Project Title:

“Sunlight Tracking Flower Pot” with Autonomous Robot

BY: FATEMEH CHANGIZIAN, MOHSEN AHADI

Project Description:

This mobile flower pot can track the brightest spots in a room using its light sensitive module and navigate to the spot with avoiding the obstacles in its path. It has the ability to gather information about flower’s health using temperature and soil humidity sensors and send it to the user through a cloud-based platform by using ThingSpeak Platform and also these data will be available on an android app.

Project Requirements:

- Hardware part:

1. Arduino UNO as The Main Processor
2. ESP8266-01 (Communication Module (WIFI))
3. DS18B20 (1-Wire Digital Temperature Sensor for use on soil)
4. YL-69 + LM393 (Soil Humidity Sensor)
5. L298N H-bridge Dual Motor Controller Module (Motor Driver)
6. TSL2561 Luminosity Sensor
7. SRF04 Ultra Sonic Ranger Module
8. SG90 Mini Gear Micro Servo Motor
9. Power Bank REMAX
10. 2 Wheeled Smart Car Robot Chassis

- Software part:

1. ThingSpeak Platform (Open Source IOT Platform)
2. FAMO App (For Sensor Status Monitoring - MIT App Inventor)

Project Features:

1) The ability to detect the intensity of ambient light with the help of the Luminosity Sensor and orientation towards the brightest spots in a room
2) Ability to stop at the light intensity required to grow the plant and provide appropriate conditions for plant

3) Ability to detect obstacles using the ultra-sonic sensor and continue the initial route after crossing the obstacle
4) Use Servo Motor to reduce the number of required range finders and cover over 180 degrees of robot vision
5) Ability to adjust the speed of the robot
6) It's easy to save battery life by going into Standby (sleep) mode
7) Benefit from rechargeable power supplies for powering the system (Power Bank)
8) Ability to show percentage of battery charge on FAMO app and ThingSpeak platform
9) Possibility to declare the label in the event of a lack of temperature or soil moisture on the app
10) Ability to send information (Light in %, Soil Humidity in %, Soil Temperature in °C, Battery Charge in %) using the ThinkSpeak platform Wi-Fi module and the FAMO application
11) Ability to login in the app
12) Ability to save instant data as text file by application
13) Ability to share the data file within the app

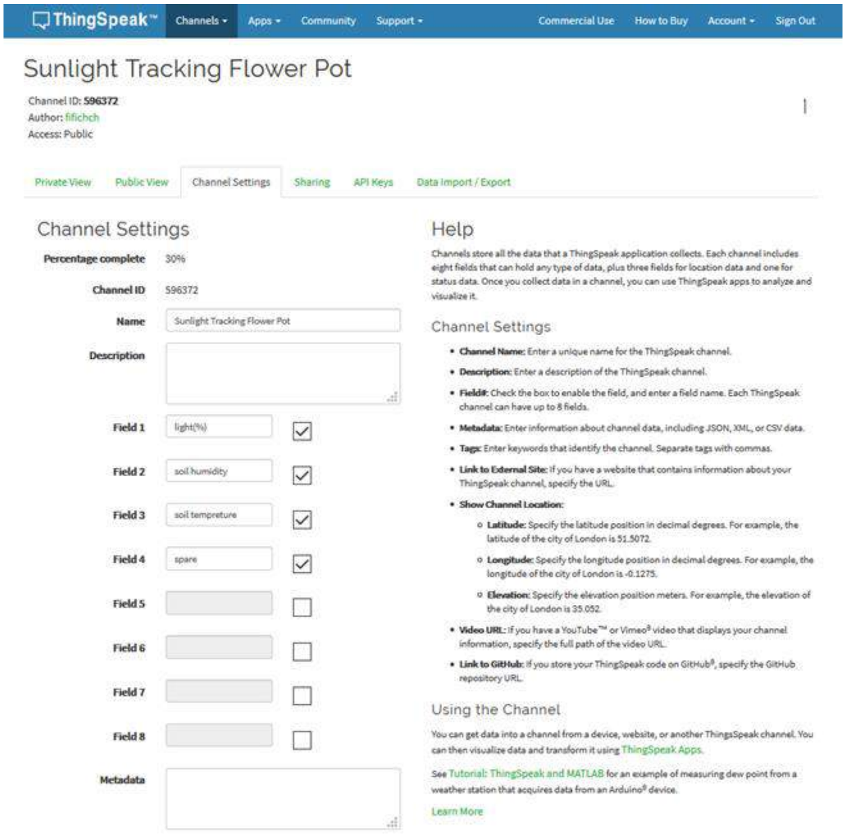
Possible Extra Features to Add:

1) Controlling and commanding the robot remotely using FAMO application on a mobile phone.
2) The ability to turn back and find a shady spot after the plant is fed properly from the sunlight.
3) Information about Numerous types of plants can be available to choose from, on the application. The user can choose what plant is going to be on the robot and leave it to the application to take care of it.
4) Robot's power supply can be more efficient environmentally if we replace the batteries with a small solar panel, since the robot's mission is to stay in the sunlight too.

Programming part:

Here is the code for software part:

1. The ThingSpeak platform



Sunlight Tracking Flower Pot

Channel ID: **596372**
Author: fifichch
Access: Public

Channel Settings

Percentage complete	30%	
Channel ID	596372	
Name	Sunlight Tracking Flower Pot	
Description		
Field 1	light(%)	☑
Field 2	soil humidity	☑
Field 3	soil tempreture	☑
Field 4	spare	☑
Field 5		☐
Field 6		☐
Field 7		☐
Field 8		☐
Metadata		

Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

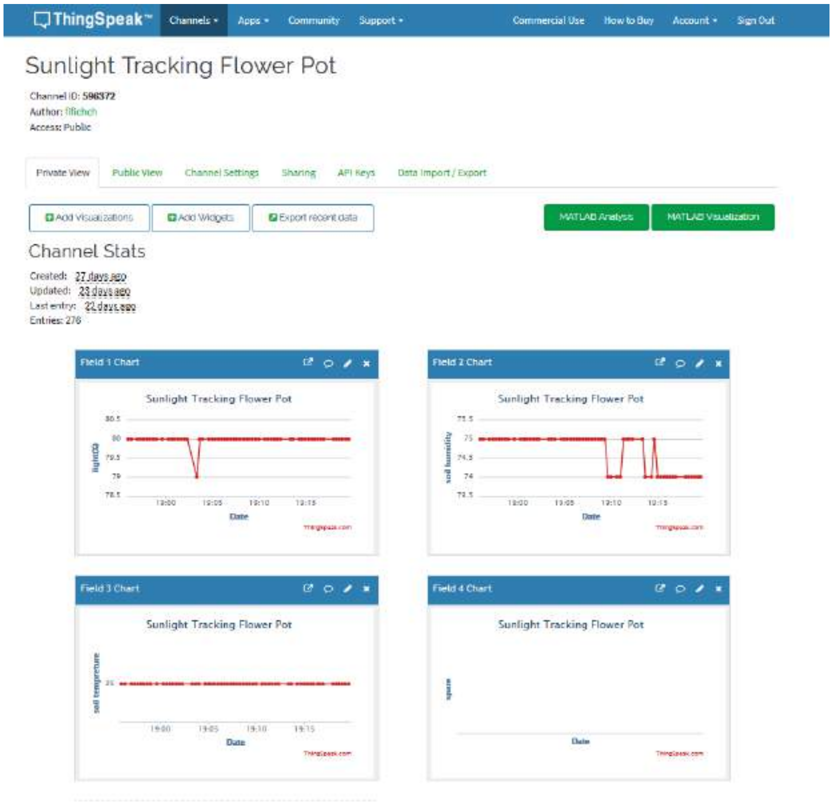
- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Link to External Site:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- **Show Channel Location:**
  - **Latitude:** Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
  - **Longitude:** Specify the longitude position in decimal degrees. For example, the longitude of the city of London is -0.1275.
  - **Elevation:** Specify the elevation position meters. For example, the elevation of the city of London is 35.052.
- **Video URL:** If you have a YouTube™ or Vimeo® video that displays your channel information, specify the full path of the video URL.
- **Link to GitHub:** If you store your ThingSpeak code on GitHub®, specify the GitHub repository URL.

Using the Channel

You can get data into a channel from a device, website, or another ThingsSpeak channel. You can then visualize data and transform it using ThingSpeak Apps.

See Tutorial: ThingSpeak and MATLAB for an example of measuring dew point from a weather station that acquires data from an Arduino® device.

Learn More

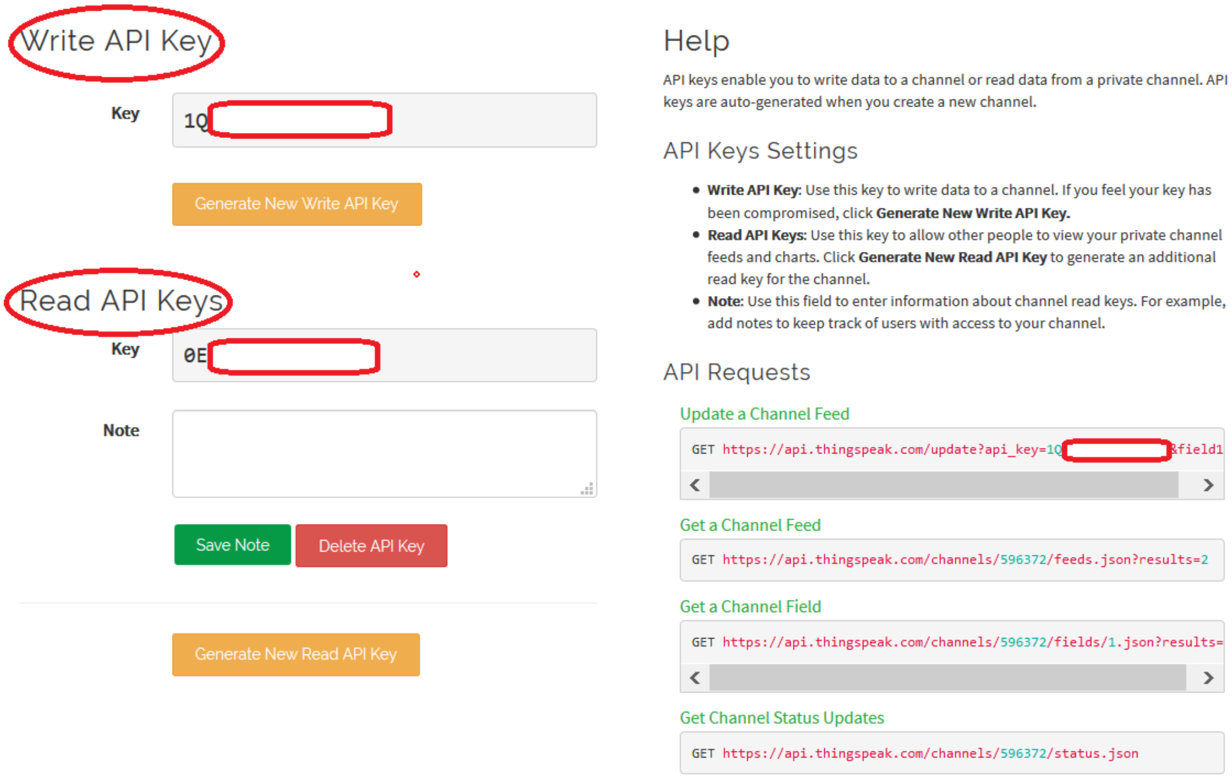


Sunlight Tracking Flower Pot

Channel ID: **596372**
Author: fifichch
Access: Public

Channel Stats

Created: 27 days ago
Updated: 23 days ago
Last entry: 22 days ago
Entries: 276



Write API Key

Key 1Q[redacted]

Read API Keys

Key 0E[redacted]

Note

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- **Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key.**
- **Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- **Note:** Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

Update a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=1Q[redacted]&field1
```

Get a Channel Feed

```
GET https://api.thingspeak.com/channels/596372/feeds.json?results=2
```

Get a Channel Field

```
GET https://api.thingspeak.com/channels/596372/fields/1.json?results=
```

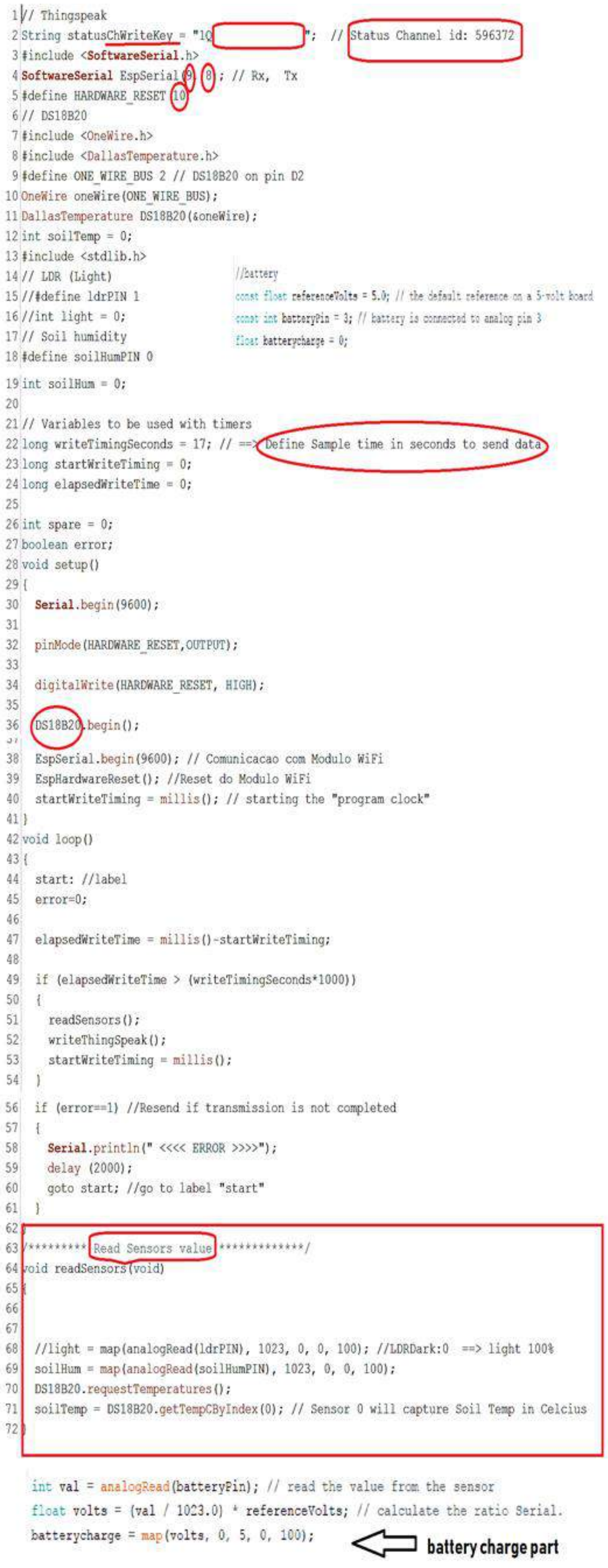
Get Channel Status Updates

```
GET https://api.thingspeak.com/channels/596372/status.json
```

2. Sending Sensors Status from Arduino to the Cloud

The ESP-01 will be used as a Serial Bridge.

The below code will do the work for us:

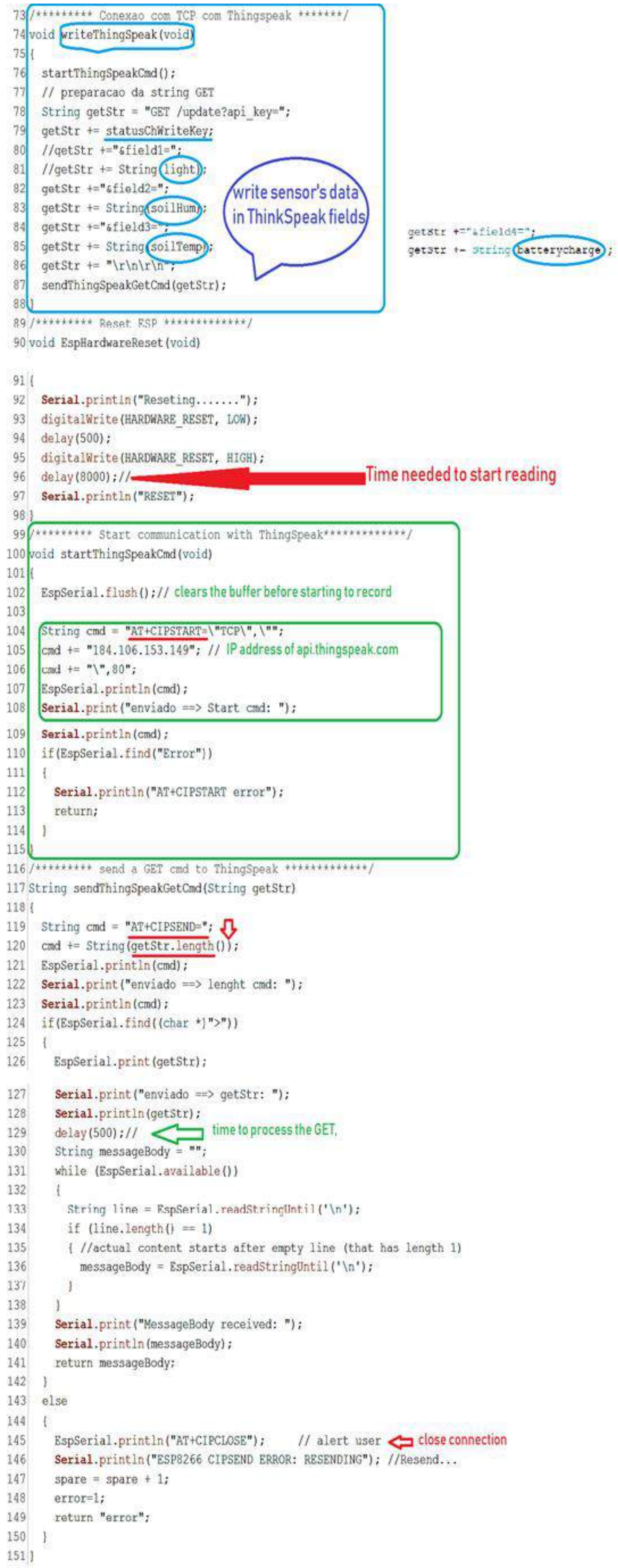


```
// Thingspeak
String statusChWriteKey = "1Q[redacted]";  // Status Channel id: 596372
#include <SoftwareSerial.h>
SoftwareSerial EspSerial(9, 8); // Rx,  Tx
#define HARDWARE_RESET 10
// DS18B20
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 2 // DS18B20 on pin D2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature DS18B20(&oneWire);
int soilTemp = 0;
#include <stdlib.h>
// LDR (Light)
//#define ldrPIN 1
//int light = 0;
// Soil humidity
#define soilHumPIN 0
int soilHum = 0;

// Variables to be used with timers
long writeTimingSeconds = 17; // ==> Define Sample time in seconds to send data
long startWriteTiming = 0;
long elapsedWriteTime = 0;

int spare = 0;
boolean error;
void setup()
{
  Serial.begin(9600);

  pinMode(HARDWARE_RESET,OUTPUT);

  digitalWrite(HARDWARE_RESET, HIGH);

  DS18B20.begin();

  EspSerial.begin(9600); // Comunicacao com Modulo WiFi
  EspHardwareReset(); //Reset do Modulo WiFi
  startWriteTiming = millis(); // starting the "program clock"
}
void loop()
{
  start: //label
  error=0;

  elapsedWriteTime = millis()-startWriteTiming;

  if (elapsedWriteTime > (writeTimingSeconds*1000))
  {
    readSensors();
    writeThingSpeak();
    startWriteTiming = millis();
  }
  if (error==1) //Resend if transmission is not completed
  {
    Serial.println(" <<<< ERROR >>>>");
    delay (2000);
    goto start; //go to label "start"
  }
}
/********* Read Sensors value *************/
void readSensors(void)
{


  //light = map(analogRead(ldrPIN), 1023, 0, 0, 100); //LDRDark:0  ==> light 100%
  soilHum = map(analogRead(soilHumPIN), 1023, 0, 0, 100);
  DS18B20.requestTemperatures();
  soilTemp = DS18B20.getTempCByIndex(0); // Sensor 0 will capture Soil Temp in Celcius
}
```

```
//battery
const float referenceVolts = 5.0; // the default reference on a 5-volt board
const int batteryPin = 3; // battery is connected to analog pin 3
float batterycharge = 0;
```

```
int val = analogRead(batteryPin); // read the value from the sensor
float volts = (val / 1023.0) * referenceVolts; // calculate the ratio Serial.
batterycharge = map(volts, 0, 5, 0, 100);
```

battery charge part



```
/********* Conexao com TCP com Thingspeak *******/
void writeThingSpeak(void)
{
  startThingSpeakCmd();
  // preparacao da string GET
  String getStr = "GET /update?api_key=";
  getStr += statusChWriteKey;
  //getStr +="&field1=";
  //getStr += String(light);
  getStr +="&field2=";
  getStr += String(soilHum);
  getStr +="&field3=";
  getStr += String(soilTemp);
  getStr += "\r\n\r\n";
  sendThingSpeakGetCmd(getStr);
}
/********* Reset ESP *************/
void EspHardwareReset(void)
{
  Serial.println("Reseting.......");
  digitalWrite(HARDWARE_RESET, LOW);
  delay(500);
  digitalWrite(HARDWARE_RESET, HIGH);
  delay(8000);//
  Serial.println("RESET");
}
/********* Start communication with ThingSpeak*************/
void startThingSpeakCmd(void)
{
  EspSerial.flush();// clears the buffer before starting to record

  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += "184.106.153.149"; // IP address of api.thingspeak.com
  cmd += "\",80";
  EspSerial.println(cmd);
  Serial.print("enviado ==> Start cmd: ");
  Serial.println(cmd);
  if(EspSerial.find("Error"))
  {
    Serial.println("AT+CIPSTART error");
    return;
  }
}
/********* send a GET cmd to ThingSpeak *************/
String sendThingSpeakGetCmd(String getStr)
{
  String cmd = "AT+CIPSEND=";
  cmd += String(getStr.length());
  EspSerial.println(cmd);
  Serial.print("enviado ==> lenght cmd: ");
  Serial.println(cmd);
  if(EspSerial.find((char *)">"))
  {
    EspSerial.print(getStr);
    Serial.print("enviado ==> getStr: ");
    Serial.println(getStr);
    delay(500);// 
    String messageBody = "";
    while (EspSerial.available())
    {
      String line = EspSerial.readStringUntil('\n');
      if (line.length() == 1)
      { //actual content starts after empty line (that has length 1)
        messageBody = EspSerial.readStringUntil('\n');
      }
    }
    Serial.print("MessageBody received: ");
    Serial.println(messageBody);
    return messageBody;
  }
  else
  {
    EspSerial.println("AT+CIPCLOSE");     // alert user
    Serial.println("ESP8266 CIPSEND ERROR: RESENDING"); //Resend...
    spare = spare + 1;
    error=1;
    return "error";
  }
}
```

write sensor's data in ThinkSpeak fields

```
getStr +="&field4=";
getStr += String(batterycharge);
```

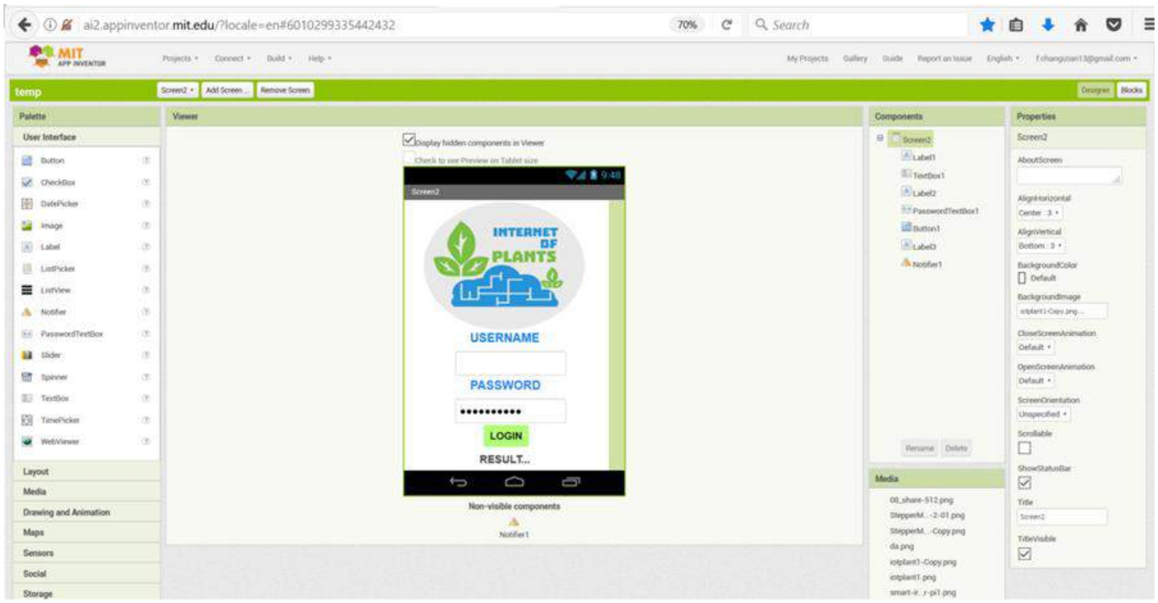
Time needed to start reading

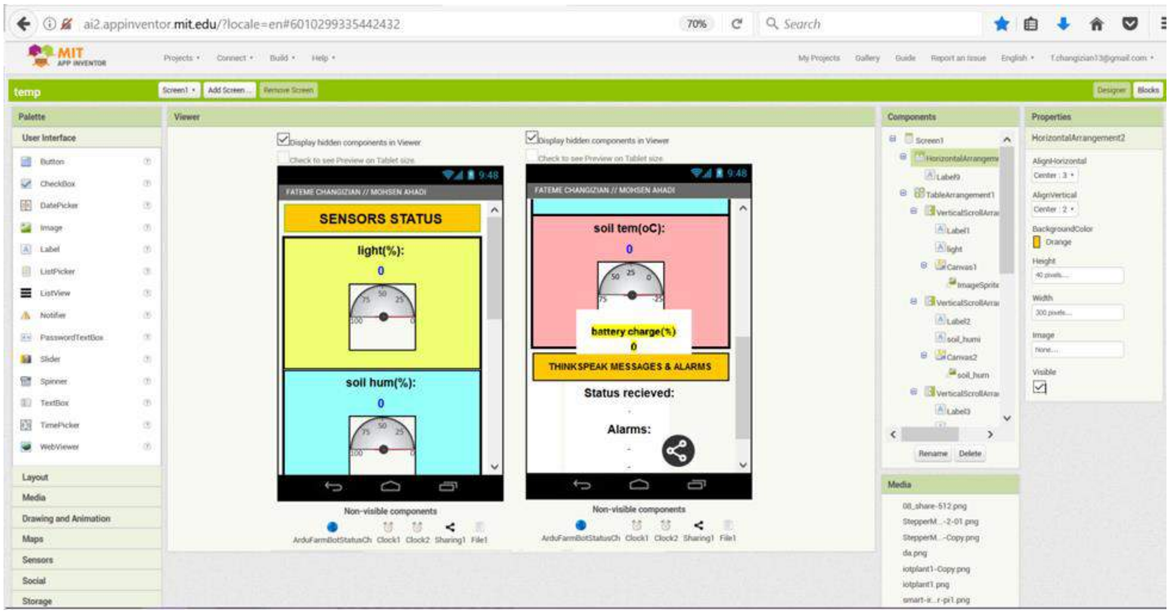
time to process the GET,

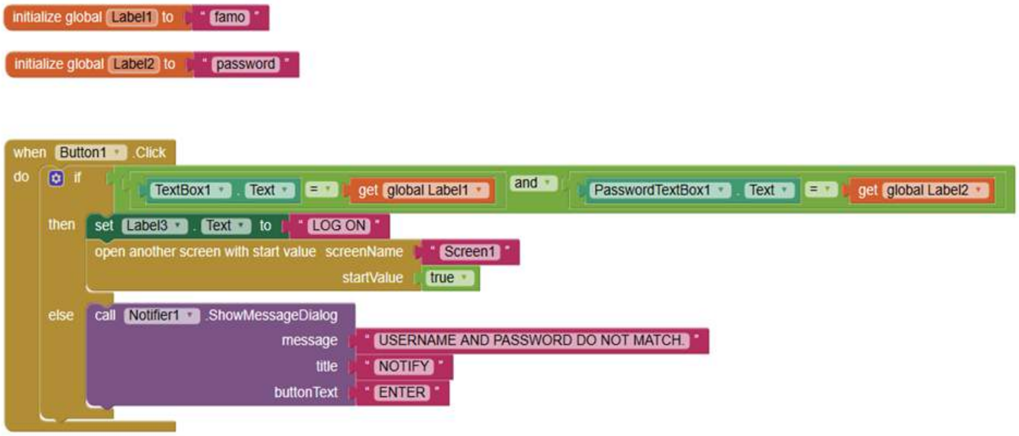
close connection

3. The Android App (FAMO) - Status Monitoring

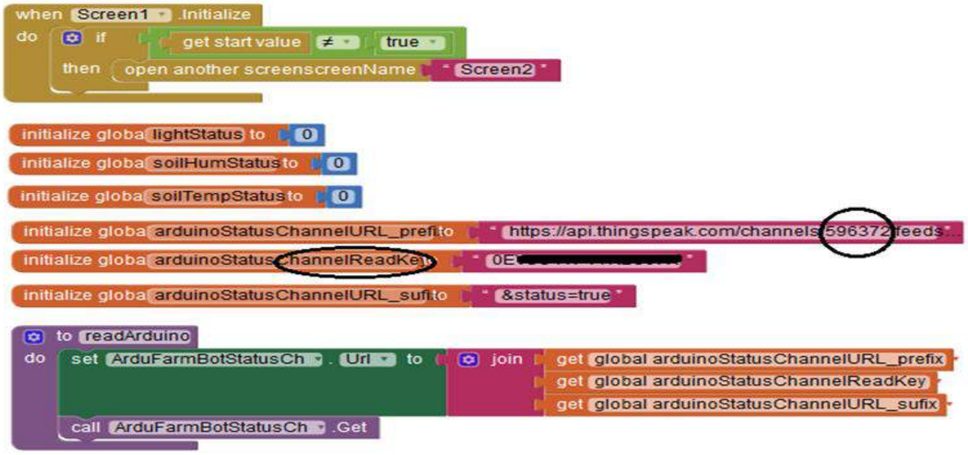
we must design the blocks and the user interface of the application.



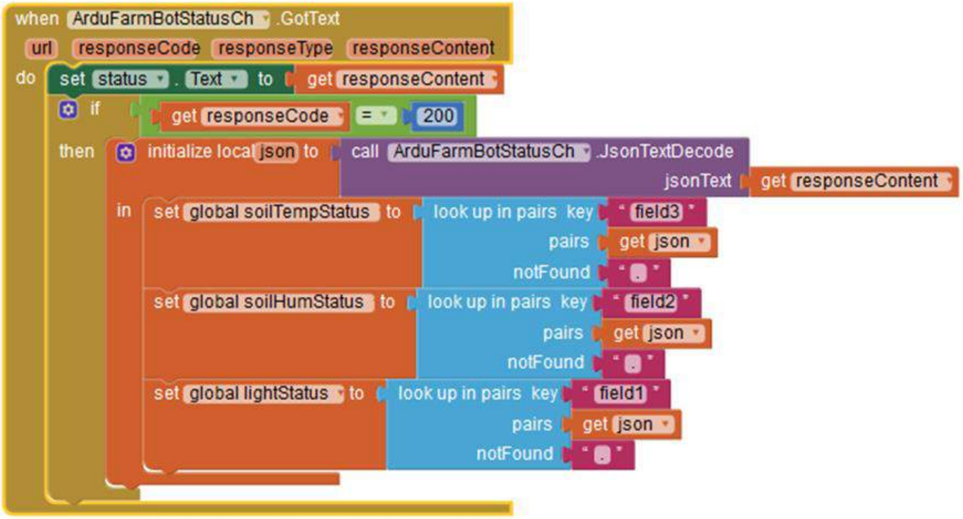




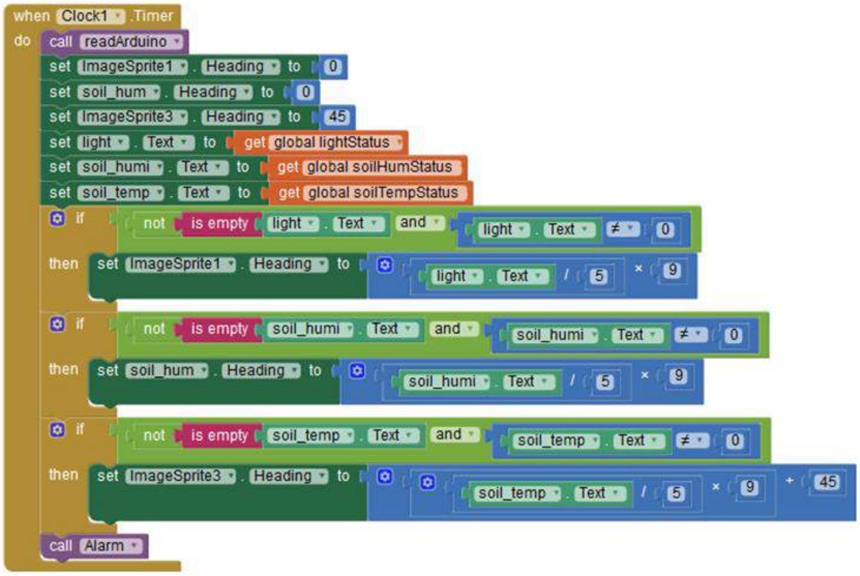
```
initialize global Label1 to " famo "

initialize global Label2 to " password "

when Button1 .Click
do  if    TextBox1 . Text = get global Label1  and  PasswordTextBox1 . Text = get global Label2
    then  set Label3 . Text to " LOG ON "
          open another screen with start value  screenName " Screen1 "
                                                startValue true
    else  call Notifier1 .ShowMessageDialog
                                    message " USERNAME AND PASSWORD DO NOT MATCH. "
                                    title " NOTIFY "
                                    buttonText " ENTER "
```



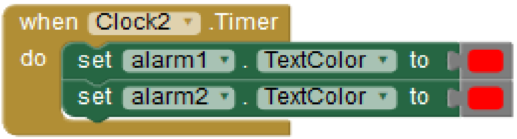
```
when Screen1 .Initialize
do  if    get start value ≠ true
    then  open another screenscreenName " Screen2 "

initialize globa lightStatus to 0
initialize globa soilHumStatus to 0
initialize globa soilTempStatus to 0
initialize globa arduinoStatusChannelURL_prefi to " https://api.thingspeak.com/channels/596372/feeds "...
initialize globa arduinoStatusChannelReadKey " 0E[redacted] "
initialize globa arduinoStatusChannelURL_sufi to " &status=true "

to readArduino
do  set ArduFarmBotStatusCh . Url to  join  get global arduinoStatusChannelURL_prefix
                                            get global arduinoStatusChannelReadKey
                                            get global arduinoStatusChannelURL_sufix
    call ArduFarmBotStatusCh .Get
```



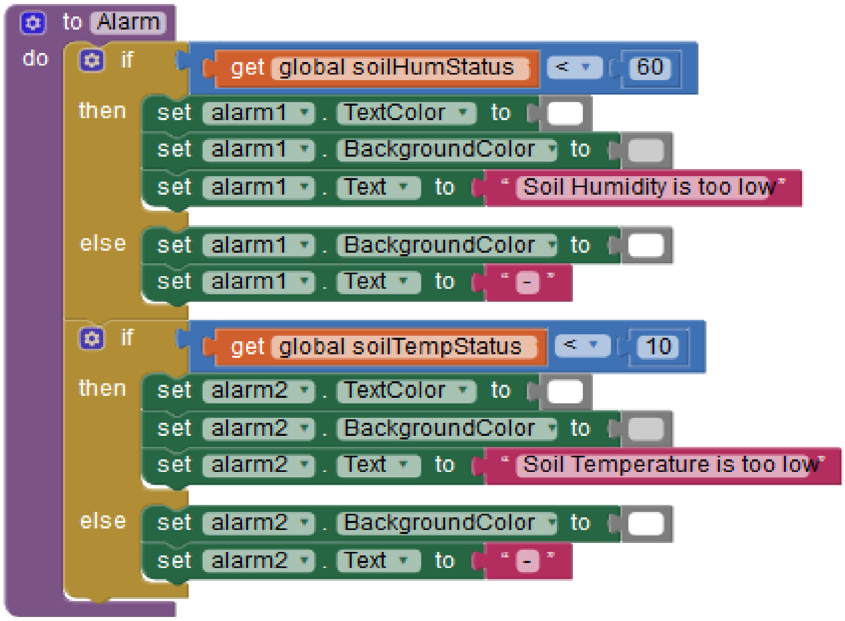
```
when ArduFarmBotStatusCh .GotText
 url  responseCode  responseType  responseContent
do  set status . Text to get responseContent
    if    get responseCode = 200
    then  initialize local json to  call ArduFarmBotStatusCh .JsonTextDecode
                                                          jsonText get responseContent
          in  set global soilTempStatus to  look up in pairs  key " field3 "
                                                              pairs get json
                                                              notFound " "
              set global soilHumStatus to  look up in pairs  key " field2 "
                                                             pairs get json
                                                             notFound " "
              set global lightStatus to  look up in pairs  key " field1 "
                                                           pairs get json
                                                           notFound " "
```



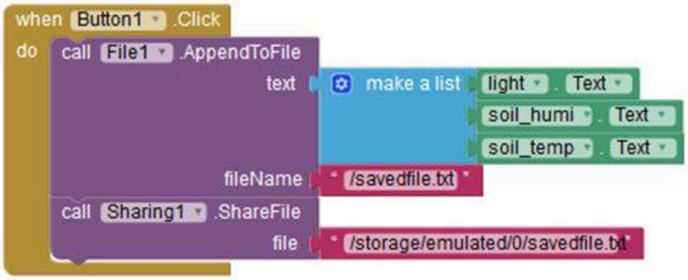
```
when Clock1 .Timer
do  call readArduino
    set ImageSprite1 . Heading to 0
    set soil_hum . Heading to 0
    set ImageSprite3 . Heading to 45
    set light . Text to get global lightStatus
    set soil_humi . Text to get global soilHumStatus
    set soil_temp . Text to get global soilTempStatus
    if   not is empty light . Text and light . Text ≠ 0
    then set ImageSprite1 . Heading to light . Text / 5 × 9
    if   not is empty soil_humi . Text and soil_humi . Text ≠ 0
    then set soil_hum . Heading to soil_humi . Text / 5 × 9
    if   not is empty soil_temp . Text and soil_temp . Text ≠ 0
    then set ImageSprite3 . Heading to soil_temp . Text / 5 × 9 + 45
    call Alarm
```



```
when Clock2 .Timer
do  set alarm1 . TextColor to 🟥
    set alarm2 . TextColor to 🟥
```

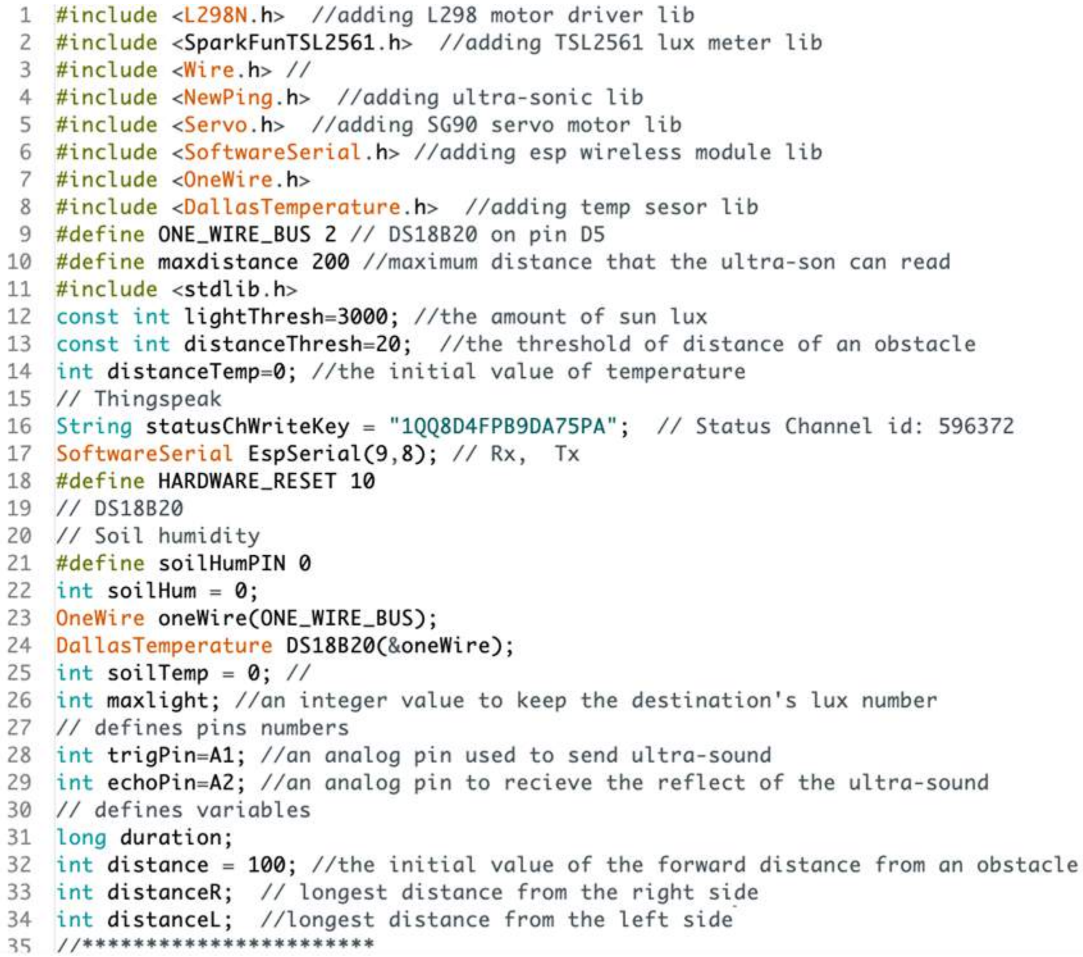


```
to Alarm
do  if   get global soilHumStatus < 60
    then set alarm1 . TextColor to ⬜
         set alarm1 . BackgroundColor to ▨
         set alarm1 . Text to " Soil Humidity is too low "
    else set alarm1 . BackgroundColor to ⬜
         set alarm1 . Text to " - "
    if   get global soilTempStatus < 10
    then set alarm2 . TextColor to ⬜
         set alarm2 . BackgroundColor to ▨
         set alarm2 . Text to " Soil Temperature is too low "
    else set alarm2 . BackgroundColor to ⬜
         set alarm2 . Text to " - "
```

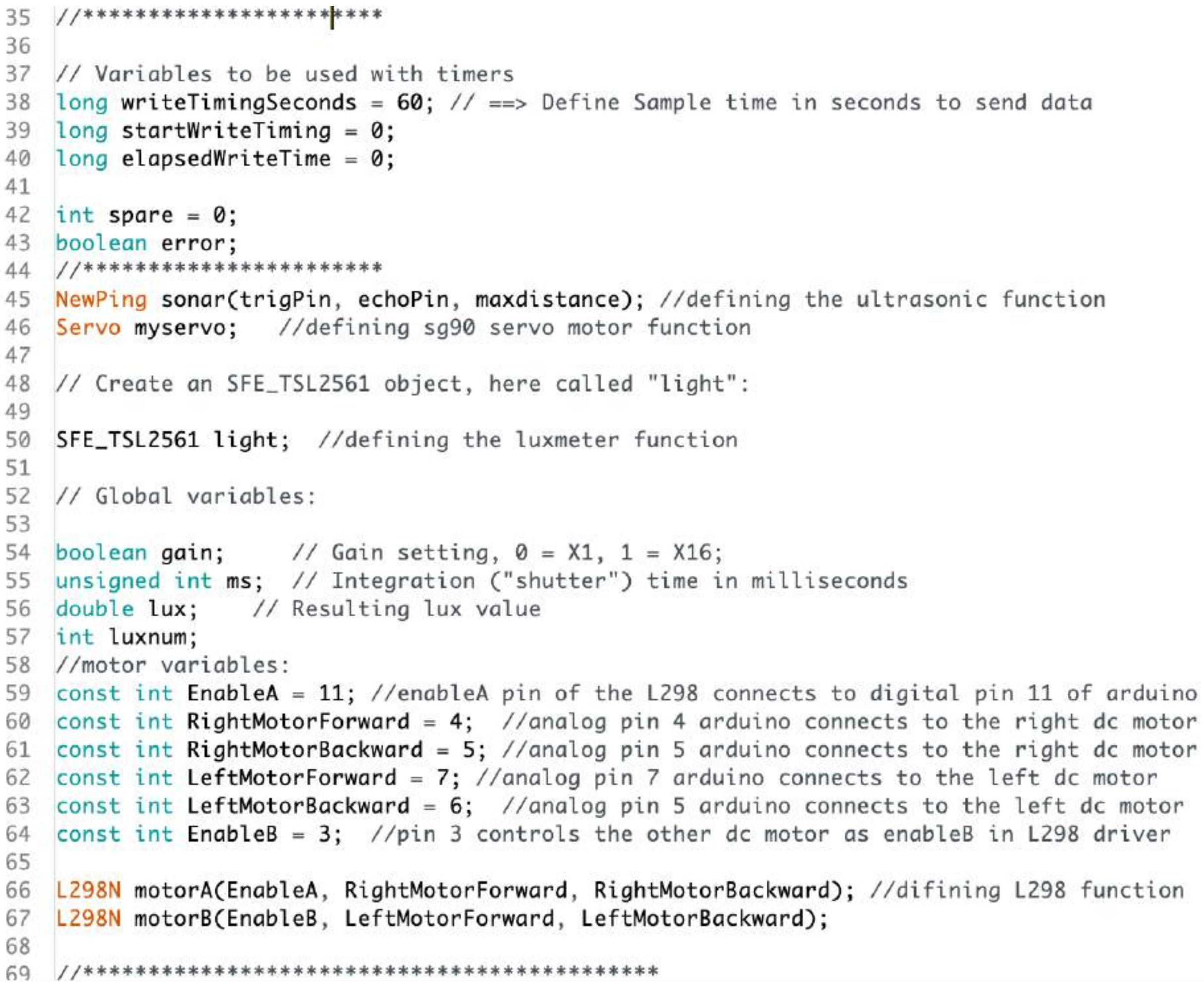


Here is the code for hardware part:

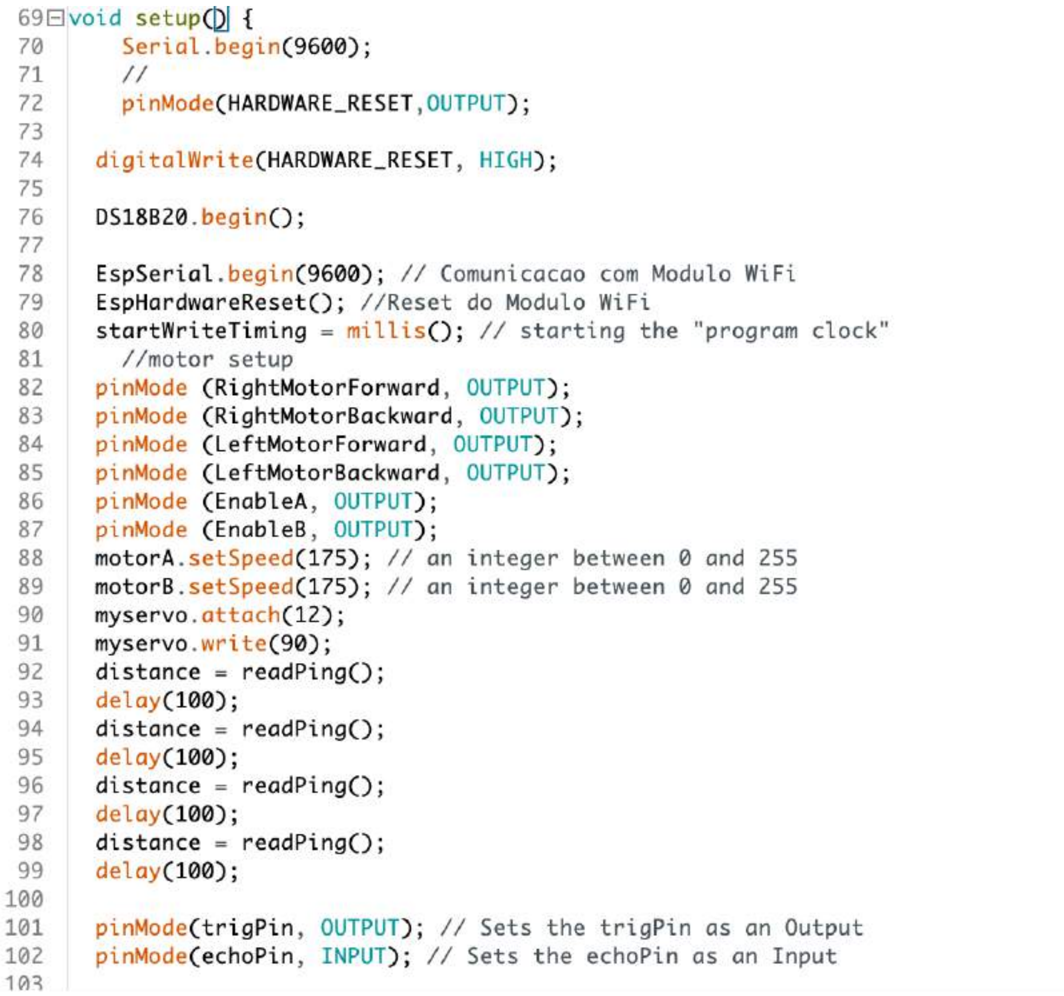
Robot programming:



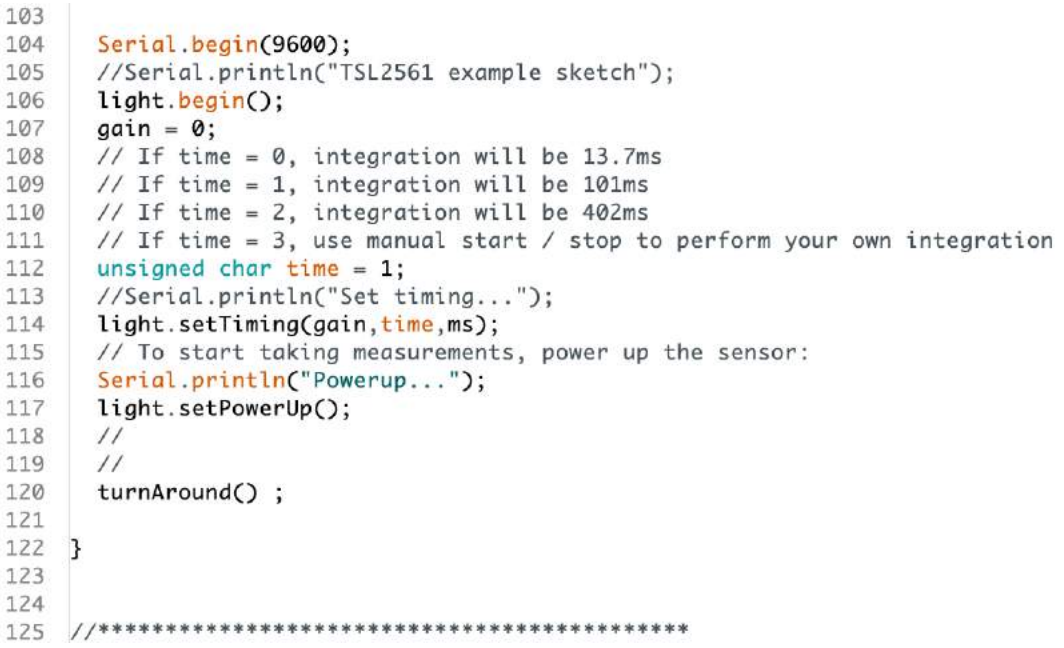
```
#include <L298N.h>  //adding L298 motor driver lib
#include <SparkFunTSL2561.h>  //adding TSL2561 lux meter lib
#include <Wire.h> //
#include <NewPing.h>  //adding ultra-sonic lib
#include <Servo.h>  //adding SG90 servo motor lib
#include <SoftwareSerial.h> //adding esp wireless module lib
#include <OneWire.h>
#include <DallasTemperature.h>  //adding temp sesor lib
#define ONE_WIRE_BUS 2 // DS18B20 on pin D5
#define maxdistance 200 //maximum distance that the ultra-son can read
#include <stdlib.h>
const int lightThresh=3000; //the amount of sun lux
const int distanceThresh=20;  //the threshold of distance of an obstacle
int distanceTemp=0; //the initial value of temperature
// Thingspeak
String statusChWriteKey = "1QQ8D4FPB9DA75PA";  // Status Channel id: 596372
SoftwareSerial EspSerial(9,8); // Rx,  Tx
#define HARDWARE_RESET 10
// DS18B20
// Soil humidity
#define soilHumPIN 0
int soilHum = 0;
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature DS18B20(&oneWire);
int soilTemp = 0; //
int maxlight; //an integer value to keep the destination's lux number
// defines pins numbers
int trigPin=A1; //an analog pin used to send ultra-sound
int echoPin=A2; //an analog pin to recieve the reflect of the ultra-sound
// defines variables
long duration;
int distance = 100; //the initial value of the forward distance from an obstacle
int distanceR;  // longest distance from the right side
int distanceL;  //longest distance from the left side
//***********************
```



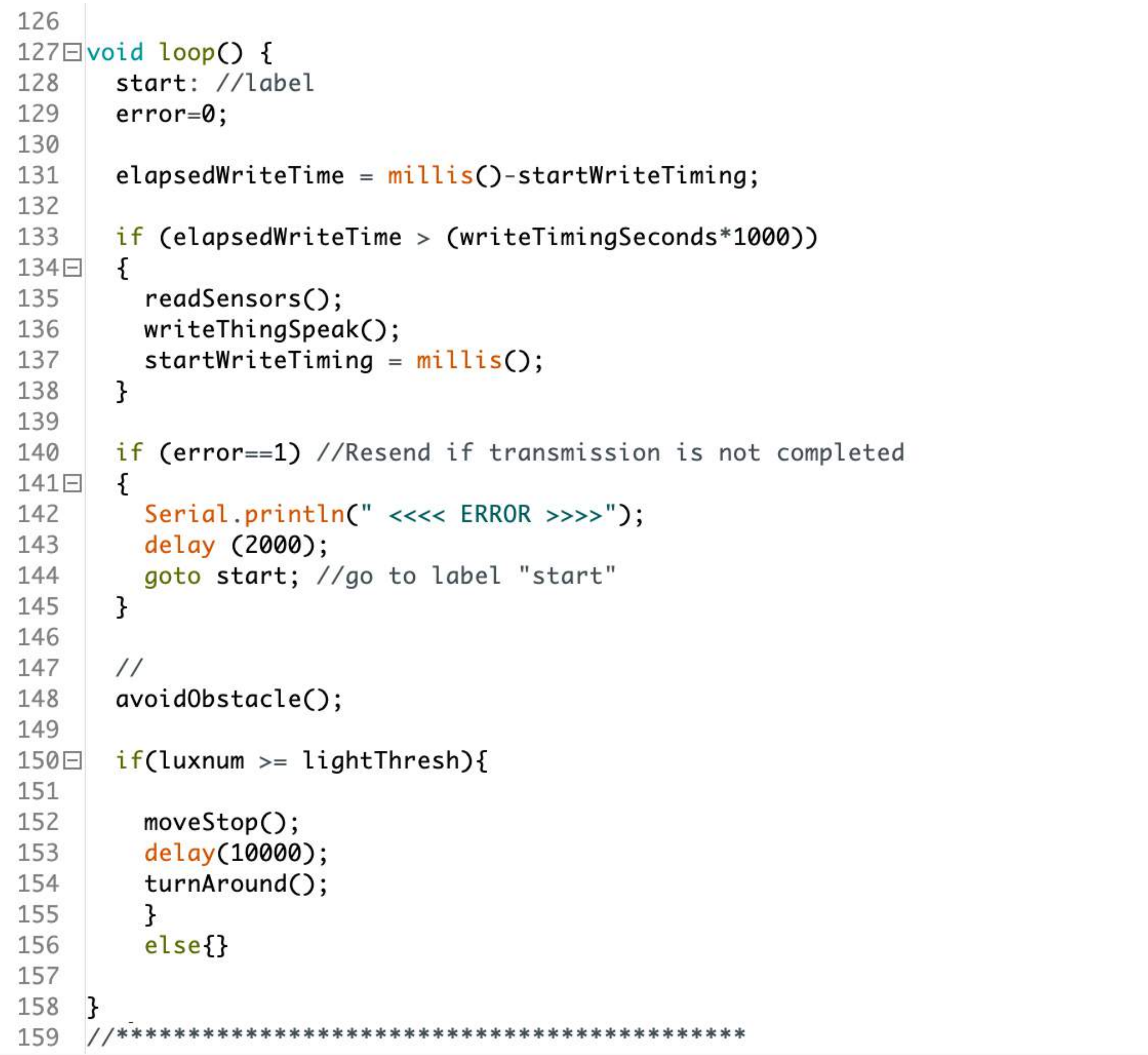
```
//***********************

// Variables to be used with timers
long writeTimingSeconds = 60; // ==> Define Sample time in seconds to send data
long startWriteTiming = 0;
long elapsedWriteTime = 0;

int spare = 0;
boolean error;
//***********************
NewPing sonar(trigPin, echoPin, maxdistance); //defining the ultrasonic function
Servo myservo;   //defining sg90 servo motor function

// Create an SFE_TSL2561 object, here called "light":

SFE_TSL2561 light;  //defining the luxmeter function

// Global variables:

boolean gain;     // Gain setting, 0 = X1, 1 = X16;
unsigned int ms;  // Integration ("shutter") time in milliseconds
double lux;    // Resulting lux value
int luxnum;
//motor variables:
const int EnableA = 11; //enableA pin of the L298 connects to digital pin 11 of arduino
const int RightMotorForward = 4;  //analog pin 4 arduino connects to the right dc motor
const int RightMotorBackward = 5; //analog pin 5 arduino connects to the right dc motor
const int LeftMotorForward = 7; //analog pin 7 arduino connects to the left dc motor
const int LeftMotorBackward = 6;  //analog pin 5 arduino connects to the left dc motor
const int EnableB = 3;  //pin 3 controls the other dc motor as enableB in L298 driver

L298N motorA(EnableA, RightMotorForward, RightMotorBackward); //difining L298 function
L298N motorB(EnableB, LeftMotorForward, LeftMotorBackward);

//********************************************
```



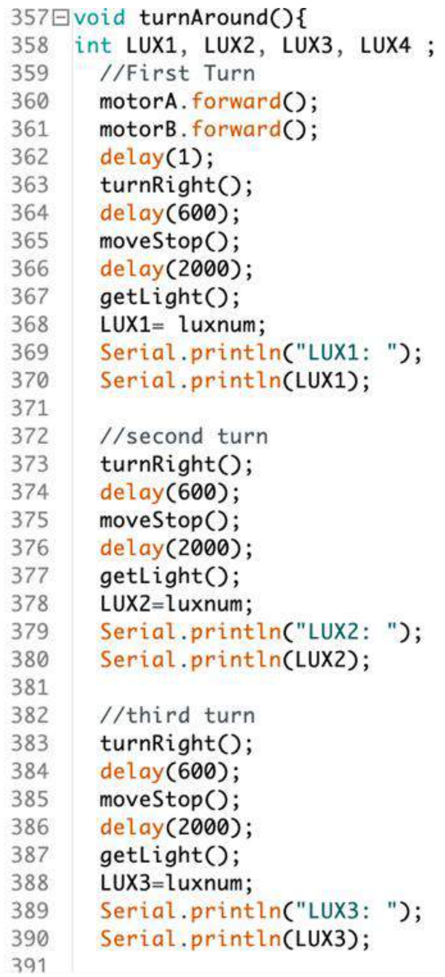
```
void setup() {
    Serial.begin(9600);
    //
    pinMode(HARDWARE_RESET,OUTPUT);

  digitalWrite(HARDWARE_RESET, HIGH);

  DS18B20.begin();

  EspSerial.begin(9600); // Comunicacao com Modulo WiFi
  EspHardwareReset(); //Reset do Modulo WiFi
  startWriteTiming = millis(); // starting the "program clock"
    //motor setup
  pinMode (RightMotorForward, OUTPUT);
  pinMode (RightMotorBackward, OUTPUT);
  pinMode (LeftMotorForward, OUTPUT);
  pinMode (LeftMotorBackward, OUTPUT);
  pinMode (EnableA, OUTPUT);
  pinMode (EnableB, OUTPUT);
  motorA.setSpeed(175); // an integer between 0 and 255
  motorB.setSpeed(175); // an integer between 0 and 255
  myservo.attach(12);
  myservo.write(90);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);

  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input

```



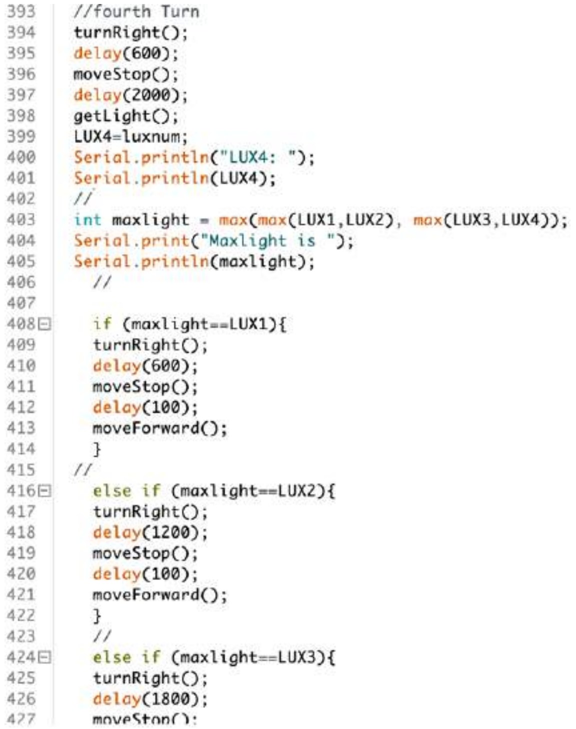
```

  Serial.begin(9600);
  //Serial.println("TSL2561 example sketch");
  light.begin();
  gain = 0;
  // If time = 0, integration will be 13.7ms
  // If time = 1, integration will be 101ms
  // If time = 2, integration will be 402ms
  // If time = 3, use manual start / stop to perform your own integration
  unsigned char time = 1;
  //Serial.println("Set timing...");
  light.setTiming(gain,time,ms);
  // To start taking measurements, power up the sensor:
  Serial.println("Powerup...");
  light.setPowerUp();
  //
  //
  turnAround() ;

}


//********************************************
```



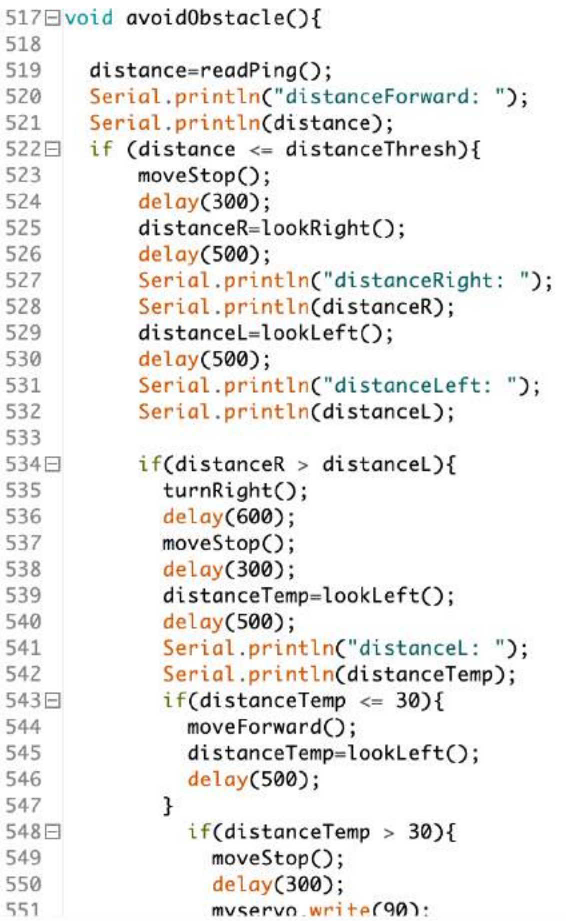
```
void loop() {
  start: //label
  error=0;

  elapsedWriteTime = millis()-startWriteTiming;

  if (elapsedWriteTime > (writeTimingSeconds*1000))
  {
    readSensors();
    writeThingSpeak();
    startWriteTiming = millis();
  }

  if (error==1) //Resend if transmission is not completed
  {
    Serial.println(" <<<< ERROR >>>>");
    delay (2000);
    goto start; //go to label "start"
  }

  //
  avoidObstacle();

  if(luxnum >= lightThresh){

    moveStop();
    delay(10000);
    turnAround();
    }
    else{}

}
//********************************************
```



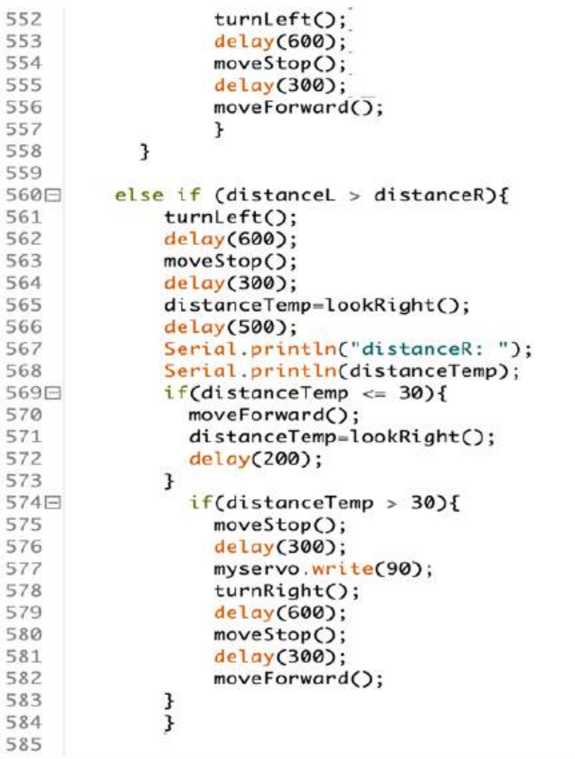
```
void turnAround(){
int LUX1, LUX2, LUX3, LUX4 ;
  //First Turn
  motorA.forward();
  motorB.forward();
  delay(1);
  turnRight();
  delay(600);
  moveStop();
  delay(2000);
  getLight();
  LUX1= luxnum;
  Serial.println("LUX1: ");
  Serial.println(LUX1);

  //second turn
  turnRight();
  delay(600);
  moveStop();
  delay(2000);
  getLight();
  LUX2=luxnum;
  Serial.println("LUX2: ");
  Serial.println(LUX2);

  //third turn
  turnRight();
  delay(600);
  moveStop();
  delay(2000);
  getLight();
  LUX3=luxnum;
  Serial.println("LUX3: ");
  Serial.println(LUX3);

```



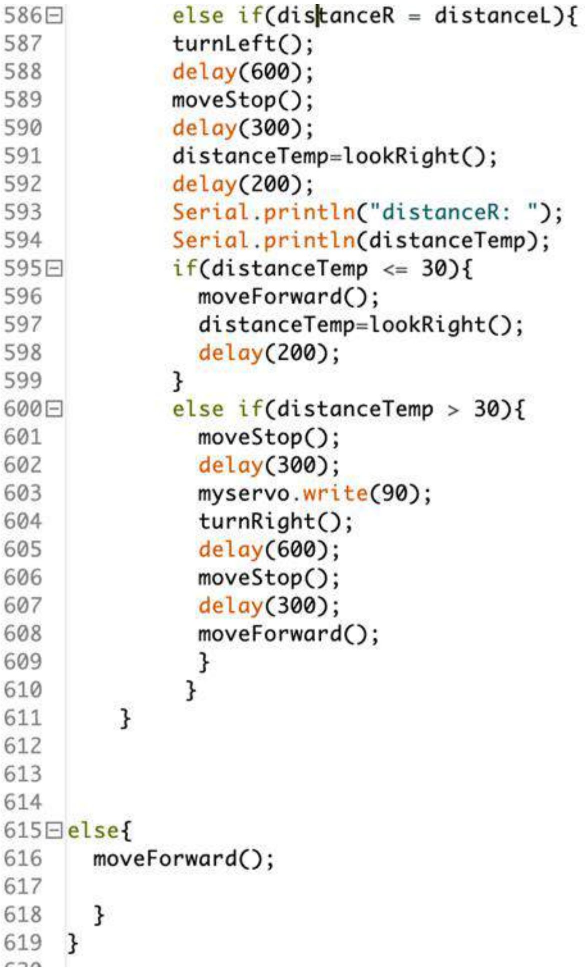
```
//fourth Turn
turnRight();
delay(600);
moveStop();
delay(2000);
getLight();
LUX4=luxnum;
Serial.println("LUX4: ");
Serial.println(LUX4);
//
int maxlight = max(max(LUX1,LUX2), max(LUX3,LUX4));
Serial.print("Maxlight is ");
Serial.println(maxlight);
  //

  if (maxlight==LUX1){
  turnRight();
  delay(600);
  moveStop();
  delay(100);
  moveForward();
  }
//
  else if (maxlight==LUX2){
  turnRight();
  delay(1200);
  moveStop();
  delay(100);
  moveForward();
  }
  //
  else if (maxlight==LUX3){
  turnRight();
  delay(1800);
  moveStop();
```



```
void avoidObstacle(){

  distance=readPing();
  Serial.println("distanceForward: ");
  Serial.println(distance);
  if (distance <= distanceThresh){
      moveStop();
      delay(300);
      distanceR=lookRight();
      delay(500);
      Serial.println("distanceRight: ");
      Serial.println(distanceR);
      distanceL=lookLeft();
      delay(500);
      Serial.println("distanceLeft: ");
      Serial.println(distanceL);

      if(distanceR > distanceL){
        turnRight();
        delay(600);
        moveStop();
        delay(300);
        distanceTemp=lookLeft();
        delay(500);
        Serial.println("distanceL: ");
        Serial.println(distanceTemp);
        if(distanceTemp <= 30){
          moveForward();
          distanceTemp=lookLeft();
          delay(500);
        }
          if(distanceTemp > 30){
            moveStop();
            delay(300);
            myservo.write(90);
```

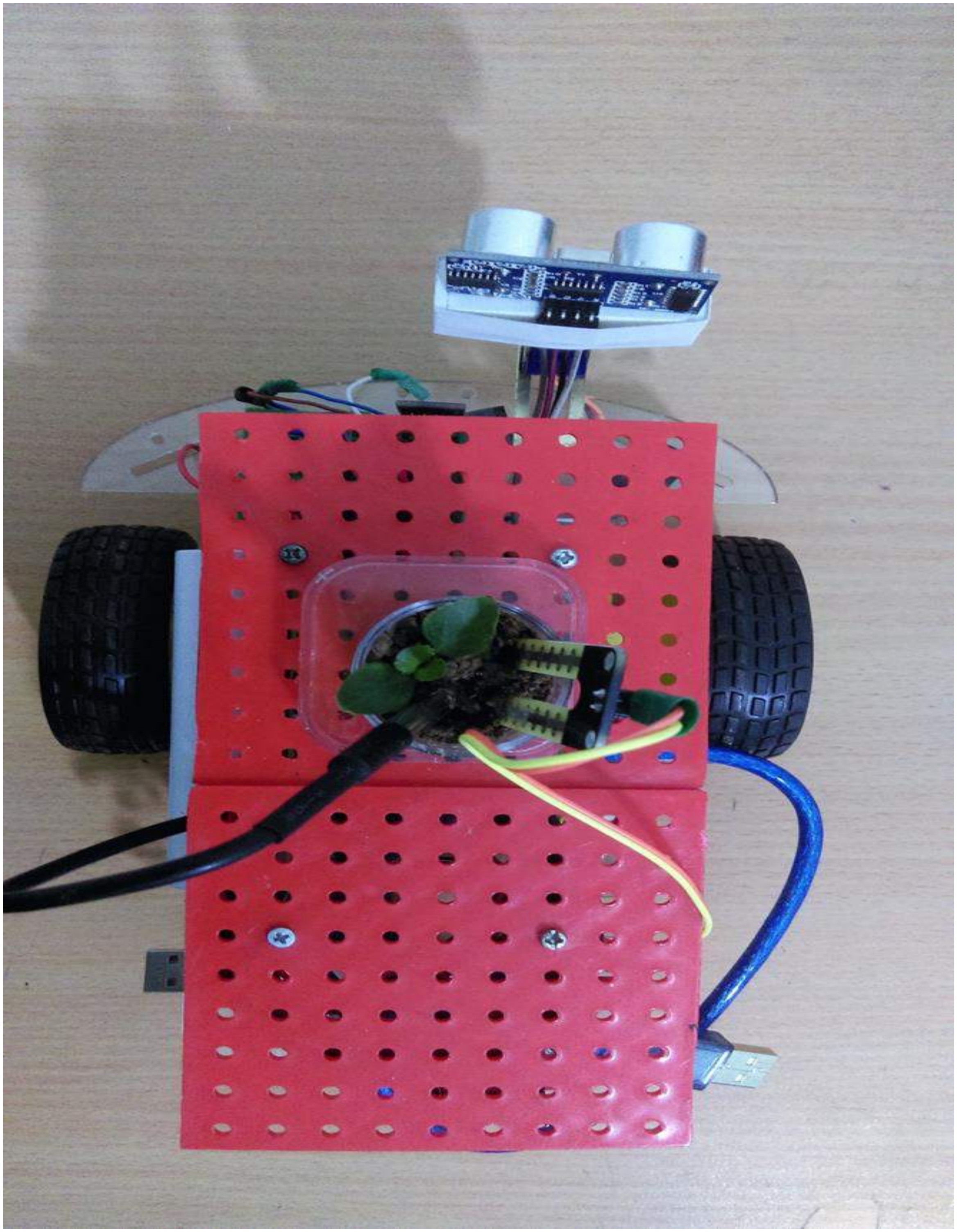


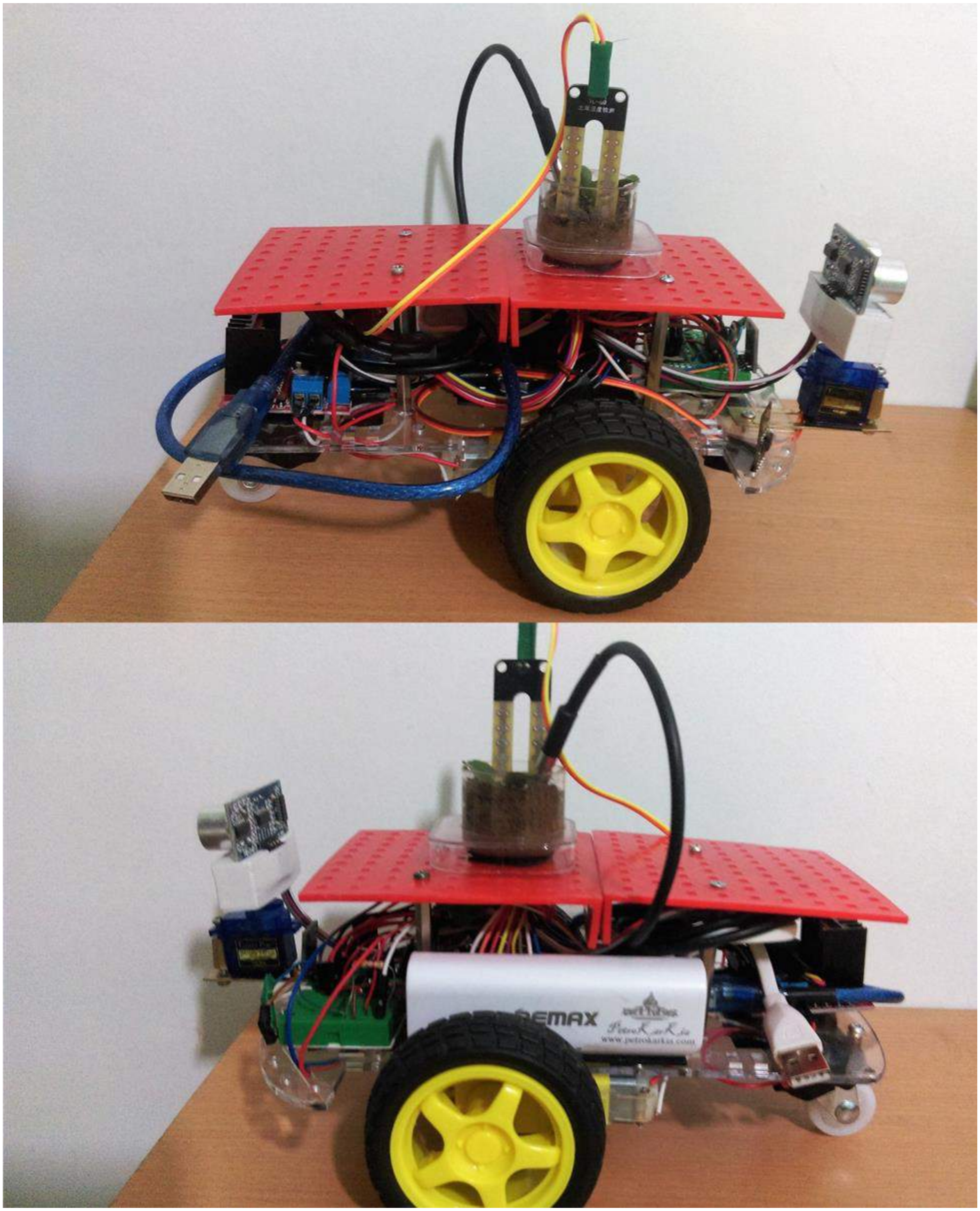
```
            turnLeft();
            delay(600);
            moveStop();
            delay(300);
            moveForward();
            }
      }

    else if (distanceL > distanceR){
        turnLeft();
        delay(600);
        moveStop();
        delay(300);
        distanceTemp=lookRight();
        delay(500);
        Serial.println("distanceR: ");
        Serial.println(distanceTemp);
        if(distanceTemp <= 30){
          moveForward();
          distanceTemp=lookRight();
          delay(200);
        }
          if(distanceTemp > 30){
            moveStop();
            delay(300);
            myservo.write(90);
            turnRight();
            delay(600);
            moveStop();
            delay(300);
            moveForward();
        }
        }

```



```
        else if(distanceR = distanceL){
        turnLeft();
        delay(600);
        moveStop();
        delay(300);
        distanceTemp=lookRight();
        delay(200);
        Serial.println("distanceR: ");
        Serial.println(distanceTemp);
        if(distanceTemp <= 30){
          moveForward();
          distanceTemp=lookRight();
          delay(200);
        }
        else if(distanceTemp > 30){
          moveStop();
          delay(300);
          myservo.write(90);
          turnRight();
          delay(600);
          moveStop();
          delay(300);
          moveForward();
          }
         }
    }



else{
  moveForward();

  }
}
```

Real Picture of Project:




REMAX
www.petrokarkas.com