Mastermind

Milestone 4 - Final Design Document

Submission date: Wednesday, April 13th, 2022

Team members:
Ben Fleming
Colby Foster
Duncan Campbell
Shea McLaughlin

ECE3232

Duncan Campbell	
Benjamin Fleming	
Colby Foster	
Shea McLaughlin	

Table of Contents

List of Figures

List of Tables

1.0 | Introduction

In the late 1970s, shortly after the release of the first video game consoles, Hasbro and other game companies began releasing electronic handheld games [1]. Simon, Merlin, Game and Watch, and many other electronic systems were flying off shelves and into the hands of North American families. These systems provided countless hours of entertainment and brain training to generations of children. Today, these devices are rarely viewed as the first option when considering handheld consoles. We wish to bring back the charm of rudimentary, single-purpose, handheld electronic games with a modified version of Mastermind.

Mastermind, originally released as a board game in 1972 is played as follows [2]:

1. Using the 6 available colours, Player 1 secretly selects a 4-colour code.
2. Using the 6 available colours, Player 2 guesses a 4-colour code.
3. Player 1 informs Player 2 how many of their colour guesses were right in the right spot, and how many of their guesses were right, but in the wrong spot.
4. Steps 2, 3 are repeated until Player 2 exceeds 12 guesses or guesses the secret code correctly.

Over the next decade, several variants of the original Mastermind game would be developed. Variants such as Word Mastermind and Number Mastermind would change the secret code to be constructed using letters or numbers, while other variants such as Mini Mastermind would modify the number of possible colour choices, the maximum allowed number of guesses, or the length of the secret code [1].

The system to be developed will be a single-player rendition of Mastermind. The secret code will be randomly generated upon starting a game and all feedback to the player will be computed and displayed by the system. Additionally, the game will operate with the following rules:

- Possible colour choices: 6
- Maximum number of allowed guesses: 8
- Length of secret code: 4 (No repeated colours)

2.0 | System Overview

The game will be implemented using several interconnected blocks, these are the inputs, the outputs, and the controller of the system all working together to create an enjoyable handheld experience. This network is shown in Figure 1 and described in the following paragraphs.

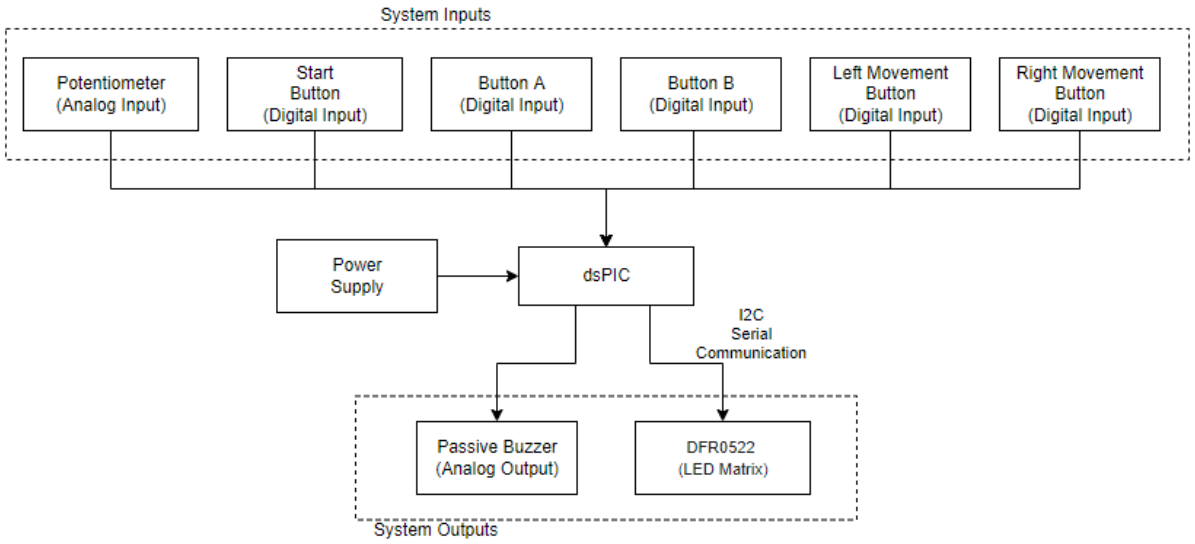

Figure 1: Block diagram showing connections between the inputs, the outputs, and the controller.

The Potentiometer block contains the colour selection method for the device. The block will communicate via an analog signal to the dsPIC. This will allow the user to control which colour they want to select for their next guess.

The Start Button block contains a button that will allow the device to be turned on and off via digital input.

The Button A block allows the user to confirm their selection of colours. Once the user has entered four colours for their guess, this button can be pressed to submit their guess to the system. The block will communicate via a digital signal to the dsPIC.

The Button B block allows the user to confirm their colour selection for an individual position. It will allow the user to build their guess of four colours so they can submit their guess. This button will communicate via a digital signal to the dsPIC.

The Left Movement Button block is the block which allows the user to move their selection cursor leftwards. The block communicates via a digital signal to the dsPIC to move the player's cursor on the UI (user interface).

The Right Movement Button block is the block which allows the user to move their selection cursor right. The block communicates via a digital signal to the dsPIC to signal to move the player's cursor on the UI.

The Power Supply, which will be a battery, will provide power for the entire device. The power supply is controlled by the Start Button. This will allow the user to turn the display for the device on and off as desired. Since the display will be the majority of the power draw for the system, this will allow for longer battery life.

The Passive Buzzer is one of the system’s outputs. The device takes an analog signal and outputs a sound based on the size and shape of that signal. This will be used to trigger

specific sounds when certain events occur in the game. For example, when the colour sequence is guessed correctly, and when the user runs out of guesses. It will be connected directly to the dsPIC, via an analog connection.

The DFR0522 block is the main display on which the game will be played. The display is an 8x16 LED matrix, which will show the user their current and previous guesses in the bottom 8x8 grid and will show the colour selection wheel in the top 8x8. The device uses I^2C to communicate serially with the dsPIC.

The dsPIC is the controller for the device. It will implement the logic of the game and handle the inputs and outputs of the game accordingly.

3.0 | Engineering Requirements

3.1 | Functional Requirements

3.1.1. The system shall have a start (power) button which will turn the system on and off.
3.1.2. The system shall allow the user to navigate left and right on the display using the Left and Right Movement Buttons.
3.1.3. The top half of the display shall display the colour selection wheel (6 colours, each a single pixel), and the currently selected colour (a 2x2 square in the centre).
3.1.4. The system shall allow the user to change the currently selected colour by rotating the potentiometer.
3.1.5. The system shall allow users to input a peg (colour selection) using the B button.
3.1.6. The system shall allow users to submit a completed guess using the A button.
3.1.7. After a completed guess is submitted, the system shall provide feedback in the four rightmost columns of the display indicating (in white) which pegs were right, in the wrong spot, and (in green) which pegs were right, in the right spot.
3.1.8. The system shall allow for sequential guesses and provide a display that shows the user their previous guesses.
3.1.9. The system shall display a visual message (a red flashing 'X") on the display after an unsuccessful game.
3.1.10. The system shall display a visual message (a green flashing checkmark) on the display after a successful game.
3.1.11. The system shall provide audio feedback (a single low-pitched buzz) for an unsuccessful game.
3.1.12. The system shall provide audio feedback (two high-pitched buzzes) for a successful game.
3.1.13. The system shall start a new game once the previous game is finished.

3.2 | Performance Requirements

3.2.1. All user input should have a response time under 0.1s and shall not have a response time exceeding 0.5s.

3.2.2. The system may be powered by a battery. The system shall remain powered when idle for at least one hour.

3.3 | Intermediate / Component Requirements

3.3.1. All 128 lights on the DFR0522 LED Matrix shall be able to be cleared and set to each of the 7 available colours.

3.3.2. Each of the 7 colours shall be distinguishable from each other at 1 ft away and should be distinguishable at 5ft away.

3.3.3. The potentiometer shall provide an output voltage ranging between 0V and 3.3V.

3.3.4. The output voltage from the potentiometer shall be converted to a binary codeword via the ADC module on the dsPIC.

3.3.5. The passive buzzer shall produce sound when supplied with a PWM signal.

3.3.6. The passive buzzer shall produce different sounds based on the characteristics of the signal provided.

3.3.7. Each of the buttons shall produce a digital signal, which will be high when pressed, and low when not pressed.

4.0 | Detailed Design

This section will delineate each of the blocks described in the System Overview, specify their inputs, outputs, and implementation. It will also describe how each block was individually tested.

4.1 | DFR0522 8x16 LED Matrix

4.1.1 | Inputs & Outputs

The DFR0522 (hereby referred to as LED matrix) acts as an output for the system and shall not have any outputs of its own. The LED matrix receives input via the I^2C connection with the dsPIC depending on the states of the system input blocks.

4.1.2 | System Implementation

The LED matrix will be connected to the dsPIC via the 5V, GND, SDA and SCL pins on MikrobusA with 10kΩ pull-up resistors connected to the SDA and SCL lines. See Figure 2 below.

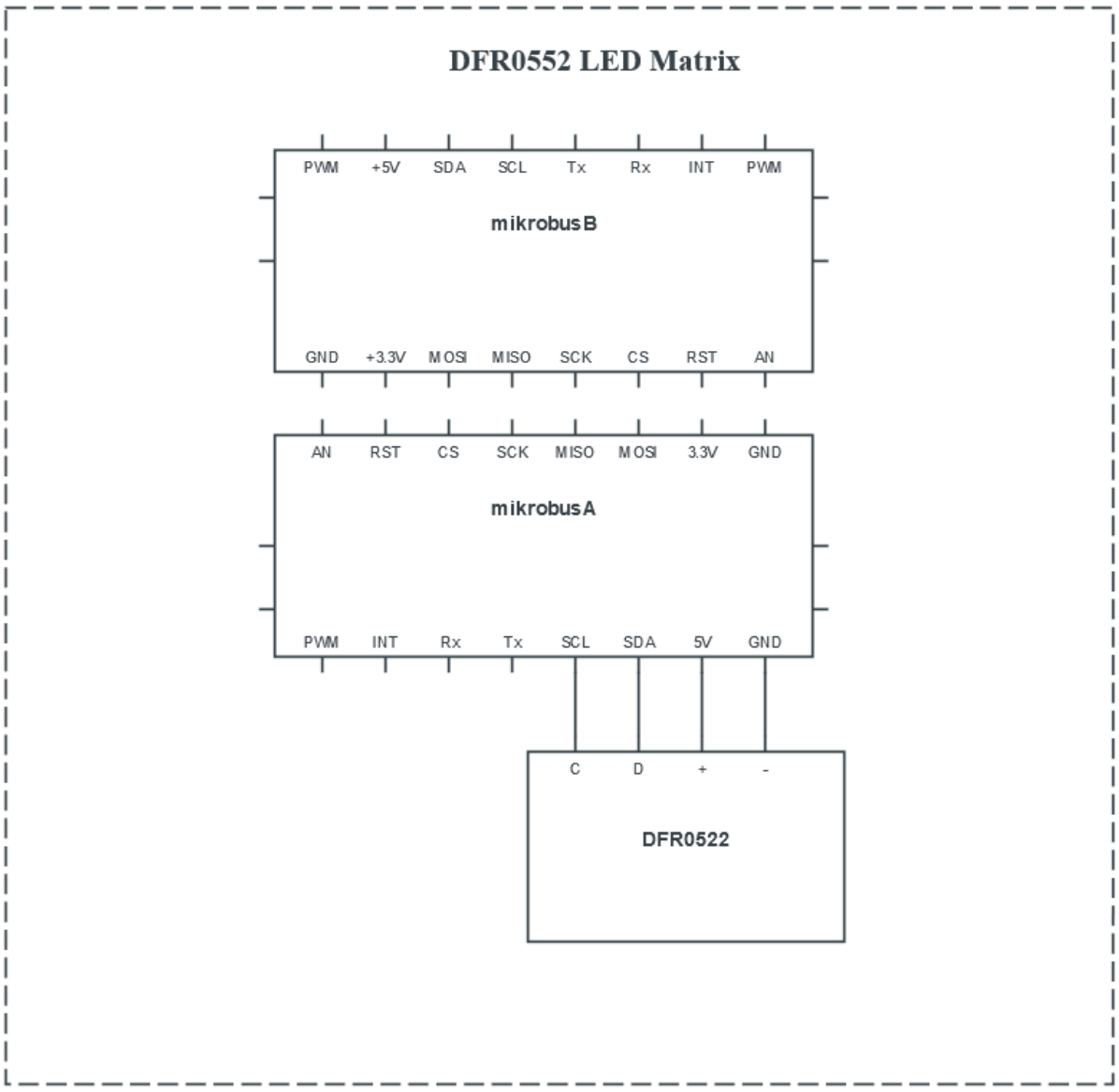

Figure 2: Schematic showing the connections from the dsPIC to the DFR0522.

The LED matrix will act as a peripheral in an I^2C connection with the dsPIC. Refer to Appendix B: block 4 for the code used to facilitate I^2C communication.

The state of the Mastermind game will be stored and manipulated in software; a single method call will be made to update the LED matrix when required. See the pseudocode below.

```
void updateDisplay(){
      for(pixel : gameArray){ // For all 128 pixels
            setPixel(pixel.x, pixel.y, pixel.color); // Send I2C command
      }
}
```

4.1.3 | Testing

Table 1 lays out the requirements which need to be met to ensure that the implementation of the DFR0522 LED matrix is satisfactory.

In order to test the LED matrix, the peripheral was connected to the board as shown in Figure 2 then, the following code was stepped through in debug mode in order to verify the functionality of all 128 LEDs. Additionally, the setPixel() method was called to set adjacent pixels to each of the 7 available colours. It was determined that all LEDs are functional and that each colour is distinguishable.

```
for(int color = 7; color >=0; color-){
      for(int x=0; x<8; x++){
            for(int y=0; y<16; y++){
                  setPixel(x, y, color);
            }
      }
}
```

4.2 | Potentiometer

4.2.1 | Inputs & Outputs

The potentiometer will be an input to the system and will therefore not receive any inputs from the system. The output of the potentiometer will be a voltage that is in the range of 0V-3.3V. It will be sent to the dsPIC for processing.

4.2.2 | System Implementation

As shown in Figure 3, the potentiometer will be connected to 3.3V, ground and the AN pin on MikrobusA. The output will be sent to the dsPIC on the AN pin to be converted into a digital value using the ADC module. This digital value then goes through the input process of the LED matrix to change the selected colour.

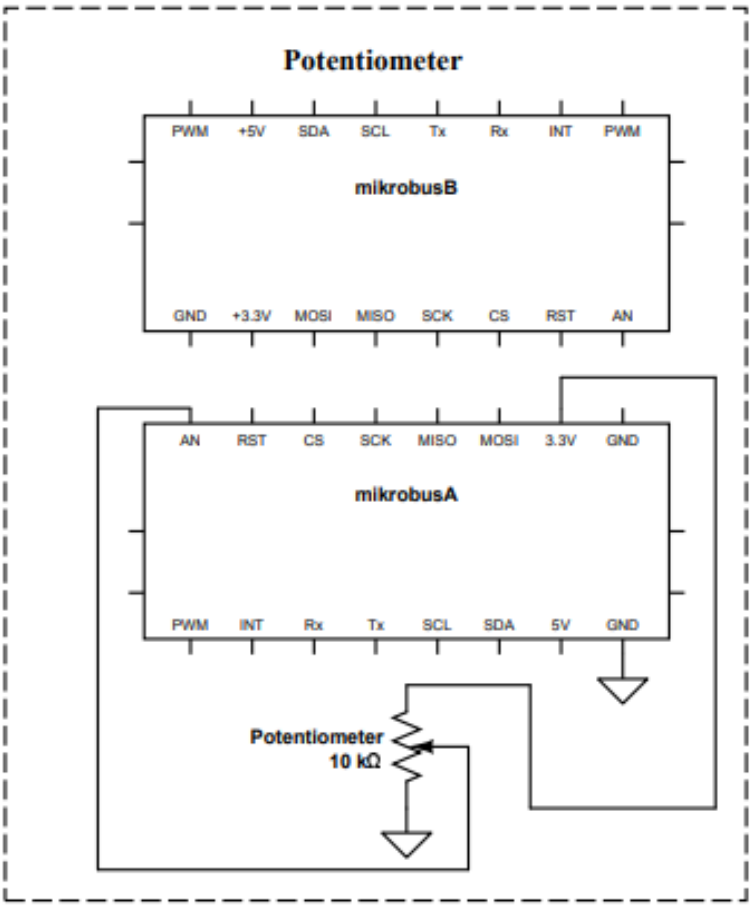

Figure 3: Schematic showing the connections used to implement the potentiometer as an analog input using an ADC.

Since the potentiometer uses an ADC module, the initialization (shown below) of an ADC was needed in order to retrieve the analog values. An interrupt was used to determine which colour was being updated depending on the analog value that was retrieved. The interrupt was called periodically alongside timer1. This was done by triggering the ADC in the timer1 interrupt which gained the appropriate value and used that value in the ADC interrupt to determine the desired colour.

```
IEC6bits.ADCAN18IE = 1; // Enables the AN_2 interrupt (18)
ADCON5Hbits.WARMTIME = 12; // 4096 source clock periods
ADCON1Lbits.ADON = 1; // Enables ADC module
ADCON5Lbits.SHRPWR = 1; // Enable ADC core
while(ADCON5Lbits.SHRRDY == 0); // Wait for ADC core to be ready

ADCON3Hbits.SHREN = 1; // Enables shared ADC core
ADIEHbits.IE18 = 1;  // Generate an interrupt specifically for AN_2 when ADC core is ready
ADTRIG3Lbits.TRGSRC18 = 1; // Enables AN_2 pin
```

4.2.3 | Testing

For testing the potentiometer, an interrupt was used along with the ADC to provide a voltage range to an LED. The schematic for this test is in Appendix A: figure 4. Using a multimeter on the pins of the potentiometer, the resulting voltage range was between 0V (potentiometer rotated to the left-most position) and 3.3V (potentiometer rotated to the right-most position). Viewing these results allowed us to confirm the intermediate requirement 3.3.3. Continuing the test, the debugger was used to provide a hexadecimal value for the left-most position and the right-most position. Once this hexadecimal range was found, the range was divided into 6 parts (colors). We found that each part had a range of 0x02AA that then was used as part of the conditions in an if statement to turn on the LED. Executing the if statements will confirm the intermediate requirement 3.3.4.

4.3 | Passive Buzzer

4.3.1 | Inputs & Outputs

The passive buzzer will act as an output for the system. It will be connected to the dsPIC and receive two inputs for the component to function. The component will output a sound depending on the input it receives.

4.3.2 | System Implementation

As shown in Figure 4, the passive buzzer will have connections to GND, 5V, and PWM. An NPN transistor will be used to drive the buzzer which will be controlled by the PWM pin. The passive buzzer requires analog voltage to function which is why PWM is being used. PWM will allow digital signals from the dsPIC to control analog devices.

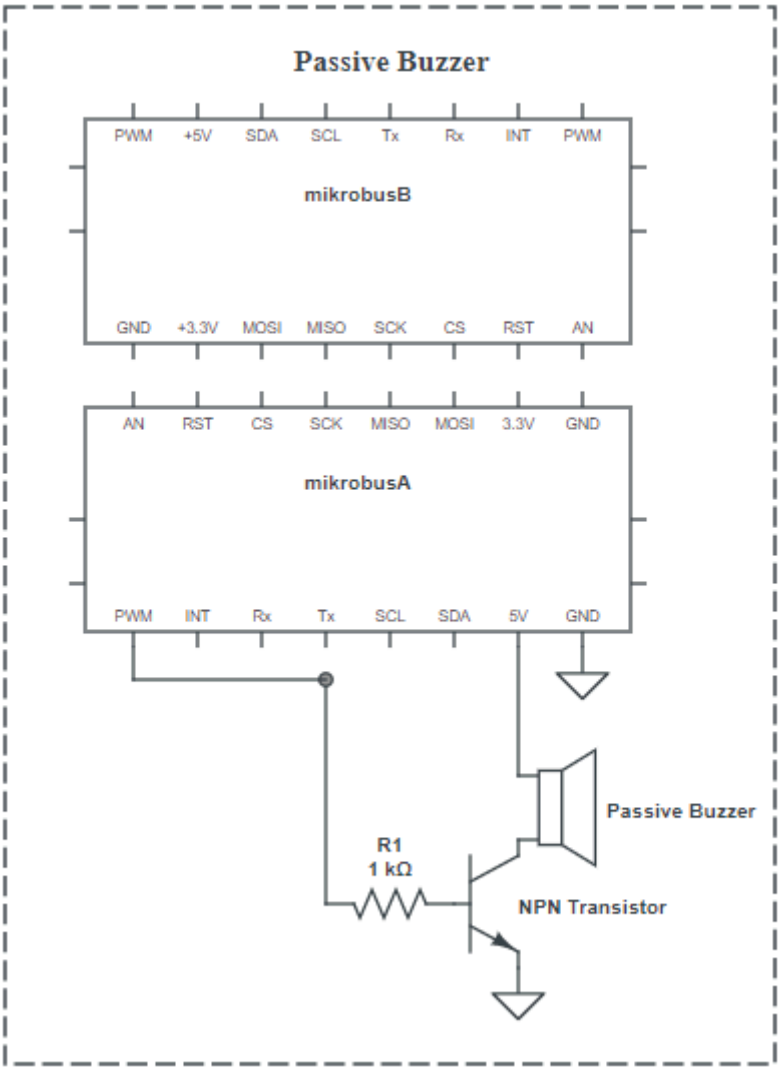

Figure 4: Schematic showing implementation of the passive buzzer.

To use PWM on the dsPIC it first had to be configured in software to function. Using a clock cycle, a specific resolution mode (independent mode), and setting the period, a PWM frequency was able to be produced. Specific methods were created such as turnOnBuzzer(), turnOnWinBuzzer(), and turnOnLoseBuzzer(). These methods were used to provide specific frequency levels to the buzzer in order to create the desired noise for the outcome. These methods required the modification of the PG8PER register and would turn on and off the PDM in order to not always have it running in the background while a game is currently being played.

```
void initPWM(void){
    // Buzzer
    PCLKCONbits.MCLKSEL = 0; // Master clock = Fosc
    PG8CONLbits.CLKSEL = 1; // Depends on master clock
    PG8CONLbits.MODSEL = 0; // Puts independent mode
    PG8CONLbits.HREN = 0; // Puts it into standard resolution mode
    PG8IOCONHbits.PENL = 1; // Uses only the Low pin
```

```
    PG8IOCONHbits.PMOD = 1; // Sets pin it into independent mode
    PG8DC = 1250;  // Duty Cycle is 25%
}
```

4.3.3 | Testing

To test the buzzer, the code shown in section 4.3.2 and the code shown in, Appendix B: block 5 was used. This allowed the buzzer to produce a sound and to meet the intermediate requirement 3.3.5. In order to test the intermediate requirement 3.3.6, we modified the PH8PER value. Increasing the value provided a lower pitch while decreasing the value provided a higher pitch.

4.4 | Button

4.4.1 | Inputs & Outputs

The buttons used will solely act as inputs for the system. Five buttons in total will be used to perform a variety of functions. They will be directly connected to the dsPIC via five different pins.

4.4.2 | System Implementation

Each button used in the system will follow a similar implementation except they will send data to different digital input pins in the final system implementation. Please refer to Appendix A: figure 7 through 10 to view the schematics for each individual button. For this document, each button was tested using the same pin on the dsPIC.

The start button will be connected to GND, and CS on MikrobusA. A 10kΩ pullup resistor is used to connect the button to the CS line. When the button is pressed, the logic level can be pulled from the CS pin on MikrobusA to then perform the function of turning the system on or off.

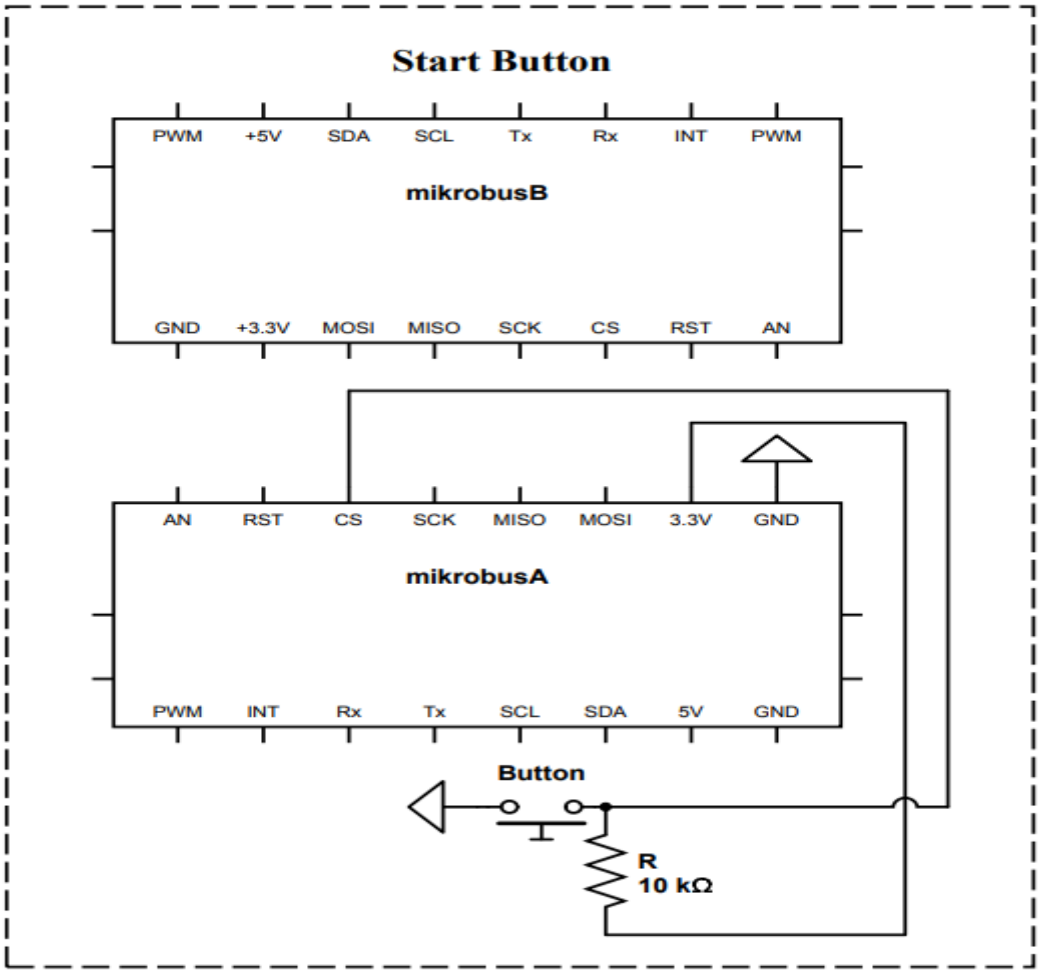

Figure 5: Schematic showing implementation of the start button.

4.4.3 | Testing

To test the buttons, the code in Appendix A: block 3 and an LED was used. An LED was configured with a resistor to receive an input from the dsPIC when the button was pressed. When the button was pressed, the LED turns on and when the button was unpressed, the LED turned off. This test revealed that two different logic levels could be received from a pin connected to a button. Therefore, requirement 3.3.7 has been met.

5.0 | Conclusion

In conclusion, we were satisfied with the final result of the project. All of our engineering requirements for the project were met, resulting in a working system. Throughout development, we were faced with several challenges. Firstly, designing software to communicate with the LED Matrix proved to be a challenge due to the lack of documentation on the DFR0522 peripheral. Secondly, configuring the PWM generator to function within our needs was a challenge. There was significant documentation on the configuration of the PWM generators, but trying to figure out what we needed from the vast amount of information proved to be time consuming. Additionally, configuring the ADC was difficult and time consuming; Determining how the ADC needed to be configured for the system proved difficult. Also, attempting to understand the result of the ADC conversion in our initial tests was challenging to decipher but as we got more comfortable with the concept, it became easy to understand. Each of these challenges proved to be frustrating, however, we were able to delegate roles and work cohesively as a team to create an end-product that we were very proud of.

While we were generally satisfied with the final result, we do recognize some downfalls of the system. Firstly, the user experience proved to be less seamless than we had envisioned. If we were to design the controls again, we would consider using a different colour selection mechanism such as a dial which would snap to specific positions, or a joystick which would replace both the colour selection and movement mechanisms. Additionally, we would have liked to implement better software debouncing of the buttons to allow for more frequent inputs from the user. Second, the solution for powering our device was not ideal. If we were to design the system again, we would have used a battery and paid closer attention to power consumption in idle mode. Finally, to fit the original vision of a handheld system, it would have been ideal to have designed an enclosure around the embedded system to hide the wiring, revealing only the control devices and screen to the user.

In total, our system met all of our engineering requirements and performed as expected, however if given more time, some design choices could have been reevaluated to provide a more seamless experience for the user.

Appendix A

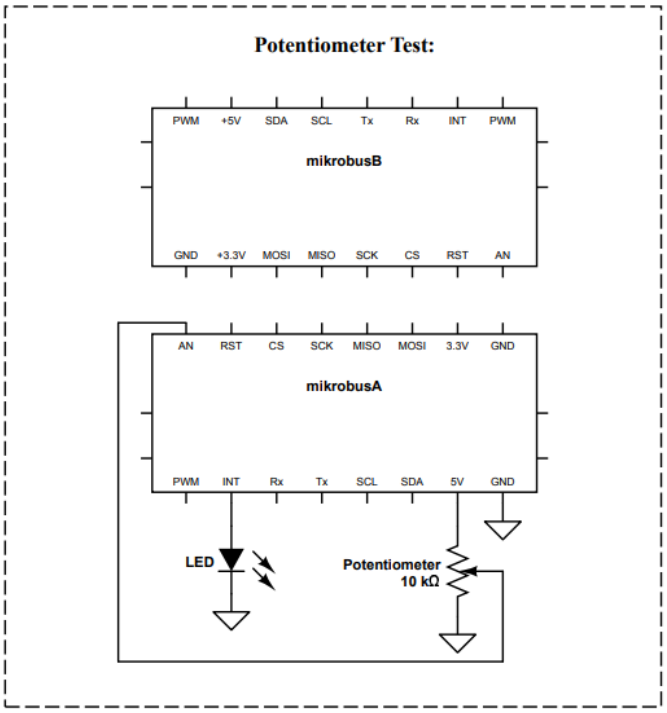

Figure 6: Schematic for testing the potentiometer.

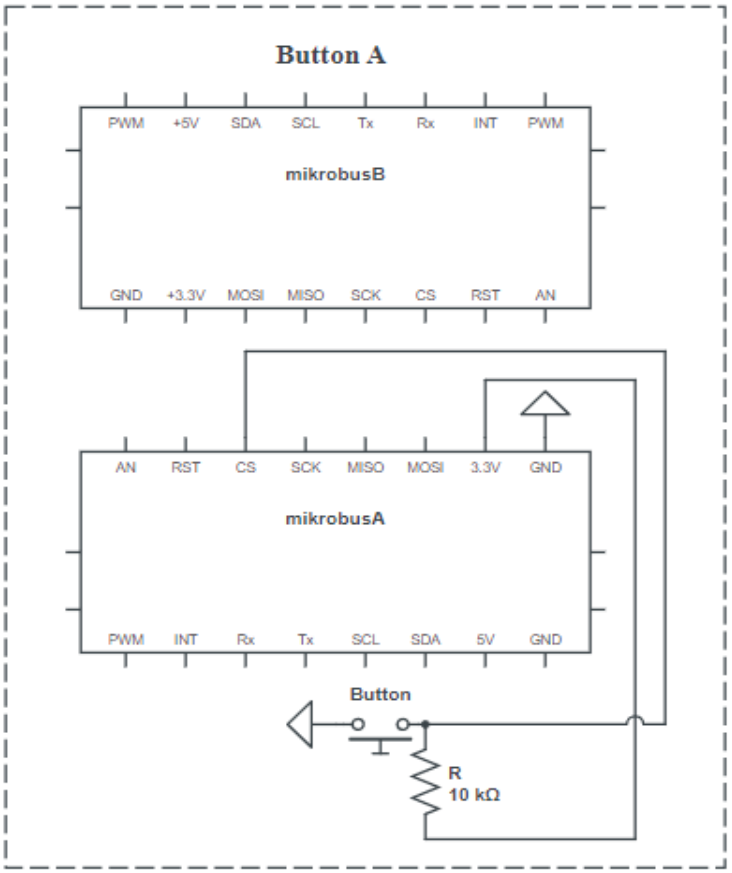

Figure 7: Schematic showing the connections used to test Button A.

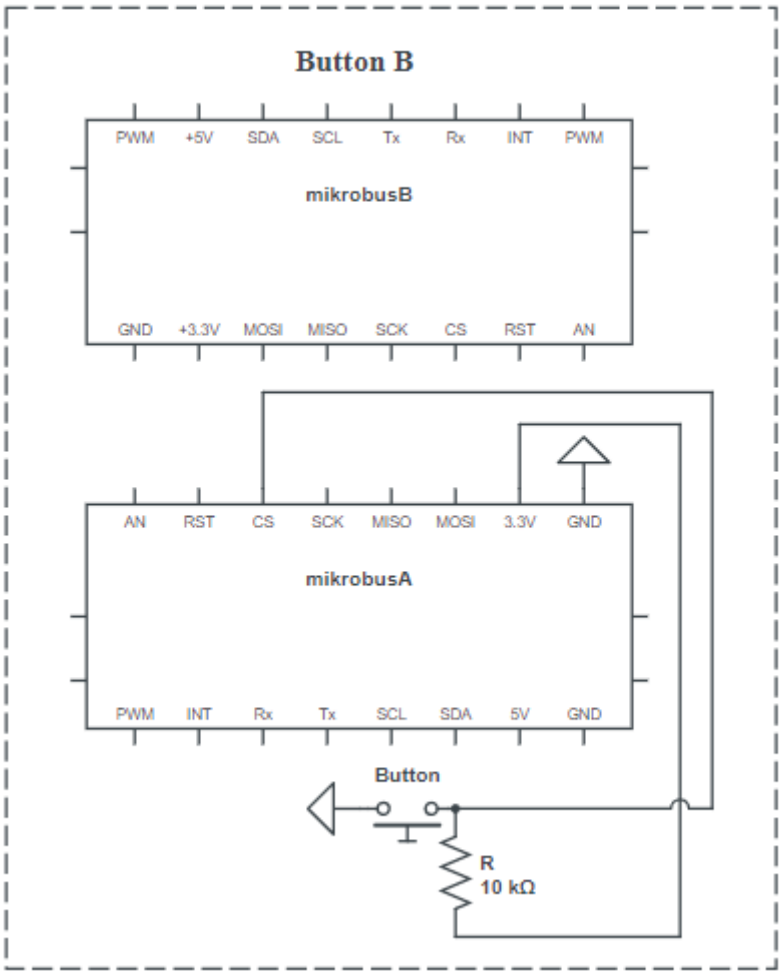

Figure 8: Schematic showing the connections used to test Button B.

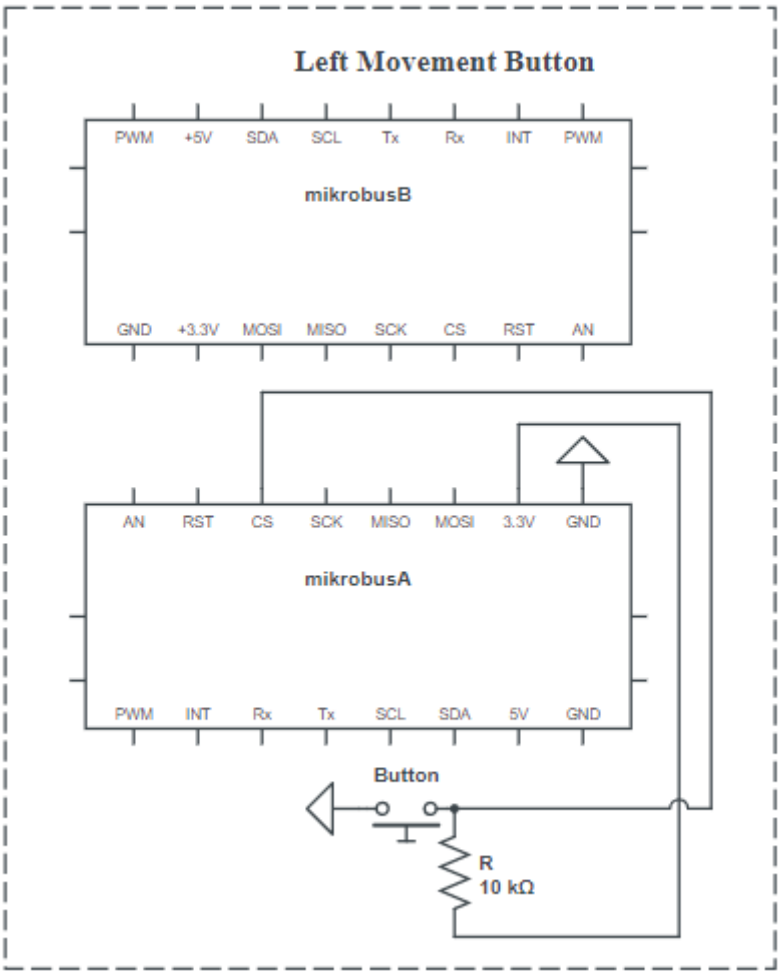

Figure 9: Schematic showing the connections used to test the Left Movement Button.

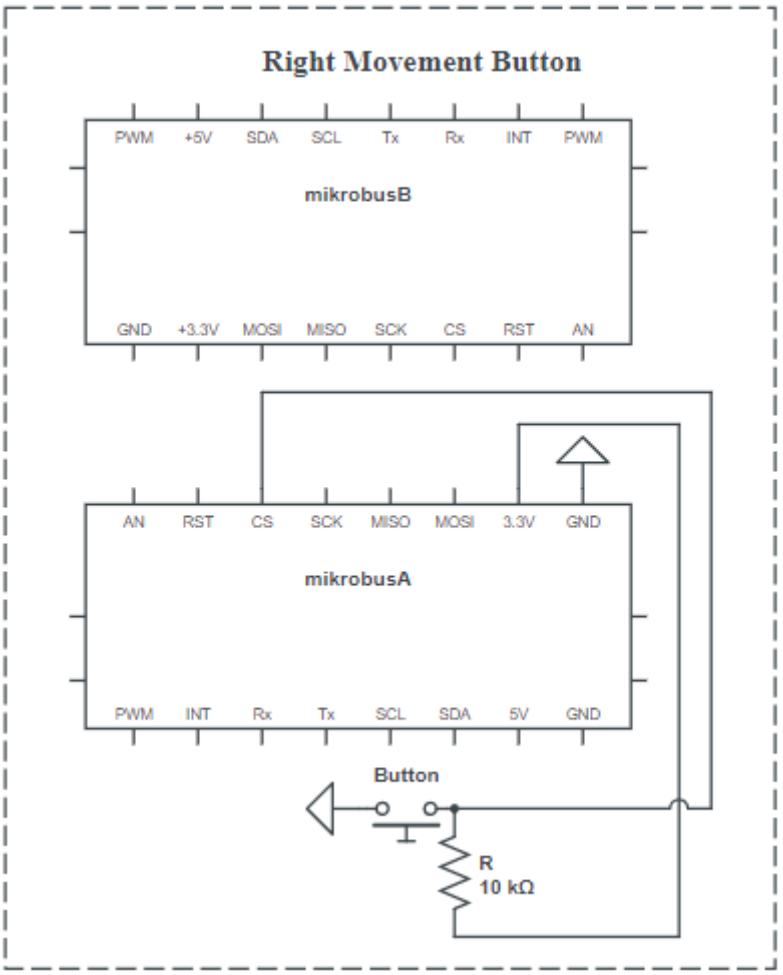

Figure 10: Schematic showing the connections used to test the Right Movement Button.

Appendix B

Code for Initializing Buzzer – Block 1

```
void initPWM(void){
    // Buzzer
    PCLKCONbits.MCLKSEL = 0; // Master clock = Fosc
    PG8CONLbits.CLKSEL = 1; // Depends on master clock
    PG8CONLbits.MODSEL = 0; // Puts independent mode
    PG8CONLbits.HREN = 0; // Puts it into standard resolution mode
    PG8IOCONHbits.PENL = 1; // Uses only the Low pin
    PG8IOCONHbits.PMOD = 1; // Sets pin it into independent mode

    PG8PER = 20000;
    PG8DC = 1250;

    PG8CONLbits.ON = 1; // Enabling the PWM generator

}
```

Code for Testing Potentiometer (Main) - Block 2

```
int main(void){
      TRISCbits.TRISC14 = 0; // LED
      init_analog();
      while(1){ LATCbits.LATC14 = getAnalogVal();
      return 0;
}
```

Code for button test - Block 3

```
int main(void){
      TRISCbits.TRISC3 = 0; // LED Output
      TRISDbits.TRISD3 = 1; // Button Input
      while(1){
            if(PORTDbits.RD3 == 1){
                  LATCbits.LATC3 = 0;
            } else {
                  LATCbits.LATC3 = 1;
            }
      }
}
```

I2C Communication Code - Block 4

```
void I2CStop(void)
{
      I2C1CONLbits.RCEN = 0;
      I2C1CONLbits.PEN = 1;
      while(I2C1CONLbits.PEN);
}
```

```
void I2CStart(void)
{
      I2C1CONLbits.ACKDT = 0;        // Reset any ACK
      I2C1CONLbits.SEN = 1;          // Start
      while(I2C1CONLbits.SEN);
}

void I2CInit(void)
{
      I2C1BRG = 20;
      I2C1CONLbits.I2CEN = 0;  // Disable I2C
      I2C1CONLbits.DISSLW = 1; // Disable slew rate control
      I2C1CONLbits.A10M = 0;   // 7-bit slave addr
      I2C1CONLbits.SCLREL = 1; // SCL release control
      IEC1bits.MI2C1IE = 0;   // Master I2C interrupt
      IFS1bits.MI2C1IF = 0;   // MI2C Flag
      I2C1CONLbits.I2CEN = 1;  // Enable I2C
}

void I2CIdle(void)
{
      while(I2C1STATbits.TRSTAT);
}

void I2CWrite(unsigned char c)
{
      I2C1TRN = c;
      while(I2C1STATbits.TBF);
}

void I2CSequentialWriteSingleReg(char addr, char byte, char* value, int
length){
      int j;
      I2CStart();
      I2CWrite((addr<<1)&0xFE);
      I2CIdle();
      I2CWrite(byte);
      I2CIdle();
      for(j = 0; j < length; j++) {
            I2CWrite(value[j]);
            I2CIdle();
      }
      I2CStop();
}

void SetPixel(int x, int y, int color){
      int addr = 16;
      int j;
      I2CStart();
```

```
	I2CWrite((addr<<1)&0xFE);
	I2CIdle();
	I2CWrite(2);
	I2CIdle();
	int buf[4] = {0};
	buf[0] = (buf[0] & (0xe6)) | (0x01 << 3);
	buf[1] = color;
	buf[2] = x;
	buf[3] = y;
	for(j = 0; j < 4; j++)
	{
		I2CWrite(buf[j]);
		I2CIdle();
	}
	I2CStop();
}
```

PWM Buzzer Test Code - Block 5

```
int main(void) {

    initPWM();

    // While loop must be here for the buzzer to work
    while(1){}
    return 0;
}
```

Appendix C

Table 2: Verification testing plan

Requirement #	Requirement Tested	Test Procedure	Expected Result	Actual Result	Pass / Fail
3.1.1	The system shall have a start (power) button which will turn the system on and off.	1. While the system is idle (off), press and release the start button	1. The LED display turns on, a game has started	Same as expected	Pass
		2. While the system is on, press and release the start button	2. The LED display turns off, the system is not consuming power	Same as expected	Pass
3.1.2	The system shall allow the user to navigate left and right on the display using the Left and Right Movement Buttons.	1. While the system is idle (off), press and release the start button	1. The LED display turns on, a game has started	Same as expected	Pass
		2. Note the bottom row of the display	2. The bottom-left most LED is periodically blinking white to indicate that it is selected	Same as expected	Pass
		3. Press the Left Movement Button	3. The bottom-left most LED is periodically blinking white to indicate that it is selected	Same as expected	Pass
		4. Press the Right Movement Button	4. The blinking LED shifts one space to the right.	Same as expected	Pass
		5. Press the Right Movement Button	5. The blinking LED shifts one space to the right.	Same as expected	Pass
		6. Press the Left Movement Button	6. The blinking LED shifts one space to the left.	Same as expected	Pass
3.1.3	The top half of the display shall display the colour selection wheel (6 colours, each a single pixel), and the currently selected colour (a 2x2 square in the centre).	1. While the system is idle (off), rotate the potentiometer to the right most position.	1. N/A as the device is not powered.	Same as expected	Pass
		2. Press and release the start button.	2. The LED display turns on, a game has started	Same as expected	Pass

		3. Note the top 8x8 display.	3. The colour wheel is clearly shown on the display. The 2x2 square in the centre of the wheel is coloured with the colour corresponding to the position of the potentiometer.	Same as expected	Pass
		4. Rotate the potentiometer to the left most position	4. The 2x2 square in the centre of the wheel is coloured with the colour corresponding to the new position of the potentiometer.	Same as expected	Pass
		5. Slowly rotate the potentiometer clockwise	5. As the potentiometer is rotated, the 2x2 square in the centre of the wheel is updated to show the colour corresponding to the current position of the potentiometer.	Same as expected	Pass
3.1.4	The system shall allow the user to change the currently selected colour by rotating the potentiometer.	1. Perform test 3.1.3	1. Results are as expected	Same as expected	Pass
3.1.5	The system shall allow users to input a peg (colour selection) using the B button.	1. While the system is idle (off), press and release the start button	1. The LED display turns on, a game has started	Same as expected	Pass
		2. Note the LED display.	2. The 2x2 square in the centre of the colour wheel is coloured corresponding to the position of the potentiometer. The bottom-left most LED is periodically blinking.	Same as expected	Pass
		3. Press the B button.	3. The bottom-left most LED is now blinking periodically between white and the	Same as expected	Pass

			selected colour.		
		4. Press the Right Movement Button.	4. The bottom-left most LED is now a solid colour. The second LED in the bottom row is periodically blinking.	Same as expected	Pass
		5. Press the Right Movement Button again.	5. The bottom-left most LED is a solid colour. The third LED in the bottom row is periodically blinking.	Same as expected	Pass
		6. Press the B button.	6. The third LED in the row is now blinking periodically between white and the selected colour.	Same as expected	Pass
		7. Rotate the potentiometer to a new position.	7. The 2x2 square in the centre of the colour wheel is updated to show the colour corresponding to the current position of the potentiometer.	Same as expected	Pass
		8. Press the B button.	8. The third LED in the row is now blinking periodically between white and the newly selected colour (Overwriting the previous colour).	Same as expected	Pass
		9. Press the Left Movement Button.	9. The first and third LEDs are solid colours. The second LED is periodically blinking.	Same as expected	Pass
		10. Press the B button.	10. The first and third LEDs are solid colours. The second LED is periodically blinking between white and the selected colour.	Same as expected	Pass
3.1.6	The system shall allow users to submit a completed guess using the A button.	1. Perform the steps outlined in test 3.1.5.	1. The first and third LEDs are solid colours. The second LED is periodically blinking between white and the selected	Same as expected	Pass

			colour.		
		2. Press the Right Movement Button twice.	2. The first three LEDs are solid colours. The fourth LED is periodically blinking.	Same as expected	Pass
		3. Press the A button.	3. No change.	Same as expected	Pass
		4. Press the B button.	4. The first three LEDs are solid colours. The fourth LED is periodically blinking between white and the selected colour.	Same as expected	Pass
		5. Press the A button.	5. The guess is submitted. The LED display is updated to show up to four response LEDs in the four rightmost columns.	Same as expected	Pass
3.1.7	After a completed guess is submitted, the system shall provide feedback in the four rightmost columns of the display indicating (in white) which pegs were right, in the wrong spot, and (in green) which pegs were right, in the right spot.	1. Start the system	1. The LED display turns on, a game has started	Same as expected	Pass
		2. Using the Left and Right Movement Buttons, the Potentiometer and the B button, input a colour sequence and submit the guess using the A button.	2. The LED display is updated to show up to four response LEDs in the four rightmost columns.	Same as expected	Pass
3.1.8	The system shall allow for sequential guesses, and provide a display that shows the user their previous guesses.	1. Perform the steps from test 3.1.7.	1. The system has the same outcomes as test 3.1.7.	Same as expected	Pass
		2. Observe the bottom half of the LED display.	2. The LED display will allow the user to input another guess on the line above the previous guess.	Same as expected	Pass
		3. Repeat step 2 of test 3.1.7 at least 3 times.	3. The LED display should perform the same as in step 2.	Same as expected	Pass

3.1.9	The system shall display a visual message (a red flashing 'X") on the display after an unsuccessful game.	1. Turn on the system.	1. The display turns on and a game is started.	Same as expected	Pass
		2. Using the Left and Right Movement Buttons, the Potentiometer and the B button, input a colour sequence and submit the guess using the A button.	2. The LED display is updated to show up to four response LEDs in the four rightmost columns. The blinking LED moves to the first position on the next line.	Same as expected	Pass
		3. Repeat step 2 until the maximum number of guesses is exceeded.	3. Outcome is the same as in step 2.	Same as expected	Pass
		4. Observe the top half of the LED display.	4. The LED display shows an animation that notifies the user that they have not guessed the correct sequence	Red X does not flash as indicated in the test description. This behaviour was expected.	Pass
3.1.10	The system shall display a visual message (a green flashing checkmark) on the display after a successful game.	1. Turn on the system	1. The display turns on and a game is started.	Same as expected	Pass
		2. Repeat step 2 from test 3.1.9 until the correct sequence is guessed.	2. The outcome should be as described in step 2 from test 3.1.9.	Same as expected	Pass
		3. Once the correct sequence is guessed, observe the LED display.	3. The LED display shows an animation that notifies user that they have guessed the correct sequence	Green checkmark does not flash as indicated in the test description. This behaviour was expected.	Pass
3.1.11	The system shall provide audio feedback (a single low pitched buzz) for an unsuccessful game.	1. Turn on the system.	1. The display turns on and a game is started.	Same as expected	Pass
		2. Using the Left and Right Movement Buttons, the Potentiometer and the B button, input a colour sequence and submit the guess using the A button.	2. The LED display is updated to show up to four response LEDs in the four rightmost columns. The blinking LED moves to the first position on the next line.	Same as expected	Pass

		3. Repeat step 2 until the maximum number of guesses is exceeded.	3. Outcome is the same as in step 2.	Same as expected	Pass
		4. Listen.	4. The passive buzzer briefly produces a low pitched buzz before stopping.	Buzzer produces two sequential low pitched tones. This behaviour is different than indicated here, but was expected.	Pass
3.1.12	The system shall provide audio feedback (two high pitched buzzes) for a successful game.	1. Turn on the system	1. The display turns on and a game is started.	Same as expected	Pass
		2. Repeat step 2 from test 3.1.9 until the correct sequence is guessed.	2. The outcome should be as described in step 2 from test 3.1.9.	Same as expected	Pass
		3. Once the correct sequence is guessed, observe the LED display.	4. The passive buzzer briefly produces a high pitched buzz before repeating once and then stopping.	Buzzer produces four tones each increasing in pitch. This behaviour is different than indicated here, but was expected.	Pass
3.1.13	The system shall start a new game once the previous game is finished.	1. Turn on the system.	1. The display turn on and a game is started.	Same as expected	Pass
		2. Using the Left and Right Movement Buttons, the Potentiometer and the B button, input a colour sequence and submit the guess using the A button.	2. The LED display is updated to show up to four response LEDs in the four rightmost columns. The blinking LED moves to the first position on the next line.	Same as expected	Pass
		3. Repeat step 2 until either the maximum number of guesses is exceeded or the correct sequence is guessed.	3. The game is finished.	Same as expected	Pass
		4. Observe the top half of the LED display.	4. The appropriate animation and sound will play.	Same as expected	Pass

		5. Once the animation/sound is finished, press the A button or B button.	5. The system will reset and allow the user to play another game.	Same as expected	Pass
3.2.1	All user input should have a response time under 0.1s and shall not have a response time exceeding 0.5s.	1. Perform test 3.1.10.	1. At all points during the execution of the test, the display should update within a 0.5s timeframe after an input.	Display response time <0.33 seconds.	Pass
3.2.2	The system may be powered by a battery. The system shall remain powered when idle for at least one hour.	1. While the system is idle (off), press and release the start button	1. The LED display turns on, a game has started	Same as expected	Pass
		2. Leave the system inactive for an hour.	2. The system remains on.	Same as expected	Pass
		3. Observe the LED display.	3. The system remains on after an hour of inactivity.	The final implementation was not powered by a battery, it did however remain powered after 1 hour of inactivity as expected.	Pass

References

[1] Electronic Game, wikipedia.org. Retrieved January 27, 2022, from https://en.wikipedia.org/wiki/Electronic_game

[2] Mastermind game rules. (n.d.). Retrieved January 27, 2022, from http://www.industrious.com/mastermind/gamerules.html