

PROJECT DEVELOPMENT PHASE

SPRINT 2

TEAM ID PROJECT	PNT2022TMID34075
TITLE	IOT BASED SMART CROP PROTECTION SYSTEM FOR AGRICULTURE

PYTHON CODE

```
import time

import sys

import ibmiotf.application

import ibmiotf.device

import random

#Provide your IBM Watson Device Credentials

organization = "1be6ey"

deviceType = " b11m3e-device"

deviceId = " b11m3edeviceid1"

authMethod = " use-token-auth"

authToken = "SJmK4i0jsT67bZb)P@ "

# Initialize GPIO

def myCommandCallback(cmd):

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

    status=cmd.data['command']

    if status=="lighton":

        print ("led is on")

    else :

        print ("led is off")

    #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:

#Get Sensor Data from DHT11

temp=random.randint(0,100)

Humid=random.randint(0,100)

data= { 'Temperature' : temp, 'Humidity': Humid }

#print data

def myOnPublishCallback():

print ("Published Temperature = %s C" % temp, "Humidity = %s %" % Humid, "to IBM Watson")
success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPublishCallback)

if not success:

print("Not connected to IoT")

time.sleep(1)

deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud

deviceCli.disconnect()

```

OUTPUT

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Published Temperature = 33 C Humidity = 61 % to IBM Watson
Published Temperature = 20 C Humidity = 34 % to IBM Watson
Published Temperature = 80 C Humidity = 45 % to IBM Watson
Published Temperature = 41 C Humidity = 64 % to IBM Watson
Published Temperature = 5 C Humidity = 1 % to IBM Watson
Published Temperature = 50 C Humidity = 18 % to IBM Watson
Published Temperature = 93 C Humidity = 86 % to IBM Watson
Published Temperature = 5 C Humidity = 69 % to IBM Watson
Published Temperature = 8 C Humidity = 69 % to IBM Watson
Published Temperature = 40 C Humidity = 55 % to IBM Watson
Published Temperature = 57 C Humidity = 84 % to IBM Watson
Published Temperature = 5 C Humidity = 33 % to IBM Watson
Published Temperature = 84 C Humidity = 79 % to IBM Watson
Published Temperature = 2 C Humidity = 26 % to IBM Watson
Published Temperature = 31 C Humidity = 55 % to IBM Watson
Published Temperature = 7 C Humidity = 36 % to IBM Watson
Published Temperature = 29 C Humidity = 90 % to IBM Watson
Published Temperature = 86 C Humidity = 9 % to IBM Watson
Published Temperature = 56 C Humidity = 77 % to IBM Watson
Published Temperature = 10 C Humidity = 25 % to IBM Watson
Published Temperature = 44 C Humidity = 52 % to IBM Watson
Published Temperature = 20 C Humidity = 87 % to IBM Watson
Published Temperature = 99 C Humidity = 4 % to IBM Watson
Published Temperature = 3 C Humidity = 17 % to IBM Watson
Published Temperature = 84 C Humidity = 8 % to IBM Watson
Published Temperature = 36 C Humidity = 65 % to IBM Watson
Published Temperature = 81 C Humidity = 25 % to IBM Watson
Published Temperature = 49 C Humidity = 12 % to IBM Watson
Published Temperature = 75 C Humidity = 89 % to IBM Watson
Published Temperature = 59 C Humidity = 14 % to IBM Watson
Published Temperature = 76 C Humidity = 36 % to IBM Watson
Published Temperature = 34 C Humidity = 32 % to IBM Watson
Published Temperature = 2 C Humidity = 50 % to IBM Watson
Published Temperature = 84 C Humidity = 54 % to IBM Watson
Published Temperature = 53 C Humidity = 24 % to IBM Watson
Published Temperature = 81 C Humidity = 7 % to IBM Watson
Published Temperature = 5 C Humidity = 7 % to IBM Watson
Published Temperature = 34 C Humidity = 4 % to IBM Watson
Published Temperature = 36 C Humidity = 34 % to IBM Watson
Published Temperature = 26 C Humidity = 3 % to IBM Watson
```

OUTPUT IN WATSON IOT PLATFORM

IBM Watson IoT Platform

1be6ey.internetofthings.ibmcloud.com/dashboard/devices/browse

960219106001@smartinternz.com
ID: 1be6ey

Browse Action Device Types Interfaces Add Device

B1devicetype_1 Disconnected B1devicetype Device Nov 15, 2022 12:17 PM

b11m3edevicel1 Connected b11m3e-device Device Nov 4, 2022 7:33 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
IoTSensor	{"Temperature":80,"Humidity":45}	json	a few seconds ago
IoTSensor	{"Temperature":20,"Humidity":34}	json	a few seconds ago
IoTSensor	{"Temperature":33,"Humidity":61}	json	a few seconds ago
IoTSensor	{"Temperature":67,"Humidity":56}	json	a few seconds ago

CIRCUIT SIMULATION IN WOKWI- CODE

```
#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);

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

#define ORG "1be6ey"

#define DEVICE_TYPE "b11m3e-device"

#define DEVICE_ID "b11m3edeviceid1"

#define TOKEN "SJmK4i0jsT67bZb)P@"

String data3;

float h, t;

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

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

char subscribetopic[] = "iot-2/cmd/command/fmt/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);

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("Temperature:");

Serial.println(t);

Serial.print("Humidity:");

Serial.println(h);

PublishData(t, h);

delay(1000);

if (!client.loop()) {

mqttconnect();

}

}

void PublishData(float temp, float humid) {

mqttconnect();

String payload = "{\"Temperature\":";

payload += temp;

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

payload += humid;

payload += "}";
```

```

Serial.print("Sending payload: ");

Serial.println(payload);

if (client.publish(publishTopic, (char*) payload.c_str())) {

Serial.println("Publish ok");

} 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);

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++) {

data3 += (char)payload[i];

}

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

if(data3=="lighton")

{

Serial.println(data3);

```

```

digitalWrite(LED,HIGH);

}

else

{

Serial.println(data3);

digitalWrite(LED,LOW);

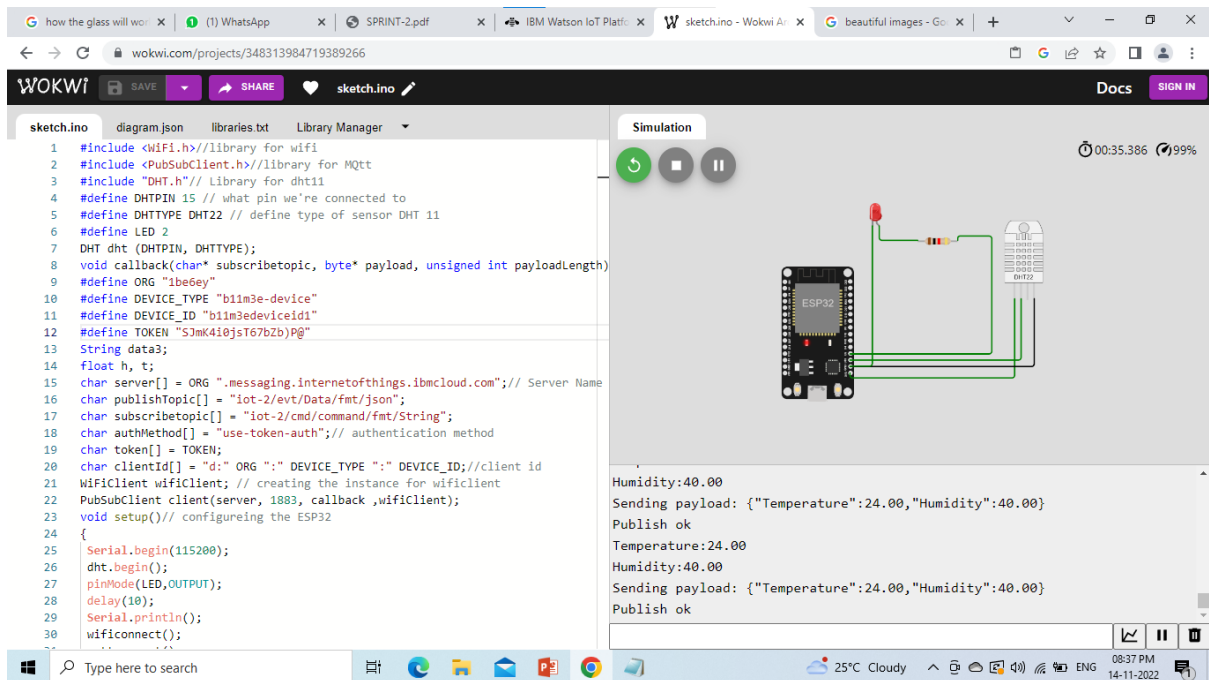
}

data3="";

}

```

OUTPUT



The screenshot shows the Wokwi simulation environment. On the left, the sketch.ino file is open, displaying the following code:

```

1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3 #include "DHT.h" // Library for dht11
4 #define DHTPIN 15 // what pin we're connected to
5 #define DHTTYPE DHT22 // define type of sensor DHT 11
6 #define LED 2
7 DHT dht (DHTPIN, DHTTYPE);
8 void callback(char* subscribtopic, byte* payload, unsigned int payloadLength)
9 #define ORG "ibe6ey"
10 #define DEVICE_TYPE "b11m3e-device"
11 #define DEVICE_ID "b11m3edeviceld1"
12 #define TOKEN "Sjmk4i8jsT67bzbP8"
13 String data3;
14 float h, t;
15 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
16 char publishTopic[] = "iot-2/evt/Data/fmt/json";
17 char subscribtopic[] = "iot-2/cmd/command/fmt/String";
18 char authMethod[] = "use-token-auth"; // authentication method
19 char token[] = TOKEN;
20 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
21 WiFiClient wificlient; // creating the instance for wificlient
22 PubSubClient client(server, 1883, callback ,wificlient);
23 void setup()// configureing the ESP32
24 {
25   Serial.begin(115200);
26   dht.begin();
27   pinMode(LED,OUTPUT);
28   delay(10);
29   Serial.println();
30   wificlient.connect();

```

On the right, the simulation window shows a visual representation of the ESP32 board connected to an LED and a DHT22 sensor. The console output at the bottom shows the following sequence of events:

```

Humidity:40.00
Sending payload: {"Temperature":24.00,"Humidity":40.00}
Publish ok
Temperature:24.00
Humidity:40.00
Sending payload: {"Temperature":24.00,"Humidity":40.00}
Publish ok

```

WOWKI SIMULATION LINK

<https://wokwi.com/projects/348313984719389266>

PUBLISH THE OUTPUT TO IBM WATSON- OUTPUT

The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains icons for various platform features. The main content area shows a list of devices, with 'b11m3edevicid1' selected and its 'Recent Events' tab active. Below the tab, a message states: 'The recent events listed show the live stream of data that is coming and going from this device.' A table follows, listing recent events with columns for Event, Value, Format, and Last Received.

Event	Value	Format	Last Received
Data	{"Temperature":24,"Humidity":40}	json	a few seconds ago
Data	{"Temperature":24,"Humidity":40}	json	a few seconds ago
Data	{"Temperature":24,"Humidity":40}	json	a few seconds ago
Data	{"Temperature":24,"Humidity":40}	json	a few seconds ago
Data	{"Temperature":24,"Humidity":40}	json	a few seconds ago

The bottom of the image shows a Windows taskbar with a search bar, application icons, and system status information including temperature (25°C), weather (Cloudy), and time (08:38 PM, 14-11-2022).