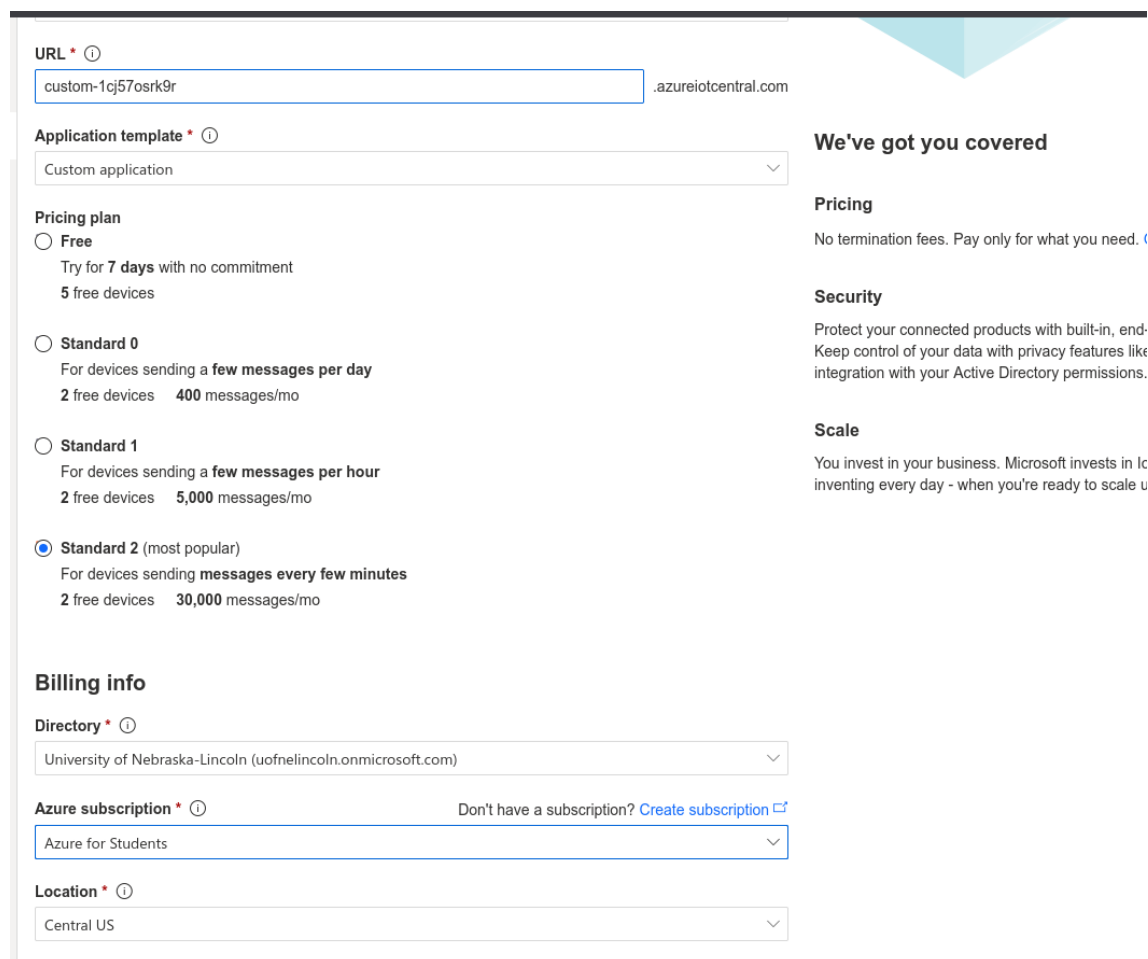# Fall 2023- CSCE 438/838: IoT-Lab 5- Connecting IoT to the Cloud with MS Azure

Create an IoT Central Application

Go to: https://apps.azureiotcentral.com/build

Now, click on **Create app** under the 'Custom app' option and configure your IoT central app as follows:

URL * ⓘ

| custom-1cj57osrk9r | .azureiotcentral.com |

Application template * ⓘ

| Custom application | ⌄ |

Pricing plan

○ **Free**
Try for **7 days** with no commitment
5 free devices

○ **Standard 0**
For devices sending a **few messages per day**
2 free devices   **400** messages/mo

○ **Standard 1**
For devices sending a **few messages per hour**
2 free devices   **5,000** messages/mo

◉ **Standard 2** (most popular)
For devices sending **messages every few minutes**
2 free devices   **30,000** messages/mo

We've got you covered

Pricing
No termination fees. Pay only for what you need. (

Security
Protect your connected products with built-in, end-
Keep control of your data with privacy features like
integration with your Active Directory permissions.

Scale
You invest in your business. Microsoft invests in lo
inventing every day - when you're ready to scale u

Billing info

Directory * ⓘ

| University of Nebraska-Lincoln (uofnelincoln.onmicrosoft.com) | ⌄ |

Azure subscription * ⓘ   Don't have a subscription? Create subscription ⊡

| Azure for Students | ⌄ |

Location * ⓘ

| Central US | ⌄ |

After creating the application you should see your application page.
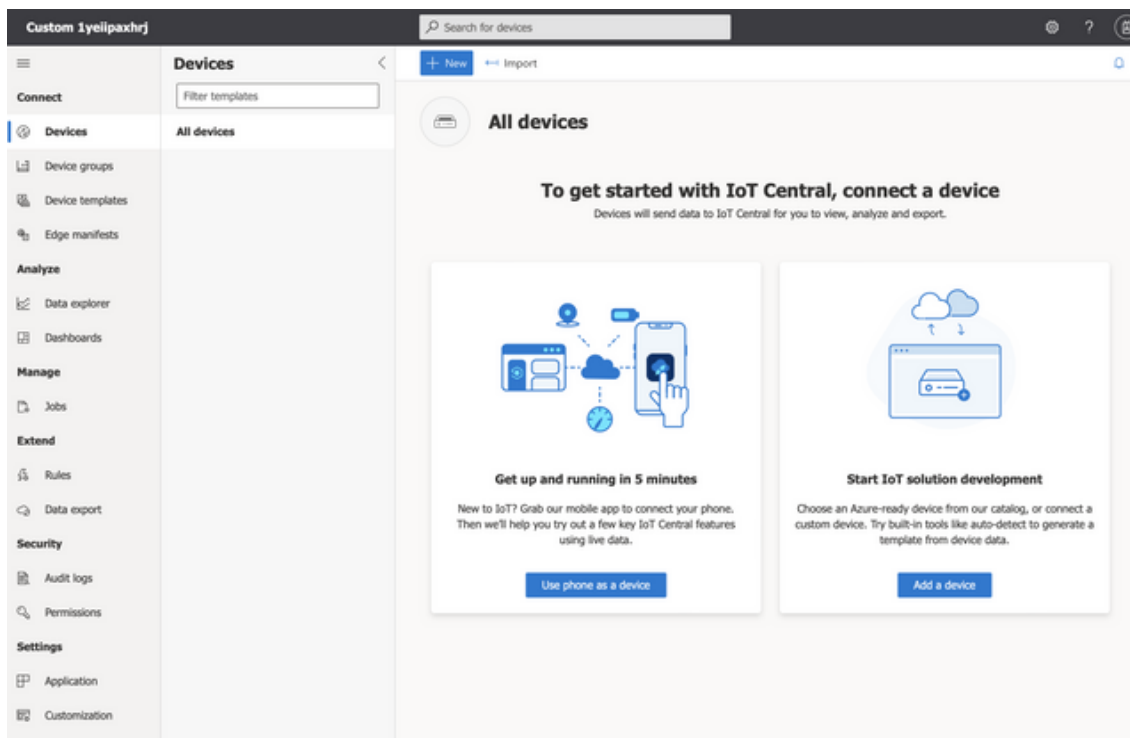
Now you can see the options of your recently created application on the left menu.

Use Your Phone as an IoT Device

DISCLAIMER: This section is mainly for fun. You do NOT have to use your personal phone and you can skip this step.

We will connect your phone to the IoT application and stream its sensor data to the IoT hub. We will also remotely turn the flash light on and off.

Click **Devices** on the left menu. Then click **New**. Now, you can create a new device for your IoTcentral app. Pick a descriptive name (e.g., **Phone 1**) and keep the other values to default and click **Create**.

Now, click the device name (e.g., **Phone 1**). You will see the device page. Click **Connect** at the top menu. Below the first three fields, you'll see **Key** and **QR code**. Click **QR Code** and follow the instructions (you will first install **IoT Plug and Play** app to your phone - iOS and Android).

Open your IoT Plug and Play app and select **Scan QR Code** and point the phone's camera at the QR code. After the connection is established, on your phone, you will see the types of telemetry information that is being sent to the IoT Central.

To see the data on the IoT central, click **Devices** and in the list of devices, click on the device name, then select Overview:
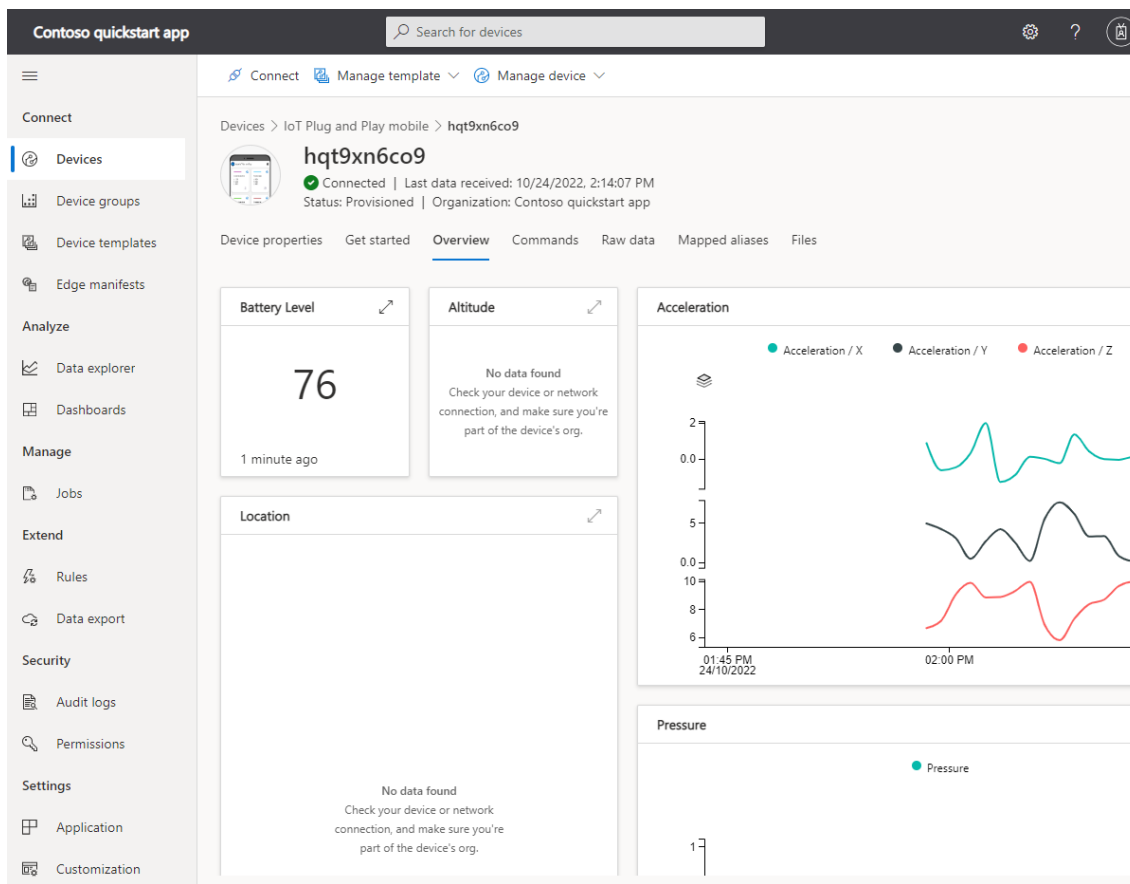
To send a command to your phone, click **Commands** and configure the **LightOn** command at the bottom with a duration of three seconds, pulse interval of five seconds, and a number of pulses of two. Click **Run** and obsorve your phone's flash light - DO NOT POINT YOUR FLASHLIGHT TOWARDS YOUR EYE!.

For good measure, go to Settings on the IoT Plug and Play app, select **Registration** and select **Clear Registration** at the bottom, delete your 'Phone 1' device from the IoTcentral, and delete the IoT Plug and Play app from your phone. [Interestingly, if you do not clear registration from the app settings, even if you delete your device from the IoTcentral and the app from your phone, your phone will connect to a provisioned IoT device on the IoTcentral if you install the app again!!]

Next, we will create a virtual IoT node that will run on your laptop and send data to the IoTcentral.
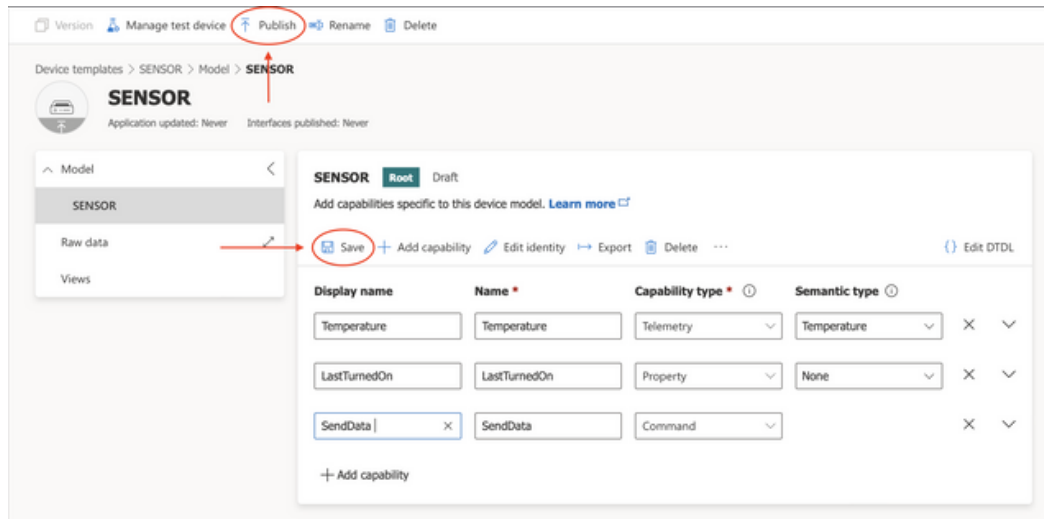
Device Template

A device template is a blueprint that defines the characteristics and behaviors of a type of device that connects to an Azure IoT Central application. It defines the

1. Telemetry that a device sends to IoT Central.
2. Properties that a device synchronizes with IoT Central.
3. Commands that the IoT Central calls on a device.

You can create a device template for a *virtual* connected Sparkfun Pro RF that has the information defined in your packet structure, including the temperature telemetry

measurement.

1. Navigate to the **Device Templates** page.
2. To create a template, start by clicking **+New** at the top.
3. Select **IoT device**, then **Next: Customize** at the bottom, enter a name (e.g., 'SENSOR'), and click **Next:Review** at the bottom, followed by **Create** to build your own template from scratch.
4. Now select **Custom model** page for your new device template.
5. Include all capablities as follows by clicking **+Add capability** when needed:

6. Click **Save** and **Publish** and **Publish** again.

Device

Click **Devices** on the left menu. Then click **New**. Now you can create a new device for your IoTcentral app. Associate it to the newly created device template.

After the deployment, you will see your new device.

Click the device name (e.g., **Sensor 1** in the example) and if you click on **Connect** at the top menu, it should look as follows:

Device connection groups   ✕

ID scope ⓘ

```
0ne0078C863
```

Device ID ⓘ

```
288s35u9dm7
```

Choose the connection type for this device. You can change this later if you need to.

Authentication type

```
Shared access signature (SAS)                                    ⌄
```

Key   QR code

Shared Access Signatures (SAS) use security tokens and keys to connect to IoT Central. Use the SAS keys from the default enrollment group shown below to register your device.

Learn more ⤢

Primary key ⓘ

Secondary key ⓘ

Close

Save the values in the following fields

- ID scope
- Device ID
- Primary key

You will use these values later to connect your device to IoT Central. Click **Close**

Depoying a Virtual Device

We are going to use the experimental **iotc** library in Python to build a simple client device. This will run in your laptops but could be deployed in any device that runs Python such as a Raspberry Pi.

1. Install Python if you don't already have it from https://www.python.org/downloads/.

2. Install `iotc` client

   ```
   pip install iotc
   ```

   If this doesn't work try

   ```
   python3 -m pip install iotc
   ```

3. Create a Python script with following code. Use the ID scope, Device ID, and Primary key values you saved above instead of YOUR SCOPE ID, YOUR DEVICE ID, and YOUR PRIMARY SAS KEY, respectively, in the code below. Then, run it:

   ```
   import random
   import time

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

   scopeId = 'YOUR SCOPE ID'
   device_id = 'YOUR DEVICE ID'
   device_key = 'YOUR PRIMARY SAS KEY'

   def on_commands(command: Command):
       print(f"{command.name} command was sent")
       command.reply()

   iotc = IoTCClient(
           device_id,
            scopeId,
                   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))
           })
           time.sleep(60)
   ```
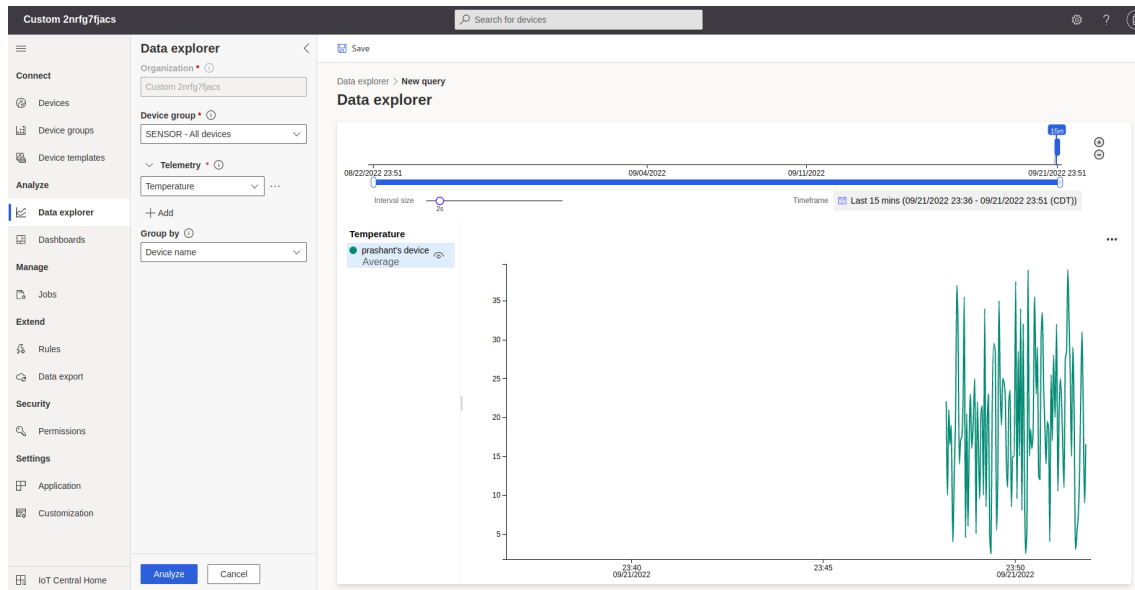
Run this (`python3 Lab05_IoTCSample.py`) for a while to send sufficient amount of data to the IoT Central. You can decrease sleep time (i.e., `time.sleep()`) to send telemetry faster.

Analytics

Go to **Data explorer** and click **+New query**. Specify all the information and group by device name and then click **Analyze**. Shorten the duration to last 15 minutes and click Analyze again.

Lab Assignment

In this lab, you will need to work together and help each other as a team to set up the IoT Cloud integration. You need to submit a lab report as a group but the report should contain the individual portions from everyone in the group.

Assignment

Requirements

In IoT Central

1. 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.
2. Create a device per team member.
3. Create **dashboard** with charts for every telemetry data type that will be received. This should include data from all the devices.

In Device

1. 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.
2. Configure the device code in each of the team member's laptop.

Results

1. Screenshots and Python code that fulfills each device requirement in this lab
2. Screenshots from Azure for the completion of the requirements.

Report format

- Development Process
  - Record your development process
  - **Acknowledge any resources that you found and helped you with your development (open-source projects/forum threads/books)**
  - Record the software/hardware bugs/pitfalls you had and your troubleshooting procedure.

- Results
  - Required results from the section above

Submission Instructions:

1. Submit your lab on Canvas on or before the deadline (Sept 29, 8:29 am)
2. Your submission should include one single pdf explaining everything that was asked in the tasks and screenshots if any
3. Your submission should also include all the code that you have worked on with proper documentation
4. Failing to follow the instructions will make you lose points

References

https://github.com/Azure/iotc-device-bridge
https://github.com/Azure/iot-central-python-client
https://learn.microsoft.com/en-us/azure/iot-central/core/concepts-telemetry-properties-commands
https://learn.microsoft.com/en-us/azure/iot-central/core/quick-deploy-iot-central