EEC 172 Lab 4: Introduction to AWS and RESTful API I
Section: A03

Manprit Heer
912839204
mkheer@ucdavis.edu

Cathy Hsieh
912753571
cchsieh@ucdavis.edu

OBJECTIVE

This lab we explore the RESTful API and use it to connect to Amazon Web Services. We create a 'thing' in AWS and use the HTTP protocol to GET status information about our thing. We will also utilize and familiar ourselves with POST. This lab will help is establish an AWS account and familiarize ourselves with the AWS platform for the labs to come.

INTRODUCTION

In the first part, we are to connect to AWS using the documentation provided and update our Shadow using the HTTP protocol. This is a straight forward lab, as the documentation on how to get create a shadow and update it is provided in the lab. After setting up the account, we set up our 'thing' and generate the required certificates and keys to the device. After this, we download the files onto the board using the CCS UniFlash tool we demo'd in Lab 1. Finally we will make a policy to allow updates and status requests from the thing shadows.

MATERIALS AND METHODS

The equipment needed for this lab are listed as follows:

1. CC3200 LaunchPad (CC3200-LAUNCHXL)
2. USB Micro-B plug to USB-A plug Cable
3. An Amazon AWS account
4. OpenSSL converter
5. AT&T Universal IR Remote

CC3200 LAUNCHPAD[1]

PROCEDURE

Part I. Connecting to AWS and Updating Shadow

Setting up an Amazon AWS account

We create a single AWS account, which will allow us to further our understanding of AWS. After setting up the account, we log into the console and go to the IoT Core service. We also make sure that the U.S. West (Oregon) AWS server is chosen for this lab, as it is recommended.

In IoT Core, we click on the interactive tutorial and continue to learn more about the system through high-level tutorials.

Setting Up Your First Device Thing/Shadow with IoT Console

We follow the step-by-step documentation provided in the lab to create our device thing/shadow. Then we generate the required certificates and keys and download them as they will not be made available afterwards. We then Activate the certificates and attach a policy to the "Thing" we made under the name of "CC3200_Thing".

Making a Policy to Allow Update/Get Status from Thing Shadows

To represent our 'thing' on the cloud, AWS uses the concept of a device shadow, which pushes the new state to the server and retrieves back the previous state through HTTP GET/POST commands. We must first take appropriate security measures by making sure that the device has the' privilege' to perform these actions.

To do this, we create a 'Policy' which we then proceed to attach to the certificates that we have created for the object.

Converting the Keys/Certificates for Use with the CC3200

In order to convert the keys and certificates, which include the public/private key, a certificate, and a root certificate. To do this, we must convert the files from .pem format to .der format so that the files will be recognized and be in effect for the CC3200 LaunchPad. We were having trouble using the OpenSSL program on the lab desktop computer so we decided to download it ourselves on our personal laptop. After a successful download, we

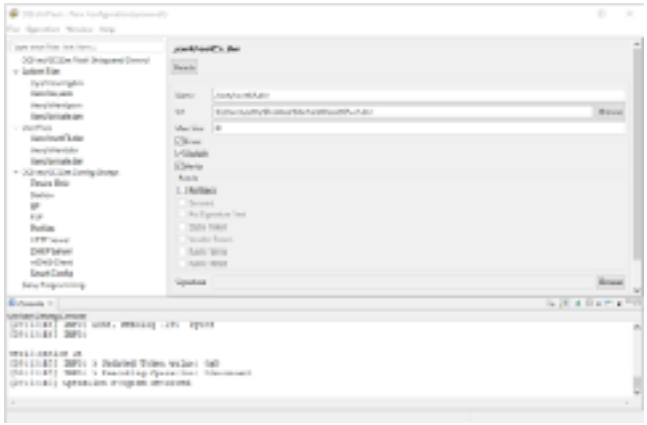
```
>openssl x509 -outform der -in c:\...\rootCA.pem -out c:\...\rootCA.der
>openssl x509 -outform der -in c:\...\certificate.pem.crt -out c:\...\client.der
>openssl rsa -outform der -in c:\...\private.pem.key -out c:\...\private.der
```

[1] https://www.researchgate.net/figure/TI-CC3200-Launchpad-board_fig1_312559421

proceed to convert each of the keys and certificates and keys with the appropriate extension and file names.

Using UniFlash to flash key and certificates to CC3200

We then proceed to put the certificates and keys into our Launchpad via UniFlash. This was a fairly straight forward process however we did make a file-naming error at this part, which hindered the process and the files failed to transfer properly.



LAB 4 | CCS UNIFLASH

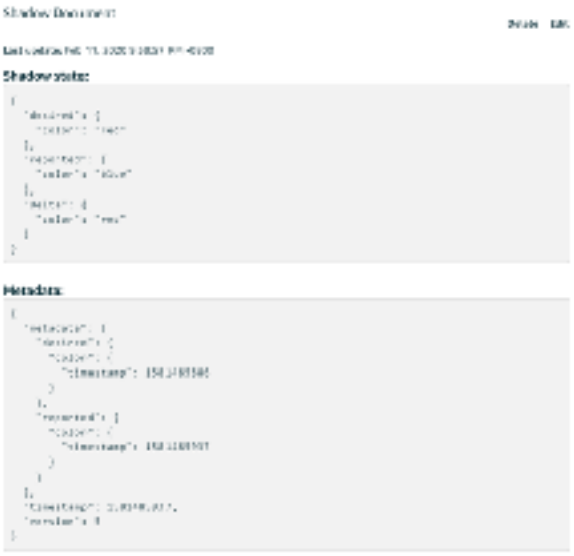
After some inquiring and research, we realize that there are certain folders on the LaunchPad that we indeed send our files to. But the way we named our files was not going to allow for the files to reach their proper destination under the User Files. After fixing the names, we were able to find and fix the bug that took us majority of the lab period to figure out.

Accessing AWS using the RESTful API

Amazon's RESTful API for AWS InT is easy to implement as we found through this portion. We were already familiar with the JSON format so we proceeded to our CCS to manipulate the given demo file to make it compatible with our project. To do this, we inserted the AWS endpoint as the host and made sure that the port was correct. We also changed the wifi so that it was under the 'eec172' wifi given to us by the lab. Unfortunately we were still

experiencing connectivity issues and we believed that this was occurring due to the fact that common.h was not recognized by our project manager. To fix this, we simply changed the path of common.h on the computer to match that of the project, and it was hence able to find the file.

After this fix, the program worked like a charm and we were able to receive the JSON-formatted status of the shadow from the cloud.



LAB 4 | INITIALIZING SHADOW STATE

LAB 4 | UPDATING SHADOW STATE

LAB 4 | SUCCESSFULLY SEND A GET

LAB 4 | SUCCESSFULLY SENDING A POST

Part II. Using SNS to send a text to your phone

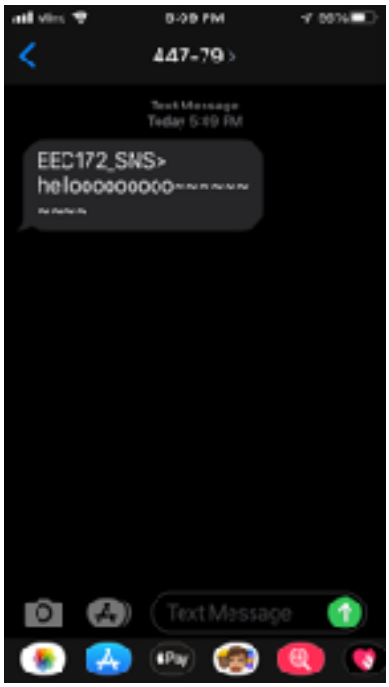
In this part, we do something with our thing by sending an SNS message from our phone when we hit SEND from our IR Remote. This is fairly easy with the documentation, as all we do at this point is set up a topic in the Simple Notification Service and register a text-capable cell phone.

Creating an SNS Topic

The SNS service is a module that allows us to push messages to subscribers. We 'Create a Topic' which then creates a subscription with the SMS protocol. The endpoint int his case is the phone number(s) we would like to send the texts to. After establishing this, we publish to the topic.

Creating an IoT Rule

We create an IoT Rule that triggers when we push updates to our device. To do this we go on Act > Create a Rule and make sure that this rule is enabled to allow for us to push an update to our device shadow through the CC3200, which is then connected to the IR remote from the last lab. After this, we were able to receive texts after typing them out with the IR Remote.



LAB 4 | SUCCESSFULLY SENDING AN SNS FROM IR REMOTE

SUMMARY

This lab successfully taught us the ins and outs of the AWS interface as well as helping us establish a connection between our board (and the IR remote) to the AWS IoT Core service back to our phones. This is a very handy tool that can definitely be built upon and manipulated for a wide arrange of uses. The power of the connection between AWS with the Internet of Things is great and will help us essentially accomplish just about anything in the world of embedded systems.