

SPRINT-2

TEAM ID	PNT2022TMID50677
PROJECT NAME	Smart Waste Management System For Metropolitan Cities

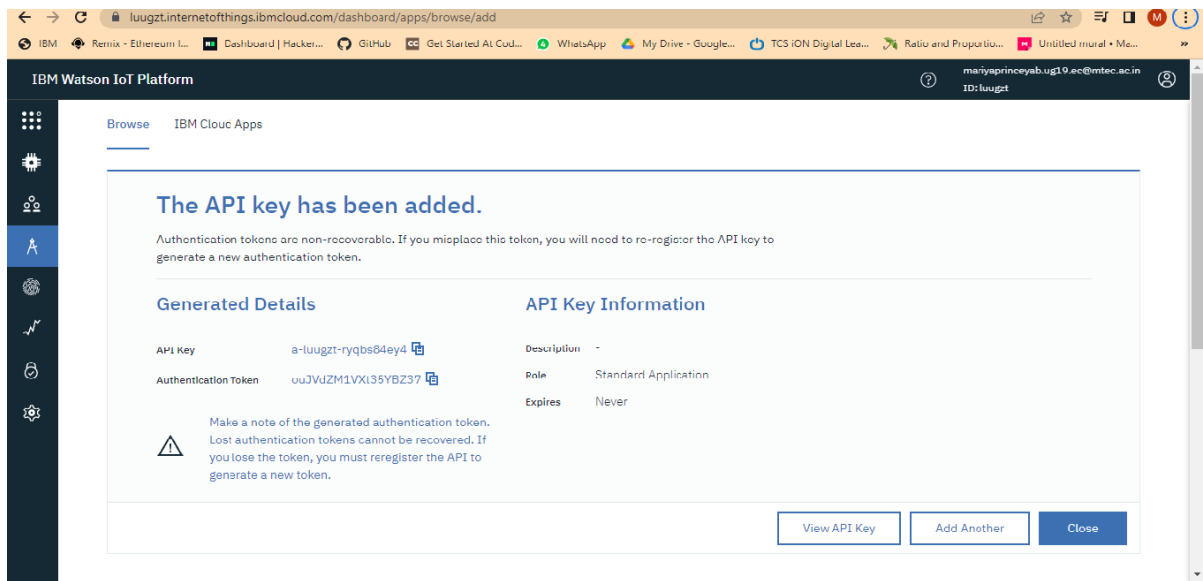
Create A Node-Red Service:

The screenshot shows the Node-RED web interface in a browser. The main workspace displays a flow with the following components: an 'IBM IoT' node (connected), a 'msg payload' node, and two 'function' nodes. The first 'function' node is connected to a 'Space Level' node, and the second 'function' node is connected to a 'Garbage status' node. The left sidebar shows the 'common' and 'function' node palettes. The right sidebar shows the 'debug' console with several error messages: 'TypeError: Cannot read properties of undefined (reading 'humidity')', 'TypeError: Cannot read properties of undefined (reading 'account')', and 'TypeError: Cannot read properties of undefined (reading 'temperature')'. The console also shows the 'msg payload' object: { dist: 40, garbage_status: 'Garbage full !' }.

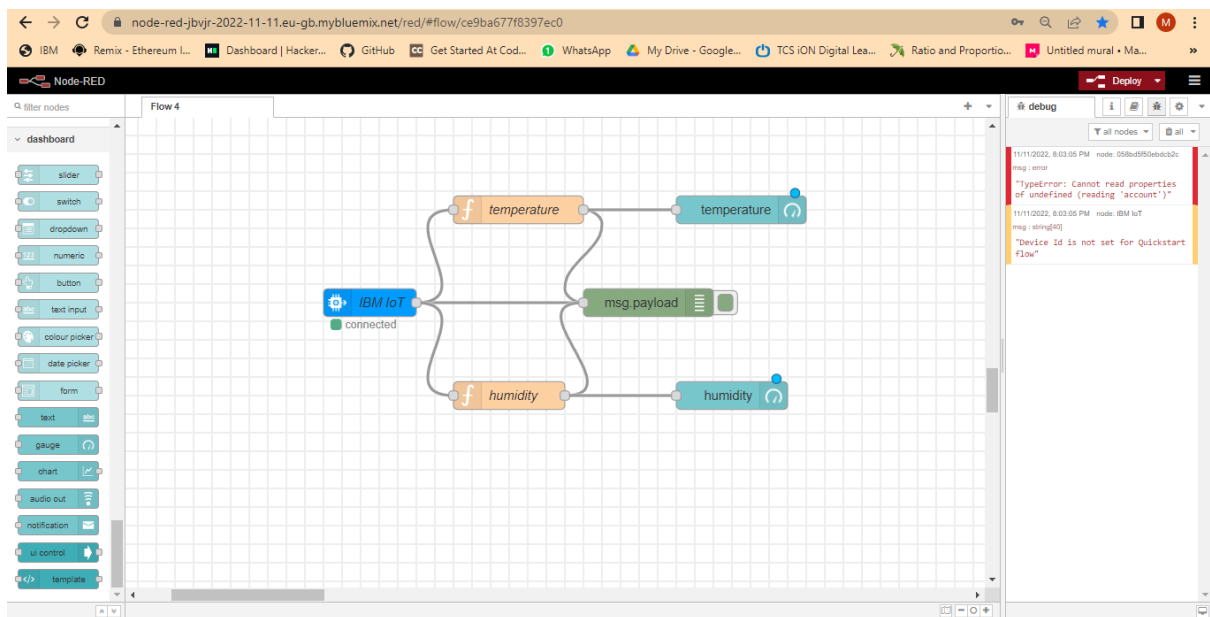
Configure The Connection Security & Create API Key:

The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes the IBM logo, a search bar, and user information (mariyaprinceyab.ug19.ec@mtec.ac.in, ID: luugzt). The main content area is titled 'Browse' and displays a table of devices. The table has columns for 'Device ID', 'Device Name', 'Device Type', 'Device Status', and 'Last Received'. The data rows show five devices, all with the same 'Device ID' (61) and 'Device Name' (Garbage is not full !). The 'Device Type' is 'json' and the 'Last Received' time is 'a few seconds ago' for the first four devices and 'a minute ago' for the fifth. The bottom of the dashboard shows a pagination bar with 'Items per page 50' and '1-2 of 2 items'.

Device ID	Device Name	Device Type	Device Status	Last Received
61	Garbage is not full !	json	Online	a few seconds ago
61	Garbage is not full !	json	Online	a few seconds ago
61	Garbage is not full !	json	Online	a few seconds ago
61	Garbage is not full !	json	Online	a few seconds ago
61	Garbage is not full !	json	Online	a minute ago



Create The Web Application:



Develop C Program:

```
#include <time.h>
```

```
#include <WiFi.h>
```

```
#include <PubSubClient.h>
```

```
#define ORG "luugzt"
```

```
#define DEVICE_TYPE "nodemcu"
```

```
#define DEVICE_ID "12345"
```

```
#define TOKEN "@StVt+)?pZUwEPT@Q"
```

```
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
```

```
char publishTopic[] = "iot-2/evt/data/fmt/json";
```

```
char authMethod[] = "use-token-auth";
```

```
char token[] = TOKEN;
```

```
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
```

```
WiFiClient wifiClient;
```

```
PubSubClient client(server, 1883, wifiClient);
```

```
const int trigP = 4; //D4 Or GPIO-2 of nodemcu
```

```
const int echoP = 2; //D3 Or GPIO-0 of nodemcu
```

```
long duration;
```

```
int distance;
```

```
String garbage_status="";
```

```
void setup() {
```

```
    pinMode(trigP, OUTPUT); // Sets the trigPin as an Output
```

```
    pinMode(echoP, INPUT); // Sets the echoPin as an Input
```

```

Serial.begin(99900);

wifiConnect();

mqttConnect();
}

void loop() {

digitalWrite(trigP, LOW); // Makes trigPin low
delayMicroseconds(2);    // 2 micro second delay

digitalWrite(trigP, HIGH); // trigPin high
delayMicroseconds(10);    // trigPin high for 10 micro seconds
digitalWrite(trigP, LOW); // trigPin low

duration = pulseIn(echoP, HIGH); //Read echo pin, time in microseconds
distance= duration*0.034/2;    //Calculating actual/real distance

Serial.print("Distance = ");    //Output distance on arduino serial monitor
Serial.println(distance);

delay(3000);

if(distance <= 60){

    garbage_status = "Garbage full !";
}
}

```

```
}  
  
else{  
    garbage_status = "Garbage is not full !";  
}
```

//json format for IBM Watson

```
String payload = "{";  
payload+="\"dist\":";  
payload+=(int)distance;  
payload+=",";  
payload+="\"garbage_status\":" + "\"" + garbage_status + "\"}";
```

```
if(client.publish(publishTopic, (char*) payload.c_str()))
```

```
{  
    Serial.println("Publish OK");  
}
```

```
else{  
    Serial.println("Publish failed");  
}
```

```
delay(100);
```

```
if (!client.loop())
```

```
{
```

```
    mqttConnect();  
}  
}
```

```
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);

}

Serial.println();

}

}

```

JIRA File (SPRINT-2):

