# CODING & SOLUTIONING

| | |
|---|---|
| **DATE** | 10 NOVEMBER 2022 |
| **TEAM ID** | PNT2022TMID23451 |
| **PROJECT NAME** | Project - SmartFarmer -IOT Enabled smart Farming Application |
| **MAXIMUM MARKS** | 4 marks |

## 7.1 FEATURE 1

This code is used for connect the IBM Watson Iot platform.

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

#Provide your IBM Watson Device Credentials
organization = "bnsfkk"
deviceType ="Weather_Monitor"
deviceId = "weather"
authMethod = "token"
authToken = "weatherravi"

# Initialize GPIO
temp=random.randint(0,100)
```

```
pulse=random.randint(0,100)
oxygen= random.randint(0,100)
lat = 17
lon = 18
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:
#Get Sensor Data from DHT11
 data = {"d":{ 'temp' : temp, 'pulse': pulse ,'oxygen': oxygen,"lat":lat,"lon":lon}}
#print data
```

```
def myOnPublishCallback():
    print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % pulse, "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()
```

**IBM Watson IoT Platform..**

**7.2 FEATURE 2**

**Connecting Sensors with wokwi using C++**

```
#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 "i3869j"//IBM ORGANITION ID
#define DEVICE_TYPE "abcd"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "1234"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678"     //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="";
}
//Program for resistor in Wokwi
{
  "version": 1,
  "author": "Anonymous maker",
  "editor": "wokwi",
```

```
 "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 4.8, "left": -127.69,
"attrs": {} },
   { "type": "wokwi-dht22", "id": "dht1", "top": -76.72, "left": 137.76, "attrs": {}
},
   {
     "type": "wokwi-led",
     "id": "led1",
     "top": -16.04,
     "left": 21.83,
     "attrs": { "color": "red" }
   },
   {
     "type": "wokwi-resistor",
     "id": "r1",
     "top": 41.63,
     "left": 48.17,
     "attrs": { "value": "100" }
   }
 ],
 "connections": [
   [ "esp:TX0", "$serialMonitor:RX", "", [] ],
   [ "esp:RX0", "$serialMonitor:TX", "", [] ],
   [ "dht1:VCC", "esp:3V3", "red", [ "v0" ] ],
   [ "dht1:GND", "esp:GND.1", "black", [ "v0" ] ],
   [ "led1:A", "r1:1", "green", [ "v0" ] ],
```

```
    [ "led1:C", "esp:GND.1", "black", [ "v0" ] ],
    [ "dht1:SDA", "esp:D15", "green", [ "v101.76", "h-2.06" ] ],
    [ "r1:2", "esp:D2", "green", [ "v80.85", "h-3.49" ] ]
]
}
```