IOT ENABLED SMART FARMING APPLICATION

Sprint Delivery – 1

Date	12 November 2022
Team ID	PNT2022TMID24904
Project Name	SmartFarmer - IoT Enabled Smart Farming Application

1. Introduction :

The main aim of this project is to help farmers automate their farms by providing them with a Web App through which they can monitor the parameters of the field like Temperature, soil moisture, humidity and etc and control the equipment like water motor and other devices remotely via internet without their actual presence in the field.

2. Problem Statement :

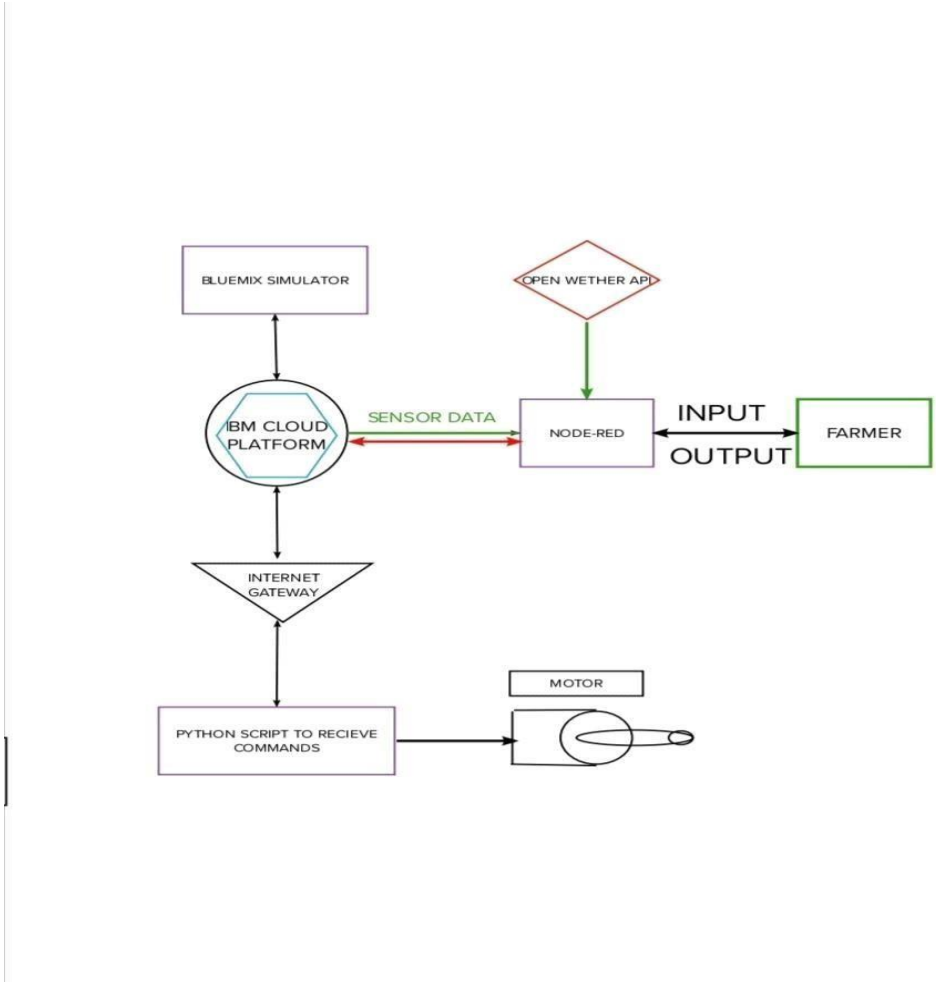
Farmers are to be present at farm for its maintenance irrespective of the weather conditions. They have to ensure that the crops are well watered and the farm status is monitored by them physically. Farmer have to stay most of the time in field in order to get a good yield. In difficult times like in the presence of pandemic also they have to work hard in their fields risking their lives to provide food for the country.

3. Proposed Solution:

In order to improve the farmer's working conditions and make them easier, we introduce IoT services to him in which we use cloud services and internet to enable farmer to continue his work remotely via internet. He can monitor the field parameters and control the devices in farm.

Block Diagram :

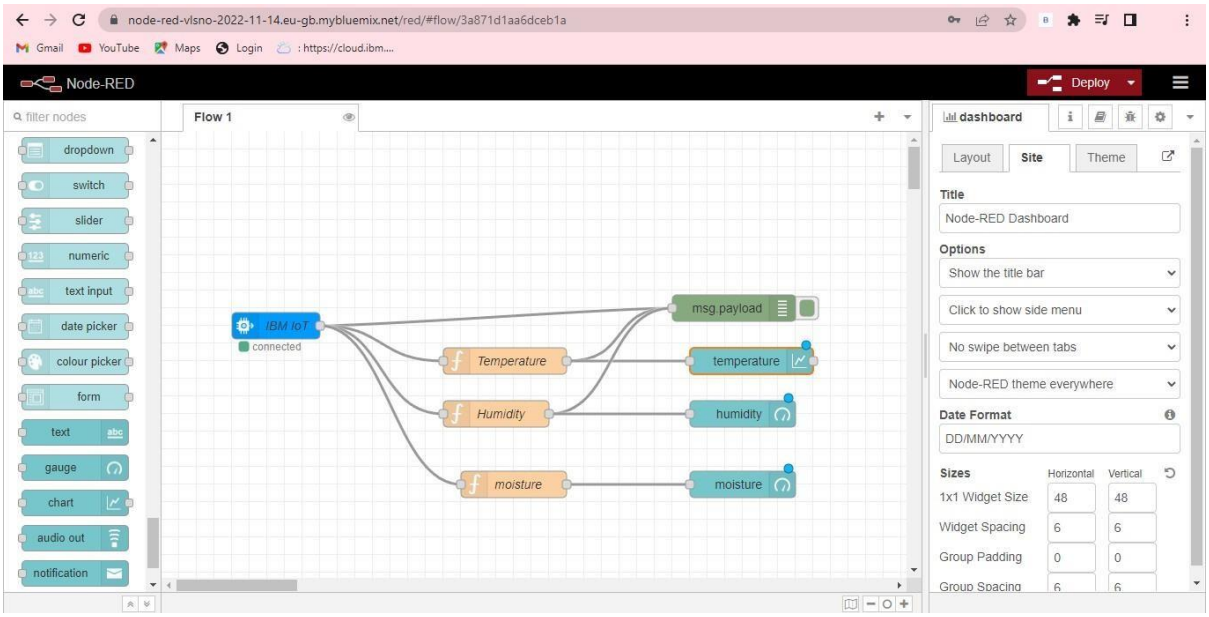
In order to implement the solution , the following approach as shown in the block diagram is used



Required Software Installation:

4.2.A Node-Red :

Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs 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.



Installation :

- First install npm/node.js
- Open cmd prompt
- Type => npm install node-red

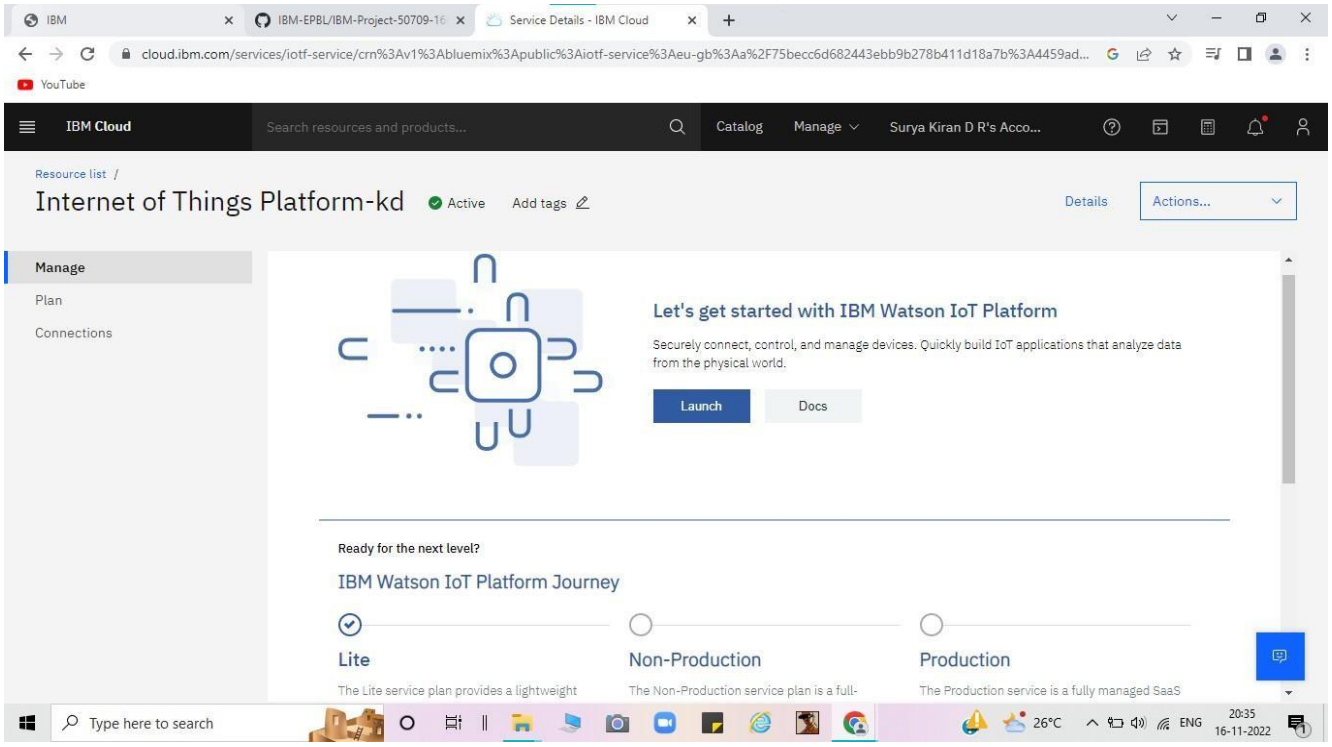
To run the application :

- Open cmd prompt
- Type=>node-red
- Then open http://localhost:1880/ in browser

Installation of IBM IoT and Dashboard nodes for Node-Red
In order to connect to IBM Watson IoT platform and create the Web App UI these nodes are required 1. IBM IoT node 2. Dashboard node

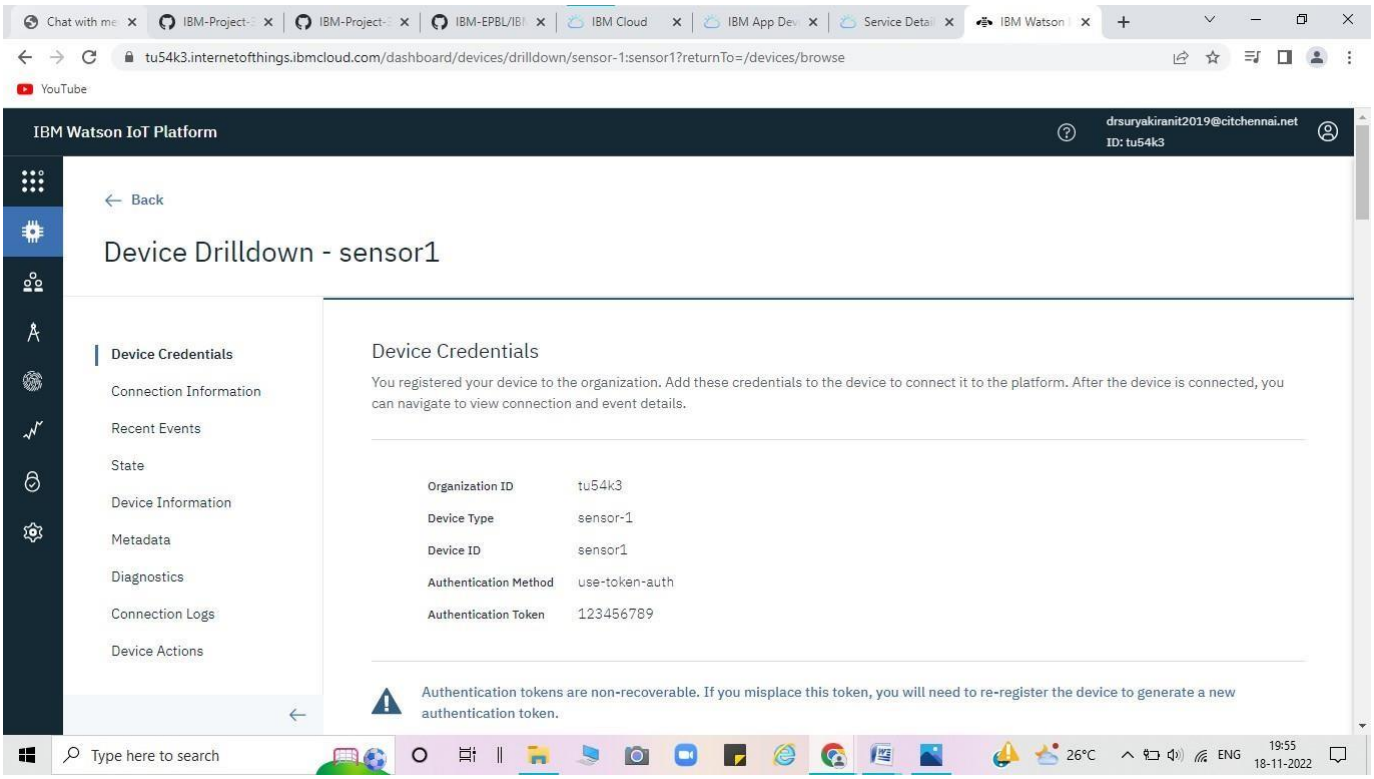
4.2.B IBM Watson IoT Platform:

A fully managed, cloud-hosted service with capabilities for device registration, connectivity, control, rapid visualization and data storage. IBM Watson IoT Platform is a managed, cloud-hosted service designed to make it simple to derive value from your IoT devices.



Steps to configure:

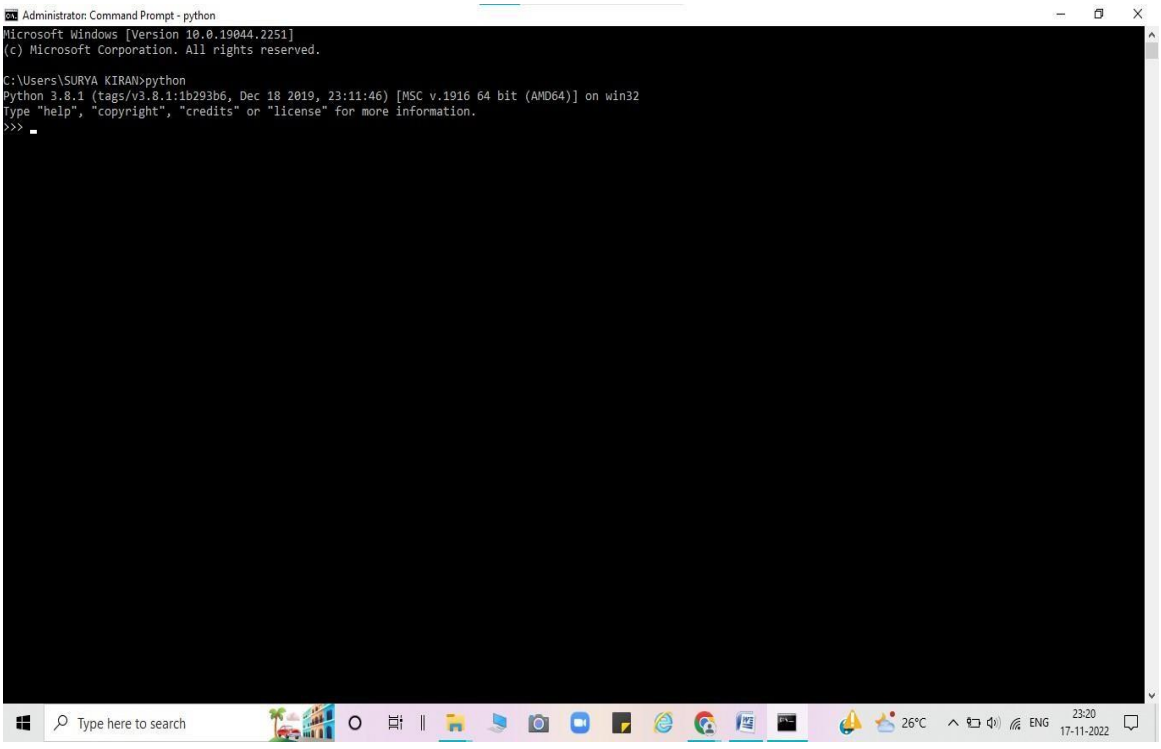
- Create an account in IBM cloud using your email ID
- Create IBM Watson Platform in services in your IBM cloud account
- Launch the IBM Watson IoT Platform
- Create a new device
- Give credentials like device type, device ID, Auth. Token
- Create API key and store API key and token elsewhere.



4.2.C Python IDE:

Install Python3 compiler

Install any python IDE to execute python scripts, in my case I used Spyder to execute the code.



Code:

```
import time
import sys

import ibmiotf.applicationimport
ibmiotf.device import random

 #Provide your IBM Watson Device Credentials

organization = "tu54k3"

deviceType = "sensor-1"

deviceId = "sensor1"
authMethod = "use-auth-
token"

authToken = "123456789"



def myCommandCallback(cmd):
        print("Command received: %s" % cmd.data['command'])print(cmd)



try:
         deviceOptions = {"org": organization, "type": deviceType,
"id": deviceId, "auth-method": authMethod, "auth-token":authToken}
         deviceCli = ibmiotf.device.Client(deviceOptions)
```

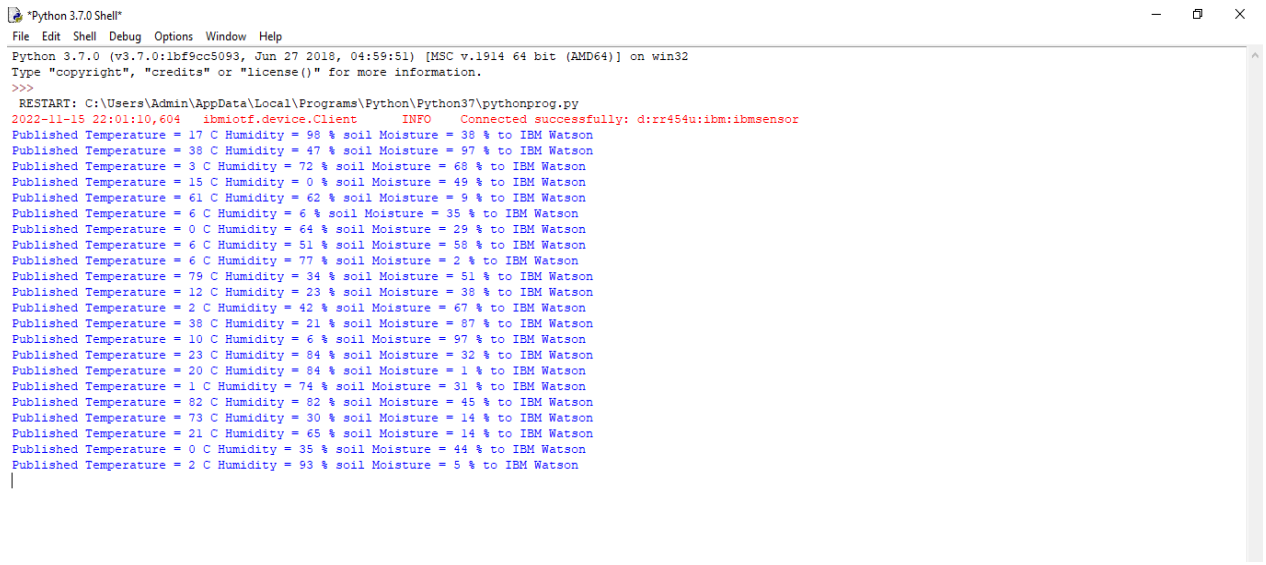
```
        #.....

except Exception as e:
        print("Caught exception connecting device: %s" % str(e))sys.exit()

# Connect and send a datapoint "hello" with value "world" intothe cloud as an
event of type "greeting" 10 times
deviceCli.connect()

while True:

            temperature=random.randint(0,100)
            humidity=random.randint(0,100) soil=
            random.randint(0,100)

            data = {'temperature' : temperature, 'humidity':humidity
,'soil':soil}
            #print data
            def myOnPublishCallback():
                print ("Published Temperature = %s C" % temperature,"Humidity = %s
%%" % humidity, "soil Moisture = %s %%"% soil,"toIBM Watson")

            success = deviceCli.publishEvent("IoTSensor", "json",data, qos=0,
on_publish=myOnPublishCallback)
            if not success:
                print("Not connected to IoTF")
            time.sleep(1)
```

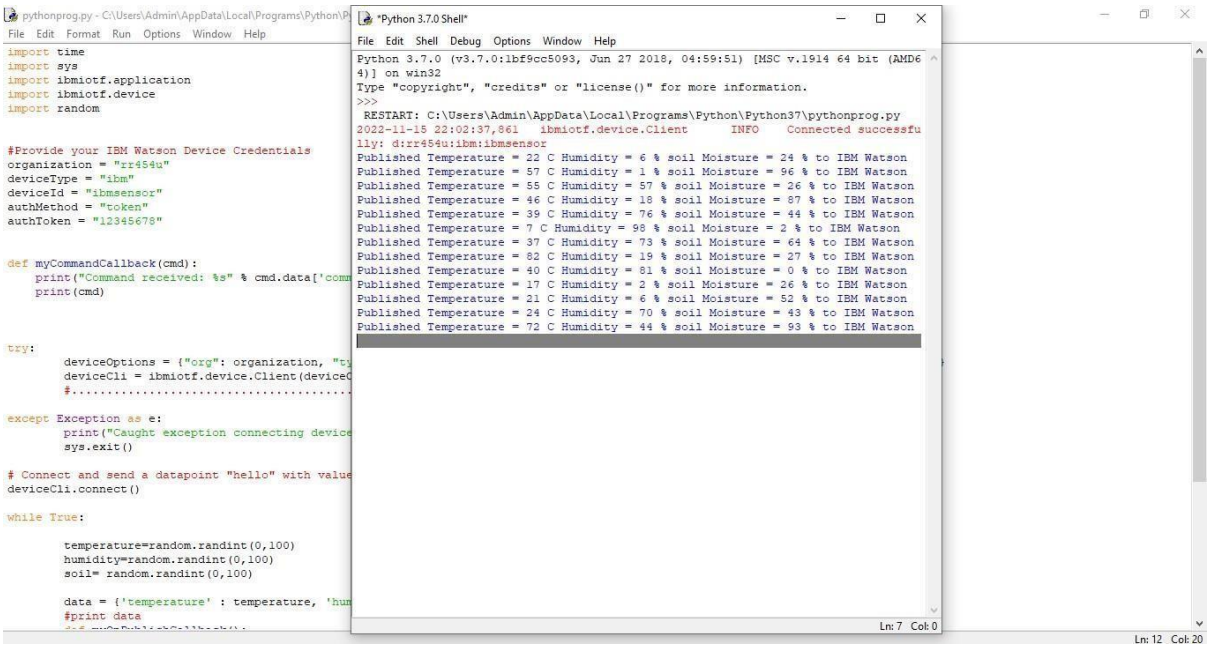
deviceCli.commandCallback =
myCommandCallback

Disconnect the device and application from the cloud

deviceCli.disconnect()

Simulation output in the python idle:



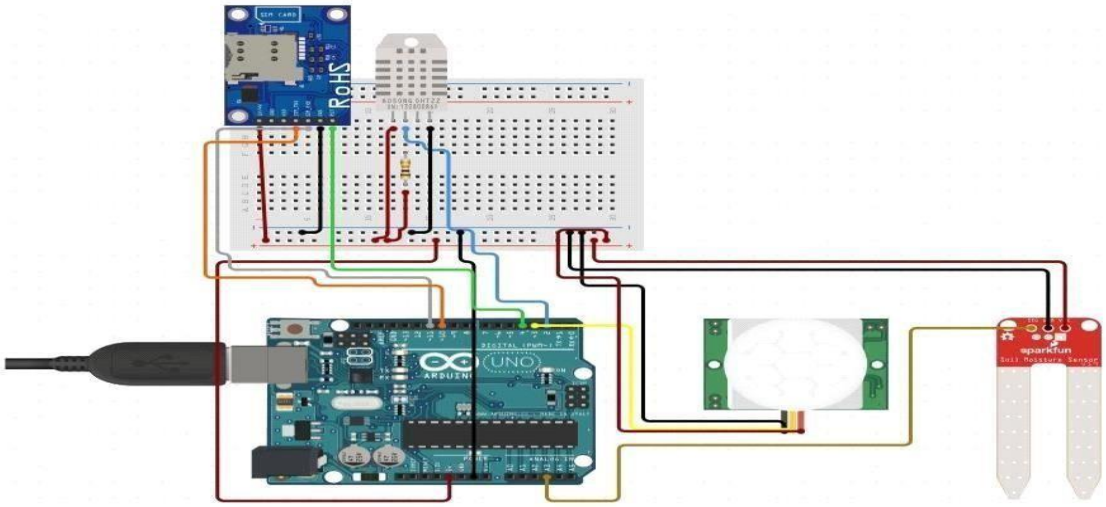


Aurdino code for C :

```
#include libraries
 #include <dht.h>
 #include <SoftwareSerial.h>

 //define pins
 #define dht_apin A0 // Analog Pin sensor is
connected    SoftwareSerial mySerial(7,8);//serial
port of gsm   const int sensor_pin = A1; // Soil
moisture sensor O/P pin   int pin_out = 9;
//allocate variables   dht DHT;
 int c=0;

 void setup()
 {
 pinMode(2, INPUT); //Pin 2 as INPUT
pinMode(3, OUTPUT); //PIN 3 as
OUTPUT
 pinMode(9, OUTPUT);//output for pump
 }
 void loop()
 {
   if (digitalRead(2) == HIGH)
   {
   digitalWrite(3, HIGH);   // turn the  LED/Buzz
ON     delay(10000); // wait for 100 msecond
digitalWrite(3, LOW);   // turn the LED/Buzz
OFF     delay(100);
   }
    Serial.begin(9600);
     delay(1000);
     DHT.read11(dht_apin);
//temprature     float
h=DHT.humidity;     float
t=DHT.temperature;
     delay(5000);
Serial.begin(9600);
```

```
      float
moisture_percentage;//moisture
int sensor_analog;
  sensor_analog = analogRead(sensor_pin);
  moisture_percentage = ( 100 - ( (sensor_analog/1023.00) * 100 ) );
float m=moisture_percentage;
  delay(1000);
  if(m<40)//pump
  {
  while(m<40)
  {
  digitalWrite(pin_out,HIGH);//open pump
sensor_analog = analogRead(sensor_pin);
  moisture_percentage = ( 100 - ( (sensor_analog/1023.00) * 100 ) );
m=moisture_percentage;
  delay(1000);
  }
  digitalWrite(pin_out,LOW);//closepump
  }
  if(c>=0)
  {

mySerial.begin(9600);
delay(15000);
Serial.begin(9600);
delay(1000);
Serial.print("\r");
  delay(1000);
  Serial.print("AT+CMGF=1\r");
delay(1000);
  Serial.print("AT+CMGS=\"+XXXXXXXXXX\"\r"); //replace X with
10 digit mobil e number        delay(1000);
  Serial.print((String)"update-
>"+(String)"Temprature="+t+(String)"Humidity="+h+(String)"Moist
ure="+m);    delay(1000);    Serial.write(0x1A);    delay(1000);
  mySerial.println("AT+CMGF=1");//Sets the GSM Module in Text
Mode    delay(1000);
  mySerial.println("AT+CMGS=\"+XXXXXXXXXX\"\r"); //replace X
 with 10 digit
mobile number
delay(1000);
```

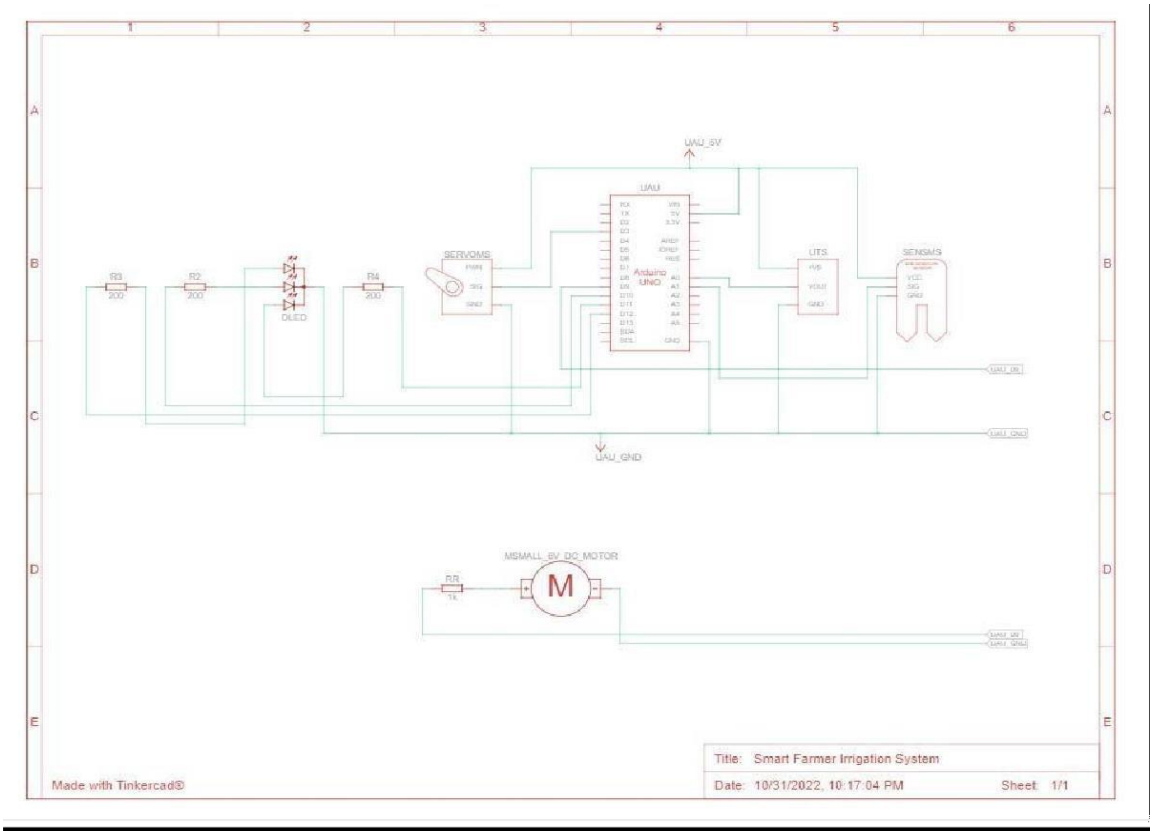
```
  mySerial.println((String)"update-
>"+(String)"Temprature="+t+(String)"Humidity="+h+(String)"Moistur
e="+m);// message format
  mySerial.println();
delay(100);
Serial.write(0x1A);
delay(1000);
  c++;

    }


 }
```



Components Used:

Name	Quantity	Component
UAU	1	Arduino Uno R3
SERVOMS	1	Positional Micro Servo
DLED	1	LED RGB
R2 R3 R4	3	200 Ω Resistor
SENSMS	1	Soil Moisture Sensor
MSmall 6V DC Motor	1	DC Motor
RR	1	1 kΩ Resistor
UTS	1	Temperature Sensor [TMP36]

Schematic View:



IoT Simulator:

In our project in the place of sensors we are going to use IoT sensor simulator which give random readings to the connected cloud.

The link to simulator:

https://watson-iot-sensor-simulator.mybluemix.net/

We need to give the credentials of the created device in IBM Watson IoT Platform to connect cloud to simulator.

OpenWeather API:

OpenWeatherMap is an online service that provides weather data. It provides current weather data, forecasts and historical data to more than 2 million customer.

Website link: https://openweathermap.org/guide

Steps to configure:

- Create account in OpenWeather o Find the name of your city by searching o Create API key to your account
- Replace “city name” and “your api key” with your city and API key in below red text

api.openweathermap.org/data/2.5/weather?q={city name}&appid={your api key}

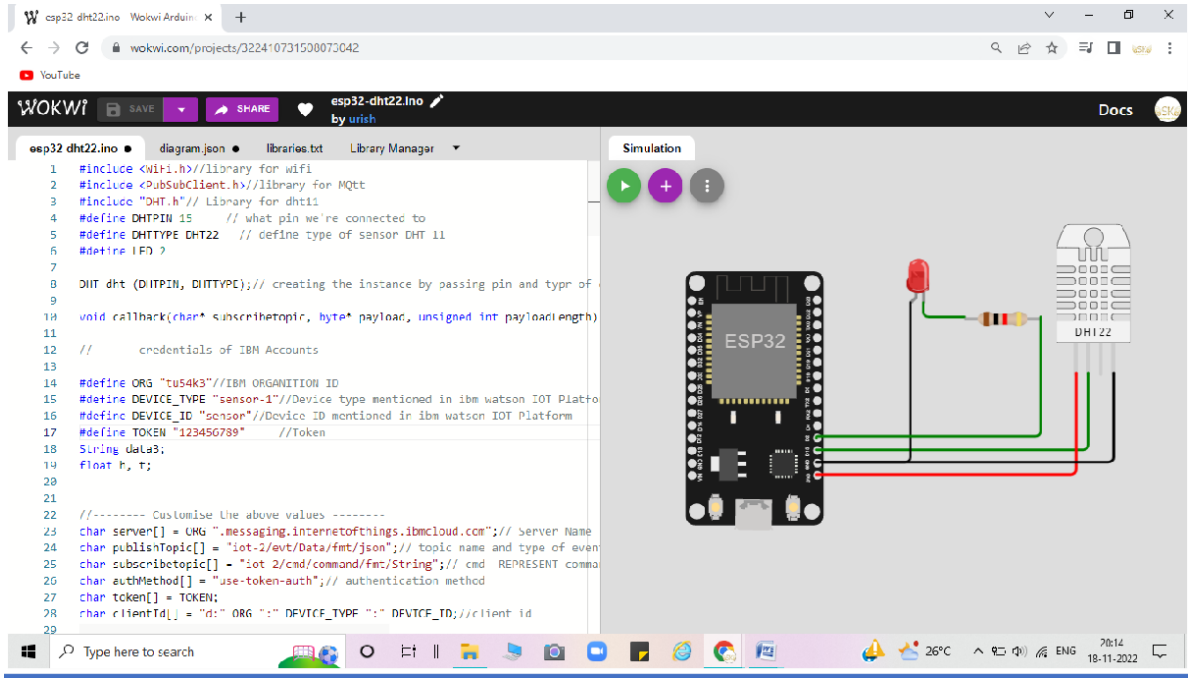
WOKWI Online Simulator ESP32 :

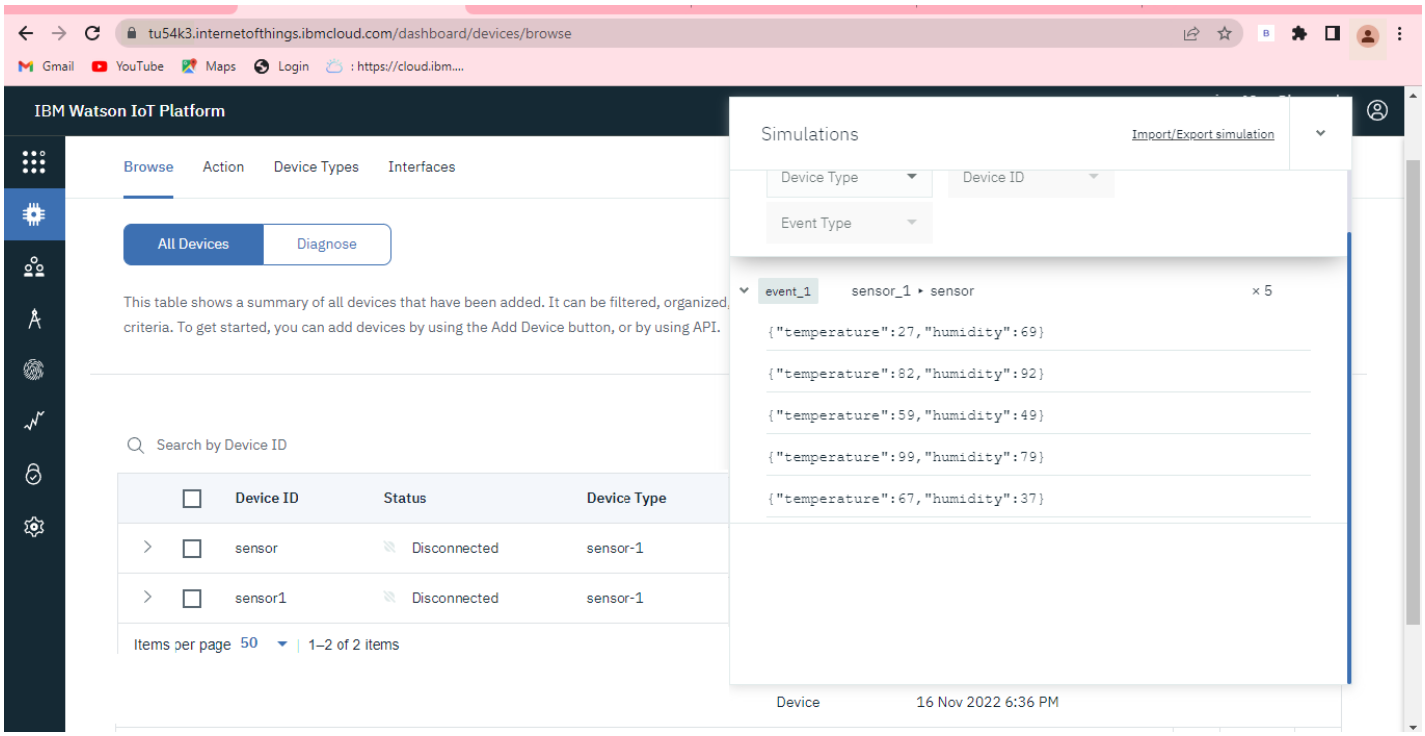
```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht11
#define DHTPIN 15     // what pin we're connected to
#define DHTTYPE DHT22   // define type of sensor DHT 11
#define LED 2

DHT dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr of dht connected

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

//-------credentials of IBM Accounts------

#define ORG "tu54k3"//IBM ORGANITION ID
#define DEVICE_TYPE "sensor-1"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "sensor"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "123456789"      //Token
String data3;
float h, t;


//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and format in which data to be send
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd REPRESENT command type AND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
```

```
//-------------------------------------------------------------------
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client id
by passing parameter like server id,portand wificredential


void setup()// configureing the ESP32
{
  Serial.begin(115200);
  dht.begin();
  pinMode(LED,OUTPUT);
  delay(10);
  Serial.println();
  wificonnect();
  mqttconnect();
}

void loop()// Recursive Function
{

  h = dht.readHumidity();
  t = dht.readTemperature();
  Serial.print("temp:");
  Serial.println(t);
  Serial.print("Humid:");
  Serial.println(h);

  PublishData(t, h);
  delay(1000);
  if (!client.loop()) {
    mqttconnect();
  }
}



/*.....................................retrieving to Cloud.............................. */

void PublishData(float temp, float humid) {
  mqttconnect();//function call for connecting to ibm
  /*
     creating the String in in form JSon to update the data to ibm cloud
  */
  String payload = "{\"temp\":";
  payload += temp;
  payload += "," "\"Humid\":";
  payload += humid;
  payload += "}";


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


  if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish ok");// if it sucessfully upload data on the cloud then it will
print publish ok in Serial monitor or else it will print publish failed
  } else {
    Serial.println("Publish failed");
  }
```

```
}


void mqttconnect() {
  if (!client.connected()) {
    Serial.print("Reconnecting client to ");
    Serial.println(server);
    while (!!!client.connect(clientId, authMethod, token)) {
      Serial.print(".");
      delay(500);
    }

     initManagedDevice();
     Serial.println();
  }
}
void wificonnect() //function defination for wificonnect
{
  Serial.println();
  Serial.print("Connecting to ");

  WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials to establish the
connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void initManagedDevice() {
  if (client.subscribe(subscribetopic)) {
    Serial.println((subscribetopic));
    Serial.println("subscribe to cmd OK");
  } else {
    Serial.println("subscribe to cmd FAILED");
  }
}

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

  Serial.print("callback invoked for topic: ");
  Serial.println(subscribetopic);
  for (int i = 0; i < payloadLength; i++) {
    //Serial.print((char)payload[i]);
    data3 += (char)payload[i];
  }
  Serial.println("data: "+ data3);
  if(data3=="lighton")
  {
Serial.println(data3);
digitalWrite(LED,HIGH);
  }
  else
  {
Serial.println(data3);
digitalWrite(LED,LOW);
```

```
  }
data3="";
}
```


esp32 dht22.ino
wokwi.com/projects/322410731500073042
WOKWI
SAVE
SHARE
esp32-dht22.ino
by urish
Docs
esp32 dht22.ino
diagram.json
libraries.txt
Library Manager
Simulation
ESP32
DHT22
Type here to search
26°C
ENG
20:14
18-11-2022


tu54k3.internetofthings.ibmcloud.com/dashboard/devices/browse
Gmail
YouTube
Maps
Login
: https://cloud.ibm....
IBM Watson IoT Platform
Browse
Action
Device Types
Interfaces
All Devices
Diagnose
This table shows a summary of all devices that have been added. It can be filtered, organized,
criteria. To get started, you can add devices by using the Add Device button, or by using API.
Search by Device ID
Device ID
Status
Device Type
sensor
Disconnected
sensor-1
sensor1
Disconnected
sensor-1
Items per page 50
1–2 of 2 items
Simulations
Import/Export simulation
Device Type
Device ID
Event Type
event_1
sensor_1 ▸ sensor
× 5
{"temperature":27,"humidity":69}
{"temperature":82,"humidity":92}
{"temperature":59,"humidity":49}
{"temperature":99,"humidity":79}
{"temperature":67,"humidity":37}
Device
16 Nov 2022 6:36 PM