**Table of Contents**

[1.1. Introduction: 3](#_Toc520202876)

[1.2. Solution Overview 3](#_Toc520202877)

[1.2.1. Solution Description 3](#_Toc520202878)

[1.2.2. IoT Device: 3](#_Toc520202879)

[1.2.3. Mosquitto MQTT Broker: 3](#_Toc520202880)

[1.2.4. Azure IoT Hub: 6](#_Toc520202881)

[1.2.5. Python Script: 7](#_Toc520202882)

## Introduction:

This document serves as a practical guide to connect an Internet of Things (IoT) device to the Azure IoT Hub. This guide is intended for devices communicating over the MQTT protocol. This guide assumes you have a version of Python installed on the server machine.

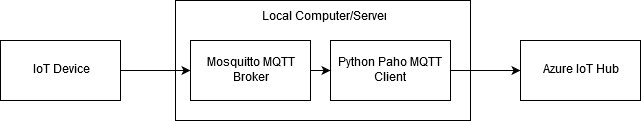
## Solution Overview

4.

3.

2.

1.



### Solution Description

The solution involves connecting the IoT device to the local server. The server has the Mosquitto MQTT Broker installed on it. The server also has a Python script which has the Paho MQTT client which is able to read the data from the Mosquitto broker. The Python script then publishes the data to the Azure IoT Hub.

The following sections will detail how to set up each device to ensure communication from the device to the IoT Hub.

### IoT Device:

For this example, an Android cell phone is used. The use of a phone running iOS will likely work in a similar, although this is untested. The cell phone has the Sensor Node Free app installed. This allows the streaming of phone sensor data (accelerometer, gyroscope, etc.) via the MQTT or XML protocols.

* 1. From the Play Store (Android) or iStore (Apple), download and install “Sensor Node Free”.
  2. Open the app. From the main menu -> *STREAM* -> *STREAM LIVE DATA(MQTT).*
  3. Select the sensors you want to stream.

### Mosquitto MQTT Broker:

* 1. The Moquitto MQTT Broker has been developed by Eclipse, and can be downloaded from their website. However, there are several dependencies the application has in terms of .dll files. The folder with all the required files can be downloaded by following this link:

<http://www.steves-internet-guide.com/wp-content/uploads/mos1.14.7z>

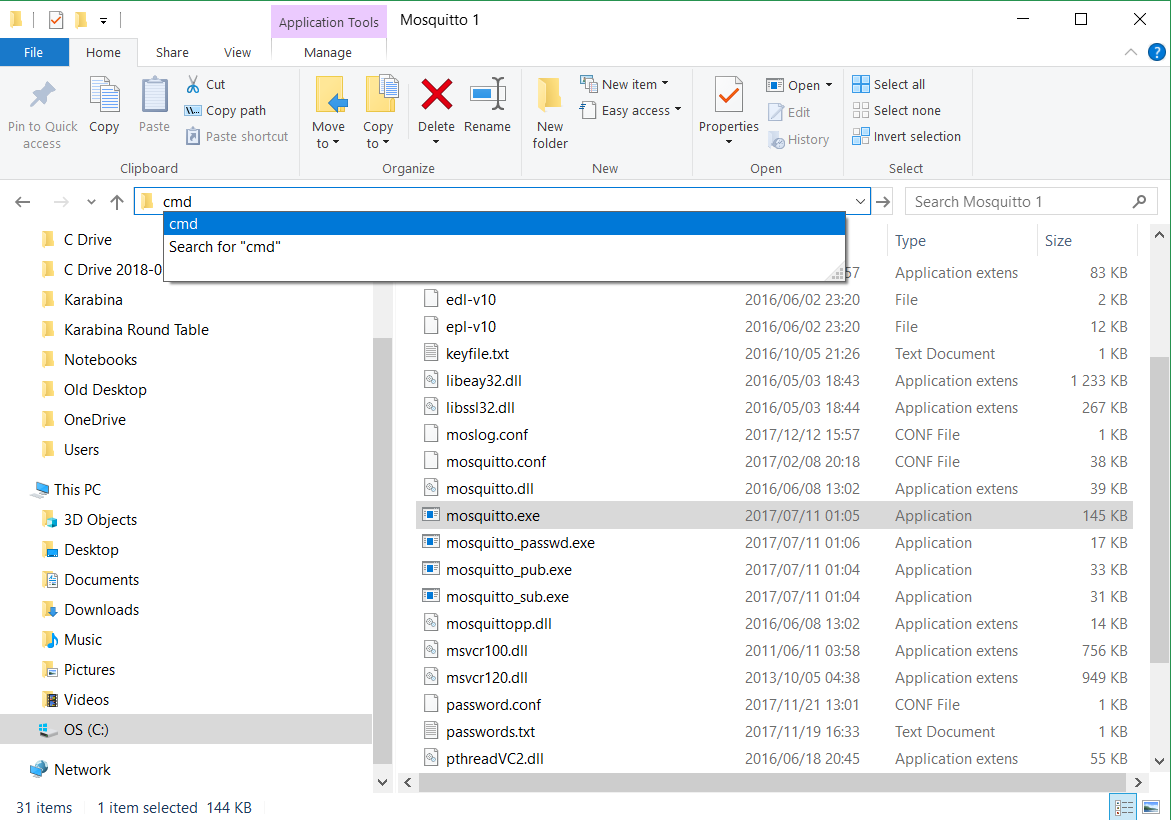
* 1. To unzip the folder, you must download and install 7-Zip.

<https://www.7-zip.org/>

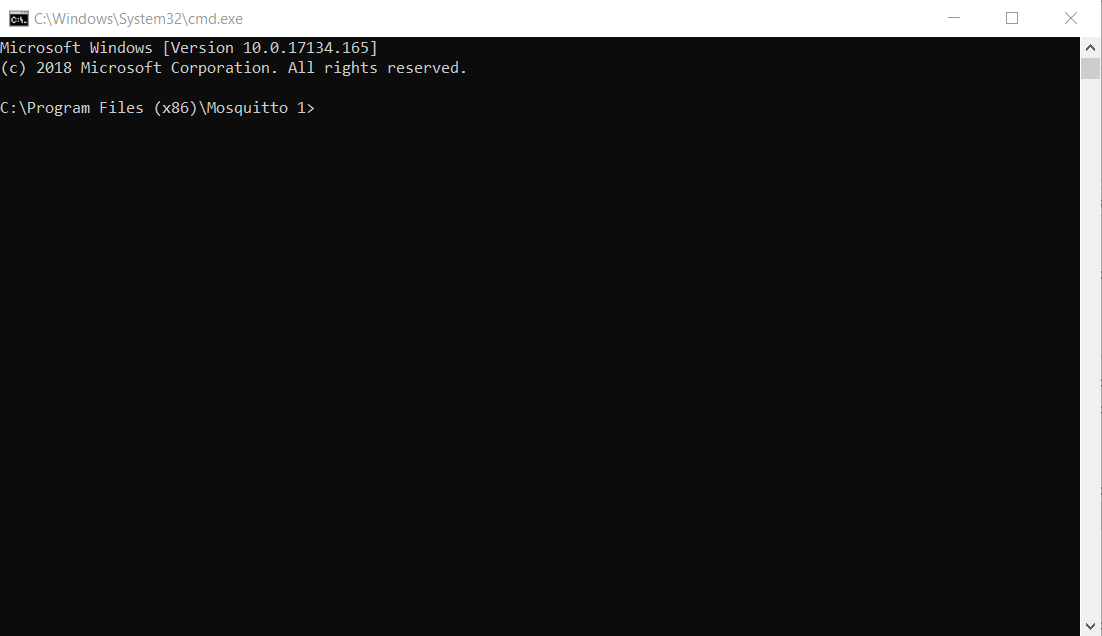
* 1. Unzip the folder and copy the contents to a folder on your C drive. For example, copy the contents to the following path:

C:\Program Files (x86)\mosquitto\

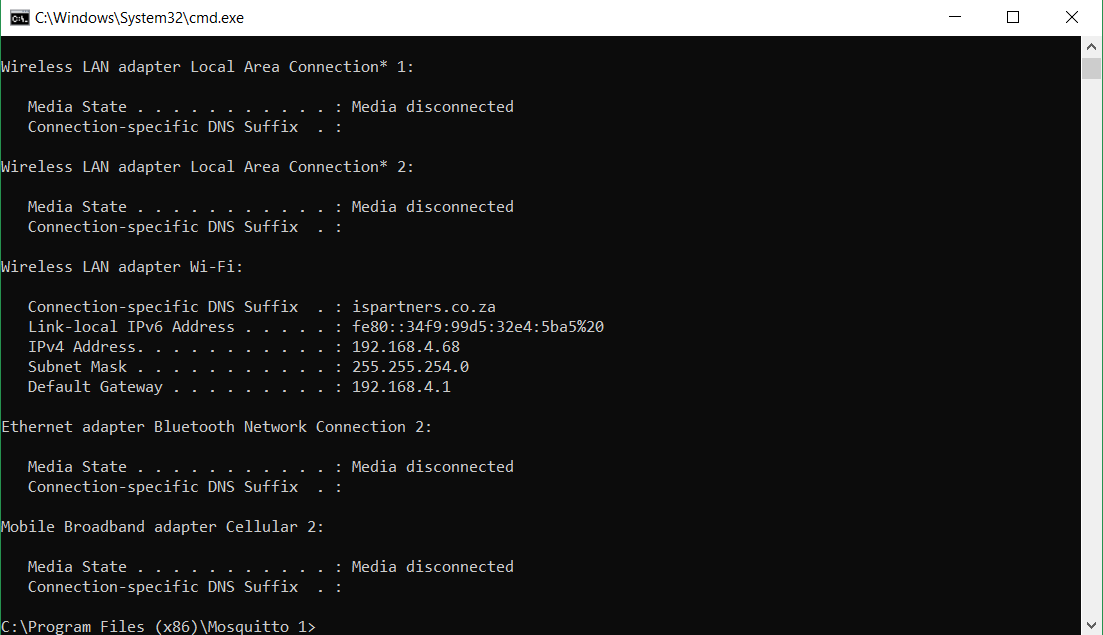
* 1. Run moquitto.exe by double left-clicking it. A command prompt window will open, but it will remain blank.



* 1. In the bar containing the current file path: Clear the contents, type *cmd* and press Enter. A new command prompt window will open in the current file directory.



* 1. Type *ipconfig* and press enter.

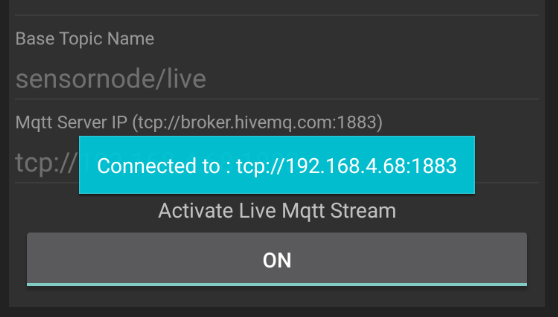
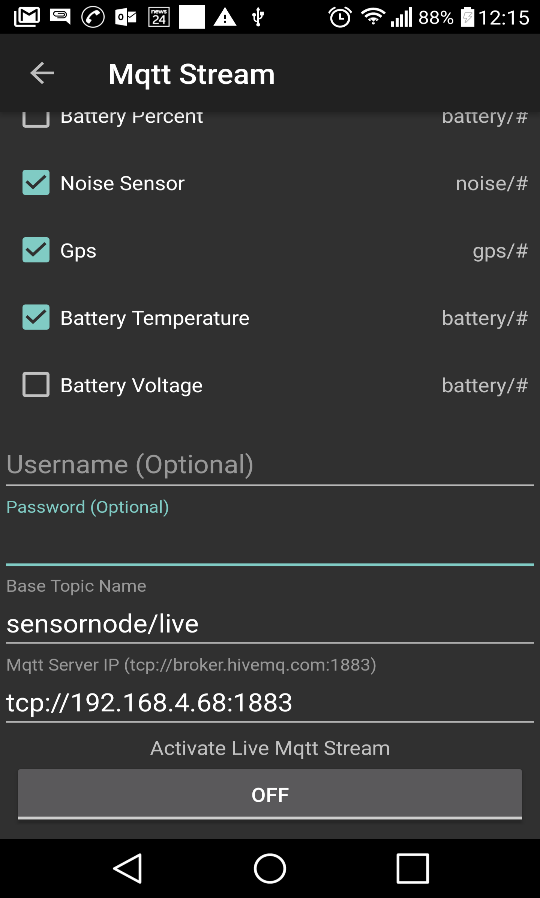


* 1. Find the IPv4 Address. Take note of this address.
  2. Type *mosquitto* and press Enter. The mosquito service is now running.
  3. Type *mosquitto* *–p 1883*. This instructs mosquito to listen to port 1883.
  4. In the Sensor Node Free app, in the *Mqtt Server IP* dialog box, type the following:

tcp://<IPv4 Address>:<Port>

In the current example:

tcp://192.168.4.68:1883



1.

2.

1. MQTT Topic

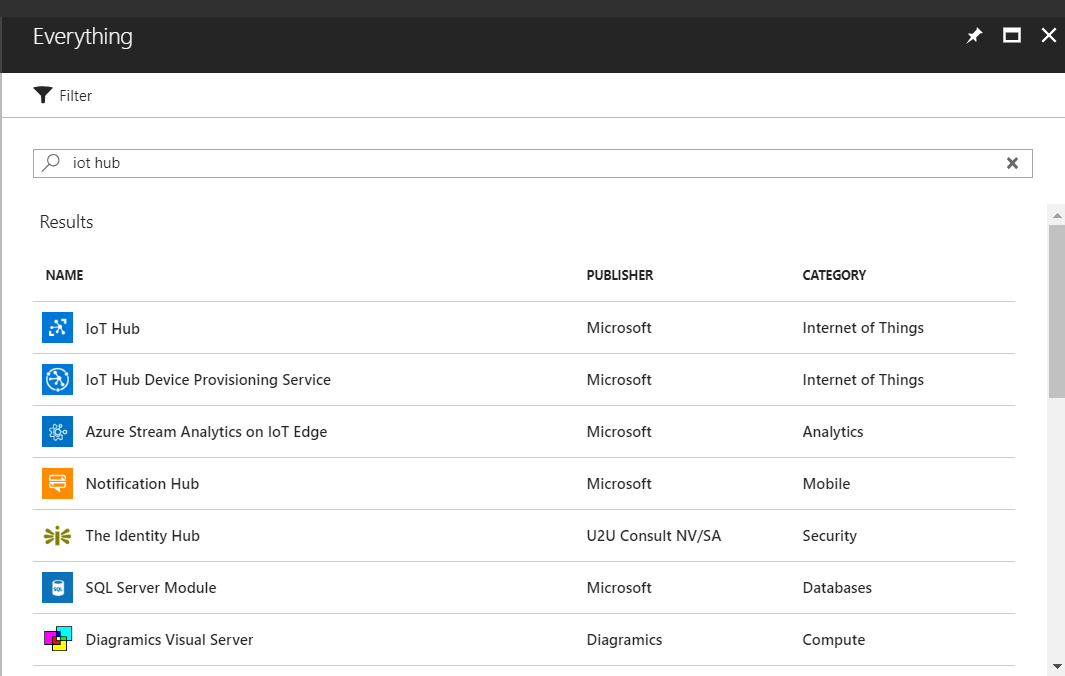
2. IPv4 address.

* 1. Now tap the *OFF* button. There will be a confirmation that the device is connected. The device is now sending the selected sensor readings to the MQTT Broker.

### Azure IoT Hub:

Before setting up the Python script, we first need our Azure IoT Hub to be created and we need out login credentials.

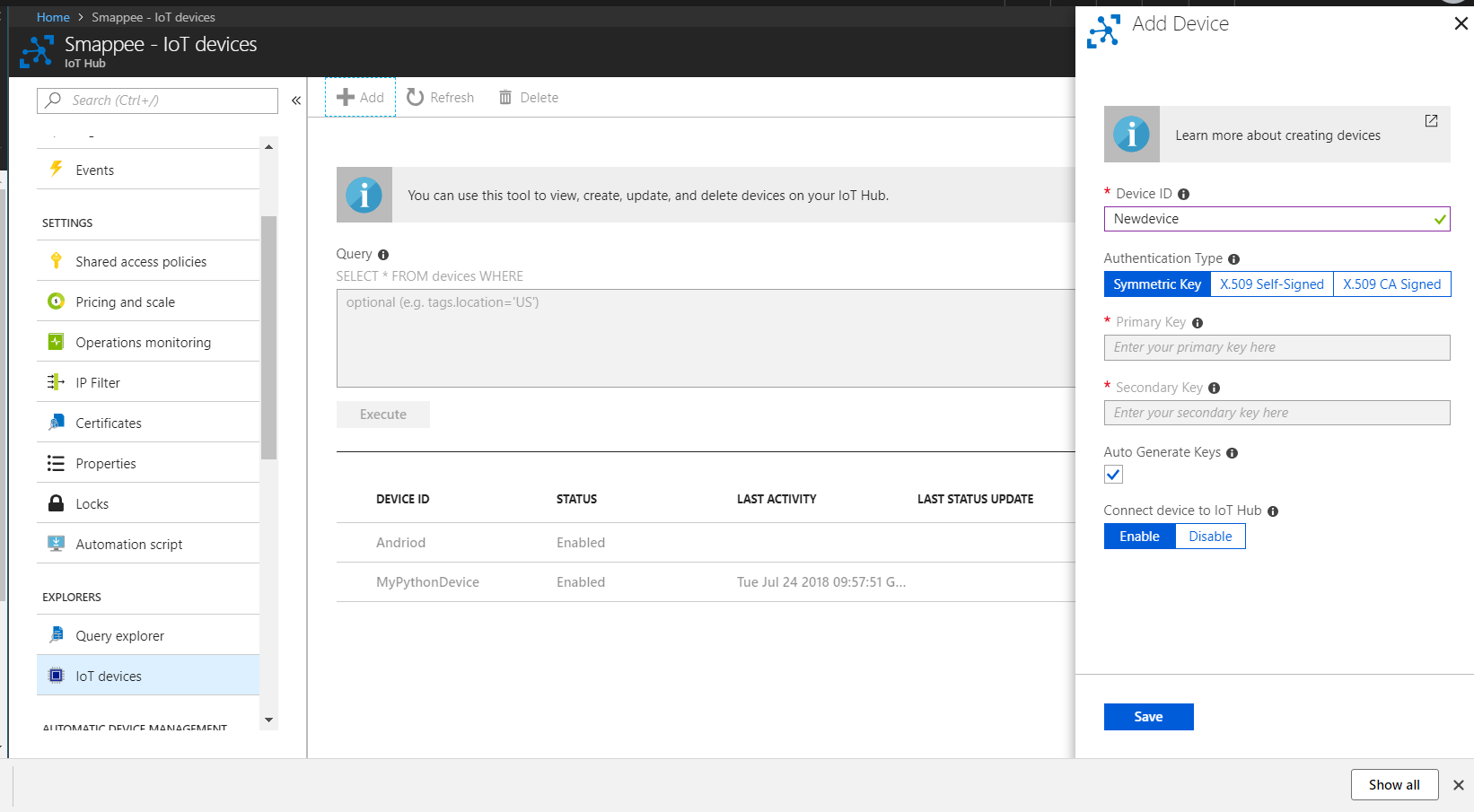
* 1. Navigate to portal.azure.com. Log in
  2. Create a new Azure IoT Hub. Click *Create a resource* (The green plus on the top left of the portal).
  3. Search for IoT Hub.



Create the resource group and choose the free pricing tier. Give your IoT Hub a name. In this example, it is called *Smappee.*

For more details, follow this link: <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-create-through-portal>

* 1. One the IoT Hub created, create a new device.



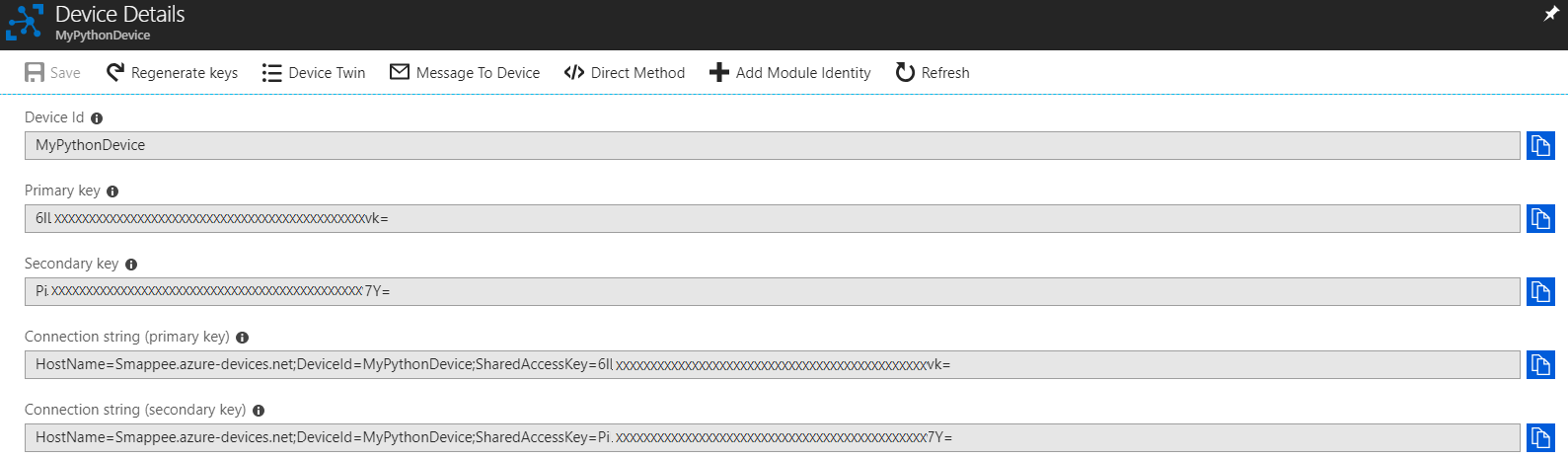
3.

4.

2.

1.

1. In the options pane.
2. look for *IoT Devices.*
3. Click +*Add*
4. Give the device a name and *Save.*
   1. Once the device has been added, it will appear in the devices list. Select the device you have just created.
   2. In the following screen, copy the connection string.



### MQTT Topics

When dealing with connection of devices to IoT hubs, you must be familiar with the concept of MQTT topics. An MQTT topic is an identifier for the message which the IoT device is sending. In the current example, the IoT device will publish a message in a topic. The word *topic* refers to an UTF-8 string that the broker uses to filter messages for each connected client. The topic consists of one or more topic levels. Each topic level is separated by a forward slash (topic level separator). For example, “myhome/groundfloor” may be the topic which is being published by the device. Each level of the topic is separated by a forward slash. In the current example, the topic being published is being published as “sensornode/live”. Each new sensor reading will be delivered in its own topic. For example, the gyroscope reading for the x-axis will be sent within the topic “sensornode/live/Gyroscope/x”. The MQTT broker will need to subscribe to this topic to receive the sensor data.

### Python Script:

The Python script can be run in a Jupyter notebook. For this script to be able to run, the installation of Python requires the following two packages:

iothub\_client

paho-mqtt

If you have an Anaconda installation of Python, then run the Anaconda Prompt as Administrator, and install the packages using the following commands:

* pip install paho-mqtt
* pip install azure-iothub-device-client
* pip install azure-iothub-service-client

Copy and paste the following script into your preferred Python IDE, and change the following items:

Change the CONNECTION\_STRING to the one copied from the step above.

Enter the IP address in the quotation marks for the *broker* variable.

Ensure the MQTT topic you have subscribed to is the same as the MQTT topic being published by the device.

# Copyright (c) Microsoft. All rights reserved.

# Licensed under the MIT license. See LICENSE file in the project root for full license information.

import paho.mqtt.client as mqtt

import random

import time

import sys

# Using the Python Device SDK for IoT Hub:

# https://github.com/Azure/azure-iot-sdk-python

# The sample connects to a device-specific MQTT endpoint on your IoT Hub.

import iothub\_client

# pylint: disable=E0611

from iothub\_client import IoTHubClient, IoTHubClientError, IoTHubTransportProvider, IoTHubClientResult

from iothub\_client import IoTHubMessage, IoTHubMessageDispositionResult, IoTHubError, DeviceMethodReturnValue

# The device connection string to authenticate the device with your IoT hub.

# Using the Azure CLI:

# az iot hub device-identity show-connection-string --hub-name {YourIoTHubName} --device-id MyNodeDevice --output table

CONNECTION\_STRING = "<YOUR CONNECTION STRING>"

# Using the MQTT protocol.

PROTOCOL = IoTHubTransportProvider.MQTT

MESSAGE\_TIMEOUT = 10000

# Define the JSON message to send to IoT Hub.

TEMPERATURE = 20.0

HUMIDITY = 60

MSG\_TXT = "{\"temperature\": %.2f,\"humidity\": %.2f}"

def send\_confirmation\_callback(message, result, user\_context):

print ( "IoT Hub responded to message with status: %s" % (result) )

def iothub\_client\_init():

# Create an IoT Hub client

client = IoTHubClient(CONNECTION\_STRING, PROTOCOL)

return client

def on\_log(client, userdata, level, buf):

print("log: " + str(buf))

def on\_connect(client, userdata, flags, rc):

if rc == 0:

print("Connection successful")

else:

print("Bad connection. Returned code: ", rc)

def on\_disconnect(client, userdata, flags, rc = 0):

print("Disconnect result code: " + str(rc))

data = ''

def on\_message(client, userdata, msg):

global data

topic = msg.topic

m\_decode = str(msg.payload.decode("utf-8", "ignore"))

msg\_out = "Message received: ", topic, ' ', m\_decode

data = m\_decode

broker = "<YOUR IP ADDRESS>"

m\_client = mqtt.Client("1")

#m\_client = mqtt.Client()

m\_client.on\_connect = on\_connect

m\_client.on\_disconnect = on\_disconnect

#client.on\_log = on\_log

m\_client.on\_message = on\_message

m\_client.username\_pw\_set()

print("Connecting to borker: ", broker)

m\_client.connect(broker)

m\_client.loop\_start()

def iothub\_client\_telemetry\_sample\_run():

global data

try:

client = iothub\_client\_init()

print ( "IoT Hub device sending periodic messages, press Ctrl-C to exit" )

while True:

# m\_client.subscribe("sensornode/live/Gyroscope/z")

m\_client.subscribe("sensornode/live/Accelerometer/z") #Your MQTT topic

# m\_client.subscribe("sensornode/live/Accelerometer/x")

print(data)

# Build the message with simulated telemetry values.

temperature = TEMPERATURE + (random.random() \* 15)

humidity = HUMIDITY + (random.random() \* 20)

msg\_txt\_formatted = MSG\_TXT % (temperature, humidity)

message = IoTHubMessage(data)

# Add a custom application property to the message.

# An IoT hub can filter on these properties without access to the message body.

# prop\_map = message.properties()

# if temperature > 30:

# prop\_map.add("temperatureAlert", "true")

# else:

# prop\_map.add("temperatureAlert", "false")

# Send the message.

#print( "Sending message: %s" % message.get\_string() )

client.send\_event\_async(message, send\_confirmation\_callback, None)

time.sleep(1)

except IoTHubError as iothub\_error:

print ( "Unexpected error %s from IoTHub" % iothub\_error )

return

except KeyboardInterrupt:

print ( "IoTHubClient sample stopped" )

if \_\_name\_\_ == '\_\_main\_\_':

print ( "IoT Hub Quickstart #1 - Simulated device" )

print ( "Press Ctrl-C to exit" )

iothub\_client\_telemetry\_sample\_run()