```
/*

Programa creado sobre MinIMU-9-Arduino-AHRS.
MRS.

MinIMU-9-Arduino-AHRS
Pololu MinIMU-9 + Arduino AHRS (Attitude and Heading Reference System)

Copyright (c) 2011 Pololu Corporation.
http://www.pololu.com/

MinIMU-9-Arduino-AHRS is based on sf9domahrs by Doug Weibel and Jose Julio:
http://code.google.com/p/sf9domahrs/

sf9domahrs is based on ArduIMU v1.5 by Jordi Munoz and William Premerlani, Jose
Julio and Doug Weibel:
http://code.google.com/p/ardu-imu/


*/

// Uncomment the below line to use this axis definition:
// X axis pointing forward
// Y axis pointing to the right
// and Z axis pointing down.
// Positive pitch : nose up
// Positive roll : right wing down
// Positive yaw : clockwise
int SENSOR_SIGN[9] = { 1, 1, 1, -1, -1, -1, 1, 1, 1 }; //Correct directions x,y,z -
gyro, accelerometer, magnetometer
int SENSOR_SIGNd[9] = { 1, 1, 1, -1, -1, -1, 1, 1, 1 }; //Correct directions x,y,z -
gyro, accelerometer, magnetometer
// Uncomment the below line to use this axis definition:
// X axis pointing forward
// Y axis pointing to the left
// and Z axis pointing up.
// Positive pitch : nose down
// Positive roll : right wing down
// Positive yaw : counterclockwise
//int SENSOR_SIGN[9] = {1,-1,-1,-1,1,1,1,-1,-1}; //Correct directions x,y,z - gyro,
//accelerometer, magnetometer
//int SENSOR_SIGN[9] = {1,-1,-1,-1,1,1,1,-1,-1}; //Correct directions x,y,z - gyro,
//accelerometer, magnetometer
//int SENSOR_SIGNd[9] = {1,-1,-1,-1,1,1,1,-1,-1}; //Correct directions x,y,z - gyro,
//accelerometer, magnetometer

// tested with Arduino Uno with ATmega328 and Arduino Duemilanove with ATMega168

#include <Wire.h>

#include <LiquidCrystal.h> // F Malpartida's NewLiquidCrystal library

#define caraZowi 0 // =1: Operador ve CARA de Zowi. =0: Operador percibe CULO.

// LSM303 accelerometer: 8 g sensitivity
// 3.9 mg/digit; 1 g = 256
```

```
#define GRAVITY 256 //this equivalent to 1G in the raw data coming from the
accelerometer

#define ToRad(x) ((x)*0.01745329252) // *pi/180
#define ToDeg(x) ((x)*57.2957795131) // *180/pi

// L3G4200D gyro: 2000 dps full scale
// 70 mdps/digit; 1 dps = 0.07
#define Gyro_Gain_X 0.07 //X axis Gyro gain
#define Gyro_Gain_Y 0.07 //Y axis Gyro gain
#define Gyro_Gain_Z 0.07 //Z axis Gyro gain

#define Gyro_Scaled_X(x) ((x)*ToRad(Gyro_Gain_X))
//Return the scaled ADC raw data of the gyro in radians for second
#define Gyro_Scaled_Y(x) ((x)*ToRad(Gyro_Gain_Y))
//Return the scaled ADC raw data of the gyro in radians for second
#define Gyro_Scaled_Z(x) ((x)*ToRad(Gyro_Gain_Z))
//Return the scaled ADC raw data of the gyro in radians for second

#define Gyro_Scaled_Xd(x) ((x)*ToRad(Gyro_Gain_X))
//Return the scaled ADC raw data of the gyro in radians for second
#define Gyro_Scaled_Yd(x) ((x)*ToRad(Gyro_Gain_Y))
//Return the scaled ADC raw data of the gyro in radians for second
#define Gyro_Scaled_Zd(x) ((x)*ToRad(Gyro_Gain_Z))
//Return the scaled ADC raw data of the gyro in radians for second

// LSM303 magnetometer calibration constants; use the Calibrate example from
// the Pololu LSM303 library to find the right values for your board
#define M_X_MIN -2666
#define M_Y_MIN -3446
#define M_Z_MIN -4782
#define M_X_MAX 4625
#define M_Y_MAX 3695
#define M_Z_MAX 4383

#define M_X_MINd -3212
#define M_Y_MINd -3273
#define M_Z_MINd -2653
#define M_X_MAXd 3505
#define M_Y_MAXd 3260
#define M_Z_MAXd 4601

#define Kp_ROLLPITCH 0.02
#define Ki_ROLLPITCH 0.00002
#define Kp_YAW 1.2
#define Ki_YAW 0.00002

// Mesa botonera: *** ELIMINAR LED Y BUZZER
#define pin_pulsador1 9
#define pin_interruptor 11
//#define pin_led 7 //OLD: 12. 7 para debug en cambio a versión raspberry.
//#define pin_buzz 8

#define umbralError 0.8
#define ErrorArtMAX 1

/*For debugging purposes*/
```

```
//OUTPUTMODE=1 will print the corrected data,
//OUTPUTMODE=0 will print uncorrected data of the gyros (with drift)
#define OUTPUTMODE 1

//#define PRINT_DCM 0         //Will print the whole direction cosine matrix
#define PRINT_ANALOGS 0 //Will print the analog raw data
#define PRINT_EULER 1 //Will print the Euler angles Roll, Pitch and Yaw

#define STATUS_LED 13

//Pines pinales de carrera zapateros
#define FCV1 2
#define FCH1 3
#define FCV2 4
#define FCH2 5

//pines leds
#define Led_rojo 14
#define Led_verde 16
#define Led_azul 15
#define Led_reserva 17
#define Pin_buzz 8

#define round(a) (int)(a + 0.5) //Para redondear

float G_Dt = 0.02; // Integration time (DCM algorithm)
                   // We will run the integration loop at 50Hz if possible

long timer = 0; //general purpuse timer
long timer_old;
long timer_restart = 0;
long timer24 = 0; //Second timer used to print values
int AN[6]; //array that stores the gyro and accelerometer data
int AN_OFFSET[6] = { 0, 0, 0, 0, 0, 0 };
//Array that stores the Offset of the sensors

int ANd[6]; //array that stores the gyro and accelerometer data
int AN_OFFSETd[6] = { 0, 0, 0, 0, 0, 0 };
//Array that stores the Offset of the sensors

int gyro_x;
int gyro_y;
int gyro_z;
int accel_x;
int accel_y;
int accel_z;
int magnetom_x;
int magnetom_y;
int magnetom_z;
float c_magnetom_x;
float c_magnetom_y;
float c_magnetom_z;
float MAG_Heading;

int gyro_xd;
int gyro_yd;
int gyro_zd;
```

```
int accel_xd;
int accel_yd;
int accel_zd;
int magnetom_xd;
int magnetom_yd;
int magnetom_zd;
float c_magnetom_xd;
float c_magnetom_yd;
float c_magnetom_zd;
float MAG_Headingd;

float Accel_Vector[3] = { 0, 0, 0 }; //Store the acceleration in a vector
float Gyro_Vector[3] = { 0, 0, 0 }; //Store the gyros turn rate in a vector
float Omega_Vector[3] = { 0, 0, 0 }; //Corrected Gyro_Vector data
float Omega_P[3] = { 0, 0, 0 }; //Omega Proportional correction
float Omega_I[3] = { 0, 0, 0 }; //Omega Integrator
float Omega[3] = { 0, 0, 0 };

float Accel_Vectord[3] = { 0, 0, 0 }; //Store the acceleration in a vector
float Gyro_Vectord[3] = { 0, 0, 0 }; //Store the gyros turn rate in a vector
float Omega_Vectord[3] = { 0, 0, 0 }; //Corrected Gyro_Vector data
float Omega_Pd[3] = { 0, 0, 0 }; //Omega Proportional correction
float Omega_Id[3] = { 0, 0, 0 }; //Omega Integrator
float Omegad[3] = { 0, 0, 0 };

// Euler angles pierna izquierda
float roll;
float pitch;
float yaw;

// Euler angles pierna derecha
float rolld;
float pitchd;
float yawd;

float errorRollPitch[3] = { 0, 0, 0 };
float errorYaw[3] = { 0, 0, 0 };

float errorRollPitchd[3] = { 0, 0, 0 };
float errorYawd[3] = { 0, 0, 0 };

unsigned int counter = 0;
byte gyro_sat = 0;
byte gyro_satd = 0;

float DCM_Matrix[3][3] = {
    { 1, 0, 0 }, { 0, 1, 0 }, { 0, 0, 1 }
};

float DCM_Matrixd[3][3] = {
    { 1, 0, 0 }, { 0, 1, 0 }, { 0, 0, 1 }
};

float Update_Matrix[3][3] = { { 0, 1, 2 }, { 3, 4, 5 }, { 6, 7, 8 } }; //Gyros here
float Update_Matrixd[3][3] = { { 0, 1, 2 }, { 3, 4, 5 }, { 6, 7, 8 } }; //Gyros here

float Temporary_Matrix[3][3] = {
```

```
    { 0, 0, 0 }, { 0, 0, 0 }, { 0, 0, 0 }
};

float Temporary_Matrixd[3][3] = {
    { 0, 0, 0 }, { 0, 0, 0 }, { 0, 0, 0 }
};

// !! botonera
int estado = 0;
long timerLed = 0;
int freqLed = 1;
int pulsador_old = 0;
int pulsador_apagar = 1;
int num_vueltas = 0;

double comprPitch_izq = 0;
double comprRoll_izq = 0;
//double comprYaw_izq=0;
double comprPitch_der = 0;
double comprRoll_der = 0;
//double comprYaw_der=0;
double contadorEstabilizacion = 0;
int flagBoton = 0;

double nuevo90_der = 0;
double nuevo90_izq = 0;
double nuevo0_der = 0;
double nuevo0_izq = 0;

double err_pie_izq = 0;
double err_cadera_izq = 0;
double err_pie_der = 0;
double err_cadera_der = 0;

double pos_old_pie_izq = 0;
double pos_old_cadera_izq = 0;
double pos_old_pie_der = 0;
double pos_old_cadera_der = 0;

int cadera_izq_OK = 0;
int cadera_der_OK = 0;
int pie_izq_OK = 0;
int pie_der_OK = 0;

int primeraVez = 0;
int numeroData = 0;
int flag_espera_zum = 0;
double timer_Zum = 0;
double timer_WD;

char lectura_RB = 0;

int FCH_1 = 0;
int FCV_1 = 0;
int FCH_2 = 0;
int FCV_2 = 0;
```

```
//Comunicaciones
int slave_add = 1; //Direccion I2C del esclavo (Zowi)
char resultado = '0';
int result = 0;

LiquidCrystal lcd(0);

char data[4] = { 0, 0, 0, 0 };
char dataTRICK[4] = { 0, 0, 0, 0 };
double dataError[4] = { 0, 0, 0, 0 };

char trama[30] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                   0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
char tramaError[40] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0 };

String str;
char temp[2];

int dataINT[4] = { 0, 0, 0, 0 };
//Comandos I2C
char C0[7] = "$IZUM:"; //Comando arrancar y cargar programa calibración ZUM
char C1[7] = "$ROFC:"; //Comando leer offset en zowi
char C2[7] = "$WOFC:"; //Comando escribir offset calibracion en zowi
char C3[7] = "$M90C:"; //Comando mover 4 servos a 90
char C4[7] = "$MHOC:"; //Comando mover 4 servos a HOME (nuevo90 calibrado)
char C5[7] = "$MSSC:"; //Comando mover 4 servos a posiciones indicadas
char C6[7] = "$MSxC:"; //Comando mover servo indicado a posicion indicada
char C7[7] = "$WERC:"; //Comando enviar error a RB
char C8[7] = "$FZUM:"; //Comando programar test en ZUM
char C9[7] = "$ROFF:"; //Comando apagar RB
char C10[7] = "$WSQL:"; //Orden de escribir a MySQL
int dato = 0;

void setup()
{

    lcd.begin(16, 2);

    pinMode(pin_interruptor, INPUT_PULLUP); //interruptor mesa botonera
    pinMode(pin_pulsador1, INPUT_PULLUP); //pulsador botonera
    //  pinMode(pin_led, OUTPUT);                //led mesa botonera
    pinMode(Pin_buzz, OUTPUT); //buzzer
    pinMode(Led_rojo, OUTPUT); //leds
    pinMode(Led_azul, OUTPUT); //leds
    pinMode(Led_verde, OUTPUT); //leds
    pinMode(Led_reserva, OUTPUT); //leds

    pinMode(FCH1, INPUT_PULLUP); //Final carrera horizontal 1
    pinMode(FCV1, INPUT_PULLUP); //Final carrera vertical 1
    pinMode(FCH2, INPUT_PULLUP); //Final carrera horizontal 2
    pinMode(FCV2, INPUT_PULLUP); //Final carrera vertical 2

    Serial.begin(115200);
    Serial1.begin(9600);
```

```
pinMode(STATUS_LED, OUTPUT); // Status LED

I2C_Init();

Serial1.println("Banco calibracion Zowi");

lcd.clear();
lcd.setCursor(0, 0);
lcd.print("    System      ");
lcd.setCursor(0, 1);
lcd.print("  start-up...   ");
digitalWrite(Led_rojo, LOW);
digitalWrite(Led_azul, LOW);
digitalWrite(Led_verde, LOW);
digitalWrite(Led_reserva, LOW);
digitalWrite(Pin_buzz, LOW);
digitalWrite(STATUS_LED, LOW);
delay(1500);

Accel_Init();
Serial1.println("acel init done");
Compass_Init();
Serial1.println("compass initdone");
Gyro_Init();
Serial1.println("gyro init done");

delay(20);

for (int i = 0; i < 32; i++) // We take some readings...
{
    Read_Gyro();
    Read_Accel();
    for (int y = 0; y < 6; y++) // Cumulate values
    {
        AN_OFFSET[y] += AN[y];
        AN_OFFSETd[y] += ANd[y];
    }
    delay(20);
}

for (int y = 0; y < 6; y++) {
    AN_OFFSET[y] = AN_OFFSET[y] / 32;
    AN_OFFSETd[y] = AN_OFFSETd[y] / 32;
}

AN_OFFSET[5] -= GRAVITY * SENSOR_SIGN[5];
AN_OFFSETd[5] -= GRAVITY * SENSOR_SIGNd[5];

delay(2000);
digitalWrite(STATUS_LED, HIGH);

timer = millis();
delay(20);
counter = 0;

estado = 97;
primeraVez = 1;
```

```
}

void loop() //Main Loop (Maquina estados)
{
    if (digitalRead(pin_interruptor) == LOW && estado != 97 && estado != 98
                                            && estado != 96) {
        estado = 0;
        delay(10);
    }

    switch (estado) {
    case 96: //Apagado de RPI
        WriteComm(9);
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("     System     ");
        lcd.setCursor(0, 1);
        lcd.print("Shutting down...");
        Serial1.println("Apagando");
        delay(15000);
        lcd.clear();
        lcd.noDisplay();
        digitalWrite(Led_rojo, LOW);
        parpadeoLed(0); //desactivamos parpadeo led azul
        digitalWrite(Led_azul, LOW);
        digitalWrite(Led_verde, LOW);
        digitalWrite(Led_reserva, LOW);
        digitalWrite(Pin_buzz, LOW);
        digitalWrite(STATUS_LED, LOW);
        break;

    case 97: //Arranque de RPI //esperamos que la RPI envíe un 1 a la Mega
        parpadeoLed(1);

        if (Serial.available() >= 1) {
            lectura_RB = Serial.read();
            Serial1.println("lectura serial RB:");
            Serial1.println(lectura_RB);
        }

        //Dependiendo del estado del switch vamos a un estado u otro:
        if ((lectura_RB == '1') && (digitalRead(pin_interruptor) == HIGH)) {
            estado = 14;
            sonidostart();
            parpadeoLed(0);
        }
        else if ((lectura_RB == '1') && (digitalRead(pin_interruptor)) == LOW) {
            estado = 0;
            sonidostart();
            parpadeoLed(0);
        }

        break;

    //Llegamos a estado 0 al subir el switch desde cualquier estado
    case 0:
        digitalWrite(Led_rojo, LOW);
```

```
    parpadeoLed(0); //desactivamos parpadeo led azul
    digitalWrite(Led_azul, LOW);
    digitalWrite(Led_verde, LOW);
    digitalWrite(Led_reserva, LOW);
    digitalWrite(Pin_buzz, LOW);
    digitalWrite(STATUS_LED, LOW);

    if (digitalRead(pin_pulsador1) == 1 && pulsador_apagar == 1) {
    //Apagado del sistema:
        delay(5);
        timer_restart = millis();
        pulsador_apagar = 0;
    }
    else if (digitalRead(pin_pulsador1) == 0) {
        timer_restart = millis();
        pulsador_apagar = 1;
    }

    if ((millis() - timer_restart) >= 2000 && digitalRead(pin_pulsador1) == 1) {
        //si mantenemos el pulsador 2 segundos apagamos sistema
        estado = 96; //estado para apagar
        break;
    }

    lcd.setCursor(0, 0);
    lcd.print("     Set to     ");
    lcd.setCursor(0, 1);
    lcd.print(" CALIBRATE pos. ");
    ////Serial1.println("Estado 0");
    /* interruptor en posición 0. Desde cualquier estado podremos volver
        al estado 0 poniendolo a 0. Los servos energizados reciben el valor 90,
        el operario deberá montar y atornillar los piés del ZOWI en la posición
        más correcta posible.
    Se deben colocar los zapatos en la plataforma ntes de bajar el interruptor*/

    //al cambiar el interruptor pasamos al funcionamiento del sistema
    if (digitalRead(pin_interruptor) == HIGH) {
        estado = 14;
        delay(10);
        primeraVez = 1;
    };

    break;

case 14: //calibracion horizontal tras pulsar el boton
    Serial1.println("Estado 14: Espera colocar zapatos y pulsar boton");
    //deben estar los zapatos insertados para calibrar las imus
    FCH_1 = digitalRead(FCH1);
    FCH_2 = digitalRead(FCH2);

    if (FCH_1 == HIGH || FCH_2 == HIGH) {

        lcd.setCursor(0, 0);
        lcd.print(" Place shoes in ");
        lcd.setCursor(0, 1);
        lcd.print(" Horizontal box ");
    }
```

```
    if (FCH_1 == LOW && FCH_2 == LOW) {

        lcd.setCursor(0, 0);
        lcd.print("  Press to set  ");
        lcd.setCursor(0, 1);
        lcd.print("   HORIZONTAL   ");

        if (digitalRead(pin_pulsador1) == 1) {
            estado = 10;
            delay(10);
            primeraVez = 1;
        };
    }

    break;

case 10: //inicio de calibración IMUS en horizontal
    Serial1.println("Estado 10: Iniciando Calib Horizontal");
    //lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("  Calibration   ");
    lcd.setCursor(0, 1);
    lcd.print(" in progress... ");
    estado = 1;
    break;

case 1:
    Serial1.println("Estado 1");

    init_imus();

    //Preparamos para lecturas comprobación imu
    timer = millis();
    comprPitch_izq = 0;
    comprRoll_izq = 0;
    comprPitch_der = 0;
    comprRoll_der = 0;
    contadorEstabilizacion = 0;
    estado = 11;
    break;

case 11: //Comprobacion de offset
    ////Serial1.println("Estado 11");
    FCH_1 = digitalRead(FCH1);
    FCH_2 = digitalRead(FCH2);
    //se realiza la calibración con lecturas a 50Hz ya que es
  //la frecuencia a la que trabajan las IMUS
    if ((millis() - timer) >= 20) // Main loop runs at 50Hz
    {
        if (FCH_1 == LOW && FCH_2 == LOW) {
        //aseguramos que los zapatos están en zapatero horizontal
            calculosIMU();

            contadorEstabilizacion++;

            if (contadorEstabilizacion > 100) {
```

```
                if (caraZowi) {
                    comprPitch_der += pitch;
                    comprPitch_izq += pitchd;
                    comprRoll_der += roll;
                    comprRoll_izq += rolld;
                }

                else {
                    comprPitch_der += pitchd;
                    comprPitch_izq += pitch;
                    comprRoll_der += rolld;
                    comprRoll_izq += roll;
                }
            }

            if (contadorEstabilizacion > 150) {
                comprPitch_der = abs(ToDeg(comprPitch_der) / 50);
                comprPitch_izq = abs(ToDeg(comprPitch_izq) / 50);
                comprRoll_izq = abs(ToDeg(comprRoll_izq) / 50);
                comprRoll_der = abs(ToDeg(comprRoll_der) / 50);
                if ((comprPitch_der > umbralError) || (comprPitch_izq >
                    umbralError) || (comprRoll_der > umbralError) ||
                    (comprRoll_izq > umbralError)) //Media
                { //Error en offset imus
                    estado = 12;
                    Serial1.println("ERROR. > a ESTADO 12 ");
                    //delay(1000);
                }
                else { //Offset en 0° IMUs OK
                    estado = 20;
                    nuevo0_der = comprPitch_der;
                    nuevo0_izq = comprPitch_izq;

                    Serial1.println("Nuevo 0:");
                    Serial1.println(nuevo0_der);
                    Serial1.println(nuevo0_izq);
                }
            }
        }
        else { //Zapato no está en zapatero horizontal
            Serial1.println("Zapatos mal puestos en posicion Horizontal");

            //lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print(" Place shoes in ");
            lcd.setCursor(0, 1);
            lcd.print(" Horizontal box ");

            delay(1000);

            estado = 14;
        }
    }
    break;

case 12: //Estado error en offset 0° de imus No se han calibrado correctamente
```

```
    sonidoNOK();
    Serial1.println("Repite Posicion Horizontal");

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(" Process failed ");
    lcd.setCursor(0, 1);
    lcd.print("     Retry      ");

    delay(1000);

    estado = 14;
    break;

case 20: //Estado calibración 0°ok, pasamos a calibración vertical
    sonidoOK();
    Serial1.println("Calibracion IMU Horizontal ok");

    //lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(" Place shoes in ");
    lcd.setCursor(0, 1);
    lcd.print("  Vertical box  ");

    contadorEstabilizacion = 0;
    estado = 21; //zapatos a 90 "calibracion"
    timer = millis();
    timerLed = millis();
    delay(20);
    counter = 0;
    flagBoton = 0;
    break;

case 21: //Estado para offset de 90°(calibración zapatos en vertical)

    FCV_1 = digitalRead(FCV1);
    FCV_2 = digitalRead(FCV2);

    if (FCV_1 == LOW && FCV_2 == LOW && flagBoton == 0) {
        //los zapatos deben estar insertados

        //lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("  Press to set  ");
        lcd.setCursor(0, 1);
        lcd.print("    VERTICAL    ");
    };

    if ((FCV_1 == HIGH || FCV_2 == HIGH) && flagBoton == 0) {

        //lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print(" Place shoes in ");
        lcd.setCursor(0, 1);
        lcd.print("  Vertical box  ");
    };
```

```
if ((millis() - timer) >= 20) // Main loop runs at 50Hz
                              //calibración zapatos en vertical
{
    calculosIMU();

    if (digitalRead(pin_pulsador1) == 1) //esperamos pulsación para calibrar
    {
        flagBoton = 1;
    }

    if (flagBoton == 1) {
        if (FCV_1 == LOW && FCV_2 == LOW) { //Zapatos en zapatero vertical
            contadorEstabilizacion++;

            //   lcd.clear();
            //lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("  Calibration   ");
            lcd.setCursor(0, 1);
            lcd.print(" in progress... ");

            if (contadorEstabilizacion > 50) //esperamos a que se estabilice
            {
                if (caraZowi) {
                    comprRoll_izq += rolld;
                    comprRoll_der += roll;
                }
                else {
                    comprRoll_izq += roll;
                    comprRoll_der += rolld;
                }
            }

            if (contadorEstabilizacion > 100)
            //se han obtenido 50 medidas tras estabilizar
            {

                comprRoll_izq = ToDeg(comprRoll_izq / 50);
                comprRoll_der = ToDeg(comprRoll_der / 50);

                nuevo90_izq = comprRoll_izq;
                nuevo90_der = comprRoll_der;

                Serial1.println("Calibracion IMU Vertical Ok");
                      //calibración vertical hecha
                //pasamos a colocar zapatos
                //lcd.clear();
                lcd.setCursor(0, 0);
                lcd.print("Connect+ON ZOWI ");
                lcd.setCursor(0, 1);
                lcd.print("Put shoes+press ");

                Serial1.println("Nuevo 90:");
                Serial1.println(nuevo90_der);
                Serial1.println(nuevo90_izq);

                flagBoton = 0;
```

```
                    estado = 2;
                    Serial1.println("Coloca zap, enchufa y pulsa
                                    boton para ajustar ");
                    sonidoOK();
                    timer = millis();
                    timerLed = millis();
                    delay(20);
                    counter = 0;
                    //Serial1.println("Aqui1");
                }
            }
            else { //Zapatos no están en zapatero vertical
                Serial1.println("Zapatos mal colocados en posicion vertical");

                lcd.setCursor(0, 0);
                lcd.print(" Place shoes in ");
                lcd.setCursor(0, 1);
                lcd.print("  Vertical box  ");

                delay(1000);

                lcd.clear();

                estado = 21;
            }
        }
    }

    break;

case 2:
    Serial1.println("case2");
    // el led parpadea, se espera que se coloquen los zapatos
    // en los pies de ZOWI.
        //Salimos del estado actuando sobre el pulsador.
    if ((millis() - timer) >= 20) // Main loop runs at 50Hz
    {
        calculosIMU();
    }
    parpadeoLed(1); //sucede cada 1*500ms

    if (digitalRead(pin_pulsador1) == 1) //AQUIIIIIII!!!!!
    { //Arrancamos calibracion con el boton
        parpadeoLed(0);
        estado = 22;
        cadera_izq_OK = 0;
        cadera_der_OK = 0;
        pie_izq_OK = 0;
        pie_der_OK = 0;
        num_vueltas = 0;

        digitalWrite(Led_rojo, LOW);
        digitalWrite(Led_verde, LOW);

        Serial1.println("Iniciando comunicacion con Zum Zowi...");

        //lcd.clear();
```

```
        lcd.setCursor(0, 0);
        lcd.print("   Checking     ");
        lcd.setCursor(0, 1);
        lcd.print(" connection...  ");
    };
    break;

case 22:
    Serial1.println("case22");
    if ((millis() - timer) >= 20) // Main loop runs at 50Hz
    {
        calculosIMU();
    }

    if (flag_espera_zum == 0) {
        timer_Zum = millis();
        flag_espera_zum = 1;

        while (Serial.available() >= 1) {
            Serial1.println(Serial.read());
        }

        WriteComm(0); //Inicio comunicación Raspberry - ZUM mandamos IZUM
    }

    if (Serial.available() >= 1) {

        lectura_RB = Serial.read();
        //Serial1.println(lectura_RB);
        if (lectura_RB == 'B') {

            //lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("Communication OK");
            lcd.setCursor(0, 1);
            lcd.print("Calibrating...  ");

            Serial1.println("Comunicacion establecida.");
            Serial1.println("Calibrando...");
            estado = 3;
            flag_espera_zum = 0;

            while (Serial.available() >= 1) {
                Serial1.println(Serial.read());
            } //vaciamos serial

            break;
        }

        else if (lectura_RB == 'M') {
              //si la RPI envía una M indica que no se
              //ha cargado el programa de calibración en Zowi

            //lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("Connection fault");
            lcd.setCursor(0, 1);
```

```
            lcd.print("   Check ZOWI   ");

            estado = 2;
            flag_espera_zum = 0;

            Serial1.println("MAL. No cargada ZUM");
            Serial1.println("Leyendo serial (3 veces):");
              //para vaciar últimos datos del serial
            Serial1.println(Serial.read());
            Serial1.println(Serial.read());
            Serial1.println(Serial.read());
        }
    }

    else if (((millis() - timer_Zum) >= 40000)) {
    //si no responde en x secs: volver a intentar
        Serial1.println("TIMEOUT");
        Serial1.println("Pulsa para volver a intentar...");

        //lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print(" Timeout. Check ");
        lcd.setCursor(0, 1);
        lcd.print(" Zowi and push  ");

        estado = 2;
        flag_espera_zum = 0;
    }

    break;

case 3: //Estado calibración articulaciones:
  // Los zapatos ya están colocados en los pies y se comienza
  // a ajustar offset de los servos.
    parpadeoLed(4);
    if ((millis() - timer) >= 20) // Main loop runs at 50Hz
    {
        calculosIMU();
        //Calculo errores de posicion respecto a los 0° y los 90°
        //ajustados para el banco

        if (caraZowi) { //dependiendo si está puesto el zowi de cara o de culo
            err_pie_izq = nuevo90_izq - ToDeg(rolld);
            err_cadera_izq = nuevo0_izq - ToDeg(pitchd);
            err_pie_der = nuevo90_der - ToDeg(roll);
            err_cadera_der = nuevo0_der - ToDeg(pitch);
        }

        else {
            err_pie_izq = nuevo90_izq - ToDeg(roll);
            err_cadera_izq = nuevo0_izq - ToDeg(pitch);
            err_pie_der = nuevo90_der - ToDeg(rolld);
            err_cadera_der = nuevo0_der - ToDeg(pitchd);
        }

        //Evaluamos error
        if (abs(err_cadera_izq) > ErrorArtMAX) {
```

```
            cadera_izq_OK = 0;
        }
        else {
            cadera_izq_OK = 1;
        }
        if (abs(err_pie_izq) > ErrorArtMAX) {
            pie_izq_OK = 0;
        }
        else {
            pie_izq_OK = 1;
        }
        if (abs(err_cadera_der) > ErrorArtMAX) {
            cadera_der_OK = 0;
        }
        else {
            cadera_der_OK = 1;
        }
        if (abs(err_pie_der) > ErrorArtMAX) {
            pie_der_OK = 0;
        }
        else {
            pie_der_OK = 1;
        }

        num_vueltas++; //para case de calibra servos

        if (!(cadera_der_OK == 1 && cadera_izq_OK == 1 &&
               pie_der_OK == 1 && pie_izq_OK == 1)) {
              //Si articulaciones con error de posicion->calibramos
            if (num_vueltas <= 330)
              //Para 5 iteraciones de calibración en servos se llega a 330
            {
                calibra_servos();
            }
            else {
                estado = 40;
            };
        }
        else {
            estado = 40;
        };
    }

    break;

case 40: //Estado chequeo de la calibracion
    num_vueltas = 0;

    Serial.read();
    Serial.read();
    Serial.read();
    Serial.read();

    if (cadera_der_OK==1 && cadera_izq_OK==1 && pie_der_OK==1 && pie_izq_OK==1) {
        //Calibracion  articulaciones OK
        resultado = '1';
        result = 1;
```

```
//          parpadeoLed(0);
//          digitalWrite(Led_rojo,LOW);
//          digitalWrite(Led_verde, HIGH);
//          sonidoOK();

Serial1.println("Calibración OK");

//lcd.clear();
lcd.setCursor(0, 0);
lcd.print(" Calibration OK ");
lcd.setCursor(0, 1);
lcd.print(" Writing EEPROM ");

cadera_der_OK = 0;
cadera_izq_OK = 0;
pie_der_OK = 0;
pie_izq_OK = 0;

dataError[0] = err_cadera_izq;
dataError[1] = err_cadera_der;
dataError[2] = err_pie_izq;
dataError[3] = err_pie_der;
WriteComm(7); //Comando WERC: errores de calibracion para RB)

data[0] = pos_old_cadera_der;
data[1] = pos_old_pie_der;
data[2] = pos_old_cadera_izq;
data[3] = pos_old_pie_izq;
WriteComm(22);
      //Comando WOFF Pasamos a Zowi las posiciones HOME
delay(300);

data[0] = pos_old_cadera_der;
data[1] = pos_old_pie_der;
data[2] = pos_old_cadera_izq;
data[3] = pos_old_pie_izq;
WriteComm(2);
delay(300);

WriteComm(10); //Comando WSQL Salva base datos

Serial1.print("Cadera Izq :");
Serial1.println(pos_old_cadera_izq);
Serial1.print("Pie Izq :");
Serial1.println(pos_old_pie_izq);
Serial1.print("Cadera Der :");
Serial1.println(pos_old_cadera_der);
Serial1.print("Pie Der :");
Serial1.println(pos_old_pie_der);

Serial1.print("Error cadera izq :");
Serial1.println(err_cadera_izq);
Serial1.print("Error pie izq :");
Serial1.println(err_pie_izq);
Serial1.print("Error cadera der :");
Serial1.println(err_cadera_der);
Serial1.print("Error pie der :");
```

```
    Serial1.println(err_pie_der);

    delay(500);

    lcd.setCursor(0, 1);
    lcd.print("Loading Test Prg");

    WriteComm(8); //Comando FZUM Cargar programa Test

    lectura_RB = 0;
    timer_WD = millis();
    do {
        if (Serial.available() >= 1) {
            lectura_RB = Serial.read();
            Serial1.println(lectura_RB);
        }
        if ((millis() - timer_WD) >= 20000) {
            sonidoNOK();
            parpadeoLed(0);
            digitalWrite(Led_rojo, HIGH);
            digitalWrite(Led_verde, LOW);
            break;
        }

    } while ((lectura_RB != 'M') && (lectura_RB != 'B'));
          //esperamos a que la RPI envíe B o M
          //dependiendo de carga programa Test

    if (lectura_RB == 'B') {
        sonidoOK();
        parpadeoLed(0);
        digitalWrite(Led_rojo, LOW);
        digitalWrite(Led_verde, HIGH);
    }

    else if (lectura_RB == 'M') {
        sonidoNOK();
        parpadeoLed(0);
        digitalWrite(Led_rojo, HIGH);
        digitalWrite(Led_verde, LOW);
    }

    lectura_RB = 0;

    Serial.read();
    Serial.read();
    Serial.read();
    Serial.read();

    delay(1000);
    estado = 4; //Realizar otra calibración si pulsador
}
else { //La calibracion de alguna articulacion ha fallado
    sonidoNOK();
    resultado = '0';
    result = 0;
    parpadeoLed(0);
```

```
        digitalWrite(Led_rojo, HIGH);
        digitalWrite(Led_verde, LOW);

        dataError[0] = err_cadera_izq;
        dataError[1] = err_cadera_der;
        dataError[2] = err_pie_izq;
        dataError[3] = err_pie_der;
        WriteComm(7); //Comando WERC: errores de calibracion para RB)

        data[0] = 90;
        data[1] = 90;
        data[2] = 90;
        data[3] = 90; //si sale mal, mandamos 90 En BBDD = 0 (90-90)
        WriteComm(2); //Comando WOFF Pasamos a Zowi las posiciones HOME
        WriteComm(10); //Salva base datos

        //lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("   CALIBRATION  ");
        lcd.setCursor(0, 1);
        lcd.print("     FAILED     ");

        Serial1.println("Calibracion MAL");
        Serial1.println("Error calibracion articulaciones");
        Serial1.print("Error cadera izq :");
        Serial1.println(err_cadera_izq);
        Serial1.print("Error pie izq :");
        Serial1.println(err_pie_izq);
        Serial1.print("Error cadera der :");
        Serial1.println(err_cadera_der);
        Serial1.print("Error pie der :");
        Serial1.println(err_pie_der);
        delay(1000);
        estado = 4; //Realizar otra calibración si pulsador
    }
    break;

case 4: //Estado otra calibracion
    num_vueltas = 0;

    lcd.setCursor(0, 0);
    lcd.print("Press for a new ");
    lcd.setCursor(0, 1);
    lcd.print("  calibration   ");

    if (digitalRead(pin_pulsador1) == 1)
              //pasamos a realizar otra calibración al pulsar
    {
        Serial1.println("Iniciamos otra calibracion 4 a 2");
        estado = 2;
        timer = millis();
        timerLed = millis();
        delay(20);
        counter = 0;
    }

    break;
```

```
    }
}

void calibra_servos()
{
    switch (num_vueltas) {
    case 50:
        data[0] = 3; //Indicador cadera izquierda
        data[1] = 90;
        data[2] = 0;
        WriteComm(6); //MsXC Comando mover servo especifico a posicion especifica

        pos_old_cadera_izq = 90;
        break;

    case 60:
        data[0] = 4; //Indicador pie izquierdo
        data[1] = 90;
        data[2] = 0;
        WriteComm(6); //MsXC Comando mover servo especifico a posicion especifica

        pos_old_pie_izq = 90;
        break;

    case 70:
        data[0] = 1; //Indicador cadera derecha
        data[1] = 90;
        data[2] = 0;
        WriteComm(6); //MsXC Comando mover servo especifico a posicion especifica
        pos_old_cadera_der = 90;
        break;

    case 80:
        data[0] = 2; //Indicador pie derecho
        data[1] = 90;
        data[2] = 0;
        WriteComm(6); //MsXC Comando mover servo especifico a posicion especifica
        pos_old_pie_der = 90;
        break;

    case 100:
    case 150:
    case 200:
    case 250:
    case 300:

        data[0] = 3; //Indicador cadera izquierda
        data[1] = round(pos_old_cadera_izq + err_cadera_izq);
        data[2] = 0;
        WriteComm(6); //MsXC Comando mover servo especifico a posicion especifica

        pos_old_cadera_izq = round(pos_old_cadera_izq + err_cadera_izq);
            //Redondeamos en lugar de cast para mejorar el error [servo.write(int)]
        break;

    case 110:
    case 160:
```

```
case 210:
case 260:
case 310:

    data[0] = 4; //Indicador pie izquierdo
    if (caraZowi) {
        data[1] = round(pos_old_pie_izq + err_pie_izq);
    }
    else {
        data[1] = round(pos_old_pie_izq - err_pie_izq);
    }
    data[2] = 0;
    WriteComm(6); //Comando mover servo especifico a posicion especifica
    if (caraZowi) {
        pos_old_pie_izq = round(pos_old_pie_izq + err_pie_izq);
    }
    else {
        pos_old_pie_izq = round(pos_old_pie_izq - err_pie_izq);
    }
    break;

case 120:
case 170:
case 220:
case 270:
case 320:
    data[0] = 1; //Indicador pie izquierdo
    data[1] = round(pos_old_cadera_der + err_cadera_der);
    data[2] = 0;
    WriteComm(6); //MsXCComando mover servo especifico a posicion especifica
    pos_old_cadera_der = round(pos_old_cadera_der + err_cadera_der);
    break;

case 130:
case 180:
case 230:
case 280:
case 330:

    data[0] = 2; //Indicador pie derecho
    if (caraZowi) {
        data[1] = round(pos_old_pie_der + err_pie_der);
    }
    else {
        data[1] = round(pos_old_pie_der - err_pie_der);
    }

    data[2] = 0;
    WriteComm(6); //MsXC Comando mover servo especifico a posicion especifica

    if (caraZowi) {
        pos_old_pie_der = round(pos_old_pie_der + err_pie_der);
    }

    else {
        pos_old_pie_der = round(pos_old_pie_der - err_pie_der);
    }
```

```
        break;

    default:
        break;
    }
}

void parpadeoLed(int freqLed)
{
    if (freqLed == 0) {
        digitalWrite(Led_azul, LOW);
    }
    else if ((millis() - timerLed) >= (500 / freqLed))
            // Led parpadea N vez por segundo, N = freqLed
    {
        timerLed = millis();
        digitalWrite(Led_azul, !digitalRead(Led_azul));
    }
}

void calculosIMU()
{

    counter++;
    timer_old = timer;
    timer = millis();
    if (timer > timer_old)
        G_Dt = (timer - timer_old) / 1000.0;
            // Real time of loop run. We use this on the DCM algorithm
            // (gyro integration time)
    else
        G_Dt = 0;

    // *** DCM algorithm
    // Data adquisition
    Read_Gyro(); // This read gyro data
    Read_Accel(); // Read I2C accelerometer

    if (counter > 5) // Read compass data at 10Hz... (5 loop runs)
    {
        counter = 0;
        Read_Compass(); // Read I2C magnetometer
        Compass_Heading(); // Calculate magnetic heading
    }

    // Calculations...
    Matrix_update();
    Normalize();
    Drift_correction();
    Euler_angles();
    // ***

    //printdata(); //Muestra datos leidos IMU
}

int init_imus()
```

```
{
    Serial1.println("Calibrando imus...");

    digitalWrite(STATUS_LED, LOW);
    delay(1500);

    Accel_Init();
    Compass_Init();
    Gyro_Init();

    delay(20);

    for (int i = 0; i < 32; i++) // We take some readings...
    {
        Read_Gyro();
        Read_Accel();
        for (int y = 0; y < 6; y++) // Cumulate values
        {
            AN_OFFSET[y] += AN[y];
            AN_OFFSETd[y] += ANd[y];
        }
        delay(20);
    }

    for (int y = 0; y < 6; y++) {
        AN_OFFSET[y] = AN_OFFSET[y] / 32;
        AN_OFFSETd[y] = AN_OFFSETd[y] / 32;
    }

    AN_OFFSET[5] -= GRAVITY * SENSOR_SIGN[5];
    AN_OFFSETd[5] -= GRAVITY * SENSOR_SIGNd[5];

    delay(4000);
    digitalWrite(STATUS_LED, HIGH);

    timer = millis();
    delay(20);
    counter = 0;
}

int WriteComm(int command)
{

    int ind = 0;

    switch (command) {

    case 0: //Iniciar comunicación con ZUM.
        for (int i = 0; i < 30; i++) {
            trama[i] = 0;
        }
        for (int i = 0; i < 6; i++) {
            trama[i] = C0[i];
        }
        trama[6] = '#';

        Serial.write(trama);
```

```
    Serial.flush();
    Serial1.println(trama);

    break;

case 8: //cargar programa test y guardar en base de datos.
    for (int i = 0; i < 30; i++) {
        trama[i] = 0;
    }
    for (int i = 0; i < 6; i++) {
        trama[i] = C8[i];
    }
    trama[6] = '#';
    Serial.write(trama);
    ;
    Serial1.println(trama);

    break;

case 1: //Leer offset

    delay(100);
    ind = 0;
    //Quedamos a espera de la lectura
    while (Wire.available()) {
        data[ind] = Wire.read();
        ind++;
    }
    dataINT[0] = data[0];
    dataINT[1] = data[1];
    dataINT[2] = data[2];
    dataINT[3] = data[3];
    Serial1.println("Posicion HOME leida de ZOWI:");
    Serial1.print("Home cadera izq: ");
    Serial1.println(dataINT[0]);
    Serial1.print("Home pie izq: ");
    Serial1.println(dataINT[1]);
    Serial1.print("Home cadera der: ");
    Serial1.println(dataINT[2]);
    Serial1.print("Home pie der: ");
    Serial1.println(dataINT[3]);
    break;

case 22: //Escribir offset WOFC Izquierda
    for (int i = 0; i < 29; i++) {
        trama[i] = 0;
    }
    ////////////////////////////////////
    /*  INFORMACION
      data[0]=pos_old_cadera_der; //COLUMNA 1 para BASE DATOS
      data[1]=pos_old_pie_der;  //COLUMNA 3 oara BASE DATOS
      data[2]=pos_old_cadera_izq;  //COLUMNA 0 oara BASE DATOS
      data[3]=pos_old_pie_izq; //COLUMNA 2 para base DATOS
        */ ////////////////////////////////////////////////

    for (int i = 0; i < 6; i++) {
        trama[i] = C2[i];
```

```
    }

    trama[6] = '2';

    numeroData = 2;
    for (int j = 7; j < 19;) {
        str = String(int(data[numeroData] / 100));
        str.toCharArray(temp, 2);
        trama[j] = temp[0];
        j++;
        str = String(int((data[numeroData] % 100) / 10));
        str.toCharArray(temp, 2);
        trama[j] = temp[0];
        j++;
        str = String(int(data[numeroData] % 10));
        str.toCharArray(temp, 2);
        trama[j] = temp[0];
        numeroData++;
        if (numeroData == 4) {
            numeroData = 0;
        };
        j++;
        //Serial1.println("aqui");
    }
    trama[19] = '#';

    Serial1.println(trama);
    //Mandamos los valores
    Serial.write(trama);
    break;

case 2: //Escribir offset WOFC
    for (int i = 0; i < 29; i++) {
        trama[i] = 0;
    }
    ////////////////////////////////////
    /*  INFORMACION
      data[0]=pos_old_cadera_der; //COLUMNA 1 para BASE DATOS
      data[1]=pos_old_pie_der;  //COLUMNA 3 oara BASE DATOS
      data[2]=pos_old_cadera_izq;  //COLUMNA 0 oara BASE DATOS
      data[3]=pos_old_pie_izq; //COLUMNA 2 para base DATOS
        */ ////////////////////////////////////////////////

    for (int i = 0; i < 6; i++) {
        trama[i] = C2[i];
    }

    trama[6] = '1';

    numeroData = 0;
    for (int z = 7; z < 19;) {
        str = String(int(data[numeroData] / 100));
        str.toCharArray(temp, 2);
        trama[z] = temp[0];
        z++;
        str = String(int((data[numeroData] % 100) / 10));
        str.toCharArray(temp, 2);
```

```
            trama[z] = temp[0];
            z++;
            str = String(int(data[numeroData] % 10));
            str.toCharArray(temp, 2);
            trama[z] = temp[0];
            numeroData++;
            z++;
            //Serial1.println("aqui");
        }
        trama[19] = '#';

        Serial1.println(trama);
        //Mandamos los valores
        Serial.write(trama);
        break;

    case 3: //Mover servos a 90°
        for (int i = 0; i < 30; i++) {
            trama[i] = 0;
        }
        for (int i = 0; i < 6; i++) {
            trama[i] = C3[i];
        }
        trama[6] = '#';

        Serial.write(trama);

        Serial1.println("Enviado M90C:");
        break;
    case 4: //Mover servos a HOME (90 calibrado)
        for (int i = 0; i < 30; i++) {
            trama[i] = 0;
        }
        for (int i = 0; i < 6; i++) {
            trama[i] = C4[i];
        }
        trama[6] = '#';

        Serial.write(trama);

        break;
    case 5: //Mover 4 servos a posicion especifica (para calibracion con IMU)
        Serial1.println("Enviado MSSC:");
        //Mandamos los valores
        //Wire.write(data);
        break;
    case 6: //Mover un servo a posicion especifica (para calibracion con IMU)MSxC

        for (int i = 0; i < 6; i++) {
            trama[i] = C6[i];
        }

        //Serial1.println(int(data[1]));

        str = String(int(data[0]));
        str.toCharArray(temp, 2);
        trama[6] = temp[0];
```

```
    str = String(int(data[1] / 100));
    str.toCharArray(temp, 2);
    trama[7] = temp[0];
    str = String(int((data[1] % 100) / 10));
    str.toCharArray(temp, 2);
    trama[8] = temp[0];
    str = String(int(data[1] % 10));
    str.toCharArray(temp, 2);
    trama[9] = temp[0];

    trama[10] = '#';
    Serial.write(trama);
    ////Serial1.write(trama);
    Serial1.println(trama);
    break;

case 7: //WERC
    //      for (int i =0; i<=40;i++){
    //      tramaError[i]=0;
    //      }

    for (int i = 0; i < 6; i++) {
        tramaError[i] = C7[i];
    }

    for (int i = 0; i < 4; i++) {
        if (dataError[i] < 0) {
            tramaError[(6 + i * 8)] = '-';
        }
        else {
            tramaError[(6 + i * 8)] = '+';
        }
        dataError[i] = abs(int(dataError[i] * 100));

        str = String(int(dataError[i] / 10000));
        str.toCharArray(temp, 2);
        tramaError[(7 + i * 8)] = temp[0];

        str = String((int(dataError[i] / 1000)) % 10);
        str.toCharArray(temp, 2);
        tramaError[(8 + i * 8)] = temp[0];

        str = String((int(dataError[i] / 100) % 10));
        str.toCharArray(temp, 2);
        tramaError[(9 + i * 8)] = temp[0];

        tramaError[(10 + i * 8)] = '.';

        str = String((int(dataError[i]) / 10) % 10);
        str.toCharArray(temp, 2);
        tramaError[(11 + i * 8)] = temp[0];

        str = String((int(dataError[i]) % 10));
        str.toCharArray(temp, 2);
        tramaError[(12 + i * 8)] = temp[0];
```

```
            tramaError[(13 + i * 8)] = '*';
        }

        tramaError[38] = resultado;
        tramaError[39] = '#';

        Serial1.println(tramaError);
        Serial.write(tramaError);
        //      for (int i =0; i<=40;i++){
        //      tramaError[i]=0;
        //      }
        break;

    case 9: //Apagar RB
        for (int i = 0; i < 30; i++) {
            trama[i] = 0;
        }
        for (int i = 0; i < 6; i++) {
            trama[i] = C9[i];
        }

        trama[6] = '#';
        Serial.write(trama);

        break;

    case 10:
        for (int i = 0; i < 30; i++) {
            trama[i] = 0;
        }
        for (int i = 0; i < 6; i++) {
            trama[i] = C10[i];
        }

        trama[6] = '#';
        Serial.write(trama);

    default:
        break;
    }

    data[0] = 0;
    data[1] = 0;
    data[2] = 0;
    data[3] = 0;
    data[4] = 0;

    return 1;
}

void sonidoNOK()
{
    tone(Pin_buzz, 100);
    delay(800);
    noTone(Pin_buzz);
}
```

```
void sonidoOK()
{
    tone(Pin_buzz, 400);
    delay(300);
    noTone(Pin_buzz);
    delay(50);
}

void sonidostart()
{
    tone(Pin_buzz, 560);
    delay(300);
    noTone(Pin_buzz);
    delay(10);
    tone(Pin_buzz, 545);
    delay(150);
    tone(Pin_buzz, 575);
    delay(400);
    noTone(Pin_buzz);
    delay(50);
}
```