**FINAL CODE**

|  |
| --- |
| #include <WiFi.h> |
|  | #include<WiFiClient.h> |
|  | #include<PubSubClient.h> |
|  |  |
|  | int pH; |
|  | int Turbidity; |
|  | String data = ""; |
|  | String command; |
|  | boolean a ; |
|  | void callback(char\* topic, byte\* payload, unsigned int payLoadLength); |
|  |  |
|  | const char\* ssid = "Wokwi-GUEST"; |
|  | const char\* password = ""; |
|  |  |
|  | #define ORG "vzye9j" |
|  | #define DEVICE\_TYPE "ibm-final-esp32" |
|  | #define DEVICE\_ID "789456123" |
|  | #define TOKEN "147258369" |
|  |  |
|  | // pin declaration |
|  | #define LED 19 |
|  |  |
|  |  |
|  | char topic [] = "iot-2/cmd/command/fmt/String"; |
|  | char server [] = ORG ".messaging.Internetofthings.ibmcloud.com"; |
|  | char pubTopic1 [] = "iot-2/evt/Data/fmt/json"; |
|  | char authMethod [] = "use-token-auth"; |
|  | char token [] = TOKEN; |
|  | char clientId [] = "d:" ORG ":" DEVICE\_TYPE ":" DEVICE\_ID; |
|  |  |
|  | WiFiClient wifiClient; |
|  | PubSubClient client(server, 1883, callback, wifiClient); |
|  |  |
|  | int publishInterval = 1000; |
|  | long lastmsg; |
|  | void publishData(); |
|  | void wifiConnect(); |
|  | void mqttConnect(); |
|  | void setup() { |
|  |  |
|  | Serial.begin(115200); |
|  | Serial.println(); |
|  | pinMode(LED, OUTPUT); |
|  |  |
|  | wifiConnect(); // to connect to wifi |
|  | mqttConnect(); // to connect to mqtt client |
|  | } |
|  |  |
|  | void loop() { |
|  | if (millis() - lastmsg > publishInterval) { |
|  | publishData(); |
|  | lastmsg = millis(); |
|  | } |
|  |  |
|  | if (!client.loop()) { |
|  | mqttConnect(); |
|  | } |
|  | } |
|  | void wifiConnect() { |
|  | Serial.print(ssid); |
|  | WiFi.begin(ssid, password); |
|  | while (WiFi.status() != WL\_CONNECTED) { |
|  | delay(500); |
|  | Serial.print("."); |
|  | } |
|  | Serial.println(); |
|  | Serial.print("WiFi connected, Ip Address: "); |
|  | Serial.println(WiFi.localIP()); |
|  | } |
|  | 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("Bluemix connected"); |
|  | } |
|  | } |
|  | void initManagedDevice() { |
|  | if (client.subscribe(topic)) { |
|  | Serial.println("Subscribe to cmd OK"); |
|  | } |
|  | else { |
|  | Serial.println("Subscribe to cmd FAILED"); |
|  | } |
|  | } |
|  | void callback(char\* topic, byte\* payload, unsigned int payLoadLength) { |
|  | Serial.print("callback invoked for topic: "); |
|  | Serial.println(topic); |
|  | for (int i = 0; i < payLoadLength; i++) { |
|  | command += (char)payload[i]; |
|  | } |
|  | Serial.print("data: " + command); |
|  | control\_function(); |
|  | command = ""; |
|  | } |
|  | void control\_function() { |
|  | if (command == "OFF") { |
|  | Serial.println("....... motor is off ......."); |
|  | digitalWrite(LED, LOW); |
|  | } |
|  | else if (command == "ON") { |
|  | Serial.println("....... motor is on ......."); |
|  | digitalWrite(LED, HIGH); |
|  | } |
|  | else { |
|  | Serial.println("....... no command have been subscribed ......."); |
|  | } |
|  | } |
|  |  |
|  | void publishData() { |
|  | pH = random(14); |
|  | Turbidity = random(10); |
|  | if (isnan(pH) || isnan(Turbidity)) { |
|  | Serial.println("Failed to read data from Sensors"); |
|  | return; |
|  | } |
|  | String payload = "{\"data\":{\"DEVICE\_ID\":\"" DEVICE\_ID "\""; |
|  | payload += ",""\"pH\":"; |
|  | payload += pH; |
|  | payload += ",""\"Turbidity\":"; |
|  | payload += Turbidity; |
|  | payload += ",""\"flag\":"; |
|  | payload += "false"; |
|  | payload += "}}"; |
|  | Serial.println(); |
|  | Serial.print("Sending payload: "); |
|  | Serial.println(payload); |
|  | if (client.publish(pubTopic1, (char\*) payload.c\_str())) { |
|  | Serial.println("Published"); |
|  | } |
|  | else { |
|  | Serial.println("Publish FAILED"); |
|  | } |
|  | } |