



THE UNIVERSITY *of*
TULSA

*Electrical and Computer
Engineering*

Rock, Paper, Scissors, Lizard, and Spock
Final Design Report

Noah Simpson
Will Ferguson
Zade Mahayni

December 14, 2022

Revision History

| Date | Version | Author | Description |
|-----------|---------|-------------------------------------|------------------|
| 14DEC2022 | 1.0 | Simpson N., Ferguson W., Mahayni Z. | Finished project |

Customer Approval

| Name | Role | Signature of Approval | Approval Date |
|-----------------|----------|-----------------------|---------------|
| Team Member | Engineer | | 15DEC2022 |
| Team Member | Engineer | | 15DEC2022 |
| Nathan Hutchins | Customer | | 15DEC2022 |

Contents

| | |
|--|-----------|
| Revision History | i |
| Customer Approval | ii |
| Table of Contents | iv |
| List of Figures | v |
| List of Listings | vi |
| Abstract | 1 |
| Nomenclature | 2 |
| 1 Introduction | 3 |
| 1.1 Introduction | 3 |
| 1.2 Objective | 3 |
| 1.3 Customer | 3 |
| 1.4 Design Team | 3 |
| 1.4.1 Engineering Co-Manager/Engineer/Programming Specialist | 3 |
| 1.4.2 Engineering Co-Manager/Engineer/LCD Specialist | 3 |
| 1.4.3 Engineering Co-Manager/Programmer/Keypad Specialist | 3 |
| 2 Project Requirements | 4 |
| 2.1 Hardware Requirements | 4 |
| 2.1.1 NUCLEO F411RE | 4 |
| 2.1.2 Keypad | 4 |
| 2.1.3 LCD | 4 |
| 2.2 Electrical Requirements | 4 |
| 2.2.1 Electrical Power | 4 |
| 2.3 Software Requirements | 4 |
| 2.3.1 STM32F411 | 4 |
| 2.4 Game Requirements | 4 |
| 2.4.1 Must Resolve All Inputs Correctly | 4 |
| 2.4.2 Game Modes | 5 |
| 2.4.3 Timer | 5 |
| 2.4.4 Win/Loss Record | 5 |
| 2.4.5 Victory/Defeat Screen | 5 |
| 3 Hardware Design | 6 |
| 3.1 Mechanical Design | 6 |
| 3.2 Electrical System Design | 6 |
| 3.3 Electronic System Design | 7 |

| | |
|--|-----------|
| 4 Software Design | 9 |
| 4.1 Introduction | 9 |
| 5 Testing Requirements | 15 |
| 5.1 Hardware Testing Requirements | 15 |
| 5.1.1 The device shall recover from power loss | 15 |
| 5.1.2 Nucleo | 15 |
| 5.1.3 Keypad x2 | 15 |
| 5.1.4 LCD x2 | 15 |
| 5.2 Electrical Testing Requirements | 15 |
| 5.2.1 Power Supply | 15 |
| 5.3 Software Testing Requirements | 15 |
| 5.3.1 STM32CubeIDE | 15 |
| 5.4 Game Testing Requirements | 15 |
| 5.4.1 Must Resolve All Inputs Correctly | 15 |
| 5.4.2 Game Modes | 16 |
| 5.4.3 Timer | 16 |
| 5.4.4 Win/Loss Record | 16 |
| 5.4.5 Victory/Defeat Screen | 16 |
| 6 Users Guide | 17 |
| 6.1 Introduction | 17 |
| 6.2 Single-Player Mode | 17 |
| 6.3 Two-Player Mode | 17 |
| 6.4 Controls | 18 |
| 6.4.1 Welcome Screen Controls | 18 |
| 6.4.2 In-Game Controls | 18 |
| 7 Lessons Learned | 19 |
| A Source Code | 20 |
| B Bill of Materials | 43 |
| B.1 Cost breakdown | 43 |
| C Rating | 44 |

List of Figures

| | | |
|-----|---------------------------|----|
| 3.1 | Hardware setup | 6 |
| 3.2 | Power Overview | 7 |
| 3.3 | Nucleo Pinout | 8 |
| 4.1 | Small Flowchart | 10 |
| 4.2 | Large Flowchart | 14 |

Listings

| | | |
|-----|---|----|
| 4.1 | Libraries included | 9 |
| 4.2 | Possible Messages | 9 |
| 4.3 | Clear Screen Command | 10 |
| 4.4 | Keypad interrupt code | 11 |
| 4.5 | Welcome Code | 11 |
| 4.6 | Beginning of Player 1 Countdown | 12 |
| 4.7 | Example of logic for determining winner | 12 |
| 4.8 | Assigning value from player 2 choice | 13 |
| 7.1 | Seed set with 'tim' | 19 |
| 7.2 | Random number chosen | 19 |
| A.1 | Source Code | 20 |

Abstract

In this project, we will be building a twist on the game of Rock Paper Scissors that adds two more options for an extra level of complexity. The goal of this project is to correctly simulate the game. In this game, two players choose an option for their turn. The five options are Lizard, Spock, Rock, Paper, and Scissors. Each option beats two other options. Scissors cuts Paper. Paper covers Rock. Rock crushes Lizard. Lizard poisons Spock. Spock smashes Scissors. Scissors decapitates Lizard. Lizard eats Paper. Paper disproves Spock. Spock vaporizes Rock. Rock crushes Scissors. There will be a timer displayed that gives each player five seconds to choose their option. Once a player chooses an option, the timer stops. If a player has not chosen an option after five seconds, their turn will be forfeited. The first to win three rounds will be the winner of the match. During the match, the win/loss record will be displayed on the LCD. When a player starts the program, they have the option to choose a multiplayer mode with another player, or a single-player mode where they play against an AI. The AI randomly selects one of the five options, and the outcome is determined the same way as in the multiplayer mode.

Nomenclature

| | |
|--------------|--------------------------|
| ECO | Engineering Change Order |
| Nucleo | NUCLEO F411RE |
| AI | Artificial Intelligence |
| LCD | Liquid-Crystal Display |

Chapter 1

Introduction

1.1 Introduction

Rock Paper Scissors is a classic game where two players use hand signals to make their selection based on what they think the opposition will choose. In the classic game, there are only three options to choose from. However, in this iteration of the game, we are increasing complexity by adding Lizard and Spock as options. Rock Paper Scissors Lizard Spock will allow the user to choose from five options, and each one of those options can beat two of the other options. This will allow for dynamic game play and deeper thought into making a selection.

1.2 Objective

Simulate a game of Rock Paper Scissors Lizard Spock. Each player has 5 seconds to choose an option, and after the timer runs out, it displays if you lose or win. The game is a best of 5. There will also be a single player option where you can play against an AI.

1.3 Customer

This game can be played by everyone over the age of 3.

1.4 Design Team

1.4.1 Engineering Co-Manager/Engineer/Programming Specialist

Noah Simpson is a computer engineering student at the University of Tulsa.

1.4.2 Engineering Co-Manager/Engineer/LCD Specialist

Will Ferguson is a computer engineering student at the University of Tulsa.

1.4.3 Engineering Co-Manager/Programmer/Keypad Specialist

Zade Mahayni is a computer engineering student at the University of Tulsa.

Chapter 2

Project Requirements

2.1 Hardware Requirements

2.1.1 NUCLEO F411RE

We shall store and run the game on the NUCLEO F411RE which is provided by The University of Tulsa. This board shall have the program uploaded to it, and the remaining hardware shall be attached to the board.

2.1.2 Keypad

The game shall use one or two keypads, but no more. The keypads will be used to make a selection. The keypads that shall be used are the 96BB2-056-R from Digi-Key Electronics. The single-player option shall require one keypad and the two-player option shall require two keypads, one for each player. The keypads have eight pins each, and each pin shall be connected to a specific port on the Nucleo.

2.1.3 LCD

The game shall use two LCD screens. The two LCDs that shall be used are the SparkFun 20x4 SerLCD. Each player will have a personal LCD screen that shall display the win/loss record, the countdown timer, and what their selection for the current round is.

2.2 Electrical Requirements

2.2.1 Electrical Power

The project shall run on 5 volts provided by the USB cable plugged into the Nucleo. That cable will be plugged into a computer or a wall outlet to deliver the power.

2.3 Software Requirements

2.3.1 STM32F411

The project shall be created using the STM32CubeIDE compiler.

2.4 Game Requirements

2.4.1 Must Resolve All Inputs Correctly

The game shall adhere to the rules of Rock Paper Scissors Lizard Spock.

- Scissors beats Paper and Lizard
- Paper beats Rock and Spock
- Rock beats Scissors and Lizard
- Lizard beats Spock and Paper
- Spock beats Rock and Scissors

2.4.2 Game Modes

- Single Player
 - The user plays against an AI that will randomly select an option.
- Multiplayer
 - The user plays against another user.

2.4.3 Timer

There shall be a five-second timer on the LCDs that terminates when a user selects an option. If no option is selected when the timer reaches zero, the user's turn will automatically be forfeited.

2.4.4 Win/Loss Record

There will be a win/loss record that is displayed on each LCD. In the event of a tie, the current round will be replayed. Once the winner of a round is determined, the record will update and will not include ties.

2.4.5 Victory/Defeat Screen

Once a player wins three times, the winner's LCD will display a victory screen, and the loser's LCD will display a defeat screen.

Chapter 3

Hardware Design

3.1 Mechanical Design

Both keypads are connected to the Nucleo. Both LCDs are connected to the Nucleo and share the same power which is connected to a breadboard from the Nucleo. See Figure 3.1 for the setup.

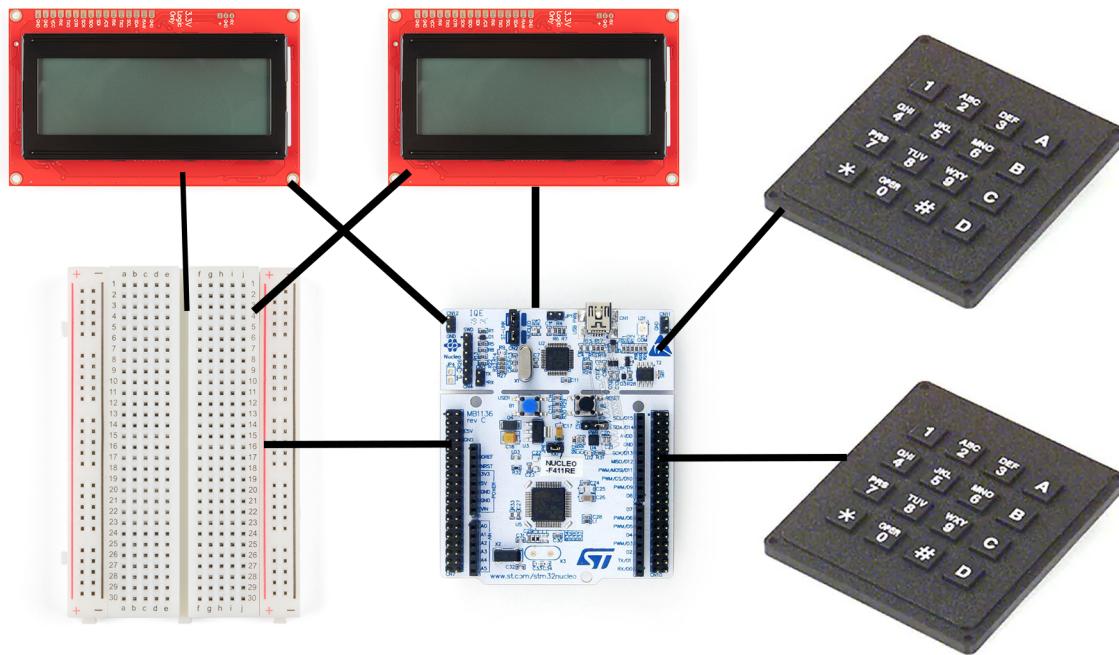


Figure 3.1: Hardware setup

3.2 Electrical System Design

The Nucleo receives power via USB from a source such as a laptop or power brick. The LCDs get 5V of power from the Nucleo by connecting to the breadboard.

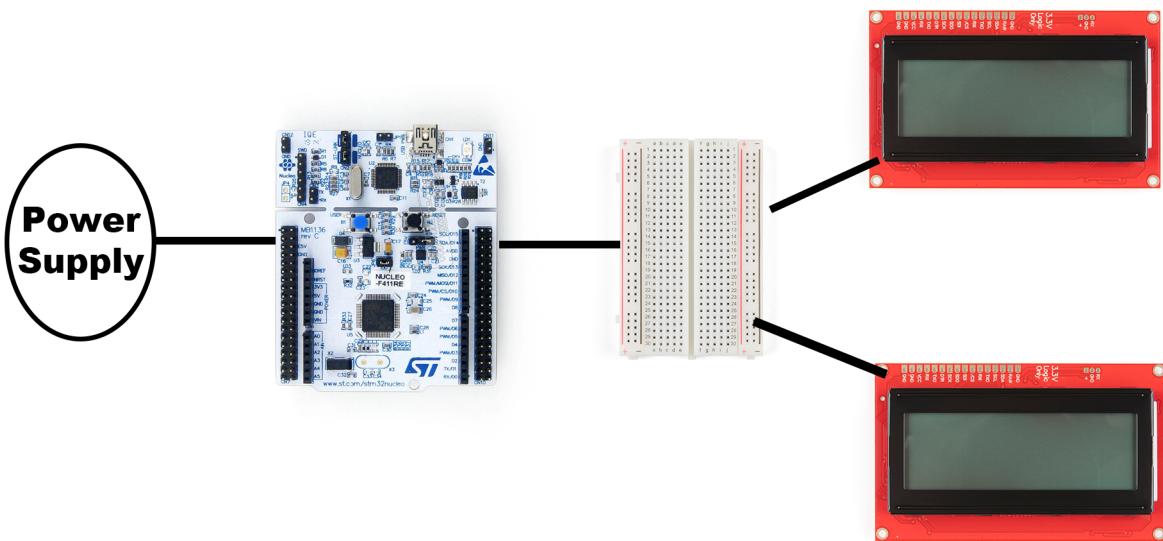


Figure 3.2: Power Overview

3.3 Electronic System Design

The LCDs have three wires that need to be connected: a power wire, a grounded wire, and a RX wire. The RX wire is connected to the TX port on the Nucleo. The keypads have eight pins each, with four of them being connected to the rows, and the other four pins being connected to the columns. These are connected to the Nucleo as shown in Figure 3.3 below.

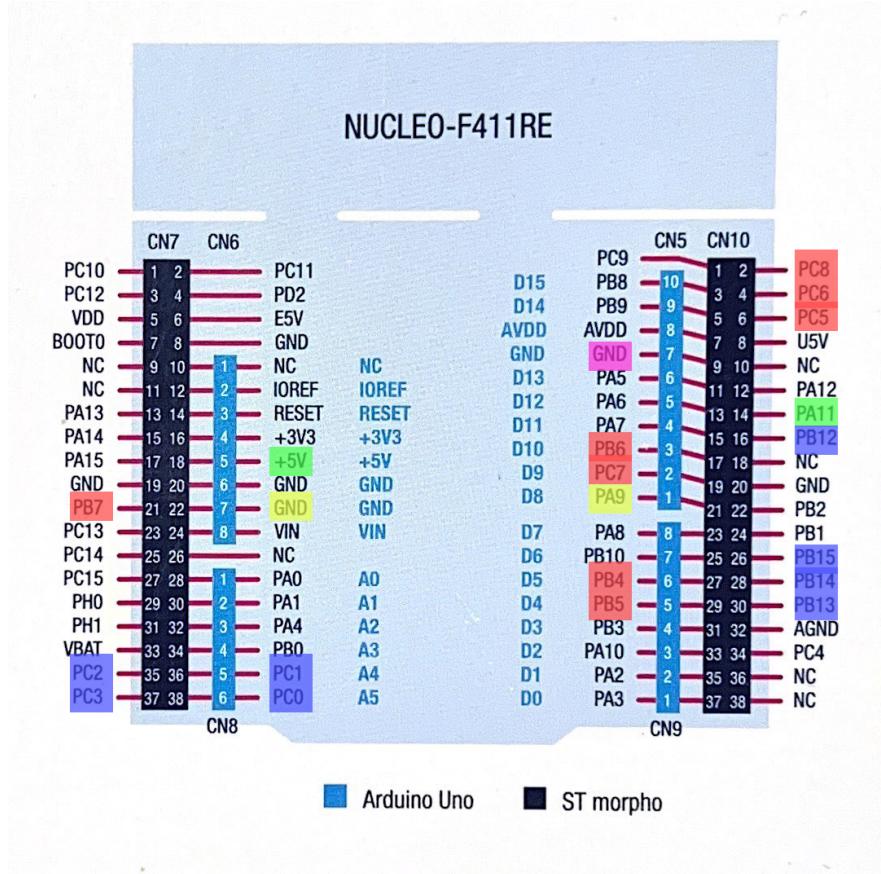


Figure 3.3: Nucleo Pinout

- **Yellow** - LCD 1
- **Pink** - LCD 2
- **Red** - Keypad 1
- **Blue** - Keypad 2
- **Green** - Breadboard

Chapter 4

Software Design

4.1 Introduction

We included several libraries in the main.c file as shown below.

```

1 #include "main.h"
2 #include "string.h"
3 #include "stdlib.h"
4 #include "stdbool.h"
5 #include "stdio.h"
6 #include "memory.h"
7 #include "stdint.h"
8 #include "stm32f4xx_hal_rcc.h"
9 #include "stm32f4xx_hal.h"
10 #include "stm32f4xx_hal_exti.h"
11 #include "stm32f4xx_hal_flash_ex.h"
12 #include "stm32f4xx_hal_flash.h"
13 #include "stm32f4xx_hal_gpio.h"

```

Listing 4.1: Libraries included

We also included the variables for the keypad in the main.c file. Since we were going to display many messages throughout the program, we created a variable for each message that would be displayed.

| | | | |
|----|--|------------------|------------------------------------|
| 1 | char welcome_player_one[]=" | Player One | How many players? (1 or 2); |
| 2 | char welcome_player_two[]=" | Player Two | Wait for player one.; |
| 3 | char player_one_wait_5 []="Game starts in 5 | Rock(1) Paper(2) | Scissors(3) Lizard(4) Spock(5); |
| 4 | char player_one_wait_4 []="Game starts in 4 | Rock(1) Paper(2) | Scissors(3) Lizard(4) Spock(5); |
| 5 | char player_one_wait_3 []="Game starts in 3 | Rock(1) Paper(2) | Scissors(3) Lizard(4) Spock(5); |
| 6 | char player_one_wait_2 []="Game starts in 2 | Rock(1) Paper(2) | Scissors(3) Lizard(4) Spock(5); |
| 7 | char player_one_wait_1 []="Game starts in 1 | Rock(1) Paper(2) | Scissors(3) Lizard(4) Spock(5); |
| 8 | char player_one_choice_5 []="Make your choice! 5 | Rock(1) Paper(2) | Scissors(3) Lizard(4) Spock(5); |
| 9 | char player_one_choice_4 []="Make your choice! 4 | Rock(1) Paper(2) | Scissors(3) Lizard(4) Spock(5); |
| 10 | char player_one_choice_3 []="Make your choice! 3 | Rock(1) Paper(2) | Scissors(3) Lizard(4) Spock(5); |
| 11 | char player_one_choice_2 []="Make your choice! 2 | Rock(1) Paper(2) | Scissors(3) Lizard(4) Spock(5); |
| 12 | char player_one_choice_1 []="Make your choice! 1 | Rock(1) Paper(2) | Scissors(3) Lizard(4) Spock(5); |
| 13 | char please_wait_1 []="Please wait for | player 2"; | |
| 14 | char please_wait_2 []="Player 1 is | selecting"; | |
| 15 | char buffer []=" | Please Wait"; | |

```

16 char player1_choice_message []="Player 1 chose:      ";
17 char player_one_choice_final []="                      ";
18 char player2_choice_message []="Player 2 chose:      ";
19 char player_two_choice_final []="                      ";
20 char teststatement []="hi mom";
21 char winstatement []="          You Win!          ";
22 char loststatement []="        DEFEATED! XD      ";
23 char score[32] = "          0--0";

```

Listing 4.2: Possible Messages

We also created a function for each LCD that clears its screen as shown below.

```

1 void clearScreen1(void){
2     uint8_t clearscreen1[] = {0x7C, 0x2D};
3     HAL_UART_Transmit(&huart1, clearscreen1, sizeof(clearscreen1), HAL_MAX_DELAY);
4 }
5 void clearScreen2(void){
6     uint8_t clearscreen2[] = {0x7C, 0x2D};
7     HAL_UART_Transmit(&huart6, clearscreen2, sizeof(clearscreen2), HAL_MAX_DELAY);
8 }

```

Listing 4.3: Clear Screen Command

Figure 4.1 is the flowchart that gives a simple description of the process that the program runs through.

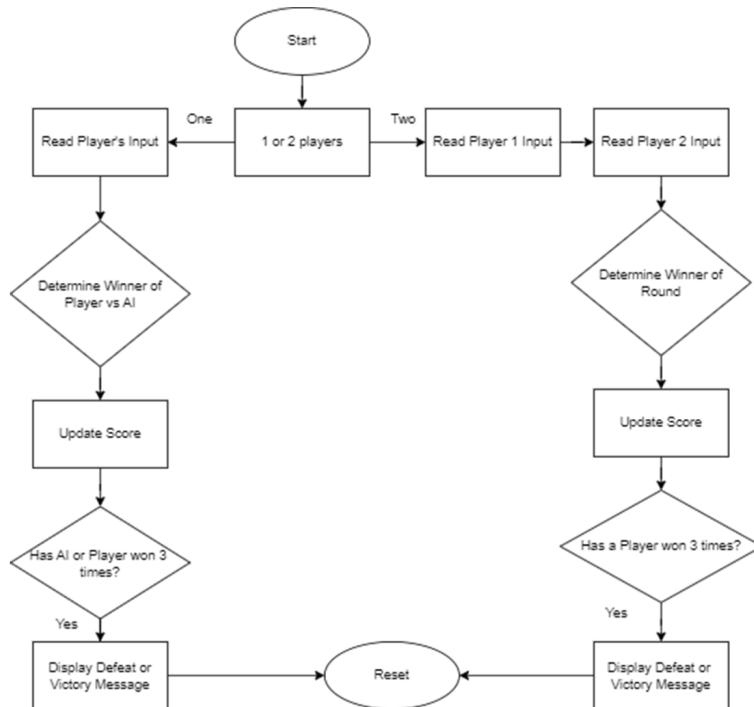


Figure 4.1: Small Flowchart

To make the keypad functional, we used the code from Dr. Hutchins' GitHub. We copied this into our interrupt file. We then created a second version of each variable. We then copied the original function from the first keypad, and changed all the variables to the second versions in order to make the second keypad functional without interrupting the first keypad.

```

1 void EXTI9_5_IRQHandler(void)
2 {
3     /* USER CODE BEGIN EXTI9_5_IRQHandler */
4     key = 0;
5     for(uint8_t c=0 ; c<4 ; c++){
6         for(uint8_t i=0 ; i<4 ; i++){
7             HAL_GPIO_WritePin((GPIO_TypeDef*)_KEYPAD_COLUMN_GPIO_PORT[i],
8             _KEYPAD_COLUMN_GPIO_PIN[i], GPIO_PIN_SET);
9         }
10        HAL_GPIO_WritePin((GPIO_TypeDef*)_KEYPAD_COLUMN_GPIO_PORT[c],
11        _KEYPAD_COLUMN_GPIO_PIN[c], GPIO_PIN_RESET);
12        for(uint8_t r=0 ; r<4 ; r++){
13            if(HAL_GPIO_ReadPin((GPIO_TypeDef*)_KEYPAD_ROW_GPIO_PORT[r],
14            _KEYPAD_ROW_GPIO_PIN[r]) == GPIO_PIN_RESET){
15                if(HAL_GPIO_ReadPin((GPIO_TypeDef*)_KEYPAD_ROW_GPIO_PORT[r],
16                _KEYPAD_ROW_GPIO_PIN[r]) == GPIO_PIN_RESET){
17                    key |= 1<<c;
18                    key |= 1<<(r+8);
19                    while(HAL_GPIO_ReadPin((GPIO_TypeDef*)_KEYPAD_ROW_GPIO_PORT[r],
20                    _KEYPAD_ROW_GPIO_PIN[r]) == GPIO_PIN_RESET){
21
22
23
24            keyChar = KeyPadGetChar(key);
25            key = 0;
26
27            HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7, GPIO_PIN_RESET);

```

Listing 4.4: Keypad interrupt code

The main block of code consists of many if-statements. It begins by displaying a welcome message and waiting for the player's decision on which mode they want to play. If the player selects the single player mode, the program jumps to the single player portion. In this block, the program reads the player's input and then goes to a function that generates a random number. This random number is read as the AI's "input." The program then jumps to a series of if-statements that decide who won the round. After the choices and scores are displayed, it goes to the next round and repeats the process. After either the player or the AI has won three rounds, a victory or defeat screen is displayed depending on who has reached 3 victories first. Then it goes back to the welcome screen.

```

1 welcome:
2     clearScreen1();
3     clearScreen2();
4
5     HAL_UART_Transmit(&huart1, welcome_player_one, sizeof(welcome_player_one)-1,
6     HAL_MAX_DELAY);
7     HAL_UART_Transmit(&huart6, welcome_player_two, sizeof(welcome_player_two)-1,
8     HAL_MAX_DELAY);
9     player1_score=0;
10    player2_score=0;
11    start:
12    if (keyChar == 0){
13        tim++;
14        goto start;
15    }
16    else if (keyChar == '1'){
17        clearScreen1();
18        keyChar=0;

```

```

17         goto one_player;
18     }
19     else if (keyChar == '2'){
20         clearScreen1();
21         clearScreen2();
22         keyChar=0;
23         keyChar2=0;
24         goto two_player;
25     }
26     else if (keyChar != 0){
27         keyChar=0;
28         goto start;
29     }

```

Listing 4.5: Welcome Code

```

1 if(player1_score < 3 && player2_score < 3){
2     HAL_UART_Transmit(&huart1, player_one_wait_5, sizeof(player_one_wait_5)-1,
3     HAL_MAX_DELAY);
4     HAL_Delay(1000);
5     clearScreen1();
6     HAL_UART_Transmit(&huart1, player_one_wait_4, sizeof(player_one_wait_4)-1,
7     HAL_MAX_DELAY);
8     HAL_Delay(1000);
9     clearScreen1();

```

Listing 4.6: Beginning of Player 1 Countdown

The code below shows a series of if statements nested within a single if statement. If player 1's keyChar is equal to 1, then that means that they have selected 'rock' as their choice. The nested if statements are the possible choices for the AI, and the winner's score will be updated based on if the AI's choice defeats 'rock' or not.

```

1 clearScreen1();
2     if (keyChar == '1'){
3         if (rand_number == 1){
4             sprintf(score, "%d--%d", player1_score,
5                     player2_score);
6             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
7             HAL_Delay(3000);
8             keyChar = 0;
9         }
10        else if (rand_number ==2){
11            player2_score++;
12            sprintf(score, "%d--%d", player1_score,
13                    player2_score);
14            HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
15            HAL_Delay(3000);
16            keyChar = 0;
17        }
18        else if (rand_number ==3){
19            player1_score++;
20            sprintf(score, "%d--%d", player1_score
21                     , player2_score);
22            HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
23            HAL_Delay(3000);
24            keyChar = 0;
25        }
26        else if (rand_number ==4){
27            player1_score++;
28            sprintf(score, "%d--%d", player1_score
29                     , player2_score);
30            HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
31            HAL_Delay(3000);
            keyChar = 0;
        }
        else if (rand_number ==5){
            player2_score++;
        }

```

```

32     , player2_score);
33         sprintf(score, "%d--%d", player1_score
34             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
35             HAL_Delay(3000);
36             keyChar = 0;
37     }
}

```

Listing 4.7: Example of logic for determining winner

If the player chooses the two player mode, the program jumps to reading the input of the first player. It then jumps to a block of code that reads the second player's input. This input from player 2 is defined as keyChar2. After both inputs have been submitted, it jumps to a block of code that decides who the winner is. It then displayed the choices of both players and then the updated score. Once a player has won three times, a defeat and victory screen are displayed.

```

1 if (keyChar2 == '1'){
2     sprintf(player_two_choice_final, "Rock");
3 }
4 else if (keyChar2 == '2'){
5     sprintf(player_two_choice_final, "Paper");
6 }
7 else if (keyChar2 == '3'){
8     sprintf(player_two_choice_final, "Scissors");
9 }
10 else if (keyChar2 == '4'){
11     sprintf(player_two_choice_final, "Lizard");
12 }
13 else if (keyChar2 == '5'){
14     sprintf(player_two_choice_final, "Spock");
15 }
16 else{
17     keyChar2=0;
18     sprintf(player_two_choice_final, "no_input_2 = 1");
19 }
}

```

Listing 4.8: Assigning value from player 2 choice

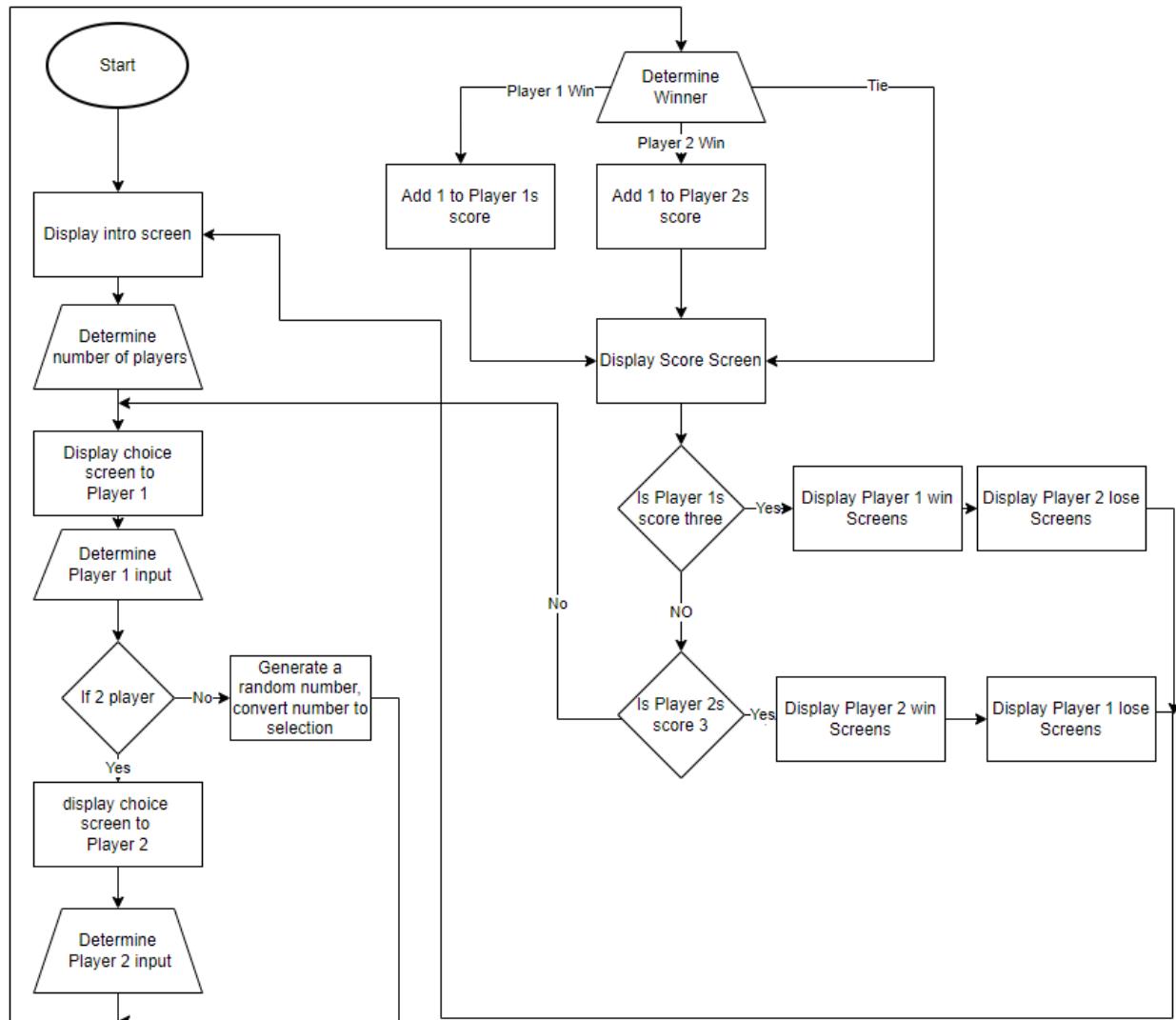


Figure 4.2: Large Flowchart

Chapter 5

Testing Requirements

5.1 Hardware Testing Requirements

5.1.1 The device shall recover from power loss

When the device is unplugged from a power source and then plugged back in, the game starts from the beginning.

5.1.2 Nucleo

This requirement was fulfilled by building the code onto our Nucleo and running it.

5.1.3 Keypad x2

This requirement was fulfilled by connecting the keypads to the Nucleo as shown in the Hardware Design section.

5.1.4 LCD x2

This requirement was fulfilled by connecting the LCD's to the Nucleo as shown earlier in the Hardware Design section.

5.2 Electrical Testing Requirements

5.2.1 Power Supply

This requirement was fulfilled by connecting the Nucleo to a laptop.

5.3 Software Testing Requirements

5.3.1 STM32CubeIDE

This requirement was fulfilled by using the STM32CubeIDE compiler to create our program.

5.4 Game Testing Requirements

5.4.1 Must Resolve All Inputs Correctly

This requirement was fulfilled by using a block of if-statements that determines the winner of a round depending on the inputs of the players. See Listing 4.8 for an example.

5.4.2 Game Modes

This requirement was fulfilled by prompting the player to decide which game mode to play on the welcome screen, and then going to the appropriate portion of code depending on what the user has chosen.

5.4.3 Timer

This requirement was fulfilled by displaying a 5 second timer on the top right of the LCD that counts down.

5.4.4 Win/Loss Record

This requirement was fulfilled by displaying the score at the end of each round.

5.4.5 Victory/Defeat Screen

This requirement was fulfilled by displaying a victory and defeat message once a player won three rounds.

Chapter 6

Users Guide

6.1 Introduction

This is a turn-based game of Rock Paper Scissors Lizard Spock. The first player chooses whether to play against on their own or against a second player. In both modes, the players have 5 seconds to choose an option. If no option is chosen after 5 seconds, it will count as forfeiting a round. However, if both players choose no option, the round will not be counted. The match ends when a player has won three rounds, and they will be shown a victory or defeat message depending on their result. The score is displayed after every round. Ties have no effect on the score.

6.2 Single-Player Mode

If the single player mode is chosen, the user will be put against an AI that randomly chooses an option every round.

6.3 Two-Player Mode

If the user chooses the two player mode, the game will be played in a turn-based process. The first player has five seconds to choose their option. Once an option has been chosen, the second player chooses their option. The options selected will not be displayed until both turns have passed. Make sure to hide your keypad while selecting so the second player does not cheat! It then updates the score depending on who won, and will repeat this process until someone has won three games.

6.4 Controls

6.4.1 Welcome Screen Controls

- 1 - Single-Player Mode
- 2 - Two-Player Mode

6.4.2 In-Game Controls

Pressing this number on the keypad when prompted on the screen will make that your choice.

- 1 - Rock
- 2 - Paper
- 3 - Scissors
- 4 - Lizard
- 5 - Spock

Chapter 7

Lessons Learned

In this project we learned several things such as how to use multiple devices with the Nucleo board as well as how to create a random number generator. The random number generator was done by having a variable named 'tim' that increases rapidly at the beginning of the game until the user selects an option. Once the option is selected, a seed is set by using the srand function. Then, a variable called 'rand_number' is set with the rand function. After each round of gameplay, the variable 'tim' is increased by one to change the seed so that the same number is not being constantly chosen.

```

1 if (keyChar == 0){
2     tim++;
3     goto start;
4 }
```

Listing 7.1: Seed set with 'tim'

```

1 results:
2     tim++;
3     srand(tim);
4     rand_number = (rand() % (upper-lower+1))+lower;
```

Listing 7.2: Random number chosen

Appendix A

Source Code

```

1  /* USER CODE BEGIN Header */
2  /**
3   * @file          : main.c
4   * @brief         : Main program body
5   * @attention
6   *
7   * Copyright (c) 2022 STMicroelectronics.
8   * All rights reserved.
9   *
10  * This software is licensed under terms that can be found in the LICENSE file
11  * in the root directory of this software component.
12  * If no LICENSE file comes with this software, it is provided AS-IS.
13  *
14  */
15  ****
16  ****
17  */
18  /* USER CODE END Header */
19  /* Includes -----*/
20  #include "main.h"
21
22 /* Private includes -----*/
23 /* USER CODE BEGIN Includes */
24 #include "string.h"
25 #include "stdlib.h"
26 #include "stdbool.h"
27 #include "stdio.h"
28 #include "memory.h"
29 #include "stdint.h"
30 #include "stm32f4xx_hal_rcc.h"
31 #include "stm32f4xx_hal.h"
32 #include "stm32f4xx_hal_exti.h"
33 #include "stm32f4xx_hal_flash_ex.h"
34 #include "stm32f4xx_hal_flash.h"
35 #include "stm32f4xx_hal_gpio.h"
36 /* USER CODE END Includes */
37
38 /* Private typedef -----*/
39 /* USER CODE BEGIN PTD */
40
41 /* USER CODE END PTD */
42
43 /* Private define -----*/
44 /* USER CODE BEGIN PD */
45 /* USER CODE END PD */
46
47 /* Private macro -----*/
48 /* USER CODE BEGIN PM */
49

```

```

50 /* USER CODE END PM */
51
52 /* Private variables -----*/
53 UART_HandleTypeDef huart1;
54 UART_HandleTypeDef huart2;
55 UART_HandleTypeDef huart6;
56
57 /* USER CODE BEGIN PV */
58 uint16_t key = 0;
59 char keyChar = 0;
60 uint16_t key2 = 0;
61 char keyChar2 = 0;
62 /* USER CODE END PV */
63
64 /* Private function prototypes -----*/
65 void SystemClock_Config(void);
66 static void MX_GPIO_Init(void);
67 static void MX_USART2_UART_Init(void);
68 static void MX_USART1_UART_Init(void);
69 static void MX_USART6_UART_Init(void);
70 /* USER CODE BEGIN PFP */
71
72 /* USER CODE END PFP */
73
74 /* Private user code -----*/
75 /* USER CODE BEGIN 0 */
76
77 /* USER CODE END 0 */
78
79 /**
80 * @brief The application entry point.
81 * @retval int
82 */
83 int main(void)
84 {
85     /* USER CODE BEGIN 1 */
86
87 //goto begin;
88     /* USER CODE END 1 */
89
90     /* MCU Configuration-----*/
91
92     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
93     HAL_Init();
94
95     /* USER CODE BEGIN Init */
96
97     /* USER CODE END Init */
98
99     /* Configure the system clock */
100    SystemClock_Config();
101
102    /* USER CODE BEGIN SysInit */
103
104    /* USER CODE END SysInit */
105
106    /* Initialize all configured peripherals */
107    MX_GPIO_Init();
108    MX_USART2_UART_Init();
109    MX_USART1_UART_Init();
110    MX_USART6_UART_Init();
111    /* USER CODE BEGIN 2 */
112 //    char FirstNumber[20];
113 //    char SecondNumber[20];
114 //    char tmpFirstNumber[20];
115 //    char tmpSecondNumber[20];
116 //    char limit[32];
117    void clearScreen1(void);

```

```

118 void clearScreen2(void);
119 char sendMessage[32];
120
121 // char notvalid[32];
122 // for(int i = 0; i < 20; i++){
123 //     FirstNumber[i] = '\0';
124 //     SecondNumber[i] = '\0';
125 //     tmpFirstNumber[i] = '\0';
126 //     tmpSecondNumber[i] = '\0';
127 //}
128 // clearScreen();
129 //
130 // HAL_UART_Transmit(&huart1, (unsigned char *)"Enter number 1:\r\n", 17, HAL_MAX_DELAY);
131 begin:
132 clearScreen1();
133
134 char welcome_player_one[]="          Player One           How many players?   (1
135 or 2)";
136 char welcome_player_two[]="          Player Two           Wait for player one.";
137 char player_one_wait_5[]="Game starts in 5      Rock(1) Paper(2)    Scissors(3)
138 Lizard(4) Spock(5)";
139 char player_one_wait_4[]="Game starts in 4      Rock(1) Paper(2)    Scissors(3)
140 Lizard(4) Spock(5)";
141 char player_one_wait_3[]="Game starts in 3      Rock(1) Paper(2)    Scissors(3)
142 Lizard(4) Spock(5)";
143 char player_one_wait_2[]="Game starts in 2      Rock(1) Paper(2)    Scissors(3)
144 Lizard(4) Spock(5)";
145 char player_one_wait_1[]="Game starts in 1      Rock(1) Paper(2)    Scissors(3)
146 Lizard(4) Spock(5)";
147 char player_one_choice_5[]="Make your choice!  5Rock(1) Paper(2)    Scissors(3)
148 Lizard(4) Spock(5)";
149 char player_one_choice_4[]="Make your choice!  4Rock(1) Paper(2)    Scissors(3)
150 Lizard(4) Spock(5)";
151 char player_one_choice_3[]="Make your choice!  3Rock(1) Paper(2)    Scissors(3)
152 Lizard(4) Spock(5)";
153 char player_one_choice_2[]="Make your choice!  2Rock(1) Paper(2)    Scissors(3)
154 Lizard(4) Spock(5)";
155 char player_one_choice_1[]="Make your choice!  1Rock(1) Paper(2)    Scissors(3)
156 Lizard(4) Spock(5)";
157 char please_wait_1[]="Please wait for      player 2";
158 char please_wait_2[]="Player 1 is           selecting";
159 char buffer[]="      Please Wait";
160
161 char player1_choice_message[]="Player 1 chose:      ";
162 char player_one_choice_final[]="";
163 char player2_choice_message[]="Player 2 chose:      ";
164 char player_two_choice_final[]="";
165
166 char teststatement[]="hi mom";
167 char winstatement[]="          You Win!        ";
168 char loststatement[]="          DEFEATED! XD    ";
169 char score[32]="          0--0";
170
171 int game_counter = 0;
172 int no_input_1 = 0;
173 int no_input_2 = 0;
174 int rand_number;
175 int lower = 1;
176 int upper = 5;
177 int player1_score=0;
178 int player2_score=0;
179 int tim=0;
180
181 //goto welcome;
182 /* USER CODE END 2 */
183
184 /* Infinite loop */
185 /* USER CODE BEGIN WHILE */
186 while (1)
187 {
188
189     HAL_Delay(3000);

```

```

175
176     welcome:
177     clearScreen1();
178     clearScreen2();
179
180     HAL_UART_Transmit(&huart1, welcome_player_one, sizeof(welcome_player_one)-1,
181     HAL_MAX_DELAY);
181     HAL_UART_Transmit(&huart6, welcome_player_two, sizeof(welcome_player_two)-1,
182     HAL_MAX_DELAY);
182     player1_score=0;
183     player2_score=0;
184     start:
185     if (keyChar == 0){
186         tim++;
187         goto start;
188     }
189     else if (keyChar == '1'){
190         clearScreen1();
191         keyChar=0;
192         goto one_player;
193     }
194     else if (keyChar == '2'){
195         clearScreen1();
196         clearScreen2();
197         keyChar=0;
198         keyChar2=0;
199         goto two_player;
200     }
201     else if (keyChar != 0){
202         keyChar=0;
203         goto start;
204     }
205
206
207
208
209     one_player:
210
211     if(player1_score < 3 && player2_score < 3){
212         HAL_UART_Transmit(&huart1, player_one_wait_5, sizeof(player_one_wait_5)-1,
213         HAL_MAX_DELAY);
213         HAL_Delay(1000);
214         clearScreen1();
215         HAL_UART_Transmit(&huart1, player_one_wait_4, sizeof(player_one_wait_4)-1,
216         HAL_MAX_DELAY);
216         HAL_Delay(1000);
217         clearScreen1();
218         HAL_UART_Transmit(&huart1, player_one_wait_3, sizeof(player_one_wait_3)-1,
219         HAL_MAX_DELAY);
219         HAL_Delay(1000);
220         clearScreen1();
221         HAL_UART_Transmit(&huart1, player_one_wait_2, sizeof(player_one_wait_2)-1,
222         HAL_MAX_DELAY);
222         HAL_Delay(1000);
223         clearScreen1();
224         HAL_UART_Transmit(&huart1, player_one_wait_1, sizeof(player_one_wait_1)-1,
225         HAL_MAX_DELAY);
225         HAL_Delay(1000);
226         clearScreen1();
227
228
229         HAL_UART_Transmit(&huart1, player_one_choice_5, sizeof(player_one_choice_5)-1,
230         HAL_MAX_DELAY);
230         HAL_Delay(1000);
231         clearScreen1();
232         if(keyChar !=0){
233             goto pleasewait1;
234         }

```

```

235     HAL_UART_Transmit(&huart1, player_one_choice_4, sizeof(player_one_choice_4)-1,
236     HAL_MAX_DELAY);
237     HAL_Delay(1000);
238     clearScreen1();
239     if(keyChar !=0){
240         goto pleasewait1;
241     }
242     HAL_UART_Transmit(&huart1, player_one_choice_3, sizeof(player_one_choice_3)-1,
243     HAL_MAX_DELAY);
244     HAL_Delay(1000);
245     clearScreen1();
246     if(keyChar !=0){
247         goto pleasewait1;
248     }
249     HAL_UART_Transmit(&huart1, player_one_choice_2, sizeof(player_one_choice_2)-1,
250     HAL_MAX_DELAY);
251     HAL_Delay(1000);
252     clearScreen1();
253     if(keyChar !=0){
254         goto pleasewait1;
255     }
256     HAL_UART_Transmit(&huart1, player_one_choice_1, sizeof(player_one_choice_1)-1,
257     HAL_MAX_DELAY);
258     HAL_Delay(1000);
259     clearScreen1();
260     goto pleasewait1;
261     clearScreen1();
262     HAL_UART_Transmit(&huart1, please_wait_1, sizeof(please_wait_1)-1, HAL_MAX_DELAY);
263     if (keyChar == '1'){
264         sprintf(player_one_choice_final, "Rock");
265     }
266     else if (keyChar == '2'){
267         sprintf(player_one_choice_final, "Paper");
268     }
269     else if (keyChar == '3'){
270         sprintf(player_one_choice_final, "Scissors");
271     }
272     else if (keyChar == '4'){
273         sprintf(player_one_choice_final, "Lizard");
274     }
275     else if (keyChar == '5'){
276         sprintf(player_one_choice_final, "Spock");
277     }
278     else{
279         keyChar=0;
280         sprintf(player_one_choice_final, "");
281         no_input_1 = 1;
282     }
283     HAL_Delay(5000);
284     clearScreen1();
285     HAL_Delay(100);
286     goto results;
287
288     results:
289     tim++;
290     srand(tim);
291     rand_number = (rand() % (upper-lower+1))+lower;
292 //rand_number = 3;
293
294     if (rand_number == 1){
295         sprintf(player_two_choice_final, "Rock");
296     }
297     else if (rand_number == 2){
298         sprintf(player_two_choice_final, "Paper");

```

```

299     }
300     else if (rand_number == 3){
301         sprintf(player_two_choice_final, "Scissors" );
302     }
303     else if (rand_number == 4){
304         sprintf(player_two_choice_final, "Lizard" );
305     }
306     else if (rand_number == 5){
307         sprintf(player_two_choice_final, "Spock" );
308     }
309     else{
310
311         no_input_2 = 1;
312     }
313
314
315     HAL_UART_Transmit(&huart1, player1_choice_message, sizeof(player1_choice_message)-1,
316     HAL_MAX_DELAY);
316     HAL_UART_Transmit(&huart1, player_one_choice_final, sizeof(player_one_choice_final)-1,
317     HAL_MAX_DELAY);
317     HAL_UART_Transmit(&huart1, player2_choice_message, sizeof(player2_choice_message)-1,
318     HAL_MAX_DELAY);
318     HAL_UART_Transmit(&huart1, player_two_choice_final, sizeof(player_two_choice_final)
319     )-1, HAL_MAX_DELAY);
320
321     HAL_Delay(5000);
322
323
324 clearScreen1();
325     if (keyChar == '1'){
326         if (rand_number == 1){
327             sprintf(score, "%d--%d", player1_score,
328             player2_score);
329             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
330             HAL_Delay(3000);
331             keyChar = 0;
332         }
333         else if (rand_number ==2){
334             player2_score++;
335             sprintf(score, "%d--%d", player1_score,
336             player2_score);
337             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
338             HAL_Delay(3000);
339             keyChar = 0;
340         }
341         else if (rand_number ==3){
342             player1_score++;
343             sprintf(score, "%d--%d", player1_score
344             , player2_score);
345             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
346             HAL_Delay(3000);
347             keyChar = 0;
348         }
349         else if (rand_number ==4){
350             player1_score++;
351             sprintf(score, "%d--%d", player1_score
352             , player2_score);
353             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
354             HAL_Delay(3000);
355             keyChar = 0;
356         }
357         else if (rand_number ==5){
358             player2_score++;
359             sprintf(score, "%d--%d", player1_score
360             , player2_score);
361             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
362             HAL_Delay(3000);

```

```

358             keyChar = 0;
359         }
360     }
361
362     else if (keyChar == '2'){
363         if (rand_number == 1){
364             player1_score++;
365             sprintf(score, "                                %d--%d", player1_score,
366                     player2_score);
366             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
367             HAL_Delay(3000);
368             keyChar = 0;
369         }
370         else if (rand_number ==2){
371             sprintf(score, "                                %d--%d", player1_score,
372                     player2_score);
372             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
373             HAL_Delay(3000);
374             keyChar = 0;
375         }
376         else if (rand_number ==3){
377             player2_score++;
378             sprintf(score, "                                %d--%d",
379                     player1_score, player2_score);
379             HAL_UART_Transmit(&huart1, score, sizeof(score),
380                     HAL_MAX_DELAY);
380             HAL_Delay(3000);
381             keyChar = 0;
382         }
383         else if (rand_number ==4){
384             player2_score++;
385             sprintf(score, "                                %d--%d",
386                     player1_score, player2_score);
386             HAL_UART_Transmit(&huart1, score, sizeof(score),
387                     HAL_MAX_DELAY);
387             HAL_Delay(3000);
388             keyChar = 0;
389         }
390         else if (rand_number ==5){
391             player1_score++;
392             sprintf(score, "                                %d--%d", player1_score,
392                     player2_score);
393             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
394             HAL_Delay(3000);
395             keyChar = 0;
396         }
397     }
398     else if (keyChar == '3'){
399         if (rand_number == 1){
400             player2_score++;
401             sprintf(score, "                                %d--%d",
401                     player1_score, player2_score);
402             HAL_UART_Transmit(&huart1, score, sizeof(score),
402                     HAL_MAX_DELAY);
403             HAL_Delay(3000);
404             keyChar = 0;
405         }
406         else if (rand_number ==2){
407             player1_score++;
408             sprintf(score, "                                %d--%d",
408                     player1_score, player2_score);
409             HAL_UART_Transmit(&huart1, score, sizeof(score),
409                     HAL_MAX_DELAY);
410             HAL_Delay(3000);
411             keyChar = 0;
412         }
413         else if (rand_number ==3){
414             sprintf(score, "                                %d--%d", player1_score,

```

```

    player2_score);
415                                HAL_UART_Transmit(&huart1, score, sizeof(score),
416                                HAL_MAX_DELAY);
417                                HAL_Delay(3000);
418                                keyChar = 0;
419                            }
420                            else if (rand_number ==4){
421                                player1_score++;
422                                sprintf(score, "%d--%d",
423                                player1_score, player2_score);
424                                HAL_UART_Transmit(&huart1, score, sizeof(score),
425                                HAL_MAX_DELAY);
426                                HAL_Delay(3000);
427                                keyChar = 0;
428                            }
429                            else if (rand_number ==5){
430                                player2_score++;
431                                sprintf(score, "%d--%d",
432                                player1_score, player2_score);
433                                HAL_UART_Transmit(&huart1, score, sizeof(score),
434                                HAL_MAX_DELAY);
435                                HAL_Delay(3000);
436                                keyChar = 0;
437                            }
438                            else if (keyChar == '4'){
439                                if (rand_number == 1){
440                                    player2_score++;
441                                    sprintf(score, "%d--%d",
442                                    player1_score, player2_score);
443                                    HAL_UART_Transmit(&huart1, score, sizeof(score),
444                                    HAL_MAX_DELAY);
445                                    HAL_Delay(3000);
446                                    keyChar = 0;
447                                }
448                                else if (rand_number ==2){
449                                    player1_score++;
450                                    sprintf(score, "%d--%d",
451                                    player1_score, player2_score);
452                                    HAL_UART_Transmit(&huart1, score, sizeof(score),
453                                    HAL_MAX_DELAY);
454                                    HAL_Delay(3000);
455                                    keyChar = 0;
456                                }
457                                else if (rand_number ==3){
458                                    player2_score++;
459                                    sprintf(score, "%d--%d",
460                                    player1_score, player2_score);
461                                    HAL_UART_Transmit(&huart1, score, sizeof(score),
462                                    HAL_MAX_DELAY);
463                                    HAL_Delay(3000);
464                                    keyChar = 0;
465                                }
466                                else if (rand_number ==4){
467                                    sprintf(score, "%d--%d", player1_score,
468                                    player2_score);
469                                    HAL_UART_Transmit(&huart1, score, sizeof(score),
470                                    HAL_MAX_DELAY);
471                                    HAL_Delay(3000);
472                                    keyChar = 0;
473                                }
474                                else if (rand_number ==5){
475                                    player1_score++;
476                                    sprintf(score, "%d--%d",
477                                    player1_score, player2_score);
478                                    HAL_UART_Transmit(&huart1, score, sizeof(score),
479                                    HAL_MAX_DELAY);
480                                    HAL_Delay(3000);
481                                }
482                            }
483                        }
484                    }
485                }
486            }
487        }
488    }
489}

```

```

467                     keyChar = 0;
468                 }
469             }
470         else if (keyChar == '5'){
471             if (rand_number == 1){
472                 player1_score++;
473                 sprintf(score, "                                %d--%d",
474                         player1_score, player2_score);
475                 HAL_UART_Transmit(&huart1, score, sizeof(score),
476                         HAL_MAX_DELAY);
477                 HAL_Delay(3000);
478                 keyChar = 0;
479             }
480             else if (rand_number ==2){
481                 player2_score++;
482                 sprintf(score, "                                %d--%d",
483                         player1_score, player2_score);
484                 HAL_UART_Transmit(&huart1, score, sizeof(score),
485                         HAL_MAX_DELAY);
486                 HAL_Delay(3000);
487                 keyChar = 0;
488             }
489             else if (rand_number ==3){
490                 player1_score++;
491                 sprintf(score, "                                %d--%d",
492                         player1_score, player2_score);
493                 HAL_UART_Transmit(&huart1, score, sizeof(score),
494                         HAL_MAX_DELAY);
495                 HAL_Delay(3000);
496                 keyChar = 0;
497             }
498             else if (rand_number ==4){
499                 player2_score++;
500                 sprintf(score, "                                %d--%d",
501                         player1_score,
502                         player2_score);
503                 HAL_UART_Transmit(&huart1, score, sizeof(score),
504                         HAL_MAX_DELAY);
505                 HAL_Delay(3000);
506                 keyChar = 0;
507             }
508         else{
509             player2_score++;
510             sprintf(score, "                                %d--%d", player1_score,
511                         player2_score);
512             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
513             HAL_Delay(3000);
514             keyChar = 0;
515         }
516     clearScreen1();
517     HAL_Delay(100);
518
519
520     goto one_player;
521
522 }

```

```

524
525
526 if(player1_score > player2_score){
527     HAL_UART_Transmit(&huart1, winstatement, sizeof(winstatement), HAL_MAX_DELAY);
528     HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
529     HAL_Delay(7000);
530 }
531 else{
532     HAL_UART_Transmit(&huart1, loststatement, sizeof(loststatement), HAL_MAX_DELAY);
533     HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
534     HAL_Delay(7000);
535 }
536 goto welcome;
537
538
539
540
541
542
543
544     two_player:
545
546     if(player1_score < 3 && player2_score < 3){
547         HAL_UART_Transmit(&huart6, please_wait_2, sizeof(please_wait_2)-1, HAL_MAX_DELAY);
548         HAL_UART_Transmit(&huart1, player_one_wait_5, sizeof(player_one_wait_5)-1,
549         HAL_MAX_DELAY);
550         HAL_Delay(1000);
551         clearScreen1();
552         HAL_UART_Transmit(&huart1, player_one_wait_4, sizeof(player_one_wait_4)-1,
553         HAL_MAX_DELAY);
554         HAL_Delay(1000);
555         clearScreen1();
556         HAL_UART_Transmit(&huart1, player_one_wait_3, sizeof(player_one_wait_3)-1,
557         HAL_MAX_DELAY);
558         HAL_Delay(1000);
559         clearScreen1();
560         HAL_UART_Transmit(&huart1, player_one_wait_2, sizeof(player_one_wait_2)-1,
561         HAL_MAX_DELAY);
562         HAL_Delay(1000);
563         clearScreen1();
564
565         HAL_UART_Transmit(&huart1, player_one_choice_5, sizeof(player_one_choice_5)-1,
566         HAL_MAX_DELAY);
567         HAL_Delay(1000);
568         clearScreen1();
569         if(keyChar !=0){
570             goto pleasewait2;
571         }
572         HAL_UART_Transmit(&huart1, player_one_choice_4, sizeof(player_one_choice_4)-1,
573         HAL_MAX_DELAY);
574         HAL_Delay(1000);
575         clearScreen1();
576         if(keyChar !=0){
577             goto pleasewait2;
578         }
579         HAL_UART_Transmit(&huart1, player_one_choice_3, sizeof(player_one_choice_3)-1,
580         HAL_MAX_DELAY);
581         HAL_Delay(1000);
582         clearScreen1();
583         if(keyChar !=0){
584             goto pleasewait2;
585         }
586         HAL_UART_Transmit(&huart1, player_one_choice_2, sizeof(player_one_choice_2)-1,

```

```

    HAL_MAX_DELAY);
584     HAL_Delay(1000);
585     clearScreen1();
586     if(keyChar !=0){
587         goto pleasewait2;
588     }
589     HAL_UART_Transmit(&huart1, player_one_choice_1, sizeof(player_one_choice_1)-1,
HAL_MAX_DELAY);
590     HAL_Delay(1000);
591     clearScreen1();
592     goto pleasewait2;
593
594
595
596     pleasewait2:
597     clearScreen1();
598     HAL_UART_Transmit(&huart1, please_wait_1, sizeof(please_wait_1)-1, HAL_MAX_DELAY);
599     if (keyChar == '1'){
600         sprintf(player_one_choice_final, "Rock");
601     }
602     else if (keyChar == '2'){
603         sprintf(player_one_choice_final, "Paper");
604     }
605     else if (keyChar == '3'){
606         sprintf(player_one_choice_final, "Scissors");
607     }
608     else if (keyChar == '4'){
609         sprintf(player_one_choice_final, "Lizard");
610     }
611     else if (keyChar == '5'){
612         sprintf(player_one_choice_final, "Spock");
613     }
614     else{
615         keyChar=0;
616         sprintf(player_one_choice_final, "");
617         no_input_1 = 1;
618     }
619     clearScreen2();
620     HAL_Delay(100);
621     HAL_UART_Transmit(&huart6, player_one_wait_5, sizeof(player_one_wait_5)-1,
HAL_MAX_DELAY);
622     HAL_Delay(1000);
623     clearScreen2();
624     HAL_UART_Transmit(&huart6, player_one_wait_4, sizeof(player_one_wait_4)-1,
HAL_MAX_DELAY);
625     HAL_Delay(1000);
626     clearScreen2();
627     HAL_UART_Transmit(&huart6, player_one_wait_3, sizeof(player_one_wait_3)-1,
HAL_MAX_DELAY);
628     HAL_Delay(1000);
629     clearScreen2();
630     HAL_UART_Transmit(&huart6, player_one_wait_2, sizeof(player_one_wait_2)-1,
HAL_MAX_DELAY);
631     HAL_Delay(1000);
632     clearScreen2();
633     HAL_UART_Transmit(&huart6, player_one_wait_1, sizeof(player_one_wait_1)-1,
HAL_MAX_DELAY);
634     HAL_Delay(1000);
635     clearScreen2();
636
637
638     HAL_UART_Transmit(&huart6, player_one_choice_5, sizeof(player_one_choice_5)
)-1, HAL_MAX_DELAY);
639     HAL_Delay(1000);
640     clearScreen2();
641     if(keyChar2 !=0){
642         goto pleasewait3;
643     }

```

```

644     HAL_UART_Transmit(&huart6, player_one_choice_4, sizeof(player_one_choice_4
645 )-1, HAL_MAX_DELAY);
646     HAL_Delay(1000);
647     clearScreen2();
648     if(keyChar2 !=0){
649         goto pleasewait3;
650     }
651     HAL_UART_Transmit(&huart6, player_one_choice_3, sizeof(player_one_choice_3
652 )-1, HAL_MAX_DELAY);
653     HAL_Delay(1000);
654     clearScreen2();
655     if(keyChar2 !=0){
656         goto pleasewait3;
657     }
658     HAL_UART_Transmit(&huart6, player_one_choice_2, sizeof(player_one_choice_2
659 )-1, HAL_MAX_DELAY);
660     HAL_Delay(1000);
661     clearScreen2();
662     if(keyChar2 !=0){
663         goto pleasewait3;
664     }
665     HAL_UART_Transmit(&huart6, player_one_choice_1, sizeof(player_one_choice_1
666 )-1, HAL_MAX_DELAY);
667     HAL_Delay(1000);
668     clearScreen2();
669     goto pleasewait3;
670
671     pleasewait3:
672     clearScreen1();
673     clearScreen2();
674     HAL_Delay(100);
675     HAL_UART_Transmit(&huart1, buffer, sizeof(buffer)-1, HAL_MAX_DELAY);
676     HAL_UART_Transmit(&huart6, buffer, sizeof(buffer)-1, HAL_MAX_DELAY);
677     HAL_Delay(2000);
678
679     clearScreen2();
680     clearScreen1();
681     HAL_Delay(100);
682     goto results2;
683
684 results2:
685
686     if (keyChar2 == '1'){
687         sprintf(player_two_choice_final, "Rock");
688     }
689     else if (keyChar2 == '2'){
690         sprintf(player_two_choice_final, "Paper");
691     }
692     else if (keyChar2 == '3'){
693         sprintf(player_two_choice_final, "Scissors");
694     }
695     else if (keyChar2 == '4'){
696         sprintf(player_two_choice_final, "Lizard");
697     }
698     else if (keyChar2 == '5'){
699         sprintf(player_two_choice_final, "Spock");
700     }
701     else{
702         keyChar2=0;
703         sprintf(player_two_choice_final, "no_input_2 = 1");
704     }
705
706     HAL_UART_Transmit(&huart1, player1_choice_message, sizeof(player1_choice_message)
707 -1, HAL_MAX_DELAY);
708     HAL_UART_Transmit(&huart1, player_one_choice_final, sizeof(player_one_choice_final
709 )-1, HAL_MAX_DELAY);

```

```

706     HAL_UART_Transmit(&huart1, player2_choice_message, sizeof(player2_choice_message)
707     -1, HAL_MAX_DELAY);
708     HAL_UART_Transmit(&huart1, player_two_choice_final, sizeof(
709     player_two_choice_final)-1, HAL_MAX_DELAY);
710     HAL_UART_Transmit(&huart6, player1_choice_message, sizeof(player1_choice_message)
711     -1, HAL_MAX_DELAY);
712     HAL_UART_Transmit(&huart6, player_one_choice_final, sizeof(
713     player_one_choice_final)-1, HAL_MAX_DELAY);
714     HAL_UART_Transmit(&huart6, player2_choice_message, sizeof(
715     player2_choice_message)-1, HAL_MAX_DELAY);
716     HAL_UART_Transmit(&huart6, player_two_choice_final, sizeof(
717     player_two_choice_final)-1, HAL_MAX_DELAY);

718     HAL_Delay(5000);

719     clearScreen1();
720     clearScreen2();
721     if (keyChar == '1'){
722         if (keyChar2 == '1'){
723             sprintf(score, "                                %d-%d",
724                     player1_score,
725                     player2_score);
726             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
727             HAL_UART_Transmit(&huart6, score, sizeof(score), HAL_MAX_DELAY);
728             HAL_Delay(3000);
729             keyChar = 0;
730             keyChar2 = 0;
731         }
732         else if (keyChar2 == '2'){
733             player2_score++;
734             sprintf(score, "                                %d-%d",
735                     player1_score,
736                     player2_score);
737             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
738             HAL_UART_Transmit(&huart6, score, sizeof(score), HAL_MAX_DELAY);
739             HAL_Delay(3000);
740             keyChar = 0;
741             keyChar2 = 0;
742         }
743         else if (keyChar2 == '3'){
744             player1_score++;
745             sprintf(score, "                                %d-%d",
746                     player1_score,
747                     player2_score);
748             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
749             HAL_UART_Transmit(&huart6, score, sizeof(score), HAL_MAX_DELAY);
750             HAL_Delay(3000);
751             keyChar = 0;
752             keyChar2 = 0;
753         }
754         else if (keyChar2 == '4'){
755             player1_score++;
756             sprintf(score, "                                %d-%d",
757                     player1_score,
758                     player2_score);
759             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
760             HAL_UART_Transmit(&huart6, score, sizeof(score), HAL_MAX_DELAY);
761             HAL_Delay(3000);
762             keyChar = 0;
763             keyChar2 = 0;
764         }
765         else if (keyChar2 == '5'){
766             player2_score++;
767             sprintf(score, "                                %d-%d",
768                     player1_score,
769                     player2_score);
770             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
771             HAL_UART_Transmit(&huart6, score, sizeof(score), HAL_MAX_DELAY);
772             HAL_Delay(3000);
773             keyChar = 0;
774             keyChar2 = 0;
775         }
776     }
777 
```

```

760             HAL_Delay(3000);
761             keyChar = 0;
762             keyChar2 = 0;
763
764         }
765         else{
766             player1_score++;
767             sprintf(score, "%d--%d", player1_score,
768             player2_score);
769             HAL_UART_Transmit(&huart1, score, sizeof(score),
770             HAL_MAX_DELAY);
771             HAL_UART_Transmit(&huart6, score, sizeof(score),
772             HAL_MAX_DELAY);
773             HAL_Delay(3000);
774             keyChar = 0;
775             keyChar2 = 0;
776         }
777
778     else if (keyChar == '2'){
779         if (keyChar2 == '1'){
780             player1_score++;
781             sprintf(score, "%d--%d", player1_score,
782             player2_score);
783             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
784             HAL_UART_Transmit(&huart6, score, sizeof(score), HAL_MAX_DELAY);
785             HAL_Delay(3000);
786             keyChar = 0;
787             keyChar2 = 0;
788         }
789         else if (keyChar2 == '2'){
790             sprintf(score, "%d--%d", player1_score,
791             player2_score);
792             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
793             HAL_UART_Transmit(&huart6, score, sizeof(score), HAL_MAX_DELAY);
794             HAL_Delay(3000);
795             keyChar = 0;
796             keyChar2 = 0;
797         }
798         else if (keyChar2 == '3'){
799             player2_score++;
800             sprintf(score, "%d--%d", player1_score,
801             player2_score);
802             HAL_UART_Transmit(&huart1, score, sizeof(score),
803             HAL_MAX_DELAY);
804             HAL_UART_Transmit(&huart6, score, sizeof(score),
805             HAL_MAX_DELAY);
806             HAL_Delay(3000);
807             keyChar = 0;
808             keyChar2 = 0;
809         }
810         else if (keyChar2 == '4'){
811             player2_score++;
812             sprintf(score, "%d--%d", player1_score,
813             player2_score);
814             HAL_UART_Transmit(&huart1, score, sizeof(score),
815             HAL_MAX_DELAY);
816             HAL_UART_Transmit(&huart6, score, sizeof(score),
817             HAL_MAX_DELAY);
818             HAL_Delay(3000);
819             keyChar = 0;
820             keyChar2 = 0;
821         }
822         else if (keyChar2 == '5'){
823             player1_score++;
824             sprintf(score, "%d--%d", player1_score,
825             player2_score);
826             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
827             HAL_UART_Transmit(&huart6, score, sizeof(score), HAL_MAX_DELAY);
828             HAL_Delay(3000);
829             keyChar = 0;
830             keyChar2 = 0;
831         }
832     }
833 }
```

```

816             HAL_UART_Transmit(&huart6, score, sizeof(score), HAL_MAX_DELAY);
817             HAL_Delay(3000);
818             keyChar = 0;
819             keyChar2 = 0;
820         }
821     else{
822         player1_score++;
823         sprintf(score, "%d--%d",
824             player1_score, player2_score);
825         HAL_UART_Transmit(&huart1,
826             score, sizeof(score), HAL_MAX_DELAY);
827         HAL_UART_Transmit(&huart6, score
828             , sizeof(score), HAL_MAX_DELAY);
829         HAL_Delay(3000);
830         keyChar = 0;
831         keyChar2 = 0;
832     }
833     else if (keyChar == '3'){
834         if (keyChar2 == '1'){
835             player2_score++;
836             sprintf(score, "%d--%d",
837                 player1_score, player2_score);
838             HAL_UART_Transmit(&huart1, score, sizeof(score),
839                 HAL_MAX_DELAY);
840             HAL_UART_Transmit(&huart6, score, sizeof(score),
841                 HAL_MAX_DELAY);
842             HAL_Delay(3000);
843             keyChar = 0;
844             keyChar2 = 0;
845         }
846         else if (keyChar2 == '2'){
847             player1_score++;
848             sprintf(score, "%d--%d",
849                 player1_score, player2_score);
850             HAL_UART_Transmit(&huart1, score, sizeof(score),
851                 HAL_MAX_DELAY);
852             HAL_UART_Transmit(&huart6, score, sizeof(score),
853                 HAL_MAX_DELAY);
854             HAL_Delay(3000);
855             keyChar = 0;
856             keyChar2 = 0;
857         }
858         else if (keyChar2 == '3'){
859             sprintf(score, "%d--%d",
860                 player1_score, player2_score);
861             HAL_UART_Transmit(&huart1, score, sizeof(score),
862                 HAL_MAX_DELAY);
863             HAL_UART_Transmit(&huart6, score, sizeof(score),
864                 HAL_MAX_DELAY);
865             HAL_Delay(3000);
866             keyChar = 0;
867             keyChar2 = 0;
868         }
869     else if (keyChar2 == '4'){
870         player1_score++;
871         sprintf(score, "%d--%d",
872             player1_score, player2_score);
873         HAL_UART_Transmit(&huart1, score, sizeof(score),
874             HAL_MAX_DELAY);
875         HAL_UART_Transmit(&huart6, score, sizeof(score),
876             HAL_MAX_DELAY);
877         HAL_Delay(3000);
878         keyChar = 0;
879         keyChar2 = 0;
880     }
881 
```

```

869                                     sprintf(score, "%d--%d"
870     ", player1_score, player2_score);
871     HAL_MAX_DELAY);
872     HAL_MAX_DELAY);
873     HAL_MAX_DELAY);
874     }
875     else{
876         player1_score++;
877         sprintf(score, "%d--%d",
878             player1_score, player2_score);
879         HAL_UART_Transmit(&huart1, score, sizeof(score),
880             HAL_MAX_DELAY);
881         HAL_UART_Transmit(&huart6, score, sizeof(score),
882             (score), HAL_MAX_DELAY);
883         HAL_Delay(3000);
884         keyChar = 0;
885     }
886     else if (keyChar == '4'){
887         if (keyChar2 == '1'){
888             player2_score++;
889             sprintf(score, "%d--%d",
890                 player1_score, player2_score);
891             HAL_UART_Transmit(&huart1, score, sizeof(score),
892                 HAL_MAX_DELAY);
893             HAL_UART_Transmit(&huart6, score, sizeof(score),
894                 HAL_MAX_DELAY);
895             HAL_Delay(3000);
896             keyChar = 0;
897             keyChar2 = 0;
898         }
899         else if (keyChar2 == '2'){
900             player1_score++;
901             sprintf(score, "%d--%d",
902                 player1_score, player2_score);
903             HAL_UART_Transmit(&huart1, score, sizeof(score),
904                 HAL_MAX_DELAY);
905             HAL_UART_Transmit(&huart6, score, sizeof(score),
906                 HAL_MAX_DELAY);
907             HAL_Delay(3000);
908             keyChar = 0;
909             keyChar2 = 0;
910         }
911         else if (keyChar2 == '3'){
912             player2_score++;
913             sprintf(score, "%d--%d",
914                 player1_score, player2_score);
915             HAL_UART_Transmit(&huart1, score, sizeof(score),
916                 HAL_MAX_DELAY);
917             HAL_UART_Transmit(&huart6, score, sizeof(score),
918                 HAL_MAX_DELAY);
919             HAL_Delay(3000);

```

```

919                     keyChar = 0;
920                     keyChar2 = 0;
921                 }
922             else if (keyChar2 == '5'){
923                 player1_score++;
924                 sprintf(score, "%d--%d",
925                         player1_score, player2_score);
926                 HAL_UART_Transmit(&huart1, score, sizeof(score),
927                                     HAL_MAX_DELAY);
928                 HAL_UART_Transmit(&huart6, score, sizeof(score),
929                                     HAL_MAX_DELAY);
930                 HAL_Delay(3000);
931                 keyChar = 0;
932                 keyChar2 = 0;
933             }
934         else{
935             player1_score++;
936             sprintf(score, "%d--%d",
937                     player1_score, player2_score);
938             HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
939             HAL_UART_Transmit(&huart6, score, sizeof(score), HAL_MAX_DELAY);
940             HAL_Delay(3000);
941             keyChar = 0;
942             keyChar2 = 0;
943         }
944     else if (keyChar == '5'){
945         if (keyChar2 == '1'){
946             player2_score++;
947             sprintf(score, "%d--%d",
948                     player1_score, player2_score);
949             HAL_UART_Transmit(&huart1, score, sizeof(score),
950                               HAL_MAX_DELAY);
951             HAL_UART_Transmit(&huart6, score, sizeof(score),
952                               HAL_MAX_DELAY);
953             HAL_Delay(3000);
954             keyChar = 0;
955             keyChar2 = 0;
956         }
957         else if (keyChar2 == '2'){
958             player1_score++;
959             sprintf(score, "%d--%d",
960                     player1_score, player2_score);
961             HAL_UART_Transmit(&huart1, score, sizeof(score),
962                               HAL_MAX_DELAY);
963             HAL_UART_Transmit(&huart6, score, sizeof(score),
964                               HAL_MAX_DELAY);
965             HAL_Delay(3000);
966             keyChar = 0;
967             keyChar2 = 0;
968         }
969         else if (keyChar2 == '3'){
970             player1_score++;
971             sprintf(score, "%d--%d",
972                     player1_score, player2_score);
973             HAL_UART_Transmit(&huart1, score, sizeof(score),
974                               HAL_MAX_DELAY);
975             HAL_UART_Transmit(&huart6, score, sizeof(score),
976                               HAL_MAX_DELAY);
977             HAL_Delay(3000);
978             keyChar = 0;
979             keyChar2 = 0;
980         }
981     else if (keyChar2 == '4'){
982         player2_score++;
983         sprintf(score, "%d--%d",
984                     player1_score, player2_score);
985     }
986 }
987

```

```

972     HAL_MAX_DELAY);
973     HAL_MAX_DELAY);
974     HAL_Delay(3000);
975     keyChar = 0;
976     keyChar2 = 0;
977 }
978 else if (keyChar2 == '5'){
979     sprintf(score, "%d--%d", player1_score,
980     player2_score);
981     HAL_MAX_DELAY);
982     HAL_MAX_DELAY);
983     HAL_Delay(3000);
984     keyChar = 0;
985     keyChar2 = 0;
986 }
987 else{
988     player1_score++;
989     sprintf(score, "%d--%d", player1_score, player2_score);
990     HAL_UART_Transmit(&huart1, score, sizeof(score),
991     HAL_MAX_DELAY);
992     HAL_UART_Transmit(&huart6, score, sizeof(score), HAL_MAX_DELAY);
993     HAL_Delay(3000);
994     keyChar = 0;
995     keyChar2 = 0;
996 }
997 else{
998     if (keyChar2 == '1' || keyChar2 == '2' || keyChar2 == '3' || keyChar2 == '4'
999     || keyChar2 == '5'){
1000         player2_score++;
1001
1002         sprintf(score, "%d--%d", player1_score,
1003         player2_score);
1004         HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
1005         HAL_UART_Transmit(&huart6, score, sizeof(score), HAL_MAX_DELAY);
1006         HAL_Delay(3000);
1007         keyChar = 0;
1008         keyChar2=0;
1009     }
1010     else{
1011         sprintf(score, "%d--%d", player1_score,
1012         player2_score);
1013         HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
1014         HAL_UART_Transmit(&huart6, score, sizeof(score), HAL_MAX_DELAY);
1015         HAL_Delay(3000);
1016         keyChar = 0;
1017         clearScreen1();
1018         clearScreen2();
1019         HAL_Delay(100);
1020
1021         game_counter++;
1022
1023
1024
1025         goto two_player;
1026
1027 }
1028
1029

```

```

1030     if(player1_score > player2_score){
1031         HAL_UART_Transmit(&huart1, winstatement, sizeof(winstatement)-1, HAL_MAX_DELAY);
1032         HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
1033         HAL_UART_Transmit(&huart6, loststatement, sizeof(loststatement), HAL_MAX_DELAY);
1034             HAL_UART_Transmit(&huart6, score, sizeof(score), HAL_MAX_DELAY);
1035         HAL_Delay(7000);
1036     }
1037     else{
1038         HAL_UART_Transmit(&huart1, loststatement, sizeof(loststatement), HAL_MAX_DELAY);
1039         HAL_UART_Transmit(&huart1, score, sizeof(score), HAL_MAX_DELAY);
1040         HAL_UART_Transmit(&huart6, winstatement, sizeof(winstatement)-1, HAL_MAX_DELAY);
1041             HAL_UART_Transmit(&huart6, score, sizeof(score), HAL_MAX_DELAY);
1042         HAL_Delay(7000);
1043     }
1044     goto welcome;
1045
1046
1047
1048
1049
1050     /* USER CODE END WHILE */
1051
1052     /* USER CODE BEGIN 3 */
1053 }
1054
1055 /* USER CODE END 3 */
1056
1057 /**
1058 * @brief System Clock Configuration
1059 * @retval None
1060 */
1061 void SystemClock_Config(void)
1062 {
1063     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
1064     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
1065
1066     /** Configure the main internal regulator output voltage
1067     */
1068     __HAL_RCC_PWR_CLK_ENABLE();
1069     __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
1070
1071     /** Initializes the RCC Oscillators according to the specified parameters
1072     * in the RCC_OscInitTypeDef structure.
1073     */
1074     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
1075     RCC_OscInitStruct.HSISState = RCC_HSI_ON;
1076     RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
1077     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
1078     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
1079     RCC_OscInitStruct.PLL.PLLM = 16;
1080     RCC_OscInitStruct.PLL.PLLN = 336;
1081     RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
1082     RCC_OscInitStruct.PLL.PLLQ = 4;
1083     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
1084     {
1085         Error_Handler();
1086     }
1087
1088     /** Initializes the CPU, AHB and APB buses clocks
1089     */
1090     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
1091             |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
1092     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
1093     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
1094     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
1095     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
1096
1097     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)

```

```

1098     {
1099         Error_Handler();
1100     }
1101 }
1102
1103 /**
1104 * @brief USART1 Initialization Function
1105 * @param None
1106 * @retval None
1107 */
1108 static void MX_USART1_UART_Init(void)
1109 {
1110
1111     /* USER CODE BEGIN USART1_Init 0 */
1112
1113     /* USER CODE END USART1_Init 0 */
1114
1115     /* USER CODE BEGIN USART1_Init 1 */
1116
1117     /* USER CODE END USART1_Init 1 */
1118     huart1.Instance = USART1;
1119     huart1.Init.BaudRate = 9600;
1120     huart1.Init.WordLength = UART_WORDLENGTH_8B;
1121     huart1.Init.StopBits = UART_STOPBITS_1;
1122     huart1.Init.Parity = UART_PARITY_NONE;
1123     huart1.Init.Mode = UART_MODE_TX_RX;
1124     huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
1125     huart1.Init.OverSampling = UART_OVERSAMPLING_16;
1126     if (HAL_UART_Init(&huart1) != HAL_OK)
1127     {
1128         Error_Handler();
1129     }
1130     /* USER CODE BEGIN USART1_Init 2 */
1131
1132     /* USER CODE END USART1_Init 2 */
1133
1134 }
1135
1136 /**
1137 * @brief USART2 Initialization Function
1138 * @param None
1139 * @retval None
1140 */
1141 static void MX_USART2_UART_Init(void)
1142 {
1143
1144     /* USER CODE BEGIN USART2_Init 0 */
1145
1146     /* USER CODE END USART2_Init 0 */
1147
1148     /* USER CODE BEGIN USART2_Init 1 */
1149
1150     /* USER CODE END USART2_Init 1 */
1151     huart2.Instance = USART2;
1152     huart2.Init.BaudRate = 115200;
1153     huart2.Init.WordLength = UART_WORDLENGTH_8B;
1154     huart2.Init.StopBits = UART_STOPBITS_1;
1155     huart2.Init.Parity = UART_PARITY_NONE;
1156     huart2.Init.Mode = UART_MODE_TX_RX;
1157     huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
1158     huart2.Init.OverSampling = UART_OVERSAMPLING_16;
1159     if (HAL_UART_Init(&huart2) != HAL_OK)
1160     {
1161         Error_Handler();
1162     }
1163     /* USER CODE BEGIN USART2_Init 2 */
1164
1165     /* USER CODE END USART2_Init 2 */

```

```

1166 }
1167 /**
1168 * @brief USART6 Initialization Function
1169 * @param None
1170 * @retval None
1171 */
1172 static void MX_USART6_UART_Init(void)
1173 {
1174 /* USER CODE BEGIN USART6_Init_0 */
1175
1176 /* USER CODE END USART6_Init_0 */
1177
1178 /* USER CODE BEGIN USART6_Init_1 */
1179
1180 /* USER CODE END USART6_Init_1 */
1181
1182 /* USER CODE END USART6_Init_1 */
1183 huart6.Instance = USART6;
1184 huart6.Init.BaudRate = 9600;
1185 huart6.Init.WordLength = UART_WORDLENGTH_8B;
1186 huart6.Init.StopBits = UART_STOPBITS_1;
1187 huart6.Init.Parity = UART_PARITY_NONE;
1188 huart6.Init.Mode = UART_MODE_TX_RX;
1189 huart6.Init.HwFlowCtl = UART_HWCONTROL_NONE;
1190 huart6.Init.OverSampling = UART_OVERSAMPLING_16;
1191 if (HAL_UART_Init(&huart6) != HAL_OK)
1192 {
1193     Error_Handler();
1194 }
1195 /* USER CODE BEGIN USART6_Init_2 */
1196
1197 /* USER CODE END USART6_Init_2 */
1198
1199 }
1200
1201 /**
1202 * @brief GPIO Initialization Function
1203 * @param None
1204 * @retval None
1205 */
1206 static void MX_GPIO_Init(void)
1207 {
1208     GPIO_InitTypeDef GPIO_InitStruct = {0};
1209
1210     /* GPIO Ports Clock Enable */
1211     __HAL_RCC_GPIOC_CLK_ENABLE();
1212     __HAL_RCC_GPIOH_CLK_ENABLE();
1213     __HAL_RCC_GPIOA_CLK_ENABLE();
1214     __HAL_RCC_GPIOB_CLK_ENABLE();
1215
1216     /*Configure GPIO pin Output Level*/
1217     HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, GPIO_PIN_RESET);
1218
1219     /*Configure GPIO pin Output Level*/
1220     HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
1221
1222     /*Configure GPIO pin Output Level*/
1223     HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7, GPIO_PIN_RESET);
1224
1225     /*Configure GPIO pins : PC0 PC1 PC2 PC3 */
1226     GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3;
1227     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
1228     GPIO_InitStruct.Pull = GPIO_PULLDOWN;
1229     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
1230     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
1231
1232     /*Configure GPIO pin : LD2_Pin */
1233

```

```

1234     GPIO_InitStruct.Pin = LD2_Pin;
1235     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
1236     GPIO_InitStruct.Pull = GPIO_NOPULL;
1237     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
1238     HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
1239
1240     /*Configure GPIO pins : PC5 PC6 PC7 PC8 */
1241     GPIO_InitStruct.Pin = GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_8;
1242     GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
1243     GPIO_InitStruct.Pull = GPIO_PULLUP;
1244     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
1245
1246     /*Configure GPIO pins : PB12 PB13 PB14 PB15 */
1247     GPIO_InitStruct.Pin = GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
1248     GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
1249     GPIO_InitStruct.Pull = GPIO_PULLUP;
1250     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
1251
1252     /*Configure GPIO pins : PB4 PB5 PB6 PB7 */
1253     GPIO_InitStruct.Pin = GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7;
1254     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
1255     GPIO_InitStruct.Pull = GPIO_NOPULL;
1256     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
1257     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
1258
1259     /* EXTI interrupt init*/
1260     HAL_NVIC_SetPriority(EXTI9_5_IRQn, 0, 0);
1261     HAL_NVIC_EnableIRQ(EXTI9_5_IRQn);
1262
1263     HAL_NVIC_SetPriority(EXTI15_10_IRQn, 0, 0);
1264     HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
1265 }
1266
1267
1268 /* USER CODE BEGIN 4 */
1269 void clearScreen1(void){
1270     uint8_t clearscreen1[] = {0x7C, 0x2D};
1271     HAL_UART_Transmit(&huart1, clearscreen1, sizeof(clearscreen1), HAL_MAX_DELAY);
1272 }
1273 void clearScreen2(void){
1274     uint8_t clearscreen2[] = {0x7C, 0x2D};
1275     HAL_UART_Transmit(&huart6, clearscreen2, sizeof(clearscreen2), HAL_MAX_DELAY);
1276 }
1277
1278 /* USER CODE END 4 */
1279
1280 /**
1281 * @brief This function is executed in case of error occurrence.
1282 * @retval None
1283 */
1284 void Error_Handler(void)
1285 {
1286     /* USER CODE BEGIN Error_Handler_Debug */
1287     /* User can add his own implementation to report the HAL error return state */
1288     __disable_irq();
1289     while (1)
1290     {
1291     }
1292     /* USER CODE END Error_Handler_Debug */
1293 }
1294
1295 #ifdef USE_FULL_ASSERT
1296 /**
1297 * @brief Reports the name of the source file and the source line number
1298 * where the assert_param error has occurred.
1299 * @param file: pointer to the source file name
1300 * @param line: assert_param error line source number
1301 * @retval None

```

```
1302 */  
1303 void assert_failed(uint8_t *file, uint32_t line)  
1304 {  
1305     /* USER CODE BEGIN 6 */  
1306     /* User can add his own implementation to report the file name and line number,  
1307      ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */  
1308     /* USER CODE END 6 */  
1309 }  
1310 #endif /* USE_FULL_ASSERT */
```

Listing A.1: Source Code

Appendix B

Bill of Materials

B.1 Cost breakdown

| | |
|-----------------------|-----------------|
| NUCLEO F411RE | \$12.74 |
| LCD Screen (x2) | \$59.90 |
| . (@ \$29.95/ea) | |
| Keypad (x2) | \$70.92 |
| . (@ \$35.46/ea) | |
| Breadboard | \$4.20 |
| TOTAL | \$141.76 |

Appendix C

Rating

This game is rated G for everyone 3 and up.

