Internet of Things lab 5 Fall 2023

Group 2 : Avhishek Biswas, Anuruddha Ekanayake, Amlan Balabantaray, Shaswati Behera

Task 1 : In IoT Central and Device

Requirements

1. Configuring IoT Central
   a. Configure a single IoT Central app (one per team) to observe telemetry of multiple virtual IoT devices that send temperature, pressure, and humidity data and receive SendData commands. The virtual IoT devices should also send LastCommandReceived and LastPowerOn property to record the epoch times for when the device last received a command and when they were last powered on.
   b. Create a device per team member.
   c. Create dashboard with charts for every telemetry data type that will be received. This should include data from all the devices.
2. Configure Device
   a. Implement device code to send temperature, pressure, and humidity using the iotc library. You can simulate the actual values. A device should send data every 60 seconds or whenever it receives a command from IoT Central. It should also send properties whenever appropriate.
   b. Configure the device code in each of the team member's laptop.

Development Plan:

a. Procedure of Solving the Problem

1. Create an App in IoT Central.
2. Add a device template in the App. We call the device template as Sensor.
3. Create a Device after going to the device tab, configure the device to use the Sensor template.
4. Go to the connect tab and copy the Scope ID, Device ID and the primary key.
5. Paste these above copied values in the **IOTC** connection code.

b. Run-time Errors

Error Code	Error
Connect_Fail	fail connection to IoT central
Config	Wrong configuration of the template

2. Test Output:

Subtask 1:

Subpart a.

We configured 2 IoT Central App, each communicating to 2 members in our team.

Avhi_App - Configured to Avhishek and Anuruddha

Amlan_App - Configured to Amlan and Shaswati

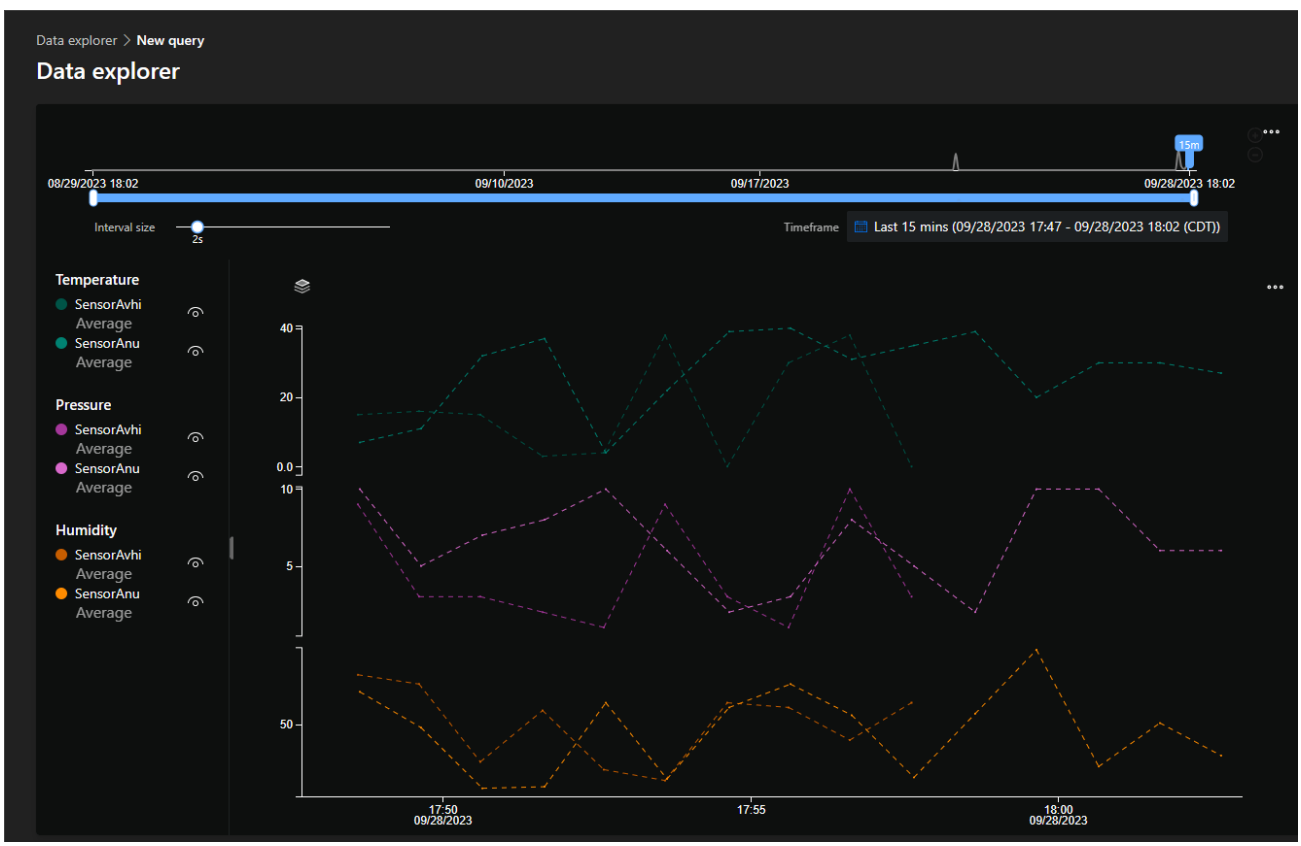


Fig : Output showing data from 2 configured devices in Avhi App.

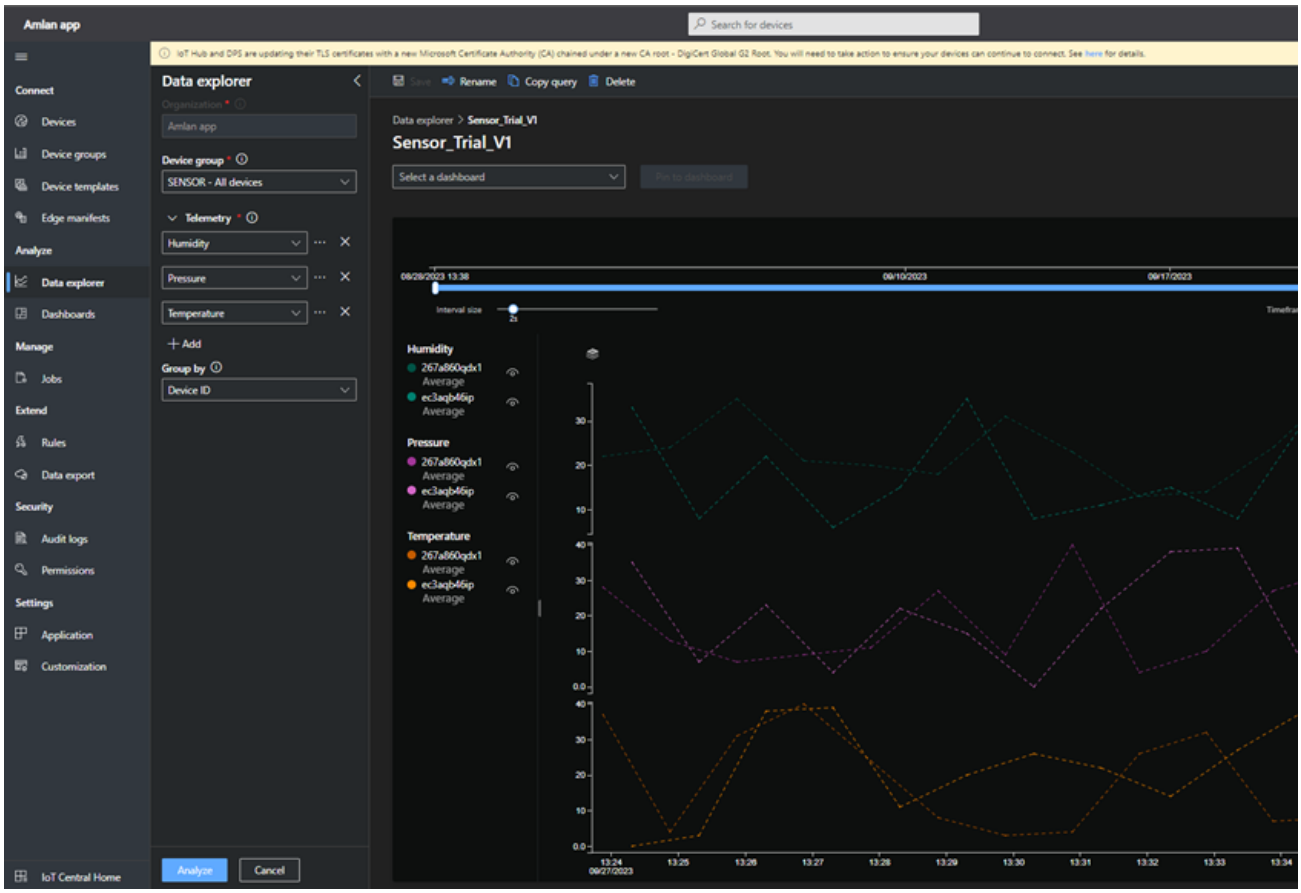


Fig : Output showing data from 2 configured devices in Amlan App.

Subpart b.

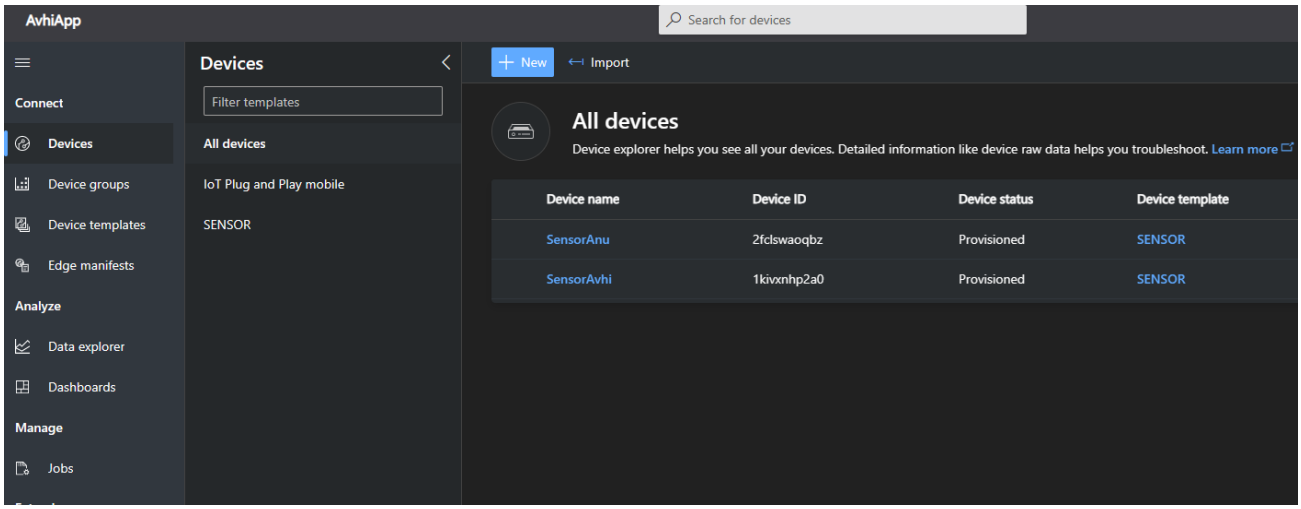


Fig 1:Avhi App Devices page showing 2 connected devices

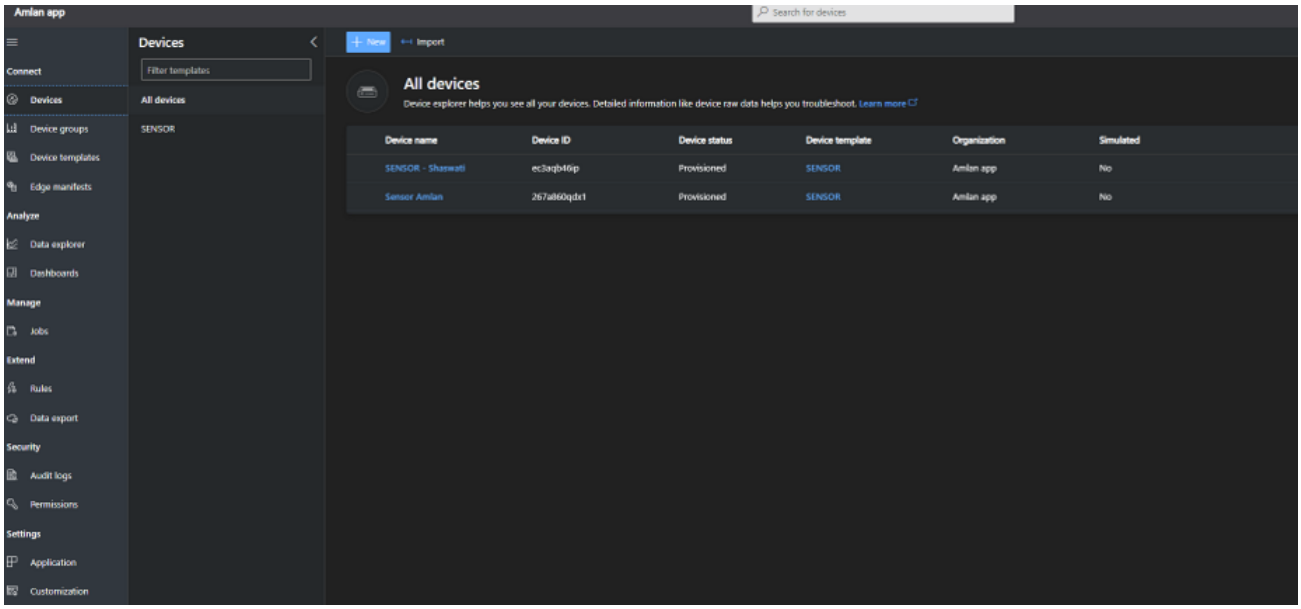


Fig 1:Amlan App Devices page showing 2 connected devices

Screenshots:

Outputs from Avhi:

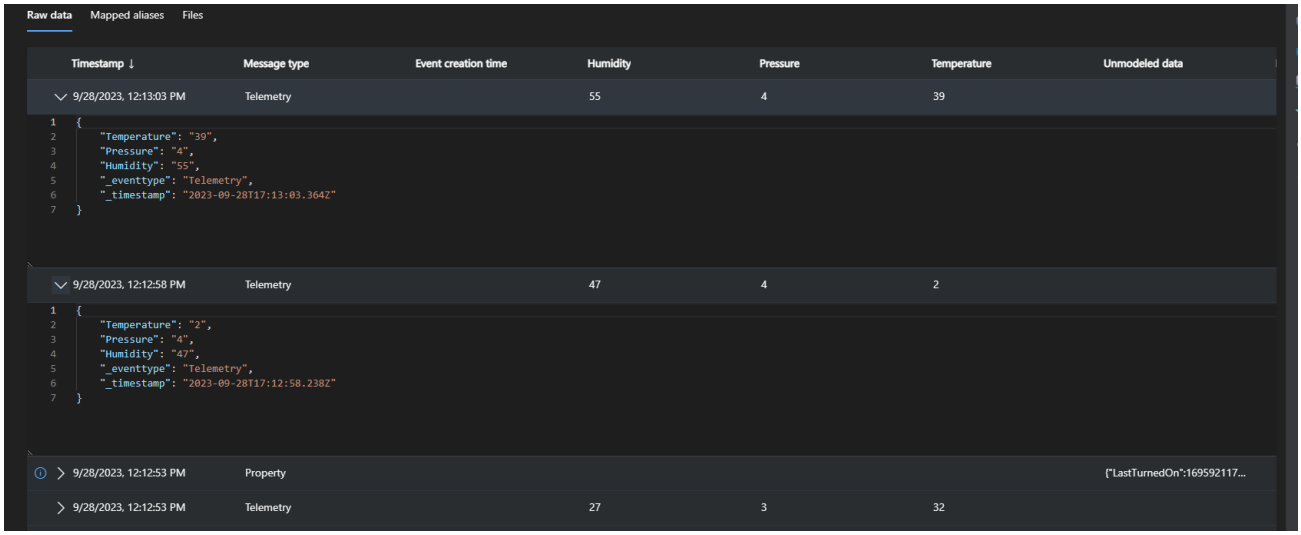


Fig : Raw data in Avhi App for sensor of Avhishek, showing data from Temperature, Pressure and Humidity.

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   PORTS   SERIAL MONITOR   GITLENS   TERMINAL                                   Code

Sending telemetry message: {'Temperature': '18', 'Pressure': '6', 'Humidity': '49'}

Sending telemetry message: {'Temperature': '40', 'Pressure': '4', 'Humidity': '24'}

Sending telemetry message: {'Temperature': '4', 'Pressure': '10', 'Humidity': '28'}

Sending telemetry message: {'Temperature': '25', 'Pressure': '5', 'Humidity': '38'}

Sending telemetry message: {'Temperature': '1', 'Pressure': '5', 'Humidity': '62'}

Sending telemetry message: {'Temperature': '37', 'Pressure': '8', 'Humidity': '78'}

Sending telemetry message: {'Temperature': '32', 'Pressure': '1', 'Humidity': '51'}

Sending telemetry message: {'Temperature': '37', 'Pressure': '3', 'Humidity': '54'}

Sending telemetry message: {'Temperature': '2', 'Pressure': '6', 'Humidity': '7'}

Sending telemetry message: {'Temperature': '26', 'Pressure': '10', 'Humidity': '55'}

Sending telemetry message: {'Temperature': '2', 'Pressure': '8', 'Humidity': '36'}

Sending telemetry message: {'Temperature': '11', 'Pressure': '6', 'Humidity': '67'}
```

Fig : Raw data in Avhi App for sensor of Avhishek, being sent every 60 seconds.

Devices > SENSOR > SensorAvhi

SensorAvhi

Connected | Last data received: 9/28/2023, 9:00:16 PM | Status: Provisioned | Organization: AvhiApp

Commands | Raw data | Mapped aliases | Files

Timestamp ↓	Message type	Event creation time	Humidity	Pressure	Temperature
9/28/2023, 8:59:43 PM	Command response				

```
1 {
2     "LastCommandReceived": 1695952782.3572876,
3     "_eventtype": "Command response",
4     "_timestamp": "2023-09-29T01:59:43.335Z"
5 }
```

9/28/2023, 8:59:43 PM	Command response				

```
1 {
2     "LastPowerOn": 1695952752.1894763,
3     "_eventtype": "Command response",
4     "_timestamp": "2023-09-29T01:59:43.148Z"
5 }
```

Fig : Output of LastPowerOn and LastCommandRecieved Command.

```
[Running] python -u "c:\Users\avhis\Documents\UNL Courses\Internet of Things\Arduino Codes\Lab 5\Lab05_IoTCSample_Avhi.py"
Syncing property '$version'
Sending telemetry message: {'Temperature': '22', 'Pressure': '2', 'Humidity': '38'}

SendTelemetry command was sent
```

Fig : Telemetry Command Recieved.

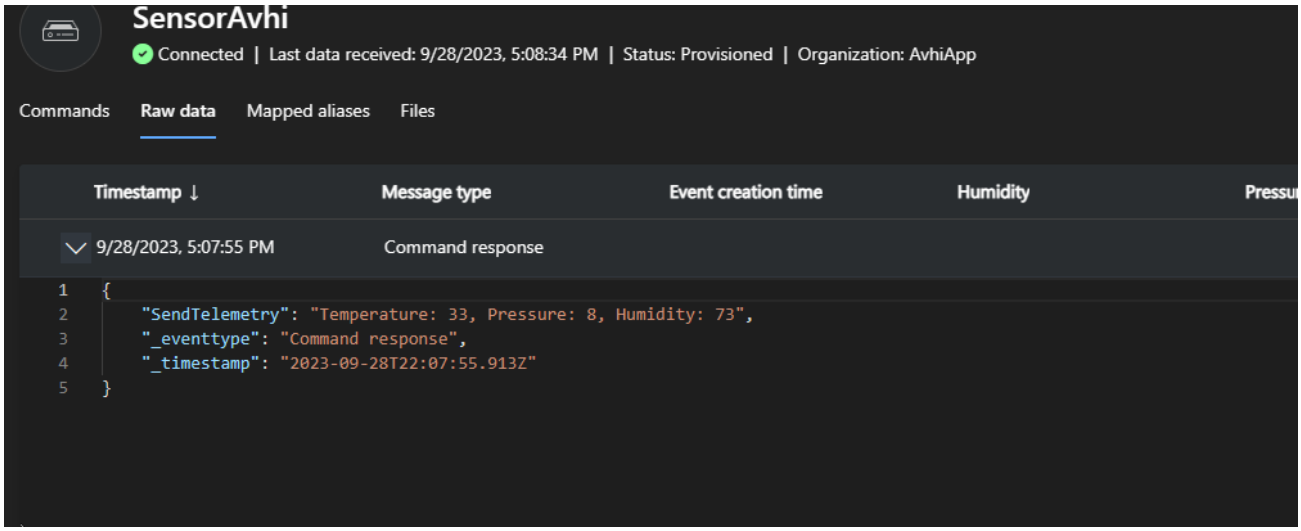
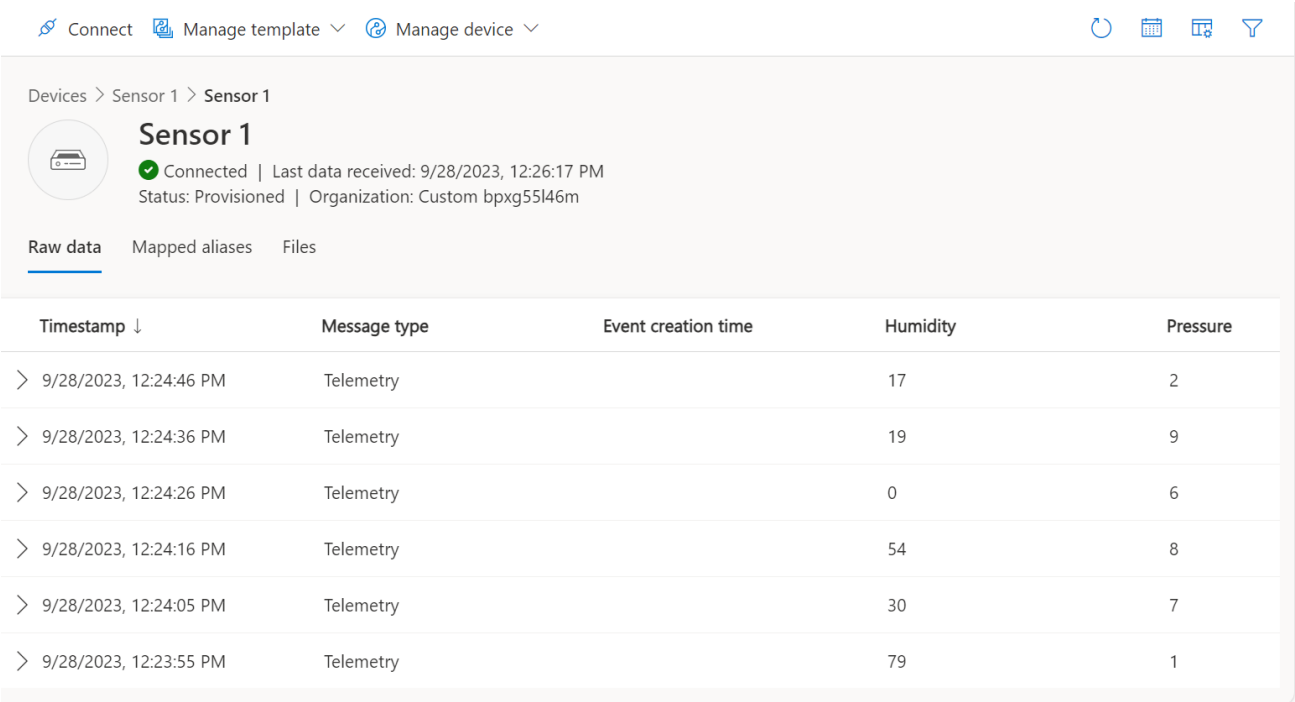


Fig : Telemetry command output

Outputs from Anu:



Timestamp ↓	Message type	Event creation time	Humidity	Pressure
9/28/2023, 12:24:46 PM	Telemetry		17	2
9/28/2023, 12:24:36 PM	Telemetry		19	9
9/28/2023, 12:24:26 PM	Telemetry		0	6
9/28/2023, 12:24:16 PM	Telemetry		54	8
9/28/2023, 12:24:05 PM	Telemetry		30	7
9/28/2023, 12:23:55 PM	Telemetry		79	1

Fig : Raw data in Avhi App for sensors of Anuruddha, showing data from Temperature, Pressure and Humidity.

Lab05_IoTCSample.py Lab5Assignment_Anu.py

Lab5Assignment_Anu.py > ...

```python
    iotc.send_telemetry(telemetry_data)

    # Update the LastTimeTurnedOn property
    current_time = time.time()
    if current_time - last_turned_on_time >= 60:  # Update every 60 seconds
        iotc.send_property({
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE Python

```
Sending telemetry message: {'Temperature': '14', 'Pressure': '8', 'Humidity': '99'}

Sending telemetry message: {'Temperature': '16', 'Pressure': '9', 'Humidity': '78'}

Sending telemetry message: {'Temperature': '19', 'Pressure': '1', 'Humidity': '2'}

Sending telemetry message: {'Temperature': '39', 'Pressure': '7', 'Humidity': '52'}

Sending telemetry message: {'Temperature': '0', 'Pressure': '8', 'Humidity': '1'}

Sending telemetry message: {'Temperature': '22', 'Pressure': '3', 'Humidity': '85'}

Sending telemetry message: {'Temperature': '5', 'Pressure': '4', 'Humidity': '90'}

Sending telemetry message: {'Temperature': '28', 'Pressure': '8', 'Humidity': '24'}

Sending telemetry message: {'Temperature': '17', 'Pressure': '5', 'Humidity': '28'}

Sending telemetry message: {'Temperature': '15', 'Pressure': '1', 'Humidity': '98'}

Sending telemetry message: {'Temperature': '6', 'Pressure': '7', 'Humidity': '64'}
```

Fig : Raw data in Avhi App for sensors of Anuruddha, being sent every 60 seconds.

Lab05_IoTCSample_Anu.py

Lab05_IoTCSample_Anu.py > ...

```python
import random
import time

from iotc.models import Command, Property
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE Python

```
PS E:\IoT\Labs\Lab 5> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python311/python.exe "e:/IoT/Labs/Lab 5/Lab05_IoTCSample
_Anu.py"
Syncing property '$version'
Sending telemetry message: {'Temperature': '32', 'Pressure': '3', 'Humidity': '85'}

Sending telemetry message: {'Temperature': '10', 'Pressure': '7', 'Humidity': '17'}

Sending telemetry message: {'Temperature': '4', 'Pressure': '3', 'Humidity': '95'}

Sending telemetry message: {'Temperature': '38', 'Pressure': '3', 'Humidity': '62'}

Sending telemetry message: {'Temperature': '39', 'Pressure': '7', 'Humidity': '93'}

SendTelemetry command was sent
```

Ln 11, Col 27 Spaces: 4 UTF-8 LF Python 3.11.5 64-bit

Fig : Telemetry Command Recieved.

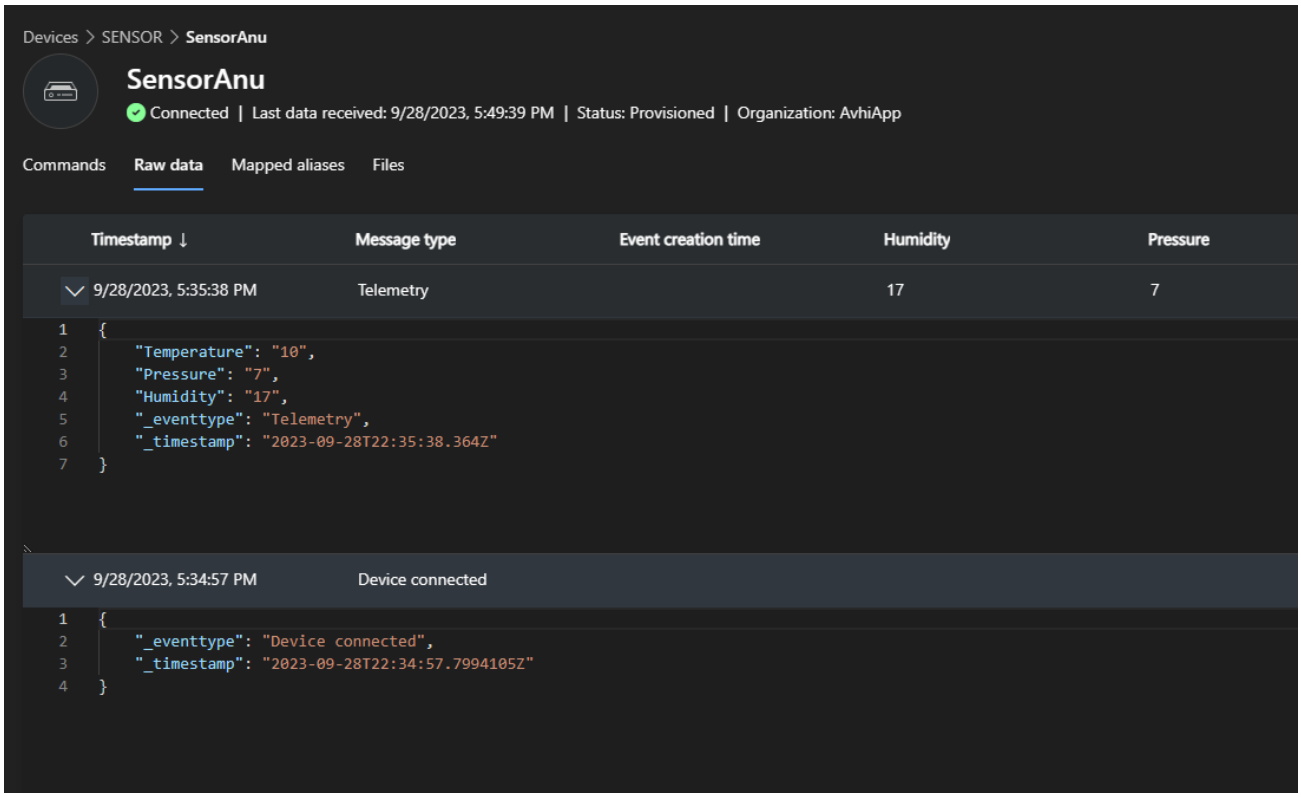


Fig : Telemetry command output

Outputs from Amlan:

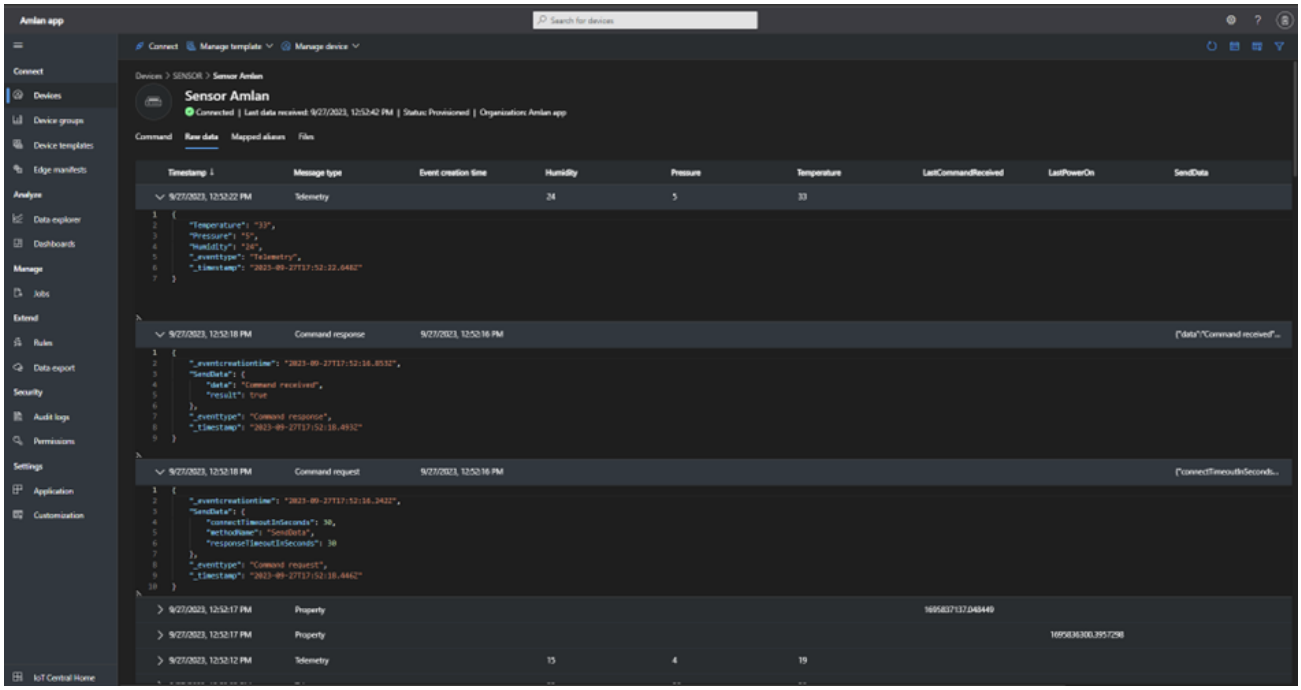


Fig : Raw data in Amlan App for sensors of Amlan, showing data from Temperature, Pressure and Humidity.

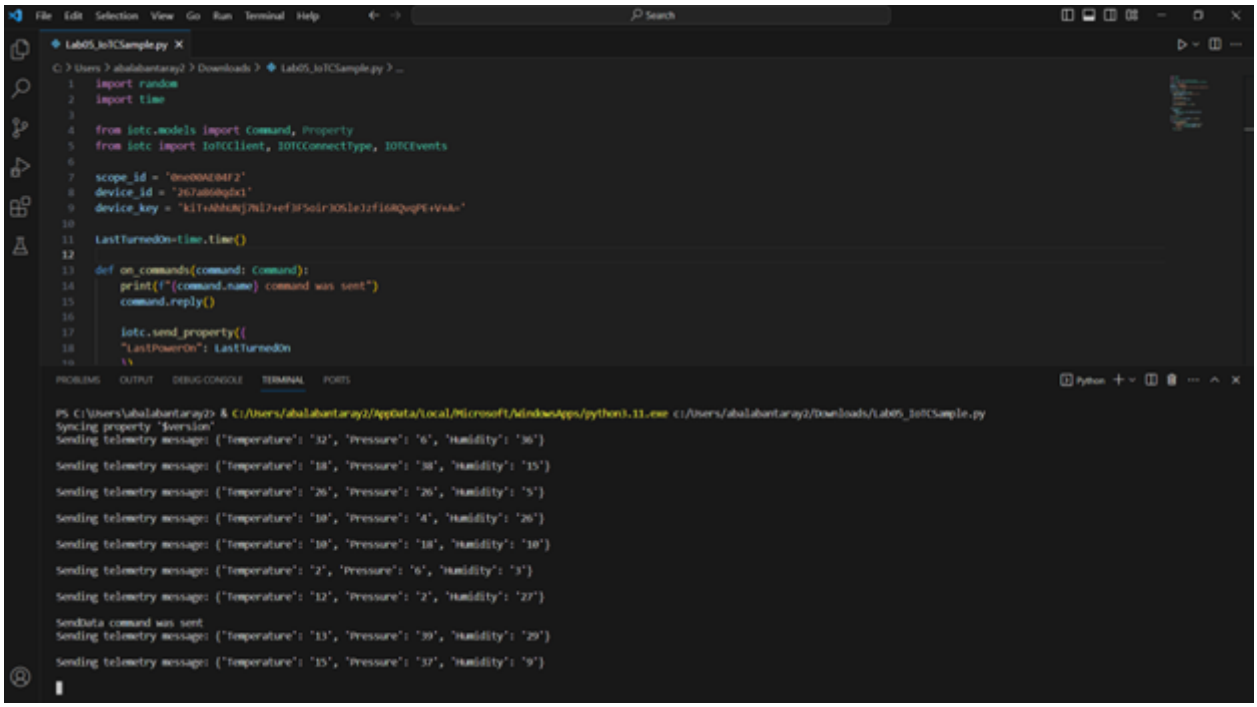


Fig : Raw data in Amlan App for sensors of Amlan, being sent every 60 seconds.

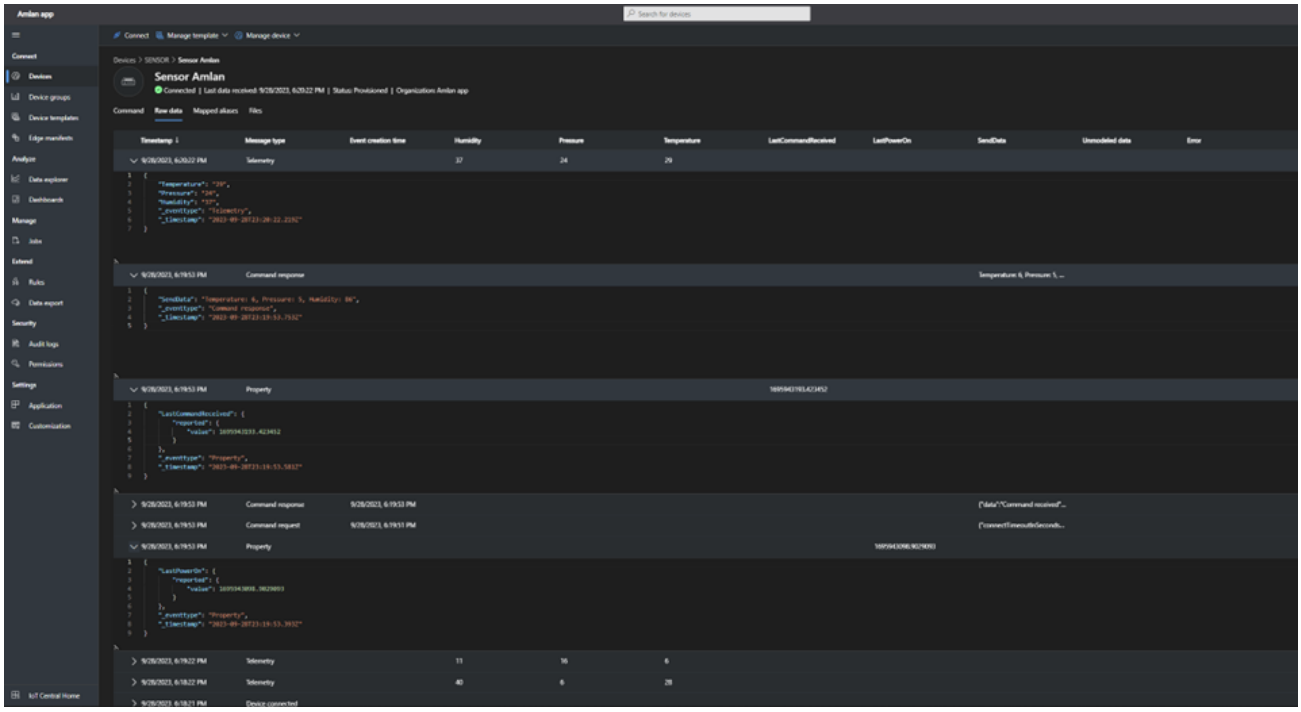


Fig : Output of LastPowerOn and LastCommandRecieved Command.

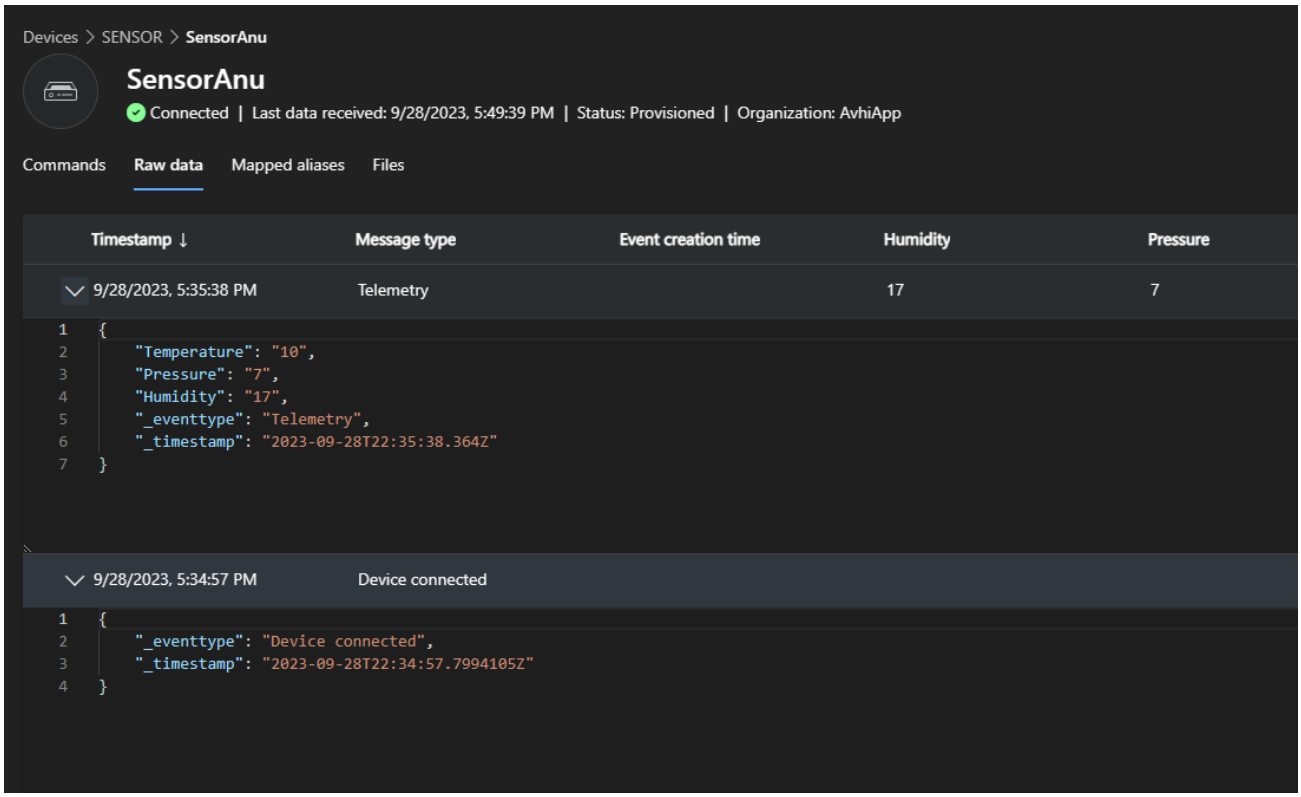


Fig : Telemetry command output

Outputs from Shaswati:

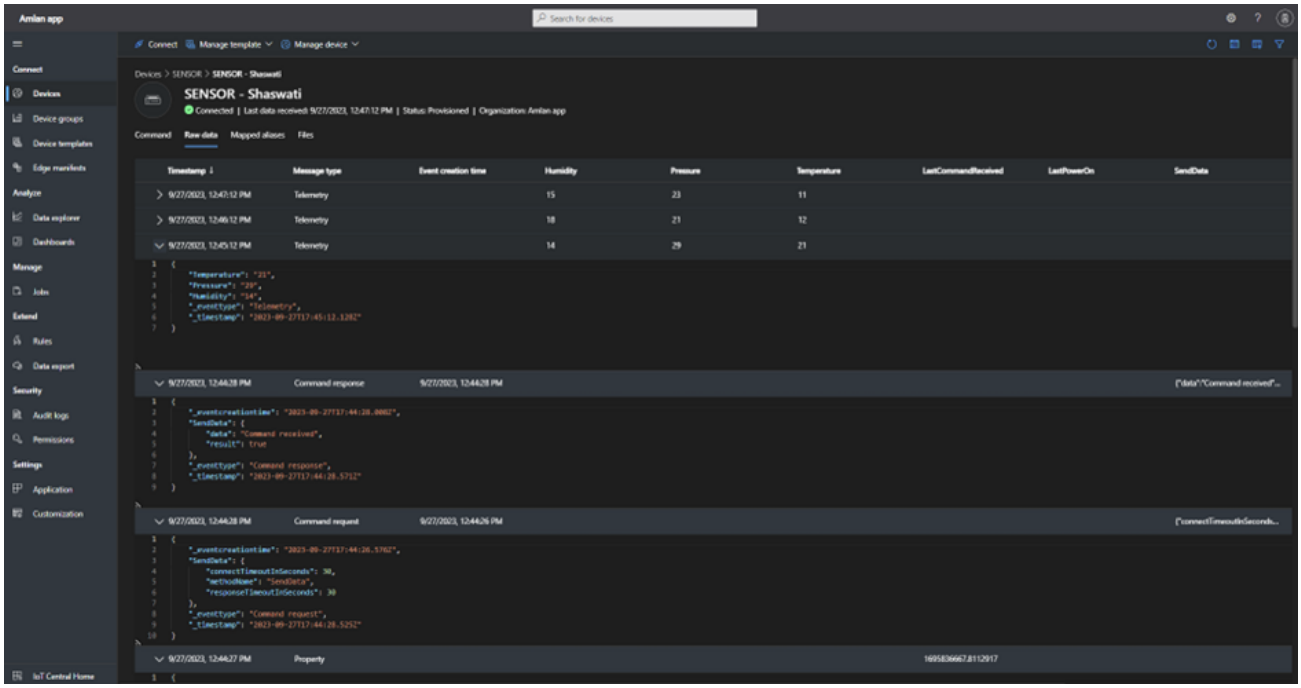


Fig : Raw data in Amlan App for sensors of Shaswati, showing data from Temperature, Pressure and Humidity.

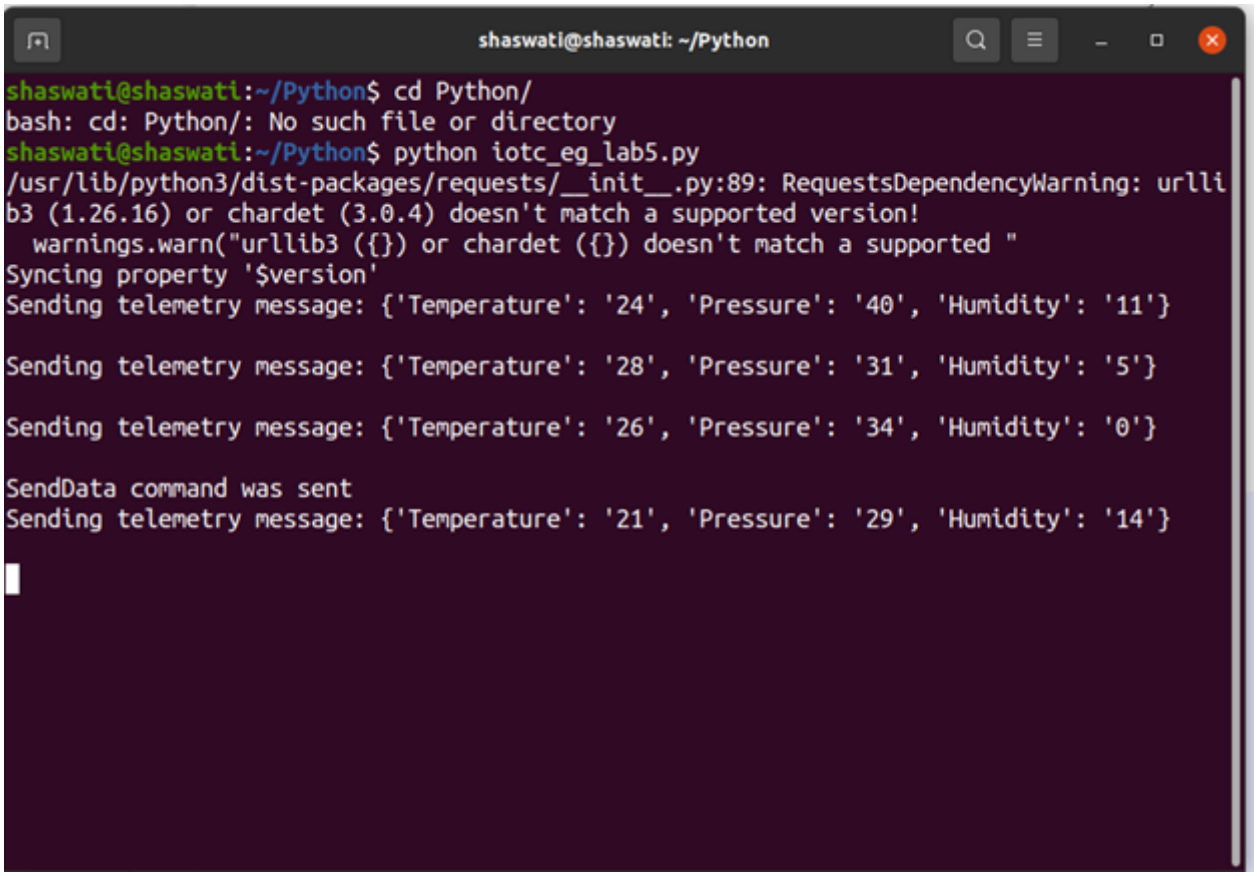


Fig : Raw data in Amlan App for sensors of Shaswati, being sent every 60 seconds.

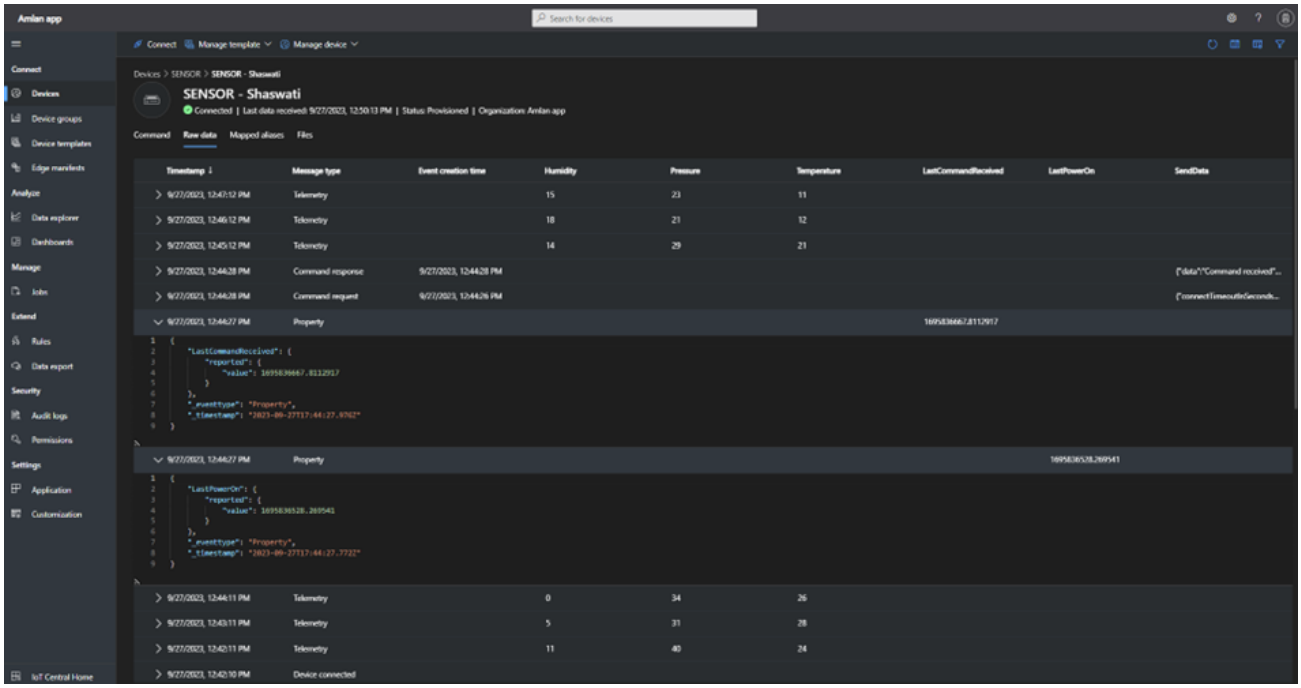


Fig : Output of LastPowerOn and LastCommandRecieved Command.

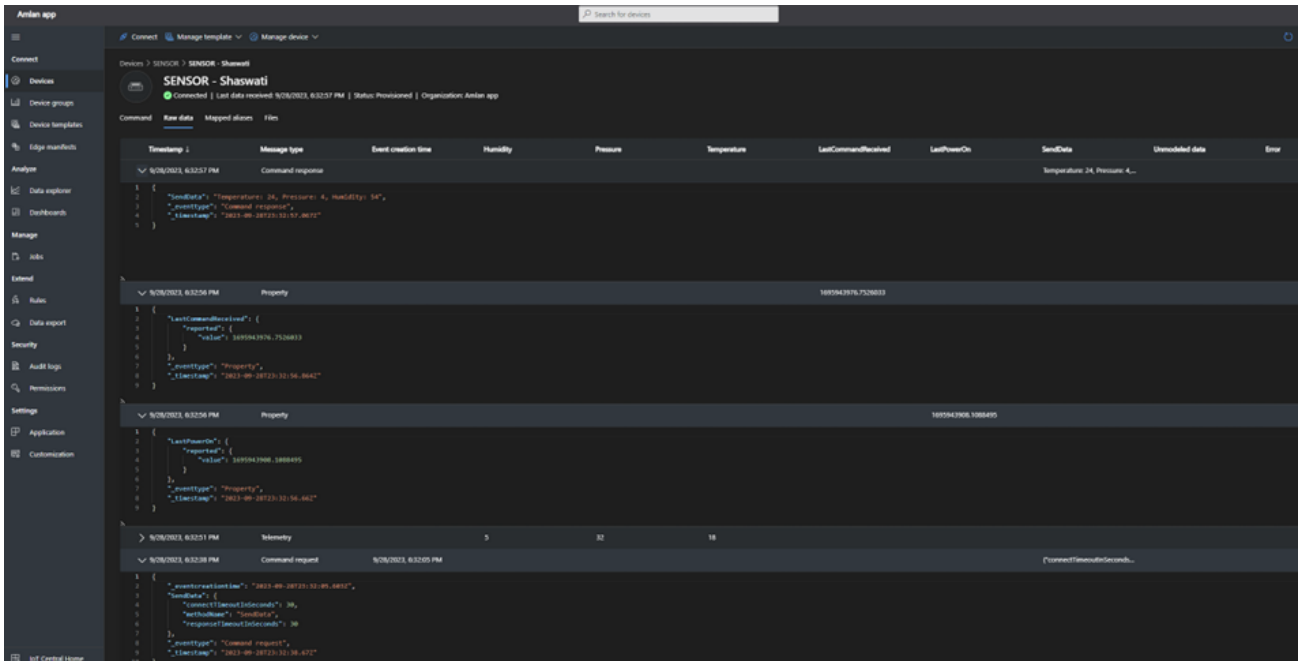


Fig : Output of telemetry Command Recieved.

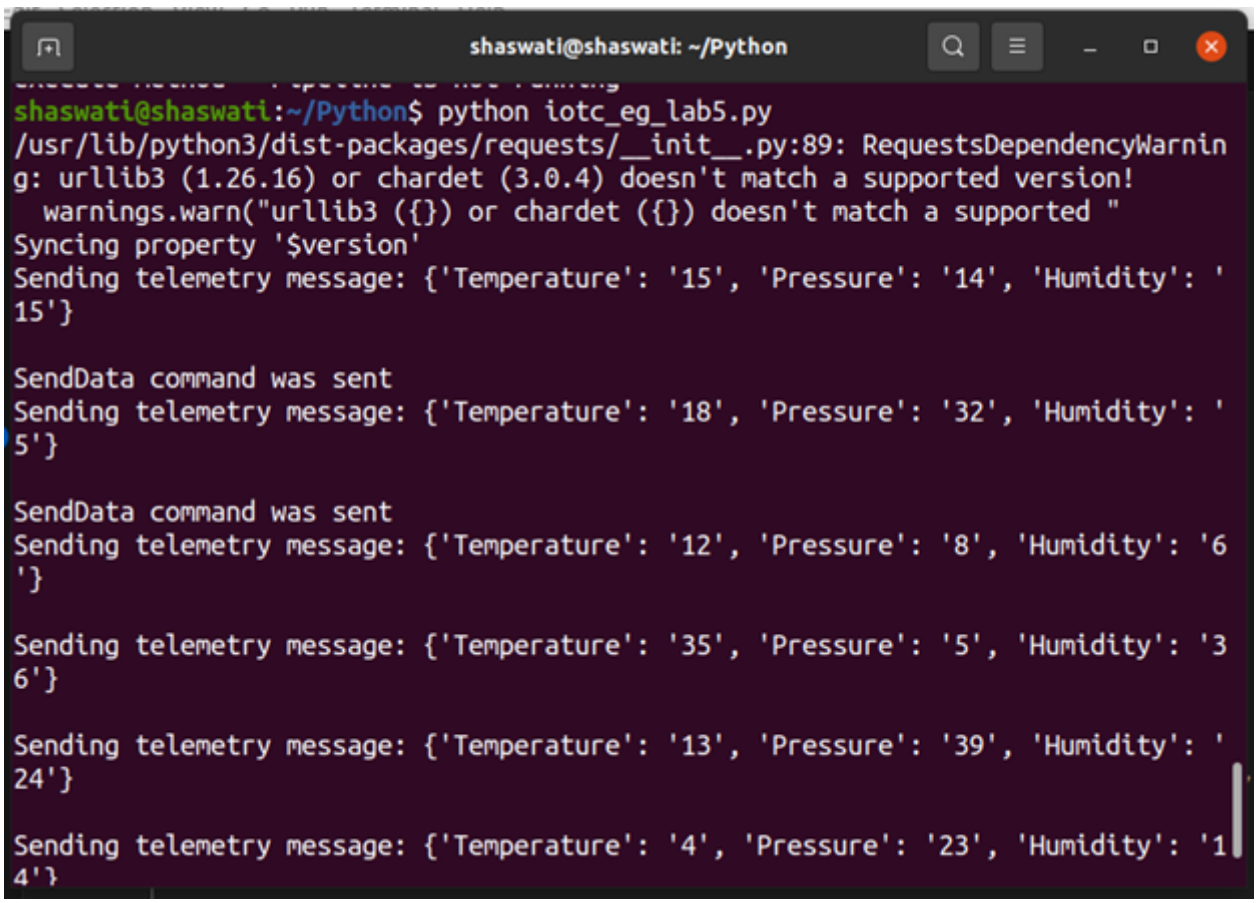


Fig : Telemetry command recieved in device.

Subtask 2:

Subpart a and b.

To send the temperature, pressure and humidity, we add more fields in the json data. Which also sends the data after 60secs which is configured in time.sleep().

```
while iotc.is_connected():
    iotc.send_telemetry({
        'Temperature': str(random.randint(0, 40)),
        'Pressure' : str(random.randint(1, 10)),
        'Humidity' : str(random.randint(0,100))
    })
    # iotc._send_message(str('hello'))
    time.sleep(60)
```

To get the devices to send telemetry when ever it is asked from the IOT Central,first we configure the device template, and then change the code to reply to such a command:

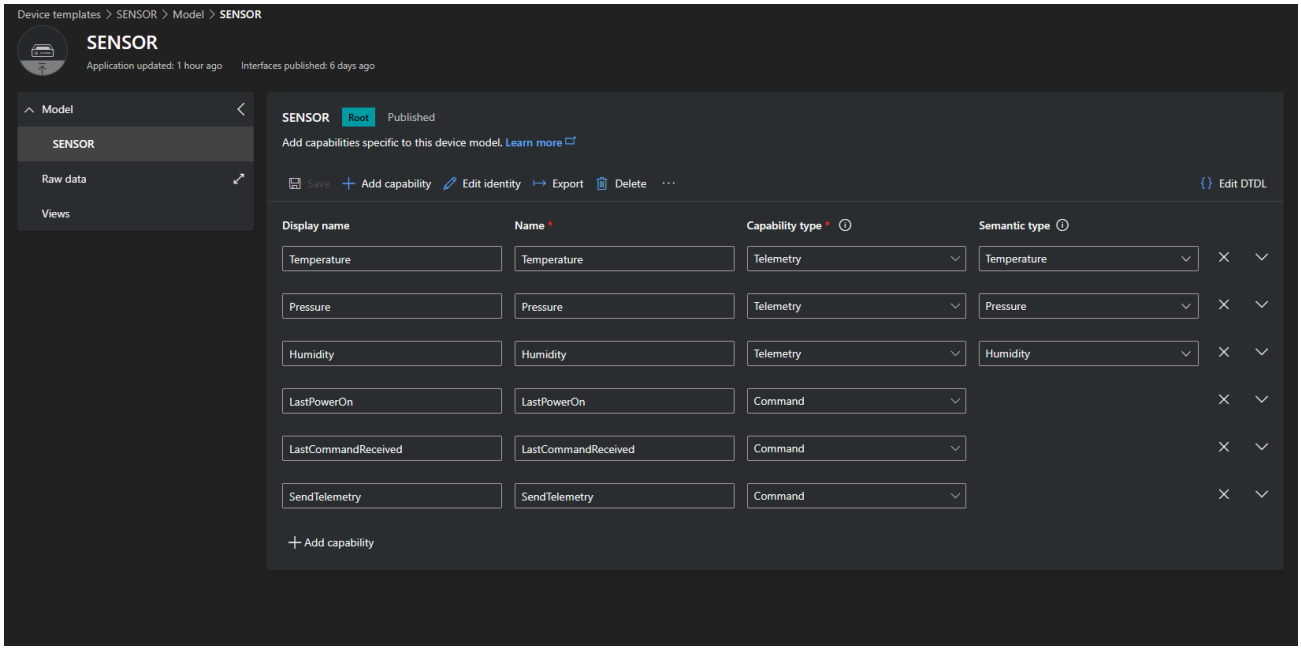


Fig 1:Sensor Template

Code to return to the IoT Central -

```
temp = str(random.randint(0, 40))
pressure = str(random.randint(1, 10))
humidity =  str(random.randint(0,100))

telemetry_str = "Temperature: {}, Pressure: {}, Humidity: {}".format(temp, pressure, humidity)


def on_commands(command: Command):

    print(f"{command.name} command was sent")
    iotc.send_property({
    "LastPowerOn": LastTurnedOn
    })
    iotc.send_property({
    "LastCommandReceived": time.time()
    })
    iotc.send_property({
        "SendTelemetry" : telemetry_str
    })
```

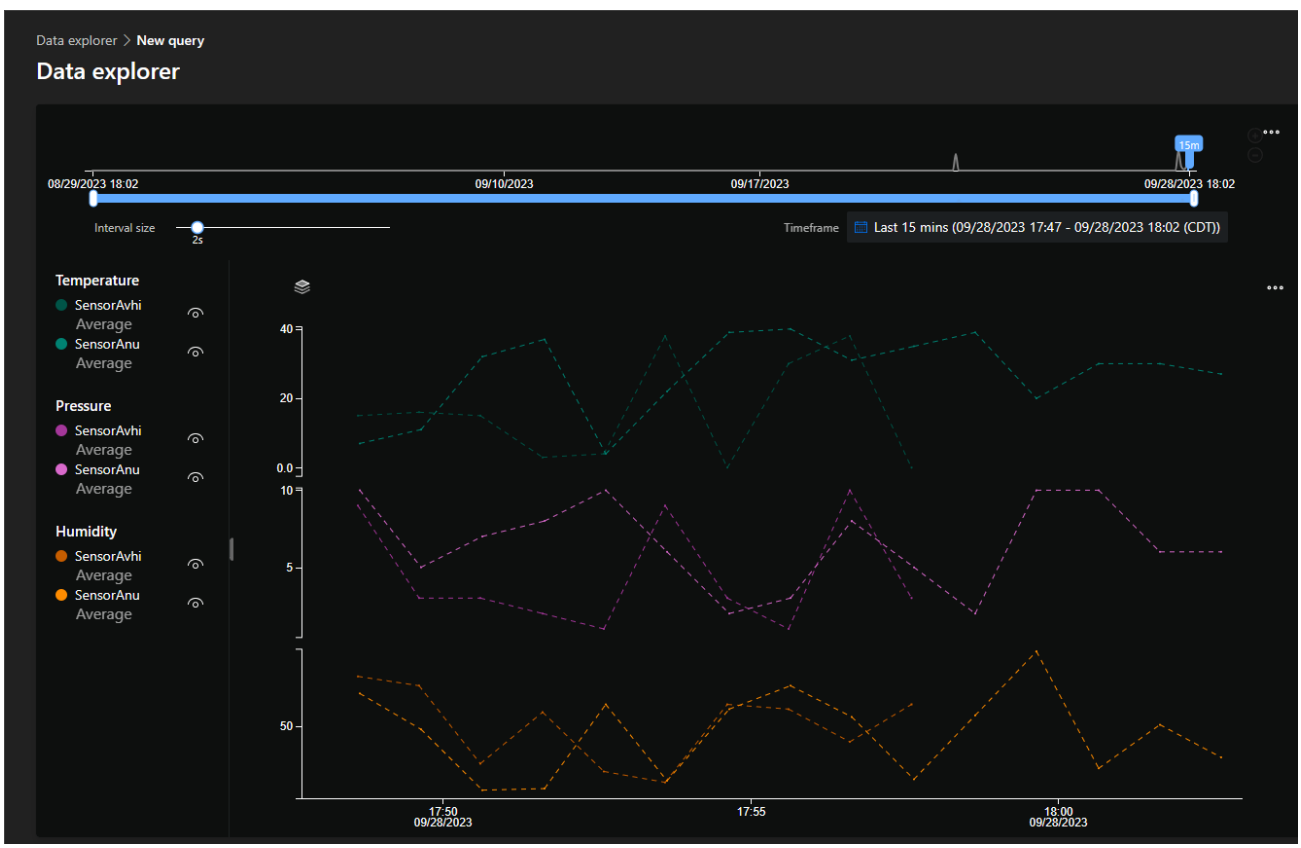


Fig : Output showing data from 2 devices in Avhi App.

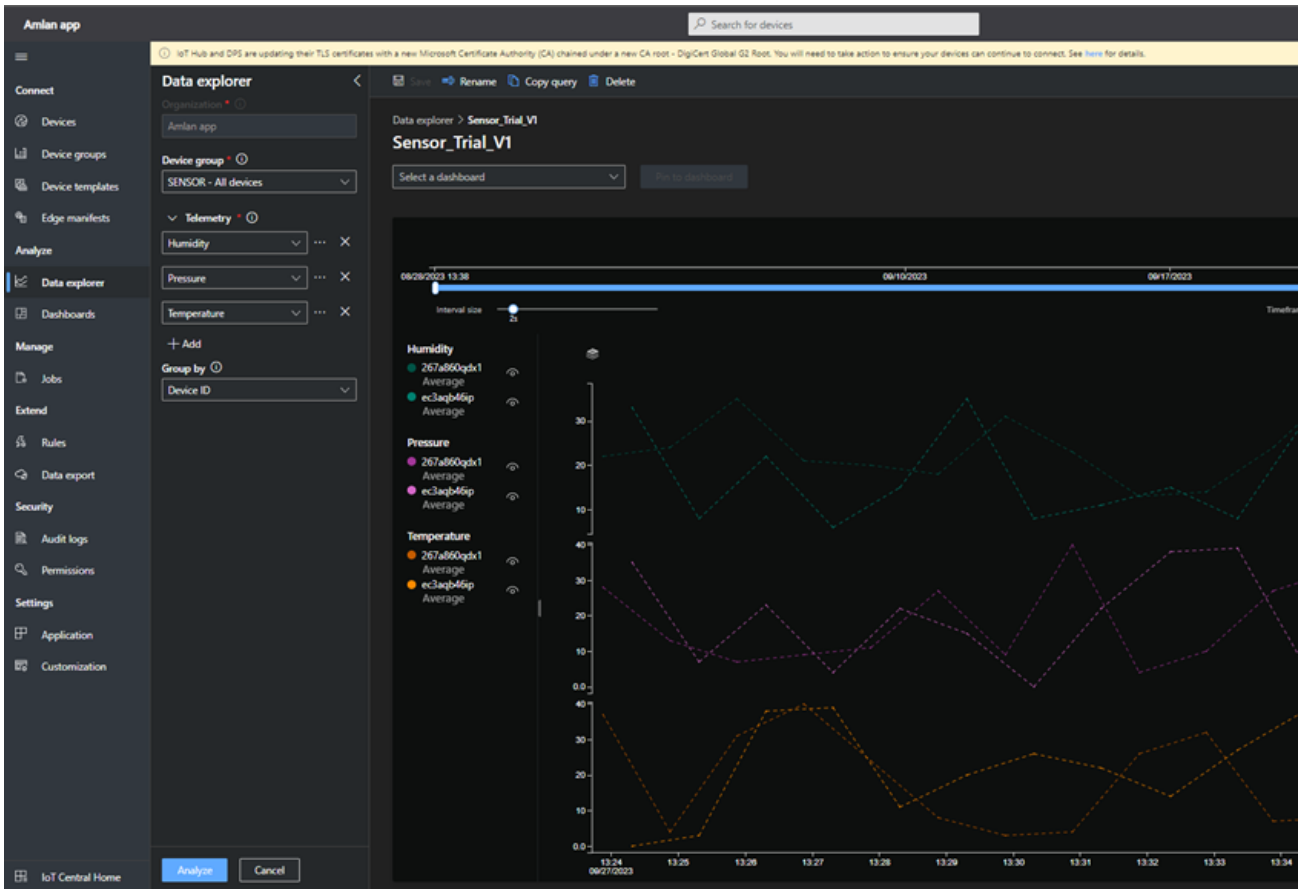


Fig : Output showing data from 2 devices in Amlan App.

Appendix

Amlan and Shaswati :

Amlan Code :

```
import random
import time

from iotc.models import Command, Property
from iotc import IoTCClient, IOTCConnectType, IOTCEvents

scope_id = '0ne00AE04F2'
device_id = '267a860qdx1'
device_key = 'kiT+AhhUNj7Nl7+ef3F5oir3OSleJzfi6RQvqPE+V+A='

LastTurnedOn = time.time()
temp = str(random.randint(0, 40))
pressure = str(random.randint(1, 10))
humidity =  str(random.randint(0,100))

telemetry_str = "Temperature: {}, Pressure: {}, Humidity: {}".format(temp, pressure, humidity)


def on_commands(command: Command):

    print(f"{command.name} command was sent")
    iotc.send_property({
    "LastPowerOn": LastTurnedOn
    })
    iotc.send_property({
    "LastCommandReceived": time.time()
    })
    iotc.send_property({
        # "SendTelemetry" : iotc.send_telemetry({
        #                                         'Temperature': str(random.randint(0, 40)),
        #                                         'Pressure' : str(random.randint(1, 10)),
        #                                         'Humidity' : str(random.randint(0,100))
        #                                         })
        "SendTelemetry" : telemetry_str
    })
```

```
iotc = IoTCClient(
    device_id,
    scope_id,
        IOTCConnectType.IOTC_CONNECT_DEVICE_KEY,
        device_key)

iotc.connect()

iotc.on(IOTCEvents.IOTC_COMMAND, on_commands)

iotc.send_property({
    "LastTurnedOn": time.time()
})

while iotc.is_connected():
    iotc.send_telemetry({
        'Temperature': str(random.randint(0, 40)),
        'Pressure' : str(random.randint(1, 10)),
        'Humidity' : str(random.randint(0,100))
    })
    # iotc._send_message(str('hello'))
    time.sleep(60)
```

Shaswati Code :

```
import random
import time

from iotc.models import Command, Property
from iotc import IoTCClient, IOTCConnectType, IOTCEvents

scopeId = '0ne00AE04F2'
device_id = 'ec3aqb46ip'
device_key = 'WDhq8Ep7MheHsWci1qGYqq26XrrimZfJ8nquRgT7A4Y='

LastTurnedOn = time.time()
temp = str(random.randint(0, 40))
pressure = str(random.randint(1, 10))
humidity =  str(random.randint(0,100))

telemetry_str = "Temperature: {}, Pressure: {}, Humidity: {}".format(temp, pressure, humidity)



def on_commands(command: Command):

    print(f"{command.name} command was sent")
    iotc.send_property({
    "LastPowerOn": LastTurnedOn
    })
    iotc.send_property({
    "LastCommandReceived": time.time()
    })
    iotc.send_property({
        # "SendTelemetry" : iotc.send_telemetry({
        #                                         'Temperature': str(random.randint(0, 40)),
        #                                         'Pressure' : str(random.randint(1, 10)),
        #                                         'Humidity' : str(random.randint(0,100))
        #                                         })
        "SendTelemetry" : telemetry_str
    })



iotc = IoTCClient(
    device_id,
    scope_id,
```

```
        IOTCConnectType.IOTC_CONNECT_DEVICE_KEY,
        device_key)

iotc.connect()

iotc.on(IOTCEvents.IOTC_COMMAND, on_commands)

iotc.send_property({
    "LastTurnedOn": time.time()
})

while iotc.is_connected():
    iotc.send_telemetry({
        'Temperature': str(random.randint(0, 40)),
        'Pressure' : str(random.randint(1, 10)),
        'Humidity' : str(random.randint(0,100))
    })
    # iotc._send_message(str('hello'))
    time.sleep(60)
```

Avhi and Anuruddha :

Avhi Code :

```
import random
import time

from iotc.models import Command, Property
from iotc import IoTCClient, IOTCConnectType, IOTCEvents

scope_id = '0ne00ADFCBE'
device_id = '1kivxnhp2a0'
device_key = 'bmOIBczHO7aCsoNhxf+tqhy+BxtbRrnxMmQWnKx8SVI='

LastTurnedOn = time.time()
temp = str(random.randint(0, 40))
pressure = str(random.randint(1, 10))
humidity =  str(random.randint(0,100))

telemetry_str = "Temperature: {}, Pressure: {}, Humidity: {}".format(temp, pressure, humidity)


def on_commands(command: Command):

    print(f"{command.name} command was sent")
    iotc.send_property({
    "LastPowerOn": LastTurnedOn
    })
    iotc.send_property({
    "LastCommandReceived": time.time()
    })
    iotc.send_property({
        "SendTelemetry" : telemetry_str
    })



iotc = IoTCClient(
    device_id,
    scope_id,
        IOTCConnectType.IOTC_CONNECT_DEVICE_KEY,
        device_key)
```

```
iotc.connect()

iotc.on(IOTCEvents.IOTC_COMMAND, on_commands)

iotc.send_property({
    "LastTurnedOn": time.time()
})

while iotc.is_connected():
    iotc.send_telemetry({
        'Temperature': str(random.randint(0, 40)),
        'Pressure' : str(random.randint(1, 10)),
        'Humidity' : str(random.randint(0,100))
    })
    # iotc._send_message(str('hello'))
    time.sleep(60)
```

Anuruddha Code :

```
import random
import time

from iotc.models import Command, Property
from iotc import IoTCClient, IOTCConnectType, IOTCEvents

scope_id = '0ne00ADFCBE'
device_id = '2fclswaoqbz'
device_key = '2s83CCMrNp+8MspcJ45dg4Bxz2YGLQpxRebjrZm2z70='

LastTurnedOn = time.time()
temp = str(random.randint(0, 40))
pressure = str(random.randint(1, 10))
humidity =  str(random.randint(0,100))

telemetry_str = "Temperature: {}, Pressure: {}, Humidity: {}".format(temp, pressure, humidity)



def on_commands(command: Command):

    print(f"{command.name} command was sent")
    iotc.send_property({
    "LastPowerOn": LastTurnedOn
    })
    iotc.send_property({
    "LastCommandReceived": time.time()
    })
    iotc.send_property({
        # "SendTelemetry" : iotc.send_telemetry({
        #                                         'Temperature': str(random.randint(0, 40)),
        #                                         'Pressure' : str(random.randint(1, 10)),
        #                                         'Humidity' : str(random.randint(0,100))
        #                                         })
        "SendTelemetry" : telemetry_str
    })



iotc = IoTCClient(
    device_id,
    scope_id,
```

```
        IOTCConnectType.IOTC_CONNECT_DEVICE_KEY,
        device_key)

iotc.connect()

iotc.on(IOTCEvents.IOTC_COMMAND, on_commands)

iotc.send_property({
    "LastTurnedOn": time.time()
})

while iotc.is_connected():
    iotc.send_telemetry({
        'Temperature': str(random.randint(0, 40)),
        'Pressure' : str(random.randint(1, 10)),
        'Humidity' : str(random.randint(0,100))
    })
    # iotc._send_message(str('hello'))
    time.sleep(60)
```

References:

The refernces are as follows:

1. Azure Tutorial