
Diseño y construcción de Flight Data Recorder por medio del microprocesador Arduino y un lenguaje C++

Sergio Axel Medrano Cueto

IAE

1884510

ABSTRACT

El presente documento presenta el desarrollo, elaboración y análisis de resultados obtenidos del proyecto de la construcción de una Caja Negra de una aeronave. Esto orientado desde el enfoque de uso de Arduino y su lenguaje de programación basado en C++.

Palabras Clave: caja negra, aeronaves, registrador de datos de vuelo

INTRODUCCIÓN

La eficiencia de cualquier aeronave moderna, se beneficia y a su vez depende en gran medida, en la fiabilidad continua de la electricidad. Sistemas y subsistemas [1]. Instalado incorrectamente o sin cuidado o el cableado mantenido puede ser una fuente tanto de inmediato y el peligro potencial. La continua ejecución adecuada de los sistemas eléctricos dependen del conocimiento y la técnica del mecánico que instala, inspecciona y mantiene el sistema eléctrico de alambres y cables.

Un elemento fundamental en la parte electrónica de un aeronave es la Caja Negra, el cual es un elemento fundamental, puesto que ese vuelo es un dispositivo de grabación electrónico colocado en una aeronave con el fin de facilitar la investigación de accidentes e incidentes de aviación.



Figura 2. Caja Negra de un avión

Los datos registrados por la Caja Negra se utilizan para la investigación de accidentes e incidentes. Debido a su importancia en la investigación de accidentes, estos dispositivos regulados por la OACI están diseñados y contruidos cuidadosamente para soportar la fuerza de un impacto de alta velocidad y el calor de un fuego intenso[3]. Contrariamente al término popular "caja negra", el exterior de la Caja Negra está cubierto con pintura naranja brillante resistente al calor para una alta visibilidad en los restos, y la unidad generalmente se monta en la sección de cola del avión, donde es más probable que sobreviva a una accidente severo. Después de un accidente, la recuperación de la la Caja Negra suele ser una alta prioridad para el organismo investigador, ya que el análisis de los parámetros registrados a menudo puede detectar e identificar causas o factores contribuyentes.

Las Cajas Negras de hoy en día reciben entradas a través de marcos de datos específicos de las Unidades de Adquisición de Datos de Vuelo (FDAU). Registran parámetros de vuelo significativos, incluidas las posiciones de control y actuador, información del motor y hora del día[3].

En general, cada parámetro se registra varias veces por segundo, aunque algunas unidades almacenan "ráfagas" de datos a una frecuencia mucho mayor si los datos comienzan a cambiar rápidamente. "La mayoría de los FDR registran aproximadamente de 17 a 25 horas de datos en un ciclo continuo. Las regulaciones exigen que se realice una verificación de verificación de la Caja Negra anualmente para verificar que todos los parámetros obligatorios estén registrados"[4]

Las Cajas Negras de hoy en día, suelen tener doble envoltura en acero inoxidable fuerte resistente a la corrosión o titanio, con aislamiento de alta temperatura en el interior. Los FDR modernos están acompañados por una baliza localizadora submarina

que emite un "ping" ultrasónico para ayudar en la detección cuando se sumerge. Estas balizas funcionan hasta por 30 días y pueden operar mientras están sumergidas a una profundidad de hasta 6,000 metros (20,000 pies).[5]

ELABORACIÓN

La construcción de la Caja Negra, fue posterior al ver el íntegro y eficiente funcionamiento de nuestro código y el circuito hecho con nuestros sensores y demás componentes. La caja negra se hizo a partir de un gabinete adquirido previamente para un proyecto precedente. A este gabinete se le hicieron los orificios pertinentes para adaptarlo a nuestro proyecto por medio de una cortada láser.



Figura 3. Corte láser en el gabinete

Para obtener un circuito más eficiente y optimizar los espacios, adaptamos el sistema a una Placa PSV, armando el mismo circuito de la protoboard, pero añadiendo un proceso de soldado en el circuito como se ve en la Figura.

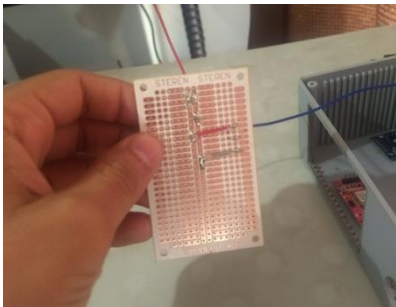


Figura 4. Placa psv

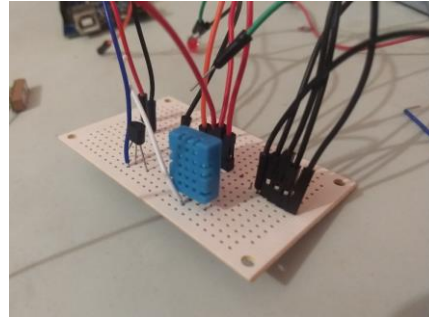


Figura 5. Circuito soldado a la placa psv

Se midieron los espacios de cada componente y se le hicieron los orificios correspondientes para tener los sensores y demás componentes lo más fijos posibles a la caja negra.



Figura 6. Taladrado de orificios



Figura 7. Pre Ensamble de todos los componentes

Posteriormente se ensamblan todos los componentes electrónicos a utilizar, para un ensamble eficiente de estos, se ajustó todos los componentes atornillarlos

con sus orificios y los orificios hechos previamente en la Caja Negra.



Figura 8. Ensamblado de todos los componentes

Para mejorar el factor de amortiguamiento de la caja, y así lograr que aguante el impacto de la inevitable caída que se hará el día de la prueba. Se le pego Foam Cardboard en las caras de la caja, también para mejorar su resistencia al impacto y la estética del producto a presentar se le agregan capas de papel fomi.

```
PIA_SISELA Arduino 1.8.8
Archivo Editar Programa Herramientas Ayuda

PIA_SISELA $

// PIA SISELA 2019 ARMANDO CANTU Y SERGIO MEDRANO

// Librerias
#include <TinyGPS++.h>
#include "DHT.h"
#include <Wire.h>
#include <UTFT.h>
#include <SD.h> //Load SD library
#include <SPI.h> //Load the SPI communication library
#include "MPU9250.h"
#include <UTFT.h>
extern uint8_t BigFont[];
extern uint8_t SmallFont[];
extern uint8_t SevenSegNumFont[];
UTFT myGLCD(HX8357C, 38, 39, 40, 41);

#define A 0.962
#define dt 0.020

#define DHTPIN 4 //PIN DIGITAL AL QUE SE CONECTA EL SENSOR DHT 1
#define DHTTYPE DHT11
#define PIN_ECHO 7
#define PIN_TRIGGER 8

//Definiciones para el MPU9250
#define MPU9250_ADDRESS 0x68
#define MAG_ADDRESS 0x0C
#define ACC_FULL_SCALE_2_G 0x00
#define GYRO_FULL_SCALE_1000_DPS 0x10
//Sensitivity scale factor 2G=16384=LSC/G
double SensitivityScaleFactor = 16384;

int i=1;
int ledV=12;
int ledR=11;
const int CS_mega = 53; //CS for Mega2560
File mySensorData; // Variable for working with our file object

unsigned int tiempo, distancia;
```

```
TinyGPSPlus gps;
DHT dht(DHTPIN, DHTTYPE);

float rollangle, pitchangle;
float roll, pitch, yaw;

//MPU 9250 LEER REGISTROS
void I2Cread(uint8_t Address, uint8_t Register, uint8_t Nbytes, uint8_t* Data)
{
    // Set register address
    Wire.beginTransmission(Address);
    Wire.write(Register);
    Wire.endTransmission();

    // Read Nbytes
    Wire.requestFrom(Address, Nbytes);
    uint8_t index=0;
    while (Wire.available())
        Data[index++]=Wire.read();
}

// Write a byte (Data) in device (Address) at register (Register)
void I2CwriteByte(uint8_t Address, uint8_t Register, uint8_t Data)
{
    // Set register address
    Wire.beginTransmission(Address);
    Wire.write(Register);
    Wire.write(Data);
    Wire.endTransmission();
}
```

1. Para empezar el código primeramente se tiene que declarar las librerías correspondientes; en este caso se declaran las librerías de: GPS, Sensor DHT11, SD, MPU9250, Comunicación SPI, Comunicación con los pins Análogos (Wire.h), y LCD TFT.

Se definen los pins de los sensores que se comunican específicamente a ciertos pins del Arduino Mega como el DHT11, Sensor Ultrasónico (Echo y Trigger), después de esto se definen los registros del MPU 9250, por ultimo declararemos dos LEDS en los pins (12=verde, 11=roja), declararemos que el pin 53 será nuestro CS para la comunicación con el módulo micro SD, declararemos tiempo y distancia para nuestro sensor ultrasónico y declararemos la variable de donde estará trabajando nuestro archivo SD.

2. Declararemos variables para la obtencion del pitch, roll y yaw, empezaremos a abrir los registros específicos del MPU 9250

```
PIA_SISELA Arduino 1.8.8
Archivo Editar Programa Herramientas Ayuda

PIA_SISELA $

void setup() {
  Wire.begin();
  Serial.begin(9600);
  Serial2.begin(115200);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  //SETUP SENSORES
  dht.begin();
  pinMode(A1, INPUT); //PIN ANALOGICO AL QUE SE CONECTA EL SENSOR LM35

  pinMode(PIN_ECHO, INPUT);
  pinMode(PIN_TRIGGER, OUTPUT);

  pinMode(10, OUTPUT); //Reserve pin 10 as an output, dont use it for other parts of circuit
  SD.begin(53); // Initialize SD card with chipset connected to pin 4

  myGLCD.InitLCD();
  myGLCD.clrScr();
  //CORRECT ROTATION SCREEN
  cbi(myGLCD.P_CS, myGLCD.B_CS); // CS Active
  myGLCD.LCD_Write_COM(0x36);
  myGLCD.LCD_Write_DATA(0x48 ^ 0xC0);
  sbi(myGLCD.P_CS, myGLCD.B_CS); // CS Idle

  //MPU 9250 SET UP
  // Set accelerometers low pass filter at 5Hz
  I2CwriteByte(MPU9250_ADDRESS, 29, 0x06);
  // Set gyroscope low pass filter at 5Hz
  I2CwriteByte(MPU9250_ADDRESS, 26, 0x06);

  // Configure accelerometers range
  I2CwriteByte(MPU9250_ADDRESS, 28, ACC_FULL_SCALE_2_G);
  // Configure gyroscope range
  I2CwriteByte(MPU9250_ADDRESS, 27, GYRO_FULL_SCALE_1000_DPS);

  //Set by pass mode for the magnetometer
  I2CwriteByte(MPU9250_ADDRESS, 30, 0x00);
}
```

3

3. Dentro del set up declaramos la velocidad en la que queremos que las variables impresas en el monitor serial se impriman, declaramos que tipo de pins son los LEDS en este caso un output, comenzaremos a el DHT11, declaramos el pin A1 como input para el sensor LM35 significa que recibiremos la señal de ese sensor, definiremos el pin Echo como input y el Trigger como output, inicialisaremos la SD en el pin 53, inicialisaremos la pantalla LCD y ejecutaremos un comando para que nos de la correcta rotacion de la pantalla.

```
PIA_SISELA Arduino 1.8.8
Archivo Editar Programa Herramientas Ayuda

PIA_SISELA $

void loop() {

mySensorData= SD.open("DataLog.txt", FILE_WRITE); //Open XYZdata.txt on the SD card as a file to write to

if (mySensorData == true){ //Only do these things if data file opened succesfully
  digitalWrite (12,HIGH);

  //GPS//
  while(Serial2.available() > 0)
    if(gps.encode(Serial2.read()));

  Serial.print(gps.time.hour());
  Serial.print(":");
  Serial.print(gps.time.minute());
  Serial.print(":");
  Serial.print(gps.time.second());
  Serial.print(",");
  Serial.print(gps.location.lat(), 6); //LATITUD
  Serial.print(",");
  Serial.print(gps.location.lng(), 6); //LONGITUD
  Serial.print(",");
  Serial.print(gps.altitude.meters()); //ALTITUD
  Serial.print(",");

  //DHT11 y LM35//

  //LM35
  int sensor = analogRead(A1);
  float mv=(sensor/1024.0)*5000;
  float cel=mv/10;

  //DHT11
  float t= dht.readTemperature();
  float h= dht.readHumidity();

  //IMPRESION DE DHT11 y LM35

  Serial.print(cel);
  Serial.print(",");
  Serial.print(h);
  Serial.print(",");
  Serial.print(t);
  Serial.print(",");

  //SENSOR ULTRASONICO//

  digitalWrite(PIN_TRIGGER, 0);
  delayMicroseconds(2);
  digitalWrite(PIN_TRIGGER, 1);
  delayMicroseconds(10);
  digitalWrite(PIN_TRIGGER, 0);
  tiempo=pulseIn(PIN_ECHO,1);
  distancia=tiempo/58;

  //IMPRIMIR DATOS DE DISTANCIA SENSOR ULTRASONICO
  Serial.print(distancia);
  Serial.print(",");
}
```

4. En el void loop primeramente nos aseguraremos de que la sd este funcionando bien, con un if diremos que si se abrio correctamente el archivo DataLog.txt haga entonces todo la funcion debajo del if, esto incluye encender el LED verde indicador de que si esta funcionando bien la SD, despues de esto iniciaremos comunicaci3n con el modulo GPS pidiendo los datos codificados como el tiempo, latitud, longitud y altitud y imprimiendolos al monitor serial.

5. Esta parte la lectura de los sensores LM35 y DHT11, primeramente, para que el sensor LM35 nos arroje un dato l3gico tenemos que hacer una operaci3n que nos convierte la se1al anal3gica a grados Celsius, el sensor DHT11 ya que es digital nos arroja autom1ticamente la temperatura y humedad en grados Celsius y porcentaje para la humedad. Por ulti3mo, imprimiremos al monitor serial estas variables

6. En esta parte leeremos los datos del sensor ultras3nicos, en esta parte los datos se tienen que convertir a metros y despu3s imprimimos en el monitor serial.

```

//MPU 9250//

//read accelerometer
uint8_t Buf[6];
I2Cread(MPU9250_ADDRESS, 0x3b, 6, Buf);

// Accelerometer
int16_t ax = -(Buf[0]<<8 | Buf[1]);
int16_t ay = -(Buf[2]<<8 | Buf[3]);
int16_t az = Buf[4]<<8 | Buf[5];
//change to SensitivityScaleFactor:
float AX = ax/SensitivityScaleFactor;
float AY = ay/SensitivityScaleFactor;
float AZ = az/SensitivityScaleFactor;
//print Accelerometer
Serial.print(AX);
Serial.print (" ");
Serial.print(AY);
Serial.print (" ");
Serial.print(AZ);
Serial.print (" ");

//Gyroscope
int16_t gx = -(Buf[0]<<8 | Buf[1]);
int16_t gy = -(Buf[2]<<8 | Buf[3]);
int16_t gz = Buf[4]<<8 | Buf[5];
//Print Gyroscope
Serial.print(gx/1000);
Serial.print(",");
Serial.print(gy/1000);
Serial.print(",");
Serial.print(gz/1000);
Serial.print(",");

```

7. En esta parte leeremos el MPU 9250 y imprimimos al monitor serial.

7

```

//Magnetometer
uint8_t ST1;
do
{
    I2Cread(MAG_ADDRESS, 0x02, 1, &ST1);
}
while(! (ST1 < 0x01));
//read magnetometer data
uint8_t Mag[7];
I2Cread(MAG_ADDRESS, 0x02, 7, Mag);

    int16_t mx = -(Mag[3]<<8 | Mag[2]);
    int16_t my = -(Mag[1]<<8 | Mag[0]);
    int16_t mz = -(Mag[5]<<8 | Mag[4]);

//impresion magnetometro

Serial.print(mx+200);
Serial.print(",");
Serial.print(my-70);
Serial.print(",");
Serial.print(mz-700);
Serial.print(",");

// PITCH ROLL YAW

Serial.print(" ROLL=");
Serial.print(roll);
Serial.print(" angle");
Serial.print(" PITCH=");
Serial.print(pitch);
Serial.print(" angle");
Serial.print(" YAW=");
Serial.print(yaw);
Serial.print("\n");

// IMPRESION SD CARD, LCD

//GPS

mySensorData.print(gps.time.hour());
mySensorData.print(":");
mySensorData.print(gps.time.minute());
mySensorData.print(":");
mySensorData.print(gps.time.second());
mySensorData.print(",");
mySensorData.print(gps.location.lat(), 6); //LATITUD
mySensorData.print(",");
mySensorData.print(gps.location.lng(), 6); //LONGITUD
mySensorData.print(",");
mySensorData.print(gps.altitude.meters()); //ALTITUD
mySensorData.print(",");

//LM35 DHT11
mySensorData.print(ceil);
mySensorData.print(",");
mySensorData.print(h);
mySensorData.print(",");
mySensorData.print(t);
mySensorData.print(",");

//ULTRASONICO
mySensorData.print(distancia);
mySensorData.print(",");

//MPU 9250
mySensorData.print(AX);
mySensorData.print(",");
mySensorData.print(AY);
mySensorData.print(",");
mySensorData.print(AZ);
mySensorData.print(",");
mySensorData.print(gx/1000);
mySensorData.print(",");
mySensorData.print(gy/1000);
mySensorData.print("\n");

```

7

8

8. En esta parte ya imprimimos y guardamos en la SD todos los datos ya sacados en las partes anteriores.


```

//IMPRESION LCD

myGLCD.setFont(BigFont);
myGLCD.setColor(255,255,255);
myGLCD.print("PIA SISELA",CENTER,0);

myGLCD.setColor(255,12,12);
myGLCD.print("DHT11          LM35",45,35);

myGLCD.setColor(255,12,12);
myGLCD.setFont(SmallFont);
myGLCD.print("TEMPERATURA:",45,60);
myGLCD.print("TEMPERATURA:",270,60);
myGLCD.printNumI(t,140,60);
myGLCD.print("'C",160,60);
myGLCD.printNumI(ce1,365,60);
myGLCD.print("'C",385,60);
myGLCD.print("HUMEDAD:",45,80);
myGLCD.printNumI(h,115,80);
myGLCD.print("%",135,80);

myGLCD.setFont(BigFont);
myGLCD.setColor(15,15,255);
myGLCD.print("Sensor Ultrasonico",45,105);
myGLCD.setFont(SmallFont);
myGLCD.setColor(15,15,255);
myGLCD.print("DISTANCIA:",45,130);
myGLCD.print("          ",127,130);
myGLCD.printNumI(distancia,127,130);
myGLCD.print("cm",152,130);

myGLCD.setFont(BigFont);
myGLCD.setColor(0,255,0);
myGLCD.print("-----");
myGLCD.setFont(SmallFont);
myGLCD.setColor(0,255,0);
myGLCD.print("AX:",45,185);
myGLCD.print("GX:",190,185);
myGLCD.print("MX:",333,185);
myGLCD.printNumF(AX,3,80,185);
myGLCD.printNumF(gx/1000,0,220,185);
myGLCD.printNumF(mx+200,0,365,185);
myGLCD.print("AY:",45,200);
myGLCD.print("GY:",190,200);
myGLCD.print("MY:",333,200);
myGLCD.printNumF(AY,3,80,200);
myGLCD.printNumF(gy/1000,0,220,200);
myGLCD.printNumF(my-70,0,365,200);
myGLCD.print("AZ:",45,215);
myGLCD.print("GZ:",190,215);
myGLCD.print("MZ:",333,215);
myGLCD.printNumF(AZ,4,80,215);
myGLCD.printNumF(gz/1000,0,220,215);
myGLCD.printNumF(my-700,0,365,215);

myGLCD.setFont(BigFont);
myGLCD.setColor(255,0,127);
myGLCD.print("GPS",45,240);

myGLCD.setFont(SmallFont);
myGLCD.setColor(255,0,127);
myGLCD.print("LATITUD:",45,270);
myGLCD.print("LONGITUD:",190,270);
myGLCD.print("ALTITUD:",333,270);
myGLCD.printNumF(gps.location.lat(),4,120,270);
myGLCD.printNumF(gps.location.lng(),4,270,270);
myGLCD.printNumF(gps.altitude.meters(),0,410,270);
myGLCD.print("m",445,270);

myGLCD.printNumI(gps.time.hour(),410,0);
myGLCD.print(":",422,0);
myGLCD.printNumI(gps.time.minute(),435,0);
myGLCD.print(":",448,0);
myGLCD.printNumI(gps.time.second(),455,0);

```

```
myGLCD.printNumI(gps.time.hour(), 410, 0);  
myGLCD.print(":", 422, 0);  
myGLCD.printNumI(gps.time.minute(), 435, 0);  
myGLCD.print(":", 448, 0);  
myGLCD.printNumI(gps.time.second(), 460, 0);
```

10

```
}  
else {  
    digitalWrite (11,HIGH);  
    digitalWrite (12,LOW);  
}  
}
```

10. En esta parte diremos que si el primer if del SD no se cumple encienda la LED roja y apague la verde en señal de que algo esta mal con la SD.