Particle Photon Sends Data and Receives Commands from IBM Cloud via MQTT – Dr. Li's C++ Example Code

Dr. Xiaohai Li (xhli@citytech.cuny.edu)

Robotics & Intelligent System Research Lab
Department of Computer Engineering Technology
New York City College of Technology/CUNY

Spring 2018

The example code here is written for Particle Photon to send data (for example, from sensors) to IBM Cloud/Watson IoT Platform, receive commands from IBM Cloud, parse received commands, and take certain actions according to the commands. On the cloud side, a Node-RED application is running to receive sensor data, analyze data, and generate certain commands according the data. You can create your own Node-RED application allowing users to initiate a command (for example, a text/Twitter message).

The MQTT library used in this code is accessible from online Particle Build, which was written by Hirotaka Niisato [4]. Due to possible update of the MQTT library, the code here might need some modifications. Search and read *MQTT.h* and *MQTT.cpp* on Particle Build or Niisato's GitHub repository[4] for complete detail of the latest MQTT library.

Thoroughly read the following code, and revise it for your needs. ***Note:*** To help you understand the code, some of the important comments are highlighted.

```
/*This code is to send an integer dummydata (changing from 0 to 100) to IBM Watson IoT Platform, subscribe commands from a Node-RED app running on IBM Cloud, and blinks corresponding indicator LEDs according to the received commands.

Note: the Node-RED app on IBM Cloud sends command "turnlow" back to Photon if the app receives a dummydata in 0~32, sends command "turnmid" if receiving a daummydata in 33~66, "turnhigh" if receiving a dummydata in 67~100.
*/
```

```
 1 /* Author: Xiaohai Li
 2  * License: GPL v2.0
 3  * 05/05/2018
 4  */
 5
 6 // This #include statement was automatically added by the Particle IDE.
 7 #include <MQTT.h>        //search and include the mqtt library via Particle online IDE
 8
 9 // This #include statement was automatically added by the Particle IDE.
10 #include <RdJson.h>
11
12 #define HOST_PORT  1883 //port# used by host (IBM Watson IoT Platform): 1883 (default)
13                         //change part# to 443 for secure connection
14
15 #define MQTT_QoS   0   //MQTT QoS = 0: message will be delivered zero or once (default)
16                        //MQTT QoS = 1: message will be delivered at least once
17                        //MQTT QoS = 2: message will be delivered exactly once
18
19 //Use your device information to fill in the following code in line20~24
20 char *MQTT_HOST = "your_organizationID_here.messaging.internetofthings.ibmcloud.com";
21 char *MQTT_CLIENT = "d:your_organizationID_here:your_Device_Type_here:your_DeviceID_here";
22                   //an example of MQTT_CLIENT is "d:myorgID:photons:photon_a1";
23 char *MQTT_USERNAME = "use-token-auth";
24 char *MQTT_PASSWORD = "your_token_here";
25
26 // Set MQTT event topic for the data that will be published to Watson IoT Platform.
27 // iot-2    --> The protocol
28 // evt      --> Specifies the message type, use "cmd" for applications
29 // testevent--> Name of event        //it is also EventID. Important! Needed by cloud!
30 // fmt/json --> Message will be send in JSON format //Watson Platform uses JSON format as default
31 char *EVENT_TOPIC = "iot-2/evt/testevent/fmt/json";     // Segment can be customized by you
```

```
32
33 //Specify topic of the commands that Photon will subscribe (which is sent from IBM Cloud).
34 //Set CommandID as “testcommand”, which should match CommandType setting in Node-RED app
35 //char *COMMAND_TOPIC = "iot-2/cmd/testcommand/fmt/json"   //use json formatted commands
36 char *COMMAND_TOPIC = "iot-2/cmd/testcommand/fmt/string";  //use string formatted commands
37
38 MQTT client( MQTT_HOST, HOST_PORT, callback ); //Create a MQTT client with callback function
39
40 char payload[80];      //MQTT data message's payload. A string in JASON format.
41                        // 80Bytes used here. Change the size according to your data.
42
43 int dummydata = 0;
44
45 int LedIndicator_Publish = D7;   //set LED D7 as publishing indicator
46 int LedIndicator_CommLow = D6;   //set LED connected on D6 as indicator for command “turnlow”
47 int LedIndicator_CommMid = D5;   //set LED connected on D6 as indicator for command “turnmid”
48 int LedIndicator_CommHigh = D4;  //set LED connected on D6 as indicator for command “turnhigh”
49
50 void BlinkLed(int LedPin, int BlinkTimes, int BlinkPeriod);  //A func to blink a LED
51
52 void setup() {
53   pinMode(LedIndicator_Publish, OUTPUT);
54   pinMode(LedIndicator_CommLow, OUTPUT);
55   pinMode(LedIndicator_CommMid, OUTPUT);
56   pinMode(LedIndicator_CommHigh, OUTPUT);
57
58   BlinkLed(LedIndicator_CommLow, 1, 300);
59   BlinkLed(LedIndicator_CommMid, 1, 300);
60   BlinkLed(LedIndicator_CommHigh, 1, 300);
61
62   RGB.control(true);
63
64   Serial.begin( 9600 );    //Use serial monitor to debug the code.
65                            //Read “Particle serial tutorial” for detailed how-to.
66   Serial.println( "Connecting Photon to IBM Watson IoT Platform ..... " );
67
68   while( !Serial.available() ) {
69     Particle.process();
70   }
71
72   client.connect(MQTT_CLIENT,MQTT_USERNAME,MQTT_PASSWORD);//Connect Photon to Watson IoT Platform
73
74   if( client.isConnected() ) {              //Verify the connection
75     Serial.println( "Now connected!" );
76     client.subscribe(COMMAND_TOPIC);    //subscribe commands with CommandType specified in line36
77   }
78 }
79
80 void loop() {
81   dummydata ++;
82   if (dummydata > 100)  dummydata = 0;
83
84   //Convert data into a JSON formatted object by line86:
85   //JSON format example: {“property”: value1, ”property2”: value2,…}
86   sprintf( payload, "{ \"dataproperty\": \"%d\" }", dummydata );
87
88   //Publish the JSON formatted payload to IBM cloud under the pre-defined Event Topic in line31
89   client.publish( EVENT_TOPIC, const_cast<char*> (payload) );
90
91   BlinkLed(LedIndicator_Publish, 1, 200); //blink the publishing LED once when a data is sent
92
93   client.loop();
94
95   Serial.print( "Data being sent to Cloud: " ); //Display the sent data on SerialMonitor
96   Serial.println( dummydata );
97
98   delay( 5000 );  //delay 5sec: Set the period of publishing data to cloud
99 }
100
101 void callback( char* topic, byte* payload, unsigned int length ) {
102   RGB.color(255,10,255);  //blink the RGB LED in pink color
103   delay(300);
104   RGB.color(0,0,0);
105
106   char p[length + 1];
107
108   memcpy( p, payload, length );  //read in received command message payload
109   p[length] = NULL;
110
```

```
111   String message( p );
112
113   Serial.print( "Received command from Cloud: ");  //display received command on serial terminal
114   Serial.println(p);
115
116  //Line118: to parse received json object to obtain commands. If the commands are in string
117  //format, no need of this.
118  // String CmdStr = RdJson::getString("testcommand", "", p.c_str());
119  // Serial.printlnf("Received Command from cloud: %s", CmdStr.c_str());
120
121  //Line122-138: Take diff actions according to received commands. Make changes for your needs.
122   if (!strcmp(p, "turnlow"))
123   {
124       BlinkLed(LedIndicator_CommLow, 1, 300);
125   }
126   else if (!strcmp(p, "turnmid"))
127   {
128       BlinkLed(LedIndicator_CommMid, 1, 300);
129   }
130   else if (!strcmp(p, "turnhigh") )
131   {
132       BlinkLed(LedIndicator_CommHigh, 1, 300);
133   }
134   else {
135       RGB.color(255,255,255); delay(1000); RGB.color(0,0,0);  //Blink the RGB LED in white
136       BlinkLed(LedIndicator_CommLow, 1, 100);
137       BlinkLed(LedIndicator_CommMid, 1, 100);
138       BlinkLed(LedIndicator_CommHigh, 1, 100);
139   }
140 }
141
142 void BlinkLed(int LedPin, int BlinkTimes, int BlinkPeriod) //A func to blink a LED for BlinkTimes
143 {
144     for (int k=0; k<BlinkTimes; k++){
145         digitalWrite(LedPin, HIGH);
146         delay(BlinkPeriod);
          digitalWrite(LedPin, LOW);
          delay(BlinkPeriod);
      }
  }
```

Note: Refer to Particle Serial Tutorial to learn how to use Serial Monitor with Particle for testing and debugging.
Link: https://community.particle.io/t/serial-tutorial/26946
or: https://github.com/rickkas7/serial_tutorial

References

1. “MQTT connectivity for devices”, IBM Cloud Docs,
https://console.bluemix.net/docs/services/IoT/devices/mqtt.html#mqtt
2. “Getting started with Watson IoT Platform using Node-RED”, IBM developerWorks Recipes,
https://developer.ibm.com/recipes/tutorials/getting-started-with-watson-iot-platform-using-node-red/
3. https://console.bluemix.net/docs/services/IoT/platform_authorization.html#connecting-applications
4. https://github.com/hirotakaster/MQTT