

# Sprint 2

## Code:

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 20, 4);

#define ORG "x9oggs"
#define DEVICE_TYPE "iot_device"
#define DEVICE_ID "1234"
#define TOKEN "12345678"

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/data/fmt/json";
char topic[] = "iot-2/cmd/led/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

WiFiClient wifiClient;
PubSubClient client(server, 1883, wifiClient);

#define ECHO_PIN 12
#define TRIG_PIN 13
float dist;

void setup()
{
    Serial.begin(115200);
    pinMode(LED_BUILTIN, OUTPUT);
    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
    //pir pin
    pinMode(34, INPUT);

    //ledpins
    pinMode(23, OUTPUT);
    pinMode(2, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(15, OUTPUT);

    lcd.init();
    lcd.backlight();
    lcd.setCursor(1, 0);
    lcd.print("");
    wifiConnect();
    mqttConnect();
}

float readcmCM()
{
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
```

Team ID: PNT2022TMID19008

```
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    int duration = pulseIn(ECHO_PIN, HIGH);
    return duration * 0.034 / 2;
}

void loop()
{
    lcd.clear();

    publishData();
    delay(500);
    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("IBM subscribe to cmd OK");
    }
    else
    {
        Serial.println("subscribe to cmd FAILED");
    }
}

void publishData()
```

Team ID: PNT2022TMID19008

```
{
  float cm = readcmCM();

  if(digitalRead(34))
  {
    Serial.println("Motion Detected");
    Serial.println("Lid Opened");
    digitalWrite(15, HIGH);

    if(digitalRead(34)== true)
    {
      if(cm <= 60)
      {
        digitalWrite(2, HIGH);
        Serial.println("High Alert!!!,Trash bin is about to be full");
        Serial.println("Lid Closed");
        lcd.print("Full! Don't use");
        delay(2000);
        lcd.clear();
        digitalWrite(4, LOW);
        digitalWrite(23, LOW);
      }
      else if(cm > 60 && cm < 120)
      {
        digitalWrite(4, HIGH);
        Serial.println("Warning!!,Trash is about to cross 50% of bin level");
        digitalWrite(2, LOW);
        digitalWrite(23, LOW);
      }
      else if(cm > 120)
      {
        digitalWrite(23, HIGH);
        Serial.println("Bin is available");
        digitalWrite(2,LOW);
        digitalWrite(4, LOW);
      }
      delay(10000);
      Serial.println("Lid Closed");
    }
    else
    {
      Serial.println("No motion detected");
      digitalWrite(2, LOW);
      digitalWrite(15, LOW);
      digitalWrite(4, LOW);
      digitalWrite(23, LOW);
    }
  }
  else
  {
    digitalWrite(15, LOW);
  }

  if(cm <= 60)
  {
```

Team ID: PNT2022TMID19008

```
digitalWrite(21,HIGH);
String payload = "{\"High_Alert\":\"";
payload += cm;
payload += " }";
Serial.print("\n");
Serial.print("Sending payload: ");
Serial.println(payload);

if (client.publish(publishTopic, (char*) payload.c_str()))
{
    Serial.println("Publish OK");
}
}
else if(cm <= 120)
{
    digitalWrite(22,HIGH);
    String payload = "{\"Warning\":\"";
    payload += cm ;
    payload += " }";
    Serial.print("\n");
    Serial.print("Sending payload: ");
    Serial.println(payload);
    if(client.publish(publishTopic, (char*) payload.c_str()))
    {
        Serial.println("Publish OK");
    }
    else
    {
        Serial.println("Publish FAILED");
    }
}
else
{
    Serial.println();
}

float inches = (cm / 2.54);
lcd.setCursor(0,0);
lcd.print("Inches");
lcd.setCursor(4,0);
lcd.setCursor(12,0);
lcd.print("cm");
lcd.setCursor(1,1);
lcd.print(inches, 1);
lcd.setCursor(11,1);
lcd.print(cm, 1);
lcd.setCursor(14,1);
delay(1000);
lcd.clear();
}
```

Link:

<https://wokwi.com/projects/348051630039499347>

## Screenshots:

The screenshot shows the Wokwi IDE interface. On the left, the code editor displays the following C++ code:

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 #include <LiquidCrystal_I2C.h>
4 LiquidCrystal_I2C lcd(0x27, 20, 4);
5
6 #define ORG "x9oggs"
7 #define DEVICE_TYPE "iot_device"
8 #define DEVICE_ID "1234"
9 #define TOKEN "12345678"
10
11
12 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
13 char publishTopic[] = "iot-2/evt/data/fmt/json";
14 char topic[] = "iot-2/cmd/led/fmt/String";
15 char authMethod[] = "use-token-auth";
16 char token[] = TOKEN;
17 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
18
19
20
21 WiFiClient wificlient;
22 PubSubClient client(server, 1883, wificlient);
23
24 #define ECHO_PIN 12
25 #define TRIG_PIN 13
26 float dist;
27
28 void setup()
29 {
30   Serial.begin(115200);
```

The simulation window on the right shows a 3D model of the ESP32 board connected to an LCD and an ultrasonic sensor. The console log displays the following messages:

```
Connecting to Wifi.....WiFi connected, IP address: 10.10.0.2
Reconnecting MQTT client to x9oggs.messaging.internetofthings.ibmcloud.com
IBM subscribe to cmd OK
```

The screenshot shows the IBM Watson IoT Device Drilldown page for device 1234. The page displays the following information:

- Client Address:** 50.31.197.64 Insecure
- Recent Events:** A table showing the live stream of data coming and going from the device.
- State:** A table showing a list of data points reported by the device.

Event	Value	Format	Last Received
data	[\"High_Alert\":47.97]	json	a few seconds ago
data	[\"High_Alert\":47.97]	json	a minute ago
data	[\"High_Alert\":47.97]	json	a minute ago
data	[\"High_Alert\":47.97]	json	a minute ago
data	[\"High_Alert\":47.97]	json	a minute ago

Property	Value	Type	Event	Last Received
High_Alert	47.97	Number	State	a few seconds ago

0 Simulations running