

Delivery of Sprint-2

DATE	05/11/2022
TEAM ID	PNT2022TMID52815
PROJECT NAME	SMART WASTE MANAGEMENT FOR METROPOLITAN CITIES

Code for Data Transfer from Sensors

```
#include <WiFi.h> // library for wifi
#include <PubSubClient.h> // library for MQTT
#include <LiquidCrystal_I2C.h>
#include <mjson.h>
LiquidCrystal_I2C lcd(0x27, 20, 4);

//----- credentials of IBM Accounts -----

#define ORG "gw0bk3" // IBM organisation id
#define DEVICE_TYPE "NodeMCU" // Device type mentioned in ibm watson
iot platform
#define DEVICE_ID "Anu" // Device ID mentioned in ibm watson iot
platform
#define TOKEN "123456789" // Token

//----- customise above values -----
-----

char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; //
server name
char publishTopic[] = "iot-2/evt/data/fmt/json"; //
topic name and type of event perform and format in which data to be send
char topic[] = "iot-2/cmd/led/fmt/String"; // cmd
Represent type and command is test format of strings
char authMethod[] = "use-token-auth"; //
authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //Client
id

//-----
-----

WiFiClient wifiClient; //
creating instance for wificlient
PubSubClient client(server, 1883, wifiClient);

#define ECHO_PIN 12
```

```

#define TRIG_PIN 13
float dist;
String data3;
bool SealBin = true;
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();
    }
}

```

// function call to connect to IBM

```

}

/* -----retrieving to cloud-----
-----*/
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)) //pir motion detection

```

```

{
    Serial.println("Motion Detected");
    Serial.println("Lid Opened");
    digitalWrite(15, HIGH);
    if(digitalRead(34)== true)
    {
        if(cm <= 60)                                //Bin level detection
        {
            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()))           // if data is
        uploaded to cloud successfully,prints publish ok else prints publish failed
        {
            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 if(cm > 120)
    {
        digitalWrite(23,HIGH);
        String payload = "{";
        payload += cm;
        payload += " }";
        Serial.print("\n");
        Serial.print("Sending payload: ");
        Serial.println(payload);

        if (client.publish(publishTopic, (char*) payload.c_str()))           // if data is
        uploaded to cloud successfully,prints publish ok else prints publish failed

```

```

{
Serial.println("Publish OK");
}

}

float inches = (cm / 2.54); //print on lcd
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();
}

//handles commands from user side

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++) {

    data3 += (char)payload[i];
}
Serial.println("data: "+ data3);

const char *s =(char*) data3.c_str();
double pincode = 0;

const char *buf;
int len;

if (mjson_find(s, strlen(s), ".$command", &buf, &len)) // And print it
{

    String command(buf,len);

    if(command=="\"SealBin\"")
    {
        SealBin = true;
    }
}
}

```

}

}

data3="";

}

WOKWI

esp32-blink.ino

```
1 #include <WiFi.h> // library for wifi
2 #include <PubSubClient.h> // library for MQTT
3 #include <LiquidCrystal_I2C.h>
4 #include <json.h>
5 LiquidCrystal_I2C lcd(0x27, 20, 4);
6
7 //----- credentials of IBM Accounts -----
8
9 #define ORG "gw0bk3" // IBM organisation id
10 #define DEVICE_TYPE "NodeMCU" // Device type mentioned in ibm watson iot platform
11 #define DEVICE_ID "Anu" // Device ID mentioned in ibm watson iot platform
12 #define TOKEN "123456789" // Token
13
14 //----- customise above values -----
15
16 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // server name
17 char publishTopic[] = "iot-2/evt/data/fmt/json"; // topic name
18 char topic[] = "iot-2/cmd/led/fmt/string"; // cmd Rep
19 char authMethod[] = "use-token-auth"; // authentication
20 char token[] = TOKEN;
21 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //Client id
22
23 //-----
24
25 WiFiClient wificlient; // creating
26 PubSubClient client(server, 1883, wificlient);
27
28 #define ECHO_PIN 12
29 #define TRIG_PIN 13
30 float dist;
31 String data3;
32 bool SealBin = true;
33 void setup()
34
```

Simulation

00:28.255 99%

Publish OK

Sending payload: {169.97 }

Publish OK

Sending payload: {169.97 }

Publish OK

IBM Watson IoT Platform

1904067ece@cit.edu.in ID: gw0bk3

Browse Action Device Types Interfaces

Search by Device ID

Device Simulator

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
1904	Disconnected	Smart_waste_management_system	Device	15 Nov 2022 4:27 PM	
Anu	Disconnected	NodeMCU	Device	15 Nov 2022 4:19 PM	

Identity Device Information Recent Events State Logs

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
data	{ "type": "Buffer", "data": [123,51,50,54,46,57,52,...	json	a few seconds ago
data	{ "type": "Buffer", "data": [123,51,50,54,46,57,54,...	json	a few seconds ago
data	{ "type": "Buffer", "data": [123,51,50,54,46,57,54,...	json	a few seconds ago
data	{ "type": "Buffer", "data": [123,51,50,54,46,57,52,...	json	a few seconds ago
data	{ "type": "Buffer", "data": [123,51,50,54,46,57,52,...	json	a few seconds ago

