

CSCE 438 - 838 Lab 5 Connecting IoT to the Cloud with MS Azure

Deadline: By 8:29 am on October 11, 2024

In this lab, we will use the IoT Central service instead of the IoT Hub service we used for Lab 4.

IoT Central is a software-as-a-service (SaaS) offering that abstracts many details of Cloud components of an IoT application. IoT Central does indeed include an IoT Hub, but the details are hidden. On the other hand, IoT Hub is a platform-as-a-service (PaaS), which allows more flexibility in developing IoT workflows on top of the communication hub. Think of IoT Hub as an under-the-hood service, whereas IoT Central is the driver's seat. Each has its benefits depending on your IoT application.


CREATE AN IoT CENTRAL APPLICATION

Please note that we will be using a different website for Lab 5 than what was used for Lab 4.

Go to [Azure IoT Central](#) and click the 'Create app' button on the 'Custom app' option.

Build your IoT application

Featured



Custom app

Create a custom application to build a unique solution for your business using powerful tools to connect, monitor, and manage your IoT data.

[Create app](#)

[Learn more](#)

To create the app:

- A custom URL should generate automatically.
- Select the 'Standard 2' pricing plan.
- The directory should be set automatically to bill UNL.
- Select 'Azure for Students' as the subscription.
- Set location to 'Central US'.

Build > New application

New application

Custom

Answer a few quick questions and we'll get your app up and running.

About your app

Application name * ⓘ

Custom 1g4ozi51hz8

URL * ⓘ

custom-1g4ozi51hz8

.azureiotcentral.com

Application template * ⓘ

Custom application

Pricing plan

☐ 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

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

Click create after ensuring your settings are correct.

SETTING UP A DEVICE

[Optional] Use Your Phone as an IoT Device

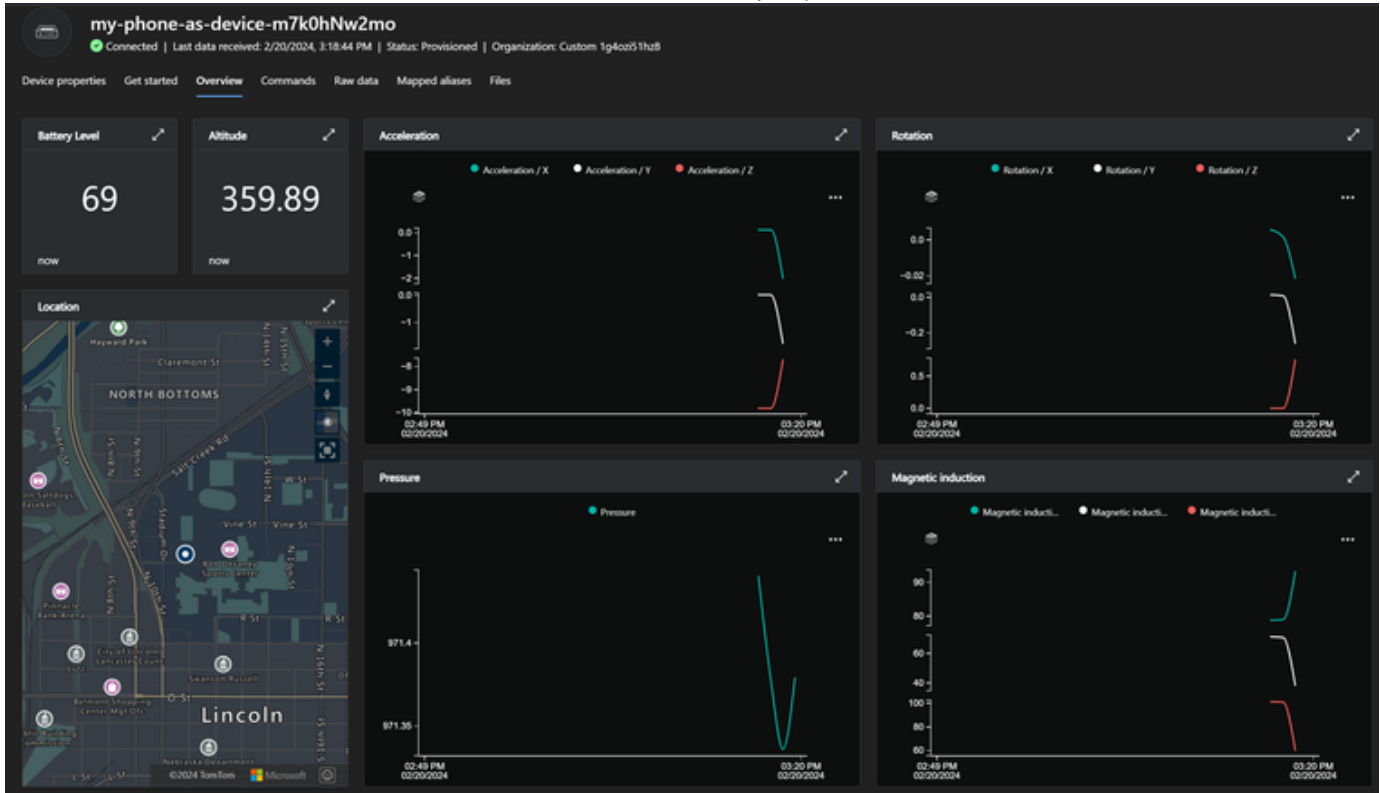
This section is mainly for fun. You do NOT have to use your personal phone and you can skip this step if you want to.

In this section, 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.

To connect your phone:

1. Click 'Devices' on the left menu.
2. Click 'Use phone as a device'.
3. Scan the QR code and install the IoT Plug and Play app on your phone.
4. Click 'Next' on the website (not app). You will scan this new QR code in the next step.
5. From within the app, click 'Scan QR code' and scan the QR code from step 4.

To see the data on IoT Central, click 'Devices' and in the list of devices, click on your phone, and selected 'Overview'.



To send a command to your phone, click 'Commands' and try turning the flashlight on and off.

- Under commands, scroll down to 'IoT Plug and Play mobile / LightOn'.
- Duration is the number of seconds the light will be on for.
- Pulses interval is the number of seconds between pulses.
- Pulses is how many times the flashlight will turn off/on (use smaller numbers to avoid very long program execution). See if you can get the flashlight to turn on/off every 5 seconds twice.

If you want to delete the device:

- Go to settings (gear on the top right) on the IoT Plug and Play app and select 'Registration'.
- Click 'Clear Registration' at the bottom.
 - If you do not clear the registration from the app settings, even if you delete your device from IoT Central and the app from your phone, your phone will connect to a provisioned IoT device on IoT Central if you install the app again.
- Delete your phone from the IoT Central website.
- Delete the IoT Plug and Play app from your phone.

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

Creating a Device Template

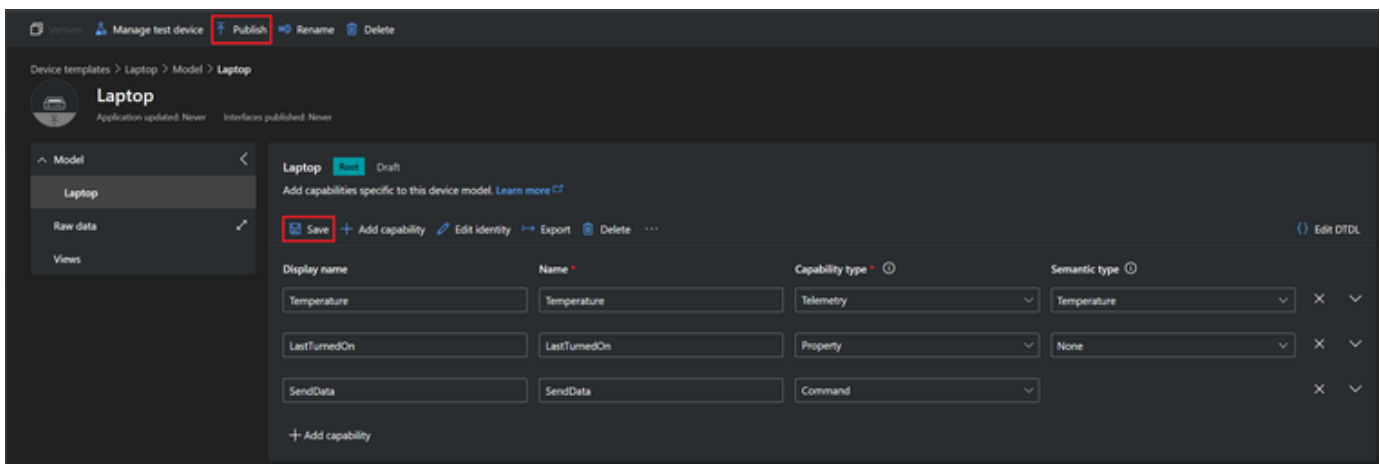
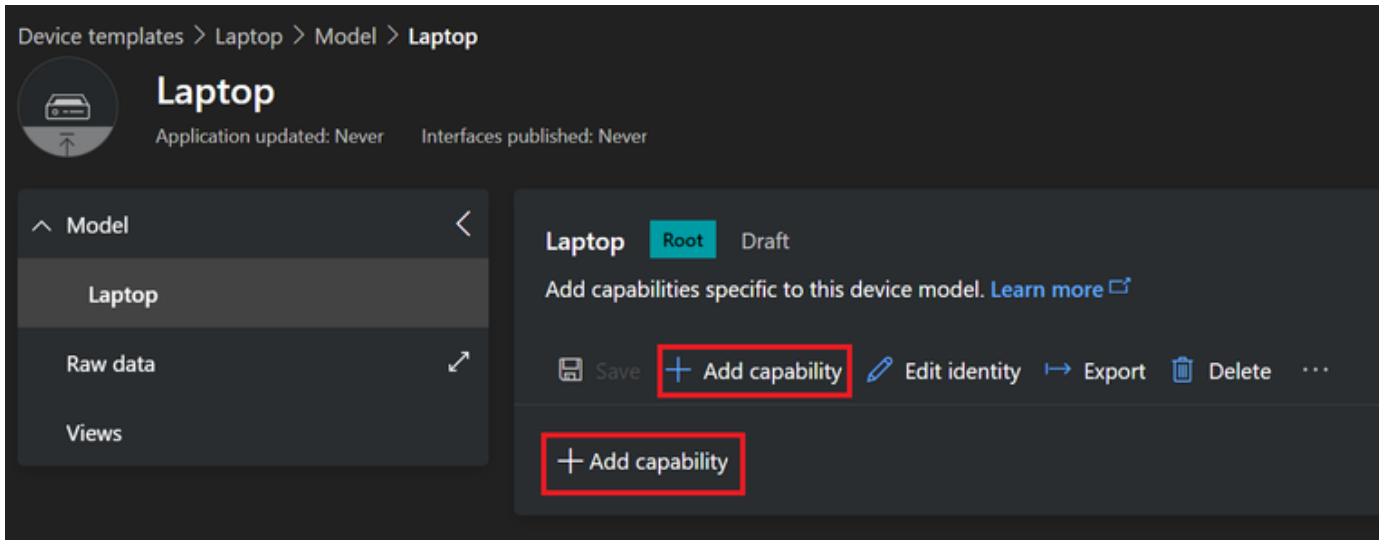
A device template is a blueprint that defines the characteristics and behaviors of a type of device that connects to an Azure IoT Central application. It defines the:

- Telemetry that a device sends to IoT Central.
- Properties that a device synchronizes with IoT Central.
- 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. To do so:

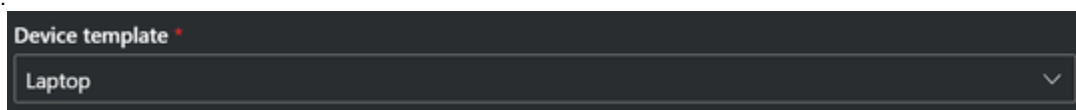
- Navigate to the 'Device Templates' page on the left sidebar (two icons below 'Devices').
- Select '+ New' in the top left.
- Select 'IoT device'.
- Select 'Next: Customize' and at the bottom of the screen.
- Give the device template a name.
- Click 'Next: Review' and then 'Create'.
- Select 'Custom model'.
- Include all the capabilities as follows by clicking either '+ Add capability' when needed.
- Click 'Save'.
- Click 'Publish' (at the very top)

Now, your device template should be published. To double-check, click 'Device templates' again on the left menu and check that your new device template has been published.

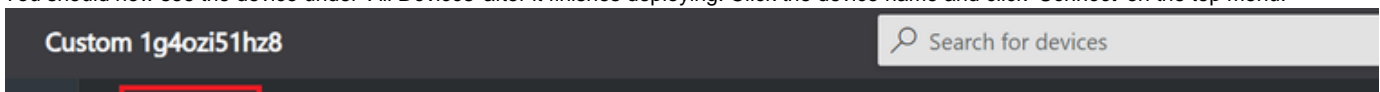


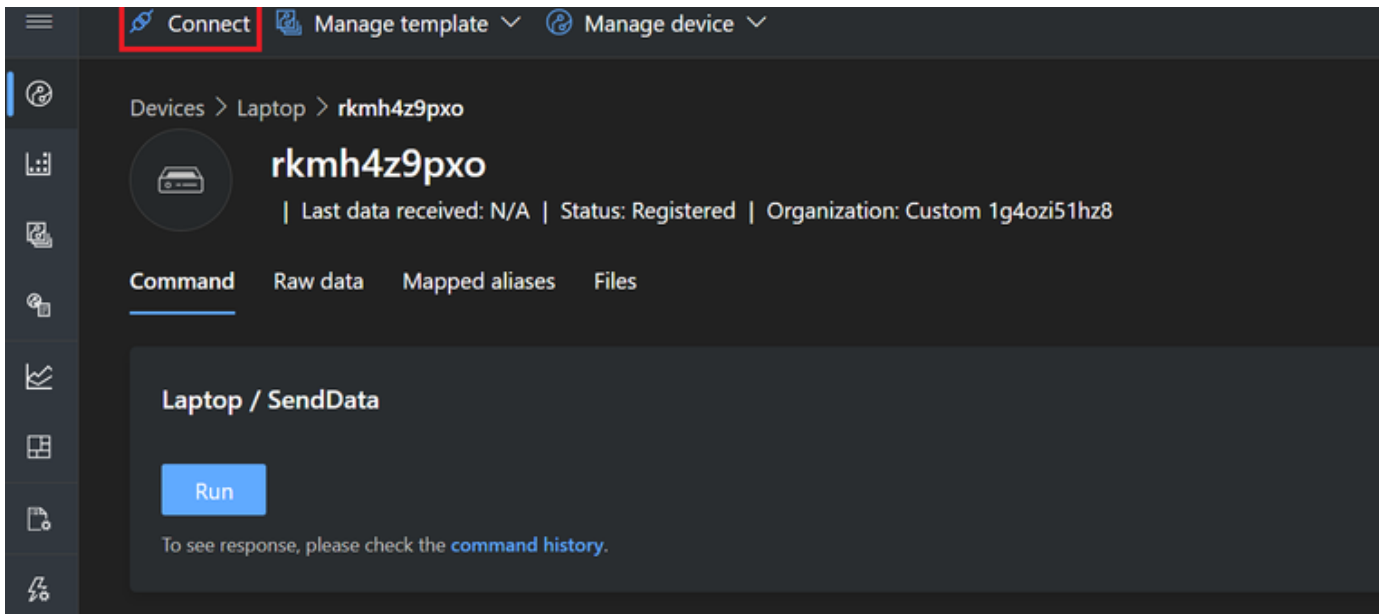
Creating a New Device from Template

Now, it's time to use your newly created template to create a device. Click 'Devices' on the left menu. Then, click 'New' and associate your device template with it.



You should now see the device under 'All Devices' after it finishes deploying. Click the device name and click 'Connect' on the top menu.





After connecting to the device, you will be presented with a few fields. You will need to save the following information to connect your device to IoT Central later:

- ID scope
 - Device ID
 - Primary key
- After you have saved these values, click 'Close'.

DEPLOYING A VIRTUAL DEVICE

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

Installing the iotc Library

1. Install Python if you don't already have it.
 - a. You can check to see if Python is installed by enter `python` into the terminal.
 - b. If needed, download Python from this link: <https://www.python.org/downloads/>.
2. Install the iotc client by entering `pip install iotc` into the terminal.
 - a. If the above command doesn't work, try `python3 -m pip install iotc`.
3. Create a Python script with the following code. Replace **YOUR SCOPE ID**, **YOUR DEVICE ID**, and **YOUR PRIMARY SAS KEY** with the values you saved earlier (keep them between single quotation marks ' ').

```
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)

```

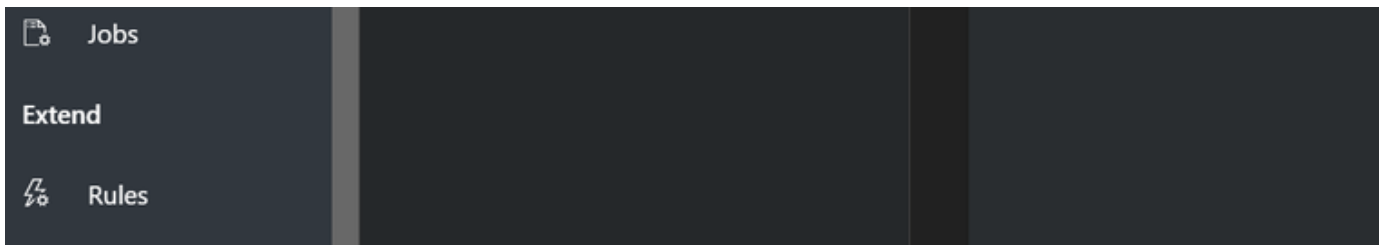
Run this (`python3 Lab05_IoT_Sample.py`) for a while to send sufficient amount of data to the IoT Central. You can decrease sleep time (`time.sleep()`) to send telemetry faster. Keep in mind the message limit for the selected payment plan.

You should be able to see this data on Azure under the 'Raw Data' tab of the device.

Analytics

On IoT Central, go to 'Data explorer' on the leftmost menu 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.

The screenshot displays the Azure IoT Central 'Data explorer' interface. On the left, a sidebar contains navigation links: 'Devices', 'Device groups', 'Device templates', 'Edge manifests', 'Analyze', 'Data explorer' (highlighted with a red underline), and 'Dashboards'. The main content area is titled 'Data explorer' and includes a search bar 'Search for devices'. Below the title, there are configuration fields: 'Organization' (Custom 1g4ozi51hz8), 'Device group' (Laptop - All devices), and 'Telemetry' (Temperature). An '+ Add' button and a 'Group by' dropdown (set to None) are also present. A 'Save' button is located in the top right corner. The right sidebar shows 'Data explorer > New query' and another 'Data explorer' header.



You should then see something like the image below. If the lines are too close together to see, note that the slider at the top of the screen can be used to limit the timeframe you're looking at. The default is the last 24 hours which makes a few minutes of temperature data look unreadable.



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.

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**, **wind speed**, 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 a **dashboard** with charts for every telemetry data type that will be received. This should include data from all the devices.

In Device,

1. Implement the virtual device code to send **temperature**, **wind speed**, and **humidity** using the `iopc` 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

- 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.
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 (Do not attach your code separately as an .ino file. Instead, copy and paste your code in the Appendix. Do not use screenshots in the Appendix.).
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-telemetryproperties-commands>

<https://learn.microsoft.com/en-us/azure/iot-central/core/quick-deploy-iot-central>