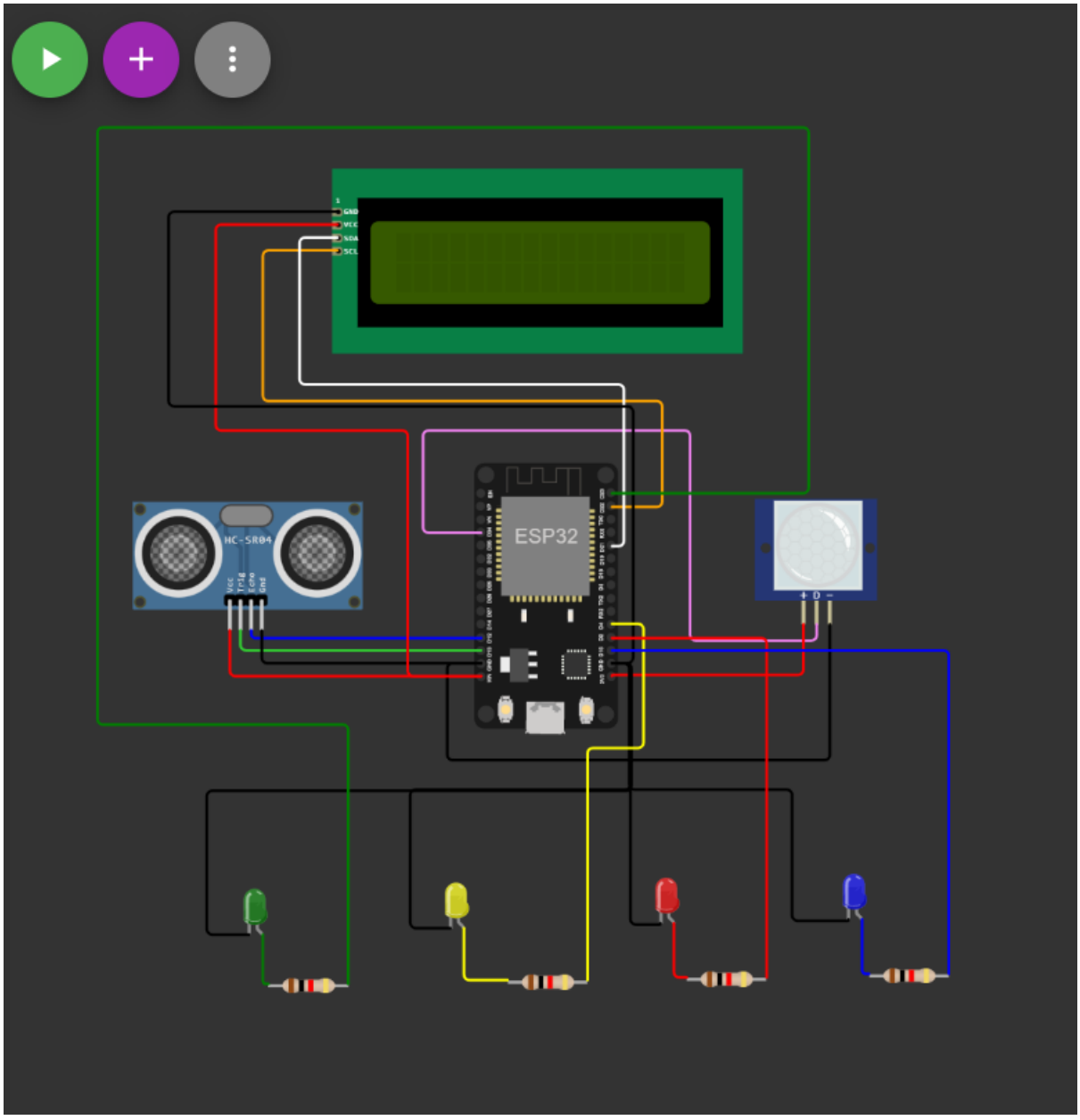
Sprint-1
Simulation Creation (Connect Sensor Arduino with Python code)

Date	29 October 2022
Team ID	**PNT2022TMID46943**
Project Name	Smart Waste Management System For Metropolitan Cities
Maximum Marks	20 Marks

Wokwi Link:

https://wokwi.com/projects/347927859012043348

Wokwi Simulation:



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 "jrbl5n"                           // IBM organisation id
#define DEVICE_TYPE "Assignment4"                 // Device type mentioned in
ibm watson iot platform
#define DEVICE_ID "12345"                // Device ID mentioned in ibm watson
iot platform
#define TOKEN "12345678"           // 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(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
{
Serial.println();
}

  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();
}
```

Python Code:

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random


#Provide your IBM Watson Device Credentials
organization = "jrbl5n"
deviceType = "Arduino"
deviceId = "12345"
authMethod = "token"
authToken = "12345678"

# Initialize GPIO


def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="lighton":
        print ("led is on")
    else :
        print ("led is off")

    #print(cmd)
```

```
try:
  deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}
  deviceCli = ibmiotf.device.Client(deviceOptions)
  #.....................................


except Exception as e:
  print("Caught exception connecting device: %s" % str(e))
  sys.exit()


# Connect and send a datapoint "hello" with value "world" into the cloud as an event of type "greeting" 10 times
deviceCli.connect()


while True:
        #Get Sensor Data from esp32


        weightSensor=random.randint(0,100)
        irSensor=random.randint(0,100)
        ultrasSensor=random.randint(0,100)


        data = { 'WeightSensors' : weightSensor, 'IRSensor':irSensor, 'Ultrasonic Sensor':ultrasSensor  }
        #print data
        def myOnPublishCallback():
```

```
        print ("Published Weight of Trashcan is = %s C" % weightSensor, "IR
Sensor = %s %%" % irSensor, "Ultrasonic Sensor = %s %%" % ultrasSensor,
"to IBM Watson")


    success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,
on_publish=myOnPublishCallback)
    if not success:
        print("Not connected to IoTF")
    time.sleep(1)


    deviceCli.commandCallback = myCommandCallback


# Disconnect the device and application from the cloud
deviceCli.disconnect()
```