***Green House Monitoring & Control  System Using IBM Watson***

                      By:

                           Sanjana Reddy Nellipalli

                           T. Aneesha

                           Maanvi Keesara

*Abstract:*

Greenhouse environment, used to grow plants under controlled climatic

conditions for efficient production, forms an important part of the agriculture

and horticulture sectors. Appropriate environmental conditions are necessary

for optimum plant growth, improved crop yields, and efficient use of water and

other resources. Automating the data acquisition process of the soil conditions

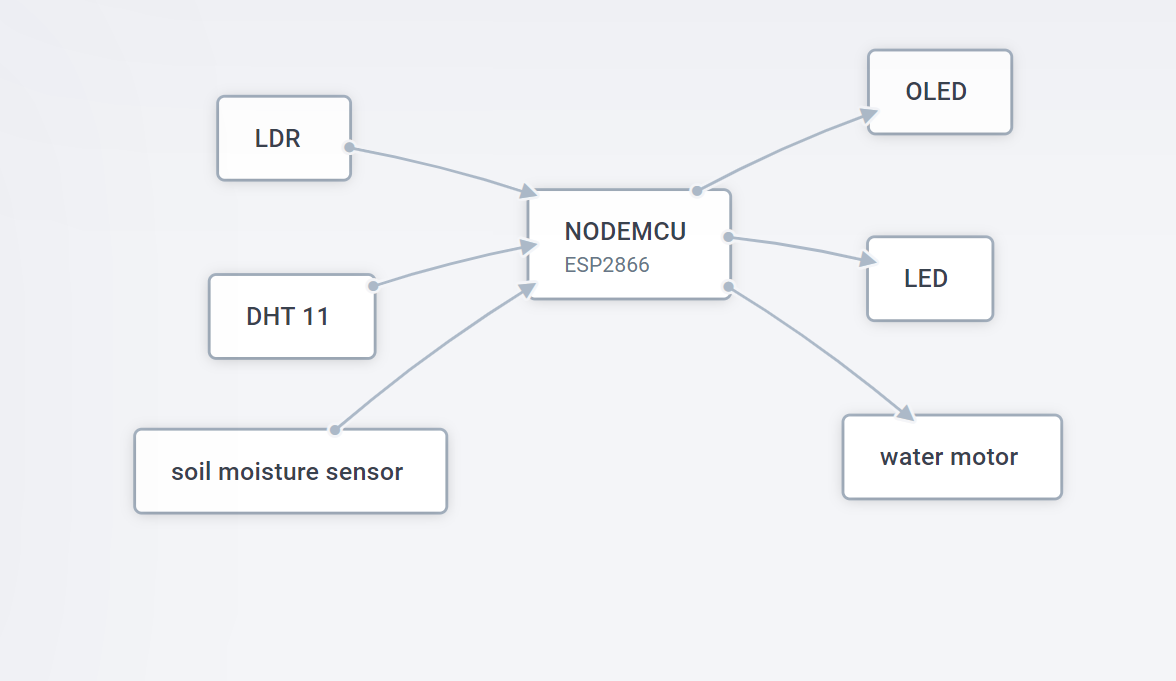
and various climatic parameters that govern plant growth allows information to

be collected with less labour requirements. In this device we will be monitoring the soil moisture levels, the amount of sunlight, humidity and the temperature of the greenhouse with the help of sensors, which will assist us in optimising the conditions to make it an ideal place for plant growth. The values that are obtained from the sensor are then used in our code with which we switch on the light and sprinklers whenever necessary along with that with the IOT platform we give the user an advantage of getting the humidity and temperature conditions along with a function to control the devices in the greenhouse with the help of an app.

*Problem Statement:*

A green House requires a set of controlled temperature conditions which is difficult to do manually it requires a lot of skilled manual labour, which is not only expensive but also difficult to find. Another major problem is the excess requirement of resources.

*Working:*



Using the Arduino software, a code was developed to control the working of all the sensors and equipment explained below. IBM cloud was used as a communication device in between the user and the processors

The system consists of 3 subsystems in it and they are:

I. Temperature and Humidity monitoring

II. Light intensity monitoring and control system

III. Soil moisture monitoring and control system

The system’s temperature and humidity are monitored by the DHT11 sensor and the temperature and humidity change are displayed on the OLED screen for user reference.

Other than that, the user is notified if there is a drastic change in climatic conditions in the green house through a message notification and whenever necessary user can check the temperature and humidity with the help of the app.

The light control system controls the light falling on the greenhouse. When there is not enough light the LDR detects this, and the light bulbs are switched ON. When there is lighter the light bulbs are turned OFF. But at night the system will automatically get turned OFF because if there is light for the plants at all times it is not ideal. Another advantage the user has is he/she can completely switch off or switch on the lights with the help of the app.

The soil moisture sensor detects whenever the moisture level in the soil is less and notifies and automatically switches the water motor on to sprinkle water to the plants. The user can also choose to operate the sprinkler whenever he/she wishes to with the help of the app.

*Components:*

DHT11:



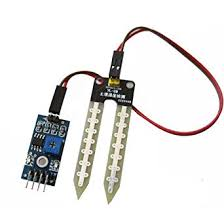
The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). It’s fairly simple to use, but requires careful timing to grab data.

LDR:



A Light Dependent Resistor (LDR) or a photoresistor 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.

Soil moisture sensor:



Soil moisture sensors measure the volumetric [water content](https://en.wikipedia.org/wiki/Water_content) in [soil](https://en.wikipedia.org/wiki/Soil). Since the direct [gravimetric measurement](https://en.wikipedia.org/wiki/Gravimetric_analysis) of free soil moisture requires removing, drying, and weighting of a sample, soil moisture sensors measure the volumetric water content indirectly by using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with [neutrons](https://en.wikipedia.org/wiki/Neutron), as a proxy for the moisture content.

OLED:



An organic light-emitting diode (**OLED**) is a light-emitting diode (LED) in which the emissive electroluminescent layer is a film of organic compound that emits light in response to an electric current. This device displays the humidity and temperature.

LED light:



It is a small light bulb which emits light. This LED turns on only when there is insufficient light in the greenhouse.

Water pump:



water pump is a machine that delivers or pressurizes a liquid. It is connected to a tube which sprays water whenever there is insufficient soil moisture.

NODE MCU(ESP8266):



NodeMCU is an open source [IoT](https://en.wikipedia.org/wiki/Internet_of_Things) platform. It includes [firmware](https://en.wikipedia.org/wiki/Firmware) which runs on the [ESP8266](https://en.wikipedia.org/wiki/ESP8266) [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi) [SoC](https://en.wikipedia.org/wiki/System_on_a_chip) from [Espressif Systems](https://en.wikipedia.org/w/index.php?title=Espressif_Systems&action=edit&redlink=1), 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](https://en.wikipedia.org/wiki/Lua_(programming_language)) 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

*Code:*

#include <ESP8266WiFi.h>    //--------  
#include <PubSubClient.h>  
   
//-------- Customise these values -----------  
const char\* ssid = "iPhone";  
const char\* password = "kohlibabyy18";  
  
   
#define ORG "bj4nz4"  
#define DEVICE\_TYPE "Project"  
#define DEVICE\_ID "5600"  
#define TOKEN "aneesharao13"  
//-------- Customise the above values --------  
   
const char publishTopic[] = "iot-2/evt/robo/fmt/json";  
char server[] = ORG ".[messaging.internetofthings.ibmcloud.com](http://messaging.internetofthings.ibmcloud.com/)";  
char topic[] = "iot-2/cmd/test/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;  
  
#include"DHT.h"  
#include<Wire.h>  
#include <Adafruit\_SSD1306.h>  
  
#define SSD1306\_LCDHEIGHT 32  
#define OLED\_ADDR   0x3C     
  
#define DHTPIN D6  
#define DHTTYPE DHT11  
#define LED D3  
//#define MOTOR1 6  
//#define MOTOR2 7  
  
#define SM D4  
#define M D5  
  
DHT dht(DHTPIN, DHTTYPE);  
Adafruit\_SSD1306 display(-1);  
  
void callback(char\* topic, byte\* payload, unsigned int payloadLength);  
String data3;  
String data="";  
int ldr;  
float sm;  
WiFiClient wifiClient;  
PubSubClient client(server, 1883, callback, wifiClient);  
  
int publishInterval = 5000; // 30 seconds  
long lastPublishMillis;  
void publishData();  
  
void setup() {  
  // put your setup code here, to run once:  
Serial.begin(9600);  
  Serial.println();  
    dht.begin();  
  display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR);  
  pinMode(LED, OUTPUT);  
  pinMode(M, OUTPUT);  
  wifiConnect();  
  mqttConnect();  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
    
 delay(5000);  
  display.clearDisplay();  
  
  
  float h=dht.readHumidity();  
  float t=dht.readTemperature();  
  ldr=analogRead(A0);  
  sm=digitalRead(SM);  
  
    
  //if(sm>1000)  
  //{  
    //digitalWrite(MOTOR1, LOW);  
    //digitalWrite(MOTOR2, HIGH);  
  //}  
  
  
  Serial.print("humidity: ");  
  Serial.println(h);  
  Serial.print("temperature:");  
  Serial.println(t);  
  Serial.print("moisture:");  
  Serial.println(sm);  
  
  display.setTextSize(1);//range: 1to 8  
  display.setTextColor(WHITE, BLACK);  
  display.setCursor(0,1);  
  display.print("humidity: ");  
  display.println(h);  
  display.print("temperature:");  
  display.println(t);  
  display.print("moisture:");  
  display.println(sm);  
  display.display();  
    
 if (millis() - lastPublishMillis > publishInterval)  
  {  
    PublishData(t,h,sm);  
    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(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++) {  
    //Serial.print((char)payload[i]);  
    data3 += (char)payload[i];  
  }  
    
  Serial.print("data: "+ data3);  
  control\_func();  
  data3 = "";  
}  
void control\_func()  
{  
  if(data3=="lton")  
  {  
    digitalWrite(LED, HIGH);  
  }  
  else if(data3="ltoff")  
  {  
    digitalWrite(LED, LOW);  
  }  
  else if(data3="ltauto")  
  {  
    analogWrite(LED, ldr);  
  }  
  else if(data3="mon")  
  {  
    digitalWrite(M, HIGH);  
  }  
  else if(data3="moff")  
  {  
    digitalWrite(M, LOW);  
  }  
  else if(data3="mauto")  
  {  
    if(sm==0)  
    {  
      digitalWrite(M, LOW);  
    }  
    else  
    {  
      digitalWrite(M, HIGH);  
    }  
  }  
  
}  
void PublishData(float temp, float humid, float moisture){  
 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+="," "\"soilmoisture\":";  
  payload += moisture;  
  payload += "}}";  
 Serial.print("Sending payload: ");  
 Serial.println(payload);  
    
 if (client.publish(topic, (char\*) payload.c\_str())) {  
 Serial.println("Publish ok");  
 } else {  
 Serial.println("Publish failed");  
 }  
}

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

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

// CHANGE TO YOUR WIFI CREDENTIALS

const char\* ssid = "iPhone";

const char\* password = "password5600";

#define ORG "bj4nz4"

#define DEVICE\_TYPE "Project"

#define DEVICE\_ID "5600"

#define TOKEN "aneesharao13"

const char publishTopic[] = "iot-2/evt/robo/fmt/json";

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";

char topic[] = "iot-2/cmd/test/fmt/String";// cmd REPRESENT command type AND COMMAND IS TEST OF FORMAT STRING

char topic1[]="iot-2/cmd/auto/fmt/String";

char authMethod[] = "use-token-auth";

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE\_TYPE ":" DEVICE\_ID;

const char\* host = "api.msg91.com";

#include "DHT.h"

#define DHTPIN D3

#define DHTTYPE DHT11

DHT dht (DHTPIN, DHTTYPE);

float temperature;

int humidity;

int soil;

int moisture;

String data3;

int light;

String data="";

boolean flag;

#define LED D4

#define SM D5

#define SPRINKLER D6

long publishInterval = 15000;

long lastPublishMillis;

void publishData();

long notificationInterval=600000;

long lastNotificationHumidity;

long lastNotificationTemperature;

WiFiClient wifiClient;

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

void setup() {

// put your setup code here, to run once:

wifiConnect();

mqttConnect();

Serial.begin(115200);

pinMode(LED, OUTPUT);

pinMode(SPRINKLER, OUTPUT);

}

void loop() {

// put your main code here, to run repeatedly:

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(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++) {

//Serial.print((char)payload[i]);

data3 += (char)payload[i];

}

Serial.print("data: "+ data3);

if(data3=="lton")

digitalWrite(LED, HIGH);

flag=0;

if(data3=="ltoff")

digitalWrite(LED, LOW);

flag=0;

if(data3=="ltauto")

analogWrite(LED, analogRead(A0));

flag=1;

if(data3=="mon")

digitalWrite(SPRINKLER, HIGH);

if(data3=="moff")

digitalWrite(SPRINKLER, LOW);

if(data3=="mauto")

{

if(moisture==0)

{

digitalWrite(SPRINKLER, LOW);

}

if(moisture==1)

{

digitalWrite(SPRINKLER, HIGH);

}

}

if(flag==1&&data3="ltofft")

digitalWrite(LED, HIGH);

if(flag==1&&data3="ltautot")

analogWrite(LED, analogRead(A0));

data3 = "";

}

void publishData()

{

humidity = dht.readHumidity();

temperature = dht.readTemperature();

moisture=digitalRead(SM);

light=map(analogRead(A0), 0, 1023, 0, 100);

if(moisture==0)

soil=1;

else

soil=0;

if (isnan(humidity) || isnan(temperature)) {

Serial.println("Failed to read from DHT sensor!");

return;

}

String payload = "{\"d\":{\"temperature\":";

payload += temperature;

payload += ",""\"humidity\":";

payload += humidity;

payload += ",""\"soilMoisture\":";

payload += soil;

payload += ",""\"avaliableLightPercentage\":";

payload += light;

payload += "}}";

if(humidity>40&&(millis() - lastNotificationHumidity > notificationInterval))

{

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?route=4&sender=TESTIN&mobiles=7093308895&authkey=282040Ad6VPGNU5d0c5cd6&message=Humidity very high, check Greenhouse&country=91";

// 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");

}

if(temperature>40&&(millis() - lastNotificationTemperature > notificationInterval))

{

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?route=4&sender=TESTIN&mobiles=7093308895&authkey=282040Ad6VPGNU5d0c5cd6&message=Temperature very high, check Greenhouse&country=91";

// 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");

}

Serial.print("\n");

Serial.print("Sending payload: "); Serial.println(payload);

if (client.publish(publishTopic, (char\*) payload.c\_str())) {

Serial.println("Publish OK");

} else {

Serial.println("Publish FAILED");

}

}