Jordan Ditzler – G00967092
ECE 350-001 – FALL 2020

Lab 3 Report

Description of Game:

My Simon Says game follows the original concept very closely.

On startup, the computer generates a randomly generated sequence of Red, Green, Yellow, and Blue. The player, or user, starts at level 1 and must use the corresponding buttons to input the correct sequence of LEDs to match which flashed. Upon a correct sequence input, the computer will increase the user's level by 1, which will increase the sequence shown by 1, and repeat the process until the user has reached the game winning level. For testing purposes, and easy demonstration, the user must complete **level 5** to win the game. This can easily be changed by editing the variable "winLevel" in the source code. Upon an incorrect sequence input, the computer will end the game.

Every time the user inputs a LED into the sequence, the current user sequence will be shown in the console. The current user sequence resets upon level up. Also, the console will display if the LED input is correct or incorrect, when the game has been won or lost, and what the current level the user is on. Upon an incorrect input, the computer will display the current user sequence and the entire correct sequence before exiting the game.

A buzzer will sound every time a LED has flashed, both when the LED is flashing the randomly generated sequence and when the user is inputting a LED. Each LED is associated with a unique buzzer signal.

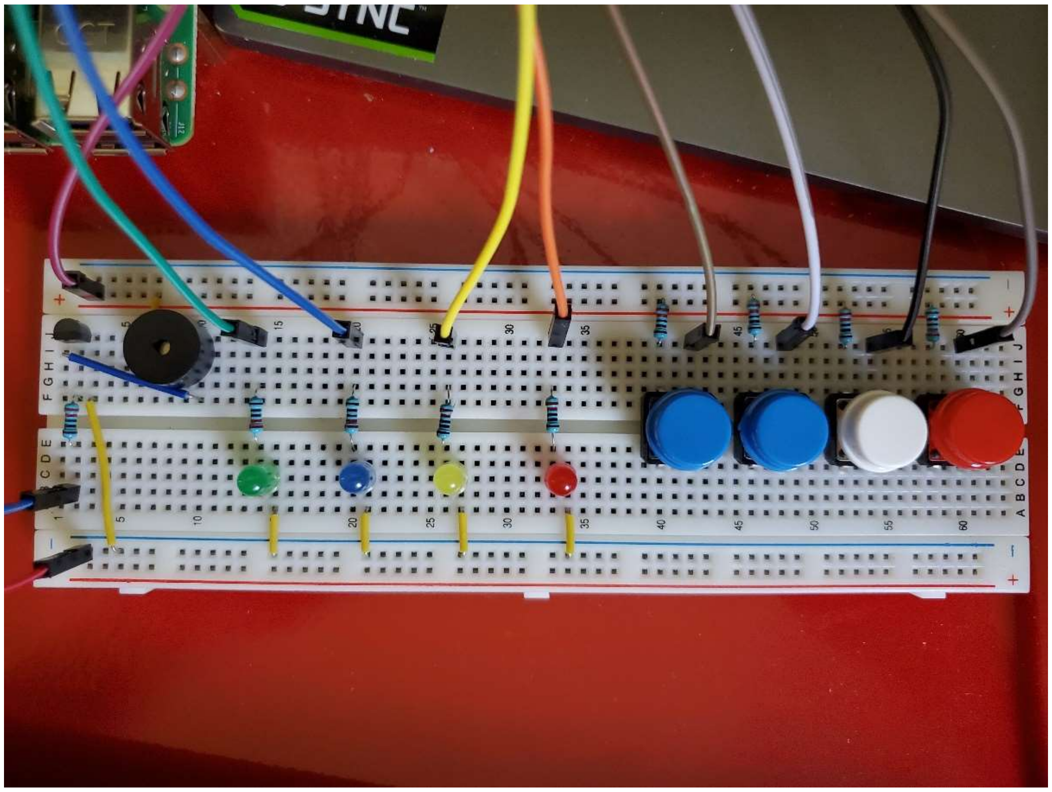
Board and Circuit:

Each device has a 220 Ω resistor is series with it to limit current flow.

The transistor (connected to the piezo buzzer) is to amplify the PWM signal input.

Wiring (Left → Right):

> Magenta: Connected to Vcc (3.3V) on board pin 1. Applies Vcc power to all devices connected to the top power rail.
> Leftmost Blue: Connected to GPIO11, board pin 23. Applies a PWM signal for the buzzer.
> Red: Connected to GND on board pin 39. Applies GND connection to all devices connected to bottom GND rail.
> Green: Connected to GPIO27, board pin 13. Applies output power to the **green LED**.
> Middle Blue: Connected to GPIO17, board pin 11. Applies output power to the **blue LED**.
> Yellow: Connected to GPIO18, board pin 12. Applies output power to the **yellow LED**.
> Orange: Connected to GPIO15, board pin 10. Applies output power to the **red LED**.
> Brown: Connected to GPIO23, board pin 16. Takes input and is associated with the **green LED**.
> White: Connected to GPIO24, board pin 18. Takes input and is associated with the **blue LED**.
> Black: Connected to GPIO10, board pin 19. Takes input and is associated with the **yellow LED**.
> Grey: Connected to GPIO9, board pin 21. Takes input and is associated with the **red LED**.



C Code:

```c
#include <wiringPi.h>
#include <softPwm.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <time.h>

// Define all LED, Button, and Buzzer macros
#define RedLED  16  // GPIO15, pin 10
#define YelLED  1   // GPIO18, pin 12
#define BluLED  0   // GPIO17, pin 11
#define GreLED  2   // GPIO27, pin 13

#define BtnG    4   // GPIO23, pin 16
#define BtnB    5   // GPIO24, pin 18
#define BtnY    12  // GPIO10, pin 19
#define BtnR    13  // GPIO9,  pin 21

#define Buzzer  14  // GPIO11, pin 23
#define GStr    25  // Green Strength 25%
#define BStr    50  // Blue Strength 50%
#define YStr    75  // Yellow Strength 75%
#define RStr    100 // Red Strength 100%

// Easily changeable level to win the game
int winLevel = 5;

int BtnNum, userLevel = 1, FLAG, testLevel;
int compSeq[10], userSeq[10];

// Interrupt service for Green LED and its button
void BtnGInterrupt(void)
{
    BtnNum = 1;
    digitalWrite(GreLED, HIGH);
    softPwmWrite(Buzzer, GStr);
    delay(500);
    digitalWrite(GreLED, LOW);
    softPwmWrite(Buzzer, 0);
}

// Interrupt service for Blue LED and its button
void BtnBInterrupt(void)
{
    BtnNum = 2;
    digitalWrite(BluLED, HIGH);
    softPwmWrite(Buzzer, BStr);
    delay(500);
    digitalWrite(BluLED, LOW);
    softPwmWrite(Buzzer, 0);
}

// Interrupt service for Yellow LED and its button
void BtnYInterrupt(void)
```

```
{
    BtnNum = 3;
    digitalWrite(YelLED, HIGH);
    softPwmWrite(Buzzer, YStr);
    delay(500);
    digitalWrite(YelLED, LOW);
    softPwmWrite(Buzzer, 0);
}

// Interrupt service for Red LED and its button
void BtnRInterrupt(void)
{
    BtnNum = 4;
    digitalWrite(RedLED, HIGH);
    softPwmWrite(Buzzer, RStr);
    delay(500);
    digitalWrite(RedLED, LOW);
    softPwmWrite(Buzzer, 0);
}

// Flashes Green LED as part of the computer sequence
void flashGLED(void)
{
    digitalWrite(GreLED, HIGH);
    softPwmWrite(Buzzer, GStr);
    delay(500);
    digitalWrite(GreLED, LOW);
    softPwmWrite(Buzzer, 0);
    delay(100);
}

// Flashes Blue LED as part of the computer sequence
void flashBLED(void)
{
    digitalWrite(BluLED, HIGH);
    softPwmWrite(Buzzer, BStr);
    delay(500);
    digitalWrite(BluLED, LOW);
    softPwmWrite(Buzzer, 0);
    delay(100);
}

// Flashes Yellow LED as part of the computer sequence
void flashYLED(void)
{
    digitalWrite(YelLED, HIGH);
    softPwmWrite(Buzzer, YStr);
    delay(500);
    digitalWrite(YelLED, LOW);
    softPwmWrite(Buzzer, 0);
    delay(100);
}

// Flashes Red LED as part of the computer sequence
void flashRLED(void)
{
    digitalWrite(RedLED, HIGH);
    softPwmWrite(Buzzer, RStr);
    delay(500);
    digitalWrite(RedLED, LOW);
    softPwmWrite(Buzzer, 0);
    delay(100);
}

// Converts LED # to char
void printLED(int num)
{
    switch(num)
    {
        case 1: printf(" G"); break;
        case 2: printf(" B"); break;
        case 3: printf(" Y"); break;
        case 4: printf(" R"); break;
    }
}

int main(void)
{
    // Sets up WiringPi
    if (wiringPiSetup() < 0)
    {
        printf("Setup wiringPi failed!\n");
        return -1;
    }

    // Sets up LED pins
    pinMode(RedLED, OUTPUT); // Red LED
    pinMode(YelLED, OUTPUT); // Yellow LED
    pinMode(BluLED, OUTPUT); // Blue LED
    pinMode(GreLED, OUTPUT); // Green LED

    // Sets up Button pins
    pinMode(BtnG, INPUT); // Button 1
    pinMode(BtnB, INPUT); // Button 2
    pinMode(BtnY, INPUT); // Button 3
    pinMode(BtnR, INPUT); // Button 4

    // Sets up Buzzer pin
    pinMode(Buzzer, PWM_OUTPUT); // Buzzer

    // Sets up Buzzer PWM
    if (softPwmCreate(Buzzer, 0, 100) < 0)
    {
        printf("Unable to set up PWM.");
        return -1;
    }

    // Sets up Green LED interrupt
    if (wiringPiISR(BtnG, INT_EDGE_FALLING, &BtnGInterrupt) < 0)
    {
        fprintf (stderr, "Unable to setup ISR: %s\n", strerror (errno));
        return -1;
    }
```

```
// Sets up Blue LED interrupt
if (wiringPiISR(BtnB, INT_EDGE_FALLING, &BtnBInterrupt) < 0)
{
    fprintf (stderr, "Unable to setup ISR: %s\n", strerror (errno));
    return -1;
}

// Sets up Yellow LED interrupt
if (wiringPiISR(BtnY, INT_EDGE_FALLING, &BtnYInterrupt) < 0)
{
    fprintf (stderr, "Unable to setup ISR: %s\n", strerror (errno));
    return -1;
}

// Sets up Red LED interrupt
if (wiringPiISR(BtnR, INT_EDGE_FALLING, &BtnRInterrupt) < 0)
{
    fprintf (stderr, "Unable to setup ISR: %s\n", strerror (errno));
    return -1;
}

// Generates random computer sequence
srand(time(NULL));
int i;
for (i = 0; i < 10; i++) compSeq[i] = (rand() % 4) + 1;

while (1)
{
    // Resets variables for current iteration
    FLAG = 1;
    BtnNum = -1;

    // Prints current level
    printf("Current level is %d.\n", userLevel);

    // Flashes computer LED sequence
    for(i = 0; i < userLevel; i++)
    {
        switch(compSeq[i])
        {
            case 1: flashGLED(); break;
            case 2: flashBLED(); break;
            case 3: flashYLED(); break;
            case 4: flashRLED(); break;
        }
    }

    // Resets test iterator
    testLevel = 1;

    // While FLAG has not been tripped
    while (FLAG)
    {
        // If any button has been pressed
        if (BtnNum != -1)
        {
            // Sets current button press to user sequence
```

```
                userSeq[testLevel-1] = BtnNum;

                // If button press is correct
                if (userSeq[testLevel-1] == compSeq[testLevel-1])
                {
                    // Continues iterating until user sequence is equal to
current level
                    // Prints current user sequence
                    testLevel++;
                    BtnNum = -1;
                    printf("Correct!\n");
                    printf("User sequence:");
                    for (i = 0; i < testLevel-1; i++) printLED(userSeq[i]);
                    printf("\n");

                    // If user sequence reached current level stop iteration
and go to next iteration
                    if (testLevel > userLevel)
                    {
                        FLAG = 0;
                        userLevel++;
                    }
                }
                // If button press is incorrect
                else
                {
                    // Ends the game
                    // Prints current user sequence
                    printf("Incorrect!\n");
                    printf("User sequence:");
                    for (i = 0; i < testLevel; i++) printLED(userSeq[i]);
                    printf("\n");
                    printf("Correct sequence:");
                    for (i = 0; i < winLevel; i++) printLED(compSeq[i]);
                    printf("\n");
                    printf("Game over!\n");
                    exit(0);
                }
            }
            delay(2000);
        }

        // Check for user win
        if (userLevel > winLevel)
        {
            printf("You win!\n");
            exit(0);
        }
    }
}
```