UADY
UNIVERSIDAD AUTÓNOMA DE YUCATÁN

Simulador de cuidado maternal

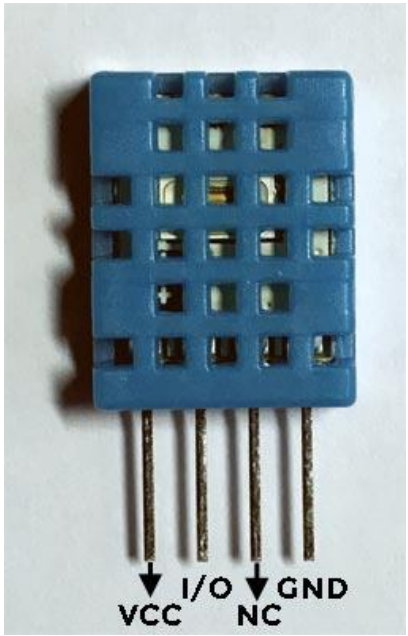
Saúl Gamboa León Alejandro Santoscoy Rivero Eduardo Catzín Uc

5 de Julio de 2019

Introducción

Sensores

IMU-GY 91: Este módulo integra el IMU MPU9250 y el barómetro BMP280, logrando un total de 10 grados de libertad (DoF) y contiene todo lo necesario para realizar rastreo de movimiento espacial de un Drone o UAV. Combina un giroscopio de 3 ejes, un acelerómetro de 3 ejes, un magnetómetro de 3 ejes y un altímetro en un mismo chip.



Una de las ventajas que nos ofrece el **DHT11**, además de medir la temperatura y la humedad, es que es digital. A diferencia de sensores como el LM35, este sensor utiliza un pin digital para enviarnos la información y por lo tanto, estaremos más protegidos frente al ruido.

Componentes

Buzzer. Permite generar una respuesta sonora cuando se somete al bebé a condiciones anómalas

Módulo RTC (Real Time Clock) o "Reloj de tiempo real" consiste en un circuito integrado alimentado por una batería el cual, en todo momento, registra la fecha, día de la semana y hora al igual que un reloj digital convencional.

Modulo Micro SD: nos permite insertar una memoria Micro SD que son las más comunes en el mercado, el modulo se puede alimentar con 3.3V o 5V usando los pines respectivos.

Metodología

```
#include <Wire.h>
#include "EmonLib.h"
#include <DHT.h>
#include <DS1302.h>
#include <SD.h>
#include <SPI.h>
File myFile;
DS1302 rtc(8,7,6);
Time t;
int buzzer=3;
int pinCS = 10;
int contador = 0;
DHT dht(4, DHT11);
int gyro_x, gyro_y, gyro_z;
long acc_x, acc_y, acc_z, acc_total_vector;
int temperature;
long gyro_x_cal, gyro_y_cal, gyro_z_cal;
long loop_timer;
int lcd_loop_counter;
float angle_pitch, angle_roll;
int angle_pitch_buffer, angle_roll_buffer;
boolean set_gyro_angles;
float angle_roll_acc, angle_pitch_acc;
float angle_pitch_output, angle_roll_output;
bool maltrato=false;
float old_gyro_x, old_gyro_y, old_gyro_z, old_acc_x, old_acc_y, old_acc_z;
float difgyrox,difgyroy,difgyroz,difaccx,difaccy,difaccz;
long buzzer_interval=1000;
unsigned long buzzerMillis=0;
int buzzer_state=262;
```

```
void setup() {
  Wire.begin();
  Serial.begin(9600);
  pinMode(buzzer, OUTPUT);
  pinMode(pinCS,OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(5, INPUT);
  if(SD.begin()){
    Serial.println("contador,fecha,hora,humedad,temperatura,acelX,acelY,acelZ,gyroX,gyroY,gyroZ,pitch,roll,maltrato");
  }
  else{
    Serial.println("La tarjeta SD no pudo iniciar correctamente");
}
```

```
dht.begin();
  setup_mpu_9250_registers();
  digitalWrite(2, HIGH);
  Serial.println("Calibrating gyro");
  for(int cal_int=0; cal_int<2000; cal_int ++){
    if(cal_int%125==0)Serial.print(".");
    read_mpu_9250_data();                          //Read the raw acc and gyro data from the MPU-6050
    gyro_x_cal += gyro_x;                          //Add the gyro x-axis offset to the gyro_x_cal variable
    gyro_y_cal += gyro_y;                          //Add the gyro y-axis offset to the gyro_y_cal variable
    gyro_z_cal += gyro_z;                          //Add the gyro z-axis offset to the gyro_z_cal variable
    delay(3);
  }
  gyro_x_cal /= 2000;                              //Divide the gyro_x_cal variable by 2000 to get the avarage offset
  gyro_y_cal /= 2000;                              //Divide the gyro_y_cal variable by 2000 to get the avarage offset
  gyro_z_cal /= 2000;
  Serial.println("");
  Serial.println("Pitch,Roll,Ax,Ay,Az,Gx,Gy,Gz");
  digitalWrite(2,LOW);
  loop_timer=micros();
}
```

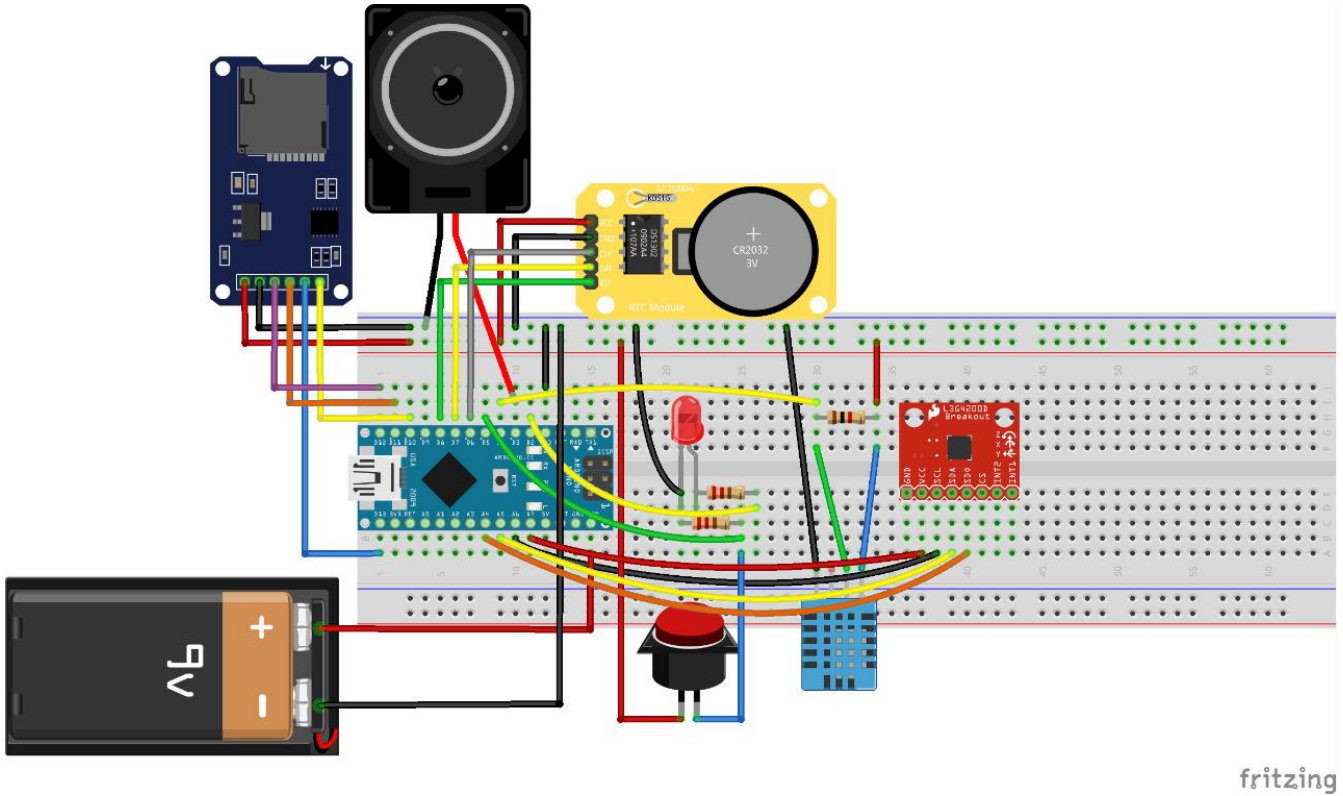
```
void loop() {
  contador=contador+1;
  t=rtc.getTime();
  myFile=SD.open("bebe.txt",  FILE_WRITE);
  if(contador==1) myFile.println("contador,fecha,hora,humedad,temperatura,acelX,acelY,acelZ,gyroX,gyroY,gyroZ,pitch,roll,maltrato");
  read_mpu_9250_data();
  gyro_x -= gyro_x_cal;                                //Subtract the offset calibration value from the raw gyro_x value
  gyro_y -= gyro_y_cal;                                //Subtract the offset calibration value from the raw gyro_y value
  gyro_z -= gyro_z_cal;
  angle_pitch += gyro_x * 0.0000611;                   //Calculate the traveled pitch angle and add this to the angle_pitch variable
  angle_roll += gyro_y * 0.0000611;
  angle_pitch += angle_roll * sin(gyro_z * 0.000001066);          //If the IMU has yawed transfer the roll angle to the pitch angle
angle_roll -= angle_pitch * sin(gyro_z * 0.000001066);
```

```
//Accelerometer angle calculations
 acc_total_vector = sqrt((acc_x*acc_x)+(acc_y*acc_y)+(acc_z*acc_z)); //Calculate the total accelerometer vector
 //57.296 = 1 / (3.142 / 180) The Arduino asin function is in radians
 angle_pitch_acc = asin((float)acc_y/acc_total_vector)* 57.296;       //Calculate the pitch angle
 angle_roll_acc = asin((float)acc_x/acc_total_vector)* -57.296;       //Calculate the roll angle
 angle_pitch_acc -= 0.0;                                              //Accelerometer calibration value for pitch
 angle_roll_acc -= 0.0;                                               //Accelerometer calibration value for roll
 if(set_gyro_angles){                                                 //If the IMU is already started
  angle_pitch = angle_pitch * 0.9996 + angle_pitch_acc * 0.0004;     //Correct the drift of the gyro pitch angle with the accelerometer pitch angle
  angle_roll = angle_roll * 0.9996 + angle_roll_acc * 0.0004;        //Correct the drift of the gyro roll angle with the accelerometer roll angle
 }
 else{                                                                //At first start
  angle_pitch = angle_pitch_acc;                                      //Set the gyro pitch angle equal to the accelerometer pitch angle
  angle_roll = angle_roll_acc;                                        //Set the gyro roll angle equal to the accelerometer roll angle
  set_gyro_angles = true;                                             //Setthe IMU started flag
}
```

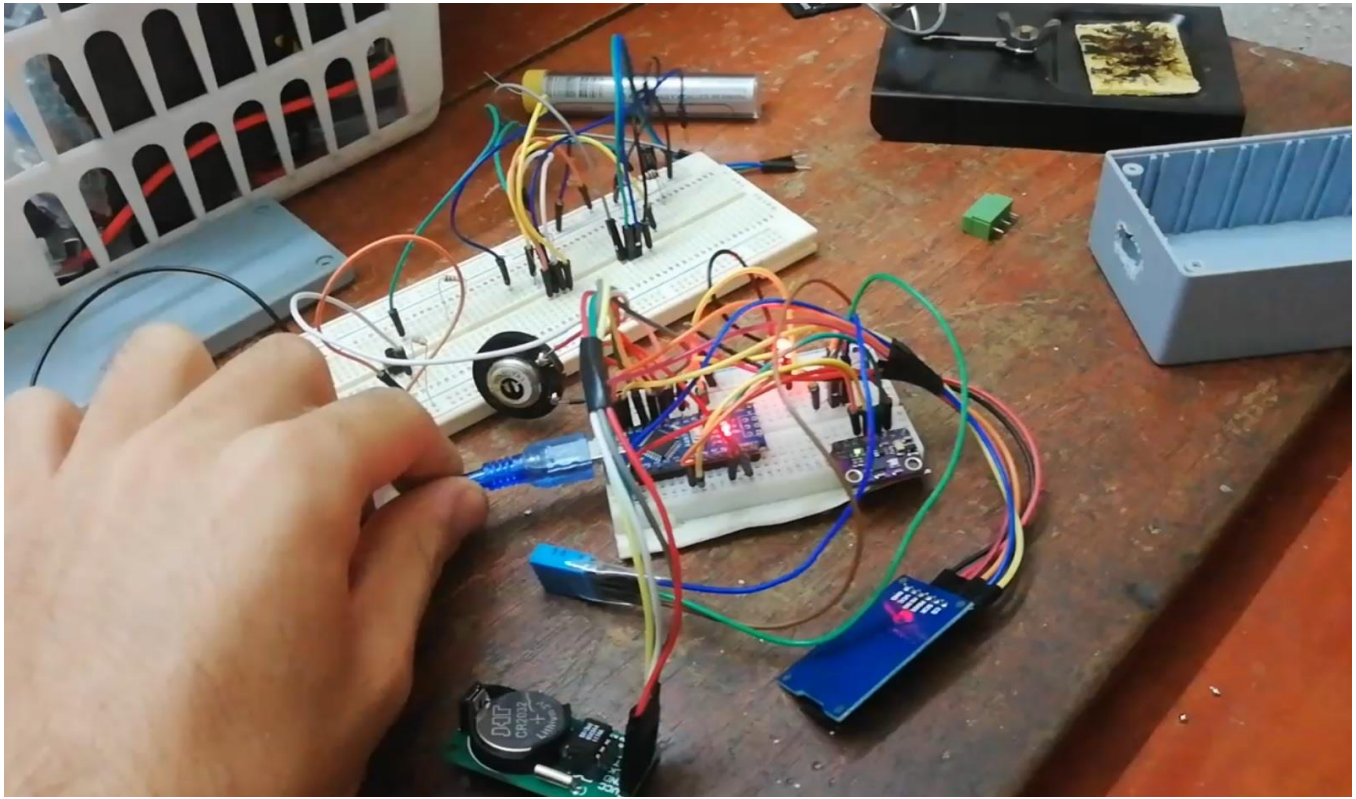
```
if(difgyrox>10000||difgyroy>10000||difgyroz>10000){
  maltrato=true;
 }
 if(difaccx>10000||difaccy>10000||difaccz>10000){
  maltrato==true;
 }
 if(t.hour==3&&t.min==0&&t.sec==0){
  maltrato=true;
 }
 if(t.hour==6&&t.min==0&&t.sec==0){
  maltrato=true;
 }
 if(t.hour==9&&t.min==0&&t.sec==0){
  maltrato=true;
 }
 if(t.hour==15&&t.min==0&&t.sec==0){
  maltrato=true;
 }
 if(t.hour==18&&t.min==0&&t.sec==0){
  maltrato=true;
 }
 if(t.hour==21&&t.min==0&&t.sec==0){
  maltrato=true;
 }
 if(dht.readTemperature()>35){
  maltrato=true;
 }
 if(dht.readTemperature()<18){
  maltrato=true;
 }
 if(digitalRead(5)){
  maltrato=false;
  noTone(buzzer);
  digitalWrite(2, LOW);
 }
 if(maltrato==true){
  if(millis()-buzzerMillis>=buzzer_interval){
   buzzerMillis=millis();
   digitalWrite(2, !digitalRead(2));
   tone(buzzer, buzzer_state);
   delay(1000);
  }
}
```

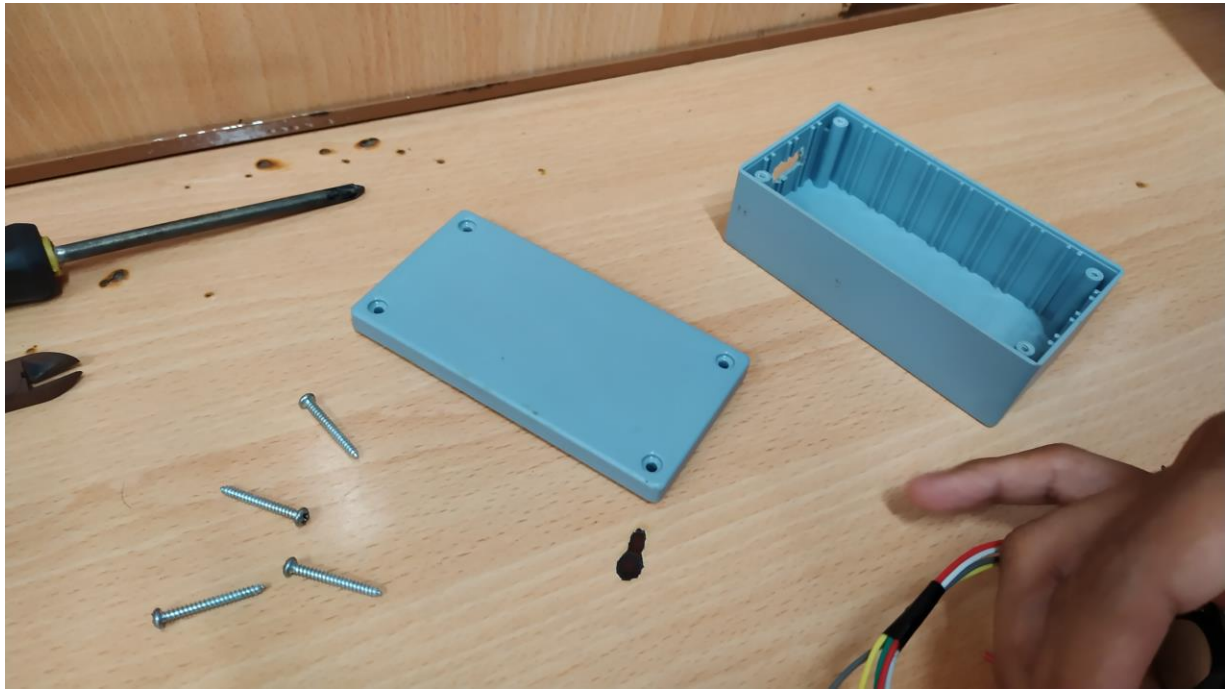
PARA DESCARGAR LAS LIBRERIAS Y EL CODIGO COMPLETO ENTRAR A

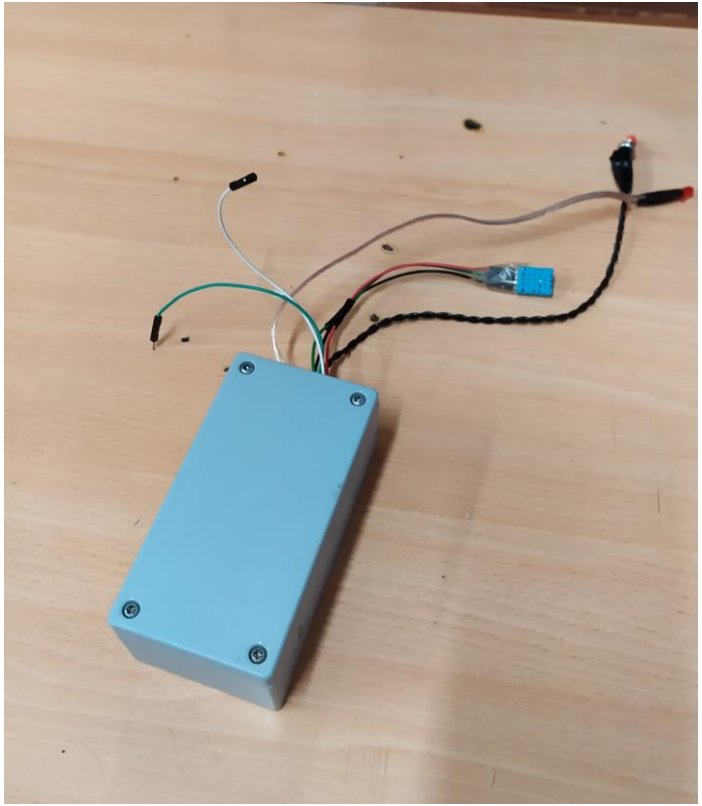
https://github.com/Santocoyo/Simulador-de-cuidado-maternal

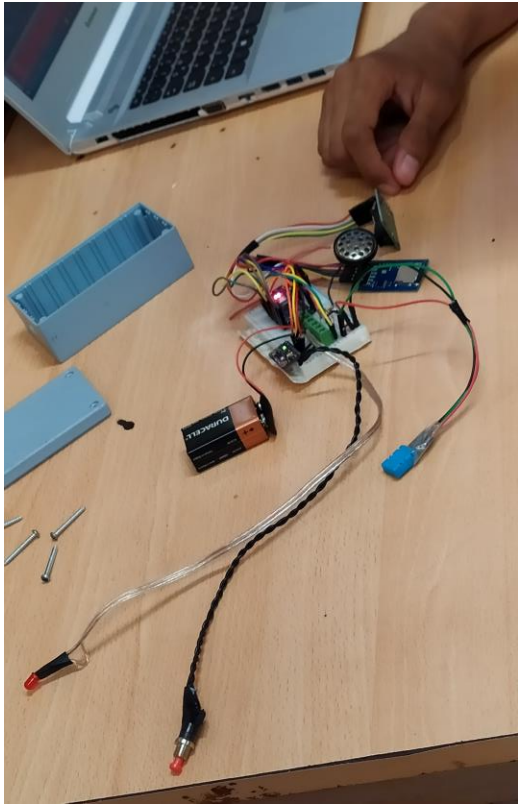

CR2032
3V
RTC Module
L3G4200D
Breakout
9V
fritzing

Integración del prototipo

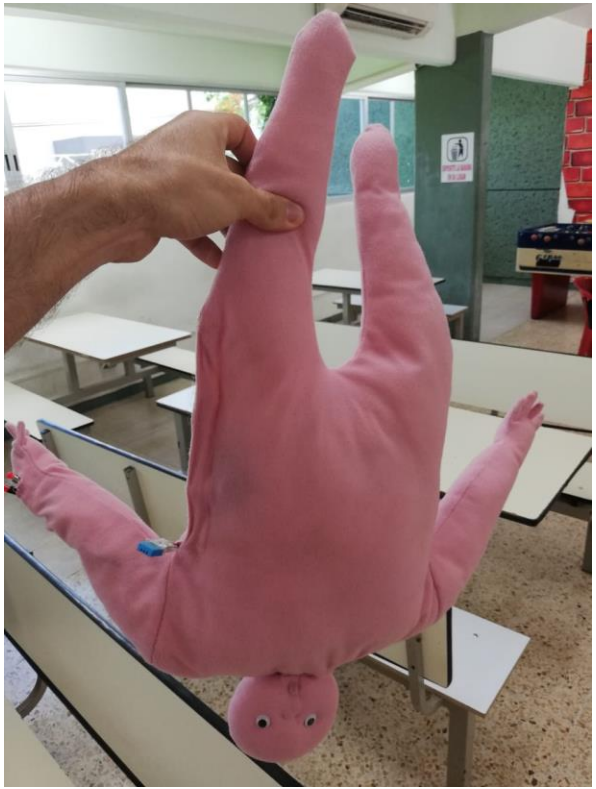




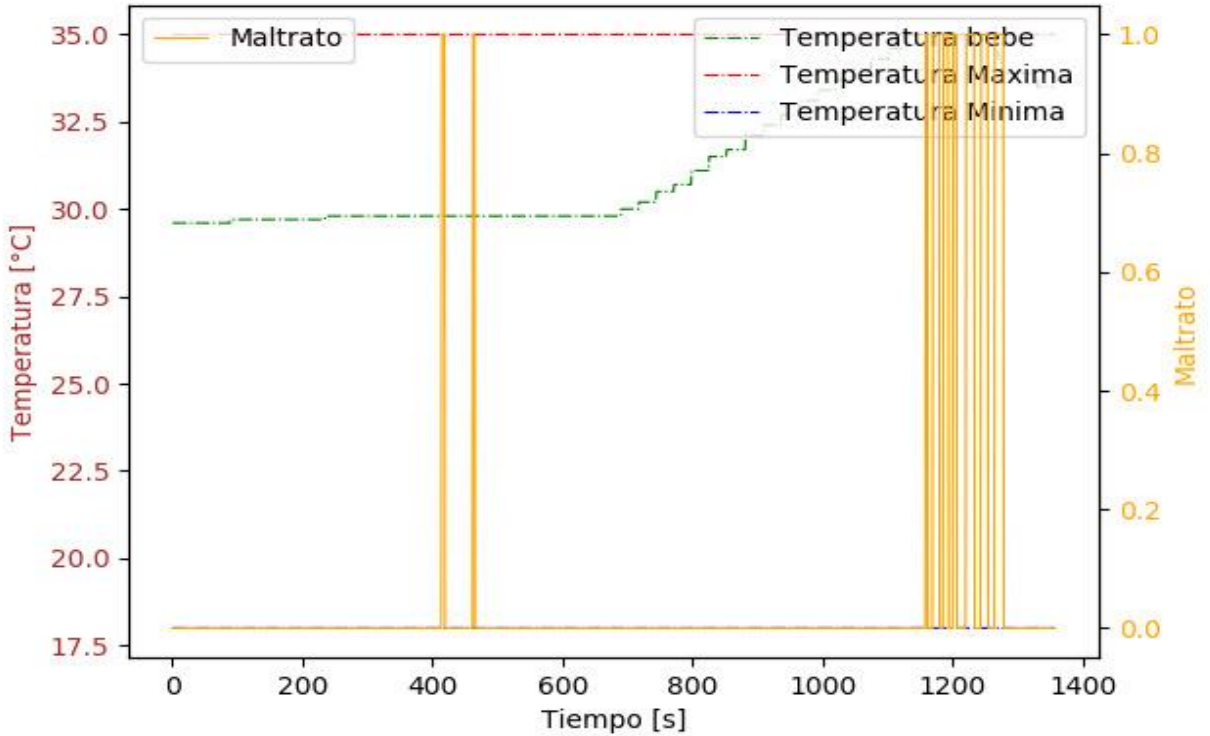


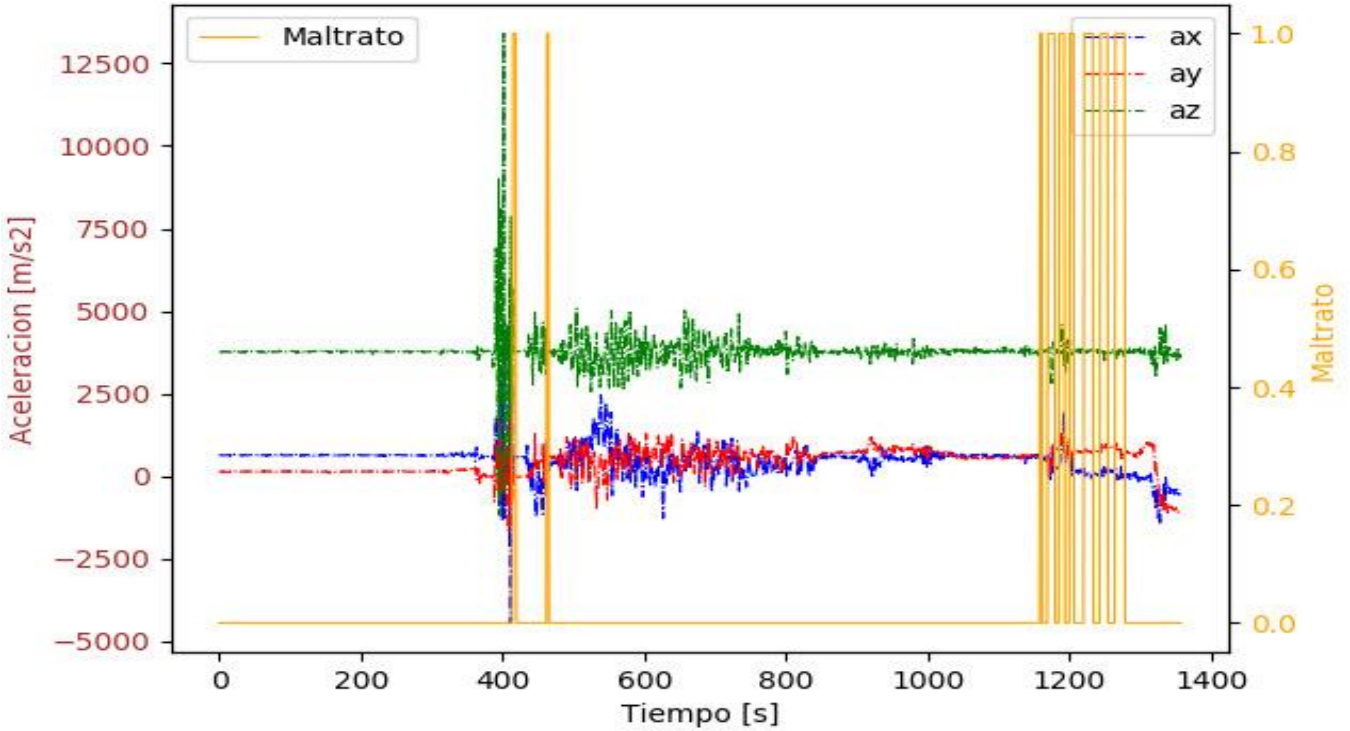


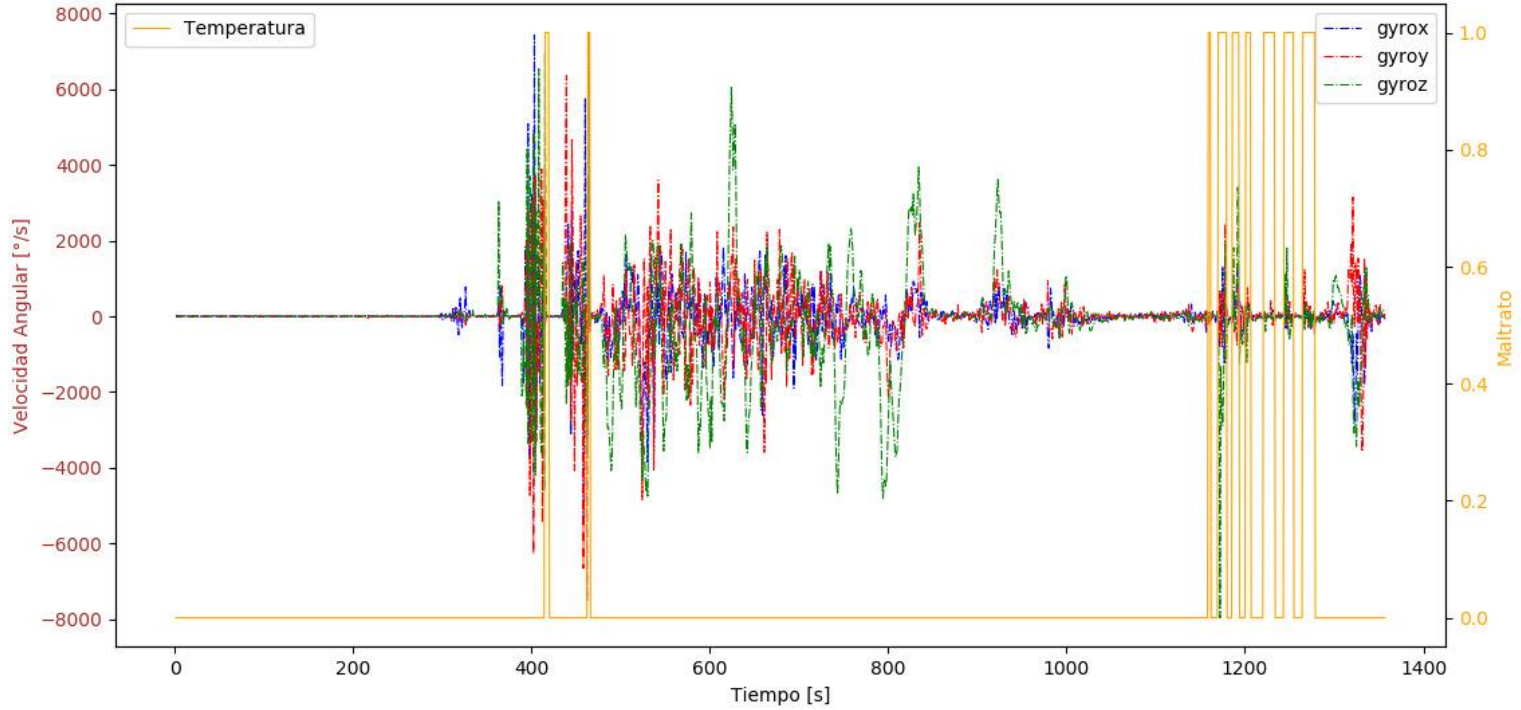
Pruebas

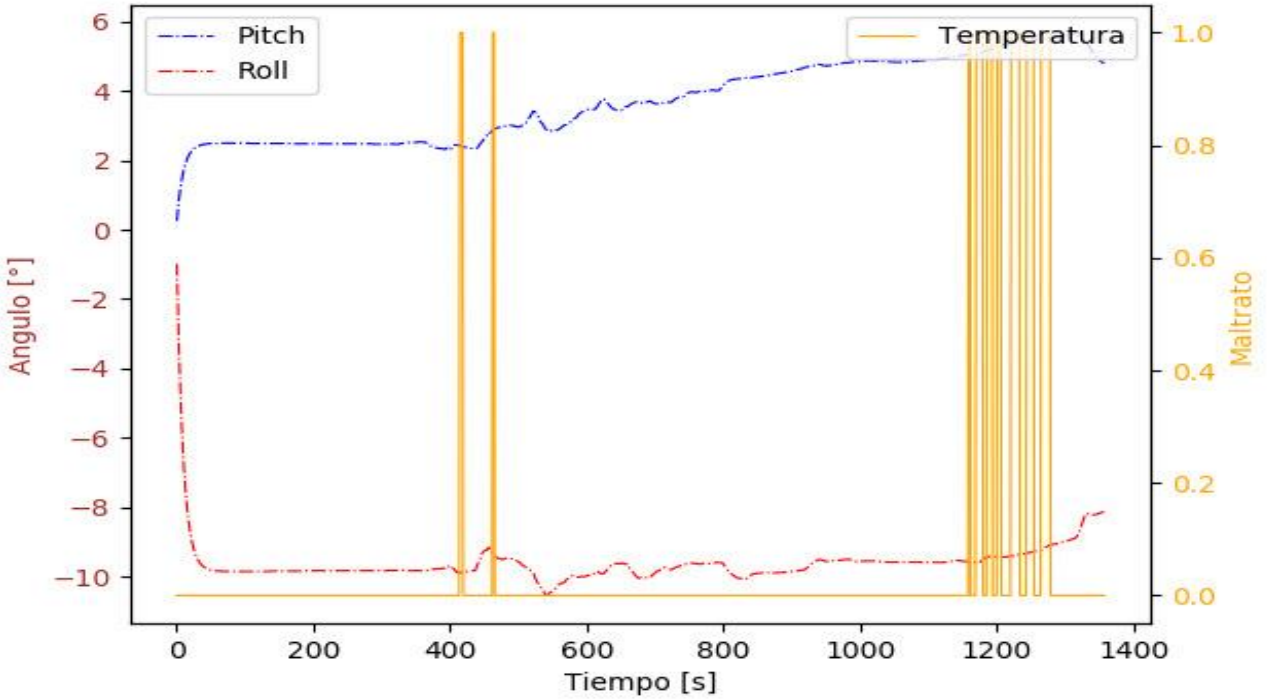


Resultados


Maltrato
Temperatura bebe
Temperatura Maxima
Temperatura Minima
Temperatura [°C]
Maltrato
Tiempo [s]
35.0
32.5
30.0
27.5
25.0
22.5
20.0
17.5
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
1200
1400

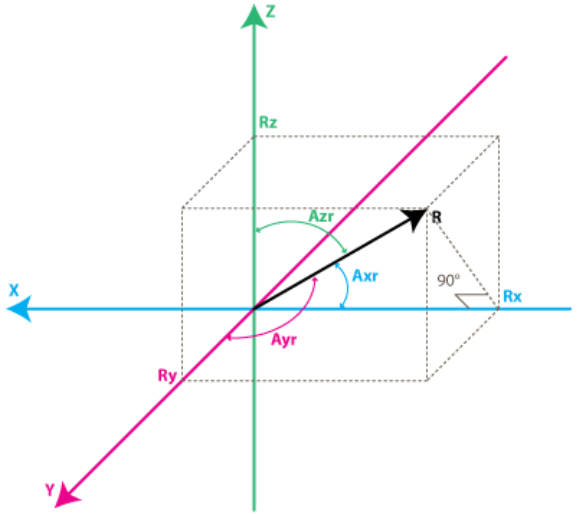

Maltrato
ax
ay
az
Aceleracion [m/s2]
Maltrato
Tiempo [s]
12500
10000
7500
5000
2500
0
−2500
−5000
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
1200
1400


Temperatura
gyrox
gyroy
gyroz
Velocidad Angular [°/s]
Maltrato
Tiempo [s]
8000
6000
4000
2000
0
−2000
−4000
−6000
−8000
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
1200
1400


Pitch
Roll
Temperatura
Angulo [°]
Maltrato
Tiempo [s]
6
4
2
0
−2
−4
−6
−8
−10
1.0
0.8
0.6
0.4
0.2
0.0
0
200
400
600
800
1000
1200
1400

Conclusiones

Durante el ensamblaje del sistema se debe verificar la correcta conexión de los componentes y que la implementación del código se adecue a los parámetros que se desea medir (e.g. la implementación de un filtro para mejorar la precisión del sistema de giroscopio y acelerómetro)



Lecciones aprendidas

1- El giroscopio requiere una calibración adecuada.

2- La precisión de los ángulos debe mejorarse mediante otros filtros implementados en el código(filtros Kalman o Cuaterniones)

3- Comprobar el almacenamiento de datos antes de realizar las mediciones

Referencias

Naylamp Mechatronics. (s.f.). Módulo GY-91 MPU9250 + BMP280. Recuperado 4 julio, 2019, de https://naylampmechatronics.com/sensores-posicion-inerciales-gps/356-modulo-gy-91-mpu9250-bmp280-acelerometro-giroscopio-magnetometro-altimetro-i2c.html

Naylamp Mechatronics. (s.f.-b). Tutorial Arduino y memoria SD y micro SD.. Recuperado 4 julio, 2019, de https://naylampmechatronics.com/blog/38_Tutorial-Arduino-y-memoria-SD-y-micro-SD-.html

Cómo utilizar el sensor DHT11 para medir la temperatura y humedad con Arduino. (2018, 24 enero). Recuperado 4 julio, 2019, de https://programarfacil.com/blog/arduino-blog/sensor-dht11-temperatura-humedad-arduino/

Brokking, J. M. (s.f.). Create a 6dof IMU with a gyro and accelerometer for (Arduino) multicopters.. Recuperado 5 julio, 2019, de http://www.brokking.net/imu.html