DELIVERY OF SPRINT – 3

|  |  |
| --- | --- |
| Team ID | PNT2022TMID17949 |
| Project Name | Project –Smart Waste management System |
| Date | 11 November 2022 |
| Marks | 4 Marks |

**Transferring the data from the sensors to the**

**IBM IoT Watson cloud**

1. **Ino code :**

#include <WiFi.h>                              // library for wifi

#include <PubSubClient.h>                      // library for MQTT

#include <LiquidCrystal\_I2C.h>

LiquidCrystal\_I2C lcd(0x27, 20, 4);

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

#define ORG "tn3xmm"                        // IBM organisation id

#define DEVICE\_TYPE "rojjer"                // Device type mentioned in ibm watson iot platform

#define DEVICE\_ID "240901"              // Device ID mentioned in ibm watson iot platform

#define TOKEN "dVDVCxWLOW7)W6vwa&"          // 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;

void setup()

{

  Serial.begin(115200);

  pinMode(LED\_BUILTIN, OUTPUT);

  pinMode(TRIG\_PIN, OUTPUT);

  pinMode(ECHO\_PIN, INPUT);

  //pir pin

  pinMode(4, 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);

  }

  else

  {

    digitalWrite(15, LOW);

  }

 if(digitalRead(34)== true)

 {

  if(cm <= 100)                                                //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 > 150 && cm < 250)

  {

    digitalWrite(4, HIGH);

    Serial.println("Warning!!,Trash is about to cross 50% of bin level");

    digitalWrite(2, LOW);

    digitalWrite(23, LOW);

  }

  else if(cm > 250 && cm <=400)

  {

    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");

 }

  if(cm <= 100)

{

digitalWrite(21,HIGH);

String payload = "{\"HighAlert !Trash bin is about to be full\":\"";

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");

}

}

///////////////////////////

if(cm > 150 && cm < 250)

{

digitalWrite(22,HIGH);

String payload = "{\"warning! Trash is about to cross 50% of bin level\":\"";

payload += cm;

payload += "\" }";

Serial.print("\n");

Serial.print("Sending distance: ");

Serial.println(cm);

if(client.publish(publishTopic, (char\*) payload.c\_str()))

{

Serial.println("Publish OK");

}

else

{

Serial.println("Publish FAILED");

}

}

///////////////////////////

if(cm > 250 && cm <=400)

{

digitalWrite(21,HIGH);

String payload = "{\"Bin is available\":\"";

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();

}

1. **Json file :**

{

  "version": 1,

  "author": "Uri Shaked",

  "editor": "wokwi",

  "parts": [

    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 1.29, "left": -1.29, "attrs": {} },

    {

      "type": "wokwi-led",

      "id": "led1",

      "top": -43.97,

      "left": 296.62,

      "attrs": { "color": "limegreen" }

    },

    {

      "type": "wokwi-led",

      "id": "led2",

      "top": 15.48,

      "left": 299.36,

      "attrs": { "color": "yellow" }

    },

    {

      "type": "wokwi-led",

      "id": "led3",

      "top": 140.83,

      "left": 302.1,

      "attrs": { "color": "blue" }

    },

    {

      "type": "wokwi-led",

      "id": "led4",

      "top": 79.19,

      "left": 300.24,

      "attrs": { "color": "red" }

    },

    {

      "type": "wokwi-resistor",

      "id": "r1",

      "top": -3.9,

      "left": 224.81,

      "attrs": { "value": "100" }

    },

    {

      "type": "wokwi-resistor",

      "id": "r2",

      "top": 55.55,

      "left": 221.42,

      "attrs": { "value": "100" }

    },

    {

      "type": "wokwi-resistor",

      "id": "r3",

      "top": 179.36,

      "left": 221.1,

      "attrs": { "value": "100" }

    },

    {

      "type": "wokwi-resistor",

      "id": "r4",

      "top": 119.28,

      "left": 220.77,

      "attrs": { "value": "100" }

    },

    {

      "type": "wokwi-lcd1602",

      "id": "lcd1",

      "top": 248.08,

      "left": 161.61,

      "attrs": { "pins": "i2c" }

    },

    {

      "type": "wokwi-hc-sr04",

      "id": "ultrasonic1",

      "top": 13.99,

      "left": -295.33,

      "attrs": { "distance": "248" }

    },

    {

      "type": "wokwi-pir-motion-sensor",

      "id": "pir1",

      "top": -147.86,

      "left": -88.23,

      "attrs": {}

    }

  ],

  "connections": [

    [ "esp:TX0", "$serialMonitor:RX", "", [] ],

    [ "esp:RX0", "$serialMonitor:TX", "", [] ],

    [ "led1:A", "r1:2", "green", [ "v0" ] ],

    [ "led2:A", "r2:2", "yellow", [ "v0" ] ],

    [ "led4:A", "r4:2", "red", [ "v0" ] ],

    [ "led3:A", "r3:2", "blue", [ "v0" ] ],

    [ "led1:C", "esp:GND.1", "black", [ "v-2.56", "h-170.98", "v116.48" ] ],

    [ "led2:C", "esp:GND.1", "black", [ "v-2.24", "h-173.72", "v91.96" ] ],

    [ "led4:C", "esp:GND.1", "black", [ "v-3.11", "h-174.6", "v27.59" ] ],

    [ "led3:C", "esp:GND.1", "black", [ "v-1.92", "h-177.99", "v-32.18" ] ],

    [ "r1:1", "esp:D23", "green", [ "v2.63", "h-71.91", "v19.92" ] ],

    [ "r3:1", "esp:D15", "blue", [ "v0.22", "h-89.65", "v-53.64" ] ],

    [ "lcd1:GND", "esp:GND.1", "black", [ "h-26.5", "v-129.82" ] ],

    [ "pir1:VCC", "esp:3V3", "red", [ "v268.96", "h172.77", "v-55.17" ] ],

    [ "pir1:GND", "esp:GND.2", "black", [ "v0" ] ],

    [ "pir1:OUT", "esp:D34", "green", [ "v0" ] ],

    [ "ultrasonic1:GND", "esp:GND.2", "black", [ "v0" ] ],

    [ "ultrasonic1:ECHO", "esp:D12", "yellow", [ "v0" ] ],

    [ "ultrasonic1:TRIG", "esp:D13", "green", [ "v0" ] ],

    [ "ultrasonic1:VCC", "esp:VIN", "red", [ "v0" ] ],

    [ "r4:1", "esp:D2", "red", [ "v0" ] ],

    [ "r2:1", "esp:D4", "yellow", [ "v0" ] ],

    [ "lcd1:SDA", "esp:D21", "cyan", [ "h-27.12", "v-252.33", "h-16.71", "v17.15" ] ],

    [ "lcd1:SCL", "esp:D22", "white", [ "h-36.27", "v-3.67" ] ],

    [ "lcd1:VCC", "esp:VIN", "red", [ "h-187.87", "v-129.69" ] ]

  ]

}

1. **Libraries file:**

# Wokwi Library List

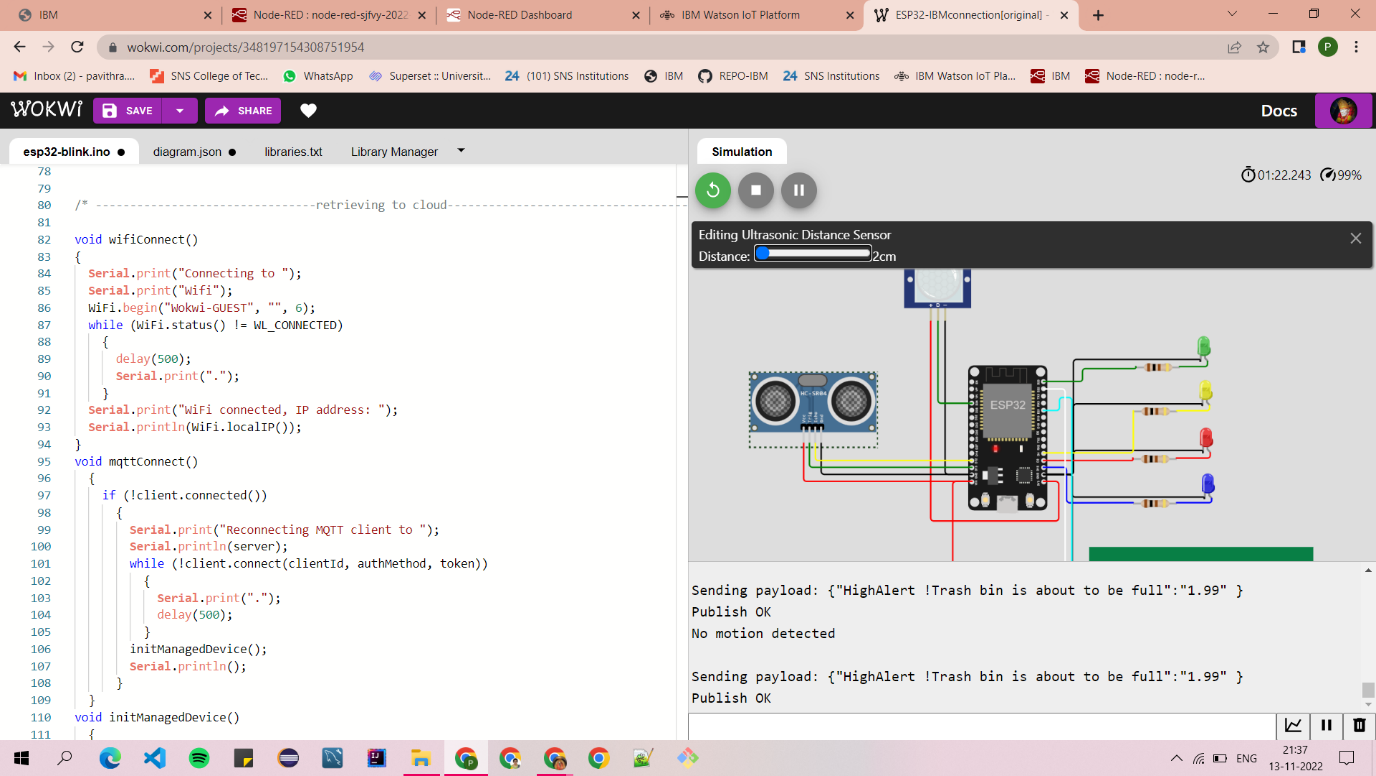
# See https://docs.wokwi.com/guides/libraries

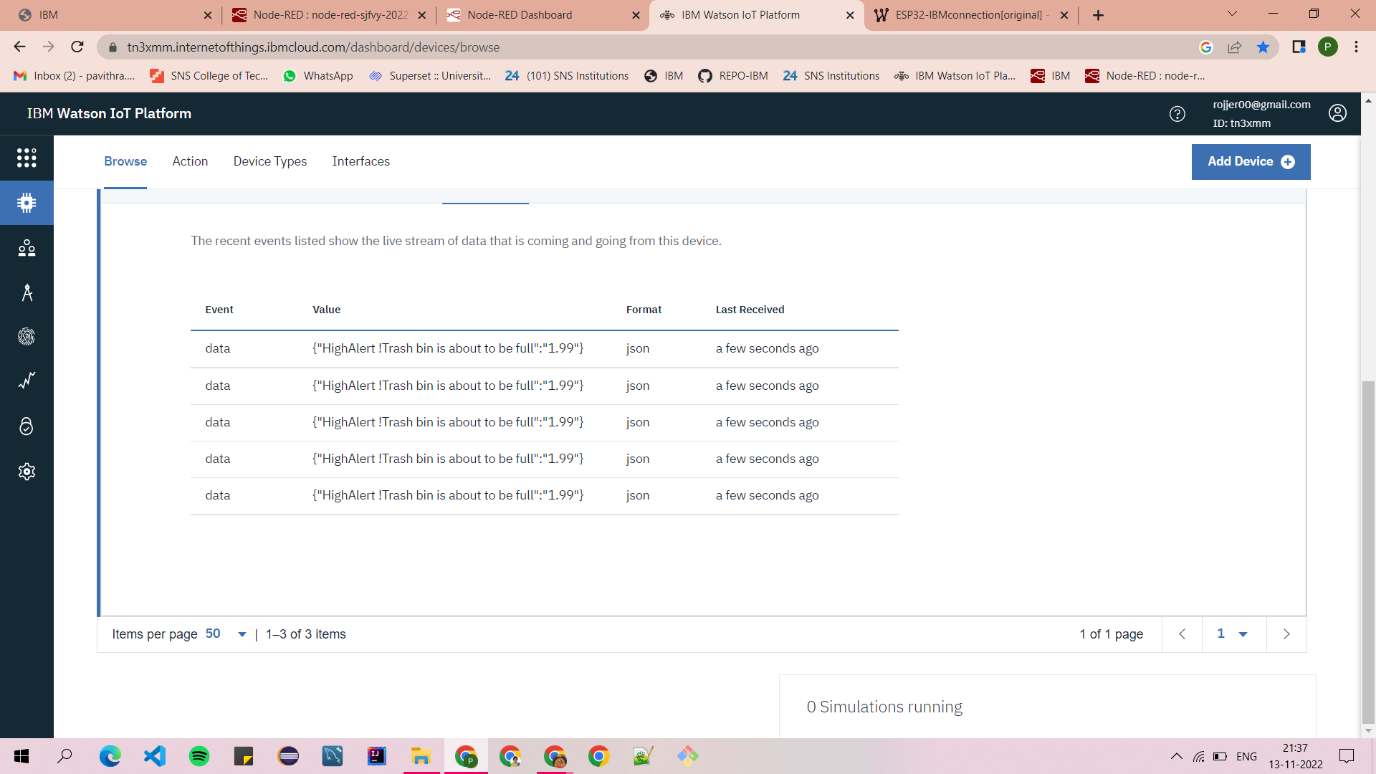
WiFi

PubSubClient

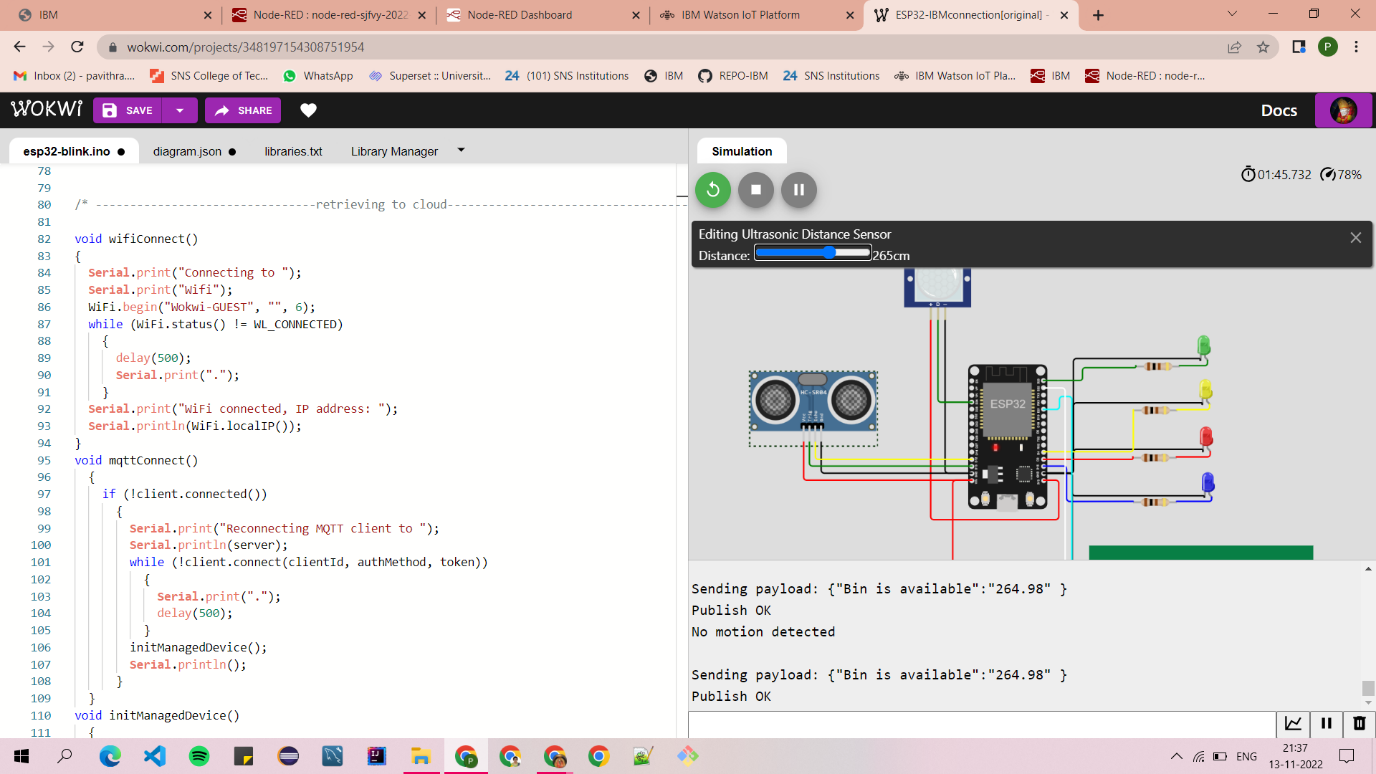
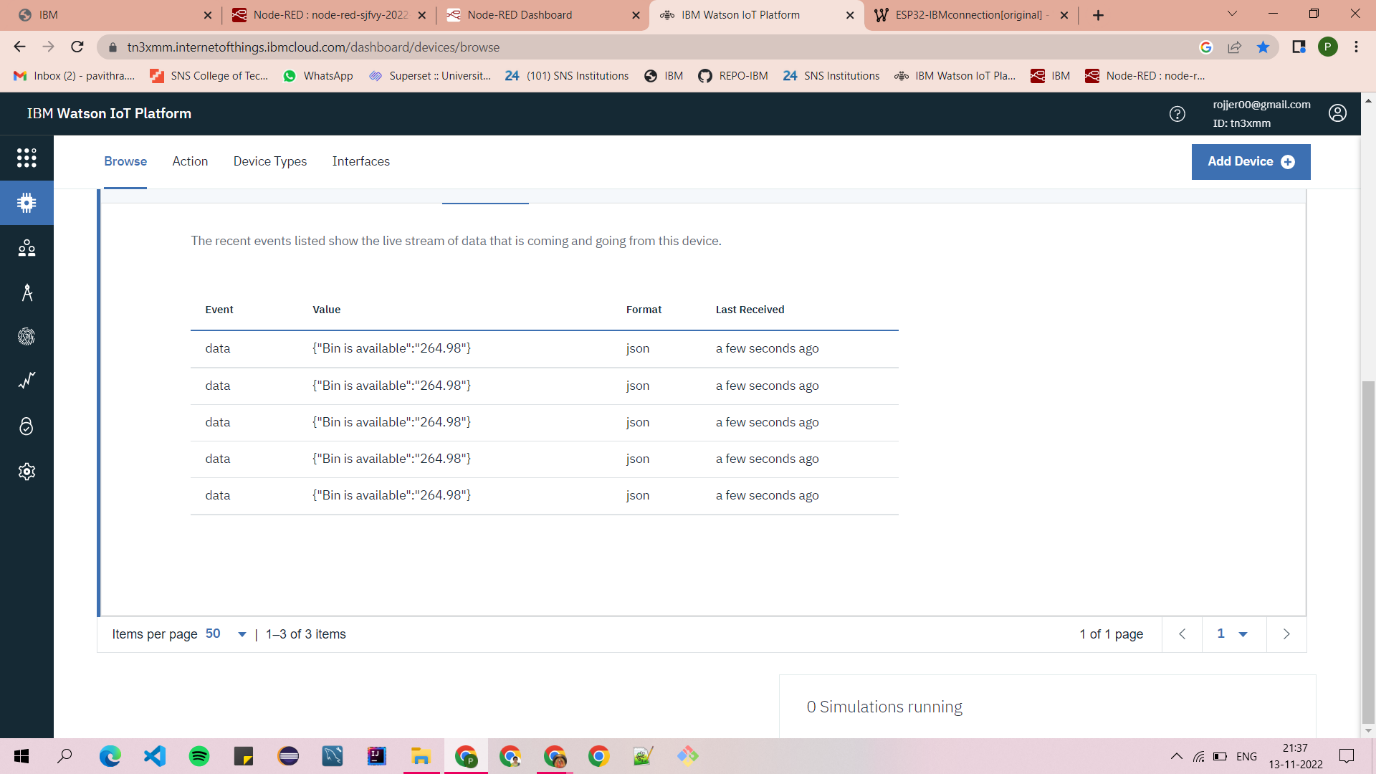
LiquidCrystal I2C

1. **Output on cloud, when Bin is fulled/ about to get filled**

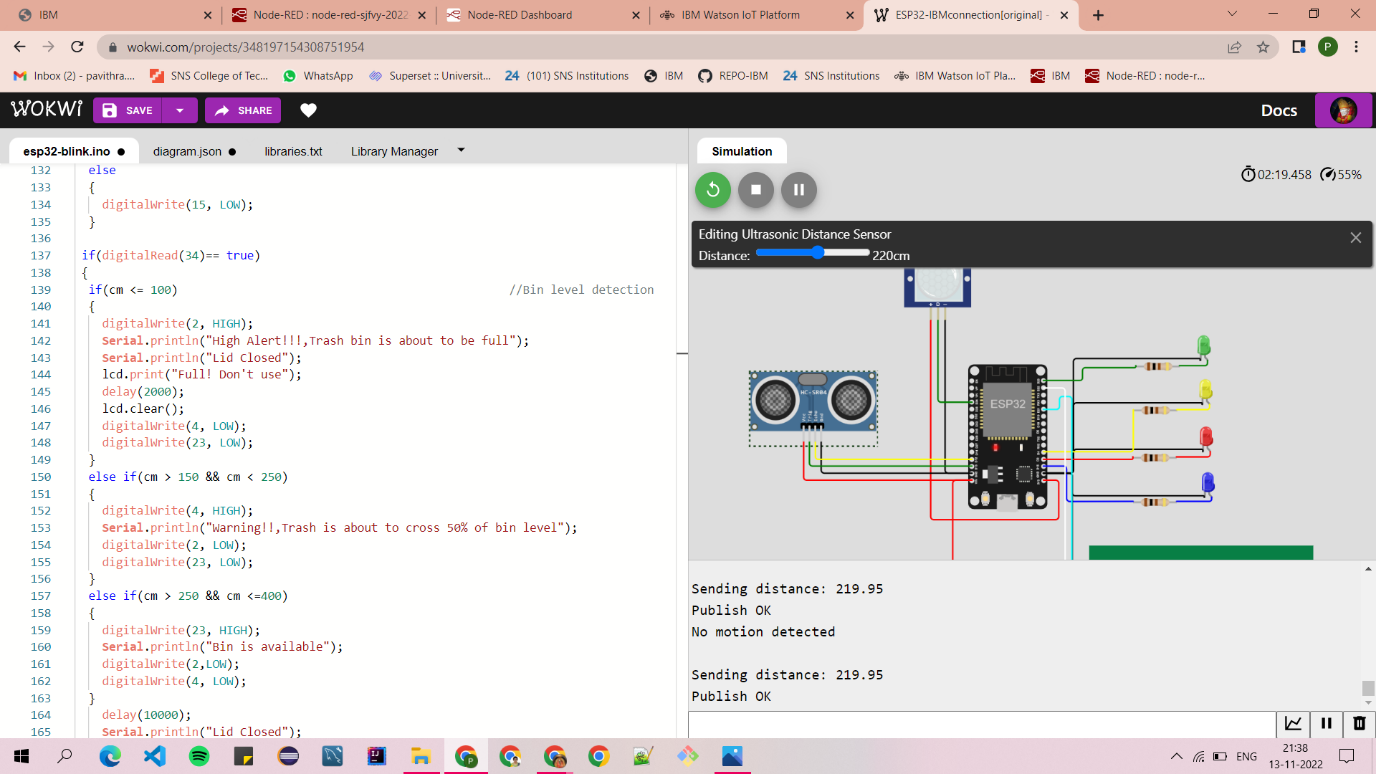
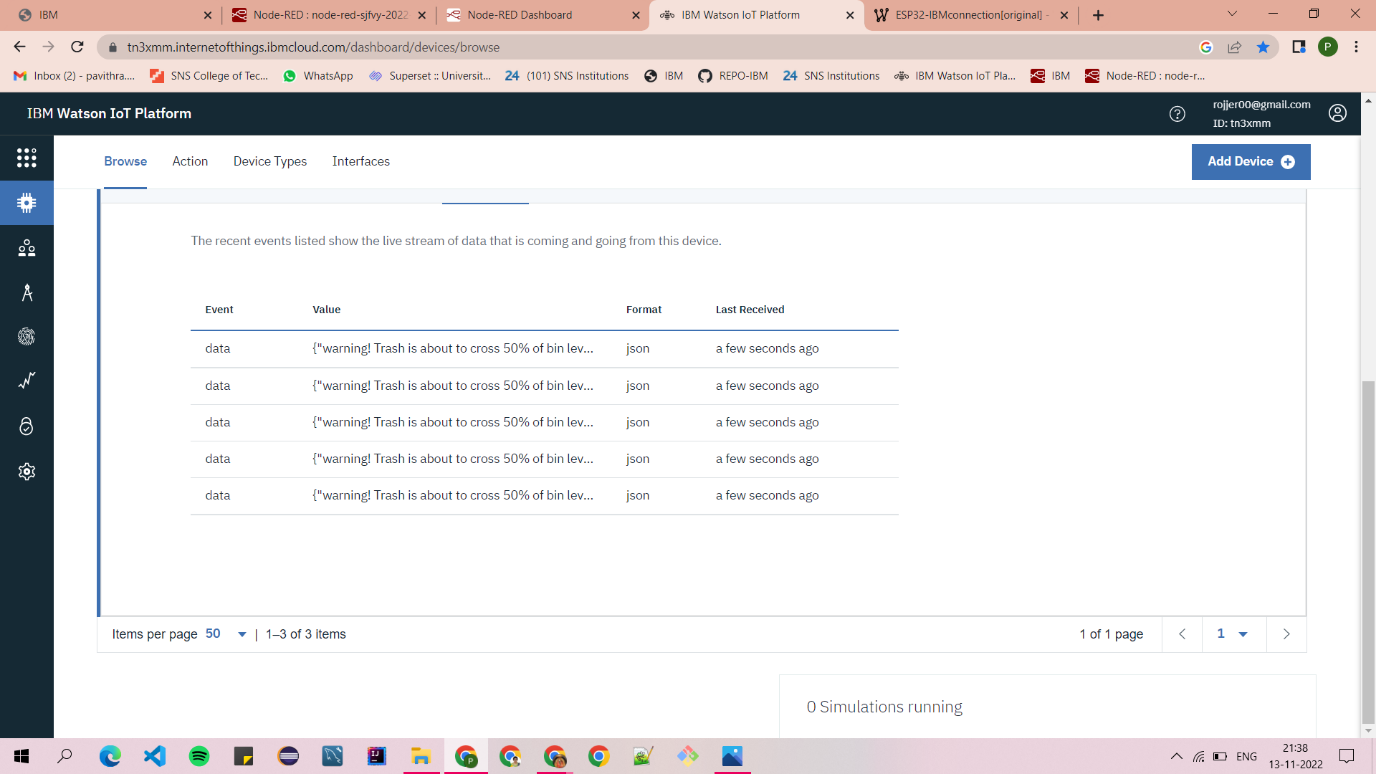
****

****

1. **Output on cloud, when Bin is Empty**

****

1. **Output on cloud, when Bin is about to cross 50% of storage**

****