

Internet of Things Workshop

Lab 2

Connect Device to IoTHub

Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Change Reference |
| 11/7/2016 | Steve Busby | 1.0 | Initial draft |

Table of Contents

[Change Record 2](#_Toc466284196)

[Introduction 4](#_Toc466284197)

[1. Configuring the IOT Hub 5](#_Toc466284198)

[2.) Posting Telemetry data to Azure 9](#_Toc466284199)

[3) making sure data and commands are flowing 12](#_Toc466284200)

[Appendix A – Configuring your Raspberry PI 13](#_Toc466284201)

[Appendix B – hooking your device up to the remote monitoring solution 14](#_Toc466284202)

Introduction

The purpose of this lab is to get an IoT device connected to and talking to Azure IoTHub and the RM-PCS. While this is demo device in a demo situation, it is very representative of the process of connecting any device to Azure. We will leverage the Raspberry PI 3 and a DHT22 temperature and humidity sensor to represent our device. The connection to Azure will leverage a Python script running on Linux, which is a very common IoT development environment. However, it is certainly not the only option. The Azure IoT Device SDKs leveraged for this lab are also available in the following languages:

* C
* Java
* C#
* Python
* Node.js

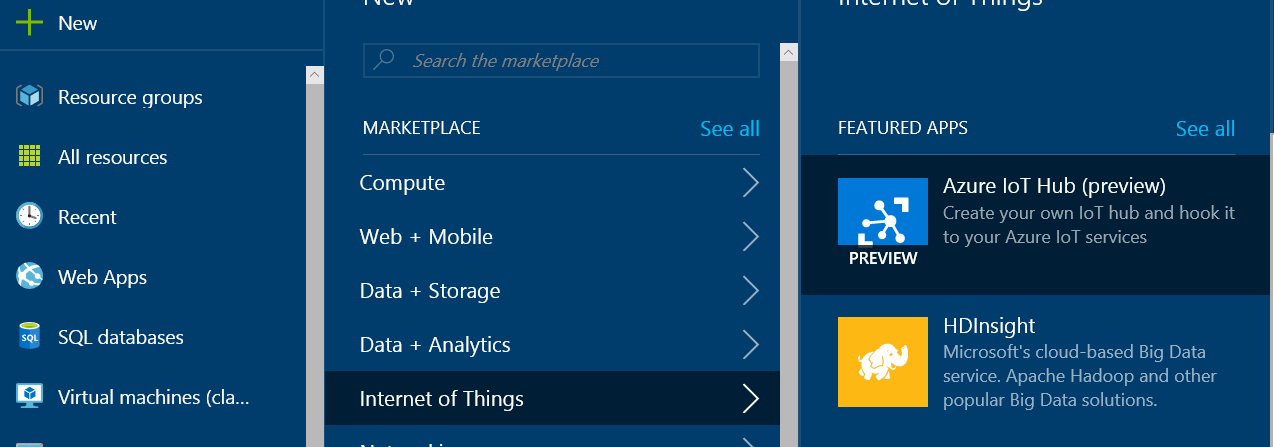
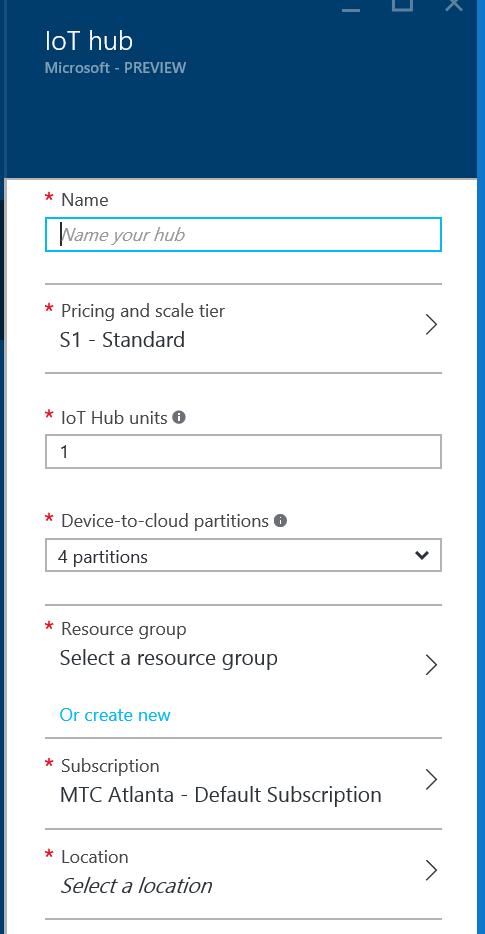
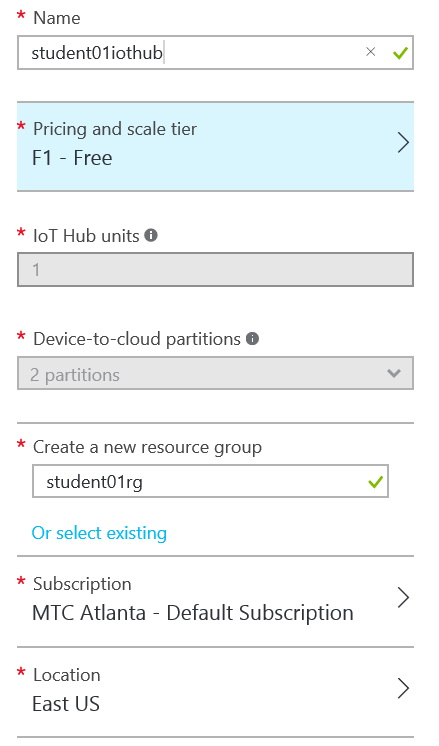
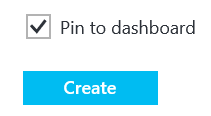
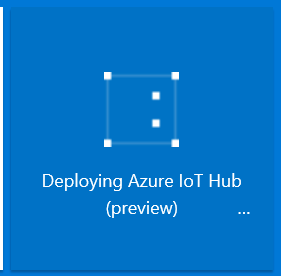
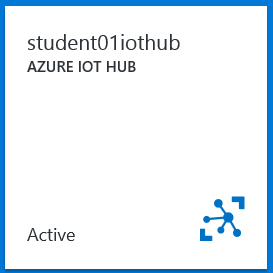
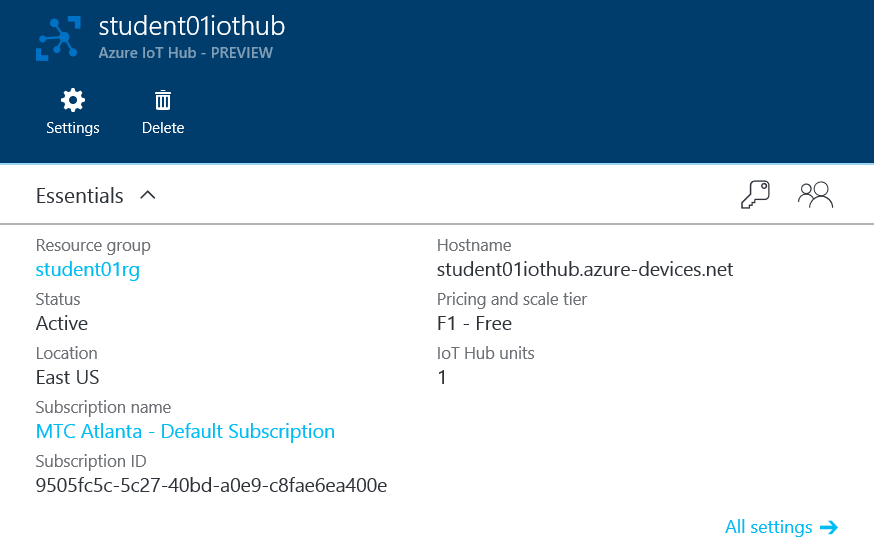
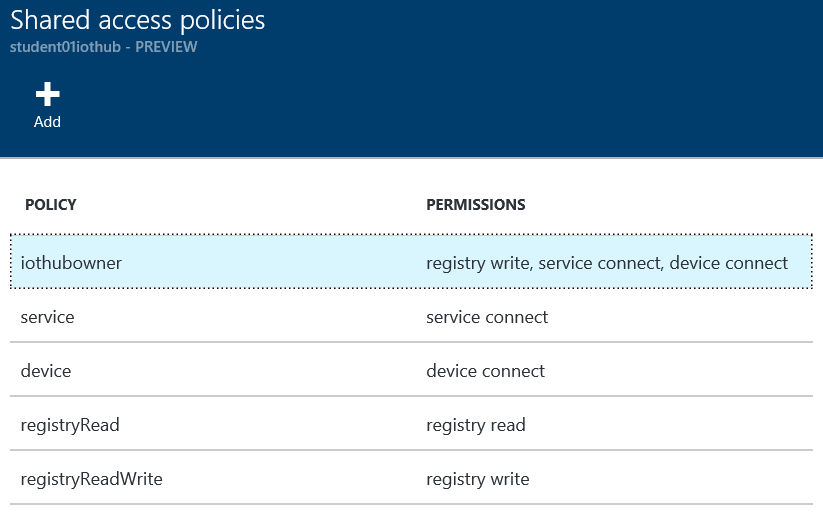
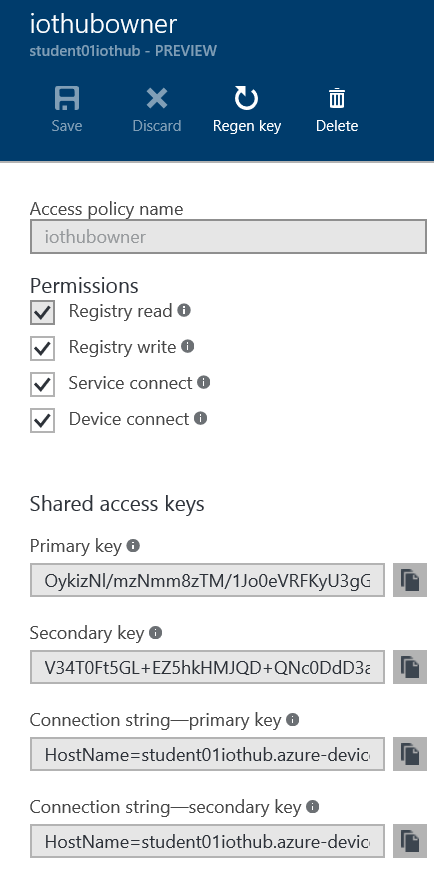
It has also been tested and used on these platforms

* Linux
* Windows (universal – desktop and mobile)
* Various Real-Time OS’s (RTOS), including FreeRTOS, Arduino, and many others
* IoS
* Android.

At the end of this lab you will have a physical IoT device connected to Wifi, sending telemetry data to Azure IoT

1. Configuring the IOT Hub

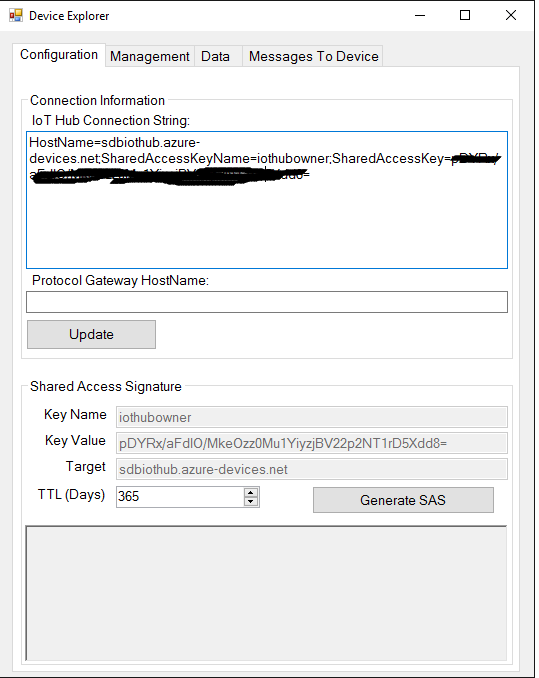
In this section you will provision an IOT hub for capturing sensor events, and setup its shared access signature for securing the hub:

1. Using a web browser navigate to portal.azure.com. Authenticate with the studentXX account given to you.
2. Create an Azure IoT Hub from the portal navigation using + New -> Internet of Things -> Azure IoT Hub.  
   
3. You’ll be presented with the create screen for IoT hub creation. On this screen enter a unique name for your IoT hub, its pricing tier, and location.   
   
4. For the name enter student<xx>iothub, choose the “Free” pricing tier, create a new resource group called student<xx>rg if one does not already exist, and pick a region (ie. East US).  
   
5. Click the “Create” button to provision the IoT Hub.  
   
6. You’ll be taken back to the azure dashboard page where you’ll see an indicator that your IoT Hub is being deployed.  
   
7. Once the hub creation is complete you’ll see the following:  
   
8. You’ll be taken to the IoT Hub home page.   
   
9. Click on the key to acces the security policies   
   
10. You’ll be presented a listing of different shared access security policies.  
    
11. Click on “iothubowner” to access the security keys for this access signature.  
    
12. Copy the primary key, and the primary connection string with the key by clicking the  icon next to the primary key, and save the copied string (to notepad or some scratch space). You’ll need this connection string later.

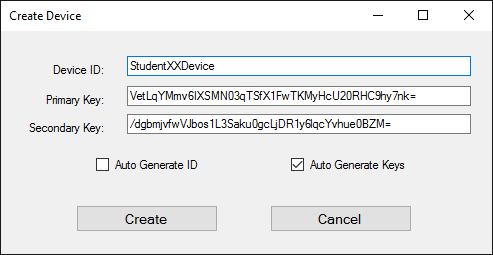
2.) Posting Telemetry data to Azure

Now that we can read the DHT sensor, and have an IoTHub to post to, we need to create the ‘device’ in the IotHub and modify the code to point to our specific IoTHub and device record

1. Now we need to create a device in IoTHub and get its key. This can be done programmatically via the RegistryManager class, but for this POC, we will do it manually through the Device Explorer application (to introduce you to that tool).
   1. On the Cortana search bar, type “Device Explorer” and open the app.
   2. If Device Explorer is not on your dev machine, you can download and install it from <https://github.com/Azure/azure-iot-sdks/releases/download/2016-06-03/SetupDeviceExplorer.msi>



* 1. In the Connection Information box, enter your connection string from your IOTHub and click the “Update” button.
  2. Switch to the “management” tab and click “Create” to create a new device
  3. Enter “StudentXXDevice” (replacing XX with your student ID) into the DeviceID field and click Create



* 1. In the “Device Created” confirmation dialog, copy the DeviceID and Keys. You’ll need this in the next step. Click Done.

1. On the RPI, we need to download the code. We’ve provided the python script, and also pre-compiled the python SDK for your use. If you are interested in the steps to compile the python module yourself if you need to start from scratch, see the details in Appendix A
2. To download the code, enter these commands on the RPI (you may have already done this in Lab 1 – if so, skip this step)
   1. cd ~
   2. git clone –recursive <https://github.com/stevebus/UTCWorkshop>
   3. cd IoTWorkshop/Lab2
3. the code is downloaded, now we need to put the details for your specific device, IoTHub, and (just for fun), location
4. edit the lab2.py script with your favorite linux editor. If you don’t have one, use nano
   1. enter “nano lab2.py”
5. Read through the code, which is well commented, and make sure you understand what it is doing.
6. scroll down to line 18 (you can see what line you are own by typing “CTRL-C”). Line 17 is where you enter the connection details you got from the “add device” process in the previous step. Copy/paste your host name, device id, and device key you copied into the placeholders (make sure to delete the < and > signs). Full screening ‘putty’ allows you to see more of the line at once.

|  |
| --- |
| ############## values to change ############## |
| deviceID = "<yourdeviceID>" |
| deviceKey = "<yourdevickey>" |
| iotHubHostName = "<youriothubname>.azure-devices.net" #ex: myiothub.azure-devices.net |

1. go to <http://www.bing.com/maps> and search for your home, your home town, or some other location that interests you. On the map you can see the latitude and longitude. Substitute those values into the lines below.

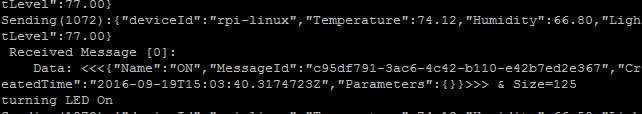
|  |
| --- |
| ######## put the lat and long of your fav place here ########## |
| ######## otherwise we default to the center of the football universe ###### |
| latitude = 33.208350 |
| longitude = -87.550320 |

1. Save the file. If you are using nano as the editor, hit CTRL-O, <enter>
2. Close the file (CTRL-X in nano)
3. Execute the script (‘python lab2.py’ from the command prompt)
4. You should see the DeviceInfo string echo’ed to the screen and sent to the IoTHub, and then every 3 seconds, you should see the temperature and humidity (in a JSON string) sent to the IoTHub. The LED should also briefly flash to indicate we are sending data
5. Congratulations, you’ve connected a physical IoT device to the Azure IoTHub. Next we can look at the telemetry, as well as test manually sending a command to the device from the portal, which we will do in the next step

3) making sure data and commands are flowing

Now we can look at the data hitting IoTHub and test sending a command to the device

1. In the Azure Portal, navigate back to your IoTHub, click on “Shared Access Policies” then “iothubowner”. Copy the “connection string – primary key” string and save for the next step
2. Launch Device Explorer and paste the connection string copied above into the “IoT Hub Connection String” box and click “update”
3. On the “Data” tab, pick your DeviceID from the drop down box and hit “monitor’. You should see messages ‘flowing’ through the hub, now we can test sending ‘commands’ down to the device
4. On the “Messages to device” tab, select your DeviceID and type the word “ON” into the message box and hit Send. You should see text similar to the following on the raspberry PI console and the LED should light



1. Now type “OFF” into the message box and send the command. Observe the LED turn back off and the corresponding message in the RPI console

Congratulations! You now have a physical IoT device talking to the RM-PCS. In the next couple of labs, we’ll do some processing of the telemetry data looking for high temperature “alarms” and responding to them.

Appendix A – Configuring your Raspberry PI

1. Setup Wireless (don’t need to do this if your PI is already connected to the internet)

* Open the wpa-supplicant configuration file in nano:
  + - sudo nano /etc/wpa\_supplicant/wpa\_supplicant.conf
* Go to the bottom of the file and add the following:
  + - network={
      * ssid="The\_ESSID\_from\_earlier"
      * psk="Your\_wifi\_password"
    - }
* You might need to reboot to have the changes take effect. To reboot:

sudo reboot

1. Install Python

sudo apt-get update

sudo apt-get install build-essential python-dev

1. Install drivers for the temperature sensor (DHT)

cd ~

git clone –recursive <https://github.com/adafruit/Adafruit_Python_DHT.git>

cd Adafruit\_Python\_DHT

sudo python setup.py install

1. Install Azure IoT Hub device SDK for Python

cd ~

git clone –recursive <https://github.com/Azure/azure-iot-sdks.git>

cd azure-iot-sdks/python/build\_all/linux

./setup.sh

./build.sh

* To build with python 3.4 or 3.5, run ./build.sh --build-python 3.4 or ./build.sh --build-python 3.5 respectively

Appendix B – hooking your device up to the remote monitoring solution

As an optional final part of the lab, you can hook your device up to the Remote Monitoring solution as generated by your lab proctors. To do so,, follow these steps

1. Edit the lab2.py script again
2. Change the deviceID, IoTHub hostname, and private key to those provided by your proctors
3. Uncomment lines 179 and 180 of your script
4. Save the script and execute
5. The lab proctors will have the solution up on the screen for you to see your device show up on the map.