Critical Design Review: Puzzle Me Chess

February 22, 2021

EN.525.743.8VL.SP21

Mr. Joseph J. Vitale

Contents

List of Figures

List of Tables

Listings

Project description

In chess there are many different ways to play the game. After each player moves three times there are more then 121 million different moves that can end the game. Many people need to practice chess in order to get better and there are many ways to do this. You can play over the internet Player Vs. Play or player Vs. computer which range from (400 - 3200) EIO or you can play puzzle's in chess. Puzzle's in chess allow you to study the board in any given state and make the next several moves. This project is dedicated to solving puzzles in the physical world.

The physical chess puzzle allows the user to practice looking at a physical board to solve each puzzle. This project is titled "Puzzle me Chess" and it will be equipped to have 3 different puzzle's. The chess board will allow the user to place piece's in the correct location that is displayed on the OLED screen. After each piece is on the board the chess puzzle will then display to the user which color he/she will play to solve the puzzle. After the user makes his first move the board will check to see if the move was correct. If not the user will be asked to reset the piece and try again until puzzle is complete.

Capabilities

Capabilites for the Puzzle me Chess project range from Indicators, User Direction, SD card Capability, and Location Detection. More detailed explanation can be found in table 1.

Capability	Description below
Indicator	LED light indicator to show user hint/show answer feature
User Direction	OLED to describe to the user where to put pieces
SD Card	Read a .csv file standardized format to quickly import puzzles
User Direction	User input switch to show user answer, indicated by LEDs
User Direction	User input Button to show user the hint
Location Detection	Check board spots are correct for a given puzzle

Table 1: Shows the Capability's for Puzzle me Chess

Assumptions

Table 2 shows the assumptions that were made for this project. The assumptions were made to reduce overall cost of the project and difficulty to lower risk.

Assumptions	Description below
User Implication	User puts correct pieces on each spot
Physical	Chess Board will be lit evenly with light
Text File	Each file added to the SDcard will be in a standardized format

Table 2: Shows the Assumptions that were made for Puzzle me Chess

Limitations	Description below
Location Detection	Not knowing which piece is on the correct spot
Auto Movement	Not being able to move the piece on the correct spot
Puzzle Selection	Not having 3+ different puzzles to choose from
Physical	Not being able to light each square along the perimeter
Physical Parts	Not having multiple colors to indicate wrong or right answer
Physical	Needing light to illuminate the chess board evenly

Table 3: Shows the Limitation's for Puzzle me Chess

Limitations

Limitations for the Puzzle me Chess project range from Location Detection, Auto Movement, Puzzle Selection, Physical and Physical parts. See table 3 for more details.

Functional description

The Puzzle me Chess project will feature a Microcontroller, one 21x21 Wooden Chess Board, 8 Multiplexers, 1 Switch, 1 Potentiometer, 1 Button, and A Display. Figure 1 shows the current circuit layout as of 02/16/2021, this circuit doesn't show the chess board. Explanation of the System Block Diagram to follow.

System Block Diagram

Figure 2 shows the sudo hand drawn block diagram. The project consist of One Microcontroller which will feature the Teensy 3.6, 8 Multiplexer's from Texas Instruments part # CD4051BE, 1 basic breadboard Switch, 1 breadboard Potentiometer, 1 basic breadbaord Button, 64 LED's, 64 Photoresister's, 1 Chess Board and a Display.

The Chess board will have 1 LED and 1 photoresister per block. The LED will be drilled and placed on the top left side of each block and will be used to show the user where to place each piece. The photoresister will be drilled in the middle of each block and will be used as a voltage drop to tell if there is a piece there or not. See Figure 3 for the circuit layout.

Multiplexer Circuit Diagram

Figure 3 shows the sudo circuit for Muxltiplexer A, the puzzle me chess project will consist of 8 Muxltiplexer one for each column A-H. Each of the Muxltiplexers will have 8 outgoing lines that will read Voltages levels from the rows # 1-8. Each row will have one resister which is set to be $2M\Omega$ and one Photoresister in series. Each Photoresister has two modes a light and a dark mode. Equation (1) shows Mux input voltage at 3.29VDC and equation (2) shows Mux input voltage at 2.2VDC.

Assume Photoresister Light; R = 500Ω

Assume Photoresister Dark; R = $1M\Omega$

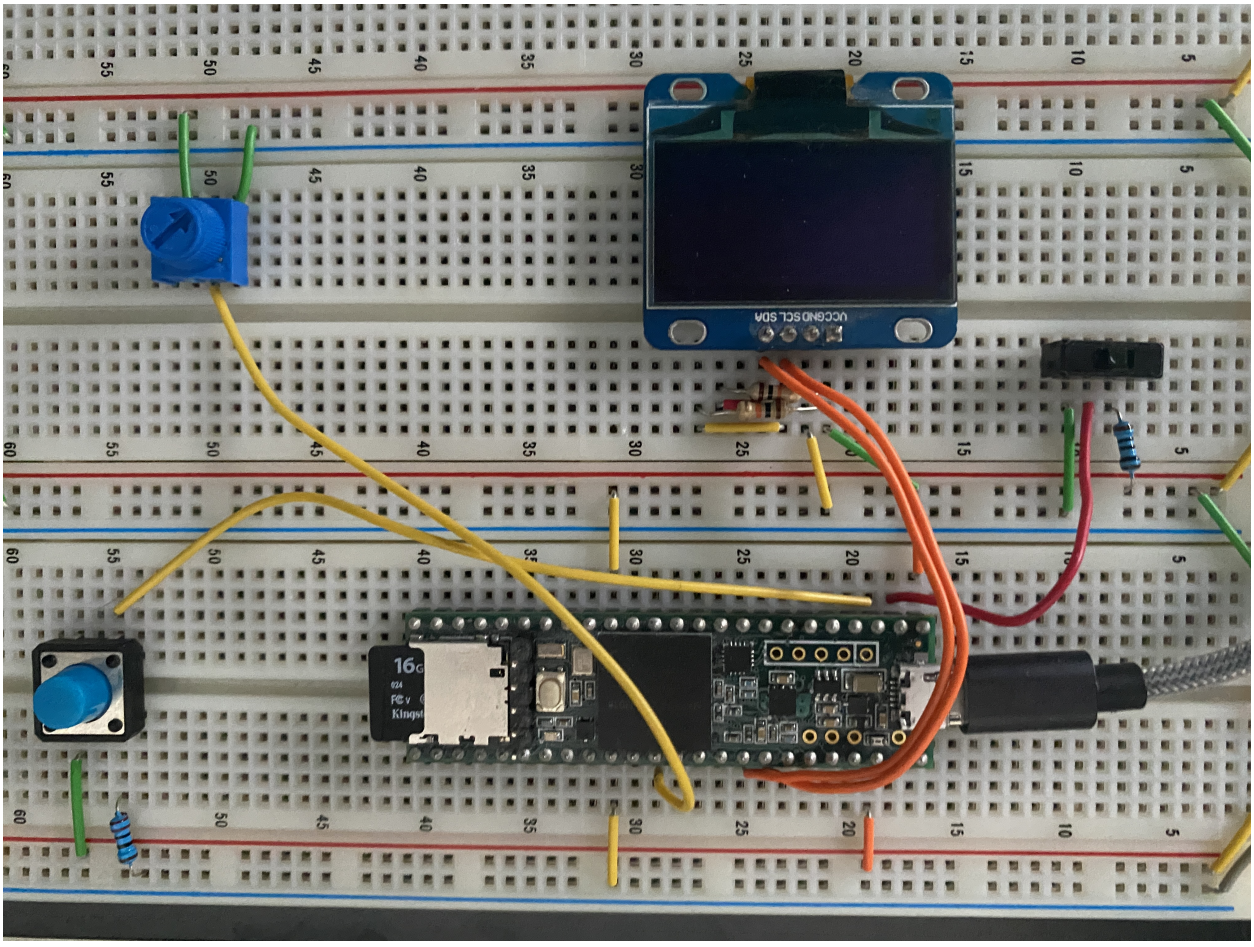

Figure 1: Circuit layout as of 02/16/2021

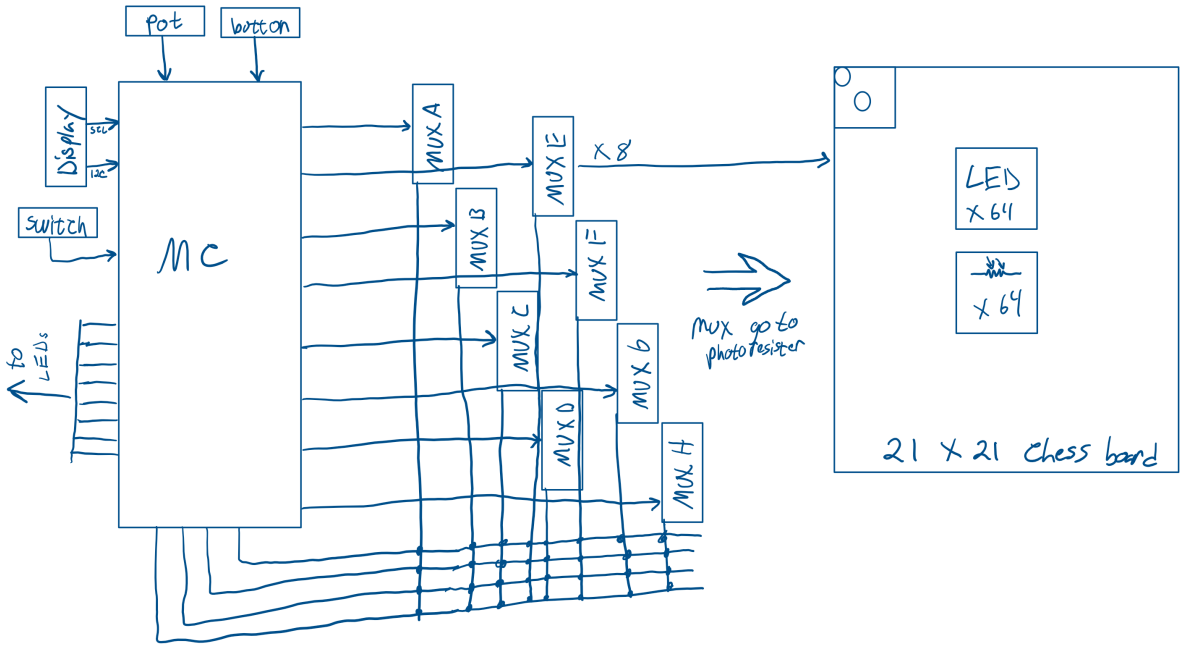

Figure 2: System Block Diagram

Light

V = IR, 3.3VDC = I*(2M + 500)Ω

I = 1.65μA, Vlight = 825μVDC, Mux input would see 3.299VDC if no piece on top of photoresister

(1)

Dark

V = IR, 3.3VDC = I*3.3MΩ

I = 1.1μA, Vdark = 1.1VDC, Mux input would see 2.2VD if piece was placed on block

(2)

Code Flow Diagram

Figure 4 shows the code flow diagram. The code starts with the display showing the user to pick a puzzle which the user will use the Potentiometer to select puzzles 1-3. After the Puzzle has been selected the display will show which puzzle the user selected and open that .csv file. The code will then mapp each space to a 2x2 matrix with a cell name attached. The code will then display to the user where to place each piece on which block. LEDs will also light up to quickly help the user

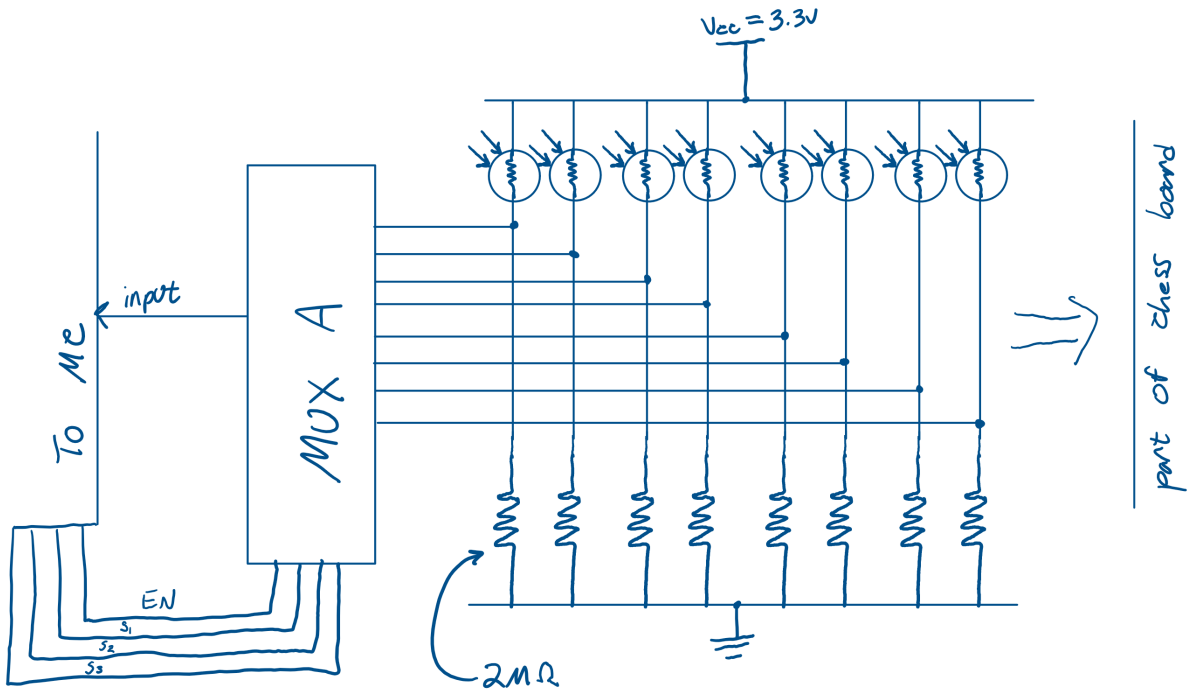

Figure 3: Multiplexer Sudo Circuit Diagram

where to place each piece. After each piece has been placed the code will shut off each LED light and display to the user which color he or she will play.

The code will have the ability to have the user use a button to light up LEDs to show the user the hint, and a switch to show the user the answer if they get stuck. If the wrong move is made on the chess board the code will have the ability to check each move and display if the move is correct or not. If the move is correct the display will show where the user needs to move the opponent pieces so the user can continue the puzzle until completion. If the move is incorrect the user will be asked to place the piece back and try again.

Interface description

The internal interface description consist of a display, switch, button, mulitiplexer, Potentiometer, and a SD Card. More information can be found in table 4. Charlieplexing is the method that will be used to turn on and off each LED that is needed to show the user where to put each piece, and also for show hint and or show answer. This equation to determine how many digital I/O pins needed to do this can be found in equation (3).

Charlieplexing

Pins $= \frac{1}{2} * (1 + \sqrt{1 + 4 * L}$; L=LED=64, Pins = 8.5

(3)

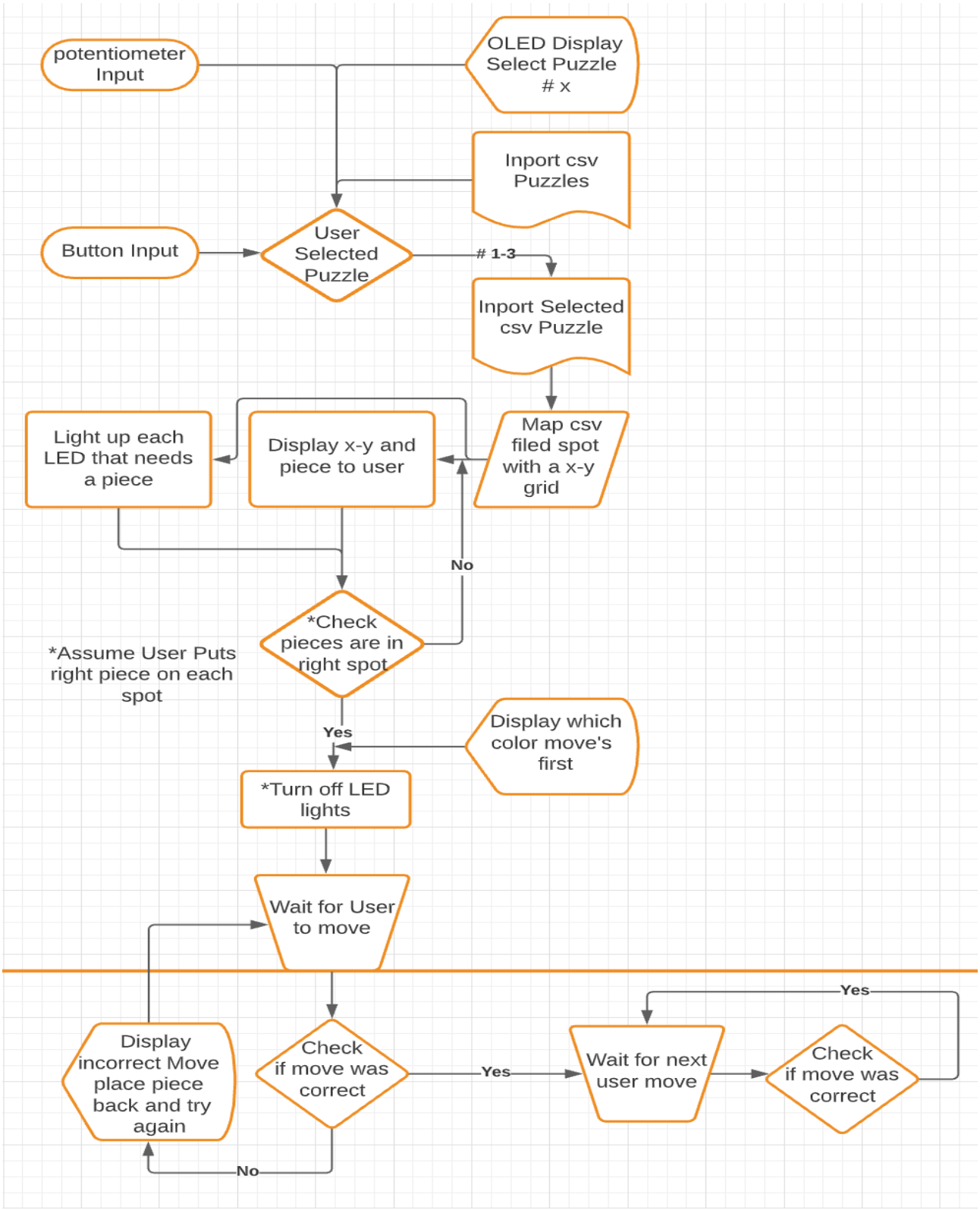

Figure 4: Code Flow Diagram

Interface Description	Description below
Display	one SCL 600MHz, one I2C
Mulitiplexer	3 select digital I/0 pin, 1 enable digital I/O Pin, 8 Input digital I/O Pins
Switch	one digital I/O input pin
Button	one digital I/O input pin
Potentiometer	one analog PWM I/O input pin
SD Card	I2C interface 4bit
LEDs	9 Digital I/O pins to light up 64 LEDS

Table 4: Shows the interface for Puzzle me Chess

Explanation of code

The following sub sections will describe different parts of the current code as it stands on 02/18/2021. There will be a brief explanation of the code layout which uses reference [1]. Other code explanations will be outlined in Main, Display, SDcard, Button,and Potentiometer, see sections below for more information.

Code layout

Figure 5 shows the modulatory of the Puzzle me Chess code. Each component of the circuit has its function and for that it will get its own .h and .cpp files. As you can see in figure 5. This code modulatory is talked about in reference [1] and it states laying out the .h files to have a method that is used for one function. An example of this would be the Display having a method for initialization of the display, a print or draw function, and a clear function. This modulatory was done for each component in the code and is clearly show in figure 5. For developing each method a .cpp file is used and these files will be discussed below in Listing 1, 2, 3, 4, and 5. All code below is current as of 02/18/2021 and is not finalized.

Main

The main.cpp file in Arduino houses the main code, Listing 1 shows the current main.cpp file. The top of main.cpp houses all the header files. Each header file is tied to its individual methods which are used in the individual .cpp files. Looking at Listing 1 you have the header files and next will come the declares. In the declare section there are method callers which grab the latest value of that method at any given time.

After the declares you call a function void setup() which generally would run one time and call most start up methods and initialization processes. Once the setup() function completes its run it then starts the main loop. In Arduino we use void loop() which loops every time it gets to the end but doesn't return any value. Currently I'm using two while loops which let the program pause as we wait for the user to enter in a value. The first loop has the user select a puzzle and then the second loop tells the user which puzzle has been selected and then will open that .csv file.

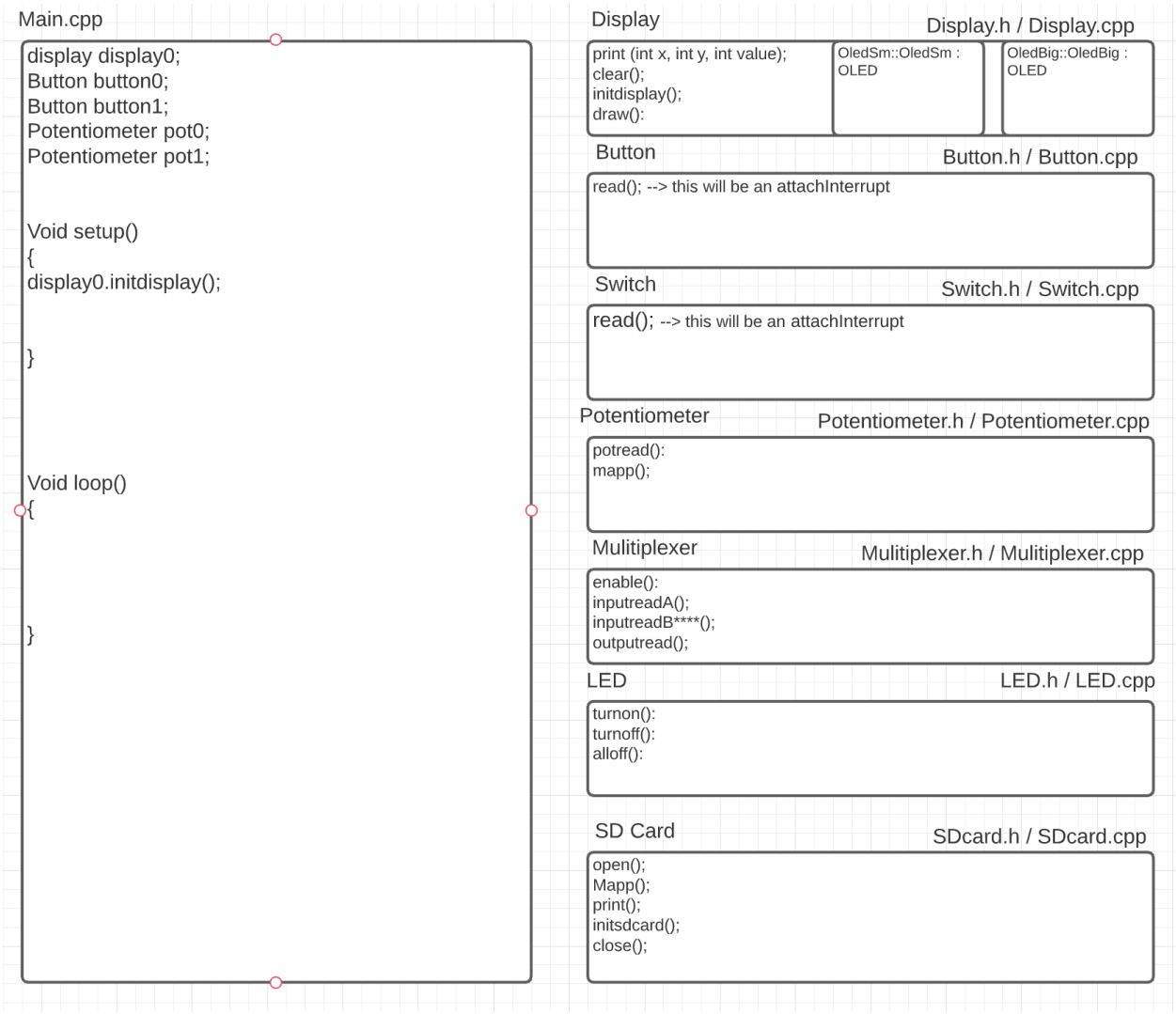

Figure 5: Module Code Layout

Listing 1: Puzzle me Chess - main.cpp file

```
//*****************************************Declare*********************//
Display Display0; // Setting Object 0 for Display
SDcard SDcard0;
Button Button0;
Potentiometer Potentiometer0;
//*****************************************Setup***********************//
void setup()
{
  Display0.int_display();
  SDcard0.int_SD();
//*****************************************Inputs**********************//
  Button0.init_button(); //setting D0 to button
  Potentiometer0.init_pot(); // setting A0 to pot
} // end setup

void loop()
{
//*****************************************Declare*********************//
Button Button1;
Potentiometer Potentiometer2;
Display Display1;
//*****************************************Start of Code***************//
while (Button1.r_button() == 0)
{

  Potentiometer Potentiometer1;
  Display1.print_select_puzzle(45, 30, Potentiometer1.r_pot());
}

delay(1000); //--> to allow for button press

while (Button1.r_button() == 0)
{
  Display1.print_user_puzzle(0,30, Potentiometer2.r_pot());
  SDcard0.open();
  break;
}
} //end void loop
```

Display

Listing 2 shows the .cpp file that holds all display methods that are used within the main.cpp file. Within the Display.cpp file there is the declare area which tells the code which display we are using and how to mapp the pins. After the declare the init.display() function is defined to start the display, followed by print.select.puzzle which shows the user which puzzle to select and the print.user.puzzle which will show the user which puzzle he or she selected.

Listing 2: Puzzle me Chess - display.cpp file

```
//*****************************************Declare*********************//
U8G2_SH1106_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/U8X8_PIN_NONE);
//*****************************************Setup***********************//
void Display::int_display()
{
    u8g2.begin(); // Start the Library code for the Display
} // end void int_display
//*****************************************Functions*******************//
void Display::print_select_puzzle(int x, int y, int value)
{
    u8g2.clearBuffer();              // clear the internal memory
    u8g2.setFlipMode(0);             // Flips display 180 (1) = True
    u8g2.setFont(u8g2_font_9x18_tf); // choose a suitable font
    u8g2.drawStr(0, 12, "Select Puzzle");
    u8g2.drawStr(x, y, "#");
    u8g2.setCursor(60, 30); // set cursor location
    u8g2.print(value);
    u8g2.sendBuffer(); // transfer internal memory to the display
} // end void print_select_puzzle

void Display::print_user_puzzle(int a, int b, int value1)
{
u8g2.clearBuffer();
u8g2.setFlipMode(0);
u8g2.setFont(u8g2_font_9x18_tf);
u8g2.drawStr(0, 12, "User Selected");
u8g2.drawStr(a, b, "#");
u8g2.setCursor(15, 30);
u8g2.print(value1);
u8g2.drawStr(30, 30, "Puzzle");
u8g2.sendBuffer();
} // end void print_user_puzzle
```

SDcard

The SD card function is a big part of the Puzzle me Chess project. The goal of this code block is to mapp each of the 64 different spots on the .csv file to a spot on the chess board. After the mapping has been complete it will be used to light up each LED to show the user where to put each chess piece. The SD card code will need to be able to expect three different .csv files and mapp each to a different chess board layout and three completely different chess puzzles.

There are three main parts of the SD card code. The first part has you initialize the SDcard using code from reference [2], second you need to open the file that was selected by the user, and lastly you need to mapp the open file to a chess board layout. Listing 3 shows the second part of the SD card code. The code begins with opening a name of the .csv file and then it reads the file and prints the file on the serial terminal. After it prints the file it will close the file. Next I need to figure out how to mapp the SD card to a chess board layout. I will look to use a Arduino - Multi-Dimensional Arrays to mapp the SD card to the chess board.

Listing 3: Puzzle me Chess - SDcard.cpp file

```
void SDcard::open()
{
    if (!SD.begin(chipSelect))
    {
        while (true);
    }
    File dataFile = SD.open("1015704.CSV" ); //opening File T015704.csv

    // if the file is available, write to it:

    if (dataFile)
    {
        while (dataFile.available())
        {

            Serial.write(dataFile.read());
        }
        dataFile.close();
    }

    // if the file isn't open, pop up an error:

    else
    {
        Serial.println("error opening 1015704.CSV");
    }
}
```

Button

The button in Puzzle me Chess is used as a user interaction button. The user will have the ability to select which puzzle he or she would like to do. Listing 4 shows the .cpp file that is currently used in the code. You have two methods. a initialize function that sets up the button to pin 0 as an input and a read button which sets the starting value of button to be equal to 1 and then to read the button and return a value of buttonstate1.

Listing 4: Puzzle me Chess - button.cpp file

```
//*****************************************Functions*******************//
void Button::init_button()
{
pinMode(2, INPUT); // sets the digital pin 0 as input
} // end init_button

int Button::r_button()
{
int buttonstate1 = 1;
buttonstate1 = digitalRead(button1);
return (buttonstate1);
} // end r_button
```

Build of Material	Item	Quantity
Display	Teyleten Robot SongHe OLED Display Module 128x64	1
Mulitiplexer	Texas Instruments: part CD4051BE	8
Switch	Ximimark 2 Position SK12D07 3 Pin	1
Button	Button Caps (square)	1
Potentiometer	DGZZI 10k Trim Potentiometer with Knob: part 3386MP-103	1
SD Card	Micro Center 16GB Micro SD Card Class 10 Micro SDHC	1
LEDs	Novelty Place 100 Pcs 5mm White LED Diode Lights	64
Photoresister	eBoot Photo Light Sensitive Resistor Light	64
2M resister	EDGELEC 100pcs 2M ohm Resistor 1/2w (0.5Watt) ±1%	64
75 resister	EDGELEC 100pcs 75 ohm Resistor 1/2w (0.5Watt) ±1%	64

Table 5: Shows the BOM for Puzzle me Chess

Potentiometer

The potentiometer is used as a user interaction point to have the user select which puzzle he or she would like to do. Listing 5. shows the .cpp file for the potentiometer. The file has a simple read pot value that is also mapped to 1-3. So when the user turns the potentiometer the user will have three values to choose from. The file also has an initialize method which sets the potentiometer to AO on the microcontroller.

Listing 5: Puzzle me Chess - Photentiometer.cpp file

```
//*****************************************Functions*******************//
void Potentiometer::init_pot()
{
  pinMode(0, INPUT); // pin A0 mapped to an INPUT --> r_pot
} // end init_pot

int Potentiometer::r_pot()
{
  int potmap1 = map(pot1, 0, 1023, 1, 3); // map values 1-3
  return (potmap1);
} // end r_pot
```

Material and resource requirements

In this section we will discuss the list of items used in Puzzle me chess, the test equipment that will be used to test the code and test the circuit, and where we are ordering all components from and when we are due in.

List of Items (BOM)

Table 5 shows the build of materials (BOM) used in the Puzzle me Chess project.

Test equipment

The Puzzle me Chess project consist of little test equipment needed to complete this project, however the ones that will be used are listing below. Test circuits will be used to insure how to

use a Charlieplexing circuit and the sudo mulitiplexer circuit layout before assembly of the entire circuit.

- BK precision 2709B MM
- Digital Analog Discovery 2

  Oscilloscope

  Logic Analyzer

  Data Logger

  Power Supply
- 6236B Triple Output Power Supply

Ording from

Table 6 shows the current list of items that need to be ordered. The Display, Switch, Potentiometer and microcontroller have arrived already.

BOM to be ordered	Website	Arrival
Mulitiplexer	Digikey	March 12
White LEDs	Amazon	February 25
Photoresister	Amazon	February 25
2MΩ resister	Amazon	February 25
75Ω resister	Amazon	February 25

Table 6: Shows the BOM for Puzzle me Chess

Development Plan and Schedule

Figure 6 and figure 7 show the Development plan and schedule for the Puzzle me Chess project. Figure 6 shows the main part of the development plan and is set up in chronological order to finish the code. You can see Figure 6 that Task 1 is complete and Tasks 2-3 have been started. Figure 7 shows building of the physical chess board and the final demo and final documentation edits.

Milestones and Schedule

Below shows the completed and scheduled milestones for the Puzzle me Chess project. These milestones are taken from the course website.

- Completed: 5 Jan 1st Meeting; Project Concept Development
- Completed: 1 Feb Project Concept Development
- Completed: 8 Feb Submit Present Preliminary Design and Present to Class
- 22 Feb Submit Critical Design Document for Grade (no presentation)
- 1 Mar - 3 May Project Development & Weekly Check-ins
- 22 Mar Holiday Break
- 10 May Project Demonstration

Project Development				
Task 1	Organize and Structure code, set up github project to push/pull	100%	2/18/21	2/28/21
Task 2	Re-Learn the basics of C++	40%	2/22/21	3/4/21
Task 3	Read Folders and Files from Micro SD Card	10%	2/20/21	2/27/21
Task 4	Mapp data in csv file to an X-Y coordinate and display piece on OLED	0%	2/21/21	3/3/21
Task 5	Display who goes first with the selected Puzzle	0%	2/22/21	2/27/21
Task 6	Figure out how to incorporate the next move from the csv file	0%	2/20/21	3/12/21
Task 7	Mapp potentiometer to 1-3 for the user to select which puzzle to do	0%	2/22/21	2/27/21
Task 8	Display on OLED User select Which puzzle to do	0%	2/22/21	2/27/21
Task 9	Build MUX circuitry and develop code to control it	0%	3/1/21	3/9/21
Task 10	Build LED Matrix circuitry and develop code to control it	0%	3/11/21	3/21/21
Task 11	Link Mapped csv to light up each LED that requires a piece	0%	3/15/21	3/22/21
Task 12	Have MUX circuitry read when a piece has been placed on the board	0%	3/22/21	4/6/21
Task 13	Have all LED turn off when all pieces are on the board	0%	4/6/21	4/16/21
Task 14	Display OLED whos turn it is on the board and wait for user to move	0%	4/6/21	4/8/21
Task 15	Check User move with Mapped CSV file	0%	4/6/21	4/8/21
Task 16	Display to use where to move opponents pieces	0%	4/10/21	4/20/21

Figure 6: Project Development

21x21 Chess Board				
Task 1	**Build 21x21 Chess Board with MDF Wood**	0%	3/6/21	3/24/21
Task 2	**Dill two holes per block for LED and Photoresister**	0%	3/6/21	3/25/21
Project Demonstation and Final Documentation				
Task 1	**Demo Script**	0%	4/8/21	4/13/21
Task 2	**Assembly instructions**	0%	4/14/21	4/24/21
Task 3	**Performance and measurement data**	0%	4/16/21	4/26/21
Task 4	**Final Documentaiton Edits**	0%	5/6/21	5/21/21

Figure 7: Chest Board Development and Project Demonstration

Risk

Below shows the Risk that will be involved in making Puzzle me Chess.

- First Real microcontroller design project
- Re-learning C++, and C#
- Learning how to mapp the SD card to an X-Y cornet system
- Making the code modular so I can upload a puzzle using an .csv file to the board
- Time, not a lot of room for a lost week
- Using proper coding techniques
- Turn on multiple LEDs to show user where to put all pieces on the chess board

Risk Meditation

- Use pre-existing libraries for the SD card reader and Display
- Break the code down into different sub files for origination and readability
  .h and .cpp files → Used in main.cpp for function call backs, less code development
- Complete a document that list major c++ functions that will be used in this project
- Keep an eye one time and keep Gnatt chart up to date

Conclusion

The Puzzle me Chess project has the capability to take in a .csv file with a standardized format and have the user select which puzzle he or she wants to do. This project also checks for completeness and will have a show hint and show answer feature. This project will challenge me and will exhaust my intelligence to complete this project. After completing this project I will be able to work with SD cards in microcontrollers, be proficient in C++ and embedded C, and be able to use Mulitiplexers with Charlieplexing method. Lastly this project will allow me to exercise my engineering and project management skills to improve for my professional engineering career.

References

[1] Brain Stone. *C++ Functions Modular Program Design.* (English) *Lecture presented at W 3.1, 3.2* . Dublin City University Glasnevin, Dublin, 2001.

[2] *Team, T. (n.d.). Using the SD library to retrieve information over a serial port. Retrieved February 18, 2021, from https://www.arduino.cc/en/Tutorial/LibraryExamples/CardInfo*