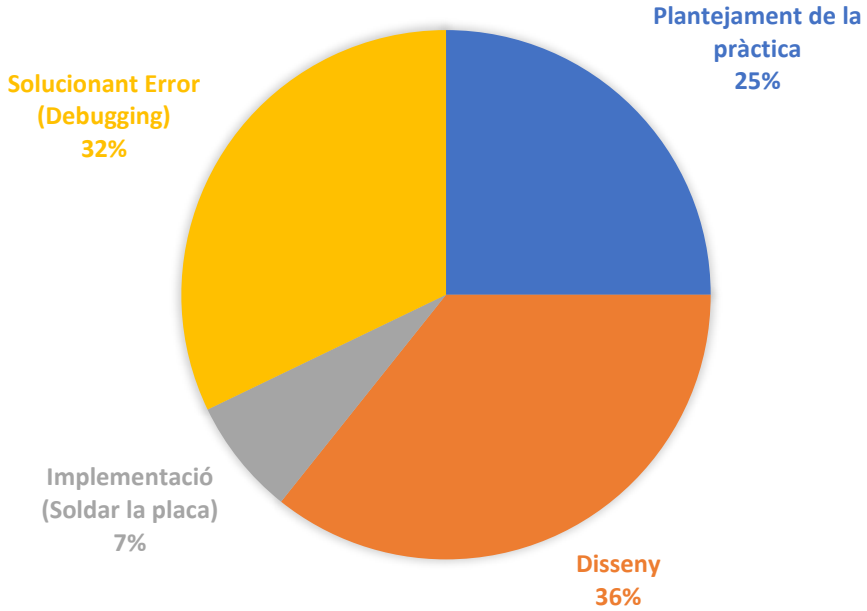
Fases de desenvolupament	Inicial (hores)	Final (hores)
Plantejament de la pràctica	4	7
Disseny	7	10
Implementació (Soldar la placa)	3	2
Solucionant Error (Debugging)	10	8

Tal com es pot veure en aquesta taula la majoria de fases ens ha suposat una dificultat i temps superior al planificat inicialment, menys en el cas de soldar la placa, ja que ens va semblar bastant senzill soldar la placa, però en conseqüència vam tenir un parell de problemes amb alguns perifèrics que va fer que triguéssim més en tenir la placa operativa.

Inicial



Final



Annex

Vídeo Part 1

https://www.youtube.com/watch?v=UY6TVsqxqp4&feature=youtu.be

Vídeo Part 2

https://www.youtube.com/watch?v=T9nzb5akC-Q&feature=youtu.be