Lab6: Open-Ended IoT Project

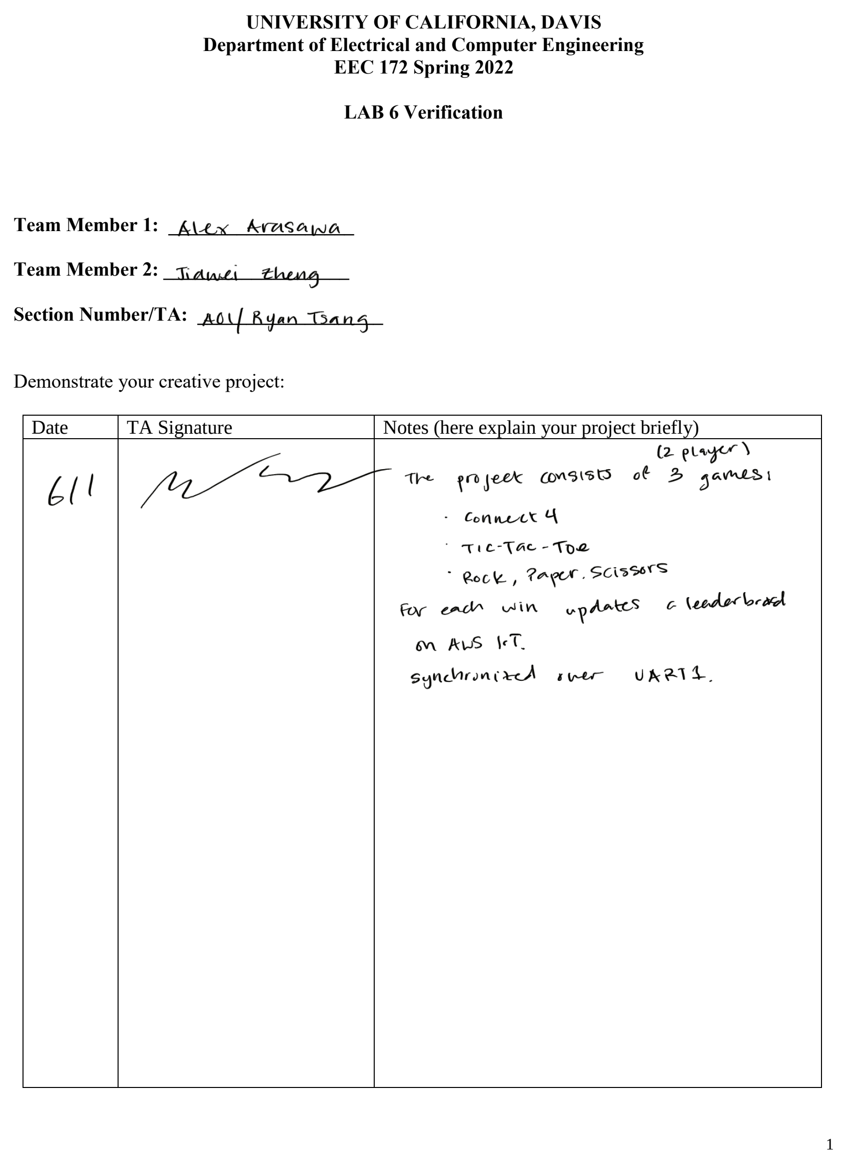
Alexander Arasawa, Jiawei Zheng

June 11, 2022

EEC 172

Professor Soheil Ghiasihafezi

1 Check off from TA:



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

LAB 6 Verification

Team Member 1: Alex Arasawa

Team Member 2: Jiawei Zheng

Section Number/TA: A01/ Ryan Tsang

Demonstrate your creative project:

Date	TA Signature	Notes (here explain your project briefly)
6/1	[illegible]	The project consists of 3 (2 player) games: · Connect 4 · Tic-Tac-Toe · Rock, Paper, Scissors For each win updates a leaderbrod on AWS IoT. synchronized over UART1.

1

2 Introduction

The main goal of this lab is the use what we learned from prior labs to produce a project of our own. We must use the CC3200 LaunchPad, interface with a device to the LaunchPad, connect to a web-service like AWS, and be interactive. We decided to implement two player games with two CC3200 boards(connect 4, tic tac toe, and rock papers scissors). AWS IOT is for keeping track of wins of the boards and UART1 is used to synchronize the games between the two boards. IR receivers and remotes were used for user inputs to the games.

3 Background

CCS software is an IDE for configuring TI's microcontrollers and embedded processors. The software contains many tools to help with creating and debugging applications: CSS includes an optimized C/C++ compiler, a code editor, a project build environment, a debugger, and many other features.

CCS Uniflash is a standalone tool used to program flash memory of TI's microcontrollers and program flash memory of Sitara processors.

TI Pin Mux Tool is a software used for configuring pins, peripheral signals, and other components of a system.

AWS IoT Core is an Amazon Web Service (AWS) that allows you to connect IoT devices and route messages to other AWS services.

References:

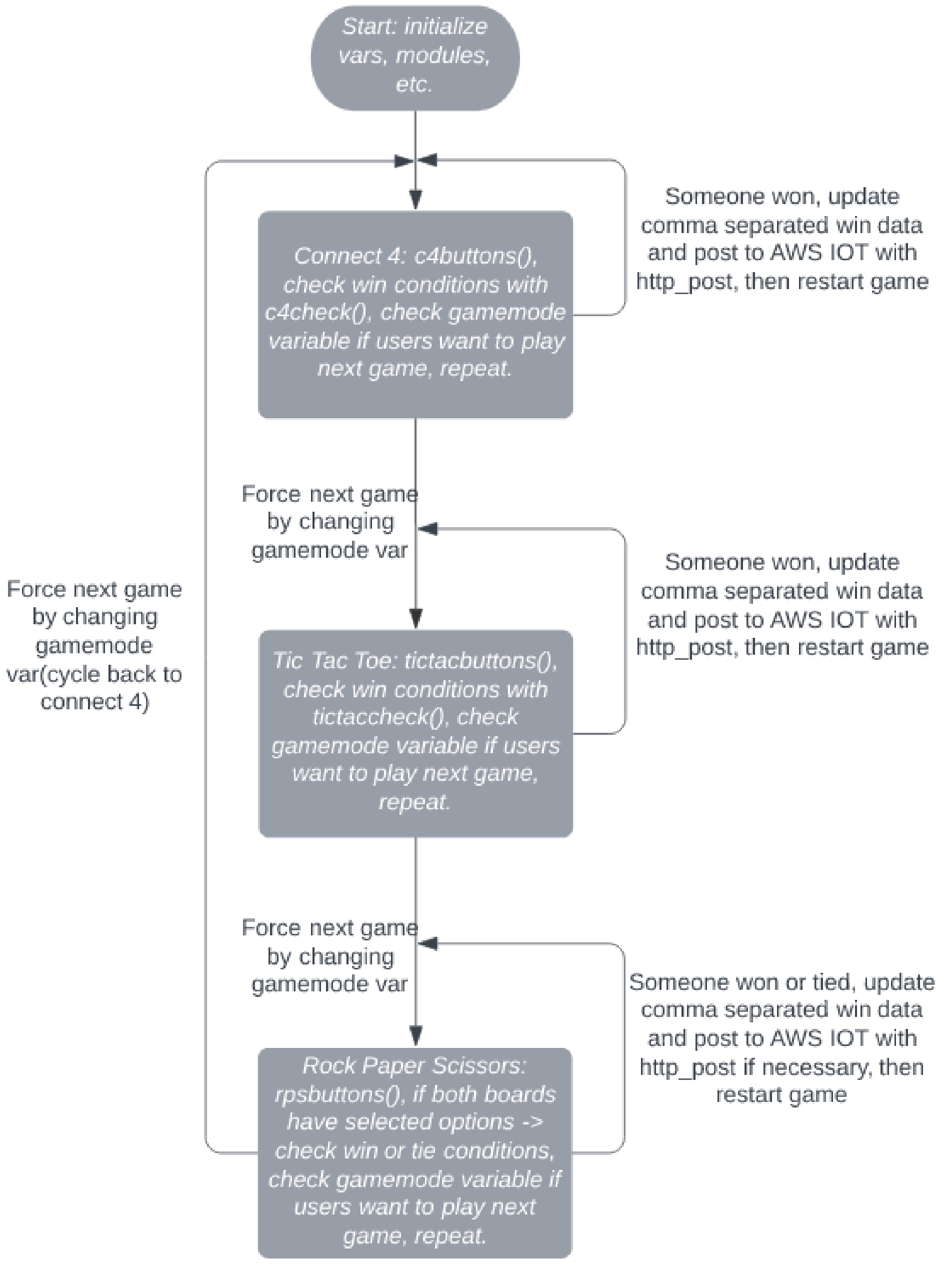
`https://www.ti.com/tool/CCSTUDIO`

`https://www.ti.com/tool/UNIFLASH`

`https://www.ti.com/tool/SYSCONFIG`

`https://aws.amazon.com/iot-core/?nc=sn&loc=2&dn=3`

4 System Block Diagram


Start: initialize vars, modules, etc.
Connect 4: c4buttons(), check win conditions with c4check(), check gamemode variable if users want to play next game, repeat.
Someone won, update comma separated win data and post to AWS IOT with http_post, then restart game
Force next game by changing gamemode var
Tic Tac Toe: tictacbuttons(), check win conditions with tictaccheck(), check gamemode variable if users want to play next game, repeat.
Someone won, update comma separated win data and post to AWS IOT with http_post, then restart game
Force next game by changing gamemode var(cycle back to connect 4)
Force next game by changing gamemode var
Rock Paper Scissors: rpsbuttons(), if both boards have selected options -> check win or tie conditions, check gamemode variable if users want to play next game, repeat.
Someone won or tied, update comma separated win data and post to AWS IOT with http_post if necessary, then restart game

5 Methods

5.1 Part I. Using What We Learned

We used the code from Lab 3 for reading in IR transmission from the IR receiver. Additionally, we used the knowledge we gained on OLED functions from prior experiences in Lab 2 to program graphics for the three games we decided on. Further, we copied over and flashed the same certificates from lab5 so the CC3200 can gain access to internet connection with AWS IOT.

5.2 Part II. Programming the Games

For connect 4, we used a 6 by 7 array of chars to represent the connect 4 board. Then we added buttons the users can press on the remote to drop pieces into the array. Also, we implemented a c4check(char) function to check for 4 of that char in a row so we can detect win conditions for red board with 'R' pieces and yellow board with 'Y'. When we detect a win, we would update local wins variables and then post updated comma separated win data to AWS IOT with HTTP_POST.

Tic tac toe is very similar to connect 4 so most of the code from connect 4 was reused to implement tic tac toe. The only difference was that the array of chars was 3 by 3 and the users can put 'X' or 'O' in any entry of tic tac toe struct unlike connect 4, which required that the previous lower entries in the columns of connect 4 struct must be filled first before you can drop pieces to your desired position in that same column(which is how connect 4 game works in real life).

For rock paper scissors, we used a array of 2 chars to represent the game. Red board can only modify one element in the 2 chars array with buttons and the yellow board can only modify the other element in the 2 chars array with buttons. When both boards has chosen either rock, paper, or scissor with remote buttons, we check for who won or if there is a tie. Then, same as connect 4 and tic tac toe, we would update local wins variables and then post updated comma separated win data to AWS IOT with HTTP_POST.

5.3 Part III. Updating HTTP_POST and HTTP_GET

We had to update http_post and http_get so that they can post or get comma separated win data to or from AWS IOT things device shadow json. http_post was modified in that we had to add how many digits were in both win variables to 'datalength' and we had to copy red wins, ',', and then yellow wins to pcBufHeaders buffer before we send it. http_get was modified in that, after we recieved the json through acRecvbuff, we index into acRecvbuff to extract "red_wins, yellow_wins".

5.4 Part IV. UART1 Interrupts Synchronization

First, we enable UART1 interrupts for both boards and connect the wires. Then we updated button press code so that when a button is pressed, UARTCharPut(char)(with chars mapped to button presses) would also be called to tell the other board that a button has been pressed for that board. Then the other board would know to update their local structs and OLED with the new information that was received through UART1 or UARTMessageInHandler().

6 Discussion

One difficulty of the lab was figuring out how to synchronize the games between the two CC3200 boards. We eventually figured out that the boards can be synchronized through the structs for the games with the help from UART1.

Another difficulty of the lab was choosing x and y positions on OLED for connect 4 and tic tac toe entries. Basically, a lot of guessing and testing was done to get decent looking connect 4 and tic tac toe tables or boards on OLED.

7 Improvements

A possible improvement could be adding more games. We actually tried to add a fourth game, battleship, but we ran out of time.

Another improvement could be separate leaderboards for different games because right now, there is only one total red win variable and total yellow win variable for all three games, which does not tell us which games were played to achieve those wins.

8 Conclusion

The main goal for this lab was to demonstrate an understanding of the material that we learned in lab. By programming the CC3200 LaunchPad, interfacing with components like the IR receiver and OLED, and using REST commands to POST/GET to a web-service online. We accomplished all three requirements towards this goal.

9 Comments

We decided to only comment the red board code because the yellow board code is basically the same but reciprocal or flipped. Also because the red board is the main communicator that updates comma separated win data to AWS IOT(the yellow board does not call http_post at all).

10 Contribution

We did all parts of the lab together so 50/50 contribution.