UNIVERSITY OF CALIFORNIA, DAVIS
Department of Electrical and Computer Engineering
EEC 172 Spring 2023

LAB 5 Verification

Team Member 1: Thomas Ke
Team Member 2: Bradley Manzo
Section Number/TA: A05 / Begum Kasap

PART1.

Date	TA Signature	Notes
5/25	Begum	http-post done. http-get done ✓

PART2.

Date	TA Signature	Notes
5/26	CJ	email message received using hard_coded msg. the email SQL is not cleaned for value only [one day late] System fully functional with 1 bl late. -R8

Implementation	**/70**
- Functionality	/55
- Robustness	/10
- Presentation	/5

Report	**/30**
- Thoroughness	/10
- Clarity	/10
- Presentation	/5
- Code Clarity/Commentary	/5

EEC 172 Lab Five

Bradley Manzo
Computer Engineering B.A.
916953788
brmanzo@ucdavis.edu

Thomas Ke
Computer Engineering B.A.
917513004
thke@ucdavis.edu

1. INTRODUCTION

This lab introduces the Amazon Web Service (AWS) as well as the RESTful API, which allows for our CC3200 device to connect to AWS and use a variety of services on the platform. This project is meant to teach us about a variety of AWS services which might be of use in allowing our CC3200 to carry out a number of services that might require the internet such as device shadows and email notifications.

2. GOALS AND TASKS

Completion of the first part of the lab requires the creation of an AWS account. We were then instructed to create a device "thing", or a representation of a device on the AWS as well as a corresponding device shadow to allow that representation to be interfaced with. After this, we then generated a set of certificates and keys and registered them with a device policy to the created device "thing". This would control certain device actions and limit access of the device only to trusted users.

Then, we used OpenSSL to convert the device keys and certificate to a format that could be used with the CC3200, flashed those items onto the CC3200, and created a program using the RESTful API to connect our CC3200 to the device shadow on AWS.

In the second part of the lab, we modified our previously created device "thing" to attach an SNS topic onto it and created a rule to have our device "thing" forward incoming messages to that SNS topic. We then subscribed to the created SNS topic using our email. After this, we adapted our IR receiver texting program from Lab 3 to, send the string wrapped within a JSON packet rather than sending the message to another board. This would then update our subscribed emails with the transmitted message.

3. CONCEPTS AND UNDERSTANDING

3.1 Part I

In the implementation of Part 1, the AWS portion was not particularly difficult as long as the instructions were properly followed, though there were some unclear portions which we had to clear up with peers or a TA. However, for the implementation of 1.F, we were required to understand the RESTful API and how to properly make use of our generated certificates in order to properly have our CC3200 interface with the device shadow.

3.2 Part II

In the implementation of Part 2, the AWS portion was similar in concept to the previous method, though we did also test that the email was properly sent to subscribed email accounts using our previous program from Part 1.

As for the new program for Part 2 where we modified our previous Lab 3 texting program to send the message data to the device shadow. We implemented this functionality by

initializing the connection with AWS using the connectToAccessPoint and tls_connect functions. The packet is then created when we call the http_post function and the string to transmit is encapsulated within a JSON header and tail to create a packet. This is then uploaded to the AWS where it is enclosed in an email and delivered to our email inbox. The conceptual understanding of our lab three code remains the same, but in summary: an interrupt handler is registered to trigger when our IR sensor is low. We then interpret every 32 consecutive signal delays into a character. The delay between these messages, determines which character is written to the text with the multitap scheme.

4. METHODS

4.1 Part I

In Part 1, we created an AWS account for use, and followed the instructions outlined in the lab manual to create a device "thing" and shadow along with the requisite device policy, certificates, and keys. We then converted the certificates and keys to .der, a CC3200 readable format, and flashed them onto the CC3200.

For the program which interfaced with the device "thing" on AWS, we followed the lab report by modifying the given example program according to the given instructions, such as changing the hard coded date and adding in the given RESTful API code.

4.2 Part II

In Part 2, we modified our existing device "thing" to further include an SNS topic and a rule to forward "thing" messages to that SNS topic, we then subscribed to the SNS topic using our emails.

After this, we modified our Lab 3 texting program using the RESTful API to post to our device "thing", which would then be forwarded onto the SNS topic. Rather than putting the captured string onto our UART connection between two CC3200 boards, we instead call the http_post function to encapsulate the string within a JSON packet. The JSON header and tail are preset strings, and the string to transmit is appended letter by letter to the header before being appended to the tail. Despite this minor change, the code is the same implementation as in Lab Three where we use an interrupt handler to detect changes in a GPIO pin corresponding to an incoming IR signal on the peripheral. We then construct a message as a 32-bit word and once we receive 32 consecutive logical delays, we decode which number button this corresponds to. We then decode these numbers into characters by waiting a certain delay between button presses. If it is less than a tenth of a second, the first letter printed on the button is printed, however if it is pressed twice or thrice within that delay, it prints the second and third respectively. The string is printed to the OLED through SPI as is wiped once we post it to the server.

5. Discussion of Challenges

In the AWS portions, we generally had no serious challenges. However, there were sections where the instructions in the lab manual were not clear enough and we consulted our peers and the TA regarding those instructions to ensure that we were on the correct path.

We also had an issue in regard to the flashed certificates and keys where they were not being read properly, leading us to have to reformat and reflash them later in the lab at the instruction of our TA.

However, regarding the Lab 5 programs, we had a great deal of trouble getting our string literal to properly append to the JSON header and tail. This took us hours to find a solution, despite having a working server and connection setup. The solution we finally found seems shoddy and we are curious to see what the proper method was.

6. Contribution Breakdown

Both of us worked together on Parts 1 and 2, Thomas generally handled the AWS portions whereas Bradley generally handled implementing the code with the RESTful API.

7. Conclusion

After the completion of Lab Five, we feel that we have gained a greater understanding of the Amazon Web Services in terms of how to utilize it to a basic extent and how useful it may be for not just web applications in the future but also for embedded systems applications.

We also learned how to create programs that allow physical devices to interface with and send data to their designated representations on AWS as well as how to automatically interact with that data using AWS services.