UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

Grau en Enginyeria Electrònica Industrial i Automàtica

DESENVOLUPAMENT D'UNA UNITAT D'ADQUISICIÓ I MONITORITZACIÓ DE DADES PER A UN VEHICLE

Annexos

Autor: Irina Selin Lorenzo
Director: Manuel Andrés Manzanares Brotons
Convocatòria: Juny 2020

Índex

1. Codi del programa

```
#include <18F4550.h> //pic
#device PASS_STRINGS = IN_RAM
#device adc=10
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

//#fuses HS //hs per cristall de mes de 4MHz
#fuses XT
#fuses MCLR //habilita reset extern
#fuses NOWDT //no whatcdog
#fuses NOVREGEN //deshabilita sortida de 3.3V
#fuses CPUDIV1 //prescales a 48MHz
#fuses NOPROTECT //memoria no protegida contra lectures
#fuses PUT //habilita timer
#fuses NOLVP //PIN-B5 entrada/sortida
#fuses PLL1 //no prescaler
#fuses NOBROWNOUT //no reset a baix voltatge

#use delay (clock=20M) //T = 500us
#use RS232(UART1,BAUD=9600, BITS=8, PARITY=N, XMIT=PIN_C6, RCV=PIN_C7, ERRORS) //
bluetooth HC-06
#use standard_io(A)
#use standard_io(B)
#use standard_io(C)
#use standard_io(D)
#use standard_io(E)

//es posen mes abaix per q sino no funcionen
#include <lcd.c> //LCD
#include <BME280_Lib.c>//temperatura, humitat, pressio
#include <ds1307.c> //RTC
#include <24256.c> //memoria

//VARIABLES GLOBALS
//rtc
BYTE sec;
BYTE min;
BYTE hrs;
BYTE dia;
BYTE mes;
BYTE any;
BYTE diaset;

int32 time_cor; //timer de 15s
```

```
int32 time_vel; //timer 2s
int16 t_mem; //timer 1s
int32 t_sens; //timer 3s

int32 temps_ini; //moment d'inici
int32 temps_fi; //moment fi
int hrs_rec; //hores de recorregut
int min_rec; //minuts recorregut
int sec_rec; //segons recorregut

char con; //conectivitat amb bluetooth
char start; //bluetooth start
char tot; //bluetooth valors totals
char fi; //bluetooth fi

int bstart; //boto start
int boto; //boto seguent

int lum; //lluminositat
int fr_cor; //frecuencia cardiaca
int fr_cor_mj; //fr. cardiaca mitja
int c_cor;
int32 dist; //distancia
int16 vel; //velocitat
int c_vel; //contador pulsos velocitat
int16 vel_mj; //velocitat mitja
int16 uva; //valor rajos uva
int16 co2; //CO2
int pedal; //cadencia de pedaleig
int pedal_mj; //cadencia de pedaleig mitja
int c_peda;
int vibr;
signed int16 temperature; //temperatura
int16 pressure; //pressio
unsigned int16 humidity; //humitat
unsigned int16 alt; //altitut
unsigned int16 alt_max; //altitut maxima
unsigned int16 alt_min; //altitut minima

int radi; //radi de la bici
float R; //constannt universal
float u; //massa molar de l'aire
float g; //constant gravitacional
float po; //pressio a nivell del mar (hPa)

long int c_ad_cor; //contador adreça cor
long int ad_cor; //adreça cor
long int c_ad_vel; //contador adreça velocitat
long int ad_vel; //adreça velocitat
long int c_ad_alt; //contador adreça altitut
long int ad_alt; //adreça altitut
long int c_ad_ped; //contador adreça cadencia de pedaleig
long int ad_ped; //adreça cadencia de pedaleig
int LOW_leer;    //llegir valor eeprom
int HIGH_leer; //llegir valor eeprom

//FUNCIONS
void guar_int16(long int adress, int16 valor); //guardar valor int16 a eeprom
void altitut(); //calcul altitut
void LDR(); //lluminositat
void uv(); //rajos uva
void gas(); //contaminacio
void temperatura(); //teperatura
void humitat(); //humitat
void vibracio(); //vibracio

//FUNCIONS
//INTERRUPCIONS
//interrupcio rb2
#int_ext2
void ext2_rb2()
{
  c_vel++;
}

//interrupcio externa rb4
#int_rb
void ext_rb4567()
{
  if (input(PIN_B6) == 1) //boto start
  {
    bstart++;
    if (bstart == 4)
    {
      bstart = 0;
    }
  }
  if (input(PIN_B7) == 1) //boto seguent
  {
    boto++;
    if(boto == 6)
    {
      boto = 0;
    }
  }
  if ((input(PIN_B4) == 1) && (bstart == 1))
  {
    c_cor++;
  }
  if ((input(PIN_B5) == 1) && (bstart == 1))
  {
    c_peda++;
  }
}

//interrupcio bluetooth
#int_rda
void rda_isr()
{
  con = getc(); //rep senyal de coneccio de bluetooth
  start = getc(); //rep start
  tot = getc(); //rep valors totals
  fi = getc(); //rep fi
}

//interrupcio T1
#int_TIMER1
void TIMER1_isr()
{
  int16 puls;
  set_timer1(64285); //1ms
  time_vel++;
  t_mem++;
  if (time_vel >= 2000) //2s
  {
    //vel = c_vel*30*2*3.14*radi*0.06; //Km/h
    //dist = ((c_vel*30*2*3.14*radi)/1000) + dist;//m
    //3.511 = 2*3.14*radi;
    puls = c_vel*30;
    dist = (puls*3.511)+ dist;
    vel = puls*3.511*0.06;
    pedal = c_peda * 30;
    c_peda = 0;
    c_vel = 0;
    time_vel = 0;
  }
  if (t_mem >= 2000) //2s -- guardar en memoria
  {
    //frecuencia cardiaca
    write_ext_eeprom(ad_cor,fr_cor);
    ad_cor++;
    c_ad_cor++;
    if (ad_cor >= 5461)
    {
      ad_cor = 0;
      c_ad_cor = 0;
    }

    //velocitat
    guar_int16(ad_vel,vel);
    ad_vel = ad_vel + 2;
    c_ad_vel++;
    if (ad_vel >= 16384)
    {
      ad_vel = 0;
      c_ad_vel = 0;
    }

    //altitut
    altitut();
    guar_int16(ad_alt,alt);
    ad_alt = ad_alt + 2;
    c_ad_alt++;
    if (ad_alt >= 27306)
    {
      ad_alt = 0;
      c_ad_alt = 0;
    }

    //c. pedaleig
    write_ext_eeprom(ad_ped,pedal);
    ad_ped++;
    c_ad_ped++;
    if (ad_ped >= 32767)
    {
      ad_ped = 0;
      c_ad_ped = 0;
    }
  }
}

//interrupcio T2
#int_TIMER3
void TIMER3_isr() //1ms
{
  set_timer3(64285);  //1ms
  t_sens++;
  time_cor++;
  if (time_cor >= 15000) //15s
  {
    fr_cor = c_cor*4;
    time_cor = 0;
    c_cor = 0;
  }
  if (t_sens >= 3000)
  {
    LDR();
    uv();
    gas();
    temperatura();
    humitat();
    vibracio();
    t_sens = 0;
  }
}

//CONFIGURACIONS
//variables 16 bits
void guar_int16(long int adress, int16 valor)
{
  int LOW;
  int HIGH;

  LOW=make8(valor, 0);
  HIGH=make8(valor, 1);

  //guardar el valor a la eeprom
  write_ext_eeprom(adress, HIGH);
  write_ext_eeprom(adress+1, LOW);
}

//configuracio interrupcions
void conf_interrupcions()
{
  enable_interrupts(INT_TIMER1); //habilita interrupcions timer1
  enable_interrupts(INT_TIMER3); //hibilita interrupcions timer3
  enable_interrupts(INT_EXT2_L2H); //interrupcio rb2 low to high
  enable_interrupts(INT_RDA); //hanilita interrupció bluetooth
  enable_interrupts(INT_RB); //interrupcio rb4 i 5
  enable_interrupts(global); //habilita les interrupcions de forma global
  EXT_INT_EDGE(L_TO_H); //low to high
  set_timer1(64285); //valor a timer1 -- 1ms
  set_timer3(64285); //valor a timer3 -- 1ms
  setup_timer_1(RTCC_INTERNAL | RTCC_DIV_4); //configuracio timer1
  setup_timer_3(RTCC_INTERNAL | RTCC_DIV_4); //configuracio timer3
}

//desactiva interrupcions
void no_interrupcions()
{
  disable_interrupts(INT_TIMER0);
  disable_interrupts(INT_TIMER1);
  disable_interrupts(INT_TIMER3);
  disable_interrupts(INT_EXT2_L2H);
}

//SENSORS
//rtc
void rtc()
{
  ds1307_get_date(dia,mes,any,diaset);
  ds1307_get_time(hrs,min,sec);
  lcd_gotoxy(1,1);
  printf(lcd_putc,"%02u/%02u/20%02u-%1u  ",dia,mes,any,diaset);
  lcd_gotoxy(1,2);
  printf(lcd_putc,"%02d:%02d:%02d", hrs,min,sec);
}

//ldr
void LDR()
{
  float v_dig; //valor digital
  int16 v_ana; //valor analogic

  set_adc_channel(0); //habilitacio canal de lectura AN0
  delay_us(20); //estabilitzacio
  v_ana = read_adc();
  v_dig = 5.0*v_ana/1024.0;
  lum = (100*v_dig)/5; //lumens
  if (lum > 41)
  {
    output_low(PIN_C0);
  }
  else
  {
    output_high(PIN_C0);
  }
}

//sensor rajos uva -- critic 142
void uv()
{

  float v_dig; //valor digital
  int16 v_ana; //valor analogic

  set_adc_channel(1); //habilitacio canal de lectura AN0
  delay_us(20); //estabilitzacio
  v_ana = read_adc();
  v_dig = 5.0*v_ana/1024.0;
  uva = v_ana;
}

//sensor gas -- + de 400 critic
void gas()
{
  set_adc_channel(2);
  delay_us(20);
  co2 = read_adc();
  if (co2 > 400)
  {
    output_high(PIN_C1);
  }
  else
  {
    output_low(PIN_C1);
  }
}

//temperatura
void temperatura()
{
  BME280_readTemperature(&temperature);
  temperature =  temperature/100;
}

//humitat
void humitat()
{
  BME280_readHumidity(&humidity);
  humidity = (humidity * 100)/1024;
}

//altitud
void altitut()
{
  int tempK; //temperatura en Kelvins

  BME280_readPressure(&pressure);
  BME280_readTemperature(&temperature);
  tempK = (temperature/100) + 273.15;
  alt = ((R*tempK)/(u*g))*log((pressure/100)/po);

}
//vibracio
void vibracio()
```

```
{
  int x;
  x = input(PIN_A3);
  if (x == 0)
  {
    output_high(PIN_C2);
    vibr = 1;
  }
  else
  {
    output_low(PIN_C2);
    vibr = 0;
  }
}

//llegir variables 16 bits
void leer_int16(long int adress)
{
  //llegir el valor a la eeprom
  HIGH_leer = read_ext_eeprom(adress);
  LOW_leer = read_ext_eeprom(adress+1);
}

//velocitat mitja
void velocitat_mitja()
{
  long int i;
  long int v_vel;
  ad_vel = 5462;

  for(i = 0 ; i >= c_ad_vel ; i++)
  {
    leer_int16(ad_vel);
    v_vel = make16(HIGH_leer,LOW_leer);
    ad_vel = ad_vel + 2;
    vel_mj = vel_mj + v_vel;
  }
  vel_mj = vel_mj/c_ad_vel;
}

//fr. cardiaca mitja
void cardiaca_mitja()
{
  long int i;
  int v_fr_cor;
  ad_cor = 0;

  for( i = 0 ; i >= c_ad_cor; i++)
  {
    v_fr_cor = read_ext_eeprom(ad_cor);
    ad_cor++;
    fr_cor_mj = v_fr_cor + fr_cor_mj;
  }
  fr_cor_mj = fr_cor_mj / c_ad_cor;
}

//cadencia de pedaleig mitja
void c_ped_mj()
{
    long int i;
    int v_ped;
    ad_ped = 27307;

    for( i = 0 ; i >= c_ad_ped; i++)
    {
      v_ped = read_ext_eeprom(ad_ped);
      ad_ped++;
      pedal_mj = v_ped + pedal_mj;
    }
    pedal_mj = pedal_mj / c_ad_ped;
}

//altitut maxima i minima
void altitut_max_min()
{
  long int i;
  long int v_alt;
  ad_alt = 16385;

  for(i = 0 ; i >= c_ad_alt ; i++)
  {
    leer_int16(ad_alt);
    v_alt = make16(HIGH_leer,LOW_leer);
    ad_alt = ad_alt + 2;
    if (v_alt > alt_max) alt_max = v_alt;
    if (v_alt < alt_min) alt_min = v_alt;
  }
}

//Temps de recorregut
void time_rec()
{
  int32 temps_rec;
  temps_rec = temps_fi - temps_ini;
  hrs_rec = temps_rec/3600;
  min_rec = (temps_rec - (temps_rec/3600))/60;
  sec_rec = temps_rec - (hrs_rec*3600) - (min_rec*60);
}

//PORGRAMA PRINCIPAL
void main()
{
  //inicializaciones
  lcd_init(); //lcd
  ds1307_init(); //rtc
  setup_adc_ports(AN0_TO_AN2); //Pins AN0 a AN2 analogics
  setup_adc(ADC_CLOCK_INTERNAL); //rellorge del adc
  init_ext_eeprom(); //inicialitzacio memoria externa 24LC256
  delay_ms(100);

  //interrupcions
  enable_interrupts(INT_RB); //interrupcio rb4, 5, 6, 7
  enable_interrupts(global); //habilita les interrupcions de forma global
  EXT_INT_EDGE(L_TO_H); //low to high

  //enviar data - linia només per la primera compilació
  //1 de maig 2020, divendres - 15:20:55
  //ds1307_set_date_time(dia,mes,any,diaset,hrs,min,sec);
  ds1307_set_date_time(26,5,20,7,9,21,55);
  delay_ms(100);

  //inicialitzacio variables
  //temporitzadors
  time_cor = 0; //timer de 15s
  time_vel = 0; //timer 2s
  t_sens = 0; //timer 3s
  t_mem = 0; //timer 2s

  //botons
  boto = 0; //contador boto
  bstart = 0; //estat bstart

  //constants
  radi = 0.559; //radi de la bici 559 mm
  R = 8.31432; //constannt universal
  u = 0.0289644; //massa molar de l'aire
  g = 8.31432; //constant gravitacional
  po = 1013.25; //pressio a nivell del mar (hPa)

  //variables sensors
  dist = 0; //distancia recorreguda
  vel = 0; //velocitat
  c_vel = 0; //contador velocitat
  vel_mj = 0;
```

```
uva = 0; //rajos uva
lum = 0; //lluminositat
fr_cor = 0; //frecuencia cardiaca
fr_cor_mj = 0; //fr. cardiaca mitja
c_cor = 0;
co2 = 0; //CO2
pedal = 0; //cadencia de pedaleig
pedal_mj = 0;
c_peda = 0; //contador pedal
temperature = 0;; //temperatura
pressure = 0; //pressio
humidity = 0; //humitat
alt = 0; //altitut
alt_max = 0;
alt_min = 6000;
vibr = 0;

//temps
temps_ini = 0; //moment inici
temps_fi = 0; //moment fi

//memoria
ad_cor = 0; //adreça cor
ad_vel = 5462; //adreça velocitat
ad_alt = 16385; //adreça altitut
ad_ped = 27307; //adreça cadencia de pedaleig
c_ad_cor = 0; //contador adreça cor
c_ad_vel = 0; //contador adreça vlocitat
c_ad_alt = 0; //contador adreça altitut
c_ad_ped = 0; //contador adreça cadencia de pedaleig
LOW_leer = 0; //llegir valor eeprom
HIGH_leer = 0; //llegir valor eeprom

//bluetooth
con = "X";
start = "X";
tot = "X";

while (1)
{
  lcd_putc("\f"); //borra LCD
  rtc();
  delay_ms(2000);
  lcd_putc("\f");
  printf(lcd_putc,"Pulsa START");
  delay_ms(2000);
  if (con == "C")
  {
```

```
while(con == "C")
{
  if (start == "S")
  {
    bstart = 1;
    conf_interrupcions();
    ds1307_get_time(hrs,min,sec);
    temps_ini = (hrs*3600) + (min*60) + sec;

    while (start == "S")
    {
      //altitut
      printf(alt);
      printf(" m");
      printf("|");
      //contaminacio
      printf(co2);
      printf(" ppm");
      printf("|");
      //cadencia de pedaleig
      printf(pedal);
      printf(" pd/min");
      printf("|");
      //distancia
      printf(dist);
      printf(" km");
      printf("|");
      //frequencia cardiaca
      printf(fr_cor);
      printf("|");
      //humitat
      printf(humidity);
      printf(" %%");
      printf("|");
      //lluminositat
      printf(lum);
      printf(" %%");
      printf("|");
      //temperatura
      printf(temperature);
      printf(" ºC");
      printf("|");
      //rajos uva
      printf(uva);
      printf("|");
      //velocitat
      printf(vel);
      printf(" km/h");
```

```
//indicadors
if (vibr == 1)
{
    printf("V");
}
else if (co2 > 400)
{
  printf("O");
}
else if (lum > 41)
{
   printf("L");
}

//valors totals
if (tot == "T")
{
   while (tot == "T")
   {
      no_interrupcions();
      bstart = 0;
      //altitut maxima
      printf(alt_max);
      printf(" m");
      printf("|");
      //altitut minima
      printf(alt_min);
      printf(" m");
      printf("|");
      //cadencia de pedaleig mitja
      printf(pedal_mj);
      printf(" pd/min");
      printf("|");
      //distancia maxima
      printf(dist);
      printf(" Km");
      printf("|");
      //frequencia cardiaca mitja
      printf(fr_cor_mj);
      printf("|");
      //temps de recorregut
      printf(hrs_rec);
      printf(":");
      printf(min_rec);
      printf(":");
      printf(sec_rec);
      printf("|");
```

```
//velocitat mitja
printf(vel_mj);
printf(" Km/h");

start = "X";
//fi
if (fi == "F")
{
  //desactivació
  con = "X";
  start = "X";
  tot = "X";
  fi = "X";

  lcd_putc("/f");
  //temps
  time_cor = 0; //timer de 15s
  time_vel = 0; //timer 2s
  t_sens = 0; //timer 3s
  t_mem = 0; //timer 2s

  //botons
  bstart = 0;
  boto = 0;

  //variables sensors
  dist = 0; //distancia recorreguda
  vel = 0; //velocitat
  c_vel = 0; //contador velocitat
  vel_mj = 0;
  uva = 0; //rajos uva
  lum = 0; //lluminositat
  fr_cor = 0; //frecuencia cardiaca
  fr_cor_mj = 0;
  c_cor = 0;
  co2 = 0; //CO2
  pedal = 0; //cadencia de pedaleig
  pedal_mj = 0;
  c_peda = 0; //contador pedal
  temperature = 0;; //temperatura
  pressure = 0; //pressio
  humidity = 0; //humitat
  alt = 0; //altitut
  alt_max = 0;
  alt_min = 6000;
  vibr = 0; //vibracio

  //temps
```

```
              temps_ini = 0; //moment inici
              temps_fi = 0; //moment fi

              //memoria
              ad_cor = 0; //adreça cor
              ad_vel = 5462; //adreça velocitat
              ad_alt = 16385; //adreça altitut
              ad_ped = 27307; //adreça cadencia de pedaleig
              c_ad_cor = 0; //contador adreça cor
              c_ad_vel = 0; //contador adreça vlocitat
              c_ad_alt = 0; //contador adreça altitut
              c_ad_ped = 0; //contador adreça cadencia de pedaleig
              LOW_leer = 0; //llegir valor eeprom
              HIGH_leer = 0; //llegir valor eeprom

            }
          }
        }
      }

      }
    }
  }

else
{
  while (bstart != 0)
  {
    if (bstart == 1)
    {
      lcd_putc("\f");
      rtc();
      temps_ini = (hrs*3600) + (min*60) + sec;

      //comença lectura sensors
      conf_interrupcions();
      while (bstart == 1)
      {
        switch(boto)
        {
          case(1):
          //pulsacions
          lcd_gotoxy(1,1);
          printf(lcd_putc, "Fr.card = %3u   ", fr_cor);

          //velocitat
          lcd_gotoxy(1,2);
          printf(lcd_putc, "Vel = %4Lu Km/h  ", vel);
```

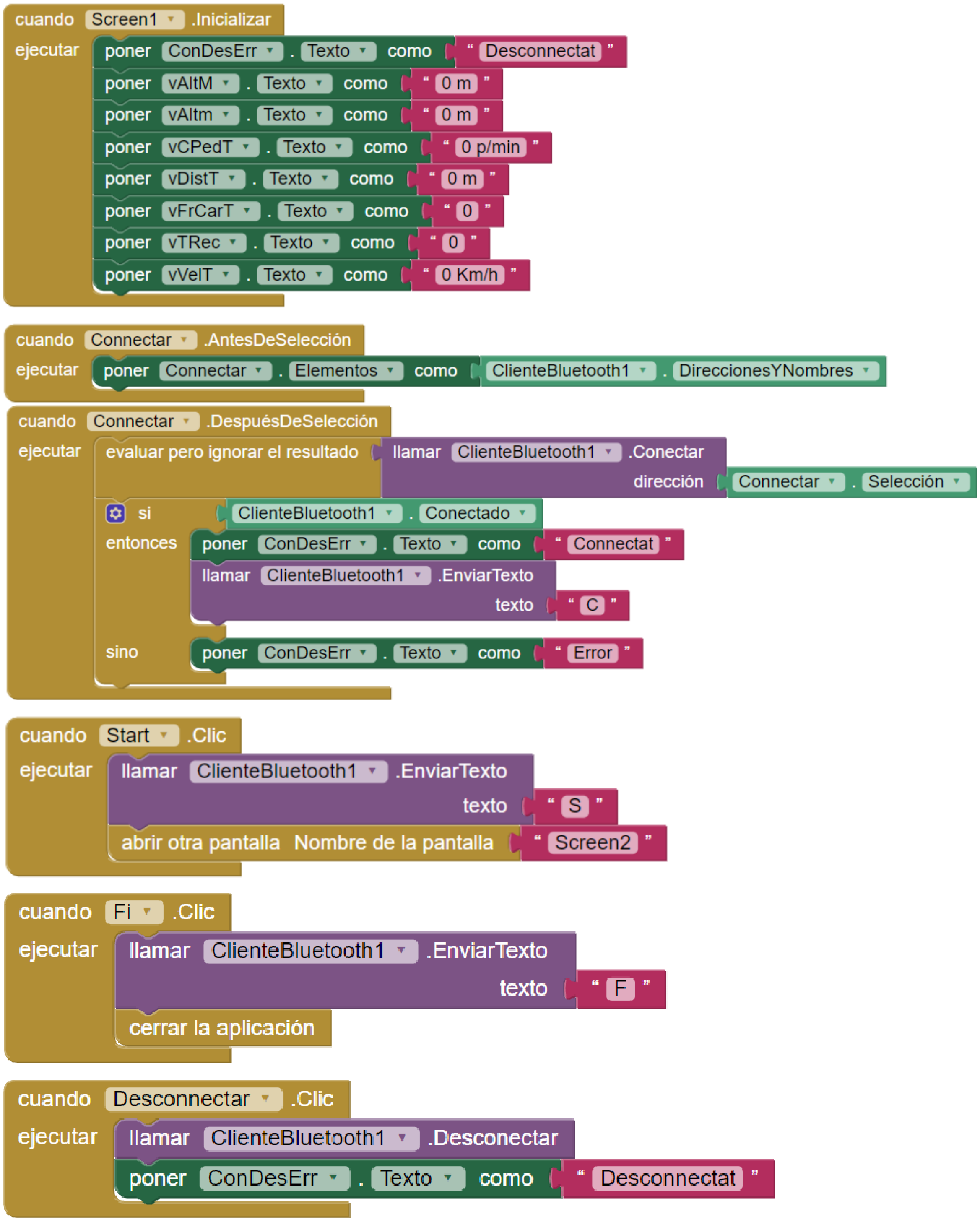
```
break;

case(2):
//temperatura
lcd_gotoxy(1,1);
if (temperature < 0)
{
  temperature = abs(temperature);
  printf(lcd_putc, "T = -%2Ld ºC      ", temperature);
}
else
{
  printf(lcd_putc, "T = %2Ld ºC        ", temperature);
}

//humitat
lcd_gotoxy(1,2);
printf(lcd_putc, "Hum = %03Lu        ", humidity % 100);
break;

case(3):
//uv
lcd_gotoxy(1,1);
printf(lcd_putc, "UV-A = %4Lu        ", uva);

//llum
lcd_gotoxy(1,2);
printf(lcd_putc, "Lumens = %3.0d%%      ", lum);
break;

case(4):
//altitut
lcd_gotoxy(1,1);
printf(lcd_putc, "Altitut = %4Lu m      ", alt);

//distancia
lcd_gotoxy(1,2);
printf(lcd_putc, "Dist = %5Lu m      ", dist);
break;

case(5):
//contaminacio
lcd_gotoxy(1,1);
printf(lcd_putc, "CO2 = %4Lu ppm       ", co2);

//cadencia de pedaleig
lcd_gotoxy(1,2);
printf(lcd_putc, "C.p = %3u pd/min       ", pedal);
```

```
      break;

      default:
      rtc();
      break;
    }
  }
}
if(bstart == 2)
{
  ds1307_get_time(hrs,min,sec);
  temps_fi = (hrs*3600) + (min*60) + sec;
  no_interrupcions();
  velocitat_mitja();
  cardiaca_mitja();
  c_ped_mj();
  altitut_max_min();
  time_rec();
  lcd_putc("\f");
  while (bstart == 2)
  {
    switch (boto)
    {
      case(1):
      //ditsnacia total
      lcd_gotoxy(1,1);
      printf(lcd_putc, "Dist.tot = %5Lu m      ", dist);

      //velocitat mitja
      lcd_gotoxy(1,2);
      printf(lcd_putc, "Vel.mj = %4Lu Km/h      ", vel_mj);
      break;

      case(2):
      //cardio
      lcd_gotoxy(1,1);
      printf(lcd_putc, "F.c.mj = %3u        ", fr_cor_mj);

      //c.pedaleig
      lcd_gotoxy(1,2);
      printf(lcd_putc, "C.p.m=%3u pd/min       ", pedal_mj);
      break;

      case(3):

      //altitut max
      lcd_gotoxy(1,1);
      printf(lcd_putc, "Alt.max = %4Lu m       ", alt_max);
      //altitut min
      lcd_gotoxy(1,2);
      printf(lcd_putc, "Alt.min = %4Lu m        ", alt_min);
      break;

      default:
      //temps de recorregut
      lcd_gotoxy(1,1);
      printf(lcd_putc,"Temps Recorregut");
      lcd_gotoxy(1,2);
      printf(lcd_putc,"%02d:%02d:%02d         ", hrs_rec,min_rec,sec_rec);
      break;
    }
  }
}
if (bstart == 3)
{
  lcd_putc("\f"); //borra LCD
  //temporitzadors
  time_cor = 0; //timer de 15s
  time_vel = 0; //timer 2s
  t_sens = 0; //timer 3s
  t_mem = 0; //timer 2s

  //botons
  bstart = 0;
  boto = 0;

  //variables sensors
  dist = 0; //distancia recorreguda
  vel = 0; //velocitat
  c_vel = 0; //contador velocitat
  vel_mj = 0;
  uva = 0; //rajos uva
  lum = 0; //lluminositat
  fr_cor = 0; //frecuencia cardiaca
  fr_cor_mj = 0;
  c_cor = 0;
  co2 = 0; //CO2
  pedal = 0; //cadencia de pedaleig
  pedal_mj = 0;
  c_peda = 0; //contador pedal
  temperature = 0;; //temperatura
  pressure = 0; //pressio
  humidity = 0; //humitat
  alt = 0; //altitut
  alt_max = 0;
```

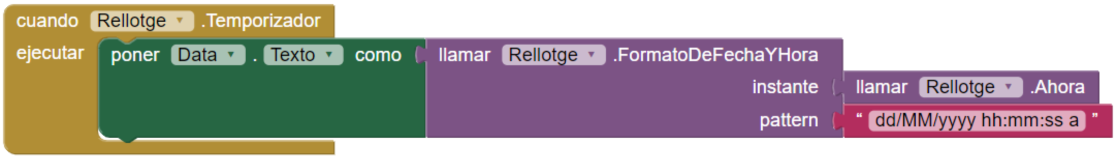
```
          alt_min = 6000;
          vibr = 0; //vibracio

          //temps
          temps_ini = 0; //moment inici
          temps_fi = 0; //moment fi

          //memoria
          ad_cor = 0; //adreça cor
          ad_vel = 5462; //adreça velocitat
          ad_alt = 16385; //adreça altitut
          ad_ped = 27307; //adreça cadencia de pedaleig
          c_ad_cor = 0; //contador adreça cor
          c_ad_vel = 0; //contador adreça vlocitat
          c_ad_alt = 0; //contador adreça altitut
          c_ad_ped = 0; //contador adreça cadencia de pedaleig
          LOW_leer = 0; //llegir valor eeprom
          HIGH_leer = 0; //llegir valor eeprom
        }
      }
    }
  }
}
```

2. Codi de l’aplicació mòbil

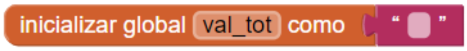
2.1. Pantalla 1



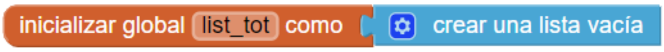
```
cuando Screen1 .Inicializar
ejecutar
    poner ConDesErr . Texto como " Desconnectat "
    poner vAltM . Texto como " 0 m "
    poner vAltm . Texto como " 0 m "
    poner vCPedT . Texto como " 0 p/min "
    poner vDistT . Texto como " 0 m "
    poner vFrCarT . Texto como " 0 "
    poner vTRec . Texto como " 0 "
    poner vVelT . Texto como " 0 Km/h "

cuando Connectar .AntesDeSelección
ejecutar
    poner Connectar . Elementos como ClienteBluetooth1 . DireccionesYNombres

cuando Connectar .DespuésDeSelección
ejecutar
    evaluar pero ignorar el resultado  llamar ClienteBluetooth1 .Conectar
                                                           dirección  Connectar . Selección
    si  ClienteBluetooth1 . Conectado
    entonces
        poner ConDesErr . Texto como " Connectat "
        llamar ClienteBluetooth1 .EnviarTexto
                                 texto " C "
    sino
        poner ConDesErr . Texto como " Error "

cuando Start .Clic
ejecutar
    llamar ClienteBluetooth1 .EnviarTexto
                             texto " S "
    abrir otra pantalla  Nombre de la pantalla " Screen2 "

cuando Fi .Clic
ejecutar
    llamar ClienteBluetooth1 .EnviarTexto
                             texto " F "
    cerrar la aplicación

cuando Desconnectar .Clic
ejecutar
    llamar ClienteBluetooth1 .Desconectar
    poner ConDesErr . Texto como " Desconnectat "
```



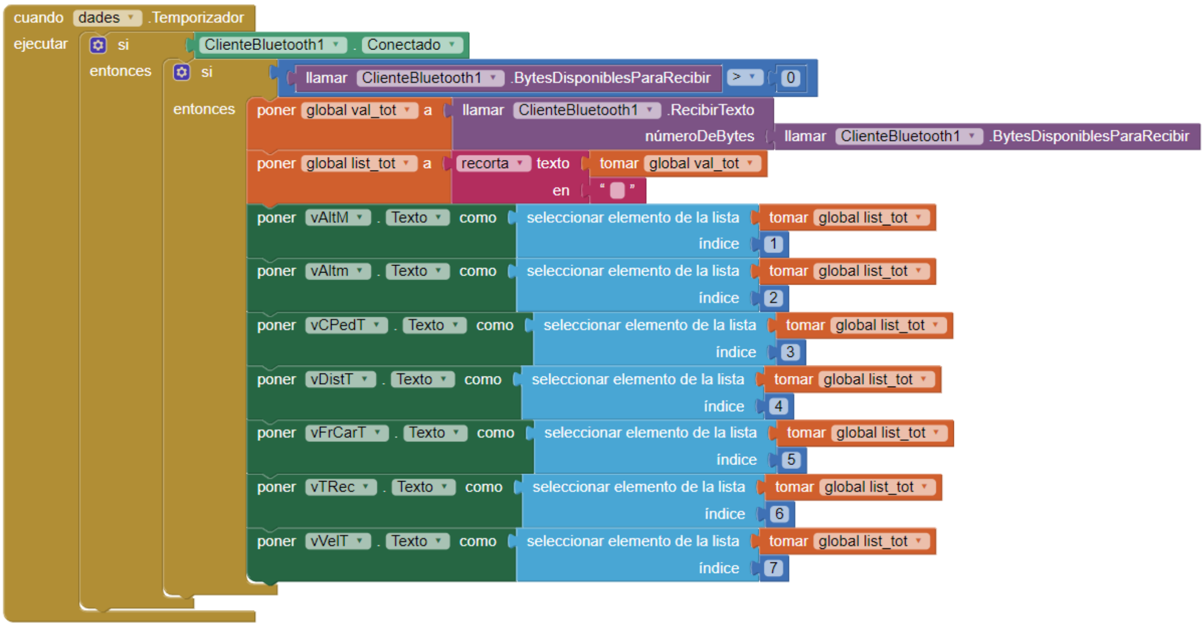
```
cuando Rellotge .Temporizador
ejecutar  poner Data . Texto como  llamar Rellotge .FormatoDeFechaYHora
                                                      instante  llamar Rellotge .Ahora
                                                      pattern   " dd/MM/yyyy hh:mm:ss a "
```



```
inicializar global val_tot como " "
```

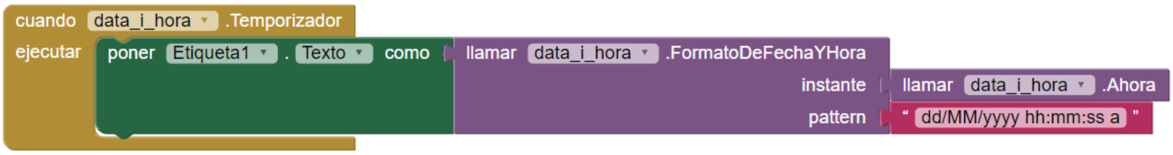


```
inicializar global list_tot como  crear una lista vacía
```

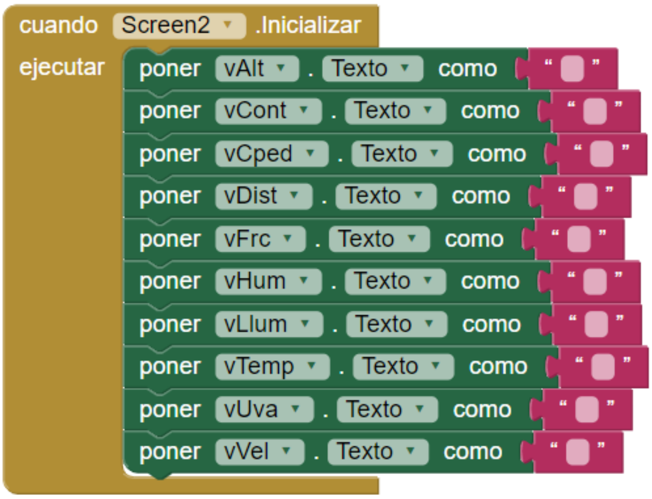


```
cuando dades .Temporizador
ejecutar  si  ClienteBluetooth1 . Conectado
          entonces  si  llamar ClienteBluetooth1 .BytesDisponiblesParaRecibir > 0
                    entonces  poner global val_tot a  llamar ClienteBluetooth1 .RecibirTexto
                                                          númeroDeBytes  llamar ClienteBluetooth1 .BytesDisponiblesParaRecibir
                              poner global list_tot a  recorta texto  tomar global val_tot
                                                                en    " "
                              poner vAltM . Texto como  seleccionar elemento de la lista  tomar global list_tot
                                                                                  índice  1
                              poner vAltm . Texto como  seleccionar elemento de la lista  tomar global list_tot
                                                                                  índice  2
                              poner vCPedT . Texto como  seleccionar elemento de la lista  tomar global list_tot
                                                                                   índice  3
                              poner vDistT . Texto como  seleccionar elemento de la lista  tomar global list_tot
                                                                                   índice  4
                              poner vFrCarT . Texto como  seleccionar elemento de la lista  tomar global list_tot
                                                                                    índice  5
                              poner vTRec . Texto como  seleccionar elemento de la lista  tomar global list_tot
                                                                                  índice  6
                              poner vVelT . Texto como  seleccionar elemento de la lista  tomar global list_tot
                                                                                  índice  7
```

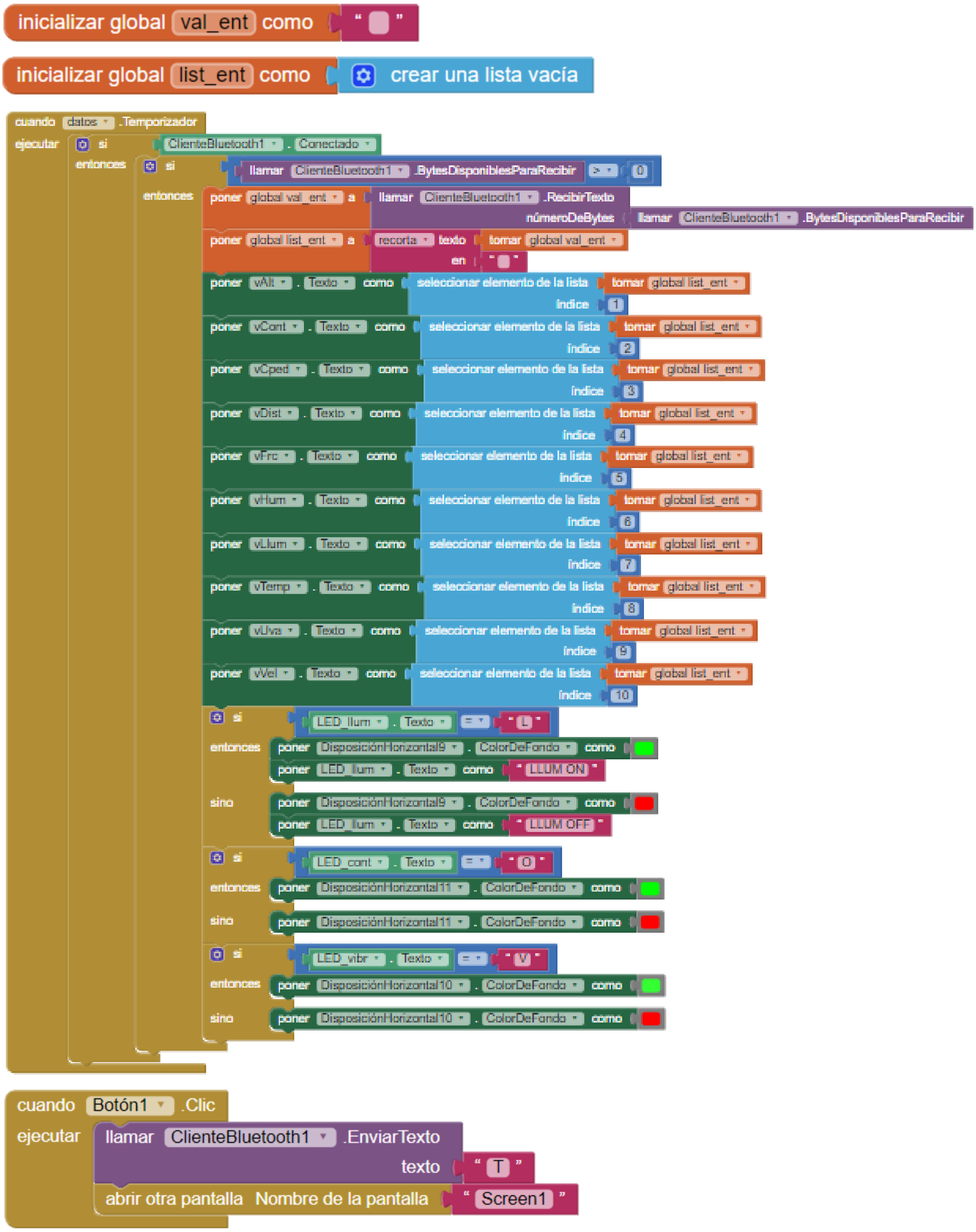
2.2. Pantalla 2



```
cuando data_i_hora .Temporizador
ejecutar  poner Etiqueta1 . Texto como  llamar data_i_hora .FormatoDeFechaYHora
                                                         instante  llamar data_i_hora .Ahora
                                                         pattern   " dd/MM/yyyy hh:mm:ss a "
```



```
cuando Screen2 .Inicializar
ejecutar  poner vAlt . Texto como " "
          poner vCont . Texto como " "
          poner vCped . Texto como " "
          poner vDist . Texto como " "
          poner vFrc . Texto como " "
          poner vHum . Texto como " "
          poner vLlum . Texto como " "
          poner vTemp . Texto como " "
          poner vUva . Texto como " "
          poner vVel . Texto como " "
```



```
inicializar global val_ent como " "

inicializar global list_ent como crear una lista vacía

cuando datos .Temporizador
ejecutar  si  ClienteBluetooth1 . Conectado
          entonces  si  llamar ClienteBluetooth1 .BytesDisponiblesParaRecibir > 0
                    entonces  poner global val_ent a  llamar ClienteBluetooth1 .RecibirTexto
                                                              númeroDeBytes  llamar ClienteBluetooth1 .BytesDisponiblesParaRecibir
                              poner global list_ent a  recorta texto  tomar global val_ent
                                                                en  " "
                              poner vAlt . Texto como  seleccionar elemento de la lista  tomar global list_ent
                                                                                 índice  1
                              poner vCont . Texto como  seleccionar elemento de la lista  tomar global list_ent
                                                                                  índice  2
                              poner vCped . Texto como  seleccionar elemento de la lista  tomar global list_ent
                                                                                  índice  3
                              poner vDist . Texto como  seleccionar elemento de la lista  tomar global list_ent
                                                                                  índice  4
                              poner vFrc . Texto como  seleccionar elemento de la lista  tomar global list_ent
                                                                                 índice  5
                              poner vHum . Texto como  seleccionar elemento de la lista  tomar global list_ent
                                                                                 índice  6
                              poner vLlum . Texto como  seleccionar elemento de la lista  tomar global list_ent
                                                                                  índice  7
                              poner vTemp . Texto como  seleccionar elemento de la lista  tomar global list_ent
                                                                                  índice  8
                              poner vUva . Texto como  seleccionar elemento de la lista  tomar global list_ent
                                                                                 índice  9
                              poner vVel . Texto como  seleccionar elemento de la lista  tomar global list_ent
                                                                                 índice  10
                              si  LED_llum . Texto = " L "
                              entonces  poner DisposiciónHorizontal9 . ColorDeFondo como 🟩
                                        poner LED_llum . Texto como " LLUM ON "
                              sino      poner DisposiciónHorizontal9 . ColorDeFondo como 🟥
                                        poner LED_llum . Texto como " LLUM OFF "
                              si  LED_cont . Texto = " O "
                              entonces  poner DisposiciónHorizontal11 . ColorDeFondo como 🟩
                              sino      poner DisposiciónHorizontal11 . ColorDeFondo como 🟥
                              si  LED_vibr . Texto = " V "
                              entonces  poner DisposiciónHorizontal10 . ColorDeFondo como 🟩
                              sino      poner DisposiciónHorizontal10 . ColorDeFondo como 🟥

cuando Botón1 .Clic
ejecutar  llamar ClienteBluetooth1 .EnviarTexto
                                   texto " T "
          abrir otra pantalla  Nombre de la pantalla " Screen1 "
```

3. Codi de la Simulació

3.1. Lluminositat

```
#include <18F4550.h> //pic
#device PASS_STRINGS = IN_RAM
#device adc=10
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#fuses HS //hs per cristall de mes de 4MHz
#fuses MCLR //habilita reset extern
#fuses NOWDT //no whatcdog
#fuses NOVREGEN //deshabilita sortida de 3.3V
#fuses CPUDIV1 //prescales a 48MHz
#fuses NOPROTECT //memoria no protegida contra lectures
#fuses PUT //habilita timer
#fuses NOLVP //PIN-B5 entrada/sortida
#fuses PLL1 //no prescaler
#fuses NOBROWNOUT //no reset a baix voltatge

#use delay (clock=20000000) //T = 500us
#use RS232(UART1,BAUD=9600, BITS=8, PARITY=N, XMIT=PIN_C6, RCV=PIN_C7, ERRORS) // bluetooth HC-06
#use standard_io(A)
#use standard_io(B)
#use standard_io(C)
#use standard_io(D)
#use standard_io(E)

//es posen mes abaix per q sino no funcionen
#include <lcd.c> //LCD

//VARIABLES GLOBALS
int32 t_sens; //timer 3s
int bstart; //boto start
int lum; //lluminositat


//FUNCIONS
void LDR(); //lluminositat

//FUNCIONS
//INTERRUPCIONS
```

```
//interrupcio externa rb4
#int_rb
void ext_rb4567()
{
  if (input(PIN_B6) == 1) //boto start
  {
    bstart++;
    delay_ms(300);
    if (bstart == 4)
    {
      bstart = 0;
    }
  }
}

//interrupcio T2
#int_TIMER3
void TIMER3_isr() //1ms
{
  set_timer3(64285);  //1ms
  t_sens++;
  if (t_sens >= 3000)
  {
    LDR();
    t_sens = 0;
  }
}

//CONFIGURACIONS

//configuracio interrupcions
void conf_interrupcions()
{
  enable_interrupts(INT_TIMER3); //hibilita interrupcions timer3
  enable_interrupts(INT_RB); //interrupcio rb4 i 5
  enable_interrupts(global); //habilita les interrupcions de forma global
  EXT_INT_EDGE(L_TO_H); //low to high
  set_timer3(64285); //valor a timer3 -- 1ms
  setup_timer_3(RTCC_INTERNAL | RTCC_DIV_4); //configuracio timer3
}

//desactiva interrupcions
void no_interrupcions()
{
  disable_interrupts(INT_TIMER3);
}

//SENSORS
```

```
//ldr
void LDR()
{
  float v_dig; //valor digital
  int16 v_ana; //valor analogic

  set_adc_channel(0); //habilitacio canal de lectura AN0
  delay_us(20); //estabilitzacio
  v_ana = read_adc();
  v_dig = 5.0*v_ana/1024.0;
  lum = (100*v_dig)/5; //lumens
  if (lum > 41)
  {
    output_low(PIN_C0);
  }
  else
  {
    output_high(PIN_C0);
  }
}

//PORGRAMA PRINCIPAL
void main()
{
  //inicializaciones
  lcd_init(); //lcd
  enable_interrupts(INT_RB); //interrupcio rb4 i 5
  enable_interrupts(global); //habilita les interrupcions de forma global
  EXT_INT_EDGE(L_TO_H); //low to high
  setup_adc_ports(AN0); //Pins AN0 analogics
  setup_adc(ADC_CLOCK_INTERNAL); //rellorge del adc
  delay_ms(100);

  //inicialitzacio variables
  //temporitzadors
  t_sens = 0; //timer 3s

  //botons i leds
  bstart = 0; //estat bstart

  //variables sensors
  lum = 0; //lluminositat

  while (1)
  {
    lcd_putc("\f"); //borra LCD
    delay_ms(100);
    printf(lcd_putc,"Pulsa START");
    delay_ms(2000);

    if (bstart == 1)
    {
      lcd_putc("\f");
      conf_interrupcions();

      while (bstart == 1)
      {
        //llum
        lcd_gotoxy(1,2);
        printf(lcd_putc, "Llum = %3.0d%%", lum);
      }
    }
    if (bstart == 3)
    {
      lcd_putc("\f"); //borra LCD
      t_sens = 0; //timer 3s
      //botons i leds
      bstart = 0;
      lum = 0; //lluminositat

    }
  }
}
```

3.2. Contaminació

```
#include <18F4550.h> //pic
#device PASS_STRINGS = IN_RAM
#device adc=10
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#fuses HS //hs per cristall de mes de 4MHz
#fuses MCLR //habilita reset extern
#fuses NOWDT //no whatcdog
#fuses NOVREGEN //deshabilita sortida de 3.3V
#fuses CPUDIV1 //prescales a 48MHz
#fuses NOPROTECT //memoria no protegida contra lectures
#fuses PUT //habilita timer
#fuses NOLVP //PIN-B5 entrada/sortida
#fuses PLL1 //no prescaler
#fuses NOBROWNOUT //no reset a baix voltatge

#use delay (clock=20M) //T = 500us
#use RS232(UART1,BAUD=9600, BITS=8, PARITY=N, XMIT=PIN_C6, RCV=PIN_C7, ERRORS) //
bluetooth HC-06
#use standard_io(A)
#use standard_io(B)
#use standard_io(C)
#use standard_io(D)
#use standard_io(E)

//es posen mes abaix per q sino no funcionen
#include <lcd.c> //LCD

//VARIABLES GLOBALS

int32 t_sens; //timer 3s
int bstart; //boto start
int16 co2; //CO2

//FUNCIONS
void gas(); //contaminacio

//FUNCIONS
//INTERRUPCIONS
//interrupcio rb2

//interrupcio externa rb4
#int_rb
void ext_rb4567()
```

```
{
  if (input(PIN_B6) == 1) //boto start
  {
    bstart++;
    delay_ms(300);
    if (bstart == 4)
    {
      bstart = 0;
    }
  }
}

//interrupcio T2
#int_TIMER3
void TIMER3_isr() //1ms
{
  set_timer3(64285);  //1ms
  t_sens++;
  if (t_sens >= 3000)
  {
    gas();
    t_sens = 0;
  }
}

//CONFIGURACIONS
//configuracio interrupcions
void conf_interrupcions()
{
  enable_interrupts(INT_TIMER3); //hibilita interrupcions timer3
  enable_interrupts(INT_RB); //interrupcio rb4 i 5
  enable_interrupts(global); //habilita les interrupcions de forma global
  EXT_INT_EDGE(L_TO_H); //low to high
  set_timer3(64285); //valor a timer3 -- 1ms
  setup_timer_3(RTCC_INTERNAL | RTCC_DIV_4); //configuracio timer3
}

//desactiva interrupcions
void no_interrupcions()
{
  disable_interrupts(INT_TIMER3);
}

//SENSORS
//sensor gas -- + de 400 critic
void gas()
{
  set_adc_channel(2);
  delay_us(20);
  co2 = read_adc();
  if (co2 > 400)
  {
    output_high(PIN_C1);
  }
  else
  {
    output_low(PIN_C1);
  }
}

//PORGRAMA PRINCIPAL
void main()
{
  //inicializaciones
  lcd_init(); //lcd
  setup_adc_ports(AN0_TO_AN2); //Pins AN0 a AN2 analogics
  setup_adc(ADC_CLOCK_INTERNAL); //rellorge del adc
  delay_ms(100);

  //interrupcions
  enable_interrupts(INT_RB); //interrupcio rb4, 5, 6, 7
  enable_interrupts(global); //habilita les interrupcions de forma global
  EXT_INT_EDGE(L_TO_H); //low to high

  //inicialitzacio variables
  //temporitzadors
  t_sens = 0; //timer 3s

  //botons
  bstart = 0; //estat bstart

  //variables sensors
  co2 = 0; //CO2

  while (1)
  {
    lcd_putc("\f");
    delay_ms(100);
    printf(lcd_putc,"Pulsa START");
    delay_ms(2000);

    if (bstart == 1)
    {
      lcd_putc("\f");
      //comença lectura sensors
      conf_interrupcions();
      while (bstart == 1)
      {
        //contaminacio
        lcd_gotoxy(1,1);
        printf(lcd_putc, "CO2 = %4Lu ppm", co2);
      }
    }
    if(bstart == 2)
    {
      no_interrupcions();
    }
    if (bstart == 3)
    {
      lcd_putc("\f"); //borra LCD
      //temporitzadors
      t_sens = 0; //timer 3s

      //botons
      bstart = 0;
      co2 = 0; //CO2
     }
  }
}
```

3.3. Vibració

```
#include <18F4550.h> //pic
#device PASS_STRINGS = IN_RAM
#device adc=10
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#fuses HS
#fuses MCLR //habilita reset extern
#fuses NOWDT //no whatcdog
#fuses NOVREGEN //deshabilita sortida de 3.3V
#fuses CPUDIV1 //prescales a 48MHz
#fuses NOPROTECT //memoria no protegida contra lectures
#fuses PUT //habilita timer
#fuses NOLVP //PIN-B5 entrada/sortida
#fuses PLL1 //no prescaler
#fuses NOBROWNOUT //no reset a baix voltatge

#use delay (clock=20M) //T = 500us
#use RS232(UART1,BAUD=9600, BITS=8, PARITY=N, XMIT=PIN_C6, RCV=PIN_C7, ERRORS) //
bluetooth HC-06
#use standard_io(A)
#use standard_io(B)
#use standard_io(C)
#use standard_io(D)
#use standard_io(E)

#include <lcd.c>

//VARIABLES GLOBALS
int32 t_sens; //timer 3s
int bstart; //boto start

//FUNCIONS
void vibracio(); //vibracio

//FUNCIONS
//INTERRUPCIONS
//interrupcio externa rb6
#int_rb
void ext_rb4567()
{
  if (input(PIN_B6) == 1) //boto start
  {
    bstart++;
    delay_ms(300); //per evitar rebots
    if (bstart == 4)
    {
      bstart = 0;
    }
  }
}

//interrupcio T2
#int_TIMER3
void TIMER3_isr() //1ms
{
  set_timer3(64285);  //1ms
  t_sens++;
  if (t_sens >= 3000)
  {
    vibracio();
    t_sens = 0;
  }
}

//CONFIGURACIONS
//configuracio interrupcions
void conf_interrupcions()
{
  enable_interrupts(INT_TIMER3); //hibilita interrupcions timer3
  enable_interrupts(INT_RB); //interrupcio rb4 i 5
  enable_interrupts(global); //habilita les interrupcions de forma global
  EXT_INT_EDGE(L_TO_H); //low to high
  set_timer3(64285); //valor a timer3 -- 1ms
  setup_timer_3(RTCC_INTERNAL | RTCC_DIV_4); //configuracio timer3
}

//desactiva interrupcions
void no_interrupcions()
{
  disable_interrupts(INT_TIMER3);
}

//SENSORS
//vibracio
void vibracio()
{
  int x;
  x = input(PIN_A3);
  if (x == 0)
  {
    output_high(PIN_C2);
  }
  else
  {
    output_low(PIN_C2);
  }
}

//PORGRAMA PRINCIPAL
void main()
{
  lcd_init();
  //interrupcions
  enable_interrupts(INT_RB); //interrupcio rb4, 5, 6, 7
  enable_interrupts(global); //habilita les interrupcions de forma global
  EXT_INT_EDGE(L_TO_H); //low to high

  //inicialitzacio variables
  //temporitzadors
  t_sens = 0; //timer 3s

  //botons
  bstart = 0; //estat bstart

  while (1)
  {
    lcd_putc("\f");
    delay_ms(100);
    printf(lcd_putc,"Pulsa START");
    delay_ms(2000);
    if (bstart == 1)
    {
      lcd_putc("\f");
      conf_interrupcions();
      while (bstart == 1)
      {
        lcd_gotoxy(1,1);
        printf(lcd_putc, "Simulant sensor");
        lcd_gotoxy(1,2);
        printf(lcd_putc, " de vibracio");
      }
    }
    if(bstart == 2)
    {
          no_interrupcions();
    }
    if (bstart == 3)
    {
      lcd_putc("\f"); //borra LCD
      t_sens = 0; //timer 3s
      bstart = 0;
    }
  }
}
```

3.4. Freqüència Cardíaca

```
#include <18F4550.h> //pic
#device PASS_STRINGS = IN_RAM
#device adc=10
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

//#fuses HS //hs per cristall de mes de 4MHz
#fuses XT
#fuses MCLR //habilita reset extern
#fuses NOWDT //no whatcdog
#fuses NOVREGEN //deshabilita sortida de 3.3V
#fuses CPUDIV1 //prescales a 48MHz
#fuses NOPROTECT //memoria no protegida contra lectures
#fuses PUT //habilita timer
#fuses NOLVP //PIN-B5 entrada/sortida
#fuses PLL1 //no prescaler
#fuses NOBROWNOUT //no reset a baix voltatge

#use delay (clock=20M) //T = 500us
#use RS232(UART1,BAUD=9600, BITS=8, PARITY=N, XMIT=PIN_C6, RCV=PIN_C7, ERRORS) // bluetooth HC-06
#use standard_io(A)
#use standard_io(B)
#use standard_io(C)
#use standard_io(D)
#use standard_io(E)

//es posen mes abaix per q sino no funcionen
#include <lcd.c> //LCD

//VARIABLES GLOBALS

int32 time_cor; //timer de 15s
int bstart; //boto start
int fr_cor; //frecuencia cardiaca
int fr_cor_mj; //fr. cardiaca mitja
int c_cor;

//FUNCIONS
//INTERRUPCIONS
//interrupcio externa rb4
#int_rb
void ext_rb4567()
{
  if (input(PIN_B6) == 1) //boto start
  {
    bstart++;
    delay_ms(300); //er evitar rebots
    if (bstart == 4)
    {
      bstart = 0;
    }
  }
  if ((input(PIN_B4) == 1) && (bstart == 1))
  {
    c_cor++;
  }
}

//interrupcio T0
#int_TIMER3
void TIMER3_isr()
{
  set_timer3(64285); //1ms
  time_cor++;
  if (time_cor >= 15000) //15s
  {
    fr_cor = c_cor*4;
    time_cor = 0;
    c_cor = 0;
  }
}

//CONFIGURACIONS
//configuracio interrupcions
void conf_interrupcions()
{
  enable_interrupts(INT_TIMER3); //habilita interrupcions timer0
  enable_interrupts(INT_RB); //interrupcio rb4 i 5
  enable_interrupts(global); //habilita les interrupcions de forma global
  EXT_INT_EDGE(L_TO_H); //low to high
  set_timer3(64285); //valor a timer0 -- 1ms
  setup_timer_3(RTCC_INTERNAL | RTCC_DIV_4); //conf timer0
}

//desactiva interrupcions
void no_interrupcions()
{
  disable_interrupts(INT_TIMER3);
}

//PORGRAMA PRINCIPAL
void main()
```

```
{
  //inicializaciones
  lcd_init(); //lcd
  delay_ms(100);

  //interrupcions
  enable_interrupts(INT_RB); //interrupcio rb4, 5, 6, 7
  enable_interrupts(global); //habilita les interrupcions de forma global
  EXT_INT_EDGE(L_TO_H); //low to high

  //inicialitzacio variables
  //temporitzadors
  time_cor = 0; //timer de 15s

  //botons
  bstart = 0; //estat bstart

  //variables sensors
  fr_cor = 0; //frecuencia cardiaca
  fr_cor_mj = 0; //fr. cardiaca mitja
  c_cor = 0;

  while (1)
  {
    lcd_putc("\f");
    printf(lcd_putc,"Pulsa START");
    delay_ms(2000);
    if (bstart == 1)
    {
      lcd_putc("\f");
      //comença lectura sensors
      conf_interrupcions();
      while (bstart == 1)
      {
          //pulsacions
          lcd_gotoxy(1,1);
          printf(lcd_putc, "Fr.card = %3u", fr_cor);
      }
    }
    if(bstart == 2)
    {
      no_interrupcions();
      lcd_putc("\f");
    }
    if (bstart == 3)
    {
      lcd_putc("\f"); //borra LCD
      //temporitzadors
```

```
        time_cor = 0; //timer de 15s

        //botons
        bstart = 0;

        fr_cor = 0; //frecuencia cardiaca
        fr_cor_mj = 0;
        c_cor = 0;
      }
    }
  }
```

3.5. Velocitat i Distancia

```
#include <18F4550.h> //pic
#device PASS_STRINGS = IN_RAM
#device adc=10
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#fuses HS //hs per cristall de mes de 4MHz
#fuses MCLR //habilita reset extern
#fuses NOWDT //no whatcdog
#fuses NOVREGEN //deshabilita sortida de 3.3V
#fuses CPUDIV1 //prescales a 48MHz
#fuses NOPROTECT //memoria no protegida contra lectures
#fuses PUT //habilita timer
#fuses NOLVP //PIN-B5 entrada/sortida
#fuses PLL1 //no prescaler
#fuses NOBROWNOUT //no reset a baix voltatge

#use delay (clock=20M) //T = 500us
#use RS232(UART1,BAUD=9600, BITS=8, PARITY=N, XMIT=PIN_C6, RCV=PIN_C7, ERRORS) //
bluetooth HC-06
#use standard_io(A)
#use standard_io(B)
#use standard_io(C)
#use standard_io(D)
#use standard_io(E)

//es posen mes abaix per q sino no funcionen
#include <lcd.c> //LCD

//VARIABLES GLOBALS
int32 time_vel; //timer 2s
int bstart; //boto start
int boto; //boto seguent
int32 dist; //distancia
int16 vel; //velocitat
int c_vel; //contador pulsos velocitat

int radi; //radi de la bici

//FUNCIONS
//INTERRUPCIONS
//interrupcio rb2
#INT_EXT2
void ext2_rb2()
{
  c_vel++;
}

//interrupcio externa rb4
#int_rb
void ext_rb4567()
{
  if (input(PIN_B6) == 1) //boto start
  {
    bstart++;
    if (bstart >= 4)
    {
      bstart = 0;
    }
  }
  if (input(PIN_B7) == 1) //boto seguent
  {
    boto++;
    if(boto >= 6)
    {
      boto = 0;
    }
  }
}

//interrupcio T1
#int_TIMER1
void TIMER1_isr()
{
  int16 puls;
  set_timer1(64285); //1ms
  time_vel++;
  if (time_vel >= 2000) //2s
  {
    //vel = c_vel*30*2*3.14*radi*0.06; //m/h
    //cnt = 2*3.14*radi;
    puls = c_vel*30;
    dist = (puls*3.511)+ dist;
    vel = puls*3.511*0.06;
    //dist = ((c_vel*30*2*3.14*radi)/1000) + dist;//m
    time_vel = 0;
    c_vel = 0;
  }
}

//CONFIGURACIONS
//configuracio interrupcions
void conf_interrupcions()
```

```
{
  enable_interrupts(INT_TIMER1); //habilita interrupcions timer1
  enable_interrupts(INT_EXT2_L2H); //interrupcio rb2 low to high
  enable_interrupts(INT_RB); //interrupcio rb4 i 5
  EXT_INT_EDGE(0,L_TO_H); //low to high
  enable_interrupts(global); //habilita les interrupcions de forma global
  set_timer1(64285); //valor a timer1 -- 1ms
  setup_timer_1(RTCC_INTERNAL | RTCC_DIV_4); //configuracio timer1
}

//desactiva interrupcions
void no_interrupcions()
{
  disable_interrupts(INT_TIMER1);
  disable_interrupts(INT_EXT2_L2H);
}

//PORGRAMA PRINCIPAL
void main()
{
  //inicializaciones
  lcd_init(); //lcd

  //interrupcions
  enable_interrupts(INT_RB); //interrupcio rb4, 5, 6, 7
  EXT_INT_EDGE(0,L_TO_H); //low to high
  enable_interrupts(global); //habilita les interrupcions de forma global


  //inicialitzacio variables
  //temporitzadors
  time_vel = 0; //timer 2s

  //botons
  boto = 0; //contador boto
  bstart = 0; //estat bstart

  //constants
  radi = 0.559; //radi de la bici 559 m

  //variables sensors
  dist = 0; //distancia recorreguda
  vel = 0; //velocitat
  c_vel = 0; //contador velocitat

  while (1)
  {
    lcd_putc("\f");
```

```
printf(lcd_putc,"Pulsa START");
delay_ms(2000);
if (bstart == 1)
{
  lcd_putc("\f");
  //comença lectura sensors
  conf_interrupcions();
  while (bstart == 1)
  {
    switch(boto)
    {
      case(1):
      //velocitat
      lcd_gotoxy(1,2);
      printf(lcd_putc, "Vel = %4Lu Km/h", vel);
      break;

      default:
      //distancia
      lcd_gotoxy(1,2);
      printf(lcd_putc, "Dist = %5Lu m", dist);
      break;
    }
  }
}
if(bstart == 2)
{
  lcd_putc("\f");
  no_interrupcions();
  while (bstart == 2)
  {
    lcd_gotoxy(1,1);
    printf(lcd_putc, "Dist.t = %5Lu m   ", dist);
  }
}
if (bstart == 3)
{
  lcd_putc("\f"); //borra LCD
  time_vel = 0; //timer 2s

  //botons
  bstart = 0;
  boto = 0;

  //variables sensors
  dist = 0; //distancia recorreguda
  vel = 0; //velocitat
  c_vel = 0; //contador velocitat
```

```
    }
  }
}
```

3.6. Cadència de pedaleig

```
#include <18F4550.h> //pic
#device PASS_STRINGS = IN_RAM
#device adc=10
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#fuses HS //hs per cristall de mes de 4MHz
#fuses MCLR //habilita reset extern
#fuses NOWDT //no whatcdog
#fuses NOVREGEN //deshabilita sortida de 3.3V
#fuses CPUDIV1 //prescales a 48MHz
#fuses NOPROTECT //memoria no protegida contra lectures
#fuses PUT //habilita timer
#fuses NOLVP //PIN-B5 entrada/sortida
#fuses PLL1 //no prescaler
#fuses NOBROWNOUT //no reset a baix voltatge

#use delay (clock=20M) //T = 500us
#use RS232(UART1,BAUD=9600, BITS=8, PARITY=N, XMIT=PIN_C6, RCV=PIN_C7, ERRORS) //
bluetooth HC-06
#use standard_io(A)
#use standard_io(B)
#use standard_io(C)
#use standard_io(D)
#use standard_io(E)

//es posen mes abaix per q sino no funcionen
#include <lcd.c> //LCD

//VARIABLES GLOBALS
int32 time_vel; //timer 2s

int bstart; //boto start

int pedal; //cadencia de pedaleig
int c_peda;

//FUNCIONS
//INTERRUPCIONS
//interrupcio externa rb4
#int_rb
void ext_rb4567()
{
  if (input(PIN_B6) == 1) //boto start
  {
    bstart++;
    if (bstart == 4)
    {
      bstart = 0;
    }
  }
  if ((input(PIN_B5) == 1) && (bstart == 1))
  {
    c_peda++;
  }
}

//interrupcio T1
#int_TIMER1
void TIMER1_isr()
{
  int16 puls;
  set_timer1(64285); //1ms
  time_vel++;
  if (time_vel >= 2000) //2s
  {
    pedal = c_peda*30;
    c_peda = 0;
    time_vel = 0;
  }
}

//CONFIGURACIONS
//configuracio interrupcions
void conf_interrupcions()
{
  enable_interrupts(INT_TIMER1); //habilita interrupcions timer1
  enable_interrupts(INT_RB); //interrupcio rb4 i 5
  enable_interrupts(global); //habilita les interrupcions de forma global
  EXT_INT_EDGE(L_TO_H); //low to high
  set_timer1(64285); //valor a timer1 -- 1ms
  setup_timer_1(RTCC_INTERNAL | RTCC_DIV_4); //configuracio timer1
}

//desactiva interrupcions
void no_interrupcions()
{
  disable_interrupts(INT_TIMER1);
}

//PORGRAMA PRINCIPAL
void main()
{
```

```
//inicializaciones
lcd_init(); //lcd

//interrupcions
enable_interrupts(INT_RB); //interrupcio rb4, 5, 6, 7
enable_interrupts(global); //habilita les interrupcions de forma global
EXT_INT_EDGE(L_TO_H); //low to high

//inicialitzacio variables
//temporitzadors
time_vel = 0; //timer 2s

//botons
bstart = 0; //estat bstart

//variables sensors
pedal = 0; //cadencia de pedaleig
c_peda = 0; //contador pedal

while (1)
{
  lcd_putc("\f");
  printf(lcd_putc,"Pulsa START");
  delay_ms(2000);

  if (bstart == 1)
  {
    lcd_putc("\f");

    //comença lectura sensors
    conf_interrupcions();
    while (bstart == 1)
    {
      //cadencia de pedaleig
      lcd_gotoxy(1,2);
      printf(lcd_putc, "C.p = %3u pd/min", pedal);
    }
  }
  if(bstart == 2)
  {
    no_interrupcions();
    lcd_putc("\f");
  }
  if (bstart == 3)
  {
    lcd_putc("\f"); //borra LCD

    //temporitzadors
```

```
      time_vel = 0; //timer 2s

      //botons
      bstart = 0;

      //variables sensors
      pedal = 0; //cadencia de pedaleig
      c_peda = 0; //contador pedal
    }
  }
}
```

3.7. Rajos UV-A

```
#include <18F4550.h> //pic
#device PASS_STRINGS = IN_RAM
#device adc=10
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

//#fuses HS //hs per cristall de mes de 4MHz
#fuses XT
#fuses MCLR //habilita reset extern
#fuses NOWDT //no whatcdog
#fuses NOVREGEN //deshabilita sortida de 3.3V
#fuses CPUDIV1 //prescales a 48MHz
#fuses NOPROTECT //memoria no protegida contra lectures
#fuses PUT //habilita timer
#fuses NOLVP //PIN-B5 entrada/sortida
#fuses PLL1 //no prescaler
#fuses NOBROWNOUT //no reset a baix voltatge

#use delay (clock=20M) //T = 500us
#use RS232(UART1,BAUD=9600, BITS=8, PARITY=N, XMIT=PIN_C6, RCV=PIN_C7, ERRORS) // bluetooth HC-06
#use standard_io(A)
#use standard_io(B)
#use standard_io(C)
#use standard_io(D)
#use standard_io(E)

//es posen mes abaix per q sino no funcionen
#include <lcd.c> //LCD

//VARIABLES GLOBALS
int32 t_sens; //timer 3s
int bstart; //boto start
int16 uva; //valor rajos uva

//FUNCIONS
void uv(); //rajos uva

//FUNCIONS
//INTERRUPCIONS
//interrupcio externa rb4
#int_rb
void ext_rb4567()
{
  if (input(PIN_B6) == 1) //boto start
  {
    bstart++;
    if (bstart == 4)
    {
      bstart = 0;
    }
  }
}

//interrupcio T2
#int_TIMER3
void TIMER3_isr() //1ms
{
  set_timer3(64285);  //1ms
  t_sens++;
  if (t_sens >= 3000)
  {
    uv();
    t_sens = 0;
  }
}

//CONFIGURACIONS
//configuracio interrupcions
void conf_interrupcions()
{
  enable_interrupts(INT_TIMER3); //hibilita interrupcions timer3
  enable_interrupts(INT_RB); //interrupcio rb4 i 5
  enable_interrupts(global); //habilita les interrupcions de forma global
  EXT_INT_EDGE(L_TO_H); //low to high
  set_timer3(64285); //valor a timer3 -- 1ms
  setup_timer_3(RTCC_INTERNAL | RTCC_DIV_4); //configuracio timer3
}

//desactiva interrupcions
void no_interrupcions()
{
  disable_interrupts(INT_TIMER3);
}

//SENSORS
//sensor rajos uva -- critic 142
void uv()
{

  float v_dig; //valor digital
  int16 v_ana; //valor analogic
  set_adc_channel(1); //habilitacio canal de lectura AN0
  delay_us(20); //estabilitzacio
  v_ana = read_adc();
  v_dig = 5.0*v_ana/1024.0;
  uva = v_ana;
}

//PORGRAMA PRINCIPAL
void main()
{
  //inicializaciones
  lcd_init(); //lcd
  setup_adc_ports(AN0_TO_AN2); //Pins AN0 a AN2 analogics
  setup_adc(ADC_CLOCK_INTERNAL); //rellorge del adc
  delay_ms(100);

  //interrupcions
  enable_interrupts(INT_RB); //interrupcio rb4, 5, 6, 7
  enable_interrupts(global); //habilita les interrupcions de forma global
  EXT_INT_EDGE(L_TO_H); //low to high

  //inicialitzacio variables
  //temporitzadors
  t_sens = 0; //timer 3s

  //botons
  bstart = 0; //estat bstart

  //variables sensors
  uva = 0; //rajos uva

  while (1)
  {
    lcd_putc("\f");
    printf(lcd_putc,"Pulsa START");
    delay_ms(2000);

    if (bstart == 1)
    {
      lcd_putc("\f");
      //comença lectura sensors
      conf_interrupcions();
      while (bstart == 1)
      {
        //uv
        lcd_gotoxy(1,1);
        printf(lcd_putc, "UV-A = %4Lu", uva);
      }
```

```
        if(bstart == 2)
        {
          no_interrupcions();
          lcd_putc("\f");
        }
        if (bstart == 3)
        {
          lcd_putc("\f"); //borra LCD
          //temporitzadors
          t_sens = 0; //timer 3s

          //botons
          bstart = 0;

          //variables sensors
          uva = 0; //rajos uva
         }
      }
   }
}
```

3.8. Rellotge de temps real

```
#include <18F4550.h> //pic
#device PASS_STRINGS = IN_RAM
#device adc=10
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#fuses HS //hs per cristall de mes de 4MHz
#fuses MCLR //habilita reset extern
#fuses NOWDT //no whatcdog
#fuses NOVREGEN //deshabilita sortida de 3.3V
#fuses CPUDIV1 //prescales a 48MHz
#fuses NOPROTECT //memoria no protegida contra lectures
#fuses PUT //habilita timer
#fuses NOLVP //PIN-B5 entrada/sortida
#fuses PLL1 //no prescaler
#fuses NOBROWNOUT //no reset a baix voltatge

#use delay (clock=20M) //T = 500us
#use RS232(UART1,BAUD=9600, BITS=8, PARITY=N, XMIT=PIN_C6, RCV=PIN_C7, ERRORS) // bluetooth HC-06
#use standard_io(A)
#use standard_io(B)
#use standard_io(C)
#use standard_io(D)
#use standard_io(E)

//es posen mes abaix per q sino no funcionen
#include <lcd.c> //LCD
#include <ds1307.c> //RTC

//VARIABLES GLOBALS
//rtc
BYTE sec;
BYTE min;
BYTE hrs;
BYTE dia;
BYTE mes;
BYTE any;
BYTE diaset;

int32 temps_ini; //moment d'inici
int32 temps_fi; //moment fi
int hrs_rec; //hores de recorregut
int min_rec; //minuts recorregut
int sec_rec; //segons recorregut
int bstart; //boto start

//FUNCIONS
//INTERRUPCIONS
//interrupcio externa rb4
#int_rb
void ext_rb4567()
{
  if (input(PIN_B6) == 1) //boto start
  {
    bstart++;
    if (bstart == 4)
    {
      bstart = 0;
    }
  }
}

//configuracio interrupcions
void conf_interrupcions()
{
  enable_interrupts(INT_RB); //interrupcio rb4 i 5
  enable_interrupts(global); //habilita les interrupcions de forma global
  EXT_INT_EDGE(L_TO_H); //low to high
}

//SENSORS
//rtc
void rtc()
{
  ds1307_get_date(dia,mes,any,diaset);
  ds1307_get_time(hrs,min,sec);
  lcd_gotoxy(1,1);
  printf(lcd_putc,"%02u/%02u/20%02u-%1u",dia,mes,any,diaset);
  lcd_gotoxy(1,2);
  printf(lcd_putc,"%02d:%02d:%02d", hrs,min,sec);
}

//Temps de recorregut
void time_rec()
{
  int32 temps_rec;
  temps_rec = temps_fi - temps_ini;
  hrs_rec = temps_rec/3600;
  min_rec = (temps_rec - (temps_rec/3600))/60;
  sec_rec = temps_rec - (hrs_rec*3600) - (min_rec*60);
}
```

```
//PORGRAMA PRINCIPAL
void main()
{
  //inicializaciones
  lcd_init(); //lcd
  ds1307_init(); //rtc

  //interrupcions
  enable_interrupts(INT_RB); //interrupcio rb4, 5, 6, 7
  enable_interrupts(global); //habilita les interrupcions de forma global
  EXT_INT_EDGE(L_TO_H); //low to high

  //enviar data - linia només per la primera compilació
  //31 de maig 2020, diumenge - 1:22:55
  ds1307_set_date_time(31,5,20,1,10,22,55);
  delay_ms(100);

  //inicialitzacio variables
  //botons
  bstart = 0; //estat bstart

  //temps
  temps_ini = 0; //moment inici
  temps_fi = 0; //moment fi
  hrs_rec = 0;
  min_rec = 0;
  sec_rec = 0;

  while (1)
  {
    lcd_putc("\f"); //borra LCD
    rtc();
    delay_ms(2000);
    lcd_putc("\f");
    printf(lcd_putc,"Pulsa START");
    delay_ms(2000);
    if (bstart != 0)
    {
      if (bstart == 1)
      {
        lcd_putc("\f");
        rtc();
        temps_ini = (hrs*3600) + (min*60) + sec;

        //comença lectura sensors
        conf_interrupcions();
        while (bstart == 1)
        {
```

```
      rtc();
    }

    if(bstart == 2)
    {
     ds1307_get_date(diaset,dia,mes,any);
     ds1307_get_time(hrs,min,sec);
     temps_fi = (hrs*3600) + (min*60) + sec;
     lcd_putc("\f");
     time_rec();
     while (bstart == 2)
     {
       //temps de recorregut
       lcd_gotoxy(1,1);
       printf(lcd_putc,"Temps Recorregut");
       lcd_gotoxy(1,2);
       printf(lcd_putc,"%02d:%02d:%02d", hrs_rec,min_rec,sec_rec);
     }
    }
    if (bstart == 3)
    {
     lcd_putc("\f"); //borra LCD
     //botons
     bstart = 0;

     //temps
     temps_ini = 0; //moment inici
     temps_fi = 0; //moment fi
     hrs_rec = 0;
     min_rec = 0;
     sec_rec = 0;
    }
   }
  }
 }
}
```

3.9. Tots els sensors

```
#include <18F4550.h> //pic
#device PASS_STRINGS = IN_RAM
#device adc=10
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

//#fuses HS //hs per cristall de mes de 4MHz
#fuses XT
#fuses MCLR //habilita reset extern
#fuses NOWDT //no whatcdog
#fuses NOVREGEN //deshabilita sortida de 3.3V
#fuses CPUDIV1 //prescales a 48MHz
#fuses NOPROTECT //memoria no protegida contra lectures
#fuses PUT //habilita timer
#fuses NOLVP //PIN-B5 entrada/sortida
#fuses PLL1 //no prescaler
#fuses NOBROWNOUT //no reset a baix voltatge

#use delay (clock=20M) //T = 500us
#use RS232(UART1,BAUD=9600, BITS=8, PARITY=N, XMIT=PIN_C6, RCV=PIN_C7, ERRORS) //
bluetooth HC-06
#use standard_io(A)
#use standard_io(B)
#use standard_io(C)
#use standard_io(D)
#use standard_io(E)

//es posen mes abaix per q sino no funcionen
#include <lcd.c> //LCD
#include <ds1307.c> //RTC

//VARIABLES GLOBALS
//rtc
BYTE sec;
BYTE min;
BYTE hrs;
BYTE dia;
BYTE mes;
BYTE any;
BYTE diaset;

int32 time_cor; //timer de 15s
int32 time_vel; //timer 2s
int16 t_mem; //timer 1s
int32 t_sens; //timer 3s
```

```
int32 temps_ini; //moment d'inici
int32 temps_fi; //moment fi
int hrs_rec; //hores de recorregut
int min_rec; //minuts recorregut
int sec_rec; //segons recorregut

char con; //conectivitat amb bluetooth
char start; //bluetooth start
char tot; //bluetooth valors totals
char fi; //bluetooth fi

int bstart; //boto start
int boto; //boto seguent

int lum; //lluminositat
int fr_cor; //frecuencia cardiaca
int fr_cor_mj; //fr. cardiaca mitja
int c_cor;
int32 dist; //distancia
int16 vel; //velocitat
int c_vel; //contador pulsos velocitat
int16 vel_mj; //velocitat mitja
int16 uva; //valor rajos uva
int16 co2; //CO2
int pedal; //cadencia de pedaleig
int pedal_mj; //cadencia de pedaleig mitja
int c_peda;
int vibr;
signed int16 temperature; //temperatura
int16 pressure; //pressio
unsigned int16 humidity; //humitat
unsigned int16 alt; //altitut
unsigned int16 alt_max; //altitut maxima
unsigned int16 alt_min; //altitut minima

int radi; //radi de la bici
float R; //constannt universal
float u; //massa molar de l'aire
float g; //constant gravitacional
float po; //pressio a nivell del mar (hPa)

long int c_ad_cor; //contador adreça cor
long int ad_cor; //adreça cor
long int c_ad_vel; //contador adreça velocitat
long int ad_vel; //adreça velocitat
long int c_ad_alt; //contador adreça altitut
long int ad_alt; //adreça altitut
long int c_ad_ped; //contador adreça cadencia de pedaleig
long int ad_ped; //adreça cadencia de pedaleig
int LOW_leer;    //llegir valor eeprom
int HIGH_leer; //llegir valor eeprom

//FUNCIONS
void LDR(); //lluminositat
void uv(); //rajos uva
void gas(); //contaminacio
void vibracio(); //vibracio

//FUNCIONS
//INTERRUPCIONS
//interrupcio rb2
#int_ext2
void ext2_rb2()
{
  c_vel++;
}

//interrupcio externa rb4
#int_rb
void ext_rb4567()
{
  if (input(PIN_B6) == 1) //boto start
  {
    bstart++;
    if (bstart == 4)
    {
      bstart = 0;
    }
  }
  if (input(PIN_B7) == 1) //boto seguent
  {
    boto++;
    if(boto == 6)
    {
      boto = 0;
    }
  }
  if ((input(PIN_B4) == 1) && (bstart == 1))
  {
    c_cor++;
  }
  if ((input(PIN_B5) == 1) && (bstart == 1))
  {
    c_peda++;
  }
```

```
}

//interrupcio bluetooth
#int_rda
void rda_isr()
{
  con = getc(); //rep senyal de coneccio de bluetooth
  start = getc(); //rep start
  tot = getc(); //rep valors totals
  fi = getc(); //rep fi
}

//interrupcio T1
#int_TIMER1
void TIMER1_isr()
{
  int16 puls;
  set_timer1(64285); //1ms
  time_vel++;
  if (time_vel >= 2000) //2s
  {
    //vel = c_vel*30*2*3.14*radi*0.06; //m/h
    //dist = ((c_vel*30*2*3.14*radi)/1000) + dist;//m
    //3.511 = 2*3.14*radi;
    puls = c_vel*30;
    dist = (puls*3.511)+ dist;
    vel = puls*3.511*0.06;
    pedal = c_peda * 30;
    c_peda = 0;
    c_vel = 0;
    time_vel = 0;
  }
}

//interrupcio T2
#int_TIMER3
void TIMER3_isr() //1ms
{
  set_timer3(64285);  //1ms
  t_sens++;
  time_cor++;
  if (time_cor >= 15000) //15s
  {
    fr_cor = c_cor*4;
    time_cor = 0;
    c_cor = 0;
  }
  if (t_sens >= 3000)
  {
    LDR();
    uv();
    gas();
    vibracio();
    t_sens = 0;
  }
}

//CONFIGURACIONS
//configuracio interrupcions
void conf_interrupcions()
{
  enable_interrupts(INT_TIMER1); //habilita interrupcions timer1
  enable_interrupts(INT_TIMER3); //hibilita interrupcions timer3
  enable_interrupts(INT_EXT2_L2H); //interrupcio rb2 low to high
  enable_interrupts(INT_RDA); //hanilita interrupció bluetooth
  enable_interrupts(INT_RB); //interrupcio rb4 i 5
  enable_interrupts(global); //habilita les interrupcions de forma global
  EXT_INT_EDGE(L_TO_H); //low to high
  set_timer1(64285); //valor a timer1 -- 1ms
  set_timer3(64285); //valor a timer3 -- 1ms
  setup_timer_1(RTCC_INTERNAL | RTCC_DIV_4); //configuracio timer1
  setup_timer_3(RTCC_INTERNAL | RTCC_DIV_4); //configuracio timer3
}

//desactiva interrupcions
void no_interrupcions()
{
  disable_interrupts(INT_TIMER0);
  disable_interrupts(INT_TIMER1);
  disable_interrupts(INT_TIMER3);
  disable_interrupts(INT_EXT2_L2H);
}

//SENSORS
//rtc
void rtc()
{
  ds1307_get_date(dia,mes,any,diaset);
  ds1307_get_time(hrs,min,sec);
  lcd_gotoxy(1,1);
  printf(lcd_putc,"%02u/%02u/20%02u-%1u  ",dia,mes,any,diaset);
  lcd_gotoxy(1,2);
  printf(lcd_putc,"%02d:%02d:%02d        ", hrs,min,sec);
}

//ldr
```

```
void LDR()
{
  float v_dig; //valor digital
  int16 v_ana; //valor analogic

  set_adc_channel(0); //habilitacio canal de lectura AN0
  delay_us(20); //estabilitzacio
  v_ana = read_adc();
  v_dig = 5.0*v_ana/1024.0;
  lum = (100*v_dig)/5; //lumens
  if (lum > 41)
  {
    output_low(PIN_C0);
  }
  else
  {
    output_high(PIN_C0);
  }
}

//sensor rajos uva -- critic 142
void uv()
{

  float v_dig; //valor digital
  int16 v_ana; //valor analogic

  set_adc_channel(1); //habilitacio canal de lectura AN0
  delay_us(20); //estabilitzacio
  v_ana = read_adc();
  v_dig = 5.0*v_ana/1024.0;
  uva = v_ana;
}

//sensor gas -- + de 400 critic
void gas()
{
  set_adc_channel(2);
  delay_us(20);
  co2 = read_adc();
  if (co2 > 400)
  {
    output_high(PIN_C1);
  }
  else
  {
    output_low(PIN_C1);
  }
```

```
}

//vibracio
void vibracio()
{
  int x;
  x = input(PIN_A3);
  if (x == 0)
  {
    output_high(PIN_C2);
    vibr = 1;
  }
  else
  {
    output_low(PIN_C2);
    vibr = 0;
  }
}

//Temps de recorregut
void time_rec()
{
  int32 temps_rec;
  temps_rec = temps_fi - temps_ini;
  hrs_rec = temps_rec/3600;
  min_rec = (temps_rec - (temps_rec/3600))/60;
  sec_rec = temps_rec - (hrs_rec*3600) - (min_rec*60);
}

//PORGRAMA PRINCIPAL
void main()
{
  //inicializaciones
  lcd_init(); //lcd
  ds1307_init(); //rtc
  setup_adc_ports(AN0_TO_AN2); //Pins AN0 a AN2 analogics
  setup_adc(ADC_CLOCK_INTERNAL); //rellorge del adc
  delay_ms(100);

  //interrupcions
  enable_interrupts(INT_RB); //interrupcio rb4, 5, 6, 7
  enable_interrupts(global); //habilita les interrupcions de forma global
  EXT_INT_EDGE(L_TO_H); //low to high

  //enviar data - linia només per la primera compilació
  //1 de maig 2020, divendres - 15:20:55
  //ds1307_set_date_time(dia,mes,any,diaset,hrs,min,sec);
  ds1307_set_date_time(26,5,20,7,9,21,55);
```

```
delay_ms(100);

//inicialitzacio variables
//temporitzadors
time_cor = 0; //timer de 15s
time_vel = 0; //timer 2s
t_sens = 0; //timer 3s
t_mem = 0; //timer 2s

//botons
boto = 0; //contador boto
bstart = 0; //estat bstart

//constants
radi = 0.559; //radi de la bici 559 mm
R = 8.31432; //constannt universal
u = 0.0289644; //massa molar de l'aire
g = 8.31432; //constant gravitacional
po = 1013.25; //pressio a nivell del mar (hPa)

//variables sensors
dist = 0; //distancia recorreguda
vel = 0; //velocitat
c_vel = 0; //contador velocitat
vel_mj = 0;
uva = 0; //rajos uva
lum = 0; //lluminositat
fr_cor = 0; //frecuencia cardiaca
fr_cor_mj = 0; //fr. cardiaca mitja
c_cor = 0;
co2 = 0; //CO2
pedal = 0; //cadencia de pedaleig
pedal_mj = 0;
c_peda = 0; //contador pedal
temperature = 0;; //temperatura
pressure = 0; //pressio
humidity = 0; //humitat
alt = 0; //altitut
alt_max = 0;
alt_min = 6000;
vibr = 0;

//temps
temps_ini = 0; //moment inici
temps_fi = 0; //moment fi

//memoria
ad_cor = 0; //adreça cor
```

```
ad_vel = 5462; //adreça velocitat
ad_alt = 16385; //adreça altitut
ad_ped = 27307; //adreça cadencia de pedaleig
c_ad_cor = 0; //contador adreça cor
c_ad_vel = 0; //contador adreça vlocitat
c_ad_alt = 0; //contador adreça altitut
c_ad_ped = 0; //contador adreça cadencia de pedaleig
LOW_leer = 0; //llegir valor eeprom
HIGH_leer = 0; //llegir valor eeprom

//bluetooth
con = "X";
start = "X";
tot = "X";

while (1)
{
  lcd_putc("\f"); //borra LCD
  rtc();
  delay_ms(2000);
  lcd_putc("\f");
  printf(lcd_putc,"Pulsa START");
  delay_ms(2000);
  if (con == "C")
  {
    while(con == "C")
    {
      if (start == "S")
      {
        bstart = 1;
        conf_interrupcions();
        ds1307_get_time(hrs,min,sec);
        temps_ini = (hrs*3600) + (min*60) + sec;

        while (start == "S")
        {
          //altitut
          printf(alt);
          printf(" m");
          printf("|");
          //contaminacio
          printf(co2);
          printf(" ppm");
          printf("|");
          //cadencia de pedaleig
          printf(pedal);
          printf(" pd/min");
          printf("|");
```

```
//distancia
printf(dist);
printf(" km");
printf("|");
//frequencia cardiaca
printf(fr_cor);
printf("|");
//humitat
printf(humidity);
printf(" %%");
printf("|");
//lluminositat
printf(lum);
printf(" %%");
printf("|");
//temperatura
printf(temperature);
printf(" ºC");
printf("|");
//rajos uva
printf(uva);
printf("|");
//velocitat
printf(vel);
printf(" km/h");

//indicadors
if (vibr == 1)
{
    printf("V");
}
else if (co2 > 400)
{
 printf("O");
}
else if (lum > 41)
{
  printf("L");
}

//valors totals
if (tot == "T")
{
  no_interrupcions();
  bstart = 0;
  //altitut maxima
  printf(alt_max);
  printf(" m");
  printf("|");
  //altitut minima
  printf(alt_min);
  printf(" m");
  printf("|");
  //cadencia de pedaleig mitja
  printf(pedal_mj);
  printf(" pd/min");
  printf("|");
  //distancia maxima
  printf(dist);
  printf(" Km");
  printf("|");
  //frequencia cardiaca mitja
  printf(fr_cor_mj);
  printf("|");
  //temps de recorregut
  printf(hrs_rec);
  printf(":");
  printf(min_rec);
  printf(":");
  printf(sec_rec);
  printf("|");
  //velocitat mitja
  printf(vel_mj);
  printf(" Km/h");
}
//fi
if (fi == "F")
{
  //desactivació
  con = "X";
  start = "X";
  tot = "X";
  fi = "X";

  lcd_putc("/f");
  //temps
  time_cor = 0; //timer de 15s
  time_vel = 0; //timer 2s
  t_sens = 0; //timer 3s
  t_mem = 0; //timer 2s

  //botons
  bstart = 0;
  boto = 0;

  //variables sensors
```

```
          dist = 0; //distancia recorreguda
          vel = 0; //velocitat
          c_vel = 0; //contador velocitat
          vel_mj = 0;
          uva = 0; //rajos uva
          lum = 0; //lluminositat
          fr_cor = 0; //frecuencia cardiaca
          fr_cor_mj = 0;
          c_cor = 0;
          co2 = 0; //CO2
          pedal = 0; //cadencia de pedaleig
          pedal_mj = 0;
          c_peda = 0; //contador pedal
          temperature = 0;; //temperatura
          pressure = 0; //pressio
          humidity = 0; //humitat
          alt = 0; //altitut
          alt_max = 0;
          alt_min = 6000;
          vibr = 0; //vibracio

          //temps
          temps_ini = 0; //moment inici
          temps_fi = 0; //moment fi

          //memoria
          ad_cor = 0; //adreça cor
          ad_vel = 5462; //adreça velocitat
          ad_alt = 16385; //adreça altitut
          ad_ped = 27307; //adreça cadencia de pedaleig
          c_ad_cor = 0; //contador adreça cor
          c_ad_vel = 0; //contador adreça vlocitat
          c_ad_alt = 0; //contador adreça altitut
          c_ad_ped = 0; //contador adreça cadencia de pedaleig
          LOW_leer = 0; //llegir valor eeprom
          HIGH_leer = 0; //llegir valor eeprom

        }
      }
    }
  }
}

else
{
  while (bstart != 0)
  {
    if(bstart == 1)
```

```
{
  lcd_putc("\f");
  rtc();
  temps_ini = (hrs*3600) + (min*60) + sec;

  //comença lectura sensors
  conf_interrupcions();
  while (bstart == 1)
  {
    switch(boto)
    {
      case(1):
      //pulsacions
      lcd_gotoxy(1,1);
      printf(lcd_putc, "Fr.card =  %3u  ", fr_cor);

      //velocitat
      lcd_gotoxy(1,2);
      printf(lcd_putc, "Vel = %4Lu m/h   ", vel);
      break;

      case(3):
      //uv
      lcd_gotoxy(1,1);
      printf(lcd_putc, "UV-A = %4Lu     ", uva);

      //llum
      lcd_gotoxy(1,2);
      printf(lcd_putc, "Lumens = %3.0d%%  ", lum);
      break;

      case(4):

      //distancia
      lcd_gotoxy(1,2);
      printf(lcd_putc, "Dist = %5Lu m  ", dist);
      break;

      case(5):
      //contaminacio
      lcd_gotoxy(1,1);
      printf(lcd_putc, "CO2 = %4Lu ppm  ", co2);

      //cadencia de pedaleig
      lcd_gotoxy(1,2);
      printf(lcd_putc, "C.p = %3u pd/min", pedal);
      break;
      default:
      rtc();
      break;
    }
  }
}
if(bstart == 2)
{
  ds1307_get_time(hrs,min,sec);
  temps_fi = (hrs*3600) + (min*60) + sec;
  no_interrupcions();
  time_rec();
  lcd_putc("\f");
  while (bstart == 2)
  {
    switch (boto)
    {
      case(1):
      //ditsnacia total
      lcd_gotoxy(1,1);
      printf(lcd_putc, "Dist.t = %5Lu m", dist);
      break;

      case(4):
      //temps de recorregut
      lcd_gotoxy(1,1);
      printf(lcd_putc,"Temps Recorregut");
      lcd_gotoxy(1,2);
      printf(lcd_putc,"%02d:%02d:%02d        ", hrs_rec,min_rec,sec_rec);
      break;
    }
  }
}
if (bstart == 3)
{
  lcd_putc("\f"); //borra LCD
  //temporitzadors
  time_cor = 0; //timer de 15s
  time_vel = 0; //timer 2s
  t_sens = 0; //timer 3s
  t_mem = 0; //timer 2s

  //botons
  bstart = 0;
  boto = 0;

  //variables sensors
  dist = 0; //distancia recorreguda
```

```
            vel = 0; //velocitat
            c_vel = 0; //contador velocitat
            vel_mj = 0;
            uva = 0; //rajos uva
            lum = 0; //lluminositat
            fr_cor = 0; //frecuencia cardiaca
            fr_cor_mj = 0;
            c_cor = 0;
            co2 = 0; //CO2
            pedal = 0; //cadencia de pedaleig
            pedal_mj = 0;
            c_peda = 0; //contador pedal
            temperature = 0;; //temperatura
            pressure = 0; //pressio
            humidity = 0; //humitat
            alt = 0; //altitut
            alt_max = 0;
            alt_min = 6000;
            vibr = 0; //vibracio

            //temps
            temps_ini = 0; //moment inici
            temps_fi = 0; //moment fi

            //memoria
            ad_cor = 0; //adreça cor
            ad_vel = 5462; //adreça velocitat
            ad_alt = 16385; //adreça altitut
            ad_ped = 27307; //adreça cadencia de pedaleig
            c_ad_cor = 0; //contador adreça cor
            c_ad_vel = 0; //contador adreça vlocitat
            c_ad_alt = 0; //contador adreça altitut
            c_ad_ped = 0; //contador adreça cadencia de pedaleig
            LOW_leer = 0; //llegir valor eeprom
            HIGH_leer = 0; //llegir valor eeprom
         }
      }
    }
  }
}
```

4. Llibreries

4.1. BME/BMP280

```
////////////////////////////////////////////////////////////////////////
////                                              ////
////                    BME280_Lib.c                      ////
////                                              ////
////              Driver for CCS C compiler                  ////
////                                              ////
//// Driver for Bosch BME280 sensor. This sensor can read temperature, ////
//// humidity and pressure.                                ////
//// This driver only supports I2C mode, it doesn't support SPI mode.  ////
////                                              ////
////////////////////////////////////////////////////////////////////////
////                                              ////
////               https://simple-circuit.com/                ////
////                                              ////
////////////////////////////////////////////////////////////////////////


#include <stdint.h>
#use I2C(MASTER, sda=PIN_B0,scl=PIN_B1, STREAM = BME280_STREAM) //configuracio i2c

#ifndef BME280_I2C_ADDRESS
  #define BME280_I2C_ADDRESS  0x76
#endif

#define BME280_CHIP_ID       0x60

#define BME280_REG_DIG_T1    0x88
#define BME280_REG_DIG_T2    0x8A
#define BME280_REG_DIG_T3    0x8C

#define BME280_REG_DIG_P1    0x8E
#define BME280_REG_DIG_P2    0x90
#define BME280_REG_DIG_P3    0x92
#define BME280_REG_DIG_P4    0x94
#define BME280_REG_DIG_P5    0x96
#define BME280_REG_DIG_P6    0x98
#define BME280_REG_DIG_P7    0x9A
#define BME280_REG_DIG_P8    0x9C
#define BME280_REG_DIG_P9    0x9E

#define BME280_REG_DIG_H1    0xA1
#define BME280_REG_DIG_H2    0xE1
```

```
#define BME280_REG_DIG_H3    0xE3
#define BME280_REG_DIG_H4    0xE4
#define BME280_REG_DIG_H5    0xE5
#define BME280_REG_DIG_H6    0xE7

#define BME280_REG_CHIPID    0xD0
#define BME280_REG_SOFTRESET  0xE0

#define BME280_REG_CTRLHUM   0xF2
#define BME280_REG_STATUS    0xF3
#define BME280_REG_CONTROL   0xF4
#define BME280_REG_CONFIG    0xF5
#define BME280_REG_PRESS_MSB  0xF7

int32_t adc_T, adc_P, adc_H, t_fine;

// BME280 sensor modes, register ctrl_meas mode[1:0]
enum bme280_mode
{
  MODE_SLEEP  = 0x00,  // sleep mode
  MODE_FORCED = 0x01,  // forced mode
  MODE_NORMAL = 0x03   // normal mode
} ;

// oversampling setting. osrs_h[2:0], osrs_t[2:0], osrs_p[2:0]
enum bme280_sampling
{
  SAMPLING_SKIPPED = 0x00,  //skipped, output set to 0x80000 (0x8000 for humidity)
  SAMPLING_X1      = 0x01,  // oversampling x1
  SAMPLING_X2      = 0x02,  // oversampling x2
  SAMPLING_X4      = 0x03,  // oversampling x4
  SAMPLING_X8      = 0x04,  // oversampling x8
  SAMPLING_X16     = 0x05   // oversampling x16
} ;

// filter setting filter[2:0]
enum bme280_filter
{
  FILTER_OFF = 0x00,  // filter off
  FILTER_2   = 0x01,  // filter coefficient = 2
  FILTER_4   = 0x02,  // filter coefficient = 4
  FILTER_8   = 0x03,  // filter coefficient = 8
  FILTER_16  = 0x04   // filter coefficient = 16
} ;

// standby (inactive) time in ms (used in normal mode), t_sb[2:0]
enum standby_time
{
  STANDBY_0_5   =  0x00,  // standby time = 0.5 ms
  STANDBY_62_5  =  0x01,  // standby time = 62.5 ms
  STANDBY_125   =  0x02,  // standby time = 125 ms
  STANDBY_250   =  0x03,  // standby time = 250 ms
  STANDBY_500   =  0x04,  // standby time = 500 ms
  STANDBY_1000  =  0x05,  // standby time = 1000 ms
  STANDBY_10    =  0x06,  // standby time = 10 ms
  STANDBY_20    =  0x07   // standby time = 20 ms
} ;

struct
{
  uint16_t dig_T1;
  int16_t  dig_T2;
  int16_t  dig_T3;
  uint16_t dig_P1;
  int16_t  dig_P2;
  int16_t  dig_P3;
  int16_t  dig_P4;
  int16_t  dig_P5;
  int16_t  dig_P6;
  int16_t  dig_P7;
  int16_t  dig_P8;
  int16_t  dig_P9;

  uint8_t  dig_H1;
  int16_t  dig_H2;
  uint8_t  dig_H3;
  int16_t  dig_H4;
  int16_t  dig_H5;
  int8_t   dig_H6;
} BME280_calib;

// writes 1 byte '_data' to register 'reg_addr'
void BME280_Write(uint8_t reg_addr, uint8_t _data)
{
  I2C_Start(BME280_STREAM);
  I2C_Write(BME280_STREAM, BME280_I2C_ADDRESS);
  I2C_Write(BME280_STREAM, reg_addr);
  I2C_Write(BME280_STREAM, _data);
  I2C_Stop(BME280_STREAM);
}

// reads 8 bits from register 'reg_addr'
uint8_t BME280_Read8(uint8_t reg_addr)
{
  uint8_t ret;
  I2C_Start(BME280_STREAM);
  I2C_Write(BME280_STREAM, BME280_I2C_ADDRESS);
  I2C_Write(BME280_STREAM, reg_addr);
  I2C_Start(BME280_STREAM);
  I2C_Write(BME280_STREAM, BME280_I2C_ADDRESS | 1);
  ret = I2C_Read(BME280_STREAM, 0);
  I2C_Stop(BME280_STREAM);

  return ret;
}

// reads 16 bits from register 'reg_addr'
uint16_t BME280_Read16(uint8_t reg_addr)
{
  union
  {
    uint8_t  b[2];
    uint16_t w;
  } ret;

  I2C_Start(BME280_STREAM);
  I2C_Write(BME280_STREAM, BME280_I2C_ADDRESS);
  I2C_Write(BME280_STREAM, reg_addr);
  I2C_Start(BME280_STREAM);
  I2C_Write(BME280_STREAM, BME280_I2C_ADDRESS | 1);
  ret.b[0] = I2C_Read(BME280_STREAM, 1);
  ret.b[1] = I2C_Read(BME280_STREAM, 0);
  I2C_Stop(BME280_STREAM);

  return(ret.w);
}

// BME280 sensor configuration function
void BME280_Configure(bme280_mode mode, bme280_sampling T_sampling, bme280_sampling H_sampling,
                      bme280_sampling P_sampling, bme280_filter filter, standby_time standby)
{
  uint8_t _ctrl_hum, _ctrl_meas, _config;

  _ctrl_hum = H_sampling;
  _config = ((standby << 5) | (filter << 2)) & 0xFC;
  _ctrl_meas = (T_sampling << 5) | (P_sampling << 2) | mode;

  BME280_Write(BME280_REG_CTRLHUM, _ctrl_hum);
  BME280_Write(BME280_REG_CONFIG,  _config);
  BME280_Write(BME280_REG_CONTROL, _ctrl_meas);
}
```

```
// initializes the BME280 sensor, returns 1 if OK and 0 if error
int1 BME280_begin(bme280_mode mode,
             bme280_sampling T_sampling = SAMPLING_X1,
             bme280_sampling H_sampling = SAMPLING_X1,
             bme280_sampling P_sampling = SAMPLING_X1,
             bme280_filter filter      = FILTER_OFF,
             standby_time  standby     = STANDBY_0_5)
{
  if(BME280_Read8(BME280_REG_CHIPID) != BME280_CHIP_ID)
    return 0;

  // reset the BME280 with soft reset
  BME280_Write(BME280_REG_SOFTRESET, 0xB6);
  delay_ms(100);

  // if NVM data are being copied to image registers, wait 100 ms
  while( (BME280_Read8(BME280_REG_STATUS) & 0x01) == 0x01 )
    delay_ms(100);

  BME280_calib.dig_T1 = BME280_Read16(BME280_REG_DIG_T1);
  BME280_calib.dig_T2 = BME280_Read16(BME280_REG_DIG_T2);
  BME280_calib.dig_T3 = BME280_Read16(BME280_REG_DIG_T3);

  BME280_calib.dig_P1 = BME280_Read16(BME280_REG_DIG_P1);
  BME280_calib.dig_P2 = BME280_Read16(BME280_REG_DIG_P2);
  BME280_calib.dig_P3 = BME280_Read16(BME280_REG_DIG_P3);
  BME280_calib.dig_P4 = BME280_Read16(BME280_REG_DIG_P4);
  BME280_calib.dig_P5 = BME280_Read16(BME280_REG_DIG_P5);
  BME280_calib.dig_P6 = BME280_Read16(BME280_REG_DIG_P6);
  BME280_calib.dig_P7 = BME280_Read16(BME280_REG_DIG_P7);
  BME280_calib.dig_P8 = BME280_Read16(BME280_REG_DIG_P8);
  BME280_calib.dig_P9 = BME280_Read16(BME280_REG_DIG_P9);

  BME280_calib.dig_H1 = BME280_Read8(BME280_REG_DIG_H1);
  BME280_calib.dig_H2 = BME280_Read16(BME280_REG_DIG_H2);
  BME280_calib.dig_H3 = BME280_Read8(BME280_REG_DIG_H3);
  BME280_calib.dig_H4 = ((uint16_t)BME280_Read8(BME280_REG_DIG_H4) << 4) |
(BME280_Read8(BME280_REG_DIG_H4 + 1) & 0x0F);
  if (BME280_calib.dig_H4 & 0x0800)   // if BME280_calib.dig_H4 < 0
    BME280_calib.dig_H4 |= 0xF000;
  BME280_calib.dig_H5 = ((uint16_t)BME280_Read8(BME280_REG_DIG_H5 + 1) << 4) |
(BME280_Read8(BME280_REG_DIG_H5) >> 4);
  if (BME280_calib.dig_H5 & 0x0800)   // if BME280_calib.dig_H5 < 0
    BME280_calib.dig_H5 |= 0xF000;
  BME280_calib.dig_H6 = BME280_Read8(BME280_REG_DIG_H6);

  BME280_Configure(mode, T_sampling, H_sampling, P_sampling, filter, standby);
  return 1;
}

// takes a new measurement, for forced mode only!
// Returns 1 if ok and 0 if error (sensor is not in sleep mode)
int1 BME280_ForcedMeasurement()
{
  uint8_t ctrl_meas_reg = BME280_Read8(BME280_REG_CONTROL);

  if ( (ctrl_meas_reg & 0x03) != 0x00 )
    return 0;   // sensor is not in sleep mode

  // set sensor to forced mode
  BME280_Write(BME280_REG_CONTROL, ctrl_meas_reg | 1);
  // wait for conversion complete
  while (BME280_Read8(BME280_REG_STATUS) & 0x08)
    delay_ms(1);

  return 1;
}

// read (updates) adc_P, adc_T and adc_H from BME280 sensor
void BME280_Update()
{
  union
  {
    uint8_t  b[4];
    uint32_t dw;
  } ret;
  ret.b[3] = 0x00;

  I2C_Start(BME280_STREAM);
  I2C_Write(BME280_STREAM, BME280_I2C_ADDRESS);
  I2C_Write(BME280_STREAM, BME280_REG_PRESS_MSB);
  I2C_Start(BME280_STREAM);
  I2C_Write(BME280_STREAM, BME280_I2C_ADDRESS | 1);
  ret.b[2] = I2C_Read(BME280_STREAM, 1);
  ret.b[1] = I2C_Read(BME280_STREAM, 1);
  ret.b[0] = I2C_Read(BME280_STREAM, 1);

  adc_P = (ret.dw >> 4) & 0xFFFFF;

  ret.b[2] = I2C_Read(BME280_STREAM, 1);
  ret.b[1] = I2C_Read(BME280_STREAM, 1);
  ret.b[0] = I2C_Read(BME280_STREAM, 1);

  adc_T = (ret.dw >> 4) & 0xFFFFF;
  ret.b[2] = 0x00;
  ret.b[1] = I2C_Read(BME280_STREAM, 1);
  ret.b[0] = I2C_Read(BME280_STREAM, 0);
  I2C_Stop(BME280_STREAM);

  adc_H = ret.dw & 0xFFFF;
}

// Reads temperature from BME280 sensor.
// Temperature is stored in hundredths C (output value of "5123" equals 51.23 DegC).
// Temperature value is saved to *temp, returns 1 if OK and 0 if error.
int1 BME280_readTemperature(int32_t *temp)
{
  int32_t var1, var2;

  BME280_Update();

  // calculate temperature
  var1 = ((((adc_T / 8) - ((int32_t)BME280_calib.dig_T1 * 2))) *
        ((int32_t)BME280_calib.dig_T2)) / 2048;

  var2 = (((((adc_T / 16) - ((int32_t)BME280_calib.dig_T1)) *
        ((adc_T / 16) - ((int32_t)BME280_calib.dig_T1))) / 4096) *
        ((int32_t)BME280_calib.dig_T3)) / 16384;

  t_fine = var1 + var2;

  *temp = (t_fine * 5 + 128) / 256;

  return 1;
}

// Reads humidity from BME280 sensor.
// Humidity is stored in relative humidity percent in 1024 steps
// (output value of "47445" represents 47445/1024 = 46.333 %RH).
// Humidity value is saved to *humi, returns 1 if OK and 0 if error.
int1 BME280_readHumidity(uint32_t *humi)
{
  int32_t v_x1_u32r;
  uint32_t H;

  v_x1_u32r = (t_fine - ((int32_t)76800));

  v_x1_u32r = (((((adc_H * 16384) - (((int32_t)BME280_calib.dig_H4) * 1048576) -
(((int32_t)BME280_calib.dig_H5) * v_x1_u32r)) +
     ((int32_t)16384)) / 32768) * (((((((v_x1_u32r * ((int32_t)BME280_calib.dig_H6)) / 1024) *
(((v_x1_u32r *
     ((int32_t)BME280_calib.dig_H3)) / 2048) + ((int32_t)32768))) / 1024) + ((int32_t)2097152)) *
```

```
    ((int32_t)BME280_calib.dig_H2) + 8192) / 16384));

  v_x1_u32r = (v_x1_u32r - (((((v_x1_u32r / 32768) * (v_x1_u32r / 32768)) / 128) *
((int32_t)BME280_calib.dig_H1)) / 16));
  v_x1_u32r = (v_x1_u32r < 0 ? 0 : v_x1_u32r);
  v_x1_u32r = (v_x1_u32r > 419430400 ? 419430400 : v_x1_u32r);

  H = (uint32_t)(v_x1_u32r / 4096);
  *humi = H;

  return 1;
}

// Reads pressure from BME280 sensor.
// Pressure is stored in Pa (output value of "96386" equals 96386 Pa = 963.86 hPa).
// Pressure value is saved to *pres, returns 1 if OK and 0 if error.
int1 BME280_readPressure(uint32_t *pres)
{
  int32_t var1, var2;
  uint32_t p;

  // calculate pressure
  var1 = (((int32_t)t_fine) / 2) - (int32_t)64000;
  var2 = (((var1/4) * (var1/4)) / 2048 ) * ((int32_t)BME280_calib.dig_P6);

  var2 = var2 + ((var1 * ((int32_t)BME280_calib.dig_P5)) * 2);
  var2 = (var2/4) + (((int32_t)BME280_calib.dig_P4) * 65536);

  var1 = ((((int32_t)BME280_calib.dig_P3 * (((var1/4) * (var1/4)) / 8192 )) / 8) +
        ((((int32_t)BME280_calib.dig_P2) * var1)/2)) / 262144;

  var1 =((((32768 + var1)) * ((int32_t)BME280_calib.dig_P1)) / 32768);

  if (var1 == 0)
    return 0; // avoid exception caused by division by zero

  p = (((uint32_t)(((int32_t)1048576) - adc_P) - (var2 / 4096))) * 3125;

  if (p < 0x80000000)
    p = (p * 2) / ((uint32_t)var1);

  else
    p = (p / (uint32_t)var1) * 2;

  var1 = (((int32_t)BME280_calib.dig_P9) * ((int32_t)(((p/8) * (p/8)) / 8192))) / 4096;
  var2 = (((int32_t)(p/4)) * ((int32_t)BME280_calib.dig_P8)) / 8192;

  p = (uint32_t)((int32_t)p + ((var1 + var2 + (int32_t)BME280_calib.dig_P7) / 16));
  *pres = p;

  return 1;
}

// end of code.
```

4.2. DS1307

```
////////////////////////////////////////////////////////////////////////////
///                          DS1307.C                              ///
///                  Driver for Real Time Clock                    ///
///                                                    ///
/// ds1307_init() - Enable oscillator without clearing the seconds register -///
///               used when PIC loses power and DS1307 run from 3V BAT    ///
///             - Disable squarewave output                        ///
///                                                    ///
/// ds1307_set_date_time(day,mth,year,dow,hour,min,sec)  Set the date/time   ///
///                                                    ///
/// ds1307_get_date(day,mth,year,dow)            Get the date           ///
///                                                    ///
/// ds1307_get_time(hr,min,sec)                  Get the time           ///
///                                                    ///
////////////////////////////////////////////////////////////////////////////

#define RTC_SDA  PIN_B0
#define RTC_SCL  PIN_B1

#use i2c(master, sda=RTC_SDA, scl=RTC_SCL)

BYTE bin2bcd(BYTE binary_value);
BYTE bcd2bin(BYTE bcd_value);

void ds1307_init(void)
{
  BYTE seconds = 0;

  i2c_start();
  i2c_write(0xD0);      // WR to RTC
  i2c_write(0x00);      // REG 0
  i2c_start();
  i2c_write(0xD1);      // RD from RTC
  seconds = bcd2bin(i2c_read(0)); // Read current "seconds" in DS1307
  i2c_stop();
  seconds &= 0x7F;

  delay_ms(1);

  i2c_start();
  i2c_write(0xD0);      // WR to RTC
  i2c_write(0x00);      // REG 0
  i2c_write(bin2bcd(seconds));     // Start oscillator with current "seconds value
  i2c_start();
  i2c_write(0xD0);      // WR to RTC
  i2c_write(0x07);      // Control Register
  i2c_write(0x80);    // Disable squarewave output pin
  i2c_stop();

}

void ds1307_set_date_time(BYTE day, BYTE mth, BYTE year, BYTE dow, BYTE hr, BYTE min, BYTE sec)
{
  sec &= 0x7F;
  hr &= 0x3F;

  i2c_start();
  i2c_write(0xD0);            // I2C write address
  i2c_write(0x00);            // Start at REG 0 - Seconds
  i2c_write(bin2bcd(sec));      // REG 0
  i2c_write(bin2bcd(min));      // REG 1
  i2c_write(bin2bcd(hr));      // REG 2
  i2c_write(bin2bcd(dow));      // REG 3
  i2c_write(bin2bcd(day));      // REG 4
  i2c_write(bin2bcd(mth));      // REG 5
  i2c_write(bin2bcd(year));      // REG 6
  i2c_write(0x80);            // REG 7 - Disable squarewave output pin
  i2c_stop();
}

void ds1307_get_date(BYTE &day, BYTE &mth, BYTE &year, BYTE &dow)
{
  i2c_start();
  i2c_write(0xD0);
  i2c_write(0x03);            // Start at REG 3 - Day of week
  i2c_start();
  i2c_write(0xD1);
  dow  = bcd2bin(i2c_read() & 0x7f);   // REG 3
  day  = bcd2bin(i2c_read() & 0x3f);   // REG 4
  mth  = bcd2bin(i2c_read() & 0x1f);   // REG 5
  year = bcd2bin(i2c_read(0));            // REG 6
  i2c_stop();
}

void ds1307_get_time(BYTE &hr, BYTE &min, BYTE &sec)
{
  i2c_start();
  i2c_write(0xD0);
  i2c_write(0x00);            // Start at REG 0 - Seconds
  i2c_start();
  i2c_write(0xD1);
  sec = bcd2bin(i2c_read() & 0x7f);
  min = bcd2bin(i2c_read() & 0x7f);
  hr  = bcd2bin(i2c_read(0) & 0x3f);
  i2c_stop();

}

BYTE bin2bcd(BYTE binary_value)
{
  BYTE temp;
  BYTE retval;

  temp = binary_value;
  retval = 0;

  while(1)
  {
    // Get the tens digit by doing multiple subtraction
    // of 10 from the binary value.
    if(temp >= 10)
    {
      temp -= 10;
      retval += 0x10;
    }
    else // Get the ones digit by adding the remainder.
    {
      retval += temp;
      break;
    }
  }

  return(retval);
}


// Input range - 00 to 99.
BYTE bcd2bin(BYTE bcd_value)
{
  BYTE temp;

  temp = bcd_value;
  // Shifting upper digit right by 1 is same as multiplying by 8.
  temp >>= 1;
  // Isolate the bits for the upper digit.
  temp &= 0x78;

  // Now return: (Tens * 8) + (Tens * 2) + Ones

  return(temp + (temp >> 2) + (bcd_value & 0x0f));
}
```