Coding and Solution

Team ID	PNT2022TMID47935
Project Name	Real-time river water quality monitoring and control system

Code Layout

```
#include <WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"// Library for dht11
#define DHTPIN 15     // what pin we're connected to
#define DHTTYPE DHT22   // define type of sensor DHT 11
DHT dht (DHTPIN, DHTTYPE);
void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);

WiFiClient wifiClient;
String data3;
#define ORG "ks8pti"
#define DEVICE_TYPE "ESP32"
#define DEVICE_ID "143143"
#define TOKEN "123456789"
#define speed 0.034
#define led 14
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char topic[] = "iot-2/cmd/command/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
PubSubClient client(server, 1883, wifiClient);
void publishData();


const int trigpin=5;
const int echopin=18;
String command;
String data="";

long duration;
float dist;
float Temp;
int pH;



void setup()
```

```
{
  Serial.begin(115200);
  dht.begin();
  pinMode(led, OUTPUT);
  pinMode(trigpin, OUTPUT);
  pinMode(echopin, INPUT);
  wifiConnect();
  mqttConnect();
}

void loop() {
  bool isNearby = dist < 100;
  digitalWrite(led, isNearby);

  pH = dht.readHumidity();
  Temp = dht.readTemperature();
  Serial.print("Temperature:");
  Serial.println(Temp);
  Serial.print("Tubidity:");
  Serial.println(pH);

  publishData();
  delay(1000);
  if (!client.loop()) {
    mqttConnect();
  }
}

void wifiConnect() {
  Serial.print("Connecting to "); Serial.print("Wifi");
  WiFi.begin("Wokwi-GUEST", "", 6);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.print("WiFi connected, IP address: ");
Serial.println(WiFi.localIP());
}

void mqttConnect() {
  if (!client.connected()) {
    Serial.print("Reconnecting MQTT client to "); Serial.println(server);
    while (!client.connect(clientId, authMethod, token)) {
      Serial.print(".");
      delay(500);
    }
    initManagedDevice();
    Serial.println();
```

```
  }
}

void initManagedDevice() {
  if (client.subscribe(topic)) {
    // Serial.println(client.subscribe(topic));
    Serial.println("IBM subscribe to cmd OK");
  } else {
    Serial.println("subscribe to cmd FAILED");
  }
}
void publishData()
{
  digitalWrite(trigpin,LOW);
  digitalWrite(trigpin,HIGH);
  delayMicroseconds(10);
  digitalWrite(trigpin,LOW);
  duration=pulseIn(echopin,HIGH);
  dist=duration*speed/2;
  if(dist<100){
    String payload = "{\"Turbidity\":";
    payload += dist;
    payload += ","\"Temperature\":";
    payload += Temp;
    payload += "," "\"pH\":";
    payload += pH;
    payload += "}";

    Serial.print("\n");
    Serial.print("Sending payload: ");
    Serial.println(payload);
      if(client.publish(publishTopic, (char*) payload.c_str())) {
      Serial.println("Warning crosses 110cm -- it automaticaly of the loop");
      digitalWrite(led,HIGH);
    }

  }
    if(dist>101 && dist<111){
    String payload = "{\"Normal Distance\":";
    payload += dist;
    payload += "}";



    Serial.print("\n");
    Serial.print("Sending payload: ");
    Serial.println(payload);

    }
```

```
  }


  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++){
    dist += (char)payload[i];
  }
  Serial.println("data:"+ data3);
  if(data3=="lighton"){
    Serial.println(data3);
    digitalWrite(led,HIGH);
  }
  data3="";
}
```

Code Readability and Reusability

- This code can easy to read and understand everything faster.
- In this code we can reuse every part code

Python Random Value Generation Code

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


#Provide your IBM Watson Device Credentials
organization = "ks8pti"
deviceType = "ESP32"
deviceId = "143143"
authMethod = "token"
authToken = "123456789"

# Initialize GPIO


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

```
    if status=="START":
        print ("Motor is Started")
    elif status=="STOP":
        print ("Motor is oFF state")
    elif status=="LEFT":
        print ("Left Side is Closed")
    elif status=="RIGHT":
        print ("Right Side is Closed")
    elif status=="FORWARD":
        print ("Message is Forward to the chief")
    else :
        print ("Send a proper command")

    #print(cmd)



try:
                                                    deviceOpti
ons = {"org": organization, "type": deviceType, "id":
```

```
deviceId, "auth-method": authMethod, "auth-token":
authToken}
                                                    deviceCli
= ibmiotf.device.Client(deviceOptions)
                                                    #.....
.....

except Exception as e:
                                                    print("Cau
ght 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

        Temperature=random.randint(0,100)
        Turbidity=random.randint(0,100)
        pH=random.randint(0,14)
```

```
        data = { 'Temperature' : Temperature, 'Turbidity':
Turbidity, 'pH' : pH }
        #print data
        def myOnPublishCallback():
            print ("Published Temperature = %s C" %
Temperature, "Turbidity = %s %%" % Turbidity, "pH = %s L"
% pH, "to IBM Watson")

        success = deviceCli.publishEvent("IoTSensor", "json",
data, qos=0, on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoTF")
        time.sleep(20)

        deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud
deviceCli.disconnect()
```