```
/*Arduino Mega 2560 R3                                                    *
Programa de control de regularitat per a motocicletes                     *
Miquel Pumarola Garcia                                                    *
************************************************************

 ESTRUCTURA DADES SDCARD
 Los bytes 0-9 contiene datos genericos
 Cada tramo ocupa 10 bytes, empezando desde 10 ej:
        tramo 1  --> byte 10-19
        tramo 2  --> byte 20-29
        --
        tramo 50 --> byte 50-59
 Nombre Fichero: CARRERA
   BYTE     DATOS
        0               Numero de tramos
        10              Hora SALIDA
        11              Minuto SALIDA
        12              Segundo SALIDA
        13              Tiempo TRAMO. Horas
        14              Tiempo TRAMO. Minutos
        15              Distancia Tramo en metros HIGHBYTE
        16              Distancia Tramo en metros LOWBYTE
        17              Distancia CRONO en metros HIGHBYTE
        18              Distancia CRONO en metros LOWBYTE
        19              Velocidad CRONO en Km/h
 */

// Declaracion de librerias ---------------------------------------------------------
#include <avr/pgmspace.h>               // libreria para poder guardar variables en FLASH
#include <avr/wdt.h>                    // libreria para poder guardar variables en FLASH
#include <Timer.h>                      // libreria para temporizaciones
#include <Adafruit_GFX.h>               // libreria grafica
#include <SPI.h>                        // librería comunicaciones SPI
#include <Wire.h>                       // libreria comunicaciones GENERICA
#include <Adafruit_ILI9341.h>           // libreria comunicaciones TFT, Touch y SD
#include <Adafruit_STMPE610.h>          // libreria gestion touch
#include <SD.h>                         // libreria gestion SD card
#include <EEPROM.h>                     // Librería gestión EEPROM
#include <Adafruit_BLE_UART.h>          // libreria gestión modulo Bluetooth
#include <swRTC.h>                      // libreria relog en tiempo real

swRTC rtc;                              // Crea una instancia al relog



// Define los pines que utiliza el modulo Bluetoth
#define ADAFRUITBLE_REQ 53
#define ADAFRUITBLE_RDY 19
#define ADAFRUITBLE_RST 49

Adafruit_BLE_UART BTLEserial = Adafruit_BLE_UART(ADAFRUITBLE_REQ, ADAFRUITBLE_RDY, ADAFRUITBLE_RST);

// Declaracion de las instancias para trabajar con la SD Card
Sd2Card card;
SdVolume volume;
SdFile root;
File myFile;
// Permite que los mensajes que se envien por Println se manejen desde Flash
char p_buffer[80];
#define P(str) (strcpy_P(p_buffer, PSTR(str)), p_buffer)


// Definicion de colores personalizados
#define  ILI9341_GRIS 0xffA7ACAD
#define  ILI9341_GRISFUERTE  0xff70686f
#define  ILI9341_VERDEPASTEL 0xffc5866f


// Datos de calibracion del panel tactil
```

```
#define TS_MINX 150
#define TS_MINY 130
#define TS_MAXX 3800
#define TS_MAXY 4000

// Define Pines para Touch
#define STMPE_CS 8
Adafruit_STMPE610 ts = Adafruit_STMPE610(STMPE_CS);

// Define Pines para TFT
#define TFT_CS 10
#define TFT_DC 9
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

// Define Pines para SD Card
#define SD_CS 4

// Definicion de Pines de ENTRADA DIGITALES
int PUL1 = 23;                          // Imagen Pulsador 1
int PUL2 = 24;                          // Imagen Pulsador 2
int PUL3 = 25;                          // Imagen Pulsador 3
int SENSOR = 22;                        // Imagen Señal SENSOR

// Definicion de Pines de SALIDA DIGITALES
int LEDS = 26;                          // Leds

//Definicion de Pines de SALIDA ANALOGICOS
int BACKLIGHT = 3;                      // SALIDA PWM para control BACKLIGHT TFT


// DEFINICION DE VARIABLES TABLA DE ENTRADA
boolean vPUL1;                          // Variable Pulsador 1
boolean vPUL2;                          // Variable Pulsador 2
boolean vPUL3;                          // Variable Pulsador 3
boolean vPUL1old, vPUL2old, vPUL3old;
boolean vSENSOR;                        // Variable SENSOR
boolean vSENSOROLD = false;             // Memoria pulso anterior


// DECLARACION DE VARIABLES TABLA DE SALIDA
int vLEDS;                              // Variable LEDS



// Declaracion de VARIABLES PUBLICAS GLOBALES
byte VESTADO = 0;                       // Estado operativo actual



// Declaracion de variables almacenadas en EEPROM
int vDiametroRueda;                     // DIAMETRO RUEDA

// Declaracion variables CALCULOS DE FECHAS
byte vhora, vmin, vseg, vdia, vmes;
int vyear;

// Declaracion variables RELOG
byte segold, minold, horaold, dayold, mesold, dsemold;
byte segtmp, mintmp, horatmp, daytmp, mestmp,dsemtmp;
int yearold;
int yeartmp;

// Declaracion variables calculo tiempo de ciclo
unsigned long vciclo;                   // tiempo de ciclo
unsigned long vciclo1;                  // tiempo de ciclo temporal para calculos
unsigned long vciclo2;
```

```
// Declaracion variables  Pulsaciones Touch
unsigned int x , y , z;
unsigned int x1, y1, z1;
unsigned int xtmp, ytmp, ztmp;
unsigned int xtmp1, ytmp1,ztmp1;
boolean pulMemo;                        // Memoria de Pulsacion

// Declaracion Variables calibracion
int vcalrueda;                          // Medida diametro rueda
int vdistOrganizacion = 0;              // Distancia organizacion
float vdistrpm;                         // Distancia recorrida real en mm por vuelta de rueda
float vdistrealTotalKM;                 // Distancia recorrida total en Km
float vdistrealtotalKmold;              // Variable temporal auxiliar
float vcorrecion;                       // Factor de correccion sobre la medida real

// Declaracion Variables tramo en curso
float vtspeedReal;
float vtspeedmediaReal;
float vtspeedmediaOrganizacion;
float vtKmparcial;
long int vttiemporealinicial;
long int vttiemporealfinal;

long int vtTiempoTramoSegs;
long int vtTiempoCronoSegs;
long int vtDistTramoMetros;
long int vtdistCronoMetros;


long vnumpulsos;
long vnumpulsosparcial;



// Declaracion Variables Datos tramo
byte vCarreratmp[200];
byte vCarrera[200];
byte vntramos;                          // numero de tramos


// Declaracion Variables RX blu
byte vnumTramo;
byte vHoraSalida;
byte vMinutoSalida;
byte vSegundoSalida;
byte vTiempoHorasTramo;
byte vTiempoMinutosTramo;
byte vDistKmTramo;
byte vDistMtrTramo;
byte vDistKmCrono;
byte vDistMtrCrono;
byte vSpeedKmCrono;
byte vSpeedmCrono;

byte numRXbytes;

// Declaracion Variables Cuenta atras
byte vcaseg, vcamin, vcahora;
signed int segcalc, horacalc, mincalc;
signed long int vcuentaatras;

// Declaracion Variables BLE
byte vble=0;
boolean estabaold =0;
byte bytetmp=1;
```

```
// Declaracion de de variables TIMER
int t4;
boolean t4on=false;                     // Timer antirebote Pul1

int t5;
boolean t5on=false;                     // timer antirebote Pul2

int t6;
boolean t6on=false;                     // timer antirebote Pul3
```

```
void setup()
{
        MCUSR = 0;                              // limpia el flag de resets anteriores.

        // Inicializa el puerto serie
        Serial.begin(38400);

        // DECLARACION E/S
        // Declaracion de entradas
        pinMode(PUL1,INPUT);                    //Entrada Pulsador 1
        pinMode(PUL2,INPUT);                    //Entrada Pulsador 2
        pinMode(PUL3,INPUT);                    //Entrada Pulsador 3
        pinMode(SENSOR,INPUT);                  //Entrada Sensor
        pinMode(ADAFRUITBLE_REQ,53);            // Bluetooth REQ
        pinMode(ADAFRUITBLE_RDY,19);            // Bluetooth RDY

        //Declaracion de Salidas
        pinMode(LEDS,OUTPUT);
        pinMode(ADAFRUITBLE_RST,49);            //Bluetooth RST

        //Declaracion de Salidas PWM
        pinMode(3,OUTPUT);                      //BACKLIGHT PWM

        //Declaracion de I/O RESERVA
        pinMode(5,INPUT_PULLUP);
        pinMode(6,INPUT_PULLUP);
        pinMode(14,INPUT_PULLUP);
        pinMode(15,INPUT_PULLUP);
        pinMode(16,INPUT_PULLUP);
        pinMode(18,INPUT_PULLUP);
        pinMode(19,INPUT_PULLUP);
        pinMode(34,INPUT_PULLUP);

        // Inicializa la periferia
        inicHardware();

        // Verifica si es la primera instalacion y crea las variables por defecto
        VerificaEeprom();

        // Carga variables de Eeprom a Memoria
        cargaEeprom();

        // Carga datos de SD
        cargaSD();

        //inicializa Bluetooth
        BTLEserial.setRXcallback(rxCallback);
        BTLEserial.setACIcallback(aciCallback);
        BTLEserial.begin();


        // Inicializa relog
        rtc.stopRTC();                          //stop the RTC
        rtc.setTime(18, 58, 50);                // Ajusta hora, minutos, seg
        rtc.setDate(1, 1, 2015);                // Ajusta Dia, Mes, Año
        rtc.startRTC();


}



void loop()
{
        // Chequea Bluetooth periodicamente
        BTLEserial.pollACI();
```

```
        // Calculos de ciclo
        vciclo1=millis();

        // Chequea timers periodicamente
        t.update();

        ///// T E M P O R I Z A C I O N E S /////
        // Actualiza el relog en pantalla
        if (t2on==false) {t2 = t.after(1000, relog); t2on=true;}
        ///// F I N    T E M P O R I Z A C I O N E S  /////


        // Actualiza TABLA ENTRADAS
        lecturaIN();

        // Detecta Pulsaciones Touch
        Pulsacion();

        // Carga Pantalla Logo o Menu
        if (VESTADO == 1) {pantalla0();}
        if (VESTADO == 2) {pantalla0();}
        if (VESTADO == 3) {pantallaInicio();}


        // Calculo del ciclo
        vciclo2 = millis();
        vciclo = vciclo2 - vciclo1;
}
```

```
// FUNCION Inicializacion Perifericos
void inicHardware(){
        // Inicializa la Pantalla
        Serial.println(P("Comprobacion Pantalla"));
        tft.begin();
        if (!ts.begin()) {
                Serial.println(P("No se ha INICIALIZADO LA PANTALLA"));
        }
        else {
                tft.fillScreen(ILI9341_BLACK);
                tft.setRotation(1);
                tft.setCursor(0, 0);
        }

        // VERIFICA LECTOR SD
        Serial.println(P("Comprobacion Lector SD"));
        if (!SD.begin(SD_CS)) {
                Serial.println(P("- SD.........NO SD"));
                tft.setTextColor(ILI9341_RED); tft.println(P("- SD.........NO SD"));
                tft.setTextColor(ILI9341_GREEN);
        }
        else{
                Serial.println(P("- SD.......OK"));
                tft.println(P("- SD.........OK"));
        }

        // verifica si la tarjeta esta insertada y formateada
        Serial.println(P("Comp.Formato SD CArd"));
        if (!card.init(SPI_HALF_SPEED, SD_CS)) {
                Serial.println(P("SD CARD Sin Formato"));
                tft.setTextColor(ILI9341_RED);tft.println(P("              SIN FORMATO"));
                tft.setTextColor(ILI9341_GREEN);
        }
        else {
                Serial.println(P("           SD CARD FORMATEADA"));
                tft.setTextColor(ILI9341_GREEN);tft.println(P("               FORMATO OK"));

        }

        // Indica el tipo de la tarjeta
        switch(card.type()) {
                case SD_CARD_TYPE_SD1:
                Serial.println(P("           SD1"));
                tft.println(P("              SD1"));
                break;
                case SD_CARD_TYPE_SD2:
                Serial.println(P("           SD2"));
                tft.println(P("              SD2"));
                break;
                case SD_CARD_TYPE_SDHC:
                Serial.println(P("           SD3"));
                tft.println(P("              SDHC"));
                break;
                default:
                Serial.println(P("           Desconocido"));
        }

        // Abrimos el volumen (FAT16 or FAT32). Es necesario para acceder a mas informacion
        if (!volume.init(card)) {
                Serial.println(P("Could not find FAT16/FAT32 partition.\nMake sure you've formatted
        the card"));
        }
        else {
                // Tipo y tamaño del volumen
                uint32_t volumesize;
                Serial.print(P("\nVolume type is FAT")); Serial.println(volume.fatType(), DEC);
                tft.print(P("              FAT")); tft.println(volume.fatType(), DEC);
```

```
                //
                volumesize = volume.blocksPerCluster();
                volumesize *= volume.clusterCount();
                volumesize *= 512;
                Serial.print(P("Volume size (Mbytes): "));
                volumesize /= 1024;
                volumesize /= 1000;
                Serial.println(volumesize);
                tft.print(P("              ")); tft.print(volumesize); tft.println(P(" Mb"));
        }

        // Inicializa el touch
        if (!ts.begin()) {

        } else {
                Serial.println("Touchscreen started.");

        }


        // Fin Inicializacion
        tft.println(P(" "));
        Serial.println(P(" "));
        tft.print(P("RAM libre.."));  tft.println(FreeRam());
        Serial.print(P("RAM libre.."));  Serial.println(FreeRam());
}
```

```
// FUNCION Carga valores por defecto en EEPROM si es necesario
void VerificaEeprom(){
        // Verifica que exista la primera variable, si no crea en eeprom valores por defecto
        byte valor = EEPROM.read(0);
        if (valor != 13){
                Serial.println(P(".. EEPROM. CORRUPTA"));
                // escribimos CEROS en las primeros 50 bytes
                for (int i = 0; i < 51; i++){EEPROM.write(i, 0);}
                Serial.println(P(".. EEPROM. FORMATEADA"));
                // Ahora creamos la variables por defecto
                EEPROM.write(0, 13);                            // Variable de control
                // Utilizamos el valor de 17" (432mm como medida referencia)
                // Diametro rueda (Byte HIGH)
                EEPROM.write(1, (highByte(432)));
                // diametro rueda (Byte LOW)
                EEPROM.write(2, (lowByte(432)));

                Serial.println(P(".. EEPROM. CREADOS DATOS POR DEFECTO"));
        }
        else {
                Serial.println(P(".. EEPROM CORRECTA"));
        }
        // SET EL ESTADO
        VESTADO=1;
        consola();
}
```

```
//Carga EEPROM a Memoria
void cargaEeprom(){
        byte lowtmp, hightmp;

        hightmp = EEPROM.read(1);                                   // Diametro rueda (bytehigh)
        lowtmp = EEPROM.read(2);                                    // Diametro rueda (bytelow)
        vDiametroRueda = word(hightmp,lowtmp);

        // Calculo distancia recorrida por rpm
        vdistrpm = ((2.0 * 3.1416) * (vDiametroRueda/2));
        vdistrpm = (vdistrpm /1000);
        // Diametro rueda (bytehigh)
        hightmp = EEPROM.read(3);
        // Diametro rueda (bytelow)
        lowtmp = EEPROM.read(4);
        vdistOrganizacion = word(hightmp,lowtmp);

        Serial.println(P("- Cargadas Variables de EEPROM a Memoria"));

}
```

```
// Verifica SD
void verificaSD(){
        // Fichero Carrera.txt
        if (SD.exists("Carrera.txt")) {
        Serial.println(P("Carrera.txt OK."));}
        else {
                Serial.println("Carrera.txt no existe");
                myFile = SD.open("Carrera.txt", FILE_WRITE);

                if (myFile){
                        for (byte a=1; a<255; a++){
                                myFile.seek(a);
                                myFile.write(byte(0));
                        }
                }
                myFile.close();
                Serial.println(P("Creado Fichero --> Carrera.txt"));
        }
}

// Carga parametros de SD a Memoria
void cargaSD(){
        // Carga datos almacenados en SD a la matriz vCarrera[]
        myFile = SD.open("Carrera.txt", FILE_READ);
        if (myFile){
                myFile.seek(1);
                if (myFile.read() !=0) {
                        for (byte a=10; a<255; a++){
                                myFile.seek(a);
                                vCarrera[a]=myFile.read();
                        }
                }
                Serial.println(P("Tramos Cargados de SD"));
        }
        else {
                for (byte a=0; a<255; a++){
                        vCarrera[a]=0;
                }
                Serial.println(P("MEMORIA TRAMOS INICIALIZADA A 0"));
        }

        myFile.close();
        Serial.println(P("Datos Carrera almacenados en SD Card"));
        myFile.close();

}
```

```
// Rutina de lectura de TABLA DE ENTRADAS
void lecturaIN(){


        // Deteccion de flancos de bajada en Pulsador 1 y antirebote
        if ((!digitalRead(PUL1)) & (vPUL1old==true) & (t4on==false)) {
                vPUL1=true; vPUL1old=false;
                if (!t4on) {t4 = t.after(100, t4antirebote); t4on=true;}
        }
        if ((t4on==false) & (digitalRead(PUL1))) {vPUL1old=true;}

        // Deteccion de flancos de bajada en Pulsador 2 y antirebote
        if ((!digitalRead(PUL2)) & (vPUL2old==true) & (t5on==false)) {
                vPUL2=true; vPUL2old=false;
                if (!t5on) {t5 = t.after(100, t5antirebote); t5on=true;}
        }
        if ((t5on==false) & (digitalRead(PUL2))) {vPUL2old=true;}

        // Deteccion de flancos de bajada en Pulsador 3 y antirebote
        if ((!digitalRead(PUL3)) & (vPUL3old==true) & (t6on==false)) {
                vPUL3=true; vPUL3old=false;
                if (!t6on) {t6 = t.after(100, t6antirebote); t6on=true;}
        }
        if ((t6on==false) & (digitalRead(PUL3))) {vPUL3old=true;}



        vSENSOR = digitalRead(SENSOR);          // Sensor


        if ((vSENSOR==true) && (vSENSOROLD==false)) {
                //vdistrpm = ((2.0 * 3.1416) * (vDiametroRueda/2));
                //vdistrpm = (vdistrpm /1000);
                //vdistrealTotalKM = vdistrealTotalKM + vdistrpm;

                vnumpulsos++;
                vnumpulsosparcial++;

                // Calculo distancia real acumulada
                vdistrealTotalKM = (vnumpulsos*vdistrpm);
                vtKmparcial = (vnumpulsosparcial*vdistrpm);
                vSENSOROLD = true;



        }

        if (vSENSOR==false) {(vSENSOROLD=false);}
        //Serial.print (P("distancia -> ")); Serial.println(vdistrealTotalKM);
}
```

```
// Timers antirebote
void t4antirebote(){
        if (t4on) {t.stop(t4); t4on=false;}
}

void t5antirebote(){
        if (t5on) {t.stop(t5); t5on=false;}
}

void t6antirebote(){
        if (t6on) {t.stop(t6); t6on=false;}
}



// Capturas Touch panel
void Pulsacion(){
        // x1 - y1 se encuentran disponibles cuando se suelta el boton

        TS_Point p= ts.getPoint();
        inicio:
        while (ts.touched()){TS_Point p = ts.getPoint();pulMemo=1 ;}

        if (pulMemo==1) {
                pulMemo=0;
                TS_Point p = ts.getPoint();

                p.x = map(p.x, TS_MINY, TS_MAXY, 0, tft.height());
                // x
                p.y = map(p.y, TS_MINX, TS_MAXX, 0, tft.width());
                // y
                ztmp= p.z;

                ytmp = tft.height() - p.x;
                xtmp = p.y;

                if ((xtmp==xtmp1) && (ytmp==ytmp1) && (ztmp==ztmp1)){goto inicio;}
                xtmp1=xtmp;ytmp1=ytmp;ztmp1=ztmp;

                x=xtmp; y=ytmp;
                x1=x; y1=y;


                Serial.print(P("X > ")); Serial.print(x);Serial.print(P("  y > "));
Serial.println(y);
        }
uint16_t f;

do {
        f = ts.bufferSize();
        TS_Point p=ts.getPoint();

} while (ts.bufferSize()!=0);

}



// TFT. Presenta la Pantalla 0 (por defecto)
void pantalla0(){
        VESTADO=2;
        consola();
        // borra Pantalla sin advertencia presetup mostrada
        tft.fillScreen(ILI9341_BLACK);

        // Dibuja recuadros
        tft.drawRoundRect(2,1,318,33,5,ILI9341_BLUE);
```

```
        tft.drawRoundRect(2,35,318,205,5,ILI9341_BLUE);
        // Dibuja Texto
        tft.setCursor(60, 11); tft.setTextColor(ILI9341_YELLOW); tft.setTextSize(2);
tft.print(P("@@@@ WELCOME @@@@"));
        tft.setCursor(65,70); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
tft.print(P("Regularitat"));
        tft.setCursor(65,115); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3); tft.print(P("
Miquel"));
        delay(2000);
        VESTADO=3; consola();

}
```

```
// TFT. Presenta MENU INICIAL
void pantallaInicio(){
        // borra Pantalla
        FborraPantalla();

        // Pinta Titulo
        tft.setCursor(150, 11); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2); tft.print(P("M
E N U"));
        byte vseleccion = 1;

        tft.setCursor(45, 65); tft.setTextColor(ILI9341_RED); tft.setTextSize(4);
tft.print(P("CALIBRACIO"));
        tft.setCursor(45, 105); tft.setTextColor(ILI9341_RED); tft.setTextSize(4);
tft.print(P("HORA"));
        tft.setCursor(45, 145); tft.setTextColor(ILI9341_RED); tft.setTextSize(4);
tft.print(P("CONEXIO"));
        tft.setCursor(45, 185); tft.setTextColor(ILI9341_RED); tft.setTextSize(4);
tft.print(P("TRAM"));


        minold=0; horaold=0;
        do
                {
                // Actualiza el relog en pantalla
                t.update();
                if (t2on==false) {t2 = t.after(1000, relog); t2on=true;}
                // Captura entradas
                lecturaIN();

                if (vPUL1==true) {
                        vseleccion--;
                        if ((vseleccion < 1) | (vseleccion==1)) vseleccion=1;
                        vPUL1=false;
                }

                if (vPUL3==true){
                        vseleccion++;
                        if (vseleccion > 4) vseleccion=4;
                        vPUL3=false;
                }

                switch (vseleccion) {
                        case 1:
                                tft.setCursor(45, 65); tft.setTextColor(ILI9341_GREEN);
tft.setTextSize(4);  tft.print(P("CALIBRACIO"));
                                tft.setCursor(45, 105); tft.setTextColor(ILI9341_RED);
tft.setTextSize(4);  tft.print(P("HORA"));
                                tft.setCursor(45, 145); tft.setTextColor(ILI9341_RED);
tft.setTextSize(4); tft.print(P("CONEXIO"));
                                tft.setCursor(45, 185); tft.setTextColor(ILI9341_RED);
tft.setTextSize(4); tft.print(P("TRAM"));
                                break;

                        case 2:
                                tft.setCursor(45, 105); tft.setTextColor(ILI9341_GREEN);
tft.setTextSize(4);  tft.print(P("HORA"));
                                tft.setCursor(45, 65); tft.setTextColor(ILI9341_RED);
tft.setTextSize(4);  tft.print(P("CALIBRACIO"));
                                tft.setCursor(45, 145); tft.setTextColor(ILI9341_RED);
tft.setTextSize(4); tft.print(P("CONEXIO"));
                                tft.setCursor(45, 185); tft.setTextColor(ILI9341_RED);
tft.setTextSize(4); tft.print(P("TRAM"));

                                break;
                        case 3:
                                tft.setCursor(45, 145); tft.setTextColor(ILI9341_GREEN);
tft.setTextSize(4); tft.print(P("CONEXIO"));
```

```
                                tft.setCursor(45, 65); tft.setTextColor(ILI9341_RED);
tft.setTextSize(4);  tft.print(P("CALIBRACIO"));
                                tft.setCursor(45, 105); tft.setTextColor(ILI9341_RED);
tft.setTextSize(4);  tft.print(P("HORA"));
                                tft.setCursor(45, 185); tft.setTextColor(ILI9341_RED);
tft.setTextSize(4); tft.print(P("TRAM"));
                                break;

                        case 4:
                                tft.setCursor(45, 185); tft.setTextColor(ILI9341_GREEN);
tft.setTextSize(4); tft.print(P("TRAM"));
                                tft.setCursor(45, 65); tft.setTextColor(ILI9341_RED);
tft.setTextSize(4);  tft.print(P("CALIBRACIO"));
                                tft.setCursor(45, 105); tft.setTextColor(ILI9341_RED);
tft.setTextSize(4);  tft.print(P("HORA"));
                                tft.setCursor(45, 145); tft.setTextColor(ILI9341_RED);
tft.setTextSize(4); tft.print(P("CONEXIO"));
                }
                }

        while (vPUL2==false);
        vPUL2=false;
        switch (vseleccion) {
                                case 1:
                                        calibracio1();
                                break;

                                case 2:
                                        horaold=0;minold=0;
                                        Prelog();

                                break;

                                case 3:
                                        horaold=0;minold=0;
                                        vPUL2=false;
                                        conexio();

                                break;

                                case 4:
                                        horaold=0;minold=0;
                                        vPUL2=false;
                                        trams();
        }
}

// Calibracion Paso 1. Datos Rueda
void calibracio1(){
        /*
        vcalrueda         --> Medida diametro rueda
        vdistOrganizacion --> Distancia organizacion
        vdistreal                 --> Distancia recorrida real
        vcorrecion        --> Factor de correccion sobre la medida real
        */

        // Borra Pantalla Menu Principal
        FborraPantalla();
        // fuerza refresco relog
        minold=0;horaold=0;

        tft.setCursor(140, 11); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P("Calibracio 1/4"));

        // Pinta Datos calibracion rueda
        tft.setCursor(78, 45); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P("Calibracio roda"));

        // Declaracion variables calculos temporales
```

```
        byte vtunidad, vtdecena, vtemporal, vtcentena;

        // Saparacion valor calibracion en digitos
        vtcentena = (vDiametroRueda / 100);
        vtdecena = ((vDiametroRueda - (vtcentena *100)) / 10);
        vtunidad = (vDiametroRueda - ((vtcentena *100)  + (vtdecena *10)));

        // Presentacion digitos
        tft.setCursor(95, 140); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(4);
tft.print(vtcentena);
        tft.setCursor(155, 140); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(4);
tft.print(vtdecena);
        tft.setCursor(215, 140); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(4);
tft.print(vtunidad);

        // Recuadros Digitos
        tft.drawRect(85,130,40,50,ILI9341_RED);   // Centena
        tft.drawRect(145,130,40,50,ILI9341_RED);  // Decena
        tft.drawRect(205,130,40,50,ILI9341_RED);  // Unidad

        // Presentacion Botones

        //tft.fillRect(85,80,40,40,ILI9341_RED);
        Ftriangulo(0,87,120,6,ILI9341_RED);

        //tft.fillRect(145,80,40,40,ILI9341_RED);
        Ftriangulo(0,147,120,6,ILI9341_RED);

        //tft.fillRect(205,80,40,40,ILI9341_RED);
        Ftriangulo(0,207,120,6,ILI9341_RED);

        //tft.fillRect(85,190,40,40,ILI9341_RED);
        Ftriangulo(2,87,190,6,ILI9341_RED);

        //tft.fillRect(145,190,40,40,ILI9341_RED);
        Ftriangulo(2,147,190,6,ILI9341_RED);

        //tft.fillRect(205,190,40,40,ILI9341_RED);
        Ftriangulo(2,207,190,6,ILI9341_RED);


        // bucle modificacion valor
        byte seleccion = 1;
        boolean salir = false;
        do
        {
                Pulsacion();
                lecturaIN();
                // salida
                if ((digitalRead(PUL1)) && (digitalRead(PUL3))) {salir=1; goto salto;}

                if (vPUL2==true) {
                        seleccion++;
                        vPUL2=false;
                        if (seleccion > 3) {seleccion=1;}
                }

                if (seleccion==1) {
                        tft.drawRect(85,130,40,50,ILI9341_YELLOW);                              //
Centena
                        tft.drawRect(145,130,40,50,ILI9341_RED);                        // Decena
                        tft.drawRect(205,130,40,50,ILI9341_RED);                        // Unidad
                }
                if (seleccion==2) {
                        tft.drawRect(85,130,40,50,ILI9341_RED);                                 //
Centena
                        tft.drawRect(145,130,40,50,ILI9341_YELLOW);                             //
Decena
                        tft.drawRect(205,130,40,50,ILI9341_RED);                        // Unidad
```

```
                }
                if (seleccion==3) {
                        tft.drawRect(85,130,40,50,ILI9341_RED);                                 //
Centena
                        tft.drawRect(145,130,40,50,ILI9341_RED);                        // Decena
                        tft.drawRect(205,130,40,50,ILI9341_YELLOW);                             //
Unidad
                }


                // Actualiza el relog en pantalla
                t.update();
                if (t2on==false) {t2 = t.after(1000, relog); t2on=true;}

                // Pulsacion  + centenas
                if (((x>83 && x<123) && (y>89 && y< 126)) || ((seleccion==1) && (vPUL1==true))) {
                        vPUL1=false;
                        x=0;y=0;
                        vtcentena++;
                        if (vtcentena>9) {vtcentena=0;}
                        tft.fillRect(86,131,38,48,ILI9341_BLACK);
                        tft.setCursor(95, 140); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(4);
tft.print(vtcentena);
                }
                // Pulsacion  + decenas
                if (((x>141 && x<182) && (y>87 && y< 123)) || ((seleccion==2) && (vPUL1==true))) {
                        vPUL1=false;
                        x=0;y=0;
                        vtdecena++;
                        if (vtdecena>9) {vtdecena=0;}
                        tft.fillRect(146,131,38,48,ILI9341_BLACK);
                        tft.setCursor(155, 140); tft.setTextColor(ILI9341_GREEN);
tft.setTextSize(4); tft.print(vtdecena);
                }
                // Pulsacion  + unidades
                if (((x>200 && x<240) && (y>87 && y< 123)) || ((seleccion==3) && (vPUL1==true))) {
                        vPUL1=false;
                        x=0;y=0;
                        vtunidad++;
                        if (vtunidad>9) {vtunidad=0;}
                        tft.fillRect(206,131,38,48,ILI9341_BLACK);
                        tft.setCursor(215, 140); tft.setTextColor(ILI9341_GREEN);
tft.setTextSize(4); tft.print(vtunidad);
                }

                // Pulsacion  - centena
                if (((x>80 && x<121) && (y>192 && y< 226)) || ((seleccion==1) && (vPUL3==true))){
                        vPUL3=false;
                        x=0;y=0;
                        vtcentena--;
                        if (vtcentena>200) {vtcentena=9;}
                        tft.fillRect(86,131,38,48,ILI9341_BLACK);
                        tft.setCursor(95, 140); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(4);
tft.print(vtcentena);
                }
                // Pulsacion  - decena
                if (((x>138 && x<180) && (y>190 && y< 226)) || ((seleccion==2) && (vPUL3==true))){
                        vPUL3=false;
                        x=0;y=0;
                        vtdecena--;
                        if (vtdecena>200) {vtdecena=9;}
                        tft.fillRect(146,131,38,48,ILI9341_BLACK);
                        tft.setCursor(155, 140); tft.setTextColor(ILI9341_GREEN);
tft.setTextSize(4); tft.print(vtdecena);
                }
                // Pulsacion  - unidad
                if (((x>199 && x<239) && (y>192 && y< 226)) || ((seleccion==3) && (vPUL3==true))){
                        vPUL3=false;
                        x=0;y=0;
```

```
                        vtunidad--;
                        if (vtunidad>200) {vtunidad=9;}
                        tft.fillRect(206,131,38,48,ILI9341_BLACK);
                        tft.setCursor(215, 140); tft.setTextColor(ILI9341_GREEN);
tft.setTextSize(4); tft.print(vtunidad);
                }

        salto:;
        //
        } while (salir==0);
        vPUL1=false;
        vPUL3=false;
        //  Nuevo valor diametro rueda
        vDiametroRueda = ((vtcentena*100) + (vtdecena * 10) + vtunidad);
        // Almacenamos el valor en EEPROM
        EEPROM.write(1, (highByte(vDiametroRueda)));                                    // Diametro
rueda (Byte HIGH)
        EEPROM.write(2, (lowByte(vDiametroRueda)));                                     // diametro
rueda (Byte LOW)

        // Calculo distancia recorrida por rpm
        vdistrpm = ((2.0 * 3.1416) * (vDiametroRueda/2));
        vdistrpm = (vdistrpm /1000);

        // Fuerza a repintar Horas y min en pantalla
        minold=255;
        horaold=255;
        calibracio2();
        //
}
```

```
// Calibracion Paso 2. Datos Organizacion
void calibracio2(){
        // Borra Pantalla Menu Principal
        FborraPantalla();
//      // Dibuja recuadros
//      tft.drawRoundRect(2,1,318,33,7,ILI9341_BLUE);
//      tft.drawRoundRect(2,35,318,205,5,ILI9341_BLUE);

        tft.setCursor(140, 11); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P("Calibracio 2/4"));

        // Pinta Datos calibracion rueda
        tft.setCursor(60, 45); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P("Distancia RoadBook"));

        // Declaracion variables calculos temporales
        byte vtunidad, vtdecena, vtcentena, vtumillar, vtdmillar;

        // Saparacion valor calibracion en digitos
        vtdmillar = (vdistOrganizacion / 10000);
        vtumillar = ((vdistOrganizacion - (vtdmillar * 10000)) / 1000);
        vtcentena = (((vdistOrganizacion - (vtdmillar*10000)) - (vtumillar*1000)) /100);
        vtdecena = ((((vdistOrganizacion - (vtdmillar*10000)) - (vtumillar*1000)) - (vtcentena*100))
/ 10);
        vtunidad = ((((vdistOrganizacion - (vtdmillar*10000)) - (vtumillar*1000)) - (vtcentena*100))
- (vtdecena * 10));



        // Presentacion digitos
        tft.setCursor(124, 150); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(4);
tft.print(P("."));
        tft.setCursor(35, 140); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(4);
tft.print(vtdmillar);
        tft.setCursor(95, 140); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(4);
tft.print(vtumillar);
        tft.setCursor(155, 140); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(4);
tft.print(vtcentena);
        tft.setCursor(215, 140); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(4);
tft.print(vtdecena);
        tft.setCursor(275, 140); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(4);
tft.print(vtunidad);

        // Recuadros Digitos
        tft.drawRect(25,130,40,50,ILI9341_RED);   // Decena de Millar
        tft.drawRect(85,130,40,50,ILI9341_RED);   // Unidad de Millar
        tft.drawRect(145,130,40,50,ILI9341_RED);  // Centena
        tft.drawRect(205,130,40,50,ILI9341_RED);  // Decena
        tft.drawRect(265,130,40,50,ILI9341_RED);  // Unidad

        // Presentacion Botones +
        //tft.fillRect(25,80,40,40,ILI9341_RED);
        Ftriangulo(0,27,119,6,ILI9341_RED);

        //tft.fillRect(85,80,40,40,ILI9341_RED);
        Ftriangulo(0,87,119,6,ILI9341_RED);

        //tft.fillRect(145,80,40,40,ILI9341_RED);
        Ftriangulo(0,147,119,6,ILI9341_RED);

        //tft.fillRect(205,80,40,40,ILI9341_RED);
        Ftriangulo(0,207,119,6,ILI9341_RED);

        //tft.fillRect(265,80,40,40,ILI9341_RED);
        Ftriangulo(0,267,119,6,ILI9341_RED);

        // Presentacion Botones -
        // tft.fillRect(25,190,40,40,ILI9341_RED);
```

```
Ftriangulo(2,27,190,6,ILI9341_RED);

// tft.fillRect(85,190,40,40,ILI9341_RED);
Ftriangulo(2,87,190,6,ILI9341_RED);

// tft.fillRect(145,190,40,40,ILI9341_RED);
Ftriangulo(2,147,190,6,ILI9341_RED);

//tft.fillRect(205,190,40,40,ILI9341_RED);
Ftriangulo(2,207,190,6,ILI9341_RED);

//tft.fillRect(265,190,40,40,ILI9341_RED);
Ftriangulo(2,267,190,6,ILI9341_RED);

// bucle modificacion valor
byte seleccion=1;
boolean salir = false;
do
{
        Pulsacion();
        lecturaIN();
        // salida
        if ((digitalRead(PUL1)) && (digitalRead(PUL3))) {salir=1; goto salto;}

        if (vPUL2==true) {
                seleccion++;
                vPUL2=false;
                if (seleccion > 5) {seleccion=1;}
        }

        if (seleccion==1) {
                tft.drawRect(25,130,40,50,ILI9341_YELLOW);   // Decena de Millar
                tft.drawRect(85,130,40,50,ILI9341_RED);   // Unidad de Millar
                tft.drawRect(145,130,40,50,ILI9341_RED);  // Centena
                tft.drawRect(205,130,40,50,ILI9341_RED);  // Decena
                tft.drawRect(265,130,40,50,ILI9341_RED);  // Unidad
        }
        if (seleccion==2) {
                tft.drawRect(25,130,40,50,ILI9341_RED);   // Decena de Millar
                tft.drawRect(85,130,40,50,ILI9341_YELLOW);   // Unidad de Millar
                tft.drawRect(145,130,40,50,ILI9341_RED);  // Centena
                tft.drawRect(205,130,40,50,ILI9341_RED);  // Decena
                tft.drawRect(265,130,40,50,ILI9341_RED);  // Unidad
        }
        if (seleccion==3) {
                tft.drawRect(25,130,40,50,ILI9341_RED);   // Decena de Millar
                tft.drawRect(85,130,40,50,ILI9341_RED);   // Unidad de Millar
                tft.drawRect(145,130,40,50,ILI9341_YELLOW);  // Centena
                tft.drawRect(205,130,40,50,ILI9341_RED);  // Decena
                tft.drawRect(265,130,40,50,ILI9341_RED);  // Unidad
        }
        if (seleccion==4) {
                tft.drawRect(25,130,40,50,ILI9341_RED);   // Decena de Millar
                tft.drawRect(85,130,40,50,ILI9341_RED);   // Unidad de Millar
                tft.drawRect(145,130,40,50,ILI9341_RED);  // Centena
                tft.drawRect(205,130,40,50,ILI9341_YELLOW);  // Decena
                tft.drawRect(265,130,40,50,ILI9341_RED);  // Unidad
        }
        if (seleccion==5) {
                tft.drawRect(25,130,40,50,ILI9341_RED);   // Decena de Millar
                tft.drawRect(85,130,40,50,ILI9341_RED);   // Unidad de Millar
                tft.drawRect(145,130,40,50,ILI9341_RED);  // Centena
                tft.drawRect(205,130,40,50,ILI9341_RED);  // Decena
                tft.drawRect(265,130,40,50,ILI9341_YELLOW);  // Unidad
        }

        // Actualiza el relog en pantalla
        t.update();
        if (t2on==false) {t2 = t.after(1000, relog); t2on=true;}
```

```
                // Pulsacion  + Decena de Millar
                if (((x>21 && x<63) && (y>88 && y< 126))  || ((seleccion==1) && (vPUL1==true))) {
                        vPUL1=false;
                        x=0;y=0;
                        vtdmillar++;
                        if (vtdmillar>9) {vtdmillar=0;}
                        tft.fillRect(26,131,38,48,ILI9341_BLACK);
                        tft.setCursor(35, 140); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(4);
tft.print(vtdmillar);
                }

                // Pulsacion  + Unidad de millar
                if (((x>83 && x<123) && (y>89 && y< 126))  || ((seleccion==2) && (vPUL1==true))) {
                        vPUL1=false;
                        x=0;y=0;
                        vtumillar++;
                        if (vtumillar>9) {vtumillar=0;}
                        tft.fillRect(86,131,38,48,ILI9341_BLACK);
                        tft.setCursor(95, 140); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(4);
tft.print(vtumillar);
                }
                // Pulsacion  + centenas
                if (((x>141 && x<182) && (y>87 && y< 123))  || ((seleccion==3) && (vPUL1==true))) {
                        vPUL1=false;
                        x=0;y=0;
                        vtcentena++;
                        if (vtcentena>9) {vtcentena=0;}
                        tft.fillRect(146,131,38,48,ILI9341_BLACK);
                        tft.setCursor(155, 140); tft.setTextColor(ILI9341_GREEN);
tft.setTextSize(4); tft.print(vtcentena);
                }
                // Pulsacion  + decena
                if (((x>200 && x<240) && (y>87 && y< 123))  || ((seleccion==4) && (vPUL1==true))) {
                        vPUL1=false;
                        x=0;y=0;
                        vtdecena++;
                        if (vtdecena>9) {vtdecena=0;}
                        tft.fillRect(206,131,38,48,ILI9341_BLACK);
                        tft.setCursor(215, 140); tft.setTextColor(ILI9341_GREEN);
tft.setTextSize(4); tft.print(vtdecena);
                }

                // Pulsacion  + unidad
                if (((x>259 && x<299) && (y>87 && y< 123))  || ((seleccion==5) && (vPUL1==true))) {
                        vPUL1=false;
                        x=0;y=0;
                        vtunidad++;
                        if (vtunidad>9) {vtunidad=0;}
                        tft.fillRect(266,131,38,48,ILI9341_BLACK);
                        tft.setCursor(275, 140); tft.setTextColor(ILI9341_GREEN);
tft.setTextSize(4); tft.print(vtunidad);
                }


                // Pulsacion  - decena de millar
                if (((x>28 && x<63) && (y>189 && y< 226))  || ((seleccion==1) && (vPUL3==true))) {
                        vPUL3=false;
                        x=0;y=0;
                        vtdmillar--;
                        if (vtdmillar>200) {vtdmillar=9;}
                        tft.fillRect(26,131,38,48,ILI9341_BLACK);
                        tft.setCursor(35, 140); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(4);
tft.print(vtdmillar);
                }

                // Pulsacion  - unidad de millar
                if (((x>80 && x<121) && (y>192 && y< 226))  || ((seleccion==2) && (vPUL3==true)))  {
                        vPUL3=false;
```

```
                        x=0;y=0;
                        vtumillar--;
                        if (vtumillar>200) {vtumillar=9;}
                        tft.fillRect(86,131,38,48,ILI9341_BLACK);
                        tft.setCursor(95, 140); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(4);
tft.print(vtumillar);
                }
                // Pulsacion  - centena
                if (((x>138 && x<180) && (y>190 && y< 226))  || ((seleccion==3) && (vPUL3==true)))
{
                        vPUL3=false;
                        x=0;y=0;
                        vtcentena--;
                        if (vtcentena>200) {vtcentena=9;}
                        tft.fillRect(146,131,38,48,ILI9341_BLACK);
                        tft.setCursor(155, 140); tft.setTextColor(ILI9341_GREEN);
tft.setTextSize(4); tft.print(vtcentena);
                }
                // Pulsacion  - decena
                if (((x>199 && x<239) && (y>192 && y< 226))  || ((seleccion==4) && (vPUL3==true))) {
                        vPUL3=false;
                        x=0;y=0;
                        vtdecena--;
                        if (vtdecena>200) {vtdecena=9;}
                        tft.fillRect(206,131,38,48,ILI9341_BLACK);
                        tft.setCursor(215, 140); tft.setTextColor(ILI9341_GREEN);
tft.setTextSize(4); tft.print(vtdecena);
                }
                // Pulsacion  - unidad
                if (((x>261 && x<301) && (y>192 && y< 226))  || ((seleccion==5) && (vPUL3==true))) {
                        vPUL3=false;
                        x=0;y=0;
                        vtunidad--;
                        if (vtunidad>200) {vtunidad=9;}
                        tft.fillRect(266,131,38,48,ILI9341_BLACK);
                        tft.setCursor(275, 140); tft.setTextColor(ILI9341_GREEN);
tft.setTextSize(4); tft.print(vtunidad);
                }
        salto:;

        } while (salir==false);


        vdistOrganizacion=
(vtdmillar*10000)+(vtumillar*1000)+(vtcentena*100)+(vtdecena*10)+vtunidad;

        // Almacenamos en eprom
        EEPROM.write(3, (highByte(vdistOrganizacion)));                                 // Dist.
organizacion (Byte HIGH)
        EEPROM.write(4, (lowByte(vdistOrganizacion)));                                  // Dist.
Organizacion (Byte LOW)

        vdistrealTotalKM=0;
        // Fuerza a repintar Horas y min en pantalla
        minold=255;
        horaold=255;

        calibracio3();
        //


}
```

```
// Calibracion Paso3.
void calibracio3(){
        String vdistantciatmp;
        int vlongitud, vx, vxold;
        vxold=4;

        // Borra Pantalla Menu Principal
        FborraPantalla();
//      // Dibuja recuadros
//      tft.drawRoundRect(2,1,318,33,7,ILI9341_BLUE);
//      tft.drawRoundRect(2,35,318,205,5,ILI9341_BLUE);

        tft.setCursor(140, 11); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P("Calibracio 3/4"));

        // Pinta Datos calibracion rueda
        tft.setCursor(100, 45); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P("DISTANCIA"));

        do
        {
        // Calculos de ciclo
        //vciclo1=millis();

        // Lectura de Pulsos
        lecturaIN();
        // Actualiza el relog en pantalla
        t.update();
        if (t2on==false) {t2 = t.after(1000, relog); t2on=true;}

        // Presenta la Distancia recorrida, solo si cambia

        if (vdistrealtotalKmold!=vdistrealTotalKM){
                // Borra dato anterior
                tft.setCursor(vxold, 125); tft.setTextColor(ILI9341_BLACK); tft.setTextSize(6);
tft.print(vdistrealtotalKmold,3);

                // rutinilla para centrar el resultado en palabra
                vdistantciatmp = String(vdistrealTotalKM);
                vlongitud = vdistantciatmp.length();
                // Calculo eje X para centrar datos
                vx = (160 - (((vlongitud+1)* 6 * 6)/2));
                tft.setCursor(vx, 125); tft.setTextColor(ILI9341_GREEN);
tft.print(vdistrealTotalKM,3);
                vdistrealtotalKmold=vdistrealTotalKM;
                vxold=vx;
        }

        // Calculo del ciclo
//      vciclo2 = millis();
//      vciclo = vciclo2 - vciclo1;
//      Serial.println(vciclo);

        } while (vPUL2==false);
        vPUL2=false;

        // Fuerza a repintar Horas y min en pantalla
        minold=255;
        horaold=255;
        // Paso 4
        calibracio4();

}
```

```
// Calibracion Paso 4. Calculo Correccion
void calibracio4(){
        // Borra Pantalla Menu Principal
        FborraPantalla();
//      // Dibuja recuadros
//      tft.drawRoundRect(2,1,318,33,7,ILI9341_BLUE);
//      tft.drawRoundRect(2,35,318,205,5,ILI9341_BLUE);
        do
                {
                // Lectura de Pulsos
                lecturaIN();
                // Actualiza el relog en pantalla
                t.update();
                if (t2on==false) {t2 = t.after(1000, relog); t2on=true;}

                tft.setCursor(140, 11); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P("Calibracio 4/4"));

                // Pinta tITULO
                tft.setCursor(100, 45); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P("CORRECCION"));

                // Pinta Datos Textuales
                tft.setCursor(5, 85); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P(" * Dist. RoadBook"));
                tft.setCursor(5, 135); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P(" * Dist. Real"));
                tft.setCursor(5, 185); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P(" * F. Correccion"));

                // Pinta Distacias
                tft.setCursor(230, 85); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(vdistOrganizacion);
                tft.setCursor(230, 135); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(vdistrealTotalKM);


                // vcorrecion=vdistOrganizacion / vdistrealTotalKM;
                vcorrecion= vdistrealTotalKM / vDiametroRueda;
                vcorrecion = vdistOrganizacion / vcorrecion;


                tft.setCursor(230, 185); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(vcorrecion,3);
                } while (vPUL2==false);
                vPUL2=false;
                VESTADO=3;consola();
}
```

```
// Bluetooth RECEPCION Y ALMACENAMIENTO DE DATOS EN SDCARD
void conexio(){
        /*
        ESTRUCTURA DATOS SDCARD
        Los bytes 0-9 contiene datos genericos
        Cada tramo ocupa 11 bytes, empezando desde 10 ej:
        tramo 1  --> byte 10-20
        tramo 2  --> byte 22-32
        --
        tramo 50 --> byte 50-59
        Nombre Fichero: CARRERA
        BYTE        DATOS
        0               Numero de tramos
        --
        10              Numero de tramo
        11              Hora SALIDA
        12              Minuto SALIDA
        13              Segundo SALIDA
        14              Tiempo TRAMO. Horas
        15              Tiempo TRAMO. Minutos
        16              Distancia Tramo en KM (ENTERO)
        17              Distancia Tramo en METROS (DECIMAL)
        18              Distancia CRONO en kM (ENTERO)
        19              Distancia CRONO en METROS (DECIMAL)
        20              Velocidad Crono en KM
        21              Velocidad CRONO en m


        */
        // Recepcion de datos
        // Alamacenamiento de datos recibidos en el array de trabajo vCarrera[]



        // Borra Pantalla Menu Principal
        FborraPantalla();
        // Pone titulo
        tft.setCursor(220, 11); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P("CONEXIO"));
        boolean ON=0;
        boolean linea1mostrada=0;
        boolean linea2mostrada=1;
        boolean linea3mostrada=1;
        boolean linea1estadomostrado = 0;
        boolean linea2estado0mostrado = 0;
        boolean linea2estado1mostrado = 0;
        boolean linea3estado0mostrado=0;
        boolean salir = false;
        horaold=90; minold=90;

        do {
                Pulsacion();
                lecturaIN();
                // Actualiza el relog en pantalla
                t.update();
                if (t2on==false) {t2 = t.after(1000, relog); t2on=true;}

                // Salida
                if ((digitalRead(PUL1)) && (digitalRead(PUL3))) {salir=1; goto salto;}

                // lineas
                if (linea1mostrada == 0){
                        tft.drawRoundRect(15,50,290,46,9,ILI9341_BLUE);
                        tft.setCursor(30, 65); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P("BLUETOOTH"));
                        tft.fillRoundRect(200,53,95,38,9,ILI9341_RED);
                        tft.setCursor(230, 65); tft.setTextColor(ILI9341_BLACK); tft.setTextSize(2);
tft.print(P("OFF"));
```

```
                        linea1mostrada=1;
                }
                if (linea2mostrada == 0) {
                        tft.drawRoundRect(15,105,290,46,9,ILI9341_BLUE);
                        tft.setCursor(30, 120); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P("ESTADO BLE"));
                        tft.fillRoundRect(200,108,95,38,9,ILI9341_RED);
                        tft.setCursor(230, 120); tft.setTextColor(ILI9341_BLACK);
tft.setTextSize(2); tft.print(P("OFF"));
                        linea2mostrada=1;
                }
                if (linea3mostrada == 0) {
                        tft.drawRoundRect(15,160,290,46,9,ILI9341_BLUE);
                        tft.setCursor(30, 175); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P("Rx TRAMOS"));
                        tft.drawRoundRect(200,163,95,38,9,ILI9341_RED);
                        tft.setCursor(230, 175); tft.setTextColor(ILI9341_BLACK);
tft.setTextSize(2); tft.print(P("0"));
                        linea3mostrada=1;
                }

                // detecta pulsacion boton central
                if (vPUL2==true) {ON=!ON; vPUL2=false;}

                // ON / OFF BLUETOOTH
                if (ON==true) {
                        if (linea1estadomostrado==0) {
                                tft.fillRoundRect(200,53,95,38,9,ILI9341_GREEN);
                                tft.setCursor(235, 65); tft.setTextColor(ILI9341_RED);
tft.setTextSize(2); tft.print(P("ON"));
                                linea1estadomostrado=1;
                                delay (1000);
                                linea2mostrada=0;
                                if (estabaold) {delay (500); vble=1;}
                        }
                        // Chequea Bluetooth
                        BTLEserial.pollACI();
                        switch (vble)
                                {
                                case 0:                                                         //
desconectado
                                        break;
                                case 1:
                                        if (linea2estado0mostrado == 0) {
                                                tft.fillRoundRect(200,108,95,38,9,ILI9341_YELLOW);
                                                tft.setCursor(207, 120);
tft.setTextColor(ILI9341_BLACK); tft.setTextSize(2); tft.print(P("VISIBLE"));
                                                linea2estado0mostrado =1 ;
                                                if (estabaold) {delay (500); vble=2;}
                                        }
                                        break;
                                case 2:
                                        if (linea2estado1mostrado == 0){
                                                tft.fillRoundRect(200,108,95,38,9,ILI9341_GREEN);
                                                tft.setCursor(205, 120);
tft.setTextColor(ILI9341_RED); tft.setTextSize(2); tft.print(P("CONNECT"));
                                                linea2estado1mostrado = 1;
                                                delay (1000);
                                                linea3mostrada=1;
                                        }
                                        break;
                                case 3:  // ERROR TAMAÑO TRAMA RECIBIDA
                                        if (linea3estado0mostrado == 0){
                                                tft.fillRect(16,161,288,63,ILI9341_BLACK);
                                                tft.drawRoundRect(15,160,290,65,9,ILI9341_BLUE);
                                                tft.setCursor(50, 185);
tft.setTextColor(ILI9341_RED); tft.setTextSize(2); tft.print(P("ERROR DATOS TRAMA"));
                                                linea3estado0mostrado = 1;
                                                delay (1000);
```

```
                                        linea3mostrada=1;
                                }
                                break;
                        case 4:
                                        tft.setCursor(50, 190);
tft.setTextColor(ILI9341_RED); tft.setTextSize(2); tft.print(P("Tramos Recibidos :  "));
                                        tft.print(vntramos);

                                        break;
                        case 5:
                                        tft.fillRect(20,170,290,70,ILI9341_BLACK);
                                        tft.drawRoundRect(15,160,290,65,9,ILI9341_BLUE);
                                        tft.setCursor(30, 165);
tft.setTextColor(ILI9341_RED); tft.setTextSize(2); tft.print(P("Datos Recibidos:  "));
                                        tft.setCursor(30, 190);
tft.setTextColor(ILI9341_WHITE); tft.setTextSize(1);

                                        for (byte a=1; a<(numRXbytes+1); a++) {


                                                tft.print(vCarreratmp[a]);tft.print(" ");

                                        }
                                        numRXbytes=0;
                                        // Guarda los datos recibidos ordenadamente en
vCarrera y en SDCard
                                        byte b ,a;
                                        b = vCarreratmp[1];
                                        b = (10+((b-1)*12));
                                        for (a=0; a<12; a++){
                                                vCarrera[a+b]=vCarreratmp[a+1];
                                                //Serial.print(a+b); Serial.print(P(" t "));
Serial.println(vCarrera[a+b]);
                                        }
                                        vble=10;
                                        break;
                        default:;
                                break;

                        }

                }

        salto:;
        } while (salir==false);

        estabaold=1;
        // Alamacenamiento de datos en SD Card
        myFile = SD.open("Carrera.txt", FILE_WRITE);
        byte a;

        if (myFile){
                for (a=1; a<255; a++){
                        myFile.seek(a);
                        myFile.write(vCarrera[a]);
                }
        }
        myFile.close();
        Serial.println(P("Datos Carrera almacenados en SD Card"));



}
```

```
// Gestion de Eventos Bluetooth
void aciCallback(aci_evt_opcode_t event)
{
        switch(event)
        {
                case ACI_EVT_DEVICE_STARTED:
                Serial.println(F("VISIBLE"));
                vble=1;
                break;
                case ACI_EVT_CONNECTED:
                Serial.println(F("CONECTADO"));
                vble=2;
                break;
                case ACI_EVT_DISCONNECTED:
                Serial.println(F("DESCONECTADO"));
                vble=0;
                break;
                default:
                break;
        }
}
```

```
// Recepcion de datos Bluetooth
void rxCallback(uint8_t *buffer, uint8_t len)
{
        Serial.print(F("Received "));
        Serial.print(len);
        Serial.println(F(" bytes: "));


        char chartmp;   // caracter recibidos
        int cifratmp;   // CARACTER recibido en decimal
        int dato=0;
        int digito=0;

        int datotmp[100];
        int multiplicador = 1;

        for(int i=0; (i<(len)); i++){
                chartmp = ((char)buffer[i]);
                cifratmp = int(chartmp);
                cifratmp = (cifratmp - 48);
                //Serial.println(chartmp);


                if (chartmp != ';') {
                        digito++;
                        datotmp[digito] = cifratmp;
                        //Serial.print("datotmp");Serial.print(" ");Serial.println(datotmp[digito]);
                }

                if (chartmp == ';') {
                        for (byte a=0; a<digito; a++){
                                dato = dato + ((datotmp[digito-(a)] * multiplicador))  ;
                                multiplicador = (multiplicador * 10);
                                //Serial.println(dato);
                        }

                        vCarreratmp[bytetmp] = dato;
                        Serial.println(dato);
                        bytetmp++;
                        digito=0;
```

```
                multiplicador = 1;
                dato=0;
        }
}
        numRXbytes= bytetmp-1;

        //Serial.print(P("numrx  ")); Serial.println(numRXbytes);

        if ((numRXbytes%12) != 0) {vble=3;}                             // Error Recepcion
                else {vble = 5; bytetmp=1;}
        // No byte TRAMO
}
```

```
// Funcion. Devuelve el numero de tramos
byte cuentaTramos(){
        vntramos=0;
        for (byte a=10; a<255; a+12){
                if (vCarrera[a]!=0){vntramos++;}
                        else {return vntramos;}
        }
}
```

```
// TRAMO 1. Seleccion de tramo
void trams(){
        // Borra Pantalla Menu Principal
        FborraPantalla();
        // dibuja Titulo
        tft.setCursor(125, 11); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P("Seleccion TRAMO"));

        // Actualiza el relog en pantalla
        t.update();
        if (t2on==false) {t2 = t.after(1000, relog); t2on=true;}

        // Presenta los tramos disponibles

        minold=0;segold=0;
        byte b=10;
        byte a=1;
        boolean salir=0;
        do
                {

                if (vCarrera[b] == 0) {salir = true; vntramos=(a-1);goto bye;}
                //Serial.print(b);Serial.print("    ");Serial.println(vCarrera[b]);
                tft.setCursor((15+((a/4)*40)),(70+(a*18))) ; tft.setTextColor(ILI9341_WHITE);
tft.setTextSize(2); tft.print(P("TR "));tft.print(a);
                a=a+1;
                b=b+12;
                bye:;
                } while (salir==false);


        horaold=0; minold=0;
        byte pos=1, posold=0;
        boolean actualiza=0;
        do
        {
                lecturaIN();
                // Actualiza el relog en pantalla
                t.update();
                if (t2on==false) {t2 = t.after(1000, relog); t2on=true;}

                // salida
                if ((digitalRead(PUL1)) && (digitalRead(PUL3))) {goto salida;}

                // Deteccion teclas up/down

                Serial.print(P("pos  "));Serial.println(pos);
                Serial.print(P("tramos  "));Serial.println(vntramos);


                if (vPUL1) {pos=pos+1;vPUL1=false;}
                if (vPUL3) {pos=pos-1;vPUL3=false;}

                if (pos>vntramos) {pos=1;}
```

```
                if (pos<1) {pos=vntramos;}

                if (pos != posold) {posold=pos; actualiza=1;}


                if (actualiza) {
                        //borra selecciones actuales
                        byte b=10;
                        byte a=1;
                        boolean salir2=false;
                        do
                                {
                                        if (vCarrera[b] == 0) {salir2 = true;
Serial.println(P("sale")); goto bye2;}
                                        tft.setCursor((15+((a/4)*40)),(70+(a*18))) ;
tft.setTextColor(ILI9341_WHITE); tft.setTextSize(2); tft.print(P("TR "));tft.print(a);
                                        b=b+12;
                                        a=a+1;
                                        bye2:;

                                }while (salir2==false);

                        // Cambia seleccion
                        if (vntramos!=0) {tft.setCursor((15+((pos/4)*40)),(70+(pos*18))) ;
tft.setTextColor(ILI9341_RED); tft.setTextSize(2); tft.print(P("TR "));tft.print(pos);
                                actualiza=0;
                        }
                }

        } while (vPUL2==false);

        // Seleccion de tramo
                if (vPUL2) {
                        vPUL2=false;


                        vnumTramo = vCarrera[pos+9];
                        vHoraSalida = vCarrera[pos+10];
                        vMinutoSalida = vCarrera[pos+11];
                        vSegundoSalida = vCarrera[pos+12];
                        vTiempoHorasTramo = vCarrera[pos+13];
                        vTiempoMinutosTramo = vCarrera[pos+14];
                        vDistKmTramo = vCarrera[pos+15];
                        vDistMtrTramo  = vCarrera[pos+16];
                        vDistKmCrono = vCarrera[pos+17];
                        vDistMtrCrono = vCarrera[pos+18];
                        vSpeedKmCrono = vCarrera[pos+19];
                        vSpeedmCrono = vCarreratmp[pos+20];


                        vtDistTramoMetros = ((vDistKmTramo*1000) + (vDistMtrTramo));
                        vtdistCronoMetros = ((vDistKmCrono*1000) + (vDistMtrCrono));
                        vtspeedmediaOrganizacion = ((vSpeedKmCrono) + (vSpeedmCrono/1000));

                        vtTiempoTramoSegs = ((vTiempoHorasTramo*3600) + (vTiempoMinutosTramo*60));
                        // vtTiempoCronoSegs =




                }
        trams2();
        salida:;
        vPUL1=0;vPUL2=0;vPUL3=0;
        pantallaInicio();
}
```

```
// TRAMO 2. CONFIRMACION
void trams2(){
        // Borra Pantalla Menu Principal
        FborraPantalla();
        // Dibuja Titulo
        tft.setCursor(112, 11); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P("CONFIRMACIO TRAM"));

        // PRESENTA MENSAJE
        tft.drawRoundRect(68,72,178,135,5,ILI9341_BLUE);
        tft.setCursor(105, 86); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
tft.print(P("INICIO"));
        tft.setCursor(113, 116); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
tft.print(P("TRAMO"));
        tft.setCursor(148, 150); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
tft.print(vnumTramo);
        tft.setCursor(105, 190); tft.setTextColor(ILI9341_WHITE); tft.setTextSize(1);
tft.print(P("Pulsa Boton CENTRAL"));

        segold=0;minold=0;horaold=0;
        boolean salir=0;
        do
                {
                // Actualiza el relog en pantalla
                t.update();
                if (t2on==false) {t2 = t.after(1000, relog); t2on=true;}
                // Detecta Pulsaciones
                lecturaIN();
                // salida
                if ((digitalRead(PUL1)) && (digitalRead(PUL3))) {salir=1; break;}


                } while (vPUL2==false);
                vPUL2=false;vPUL1=false;vPUL3=false;
                if (salir==1) {trams();}

                trams3();
}
// TRAMS 3. Cuenta atras
void trams3(){
        // Borra Pantalla Menu Principal
        FborraPantalla();
        // Dibuja Titulo
        tft.setCursor(216, 10); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P("TRAMO"));
        tft.setCursor(292, 7); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
tft.print(vnumTramo);
        // DIBUJA RECTANGULOS
        tft.drawRoundRect(80,48,177,62,5,ILI9341_BLUE);
        tft.drawRoundRect(80,48,177,20,5,ILI9341_BLUE);

        tft.drawRoundRect(24,133,274,78,5,ILI9341_BLUE);
        tft.drawRoundRect(24,133,274,20,5,ILI9341_BLUE);

        // DIBUJA TEXTO FIJO
        tft.setCursor(136, 55); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(1);
tft.print(P("HORA SALIDA"));
        tft.setCursor(122, 140); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(1);
tft.print(P("CUENTA ATRAS"));

        // DIBUJA HORA SALIDA
        tft.setTextSize(2);
        if (vHoraSalida<10)             {tft.setCursor(96, 84);tft.print(0);tft.setCursor(108, 84);
tft.print(vHoraSalida);}
                else                            {tft.setCursor(96, 84);tft.print(vHoraSalida);}

        if (vMinutoSalida<10)   {tft.setCursor(156,84);tft.print(0);tft.setCursor(168, 84);
tft.print(vMinutoSalida);}
                else                            {tft.setCursor(156,84);tft.print(vMinutoSalida);}
```

```
        if (vSegundoSalida<10)  {tft.setCursor(212, 84);tft.print(0);tft.setCursor(224, 84);
tft.print(vSegundoSalida);}
                else                            {tft.setCursor(212, 84); tft.print(vSegundoSalida);}

        // Pone ":"
        tft.setCursor(136, 84); tft.print(P(":"));
        tft.setCursor(192, 84); tft.print(P(":"));

        tft.setTextSize(3);
        tft.setCursor(108, 168); tft.print(P(":"));
        tft.setCursor(200, 168); tft.print(P(":"));

        boolean salir=false;
        minold=0;horaold=0;
        do {
                // Actualiza el relog en pantalla
                t.update();
                if (t2on==false) {t2 = t.after(1000, relog); t2on=true;}
                // Detecta Pulsaciones
                lecturaIN();
                // salida
                if ((digitalRead(PUL1)) && (digitalRead(PUL3))) {salir=1; break;}

                // Calcula la cuenta atras
                byte vcahoraold, vcaminold, vcasegold;




                FCuentaatras(vHoraSalida,vMinutoSalida,vSegundoSalida);
                horacalc = (vcuentaatras / 3600);
                mincalc = (vcuentaatras - (horacalc * 3600)) / 60;
                segcalc = (vcuentaatras - (mincalc*60));



                 Serial.print(vHoraSalida); Serial.print(P("  ")); Serial.print(vMinutoSalida);
Serial.print(P("  "));Serial.println(vSegundoSalida);
                 Serial.println(vcuentaatras);
                 Serial.print(horacalc);  Serial.print(P("
"));Serial.print(mincalc);Serial.print(P("    "));Serial.println(segcalc);

                // Dibuja cuenta atras
                tft.setTextSize(4);

                tft.setTextColor(ILI9341_GREEN);
                if (horacalc != vcahoraold) {
                        tft.fillRect(44,168,60,35,ILI9341_BLACK);
                        if (horacalc<10)        {tft.setCursor(44,
168);tft.print(0);tft.setCursor(56, 168); tft.print(horacalc);}
                        else                                    {tft.setCursor(44,
168);tft.print(horacalc);}
                }

                if (mincalc != vcaminold) {
                        tft.fillRect(132,168,60,35,ILI9341_BLACK);
                        if (mincalc<10)
        {tft.setCursor(132,168);tft.print(0);tft.setCursor(144, 168); tft.print(mincalc);}
                        else                            {tft.setCursor(132,168);tft.print(mincalc);}
                }

                if (segcalc != vcasegold) {
                        tft.fillRect(228,168,60,35,ILI9341_BLACK);
                        if (segcalc<10)         {tft.setCursor(228,
168);tft.print(0);tft.setCursor(240, 168);  tft.print(segcalc);}
                        else                            {tft.setCursor(228, 168);
tft.print(segcalc);}
```

```
                }

                vcahoraold = horacalc; vcaminold = mincalc; vcasegold = segcalc;
                if ((horacalc==0) && (mincalc==0) && (segcalc==0)) {
                        // Pone a cero contadores de distancia y tiempo
                        vdistrealTotalKM=0;
                        vtKmparcial=0;
                        vnumpulsos=0;
                        vnumpulsosparcial=0;
                        vttiemporealinicial=Ftimetoseg();



                        salir = 1;
                        break;
                }


        } while (salir==false);
        vPUL2=false;vPUL1=false;vPUL3=false;
        if ((vcahora==0) && (vcamin==0) && (vcaseg==0)) {trams4();}




}
```

```
// TRAMS 4. Datos iniciales
void trams4(){
        // Borra Pantalla Menu Principal
        FborraPantalla();
        // Dibuja Titulo
        tft.setCursor(216, 10); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P("TRAMO"));
        tft.setCursor(292, 7); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
tft.print(vnumTramo);

        // DIBUJA RECTANGULOS
        tft.drawRoundRect(9,44,304,39,5,ILI9341_BLUE);
        tft.drawRoundRect(9,92,304,39,5,ILI9341_BLUE);
        tft.drawRoundRect(9,140,304,39,5,ILI9341_BLUE);
        tft.drawRoundRect(9,188,304,39,5,ILI9341_BLUE);

        // DIBUJA TEXTO FIJO
        tft.setCursor(16, 56); tft.setTextColor(ILI9341_WHITE); tft.setTextSize(2); tft.print(P("Km
Total"));
        tft.setCursor(16, 103); tft.setTextColor(ILI9341_WHITE); tft.setTextSize(2); tft.print(P("Km
Parcial"));
        tft.setCursor(16, 152); tft.setTextColor(ILI9341_WHITE); tft.setTextSize(2);
tft.print(P("Vel. Media"));
        tft.setCursor(16, 200); tft.setTextColor(ILI9341_WHITE); tft.setTextSize(2);
tft.print(P("Diferencia"));

        // DIBUJA VALORES A CERO
        tft.setCursor(170, 52); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
tft.print(P("000.000"));
        tft.setCursor(170, 100); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
tft.print(P("000.000"));
        tft.setCursor(170, 148); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
tft.print(P("000.000"));
        tft.setCursor(170, 196); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
tft.print(P("000.000"));

        horaold=0;minold=0;
        boolean salir=false;
        do{
                // Actualiza el relog en pantalla
                t.update();
                if (t2on==false) {t2 = t.after(1000, relog); t2on=true;}
                // Detecta Pulsaciones
                lecturaIN();
                // salida
                if ((digitalRead(PUL1)) && (digitalRead(PUL3))) {salir=1; break;}
                if (vPUL3) {vPUL3=false; vnumpulsosparcial=0;}

                // PRESENTACION KM REALES
                tft.fillRect(170, 52,140,23,ILI9341_BLACK);
                tft.setCursor(206, 52); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
tft.print((vdistrealTotalKM/1000),3);
                tft.fillRect(170, 100,140,23,ILI9341_BLACK);
                tft.setCursor(206, 100); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
tft.print((vtKmparcial/1000),3);
                tft.fillRect(170, 148,140,23,ILI9341_BLACK);
                tft.setCursor(206, 148); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
tft.print((vtspeedmediaOrganizacion));

                //calculo velocidad media real
                vttiemporealfinal=Ftimetoseg();
                vtspeedmediaReal = vdistrealTotalKM / (vttiemporealinicial-vttiemporealfinal);

                tft.fillRect(170, 196,140,23,ILI9341_BLACK);
                tft.setCursor(206, 196); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
tft.print((vtspeedmediaOrganizacion-vtspeedmediaReal)/1000);
```

```
        } while (vPUL2==false);
        vPUL2=false;vPUL1=false;vPUL3=false;
        if (salir == 1) {trams();}
        //trams5();









};
```

```
void trams5(){
        // Borra Pantalla Menu Principal
        FborraPantalla();
        // Dibuja Titulo
        tft.setCursor(216, 8); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
tft.print(P("TRAMO"));
        tft.setCursor(292, 8); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(4);
tft.print(vnumTramo);

        // DIBUJA RECTANGULOS
        tft.drawRoundRect(8,52,304,39,5,ILI9341_BLUE);
        tft.drawRoundRect(8,117,304,39,5,ILI9341_BLUE);
        tft.drawRoundRect(8,183,304,39,5,ILI9341_BLUE);

        // DIBUJA TEXTO FIJO
        tft.setCursor(14, 65); tft.setTextColor(ILI9341_WHITE); tft.setTextSize(2); tft.print(P("Km
Total"));
        tft.setCursor(14, 129); tft.setTextColor(ILI9341_WHITE); tft.setTextSize(2); tft.print(P("Km
Parcial"));
        tft.setCursor(14, 195); tft.setTextColor(ILI9341_WHITE); tft.setTextSize(2);
tft.print(P("Vel. Media"));


        // DIBUJA VALORES A CERO
        tft.setCursor(160, 52); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
tft.print(P("000.000"));
        tft.setCursor(160, 100); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
tft.print(P("000.000"));
        tft.setCursor(160, 148); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(3);
tft.print(P("000.000"));


        do{









        } while (vPUL2==false);
        vPUL2=false;vPUL1=false;vPUL3=false;
        trams5();


}
```

```
// Calculo diferencia horaria "cuenta atras"
void FCuentaatras(byte hora, byte minuto, byte segundo){
        // el resultado queda almacenado en las variables publicas vcaseg, vcamin, vcahora;

        byte segactual, minactual, horaactual;
        long int segundosactuales, segundossalida;

        // Captura hora actual
        minactual = rtc.getMinutes();
        horaactual = rtc.getHours();
        segactual= rtc.getSeconds();

        // Calcula diferencia de segundos
        segundosactuales= ((horaactual*3600) + (minactual*60) + segactual);
        segundossalida = ((hora*3600) + (minuto * 60) + segundo);
        vcuentaatras = segundossalida - segundosactuales;


}

// Calcula la hora actual en segundos
int Ftimetoseg(){
        signed int segcalc, horacalc, mincalc;
        byte segactual, minactual, horaactual;
        int fsegundos;
        // Captura hora actual
        minactual = rtc.getMinutes();
        horaactual = rtc.getHours();
        segactual= rtc.getSeconds();

        // Convierte a segundos
        fsegundos=(horaactual*3600);
        fsegundos=fsegundos+(minactual*60);
        fsegundos=fsegundos+(segactual);
        return fsegundos;
}

// Funcion para borrar pantalla "bonito"
void FborraPantalla(){
        tft.fillRoundRect(3,2,316,31,5,ILI9341_BLACK);
        tft.fillRoundRect(3,36,316,203,5,ILI9341_BLACK);


}
```

```
// Puesta en Hora Relog
void Prelog(){
        // Rutina setup relog
        // Borra Pantalla Menu Principal
        FborraPantalla();
        // Dibuja recuadros
        tft.drawRoundRect(2,1,318,33,7,ILI9341_BLUE);
        tft.drawRoundRect(2,35,318,205,5,ILI9341_BLUE);

        tft.setCursor(170, 11); tft.setTextColor(ILI9341_GREEN); tft.setTextSize(2);
tft.print(P("AJUSTE HORA"));

        // Actualiza el relog en pantalla
        t.update();
        if (t2on==false) {t2 = t.after(1000, relog); t2on=true;}

        // Prepara variables
        byte hora, minuto, segundos;
        hora = horatmp;
        minuto = mintmp;
        segundos = segtmp;

        // presenta datos iniciales
        if (hora < 10) {tft.setCursor(58, 120); tft.setTextColor(ILI9341_WHITE); tft.setTextSize(5);
tft.print(hora);}
                else {tft.setCursor(43, 120); tft.setTextColor(ILI9341_WHITE); tft.setTextSize(5);
tft.print(hora);}

        tft.setCursor(100, 120); tft.setTextColor(ILI9341_WHITE); tft.setTextSize(5);
tft.print(P(":"));

        if (minuto < 10) {tft.setCursor(145, 120); tft.setTextColor(ILI9341_WHITE);
tft.setTextSize(5); tft.print(minuto);}
                else {tft.setCursor(130, 120); tft.setTextColor(ILI9341_WHITE); tft.setTextSize(5);
tft.print(minuto);}

        tft.setCursor(190, 120); tft.setTextColor(ILI9341_WHITE); tft.setTextSize(5);
tft.print(P(":"));

        if (segundos < 10) {tft.setCursor(235, 120); tft.setTextColor(ILI9341_WHITE);
tft.setTextSize(5); tft.print(segundos);}
                else {tft.setCursor(220, 120); tft.setTextColor(ILI9341_WHITE); tft.setTextSize(5);
tft.print(segundos);}

        // Dibuja controles + -
        Ftriangulo(0,52,105,6,ILI9341_RED);
        Ftriangulo(0,140,105,6,ILI9341_RED);
        Ftriangulo(0,230,105,6,ILI9341_RED);

        Ftriangulo(2,52,170,6,ILI9341_RED);
        Ftriangulo(2,140,170,6,ILI9341_RED);
        Ftriangulo(2,230,170,6,ILI9341_RED);

        // Dibuja rectangulos
        tft.drawRect(41,113,64,50,ILI9341_RED);
        tft.drawRect(124,113,64,50,ILI9341_RED);
        tft.drawRect(215,113,64,50,ILI9341_RED);



        byte seleccion=1;
        boolean salir = false;
        do      {
                t.update();
                // Actualiza el relog de la esquina
                if (t2on==false) {t2 = t.after(1000, relog); t2on=true;}
                Pulsacion();
                lecturaIN();
```

```
                // salida
                if ((digitalRead(PUL1)) && (digitalRead(PUL3))) {salir=1; goto salto;}

                if (vPUL2==true) {
                        seleccion++;
                        vPUL2=false;
                        if (seleccion > 3) {seleccion=1;}
                }
                if (seleccion==1) {
                        tft.drawRect(41,113,64,50,ILI9341_YELLOW);
                        tft.drawRect(124,113,64,50,ILI9341_RED);
                        tft.drawRect(215,113,64,50,ILI9341_RED);
                }
                if (seleccion==2) {
                        tft.drawRect(41,113,64,50,ILI9341_RED);
                        tft.drawRect(124,113,64,50,ILI9341_YELLOW);
                        tft.drawRect(215,113,64,50,ILI9341_RED);
                }
                if (seleccion==3) {
                        tft.drawRect(41,113,64,50,ILI9341_RED);
                        tft.drawRect(124,113,64,50,ILI9341_RED);
                        tft.drawRect(215,113,64,50,ILI9341_YELLOW);
                }


                // Marcado de zonas elegidas para cambiar
                // Zona hora +
                if (((x1 > 50) && (x1 < 91) && (y1 > 80) && (y1 < 105))  || ((seleccion==1) &&
(vPUL1==true))) {
                        vPUL1=false;
                        x=0;y=0; x1=0; y1=0;
                        hora ++;
                        if (hora > 23) {hora=0;}
                        tft.fillRect(43,120,60,40,ILI9341_BLACK);
                        if (hora < 10) {tft.setCursor(58, 120); tft.setTextColor(ILI9341_WHITE);
tft.setTextSize(5); tft.print(hora);}
                                else {tft.setCursor(43, 120); tft.setTextColor(ILI9341_WHITE);
tft.setTextSize(5); tft.print(hora);}
                }

                // Zona Hora -
                if (((x1 > 48) && (x1 < 92) && (y1 > 171) && (y1 < 213)) || ((seleccion==1) &&
(vPUL3==true))) {
                        vPUL3=false;
                        x=0;y=0; x1=0; y1=0;
                        hora --;
                        if (hora > 250) {hora=23;}
                        tft.fillRect(43,120,60,40,ILI9341_BLACK);
                        if (hora < 10) {tft.setCursor(58, 120); tft.setTextColor(ILI9341_WHITE);
tft.setTextSize(5); tft.print(hora);}
                                else {tft.setCursor(43, 120); tft.setTextColor(ILI9341_WHITE);
tft.setTextSize(5); tft.print(hora);}
                }

                // Zona Minuto +
                if (((x1 > 133) && (x1 < 183) && (y1 > 77) && (y1 < 114))  || ((seleccion==2) &&
(vPUL1==true))) {
                        vPUL1=false;
                        x=0;y=0; x1=0; y1=0;
                        minuto ++;
                        if (minuto > 59) {minuto=0;}
                        tft.fillRect(130,120,60,40,ILI9341_BLACK);
                        if (minuto < 10) {tft.setCursor(145, 120); tft.setTextColor(ILI9341_WHITE);
tft.setTextSize(5); tft.print(minuto);}
                                else {tft.setCursor(130, 120); tft.setTextColor(ILI9341_WHITE);
tft.setTextSize(5); tft.print(minuto);}
                }
```

```
                // Zona Minuto -
                if (((x1 > 123) && (x1 < 183) && (y1 > 172) && (y1 < 212))  || ((seleccion==2) &&
(vPUL3==true))) {
                        vPUL3=false;
                        x=0;y=0; x1=0; y1=0;
                        minuto --;
                        if (minuto > 250) {minuto=59;}
                        tft.fillRect(130,120,60,40,ILI9341_BLACK);
                        if (minuto < 10) {tft.setCursor(145, 120); tft.setTextColor(ILI9341_WHITE);
tft.setTextSize(5); tft.print(minuto);}
                        else {tft.setCursor(130, 120); tft.setTextColor(ILI9341_WHITE);
tft.setTextSize(5); tft.print(minuto);}
                }

                // Zona Segundos +
                if (((x1 > 225) && (x1 < 276) && (y1 > 73) && (y1 < 117))  || ((seleccion==3) &&
(vPUL1==true))) {
                        vPUL1=false;
                        x=0;y=0; x1=0; y1=0;
                        segundos ++;
                        if (segundos > 59) {segundos=0;}
                        tft.fillRect(220,120,60,40,ILI9341_BLACK);
                        if (segundos < 10) {tft.setCursor(235, 120);
tft.setTextColor(ILI9341_WHITE); tft.setTextSize(5); tft.print(segundos);}
                                else {tft.setCursor(220, 120); tft.setTextColor(ILI9341_WHITE);
tft.setTextSize(5); tft.print(segundos);}
                }
                // Zona Segundos -
                if (((x1 > 230) && (x1 < 281) && (y1 > 170) && (y1 < 212))  || ((seleccion==3) &&
(vPUL3==true))) {
                        vPUL3=false;
                        x=0;y=0; x1=0; y1=0;
                        segundos --;
                        if (segundos > 250) {segundos=59;}
                        tft.fillRect(220,120,60,40,ILI9341_BLACK);
                        if (segundos < 10) {tft.setCursor(235, 120);
tft.setTextColor(ILI9341_WHITE); tft.setTextSize(5); tft.print(segundos);}
                        else {tft.setCursor(220, 120); tft.setTextColor(ILI9341_WHITE);
tft.setTextSize(5); tft.print(segundos);}
                }


        salto:;
        } while (salir==false);

        // Modifica Relog segun valores introducidos
        rtc.stopRTC(); //stop the RTC
        // Ajusta hora, minutos, seg
        rtc.setTime(hora, minuto, segundos);
        // Arranca relog
        rtc.startRTC();
        VESTADO=3;
}
```

```
//Lee una linea de un archivo de la SD, Contiene un string
String ReadFile(int Linea,char Ruta[]){
        int Lin=0;
        // linea a leer
        stResultado="";
        byte Bin;
        myFile = SD.open(Ruta);
        if (myFile) {
                while (myFile.available()) {                // Bucle hasta el final del fichero
                        Bin = myFile.read();
                        stResultado = stResultado + (char(Bin));
                        if (Bin==13){Lin++; myFile.read();
                                if (Lin!=Linea){stResultado="";}
                                if (Lin==Linea){break;}
                        }
                }
        }
        myFile.close(); return stResultado;
}



//Retorna num de lineas en el fichero
int numLineas(char Ruta[]){
        // Funcion que Retorna el numero de lineas existentes en un fichero
        int lin=0;
        byte lectura;
        myFile = SD.open(Ruta);
        if (myFile) {
                while (myFile.available()) {                                        // Bucle
hasta el final del fichero
                        lectura=myFile.read();
                        if (lectura == 13) {lin++; myFile.read();}      // la funcion myfile.read,
lee el caracter line feed
                }
        }
        myFile.close();return lin;
}
```

```
// Graba un evento en SD con la fecha del sistema
boolean anotaEvento(char Ruta[]){
        myFile = SD.open(Ruta, FILE_WRITE);
        // formatea el string de fecha actual con ceros si es necesario
        stFecha="";
        if (daytmp<10){stFecha = (P("0")); stFecha = stFecha + String(daytmp);}
                else {stFecha = stFecha + String(daytmp);}
        stFecha = stFecha + (P("/"));
        if (mestmp<10){stFecha = stFecha + (P("0")); stFecha = stFecha + String(mestmp);}
        else {stFecha = stFecha + String(mestmp);}
        stFecha = stFecha + (P("/"));
        stFecha = stFecha + String(yeartmp);


        if (myFile){
                myFile.println(stFecha);
                myFile.close();
                return true;}
                else{
                        myFile.close();
                        return false;
                }
}
```

```
// Actualiza el Relog en pantalla CADA SEGUNDO
void relog(){

        tft.setTextColor(ILI9341_YELLOW); tft.setTextSize(2);
        // Pone la hora
        horatmp = rtc.getHours();
        if (horaold != horatmp){
                tft.fillRect(5,11,31,16,ILI9341_BLACK);
                if (horatmp<10) {tft.setCursor(8, 11);tft.print(0);tft.setCursor(20, 11);
tft.print(horatmp);}
                        else            {tft.setCursor(8, 11);tft.print(horatmp);}
                horaold = horatmp;
                tft.setCursor(31, 11);tft.print(P(":"));
        }
        // Pone el minuto
        mintmp = rtc.getMinutes();
        if (minold != mintmp){
                tft.fillRect(41,11,31,16,ILI9341_BLACK);
                if (mintmp<10)  {tft.setCursor(41,11);tft.print(0);tft.setCursor(53, 11);
tft.print(mintmp);}
                        else            {tft.setCursor(41,11);tft.print(mintmp);}
                minold = mintmp;
                tft.setCursor(63, 11);tft.print(P(":"));
        }
        // Pone los segundos
        segtmp = rtc.getSeconds();
        if (segold!=segtmp){
                tft.fillRect(73,11,31,16,ILI9341_BLACK);
                if (segtmp<10)  {tft.setCursor(73, 11);tft.print(0);tft.setCursor(85, 11);
tft.print(segtmp);}
                        else            {tft.setCursor(73, 11); tft.print(segtmp);}
                segold = segtmp;

        }

        t.stop(t2);
        t2on=false;
}
```

```
// Recupera mensajes de Error de FLASH
char * Fmensaje(int index){
        // Borra Buffer previo
        for (byte i=0; i<20; i++){bufferM[i] = *("");}
        // Carga en buffer el mensaje de error solicitado
        strcpy_P(bufferM, (char*)pgm_read_word(&(MERROR[index])));
        return bufferM;
}



// Funcion Aux para imprimir mensajes centrados en pantalla
int FposX (int q, int campo, int sizetext){
        // x --> Pixel inicio campo en pantalla
        // campo --> Tamaño campo en pantalla en pixels (5+1 en tamaño 1)
        // longtext --> longitud del texto en caracteres
        // sizetext --> Multiplicador de la fuente
        byte numchars = Fcuentachars(bufferM);                                          // cuenta
los caracteres del mensaje
        int posx = (q + ((campo - ((numchars * 6) * sizetext)) /2));
        return posx;
}



// funcion para dibujar un triangulo
void Ftriangulo (byte up, int x, int y, int size,  uint16_t color) {
        /* byte up -->  0  Flecha arriba
                                        1 Flecha derecha
                                        2 Flecha abajo
                                        3 Flecha izquierda
                int x ---> Coordenada X Base triangulo
                int y ---> Coordenada Y Base triangulo
                int size -> 1  Caracter 8x5
                                        2  Caracter 16x10
                                        3  Caracter 24x15
        */

if (up==0) {tft.fillTriangle(x,y,(x+(3*size)),(y-(6*size)),(x+6*size),y,color);}
if (up==2) {tft.fillTriangle(x,y,(x+(3*size)),(y+(6*size)),(x+6*size),y,color);}

}
```