

SPRINT - 2

Software (Create Device in The IOT Watson Platform, connect it with python code using IOT device Credentials and construct a node flow in node red)

Date	15 November 2022
Project Name	Smart Waste Management System for Metropolitan Cities
Project ID	PNT2022TMID03917

Step-1: Device creation in IOT Watson Platform

Device ID

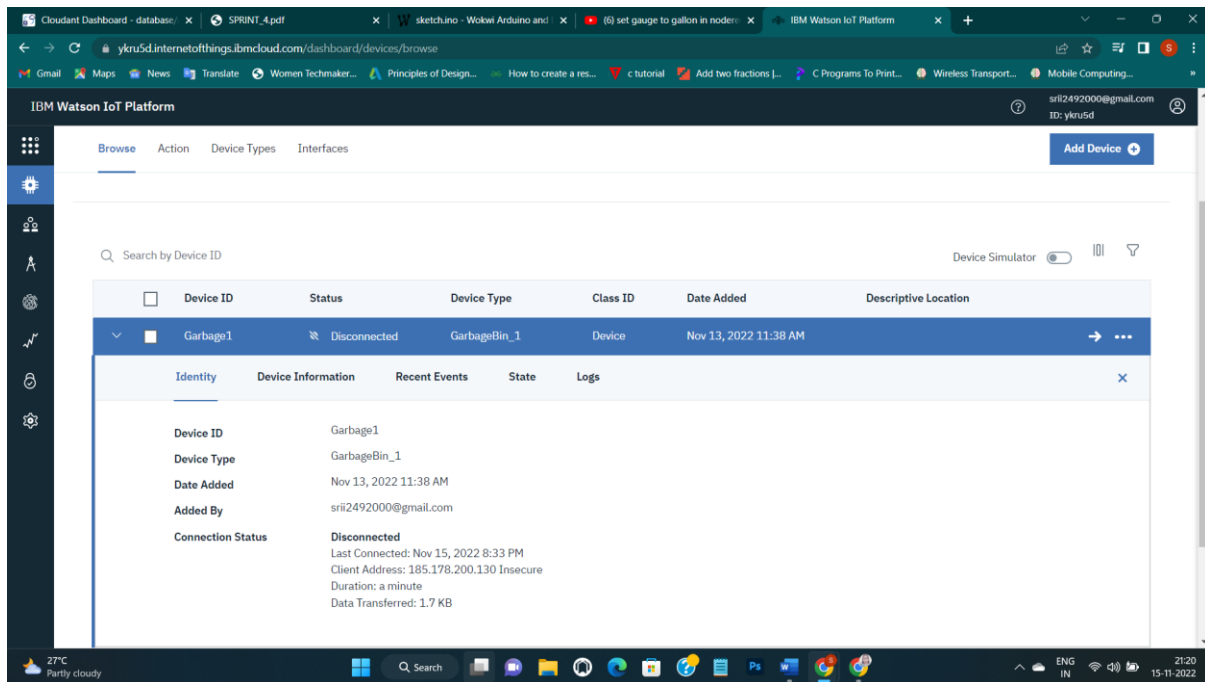
Garbage1

Device Type

GarbageBin_1

Added By

srii2492000@gmail.com



Step-2: Connect the python code written in wokwi with IOT device credentials

```
WOKWI SAVE SHARE sketch.ino
```

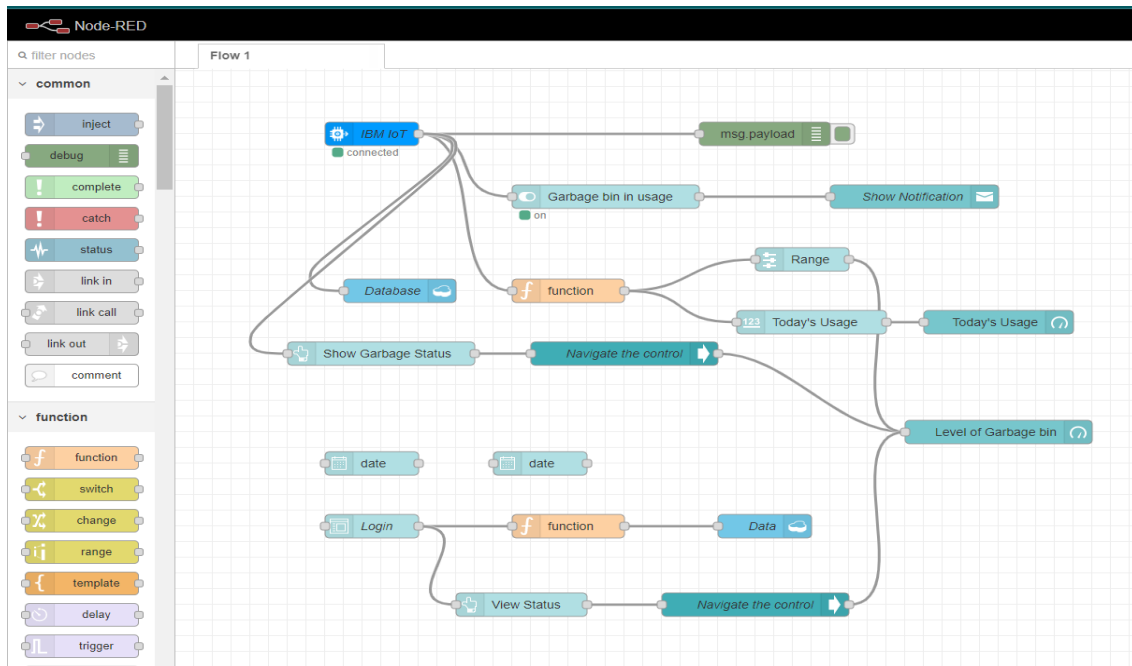
```
sketch.ino diagram.json libraries.txt Library Manager
```

```
1 #include <WiFi.h> // library for wifi
2 #include <PubSubClient.h> // library for MQ
3 #include <LiquidCrystal_I2C.h>
4 LiquidCrystal_I2C lcd(0x27, 20, 4);
5 //credentials of IBM Accounts -
6 #define ORG "ykru5d" // IBM organisation id
7 #define DEVICE_TYPE "GarbageBin_1" // Device type mentioned in ibm watson iot platform
8 #define DEVICE_ID "Garbage1" // Device ID mentioned in ibm watson iot platform
9 #define token "DKD_K)lt0VnlyQIeUf" // Token
10 #define authMethod "use-token-auth"
11 // customise above values -
12 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
13 // server name
14 char publishTopic[] = "iot-2/evt/data/fmt/json";
15 char topic[] = "iot-2/cmd/led/fmt/String"; // cmd Represent type and command is test format
16 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //Client id
17 //
18 WiFiClient wificlient; // creating instance for wificlient
19 PubSubClient client(server, 1883, wificlient);
20 #define ECHO_PIN 12
21 #define TRIG_PIN 13
22 float dist;
23 void setup()
24 {
25   Serial.begin(115200);
26   pinMode(LED_BUILTIN,OUTPUT);
27   pinMode(TRIG_PIN, OUTPUT);
28   pinMode(ECHO_PIN, INPUT);
29   //pir pin
30   pinMode(4, INPUT);
31   //ledpins
32   pinMode(23,OUTPUT);
33   pinMode(2,OUTPUT);
34   pinMode(4,OUTPUT);
35   pinMode(15, OUTPUT);
```

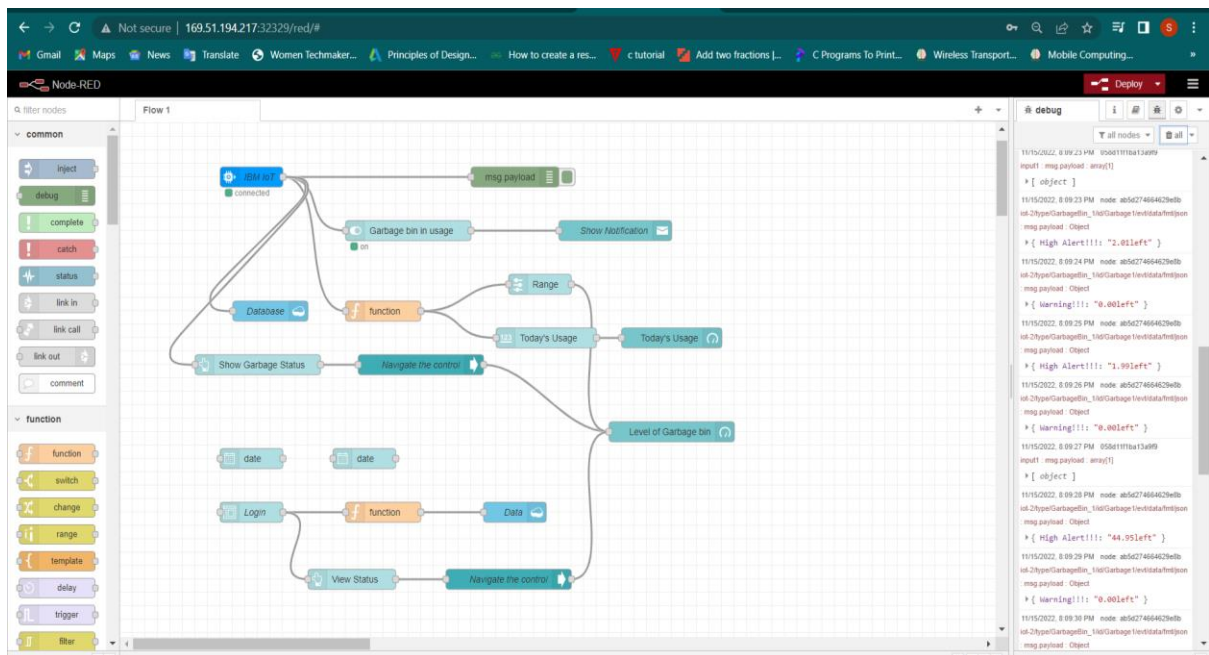
Step 3: Simulate to display the output in node red

The screenshot displays the Wokwi IDE interface. On the left, the code for the ESP32 is shown, including headers for WiFi, PubSubClient, and LiquidCrystal_I2C. It defines IBM credentials and sets up an MQTT client. The setup function initializes the serial port, pins, and the LCD display. The main loop (though not fully visible) would handle sensor readings and MQTT communication. On the right, the 'Simulation' window is active, showing a visual representation of the ESP32, an ultrasonic sensor, and an LCD. A status window titled 'Editing Ultrasonic Distance Sensor' indicates a distance of 2cm. Below this, a log shows 'No motion is detected', followed by 'Sending payload: {"High Alert!!!":"1.99left"}' and 'Publish OK'. Another log entry shows 'Sending Distance: 1.99' and 'Publish OK'. The bottom right corner of the simulation window shows a Node-RED icon, indicating the output is being sent to a Node-RED instance.

Step 4: Node-Red flow creation



Step 5: Using IBM IOT constructing a node flow according to the Python Script and simulating the wokwi python code to display the output in node red.



Step 6: Displaying the values in the Recent Events of the iot device created

The screenshot displays the IBM Watson IoT Platform dashboard. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains various icons for navigation. The main content area shows a table of devices. The first device, 'Garbage1', is selected, and its details are shown in a modal window. The 'Recent Events' tab is active, displaying a list of events.

Event	Value	Format	Last Received
data	{"Warning!!!!":"0.00left"}	json	a few seconds ago
data	{"High Alert!!!!":"98.94left"}	json	a few seconds ago
data	{"Warning!!!!":"0.00left"}	json	a few seconds ago
data	{"High Alert!!!!":"99.03left"}	json	a few seconds ago
data	{"Warning!!!!":"0.00left"}	json	a few seconds ago