# Project Development Phase

# Delivery of Sprint -1

| | |
|---|---|
| **Team ID** | PNT2022TMID08469 |
| **Project Name** | Smart Farmer-IOT Enabled Smart Farming Application |

In Sprint-1 we are going to develop the python code and Wokwi Online ESP32 Simulator and connecting to IBM Watson Platform

## 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.

## 4 . Software Requirements

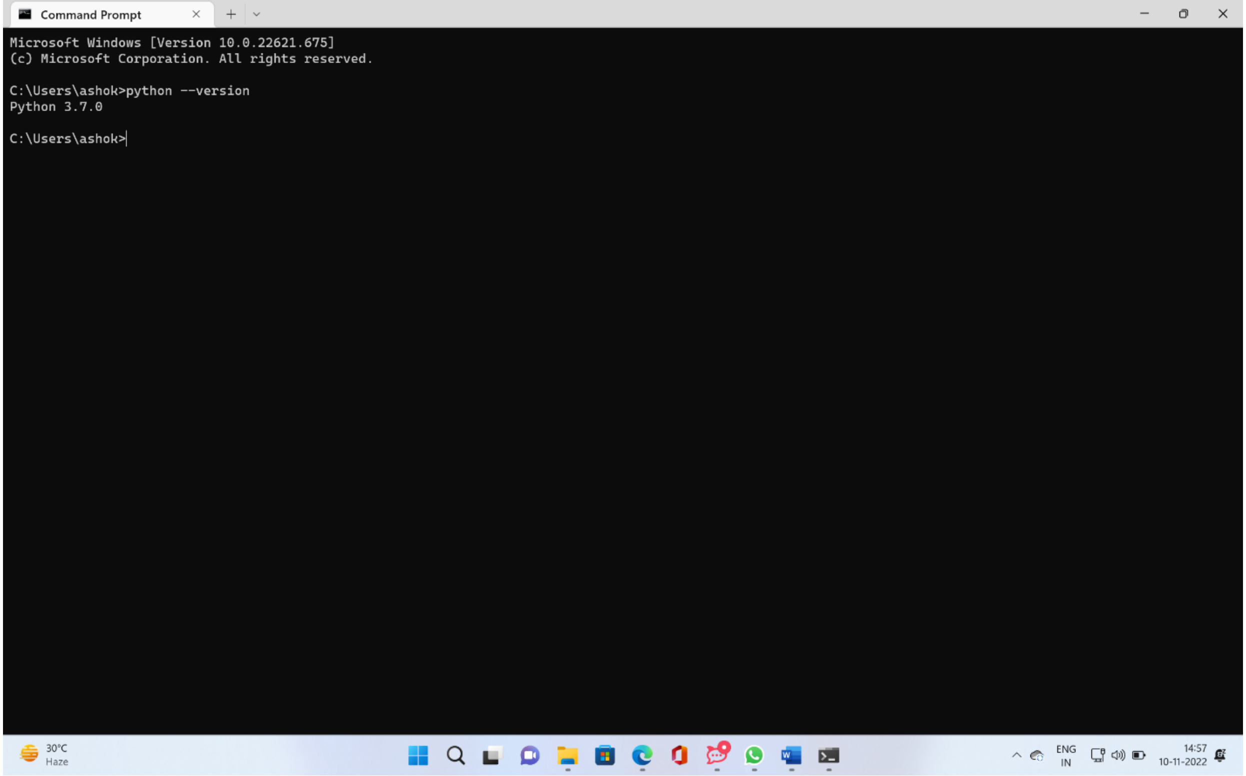
1.Python IDLE 3.7.0 (64-Bit)

2.IBM Watson Platform

3.IBM Node-Red

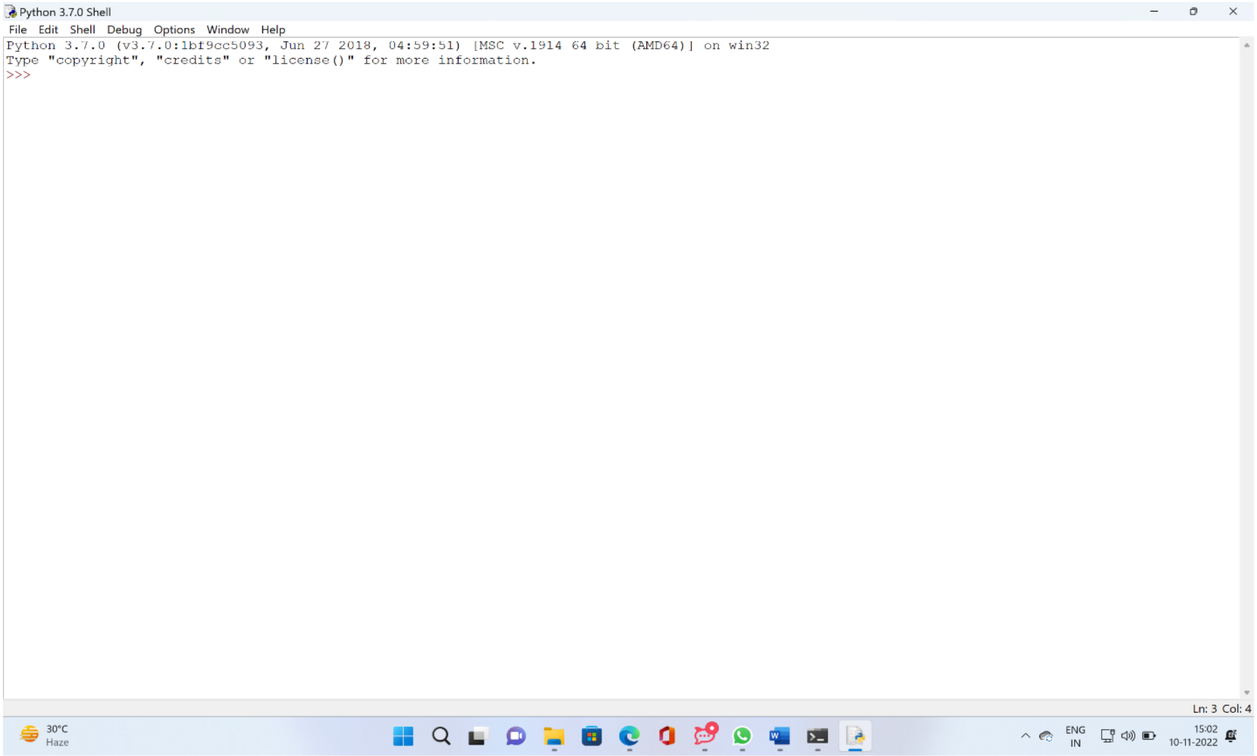
4. MIT App Inventor

First install the python 3.7.0 version idle . Go to command prompt and type python –version we can get version.



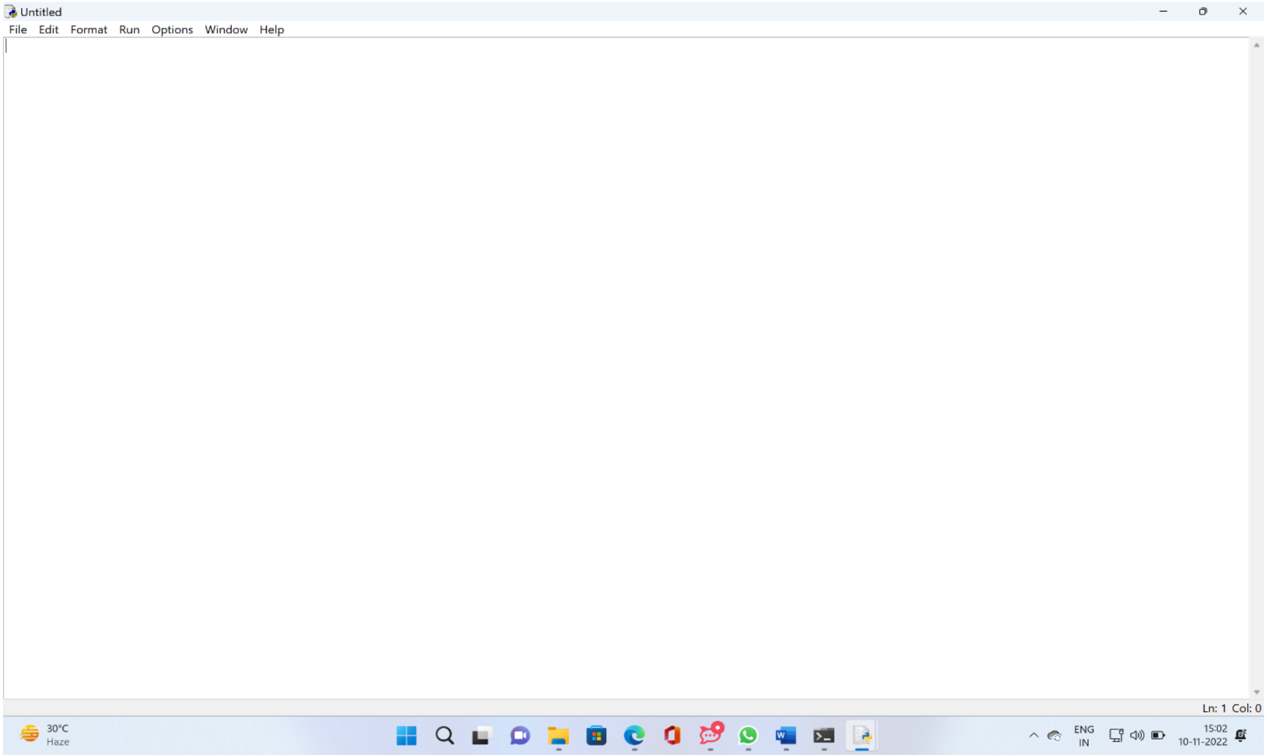
```
Microsoft Windows [Version 10.0.22621.675]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ashok>python --version
Python 3.7.0

C:\Users\ashok>
```

After that open python idle we can see python shell.

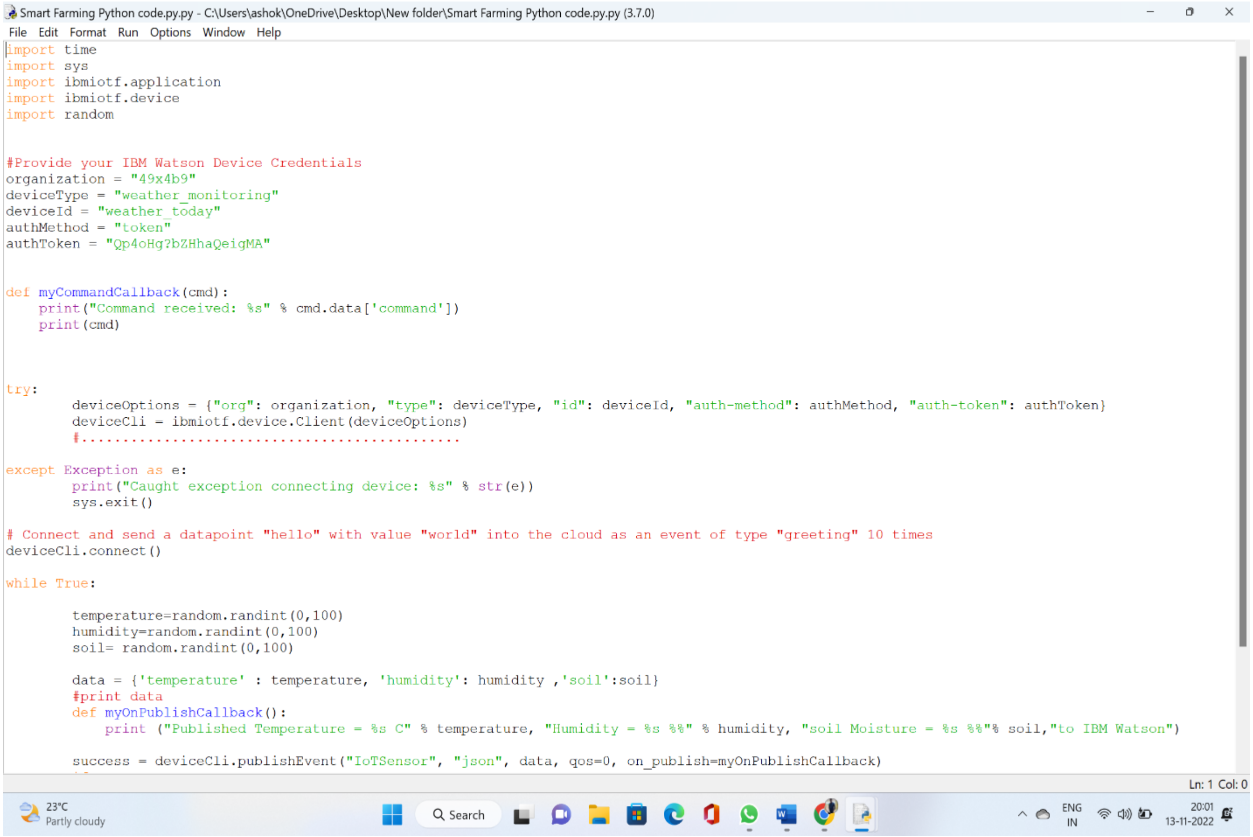


```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

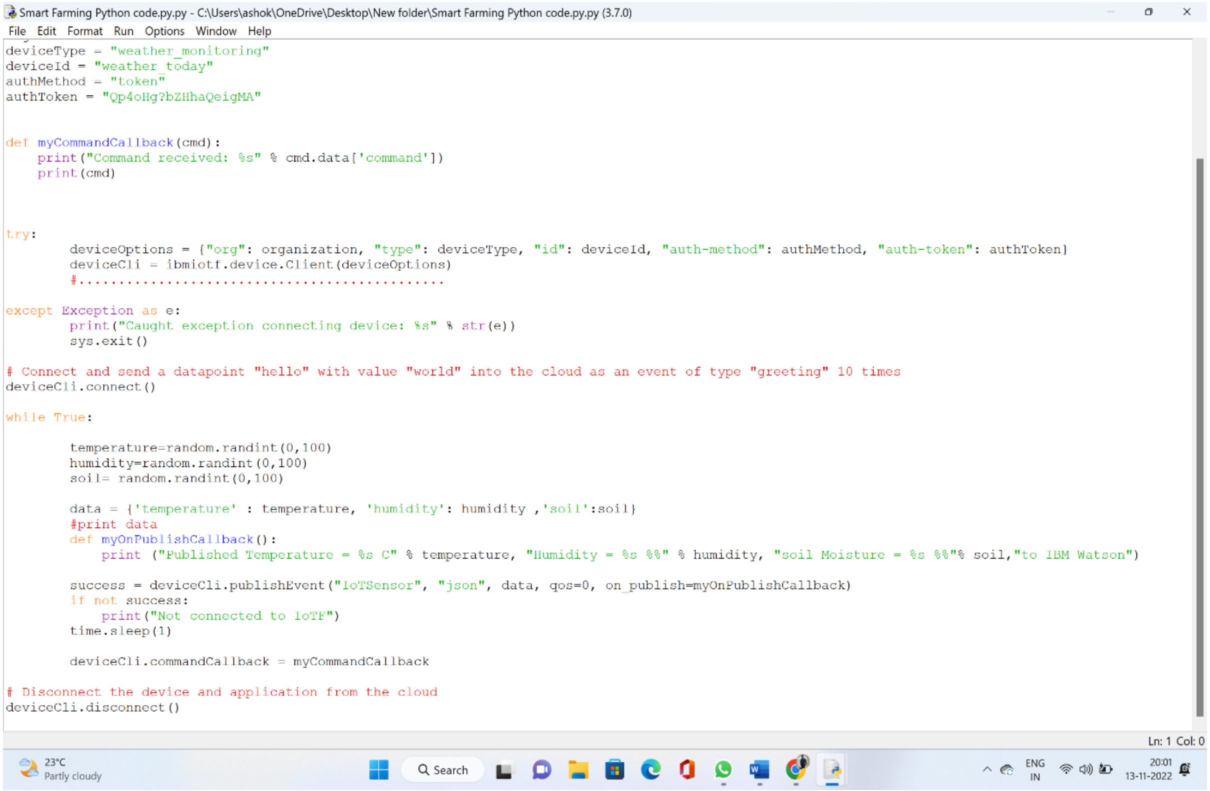
Click on file and open new file the window appear as shown below.



Before writing the python script we have install pip ibmiotf install. After that we have to write the python code.


Smart Farming Python code.py.py - C:\Users\ashok\OneDrive\Desktop\New folder\Smart Farming Python code.py.py (3.7.0)
File Edit Format Run Options Window Help
Ln: 1 Col: 0
23°C
Partly cloudy
Search
ENG
IN
20:01
13-11-2022

```python
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random


#Provide your IBM Watson Device Credentials
organization = "49x4b9"
deviceType = "weather_monitoring"
deviceId = "weather_today"
authMethod = "token"
authToken = "Qp4oHg?bZHhaQeigMA"


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" into the 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,"to IBM Watson")

        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPublishCallback)
```


Smart Farming Python code.py.py - C:\Users\ashok\OneDrive\Desktop\New folder\Smart Farming Python code.py.py (3.7.0)
File Edit Format Run Options Window Help
Ln: 1 Col: 0
23°C
Partly cloudy
Search
ENG
IN
20:01
13-11-2022

```
deviceType = "weather_monitoring"
deviceId = "weather_today"
authMethod = "token"
authToken = "Qp4oHg?bZHhaQeigMA"


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" into the 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,"to IBM 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()
```

Python code to connect the IBM Watson platform

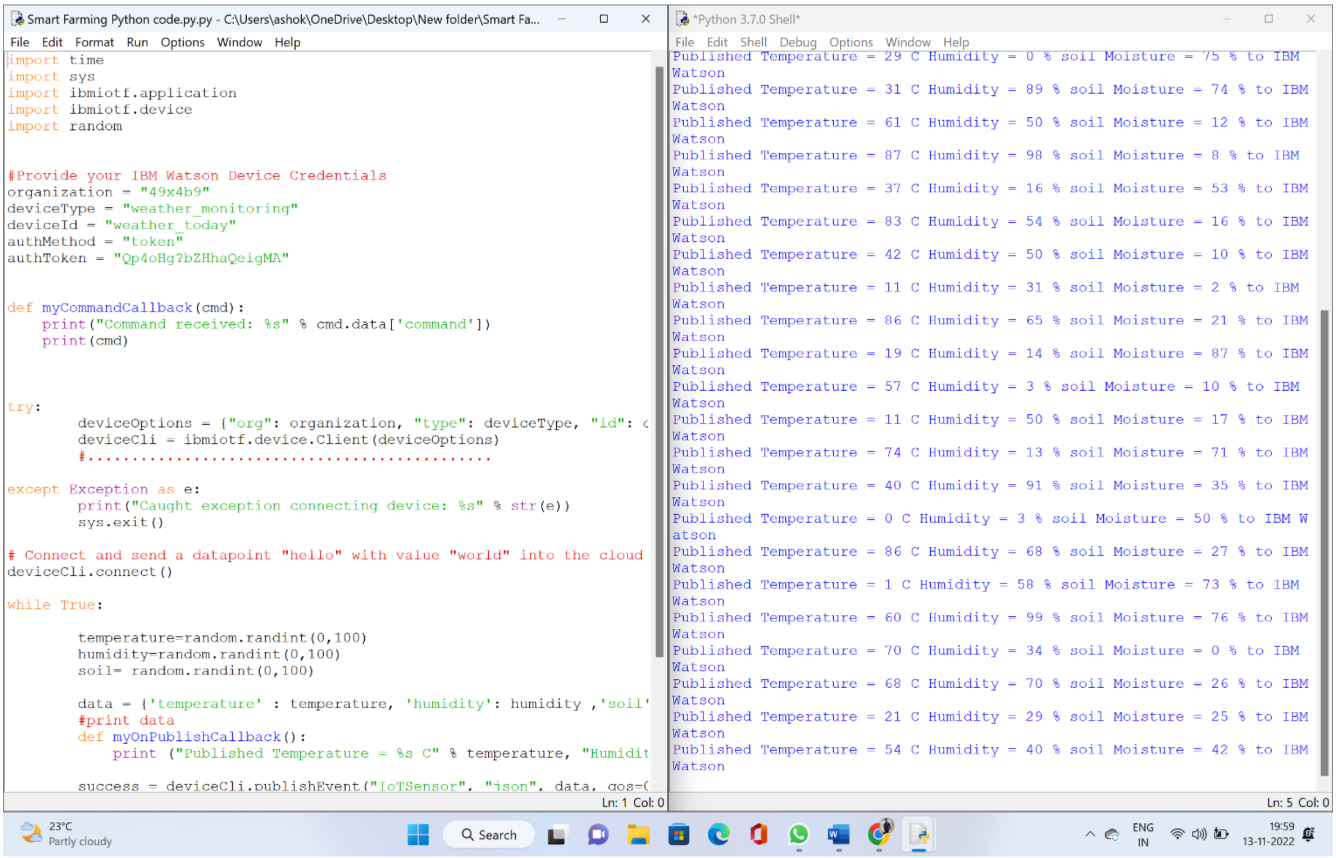
**CODE:**

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random


#Provide your IBM Watson Device Credentials
organization = "49x4b9"
deviceType = "weather_monitoring"
deviceId = "weather_today"
authMethod = "token"
authToken = "Qp4oHg?bZHhaQeigMA"
```

```
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"
into the 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,"to IBM 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()
```

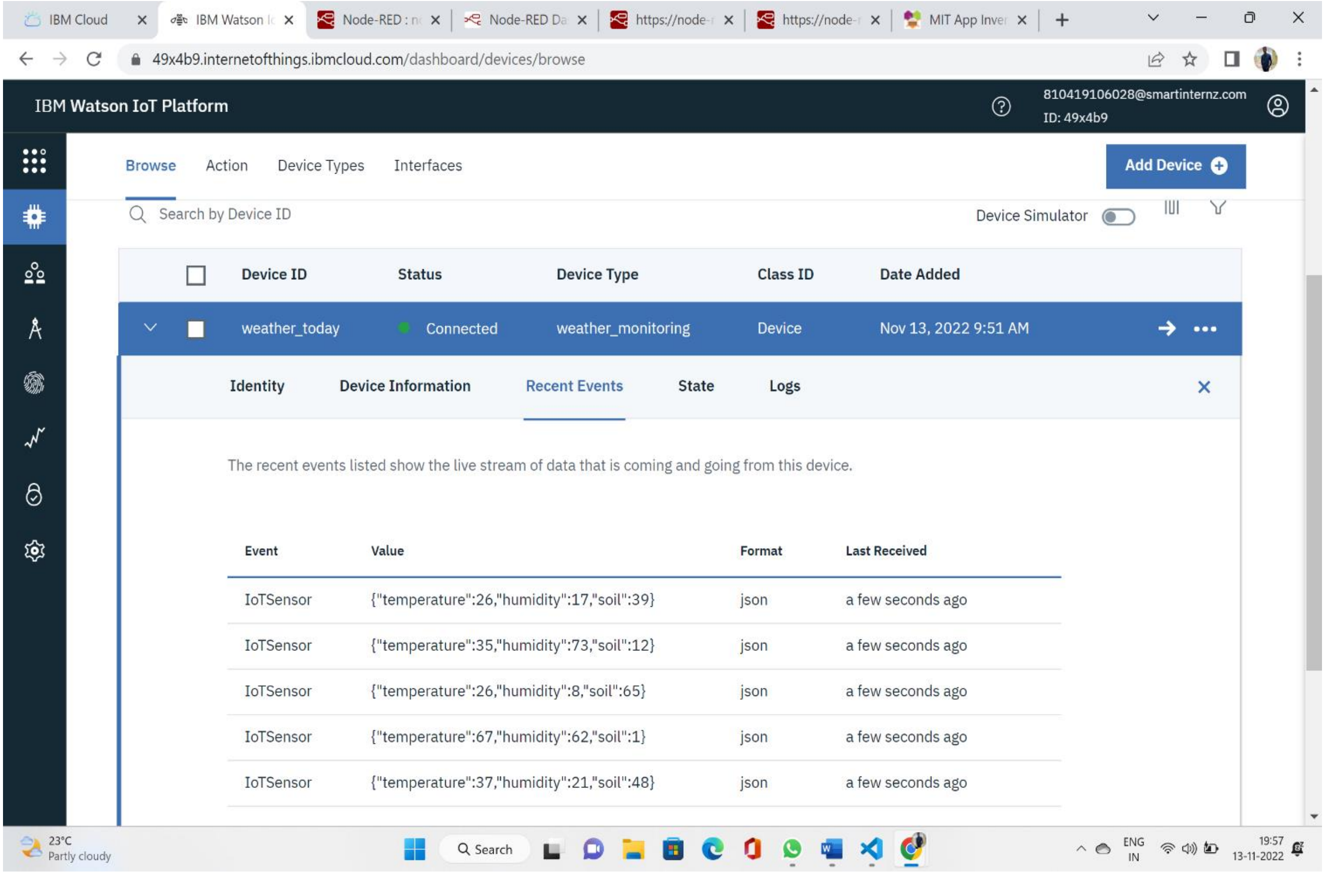
**Simulation output in the python idle:**


Smart Farming Python code.py.py - C:\Users\ashok\OneDrive\Desktop\New folder\Smart Fa...
File Edit Format Run Options Window Help
Ln: 1 Col: 0
*Python 3.7.0 Shell*
File Edit Shell Debug Options Window Help
Ln: 5 Col: 0
23°C
Partly cloudy
Search
ENG
IN
19:59
13-11-2022

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random


#Provide your IBM Watson Device Credentials
organization = "49x4b9"
deviceType = "weather_monitoring"
deviceId = "weather_today"
authMethod = "token"
authToken = "Qp4oHg?bZHhaQeigMA"


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



try:
        deviceOptions = {"org": organization, "type": deviceType, "id": c
        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" into the cloud
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'
        #print data
        def myOnPublishCallback():
            print ("Published Temperature = %s C" % temperature, "Humidit

        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=(
```

```
Published Temperature = 29 C Humidity = 0 % soil Moisture = 75 % to IBM
Watson
Published Temperature = 31 C Humidity = 89 % soil Moisture = 74 % to IBM
Watson
Published Temperature = 61 C Humidity = 50 % soil Moisture = 12 % to IBM
Watson
Published Temperature = 87 C Humidity = 98 % soil Moisture = 8 % to IBM
Watson
Published Temperature = 37 C Humidity = 16 % soil Moisture = 53 % to IBM
Watson
Published Temperature = 83 C Humidity = 54 % soil Moisture = 16 % to IBM
Watson
Published Temperature = 42 C Humidity = 50 % soil Moisture = 10 % to IBM
Watson
Published Temperature = 11 C Humidity = 31 % soil Moisture = 2 % to IBM
Watson
Published Temperature = 86 C Humidity = 65 % soil Moisture = 21 % to IBM
Watson
Published Temperature = 19 C Humidity = 14 % soil Moisture = 87 % to IBM
Watson
Published Temperature = 57 C Humidity = 3 % soil Moisture = 10 % to IBM
Watson
Published Temperature = 11 C Humidity = 50 % soil Moisture = 17 % to IBM
Watson
Published Temperature = 74 C Humidity = 13 % soil Moisture = 71 % to IBM
Watson
Published Temperature = 40 C Humidity = 91 % soil Moisture = 35 % to IBM
Watson
Published Temperature = 0 C Humidity = 3 % soil Moisture = 50 % to IBM W
atson
Published Temperature = 86 C Humidity = 68 % soil Moisture = 27 % to IBM
Watson
Published Temperature = 1 C Humidity = 58 % soil Moisture = 73 % to IBM
Watson
Published Temperature = 60 C Humidity = 99 % soil Moisture = 76 % to IBM
Watson
Published Temperature = 70 C Humidity = 34 % soil Moisture = 0 % to IBM
Watson
Published Temperature = 68 C Humidity = 70 % soil Moisture = 26 % to IBM
Watson
Published Temperature = 21 C Humidity = 29 % soil Moisture = 25 % to IBM
Watson
Published Temperature = 54 C Humidity = 40 % soil Moisture = 42 % to IBM
Watson
```

## Python output Showing in IBM Watson platform:



|  | Device ID | Status | Device Type | Class ID | Date Added |
|---|---|---|---|---|---|
| □ | weather_today | Connected | weather_monitoring | Device | Nov 13, 2022 9:51 AM |

The recent events listed show the live stream of data that is coming and going from this device.

| Event | Value | Format | Last Received |
|---|---|---|---|
| IoTSensor | {"temperature":26,"humidity":17,"soil":39} | json | a few seconds ago |
| IoTSensor | {"temperature":35,"humidity":73,"soil":12} | json | a few seconds ago |
| IoTSensor | {"temperature":26,"humidity":8,"soil":65} | json | a few seconds ago |
| IoTSensor | {"temperature":67,"humidity":62,"soil":1} | json | a few seconds ago |
| IoTSensor | {"temperature":37,"humidity":21,"soil":48} | json | a few seconds ago |

## WOKWI Online Simulator ESP32 :

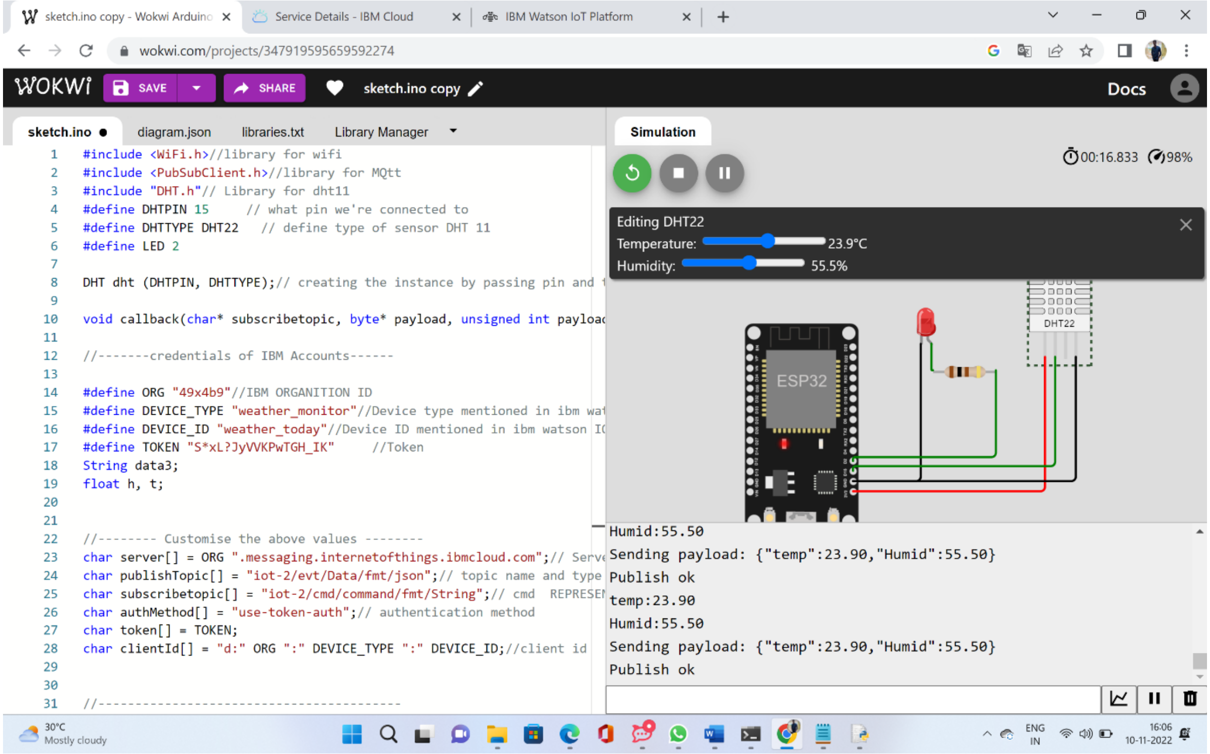
[https://wokwi.com/projects/347919595659592274](https://wokwi.com/projects/347919595659592274)

```
#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 "49x4b9"//IBM ORGANITION ID
```

```
#define DEVICE_TYPE "weather_monitor"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "weather_today"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "S*xL?JyVVKPwTGH_IK"     //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="";
}
```

## Simulation Output in the Wokwi web site:

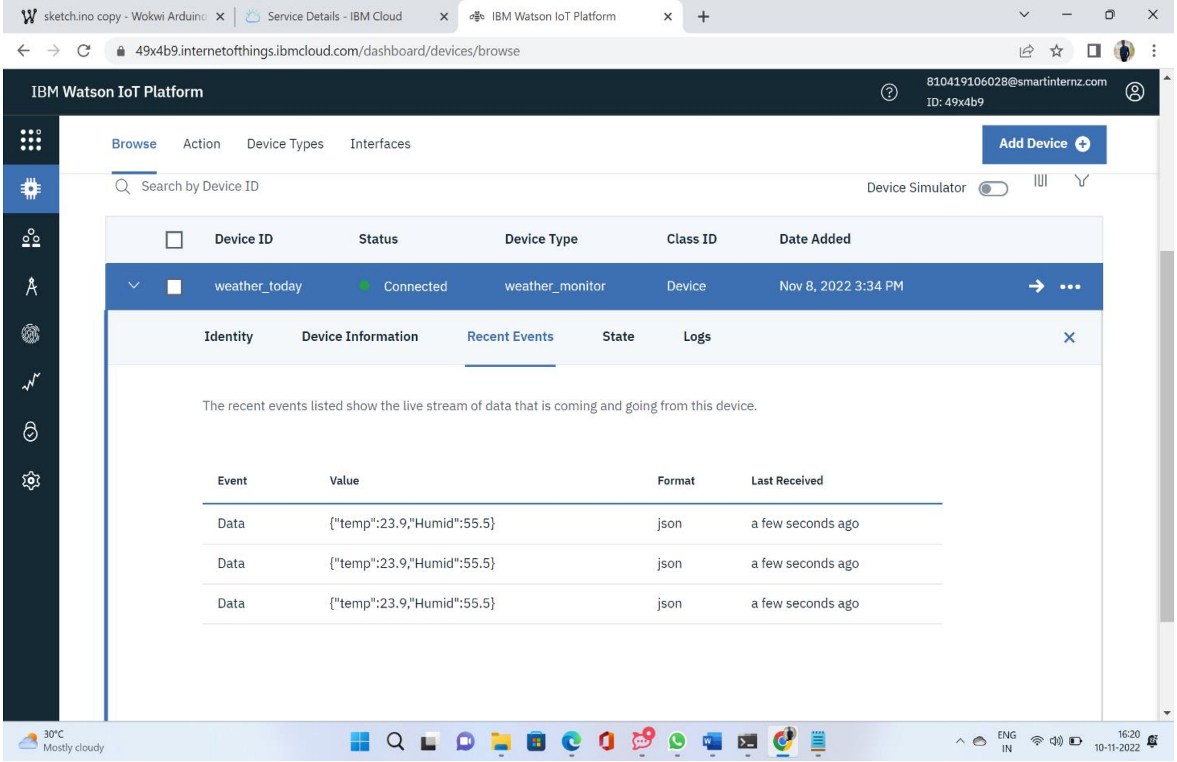


```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht11
#define DHTPIN 15     // what pin we're connected to
#define DHTTYPE DHT22   // define type of sensor DHT 11
#define LED 2

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

void callback(char* subscribetopic, byte* payload, unsigned int payload

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

#define ORG "49x4b9"//IBM ORGANITION ID
#define DEVICE_TYPE "weather_monitor"//Device type mentioned in ibm wat
#define DEVICE_ID "weather_today"//Device ID mentioned in ibm watson IO
#define TOKEN "S*xL?JyVVKPwTGH_IK"     //Token
String data3;
float h, t;


//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Serve
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd  REPRESEN
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id


//----------------------------------------
```

Editing DHT22
Temperature: 23.9°C
Humidity: 55.5%

```
Humid:55.50
Sending payload: {"temp":23.90,"Humid":55.50}
Publish ok
temp:23.90
Humid:55.50
Sending payload: {"temp":23.90,"Humid":55.50}
Publish ok
```

## Wokwi Simulation Output in the IBM Watson Platform:



| ☐ | Device ID | Status | Device Type | Class ID | Date Added |
|---|---|---|---|---|---|
| ☐ | weather_today | ● Connected | weather_monitor | Device | Nov 8, 2022 3:34 PM |

Identity · Device Information · Recent Events · State · Logs

The recent events listed show the live stream of data that is coming and going from this device.

| Event | Value | Format | Last Received |
|---|---|---|---|
| Data | {"temp":23.9,"Humid":55.5} | json | a few seconds ago |
| Data | {"temp":23.9,"Humid":55.5} | json | a few seconds ago |
| Data | {"temp":23.9,"Humid":55.5} | json | a few seconds ago |