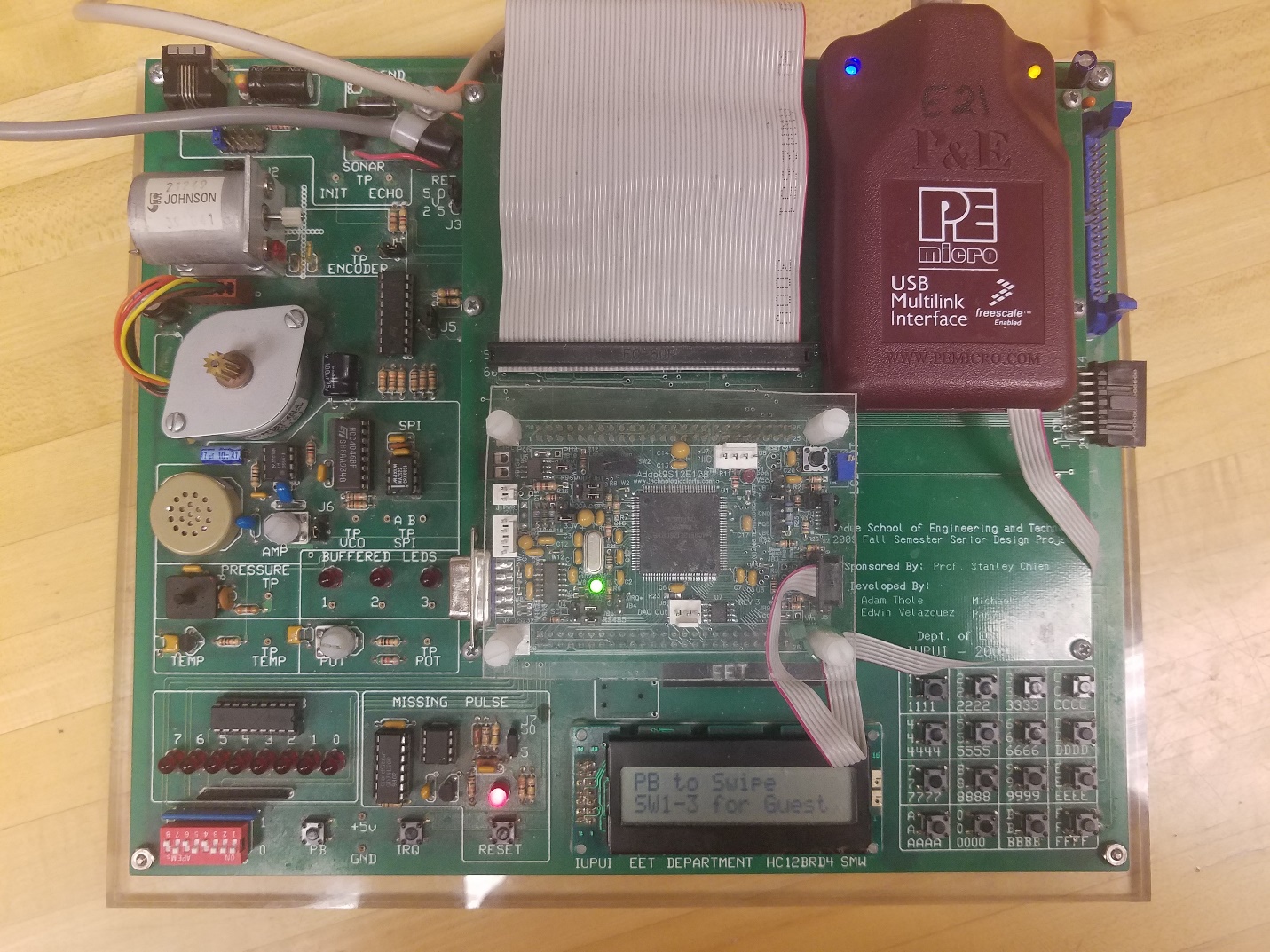
**Blackjack**

**Kyle Cooper**

**Spring 2017**

****

**Table of Contents**

1. Figures and tables………………………………………………Page 3
2. Introduction…………………………………………………….Page 5
3. Design…………………………………………………………..Page 5
4. What works what Doesn’t……………………………………...Page 11
5. Conclusion……………………………………………………...Page 11
6. Appendix……………………………………………………….Page 11
7. References……………………………………………………...Page 12

**Figures and tables:**

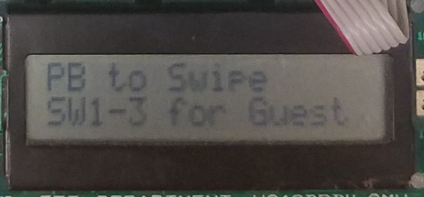


Figure 1 – LCD



Figure 2 – LEDS



Figure 3 – Switches

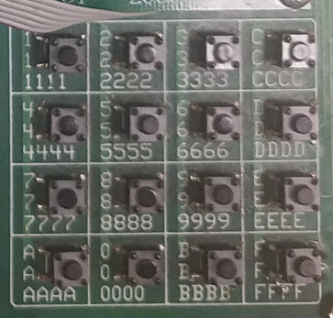


Figure 4 – Hex Keypad



Figure 5 - Potentiometer



Figure 6 – Speaker

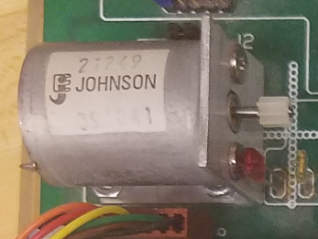


Figure 7 – DC Motor

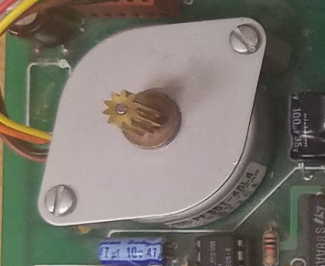


Figure 8 – Stepper Motor



Figure 9 – Push Button



Figure 10 – IRQ

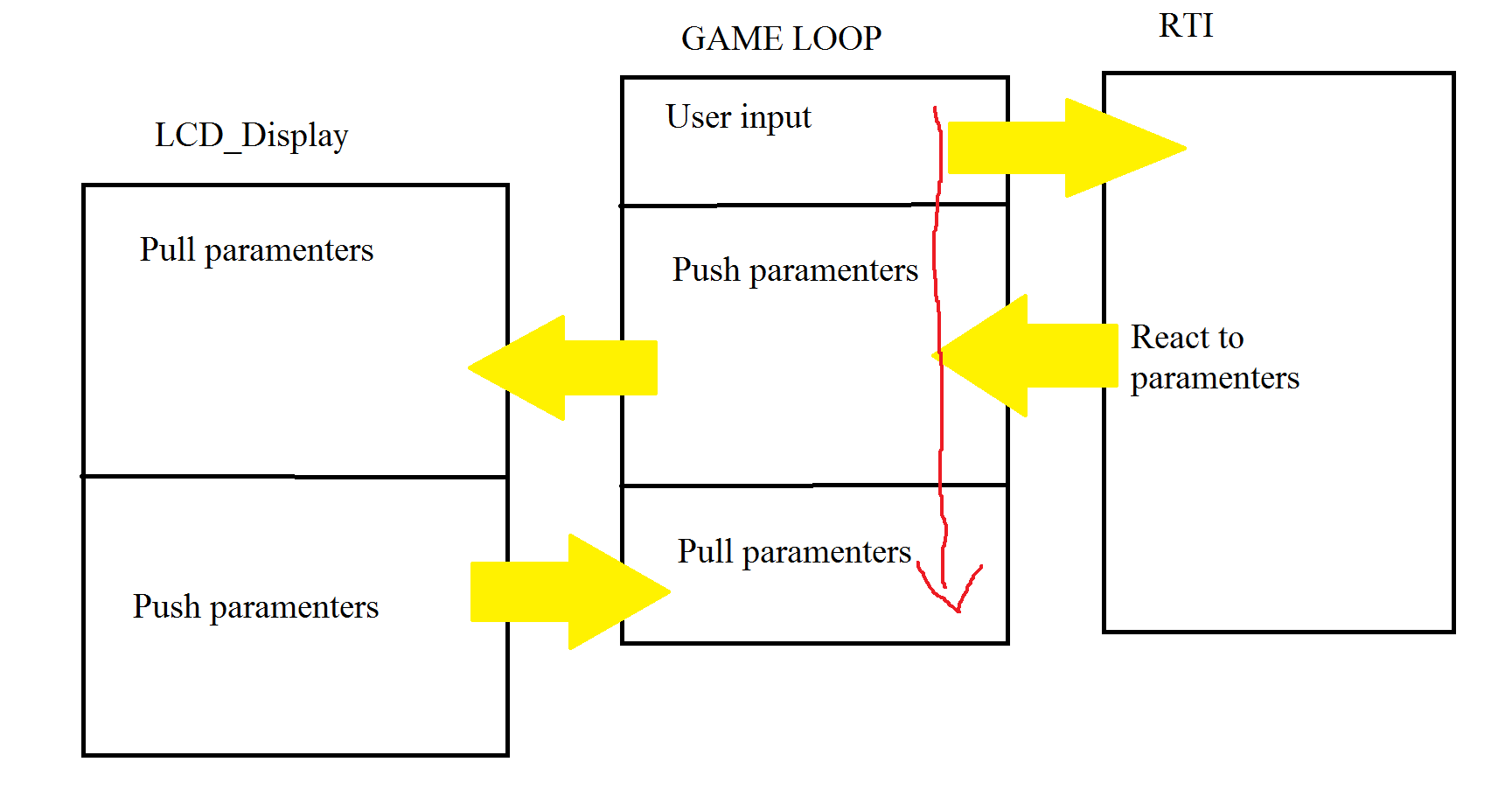


Figure 11 – General Flow Chart

**Introduction:**

The purpose of this project was to build a working game of blackjack on the board provided to us. The program needed to be easy to use for users and would require minimal knowledge to use. Starting out I didn’t think this project would take as much time as it did. Working in assembly is very simplistic, but can very tedious as I’ve learned.

**Design:**

LCD:

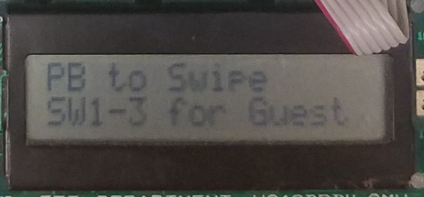


Figure 1

The LCD is what provides most of the information the user needs to know. It is what the game is displayed on and provides the user with an interface to interact with the game.

LEDS:



Figure 2

The LEDS are mostly for aesthetics, and light up with different patterns when the user loads into the game, wins/loses games, and cashes out.

SWITCHES:



Figure 3

The switches are mainly used to add money to user’s accounts and betting.

HEX KEYPAD:

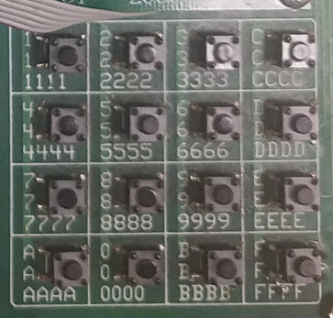


Figure 4

The hex keypad is used when moving forward and backwards in the menus, as well as hitting and staying while playing the game.

POTENTIOMETER:



Figure 5

The potentiometer is used solely inside the admin menu as a way to scroll through menu options.

SPEAKER:



Figure 6

The speaker works in unison with the LEDS to provide feedback to the user when they load into the game, win/lose games, and cash out. Just like the LEDS it is mostly for aesthetics.

DCMOTOR:

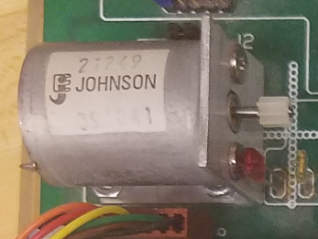


Figure 7

The DC Motor is used when the user cashes out, and is used to signify that a voucher is being

printed for the user.

STEPPER MOTOR:

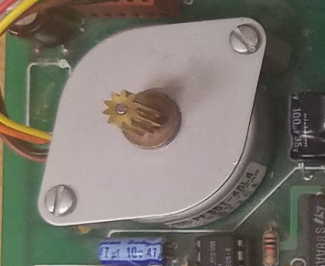


Figure 8

The Stepper Motor is used when adding money to the users account, and also in unison with the DC Motor when printing out vouchers for the user.

PUSH BUTTON:



Figure 9

The Push Button is used to load user profiles, after pressing it, users will be asked to enter a pin that identifies them and loads up their profile.

IRQ BUTTON:



Figure 10

The IRQ is used to enter into the admin menu for my project. If users fail to enter their pin before 15 seconds has elapsed the game will lock itself and alarms will continue to indefinitely.

SOFTWARE IMPLEMENTATION:

While this program has multiple parts to it the three main parts are the Main Game loop (located in the main), the RTI(located in the main), and the LCD\_Display. These three handle almost all interactions in the game, and constantly pass parameters to each other. There are a few other parts, but those other parts only deal with singular issues.

Main:

The main initializes all variables and constants that will be used inside my project. It also contains both my IRQ and RTI. I decided to include my RTI inside my main, because the main Game loop that runs the game is also inside my main. This allows the RTI and Game loop to pass parameters to each other very easily.

RTI:

The RTI handles almost all button presses and then will decide what needs to be done when a button is pressed. After a decision is made variables are updated and parameters are passed to the LCD\_Display through the Game loop that allows the LCD to update accordingly. The main parameter being passed is the d\_flag, which is what controls which “panel” of the game a user will see.

LCD\_Display:

The LCD\_Display takes parameters from the Game loop and displays the proper “panel” to the user. The LCD\_Display also controls most of the actual game mechanics.

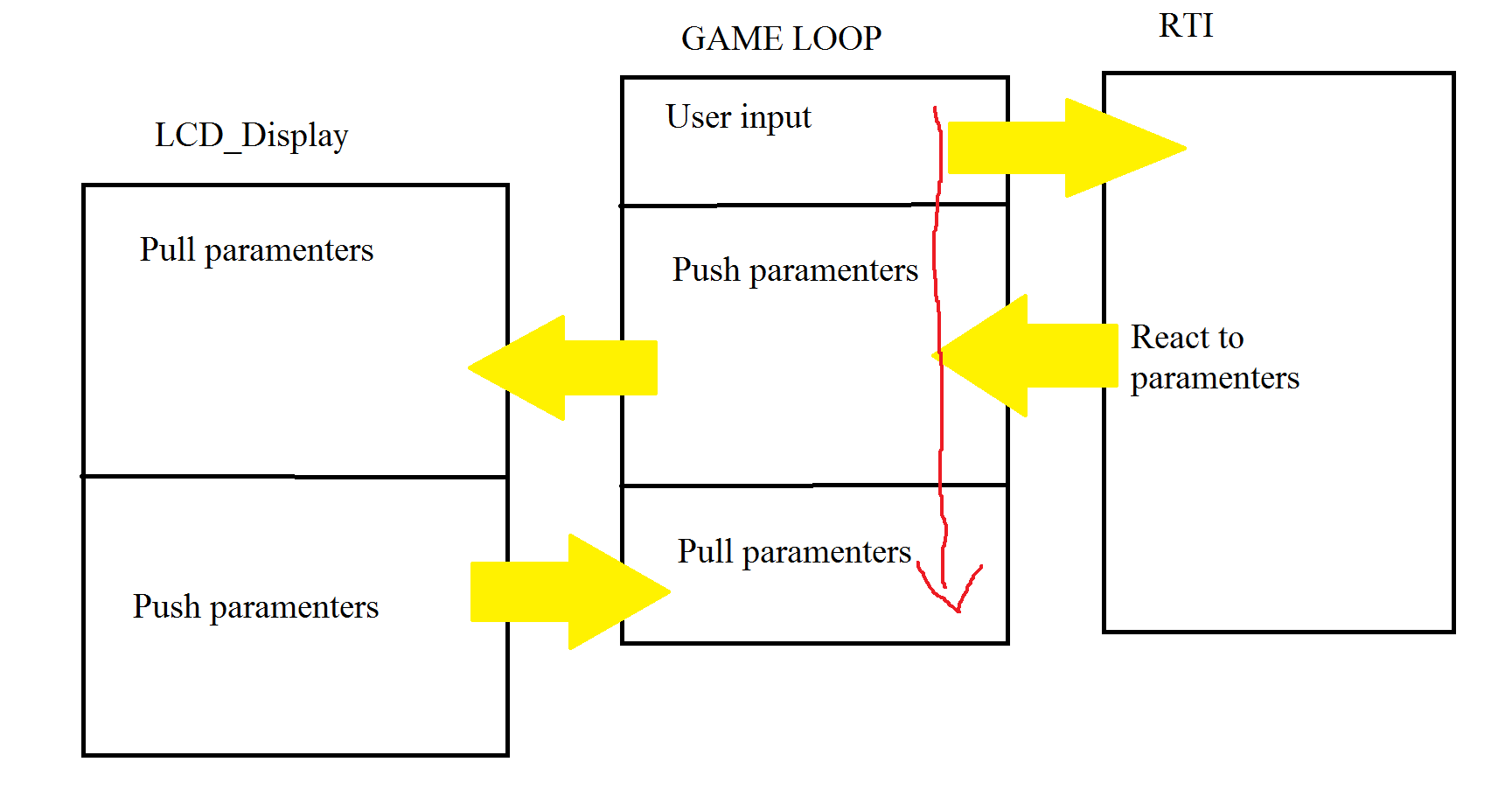


Figure 11 - Above is a simple flow chart of how all three work together

Other Modules:

DC\_MOTOR:

Most of the timings for the DC Motor happen inside the RTI, the DC\_MOTOR module is only used to turn off and on the dc motor, a 1 byte parameter is passed to it, a 0 = off, a 1 = on.

FIND\_CARD\_VAL:

FIND\_CARD\_VAL is used to assign cards there value. Inside the LCD\_Display counters are divided by 52 and the remainder is what will become the card. This remainder value is passed to FIND\_CARD\_VAL where the remainder value is used as an index value which moves through an array. Where in the array it stops at will be the value of the card, this value is then passed back to the LCD\_Display.

SELECT\_CARD:

SELECT\_CARD takes a parameter from the LCD\_Display, this parameter is the same as the one passed to FIND\_CARD\_VAL. The same as in FIND\_CARD\_VAL the parameter is used as an index to move through an array, only this time they are characters which will be the cards suit and value. SELECT\_CARD will then pass back the address location of the card to LCD\_Display, so that the card can be displayed.

HEX\_KEYPAD:

The hex\_keypad is exactly the same as the one used in our labs, only this time it passes its results back to the main, where the result is stored to a hex\_value to be used by the RTI in determining user key pad presses. If a user doesn’t press any buttons on the key pad, the hex value $20 is returned instead. This lets the RTI know that nothing has been pressed.

PIN\_CHECK:

This module is used to check user pins, it has multiple arrays which store user pins. A parameter is passed to it from the RTI, when a key press is detected and the program is also waiting for a pin. This value is then checked against user pins to see if there is a match. For each key press another check is done, until four presses have happened. At which point the pin check either passes for fails. If it fails PIN\_CHECK will send back a 0 to the RTI to let it know that there was a fail. If the pin check passes, PIN\_CHECK will return a value that is used to identify which user pin was entered to the RTI, (aka 1=kyle).

PLAY\_SOUND:

This is used to run the sound for the game as well as the LEDS, having them both be run out of the same module allows them to easily be synced together. A counter is used to make sure each sound and LED light pauses for a little bit before going on to the next one. Sounds are picked inside the RTI which passes a value to PLAY\_SOUND, this value lets PLAY\_SOUND know which sound/led pattern it should currently be displaying. Sounds and LEDS patterns are looped until they are stopped inside the RTI.

STEP\_MOTOR:

Just like the DC motor the Stepper motors timings are mostly controlled inside the RTI. This module instead takes a parameter from the RTI which lets the stepper motor know whether it should be run clockwise or counter clockwise.

SWITCHES\_HIGH:

This module is used to determine if a switch has been switched on or not, if nothing has been switched on it will return 0 to the RTI, if a switch has been turned on though it will return a corresponding value based on which switch has be flipped from high to low (aka bit 0 = 1).

CHANGES MADE:

No major design changes were made from start to finish, to save time and prevent major problems from arising I did a rough sketch up of what I wanted my program to look like before I started writing it.

ADDITIONS:

No major additions were added to the program besides the basic requirements. This was mostly due to the fact that working by myself caused the project to take a large amount of time, any addition would have greatly extended the project time and would not have been worth it over all.

**Division of labor:**

There was no division of labor, since I worked on this project solo, but I did have to prioritize my time. Since I was working solo a project is less modular than most other projects that had multiple people working on them. This cut down on the amount of time needed to implement different parts of the program.

**What works what doesn’t?**

For the most part everything works, besides a few things. For betting I use the switches instead of the hex keypad, which is an oversight on my part. Also for entering the admin menu, I use the IRQ instead of using switch 7, which is also an oversight on my part. The admin menu is the most incomplete part of the project. While you can scroll there options, none of the options currently do anything. Timestamps do not work, and you cannot change the admin pin or the time.

**Conclusion:**

Overall I am very happy with how my project has turned out even if there are a few things a need to fix. It has been a great learning opportunity and has greatly improved my ability to program in assembly. I will say in hindsight going solo (while fun) was probably was not the best idea, the amount of work required to finish this project cannot be understated, and having someone to split up the project with would have made things a lot easier on myself. Even if I had gotten a “bad” partner the worst case scenario would be I just work on the project pretty much solo like I have.

**Appendix:**

USER MANUAL:

After starting the program, the user will see a welcome screen, followed by the default screen. Users should either press the push button to enter a pin to load up a profile, or just use switch 0-2 to play as a guest. Users can also enter the admin menu from the default screen. While adding money just switch a one of the switches (0-2) from low to high 0=$5, 1=$10, 2=$20. After this users can press A on the key pad to move on to placing bets. In the add money screen users can also “cash out” by pressing C on the key pad. Betting works the same as adding money, users just use the switches. After the user places their bets(which cannot be higher than the money in their account), users have to press F on the key pad to start the game. While playing the game users can press 0 on the keypad to “stay”, and any other key on the keypad will “hit”. After winning/losing users will be returned to the add money screen where they can either play again or cash out.

**References:**

All my code is included in a zipped file that should be included in my submission. Details on what was required for my project can be found on canvas under the files, project name (SP17 Final Project 21.docx). Other than that the only reference I used was the CPU12 Reference Manual which can be found on canvas.