Lab 6 Report

Dan Tran
UC Davis
dxtran@ucdavis.edu

Timothy Nishiwaki
UC Davis
tanishiwaki@ucdavis.edu

1. ABSTRACT

In this paper, we describe the processes used, obstacles overcome, and results achieved for the final project (sixth lab) in EEC 172 - Embedded Systems (UC Davis Spring 2022).

2. INTRODUCTION

In this lab we used the IR receiver circuit created in lab 3 to implement multi-tap texting from a TV remote to an Adafruit OLED display. Using the remote as user input, we implemented 2 main functions with the help of AWS API and a dictionary API. The first function is a dictionary that would return definitions for a user requested word. The definitions could be seen on the OLED display and also sent to email upon the user's request. The other function was a Wordle game (with 5 guesses) of which the scores would be sent to the user's email.

3. BACKGROUND

This project had three main requirements that had to be met. The first requirement was to send or receive information from the internet using the CC3200. We satisfied this requirement by connecting to a dictionary API and AWS.

The second requirement was that the board had to be interfaced through a combination of peripherals. Our project uses the IR sensor to receive user input and displays the information on an OLED display.

Finally, the last requirement was that the system had to operate without being connected to a host computer. We made sure to fulfill this requirement by flashing the code on to our board for testing throughout the process.

4. GOALS

Our goal for this project was to create an interactive system that connects to a web-service via the internet.

We attempted to accomplish this by using a Dictionary API and AWS to create a dictionary and wordle game.

The dictionary would allow the user to input a word and get a definition.

The Wordle game would allow the user to play a game of Wordle.

5. METHODS

This lab had two main parts. However, in both parts, we had to set up an AWS account and create a device thing. Then we created a shadow attached to the thing. From this shadow we were given a public key, a private key, and a root CA certificate.

The next step was to update the security of our things policy to allow get and update actions. This would allow us to use GET and POST commands later in the lab.

In order to use the certificates and keys on the CC3200, we converted the files into the .der format using openssl. Then, using uniflash, we flashed the converted files onto the board.

Afterwards, we tested to see if we could connect to AWS by running some example code. In the code, we had set the wifi connection information to our personal wifi network. Then we set the date and time to be fairly accurate for the certificate to work. finally we set the hostname to the endpoint provided by AWS.

The next step was to do something similar to connect to our dictionary API. Doing so required the use of a different root CA certificate and a different endpoint. It also required we use a different cipher supported by the dictionary API.

Finally, we modified the connect and post functions provided to have dynamic input. This way we could tell the functions which API to connect to (dictionary or API) and which command to run (GET or POST).

After finishing the components that would connect to our APIs, we started to develop the functions themselves.

Firstly, we modified the code from lab 3 to give different outputs (and therefore operate different functions) depending on what mode the board was set to. The two different modes we had to implement were Dictionary and Wordle mode.

For Dictionary mode, we first had the user enter in a word that would show on the OLED (like in lab 3). However, instead of sending the word to another board, this time the word would be sent to a Dictionary API that would return definitions for the word.

The definition of the word was then stored onto the board (max 32 definitions of about 512 characters each due to board memory restrictions). The user may then use buttons to scroll through the different definitions or send the definitions to their email (not limited to 32 definitions of about 512 words due to sending directly from the original receive buffer and not the ones stored on the board).

For Wordle mode, we had the words displayed in the middle of the screen and limited to 5 characters (like wordle). We also had a "keyboard" UI shown at the bottom of the screen during gameplay. When a word was submitted, it would use the Dictionary API to check for a definition. If there were no definitions, their word was not 5 letters long, or the connection failed, we would alert the user that their input was not a valid word and they could change their guess.

Similar to how Wordle worked, the letters would have their backgrounds changed depending on the correct answer. No change would indicate the letter was not in the word. Yellow background would indicate the letter was in the word but in the wrong spot. And Green would indicate the letter was in the same spot as the correct answer. Our keyboard would also update in a similar fashion (white font changed to blue font instead of no change compared to earlier, otherwise yellow and green logic are the same).

After the user won the game by guessing the right word or lost the game by failing to guess in 5 words, their score and statistics would be sent to their email.

One could switch between the two modes freely by pressing a button on the remote (Would reset state of that mode though). There would also be a "loading" screen displayed while the board

is either communicating with the APIs or updating the OLED display.

An overview of how we achieved the overall logic of the system can be seen in Figure 1. And controls for the system can be seen in Figure 2.

6. DISCUSSION

The main problem that we faced during the planning stages of our design was finding a working API . We had a lot of different ideas for games that could be implemented on the board, but finding an API that had matching cipher suites and working rootCA files was difficult for our application.

Once we found a working dictionary API we had trouble parsing through the JSON file to only receive the information we wanted. In particular, it was difficult to match each definition with its part of speech. We had the logic figured out, but we used an int to store the position of the "part of speech" label in the JSON file. This caused problems because the values were too large so we needed to store the position as a long int.

The next major problem we had to solve was how to generate random words for Wordle. Initially, we wanted to generate random words using an API, however all the dictionary API with that function were not compatible with the board. Instead we used A word bank of common words saved as an array of strings. Then we used the rand() function in C to generate a random word. However, we had to also figure out a way to set the random seed so that we would get a different sequence of random words every time the program was flashed on the board. To do this we set the random seed using the time passed before entering Wordle mode using the systick timer.

Another problem was figuring out if the board was still able to process inputs or not. We were sometimes unable to tell if the board wasn't taking input correctly or if it wasn't processing the inputs correctly. To fix this, we simply added a "Loading…" message while the board was processing something that prevented it from receiving input (i.e. API connections or updating OLED).

Another problem was figuring out how to connect to two different API's for the dictionary function and email notifications. To do this, whenever we were sending a request, we had to reconstruct the string for the post/get request and host name. And we had to change the cipher suite in the tls_connect() function.

7. CONCLUSION

At the end of this lab we were able to use a TV remote to get definitions of words both on the OLED display and our email. We were also able to play a Wordle game (but with 5 guesses) and receive our statistics and score in our email.

One improvement that we could make would be to hold the connection to the Dictionary API in wordle mode instead of connecting every time we enter a word. This would increase the speed of checking words and eliminate the problem of failing to connect to the server multiple times before being able to check a word.

8. Contributions

Most of the coding and debugging was done together and we split the work on the lab report appropriately.

9. ACKNOWLEDGMENTS

Our thanks to our section TA, Asmita Asmita and our classmates who helped us during the lab.

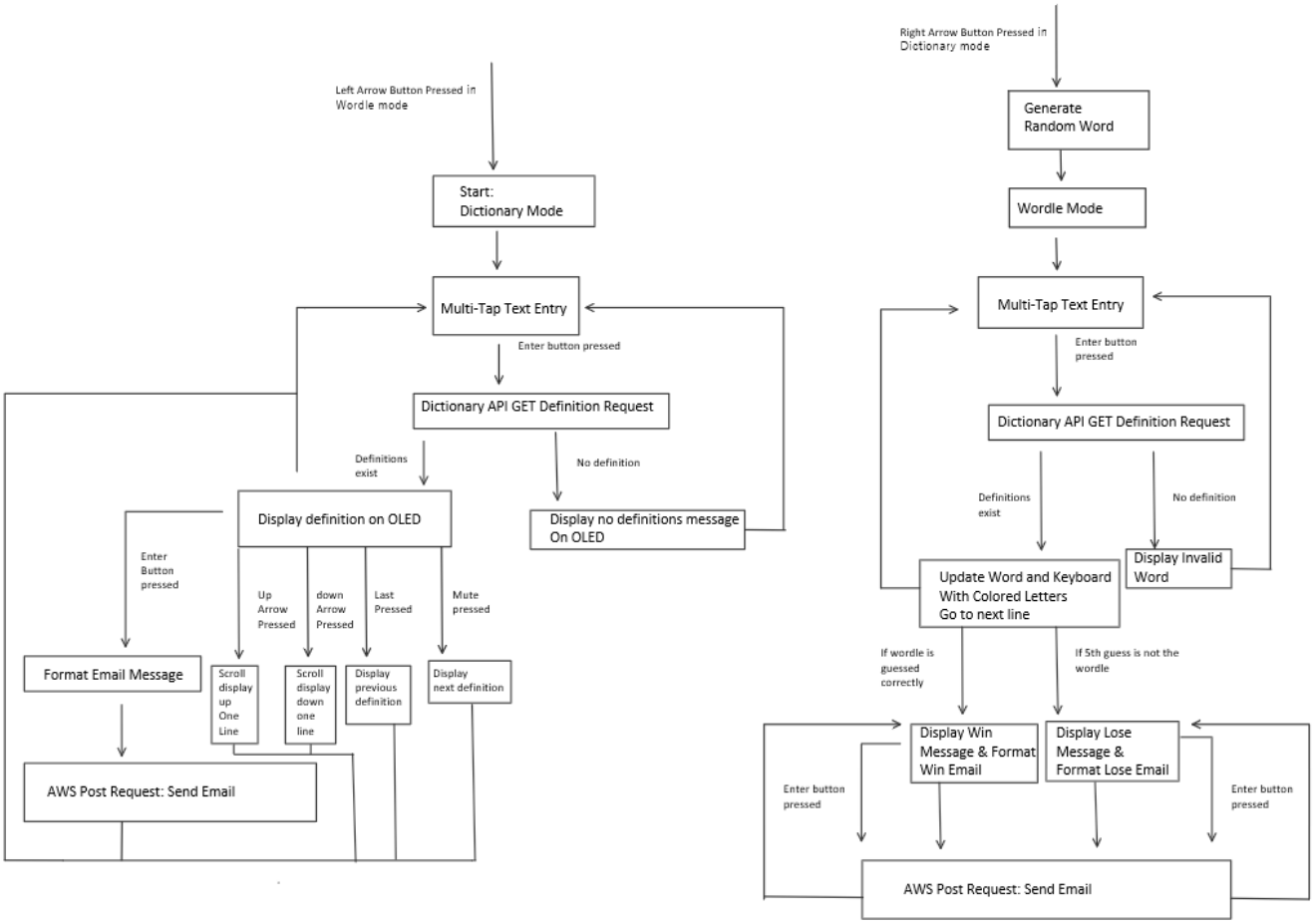
Left Arrow Button Pressed in Wordle mode
Start: Dictionary Mode
Multi-Tap Text Entry
Enter button pressed
Dictionary API GET Definition Request
Definitions exist
No definition
Display definition on OLED
Display no definitions message On OLED
Enter Button pressed
Up Arrow Pressed
down Arrow Pressed
Last Pressed
Mute pressed
Format Email Message
Scroll display up One Line
Scroll display down one line
Display previous definition
Display next definition
AWS Post Request: Send Email
Right Arrow Button Pressed in Dictionary mode
Generate Random Word
Wordle Mode
Multi-Tap Text Entry
Enter button pressed
Dictionary API GET Definition Request
Definitions exist
No definition
Display Invalid Word
Update Word and Keyboard With Colored Letters Go to next line
If wordle is guessed correctly
If 5th guess is not the wordle
Display Win Message & Format Win Email
Display Lose Message & Format Lose Email
Enter button pressed
Enter button pressed
AWS Post Request: Send Email

Figure 1

	Dictionary	Wordle
UP	Scroll up	N/A
Down	Scroll Down	N/A
Left	N/A	Dictionary Mode
Right	Wordle Mode	N/A
OK	Send Email	Resend Email
Mute	Last definition	N/A
Last	Next definition	N/A
Enter	Request definition	Check Guess
1	Backspace	Backspace
(multitap) 2	abc	abc
(multitap) 3	def	def
(multitap) 4	ghi	ghi
(multitap) 5	jkl	jkl
(multitap) 6	mno	mno
(multitap) 7	pqrs	pqrs
(multitap) 8	tuv	tuv
(multitap) 9	wxyz	wxyz
0	Space	Space

Figure 2

UNIVERSITY OF CALIFORNIA. DAVIS
Department of Electrical and Computer Engineering
EEC 172 Spring 2022

LAB 6 Verification

Team Member 1: Dan Tran

Team Member 2: Timothy Nishikawa

Section Number/TA: A01 Manisha Sivaswamy

Demonstrate your creative project:

Date	TA Signature	Notes (here explain your project briefly)
06/02		⇒ Implemented a dictionary and a game - wordle. Used two API - dictionary, AWS send email notification of the game results. ⇒ Used the random generator to generate the words for wordle game ⇒ Use the buttons on the remote to control the dictionary & wordle modes ⇒ dictionary → buttons to texting & word selection OK button to send the email notification

Amazing job! Outstanding implementation