

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);
```

```

    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()

```

```

{
  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)
  {

```

```

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 an Arduino sketch for an ESP32. The code includes logic for reading an ultrasonic sensor, publishing data to an MQTT topic, and controlling an LCD display. The right side of the interface shows a simulation of the hardware, including an ESP32 board, an ultrasonic sensor, and an LCD display showing 'Inches 18.9' and 'cm 48.0'. The console output shows the device connecting to Wi-Fi, subscribing to an MQTT topic, and sending payloads.

```
179 }
180
181 if (cn <= 60)
182 {
183   digitalWrite(21,HIGH);
184   String payload = "{\"High_Alert\":\"";
185   payload += cn;
186   payload += "\"}";
187   Serial.print("Sending payload: ");
188   Serial.println(payload);
189   if (client.publish(publishTopic, (char*) payload_c_str()))
190   {
191     Serial.println("Publish OK");
192   }
193   else if (cn <= 120)
194   {
195     digitalWrite(22,HIGH);
196     String payload = "{\"Warning\":\"";
197     payload += cn;
198     payload += "\"}";
199     Serial.print("Sending payload: ");
200     Serial.println(payload);
201     if (client.publish(publishTopic, (char*) payload_c_str()))
202     {
203       Serial.println("Publish OK");
204     }
205     else
206     {
207       Serial.println("Publish FAILED");
208     }
209   }
210   else
211   {
212     Serial.println("Publish FAILED");
213   }
214   }
215   else
216   {
217     Serial.println();
218   }
219
220 float inches = (cn / 2.54);
221 lcd.setCursor(0,0);
222 lcd.print("Inches");
223 lcd.setCursor(4,0);
224 lcd.setCursor(10,0);
225 lcd.print("cm");
226 lcd.setCursor(1,1);
227 lcd.print(inches, 1);
228 lcd.setCursor(11,1);
229 lcd.print(cn, 1);
230 delay(1000);
231 lcd.clear();
232 }
```

The screenshot shows the IBM Watson IoT Platform interface. The page is titled 'Device Drilldown - 1234'. It displays the client address '50.31.197.64 Insecure'. The 'Recent Events' section shows a list of events with columns for Event, Value, Format, and Last Received. The 'State' section shows a table of data points reported by the device. The bottom right corner indicates '0 Simulations running'.

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	data	a few seconds ago