# How to integrate RunMyProcess to Azure IoT Hub:

# **Calling Direct Methods in a simulated device with REST**

#### Rodrigo Riveros – Senior Consultant – Fujitsu Canada

Table of Contents

[How to integrate RunMyProcess to Azure IoT Hub: 1](#_Toc474152031)

[**Calling Direct Methods in a simulated device with REST** 1](#_Toc474152032)

[Pre-requisites: 2](#_Toc474152033)

[Configure your Azure IoT hub 3](#_Toc474152034)

[Create a device identity in the IoT hub 3](#_Toc474152035)

[Create a Simulated device with a method to be called. 7](#_Toc474152036)

[Test the method from Postman (Using POST method of the Rest API) 8](#_Toc474152037)

[RMP Connector Configuration 10](#_Toc474152038)

[- Create the provider 10](#_Toc474152039)

[- Create the connector 11](#_Toc474152040)

[General References: 12](#_Toc474152041)

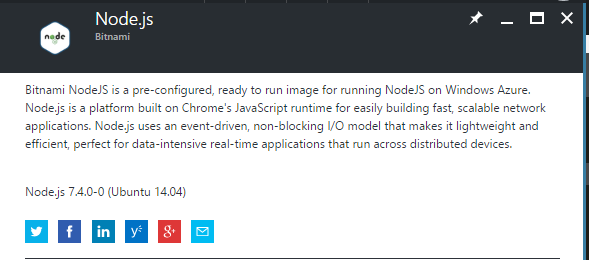
### Pre-requisites:

* An Azure Account

[https://portal.azure.com]

* A NodeJS server (If you’re going to simulate the IoT Device)

My suggestion is to use an Azure NodeJS Virtual Machine from bitnami. It came pre-configured to run NodeJS:



* Access to RMP IDE

[https://portal.runmyprocess.com/ide/]

* Postman (To test the message to send before implementing it in RMP) [https://www.getpostman.com/]
* Device Explorer Twin

[https://github.com/Azure/azure-iot-sdks/releases]

It’s a Windows tool written by the IoT hub dev team to help you manage your IoT hub connected Devices.

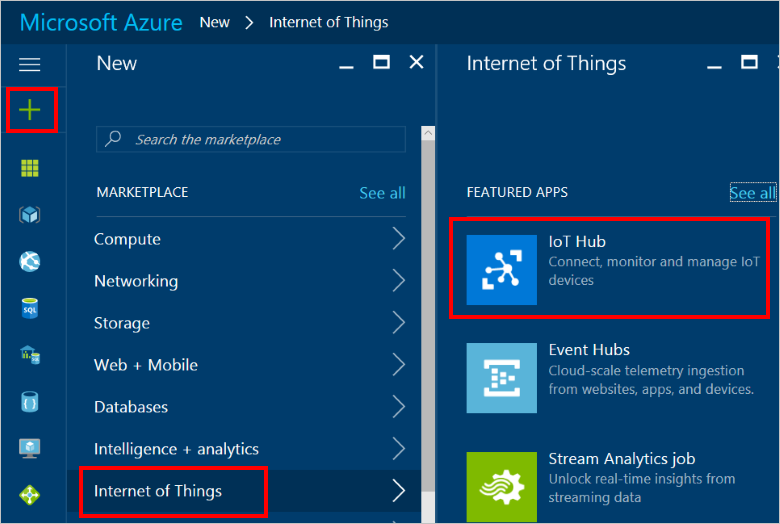
Downloadable from here: [https://github.com/Azure/azure-iot-sdks/releases/download/2016-11-17/SetupDeviceExplorer.msi]

## Configure your Azure IoT hub

Reference: <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-node-node-direct-methods#create-an-iot-hub>

To configure your Azure IoT Hub you need a valid azure subscription.

Then in the azure portal add an IoT Hub from the marketplace/Internet of things



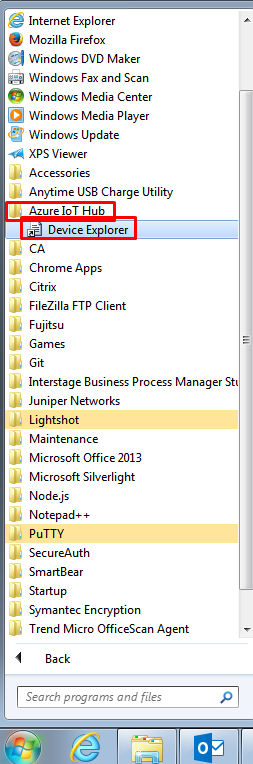
Fill the requested information and wait for the deployment to complete.

## Create a device identity in the IoT hub

Reference: <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-node-node-direct-methods#create-a-device-identity>

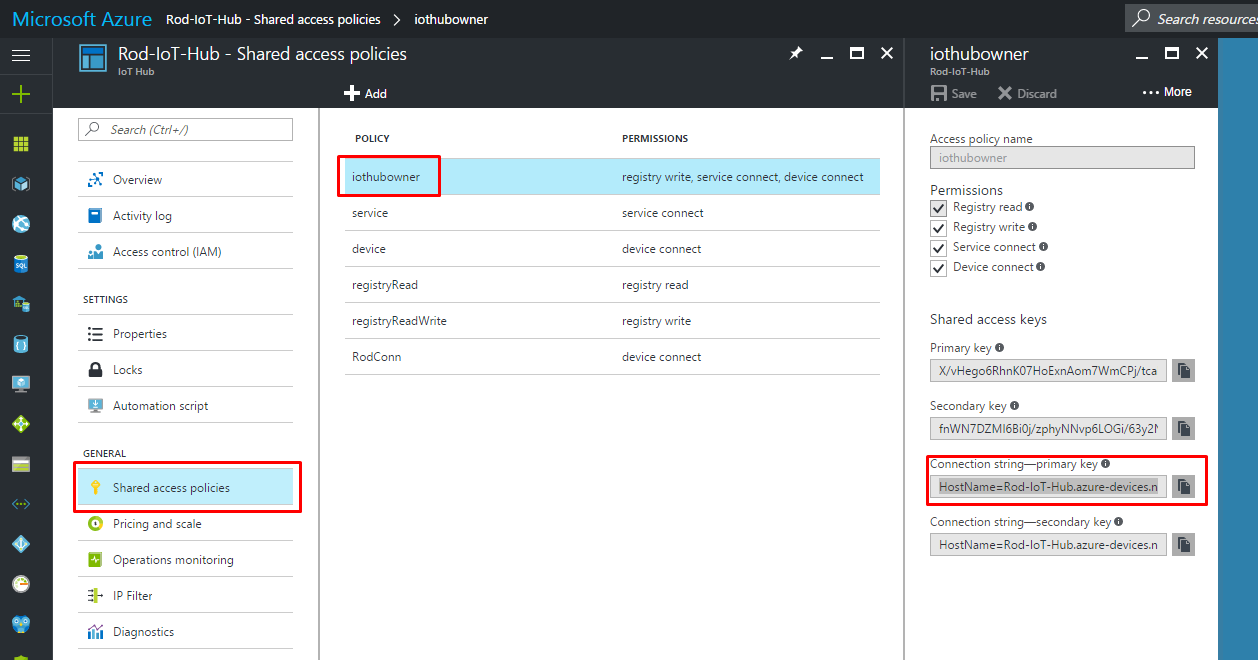
There is several ways to do this using the IoT hub SDK, I found that the simpler way for the purpose of this example, is doing it from the device explorer tool:

* Open the Device Explorer:

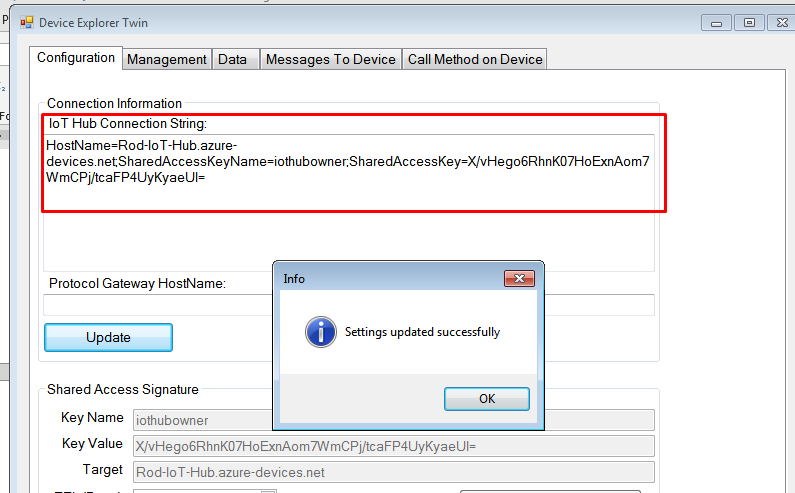


* Setup your IoT hub connection string:

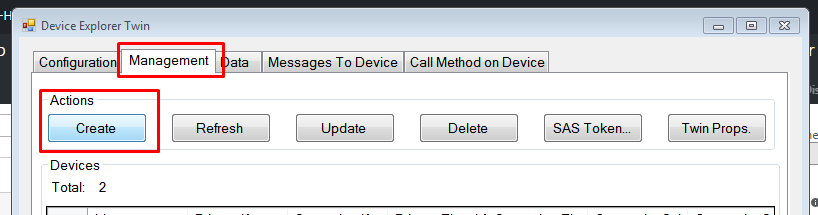
You can find the IoT hub connection string in the azure portal, then go to your hub, click on the Shared Access policies, click on the iothubowner policy, the connection string will be in the right blade:

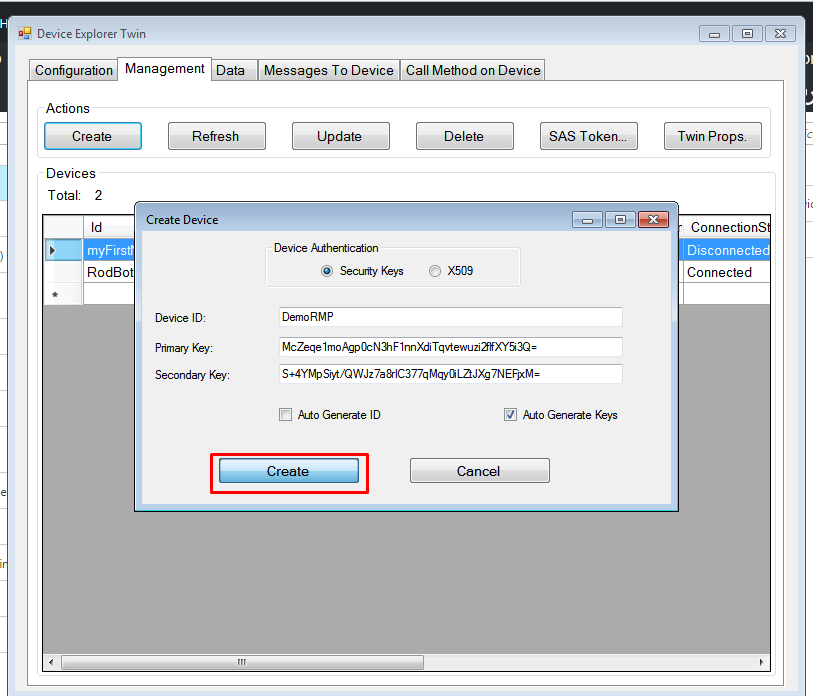


Paste the connection string in the device explorer and click on update:



* Click in the Management Tab & Click on the create Button



* Fill the Device Name and click Create:
* 

## Create a Simulated device with a method to be called.

Reference: <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-node-node-direct-methods>

* Open a SSH session to your NodeJS Server.
* Create a folder simulateddevice

Run the following commands:

|  |
| --- |
| npm init  npm install azure-iot-device azure-iot-device-mqtt –save |

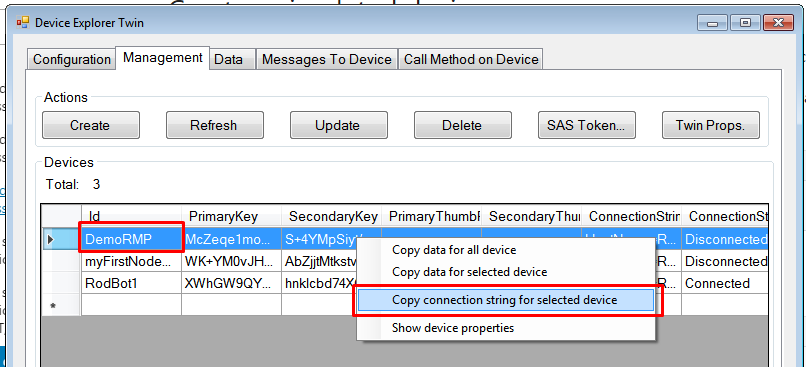
* Create a new file SimulatedDevice.js and Copy the following code:

|  |
| --- |
| 'use strict';  var Mqtt = require('azure-iot-device-mqtt').Mqtt;  var DeviceClient = require('azure-iot-device').Client;  var connectionString = '{DeviceConnectionString}';  var client = DeviceClient.fromConnectionString(connectionString, Mqtt);  function onWriteLine(request, response) {  //To print request  console.log(request.payload);  response.send(200, 'Input was written to log.', function(err) {  if (err) {  console.error('An error ocurred when sending a method response:\n' + err.toString());  } else {  console.log('Response to method \'' + request.methodName + '\' sent successfully.');  }  });  }  client.open(function(err) {  if (err) {  console.error('could not open IotHub client');  } else {  console.log('client opened');  client.onDeviceMethod('writeLine', onWriteLine);  }  }); |

* Replace the values in red with the Device Connection String

##### Where is the device Connection String?

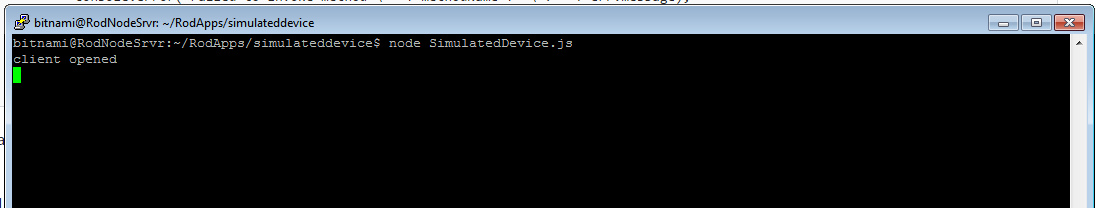
Go back to the device explorer: Right click on the device you just created, Copy connection string for selected device.



* Save the .js file and run the following command:

|  |
| --- |
| node SimulatedDevice.js |

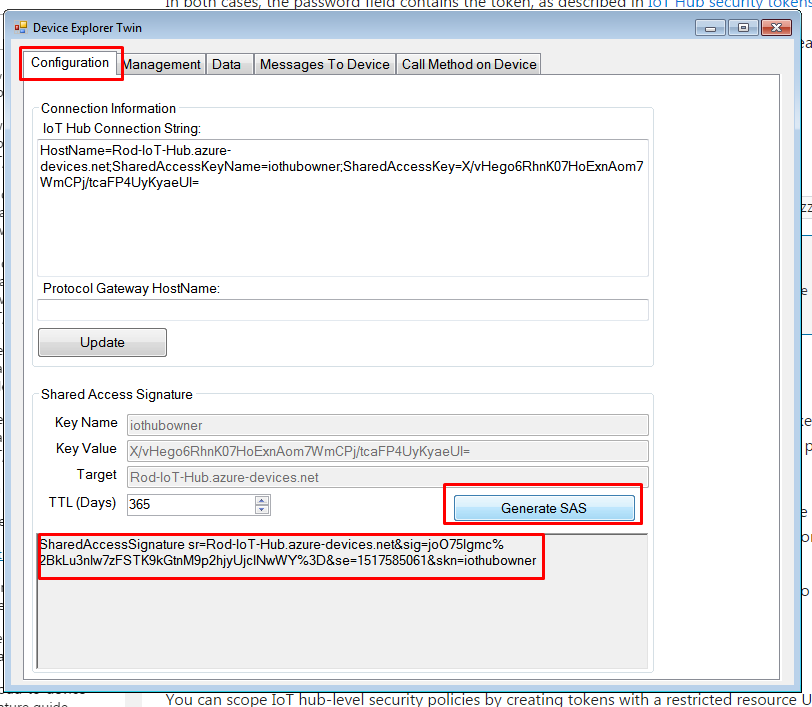
* Then you will have a client opened message and a simulated device connected to the IoT hub waiting for a message!



## Test the method from Postman (Using POST method of the Rest API)

Reference: <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-direct-methods>

Go back to the device explorer, In the Configuration tab create the SAS key for the IoT hub (Click on generate SAS)



In Postman these are the parameters to configure:

Method: POST

Request URL: https://{YourIoTHubName}.azure-devices.net/twins/{YourDeviceName}/methods?api-version=2016-11-14

Authorization: No authorization

Headers:

|  |  |
| --- | --- |
| Authorization | {YourSASValue} |
| Content-Type | application/json |

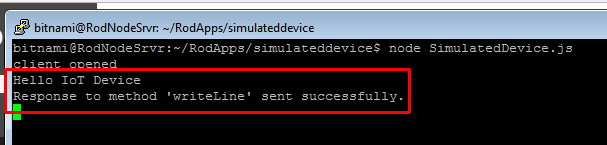
Body:

|  |
| --- |
| {  "methodName": "writeLine",  "responseTimeoutInSeconds": 120,  "payload": "Hello IoT Device"  } |

The request must return:

|  |
| --- |
| {  "status": 200,  "payload": "Input was written to log."  } |

Go to the SSH session, you must see the message you just send to the device!

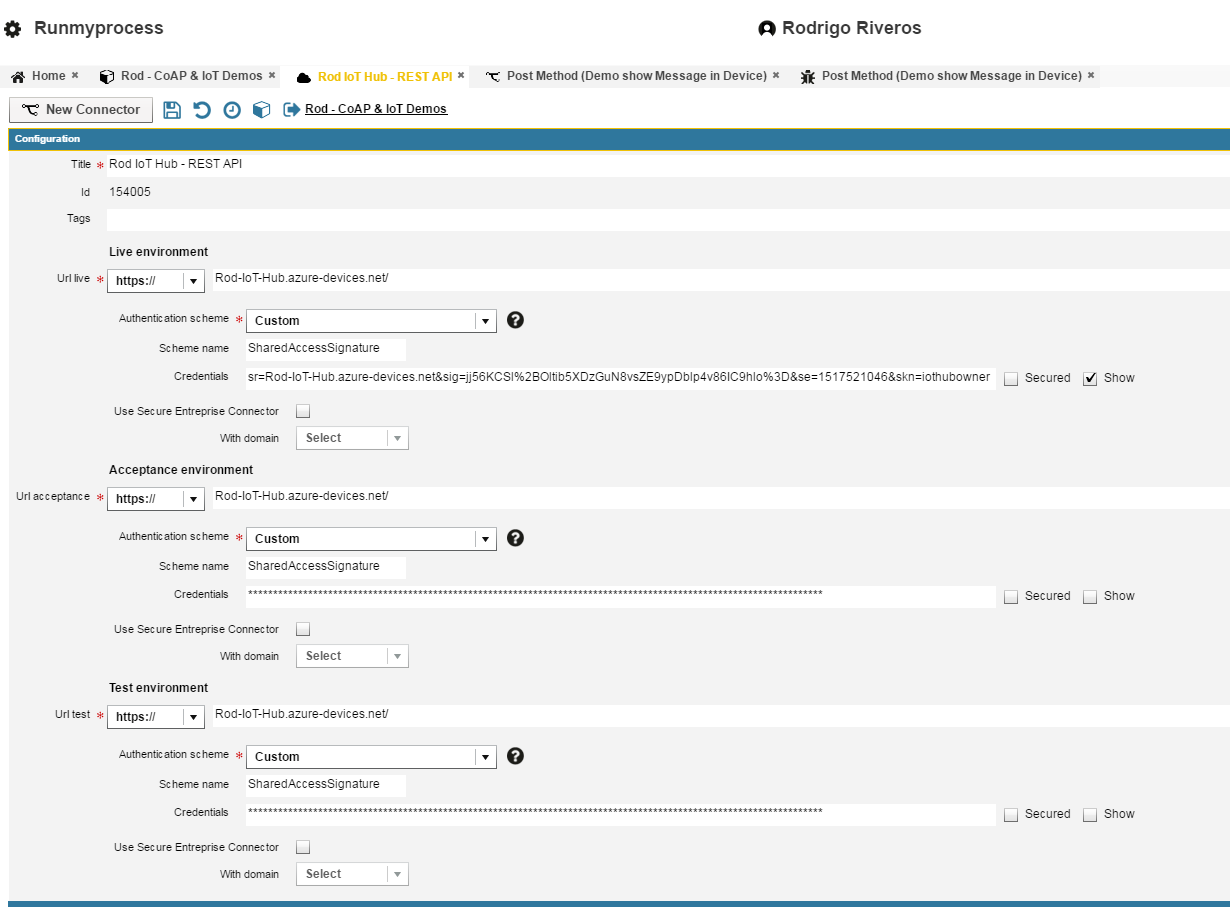


## RMP Connector Configuration

### - Create the provider

In your project create a new provider.

|  |  |
| --- | --- |
| Url | https://{YourIoTHubName}.azure-devices.net/ |
| Authentication scheme | Custom |
| Scheme name | SharedAccessSignature |
| Credentials | {YourSASValue}  *Note: Delete the SharedAccessSignature at the start, you’ve already send it as Scheme name!!* |

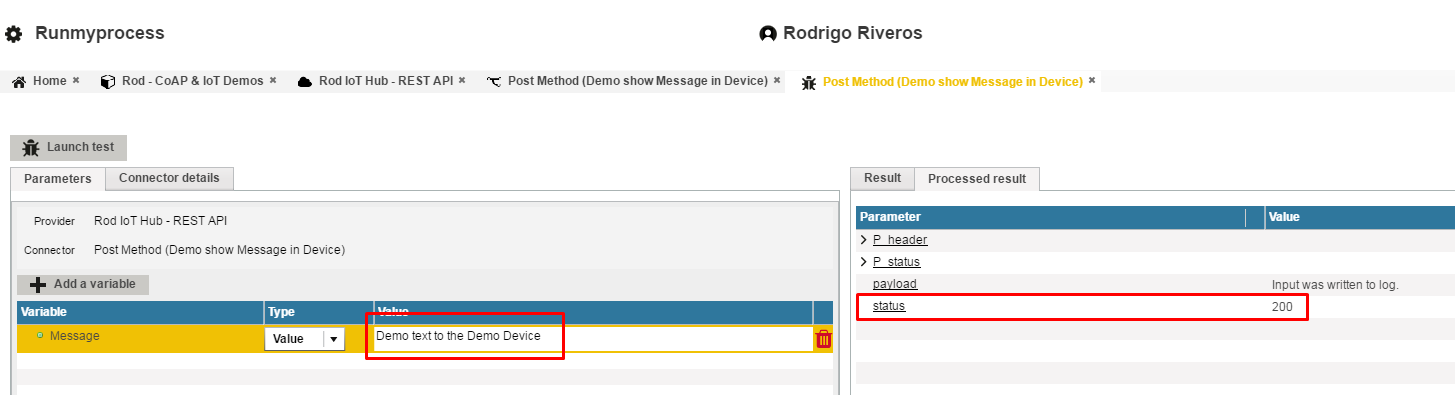


### - Create the connector

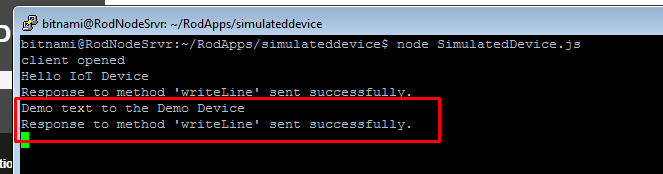
Create a new connector from the provider you just created.

|  |  |
| --- | --- |
| Connector URL | twins/{YourDeviceName}/methods?api-version=2016-11-14 |
| Architecture | REST/XML-RPC |
| Method | POST |
| Result Format | JSON |
| Accept Media Type | Application/json |
| Content | {  "methodName": "writeLine",  "responseTimeoutInSeconds": 120,  "payload": "${Message}"  } |
| Content Type | application/json |
| Result transformation |  |

Test:



Then when the test succeed, you can go back to the SSH session, you will see the message you just sent to the device.



Imagine the device is a traffic panel in a highway, or the event is to execute an action device remotely (Reboot? Standby?).

## General References:

<https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-what-is-azure-iot>

<https://www.youtube.com/watch?v=5ES-1g_mGxY>

<https://azure.microsoft.com/en-gb/services/iot-hub/>