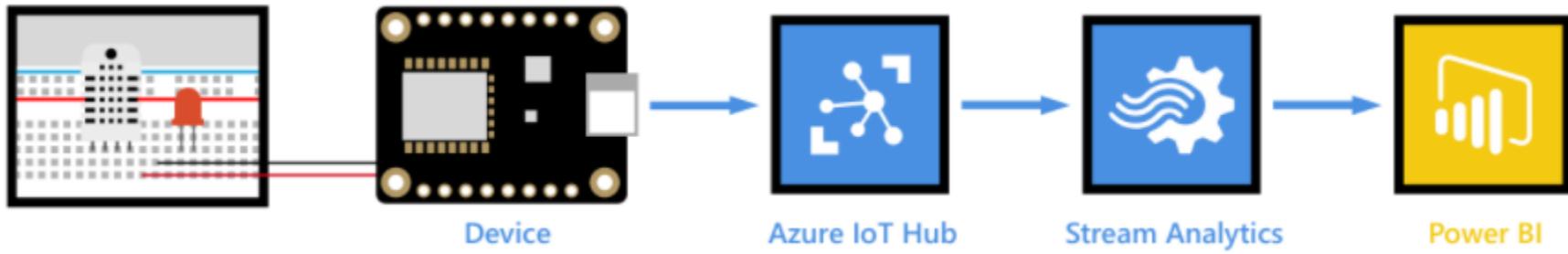


Connect Your Device to the Microsoft Azure IoT Hub

Azure IoT Hub & Azure Stream Analytics & Power BI

Arduino MKR WIFI 1010 to Azure IoT Hub

1. Connect a Arduino MKR WIFI 1010 board securely to Azure IoT Hub using an MQTT client.
2. Create a consumer group on your IoT hub.
3. Create and configure an **Azure Stream Analytics** job to read temperature telemetry from your consumer group and send it to **Power BI**.
4. Create a report of the temperature data in **Power BI** and share it to the web*.



What is the cloud?

- Before the cloud, when a company wanted to provide services to their employees (such as databases or file storage), or to the public (such as websites), they would build and run a data center.
- This ranged from a room with a small number of computers, to a building with many computers.
 - Buying computers
 - Hardware maintenance
 - Power and cooling
 - Networking
 - Security, including securing the building and securing the software on the computers
 - Software installation and updates
- This could be very expensive, require a wide range of skilled employees, and be very slow to change when needed.

A case: An online store needed to plan for a busy holiday season
Buying computers just for those moments, and sitting idle till the next



Setting Up Azure Account & Registering our device

Create a Cloud Subscription

- <https://azure.microsoft.com/en-us/free/students/>

The screenshot shows the Microsoft Azure Education Overview page. At the top, there's a navigation bar with links for Home, Education | Overview, and other account details. A prominent message encourages users to "Click here to complete your student profile". Below this, a main heading says "Start building the future with Azure for Students!" and notes that students are eligible for \$100 credit. A large blue button labeled "Claim your Azure credit now" is highlighted with a yellow circle. On the left, a sidebar lists "Overview", "Get started", "Learning resources" (with sub-options for Roles, Software, and Learning), "My account", "Profile", "Need help?", and "Support". In the center, there's a section titled "Explore Azure roles" with three cards: "Data Scientist" (skilled in technology and social sciences), "AI Engineer" (develop cognitive services, machine learning, and knowledge), and "Developer" (design, build, and test software systems). The bottom right corner features a decorative graphic of overlapping yellow circles.

Azure free services

← → ⌂ ⌂ portal.azure.com/?Microsoft_Azure_Education_correlationId=9141cf6536d74980a12bc2830f9a8b94&Microsoft_Azure_

Microsoft Azure Search resources, services, and docs (G+)

Home > Education

Education | Get started

Overview

Get started

Learning resources

- Roles
- Software
- Learning
- Templates

My account

- Profile

Need help?

- Support

the development resources you ne

Explore Azure roles

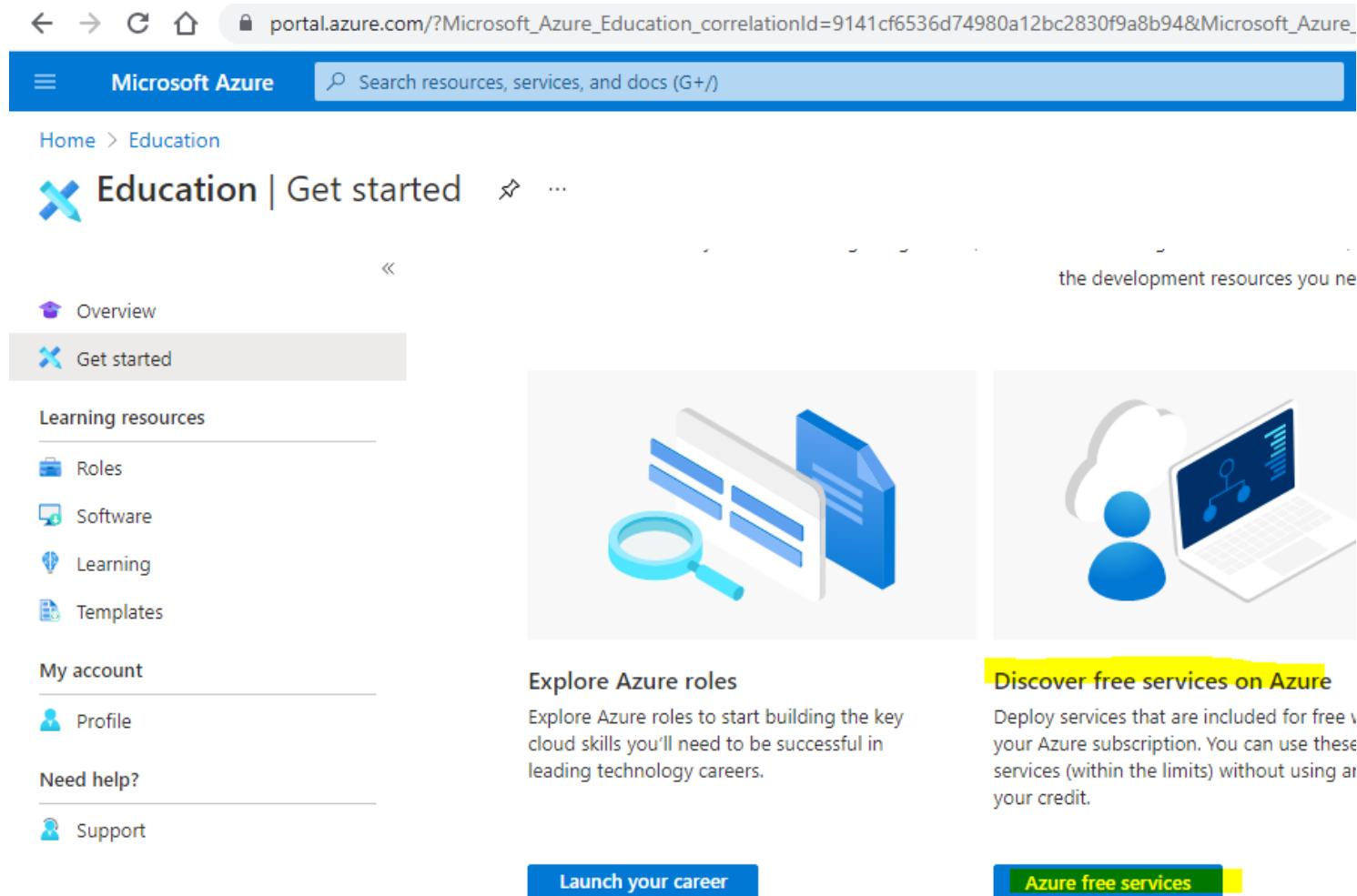
Explore Azure roles to start building the key cloud skills you'll need to be successful in leading technology careers.

Launch your career

Discover free services on Azure

Deploy services that are included for free with your Azure subscription. You can use these services (within the limits) without using any of your credit.

Azure free services



Create Microsoft IoT Hub

← → C ⌂ 🔒 portal.azure.com/#blade/Microsoft_Azure_Billing/FreeServicesBlade

Microsoft Azure Search resources, services, and docs (G+/)

Home > Free services ⚡ ...

 Security Center
SECURITY + IDENTITY

100 policy assessment and recommendations

Prevent, detect, and respond to threats with increased visibility and control over the security of your Azure resources. [Learn more](#)

 Advisor
MONITORING + MANAGEMENT

Unlimited

Get personalized recommendations to Azure best practices. [Learn more](#)

 Microsoft IoT Hub
INTERNET OF THINGS

8,000 messages per day

Connect, monitor, and manage billions of IoT assets. [Learn more](#)

[Create](#)

How to Check Your Account Balance

Microsoft | Azure Sponsorships

- <https://www.microsoftazuresponsorships.com>

- Check Your Balance

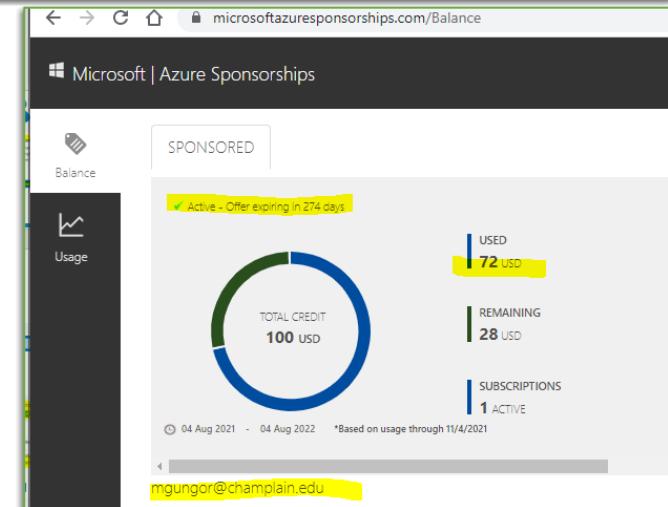
- Get the most out of your sponsored account

- Usage Details

- Drill down into the usage statistics of your subscriptions

A screenshot of a web browser displaying the Microsoft Azure Sponsorships Usage page. The URL is microsoftazuresponsorships.com/Usage. The page title is "Microsoft | Azure Sponsorships". On the left, there is a sidebar with "Balance" and "Usage" options. The main content area shows "Azure for Students" with the ID 77E53992-1B53-482D-B610-F9B54DB83A77. It indicates a "Subscription Cost: \$71.79". Below this is a table of service resource usage:

SERVICE NAME	SERVICE RESOURCE	SPEND
Stream Analytics	Standard Streaming Unit	\$26.31
Virtual Machines	E2a v4/E2as v4	\$15.39
IoT Hub	Standard S1 Unit	\$12.9
Virtual Machines	D1 v2/D5t v2	\$8.92
Storage	Hot LRS Write Operations	\$2.84
Storage	Class 2 Operations	\$2.04
Storage	S10 Disks	\$1.95
Virtual Network	Static Public IP	\$0.87
Storage	All Other Operations	\$0.21
Storage	Disk Operations	\$0.19



Connect your Device to the IoT service

- Once the hub is created, your IoT device can connect to it.
- Only **added devices** can connect to a service, so you will need to register your device first.

What is Azure IOT Hub?

<https://youtu.be/eEIPnYLj73A>

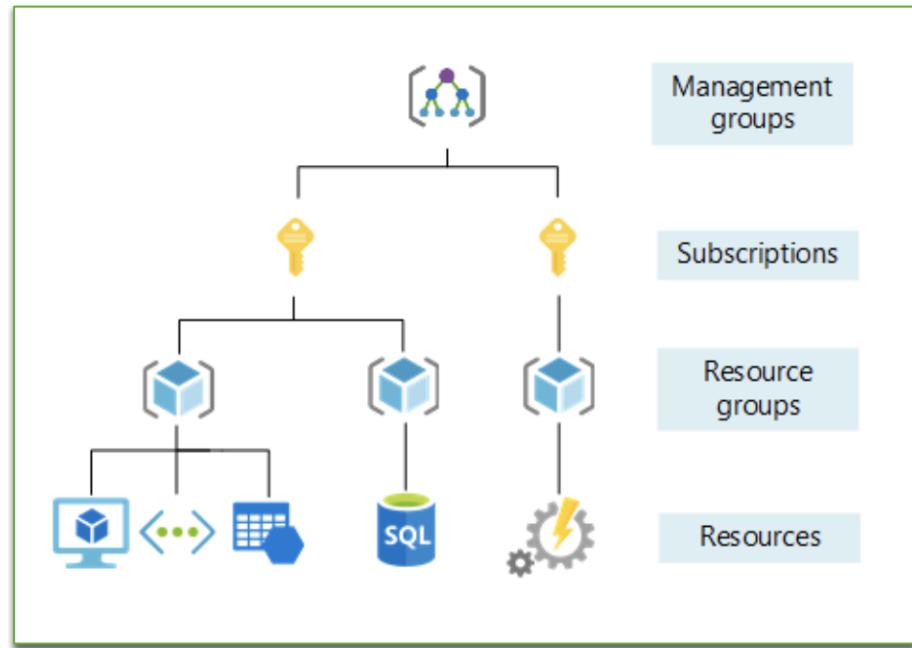
[Optional] – install the Azure CLI (Command Line Interface)

- Azure Command-Line Interface
 - <https://docs.microsoft.com/en-us/cli/azure/>
- Steps to Create an IoT service in the cloud
 - <https://github.com/microsoft/IoT-For-Beginners/blob/main/2-farm/lessons/4-migrate-your-plant-to-the-cloud/README.md#task---install-the-azure-cli>

Create IoT Hub and Resource Group

What is Resource Group?

- Azure services, such as IoT Hub instances, virtual machines, databases, or AI services, are referred to as **resources**.
- Every resource has to live inside a **Resource Group**, a logical grouping of one or more resources.
- [Optional] If you like to user Azure CLI follow the steps at the link below
 - <https://github.com/microsoft/IoT-For-Beginners/blob/main/2-farm/lessons/4-migrate-your-plant-to-the-cloud/README.md#task---create-a-resource-group>



Name your IoT Hub and Resource Group

IoT hub

Microsoft

Basics Networking Management Tags Review + create

Create an IoT hub to help you connect, monitor, and manage billions of your IoT assets. [Learn more](#)

Project details

Choose the subscription you'll use to manage deployments and costs. Use resource groups like folders to help you organize and manage resources.

Subscription * ⓘ

Azure for Students



Resource group * ⓘ

RoomTempratureResourceGroup



[Create new](#)

Instance details

IoT hub name * ⓘ

RoomTempHub

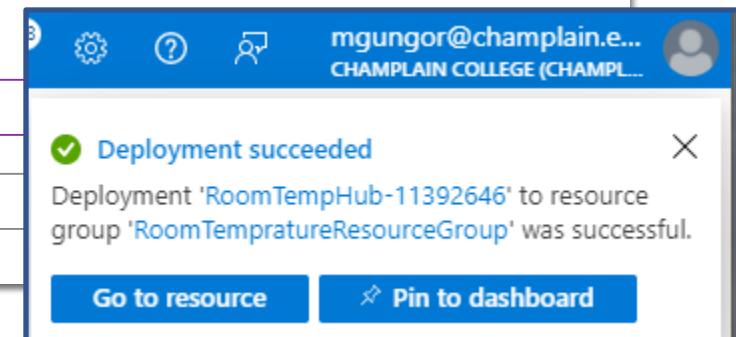
Region * ⓘ

East US

This will take several minutes
be patient!

Mine took ~4 Minutes

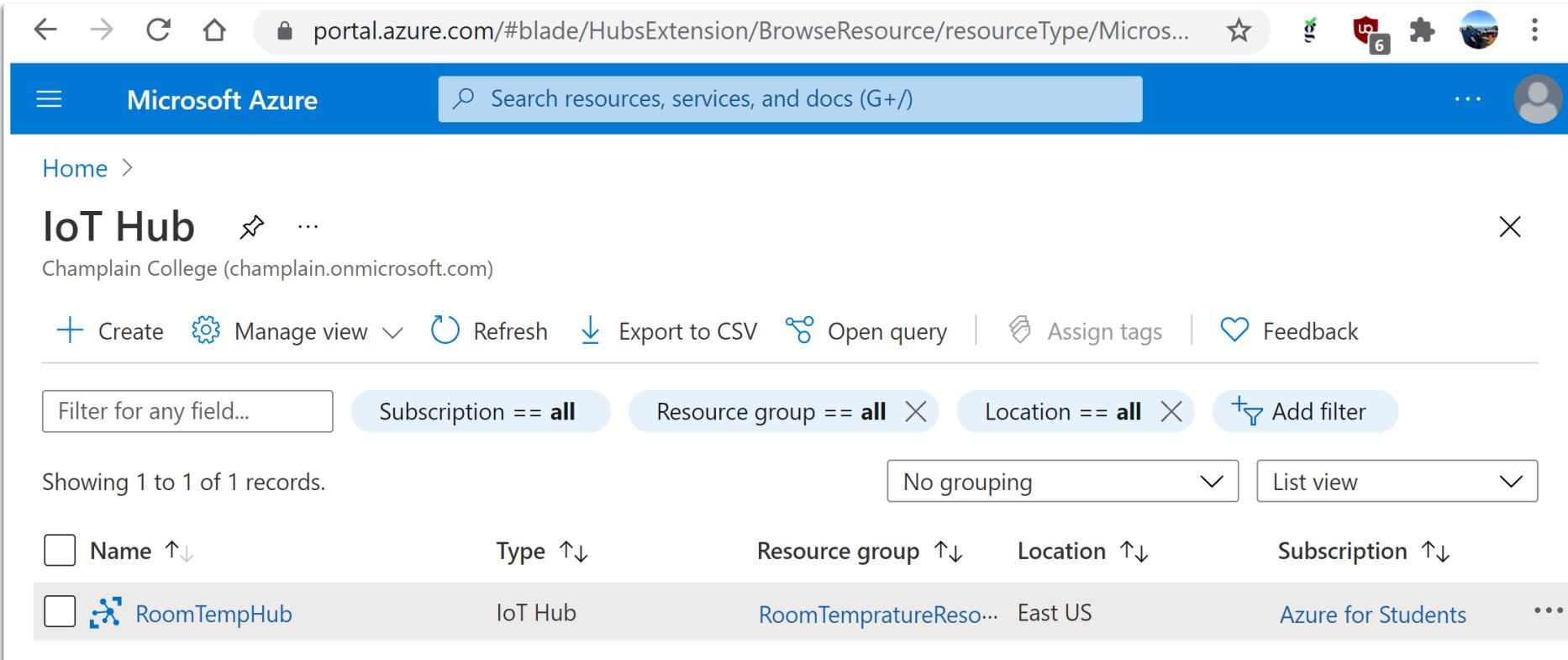
... Deployment is in progress



Completed! IoT Hub

 Deployment succeeded

- You can now have an IoT Hub in your resource group.



The screenshot shows the Microsoft Azure portal interface for managing IoT Hubs. The top navigation bar includes icons for back, forward, home, and search, along with account and notification badges. The main title is "Microsoft Azure" with a search bar below it. Below the search bar, the current location is listed as "Champlain College (champlain.onmicrosoft.com)". The main content area is titled "IoT Hub" with a "Create" button and a three-dot menu icon. A header bar contains "Manage view" (dropdown), "Refresh", "Export to CSV", "Open query", "Assign tags", and "Feedback". Below this are filter options: "Filter for any field...", "Subscription == all" (selected), "Resource group == all" (with a close X), "Location == all" (with a close X), and "Add filter". The results section shows "Showing 1 to 1 of 1 records." with a dropdown for "No grouping" and a "List view" option. The table lists one IoT Hub entry: "RoomTempHub" (Type: IoT Hub, Resource group: RoomTempratureReso..., Location: East US, Subscription: Azure for Students). The table has columns for Name, Type, Resource group, Location, and Subscription.

Name ↑↓	Type ↑↓	Resource group ↑↓	Location ↑↓	Subscription ↑↓
<input type="checkbox"/> RoomTempHub	IoT Hub	RoomTempratureReso...	East US	Azure for Students

Navigate to your hub and look for the Hostname

- The Hostname, this is your broker!

Microsoft Azure Search resources, services, and docs (G+)

Home > RoomTempHub IoT Hub

Search (Ctrl+ /) Move Delete Refresh

Overview

Essentials

Resource group (change)	:	RoomTempratureResourceGroup	Hostname	:	RoomTempHub.azure-devices.net
Status	:	Active	Pricing and scale tier	:	S1 - Standard
Current location	:	East US	Number of IoT Hub units	:	1
Subscription (change)	:	Azure for Students	Minimum TLS Version	:	1.0
Subscription ID	:	77e53992-1b53-482d-b610-f9b54db83a77			
Tags (change)	:	Click here to add tags			

Usage Get started

Show data for last: 1 Hour 6 Hours 12 Hours 1 Day 7 Days 30 Days

IoT Hub Usage Number of messages used

Add Device

Task – Add Device to IoT Hub

Next we need to provision a device, look for “Devices” along the left side and select “Add Device”

The screenshot shows the Microsoft Azure IoT Hub Devices page for a resource named "RoomTempHub". The left sidebar has a "Device management" section with "Devices" highlighted. The main area shows a search bar, a "Find devices" button, and a "Add Device" button. Below is a table with columns for Device ID, Status, and Last Status Update. A message at the bottom states "There are no IoT devices to display."

Device ID	Status	Last Status Update
There are no IoT devices to display.		

 Create a device ...

Find Certified for Azure IoT devices in the Device Catalog

Device ID * ⓘ

MKRWIFI1010DeviceID

Authentication type ⓘ

Symmetric key X.509 Self-Signed X.509 CA Signed

Auto-generate keys ⓘ



Connect this device to an IoT hub ⓘ

Enable Disable

Parent device ⓘ

No parent device

[Set a parent device](#) Save

Adding a Device

- Name the device,
- Leave all other settings as default; authentication as “Symmetric key” and
- Leave “Auto-generate keys” checked,
- “Connect this device to an IoT Hub” enabled
- and save.
- Your Device ID is the **MKRWIFI1010DeviceID**.

Device Added!

- Refresh to see the newly added device
- Your Device ID is the **MKRWIFI1010DeviceID**

Home > RoomTempHub

RoomTempHub | Devices

IoT Hub

Search (Ctrl+ /)

View, create, delete, and update devices in your IoT Hub.

Device name: enter device ID

Find devices

Add Device Refresh Delete

Find using a query

Device ID	Status	Last Status Update	Authentication Type	Cloud to Device Message...
MKRWIFI1010DeviceID	Enabled	--	Sas	0

Device management

- Devices
- IoT Edge
- Configurations
- Updates
- Queries



Coding Part 1

Sending text message

“arduino_secrets.h”



- This contains information for connecting to
 - Wi-Fi Info => WE GOT THIS
 - MQTT broker => WE GOT THIS
 - Device ID => WE GOT THIS
 - Device Pass => WE NEED THIS

A screenshot of a code editor window. The tabs at the top are 'platformio.ini', 'main.cpp', 'arduino_secrets.h', and 'Ex...'. The current file is 'arduino_secrets.h'. The code in the editor is:

```
2021Nov2Azure > src > main.cpp > ...
24
25 #include "arduino_secrets.h"
26
27 // Enter your sensitive data in arduino_secrets.h
28 const char ssid[] = SECRET_WIFI_SSID;
29 const char pass[] = SECRET_WIFI_PASS;
30 const char broker[] = SECRET_BROKER;
31 String deviceId = SECRET_DEVICE_ID;
32 String devicePass = SECRET_DEVICE_PASSWORD;
```

Partial My: arduino_secrets.h

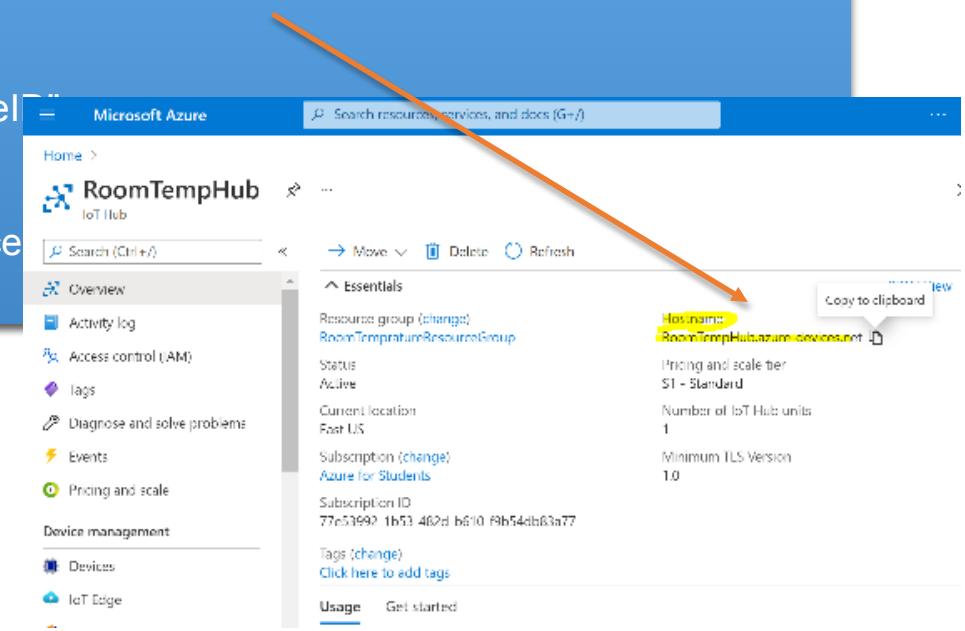
- Following is the secrets file content
- Except WIFI info the rest should be different in your file

```
// Fill in your WiFi networks SSID and password
#define SECRET_WIFI_SSID    "ChamplainPSK"
#define SECRET_WIFI_PASS     "letusdare"

// Fill in the hostname of your Azure IoT Hub broker
#define SECRET_BROKER    "RoomTempHub.azure-devices.net"

// Fill in the device id
#define SECRET_DEVICE_ID   "MKRWIFI1010Device1"

// Fill in the device password
#define SECRET_DEVICE_PASSWORD "SharedAccessKey"
```

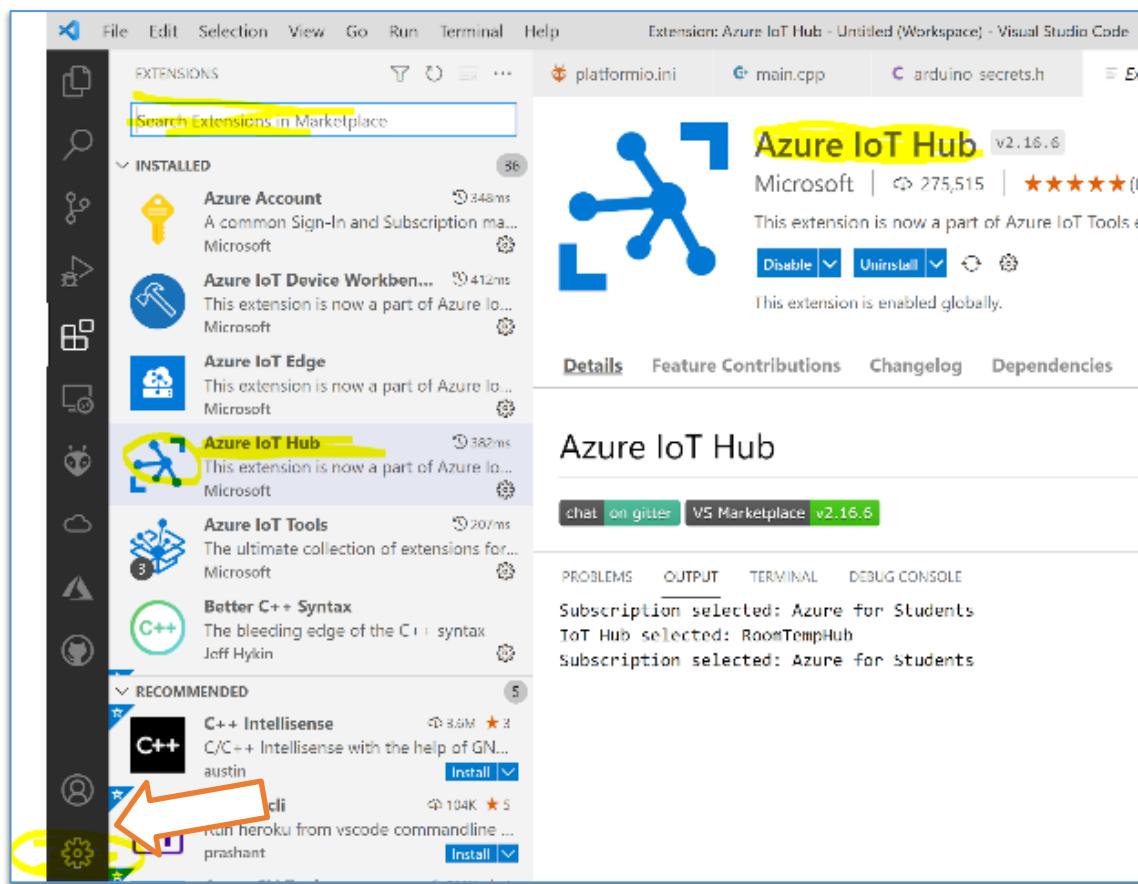




Shared Access Signature (SAS) as Device Password

Install - Azure IoT Hub Extension

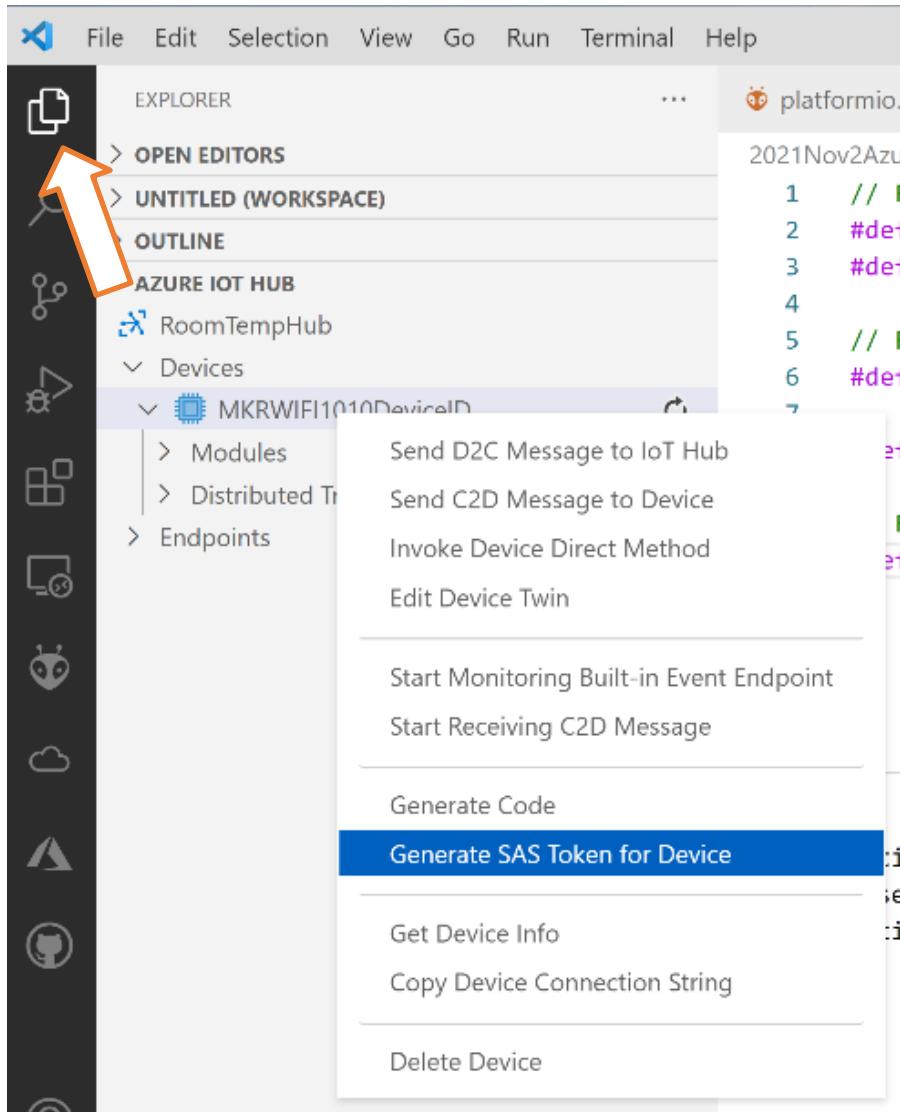
- We will use this extension to generate Shared Access Signature (SAS) as Device Password
- Installing Steps: Visual Studio Code / Manage / Extensions (Ctrl + Shift + X)



Once installed

- **Login to Azure using VS Code,**
- **Then in the Explorer pane you'll be able to select Azure IoT Hub,**
- **Locate**
 - your Hub,
 - the Device you created,
- **Right click** and Generate SAS Token

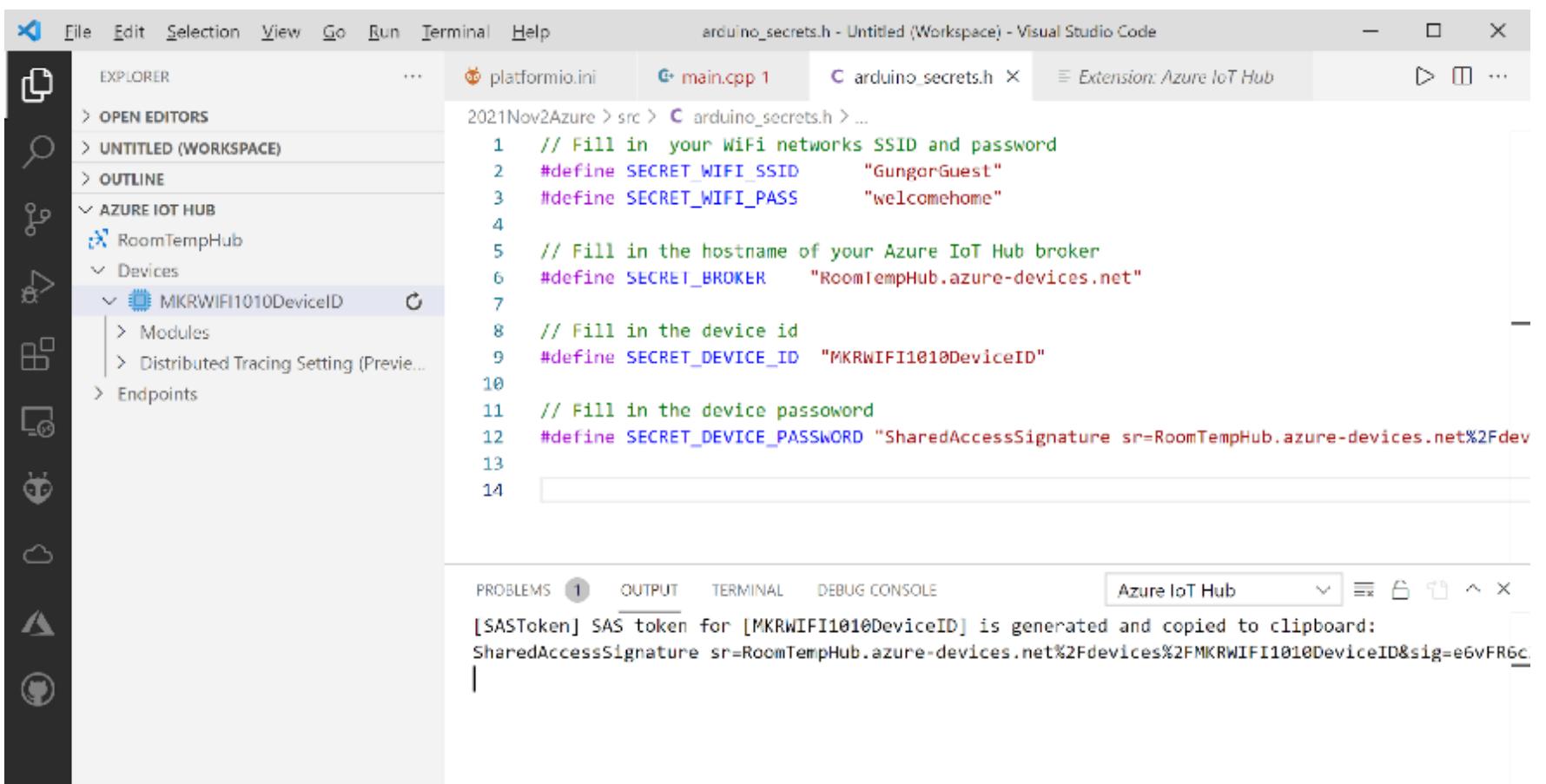
Creating the Device Password



- Click on
- Explorer
 - AZURE IOT HUB
 - Devices
 - Right Click on a device and click **Generate SAS Token for Device**

SAS:
Shared Access Signature
as Device Password

Generated SAS Token as Device Password



The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** arduino_secrets.h - Untitled (Workspace) - Visual Studio Code.
- Explorer:** Shows the project structure:
 - OPEN EDITORS
 - UNTITLED (WORKSPACE)
 - OUTLINE
 - AZURE IOT HUB
 - RoomTempHub
 - Devices
 - MKRWIFI1010DeviceID
 - Modules
 - Distributed Tracing Setting (Preview...)
 - Endpoints
- Code Editor:** Displays the arduino_secrets.h file content:

```
1 // Fill in your WiFi networks SSID and password
2 #define SECRET_WIFI_SSID      "GungorGuest"
3 #define SECRET_WIFI_PASS       "welcomemehome"
4
5 // Fill in the hostname of your Azure IoT Hub broker
6 #define SECRET_BROKER        "RoomTempHub.azure-devices.net"
7
8 // Fill in the device id
9 #define SECRET_DEVICE_ID     "MKRWIFI1010DeviceID"
10
11 // Fill in the device password
12 #define SECRET_DEVICE_PASSWORD "SharedAccessSignature sr=RoomTempHub.azure-devices.net%2Fdev
13
14
```
- Output Panel:** Shows the message: [SASToken] SAS token for [MKRWIFI1010DeviceID] is generated and copied to clipboard:
SharedAccessSignature sr=RoomTempHub.azure-devices.net%2Fdevices%2FMKRWIFI1010DeviceID&sig=e6vFR6c

Set the expiration in hours, the key is output and copied to your clipboard.

You can use it exactly as it is for the
SECRET_DEVICE_PASSWORD, just enclose it in quotes. Your
code should include "\$SharedAccessSignature sr=", do not omit this.

Running of the application

File Edit Selection View Go Run Terminal Help main.cpp - Untitled (Workspace) - Visual Studio Code

EXPLORER platformio.ini main.cpp arduino_secrets.h Extension: Azure IoT Hub

> OPEN EDITORS
> UNTITLED (WORKSPACE)
> OUTLINE
AZURE IOT HUB
RoomTempHub
Devices
MKRWIFI1010DeviceID
Modules
Distributed Tracing Settin...
Endpoints

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
79 // subscribe to a topic
80 mqttClient.subscribe("devices/" + deviceId + "/messages/devicebound")
```

--- Available filters and text transformations: colorize, debug, default, direct, hexlify, log2file, nocontrol, printable, send_on_enter, time
--- More details at <http://bit.ly/pio-monitor-filters>
--- Miniterm on COM6 9600,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H --

Username: RoomTempHub.azure-devices.net/MKRWIFI1010DeviceID/api-version=2018-06-30

Device Pass: SharedAccessSignature sr=RoomTempHub.azure-devices.net%2Fdevices%2FMKRWIFI1010DeviceID&sig=e6vFR6c3Qw54DYV3IpWU66rnB%2B6wnAXOM6Dzlhv697I%3D&se=1637749921

Attempting to connect to SSID: GungorGuest
You're connected to the network

Attempting to MQTT broker: RoomTempHub.azure-devices.net
Connecting to MQTT broker - Failed Error Code:-2

You're connected to the MQTT broker

Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message

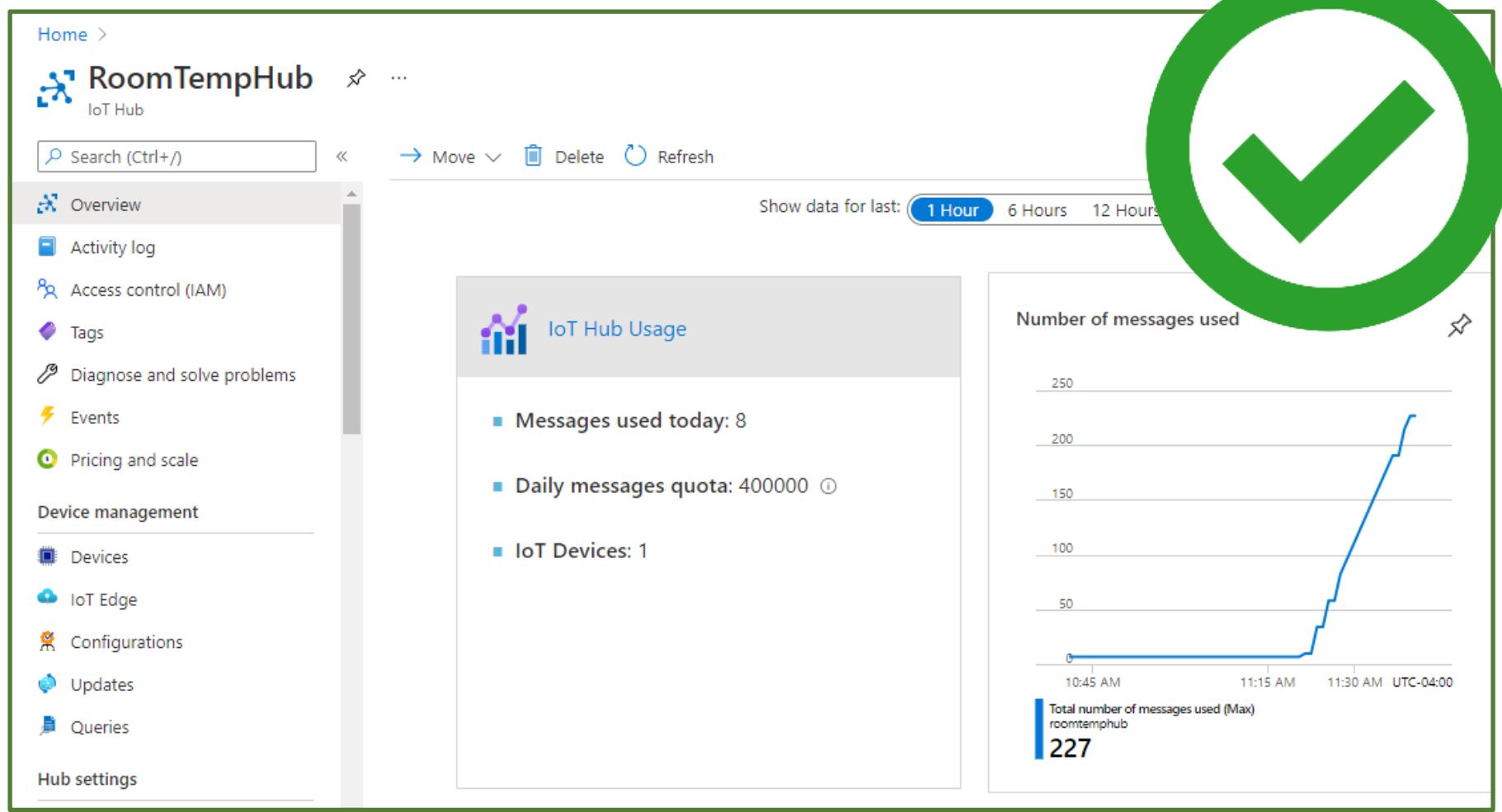
env:mkrwifi1010 (2021Nov2Azure) Azure: mgung

Possible Error Messages

#define MQTT_CONNECTION_REFUSED	-2
#define MQTT_CONNECTION_TIMEOUT	-1
#define MQTT_SUCCESS	0
#define MQTT_UNACCEPTABLE_PROTOCOL_VERSION	1
#define MQTT_IDENTIFIER_REJECTED	2
#define MQTT_SERVER_UNAVAILABLE	3
#define MQTT_BAD_USER_NAME_OR_PASSWORD	4
#define MQTT_NOT_AUTHORIZED	5

View from the IoT Hub

- Yes! We are now sending some data to Azure!

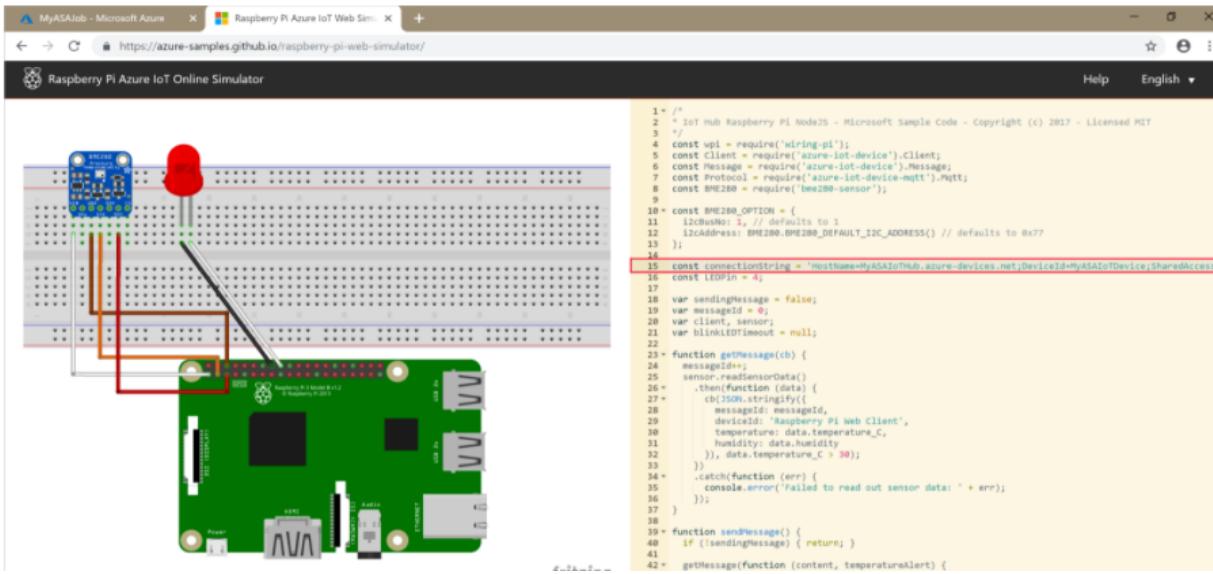


[In case] Troubleshooting: Run the IoT simulator to make sure you can connect

- Link: <https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-quick-create-portal>

Run the IoT simulator

- Open the Raspberry Pi Azure IoT Online Simulator.
- Replace the placeholder in Line 15 with the Azure IoT Hub device connection string you saved in a previous section.
- Click Run. The output should show the sensor data and messages that are being sent to your IoT Hub.



1. Open the Raspberry Pi Azure IoT Online Simulator.



[Optional] Monitor Events via Azure CLI

[Optional] Monitor Messages via Command Line Interface

- Run the following command in your command prompt or terminal to monitor messages sent to your IoT Hub:

```
az iot hub monitor-events --properties anno --hub-name <hub_name>
```

```
az iot hub monitor-events --properties anno --hub-name RoomTempHub
{
    "iothub-message-source": "Telemetry",
    "x-opt-sequence-number": 1795,
    "x-opt-offset": "702696",
    "x-opt-enqueued-time": 1629583762434
},
    "payload": "hello 439753"
}

{
    "event": {
        "origin": "MyMKRWiFi1010",
        "module": "",
        "interface": "",
        "component": "",
        "properties": {},
        "annotations": {
            "iothub-connection-device-id": "MyMKRWiFi1010",
            "iothub-connection-auth-method": "{\"scope\":\"device\",\"type\":\"x509Certificate\",
\"issuer\":\"external\",\"acceptingIpFilterRule\":null}",
            "iothub-connection-auth-generation-id": "637650923234932208",
            "iothub-enqueuedtime": 1629583767762,
            "iothub-message-source": "Telemetry",
            "x-opt-sequence-number": 1796,
            "x-opt-offset": "703088",
            "x-opt-enqueued-time": 1629583767762
},
        "payload": "hello 444754"
}
```

The time values in the annotations are in UNIX time, representing the number of seconds since midnight on 1st January 1970.

Exit the event monitor when you are done.

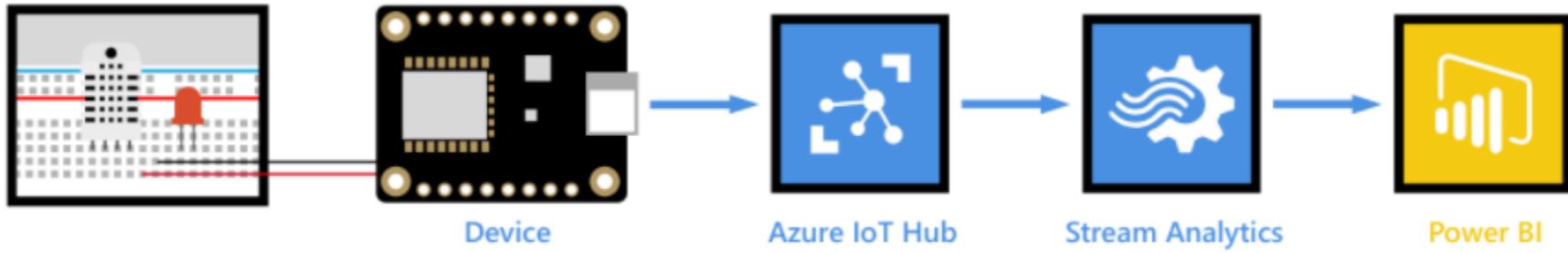
<https://www.epochconverter.com/>



Where are we now?

Arduino MKR WIFI 1010 to Azure IoT Hub

- Connect a Arduino MKR WIFI 1010 board securely to Azure IoT Hub using an MQTT client. => **COMPLETED**
- Create a consumer group on your IoT hub.
- Create and configure an **Azure Stream Analytics** job to read temperature telemetry from your consumer group and send it to **Power BI**.
- Create a report of the temperature data in **Power BI** and share it to the web.





Visualize real-time sensor data from Azure IoT Hub

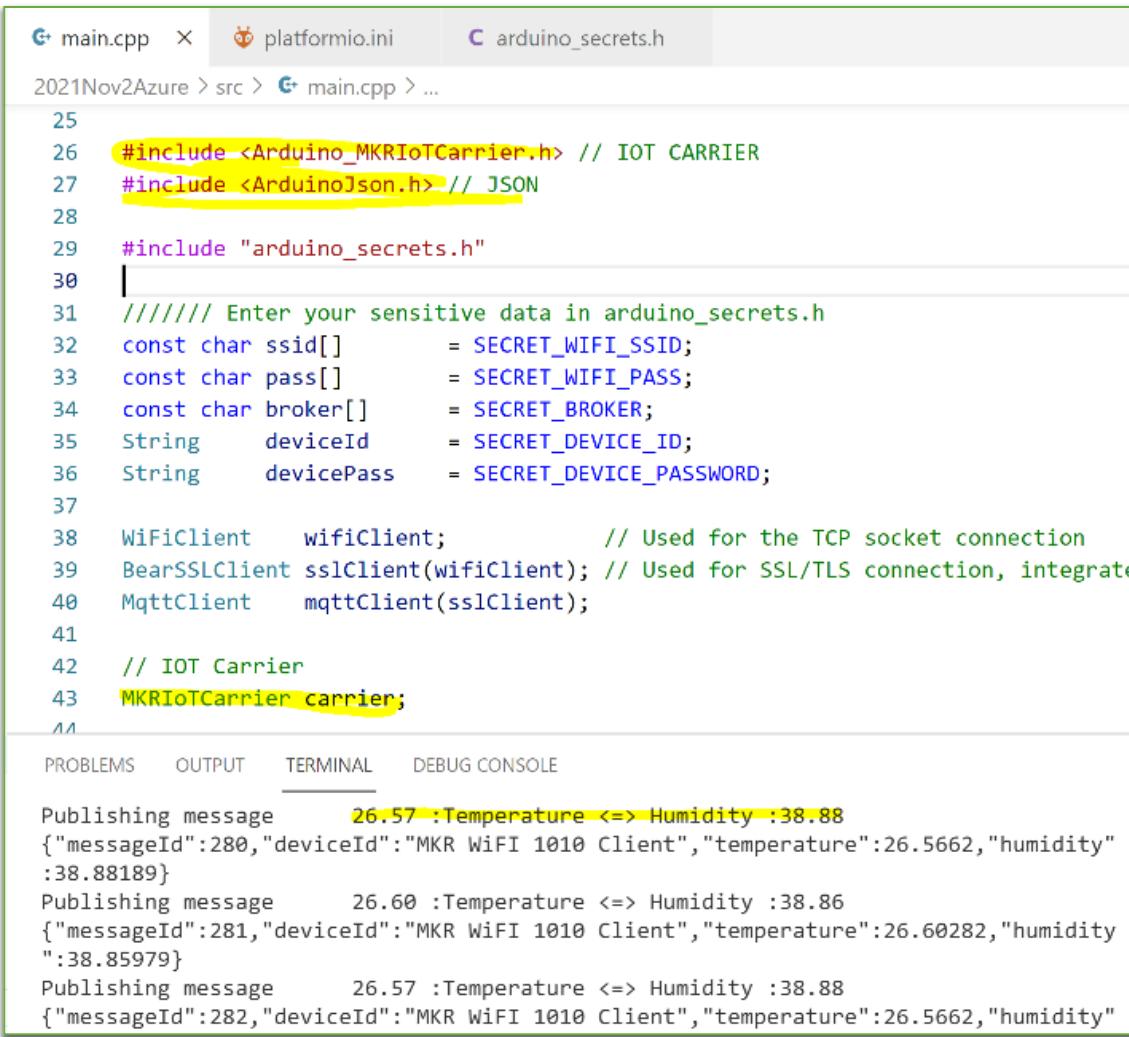
Reference

<https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-live-data-visualization-in-power-bi>

Coding Part 2

**Sending
temperature and humidity telemetry**

Coding Side – Sending Telemetry Data



The screenshot shows the PlatformIO IDE interface with three tabs: main.cpp, platformio.ini, and arduino_secrets.h. The main.cpp tab displays C++ code for an Arduino project. The code includes includes for MKR IoT Carrier and JSON libraries, defines for WiFi credentials, and initializes WiFiClient, sslClient, and mqttClient. It also includes code for the IoT Carrier. The terminal window at the bottom shows three messages being published, each containing a JSON object with messageId, deviceId, temperature, and humidity fields.

```
main.cpp  X  platformio.ini  C  arduino_secrets.h
2021Nov2Azure > src > main.cpp > ...
25
26 #include <Arduino_MKRIoTCarrier.h> // IOT CARRIER
27 #include <ArduinoJson.h> // JSON
28
29 #include "arduino_secrets.h"
30
31 // Enter your sensitive data in arduino_secrets.h
32 const char ssid[] = SECRET_WIFI_SSID;
33 const char pass[] = SECRET_WIFI_PASS;
34 const char broker[] = SECRET_BROKER;
35 String deviceId = SECRET_DEVICE_ID;
36 String devicePass = SECRET_DEVICE_PASSWORD;
37
38 WiFiClient wifiClient; // Used for the TCP socket connection
39 BearSSLClient sslClient(wifiClient); // Used for SSL/TLS connection, integrate
40 MqttClient mqttClient(sslClient);
41
42 // IOT Carrier
43 MKRIoTCarrier carrier;
44
PROBLEMS   OUTPUT   TERMINAL   DEBUG CONSOLE
Publishing message 26.57 :Temperature <= Humidity :38.88
{"messageId":280,"deviceId":"MKR WiFi 1010 Client","temperature":26.5662,"humidity":38.88189}
Publishing message 26.60 :Temperature <= Humidity :38.86
{"messageId":281,"deviceId":"MKR WiFi 1010 Client","temperature":26.60282,"humidity":38.85979}
Publishing message 26.57 :Temperature <= Humidity :38.88
{"messageId":282,"deviceId":"MKR WiFi 1010 Client","temperature":26.5662,"humidity":38.88189}
```

- This time we are using JSON format to send telemetry messages.
- publishTemperatureHumidity() function added
- IoT Carrier code changes added.



Forming JSON Message to Send

```
void publishTemperatureHumidity() {  
    Serial.print("Publishing message\t");  
  
    // Read Sensor Data  
    float temperature = carrier.Env.readTemperature();  
    float humidity = carrier.Env.readHumidity();  
  
    ...  
    DynamicJsonDocument doc(1024);  
    doc["messageId"] = meesageId++;  
    doc["deviceId"] = "MKR WiFi 1010 Client";  
    doc["temperature"] = temperature;  
    doc["humidity"] = humidity;  
  
    string telemetry;  
    serializeJson(doc, telemetry);  
  
    Serial.println(telemetry.c_str());  
  
    // send message, the Print interface can be used to set the message contents  
    mqttClient.beginMessage("devices/" + deviceId + "/messages/events/");  
    mqttClient.print(telemetry.c_str());  
    mqttClient.endMessage();  
}
```

Reading telemetry from sensors

Forming JSON payload

Sending the message



QuickStart: Create a Stream Analytics job by using the Azure portal

What Is Azure Stream Analytics?

<https://www.youtube.com/watch?v=po-R-XsSZnM>

Visualize real-time sensor data

- To visualize real-time sensor data that your **Azure IoT hub** receives.
- To do so, you configure an **Azure Stream Analytics** job **to consume** the data from IoT Hub
- and route it to a dataset in **Power BI**.

Add a consumer group to your IoT hub

To add a consumer group to your IoT hub, follow these steps:

1. In the [Azure portal](#), open your IoT hub.
2. On the left pane, select **Built-in endpoints**.
Enter a name for your new consumer group in the text box under **Consumer groups**.
3. Click anywhere outside the text box to save the consumer group.

Create a new Consumer Group

Home > IoT Hub > RoomTempHub

RoomTempHub | Built-in endpoints ⌂ ⌂ ...

IoT Hub

Search (Ctrl+ /) Save Undo

Event Hub Details

Partitions ①
4

Event Hub-compatible name ①
roomtemphub

Retain for ①

Consumer Groups ①

- Consumer Groups
- \$Default
- roomtempconsumergroup

Create new consumer group

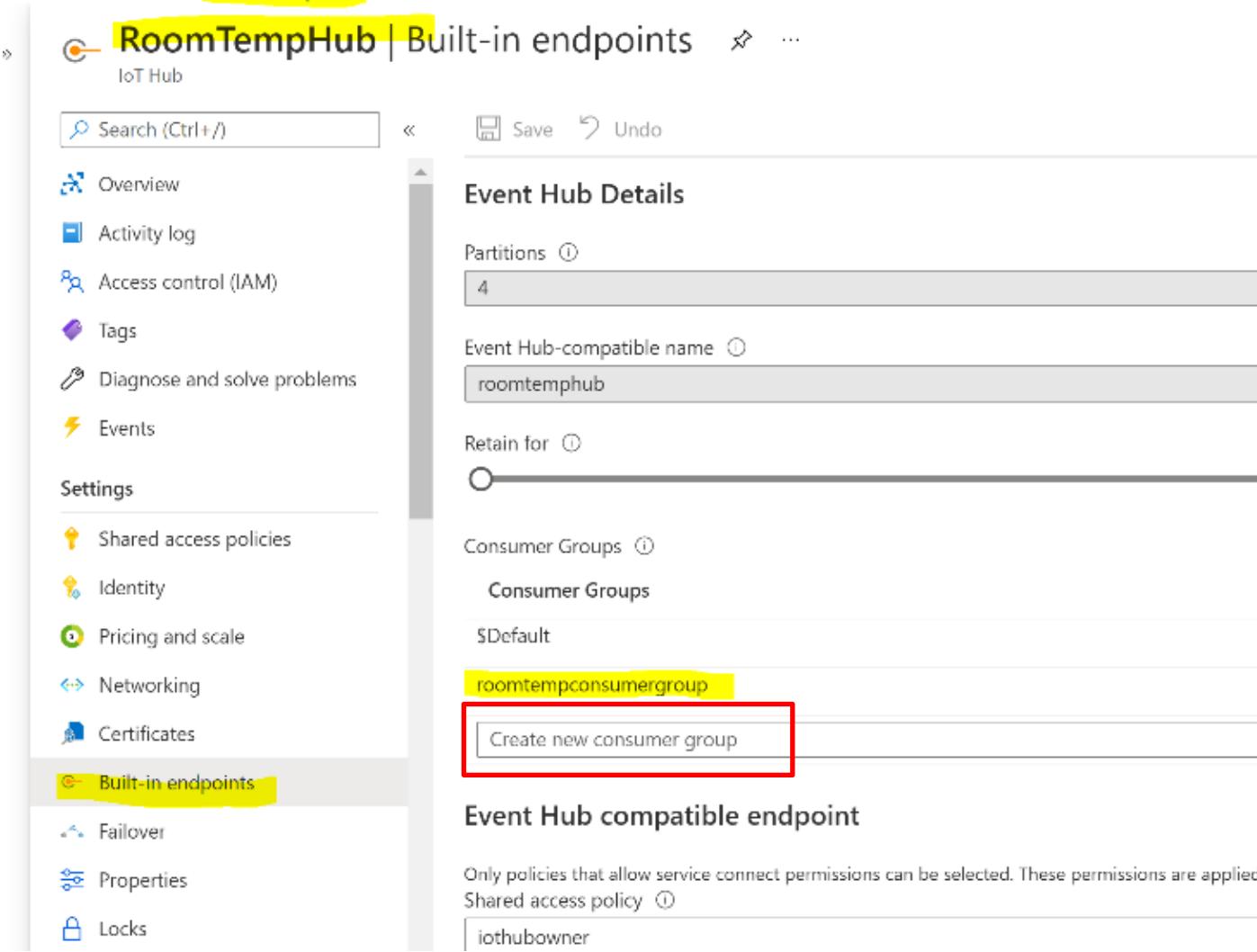
Event Hub compatible endpoint

Only policies that allow service connect permissions can be selected. These permissions are applied to Shared access policy ①

iothubowner

Built-in endpoints

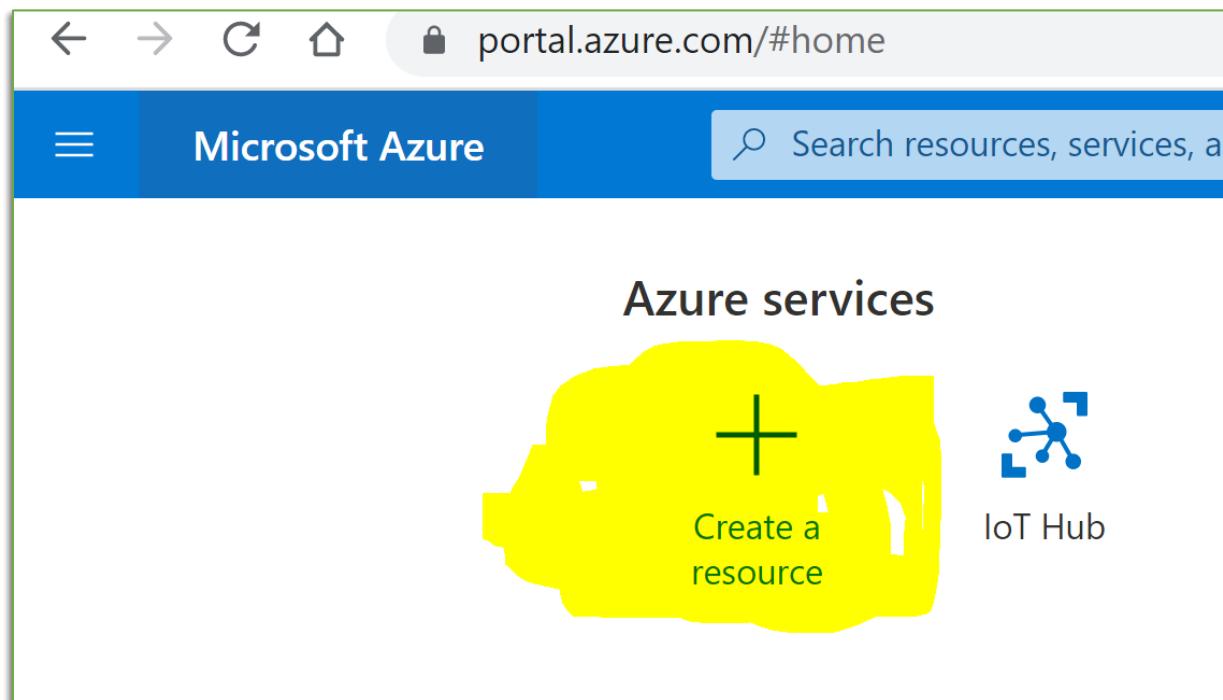
Failover Properties Locks



Create, configure, and run a Stream Analytics job

1. In the [Azure portal](#), select **Create a resource**.

Type *Stream Analytics Job* in the search box and select it from the drop-down list. On the **Stream Analytics job** overview page, select **Create**



Search: Stream Analytics job

Home >

Create a resource

Get started

Recently created

Categories

AI + Machine Learning

Analytics

Blockchain

Stream Analytics job



Windows Server 2019 Datacenter

[Create](#) | [Learn more](#)



Ubuntu Server 20.04 LTS

[Create](#) | [Learn more](#)



Microsoft Azure



Search resources, services,

Home > Create a resource >

Stream Analytics job



Microsoft



Stream Analytics job

Microsoft

★★★★★ ☆ 4.0 (185 ratings)

Create

Overview

Plans

Usage Information + Support

Reviews

Next



Microsoft Azure Search resources, services, and ...

Home > Create a resource > Stream Analytics job > New Stream Analytics job ...

i This will create a new Stream Analytics job. You will be charged according to A

Job name *****
TempDataVisualizeJob ✓

Subscription *****
Azure for Students ▼

Resource group *****
RoomTempratureResourceGroup ▼
Create new

Location *****
East US ▼

Hosting environment ⓘ
 Cloud Edge

Streaming units (1 to 192) ⓘ
 3

Secure all private data assets needed by this job in my Storage account. ⓘ

Create

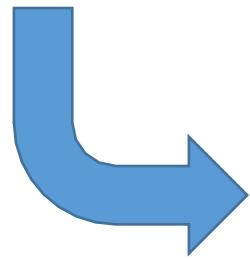
Done! Check it out: You have now the Stream Analytics job

Azure services

Create a resource

All resources

Recent resources



Microsoft Azure

Home >

All resources

Champlain College (champlain.onmicrosoft.com)

+ Create Manage view Refresh Export

Filter for any field... Subscription == all

Showing 1 to 3 of 3 records. Show hidden types

<input type="checkbox"/>	Name ↑
<input type="checkbox"/>	clusterroomtemp
<input type="checkbox"/>	RoomTempHub
<input checked="" type="checkbox"/>	TempDataVisualizeJob

 **MuratStreamAJob**
Stream Analytics job

Search (Ctrl+/)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Properties

Locks

Job topology

Inputs

Functions

Query

Outputs

Add Stream Input & Add Stream Output Add Query

Select Add stream input, then select IoT Hub from the drop-down list

The screenshot shows the Microsoft Azure Stream Analytics job configuration interface. At the top, there's a blue header bar with the Microsoft Azure logo and a search bar. Below it, the breadcrumb navigation shows 'Home > All resources > TempDataVisualizeJob'. The main area is titled 'TempDataVisualizeJob' with the subtitle 'Stream Analytics job'. On the left, there's a sidebar with 'All resources' and a list of jobs: 'clusterroomtemp', 'RoomTempHub', and 'TempDataVisualizeJob' (which is highlighted with a yellow box). The main content area has a 'Search (Ctrl+ /)' input field and a 'Job topology' section with an 'Inputs' button (also highlighted with a yellow box). A dropdown menu for 'Add stream input' is open, showing options: 'Event Hub', 'IoT Hub' (which is highlighted with a yellow box), and 'Blob storage/ADLS Gen2'. To the right of the dropdown, there's a large green octagonal button with the number '1'.

1. Open the Stream Analytics job.
2. Under **Job topology**, select **Inputs**.

Add Stream Input

IoT Hub

New input

StreamInput

Provide IoT Hub settings manually

Select IoT Hub from your subscriptions

Subscription

Azure for Students

IoT Hub * ⓘ

RoomTempHub

Consumer group * ⓘ

roomtempconsumergroup

Shared access policy name * ⓘ

service

Shared access policy key ⓘ

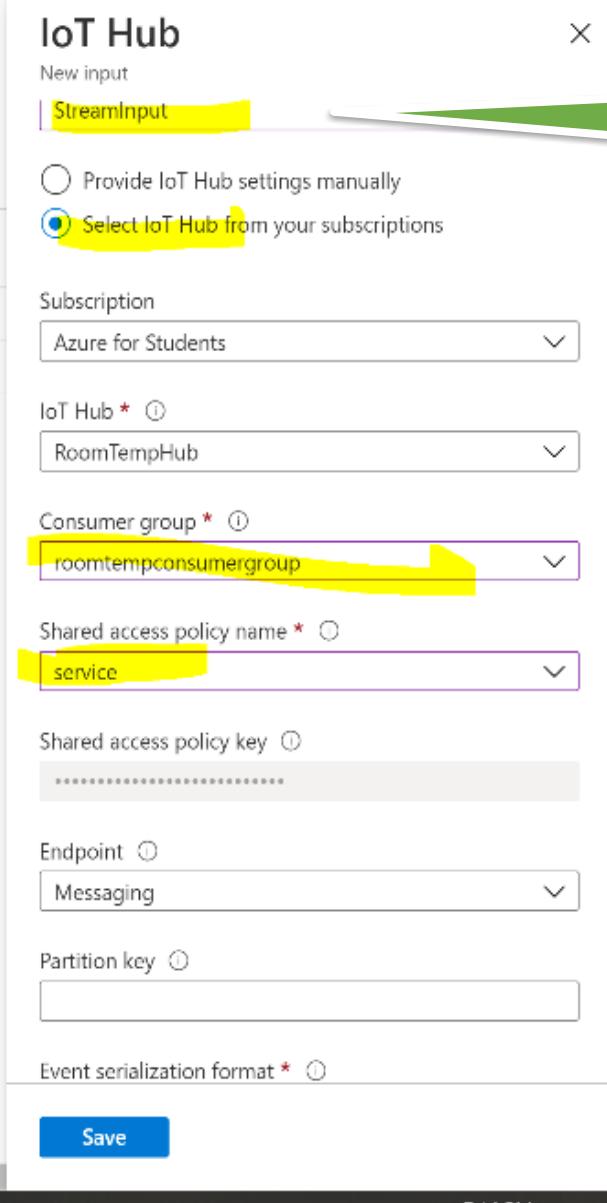
Endpoint ⓘ

Messaging

Partition key ⓘ

Event serialization format * ⓘ

Save



Remember input stream name
We will need it when we are
setting up input query

Event serialization format ⓘ

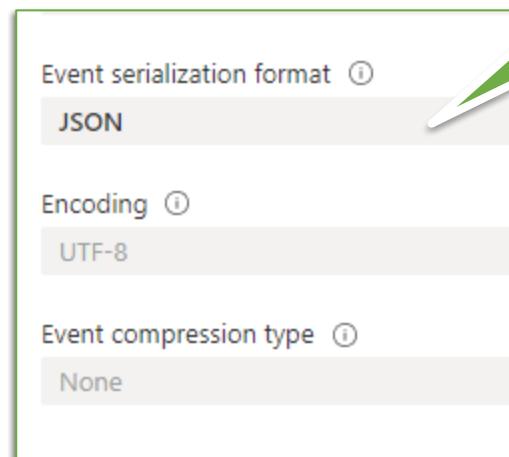
JSON

Encoding ⓘ

UTF-8

Event compression type ⓘ

None



Our code updated to send
telemetry as JSON format

2

Done! Stream Input Added

Home > MKR_StreamAnalyticsJob

MKR_StreamAnalyticsJob | Inputs

Stream Analytics job

Search (Ctrl+ /)

Add stream input Add reference input

Name	Source type	Source
StreamInput	Stream	IoT Hub

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Properties

Locks

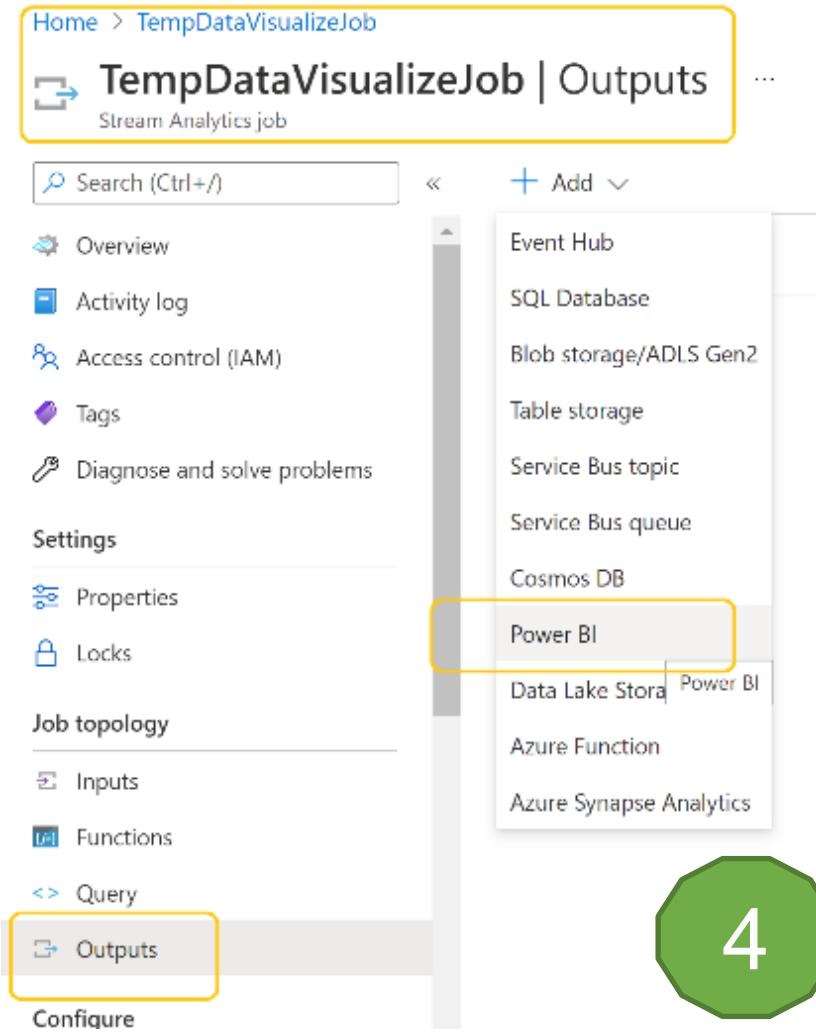
Inputs

Remember input stream name
We will need it when we are
setting up input query

3

Add a Stream Output to the Stream Analytics job

- Under **Job topology**, select **Outputs**.
- In the **Outputs** pane, select **Add**, and then select **Power BI** from the drop-down list.
- On the **Power BI – New output** pane, select **Authorize** and follow the prompts to sign in to your Power BI account.



Add a Stream Output

After you've signed in to **Power BI**, enter the following information:

- Output alias:**
A unique alias for the output.
- Group workspace:** Select your target group workspace.
- Dataset name:**
Enter a dataset name.
- Table name:**
Enter a table name.
- Authentication mode:**
Leave at the default.

Remember output stream name
We will need it when we are setting up input query

Power BI
New output

Output alias *

TempDataPowBUOutput

Provide Group workspace settings manually
 Select Group workspace from your subscriptions

Group workspace *

My workspace

Authentication mode

User token

Dataset name *

TempPowerBIDataSetName

Table name *

TempPowerBITableName

Authorize connection
You'll need to authorize with Power BI to configure your output settings.

Authorize

Don't have a Microsoft Power BI account yet?
[Sign up](#)

Save



Done! Stream Output Added

Home > MKR_StreamAnalyticsJob

MKR_StreamAnalyticsJob | Outputs

Stream Analytics job

Search (Ctrl+ /)

Add

Name	Sink
OutputPowerBI	Power BI

Remember output stream name
We will need it when we are
setting up input query

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Properties

Locks

Job topology

Inputs

Functions

Query

Outputs



55

Query – FROM and INTO update

- Test Query

The screenshot shows the Microsoft Azure Stream Analytics job configuration interface. On the left, the navigation pane includes links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Properties, Locks, Job topology, Functions, Inputs, Outputs, Configure, Environment, Storage account settings, Scale, Locale, and Event ordering. The 'Query' link is currently selected.

In the main area, the 'Inputs' section shows one input named 'MuratInputStream'. The 'Outputs' section shows one output named 'MuratOutputPowerBI'. The 'Test query' editor contains the following T-SQL code:

```
1 SELECT
2 *
3 INTO
4 MuratOutputPowerBI
5 FROM
6 MuratInputStream
```

A large blue callout box points from the right towards the 'Test query' editor. It contains three instructions:

- Under Job topology, select Query.
- Replace [YourInputAlias] with the input alias of the job.
- Replace [YourOutputAlias] with the output alias of the job.

Below the 'Test query' editor is an 'Input preview' table showing event data from 'MuratInputStream'. The columns are: messageId, deviceId, temperature, humidity, EventProcessedUtcT..., and PartitionId. The data rows are:

messageId	deviceId	temperature	humidity	EventProcessedUtcT...	PartitionId
46	"MKR WiFi 1010 Client"	26.36479	41.98602	"2021-11-04T00:35:18....	1
45	"MKR WiFi 1010 Client"	26.34648	41.89444	"2021-11-04T00:35:18....	
44	"MKR WiFi 1010 Client"	26.34648	41.82813	"2021-11-04T00:35:18....	
43	"MKR WiFi 1010 Client"	26.34648	41.85023	"2021-11-04T00:35:18....	
42	"MKR WiFi 1010 Client"	26.40141	41.94181	"2021-11-04T00:35:17....	1
41	"MKR WiFi 1010 Client"	26.3831	41.99865	"2021-11-04T00:35:17....	1
40	"MKR WiFi 1010 Client"	26.3831	42.08707	"2021-11-04T00:35:17....	1

A blue callout box points from the bottom right towards the input data table. It contains the instruction: Observe the input.

A green octagonal button in the bottom right corner contains the number 7.

START the Stream Analytics job

- In the Stream Analytics job, select **Overview**, then select **Start > Now > Start**. Once the job successfully starts, the job status changes from **Stopped** to **Running**.

The screenshot shows the Azure Stream Analytics job overview page for 'MKR_StreamAnalyticsJob'. On the left, there's a navigation bar with links like Home, Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Properties, Locks, and Job topology. The 'Overview' link is highlighted. In the center, there's a large 'Start' button with a yellow background and white text. To the right of the button, the status is shown as 'Running'. A yellow callout box with the text 'Be Patient Starting takes time' points to the status. At the bottom right, there's a green octagonal icon with the number '8' inside it. On the far right, there's a 'Query' section with the following code:

```
1 SELECT
2 *
3 INTO
4 PowerBiVisualizationOutput
5 FROM
6 PowerBiVisualizationInput
7 WHERE temperature IS NOT NULL
8
```

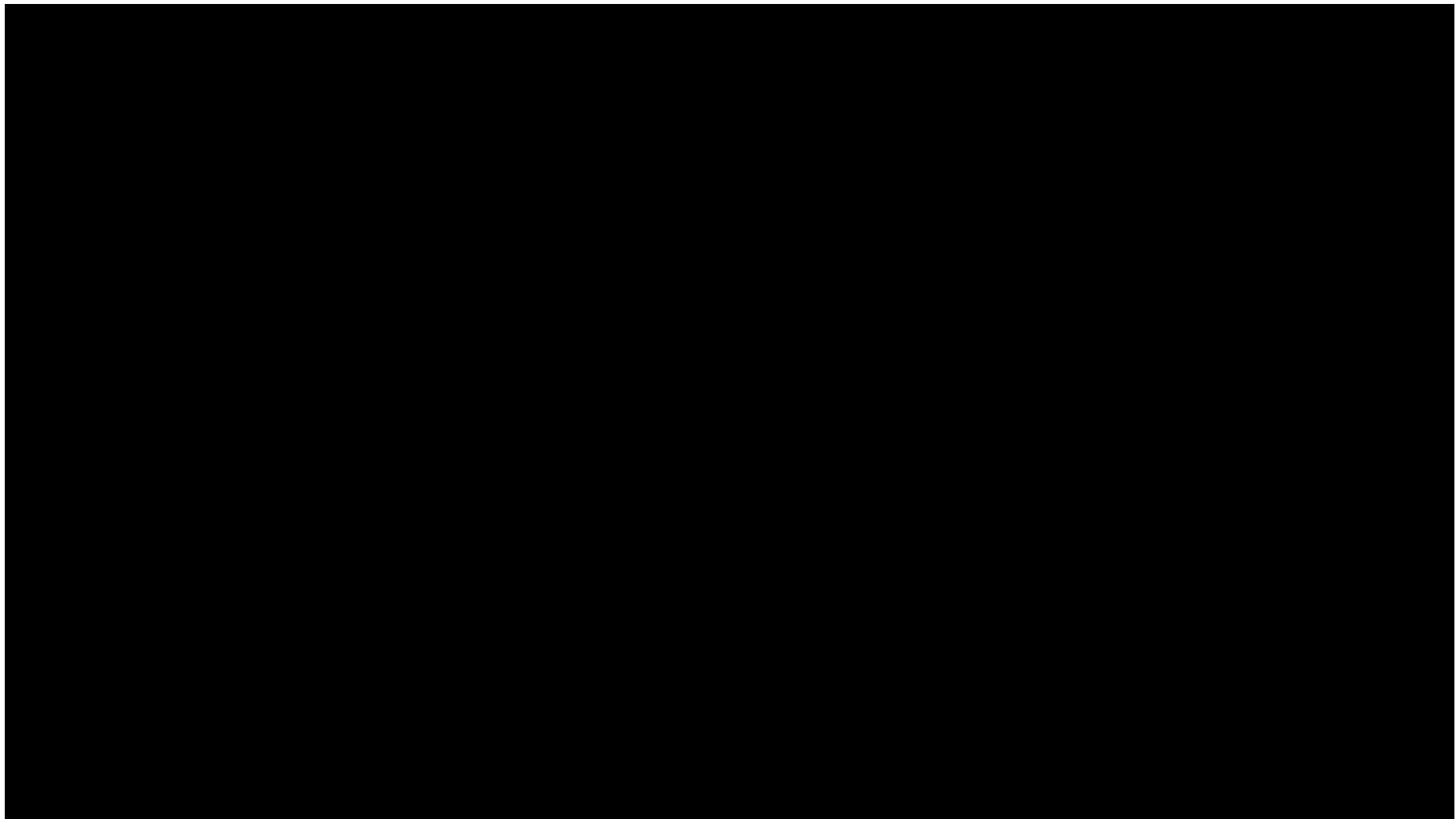
Power BI is a business analytics service by Microsoft.
Business Intelligence (BI)



PowerBI

Create Power BI report to visualize the data

What is Power BI?



<https://youtu.be/yKTSLfVGBk>

Create and publish a Power BI report to visualize the data

1. Make sure the client app is running on your device.
2. Sign in to your Power BI account and select **Power BI service** from the top menu.
 - <https://powerbi.microsoft.com/>
3. Select the workspace you used from the side menu, **My Workspace**.
4. Under the **All** tab or the **Datasets + dataflows** tab, you should see the dataset that you specified when you created the output for the Stream Analytics job.

- Hover over the dataset you created, select **More options** menu (the three dots to the right of the dataset name), and then select **Create report**.

The screenshot shows the Microsoft Power BI workspace interface. The left sidebar includes links for Home, Favorites, Recent, Create, Datasets, Goals, Apps, Shared with me, Discover, Learn, Workspaces, My workspace (which is currently selected and expanded), and Get data. The main area is titled "My workspace" and contains sections for "New", "View", "Filters", and "Search". Below these are tabs for "All", "Content", and "Datasets + dataflows", with "All" being the active tab. A table lists datasets, showing one entry: "PowerBiVisualizationDataSet" (Type: Dataset, Owner: Jimaco Brannian, Refreshed: 7/6/21, 4:15:21 PM). A context menu is open for this dataset, listing options: Create report (highlighted with a red box), Delete, Manage permissions, Quick insights, Edit, and API Info.

Name	Type	Owner	Refreshed
PowerBiVisualizationDataSet	Dataset	Jimaco Brannian	7/6/21, 4:15:21 PM

Power BI report to visualize the data

The screenshot shows the Power BI workspace interface. On the left, there's a navigation pane with links like Home, Favorites, Recent, Create, Datasets, Goals, Apps, Shared with me, Learn, Workspaces, and My workspace. Under My workspace, a report titled "TempReport" is selected. The main area displays a line chart with "temperature" on the Y-axis (ranging from 20 to 32) and "EventEnqueuedUtcTime" on the X-axis. The chart shows several sharp peaks, with one major peak reaching approximately 32 degrees around 2021-08-22T10:00:00Z. A context menu is open over the chart, showing options like Save, Save as, Print, Export to PowerPoint, Export to PDF, Download report (Preview), and Add data fields here.

On the right side, there are three floating windows:

- Visualizations:** A list of visualization icons, with the "Line chart" icon highlighted by a yellow circle.
- Fields:** A list of fields from the data source "MuratPBITableName":
 - DeviceId
 - EventEnqueuedUtcTime
 - EventProcessedUtcTime (selected)
 - Humidity (summarized)
 - IoHub
 - MessageId (summarized)
 - PartitionId
 - Temperature (summarized)
- Filters:** A list of filters:
 - Filters on this page
 - Add data fields here
 - Filters on all pages
 - Add data fields here

Two large orange arrows point towards the "Visualizations" and "Fields" windows from the bottom right, indicating they are the focus of the current discussion.

Cleanup resources

Cleaning Up Resources

- You've created a resource group, an IoT hub, a Stream Analytics job, and a dataset in Power BI.
- If you don't need the IoT hub or the other resources you created any longer, you can delete the resource group in the portal.
- Clean up Power BI resources as well too

More Detail:

<https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-live-data-visualization-in-power-bi#cleanup-resources>

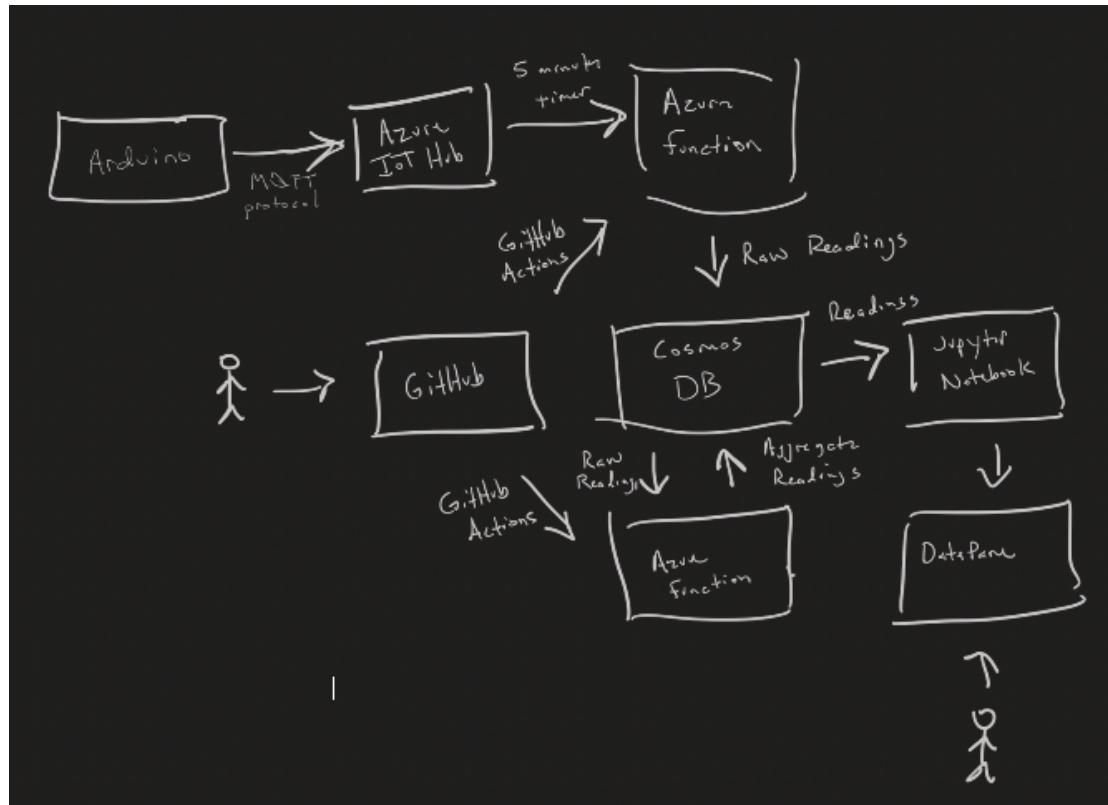


Want to Explore More

Good Tutorials:



- How can we save telemetry an Azure SQL database?
 - <https://www.mssqltips.com/sqlservertip/6335/how-to-capture-iot-data-in-azure/>
- IoT Azure Pipeline
 - <https://paul-bruffett.medium.com/iot-azure-pipeline-9725ac2b6a00>



More Application

- Visualize real-time sensor data from your Azure IoT hub in a web application
 - <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-live-data-visualization-in-web-apps>
- Tutorial: Implement IoT spatial analytics by using Azure Maps
 - <https://docs.microsoft.com/en-us/azure/azure-maps/tutorial-iot-hub-maps>
- Arduino IoT Cloud Google Sheets Integration
 - https://create.arduino.cc/projecthub/Arduino_Genuino/arduino-iot-cloud-google-sheets-integration-71b6bc

Some Terminology

Communicate with IoT Hub

- Device to cloud (D2C) messages
 - - these are messages sent from a device to IoT Hub, such as telemetry. They can then be read off the IoT Hub by your application code.
- Cloud to device (C2D) messages
 - - these are messages sent from application code, via an IoT Hub to an IoT device
- Direct method requests
 - - these are messages sent from application code via an IoT Hub to an IoT device to request that the device does something, such as control an actuator. These messages require a response so your application code can tell if it was successfully processed.
- Device twins
 - - these are JSON documents kept synchronized between the device and IoT Hub, and are used to store settings or other properties either reported by the device, or should be set on the device (known as desired) by the IoT Hub.



Extra Information: Device Provisioning Service

Azure IoT Hub Device Provisioning Service (DPS)

- Link IoT Device Provisioning Service and IoT hub so that the IoT Hub Device Provisioning Service can register devices to that hub
 - [How to link an IoT hub](#)
- Set up allocation policy, which is a IoT Hub Device Provisioning Service setting that determines how devices are assigned to an IoT hub
 - [How to set up an allocation policy](#)
- Add the device's unique security artifacts to the Device Provisioning Service
 - [How to enroll device](#)
- Start the device to allow your client application to start the registration with your Device Provisioning service
 - [How to start and verify a device](#)

QuickStart: Set up the IoT Hub Device Provisioning Service with the Azure portal

- Link the IoT hub and your Device Provisioning Service
 - <https://docs.microsoft.com/en-us/azure/iot-dps/quick-setup-auto-provision#link-the-iot-hub-and-your-device-provisioning-service>
- Azure IoT Hub Device Provisioning Service (DPS) Documentation
 - <https://docs.microsoft.com/en-us/azure/iot-dps/quick-create-simulated-device-x509>

IoT Hub DPS Demo

- View a demo of an IoT device being automatically provisioned, in a secure and scalable way, with IoT Hub Device Provisioning Service. You'll understand how you can provision thousands of IoT devices with zero-touch.
- Click to watch the video
 - <https://azure.microsoft.com/en-us/resources/videos/iot-hub-dps-demo/>
 - **Quickstart: Set up the IoT Hub Device Provisioning Service with Azure CLI**
 - <https://docs.microsoft.com/en-us/azure/iot-dps/quick-setup-auto-provision-cli>

References

- IoT Azure Pipeline
 - <https://paul-bruffett.medium.com/iot-azure-pipeline-9725ac2b6a00>
- Securely Connecting an Arduino NB 1500 to Azure IoT Hub © CC BY
 - https://create.arduino.cc/projecthub/Arduino_Genuino/securing-an-arduino-nb-1500-to-azure-iot-hub-af6470
- Securely Connecting an Arduino MKR WiFi 1010 to AWS IoT Core
 - https://create.arduino.cc/projecthub/Arduino_Genuino/securing-an-arduino-mkr-wifi-1010-to-aws-iot-core-a9f365
- Quickstart: Create a Stream Analytics job by using the Azure portal
 - <https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-quick-create-portal>
- Tutorial: Visualize real-time sensor data from Azure IoT Hub using Power BI
 - <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-live-data-visualization-in-power-bi>
- Some more in the notes section of these slides.

Let us dare!

Thanks!

Any questions?

**These could be useful
Most likely you will not
need for this class**

Adding a device PAUSE!

First we need device certification

Home > RoomTempHub >

Create a device ...

 Find Certified for Azure IoT devices in the Device Catalog

Device ID * ⓘ
MyMKRWiFi1010

Authentication type ⓘ
Symmetric key X.509 Self-Signed X.509 CA Signed

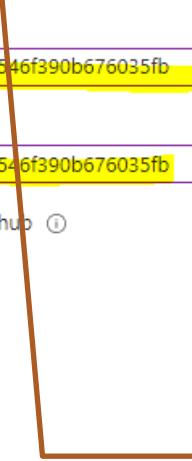
Primary Thumbprint * ⓘ
630a8758e2b86c29aef35f56546f390b676035fb

Secondary Thumbprint * ⓘ
630a8758e2b86c29aef35f56546f390b676035fb

Connect this device to an IoT hub ⓘ
Enable Disable

Parent device ⓘ
No parent device Set a parent device

Save



- IoT device connects to your IoT Hub using Certification

Pause Here!

- To complete this we need to jump to next steps to **create our certification**





Creating Certification for your Arduino

X.509 certificates for authentication

- Azure IoT Hub allows devices that connect using the MQTT protocol and use X.509 certificates for authentication.
- We'll use a sketch to generate a self signed **X.509 Certificate** on the board and then add the SHA1 of this certificate to Azure IoT Hub portal.
- Every Arduino MKR board with on-board connectivity is equipped with a Microchip crypto element.

This crypto element can be used to securely generate and store a 256-bit ECC (Elliptic Curve Cryptography) key.

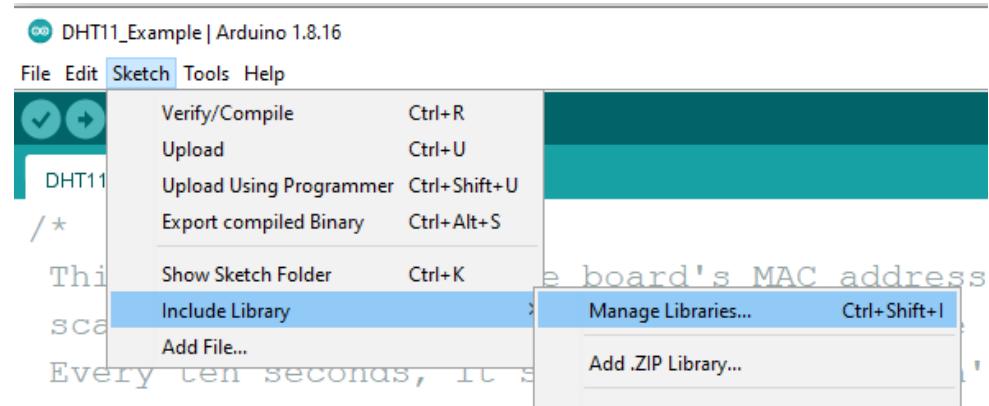
Libraries to Install

We will install the Arduino libraries that will be used, using the Arduino IDE's library manager.

Open the *Sketch -> Include Library -> Manage Libraries...* menu, search for and individually install each of the following libraries:

Libraries

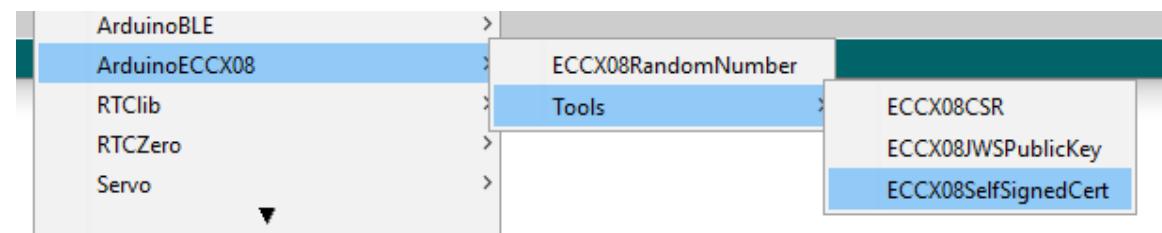
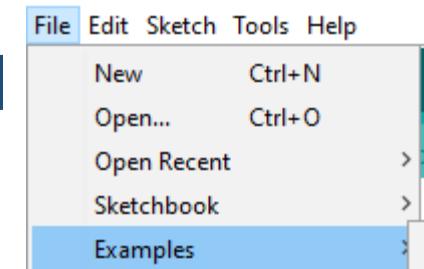
1. ArduinoBearSSL
2. ArduinoECCX08
3. ArduinoMqttClient
4. Arduino Cloud Provider Examples



X.509 certificates for authentication

The self signed certificate can be generated using an example sketch from the ArduinoECCX08 library.

- “Open” the sketch in the Arduino IDE using the
File \ Examples \ ArduinoECCX08 \ Tools \ ECCX08SelfSignedCert
- Click the "Upload" button to build and upload the sketch to your board, then open the Serial Monitor.



- After this, you will be prompted for information to include in the self signed certificate, including the **issue year**, month, **day** and **hour** of the certificate and the **expiry period in years**.
- For this tutorial we'll be using
- **slot 0** to generate and store the private key used to sign the self signed certificate (slots 1 to 4 can be used to generate and store additional private keys if needed) - then
- **slot 8** will be used to store the issue and expiry date of the certificate along with its signature.

Serial Port - Enter the data requested

ECCX08 Serial Number = 012385338ACB4748EE

Hi there, in order to generate a new self signed cert for your board, we'll need the following information ...

Please enter the issue year of the certificate? (2000 - 2031) [2019]: **2021**

Please enter the issue month of the certificate? (1 - 12) [1]: **8**

Please enter the issue day of the certificate? (1 - 31) [1]: **18**

Please enter the issue hour of the certificate? (0 - 23) [0]: **11**

Please enter how many years the certificate is valid for? (1 - 31) [31]: **11**

What slot would you like to use for the private key? (0 - 4) [0]: **0**

What slot would you like to use for storage? (8 - 15) [8]: **8**

Would you like to generate a new private key? (Y/n) [Y]: **Y**

Here's your self signed cert, enjoy!

Serial Monitor - Executing

DON'T CLOSE

```
COM6
ECCX08 Serial Number = 0123324CA129AFBEEE

Hi there, in order to generate a new self signed cert for your board, we'll need

Please enter the issue year of the certificate? (2000 - 2031) [2019]: 2021
Please enter the issue month of the certificate? (1 - 12) [1]: 11
Please enter the issue day of the certificate? (1 - 31) [1]: 1
Please enter the issue hour of the certificate? (0 - 23) [0]: 1
Please enter how many years the certificate is valid for? (1 - 31) [31]: 10
What slot would you like to use for the private key? (0 - 4) [0]: 1
What slot would you like to use for storage? (8 - 15) [8]: 11
Would you like to generate a new private key? (Y/n) [Y]: Y

Here's your self signed cert, enjoy!

-----BEGIN CERTIFICATE-----
MIIBKjCB0aADAgECAgEBMAoGCCqGSM49BAMCMB0xGzAZBgNVBAMTEjAxMjMzMjRDQTEyOUFGQkVF
RTAeFw0yMTExMDEwMTAwMDBaFw0zMTExMDEwMTAwMDBaMB0xGzAZBgNVBAMTEjAxMjMzMjRDQTEy
OUFGQkVFRTBZMBMGBByqGSM49AgEGCCqGSM49AwEHA0IABJB/FSRXdcI4KcQwPgaxiL2pVu2LLVvY
7IkwoLD7m/pwiIn57EouI5ossuK52sZ1GAlmN8IBBI0I5gRjs13+flojAjAAAMAoGCCqGSM49BAMC
A0gAMEUCIGfwZHHVGAgN8EKhQmy7vCDDfYcr5eqW6utP+szGnjTUAiEA1sC1jiqZEuNg5E5FEuE0
hhMtZ/M70GNShzOqiOoVa4=
-----END CERTIFICATE-----

SHA1: e2ae6f4f8dea237ec47cc6ceb5bf013f19604663
```

Autoscroll Show timestamp Newline 9600 baud Clear output

Copy everything
Save into a text file
for further
processing
Use Notepad++

Note: Since the private key is generated inside the crypto element it never leaves the device and is stored securely and cannot be read.

SHA1 KEY

Sample Run Output

-----BEGIN CERTIFICATE-----

MIIBKjCB0aADAgECAgEBMAoGCCqGSM49BAMCMB0xGzAZBgNVBAMTEjAxMjM4NTMzOEFQjQ3NDhF
RTAeFw0yMTA4MTgxMTAwMDBaFw0zMjA4MTgxMTAwMDBaMB0xGzAZBgNVBAMTEjAxMjM4NTMzOEFQ
jQ3NDhFRTBZMBMGBYqGSM49AgEGCCqGSM49AwEHA0IABDJ7ddEvDvp7eVK6gp2dCyGNMN0N0re0
WpBofjzOMcOYbwGTiOZeQtoUghi41le3avK1Im9WV67/OnIystgeSFWjAjAAMAoGCCqGSM49BAMC
A0gAMEUCIQCObICMKzAMcASi0maa/P9hW8UXvsurk2whrljxQEbdVgIgeFrzdnFjWhJR7BFyKBek
TlqW0q7BYfTSQkd3/YIg7cg=DONT

-----END CERTIFICATE-----

From Your Serial Port:

Save the above text in orange color to **ECCX08cert.pem**
(You can name the file anything but keep the extension)

SHA1: **630a8758e2b86c29aef35f56546f390b676035DONTfb**

We need this blue key while adding our device to Azure Cloud



**Go back to Azure to
Add our Device**

Create a device

Home > RoomTempHub >

 Create a device ...



Find Certified for Azure IoT devices in the Device Catalog

Device ID * ⓘ

MyMKRWiFi1010

Authentication type ⓘ

Symmetric key X.509 Self-Signed X.509 CA Signed

Primary Thumbprint * ⓘ

630a8758e2b86c29aef35f56546f390b676035fb

Secondary Thumbprint * ⓘ

630a8758e2b86c29aef35f56546f390b676035fb

Connect this device to an IoT hub ⓘ

Enable Disable

Parent device ⓘ

No parent device

[Set a parent device](#)

[Save](#)

- When your IoT device connects to your IoT Hub using Certification

SHA1 KEY

Device Added to Azure Cloud

The screenshot shows the Azure IoT Hub management interface for the 'RoomTempHub' IoT Hub. The left sidebar contains navigation links for Home, RoomTempHub, Settings (Shared access policies, Identity, Pricing and scale, Networking, Certificates, Built-in endpoints, Failover, Properties, Locks), Explorers (Query explorer, IoT devices), and a search bar. The main content area displays the 'IoT devices' page with a search bar, a 'Find devices' button, and a table listing two devices: 'MyArduinoMKR1010-DeviceID' and 'MyMKRWiFi1010'. The table columns include Device ID, Status, Last Status Update, Authentication Type, and Cloud t... (partially visible). The 'SelfSigned' authentication type for the second device is highlighted with a yellow box.

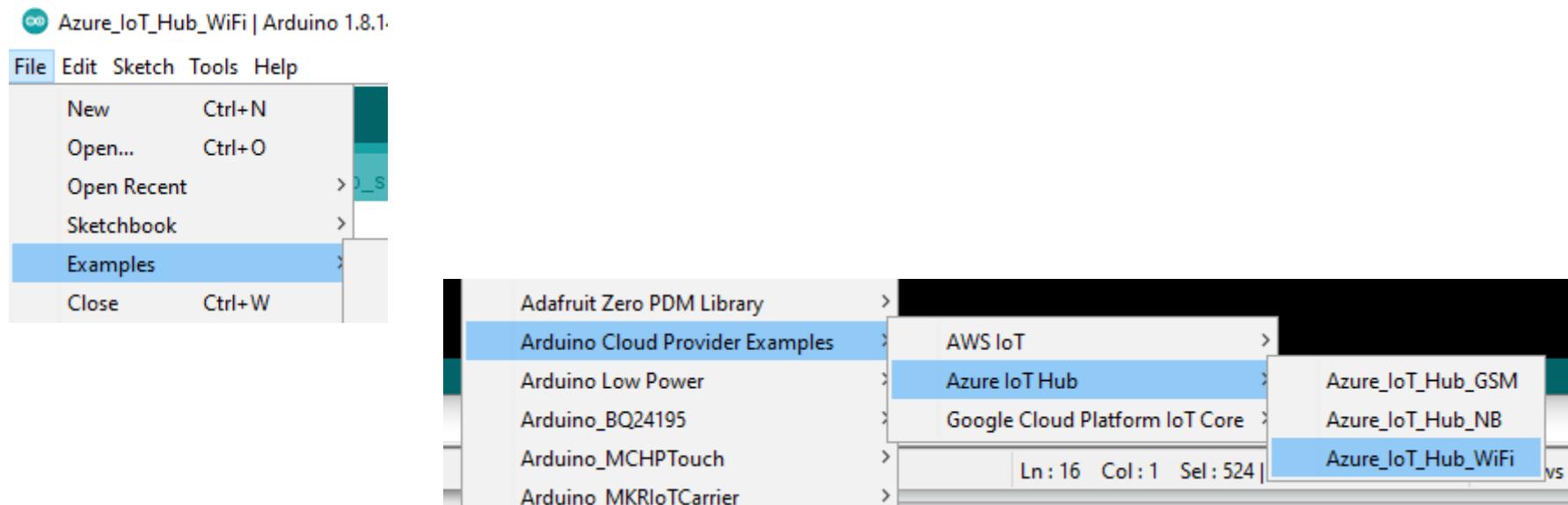
Device ID	Status	Last Status Update	Authentication Type	Cloud t...
MyArduinoMKR1010-DeviceID	Enabled	--	Sas	0
MyMKRWiFi1010	Enabled	--	SelfSigned	0

Code Part at Arduino

Open This Example Code

Open

Azure Examples /
Arduino Cloud Provider Examples /
Azure IoT Hub /
Azure IoT Hub WiFi



Visual Studio Code

The screenshot shows the Visual Studio Code interface with the following components:

- File Explorer (Left):** Shows files in the workspace:
 - platformio.ini
 - main.cpp
 - arduino_secrets.h
- Open Editors (Top):** Shows the current files being edited:
 - platformio.ini
 - .gitignore
 - main.cpp
 - src (selected)
 - arduino_secrets.h
- Code Editor (Right):** Displays the main.cpp file content, which includes code for WiFi and MQTT connections.

```
Test > src > C main.cpp > ...
27 // Enter your sensitive data in arduino_secrets.h
28 const char ssid[] = SECRET_SSID;
29 const char pass[] = SECRET_PASS;
30 const char broker[] = SECRET_BROKER;
31 String deviceId = SECRET_DEVICE_ID;
32
33 WiFiClient wifiClient; // Used for the TCP socket connection
34 BearSSLClient sslClient(wifiClient); // Used for SSL/TLS connection, integra
35 MqttClient mqttClient(sslClient);
36
37 unsigned long lastMillis = 0;
38
39 unsigned long getTime() {
40     // get the current time from the WiFi module
41     return WiFi.getTime();
42 }
43
44
45 void connectWiFi() {
46     Serial.print("Attempting to connect to SSID: "):
```
- Terminal (Bottom):** Shows the output of the terminal commands:

```
--- More details at http://bit.ly/pio-monitor-filters
--- Miniterm on COM4 9600,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
Attempting to connect to SSID: GungorGuest
You're connected to the network

Attempting to MQTT broker: RoomTempHub.azure-devices.net
.-2

You're connected to the MQTT broker

Publishing message
Publishing message
Publishing message
```

My Sample Run at Serial Monitor

```
COM3
| Send
Attempting to connect to SSID: ChamplainPSK
You're connected to the network

Time:0
Attempting to MQTT broker: RoomTempHub.azure-devices.net
Again Time:0
.-2
Again Time:1629496136

You're connected to the MQTT broker

Publishing message
Publishing message
```

#define MQTT_CONNECTION_REFUSED	-2
#define MQTT_CONNECTION_TIMEOUT	-1
#define MQTT_SUCCESS	0
#define MQTT_UNACCEPTABLE_PROTOCOL_VERSION	1
#define MQTT_IDENTIFIER_REJECTED	2
#define MQTT_SERVER_UNAVAILABLE	3
#define MQTT_BAD_USER_NAME_OR_PASSWORD	4
#define MQTT_NOT_AUTHORIZED	5

Autoscroll Show timestamp Both NL & CR 9600 baud Clear output

Using JSON format



platformio.ini

Publishing message

```
{  
    "messageId":968,  
    "deviceId":"MKR WiFi 1010 Client",  
    "temperature":28.24657,  
    "humidity":70.85092  
}
```



arduino_secrets.h



main.cpp

```
File Edit Selection View Go Run Terminal Help mainapp - Azure IoT Hub WiFi (Workspace) - Visual Studio Code  
EXPLORER main.cpp X platformio.ini C arduino_secrets.h  
OPEN EDITORS main.cpp src platformio.ini arduino_secrets.h  
AZURE IOT HUB WiFi (WORKSPACE) mainapp - Azure IoT Hub WiFi-VSCODE...  
Azure IoT Hub WiFi-VSCODE...  
> .ipd  
> vscode  
> include  
> lib  
> src  
> C arduino_secrets.h  
> main.cpp  
> test  
> .gitignore  
{ Azure IoT Hub WiFiCode...  
> platformioini  
& SampleRun.PNG  
95 }  
96  
97 int messageId = 1;  
98 void publishTemperatureHumidity() {  
99     Serial.println("Publishing message");  
100  
101     // Read Sensor Data  
102     float temperature = carrier.Env.readTemperature();  
103     float humidity = carrier.Env.readHumidity();  
104  
105     DynamicJsonDocument doc(1024);  
106     doc["messageId"] = messageId++;  
107     doc["deviceId"] = "MKR WiFi 1010 Client";  
108     doc["temperature"] = temperature;  
109     doc["humidity"] = humidity;  
110  
111     String telemetry;  
112     serializeJson(doc, telemetry);  
113  
114     Serial.println(telemetry.c_str());  
115  
116     // send message, the Print interface can be used to set the message contents  
117     mqttClient.beginMessage("devices/" + deviceId + "/messages/events/");  
118     mqttClient.print(telemetry.c_str());  
119     mqttClient.endMessage();  
120  
121  
122     #include <PubSubClient.h>  
123     #include <WiFiClient.h>  
124  
125     WiFiClient client;  
126     PubSubClient mqttClient(client, "127.0.0.1", 1883);  
127  
128     if (!mqttClient.connect("Arduino WiFi")) {  
129         Serial.println("MQTT connection failed");  
130     } else {  
131         Serial.println("MQTT connected");  
132         mqttClient.publish("temperature", String(temperature));  
133         mqttClient.publish("humidity", String(humidity));  
134     }  
135  
136     delay(1000);  
137  
138     // Publish a few more messages to demonstrate the publish loop  
139     for (int i = 0; i < 5; i++) {  
140         publishTemperatureHumidity();  
141         delay(1000);  
142     }  
143  
144     // Disconnect from the MQTT broker  
145     mqttClient.disconnect();  
146  
147     // Wait for a few seconds before disconnecting from WiFi  
148     delay(5000);  
149  
150     // Disconnect from WiFi  
151     WiFi.disconnect();  
152  
153     // Turn off the LED  
154     digitalWrite(LED_BUILTIN, LOW);  
155  
156     // Turn off the WiFi module  
157     WiFi.modulePowerOff();  
158  
159     // Turn off the entire Arduino board  
160     powerManagement.powerDown();  
161  
162     // Turn on the WiFi module again  
163     WiFi.modulePowerOn();  
164  
165     // Turn on the LED  
166     digitalWrite(LED_BUILTIN, HIGH);  
167  
168     // Turn on the WiFi module again  
169     WiFi.modulePowerOn();  
170  
171     // Turn on the entire Arduino board  
172     powerManagement.powerUp();  
173  
174     // Turn off the WiFi module again  
175     WiFi.modulePowerOff();  
176  
177     // Turn off the LED  
178     digitalWrite(LED_BUILTIN, LOW);  
179  
180     // Turn off the WiFi module again  
181     WiFi.modulePowerOff();  
182  
183     // Turn off the entire Arduino board  
184     powerManagement.powerDown();  
185  
186     // Turn on the WiFi module again  
187     WiFi.modulePowerOn();  
188  
189     // Turn on the LED  
190     digitalWrite(LED_BUILTIN, HIGH);  
191  
192     // Turn on the WiFi module again  
193     WiFi.modulePowerOn();  
194  
195     // Turn on the entire Arduino board  
196     powerManagement.powerUp();  
197  
198     // Turn off the WiFi module again  
199     WiFi.modulePowerOff();  
200  
201     // Turn off the LED  
202     digitalWrite(LED_BUILTIN, LOW);  
203  
204     // Turn off the WiFi module again  
205     WiFi.modulePowerOff();  
206  
207     // Turn off the entire Arduino board  
208     powerManagement.powerDown();  
209  
210     // Turn on the WiFi module again  
211     WiFi.modulePowerOn();  
212  
213     // Turn on the LED  
214     digitalWrite(LED_BUILTIN, HIGH);  
215  
216     // Turn on the WiFi module again  
217     WiFi.modulePowerOn();  
218  
219     // Turn on the entire Arduino board  
220     powerManagement.powerUp();  
221  
222     // Turn off the WiFi module again  
223     WiFi.modulePowerOff();  
224  
225     // Turn off the LED  
226     digitalWrite(LED_BUILTIN, LOW);  
227  
228     // Turn off the WiFi module again  
229     WiFi.modulePowerOff();  
230  
231     // Turn off the entire Arduino board  
232     powerManagement.powerDown();  
233  
234     // Turn on the WiFi module again  
235     WiFi.modulePowerOn();  
236  
237     // Turn on the LED  
238     digitalWrite(LED_BUILTIN, HIGH);  
239  
240     // Turn on the WiFi module again  
241     WiFi.modulePowerOn();  
242  
243     // Turn on the entire Arduino board  
244     powerManagement.powerUp();  
245  
246     // Turn off the WiFi module again  
247     WiFi.modulePowerOff();  
248  
249     // Turn off the LED  
250     digitalWrite(LED_BUILTIN, LOW);  
251  
252     // Turn off the WiFi module again  
253     WiFi.modulePowerOff();  
254  
255     // Turn off the entire Arduino board  
256     powerManagement.powerDown();  
257  
258     // Turn on the WiFi module again  
259     WiFi.modulePowerOn();  
260  
261     // Turn on the LED  
262     digitalWrite(LED_BUILTIN, HIGH);  
263  
264     // Turn on the WiFi module again  
265     WiFi.modulePowerOn();  
266  
267     // Turn on the entire Arduino board  
268     powerManagement.powerUp();  
269  
270     // Turn off the WiFi module again  
271     WiFi.modulePowerOff();  
272  
273     // Turn off the LED  
274     digitalWrite(LED_BUILTIN, LOW);  
275  
276     // Turn off the WiFi module again  
277     WiFi.modulePowerOff();  
278  
279     // Turn off the entire Arduino board  
280     powerManagement.powerDown();  
281  
282     // Turn on the WiFi module again  
283     WiFi.modulePowerOn();  
284  
285     // Turn on the LED  
286     digitalWrite(LED_BUILTIN, HIGH);  
287  
288     // Turn on the WiFi module again  
289     WiFi.modulePowerOn();  
290  
291     // Turn on the entire Arduino board  
292     powerManagement.powerUp();  
293  
294     // Turn off the WiFi module again  
295     WiFi.modulePowerOff();  
296  
297     // Turn off the LED  
298     digitalWrite(LED_BUILTIN, LOW);  
299  
300     // Turn off the WiFi module again  
301     WiFi.modulePowerOff();  
302  
303     // Turn off the entire Arduino board  
304     powerManagement.powerDown();  
305  
306     // Turn on the WiFi module again  
307     WiFi.modulePowerOn();  
308  
309     // Turn on the LED  
310     digitalWrite(LED_BUILTIN, HIGH);  
311  
312     // Turn on the WiFi module again  
313     WiFi.modulePowerOn();  
314  
315     // Turn on the entire Arduino board  
316     powerManagement.powerUp();  
317  
318     // Turn off the WiFi module again  
319     WiFi.modulePowerOff();  
320  
321     // Turn off the LED  
322     digitalWrite(LED_BUILTIN, LOW);  
323  
324     // Turn off the WiFi module again  
325     WiFi.modulePowerOff();  
326  
327     // Turn off the entire Arduino board  
328     powerManagement.powerDown();  
329  
330     // Turn on the WiFi module again  
331     WiFi.modulePowerOn();  
332  
333     // Turn on the LED  
334     digitalWrite(LED_BUILTIN, HIGH);  
335  
336     // Turn on the WiFi module again  
337     WiFi.modulePowerOn();  
338  
339     // Turn on the entire Arduino board  
340     powerManagement.powerUp();  
341  
342     // Turn off the WiFi module again  
343     WiFi.modulePowerOff();  
344  
345     // Turn off the LED  
346     digitalWrite(LED_BUILTIN, LOW);  
347  
348     // Turn off the WiFi module again  
349     WiFi.modulePowerOff();  
350  
351     // Turn off the entire Arduino board  
352     powerManagement.powerDown();  
353  
354     // Turn on the WiFi module again  
355     WiFi.modulePowerOn();  
356  
357     // Turn on the LED  
358     digitalWrite(LED_BUILTIN, HIGH);  
359  
360     // Turn on the WiFi module again  
361     WiFi.modulePowerOn();  
362  
363     // Turn on the entire Arduino board  
364     powerManagement.powerUp();  
365  
366     // Turn off the WiFi module again  
367     WiFi.modulePowerOff();  
368  
369     // Turn off the LED  
370     digitalWrite(LED_BUILTIN, LOW);  
371  
372     // Turn off the WiFi module again  
373     WiFi.modulePowerOff();  
374  
375     // Turn off the entire Arduino board  
376     powerManagement.powerDown();  
377  
378     // Turn on the WiFi module again  
379     WiFi.modulePowerOn();  
380  
381     // Turn on the LED  
382     digitalWrite(LED_BUILTIN, HIGH);  
383  
384     // Turn on the WiFi module again  
385     WiFi.modulePowerOn();  
386  
387     // Turn on the entire Arduino board  
388     powerManagement.powerUp();  
389  
390     // Turn off the WiFi module again  
391     WiFi.modulePowerOff();  
392  
393     // Turn off the LED  
394     digitalWrite(LED_BUILTIN, LOW);  
395  
396     // Turn off the WiFi module again  
397     WiFi.modulePowerOff();  
398  
399     // Turn off the entire Arduino board  
400     powerManagement.powerDown();  
401  
402     // Turn on the WiFi module again  
403     WiFi.modulePowerOn();  
404  
405     // Turn on the LED  
406     digitalWrite(LED_BUILTIN, HIGH);  
407  
408     // Turn on the WiFi module again  
409     WiFi.modulePowerOn();  
410  
411     // Turn on the entire Arduino board  
412     powerManagement.powerUp();  
413  
414     // Turn off the WiFi module again  
415     WiFi.modulePowerOff();  
416  
417     // Turn off the LED  
418     digitalWrite(LED_BUILTIN, LOW);  
419  
420     // Turn off the WiFi module again  
421     WiFi.modulePowerOff();  
422  
423     // Turn off the entire Arduino board  
424     powerManagement.powerDown();  
425  
426     // Turn on the WiFi module again  
427     WiFi.modulePowerOn();  
428  
429     // Turn on the LED  
430     digitalWrite(LED_BUILTIN, HIGH);  
431  
432     // Turn on the WiFi module again  
433     WiFi.modulePowerOn();  
434  
435     // Turn on the entire Arduino board  
436     powerManagement.powerUp();  
437  
438     // Turn off the WiFi module again  
439     WiFi.modulePowerOff();  
440  
441     // Turn off the LED  
442     digitalWrite(LED_BUILTIN, LOW);  
443  
444     // Turn off the WiFi module again  
445     WiFi.modulePowerOff();  
446  
447     // Turn off the entire Arduino board  
448     powerManagement.powerDown();  
449  
450     // Turn on the WiFi module again  
451     WiFi.modulePowerOn();  
452  
453     // Turn on the LED  
454     digitalWrite(LED_BUILTIN, HIGH);  
455  
456     // Turn on the WiFi module again  
457     WiFi.modulePowerOn();  
458  
459     // Turn on the entire Arduino board  
460     powerManagement.powerUp();  
461  
462     // Turn off the WiFi module again  
463     WiFi.modulePowerOff();  
464  
465     // Turn off the LED  
466     digitalWrite(LED_BUILTIN, LOW);  
467  
468     // Turn off the WiFi module again  
469     WiFi.modulePowerOff();  
470  
471     // Turn off the entire Arduino board  
472     powerManagement.powerDown();  
473  
474     // Turn on the WiFi module again  
475     WiFi.modulePowerOn();  
476  
477     // Turn on the LED  
478     digitalWrite(LED_BUILTIN, HIGH);  
479  
480     // Turn on the WiFi module again  
481     WiFi.modulePowerOn();  
482  
483     // Turn on the entire Arduino board  
484     powerManagement.powerUp();  
485  
486     // Turn off the WiFi module again  
487     WiFi.modulePowerOff();  
488  
489     // Turn off the LED  
490     digitalWrite(LED_BUILTIN, LOW);  
491  
492     // Turn off the WiFi module again  
493     WiFi.modulePowerOff();  
494  
495     // Turn off the entire Arduino board  
496     powerManagement.powerDown();  
497  
498     // Turn on the WiFi module again  
499     WiFi.modulePowerOn();  
500  
501     // Turn on the LED  
502     digitalWrite(LED_BUILTIN, HIGH);  
503  
504     // Turn on the WiFi module again  
505     WiFi.modulePowerOn();  
506  
507     // Turn on the entire Arduino board  
508     powerManagement.powerUp();  
509  
510     // Turn off the WiFi module again  
511     WiFi.modulePowerOff();  
512  
513     // Turn off the LED  
514     digitalWrite(LED_BUILTIN, LOW);  
515  
516     // Turn off the WiFi module again  
517     WiFi.modulePowerOff();  
518  
519     // Turn off the entire Arduino board  
520     powerManagement.powerDown();  
521  
522     // Turn on the WiFi module again  
523     WiFi.modulePowerOn();  
524  
525     // Turn on the LED  
526     digitalWrite(LED_BUILTIN, HIGH);  
527  
528     // Turn on the WiFi module again  
529     WiFi.modulePowerOn();  
530  
531     // Turn on the entire Arduino board  
532     powerManagement.powerUp();  
533  
534     // Turn off the WiFi module again  
535     WiFi.modulePowerOff();  
536  
537     // Turn off the LED  
538     digitalWrite(LED_BUILTIN, LOW);  
539  
540     // Turn off the WiFi module again  
541     WiFi.modulePowerOff();  
542  
543     // Turn off the entire Arduino board  
544     powerManagement.powerDown();  
545  
546     // Turn on the WiFi module again  
547     WiFi.modulePowerOn();  
548  
549     // Turn on the LED  
550     digitalWrite(LED_BUILTIN, HIGH);  
551  
552     // Turn on the WiFi module again  
553     WiFi.modulePowerOn();  
554  
555     // Turn on the entire Arduino board  
556     powerManagement.powerUp();  
557  
558     // Turn off the WiFi module again  
559     WiFi.modulePowerOff();  
560  
561     // Turn off the LED  
562     digitalWrite(LED_BUILTIN, LOW);  
563  
564     // Turn off the WiFi module again  
565     WiFi.modulePowerOff();  
566  
567     // Turn off the entire Arduino board  
568     powerManagement.powerDown();  
569  
570     // Turn on the WiFi module again  
571     WiFi.modulePowerOn();  
572  
573     // Turn on the LED  
574     digitalWrite(LED_BUILTIN, HIGH);  
575  
576     // Turn on the WiFi module again  
577     WiFi.modulePowerOn();  
578  
579     // Turn on the entire Arduino board  
580     powerManagement.powerUp();  
581  
582     // Turn off the WiFi module again  
583     WiFi.modulePowerOff();  
584  
585     // Turn off the LED  
586     digitalWrite(LED_BUILTIN, LOW);  
587  
588     // Turn off the WiFi module again  
589     WiFi.modulePowerOff();  
590  
591     // Turn off the entire Arduino board  
592     powerManagement.powerDown();  
593  
594     // Turn on the WiFi module again  
595     WiFi.modulePowerOn();  
596  
597     // Turn on the LED  
598     digitalWrite(LED_BUILTIN, HIGH);  
599  
600     // Turn on the WiFi module again  
601     WiFi.modulePowerOn();  
602  
603     // Turn on the entire Arduino board  
604     powerManagement.powerUp();  
605  
606     // Turn off the WiFi module again  
607     WiFi.modulePowerOff();  
608  
609     // Turn off the LED  
610     digitalWrite(LED_BUILTIN, LOW);  
611  
612     // Turn off the WiFi module again  
613     WiFi.modulePowerOff();  
614  
615     // Turn off the entire Arduino board  
616     powerManagement.powerDown();  
617  
618     // Turn on the WiFi module again  
619     WiFi.modulePowerOn();  
620  
621     // Turn on the LED  
622     digitalWrite(LED_BUILTIN, HIGH);  
623  
624     // Turn on the WiFi module again  
625     WiFi.modulePowerOn();  
626  
627     // Turn on the entire Arduino board  
628     powerManagement.powerUp();  
629  
630     // Turn off the WiFi module again  
631     WiFi.modulePowerOff();  
632  
633     // Turn off the LED  
634     digitalWrite(LED_BUILTIN, LOW);  
635  
636     // Turn off the WiFi module again  
637     WiFi.modulePowerOff();  
638  
639     // Turn off the entire Arduino board  
640     powerManagement.powerDown();  
641  
642     // Turn on the WiFi module again  
643     WiFi.modulePowerOn();  
644  
645     // Turn on the LED  
646     digitalWrite(LED_BUILTIN, HIGH);  
647  
648     // Turn on the WiFi module again  
649     WiFi.modulePowerOn();  
650  
651     // Turn on the entire Arduino board  
652     powerManagement.powerUp();  
653  
654     // Turn off the WiFi module again  
655     WiFi.modulePowerOff();  
656  
657     // Turn off the LED  
658     digitalWrite(LED_BUILTIN, LOW);  
659  
660     // Turn off the WiFi module again  
661     WiFi.modulePowerOff();  
662  
663     // Turn off the entire Arduino board  
664     powerManagement.powerDown();  
665  
666     // Turn on the WiFi module again  
667     WiFi.modulePowerOn();  
668  
669     // Turn on the LED  
670     digitalWrite(LED_BUILTIN, HIGH);  
671  
672     // Turn on the WiFi module again  
673     WiFi.modulePowerOn();  
674  
675     // Turn on the entire Arduino board  
676     powerManagement.powerUp();  
677  
678     // Turn off the WiFi module again  
679     WiFi.modulePowerOff();  
680  
681     // Turn off the LED  
682     digitalWrite(LED_BUILTIN, LOW);  
683  
684     // Turn off the WiFi module again  
685     WiFi.modulePowerOff();  
686  
687     // Turn off the entire Arduino board  
688     powerManagement.powerDown();  
689  
690     // Turn on the WiFi module again  
691     WiFi.modulePowerOn();  
692  
693     // Turn on the LED  
694     digitalWrite(LED_BUILTIN, HIGH);  
695  
696     // Turn on the WiFi module again  
697     WiFi.modulePowerOn();  
698  
699     // Turn on the entire Arduino board  
700     powerManagement.powerUp();  
701  
702     // Turn off the WiFi module again  
703     WiFi.modulePowerOff();  
704  
705     // Turn off the LED  
706     digitalWrite(LED_BUILTIN, LOW);  
707  
708     // Turn off the WiFi module again  
709     WiFi.modulePowerOff();  
710  
711     // Turn off the entire Arduino board  
712     powerManagement.powerDown();  
713  
714     // Turn on the WiFi module again  
715     WiFi.modulePowerOn();  
716  
717     // Turn on the LED  
718     digitalWrite(LED_BUILTIN, HIGH);  
719  
720     // Turn on the WiFi module again  
721     WiFi.modulePowerOn();  
722  
723     // Turn on the entire Arduino board  
724     powerManagement.powerUp();  
725  
726     // Turn off the WiFi module again  
727     WiFi.modulePowerOff();  
728  
729     // Turn off the LED  
730     digitalWrite(LED_BUILTIN, LOW);  
731  
732     // Turn off the WiFi module again  
733     WiFi.modulePowerOff();  
734  
735     // Turn off the entire Arduino board  
736     powerManagement.powerDown();  
737  
738     // Turn on the WiFi module again  
739     WiFi.modulePowerOn();  
740  
741     // Turn on the LED  
742     digitalWrite(LED_BUILTIN, HIGH);  
743  
744     // Turn on the WiFi module again  
745     WiFi.modulePowerOn();  
746  
747     // Turn on the entire Arduino board  
748     powerManagement.powerUp();  
749  
750     // Turn off the WiFi module again  
751     WiFi.modulePowerOff();  
752  
753     // Turn off the LED  
754     digitalWrite(LED_BUILTIN, LOW);  
755  
756     // Turn off the WiFi module again  
757     WiFi.modulePowerOff();  
758  
759     // Turn off the entire Arduino board  
760     powerManagement.powerDown();  
761  
762     // Turn on the WiFi module again  
763     WiFi.modulePowerOn();  
764  
765     // Turn on the LED  
766     digitalWrite(LED_BUILTIN, HIGH);  
767  
768     // Turn on the WiFi module again  
769     WiFi.modulePowerOn();  
770  
771     // Turn on the entire Arduino board  
772     powerManagement.powerUp();  
773  
774     // Turn off the WiFi module again  
775     WiFi.modulePowerOff();  
776  
777     // Turn off the LED  
778     digitalWrite(LED_BUILTIN, LOW);  
779  
780     // Turn off the WiFi module again  
781     WiFi.modulePowerOff();  
782  
783     // Turn off the entire Arduino board  
784     powerManagement.powerDown();  
785  
786     // Turn on the WiFi module again  
787     WiFi.modulePowerOn();  
788  
789     // Turn on the LED  
790     digitalWrite(LED_BUILTIN, HIGH);  
791  
792     // Turn on the WiFi module again  
793     WiFi.modulePowerOn();  
794  
795     // Turn on the entire Arduino board  
796     powerManagement.powerUp();  
797  
798     // Turn off the WiFi module again  
799     WiFi.modulePowerOff();  
800  
801     // Turn off the LED  
802     digitalWrite(LED_BUILTIN, LOW);  
803  
804     // Turn off the WiFi module again  
805     WiFi.modulePowerOff();  
806  
807     // Turn off the entire Arduino board  
808     powerManagement.powerDown();  
809  
810     // Turn on the WiFi module again  
811     WiFi.modulePowerOn();  
812  
813     // Turn on the LED  
814     digitalWrite(LED_BUILTIN, HIGH);  
815  
816     // Turn on the WiFi module again  
817     WiFi.modulePowerOn();  
818  
819     // Turn on the entire Arduino board  
820     powerManagement.powerUp();  
821  
822     // Turn off the WiFi module again  
823     WiFi.modulePowerOff();  
824  
825     // Turn off the LED  
826     digitalWrite(LED_BUILTIN, LOW);  
827  
828     // Turn off the WiFi module again  
829     WiFi.modulePowerOff();  
830  
831     // Turn off the entire Arduino board  
832     powerManagement.powerDown();  
833  
834     // Turn on the WiFi module again  
835     WiFi.modulePowerOn();  
836  
837     // Turn on the LED  
838     digitalWrite(LED_BUILTIN, HIGH);  
839  
840     // Turn on the WiFi module again  
841     WiFi.modulePowerOn();  
842  
843     // Turn on the entire Arduino board  
844     powerManagement.powerUp();  
845  
846     // Turn off the WiFi module again  
847     WiFi.modulePowerOff();  
848  
849     // Turn off the LED  
850     digitalWrite(LED_BUILTIN, LOW);  
851  
852     // Turn off the WiFi module again  
853     WiFi.modulePowerOff();  
854  
855     // Turn off the entire Arduino board  
856     powerManagement.powerDown();  
857  
858     // Turn on the WiFi module again  
859     WiFi.modulePowerOn();  
860  
861     // Turn on the LED  
862     digitalWrite(LED_BUILTIN, HIGH);  
863  
864     // Turn on the WiFi module again  
865     WiFi.modulePowerOn();  
866  
867     // Turn on the entire Arduino board  
868     powerManagement.powerUp();  
869  
870     // Turn off the WiFi module again  
871     WiFi.modulePowerOff();  
872  
873     // Turn off the LED  
874     digitalWrite(LED_BUILTIN, LOW);  
875  
876     // Turn off the WiFi module again  
877     WiFi.modulePowerOff();  
878  
879     // Turn off the entire Arduino board  
880     powerManagement.powerDown();  
881  
882     // Turn on the WiFi module again  
883     WiFi.modulePowerOn();  
884  
885     // Turn on the LED  
886     digitalWrite(LED_BUILTIN, HIGH);  
887  
888     // Turn on the WiFi module again  
889     WiFi.modulePowerOn();  
890  
891     // Turn on the entire Arduino board  
892     powerManagement.powerUp();  
893  
894     // Turn off the WiFi module again  
895     WiFi.modulePowerOff();  
896  
897     // Turn off the LED  
898     digitalWrite(LED_BUILTIN, LOW);  
899  
900     // Turn off the WiFi module again  
901     WiFi.modulePowerOff();  
902  
903     // Turn off the entire Arduino board  
904     powerManagement.powerDown();  
905  
906     // Turn on the WiFi module again  
907     WiFi.modulePowerOn();  
908  
909     // Turn on the LED  
910     digitalWrite(LED_BUILTIN, HIGH);  
911  
912     // Turn on the WiFi module again  
913     WiFi.modulePowerOn();  
914  
915     // Turn on the entire Arduino board  
916     powerManagement.powerUp();  
917  
918     // Turn off the WiFi module again  
919     WiFi.modulePowerOff();  
920  
921     // Turn off the LED  
922     digitalWrite(LED_BUILTIN, LOW);  
923  
924     // Turn off the WiFi module again  
925     WiFi.modulePowerOff();  
926  
927     // Turn off the entire Arduino board  
928     powerManagement.powerDown();  
929  
930     // Turn on the WiFi module again  
931     WiFi.modulePowerOn();  
932  
933     // Turn on the LED  
934     digitalWrite(LED_BUILTIN, HIGH);  
935  
936     // Turn on the WiFi module again  
937     WiFi.modulePowerOn();  
938  
939     // Turn on the entire Arduino board  
940     powerManagement.powerUp();  
941  
942     // Turn off the WiFi module again  
943     WiFi.modulePowerOff();  
944  
945     // Turn off the LED  
946     digitalWrite(LED_BUILTIN, LOW);  
947  
948     // Turn off the WiFi module again  
949     WiFi.modulePowerOff();  
950  
951     // Turn off the entire Arduino board  
952     powerManagement.powerDown();  
953  
954     // Turn on the WiFi module again  
955     WiFi.modulePowerOn();  
956  
957     // Turn on the LED  
958     digitalWrite(LED_BUILTIN, HIGH);  
959  
960     // Turn on the WiFi module again  
961     WiFi.modulePowerOn();  
962  
963     // Turn on the entire Arduino board  
964     powerManagement.powerUp();  
965  
966     // Turn off the WiFi module again  
967     WiFi.modulePowerOff();  
968  
969     // Turn off the LED  
970     digitalWrite(LED_BUILTIN, LOW);  
971  
972     // Turn off the WiFi module again  
973     WiFi.modulePowerOff();  
974  
975     // Turn off the entire Arduino board  
976     powerManagement.powerDown();  
977  
978     // Turn on the WiFi module again  
979     WiFi.modulePowerOn();  
980  
981     // Turn on the LED  
982     digitalWrite(LED_BUILTIN, HIGH);  
983  
984     // Turn on the WiFi module again  
985     WiFi.modulePowerOn();  
986  
987     // Turn on the entire Arduino board  
988     powerManagement.powerUp();  
989  
990     // Turn off the WiFi module again  
991     WiFi.modulePowerOff();  
992  
993     // Turn off the LED  
994     digitalWrite(LED_BUILTIN, LOW);  
995  
996     // Turn off the WiFi module again  
997     WiFi.modulePowerOff();  
998  
999     // Turn off the entire Arduino board  
1000    powerManagement.powerDown();  
1001  
1002    // Turn on the WiFi module again  
1003    WiFi.modulePowerOn();  
1004  
1005    // Turn on the LED  
1006    digitalWrite(LED_BUILTIN, HIGH);  
1007  
1008    // Turn on the WiFi module again  
1009    WiFi.modulePowerOn();  
1010  
1011    // Turn on the entire Arduino board  
1012    powerManagement.powerUp();  
1013  
1014    // Turn off the WiFi module again  
1015    WiFi.modulePowerOff();  
1016  
1017    // Turn off the LED  
1018    digitalWrite(LED_BUILTIN, LOW);  
1019  
1020    // Turn off the WiFi module again  
1021    WiFi.modulePowerOff();  
1022  
1023    // Turn off the entire Arduino board  
1024    powerManagement.powerDown();  
1025  
1026    // Turn on the WiFi module again  
1027    WiFi.modulePowerOn();  
1028  
1029    // Turn on the LED  
1030    digitalWrite(LED_BUILTIN, HIGH);  
1031  
1032    // Turn on the WiFi module again  
1033    WiFi.modulePowerOn();  
1034  
1035    // Turn on the entire Arduino board  
1036    powerManagement.powerUp();  
1037  
1038    // Turn off the WiFi module again  
1039    WiFi.modulePowerOff();  
1040  
1041    // Turn off the LED  
1042    digitalWrite(LED_BUILTIN, LOW);  
1043  
1044    // Turn off the WiFi module again  
1045    WiFi.modulePowerOff();  
1046  
1047    // Turn off the entire Arduino board  
1048    powerManagement.powerDown();  
1049  
1050    // Turn on the WiFi module again  
1051    WiFi.modulePowerOn();  
1052  
1053    // Turn on the LED  
1054    digitalWrite(LED_BUILTIN, HIGH);  
1055  
1056    // Turn on the WiFi module again  
1057    WiFi.modulePowerOn();  
1058  
1059    // Turn on the entire Arduino board  
1060    powerManagement.powerUp();  
1061  
1062    // Turn off the WiFi module again  
1063    WiFi.modulePowerOff();  
1064  
1065    // Turn off the LED  
1066    digitalWrite(LED_BUILTIN, LOW);  
1067  
1068    // Turn off the WiFi module again  
1069    WiFi.modulePowerOff();  
1070  
1071    // Turn off the entire Arduino board  
1072    powerManagement.powerDown();  
1073  
1074    // Turn on the WiFi module again  
1075    WiFi.modulePowerOn();  
1076  
1077    // Turn on the LED  
1078    digitalWrite(LED_BUILTIN, HIGH);  
1079  
1080    // Turn on the WiFi module again  
1081    WiFi.modulePowerOn();  
1082  
1083    // Turn on the entire Arduino board  
1084    powerManagement.powerUp();  
1085  
1086    // Turn off the WiFi module again  
1087    WiFi.modulePowerOff();  
1088  
1089    // Turn off the LED  
1090    digitalWrite(LED_BUILTIN, LOW);  
1091  
1092    // Turn off the WiFi module again  
1093    WiFi.modulePowerOff();  
1094  
1095    // Turn off the entire Arduino board  
1096    powerManagement.powerDown();  
1097  
1098    // Turn on the WiFi module again  
1099    WiFi.modulePowerOn();  
1100  
1101    // Turn on the LED  
1102    digitalWrite(LED_BUILTIN, HIGH);  
1103  
1104    // Turn on the WiFi module again  
1105    WiFi.modulePowerOn();  
1106  
1107    // Turn on the entire Arduino board  
1108    powerManagement.powerUp();  
1109  
1110    // Turn off the WiFi module again  
1111    WiFi.modulePowerOff();  
1112  
1113    // Turn off the LED  
1114    digitalWrite(LED_BUILTIN, LOW);  
1115  
1116    // Turn off the WiFi module again  
1117    WiFi.modulePowerOff();  
1118  
1119    // Turn off the entire Arduino board  
1120    powerManagement.powerDown();  
1121  
1122    // Turn on the WiFi module again  
1123    WiFi.modulePowerOn();  
1124  
1125    // Turn on the LED  
1126    digitalWrite(LED_BUILTIN, HIGH);  
1127  
1128    // Turn on the WiFi module again  
1129    WiFi.modulePowerOn();  
1130  
1131    // Turn on the entire Arduino board  
1132    powerManagement.powerUp();  
1133  
1134    // Turn off the WiFi module again  
1135    WiFi.modulePowerOff();  
1136  
1137    // Turn off the LED  
1138    digitalWrite(LED_BUILTIN, LOW);  
1139  
1140    // Turn off the WiFi module again  
1141    WiFi.modulePowerOff();  
1142  
1143    // Turn off the entire Arduino board  
1144    powerManagement.powerDown();  
1145  
1146    // Turn on the WiFi module again  
1147    WiFi.modulePowerOn();  
1148  
1149    // Turn on the LED  
1150    digitalWrite(LED_BUILTIN, HIGH);  
1151  
1152    // Turn on the WiFi module again  
1153    WiFi.modulePowerOn();  
1154  
1155    // Turn on the entire Arduino board  
1156    powerManagement.powerUp();  
1157  
1158    // Turn off the WiFi module again  
1159    WiFi.modulePowerOff();  
1160  
1161    // Turn off the LED  
1162    digitalWrite(LED_BUILTIN, LOW);  
1163  
1164    // Turn off the WiFi module again  
1165    WiFi.modulePowerOff();  
1166  
1167    // Turn off the entire Arduino board  
1168    powerManagement.powerDown();  
1169  
1170    // Turn on the WiFi module again  
1171    WiFi.modulePowerOn();  
1172  
1173    // Turn on the LED  
1174    digitalWrite(LED_BUILTIN, HIGH);  
1175  
1176    // Turn on the WiFi module again  
1177    WiFi.modulePowerOn();  
1178  
1179    // Turn on the entire Arduino board  
1180    powerManagement.powerUp();  
1181  
1182    // Turn off the WiFi module again  
1183    WiFi.modulePowerOff();  
1184  
1185    // Turn off the LED  
1186    digitalWrite(LED_BUILTIN, LOW);  
1187  
1188    // Turn off the WiFi module again  
1189    WiFi.modulePowerOff();  
1190  
1191    // Turn off the entire Arduino board  
1192    powerManagement.powerDown();  
1193  
1194    // Turn on the WiFi module again  
1195    WiFi.modulePowerOn();  
1196  
1197    // Turn on the LED  
1198    digitalWrite(LED_BUILTIN, HIGH);  
1199  
1200    // Turn on the WiFi module again  
1201    WiFi.modulePowerOn();  
1202  
1203    // Turn on the entire Arduino board  
1204    powerManagement.powerUp();  
1205  
1206    // Turn off the WiFi module again  
1207    WiFi.modulePowerOff();  
1208  
1209    // Turn off the LED  
1210    digitalWrite(LED_BUILTIN, LOW);  
1211  
1212    // Turn off the WiFi module again  
1213    WiFi.modulePowerOff();  
1214  
1215    // Turn off the entire Arduino board  
1
```

Azure services



Create a
resource



Power BI
Embedded



All resources

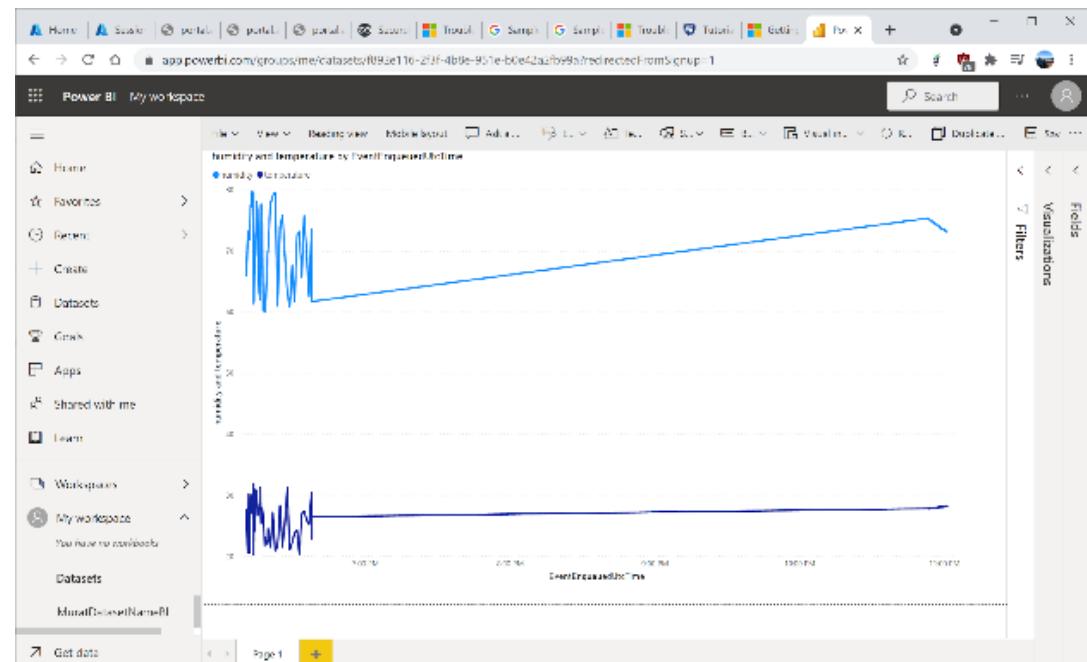
Recent resources

Name

MuratloTHub

MuratStreamAnalysJob

MuratResourceGroup



Azure Settings

The screenshot shows the Microsoft Azure Stream Analytics job configuration interface. The left sidebar lists navigation options: Overview, Activity Log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Properties, and Logs. The main area displays the 'MuratStreamAnalyticsJob | Query' configuration. It includes a search bar, a 'Query language editor' tab, and a 'Save query' button. The 'Inputs' section shows 'MuratInputAlias' with a warning icon. The 'Outputs' section shows 'MuratPowerBIOutput'. A preview pane displays the following T-SQL query:

```
1 SELECT *
2 INTO [MuratPowerBIOutput]
3 FROM [MuratInputAlias]
```

The screenshot shows the Microsoft Azure IoT Hub device configuration interface for 'MyMKRWiFi1010'. The top navigation bar includes links for Home, Search resources, services, and does (G /), and a user profile. The main area shows the device details: Device ID (MyMKRWiFi1010), Primary Thumbprint, Secondary Thumbprint, and a toggle switch for 'Enable connection to IoT Hub' which is set to 'Enable'. Below these are sections for 'Parent device' (No parent device) and 'Module identities' (Configurations). At the bottom, there are tabs for 'Module identities' and 'Configurations'.