

# 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.

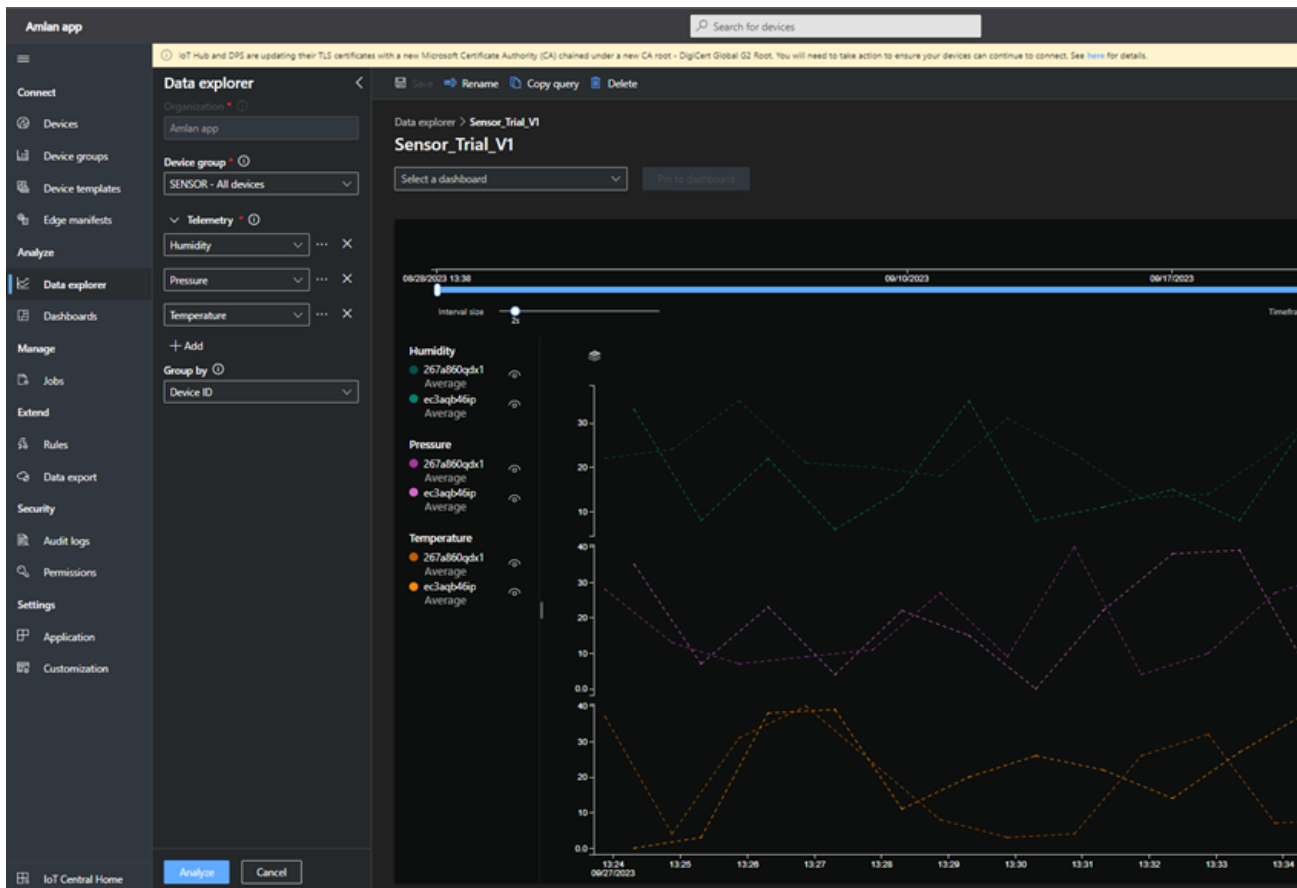


Fig : Output showing data from 2 configured devices in Amlan App.

## Subpart b.

The screenshot shows the AvhiApp interface. The left sidebar contains navigation options: Connect, Devices (selected), Device groups, Device templates, Edge manifests, Analyze, Data explorer, Dashboards, Manage, Jobs, Rules, Data export, Security, Audit logs, Permissions, Settings, Application, and Customization. The main area is titled 'Devices'. It features a 'Filter templates' input field and a 'New' button. Below this is a table showing the details of two connected devices.

Device name	Device ID	Device status	Device template
SensorAnu	2fclswaoqbz	Provisioned	SENSOR
SensorAvhi	1kivxn timerhp2a0	Provisioned	SENSOR

Fig 1:Avhi App Devices page showing 2 connected devices

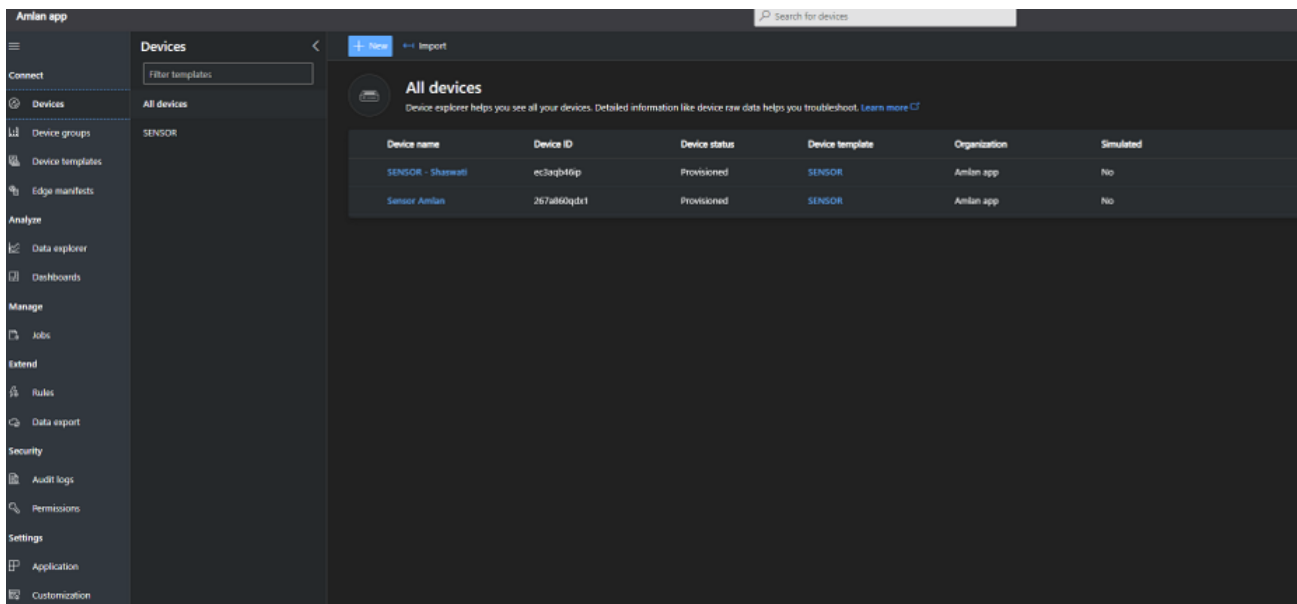


Fig 1:Amlan App Devices page showing 2 connected devices

## Screenshots:

## Outputs from Avhi:


Raw data						
Timestamp ↓	Message type	Event creation time	Humidity	Pressure	Temperature	Unmodeled data
9/28/2023, 12:13:03 PM	Telemetry		55	4	39	
<pre> 1 { 2   "Temperature": "39", 3   "Pressure": "4", 4   "Humidity": "55", 5   "_eventtype": "Telemetry", 6   "_timestamp": "2023-09-28T17:13:03.364Z" 7 } </pre>						
9/28/2023, 12:12:58 PM	Telemetry		47	4	2	
<pre> 1 { 2   "Temperature": "2", 3   "Pressure": "4", 4   "Humidity": "47", 5   "_eventtype": "Telemetry", 6   "_timestamp": "2023-09-28T17:12:58.238Z" 7 } </pre>						
9/28/2023, 12:12:53 PM	Property					["LastTurnedOn":169592117...
9/28/2023, 12:12:53 PM	Telemetry		27	3	32	

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

```
PROBLEMS OUTPUT DEBUG CONSOLE PORTS SERIAL MONITOR GIT LENS 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				
<pre>1 { 2   "LastCommandReceived": 1695952782.3572876, 3   "_eventtype": "Command response", 4   "_timestamp": "2023-09-29T01:59:43.335Z" 5 }</pre>					
9/28/2023, 8:59:43 PM	Command response				
<pre>1 { 2   "LastPowerOn": 1695952752.1894763, 3   "_eventtype": "Command response", 4   "_timestamp": "2023-09-29T01:59:43.148Z" 5 }</pre>					

**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
Sending telemetry message: {'Temperature': '10', 'Pressure': '10', 'Humidity': '37'}
```

**Fig : Telemetry Command Recieved.**



## SensorAvhi

✓ Connected | 
 Last data received: 9/28/2023, 5:08:34 PM | 
 Status: Provisioned | 
 Organization: AvhiApp

[Commands](#)
[Raw data](#)
[Mapped aliases](#)
[Files](#)

Timestamp ↓	Message type	Event creation time	Humidity	Pressure
<div> <div>▼</div> <div>9/28/2023, 5:07:55 PM</div> </div>	Command response			
<div> <div>1</div> <div>{</div> <div>2</div> <div>  "SendTelemetry": "Temperature: 33, Pressure: 8, Humidity: 73",</div> <div>3</div> <div>  "_eventtype": "Command response",</div> <div>4</div> <div>  "_timestamp": "2023-09-28T22:07:55.913Z"</div> <div>5</div> <div>}</div> </div>				

Fig : Telemetry command output

Outputs from Anu:

Connect
 Manage template
 Manage device


Devices

>

Sensor 1

>

Sensor 1



## Sensor 1

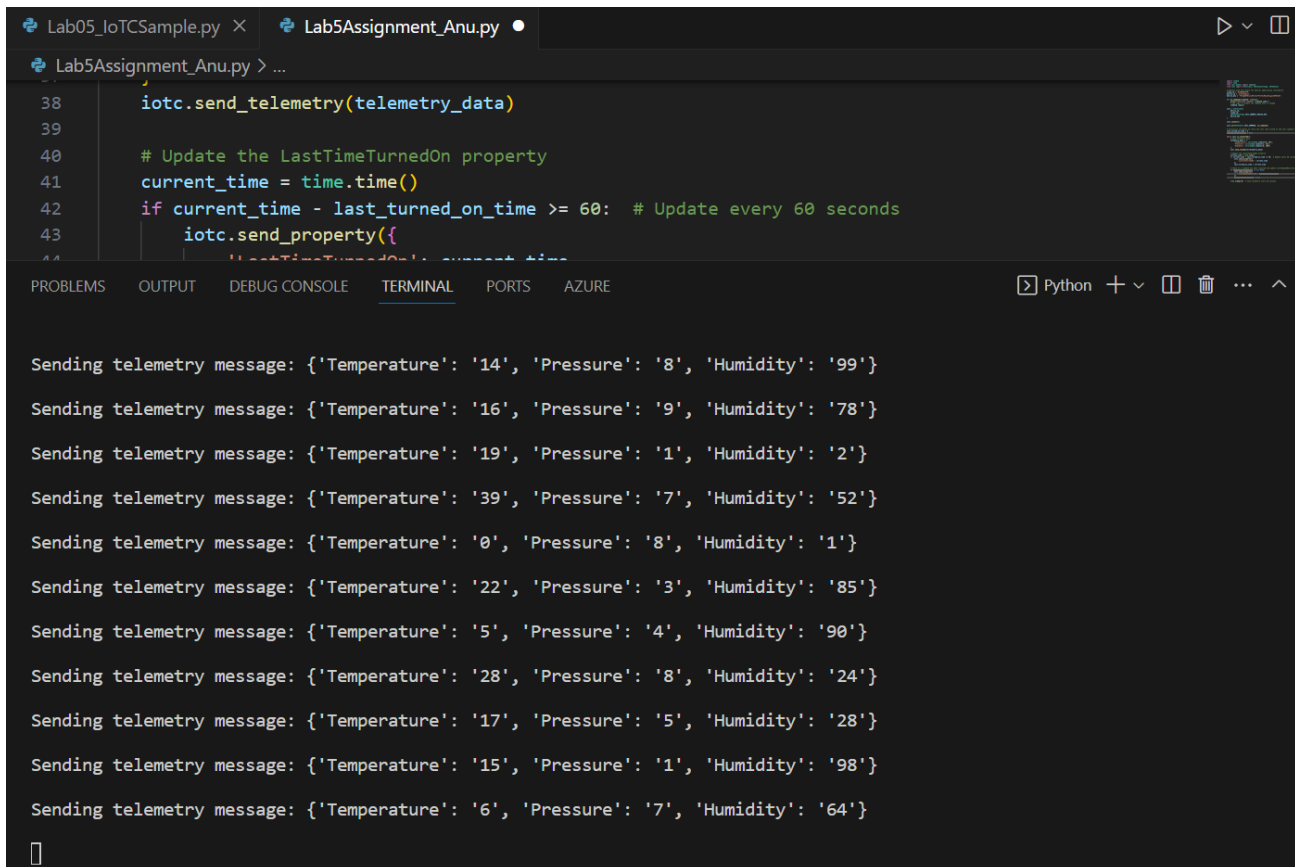
✓ Connected | 
 Last data received: 9/28/2023, 12:26:17 PM

Status: Provisioned | 
 Organization: Custom bpxg55l46m

[Raw data](#)
[Mapped aliases](#)
[Files](#)

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.



The screenshot shows a VS Code editor with two tabs: 'Lab05\_IoTCSample.py' and 'Lab5Assignment\_Anu.py'. The active tab is 'Lab5Assignment\_Anu.py', which contains a Python script. The script defines a function `iotc.send_telemetry(telemetry_data)` and uses it to send telemetry messages. The terminal output shows a series of telemetry messages being sent every 60 seconds.

```
Lab5Assignment_Anu.py > ...
38 iotc.send_telemetry(telemetry_data)
39
40 # Update the LastTimeTurnedOn property
41 current_time = time.time()
42 if current_time - last_turned_on_time >= 60: # Update every 60 seconds
43     iotc.send_property({
44         'LastTimeTurnedOn': current_time
45     })
46
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

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'}

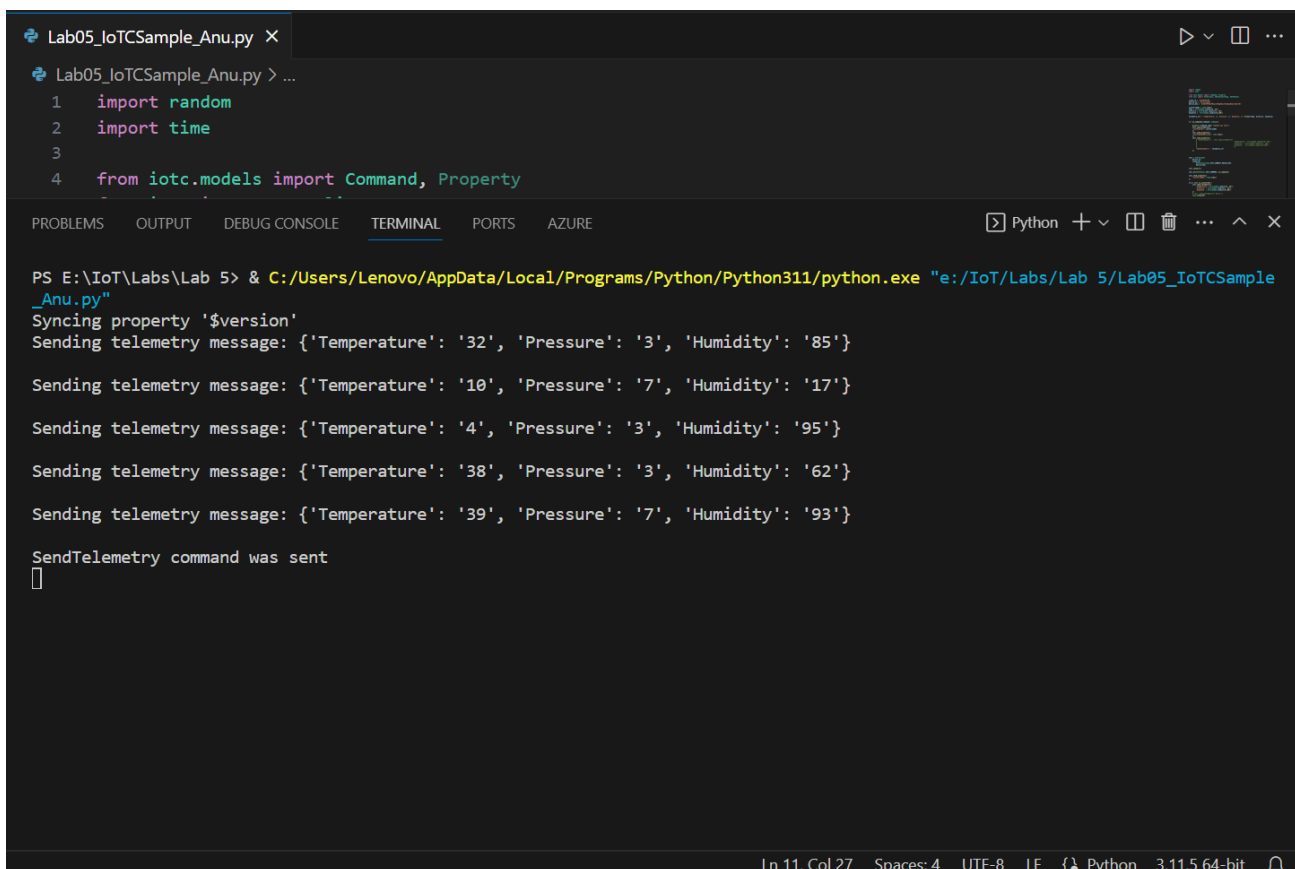
Sending telemetry message: {'Temperature': '28', 'Pressure': '8', 'Humidity': '24'}

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.



The screenshot shows a VS Code editor with two tabs: 'Lab05\_IoTCSample\_Anu.py' and 'Lab05\_IoTCSample.py'. The active tab is 'Lab05\_IoTCSample\_Anu.py', which contains a Python script. The script imports `random` and `time` modules, and imports `Command` and `Property` from `iotc.models`. The terminal output shows the execution of the script, including syncing a property and sending telemetry messages.

```
Lab05_IoTCSample_Anu.py X
Lab05_IoTCSample_Anu.py > ...
1 import random
2 import time
3
4 from iotc.models import Command, Property

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

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

Fig : Telemetry Command Recieved.

Devices > SENSOR > SensorAnu

**SensorAnu**  
 Connected | Last data received: 9/28/2023, 5:49:39 PM | Status: Provisioned | Organization: AvhiApp

Commands Raw data Mapped aliases Files

Timestamp ↓	Message type	Event creation time	Humidity	Pressure
9/28/2023, 5:35:38 PM	Telemetry		17	7
<pre> 1 { 2   "Temperature": "10", 3   "Pressure": "7", 4   "Humidity": "17", 5   "_eventtype": "Telemetry", 6   "_timestamp": "2023-09-28T22:35:38.364Z" 7 } </pre>				
9/28/2023, 5:34:57 PM	Device connected			
<pre> 1 { 2   "_eventtype": "Device connected", 3   "_timestamp": "2023-09-28T22:34:57.7994105Z" 4 } </pre>				

Fig : Telemetry command output

## Outputs from Amlan:

Amlan app

Connect Manage template Manage device

Devices > SENSOR > Sensor Amlan

**Sensor Amlan**  
 Connected | Last data received: 9/27/2023, 12:52:42 PM | Status: Provisioned | Organization: Amlan app

Command Raw data Mapped aliases Files

Timestamp ↓	Message type	Event creation time	Humidity	Pressure	Temperature	LastCommandReceived	LastPowerOn	SendData
9/27/2023, 12:52:22 PM	Telemetry			24	5	33		
<pre> 1 { 2   "Temperature": "33", 3   "Pressure": "5", 4   "Humidity": "24", 5   "_eventtype": "Telemetry", 6   "_timestamp": "2023-09-27T17:52:22.646Z" 7 } </pre>								
9/27/2023, 12:52:18 PM	Command response	9/27/2023, 12:52:18 PM						[data]"Command received"
<pre> 1 { 2   "_eventcreationtime": "2023-09-27T17:52:18.403Z", 3   "sendData": { 4     "data": "Command received", 5     "result": true 6   }, 7   "_eventtype": "Command response", 8   "_timestamp": "2023-09-27T17:52:18.483Z" 9 } </pre>								
9/27/2023, 12:52:18 PM	Command request	9/27/2023, 12:52:18 PM						[connectTimeoutSeconds]
<pre> 1 { 2   "_eventcreationtime": "2023-09-27T17:52:18.342Z", 3   "sendData": { 4     "connectTimeoutSeconds": 30, 5     "methodName": "SendData", 6     "responseTimeoutSeconds": 30 7   }, 8   "_eventtype": "Command request", 9   "_timestamp": "2023-09-27T17:52:18.446Z" 10 } </pre>								
9/27/2023, 12:52:17 PM	Property						1695637137.048449	
9/27/2023, 12:52:17 PM	Property							1695637137.057298
9/27/2023, 12:52:12 PM	Telemetry			15	4	19		

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



```

1 import random
2 import time
3
4 from iotc.models import Command, Property
5 from iotc import IoTConnectClient, IoTConnectType, IoTCEvents
6
7 scope_id = '0ne000a204f2'
8 device_id = '267ab0d9d1'
9 device_key = 'k1t+Abbkjml7xef3f5o1r3051e3zf16kqgfrvva='
10
11 LastTurnedOn=time.time()
12
13 def on_commands(command: Command):
14     print(f'({command.name}) command was sent')
15     command.reply()
16
17     iotc.send_property({
18         "LastPowerOn": LastTurnedOn
19     })
20
21 PS C:\Users\abalabartay2> & C:\Users\abalabartay2\AppData\Local\Microsoft\WindowsApps\python3.11.exe c:\Users\abalabartay2\Downloads\Lab05_IoTCSample.py
Syncing property "version"
Sending telemetry messages: {"temperature": "32", "pressure": "6", "humidity": "36"}
Sending telemetry messages: {"temperature": "18", "pressure": "18", "humidity": "15"}
Sending telemetry messages: {"temperature": "26", "pressure": "26", "humidity": "5"}
Sending telemetry messages: {"temperature": "18", "pressure": "4", "humidity": "26"}
Sending telemetry messages: {"temperature": "18", "pressure": "18", "humidity": "18"}
Sending telemetry messages: {"temperature": "2", "pressure": "6", "humidity": "3"}
Sending telemetry messages: {"temperature": "12", "pressure": "2", "humidity": "22"}
SendData command was sent
Sending telemetry messages: {"temperature": "13", "pressure": "39", "humidity": "29"}
Sending telemetry messages: {"temperature": "15", "pressure": "37", "humidity": "9"}

```

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

**Sensor Amlan**  
 Connected | Last data received 5/28/2023, 6:20:22 PM | Status: Positioned | Organization: Amlan app

Command	Raw data	Mapped alerts	File
SendData	Telemetry		

Message type	Event creation time	Humidity	Pressure	Temperature	LastCommandReceived	LastPowerOn	SendData	Unsubscribed data	Error
Telemetry	5/28/2023, 6:20:22 PM	32	24	26					

**Command history:**

- 5/28/2023, 6:19:53 PM - Command request**  
 {"temperature": "32", "pressure": "24", "humidity": "26", "eventtype": "Telemetry", "timestamp": "2023-05-28T23:20:22Z"}
- 5/28/2023, 6:19:53 PM - Command response**  
 {"timestamp": "Temperature & Pressure & Humidity OK", "eventtype": "Command response", "timestamp": "2023-05-28T23:20:22Z"}
- 5/28/2023, 6:19:53 PM - Property**  
 {"timestamp": "18954219623452", "lastCommandReceived": {"timestamp": "2023-05-28T23:20:22Z", "value": "18954219623452"}, "lastPowerOn": {"timestamp": "2023-05-28T23:20:22Z", "value": "18954219623452"}, "eventtype": "Property", "timestamp": "2023-05-28T23:20:22Z"}
- 5/28/2023, 6:19:53 PM - Command request**  
 [data]"Command received..."
- 5/28/2023, 6:19:53 PM - Command request**  
 [timestamp]"Command received..."
- 5/28/2023, 6:19:53 PM - Property**  
 {"timestamp": "18954219623452", "lastCommandReceived": {"timestamp": "2023-05-28T23:20:22Z", "value": "18954219623452"}, "lastPowerOn": {"timestamp": "2023-05-28T23:20:22Z", "value": "18954219623452"}, "eventtype": "Property", "timestamp": "2023-05-28T23:20:22Z"}

Message type	Event creation time	Humidity	Pressure	Temperature
Telemetry	5/28/2023, 6:19:53 PM	11	16	6
Telemetry	5/28/2023, 6:19:53 PM	40	6	28

Fig : Output of LastPowerOn and LastCommandRecieved Command.

Devices > SENSOR > SensorAnu

**SensorAnu**  
 Connected | Last data received: 9/28/2023, 5:49:39 PM | Status: Provisioned | Organization: AvhiApp

Commands Raw data Mapped aliases Files

Timestamp ↓	Message type	Event creation time	Humidity	Pressure
9/28/2023, 5:35:38 PM	Telemetry		17	7
<pre> 1 { 2   "Temperature": "10", 3   "Pressure": "7", 4   "Humidity": "17", 5   "_eventtype": "Telemetry", 6   "_timestamp": "2023-09-28T22:35:38.364Z" 7 } </pre>				
9/28/2023, 5:34:57 PM	Device connected			
<pre> 1 { 2   "_eventtype": "Device connected", 3   "_timestamp": "2023-09-28T22:34:57.7994105Z" 4 } </pre>				

Fig : Telemetry command output

## Outputs from Shaswati:

Amlan app

Connect

Manage template

Manage device

Search for devices

?

?

?

Devices > SENSOR > SENSOR - Shaswati

SENSOR - Shaswati

Corrected

Last data received 9/27/2023, 12:47:12 PM

Status: Provisioned

Organization: Amlan app

Command

Raw data

Mapped aliases

Files

Timestamp ↓	Message type	Event creation time	Humidity	Pressure	Temperature	LastCommandReceived	LastPowerOn	SendData
> 9/27/2023, 12:45:12 PM	Telemetry		15	23	11			
> 9/27/2023, 12:46:12 PM	Telemetry		18	21	12			
✓ 9/27/2023, 12:45:12 PM	Telemetry		14	29	21			

1 {

2 "Temperature": "21",

3 "Pressure": "29",

4 "Humidity": "14",

5 "\_eventtype": "Telemetry",

6 "\_timestamp": "2023-09-27T17:45:12.126Z"

7 }

✓ 9/27/2023, 12:46:28 PM

Command response

9/27/2023, 12:46:28 PM

[{"data":"Command received"}]

1 {

2 "\_eventcreationtime": "2023-09-27T17:46:28.880Z",

3 "sendData": {

4 "data": "Command received",

5 "result": true

6 },

7 "\_eventtype": "Command response",

8 "\_timestamp": "2023-09-27T17:46:28.571Z"

9 }

✓ 9/27/2023, 12:46:28 PM

Command request

9/27/2023, 12:46:28 PM

[{"connectTimeoutInSeconds": 30, "responseTimeoutInSeconds": 30}]

1 {

2 "\_eventcreationtime": "2023-09-27T17:46:28.576Z",

3 "sendData": {

4 "connectTimeoutInSeconds": 30,

5 "methodName": "SendData",

6 "responseTimeoutInSeconds": 30

7 },

8 "\_eventtype": "Command request",

9 "\_timestamp": "2023-09-27T17:46:28.525Z"

10 }

✓ 9/27/2023, 12:46:27 PM

Property

1001036667.8112917

Central Home

1 {

2

3

4

5

6

7

8

9

10

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

```
shaswati@shaswati: ~/Python
shaswati@shaswati:~/Python$ cd Python/
bash: cd: Python/: No such file or directory
shaswati@shaswati:~/Python$ python iotc_eg_lab5.py
/usr/lib/python3/dist-packages/requests/__init__.py:89: RequestsDependencyWarning: urllib3 (1.26.16) or chardet (3.0.4) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({}) doesn't match a supported version "
Syncing property '$version'
Sending telemetry message: {'Temperature': '24', 'Pressure': '40', 'Humidity': '11'}

Sending telemetry message: {'Temperature': '28', 'Pressure': '31', 'Humidity': '5'}

Sending telemetry message: {'Temperature': '26', 'Pressure': '34', 'Humidity': '0'}

SendData command was sent
Sending telemetry message: {'Temperature': '21', 'Pressure': '29', 'Humidity': '14'}
```

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

Timestamp	Message type	Event creation time	Humidity	Pressure	Temperature	LastCommandReceived	LastPowerOn	SendData
> 9/27/2023, 12:40:12 PM	Telemetry		15	23	11			
> 9/27/2023, 12:46:12 PM	Telemetry		18	21	12			
> 9/27/2023, 12:45:12 PM	Telemetry		14	29	21			
> 9/27/2023, 12:46:28 PM	Command response	9/27/2023, 12:44:28 PM						["data":"Command received"]
> 9/27/2023, 12:46:28 PM	Command request	9/27/2023, 12:44:28 PM						["connectTimeoutInSeconds"]
> 9/27/2023, 12:46:27 PM	Property					169583667.8112917		
> 9/27/2023, 12:46:27 PM	Property					169583673.269541		
> 9/27/2023, 12:46:11 PM	Telemetry		0	34	26			
> 9/27/2023, 12:43:11 PM	Telemetry		5	31	28			
> 9/27/2023, 12:40:11 PM	Telemetry		11	40	24			
> 9/27/2023, 12:40:10 PM	Device connected							

Fig : Output of LastPowerOn and LastCommandRecieved Command.



## 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:

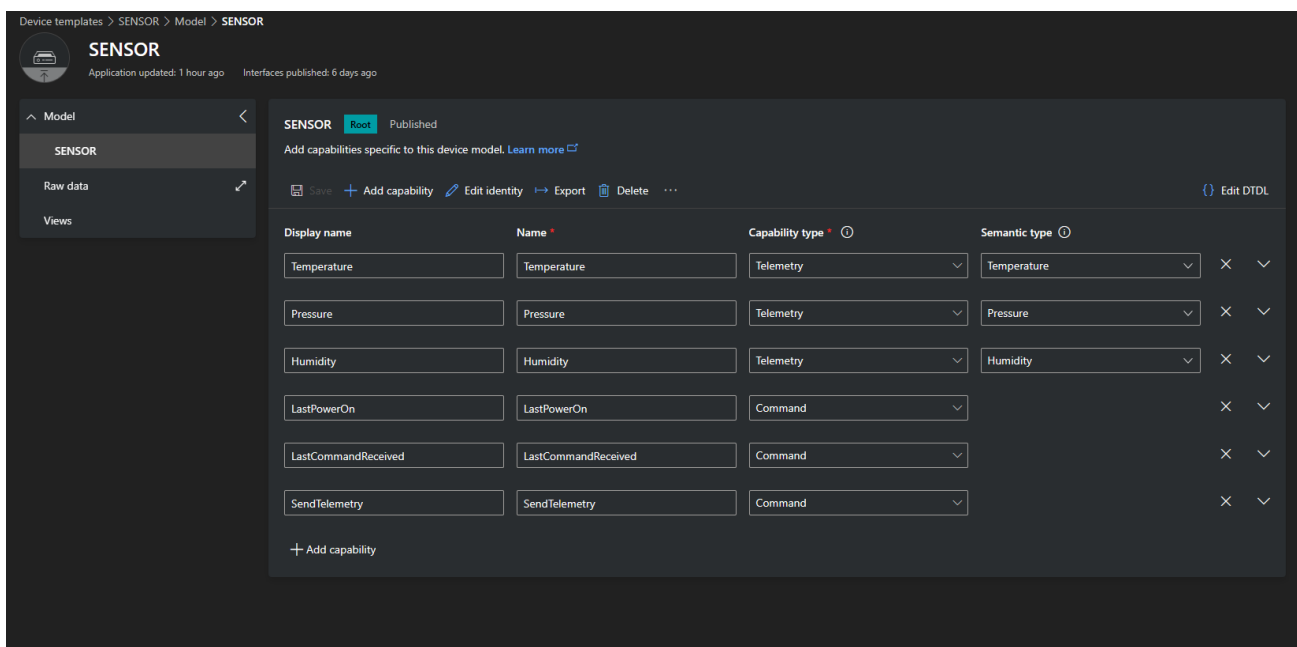


Fig 1:Sensor Template

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.

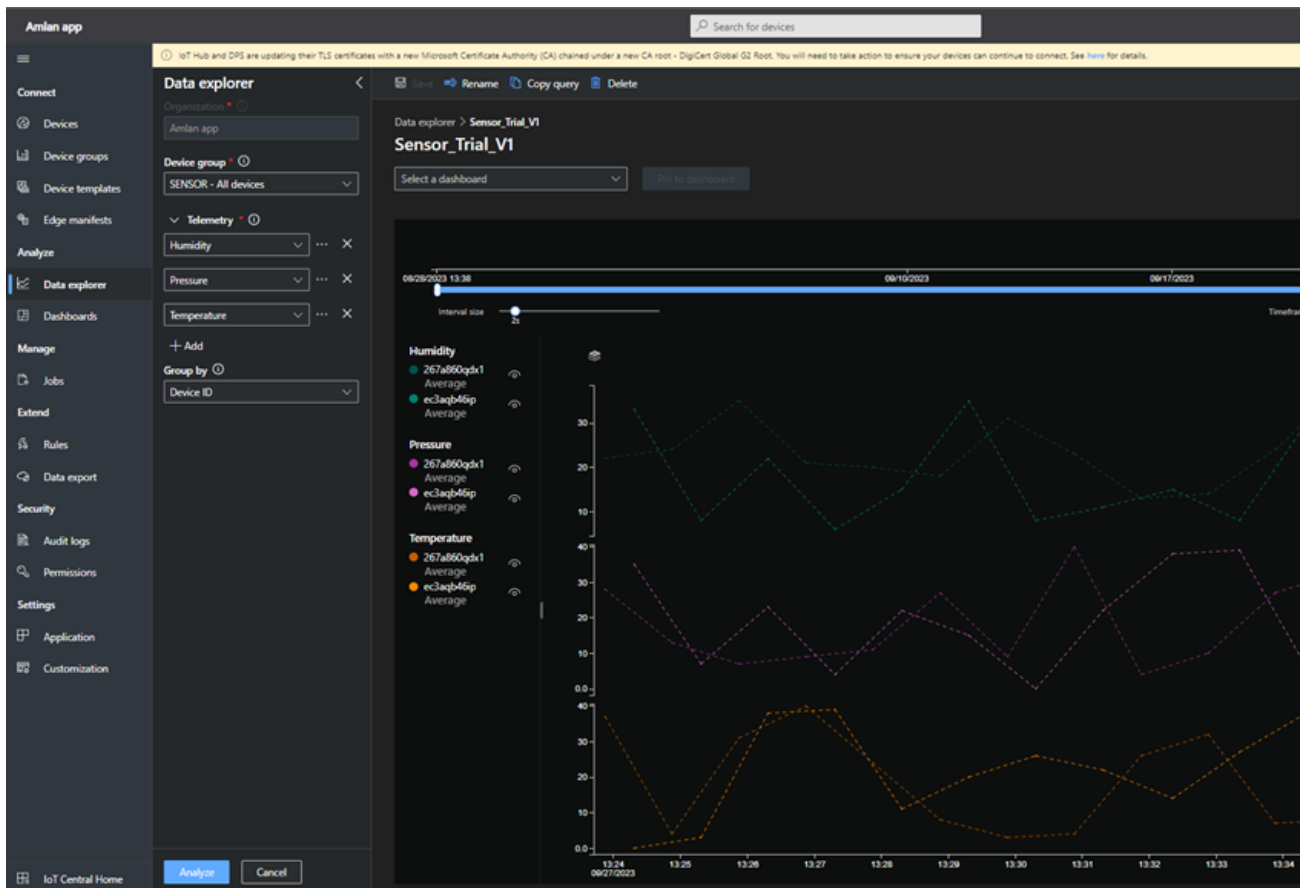


Fig : Output showing data from 2 devices in Amlan App.

# 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 IoTClient, IoTConnectType, IoTEvents

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 = IoTClient(
    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 IoTClient, IoTConnectType, IoTEvents

scope_id = '0ne00ADFCBE'
device_id = '1kivxnhp2a0'
device_key = 'bm0IBczH07aCsoNhxf+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 = IoTClient(
    device_id,
    scope_id,
    IoTConnectType.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 IoTClient, IoTConnectType, IoTEvents

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 = IoTClient(
    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](#)