

## DELIVERY OF SPRINT – 3

DATE	18- NOVEMBER 2022
TEAM ID	PNT2022TMID48657
PROJECT NAME	SMART WASTE MANAGEENT FOR METROPOLITAN CITIES

### **Code :**

```
import time
```

```
import sys
```

```
import ibmiotf.application
```

```
import ibmiotf.device
```

```
import random
```

```
#Provide your IBM Watson Device Credentials
```

```
organization = "9opizh"
```

```
deviceType = "NodeMCU"
```

```
deviceId = "123456"
```

```
authMethod = "token"
```

```
authToken = "8098439666"
```

```
# 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 DHT11
```

```
    weight=random.randint(0,100)
```

```
    level=random.randint(0,100)
```

```
    data = { 'weight' : weight, 'level':level }
```

```
    #print data
```

```
    def myOnPublishCallback():
```

```
        print ("Published Weight = %s Kg" % weight, "level = %s %" % level,
"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
```

# Creation of device in IBM Watson platform and running simulation

The screenshot displays the IBM Watson IoT Platform dashboard. The top navigation bar includes tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar labeled 'Search by Device ID' is present. The main content area shows a table of devices with columns: Device ID, Status, Device Type, Class ID, and Date Added. A device with ID '123456' is highlighted, showing a status of 'Connected' and a device type of 'NodeMCU'. Below the table, there are tabs for 'Identity', 'Device Information', 'Recent Events', 'State', and 'Logs'. The 'Recent Events' tab is active, displaying a table of events with columns: Event, Value, Format, and Last Received. The events are from an 'IoTSensor' and represent weight and level data. A '1 Simulation running' notification is visible at the bottom right of the events table.

Device ID	Status	Device Type	Class ID	Date Added
123456	Connected	NodeMCU	Device	Nov 16, 2022 11:42 AM

Event	Value	Format	Last Received
IoTSensor	{"weight":0,"level":72}	json	a few seconds ago
IoTSensor	{"weight":76,"level":94}	json	a few seconds ago
IoTSensor	{"weight":58,"level":64}	json	a few seconds ago
IoTSensor	{"weight":37,"level":9}	json	a few seconds ago

## Node Red platform

The screenshot shows the Node-RED web interface. The left sidebar contains a 'filter nodes' search bar and a list of nodes categorized by 'location'. The main workspace displays a flow named 'Flow 2'. The flow starts with an 'IBM IoT' node (connected). It branches into two parallel paths. The top path goes through a 'bin level' function node, then a 'distance' node (with a value of 123), and finally a 'level of bin' gauge node. The bottom path goes through a 'bin weight' function node, then a 'weight' node (with a value of 123), and finally a 'weight of waste' gauge node. A 'msg payload' node is also present in the flow. The right sidebar shows a 'debug' console with a list of messages received from the flow, including JSON objects with 'weight' and 'level' properties.

```
graph LR
    IoT[IBM IoT] --> bin_level[bin level]
    IoT --> bin_weight[bin weight]
    bin_level --> distance1[distance: 123]
    distance1 --> level_bin[level of bin]
    bin_weight --> weight1[weight: 123]
    weight1 --> weight_waste[weight of waste]
```