

Project Development Phase

Sprint 3

Date	7 November 2022
Team ID	PNT2022TMID41539
Project Name	Project-Signs with Smart Connectivity for Better Road Safety
Maximum marks	8 Marks

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3	Develop The Python Script	USN-6	Develop A Python Script Create a code snippet using python to <ul style="list-style-type: none">Extract weather data from OpenWeatherMap using APIsSend the extracted data to the cloudReceive data from the cloud and view it in the python compiler	10	Medium	1.Mugila R 2.Ishwariya P 3.Kalpana T 4.Shreein Fathima S

Extract weather data from OpenWeatherMap using APIs

The screenshot shows a web browser window with multiple tabs open, including IBM, Weather API, Service Det, IBM Watson, Node-RED, and MIT App Inventor. The active tab is the OpenWeatherMap API page. The browser's address bar shows the URL `openweathermap.org/api`. The page header includes the OpenWeather logo, a search bar, and navigation links like Guide, API, Dashboard, Marketplace, Pricing, Maps, Our Initiatives, Partners, Blog, For Business, Sign in, and Support. The main heading is "Weather API" with a breadcrumb trail "Home / Weather API". The content area explains the One Call API 3.0, highlighting its features and subscription options. A sidebar on the right lists various API products. A cookie consent banner is visible at the bottom of the page.

IBM Weather API Service Det IBM Watson Node-RED Node-RED MIT App In MIT App In Inbox (24)

openweathermap.org/api

OpenWeather Weather in your city Guide API Dashboard Marketplace Pricing Maps Our Initiatives Partners Blog For Business Sign in Support

Weather API

Home / Weather API

Please, [sign up](#) to use our fast and easy-to-work weather APIs. As a start to use OpenWeather products, we recommend our [One Call API 3.0](#). For more functionality, please consider our products, which are included in [professional collections](#).

One Call API 3.0 **NEW**

[API doc](#) [Subscribe](#)

Make one API call and receive all essential weather data in one response:

- Minute forecast for 1 hour
- Hourly forecast for 48 hours
- Daily forecast for 8 days
- Historical data for 40+ years back by timestamp
- National weather alerts

Read more about this API and subscription plan in the [FAQ](#).

This is a separate subscription plan, which include only One Call API.

Pay as you call

1,000 API calls per day for free
0.0012 GBP per API call over the daily limit

[Subscribe to One Call by Call](#)

We use cookies which are essential for the site to work. We also use non-essential cookies to help us improve our services. Any data collected is anonymised. You can allow all cookies or manage them individually.

[Allow all](#) [Manage cookies](#)

Waiting for tpc.google syndication.com...

00:01 19-11-2022

Developed python code

```
import wiotp.sdk.device
```

```
import time
```

```
import random
```

```
import ibmiotf.application
```

```
import ibmiotf.device
```

```
import requests, json
```

```
myConfig = {
```

```
    #Configuration
```

```
    "identity": {
```

```
        "orgId": "3dpjnk",
```

```
        "typeId": "Sign_Board",
```

```
        "deviceId": "Board_1"
```

```
    },
```

```
    #API Key
```

```
    "auth": {
```

```
        "token": "1234567890"
```

```
    }
```

```
}
```

```
#Receiving callbacks from IBM IOT platform
```

```
def myCommandCallback(cmd):
```

```
    print("Message received from IBM IoT Platform: %s" % cmd.data['command'])
```

```
m=cmd.data['command']
```

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

```
client.connect()
```

```
#OpenWeatherMap Credentials
```

```
BASE_URL ="https://api.openweathermap.org/data/2.5/weather?"
```

```
CITY = "Nagercoil"
```

```
URL = BASE_URL + "q=" + "Chennai" + "&appid=" + "01df65417ab3968e3fc2a38c4aee27bb"
```

```
while True:
```

```
    response = requests.get(URL)
```

```
    if response.status_code ==200:
```

```
        data = response.json()
```

```
        main = data['main']
```

```
        temperature =main['temp']
```

```
        humidity = main['humidity']
```

```
        pressure = main['pressure']
```

```
        report = data['visibility']
```

```
#messge part
```

```
msg=random.randint(0,5)
```

```
if msg==1:
```

```
    message="SLOW DOWN, SCHOOL IS NEAR"
```

```
elif msg==3:  
    message="SLOW DOWN, HOSPITAL NEARBY"  
elif msg==5:  
    message="NEED HELP, POLICE STATION NEARBY"  
else:  
    message=""
```

#Speed part

```
speed=random.randint(0,150)  
if speed>=100:  
    speedMsg=" SLOW DOWN, speed Limit Exceeded"  
elif speed>=60 and speed<100:  
    speedMsg="Moderate"  
else:  
    speedMsg="Slow"
```

#Sign part

```
sign=random.randint(0,5)  
if sign==1:  
    signMsg="Right Diversion"  
elif sign==3:  
    signMsg="Left Diversion"  
elif sign==5:  
    signmsg="U Turn"
```

else:

signMsg=""

#Visibility

if temperature < 50:

visibility="Fog Ahead, Drive Slow"

else:

visibility="Clear Weather"

else:

print("Error in the HTTP request")

myData={'Temperature':temperature, 'Message':message, 'Sign':signMsg, 'Speed':speedMsg, 'Visibility':visibility}

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

#PUBLISHING TO IOT WATSON

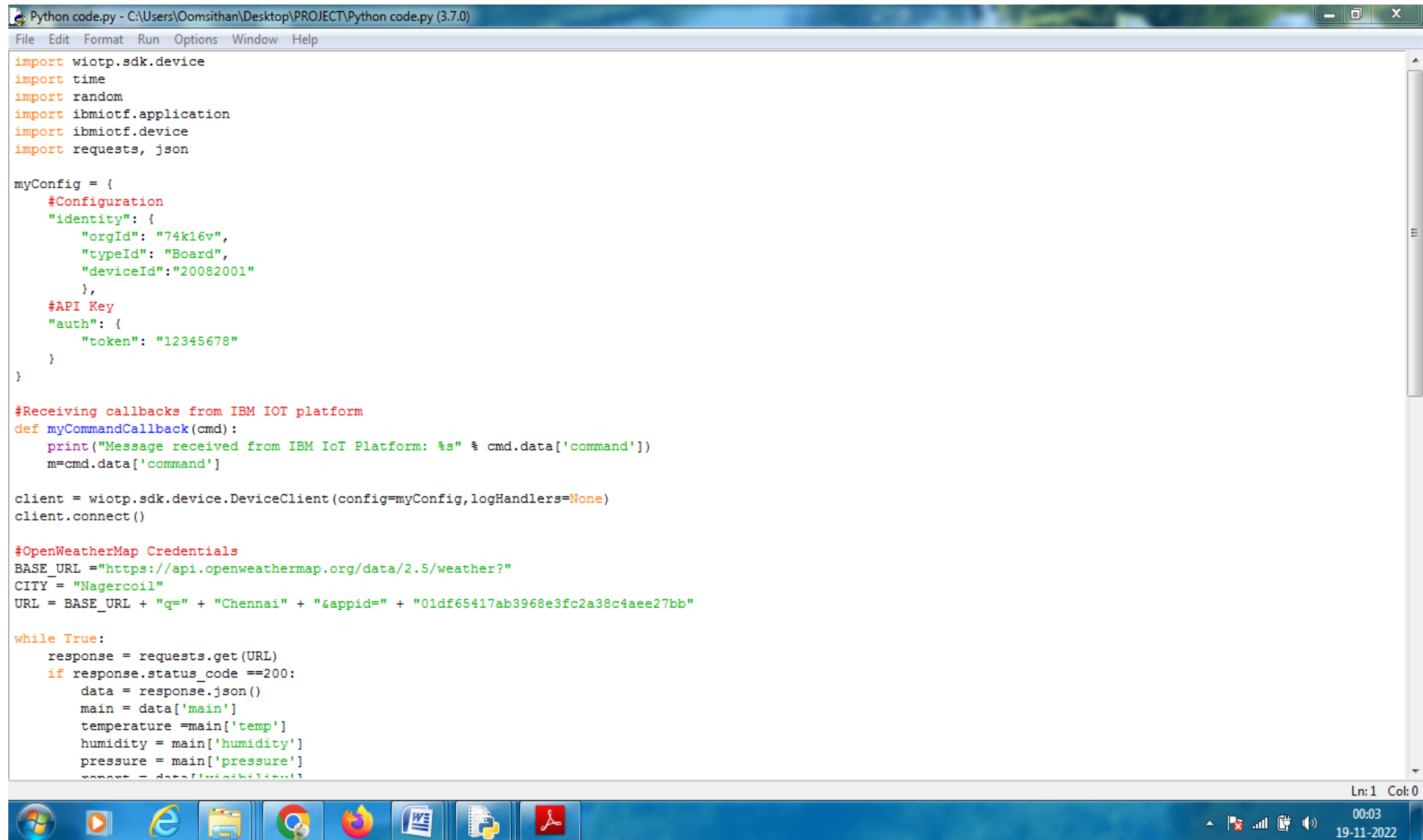
print("Published data Successfully: %s", myData)

client.commandCallback = myCommandCallback

time.sleep(5)

client.disconnect()

Python Script Developed



The image shows a screenshot of a Python script titled "Python code.py" in a code editor window. The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The script is written in Python 3.7.0 and includes imports for "wiottp.sdk.device", "time", "random", "ibmiotf.application", "ibmiotf.device", and "requests, json". It defines a "myConfig" dictionary with "identity" and "API Key" sections. A function "myCommandCallback" is defined to handle incoming commands. The script then creates a "DeviceClient" and connects it. It also defines "BASE_URL", "CITY", and "URL" for fetching weather data from OpenWeatherMap. A "while True" loop is used to continuously fetch and print weather data.

```
Python code.py - C:\Users\Oomsithan\Desktop\PROJECT\Python code.py (3.7.0)
File Edit Format Run Options Window Help

import wiottp.sdk.device
import time
import random
import ibmiotf.application
import ibmiotf.device
import requests, json

myConfig = {
    #Configuration
    "identity": {
        "orgId": "74k16v",
        "typeId": "Board",
        "deviceId": "20082001"
    },
    #API Key
    "auth": {
        "token": "12345678"
    }
}

#Receiving callbacks from IBM IOT platform
def myCommandCallback(cmd):
    print("Message received from IBM IoT Platform: %s" % cmd.data['command'])
    m=cmd.data['command']

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

#OpenWeatherMap Credentials
BASE_URL ="https://api.openweathermap.org/data/2.5/weather?"
CITY = "Nagercoil"
URL = BASE_URL + "q=" + "Chennai" + "&appid=" + "01df65417ab3968e3fc2a38c4aee27bb"

while True:
    response = requests.get(URL)
    if response.status_code ==200:
        data = response.json()
        main = data['main']
        temperature =main['temp']
        humidity = main['humidity']
        pressure = main['pressure']
        report = data['visibility']
```

Ln:1 Col:0

00:03
19-11-2022

```
response = requests.get(URL)
if response.status_code ==200:
    data = response.json()
    main = data['main']
    temperature =main['temp']
    humidity = main['humidity']
    pressure = main['pressure']
    report = data['visibility']

#message part
msg=random.randint(0,5)
if msg==1:
    message="MESSAGE: SLOW DOWN, SCHOOL IS NEAR"
elif msg==3:
    message="MESSAGE: SLOW DOWN, HOSPITAL NEARBY"
elif msg==5:
    message="NEED HELP, POLICE STATION NEARBY"
else:
    message=""

#Speed part
speed=random.randint(0,150)
if speed>=100:
    speedMsg="SPEED MESSAGE: SLOW DOWN, speed Limit Exceeded"
elif speed>=60 and speed<100:
    speedMsg="SPEED MESSAGE: Moderate"
else:
    speedMsg="SPEED MESSAGE: Slow"

#Sign part
sign=random.randint(0,5)
if sign==1:
    signMsg="SIGN: Right Diversion"
elif sign==3:
    signMsg="SIGN: Left Diversion"
elif sign==5:
    signMsg="SIGN: U Turn"
else:
    signMsg=""

#Visibility
if temperature < 50:
```




```
Python code.py - C:\Users\Oomsithan\Desktop\PROJECT\Python code.py (3.7.0)
File Edit Format Run Options Window Help
message= MESSAGE: SLOW DOWN, HOSPITAL NEARBY
elif msg==5:
    message="NEED HELP, POLICE STATION NEARBY"
else:
    message=""

#Speed part
speed=random.randint(0,150)
if speed>=100:
    speedMsg="SPEED MESSAGE: SLOW DOWN, speed Limit Exceeded"
elif speed>=60 and speed<100:
    speedMsg="SPEED MESSAGE: Moderate"
else:
    speedMsg="SPEED MESSAGE: Slow"

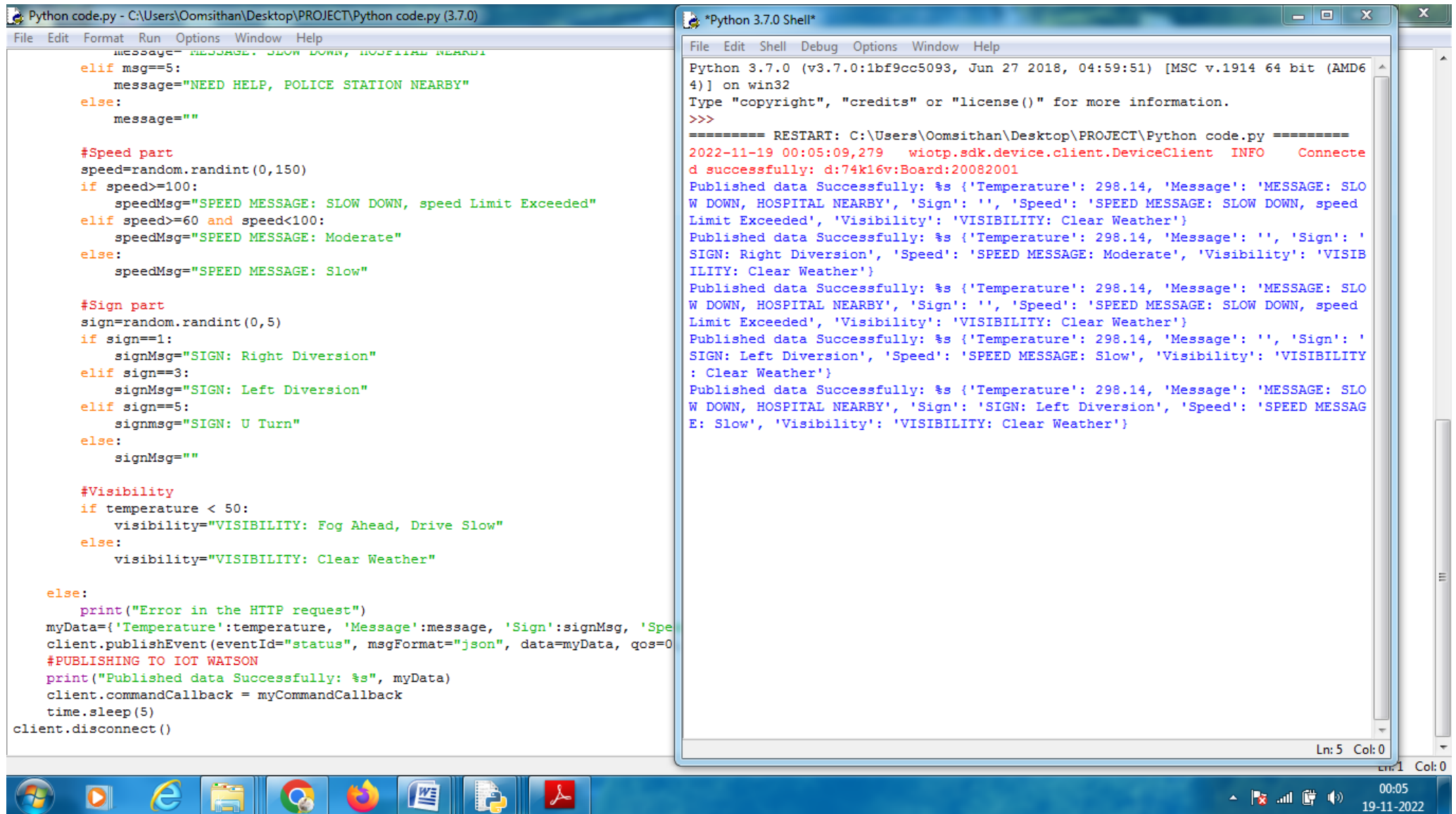
#Sign part
sign=random.randint(0,5)
if sign==1:
    signMsg="SIGN: Right Diversion"
elif sign==3:
    signMsg="SIGN: Left Diversion"
elif sign==5:
    signMsg="SIGN: U Turn"
else:
    signMsg=""

#Visibility
if temperature < 50:
    visibility="VISIBILITY: Fog Ahead, Drive Slow"
else:
    visibility="VISIBILITY: Clear Weather"

else:
    print("Error in the HTTP request")
myData={'Temperature':temperature, 'Message':message, 'Sign':signMsg, 'Speed':speedMsg, 'Visibility':visibility}
client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
#PUBLISHING TO IOT WATSON
print("Published data Successfully: %s", myData)
client.commandCallback = myCommandCallback
time.sleep(5)
client.disconnect()
```

Send the extracted data to the cloud by running the code

Receive data from the cloud and view it in the python compiler(Python code output)



The image shows a screenshot of a Python IDE with two windows. The left window, titled 'Python code.py - C:\Users\Oomsithan\Desktop\PROJECT\Python code.py (3.7.0)', contains Python code that generates a JSON message based on speed, sign, and visibility. The right window, titled '*Python 3.7.0 Shell*', shows the output of the code execution, including a restart message and several 'Published data Successfully' logs with JSON data.

```
Python code.py - C:\Users\Oomsithan\Desktop\PROJECT\Python code.py (3.7.0)
File Edit Format Run Options Window Help
message="MESSAGE: SLOW DOWN, HOSPITAL NEARBY"
elif msg==5:
    message="NEED HELP, POLICE STATION NEARBY"
else:
    message=""

#Speed part
speed=random.randint(0,150)
if speed>=100:
    speedMsg="SPEED MESSAGE: SLOW DOWN, speed Limit Exceeded"
elif speed>=60 and speed<100:
    speedMsg="SPEED MESSAGE: Moderate"
else:
    speedMsg="SPEED MESSAGE: Slow"

#Sign part
sign=random.randint(0,5)
if sign==1:
    signMsg="SIGN: Right Diversion"
elif sign==3:
    signMsg="SIGN: Left Diversion"
elif sign==5:
    signMsg="SIGN: U Turn"
else:
    signMsg=""

#Visibility
if temperature < 50:
    visibility="VISIBILITY: Fog Ahead, Drive Slow"
else:
    visibility="VISIBILITY: Clear Weather"

else:
    print("Error in the HTTP request")
myData={'Temperature':temperature, 'Message':message, 'Sign':signMsg, 'Speed':speedMsg, 'Visibility':visibility}
client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0)
#PUBLISHING TO IOT WATSON
print("Published data Successfully: %s", myData)
client.commandCallback = myCommandCallback
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\Oomsithan\Desktop\PROJECT\Python code.py =====
2022-11-19 00:05:09,279 wiotp.sdk.device.client.DeviceClient INFO Connected successfully: d:74k16v:Board:20082001
Published data Successfully: %s {'Temperature': 298.14, 'Message': 'MESSAGE: SLOW DOWN, HOSPITAL NEARBY', 'Sign': '', 'Speed': 'SPEED MESSAGE: SLOW DOWN, speed Limit Exceeded', 'Visibility': 'VISIBILITY: Clear Weather'}
Published data Successfully: %s {'Temperature': 298.14, 'Message': '', 'Sign': 'SIGN: Right Diversion', 'Speed': 'SPEED MESSAGE: Moderate', 'Visibility': 'VISIBILITY: Clear Weather'}
Published data Successfully: %s {'Temperature': 298.14, 'Message': 'MESSAGE: SLOW DOWN, HOSPITAL NEARBY', 'Sign': '', 'Speed': 'SPEED MESSAGE: SLOW DOWN, speed Limit Exceeded', 'Visibility': 'VISIBILITY: Clear Weather'}
Published data Successfully: %s {'Temperature': 298.14, 'Message': '', 'Sign': 'SIGN: Left Diversion', 'Speed': 'SPEED MESSAGE: Slow', 'Visibility': 'VISIBILITY: Clear Weather'}
Published data Successfully: %s {'Temperature': 298.14, 'Message': 'MESSAGE: SLOW DOWN, HOSPITAL NEARBY', 'Sign': 'SIGN: Left Diversion', 'Speed': 'SPEED MESSAGE: Slow', 'Visibility': 'VISIBILITY: Clear Weather'}
Ln: 5 Col: 0
```

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3	Develop The Python Script	USN-7	Publish Data To The IBM Cloud Python code is used to send random sensor data to the cloud and also to receive commands from the cloud. When the commands are received just print the statements which represent the control of the devices.	10	High	1.Mugila R 2.Ishwariya P 3.Kalpana T 4.Shreein Fathima S

Send the extracted data to the cloud by running the code

The screenshot displays a Python IDE with two windows. The left window, titled 'Python code.py', contains the following code:

```

message="MESSAGE: SLOW DOWN, HOSPITAL NEARBY"
elif msg==5:
    message="NEED HELP, POLICE STATION NEARBY"
else:
    message=""

#Speed part
speed=random.randint(0,150)
if speed>100:
    speedMsg="SPEED MESSAGE: SLOW DOWN, speed Limit Exceeded"
elif speed>60 and speed<100:
    speedMsg="SPEED MESSAGE: Moderate"
else:
    speedMsg="SPEED MESSAGE: Slow"

#Sign part
sign=random.randint(0,5)
if sign==1:
    signMsg="SIGN: Right Diversion"
elif sign==3:
    signMsg="SIGN: Left Diversion"
elif sign==5:
    signMsg="SIGN: U Turn"
else:
    signMsg=""

#Visibility
if temperature < 50:
    visibility="VISIBILITY: Fog Ahead, Drive Slow"
else:
    visibility="VISIBILITY: Clear Weather"

else:
    print("Error in the HTTP request")
myData={'Temperature':temperature, 'Message':message, 'Sign':signMsg, 'Speed':speed}
client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0)
#PUBLISHING TO IOT WATSON
print("Published data Successfully: %s", myData)
client.commandCallback = myCommandCallback
time.sleep(5)
client.disconnect()

```

The right window, titled '*Python 3.7.0 Shell*', shows the execution output:

```

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\Oomsithan\Desktop\PROJECT\Python code.py =====
2022-11-19 00:05:09,279   wiotp.sdk.device.client.DeviceClient INFO   Connecte
d successfully: d:74k16v:Board:20082001
Published data Successfully: %s {'Temperature': 298.14, 'Message': 'MESSAGE: SLO
W DOWN, HOSPITAL NEARBY', 'Sign': '', 'Speed': 'SPEED MESSAGE: SLOW DOWN, speed
Limit Exceeded', 'Visibility': 'VISIBILITY: Clear Weather'}
Published data Successfully: %s {'Temperature': 298.14, 'Message': '', 'Sign': '
SIGN: Right Diversion', 'Speed': 'SPEED MESSAGE: Moderate', 'Visibility': 'VISIB
ILITY: Clear Weather'}
Published data Successfully: %s {'Temperature': 298.14, 'Message': 'MESSAGE: SLO
W DOWN, HOSPITAL NEARBY', 'Sign': '', 'Speed': 'SPEED MESSAGE: SLOW DOWN, speed
Limit Exceeded', 'Visibility': 'VISIBILITY: Clear Weather'}
Published data Successfully: %s {'Temperature': 298.14, 'Message': '', 'Sign': '
SIGN: Left Diversion', 'Speed': 'SPEED MESSAGE: Slow', 'Visibility': 'VISIBILI
TY: Clear Weather'}
Published data Successfully: %s {'Temperature': 298.14, 'Message': 'MESSAGE: SLO
W DOWN, HOSPITAL NEARBY', 'Sign': 'SIGN: Left Diversion', 'Speed': 'SPEED MESSAG
E: Slow', 'Visibility': 'VISIBILITY: Clear Weather'}

```

IBM Watson IoT device connected

The screenshot displays the IBM Watson IoT Platform dashboard in a web browser. The browser's address bar shows the URL `74k16v.internetofthings.ibmcloud.com/dashboard/devices/browse`. The dashboard header includes the IBM Watson IoT Platform logo, a user profile icon, and the email `621119106017@smartinternz.com` with the device ID `ID: 74k16v`. The main navigation bar has tabs for **Browse**, **Action**, **Device Types**, and **Interfaces**, along with an **Add Device** button. Below the navigation bar, there are buttons for **All Devices** and **Diagnose**. A text block explains that the table shows a summary of all devices and can be filtered, organized, and searched. A search bar labeled **Search by Device ID** is present. The **Device Simulator** toggle is turned on. The main table lists devices with columns for **Device ID**, **Status**, **Device Type**, **Class ID**, and **Date Added**. One device is listed with ID `20082001`, status `Connected`, type `Board`, class ID `Device`, and date `Oct 3, 2022 7:40 PM`. Below the table, there are tabs for **Identity**, **Device Information**, **Recent Events**, **State**, and **Logs**. The **Recent Events** tab is active, showing a message: "The recent events listed show the live stream of data that is coming and going from this device." At the bottom, there is a section for **0 Simulations running** with columns for **Event** and **Value**. The Windows taskbar at the bottom shows various application icons and the system clock indicating `00:06 19-11-2022`.

IBM Watson IoT Platform

621119106017@smartinternz.com
ID: 74k16v

Browse Action Device Types Interfaces

Add Device

All Devices Diagnose

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Search by Device ID

Device Simulator

Device ID	Status	Device Type	Class ID	Date Added
20082001	Connected	Board	Device	Oct 3, 2022 7:40 PM

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.

0 Simulations running

Event Value

Checking Recent events(datas received)

The screenshot shows the IBM Watson IoT Platform interface. The top navigation bar includes the IBM logo and a search icon. The main header displays the user's email (621119106017@smartinternz.com) and ID (74k16v). The left sidebar contains icons for various IoT functions. The main content area shows the 'Browse' tab selected, with a list of devices. The device '20082001' is highlighted, showing it is 'Connected'. Below the device list, the 'Recent Events' tab is active, displaying a table of events. The table has columns for 'Event', 'Value', 'Format', and 'Last Received'. Five events are listed, all with a 'status' event type and a 'json' format. The 'Last Received' column indicates 'a few seconds ago' for each event. At the bottom right, a box indicates '0 Simulations running'.

IBM Watson IoT Platform

621119106017@smartinternz.com
ID: 74k16v

Browse Action Device Types Interfaces

Add Device +

20082001 Connected Board Device Oct 3, 2022 7:40 PM

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	{"Temperature":298.14,"Message":"","Sign":"SIG...	json	a few seconds ago
status	{"Temperature":298.14,"Message":"","Sign":"","S...	json	a few seconds ago
status	{"Temperature":298.14,"Message":"","Sign":"","S...	json	a few seconds ago
status	{"Temperature":298.14,"Message":"","Sign":"","S...	json	a few seconds ago
status	{"Temperature":298.14,"Message":"MESSAGE: S...	json	a few seconds ago

0 Simulations running

Datas in Event Payload

The screenshot displays the IBM Watson IoT Platform dashboard. A modal window titled "Event Payload" is open, showing details for an event. The event name is "status" and it was received on "Nov 19, 2022 12:07 AM". The payload is a JSON object with the following fields:

```
1 {  
2   "Temperature": 298.14,  
3   "Message": "",  
4   "Sign": "SIGN: Right Diversion",  
5   "Speed": "SPEED MESSAGE: SLOW DOWN, speed Limit Exceeded",  
6   "Visibility": "VISIBILITY: Clear Weather"  
7 }
```

The background dashboard shows a list of devices, with "20082001" selected. The "Identity" tab is active, and a table of recent events is visible. The bottom status bar indicates "0 Simulations running".

IBM Watson IoT Platform

621119106017@smartinternz.com
ID: 74k16v

Browse Action Device Types

20082001

Identity Device

The recent events listed:

Event	Value
status	{ "T
status	{ "T
status	{ "T
status	{ "T
status	{ "T
status	{ "T

0 Simulations running

00:07
19-11-2022