```
//  Este programa esta  dirigido a clientes de FADISHOP que hayan adquirido
//  la tarjeta FADICLOCK en alguna de sus versiones  http://www.fadishop.eu
//  Este programa sirve de referencia para realizar ensayos y pruebas. ARDUINO UNO REV2
//  Para que resulte pra¡ctico no esta¡ optimizado. Se trabaja a nivel de byte i bit.
//  LECTURA PERIODICA DE TEMPERATURA Y MEMORIZA EN SRAM LOS PROMEDIOS
//  Proximamente sera¡ revisado y mejorado. ARDUINO ALPHA VERSION 0023.
//  Por favor no suprima estas lineas y annada su nombre o entidad debajo.
//  AUTOR ORIGINAL:  LLEONARD GARCIA I LLOP  2 Mayo 2012  http://www.fadishop.eu
//                                           10 SET 2012 (revisado, en revision)
//  AUTORITZATS:     usted
#include "Wire.h"
// Variables generals

int I2C_DS3232=0xD0 >> 1;      // AdreÃ§a perifÃ¨ric I2C sense LSB(read/write).
byte DS3232_00, DS3232_01, DS3232_02, DS3232_03, DS3232_04, DS3232_05, DS3232_06;
byte DS3232_07, DS3232_08, DS3232_09, DS3232_0A;
byte DS3232_0B, DS3232_0C, DS3232_0D;
byte DS3232control=0b00011100, DS3232status=0b11001000 ,DS3232aging;
int DS3232Msb ,DS3232Lsb;
int command = 0;        // This is the command char, in ascii form, sent from the serial port
int z=0;
int bb=0;
byte restaurar_alarmes=0;
bool bufferAcmpleto=0;
//////////////////////////////////////////////////////////////////////////////////
#define punteros 0x14            // @ de la SRAM donde estÃ¡n los punteros.
#define adr_min  0x30            // @ de la SRAM on empieza el buffer_min
byte buffer_min[60];              // DimensiÃ³n del buffer de minutos.
byte i_min=0;                     // Puntero corriente del buffer de minutos.
byte x_min=59;                    // Valor mÃ ximo del buffer de minutos.
byte DS3232min;
byte punteralia[3][3]={0x20,0x14,0x3C,0x9A,0x15,0x18,0xE8,0x16,0x1E};// Mapa [0.-@buffer,1.-@punter,2.-@limit][area]
int puntero;
byte sample[2]={0x00,0x00};
byte buffer[256];
int  i,j,k,zz,hh,index;
byte BuffSegundos[300], SegMaxMsb, SegMaxLsb, SegMaxTim[6], SegMinMsb, SegMinLsb, SegMinTim[6];
byte SegPromMsb, SegPromLsb;
byte Rel, Abs;
byte scan[60];
```

```
//byte PromMsb, PromLsb;
byte FinalMinuto;
int xSeg=0;
int PromMsb=0;
int PromLsb=0;
////////////////////////////////////////////////////////////////////////////////
void EntrarMuestra(byte Msb, byte Lsb, int index){
//       byte zona=punteralia[index][0];          // 0.-Absoluto  @0x30    0x6C
//       byte puntero=punteralia[index][1];       // 1.-Relativo  @0x14    0x15
//       byte limite=punteralia[index][2];        // 2.-ParÃ metro 60       60

       grabarSramTest(Msb, Lsb, index);      // (dato1, dato2, direcciÃ³n, maximo)
}
////////////////////////////////////////////////////////////////////////////////
void grabarSramTest(byte Msb, byte Lsb, int index){
       //  Leer el lugar donde deben guardarse los 2bytes de la muestra.
       byte absoluto=punteralia[index][0];
       byte relativo=Readbyte(punteralia[index][1]);//(Dato leido, direcciÃ³n donde leer)
       relativo=relativo+2;
       if (relativo >= punteralia[index][2]){
              relativo=0;
//              promediar();
       }
       WriteByte(relativo, punteralia[index][1]);    //Desa la nova adreÃ§a
       bb=relativo+absoluto;
       //  Guardar los 2 bytes de la muestra
       Write2Byte(Msb, Lsb, bb);                // (Dato1, Dato2, direcciÃ³n donde guardar)
       //  Actualizar el puntero para las nuevas muestras

       if (relativo >=punteralia[index][2])
         {
           //aa=0;
           bool bufferAcmpleto=true;
         }
       //WriteByte(aa, puntero);                // (Dato1, Dato2, direcciÃ³n donde guardar)
}
/*
void promediar()
{
                   for (zz=0; zz<60; zz++){
```

```
//  leer punteralia[index][2] lectures
              Wire.beginTransmission(I2C_DS3232);          // START + adreÃ§a perifÃ©ric I2C.
              Wire.send(punteralia[index][0]);                              // AdreÃ§a interna
              Wire.endTransmission();                      // STOP
              Wire.requestFrom(I2C_DS3232, punteralia[index][2]);             // RESTART + @esclau +7bytes + STOP
              for (i=0; i<punteralia[index][2]; i++){
                    buffer[(i)]=Wire.receive();
                    }
              mitjana=0;
              for (i=0; i<punteralia[index][2]; i++){
                    mitjana=buffer[(i)]+mitjana;
                    }
    //desar mitjana
                        Serial.print(buffer[(zz)],DEC);
                        Serial.print(" ");
                        }
}*/
////////////////////////////////////////////////////////////////////////////////
byte Readbyte(byte address3){           // (Dato leido, direcciÃ³n donde leer)
       Wire.beginTransmission(I2C_DS3232);          // START + @perifÃ©rico I2C.
       Wire.send(address3);                          // DirecciÃ³n interna. Definido por la variable lugar.
       Wire.endTransmission();                      // STOP
       Wire.requestFrom(I2C_DS3232,1);              // RESTART + @perifÃ©rico + 1 byte + STOP
       return (Wire.receive());
}
////////////////////////////////////////////////////////////////////////////////
void WriteByte(byte data_a, byte address1)
{
       Wire.beginTransmission(I2C_DS3232);          // START + adreÃ§a perifÃ©ric I2C.
       Wire.send(address1);                         // DirecciÃ³n interna. Definido por la variable lugar.
       Wire.send(data_a);                           // Byte alto almacenado.
       Wire.endTransmission();                      // STOP.
}
////////////////////////////////////////////////////////////////////////////////
//void Write2Byte(byte data_b, byte data_c, byte address2){    // (Dato1, Dato2, direcciÃ³n donde guardar)
void Write2Byte(byte data_b,byte data_c,byte address2)
{
       Wire.beginTransmission(I2C_DS3232);          // START + adreÃ§a perifÃ©ric I2C.
       Wire.send(address2);                          // DirecciÃ³n interna. Definido por la variable lugar.
       Wire.send(data_b);                           // Byte alto almacenado.
```

```
      Wire.send(data_c);                          // Byte bajo almacenado.
      Wire.endTransmission();                     // STOP.
}
////////////////////////////////////////////////////////////////////////////////////////



//  CONFIGURACIÃ" HORA EN CLAR   ////////////////////////////////////////////////////
byte SegonsRtc=40;              // RTC: segons en decimal.
byte MinutsRtc=59;              // RTC: minuts en decimal.
byte HoraRtc=23;                // RTC: hora en decimal.
byte DiaMesRtc=31;              // RTC: Dia mensual en decimal.
byte MesRtc=12;                  // RTC: Mes en decimal.
byte AnyRtc=99;                 // RTC: Any en decimal.
byte DiaSemRtc=1;               // RTC: Dia setmanal en decimal.
bool ModeAmpmRtc=false;         // RTC: false:24h, true:12h PM-AM.
////////////////////////////////////////////////////////////////////////////////////////
//  CONFIGURACIÃ" DE L'ALARMA_1 EN CLAR   ////////////////////////////////////////////
byte SegonsAlarma1=10;          // Alarma1: segons en decimal.
byte MinutsAlarma1=31;          // Alarma1: minuts en decimal.
byte HoraAlarma1=18;            // Alarma1: hora en decimal.
byte DiaMesAlarma1=8;          // Alarma1: Dia mensual en decimal.
byte MesAlarma1=5;              // Alarma1: Mes en decimal.
byte DiaSemAlarma1=1;           // Alarma1: Dia setmanal en decimal.
bool ModeDiaAlarma1=true;      // Alarma1: false:dia mensual, true:dia setmanal.
bool ModeAmpmAlarma1=false;     // Alarma1: false:24h, true:12h PM-AM.
int reactibleAlarma1=0;         // Alarma1: reactibilitat: (0 un cop en data/hora/minut/segon).
                                // Alarma1: reactibilitat: (1 cada hora/minut/segon).
                                // Alarma1: reactibilitat: (2 cada en minut/segon).
                                // Alarma1: reactibilitat: (3 cada en segons).
//////////////////////////////// Alarma1: reactibilitat: (4 generador de seÃ±al de 1 segon).
////////////////////////////////////////////////////////////////////////////////////////
//  CONFIGURACIÃ" DE L'ALARMA_2 EN CLAR   ////////////////////////////////////////////
byte MinutsAlarma2=31;           // Alarma2: minuts en decimal.
byte HoraAlarma2=19;             // Alarma2: hora en decimal.
byte DiaMesAlarma2=8;           // Alarma2: Dia mensual en decimal.
byte MesAlarma2=05;              // Alarma2: Mes en decimal.
byte DiaSemAlarma2=01;           // Alarma2: Dia setmanal en decimal.
bool ModeDiaAlarma2=true;      // Alarma2: false:dia mensual, true:dia setmanal.
```

```
bool ModeAmpmAlarma2=false;      // Alarma2: false:24h, true:12h PM-AM.
int reactibleAlarma2=3;          // Alarma1: reactibilitat: (0 un cop en data/hora/minut).
                                 // Alarma1: reactibilitat: (1 cada hora/minut).
                                 // Alarma1: reactibilitat: (2 cada en minuts).
//////////////////////////////// Alarma1: reactibilitat: (3 i 4 generador de seÃ±al de 1 minut).
//*************************************************************
byte LutBcdDec[256]={0x00,0x01,0x02,0x03,0x04,0x05,0X06,0x07,0x08,0x09,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
                     0x0A,0x0B,0x0C,0X0D,0X0E,0X0F,0X10,0X11,0X12,0X13,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
                     0x14,0x15,0x16,0X17,0X18,0X19,0X1A,0X1B,0X1C,0X1D,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
                     0x1E,0x1F,0x20,0X21,0X22,0X23,0X24,0X25,0X26,0X27,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,

                     0x28,0x29,0x2A,0X2B,0X2C,0X2D,0X2E,0X2F,0X30,0X31,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
                     0x32,0x33,0x34,0X35,0X36,0X37,0X38,0X39,0X3A,0X3B,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
                     0x3C,0x3D,0x3E,0x3F,0X40,0X41,0X42,0X43,0X44,0X45,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
                     0X46,0x47,0x48,0x49,0X4A,0X4B,0X4C,0X4D,0X4E,0X4F,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,

                     0X50,0x51,0x52,0x53,0X54,0X55,0X56,0X57,0X58,0X59,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
                     0X5A,0x5B,0x5C,0x5D,0X5E,0X5F,0X60,0X61,0X62,0X63,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
                     0XFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
                     0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,

                     0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
                     0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
                     0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
                     0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF};

//*************************************************************
byte LutDecBcd[128]={0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x10,0x11,0x12,0x13,0x14,0x15,
                     0x16,0x17,0x18,0x19,0x20,0x21,0x22,0x23,0x24,0x25,0x26,0x27,0x28,0x29,0x30,0x31,
                     0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x40,0x41,0x42,0x43,0x44,0x45,0x46,0x47,
                     0x48,0x49,0x50,0x51,0x52,0x53,0x54,0x55,0x56,0x57,0x58,0x59,0x60,0x61,0x62,0x63,
                     0x64,0x65,0x66,0x67,0x68,0x69,0x70,0x71,0x72,0x73,0x74,0x75,0x76,0x77,0x78,0x79,
                     0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x90,0x91,0x92,0x93,0x94,0x95,
                     0x96,0x97,0x98,0x99,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
                     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
//*************************************************************
byte AxM1[5]={0b00000000,0b00000000,0b00000000,0b00000000,0b10000000};
byte AxM2[5]={0b00000000,0b00000000,0b00000000,0b10000000,0b10000000};
byte AxM3[5]={0b00000000,0b00000000,0b10000000,0b10000000,0b10000000};
byte AxM4[5]={0b00000000,0b10000000,0b10000000,0b10000000,0b10000000};
```

```
//************************************************************
byte BoolByte (byte a, bool b)
{
if  ((bool)b == true){return a;
                     }
                      else {return 0b00000000;
                     }
     }
//************************************************************
byte FastDecBcd(int num){
//    return LutDecBcd[num&&0x3f];
      return LutDecBcd[num];
    }
//************************************************************
byte FastBcdDec(int gg){
//    return LutBcdDec[gg&&0x7f];
      return LutBcdDec[gg];
}
//************************************************************
  void CodificarRtc()
       {
           DS3232_00=FastDecBcd(SegonsRtc&0x7f);
           DS3232_01=FastDecBcd(MinutsRtc&0x7f);
           DS3232_02=(FastDecBcd(HoraRtc&0x3f) | BoolByte(0b01000000,ModeAmpmRtc));
           DS3232_03=FastDecBcd(DiaSemRtc&0x07);
           DS3232_04=FastDecBcd(DiaMesRtc&0x3f);
           DS3232_05=FastDecBcd(MesRtc&0x1f);
           DS3232_06=FastDecBcd(AnyRtc&0xff);
       }
 //************************************************************
  void DecodificarRtc()
       {
           SegonsRtc=FastBcdDec(DS3232_00&0x7f);
           MinutsRtc=FastBcdDec(DS3232_01&0x7f);
           HoraRtc=(FastBcdDec(DS3232_02&0x3f));
           DiaSemRtc=FastBcdDec(DS3232_03&0x07);
           DiaMesRtc=FastBcdDec(DS3232_04&0x3f);
           MesRtc=FastBcdDec(DS3232_05&0x1f);
           AnyRtc=FastBcdDec(DS3232_06&0xff);
           z=z+1;
```

```
      }
 //************************************************************
  void DecodificarAlarma1()
      {
          DS3232_0A=FastBcdDec(DS3232_0A&0x3f);
          DS3232_09=FastBcdDec(DS3232_09&0x3f);
          DS3232_08=(FastBcdDec(DS3232_08&0x7f));
          DS3232_07=FastBcdDec(DS3232_07&0x7f);
       }
 //************************************************************
void codificarAlarma1()
{
        DS3232_07=(FastDecBcd(SegonsAlarma1 & 0x7f) | AxM1[reactibleAlarma1]);
        DS3232_08=(FastDecBcd(MinutsAlarma1 & 0x7f) | AxM2[reactibleAlarma1]);
        DS3232_09=(FastDecBcd(HoraAlarma1 & 0x3f) | AxM3[reactibleAlarma1] | BoolByte(0b01000000,ModeAmpmAlarma1));
        if (ModeDiaAlarma1==false){
        DS3232_0A=(FastDecBcd(DiaMesAlarma1 & 0x3f) | AxM4[reactibleAlarma1] | BoolByte(0b01000000,ModeDiaAlarma1));}
        else{DS3232_0A=(((FastDecBcd(DiaSemAlarma1& 0x07)) | AxM4[reactibleAlarma1]) | BoolByte(0b01000000,ModeDiaAlarma1));}
}
//************************************************************
void codificarAlarma2()
{
        DS3232_0B=((FastDecBcd(MinutsAlarma2& 0x7f)) | (AxM2[reactibleAlarma2]));
        DS3232_0C=(((FastDecBcd(HoraAlarma2& 0x3f)) | AxM3[reactibleAlarma2]) | BoolByte(0b01000000,ModeAmpmAlarma2));
        if (ModeDiaAlarma2==false){
        DS3232_0D=(((FastDecBcd(DiaMesAlarma2& 0x3f)) | AxM4[reactibleAlarma2]) | BoolByte(0b01000000,ModeDiaAlarma2));}
        else{DS3232_0D=(((FastDecBcd(DiaSemAlarma2& 0x07)) | AxM4[reactibleAlarma2]) | BoolByte(0b01000000,ModeDiaAlarma2));}
}
//************************************************************
void ficarAlm1DS3232(){
       codificarAlarma1();
       delay(100);
       Wire.beginTransmission(I2C_DS3232);           // START + adreÃ$a perifÃ©ric I2C.
       Wire.send(0x07);                              // AdreÃ$a interna.
       Wire.send (DS3232_07);                        // Segons.
       Wire.send (DS3232_08);                        // Minuts.
       Wire.send (DS3232_09);                        // Hora.
       Wire.send (DS3232_0A);                        // Dia semanal/mensual.
       Wire.endTransmission();                       // STOP.
```

```
        DS3232control=DS3232control | 0b00000100;      // Pin INT per alarmes (INTCN=1).
        DS3232control=DS3232control | 0b00000001;      // Habilita alarma_1 (A1IE=1) INT0 pin2 B.2.
        DS3232status=DS3232status & 0b11111110;        // Borra flag alarma1 (A1IF=0).

        delay(100);
        Wire.beginTransmission(I2C_DS3232);            // START + adreÃ§a perifÃ©ric I2C.
        Wire.send(0x0E);                               // AdreÃ§a interna.
        Wire.send (DS3232control);                     // Byte de control.
        Wire.send (DS3232status);                      // Byte de control/status.
        Wire.endTransmission();                        // STOP.

        attachInterrupt(0,AtenAlarma,FALLING);         // Activada interrupciÃ³ INT0 pin2 B.2. per nivell baix.
        }
 //************************************************************
void ficarAlm2DS3232(){
        codificarAlarma2();
        delay(100);
        Wire.beginTransmission(I2C_DS3232);            // START + adreÃ§a perifÃ©ric I2C.
        Wire.send(0x0B);                               // AdreÃ§a interna.
        Wire.send (DS3232_0B);                         // Minuts.
        Wire.send (DS3232_0C);                         // Hora.
        Wire.send (DS3232_0D);                         // Dia semanal/mensual.
        Wire.endTransmission();                        // STOP.

        DS3232control=DS3232control | 0b00000100;      // Pin INT per alarmes (INTCN=1).
        DS3232control=DS3232control | 0b00000010;      // Habilita alarma_2 (A2IE=1) INT1 pin3 B.0.
        DS3232status=DS3232status & 0b11111101;        // Borra flag alarma1 (A2IF=0).

        delay(100);
        Wire.beginTransmission(I2C_DS3232);            // START + adreÃ§a perifÃ©ric I2C.
        Wire.send(0x0E);                               // AdreÃ§a interna.
        Wire.send (DS3232control);                     // Byte de control.
        Wire.send (DS3232status);                      // Byte de control/status.
        Wire.endTransmission();                        // STOP.

        attachInterrupt(0,AtenAlarma,FALLING);         // Activada interrupciÃ³ INT1 pin3 B.0. per nivell baix
        }
//************************************************************
void FicarEnHoraDs3232(){
  CodificarRtc();
```

```
 Wire.beginTransmission(I2C_DS3232);          // START + adreÃ§a perifÃ©ric I2C.
 Wire.send(0x00);                             // AdreÃ§a interna.
 Wire.send (DS3232_00);                       // Segons.
 Wire.send (DS3232_01);                       // Minuts.
 Wire.send (DS3232_02);                       // Hora.
 Wire.send (DS3232_03);                       // Dia de la setmana.
 Wire.send (DS3232_04);                       // Dia del mes.
 Wire.send (DS3232_05);                       // Mes.
 Wire.send (DS3232_06);                       // Any.
 Wire.endTransmission();                      // STOP.
 }
 //*************************************************************
void AtenAlarma()
       {restaurar_alarmes=!restaurar_alarmes;
       }

 //*************************************************************
 void ReactivarAlm1()
      {delay(100);                                    // Obviable. Cortesia anterior op I2C.
       Wire.beginTransmission(I2C_DS3232);            // START + adreÃ§a perifÃ©ric I2C.
       Wire.send(0x0F);                               // AdreÃ§a interna control/estat.
       Wire.endTransmission();                        // STOP
       Wire.requestFrom(I2C_DS3232, 1);               // RESTART + @esclau +1byte + STOP
       DS3232status=Wire.receive();                   // Byte de control/estat.
       delay(100);                                    // Obviable. Cortesia anterior op I2C.
       if ((DS3232status & 0b00000001) == 0b00000001){;// Flag alarma1=1?  A1IF=1?
             Serial.println(" alarma 1 ");
             DS3232status=DS3232status & 0b11111110;       // Restaura flag alarma1 (A1IF=0).
             Wire.beginTransmission(I2C_DS3232);           // START + adreÃ§a perifÃ©ric I2C.
             Wire.send(0x0F);                              // AdreÃ§a interna.
             Wire.send (DS3232status);                     // Byte de control/status.
             Wire.endTransmission();                 // STOP.
            }
          else {Serial.println(" no alarma 1 ");
          }
       }
 //*************************************************************
 void ReactivarAlm2()
      {delay(100);                                    // Obviable. Cortesia anterior op I2C.
       Wire.beginTransmission(I2C_DS3232);            // START + adreÃ§a perifÃ©ric I2C.
```

```
        Wire.send(0x0F);                                  // AdreÃ§a interna control/estat.
        Wire.endTransmission();                           // STOP
        Wire.requestFrom(I2C_DS3232, 1);                  // RESTART + @esclau +1byte + STOP
        DS3232status=Wire.receive();                      // Byte de control/estat.
        delay(100);                                       // Obviable. Cortesia anterior op I2C.
        if ((DS3232status & 0b00000010) == 0b00000010){;// Flag alarma1=1?  A1IF=1?
              Serial.println(" alarma 2 ");
              FinalMinuto=1;                 // Computar muestras
              DS3232status=DS3232status & 0b11111101;       // Restaura flag alarma1 (A1IF=0).
              Wire.beginTransmission(I2C_DS3232);           // START + adreÃ§a perifÃ©ric I2C.
              Wire.send(0x0F);                              // AdreÃ§a interna.
              Wire.send (DS3232status);                     // Byte de control/status.
              Wire.endTransmission();                       // STOP.

            }
         else {Serial.println(" no alarma 2 ");
           }
        }
 //*************************************************************
void AraAvuiDs3232(){
       delay(100);
  Wire.beginTransmission(I2C_DS3232);           // START + adreÃ§a perifÃ©ric I2C.
    Wire.send(0x00);                              // AdreÃ§a interna
    Wire.endTransmission();                       // STOP
    Wire.requestFrom(I2C_DS3232, 16);              // RESTART + @esclau +7bytes + STOP
    DS3232_00=Wire.receive();                     // Segons.
    DS3232_01=Wire.receive();                     // Minuts
    DS3232_02=Wire.receive();                     // Hora
    DS3232_03=Wire.receive();                     // Dia de la setmana
    DS3232_04=Wire.receive();                     // Dia del mes
    DS3232_05=Wire.receive();                     // Mes
    DS3232_06=Wire.receive();                     // Any
    DS3232_07=Wire.receive();                     // ALARMA_1 TEST
    DS3232_08=Wire.receive();                     // ALARMA_1 TEST
    DS3232_09=Wire.receive();                     // ALARMA_1 TEST
    DS3232_0A=Wire.receive();                     // ALARMA_1 TEST
    DS3232_0B=Wire.receive();                     // ALARMA_2 TEST
    DS3232_0C=Wire.receive();                     // ALARMA_2 TEST
    DS3232_0D=Wire.receive();                     // ALARMA_2 TEST
    DS3232control=Wire.receive();                 // CONTROL TEST
```

```
    DS3232status=Wire.receive();                    // STATUS TEST
    DecodificarRtc();
//    DecodificarAlarma1();

  Serial.print("\nCICLE    HORA    DATA    control    status       ALARMES              TEMPERATURA\n");
    Serial.print("num:");
    Serial.print(z, DEC);
    Serial.print("   ");
    Serial.print(HoraRtc, DEC);
    Serial.print(":");
    Serial.print(MinutsRtc, DEC);
    Serial.print(":");
    Serial.print(SegonsRtc, DEC);
    Serial.print("  ");
    Serial.print(DiaMesRtc, DEC);
    Serial.print("/");
    Serial.print(MesRtc, DEC);
    Serial.print("/");
    Serial.print(AnyRtc, DEC);
    Serial.print("    ");
    Serial.print(DS3232control, DEC);
    Serial.print("     ");
    Serial.print(DS3232status, DEC);
    Serial.print("    a");
    Serial.print(DS3232_07, DEC);
    Serial.print(" a");
    Serial.print(DS3232_08, DEC);
    Serial.print(" a");
    Serial.print(DS3232_09, DEC);
    Serial.print(" a");
    Serial.print(DS3232_0A, DEC);
    Serial.print(" b");
    Serial.print(DS3232_0B, DEC);
    Serial.print(" b");
    Serial.print(DS3232_0C, DEC);
    Serial.print(" b");
    Serial.print(DS3232_0D, DEC);
    Serial.print("     ");
    Serial.print(DS3232Msb, DEC);
    Serial.print(" ");
```

```
    Serial.print(DS3232Lsb, DEC);
//    Serial.print("\n");


    Wire.beginTransmission(I2C_DS3232);           // START + adreÃ§a perifÃ©ric I2C.
    Wire.send(0x14);                              // AdreÃ§a interna
    Wire.endTransmission();                       // STOP
    Wire.requestFrom(I2C_DS3232, 3);              // RESTART + @esclau +7bytes + STOP
    DS3232_0A=Wire.receive();                     // Segons.
    DS3232_0B=Wire.receive();                     // Minuts
    DS3232_0C=Wire.receive();                     // Minuts

}

byte decimal(byte dec){
    byte rr=(10*dec/25,6);
    return rr;
}

void PrinterI2c(byte addss, int qq){
    int qqq=0;
    qq=2*qq;
    Wire.beginTransmission(I2C_DS3232);           // START + adreÃ§a perifÃ©ric I2C.
    Wire.send(addss);                              // AdreÃ§a interna
    Wire.endTransmission();                       // STOP
    Wire.requestFrom(I2C_DS3232,qq);              // RESTART + @esclau +7bytes + STOP
    do{
     scan[0]=Wire.receive();
     scan[1]=Wire.receive();
     Serial.print(scan[0], DEC);
     Serial.print("'");
     Serial.print(decimal(scan[1]), DEC);
     Serial.print("  ");
     qqq=qqq+2;
    }while (qqq<qq);
}
void Imprimidor2()
{
    Serial.print("\n");
  Serial.print(xSeg, DEC);
```

```
    Serial.print("  SRAM segundos    ");
    for (i=0;i<120;i=i+2){
    Serial.print(BuffSegundos[i], DEC);
    Serial.print("'");
    Serial.print(decimal(BuffSegundos[i+1]), DEC);
    Serial.print("  ");
    if (i==60)     Serial.print("\n");
    }

    Serial.print("\n");
    Serial.print(DS3232_0A, DEC);
    Serial.print("  SRAM minutos    ");
 PrinterI2c(0x20,4);
    Serial.print("  mmmmmmm  ");
  PrinterI2c((0x20+2*56),4);
    Serial.print("\n");
    Serial.print(DS3232_0B, DEC);
    Serial.print("  SRAM horas    ");
 PrinterI2c(0x9A,4);
    Serial.print("  hhhhhhhhhh  ");
  PrinterI2c((0x9A+2*22),4);
    Serial.print("\n");
    Serial.print(DS3232_0C, DEC);
    Serial.print("  SRAM dias    ");
 PrinterI2c(0xE8,4);
    Serial.print("  ddddddddd  ");
  PrinterI2c((0xE8+2*28),4);
    Serial.print("\n");
}

//**********************************************************
void SolicitarTemp(){
       delay(100);                                    // Obviable. Cortesia anterior op I2C.
       Wire.beginTransmission(I2C_DS3232);            // START + adreÃ$a perifÃ©ric I2C.
       Wire.send(0x0F);                               // AdreÃ$a interna control/estat.
       Wire.endTransmission();                        // STOP
       Wire.requestFrom(I2C_DS3232, 1);               // RESTART + @esclau +1bytes + STOP
       DS3232status=Wire.receive();                   // Byte de control/estat
       if (( DS3232status & 0b00000100)== 0b00000000)// Bit BSY=0 de control/estat? ConversiÃ³ esdevinguda?
           {DS3232control=DS3232control | 0b00100000;// (SI)ForÃ$ar una conversiÃ³ (CONV=1).
```

```
            delay(100);                                 // (SI)Obviable. Cortesia anterior op I2C.
            Wire.beginTransmission(I2C_DS3232);         // (SI)START + adreÃ§a perifÃ©ric I2C.
            Wire.send(0x0E);                            // (SI)AdreÃ§a interna.
            Wire.send (DS3232control);                  // (SI)Byte de control.
            Wire.endTransmission();                     // (SI)STOP.
           }                                            // (NO)Deixar escapar. No solÂ·licitar conversiÃ³..
      }
//*************************************************************
void TemperaturaDs3232()
      {delay(100);                                      // Obviable. Cortesia anterior op I2C.
       Wire.beginTransmission(I2C_DS3232);              // START + adreÃ§a perifÃ©ric I2C.
       Wire.send(0x0F);                                 // AdreÃ§a interna control/estat.
       Wire.endTransmission();                          // STOP
       Wire.requestFrom(I2C_DS3232, 1);                 // RESTART + @esclau +1byte + STOP
       DS3232status=Wire.receive();                     // Byte de control/estat.
       if (( DS3232status & 0b00000100)== 0b00000000)// Bit BSY=0 de control/estat? Lliure?
             {delay(100);                               // (SI)Obviable. Cortesia anterior op I2C.
              Wire.beginTransmission(I2C_DS3232);       // (SI)START + adreÃ§a perifÃ©ric I2C.
              Wire.send(0x11);                          // (SI)AdreÃ§a interna MSB TEMPERATURA
              Wire.endTransmission();                   // (SI)STOP
              Wire.requestFrom(I2C_DS3232, 2);          // (SI)RESTART + @esclau +2bytes + STOP
              DS3232Msb=Wire.receive();                 // (SI)Part entera.
              DS3232Lsb=Wire.receive();                 // (SI)Part decimal.
//              EntrarMuestra(DS3232Msb, DS3232Lsb, 0);
             }                                          // (NO)Deixar escapar. No llegir la temperatura.
      }
//*************************************************************
//BuffSegundos[300], SegMaxMsb, SegMaxLsb, SegMaxTim[6], SegMinMsb, SegMinLsb, SegMinTim[6];
 void SegundosRegistrar()
 {
  BuffSegundos[xSeg]=DS3232Msb;
  BuffSegundos[xSeg+1]=DS3232Lsb;
  xSeg=xSeg+2;
 }
//*************************************************************
void SegundosPromediar()
{
  PromMsb=0;
  PromLsb=0;
  for (i=0; i<(xSeg); i=i+2)
```

```
 {  PromMsb=(BuffSegundos[i]+PromMsb);
    PromLsb=(BuffSegundos[i+1]+PromLsb);
}
PromMsb=2*PromMsb/xSeg;
PromLsb=2*PromLsb/xSeg;
xSeg=0;
}
//*********************************************************
void MinutosRegistrar()
{
   index=0;                                        // Tabla de minutos
//  Calculo de la dirección de la SRAM donde se guardará.
   byte Rel=Readbyte(punteralia[index][1]);        // Lee el puntero interno minutos SRAM
   byte Abs=punteralia[index][0]+2*Rel;            // Dirección absoluta donde guardar los 2 bytes.
// Guardar promedio minutos.
   Wire.beginTransmission(I2C_DS3232);             // START + Dirección periferico I2C.
   Wire.send(Abs);                                 // Dirección interna.
   Wire.send (PromMsb);                          // Promedio últimos 60 segundos. Msb
   Wire.send (PromLsb);                          // Promedio últimos 60 segundos. Lsb
   Wire.endTransmission();                         // STOP.
// Incrementar puntero minutos SRAM.
   if (Rel < (punteralia[index][2] - 1)){
     Rel=Rel+1;
   }
   else{
     Rel=0;
   }
   Wire.beginTransmission(I2C_DS3232);             // START + Dirección periferico I2C.
   Wire.send(punteralia[index][1]);                // Dirección interna.
   Wire.send (Rel);                                // Punter segons.
   Wire.endTransmission();                         // STOP.
}
//*********************************************************
void MinutosPromediar()
{
  index=0;                                           // Se trabaja con Tabla minutos
  int PromMsb=0;                                     // Inicializar variable promedio.
  int PromLsb=0;                                     // Inicializar variable promedio.
  byte NumMuestra=Readbyte(punteralia[index][1]);    // Tamaño real de la muestra. Num Muestra
  for (i=0 ; i<(2*NumMuestra) ; i=i+2){                 // Bucle NumMuestra veces de 2 en 2.
```

```
    PromMsb=(Readbyte(punteralia[index][i]+PromMsb));  // Sumatorio parte alta.
    PromLsb=(Readbyte(punteralia[index][(i+1)]+PromLsb)); // Sumatorio parte baja.
 }
 PromMsb=PromMsb/NumMuestra;                    // Sumatorio parte alta/NumMuestra.
 PromLsb=PromLsb/NumMuestra;                    // Sumatorio parte baja/NumMuestra.
}
//*********************************************************
void HorasRegistrar()
{
   index=1;                                     // Tabla de horas
//  Calculo de la dirección de la SRAM donde se guardará.
   byte Rel=Readbyte(punteralia[index][1]);     // Lee el puntero interno horario SRAM
   byte Abs=punteralia[index][0]+2*Rel;         // Dirección absoluta donde guardar los 2 bytes.
// Guardar promedio minutos.
   Wire.beginTransmission(I2C_DS3232);          // START + Dirección periferico I2C.
   Wire.send (Abs);                             // Dirección interna.
   Wire.send (PromMsb);                        // Promedio últimos 60 minutos. Msb
   Wire.send (PromLsb);                        // Promedio últimos 60 minutos. Lsb
   Wire.endTransmission();                      // STOP.
// Incrementar puntero horas SRAM.
   if (Rel < (punteralia[index][2] - 1)){
     Rel=Rel+1;
   }
   else{
     Rel=0;
   }
   Wire.beginTransmission(I2C_DS3232);          // START + Dirección periferico I2C.
   Wire.send(punteralia[index][1]);             // Dirección interna.
   Wire.send (Rel);                             // Segons.
   Wire.endTransmission();                      // STOP.
}
//*********************************************************
void HorasPromediar()
{
  index=2;                                      // Se trabaja con la matriz minutos
  int PromMsb=0;                                // Inicializar variable promedio.
  int PromLsb=0;                                // Inicializar variable promedio.
  byte NumMuestra=Readbyte(punteralia[index][1]);   // Tamaño real de la muestra. NumMuestra
  for (i=0; i<(2*NumMuestra); i=i+2)            // Bucle NumMuestra veces de 2 en 2.
 {  PromMsb=Readbyte(punteralia[index][i]+PromMsb);  // Sumatorio parte alta.
```

```
    PromLsb=Readbyte(punteralia[index][(i+1)]+PromLsb); // Sumatorio parte baja.
 }
 PromMsb=PromMsb/NumMuestra;                    // Sumatorio parte alta/NumMuestra.
 PromLsb=PromLsb/NumMuestra;                    // Sumatorio parte baja/NumMuestra.
}
//*********************************************************
void DiasRegistrar()
{
   index=2;                                     // Tabla de horas
//  Calculo de la dirección de la SRAM donde se guardará.
   byte Rel=Readbyte(punteralia[index][1]);     // Lee el puntero interno horario SRAM
   byte Abs=punteralia[index][0]+2*Rel;         // Dirección absoluta donde guardar los 2 bytes.
// Guardar promedio minutos.
   Wire.beginTransmission(I2C_DS3232);          // START + Dirección periferico I2C.
   Wire.send (Abs);                             // Dirección interna.
   Wire.send (PromMsb);                         // Promedio últimas 24 horas. Msb
   Wire.send (PromLsb);                         // Promedio últimas 24 horas. Lsb
   Wire.endTransmission();                      // STOP.
// Incrementar puntero horas SRAM.
   if (Rel < (punteralia[index][2] - 1)){
     Rel=Rel+1;
   }
   else{
     Rel=0;
   }
   Wire.beginTransmission(I2C_DS3232);          // START + Dirección periferico I2C.
   Wire.send(punteralia[index][1]);             // Dirección interna.
   Wire.send (Rel);                             // Segons.
   Wire.endTransmission();                      // STOP.
}
//*********************************************************
void RegistrarMuestra()
{
  SegundosRegistrar();
//                      Serial.print(FinalMinuto,DEC);
//                      Serial.print(" ");
  if (FinalMinuto ==1) {                    // ¿Debe procederse a promediar?
    FinalMinuto=0;
    SegundosPromediar();                        // SI Promediar segundos
    MinutosRegistrar();                         // SI Registrar promedio
```

```
    if (MinutsRtc == 0) {                                  // x/x/x x:00:00
          MinutosPromediar();
          HorasRegistrar();
          if (HoraRtc == 0) {                                  //x/x/x 00:00:00
            HorasPromediar();
            DiasRegistrar();
            if (DiaMesRtc == 1) {                                 //x/x/1 00:00:00
               Wire.beginTransmission(I2C_DS3232);               // START + Dirección periferico I2C.
               Wire.send(punteralia[2][1]);                      // Puntero del dia
               Wire.send (0x00);                                 // Restaurado a 0
               Wire.endTransmission();                           // STOP.
            }
          }
      }
  }
}
//*************************************************************
void setup(){
  Wire.begin();
  Serial.begin(9600);
  FicarEnHoraDs3232();
//  ficarAlm1DS3232();
  ficarAlm2DS3232();
 Wire.beginTransmission(I2C_DS3232);          // START + adreÃ§a perifÃ©ric I2C.
 Wire.send(0x14);                              // AdreÃ§a interna.
 Wire.send (59);                          // Segons.
 Wire.send (23);                          // Minuts.
 Wire.send (30);                          // Hora.
 Wire.endTransmission();                        // STOP.
}
//*************************************************************
void loop(){
      SolicitarTemp();                                  // Sol·licita temperatura.
      AraAvuiDs3232();                                 // LLegeix data-hora.
      Imprimidor2();
      TemperaturaDs3232();                              // Llegix temperatura encomanada. DS3232Msb. DS3232Lsb.
      RegistrarMuestra();
      if ((bool)restaurar_alarmes == true) {          // Semafor. true-> cal restaurar.
                                          restaurar_alarmes=!restaurar_alarmes;
                                          ReactivarAlm1();
```

```
                                                ReactivarAlm2();
                                                }
      delay(413);                                         // Espera.
}
//*************************************************************
//********LleonardGarciaa********http://www.fadishop.eu*******
//*************************************************************
```