

# Smart Builds, Azure IoT Hub, Raspberry Pi for Mac OS Developers (v.7)

This guide is intended for developers with an Apple Mac running Mac OS and building “Internet of Things” solutions on Azure [IoT Hub](https://azure.microsoft.com/en-us/services/iot-hub/), [Stream Analytics](https://azure.microsoft.com/en-us/services/stream-analytics/), [Power BI](http://www.powerbi.com/), [Machine Learning](https://azure.microsoft.com/en-us/services/machine-learning/), [Functions](https://azure.microsoft.com/en-us/services/functions/), Data services and more.

The JSON data streamed from this Smart Building sample is compatible with, and intended to be used in conjunction with Section 4 “Microsoft Azure Cloud Development” of the [Maker Den User Guide](https://github.com/MakerDen/Maker-Den-Documentation-and-Resources-FezHat/blob/master/IoT%20Den%20for%20Windows%2010%20IoT%20Core%20User%20Guide%20FEZ%20HAT%20-%20Lite.pdf).

This guide assumes you have a working understanding of Mac OS, some development experience and some knowledge of [Python](https://www.python.org/doc/).

The Smart Building Environment sample is written in Python3 and has been tested on Mac OS, Raspberry Pi, Windows and Ubuntu Linux. It should run on any platform supporting Python3 and the required pip3 packages such as a Beagleboard.

When the Smart Building Environment sample is run on your Apple Mac the environmental data is requested from the [free Open Weather Map](http://openweathermap.org/) service.

Running the Smart Building Environment sample on your Apple Mac will enable you to make a fast start streaming sensor data to Azure IoT Hub.

With sensor data in the Azure cloud you can use advanced Azure services such as Stream Analytics, Power BI, Machine Learning, Functions, Data services and more to build amazing solutions.

The goal is to extend the Smart Building sample app and run on the Raspberry Pi using local environmental data from a Raspberry Pi Sense HAT.

The Raspberry Pi Sense HAT has a great selection of sensors, a display and a joystick.

There are temperature, barometric and humidity sensors. You could use the accelerometer to simulate an earthquake, a lift vibrating in need of maintenance.

Joystick presses could be used to simulate the number of people in an area or using a facility such as the bathroom.

The magnetometer (compass) to simulate the sun or wind direction. You could use the LED display to graph data, simulate blinds opening and closing or lights being turned off and on. These are just some ideas, let your imagination run wild.

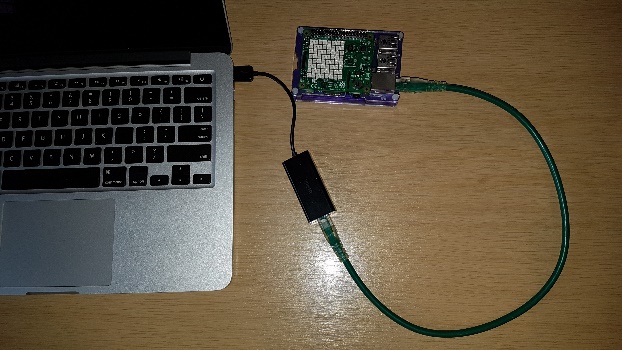
This Guide Covers

1. Setting up Raspbian Linux for the Raspberry Pi from an Apple Mac
2. Setting up Internet Sharing on Mac OS
3. Setting up your Raspberry Pi
4. Installing Recommended Mac OS Packages
5. Provisioning an Azure IoT Hub Device
6. Azure IoT Hub Device Identity Management
7. Smart Building Environment Sample Configuration
8. Developer Workflow

Setting up Raspbian Linux for the Raspberry Pi from an Apple Mac

1. Download from <https://www.raspberrypi.org/downloads/raspbian/>
2. Setting up the Raspberry Pi operating system on an SD Card from Mac OS <https://www.raspberrypi.org/documentation/installation/installing-images/mac.md>
3. Eject the SD card from Mac OS

Powering up your Raspberry Pi

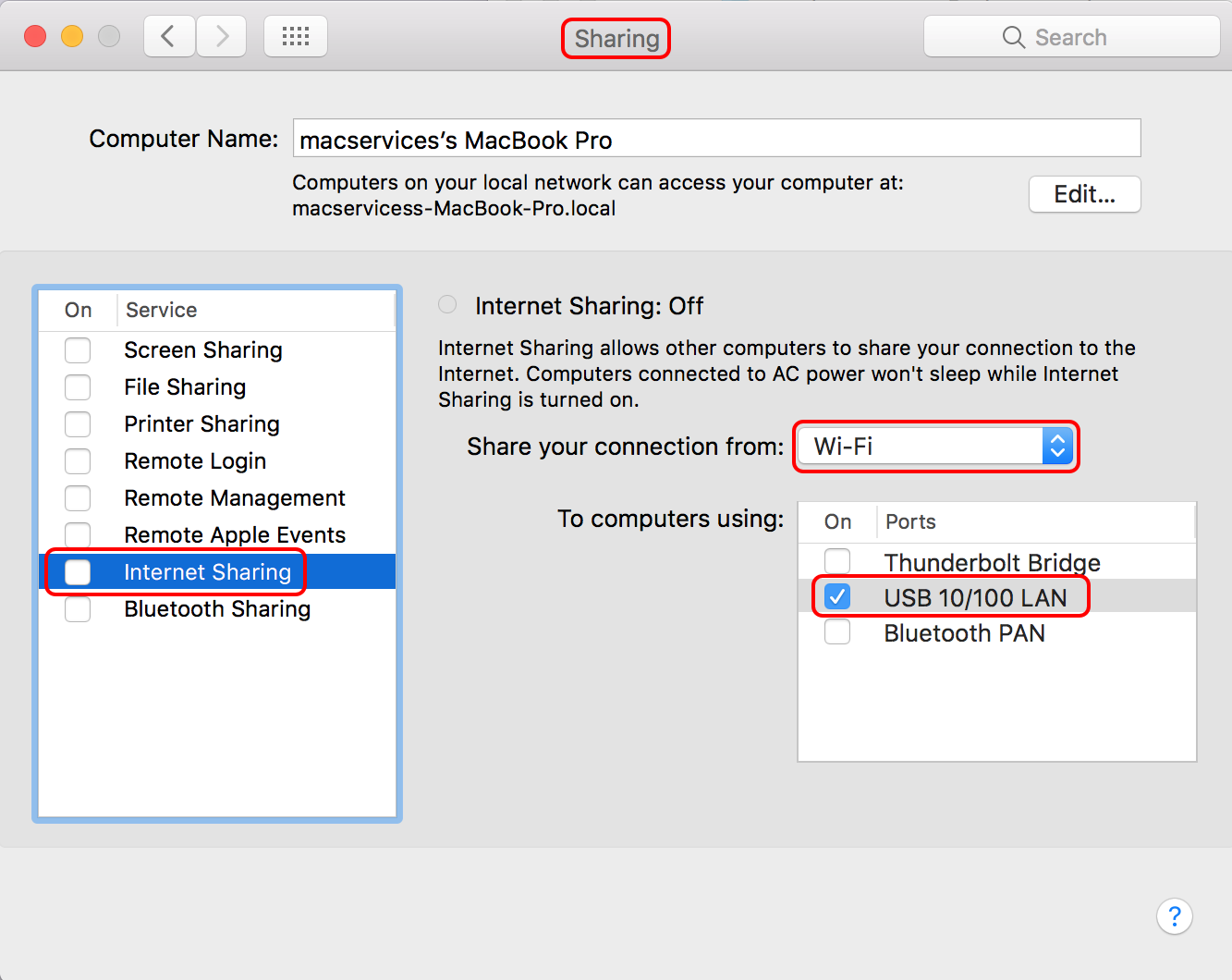
1. Ensure the SD Card is properly seated in the Raspberry Pi
2. Connected the Raspberry Pi via the Ethernet cable to the USB Ethernet Dongle that is plugged into your Apple Mac
3. Power on the Raspberry Pi
4. Wait for a couple of minutes for the Raspberry Pi to perform its initial boot up sequence.
5. The green disk activity will stop flashing when it is ready

Setting up Internet Sharing on Mac OS

Enabling Internet Sharing on your Apple Mac will allow internet requests from the Raspberry Pi to pass through you’re Apple Mac and its Wifi connection to the Internet. This connection is needed to update the Raspberry Pi with the latest patches and allow IoT traffic to flow between the Raspberry Pi and Azure IoT Hub.

See <https://support.apple.com/kb/PH18704?locale=en_AU> for more information.

1. Ensure you have an active Wi-Fi connection on your Apple Mac
2. From the Apple menu > System Preferences, then click Sharing
3. Share your Wi-Fi connection
4. To computers using the USB 10/100 LAN
5. Enable Internet Sharing

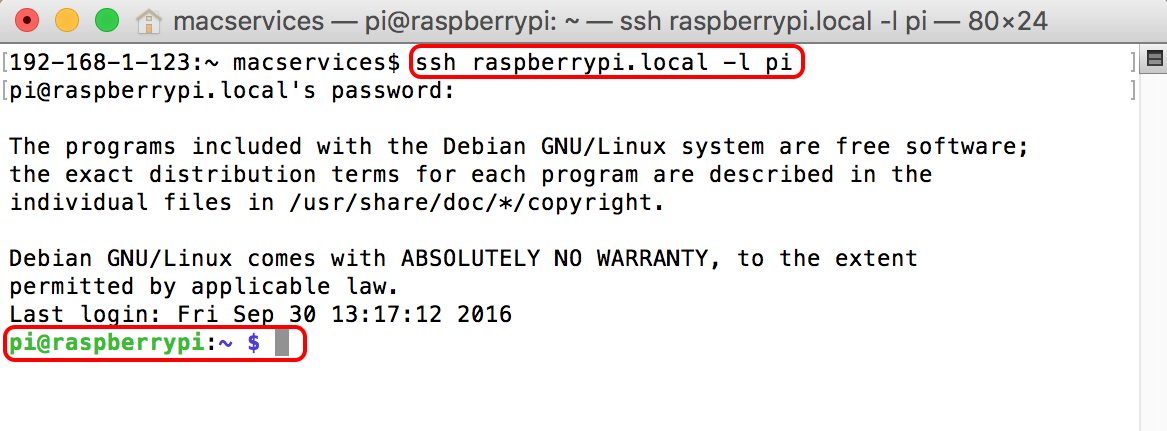


Setting up your Raspberry Pi

From a Mac OS Terminal session, SSH (Secure Shell) to the Raspberry Pi as user “pi”.

**ssh raspberry.local -l pi**

You will be prompted to enter the password for user “**pi**”. The default password is “**raspberry**”.



From the SSH session run the following commands on the Raspberry Pi.

1. Update Rasbian Linux with the latest updates. Depending on the number of updates this process can take between 5 and 20 minutes.

**sudo apt update  
sudo apt dist-upgrade  
sudo reboot**

1. Rebooting the Raspberry Pi will drop the SSH Session. You will need to re-establish the SSH session.

**ssh raspberry.local -l pi**

1. Install Apple Talk support on the Raspberry Pi. This will allow you to browse the file system on the Raspberry Pi from the Mac OS Finder.

**sudo apt install netatalk**

1. Install Remote Desktop Services on the Raspberry Pi.

**sudo apt install xrdp**

1. **Install the following pip3 (**<https://en.wikipedia.org/wiki/Pip_(package_manager)>**) packages required for the Python3 Weather Data Sample to run on the Raspberry Pi**

**sudo pip3 paho-mqtt**

**sudo pip3 pyowm**

1. **Clone the Python3** Smart Building Environment sample **to the Raspberry Pi**

**git clone https://github.com/gloveboxes/Smart-Building-Environmental-Data--Azure-IoT-Hub--Python3--MQTT.git iothub**

Installing Recommended Mac OS Packages

### Visual Studio Code IDE

Visual Studio Code is a cross platform light weight pluggable IDE that supports a wide range of programming languages. You will also need to add the Python language extension.

1. Download and install from <https://code.visualstudio.com>
2. Add the Python Language Extension from <https://marketplace.visualstudio.com/items?itemName=donjayamanne.python>
3. Review the other extensions available for Visual Studio Code from <https://marketplace.visualstudio.com/VSCode>

### Python3 Support

Mac OS ships with Python 2.x support. To run the Python3 Weather Data Simulator on your Apple Mac you will need to install Python3.

To install Python3 support see <https://www.python.org/downloads/mac-osx> and install the latest version of Python 3. At the time of writing this is 3.5.2.

### **Clone the Python3 Smart Building Environment sample to your Apple Mac**

The Smart Building Environment sample runs both on the Raspberry Pi and on your Apple Mac. When run on your Apple Mac the weather data is obtained from the Open Weather Map Service.

Running the Smart Building Environment sample will enable you to get started quickly streaming data to Azure IoT Hub. But the ultimate goal is to stream real weather from the Raspberry Pi using the Pi Sense HAT.

**cd Documents**

**git clone https://github.com/gloveboxes/Smart-Building-Environmental-Data--Azure-IoT-Hub--Python3--MQTT.git iothub**

### Mac OS Python3 PIP3 Libraries

The following pip3 packages are required on your Apple Mac to run the Python3 Smart Building Environment sample on your Apple Mac.

**pip3 paho-mqtt**

**pip3 pyowm**

**pip3 requests**

Provisioning an Azure IoT Hub Device

All devices need an identity before they can connect and stream data to or from Azure IoT Hub.

Before you can create a device identify in Azure IoT Hub you need the IoT Hub Connection String. This will either be provided to you if you by your workshop mentor or from your own instance of Azure IoT Hub.

|  |  |
| --- | --- |
| **To obtain your own IoT Hub connection string.**   1. Sign in to azure at <http://portal.azure.com> with your credentials and either create or navigate to your IoT Hub. 2. Create or navigate to your instance of IoT Hub. 3. Go to **Settings -> Shared Access Policies -> iothubowner** 4. Copy the connection string | [Get IoT Hub owner connection string](https://github.com/gloveboxes/IoT-Camp-2016/blob/master/Module2-WindowsIoTCorePi2FezHat-IoTHubs/Images/get-iot-hub-owner-connection-string.png?raw=true) |

### Device Identity Management with Node.js [Azure IoT Hub Explorer](https://www.npmjs.com/package/iothub-explorer)

[Azure IoT Hub Explorer](https://www.npmjs.com/package/iothub-explorer) is an easy Node.js command line tool to:-

1. Create, delete, and list device identities
2. Send cloud to device messages
3. Listen to device to cloud messages

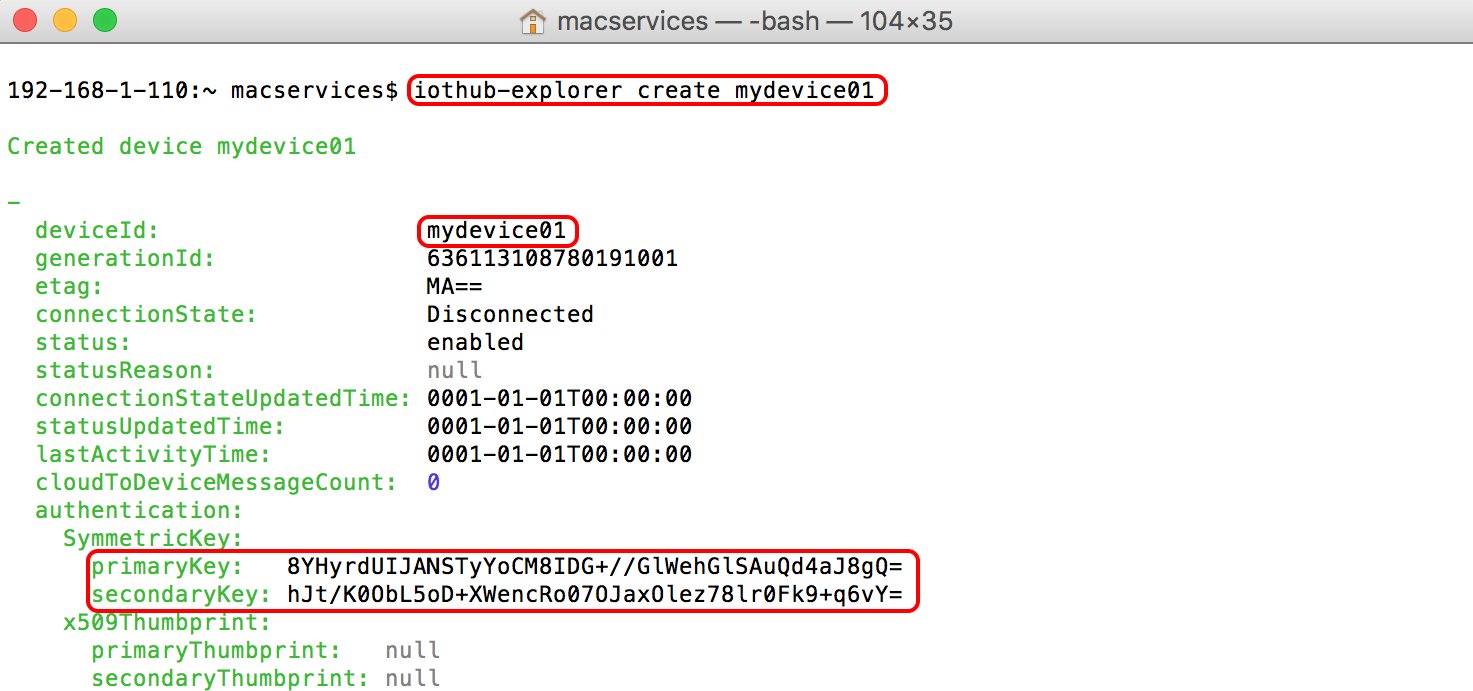
|  |  |
| --- | --- |
| Summary on Azure IoT Hub Explorer | |
| Install Node.js | [www.nodejs.org](http://www.nodejs.org) |
| Install Azure IoT Hub Explorer | **npm install -g iothub-explorer@latest** |
| Help | **iothub-explorer help** |
| Cache the Connection String | **iothub-explorer login <connection-string>** |
| Create a new device identity | **iothub-explorer [<connection-string>] create <device-id>** |
| Delete a device identity | **iothub-explorer [<connection-string>] delete <device-id>** |
| Get device information | **iothub-explorer [<connection-string>] get <device-id>** |
| Send a cloud to device message | **iothub-explorer [<connection-string>] send <device-id> <msg>** |
| Monitor device to cloud messages | **iothub-explorer <connection-string> monitor-events <device-id>** |

Example usage

**iothub-explorer login "HostName=YourIoTHub.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=iVfr5G12UcCBQuju8OD98TAP0gmORQsLSXe0TPmaNEw="**

### **Create an Azure IoT Hub Device Identity**

**iothub-explorer create mydevice01**

****

**Be sure to copy one of the keys as you’ll need when configuring the Smart Building Environment app.**

Smart Building Environment Sample Configuration

The main Smart Building Environment sample is located in the **IoTHub/smartbuilding** directory.

A configuration file is passed in to the environment.py app at start-up time. It contains the Azure IoT Hub address, the device identity including the shared access key, the sensor module to load, along with the Open Weather Map API key and location.

Sample code includes support for the following HATS

1. [Open Weather Map](http://openweathermap.org/) Virtual Environmental Sensor HAT: sensor\_openweather. This virtual Environment HAT can be used on your Apple Mac or the Raspberry Pi as it does not require as specific hardware and gets environmental data from the free Open Weather Map service.
2. [Raspberry Pi Sense HAT](https://www.raspberrypi.org/products/sense-hat/): sensor\_envirophat
3. [Enviro pHAT](https://shop.pimoroni.com/products/enviro-phat): sensor\_sensehat

The following is an example for the Raspberry Pi Sense HAT (**sensor\_openweather.json)**

{

"IotHubAddress":"YourIoTHub.azure-devices.net",

"DeviceId":"rpi3mlb",

"SharedAccessKey":"uJ21qp9LUvjkohipkXycvb7RoYwmUDE+4gXyIYS00feZg=",

"SensorModule":"sensor\_openweather",

"OpenWeatherMapApiKey":"c2044448a2f55555925f27b9e21296dd",

"OpenWeatherMapLocationId":"Melbourne, AU"

}

The configuration file is passed in as a start-up argument. The following are examples of running the Smart Buildings sample app using the three HAT configuration files included in the sample.

* **python3 environment.py sensor\_openweather.json**
* **python3 environment.py sensor\_sensehat.json**
* **python3 environment.py sensor\_envirophat.json**

### Open Weather Sample in Actions

When the app starts successfully it will display the sent message count and the device id.

****

The data is serialised in the following JSON format.

{"Geo":"Melbourne, AU","Humidity":50,"HPa":1011,"Celsius":18.40,"Light":0,"Id":199783}

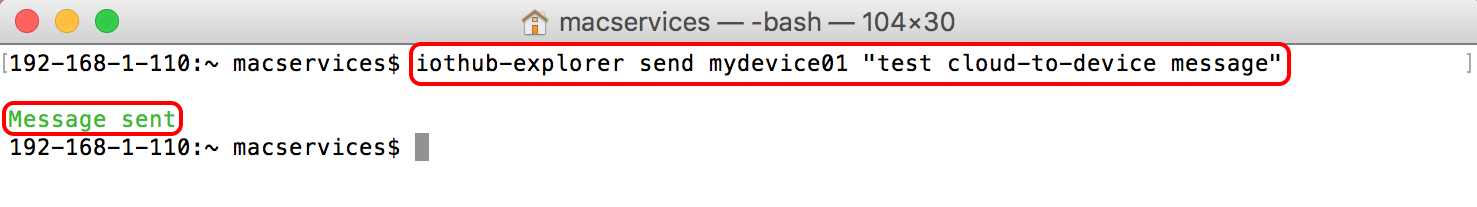
**Monitoring Azure IoT Hub Device to Cloud Messages**

**iothub-explorer "HostName=YourIoTHub.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=iVfr5G12UcCBQuju8OD98TAP0gmORQsLSXe0TPmaNEw=" monitor-events mydevice01**

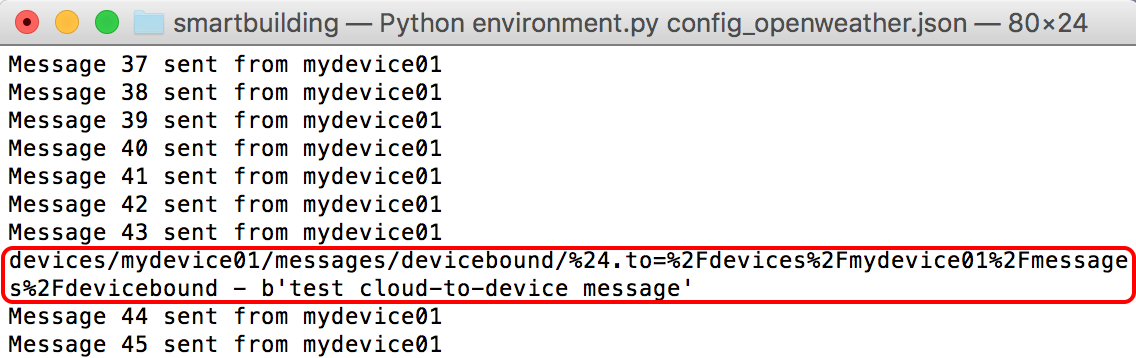
****

**Send a Cloud to Device Message**

**iothub-explorer send mdevice01 “test cloud-to-device message”**

****

**Message received by the Smart Building Environment Python3 app.**

****

Developer Workflow

Visual Studio Code with the Python Extension added is an excellent way to edit the Smart Building sample.

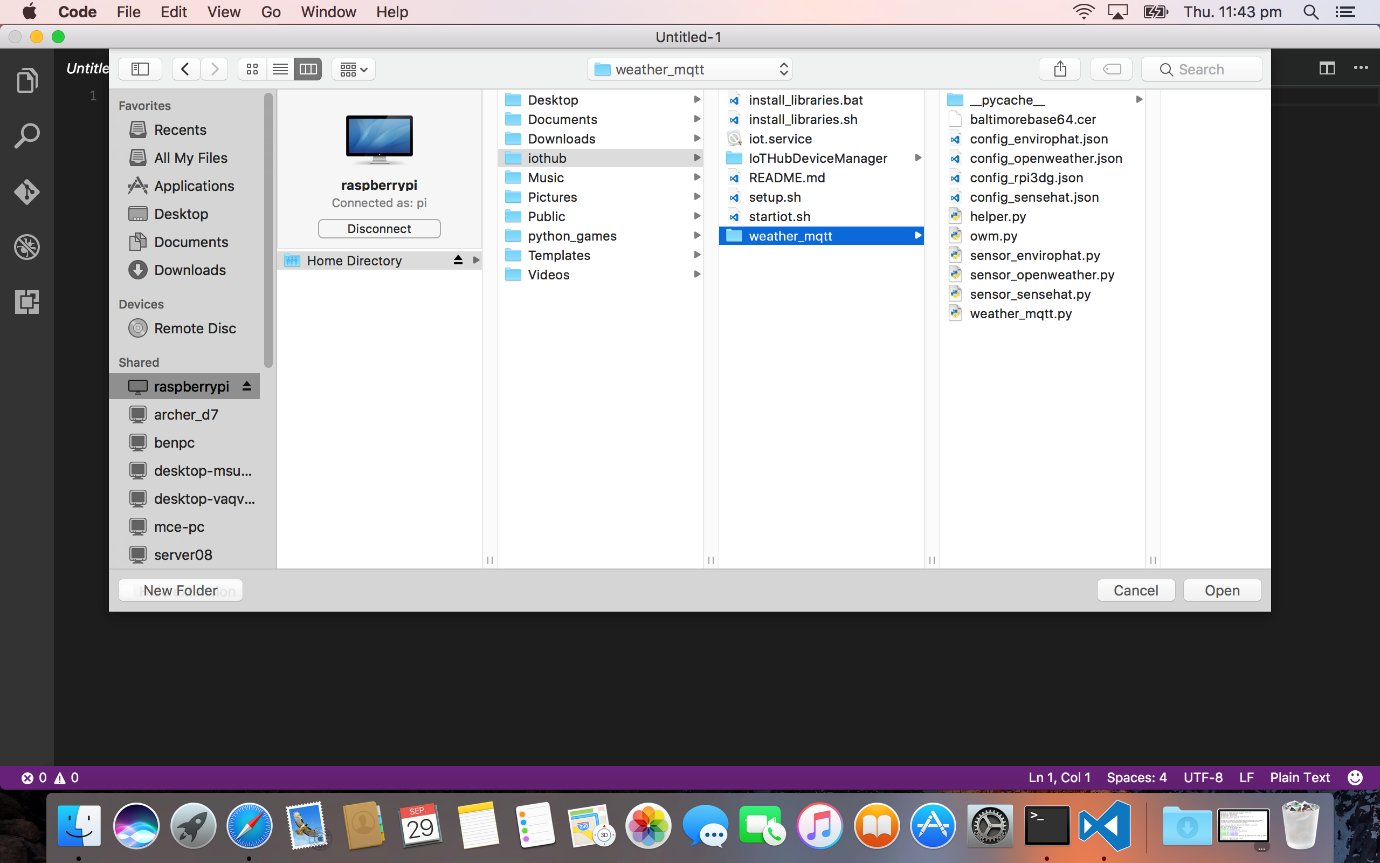
If you are running the environment app on your Apple Mac with the Open Weather Sensor Virtual HAT then you can edit and debug the app all from your Apple Mac.

If you are editing the Smart Building sample on the Raspberry Pi then there are a couple of options.

### **Visual Studio Code**

Given you installed the Apple Talk package on the Raspberry Pi the Raspberry Pi file system will appear in the Mac OS Finder. You will need to provide network credentials for the Raspberry Pi, which are user ‘pi’ and password ‘raspberry’. Then open the IoTHub\smartbuilding at the directory level, this will load all the files in the solution and you can start editing.

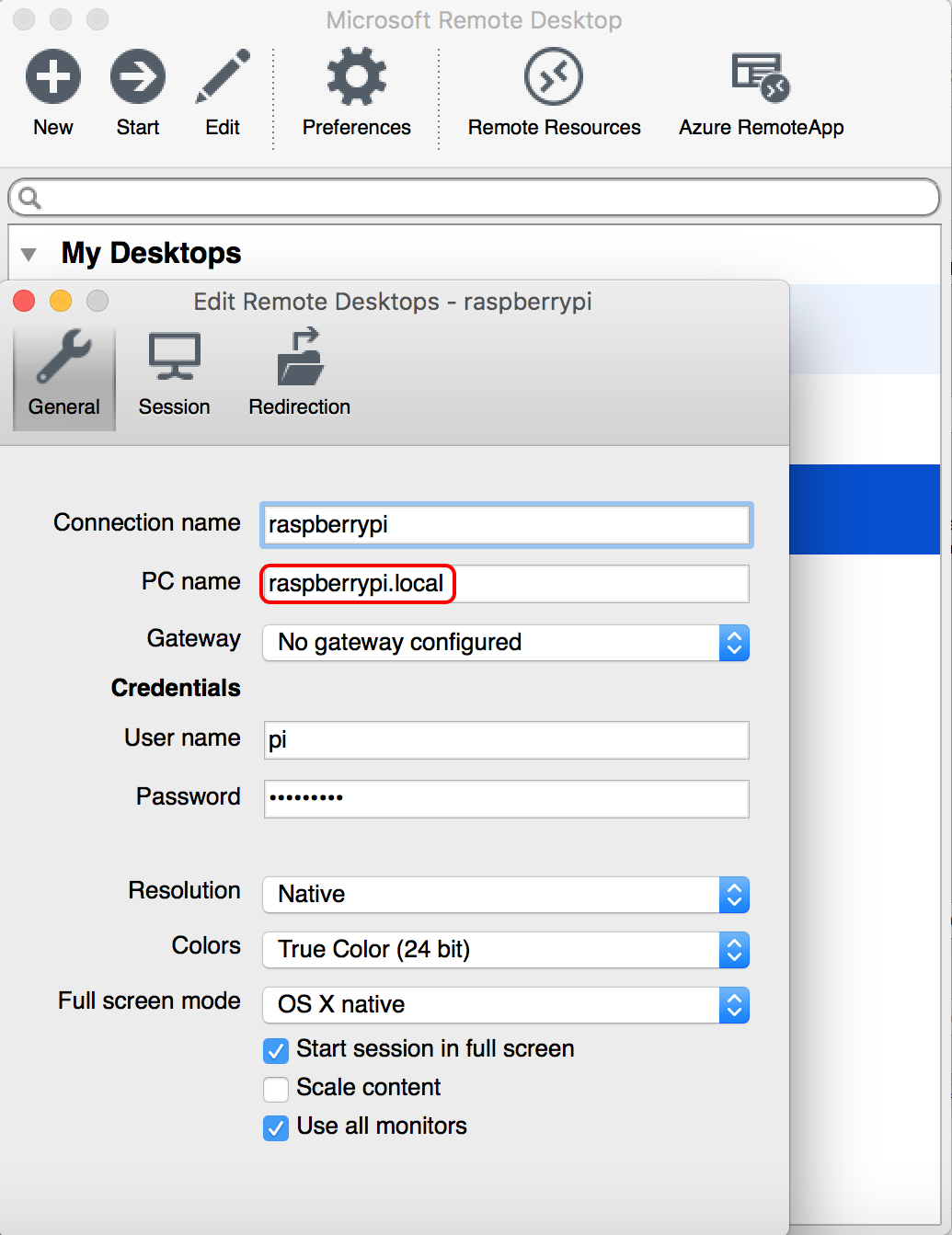
This approach will provide you with a rich editing intelisense environment but you will need to SSH in to the Raspberry Pi separately to run the app.



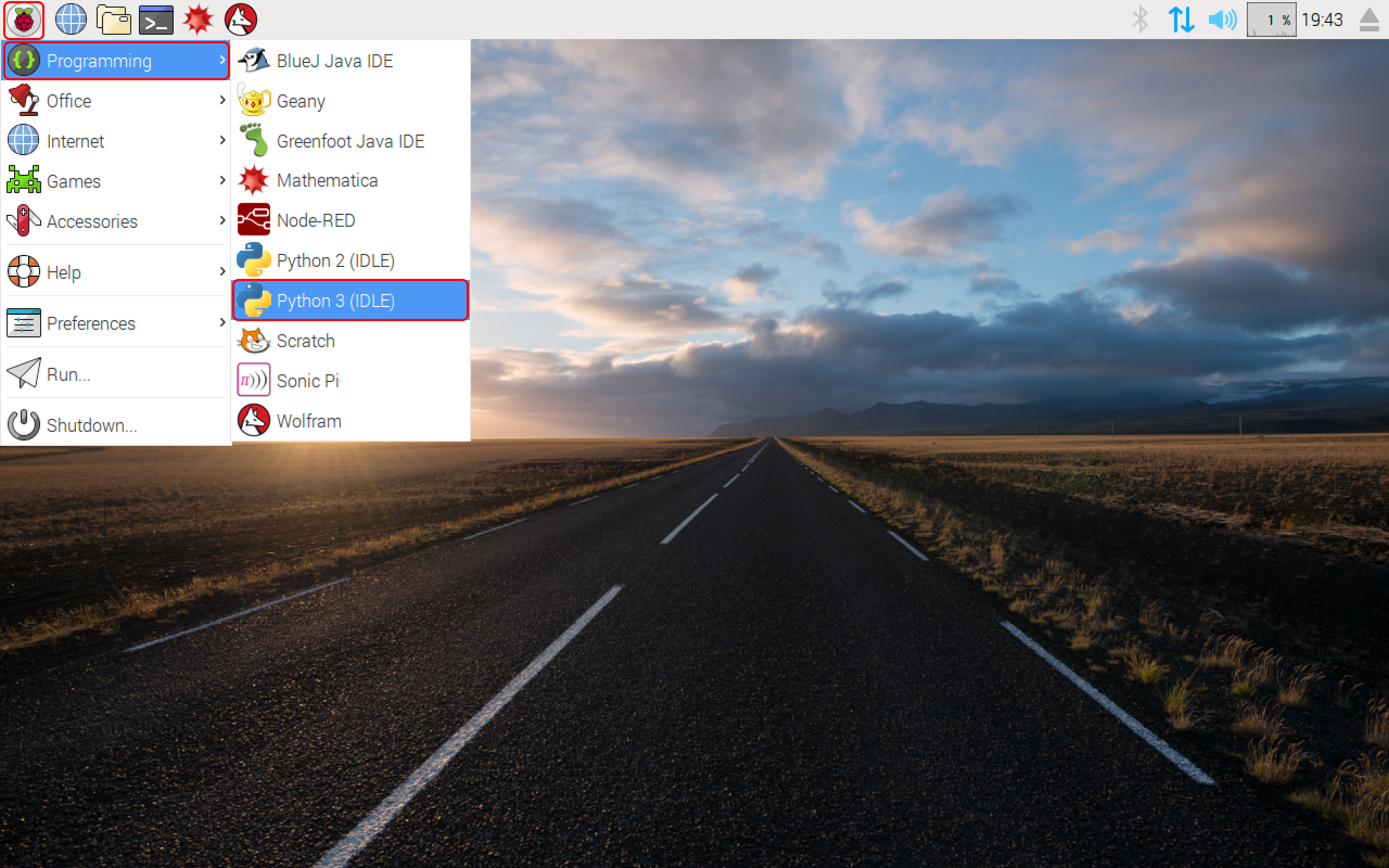
### **Microsoft Remote Desktop**

Microsoft Remote Desktop along with the xrdp Raspberry Pi package allows you to start a remote desktop session with your Raspberry Pi from your Apple Mac. You can edit/run/debug the Smart Building app on the Raspberry Pi itself with the editor of your choice such as nano or idle3.

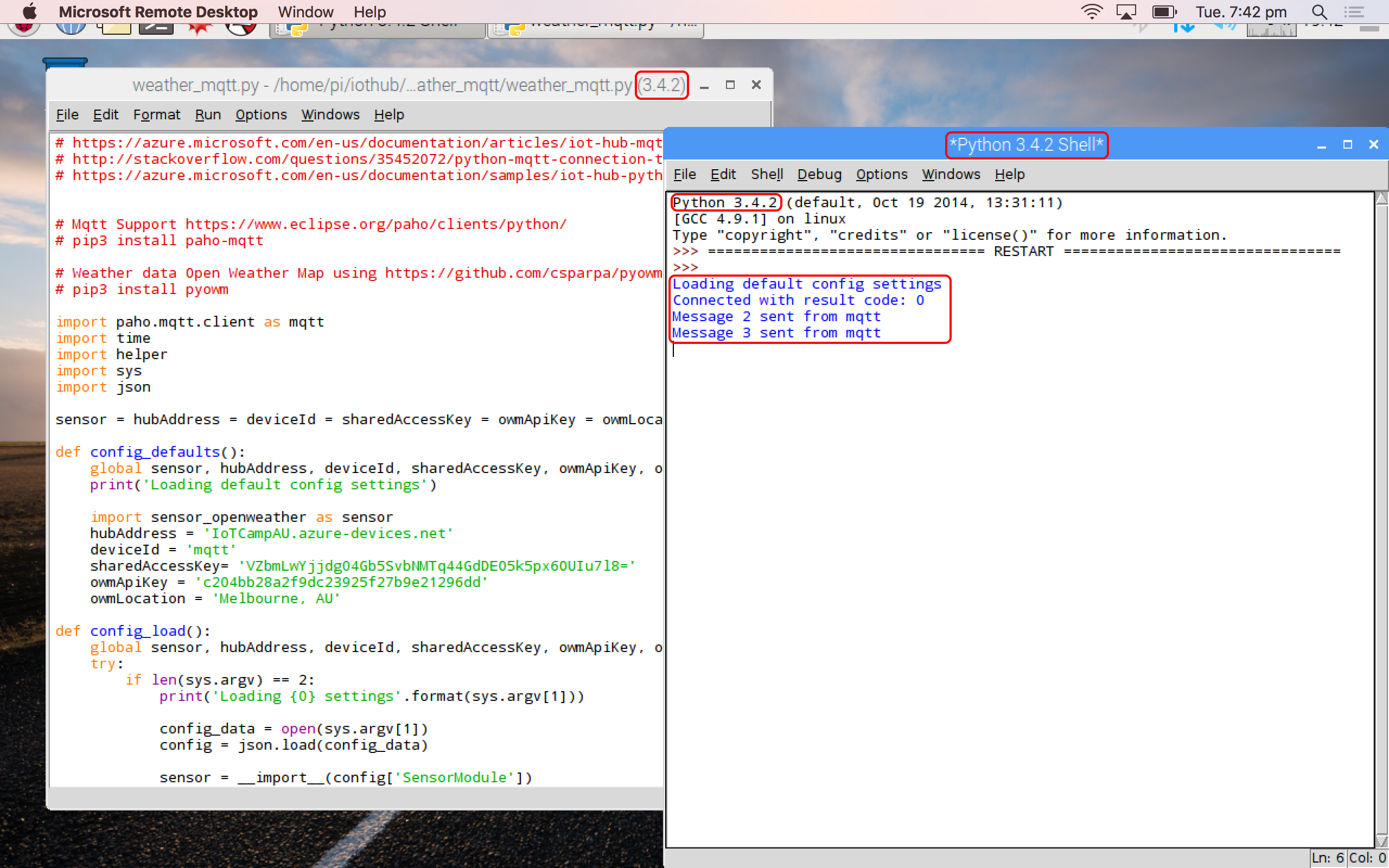
Configure Microsoft Desktop as follows.



Start Python 3 (IDLE) and open the environment.py sample add from the IoTHub/smartbuilding folder.



Once the Smart Building sample app has opened you can edit/run/debug and make enhancements to the sample app.



Azure IoT Resources for Apple Mac Based Developers

### Node.js IoTHub Explorer

<https://www.npmjs.com/package/iothub-explorer>

This sample has some extended capabilities over the Python3 IoT Hub Devices sample most notably being able to send a **cloud-to-device messages**.

### Python Resources

Getting started with IoT Hub REST API and Python

1. <https://azure.microsoft.com/en-us/documentation/samples/iot-hub-python-get-started>

### Node.js

Getting started with IoT Hub REST API and Python

1. <https://azure.microsoft.com/en-us/documentation/articles/iot-hub-node-node-getstarted/>
2. <https://github.com/juanjperez/azure-iot-device-management>
3. <https://www.npmjs.com/package/iothub-explorer>