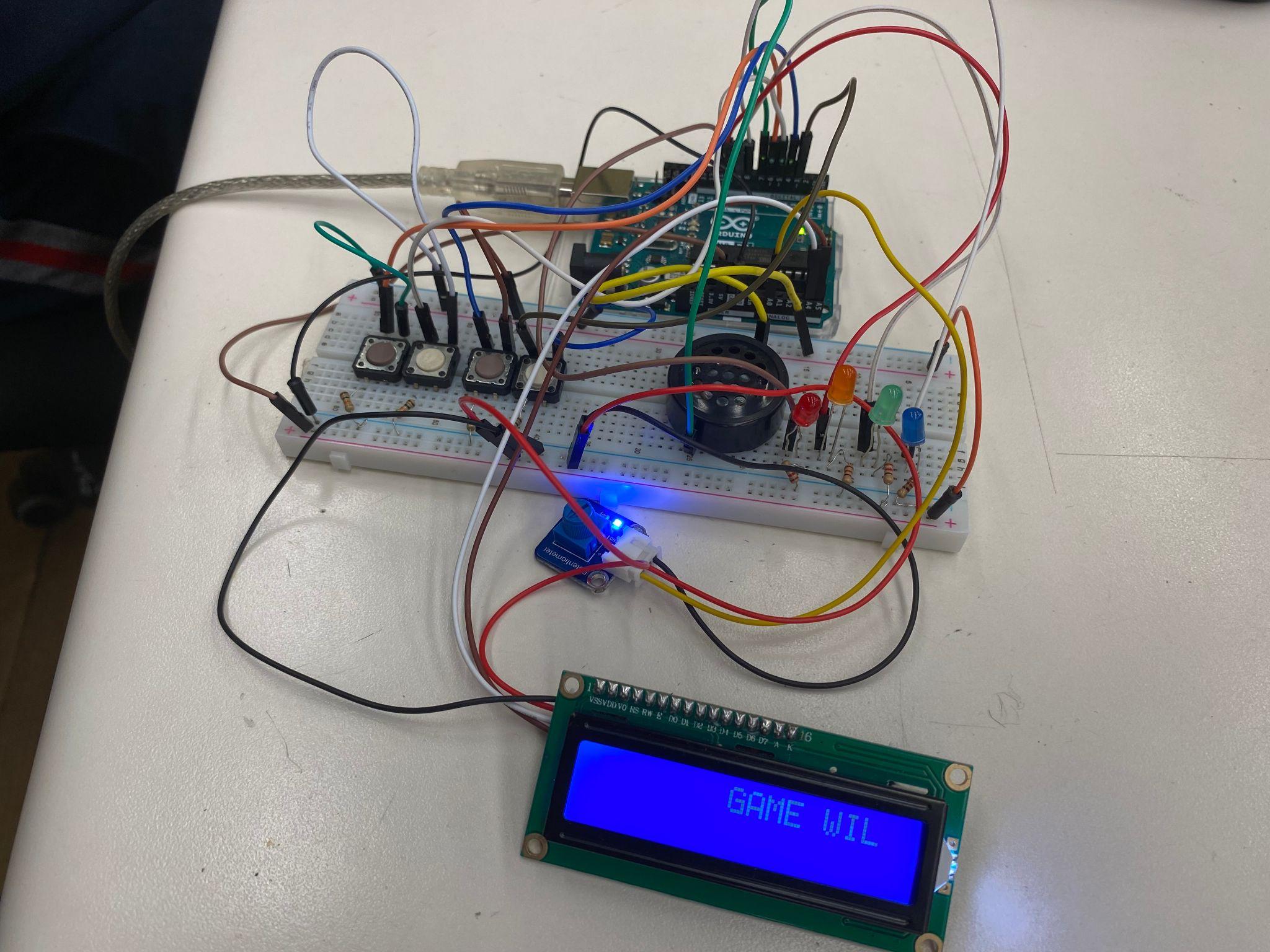
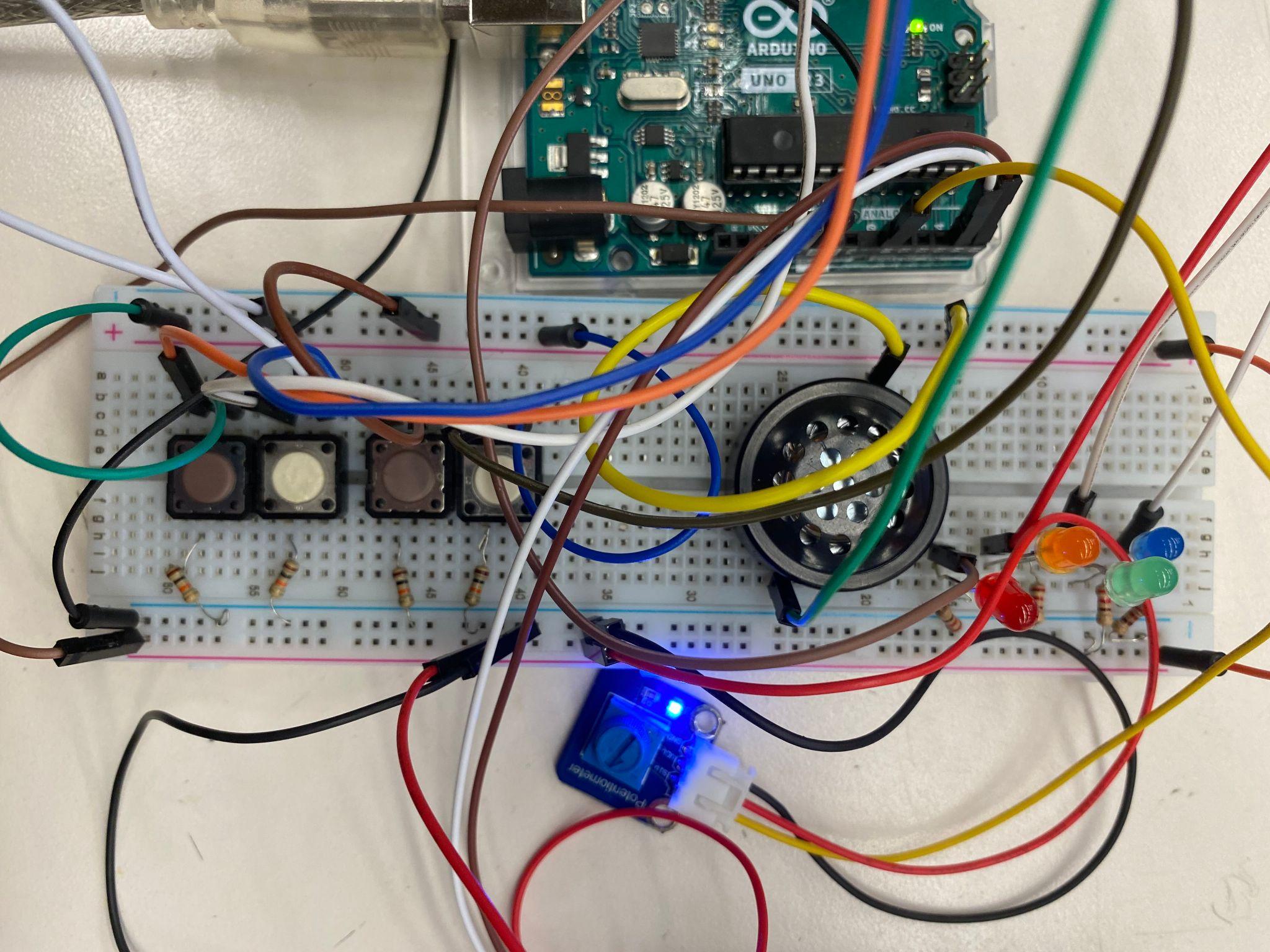
***SIMON   
SAYS***

***IN***

***ARDUINO***



**Introduction:**

**This is a simple Simon Says game that allows players to test their memory and reaction time!**

**The game consists of four push buttons, each corresponding to a different colour LED and a potentiometer to control the speed of the sequence display. The level is displayed on an LCD with sounds played.**

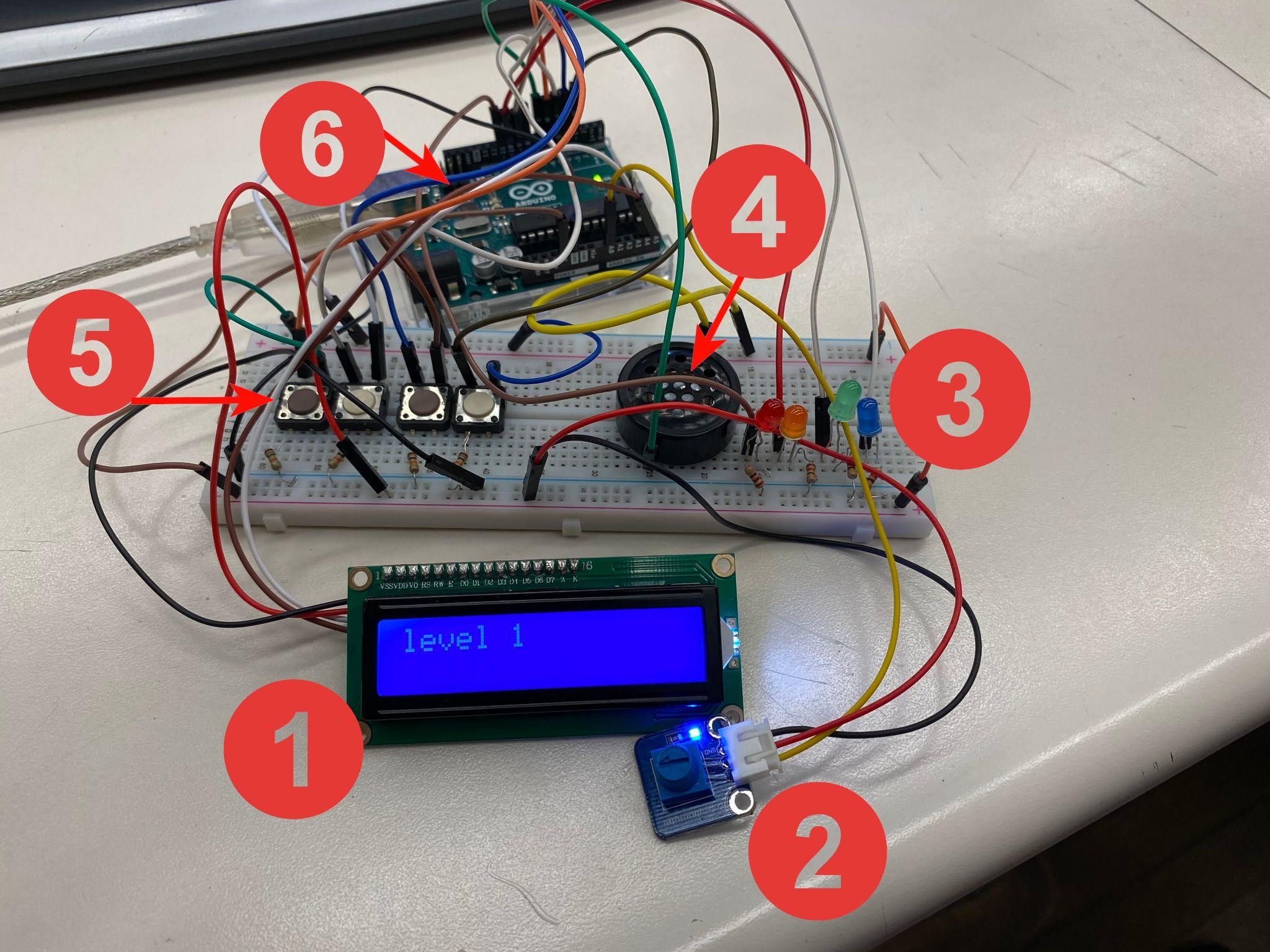
**Gameplay:**

The game will start with a random sequence of LED flashes and the player must repeat the sequence by pressing the buttons in the same order for a certain amount of time.

If the player correctly replays the sequence, the game will increase by one more button press, and the player must repeat the new extended sequence.

If the player makes a mistake, the game will reset playing a sound and “GAME OVER” will be displayed on the LCD monitor and the player will have to start over with a new random sequence.

**PARTS OF THE GAME:**

**1. LCD Monitor:** This is the primary display of the Simon Says game, displaying the current level, sequence, and victory/loss screens.

**2. Potentiometer:** This input device allows the player to adjust the difficulty of the game by altering the duration of the LED flashes.

**3. LEDs:**

These are used to display the sequence for the player to memorize and repeat.

**4. Speaker:**

This output device plays sounds corresponding to different events in the game, such as a correct or incorrect sequence input.

**5. Push Buttons:**

These buttons are used as the primary input method for the player, allowing them to input their guesses for the sequence. Each button corresponds to a specific LED in the sequence.

**6. Arduino Uno:**

The microcontroller that serves as the brains of the game, controlling the interactions between the various components. The Arduino processes input from the push buttons and responds accordingly using the code programmed into it.

**HOW TO PLAY**

1. Watch the sequence of button presses and corresponding LED flashes carefully.
2. Repeat the sequence by pressing the buttons in the same order.
3. If you successfully repeat the sequence, the game will add an additional button press to the sequence and the player must repeat the new, longer sequence.
4. If you make a mistake or take too long to repeat the sequence, the game will reset and the player will have to start over.

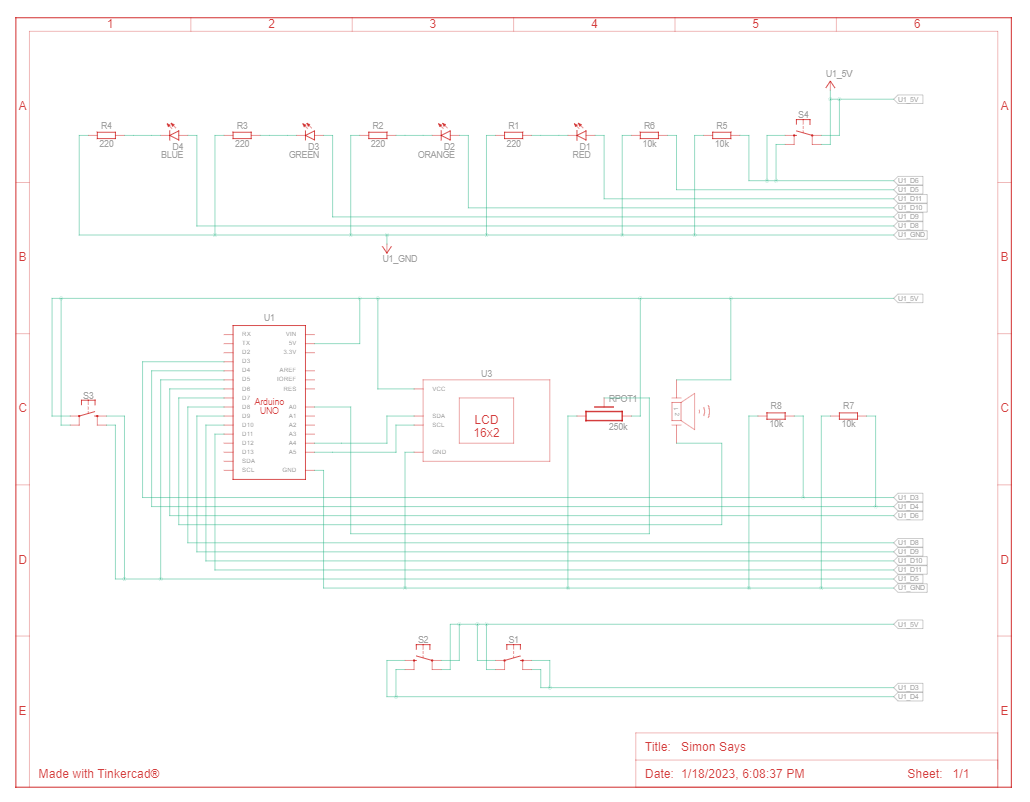
How to end the game:

* By reaching the final level, the LEDs will flash altogether 6 times and a “YOU WIN” screen will appear on the LCD monitor and the game will restart

**Good LUCK!!**

(please report any issues to Nehan Mohammed)

**Schematic:**



**ARDUINO CODE:**

// Nehan Mohammed Simon Says Jan 18/22

//This is a program for a Simon Says game. It uses an LCD

// screen, potentiometer, LEDs, speaker, and push buttons as the

//primary components.The program generates a random LED sequence, displays it,

// then waits for the player to input the correct sequence using the push

//buttons. It checks if the input is correct, plays appropriate sound and

// proceeds to the next level or ends the game. The program also includes

//a function to restart the game when it's over.

#include <LiquidCrystal\_I2C.h>//LCD library

#define num\_LEDs 4

#define num\_PBs 4

LiquidCrystal\_I2C lcd(0x27,16,2);//set dimensions of LCD

const int s = 7; //speaker

const int PBs [num\_PBs] = {3,4,5,6}; //row of PBs

const int LEDS [num\_LEDs] = {8,9,10,11}; //row of LEDs

char array1[]= "GAME WILL RESTART ";

const int PM = A0;//potentiometer to increase time limit

unsigned long t;//record time

int tim = 300;

int timeout = 10000;

int compArray [11];//number of levels

int level = 0;

int randomLED;

int timeInterval;

bool gameOver = 0;

void setup() {

for (int i=0; i < num\_LEDs; i++){

pinMode(LEDS[i],OUTPUT);}

for (int i=0; i < num\_PBs; i++){

pinMode(PBs[i],INPUT);

}

gameOver = 0;

lcd.init(); //initialize the LCD

lcd.backlight(); //open the backlight

pinMode(s,INPUT);

Serial.begin(9600);

}

void loop() {

randomSeed(analogRead(A3));

randomLED = random(num\_LEDs);//take a random LED from list

while(randomLED == compArray[level-1])//make sure previous LED is not the same

randomLED = random(num\_LEDs);

timeInterval = map(analogRead(PM), 0,1023, 100,500); //duration of LED flashes

lcd.printstr("level ");

lcd.print(level);

compArray[level] = randomLED; //assign the random LED to the level

for (int i = 0; i < level; i++){

digitalWrite(LEDS[compArray[i]], HIGH);//random LED on

delay(timeInterval);//delay changed by potentiometer

digitalWrite(LEDS[compArray[i]], LOW);//random LED off

delay(timeInterval);//delay changed by potentiometer

}

t = millis();

for (int i = 0; i < level; i++){ loop per level

while(digitalRead(PBs[compArray[i]]) == 0 ){

if (millis()-t >= timeout){ //end game if there is no input from user

lcd.clear();

lcd.printstr("GAME OVER");

lcd.setCursor(0,1);//set text to begin from bottom line

lcd.printstr("TIMEOUT");

tone(s,1245,100);//play wrong sound

delay(1500);

gg();

break;

}

for (int j= 0; j < 4; j++){//push button array

if(digitalRead(PBs[j]) == 1 && j != compArray[i]) {//wrong PB is pressed

lcd.clear();

lcd.printstr("GAME OVER");

lcd.setCursor(0,1);

lcd.printstr("WRONG BUTTON");

tone(s,900,100);//play wrong sound

delay(1500);

gg();

}

}

if(gameOver)//restart loop

break;

}

if(!gameOver){//correct PB was pressed

if (digitalRead(PBs[compArray[i]]) == 1){

lcd.clear();

lcd.printstr("correct");

tone(s,1245,200);//play correct sound

delay(500);

}

}

else{

lcd.clear();

lcd.print("loser");

delay(500);

}

gameOver = 0;//restart loop

}

lcd.clear();

level++;//add level when correct sequence is replayed

Serial.print("TIME INTERVAL ");

Serial.println(timeout);

for (int i = 0; i < level; i++){

Serial.println(compArray[i]);//cheat code

}

if (level == 11){//end game results

lcd.clear();

lcd.print("YOU WIN!!");

delay(100);

lcd.clear();

for (int j = 0; j < 6; j++) {//flash all LEDs 6 times

for (int i = 0; i < 5; i++) {

digitalWrite(LEDS[i], HIGH);

}

delay(300);

for (int i = 0; i < 5; i++) {

digitalWrite(LEDS[i], LOW);

}

delay(300);

}

level = 0;//restart game

}

}

void gg(){//game over screen

lcd.clear();

lcd.setCursor(15,0); // set the cursor to column 15, line 0

for (int positionCounter1 = 0; positionCounter1 < 20; positionCounter1++)

{

lcd.scrollDisplayLeft(); //Scrolls the contents of the display one space to the left.

lcd.print(array1[positionCounter1]); // Print a message to the LCD.

delay(tim);

tone(s,622,200);

}

lcd.clear(); //Clears the LCD screen and positions the cursor in the upper-left corner.

t = 0;//set time to zero

level = 0;//reset level

gameOver = 1;//end game to restart loop

}