- Y la proxima vez te juro que seré, o patria algo mas practico: te dejaré un borrego, una fotonovela y una flor de plastico
- No habrá proxima vez, dejalo ya Marcial- le respondio la muerte

Javier Krahe

1.-INTRODUCCIÓ 5

1.1.- ORIGEN DEL PROJECTE 5
1.2.- LA SUBHASTA DE PEIX TRADICIONAL 6
1.3.- LA SUBHASTA DE PEIX AUTOMATITZADA 7
1.4.- DEFINICIÓ DEL PROJECTE 8
1.4.1.- OBJECTIU DEL PROJECTE 8
1.4.2.- NECESSITATS 8
1.4.3.- ESPECIFICACIONS DISSENY 9

2.- COMUNICACIÓ IR 10

2.1.- PRINCIPIS FÍSICS 10
2.1.1.- LA LLUM 10
2.1.2.- LA LLUM IR 11
2.1.3.- EFECTE FOTOELÈCTRIC 11
2.1.4.- PERQUÈ IR 12
2.2.- CODIFICACIÓ IR 13
2.2.1.- PRINCIPIS 13
2.2.2.- CODIFICACIONS TÍPIQUES DE BIT 14
2.2.3.- CODIFICACIÓ DE DADES 15
2.3.- MATERIAL IR 17
2.3.1.- CONSIDERACIONS GENERALS 17
2.3.2.- VELOCITAT-DISTÀNCIA-ANGLE 17
2.3.3.- EMISSORS 18
2.3.4.- DETECTORS 19

3. DISSENY 21

3.1.- INTRODUCCIÓ 21
3.1.1- CONSIDERACIONS SOBRE ELS COSTOS 21
3.1.2.- PRIORITATS DEL DISSENY 21
3.1.3.- REDUCCIÓ DE LA COMPLEXITAT 23
3.2.- HARDWARE UTILITZAT 24
3.2.1.- RECEPTOR IR: VISHAY TSOP 7000 24
3.2.2.- LED IR: VISHAY TSFF5400 26
3.2.3.- MICROCONTROLADOR: MICROCHIP 16F627 27
3.2.4.- ALTRES CONSIDERACIONS SOBRE EL HARDWARE 28
3.3.- CODIFICACIÓ 29
3.3.1.- TAMANY DEL CODI I SIGNIFICAT 29
3.3.3.- PORTADORA 30
3.3.3.- TRAMA IR 30

4.- EMISSOR 32

4.1.- DESCRIPCIÓ 32
4.1.1.- ESPECIFICACIONS 32
4.1.2.- CONTINGUT EEPROM 32
4.1.3.- PROTOCOL PROGRAMACIÓ EEPROM 33

4.1.4.- ESQUEMA ELÈCTRIC I PCB 35
4.2.- PARTS 37
4.2.1.- TECLAT 37
4.2.2.- ALIMENTACIÓ 38
4.2.3.- POTÈNCIA 39
4.2.4.- CONNECTOR PORT SÈRIE 40
4.3.- CODI 41
4.3.1.- ESTRUCTURA DEL PROGRAMA 41
4.3.2.- RUTINES 42
4.3.3.- RUTINA _ENVIACODI 43
4.3.4.- LLISTAT EMISSOR.ASM 46

5.- RECEPTOR 52

5.1.- DESCRIPCIÓ 52
5.1.1.- ESPECIFICACIONS 52
5.1.2.- PROTOCOL COMUNICACIÓ 52
5.2.- PARTS 60
5.2.1.- ALIMENTACIÓ 60
5.2.2.- PORT SÈRIE 60
5.2.3.- RECEPTOR IR 62
5.2.4.- POLSADORS I JUMPERS 62
5.2.5.- LEDS 63
5.3.- CODI 64
5.3.1.- ESTRUCTURA DEL PROGRAMA 64
5.3.2.- RUTINES 66
5.3.3.- DESCODIFICACIÓ DEL CODI IR 67
5.3.4.- LLISTAT RECEPTOR.ASM 72

6.- PROGRAMES PC 82

6.1.- RECEPTOR 82
6.1.1.- PANTALLA PRINCIPAL 82
6.2.2.- FUNCIONAMENT 83
6.1.3.- CODI 85
6.2.- PROGMASTER 92
6.2.1.- PANTALLA PRINCIPAL 92
6.2.2.- FUNCIONAMENT 92
6.3.3.- CODI 94

7.- TESTS 100

7.1.- CAPTURES OSCIL·LOSCOPI 100
7.1.1.- ENVIAMENT DE TRES CODIS 100
7.1.2.- DETALL FRAGMENT DE CODI 101
7.1.3.- DETALL PORTADORA 102
7.1.4.- SORTIDA DETECTOR IR 103
7.1.5.- RETARD EN L'EXECUCIÓ DE LA ISR 104
7.2.- DETERMINACIÓ CONSTANTS DEL RECEPTOR 105
7.2.1.- PROGRAMA TEST.ASM 105
7.2.2.- PROGRAMA TEST_IR.BAS 108

8.- CONCLUSIONS I EVOLUCIÓ 111

8.1.- CONCLUSIONS 111
8.2.- MILLORES PROPOSADES 112

BIBLIOGRAFIA 113

1.-INTRODUCCIÓ

1.1.- Origen del projecte

L'any 1998 vaig entrar a treballar a l'empresa RegisCompte, la qual estava iniciant un complex projecte per automatitzar completament el sistema de subhasta d'una llotja de peix i entrar en aquest sector.

La meva part va ser la creació d'un sistema d'identificació de compradors mitjançant comandaments a distància per IR. En aquell moment, vaig decidir realitzar el sistema basant-me en material el màxim d'estàndard possible, assegurant la facilitat en el disseny i la seva fiabilitat. Seguint aquest principi es van crear comandaments amb un circuit integrat genèric per aquest tipus d'aplicacions de la casa Holtek mentre que el receptor es va basar en una placa microcontrolada x86 de la casa Altair.

L'any següent es va realitzar la instal·lació a la primera llotja, i durant els anys següents es van fer instal·lacions amb el mateix sistema a diverses llotges del litoral Català i Valencià (Llança, L'Escala, Blanes, Mataró, L'Ampolla, Benicarló, Cullera). Les úniques modificacions van consistir en crear comandaments de fins a 16 botons, que no havien estat previstos en el disseny original.

L'any 2002 es decidí millorar el sistema, creant nous comandaments i receptors específicament dissenyats per aquest tipus d'aplicacions. Un dels objectius era permetre la possibilitat d'anar amb un sol comandament a diferents llotges, de manera que s'havia de garantir la no existència de codis repetits i també, que un majorista amb un sol comandament pogués comprar en nom de mes d'un comprador. Tot plegat passava per permetre més codis (1024 codis possibles en el primer disseny) i eliminar la limitació d'utilitzar codis consecutius en els comandaments múltiples. S'aprofitaria, si fos possible, per augmentar la velocitat i abast de l'emissió i simplificar el nombre de components de l'electrònica.

El projecte era completament nou, per realitzar-lo es van utilitzar microcontroladors tant en els comandaments com en el receptor i material d'última generació en els elements optoelectrònics. El mètode de codificació es va crear 'a mida' i es van aconseguir tots els objectius marcats a més d'altres millores com una reducció del consum i millors característiques en la comunicació receptor - PC. Progressivament aquest sistema ha anat substituint l'anterior a la vegada que s'ha anat instal·lant a noves llotges (València, Sagunt, Deltebre, Les cases d'Alcanar...).

El projecte presentat aquí està basat en aquest últim sistema de comandaments. Les principals diferències consisteixen en un nou disseny dels circuits impresos, que no havia realitzat jo en el sistema original i l'utilització d'un altre tipus de trama IR més sofisticat que l'original i més interessant tècnicament.

1.2.- La subhasta de peix tradicional

El primer pas en el procés de comercialització del peix ha estat en la llotja. És allà on els propis pescadors subhasten les seves captures a majoristes i detallistes de manera que, de manera natural, s'ajusta el preu segons l'oferta i la demanda. Per ambdúes parts (pescadors i peixaters), la subhasta resulta de vital importància ja que s'hi juguen els seus guanys.

Quan una barca arriba a la llotja, ja porta la seva captura en caixes i separada per espècies i qualitats, formant lots. Un lot està composat d'una o més caixes que contenen peix de la mateixa espècie i qualitat similar. A mesura que va arribant la mercaderia a la llotja, aquests lots són pesats i subhastats pel personal de la confraria.

Tradicionalment, després de pesar el gènere, un encarregat de la confraria era qui marcava un preu de sortida a un lot i, a viva veu, anava cantant l'evolució del preu a la baixa. La baixada del preu es parava quan un comprador acceptava aquell lot a l'últim preu cantat. Si coincidien més d'un comprador al mateix moment, es reiniciava la cantada des d'un preu lleugerament superior a l'últim donat fins que apareixia un únic comprador. Aquest, podia decidir quedar-se només amb una part del lot, amb el que calia repetir el procés amb la resta del gènere que el composava. Posteriorment s'havia d'apuntar per a cada lot la seva procedència i característiques (espècie, qualitat i pes), el comprador i el preu al qual s'havia venut.

1.3.- La subhasta de peix automatitzada

Queda clar que aquest es tractava d'un procés totalment manual, lent i feixuc. A més, era molt fàcil que s'hi produïssin errors, donat s'havien d'apuntar un gran nombre de dades a mà en condicions un tant difícils. El fet de dependre de l'oïda i la vista de l'encarregat de la subhasta també podia donar lloc a disputes quan algun comprador afirmava haver acceptat un preu i no ser sentit.

Tot plegat ha portat a una automatització progressiva de les subhastes conservant però, el mètode de subhasta a la baixa i el sistema de lots. Actualment es disposa d'una cinta mecànica (en realitat vàries cintes successives) on, a un extrem, es dipositen les caixes de peix i a l'altre surten etiquetades amb tota la informació correcte. Pel camí s'han pesat, identificat, subhastat i totes les dades ja es troben dins del sistema informàtic de la llotja.

En una subhasta automatitzada tipus, podem diferenciar tres parts funcionals en la cinta:

- **Pesada i introducció de les dades per part d'un operador:** un operador introdueix en un terminal el tipus de peix i el preu de sortida. El pes el llegeix el sistema directament d'una bàscula instal·lada en la pròpia cinta. La caixa pesada és acumulada en un sector de la cinta.

- **Subhasta:** En un monitor o pannell apareix informació sobre el lot que està en el punt de subhasta amb el preu de sortida. Després de sonar una senyal s'inicia la subhasta pròpiament dita i aquest preu evoluciona a la baixa fins que un comprador prem un polsador per indicar que accepta el preu. El sistema identifica el comprador, i aquest introdueix d'alguna manera el nombre de caixes que li interessen d'aquell lot.

- **Etiquetatge:** S'etiqueta la caixa amb informació sobre el tipus de peix, pes de la caixa i comprador.

1.4.- Definició del projecte

1.4.1.- Objectiu del projecte

Disseny i desenvolupament d'un sistema per la identificació de compradors en processos de subhasta, integrable dins d'un sistema de subhasta automatitzada.

1.4.2.- Necessitats

En una subhasta és important permetre la mobilitat dels compradors. Tant per permetre'ls acostar-se al gènere per veure'l, com per la simple comoditat de no haver d'estar-se hores en el mateix lloc. Això obliga a crear un sistema no cablejat.

Cada comprador ha de disposar d'un comandament amb un codi assignat. Quan es vulgui realitzar una compra, es prem un polsador del comandament, i aquest envia el codi a un receptor. Aquest rep el codi i l'envia al sistema de control de subhasta. Serà tasca del control de la subhasta determinar a qui correspon aquell codi per recuperar les dades que hi estiguin vinculades.

Cal crear comandaments amb diferent nombre de polsadors amb els seus respectius codis. En determinats casos, una sola persona compra per diferents clients. Independentment de qui premi el polsador, cal registrar el comprador efectiu.

Aquest sistema remot es basa en tecnologia d'infraroig (IR) per la seva direccionalitat respecte mètodes via ràdio. Aparentment aquest és un desavantatge respecte mètodes via ràdio, a on no cal apuntar a un receptor per realitzar una comunicació. Però cal recordar que, sobretot el petit comprador, sovint ha de realitzar tasques físiques com carregar alguna caixa. Resulta molt fàcil que amb el comandament a la butxaca, s'accioni accidentalment i es realitzi una compra no desitjada. Per això, en aquest cas, la direccionalitat dels comandaments juga a favor de l'usuari. Per altra banda, sistemes de ràdio amb similars característiques de velocitat, resulten menys econòmics.

La comunicació amb el sistema de control es realitzarà mitjançant port sèrie RS-232. Es tracta d'un sistema totalment estès que permetrà l'interconnexió amb pràcticament qualsevol tipus de hardware.

La velocitat també és un paràmetre important. Cal reduir al màxim el temps de latència des que es prem el polsador fins que el codi arriba al control de la subhasta. Tractant-se d'una subhasta on guanya el més ràpid, retards apreciables generen molta desconfiança. Per això cal plantejar-nos un temps màxim de 100mS entre l'enviament del codi mitjançant IR, la seva descodificació per part del receptor i l'enviament via port sèrie. Garantitzarem que el retard no sigui apreciable.

Una última característica tècnica és l'abast dels comandaments. Per permetre una correcta llibertat de moviments dels compradors, considerem com a mínim un abast de 15 metres. És difícil que algú es posi a distàncies majors del receptor si aquest s'instal·la juntament amb la pantalla d'informació: a més distància no es pot llegir la informació correctament. De totes maneres es poden instal·lar més receptors i deixar que el sistema de control de la subhasta elimini els codis que l'hi arribin duplicats.

En aquest projecte no es tractarà l'ergonomia dels comandaments. Només apuntar en aquest sentit que cal un disseny de carcassa i polsadors resistents a cops i esquitxades donat que estan en un entorn de treball difícil. El tamany lògic serà semblant al d'un paquet de tabac, tamanys majors resultaran poc pràctics mentre que tamanys menors poden resultar difícils d'utilitzar.

1.4.3.- Especificacions disseny

A partir de l'apartat anterior, les característiques mínimes que ha de complir el sistema son:

Generals

- Distància màxima: >15 mtrs
- Latència total: <100mS
- Codis possibles: >10000

Emissor

- Possibilitat de crear-lo amb varis polsadors
- Codis fàcilment modificables
- Mínim consum per allargar la vida de la pila

Receptor

- Comunicació sèrie de velocitat configurable (9600 – 115200bps)
- Incorporar ordres al receptor per activar i desactivar la recepció IR
- Estabilitzador de la tensió d'alimentació (entorn industrial)

2.- COMUNICACIÓ IR

En aquest capítol es farà un repàs a les principals característiques de la llum infraroja. També es descriuran els principals mètodes utilitzats per enviar informació a través d'IR i l'estat actual d'aquest tipus de tecnologia.

Amb aquesta informació i els requisits plantejats en l'anterior apartat, en el pròxim capítol s'acabarà definint el tipus de codificació que s'utilitzarà, els transductors i la resta de hardware utilitzat.

2.1.- PRINCIPIS FÍSICS

2.1.1.- La llum

L'estudi de la llum ha ocupat bona part de la física al llarg de la història. La mesura de la seva velocitat, per exemple, va ocupar Galileu, va enfrontar astrònoms com Rømer i Cassini (quan el primer la va aproximar amb gran exactitud) i va provocar nombrosos experiments a finals del XIX. I és que les propietats de la llum no s'assemblaven a les de cap altre element i fins el segle XIX encara es discutia si es composava de partícules (teoria corpuscular) o es tractava d'ones que es propagaven a través d'un mitjà encara desconegut però anomenat ja com a èter (teoria ondulatòria).

Un avenç importantíssim el va fer Maxwell el 1837 quan relacionant els camps elèctrics i magnètics va notar que es desplaçaven a la velocitat de la llum i va acabar teoritzant que la llum era un tipus més de radiació electromagnètica, teoria que es va demostrar a finals del segle XIX.

Actualment, la classificació genèrica que es fa de les radiacions electromagnètiques és la següent:

Longitud d'ona (en metres)	**Tipus**
> 0.3	Radiofreqüència
$0.3 - 10^{-3}$	Microones
$10^{-3} - 7.8 \cdot 10^{-7}$	Infraroig
$7.8 \cdot 10^{-7} - 3.8 \cdot 10^{-7}$	Llum visible
$3.8 \cdot 10^{-7} - 10^{-9}$	Ultravioleta
$10^{-9} - 10^{-12}$	Raigs X
$<10^{-12}$	Raigs gamma

Veiem en aquesta taula que l'interval més ben definit és el corresponent a la llum visible (380nm a 780nm) que, com bé indica el nom, són aquelles radiacions detectables per l'ull humà. La resta d'intervals no indiquen cap 'frontera' física que suposi una discontinuïtat en les propietats d'aquestes emissions. Dins l'interval de la llum visible, tenim una correspondència unívoca entre la longitud d'ona i el color i és que, en realitat, el color és una

representació cerebral de la longitud d'ona. En un ull sa, aquesta correspondència és la següent:

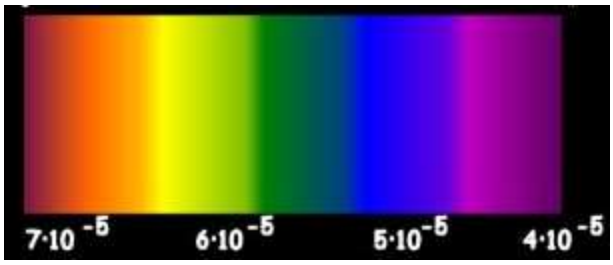


Fig 2.1: Espectre llum visible

A on la longitud d'ona més curta (380nm) correspon al violeta, mentre que la més llarga (780nm) correspon al vermell. És per això que les radiacions amb longituds d'ona inferiors (freqüències superiors) al violeta s'anomenen ultravioleta mentre que les de longitud d'ona superior (freqüències inferiors) s'anomenen infraroig (IR).

2.1.2.- La llum IR

Abans de la teoria electromagnètica de la llum, William Herschel va descobrir la llum infraroja l'any 1800 mentre estudiava la descomposició de la llum blanca a través d'un prisma. Mesurant amb un termòmetre la temperatura de cada un dels colors que apareixien en la descomposició va veure com la temperatura més alta es trobava més enllà del vermell, indici de l'existència d'una llum invisible a l'ull humà.

En realitat, de les radiacions infraroges (IR) s'en fa la següent classificació:

Longitud d'ona	**Tipus**
780nm – 3μm	IR proper
3μm – 6μm	IR mig
6μm – 15μm	IR llunyà
15μm – 1mm	IR extrem

De fet, sota la denominació IR s'engloben radiacions amb orígens molt diferents. Per exemple, l'infraroig utilitzat en dispositius electrònics, com és el cas d'aquest projecte, sol estar entre 800nm i 1000nm (IR proper), mentre que l'IR detectat per cameres termogràfiques està dins de l'IR mig i llunyà.

2.1.3.- Efecte fotoelèctric

A finals del segle XIX ja s'havia observat que l'incidència de llum sobre dos electrodes afectava l'intensitat del salt elèctric que es produïa entre ells. El 1905, Albert Einstein va donar una explicació matemàtica on s'incloïa l'existència de partícules discretes en la llum (fotons) i que determinats materials i interactuaven lliurant electrons. Un cop desenvolupada, aquesta teoria va ser l'inici de la mecànica quàntica i va donar el premi nobel a Einstein l'any 1921.

Entre altres particularitats de l'efecte fotoelèctric, està que és major quan més alta és la freqüència de la llum (i més curta la seva longitud d'ona), i l'existència d'una freqüència umbral per a cada material, sota de la qual aquests deixen d'emitir electrons.

Queda clar que en aquesta teoria es basa qualsevol element electrònic detector de llum.

2.1.4.- Perquè IR

La fabricació de detectors de llum a gran escala té per objectiu buscar transductors amb la màxima relació rendiment/cost. Això s'ha aconseguit amb detectors basats en silici, que tenen la següent resposta:

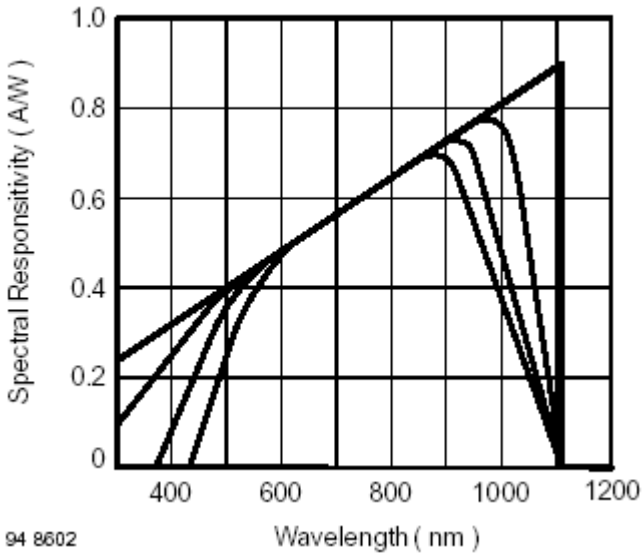


Fig 2.2: Resposta detectors Si

La línia recta que veiem és el cas ideal, mentre que la resta de corbes depèn del disseny del detector. En tot cas tenim que els millors rendiments s'obtenen per a longituds d'ona entre 800 i 1000nm, corresponent a llum IR.

Per altra banda l'utilització de llum no visible ens aporta dues avantatges: no resulta molesta a l'ull humà i està poc present en entorns amb il·luminació artificial.

2.2.- CODIFICACIÓ IR

2.2.1.- Principis

La manera intuïtiva d'enviar un bit d'informació mitjançant llum és: presencia de llum equival a u, absència de llum equival a zero. Per enviar informació només cal enviar en sèrie un bit rera l'altre de manera que tindríem seqüències de llum/no llum equivalents al 1- 0 binari. Desgraciadament aquest sistema presenta una debilitat: no ha previst l'altíssima fressa ambiental que han de suportar les comunicacions IR.

En el desenvolupament d'un sistema de comunicació IR, pràcticament tot l'esforç es dirigeix a eliminar o, com a mal menor, detectar els errors produïts per llum no controlada. Com ja s'ha dit, s'utilitza llum IR perquè és menys present que la llum visible, especialment en il·luminació artificial, i així reduir la fressa, però la pràctica totalitat de fonts de llum contenen en el seu espectre traces d'IR.

Bàsicament tenim dos mecanismes per minimitzar aquests errors:

Utilització de portadora

Podem emetre llum amb una freqüència portadora. Generalment s'emet la llum amb pulsos d'una freqüència d'entre 30 i 50 kHz. Llavors el sistema receptor pot incloure un filtre de manera que, per exemple, una incidència de llum IR constant pot ser discriminada respecte la llum que prové del nostre emissor.

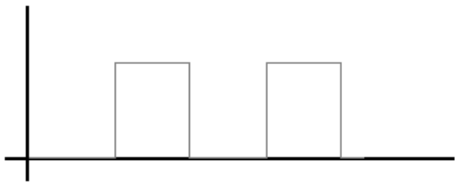


Fig 2.3: Pulsos sense portadora

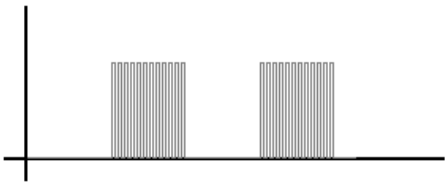


Fig 2.4:Pulsos utilitzant portadora

Codificacions específiques

Generalment s'utilitzen mètodes de codificació on la representació d'uns i zeros no és fa per la presència o absència de llum sinó que es codifiquen segons la llargada dels pulsos o del temps entre pulsos de llum d'una durada concreta. Amb aquests mètodes, la presència de fressa pot ser detectada i permet, almenys, descartar la senyal rebuda si està contaminada.

2.2.2.- Codificacions típiques de bit

Els mètodes mes estesos per codificar un bit en aplicacions de comandaments són els següents:

Longitud de pols: Es diferencia el valor del bit per la llargada del pulsos. L'espai entre pulsos es manté constant.

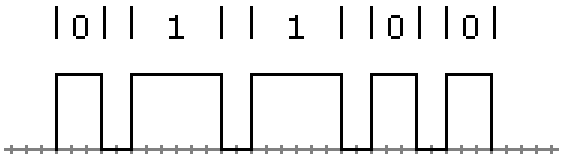


Fig 2.5: Codificació amb longitud de pols

Distància entre pulsos: Es diferencia el valor del bit per la distància entre pulsos. La duració d'aquests és constant.

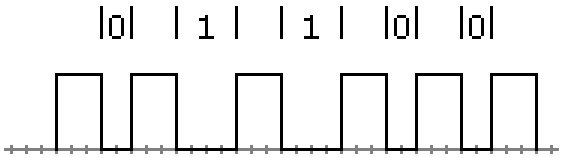


Fig 2.6: Codificació amb distància entre pulsos

Bi-fase: Cada cert temps (sempre el mateix) la senyal conté un flanc el sentit del qual (ascendent o descendent) ens dóna el valor del bit. La resta de flancs s'ignoren.

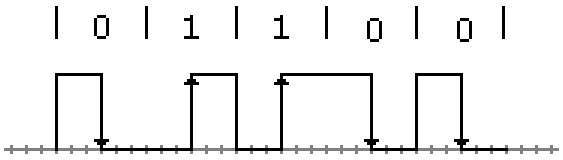


Fig 2.7: Codificació bi-fase

Normalment, en tots aquests mètodes s'aplica portadora tot i que no s'ha detallat en els gràfics. La inclusió de portadora ens obliga a un temps mínim de pols necessari per poder fer el filtratge posterior del senyal. Lògicament aquest temps és independent del mètode de codificació utilitzat i ens ve determinat pel sistema receptor.

S'aprecia fàcilment com una modificació puntual del senyal pot ser detectada. En tots els casos s'han de complir regles estrictes respecte les llargades dels pols. Si el senyal es contamina i és rebut amb un pols més llarg que l' original enviat, es detectarà directament si utilitzem codificació amb

llargada de pols, codificant en distància de pols és detectarà per reduir-se l'espai entre el pols afectat i el següent i utilitzant bi-fase és detectarà al alterar-se les posicions dels flancs.

Entre els dos primers mètodes podem observar com, codificant amb distància entre pulsos, tots ells poden tenir la longitud mínima. La conseqüència és un menor consum d'energia, fet important en aplicacions que utilitzin bateries.

La principal característica de la codificació bi-fase és que la seva longitud no varia segons la informació enviada, tant la codificació d'uns com la dels zeros comporta el mateix temps. També es pot destacar que és el mètode més ràpid ja que en tots tres necessitem un pols i un espai per codificar un bit, però en aquest cas tots ells poden ser de la mida mínima necessària pel sistema receptor.

2.2.3.- Codificació de dades

Un cop repassats els mètodes bàsics per codificar un bit, és el moment de veure com enviar informació efectiva. Com ja s'ha dit, el principal problema que ens trobem en aquest tipus de comunicacions és l'alta probabilitat d'errors de transmissió, per tant, també en aquest nivell es prioritzarà la detecció d'errors.

La utilització de codis Gray que permeten detectar i corregir errors produïts en la transmissió de dades no són adequats en aquest cas. Els codis Gray només son vàlids quan es produeixen pocs errors i aïllats. En les comunicacions IR passa just el contrari: es produeixen molts errors i sovint estan agrupats.

En l'àmbit dels comandaments a distància, podem seguir la següent lògica: tenim la seguretat que seran descartades un gran nombre de dades per errors de transmissió, i sabem que el volum de dades enviades és baix. Llavors, podem adoptar com a solució re-enviar dos o tres cops cada dada que vulguem emetre. Si en algun cas això no fos suficient, serà el propi usuari qui repeteixi l'ordre. Es tracta d'un mètode poc elegant (acceptem que el sistema falla) però assumible. En canvi, l'enviament de volums importants de dades obliga a utilitzar sistemes bidireccionals pel control d'errors i, en general, reduir la distància entre emissor i receptor a nivell de centímetres per minimitzar l'efecte de la llum ambiental. Aquest seria el cas de l'especificació IrDA de comunicació de dades per IR, bastant generalitzada en ordinadors, telèfons mòbils i PDAs. Tot i així s'han realitzat instal·lacions de transmissió de dades a llargues distàncies mitjançant IR.

Veurem ara un codi real i creat per la indústria com és el codi NEC. Aquest està pensat per a comandaments IR per electrodomèstics i només és un dels molts existents. Encara que tots els electrodomèstics només necessiten rebre una parella de bytes per fer les funcions, cada empresa sol crear el seu codi propi que utilitza en tot el seu catàleg. Podem considerar aquesta varietat

de solucions pel mateix problema com prova de la complexitat que suposa evitar errors.

Codi NEC

Aquest codi envia dos bytes d'informació efectiva (adreça i dada). Utilitza portadora a 38KHz i codificació amb longitud entre pulsos.

El procediment és el següent: primer s'envia el byte d'adreça seguit del mateix byte negat, després és repeteix el procediment amb el byte de dada que s'envia seguit del seu negat. Si es pretén enviar l'adreça 10011011 amb la dada 01010110 s'enviarà la següent cadena de bits:

10011011 01100100 01010110 10101001

L'objectiu d'enviar bytes negats i no simplement duplicar la informació és doble: per una banda s'aconsegueix que tot i utilitzant codificació amb llargada de pols, totes les trames tinguin la mateixa longitud al tenir sempre la mateixa quantitat d'uns i de zeros. Per altra banda, i considerant que els errors vénen donats per l'adició de senyal (la fressa ens aportarà més senyal, no ens la treurà) ens assegurem la impossibilitat de que dos pulsos separats pel temps d'enviament d'un byte ens modifiquin 'correctament' el senyal.

Si observem el senyal (fig 2.8) veurem com, previ a l'enviament de les dades, s'envia una senyal a 38KHz durant 9mS (leader) seguit d'un espai de 4.5mS. Aquesta té com objectiu que el detector es pugui auto-ajustar al nivell del senyal. Amb els receptors actuals però, ja no és necessari, però ja ens dóna una idea de la complexitat del procés de recepció.

Fig 2.8: Codi NEC

Cada pols de llum (excepte el leader) consta en realitat de 22 cicles de 8.77uS de llum i 17.53uS sense llum, es a dir, 22 cicles a 38KHz amb un proporció de 1/3 on i 2/3 off. No s'ha representat en el gràfic per millorar la legibil·litat.

2.3.- MATERIAL IR

2.3.1.- Consideracions generals

La tecnologia IR per comandaments a distància té el seu origen en el control d'electrodomèstics. Tècnicament, el seu objectiu és transmetre una petita quantitat de dades (poques desenes de bits) a curta distància sense importar unes grans prestacions. Com les necessitats no han variat amb els anys, tampoc ho han fet els mètodes utilitzats, així que aquesta tecnologia s'ha orientat més en la reducció de costos i facilitat d'ensamblatge que en les prestacions.

Així, hem vist com el codi NEC explicat en l'apartat anterior, i que encara s'utilitza actualment, necessita 67.5 mS per enviar 16 bits d'informació. Altres com el Sony 12 de bits o el RC5 estan en temps similars i en casos com el Sony 12 de bits ni tan sols s'envien bits de detecció d'error: el receptor espera vàries trames iguals.

Pel que respecta a la distància de treball, pocs són els fabricants que donen distàncies, ni que sigui aproximades, de parells emissor/detector. El fabricant dona per suposat que compliran els modestos requeriments...

2.3.2.- Velocitat-distància-angle

La cerca d'elements que treballin a velocitats i distàncies elevades i amb un bon angle de treball (que no hagin d'estar perfectament orientats emissor i receptor) ha estat complicada. Sobretot perquè són conceptes antagònics en el terreny dels detectors.

Per aconseguir una major sensibilitat, distància al capdavall, una solució lògica és augmentar el tamany del detector. Desgraciadament, per la seva construcció els detectors tenen una certa similitud amb un condensador, que fa les funcions de filtre de la seva pròpia senyal de sortida i impossibilita freqüències altes si el seu tamany augmenta.

Una segona opció és utilitzar una lent que concentri la llum que rep a un detector relativament petit. En aquest cas, el problema ens apareixerà en forma d'una alta direccionalitat.

La decisió final correspon al dissenyador. Tots els catàlegs ofereixen diversos components que inclouen en les fulles de característiques gràfics amb la sensibilitat en els detectors i la potència dels leds emissors segons l'angle respecte la normal i de la freqüència de la portadora.

2.3.3.- Emissors

Els emissors IR utilitzats per transmissió de dades són leds. Entre les seves avantatges estan el seu funcionament a altes freqüències, el tamany, i el seu rendiment.

Les característiques més importants per seleccionar-los són:

- **Intensitat lumínica:** És la potència irradiada per unitat de superfície. Leds molt directius tenen valors elevats per aquesta característica.
- **Angle d'emissió:** Ens diu dins de quin angle l'intensitat lumínica és, almenys, la meitat del màxim.
- **Longitud d'ona:** És evident que ha de ser el màxim de semblant a la longitud d'ona de màxima sensibilitat del receptor. Els valors típics son 880nm i 940nm.
- **Freqüència de treball:** Freqüència màxima del senyal que poden emetre.

Longitud d'ona i angle d'emissió són dos valors contraposats. Donada una potència tindrem més intensitat quan més petit sigui l'angle d'emissió. Aquest concepte queda clar comparant la distribució d'un led orientat a comunicacions IrDA i un orientat per a comandaments:

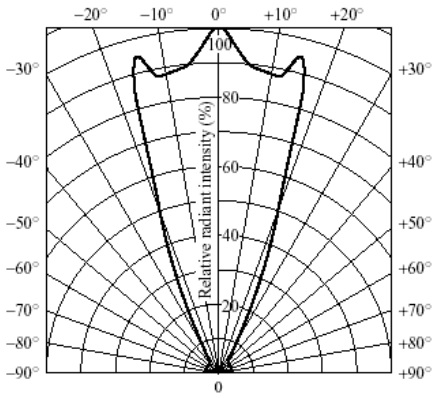


Fig 2.9: Distr. Led GL1F20

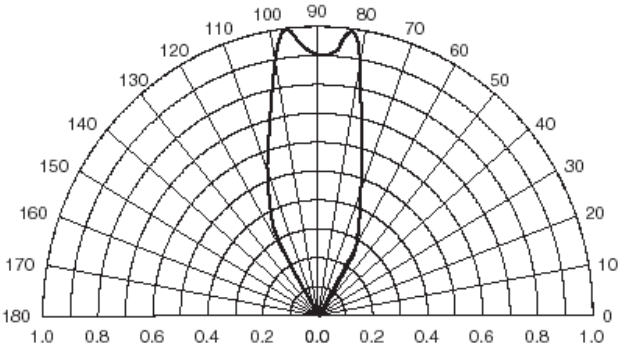


Fig 2.10: Distr. Led QED33

En la figura 2.9, corresponent a un led GL1F20 de sharp (actualment descatalogat) l'angle és mes reduït però constant a mesura que ens allunyem, mentre que en la figura 2.10, del led QED233 de Fairchild, aquest angle és superior però es va reduint a mesura que augmenta la distància.

A part d'aquesta característica, cada led està orientat específicament a una aplicació concreta que no sempre ens explicaran els seus valors absoluts. Per exemple, una característica del led led GL1F20 es que incorpora un transistor en el propi encapsulat, de manera que és fàcilment integrable en qualsevol sistema microcontrolat.

Com amb qualsevol altre component electrònic, es disposa de diferents tipus d'encapsulat. Així tenim el format típic de led de 3 i 5mm, d'altres per muntatge en superfície (smd) i altres encapsulats molt específics. De totes maneres, els més potents solen estar amb encapsulat clàssic de 5mm.

2.3.4.- Detectors

Els components bàsics per detectar llum són els fotodiodes i els fototransistors. Els dos tipus són variacions de diodes i transistors normals, disposant d'una unió pn exposada a la llum a detectar i d'una construcció i dopatje específics.

El disseny d'un circuit detector amb aquests components és realment complexe. Les variacions d'intensitat que hi provoca l'incidència de llum no passen de pocs nanoampers i a més, són valors molt dependents de la temperatura i, no cal dir-ho, de la distància de l'emissor. En el nostre cas, la senyal amplificada s'ha de filtrar per detectar la portadora i reconstruir la senyal original que, finalment, podrem descodificar.

Afortunadament, els fabricants posen a disposició receptors IR que integren tot el conjunt: detector, amplificador i filtre passa-banda. Fotodiodes i fototransistors queden majoritàriament reservats per a fer mesures d'intensitat de llum ambiental per a cameres o tasques simples com encoders.

Receptors IR

L'estructura d'un receptor IR és el següent:

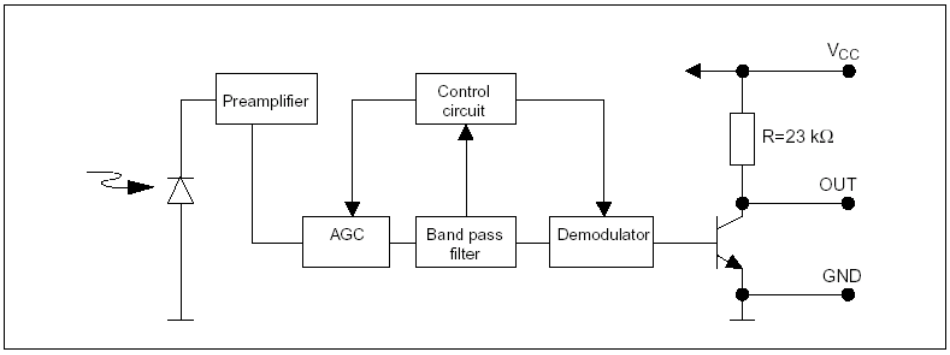


Fig 2.11: Estructura receptor IR

La senyal proporcionada per un fotodiode es preamplifica, passa per un amplificador amb control de guany automàtic, posteriorment per un filtre passa-banda i finalment és demodulada. Encara que fins fa pocs anys, es solia integrar aquest conjunt dins d'un cub metàlic d'un centímetre de costat, actualment s'han imposat encapsulats com el següent, que pertany al SFH-5110 de Osram:

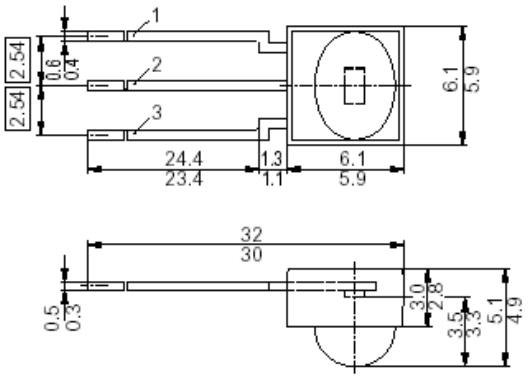


Fig 2.12: Dimensions del receptor SFH- 5110

Veiem que el tamany és lleugerament major a un transistor de baixa potència. El semicercle existent en la part frontal fa les funcions de lent, que per llum IR, són bastant opacs a la llum visible.

Les característiques més importants de qualsevol receptor són:

- **Longitud d'ona detectada:** Ha de ser la mateixa que la de l'emissor.

- **Freqüència passa-banda:** Lògicament, cal que estigui centrada a la freqüència de la portadora del senyal enviat i que sigui el màxim de tancada possible.

- **Numero mínim i màxim cicles:** El detector necessita rebre un mínim de cicles per detectar el senyal. En molts casos també s'indica un tamany màxim.

- **Cicle treball:** No cal que els cicles siguin de 50%. Cicles del 20%, per exemple, poden ser igualment vàlids i ens permeten estalviar energia en l'emissor.

- **Distorsió del senyal:** El senyal de sortida no es correspon exactament a l'envolvent de la portadora. Caldrà tenir en compte aquestes variacions que introdueix el procés de filtratge.

En aquest projecte es crearà una codificació expressa pel detector escollit, pel que aquests valors seran estudiats en detall. Per codificacions stàndard, són els fabricants els que disposen de taules indicant el detector més adequat per a cada sistema de codificació.

3. DISSENY

3.1.- INTRODUCCIÓ

Amb tota la documentació que ja tenim es pot començar el procés de disseny del sistema. Aquest comença amb una sèrie de consideracions generals sobre la seva estructura i en l'elecció de la codificació que s'utilitzarà, que ha de complir els requisits de velocitat, tamany de la dada a enviar i seguretat en la transmissió. A partir d'aquí caldrà seleccionar els transductors i microcontroladors que utilitzarem i, ja en el capítol posterior, dissenyar els circuits amb els components auxiliars que necessitem. Amb el hardware definit i coneixent amb detall el potencial de que disposem, restarà fer la programació de cada una de les parts (emissors i receptors).

3.1.1- Consideracions sobre els costos

Al ser un projecte creat per una necessitat comercial, és evident que caldrà contenir els costos. Minimitzar el cost d'un projecte inclou tant el cost de producció com el de disseny i no podem oblidar que aquests solen ser inversament proporcionals: un baix cost de producció s'aconsegueix amb dissenys altament optimitzats, a on optimitzat és sinònim de temps que, a la seva vegada (sobretot en el mon empresarial) és sinònim de diners.

En el disseny de productes destinats a ser produïts massivament, s'inverteixen molts esforços en la reducció de costos de producció, a la llarga la inversió es recupera. De totes maneres no ens trobem en aquest cas. Per començar, aquí podem considerar una previsió de 5000 comandaments en la vida comercial del projecte, però a més és tracta d'un producte a on el preu final no és determinant. Una reducció de cèntims en el cost de components no justificarà grans esforços.

Per tot això, més que una reducció de costos, el que es busca en aquest projecte és una contenció. I per aconseguir-la es farà via homogeneització dels components.

3.1.2.- Prioritats del disseny

De bon principi hem d'adoptar tres factors com a prioritaris:

- Distància màxima de la comunicació
- Minimitzar el consum de l'emissor
- Control absolut dels temps

Distància màxima de la comunicació

La distància màxima entre el receptor i emissor en línia recta no pot ser menor de 15 metres. Aquesta distància és la que garantirà suficient mobilitat als compradors i tot i que en un primer moment pot semblar elevada, cal pensar que es veurà reduïda quan la incidència de la llum al receptor no sigui totalment recta. També, utilitzant alimentació amb pila i reduir aquesta el seu subministre de corrent al llarg de la seva vida útil, la distància de treball es reduirà progressivament.

Els factors que influeixen en la distància són l'elecció del parell led IR – detector i la capacitat del circuit emissor de donar suficient intensitat de corrent al led. Dins d'un mateix tipus de tecnologia, més intensitat equivaldrà a més distància.

Minimitzar el consum de l'emissor

En una aplicació alimentada per piles, la durada d'aquestes depèn del consum. En el cas dels comandaments IR tenim dos tipus de consum: molt alt durant el poc temps que s'envien dades i molt baix durant la resta de temps.

Si agafem com a referència el codi NEC, podem fer el càlcul del temps que el led està en funcionament:

Ton = nº bits x nº cicles/bits x ton/cicle

A on:

Ton: temps de funcionament del led
Nº bits: nombre de bits que enviem en total
Nº cicles/bits: nombre de cicles de la portadora per enviar un bit
Ton/cicle: temps de led per cada cicle de la portadora

Substituint:

Ton = 32 x 22 x 8.77uS = 6174uS

Veiem com per cada dada que enviem, el led només funciona durant 6'174mS (excloem l'enviament del leader). Centenars d'enviament de codis és traduiran en només uns segons de consum elèctric, però de molt alt consum elèctric ja que pot superar els 500mA. Per tant el problema del consum s'haurà d'atacar per les dos vies.

Control absolut dels temps

Independentment del codi que s'utilitzi, per generar-lo i descodificar-lo serà necessari un control exhaustiu del temps. Depenent de la freqüència de la portadora, s'ha de treballar amb resolucions de pocs microsegons. A la vegada, per a cada ràfega de llum només s'envien pocs cicles de llum, el que ens complica la utilització dels generadors de PWM integrats en la majoria de microcontroladors. Tot plegat obligarà a l'elecció de microcontroladors ràpids.

3.1.3.- Reducció de la complexitat

Tal i com ja s'ha comentat, i pel tipus de projecte del que es tracta, és important reduir la complexitat del desenvolupament del producte. Per això ja en els primers estadis del disseny es prenen dues decisions en aquest sentit:

- Utilitzar la mateixa família de microcontroladors per emissors i receptors
- Utilitzar el mateix programa/microcontrolador en tots els comandaments

Una sola família de microcontroladors

Tots els fabricants de microcontroladors agrupen els seus productes per famílies. Els membres d'una mateixa família estan fabricats amb la mateixa tecnologia, comparteixen la mateixa estructura lògica i tenen prestacions, com a mínim, comparables. Evidentment, també comparteixen a grans trets les instruccions i l'entorn de desenvolupament. És evident que en aquest cas, amb la necessitat de desenvolupar dos elements i amb produccions relativament discretes, serà preferible buscar una gamma de microcontroladors adient, abans que un model concret que ens aporti alguna petita avantatge tècnica o de preu però que ens dificulti el desenvolupament.

Programa/microcontrolador únic ens tots els comandaments

L'única diferència entre els comandaments és el nombre de botons i el nombre de codis que han de tenir memoritzats. Per aquest motiu tots els comandaments disposaran del mateix microcontrolador i del mateix programa. El que sí variarà serà el nombre de tecles, i per permetre-ho caldrà un sistema per deshabilitar les tecles no existents o, que estant físicament disponibles, no volem que siguin utilitzades.

3.2.- HARDWARE UTILITZAT

Si alguna cosa condiciona la resta del projecte, és l'elecció del parell emissor-receptor. Donat que es busquen les màximes prestacions possibles, les seves característiques marquen l'especificació del codi a utilitzar i la velocitat necessària dels microcontroladors. Per aquest motiu, son els components que primer es detallen.

3.2.1.- Receptor IR: Vishay TSOP 7000

Aquest mòdul receptor integra el conjunt fotodiode, amplificador i filtre de portadora. El motiu de més pes en la seva elecció és que es tracta del receptor que treballa a la freqüència més alta dels que s'han trobat al mercat: 455Khz. Cal recordar que les freqüències típiques de portadora en aquestes aplicacions van dels 33Khz (Mitsubishi code) als 56.7KHz (Thomson RCA code). Aquesta velocitat que decuplica les stàndart ens permet:

- Reduïr el consum de l'emissor.
- Disminuir el nombre de vegades que coincideixin dos o més comandaments en funcionament.

Les característiques més importants son:

Freqüència portadora	455KHz
Longitud ona	870nm
Nombre òptim de cicles per detectar	14
Tamanys de les ràfegues	22µS a 500µS
Cicle treball portadora	10% al 50%
Nivells sortida	TTL (sortida negada)
Distància detecció	20m amb led Vishay TSHF5400

Veiem com, a més de la freqüència de treball, permet utilitzar una codificació amb pocs cicles (menys temps per codificar) i que la part activa d'aquests només sigui d'un 10% (millora la duració de la bateria). El fabricant també garantitza una elevada distància de treball.

Els gràfics més interessants que ens facilita la documentació són els següents:

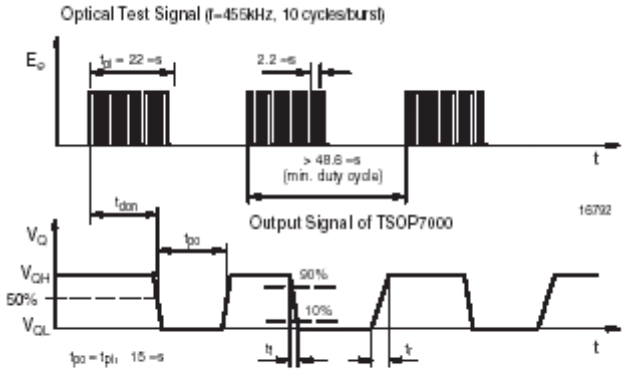


Fig 3.1: Sortida receptor

Senyal de sortida

Caldrà tenir en compte que:

$15\mu S < t_{don} < 36\mu S$

$-15\mu S < \Delta_{tpo} < +15\mu$

És a dir, el senyal de sortida del receptor conté variacions importants.

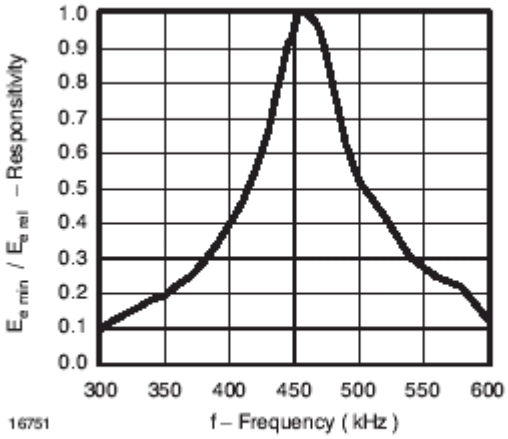


Fig 3.2: Resposta freqüencial

Resposta del filtre

Està centrada a 455KHz. Un error d'un 10% en la freqüència suposa una pèrdua del 50% en el filtre.

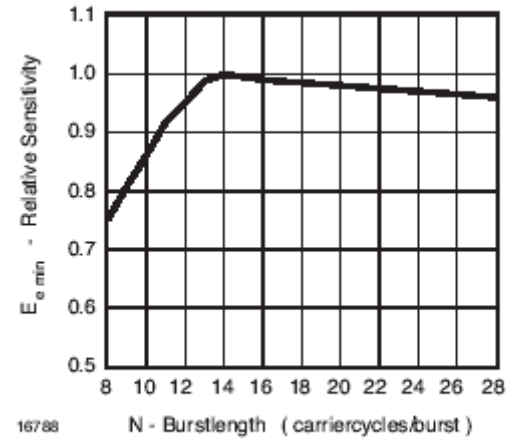


Fig 3.3: Sensibilitat vs tamany ràfega

Influència tamany de ràfega

S'aprecia que el tamany ideal de les ràfegues és de 14 cicles o, com a mínim, major de 12 cicles.

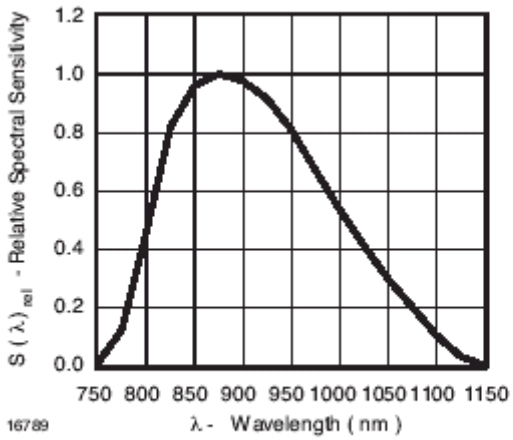


Fig 3.4: Sensibilitat vs longitud d'ona

Longitud d'ona

La major sensibilitat del receptor s'aconsegueix amb llum IR de 870nm. Caldrà tenir-ho en compte al escollir el led.

3.2.2.- Led IR: Vishay TSFF5400

Un cop decidit el receptor que s'utilitzarà, escollir el led adequat és una tasca simple. Ens cal un led que compleixi:

- Longitud d'ona de 870nm (o molt similar)
- Màxima potència
- Angle d'emissió acceptable, uns ±20º

I el led que s'hi adapta millor és el TSFF5400 de Vishay amb encapsulat Led de 5mm. És evident que no cal que sigui del mateix fabricant que el receptor, però en cap altre catàleg s'ha trobat (no vol dir que no existeixi) un led tan adequat.

Indicar que la mateixa casa disposa d'un led mes potent, TSFF5210, però que no és adequat per tenir un angle d'emissió d'únicament ±10º. A continuació les gràfiques mes interessants del 5400:

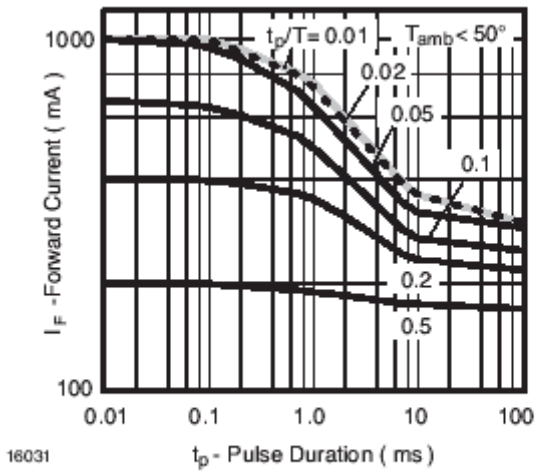


Fig 3.5: Consum vs duració pols

Consum VS duració pols

El consum del led és dependent de la duració del pols. Treballant a 455KHz, aquesta durada és, amb un cicle de 50% de 1.1µS, el que suposarà el consum màxim d'un amper.

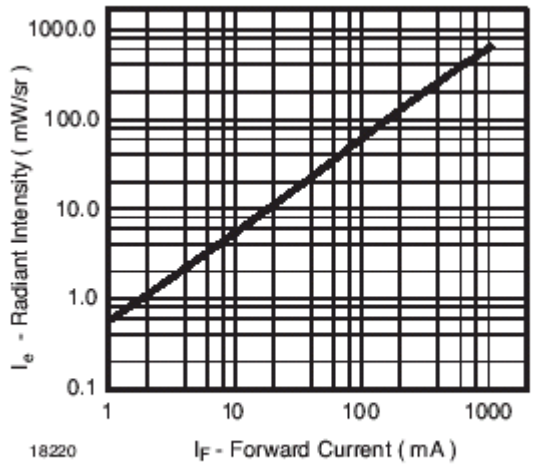
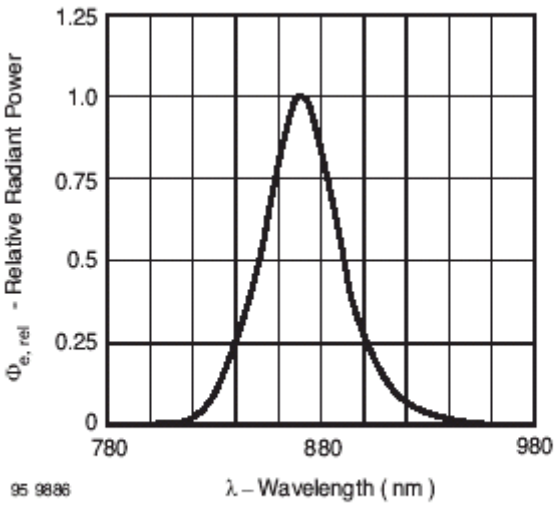


Fig 3.6: Radiació vs consum

Radiació VS consum

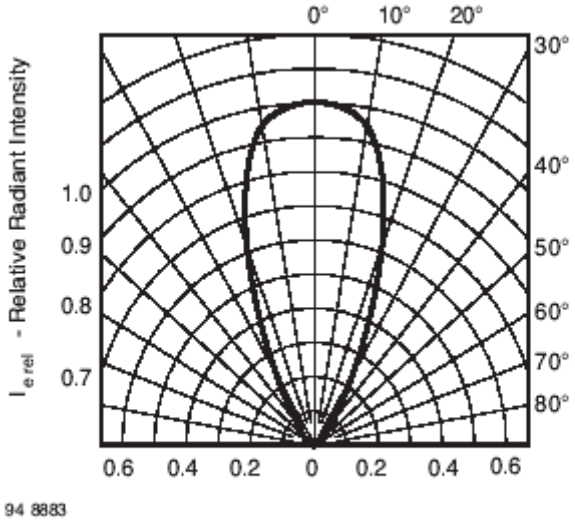
Aquesta gràfica indica que la potència de radiació (llum) és proporcional al consum. Conjuntament amb la gràfica anterior, podem comprovar que obtindrem del led la màxima potència d'emissió.



Longitud d'ona

Principalment emet llum a 870nm. Casa perfectament amb el detector

Fig 3.7: Espectre llum emesa



Distribució angular

Té una bona distribució fins i tot a llargues distàncies.

Fig 3.8: Distribució llum

3.2.3.- Microcontrolador: Microchip 16F627

Després de buscar informació de diversos microcontroladors de vuit bits (no necessitem fer càlculs ni tenim necessitats de memòria), vaig decidir utilitzar el 16F627A de Microchip.

Una de les seves principals característiques és la seva velocitat, amb un rellotge de 20MHz el temps d'instrucció és, excepte en instruccions de salt, de 200nS. Aquesta velocitat permetrà que el mateix microcontrolador pugui crear la portadora dels comandaments amb un timing exacte, estalviant-nos qualsevol circuit addicional. Les seves prestacions inclouen:

- Temps instrucció de 200nS
- Port sèrie integrat (nivells TTL)
- Memòria EEPROM de 128 bytes
- Memòria programa de 1024 words en FLASH
- 224 bytes de SRAM
- Mode sleep de baix consum
- Interrupció per canvi d'estat de les entrades
- Resistències de pull-up integrades
- Intensitat màxima de les sortides de 20mA

En el projecte s'utilitza la versió amb encapsulat PDIP de 18 potes, però estan disponibles encapsulats SOIC, SSOP i QFN.

3.2.4.- Altres consideracions sobre el hardware

En l'elecció del hardware, han influït aspectes diferents a les prestacions dels productes en sí.

En el cas de Microchip, el microcontrolador escollit forma part d'una amplia gamma de microcontroladors que comparteixen arquitectura i repertori d'instruccions. És una garantia que, arribat el cas d'haver de modificar les prestacions del sistema de comandaments, serà relativament fàcil utilitzar un nou microcontrolador sense que realitzar la migració suposi fer un nou projecte. També cal destacar que l'entorn de programació gratuït que proporciona la marca és realment complert i inclou un simulador fiable.

En el cas de Vishay cal destacar que el seu catàleg és, en diversitat, el més extens que he trobat, i amb documentació abundant. Així com tots els fabricants de microcontroladors consultats (Microchip, Atmel, Phillips,..) disposen d'una bona i accessible documentació, els fabricants d'optoelectrònica no són tant curosos. Per exemple, encara suposa un misteri saber si els leds de la casa Lumex son adequats pel projecte.

No és la meva intenció, i menys en un document eminentment acadèmic, fer propaganda de cap casa comercial, però sí fer notar que l'elecció d'un producte ha d'estar recolzada amb la seguretat d'obtenir suport suficient per part del fabricant.

3.3.- CODIFICACIÓ

El mètode de codificació inclou: la transformació d'un bit a un patró de llum, el nombre de bits d'informació a enviar i el seu significat i l'inclusió o no de bits de control i de detecció d'error.

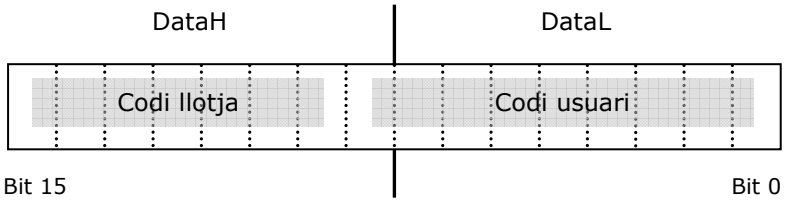
Com en qualsevol sistema de transmissió de dades, l'objectiu és que les dades rebudes siguin fiables. Aquest és el principal problema en comunicacions IR: el medi conté molt de soroll. Però també tenim en contra l'impossibilitat de realitzar un control d'accés al canal de comunicació. La conseqüència és que poden coincidir múltiples intents simultanis de comunicació des d'emissors diferents.

Per minimitzar el segon cas cal reduir el temps d'enviament de la informació. Quan mes curt, més difícil que coincideixin. Un pas important en aquest sentit ha estat l'elecció dels transductors, però també hi influeix la codificació, que inclou el tamany de les dades a enviar. Aquest tamany, però ha d'estar suficientment dimensionat per garantir de no quedar-nos sense codis disponibles si l'èxit comercial del sistema desborda les previsions.

3.3.1.- Tamany del codi i significat

La gestió dels comandaments (adquisició, assignació del codi) el fa la pròpia llotja que, en molts casos, n'és la propietària. El codi complert, identificarà l'usuari i la seva llotja origen. A més, l'encarregat de subhasta ha de disposar d'un comandament que permeti realitzar certes operacions, com arrancar i parar la subhasta o rectificar en qualsevol moment el preu inicial de la partida a subhastar.

Per fer-ho possible, el tamany de la informació útil a enviar és de 16 bits, repartits en dos bytes (DataH i DataL) de la següent manera:



Codi llotja: identifica la llotja propietària del comandament. 127 codis possibles.

Codi usuari: identifica l'usuari del comandament. 512 codis possibles.

Cas especial, llotja 0: Els codis amb codi de llotja 0 seran reservats pels comandaments dels encarregats de subhasta.

En total disposarem de 65024 codis diferents per a usuaris (127 x 512) i 511 codis possibles de control (el codi 0 no s'enviarà). Cal pensar que una llotja difícilment té mes de 200 usuaris autoritzats a assistir a la subhasta.

3.3.3.- Portadora

La portadora que podem utilitzar ve definida per les característiques del receptor, resumides en l'apartat 3.2.1. Els valors utilitzats seran els següents:

- Freqüència: 454.455KHz, (període de 2,2 µS)
- Cicle de treball: 9.1% (0.2µS on / 2µS off)[1]
- Tamany mínim de ràfega: 30.8µs (14 cicles)

La freqüència utilitzada varia un ínfim 0.1% de la ideal i, molt possiblement, serà una error menor a les diferències entre diverses unitats del receptor. Com el temps d'instrucció del microcontrolador és de 0.2µs, es podrà generar sense problemes amb el valor adequat i el cicle de treball mínim donat pel fabricant.

3.3.3.- Trama IR

La codificació dels bits es realitzarà mitjançant codificació bi-fase utilitzant la portadora definida anteriorment. En aquest mètode es codifiquen els uns i zeros amb u flanc de pujada o baixada segons el valor del bit:

Fig 3.10: Representació de 1 i 0 per flancs

Podem dir que cada bit està composat de dos semibits (o bits de llum): 10 en el cas de codificar un 0 i 01 en cas de codificar un 1. Un semibit 1 es traduirà en una ràfega de 14 cicles de portadora, que és el nombre de cicles òptim per ser detectat pel receptor.

Qualsevol codi es traduirà en un seguit de ràfegues de 14 o 28 cicles (de 30.8µs i 61.6µs respectivament) separats per espais de 30.8µs o 61.6µs. Per la construcció del codi, és evident que mai tindrem més de dos semibits iguals.

La trama complerta es composarà d'un 1 com a bit d'Start, seguit dels 16 bits del codi (7 de codi de confraria + 9 de codi d'usuari) començant pel de mes pes. Per exemple, per l'usuari amb el codi 327 de la confraria amb codi 25, s'enviarà la següent cadena de bits:

1 0011001 101000111

[1] En la pàg. 5 del datasheet del receptor (doc. number 82147 rev. 6, Vishay) es dona com a valor de cicle mínim 'between 50% (1.1µs pulses) and 10% (0.2µs pulses)'. Aqui he preferit posar el percentatge real.

Mostrada en semibits:

01 10100101101001 011001101010010101

En detall, la trama IR generada serà la següent:

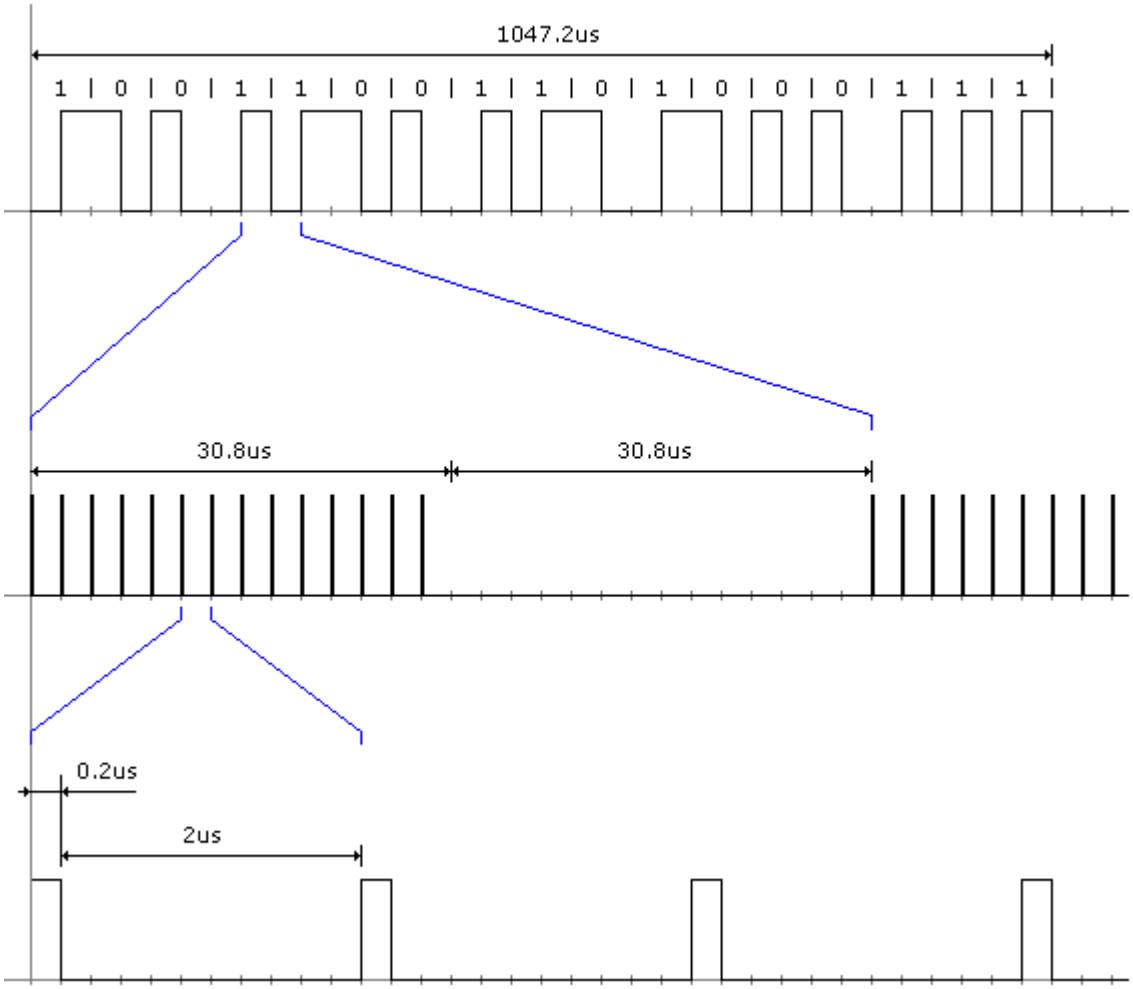


Fig 3.11: Trama utilitzada

Aquest mètode assegura que, independentment del codi a enviar, la trama sempre compleix determinades propietats que l'algorisme de descodificació del receptor haurà de controlar. Si la trama rebuda no les compleix és podrà descartar per errònia. Aquestes son les principals:

- Contenir únicament pulsos i espais de 30.8µs o 61.6µs.
- Llargada total de la trama de 1047.2µs
- Existència d'un flanc al centre de cada bit.

Qualsevol alteració provocada per soroll o la suma de les trames de dos o mes comandaments trencarà una d'aquestes propietats i serà detectada.

4.- EMISSOR

4.1.- DESCRIPCIÓ

4.1.1.- Especificacions

Teclat	Matricial 16 tecles. Extern.
Codis	16 bits (7 codi confraria + 9 codi usuari) Programables per cada tecla mitjançant port sèrie.
EEPROM	128 bytes. (32 per codis)
XTAL	20MHz
T. envio trama	1047.2µs
V. Transmissió	16233bps
Alimentació	Pila 9v
Tensió MC	5v
Consum repòs	8µA
Consum en emissió	<20mA

4.1.2.- Contingut EEPROM

La informació indispensable que conté la EEPROM són els codis assignats a cada tecla i el nombre de vagades que s'envia el codi, però s'ha aprofitat per afegir altre informació per aprofitar els 128 bytes disponibles. Així, el mapa de la EEPROM queda com segueix:

Posició	Dada
00	Versió codi
01	Sub–versió codi
02 – 03	Codi tecla 1
04 – 05	Codi tecla 2
06 – 07	Codi tecla 3
08 – 09	Codi tecla 4
0A – 0B	Codi tecla 5
0C – 0D	Codi tecla 6
0E – 0F	Codi tecla 7
10 – 11	Codi tecla 8
12 – 13	Codi tecla 9
14 – 15	Codi tecla 10
16 – 17	Codi tecla 11
18 – 19	Codi tecla 12
1A – 1B	Codi tecla 13
1C – 1D	Codi tecla 14
1E – 1F	Codi tecla 15
20 – 21	Codi tecla 16
22	Nombre repeticions
23	Codi confraria
24 – 37	Nom confraria (txt 20 char)
38 – 3A	Data ultima modificació (dd – mm – aa)
3B – 7F	Cadena lliure (txt 69 char)

El codi del comandament permet modificar el contingut de la EEPROM mitjançant el port sèrie. Cal però, definir un petit protocol per fer-ho possible.

4.1.3.- Protocol programació EEPROM

Aquest protocol està pensat per realitzar la lectura o escriptura d'un byte de la EEPROM. L'espai de direccions és de 128 posicions que contenen un byte cada una.

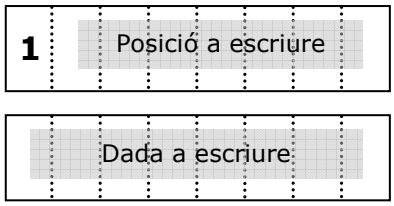
Cap de les operacions utilitza un control d'errors. L'únic a verificar és que el comandament ha realitzat l'operació, ja que sempre torna resposta, sigui una dada (en el cas de lectura) o un ACK (cas de l'escriptura). S'ha renunciat al control d'errors pel tamany bàsic de les dades a tractar, que aquestes permeten un control mitjançant la repetició d'ordres i que, en l'improbable cas de produir-se un error, aquest es detecta molt ràpidament durant el funcionament normal.

La configuració del port sèrie és 8,n,1 a 9600 bps.

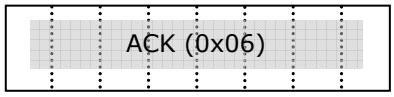
Escriptura d'una posició

Per a aquesta operació s'envien dos bytes. El primer conté un bit indicant l'operació a realitzar (bit 7, ha de valer 1) i 7 bits amb la posició a modificar. Posteriorment s'ha d'enviar un byte amb la dada a inserir. Quan el comandament ha acabat el procés d'escriptura de la EEPROM (tarda uns 4ms), enviarà un byte d'ACK (valor 0x06).

Programador a comandament:



Comandament a programador:



Lectura d'una posició

S'envia un sol byte que conté l'operació a realitzar (lectura) i una posició. El comandament tornarà un byte amb el contingut d'aquesta posició. No es realitza cap control d'errors.

Programador a comandament:

Comandament a programador:

4.1.4.- Esquema elèctric i PCB

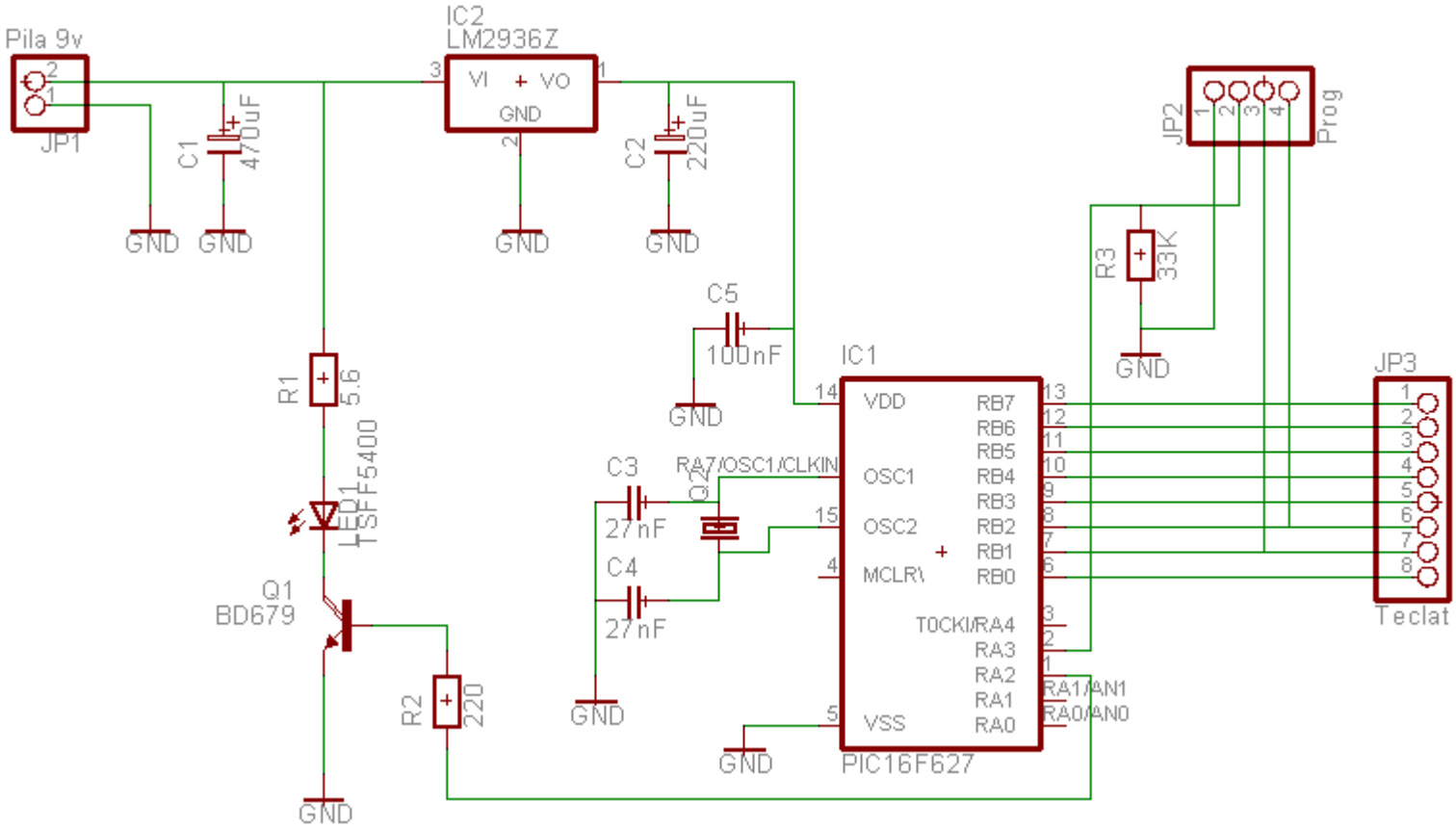


Fig 4.1: Esquema emissor

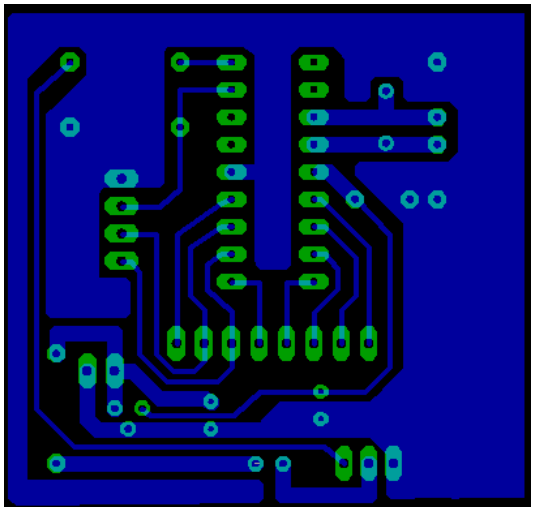


Fig 4.2: Circuit imprès, pistes

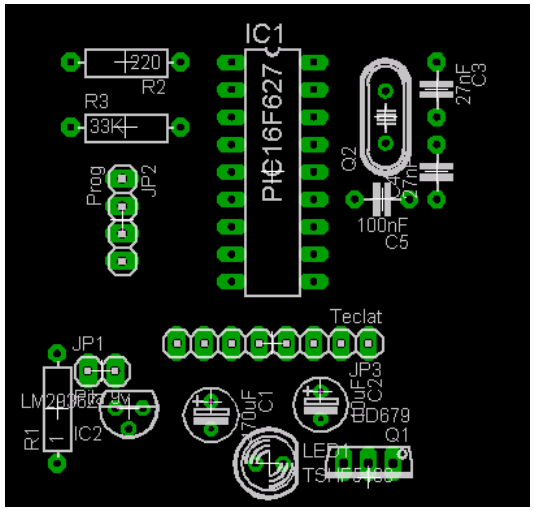


Fig 4.3: Circuit imprès, posició components

4.2.- PARTS

4.2.1.- Teclat

S'utilitza un teclat matricial amb el següent esquema:

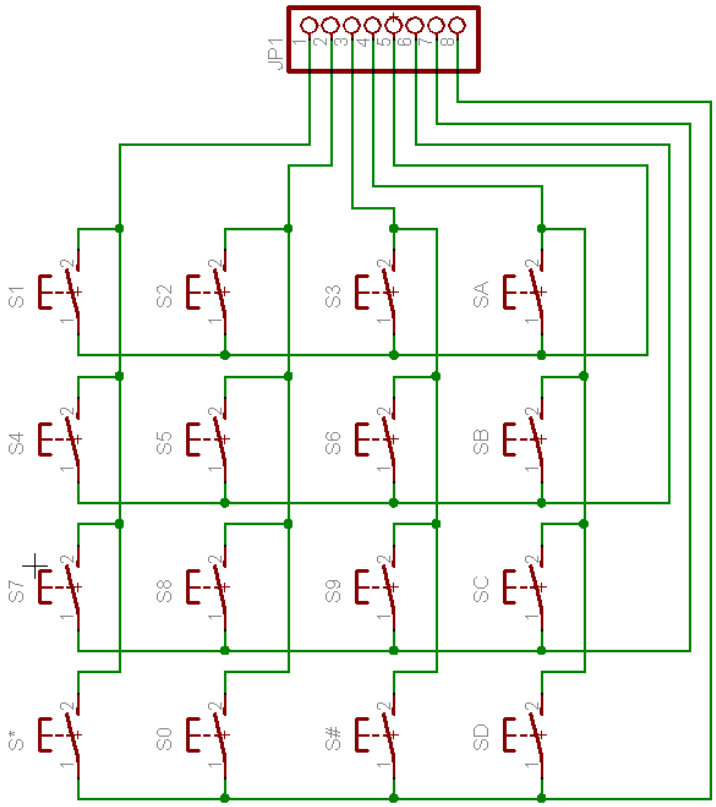


Fig 4.4: Esquema teclat

El teclat s''interconnecta al microcontrolador mitjançant el PortB, configurant els bits 0 a 3 com a sortides (columnes) i els bits 4 a 7 (files) com a entrades. A més, en el moment de configurar les sortides, s'activen les resistències de pull-up que porta integrat aquest port, estalviant-nos incorporar-les externament. Aquestes resistències suposen un consum aproximat de 200µA per pin, però es desactiven automàticament quan el dispositiu està en mode de baix consum.

4.2.2.- Alimentació

El microcontrolador s'alimenta a 5v. Aquest valor, dependent de la freqüència de treball, s'extreu del següent gràfic:

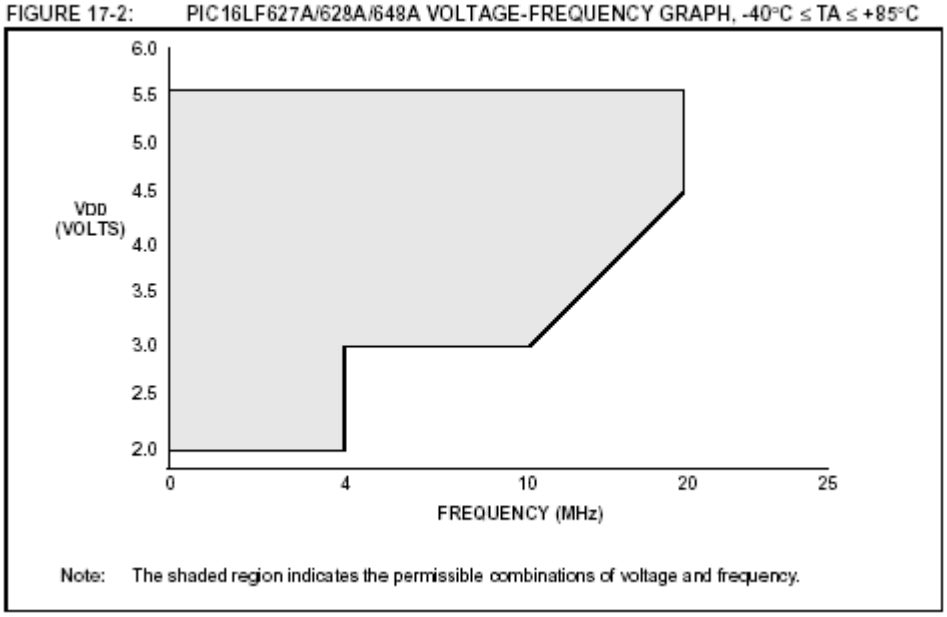


Fig 4.5: Tensió alimentació vs freqüència

Un regulador de tensió (LM2936-5) és el que redueix la tensió d'alimentació dels 9v de la pila als 5v que necessita el microcontrolador. El regulador té que subministrar tensió encara que el microcontrolador estigui en mode iddle, pel que és prioritari que el seu consum sigui mínim. Segons la següent taula, en aquestes condicions, el regulador absorbeix una intensitat de 8µA, quan en un regulador típic com el LM7805 és d'uns 4.5 mA:

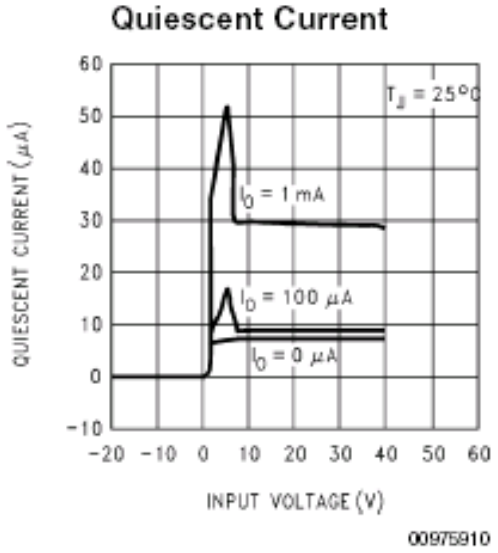


Fig 4.6: Consum del regulador

4.2.3.- Potència

La part de potència permet controlar amb la sortida del microcontrolador de 20mA el led IR que pot consumir un pic d'1 amper. En total comprèn el transistor Q1, el led IR, la resistència del led R1 i la resistència de base R2:

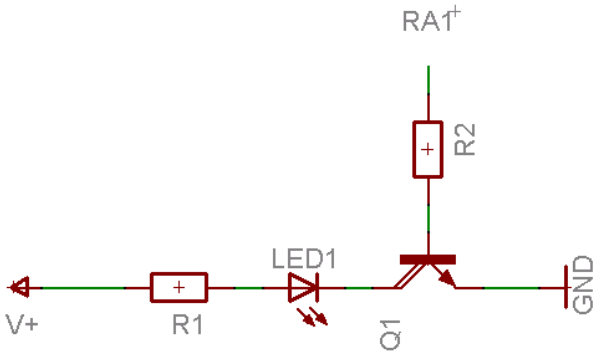


Fig 4.7: Circuit potència led

La intensitat i la tensió del led l'obtenim de les següents taules:

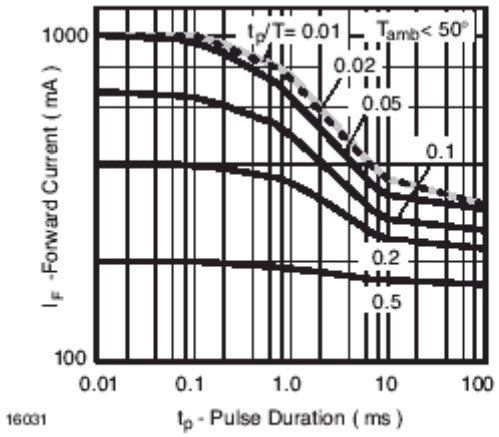


Fig 4.8: Intensitat vs durada pols

Valors de la portadora utilitzada**:**

t_p=0.0022ms
t_p /T=0.1

Per tant, la intensitat màxima del led és d'uns 700mA

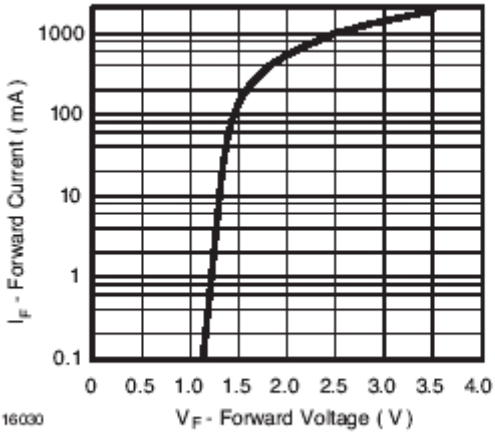


Fig 4.9: Intensitat vs tensió

Amb el valor de 700mA trobat en la gràfica anterior, aquesta ens indica una tensió d'uns 2.25v

Amb el que obtenim una intensitat d'uns 700mA amb una tensió de 2.25v. Donat que els ports del microcontrolador poden donar un màxim de 20mA, el transistor utilitzat cal que disposi d'una β mínima de 70 i permetre funcionar al port a uns 10mA amb seguretat, pel que s'utilitzarà el transistor BD679. En les seves fulles de característiques hi figura la V_{BE} i la V_{CE}:

$V_{CE(sat)}$*	Collector-Emitter Saturation Voltage	for **BD677/678/679/680/681/682** I_C = 1.5 A I_B = 30 mA for **BD677A/678A/679A/680A** I_C = 2 A I_B = 40 mA			2.5 2.8	V V
V_{BE}*	Base-Emitter Voltage	for **BD677/678/679/680/681/682** I_C = 1.5 A V_{CE} = 3 V for **BD677A/678A/679A/680A** I_C = 2 A V_{CE} = 3 V			2.5 2.5	V V

Pel que podem fer els següents càlculs:

$$\mathbf{R2} = (V_{RA1} - V_{BE})/ I_B = (5 – 2.5)/0.01 = 250\ \Omega$$
$$\mathbf{R1} = (V – V_{CE} - V_{LED1)} / I_{LED1} = (5 – 2.5 – 2.25)/0.7 = 6.07\Omega$$

Els valors normalitzats seleccionats finalment són:

R2 = 220Ω
R1 = 5.6Ω

4.2.4.- Connector port sèrie

El comandament disposa d'un conector de 5 pins per realitzar la programació del port sèrie:

Pin	Significat	Port MC
1	GND	--
2	Connector endollat	RA3
3	Rx (del microcontrolador)	RB1
4	Tx (del microcontrolador)	RB2

El port funciona a nivells TTL, per tant caldrà un conversor de nivells si el programador és un PC o, en general, qualsevol dispositiu amb nivells RS-232.

El cable programador ha d'alimentar el pin 2 a 5 volts. D'aquesta manera, el comandament pot saber si té un cable programador connectat. Els pins del port sèrie s'utilitzen normalment per fer la lectura del teclat, pel que el soft en canviarà la funció segons el valor del pin 2 del connector. També haurà d'incorporar una resistència (es recomana de 10k) entre el pin 3 i la linia Tx del port (connectada a Rx del comandament), per evitar un cortcircuit abans no es canvia la configuració del port B.

4.3.- CODI

En aquest apartat s'explicarà les característiques més importants del codi, com està estructurat i la rutina _EnviaCodi que és la que fa l'enviament efectiu del codi per IR.

4.3.1.- Estructura del programa

A grans trets, les dues tasques que ha de realitzar el programa són l'enviament de codis i la programació de l'EEPROM. La selecció de la tasca a realitzar es fa segons el valor del pin RA3, si està a nivell alt és que tenim un cable de programador connectat i s'han de programar codis, si està a nivell baix s'ha d'enviar un codi segons la tecla polsada.

Normalment, el comandament està en mode de baix consum. Només la pulsació d'una tecla pot 'despertar-lo' per començar a operar. Quan això passa, s'activa l'interrupció OPC (On Pin Change) i es salta a la posició 0x04 que és a on estan vectoritzades totes les interrupcions. Únicament es desactiva el flag d'interrupció activa i es retorna el control al programa principal. Per aquest motiu, per entrar en programació cal endollar el cable per, posteriorment, polsar alguna tecla, serà llavors quan el programa decidirà que ha de fer.

La estructura del programa és la següent:

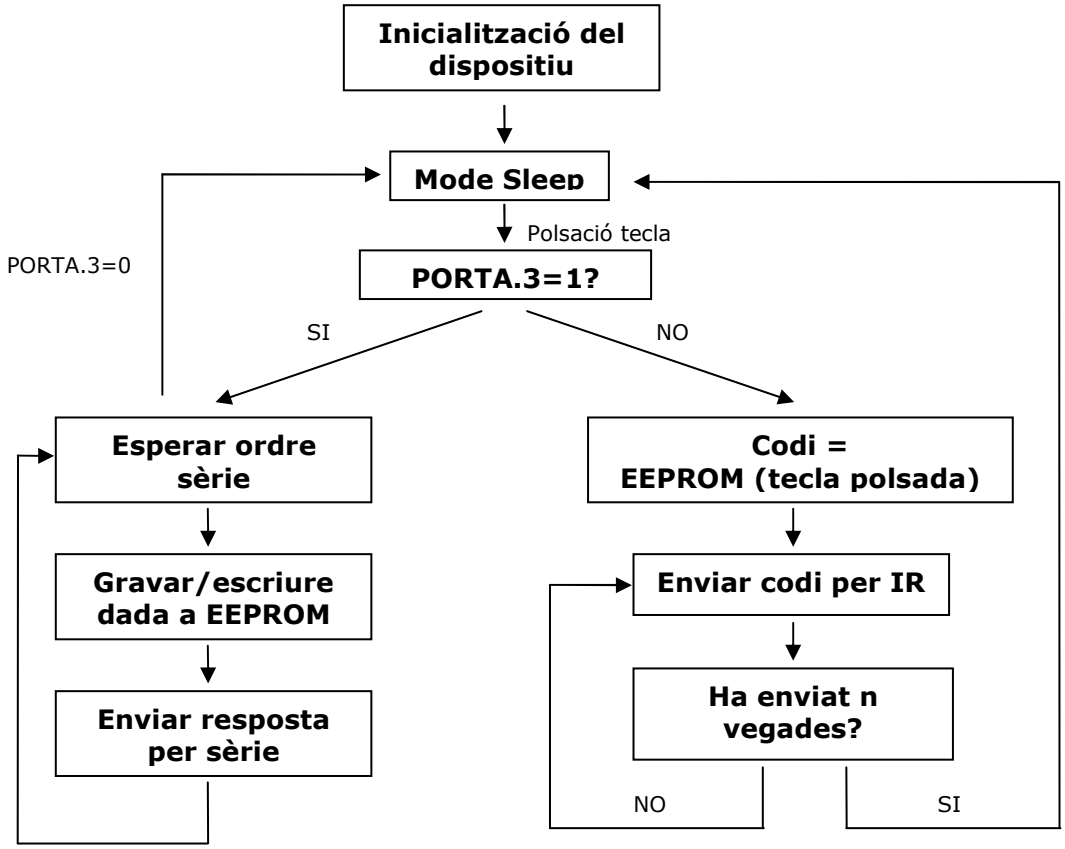


Fig 4.10: Estructura programa emisor

4.3.2.- Rutines

En aquest programa l'existència de rutines es deu únicament a ordenar el codi. Únicament _GetByte es crida desde tres punts diferents.

Totes les variables (registres) utilitzats estan declarades en l'inici del llistat. Dotze registres i el registre W (registre de treball) son suficients per fer el pas de paràmetres entre rutines i crear els comptadors necessaris.

A continuació, breu explicació de les tasques que realitzen cada una d'elles:

_Inici			Configura el microcontrolador i els seus perifèrics
_Main			Activa int teclat i entra en mode sleep. Quan surt crida _OpCodis o _OpMando segons el valor de PORTA.3
_OpCodis			Segons l'ordre rebuda pel sèrie, llegeix o escriu byte a l'EEPROM i torna resposta pel sèrie.
	_GetByte		Recupera el byte de la posició n_Adr de l'EEPROM i el torna per n_Data
	_PutByte		Guarda el byte n_Data a la posició n_Adr de l'EEPROM
_OpMando			Envia per IR el codi associat a la tecla polsada vàries vegades.
	_LecturaTeclat		Retorna per n_Tecla quina tecla està polsada (0 si no n'hi ha cap de polsada)
	_GetCodi		Recupera el codi (2 bytes: DataH/DataL) associat a n_Tecla
	_EnviaCodi		Envia el codi DataH/DataL per IR
		_Delay	Realitza una espera de 2.2mS
_Interrupt			ISR. Reseteja el flag de l'interrupció OPC.

4.3.3.- Rutina _EnviaCodi

Aquesta rutina s'explica en detall per ser la més delicada (que no complexe) del programa. Conforma la trama IR i ho fa seguint escrupolosament els temps marcats pel format explicat en l'apartat 3.3.3.

El codi a enviar està en els registres DataH i DataL, essent el bit de més pes (i el primer que s'envia) el bit 7 de DataH. El primer que es fa és rotar un bit a la dreta aquest dos bytes i posar a 1 el bit 7 de DataH, que serà el bit d'Start. D'aquesta manera, el bit 0 de DataL queda guardat al flag C (carry) del registre STATUS. A partir d'aquesta inicialització, el procediment a seguir és enviar el bit DataH.7 i rotar el parell DataHL un bit a l'esquerra. Aquests dos passos s'aniran repetint fins enviar 17 bits en total (16 de codi + 1 d'Start).

```
movlw  0x11
movwf  n_Bit
rrf    n_DataH, F   ;Podem guardar el bit DataL.0 a C perquè ningu
rrf    n_DataL, F   ;ens el modifica abans de fer la primera rotació
bsf    n_DataH, 7

movlw  kCiclesBurst ;Carreguem el nombre de cicles de burst
movwf  cntCicles
```

Inicialització registres

Dos registres importants son n_Bit i cntCicles. n_Bit conté el nombre de bits que resten per enviar, mentre que cntCicles guarda, per cada semibit, el nombre de cicles que queden per executar. Aquest últim s'haurà de recarregar dues vegades per cada bit enviat.

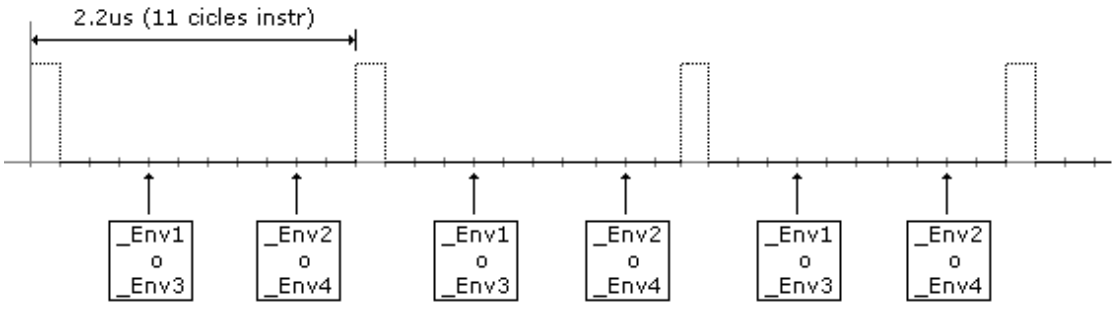
Per enviar un bit, es realitzen 14 cicles d'un bucle per fer el primer semibit i 14 cicles d'un altre bucle per formar el segon semibit. En el primer s'activa el led 0,2µs cada 2µs si DataH.7 és 1 mentre que en el segon bucle s'activa el led si DataH.7 és 0. Enmig dels dos bucles es realitza la recàrrega del registre cntCicles.

Cada cop que s'envia un bit, es decrementa el registre n_Bit (que estava inicialitzat a 17) fins que aquest arriba a 0, indicador de que ja s'ha enviat tot el codi.

```
;---------------------------- Repetim per a cada bit ----------------
_Env1  nop                  ;----- Enviem primer semibit
       nop
       nop
       nop
       nop
_Env2  btfss  n_DataH,7     ;Nomes activem el led si el bit és 0
       bsf    LED
       bcf    LED
       decfsz cntCicles,F
       goto   _Env1

       rlf    n_DataL,F     ;Rotem byte baix del codi ara. Mes endavant no
                            ;tindrem temps. C passa a guardar el bit DataL.7
       movlw  kCiclesBurst
       movwf  cntCicles
       nop                  ;Per cuadrar temps
       goto   _Env4

_Env3  nop                  ;----- Enviem segon semibit
       nop
       nop
       nop
       nop
_Env4  btfsc  n_DataH,7     ;Nomes activem el led si el bit és 1
       bsf    LED
       bcf    LED
       decfsz cntCicles,F
       goto   _Env3
;--------------------------- Fi enviament d'un bit -----------------
       rlf    n_DataH,F     ;Rotem byte alt del codi.
                            ;C encara conté bit 7 de n_DataL

       movlw  kCiclesBurst ;Carreguem el nombre de cicles del burst
       movwf  cntCicles
       decfsz n_Bit,F       ;Hem enviat tots els bits?
       goto   _Env2         ;      No: anem a per el següent
       return               ;      Si: acabem
```

Enviament del codi

En el següent gràfic es relaciona l'evolució del programa amb l'evolució de la trama de llum. Cada divisió de l'eix correspon al temps de cicle d'una instrucció (que és de 0.2µs). Únicament les instruccions de salt (goto) tenen un temps de cicle de 0.4µs. Les instruccions btfss i btfsc (salta instrucció si bit igual a u o bit igual a zero respectivament) substitueixen la instrucció saltada per un nop, pel qual sempre consumeixen el mateix nombre de cicles.



Les etiquetes _Env1 i _Env2 formen part del primer semibit mentre que les etiquetes _Env3 i Env4 formen part del segon semibit. Encara que s'han dibuixat tots els pulsos, cal notar que només existiran quan s'estigui enviant el primer semibit i DataH.7 sigui u o quan s'estigui enviant el segon semibit i DataH.7 sigui zero.

El codi compleix les següents distàncies (en nombre de cicles) i fa les següents tasques entre els punts etiquetats:

Recorregut	**Dist.**	**Tasca**
De **_Env1** a **_Env1**	11	S'envia un cicle, es decrementa cntCicles i es comprova si és zero
De **_Env4** a **_Env4**	11	S'envia un cicle, es decrementa cntCicles i es comprova si és zero
De **_Env2** a **_Env4**	11	Es recarrega cntCicles i es rota DataL
De **_Env4** a **_Env2**	11	Es recarrega cntCicles, es rota DataL, es decrementa n_Bit i es comprova si es zero

Quan es passa del primer semibit al segon semibit, el programa rota DataL, recarrega cntCicles i salta directament a _Env4 sense passar per _Env3 i executar els nops situats entre _Env3 i _Env4. Així es recupera el temps invertit en les operacions realitzades.

De manera similar, quan es passa d'un bit al següent, es recarrega cntCicles, es decrementa n_Bit i, enlloc de passar per _Env1, es salta a _Env2 sense executar els nops situats entre _Env1 i _Env2.

4.3.4.- Llistat EMISSOR.ASM

```
        List p=16F627a
        __Config 0x210A       ;Configuració: memoria no protegida, no habilita Watchdog
        include "P16F627a.INC"

;------ DEFINICIO DE VARIABLES I CONSTANTS ----------------------------------------------
;
;--------------------------------------------------------------------------------------

;----- Constants ------

#define kCiclesBurst 0x0E     ;Numero de cicles de cada burst
#define LED PORTA,2           ;Sortida utilitzada per al led

;----- Registres ------

n_Bit           equ     0x21    ;Numero de bit de la trama que s'envia
n_Fila          equ     0x22    ;Per rutina de teclat
n_Tecla         equ     0x23    ;Tecla apretada
cntCicles       equ     0x24    ;Comptador de cicles del burst realitzades

n_DataL         equ     0x25    ;Byte baix del codi a llegir/guardar EEPROM
n_DataH         equ     0x26    ;Byte1 alt codi a llegir/guardar EEPROM

e_DataL         equ     0x27    ;Byte baix del codi a enviar
e_DataH         equ     0x28    ;Byte alt del codi a enviar

n_T1            equ     0x29    ;Comptador per temporitzar entre enviaments
n_T2            equ     0x30    ;Comptador per temporitzar entre enviaments

n_Data          equ     0x31    ;Dada EEPROM
n_Adr           equ     0x32    ;Adreça EEPROM

                org     0x00

                goto    _Inici

                org     0x04

;----- ISR -----------------------------------------------------------------------------
;
;       Desactiva el flag i torna el control al programa.
;
;--------------------------------------------------------------------------------------
_Interrupt      bcf     INTCON,RBIF     ;Elimina flag On Pin Change.
                return                  ;No habilitem les interrupcions

;----- INICIALITZACIONS ----------------------------------------------------------------
;
;       Tot el seguit d'inicialitzacions que cal fer: ports de sortida, port serie,
;    deshabilitzar comparadors, interrupcions i inicialitzar registres.
;
;--------------------------------------------------------------------------------------

_Inici          bsf     STATUS,RP0      ;Selecciona bank1
                movlw   0x00            ;Pull-up entrades habilitades
                movwf   OPTION_REG
                movlw   0x08
                movwf   TRISA           ;PortA:3 com a entrada, resta son sortides
                movlw   0xF0
                movwf   TRISB           ;PortB4:7 IN, PortB0:3 OUT
                movlw   0x81            ;Conf. serie: 9600 assíncron. Encara no s'habilita.
                movwf   SPBRG
                movlw   0x04
                movwf   TXSTA
                clrf    PIE1            ;Desactivem interrupcions
                bcf     STATUS,RP0      ;Selecciona bank0

                movlw   0x07
                movwf   CMCON           ;Desactiva els comparadors
                clrf    INTCON          ;Deshabilitem totes les interrupcions
                clrf    PORTA           ;Sortides a 0
```

```
;----- PROGRAMA PRINCIPAL --------------------------------------------------------
;
;       Posa el micro en sleep. Quan es desperta pel On Pin Change consulta si tenim que
;       enviar codis o programar la EEPROM
;
;-------------------------------------------------------------------------------


_Main           clrf    PORTB           ;Cap sortida activa
                movf    PORTB,F         ;Lectura PortB abans d'sleep
                movlw   0x88            ;Activem interrupció teclat (OPC)
                movwf   INTCON
                sleep
                nop

                btfsc   PORTA,3         ;Si esta a 1, entrem en programacio
                call    _OpCodis
                btfss   PORTA,3         ;Si esta a 0, funcio comanament
                call    _OpMando
                goto    _Main


;----- OPCODIS -------------------------------------------------------------------
;
;       Fa l'operacio de programacio de codis. El primer byte rebut indica una posicio
;       de la EEPROM i si es vol llegir (bit7 = 0) o gravar (bit7 = 1). En el primer cas
;       tornem pel serie el byte llegit, mentre que en el segon cas esperem el byte a
;       gravar, el guardem i enviem ACK. En cap cas enviem NACK.
;
;-------------------------------------------------------------------------------

_OpCodis        bsf     RCSTA,CREN              ;Habilita recepcio serie
                bsf     STATUS,RP0              ;Bank1
                bsf     TRISB,1                 ;Pin Rx com entrada
                bsf     TXSTA,TXEN              ;Habilita transmissió serie
                bcf     STATUS,RP0              ;Bank 0
                bsf     RCSTA, SPEN             ;Habilita pins port serie

_CEspera        btfss   PORTA,3                 ;Si esta a 0, acabem
                goto    _CFi
                btfss   PIR1, RCIF              ;Si tenim dada tira endavant
                goto    _CEspera
                btfss   RCSTA, OERR             ;En cas d'error Overrun
                btfsc   RCSTA, FERR             ;  ..o d'error Framing
                goto    _CError                 ;  ..sortim
                movfw   RCREG                   ;Llegim la dada rebuda
                movwf   n_Adr
                btfss   n_Adr,7                 ;Segons el bit de mes pes
                goto    _CGet                   ;   0: llegim
                bcf     n_Adr,7
                goto    _CPut                   ;   1: gravem

_CError         movfw   RCREG                   ;Buidem buffer sèrie. Teòricament no cal
                bcf     RCSTA,CREN              ;Resetejem errors deshabilitant i...
                bsf     RCSTA,CREN              ; ...habilitant sèrie
                goto    _CEspera

                                        ;-------------- LLEGIR ----------------
_CGet           call    _GetByte                ;Recupera byte pos. n_Adr
_C2             btfss   PIR1,TXIF               ;Espera a que el port estigui a punt
                goto    _C2
                movfw   n_Data
                movwf   TXREG                   ;Envio n_Data
                goto    _CEspera                ;Torna a esperar ordre

                                        ;-------------- GRAVAR ----------------
_CPut           btfss   PORTA,3                 ;Si esta a 0, acabem
                goto    _CFi
                btfss   PIR1, RCIF              ;Si tenim dada continua
                goto    _CPut
                btfsc   RCSTA, OERR             ;En cas d'error..
                goto    _CError
                btfsc   RCSTA, FERR
                goto    _CError                 ;..marxem
                movfw   RCREG
                movwf   n_Data                  ;Ja tenim el byte a guardar
```

```
                bcf     RCSTA,CREN              ;Deshabilitem recepcio serie
                call    _PutByte                ;Gravem el byte
                bsf     RCSTA,CREN              ;Tornem a habilitar el sèrie

_C3             btfss   PIR1,TXIF               ;Espera a que el sèrie estigui a punt
                goto    _C3

                movlw   0x06
                movwf   TXREG                   ;Envio ACK
                goto    _CEspera                ;S'en torna a esperar algo

_CFi            bcf     RCSTA, SPEN             ;Desabilita pins port serie
                bcf     RCSTA,CREN              ;Desabilita recepcio serie
                bsf     STATUS,RP0              ;Bank1
                bcf     TRISB,1                 ;Pin Rx com sortida
                bcf     TXSTA,TXEN              ;Deshabilita transmissió serie
                bcf     STATUS,RP0              ;Bank 0
                return


;----- PUTBYTE --------------------------------------------------------------------
;
;       Grava el byte n_Data a la posicio n_Adr de la EEPROM. Reintenta la gravacio fins
;       aconseguir-ho. Un malfuncionament de la EEPROM <=> Micro penjat. Ho acceptem
;
;--------------------------------------------------------------------------------------

_PutByte        movfw   n_Adr
                bsf     STATUS,RP0              ;Bank1
                movwf   EEADR                   ;Carrega adreça
                bcf     STATUS,RP0              ;Bank0
                movfw   n_Data
                bsf     STATUS,RP0              ;Bank1
                movwf   EEDATA                  ;Carrega dada
                bsf     EECON1,WREN             ;Habilita escriptura EEPROM

                movlw   0x55                    ;Seqüència necessària per manual
                movwf   EECON2
                movlw   0xAA
                movwf   EECON2

                bsf     EECON1,WR               ;Escriu
                bcf     STATUS,RP0              ;Bank0

_P1             btfss   PIR1,EEIF               ;Espera a acabar l'escriptura
                goto    _P1
                bcf     PIR1,EEIF               ;Reset del flag

                bsf     STATUS,RP0              ;Bank1
                bcf     EECON1,WREN             ;Deshabilita escriptura EEPROM
                movfw   EEDATA
                bsf     EECON1,RD               ;A llegir (No hem modificat EEADR)
                subwf   EEDATA,W                ;Restem dada gravada i llegida
                bcf     STATUS,RP0              ;Bank0
                btfss   STATUS,Z                ;Resultat és zero?
                goto    _PutByte                ;       No: Ho tornem a intentar
                return                          ;   Si: Acabem funció


;----- GETBYTE --------------------------------------------------------------------
;
;       Recupera el byte n_Data de la posicio n_Adr de la EEPROM
;
;--------------------------------------------------------------------------------------

_GetByte        movfw   n_Adr
                bsf     STATUS, RP0             ;Bank 1
                movwf   EEADR                   ;Carrega adreça
                bsf     EECON1, RD              ;Ordre lectura
                movfw   EEDATA                  ;Recupera dada llegida
                bcf     STATUS, RP0             ;Bank 0
                movwf   n_Data                  ;Guarda dada
                return


;----- OPMANDO --------------------------------------------------------------------
```

```
;
;       Fa l'operacio de mando. Envia per IR el codi de la tecla polsada
;
;-------------------------------------------------------------------------------

_OpMando        call    Lectura_Teclat

                andlw   0xFF            ;Acaba si no hi ha cap tecla apretada
                btfsc   STATUS,Z
                return

                call    _GetCodi        ;Busca el codi a enviar

                movf    n_DataH,F       ;Si el codi es zero, torna
                btfss   STATUS,Z
                goto    _M1
                movf    n_DataL,F
                btfsc   STATUS,Z
                return

_M1             movlw   0x22            ;Busca el ombre de vegades a enviar
                movwf   n_Adr
                call    _GetByte

_M2             movfw   n_DataH         ;Copia codi a enviar. _EnviaCodi el modifica
                movwf   e_DataH
                movfw   n_DataL
                movwf   e_DataL

                call    _EnviaCodi      ;Envia el codi

                decfsz  n_Data          ;Ha enviat n_Data vegades?
                goto    _M3                     ;       No: salta
                return                          ;       Si: acaba

_M3             call    _Delay          ;Espera una estona entre enviaments
                goto    _M2

;-----  DELAY ----------------------------------------------------------------------
;
;       Realitza una espera de 2.2 mS amb dos contadors anidats
;
;-------------------------------------------------------------------------------

_Delay          movlw   0x0E    ;Recarrega comptador n_T1
                movwf   n_T1
_D1             movlw   0x00            ;Recarrega comptador n_T2
                movwf   n_T2
_D2             decfsz  n_T2            ;Decrementa n_T1. Es zero?
                goto    _D2             ;       No: hi torna
                decfsz  n_T1            ;       Si: Decrementa n_T2. Es zero?
                goto    _D1             ;                       No: Recarrega n_T1
                return                  ;                       Si: Acaba
;----- GETCODI ----------------------------------------------------------------------
;
;       Busca el codi associat a la tecla indicada per n_Tecla a la EEPROM. Les posicions
;       son 2*nTecla (DataH) i la seguent (DataL)
;
;-------------------------------------------------------------------------------

_GetCodi        bcf     STATUS,C
                rlf     n_Tecla,W       ;(rlf n_Tecla <=> 2*n_Tecla)
                movwf   n_Adr           ;Llegeix DataH
                call    _GetByte
                movfw   n_Data
                movwf   n_DataH

                incf    n_Adr,F         ;Llegeix DataL a la posició seguent
                call    _GetByte
                movfw   n_Data
                movwf   n_DataL
                return


;----- ENVIACODI --------------------------------------------------------------------
;
```

```
;       Afegeix al codi a enviar (16+16 bits) un bit d'START i el rota per enviar START,
;       el codi (començant pel bit de mes pes) i una altra vegada el bit d'START.
;
;-------------------------------------------------------------------------------------

_EnviaCodi      movlw   0x11
                movwf   n_Bit
                rrf     n_DataH, F      ;Podem guardar el bit DataL.0 a C perquè ningu ens
                rrf     n_DataL, F      ;el modifica abans de fer la primera rotació
                bsf     n_DataH, 7

                movlw   kCiclesBurst    ;Carreguem el nombre de cicles de burst
                movwf   cntCicles
                ;----------------------------- Repetim per a cada bit ----------------
_Env1           nop                     ;----- Enviem primer semibit
                nop
                nop
                nop
                nop
_Env2           btfss   n_DataH,7       ;Nomes activem el led si el bit és 0
                bsf     LED
                bcf     LED
                decfsz  cntCicles,F
                goto    _Env1



                rlf     n_DataL,F       ;Rotem byte baix del codi ara. Mes endavant no
                                        ;tindrem temps.C passa a guardar el bit DataL.
                movlw   kCiclesBurst
                movwf   cntCicles
                nop                     ;Per cuadrar temps
                goto    _Env4

_Env3           nop                     ;----- Enviem segon semibit
                nop
                nop
                nop
                nop
_Env4           btfsc   n_DataH,7       ;Nomes activem el led si el bit és 1
                bsf     LED
                bcf     LED
                decfsz  cntCicles,F
                goto    _Env3
                ;---------------------------- Fi enviament d'un bit -----------------
                rlf     n_DataH,F       ;Rotem byte alt del codi. C conté bit 7 de n_DataL

                movlw   kCiclesBurst    ;Carreguem el nombre de cicles del burst
                movwf   cntCicles
                decfsz  n_Bit,F         ;Hem enviat tots els bits?
                goto    _Env2           ;       No: anem a per el següent
                return                  ;       Si: acabem

;----- LECTURA_TECLAT ---------------------------------------------------------------
;
;       Realitza la lectura del teclat i torna el numero de tecla polsada per W. El
teclat
;       utilitzat ha de ser el següent:
;
;  Files    -> PORTB,4 - PORTB,7 (In)
;  Columnes -> PORTB,0 - PORTB,3 (Out)
;
;                       RB3 RB2 RB1 RB0
;                        |   |   |   |
;                        V   V   V   V
;                      |---|---|---|---|
;               RB7 --<| 1 | 2 | 3 | 4 |
;                      |---|---|---|---|
;               RB6 --<| 5 | 6 | 7 | 8 |
;                      |---|---|---|---|
;               RB5 --<| 9 | 0 | A | B |
;                      |---|---|---|---|
;               RB4 --<| D | E | F | 10|
;                      |---|---|---|---|
;
;-------------------------------------------------------------------------------------
```

```
Lectura_Teclat movlw   0x04
               movwf   n_Fila         ;Carrego comptador de files a llegir
               movlw   0xFE           ;Inicialitzem scan de columnes
               movwf   PORTB
               clrw
               goto    _L2

_L1            bsf     STATUS,C
               rlf     PORTB,F
_L2            call    _Tecla
               andlw   0xFF           ;Per actualitzar flag Z
               btfss   STATUS,Z
               goto    _LFi           ;Acaba si te alguna cosa
               decfsz  n_Fila,F       ;Mira si ha repassat totes les columnes
               goto    _L1
               return

_LFi           andlw   0x0F           ;Elimina flag de tecla apretada
               addwf   n_Fila,W       ;Suma n. columna al codi de fila apretat
               movwf   n_Tecla
               return

_Tecla         btfss   PORTB,7        ;Si hi ha alguna tecla apretada, a W tenim
               movlw   0xF0           ; 0xFn, on amb F indiquem que s'ha apretat
               btfss   PORTB,6        ; una tecla i n depen de la fila de la tecla
               movlw   0xF4
               btfss   PORTB,5
               movlw   0xF8
               btfss   PORTB,4
               movlw   0xFC
               return

;----- FI DEL CODI EXECUTABLE  ------------------------------------------------------------
               nop            ;nops necessaris per decret del manual
               nop

;----- EEPROM PER DEFECTE -----------------------------------------------------------------
               org     0x2100
               ;Versio
               de      0x01,0x00

               org 0x2102
               ;Codis de tecles
               de      0x00,0xFF,0x0F,0x0F,0x03,0x03,0x55,0x55
               de      0xFF,0xFF,0x00,0x01,0x34,0x27,0x55,0xAA
               de      0xFF,0x00,0xF0,0xF0,0xCC,0xCC,0xAA,0xAA
               de      0x00,0x00,0xFF,0xFE,0x73,0x29,0x14,0x11

               org 0x2122
               ; Repeticions enviament
               de      0x03

               org 0x2123
               ;Codi confraria
               de      0x00

               org 0x2124
               ;Confraria
               de      "Provisional         "

               org     0x2138
               ;Data ultima modificacio
               de      0x0A, 0x06, 0x06

               org 0x213B
               ;String
               de      "Programacio EEPROM per defecte.    "
               de      "                                   "

;----- FI PROGRAMA ------------------------------------------------------------------------

       end
```

5.- RECEPTOR

5.1.- DESCRIPCIÓ

5.1.1.- Especificacions

Velocitat sèrie	9600/19200/57600/115200 Nivells RS 232
Buffer recepció	48 codis (96 bytes)
XTAL	20MHz
Alimentació	8.5v a 28v cc o ca
Consum	30 mA en repòs
Tecles	Reset i Test
Jumpers	Ajust velocitat sèrie i Inici automàtic
Leds	Alimentació, On, IR, TX, TX i buffer ple
Ordres	Start, Stop, Test

5.1.2.- Protocol comunicació

L'enviament d'informació entre el receptor i el PC necessita d'un protocol per garantitzar que aquesta està lliure d'errors. La distància entre els dos elements pot arribar a ser bastant llarga (més de 20 metres) i el cable de comunicació en molts casos haurà de passar prop de cablejat elèctric de motors.

La informació a creuar és de tres tipus: ordres, dades i reinici:

Tipus	Dada	Descripció
Ordres	**START**	Habilita la recepció de codis IR
	STOP	Para la recepció de codis IR
	TEST	El receptor envia una trama de text amb el seu estat i una cadena de caràcters 'T'
Dades	**CODI**	Enviament d'un codi rebut
Reinici	**RESET**	Indica que el receptor s'ha resetejat
	PENJAT	Indica que el receptor s'ha reiniciat pel Watchdog

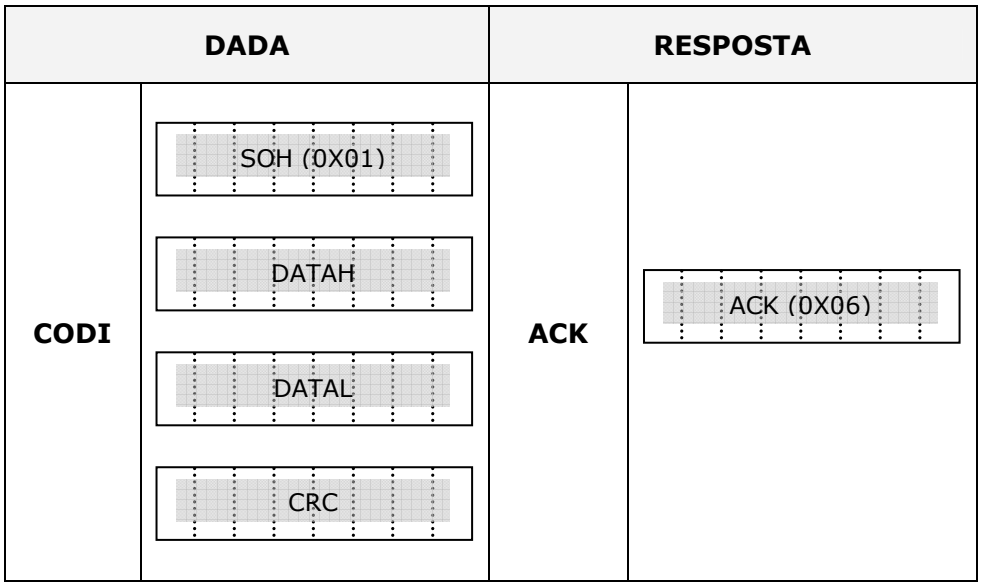
Ordres

Per les ordres s'envien dos bytes. El primer és STX i el segon és START, STOP o TEST segons convingui. Un cop executada l'ordre, el receptor enviarà ACK en els dos primers casos. Si l'ordre enviada és TEST, que consisteix en l'enviament d'una trama d'informació, ja no s'envia ACK per resultar redundant.

ORDRE		**RESPOSTA**	
START	STX (0X02) START (0X0B)	**ACK**	ACK (0X06)
STOP	STX (0X02) STOP (0X0D)	**ACK**	ACK (0X06)
TEST	STX (0X02) TEST (0X54)	**trama ASCII** (>500 bytes)	(ASCII) (ASCII)

Dades

Si el receptor està activat (amb l'ordre Start, per exemple) cada cop que identifiqui un codi l'enviarà immediatament a través del port sèrie. En total s'envien quatre bytes i s'espera un byte ACK des del PC com a confirmació de recepció correcte:



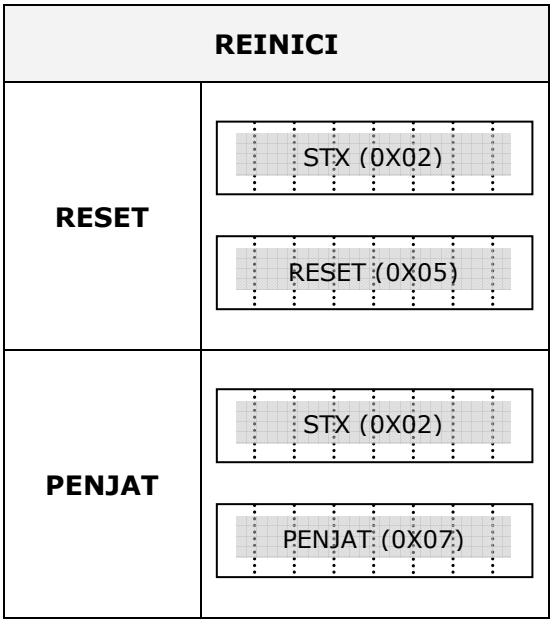
DATAH son els 8 bits de mes pes del codi
DATAL son els 8 bits de menys pes del codi
CRC equival a DATAH XOR DATAL

Si la resposta del PC tarda més de 100ms, es reenvia el mateix codi. Si passats quatre intents no s'ha aconseguit resposta es descarta aquest codi i es passa al següent en cas d'existir.

El receptor té implementat un buffer FIFO amb capacitat de fins a 48 codis per evitar que problemes puntuals amb la comunicació puguin impedir la recepció de nous codis.

Reinici

Quan el receptor s'inicia diferencia si ha estat perquè ha estat connectat a l'alimentació o bé si ha estat perquè ha saltat el timer watchdog per un mal funcionament del microcontrolador. Les dues possibilitats són transmeses via sèrie per poder detectar errors hardware. En aquests casos no s'espera ACK per part del PC.



5.1.3.- ESQUEMA ELÈCTRIC I PCB

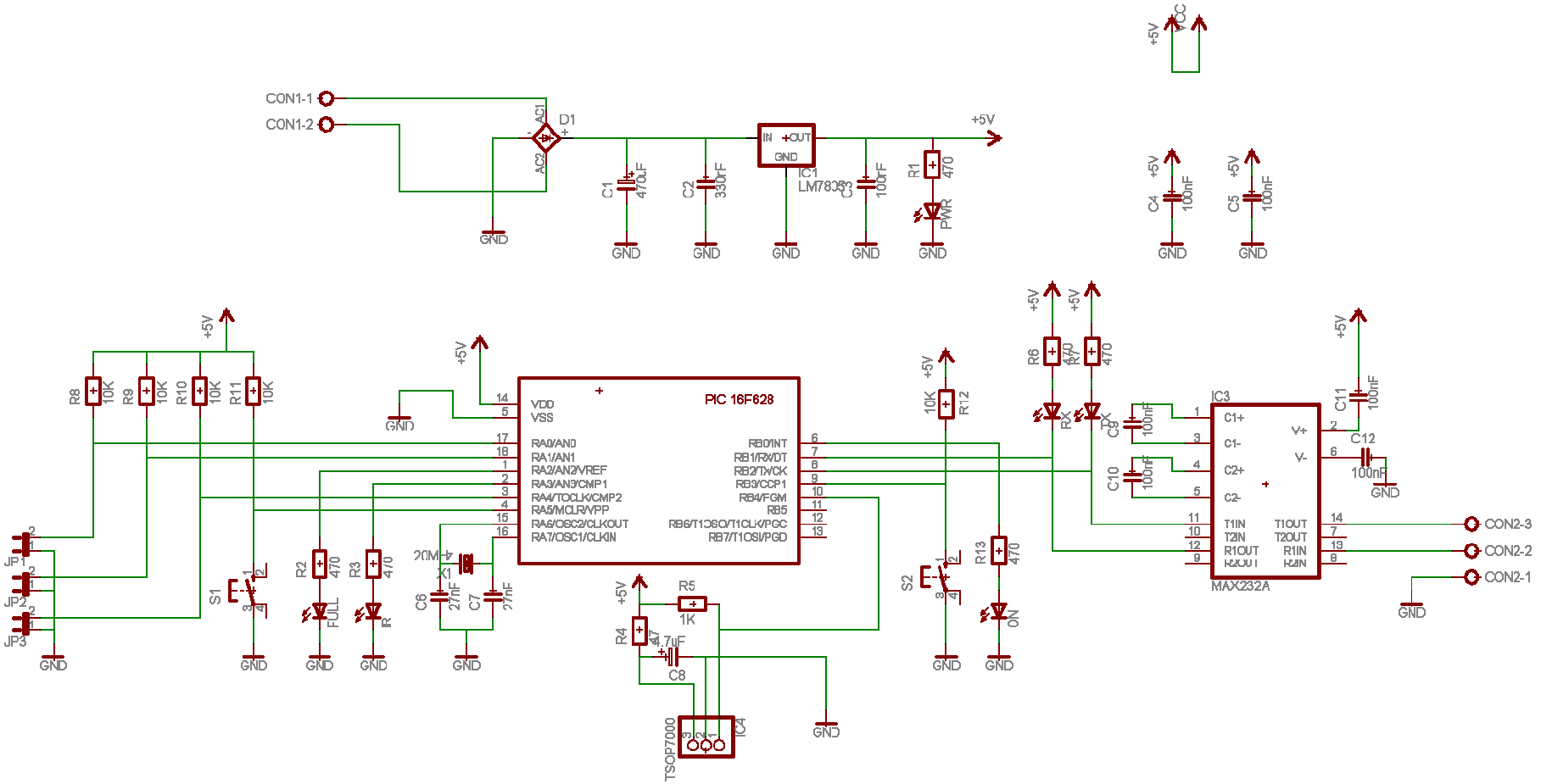


Fig 5.1: Esquema receptor

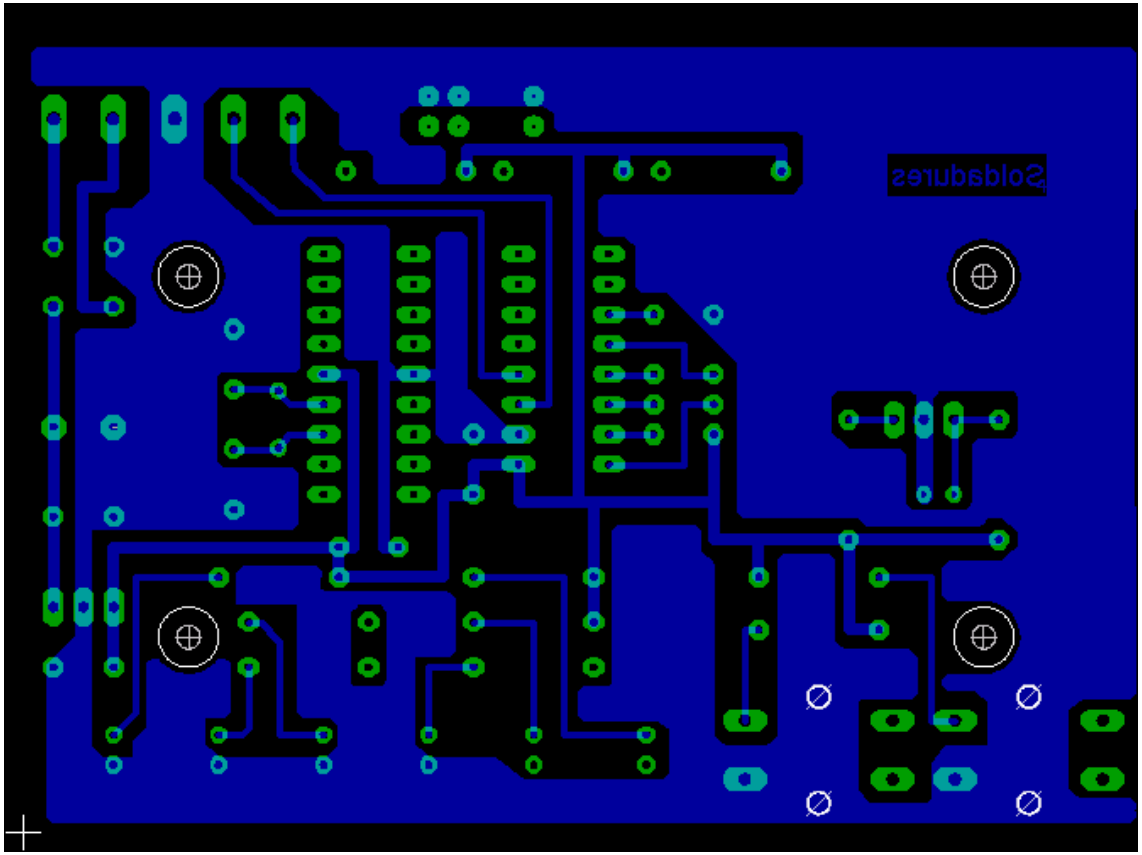


Fig 5.2: Circuit imprès, pistes cara inferior

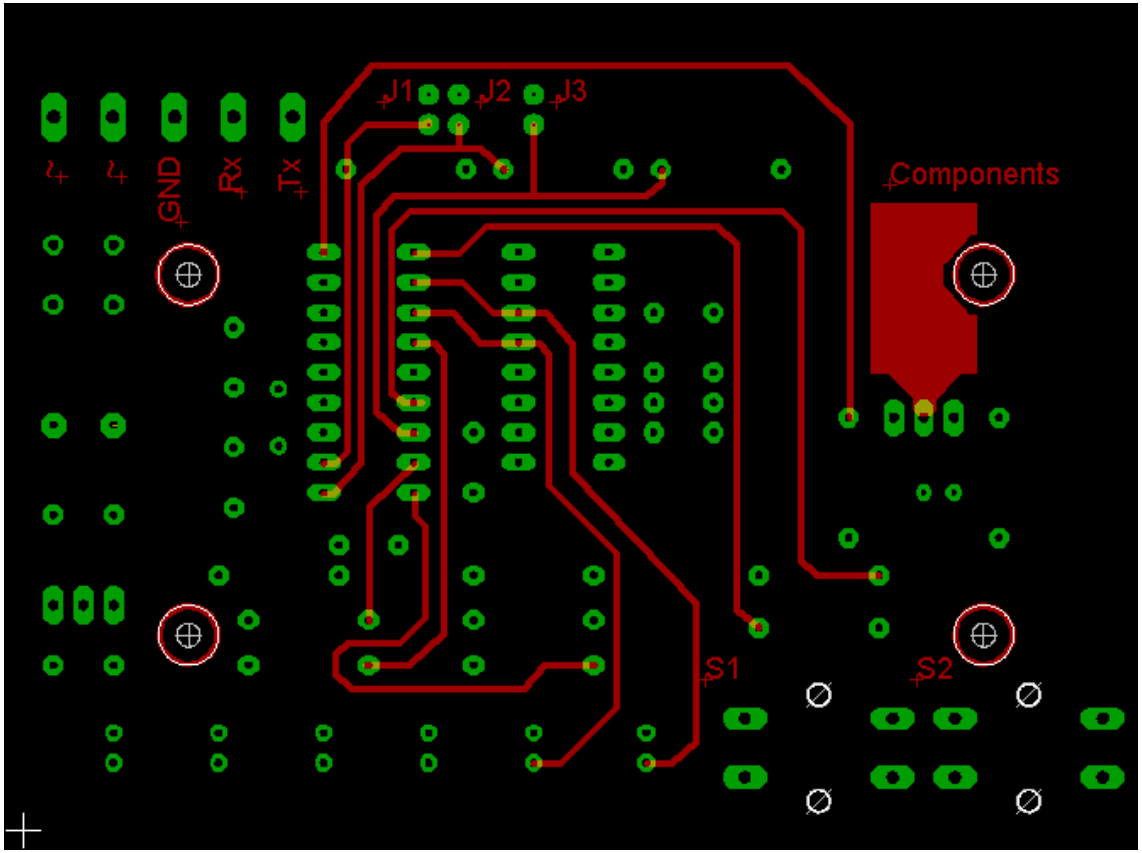


Fig 5.3: Circuit imprès, pistes cara superior

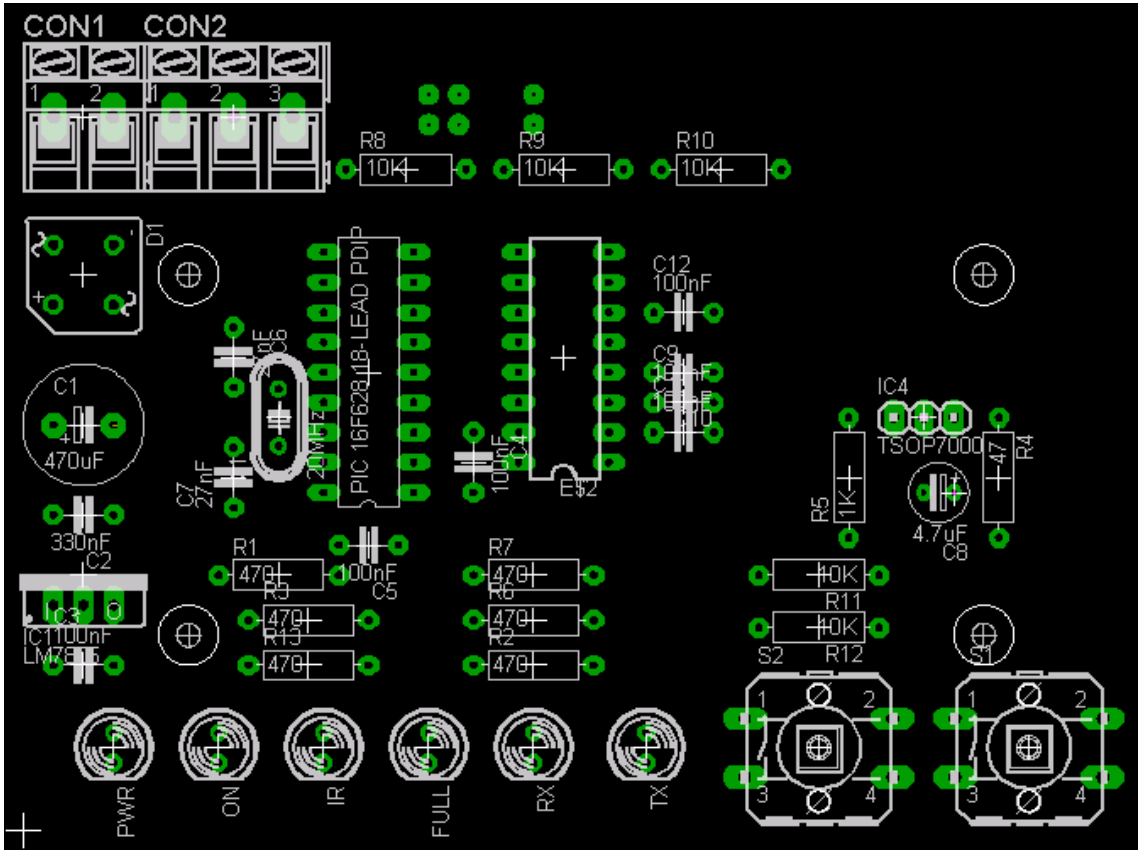


Fig 5.4: Circuit imprès, posició components

5.2.- PARTS

5.2.1.- Alimentació

Per permetre un ampli marge d'utilització s'ha inclòs en el circuit del receptor un conjunt de rectificador, filtre i regulador de tensió. D'aquesta manera es facilita poder alimentar el circuit directament des d'un transformador endollat a corrent de xarxa o des d'un alimentador comercial genèric.

El component principal és un regulador LM7805 que dona una sortida estabilitzada de 5v a partir d'una tensió d'entrada que pot oscil·lar entre els 7 i els 25 volts. Entre l'entrada d'alimentació del circuit i el regulador, però, s'ha disposat un pont de diodes i un condensador per rectificar i filtrar la tensió d'entrada.

Un led, connectat a la sortida del regulador, amb la seva resistència corresponent ens indicarà si el circuit està alimentat.

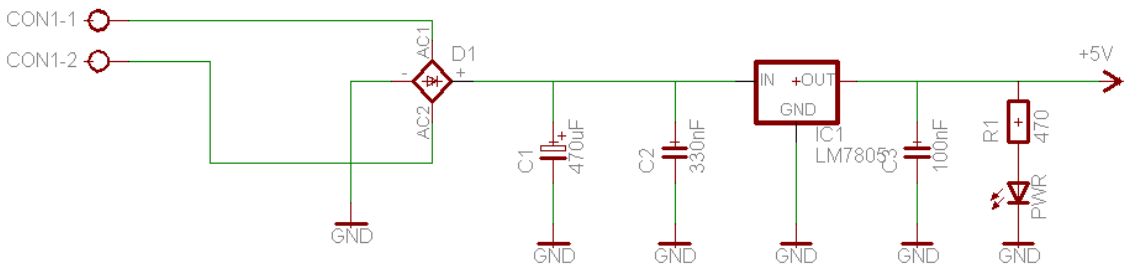


Fig 5.5: Circuit d'alimentació del receptor

5.2.2.- Port sèrie

El port sèrie del microcontrolador funciona amb nivells TTL que s'han de transformar a nivells RS232 per fer possible la connexió amb un PC. La correspondència és:

TTL	RS232
0v	+12v
5v	-12v

Per fer aquesta adaptació de tensions s'utilitza un circuit integrat MAX232 de la casa MAXIM connectat de la següent manera:

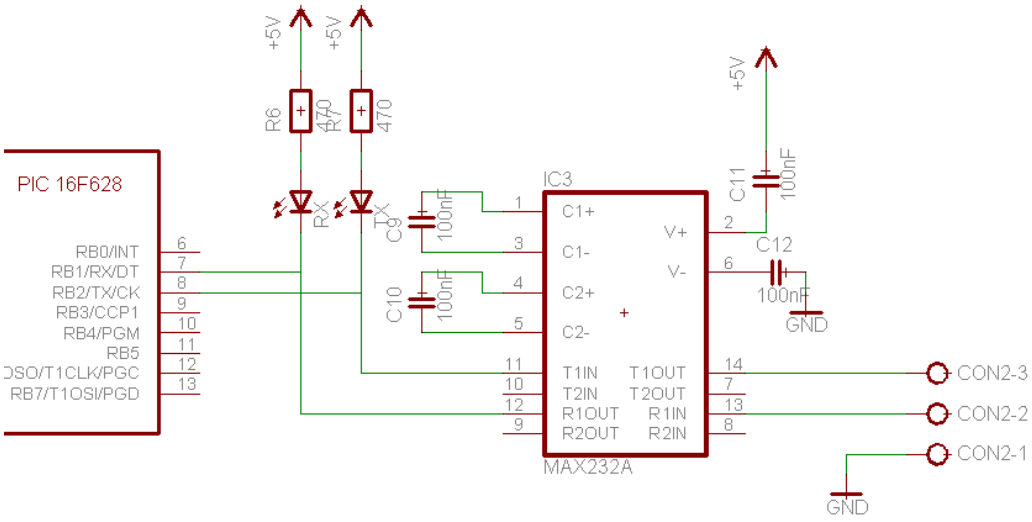


Fig 5.6: Esquema port sèrie

La sortida TX del microcontrolador es connecta a l'entrada T1IN del MAX232, donant per la sortida T1OUT la senyal amb els nivells adequats. El mateix ens trobem en el circuit d'entrada: l'entrada R1IN a nivells RS232 està adaptada a nivells TTL a la sortida R1OUT i es connecta al pint RX del microcontrolador.

En realitzat, aquest integrat genera la senyal a +10v i -10v i, encara que estrictament incorrecte, és suficient per comunicar-se amb la majoria de ports RS232. Els condensadors C9, C10, C11 i C12 són els encarregats de duplicar la tensió i invertir-la aconseguint així tensions de 10 volts a partir dels 5 volts de la tensió d'entrada. Determinades versions del MAX232 (com el MAX232A aqui utilitzat) poden utilitzar condensadors de 0.1uF, d'altres necessiten condensadors de 1uF..

Dos leds amb les seves respectives resistències, indicaran l'activitat dels pins RxD i TxD del microcontrolador, facilitant la detecció d'errors de comunicació.

5.2.3.- Receptor IR

El mòdul receptor IR es connecta seguint les indicacions del seu manual. La seva sortida es connecta a RB.4 del microcontrolador. Aquest pin és un dels quatre que poden generar una interrupció, característica en la que es basa el mètode utilitzat per realitzar l'anàlisi de la trama rebuda.

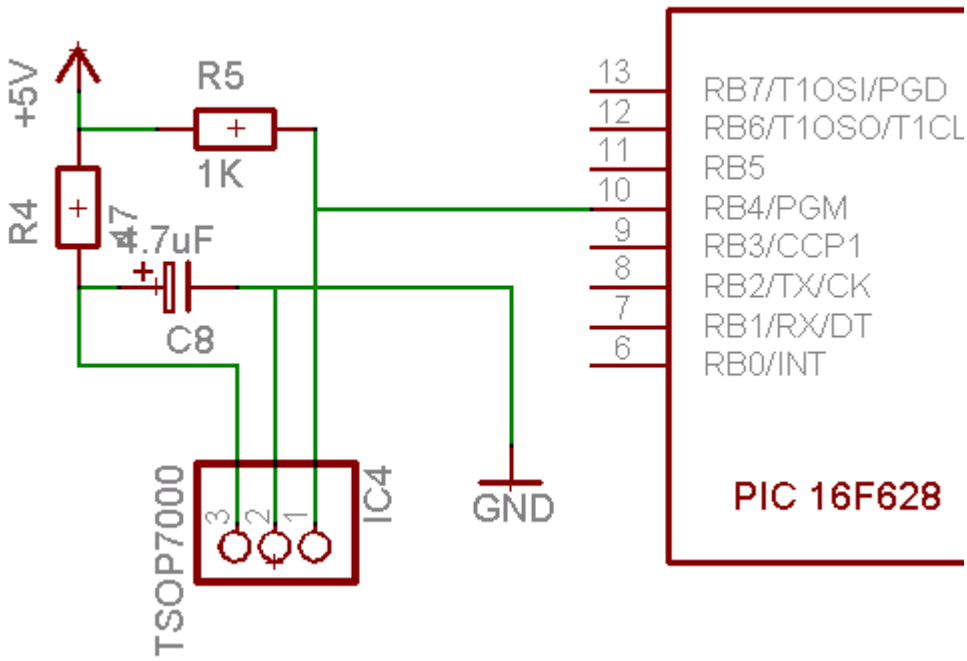


Fig 5.7: Circuit detector IR

5.2.4.- Polsadors i jumpers

Els polsadors i els jumpers estan connectats de la següent manera:

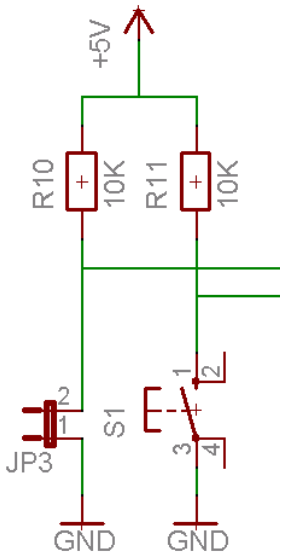


Fig 5.8: Conexionat polsadors i jumpers

Un zero a l'entrada del microcontrolador indica que el jumper o polsador associat està tancat, mentre que un u indica el contrari.

La funció de cada un d'ells queda recollida en la següent taula:

Nom	Funció	Descripció
S1	Reset	Reseteja el microcontrolador
S2	Test	Si es manté polsat durant un reset, s'envia la trama de test pel port sèrie
J1 **J2**	Vel. sèrie	Selecció de la velocitat del port (0: no posat, 1: posat) 00: 9600bps 01: 19200bps 10: 57600bps 11: 115200bps
J3	Mode inici	Si està posat, la recepció IR estarà habilitada sense necessitat de rebre cap ordre.

5.2.5.- Leds

Excepte els leds que indiquen el trànsit en el port sèrie, la restà estan connectats al microcontrolador per l'ànode mitjançant una resistència de 470Ω.

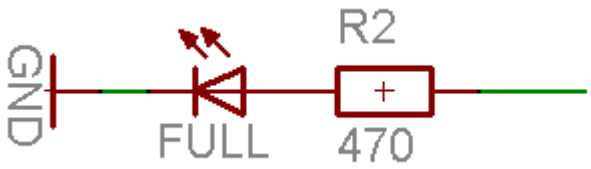


Fig 5.9: Conexionat leds

El significat dels leds és el següent:

Nom	Significat
PWR	El circuit està alimentat
ON	La recepció de codis està habilitada
IR	Detecció de llum IR
FULL	Buffer de codis ple
RX	Trànsit a la linia Rx del port sèrie
TX	Trànsit a la linia Tx del port sèrie

5.3.- CODI

El programa del receptor és clarament més complexe que el del comandament. Les dos tasques bàsiques del receptor són rebre i descodificar trames IR i comunicar-se amb el PC al qual està connectat. S'ha dissenyat de tal manera que aquestes tasques s'executen independentment, de manera que una d'elles no pugui bloquejar l'altra.

En els següents apartats s'explicarà l'estructura del programa amb les parts que el composen i s'analitzarà en detall la rutina INT_PB que és la que fa la descodificació de la trama.

5.3.1.- Estructura del programa

Tal com s'ha dit, s'ha separat la part de recepció i descodificació de la senyal IR de la part de comunicació. A la vegada, la comunicació s'ha separat entre la recepció i la transmissió, podent dividir les tasques a realitzar en tres:

- Recepció/descodificació de trames IR
- Enviament dels codis IR per sèrie
- Recepció d'ordres per sèrie (start, stop i test)

Dos elements fan de pont entre aquestes parts: un buffer FIFO i un algoristme de control. Quan es rep un codi aquest s'emmagatzema en el buffer en espera de que es pugui enviar pel port sèrie. La part que controla la transmissió de codis enviarà aquests codis pel port sèrie quan hi hagi temps. D'aquesta manera, la lentitud del port sèrie i les esperes de resposta no perjudiquen la recepció de trames. L'algoristme de control consulta les ordres rebudes i les executa habilitant (start) o deshabilitant (stop) la lectura IR, o realitzant el test de comunicacions (test).

El següent diagrama mostra la relació entre les tasques realitzades i els elements de què disposen:

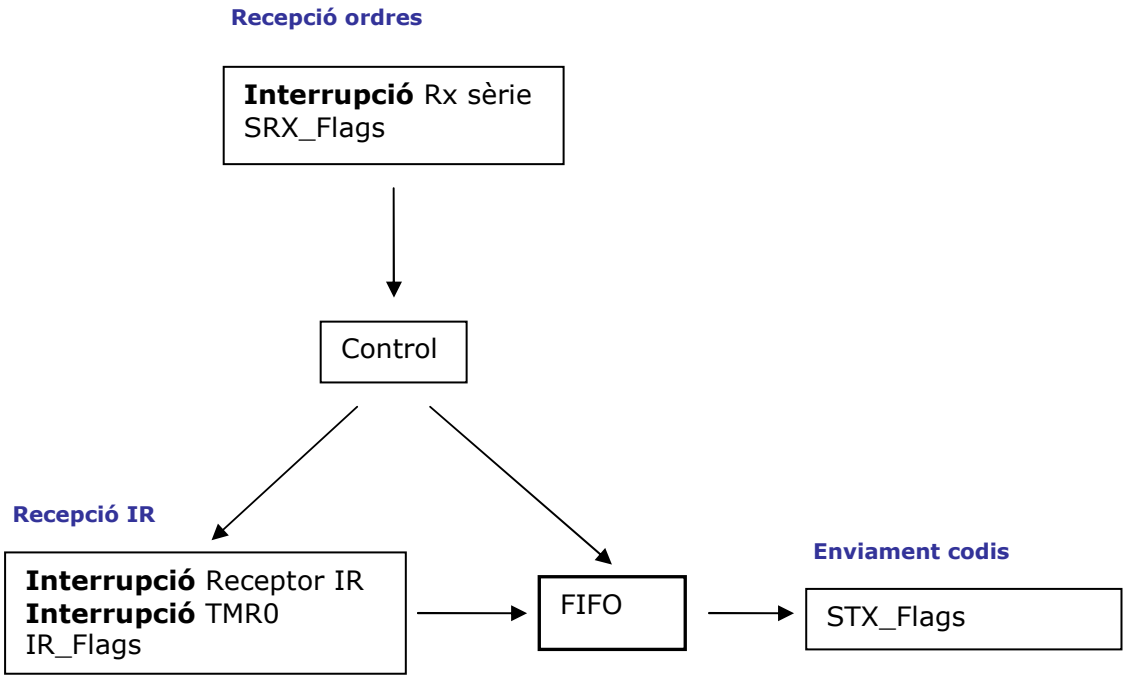


Fig 5.10: Relació entre tasques

SRX_Flags, STX_Flags i IR_Flags estan implementats en un byte cada un i contenen els flags utilitzats per cada part. Per exemple: quan s'ha rebut l'ordre d'START pel sèrie, s'activa el flag fSTART. Aquest flag provoca que la rutina _Control activi la interrupció del receptor IR, que s'activa a cada flanc del senyal provinent del detector. La ISR d'aquesta interrupció utilitza els flags continguts a IR_Flags (i també el TMR0) per controlar el procés de la lectura de la trama. Si salta la interrupció de TMR0, és la seva ISR qui, segons els flags de IR_Flags considera si la trama ja s'ha rebut correctament i la guarda a la FIFO. Posteriorment un procès a part enviarà aquest codi ajudant-se dels flags de STX_Flags. Per la seva complexitat, la descodificació de trames es tractarà en detall en l'apartat 5.3.3.

Les tasques segueixen el següent ordre de prioritats:

- Recepció IR
- Recepció ordres
- Enviament codis i Control

Les dues primeres tasques s'executen per interrupció, pel que la seva prioritat ve marcada per la rutina que consulta quin dispositiu ha provocat l'interrupció. Així primer es consulta si l'origen ha estat un flanc del detector IR, després es consulta si ha estat el TMR0 i, finalment, es comprova si ha saltat la interrupció del port sèrie. Quan es detecta l'origen, s'executa la seva ISR i, posteriorment, es consulta la resta de fonts d'interrupció.

L'enviament de codis i el control es gestionen sense utilitzar cap interrupció ni cap crida que en provingui. S'executen sempre que no s'estigui atenent una interrupció.

Així doncs, el programa final és un bucle infinit que comprova contínuament si s'ha rebut alguna ordre o algun codi IR consultant SRX_Flags i el buffer respectivament. La recepció de trames o d'ordres pararan aquests processos.

Aquest sistema facilita la utilització del timer Watchdog. Totes les rutines (excepte _TEST) tenen un temps d'execució menor de 10 µs, quan el temps nominal del watchdog és de 18ms. La seva recàrrega es fa en dos punts: enmig del bucle infinit comentat al pàrraf anterior i durant l'execució de _TEST, que envia una gran informació pel port sèrie i tarda un mínim de 50ms en acabar.

5.3.2.- Rutines

Al igual que en el cas del comandament, l'existència de les rutines és per ordenar el codi. Cap d'elles es crida des de més d'un lloc diferent.

La següent taula mostra les rutines existents, la seva descripció i a quina de les tasques pertanyen:

Nom	**Tasca**	**Descripció**
_Config		Configura el microcontrolador i els seus perifèrics.
_Main		Bucle infinit. Crida _Control, _TX i recarrega el watchdog timer.
_Control	control	Executa les comandes marcades a SRX_Flags.
_START	control	Habilita la recepció IR.
_STOP	control	Para la recepció IR i buida el buffer.
_TEST	control	Para la recepció IR i envia la trama de text de test pel port sèrie.
_TX	Env. Codis	Si el buffer conté codis, els envia pel sèrie.
_Desencua	Env. Codis	Extreu un codi (dos bytes) del buffer.
_Interrupt		Rutina ISR principal. Determina l'origen i crida la rutina associada.
INT_PB	Rec. IR	ISR del receptor IR. S'executa a cada flanc del senyal. Realitza la descodificació de la trama
INT_T0	Rec. IR	ISR de TMR0. Si s'habia comletat la lectura d'un codi IR el guarda al buffer. En cas contrari reseteja flags IR_Flags
_Encua	Rec. IR	Guarda un codi (dos bytes) al buffer.
INT_RX	Rec. Ordres	ISR recepció sèrie. Identifica les ordres rebudes i les marca a SRX_Flags

5.3.3.- Descodificació del codi IR

La lectura i descodificació de la trama IR es realitza mesurant el temps existent entre cada flanc del senyal del detector. Cada cop que es produeix un d'aquests flancs, tant ascendents com descendents, s'executa la ISR INT_PB. Aquesta s'ajuda del TMR0 per fer la mesura efectiva dels temps. A partir d'aquestes mesures i del sentit dels flancs es verifica que el codi compleix les propietats explicades a l'apartat 3.3.3 per determinar l'existència d'errors:

- Contenir únicament pulsos i espais de 30.8µs o 61.6µs.
- Llargada total de la trama de 1047.2µs
- Existència d'un flanc al centre de cada bit.

Mesura de la llargada d'un espai o d'un pols

Encara que tots els pulsos i espais duren 30.8µs o 61.6µs, la senyal que surt del detector està distorsionada i pot variar aquests temps. Per tant s'utilitzen quatre constants per determinar si un pols és correcte: kTMin1, kTMax1, kTMin2 i kTMax2 que equivalen als tamanys màxim i mínim de polsos d'un semibit (30.8µs) o de dos semibits (61.6µs). Si entre dues crides a la ISR INT_PB el TMR0 ha contat entre kTMin1 i kTMax1 cicles, l'últim pols o cicle ha estat de 30.8µs.

El procedimentper determirar-ho és el següent:

Cada cop que s'entra a la ISR es guarda el valor de TMR0 a IR_BfrTmr i es recarrega amb el valor 0xFF – kMax2. De manera que:

IR_BfrTmr = 0xFF – kTMax2 + nCicles (nCicles = temps mesurat en cicles)

Llavors a aquest valor kTMax2 – kTMin2 :

IR_BfrTmr = IR_BfrTmr + (kTMax2 – kTMin2)

I tindrem:

IR_BfrTmr = 0xFF – kTMax2 + nCicles + (kTMax2 – kTMin2)
IR_BfrTmr = 0xFF + nCicles – kTMin2

Si IR_Bfr_Tmr és major de 0xFF (s'ha desbordat), el pols mesurat és major de KTMin2, ja que:

0xFF + nCicles – kTMin2 > 0xFF
nCicles > kTMin2

Si no s'ha desbordat, repetim el procés afegint ara kTMin2- kTMax1 a IR_BfrTmr de manera que valdrà:

IR_BfrTmr = 0xFF + nCicles – kTMin2 + (kTMin2 – kTMax1)
IR_BfrTmr = 0xFF + nCicles - kTMax1

Si el resultat és major de 0xFF, nCicles està entre kTMin2 i kTMax1:

0xFF + nCicles - kTmax1 > 0xFF
nCicles > kTmax1

Si encara no s'ha desbordat, sumem kTMax1 – kTMin1

IR_BfrTmr = 0xFF + nCicles - kTMax1 + (kTMax1 – kTMin1)
IR_BfrTmr = 0xFF + nCicles - kTMin1

I comprovem novament si el valor resultant és major de 0xFF

IR_BfrTmr = 0xFF + nCicles - kTMin1 > 0xFF
nCicles > kTMin1

Si no és aixi el valor mesurat és menor de kTMin1

Aquest mètode es tradueix en només onze instruccions ensamblador, situades a la secció _PB_Lectura de la ISR _INT_PB:

```
_PB_Lectura
       movfw  IR_BfrTmr
       addlw  kTMax2 - kTMin2
       btfsc  STATUS,C
       goto   _PB_A2              ;El pols medeix 2 semibits
       addlw  kTMin2 - kTMax1
       btfsc  STATUS,C
       goto   _PB_Error           ;Error per dubtos
       addlw  kTMax1 - kTMin1
       btfsc  STATUS,C
       goto   _PB_A1              ;El pols medeix 1 semibits
       goto   _PB_Error           ;Error per massa petit
```

El TMR0

Quan s'inicia la lectura d'una trama s'activa la interrupció del TMR0. Si el TMR0 es desborda vol dir que han passat més de kTMax2 cicles des de l'últim flanc. Si l'últim flanc rebut no era l'últim de la trama vol dir que podem descartar la trama actual, ja s'havia d'haver detectat un nou flanc. Si, pel contrari, ja havíem rebut l'últim flanc, vol dir que podem guardar el codi al buffer. D'aquesta manera, cada cop que es rep un codi, es realitza una espera de 60.8µs durant la qual, si arriba un flanc, la trama serà descartada dins la ISR del detector. El motiu de realitzar aquests controls és evitar solapaments de trames diferents.

```
INT_T0
       btfsc  IR_Flags,fComplert
       call   _Encua       ;Encua si s'ha completat el codi
       clrf   IR_Flags     ;Reset dels flags
       bcf    INTCON,T0IE  ;Desactiva INT TMR0
       bcf    O_Curring    ;Desactivem led IR
       return
```

Descodificació

Recordem que la trama IR està composada de 32 bits i cada un d'ells està format per dos semibits. Al inici s'afegeix un bit d'start, però podem considerar que únicament s'afegeix un semibit (el primer semibit 0 no el tenim en compte).

Podem considerar que un flanc pot estar en una posició parell o senar, a on els que estan en posicions parell són els que indiquen el valor del bit (ascendents: 1, descendents: 0) i la resta no aporten informació. Una trama correcte contindrà com a mínim un flanc en totes les posicions parells.

L'algorisme de descodificació, a partir de la mesura dels pulsos, compta aquests semibits amb el registre IR_LongTrLec i controla si el flanc detectat ha estat senar o parell. D'aquesta manera pot arribar a les següents conclusions:

	Espai entre ultims flancs	**Flanc Parell/Senar**	**Conclusió**
A	1 semibit	Parell	Bit informació
B	2 semibit	Parell	Bit informació
C	1 semibit	Senar	Ignorem
D	2 semibit	Senar	Error

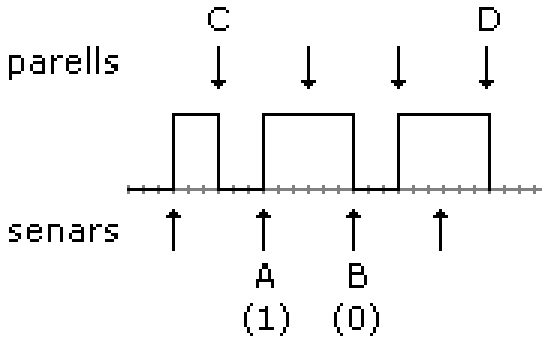


Fig 5.11: Significat flancs

La part que mesura l'espai entre els flancs salta a les etiquetes _PB_A1 o _PB_A2 segons si s'ha mesurat un espai d'un o dos semibits respectivament. És en aquestes a on es fa el control de la trama, es manté el comptador IR_LongTrLec i es guarda el codi rebut:

```
_PB_A1
       incf   IR_LongTrLec,F
       btfss  IR_LongTrLec,0      ;Mirem si es parell
       goto   _PB_Data            ;      Si: tenim un bit del codi
       btfsc  IR_Flags, fWait     ;      No: estavem esperant final ultim 1?
       goto   _PB_Complert        ;            Si, ja el tenim
       return                     ;            No, tornem

_PB_A2
       incf   IR_LongTrLec, F
       incf   IR_LongTrLec, F
       btfsc  IR_LongTrLec, 0     ;Mirem si es parell
       goto   _PB_Error           ;      No: Tenim error
                                  ;      Si: Tenim un bit del codi
_PB_Data
       rlf    IR_DataL,F          ;Guardem el bit
       rlf    IR_DataH,F
       bcf    IR_DataL,0
       btfsc  IR_Flags, fData
       bsf    IR_DataL,0

       movfw  IR_LongTrLec        ;Mirem si portem 32 semi-bits llegits
       sublw  0x20
       btfss  STATUS,C
       goto   _PB_Error           ;   IR_LongTrLec > 32
       btfss  STATUS,Z
       return                     ;   IR_LongTrLec < 32
       btfss  IR_Flags, fData     ;   IR_LongTrLec = 32. Ultim bit. És un:
       goto   _PB_Complert        ;      0: Ja tenim el codi complert
       bsf    IR_Flags, fWait     ;      1: Haurem d'esperar el flanc final
       return
```

A _PB_A1 es diferencia si hem rebut l'últim bit del codi i aquest és 1. És un cas específic, ja que tenim completat el codi però s'ha d'esperar a rebre encara un flanc de baixada.

Veiem com a _PB_DATA es guarda el codi a mesura que es va rebent i com, posteriorment, es comprova si ja està tot rebut consultant el valor de IR_TrLongRec.

El següent gràfic mostra el funcionament complert de _INT_PB, útil per seguir el llistat complert de la rutina:

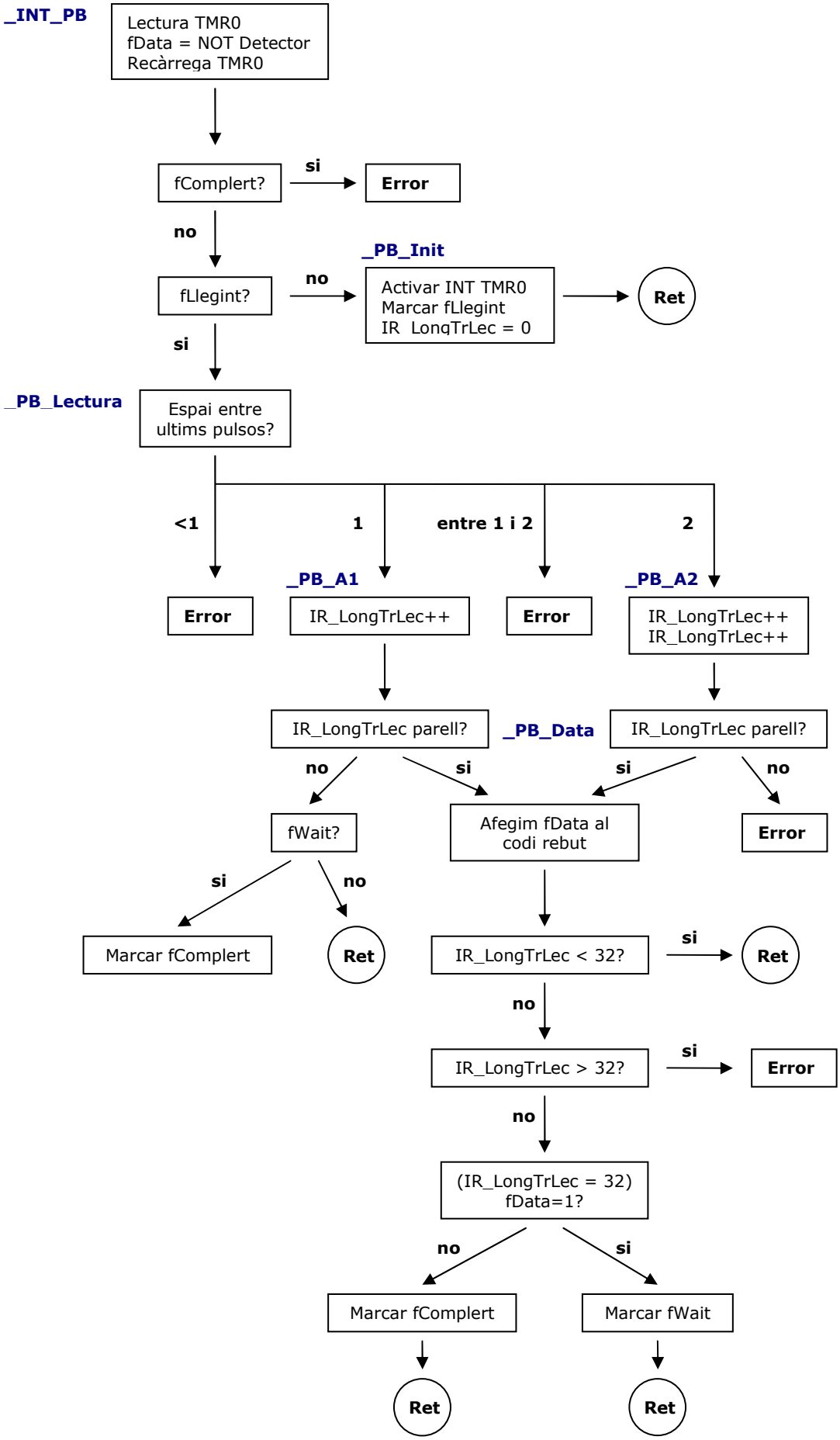


Fig 5.12: Estructura programa receptor

5.3.4.- Llistat RECEPTOR.ASM

```
        List p=16F627a
        __config 0x2126       ;Configuracio: Memoria no protegida. Habilita WDT i PWRT
        include "P16F627a.INC"


;------ DEFINICIO DE VARIABLES I CONSTANTS ----------------------------------------------
;
;--------------------------------------------------------------------------------------

;----- CONSTANTS DE TEMPS ------

;Interval en cicles/2 de 0.5 bit (teoric=0x4D; 14x2.2uS = 30.8uS)
#define kTMin1 0x3D
#define kTMax1 0x5D

;Interval en cicles/s de 1 bit (teoric=0x9A; 28x2.2uS = 61.6uS)
#define kTMin2 0x7A
#define kTMax2 0xBA

;----- CONSTANTS DEL SERIE -----

#define SOH             0x01
#define STX             0x02
#define ACK             0x06
#define NACK            0x15
#define START           0x0B
#define STOP            0x0D
#define TEST            0x54
#define RESET           0x05
#define PENJAT          0x07

;----- I/O ---------------------

#define I_Detector      PORTB,4 ;Detector IR
#define I_Sp0           PORTA,0 ;Jumper velocitat serie
#define I_Sp1           PORTA,1 ;Jumper velocitat serie
#define I_TEST          PORTB,3 ;Jumper test
#define I_STOP          PORTA,4 ;Jumper START/STOP

#define O_FULL          PORTA,2 ;Buffer ple
#define O_Curring       PORTA,3 ;Activitat IR
#define O_ON            PORTB,0 ;Receptor activat

;----- REGISTRES ---------------

;----- Registres lectura IR

IR_DataH        equ     0x20    ;Codi rebut
IR_DataL        equ     0x21
IR_LongTrLec    equ     0x36    ;Nombre de meitat de bits rebuts
IR_BfrTmr       equ     0x35    ;Mesura del temps entre flancs
IR_Flags        equ     0x23    ;Flags recepcio IR

        #define fLlegint        0       ;Esta en plna lectura de trama
        #define fComplert       1       ;Tenim tot el codi llegit
        #define fWait           2       ;Efectua espera anti-solapament
        #define fData           3       ;Buffer lectura detector


;----- Registres rutina serie (TX)

S_DataH         equ     0x24    ;Codi a enviar
S_DataL         equ     0x25
STX_NRep        equ     0x32    ;Vegades que ha enviat el mateix codi
STX_Flags       equ     0x26    ;Flags serie (TX)

        #define fActiu          0       ;Esta enviant trama
        #define fDataH          1       ;Ha enviat DataH
        #define fDataL          2       ;Ha enviat DataL
        #define fCRC            3       ;Ha enviat CRC

;----- Registres rutina serie (RX)

Bfr_RCREG       equ     0x31    ;
SRX_Flags       equ     0x27    ;Flags serie (RX)
```

```
       #define fALGO         0      ;Ha rebut algo important
       #define fSTX          1      ;Ha rebut byte STX
       #define fSTART        2      ;Ha rebut ordre START
       #define fSTOP         3      ;Ha rebut ordre STOP
       #define fTEST         4      ;Ha rebut ordre TEST
       #define fACK          5      ;Ha rebut byte ACK
       #define fNACK         6      ;Ha rebut byte NACK
       #define fCOMORL       7      ;Recepció sense sentit

;----- Registres FIFO

Bfr_NBytes     equ     0x28    ;Numero bytes ocupat
Bfr_Pointer    equ     0x29    ;Pointer primer element

;----- Registres varis

Ord_Flags      equ     0x2A    ;Flags ordre (START o no)
Bk_W           equ     0x2D    ;Backup W
Bk_Status      equ     0x2E    ;Backup STATUS
CGen           equ     0x2F    ;Registre general
Num            equ     0x30    ;Registre general



               org     0x00

                       goto    _Config

               org     0x04


;----- RUTINA INTERRUPCIO ----------------------------------------------------------------
;
;      Salta a la rutina d'atenció d'interrupció que toca. Només per sèrie, PB i TMR0
;      (si està habilitat). Ordenades per prioritat.
;
;-------------------------------------------------------------------------------------------


_Interrupt
               movwf   Bk_W           ;Guarda W i STATUS
               swapf   STATUS,W       ;(swap no afecta C i Z)
               movwf   Bk_Status

_IntPB         btfsc   INTCON, RBIF   ;Ha saltat el PortB?
               call    INT_PB         ;      Si

_IntT0         btfss   INTCON, T0IE   ;Si int T0 esta deshabilitada comprova el serie
               goto    _IntRX
               btfsc   INTCON, T0IF   ;Ha petat T0 (IR)?
               call    INT_T0         ;      Si

_IntRX         btfsc   PIR1, RCIF     ;Ha saltat el serie?
               call    INT_RX         ;      Si

               swapf   Bk_Status,W    ;Recuperem W i STATUS
               movwf   STATUS
               swapf   Bk_W,F         ;(swap no afecta C i Z)
               swapf   Bk_W,W
               retfie                 ;Continuem


;----- OPC -------------------------------------------------------------------------------
;
;      Fa la lectura de la trama. La trama comença amb un bit d'Start seguit de la resta
;      del codi començant pel bit de mes pes.
;
;-------------------------------------------------------------------------------------------

INT_PB         movfw   TMR0           ;Apuntem contingut de TMR0 i el recarreguem
               movwf   IR_BfrTmr
               movlw   0xFF - kTMax2
               movwf   TMR0
               bcf     INTCON, T0IF    ;Reset flag interrupcio TMR0 per si acaba de petar

               bcf     IR_Flags, fData ;fData = NOT Detector
               btfss   I_Detector
               bsf     IR_Flags, fData
```

```
                bcf     INTCON, RBIF    ;Reset flag interrupcio PORTB

                btfsc   IR_Flags, fComplert
                goto    _PB_Error
                btfsc   IR_Flags, fLlegint
                goto    _PB_Lectura

_PB_Init        bsf     O_Curring       ;Activem led IR
                bsf     INTCON, T0IE    ;Activem int TMR0 (ja el tenim inicialitzat)
                btfss   IR_Flags, fData ;Si estavem controlant el gap i hem rebut 0: error
                goto    _PB_Error
                bsf     IR_Flags, fLlegint
                clrf    IR_LongTrLec
                return

_PB_Lectura     movfw   IR_BfrTmr       ;Determinem el temps passat (1T, 2T o indeterminat)
                addlw   kTMax2 - kTMin2
                btfsc   STATUS,C
                goto    _PB_A2          ;El pols medeix 2 semicicles.
                addlw   kTMin2 - kTMax1
                btfsc   STATUS,C
                goto    _PB_Error       ;Error per dubtos
                addlw   kTMax1 - kTMin1
                btfsc   STATUS,C
                goto    _PB_A1          ;El pols medeix 1 semicicle
                goto    _PB_Error       ;Error per massa petit

_PB_A1          incf    IR_LongTrLec,F
                btfss   IR_LongTrLec,0  ;Mirem si es parell
                goto    _PB_Data        ;     Si: tenim un bit del codi
                btfsc   IR_Flags, fWait ;     No: estavem esperant final ultim 1?
                goto    _PB_Complert    ;             Si, ja el tenim
                return                  ;             No, tornem

_PB_A2          incf    IR_LongTrLec, F
                incf    IR_LongTrLec, F
                btfsc   IR_LongTrLec, 0 ;Mirem si es parell
                goto    _PB_Error       ;     No: Tenim error
                                        ;     Si: Tenim un bit del codi
_PB_Data        rlf     IR_DataL,F      ;Guardem el bit
                rlf     IR_DataH,F
                bcf     IR_DataL,0
                btfsc   IR_Flags, fData
                bsf     IR_DataL,0

                movfw   IR_LongTrLec    ;Mirem si portem 32 semi-bits llegits:
                sublw   0x20
                btfss   STATUS,C
                goto    _PB_Error       ;  IR_LongTrLec > 32
                btfss   STATUS,Z
                return                  ;  IR_LongTrLec < 32
                btfss   IR_Flags, fData ;  IR_LongTrLec = 32. Es l'ultim bit i és un:
                goto    _PB_Complert    ;    0: Ja tenim el codi complert
                bsf     IR_Flags, fWait ;    1: Haurem d'esperar el flanc de baixada final
                return

_PB_Complert    bcf     O_Curring             ;Desactivem led IR
                bsf     IR_Flags, fComplert   ;Marquem i marxem
                return

_PB_Error
                bcf     INTCON,T0IE     ;Parem timer
                bcf     O_Curring       ;Desactivem led IR
                clrf    IR_Flags        ;Reset dels flags
                return




;----- T0 ----------------------------------------------------------------------
;
;       Si arriba a desbordar-se el TMR0 per no haver-hi activitat quan ja teniem el codi
;       complert, és que tot ha anat bé i encuem. Si no reset i tanquem.
;
;-------------------------------------------------------------------------------

INT_T0          btfsc   IR_Flags,fComplert    ;Si tenim el codi complert...
```

```
                call    _Encua                  ;...el guardem
                clrf    IR_Flags                ;Reset de flags IR
                bcf     INTCON,T0IE             ;Desactivem INT TMR0
                bcf     O_Curring               ;Desactivem led IR
                return

;----- SERIE INPUT --------------------------------------------------------------
;
;       Segons el que arriba pel port serie, activa el flag corresponent (el reset l'ha
;    de fer qui li toqui). Si es una ordre, marca el flag fAlgo apart del correponent
;    de l'ordre.
;
;-------------------------------------------------------------------------------

INT_RX          movfw   RCSTA
                andlw   0x06
                btfsc   STATUS,Z        ;Consulta FERR i OERR
                goto    _NoError        ;     Tots a zero, continua
                movfw   RCREG
                bcf     RCSTA, CREN     ;En cas d'error reseteja flags d'error i torna
                bsf     RCSTA, CREN
                return

_NoError        movfw   RCREG           ;RCREG nomes es pot llegir una vegada...
                movwf   Bfr_RCREG
                btfsc   SRX_Flags, fSTX
                goto    _TEST?
                                        ;-------------- NO TENIEM STX ---------------

                sublw   STX             ;Es STX?
                btfss   STATUS,Z
                goto    _ACK?           ;   No
                bsf     SRX_Flags, fSTX  ;   Si, marca i surt
                return

_ACK?           movfw   Bfr_RCREG
                sublw   ACK             ;Es ACK?
                btfss   STATUS,Z
                goto    _NACK?          ;   No
                bsf     SRX_Flags, fACK  ;   Si, marca i surt
                return

_NACK?          movfw   Bfr_RCREG
                sublw   NACK            ;Es NACK?
                btfss   STATUS,Z
                goto    _COMORL         ;  No, no te sentit
                bsf     SRX_Flags, fNACK ;  Si, marca i fuig
                return
                                        ;-------------- JA TENIEM STX ---------------
_TEST?          sublw   TEST
                btfss   STATUS,Z        ;Sera TEST?
                goto    _START?         ;   No
                bsf     SRX_Flags, fTEST ;  Si
                bsf     SRX_Flags, fALGO
                return

_START?         movfw   Bfr_RCREG
                sublw   START
                btfss   STATUS,Z        ;Sera START?
                goto    _STOP?           ;   No
                bsf     SRX_Flags, fSTART ;   Si
                bsf     SRX_Flags, fALGO
                return

_STOP?          movfw   Bfr_RCREG
                sublw   STOP
                btfss   STATUS,Z        ;Sera STOP?
                goto    _COMORL         ;   No
                bsf     SRX_Flags, fSTOP  ;   Si
                bsf     SRX_Flags, fALGO
                return

_COMORL         bsf     SRX_Flags, fCOMORL ;Marquem incorrecte
                bsf     SRX_Flags, fALGO
                return
```

```
;----- TAULA_SERIE ------------------------------------------------------------------
;
;      Retorna els valors a carregar a SPBGR per aconseguir cada una de les velocitats
;       de serie seleccionables per jumpers.
;-------------------------------------------------------------------------------------

Taula_Serie
               addwf   PCL,F
               retlw   0x0A            ;115200
               retlw   0x15            ;57600
               retlw   0x40            ;19200
               retlw   0x81            ;9600

;----- STRINGS A MOSTRAR ------------------------------------------------------------
;
;      Contenen el text a mostrar quan fem el test. Condició: acabar en \0 i no
;   excedir els 255 caràcters (facil)
;
;-------------------------------------------------------------------------------------

_StrGen
               addwf   PCL,F                       ;Text que sempre es mostra
               dt      "\r\nReceptor IR 1.0\r\n"
               dt      "10/06/2006\r\n\0"          ;Conté la data

_StrOn
               addwf   PCL,F
               dt      "Estat: ON\r\n\0"           ;En cas que el micro estigui ON

_StrOff
               addwf   PCL,F
               dt      "Estat: OFF\r\n\0"          ;Si esta OFF

_Str9600
               addwf   PCL,F
               dt      "Velocitat serie: 9600\r\n\r\n\0"    ;Amb el serie a 9600

_Str19200
               addwf   PCL,F
               dt      "Velocitat serie: 19200\r\n\r\n\0"   ;Idem a 19200

_Str57600
               addwf   PCL,F
               dt      "Velocitat serie: 57600\r\n\r\n\0"   ;I a 57600

_Str115200
               addwf   PCL,F
               dt      "Velocitat serie: 115200\r\n\r\n\0"  ;I a 115200

;----- CONFIG -----------------------------------------------------------------------
;
;      Tot el seguit d'inicialitzacions que cal fer: ports de sortida, port serie,
;    deshabilitzar comparadors, interrupcions i inicialitzar registres.
;
;-------------------------------------------------------------------------------------

_Config        bsf     STATUS,RP0      ;Selecciona bank1

               movlw   0x80            ;Pre-escaler de 2 a TMR0 i de 1 a WDT. NO pull-ups
               movwf   OPTION_REG      ;WDT: nominal de 18mS (amb min 7mS i max 33mS!!!)

               movlw   0x33            ;Configura els ports (0: output; 1: input)
               movwf   TRISA           ;      PortA= 00110011
               movlw   0x1A
               movwf   TRISB           ;      PortB= 00011010

               movlw   0x24
               movwf   TXSTA           ;Serie: Assincron, 8bits HighSpeed, TX habilitat

               bsf     PIE1, RCIE      ;Habilita int recepcio serie

               bcf     STATUS,RP0      ;Selecciona bank0

               bsf     RCSTA,SPEN      ;Ports=Port Serie
               bsf     RCSTA,CREN      ;Habilita recepcio serie

               movlw   0x07
               movwf   CMCON           ;Desactiva els comparadors
```

```
                movlw   0xC0            ;Habilita interrupcions
                movwf   INTCON

                movlw   0x30            ;TMR1: intern preescaler a 8, stop
                movwf   T1CON           ;L'utilitzarem per controlar temps del port sèrie

                movfw   PORTA
                andlw   0x03            ;Lectura de la velocitat del serie seleccionada
                call    Taula_Serie     ;Busca valor SPBRG a carregar
                bsf     STATUS,RP0      ;Selecciona Bank1
                movwf   SPBRG           ;Carrega SPBRG
                bcf     STATUS,RP0      ;Selecciona Bank0

                clrf    Ord_Flags       ;Marquem que no tenim cap ordre
                clrf    SRX_Flags       ;Reset dels flags de gestió del port sèrie
                clrf    STX_Flags

                movlw   0xA0            ;Inicialitzam FIFO circular. Ocupa de 0xA0 a 0xFF...
                movwf   Bfr_Pointer     ;...Vigilar: 0xF0 a 0xFF es solapen amb 0x70 a 0x7F
                clrf    Bfr_NBytes

                movlw   STX             ;Envia STX
                movwf   TXREG

                movlw   RESET           ;Suposem Reset normal
                btfss   STATUS, NOT_TO ;Envia RESET o PENJAT segons el cas
                movlw   PENJAT          ;Ui! Ha petat el WDT. Ho comuniquem
                movwf   TXREG

                clrf    PORTA           ;Apaguem les llums
                clrf    PORTB

                bsf     STATUS, RP0
_cfg1           btfss   TXSTA, TRMT     ;Espera a enviar byte. Cridar abans _STOP el mataria
                goto    _cfg1
                bcf     STATUS, RP0

                call    _STOP           ;Ara si: START o STOP segons jumper
                btfss   I_STOP
                call    _START

                btfss   I_TEST          ;Enviem trama de test si el botó està apretat.
                call    _TEST

;----- MAIN ------------------------------------------------------------------------------
;
;       Rutina principal. És un bucle infinit molt senzill
;
;-----------------------------------------------------------------------------------------

_Main           call    _Control        ;Executa les ordres rebudes
                call    _TX             ;Gestiona l'enviament de codis pel sèrie
                clrwdt                  ;Reset WDT.
                goto    _Main

;----- CONTROL ---------------------------------------------------------------------------
;
;       Segons lo que s'ha arribat pel serie, crida la funcio pertinent i envia
;    ACK o NACK segons convingui. Elimina tots els flags del port serie (RX)
;
;-----------------------------------------------------------------------------------------

_Control        btfss   SRX_Flags, fALGO        ;Si no hi ha res, marxa
                return
                btfsc   SRX_Flags, fSTART       ;Identifica quin flag s'ha activat
                goto    _CnStart
                btfsc   SRX_Flags, fSTOP
                goto    _CnStop
                btfsc   SRX_Flags, fTEST
                goto    _CnTest
                btfss   SRX_Flags, fCOMORL
                return

_CnComorl       clrf    SRX_Flags
                btfss   PIR1, TXIF              ;Envia un NACK
                goto    _CnComorl
                movlw   NACK
                movwf   TXREG
                return
```

```
_CnStart        call    _START                  ;START i envia ACK
                clrf    SRX_Flags
                goto    _EnvACK                 ;(es un goto)

_CnStop         call    _STOP                   ;STOP i envia ACK
                clrf    SRX_Flags
                goto    _EnvACK

_CnTest         btfsc   Ord_Flags, fSTART       ;STOP, si cal, TEST i envia ACK
                call    _STOP
                call    _TEST
                clrf    SRX_Flags               ;Quan a acaba no torna a arrancar IR
                return

_EnvACK         btfss   PIR1, TXIF              ;S'espera a tenir TXREG lliure
                goto    _EnvACK
                movlw   ACK                     ;I envia l'ACK
                movwf   TXREG
                return

;----- SERIE OUTPUT --------------------------------------------------------------------
;
;       Llegeix els codis de la cua i els envia pel serie amb SOH davant i CRC darrera.
;   Per cada codi enviat espera ACK o NACK, si rep ACK passa al seguent codi a enviar,
;   si rep NACK reenvia el mateix, i si no te resposta en 65mS reenvia el mateix fins
;   a un limit de 4 vegades. Després passarà al següent codi de la cua. Utilitza TMR1.
;
;---------------------------------------------------------------------------------------

_TX             btfss   PIR1, TXIF              ;Si TXREG ple, marxa
                return

                btfsc   STX_Flags, fActiu       ;Si no esta enviant res, mira si hi ha algo
                goto    _DataH                  ;Que estava enviant?

                movfw   Bfr_NBytes              ;Tenim algo per enviar?
                btfsc   STATUS,Z
                return                          ;  No

                bsf     STX_Flags, fActiu       ;  Si
                clrf    STX_NRep                ;Comptador de vegades enviat a zero
                call    _Desencua               ;Agafem DataHL de la FIFO
_SOH            movlw   SOH                     ;Envia SOH
                movwf   TXREG
                return

_DataH          btfsc   STX_Flags, fDataH       ;Ja hem enviat DataH?
                goto    _DataL                  ;  Si
                bsf     STX_Flags, fDataH       ;  No, enviem DataH i marquem
                movfw   S_DataH
                movwf   TXREG
                return

_DataL          btfsc   STX_Flags, fDataL       ;Ja hem enviat DataL?
                goto    _CRC                    ;  Si
                bsf     STX_Flags, fDataL       ;  No, enviem DataL i marquem
                movfw   S_DataL
                movwf   TXREG
                return

_CRC            btfsc   STX_Flags, fCRC         ;Ja hem enviat CRC
                goto    _WaitACK                ;  Si
                bsf     STX_Flags, fCRC         ;  No, marquem
                movfw   S_DataH                 ;Calcula CRC
                xorwf   S_DataL, W
                movwf   TXREG                   ;i l'envia
                bcf     SRX_Flags, fACK
                bcf     SRX_Flags, fNACK
                clrf    TMR1H                   ;Reset TMR1
                clrf    TMR1L
                bcf     PIR1, TMR1IF
                bsf     T1CON, TMR1ON           ;Comencem a temporitzar
                return

_WaitACK        btfsc   SRX_Flags, fACK         ;Ha rebut ACK?
                goto    _AckOk
                btfsc   SRX_Flags, fNACK        ;Ha rebut NACK?
                goto    _NackOk
```

```
                btfss   PIR1, TMR1IF          ;Peta per temps?
                return                        ;Cap de les anteriors: marxa

_NoAnswer
                incf    STX_NRep,F
                btfss   STX_NRep, 2           ;Mira si ha enviat 4 vegades
                goto    _EnvAgain             ;       No: Torna a enviar
                clrf    STX_Flags             ;       Si: Passa al seguent
                goto    _TX


_NackOk         bcf     SRX_Flags, fNACK      ;Ok, reset flag
                bcf     T1CON, TMR1ON         ;Para TMR1
                clrf    STX_NRep              ;Comptador de vegades enviat a zero
_EnvAgain       clrf    STX_Flags             ;Com si no hagues enviat res
                bsf     STX_Flags, fActiu
                goto    _SOH                  ;Torna a enviar

_AckOk          bcf     SRX_Flags, fACK       ;Ok, reset flag ACK
                clrf    STX_Flags             ;Reset flags transmisió
                bcf     T1CON, TMR1ON         ;Parem el temporitzador
                goto    _TX                   ;N'enviem un altre, JA!
                return









;----- ENCUA --------------------------------------------------------------------------
;
;       Encua IR_DataH i IR_DataL a les posicions Pointer + NBytes i posterior. Si
;   cua esta plena no encua i es queda tant ample. Actua sobre el led OK
;
;-------------------------------------------------------------------------------------

_Encua          movlw   0x60            ;Comprova si la cua esta plena (96 bytes = 48 codis)
                subwf   Bfr_NBytes,W
                btfss   STATUS,Z
                goto    _EOK
                bsf     O_FULL          ;Indica amb el led si buffer ple
                return

_EOK            movfw   Bfr_Pointer     ;Busca la posicio a afegir la dada
                addwf   Bfr_NBytes, W   ;       Pos = (Pointer + Nbytes)
                btfsc   STATUS,C
                addlw   0xA0            ;Si sobrepassa 0xFF, suma 0xA0
                movwf   FSR             ;Passa la posicio al apuntador
                movfw   IR_DataH
                movwf   INDF            ;Guarda IR_DataH a @FSR
                incf    Bfr_NBytes,F    ;Ja tenim un byte mes
                incf    FSR,F           ;Passem a seguent posicio
                movfw   IR_DataL
                movwf   INDF            ;Guarda IR_DataL a @FSR
                incf    Bfr_NBytes,F    ;Tenim un altre byte
                return

;----- DESENCUA -----------------------------------------------------------------------
;
;       Desencua S_DataH i S_DataL de les posicions Pointer i posterior. No fa res
;    si NBytes = 0.
;
;-------------------------------------------------------------------------------------

_Desencua       bcf     O_FULL
                movfw   Bfr_NBytes      ;Si no hi ha res encuat plega
                btfsc   STATUS,Z
                return

                bcf     INTCON,GIE      ;Deshabilita interrupcions
                movfw   Bfr_Pointer     ;Desencua per l'apuntador
                movwf   FSR
                movfw   INDF
                movwf   S_DataH         ;Treu DataH
                decf    Bfr_NBytes,F    ;Tenim una dada menys..
```

```
                incf    Bfr_Pointer,F
                incf    FSR,F           ;..i apuntem una pos. mes amunt
                movfw   INDF
                movwf   S_DataL         ;Ja tenim dataL
                decf    Bfr_NBytes,F    ;i un byte menys a la cua
                incf    Bfr_Pointer,F   ;apuntem una pos mes amunt..
                bsf     INTCON,GIE      ;Habilita interrupcions

                btfss   STATUS,Z        ;Si apuntem a pos 0...
                return
                movlw   0xA0            ;...passem a apuntar al inici (0xA0)
                movwf   Bfr_Pointer
                return


;----- STOP --------------------------------------------------------------------------
;
;       Para el detector i borra el buffer
;
;-------------------------------------------------------------------------------------

_STOP           bcf     INTCON, RBIE            ;Deshabilita interrupcio OPC
                bcf     INTCON, T0IE            ;Dehabilita interrupcio T0
                clrf    Bfr_NBytes              ;No hi ha res per enviar
                bcf     O_ON
                bcf     Ord_Flags, fSTART       ;Marca STOP
                clrf    STX_Flags               ;Deixa d'enviar el que tenia
                bsf     STATUS, RP0
                bcf     TXSTA, TXEN             ;Elimina bytes del buffer del sèrie
                bsf     TXSTA, TXEN
                bcf     STATUS, RP0
                return

;----- START -------------------------------------------------------------------------
;
;       Marca IR activat, reseteja flags control recepció i activa interrupció OPC
;
;-------------------------------------------------------------------------------------

_START          clrf    IR_Flags                ;Reseteja flags recepcio IR
                bsf     Ord_Flags, fSTART       ;Marca START
                bcf     INTCON, RBIF            ;Elimina flag OPC
                bsf     INTCON, RBIE            ;Activa OPC
                bsf     O_ON
                return

;----- TEST --------------------------------------------------------------------------
;
;       Envia la trama de test: Si esta activat, configuracio del serie i un total de
;       420 caràcters 'T' (facils de veure i '01010100' en binari)
;
;-------------------------------------------------------------------------------------

_TEST           clrf    CGen            ;Comptador bytes enviats
_Gen            movfw   CGen
                call    _StrGen         ;Busca caracter del string _StrGen
                andlw   0xFF            ;Activa flag Z
                btfsc   STATUS,Z        ;El caracter es '\0'?
                goto    _OnOff          ;   Si: passa a seguent string
                call    _Envia          ;   No: enviem i passem a seguent caracter
                incf    CGen,F          ;Un mes
                goto    _Gen            ;Torna-hi

_OnOff          clrf    CGen            ;Comptador a zero
_Cont           movfw   CGen
                btfss   Ord_Flags,fSTART ;Receptor activat?
                goto    _Off            ;   No: enviem de StrOff
                call    _StrOn          ;   Si: enviem de StrOn
                goto    _Tst1
_Off            call    _StrOff
_Tst1           andlw   0xFF
                btfsc   STATUS,Z        ;El caracter es '\0'?
                goto    _Speed          ;   Si: passem a seguent string
                call    _Envia          ;   No: enviem i passem a seguent caracter
                incf    CGen,F          ;Un mes
                goto    _Cont

_Speed          clrf    CGen            ;Comptador a zero
_Cont2          bsf     STATUS, RP0
```

```
               movfw   SPBRG           ;Llegeix SPBRG per saber la velocitat del serie
               bcf     STATUS, RP0
               movwf   Num
               movfw   CGen            ;Carrega el numero de caracter a buscar a W

               btfss   Num,7           ;Mira la velocitat consultant bit alt de SPBRG
               goto    _19200          ;      9600 -> 0x81 -> 1000 0001 (Mira bit7)
_9600          call    _Str9600        ;      19200 -> 0x40 -> 0100 0000 (Mira bit6)
               goto    _Tst2           ;      57600 -> 0x15 -> 0001 0101 (Mira bit4)
                                       ;     115200 -> 0x0A -> 0000 1010 (Per eliminacio)
_19200         btfss   Num,6
               goto    _57600
               call    _Str19200
               goto    _Tst2

_57600         btfss   Num,4
               goto    _115200
               call    _Str57600
               goto    _Tst2

_115200        call    _Str115200

_Tst2          andlw   0xFF
               btfsc   STATUS,Z        ;El caracter es '\0'?
               goto    _Traca          ;    Si: passem a la traca final
               call    _Envia          ;    No, enviem i passem a seguent caracter
               incf    CGen,F
               goto    _Cont2

_Traca         movlw   0x05            ;Utilitza dos comptadors concatenats per
               movwf   Num             ;arribar a fer 420 iteracions
_Cont4         clrf    CGen
_Cont3         movlw   'T'             ;Envia ASCII 'T'. Es fàcil de veure per telnet
               call    _Envia
               decfsz  CGen,F
               goto    _Cont3
               decfsz  Num,F
               goto    _Cont4
               return

_Envia         clrwdt                  ;Reset WDT
               btfss   PIR1,TXIF
               goto    _Envia
               movwf   TXREG           ;Enviem byte
               return


;----- FINAL DEL CODI ------------------------------------------------------------------
               nop             ;Necessaris segons manual
               nop

               end
```

6.- PROGRAMES PC

Per realitzar proves amb els dos circuits s'han realitzat tres programes per PC en VisualBasic 6.0:

- **Receptor**: Visualitza el trànsit de comunicació entre el PC i el receptor i permet enviar-li ordres. Incorpora eines per facilitar la detecció de recepcions errònies.

- **ProgMaster**: Permet programar la EEPROM dels comandaments.

- **Test_IR**: Utilitat que, conjuntament amb el programa test.asm carregat al receptor, permet rebre la longitud entre pulsos detectada per aquest. S'explica en l'annex 2.

6.1.- Receptor

6.1.1.- Pantalla principal

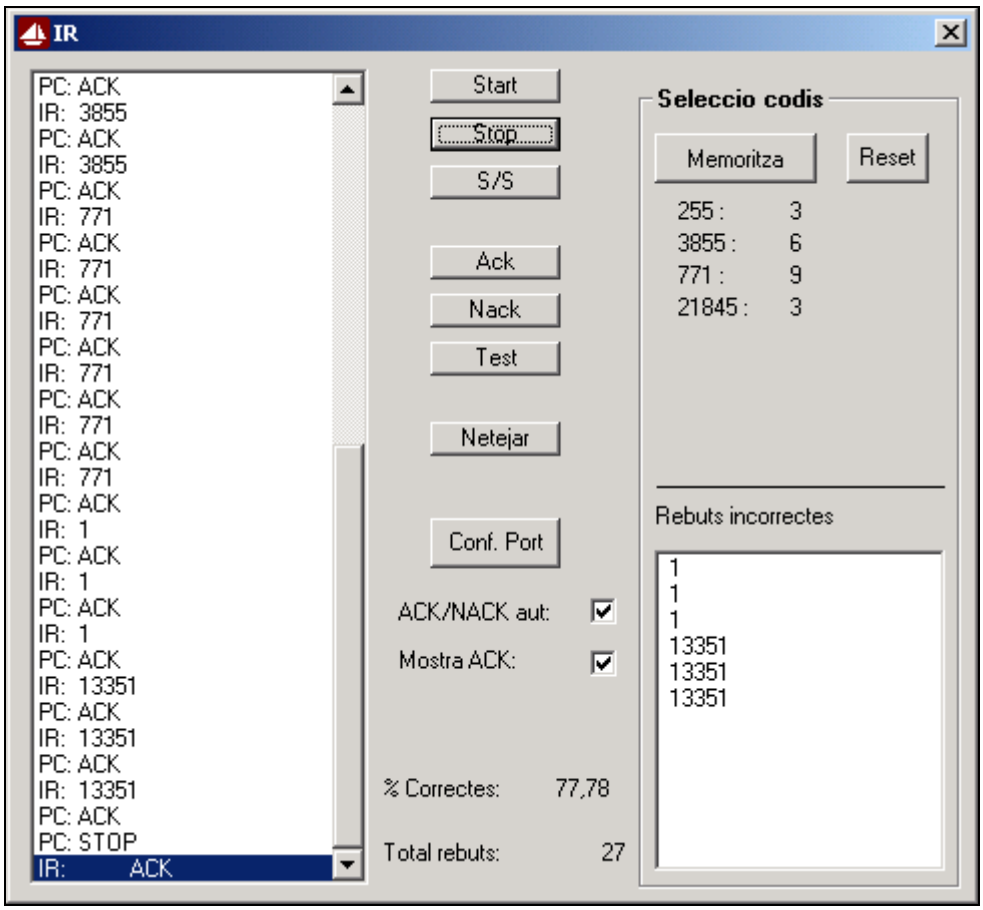


Fig 6.1: Pantalla principal

6.2.2.- Funcionament

La pantalla principal està dividida en tres parts. L'esquerra mostra el trànsit del port sèrie, la central s'utilitza per enviar ordres, configurar el port i canviar opcions de visualització, mentre que la part dreta permet memoritzar fins a vuit codis i mostrar quins del codis rebuts són diferents als memoritzats.

Visualització del trànsit

La visualització del trànsit no es fa a nivell de byte. És a dir, si es transmet un codi IR del receptor al PC és mostrarà el codi rebut i no els quatre bytes utilitzats per enviar aquest codi. Els diferents tipus d'informació que es poden mostrar són els següents:

Dada	Descripció	Origen
START	Enviament ordre Start	PC
STOP	Enviament ordre Stop	PC
TEST	Enviament ordre Test	PC
(CODI)	Codi rebut	Receptor
RESET	El receptor s'ha reiniciat per fallada d'alimentació o per polsació del botó de reset.	Receptor
HANG	El receptor s'ha reiniciat pel Watchdog	Receptor
ACK	Enviament ACK	PC o receptor
NACK	Enviament NAC	PC o receptor

A la finestra sempre es mostra d'on prové l'informació mostrada si del PC o del receptor (IR).

Un cas especial és la comanda Test. Aquesta provoca l'enviament d'una gran quantitat de bytes en text i per visualitzar-la s'allarga la finestra del programa per mostrar una nova caixa de text amb la informació rebuda:

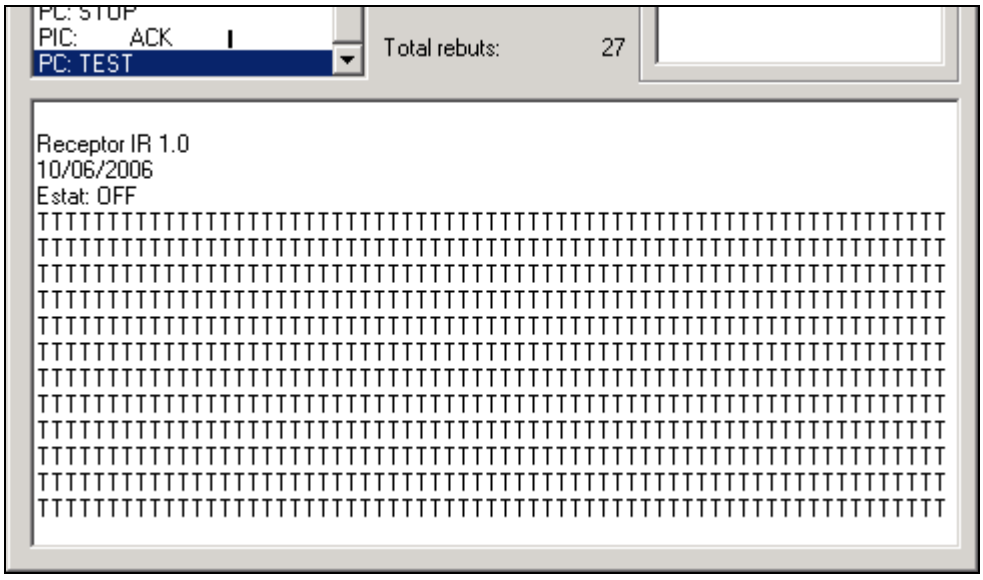


Fig 6.2: Visualització test

Botons

A la part central es disposa de vuit botons que tenen la següent funció:

Botó	Funció
Start	Envia ordre Start
Stop	Envia ordre stop
S/S	Envia alternativament una ordre Start o Stop a cada polsació
Ack	Envia ACK
Nack	Envia NACK
Test	Envia ordre Test
Netejar	Esborra el contingut de les finestres de trànsit i incorrectes. Posa a zero els comptadors.
Conf. Port	Permet indicar el port utilitzat i configurar-lo a una de les quatre velocitats disponibles.

El botó 'S/S' és útil per 'bombardejar' el receptor i comprovar que la recepció d'ordres no influeix en la descodificació dels codis rebuts.

'Conf. Port' mostra una finestra on es pot indicar el port utilitzat i configurar-lo a qualsevol de les velocitats predeterminades del receptor. Aquest és el seu aspecte:

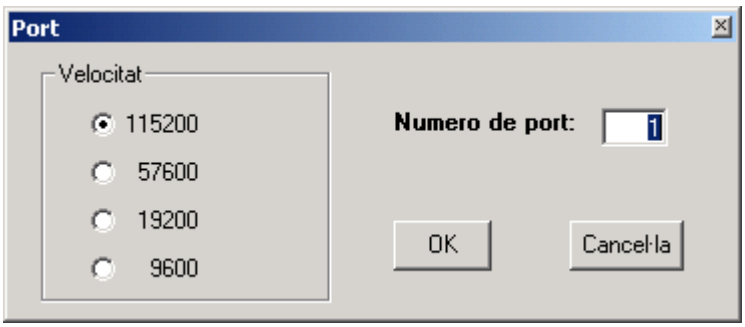


Fig 6.3: Configuració port sèrie

Opcions

Dues caselles permeten seleccionar les opcions 'ACK automàtic' i 'Mostra ACK'. Per defecte, al iniciar el programa, la primera està seleccionada i la segona no. No mostrar els ACK que confirmen les comunicació millora la legibil·litat del trànsit rebut. Desactivar 'ACK automàtic' permet comprovar el protocol de comunicacions i enviar manualment ACK o NACK per comprovar com el receptor re-envia els codis fins que rep un ACK.

Una prova interessant és desactivar l'enviament d'ACK i fer funcionar varis comandaments per omplir el buffer del receptor. Posteriorment, en tornar a activar aquesta opció s'enviaran tots els codis emmagatzemats seguits.

Selecció de codis

Un click al botó 'memoritza' fa que aquest es quedi polsat fins que rebi un altre click. Durant aquest temps els codis rebuts es memoritzen i es mostren a l'espai que hi ha sota aquest botó (fins a un màxim de vuit).

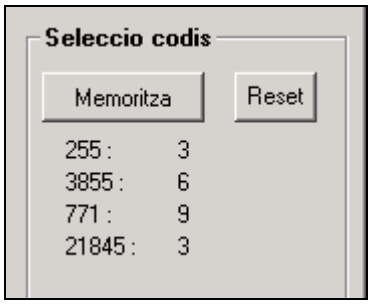


Fig 6.4: Visualització selecció de codis

Posteriorment, cada codi rebut es compararà amb els memoritzats. Si és un d'ells s'incrementarà el seu comptador associat, altrament apareixerà dins la caixa de text 'Rebuts incorrectes'. El botó reset eliminarà els codis memoritzats i els incorrectes.

Aquesta opció té la finalitat de comprovar fàcilment amb diversos comandaments a la vegada si l'enviament simultani de varis codis en poden generar un de diferent.

6.1.3.- Codi

```
'--------------------------------------------------------------------------------
' Receptor.vbp
'
'       L'objectiu del programa es provar totes les característiques del receptor IR.
'   Permet enviar ordres i mostra els codis rebuts. També mostra la trama que s'envia
'   amb la comanda Test.
'
'       Per facilitar la detecció d'errors es poden memoritzar fins a vuit codis
'   diferents i es diferencia si els codis rebuts pertanyen a aquest grup.
'
'       Es pot configurar qualsevol port del PC per treballar amb una de les quatre
'   velocitats predeterminades que permet el receptor.
'--------------------------------------------------------------------------------

Option Explicit

'--------------------------------------------------------------------------------
'----- TIPUS, VARIABLES I CONSTANTS -----
'----- Tipus de dades -------

Enum TpEstat    'TpEstat ens guarda en quin punt tenim la recepció
    none = 0        'No tenim res pendent
    SOH = 1         'Hem rebut un SOH (inici d'una trama de codi)
    STX = 2         'Hem rebut un STX
    B1 = 3          'Hem rebut el primer byte d'un codi
    B2 = 4          'Hem rebut el segon byte d'un codis
    Test = 5        'Estem rebent una trama de test
End Enum

'----- Variables -----

Dim Estat As TpEstat            'Punt actual de la recepció
Dim Ultim As Byte               'Estat del receptor: On o Off
Dim cntCodis As Integer         'Nombre total de codis correctes rebuts
Dim ArrayCodis(1 To 8) As Long  'Llista de codis memoritzats
Dim nCodis As Integer           'Nombre de codis guardats a ArrayCodis
```

```
'----- Constants -----

Const kACK As Byte = 6
Const kNACK As Byte = 21
Const kSOH As Byte = 1
Const kSTX As Byte = 2
Const kSTART As Byte = 11
Const kSTOP As Byte = 13
Const kRESET As Byte = 5
Const kHANG As Byte = 7
Const kTEST As Byte = 84

'-------------------------------------------------------------------------------
'----- CODI BOTONS -----

Private Sub cmdStart_Click()

'   cmdStart_Click
'
'       Envia ordre d'Start al receptor. No es controla la recepció d'ACK

    Serie.Output = Chr(kSTX) + Chr(kSTART)
    lstTransit.AddItem ("PC: START")
    lstTransit.ListIndex = lstTransit.ListCount - 1
    Ultim = kSTART
End Sub

Private Sub cmdStop_Click()

'   cmdStop_Click
'
'       Envia ordre d'Stop al receptor. No es controla la recepció d'ACK

    Serie.Output = Chr(kSTX) + Chr(kSTOP)
    lstTransit.AddItem ("PC: STOP")
    lstTransit.ListIndex = lstTransit.ListCount - 1
    Ultim = kSTOP
End Sub

Private Sub CmdSS_Click()

'   CmdSS_Click
'
'       A cada polsació canvia l'estat del receptor de activat a desactivat i viceversa

    If Ultim = kSTART Then
        cmdStop_Click
    Else
        cmdStart_Click
    End If
End Sub

Private Sub cmdAck_Click()

'   CmdAck_Click
'
'       Envia la comanda d'ACK pel port sèrie i la mostra si tenim l'opció
'   "Mostra ACK" seleccionada

    Serie.Output = Chr(kACK)
    If chkEchoACK.Value = 1 Then lstTransit.AddItem ("PC: ACK")
    lstTransit.ListIndex = lstTransit.ListCount - 1
End Sub

Private Sub cmdNack_Click()

'   CmdNack_Click
'
'       Envia la comanda de NACK pel port sèrie i la mostra

    Serie.Output = Chr(kNACK)
    lstTransit.AddItem ("PC: NACK")
    lstTransit.ListIndex = lstTransit.ListCount - 1
End Sub

Private Sub cmdTest_Click()

'   cmdTest_Click
```

```
'
'       Envia ordre de Test al receptor i allarga el formulari per poder visualitzar
'   la informació rebuda. Tornant-lo a polsar el formuari recuperarà les seves
'   dimensions originals. També desactiva la resta de botons durant el procés.

    If Estat = none Then
        cmdTest.Caption = "Fi test"     'Canvia el contingut del botó
        Serie.Output = Chr(kSTX) + Chr(kTEST)
        lstTransit.AddItem ("PC: TEST")
        lstTransit.ListIndex = lstTransit.ListCount - 1
        Estat = Test                    'Indiquem que estem en Test

        frmPrincipal.Height = 10260     'Redimensiona el formulari

        cmdStart.Enabled = False        'Deshabilita botons
        cmdStop.Enabled = False
        CmdSS.Enabled = False
        cmdAck.Enabled = False
        cmdNack.Enabled = False

    ElseIf Estat = Test Then
        txtTest.Text = ""               'Elimina informació rebuda de la caixa de text
        cmdTest.Caption = "Test"        'Canvia contingut del botó
        Estat = none                    'Estat de la recepció

        frmPrincipal.Height = 6720      'Redimensiona per no mostrar caixa de text de
test

        cmdStart.Enabled = True         'Torna a habilitar els demés botons
        cmdStop.Enabled = True
        CmdSS.Enabled = True
        cmdAck.Enabled = True
        cmdNack.Enabled = True
    End If
End Sub

Private Sub cmdLimpia_Click()

'   cmdLimpia_Click
'
'       Elimina les llistes de codis rebuts i els comptadors. Conserva el llistat de
'   codis a rebre.

    Dim i As Integer

    lstTransit.Clear
    lblNCodis.Caption = ""
    lstFalsos.Clear

    For i = 1 To nCodis
        lblNCodi(i) = ""
    Next i

    cntCodis = 0
    lblPrp.Caption = ""
End Sub

Private Sub cmdPort_Click()

'   cmdPort_Click
'
'       Mostra el formulari de configuració del port sèrie. Aquest accedeix directament
'   al control del sèrie per carregar els paràmetres que indiqui l'usuari.

    frmNPort.Show
End Sub

Private Sub cmdReset_Click()

'   cmdReset_Click
'
'       Buida la llista de codis a rebre i la de codis rebuts incorrectes.

Dim i As Integer

    For i = 1 To nCodis         'Elimina dades mostrades de codis guardats
        lblCodi(i).Caption = ""
        lblNCodi(i).Caption = ""
    Next i
```

```
    nCodis = 0                  'Marca que no tenim codis guardats
    lstFalsos.Clear             'Neteja llistat de codis incorrectes
End Sub

'--------------------------------------------------------------------------------
'----- FUNCIONS PRINCIPALS -----

Private Sub Form_Load()

'   Form_Load
'
'       Inicia el programa

On Error GoTo ErrPort

Dim i As Byte

    ChckACK.Value = 1           'Marquem que s'enviïn els ACK automàticament
    chkEchoACK.Value = 0        'Marquem que no cal que ens mostri els ACK enviats
    nCodis = 0                  'Nombre de codis rebuts fins el moment: 0

    With Serie                  'Configuració inicial del port sèrie
        .InputLen = 1               'Lleguim el contingut caràcter a caràcter
        .InputMode = comInputModeText   'Entrada en mode text
        .RThreshold = 1             'Nombre de caràcters en buffer per genera event
OnComm
        .CommPort = 5               'Port utilitzat
        .PortOpen = True            'Obre el port
    End With
    Exit Sub

ErrPort:
    If Err = 8002 Then
        frmNPort.Show vbModal, Me
    End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
    Unload frmNPort
End Sub

Private Sub Serie_OnComm()

'   Serie_OnComm
'
'       És l'event que genera el port sèrie quan arriba un byte (entre altres motius).
'   Funciona com una màquina d'estats, guardant l'estat actual a la variable Estat.
'   Rep els bytes, controla el CRC i, si cal, guarda la informació rebuda.

    Dim ch As Integer
    Dim s As String
    Static codi As Long
    Static crc As Byte

    Serie.RThreshold = 0        'Desactivem event per no entrar en recursivitat
    While Serie.InBufferCount > 0   'Mentre tinguem alguna cosa en el buffer
d'entrada...
        s = Serie.Input     'Lleguim el buffer. Per configuració es tractarà d'una
caden d'un caràcter
        ch = Asc(s)         'ch = valor ASCII del primer (i únic) caràcter rebut
        Select Case Estat
            Case none:  'Si no teniem res pendent...
                        If (Significat(ch) <> "") Then      'Si no ho entenem, ignorem
                            lstTransit.AddItem ("IR: " + vbTab + Significat(ch))
                            lstTransit.ListIndex = lstTransit.ListCount - 1
                        End If
                        If ch = kSOH Then Estat = SOH   'Passem a següent estat si
estem rebent un codi

            Case SOH:   'Ja hem rebut un inici de codi. Rebem el primer byte
                        Estat = B1
                        codi = ch

            Case B1:    'Rebem el segon byte del codi i calculem CRC
                        Estat = B2
                        crc = codi Xor ch
                        codi = codi * 256 + ch

            Case B2:    'Segons si el CRC rebut és correcte...
```

```
                        If crc = ch Then
                                'Mostra codi rebut i actualitza numero de codis
                                lstTransit.AddItem ("IR: " + Str(codi))
                                cntCodis = cntCodis + 1
                                lblNCodis.Caption = Str(cntCodis)


                                'Si estem memoritzant codis, el guarda si no el tenim
                                If chkCodis.Value = Checked Then EntraCodi (codi)

                                'Si tenim codis guardats, comprova si és un d'ells
                                If nCodis <> 0 Then VerificaCodi (codi)

                                'Actualitza percentatge de codis correctes
                                lblPrp.Caption = Format(100 * (cntCodis -
lstFalsos.ListCount) / cntCodis, "##0.00")

                                'Enviem ACK si tenim opció marcada
                                If ChckACK.Value = Checked Then
                                    cmdAck_Click
                                End If
                           Else
                                'El CRC no coincideix. Enviem NACK
                                cmdNack_Click
                        End If
                        lstTransit.ListIndex = lstTransit.ListCount - 1
                        Estat = none    'Tornem a estat inicial

            Case Estat:
                        'Si estem en Test, ens limitem a mostrar el byte rebut
                        txtTest.Text = txtTest + Chr(ch)
        End Select
    Wend
    Serie.RThreshold = 1
End Sub

'---------------------------------------------------------------------------------------
'----- FUNCIONS SECUNDÀRIES -----

Private Function Significat(ch As Integer) As String

'   Significat
'
'       Retorna una cadena amb el significat del valor passat com a paràmetre.

    Select Case ch
        Case kACK:  Significat = "ACK"
        Case kNACK: Significat = "NACK"
        'Case kSOH:  Significat = "SOH"
        'Case kSTX:  Significat = "STX"
        Case kRESET: Significat = "RESET"
        Case kHANG: Significat = "HANG"
        Case Else:  Significat = ""
    End Select
End Function

Private Sub EntraCodi(codi As Long)

'   EntraCodi
'
'       Guarda i mostra el codi passat com a paràmetre si encara no el tenim. Comprova
'   que no tinguem la llista plena.

    Dim i As Integer
    Dim trobat As Boolean

    'Busca si el tenim
    trobat = False
    i = 1
    While Not trobat And i <= nCodis
        If ArrayCodis(i) = codi Then
            trobat = True
        Else
            i = i + 1
        End If
    Wend

    'Si l'ha trobat el guarda i el mostra
    If trobat = False And nCodis < 8 Then
```

```
        nCodis = nCodis + 1
        ArrayCodis(i) = codi

        lblCodi(i).Caption = Str(codi) + " :"

        'Si no en caben mes desactiva el mode
        If nCodis = 8 Then chkCodis.Value = 0
    End If

End Sub
Private Sub VerificaCodi(codi As Long)

'   VerificaCodi
'
'       Comprova si tenim el codi passat com a paràmetre a la llista de codis guardats.
'   Si el tenim incrementa el seu comptador, si no el tenim l'afegeix a la llista
'   d'incorrectes

    Dim i As Integer
    Dim trobat As Boolean

    'Busca si el tenim
    trobat = False
    i = 1
    While Not trobat And i <= nCodis
        If codi = ArrayCodis(i) Then
            trobat = True
        Else
            i = i + 1
        End If
    Wend

    'Segons si l'ha trobat
    If trobat = True Then
        'Incrementa el seu comptador
        lblNCodi(i).Caption = Str(Val(lblNCodi(i).Caption) + 1)
    Else
        'L'afegeix a la llista d'incorrectes
        lstFalsos.AddItem (Str(codi))
    End If
End Sub
```

```
'--------------------------------------------------------------------------------
'   frmNPort
'
'   Aquest formulari permet la configuració del port per part de l'usuari. Aquest
'   podrà escollir el port a utilitzar amb quatre velocitats diferents (9600,
'   19200,57600 i 115200). Es detecta si es genera un error de port incorrecte.
'--------------------------------------------------------------------------------

Private Sub cmdOK_Click()

'   cmdOK_Click
'
'       Intenta obrir el port indicat al quadre de texte amb la velocitat
'   marcada mitjançant els botons d'opció. Si es genera error ho informa i deixa
'   la configuració anterior.


    Dim strSpeed As String
    Dim nPortOrg As Integer
    Dim SetPortOrg As String

    On Error GoTo ProduitErr

    'Es selecciona la configuració del port segons l'option seleccionat
    If OptSpeed(0).Value = True Then
        strSpeed = "115200,n,8,1"
    ElseIf OptSpeed(1).Value = True Then
        strSpeed = "57600,n,8,1"
    ElseIf OptSpeed(2).Value = True Then
        strSpeed = "19200,n,8,1"
    ElseIf OptSpeed(3).Value = True Then
        strSpeed = "9600,n,8,1"
    End If

    'Es selecciona el port i es configura segons les seleccions. S'intenta obrir.
    With frmPrincipal.Serie
```

```
        If .PortOpen = True Then .PortOpen = False
        nPortOrg = .CommPort
        .CommPort = Val(txtPort.Text)
        SetPortOrg = .Settings
        .Settings = strSpeed
        .PortOpen = True
    End With
    Unload Me
    Exit Sub

ProduitErr:
    'S'hi es produeix error l'indiquem amb un missatge...
    If Err = 8002 Then
        MsgBox "El port indicat no és vàlid"
    Else
        MsgBox Err.Description, vbOKOnly, "Error" + Str(Err.Number)
    End If

    '...i carreguem la configuració original
    frmPrincipal.Serie.CommPort = nPortOrg
    frmPrincipal.Serie.Settings = SetPortOrg
    frmPrincipal.Serie.PortOpen = True
    txtPort.SelStart = 1
    txtPort.SelLength = 3
End Sub


Private Sub Form_Activate()
    txtPort.SetFocus
End Sub

Private Sub Form_Load()

'   Form_Load
'
'       Mostra la configuració actual del port i selecciona el text amb el
'      numero de port.

    txtPort.Text = Str(frmPrincipal.Serie.CommPort)
    txtPort.SelStart = 1
    txtPort.SelLength = 3
    Select Case Val(Left$(frmPrincipal.Serie.Settings,
InStr(frmPrincipal.Serie.Settings, ",") - 1))
        Case 9600
            OptSpeed(3).Value = True
        Case 19200
            OptSpeed(2).Value = True
        Case 57600
            OptSpeed(1).Value = True
        Case 115200
            OptSpeed(0).Value = True
    End Select


End Sub

Private Sub CmdCancel_Click()
    Unload Me
End Sub
```

6.2.- ProgMaster

6.2.1.- Pantalla principal

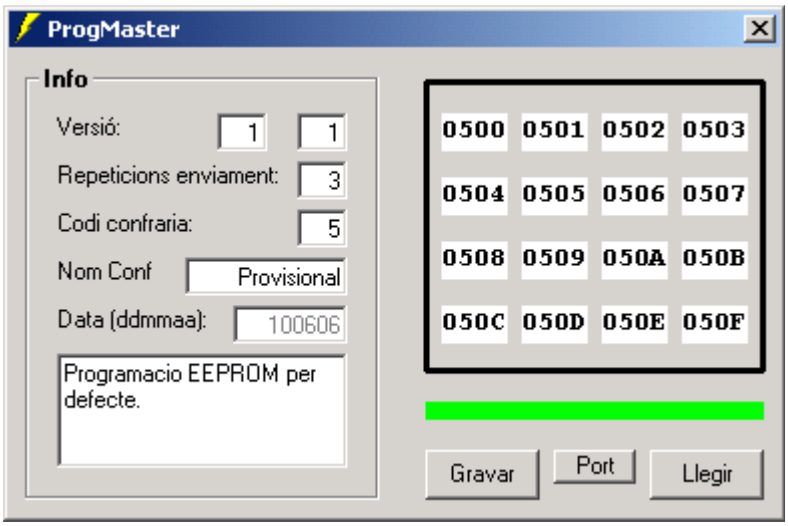


Fig 6.5: Pantalla principal

6.2.2.- Funcionament

El programa permet visualitzar i modificar el contingut de la EEPROM del comandament que s'hi connecti. No fa un llistat pla del contingut de les 128 posicions de la memòria sino que el mostra degudament classificat segons el seu significat.

En el requadre de la part dreta de la finestra hi ha una representació del teclat amb els codis que envia cada tecla (en hexadecimal). El requadre info, situat a la seva esquerra, mostra la resta de la informació.

Tots els continguts son modificables excepte la data de la última modificació. Aquesta es genera automàticament en realitzar una gravació.

Connexió

Amb un cable de programació connectant el port sèrie utilitzat amb el comandament, cal prèmer qualsevol tecla d'aquest per que surti del mode baix consum i entri en el mode de programació.

El programa realitza un pooling constant per saber si hi ha algun comandament connectat. Si es així, la barra situada a sota del teclat serà de color verd, en cas contrari, la barra serà vermella. Aquest pooling es continua realitzant quan tenim un comandament connectat per poder detectar en qualsevol moment si es perd la comunicació.

El pooling consisteix en intentar llegir la posició 0 de la EEPROM i comprovar que es rep alguna resposta.

Botons

Botó	Funció
Gravar	Guarda la informació mostrada al comandament
Llegir	Mostra la informació que conté el comandament
Port	Permet seleccionar el port utilitzat

Els botons 'Gravar' i 'Llegir' permeten fer un volcat de la informació mostrada en pantalla al comandament i viceversa. Aquests botons només estan habilitats quan tenim connectat un comandament en mode programació i, per tant, la barra que ens ho indica és verda. Aquestes operacions esperen una resposta per cada byte llegit o gravat i mostren un missatge d'error si aquesta no arriba.

El boto 'Port' mostra la següent finestra per seleccionar el port utilitzat. La configuració d'aquest és fixe: 8,n,1 a 9600 bps.

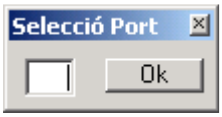


Fig 6.6: Selecció del port

6.3.3.- Codi

```
'--------------------------------------------------------------------------------
' ProgMaster.vbp
'
'    El programa permet visualitzar i modificar el contingut de la EEPROM del
'   comandament que s'hi connecti. No fa un llistat pla del contingut de les 128
'   posicions de la memòria sino que el mostra degudament classificat segons el seu
'   significat.
'--------------------------------------------------------------------------------

Option Explicit

'--------------------------------------------------------------------------------
'----- VARIABLES -----

Dim TimeOut As Boolean          'True indica que tenim un comandament connectat

'--------------------------------------------------------------------------------
'----- CODI TIMERS -----

Private Sub tmrPooling_Timer()

'   tmrPooling_Timer
'
'       Si està habilitat, s'executa periòdicament amb una periodicitat configurada
'   a Form_Load. Cada cop que s'executa envia l'ordre de lectura de la posició 0 de
'   la EEPROM. Previament consulta si ha arribat algun byte pel port en resposta a
'   una ordre anterior. Així comprova si hi ha algún comandament correctament con-
'   nectat i ho indica canviat el color de lblOnline.

    Static Cont As Integer  'Conta les vegades seguides que ha rebut o no resposta
    Static OnLine As Integer
    Dim i As Integer

    If OnLine = True Then
        If ctrSerie.InBufferCount < 1 Then
            Cont = Cont + 1             'No hem rebut resposta a l'ultima ordre
            If Cont > 3 Then
                OnLine = False              'Tres seguides. Considerem conexió perduda
                Cont = 0

                cmdGravar.Enabled = False   'Deshabilitem botons
                cmdLlegir.Enabled = False

                lblOnline.BackColor = &HFF& 'Indiquem amb color vermell
            End If
        Else
            ctrSerie.InBufferCount = 0      'Hem rebut resposta. Buidem buffer sèrie
            Cont = 0
        End If
    Else                                    '(No tenim connexió)
        If ctrSerie.InBufferCount < 1 Then
            Cont = 0                        'Continuem sense rebre resposta
        Else
            ctrSerie.InBufferCount = 0      'Hem rebut resposta!
            Cont = Cont + 1                 'Acumulem
            If Cont > 3 Then
                OnLine = True                   'Tres seguides. Considerem conexió OK

                lblOnline.BackColor = &HFF00&   'Marquem amb verd
                cmdLlegir.Enabled = True        'Habilitem botons
                cmdGravar.Enabled = True
            End If
        End If
    End If
    If tmrPooling.Enabled = True Then ctrSerie.Output = Chr(0)  'Tornem a consultar
End Sub

Private Sub tmrTimeOut_Timer()

'   tmrTimeOut_Timer
'
'       Si està habilitat, s'executa periòdicament amb una periodicitat configurada
'   a Form_Load. Únicament marca un flag conforme s'ha activat
```

```
    TimeOut = True
    tmrTimeOut.Enabled = False
End Sub

'--------------------------------------------------------------------------------
'----- FUNCIONS EEPROM -----

Public Function LlegirEEprom() As Boolean

'   LlegirEEprom
'
'       Omple un array amb l'informació llegida del comandament. Posteriorment
'   presenta l'informació. Retorna si ha completat l'operació

    Dim EEprom(0 To 127) As Byte
    Dim i As Integer
    Dim s As String
    Dim Error As Boolean

    MousePointer = vbHourglass      'Es mostra cursor rellotge de sorra

    tmrPooling.Enabled = False      'Deshabilita pooling perquè no interfereixi

    TimeOut = False                 'Espera un temps per a una possible resposta..
    tmrTimeOut.Enabled = True       '..a una ordre de pooling perduda
    While TimeOut = False
        DoEvents
    Wend

    ctrSerie.InBufferCount = 0      'Buida el buffer del sèrie

    i = 0
    Error = False
    While i <= 127 And Not Error
        TimeOut = False                 'Reset flag TimeOut
        tmrTimeOut.Enabled = True       'Engeguem timer
        ctrSerie.Output = Chr(i)        'Ordre de lectura de la posició 'i'

        While (ctrSerie.InBufferCount < 1) And (TimeOut = False)
            DoEvents                    'Esperem sense penjar el Windows...
        Wend
        tmrTimeOut.Enabled = False

        If ctrSerie.InBufferCount > 0 Then
            s = ctrSerie.Input              'Hem rebut byte
            EEprom(i) = Asc(Mid(s, 1, 1))   'Guardem a l'array
            i = i + 1
        Else
            Error = True                'Hem saltat per time-out. Abortem
        End If
    Wend

    If Not TimeOut Then    'Si hem completat la lectura mostrem informació
        txtVersio(0).Text = Val(EEprom(0))  'Versió
        txtVersio(1).Text = Val(EEprom(1))

        For i = 0 To 15                     'Codis assignats a tecles
            s = Hex(CLng(EEprom(2 + 2 * i)) * 256 + EEprom(3 + 2 * i))
            txtCodiTecla(i).Text = Right("000" + s, 4)
        Next i

        txtRepCodi.Text = Val(EEprom(34))   'N. enviaments per codi
        txtNConfra.Text = Val(EEprom(35))   'Codi de la confraria

        s = ""
        For i = 36 To 55                    'Nom confraria
            s = s + Chr(EEprom(i))
        Next i
        txtNomConfra.Text = s

        s = Format(Str(EEprom(56)), "00") + Format(Str(EEprom(57)), "00") +
Format(Str(EEprom(58)), "00")
        txtData.Text = s                    'Data última modificació

        s = ""                              'Cadena de text lliure
        For i = 59 To 127
            s = s + Chr(EEprom(i))
        Next i
        txtLliure.Text = s
```

```
    End If

    tmrPooling.Enabled = True  'Torna a activar el pooling
    MousePointer = vbArrow                  'El cursor torna a ser una fletxa
    LlegirEEprom = Not Error                'Retorna si ho ha aconseguit
End Function

Public Function GravarEeprom() As Boolean

'   GravarEEprom:
'
'       Construeix un array amb la informació a guardar al comandament. Un cop creat
'   s'envia byte a byte controlant resposta ACK del mando per a cada byte enviat.
'   Retorna si ha completat l'operació

    Dim EEprom(0 To 127) As Byte
    Dim i As Integer
    Dim s As String
    Dim Error As Boolean

    MousePointer = vbHourglass              'Es mostra cursor rellotge de sorra

                                            'Composa contingut EEPROM
    EEprom(0) = Val(txtVersio(0).Text)      'Versió
    EEprom(1) = Val(txtVersio(1).Text)

    For i = 0 To 15                         'Codis tecles
        EEprom(2 + i * 2) = HexToInteger(txtCodiTecla(i).Text) \ 256
        EEprom(3 + i * 2) = HexToInteger(txtCodiTecla(i).Text) Mod 256
    Next i

    EEprom(34) = Val(txtRepCodi.Text)       'N. enviaments per codi
    EEprom(35) = Val(txtNConfra.Text)       'Codi de la confraria

    For i = 1 To 20                         'Nom de la confraria
        s = (Mid(txtNomConfra.Text, i, 1))
        If s <> "" Then
            EEprom(35 + i) = Asc(s)         '(Asc("") genera un error)
        Else
            EEprom(35 + i) = 32
        End If
    Next i

    EEprom(56) = Day(Date)                  'Data ultima modificació = data actual
    EEprom(57) = Month(Date)
    EEprom(58) = Year(Date) Mod 100

    For i = 1 To 69                         'Cadena de text lliure
        s = (Mid(txtLliure.Text, i, 1))
        If s <> "" Then
            EEprom(58 + i) = Asc(s)
        Else
            EEprom(58 + i) = 32
        End If
    Next i

    tmrPooling.Enabled = False            'Deshabilita pooling perquè no interfereixi

    TimeOut = False                       'Espera un temps per a una possible
resposta..
    tmrTimeOut.Enabled = True             '..a una ordre de pooling perduda
    While TimeOut = False
        DoEvents
    Wend

    ctrSerie.InBufferCount = 0            'Buida el buffer del sèrie

    Error = False
    i = 0

    While i <= 127 And Not Error
        ctrSerie.Output = Chr(i + 128) & Chr(EEprom(i)) 'Dos bytes: posició i dada

        TimeOut = False                     'Reset flag TimeOut
        tmrTimeOut.Enabled = True           'Engeguem timer

        While ctrSerie.InBufferCount < 1 And TimeOut = False
            DoEvents                        'Esperem sense penjar el Windows...
        Wend
```

```
        tmrTimeOut.Enabled = False          'Parem timer

        i = i + 1                           'Pròxima posició a guardar

        If ctrSerie.InBufferCount > 0 Then
            s = ctrSerie.Input                          'Tenim resposta
            If Asc(Mid(s, 1, 1)) <> 6 Then Error = True 'No és ACK. Malament
        Else
            Error = True                    'No tenim ni resposta. Marquem error
        End If
    Wend

    tmrPooling.Enabled = True               'Torna a activar el pooling
    MousePointer = vbArrow                  'El cursor torna a ser una fletxa
    GravarEeprom = Not Error                'Retorna si ho ha aconseguit
End Function

'-------------------------------------------------------------------------------
'----- INICI I FINAL DEL PROGRAMA -----

Private Sub Form_Load()

'   Form_Load
'
'       Configura i activa timers i port sèrie

    tmrPooling.Interval = 200       'Pooling s'executa cada 200mS
    tmrPooling.Enabled = True

    tmrTimeOut.Interval = 100       'Temps màxim d'espera de resposta, 200mS
    tmrTimeOut.Enabled = False

    With ctrSerie                   'Configurem paràmetres port
        .CommPort = 1
        .InputLen = 1
        .RThreshold = 0
        .OutBufferCount = 0
        .InBufferCount = 0
        .PortOpen = True
        .Settings = "9600,n,8,1"
    End With
End Sub


Private Sub Form_Unload(Cancel As Integer)
    Unload frmPort
End Sub

'-------------------------------------------------------------------------------
'----- CODI BOTONS -----

Private Sub cmdGravar_Click()

'   cmdGravar_Click
'
'       Grava tota la informació mostrada al comandament

    Dim opOk As Boolean

    opOk = GravarEeprom
    If opOk = False Then MsgBox "Error al gravar el comandament", vbOKOnly
End Sub

Private Sub cmdLlegir_Click()

'   cmdLlegir_Click
'
'       Realitza la lectura de la informació del comandament

    Dim opOk As Boolean

    opOk = LlegirEEprom
    If opOk = False Then MsgBox "Error al llegir el comandament", vbOKOnly
End Sub

Private Sub cmdPort_Click()

'   cmdPort_Click
```

```
'
'       Mostra el formulari frmPort que permet seleccionar el port utilitzat

    frmPort.Show vbModal, Me
End Sub

'--------------------------------------------------------------------------------
'----- VALIDACIÓ ENTRADES NOMBRES -----

'       Únicament es deixen entrar caràcters del '0' al '9' per nombres decimals.
'   Del '0' a '9' i de 'A' a 'F' (minus i majus) per nombres hexadecimals.
'   En tots els casos es deixa l'ASCII 8 (backspace)

Private Sub txtCodiTecla_KeyPress(Index As Integer, KeyAscii As Integer)
    KeyAscii = Asc(UCase(Chr(KeyAscii)))
    If (KeyAscii < Asc("0") Or (KeyAscii > Asc("9") And KeyAscii < Asc("A")) Or
KeyAscii > Asc("F")) And KeyAscii <> 8 Then
        KeyAscii = 0
    End If
End Sub

Private Sub txtCodiTecla_Validate(Index As Integer, Cancel As Boolean)
    txtCodiTecla(Index).Text = Right("0000" + txtCodiTecla(Index).Text, 4)
End Sub

Private Sub txtNConfra_KeyPress(KeyAscii As Integer)
    If (KeyAscii > Asc("9") Or KeyAscii < ("0")) And KeyAscii <> 8 Then
        KeyAscii = 0
    End If
End Sub

Private Sub txtNConfra_Validate(Cancel As Boolean)
    Dim i As Integer

    i = Val(txtNConfra.Text)
    If (i > 255) Or (i < 0) Then
        txtNConfra.Text = "0"
    End If
End Sub

Private Sub txtRepCodi_KeyPress(KeyAscii As Integer)
    If (KeyAscii > Asc("9") Or KeyAscii < ("0")) And KeyAscii <> 8 Then
        KeyAscii = 0
    End If
End Sub

Private Sub txtRepCodi_Validate(Cancel As Boolean)
    Dim i As Integer

    i = Val(txtRepCodi.Text)
    If (i > 255) Or (i < 0) Then
        txtRepCodi.Text = "0"
    End If
End Sub

Private Sub txtVersio_KeyPress(Index As Integer, KeyAscii As Integer)
    If (KeyAscii > Asc("9") Or KeyAscii < ("0")) And KeyAscii <> 8 Then
        KeyAscii = 0
    End If
End Sub

Private Sub txtVersio_Validate(Index As Integer, Cancel As Boolean)
    Dim i As Integer

    i = Val(txtVersio(i).Text)
    If (i > 255) Or (i < 0) Then
        txtVersio(Index).Text = "0"
    End If
End Sub

'--------------------------------------------------------------------------------
'----- FUNCIONS VÀRIES -----

Public Function HexToInteger(s As String) As Long

'   HexToInteger
'
'       Converteix un nombre expresat en hexadecimal (passat com a string) al seu
'   equivalent numèric
```

```
    Dim i As Integer
    Dim res As Long
    Dim c As Byte
    Dim Valid As Boolean

    i = 1
    res = 0
    Valid = True
    s = UCase(s)                    'Convertim minus en majus

    While Valid And i <= Len(s)
        c = Asc(Mid(s, i, 1))
        i = i + 1

        If (c >= Asc("0")) And (c <= Asc("9")) Then
            c = c - Asc("0")
        ElseIf (c >= Asc("A")) And (c <= Asc("F")) Then
            c = c - Asc("A") + 10
        Else
            Valid = False
        End If

        res = res * 16 + c
    Wend

    If Valid Then
        HexToInteger = res
    Else
        HexToInteger = 0
    End If
End Function
```

```
'--------------------------------------------------------------------------------
'   frmPort
'
'       Aquest formulari permet escollir el port a utilitzar. La resta de
'   configuracions són fixes.
'--------------------------------------------------------------------------------Option
Explicit

Private Sub cmdOk_Click()

'   cmdOk_Click
'
'       Configura el port demanat i l'intenta obrir. En cas d'error ho notifica

    On Error GoTo ErrorPort

    frmPrincipal.ctrSerie.CommPort = Val(txtnPort.Text)
    frmPrincipal.ctrSerie.PortOpen = True
    frmPrincipal.tmrPooling.Enabled = True
    Unload Me
    Exit Sub

ErrorPort:
    If Err = 8002 Then
        MsgBox "Port seleccionat erroni", vbOKOnly, "Error"
    End If
End Sub

Private Sub Form_Load()

'   Form_Load
'
'       Tanca el port i desactiva el pooling.

    If frmPrincipal.ctrSerie.PortOpen = True Then frmPrincipal.ctrSerie.PortOpen =
False
    frmPrincipal.tmrPooling.Enabled = False
End Sub
```

7.- TESTS

7.1.- CAPTURES OSCIL·LOSCOPI

En aquest apartat es mostren diferents mesures realitzades amb oscil·loscop. Bàsicament es tracta de mesurar la trama IR per verificar que es genera correctament. També s'ha mesurat el temps que es tarda a entrar a la ISR que realitza la mesura del temps entre flancs del senyal del detector.

Totes les imatges contenen, a la barra blava de la part superior, les escales de temps i tensió utilitzades. També hi apareixen dues linies discontínues verticals a cada imatge, que marquen punts d'interès. El temps entre les dues linies es mostra a la finestra dt inclosa en cada captura.

7.1.1.- Enviament de tres codis

- **Conté**: Enviament de tres codis seguits.
- **Origen**: Comandament. Sortida microcontrolador a transistor(RA2)
- **Observacions**: Es veu com el temps d'enviament d'un codi és d'1 mS i l'espera entre dos enviaments és d'uns 2 ms (temps aproximats).
- **Base temps**: 1ms

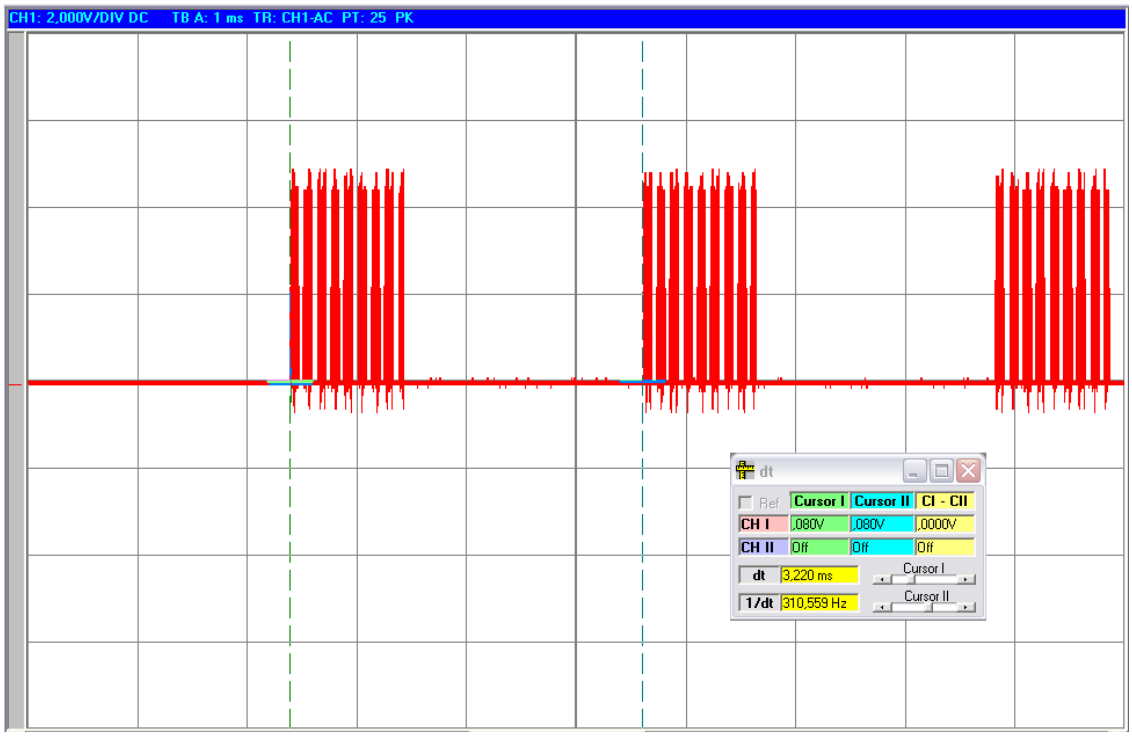


Fig 7.1: Captura enviament de tres codis

7.1.2.- Detall fragment de codi

- **Conté**: Enviament bit d'start i dels 5 primers bits del codi.
- **Origen**: Comandament. Sortida microcontrolador a transistor(RA2)
- **Observacions**: Es mostra el temps entre centres de bit (els flancs que donen el valor del bit). Resultat de 61.5µs.
- **Base temps**: 50µs

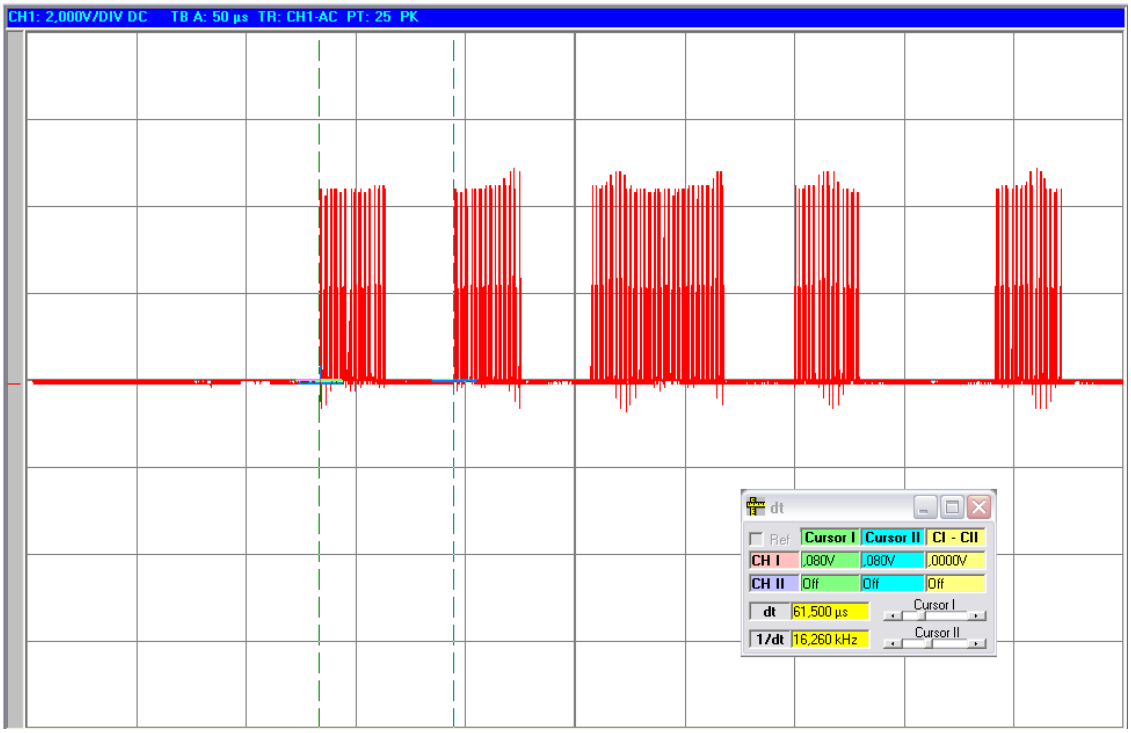


Fig 7.2: Captura enviament primers bits del codi

7.1.3.- Detall portadora

- **Conté**: Primers pulsos de la portadora
- **Origen**: Comandament. Sortida microcontrolador a transistor (RA2)
- **Observacions**: Es pot comprovar que el temps de cicle de la portadora és de 2.2µs i un cicle de treball del 10%.
- **Base temps**: 50µs

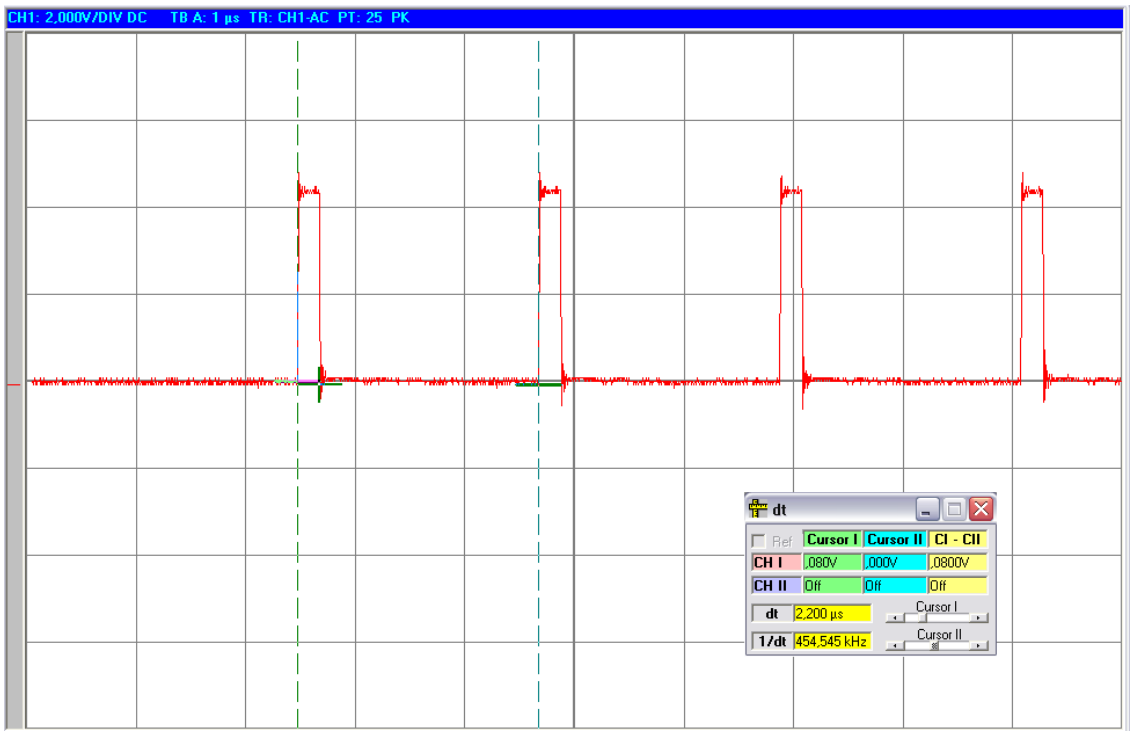


Fig 7.3: Captura portadora

7.1.4.- Sortida detector IR

- **Conté**: Comparació del senyal de sortida del comandament i del senyal de sortida del detector IR.

- **Origen**:
  - Superior: Sortida microcontrolador del comandament.
  - Inferior: Sortida detector IR del receptor.

- **Observacions**: El temps de retard en la sortida del receptor és, en aquesta mesura, de 18µs (15µS < t_{don} < 36µS segons datasheet). També es veu com la sortida està invertida i altera significativament els temps.

- **Base temps**: 50µs

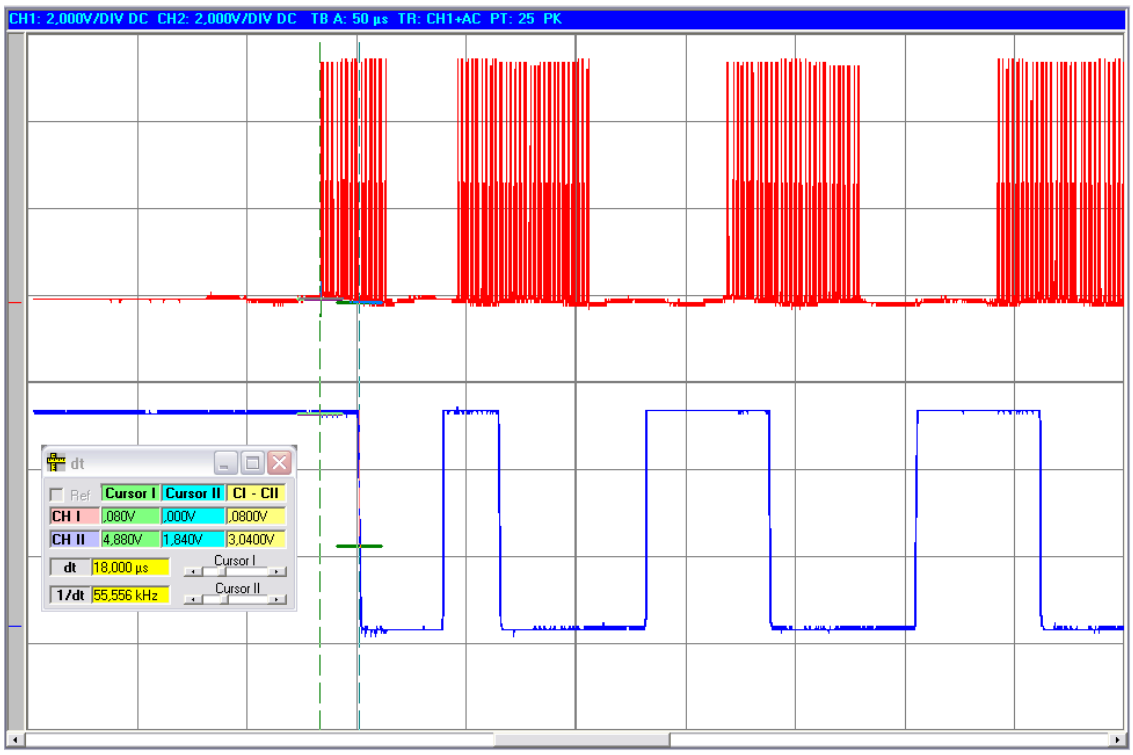


Fig 7.4: Captura sortida detector IR

7.1.5.- Retard en l'execució de la ISR

- **Conté**: Comparació del senyal de sortida del comandament i del senyal de sortida del detector IR.

- **Origen**:
  - Superior: Sortida detector IR del receptor.
  - Inferior: Senyal led IR

- **Observacions**: El led IR s'activa en la ISR de mesura de temps després de guardar el valor de TMR0. És una mesura que permet comprovar que no es produeix cap retard important en l'activació l'interrupció. El temps mesurat és de 4.8µs, al que hem de restar el temps d'execució de 20 instruccions (les que hi han entre l'inici de la ISR il'activació del led) i que tarden 4µs a executar-se.

- **Base temps**: 2µs

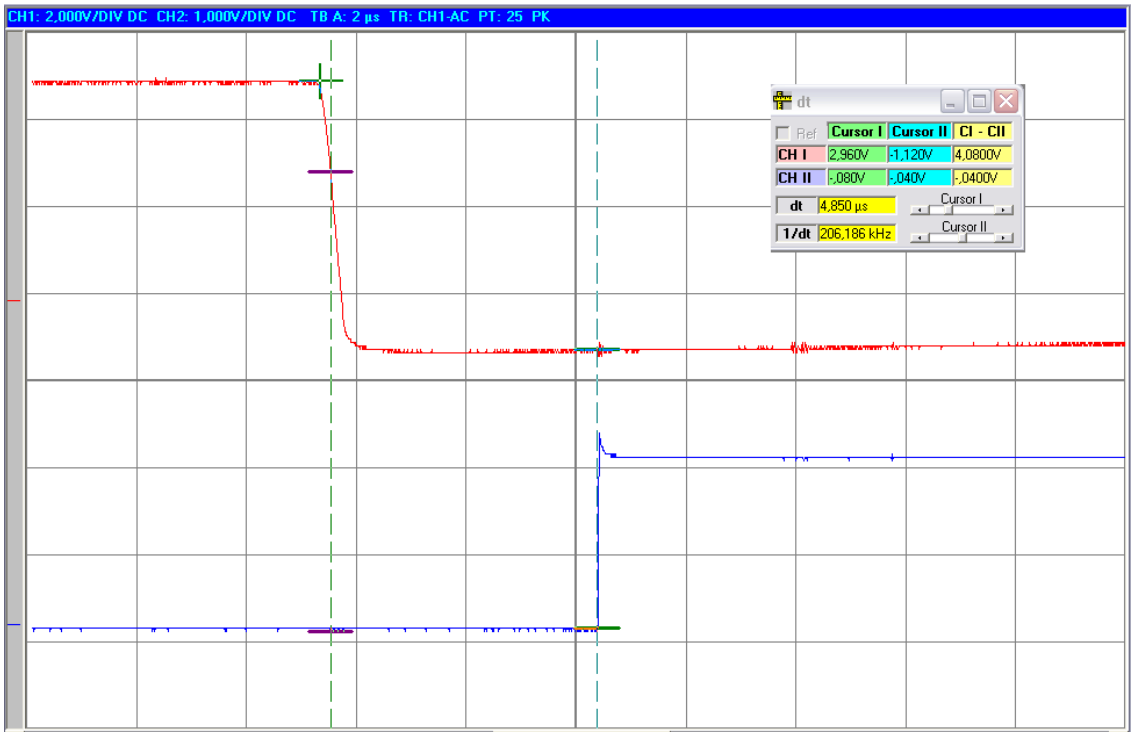


Fig 7.5: Captura execució ISR del detector

7.2.- DETERMINACIÓ CONSTANTS DEL RECEPTOR

El funcionament del receptor es basa en la mesura del temps passat entre dos flancs de la senyal del detector. Com s'indica en les seves coaracterístiques, i s'ha comprovar en les mesures realitzades, aquest temps és molt variable. Per aquest motiu cal utilitzar intèrvals per poder descodificar la trama rebuda.

Per trobar els més adequats s'ha creat el programa Test.asm a partir del programa del receptor perquè es envïi pel port sèrie les mesures realitzades. D'aquesta manera, cada cop que detecta un flanc, realitza la lectura del TMR0 (utilitzat per fer les mesures) i la guarda al buffer. Al igual que en el programa stàndard, aquest buffer és constantment comprovat per enviar-ne el contingut pel port sèrie. En total, per realitzar el programa s'han modificat les rutines _INT_PB (ISR del senyal del detector), _TX (envia contingut del buffer), _Encua i _Desencua.

Per recollir i visualitzar les dades també s'ha creat el programa Test_IR en VisualBasic. Aquest mostra una taula de freqüències amb les mesures enviades pel receptor amb el programa test.

7.2.1.- Programa Test.asm

A continuació es llisten les rutines que s'han modificat respecte el programa Receptor.asm:

_INT_PB

Recarrega el TMR0 a 0 per saber el temps, en cicles, des de l'últim flanc detectat. La recàrrega i lectura del timer es fa en el mateix moment que en el programa original per no falsejar els resultats.

```
;----- OPC --------------------------------------------------------------------------
;
;       Encua el valor del TMR0 i el recarrega a 0
;
;--------------------------------------------------------------------------------

INT_PB          movfw   TMR0            ;Apuntem contingut de TMR0 i el resetejem
                movwf   IR_TMR0
                nop                     ;Per coincidir amb els temps originals
                clrf    TMR0

                movfw   PORTB           ;Fa una lectura del portB
                bcf     INTCON, T0IF    ;Reset flag interrupcio TMR0 per si acaba de saltar
                bcf     INTCON, RBIF    ;Reset flag interrupcio PORTB

                btfsc   INTCON,T0IE     ;Tenim activat el TMR?
                goto    _PBData         ;       No: Saltem instruccions

_PBStart
                bsf     INTCON, T0IE    ;Activem la interrupció del TMR
                bsf     O_Curring       ;Encen led IR
                clrf    IR_TMR0         ;Indiquem enviant un 0 que és bit d'Start
_PBData
                call    _Encua
                return
```

_TX

És l'encarregada de buidar el buffer. Originalment envia quatre bytes per cada codi a enviar, aquí està molt simplificada ja que només envia un byte i no es realitza cap control d'error de comunicació.

```
;----- SERIE OUTPUT --------------------------------------------------------------
;
;       Si queda algun byte a la cua l'envia. No espera cap tipus de resposta
;
;-------------------------------------------------------------------------------

_TX             btfss   PIR1, TXIF      ;Si TXREG ple, marxa
                return

                movfw   Bfr_NBytes      ;Tenim algo per enviar?
                btfsc   STATUS,Z
                return                          ;  No

                call    _Desencua       ;Treu un byte i l'envia
                movfw   S_TMR0
                movwf   TXREG
                return
```

_Encua

Molt similar a l'original però guarda un sol byte (IR_TMR0) enlloc dels dos bytes d'un codi (IR_DataH i IR_DataL).

```
;----- ENCUA ---------------------------------------------------------------------
;
;       Encua IR_TMR0 a la cua. Si aquesta està plena no encua i surt
;
;-------------------------------------------------------------------------------

_Encua          movlw   0x60                    ;Comprova si la cua esta plena (96 bytes)
                subwf   Bfr_NBytes,W
                btfss   STATUS,Z
                goto    _EOK
                bsf     O_FULL                  ;Indica amb el led si buffer ple
                return

_EOK            movfw   Bfr_Pointer             ;Busca la posicio a afegir la dada
                addwf   Bfr_NBytes, W           ;       Pos = (Pointer + Nbytes)
                btfsc   STATUS,C
                addlw   0xA0                            ;Si sobrepassa 0xFF, suma 0xA0
                movwf   FSR                     ;Passa la posicio al apuntador
                movfw   IR_TMR0
                movwf   INDF                    ;Guarda IR_TMR0 a @FSR
                incf    Bfr_NBytes,F            ;Ja tenim un byte mes
                return
```

_Desencua

Funció complementaria a l'anterior. Igualment passa a treballa amb un sol byte.

```
;----- DESENCUA ---------------------------------------------------------------
;
;      Desencua S_DataH i S_DataL de les posicions Pointer i posterior. No fa res
;    si NBytes = 0.
;
;------------------------------------------------------------------------------

_Desencua
               bcf     O_FULL
               movfw   Bfr_NBytes            ;Si no hi ha res encuat plega
               btfsc   STATUS,Z
               return

               bcf     INTCON,GIE            ;Deshabilita interrupcions
               movfw   Bfr_Pointer           ;Desencua per l'apuntador
               movwf   FSR
               movfw   INDF
               movwf   S_DataH               ;Treu DataH
               decf    Bfr_NBytes,F          ;Tenim una dada menys..
               incf    Bfr_Pointer,F
               incf    FSR,F                         ;..i apuntem una pos. mes amunt
               movfw   INDF
               movwf   S_DataL                       ;Ja tenim dataL
               decf    Bfr_NBytes,F          ;i un byte menys a la cua
               incf    Bfr_Pointer,F         ;apuntem una pos mes amunt..
               bsf     INTCON,GIE            ;Habilita interrupcions

               btfss   STATUS,Z              ;Si apuntem a pos 0...
               return
               movlw   0xA0                  ;...passem a apuntar al inici (0xA0)
               movwf   Bfr_Pointer
               return
```

7.2.2.- Programa Test_IR.bas

Pantalla principal

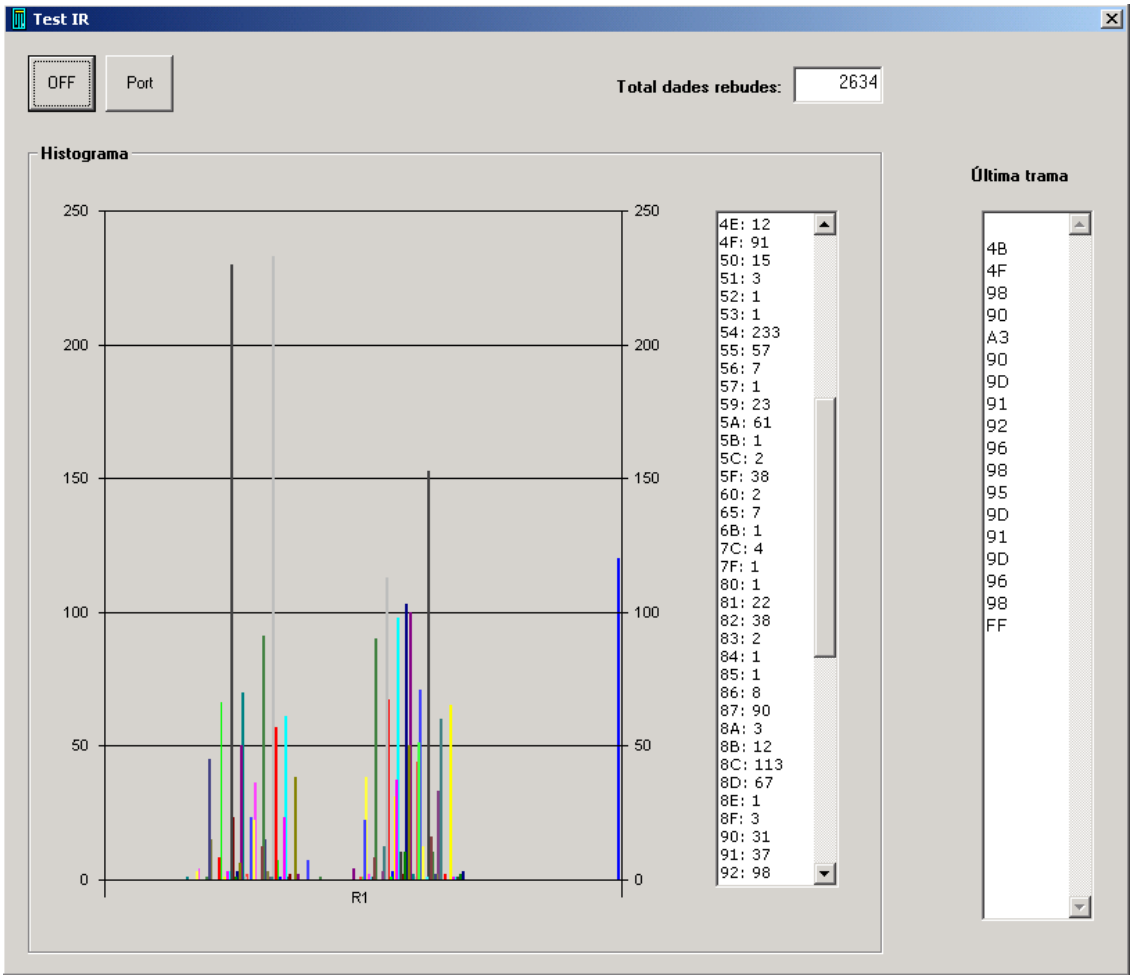


Fig 7.6: Pantalla principal Test_IR

Utilització

Aquest és un programa molt bàsic que utilitza un cotrol MSChart per guardar i mostrar les dades rebudes. Mostra la taula de freqüència de les dades rebudes i la seva representació gràfica. També mostra en la columna situada a la dreta, les dades corresponents a la última trama IR rebuda.

A la part superior s'informa del nombre de bytes rebuts i disposem de botons per configurar el port sèrie utilitzat i parar la recepció de dades.

En el cas ideal, en que la senyal rebuda fos idèntica a l'enviada, la taula de freqüències contindria únicament els valors 0x4D i 0x9A. Recordem que l'especificació de la trama IR utilitza espais entre flancs de 30.8µs i 61.6µs i el timer està configurat amb un preeescaler de 2. Per tant, cada increment del timer correspon a 0.4µs, i podem fer el càlcul:

Valor Timer = Temps entre flancs / Temps increment timer

Valor Timer = 30.8/0.4 = 77 = 0x4D (espais de 30.8µs)
Valor Timer = 61.6/0.4 = 144 = 0x9A (espais de 30.8µs)

Però com ja s'ha dit, el senyal de sortida del detector té unes toleràncies molt amplies, que es reflexen en aquesta dispersió de les dades rebudes. Aquestes formen clarament dos grups, però no tant separats com seria desitjable.

Finalment, i despres de realitzar multitud de proves, els intèrvals utilitzats en el programa són:

Espais de 30.8µs: [0x3D – 0x5D]
Espais de 61.6µs: [0x7A – 0xBA]

Codi

```
'frmPrincipal
'============

Option Explicit

Dim TotalLectures As Long



Private Sub cmdOnOff_Click()
    Dim i As Integer

    'Tanquem el port
    If ctrSerie.PortOpen = True Then
        ctrSerie.PortOpen = False
        cmdOnOff.Caption = "ON"
    'Obrim el port
    Else
        cmdOnOff.Caption = "OFF"
        TotalLectures = 0
        For i = 1 To 256
            MSChart1.Column = i
            MSChart1.Data = 0
        Next
        ctrSerie.PortOpen = True
    End If
End Sub

Private Sub cmdPort_Click()
    frmPort.Show vbModal, Me
End Sub

Private Sub ctrSerie_OnComm()
    Dim s As String
    Dim i, j As Integer

    'Mentre quedi alguna cosa en el port, ho va llegint
    While ctrSerie.InBufferCount > 0
        s = ctrSerie.Input  'Lleguim una cadena d'un caràcter (propietat InputLen)
        i = Asc(s)          'i = valor ASCII del primer (i únic) caràcter de s

        'Afegim valor al llistat de la trama
        txtBytes.Text = txtBytes.Text + vbCrLf + Hex(i)

        'Si el valor no es zero (indica inici de trama), inserim valor al gràfic
        If i <> 0 Then
            MSChart1.Column = i
            MSChart1.Data = MSChart1.Data + 1
```

```
            TotalLectures = TotalLectures + 1
            txtHistograma.Text = ""

            'Actualitzem histograma
            For j = 1 To 255
                MSChart1.Column = j
                If MSChart1.Data <> 0 Then
                    txtHistograma = txtHistograma + Hex(j) & ":" + Str(MSChart1.Data) +
vbCrLf
                End If
            Next
        Else
            txtBytes.Text = ""
        End If

        txtNLectures.Text = Str(TotalLectures)
    Wend
End Sub

Private Sub Form_Load()
    cmdOnOff.Caption = "ON"
    MSChart1.ColumnCount = 256
End Sub
```

```
'frmPort
'=======

Option Explicit

Private Sub cmdOk_Click()
    Dim EstatPort As Boolean

    'Tanca el port (si cal) i selecciona el port marcat amb l'option
    With frmPrincipal.ctrSerie
        EstatPort = .PortOpen
        If EstatPort = True Then .PortOpen = False
        If OptSerie(0).Value = True Then
            .CommPort = 1
        Else
            .CommPort = 2
        End If
        'Si anteriorment el teniem obert, l'obre.
        If .PortOpen <> EstatPort Then .PortOpen = EstatPort
    End With
    Unload Me
End Sub

Private Sub Form_Load()
    'Marca l'Option corresponent al sèrie utilitzat actualment
    OptSerie(frmPrincipal.ctrSerie.CommPort - 1).Value = True
End Sub
```

8.- CONCLUSIONS I EVOLUCIÓ

8.1.- CONCLUSIONS

Són diversos els fets que m'han cridat l'atenció i que considero que cal mencionar dins el projecte. En aquest sentit no crec rellevants qüestions puntuals relatives a la tecnologia utilitzada o al seu estat actual, però sí a questions relacionades amb la metodologia..

Una d'elles és que el coneixement del destí del projecte ha estat molt important en la seva realització. Aquest projecte ha estat realitzat per a una aplicació concreta com és l'automatització d'una subhasta de peix. I per saber l'importància de certs factors ha estat vital veure i conèixer el funcionament de subhastes reals i els seus usuaris. Passar d'un mètode manual a un automatitzat sol provocar una certa desconfiança en els usuaris, desconfiança que cal eliminar. Aquests, per exemple, valoren molt positivament la sensació d'inmediatesa que aporta un temps reduït entre la polsació d'un polsador i l'aparició per pantalla del seu nom. Tècnicament és una característica més, que no millora significativament el funcionament, però potser ha estat la més valorada. També, conèixer en detall les condicions dels llocs a on s'instal·laria ha estat el que ha provocat la inclusió en el receptor de mètodes per comprovar amb facilitat el seu funcionament, com els leds i l'ordre de test.

En un sentit més tècnic cal destacar l'importància de la cerca de documentació que he necessitat per poder-lo realitzar. La pedra angular del projecte ha estat la selecció dels components optoelectrònics (led i receptor IR) i l'especificació de la trama IR. Són dos factors totalment dependents, que no coneixia en absolut i que ha calgut estudiar amb deteniment. De la mateixa manera ha estat necessari conèixer l'oferta existent abans de seleccionar un microcontrolador adient. En tot cas puc afirmar que, literalment, és un temps ben invertit. No hi ha res més problemàtic que basar-se en fonaments dèbils.

Possiblement, la sorpresa més important ha estat el temps necessari per portar-lo a terme. No tant per la durada en sí com pel seu repartiment. Si abans de començar-lo, pensava que la programació dels microcontroladors s'enduria la major part del temps, la realitat ha estat molt diferent. Com ja s'ha comentat, la cerca de documentació prèvia ha estat important, i el procés de testeig i depuració ha resultat lent. Ha obligat a la creació de programes específics per comprovar el correcte funcionament de comandaments i receptor i per comprovar les mesures fetes per aquest. Són feines, necessàries i previstes, però que en realitzat menyspreava abans de començar.

En realitat, el fet de tenir que buscar una gran quantitat de documentació, conèixer noves eines, barrejar electrònica i programació en alt i baix nivell, ha resultat molt interessant. I ha estat precisament aquesta diversitat d'aspectes la que m'ha permés comprovar l'importància d'una bona planificació i preparació abans, o més ben dit, en les primeres fases d'un projecte.

8.2.- MILLORES PROPOSADES

Independentment de que considero que s'han assolit els objectius del projecte, penso que hi han aspectes que són susceptibles de millora. Els explico breument a continuació:

Distància entre repeticions

Si s'accionen dos comandaments en un temps menor o igual al temps d'enviament d'una trama IR (1 milisegon), els dos codis enviats s'interfereixen i seran descartats pel receptor.

Una possible solució és modificar el retard que es realitza entre cada enviament del codi perquè sigui aleatori i múltiple del temps de trama. És una solució similar a la utilitzada per les comunicacions Ethernet en cas de col·lisió en l'accés al medi, però aplicada en tots els enviaments.

L'única dificultat pot venir per aconseguir un nombre aleatori en el microcontrolador. Utilitzar el temps que dura la pulsació del botó (que depèn del usuari) implica esperar a deixar de polsar el botó abans d'enviar la trama, i pot resultar molest per l'usuari.

Alimentació Mando

L'utilització del regulador LM2936 en els comandaments per alimentar el microcontrolador és bona però relativament cara, suposa aproximadament un 20% del cost en components. Aquest percentatge, però, baixa a un 5-10% si comptem la resta de components (placa, caixa i teclat) i encara baixa més si comptem el cost de montatge.

Possiblement es pugui substituïr per un circuit d'alimentació basat en un diode zèner i altres components discrets. Caldria però, que es verifiqués el baix consum en repòs.

Programació en C del receptor

En el projecte s'ha programat l'emissor i el receptor en ensamblador. Possiblement el receptor, menys crític amb els temps, podria programar-se en part amb el llenguatge C per facilitar la creació de noves funcionalitats (actualment no previstes). La rutina d'interrupció que detecta els flancs de la senyal del detector, molt crítica amb el control del temps, s'hauría de mantenir programada en ensamblador.

BIBLIOGRAFIA

[1] Hecht, Eugene
ÓPTICA
Addisson Wesley Iberoaméricana. Madrid, 2000
ISBN: 84-7829-025-7

[2] Bueche, Frederick J.
FÍSICA GENERAL
McGraw – Hill Interamericana editores
ISBN: 970-10-3455-4

[3] Vishay Semiconductors
PHISICS AND TECHNOLOGY
Document Number: 80086 Rev 1.2

[4] Vishay Semiconductors
DATA FORMATS FOR IR REMOTE CONTROL
Document Number: 80071 Rev A2

[5] ATMEL
AVR410: RC5 REMOTE CONTROL RECEIVER
Document Number: 1473B-AVR-05/02

[6] Microchip
PIC16F627A/628A/648A DATA SHEET
Document Number: DS40044D

[7] Microchip
MPASM USER'S GUIDE
Document Number: DS33014G

[8] Joyanes Aguilar, Luis
MICROSOFT VISUAL BASIC 6.0: INICIACION Y REFERENCIA
McGraw – Hill Interamericana editores
ISBN: 84-4812-428-6