Smart Farmer-IOT Enabled Smart FarmingProject

Development Phase

SPRINT DELIVERY-1

DATE	17 November 2022
Team ID	PNT2022TMID33827
Project Name	Smart Farmer-IOT Enabled Smart Farming Application
Leader Name	Ajitha S
Team Members Name	Augusta Blessy L Jenifer Gloria Daphne V Maheswari V

INTRODUCTION:

The main aim of this project is to minimize the involvement of farmers in every aspects of farming by invloving the usage of sensors for detecting the amount of temperature,humidity,soil moisture so that the farmers can measure everything effortlessly by their

smartphones and grow their plants effectively and get a lot of profits.

PROBLEM STATEMENT:

Farmers are not able to measure the amount of mositure,temperature accurately so without knowing them accurately they are watering the plants irrespective of knowing the water contents and this is leading to overwatering of crops and affecting the growth of plants.A major problem is that the farmers have to spend most of their life time in their farm inorder to monitor the growth of plants continuously to grow the plant effectively.

PROPOSED SOLUTION:

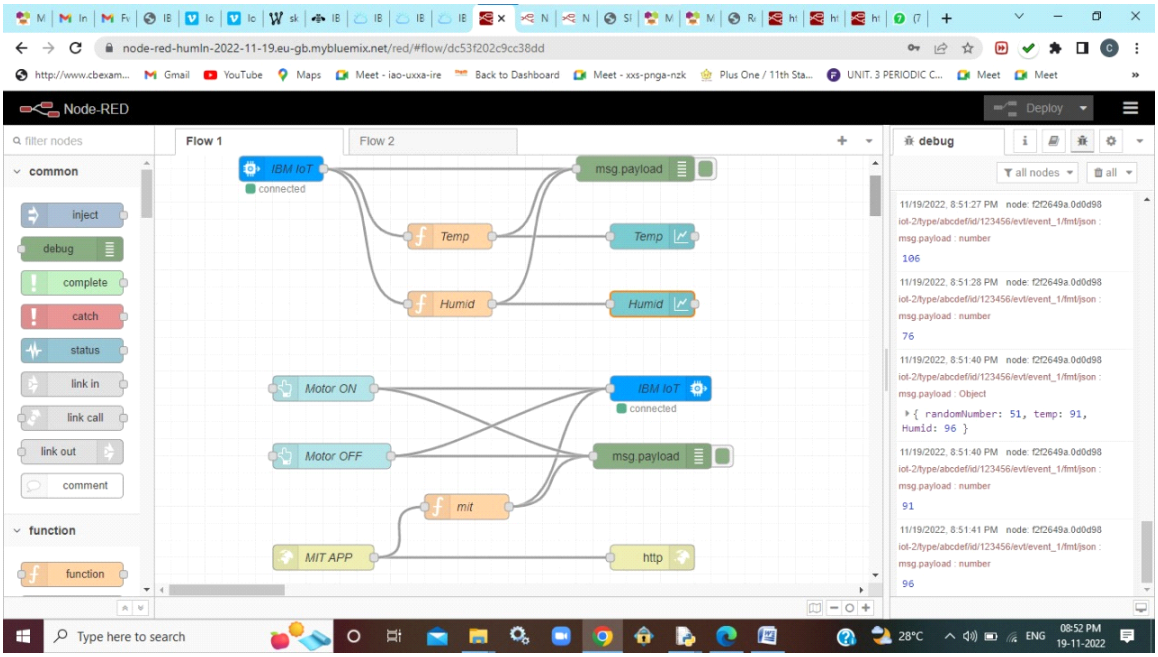
A suitable solution to this problem is invloving the usage of sensors to measure the temperature,humidity,soil moisture and it will be informed to the farmers through their mobile phones. This will help the farmers to grow plants effectively without the farmers staying all the time in their respective farms.

REQUIRED SOFTWARE INSTALLMENTS:

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 on your browser.

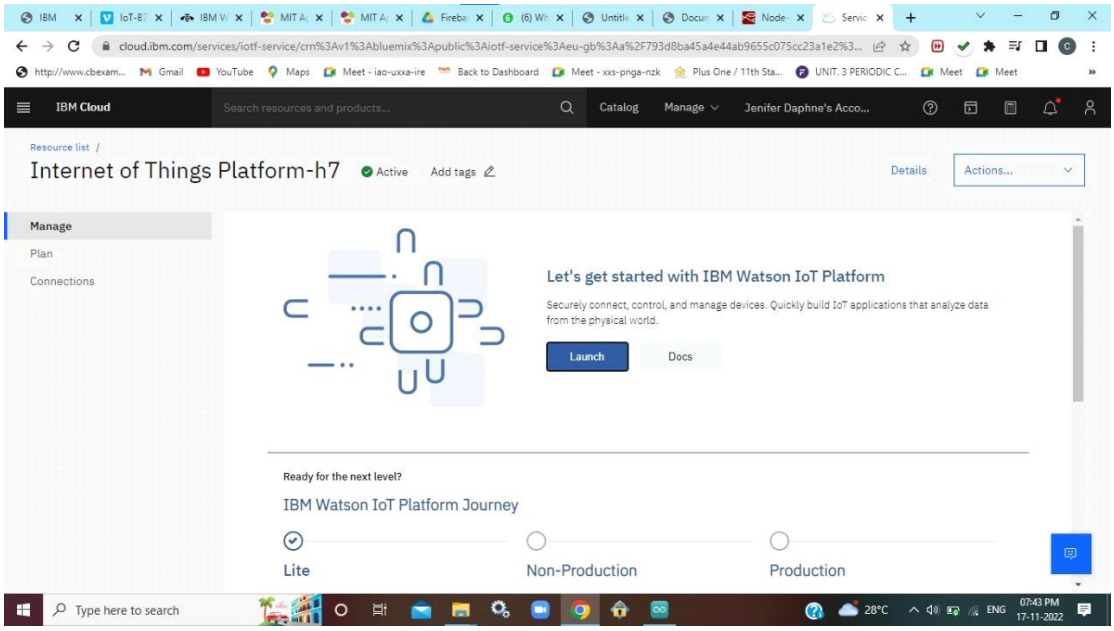
INSTALLATION OF IBM IOT AND DASHBOARD MODESFOR NODE-RED:

Inorder to connect to IBM Watson IOT platform and create the WEB UI these nodes are required.

- IBM IOT node.
- Dashboaard node.

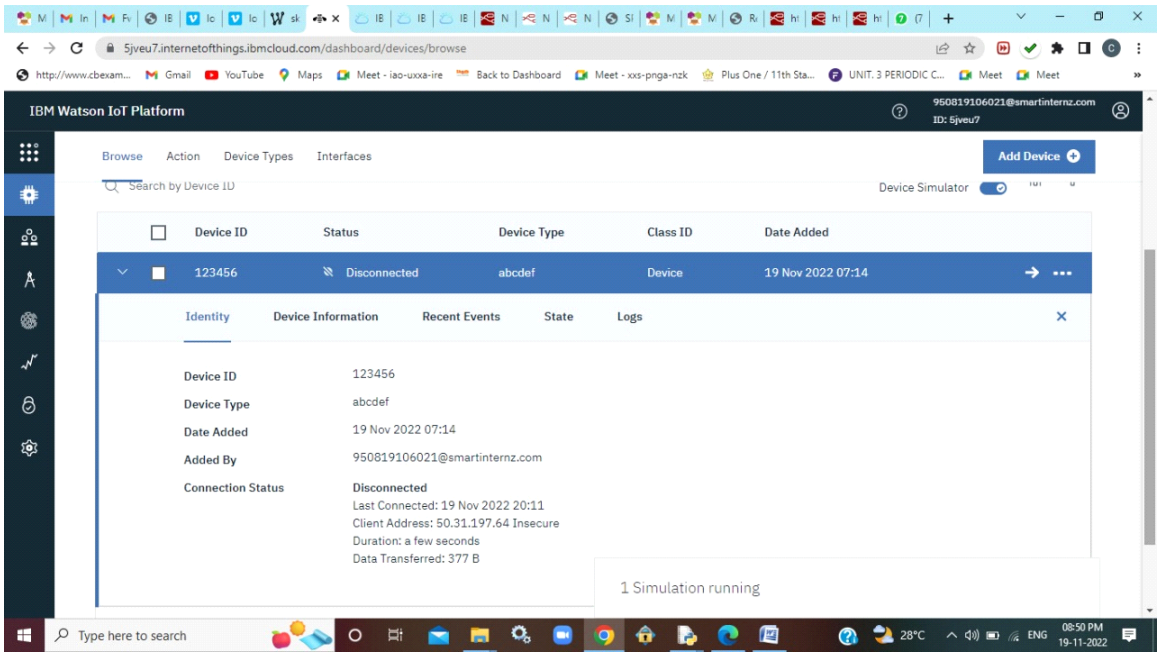
IBM WATSON IOT PLATFORM:

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:

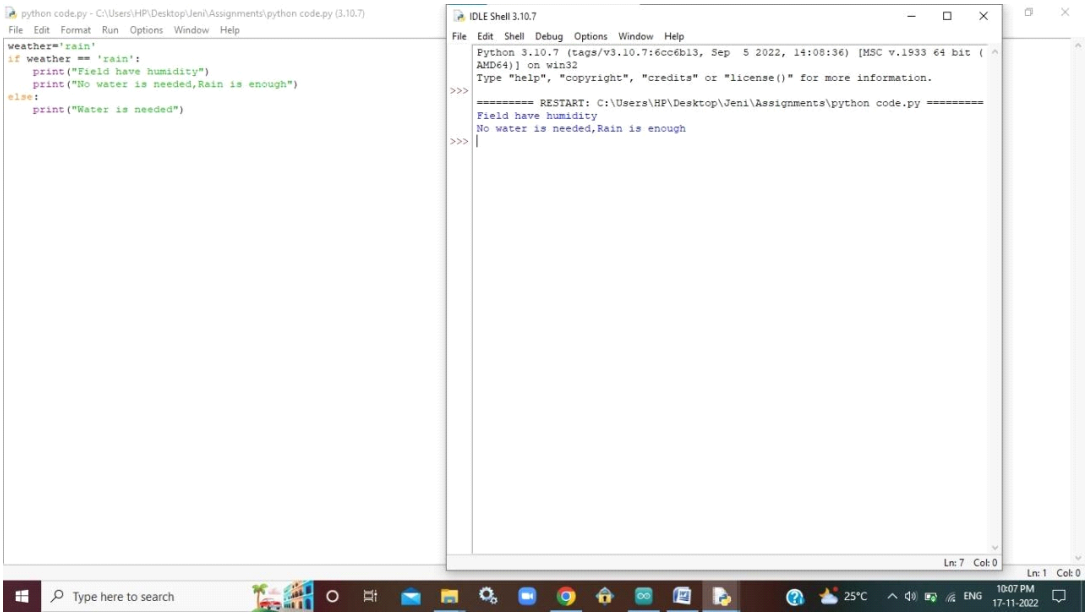
- Create an account in IBM cloud using your email ID.
- Create IBM Watson platform services in your IBM cloudAccount.
- Launch the IBM Watson IOT platform.Create a new device.
- Give credentials like device type,device ID,Auth.Token
- Create API key and store API key and token elsewhere.



PYTHON IDE:

Install python 3 compiler

Install and python IDE to execute python scripts in many case Iused command prompt to execute.



CODE:

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht11
#include <ESP32Servo.h>
#define DHTPIN 15     // what pin we're connected to
#define DHTTYPE DHT22   // define type of sensor DHT
11
#define LED 2
const int servoPin = 13;
DHT dht (DHTPIN, DHTTYPE);// creating the instance by
passing pin and typr of dht connected
Servo motor;
void callback(char* subscribetopic, byte* payload,
unsigned int payloadLength);

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

#define ORG "5jveu7"//IBM ORGANITION ID
```

```
#define DEVICE_TYPE "abcdef"//Device type mentioned
in ibm watson IOT Platform
#define DEVICE_ID "123456"//Device ID mentioned in
ibm watson IOT Platform
#define TOKEN "123456789"     //Token
String data3;
float h, t;
int pos;


//-------- 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);
  motor.attach(servoPin,500,2400);
  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);
  condition(h);
  PublishData(t, h);
  delay(2000);
  if (!client.loop()) {
    mqttconnect();
  }
}
void condition(float h)
{
  if(h>30)
  {
```

```
    digitalWrite(LED,LOW);
    Serial.println("Humidity Normal,No need of
Water");
    delay(500);
  }
  else
  {
    digitalWrite(LED,HIGH);
    Serial.println("Alert!!! Humidity Low, Required
Water ");
    delay(500);
  }
}


/.....................................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 ");
```

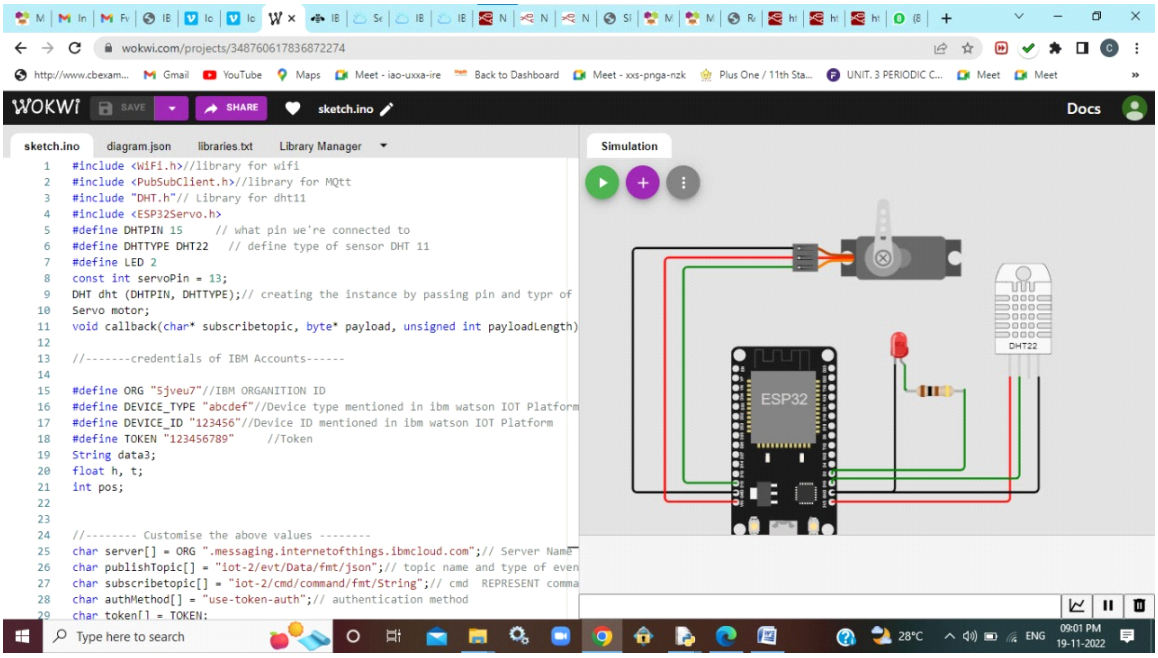
```
  WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi
credentials to establish the connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());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

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=="motoron")
  {
```

```
Serial.println(data3);
pos = 180;
motor.write(pos);
  }
  else
  {
  Serial.println(data3);
  pos=0;
  motor.write(pos);
  }
data3="";
}
[07:30, 11/19/2022] Jenifer: wokwi program
```



IOT SIMULATOR:

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

THE LINK TO SIMULATOR:

https://wokwi.com/projects/348760617836872274 We need to give the credentials of the created device in IBMWatson IOT platform to connect cloud to simulator.