

SPRINT-3

Team ID	PNT2022TMID38943
Project Name	Smart Waste Management for Metropolitan Cities

1. Create a IBM cloud account, and login to IBM Watson.
2. Add new device in the ibm Watson .
3. Create a python code, the connect to IBM Watson.

The screenshot displays the IBM Watson IoT Platform interface. The top section shows a list of devices with columns for Device ID, Status, and Device Type. Two devices are listed: 02468 (Disconnected, Bin1) and 54321 (Connected, BIN1). The device 54321 is selected, and its details are shown in the bottom section, including Identity, Device Information, Recent Events, State, and Logs. The Recent Events section shows a stream of data points from an IoT sensor, such as {"distance":85}, {"distance":58}, {"distance":57}, {"distance":96}, {"distance":52}.

On the right side, a Python script titled "waste bin detect program.py" is shown. The script imports the random module and defines IBM Watson credentials. It initializes the IoT client and sets up a command callback function. The script then connects to the device and sends a datapoint "hello" with a value "world". A while loop follows, generating random distance values and printing alerts based on predefined thresholds (e.g., "waste bin level high is 90%, Time to collect").

```
import random

#Provide your IBM Watson Device Credentials
organization = "yal2ec"
deviceType = "BIN1"
deviceId = "54321"
authMethod = "token"
authToken = "12345678"

# Initialize GPIO
def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']

try:
    deviceOptions = {"org": organization, "type": deviceType, "id": deviceId}
    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 e
deviceCli.connect()

while True:
    #Get Sensor Data from ultrasonic
    distance= random.randint(5,110)
    data= {'distance':distance}
    if distance >5 and distance<=35:
        print("Alert!! 'waste bin level high is 90%, Time to collect")
    elif distance>35 and distance<=50:
        print("Risk warning:' 'waste Bin is above 60%")
    elif distance >35 and distance <=70:
        print("waste Bin level is above 40%")
    elif distance >70 and distance <=90:
        print("waste Bin level is above 25%")
```

4. Run the python code and see the recent event.
5. Then connect IBM Watson and NODE-RED.

The screenshot displays the IBM Watson IoT Platform dashboard with a modal window open for configuring a device event. The background dashboard shows a table of devices, with one device (ID: 54321, Status: Disconnected, Type: BIN1) selected. The modal window, titled 'Device Type: BIN1', is used to define an event named 'event_1'. It includes a 'Schedule' section set to 'Every Minute' and a 'Payload' section containing a JSON object with random values for 'randomNumber' and 'distance'. The payload is as follows:

```
0 {
1   "randomNumber": random(0, 100),
2   "distance": random(5, 100)
3 }
4
```

The dashboard also shows a 'Recent Events' section with two entries for 'event_1'.

Event	Value
event_1	{"randomNumber":99,"distance":67}
event_1	{"randomNumber":56,"distance":72}