Universitat de Vic
Escola Politècnica Superior

Treball Final de Carrera

Software per al cronometratge de temps mitjançant sensors digitals

Albert Marti Serrano

Enginyeria Tècnica Industrial, especialitat en electrònica industrial

Director: Moisès Serra

Vic, setembre de 2008

ÍNDEX

Universitat de Vic
Escola Politècnica Superior

Resum de Treball Final de Carrera

Enginyeria Tècnica Industrial, especialitat en electrònica industrial

Títol: Software per al cronometratge de temps mitjançant sensors digitals
Paraules clau: Cronometratge en temps real, Slot, Comunicació USB
Autor: Albert Marti Serrano
Direcció: Moisès Serra
Data: Setembre de 2008

Resum

L'Slot, conegut per tots amb el nom d'Scalextric, s'ha implantat com a una forma d'oci habitual, la pràctica del qual no queda restringida als més petits, sinó que cada vegada crea més afició entre els grans.

El fet que l'Slot s'hagi extès entre els adults n'ha revolucionat la pràctica. L'entrada al mercat de l'Slot de gent adulta, i amb poder adquisitiu molt superior als adolescents, ha provocat que les marques especialitzades vagin evolucionant els seus productes cada vegada més.

Totes les marques s'han vist obligades a desenvolupar vehicles més competitius i alhora treure al mercat accessoris que augmentin la realitat del joc. Una de les necessitats que s'ha creat és la de competir entre jugadors. Aquesta competició tan pot ser en forma de carrera entre diversos participants, com de forma individual, cronometrant el temps de cada participant en un circuit.

L'objectiu principal del projecte és crear un sistema capaç de realitzar cronometratges en temps real mitjançant sensors digitals ja existents en el mercat de l'Slot i poder controlar i visualitzar la informació des d'un PC.

Per a poder captar els senyals dels sensors s'ha utilitzat un sistema microcontrolat, que garanteix gran velocitat d'adquisició, processament de dades i transmissió.

La comunicació del Microcontrolador amb el PC s'ha realizat mitjançant el bus USB. El PC serà el controlador del sistema i donarà les ordres al Microcontrolador, podent així tenir control total sobre el funcionament del programa. També serà el PC el que tractarà els crocometratges enregistrats i els mostrarà per pantalla.

Universitat de Vic
Escola Politècnica Superior

Senior Thesis

Industrial Technical Engineering, specialized in industrial electronics

Title: Timing software using digital sensors
Keywords: Real-time timing, Slot, USB communication
Author: Albert Marti Serrano
Headship: Moisès Serra
Date: September 2008

Abstract

The Slot, known to everyone with the name of Scalextric, has been introduced as a usual leisure practice not only restricted to the youngest, but also increasingly creating more fans among the grown-ups.

The fact that the slot has been widespread among adults has revolutionized its practice. The launching of slot for adult people, who have much higher purchasing power than adolescents, has caused that the specialized brands keep evolving their products.

All brands have been forced to develop more competitive vehicles and simultaneously put on the market accessories that enhance the reality of the game. One of the needs that has been created is to compete among players. This competition can be either in the form of a race between several participants, or on an individual basis, timing each participant alone in a circuit.

The main objective of the project is to create a system capable of performing real-time timing by means of digital sensors already on the slot market and being able to control and display information using a PC.

In order to receive the sensors' signal, a microcontrolled system has been used, which guarantees high speed in reception, data processing and transmission. Communication from the Microcontroller to the PC has been achieved by a Universal Serial Bus (USB). The PC will be the system controller and will give orders to the Microcontroller, thus having total control over the functioning of the program. The PC will also deal with the recorded timing, displaying them on screen.

1. INTRODUCCIÓ

1.1 INTRODUCCIÓ

Es vol realitzar un sistema que permeti el cronometratge en temps real mitjançant sensors digitals. Tot i que ha de ser compatible per a qualsevol competició, es vol encarar el projecte al món de L'*Slot*. Hi ha molts aficionats al món de l'*Slot* que disposen de petits aparells amb sensors que capturen el temps i el mostren per un petit LCD. Es tracta de que amb una petita modificació dels aparells es puguin aprofitar els sensors d'aquests com a sensors del projecte i així aprofitar els avantatges que això comporta, com són el poder visualitzar els temps de cronometratge en temps real en una pantalla de l'ordinador personal (PC), la possibilitat de guardar tots els temps realitzats amb les modificacions que s'han anat fent i la consulta de dades que s'han anat guardant.

La idea és realitzar un sistema que sigui capaç d'adquirir dades a gran velocitat i que alhora les mostri d'una manera molt visual. Per això s'ha cregut convenient dividir el projecte de la següent manera:

La part de software, dividida en dues parts :

- La programació d'un microcontrolador (µc) que permeti captar els senyals dels sensors digitals, la realització del cronometratge i la transferència de dades cap al PC mitjançant una connexió USB.
- El desenvolupament d'un software amb base Visual Basic que permeti la connexió amb el PIC, tan per rebre dades d'aquest com per transmetre-li paràmetres. També ha de permetre l'emmagatzematge i consulta dels temps de cronometratge realitzats.

La part de hardware, que serà una placa on hi situarem el PIC. En aquesta placa hi hauran un parell de connectors que ens serviran per connectar-hi els sensors i el cable USB, a part de tots els components convencionals per a fer funcionar el PIC. En el *Diagrama 1* es pot veure el conjunt de blocs que formarien el nostre sistema.

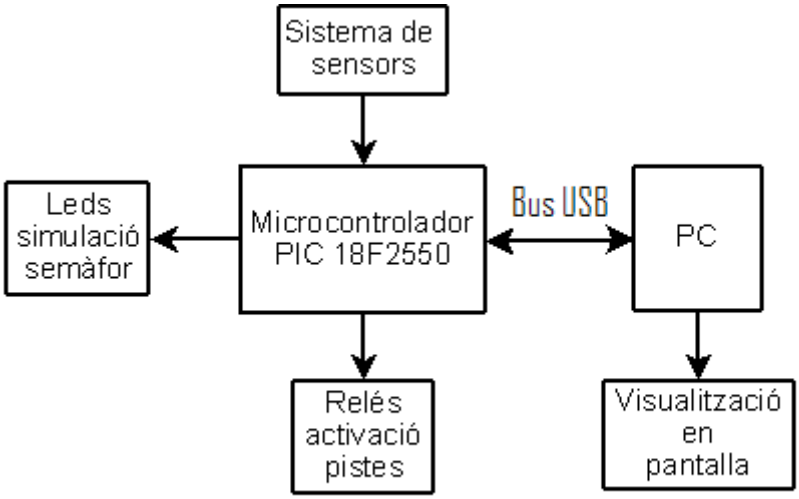


Diagrama 1: Conjunt de blocs que formen el sistema

1.2 OBJECTIUS

Perquè el sistema compleixi amb les especificacions desitjades, s'han plantejat els següents objectius:

- Implementar una estructura *Mestre-Esclau* on el PC sigui el *Mestre* i el Microcontrolador l'*Esclau*.

- Per a la comunicació entre el PC i el Microcontrolador, s'utilitzarà el Port de Comunicacions USB.

- El sistema de control del PC ha de ser molt visual i de fàcil ús.

- El PC ha de tenir el control de les alimentacions de les pistes.

- Els sensors que permetran comptabilitzar les voltes i el temps s'han d'adaptar amb facilitat a la pista.

- El sistema de cronometratge ha de permetre visualitzar els cronometratges en temps real.

- El sistema ha de permetre poder guardar els cronometratges fets en un registre i també ha de permetre consultar els cronometratges ja realitzats.

1.3 PLA DE TREBALL DEL PROJECTE

El pla de Treball a seguir per arribar als objectius marcats és el següent:

- Buscar un Microcontrolador que s'adeqüi a les necessitats del projecte i que disposi de port de comunicacions USB.

- Elecció dels softwares i hardwares:

    - Software per a l'edició i compilació del codi pel Microcontrolador, i també l'elecció del hardware per a la gravació.

    - Elecció del software que s'utilitzarà en el PC per comunicar-se amb el Microcontrolador.

    - Elecció dels softwares que s'utilitzaran per a dissenyar l'esquema del connexionat.

- Disseny del connexionat de la placa.

- Edició del codi tant pel Microcontrolador com pel PC.

- Instal·lació dels drivers en el PC i el Microcontrolador per a poder transmetre dades a través del port de comunicacions USB.

- Adaptar un sistema de sensors per a poder captar el pas dels cotxes.

1.4 ESPECIFICACIONS TÈCNIQUES DEL PROJECTE

En aquest apartat s'introduirà alguns aspectes tècnics com l'alimentació, el consum i el tipus de comunicació del projecte.

El sistema està dissenyat per funcionar amb l'alimentació que proporciona el bus USB. L'alimentació del Microcontrolador, els leds, el sistema de sensors i els relés es realitzarà via el Bus USB. Com es veurà més endavant, el Bus USB subministra una tensió de +5v i un corrent màxim de 500mA, per tant, un dels requeriments del sistema és que funcioni a una tensió de +5v i que consumeixi com a màxim 500mA.

Per tal de poder gravar el Microcontrolador s'ha hagut de dissenyar una font d'alimentació. El gravador MPLAB ICD2 consumeix més de 500mA i necessita una alimentació externa, ja que el bus USB no els pot subministrar. En l'apartat de disseny de la placa es pot trobar l'esquema de connexionat de la Font d'alimentació.

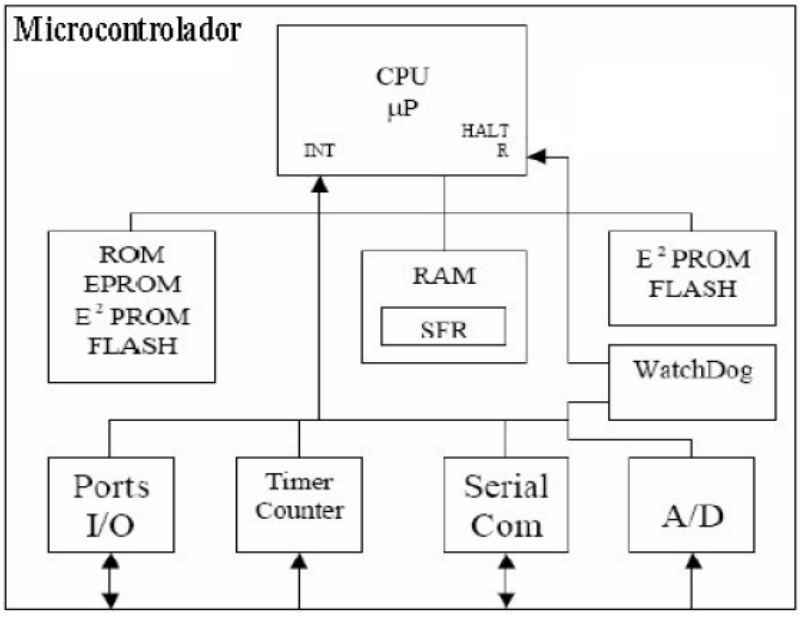
El tipus de comunicació que s'utilitzarà serà USB, i es transmetran les dades utilitzant el Mode de transmissió USB CDC. Més endavant ja es veurà quines són les particularitats d'aquest tipus de transmissió i quines funcions s'han d'implementar per a realitzar-la.

2. FONAMENTS TEÒRICS

Aquest capítol comença introduint el concepte de Microcontrolador, es veuen quines són les parts més importants i les seves funcions. Seguidament s'enumeren les característiques principals de les que disposa el Microcontrolador 18F2550. A continuació s'introdueix el Protocol USB, parlant de les seves característiques principals, versions del Bus USB, funcionament físic i els tipus de transmissions de dades. El següent punt introdueix el concepte VID&PID, que ja es veurà que és del tot necessari per a connectar el Microcontrolador amb el PC. A continuació es definirà la trama de caràcters utilitzada per a enviar informació a través del bus USB. Finalment, s'expliquen els Modes de Funcionament que tindrà el sistema i les ordres que s'utilitzaran per a identificar cada Mode.

2.1 INTRODUCCIÓ ALS MICROCONTROLADORS

Els Microcontroladors són circuits integrats que inclouen com a mínim una CPU, unitats d'entrada i sortida i memòria de programa, a més d'una sèrie de característiques que fan que s'adaptin a projectes en concret. A la *imatge 1* es pot veure el que seria l'esquema intern d'un Microcontrolador.



Imatge 1: Blocs interns d'un Microcontrolador

2.1.1 Definició de les parts més importants d'un Microcontrolador:

Memòria de Programa – Els microcontroladors necessiten un programa que marqui les ordres a executar. Aquesta memòria pot ser de diferents tipus: ROM, EPROM (es pot esborrar amb UV a la finestra), EEPROM (es pot esborrar amb 12v), FLASH (s'esborra en funcionament i amb la mateixa tensió).

Memòria de dades – A part de la memòria del programa també és necessària una memòria per a poder guardar dades per al funcionament del programa. Aquesta memòria pot ser volàtil (per a variables i necessitats puntuals del programa) o no volàtil (per a dades que es necessitin guardar si falla l'alimentació).

Entrades/Sortides – per a poder captar canvis de l'exterior i actuar, es disposa d'una sèrie de pins d'entrada/sortida. Depenent del Microcontrolador aquets pins es podran configurar de diferents maneres.

Timers i comptadors – Els timers són controladors de temps que es poden utilitzar en el programa i que es poden incrementar per cicles de rellotge intern o per un rellotge extern. Els comptadors són elements que guarden el nombre de pulsacions d'una entrada asíncrona exterior.

Watchdog – Aquest dispositiu és un temporitzador programable especial, la funció del qual és realitzar un reset del sistema si no el recarreguem i es desborda el seu temps, d'aquesta manera podem vigilar que el nostre sistema no hagi entrat en una part incontrolada.

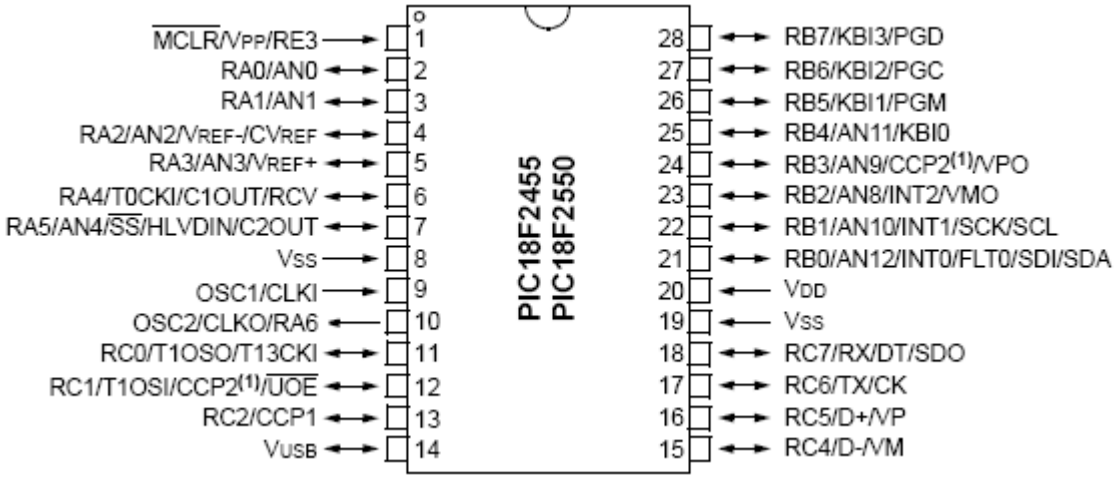
Convertidor A/D – els microcontroladors només poden tractar dades en forma digital. Per entrades analògiques i per poder captar el nivell de tensió d'un element, s'incorpora un convertidor de nivell digital.

Interrupcions –Serveixen per a tractar canvis d'estats que es produeixen durant l'execució del programa.

Comunicacions – Per a poder interactuar amb l'exterior s'incorporen varis mòduls per poder rebre i enviar dades. Alguns dels dispositius de comunicació són els següents: SPI (Serial Peripherical Interface), I^2C (Inter-Integrated Circuit), UART (Universal Asynchronous Receiver-Transmiter), USB (Universal Serial Bus), CAN (Controller Area Network).

2.2 CARACTERÍSTIQUES DEL MICROCONTROLADOR PIC18F2550

Seguidament es troben detallades les principals característiques del Microcontrolador. En la *imatge 2* es pot veure la distribució dels pins del Microcontrolador.



Imatge 2: Distribució de pins del Microcontrolador

Característiques principals

Arquitectura: 8 bits

Tipus de Memòria: Flash

Memòria Programa: 32 KByte

Memòria EEPROM: 256

RAM: 2048

I/O Pins: 24

Max. Velocitat de la CPU: 48 MHz

Port de comunicacions USB 2.0 màxima velocitat

Port de comunicació sèrie

Master I^2C

Timers: 1 de 8-bits, 3 de 16-bits

Té 3 Interrupcions externes situades a RB0, RB1 i RB2

Disposa de 4 entrades analògiques situades a AN0, AN1, AN2 i AN3

Detecció de canvis d'estat de RB4 a RB7

Canals de conversió analògic digital (ADC)

2 comparadors

Watchdog Timer (WDT). Programable des de 41ms a 131s.

Prioritat per les interrupcions

2.3 PROTOCOL USB

El protocol USB (Universal Serial Bus) ha estat una veritable revolució en el món de la computació, característiques com senzillesa en la connexió i velocitat de transmissió han fet massiu l'ús d'aquesta tecnologia, deixant enrere antigues formes de connectar perifèrics pels ports paral·lels o el sèrie RS-232. L'USB té el gran avantatge que és un estàndard obert, la qual cosa permet anar millorant el protocol segons les noves necessitats que vagin sortint en el mercat, a més ja ha estat adoptat per centenars de fabricants de perifèrics i ha rebut una gran acceptació entre els fabricants de computadors personals. Cada nou PC que surt al mercat inclou ports USB que poden connectar automàticament als dispositius estàndard tals com: teclats, ratolins, escàners, impressores, etc., o també dispositius amb software personalitzat per a qualsevol tipus de propòsit. Un *host* controlador és l'encarregat de controlar i dirigir totes les comunicacions amb els dispositius USB.

2.3.1 CARACTERÍSTIQUES GENERALS DEL PROTOCOL USB

L'USB va ser dissenyat expressament per a proporcionar les característiques més requerides pels usuaris, les principals són:

- Una interfície per a molts dispositius: L'USB és prou versàtil per a ser utilitzat en una varietat de perifèrics. En lloc de tenir un tipus de connector diferent per a cada dispositiu i tenir un suport per a cada maquinària, una interfície en USB serveix per a tots.

- Configuració automàtica per a classes conegudes: Quan un usuari connecta un dispositiu USB, el sistema operatiu detecta el perifèric i carrega el programari apropiat. Si una classe desconeguda es connecta, els sistemes operatius com Windows adverteixen a l'usuari per inserir un disc amb els drivers, però a part d'això, la instal·lació és automàtica.

- Simple connexió: Els connectors del cable USB són com una clau, així que no és possible poder connectar incorrectament el dispositiu. L'abast pot arribar fins als 5 metres. Amb hubs, un dispositiu pot arribar fins als 30 metres de distància des del host base.

- No requereix alimentació externa: La interfície USB inclou subministrament d'energia a través de la línia de terra i els +5V nominals lliurat pel computador o el hub. Un dispositiu pot prendre fins a màxim 500mA a través del bus. Pel contrari, els dispositius que usen interfícies en el qual requereixin més corrent, poden incloure un subministrament d'energia dintre del dispositiu o usar un subministrament extern.

- Connectivitat: Fins a 127 dispositius diferents poden estar connectats simultàniament i operant amb el mateix computador.

- Baixos costos de disseny: Per a gairebé la majoria de les aplicacions que són estàndard i no tenen una configuració pròpia, els costos de disseny són comparativament menors que altres tecnologies.

Qualsevol pot desenvolupar drivers, programes i dispositius USB sense pagar alguna tarifa de llicències, no obstant això qualsevol que distribueix un dispositiu amb una interfície USB ha d'obtenir els drets per a usar un Vendor ID. Actualment, el càrrec administratiu per a obtenir un Vendor ID és de $1500 i una quota anual de $2500. Aquestes quotes no són problemes per a desenvolupadors de productes amb volums alts de dispositius, sinó més aviat pot ser un impediment per als desenvolupadors que planegen realitzar quantitats petites de dispositius i a baixos costos.

En l'actualitat existeixen quatre versions diferents d'USB. Aquestes han aparegut principalment per la necessitat d'anar millorant la velocitat de la transmissió sèrie i també per les noves formes d'interconnectar dispositius.

2.3.1.1 USB 1.1

Aquesta especificació va ser realitzada per les empreses Compaq, Intel, Microsoft i NEC, el setembre de 1998 i marcava la primera versió final d'una sèrie d'antigues versions. La versió USB 1.1 defineix i caracteritza gairebé per complet el que és avui dia el protocol USB, doncs en ella s'especifica un canal sèrie per a suportar una gran gamma de perifèrics de mitjana i baixa velocitat, amb suport integral per a transferències en temps real com veu, àudio i vídeo. En la *imatge 3* es pot veure el logotip d' USB 1.1.

Imatge 3: Logotip USB 1.1

2.3.1.2 USB 2.0

USB 1.1 va guanyar gran popularitat entre els usuaris i dissenyadors de productes. L'abril de l'any 2000, en conjunt amb noves empreses per al millorament del protocol com Hewlett-Packard, Lucent i Philips, es publica finalment USB 2.0, el logotip del qual es mostra en la *imatge 4*. Això demostrava el gran interès per a desenvolupar i integrar nous dispositius més ràpids i que suportessin gran quantitat de dades de transferències. La nova velocitat que s'incorpora es diu high-speed i és de 480 Mbps, que és quaranta vegades més ràpida que l'antiga fullspeed. La incorporació de high-speed va fer que l'USB fos molt més atractiu per al disseny de dispositius com impressores, escàners, càmeres fotogràfiques i pendrives.

Imatge 4: Logotip USB 2.0

2.3.1.3 USB OTG

Els desenvolupadors de perifèrics van començar a sol·licitar una nova forma per a poder connectar els dispositius USB. Per exemple, un usuari podria voler usar una impressora directament amb una càmera fotogràfica. El protocol *On-The-Go*, el logotip del qual es mostra en la *imatge 5*, va ser el que va fer possible la connexió demanada pels fabricants. Es va publicar el desembre del 2001 i correspon a una variació de l'USB 2.0. Amb aquest nou protocol el host té una capacitat limitada, però possibilita la comunicació entre perifèrics i elimina la indispensabilitat de tenir un PC per a establir la comunicació. Igualment, USB OTG permet a un dispositiu actuar com a servidor o com a client, depenent de com originalment es va connectar el cablejat. Fins i tot després de que les unitats s'estiguin comunicant, els dos dispositius poden canviar el rol sota el control d'un programa. Aquesta facilitat està específicament dissenyada per a dispositius com PDA, càmeres i impressores.

Imatge 5: Logotip USB Ondejo 2.0.

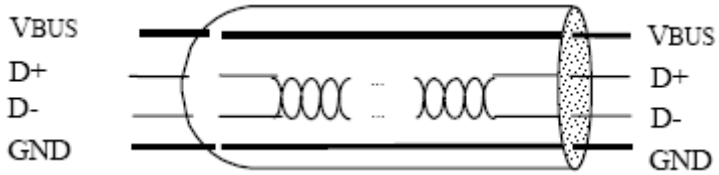
2.3.1.4 USB Wireless

L'USB sense fil (WUSB) és la tecnologia desenvolupada més recent per USB-IF. La versió 1.0 de WUSB va ser publicada el maig del 2005 i dóna més facilitats i mobilitat als dispositius que es connecten als PC. Aquesta especificació manté el mateix ús i arquitectura que l'USB amb cables, amb una connexió high-speed des del host al dispositiu. La utilització de WUSB no ha estat aprovada encara per una bona quantitat d'organismes reguladors europeus i asiàtics, a causa de potencials conflictes amb altres tecnologies sense fil. Això podria enfosquir les perspectives del Wireless USB d'arribar a economies d'escala que puguin reduir costos i competir amb Bluetooth, que ha arribat a un important creixement en el camp de la tecnologia sense fil durant els últims anys. En la *imatge 6* es pot veure el logotip de l'USB wireless.

Imatge 6: Logotip USB Wireless.

2.3.2 Funcionament del Bus USB vist des del Nivell Físic

La interfície USB es connecta amb l'equip de l'usuari a través d'un cable de quatre fils. Dos són dedicats a l'alimentació i els dos restants al senyal de dades (D+ i D-), aquests últims estan com parell trenat. Els conductors d'alimentació (*imatge 7*) VBus i GND lliuren una tensió contínua de 5V i 500mA com a màxim. En funció de les necessitats d'alimentació elèctrica dels dispositius, aquests poden prendre l'alimentació d'aquestes línies, o bé tenir una font d'alimentació alternativa.

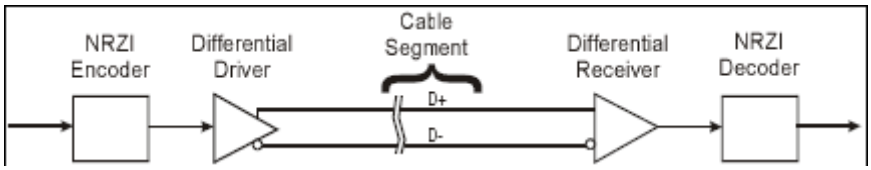


Imatge 7: Cable de quatre fils USB.

2.3.3 Transmissió

El bus USB és síncron i els paquets de dades estan codificats usant NRZI . En la *imatge 8* es pot veure la transmissió de dades mitjançant el bus USB.

NRZI - Non Return to Zero Inverted, la línia canvia de nivell si es transmet un “0” i no canvia si es transmet un “1”.



Imatge 8: Transmissió de les dades al cable USB.

2.3.4 TIPUS DE TRANSFERÈNCIES USB

L'USB va ser dissenyat per a manipular diferents perifèrics amb diversos requisits com ara velocitat de transferència, temps de resposta, grandària de dades enviades i correcció d'errors. Els quatre tipus de transferències que existeixen en USB compleixen amb les necessitats que es van nomenar. Les transferències són: Control, Interrupció, Bulk i Isocròna. Un dispositiu pot integrar els tipus de transferències que més li convinguin i satisfacin, segons siguin els seus propòsits, sent el del tipus Control l'únic d'ús obligatori per als dispositius USB.

2.3.4.1 Interrupció

És utilitzat per aquells dispositius que desitgen transferir molt poca informació i poc freqüent, assegurant la lectura de les dades dins d'un període màxim de temps. Interrupció té la particularitat de ser unidireccional per cada endpoint que es defineixi (des del host al dispositiu o des del dispositiu al host).

2.3.4.2 Bulk

La transferència Bulk és una comunicació no periòdica, típicament emprada per transferències que requereixen usar tot l'ample de banda disponible o esperar fins que l'ample de banda estigui disponible. Això implica moviments d'imatges, arxius o vídeo, on es requereix de gran potencial de transferència en poc temps. Per a aquestes aplicacions, les transferències poden ser molt ràpides però les dades poden esperar si és necessari. Si el bus USB està molt ocupat, llavors les transferències Bulk són lentes, però si el bus en cas contrari està desocupat, llavors les transferències Bulk són molt ràpides.

2.3.4.3 Isòcrona

La transmissió isòcrona ha estat desenvolupada especialment per a satisfer les demandes de transmissió en temps real de veu, vídeo i imatges. La transferència isòcrona proveeix comunicació contínua i periòdica entre el host i el dispositiu, on la informació útil per paquet pot oscil·lar entre 1 i 1,023 bytes. Només els dispositius full i high-speed poden fer transferències isòcrones i per al cas de full-speed s'ha de reservar un ample de banda de fins a un 90 %.

2.3.4.4 Control

Les transferències de control estan definides per a un sol ús: transportar les peticions (request) standard i especifiques definides per l'especificació USB. L'objectiu és facultar al host perquè aquest interrogui i configuri als dispositius connectats en els ports. Per aquesta raó, la intervenció d'aquest tipus de comunicació és obligatòria i només apareix al principi de la connexió d'un dispositiu, per després donar pas a les altres tres transferències possibles.

Un cop vistes les diferents versions d'USB, i les possibilitats de transmissió de dades que ofereix, s'analitzaran la versió USB 2.0 i la modalitat de transmissió BULK que són les que s'utilitzaran pel projecte.

El fet d'utilitzar la versió d'USB 2.0 és una condició ja que el microcontrolador només disposa d'aquesta velocitat de transmissió. Pel que fa a la modalitat de transmissió s'utilitza la BULK, però en una variant anomenada CDC (Communications Device Class).

2.3.4.5 USB CDC

Una comunicació USB entre un Microcontrolador i un PC genera força complicacions. No només es tracta d'editar el codi del microcontrolador amb funcions USB específiques, sinó que també s'ha de configurar i instal·lar el driver en el PC perquè detecti el microcontrolador, s'ha d'utilitzar l'API que connecta amb la DLL que subministra Microchip, que serveix perquè el software instal·lat en el PC pugui connectar amb el driver, i s'ha de programar l'entorn del PC també amb funcions específiques USB.

L'USB CDC és una versió de la transmissió de dades BULK, però que utilitza funcions que disminueixen la dificultat de programació. La característica principal de la transmissió CDC és que té la capacitat de tractar el port USB com si fos un port sèrie virtual. El codi del Microcontrolador s'ha d'editar igualment amb funcions específiques USB, però el gran avantatge es troba a l'hora de programar el PC. Al crear un port sèrie virtual es pot editar el codi del software del PC com si d'un port sèrie es tractés, sense cap diferència respecte una comunicació RS-232, i amb

l'avantatge que es disposa pràcticament de la mateixa velocitat de transmissió que en una transmissió BULK.

La transmissió USB CDC permet enviar i rebre dades a una velocitat de 1.5Mbits/s. En la transmissió BULK es poden arribar a aconseguir 12Mbits/s, però rarament s'aconsegueixen ja que la velocitat depèn molt de l'estructura del codi editat.

2.4 VID&PID (Vendor id & Product id)

Per a poder connectar el Microcontrolador PIC amb el PC a través del port USB, s'ha de fer ús del driver que proporciona la casa Microchip, del qual ja se'n parlarà més endavant. Un nou concepte és el *VID&PID*, sense el qual no seria possible la connexió amb el PC. Aquest VID&PID serà l'encarregat que el sistema operatiu Windows reconegui el nostre Microcontrolador.

El VID, de valor 0x04D8, és la identificació que tenen per defecte tots els productes que disposen de port de comunicacions USB de la casa Microchip. Per tant, quan el sistema operatiu Windows rep un ID amb aquest valor, ja sap que es tracta d'un Microcontrolador de la casa Microchip.

El PID, l'altra variable i de valor 0x000A, és la que identifica el dispositiu en concret. Aquesta variable serveix per a configurar cada Microcontrolador com a únic.

Aquest VID&PID està definit dins la llibreria *rr2_usb_cdc_Monitor.h,* que és la llibreria on estan definits tots els descriptors del Microcontrolador i que es troba en l'arrel del programa.

2.5 TRAMA UTILITZADA PER A ENVIAR DADES DEL PC CAP AL MICROCONTROLADOR I AL REVÉS

Per a poder transmetre dades d'una forma fiable, s'ha creat una trama que utilitzen tan el PC com el Microcontrolador a l'hora de comunicar-se i que permet minimitzar els errors que es puguin produir durant les transmissions. Aquesta trama està formada de la següent manera:

SOH	ID	Nº d'Ordre	Bloc de dades 1	Bloc de dades 2	ETX	CRCH	CRCL
1 car.	1car.	1 car.	4 car.	4 car.	1 car.	1 car.	1 car.

Car. = caràcter

La trama està formada per 14 caràcters. (Seguidament es definirà el significat de cada part de la trama.)

SOH - És un caràcter constant (0x01) que serveix per a inicialitzar la trama. Així doncs la trama sempre ha d'estar encapçalada per aquest caràcter.

ID - També és un únic caràcter i serveix per a identificar el Microcontrolador. De moment, en el projecte només s'hi utilitza un Microcontrolador, però si se n'hi volgués afegir un altre, aquest caràcter serviria per a utilitzar el mateix bus de dades per a enviar informació als Microcontroladors, i només el que tingués la mateixa ID podria rebre la trama.

Nº d'Ordre - És un caràcter que s'utilitza per a enviar l'ordre que ha de realitzar el Microcontrolador. Com ja es definirà posteriorment, el PC envia unes ordres al Microcontrolador que li serveixen per a configurar el mode de funcionament.

Bloc de dades 1 – És un bloc format per 4 caràcters que serveix per a enviar els resultats obtinguts pel Microcontrolador cap al PC. Si fes falta, també és podria utilitzar per enviar informació del PC cap al Microcontrolador. En aquest bloc s'envien les dades en format hexadecimal.

Bloc de dades 2- Igualment que el Bloc de dades 1, és un altre bloc de 4 caràcters que també s'utilitza per a enviar informació en els dos sentits de la transmissió. També s'envien les dades en format hexadecimal.

ETX – És un caràcter constant (0x03) que marca la finalització de la trama.

CRCH i CRCL - Són dos caràcters que serveixen per a verificar que no hi hagi cap error en la transmissió. CRCH (High) és la part alta del CRC i CRCL (Low) és la part baixa del CRC. Funcionen de la següent manera: quan el PC o el Microcontrolador envien una trama, en les dues últimes posicions hi envien el càlcul del CRC, que és la suma de tots els caràcters de la trama, i envien aquest valor en format Hexadecimal. Quan el PC o el Microcontrolador reben la trama, van verificant tots els caràcters i alhora van incrementant el valor del CRC intern. Un cop tots els caràcters han estat rebuts, es converteix el CRC intern a valor Hexadecimal i es compara amb el valor del CRC rebut en les dues posicions finals de la trama. Si coincideixen, la trama rebuda és correcte.

L'exemple següent explica el funcionament de la trama:

S'envia la següent trama:

SOH	ID	Nº d'Ordre	Bloc de dades 1	Bloc de dades 2	ETX	CRCH	CRCL
0x01	1	40	0x0000	0x0000	0x03		

Per a calcular el CRC s'utilitzen les ordres ID, NºOrdre, Bloc1, Bloc2 i ETX. El caràcter SOH només s'utilitza per a marcar l'inici de trama.

Un cop definits tots els valors, es passa a calcular el CRC. Es calcula el seu valor fent la conversió dels valors Hexadecimals a decimals.

Tots els caràcters es poden trobar a la taula de caràcters ASCII que hi ha a l'annex 4.

ID = 1

Nº d'Ordre = 40

Bloc de dades 1 = 0x0000 = 0 en decimal

Bloc de dades 2 = 0x0000 = 0 en decimal

ETX = 0x03 = 3 en decimal

Suma Total = 44

El seu valor en hexadecimal equival a 2C.

Llavors:

- CRCH = 2
- CRCL= C

Per tant, la trama que s'envia queda de la següent manera:

SOH	ID	Nº d'Ordre	Bloc de dades 1	Bloc de dades 2	ETX	CRCH	CRCL
0x01	1	40	0x0000	0x0000	0x03	2	C

Un cop el PC o el Microcontrolador rep la trama, calcula el CRC de la trama que ha rebut i el compara amb el CRCH i CRCL. Si coincideixen vol dir que no s'ha alterat la informació de la trama i que, per tant, la trama rebuda és correcte.

Si la suma Total del CRC és superior a 255, no es disposa de més bits per a comparar el valor total, però com que els dos bits inferiors són els que més canvien si es produeix un error en l'enviament de la trama, les possibilitats de detectar l'error són molt altes.

2.6 MODES DE FUNCIONAMENT DEL SISTEMA

El sistema tindrà quatre modes de funcionament, un mode de comprovació de comunicacions i el mode de parar.

Els modes de funcionament seran:

- Mode Pràctica
- Mode Pràctica Cronometrada
- Mode Cronometratge
- Mode Carrera

Mode Pràctica

En aquest mode només s'activen les pistes i no es cronometra els temps.

Mode Pràctica Cronometrada

En aquest mode s'activaran les pistes i es cronometrarà el temps. No s'establirà un número de voltes a fer ni es donarà la sortida amb un semàfor. El cronòmetre anirà comptant i mostrà per pantalla els temps que es van realitzant. El programa donarà l'opció de guardar els cronometratges realitzats.

Mode Cronometratge

Aquest mode permet el cronometratge individual, només activa una pista o l'altra. S'ha de determinar el número de voltes a realitzar i la sortida la donarà el semàfor. Un cop s'hagin fet les voltes marcades es podran guardar els temps realitzats.

Mode Carrera

Aquest mode permet la competició d'una pista contra l'altra. S'ha de determinar el número de voltes a realitzar i la sortida la donarà el semàfor. Un cop la carrera acabada, també es donarà l'opció de guardar els temps enregistrats de cada pista.

2.7 RELACIÓ D'ORDRES PER AL CONTROL DELS MODES DE FUNCIONAMENT

Per a poder controlar els modes de funcionament vistos en el punt anterior, s'utilitzaran unes ordres de configuració. Aquestes ordres s'envien en la trama de dades, que també s'han vist anteriorment i serveixen per a configurar el programa.

Ordre de Test de Comunicacions

Ordre 00

Aquesta ordre retorna la mateixa trama rebuda. Serveix per a testejar les comunicacions.

Ordres Mode Pràctica

Ordre 10

Aquest mode activa les dues pistes, però no compta els temps.

Ordre 11

Igual que en l'ordre 10, però només s'activa la pista1.

Ordre 12

Igual que en l'ordre 10 però només s'activa la pista2.

Ordres Mode Pràctica Cronometrada

Ordre 20

En aquest mode s'activen les dues pistes i es van comptabilitzant els temps.

Ordre 21

Igual que en l'ordre 20, però només s'activa la pista1.

Ordre 22

Igual que en l'ordre 20, però només s'activa la pista2.

Ordres Mode Cronometratge

Ordre 31

En aquest mode es poden programar les voltes a realitzar. S'utilitza un semàfor per a donar la sortida. Només s'activa la pista1.

Ordre 32

Igual que en l'ordre 31, però activant la pista2 en comptes de la pista1.

Ordres Mode Carrera

Ordre 40

Aquest mode permet realitzar una cursa utilitzant les dues pistes. Es poden programar les voltes de la carrera. S'utilitza un semàfor per a donar la sortida i activar les pistes.

Ordre STOP

Ordre 50

Aquesta ordre serveix per a neutralitzar qualsevol mode entrat. Apaga les alimentacions de les pistes i el Microcontrolador espera nova configuració.

A part de les ordres de configuració, s'utilitzen unes altres ordres que serveixen per a transmetre informació del PC cap al Microcontrolador i al revés.

Ordre 60

Aquesta ordre s'utilitza quan des del microcontrolador es vol avisar al PC que s'ha iniciat un mode de cronometratge. El PC sincronitza el temps i inicialitza el timer. En l'apartat de programació del Microcontrolador ja es veurà com s'envia la informació.

Ordre 70

Aquesta ordre s'utilitza un cop ja s'ha arrencat algun mode de cronometratge. És l'ordre que utilitza el microcontrolador quan vol passar els cronometratges realitzats al PC. En l'apartat de programació del Microcontrolador ja es veurà com s'envia la informació.

Ordre 80

Aquesta ordre la farà servir el PC per a marcar el final de cronometratge o de carrera. Un cop el cotxe ha realitzat les voltes escollides, el PC enviarà una ordre per a parar l'alimentació de la pista. En l'apartat de programació del PC ja es veurà com s'envia la informació.

El *diagrama 2* mostra la relació existent entre modes de funcionament i els Ordres a realitzar.

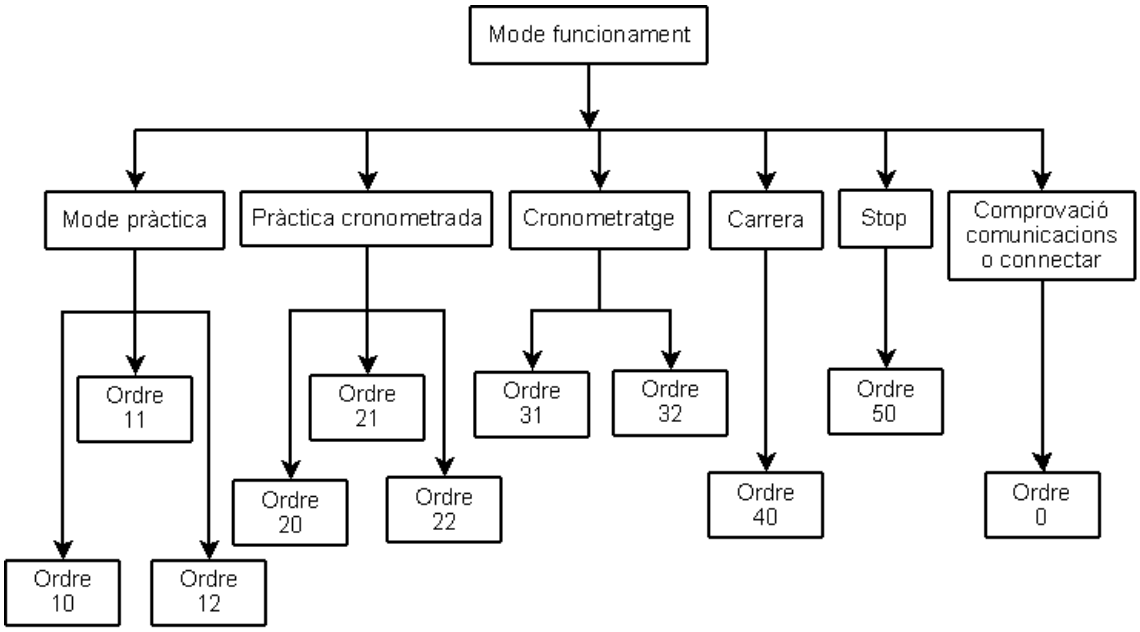


Diagrama 2: Relació Modes Funcionament i Ordres a realitzar

3. EXPOSICIÓ DEL TREBALL

El capítol que segueix està dividit en sis parts. En el primer punt es mostra el procés seguit a l'hora d'escollir el Microcontrolador. El segon punt tracta l'elecció del compilador i del gravador del Microcontrolador. El tercer punt explica el disseny del circuit, totes les connexions realitzades i components utilitzats per a fer funcionar el sistema. En el quart punt es tracta la programació del Microcontrolador, des de la configuració dels registres interns fins al funcionament general del programa. En el cinquè punt s'exposa la programació del PC, en el qual s'ha utilitzat l'entorn Visual Basic 6.0. Finalment, conclou el capítol l'adaptació del sistema de sensors utilitzat per a la captació del pas dels cotxes.

3.1 ELECCIÓ DEL MICROCONTROLADOR

Un dels objectius del projecte era que la comunicació entre el Microcontrolador i el PC fos mitjançant la connexió USB. Un dels motius que van fer que es descartés la comunicació sèrie va ser que avui en dia la gran majoria de PC's, sobretot els portàtils, ja no disposen de port sèrie i només tenen ports USB.

Aquesta condició només permetia dues possibilitats:

- Escollir un Microcontrolador amb port de comunicacions sèrie i afegir un integrat convertidor de comunicacions sèrie a USB.

- Escollir un Microcontrolador que portés incorporat un port de comunicacions USB.

Tot i que la programació de les comunicacions entre un Microcontrolador i un PC via comunicació sèrie hauria resultat menys complicada, es va escollir la opció d'un microcontrolador que incorporés un port de comunicacions USB.

El Microcontrolador escollit és un PIC de la casa Microchip. Microchip ha desenvolupat la sèrie de microcontroladors 18Fxx5x que incorporen port de comunicacions USB. Hi ha bastants models, però el que es va creure que

s'adaptava més al projecte va ser el 18F2550. Aquest microcontrolador és un dels de la sèrie 18F més utilitzat, ja que és bastant complert.

La taula que hi ha a continuació, mostra les diferències entre els Microcontroladors, que serviran per a explicar els motius pels quals es va triar el 18F2550 i no un altre.

PIC	Program Memory KBytes	EEPROM Data Memory	RAM	I/O Pins	Pin count	Digital Communications	Capture/Compare/ PWM Peripherals	Package
18F2450	16	0	768	23	28	0 -UART 1 -A/E/USART 0 -SPI 0 -I2C 0 -MSSP(SPI/I2C)	1 -CCP 0 -ECCP 10-bit PWM resolutions	28/SOIC 300mil 28/SPDIP 28/QFN
18F2455	24	256	2048	24	28	0 -UART 1 -A/E/USART 0 -SPI 0 -I2C 1 -MSSP(SPI/I2C)	2 -CCP 0 -ECCP 10-bit PWM resolutions	28/PDIP 300mil 28/SOIC 300mil 28/SPDIP
18F2550	32	256	2048	24	28	0 -UART 1 -A/E/USART 0 -SPI 0 -I2C 1 -MSSP(SPI/I2C)	2 -CCP 0 -ECCP 10-bit PWM resolutions	28/PDIP 300mil 28/SOIC 300mil 28/SPDIP
18F4450	16	0	768	34	40	0 -UART 1 -A/E/USART 0 -SPI 0 -I2C 0 -MSSP(SPI/I2C)	1 -CCP 0 -ECCP 10-bit PWM resolutions	40/PDIP 44/TQFP 44/QFN
18F4455	24	256	2048	35	40	0 -UART 1 -A/E/USART 0 -SPI 0 -I2C 1 -MSSP(SPI/I2C)	1 -CCP 1 -ECCP 10-bit PWM resolutions	40/PDIP 40/PDIP 600mil 44/TQFP 44/QFN
18F4550	32	256	2048	35	40	0 -UART 1 -A/E/USART 0 -SPI 0 -I2C 1 -MSSP(SPI/I2C)	1 -CCP 1 -ECCP 10-bit PWM resolutions	40/PDIP 40/PDIP 600mil 44/TQFP 44/QFN

Taula comparativa característiques Microcontroladors

En una primera tria es van descartar tots els microcontroladors que no estiguessin disponibles amb el tipus d'encapsulat PDIP. Aquest encapsulat és el més típic, el que solem anomenar convencional i que ens permet, per dimensions dels pins i per la separació que hi ha entre aquests, muntar el microcontrolador en una Protoboard.

Seguidament es van descartar tots els microcontroladors de 40 pins, quedant només amb els de 28 pins, que per tema d'entrades i sortides són suficients pel que es necessita pel projecte.

Un cop aquí, el dubte era escollir entre el 18F2455 i el 18F2550. La diferència entre els dos és que el 18F2550 disposa de més capacitat de memòria de programa. Com que no se sabia quants Kbytes de memòria es necessitarien, i com tampoc no es disposava de molt temps per fer proves, es va decidir escollir el 18F2550, ja que és el que té més capacitat de memòria de programa.

3.2 ELECCIÓ DE L'ENTORN DE PROGRAMACIÓ PER AL MICROCONTROLADOR

Per tal de poder gravar un software en un Microcontrolador, s'ha de disposar d'un editor de codi, d'un compilador que generi l'arxiu que s'ha de gravar en el Microcontrolador i del gravador que et permet gravar aquest fitxer.

Per l'editor de codi s'ha fet servir l'MPLAB IDE, que és de la casa Microchip igual que el Microcontrolador. Aquest programa és gratuït i es pot realitzar la descàrrega des de la pàgina http://www.microchip.com.

El motiu pel qual s'ha escollit aquest i no un altre és que es disposava del gravador Microchip MPLAB ICD2, que és el que s'ha fet servir per a gravar el codi al Microcontrolador.

On quedava el dubte és en el compilador. L'entorn MPLAB IDE, per defecte, és un compilador de llenguatge Assemblador. Per comoditat a l'hora d'escriure el codi del Microcontrolador, es va escollir fer-ho amb llenguatge C. Això va fer que

s'hagués de buscar un altre compilador que no fos el que ve per defecte. Buscant per internet es van trobar diferents compiladors que podien funcionar amb l'MPLAB IDE. Un d'ells és el propi de la casa Microchip anomenat C18, i l'altre és d'una altra marca, però que també pot funcionar en l'entorn MPLAB IDE, és el CCS C.

El que es va fer va ser instal·lar els *plug-ins* que permeten treballar amb C18 i CCS C, i es van dedicar uns dies a conèixer bé com funcionaven. Com que el Microcontrolador ja estava muntat en una placa de proves, es van editar, compilar i gravar senzills programes que permetessin veure el funcionament de cada compilador. Tots dos utilitzen el llenguatge C, però cada un utilitza les seves pròpies llibreries i això vol dir que tot i ser llenguatge C tenen les seves diferències a l'hora d'escriure el codi.

Després de provar amb els dos compiladors es va decidir quedar-se amb el CCS C. No és el de Microchip, però és més còmode programar amb ell, bastant més entenedor i, per tant, més fàcil d'utilitzar.

Resumint doncs, s'ha fet servir l'entorn MPLAB IDE amb un *plug-in* del compilador CCS C, que permet gravar amb el gravador MPLAB ICD2, que és l'únic del que es disposava abans d'escollir res de tot això.

3.3 DISSENY DE LES CONNEXIONS PEL MICROCONTROLADOR PIC

Per al disseny de les connexions s'ha utilitzat el software OrCAD 10.5. No s'ha escollit aquest programa per ser millor que un altre, sinó perquè ja se'n coneixia el funcionament.

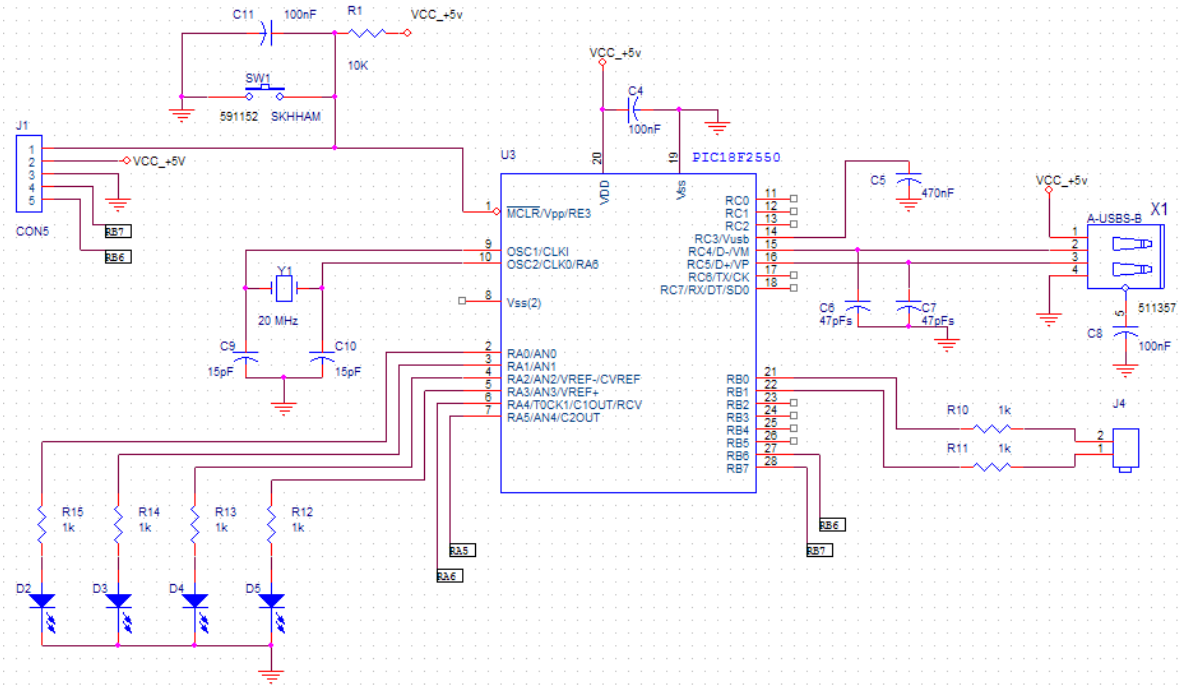
Connexionat de la placa

S'ha dividit l'esquemàtic de la placa en tres parts:

- La part de control, on hi ha el Microcontrolador amb tot el seu hardware necessari.
- La font d'alimentació, on hi ha l'alimentació pel gravador.
- Els transistors que donen potència als relés per a activar les pistes.

Esquemàtic del hardware per al Microcontrolador

L'esquemàtic que està a continuació mostra tots els components que serveixen per a gestionar el control del Microcontrolador.



En el pin_1 del microcontrolador hi ha el reset, que està format per una resistència a 5v i un polsador que dóna massa. Si es prem el polsador es posa massa al pin_1 i el Microcontrolador es reseteja.

En el pin_9 i pin_10 hi ha connectat el cristall de 20Mhz. Segons el *data sheet* del Microcontrolador s'han de col·locar dos condensadors a massa com mostra l'esquemàtic. El quadre extret del *data sheet* que mostra aquesta informació és a continuació:

TABLE 2-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

Osc Type	Crystal Freq	Typical Capacitor Values Tested:	
		C1	C2
XT	4 MHz	27 pF	27 pF
HS	4 MHz	27 pF	27 pF
	8 MHz	22 pF	22 pF
	20 MHz	15 pF	15 pF

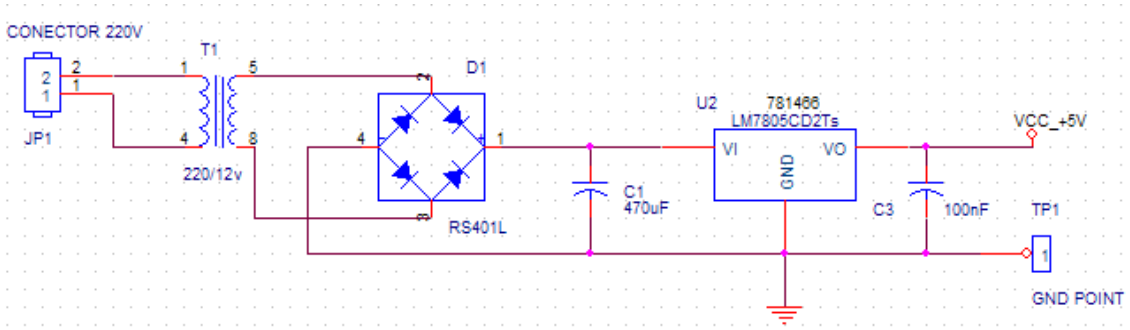
El PORT_A s'utilitza per les sortides que s'han de controlar. En els quatre primers pins (pin_2, pin_3, pin_4, pin_5) hi ha connectats els leds que faran de semàfor. En els dos següents pins (pin_6, pin_7) hi ha les sortides que activaran l'alimentació de les pistes.

El PORT_B s'utilitza per les entrades procedents dels sensors, ja que els pins RB0 i RB1 són les entrades d'interrupcions externes del Microcontrolador. S'hi ha col·locat una resistència per a evitar un excés de corrent que pogués cremar el microcontrolador. A part, també hi ha els pins de gravació, que són els pins RB6 i RB7, que van directament al connector de gravació.

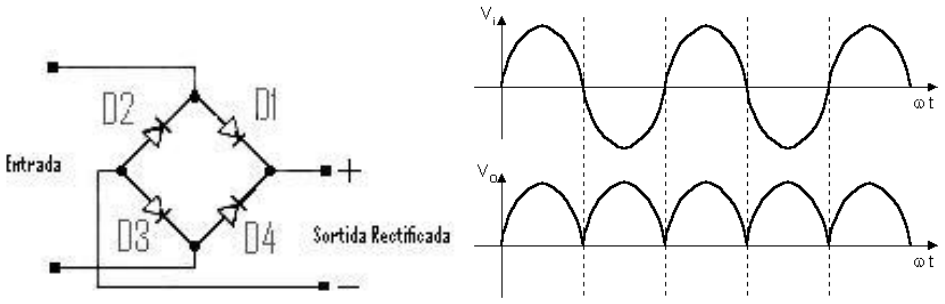
Per últim, en el PORT_C, hi ha el connector USB. Els condensadors situats en les línies D+ i D- del connector són simples filtres que poden evitar petits sorolls en la transmissió. El condensador col·locat en el pin_14 i de valor 470nF, està indicat en el *data sheet* del Microcontrolador que s'ha de col·locar per al correcte funcionament d'aquest.

Esquemàtic de la Font d'alimentació

El següent esquemàtic mostra la Font d'alimentació.



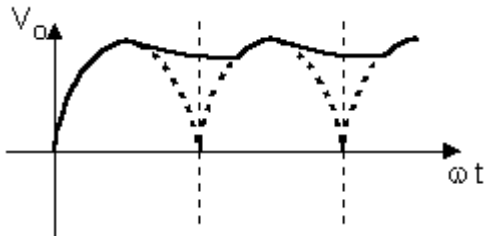
En el connector de 220v hi arriba la tensió de la xarxa. El transformador T1 té un secundari que ofereix 12v. Aquests 12v s'entren a un pont rectificador, la funció del qual és convertir el semiperíode negatiu a positiu. La *imatge 9* mostra el funcionament del pont de díodes i la gràfica amb la ona rectificada resultant.



Imatge 9: Pont de díodes i Ona rectificada

Durant el semiperíode positiu funcionen els díodes D1 i D3. Durant el semiperíode negatiu funcionen els díodes D4 i D2. Sempre funcionen dos díodes per a poder tancar el circuit a través de la carga.

El condensador C1 serveix per a aconseguir una senyal més contínua, en comptes de tant ondulada. La *imatge 10* mostra com quedaria després de passar pel condensador.



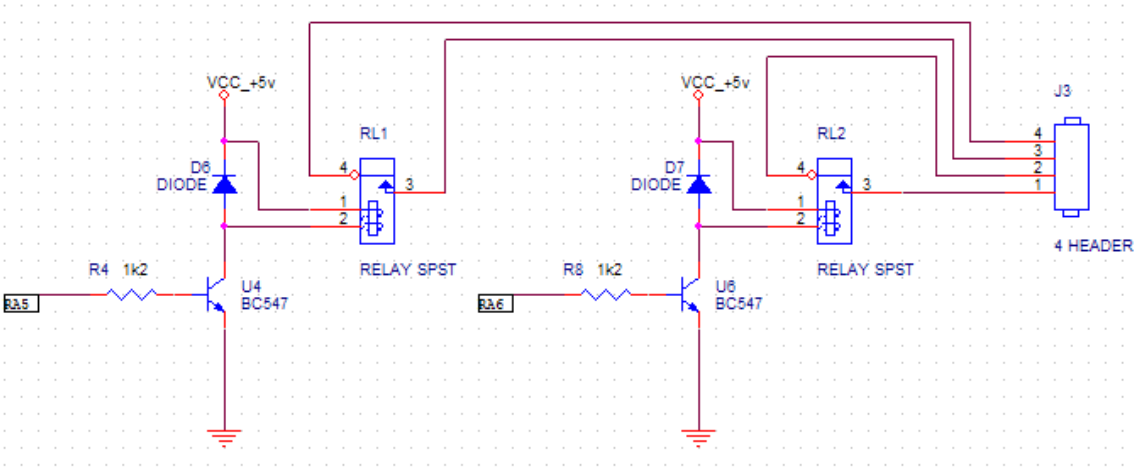
Imatge 10: Ona rectificada i estabilitzada

Finalment fem ús d'un regulador de tensió LM7805. La funció d'aquest regulador és la d'estabilitzar la tensió a la sortida a +5v.

Esquemàtic dels transistors de potència

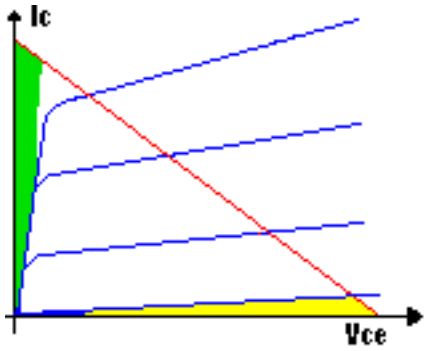
Els pins RA5 i RA6 del Microcontrolador ja s'ha vist que s'utilitzen per a accionar les sortides d'alimentació de les pistes. L'alimentació d'una pista d'Slot és aproximadament de 13-14v. El Microcontrolador no té prou tensió per a alimentar les pistes i s'ha buscat una alternativa. La sortida del Microcontrolador activa

l'entrada d'un transistor. Aquest transistor genera prou corrent a la sortida per a activar la bobina del relé. Els contactes del relé tanquen el circuit d'alimentació de les pistes, aconseguint així que hi passi corrent. En la següent imatge podem veure com quedaria:



El díode en paral·lel amb la bobina del relé té la funció d'absorbir les tensions que es generen en els circuits inductius.

Per a realitzar un interruptor mitjançant el transistor, s'ha de fer treballar aquest en la zona de saturació i tall. La següent gràfica mostra les zones de saturació i tall en un transistor.



En l'eix vertical hi ha la intensitat de col·lector i en l'eix horitzontal la tensió col·lector-emissor. Si es vol que el transistor treballi en la zona de tall i saturació s'ha d'aconseguir el següent:

Zona tall (zona groga)

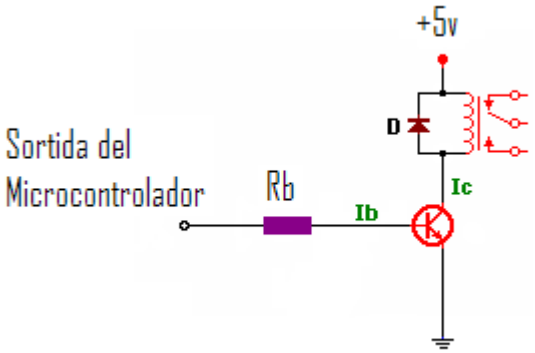
- El corrent que circula pel col·lector és mínim i la tensió col·lector-emissor és màxima. Amb això s'aconsegueix simular un contacte obert.

Zona saturació (zona verda)

- El corrent que circula pel col·lector és màxim i la tensió col·lector-emissor és quasi nul·la.

La línea vermella mostra la zona de treball del transistor. Només s'ha de fer treballar el transistor en les zones on la línia vermella coincideix amb les franges groga i verda.

L'esquema de connexionat del transistor quedaria de la següent manera:



Per a calcular la resistència de base és necessari saber el guany del transistor. Si s'utilitza el *data sheet*, es veurà que hi ha un punt que indica que el guany del transistor quant aquest treballa en saturació és igual a 20 aproximadament.

Per a calcular el corrent que circularà pel col·lector quan s'activi el relé, s'ha de saber la resistència que té aquest. Si s'utilitza el *data sheet* del relé, s'hi pot trobar una taula que indica que el relé té una resistència de 75 ohm quan treballa a +5 volts.

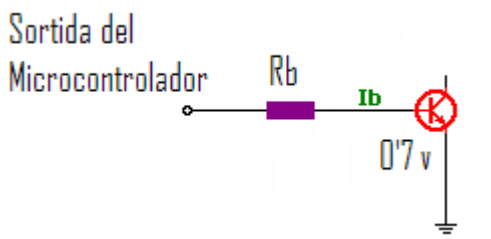
També s'ha de tenir en compte que no tots els 5 volts cauràn a la bobina del relé, ja que quan el transistor es saturi tindrà una tensió entre l'emissor i el col·lector d'uns 200 mV.

Llavors, quan el transistor es saturi, el corrent que circularà pel col·lector serà:

Ic = (5v – 0,2v) / 75 ohm = 64 mA

Si es divideix el resultat per 20, es pot saber el valor del corrent necessari a la base per a aconseguir activar el relé.

La resistència de base s'ha de calcular de la següent manera:



La sortida del Microcontrolador donarà +5v i la unió base-emissor del transistor tindrà una caiguda de 0'7v. Per tant la Rb serà:

Rb = (5v – 0'7v) / 3,2mA = 1343,75Ω

Ja que el valor de 1343,75 Ω no és estàndard, s'ha escollit un valor per a la resistència de base de 1K2 Ω.

3.4 DESENVOLUPAMENT DEL SOFTWARE PEL MICROCONTROLADOR PIC

3.4.1 CONFIGURACIÓ DELS FUSIBLES INTERNS

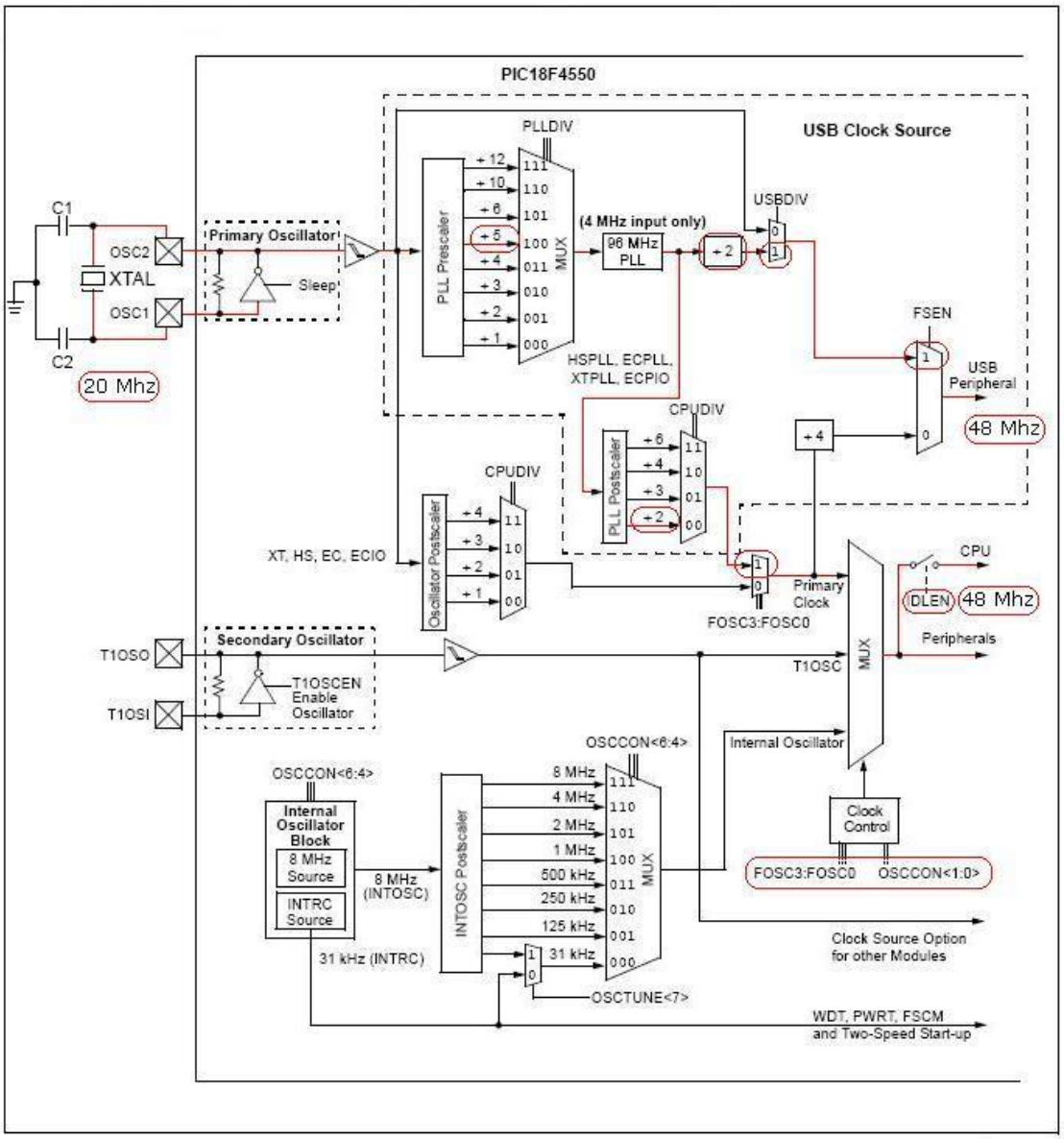
Com en tots els Microcontroladors s'han de configurar uns fusibles interns que permetin activar o desactivar registres. Aquests registres serveixen per a configurar internament el Microcontrolador.

En el cas dels Microcontroladors de la família 18Fxx5x, que incorporen port de comunicacions USB, es necessita un oscil·lador de 48MHz per a la comunicació USB, que pot ser diferent de l'oscil·lador que utilitzem pel programa. Això és possible utilitzant els fusibles. A continuació es poden veure els fusibles que s'utilitzen i la seva funció:

HSPLL - Alta Velocitat del cristall amb PLL habilitat
NOWDT - No activa el Watch Dog
NOPROTECT - No protegeix el codi de la lectura
NODEBUG - No habilita el mode de debugació
USBDIV - El rellotge procedeix del PLL dividit per 2
PLL5 - Divideix per 5 els 20Mhz del rellotge d'entrada
CPUDIV1 - El rellotge té factor 1:1 (no postcaler)
VREGEN- Habilita el generador d'alimentació per a el bus USB

La *imatge 11* ens servirà per a explicar la configuració dels fusibles. La línia vermella marca el recorregut que es seguirà. El port USB funciona a 48Mhz. El cristall que s'utilitza és de 20Mhz, i amb l'ajuda dels fusibles s'aconseguiran els 48Mhz necessaris. S'utilitzaran els següents fusibles:

- USBDIV
- PLL5
- CPUDIV1



Imatge 11: Configuració fusibles per a la configuració de l'oscil·lador

Com es pot veure s'ha de col·locar el cristall entre els pins OSC1 i OSC2. El cristall ha de ser de 4Mhz, 8Mhz, 12Mhz, 16Mhz, 20Mhz, 24Mhz, 40Mhz o 48Mhz per a ser compatible amb el sistema que ve a continuació.

El cristall que genera els 48Mhz per l'USB és el mateix que s'utilitza per al rellotge del PIC, que pot tenir la mateixa freqüència o no, segons la configuració dels fusibles. Es comença a partir de les tres línies que surten després del Trigger del Primary Oscillator.

La primera línia, la superior, va directament al switch USBDIV que si està a zero indica que la freqüència base original del cristall és directament injectada a l'USB, també ha de passar el switch FSEN que tria entre el sistema directe/PLL o el Primary Clock del CPU. Aquesta opció d'injectar directament la freqüència del cristall és només possible si s'utlilitza un cristall de 48 Mhz, que és el que fa falta per a l'USB. Qualsevol altre cristall ha de ser tractat per a aconseguir els 48Mhz necessaris.

La segona línia va a l'entrada del PLL Prescaler, és a dir, un divisor de freqüència. En cadascuna de les seves sortides hi haurà FOSC dividida per 1, 2, 3, 4, 5, 6, 10 o 12. Mitjançant PLLDIV, que no és més que un Multiplexor, es selecciomarà la que es vulgui fer servir.

Així, amb el cristall de 20Mhz utilitzat en el projecte, si a PLLDIV s'hi col·loca un 100, es dividirà per 5 el valor de FOSC i s'aconseguiran 4Mhz a la sortida del MUX. Si per contra el cristall és de 20Mhz i en PLLDIV s'hi col·loca un 100, llavors es dividirà per 5 FOSC i també s'aconseguiran 4Mhz a la sortida del MUX.

Aquesta sortida del MUX és el que s'utilitza per a injectar-la al PLL de 96Mhz. Si s'hi posen 4Mhz, ell genera 96Mhz. És aquesta capacitat de passar de 4Mhz a 96Mhz la que dóna la possibilitat d'usar molts cristalls diferents.

Però 96Mhz és el doble del que fa falta per a l'USB, que són 48Mhz. Així que immediatament després és necessari un divisor per 2, que és el segon camí pel qual s'arriba a USBDIV, i en aquest cas se li posa un 1 per a usar el senyal provinent del PLL.

A més d'injectar el senyal oscil·lant en USBDIV, també es connecta el senyal del PLL a 96Mhz en un Postscaler, un altre divisor, en aquest cas per 2, 3, 4 o 6 i els senyals del qual van al CPUDIV. O sigui que es pot generar un senyal de rellotge per al PIC, no per a l'USB sinó per a la velocitat d'execució del programa prenent-la del PLL i que pot ser de 16Mhz, 24Mhz, 32Mhz o 48Mhz.

Però, a més, el senyal original arribava en paral·lel a l'Oscilator Postcaler, un altre divisor més, que de forma directa, sense passar pel mòdul PLL, divideix la freqüència original del cristall per 1, 2, 3 o 4 i que també va a parar al CPUDIV, però des d'un altre origen. Amb aquest mòdul es pot obtenir una altra gamma de freqüències diferent per a fer funcionar el programa.

Quin dels dos CPUDIV serà utilitzat ho seleccionem amb el switch FOSC3:FOSC0, que és d'on es treu la freqüència definitiva d'execució de programes.

Per últim, també hi ha disponible una entrada provinent del Primary Clock i que dividida per 4 també arriba al FSEN i pot ser utilitzada en lloc de la que li arriba des del canal directe/PLL.

Finalment, els fusibles ja estan configurats per a aconseguir els 48Mhz necessaris per a l'USB. Pel que fa la velocitat del programa del Microcontrolador, s'utilitzarà també una configuració de 48Mhz.

3.4.2 CONFIGURACIÓ DELS TIMERS I REGISTRES D'INTERRUPCIÓ

3.4.2.1 Configuració TIMER0

El timer 0 ens serveix per a realitzar el rellotge. La interrupció que es genera quan es desborda el TIMER0 es produeix cada 10 mil·lisegons, que és la unitat mínima de mesura que té el sistema. (Anem a veure com s'aconsegueixen aquets 10 mil·lisegons:)

S'utilitza un rellotge intern que funciona a 48Mhz per tant:

T=[(65535 - precàrrega) * Preescaler * 4] / Fosc

T = 10 mil·lisegons
Fosc = 48Mhz
Prescaler = 2

S'ha de jugar amb el prescaler i la precàrrega per a aconseguir els 10ms. Per a trobar la precàrrega s'ha suposat un prescaler de 2.

T = ((65535 - precàrrega) * 2 * 4) / 48 MHz

=> precàrrega = [(0.01*48000000) / 2*4] – 65535

Precàrrega = 5535

Quan s'inicialitzi cada vegada el TIMER0 s'haurà de fer de la següent manera:

Set_timer0(5535);

Això farà que el timer comenci a comptar a partir de 5535 en comptes de 0 i que cada vegada que es desbordi hagi passat una centèsima, que és la unitat mínima del cronòmetre.

3.4.2.2 CONFIGURACIÓ TIMER1 i TIMER3

Aquets dos timers s'utilitzen per a evitar els rebots dels sensors a l'hora de comptabilitzar les voltes dels cotxes. Quan el sensor detecta el vehicle, genera una interrupció externa. Degut a què el Microcontrolador és tant ràpid, genera moltes interrupcions cada vegada que es dispara el sensor per culpa dels petits rebots que aquest provoca. Això s'evita engegant els timers.

Tan el timer1 com el timer3, estan programats per a desbordar-se cada mig segon aproximadament. Quan el sensor genera la interrupció externa, a part de fer les captures de temps pel cronòmetre, activa els timers. La pista1 activa el timer 1 i la Pista2 activa el timer3. El que fan els timers quan es desborden és canviar d'estat 0 a 1 una variable. Quan el sensor genera la interrupció, abans de poder entrar a la funció d'interrupció, té la condició de la variable, si és 1 entra i si és 0 passa de llarg.

Amb això s'aconsegueix que, quan el sensor es dispara, el Microcontrolador llegeix la captura i no pot llegir més captures del mateix sensor fins al cap de mig segon, evitant així falses lectures degudes als rebots que pot provocar el sensor.

3.4.2.3 CONFIGURACIÓ DE LES INTERRUPCIONS EXTERNES

El microcontrolador disposa de tres interrupcions externes. Només se n'utilitzen dues, una per cada pista, però totes dues funcionen de la mateixa manera.

Aquestes interrupcions s'activen quan detecten un canvi d'estat en el pin del Microcontrolador. El canvi d'estat a detectar és configurable de nivell alt a nivell baix o al revés. Això es fa amb la següent funció:

```
ext_int_edge(0, H_TO_L );
ext_int_edge(1, H_TO_L );
```

El 0 i 1 marcats en color verd indiquen la interrupció que és, la interrupció 0 i la 1 respectivament. H_TO_L, significa que es generarà interrupció quan el pin del Microcontrolador passi de High a Low, d'estat alt a baix o, més ben dit, de +5volts a 0 volts.

El sensor que s'utilitza per a detectar el pas dels cotxes està sempre a nivell alt, excepte quan detecta el pas del cotxe, que canvia a nivell baix, 0 volts. Amb aquesta configuració del Microcontrolador es generarà una interrupció cada vegada que passi un cotxe.

3.4.3 FUNCIONAMENT GENERAL DEL PROGRAMA

Com ja s'ha definit en els objectius del projecte, s'ha de programar el sistema utilitzant una arquitectura Mestre-Esclau on el PC és el Mestre i el Microcontrolador l'Esclau. Això significa que el PC és el que s'ha d'encarregar de donar ordres i el Microcontrolador ha de realitzar aquestes ordres.

Aquesta arquitectura de Mestre-Esclau és manté durant l'execució de tot el programa, excepte quan s'han de passar cronometratges, que per comoditat de

treball, el Microcontrolador va enviant els temps sense prèvia pregunta del PC. Quan s'executa el programa és el PC qui s'encarrega de configurar els paràmetres del propi software del PC i també els del Microcontrolador. També és el PC el que dóna les ordres de començament de cronometratge i de finalització, però durant el període de cronometratge, el PC resta a l'espera de que el Microcontrolador l'hi vagi passant temps. El PC no perd la condició de Mestre ja que en tot moment pot parar el cronometratge i configurar un nou mode de funcionament.

Aquest fet no suposa cap problema a l'hora d'enviar dades, ja que de la mateixa manera que quan el PC envia una trama al Microcontrolador, aquest ha de respondre confirmant que la trama ha estat rebuda, quan el Microcontrolador envia una trama al PC, que inclou el cronometratge realitzat, aquest ha de respondre que la trama s'ha rebut correctament, si no, es realitzarà l'operació fins que les dades siguin enviades correctament.

Com ja s'ha dit en la introducció del projecte, s'utilitzarà la comunicació USB CDC (Universal Serial Bus, Communications Device Class) que permet implementar un port sèrie virtual en el PC. A continuació es veuran les funcions que fan falta a l'hora d'inicialitzar el Microcontrolador per tal de configurar-lo com a USB CDC.

Dins el Programa Principal (Main), les dues primeres instruccions que s'han de carregar són les següents:

- usb_cdc_init()
- usb_init()

usb_cdc_init()

Aquesta funció configura el bus USB per a treballar en mode de port sèrie virtual. Per a poder compilar el programa farà falta incloure en l'arrel del programa la llibreria *usb_cdc.h*, que inclou la funció *usb_cdc_init()*. Aquesta funció permet, entre altres coses, configurar la velocitat de transmissió de dades. També inclou les ordres que permeten detectar l'estat del buffer, o les ordres que permeten configurar la longitud de les dades a enviar o rebre.

usb_init()

Un cop configurat el bus com a port sèrie virtual, s'utilitza la funció *usb_init()* que inicialitza el port USB. Aquesta funció inicialitza el perifèric i el manté connectat al port USB. També s'encarrega d'habilitar les interrupcions, que permetran detectar si hi ha algun bit en el buffer. Aquesta funció està inclosa en la llibreria *pic18_usb.h,* que s'haurà d'incloure en l'arrel del programa.

Per a poder llegir qualsevol bit del port USB fan falta les següents funcions:

- usb_enumerated()
- usb_kbhit()
- usb_cdc_getc()

Usb_enumerated()

Aquesta funció és la que permet connectar amb el PC. Com ja s'ha definit anteriorment, el dispositiu d'aquest projecte s'identifica mitjançant un VID&PID. És ara, a l'hora de connectar amb el PC, que es fa ús d'aquest VID&PID, i la funció *usb_enumerated* retornarà TRUE si el PC l'ha identificat i s'hi ha connectat. Un cop connectats s'ha d'esperar a què el PC enviï algun bit.

Usb_kbhit()

Seguint del punt anterior, es disposa de la funció usb_kbhit, que causa interrupció quan un o més bits estan al buffer d'entrada del PIC. Per tant, cal esperar que la funció retorni TRUE per a poder començar a llegir dades del buffer d'entrada.

Usb_cdc_getc()

Un cop usb_kbhit ha interromput, s'utilitza la funció usb_cdc_getc() per anar llegint els caràcters que hi ha al buffer. Com que es treballa amb trames de caràcters, quan es llegeixen caràcters del buffer s'han d'anar guardant en una cadena, la qual ja es verificarà després.

Per a enviar caràcters a través del bus USB es necessiten les funcions següents:

- usb_cdc_putready()
- usb_cdc_putc()

Usb_cdc_putready()

Aquesta funció retorna TRUE si el bus de comunicacions està lliure i, per tant, pot ser utilitzat per a enviar informació.

Usb_cdc_putc()

Aquesta funció serveix per a enviar caràcters a través del bus de comunicacions, sempre i quan usb_cdc_putready hagi detectat que el bus està lliure.

Un cop explicades les funcions principals que s'utilitzaran per programar el Microcontrolador, es passarà a analitzar més detalladament el seu funcionament. Hi ha dos parts molt diferenciades en el codi del Microcontrolador, una és la part de configuració i l'altra és la part de funcionament. A continuació, s'expliquen els processos utilitzant el *diagrama 3* que mostra el funcionament del Microcontrolador:

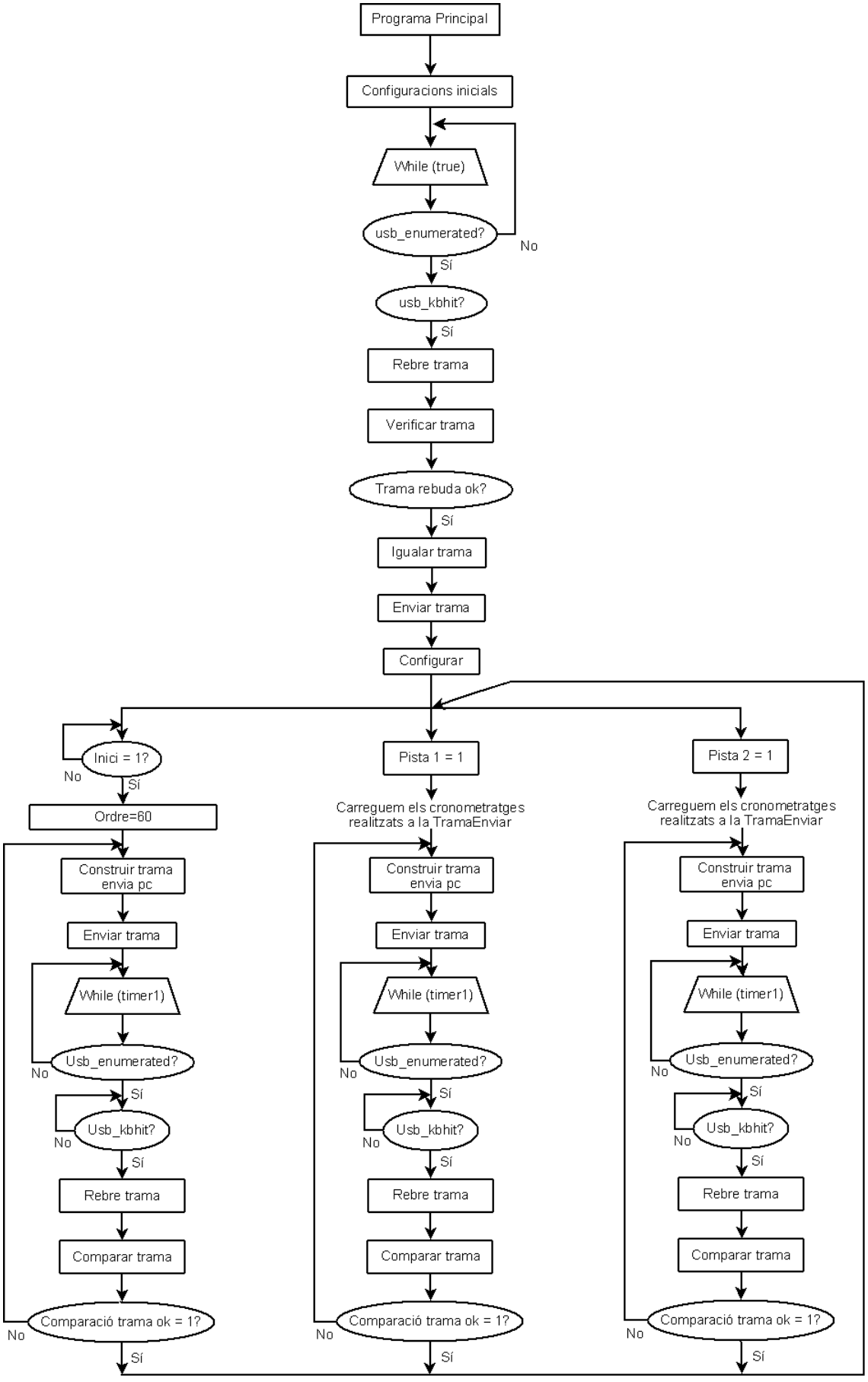


Diagrama 3: Funcionament del Microcontrolador

Com es veu en el *diagrama 3*, el primer que es fa a l'executar el programa principal és inicialitzar les variables, configurar timers i registres d'interrupció. La primera funció que es crida és *usb_enumerated()*, que com ja s'ha vist, és la funció que connecta amb el PC. Un cop connectats, el Microcontrolador resta a l'espera que li arribi una trama del PC. Un cop arriba una trama es crida la funció *RebreTrama()*, que emmagatzema els caràcters rebuts en una cadena. Quan ja s'ha rebut tota la trama, es crida la funció *VerificarTramaRebuda()*, que mitjançant un càlcul de CRC verifica que la trama sigui correcte. Si la trama rebuda és errònia, s'espera que arribi una altra trama i, si és correcte, ja es pot continuar. El que primer es fa és respondre al PC amb la mateixa trama que s'ha rebut, mitjançant les funcions *IgualarTrama()* i *EnviarTrama()*, així el PC ja sabrà que s'ha rebut la trama correctament. Tot seguit ja es pot cridar la funció *Configurar()*, que, com el seu nom indica, configurarà el Microcontrolador amb el Mode que s'hagi escollit des del PC. Un cop arribats a aquest punt, ja s'entra en el Mode de funcionament. A partir d'ara serà el Microcontrolador qui enviarà resultats al PC i aquest respondrà si els ha rebut correctament, si no es repetirà l'operació fins que l'enviament sigui correcte.

Després de *configurar()* sempre s'entra en la branca *Inici = 1*, excepte si s'ha escollit la opció de pràctica, que només activa les pistes i no cronometra temps. La branca *inici = 1* el que fa és enviar una trama al PC per a marcar-li l'inici exacte de cronometratge. Amb aquesta trama el que es fa és sincronitzar el temps entre el Microcontrolador i el PC.

La trama que s'enviarà al PC serà la següent:

SOH	ID	60	0x000	0x000	ETX	CRCH	CRCL

En l'apartat de Fonaments Teòrics ja s'ha analitzat la trama. Aquí només es mostra el contingut que s'envia. L'ordre que s'envia és la 60, que és la que marca l'inici de cronometratge.

A les branques de *TempsPista1* i *TempsPista2* el programa hi entrarà quan el sensor hagi detectat el pas del cotxe i s'hagi creat una interrupció. En cada una de les branques hi ha les funcions necessàries per a enviar la trama amb els cronometratges enregistrats cap al PC.

Anem a veure com seria la trama que hem d'enviar:

SOH	ID	70	Segons	Pista +Centèsimes	ETX	CRCH	CRCL

En aquest cas es canvia l'ordre 60 per 70, ja que no és inici de cronometratge sinó temps de volta realitzada. En el primer bloc de dades s'hi envien els segons, com ja s'ha dit, en format hexadecimal. Com que els segons poden anar des de 0 fins a 65535, fan falta quatre caràcters en hexadecimal, per tant, tot un bloc. En el segon bloc s'hi envien el número de pista i les centèsimes, també en format hexadecimal. Les centèsimes només poden anar des de 0 fins a 99, per tant, només seran necessaris dos caràcters en Hexadecimal. La pista ocuparia el tercer caràcter del bloc, sent 1 per a la pista 1 i 2 per a la pista 2. Per tal de tenir un 1 o un 2 en la tercera posició d'un número hexadecimal s'ha de fer el següent:

Pista1 = 256 en decimal = 0x100 en hexadecimal
Pista2 = 512 en decimal = 0x200 en hexadecimal

Un exemple per entendre més bé com funciona:

Si el temps que s'ha d'enviar és:

11 segons
50 centèsimes
Pista = 1 (sumar 256 a centèsimes)

El bloc 1 : 11 en decimal = B en hexadecimal
El bloc 2 : (50 + 256) = 306 en decimal = 132 en hexadecimal

La trama que s'ha d'enviar quedaria de la següent manera:

SOH	ID	70	0x000B	0x0132	ETX	CRCH	CRCL

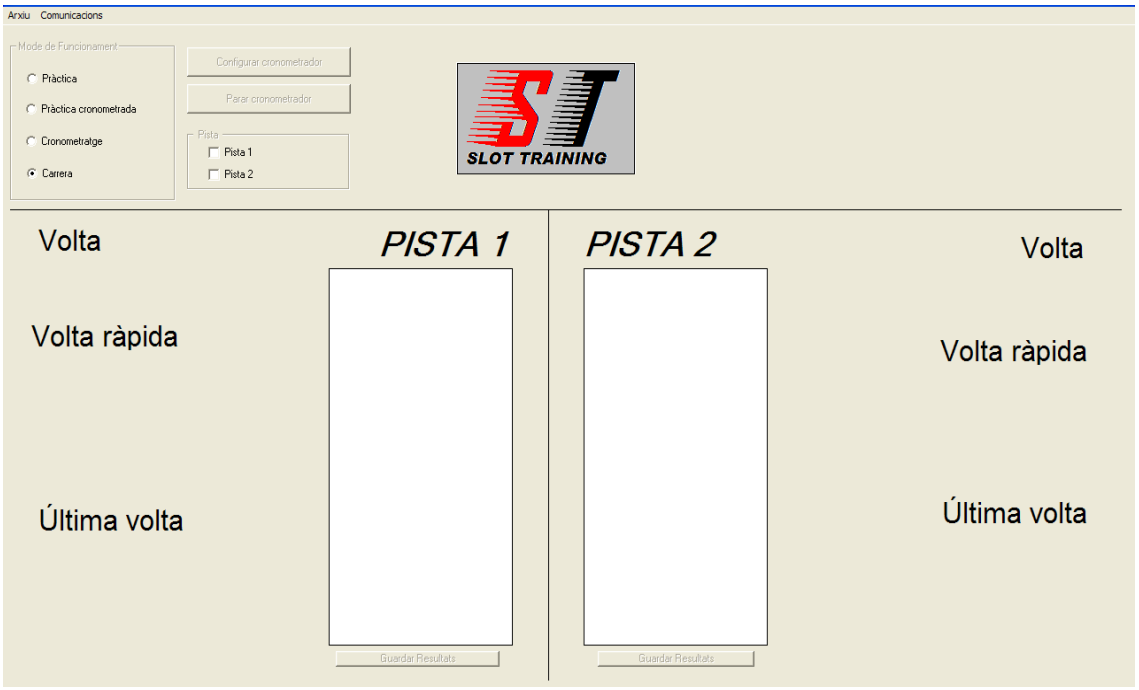
El CRCH i CRCL no està calculat ja que es pot veure un exemple de com quedaria en l'apartat on s'explica la trama.

Un cop el programa està configurat i inicialitzat, va enviant temps sempre i quan el PC no li indiqui el contrari. Això passaria si rebés un trama amb l'ordre 50 o 80. Com ja s'ha definit en l'apartat de les ordres, l'ordre 80 serveix per a indicar finalització de voltes i la 50 per a parar el mode en funcionament.

En el cas que s'hagi completat el mode entrat o s'hagi aturat el mode en funcionament, el Microcontrolador sempre queda a l'espera de nova configuració.

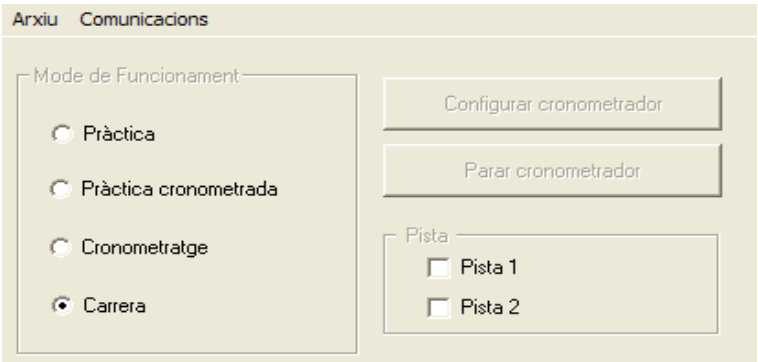
3.5 DESENVOLUPAMENT DEL SOFTWARE PEL PC

El PC és el Mestre del sistema, això significa que sempre portarà el control. Un dels objectius del projecte era que el sistema havia de ser molt visual i de fàcil ús. Tot seguit es veurà una perspectiva general del funcionament del programa. En la *imatge 12* es pot veure la pantalla principal del sistema just després d'executar-se el programa.



Imatge 12: Pantalla principal programa PC

A simple vista es veuen tres regions separades per unes línies. La zona de dalt és on hi ha tot el mode de configuració. La zona de baix mostra les dades referents a les captures dels cronometratges. En la *imatge 13* es veu l'ampliació de la zona de configuració.



Imatge 13: Zona de configuració

Per tenir control sobre el sistema s'han creat diferents estats en els que es pot trobar el programa:

Comunicacions Parades – és l'estat en el qual es troba el programa just quan s'executa i espera connectar-se amb el microcontrolador.

Pendent escollir Mode – un cop connectats, s'espera a què l'usuari esculli un mode de funcionament. Els modes de funcionament són:

Mode Pràctica
Mode Pràctica Cronometrada
Mode Cronometratge
Mode Carrera

En els quatre modes anteriors s'hi serà mentre no s'acabi el mode o l'usuari decideixi parar el mode en funcionament. Un cop finalitzi el mode o s'aturi, es retornarà a l'estat de *Pendent escollir mode.*

El *diagrama 4* que hi ha a continuació mostra el procediment que segueix el programa per a canviar d'estat.

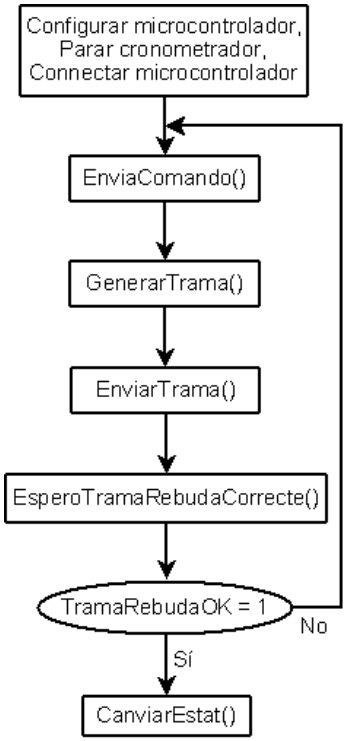


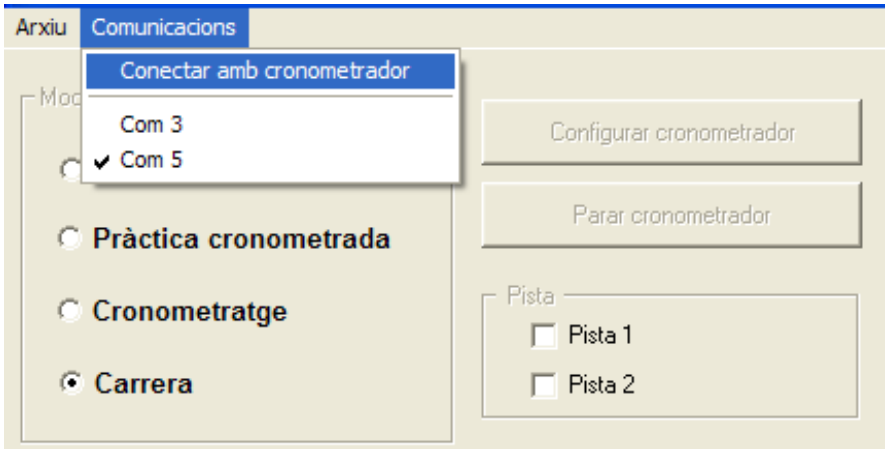
Diagrama 4: Canvis d'Estat de Funcionament

La funció *EnviaComando()* comprova que els caràcters que s'estan carregant a la trama estiguin dins els límits marcats, i si ho estan crida la funció *GenerarTrama().* La funció *GenerarTrama()* col·loca tots els caràcters en la posició que els hi corresponen en la trama, calcula el CRC de la trama que s'envia i crida la funció *EnviarTrama().* La funció *EnviarTrama()* va enviant via port sèrie tots els caràcters de la trama.

Quan s'han enviat tots els caràcters s'entra en la funció *EsperoTramaRebudaCorrecte().* Aquesta funció resta a l'espera que el Microcontrolador respongui amb la mateixa trama que s'ha enviat des del PC. No s'espera infinitament, ja que hi ha un Timer que marca un temps màxim d'espera. Si no es rep la trama, apareixerà el missatge de "no es pot comunicar amb el cronometrador" i el PC esperarà que entrem una nova configuració. Si la trama rebuda és correcte, es farà *CanviarEstat(),* i així el programa mai no es perdrà.

A continuació es veurà el procediment a seguir per a connectar amb el Microcontrolador, ja que quan s'executa el programa, aquest es troba en l'estat de *comunicacions parades* i no permet fer res més que connectar amb el

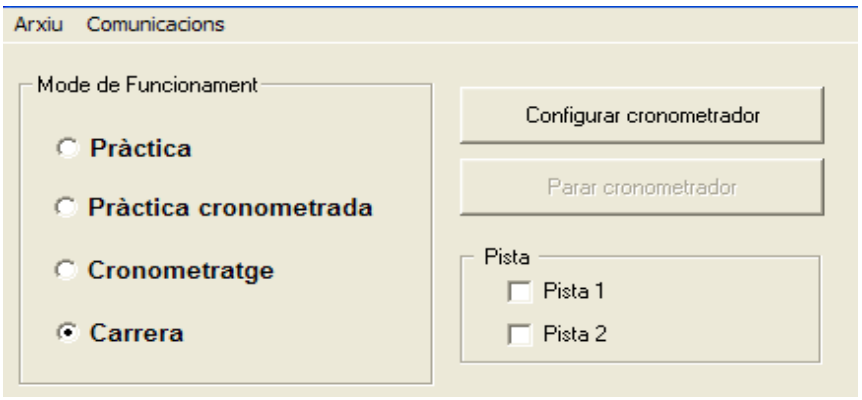
Microcontrolador. En la *imatge 14* es mostra el procediment per a connectar amb el Microcontrolador:



Imatge 14: Procediment per connectar amb el Microcontrolador

S'ha de clicar al damunt de *comunicacions*, s'obrirà un menú on hi ha l'opció de *Connectar amb el Cronometrador* i on també permet seleccionar a quin *Com* està connectat el Cronometrador. Per a connectar amb el cronometrador s'ha d'enviar una trama amb l'ordre 0. L'ordre 0 és el que permet comprovar les comunicacions, ja que el Microcontrolador retorna la mateixa trama que li envia el PC. Si les comunicacions funcionen, el sistema ja està connectat i, per tant, l'estat ha canviat de *comunicacions parades* a *pendent escollir mode*.

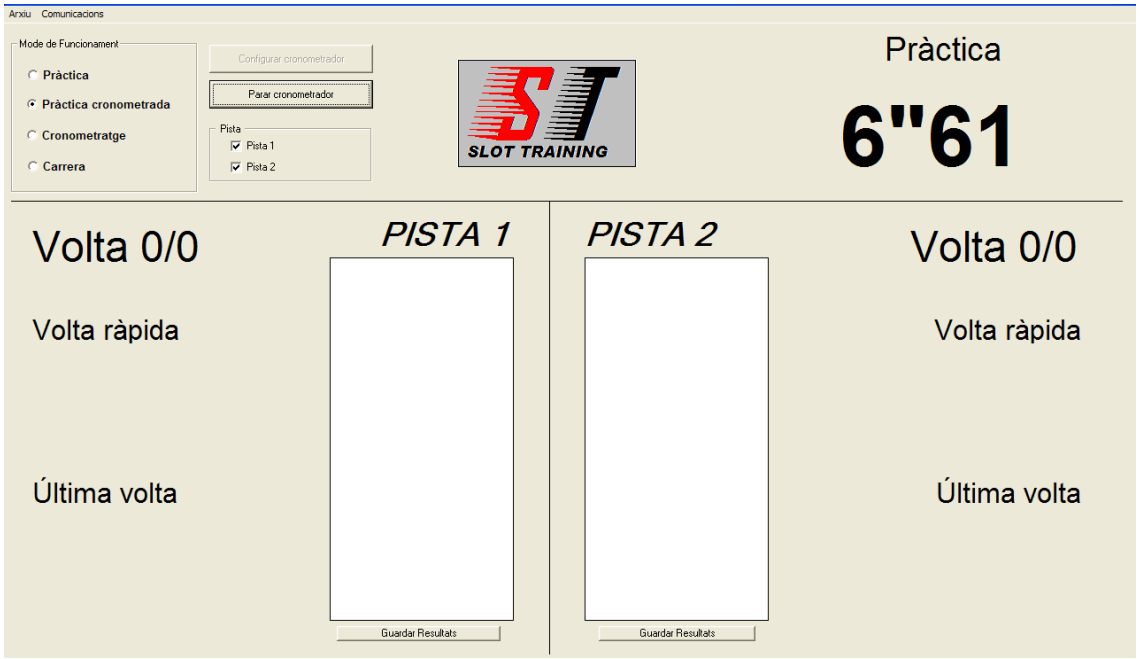
Un cop el sistema ja està connectat, ja es poden marcar les opcions que es desitgin. Com es pot veure en la *imatge 15*, ja s'ha activat el botó de *Configurar Cronometrador,* que és el que permet començar un Mode.



Imatge 15: Activació botó Configurar Cronometrador

Els Modes de Funcionament ja estan explicats dins el seu apartat del capítol de Fonaments Teòrics, ara però, es veuran les opcions que s'han de marcar per a poder iniciar un Mode. Tan si s'arrenca el Mode de *Pràctica* com el Mode de *Pràctica Cronometrada*, es té l'opció de marcar *Pista1* o *Pista2*, o les dues pistes alhora. Si s'escull l'opció *cronometratge,* el programa només permet activar una de les dues pistes. Si s'escull l'opció *carrera*, no cal marcar cap pista, ja que la carrera sempre és una pista contra l'altra. Un cop triat el Mode de Funcionament, s'ha de clicar el botó *Configurar Cronometrador* per a començar la partida.

Quan es clica a *Configurar*, el programa segueix el *diagrama 4* explicat anteriorment i canvia l'estat segons la configuració que s'hagi escollit. Seguidament s'iniciarà el Mode. El que marcarà que s'ha iniciat el mode, a part del semàfor en el Mode *Carrera* o *Cronometratge*, és que en la pantalla del programa s'iniciarà un cronòmetre, i just damunt del Cronòmetre es marcarà el Mode que s'ha escollit. En la *imatge 16* es mostra com s'inicialitzaria el Mode.



Imatge 16: Inicialització del Mode

Un cop amb el Mode inicialitzat, el programa queda a l'espera que el Microcontrolador li enviï els Cronometratges enregistrats. El *diagrama 5* mostra l'estat en el que es troba el programa:

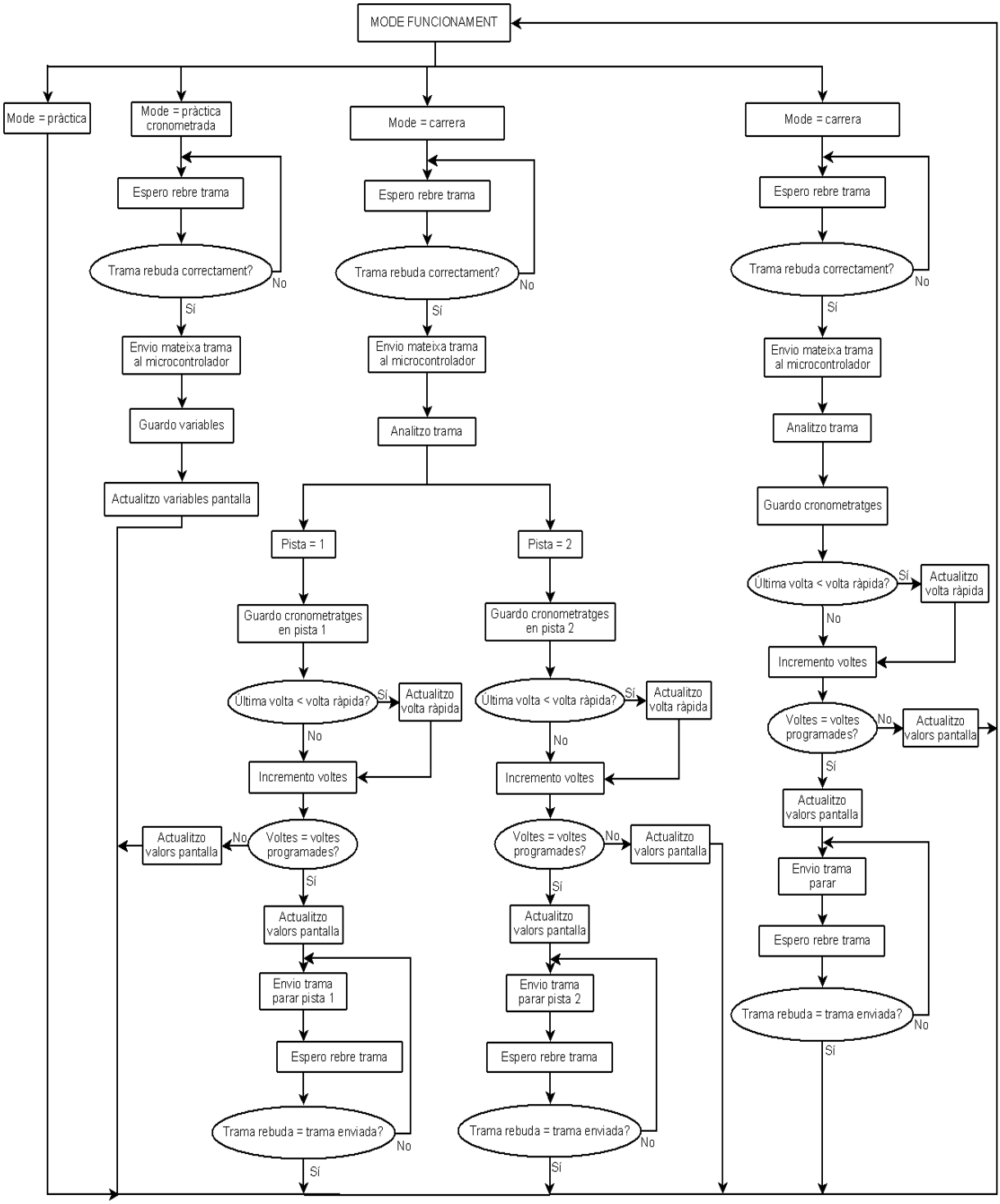


Diagrama 5: Mode de Funcionament

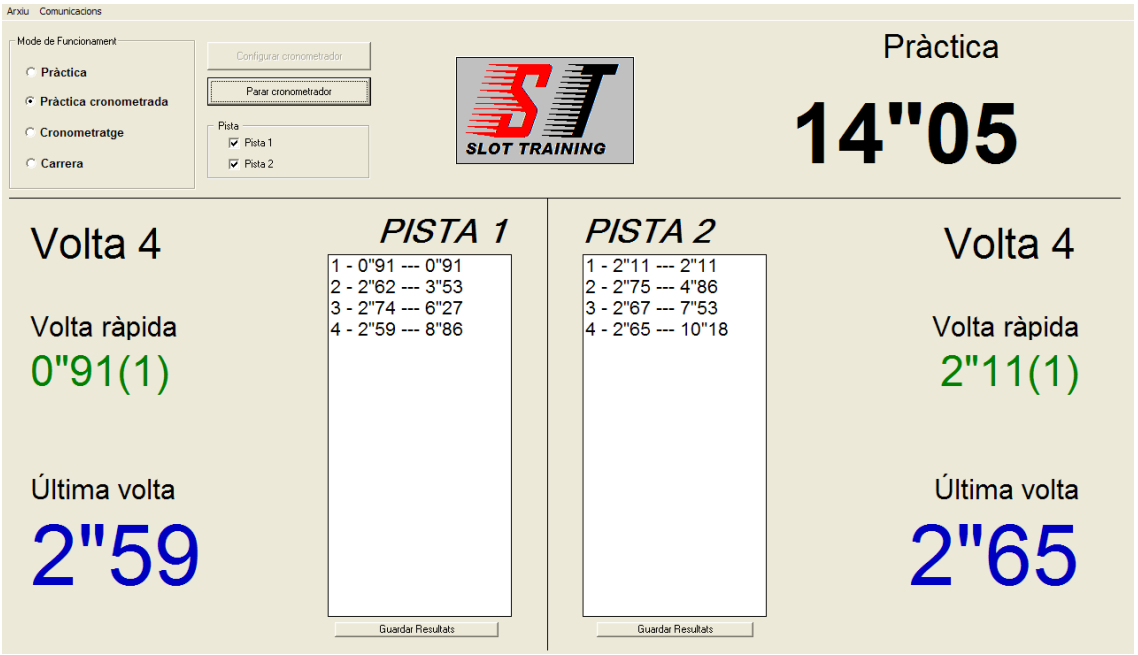
El programa executarà el mode fins que aquest s'acabi o fins que s'activi l'ordre de parar. En el mode *pràctica*, com que només s'han activat les pistes i no es cronometra el temps, l'única manera de sortir-ne és parant el mode. En el Mode *Pràctica Cronometrada*, també s'ha de sortir-ne parant el mode, ja que, tot i que es cronometra el temps, no s'ha establert un número de voltes a fer i el programa

quedaria en aquest estat indefinidament. En els modes *Cronometratge* i *Carrera* es sortirà del Mode quan s'acabin les voltes que s'hagin de fer.

Seguint amb l'execució del Mode, quan el PC rep una trama on hi ha inclòs un Cronometratge, realitza els següents passos:

- Crida la funció per a verificar que la trama rebuda sigui correcte.
- Si la trama és correcte, guarda el temps i pista enviats pel Microcontrolador i retorna confirmació al Microcontrolador.
- Incrementa el número de volta i comprova que no sigui l'última.
- Compara el temps rebut amb tots els temps que s'han enregistrat en la mateixa pista per extreure'n si és volta ràpida. En el cas que sigui volta ràpida, guarda el número de volta i el temps en un registre.
- Actualitza els valors de la pantalla del PC.

En la *imatge 17* es veu com quedaria la pantalla després de que s'haguessin realitzat unes quantes voltes.



Imatge 17: Mode en funcionament

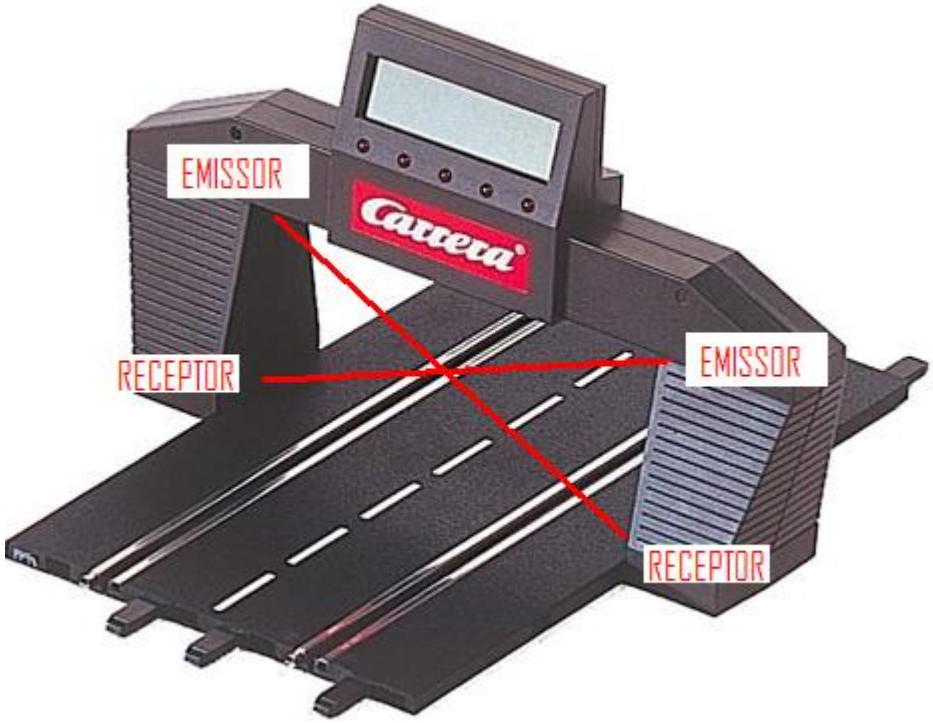
En la *imatge 17* es pot veure com per cada pista tenim la següent informació:

- Una llista amb totes les voltes que es van fent.
- El número de voltes fetes sobre el total a fer.
- La volta ràpida de totes les voltes i a quina volta l'hem fet.
- El temps de l'última volta que hem fet.

Un cop finalitzat el mode, i abans de configurar-ne un de nou, si es vol hi ha l'opció de guardar els cronometratges enregistrats. Si es clica a Guardar Resultats, es poden guardar els cronometratges en un fitxer *txt. El botó de Guardar Resultats es troba just sota la llista de temps fets.

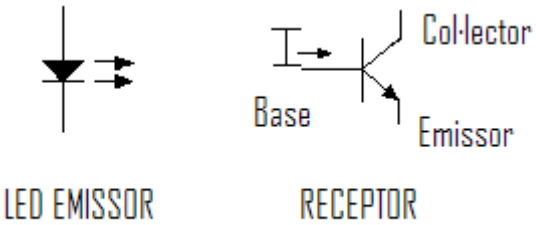
3.6 ADAPTACIÓ DELS SENSORS AL SISTEMA

Per a poder detectar el pas dels cotxes, s'ha utilitzat el següent sistema de sensors. La *imatge 18* mostra el dispositiu utilitzat.



Imatge 18: Sistema de sensors

Els sensors estan integrats dins el dispositiu, i estan formats per un conjunt de leds emissor i receptor d'infraroig. El led emissor és un led que emet llum infraroja. El receptor és un Fototransistor capaç de captar aquesta llum infraroja. En la *imatge 19* es mostra un led emissor d'infraroig i un Fototransistor receptor de llum infraroja.

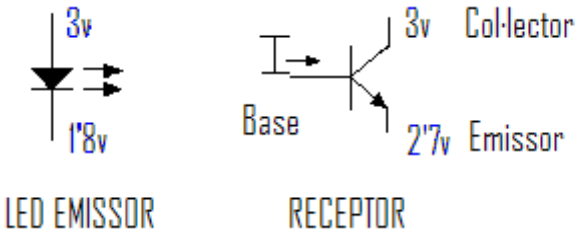


Imatge 19: Led Emissor i Receptor

Per a poder utilitzar els sensors del dispositiu com a sensors del projecte s'ha fet el següent:

- Alimentar el dispositiu a través de l'alimentació de la placa, ja que aquest s'alimenta a través de 4 piles de 1'5v.
- Llegir les variacions de tensió que es produeixen en el Fototransistor quan la llum infraroja queda obstruïda pel pas d'un cotxe.

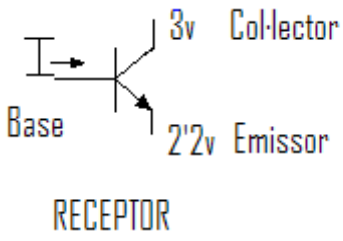
A la *imatge 20* es poden veure les tensions que hi ha en els extrems de l'emissor i el receptor quan la llum no és interrompuda:



Imatge 20: Tensions sense la llum interrumpuda

En el led emissor sempre hi ha la mateixa tensió, 3v a l'entrada i 1'8v a la sortida, ja que el led té una caiguda de tensió de 1'2v.

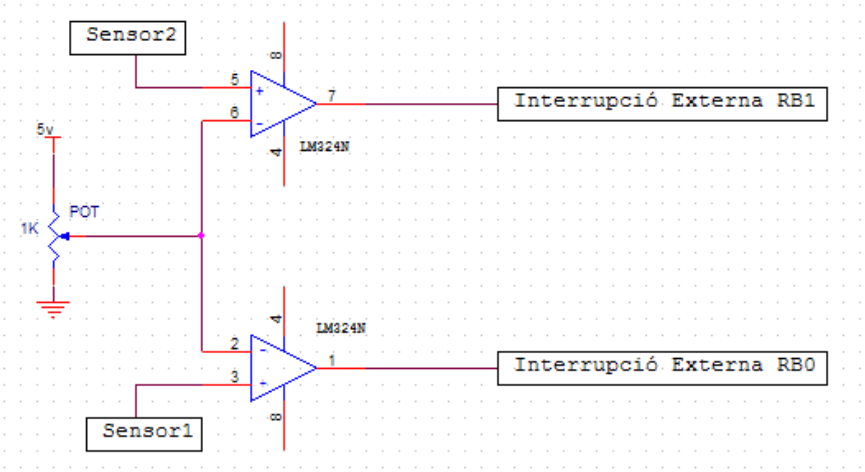
En el Fototransistor hi ha 3v al col·lector i 2'7v a l'emissor. Quan la llum infraroja es veu interrompuda pel pas d'un cotxe, les tensions que obtenim canvien segons la *imatge 21* que es mostra a continuació:



Imatge 21: Tensions amb la llum interrumpuda

L'emissor passa a tenir 2'2v. El que s'ha de fer, doncs, és dissenyar un circuit capaç de captar aquesta variació de tensió. La solució adoptada ha estat el

muntatge d'un circuit comparador utilitzant l'integrat LM324N. Aquest integrat disposa de 4 circuits comparadors en el seu interior. En l'annex 7 hi ha el *data sheet* de l'integrat per si es vol consultar la distribució de pins. L'esquema del comparador quedaria de la següent manera:



En les entrades positives 5 i 3, hi aniran les dues sortides dels Fototransistros Sensor1 i Sensor2. En les entrades negatives 2 i 6 del comparador s'hi posarà una tensió de referència de 2'5v. Aquesta tensió s'aconseguirà ajustant el Potenciòmetre d'1K ohm a la meitat. El comparador s'ha d'alimentar a una tensió de 0 i +5v. Amb això s'aconseguirà que quan a les sortides dels sensors hi hagin 2'7v, a la sortida dels comparadors hi hauran +5v, i que quan a la sortida dels sensors hi hagin 2'2v, perquè ha interromput un cotxe, a la sortida del comparador hi hauran 0v.

Amb la solució del comparador ja s'ha aconseguit adaptar el dispositiu al sistema del projecte.

4 CONCLUSIONS

La realització del projecte ha estat correcte ja que he aconseguit solucionar tots els objectius plantejats. Gràcies a la utilització d'un Pla de Treball correcte he pogut desenvolupar el projecte amb ordre, i arribar a cada punt del projecte amb els problemes anteriors solucionats. Cal dir que hi ha hagut certs punts que m'han complicat una mica el treball. Un d'ells ha estat la recerca d'informació, que ha estat llarga, degut a que el projecte abarcava molts camps. També va complicar bastant el projecte el fet de realitzar les Comunicacions amb USB. La recerca d'informació sobre els diferents tipus de comunicacions USB i la implementació d'aquestes per a poder transmetre dades mitjançant el bus m'ha ocupat la major part del projecte.

Un cop amb les comunicacions solucionades, tot ha estat menys difícil. El desenvolupament del software que gestiona el Microcontrolador, i del software del PC, no ha estat complicat. Evidentment, m'han generat dubtes, però el poder disposar d'internet per a realitzar consultes, ha fet que els pogués solucionar ràpidament. Segurament, el fet de disposar d'una trama robusta per a enviar i rebre dades, m'ha ajudat a no preocupar-me per la transmissió, i amb això he aconseguit disposar de més temps per a optimitzar els programes.

Per acabar, m'agradaria dir que en l'aspecte d'aprenentatge personal, la realització del projecte m'ha servit molt, planificar un mètode de treball i portar-lo a terme, la recerca d'informació, la utilització de softwares i hardwares que no havia vist pràcticament mai, etc. Ha estat un treball bastant llarg, i en alguns aspectes penso que complicat, però que també m'ha aportat molts nous coneixements.

WEBGRAFIA

- **C Compiler Reference Manual,** June 2008 (pdf)

- **PIC18F2455/2550/4455/4550 *Data sheet* ,** 28/40/44-Pin, High Performance, Enhanced Flash, USB Microcontrollers with nanoWatt Technology (pdf)

- **Manual Microsoft Visual Basic 6.0** (pdf)

- http://www.microchip.com

- http://www.ccsinfo.com

- http://www.datasheetcatalog.com

- http://www.todopic.com.ar

ANNEXES

En aquest apartat s'hi troba informació complementària que servirà per a completar la memòria del Treball. En el primer punt s'especifica pas a pas com s'ha d'instal·lar el driver en el PC, punt imprescindible per a poder connectar el PC i el Microcontrolador. Els dos següents punts mostren el Codi de programa del Microcontrolador i el Codi de programa de Visual Basic 6.0. A continuació i fins a finalitzar l'apartat, hi ha els *data sheet* dels components utilitzats, que han servit per a calcular diferents paràmetres del sistema.

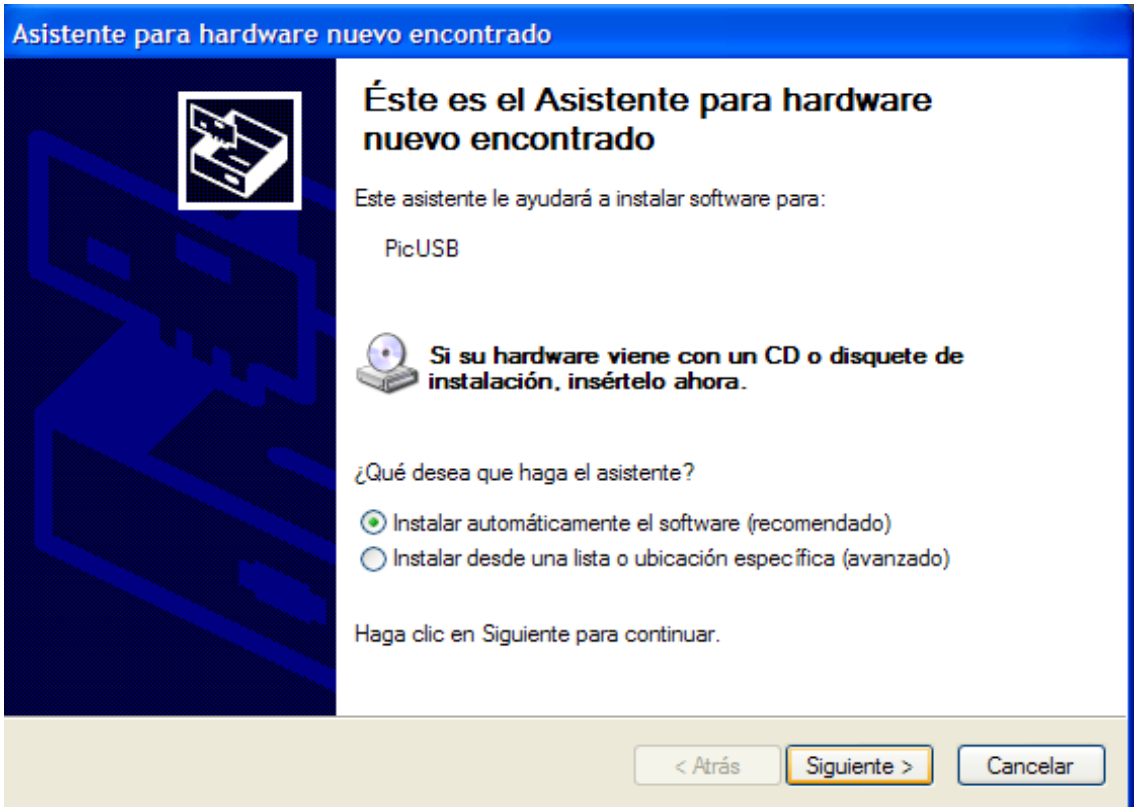
En els punts on s'exposa els codis del Microcontrolador i del PC, hi ha frases marcades de color verd. Aquestes frases no formen part del codi de programa, són comentaris afegits a aquest.

ANNEX 1: INSTAL·LACIÓ DEL DRIVER USB

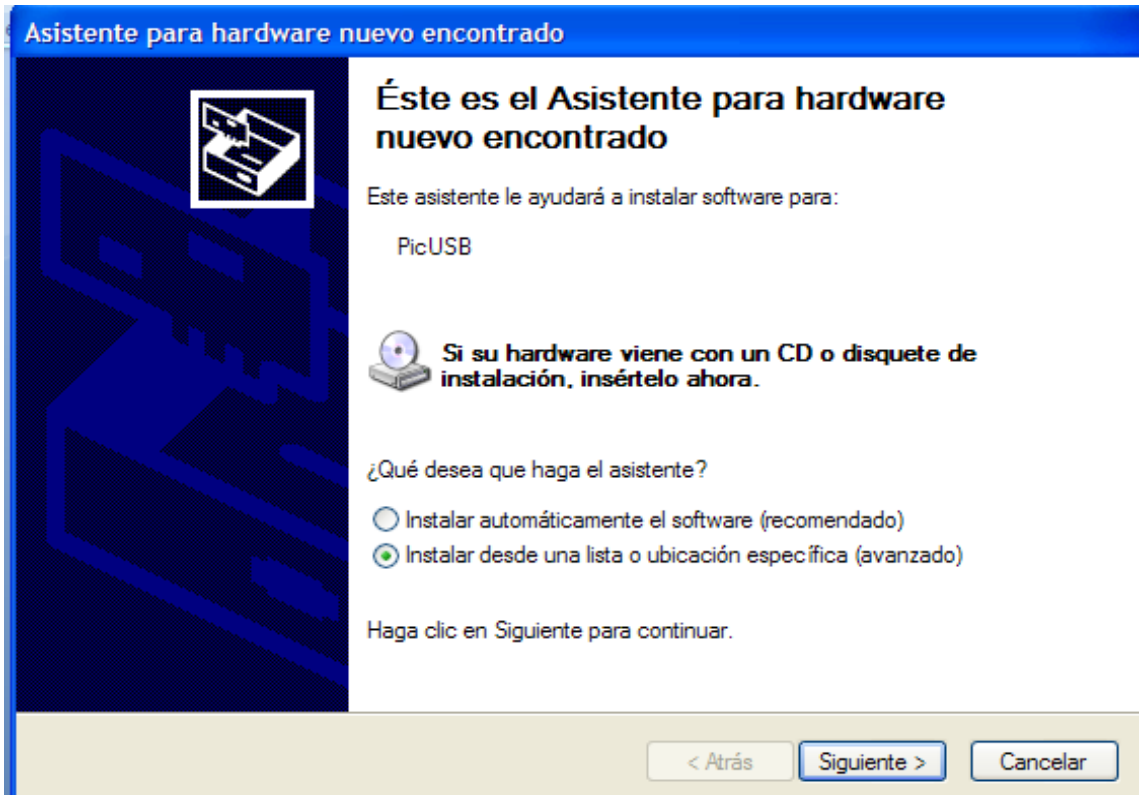
Un cop es tingui el software del microcontrolador PIC correctament gravat i es vulgui connectar al PC, s'haurà de fer ús del driver. Aquest driver s'haurà d'instal·lar en el PC i serà el que permetrà que l'ordinador reconegui el microcontrolador com a un dispositiu vàlid i permeti la comunicació amb ell. Aquest driver el subministra Microchip i només és vàlid per al Microsoft Windows XP.

Els procediments a seguir seran els següents:

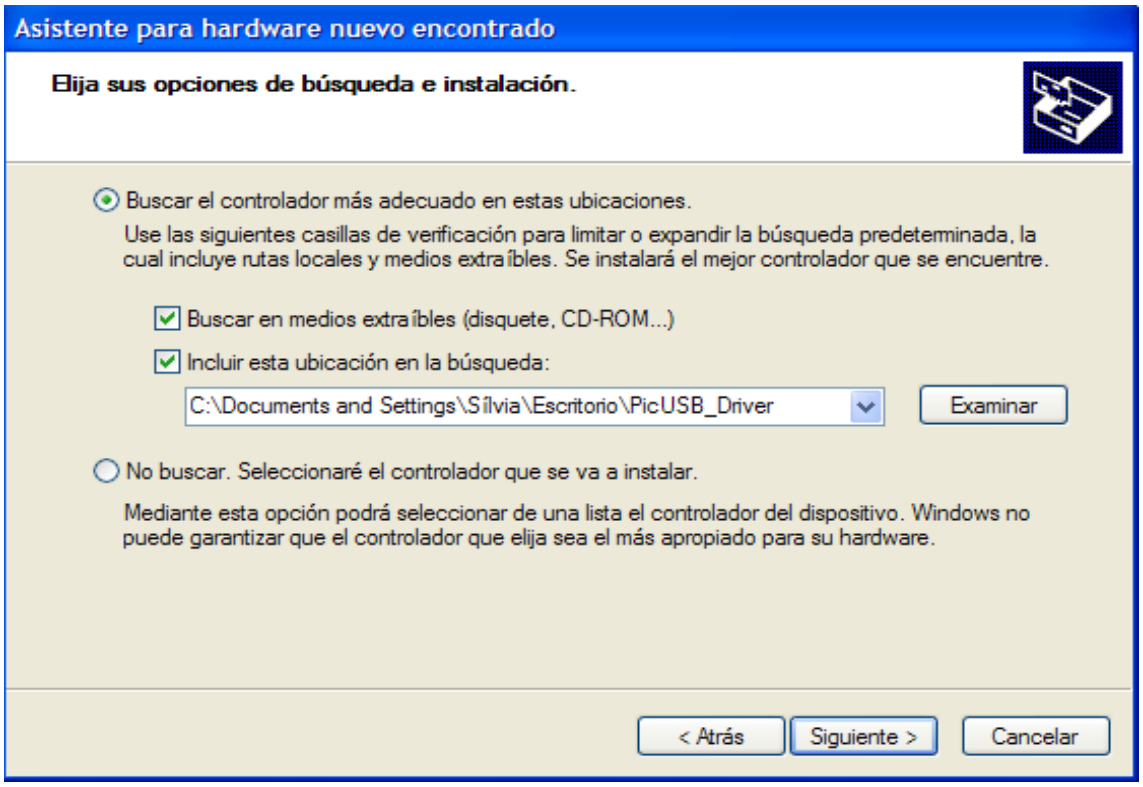
Es connecta el cable USB per una banda al connector de la nostra placa i per l'altra banda al PC. Si els pins del connector USB de la nostra placa estan ben connectats, a la part dreta inferior del nostre PC hi apareixerà el missatge de "nou hardware trobat". En aquest moment el nostre PC el que està rebent és el VID&PID que s'ha gravat en el microcontrolador i automàticament es posarà a buscar un driver instal·lat que correspongui amb aquesta identificació. Com que no el trobarà, ens demanarà que l'hi indiquem la ubicació.



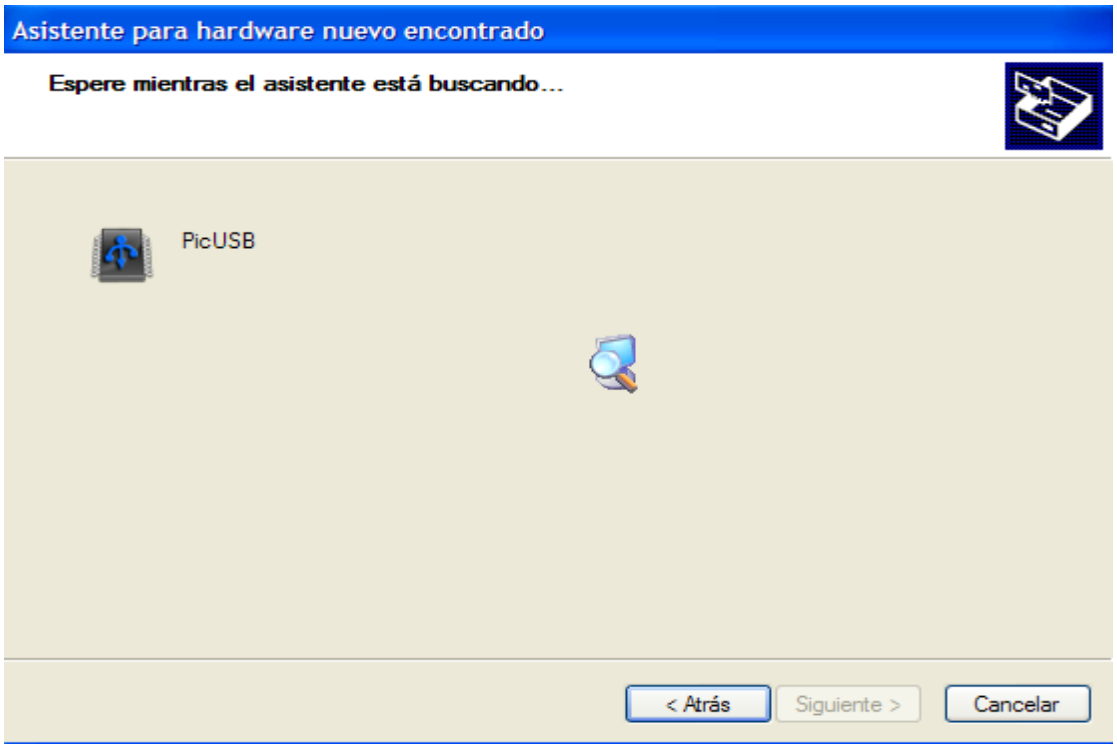
L'hi indiquem que el volem instal·lar des d'una llista o ubicació específica.



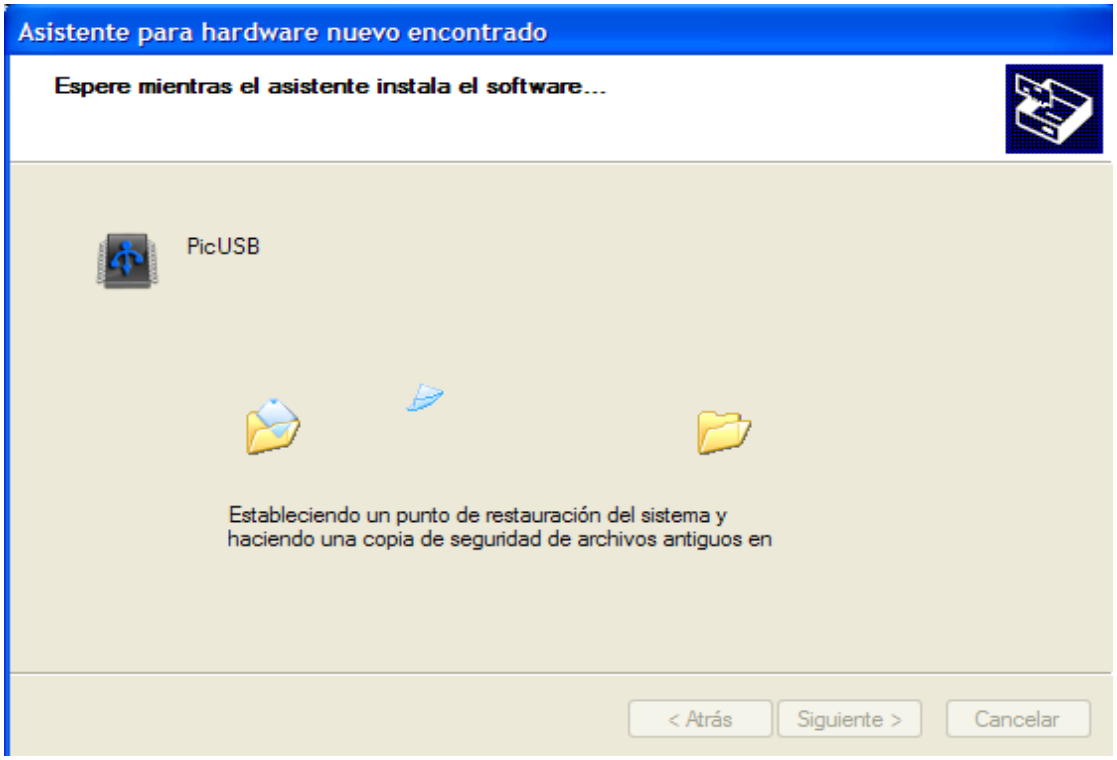
Se'ns obrirà un altre quadre on ens demanarà la ubicació. Cliquem a "examinar" i l'hi indiquem on tenim la carpeta amb el driver.

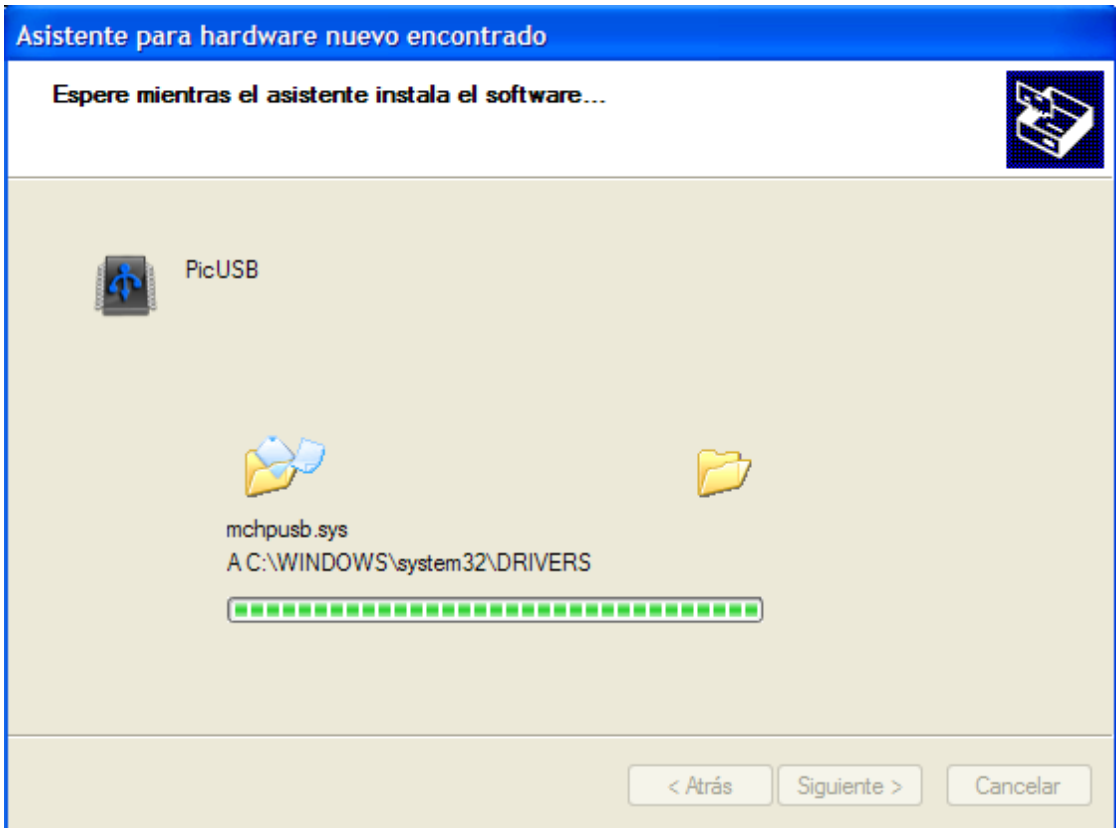


L'ordinador comprovarà si el driver indicat és el correcte.

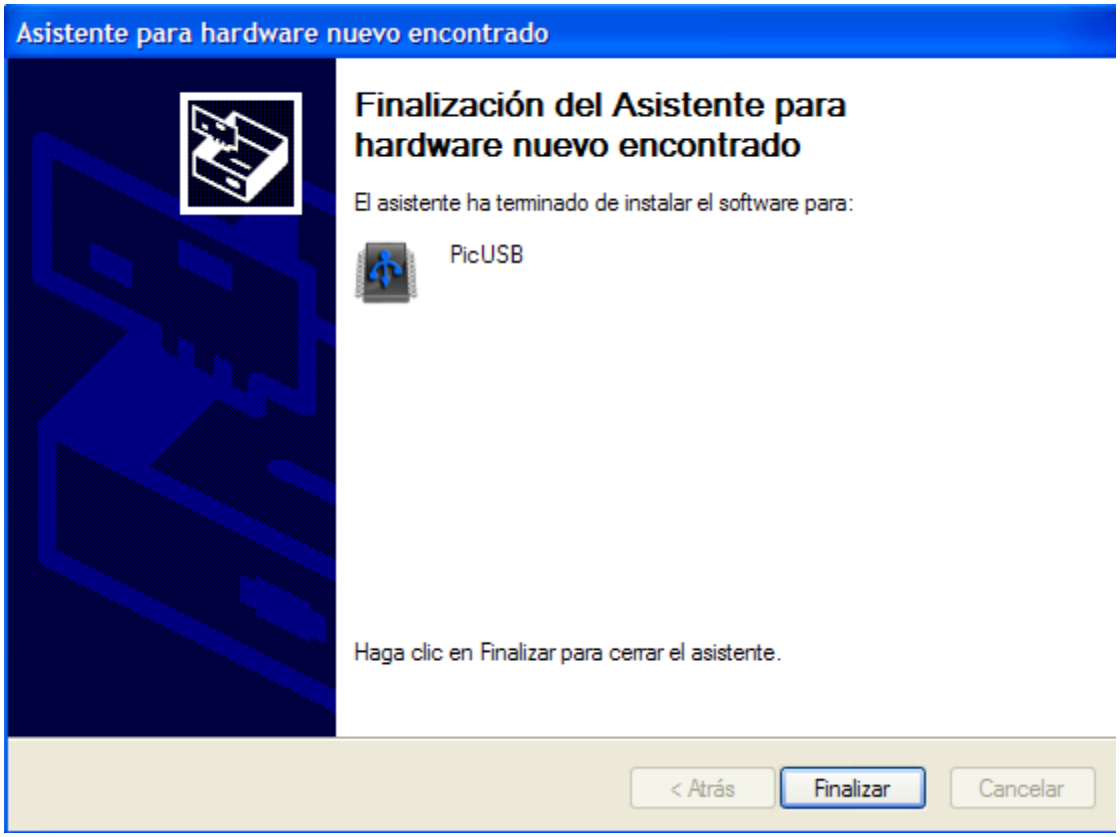


Seguidament, començarà a instal·lar el driver.

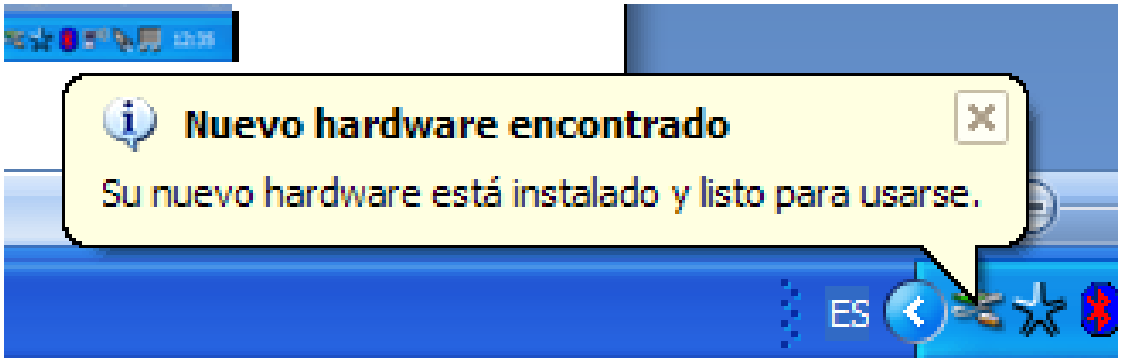




Un cop instal·lat ens apareixerà la següent pantalla. Cliquem a la opció “Finalitzar”.



A la part dreta inferior de la pantalla se’ns obrirà el següent quadre.



Ja tenim el driver correctament instal·lat i a punt de funcionar.

ANNEX 2: CODI DE PROGRAMA DEL MICROCONTROLADOR

```
#include <18F2550.h>
#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,VREGEN
#use delay(clock=20000000)

#include ".\include\usb_cdc.h"
#include "string.h"
#include "stdlib.h"

#define LED_VERD       PIN_A0
#define LED_GROC       PIN_A1
#define LED_VERMELL    PIN_A2
#define LED_GROC2      PIN_A3

#define ALIMENTACIO_PISTA1   PIN_A4
#define ALIMENTACIO_PISTA2   PIN_A5

#define Engega    Output_High
#define Apaga     Output_Low
#define Commuta   Output_Toggle

#define SOH       0x01
#define ETX       0x03
#define ID            '1'


////////////////////////////////////////////////////////////////
////////   Defineixo variables globals   ////////
////////////////////////////////////////////////////////////////

int centessima = 0, segon = 0;
int centessimes_pista_1 = 0, segons_pista_1 = 0;
int centessimes_pista_2 = 0, segons_pista_2 = 0;

unsigned char TramaEnviar[14];

unsigned char TramaRebuda[14];

unsigned char IntToHex[4];

unsigned char IntToHex1[4];

unsigned int HexToInt;

unsigned char Ordre_a_realitzar;

char TramaRebudaOK = 0;

unsigned int CrcRebut = 0;

unsigned int Crc;

unsigned int16 Bloc1 = 0;
```

```
unsigned int16 Bloc2= 0;

int TestComunicacions = 0;

int TempsPista1 = 0;

int TempsPista2 = 0;

int IniciPractica = 0;

int IniciCronoCarrera = 0;

int ComparacioTramaOK = 0;

int flag = 0;

int contador = 0;

int flag2 = 0;

int contador2 = 0;

int PararPista = 0;


////////////////////////////////////////////////////////////////////////////////
///////   Funcions a realitzar segons Ordre rebut  ////////////
////////////////////////////////////////////////////////////////////////////////

void Configurar()
{
        switch(Ordre_a_realitzar)
        {
        case 0: TestComunicacions = 1;//Aquest Ordre retorna la mateixa trama rebuda
                break;

        case 10: Engega(ALIMENTACIO_PISTA1);       //Mode practica, activar les dues pistes
                 Engega(ALIMENTACIO_PISTA2);
                 break;

        case 11: Engega(ALIMENTACIO_PISTA1);       //Mode practica, activar la pista1
                 break;

        case 12: Engega(ALIMENTACIO_PISTA2);       //Mode practica, activar la pista2
                 break;


        case 20: delay_ms(3000);    //Mode practica cronometrada, activar les dues pistes
                 enable_interrupts(INT_EXT);
                 enable_interrupts(INT_EXT1);
                 Engega(ALIMENTACIO_PISTA1);
                 Engega(ALIMENTACIO_PISTA2);
                 IniciPractica = 1;
                 break;
```

```
case 21: delay_ms(3000);      //Mode practica cronometrada, activar la pista1
         enable_interrupts(INT_EXT);
         Engega(ALIMENTACIO_PISTA1);
         IniciPractica = 1;
         break;

case 22: delay_ms(3000);     //Mode practica cronometrada, activar la pista2
         enable_interrupts(INT_EXT1);
         Engega(ALIMENTACIO_PISTA2);
         IniciPractica = 1;
         break;



case 31: delay_ms(3000);   //Mode crono, semafor, activar pista1 i anar passant temps
         Engega(LED_VERD);
         delay_ms(2000);
         Engega(LED_GROC);
         delay_ms(2000);
         Engega(LED_VERMELL);
         delay_ms(2000);
         Apaga(LED_VERD);
         Apaga(LED_GROC);
         Apaga(LED_VERMELL);
         disable_interrupts(INT_EXT1);
         enable_interrupts(INT_EXT);
         Engega(ALIMENTACIO_PISTA1);
         IniciCronoCarrera = 1;
         break;

case 32: delay_ms(3000);  //Mode crono, semafor, activar pista2 i anar passant temps
         Engega(LED_VERD);
         delay_ms(2000);
         Engega(LED_GROC);
         delay_ms(2000);
         Engega(LED_VERMELL);
         delay_ms(2000);
         Apaga(LED_VERD);
         Apaga(LED_GROC);
         Apaga(LED_VERMELL);
         disable_interrupts(INT_EXT);
         enable_interrupts(INT_EXT1);
         Engega(ALIMENTACIO_PISTA2);
         IniciCronoCarrera = 1;
         break;



case 40: delay_ms(3000);    //Mode carrera, semàfor, activar dues pistes passar temps
         Engega(LED_VERD);
         delay_ms(2000);
         Engega(LED_GROC);
         delay_ms(2000);
         Engega(LED_VERMELL);
```

```
                delay_ms(2000);
                Apaga(LED_VERD);
                Apaga(LED_GROC);
                Apaga(LED_VERMELL);
                enable_interrupts(INT_EXT);
                enable_interrupts(INT_EXT1);
                Engega(ALIMENTACIO_PISTA1);
                Engega(ALIMENTACIO_PISTA2);
                IniciCronoCarrera = 1;
                break;



        case 50: Apaga(ALIMENTACIO_PISTA1);          //Apaga alimentacions
                Apaga(ALIMENTACIO_PISTA2);
                break;

        case 80: if(PararPista == 1)        //Apaga alimentacions quan acaba carrera o crono.
                {
                        Apaga(ALIMENTACIO_PISTA1);
                }
                if(PararPista == 2)
                {
                        Apaga(ALIMENTACIO_PISTA2);
                }
        }
}



////////////////////////////////////////////////////////////////////////////////
/////   Funció per construir la trama a enviar al PC   /////
////////////////////////////////////////////////////////////////////////////////

void ConstruirTramaEnviarPC(void)
{

        unsigned char j;

        TramaEnviar[0] = SOH;

        TramaEnviar[1] = ID;

        TramaEnviar[2] = '0' + Ordre_a_realitzar;


        ///////////   ConvertirIntToHex(Bloc1)   ////////

        IntToHex[0] = '0' + (Bloc1 % 16);
        if (IntToHex[0] > '9')                          ///// El caracter Hexadecimal és un lletra ////
        IntToHex[0] += 7;

        IntToHex[1] = '0' + ((Bloc1 % 256) / 16);
        if (IntToHex[1] > '9')
```

```
IntToHex[1] += 7;

IntToHex[2] = '0' + ((Bloc1 % 4096) / 256);
if (IntToHex[2] > '9')
IntToHex[2] += 7;

IntToHex[3] = '0' + (Bloc1 / 4096);
if (IntToHex[3] > '9')
IntToHex[3] += 7;

TramaEnviar[3] = IntToHex[3];
TramaEnviar[4] = IntToHex[2];
TramaEnviar[5] = IntToHex[1];
TramaEnviar[6] = IntToHex[0];


//////////   ConvertirIntToHex(Bloc2)   ///////

IntToHex[0] = '0' + (Bloc2 % 16);
if (IntToHex[0] > '9')                          ///// El caracter Hexadecimal és un lletra ////
IntToHex[0] += 7;

IntToHex[1] = '0' + ((Bloc2 % 256) / 16);
if (IntToHex[1] > '9')
IntToHex[1] += 7;

IntToHex[2] = '0' + ((Bloc2 % 4096) / 256);
if (IntToHex[2] > '9')
IntToHex[2] += 7;

IntToHex[3] = '0' + (Bloc2 / 4096);
if (IntToHex[3] > '9')
IntToHex[3] += 7;


TramaEnviar[7] = IntToHex[3];
TramaEnviar[8] = IntToHex[2];
TramaEnviar[9] = IntToHex[1];
TramaEnviar[10] = IntToHex[0];

TramaEnviar[11] = ETX;

Crc = 0;
for (j=1;j<12;j++)
{
Crc = Crc + TramaEnviar[j];
}

////////////////////////////////////////////////////////////
//////////   ConvertirIntToHex(Crc)   ////////
////////////////////////////////////////////////////////////

IntToHex[0] = '0' + (Crc % 16);
if (IntToHex[0] > '9')                          ///// El caracter Hexadecimal és un lletra ////
IntToHex[0] += 7;
```

```
	IntToHex[1] = '0' + ((Crc % 256) / 16);
	if (IntToHex[1] > '9')
	IntToHex[1] += 7;

	IntToHex[2] = '0' + ((Crc % 4096) / 256);
	if (IntToHex[2] > '9')
	IntToHex[2] += 7;

	IntToHex[3] = '0' + (Crc / 4096);
	if (IntToHex[3] > '9')
	IntToHex[3] += 7;

	TramaEnviar[12] = IntToHex[1];    // Crc High
	TramaEnviar[13] = IntToHex[0];    // Crc Low

}


////////////////////////////////////////////////////////////////////////////
//////////   Funció per verificar la trama rebuda    /////
////////////////////////////////////////////////////////////////////////////

void VerificarTramaRebuda(void)
{
	char sortir = 1;
	char fase = 0;
	unsigned char c = 0;
	while (sortir!= 0)
	{

		switch(fase)
		{
			case 0:
				if (TramaRebuda[c] == SOH)
				{
					fase = 1;
					c++;
				}
				else {sortir = 0;}break;

			case 1:
				CrcRebut = 0;
				CrcRebut = CrcRebut + TramaRebuda[c];
				if (TramaRebuda[c] == ID)
				{
					fase = 2;
					c++;
				}
				else {sortir = 0;}break;

			case 2:
				CrcRebut = CrcRebut + TramaRebuda[c];
```

```
                Ordre_a_realitzar = TramaRebuda[c] - '0'; // Restem el '0' per
                obtenir el numero de Ordre
                fase = 3;c++;
                break;

        case 3:
                CrcRebut = CrcRebut + TramaRebuda[c];
                HexToInt = 0;
                if (TramaRebuda[c] <= '9')
                {
                        HexToInt = (unsigned int)((TramaRebuda[c] - 48) * 4096);
                }
                else
                {
                        HexToInt = (unsigned int)((TramaRebuda[c] - 55) * 4096);
                }
                fase = 4;c++;
                break;

        case 4:
                CrcRebut = CrcRebut + TramaRebuda[c];
                if (TramaRebuda[c] <= '9')
                {
                        HexToInt += (unsigned int)((TramaRebuda[c] - 48) * 256);
                }
                else
                {
                        HexToInt += (unsigned int)((TramaRebuda[c] - 55) * 256);
                }
                fase = 5;c++;
                break;

        case 5:
                CrcRebut = CrcRebut + TramaRebuda[c];
                if (TramaRebuda[c] <= '9')
                {
                        HexToInt += (unsigned int)((TramaRebuda[c] - 48) * 16);
                }
                else
                {
                        HexToInt += (unsigned int)((TramaRebuda[c] - 55) * 16);
                }
                fase = 6;c++;
                break;

        case 6:
                CrcRebut = CrcRebut + TramaRebuda[c];
                if (TramaRebuda[c] <= '9')
                {
                        HexToInt += (unsigned int)(TramaRebuda[c] - 48);
                        PararPista = HexToInt;
                }
                else
                {
                        HexToInt += (unsigned int)(TramaRebuda[c] - 55);
```

```
		}
		fase = 7;c++;
		break;

	case 7:
		CrcRebut = CrcRebut + TramaRebuda[c];
		HexToInt = 0;
		if (TramaRebuda[c] <= '9')
		{
			HexToInt = (unsigned int)((TramaRebuda[c] - 48) * 4096);
		}
		else
		{
			HexToInt = (unsigned int)((TramaRebuda[c] - 55) * 4096);
		}
		fase = 8;c++;
		break;

	case 8:
		CrcRebut = CrcRebut + TramaRebuda[c];
		if (TramaRebuda[c] <= '9')
		{
			HexToInt += (unsigned int)((TramaRebuda[c] - 48) * 256);
		}
		else
		{
			HexToInt += (unsigned int)((TramaRebuda[c] - 55) * 256);
		}
		fase = 9;c++;
		break;

	case 9:
		CrcRebut = CrcRebut + TramaRebuda[c];
		if (TramaRebuda[c] <= '9')
		{
			HexToInt += (unsigned int)((TramaRebuda[c] - 48) * 16);
		}
		else
		{
			HexToInt += (unsigned int)((TramaRebuda[c] - 55) * 16);
		}
		fase = 10;c++;
		break;

	case 10:
		CrcRebut = CrcRebut + TramaRebuda[c];
		if (TramaRebuda[c] <= '9')
		{
			HexToInt += (unsigned int)(TramaRebuda[c] - 48);
		}
		else
		{
			HexToInt += (unsigned int)(TramaRebuda[c] - 55);
		}
		fase = 11;c++;
```

```
                        break;

                case 11:
                        if (TramaRebuda[c] == ETX)
                        {
                                CrcRebut = CrcRebut + TramaRebuda[c];

                                IntToHex[0] = '0' + (CrcRebut % 16);
                                if (IntToHex[0] > '9')
                                IntToHex[0] += 7;

                                IntToHex[1] = '0' + ((CrcRebut % 256) / 16);
                                if (IntToHex[1] > '9')
                                IntToHex[1] += 7;

                                IntToHex[2] = '0' + ((CrcRebut % 4096) / 256);
                                if (IntToHex[2] > '9')
                                IntToHex[2] += 7;

                                IntToHex[3] = '0' + (CrcRebut / 4096);
                                if (IntToHex[3] > '9')
                                IntToHex[3] += 7;

                                fase = 12;
                                c++;

                        }
                        else {sortir = 0;}break;

                case 12:
                        if (TramaRebuda[c] == IntToHex[1])
                        {
                                fase = 13;
                                c++;
                        }
                        else {sortir = 0;}break;

                case 13:
                        if (TramaRebuda[c] == IntToHex[0])
                        {
                                // La trama s'ha rebut correctament ja podem enviar la
                                confirmacio i executar el Ordre
                                fase = 0;
                                TramaRebudaOK = 1;
                                sortir = 0;

                        }
                        else {sortir = 0;} break;
        }
    }
}
```

```
////////////////////////////////////////////////////////
//////////   Funció interrupció RTCC   ////
////////////////////////////////////////////////////////

#INT_RTCC
void clock_isr()
{
        ++centessima;           /// Cada vegada que entrem incrementem una centèssima
        set_RTCC(5500);         /// Donem el valor d'inici al Timer0
        if(centessima == 100)
        {
                ++segon;        ///Cada 100 centèssimes toca incrementar un segon
                centessima = 0; ///Resetejem les centèssimes
        }
}


////////////////////////////////////////////////////////////////
//////////  Funció interrupció TIMER1  //////////////
////////////////////////////////////////////////////////////////

#INT_TIMER1
void timer1()
{
        contador++;     ///Vaig incrementant fins a 20 vegades
        set_timer1(0);
        if(contador == 20)   ///Amb contador = 20 tenim una temporització d'un segon aprox.
        {
                contador = 0;
                flag = 1;    ///Aquesta variable permet registrar una nova lectura del sensor1
                set_timer1(T1_DISABLED);  ///Deshabilitem les interrupcions del timer1
        }
}


////////////////////////////////////////////////////////////////
//////////   Funció interrupció Timer3   //////////////
////////////////////////////////////////////////////////////////

#INT_TIMER3
void timer3()
{
        contador2++;   ///Vaig incrementant fins a 20 vegades
        set_timer3(0);
        if(contador2 == 20)  ///Amb contador = 20 tenim una temporització d'un segon aprox.
        {
                contador2 = 0;
                flag2 = 1;  ///Aquesta variable permet registrar una nova lectura del sensor2
                set_timer3(T3_DISABLED);  ///Deshabilitem les interrupcions del timer3
        }
}
```

```
////////////////////////////////////////////////////////////////////////
//////////   Funció interrupció externa 0    ///////////////////
////////////////////////////////////////////////////////////////////////

#int_EXT  ///Cada vegada que el sensor detecta un cotxe entrem en aquesta funció
void EXT_isr()
{
        if(flag == 1) ///Només si el flag és = 1 comptabilitzarem la lectura, és per evitar rebots
        {
                centessimes_pista_1 = centessima; ///Guardem valor centèssimes actuals
                segons_pista_1 = segon;   ///Guardem valor segons actuals
                TempsPista1 = 1;          ///Serveix per enviar la trama de temps en el MAIN
        }
}

////////////////////////////////////////////////////////////////////////
//////////   Funció interrupció externa 1    ///////////////////
////////////////////////////////////////////////////////////////////////


#int_EXT1   ///Cada vegada que el sensor detecta un cotxe entrem en aquesta funció
void EXT1_isr()
{
        if(flag2 == 1)  ///Només si el flag2 és = 1 comptabilitzarem la lectura, és per evitar rebots
        {
                centessimes_pista_2 = centessima;  ///Guardem valor centèssimes actuals
                segons_pista_2 = segon;  ///Guardem valor segons actuals
                TempsPista2 = 1;   ///Serveix per enviar la trama de temps en el MAIN
        }
}



////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////   Funcions per Rebre i Enviar Trames   //////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////


void RebreTrama()
{
        char s;   ///Defineixo una variable interna s
        for(s=0;s<=13;s++)   ///La variable s pren valors desde 0 fins a 13
        {
                TramaRebuda[s] = usb_cdc_getc();  ///llegeixo cada caràcter de la trama
        }
}
```

```
void EnviarTrama()
{
	char s;    ///Defineixo una variable interna s
	if(usb_cdc_putready())   ////Si  el bus USB està lliure
	{
		for(s=0;s<=13;s++)   ///Donem valors a s des de 0 fins a 13
		{
			usb_cdc_putc(TramaEnviar[s]);  ///Envio els 14 caràcters de la trama
		}
	}
}



void IgualarTrama()   ///Funció per crear la trama per respondre al PC quan la trama rebuda és correcte
{
	TramaEnviar[0] = TramaRebuda[0];
	TramaEnviar[1] = TramaRebuda[1];
	TramaEnviar[2] = TramaRebuda[2];
	TramaEnviar[3] = TramaRebuda[3];
	TramaEnviar[4] = TramaRebuda[4];
	TramaEnviar[5] = TramaRebuda[5];
	TramaEnviar[6] = TramaRebuda[6];
	TramaEnviar[7] = TramaRebuda[7];
	TramaEnviar[8] = TramaRebuda[8];
	TramaEnviar[9] = TramaRebuda[9];
	TramaEnviar[10] = TramaRebuda[10];
	TramaEnviar[11] = TramaRebuda[11];
	TramaEnviar[12] = TramaRebuda[12];
	TramaEnviar[13] = TramaRebuda[13];
}



////////////////////////////////////////////////////////////////////////////////
//////////////    Funció per inicialitzar variables    ////////////////////
////////////////////////////////////////////////////////////////////////////////

void InicialitzarVariables()
{
	Bloc1 = 0;
	Bloc2 = 0;
	Segons_Pista_1 = 0;
	Centessimes_Pista_1 = 0;
	Segons_Pista_2 = 0;
	Centessimes_Pista_2 = 0;
}
```

```
////////////////////////////////////////////////////////////////////////
//////////////////   Programa principal   ///////////////////////
////////////////////////////////////////////////////////////////////////

void main()
{
        usb_cdc_init();         //// Inicialitzem USB CDC
        usb_init();

        Apaga(LED_VERD);
        Apaga(LED_GROC);
        Apaga(LED_VERMELL);
        Apaga(LED_GROC2);
        Apaga(ALIMENTACIO_PISTA1);
        Apaga(ALIMENTACIO_PISTA2);

        setup_counters( RTCC_INTERNAL,  RTCC_DIV_2 );       //// Definim Timers
        setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
        setup_timer_3(T3_INTERNAL|T3_DIV_BY_8);

        enable_interrupts(INT_RTCC);      ///// Habilito interrupció timer0
        enable_interrupts(global);        ///// Habilito interrupcions globals

        ext_int_edge(0, H_TO_L );         ///// Defineixo com detecta interrupció externa
        ext_int_edge(1, H_TO_L );

        port_b_pullups(TRUE);             ///// Habilito pull-ups interns per entrades interrupció

        set_tris_a(0x00);                 //// Defineixo el port A com a sortida
        set_tris_b(0xFF);                 //// Defineixo el port B com a Entrada

        while(TRUE)
        {
                if (usb_enumerated())    //// Funció que trona 1 si el PC reconeix l'USB
                {
                        if(usb_cdc_kbhit())        /// Retorna 1 si hi ha un caràcter al buffer
                        {

                                RebreTrama();  /// Cridem la funció Rebre Trama
                                VerificarTramaRebuda();
        if(TramaRebudaOK == 1&&Ordre_a_realitzar != 60&&Ordre_a_realitzar !=70)
                                {
                                        TramaRebudaOK = 0;     /// Si la trama rebuda és correcte
                                        IgualarTrama();            /// retornem la mateixa trama
                                        EnviarTrama();
                                        configurar();      /// Configurem Mode de Funcionament
                                }
                        }


                        if(IniciPractica == 1)   /// Enviar trama inici pràctica cronometrada
                        {
                                IniciPractica = 0;
                                InicialitzarVariables();
                                while(TempsPista1 != 1 || TempsPista2 != 1);
```

```
            TempsPista1 = 0;
            TempsPista2 = 0;
            Centessima = 0;
            Segon = 0;
            set_RTCC(5500);
            set_timer1(0);
            set_timer3(0);
            Ordre_a_realitzar = 60;
            ConstruirTramaEnviarPC();
            EnviarTrama();
}

if(IniciCronoCarrera == 1)    //// Enviar trama inici crono o carrera
{
            IniciCronoCarrera = 0;
            InicialitzarVariables();
            Ordre_a_realitzar = 60;
            ConstruirTramaEnviarPC();
            EnviarTrama();

            Centessima = 0;
            Segon = 0;
            enable_interrupts(INT_TIMER1);
            enable_interrupts(INT_TIMER3);
            set_RTCC(5500);
            set_timer1(0);
            set_timer3(0);
}

if(TestComunicacions == 1)   //// Retornem la mateixa trama
{
            TestComunicacions = 0;
            IgualarTrama();
            EnviarTrama();
}



if(TempsPista1 == 1)        ////  Si ha interromput pista1
{
            flag = 0;
            TempsPista1 = 0;
            set_timer1(0);
            enable_interrupts(INT_TIMER1);
            Bloc1 = Segons_Pista_1;
            Bloc2 = Centessimes_Pista_1 + 256;
            Ordre_a_realitzar = 70;
            ConstruirTramaEnviarPC();
            EnviarTrama();
            InicialitzarVariables();
}

if(TempsPista2 == 1)      ////  Si ha interromput pista2
{
```

```
                                TempsPista2 = 0;
                                flag2 = 0;
                                set_timer3(0);
                                enable_interrupts(INT_TIMER3);
                                Bloc1 = Segons_Pista_2;
                                Bloc2 = Centessimes_Pista_2 + 512;
                                Ordre_a_realitzar = 70;
                                ConstruirTramaEnviarPC();
                                EnviarTrama();
                                InicialitzarVariables();
                        }

                }
        }
}
```

ANNEX 3: CODI DE PROGRAMA DE VISUAL BASIC 6.0

'El codi del programa de Visual Basic 6.0 està dividit en diferents Mòduls que es poden trobar tots ells a continuació

Mòdul declaració de variables

```
Option Explicit

Public Rell As New Rellotge
Public FitxerConfig As New Fitxer
Public Estat As Estats
Public VoltesCursa As Integer
Public Volta1 As Integer
Public Volta2 As Integer
Public VoltaRapida1 As Integer
Public VoltaRapida2 As Integer
Public TempsVoltaRapida1 As Double
Public TempsVoltaRapida2 As Double
Public SegonsAnt1 As Double
Public SegonsAnt2 As Double
Public LagTrama As Double
Public Temps As Double
Public PrimerCop As Boolean
Public Pistes As Byte
Public Tipus As Byte
Public PunterBuffer As Byte
Public PunterLectura As Byte
Public Buffer(0 To 25) As String
Public PendentFinalitzarPista As Byte
Public TempBackgroudLabelPista As Long
Public FormulariCancelat As Boolean

Public Enum Estats
   Comunicacionsparades = 0
   PendentEscollirMode = 1
   Practica = 2
   PracticaCrono = 3
   Crono = 4
   Carrera = 5
End Enum
```

Mòdul de Comunicacions

```
Option Explicit

Public BufferEntrada As String  'Declaro les variables amb les que treballaré
Public TramaRebuda As String
Public TramaRecepcionada As Boolean
Public TramaRebudaOk As Boolean
Public BlocDades1 As Long
Public BlocDades2 As Long
Public TramaEnviada As String
Public OrdreRebut As Byte

Public Const SOH As Byte = &H1   'Defineixo el valor de SOH = 0x01

'Funció per enviar Ordre al Microcontrolador
Public Function EnviaComando(ByRef Serie As MSComm, Optional ByVal Placa As
Integer = 0, Optional ByVal Ordre As Byte = 0, Optional ByVal Dada1 As Long = 0,
Optional ByVal Dada2 As Long = 0) As Byte
On Error GoTo ControlErrors
Dim tmp As String
Dim r As Integer

'Poso límits als caràcters i als blocs de dades de la trama
   If Placa > 225 Or Placa < 0 Or Ordre < 0 Or Ordre > 225 Or Dada1 < 0 Or Dada1 >
65535 Or Dada2 < 0 Or Dada2 > 65535 Then
      EnviaOrdre = 4
      Exit Function
   End If

      'Crido la funció GenerarTrama,li passo els paràmetres i ho guardo a TramaEnviada
   TramaEnviada = GenerarTrama(Placa, Ordre, Dada1, Dada2)
      'Crido la funció EnviaTrama, que envia per el port sèrie (sèrie virtual) els
      paràmetres TramaEnviada
   EnviaOrdre = EnviaTrama(Serie, TramaEnviada)

   Exit Function

ControlErrors:
  r = MsgBox("Error: " & Err.Number & vbCrLf & Err.Description, vbCritical, Err.Source)
End Function

'Funció per enviar la Trama
Public Function EnviaTrama(ByRef Serie As MSComm, ByVal Trama As String) As Byte
Dim r As Integer
On Error GoTo ControlErrors

   OrdreRebut = 0   'Inicialitzo variables
   BlocDades1 = 0
   BlocDades2 = 0
```

```
    TramaEnviada = Trama     'Guardo els valors de TramaEnviada a Trama
    Serie.Output = Trama    'Envio trama

    Exit Function

ControlErrors:
    If Serie.PortOpen = True Then Serie.PortOpen = False
    r = MsgBox("Error: " & Err.Number & vbCrLf & Err.Description, vbCritical, Err.Source)
    EnviaTrama = 3
End Function

' Sumem el valor decimal caràcter a caràcter d'una cadena per calcular el CRC
Public Function CRC(ByVal Cadena As String) As String
Dim CalculCRC As Integer
Dim r As Integer
On Error GoTo ControlErrors

    CalculCRC = 0
    For r = 1 To Len(Cadena) Step 1 'Passarem per tots els caràcters de la cadena

        CalculCRC = CalculCRC + Asc(Mid(Cadena, r, 1))  'Anem sumant el CalculCRC
    Next r

    CRC = Hex(CalculCRC)  'Convertim el CalculCRC a Hexadecimal
    If Len(CRC) = 1 Then CRC = "0" & CRC

    Exit Function

ControlErrors:
    r = MsgBox("Error: " & Err.Number & vbCrLf & Err.Description, vbCritical, Err.Source)
End Function

'Funció per generar Trama amb els valors que li passem
Public Function GenerarTrama(ByVal Placa As Integer, ByVal Ordre As Integer, ByVal
Dada1 As Long, ByVal Dada2 As Long) As String
Dim Trama, tmp As String
Dim CCRC As String
Dim r As Integer
On Error GoTo ControlErrors

    Trama = ""  'Buidem la Cadena de caràcters
    Trama = Chr$(SOH)   'El primer caràcter és SOH
    Trama = Trama & Chr$(Placa + Asc("0"))   'Aquest caràcter és l'identificador de Placa
    Trama = Trama & Chr$(Ordre + Asc("0")) 'El número de Ordre
    tmp = NumeroHexadecimal(Dada1, 4) 'El bloc de Dades1 en hexadecimal
    Trama = Trama & Mid(tmp, 1, 1)
    Trama = Trama & Mid(tmp, 2, 1)
    Trama = Trama & Mid(tmp, 3, 1)
    Trama = Trama & Mid(tmp, 4, 1)
    tmp = NumeroHexadecimal(Dada2, 4)  'El bloc de Dades2 en hexadecimal
```

```
    Trama = Trama & Mid(tmp, 1, 1)
    Trama = Trama & Mid(tmp, 2, 1)
    Trama = Trama & Mid(tmp, 3, 1)
    Trama = Trama & Mid(tmp, 4, 1)
    Trama = Trama & Chr$(&H3)   'Finalització de Trama amb el caràcter ETX
    CCRC = CRC(Mid(Trama, 2, Len(Trama) - 1))  'Cridem funció per calcular el CRC
    Trama = Trama & Mid(CCRC, Len(CCRC) - 1, 1)  'Valor CRC High
    Trama = Trama & Mid(CCRC, Len(CCRC), 1)  'Valor CRC Low

    GenerarTrama = Trama  'GenerarTrama té el valor de Trama (la que hem generat)

    Exit Function

ControlErrors:
    r = MsgBox("Error: " & Err.Number & vbCrLf & Err.Description, vbCritical, Err.Source)
End Function


'Funció per passar un caràcter de Hexadecimal a Decimal
Public Function NumeroDecimal(ByVal Cadena As String) As Double
Dim Caracter As String
Dim r As Integer
On Error GoTo ControlErrors

    NumeroDecimal = 0
    For r = 0 To Len(Cadena) - 1 Step 1
        Caracter = Mid(Cadena, r + 1, 1)
        If Asc(Caracter) < 65 Then
            NumeroDecimal = NumeroDecimal + (Val(Caracter) * 16 ^ (Len(Cadena) - r - 1))
        Else
            NumeroDecimal = NumeroDecimal + ((Asc(Caracter) - 55) * 16 ^ (Len(Cadena) - r -
1))
        End If
    Next

    Exit Function

ControlErrors:
    r = MsgBox("Error: " & Err.Number & vbCrLf & Err.Description, vbCritical, Err.Source)
End Function


'Funció per passar un número decimal a Hexadecimal
Public Function NumeroHexadecimal(ByVal Numero As Double, Optional ByVal
LongitudCadena As Byte = 4) As String
Dim Var As Double
Dim s As String
Dim r As Integer
On Error GoTo ControlErrors
```

```
    NumeroHexadecimal = "0"

    If Numero > ((2 ^ 16) - 1) Then          'Calculo el nivell alt
        Var = Numero / 2 ^ 16
        Var = Int(Var)
        NumeroHexadecimal = Hex(Var)         'Calculo el nivell baix
        Var = Numero - (Var * 2 ^ 16)
        s = Hex(Var)
        Do Until Len(s) >= 4
            s = "0" + s
        Loop
        NumeroHexadecimal = NumeroHexadecimal & s
    Else
        NumeroHexadecimal = Hex(Numero)
    End If

    Do Until Len(NumeroHexadecimal) >= LongitudCadena
        NumeroHexadecimal = "0" + NumeroHexadecimal
    Loop

    Exit Function

ControlErrors:
    r = MsgBox("Error: " & Err.Number & vbCrLf & Err.Description, vbCritical, Err.Source)
End Function

‘Funció que controla que els valors de la Trama que hem rebut siguin correctes
Public Function TramaRebudaCorrecte() As Boolean
Dim CRCEnviat, CRCRebut As String
Dim ValorRebut As Double

    TramaRebudaCorrecte = False
        ‘Calculem el CRC i comprovem que sigui correcte
    CRCRebut = Mid(TramaRebuda, Len(TramaRebuda) - 1, 2)
    CRCEnviat = CRC(Mid(TramaRebuda, 2, Len(TramaRebuda) - 3))
    CRCEnviat = Mid(CRCEnviat, Len(CRCEnviat) - 1, 2)

    If CRCEnviat = CRCRebut Then ‘si el CRC és correcte guardem valors
        TramaRebudaCorrecte = True
        BlocDades1 = NumeroDecimal(Mid(TramaRebuda, 4, 4))
        BlocDades2 = NumeroDecimal(Mid(TramaRebuda, 8, 4))
        OrdreRebut = Asc(Mid(TramaRebuda, 3, 1)) - Asc("0")
        TramaRebudaCorrecte = True
        TramaRebudaOk = True
    End If
    ‘Responem al Micro que la trama s’ha rebut correctament retornant la mateixa trama
    Select Case OrdreRebut
        Case 0:
        Case 10, 11, 12, 20, 21, 22, 31, 32, 40, 50, 80:
```

```
        Case 60:
            FMain.Timer1.Interval = 1
            Rell.IniciaTemporitzacio 7
            EnviaTrama FMain.Com, TramaRebuda
        Case 70:
            EnviaTrama FMain.Com, TramaRebuda
        Case Else:
            MsgBox "Ordre rebut " & OrdreRebut & " no està implementat", vbExclamation,
"Comunicacions"
    End Select
    TramaRecepcionada = False
End Function

'funció que espera que el micro respongui si la trama que li ha enviat és correcte
Public Function EsperaTramaRebudaCorrecte() As Boolean

    EsperaTramaRebudaCorrecte = False
    Rell.IniciaTemporitzacio 1
    Do
        DoEvents
    Loop Until Rell.TempsSuperat(1, 1) Or TramaRebudaOk
    If TramaRebudaOk Then EsperaTramaRebudaCorrecte = True
    TramaRebudaOk = False
End Function
```

Mòdul Altres funcions

```
Option Explicit

'Funció que dectecta els com i els mostra a escollir
Function DeteccioComS(Optional ComActiu As Byte = 0) As Boolean
On Error GoTo ControlErrors
Dim i, cnt As Byte
Dim Existeix As Boolean
Dim PortEstavaObert As Boolean

    DeteccioComS = False
    cnt = 0
    PortEstavaObert = FMain.Com.PortOpen
    For i = 1 To 20
        If i < 1 Then
            FMain.MCom(i).Visible = True
        Else
            FMain.MCom(i).Visible = False
        End If
        FMain.MCom(i).Checked = False
        Existeix = True
        If FMain.Com.PortOpen Then
            PortEstavaObert = FMain.Com.CommPort
```

```
            FMain.Com.PortOpen = False
        End If
        FMain.Com.CommPort = i
        If Existeix Then
            FMain.Com.PortOpen = True
            If FMain.Com.PortOpen Then
                cnt = cnt + 1
                FMain.Com.PortOpen = False
                FMain.MCom.Item(i).Visible = True
                FMain.MCom.Item(i).Enabled = True
            End If
        End If
    Next
    If cnt > 0 Then
        DeteccioComS = True
        If ComActiu = 0 Then ComActiu = FMain.Com.CommPort
        If FMain.MCom.Item(ComActiu).Enabled Then
            If PortEstavaObert Then
                FMain.Com.CommPort = PortEstavaObert
                FMain.Com.PortOpen = True
                FMain.MCom(ComActiu).Checked = True
            End If
        End If
    End If
    Exit Function
ControlErrors:
    Existeix = False
    Resume Next
End Function

'Funció que canvia el port al que ens conectem
Public Sub CanviarCom(ByVal Port As Byte)
Dim PortOpen As Boolean
Dim i As Byte

    For i = 1 To FMain.MCom.Count
        FMain.MCom(i).Checked = False
    Next

    PortOpen = FMain.Com.PortOpen
    If PortOpen Then FMain.Com.PortOpen = False
    FMain.Com.CommPort = Port
    FMain.MCom(Port).Checked = True
    If PortOpen Then FMain.Com.PortOpen = True
    FitxerConfig.SetParametre "Port", Port
    FMain.MCom(Port).Checked = True

End Sub

'Funció que controla l'estat en el que s'està i controla les opcions de cada estat
```

```
Public Sub CanviarEstat(ByVal EstatEscollit As Estats)
On Error GoTo ControlErrors
Dim i As Byte

   Select Case EstatEscollit
      Case Estats.Comunicacionsparades:

         FMain.Frame1.Enabled = False
         FMain.Frame2.Enabled = False
         FMain.Boto(0).Enabled = False
         FMain.Boto(1).Enabled = False
         FMain.Llista(0).Enabled = False
         FMain.Llista(1).Enabled = False
         FMain.Command1(0).Enabled = False
         FMain.Command1(1).Enabled = False
         If FMain.Com.PortOpen Then FMain.Com.PortOpen = False
         FMain.MConectar.Caption = "Conectar amb cronometrador"
         If FMain.Com.PortOpen = True Then FMain.Com.PortOpen = False
         Estat = EstatEscollit

      Case Estats.PendentEscollirMode:
         FMain.Command1(0).Enabled = True
         FMain.Command1(1).Enabled = True
         PunterBuffer = 0
         PunterLectura = 0
         For i = 0 To 25
            Buffer(0) = ""
         Next
         FMain.Timer1.Interval = 0
         PrimerCop = True
         FMain.Frame1.Enabled = True
         FMain.Frame2.Enabled = True
         FMain.Boto(0).Enabled = True
         FMain.Boto(1).Enabled = False
         FMain.Llista(0).Enabled = False
         FMain.Llista(1).Enabled = False
         FMain.MComunicacions.Enabled = False
         If Not FMain.Com.PortOpen Then FMain.Com.PortOpen = True
         FMain.MConectar.Caption = "Desconectar amb cronometrador"
         EnviaOrdre FMain.Com, 1, 50
         Rell.IniciaTemporitzacio 1
         If Not EsperaTramaRebudaCorrecte Then
            MsgBox "No s'ha pogut establir comunicació amb el cronometrador",
vbExclamation, "Error comunicacions"
            CanviarEstat Comunicacionsparades
            Estat = Estats.Comunicacionsparades
         Else
            LagTrama = Rell.SegonsTranscorreguts(1) / 2
         End If
         FMain.MComunicacions.Enabled = True
```

```
Case Estats.Practica:
   FMain.Llista(0).Enabled = False
   FMain.Llista(1).Enabled = False
   FMain.Command1(0).Enabled = True
   FMain.Command1(1).Enabled = True
   InicialitzacionsVariablesCarrera
   FMain.Label1.Caption = ""
   FMain.Label2(2).Caption = FMain.TipusCrono(Tipus).Caption
   If Pistes = 1 Or Pistes = 0 Then
      FMain.Command1(0).Enabled = True
      FMain.Llista(0).Enabled = True
      FMain.Llista(0).Clear
      FMain.Label2(0).Caption = "Volta 0"
   End If
   If Pistes = 2 Or Pistes = 0 Then
      FMain.Command1(1).Enabled = True
      FMain.Llista(1).Enabled = True
      FMain.Llista(1).Clear
      FMain.Label2(1).Caption = "Volta 0"
   End If
   FMain.Boto(1).Enabled = True
   FMain.Boto(0).Enabled = False
   Estat = EstatEscollit

Case Estats.PracticaCrono:
   FMain.Llista(0).Enabled = False
   FMain.Llista(1).Enabled = False
   FMain.Command1(0).Enabled = True
   FMain.Command1(1).Enabled = True
   InicialitzacionsVariablesCarrera
   FMain.Label1.Caption = ""
   FMain.Label2(2).Caption = FMain.TipusCrono(Tipus).Caption
   If Pistes = 1 Or Pistes = 0 Then
      FMain.Command1(0).Enabled = True
      FMain.Llista(0).Enabled = True
      FMain.Llista(0).Clear
      FMain.Label2(0).Caption = "Volta 0"
   End If
   If Pistes = 2 Or Pistes = 0 Then
      FMain.Command1(1).Enabled = True
      FMain.Llista(1).Enabled = True
      FMain.Llista(1).Clear
      FMain.Label2(1).Caption = "Volta 0"
   End If
   FMain.Boto(1).Enabled = True
   FMain.Boto(0).Enabled = False
   Estat = EstatEscollit

Case Estats.Crono:
```

```
            FMain.Llista(0).Enabled = False
            FMain.Llista(1).Enabled = False
            InicialitzacionsVariablesCarrera
            FMain.Label1.Caption = ""
            FMain.Label2(2).Caption = FMain.TipusCrono(Tipus).Caption
            If Pistes = 1 Or Pistes = 0 Then
               FMain.Command1(0).Enabled = True
               FMain.Llista(0).Enabled = True
               FMain.Llista(0).Clear
               FMain.Label2(0).Caption = "Volta 0/" & VoltesCursa
            End If
            If Pistes = 2 Or Pistes = 0 Then
               FMain.Command1(1).Enabled = True
               FMain.Llista(1).Enabled = True
               FMain.Llista(1).Clear
               FMain.Label2(1).Caption = "Volta 0/" & VoltesCursa
            End If
            FMain.Boto(1).Enabled = True
            FMain.Boto(0).Enabled = False
            Estat = EstatEscollit

         Case Estats.Carrera:
            FMain.Llista(0).Enabled = True
            FMain.Llista(1).Enabled = True
            PendentFinalitzarPista = False
            FMain.Command1(0).Enabled = True
            FMain.Command1(1).Enabled = True
            InicialitzacionsVariablesCarrera
            FMain.Label1.Caption = ""
            FMain.Label2(2).Caption = FMain.TipusCrono(Tipus).Caption
            FMain.Llista(0).Clear
            FMain.Label2(0).Caption = "Volta 0/" & VoltesCursa
            FMain.Llista(1).Clear
            FMain.Label2(1).Caption = "Volta 0/" & VoltesCursa
            FMain.Boto(1).Enabled = True
            FMain.Boto(0).Enabled = False
            Estat = EstatEscollit

         Case Else:
            CanviarEstat Estats.Comunicacionsparades
            Estat = Estats.Comunicacionsparades

      End Select

      Exit Sub
ControlErrors:
      CanviarEstat Estats.Comunicacionsparades
End Sub

‘Funció per inicialitzar les variables
```

```
Sub InicialitzacionsVariablesCarrera()

    SegonsAnt1 = 0
    SegonsAnt2 = 0
    Volta1 = 0
    Volta2 = 0
    TempsVoltaRapida1 = 1000000
    TempsVoltaRapida2 = 1000000
    VoltaRapida1 = 0
    VoltaRapida1 = 0
    FMain.Llista(0).Clear
    FMain.Llista(1).Clear
    FMain.Label1.Caption = ""
    FMain.Label2(0).Caption = "Volta 0/" & VoltesCursa
    FMain.Label2(1).Caption = "Volta 0/" & VoltesCursa
    FMain.Boto(1).Enabled = True
    FMain.Boto(0).Enabled = False
    FMain.Label9(0).BackColor = TempBackgroudLabelPista
    FMain.Label9(1).BackColor = TempBackgroudLabelPista

    If FMain.Pista(0).value = Checked Then
        FMain.Llista(0).Enabled = True
        FMain.Command1(0).Enabled = True
    End If

    If FMain.Pista(1).value = Checked Then
        FMain.Llista(1).Enabled = True
        FMain.Command1(1).Enabled = True

    End IfEnd Sub

'Funció per tractar els cronometratges rebuts i guardar-los en les variables
Function SegToString(ByVal Minuts As Integer, ByVal Segons As Double, ByVal
Centesimes As Double) As String
Dim m As Integer

    m = Segons \ 60
    Minuts = Minuts + m
    Segons = Segons - (m * 60)
    Segons = Segons + (Centesimes / 100)
    SegToString = IIf(Minuts <> 0, Minuts & """, "") & Replace(CStr(Round(Segons, 2)), ",",
""""")
End Function
```

Mòdul Rellotge

```
'Dins aquest mòdul hi ha funcions que  controlen el temps
Option Explicit

Dim ContadorDisponible As Boolean 'Si el sistema operatiu disposa d'aquest contador
```

```
Private Contador1(1 To 10), Contador2, Frequencia As Currency

Private Declare Function QueryPerformanceCounter Lib "Kernel32" (X As Currency) As
Boolean
Private Declare Function QueryPerformanceFrequency Lib "Kernel32" (X As Currency) As
Boolean


'Aquesta funció inicialitza el timer
Public Sub IniciaTemporitzacio(ByVal NumeroRellotge As Byte)

   If ContadorDisponible Then
      QueryPerformanceCounter Contador1(NumeroRellotge)
   End If
End Sub


Public Function SegonsTranscorreguts(ByVal NumeroRellotge As Byte) As Double

   If ContadorDisponible Then
      If Contador1(NumeroRellotge) <> 0 Then
         QueryPerformanceCounter Contador2
         If Contador2 < Contador1(NumeroRellotge) Then Contador1(NumeroRellotge) = 0
         SegonsTranscorreguts = (Contador2 - Contador1(NumeroRellotge)) / Frequencia
      Else
         SegonsTranscorreguts = 0
      End If
   End If
End Function


Public Function TempsSuperat(ByVal NumeroRellotge As Byte, ByVal TimeOut As
Double) As Boolean
Dim Temps As Double

   If ContadorDisponible Then
      TempsSuperat = False
      If Contador1(NumeroRellotge) <> 0 Then
         QueryPerformanceCounter Contador2
         If Contador2 < Contador1(NumeroRellotge) Then Contador1(NumeroRellotge) = 0
         Temps = (Contador2 - Contador1(NumeroRellotge)) / Frequencia
         If Temps > TimeOut Then TempsSuperat = True
      End If
   End If
End Function
```

Mòdul Fitxer

```
'En aquest mòdul hi ha funcions que es criden dins d'altres funcions al llarg del programa. La finalitat d'aquestes funcions és la d'estalviar haver d'escriure cada vegada el codi i tenir el programa molt més organitzat i accessible.
Option Explicit

Private Declare Function GetPrivateProfileString Lib "Kernel32" Alias "GetPrivateProfileStringA" (ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpDefault As String, ByVal lpReturnedString As String, ByVal nSize As Long, ByVal lpFileName As String) As Long
Private Declare Function WritePrivateProfileString Lib "Kernel32" Alias "WritePrivateProfileStringA" (ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpString As Any, ByVal lpFileName As String) As Long


Public ContingutFitxer As String
Public RutaFitxer As String
Private NumError As Integer
Private DesError As String
Private UltimParametre As String
Public SeparadorParametre As String
Public MissatgesErrorHabilitat As Boolean

Property Get Error() As Integer
   Error = NumError
End Property

Property Get DescripcioError() As Integer
   DescripcioError = DesError
End Property


Public Function GetParametre(ByVal Parametre As String, Optional NomFitxer As String = "") As String
On Error GoTo ControlErrors
Dim PosInicial, PosFinal, PosIniComentari As Variant

   UltimParametre = Parametre
   GestioError 0
   GetParametre = ""
   If NomFitxer <> "" Then RutaFitxer = NomFitxer
   If RutaFitxer = "" Then GestioError 1
   If NumError = 0 Then
      If ContingutFitxer = "" Then Me.CapturarContingutFitxer RutaFitxer
      If ContingutFitxer <> "" And NumError = 0 Then
         PosInicial = 0
         PosFinal = 0
         PosInicial = InStr(1, ContingutFitxer, Parametre & SeparadorParametre, vbTextCompare)
         If PosInicial > 2 Then
```

```
            If Mid(ContingutFitxer, PosInicial - 2, 2) <> vbCrLf Then PosInicial = 0
         End If
         If PosInicial > 0 Then 'si parametre trobat
            PosInicial = Len(Parametre) + 1 + PosInicial
            PosIniComentari = InStr(PosInicial, ContingutFitxer, "/*", vbTextCompare)
            PosFinal = InStr(PosInicial, ContingutFitxer, vbCrLf, vbTextCompare)
            If PosIniComentari <> 0 And PosIniComentari < PosFinal Then PosFinal =
PosIniComentari
            If PosFinal = 0 Then PosFinal = Len(ContingutFitxer) + 1
            GetParametre = Mid(ContingutFitxer, PosInicial, PosFinal - PosInicial)
            GetParametre = Replace(GetParametre, vbTab, "")
            GetParametre = Trim(GetParametre)
         Else
            GestioError 3
         End If
      End If
   End If
   Exit Function

ControlErrors:
   MsgBox Err.Description, vbCritical, "Error sistema"
End Function


Public Sub SetParametre(ByVal Parametre As String, ByVal Valor As String, Optional
ByVal CrearSiNoTrobat As Boolean = True, Optional NomFitxer As String = "")
On Error GoTo ControlErrors
Dim PosInicial, PosFinal, PosIniComentari As Variant
Dim car As String
Dim i As Byte

   UltimParametre = Parametre
   GestioError 0
   If NomFitxer <> "" Then RutaFitxer = NomFitxer
   If RutaFitxer = "" Then GestioError 1
   If NumError = 0 Then
      If ContingutFitxer = "" Then Me.CapturarContingutFitxer RutaFitxer
      If ContingutFitxer <> "" Then
         PosInicial = 0
         PosFinal = 0
         PosInicial = InStr(1, ContingutFitxer, Parametre & SeparadorParametre,
vbTextCompare)
         If PosInicial > 2 Then
            If Mid(ContingutFitxer, PosInicial - 2, 2) <> vbCrLf Then PosInicial = 0
         End If
         If PosInicial > 0 Then 'si parametre trobat
            PosInicial = Len(Parametre) + 1 + PosInicial
            PosIniComentari = InStr(PosInicial, ContingutFitxer, "/*", vbTextCompare)
            PosFinal = InStr(PosInicial, ContingutFitxer, vbCrLf, vbTextCompare)
            If PosFinal = 0 Then
```

```
            PosFinal = Len(ContingutFitxer) + 1
         Else
            If PosIniComentari < PosFinal And PosIniComentari > 0 Then
               i = 0
               Do
                  car = Mid(ContingutFitxer, PosInicial + i, 1)
                  i = i + 1
               Loop Until car = " " Or car = vbTab Or PosInicial + i - 1 >= PosIniComentari
               PosFinal = PosInicial + i - 1
            End If
         End If
         ContingutFitxer = Left(ContingutFitxer, PosInicial - 1) & Valor &
Right(ContingutFitxer, Len(ContingutFitxer) - PosFinal + 1)
         GuardarCadenaEnFitxer ContingutFitxer, RutaFitxer
      Else 'si no trobat
         If CrearSiNoTrobat Then
            If Right(ContingutFitxer, 2) <> vbCrLf Then ContingutFitxer = ContingutFitxer
& vbCrLf
            ContingutFitxer = ContingutFitxer & Parametre & SeparadorParametre & Valor
            GuardarCadenaEnFitxer ContingutFitxer, RutaFitxer
         Else
            GestioError 3
         End If
      End If
   End If
   End If
   Exit Sub

ControlErrors:
   MsgBox Err.Description, vbCritical, "Error sistema"
End Sub


Function CapturarContingutFitxer(NomFitxer As String, Optional MostrarMissagtgeError As
Boolean = True) As Variant
On Error GoTo ControlErrors
Dim Fitxer As Integer
Dim LongitudArxiu As Long

   CapturarContingutFitxer = ""
   GestioError 0
   If NomFitxer <> "" Then RutaFitxer = NomFitxer
   If RutaFitxer = "" Then GestioError 1
   If NumError = 0 Then
      Fitxer = FreeFile
      Open RutaFitxer For Binary As Fitxer   ' Abre el archivo.
      LongitudArxiu = LOF(Fitxer)   ' Obtiene la longitud del archivo.
      CapturarContingutFitxer = Input(LongitudArxiu, Fitxer)
      Close Fitxer   ' Cierra el archivo.
      ContingutFitxer = CapturarContingutFitxer
```

```
    Else
        If MostrarMissagtgeError Then MsgBox "No s'ha indicat la ruta del fitxer", vbCritical,
"Captura contingut fitxer"
    End If
    Exit Function

ControlErrors:
    If Err.Number = 53 Then
        GestioError 2
        MsgBox "No s'ha trobat el fitxer:" & vbCrLf & NomFitxer, vbCritical, "Error sistema"
    Else
        GestioError Err.Number
        If Err.Number = 62 Then
            Resume Next
        Else
            MsgBox Err.Description, vbCritical, "Error sistema"
        End If
    End If
End Function


Sub GuardarCadenaEnFitxer(ByVal Cadena As Variant, ByVal NomFitxer As String)
On Error GoTo ControlErrors
Dim FitxerImp As Integer

    GestioError 0
    If NomFitxer <> "" Then RutaFitxer = NomFitxer
    If RutaFitxer = "" Then GestioError 1
    If NumError = 0 Then
        FitxerImp = FreeFile
        Open NomFitxer For Output As #FitxerImp
        Print #FitxerImp, Cadena
        Close #FitxerImp
        RutaFitxer = NomFitxer
    End If

    Exit Sub
ControlErrors:
    MsgBox Err.Description, vbCritical, Err.Number
End Sub


Function ExisteixFitxer(ByVal NomFitxer As String) As Boolean
Dim fs

    Set fs = CreateObject("Scripting.FileSystemObject")
    ExisteixFitxer = fs.fileexists(NomFitxer)
End Function
```

```
Private Sub GestioError(ByVal NumeroError As Integer)

    Select Case NumeroError
        Case 0:
            NumError = NumeroError
            DesError = ""
        Case 1:
            NumError = NumeroError
            DesError = "No s'ha pasat cap ruta de fitxer a la variable al paràmetre RutaFitxer"
        Case 2:
            NumError = NumeroError
            DesError = "No s'ha pogut obrir el fitxer " & RutaFitxer
        Case 3:
            NumError = NumeroError
            DesError = "No s'ha trobat el parametre " & UltimParametre
        Case Else:
            If NumeroError > 10 Then
                NumError = Err.Number
                DesError = Err.Description
            Else
                NumError = NumeroError
                DesError = "Error no definit"
            End If
    End Select
    If NumError <> 0 And MissatgesErrorHabilitat Then MsgBox NumeroError & "." &
DesError, vbCritical, "Error fitxer"
End Sub
```

Mòdul FMain

```
'Dins aquest formulari s'hi troben totes les funcions que gestionen el funcionament de la
pantalla principal del programa
Option Explicit

Private Sub Boto_Click(Index As Integer)
Dim Voltes As String
Dim i As Byte

    If Boto(Index).Enabled Then
        Select Case Index
            Case 0:
                For i = 0 To TipusCrono.Count - 1
                    If TipusCrono(i).value = True Then
                        Tipus = i
                        Exit For
                    End If
                Next
                Pistes = 0
```

```
            For i = 0 To Pista.Count - 1
               If Pista(i).value = vbChecked Then Pistes = Pistes + i + 1
            Next
            If Pistes <> 0 And (Tipus <> 3) Or Tipus = 3 Then
               If Pistes = 3 Then Pistes = 0
               Select Case Tipus
                  Case 0:
                     EnviaOrdre Com, 1, 10 + Pistes
                     CanviarEstat Practica
                  Case 1:
                     EnviaOrdre Com, 1, 20 + Pistes
                     CanviarEstat PracticaCrono
                  Case 2:
                     Do
                        Voltes = InputBox("Entra el número de voltes ha donar", "Configuració
Cronometratge")
                     Loop Until IsNumeric(Voltes)
                     VoltesCursa = CInt(Voltes)
                     EnviaOrdre Com, 1, 30 + Pistes, CLng(Voltes)
                     CanviarEstat Crono
                  Case 3:
                     Do
                        Voltes = InputBox("Entra el número de voltes ha donar", "Configuració
Carrera")
                     Loop Until IsNumeric(Voltes)
                     VoltesCursa = CInt(Voltes)
                     EnviaOrdre Com, 1, 40, CLng(Voltes)
                     CanviarEstat Carrera
               End Select
               If Not EsperaTramaRebudaCorrecte Then
                  MsgBox "No s'ha pogut establir comunicació amb el cronometrador",
vbExclamation, "Error comunicacions"
                  CanviarEstat Comunicacionsparades
               End If
            Else
               MsgBox "Has de seleccionar almenys una pista", vbExclamation, "Configurant
cronometrador"
            End If
         Case 1:
            CanviarEstat PendentEscollirMode
      End Select
   End If
End Sub


Private Sub Command1_Click(Index As Integer)
On Error GoTo ControlErrors
Dim i As Byte
Dim Contingut As String
Dim FitxerVoltes As New Fitxer
```

```
    If Me.Llista(Index).ListCount > 0 Then
        Me.CommonDialog1.Filter = "txt|.txt"
        Me.CommonDialog1.CancelError = True
        Me.CommonDialog1.ShowSave

        Contingut = ""
        For i = 0 To Me.Llista(Index).ListCount - 1
            Contingut = Contingut & Me.Llista(Index).List(i) & vbCrLf
        Next
        FitxerVoltes.GuardarCadenaEnFitxer Contingut, Me.CommonDialog1.filename
    End If

    Exit Sub
ControlErrors:
End Sub


Private Sub Form_Load()
Dim str As String
Dim TmpByte As Byte
Dim ErrorEnFitxer As Boolean

    FitxerConfig.CapturarContingutFitxer App.Path & "\config.ini"
    str = FitxerConfig.GetParametre("TipusCrono")
    If Not IsNumeric(str) Then ErrorEnFitxer = True
    If Not ErrorEnFitxer Then Me.TipusCrono(CByte(str)).value = True
    str = FitxerConfig.GetParametre("Pista1")
    If Not IsNumeric(str) Then ErrorEnFitxer = True
    If Not ErrorEnFitxer Then Me.Pista(0).value = IIf(str = "1", vbChecked, vbUnchecked)
    str = FitxerConfig.GetParametre("Pista2")
    If Not IsNumeric(str) Then ErrorEnFitxer = True
    If Not ErrorEnFitxer Then Me.Pista(1).value = IIf(str = "1", vbChecked, vbUnchecked)
    str = FitxerConfig.GetParametre("Port")
    If Not IsNumeric(str) Then ErrorEnFitxer = True
    If ErrorEnFitxer Then
        MsgBox "Valor d'algun paràmetre del fitxer config incorrecte", vbCritical, "Fitxer
config"
        Unload Me
    End If

    DeteccioComS CByte(str)
    CanviarCom CByte(str)
    CanviarEstat Comunicacionsparades
End Sub


Private Sub Form_Unload(Cancel As Integer)

    If Com.PortOpen Then Com.PortOpen = False
```

```
End Sub


Private Sub Com_OnComm()
Dim i As Byte
Dim BufferTmp As String
Dim PosInici As Byte

   Select Case Com.CommEvent

      Case comEvReceive:
         BufferEntrada = BufferEntrada & Com.Input
         Com.InputLen = 0
         If Len(BufferEntrada) >= 14 Then 'comprovem si tenime tota una trama al buffer
            'recorrem buffer d'entrada en busca de la trama. Busquem inici de trama
            For i = 1 To Len(BufferEntrada)
               If Asc(Mid(BufferEntrada, i, 1)) = SOH Then 'si és inici de trama
                  PosInici = i
                  Exit For
               End If
            Next
            'borro la part de buffer que tingui per davant l'inici trama
            BufferEntrada = Right(BufferEntrada, (Len(BufferEntrada) - PosInici) + 1)
            If Len(BufferEntrada) >= 14 Then 'Comprovo hagi rebut la trama al complert
               TramaRebuda = Left(BufferEntrada, 14)
               If TramaRebudaCorrecte Then
                  TramaRebudaOk = True
                  Buffer(PunterBuffer) = TramaRebuda
                  PunterBuffer = PunterBuffer + 1
                  If PunterBuffer > 25 Then PunterBuffer = 0
               End If
               BufferEntrada = ""
            End If
         End If
   End Select

End Sub


Private Sub MCom_Click(Index As Integer)

   CanviarCom Index
End Sub

‘Funció per connectar amb el cronomotrador i canvia l’estat
Private Sub MConectar_Click()

   If Estat = Estats.Comunicacionsparades Then
      CanviarEstat Estats.PendentEscollirMode
   Else
```

```
        CanviarEstat Estats.Comunicacionsparades
    End If

End Sub


Private Sub MSortir_Click()

    Unload Me
End Sub


Private Sub Pista_Click(Index As Integer)

    FitxerConfig.SetParametre "Pista1", IIf(Pista(0).value = vbChecked, "1", "0")
    FitxerConfig.SetParametre "Pista2", IIf(Pista(1).value = vbChecked, "1", "0")
End Sub


Private Sub Timer1_Timer()
Dim Minuts As Integer
Dim Segons As Double
Dim p As Byte
Dim Pista As Byte
Dim Centesimes As Integer
Dim Volta As Integer
Dim str As String
Dim Bloc1Hex, Bloc2Hex As String
Dim Seg As Double

    Temps = Rell.SegonsTranscorreguts(7)
    Minuts = Temps \ 60
    Segons = Abs(Round(Temps - (Minuts * 60), 3))
    Me.Label1.Caption = IIf(Minuts <> 0, Minuts & """, "") & Replace(Format(Segons, "0.00"),
",", """""")

    If Len(Buffer(PunterLectura)) <> 0 Then
        Bloc1Hex = Mid(Buffer(PunterLectura), 4, 4)
        Bloc2Hex = Mid(Buffer(PunterLectura), 8, 4)
        BlocDades1 = NumeroDecimal(Bloc1Hex)
        BlocDades2 = NumeroDecimal(Bloc2Hex)
        OrdreRebut = Asc(Mid(Buffer(PunterLectura), 3, 1)) - Asc("0")
        Buffer(PunterLectura) = ""
        PunterLectura = PunterLectura + 1
        If PunterLectura > 25 Then PunterLectura = 0
        If OrdreRebut = 70 Then
            Pista = CByte(Mid(Bloc2Hex, 2, 1))
            Seg = BlocDades1
            Centesimes = NumeroDecimal(Mid(Bloc2Hex, 3, 2))
            Seg = Seg + (Centesimes / 100)
```

```
        Select Case Pista
            Case 1:
                Segons = Abs(Seg - SegonsAnt1)
            Case 2:
                Segons = Abs(Seg - SegonsAnt2)
        End Select
        Select Case Pista
            Case 1:
                If Volta1 < VoltesCursa Or Tipus <> 0 Or Tipus <> 1 Then
                    Volta1 = Volta1 + 1
                    Volta = Volta1
                    If Segons < TempsVoltaRapida1 Then
                        TempsVoltaRapida1 = Segons
                        VoltaRapida1 = Volta1
                        Me.Label3.Caption = SegToString(0, Segons, 0) & "(" & VoltaRapida1 &
")"
                    End If
                    Me.Llista(0).AddItem Volta1 & " - " & SegToString(0, Segons, 0) & " --- " &
SegToString(0, Seg, 0)
                    Me.Label7.Caption = SegToString(0, Segons, 0)
                    SegonsAnt1 = Seg
                End If
            Case 2:
                If Volta2 < VoltesCursa Or Tipus <> 0 Or Tipus <> 1 Then
                    Volta2 = Volta2 + 1
                    Volta = Volta2
                    If Segons < TempsVoltaRapida1 Then
                        TempsVoltaRapida2 = Segons
                        VoltaRapida2 = Volta2
                        Me.Label4.Caption = SegToString(0, Segons, 0) & "(" & VoltaRapida2 &
")"
                    End If
                    Me.Llista(1).AddItem Volta2 & " - " & SegToString(0, Segons, 0) & " --- " &
SegToString(0, Seg, 0)
                    Me.Label8.Caption = SegToString(0, Segons, 0)
                    SegonsAnt2 = Seg
                End If
        End Select
        Select Case Tipus
            Case 0, 1:
                FMain.Label2(Pista - 1).Caption = "Volta " & Volta
            Case 2:
                FMain.Label2(Pista - 1).Caption = "Volta " & Volta & "/" & VoltesCursa
                If Volta >= VoltesCursa Then CanviarEstat PendentEscollirMode
            Case 3:
                FMain.Label2(Pista - 1).Caption = "Volta " & Volta & "/" & VoltesCursa
                If PendentFinalitzarPista Then
                    CanviarEstat PendentEscollirMode
                End If
                If Volta1 >= VoltesCursa Or Volta2 >= VoltesCursa Then
```

```
                EnviaOrdre Me.Com, 1, 80, Pista, 0
                PendentFinalitzarPista = True
            End If

        End Select
    End If
End If

End Sub


Private Sub TipusCrono_Click(Index As Integer)

    FitxerConfig.SetParametre "TipusCrono", Index
End Sub
```

Mòdul Fvoltes

```
'Aquest formulari és la pantalla que ens permet escollir les voltes a realitzar en els
modes carrera i cronometratge
Private Sub Command1_Click()

    If Not IsNumeric(Me.Text1.Text) Then
        MsgBox "Número de voltes entrades incorrecte", vbExclamation, "Configuració
competició"
        Me.Text1.Text = VoltesCursa
    Else
        If Me.Text1.Text <= 0 Then
            MsgBox "Número de voltes entrades incorrecte", vbExclamation, "Configuració
competició"
            Me.Text1.Text = VoltesCursa
        Else
            VoltesCursa = Me.Text1.Text
            FormulariCancelat = False
            Unload Me
        End If
    End If

End Sub

Private Sub Command2_Click()

    FormulariCancelat = True
    Unload Me
End Sub

Private Sub Form_Load()
    FormulariCancelat = True
    Me.Text1.Text = VoltesCursa
End Sub
```

ANNEX 4: TAULA DE CARÀCTERS ASCII

Decimal	Octal	Hex	Binari	Valor		Decimal	Octal	Hex	Binari	Valor
0	0	0	0	NUL		64	100	40	1000000	@
1	1	1	1	SOH		65	101	41	1000001	A
2	2	2	10	STX		66	102	42	1000010	B
3	3	3	11	ETX		67	103	43	1000011	C
4	4	4	100	EOT		68	104	44	1000100	D
5	5	5	101	ENQ		69	105	45	1000101	E
6	6	6	110	ACK		70	106	46	1000110	F
7	7	7	111	BEL		71	107	47	1000111	G
8	10	8	1000	BS		72	110	48	1001000	H
9	11	9	1001	HT		73	111	49	1001001	I
10	12	00A	1010	LF		74	112	04A	1001010	J
11	13	00B	1011	VT		75	113	04B	1001011	K
12	14	00C	1100	FF		76	114	04C	1001100	L
13	15	00D	1101	CR		77	115	04D	1001101	M
14	16	00E	1110	SO		78	116	04E	1001110	N
15	17	00F	1111	SI		79	117	04F	1001111	O
16	20	10	10000	DLE		80	120	50	1010000	P
17	21	11	10001	DC1		81	121	51	1010001	Q
18	22	12	10010	DC2		82	122	52	1010010	R
19	23	13	10011	DC3		83	123	53	1010011	S
20	24	14	10100	DC4		84	124	54	1010100	T
21	25	15	10101	NAK		85	125	55	1010101	U
22	26	16	10110	SYN		86	126	56	1010110	V
23	27	17	10111	ETB		87	127	57	1010111	W
24	30	18	11000	CAN		88	130	58	1011000	X
25	31	19	11001	EM		89	131	59	1011001	Y
26	32	01A	11010	SUB		90	132	05A	1011010	Z
27	33	01B	11011	ESC		91	133	05B	1011011	[
28	34	01C	11100	FS		92	134	05C	1011100	\
29	35	01D	11101	GS		93	135	05D	1011101	]
30	36	01E	11110	RS		94	136	05E	1011110	^
31	37	01F	11111	US		95	137	05F	1011111	_
32	40	20	100000	SP		96	140	60	1100000	`
33	41	21	100001	!		97	141	61	1100001	a
34	42	22	100010	"		98	142	62	1100010	b
35	43	23	100011	#		99	143	63	1100011	c
36	44	24	100100	$		100	144	64	1100100	d
37	45	25	100101	%		101	145	65	1100101	e
38	46	26	100110	&		102	146	66	1100110	f
39	47	27	100111	'		103	147	67	1100111	g
40	50	28	101000	(		104	150	68	1101000	h

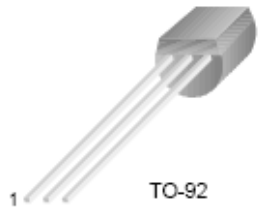
Decimal	Octal	Hex	Binari	Valor		Decimal	Octal	Hex	Binari	Valor
41	51	29	101001	)		105	151	69	1101001	i
42	52	02A	101010	*		106	152	06A	1101010	j
43	53	02B	101011	+		107	153	06B	1101011	k
44	54	02C	101100	,		108	154	06C	1101100	l
45	55	02D	101101	-		109	155	06D	1101101	m
46	56	02E	101110	.		110	156	06E	1101110	n
47	57	02F	101111	/		111	157	06F	1101111	o
48	60	30	110000	0		112	160	70	1110000	p
49	61	31	110001	1		113	161	71	1110001	q
50	62	32	110010	2		114	162	72	1110010	r
51	63	33	110011	3		115	163	73	1110011	s
52	64	34	110100	4		116	164	74	1110100	t
53	65	35	110101	5		117	165	75	1110101	u
54	66	36	110110	6		118	166	76	1110110	v
55	67	37	110111	7		119	167	77	1110111	w
56	70	38	111000	8		120	170	78	1111000	x
57	71	39	111001	9		121	171	79	1111001	y
58	72	03A	111010	:		122	172	07A	1111010	z
59	73	03B	111011	;		123	173	07B	1111011	{
60	74	03C	111100	<		124	174	07C	1111100	|
61	75	03D	111101	=		125	175	07D	1111101	}
62	76	03E	111110	>		126	176	07E	1111110	~
63	77	03F	111111	?		127	177	07F	1111111	DEL

ANNEX 5: *DATA SHEET* DEL TRANSISTOR BC547B

BC546/547/548/549/550

Switching and Applications

- High Voltage: BC546, V_{CEO}=65V
- Low Noise: BC549, BC550
- Complement to BC556 ... BC560



TO-92

1. Collector 2. Base 3. Emitter

NPN Epitaxial Silicon Transistor

Absolute Maximum Ratings T_a=25°C unless otherwise noted

Symbol	Parameter	Value	Units
V_{CBO}	Collector-Base Voltage : BC546	80	V
	: BC547/550	50	V
	: BC548/549	30	V
V_{CEO}	Collector-Emitter Voltage : BC546	65	V
	: BC547/550	45	V
	: BC548/549	30	V
V_{EBO}	Emitter-Base Voltage : BC546/547	6	V
	: BC548/549/550	5	V
I_C	Collector Current (DC)	100	mA
P_C	Collector Power Dissipation	500	mW
T_J	Junction Temperature	150	°C
T_{STG}	Storage Temperature	-65 ~ 150	°C

Electrical Characteristics T_a=25°C unless otherwise noted

Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Units
I_{CBO}	Collector Cut-off Current	V_{CB}=30V, I_E=0			15	nA
h_{FE}	DC Current Gain	V_{CE}=5V, I_C=2mA	110		800	
V_{CE} (sat)	Collector-Emitter Saturation Voltage	I_C=10mA, I_B=0.5mA		90	250	mV
		I_C=100mA, I_B=5mA		200	600	mV
V_{BE} (sat)	Base-Emitter Saturation Voltage	I_C=10mA, I_B=0.5mA		700		mV
		I_C=100mA, I_B=5mA		900		mV
V_{BE} (on)	Base-Emitter On Voltage	V_{CE}=5V, I_C=2mA	580	660	700	mV
		V_{CE}=5V, I_C=10mA			720	mV
f_T	Current Gain Bandwidth Product	V_{CE}=5V, I_C=10mA, f=100MHz		300		MHz
C_{ob}	Output Capacitance	V_{CB}=10V, I_E=0, f=1MHz		3.5	6	pF
C_{ib}	Input Capacitance	V_{EB}=0.5V, I_C=0, f=1MHz		9		pF
NF	Noise Figure : BC546/547/548	V_{CE}=5V, I_C=200µA		2	10	dB
	: BC549/550	f=1KHz, R_G=2KΩ		1.2	4	dB
	: BC549	V_{CE}=5V, I_C=200µA		1.4	4	dB
	: BC550	R_G=2KΩ, f=30~15000MHz		1.4	3	dB

h_{FE} Classification

Classification	A	B	C
h_{FE}	110 ~ 220	200 ~ 450	420 ~ 800

Typical Characteristics

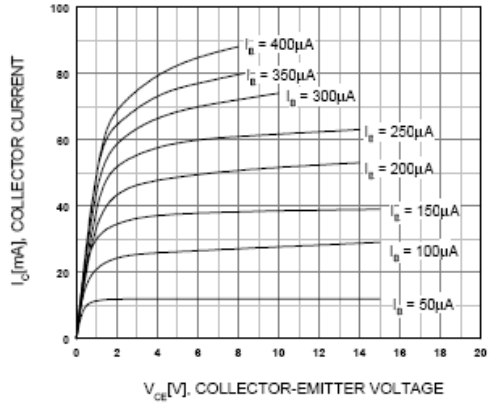


Figure 1. Static Characteristic

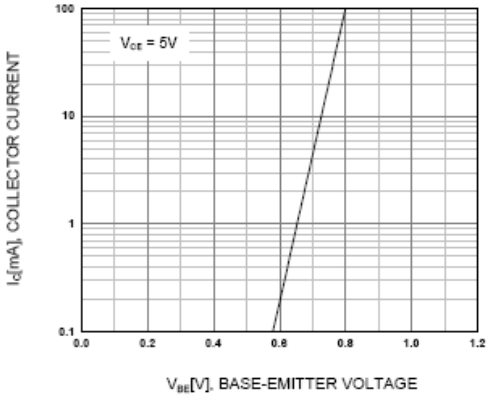


Figure 2. Transfer Characteristic

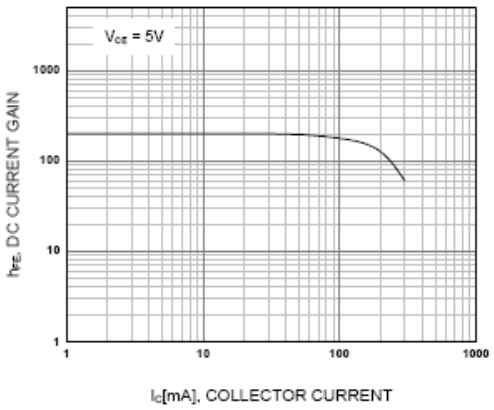


Figure 3. DC current Gain

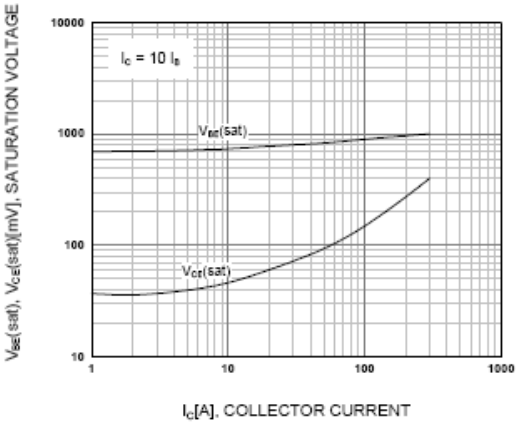


Figure 4. Base-Emitter Saturation Voltage
Collector-Emitter Saturation Voltage

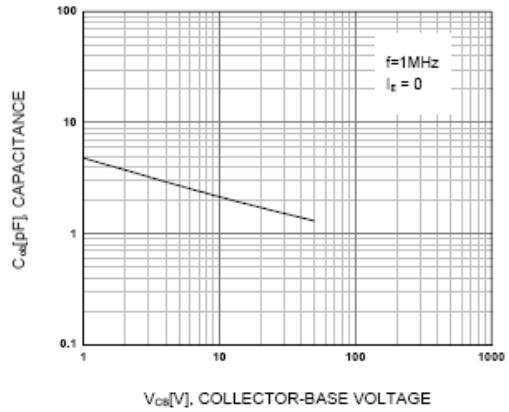


Figure 5. Output Capacitance

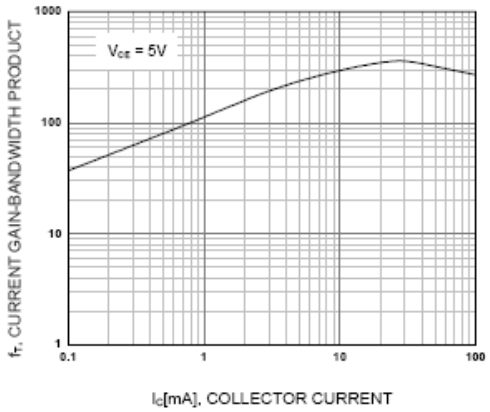


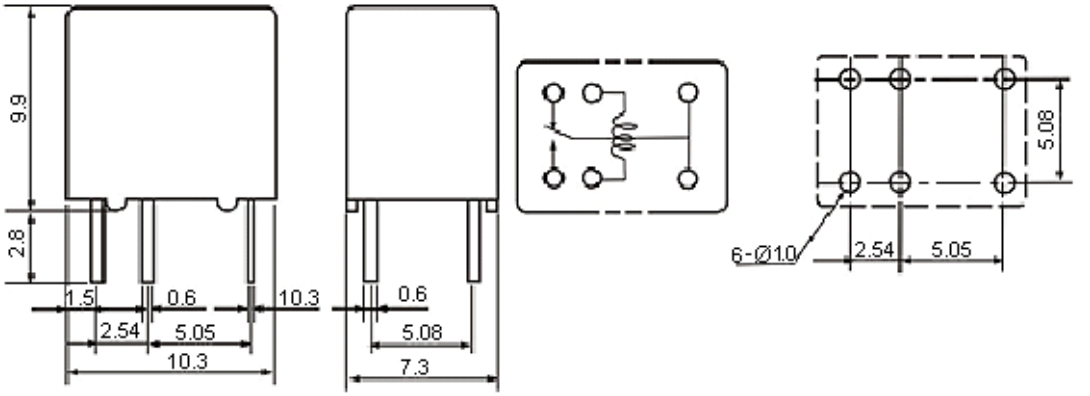
Figure 6. Current Gain Bandwidth Product

ANNEX 6: *DATA SHEET* DEL RELÉ

Features:

- SPCO series of relays.
- Small size and light weight.
- Plastic sealed (washable).
- DIL pitch terminal.

Dimensions



Dimensions : Millimetres

Specifications:

Contact Data

Contact arrangement	: SPCO (1 Form C).	
Contact material	: AuAg.	
Contact rating	: 1.0A at 24V dc/120V ac.	
Maximum switching voltage	: 30V dc/120V ac.	
Maximum switching current	: 2.0A.	
Maximum switching power	: 120VA/30W.	
Contact resistance	: 50mΩ.	
Life expectancy	: Electrical	: 100,000 operations.
	Mechanical	: 10,000,000 operations.

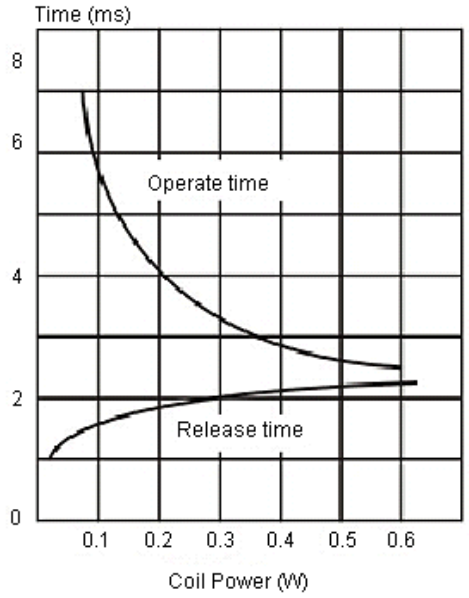
Specifications:

General Data

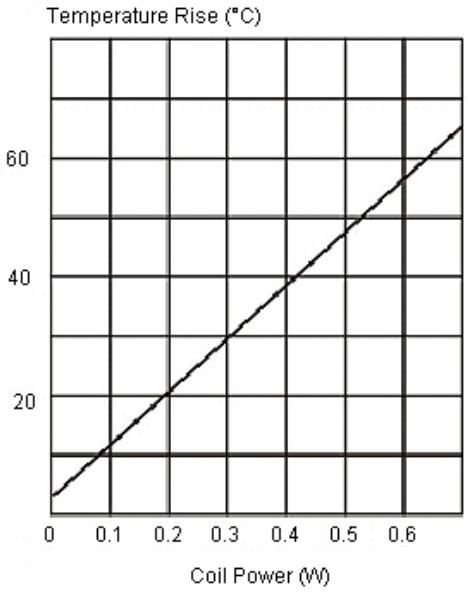
Insulation resistance	: 100MΩ.	
Dielectric strength	: 500V ac 1 minute between open contacts.	
	1000V ac 1 minute, contacts to coil.	
Operate time	: 6ms maximum.	
Release time	: 4ms maximum.	
Temperature range	: -25 to +70°C.	
Shock resistance	: Endurance	: 1000m/s^2
	Misoperation	: 100m/s^2.
Vibration resistance	: Endurance	: 10 to 55Hz, 1.5mm Double Amplitude.
	Misoperation	: 10 to 55Hz, 1.5mm Double Amplitude.
Heating	: 80 ±2°C 96 hours.	
Cold	: -40 ±2°C 96 hours.	
Humidity	: 85%RH.	

HRA(H) Characteristic Data:

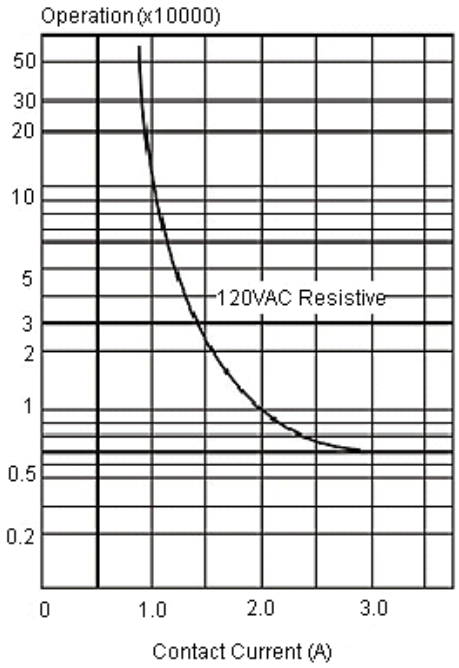
Timing



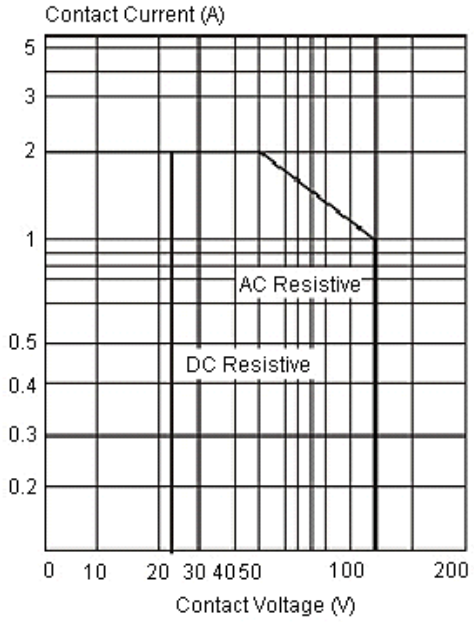
Coil Temperature Rise



Life Curve



Maximum Switching Power



Coil Data Chart **At 20°C**

Coil Voltage (V dc)	Coil Resistance (Ω)	Coil Operate Voltage (V dc)	Coil Release Voltage (V dc)	Coil Power Consumption (mW)	Contact Arrangement	Part Number
12	320	8.4	0.60	450	SPCO - 1 Form C	HRA-S-DC12V-C
24	1280	16.8	1.20			HRA-S-DC24V-C
5	75	3.5	0.25	330	SPCO - 1 Form C High Sensitivity	HRAH-S-DC5V-C
12	440	8.4	0.60			HRAH-S-DC12V-C
24	1560	16.8	1.20	380		HRAH-S-DC24V-C

ANNEX 7: *DATA SHEET* DE L'OPERACIONAL LM324N

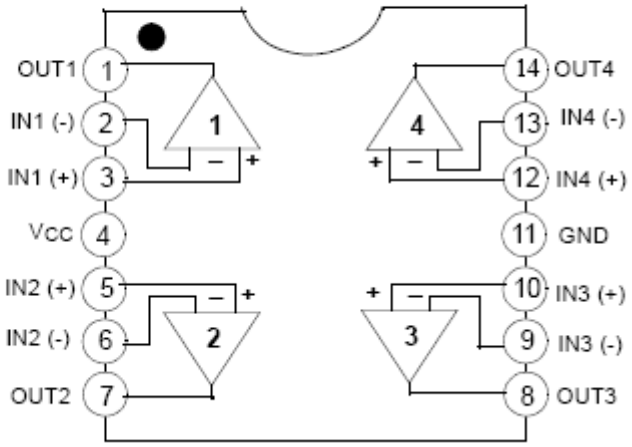
Features

- Internally Frequency Compensated for Unity Gain
- Large DC Voltage Gain: 100dB
- Wide Power Supply Range:
  LM224/LM224A, LM324/LM324A : 3V~32V (or ±1.5 ~ 16V)
  LM2902: 3V~26V (or ±1.5V ~ 13V)
- Input Common Mode Voltage Range Includes Ground
- Large Output Voltage Swing: 0V to V_{CC} -1.5V
- Power Drain Suitable for Battery Operation

Description

The LM324/LM324A,LM2902,LM224/LM224A consist of four independent, high gain, internally frequency compensated operational amplifiers which were designed specifically to operate from a single power supply over a wide voltage range. operation from split power supplies is also possible so long as the difference between the two supplies is 3 volts to 32 volts. Application areas include transducer amplifier, DC gain blocks and all the conventional OP Amp circuits which now can be easily implemented in single power supply systems.

Internal Block Diagram



Absolute Maximum Ratings

Parameter	Symbol	LM224/LM224A	LM324/LM324A	LM2902	Unit
Power Supply Voltage	V_{CC}	±16 or 32	±16 or 32	±13 or 26	V
Differential Input Voltage	$V_{I(DIFF)}$	32	32	26	V
Input Voltage	V_I	-0.3 to +32	-0.3 to +32	-0.3 to +26	V
Output Short Circuit to GND Vcc≤15V, T_A=25°C(one Amp)	-	Continuous	Continuous	Continuous	-
Power Dissipation, T_A=25°C 14-DIP 14-SOP	P_D	 1310 640	 1310 640	 1310 640	mW
Operating Temperature Range	T_{OPR}	-25 ~ +85	0 ~ +70	-40 ~ +85	°C
Storage Temperature Range	T_{STG}	-65 ~ +150	-65 ~ +150	-65 ~ +150	°C

Thermal Data

Parameter	Symbol	Value	Unit
Thermal Resistance Junction-Ambient Max. 14-DIP 14-SOP	Rθja	 95 195	°C/W

Electrical Characteristics

(V_{CC} = 5.0V, V_{EE} = GND, T_A = 25°C, unless otherwise specified)

Parameter	Symbol	Conditions	LM224			LM324			LM2902			Unit
			Min.	Typ.	Max.	Min.	Typ.	Max.	Min.	Typ.	Max.	
Input Offset Voltage	V_{IO}	V_{CM} = 0V to V_{CC} -1.5V $V_{O(P)}$ = 1.4V, R_S = 0Ω (Note1)	-	1.5	5.0	-	1.5	7.0	-	1.5	7.0	mV
Input Offset Current	I_{IO}	V_{CM} = 0V	-	2.0	30	-	3.0	50	-	3.0	50	nA
Input Bias Current	I_{BIAS}	V_{CM} = 0V	-	40	150	-	40	250	-	40	250	nA
Input Common-Mode Voltage Range	$V_{I(R)}$	Note1	0	-	V_{CC} -1.5	0	V_{CC} -1.5	-	0	-	V_{CC} -1.5	V
Supply Current	I_{CC}	R_L = ∞,V_{CC} = 30V (LM2902,V_{CC}=26V)	-	1.0	3	-	1.0	3	-	1.0	3	mA
		R_L = ∞,V_{CC} = 5V	-	0.7	1.2	-	0.7	1.2	-	0.7	1.2	mA
Large Signal Voltage Gain	G_V	V_{CC} = 15V,R_L=2kΩ $V_{O(P)}$ = 1V to 11V	50	100	-	25	100	-	25	100	-	V/mV
Output Voltage Swing	$V_{O(H)}$	Note1, R_L = 2kΩ	26	-	-	26	-	-	22	-	-	V
		Note1, R_L=10kΩ	27	28	-	27	28	-	23	24	-	V
	$V_{O(L)}$	V_{CC} = 5V,R_L=10kΩ	-	5	20	-	5	20	-	5	100	mV
Common-Mode Rejection Ratio	CMRR	-	70	85	-	65	75	-	50	75	-	dB
Power Supply Rejection Ratio	PSRR	-	65	100	-	65	100	-	50	100	-	dB
Channel Separation	CS	f = 1kHz to 20kHz (Note2)	-	120	-	-	120	-	-	120	-	dB
Short Circuit to GND	I_{SC}	V_{CC} = 15V	-	40	60	-	40	60	-	40	60	mA
Output Current	I_{SOURCE}	$V_{I(+)}$ = 1V, $V_{I(-)}$ = 0V V_{CC} = 15V $V_{O(P)}$ = 2V	20	40	-	20	40	-	20	40	-	mA
	I_{SINK}	$V_{I(+)}$ = 0V, $V_{I(-)}$ = 1V V_{CC} = 15V $V_{O(P)}$ = 2V	10	13	-	10	13	-	10	13	-	mA
		$V_{I(+)}$ = 0V, $V_{I(-)}$ = 1V V_{CC} = 5V,$V_{O(R)}$ = 200mV	12	45	-	12	45	-	-	-	-	µA
Differential Input Voltage	$V_{I(DIFF)}$	-	-	-	V_{CC}	-	-	V_{CC}	-	-	V_{CC}	V

Note :

1. V_{CC}=30V for LM224 and LM324 , V_{CC} = 26V for LM2902
2. This parameter, although guaranteed, is not 100% tested in production.