

Práctica de Sistemas Electrónicos Digitales de la Ingeniería Técnica de Telecomunicación, especialidad Telemática. Se trata de enviar una serie de 1 ó 0 de una placa a otra usando como medio de transmisión cables conectados a los distintos pines de las placas de pruebas con microcontroladores PIC 16F84A.

Se incluyen los códigos fuente de 4 versiones diferentes:

- Práctica completa con funciones básicas.
- Práctica completa con conexión a 4 hilos + Detector de transmisor y receptor: esta práctica tiene la práctica completa con funciones básicas en la que se usan 2 emisores y 2 receptores, los cuales usan un emisor y un receptor para enviar los datos y bits de Start y Stop y otros emisores y receptor para comprobar si hay un emisor o un receptor al otro lado del cable.
- Práctica completa con conexión a 2 hilos + Detector de transmisor y receptor: es igual que la anterior, excepto que usa sólo un cable de emisión y otro de recepción, realizando la comprobación de si existe otro emisor o receptor al otro lado del cable y la posterior emisión o recepción de los datos y bits de Start y Stop.
- Práctica completa con conexión a 2 hilos + Detector de transmisor y receptor + Efecto de luces en los LEDs: es exactamente igual a la anterior con el añadido de un efecto de luces en los LEDs al terminar el programa.

Se podrá ver un vídeo de funcionamiento grabado por Francisco Javier Merchán Macías en el que se puede ver el funcionamiento de las dos placas.

Javier Merchán

**[javier\[arroba\]javiermerchan.com](mailto:javier[arroba]javiermerchan.com)**

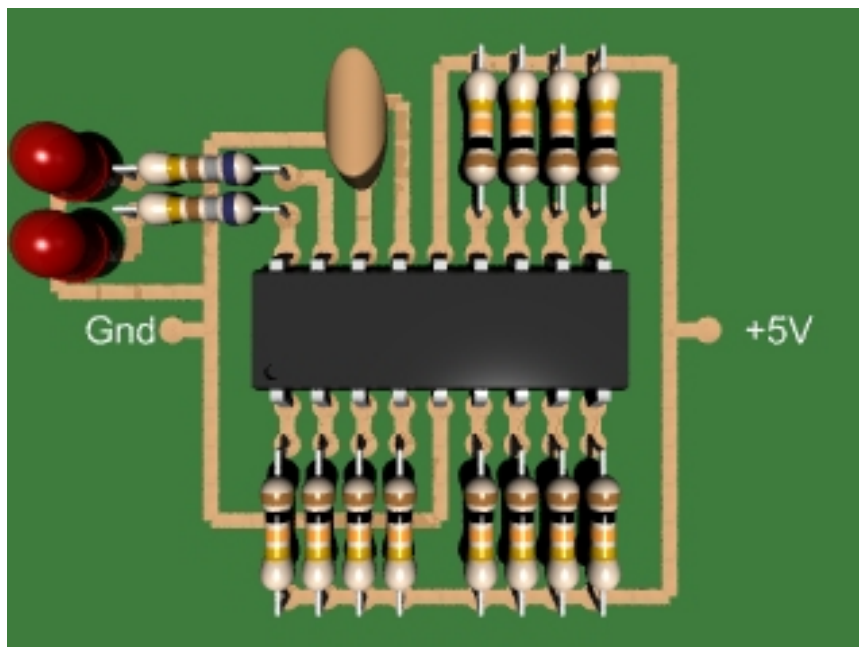
---

Copyright 2008 Francisco Javier Merchán Macías, Fátima Romero Romero y Esmeralda Rubio Parra.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.



Memoria de la práctica de SED

## **Comunicación serie entre dos PIC16F84A**

**Francisco Javier Merchán Macías**  
**Esmeralda Rubio Parra**  
**Fátima Romero Romero**

Ingeniería Técnica en Telecomunicación. Esp. Telemática  
Asignatura: Sistemas Electrónicos Digitales  
Profesor: Juan Ángel García Martínez

# Índice

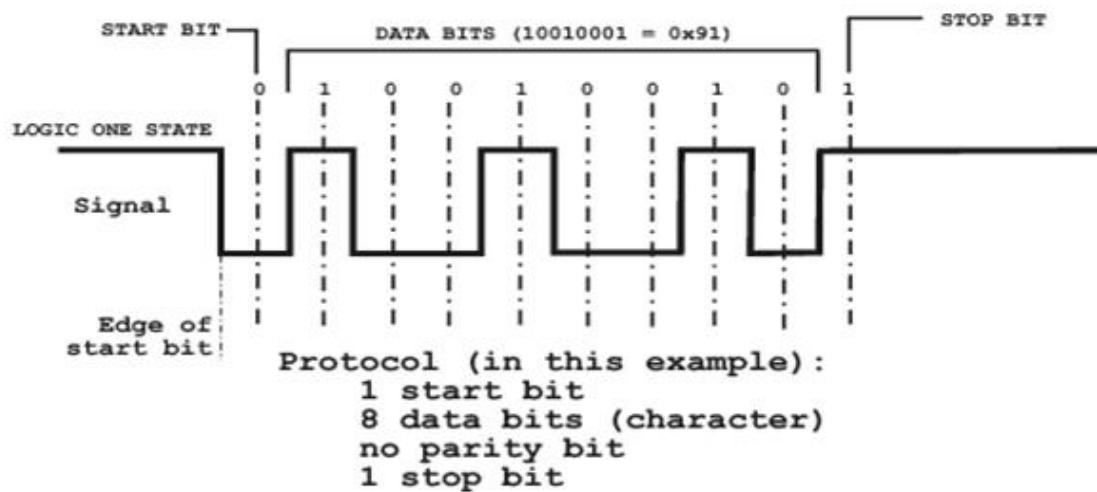
Enunciado de la práctica. Comunicación serie entre dos PIC16F84A .....	5
Práctica básica: .....	11
Código Fuente del programa básico: .....	14
Plantilla.asm .....	14
Elegir.inc .....	15
Cargando.inc.....	15
Emitir.inc .....	16
Recibir.inc .....	17
Explicación del funcionamiento del programa básico:.....	20
Capturas del Proteus, de un funcionamiento básico: .....	21
Detector emisor y receptor, detectando el emisor y el receptor: .....	21
Esquema Proteus con un emisor y un receptor (emitiendo): .....	22
Comentarios sobre la realización del programa básico : .....	23
Problema con las puertas que tuvimos y por el cual nos dejó las placas.....	23
Mejoras: .....	23
1. Detección de los roles (receptor-receptor, emisor-emisor y emisor-receptor y viceversa) de las placas. ....	23
Explicación del funcionamiento y el porqué de la mejora .....	23
Funcionamiento: .....	23
Capturas de pantalla de proteus:.....	24
Esquema Proteus con dos emisores (dando error):.....	24
Esquema Proteus con dos receptores (dando error): .....	24
Detector emisor y receptor, detectando el emisor y el receptor: .....	25
Código fuente relativo a ello. ....	25
2. Detección del bit de Start en dos ocasiones, para fiabilizar el funcionamiento de la práctica. ....	26
Motivación y funcionamiento de la mejora:.....	26
3. Emulación de las luces del “coche fantástico” con el banco de leds y el display de 7 segmentos , indicando la finalización de la tarea. ....	27
Motivo de la mejora: .....	27
Código fuente del mismo.....	27
4. Mejora sustancial de la práctica, anteriormente diseñada a 2 hilos, utilizando 4 hilos, ya que se detectó que en algunas ocasiones que podría detectarse un fallo debido al funcionamiento de las puertas usadas en la versión a 2 hilos.....	29
¿Por qué necesitamos esta mejora? .....	29
Cambios de la práctica de 2 hilos a 4 hilos: .....	30
Diagrama de la práctica con mejoras.....	31
Bibliografía.....	32

## Enunciado de la práctica. Comunicación serie entre dos PIC16F84A

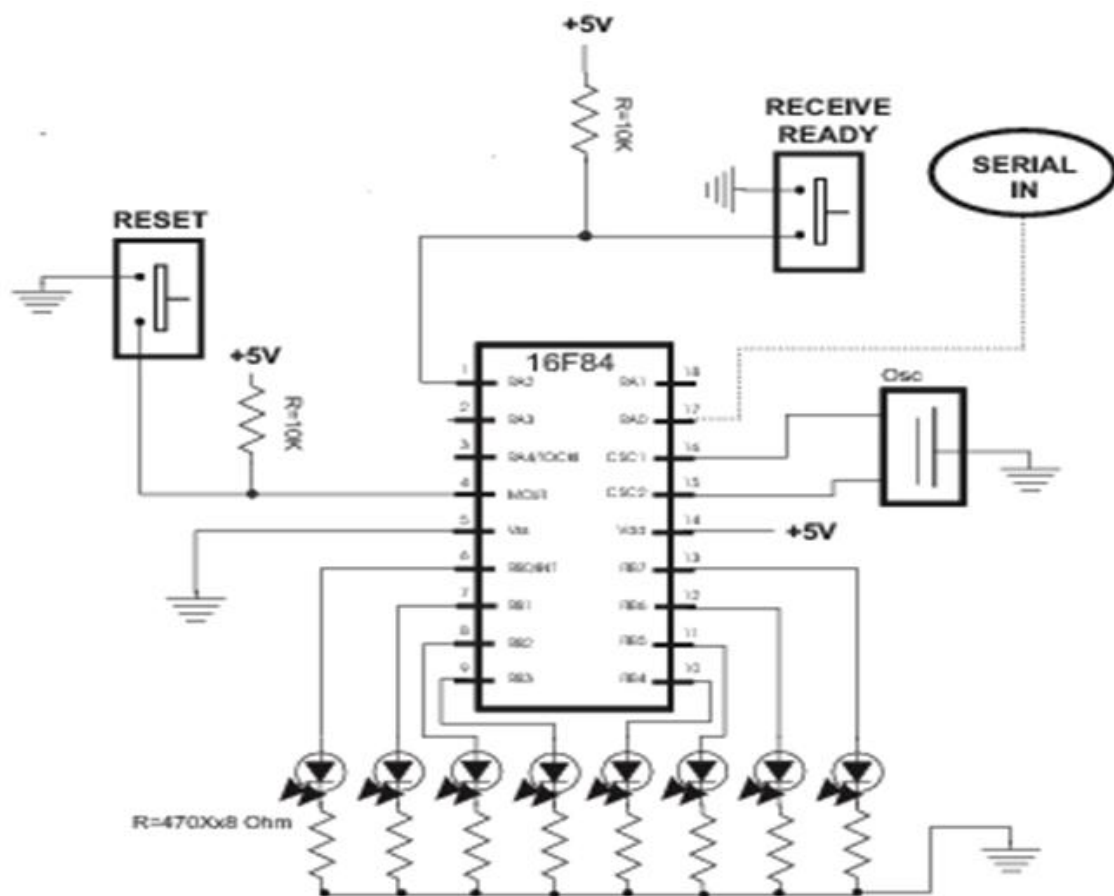
Realizar el diseño de un SED formado por dos PIC interconectados, en el que un PIC interroga el estado de un banco de interruptores, informa de su papel (emisor / receptor) y envía el resultado a un segundo PIC, que informa también de su papel (receptor / emisor), y que controla un banco de LED's que se iluminan de acuerdo al estado de los interruptores del PIC inicial.

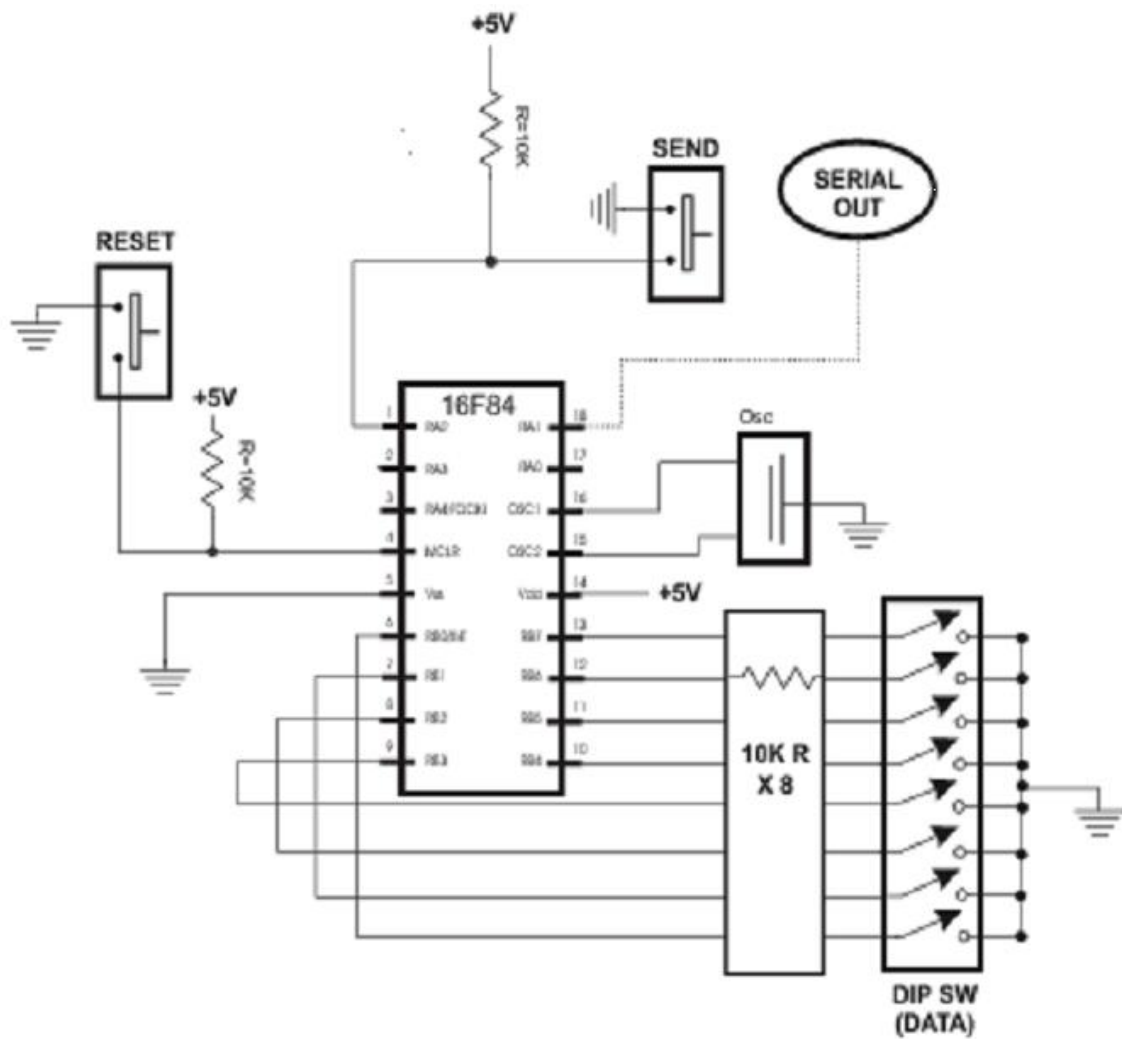
Por tanto, la comunicación serie que se establece implica que un PIC actúe como emisor o maestro, y el otro como receptor o esclavo, siendo posible el intercambio de papeles.

La estructura de los datos a transmitir vía serie es la siguiente:

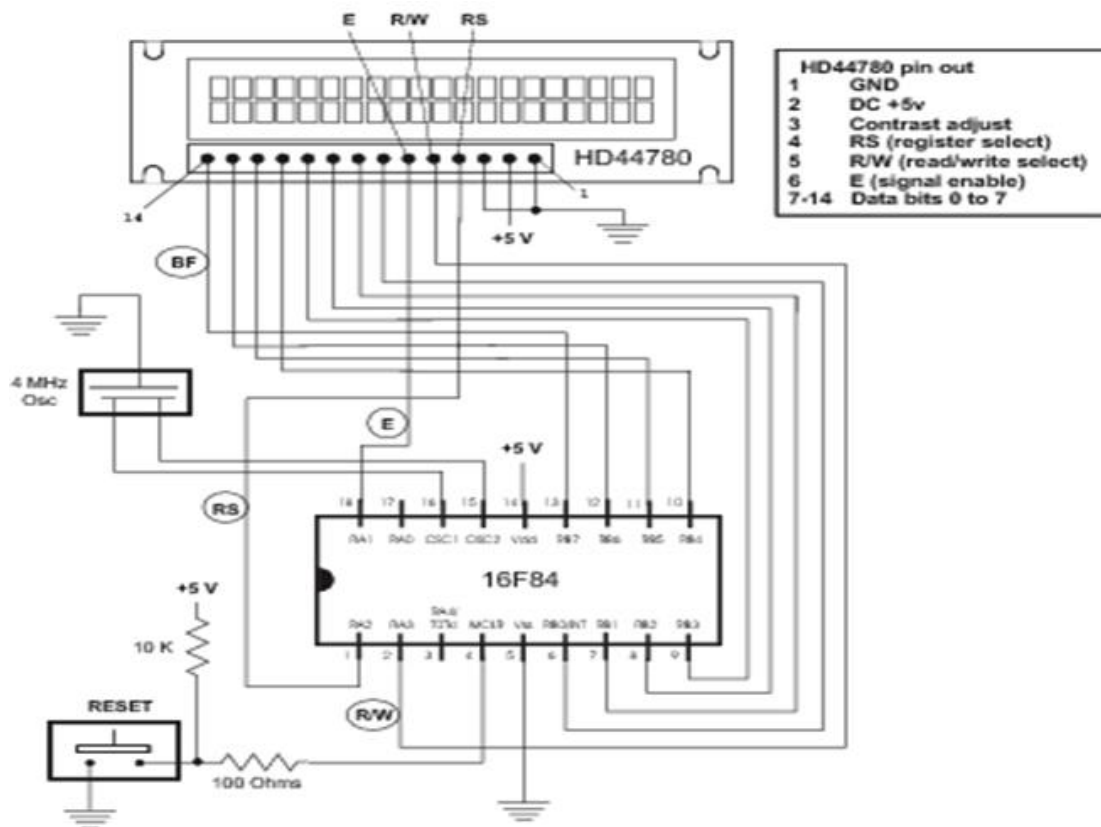


Circuitos para la comunicación serie entre los dos PIC, uno de ellos actuando como receptor, y el otro como emisor:





Interface entre el PIC y el módulo LCD, para representar el estado (emisor o receptor) de cada uno de los PIC, de forma independiente



La realización de este proyecto de SED debe seguir las líneas generales, de acuerdo al diagrama siguiente:

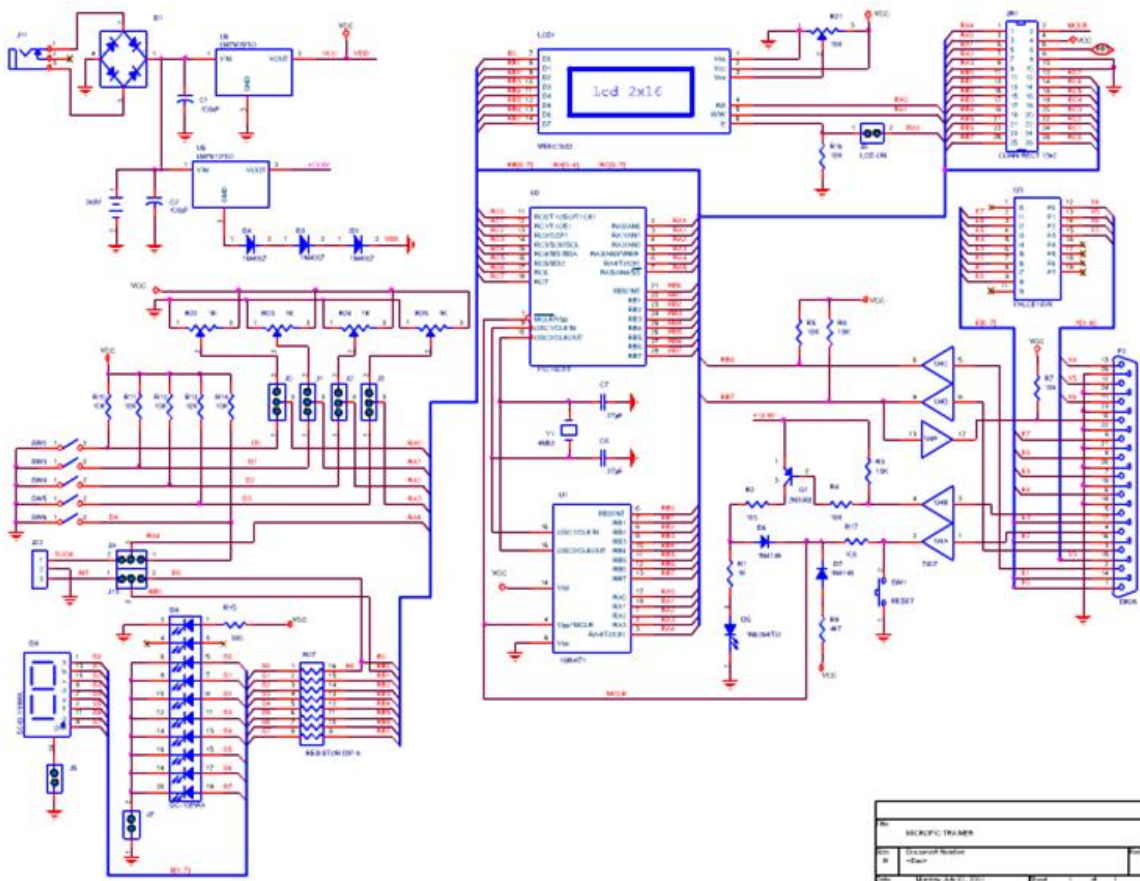




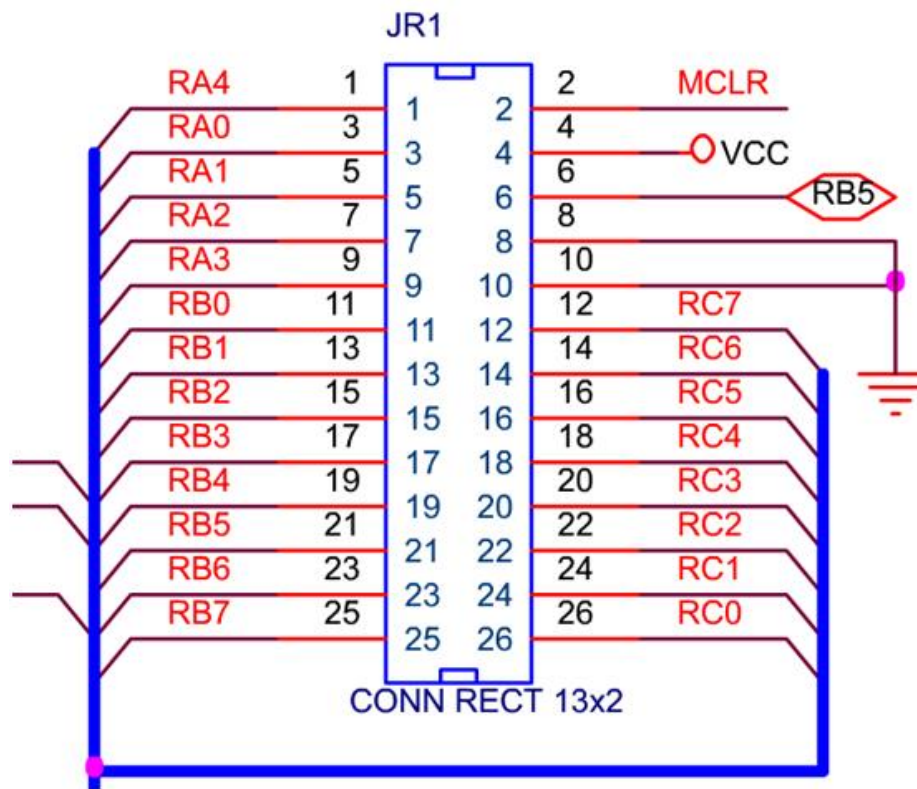
La realización del SED forma la parte de Laboratorio de la asignatura, que será evaluada de forma conjunta entre los miembros del grupo que la realice, y que según el Plan Docente de la asignatura, contribuye en un 30% de la nota global de la misma.

NOTA: La fecha de entrega del proyecto de SED será el día del examen de la parte teórica de la asignatura, que coincidirá con la fecha que la Subdirección Académica del CUM fije para el examen de la asignatura de SED.

Esquema eléctrico de la microPIC Trainer.

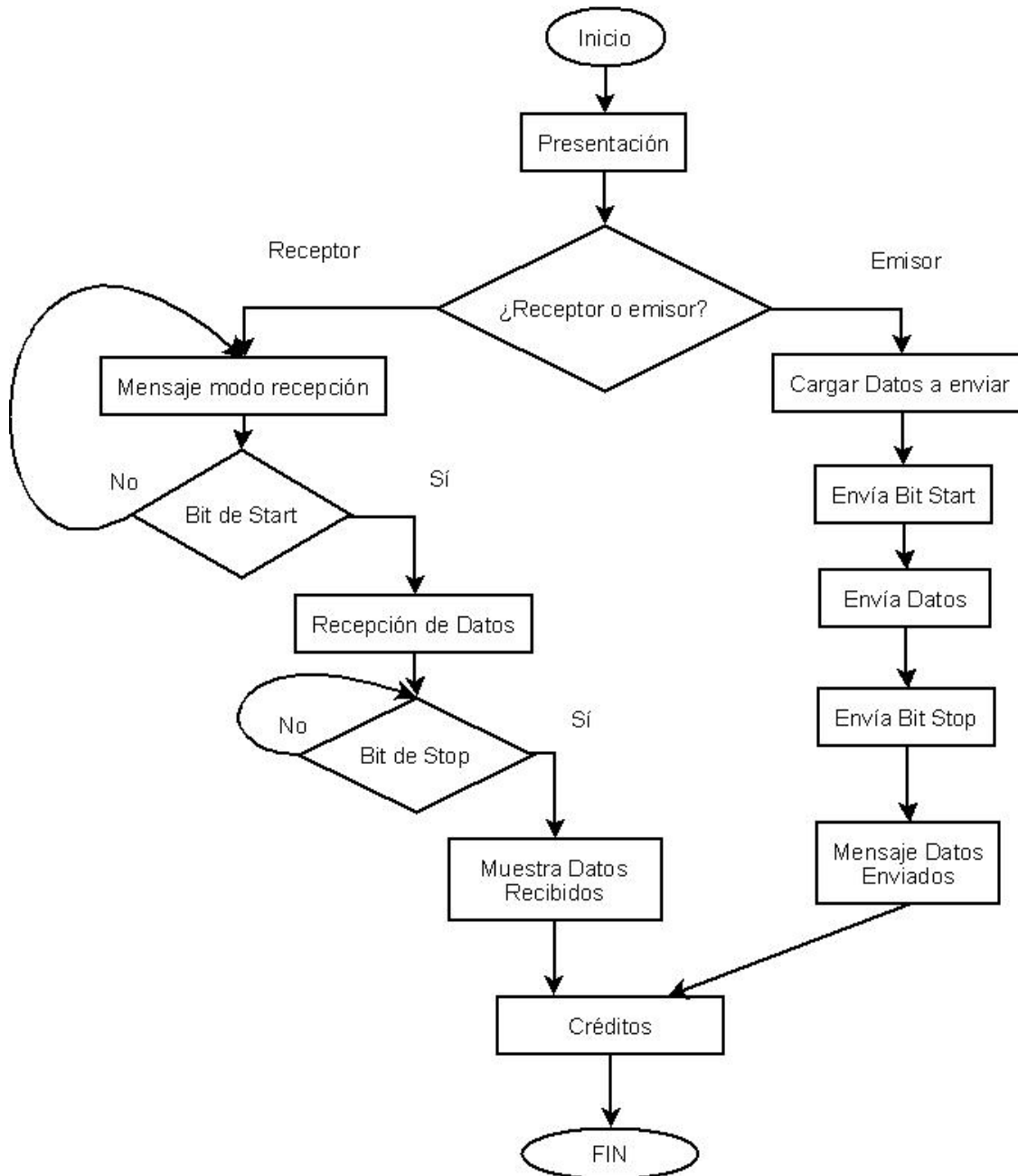


Esquema del conector JR1 para conectar la Trainer con el exterior:

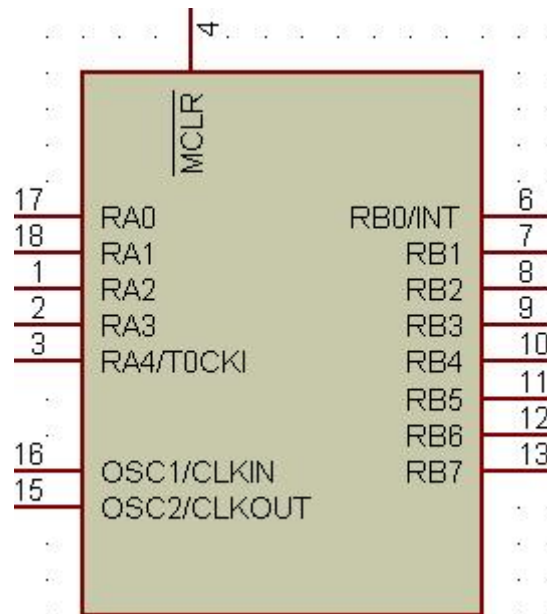


## Práctica básica:

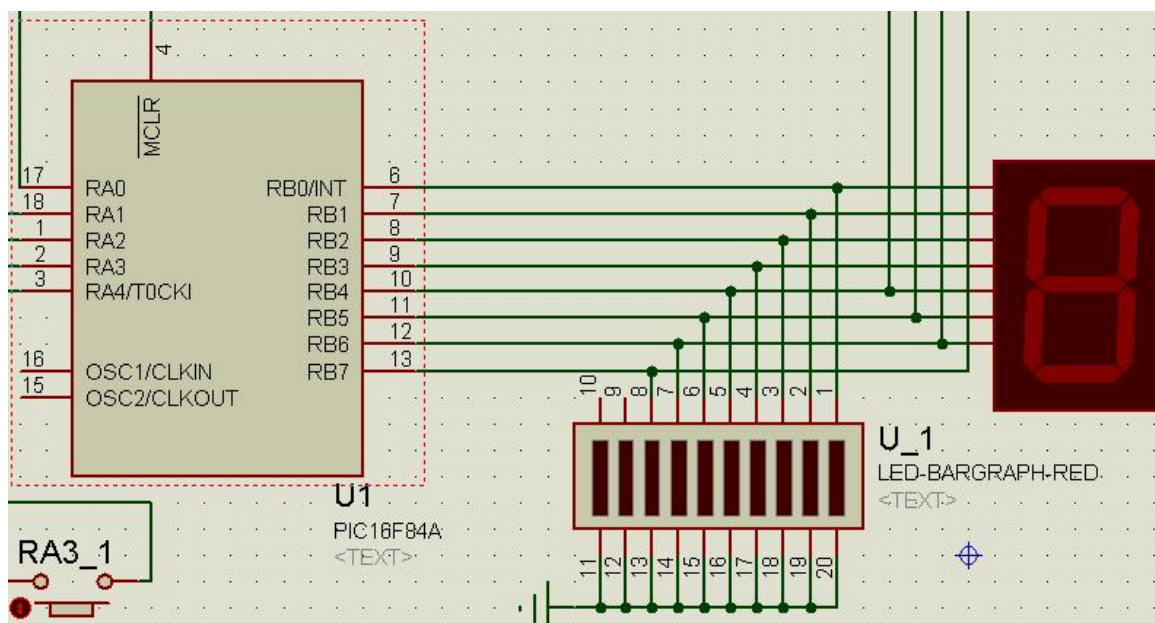
- Definimos el programa básico mediante este diagrama de flujo.



- A continuación usaremos el programa Proteus para diseñar el circuito, para ello hemos introducido el integrado PIC 16F84A.



- A continuación conectamos en las puertas RA3 y RA4/T0CKI unos interruptores con una resistencia de 10k en serie y lo conectamos a tierra, de esta forma tendremos la forma de introducir información al integrado.
- Implementaremos el esquema del MCLR tal y como observamos en los apuntes de la asignatura.
- En las puertas RB0 a RB7 implementaremos un banco de LEDs y un display de 7 segmentos (tal y como puede apreciarse en la imagen).



- Finalmente conectaremos una pantalla LCD siguiendo el esquema de los apuntes, por lo que lo conectaremos como podemos ver en la siguiente imagen.



# Código Fuente del programa básico:

## Plantilla.asm

```
; ** PLANTILLA.ASM *****
; **          Francisco Javier Merchán Macías          **
; **          Fatima Romero Romero                    **
; **          Esmeralda Rubio Parra                    **
; **          I.T.T.Telemática                        **
; *****
; ** Descripción práctica:                             **
; **          .                                         **
; ** Emisor y receptor.                                **
; **                                                    **
; **                                                    **
; **                                                    **
; *****
; ZONA DE DATOS *****
__CONFIG _CP_OFF& _WDT_OFF& _PWRTE_ON& _XT_OSC ; Configuración para el grabador
LIST P=16F84A ; Indica el tipo de mC utilizado.
INCLUDE <P16F84A.INC> ; Incluye las definiciones de registros y constantes asociadas
a este mC.
CBLOCK 0x0C ;La zona de memoria de usuario comienza en esta dirección de memoria RAM
de datos
ENVIO
ContadorEnvio
ContadorEmisor ;Contador para ENVIO.INC
ContadorReceptor ;Contador para RECIBIR.INC
DatosRecibido ;Para guardar el dato en RECIBIR.INC
ENDC
; CONSTANTES *****
#define Pulsador PORTA,4
#define PuertaEnviar PORTB,1
#define PuertaRecibir PORTB,0
; ZONA DE CODIGO *****
ORG 0
Inicio
; CONFIGURACIÓN DE ENTRADAS Y SALIDAS *****
bsf STATUS, RP0 ;Acceso al banco 1
bcf TRISB,1 ;RB1 como salida. (PUERTA DE ENVIO DE DATOS)
bsf TRISB,0 ;RB0 como entrada. (PUERTA DE RECEPCION DE DATOS)
bsf TRISA,3 ;RA3 como entrada (INTERRUPTOR)
bsf TRISA,4 ;RA4 como entrada (INTERRUPTOR)
bcf STATUS, RP0 ;Acceso al banco 0
;Aquí empieza el programa en sí...
Principal
call Presentacion
call Elegir
Dormir
sleep
goto Dormir
;-----INICIO Instrucciones para el inicio y presentacion-----
;-----
Presentacion
call LCD_Inicializa
call LCD_Borra
movlw MensajeBienvenida
call LCD_Mensaje
call Retardo_2s
movlw 0x40
call LCD_PosicionLineal
movlw MensajeElige
call LCD_Mensaje
call Retardo_2s
return
;-----FIN Instrucciones para el inicio y presentacion-----
;-----
; Mensajes -----
Mensajes
addwf PCL,F
MensajeBienvenida ;En subrutina Presentacion de PLANTILLA.ASM
DT "Practica SED", 0x00
MensajeElige ;En subrutina Presentacion de PLANTILLA.ASM
```

```

DT "Elige TX o RX:", 0x00
Mensaje0      ;En subrutina de ELIGIR.INC
DT "0=Recept", 0x00
Mensaje1      ;En subrutina de ELIGIR.INC
DT "1=Emisor", 0x00
MensajeDatCargado ;En subrutina de CARGANDO.INC
DT "Datos Cargados", 0x00
MensajeEmitiendo ;En subrutina de EMITIR.INC
DT "EMITIENDO RB1", 0x00
MensajeRecibiendo ;En subrutina de RECIBIR.INC
DT "RECIBIENDO RB0", 0x00
MensDatosEnv ;En subrutina de EMITIR.INC
DT "Datos ENVIADOS", 0x00
RecCorrect ;En subrutina de RECIBIR.INC
DT "RECEPCION OK", 0x00
MensajeBitStart ;En subrutina de RECIBIR.INC
DT "Bit Start", 0x00
MensajeBStop ;En subrutina de RECIBIR.INC
DT "Bit Stop ", 0x00
INCLUDE <LCD_4BIT.INC>
INCLUDE <LCD_MENS.INC>
INCLUDE <RETARDOS.INC>
INCLUDE <CARGANDO.INC> ;Includes Propios
INCLUDE <ELEGIR.INC>
INCLUDE <EMITIR.INC>
INCLUDE <RECIBIR.INC>
END

```

## Elegir.inc

```

; ** ELEGIR.INC *****
; **          Francisco Javier Merchán Macías          **
; **          Fatima Romero Romero                    **
; **          Esmeralda Rubio Parra                   **
; **          I.T.T.Telemática                        **
; *****
; ** Descripción:                                     **
; **                                                     **
; **  Libreria para elegir si es emisor o receptor    **
; **                                                     **
; **                                                     **
; *****
Elegir
    btfss Pulsador ;Si es 0 ejecuta la siguiente instrucción, si es 1 salta la instrucción
    goto ElegirEsCero
ElegirNoEsCero ;Esto es el EMISOR
    call LCD_Borra
    movlw Mensaje1
    call LCD_Mensaje
    call Retardo_2s
    call Cargando
    call Emitir
    goto ElegirFIN
ElegirEsCero ;Este es el RECEPTOR
    call LCD_Borra
    movlw Mensaje0
    call LCD_Mensaje
    call Retardo_2s
    call Recibir
    goto ElegirFIN
ElegirFIN
return

```

## Cargando.inc

```

; ** CARGANDO.INC *****
; **          Francisco Javier Merchán Macías          **
; **          Fatima Romero Romero                    **
; **          Esmeralda Rubio Parra                   **
; **          I.T.T.Telemática                        **
; *****

```

```

; ** Descripción: **
; ** ** **
; ** Libreria para cargar los datos a enviar **
; ** ** **
; ** ** **
; ** ** **
; *****
Cargando
call Retardo_2s
movlw 0x08
movwf ContadorEnvio ;Cargamos el contador con un 8
clrf ENVIO
call LCD_Inicializa ;Hacemos un cuadro para meter los valores que carguemos.
call LCD_Borra
movlw 0x03
call LCD_PosicionLineal
movlw '['
call LCD_Caracter
movlw 0x0c
call LCD_PosicionLineal
movlw ']'
call LCD_Caracter
movlw 0x04
call LCD_PosicionLineal
CargandoInicio ;Leemos el interruptor y lo cargamos en un registro.
btfsc Pulsador
goto CargandoUno
goto CargandoCero
CargandoUno
bsf STATUS,C
rrf ENVIO,1
; ESCRIBE UN 1 EN LA PANTALLA LCD
movlw '1'
call LCD_Caracter
call Retardo_2s
; DECREMENTAMOS EL CONTADOR
decfsz ContadorEnvio,1
goto CargandoInicio
goto CargandoFin
CargandoCero
bcf STATUS,C
rrf ENVIO,1
; ESCRIBE UN 0 EN LA PANTALLA LCD
movlw '0'
call LCD_Caracter
call Retardo_2s
; DECREMENTAMOS EL CONTADOR
decfsz ContadorEnvio,1
goto CargandoInicio
CargandoFin
call LCD_Borra
call LCD_Lineas2
movlw MensajeDatCargado
call LCD_Mensaje
call Retardo_2s
return

```

## Emitir.inc

```

; ** EMITIR.INC *****
; ** Francisco Javier Merchán Macías **
; ** Fatima Romero Romero **
; ** Esmeralda Rubio Parra **
; ** I.T.T.Telemática **
; *****
; ** Descripción: **
; ** ** **
; ** Emitir **
; ** ** **
; ** El registro que contiene los datos es ENVIO **
; ** PuertaEnviar: Emitimos por Puerta Emitir **
; ** PuertaRecibir: Recibimos por Puerta Recibir **
; *****
Emitir

```



```

; Comprobamos si el receptor es un receptor o emisor, para evitar problemas de emisión
movlw 0x08
movwf ContadorEmisor
; Empezamos a enviar el bit START
clrw      ;Ponemos mensaje EMITIENDO
call LCD_Borra
movlw MensajeEmitiendo
call LCD_Mensaje
call Retardo_1s
;PRUEBA PRUEBA PRUEBA PRUEBA PRUEBA PRUEBA PRUEBA PRUEBA
bsf TRISB,0      ;Ponemos a 1 las puertas de salida
bsf TRISB,1      ;Ponemos a 1 las puertas de salida
;PRUEBA PRUEBA PRUEBA PRUEBA PRUEBA PRUEBA PRUEBA PRUEBA

bcf PuertaEnviar      ;Enviamos un 0. BIT de START
call Retardo_1s
; Emitimos la cadena almacenada
ComenzamosAEmitir
rrf ENVIO,1
btfsc STATUS,C
goto EnviarUno      ;Envia un 1
goto EnviarCero      ;Envia un 0
EnviarUno
bsf PuertaEnviar
call Retardo_1s
decfsz ContadorEmisor,F
goto ComenzamosAEmitir
goto EmitirFIN
EnviarCero
bcf PuertaEnviar
; call Retardo_5s
call Retardo_1s
decfsz ContadorEmisor,F
goto ComenzamosAEmitir
; goto EmitirFIN      ; No hace falta esta instrucción porque tenemos el código debajo.
EmitirFIN
;Ponemos el BIT de STOP
bsf PuertaEnviar
call Retardo_2s
call LCD_Borra
movlw MensDatosEnv
call LCD_Mensaje
call Retardo_2s
return

```

## Recibir.inc

```

; ** RECIBIR.INC *****
; **          Francisco Javier Merchán Macías          **
; **          Fatima Romero Romero                    **
; **          Esmeralda Rubio Parra                    **
; **          I.T.T.Telemática                        **
; *****
; ** Descripción:                                     **
; **                                                     **
; ** Recibir                                           **
; **                                                     **
; ** Emitimos por RA1                                 **
; ** Recibimos por RA0 (Para comprobar si el otro es emisor o no)**
; *****
Recibir
clrw
movlw 0x08
movwf ContadorReceptor ;Cargamos el contador con un 8
clrw
call LCD_Borra
movlw MensajeRecibiendo
call LCD_Mensaje
call Retardo_2s
;-----PRUEBA-----
;Comprobar
; btfsc PuertaRecibir
; goto RecibirUno

```

```

; goto RecibirCero
;RecibirUno
; call LCD_Borra
; movlw RecibimosUno
; call LCD_Mensaje
; call Retardo_100ms
; goto Comprobar
;RecibirCero
; call LCD_Borra
; movlw RecibimosCero
; call LCD_Mensaje
; call Retardo_100ms
; goto Comprobar
;-----FIN PRUEBA-----
;Empezamos a Recibir el bit START
BitStart
;Ponemos a 1 las puertas para evitar Falsos Positivos
;PRUEBA PRUEBA PRUEBA PRUEBA PRUEBA PRUEBA PRUEBA PRUEBA
bsf TRISB,0 ;Ponemos a 1 las puertas de salida
bsf TRISB,1 ;Ponemos a 1 las puertas de salida
;PRUEBA PRUEBA PRUEBA PRUEBA PRUEBA PRUEBA PRUEBA PRUEBA
btfsc PuertaRecibir ;COMPROBAMOS SI ES UN 0 (Bit de Start)
goto BitStart ;Si es 1 vuelve a comprobar hasta que le llega un 0
;Si es 0 POSIBLEMENTE tenemos el bit de Start y continuamos el codigo.
call Retardo_200ms
call Retardo_200ms
call Retardo_100ms ;Esperamos medio segundo para comprobar si es un 0 o no
btfsc PuertaRecibir ;VOLVEMOS A COMPROBAR QUE ES UN 0.
goto BitStart ;No es el bit de Start, es un error.
call LCD_Borra ;Si es el bit de Start lo muestra en pantalla
call LCD_Lineal
movlw MensajeBitStart
call LCD_Mensaje
call Retardo_200ms
call Retardo_200ms
call Retardo_100ms ;Esperamos otro medio segundo
call Retardo_200ms ;200ms más para no estar al comienzo del bit, sino un poco más al
medio.
call LCD_Lineal2
DRecibido ;SI ES EL BIT DE START (comenzamos a recibir)
btfsc PuertaRecibir
goto RecUno
goto RecCero
RecUno
; ESCRIBE UN 1 EN LA PANTALLA LCD
movlw '1'
call LCD_Caracter
call Retardo_1s
; GUARDAMOS LO QUE RECIBIMOS EN UN REGISTRO
bsf STATUS,C
rrf DatosRecibido,1
; DECREMENTAMOS EL CONTADOR
decfsz ContadorReceptor,1
goto DRecibido ;Ir al inicio de DRecibido o lo que es lo mismo DatoRecibido
goto BitStop
RecCero
; ESCRIBE UN 0 EN LA PANTALLA LCD
movlw '0'
call LCD_Caracter
call Retardo_1s
; GUARDAMOS LO QUE RECIBIMOS EN UN REGISTRO
bcf STATUS,C
rrf DatosRecibido,1
; DECREMENTAMOS EL CONTADOR
decfsz ContadorReceptor,1
goto DRecibido ;Ir al inicio de DRecibido o lo que es lo mismo DatoRecibido

;*****PARTE DE COMPROBACION DEL BIT DE STOP
BitStop
call Retardo_1s
btfsc PuertaRecibir ;COMPROBAMOS SI ES UN 0
goto EncontramosBitStop ;Si es 1 tenemos el bit de Stop
goto BitStop ;No tenemos bit de stop.
EncontramosBitStop
call LCD_Lineal
movlw MensajeBStop

```

```

call LCD_Mensaje
call Retardo_2s
;*****PARTE DE COMPROBACION DEL BIT DE STOP
;BitStop
; call Retardo_1s
; call LCD_Lineal
; movlw MensajeBStop
; call LCD_Mensaje
; call Retardo_2s

FinReceptor
call LCD_Borra
movlw RecCorrect
call LCD_Mensaje
call Retardo_2s
PintarEnLeds
; call LCD_OFF ;Apagamos el LCD
; CONFIGURACIÓN DE ENTRADAS Y SALIDAS *****
bsf STATUS, RP0 ;Acceso al banco 1
clrf TRISB ;Todas las líneas del Puerto B como salida
bcf STATUS, RP0 ;Acceso al banco 0
; FIN DE CONFIGURACIÓN DE LAS ENTRADAS Y SALIDAS
; MOSTRAMOS EL DATO RECIBIDO EN LOS LEDS
rrf DatosRecibido,1
btfsc STATUS,C
goto RB7_1
goto RB7_0
RB7_1
bsf PORTB,7
goto RB6_Continua
RB7_0
bcf PORTB,7
RB6_Continua
rrf DatosRecibido,1
btfsc STATUS,C
goto RB6_1
goto RB6_0
RB6_1
bsf PORTB,6
goto RB5_Continua
RB6_0
bcf PORTB,6
RB5_Continua
rrf DatosRecibido,1
btfsc STATUS,C
goto RB5_1
goto RB5_0
RB5_1
bsf PORTB,5
goto RB4_Continua
RB5_0
bcf PORTB,5
RB4_Continua
rrf DatosRecibido,1
btfsc STATUS,C
goto RB4_1
goto RB4_0
RB4_1
bsf PORTB,4
goto RB3_Continua
RB4_0
bcf PORTB,4
RB3_Continua
rrf DatosRecibido,1
btfsc STATUS,C
goto RB3_1
goto RB3_0
RB3_1
bsf PORTB,3
goto RB2_Continua
RB3_0
bcf PORTB,3
RB2_Continua
rrf DatosRecibido,1
btfsc STATUS,C
goto RB2_1

```

```

goto RB2_0
RB2_1
bsf PORTB,2
goto RB1_Continua
RB2_0
bcf PORTB,2
RB1_Continua
rrf DatosRecibido,1
btfsc STATUS,C
goto RB1_1
goto RB1_0
RB1_1
bsf PORTB,1
goto RB0_Continua
RB1_0
bcf PORTB,1
RB0_Continua
rrf DatosRecibido,1
btfsc STATUS,C
goto RB0_1
goto RB0_0
RB0_1
bsf PORTB,0
goto RB_Fin
RB0_0
bcf PORTB,0
RB_Fin
return

```

## Explicación del funcionamiento del programa básico:

El objetivo principal de esta práctica es que el alumno realice un programa en lenguaje ensamblador que reproduzca el funcionamiento de una comunicación serie entre dos PIC 16F84A.

Para ello, lo primero que hacemos es mostrar un mensaje de bienvenida como presentación a la práctica, este mensaje es el mismo en las dos placas.

Después damos al usuario la oportunidad de elegir si quiere que la placa haga el papel de emisor o de receptor, poniendo el switch RA4 a uno o a cero, respectivamente. El PIC interroga el estado de este interruptor y otorga a cada placa su rol correspondiente. Teniendo en cuenta que éste debe ser distinto para cada placa, es decir, una debe ser emisor y la otra, receptor.

La placa receptora, se quedará esperando recibir el bit de start, mientras en la placa emisora, el usuario va introduciendo manualmente los datos que desea transmitir, moviendo el switch RA4 según quiera transmitir un cero o un uno, hasta llegar a 8 bits.

Una vez que tenemos los datos que queremos transmitir, la placa emisora envía un cero (el bit de start), después envía los datos y por último envía un uno (el bit de stop).

La placa receptora que estaba esperando, recibe el bit de start, y después va recibiendo bits de datos y los va guardando en un registro, por último recibe el bit de stop.

El registro que guarda los datos recibidos hay que rotarlo a la derecha, porque al enviar, el programa ha ido enviando primero los bits de menor peso, por lo que al

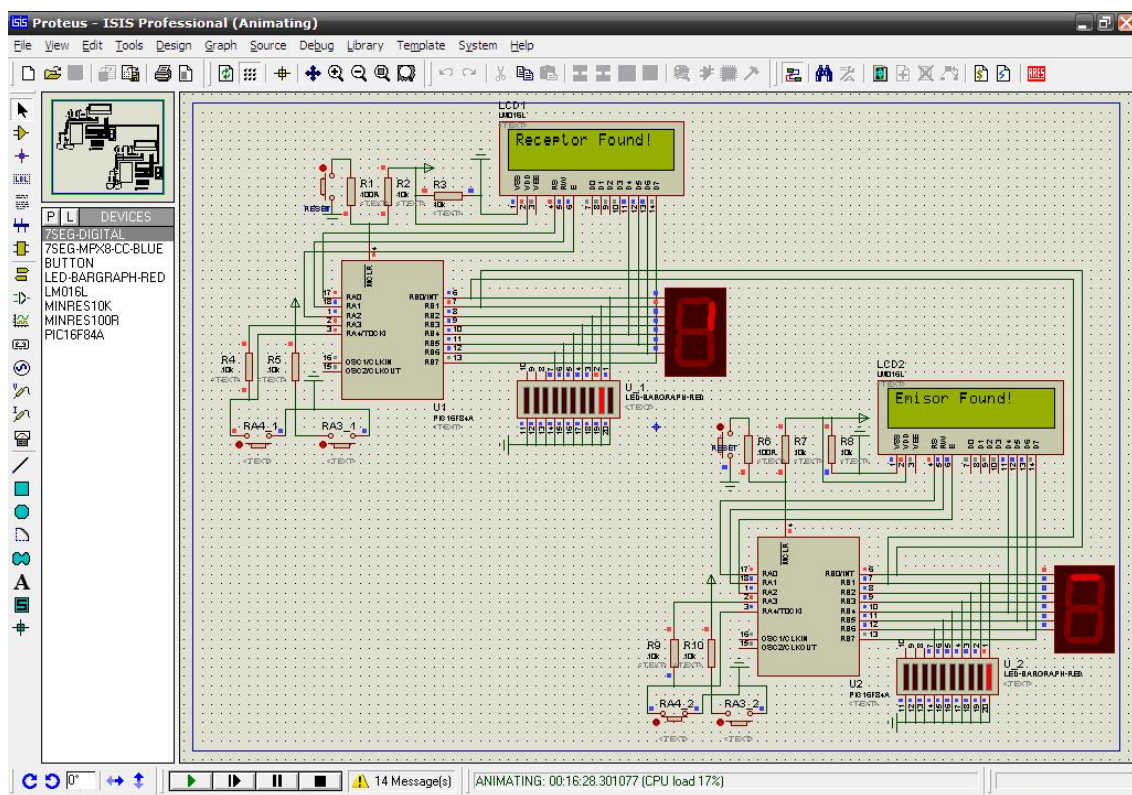
recibir nos llegan también primero los de menor peso. Es decir, tenemos los datos justo al revés de cómo eran antes de enviarlos.

Una vez reordenados los datos, procedemos a la iluminación del array de LEDs de la placa, que será: si el bit de dato es un cero, el LED permanecerá apagado; si el bit de dato es un uno, el LED se enciende.

Por último se muestran los créditos, esto es, un mensaje con los nombres de los componentes del grupo.

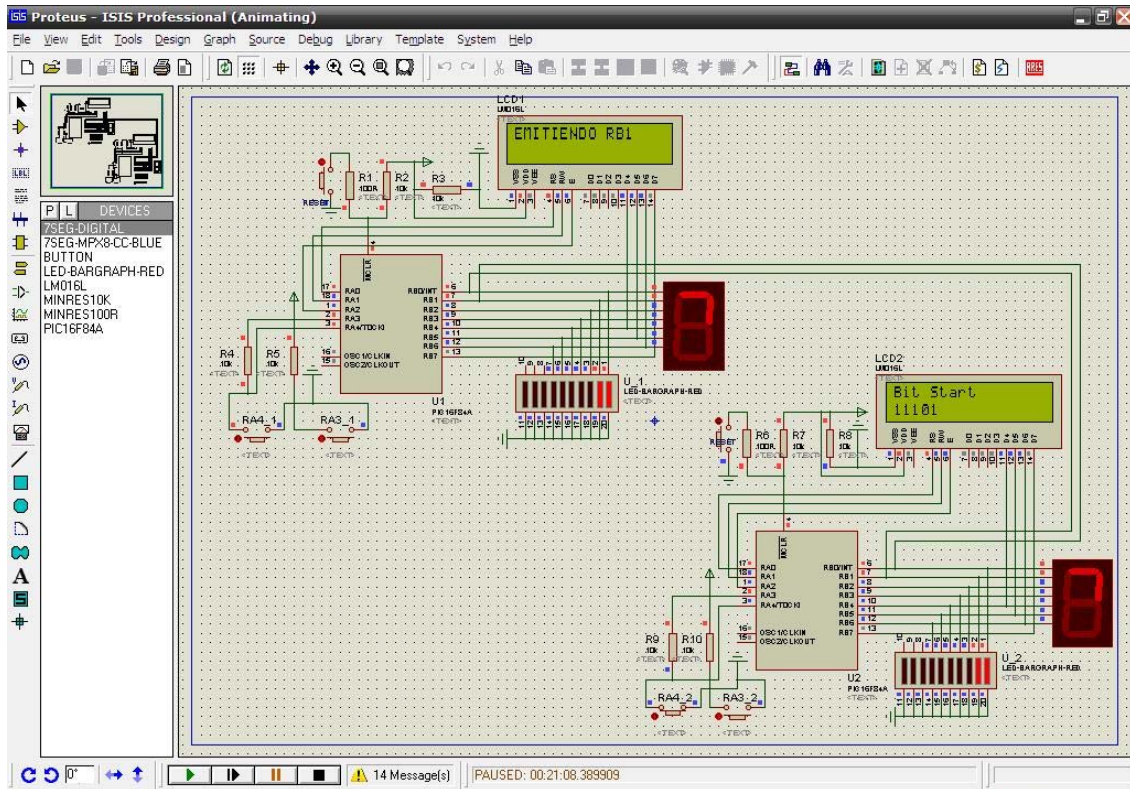
## Capturas del Proteus, de un funcionamiento básico:

### Detector emisor y receptor, detectando el emisor y el receptor:

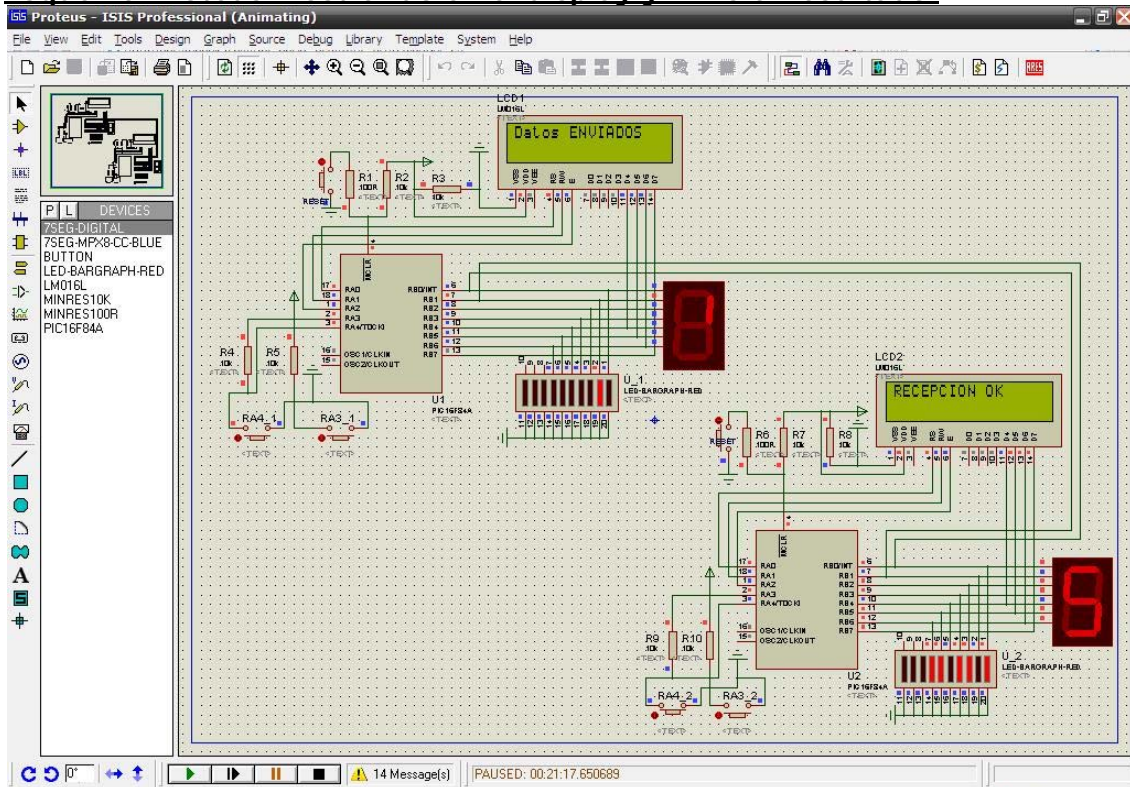




## Esquema Proteus con un emisor y un receptor (emitiendo):



## Esquema Proteus mostrando en el display y LEDs el resultado:



## Comentarios sobre la realización del programa básico :

### Problema con las puertas que tuvimos y por el cual nos dejó las placas

Hemos tenido problemas en varias cosas:

- Teníamos problemas con el bucle de decremento para ir almacenando los datos. El programa se volvía loco y no paraba cuando le introducíamos los 8 bits, y el problema estaba en la instrucción “decfsz ContadorEnvio,1” que no le poníamos el 1 para indicarle que queríamos guardar el decremento en el mismo registro en vez de guardarlo en W.
- Por otro lado, también había cierto descontrol en los archivos include por lo que hubo que borrar la instrucción “CBLOCK” y añadir un “return” en los ficheros include, ya que no los pusimos y el programa daba errores de funcionamiento.
- También hubo un problema con la carga de la librería CARGANDO.INC. El problema fue que habíamos hecho un salvado con un nombre diferente y los cambios los hacíamos en un fichero diferente, por lo que al compilar siempre hacia cosas raras ya que compilaba con el fichero antiguo y no con el que íbamos modificando.
- Sobre todo ha costado en sincronizar para que detectase bien los bits. Hemos hecho múltiples pruebas para cuadrarlo perfectamente hasta conseguir que nunca falle.

## Mejoras:

### 1. Detección de los roles (receptor-receptor, emisor-emisor y emisor-receptor y viceversa) de las placas.

#### Explicación del funcionamiento y el porqué de la mejora

Si tenemos un PIC en modo Receptor esperando el bit de Start y en ese momento reiniciamos el otro PIC (da igual el modo que esté porque se ha reseteado) el Pic Receptor detecta un 0 y se pone a coger 8 ceros. Por eso implementamos la mejora de comprobar si es emisor o receptor.

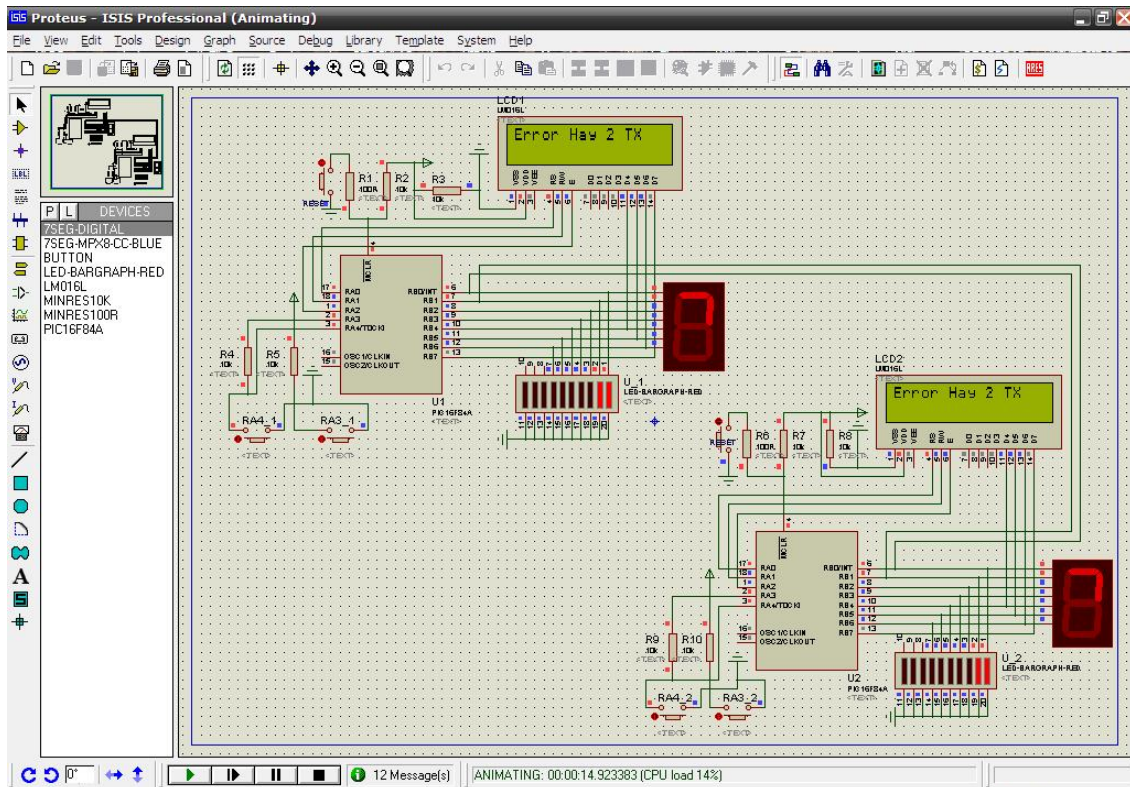
#### Funcionamiento:

El funcionamiento de esta mejora se encuentra explicado en el apartado de la Mejora sustancial de la práctica, anteriormente diseñada a 2 hilos, utilizando 4 hilos.

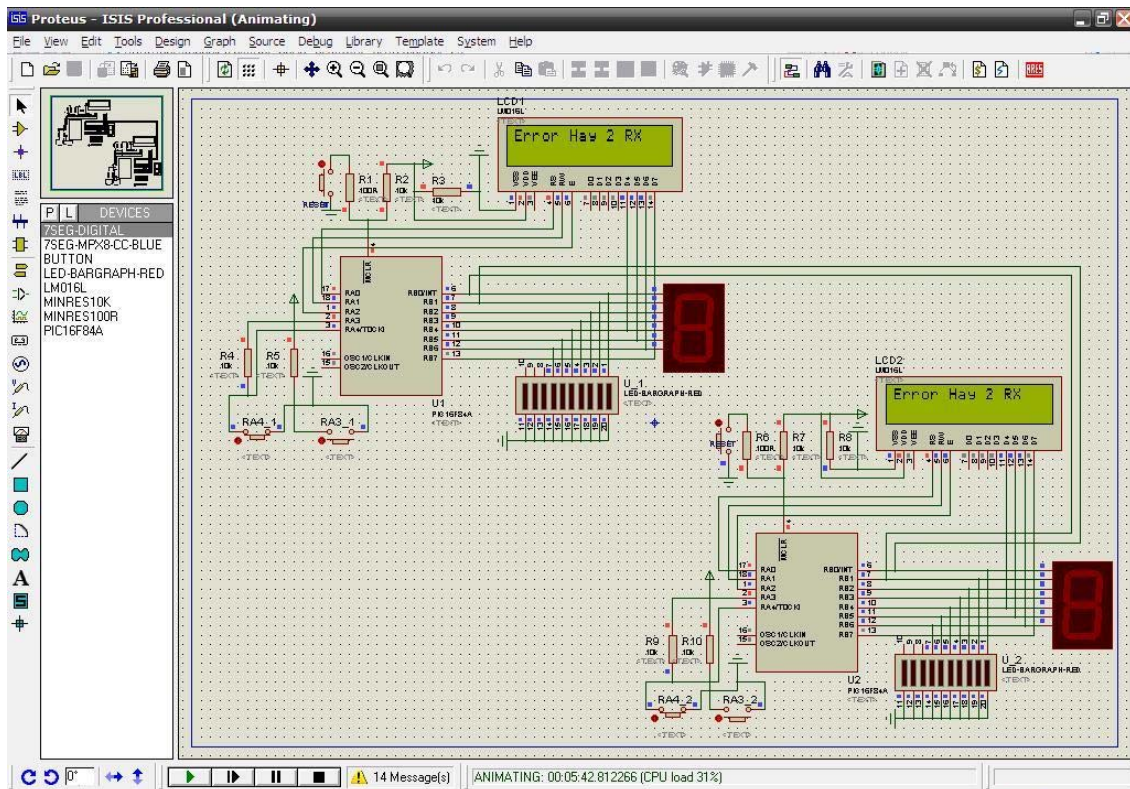


## Capturas de pantalla de proteus:

### Esquema Proteus con dos emisores (dando error):



### Esquema Proteus con dos receptores (dando error):





```

; ** ELEGIR.INC *****
; **          Francisco Javier Merchán Macías          **
; **          Fatima Romero Romero                    **
; **          Esmeralda Rubio Parra                    **
; **          I.T.T.Telemática                        **
; *****
; ** Descripción:                                     **
; **                                                     **
; **   Libreria para elegir si es emisor o receptor   **
; **                                                     **
; **                                                     **
; **                                                     **
; *****
Elegir
    btfss Pulsador    ;Si es 0 ejecuta la siguiente instrucción, si es 1 salta la
instrucción
    goto ElegirEsCero
; //////////////////////////////////////
; ////////////////////////////////////// INICIO EMISOR //////////////////////////////////////
; //////////////////////////////////////
; //////////////////////////////////////
ElegirNoEsCero    ;Esto es el EMISOR *****
    call LCD_Borra
    movlw Mensaje1
    call LCD_Mensaje
    call Retardo_2s
;Informar al otro pic que somos Emisor
; bcf SalidaComprobar    ;Ponemos a 0 la Puerta RB3
    bsf PuertaEnviar
    call Retardo_5s    ;Esperamos antes de comprobar las entradas
;-----Comprobamos que el otro pic es Receptor
; btfsc EntradaComprobar
    btfss PuertaRecibir
    goto BIENTenemosReceptor    ;RB2 = 1 ESTA BIEN, TEMEMOS UN RECEPTOR AL OTRO LADO.
    goto ErrorHayDosEmisores    ;RB2 = 0 No hay receptor al otro lado!!!!
;-----Comprobamos que el otro pic es Receptor
BIENTenemosReceptor
;*****

```

```

; ** ENCONTRAMOS EL RECEPTOR **
; *****
    clrw
    call LCD_Borra
    movlw MensajeReceptorEncontrado
    call LCD_Mensaje
    call Retardo_2s
    call Cargando
    call Emitir
    goto ElegirFIN
; ////////////////////////////////////////
; //////////////////////////////////////// INICIO RECEPTOR ////////////////////////////////////////
; ////////////////////////////////////////
ElegirEsCero    ;Este es el RECEPTOR *****
    call LCD_Borra
    movlw Mensaje0
    call LCD_Mensaje
    call Retardo_2s
; Informar al otro pic que somos Receptor
; bsf SalidaComprobar    ; Ponemos a 0 la Puerta RB3
    bcf PuertaEnviar
    call Retardo_5s    ; Esperamos antes de comprobar las entradas
; -----Comprobamos que el otro pic es Emisor
; btfsc EntradaComprobar
    btfss PuertaRecibir
    goto ErrorHayDosReceptores    ; RB2 = 0 No hay receptor al otro lado!!!!
    goto BIENTenemosEmisor    ; RB2 = 1 ESTA BIEN, TEMEMOS UN RECEPTOR AL OTRO LADO.
; -----Comprobamos que el otro pic es Receptor
BIENTenemosEmisor
; *****
; ** ENCONTRAMOS EL EMISOR **
; *****
    clrw
    call LCD_Borra
    movlw MensajeEmisorEncontrado
    call LCD_Mensaje
    call Retardo_2s
    call Recibir
    goto ElegirFIN
ErrorHayDosEmisores
    clrw
    call LCD_Borra
    movlw MensajeErrorHayDosEmisores
    call LCD_Mensaje
    call Retardo_2s
    goto Dormir
ErrorHayDosReceptores
    clrw
    call LCD_Borra
    movlw MensajeErrorHayDosReceptores
    call LCD_Mensaje
    call Retardo_2s
    goto Dormir
ElegirFIN
return

```

## 2. Detección del bit de Start en dos ocasiones, para fiabilizar el funcionamiento de la práctica.

### Motivación y funcionamiento de la mejora:

Comprobamos una vez más el bit de Start, para asegurarnos que realmente el bit cero que recibimos es el bit de Start, ya que uno de los motivos de la realización de la mejora anterior se debía a que se podía recibir un bit de Start erróneo debido a los rebotes en los interruptores, provocado por el cambio de bits en la puertas de recepción y envío, al comprobarlo dos veces, estamos seguros de que lo que venga después serán los datos que nos envían, y no bits “perdidos” o erróneos.

### 3. Emulación de las luces del “coche fantástico” con el banco de leds y el display de 7 segmentos, indicando la finalización de la tarea.

#### Motivo de la mejora:

Con el fin de mejorar el aspecto visual de la práctica y hacerla más amigable, hemos introducido como mejora unos efectos lumínicos con el banco de leds y el display de 7 segmentos. Simplemente se trata de cargar un literal con unos y ceros al registro de trabajo (W) y enviarlos a la puerta de salida, que en nuestro caso son las puertas B (RB0 a RB7) y esperar un tiempo. Repitiendo esto de forma rápida hemos conseguido un efecto de luces en los LEDs y si tenemos activado el display de siete segmentos, observaremos un efecto lumínico con menos coherencia que el banco de LEDs. La implementamos simplemente como una llamada CALL LucesInicio en el fichero PLANTILLA.ASM e incluyendo dicho fichero con un INCLUDE <LUCES.INC>

#### Código fuente del mismo

```
; ** LUCES.INC *****
; **          Francisco Javier Merchán Macías          **
; **          Fatima Romero Romero                    **
; **          Esmeralda Rubio Parra                   **
; **          I.T.T.Telemática                        **
; *****
; ** Descripción práctica:                             **
; **          .                                         **
; ** Procedimiento para generar luces variadas en el banco de **
; ** LED y en el display de 7 segmentos.                **
; **                                                     **
; **                                                     **
; *****
LucesInicio
    bsf STATUS,RP0    ;Configuramos las puertas RB0 a RB7 como salidas
    clrf PORTB
    bcf STATUS,RP0
LucesContinua
    movlw b'00000000'
    movwf PORTB
    call Retardo_200ms
    movlw b'10000001'
    movwf PORTB
    call Retardo_200ms
    movlw b'01000010'
    movwf PORTB
    call Retardo_200ms
    movlw b'00100100'
    movwf PORTB
    call Retardo_200ms
    movlw b'00011000'
    movwf PORTB
    call Retardo_200ms
    movlw b'00111100'
    movwf PORTB
    call Retardo_200ms
    movlw b'01111110'
    movwf PORTB
    call Retardo_200ms
    movlw b'11111111'
    movwf PORTB
    call Retardo_200ms
    movlw b'11100111'
    movwf PORTB
    call Retardo_200ms
    movlw b'11000011'
    movwf PORTB
    call Retardo_200ms
```

```

movlw b'10000001'
movwf PORTB
call Retardo_200ms
movlw b'10000001'
movwf PORTB
call Retardo_200ms
movlw b'11000011'
movwf PORTB
call Retardo_200ms
movlw b'11100111'
movwf PORTB
call Retardo_200ms
movlw b'11111111'
movwf PORTB
call Retardo_200ms
movlw b'11111111'
movwf PORTB
call Retardo_200ms
movlw b'01111110'
movwf PORTB
call Retardo_200ms
movlw b'00111100'
movwf PORTB
call Retardo_200ms
movlw b'00011000'
movwf PORTB
call Retardo_200ms
movlw b'00100100'
movwf PORTB
call Retardo_200ms
movlw b'01000010'
movwf PORTB
call Retardo_200ms
movlw b'10000001'
movwf PORTB
call Retardo_200ms
movlw b'00000000'
movwf PORTB
call Retardo_200ms
movlw b'00000001'
movwf PORTB
call Retardo_200ms
movlw b'00000010'
movwf PORTB
call Retardo_200ms
movlw b'00000100'
movwf PORTB
call Retardo_200ms
movlw b'00001000'
movwf PORTB
call Retardo_200ms
movlw b'00010000'
movwf PORTB
call Retardo_200ms
movlw b'00100000'
movwf PORTB
call Retardo_200ms
movlw b'01000000'
movwf PORTB
call Retardo_200ms
movlw b'10000000'
movwf PORTB
call Retardo_200ms
movlw b'00000000'
movwf PORTB
call Retardo_200ms
movlw b'10000000'
movwf PORTB
call Retardo_200ms
movlw b'01000000'
movwf PORTB
call Retardo_200ms
movlw b'00100000'
movwf PORTB
call Retardo_200ms
movlw b'00010000'
movwf PORTB

```

```

call Retardo_200ms
movlw b'00001000'
movwf PORTB
call Retardo_200ms
movlw b'00000100'
movwf PORTB
call Retardo_200ms
movlw b'00000010'
movwf PORTB
call Retardo_200ms
movlw b'00000001'
movwf PORTB
call Retardo_200ms
goto LucesContinua

```

4. Mejora sustancial de la práctica, anteriormente diseñada a 2 hilos, utilizando 4 hilos, ya que se detectó que en algunas ocasiones que podría detectarse un fallo debido al funcionamiento de las puertas usadas en la versión a 2 hilos.

### ¿Por qué necesitamos esta mejora?

En las prácticas anteriores usábamos dos hilos de conexión entre los PIC, es decir, un hilo para emitir y otro para recibir. A la hora de enviar datos no había problemas, ya que enviábamos sin saber si el receptor existía o no, por lo que diseñamos el detector de emisor y receptor. Una vez implementado en la práctica de dos hilos de conexión el detector, a veces podía tener problemas al detectar correctamente al otro dispositivo, ya que transcurridos 5 segundos de la elección de rol (emisor o receptor) las puertas de salida cambian. Puesto que nuestro detector se basa en el estado de las puertas de salida del otro circuito, si un receptor esperaba encontrar un 0 en la puerta de entrada y un emisor se esperaba encontrar un 1 en la puerta de entrada, al pasar un tiempo dichos valores se habían modificado, por lo que daba falsos positivos en ciertas ocasiones, como por ejemplo, cuando un receptor estaba esperando el bit de Start y reseteamos en ese mismo momento el otro circuito (dando igual si está en modo emisor o receptor), el receptor que esperaba el bit de Start, en ese momento detectaba el bit 0 del otro circuito, por lo que se suponía que era el bit Start y empezaba a recoger el valor 0 hasta que no encontraba el bit de stop y se quedaba esperando. Para evitar esto, había dos posibles soluciones:

- Implementar 2 líneas más (emisor y receptor) para informar el rol de cada circuito (que es el caso que explicamos en este apartado).
- Cambiar la forma de identificar al emisor y al receptor, por lo que en prácticas anteriores usábamos la PuertaEnviar (PORTB,1) con el valor 0 para identificar el emisor y el valor 1 para identificar el receptor, por lo que el valor 0 que envía el emisor al receptor hace que el receptor detecte un bit 0 en su puerta de entrada y lo confunda con el bit de Start. Para solucionarlo simplemente cambiamos los valores que identificaban a cada circuito, quedando al emisor como valor 1 y al receptor como valor 0, es decir, el emisor en PuertaEnviar (PORTB,1) la ponía a 1 y el receptor en su PuertaEnviar (PORTB,1) la ponía a 0.

## Cambios de la práctica de 2 hilos a 4 hilos:

Para implementar esta mejora simplemente creamos dos puertas más, una de entrada y otra de salida, dejando de forma independiente el camino de envío de datos y por otro el identificación de roles. Para ello le asignamos a la PORTB,2 el nombre de EntradaComprobar y al PORTB,3 el nombre SalidaComprobar.

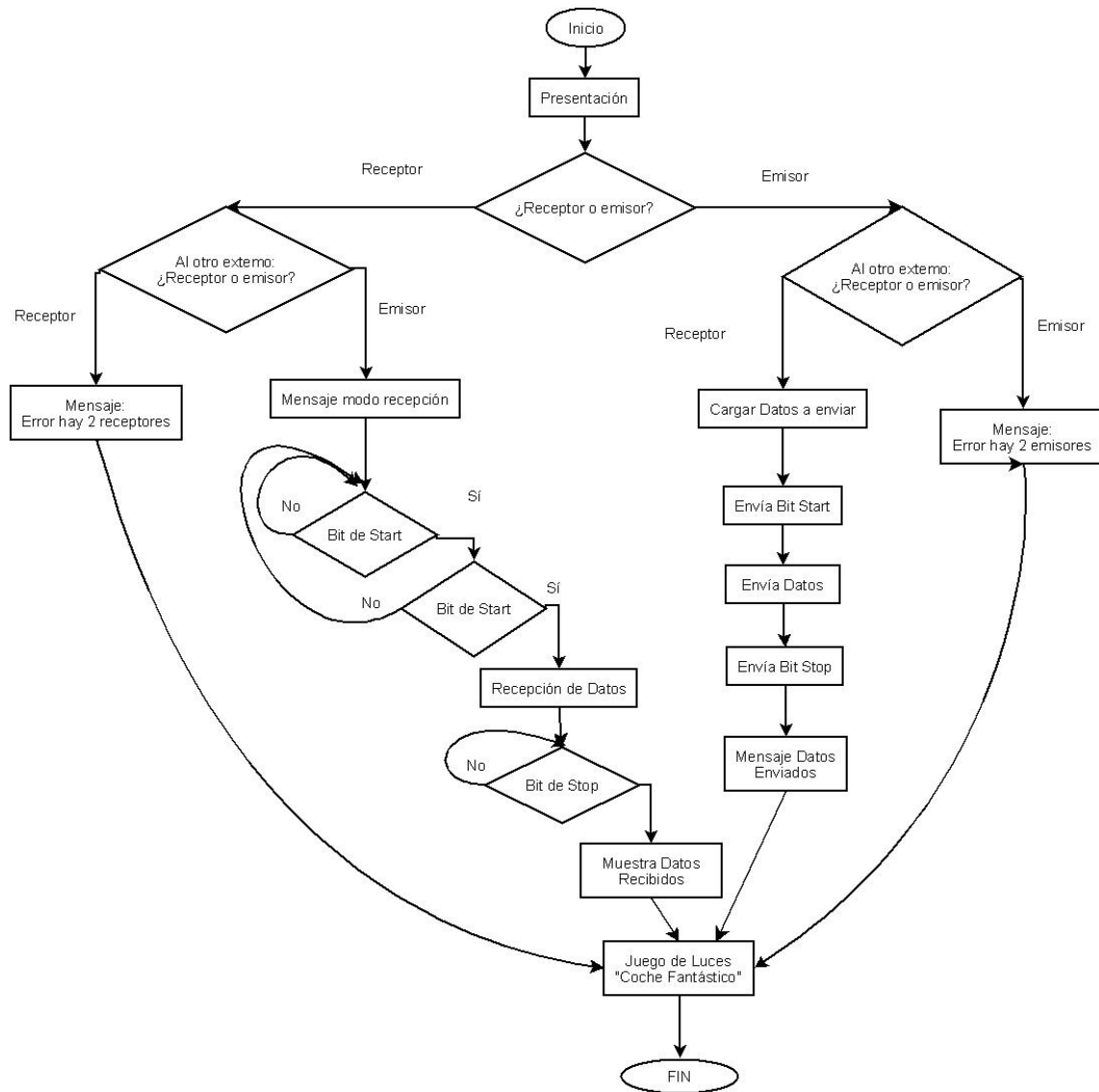
```
#DEFINE PuertaEnviar PORTB,1
#DEFINE PuertaRecibir PORTB,0
#DEFINE SalidaComprobar PORTB,3
#DEFINE EntradaComprobar PORTB,2
```

A continuación configuramos dichas puertas como entradas o salidas.

```
bsf    STATUS, RP0        ;Acceso al banco 1
bcf    TRISB,1             ;RB1 como salida. (PUERTA DE ENVIO DE
                           ;DATOS)
bsf    TRISB,0             ;RB0 como entrada. (PUERTA DE RECEPCION
                           ; DE DATOS)
bcf    TRISB,3             ;RB3 como salida. COMPROBACIONES.
Bsf    TRISB,2             ;RB2 como entrada. COMPROBACIONES.
bsf    TRISA,3             ;RA3 como entrada (INTERRUPTOR)
bsf    TRISA,4             ;RA3 como entrada (INTERRUPTOR)
bcf    STATUS, RP0        ;Acceso al banco 0
```

Y finalmente en el include ELEGIR.INC cuando elegimos un emisor ponemos la puerta SalidaComprobar a 0 y si es receptor ponemos la puerta SalidaComprobar a 1, de esta forma, dejaremos fijos estos valores a lo largo del programa y en todo momento el otro circuito podrá saber si tiene un emisor o un receptor al otro lado de los hilos. Aún podría haber un falso positivo, es decir, si encendemos un posible receptor, al comprobar su puerta de entrada, detectará un 0 al no tener nada conectado. Pero ese caso no lo hemos contemplado ya que no se trata de diseñar un protocolo de comunicaciones que soluciones todos y cada uno de los posibles casos que se puedan dar.

## Diagrama de la práctica con mejoras



## Bibliografía

- Apuntes de clase. Sistemas Electrónicos Digitales. Juan Ángel García Martínez.
- Microcontrolador PIC 16F84 Desarrollo de proyectos. Enrique Palacios. Ra-Ma.