UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TRABAJO FINAL DE GRADO

Grado en Ingeniería Electrónica Industrial y Automática

DESARROLLO DE UN SISTEMA DE ADQUISICIÓN DE DATOS PARA UN CUADRICOPTERO

Anexos

Autor: Javier de Frutos González
Director: Manuel Andrés Manzanares Brotons

Convocatoria: enero 2020

Índice

CÓDIGO DE ARDUINO IDE _____ **5**

CÓDIGO DE PYTHON PARA LA EXTRACCIÓN DE DATOS (1) _____ **15**

PROGRAMA DE PYTHON PARA LA EXTRACCIÓN DE DATOS (2) _____ **17**

INTERFAZ MIT APP INVENTOR _____ **18**

CÓDIGO MIT APP INVENTOR _____ **19**

Código de Arduino IDE

```
/*-----------------------------------------------------------------------------------------------------
 * ------------------SISTEMA DE ADQUISICIÓN DE DATOS PARA UN CUADRICOPTERO-------------------------
------
 -----------------------------------------------------------------------------------------------------*/

//Librerias necesarias para la implementación de los sensores
#include <LiquidCrystal.h>
#include "IRremote.h"
#include "DHT.h"
#include <SFE_BMP180.h>
#include <Wire.h>
#include <SoftwareSerial.h>

#define txPin 11
#define rxPin 10
SoftwareSerial BT1(10,11);
#define DHTTYPE DHT11   // DHT 11

#define anInput A0 // pin analogo del MQ135
#define co2Zero 55 //calibracion CO2 0 nivel
char buffer[10];
SFE_BMP180 pressure;
//Se declaran las variables. Es necesario tomar en cuenta una presión inicial
//esta será la presión que se tome en cuenta en el cálculo de la diferencia de altura
double PresionBase;
double Presion = 0;
int Presion2;
double Altura = 0;
int Altura2;
double Temperatura = 0;
```

```
char status;
LiquidCrystal lcd(7,6,5,4,3,2);
int receiver = 9;
const int DHTPin = 8;
int VALOR;

DHT dht(DHTPin, DHTTYPE);

IRrecv irrecv(receiver);
decode_results results;

void setup()
{
  BT1.flush();
   BT1.begin(9600);
  lcd.begin(16,2);
  dht.begin();
  Serial.begin(9600);
  irrecv.enableIRIn();
  pinMode(anInput,INPUT); //MQ135 declarado como entrada analogica

  //Menu al iniciar el programa
     lcd.clear();
     lcd.setCursor(0,0);
     lcd.print("1-Temp");
     lcd.setCursor(7,0);
     lcd.print("2-Hum");
     lcd.setCursor(0,1);
     lcd.print("3-CO2");
     lcd.setCursor(7,1);
     lcd.print("MENU 2 >>");
```

```
//Inicio de la funcion SensorStart()
SensorStart();

}

 //Función para iniciar el sensor BMP180
 void SensorStart() {
//Secuencia de inicio del sensor
if (pressure.begin())
Serial.println("");
else
{
Serial.println("BMP180 fallo al iniciar (Desconectado?)\n\n");
while (1);
}
//Se inicia la lectura de temperatura
status = pressure.startTemperature();
if (status != 0)  {
delay(status);
//Se lee una temperatura inicial
status = pressure.getTemperature(Temperatura);
if (status != 0)    {
//Se inicia la lectura de presiones
status = pressure.startPressure(3);
if (status != 0)
{
delay(status);
//Se lee la presión inicial incidente sobre el sensor en la primera ejecución
status = pressure.getPressure(PresionBase, Temperatura);
}
}
}
```

```
}


//Funcion para obtener los datos del sensor BMP180
void ReadSensor() {
//En este método se hacen las lecturas de presión y temperatura y se calcula la altura
//Se inicia la lectura de temperatura
status = pressure.startTemperature();
if (status != 0)
{
delay(status);
//Se realiza la lectura de temperatura
status = pressure.getTemperature(Temperatura);
if (status != 0)
{
//Se inicia la lectura de presión
status = pressure.startPressure(3);
if (status != 0)
{
delay(status);
//Se lleva a cabo la lectura de presión,</span>
//considerando la temperatura que afecta el desempeño del sensor</span>
status = pressure.getPressure(Presion, Temperatura);
if (status != 0)
{
//Cálculo de la altura en base a la presión leída en el Setup
Altura = pressure.altitude(Presion, PresionBase);
}
else Serial.println("Error en la lectura de presion\n");
}
else Serial.println("Error iniciando la lectura de presion\n");
}
```

```
else Serial.println("Error en la lectura de temperatura\n");
}
else Serial.println("Error iniciando la lectura de temperatura\n");
}


void loop()
{
 delay(50);
//Se llama la función ReadSensor()
 ReadSensor();
//======LECTURA Y CALCULO DEL VALOR DE MQ135=====

int co2now[10];                    //Vector para las lecturas de MQ135
  int co2raw = 0;                  //Valor original
  int co2ppm = 0;                   //Valor calculado en ppm
  int zzz = 0;                    //Entero para calcular el promedio


  for (int x = 0;x<10;x++)  //guarda 10 valores del CO2 en el vector
  {
    co2now[x]=analogRead(A0);
    delay(50);
  }

  for (int x = 0;x<10;x++)  //Añade los muestreos en zzz
  {
    zzz=zzz + co2now[x];
  }

  co2raw = zzz/10;                  //Divide por 10
  co2ppm = co2raw - co2Zero;            //calcula el valor el ppm
```

```
//======FIN====


//Valor de humedad y temperatura guardado como entero
 int h = dht.readHumidity();
 int t = dht.readTemperature();
  if (irrecv.decode(&results))

 {
   switch(results.value)
   {
     //MENU 1 con la tecla 0
     case 0x00FF6897:
     lcd.clear();
     lcd.setCursor(0,0);
     lcd.print("1-Temp");
     lcd.setCursor(7,0);
     lcd.print("2-Hum");
     lcd.setCursor(0,1);
     lcd.print("3-CO2");
     lcd.setCursor(7,1);
     lcd.print("MENU 2 >>");
     break;
     //MENU 1 con la tecla <<
     case 0x00FF22DD:
     lcd.clear();
     lcd.setCursor(0,0);
     lcd.print("1-Temp");
     lcd.setCursor(7,0);
     lcd.print("2-Hum");
     lcd.setCursor(0,1);
```

```
lcd.print("3-CO2");
lcd.setCursor(7,1);
lcd.print(">> MENU 2");
break;

//segunda parte del menu con el boton >>
case 0x00FFC23D:
lcd.clear();
lcd.setCursor(0,0);
lcd.print("4-Pres");
lcd.setCursor(7,0);
lcd.print("5-Altura");
lcd.setCursor(0,1);
lcd.print("<< MENU 1");
break;

//Temperatura (Boton 1)
case 0x00FF30CF:
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Temp");
lcd.setCursor(0,1);
lcd.print(t);
lcd.print(" C");
lcd.setCursor(0,0);
lcd.setCursor(12,0);
lcd.print("Menu");
lcd.setCursor(14,1);
lcd.print("0");
break;

//Humedad (Boton 2)
```

```
case 0x00FF18E7:
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Humedad");
lcd.setCursor(0,1);
lcd.print(h);
lcd.print(" %");
lcd.setCursor(12,0);
lcd.print("Menu");
lcd.setCursor(14,1);
lcd.print("0");
break;

//CO2 (Boton 3)
case 0x00FF7A85:
lcd.clear();
lcd.setCursor(0,0);
lcd.print("CO2");
lcd.setCursor(0,1);
lcd.print(co2ppm+200);
lcd.print(" ppm");
lcd.setCursor(12,0);
lcd.print("Menu");
lcd.setCursor(14,1);
lcd.print("0");
break;

//Presion (Boton 4)
case 0x00FF10EF:
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Presion");
```

```
    lcd.setCursor(0,1);
    lcd.print(Presion);
    lcd.print(" mmBar");
    lcd.setCursor(12,0);
    lcd.print("Menu");
    lcd.setCursor(14,1);
    lcd.print("0");
    break;

    //Altura (Boton 5)
    case 0x00FF38C7:
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Altura");
    lcd.setCursor(0,1);
    lcd.print(Altura);
    lcd.print(" m");
    lcd.setCursor(12,0);
    lcd.print("Menu");
    lcd.setCursor(14,1);
    lcd.print("0");
   }
   irrecv.resume();
   Serial.println(results.value, HEX);
   Serial.println(buffer);
  }
  //Guardamos los valores de presion y altura en valores enteros para mostrarlo en la App
   Presion2=int(Presion);
   Altura2=int(Altura);
   //Guardamos los valores de temperatura, humedad, CO2, presion y altura en el buffer (Se mostrarán
en ese orden)
   sprintf(buffer, "%d,%d,%d,%d,%d",t,h,co2ppm+200,Presion2,Altura2);
```

```
  //Se mandan los valores del buffer a traves del modulo bluetooth
   BT1.println(buffer);
   Serial.println(buffer);
 delay(50);
}
```

Código de Python para la extracción de datos (1)

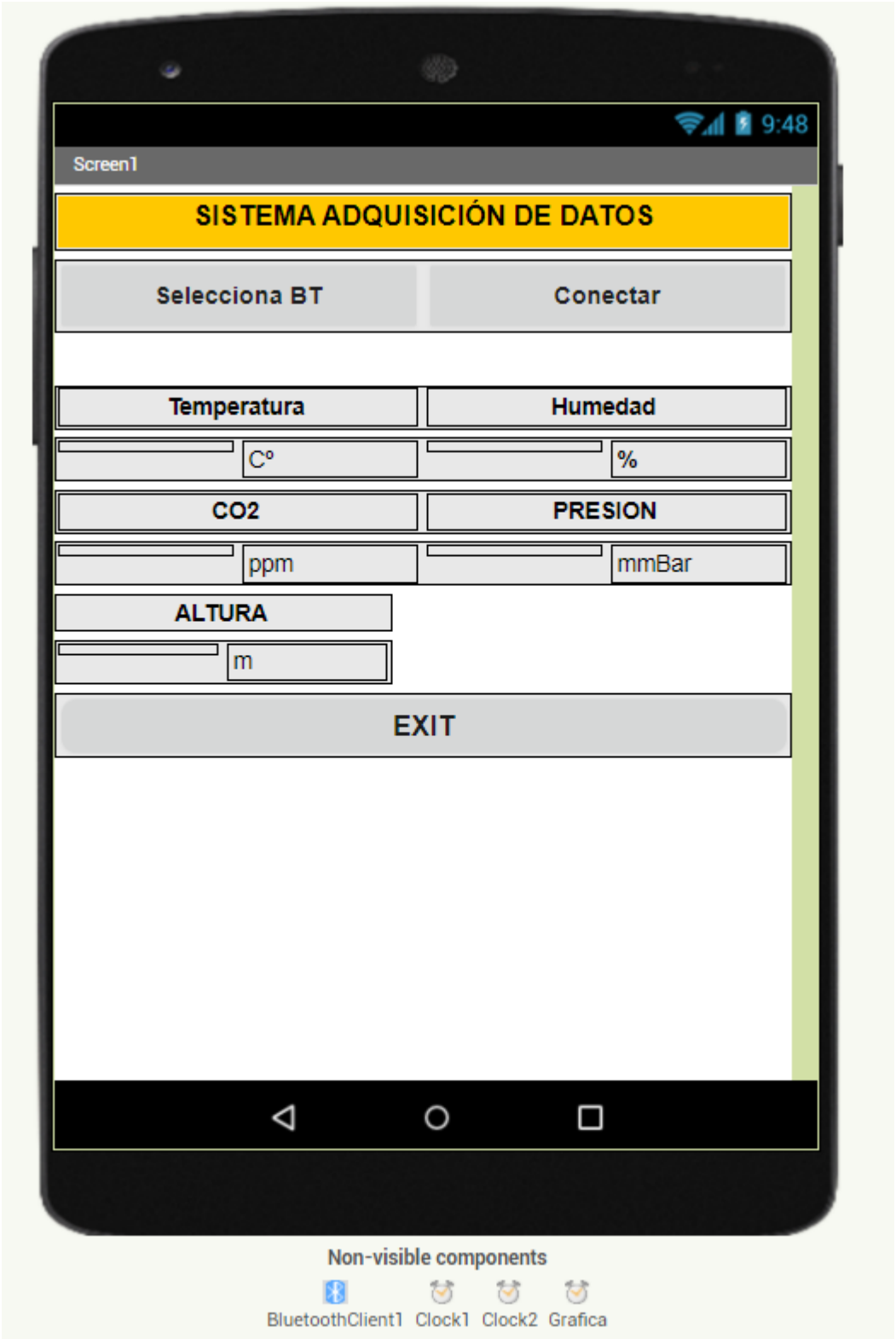
```
import serial
import xlwt
from datetime import datetime

class SerialToExcel:

    def __init__(self,port,speed):

        self.port = port
        self.speed = speed

        self.wb = xlwt.Workbook()
        self.ws = self.wb.add_sheet("Data from Serial",cell_overwrite_ok=True)
        self.ws.write(0, 0, "Data from Serial")
        self.columns = ["Date Time"]
        self.number = 100


    def setColumns(self,col):
        self.columns.extend(col)

    def setRecordsNumber(self,number):
        self.number = number

    def readPort(self):
        ser = serial.Serial(self.port, self.speed, timeout=1)
        c = 0
        for col in self.columns:
            self.ws.write(1, c, col)
            c = c + 1
        self.fila = 2
```

```
    i = 0
    while(i<self.number):
        line = str(ser.readline())
        if(len(line) > 0):
            now = datetime.now()
            date_time = now.strftime("%m/%d/%Y, %H:%M:%S")
            print(date_time,line)
            if(line.find(",")):
                c = 1
                self.ws.write(self.fila, 0, date_time)
                columnas = line.split(",")
                for col in columnas:
                    self.ws.write(self.fila, c, col)
                    c = c + 1

            i = i + 1
            self.fila = self.fila + 1

def writeFile(self,archivo):
            self.wb.save(archivo)
```

Programa de Python para la extracción de datos (2)

```
from serialToExcel import SerialToExcel

serialToExcel = SerialToExcel("COM3",9600)

columnas = ["Temperatura","Humedad","CO2","Presion","Altura"]

serialToExcel.setColumns(["Temperatura","Humedad","CO2","Presion","Altura"])
serialToExcel.setRecordsNumber(100)
serialToExcel.readPort()

serialToExcel.writeFile("Prueba.xls")
```

Interfaz MIT App Inventor



Código MIT App Inventor