IOT Based Smart Window with Remote Controlling System

Riajul Islam

ID: 2014-2-60-087

Md. Mahamudul Hasan

ID: 2014-3-60-043

Supervised By:

Dr. Md. Nawab Yousuf Ali

Associate Professor

Computer Science and Engineering Department

East West University

A project submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering

Department of Computer Science and Engineering East West University

Dhaka-1212, Bangladesh

September, 2019

Declaration

We, hereby, declare that the work presented in this project titled "**IOT Based Smart Window with Remote Controlling System**" is the outcome of the academic knowledge performed by us under the supervision of **Dr. Md. Nawab Yousuf Ali**, Associate Professor, Department of Computer Science and engineering, East West University. We also declare that no part of this project has been or is being submitted elsewhere for the award of any degree or diploma. This project has been completed by September 2019 in full supervision of **Dr. Md. Nawab Yousuf Ali**.

. . . . . .

Project Supervisor

Md. Nawab Yousuf Ali

Associate Professor

Computer Science and Engineering Department

. . . . . .

Author

Riajul Islam

ID: 2014-2-60-087

Computer Science and Engineering Department

. . . . . .

Author

Md. Mahamudul Hasan

ID: 2014-3-60-043

Computer Science and Engineering Department

Acknowledgement

First of all, we would like to express my deepest gratitude to the almighty for His blessings on us.

This dissertation concludes our Bachelor of Science degree in Computer Science & Engineering at East West University. Completion of this work has been both interesting and challenging to us. We would like to extend our gratitude to all the people who helped and supported us during this project work. We wish to express our deepest appreciation to our supervisor, **Dr. Md. Nawab Yousuf Ali**, Associate Professor, Department of CSE, EWU for his guidance and encouragement that gave us the opportunity to progress and broaden our knowledge. His encouragements, visionaries and thoughtful comments and suggestions, unforgettable support at every stage of our B.Sc. study were simply appreciating and essential. His ability to muddle us enough to finally answer our own question correctly is something valuable what we have learned and we would try to emulate, if ever we get the opportunity.

We would also like to thank Department of Computer Science and Engineering for providing all kinds of support which have made it possible to complete our document successfully.

Last but not the least, we would like to thank our parents for their unending support, encouragement and prayers.

There are numerous other people too who have shown me their constant support and friendship in various ways, directly or indirectly related to our academic life. We will remember them in our heart and hope to find a more appropriate place to acknowledge them in the future.

Riajul Islam

September, 2019

Md. Mahamudul Hasan

September, 2019

Abstract

Window is well-known and part and parcel of a room or a building. Window is really important for air ventilation. Window cannot be only considered as air ventilation system as privacy and security is also related to window. Human are being very busy by these days and also being depended on machine hugely. That is why it is very important to keep a window responsive as human need.

In our project our main goal was to build a window that will be responsive with the environmental status and also can be monitored and controlled by owner from any place of the world via smart phone using internet. We have also strongly concentrated on window security.

This is designed to be a socio impact-based project via **NodeMCU** or Open Source IOT Platform based system. Basically, we have used **NodeMCU** as a microprocessor so that it can control, program and perform our whole objective. Since we have used some smart sensors such as **Temperature Humidity** sensor, **PIR motion detector** sensor, **Dust** Sensor and **Servo** motors, we have achieved three goals here. The first goal we have achieved is, we have made an automated window system. It will prevent our precious and valuable things to get any harm by any intruder or by any object. The second goal we have achieved is, we can get live updates of the window status via internet. And the last goal we have achieved is, we can control the window remotely. The window can be updated to more advanced level but we think that would be a matter of huge financial investment.

Table of Contents

Chapter 1

Introduction

Now it is era of technology where human is being dependent on machine. A window is a part of human accommodation which is sometimes becomes a way of harm for human. Window mainly used for air ventilation but it is also related to security. Through the window dust and intruder can come inside the room. As human can't stay at home all the time, they need go outside of the room for many reasons the become a cause of tension for them. Sometimes we go out from our home for a long time of period and we forget to close the window.

Figure 1: Motion Alert on Smart Phone

Figure 2: Dust Coming through Window

Keeping these circumstances in mind, we started to think about the possible solutions for the window. We wanted to build a smart window which will be responsive to the environmental situations and also be responsive if any intruder or object tries to enter through the window. We also wanted to provide the live status of the window to the owner and provide manual control to the user to open/close the window from any distant location of the world remotely.

Before starting our project, we searched over the internet about what kind of smart window projects are already being used.[9] We found that most of the smart window project was built with simple 1 or 2 sensors. Which means those smart windows will be only used for simple 1 or 2 purpose. Those projects are available in various combination of sensors and various names. But no one project is perfectly match the human needs.

As those types of projects are not fulfilling the needs of human for their window, we started develop a system that could fulfill human desires of smart window system. We have used the most important sensors for the automated window and also have live status monitor and user control system for the window.

1.1 Objectives

- To build a reliable automatic window system.
- To get alarmed for any intruder or any object presence.
- To get status of room Temperature and Humidity.
- To get dust density of the incoming air.
- To get all the live status of the sensor over the internet from anywhere of the world.
- To control the window via smart phone remotely.

1.2 Purpose

There are other projects have been done on window. But those projects are not dealing with all real situations of environment or those projects are based on Bluetooth module which can be only controlled from inside the room or via the connection from same router. That's why we have built a project with most needed sensors to keep pace with human needs and also created the window automated window system which's data can be monitored and the window also can be controlled over the internet from any place of the world.

1.3 Background

Over the time the pattern, size and design of window has been changed. For this reason, there are some primitive types of security feature are available right now. Or which projects are coming now like MIT are all designed inside the glass. Some little projects are available on the internet which are not that much fruitful for the mostly used and common window system. They calling the primitive projects which are now old dated technology. Our project will be related to commonly used window system and also will smart in real sense.

Chapter 2

System Introduction

There are three parts of our project. First part is the Mechanical part, second part is the Data Read Write on the Website part and last one is the Phone Application part. Combining these parts, we are calling the window system as a smart window system.

2.1 Mechanical Part

In this part we used software and hardware to build the automatic window which will response to the environmental situations using sensors and will response accordingly. This system will automatically operate the window although the other 2 parts get any failure or go down.

Hardware

In hardware part we have joined some individual equipment together by their build system. Every individual equipment has their own functionality, pin diagram and working principle.

List of Component's:

- **NodeMCU with ESP8266**
- **lot PL2303 USB To RS232 TTL Converter Adapter Module**
- **Passive Infrared (PIR Motion) Sensor**
- **DHT11 (Temperature and Humidity) Sensor**
- **GP2Y1010AU0F Optical Dust Sensor**
- **Servo Motor**
- **Jumper Wires**
- **150-ohm Resistor**
- **220uF Capacitor**
- **Bread Boards**
- **USB Cable**

Node MCU with ESP8266:

NodeMCU is an open source IoT platform.[1] It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson and SPIFFS.[6] **Figure 3** shows the practical view of NodeMCU module and using this figure we can have an idea of the module.

Figure 1: Node MCU with ESP8266

Pin Diagram of NodeMCU with ESP8266:

General-purpose input/output (GPIO) is a pin on an IC (Integrated Circuit). It can be either input pin or output pin, whose behavior can be controlled at the run time.

NodeMCU Development kit provides access to these GPIOs of ESP8266. The only thing to take care is that NodeMCU Dev kit pins are numbered differently than internal GPIO notations of ESP8266 as shown in below figure and table. For example, the D0 pin on the NodeMCU Dev kit is mapped to the internal GPIO pin 16 of ESP8266.[10]

Below table gives NodeMCU Dev Kit IO pins and ESP8266 internal GPIO pins mapping:

Pin Names on NodeMCU Development Kit	ESP8266 Internal GPIO Pin number
D0	GPIO16
D1	GPIO5
D2	GPIO4
D3	GPIO0
D4	GPIO2
D5	GPIO14
D6	GPIO12
D7	GPIO13
D8	GPIO15
D9/RX	GPIO3
D10/TX	GPIO1
D11/SD2	GPIO9
D12/SD3	GPIO10

The GPIO's (1, 3, 9, 10) are mostly not used for GPIO purpose on Dev Kit. ESP8266 is a system on a chip (SoC) design with components like the processor chip. The processor has around 16 GPIO lines, some of which are used internally to interface with other components of the SoC, like flash memory.

Since several lines are used internally within the ESP8266 SoC, we have about 11 GPIO pins remaining for GPIO purpose.

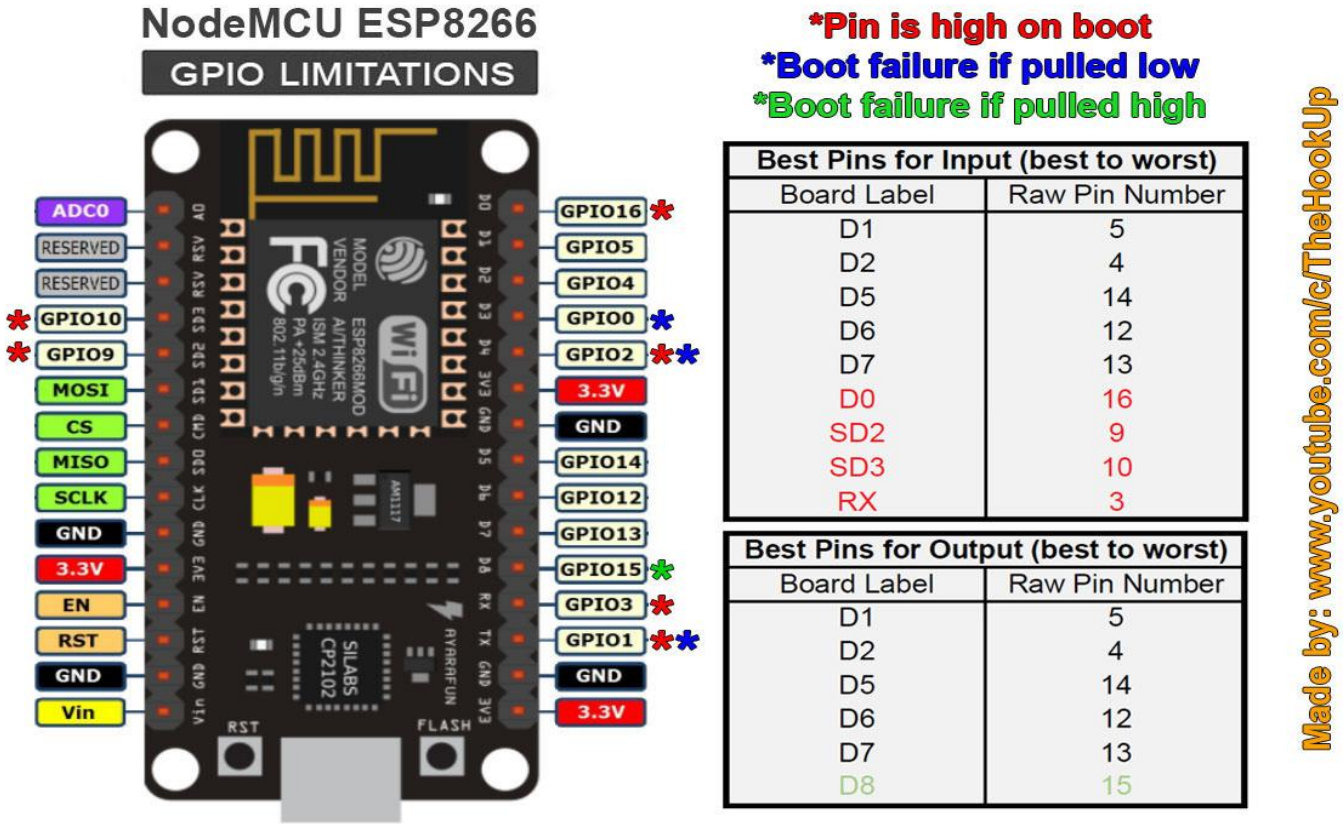

Figure 2: Node MCU with ESP8266 Pin Diagram

Now again 2 pins out of 11 are generally reserved for RX and TX in order to communicate with a host PC from which compiled object code is downloaded. Hence finally, this leaves just 9 general purpose I/O pins i.e. D0 to D8. As shown in above **Figure 4** of NodeMCU Dev Kit. We can see RX, TX, SD2, SD3 pins are not mostly used as GPIOs since they are used for other internal process. But we can try with SD3 (D12) pin which mostly like to respond for GPIO/PWM/interrupt like functions.

Note that D0/GPIO16 pin can be only used as GPIO read/write, no special functions are supported on it.

lot PL2303 USB To RS232 TTL Converter Adapter Module:

It's a small USB to TTL serial tool, using the PL2303 chip. As shown in below **Figure 5** you can use it to connect some serial device to your PC via USB port.

Specification:

Module Type	: Adapter Board
Size	: 4.6 x 1.5 x 1.1cm
Operation Level	: Digital 5V
Power Supply	: External 5V

Figure 3: PL2303 USB To RS232 TTL Converter Adapter Module

Passive Infrared (PIR Motion) Sensor:

The PIR sensor itself has two slots in it, each slot is made of a special material that is sensitive to IR. The lens used here is not really doing much and so we see that the two slots can 'see' out past some distance (basically the sensitivity of the sensor). When the sensor is idle, both slots detect the same amount of IR, the ambient amount radiated from the room or walls or outdoors.[3]

Figure 4: Passive Infrared (PIR Motion) Sensor

When a warm body like a human or animal passes by, it first intercepts one half of the PIR sensor, which causes a positive differential change between the two halves. As shown in above **Figure 6** the PIR sensor when the warm body leaves the sensing area, the reverse happens, whereby the sensor generates a negative differential change. These change pulses are what is detected.

DHT11 (Temperature and Humidity) Sensor:

This DHT11 Temperature and Humidity Sensor features a calibrated digital signal output with the temperature and humidity sensor capability. It is integrated with a high-performance 8-bit microcontroller. Its technology ensures the high reliability and excellent long-term stability.[2] This sensor includes a resistive element and a sensor for wet NTC temperature measuring devices. It has excellent quality, fast response, anti-interference ability and high performance.

Each DHT11 sensors features extremely accurate calibration of humidity calibration chamber. The calibration coefficients stored in the OTP program memory, internal sensors detect signals in the process, we should call these calibration coefficients. The single-wire serial interface system is integrated to become quick and easy. Small size, low power, signal transmission distance up to 20 meters, enabling a variety of applications and even the most demanding ones. As shown in below **Figure 7** the product is 4-pin single row pin package. Convenient connection, special packages can be provided according to users need.

Specification:

Supply Voltage: +5 V

Temperature range :0-50 °C error of ± 2 °C

Humidity :20-90% RH ± 5% RH error

Interface: Digital

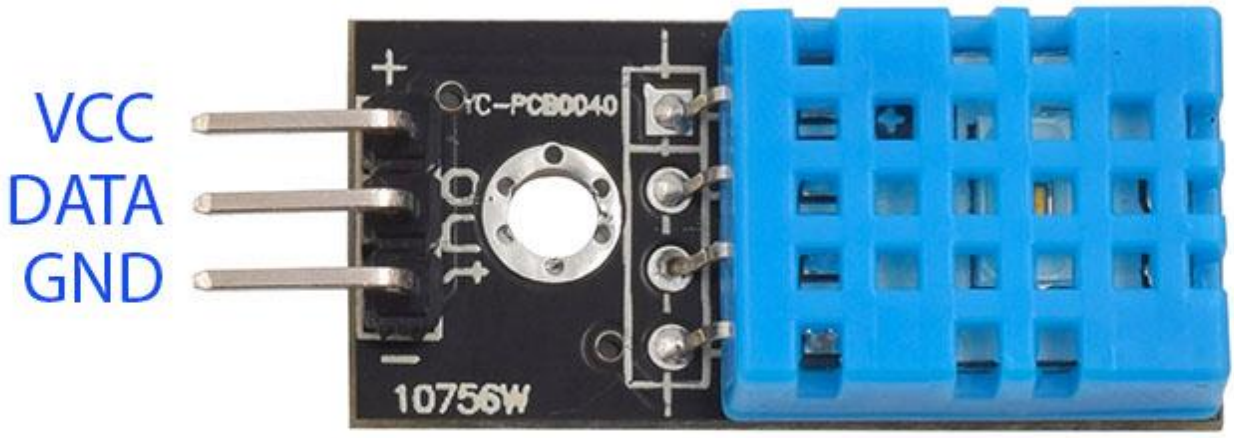

Figure 5: DHT11 (Temperature and Humidity) Sensor

GP2Y1010AU0F Optical Dust Sensor:

Sharp's GP2Y1010AU0F is an optical air quality sensor, designed to sense dust particles. An infrared emitting diode and a phototransistor are diagonally arranged into this device, to allow it to detect the reflected light of dust in air. As shown in below **Figure 8** it is especially effective in detecting very fine particles like cigarette smoke, and is commonly used in air purifier systems.[4]

The sensor has a very low current consumption (20mA max, 11mA typical), and can be powered with up to 7VDC. The output of the sensor is an analog voltage proportional to the measured dust density, with a sensitivity of 0.5V/0.1mg/m^3.

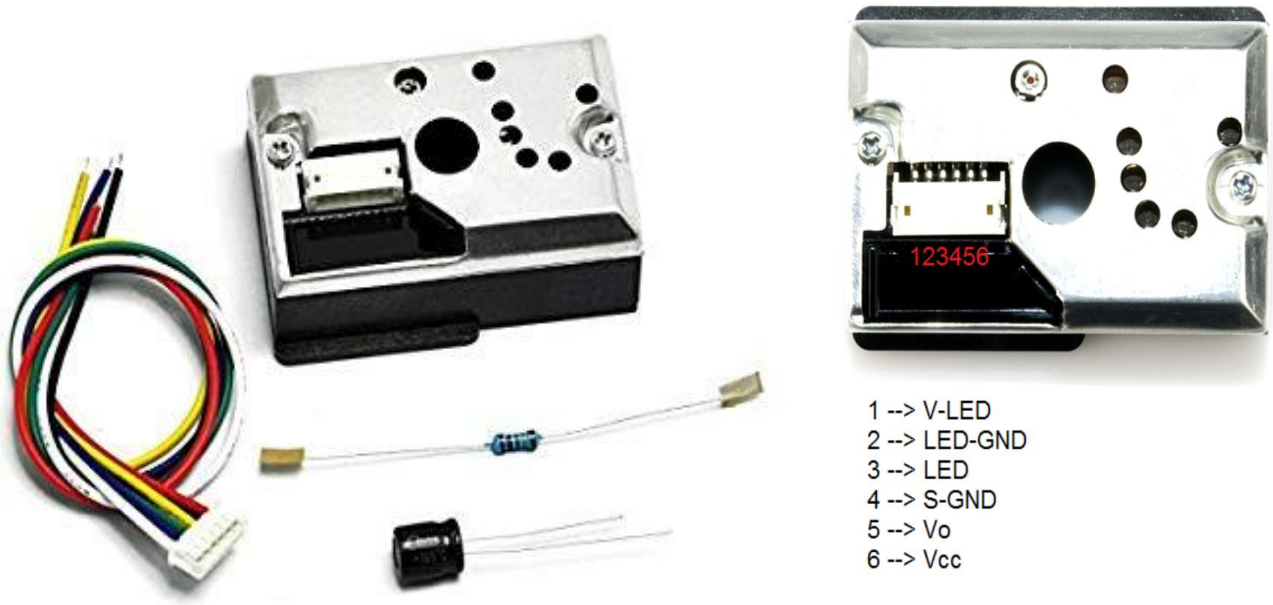

Figure 6: GP2Y1010AU0F Optical Dust Sensor

Servo Motor:

A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.[5]

As shown in below **Figure 9** Servomotors are not a specific class of motor, although the term servomotor is often used to refer to a motor suitable for use in a closed-loop control system.

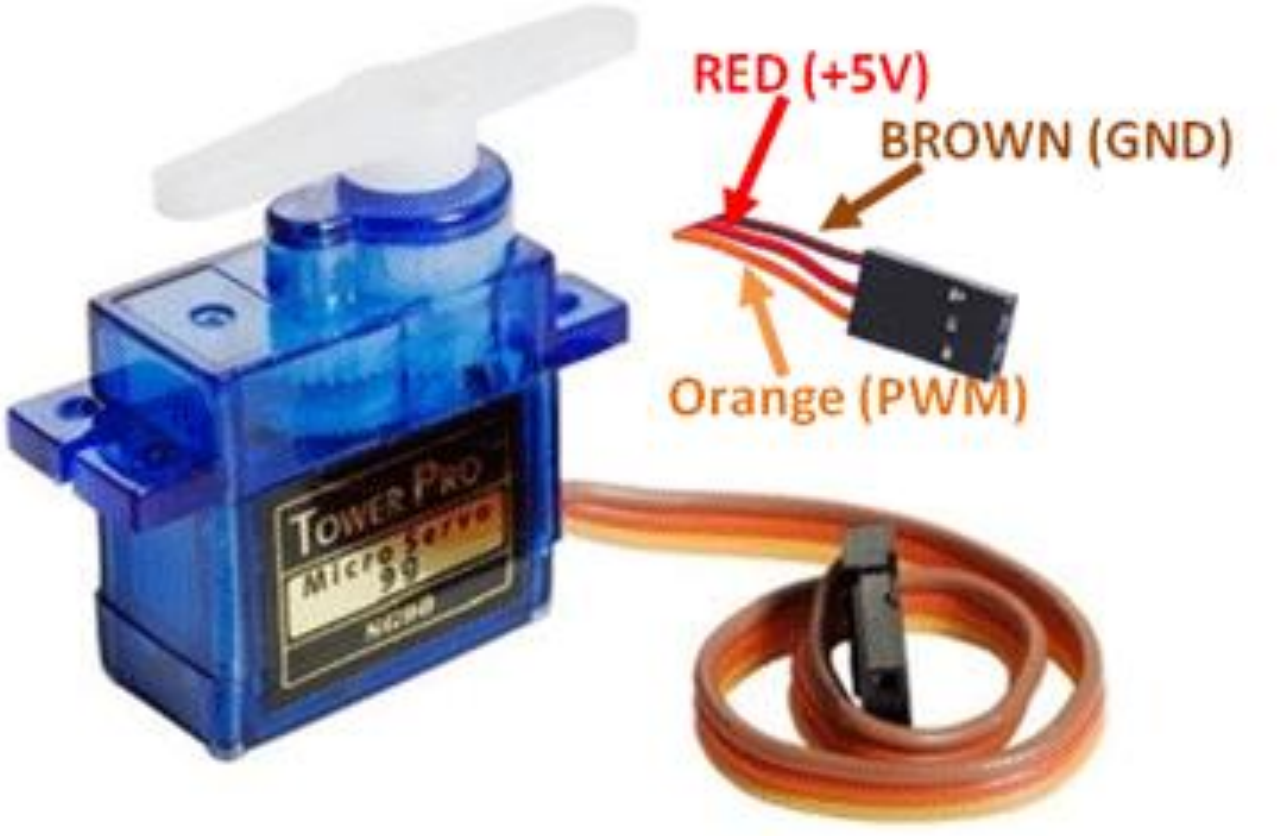

Figure 7: Servo Motor

Servomotors are used in applications such as robotics, CNC machinery or automated manufacturing.

Jumper Wires:

A jump wire is an electrical wire, or group of them in a cable, with a connector or pin at each end, which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering. **Figure 10** shows the practical structure of Jumper Wires.

Figure 8: Jumper Wires

150-ohm Resistor:

Metal Oxide Resistors 1 Watt 150 ohm. **Figure 11** shows the practical view of 150 ohm Resistor.

Figure 9: 150-ohm Resistor

220uF Capacitor:

The Electrolytic Capacitors have polarity. Meaning they have a positive and negative pin. The pin which is long is the positive pin and the pin which is short is the negative pin. You can also identify the polarity using the negative strip on the capacitor label. As shown in the **Figure 12** below the negative pin will be directly under the negative symbol.

Figure 10: 220uF Capacitor

Bread Board:

A breadboard is a construction base for prototyping of electronics. Originally the word referred to a literal bread board, a polished piece of wood used for slicing bread. In the 1970s the solderless breadboard became available and nowadays the term "breadboard" is commonly used to refer to these. **Figure 13** is an example that why it is called Bread Board.

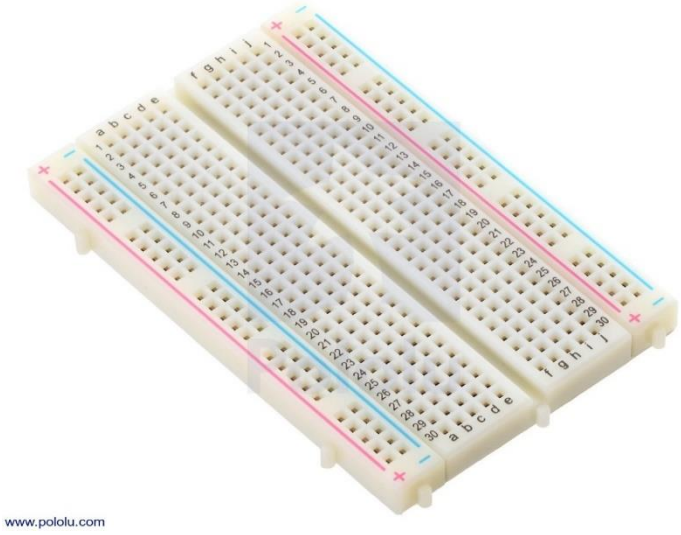

Figure 11: Bread Board

Connection Details:

DHT11:

VCC – 5V

GND – GND

DATA – D3 (Pin 12, GPIO0)

Passive Infrared (PIR Motion) Sensor:

VCC – 5V

GND – GND

OUT – D0 (Pin 15, GPIO16)

GP2Y1010AU0F Optical Dust Sensor:

Pin 1 – Is connected to Capacitor through 150-ohm Resistor who's another hand is connected to 5V

Pin 2 – GND along Capacitor's GND

Pin 3 – D5 (Pin 8, GPIO14)

Pin 4 – GND

Pin 5 – A0 (Pin 16)

Pin 6 – 5V

Servo Motor 1 (Motion):

VCC – 5V

GND – GND

IN – D1 (Pin 14, GPIO5)

Servo Motor 2 (Motion):

VCC – 5V

GND – GND

IN – D1 (Pin 13, GPIO4)

Servo Motor (Dust):

VCC – 5V

GND – GND

IN – D6 (Pin 7, GPIO12)

Circuit Diagram:

Figure 14 is the circuit diagram of the hardware part. Here all the modules are connected to the NodeMCU.

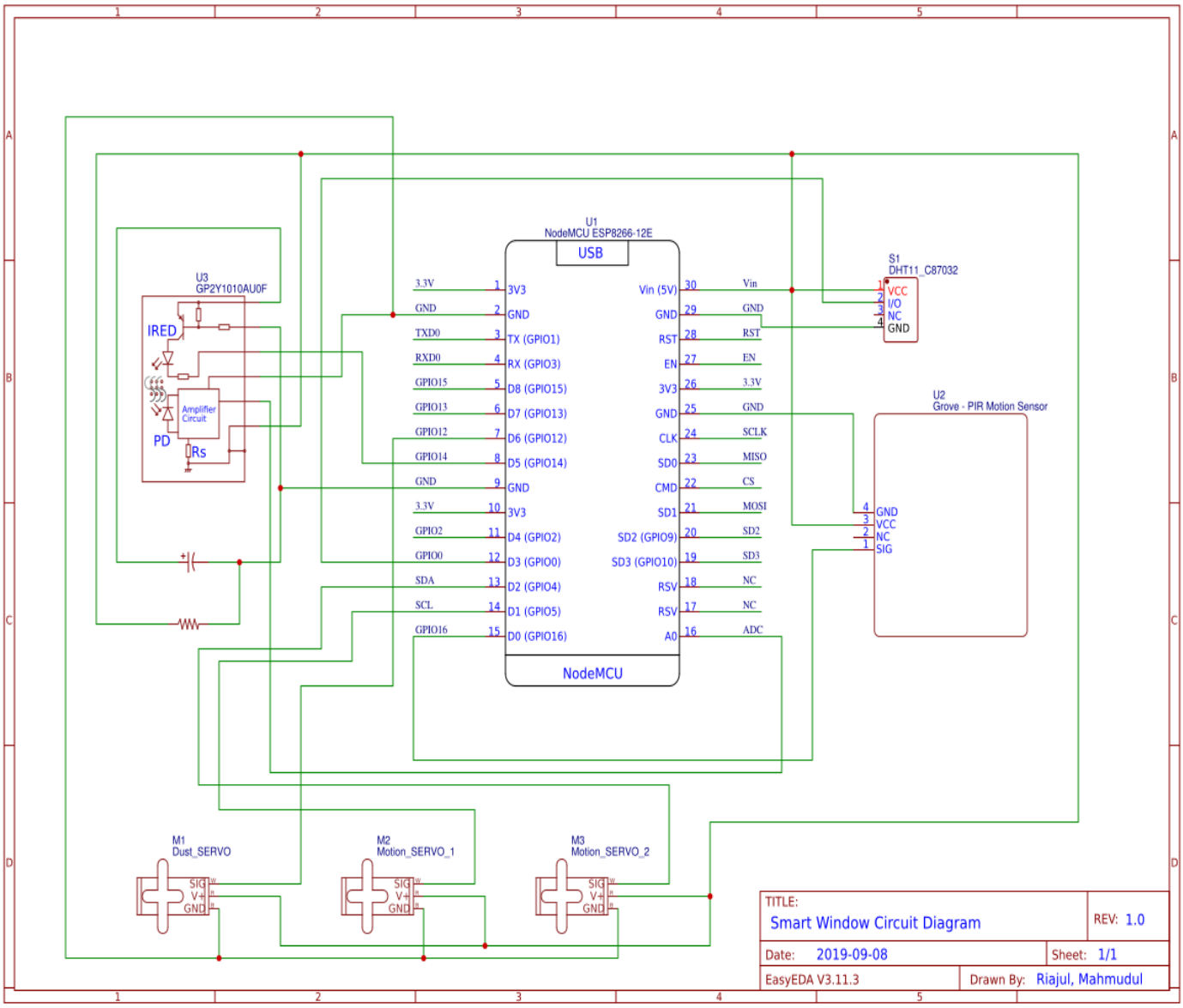

Figure 12: Circuit Diagram

Software

In software part we have used **Arduino IDE** software to write the code, reading data in **Serial Monitor** and to upload the code into the **Node MCU** module. **Figure 15** is an example of the **Arduino IDE.**[7]

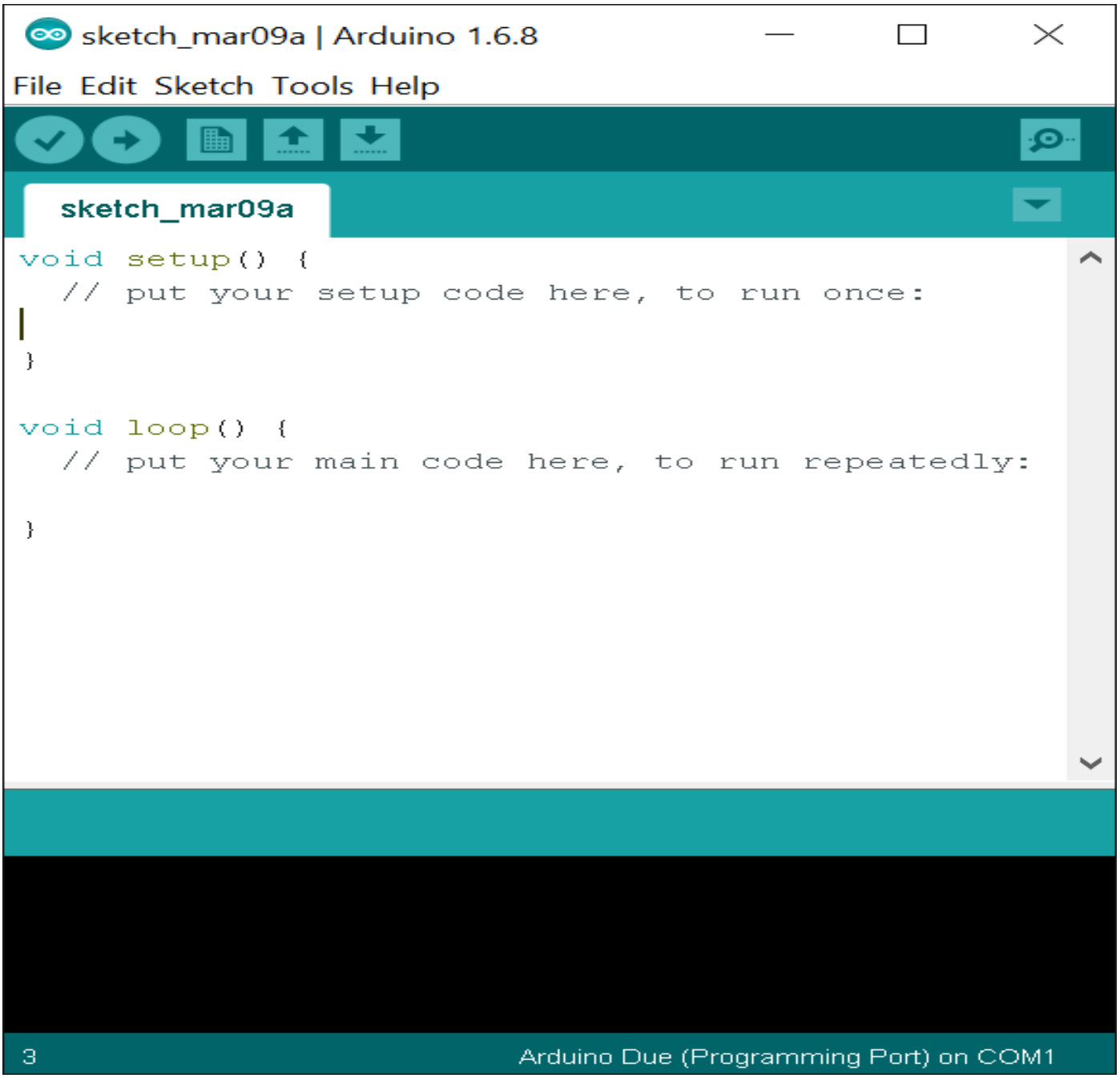

Figure 13: Arduino IDE

We have used some of the libraries in our code to run the modules. The libraries are given below:

- ThingSpeak.h
- DHT.h
- Servo.h
- ESP8266WiFi.h

We have used **NodeMCU 0.9 (ESP-12 Module)** Board and used upload speed **115200. Figure 16** is the screenshot of the settings used in Software part.

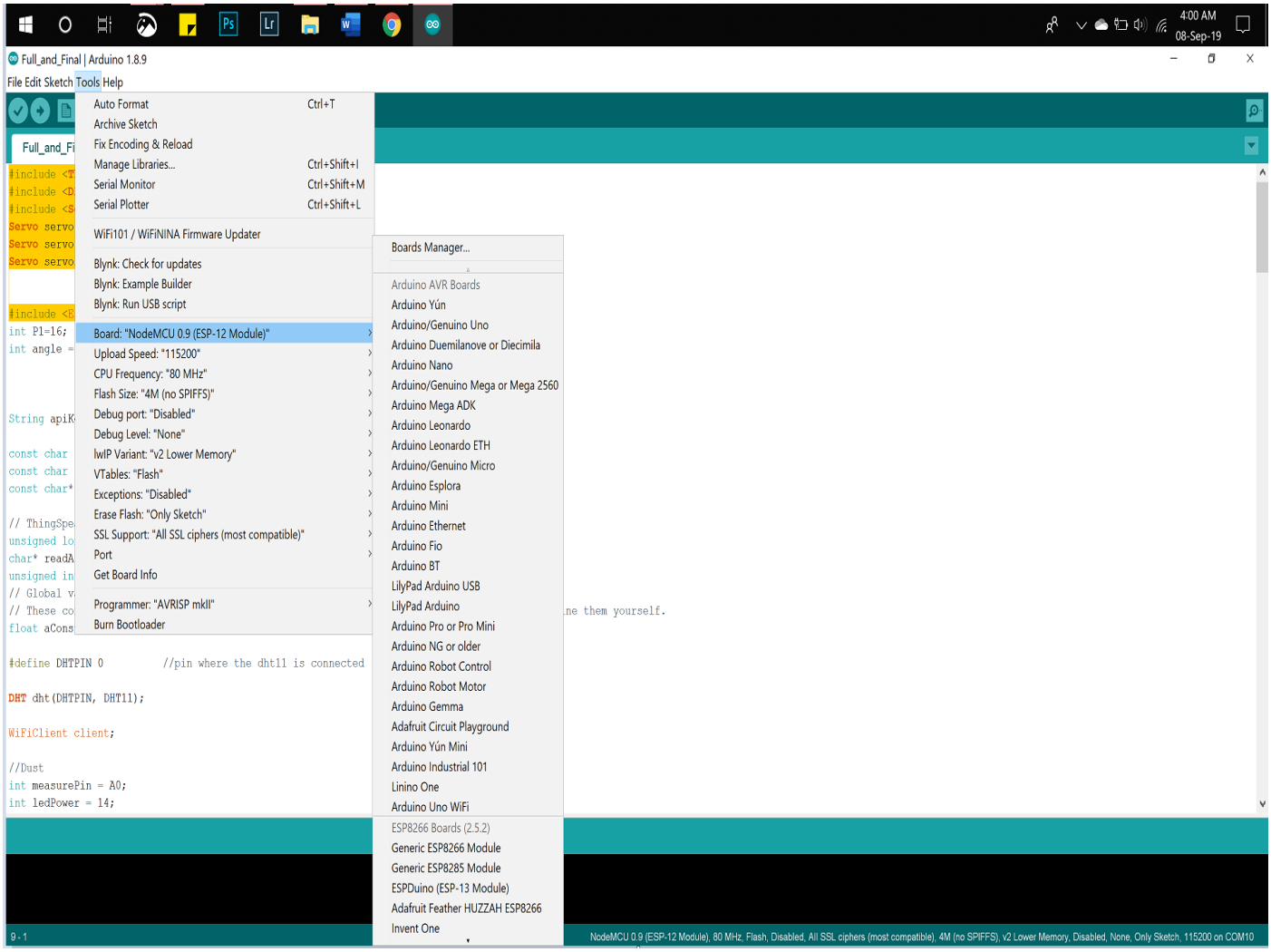

Figure 14: Board Type

Below we have provided the screenshot of Serial Monitor:

Figure 17 is the screenshot of serial monitor. Here we can see all the data coming from sensors and also data coming from **Thingspeak** server.

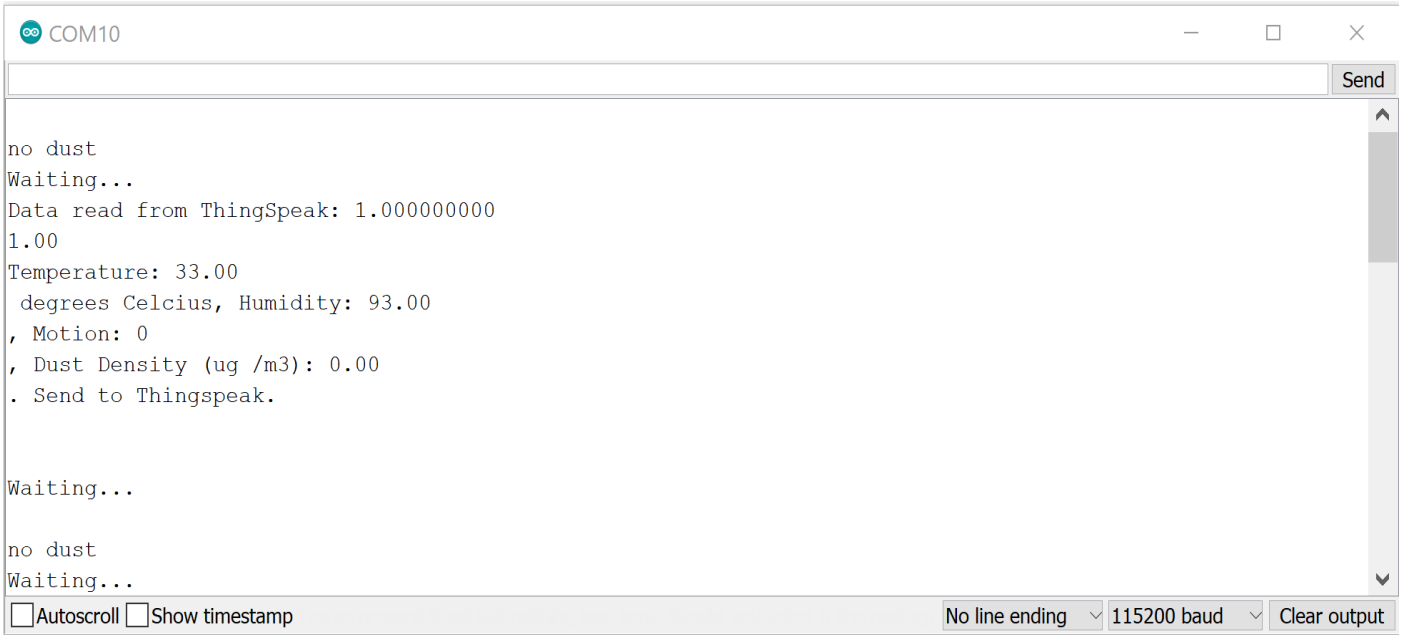

Figure 15: Serial Monitor

2.3 Data Read Write on the Website part:

We have used a website called **Thingspeak** to read from and write on the ESP8266 Server. Firstly, we have created a channel in **Thinspeak**. This website provides read and write API Keys for the use of read and write.[8] We have created 6 fields called Temperature, Humidity, Motion, Window Control, Dust and Window on/off. From this website owner will be able to monitor the sensors live status and can give command from smart phone to this website.

Channel Creation:

Figure 18 is the screenshot of Channel creation menu. From here Channel ID can be used and Fields can be created as per need.

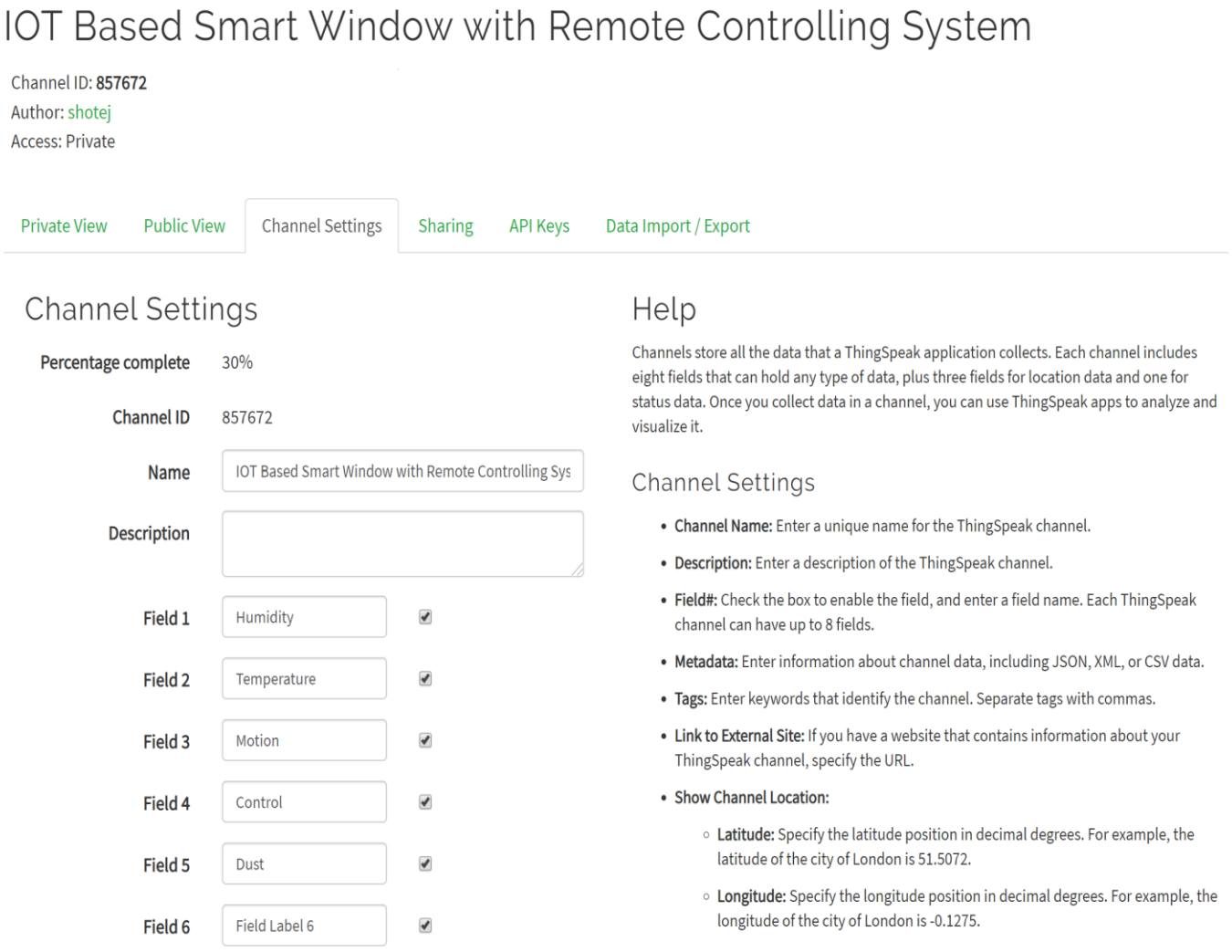

Figure 16: Channel Settings

API Keys:

Figure 19 is the screenshot of **API Keys** menu. From here we can get API Keys which will be used to send and receive data.

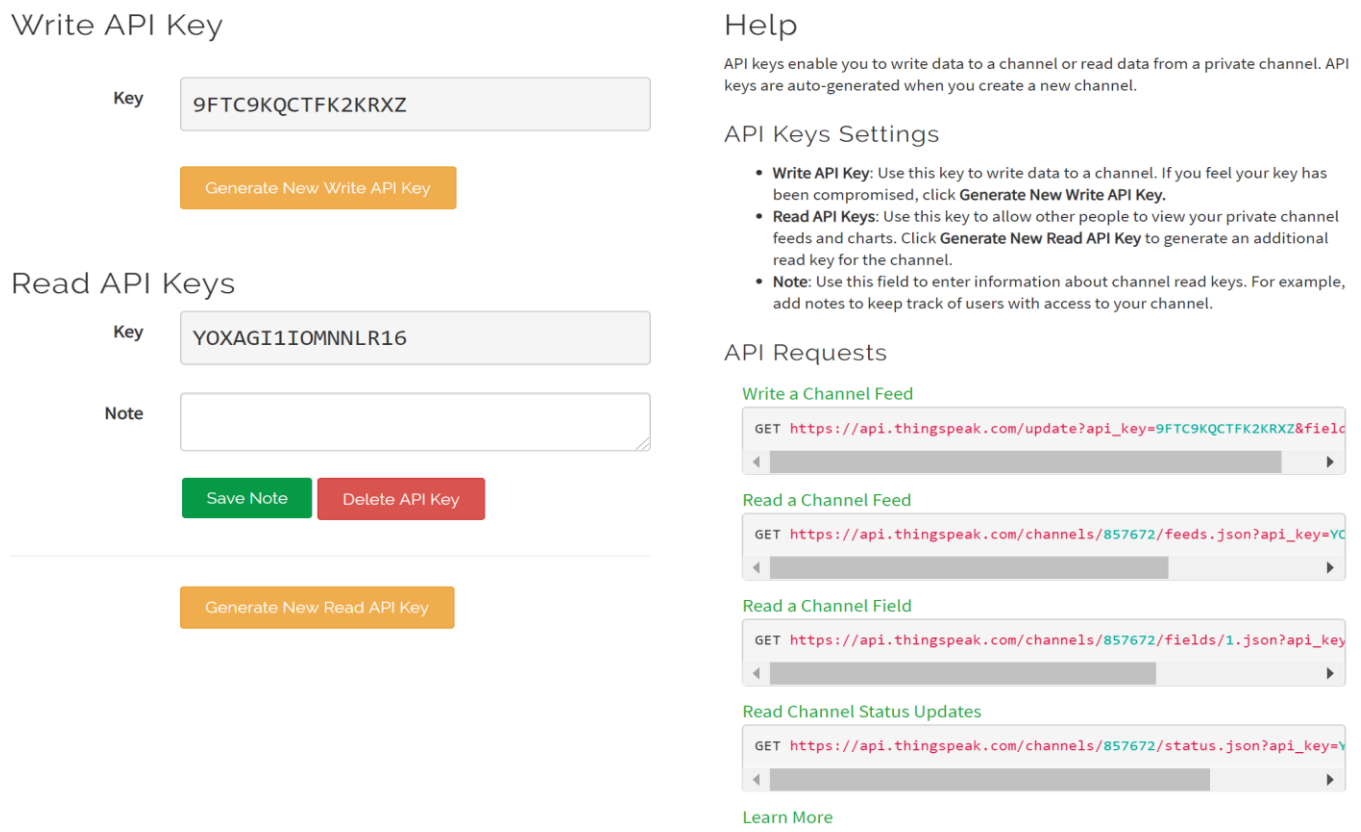

Figure 17: API Keys

Data Fields:

Below respectively **Figure 20,21 & 22** are the screenshot of **Data** getting from sensors and from Smart Phone.

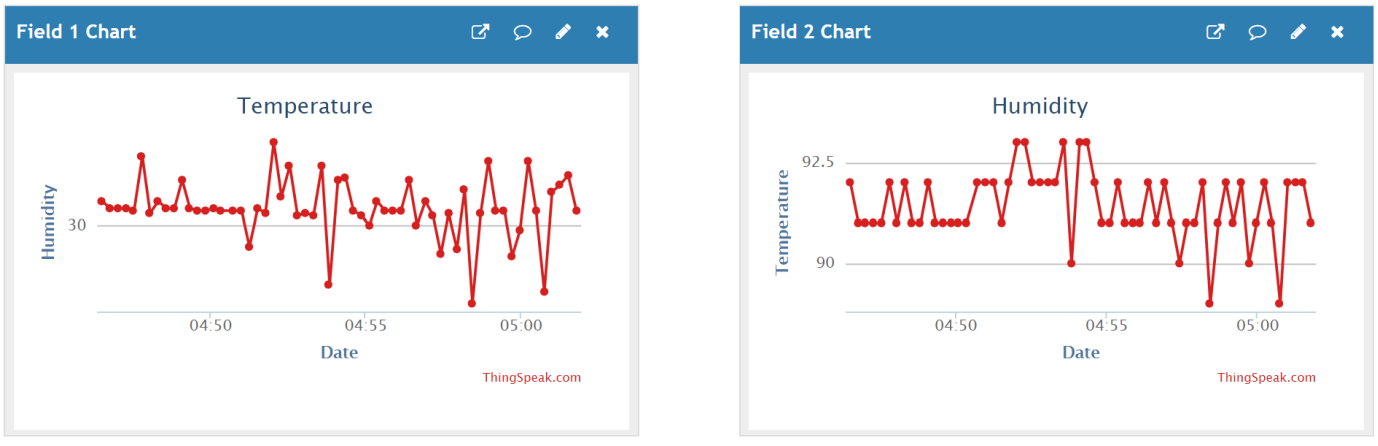

Figure 18: Temperature Humidity Fields

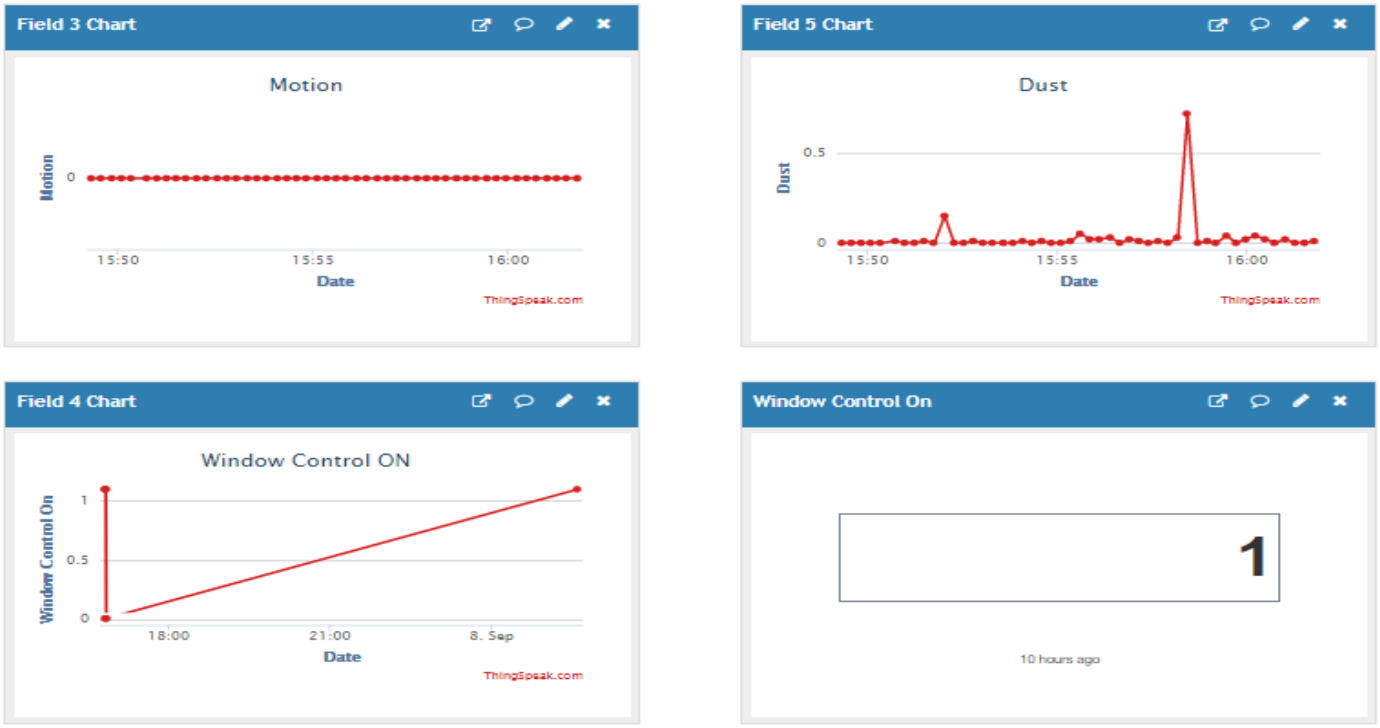

Figure 19: Motion, Dust & Window Control On

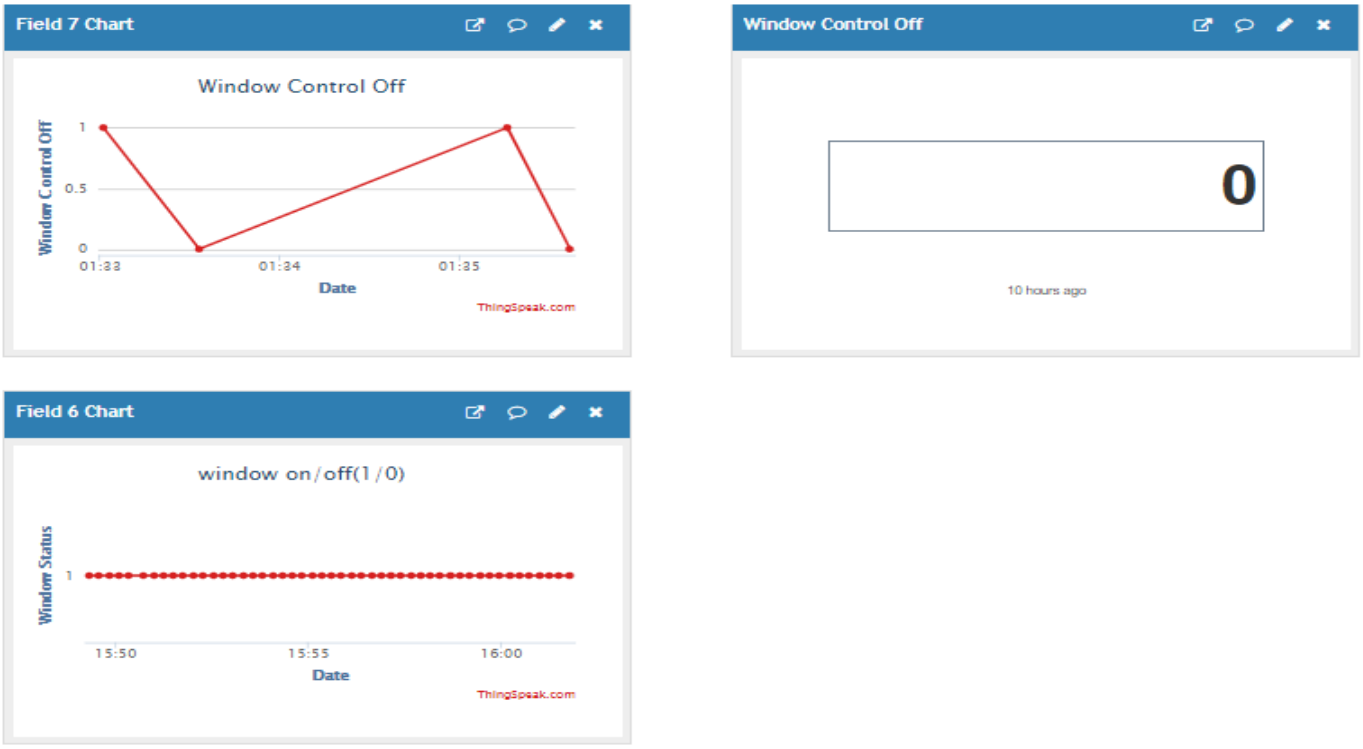

Figure 20: Window Control Off and Window Status

2.4 Phone Application part:

In this part we have used a phone application named **Virtuino** to communicate with the **Window Module** through the **Thingspeak** website. We have created a user interface in the app according to our need and made connection with the website using read and write API keys.

User UI:

Below **Figure 23 & 24** are the screenshot of Home Screen & Sever Setting Menu.

Figure 23: Home Screen

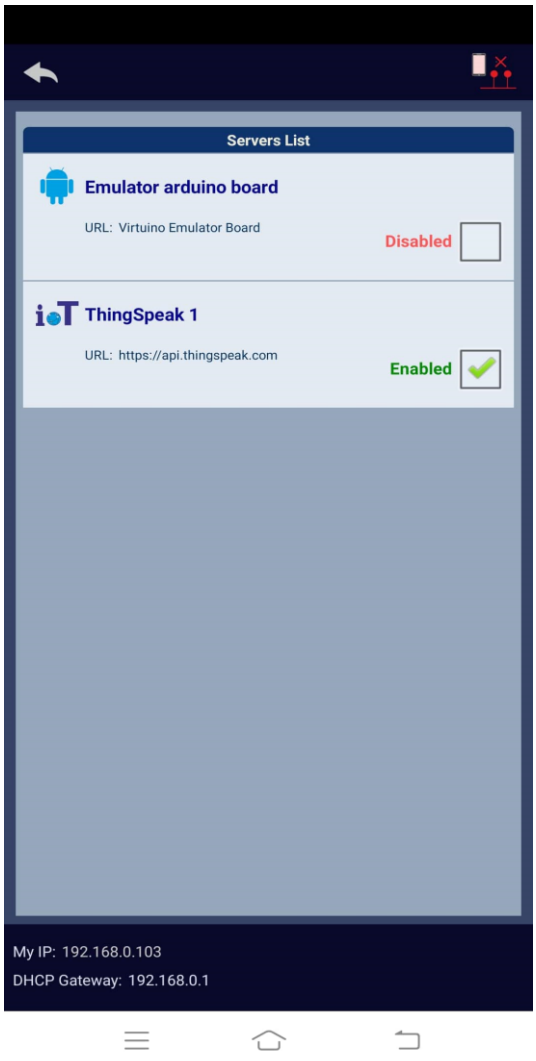

Figure 24: Server Connection Settings

Below respectively **Figure 25 & 26** are the screenshot of Server Setting Read and Write API Keys & Setting Command Buttons.

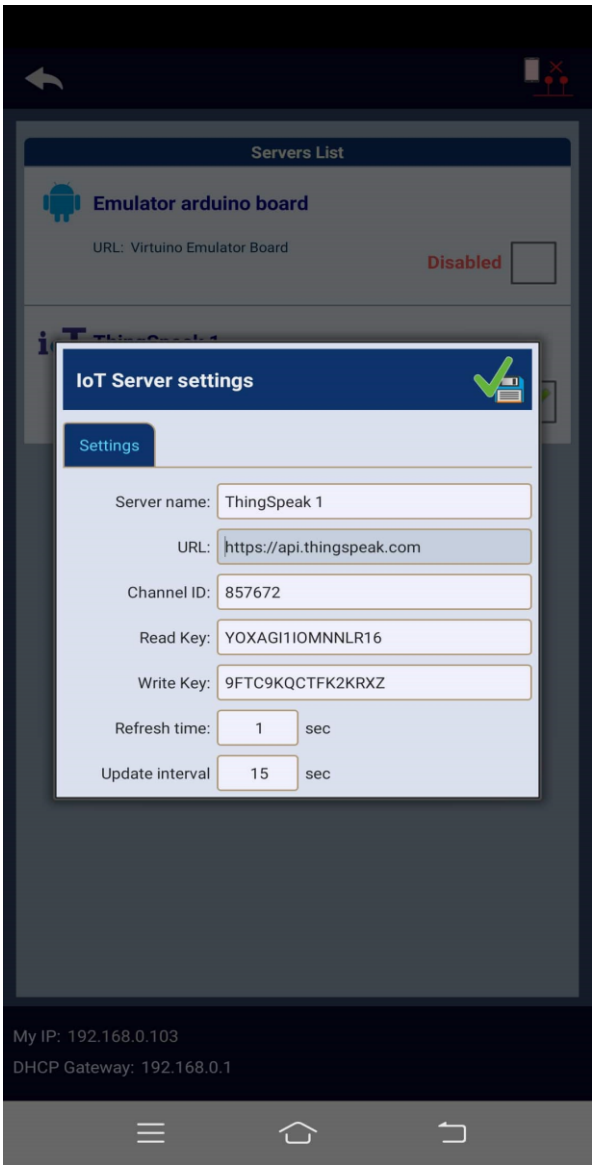

Figure 25: Setting read and write API Keys

Figure 26: Setting Command Button

2.5 Working Principle:

In hardware part there are 3 sensor modules working simultaneously. The Sensor have individual dedicated pins to take voltage, ground for release voltage and Data Signal pin for providing data to NodeMCU module. NodeMCU module collects the data gained from the sensors and continuously send those data to the Thingspeak Web Server's Channel through the ESP8266 Wi-Fi module. Thingspeak displays those real time data in using user defined five different fields. Data will be updated continuously. Simultaneously manual command data is coming towards Thingspeak from smart phone application called Virtuino configured by user using Read and Write API keys.

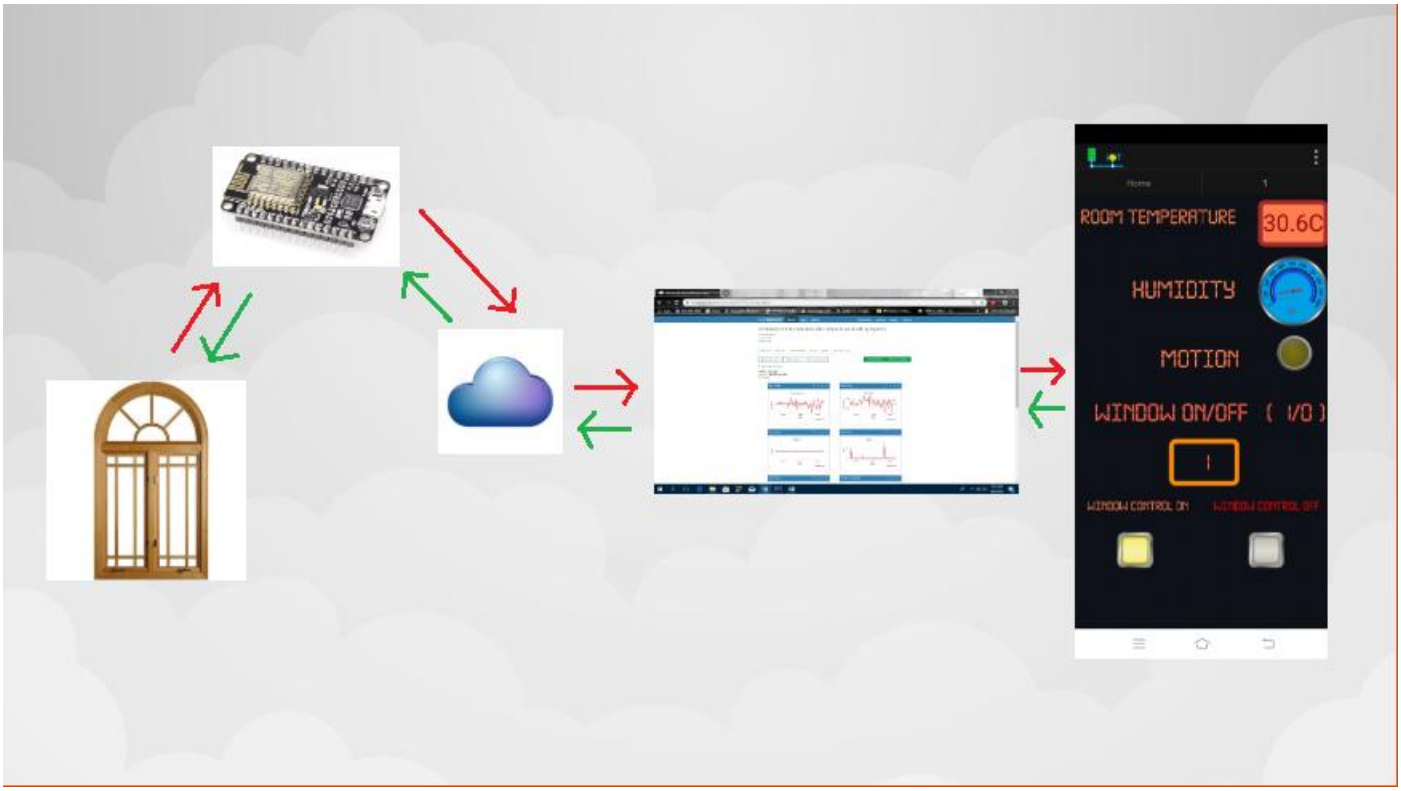

Figure 21: Working Principle

This application will display the real time data of the sensors onto the user UI. There is an alarm system in the application for any kind of motion detection. There is a window on/off push button on the application home screen to send data to field number 4 and 5 of the Thingspeak Channel. **Figure 27** shows that the Thingspeak will continuously collect data and will be showing into particular field. From the data the window system will response according to the command. If the window system gets value 1 from user controlled open window command, then it will open the window. And if the window system gets value 1 from user controlled Close window command, then it will open the window.

Chapter 3

Code for Module:

```
#include <ThingSpeak.h>
#include <DHT.h>
#include <Servo.h>
Servo servo;
Servo servo1;
Servo servo2;

#include <ESP8266WiFi.h>
int P1=16;
int angle = 0;

String apiKey = "9FTC9KQCTFK2KRXZ";     //  Enter your Write API key from ThingSpeak

const char *ssid =  "SazzaD_wifi";     // replace with your wifi ssid and wpa2 key
const char *pass =  "forgotpassword007";
const char* server = "api.thingspeak.com";

// ThingSpeak information
unsigned long channelID = 857672;
char* readAPIKey = "YOXAGI1IOMNNLR16";
unsigned int dataFieldOne = 4; // Field to write control data 1
unsigned int dataFieldTwo = 7;

// Global variables
// These constants are device specific.  You need to get them from the manufacturer or determine them yourself.
float aConst = 2.25E-02;
float bConst = 2.25E-02;

#define DHTPIN 0         //pin where the dht11 is connected

DHT dht(DHTPIN, DHT11);
```

```
WiFiClient client;

//Dust
int measurePin = A0;
int ledPower = 14;

unsigned int samplingTime = 280;
unsigned int deltaTime = 40;
unsigned int sleepTime = 9680;

float voMeasured = 0;
float calcVoltage = 0;
float dustDensity = 0;
void setup()
{
    Serial.begin(115200);
    delay(10);
    dht.begin();
    pinMode(P1,INPUT);

    servo.attach(5);
    servo.write(0);
    delay(100);

    servo1.attach(4);
    servo1.write(0);
    delay(100);

    servo2.attach(12);
    servo2.write(0);
    delay(100);

    Serial.println("Connecting to ");
    Serial.println(ssid);


    WiFi.begin(ssid, pass);

   while (WiFi.status() != WL_CONNECTED)
```

```
    {
        delay(500);
        Serial.print(".");
    }
     Serial.println("");
     Serial.println("WiFi connected");

     ThingSpeak.begin( client );

     // Read the constants at startup.
  aConst = readTSData( channelID, dataFieldOne );
  bConst = readTSData( channelID, dataFieldTwo );
  //Serial.println(aConst);
  delay(1000);

  //Dust
  pinMode(ledPower,OUTPUT);
}

void loop()
{
  //Dust
    Serial.println (" ");
    digitalWrite(ledPower,LOW);
  delayMicroseconds(samplingTime);

  voMeasured = analogRead(A0);

  delayMicroseconds(deltaTime);
  digitalWrite(ledPower,HIGH);
  delayMicroseconds(sleepTime);

  calcVoltage = voMeasured*(5.0/1024);
  dustDensity = 0.17*calcVoltage-0.1;

  if ( dustDensity < 0)
  {
    dustDensity = 0.00;
  }
  //Serial.println("Dust Density:");
```

```
// Serial.println(dustDensity);
  if (dustDensity<0.10)
 {
 // digitalWrite(13,HIGH);
 Serial.println("no dust");

  servo2.write(180);              //command to rotate the servo to the specified angle
  delay(1000);
 }
  else if(dustDensity>0.10)
   {
servo2.write(0);
delay(1000);
    }

   float h = dht.readHumidity();
   float t = dht.readTemperature();
   int P1 = digitalRead(16);
   int massage =1;

   if(P1==HIGH)
    {
       for(angle = 0; angle < 180; angle += 5)    // command to move from 0 degrees to 180 degrees
          {
           servo.write(angle);//command to rotate the servo to the specified angle
           servo1.write(angle);
           delay(0);
          }
    }

    delay(1000);
   /* else if(P1==LOW)
    {
      servo.write(0);
      delay(1000);
     }*/

Serial.println("Waiting...");  //reading data from thing speak
 aConst = readTSData( channelID, dataFieldOne );
 Serial.println(aConst);
```

```
 if(aConst==HIGH)
 {
  for(angle; angle>=1; angle-=5)     // command to move from 180 degrees to 0 degrees
    {
     servo.write(angle);//command to rotate the servo to the specified angle
     servo1.write(angle);
     delay(5);
    }
 }

 Serial.println("Waiting...");  //reading data from thing speak
bConst = readTSData( channelID, dataFieldTwo );
Serial.println(bConst);

 if(bConst==HIGH && angle<1)
 {
  for(angle ; angle < 180; angle += 5)     // command to move from 0 degrees to 180 degrees
    {
     servo.write(angle);//command to rotate the servo to the specified angle
     servo1.write(angle);
     delay(5);
    }
  }

if(t>38 && P1==LOW)
    {
     if(angle>10)
     {
      for(angle; angle>=1; angle-=5)     // command to move from 180 degrees to 0 degrees
           {
            servo.write(angle);//command to rotate the servo to the specified angle
           servo1.write(angle);
                   delay(5);
            }
      }
     }


 else if(t<25)
  {
```

```
  if(angle<1)
  {
     for(angle ; angle < 180; angle += 5)    // command to move from 0 degrees to 180 degrees
      {
       servo.write(angle);//command to rotate the servo to the specified angle
       servo1.write(angle);
       delay(0);
      }
   }
}


     if (isnan(h) || isnan(t))
       {
          Serial.println("Failed to read from DHT sensor!");
           return;
       }

             if (client.connect(server,80))   //   "184.106.153.149" or api.thingspeak.com
          {
                String postStr = apiKey;
                postStr +="&field1=";
                postStr += String(t);
                postStr +="&field2=";
                postStr += String(h);
                postStr +="&field3=";
                postStr += String(P1);
                postStr +="&field5=";
                postStr += String(dustDensity);
                if(angle<1)
                {
                postStr +="&field6=";
                postStr += String(massage);
                }
                else if(angle>10)
                {
                massage = 0;
                postStr +="&field6=";
                postStr += String(massage);
                 }
                postStr += "\r\n\r\n";
```

```
                client.print("POST /update HTTP/1.1\n");
                client.print("Host: api.thingspeak.com\n");
                client.print("Connection: close\n");
                client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
                client.print("Content-Type: application/x-www-form-urlencoded\n");
                client.print("Content-Length: ");
                client.print(postStr.length());
                client.print("\n\n");
                client.print(postStr);

                Serial.print("Temperature: ");
                Serial.print(t);
                Serial.println (" ");
                Serial.print(" degrees Celcius, Humidity: ");
                Serial.print(h);
                Serial.println (" ");
                Serial.print(", Motion: ");
                Serial.print(P1);
                Serial.println (" ");
                Serial.print(", Dust Density (ug /m3): ");
                Serial.print(dustDensity);
                Serial.println (" ");
                Serial.println(". Send to Thingspeak.");
            }
    client.stop();
    Serial.println (" ");
    Serial.println (" ");

    Serial.println("Waiting...");

  // thingspeak needs minimum 15 sec delay between updates
  delay(1000);

  //Dust

/*Serial.print ("Dust Density (ug /m3) = ");

Serial.print (FineDust);
```

```
Serial.println (" ");*/

delay (3000);

}

float readTSData( long TSChannel, unsigned int TSField ) {

  float data =  ThingSpeak.readFloatField( TSChannel, TSField, readAPIKey );
  Serial.println( "Data read from ThingSpeak: " + String( data, 9 ) );
  return data;

}
```

Chapter 4

Conclusion and Future Work:

The speed of the developing world is fast and to keep pace with it human is being very busy. That's why some little jobs are being done by machine so that we can give our full concentration on other important works. The window is nothing but a tension for us. It can be a huge pain in the head while we are outside of the house unable return soon. Our Smart Window is able to control the window automatically and it is also possible to monitor and control the window by the owner from any distant location of the world. That's why we think our project will a strong solution for this window problem as we have built the project while keeping in mind that what actually people are need.

In future we will analyze more about the window problem and if necessary, we will add more sensor modules. We will build our own Read and Write Website for data monitoring. The smart phone application which we are using for user, it is a free application and for this reason the command data from user is not that fast responding. We will build our own smart phone application for this purpose ahead. If our project starts to manufactured by any company and people from different location buy and install our window system then it will possible to introduce **Machine Learning** system into our project. As, we will get data from different locations it will be possible to generate data about which area is not safe or motion sensor gets intruder alert more. We can say which area's probability of rain is by analyzing the humidity values. We will keep correcting bugs and any lacking of our project. We will highly appreciate if any researcher found any correction in our project and contact us for upgradation. We will be very glad if we can help others on this subject. Due to time and financial limitation, we were unable to add more features to enrich our project. So, in future we will add more useful features in our IOT Based Smart Window with Remote Controlling System.

Reference

1. Vowstar. "NodeMCU Devkit". *Github*. NodeMCU Team. Retrieved 2 April 2015.
2. 101, c. (2019). *DHT11 Sensor Pinout, Features, Equivalents & Datasheet*. [online] Components101.com. Available at: https://components101.com/dht11-temperature-sensor [Accessed 29 Sep. 2019].
3. Adafruit Learning System. (2019). *PIR Motion Sensor*. [online] Available at: https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor/how-pirs-work [Accessed 29 Sep. 2019].
4. impulse, o. (2019). *GP2Y1010AU0F Optical Dust Sensor | Open ImpulseOpen Impulse*. [online] Openimpulse.com. Available at: https://www.openimpulse.com/blog/products-page/product-category/sharp-gp2y1010au0f-optical-dust-sensor/ [Accessed 29 Sep. 2019].
5. Kankaria, R. (2019). [online] Engineering.eckovation.com. Available at: https://engineering.eckovation.com/servo-motor-types-working-principle-explained/ [Accessed 29 Sep. 2019].
6. "gpio - NodeMCU Documentation". *nodemcu.readthedocs.io*. Retrieved 2018-11-11.
7. Celtare, A. (2019). *Arduino IDE*. [online] En.wikipedia.org. Available at: https://en.wikipedia.org/wiki/Arduino_IDE [Accessed 8 Sep. 2019].
8. Slinger, N. (2019). *ThingSpeak*. [online] En.wikipedia.org. Available at: https://en.wikipedia.org/wiki/ThingSpeak [Accessed 8 Sep. 2019].
9. Song, M. (2019). *A Study on Business Types of IoT-based Smarthome : Based on the Theory of Platform Typology*. [online] http://www.koreascience.or.kr. Available at: http://www.koreascience.or.kr/article/JAKO201614139534980.page [Accessed 8 Sep. 2019].
10. Wings, E. (2019). *NodeMCU GPIO with Arduino IDE | NodeMCU*. [online] Electronicwings.com. Available at: https://www.electronicwings.com/nodemcu/nodemcu-gpio-with-arduino-ide [Accessed 29 Sep. 2019].

.