INNOVATION GEEKS

Project Name: SMART BUILDING

Team Details:

Peddi Sakila

Veerannagari Akshita

Gande Varshini

SMART BUILDING - THE FUTURE OF YOUR BUILDING

A smart building is any structure that uses automated processes to automatically control the building's operations including heating, ventilation, air conditioning, lighting, security and other systems. A smart building uses sensors, actuators and microchips, in order to collect data and manage it according to a business' functions and services.

Smart buildings are the ones where various parameters like temperature, lights, directions(i.e. ,indoor navigation) etc. are monitored and analysed so as to make it highly efficient. In this project, we have worked on smart lights , smart fans, smart doors in a smart building and how to develop such a system and how to send data from such a system to the cloud for analysis.

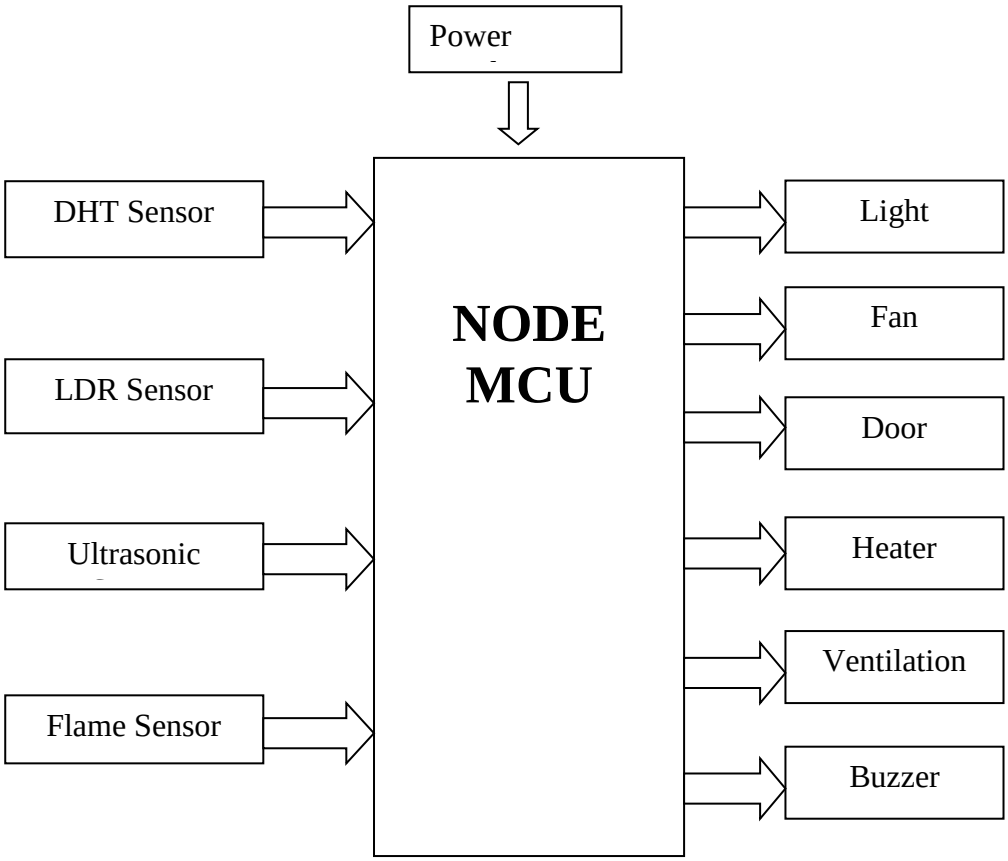
The project which we have developed can sense the brightness using LDR sensor and can switch ON/OFF the lights .The distance of the object can be measured using ultrasonic sensors and it will automatically open and close the doors. With the help of humidity values taken from the DHT sensor the room ventilation can be automatically switched ON/OFF. According to the values withdrawn from the DHT sensor the speed of the electrical fan can be regulated.

PROBLEM STATEMENT:

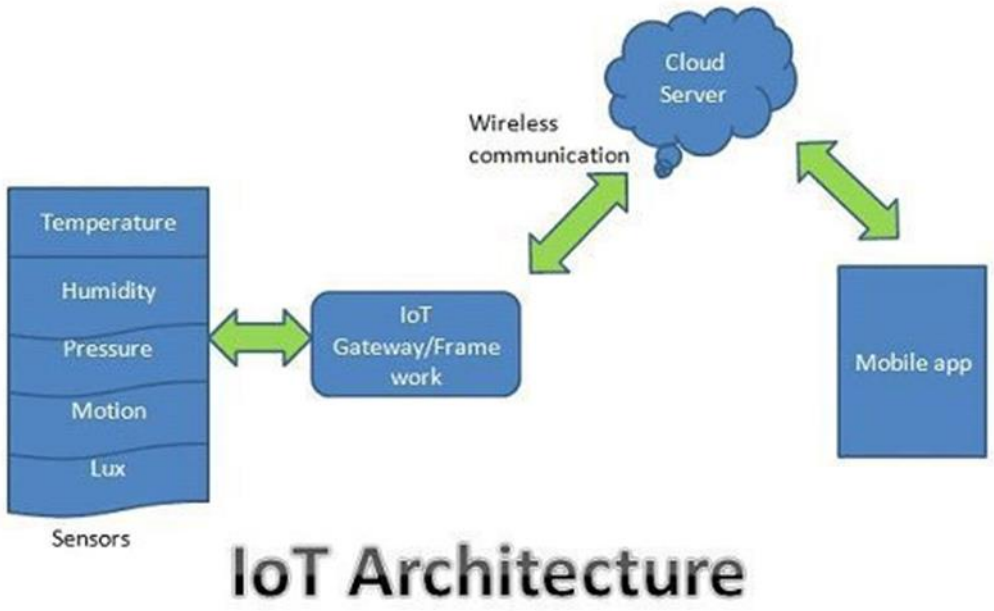
Reducing electrical energy consumption is of paramount importance. One unit of energy saved is equal to one unit of energy generated. Apart from reduction in electricity bills this has a great environmental impact in reducing pollution causing gaseous and liquid pollutants. In this project energy saving 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 visualised in the Node Red UI and through that user interface(UI) the user can control the appliances. Necessary notifications are sent whenever required.

BLOCK DIAGRAM:



ARCHITECTURE:



PROJECT WORKING :

The project is based on three modes :

1. "ECO" Mode
2. "AWAY" Mode
3. **"**MANUAL" Mode

All these modes are implemented in an App.

ECO Mode:

This is the power conserving mode or the green mode. This mode is the basis of the idea of the green building concept. Conservation of power is supported here. Based on various conditions of weather like temperature and humidity and light intensity appliances like fan, heater ,ventilation and light are controlled.

This reduces human effort and automation is done. When not required the said appliances are turned off, thus saving power and energy. In this mode the weather parameters namely temperature and humidity are continuously sensed by the means of a sensor (DHT 11) and are published or sent to the cloud. Light intensity is sensed by the light dependent resistor (LDR) and light is turned on only when there is a necessity. There is even a facility for subscribing to the values published in the cloud by means of which we can get the weather parameters of the room at any instant.

AWAY Mode:

This is the mode which saves power by turning of all the appliances present in the room which include light, ventilation, heater, and fan. This mode can be used when there is no person present in the room and all the appliances are to be turned off. When leaving from the building this mode can be used for turning off all the appliances.

MANUAL Mode:

This mode allows all the appliances to be handled manually by the user. Various buttons are present for turning on and turning off fan, heater, ventilation and light. This is the mode which allows full control to the user over the appliances. Since it is under the control of the user the problem of energy conservation and power conservation solely rests on the user.

Apart from these three modes and their functioning two other features are included in this project. They are as follows:

FIRE Alert:

In this feature a fire in the building is detected. Flame sensor is used for this purpose. Once the flame is detected in the building alert is made by the ringing of the buzzer. Also at this time an automated message is made to be sent to the nearby fire-station regarding the fire outbreak and a notification is sent to the owner of the building as well. In order to accomplish this task an API of msg91.com is utilised.

AUTOMATED Door:

This is a mechanism which is used to detect the presence of a human-being in front of the door and open or close the door based on that result. This is accomplished with the help of ultra-sonic sensor which detects the presence of objects. A certain limit is set if an object is within that distance it is detected and the door is opened automatically, else the door remains closed.

COMPONENTS USED:

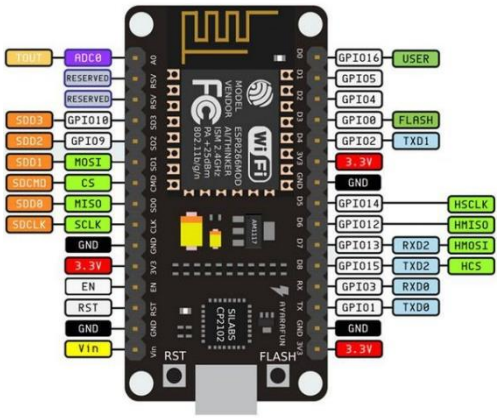
1. Node MCU
2. DHT-11 Sensor
3. Flame Sensor
4. Ultrasonic sensor
5. Light Dependant Resistor (LDR)
6. CPU Fan
7. Bulb
8. Relay setup
9. Basic Shield
10. Buzzer

Node MCU :

Node MCU development board is an open source IOT development kit. It includes firmware which runs on the ESP8266 Wi-Fi Soc Espressif Systems and Hardware which is based on the ESP-12E module. the term Node MCU by default refers to the firmware rather than the device kits. The firmware uses the LUA scripting language. It is a low-cost hardware platform available for development of IOT applications.

Node MCU Development board is featured with wi-fi capability, analog pin, digital pins and serial communication protocols. Lua Scripts are generally used to code Node MCU. Lua is a light-weight scripting language embeddable scripting language built on top of C programming language. Another way to code Node MCU is using the Arduino IDE. In this project we have made use of Arduino to code the node MCU.

PIN DESCRIPTION:



- Node MCU has 13 GPIO pins(General Purpose Input and Output Pins).
- It has a 32 bit micro controller unit.
- The size of flash memory used is 4MB.
- It has a 128KB SRAM. It can be connected via a CP2102 USB connector.
- There is only one analog pin A0.
- The pins which support Pulse With Modulation (PWM) are D2,D5,D6,D8.
- These pins are the analog output pins. The digital input pins are D0 to D9, SD0, SD1, SD2.
- There is a reset and flash pin available. 3.3V and GND useful for giving power supply.
- EN mode is used to put the board in deep sleep mode.
- TX and RX pins are the transmitter and receiver pins.
- D1 and D2 support I2C communication.
- ESP8266 is wifi module that is present here on the chip .
- There is possibility to work with the wi-fi in three modes: STA (Station Mode) , AP (Access Point Mode) and STA+AP (Station + Access Point mode).

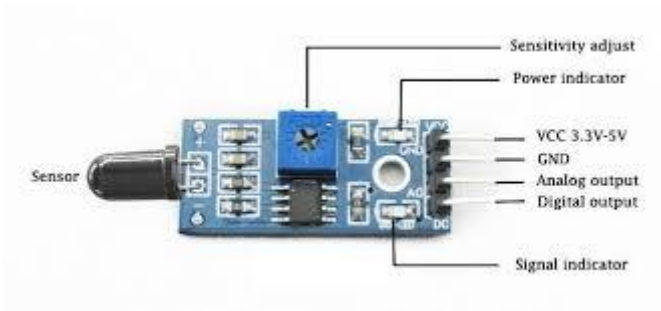
DHT-11 SENSOR:

The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). Its fairly simple to use, but requires careful timing to grab data. The only real downside of this sensor is you can only get new data from it once every 2 seconds, so when using our library, sensor readings can be up to 2 seconds old.

The technical features of this sensor are:

- Low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 20-80% humidity readings with 5% accuracy
- Good for 0-50°C temperature readings ±2°C accuracy
- No more than 1 Hz sampling rate (once every second)
- Body size 15.5mm x 12mm x 5.5mm
- 4 pins with 0.1" spacing

FLAME SENSOR:



A **flame detector** is a sensor designed to detect and respond to the presence of a flame or fire , allowing **flame detection**.

Features:

- Detection angle about 60 degrees, it is sensitive to the flame spectrum. Accuracy adjustable
- Operating voltage 3.3V-5V
- Output a. analog voltage output b. digital switch outputs (0 and 1)
- With a mounting screw holePCB size: 3cm * 1.6cm
- Power indicator (red) and digital switch output indicator (green)
- Comparator chip LM393 ,it is stable.
- Flame detection distance, lighter flame test can be triggered within 0.8m, if the intensity of flame is high , the detection distance will be increased.

ULTRASONIC SENSOR:

An Ultrasonic sensor is a device that can measure the distance to an object by using sound waves. It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back. By recording the elapsed time between the sound wave being generated and the sound wave bouncing back, it is possible to calculate the distance between the sonar sensor and the object.

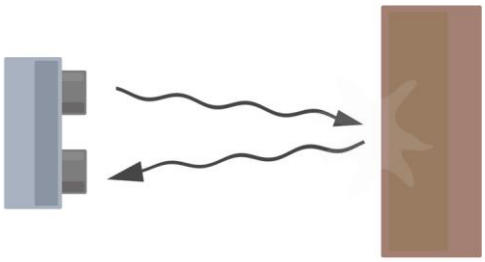


Diagram of the basic ultrasonic sensor operation

$$distance = \frac{speed\ of\ sound\ \times time\ taken}{2}$$

Since it is known that sound travels through air at about 344 m/s (1129 ft/s), you can take the time for the sound wave to return and multiply it by 344 meters (or 1129 feet) to find the total round-trip distance of the sound wave. Round-trip means that the sound wave traveled 2 times the distance to the object before it was detected by the sensor; it includes the 'trip' from the sonar sensor to the object AND the 'trip' from the object to the Ultrasonic sensor (after the sound wave bounced off the object). To find the distance to the object, simply divide the round-trip distance in half.

Features of Ultrasonic Distance Sensor:

1. Supply voltage: 5V (DC).
2. Supply current: 15mA.
3. Modulation frequency: 40Hz.
4. Output: 0 – 5V (Output high when obstacle detected in range).
5. Beam Angle: Max 15 degree.
6. Distance: 2cm – 400cm.
7. Accuracy: 0.3cm.
8. Communication: Positive TTL pulse.

LIGHT DEPENDANT RESISTOR:

A Light Dependent Resistor (LDR) is also called a photo resistor or a cadmium sulphide (CdS) cell. It is also called a photoconductor. It is basically a photocell that works on the principle of photoconductivity. The passive component is basically a resistor whose resistance value decreases when the intensity of light decreases. This **optoelectronic device** is mostly used in light varying sensor circuit, and light and dark activated switching circuit.

WORKING PRINCIPLE OF LDR:

Electrical current consists of the movement of electrons within a material. Good conductors have a large number of free electrons that can drift in a given direction under the action of a potential difference. Insulators with a high resistance have very few free electrons, and therefore it is hard to make the them move and hence a current to flow.

An LDR or photo resistor is made any semiconductor material with a high resistance. It has a high resistance because there are very few electrons that are free and able to move - the vast majority of the electrons are locked into the crystal lattice and unable to move. Therefore in this state there is a high LDR resistance.

As light falls on the semiconductor, the light photons are absorbed by the semiconductor lattice and some of their energy is transferred to the electrons. This gives some of them sufficient energy to break free from the crystal lattice so that they can then conduct electricity. This results in a lowering of the resistance of the semiconductor and hence the overall LDR resistance.

The process is progressive, and as more light shines on the LDR semiconductor, so more electrons are released to conduct electricity and the resistance falls further.

CPU FAN:

A **computer fan** is any fan inside, or attached to, a computer case used for active cooling. Fans are used to draw cooler air into the case from the outside, expel warm air from inside, and move air across a heat sink to cool a particular component. Here in this project we are just using this cpu fan to simulate the functioning of an actual fan.

BULB:

An electric bulb is used in this project to simulate the functioning of light on and light off case scenarios.

RELAY:

Relay is basically act a switch between electrical load and **Node MCU**. **Relay** have two configuration NO (Normally Open) & NC (Normally Close). **Relay** have coil which is energized by 5v, when coil energized switching action takes place, based on NO-NC configuration. The relay uses an **electromagnet** to **mechanically** switch electric appliances. A relay can be operated by a relatively small electric current that can turn **ON** or **OFF** a much larger electric current. Using relays is safe as there is no any physical contact between Node MCU and AC devices. **Relay** is basically act a **switch** between electrical load and Node MCU .Relay have two configuration **NO** (**N**ormally **O**pen) & **NC** (**N**ormally **C**lose).Relay have coil which is energized by 5v, when coil energized switching action takes place, based on NO-NC configuration. If relay is **NO** configuration then when coil is energized switching action takes place from NO-NC then load will be connected.

CONNECTIONS:

Connections are very simple, initially you need to power-up the Relay Module.

Relay Connections:

Connect **Node MCU** Ground (**GND**) pin to **-ve(Negative)** pin of **Relay**.

Connect **Node MCU** Supply (**3v3**) pin to **+ve(positive)** pin of **Relay**.

Connect **Node MCU** Digital pin (**D4**) to **Input** pin of **Relay**.

BASIC SHIELD:

Basic Shield is an input-output expansion board designed for use with micro controller board (here in connection with Node MCU) . It is used to provide a range of input/output devices. The basic I/O shield provides simple digital input devices such as switches and buttons, and digital output devices such as discrete LED's.

BUZZER:

A **buzzer** or **beeper** is an audio signalling device, which may be mechanical, electromechanical, or piezoelectric (*piezo* for short). Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke .

A "piezo buzzer" is basically a tiny speaker that is connected directly to an Arduino.

"Piezoelectricity" is an effect where certain crystals will change shape when you apply electricity to them. By applying an electric signal at the right frequency, the crystal can make sound.

From the Arduino IDE, we can make sounds with a buzzer by using **tone**. We should tell which pin the buzzer is on, what frequency (in Hertz, Hz) you want, and how long (in milliseconds) you want it to keep making the tone.When subjected to an alternating electric field they stretch or compress, in accordance with the frequency of the signal thereby producing sound.

The Connections are as follows:

Connect the Supply wire of the buzzer to the any **Digital Pin** of the N**ode MCU** then connect the Ground wire of the buzzer to any **Ground Pin** on the Node MCU.

The complete code of our project is written in“embedded C”(coding language) in ARDUINO IDE is as follows:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <Servo.h>
#include "DHT.h"

Servo myservodoor;
void callback(char* topic, byte* payload, unsigned int payloadLength);
void building();
const char* host = "api.msg91.com";

// CHANGE TO YOUR WIFI CREDENTIALS
const char* ssid = "GUEST";
const char* password = "12345678";


// CHANGE TO YOUR DEVICE CREDENTIALS AS PER IN IBM BLUMIX

#define ORG "4usw7n"
#define DEVICE_TYPE "sakiladev"
#define DEVICE_ID "3579"
#define TOKEN "0123456789" //  Authentication Token OF THE DEVICE

//  PIN DECLARATIONS FOR MOTORS


#define DHTPIN D2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
String data3;
String data="";
String payload1;
String command;
```

```
int sdata=0;
int pos;
long duration;
int distance;
int firemsg=0;
int fan;
int publishInterval = 5000; // 30 seconds
long lastPublishMillis;
void publishData();

const char publishTopic[] = "iot-2/evt/robo/fmt/json";
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char topic[] = "iot-2/cmd/home/fmt/String";// cmd  REPRESENT command type AND COMMAND
IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;


WiFiClient wifiClient;
PubSubClient client(server, 1883, callback, wifiClient);


void setup() {

  Serial.begin(9600);
  Serial.println();

  pinMode(D0,INPUT);
  pinMode(D4,OUTPUT);
  pinMode(D5,OUTPUT);
  pinMode(D6,OUTPUT);
  pinMode(D7,OUTPUT);
  pinMode(D8, OUTPUT);  // Sets the trigPin as an Output
  pinMode(D3, INPUT);   // Sets the echoPin as an Input
```

```
  myservodoor.attach(D1);

  dht.begin();
  wifiConnect();
  mqttConnect();
}

void loop() {

  if (millis() - lastPublishMillis > publishInterval)
  {
    publishData();
    lastPublishMillis = millis();
  }

  if (!client.loop()) {
    mqttConnect();
  }

}

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(topic)) {
    Serial.println("subscribe to cmd OK");
  } else {
    Serial.println("subscribe to cmd FAILED");
  }
}

void callback(char* topic, byte* payload, unsigned int payloadLength) {

  Serial.print("callback invoked for topic: "); Serial.println(topic);

  for (int i = 0; i < payloadLength; i++) {
   if(i==0){ command ="";}
    command += (char)payload[i];
  }
  Serial.println(command);
  if(command=="echo"){
    sdata=1;
  }
  else if(command=="alloff")
  {
    sdata=2;
  }
                                                                    else
if(command=="lighton"||command=="lightoff"||command=="fanon"||command=="fanoff"||comman
d=="venton"||command=="ventoff"||command=="heaton"||command=="heatoff")
```

```
  {
    sdata=3;
  }
  building();
}


void publishData()
{
  float h = dht.readHumidity();
  float t = dht.readTemperature();

   if (isnan(h) || isnan(t)) {
    h=64;
    t=30;
    Serial.println("Failed to read from DHT sensor!");

  }

  String payload = "{\"d\":{\"temperature\":";
  payload += t;
  payload += ","\"humidity\":";
  payload +=  h;
  payload += "}}";

  if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish OK");
  } else {
    Serial.println("Publish FAILED");
  }
}


 void building()
{
```

```
digitalWrite(D8, LOW); // Makes trigPin low
delay(0.000002);       // 2 micro second delay
digitalWrite(D8, HIGH);// tigPin high
delay(0.000010);      // trigPin high for 10 micro seconds
digitalWrite(D8, LOW); // trigPin low

duration = pulseIn(D3, HIGH); //Read echo pin, time in microseconds
distance= duration*0.034/2;         //Calculating actual/real distance

Serial.print("Distance = ");        //Output distance on arduino serial monitor
Serial.println(distance);

if(distance<30){
  myservodoor.write(90);
}
else{
  myservodoor.write(0);
}
delay(1000);



int fire=digitalRead(D0);

if(fire==1 && firemsg==0){
 digitalWrite(D6,HIGH);
 WiFiClient client;
 const int httpPort = 80;
 if (!client.connect(host, httpPort)) {
  Serial.println("connection failed");
  return;
 }

 // We now create a URI for the request
                                    String                                  url                                  =
"/api/sendhttp.php?mobiles=9032581183&authkey=280147AdghLYYhMg5cfb6065&route=4&sen
```

```
der=TESTIN&message=Fire%20at:%20https://goo.gl/maps/6ZMWoKbLHhPw6ueB7&country=91
";
 //this url frm url where we send the msg

 Serial.print("Requesting URL: ");
 Serial.println(url);

 // This will send the request to the server
 client.print(String("GET ") + url + " HTTP/1.1\r\n" +
              "Host: " + host + "\r\n" +
              "Connection: close\r\n\r\n");
 delay(10);

 // Read all the lines of the reply from server and print them to Serial
 while(client.available()){
   String line = client.readStringUntil('\r');
   Serial.print(line);
 }
 firemsg=1;

}
else if(fire==0){
   digitalWrite(D6,LOW);
   firemsg=0;
}
delay(1000);
Serial.print(firemsg);

if(sdata==2){
 digitalWrite(D1,LOW);
 digitalWrite(D4,LOW);
 digitalWrite(D3,LOW);
 digitalWrite(D5,LOW);
 myservo.write(0);
 Serial.println("Light is Switched Off");
```

```
}
else if(sdata==1){
  float h= dht.readHumidity();
  float t = dht.readTemperature();
  int ldr=analogRead(A0);
  int fire=digitalRead(D0);
  if (isnan(h) || isnan(t)) {
   h=64;
   t=30;
   Serial.println("Failed to read from DHT sensor!");
  }
  Serial.print("\n temperature:");
  Serial.print(t);
  Serial.print(",humidity:");
  Serial.print(h);
  Serial.print(",ldr:");
  Serial.print(ldr);
   Serial.print(",fire:");
  Serial.print(fire);
  Serial.print("\n");
  delay(1000);
  if(t>26){
   digitalWrite(D7,HIGH);
   digitalWrite(D3,LOW);
   }
 else{
   digitalWrite(D7,HIGH);
   digitalWrite(D3,HIGH);
 }
 if(h>60){
   digitalWrite(D4,HIGH);
 }
 else{
   digitalWrite(D4,LOW);
 }
```

```
 if(ldr>800){
  digitalWrite(D5,HIGH);
 }
 else{
  digitalWrite(D5,LOW);
 }
}

else {
  if(command == "fanon"){
    digitalWrite(D7,HIGH);
  }
  if(command == "fanoff"){
    digitalWrite(D7,LOW);
  }
  if(command == "heaton"){
    digitalWrite(D3,HIGH);
  }
  if(command == "heatoff"){
    digitalWrite(D3,LOW);
  }
  if(command == "venton"){
    digitalWrite(D4,HIGH);
  }
  if(command == "ventoff"){
    digitalWrite(D4,LOW);
  }
  if(command == "lighton"){
    digitalWrite(D5,HIGH);
  }


  if(command == "lightoff"){
    digitalWrite(D5,LOW);
  }
```

```
    }
}
```

SCREENSHOTS OF THE APP: