

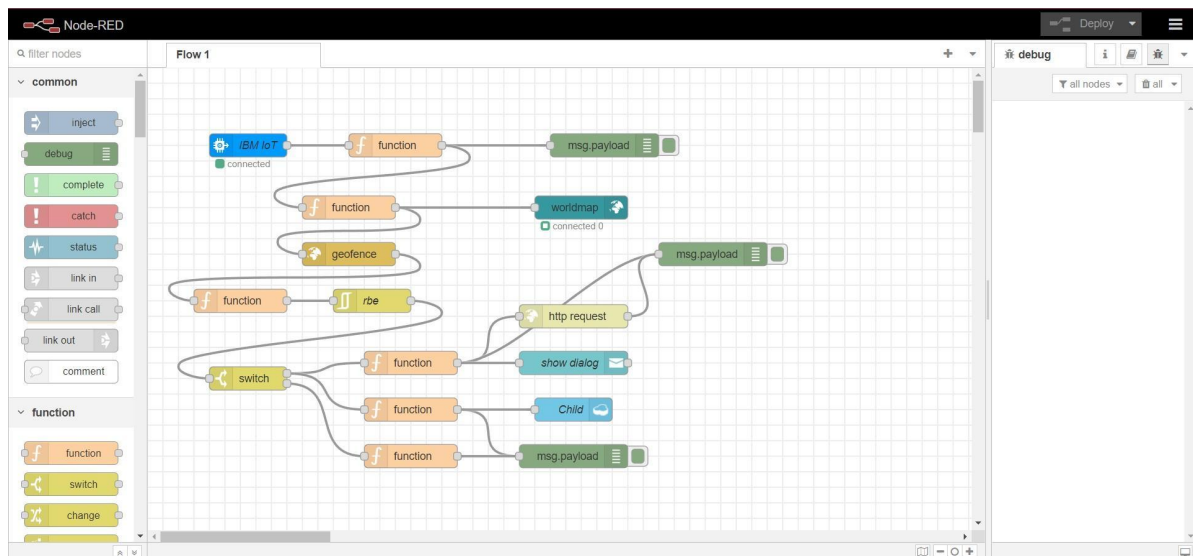
Project Development – Delivery plan sprint-3

IoT Based Safety Gadget for Child Safety Monitoring & Notification

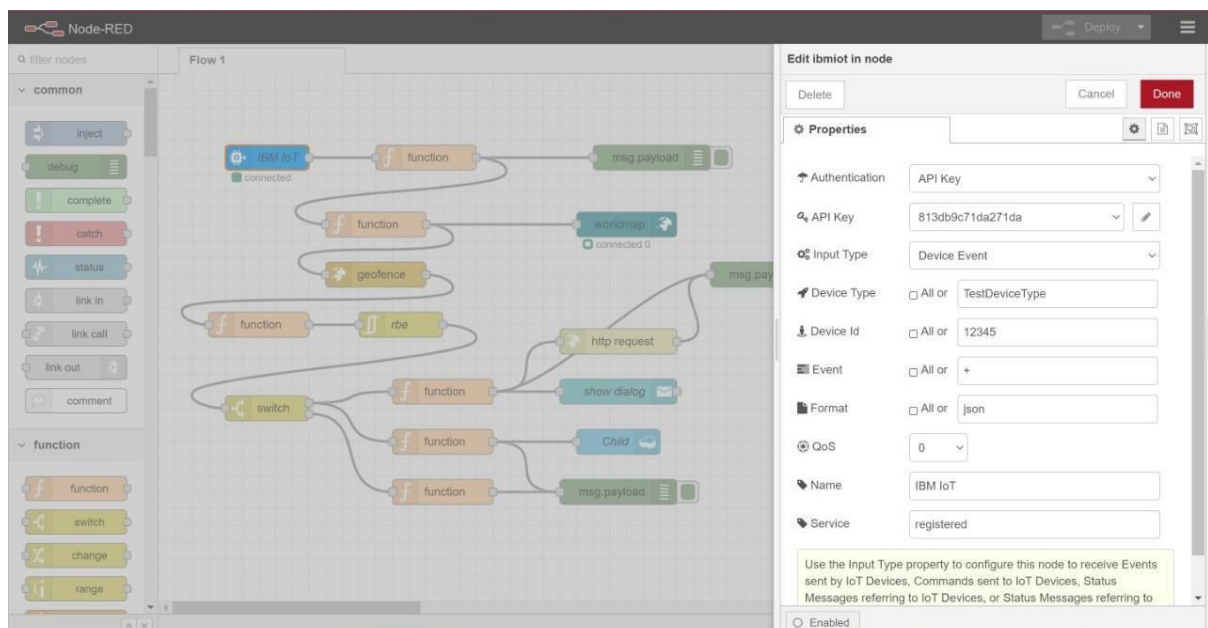
TEAM ID:PNT2022TMID38441

Creating Node-Red service and connecting with IBM cloud

Creating Node-Red service:



Codes in each Node:



Node-RED interface showing a flow named "Child Tracker" with a function node being edited. The function node code is:

```
1 var name = msg.payload.name
2 var lat = msg.payload.lat
3 var lon = msg.payload.lon
4 global.set('latitude', lat)
5 global.set('longitude', lon)
6 global.set('name', name)
7 return msg;
```

The dashboard on the right shows a "Child Tracker" tab with a "Map" link.

URL: <https://node-red-opzsk-2022-11-04.eu-gb.mybluemix.net/red/#editor-tab-properties>

WhatsApp Image...jpeg

Node-RED interface showing a flow named "Child Tracker" with a debug node being edited. The debug node settings are:

- Output: msg.payload
- To: ☒ debug window
- ☐ system console
- ☐ node status (32 characters)

The dashboard on the right shows a "Child Tracker" tab with a "Map" link.

Node-RED interface showing a flow named "Child Tracker" with a function node being edited. The function node code is:

```
1- msg.payload = {
2   "name": global.get('name'),
3   "lat": global.get('latitude'),
4   "lon": global.get('longitude')
5- }
6 return msg;
```

The dashboard on the right shows a "Child Tracker" tab with a "Map" link.

URL: <https://node-red-opzsk-2022-11-04.eu-gb.mybluemix.net/red/#editor-tab-properties>

Node-RED interface showing a flow diagram and the 'Edit worldmap node' configuration panel.

Flow Diagram: The flow starts with an 'IBM IoT' node connected to a 'function' node. This 'function' node connects to another 'function' node, which then connects to a 'worldmap' node. The 'worldmap' node connects to a 'msg.payload' node. Below this, there is a 'geofence' node connected to a 'function' node, which connects to an 'rbe' node. The 'rbe' node connects to an 'http request' node. The 'http request' node connects to a 'show dialog' node, which connects to a 'Child' node. The 'Child' node connects to a 'msg.payload' node. A 'switch' node is also connected to the 'function' node and the 'rbe' node.

Edit worldmap node Properties:

- Group: [Child Tracker] Map
- Size: auto
- Start: Latitude 17.4226372, Longitude 78.5456505, Zoom 16
- Map list: 7 selected
- Base map: ESRI Satellite
- Overlays: 5 selected
- Cluster when zoom level is less than 0 (0, off - 19)
- Max age: Remove markers after 600 seconds
- User menu: Show, Layer menu: Hide
- Lock map: False, Lock zoom: False
- Auto-pan: Disable, Right click: Disable
- Enabled: ☐

Node-RED interface showing a flow diagram and the 'Edit geofence node' configuration panel.

Flow Diagram: The flow starts with an 'IBM IoT' node connected to a 'function' node. This 'function' node connects to another 'function' node, which then connects to a 'worldmap' node. The 'worldmap' node connects to a 'msg.payload' node. Below this, there is a 'geofence' node connected to a 'function' node, which connects to an 'rbe' node. The 'rbe' node connects to an 'http request' node. The 'http request' node connects to a 'show dialog' node, which connects to a 'Child' node. The 'Child' node connects to a 'msg.payload' node. A 'switch' node is also connected to the 'function' node and the 'rbe' node.

Edit geofence node Properties:

- Floor: ground, Ceiling: infinity
- Action: add "inarea" property
- Enable output of zones to WorldMap node: ☐
- Enabled: ☐

Node-RED interface showing a flow diagram, the 'Edit function node' configuration panel, and the dashboard view.

Flow Diagram: The flow starts with an 'IBM IoT' node connected to a 'function' node. This 'function' node connects to another 'function' node, which then connects to a 'geofence' node. The 'geofence' node connects to a 'function' node. Below this, there is a 'switch' node connected to the 'function' node.

Edit function node Properties:

- Name: Name
- Setup: ☐ On Start: ☐ On Message: ☒ On Stop: ☐
- Code:

```
1 msg.payload=msg.location.inarea
2 return msg;
```
- Enabled: ☐

Dashboard View:

- Layout: Site, Theme
- Child Tracker: Map

<https://node-red-opzk-2022-11-04.eu-gb.mybluemix.net/red/#editor-tab-properties>

Node-RED interface showing the "Edit filter node" dialog. The flow "Child Tracker" is visible, containing nodes: IBM IoT, function, msg.payload, function, worldmap, geofence, function, rbe, and switch. The "Edit filter node" dialog is open, showing properties: Mode (block unless value changes), Property (msg.payload), Apply mode separately for each (checked), msg.topic, and Name (rbe). The "Done" button is highlighted.

<https://node-red-opszk-2022-11-04-eu-gb.mybluemix.net/red/#editor-tab-properties>

Node-RED interface showing the "Edit switch node" dialog. The flow "Child Tracker" is visible, containing nodes: IBM IoT, function, msg.payload, function, worldmap, geofence, function, rbe, and switch. The "Edit switch node" dialog is open, showing properties: Name (Name), Property (msg.payload), and two rules: "Is false" (→ 1) and "Is true" (→ 2). The "Done" button is highlighted.

<https://node-red-opszk-2022-11-04-eu-gb.mybluemix.net/red/#editor-tab-properties>

Node-RED interface showing the "Edit function node" dialog. The flow "Child Tracker" is visible, containing nodes: IBM IoT, function, msg.payload, function, worldmap, geofence, function, rbe, and switch. The "Edit function node" dialog is open, showing the "On Message" tab with the following JavaScript code:

```
1 var d = new Date();
2 var utc = d.getTime() + (d.getTimezoneOffset() * 60000);
3
4 var offset = 5.5; // This is the offset for UTC+3, in your case (UTC+1)
5
6 newDate = new Date(utc + (3600000 * offset));
7
8
9 - msg.payload = {
10   "message": "Exit",
11   "time": newDate.toLocaleString(),
12   "name": global.get('name'),
13   "lat": global.get('latitude'),
14   "lon": global.get('longitude')
15 - };
16
17 return msg;
```

The "Done" button is highlighted.

<https://node-red-opszk-2022-11-04-eu-gb.mybluemix.net/red/#editor-tab-properties>

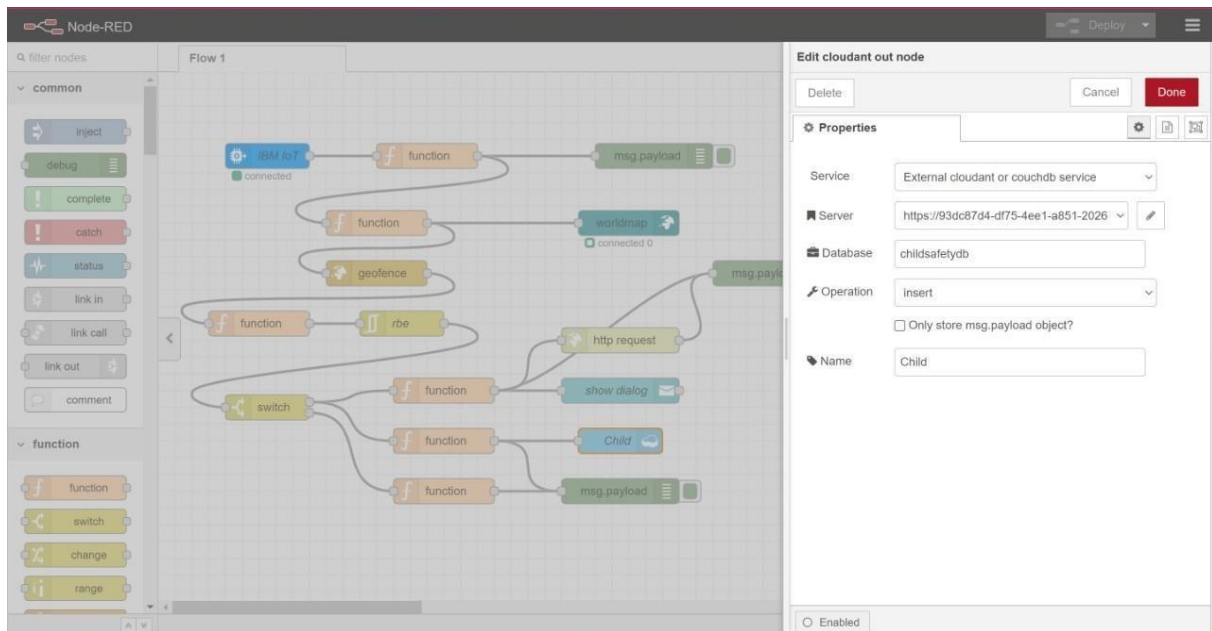
The screenshot displays the Node-RED web interface. On the left, the 'common' palette contains nodes like inject, debug, complete, catch, status, link in, link call, link out, and comment. The 'function' palette shows function and switch nodes. The main workspace shows a flow starting with an inject node, followed by a function node, then a msg.payload node, a worldmap node, a geofence node, another function node, an rbe node, and finally a switch node. The 'Edit function node' panel on the right shows a JavaScript function that calculates a UTC offset and returns a message object. The dashboard on the right shows a 'Child Tracker' tab with a 'Map' view.

```

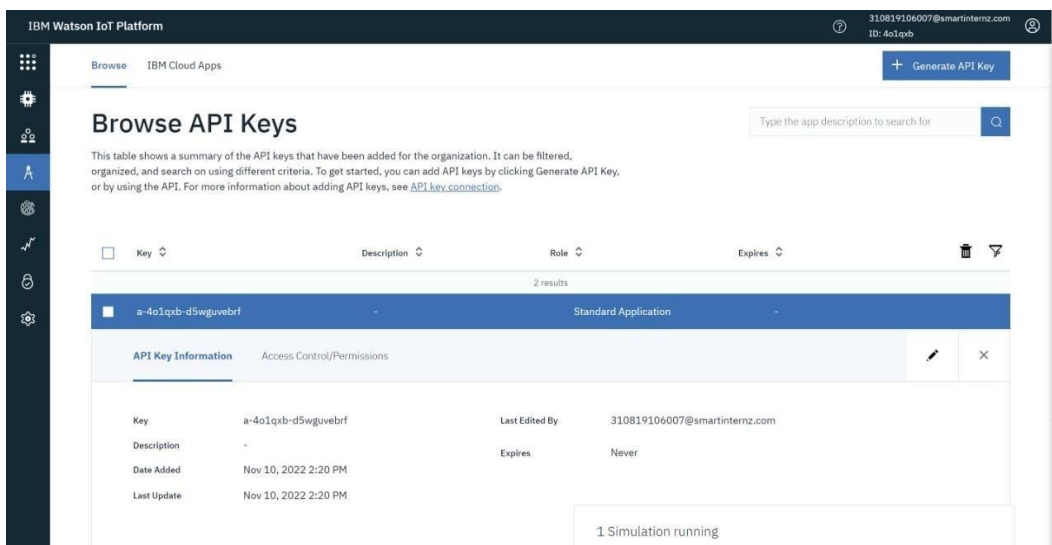
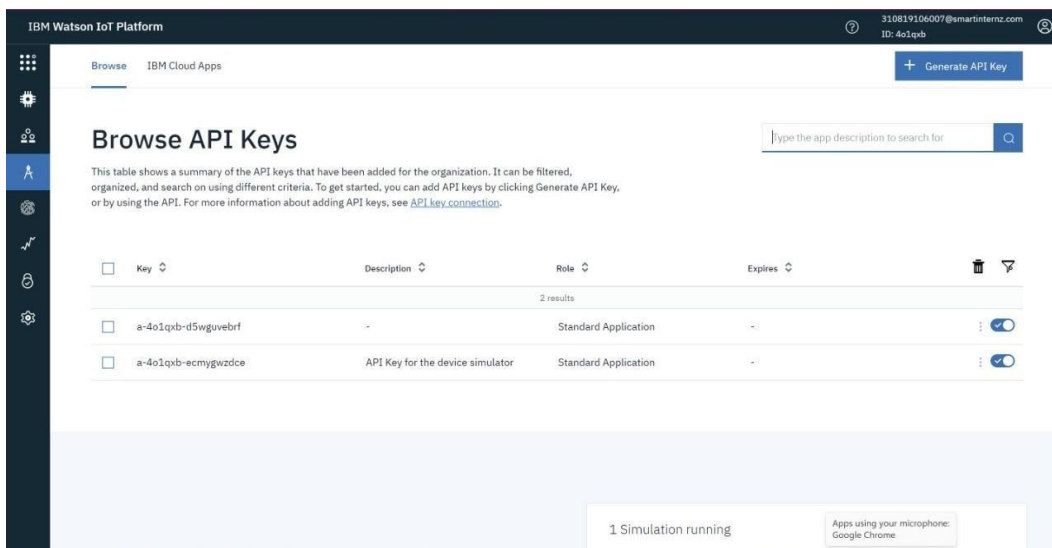
1 var d = new Date();
2
3 var utc = d.getTime() + (d.getTimezoneOffset() * 60000);
4
5 var offset = 5.5; // This is the offset for UTC+3, in your case (UTC+1)
6
7 newDate = new Date(utc + (3600000* offset));
8
9 msg.payload={
10   "message":"Entry",
11   "time": newDate.toLocaleString(),
12   "name":global.get('name'),
13   "lat":global.get('latitude'),
14   "lon":global.get('longitude')
15 };
16
17 return msg;

```

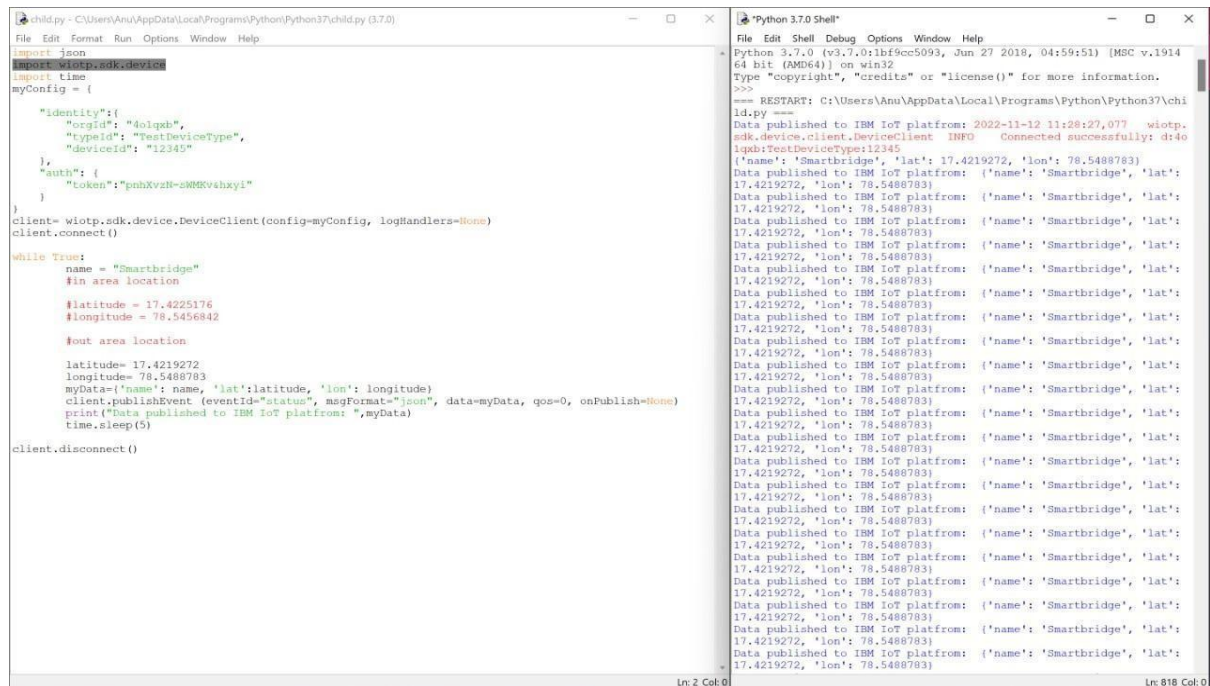
The screenshot displays the Node-RED web interface. On the left, the 'common' and 'function' node palettes are visible. The main workspace shows a flow named 'Flow 1' with several nodes: an 'inject' node, a 'debug' node, a 'complete' node, a 'catch' node, a 'status' node, a 'link in' node, a 'link call' node, a 'link out' node, and a 'comment' node. The flow starts with an 'inject' node connected to a 'function' node, which then connects to a 'msg.payload' node. This is followed by a 'worldmap' node, a 'geofence' node, and an 'rbe' node. The flow then splits into two paths: one through a 'switch' node to a 'function' node, and another through a 'function' node to a 'show dialog' node. Both paths converge at an 'http request' node, which then connects to a 'Child' node and finally to a 'msg.payload' node. On the right, the 'Edit http request node' panel is open, showing the 'Properties' tab. The 'Method' is set to 'GET', the 'URL' is 'https://www.fast2sms.com/dev/bulkV2?authorization...', the 'Payload' is set to 'Ignore', and the 'Return' is set to 'a UTF-8 string'.



Connecting with IBM Cloud: Using IBM IOT node through the API key



Transferring values from Python Code:



The image shows two windows. The left window is a Python script named 'chld.py' that uses the 'wiotp.sdk.device' library to connect to an IBM IoT platform. It defines a configuration object 'myConfig' with identity, auth, and location details. A 'while True' loop publishes location data (latitude, longitude) as JSON events to the IoT platform. The right window is a 'Python 3.7.0 Shell' showing the execution of the script. It displays a RESTAR message and a series of 'Data published to IBM IoT platform' logs, each containing a JSON object with 'name', 'lat', and 'lon' values.

```
import json
import wiotp.sdk.device
import time

myConfig = {
    "identity": {
        "orgid": "4s1qpb",
        "typeid": "TestDeviceType",
        "deviceId": "12345"
    },
    "auth": {
        "token": "pnhXvzN-cMMKvshxyi"
    }
}

client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

while True:
    name = "Smartbridge"
    #in area location
    #latitude = 17.4225176
    #longitude = 78.5456842
    #out area location

    latitude= 17.4219272
    longitude= 78.5488783
    myData={'name': name, 'lat':latitude, 'lon': longitude}
    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
    print("Data published to IBM IoT platform: ",myData)
    time.sleep(5)

client.disconnect()
```

Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
== RESTAR: C:\Users\Anu\AppData\Local\Programs\Python\Python37\chld.py ==
Data published to IBM IoT platform: 2022-11-12 11:28:27,077 wiotp.sdk.device.client.DeviceClient INFO Connected successfully: dr4o lqab:TestDeviceType:12345
{'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}
Data published to IBM IoT platform: {'name': 'Smartbridge', 'lat': 17.4219272, 'lon': 78.5488783}

Node-Red:

