Pràctiques de Sistemes Digitals i Microprocessadors

Curs 2017-2018

Pràctica 2

El Tele Generador Intel·ligent de Funcions (TGIF)

	Login	Nom
Alumnes	Ls30652	Gabriel Cammany Ruiz
	Ls30759	Samuel Tavares Da Silva

	Placa	Memòria	Nota
Entrega			

Data	08/04/2018

Portada de la memòria

Pràctiques de Sistemes Digitals i Microprocessadors

Curs 2017-2018

Pràctica 2

El Tele Generador Intel·ligent de Funcions (TGIF)

	Login	Nom
Alumnes	Ls30652	Gabriel Cammany Ruiz
	Ls30759	Samuel Tavares Da Silva

	Placa	Memòria	Nota
Entrega			

Data	08/04/2018

Portada de la memòria

Índex

Síntesi de l'enunciat

Tal com indica el nom del projecte, tele generador intel·ligent de funcions, mitjançant dues estacions, estació base i estació remota.

Per un costat l'estació base serà l'encarregada de gestionar la funció desitjada per l'usuari amb tots els paràmetres necessaris i transmetre-ho a l'estació remota. Aquesta sempre romandrà a l'espera de rebre la informació corresponent. D'aquesta manera, l'estació remota serà l'encarregada de generar la funció rebuda.

Per un costat l'estació base (fase 1), es trobarà connectada a un PC, el qual permetrà a l'usuari configurar els paràmetres necessaris per generar el senyal corresponent. Aquests seran el tipus de senyal, valor de pic a pic, offset i freqüència. Aquest PC utilitzarà una interfície Java i es comunicarà en sèrie amb el microcontrolador, utilitzant un connector DB9 amb protocol RS-232. Per un altre costat el microcontrolador haurà de gestionar la comunicació en sèrie i la transmissió per Ràdio Freqüència.

L'estació remota se centrarà a rebre el senyal amb els paràmetres corresponents que l'usuari haurà enviat des de l'estació base, mitjançant la interfície o per mitjà d'un polsador, que enviarà l'última senyal desada. Un cop obtinguda la informació del senyal l'estació remota el generarà. Destacar que per tal de poder parametritzar i veure certs aspectes estadístics referents a l'enviament per RF, s'utilitzarà un PC connectat a l'estació amb una interfície similar a la del PC de l'estació remota.

Finalment destacar que l'estació base, es troba formada per un microcontrolador programat en assemblar, que ha de realitzar 3 funcions fonamentals. Concretament:

- *Recepció de dades de l'ordinador:* Mode que es donarà quan es premi un botó de la placa o l'usuari ho indiqui per la interfície, obtenint la informació pel canal sèrie i emmagatzemant-ho a la memòria del microcontrolador.
  En el moment que es tingui la informació emmagatzemada, es mostrarà un guió baix al 7segments. A part, s'ha de realitzar un blinking a 1hz de cada LED, de manera complementària, en cas que s'enviï la informació per defecte, aquest serà de 10hz.

- *Enviar dades per RF:* Funcionalitat que es donarà quan es premi el corresponent polsador o una comanda de la interfície, iniciant així l'enviament de dades a l'estació remota.

  Destacant, que en cas que no hi hagi cap informació emmagatzemada no es realitzarà l'enviament, es mostrarà un guió en el 7 segments i els LEDs romandran al 50% de la intensitat. En cas contrari el 7 segments mostrarà quin percentatge de transmissió ja ha realitzat i un cop finalitzi la transmissió tornarà a mostrar el que hi havia abans d'iniciar aquesta ordre.

- Respecte la trama a enviar, s'ha de garantir l'enviament del número de grup com a capçalera i tota la informació necessària perquè l'estació generi el senyal.

- *Esborrar dades:* Funció que només es donarà quan es premi dues vegades el polsador de *get_info* (indicat en l'esquema elèctric), esborrant les dades referents a l'últim senyal emmagatzemat, apagar el 7 segments i els LEDs.

Plantejament

Sent l'estació base l'encarregada d'obtenir la informació del senyal a generar i transmetre'l per RF, es pot dividir en dues clares parts.
Per un costat, tenim la part de la interfície Java, que es controla des del PC. Per un altre costat tenim la gestió general, realitzada amb un microcontrolador, en el nostre cas una PIC 18f4321, de manera que gestioni l'emmagatzemament de la informació, la comunicació en sèrie amb el PC i l'enviament per RF, entre altres funcionalitats fonamentals pel correcte funcionament de l'estació.

Respecte a la interfície Java, està realitzada amb la llibreria de Java Swing/AWT. Partint d'un projecte base amb una sèrie de funcions a implementar, les vam completar i ampliar, afegint classes i elements a la interfície per facilitar el debug o garantir el correcte funcionament de la placa.

Per altra banda, pel protocol de comunicació amb la PIC, ens hem ajudat de threads, per tal de poder garantir el funcionament de dues funcionalitats diferents.
Una part està constantment escoltant pel port per si rep un byte de la PIC, indicant que comences a enviar les dades i per altra banda teníem la part dels Listeners dels botons i la selecció de certs paràmetres de configuració de la transmissió i el senyal. Sent aquesta part, l'encarregada d'esperar que l'usuari premi el boto d'enviament per RF o carrega dades, enviant a la PIC la informació mitjançant el canal sèrie.

Relacionat amb el funcionament del canal sèrie, cal destacar que utilitzem un cable USB a DB9, també conegut com a interfície RS232 i utilitzem el pin de Ring per poder saber si el cable es troba connectat a la placa, per tant un cop es detecta aquest pin s'encén un LED i es notifica a la interfície.

Per tal de realitzar la comunicació en sèrie entre el PC i la PIC, hem hagut de realitzar una capçalera i utilitzar flags. El funcionament és bastant senzill, la interfície prepara una trama amb el flag corresponent a l'acció que el microcontrolador ha de fer.
Un cop tenim la trama a punt, utilitzem la llibrera SerialPort1.8 per poder enviar byte a byte i el PC esperarà la recepció d'informació de la PIC, confirmant així la recepció o sol·licitant més informació. Destacar que els flags utilitzats es representen en una classe Java i també estan en el codi assembler.

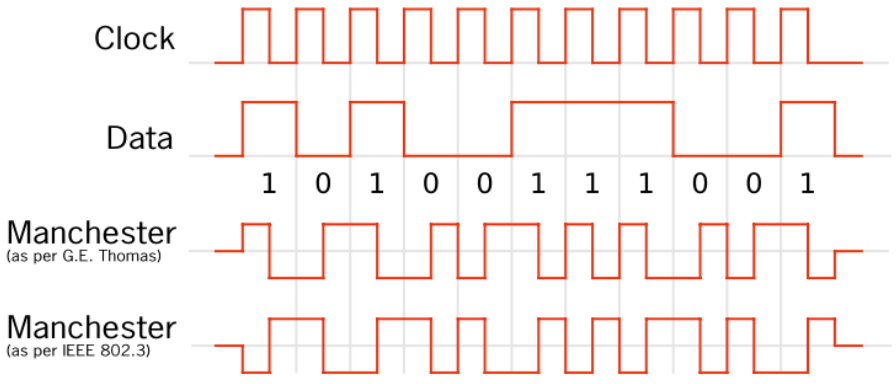
Aquests flags també són fonamentals pel funcionament dels polsadors, ja que amb dos polsadors hem de realitzar 4 accions diferents, ja que les accions interactuen amb la interfície.

Partint que cada senyal es representa a partir d'un vector de 300 mostres, en el nostre cas, quan el senyal a carregar és simètric només enviarem la meitat de les mostres, optimitzant així l'emmagatzemament de la informació en la RAM. Un cop tenim la trama, formada per el flag_desar, el vector de mostres i un flag_end, d'aquesta manera el microcontrolador cridarà al bloc d'emmagatzemament, encarregat d'escriure a la ram fins rebre un flag_end.

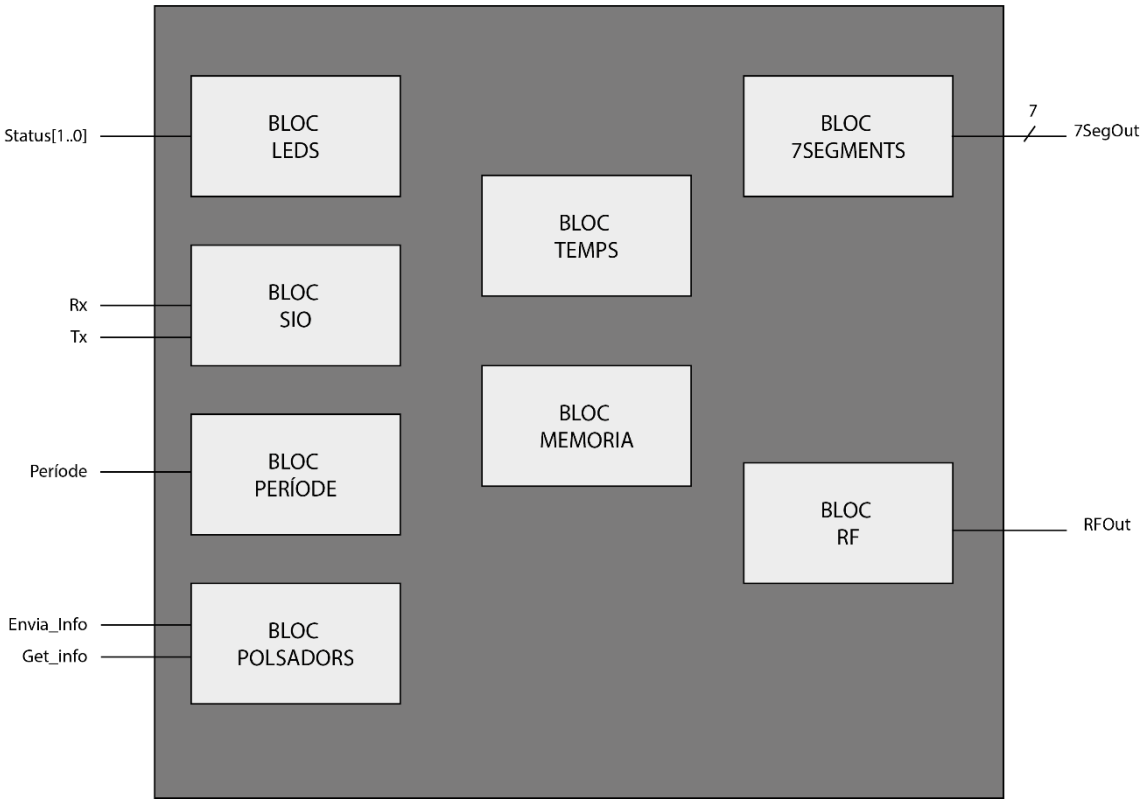
Durant i un cop finalitzi la transmissió, mitjançant un LED de tres díodes, realitzarem un Blinking de diferent freqüència, segons com transcorri l'emmagatzemament.

Finalment respecte a l'enviament per RF (Ràdio Freqüència), ens ajudem d'un pin de sortida, el qual mitjançant un protocol Manchester enviem la informació emmagatzemada quan hem carregat la funció. Durant l'enviament per RF, hem d'anar actualitzant el nombre de bytes transmesos a la interfície i el 7 segments, ja que a la interfície s'ha de veure una progress bar amb el percentatge transmès i a l'estació base s'ha de veure el mateix valor per un 7 segments.

Com es pot veure en la següent figura, el protocol Manchester en el nostre cas el (IEEE 802.3), estableix que el valor 1 en binari correspon a la meitat del període a 0 i l'altre a 1. Sent el contrari en el cas del 0.



Diagrames de mòduls



- **Bloc LEDs-** Bloc encarregat de la gestió dels LEDs status. En el nostre cas ho gestionem amb un díode multicolor, connectat directament al microcontrolador, que realitzarà el control corresponent.
- **Bloc SIO-** Destinat a la gestió de la transmissió i recepció sèrie amb l'ordinador. Caracteritzat per utilitzar una interfície RS-232, mitjançant un cable USB a DB9. El propòsit general d'aquest bloc serà la comunicació amb l'usuari, mitjançant un PC que executarà una interfície Java.
- **Bloc Període-** Amb un potenciòmetre com a input, aquest serà gestionat pel AD del microcontrolador. Per tant, aquest bloc s'encarregarà de realitzar la conversió del valor analògic a digital, sent aquest valor digital el corresponent amb la freqüència de recepció de dades.

- **Bloc Polsadors-** Bloc destinat a la gestió dels polsadors i la lògica necessària per al correcte funcionament. Destacant que amb els dos polsadors directament connectats, es podrà enviar per RF, carregar un senyal per defecte i esborrar l'últim senyal emmagatzemat.
- **Bloc Temps-** Tal com indica el nom, es destina a la gestió del temps. Sent aquest bloc fonamental pel funcionament del microcontrolador, ja que és l'encarregat de realitzar la configuració de la interrupció del timer.
- **Bloc Memòria-** Encarregat de l'emmagatzemament d'informació del senyal carregat al microcontrolador. Per tant, inclou les funcions necessàries per guardar i llegir de la memòria interna del microcontrolador.
- **Bloc 7segments-** Bloc destinat a la gestió del 7 segments, ajudat pel microcontrolador permetrà veure quin percentatge de dades transmeses per RF portem.
- **Bloc RF-** Bloc format pel conjunt de funcions necessàries per a l'enviament de dades per ràdio freqüència, ajudant-nos d'un protocol Manchester i una capçalera única que permeti identificar les trames a l'estació remota.

Disseny

Configuracions del microcontrolador

Les nostres configuracions principals son les següents:
Hem configurat el oscil·lador a 40Mhz, utilitzant el l'intern de 10Mhz amb el PLL. És a dir, el multiplicador que fa que de 10Mhz el multipliquem per 4 per poder arribar als 40Mhz. Tot i així, és podria fer perfectament aquesta fase sense l'ús del PLL.

```
CONFIG OSC = HSPLL
CONFIG PBADEN = DIG
CONFIG WDT = OFF
```

Donat que necessitem el port B en comptes d'analògic a digital, necessitarem definir el PBADEN a digital. Per un altre part, la configuració del WDT a OFF ens dóna l'opció de tenir un bucle infinit sense un reset automàtic del microcontrolador.

Timer

Un dels mòduls més importants, el Timer (El 0 en el nostre cas) és l'encarregat de portar el recompte de mil·lisegons. Com ja hem vist en les anteriors entregues, aquest és un comptador que a partir del clock del sistema, genera una interrupció a partir d'un cert número de cicles.

Com ja hem comentat anteriorment, en el nostre disseny del sistema, hem utilitzat aquest timer per realitzar el comptatge dels ms passats durant un instant de temps concret. Per aquest motiu, la rutina de servei d'interrupció d'aquest ha estat bastant curta, ja que es dedicava a incrementar variables que s'utilitzaven en diferents parts del codi.
Les configuracions d'aquest son les següents:

- **T0CON: TIMER CONTROL REGISTER**
  - 1 - TMR0ON: Habilitar el timer
  - 0 - T08BIT: Desactivat per timer de 16 bits
  - 0 - T0CS: La transició amb el rellotge d'instrucció
  - 0 - T0SE: Canvi de baix-alt
  - 1 - PSA: Desactivar el prescaler
  - 000 - T0PS2:T0PS0: No utilitzem prescaler
- **RCON: RESET CONTROL REGISTER**
  - 0 - IPEN: Des habilitar prioritat interrupcions
- **INTCON: INTERRUPT CONTROL REGISTER**
  - 1 - GIE/GIEH: Des habilitar prioritat interrupcions
  - 0 - PEIE/GIEL: Des habilitar interrupcions perifèrics
  - 1 - TMR0IE: Habilitem timer0
  - 0 - INT0IE: Des habilitar interrupció externa timer0
  - 0 - RBIE: Des habilitem interrupció canvi PORTB
  - 0 - TMR0IF: Flag interrupció timer0
  - 0 - INT0IF: Flag interrupció externa int0
  - 0 - RBIF: Flag canvi estat pins PORTB

```
INIT_TIMER
    ;10001000
    movlw 0x88
    movwf T0CON,0
    bcf RCON, IPEN, 0
    ;10100000
    movlw 0xE0
    movwf INTCON, 0
    call RESET_TIMER
    return
```

ADC

Per poder realitzar la selecció del període, com ja s'ha comentat en la síntesi del enunciat, s'utilitzarà un potenciòmetre. Aquest definirà mitjançant el canvi del voltatge a partir de la seva resistència variable un valor que utilitat un ADC, estableix la freqüència.

Les configuracions d'aquest son les següents:

- **ADCON0: A/D CONTROL REGISTER 0**
  - 00 - Sense implementar
  - 0000 - CHS3:CHS0: Seleccionar l'entrada analògica 0 (AN0)
  - 1 - GO/!DONE: Flag per començar conversió
  - 1 - ADON: Habilitar el mòdul AD
- **ADCON1: A/D CONTROL REGISTER 1**
  - 00 - Sense implementar
  - 0 - VCFG1: Voltatge negatiu de referencia, en el nostre cas els 0 Volts (Massa). Ja que el potenciòmetre estarà connectat al sistema d'alimentació del micro.
  - 0 - VCFG0: Voltatge negatiu de referencia, en el nostre cas els 5 Volts.
  - 000 - Ja que nomes volem AN0 com entrada analògica.
- **ADCON2: A/D CONTROL REGISTER 2**
  - 1 - ADFM: Ajustem a la dreta el valor, es a dir, en el ADRESL hi haurà el valor mes petit i en el ADRESH la part alta.
  - 0 - Sense implementar
  - 001 - ACQT2:ACQT0: Establim un temps de 2 T_{AD}. Ja que mirant les configuracions i la documentació establerta per el datasheet, és el valor mes recomanat, ja que el T_{AD} ha de ser el mes curt possible.
  - 000 - ADCS2:ADCS0: La freqüència que utilitzarà el conversor AD serà de $T_{osc}/2$. Establint la més rapida possible.
  - Ports

```
INIT_ADCON

    movlw 0x01
    movwf ADCON0,0

    movlw 0x0E
    movwf ADCON1,0

    movlw 0x84
    movwf ADCON2,0

    return
```

EUSART

Un perifèric important per aquesta practica es tracta de la EUSART. Com ja sabem, aquest es tracta d´un mòdul per realitzar la comunicació en sèrie.

Com que necessitàvem aquest mòdul per poder efectuar la comunicació en sèrie cap al ordinador, l´hem usat amb les següents configuracions.

```
INIT_SIO
    movlw 0x90
    movwf RCSTA,0
    movlw 0x26
    movwf TXSTA,0
    movlw 0x81
    movwf SPBRG,0
    clrf SPBRGH,0
    clrf BAUDCON, 0
    return
```

- **TXSTA: TRANSMIT STATUS AND CONTROL REGISTER**
  - 0 - CSRC: No es important, ja que utilitzem comunicació asíncrona.
  - 0 – TX9 Enviem amb 8 bits
  - 1 – TXEN Activem el enviament
  - 0 – SYNC Establim mode asíncron.
  - 0 - SENDB: No volem que envii el caràcter de nova linea.
  - 1 - BRGH: Activem per definir un baudrate alt.
  - 0 – TMRT: Flag per comprovar si s'ha enviat, no necessari en el moment de configuració.
  - 0 - TX9D: Bit extra en el mode de 9 bits, no importa el valor.
- **RCSTA: RECIEVER STATUS AND CONTROL REGISTER**
  - 1 - SPEN: Activem el port serie
  - 0 – RX9 Enviem amb 8 bits, per tant 0
  - 0 – SREN Estem mode asíncrona, no ens importa
  - 0 – CREN Activem el receptor
  - 0 - ADDEN: Anem amb mode de 8 bits, per tant ens es igual.
  - 1 - FERR: Flag, per tant en configuració no es important.
  - 0 – OERR: Com en el cas anterior, es un flag i per tant no es important en la configuració.
  - 0 – RX9D: Bit extra en el mode de 9 bits, no importa el valor.

Baud Rate

EL registre SPBRG li hem de posar en la part baixa, 0x81 i en la part alta tot zero.

Després de configurar la SIO, necessitem definir el baudrate, és a dir, la velocitat de l'enviament de les dades.
Nosaltres hem definit el nostre a 19200 bps. Per poder anar a aquesta velocitat ho hem configurat de la següent manera.

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 0											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	—	—	—	—	—	—	—	—	—
2.4	—	—	—	—	—	—	2.441	1.73	255	2.403	-0.16	207
9.6	9.766	1.73	255	9.615	0.16	129	9.615	0.16	64	9.615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19.230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55.555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

El registre SPBRG li hem de posar en la part baixa, 0x81 i en la part alta tot zeros.

Ja que com podem veure en la següent imatge, la nostra freqüència d'oscil·lació va a 40Mhz. Per tant, per anar a 19200bps, necessitarem afegir 129 decimal (81 en hexadecimal) al registre de SPBRG. Hem escollit aquesta configuració, ja que podem anar a una velocitat considerable tot tenint un error de 0.16%.

En la part de configuració, es a dir, el registre BAUDCON, l'hem posat tot a zeros ja que:

- No necessitem les dades invertides (RXDTP,TXCKP).
- Com que 129 es pot definir en 1 byte, no necessitarem la part alta del registre SPBRG. I per tant el bit BRG16 ha d'anar a 0.

Els altres bits no són d'importància per aquesta configuració.

PORTS

PORTA:
Per al port A, hem utilitzat els següents pins:

- RA0 – Per l'entrada analògica del potenciòmetre.
- RA6-RA7 – Entrades OSC1 i OSC2 per al oscil·lador extern de 10Mhz.

Per tant, utilitzant el registre TRISA, el posat el bit 0 a 1 com a entrada.

PORTB:
Hem utilitzat 4 pins del Port B. Tal com es mostra en la figura:

- RB0 – Entrada per la connexió del RS-232
- RB1 – Entrada polsador envia info
- RB2 – Entrada polsador desa info
- RB3 – Sortida LED connectat

Com a cas particular, hem activat les pull-up que el PortB té en els pins de menys pes. Per fer-ho s'ha netejat el pin RBPU del registre INTCON2.

PORTC:
Hem dedicat l'ús d'aquest port per el enviament de les dades per radiofreqüència i els LED de status.

PORTD
Finalment, s'ha utilitzat quasi tot el port D per controlar el 7 segments. Exceptuant el RD6.

```
INIT_PORTS
    ;clrf TRISA, 0
    bcf TRISC, 5, 0     ;RF Out
    bcf TRISC, 3, 0     ;LED0
    bcf TRISC, 2, 0     ;LED1

    bsf TRISB, 0, 0     ;SIO IN
    bsf TRISB, 1, 0     ;Envia Info
    bsf TRISB, 2, 0     ;Desa Info
    bcf TRISB, 3, 0     ;LED SIO

    bcf  TRISD, 0, 0 ; 7SEG1
    bcf  TRISD, 1, 0 ; 7SEG2
    bcf  TRISD, 2, 0 ; 7SEG6
    bcf  TRISD, 3, 0 ; 7SEG3
    bcf  TRISD, 4, 0 ; 7SEG4
    bcf  TRISD, 5, 0 ; 7SEG5
    bcf  TRISD, 7, 0 ; 7SEG7

    bcf SSPCON1,SSPEN,0

    bcf INTCON2,RBPU,0

    setf LATD,0

    clrf LATC,0

    return
```

Diagrama de seqüencia del software

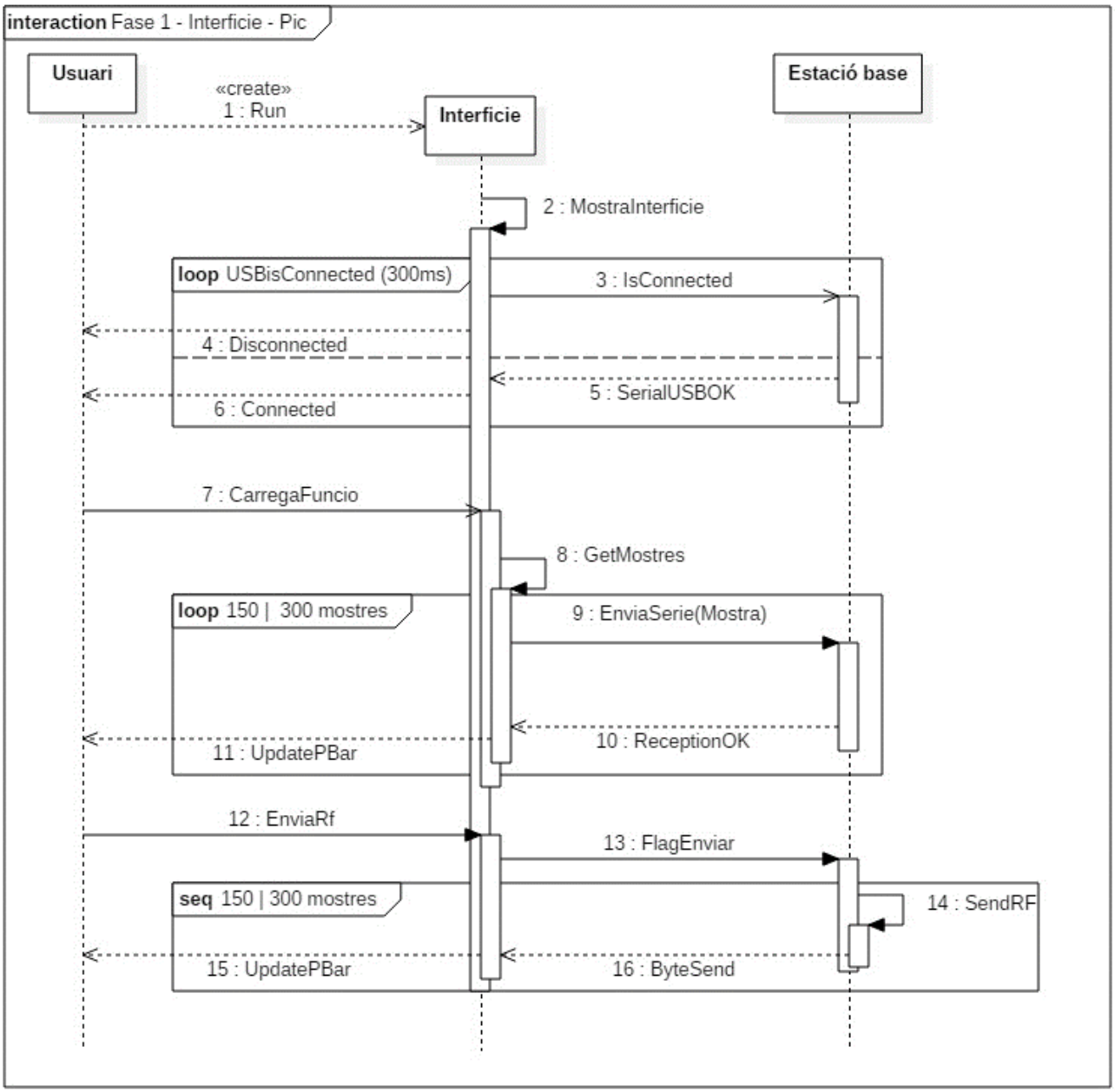
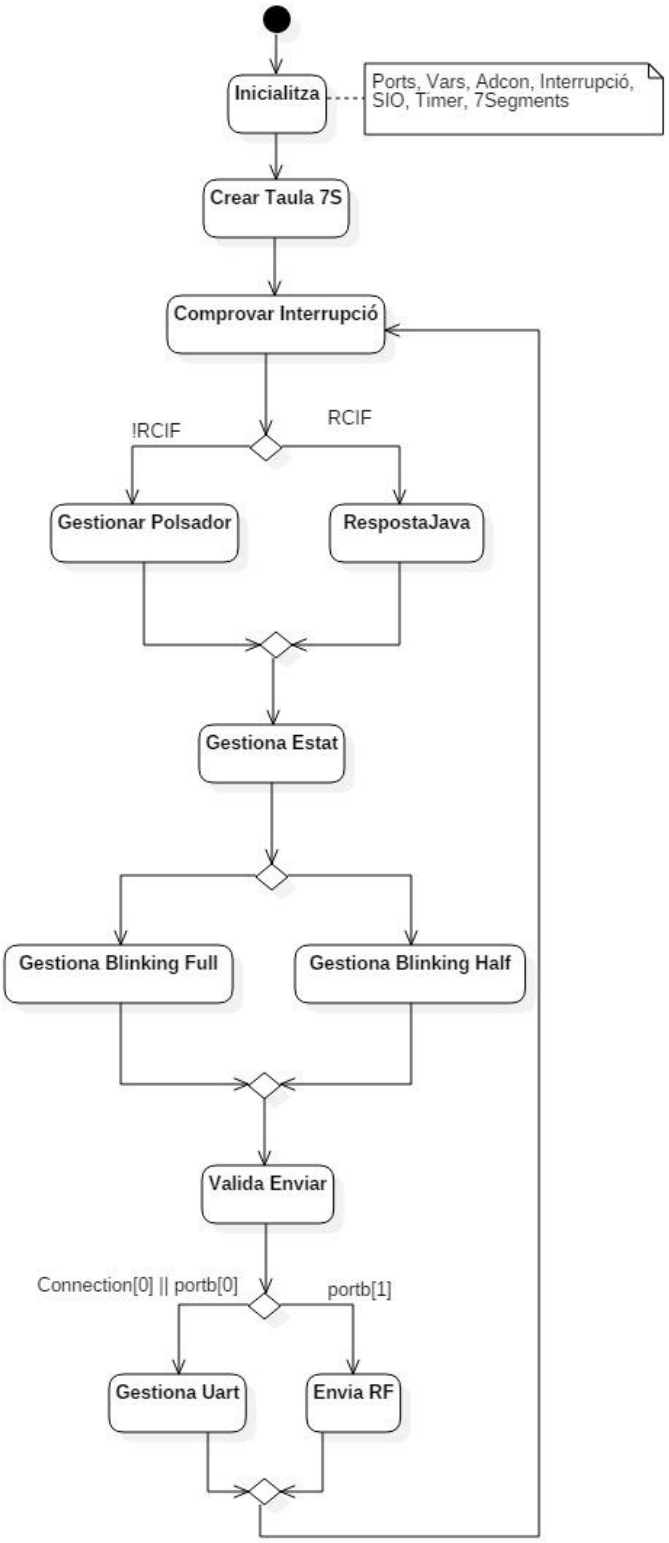


Diagrama d'activitat del software del microcontrolador



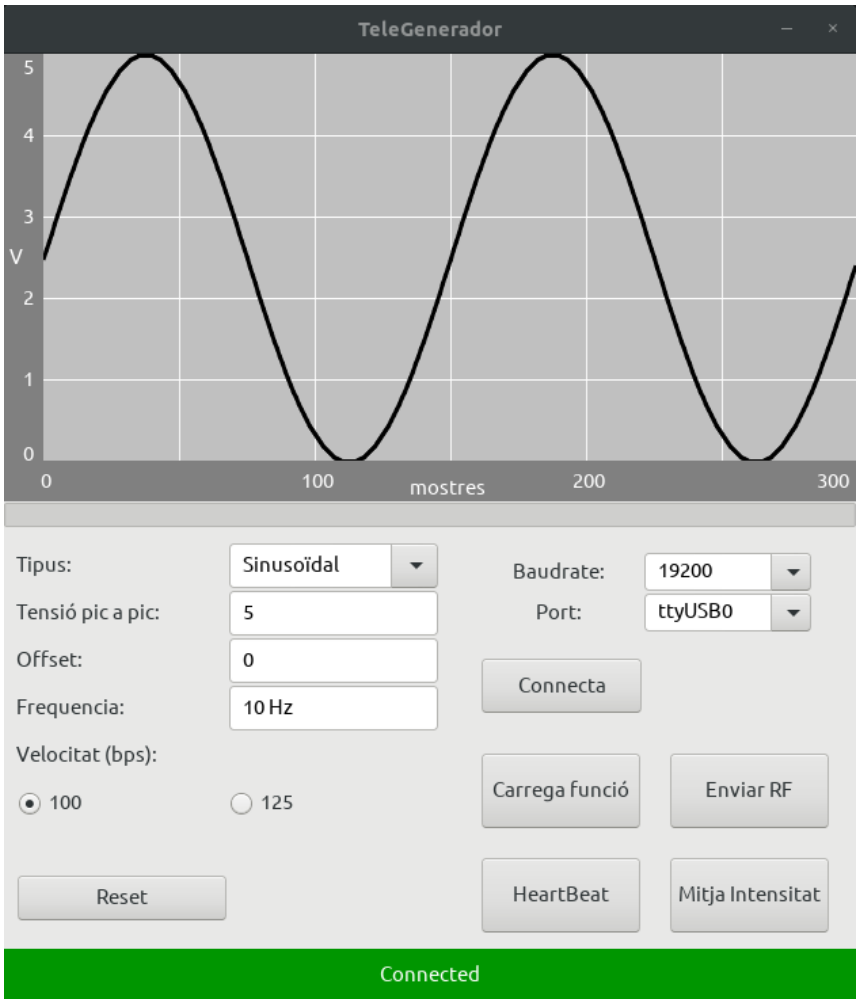
Disseny del software de l'ordinador

El software per la part del ordinador conté els següents processos explicats a continuació.

Procés principal

Com be indica el seu nom, aquest es l'encarregat de gestionar la interfície i el control sobre els dos processos que explico a continuació.

Interfície
La interfície es mostra de la següent manera.



Les funcionalitats d'aquesta a part de les ja implementades a partir del software base, s'han afegit algunes per millorar el control de l'estació base.
Per una banda, tenim la part de botons de control, com és el de càrrega la funció, enviar RF, heartbeat i mitja intensitat. I altre més de control bàsic com el de reset i connecta.

La raó d'afegir aquests dos botons ha estat per poder tenir en la interfície totes les possibles funcionalitats que es poden fer des de l'estació base.
Per altra banda, connecta l'utilitzem per evitar problemes d'execució. Ja que el fet de connectar directament amb el port sèrie tan bon punt executem el programa, pot portar a casos on es genera un error per la consola del programa. Per evitar-ho, hem decidit que només s'obrirà aquest port quan l'usuari hagi connectat bé les interfícies i així evitar possibles errors.

Això no treu el fet que, si l'usuari està connectat al sistema i aquest es desconnecta no ho visualitzi per pantalla (la barra inferior canvia de color) com també, no és necessari prémer el boto un cop es connecta de nou. És per tant una qüestió d'inicialització que només serveix per a la primera part de l'execució del programa.

Lògica

Un cop explicada la interfície, a continuació estableixo el funcionament lògic d'aquesta i el control sobre la placa.
Com a procés principal i prioritari, és necessari establir un control estricte sobre els altres dos. Aquest fet fa que, quan es premi qualsevol botó que s'hagi de comunicar directament amb l'estació i necessiti qualsevol resposta, es mataran i es reinicialitzaran un cop s'hagi acabat la petició.

```
reciever.interrupt();
timer.interrupt();
while (reciever.isAlive()) ;
while (timer.isAlive()) ;

public void restart() {
    try {
        reciever.join();
        reciever = new Thread(portThread);
        reciever.start();
        timer.join();
        timer = new Thread(connectionThread);
        timer.start();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Per poder dur-ho a terme, s'ha utilitzat interrupcions. Com es pot veure, en la primera instància, es realitzaran les interrupcions que els dos detectaran i finalitzaran l'execució. Un cop co mprovada la finalització d'aquestes es procedeix a gestionar la interrupció del botó.

Després, es cridarà a la funció restart que comprovarà que els recursos s'han alliberat pels dos processos i es crearà el thread de nou.
La raó principal d'aquest procediment a l'hora d'executar qualsevol petició de l'usuari cap a l'estació és com que la llibreria SerialPort manté el procés en espera quan es fa el read del port sèrie. Per tant, per evitar que l'altre procés consumeixi la resposta que espera el principal es procedeix a apagar-ho.
En el cas del timer però, es dóna per evitar canvis del color mentre s'executa una funció que triga més que el temps d'actualització.

Principalment, a part d'un conjunt de funcions senzilles que envien els flags definits per tant l'ordinador com l'estació base, hi ha tant la petició d'enviar per ràdio freqüència com la de desar les dades al microcontrolador.
Cal dir però, que no es realitzarà cap tipus de petició si el microcontrolador no està connectat, així ens evitem bucles infinits esperant resposta del micro.

Desar

Quan l'usuari premi el botó de desar les dades, aquest es dedicarà en una primera instància a enviar primer de tot els bytes bàsics, com serien:

- Part alta del ID de grup
- Part baixa
- Freqüència de la senyal en el moment que ha premut el botó.

```
for (byte value : utf8Bytes) {
    sp.writeByte(value);
    while ((recieved = sp.readByte()) == 0) ;

    view.changeProgressBar(false);
    view.setGreenStatus();
}
```

Un cop rebuts aquests primers bytes, enviarà de manera iterativa totes les mostres de l'array, sempre esperant confirmació que ha rebut cada mostra. Ja que si el microcontrolador no és capaç de gestionar-ho a temps, les dades desades no seran correctes.

Enviar RF

L'altre bloc important es tracta de la petició per radiofreqüència. Aquesta te més importància el fet de fer gestions abans d'enviar la petició que no el fet de fer-ho. Ja que per tal de realitzar-ho, necessitem abans comprovar que no hem desat cap dada al microcontrolador.

Per tant, haurem de fer una petició de comprovació de les dades abans. A partir de la confirmació de si hi ha, també coneixem quin tipus de funció és a partir de la resposta, que ens ajuda a definir el valor de la barra de progrés.

```
if ((recieved = dadesDesades()) > 0) {
    view.changeProgressBar(true);
    if (recieved > 15) {
        view.setProgressBarStatus(2400);
    } else {
        view.setProgressBarStatus(1200);
    }
    enviarPeticioRF();

} else {
    enviarErrorPeticioRF();
}
```

Procés secundari per escolta del port sèrie

Com bé indica el seu nom, l'objectiu principal d'aquest serà la de l'escolta del port sèrie. Donat que es necessita que hi hagi un procés constantment a l'espera a una lectura del port sèrie per si rebem una petició per part del micro.
Per tant, l'estructura principal d'aquest procés serà la d'espera en un bucle infinit mentre no rebi cap interrupció.
Al rebre qualsevol dada pel port, aquest comprova dintre d'una llista de flags possibles si hi ha algun que compleix amb els valors esperats. En cas afirmatiu, gestionarà la petició pertinent.

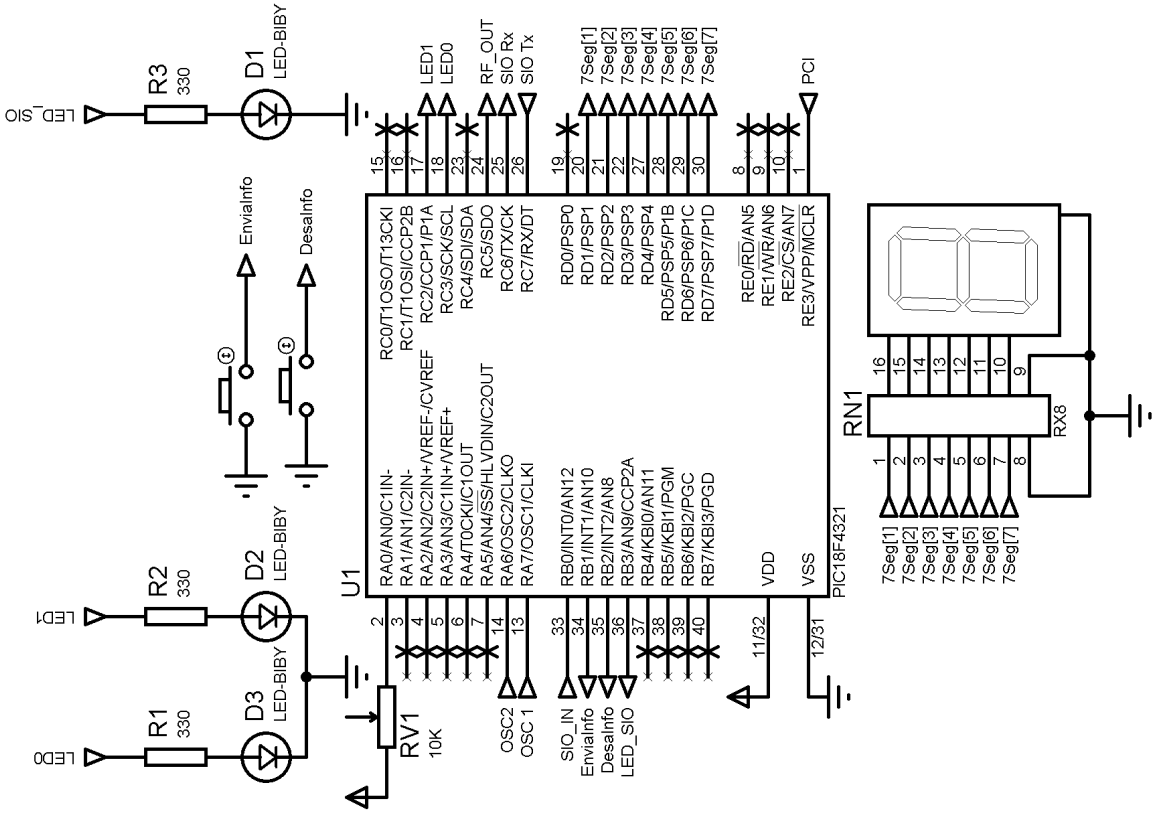
Procés dedicat a la comprovació de la connexió del port.

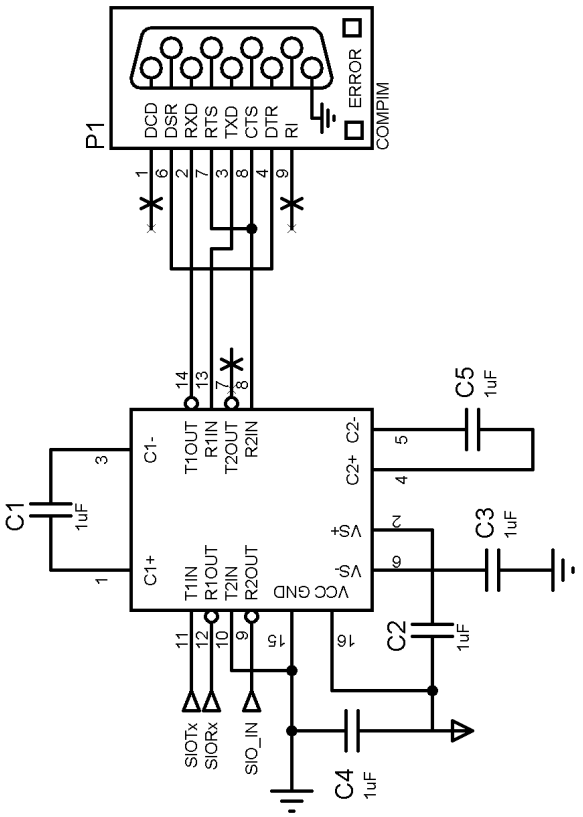
Finalment, aquest tercer procés té una estructura i objectiu molt simple. Mentre no rebi cap interrupció per part del programa principal, comprovarà cada 300 ms la diferència de temps entre peticions que hagi rebut l'ordinador. Si han passat més de 300 ms, donarem per desconnectat el microcontrolador.

Cal recordar, que l'estació base anirà enviant bytes de confirmació de connexió cap a l'ordinador cada 100 ms. D'aquesta manera, si no hi ha dades o peticions en curs, ens assegurem que estigui connectat en tot moment i disponible per executar-les.

```
while (!Thread.currentThread().isInterrupted()) {
    try {

        Thread.sleep(300);

        if(((System.nanoTime() - start_time) / 1e6) > 300){
            controller.updateStatusView(false);
        }else{
            controller.updateStatusView(true);
        }

    } catch (Exception e) {
        Thread.currentThread().interrupt(); // propagate interrupt
    }
}
```

Esquemes elèctrics





Problemes Observats

Un cop finalitzada la fase, veiem que la podem separar en dos blocs principals amb els seus problemes corresponents. Per una banda la part d'implementació de la interfície amb Java i la programació del microcontrolador.

Per tant, pel que fa a la primera menció anterior, veiem que tot i tenir més experiència amb la programació orientada a objectes de Java, la implementació no ha estat fluida.

Això és pel fet que aquesta ens ha suposat complicacions a l'hora de la sincronització entre processos a l'hora de llegir pel port sèrie. És a dir, com ja hem explicat en el disseny del software, hem necessitat un control estricte sobre els processos per part del principal, i que fins a arribar a aquesta decisió quan necessitàvem realitzat peticions senzilles, veiem que la resposta del microcontrolador no la rebíem i la interfície es quedava en espera, en un bucle infinit.

Sembla prou obvi, però quan programes amb un dispositiu com és el microcontrolador sense cap tipus de sistema operatiu que et gestioni les coses automàticament, amb solucions que semblen prou obvies un hi dedica molt de temps per comprovar casuístiques que poguessin donar error, que en un entorn d'ordinador donaries per suposades.

Una altra part d'entrebancs a l'hora de la implementació del projecte a l'ordinador, ha estat en l'aspecte visual. Els dos feia temps que no programàvem en *Swing*, per tant, per voler modificar-la i posar-la al nostre gust, hem hagut de reprendre certs conceptes força bàsics.

Definits els problemes més importants en el desenvolupament del software de l'ordinador, en l'entorn del microcontrolador, han sorgit també un conjunt de complicacions que ens han implicat hores de retard.

La programació en microcontroladors, no és complicada, el que passa és que quan un s'enfronta a un projecte com aquest on el nombre de perifèrics a gestionar no és baix, hom ha de tenir les coses clares a l'hora d'implementar-ho.

Les inicialitzacions dels perifèrics menys utilitzats fins ara com és l'EUSART i l'ADC han estat les que més temps hi hem dedicat per qüestió de desconeixença.

Un cop les configuracions ja les teníem, la part que més hem dedicat temps ha estat la de gestió general del micro. Sembla prou obi que sigui aquesta, però és que la manera que hem implementat el sistema ha estat de tal manera que hi hagi la màxima cooperativista possible entre els diferents perifèrics. Per tant, en el mateix moment que estem escoltant els polsadors hem de generar el heartbeat i a la vegada enviar peticions de connexió pel DB9 i esperar qualsevol petició per part de l'ordinador.

El cúmul de funcionalitats a implementar ha suposat que per poder arribar a implementar-les, hem necessitat més temps del pensat.

Cal destacar però, que un dels perifèrics que ens ha donat més problemes, tot i la seva simplicitat ha estat l'AD. Donat que, el fet d'afegir-lo a l'últim moment ens ha suposat que apareguin errors que no tindríem si s'hagués portat des de la primera instància.

No només quant a programació, ja que a l'hora de soldar el potenciòmetre, vàrem tardar fins a definir bé de quina manera l'hauríem de soldar per obtenir el voltatge variable correctament.

Conclusions

Finalitzada la primera fase d'aquesta pràctica, podem concloure que ha suposat un repte que ha aportat un munt de coneixement nou, tant en l'àmbit de llenguatges de programació, gestió del codi, en la creació de protocols com finalment en coneixements electrònics.

Però com ja sabem, és només la primera part d'aquest projecte, per aquest motiu donat que encara no ens hem endinsat en la implementació d'aquesta última, hem plantejat la majoria d'aspectes els més modulars possibles per tal de tenir l'habilitat de modificar-los sense cap problema.

Tot i això, a aquestes altures del curs, després de passar per la simple fase 1 de la primera practica i acabar amb la programació d'aquesta fase, hom es dóna compte de què implica cursar una assignatura com Sistemes digitals i microprocessadors.
Desenes d'anys de coneixements de sistemes digitals condensats en poc més de 6 mesos, aporta una visió totalment nova del que estem acostumats en altres assignatures.

Ja que el fet de tenir la possibilitat de poder implementar un software que sigui capaç de gestionar múltiples perifèrics a la vegada, que es comuniqui no només per medis físics sinó mitjançant la radiofreqüència, és a dir, l'ús de la porció menys energètica de l'aspecte electromagnètic (En el nostre cas a 434Mhz) per enviar dades a receptors a desenes de metres a distància, no és quelcom que qualsevol pugui realitzar sense passar per els passos que hem seguit fins a arribar al dia d´avui.