

Sprint 3

Project	IoT Based Safety Gadget for Child Safety Monitoring and Notification
Team Id	PNT2022TMID19513
Member1	Harish S
Member2	Disha D
Member3	Durga Devi S
Member4	Yuvan Sanjay K

Python program:

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

myConfig={
    "identity":{
        "orgId":"hi70w8",
        "typeId":"gps",
        "deviceId":"987654321"
    },
    "auth":{
        "token":"24688462"
    }
}
client=wiotp.sdk.device.DeviceClient(config=myConfig,logHandlers=None)
client.connect()

while True:
    name="GPS"

    #outside
    #latitude=10.820155
    #longitude=77.016172

    #inside
    latitude=10.826579
    longitude=77.059943
```

```
myData={'name':name,'lat':latitude,'lon':longitude}
```

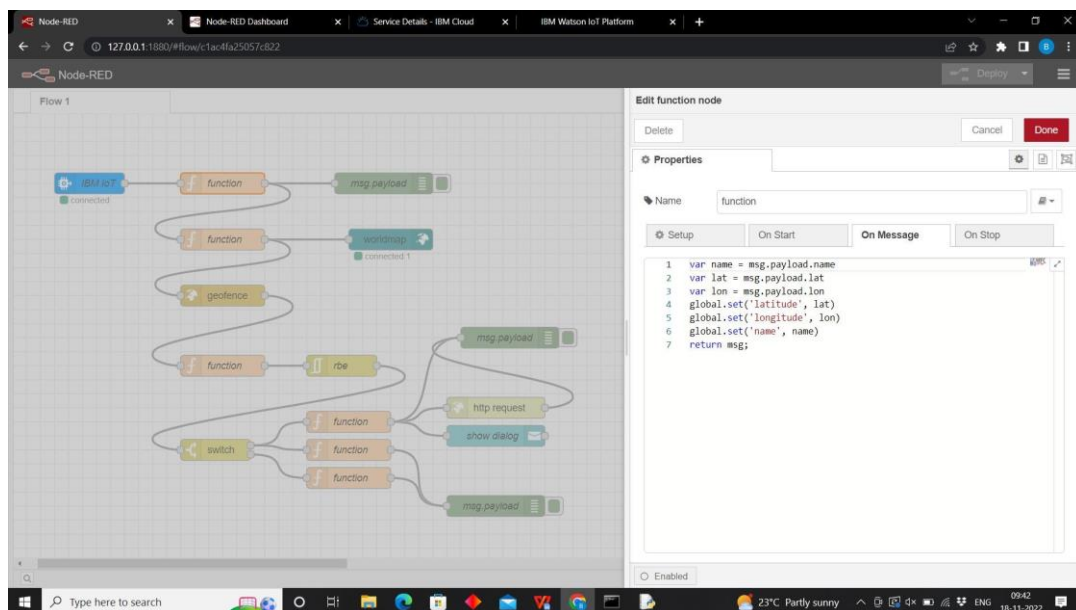
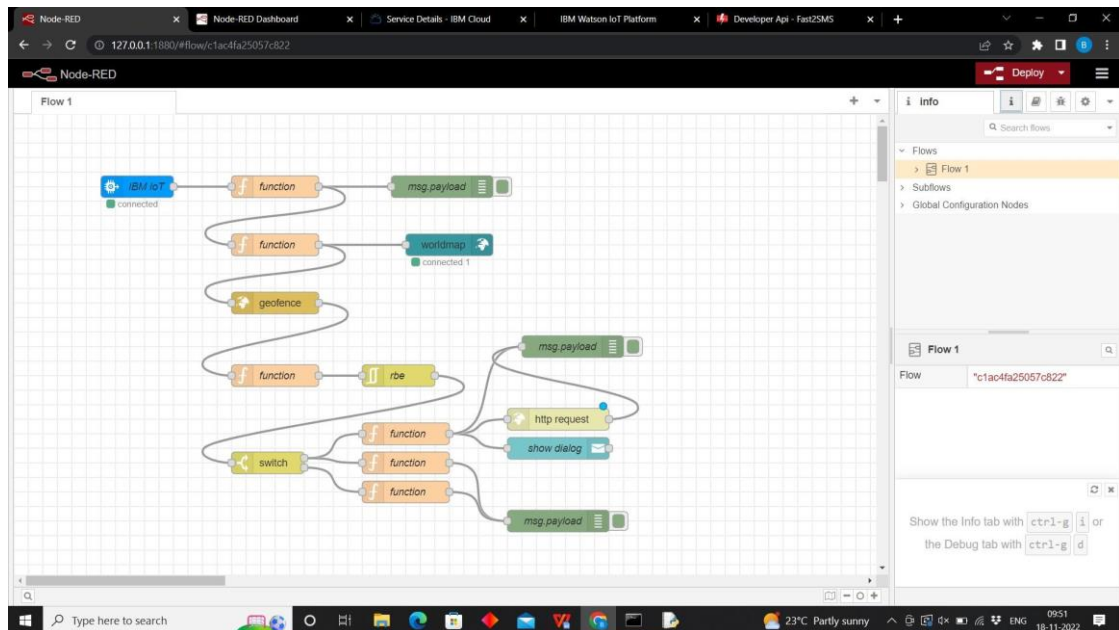
```
client.publishEvent(eventId="status",msgFormat="json",data=myData,qs=0,onPublish=None)
```

```
print("Data published to IBM platform:",myData)
```

```
time.sleep(5)
```

```
client.disconnect()
```

Node Red:



Node-RED interface showing a flow diagram and the configuration for the 'ibmiot in node' node.

Flow Diagram: The flow starts with an 'IBM IoT' node (connected), followed by a 'function' node, then a 'msg.payload' node. This is followed by another 'function' node, a 'workgroup' node (connected 1), and a 'geofence' node. The flow then splits into two paths: one through a 'function' node to an 'rbe' node, and another through a 'switch' node to three 'function' nodes. The 'rbe' node outputs to a 'msg.payload' node. The three 'function' nodes output to 'http request', 'show dialog', and 'msg.payload' nodes respectively.

Edit ibmiot in node Properties:

- Authentication: API Key
- API Key: a768a78415ab033d
- Input Type: Device Event
- Device Type: ☐ All or ☐ gps
- Device Id: ☐ All or 987654321
- Event: ☒ All or ☐ +
- Format: ☐ All or ☐ json
- QoS: 0
- Name: IBM IoT
- Service: registered

Use the Input Type property to configure this node to receive Events sent by IoT Devices, Commands sent to IoT Devices, Status Messages referring to IoT Devices, or Status Messages referring to

Enabled

Node-RED interface showing the same flow diagram and the configuration for the 'function' node.

Flow Diagram: The flow starts with an 'IBM IoT' node (connected), followed by a 'function' node, then a 'msg.payload' node. This is followed by another 'function' node, a 'workgroup' node (connected 1), and a 'geofence' node. The flow then splits into two paths: one through a 'function' node to an 'rbe' node, and another through a 'switch' node to three 'function' nodes. The 'rbe' node outputs to a 'msg.payload' node. The three 'function' nodes output to 'http request', 'show dialog', and 'msg.payload' nodes respectively.

Edit function node Properties:

- Name: function

Setup:

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

Enabled

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

Flow Diagram:

- IBM IoT (connected) → function → msg.payload →
- function → worldmap (connected 1) →
- geofence →
- function → rbe → msg.payload →
- switch → function → http request →
- function → show dialog →
- function → msg.payload →

Edit geofence node:

- Properties: Map showing a geofence area around "St. Eshwar College of Engineering".
- Floor: ground, Ceiling: infinity
- Action: add "inarea" property
- Enable output of zones to WorldMap node: ☐
- Enabled: ☐

Node-RED interface showing the same flow diagram and the "Edit function node" configuration panel.

Flow Diagram:

- IBM IoT (connected) → function → msg.payload →
- function → worldmap (connected 1) →
- geofence →
- function → rbe → msg.payload →
- switch → function → http request →
- function → show dialog →
- function → msg.payload →

Edit function node:

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

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

Node-RED interface showing a flow diagram and the Edit function node configuration.

Flow Diagram:

- IBM IoT (connected) → function → msg.payload
- function → workflow (connected 1)
- geofence → function → rbe
- function → rbe → msg.payload
- switch → function → http request
- switch → function → show dialog
- switch → function → msg.payload

Edit function node configuration:

- Name: function
- Setup: On Message
- Code:

```
1 var d=new Date();
2 var utc=d.getTime()+d.getTimezoneOffset()*60000;
3 var offset=5.5;
4 const newDate=new Date(utc+(3600000*offset));
5 msg.payload={
6   "message":"Entry",
7   "time":newDate.toLocaleString(),
8   "name":global.get('name'),
9   "lat": global.get('latitude'),
10  "lon": global.get('longitude')
11 };
12 return msg;
```

Node-RED interface showing a flow diagram and the Edit function node configuration.

Flow Diagram:

- IBM IoT (connected) → function → msg.payload
- function → workflow (connected 1)
- geofence → function → rbe
- function → rbe → msg.payload
- switch → function → http request
- switch → function → show dialog
- switch → function → msg.payload

Edit function node configuration:

- Name: function
- Setup: On Message
- Code:

```
1
2 var d=new Date();
3 var utc=d.getTime()+d.getTimezoneOffset()*60000;
4 var offset=5.5;
5 const newDate=new Date(utc+(3600000*offset));
6 msg.payload={
7   "message":"Exit",
8   "time":newDate.toLocaleString(),
9   "name":global.get('name'),
10  "lat": global.get('latitude'),
11  "lon": global.get('longitude')
12 };
13 return msg;
```

Node-RED interface showing a flow diagram and the configuration for an HTTP request node.

Flow Diagram:

- IBM IoT node (connected) feeds into a function node.
- The function node feeds into a msg payload node.
- The msg payload node feeds into another function node.
- This function node feeds into a worldmap node (connected 1).
- The worldmap node feeds into a geofence node.
- The geofence node feeds into a function node.
- This function node feeds into an rbe node.
- The rbe node feeds into a function node.
- This function node feeds into a switch node.
- The switch node feeds into three function nodes.
- These three function nodes feed into an http request node, a show dialog node, and a msg payload node.

Edit http request node configuration:

- Method: GET
- URL: <https://www.fast2sms.com/dev/bulkV2?authorizat>
- Payload: Ignore
- Enable secure (SSL/TLS) connection: ☐
- Use authentication: ☐
- Enable connection keep-alive: ☐
- Use proxy: ☐
- Only send non-2xx responses to Catch node: ☐
- Disable strict HTTP parsing: ☐
- Return: a UTF-8 string
- Headers: (empty)
- Enabled: ☐

Node-RED interface showing a flow diagram and the configuration for a worldmap node.

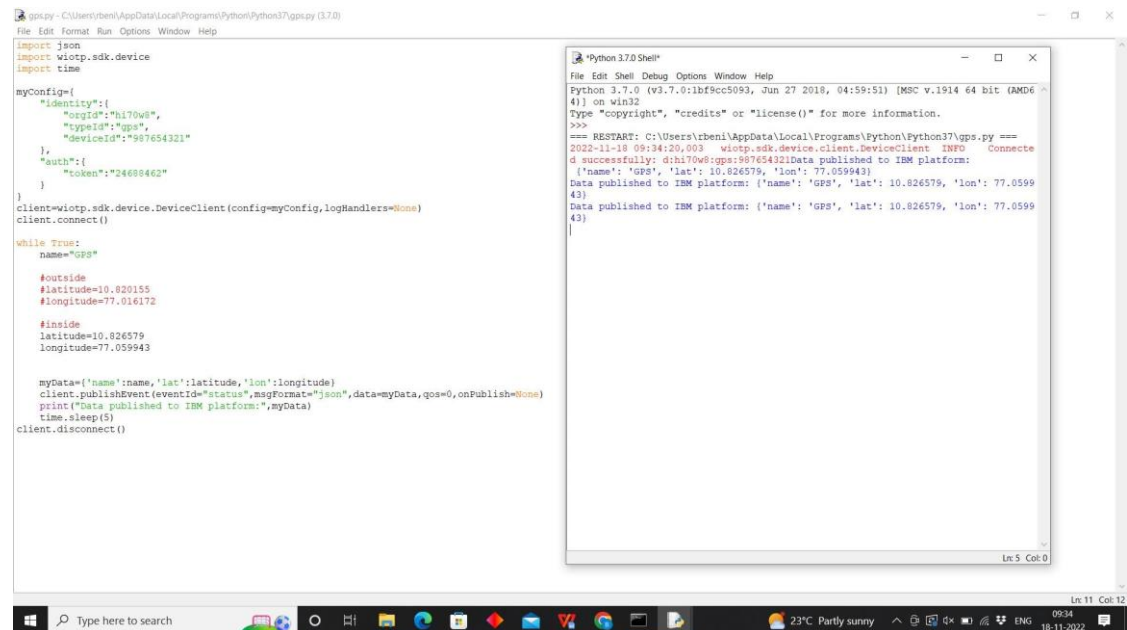
Flow Diagram:

- IBM IoT node (connected) feeds into a function node.
- The function node feeds into a msg payload node.
- The msg payload node feeds into another function node.
- This function node feeds into a worldmap node (connected 1).
- The worldmap node feeds into a geofence node.
- The geofence node feeds into a function node.
- This function node feeds into an rbe node.
- The rbe node feeds into a function node.
- This function node feeds into a switch node.
- The switch node feeds into three function nodes.
- These three function nodes feed into an http request node, a show dialog node, and a msg payload node.

Edit worldmap node configuration:

- Group: [Home] Map
- Size: 10 x 12
- Start: Latitude 10.820155, Longitude 77.016172, Zoom 1 - 18
- Map list: 8 selected
- Base map: OpenStreetMap
- Overlays: 6 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: ☐

Output:



```
gps.py - C:\Users\beni\AppData\Local\Programs\Python\Python37\gps.py (3.7.0)
File Edit Format Run Options Window Help

import json
import wiotp.sdk.device
import time

myConfig={
    "identity":{
        "orgId":"h170w8",
        "typeId":"gps",
        "deviceId":"987654321"
    },
    "auth":{
        "token":"34608462"
    }
}

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

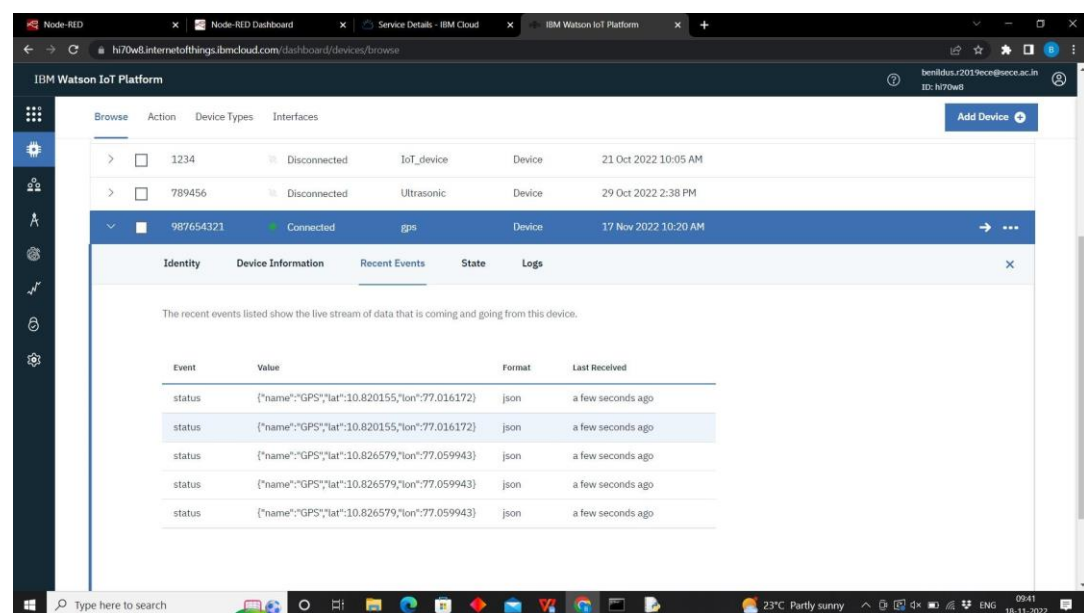
while True:
    name="GPS"

    #outside
    #latitude=10.820155
    #longitude=77.016172

    #inside
    latitude=10.826579
    longitude=77.059943

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

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
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.
>>>
==== RESTART: C:\Users\beni\AppData\Local\Programs\Python\Python37\gps.py ===
2022-11-10 09:34:20,003 wiotp.sdk.device.client.DeviceClient INFO Connecte
d successfully: d:h170w8:987654321Data published to IBM platform:
{'name': 'GPS', 'lat': 10.826579, 'lon': 77.059943}
Data published to IBM platform: {'name': 'GPS', 'lat': 10.826579, 'lon': 77.0599
43}
Data published to IBM platform: {'name': 'GPS', 'lat': 10.826579, 'lon': 77.0599
43}
>>>
```



IBM Watson IoT Platform

beni@beni2019ece@ece.ac.in
ID: h170w8

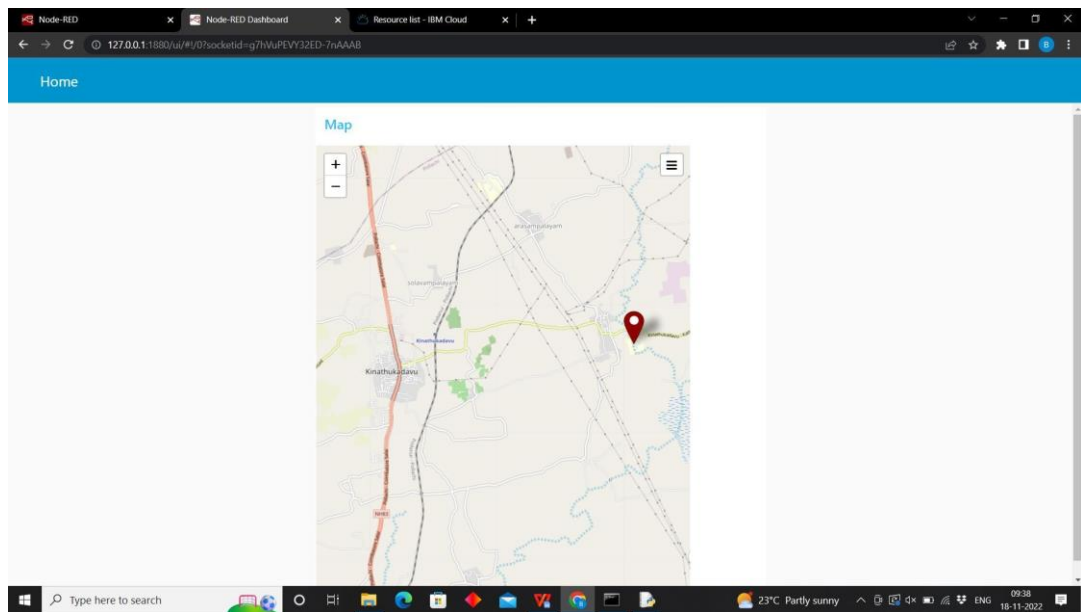
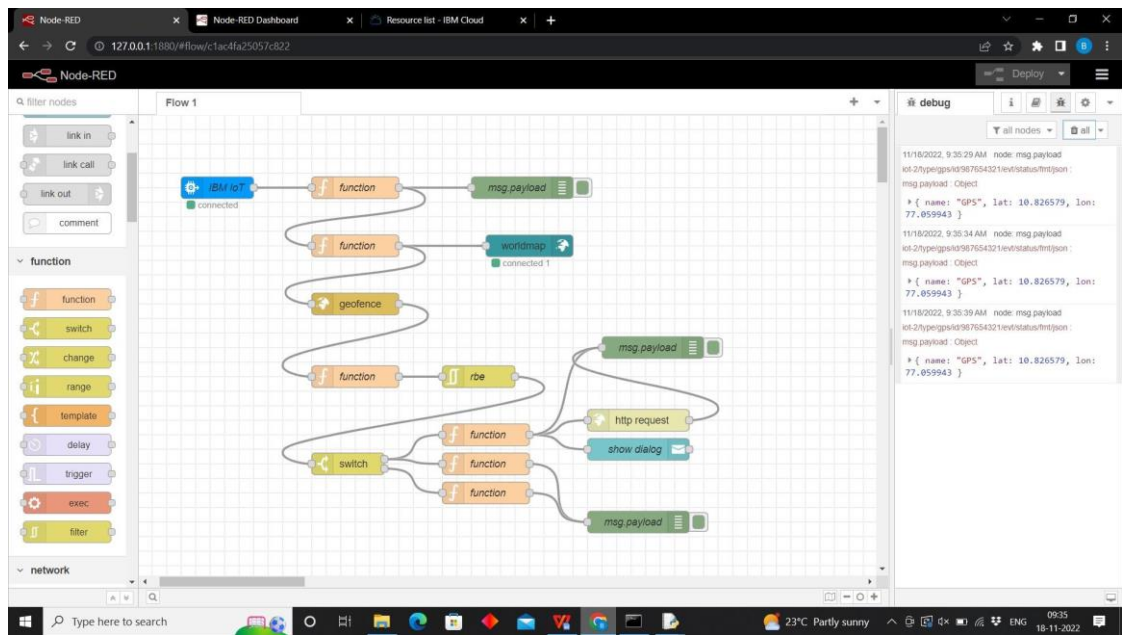
Browse Action Device Types Interfaces Add Device

ID	Status	Type	Device	Last Seen
1234	Disconnected	IoT_device	Device	21 Oct 2022 10:05 AM
789456	Disconnected	Ultrasonic	Device	29 Oct 2022 2:38 PM
987654321	Connected	gps	Device	17 Nov 2022 10:20 AM

Identity Device Information Recent Events State Logs

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
status	{ "name": "GPS", "lat": 10.820155, "lon": 77.016172 }	json	a few seconds ago
status	{ "name": "GPS", "lat": 10.820155, "lon": 77.016172 }	json	a few seconds ago
status	{ "name": "GPS", "lat": 10.826579, "lon": 77.059943 }	json	a few seconds ago
status	{ "name": "GPS", "lat": 10.826579, "lon": 77.059943 }	json	a few seconds ago
status	{ "name": "GPS", "lat": 10.826579, "lon": 77.059943 }	json	a few seconds ago




```
gps.py - C:\Users\rbeni\AppData\Local\Programs\Python\Python37\gps.py (3.7.0)
File Edit Format Run Options Window Help

import json
import wiotp.sdk.device
import time

myConfig={
    "identity":{
        "orgId":"hl70w8",
        "typeId":"gps",
        "deviceId":"987654321"
    },
    "auth":{
        "token":"24680462"
    }
}

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

while True:
    name="GPS"

    #outside
    latitude=10.820155
    longitude=77.016172

    #inside
    latitude=10.826579
    longitude=77.059943

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

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help

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.
>>>
=== RESTART: C:\Users\rbeni\AppData\Local\Programs\Python\Python37\gps.py ===
2022-11-18 09:35:28,608 wiotp.sdk.device.client.DeviceClient INFO Connecte
d successfully: dhl70w8:987654321Data published to IBM platform:
{'name': 'GPS', 'lat': 10.826579, 'lon': 77.059943}
Data published to IBM platform: {'name': 'GPS', 'lat': 10.826579, 'lon': 77.0599
43}
Data published to IBM platform: {'name': 'GPS', 'lat': 10.826579, 'lon': 77.0599
43}
Data published to IBM platform: {'name': 'GPS', 'lat': 10.826579, 'lon': 77.0599
43}
Data published to IBM platform: {'name': 'GPS', 'lat': 10.826579, 'lon': 77.0599
43}
Data published to IBM platform: {'name': 'GPS', 'lat': 10.826579, 'lon': 77.0599
43}
Data published to IBM platform: {'name': 'GPS', 'lat': 10.826579, 'lon': 77.0599
43}
=== RESTART: C:\Users\rbeni\AppData\Local\Programs\Python\Python37\gps.py ===
2022-11-18 09:36:12,641 wiotp.sdk.device.client.DeviceClient INFO Connecte
d successfully: dhl70w8:987654321Data published to IBM platform:
{'name': 'GPS', 'lat': 10.820155, 'lon': 77.016172}
Data published to IBM platform: {'name': 'GPS', 'lat': 10.820155, 'lon': 77.0161
72}
Data published to IBM platform: {'name': 'GPS', 'lat': 10.820155, 'lon': 77.0161
72}
Data published to IBM platform: {'name': 'GPS', 'lat': 10.820155, 'lon': 77.0161
72}
Data published to IBM platform: {'name': 'GPS', 'lat': 10.820155, 'lon': 77.0161
72}
```

