**IOT ENABLED**

**SMART POULTRY**

***Team Name: ECHO***

***Team Members*:**

***M.A.CHAKRAVARTHI***

***R.ADITHYA***

***K.MUNI TEJA REDDY***

***Abstract:***

*Internet of Things (IOT) is a vision towards Future Internet* where “things” are provided with enough intelligence to communicate with each other without the human intervention. With the proliferation of Internet of Things (IOT) Devices such as smart phones, sensors, cameras, and RFID etc. It is possible to collect massive amount of data for localization and tracking of people within commercial buildings. This proposed occupancy monitoring develop effective data fusion techniques for improving occupancy monitoring accuracy using a multitude of sources for the occupancy collection of data, IR sensors are used for the detection of existence of the persons and it will count the people in the buildings entering.

Advances in building technologies are combining energy efficiency, networked sensors, and data recording in exciting ways. Modern facilities can adjust lighting, heating, and cooling outputs to maximize efficiency, provide better physical security, improve wayfinding for occupants, and provide detailed reports of building use. This column will briefly explore the idea of “smart buildings,” describe some of the technologies that are being developed for these buildings, and explore their implications for libraries. A brief listing of selected smart building technologies is also provided.

Advances in building technologies are combining energy

efficiency, networked sensors, and data recording in exciting

ways. Modern facilities can adjust lighting, heating, and cooling

outputs to maximize efficiency, provide better physical security,

improve wayfinding for occupants, and provide detailed reports

of building use. This column will briefly explore the idea of

“smart buildings,” describe some of the technologies that are

being developed for these buildings, and explore their implica-

tions for libraries. A brief listing of selected smart building

technologies is also provided.

Advances in building technologies are combining energy

efficiency, networked sensors, and data recording in exciting

ways. Modern facilities can adjust lighting, heating, and cooling

outputs to maximize efficiency, provide better physical security,

improve wayfinding for occupants, and provide detailed reports

of building use. This column will briefly explore the idea of

“smart buildings,” describe some of the technologies that are

being developed for these buildings, and explore their implica-

tions for libraries. A brief listing of selected smart building

technologies is also provided.

Advances in building technologies are combining energy

efficiency, networked sensors, and data recording in exciting

ways. Modern facilities can adjust lighting, heating, and cooling

outputs to maximize efficiency, provide better physical security,

improve wayfinding for occupants, and provide detailed reports

of building use. This column will briefly explore the idea of

“smart buildings,” describe some of the technologies that are

being developed for these buildings, and explore their implica-

tions for libraries. A brief listing of selected smart building

technologies is also provided.

Advances in building technologies are combining energy

efficiency, networked sensors, and data recording in exciting

ways. Modern facilities can adjust lighting, heating, and cooling

outputs to maximize efficiency, provide better physical security,

improve wayfinding for occupants, and provide detailed reports

of building use. This column will briefly explore the idea of

“smart buildings,” describe some of the technologies that are

being developed for these buildings, and explore their implica-

tions for libraries. A brief listing of selected smart building

technologies is also provided.

Advances in building technologies are combining energy

efficiency, networked sensors, and data recording in exciting

ways. Modern facilities can adjust lighting, heating, and cooling

outputs to maximize efficiency, provide better physical security,

improve wayfinding for occupants, and provide detailed reports

of building use. This column will briefly explore the idea of

“smart buildings,” describe some of the technologies that are

being developed for these buildings, and explore their implica-

tions for libraries. A brief listing of selected smart building

technologies is also provided.

Advances in building technologies are combining energy

efficiency, networked sensors, and data recording in exciting

ways. Modern facilities can adjust lighting, heating, and cooling

outputs to maximize efficiency, provide better physical security,

improve wayfinding for occupants, and provide detailed reports

of building use. This column will briefly explore the idea of

“smart buildings,” describe some of the technologies that are

being developed for these buildings, and explore their implica-

tions for libraries. A brief listing of selected smart building

technologies is also provided.

**Keywords**: IOT, Sensor, occupancy monitoring, Energy Efficiency, smart buildings, smart libraries, space planning.

***Introduction :***

Reducing electrical energy consumption is of paramount importance. One unit of energy saved is equal to one unit of energy generated. Apart from reduction in electricity bills this has a great environmental impact in reducing pollution causing gaseous and liquid pollutants. In this project energy saving system by using Green building concept is brought out.

The model demonstrates the energy consumption in Conventional Building using energy saving parameters such as Temperature, Humidity, lighting controls and some of the measures. The energy saving system consists of three modes i.e., Eco Mode, Away Mode, manual Mode which turns on/off the electrical appliances when required, all the sensor parameters are sent to the IBM Watson cloud and these values are stored in the database . This data can be visualized in the Node Red UI and through that user interface (UI) the user can control the appliances. necessary notifications are sent whenever required through third party services.

Buildings used to be fairly simple: their own requirements were four walls, a roof, and maybe some flooring. As humans have developed more complicated technologies such as plumbing, electricity, telephone systems, networked computing, and wireless networking, the level of complexity required in building projects has increased exponentially. Modern buildings are almost like living organisms, with intricate systems for providing amenities and information. Advances in sensors and networking are creating new ways for buildings to meet and even anticipate the needs of their users while reducing costs and increasing efficiency. This column will explore “smart building” technologies and discuss their use in libraries. A brief listing of smart building technologies is also provided.

***Hardware Components :***

**Node MCU :**

1.Node MCU development board is an open-source IOT development kit.

2.It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12E module.

3.The term "Node MCU" by default refers to the firmware rather than the dev kits. The firmware uses the Lua scripting language.

4.It is a low cost hardware platform available for development of IOT applicationsESP8266 is high integration **wireless SOC**

5.ESP-12E Wi-Fi module is developed by **Ai-thinker Team**

6.Integrated low power **32-bit MCU -Tensilica L106, with 4MB Flash memory, 128KB SRAM**

**7.Clock speed** support **80 MHz** and over clock 160 MHz,

8.Integrated **10-bit ADC and 13 GPIO’s**

9.Integrated **TCP/IP protocol** stack

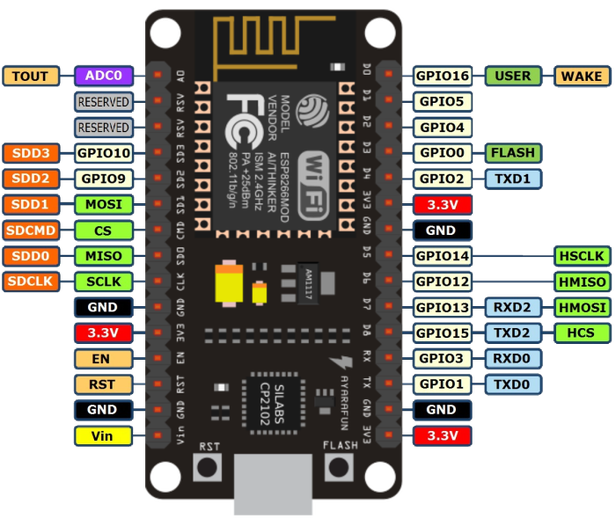
10.Supports IEEE 802.11 b/g/n , Wi-Fi 2.4 GHz, WPA/WPA2 , +20dBm output power

11.Support **STA/AP/STA+AP** operation modes

12.Integrated UART, I2C, I2S, IRDA, PWM, GPIO, SDIO 2.0, (H) SPI interface

**13.Deep sleep power** <10uA, Standby power consumption less than 1.0mW

14.Supports the RTOS (Real-Time Operating System)



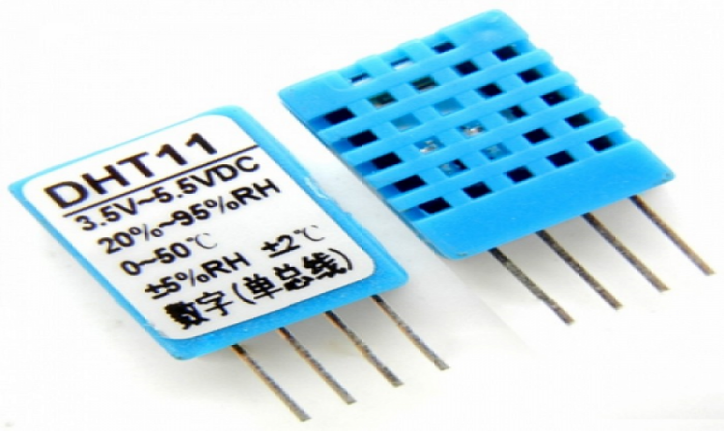
**DHT11 :**

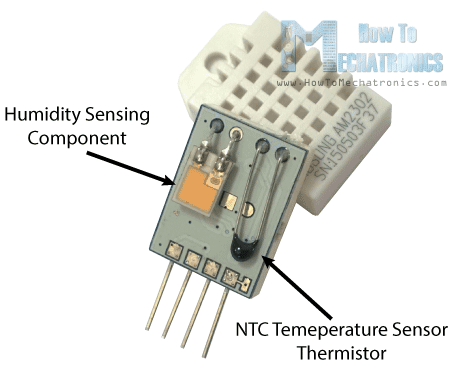
The DHT11 is a basic, ultra low-cost **digital temperature and humidity sensor.** It uses a **resistive type humidity sensor** and **a thermistor** to measure the surrounding air, and spits out a digital signal on the data pin .

* Operating Voltage: 3.5V to 5.5V
* Operating current: 0.3mA (measuring) 60uA (standby)
* Output: Serial data
* Temperature Range: 0°C to 50°C
* Humidity Range: 20% to 90%
* Resolution: Temperature and Humidity both are 16-bit
* Accuracy: ±1°C and ±1%
* These are applied in :
* Measure temperature and humidity
* Local Weather station
* Automatic climate control
* Environment monitoring

The **DHT11**is a commonly used **Temperature and humidity sensor.** The sensor comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data. The sensor is also factory calibrated and hence easy to interface with other microcontrollers.

The sensor can measure temperature from 0°C to 50°C and humidity from 20% to 90% with an accuracy of ±1°C and ±1%. So if you are looking to measure in this range then this sensor might be the right choice for you.





**LDR :**

LDR is Light Dependent Resistor

An LDR is a component that has a (variable) resistance that changes with the light intensity that falls upon it. This allows them to be used in light sensing circuits.

A **L**ight **D**ependent **R**esistor (**LDR**) or a photo resistor is a device whose resistivity is a function of the incident electromagnetic radiation. Hence, they are light sensitive devices. They are also called as photo conductors, photo conductive cells or simply photocells. They are made up of semiconductor materials having high resistance.

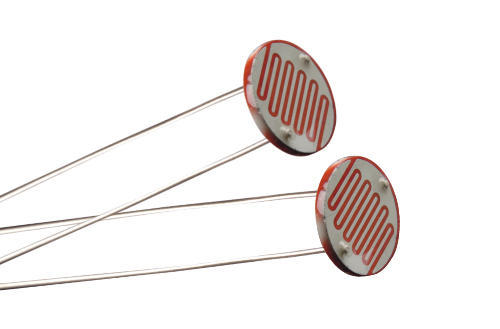
The most common type of LDR has a resistance that falls with an increase in the light intensity falling upon the device. The resistance of an LDR may typically have the following resistances:

Daylight= 5000Ω

Dark= 20000000Ω

The most obvious application for an LDR is to automatically turn on a light at a certain light level. An example of this could be a street light or a garden light.

LDRs can be used to control the shutter speed on a camera. The LDR would be used to measure the light intensity which then adjusts the camera shutter speed to the appropriate level.



**Ultrasonic Sensor :**

Operating voltage: +5V

Theoretical  Measuring Distance: 2cm to 450cm

Practical Measuring Distance: 2cm to 80cm

Accuracy: 3mm

Measuring angle covered: <15°

Operating Current: <15mA

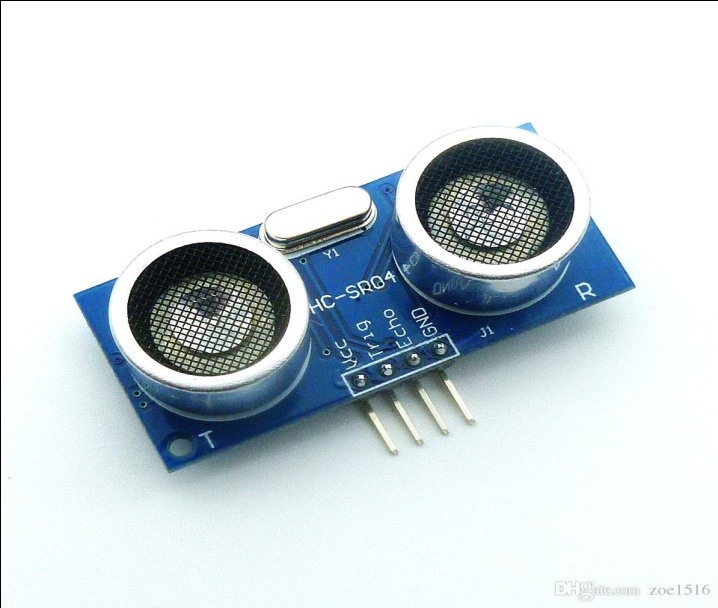
Operating Frequency: 40Hz

1.The Vcc pin powers the sensor, typically with +5V

2.Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave.

3.Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor.

4.This pin is connected to the Ground of the system.



**LED:**

A little used technique is to use an LED as both a light emitter and a light detector. Just like a dog walking on its hind legs one marvels, not that it is done well, but that it is done at all.

"A LED is simply a diode that has been doped specifically for efficient light emission and has been packaged in a transparent case. Therefore, if inserted into a circuit in the same way as a photodiode, which is essentially the same thing, the LED will perform the same function."

**Due to the high efficiency of LED sensor systems, they are highly sought-after for multiple applications. One particular application for LEDs is their use as photodiodes for light detection. While acting as photodiodes, LEDs are very sensitive to wavelengths that are equal to or shorter than the predominant wavelength they emit. For instance, a green LED is sensitive to blue light and some green light, but not to red or yellow light.**

The LED charges to 5V in a detector mode, and this charge is sustained in the circuit by the inherent capacitance of the diode.LEDs can be used as a light sensor by combining them with a microcontroller.

LEDs detect a much narrower band of light having peak sensitivity at a wavelength slightly lower than the wavelength these emit. For example, a transparent red LED emits around 660nm light and responds better to orange light at 610nm.



***Software Components :***

**Arduino IDE :**

The arduino code is actually just plain old c without all the header part (the includes and all). when you press the 'compile' button, the IDE saves the current file as arduino.c in the 'lib/build' directory then it calls a makefile contained in the 'lib' directory.

This makefile copies arduino.c as prog.c into 'lib/tmp' adding 'wiringlite.inc' as the beginning of it. this operation makes the arduino/wiring code into a proper c file (called prog.c).

After this, it copies all the files in the 'core' directory into 'lib/tmp'. these files are the implementation of the various arduino/wiring commands adding to these files adds commands to the language

The core files are supported by pascal stang'sprocyonavr-lib that is contained in the 'lib/avrlib' directory

At this point the code contained in lib/tmp is ready to be compiled with the c compiler contained in 'tools'. If the make operation is successful then you'll have prog.hex ready to be downloaded into the processor.

The next release will see each architecture (avr/pic/8051) to treated as a 'plug-in' to the IDE so that the user can just select from a menu the microcontroller board to use and the IDE will pick the right compilation sequence.

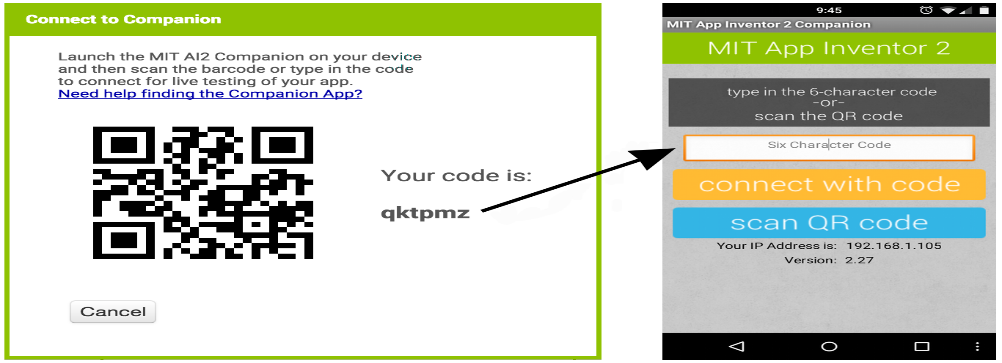
**MIT APP :**

**MIT App Inventor** is an intuitive, visual programming environment that allows everyone – even children – to build fully functional apps for smartphones and tablets. Those new to MIT App Inventor can have a simple first app up and running in less than 30 minutes. And what's more, our blocks-based tool facilitates the creation of complex, high-impact apps in significantly less time than traditional programming environments. The MIT App Inventor project seeks to democratize software development by empowering all people, especially young people, to move from technology consumption to technology creation.

App Inventor lets you develop applications for Android phones using a web browser and either a connected phone or emulator. The App Inventor servers store your work and help you keep track of your projects.

We build apps by working with:

* The App Inventor Designer, where you select the components for your app.
* The App Inventor Blocks Editor, where you assemble program blocks that specify how the components should behave. You assemble programs visually, fitting pieces together like pieces of a puzzle.



**IBM Watson :**

Watson IoT Platform service is an IoT device message broker for device registration, IoT data management, and IoT device management. Watson IoT Platform service provides secure communication to and from your devices by using MQTT and TLS.

Watson IoT Platform service is built on the following key areas:

* Connection Management - Connect and control IoT devices.
* Data Management - Use device twins to normalize, transform, and review device data for use with the Watson IoT Platform components and other services.
* Risk Management - Configure secure connectivity and architecture with access control for users and applications.

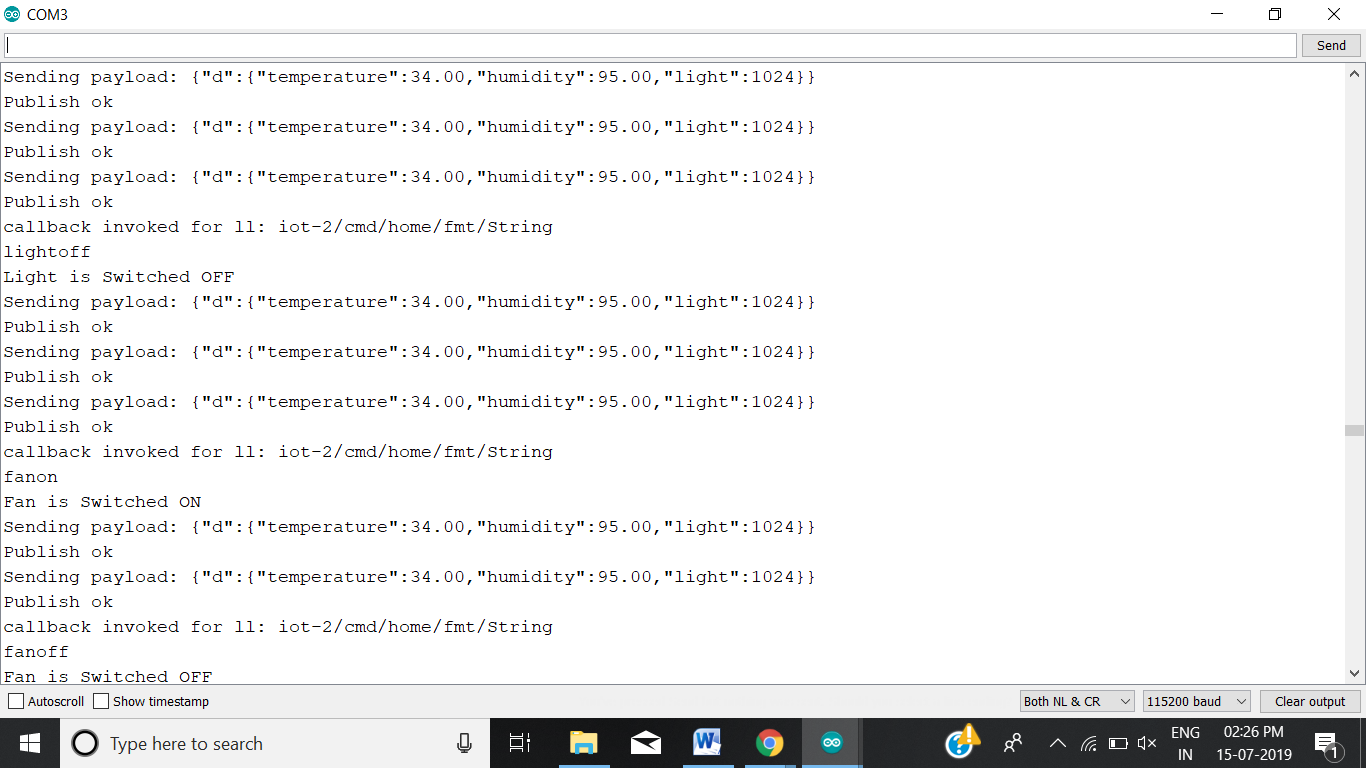
Use the [Watson IoT Platform dashboard](https://www.ibm.com/support/knowledgecenter/SSQP8H/iot/getting_started/external_dashboard.html?view=kc) to log in to your Watson IoT Platform service organization.

1. In the Watson IoT Platform dashboard, click **View Details** for the Watson IoT Platform service dashboard that you want to access.
2. Click **Launch** to open the dashboard.
3. Click **Sign In** on the IBM Watson™ IoT Platform service dashboard landing page and then log in with the IBM id that is specified in your welcome letter, or with an IBM id that you have granted access.
4. If your IBM id is a member of more than one Watson™ IoT Platform service organization, select the Watson IoT Platform organization from the  **My Organizations** menu.  
   **Tip:** The six-letter organization ID is included in the credentials page.



***Output :***

**Serial Monitor:**



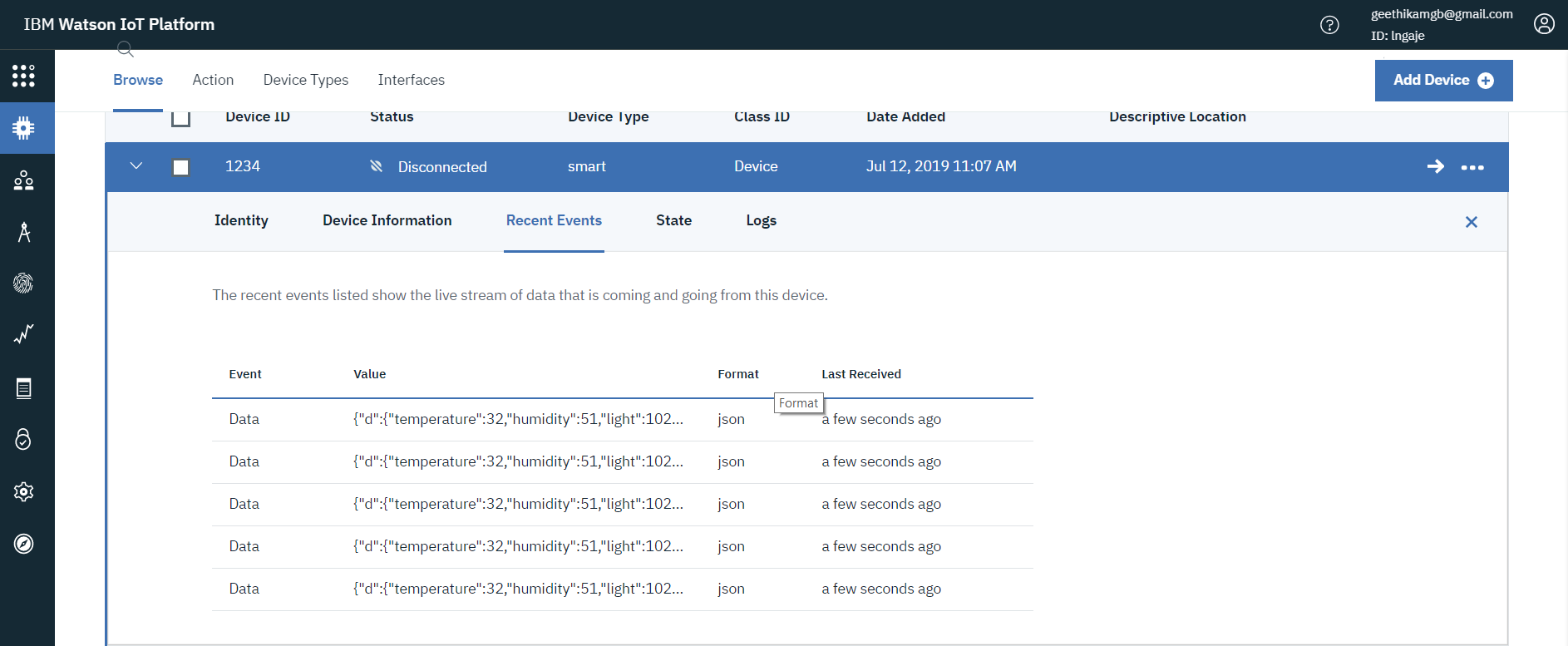
This window is called the Serial Monitor and it is part of the Arduino IDE software. Its job is to allow you to both send messages from your computer to an Arduino board (over USB) and also to receive messages from the Arduino.

We will use the Serial Monitor to debug Arduino Software Sketches or to view data sent by a working Sketch. We must have an Arduino connected by USB to our computer to be able to activate the Serial Monitor.

To get familiar with using the Serial Monitor, Copy and Paste the following example Sketch into a blank Arduino IE window. Then Verify it and if it's OK, Upload it. The next step will show us what to expect.

This is a picture of output in the serial monitor. In this we get the values of temperature, humidity , light , ultrasonic sensor , light on , light off , gate open , gate close & LDR .

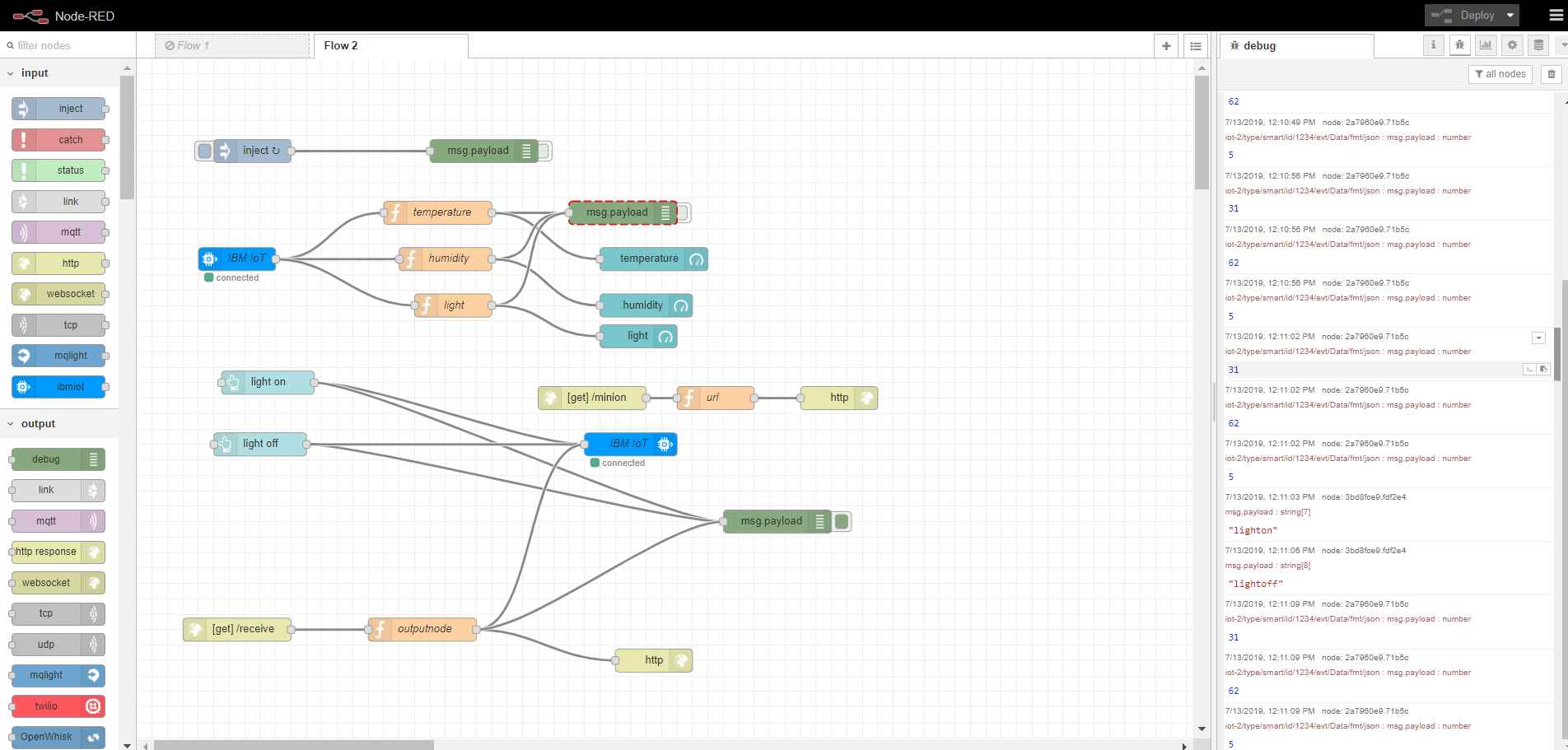
IBM Watson :



Watson, at its core, is a natural language processing tool. It uses a variety of machine learning techniques to analyze text, process data, and generate insights from large amounts of unstructured data. It’s this capability that makes Watson appear so impressive with its “understanding” of data.

In this we get the values of temperature, humidity and light. The format of this is json. These values appear in recent events. To get this output we should create new device.

**Node Red:**

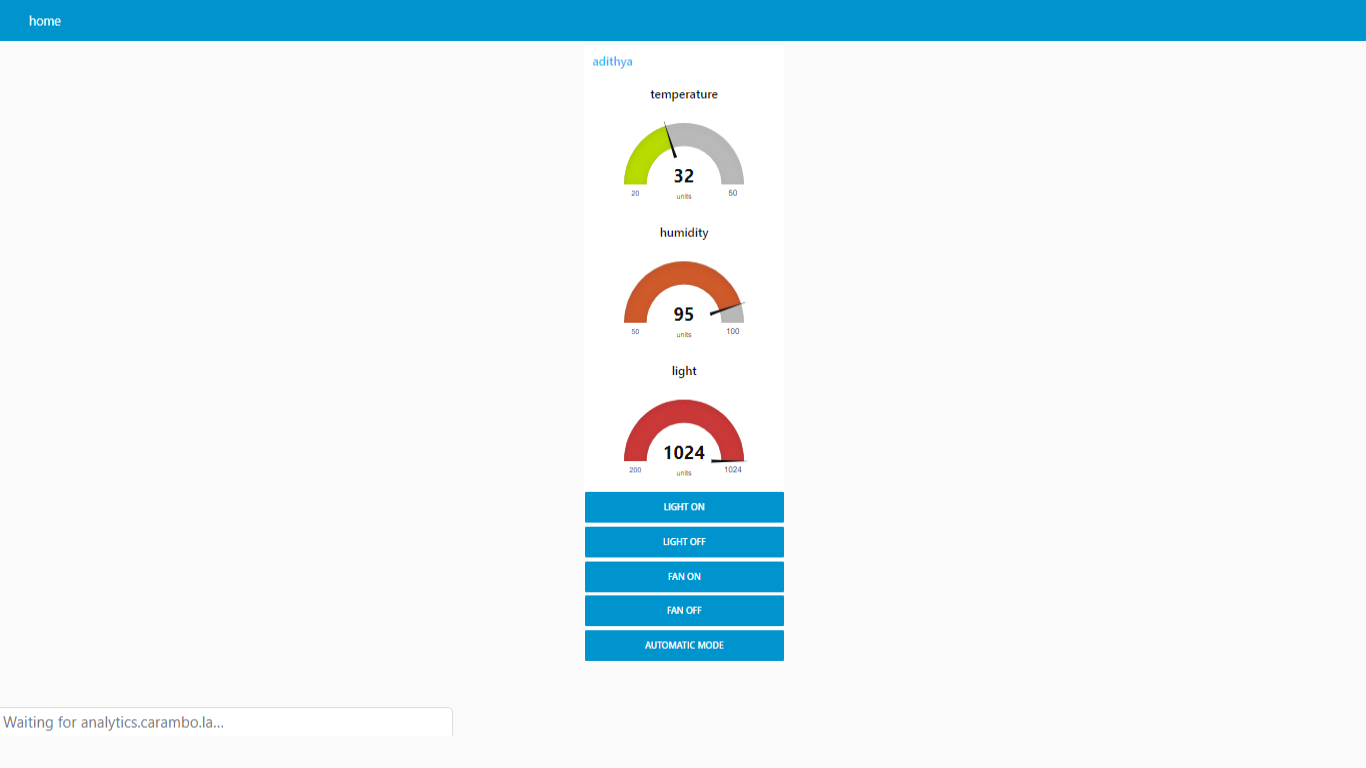


**Node-RED** is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, API’s and online services as part of the Internet of Things.

Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions. Elements of applications can be saved or shared for re-use. The runtime is built on Node.js. The flows created in Node-RED are stored using JSON. Since version 0.14 MQTT nodes can make properly configured TLS connections.

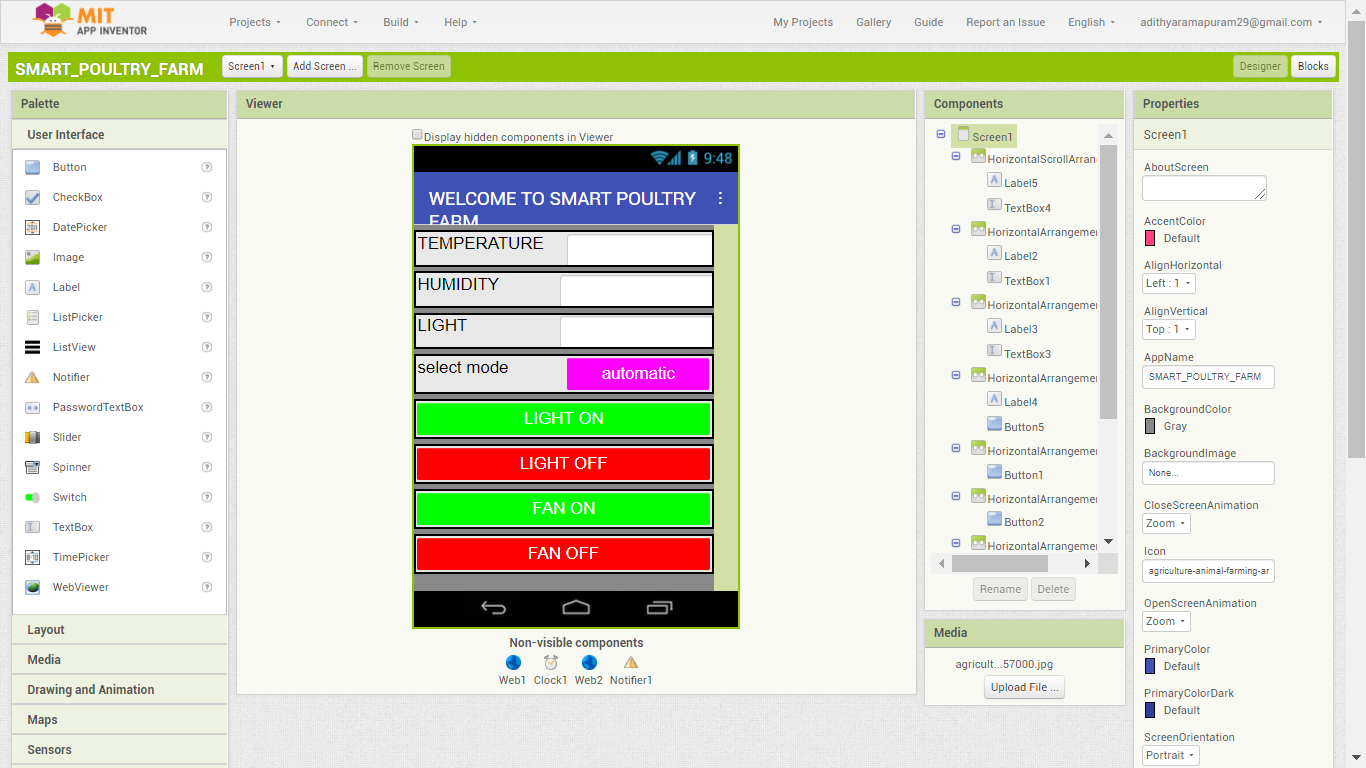
In this we create the flow chart off our project. By debugging we get the output .By clearing all the output on the right side gets removed .We created some function nodes for temperature, humidity and light.

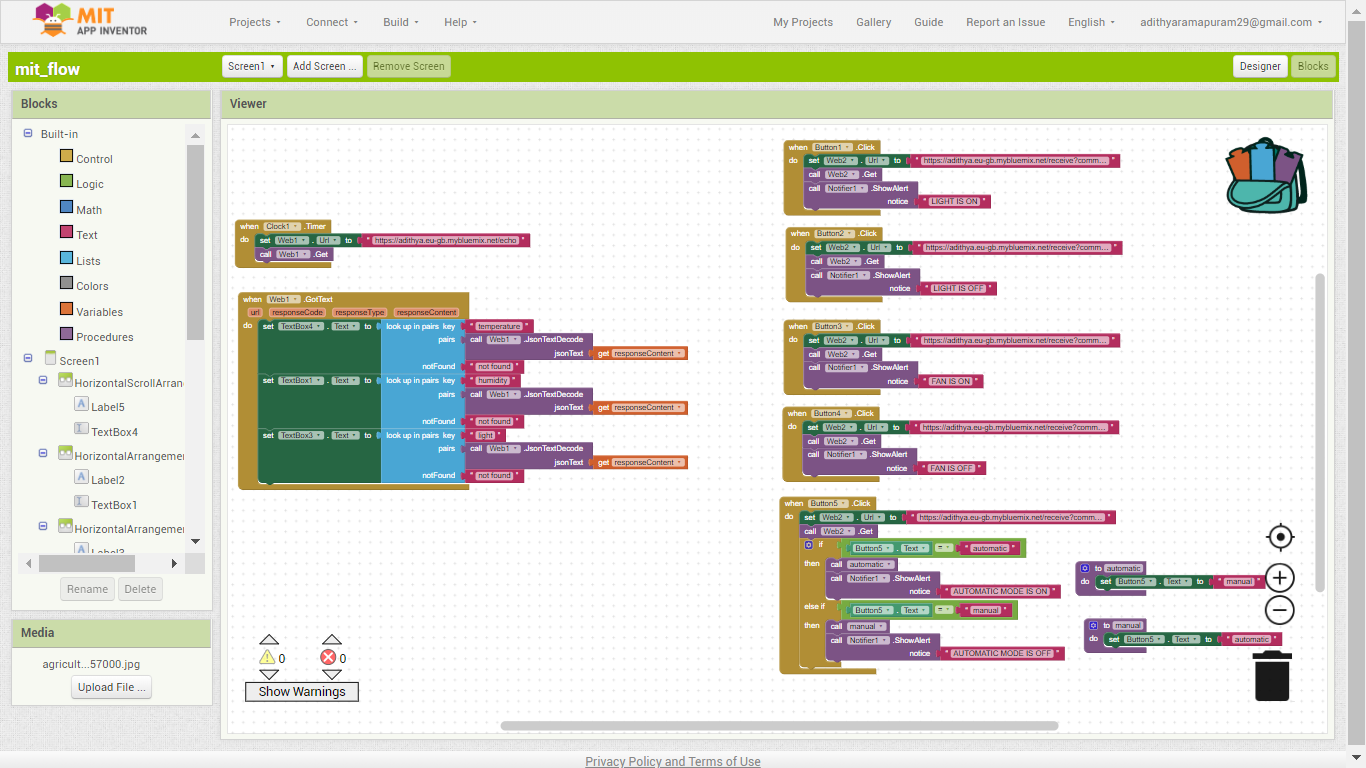
**UI Image:**



In this we get the temperature, humidity and light values. By using light on and light off we can see it in nodered .With the help of nodered we can get this output. The value of this varies.

**MIT App:**





**App Inventor** is a web application integrated development environment originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT). It allows newcomers to computer programming to create application software (apps) for two operating systems (OS): Android, and iOS.

We can install this in our android phones .We create blocks to get output. We can use web, clock, text, etc. We can scan it in our mobiles. We can operate this by using our android phones .The values of these changes frequently.

***Code :***

#include <ESP8266WiFi.h>  
#include <PubSubClient.h>  
   
//-------- Customise these values -----------  
const char\* ssid = "FoxFi69";  
const char\* password = "adithya123";  
//String command;  
#include "DHT.h"  
#define led D3  
#define led D4  
int count=1;  
  
#define DHTPIN D2    // what pin we're connected to  
#define DHTTYPE DHT11   // define type of sensor DH 11  
DHT dht (DHTPIN, DHTTYPE);  
   
#define ORG "2qp4yw"  
#define DEVICE\_TYPE "echo"  
#define DEVICE\_ID "echo"  
#define TOKEN "echoecho"  
//-------- Customise the above values --------  
 String command;  
char server[] = ORG ".[messaging.internetofthings.ibmcloud.com](http://messaging.internetofthings.ibmcloud.com/)";  
char subtopic[] = "iot-2/cmd/home/fmt/String";  
char pubtopic[] = "iot-2/evt/Data/fmt/json";  
char authMethod[] = "use-token-auth";  
char token[] = TOKEN;  
char clientId[] = "d:" ORG ":" DEVICE\_TYPE ":" DEVICE\_ID;  
   
WiFiClient wifiClient;  
void callback(char\* subtopic, byte\* payload, unsigned int payloadLength);  
PubSubClient client(server,1883, callback,wifiClient);  
  
void setup() {  
 Serial.begin(115200);  
   pinMode(D3,OUTPUT);  
  pinMode(D4,OUTPUT);  
  wifiConnect();  
  mqttConnect();  
   
 Serial.println();  
 dht.begin();  
 Serial.print("Connecting to ");  
 Serial.print(ssid);  
 WiFi.begin(ssid, password);  
 while (WiFi.status() != WL\_CONNECTED) {  
 delay(500);  
 Serial.print(".");  
 }   
 Serial.println("");  
   
 Serial.print("WiFi connected, IP address: ");  
 Serial.println(WiFi.localIP());  
}  
   
void loop() {  
 int sensorvalue=analogRead(A0);  
 //Serial.println(sensorvalue);  
float h = dht.readHumidity();  
float t = dht.readTemperature();  
if (isnan(h) || isnan(t))  
{  
Serial.println("Failed to read from DHT sensor!");  
delay(1000);  
return;  
}  
PublishData(t,h,sensorvalue);  
 if (!client.loop()) {  
    mqttConnect();  
  }  
delay(100);  
}  
void wifiConnect() {  
  Serial.print("Connecting to "); Serial.print(ssid);  
  WiFi.begin(ssid, password);  
  while (WiFi.status() != WL\_CONNECTED) {  
    delay(500);  
    Serial.print(".");  
  }  
  Serial.print("nWiFi connected, IP address: "); Serial.println(WiFi.localIP());  
}  
void mqttConnect() {  
  if (!client.connected()) {  
    Serial.print("Reconnecting MQTT client to "); Serial.println(server);  
    while (!client.connect(clientId, authMethod, token)) {  
      Serial.print(".");  
      delay(500);  
    }  
    initManagedDevice();  
    Serial.println();  
  }  
}  
void initManagedDevice() {  
  if (client.subscribe(subtopic)) {  
    Serial.println("subscribe to cmd OK");  
  } else {  
    Serial.println("subscribe to cmd FAILED");  
  }  
}  
void automatic()  
{  
  int sensorvalue=analogRead(A0);  
if(sensorvalue==1024)  
  digitalWrite(D3,HIGH);  
 else  
   digitalWrite(D3,LOW);  
float t = dht.readTemperature();  
if(t<30)  
 digitalWrite(D4,LOW);  
 else  
 digitalWrite(D4,HIGH);  
   
}   
void callback(char\* subtopic, byte\* payload, unsigned int payloadLength) {  
  Serial.print("callback invoked for ll: ");   
  Serial.println(subtopic);  
  
  for (int i = 0; i < payloadLength; i++) {  
    //Serial.println((char)payload[i]);  
    command += (char)payload[i];  
  }  
Serial.println(command);  
  
if(command=="auto" && count%2!=0 )  
{  
  automatic();  
  count++;  
 }  
 else if(command=="auto" && count%2==0 )  
  count++;  
  
else  
{  
if(command == "lighton" ){  
  digitalWrite(D3,HIGH);  
  Serial.println("Light is Switched ON");  
}  
else if(command == "lightoff"){  
  digitalWrite(D3,LOW);  
  Serial.println("Light is Switched OFF");  
}  
}  
if(command == "fanon"){  
  digitalWrite(D4,HIGH);  
  Serial.println("Fan is Switched ON");  
}  
else if(command == "fanoff"){  
  digitalWrite(D4,LOW);  
  Serial.println("Fan is Switched OFF");  
}  
command ="";  
}  
  
void PublishData(float temp, float humid,int light){  
 if (!!!client.connected()) {  
 Serial.print("Reconnecting client to ");  
 Serial.println(server);  
 while (!!!client.connect(clientId, authMethod, token)) {  
 Serial.print(".");  
 delay(500);  
 }  
 Serial.println();  
 }  
  String payload = "{\"d\":{\"temperature\":";  
  payload += temp;  
  payload+="," "\"humidity\":";  
  payload += humid;  
  payload+="," "\"light\":";  
  payload += light;  
  payload += "}}";  
 Serial.print("Sending payload: ");  
 Serial.println(payload);  
  if (client.publish(pubtopic, (char\*) payload.c\_str())) {  
 Serial.println("Publish ok");  
 } else {  
 Serial.println("Publish failed");  
 }  
}