

FINAL CODE

Team ID	PNT2022TMID34114
Project Name	Smart Waste Management for Metropolitan cities

Final code

```
<!DOCTYPE html>

<html>

<head>

<title>Registration system PHP and MySQL</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<div class="header">

<h2>Register</h2>

</div>

<form method="post" action="register.php">

<div class="input-group">

<label>Username</label>

<input type="text" name="username" value="">

</div>

<div class="input-group">

<label>Email</label>

<input type="email" name="email" value="">

</div>
```

```
<div class="input-group">
    <label>Password</label>
    <input type="password" name="password_1">
</div>
<div class="input-group">
    <label>Confirm password</label>
    <input type="password" name="password_2">
</div>
<div class="input-group">
    <button type="submit" class="btn"
name="register_btn">Register</button>
</div>
<p>
    Already a member? <a href="login.php">Sign in</a>
</p>
</form>
</body>
</html>
```

Style.css

```
* { margin: 0px;
padding: 0px; } body {
    font-size: 120%;
    background:
#F8F8FF;
}
.header {
```

```
        width: 40%;
        margin: 50px
auto 0px;
        color: white;
        background:
#5F9EA0;      text-
align: center;
        border: 1px solid
#B0C4DE;
        border-bottom:
none; border-radius:
10px 10px 0px 0px;

        padding: 20px;
    }

form, .content {

        width: 40%;
        margin: 0px auto;
        padding: 20px;
        border: 1px solid
#B0C4DE;
        background: white;
        border-radius: 0px 0px
10px 10px;
    }

.input-group {

margin: 10px 0px 10px 0px;
}

.input-group label {

        display: block;
        text-align:
```

```
left;    margin:
3px;

}

.input-group input {
    height: 30px;
    width: 93%;
    padding: 5px
10px;    font-
size: 16px;
    border-
radius: 5px;
    border: 1px
solid gray;
}

#user_type {
    height: 40px;
    width: 98%;
    padding: 5px
10px;
    background:
white;    font-
size: 16px;
    border-
radius: 5px;
    border: 1px
solid gray;
}

.
b
t
n
{
```

```
padding:
10px; font-
size: 15px;
color: white;
background:
#5F9EA0;
border: none;
border-
radius: 5px;
}

.error {
width: 92%;
margin: 0px auto;
padding: 10px;
border: 1px solid
#a94442;
color: #a94442;
background:
#f2dede;
border-radius:
5px; text-align:
left;
}

.success {
color: #3c763d;
background:
#dff0d8;
border: 1px solid
#3c763d;
margin-bottom:
20px;
```

```

}

.profile_info img {
    display:
inline-block;
    width:
50px;
    height:
50px;
    margin:
5px;    float:
left;
}

.profile_info div {
    display: inline-
block;    margin: 5px;
}

.profile_info:af
ter {
    content: "";
    display:
block;    clear:
both;
}

```

PYTHON CODE FOR TRACKING LIVE LOCATION OF THE BIN

```

import wiotp.sdk.device
import time import
random myConfig = {
    "identity":{
        "orgId":"j5bxb7",

```

```

        "typeId":"IOT123edevicetype",
        "deviceId":"IOTece4"
    },
    "auth": {
        "token":"e2)-17xkqIFMvm3@II"
    }
}

def myCommandCallback(cmd):
    print("Message received from IBM IoT
Platform:%s"%cmd.data['command'])
    m=cmd.data['command']
    client=wiotp.sdk.device.DeviceClient(config=myConfig,logHandlers=None)
    client.connect()
    def pub(data):
        client.publishEvent(eventId="binstatus",msgFormat="json",data="myData",qos
        =0,onPublish=None)
        print("Published data Successfully:%s",myData)
    while True:
        myData={'name':'Bin1','lat':13.092677,'lon':80.188314}
        pub(myData)    time.sleep(3)
        client.commandCallback=myCommandCallback
    client.disconnect()

```

CODE FOR DATA TRANSFER FROM SENSORS:

```

#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 "j5bxb7" //IBM organisation id
#define DEVICE_TYPE "IOT123edevicetype" // Device type mentioned in ibm
watson iot platform
#define DEVICE_ID "IOTece4" // Device ID mentioned in ibm watson iot
platform
#define TOKEN "e2)-17xkqIFMvm3@II" // Token
// customise above values - char server[] = ORG
".messaging.internetofthings.ibmcloud.com"; // server name char
publishTopic[] = "iot-2/evt/data/fmt/json";
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 = "{\nHigh
Alert!!\":\ \""; payload += cm;
payload += "left\n }";
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 or prints publish failed
{

```

```

Serial.println("Publish OK");
}
}
if(cm <= 250)
{
digitalWrite(22,HIGH);
String payload =
"{\"Warning!!\":\":"; payload
+= dist; payload += "left\" }";
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");
}
}
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 SCRIPT:

```
import requests  
import json  
import ibmiotf.application  
import ibmiotf.device  
import time  
import random  
import sys  
# watson device details  
organization = "j5bxb7"  
devicType =  
"IOT123edevicetype" deviceId
```



```

= "IOTece4" authMethod=
"token" authToken= "e2)-
17xkqIFMvm3@II"
#generate random values for random variables
(temperature&humidity) def myCommandCallback(cmd):
    global a
    print("command recieved:%s" %cmd.data['command'])
control=cmd.data['command']
    print(control)
try:
    deviceOptions={"org": organization, "type": devicType,"id":
deviceId,"authmethod":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 "temp" with value integer value into the
cloud as a type of event for every 10 seconds deviceCli.connect() while True:
    distance=
random.randint(10,70)
loadcell= random.randint(5,15)
data=
{'dist':distance,'load':loadcell}

```

if loadcell < 13 and loadcell > 15:

load = "90 %"

elif loadcell < 8 and loadcell > 12:

load = "60 %"

elif loadcell < 4 and loadcell > 7:

load = "40 %"

else:

load = "0 %"

if distance < 15:

dist = 'Risk warning:' 'Dumpster poundage getting high, Time to collect
:) 90 %'

elif distance < 40 and distance > 16:

dist = 'Risk warning:' 'dumpster is above 60%'

elif distance < 60 and distance > 41:

dist = 'Risk warning:' '40 %'

else:

dist = 'Risk warning:' '17 %'

```

if load == "90 %" or distance == "90 %":
    warn = 'alert : ' ' Dumpster poundage getting high, Time to collect :)'

elif load == "60 %" or distance == "60 %":

    warn = 'alert : ' 'dumpster is above 60%'
else :

    warn = 'alert : ' 'No need to collect right now '
def myOnPublishCallback(lat=10.678991,long=78.177731):
    print("Gandigramam, Karur")
    print("published distance = %s " %distance,"loadcell:%s " %loadcell,"lon
= %s " %long,"lat = %s" %lat)
print(load)
print(dist)
print(warn)

time.sleep(10)

success=deviceCli.publishEvent
("IoTSensor","json",warn,qos=0,on_publish= myOnPublishCallback)

```

```
    success=deviceCli.publishEvent  
    ("IoTSensor","json",data,qos=0,on_publish= myOnPublishCallback)
```

```
    if not success:  
        print("not connected to ibmiot")  
    time.sleep(30)
```

```
    deviceCli.commandCallback=myCommandCallback  
#disconnect the device  
deviceCli.disconnect
```