Treball de Fi de Grau

Grau en Enginyeria en Tecnologies Industrials

Disseny i construcció d'aparells electrònics per identificar el color d'objectes

ANNEX

Autores: Cristina Rodríguez Murciano
Judit Sanahuja Roig
Directora: Rosa Rodríguez Montañés

Convocatòria: Gener 2017

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Sumari

1. Circuit_1.c

```
/*
Treball Final de Grau: Disseny i construcció d'aparells electrònics per identificar el color d'objectes
Autores: Cristina Rodríguez Murciano i Judit Sanahuja Roig
Directora: Rosa Rodríguez Montañés
Convocatòria: Gener 2017
*/

// PROGRAMACIÓ DEL MICROCONTROLADOR PER AL CIRCUIT 1: CONNEXIÓ AMB DISPOSITIU ANDROID

// CONFIGURACIÓ GENERAL INICIAL
#include <pic16f690.h>  // S'inclou la llibreria del PIC16F690
#include <xc.h>     // S'inclou la llibreria del compilador XC
#include "uart.h"   // S'inclou el fitxer UART.h
#define _XTAL_FREQ 4000000  // Es defineix la freqüència de funcionament del microcontrolador (4MHz)
#define ON  1   // Es defineix que ON és 1
#define OFF 0   // Es defineix que OFF és 0


// CONFIGURACIÓ MICROCONTROLADOR
#pragma config FOSC = INTRCIO // Es selecciona l'oscil·lador intern mitjançant els Oscillator Selection bits.
                              // Disponibilitat dels pins RA4 i RA5
#pragma config WDTE = OFF     // Es desactiva el Watchdog Timer
#pragma config PWRTE = OFF    // Es desactiva el Power-up Timer
#pragma config MCLRE = OFF    // Entrada digital
#pragma config CP = OFF       // S'activa el program memory code protection
#pragma config CPD = OFF      // S'aciva el data memory code protection
#pragma config BOREN = OFF    // Es desactiven els brown-out selection bits. BOR desactivat
#pragma config IESO = OFF     // Es desactiva el mode Internal External Switchover
#pragma config FCMEN = OFF    // Es desactiva el Fail-Safe Clock Monitor


// INFORMACIÓ DEL SENSOR
#define S0 RA0 // Es defineix que S0 és el pin RA0 del microcontrolador
#define S1 RA1 // Es defineix que S1 és el pin RA1 del microcontrolador
#define S2 RA2 // Es defineix que S2 és el pin RA2 del microcontrolador
#define S3 RA4 // Es defineix que S3 és el pin RA4 del microcontrolador
// OUT al RA5 que és el senyal del qual es vol mesurar la seva freqüència (rellotge extern TMR1 al pin RA5)


//INFORMACIÓ LED RGB
#define LEDR RC6 // Es defineix que LEDR és el pin RC6
#define LEDG RC1 // Es defineix que LEDG és el pin RC1
#define LEDB RC7 // Es defineix que LEDB és el pin RC7

//DEFINICIÓ I INICIALITZACIÓ DE VARIABLES
char amplepolsR; // Es crea la variable amplepolsR
char amplepolsG; // Es crea la variable amplepolsG
char amplepolsB; // Es crea la variable amplepolsB
char compteR = 0; // Es crea la variable compteR i s'inicialitza a 0
char compteG = 0; // Es crea la variable compteG i s'inicialitza a 0
char compteB = 0; // Es crea la variable compteB i s'inicialitza a 0
char colorR, colorG, colorB; //Es creen les variables colorR, colorG i colorB
unsigned int FREQ, FREQ_R, FREQ_G, FREQ_B, FREQ_C;  // Es creen les diferents variables FREQ (sempre positives)
int Valor_R, Valor_G, Valor_B, Valor_C; // Es creen les variables enteres Valor_R, Valor_G, Valor_B i Valor_C
char DADA = 0; // Es crea la variable DADA i s'inicialitza a 0

unsigned int mesura_frequencia(void); // Es defineix la funció mesura_frequencia


//FUNCIÓ PRINCIPAL
void main (void)
{
    // Inicialització microcontrolador
    TRISA = 0B00100000;; // Selecció de pins port A com a sortides excepte el pin RA5 que serà entrada
    TRISC = 0B00000000;  // Selecció pins port B com a sortides
    TRISB = 0x00;  // Selecció pins port B com a sortides
    ANSEL = 0X00;  // Selecció sortides digitals
    ANSELH = 0X00; // Selecció sortides digitals

    // CONFIGURACIÓ TMR1 PER mesura_frequencia
    T1CON = 0B01000010;  // S'habiliten el Timer1 Gate Enable bit (TMR1GE) i el rellotge extern del Timer1 (TMR1CS)
    T1CONbits.TMR1ON = OFF; // Timer1 desconnectat
    CM2CON1bits.T1GSS = 0;  // Es desactiva el clock intern del TMR1 per a poder utilitzar RA4 com a entrada
```

```
//CONFIGURACIÓ TIMER0
INTCONbits.T0IE = 0;  // Es desactiven les interrupcions del Timer0
OPTION_REG = 0;       // Selecció rellotge intern pel Timer0


// CONFIGURACIÓ UART
TRISBbits.TRISB7 = 0; // Es configura el pin RB7 com a sortida. Serà la transmissió de l'UART: TX
TRISBbits.TRISB5 = 1; // Es configura el pin RB5 com a entrada. Serà la recepció de l'UART: RX
UART_Init(9600);      // Es crida a la funció d'inicialització de l'UART (fitxer uart.h)
INTCONbits.PEIE = 1;  // S'activen les interrupcions dels perifèrics
INTCONbits.GIE = 1;   // S'activen les interrupcions globals
PIE1bits.RCIE = 1;    // S'habilita bandera interrupció recepció
PIE1bits.TXIE = 0;    // S'habilita bandera interrupció transmissió


FREQ = 0; // S'inicialitza la variable FREQ a 0

// S'escull una escala de freqüència del 2% a la sortida del sensor
S0 = 0;
S1 = 1;


while (1)       // Executar mentre el contingut del parèntesis sigui 1 (sempre)
{

    if (DADA==0) // Si s'envia un 0 des de l'app
                 // Es llegeixen els valor captats pel sensor i s'envien els tres valors RGB al telèfon
    {

    amplepolsR = 0; //Fa que s'apagui el vermell
    amplepolsG = 0; //Fa que s'apagui el verd
    amplepolsB = 0; //Fa que s'apagui el blau

    // RED (vermell)
    S2 = OFF; // Configuració per tal que es mostri la frequencia mesurada segons el filtre vermell (RED)
    S3 = OFF; // Configuració per tal que es mostri la frequencia mesurada segons el filtre vermell (RED)
    FREQ_R = mesura_frequencia(); // La variable FREQ_R pren el valor del resultat de la funció mesura_frequencia
    Valor_R=(4*FREQ_R)*(255/1316.); // La variable Valor_R pren el valor de la freqüència en vermell
                                    // multiplicat per 255/1316 perquè el resultat sigui un nombre entre el 0 i el 255


    // GREEN (verd)
    S2 = ON; // Configuració per tal que es mostri la frequencia mesurada segons el filtre verd(GREEN)
    S3 = ON; // Configuració per tal que es mostri la frequencia mesurada segons el filtre verd(GREEN)
    FREQ_G = mesura_frequencia(); // La variable FREQ_G pren el valor del resultat de la funció mesura_frequencia
    Valor_G=(4*FREQ_G)*(255/1316.); // La variable Valor_G pren el valor de la freqüència en verd
                                    // multiplicat per 255/1316 perquè el resultat sigui un nombre entre el 0 i el 255

    // BLUE (blau)
    S2 = OFF; // Configuració per tal que es mostri la frequencia mesurada segons el filtre blau (BLUE)
    S3 = ON;  // Configuració per tal que es mostri la frequencia mesurada segons el filtre blau (BLUE)
    FREQ_B = mesura_frequencia(); // La variable FREQ_B pren el valor del resultat de la funció mesura_frequencia
    Valor_B=(4*FREQ_B)*(255/1316.); // La variable Valor_B pren el valor de la freqüència en blau
                                    // multiplicat per 255/1316 perquè el resultat sigui un nombre entre el 0 i el 255

    // CLEAR (blanc)
    S2 = ON;  // Configuració per tal que es mostri la frequencia mesurada segons el filtre blanc (CLEAR)
    S3 = OFF; // Configuració per tal que es mostri la frequencia mesurada segons el filtre blanc (CLEAR)
    FREQ_C = mesura_frequencia(); // La variable FREQ_C pren el valor del resultat de la funció mesura_frequencia
    Valor_C=(4*FREQ_C)*(255/1316.); // La variable Valor_C pren el valor de la freqüència en blanc
                                    // multiplicat per 255/1316 perquè el resultat sigui un nombre entre el 0 i el 255

    /* FREQ_R, FREQ_G, FREQ_B i FREQ_C es multipliquen per 4 per aconseguir el valor de la freqüència
        ja que només es tenia un quart d'ella (s'havia agafat el valor del Timer1 per un quart de segon)*/
```

```
        TXEN=1; // S'activa la transmissió
        UART_Write((char)Valor_R); // Es transforma Valor_R en una variable de 8 bits i s'escriu a l'UART
        __delay_ms(2); // S'esperen 2ms
        UART_Write((char)Valor_G); // Es transforma Valor_G en una variable de 8 bits i s'escriu a l'UART
        __delay_ms(2); // S'esperen 2ms
        UART_Write((char)Valor_B); // Es transforma Valor_B en una variable de 8 bits i s'escriu a l'UART
        __delay_ms(2); // S'esperen 2ms
        TXEN=0; // S'atura la transmissió perquè només cal enviar les dades un cop
        DADA=2; // La variable DADA passa a valdre 2.
                // Així, no torna a entrar en aquest if fins que no es torni a enviar un 0 des del mòbil

        }//Final de l'if

      } // Final del bucle while

} //Final de la funció main


//FUNCIÓ QUE MESURA LA FREQÜÈNCIA R,G,B I CLEAR
unsigned int mesura_frequencia(void)  //Aquesta funció capta la informació del sensor
    {
    T1CONbits.TMR1ON = 1; // S'activa el Timer1
    TMR1 = 0;             // Es posa el Timer1 a 0
    __delay_ms(250);      // S'espera un quart de segon
    FREQ = TMR1;          // Es llegeix el valor del Timer1 i es guarda a la variable FREQ
    T1CONbits.TMR1ON = 0; // Es desactiva el Timer1
    return(FREQ);         // Retorna el valor de la variable FREQ (Timer1)
    }



//SERVEI INTERRUPCIONS
void interrupt Interrupcion() //S'atenen les interrupcions.S'executa quan hi ha una paraula a RCREG Register
{
    if (PIR1bits.RCIF) //Sempre que la bandera de recepció en el buffer d'entrada de l'UART estigui activada
    {
    DADA = UART_Read(); // Es llegeix el primer valor enviat des del mòbil.
                        // Si dada és 0, anirà a executar l'if de la funció main. Sinó, s'executarà l'if de les interrupcions
        if (DADA == 1) // Si s'envia un 1 des de l'app, es llegeixen les tres variables RGB enviades a continuació pel mòbil
        {
        colorR = UART_Read();  // Llegeix la dada en RX de l'UART i s'associa a la variable amplepolsR
                               //(serà un número entre el 0 i el 255)
        colorG = UART_Read();  // Llegeix la dada en RX de l'UART i s'associa a la variable amplepolsG
                               //(serà un número entre el 0 i el 255)
        colorB = UART_Read();  // Llegeix la dada en RX de l'UART i s'associa a la variable amplepolsB
                               //(serà un número entre el 0 i el 255)
        amplepolsR = (colorR*100)/255; // Es passa el valor de colorR entre un número del 0 al 100
        amplepolsG = (colorG*100)/255; // Es passa el valor de colorG entre un número del 0 al 100
        amplepolsB = (colorB*100)/255; // Es passa el valor de colorB entre un número del 0 al 100
        DADA = 2;// La variable DADA passa a valdre 2.
                 // Així, no torna a entrar en aquest if fins que no es torni a enviar un 1
        TMR0 = 215; // El Timer0 s'inicialitza a 215, un valor prou alt perquè no es vegin pampallugues al LED RGB
        INTCONbits.T0IE = 1; // S'activen les interrupcions del Timer0
        }

    PIR1bits.RCIF = 0;  //Es desactiva la bandera de recepció en el buffer d'entrada de l'UART
    }

    if (INTCONbits.T0IF)  // Sempre que la bandera de recepció del buffer d'entrada del Timer0 estigui activada
    {
        compteR++;  // S'incrementa una unitat cada cop que entra en el bucle
        compteG++;  // S'incrementa una unitat cada cop que entra en el bucle
        compteB++;  // S'incrementa una unitat cada cop que entra en el bucle


    if (compteR==amplepolsR + 1) //Quan l'incrementador compteR sigui igual a amplepolsR + 1
    {
        LEDR = 0; // La variable LEDR pren valor 0 (LED vermell s'apaga)
    }
    if (compteG==amplepolsG + 1) // Quan l'incrementador compteG sigui igual a amplepolsG + 1
    {
        LEDG = 0; // La variable LEDG pren valor 0 (LED verd s'apaga)
    }
    if (compteB==amplepolsB + 1) // Quan l'incrementador compteB sigui igual a amplepolsB + 1
    {
        LEDB = 0; //La variable LEDB pren valor 0 (LED blau s'apaga)
    }
```

```
      if (compteR == 100) // Quan l'incrementador arriba a 100
      {
          LEDR = 1; // La compoment vermella del LED RGB s'encén
          LEDG = 1; // La compoment verda del LED RGB s'encén
          LEDB = 1; // La compoment blava del LED RGB s'encén
          compteR = 0; // La variable compteR pren valor 0
          compteG = 0; // La variable compteG pren valor 0
          compteB = 0; // La variable compteB pren valor 0
      }
      TMR0 = 215; // El Timer0 es posa a 215, un valor prou alt perquè no es vegin pampallugues al LED RGB
      INTCONbits.T0IF = 0;  // Es desactiva la bandera d'interrupció del Timer0
  }
//Final servei interrupcions
```

1.1. uart.h

```
/*
Treball Final de Grau: Disseny i construcció d'aparells electrònics per identificar el color d'objectes
Autores: Cristina Rodríguez Murciano i Judit Sanahuja Roig
Directora: Rosa Rodríguez Montañés
Convocatòria: Gener 2017
*/

//INICIALITZACIÓ UART
UART_Init(int baudrate)
{
    TXSTA=0x24; // S'activa la transmissió, se selecciona mode asíncron i se selecciona alta velocitat
    SPBRG=25;   // Se selecciona velocitat de transmissió
    BAUDCTLbits.BRG16=0; // Es desactiva timer 16 bits
    RCSTA=0xB0; // Es configuren els pins RX i TX com a ports en sèrie i s'activa mode recepció continua
}

//FUNCIÓ ESCRIPTURA
void UART_Write(char data)
{
 while(!TRMT); //Quan TRMT sigui 0, la transmissió està en curs
 TXREG = data; //Es guarda a TXREG la dada
}

//FUNCIÓ LECTURA
char UART_Read()
{
  while(!RCIF); // Mentre RCIF sigui 0 la bandera de recepció està desactivada i va llegint.
  return RCREG; //Retorna el valor que llegeix
}
```

2. Circuit_2.c

```
/*
Treball Final de Grau: Disseny i construcció d'aparells electrònics per identificar el color d'objectes
Autores: Cristina Rodríguez Murciano i Judit Sanahuja Roig
Directora: Rosa Rodríguez Montañés
Convocatòria: Gener 2017
*/

//PROGRAMACIÓ DEL MICROCONTROLADOR PER AL CIRCUIT 2: CONNEXIÓ AMB DISPOSITIU LCD

// CONFIGURACIÓ GENERAL INICIAL
#include <pic16f690.h> // Inclou la llibreria pic16f690.h, la del microcontrolador
#include <xc.h> // Inclou la llibreria xc.h, la del compilador
#define _XTAL_FREQ 4000000 //Defineix la freqüència de funcionament del microcontrolador, 4MHz
#define ON  1 // Es defineix que ON és 1
#define OFF 0 // Es defineix que OFF és 0

// CONFIGURACIÓ MICROCONTROLADOR
#pragma config FOSC = INTRCIO // Es selecciona l'oscil·lador intern mitjançant els Oscillator Selection bits
#pragma config WDTE = OFF // El Watchdog Timer Enable bit està desactivat
#pragma config PWRTE = OFF // El Power-up Timer Enable bit està activat
#pragma config MCLRE = OFF // El MCLR Pin Function Select bit és una entrada digital
#pragma config CP = OFF // El Code Protection bit està activat
#pragma config CPD = OFF // El Data Code Protection bit està activat
#pragma config BOREN = OFF // Els Brown-Out Reset bits estan desactivats
#pragma config IESO = OFF // El Internal Extrnal Switchover bit està desactivat
#pragma config FCMEN = OFF // El Fail-Safe Clock Monitor Enabled bit està desactivat

// CONFIGURACIÓ PANTALLA LCD
#define RS RC1 // El RS, Register Select, de la LCD està connectat al pin RC1 del microcontrolador
               // quan s'escrigui RS es referirà a aquest pin
#define E RC2 // El E, Enable, de la LCD està connectat al pin RC2, quan s'escrigui E es referirà a aquest pin
#define RW RC3 // El RW, Read/Write, de la LCD està connectat al pin RC3, quan s'escrigui RW es referirà a aquest pin
#define D4 RC4 // El D4, Data4, de la LCD està connectat al pin RC4, quan s'escrigui D4 es referirà a aquest pin
#define D5 RC5 // El D5, Data5, de la LCD està connectat al pin RC5, quan s'escrigui D5 es referirà a aquest pin
#define D6 RC6 // El D6, Data6, de la LCD està connectat al pin RC6, quan s'escrigui D6 es referirà a aquest pin
#define D7 RC7 // El D7, Data7, de la LCD està connectat al pin RC7, quan s'escrigui D7 es referirà a aquest pin
#include "lcd_hd44780_pic16.h" // Inclou la llibreria lcd_hd44780_pic16.h, la del model de LCD utilitzat


// CONFIGURACIÓ LED RGB
#define LEDR RB4 // El LEDR está connectat al pin RB4 del microcontrolador
                 // quan s'escrigui LEDR es referirà a aquest pin
#define LEDG RB5 // El LEDG está connectat al pin RB5, quan s'escrigui LEDG es referirà a aquest pin
#define LEDB RB6 // El LEDB está connectat al pin RB6, quan s'escrigui LEDB es referirà a aquest pin

// CONFIGURACIÓ SENSOR DE COLOR
#define S0 RA0 // El pin S0 del sensor està connectat al pin RA0 del microcontrolador
               // quan s'escrigui RA0 es referirà a aquest pin
#define S1 RA1 // El pin S1 del sensor està connectat al pin RA1 del microcontrolador
               // quan s'escrigui RA1 es referirà a aquest pin
#define S2 RA2 // El pin S2 del sensor està connectat al pin RA2 del microcontrolador
               // quan s'escrigui RA2 es referirà a aquest pin
#define S3 RA4 // El pin S3 del sensor està connectat al pin RA4 del microcontrolador
               // quan s'escrigui RA4 es referirà a aquest pin
// OUT al RA5 que és el senyal del qual es vol mesurar la seva freqüència (rellotge extern TMR1 al pin RA5)

// CONFIGURACIÓ DEL POLSADOR
#define boto RB7 // El polsador està connectat al pin RB7 del microcontrolador, quan s'escrigui boto es referirà a aquest pin
#define premut 0 // Es defineix que premut és 0
#define no_premut 1 // Es defineix que no_premut és 1

// DEFINICIÓ I INICIALITZACIÓ DE VARIABLES
char amplepolsR; // Es crea la variable amplepolsR
char amplepolsG; // Es crea la variable amplepolsG
char amplepolsB; // Es crea la variable amplepolsB
char compteR = 0; // Es crea la variable compteR i s'inicialitza a 0
char compteG = 0; // Es crea la variable compteG i s'inicialitza a 0
char compteB = 0; // Es crea la variable compteB i s'inicialitza a 0
bit estat = 0 ; // Es crea la variable binaria estat i s'inicialitza a 0
unsigned int FREQ, FREQ_R, FREQ_G, FREQ_B, FREQ_C; // Es creen les variables enteres i positives
                                                   // anomenades FREQ, FREQ_R, FREQ_G, FREQ_B i FREQ_C
int Valor_R, Valor_G, Valor_B, Valor_C; // Es creen les variables enteres Valor_R, Valor_G, Valor_B i Valor_C
```

```
unsigned int mesura_frequencia(void); // Es crea la funció mesura_frequencia
                                      // que és la que servirà per mesurar la freqüència dels colors

// FUNCIÓ PRINCIPAL
void main (void)
{
    // INICIALITZACIÓ DEL MICROCONTROLADOR
    TRISA = 0B00100000; // Configurar el PORTA com a sortida, excepte el pin RA5
    TRISC = 0B00000000; // Configurar el PORTC com a sortida
    TRISB = 0x80; // Configurar el pin RB7 com a entrada, els altres del PORTB com a sortida
    ANSEL = 0X00; // Configuració de mode digital
    ANSELH = 0X00; // Configuració de mode digital
    OPTION_REG = 0B00000000; // S'activa el Pull-up general del PORT AB
                             // per definir que el polsador, quan no està premut, està a 1 i, aquest, està connectat al pin RB7
    WPUB = 0B10000000; // S'activa el Weak Pull-up del pin RB7
                       // necessari pel funcionament del polsador, que es troba connectat en aquest pin

    // CONFIGURACIÓ TMR1 PER MESURAR LA FREQÜÈNCIA
    T1CON = 0B01000010; // S'habiliten el Timer1 Gate Enable bit, TMR1GE, i el rellotge extern del Timer1, TMR1CS.
                        // A més, aquest pin és el RA5, que és on va connectada la sortida del sensor, la freqüència
    T1CONbits.TMR1ON = OFF; // Inicialment, el Timer1 està desconnectat
    CM2CON1bits.T1GSS = 0; // El Timer1 Gate Source Select, T1GSS, es posa a 0 per a poder utilitzar el pin RA4 com a entrada.
                           // Si es posés a 1, aquest pin seria el pin de clock del Timer1

    // CONFIGURACIÓ DE LES INTERRUPCIONS
    INTCONbits.PEIE = 1; // S'habiliten les interrupcions dels perifèrics
    INTCONbits.GIE  = 1;  // S'habiliten les interrupcions globals
    INTCONbits.T0IE = 0; // Es desactiven les interrupcions del Timer0

    // FUNCIÓ PER INICIALITZAR EL MÒDUL LCD
    LCDInit(LS_NONE); // S'inicialitza la LCD
    __delay_ms(100); // S'afegeixen delays perquè si es va molt ràpid no funciona, en aquest cas, s'espera 100 milisegons


     // FUNCIÓ CLEAR THE DISPLAY
     LCDClear(); // La pantalla del dispositiu LCD es posa en blanc
     __delay_ms(100); // S'espera 100 milisegons

     // FUNCIÓ WRITE A STRING
     LCDGotoXY(2,0); // S'escriu a la pantalla a partir de la posició (2,0), tercera posició de la fila de dalt
     __delay_ms(100); // S'espera 100 milisegons
     LCDWriteString("Colors en RGB"); // Per la pantalla es mostra el text Colors en RGB
     __delay_ms(100); // S'espera 100 milisegons
     LCDGotoXY(7,1); // S'escriu a la pantalla a partir de la posició (7,1), vuitena posició de la fila de sota
     __delay_ms(100); // S'espera 100 milisegons
     LCDWriteString("TFG"); // Per la pantalla es mostra el text TFG
     __delay_ms(3000); // S'espera 3 segons

     FREQ = 0; // S'inicialitza la variable FREQ a 0

     // Els pins S0 i S1 del sensor de color es connecten
     // de tal manera que l'escala de la seva freqüència de sortida sigui del 2%
     S0 = 0;
     S1 = 1;

     LCDClear(); // Es neteja la pantalla, posant-la en blanc

     LEDR = 0;
     LEDG = 0;
     LEDB = 0;
```

```
while (1) // Mentre el contingut del parèntesi sigui 1
{
    if (boto==no_premut && estat== 0) // Quan la variable boto sigui 1 i estat sigui 0, hi haurà el següent procediment
    {
        // RED - VERMELL
        S2 = OFF; // Configuració dels pins per tal que es mostri la freqüència mesurada segons el filtre vermell
        S3 = OFF; // Configuració dels pins per tal que es mostri la freqüència mesurada segons el filtre vermell
        FREQ_R = mesura_frequencia(); // La variable FREQ_R pren el valor del resultat de la funció mesura_frequencia
        LCDGotoXY(0,0); // S'escriu a la pantalla a partir de la posició (0,0), primera posició de la fila de dalt
        __delay_ms(100); // S'espera 100 milisegons
        LCDWriteString("R:"); // Mostrar el text R:
        Valor_R=(4*FREQ_R)*(255/1316.); // La variable Valor_R pren el valor de la freqüència en vermell
                                        // multiplicat per 255/1316 perquè el resultat sigui un nombre entre el 0 i el 255
        // FREQ_R es multiplica per 4 per aconseguir el valor de la freqüència,
        // ja que només es tenia un quart d'ella (s'havia agafat el valor del Timer1 per un quart de segon)
        LCDWriteInt(Valor_R,5); // Mostra el valor donat per la pantalla amb una llargada de 5 xifres

        // GREEN - VERD
        S2 = ON; // Configuració dels pins per tal que es mostri la freqüència mesurada segons el filtre verd
        S3 = ON; // Configuració dels pins per tal que es mostri la freqüència mesurada segons el filtre verd
        FREQ_G = mesura_frequencia(); // La variable FREQ_G pren el valor del resultat de la funció mesura_frequencia
        LCDGotoXY(8,0); // S'escriu a la pantalla a partir de la posició (8,0), novena posició de la fila de dalt
        __delay_ms(100); // S'espera 100 milisegons
        LCDWriteString("G:"); // Mostrar el text G:
        Valor_G=(4*FREQ_G)*(255/1316.); // La variable Valor_G pren el valor de la freqüència en verd
                                        // multiplicat per 255/1316 perquè el resultat sigui un nombre entre el 0 i el 255
        // FREQ_G es multiplica per 4 per aconseguir el valor de la freqüència,
        // ja que només es tenia un quart d'ella (s'havia agafat el valor del Timer1 per un quart de segon)
        LCDWriteInt(Valor_G,5); // Mostra el valor donat per la pantalla amb una llargada de 5 xifres


        // BLUE - BLAU
        S2 = OFF; // Configuració dels pins per tal que es mostri la freqüència mesurada segons el filtre blau
        S3 = ON; // Configuració dels pins per tal que es mostri la freqüència mesurada segons el filtre blau
        FREQ_B = mesura_frequencia(); // La variable FREQ_B pren el valor del resultat de la funció mesura_frequencia
        LCDGotoXY(0,1); // S'escriu a la pantalla a partir de la posició (0,1), primera posició de la fila de sota
        __delay_ms(100); // S'espera 100 milisegons
        LCDWriteString("B:"); // Mostrar el text B:
        Valor_B=(4*FREQ_B)*(255/1316.); // La variable Valor_B pren el valor de la freqüència en blau
                                        // multiplicat per 255/1316 perquè el resultat sigui un nombre entre el 0 i el 255
        // FREQ_B es multiplica per 4 per aconseguir el valor de la freqüència,
        // ja que només es tenia un quart d'ella (s'havia agafat el valor del Timer1 per un quart de segon)
        LCDWriteInt(Valor_B,5); // Mostra el valor donat per la pantalla amb una llargada de 5 xifres

        // CLEAR - BLANC
        S2 = ON; // Configuració dels pins per tal que es mostri la freqüència mesurada segons el filtre blanc
        S3 = OFF; // Configuració dels pins per tal que es mostri la freqüència mesurada segons el filtre blanc
        FREQ_C = mesura_frequencia(); // La variable FREQ_C pren el valor del resultat de la funció mesura_frequencia
        LCDGotoXY(8,1); // S'escriu a la pantalla a partir de la posició (8,1), novena posició de la fila de sota
        __delay_ms(100); // S'espera 100 milisegons
        LCDWriteString("C:"); // Mostrar el text C:
        Valor_C=(4*FREQ_C)*(255/1316.); // La variable Valor_C pren el valor de la freqüència en blanc
                                        // multiplicat per 255/1316 perquè el resultat sigui un nombre entre el 0 i el 255
        // FREQ_C es multiplica per 4 per aconseguir el valor de la freqüència,
        // ja que només es tenia un quart d'ella (s'havia agafat el valor del Timer1 per un quart de segon)
        LCDWriteInt(Valor_C,5); // Mostra el valor donat per la pantalla amb una llargada de 5 xifres

        amplepolsR = (FREQ_R*100)/(FREQ_R+FREQ_G+FREQ_B); // Es passa el valor de FREQ_R entre un numero del 0 al 100
        amplepolsG = (FREQ_G*100)/(FREQ_R+FREQ_G+FREQ_B); // Es passa el valor de FREQ_G entre un numero del 0 al 100
        amplepolsB = (FREQ_B*100)/(FREQ_R+FREQ_G+FREQ_B); // Es passa el valor de FREQ_B entre un numero del 0 al 100

    } // Final de la condició de l'if

    //INICI DEL PROCEDIMENT PER ENCENDRE EL LED RGB
    if (boto==no_premut && estat==1) // Quan la variable boto sigui igual a no_premut i estat sigui 1
    {
    // No es realitza cap càlcul per a què es quedi amb l'última lectura del sensor de color
    } // Final de la condició de l'if

    // Per controlar quan s'encén el LED RGB
    if (boto==premut) // Quan la variable boto sigui igual a premut
    {
        INTCONbits.T0IE = OFF; // Es desactiven les interrupcions del Timer0
        estat=!estat; // L'estat canviarà el seu valor, passarà de 0 a 1 o, en el cas en què fos 1, passaria a ser 0
        LEDR = 0; // El LED vermell del RGB s'apaga
        LEDG = 0; // El LED verd del RGB s'apaga
        LEDB = 0; // El LED blau del RGB s'apaga
        while (boto==premut); // El programa espera en aquest punt mentre no s'allibera el polsador
        if (estat == 1)
        {
            INTCONbits.T0IE = ON; // S'abiliten les interrupcions del Timer0
        } // Final de la condició de l'if
        __delay_ms(100); // S'espera 100 milisegons. S'afegeix el delay per evitar rebots del polsador
    } // Final de la condició de l'if


  } // Final del bucle while
} // Final de la funció principal
```

```
        //INICI DEL PROCEDIMENT PER ENCENDRE EL LED RGB
        if (boto==no_premut && estat==1) // Quan la variable boto sigui igual a no_premut i estat sigui 1
        {
        // No es realitza cap càlcul per a què es quedi amb l'última lectura del sensor de color
        } // Final de la condició de l'if

        // Per controlar quan s'encén el LED RGB
        if (boto==premut) // Quan la variable boto sigui igual a premut
        {
            INTCONbits.T0IE = OFF; // Es desactiven les interrupcions del Timer0
            estat=!estat; // L'estat canviarà el seu valor, passarà de 0 a 1 o, en el cas en què fos 1, passaria a ser 0
            LEDR = 0; // El LED vermell del RGB s'apaga
            LEDG = 0; // El LED verd del RGB s'apaga
            LEDB = 0; // El LED blau del RGB s'apaga
            while (boto==premut); // El programa espera en aquest punt mentre no s'allibera el polsador
            if (estat == 1)
            {
                INTCONbits.T0IE = ON; // S'abiliten les interrupcions del Timer0
            } // Final de la condició de l'if
            __delay_ms(100); // S'espera 100 milisegons. S'afegeix el delay per evitar rebots del polsador
        } // Final de la condició de l'if


      } // Final del bucle while
} // Final de la funció principal

// FUNCIÓ QUE MESURA LA FREQÜÈNCIA R, G, B I CLEAR
unsigned int mesura_frequencia(void) //Funció mesura_frequencia, serveix per captar la informació del sensor
    {
    T1CONbits.TMR1ON = 1; // S'activa el Timer1
    TMR1 = 0; // El Timer1 es posa a 0
    __delay_ms(250); // S'espera un quart de segon
    FREQ = TMR1; // Es llegeix el valor del Timer1 i aquest valor el pren la variable FREQ
    T1CONbits.TMR1ON = 0; // Es desactiva el Timer1
    return(FREQ); // Retorna el valor de la variable FREQ, que és el valor del Timer1
    }


// SERVEI D'INTERRUPCIONS PER PODER ENCENDRE EL LED RGB SEGONS EL COLOR CAPTAT PEL SENSOR
void interrupt Interrupcion()   // Funció que atén les interrupcions
{
    if (INTCONbits.T0IF) // Sempre que la bandera d'interrupció del Timer0 estigui activada
    {
        compteR++; // La variable compteR augmenta una unitat cada vegada que hi ha una interrupció
        compteG++; // La variable compteG augmenta una unitat cada vegada que hi ha una interrupció
        compteB++; // La variable compteB augmenta una unitat cada vegada que hi ha una interrupció
        if (compteR == amplepolsR + 1) // Quan la variable compteR és igual a amplepolsR més 1
                                       // (si no se li suma 1 a amplepolsR, aquesta mai podria ser 0)
        {
            LEDR = 0; // La component vermella del LED RGB s'apaga
        }
        if (compteG == amplepolsG + 1) // Quan la variable compteG és igual a amplepolsG més 1
                                       // (si no se li suma 1 a amplepolG, aquesta mai podria ser 0)
        {
            LEDG = 0; // La component verda del LED RGB s'apaga
        }
        if (compteB == amplepolsB + 1) // Quan la variable compteB sigui igual a amplepolsB més 1
                                       // (si no se li suma 1 a amplepolsB, aquesta mai podria ser 0)
        {
            LEDB = 0; // La component blava del LED RGB s'apaga
        }
        if (compteR == 100) // Quan el compteR val 100 es segueix amb el procediment següent
                            // (les variables compteR, compteG i compteB són iguals en tot moment)
        {
            compteR = 0; // La variable compteR torna a 0
            compteG = 0; // La variable compteG torna a 0
            compteB = 0; // La variable compteB torna a 0
            LEDR = 1; // La compoment vermella del LED RGB s'encén
            LEDG = 1; // La compoment verda del LED RGB s'encén
            LEDB = 1; // La compoment blava del LED RGB s'encén
        }
        TMR0 = 215; // El Timer0 es posa a 215, un valor prou alt perquè no es vegin pampallugues al LED RGB
        INTCONbits.T0IF = 0;  // Es desactiva la bandera d'interrupció del Timer0
    }
}
```

2.1. lcd_hd44780_pic16.c

```
/*******************************************************************************
*******************************************************************************/

#include <stdint.h>

#include "lcd_hd44780_pic16.h"
#include "myutils.h"
#include "custom_char.h"
#define _XTAL_FREQ 4000000

#define LCD_DATA_PORT   PORT(LCD_DATA)
#define LCD_DATA_TRIS   TRIS(LCD_DATA)

#define LCD_E           PORTBIT(LCD_E_PORT,LCD_E_POS)
#define LCD_E_TRIS      TRISBIT(LCD_E_PORT,LCD_E_POS)

#define LCD_RS          PORTBIT(LCD_RS_PORT,LCD_RS_POS)
#define LCD_RS_TRIS     TRISBIT(LCD_RS_PORT,LCD_RS_POS)

#define LCD_RW          PORTBIT(LCD_RW_PORT,LCD_RW_POS)
#define LCD_RW_TRIS     TRISBIT(LCD_RW_PORT,LCD_RW_POS)


#define SET_E() (LCD_E=1)
#define SET_RS() (LCD_RS=1)
#define SET_RW() (LCD_RW=1)

#define CLEAR_E() (LCD_E=0)
#define CLEAR_RS() (LCD_RS=0)
#define CLEAR_RW() (LCD_RW=0)


void LCDByte(uint8_t c,uint8_t isdata)
{
//Sends a byte to the LCD in 4bit mode
//cmd=0 for data
//cmd=1 for command


//NOTE: THIS FUNCTION RETURS ONLY WHEN LCD HAS PROCESSED THE COMMAND

uint8_t hn,ln;          //Nibbles
uint8_t temp;

hn=c>>4;
ln=(c & 0x0F);

if(isdata==0)
    CLEAR_RS();
else
    SET_RS();

__delay_us(0.5);        //tAS

SET_E();

//Send high nibble

temp=(LCD_DATA_PORT & (~(0X0F<<LCD_DATA_POS)))|((hn<<LCD_DATA_POS));
LCD_DATA_PORT=temp;

__delay_us(1);          //tEH

//Now data lines are stable pull E low for transmission

CLEAR_E();

__delay_us(1);

//Send the lower nibble
SET_E();

temp=(LCD_DATA_PORT & (~(0X0F<<LCD_DATA_POS)))|((ln<<LCD_DATA_POS));
```

```
LCD_DATA_PORT=temp;

__delay_us(1);          //tEH

//SEND

CLEAR_E();

__delay_us(1);          //tEL

LCDBusyLoop();
}

void LCDBusyLoop()
{
    //This function waits till lcd is BUSY

    uint8_t busy,status=0x00,temp;

    //Change Port to input type because we are reading data
    LCD_DATA_TRIS|=(0x0f<<LCD_DATA_POS);

    //change LCD mode
    SET_RW();       //Read mode
    CLEAR_RS();     //Read status

    //Let the RW/RS lines stabilize

    __delay_us(0.5);        //tAS



    do
    {

        SET_E();

        //Wait tDA for data to become available
        __delay_us(0.5);

        status=(LCD_DATA_PORT>>LCD_DATA_POS);
        status=status<<4;

        __delay_us(0.5);

        //Pull E low
        CLEAR_E();
        __delay_us(1);  //tEL

        SET_E();
        __delay_us(0.5);

        temp=(LCD_DATA_PORT>>LCD_DATA_POS);
        temp&=0x0F;

        status=status|temp;

        busy=status & 0b10000000;

        __delay_us(0.5);

                CLEAR_E();
        __delay_us(1);  //tEL
    }while(busy);

    CLEAR_RW();     //write mode


        //Change Port to output
    LCD_DATA_TRIS&=(~(0x0F<<LCD_DATA_POS));

}
```

```
void LCDInit(uint8_t style)
{
    /*********************************************************************

    This function Initializes the lcd module
    must be called before calling lcd related functions

    Arguments:
    style = LS_BLINK,LS_ULINE(can be "OR"ed for combination)
    LS_BLINK : The cursor is blinking type
    LS_ULINE : Cursor is "underline" type else "block" type
        LS_NONE : No visible cursor

    *********************************************************************/

    //After power on Wait for LCD to Initialize
    __delay_ms(30);

    //Set IO Ports
    LCD_DATA_TRIS&=(~(0x0F<<LCD_DATA_POS)); //Output

        LCD_E_TRIS=0;   //Output
        LCD_RS_TRIS=0;  //Output
        LCD_RW_TRIS=0;  //Output

    LCD_DATA_PORT&=(~(0x0F<<LCD_DATA_POS));//Clear data port

        CLEAR_E();
    CLEAR_RW();
    CLEAR_RS();


    //Set 4-bit mode
    __delay_us(0.5);    //tAS

    SET_E();
    LCD_DATA_PORT|=((0b00000010)<<LCD_DATA_POS); //[B] To transfer 0b00100000 i was using LCD_DATA_PORT|=0b00100000
    __delay_us(1);
    CLEAR_E();
    __delay_us(1);

    //Wait for LCD to execute the Functionset Command
    LCDBusyLoop();                                  //[B] Forgot this delay

    //Now the LCD is in 4-bit mode


    LCDCmd(0b00101000);             //function set 4-bit,2 line 5x7 dot format
        LCDCmd(0b00001100|style);   //Display On

    /* Custom Char */
        LCDCmd(0b01000000);

    uint8_t __i;
    for(__i=0;__i<sizeof(__cgram);__i++)
        LCDData(__cgram[__i]);


}



void LCDWriteString(const char *msg)
{
    /*********************************************************************

    This function Writes a given string to lcd at the current cursor
    location.

    Arguments:
    msg: a null terminated C style string to print

    Their are 8 custom char in the LCD they can be defined using
    "LCD Custom Character Builder" PC Software.
```

```
  You can print custom character using the % symbol. For example
  to print custom char number 0 (which is a degree symbol), you
  need to write

  LCDWriteString("Temp is 30%0C");
                              ^^
                               |----> %0 will be replaced by
                                      custom char 0.

  So it will be printed like.

      Temp is 30°C

  In the same way you can insert any symbols numbered 0-7


  ************************************************************/


 while(*msg!='\0')
 {
    //Custom Char Support
    if(*msg=='%')
    {
        msg++;
        int8_t cc=*msg-'0';

        if(cc>=0 && cc<=7)
        {
            LCDData(cc);
        }
        else
        {
            LCDData('%');
            LCDData(*msg);
        }
    }
    else
    {
        LCDData(*msg);
    }
    msg++;
 }
}

void LCDWriteInt(int val,char field_length)
{
    /*************************************************************
    This function writes a integer type value to LCD module

    Arguments:
    1)int val    : Value to print

    2)unsigned int field_length :total length of field in which the value is printed
    must be between 1-5 if it is -1 the field length is no of digits in the val

    *************************************************************/


    char str[5]={0,0,0,0,0};
    int i=4,j=0;

        //Handle negative integers
        if(val<0)
        {
            LCDData('-');    //Write Negative sign
            val=val*-1;      //convert to positive
        }

    while(val)
    {
            str[i]=val%10;
            val=val/10;
            i--;
    }
    if(field_length==-1)
        while(str[j]==0) j++;
    else
        j=5-field_length;


    for(i=j;i<5;i++)
    {
    LCDData(48+str[i]);
    }
}
```

```
/************************************************************************

Position the cursor to specific part of the screen

************************************************************************/
void LCDGotoXY(uint8_t x,uint8_t y)
{
    if(x>=20) return;

    switch(y)
    {
        case 0:
            break;
        case 1:
            x|=0b01000000;
            break;
    }

    x|=0b10000000;
    LCDCmd(x);
}
```

2.2. lcd_hd44780_pic16.h

```
#include <xc.h>
#include <stdint.h>

/************************************************
    LCD CONNECTIONS
************************************************/

//LCD Data Port
//Port RC4-RC7 are connected to D4-D7
#define LCD_DATA        C   //Port RC0-RC3 are connected to D4-D7
#define LCD_DATA_POS    4

//Register Select (RS)
//RS is connected to Port D bit 4
#define LCD_RS_PORT     C
#define LCD_RS_POS      1

//Read/Write (RW)
//RW is connected to Port D bit 5
#define LCD_RW_PORT     C
#define LCD_RW_POS      3

//Enable signal (E)
//E is connected to Port D bit 6
#define LCD_E_PORT      C
#define LCD_E_POS       2


/*********************************************
LCD Type Selection
Uncomment Just one of them
*********************************************/

//#define LCD_TYPE_202  //For 20 Chars by 2 lines

//#define LCD_TYPE_204  //For 20 Chars by 4 lines

#define LCD_TYPE_162    //For 16 Chars by 2 lines

//#define LCD_TYPE_164  //For 16 Chars by 4 lines


/*********************************************
Possible Configurations
*********************************************/

#define LS_BLINK 0B00000001
#define LS_ULINE 0B00000010
#define LS_NONE  0B00000000


/************************************************
          F U N C T I O N S
************************************************/

void LCDInit(char style);
void LCDWriteString(const char *msg);
void LCDWriteInt(int val,char field_length);
void LCDGotoXY(char x,char y);
```

```
//Low level
void LCDByte(char,char);
#define LCDCmd(c) (LCDByte(c,0))
#define LCDData(d) (LCDByte(d,1))

void LCDBusyLoop();

/************************************************
          F U N C T I O N S     E N D
************************************************/


/************************************************
    M A C R O S
************************************************/
#define LCDClear() LCDCmd(0b00000001)
#define LCDHome()  LCDCmd(0b00000010)

#define LCDWriteStringXY(x,y,msg) {\
 LCDGotoXY(x,y);\
 LCDWriteString(msg);\
}

#define LCDWriteIntXY(x,y,val,fl) {\
 LCDGotoXY(x,y);\
 LCDWriteInt(val,fl);\
}
/************************************************/
```