

Arduino Pro Mini (ATmega328PB/ ATmega328P)

Part E: Sending data via LoraWAN to Azure

This how-to guide / exercise will show how to send data that is collected via an Arduino ProMini to the Microsoft **Azure IoT Central** via the Heliums LoraWAN network. Azure also provides an IoT connectivity solution using their *Azure IoT Hub* resource, but this will not be covered here. *Azure IoT Central* is a software as a service (SaaS) solution that uses a model-based approach to help you to build enterprise-grade IoT solutions without requiring expertise in cloud-solution development. *Azure IoT Hub* is platform as a service (PaaS) that is a more “manual” approach that gives more control, e.g. choice of frontend, storage type, messaging format, templates, etc. Both solutions provide enterprise scalability.

More information and how-to guides can be found here.

The exercise is based on the hardware setup used in Part D “LoraWAN communication using RN2483”.

Hardware requirements:

- Arduino Pro Mini with Arduino bootloader.
TTL-USB adapter / RS232 used for UART communication with the board.
- RN2483 breakout board with 868 MHz antenna
- A sensor of some sort (temperature, humidity, or similar).
- Jumper wires (Dupont wires), breadboard etc. The usual stuff.

Software/document/other requirements:

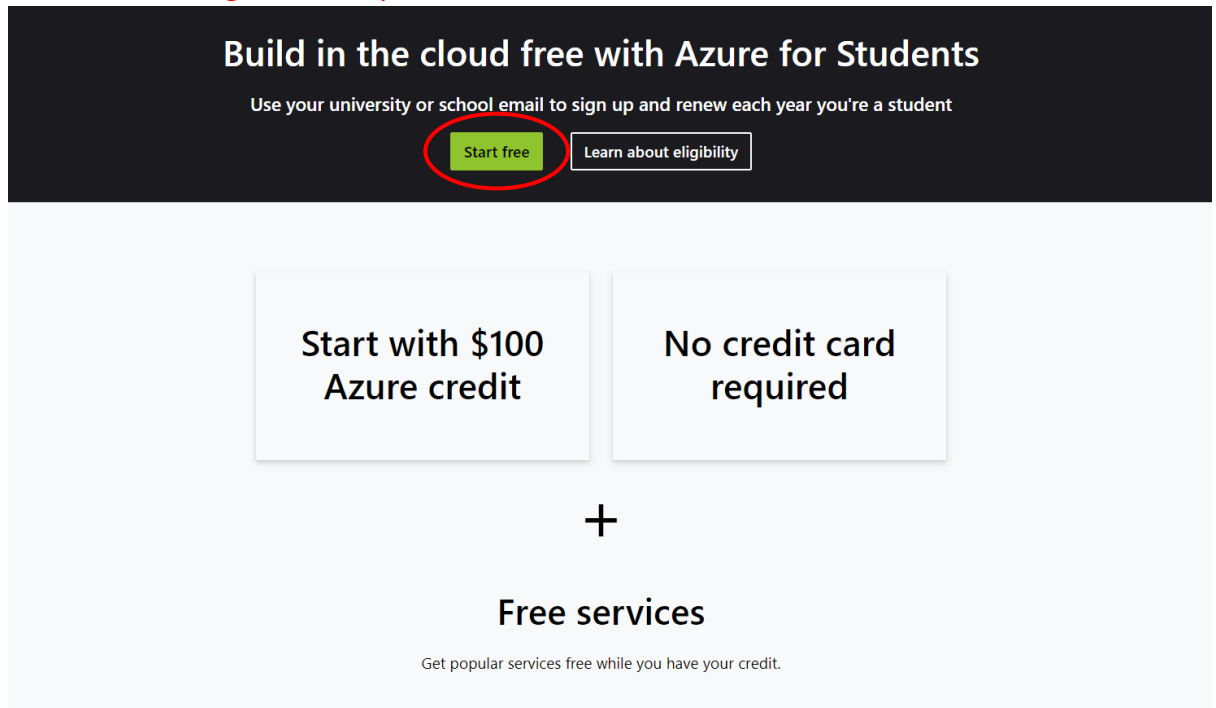
- Account on Azure: <https://azure.microsoft.com/en-us/> (guide will follow)
- Arduino IDE <https://www.arduino.cc/en/software>
- Account on Helium LoraWAN network (<https://console.helium.com>)
- LoraWAN network coverage. See coverage map here: <https://explorer.helium.com>

The **Qs**: Throughout the guide/exercise you will find questions (Q1, Q2..). Collect the answer in a document and submit this as a PDF on the DTU Learn assignment when done.

This exercise will be bundled with previous week's exercise. Deadline is at Oct 4th.

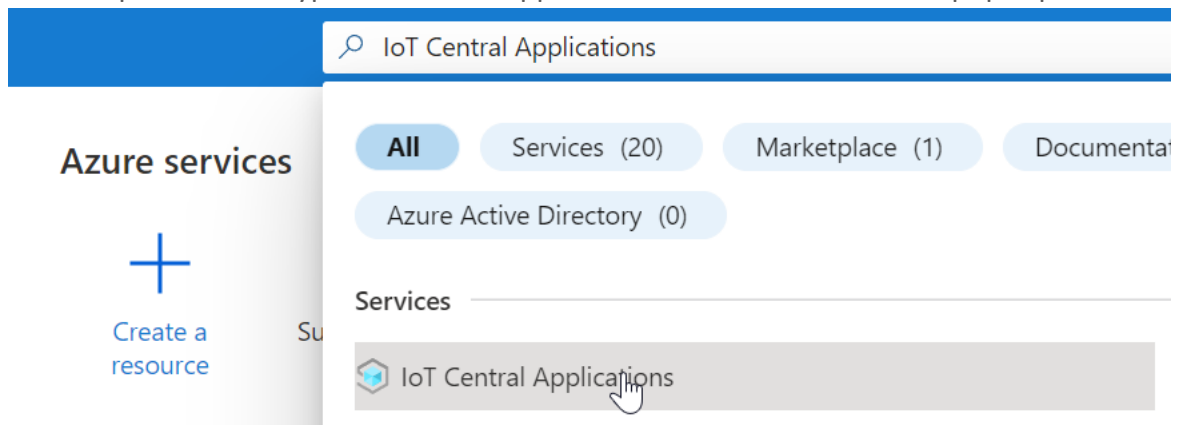
Creating Microsoft Azure Account

1. Go to <https://azure.microsoft.com/en-us/free/students/> where you can create a free student account using your DTU student credentials. **Important: Click the green button, do not sign in directly.**



Once logged in you will see the Azure Education portal page. Here you can discover various free services. You can also click on Home (upper left corner) to get an overview of Azure services.

2. In the top search bar type "iot central applications" and select it when it pops up:



3. Now click "Create iot central application" to create the application that we will use:



No IoT Central applications to display

Try changing or clearing your filters.

Create IoT Central application

4. In the *Basic* information tab create a new *Resource group* and type self-chosen names for *Resource name*, *application URL*. As *Template*, choose *Custom template*, choose a *Region* close to you, and select *Standard 2* as pricing plan. See example in the figure below.

When done, click on *Review + Create*.

Validation will take a moment, and when done click *Create*. Now the application is being deployed – this will also take a moment.

IoT Central Application ...

Basics Tags Review + create

Create an IoT Central application with an application template. IoT Central is an IoT app platform that allows you to rapidly build enterprise-grade IoT solutions on a secure, reliable and scalable infrastructure. [Learn more](#)

Project details

Select the subscription to manage the deployed IoT Central resource and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Azure til studerende
Resource group *	(New) 34365_E22

[Create new](#)

Instance details

Resource name *	34365-22
Application URL *	34365-e22
Template *	Custom application
Region *	North Europe
Pricing plan *	Standard 2

azureiotcentral.com

5. When the deployment is done, click on *Go to resource*:

We'd love your feedback! →

✓ Your deployment is complete

Deployment name: Microsoft.IoTCentral-20220831145734 Start time: 8/31/2022, 3:07:25 PM
Subscription: [Azure til studerende](#) Correlation ID: 1c84719b-f737-40c7-883b-c47eb069d116
Resource group: [34365_E22](#)

Deployment details

Next steps

[Go to resource](#)

Go to resource

Setting up the device template


Before moving on we need a template for the device. The template will eventually be structured according to the type of data we are expecting from our device.

6. From the Overview page you arrived at after clicking *Go to resource* you can find your IoT Central Application URL. Click on this link.

IoT Central Application U... : <https://34365-e22.azureiotcentral.com>

7. At this point you may be prompted / suggested to connect your phone to the IoT Central. Consider doing this – if not, it can be done later. It is not necessary for this exercise to connect your smartphone. What this does is simply to add the phone as a device sending telemetry data (Accelerometer, Gyroscope, Geolocation etc.) to the Azure cloud.

Download the free IoT Plug and Play app




Download on the App Store GET IT ON Google Play

Install and open the app on your phone, then click **Next**.

Next

Connect phone to IoT Central




From the IoT Plug and Play app, scan this QR code to open a live, secure datastream from your phone.

We haven't seen your device yet! Use the app to connect it.

Back

Tour IoT Central right now, with live data

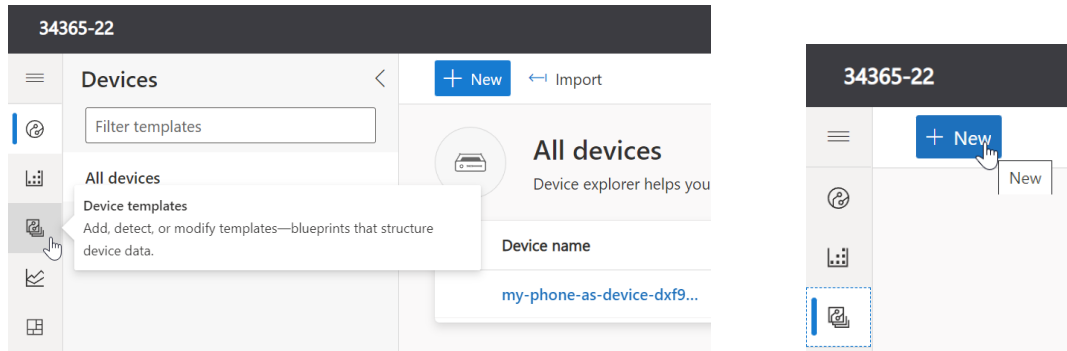


Explore top features using real data from your phone with our IoT Plug and Play app

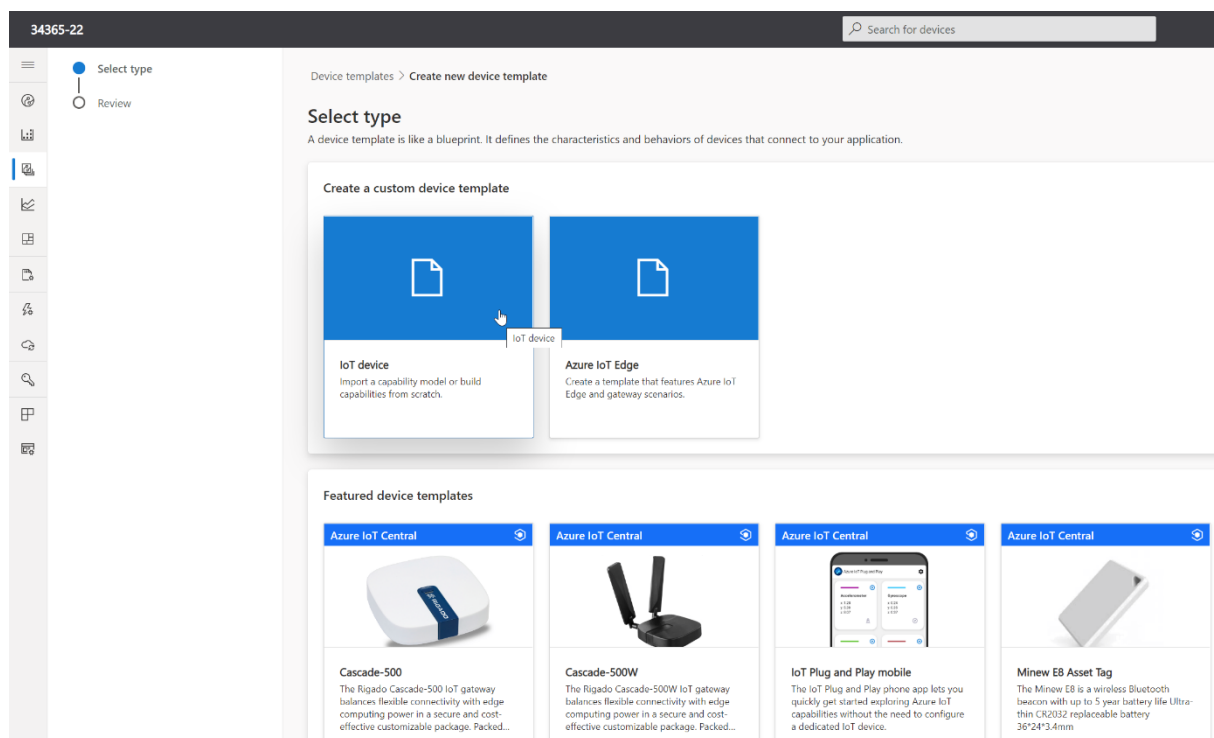
Your data is only available within your IoT Central application and is not used for any other purposes.

Cancel **Next**

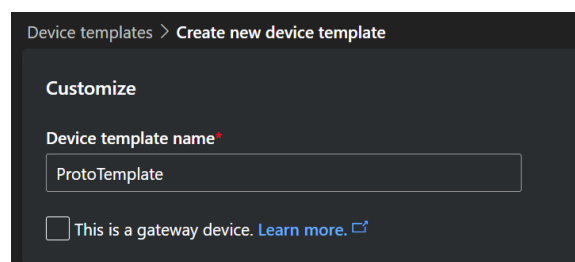
8. Next, choose *Device Templates* from the left menu and click *+ New* to create a new template. Depending on your theme it may look slightly different:



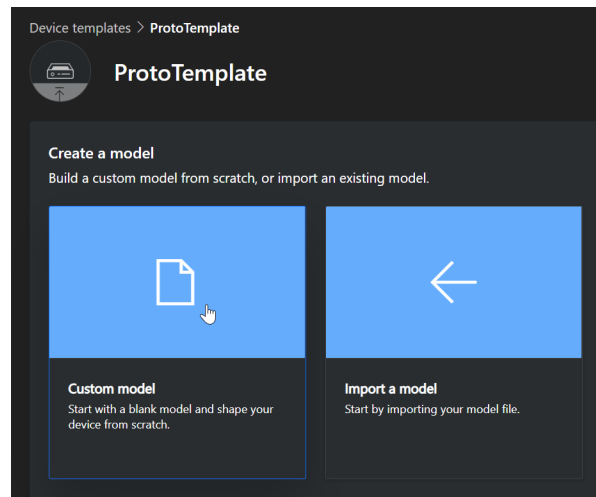
9. Here you will see a number of readymade templates for existing and commercial devices. Select *IoT device* to create a custom device template. Click [Next: Customize](#) at the button at the bottom of the page after selecting *IoT device*



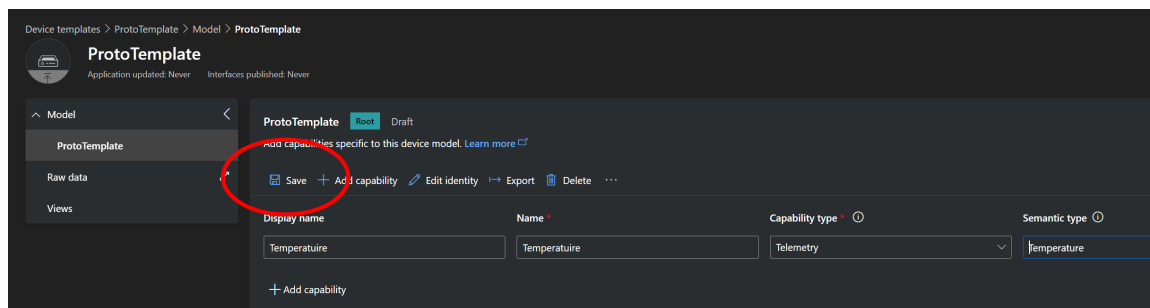
10. Give the device template a name and click [Next: Review](#) to review the information. And finally [Create](#) to create the template.



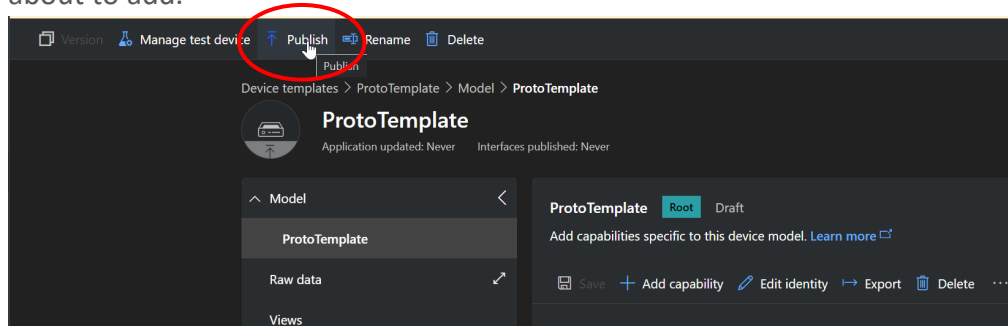
11. In the next screen select *Custom model* to be able to customize the so-called *Capabilities*.



12. As we will have as a minimum temperature this as a capability, as shown below. When done click on *Save* and then *Views*



13. When done click on *Publish* to make the Template available for the Device we are about to add.



Now the Azure account is created, and a basic template is ready for the device data when that will arrive. Later we will return to the Template and configure this further. Now it is time to integrate Azure with Helium.

Setting up a Azure integration on Helium

14. Start with the setup used in the last exercise, Part D, where a temperature measurement is sent to Helium using the LoraWAN RN2483 module. You don't need to start transmitting yet.
15. As we now need to send data to Azure we need to create a new integration so head over to Integrations in the Helium Console. Click on Add New Integration and choose Azure IoT Central (not Azure IoT Hub). Click on Add Integration.

The screenshot shows the Helium console interface for setting up a new integration. It is divided into two main sections: 'STEP 1 - CHOOSE AN INTEGRATION TYPE' and 'STEP 2 - ENDPOINT DETAILS'.

STEP 1 - CHOOSE AN INTEGRATION TYPE

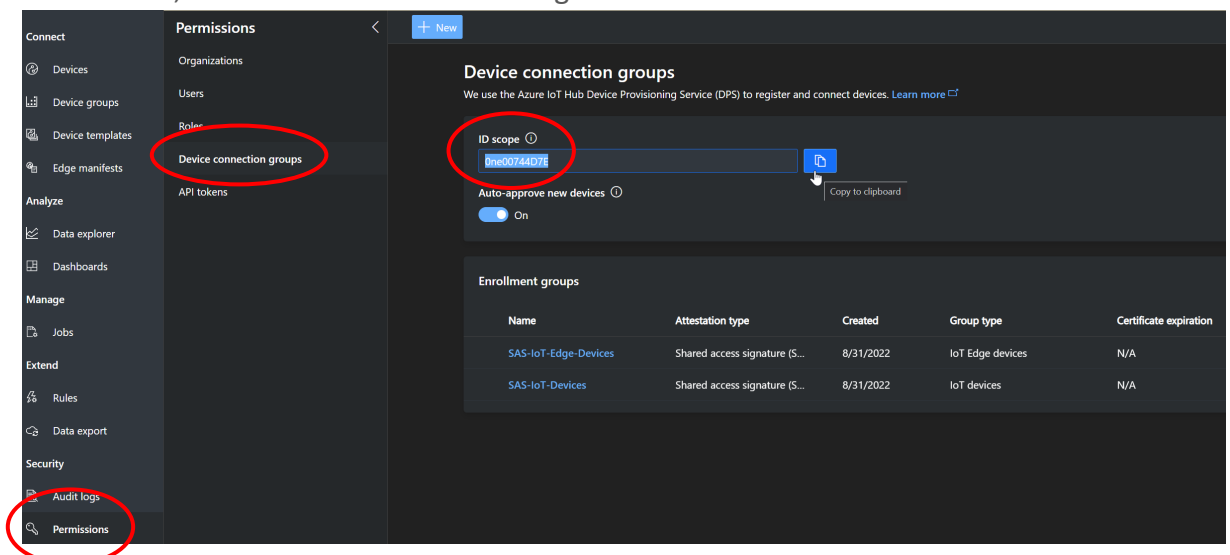
Under this heading, there is a card for 'Azure IoT Central'. The card includes the Azure IoT Central logo, a description: 'Azure IoT Central is highly secure, scales with your business as it grows, ensures your investments are repeatable, and integrates with your existing business apps', and a link: 'Tell me more about setting up this integration.' Below the card is a 'Change' button.

STEP 2 - ENDPOINT DETAILS

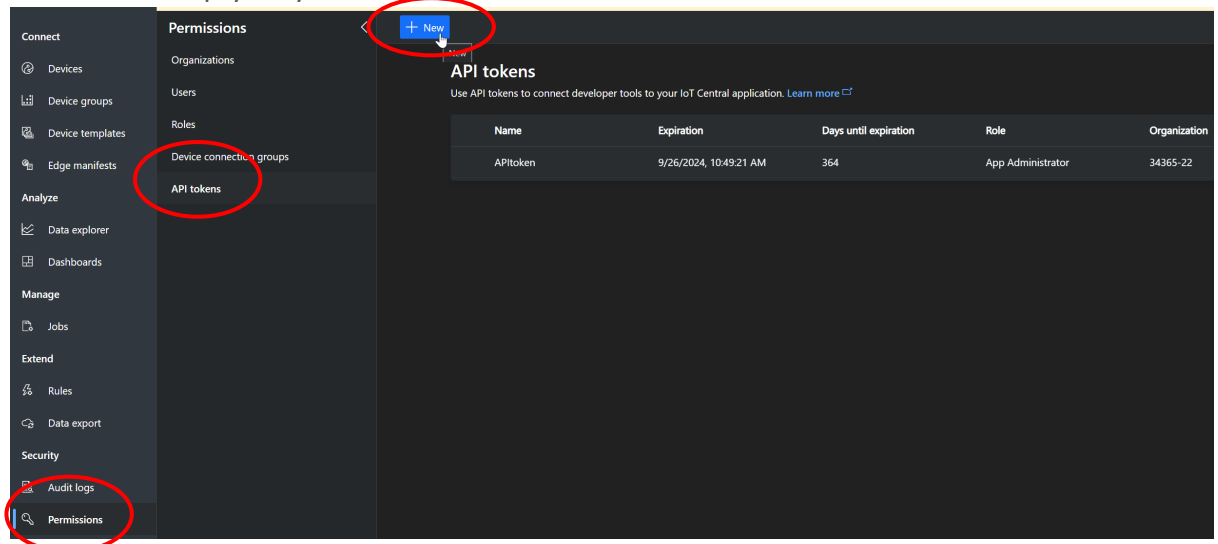
This section contains three input fields for configuration:

- API Token
- Scope ID
- App Name

16. In order to connect Helium to Azure IoT Central, a bit of information is needed. The Scope ID is a unique value that Azure uses for provisioning, basically a part of the information needed to authenticate devices. This can be found under *Permissions->Device connection groups* in the Azure IoT Central console. Get this value and cope it into Helium, in the Azure IoT Central Integration details.



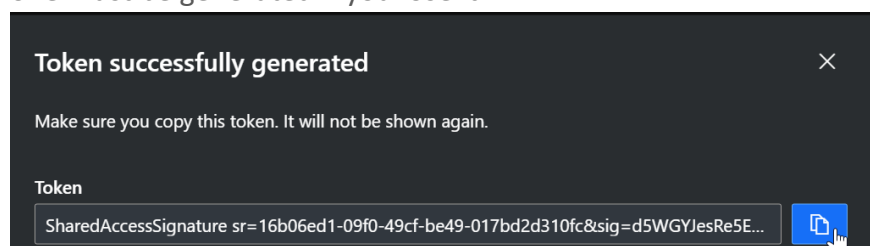
17. The API Token can be retrieved also from the Permissions menu, under API Tokens. This will be empty on your screen so select to create a New.



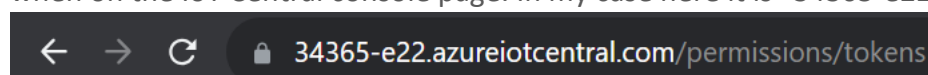
18. Give it a name, select your Organization and generate a new token.

The 'Generate token' dialog box is shown. It contains three input fields: 'Token name' (filled with 'GiveltaName'), 'Organization' (dropdown menu set to '34365-22'), and 'Role' (dropdown menu set to 'App Administrator'). At the bottom right, there are two buttons: 'Generate' (highlighted) and 'Cancel'.

19. Once you click Generate you will get the long token, which can be copied into the correct field in Helium. Notice that you will only be shown this token once so a new one must be generated if you lose it.



20. Finally, the App name can be retrieved from your IoT Central url. I.e. look at your URL when on the IoT Central console page. In my case here it is "34365-e22"



21. With these three pieces of information entered, plus a name for the Integration click Add Integration.

Azure IoT Central

INTEGRATION DETAILS

Azure IoT Central0/50Update

Type: Azure IoT Central

ID: 79979e4c-1fae-4318-a2d2-620e1cb54020

Receive Device Joins: ☐

Piped Devices: 0

UPDATE YOUR CONNECTION DETAILS

API Token

SharedAccessSignature sr=16b06ed1-09f0-49cf-be49-017bd2d310fc&sig=N8%2BuRWEJYW0ibwrM09yyxwx0

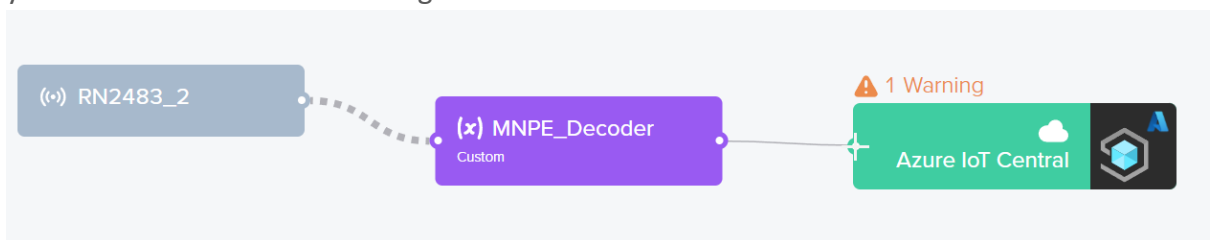
Scope ID

One00744D7E

App Name

34365-e22

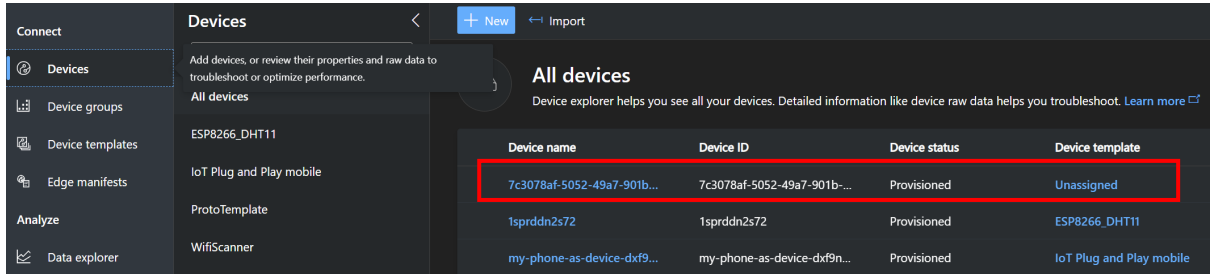
22. With this head over to Flows in the Helium console. Create a new flow that includes your new Azure IoT Central integration:



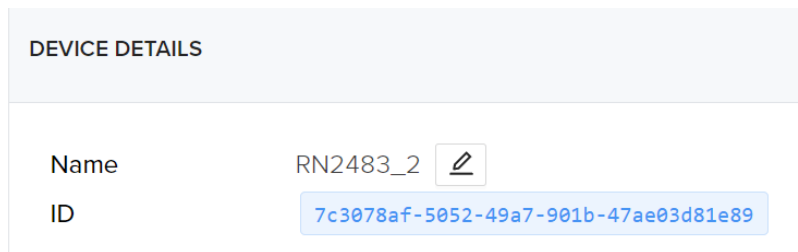
Now the setup ready to start sending some data.

Sending data to IoT Central

23. Now it is time to use the setup from the last exercise, Part D, where a temperature measurement it sent to Helium. Start sending data and check in Helium that the data is received here.
24. Once confirmed that data is sent to Helium, head over to IoT Central and click on Devices. Once the integration (Helium-Azure) is complete, which may take a few minutes, you should see a new device appear in the list:

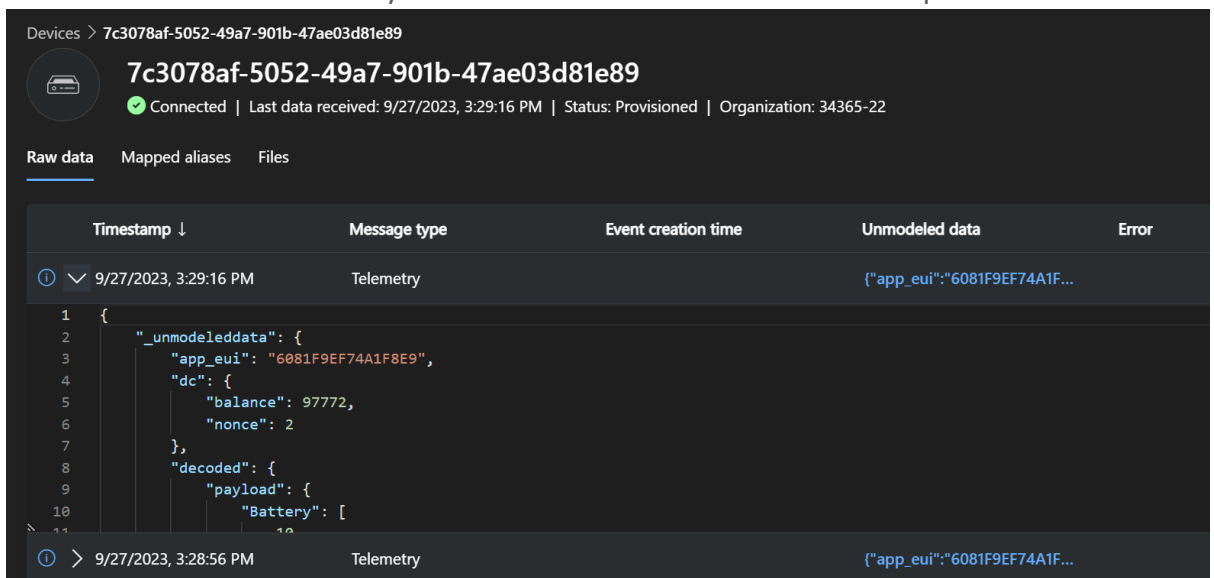


The device will be auto-created and given a name according to the information sent from Helium. In this case the device is named “7c3078af-5052-49a7-901b-47ae03d81e89”. This name is identical to the ID created in Helium:



You can rename the device or leave it as is. Notice that there is no Device Template assigned at this time.

25. Click on Raw data to see if any data has arrived and click on the > to expand:

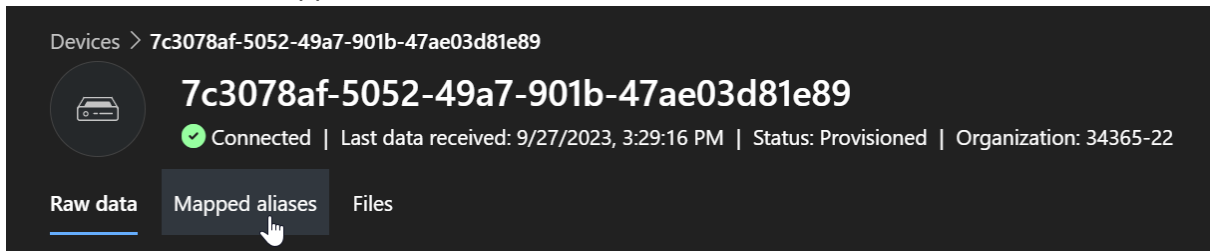


The data is structured as a JSON object, similar to what can be seen on the Helium console. If all is well, you should see the Temperature data somewhere within the data.

Q1: Take a screenshot of your data coming in, similar to the one above.

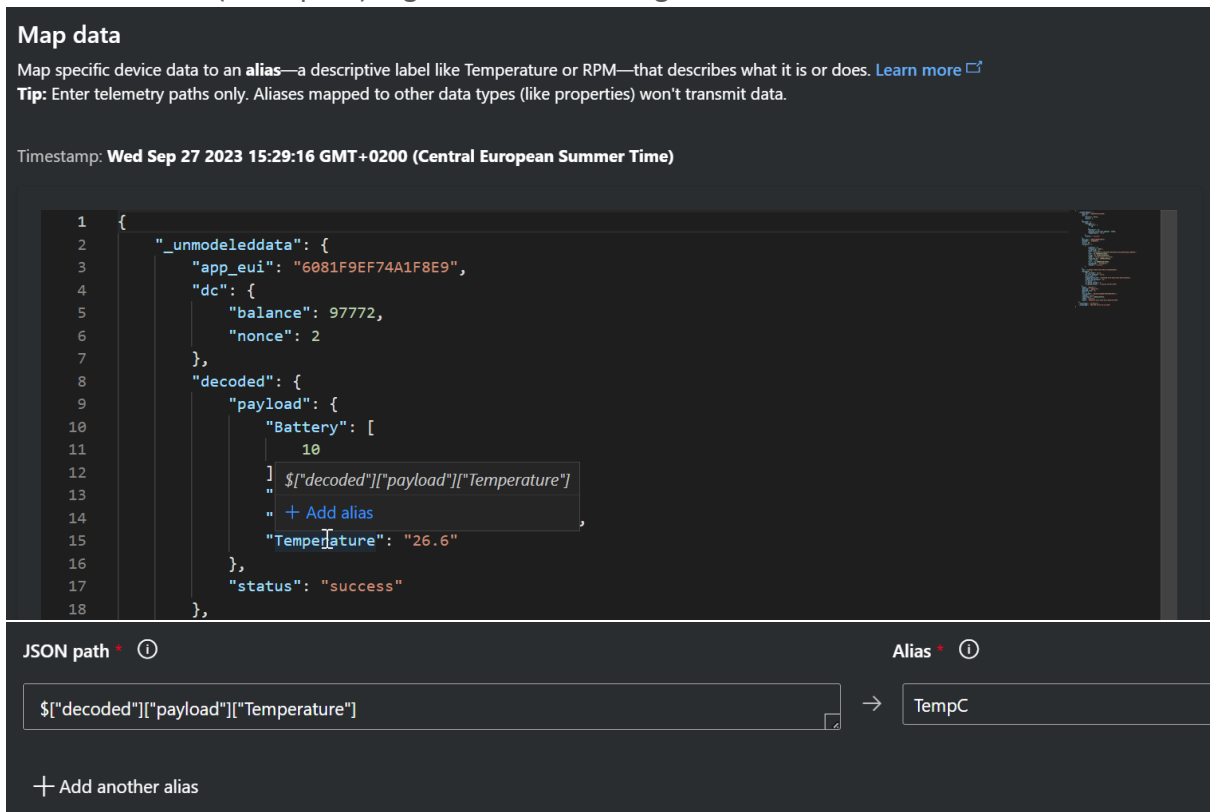
Extracting and visualizing data in IoT Central

26. In order to extract the relevant data it is necessary to map the data into JSON keys (items) that are readable to IoT Central. The JSON structure created by Helium is not directly readable to IoT Central and to make it readable, the feature Mapped aliases is needed. Click on Mapped aliases



Click on *Create an alias* to create a new one.

First, it is necessary to make IoT Central rewrite, or extract, the relevant information into new JSON keys. In the JSON editor that appears, locate the key(s) that you want and do a mouse-over to reveal the “Add alias” feature. Click on Add alias and an extraction code (JSON path) is generated on the right side¹.



When that is done, give it an Alias name:

¹ More information (Although slightly outdated) available here:
<https://learn.microsoft.com/en-us/azure/iot-central/core/howto-map-data>

One can continue to generate more if needed.

JSON path	Alias
<code>["\$decoded"]["payload"]["Battery"]</code>	BattLevel
<code>["\$decoded"]["payload"]["Humidity"]</code>	Hum

Once done, click on Save

27. They will now all appear in a list:

JSON path	Alias
<code>["\$decoded"]["payload"]["Temperature"]</code>	TempC
<code>["\$decoded"]["payload"]["Battery"]</code>	BattLevel
<code>["\$decoded"]["payload"]["Humidity"]</code>	Hum

Q2: Post a screenshot of your data aliases definitions. Like above.

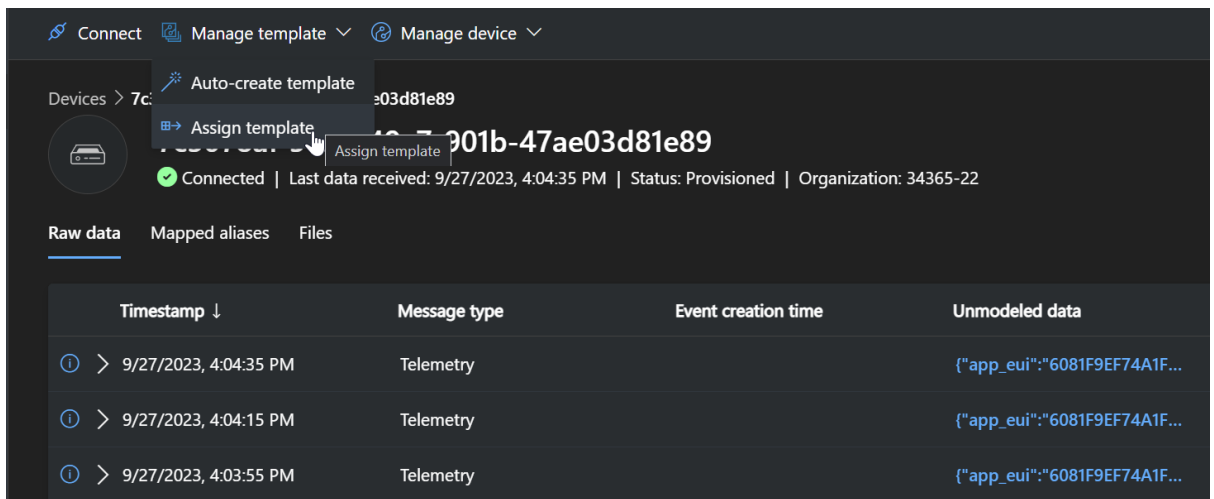
28. Assuming that the device is still running and transmitting LoraWAN packages you should be able to see the change in a few minutes. Go to Raw data, expand the last data listing and scroll down until you see your new aliases in the JSON data. This is called “mapped data”

Timestamp ↓	Message type	Event creation time	Unmodeled data
9/27/2023, 4:01:54 PM	Telemetry		<code>{"app_eui": "6081F9EF74A1F..."}</code>

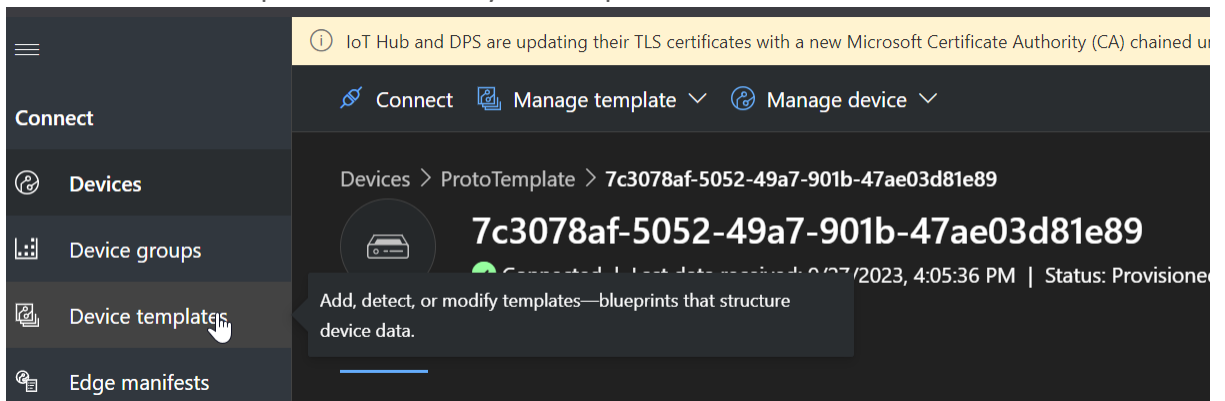
```
56     "type": "uplink",
57     "uuid": "d2746042-30f9-49d4-9f44-167992f43004",
58     "TempC": "26.6",
59     "BattLevel": [
60       10
61     ],
62     "Hum": 91
63   },
64   "_eventtype": "Telemetry",
65   "_timestamp": "2023-09-27T14:01:54.857Z"
66 }
```

Q3: Locate your mapped data and post a screenshot. Like above.

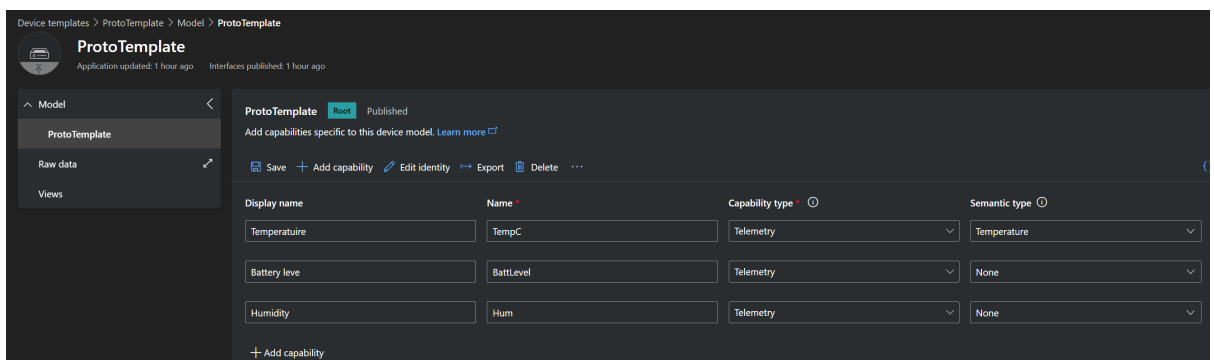
29. These newly generated aliases are readable and “parse-able” to IoT Central so that Capabilities in the Template can be assigned to these aliases.
30. Click on Manage template and choose to Assign template. Choose the Template that was created earlier and Assign it.



31. Click on Device templates and select your template

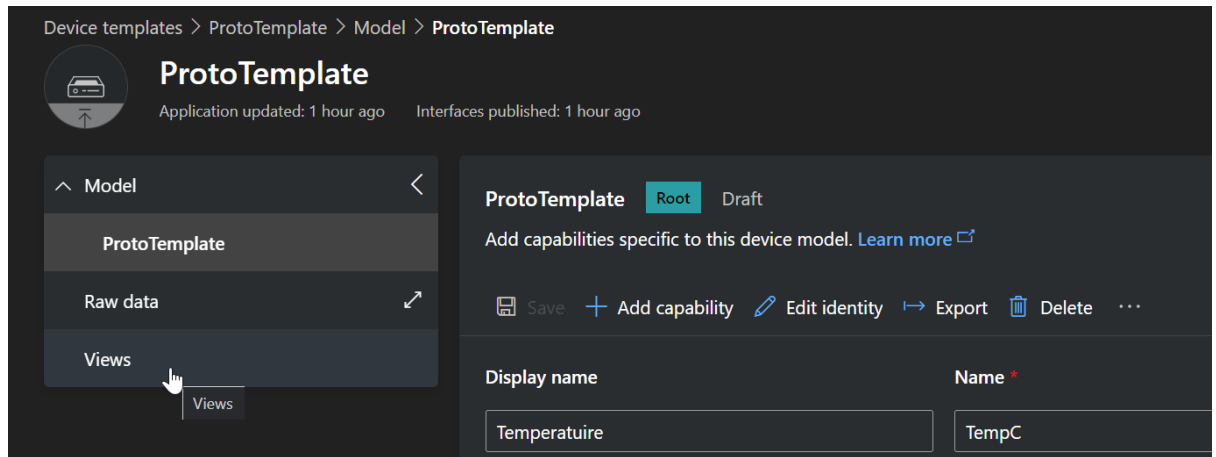


32. Add capabilities according to the alias names (Name) that you selected earlier. For example like this (below). Notice that the Name has to match the JSON key name from before.

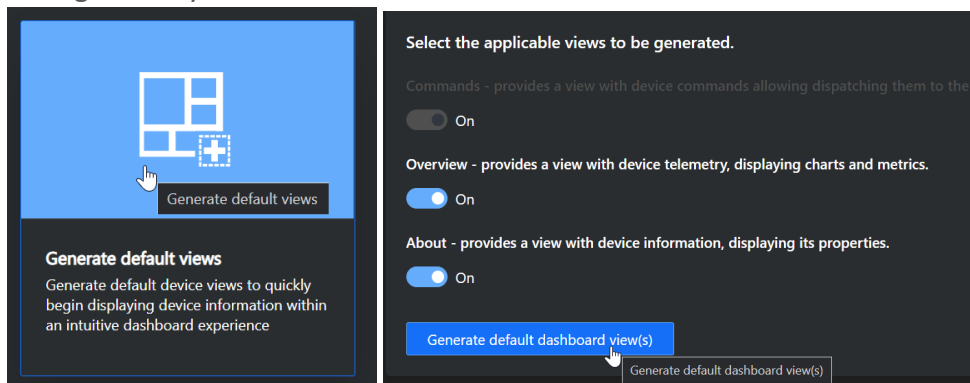


Q4: Post a screenshot of your defined capabilities. Like above.

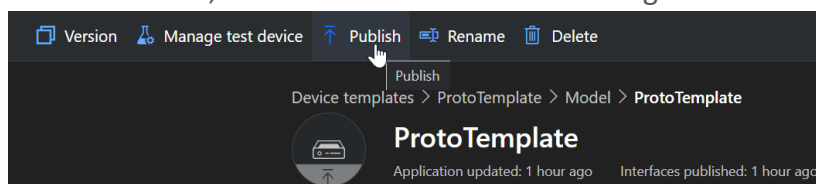
33. To select a visualization click on Views:



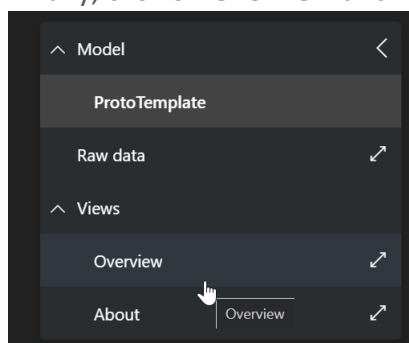
34. Select Generate default view to auto-generate a suitable view. Leave the next settings as they are.

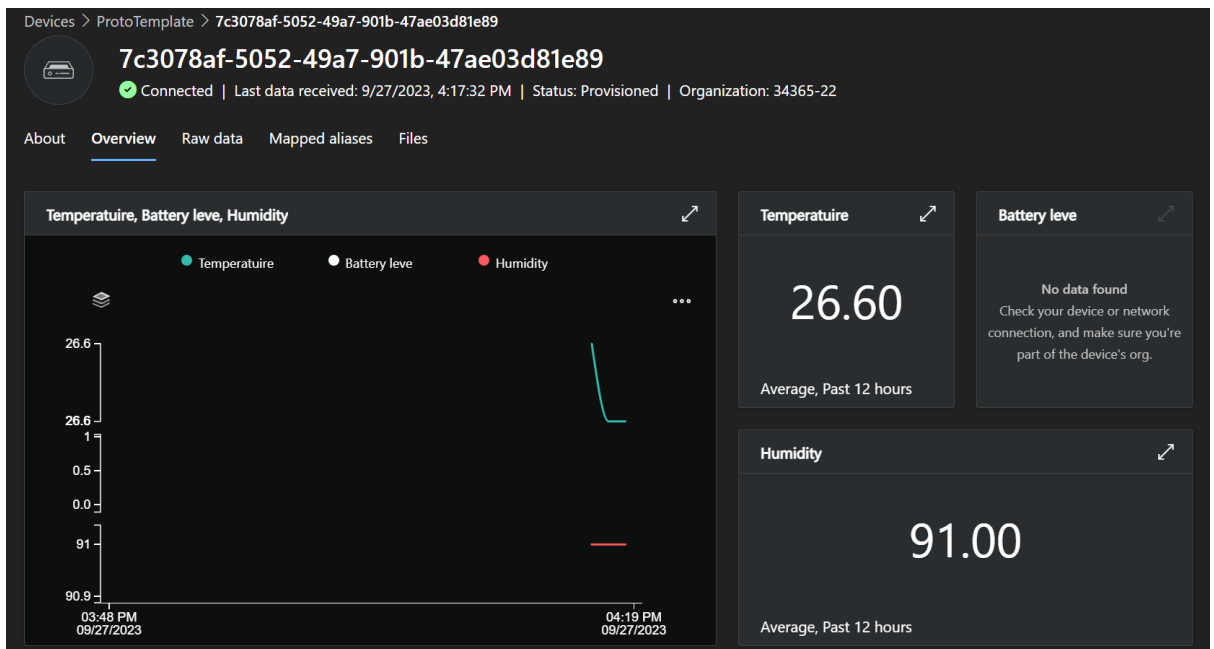


35. When returned, click on Publish to save the changes:



36. Finally, click on Overview and wait for data to arrive and be displayed





37. All data is set to “average”. This can be changed in the Template settings. Head over to Device Templates and select your template. Click on Overview.

The screenshot shows the 'Overview' tab for the 'ProtoTemplate' device template. The page displays the template's configuration, including the Name, Capability type, and Semantic type for each data point. A hand icon points to the 'Overview' tab in the left sidebar.

Name *	Capability type *	Semantic type
TempC	Telemetry	Temperature
BattLevel	Telemetry	None

38. Here you can add new view-types (graph) or edit the ones that are auto-generated. Click the “pencil” symbol or ... dots to “Edit” the various graphs. Several options are available here related to numerical presentation, graphical presentation as well as Display range. It is also possible to add a new Capability to a certain graph, if one wishes to do so.

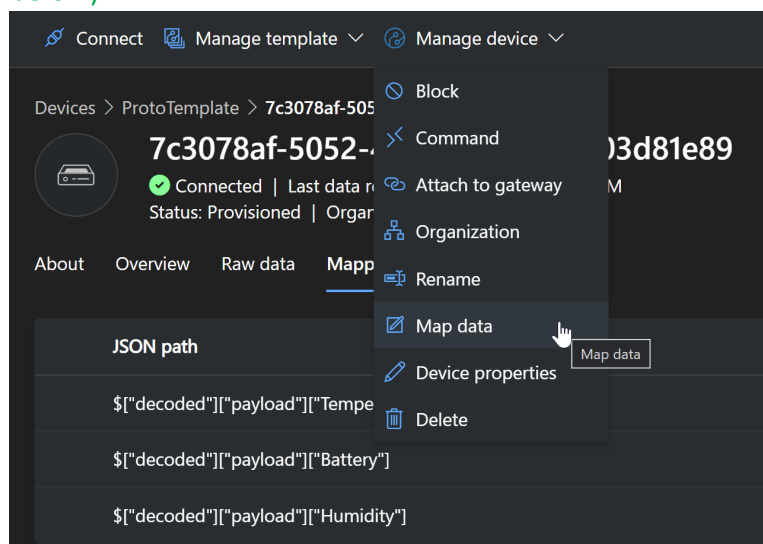
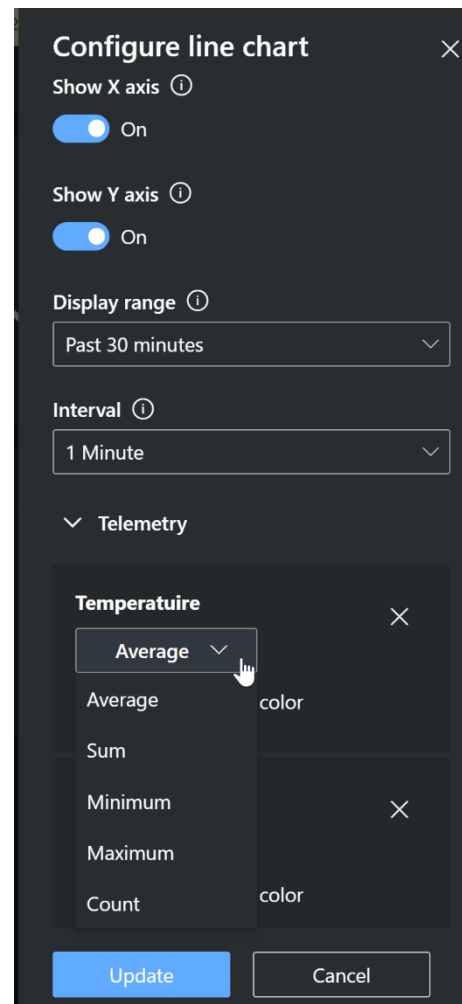
Q5: Post a screenshot of your Overview graph(s), showing data in the graph(s).

Q6: Make changes to your Arduino code to send multiple data sets (e.g. Temperature, Humidity, Battery etc). It is ok if the values are hardcoded. Note that you will have to make changes to your Function code in Helium to decode the relevant values.

Q7: Make changes to your Mapping alias’ to display frame count and RSSI values of the transmissions. Alter your Template to display/visualize these values (or alternatively you can make a new template for this purpose and assign this to your device). It is up to you to choose the type of graph or visualization that you prefer.

Hint: Go to Manage Device and select Map data to add a new Mapping alias.

Look in the JSON data to figure out how the frame count and RSSI can be retrieved (see below)



Map data

Map specific device data to an **alias**—a descriptive label like Temperature or RPM—that describes what it is or does. [Learn more](#)

Tip: Enter telemetry paths only. Aliases mapped to other data types (like properties) won't transmit data.

Timestamp: **Wed Sep 27 2023 18:38:43 GMT+0200 (Central European Summer Time)**

JSON path ⓘ

Alias ⓘ

To select a path, begin typing here, or hover

→ Enter a name

+ Add another alias

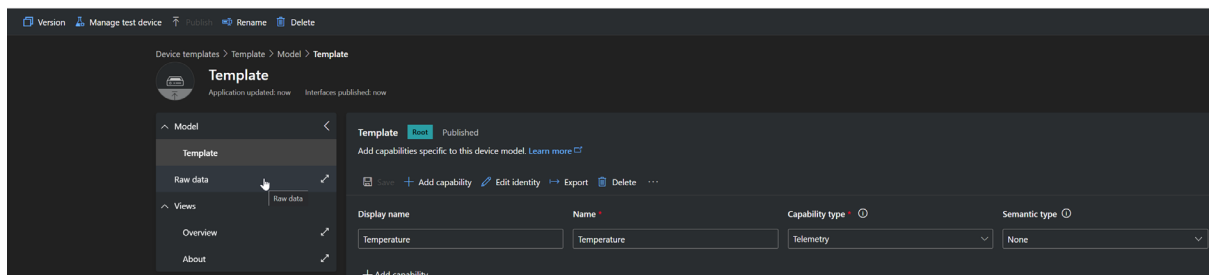
```
22      "status": "success",
23    },
24    "dev_eui": "0004A30B00F19BF4",
25    "devaddr": "06010048",
26    "fcnt": 14,
27    "hotspots": [
28      {
29        "channel": 7,
30        "frequency": 868.5,
31        "hold_time": 0,
32        "id": "11SrNjCywFSr5A8xje4WeCQs",
33        "lat": 55.78302691504508,
34        "long": 12.514492518478399,
35        "name": "fun-marmalade-cobra",
36        "reported_at": 1695832721653,
37        "rssi": -53,
38        "snr": 7.5,
39        "spreading": "SF12BW125",
40        "status": "success"
41      }
42    ],
```

Remember that after adding the new Alias' you will need to add them to the Capabilities. Once added to Capabilities they become available in the Device Template.

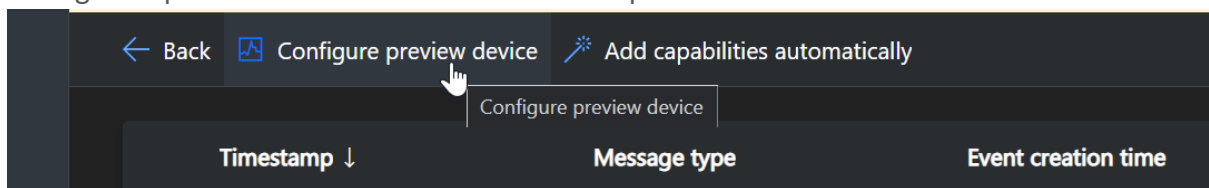
Appendix A: Notes & alternative methods

No Raw data appearing

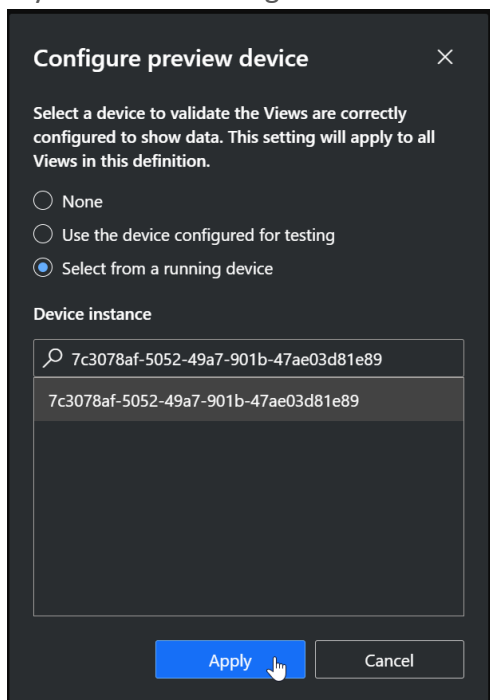
Click Raw Data:



If there are no raw data, and you are sure that your device is transmitting, you may need to Configure a preview device when in Device Templates:



If your device is assigned to the same template you can choose it below.



After clicking Apply you should see data:

← Back [Configure preview device](#) [Add capabilities automatically](#)

	Timestamp ↓	Message type	Event creation time	Temperature	Unmodeled data	Error
① >	9/27/2023, 1:52:39 PM	Telemetry			{"app_eui":"6081F9EF74A1F..."}	
① >	9/27/2023, 1:52:19 PM	Telemetry			{"app_eui":"6081F9EF74A1F..."}	
① >	9/27/2023, 1:51:59 PM	Telemetry			{"app_eui":"6081F9EF74A1F..."}	
① >	9/27/2023, 1:51:39 PM	Telemetry			{"app_eui":"6081F9EF74A1F..."}	
① >	9/27/2023, 1:51:19 PM	Telemetry			{"app_eui":"6081F9EF74A1F..."}	
① >	9/27/2023, 1:50:59 PM	Telemetry			{"app_eui":"6081F9EF74A1F..."}	
① >	9/27/2023, 1:50:39 PM	Telemetry			{"app_eui":"6081F9EF74A1F..."}	

Adding Capabilities automatically

From *Raw data* you can choose to “Add capabilities automatically”

IoT Hub and DPS are updating their TLS certificates with a new Microsoft Certificate Authority (CA) chained under a new CA root - DigiCert Global G2 Root. You will need to take action to ensure your devices can continue to connect. See [here](#) for more details.

← Back [Configure preview device](#) [Add capabilities automatically](#)

	Timestamp ↓	Message type	Event creation time	Temperature	Unmodeled data	Error
No rows found						

Now you will see a JSON object that includes all that has been sent from Helium. It would look something like this:

Data preview

Review your device's data and make any desired changes in the window below. When finished, click **Update** so you can start using this new device data.

You can edit or add to your template anytime.

Contents

```
1 {
2   "telemetries": {
3     "app_eui": "6081F9EF74A1F8E9",
4     "dc": {
5       "balance": 97899,
6       "nonce": 2
7     },
8     "decoded": {
9       "payload": {
10        "Battery": [
11          10
12        ],
13        "Humidity": 91,
14        "Seconds_since_last_wakeup": 23483,
15        "Temperature": "26.6"
16      },
17      "status": "success"
18    },
19    "dev_eui": "0004A30B00F19BF4",
20    "devaddr": "D9000048",
21    "fcnt": 176,
22    "hotspots": [
23      {
24        "channel": 6,
25        "frequency": 868.3,
26        "hold_time": 0,
27        "id": "11SrNjCywFSr5A8xje4WeCQsYZZH3CRtki75GseqVaK916A4VrQ",
28        "lat": 55.78302691504508,
29        "long": 12.514492518478399,
30        "name": "fun-marmalade-cobra",
31        "reported_at": 1695815557872,
32        "rssi": -59,
33        "snr": 8.5,
34        "spreading": "SF12BW125",
35        "status": "success"
36      }
37    ],
38    "id": "7c3078af-5052-49a7-901b-47ae03d81e89",
39    "metadata": {
40      "adr_allowed": false,
41      "cf_list_enabled": false,
42      "multi_buy": 1,
43      "organization_id": "afbead30-9c16-48aa-b9c5-8d7f6ce96a4d",
44      "preferred_hotspots": [],
45      "rx_delay": 1,
46      "rx_delay_actual": 1,
47      "rx_delay_state": "rx_delay_established"
48    },
49    "name": "RN2483_2",
50    "payload": "GgZbClu7",
51    "payload_size": 6,
52    "port": 1,
```

If we choose to automatically add capabilities using this we will get a lot of capabilities, most of which we are not interested in.

Edit the “capability template” to the items that you are interested in. For example:

Data preview

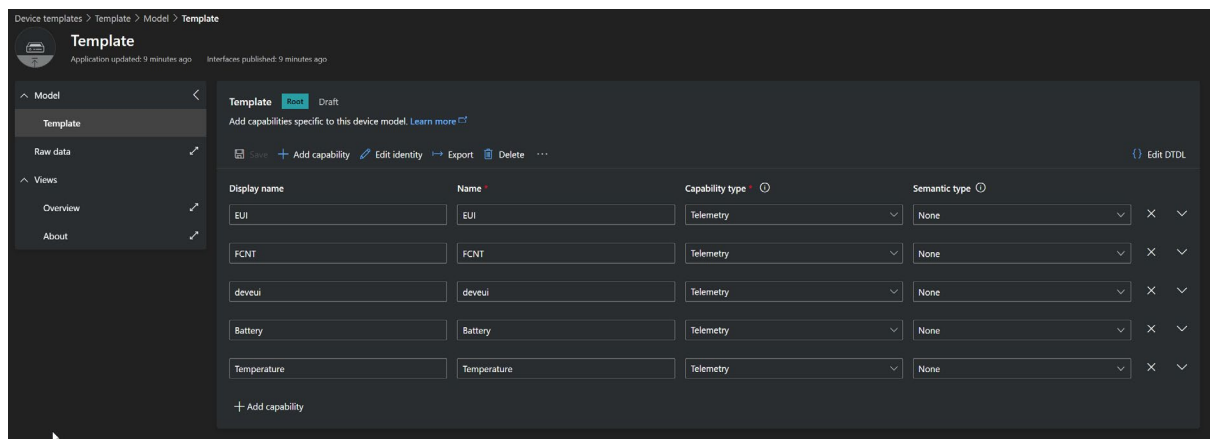
Review your device's data and make any desired changes in the window below. When finished, click **Update**

You can edit or add to your template anytime.

Contents

```
1 {
2   "telemetries": [
3     {
4       "EUI": "6081F9EF74A1F8E9",
5       "FCNT": 176,
6       "deveui": "6081F9EF74A1F8E9",
7       "Battery": [
8         10
9       ],
10      "Temperature": "26.6"
11    }
12  ]
13 }
```

After clicking update you should see something like this (depending on which items you chose):



If you need to you can also reduce this further clicking on the “X”.