This document describes how to set up the AWS IoT Core with the raspberry Pi and also how to setup the AWS SDK for python on the raspi and then modify the program AWSPubSub.py that can publish or subscribe to IoT Core. Here this program has the code to send the Raspi Emulator data to Aws.

* Follow this to create a thing, it’s policy and the necessary certificates. The naming conventions in this link are specific to that example and hence can be modified as required.

<https://docs.aws.amazon.com/iot/latest/developerguide/iot-gs-first-thing.html>

Stop after “Create AWS IoT Resources”

* After setting up the AWS online, follow this page to set up the sdk on the raspberry pi. Do only step 1
* <https://docs.aws.amazon.com/greengrass/latest/developerguide/IoT-SDK.html>

In 1)c) instead of running “sudo python setup.py install” run “sudo python3 setup.py install”

The first command installs the sdk for python2 only, so when we try to run any program on the Thonny editor which uses python3, it will throw module errors. To have the sdk for both python versions, run both of those commands.

* Step 2 is to run a sample pubsub program. It can be found in “/home/pi/aws-iot-device-sdk-python/samples/basicPubSub/basicPubSub.py”.
* This program is a simple program that can either publish or subscribe to a particular topic in the AWS IoT Core.
* Since the objective is to send data to the cloud, we modify the code under “# Publish to the same topic in a loop forever”.
* Inside this loop we place the code from IMU\_Data.py which gets the data from the emulator.
* In the code we add a device id and also send a timestamp at which the data is collected(timestamp created using same code used to create file name in IMU\_data.py).
* Milliseconds were also added to the timestamp because we get data multiple times within a second which means that two or more sets of data sent to the cloud will have the same primary keys(device id, timestamp) and hence it wont be stored because of the duplicate keys. So, we specify milliseconds so that all the data is stored.
* Data collection interval is specified with sleep in the last line but it was removed because the program itself takes nearly 0.7-0.9 sec to run every time it collects data. So, only if we want an interval higher than that, we have to specify a sleep. Here, since we want to collect data at 0.1 sec intervals, a sleep is not required.
* The message is sent in the JSON format.
* This program can be executed with:

basicPubSub.py [-h] -e HOST -r ROOTCAPATH [-c CERTIFICATEPATH]

[-k PRIVATEKEYPATH] [-p PORT] [-w] [-id CLIENTID]

[-t TOPIC] [-m MODE] [-M MESSAGE]

* In order to decide whether you want to publish or subscribe, use the –m tag which can be set to both, publisher, subscriber.
* For ex: python3 AWSPubSub.py -t raspi/imu -r ~/certs/Amazon-root-CA-1.pem -c ~/certs/device.pem.crt -k ~/certs/private.pem.key -m publish -e a1pwd0oorzzmbf-ats.iot.us-east-2.amazonaws.com
* This publishes to the topic raspi/imu and specifies the path in which the “thing’s” certificated are located at.