Report for the Project Wireless sensor Network using esp8266

Abstract

The idea behind the project is to create a wireless sensor/actuator network which gathers the sensor data at a particular time interval and uploads it on the web application giving an access to user from anywhere around the world. The projects demonstrates end to end implementation of iot application from collecting data from transducer to converting it into mqtt protocol and sending it through internet to a cloud application which is hosted on AWS instance and from there pushing it to web application through mqtt client. The project further demonstrate 2 way communication from device to web application and vice versa.

Goal

The major goal of the project is to allow temperature reading and light intensity of the environment to monitor over internet on web application and allows the web application to control the actuator which further turns on/off the fan to maintain the temperature.

Design Choices

Hardware –

Module 1- NodeMcu with Light dependent Resistor

This modules contains following hardware –

1] NodeMcu (esp826612E)

2] Light dependent Resistor (LDR)

3] Resistor

Node MCU – This is a Wifi System on chip integrated with a Tensilica Xtensa LX106 core it supports Arduino ide for programming which is one of the main reason for using this board for prototyping , Along nodemcu has controller and wifi chip together on board reduces the complexity of interfacing a different controller and wifi chip together. Also it has inbuilt Analog to digital converter which was required for this project to interface an analog sensor.

LDR – It is variable resistor which is used in the voltage divider arrangement, The resistance of the LDR changes with respect to the light. The resistance of the LDR is maximum in the absence of the light and minimum at the presence of the light so based on that the voltage across the resistor connected in voltage divider changes and that voltage is sensed by the nodemcu.

Module 2 –

1] NodeMcu

2] Lm35

3] Capacitor

4] Relay Module

5] DC Fan

Node MCU – This is same wifi SOc chip which is used in the module 1 it has same function here although it is connected to temperature sensor lm 35 in order to sense voltage change with respect to temperature

Lm35 – This is an active transducer which operates from -3 v to 35v it has 3 pins which gives a voltage change with respect to temperature. The typical lm 35 sensor gives output variations of 10mv per degree. A Capacitor is connected across the output pin so as to filter out any noise coming from the sensor.

Relay Module - Relay module is used to actuate the fan, The reason behind using relay module is fan operates on 5v or above and our esp8266 module operates on 3.3v and the sourcing current of the esp gpio is not sufficient to power up the fan, So we have used relay module , The Relay module comes with a transistor as a switch to trigger the relay and relay further makes contact for fan to power up.

Software –

At Arduino –

The Arduino is a platform which allows rapid prototyping and the major reason behind it is that it has vast collection of software libraries, for this project I have used 2 software libraries as follows –

ESP8266wifi – This library is used to enable wifi client on Node mcu, In Short it allows esp module to operate as a wifi client.

PubSubClient – This library allows programmer to send and receive the data through tcp in MQTT protocol, MQTT protocol is used in order to send real time data. Mqtt is a light weight and fast protocol for this purpose we have used PubSubClient library which allows to communicate with the server.

At Backend side –

I have used Amazon EC2 instance for installing my server and hosting my web application. The data coming from the device is received at the MQTT broker as we know that mqtt is a light weight protocol it has to be handled by the broker for this I have used Mosquitto broker which is open source broker this broker is a application which operates on the port number 1883 allows the devices and client to connect it through mqtt client and enables communication. My server here acts a mqtt client which is tuned to the broker and messages coming from it are processed and either stored in the Database or routed to the web client. The Web client is designed in Angular js and it also has a MQTT client which allows showing a real time data on web page.

Project dairy –

This part includes the steps involved in completing this project –

Step 1- Project Finalization – After project proposal I got my feedback on the same day and my project was finalized on 26th of November 2018.

Step 2- Components Gathering – The next step was to work on hardware design and select the suitable components which I did for my project I needed 2 sensor and for which I selected ldr for light sensing and lm35 for temperature as they were the within power supply range of esp and also these sensors are compatible with esp. After doing the research I finally ordered the components and all the components were available within a week.

Step 3 – Circuit assembling on breadboard – After getting all components I assembled on the breadboard and interfaced it with esp and ran following tests –

Interfacing ldr – After designing a potential circuit I assembled the ldr on breadboard and made the connections likewise and wrote a program to read the analog values coming from the ldr. I got the raw values getting converted from the adc, but I wanted to check whether these values are proportional to light or not so I checked it by focusing the light of the ldr and then I observed that the values I was reading and printing on the serial monitor were changing according to the light intensity hence I wrote the code to convert these values into the range of 0% to 100% That I did it by dividing the analog values coming from the sensor with 1024(3v) and then multiplying it with 100 to get a percentage intensity.

Interfacing the Lm35 – The sensor gives the output of 10 mv/degree change in the temperature I wanted to check if the output changes accordingly, hence I interfaced lm35 to its analog port pin and checked if the values by printing them on serial monitor then I had to convert the raw values coming from the sensor to degree Celsius, After reading the values from the analog pin it has to divided by 1024 times 3300 it is divided because a span of 1024 occupies 3.3v. Here we get the ratio of raw value to full span of 1024 and then multiply with 3300 to get millivolt value, Since the output pin can give out a maximum of 3.3 volts (1024),

1024 represents the possible range it can give out. Finally dividing it by 10 will give us the final temperature in degree Celsius.

Interfacing Relay and Fan – The Relay and Fan is used to control the temperature, here I had to interface it with esp in order to turn on the fan through web page I have used 5v dc relay to control a 5v fan or more voltage ratted fan, its interfaced with esp gpio, when logic at gpio is high the relay is triggered and relay turns on turning fan on.

This all assembling took me a day to complete, So by the end of the 7th December I got had my hardware ready.

Step 4- Was to write a program to interface the hardware to the platform which was ready and hosted on aws instance. I did my research and I found pubsbuclient as a software library to be used for interfacing my hardware to the platform, As the platform was hosting an open source mosquitto mqtt broker the pubsub client for all set to go with, so for next 2-3 days I spent coding for the hardware to speak with my platform in a efficient way.

Step 4 – Final step was testing and to maintain all the pieces of the project to keep working, The testing phase is explained in details below.

Testing -

1] Testing at sensor was done in order to make sure the output of the sensor is varying according to the change in the physical quantity. For both the sensors I have used I followed the procedure as discussed below –

I connected the sensors with the analog port pin and I wrote the program to read the data and print it on serial monitor and observed the change in the raw values by changing the physical parameters nearby, I observed the proportional change which was exactly I was expecting.

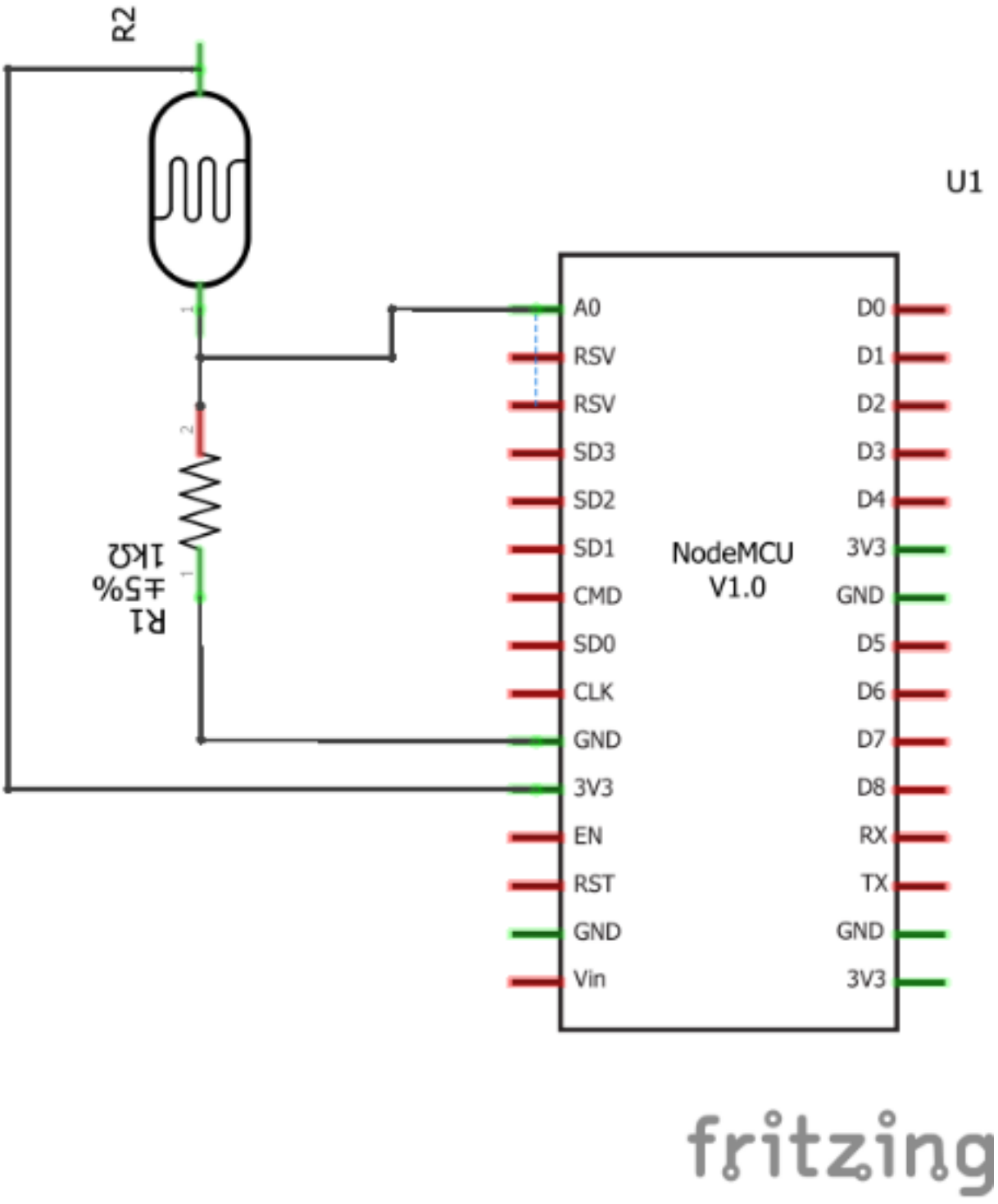
2] Testing at actuating side was to check weather relay switches upon turning the gpio high or not, as the module came along with the onboard transistor to switch my job was to just provide logic high or low to the data pin of the relay module.

So I wrote a basic code to toggle the gpio pin and connected and relay module, and checked. It worked very well.

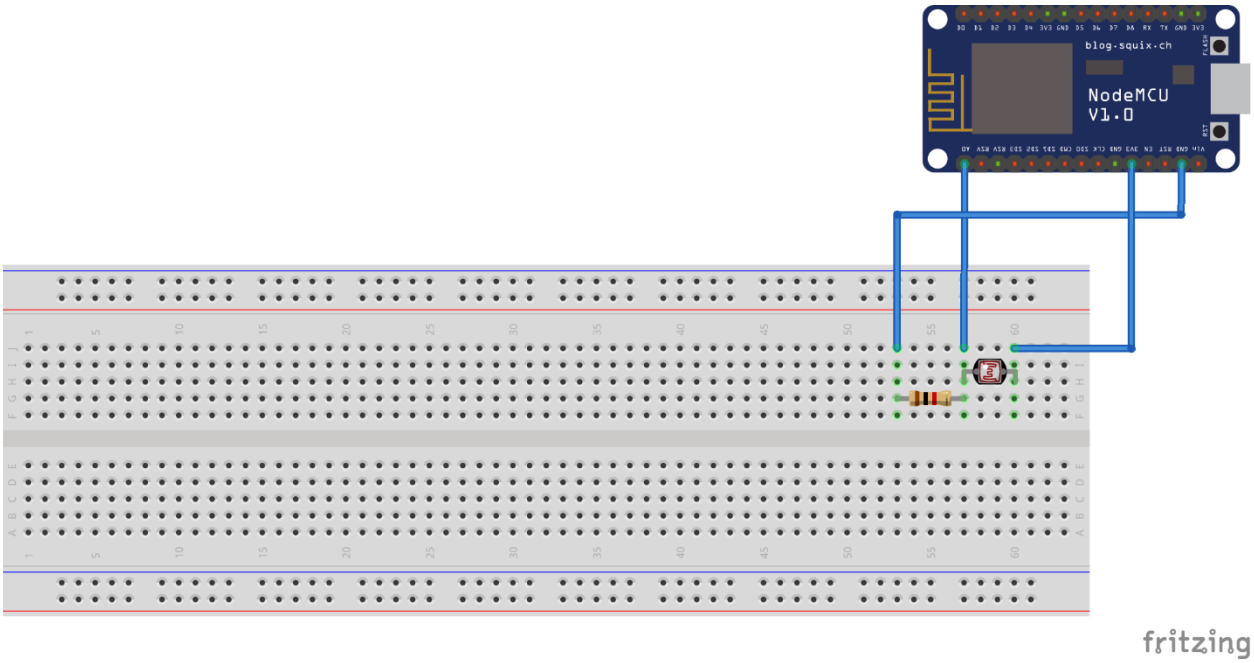
3] Testing the connection from device to cloud, Here I had got so many problems as my mqtt broker was going under crash for many times as I was uploading the data at very fast rate, so extended the delay between the uploading data and it worked later on. This problem aroused in the first place because I was using the tier free account on aws which has limited operating power.

Schematics-

Node 1 – Light intensity monitoring

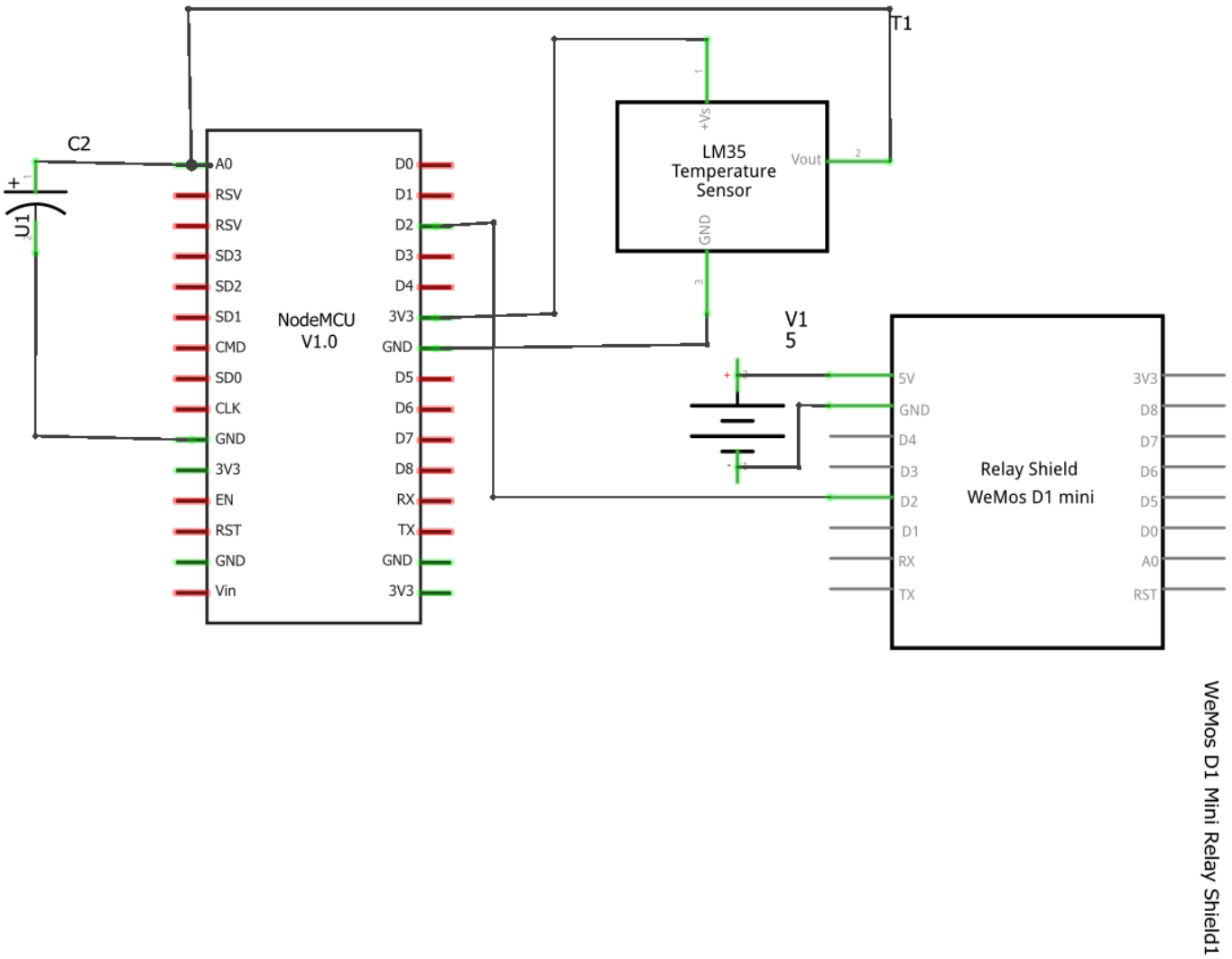


i. Circuit diagram of light intensity monitoring

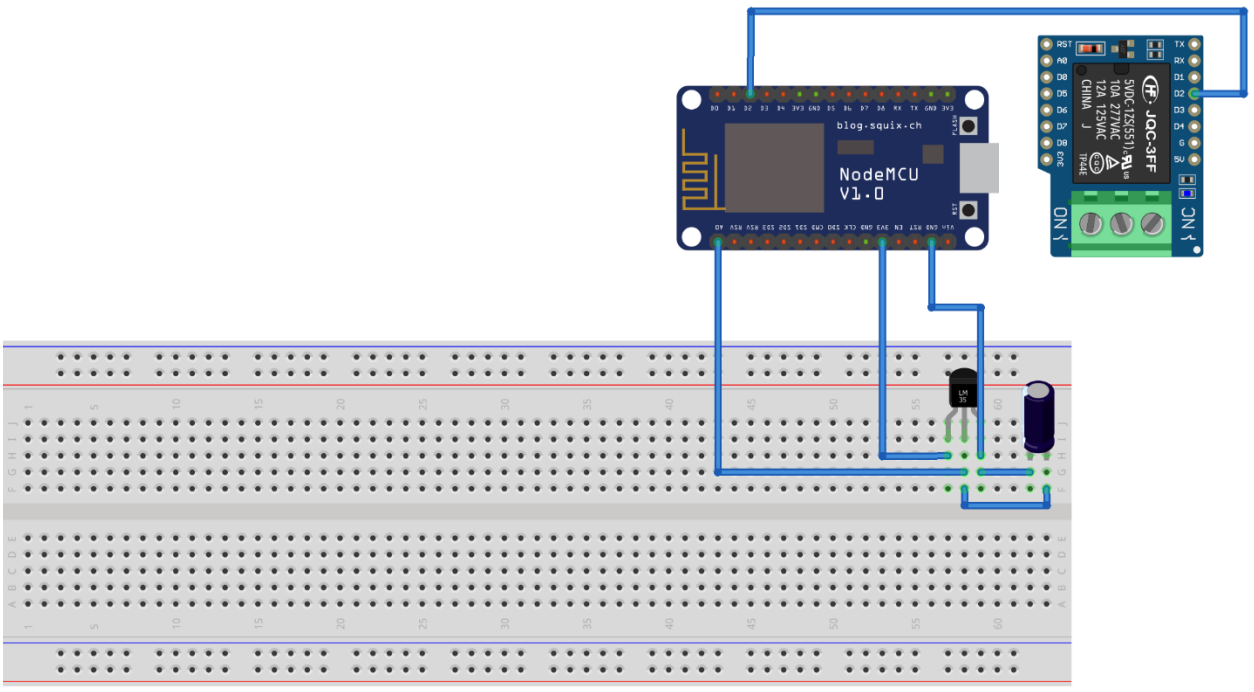


ii. Breadboard assembly of Light intensity monitoring node

Node 2- Temperature monitoring node



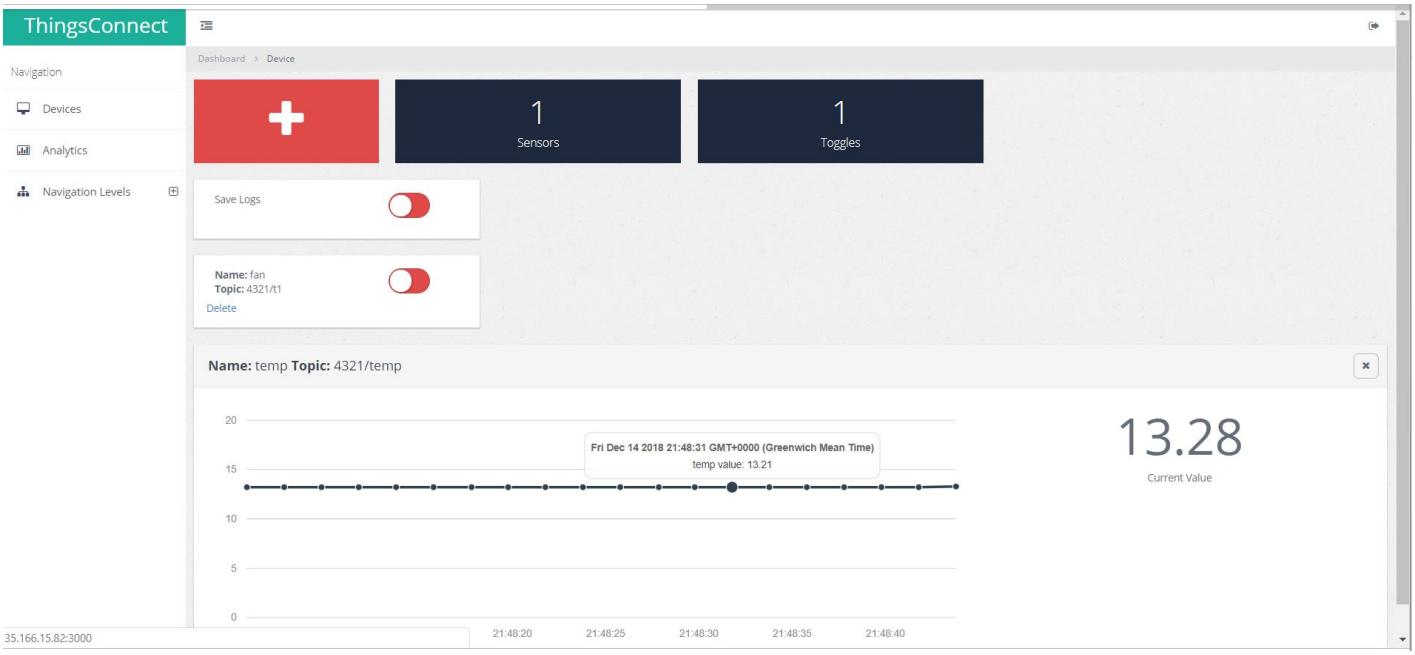
i. Circuit Diagram of the temperature node



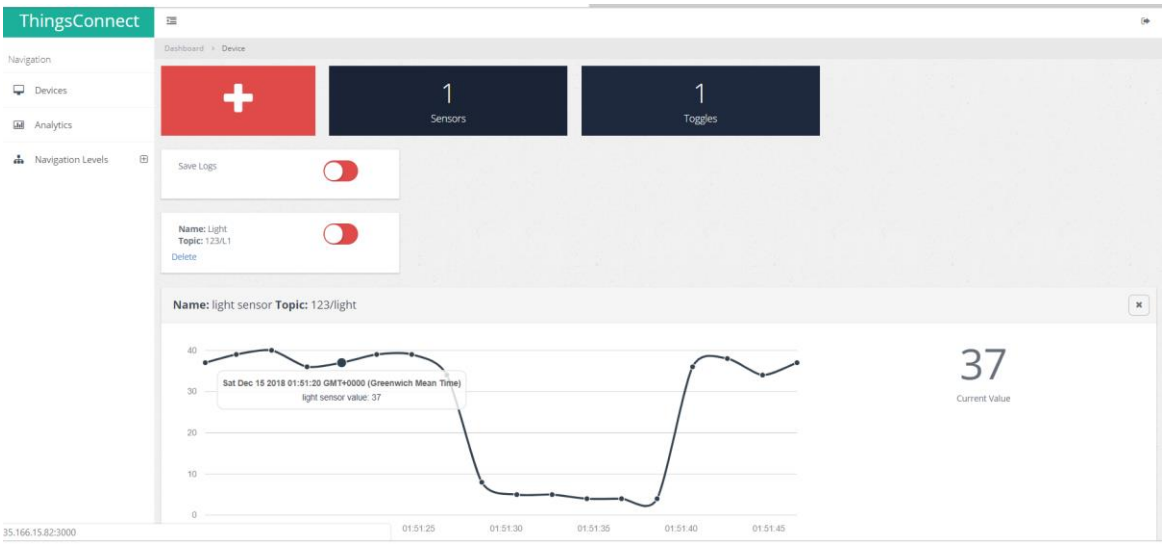
ii. Breadboard assembly of the temperature node.

Web Application snaps-

For temp node –



For light node –



Code for temp node-

```
#include <ESP8266WiFi.h>     // wifi library to access the wifi apis for esp
#include <PubSubClient.h>       // mqtt client library



const char* ssid = "DesiBoys";    // ssid for wifi netwrok
const char* password = "indopakdesiboys";
byte mqtt_server[] = {35,166,15,82};// my mqtt broker ip addresss, 1883 is a default port for broker



WiFiClient espClient;         //  declaring wifi client
PubSubClient client(espClient);              //   passing the wifii client for the pubsub client



////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////

// Global varaibles

char buf[30];


char* outTopic1     = "4321/temp";


char* inTopic1      = "4321/t1";
```

```
char* clientId     = "temp_node";


char* payloadOn1  = "true";
char* payloadOn2  = "false";
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////
// This is the function which executes to accomplish the wifi connection
void setup_wifi() {


  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);


  WiFi.begin(ssid, password);
//WiFi.begin(ssid);


  while (WiFi.status() != WL_CONNECTED) {      // this loops till the esp connects to the internet


    delay(500);
    Serial.print(".");
  }


  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
```

```
  Serial.println(WiFi.macAddress());
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////

void callback(char* topic, byte* payload, unsigned int length) {    // this function is callback function which receives the data from the mqtt server, it is event based.
   char* json;
 json = (char*) malloc(length + 1);       // json receives the data from the payload byte varaible

 memcpy(json, payload, length);                // memory has to be allocated hence memcpy functions is used to allocated dedicated memory to staore the data from payload varaible

 json[length] = '\0';
Serial.println(topic);
Serial.println(json);
 if (String(payloadOn2) == String(json))  // The payloadOn2 varaible is used store the expected value coming from the received and json has the current value of the data received which is compared and if the data matches the respective command is executed.

  {
  digitalWrite(2,HIGH);
  digitalWrite(D2,LOW);
 }

 if (String(payloadOn1) == String(json))  // The payloadOn1 varaible is used store the expected value coming from the received and json has the current value of the data received which is compared and if the data matches the respective command is executed.
```

```
   {
    digitalWrite(2,LOW);
    digitalWrite(D2,HIGH);
  }



  free(json);
}

////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
// Reconnect function is used to keep esp connected with the server



void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {     //   this loop makes sure that the connections is  alive to the broker

    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("temp_node")) {    //  this loop connects it with mqtt broker

      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish("outTopic", "hello world");
      client.subscribe("4321/t1");          // subscribes to the topic in the inTopic1 varaible
```

```
   } else {
     Serial.print("failed, rc=");
     Serial.print(client.state());
     Serial.println(" try again in 5 seconds");
     // Wait 5 seconds before retrying
     delay(5000);
   }
  }
}
```

//
//////////////////////

// This function i executed at the begining and only once all the basic setup is stated in this function

```
void setup() {

  int sensorPin = A0;

  Serial.begin(115200);

  setup_wifi();                //  Make connection to the wifi
  client.setServer(mqtt_server, 1883);    // pass mqtt broker ip address and port number

  client.setCallback(callback);     // set callback function by passing callback function

  Serial.println("start");
```

```
  pinMode(sensorPin, INPUT);     //  configure the A0 pin as input
     pinMode(2, OUTPUT);       //   configure the led pin as output
      pinMode(D2, OUTPUT);                 //   configure the D2 pin as output
//   pinMode(9, INPUT);//ir6
digitalWrite(2,HIGH);   // turn the led pin  high
digitalWrite(D2,LOW);
 }
////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////

// This loop keeps going continuosly in which main logic is written



void loop() {



 if (!client.connected()) {      //   checks for the continuos connection
   reconnect();
 }


//Serial.println("in loop");
 char buf[30];
  float sen1, sen2;


  sen1=analogRead(A0);     // Reads the analog data from A0
float  val = (float(sen1)/float(1024)) * float(3300);      // Divides raw value with 1024 (3v) maximum
```

```
Serial.println ( val);

float lit = val/float(10); // Gives the value in degree celcius
Serial.println ( lit);
 String k=String(lit);
 k.toCharArray(buf,6);
 client.publish(outTopic1,buf);
 Serial.println (sen1);




   client.loop();
  delay(2000);

}
```

Code for light intensity node –

```
#include <ESP8266WiFi.h>            // wifi library to access the wifi apis for esp
#include <PubSubClient.h>            // mqtt client library
```

```
const char* ssid = "DesiBoys";           // ssid for wifi netwrok

const char* password = "indopakdesiboys";

byte mqtt_server[] = {35, 166, 15, 82};       // my mqtt broker ip addresss, 1883 is a default port for
broker



WiFiClient espClient;                     //  declaring wifi client

PubSubClient client(espClient);                //   passing the wifii client for the pubsub client



////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////

// Global vraibles


char buf[30];



char* outTopic1     = "123/light";


char* inTopic1      = "123/L1";


char* clientId     = "arduino1";


char* payloadOn1  = "true";

char* payloadOn2  = "false";


////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////
```

```
// Setup
// This is the function which executes to accomplish the wifi connection
void setup_wifi() {

  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);



  WiFi.begin(ssid,password);  // initialise the wifi network

  while (WiFi.status() != WL_CONNECTED) {            // this loops till the esp connects to the internet

    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println(WiFi.macAddress());
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////
```

```
void callback(char* topic, byte* payload, unsigned int length) {    // this function is callback function which receives the data from the mqtt server, it is event based.

  char* json;

  json = (char*) malloc(length + 1);                // json receives the data from the payload byte varaible

  memcpy(json, payload, length);                    // memory has to be allocated hence memcpy functions is used to allocated dedicated memory to staore the data from payload varaible

  json[length] = '\0';

  Serial.println(topic);

  Serial.println(json);

  if (String(payloadOn2) == String(json))           // The payloadOn2 varaible is used store the expected value coming from the received and json has the current value of the data received which is compared and if the data matches the respective command is executed.

  {

    digitalWrite(2, HIGH);

    digitalWrite(D2, HIGH);          //  In this case the relay is turned on

  }


  if (String(payloadOn1) == String(json))    // The payloadOn1 varaible is used store the expected value coming from the received and json has the current value of the data received which is compared and if the data matches the respective command is executed.


  {

    digitalWrite(2, LOW);

    digitalWrite(D2, LOW);  // In this case the relay is turned off

  }









  free(json);

}
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////
// Reconnect function is used to keep esp connected with the server


void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {                    //   this loop makes sure that the connections is  alive to the broker
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP8266Client")) {            //  this loop connects it with mqtt broker
      Serial.println("connected");
      // Once connected, publish an announcement...
      client.publish("outTopic", "hello world");
      client.subscribe(inTopic1);                    // subscribes to the topic in the inTopic1 varaible
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}



////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////
```

```
// This function i executed at the begining and only once all the basic setup is stated in this function

void setup() {

  int sensorPin = A0;

  Serial.begin(115200);

  setup_wifi();                          //  Make connection to the wifi
  client.setServer(mqtt_server, 1883);          // pass mqtt broker ip address and port number
  client.setCallback(callback);              // set callback function by passing callback function
  Serial.println("start");



  pinMode(sensorPin, INPUT);               //  configure the A0 pin as input
  pinMode(2, OUTPUT);                 //   configure the led pin as output
  pinMode(D2, OUTPUT);                //   configure the D2 pin as output
  //   pinMode(9, INPUT);                    //
  digitalWrite(2, HIGH);                // turn the led pin  high
}



////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////

// This loop keeps going continuosly in which main logic is written

void loop() {
```

```
if (!client.connected()) {            //   checks for the continuos connection
  reconnect();
}

//Serial.println("in loop");
char buf[30];
int  sen1, sen2;

sen1 = analogRead(A0);                  // Reads the analog data from A0
float  val = float(sen1) / float(1024);          // Divides raw value with 1024 (3v) maximum
Serial.println ( val);
float  lit_percen = float(val) * float(100);      // multiplies the result with 100 to get percentage
int lit = int(lit_percen);
Serial.println ( lit);
String k = String(lit);                //  casting the result in string
k.toCharArray(buf, 6);              // convert in the char array
client.publish(outTopic1, buf);       // publish data to outopic1 varaible
Serial.println (sen1);




Serial.println ("divide ");



client.loop();
delay(2000);
```

```
}
```