

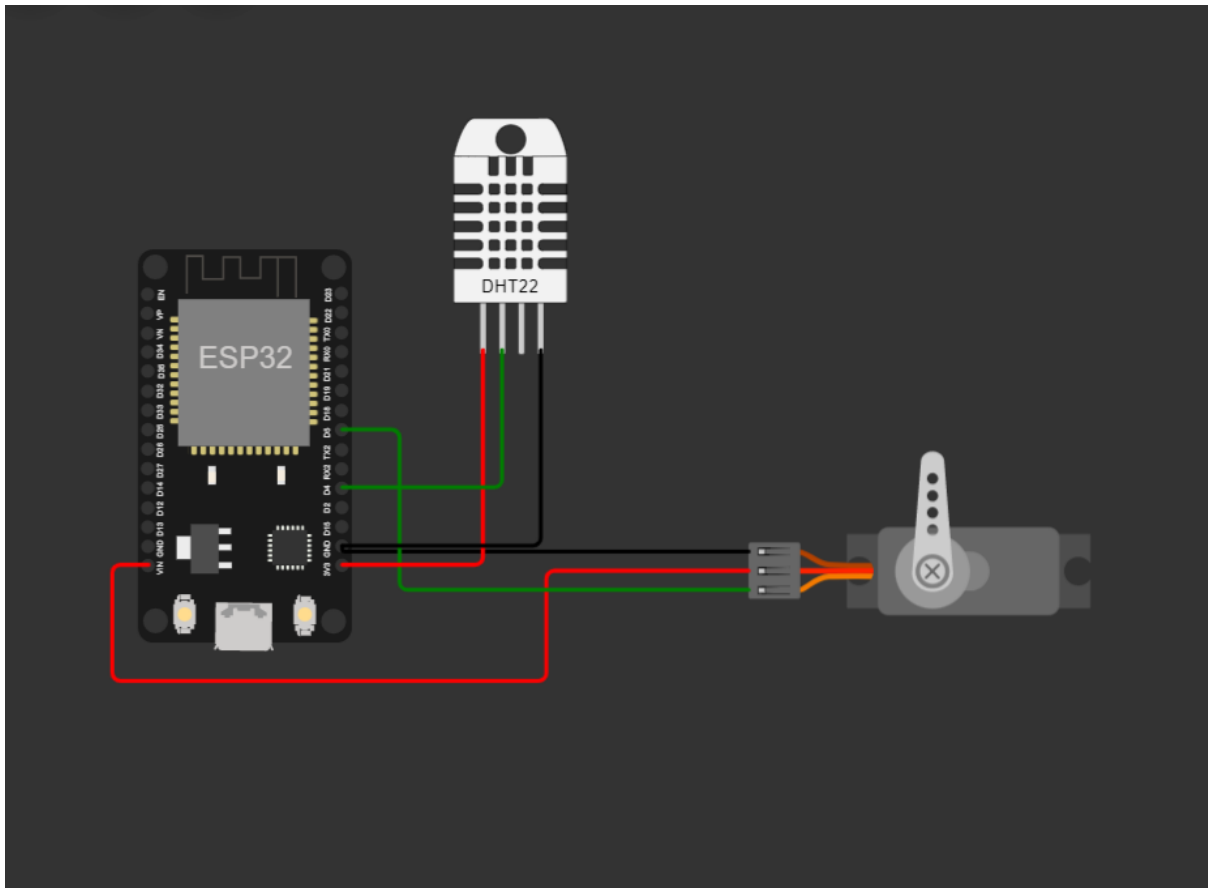
# Sprint – 1

Date	31-October-2022
Team ID	PNT2022TMID25993
Project Name	Smart Farmer – IoT Enabled Smart Farming Application
Maximum Marks	-

## Sprint 1 Task:

1. Python Code (Python Code Uploaded in GIT Repo.)
2. Circuit Connection
3. Python Program Output
4. Wokwi Simulation C++ Output

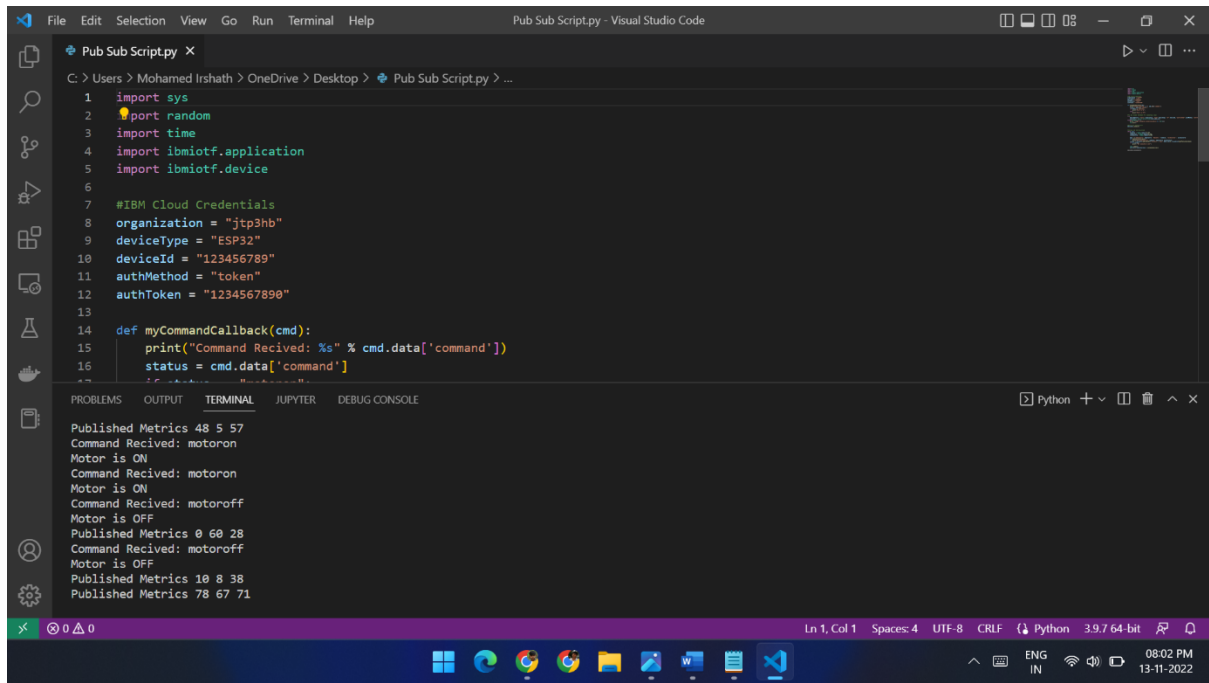
## Hardware Connection:



Used Servo Motor and DHT22 sensors along with ESP32 in Wokwi Simulation to sense the temperature, humidity and to control the motor operations.

Soil Moisture sensor is not available in Wokwi instead we used random module to generate random value to test it.

## Python Code Output:



The screenshot shows the Visual Studio Code editor with a Python script named 'Pub Sub Script.py'. The code imports necessary libraries, defines IBM Cloud credentials, and sets up a callback function to handle incoming commands. The terminal window displays the output of the script, showing commands received and metrics published.

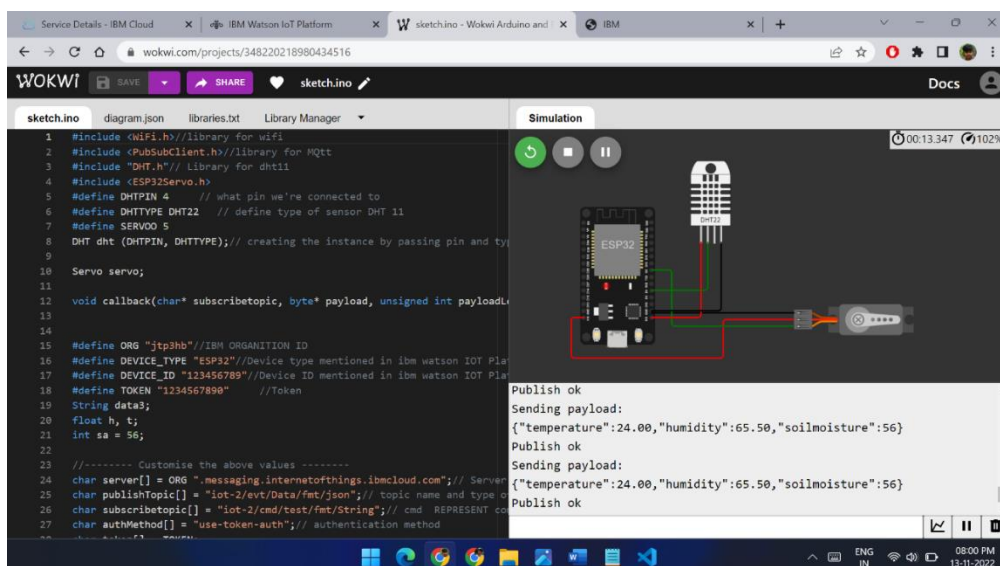
```
1 import sys
2 import random
3 import time
4 import ibmiotf.application
5 import ibmiotf.device
6
7 #IBM Cloud Credentials
8 organization = "jtp3hb"
9 deviceType = "ESP32"
10 deviceId = "123456789"
11 authMethod = "token"
12 authToken = "1234567890"
13
14 def myCommandCallback(cmd):
15     print("Command Received: %s" % cmd.data['command'])
16     status = cmd.data['command']
```

Terminal Output:

```
Published Metrics 48 5 57
Command Received: motoron
Motor is ON
Command Received: motoron
Motor is ON
Command Received: motoroff
Motor is OFF
Published Metrics 0 60 28
Command Received: motoroff
Motor is OFF
Published Metrics 10 8 38
Published Metrics 78 67 71
```

The above attached picture shows the output of the python code in the terminal of the VS Code Editor.

## Wokwi Simulation Output:



The screenshot shows the Wokwi simulator interface. On the left, the C++ code for an ESP32 is displayed, including headers for WiFi, MQTT, DHT, and Servo, and the main logic for connecting to IBM Cloud IoT Platform and publishing data. On the right, the simulation output shows the device publishing data to the cloud.

```
1 #include <WiFi.h> //library for WiFi
2 #include <PubSubClient.h> //library for MQTT
3 #include "DHT.h" // Library for dht11
4 #include <ESP32Servo.h>
5 #define DHTPIN 4 // what pin we're connected to
6 #define DHTTYPE DHT22 // define type of sensor DHT 11
7 #define SERVO0 5
8 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type
9
10 Servo servo;
11
12 void callback(char* topic, byte* payload, unsigned int length) {
13
14 }
15
16 #define ORG "jtp3hb" //IBM ORGANIZATION ID
17 #define DEVICE_TYPE "ESP32" //Device type mentioned in ibm watson IoT Platform
18 #define DEVICE_ID "123456789" //Device ID mentioned in ibm watson IoT Platform
19 #define TOKEN "1234567890" //Token
20 String data3;
21 float h, t;
22 int sa = 56;
23
24 //----- Customise the above values -----
25 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server address
26 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of data
27 char subscribeTopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command
```

Simulation Output:

```
Publish ok
Sending payload:
{"temperature":24.00,"humidity":65.50,"soilmoisture":56}
Publish ok
Sending payload:
{"temperature":24.00,"humidity":65.50,"soilmoisture":56}
Publish ok
```

The above attached picture shows the output of the C++ program running in the Wokwi simulator.