**TITLE: SIMON’S MEMORY GAME**

**Introduction**

The Simon memory game, an iconic electronic toy introduced in 1978.In this digital electronics project, we have endeavored to recreate and modernize this classic game using electronic components and programming. The project combines aspects of both hardware and software engineering. The primary objective of this project is to design and implement a functional electronic replica of the Simon game along with some new features and different implementation that offers players the same cognitive challenge and enjoyment as the original while leveraging modern technology. By doing so, we aim to explore the integration of digital logic, microcontrollers, and user interface design to create an interactive gaming experience.

**About the game:**

The game consists of a console with four colored buttons, each associated with a unique tone. The game's objective is to replicate a sequence of button presses and corresponding tones that the console generates. The sequence starts with one button press and tone, and with each successful replication, the sequence becomes longer. Players must remember and repeat the sequence accurately to progress to the next level. The game tests and improves memory and pattern recognition skills, making it both entertaining and challenging. Making an incorrect button press or sequence repetition usually ends the game.

**Components Used**:

Solderless Breadboard,

Arduino Uno,

16×2 LCD Display,

I2C LCD Module,

Push Button x 4,

100R Resistor x 4,

Red LED,

Yellow LED,

Green LED,

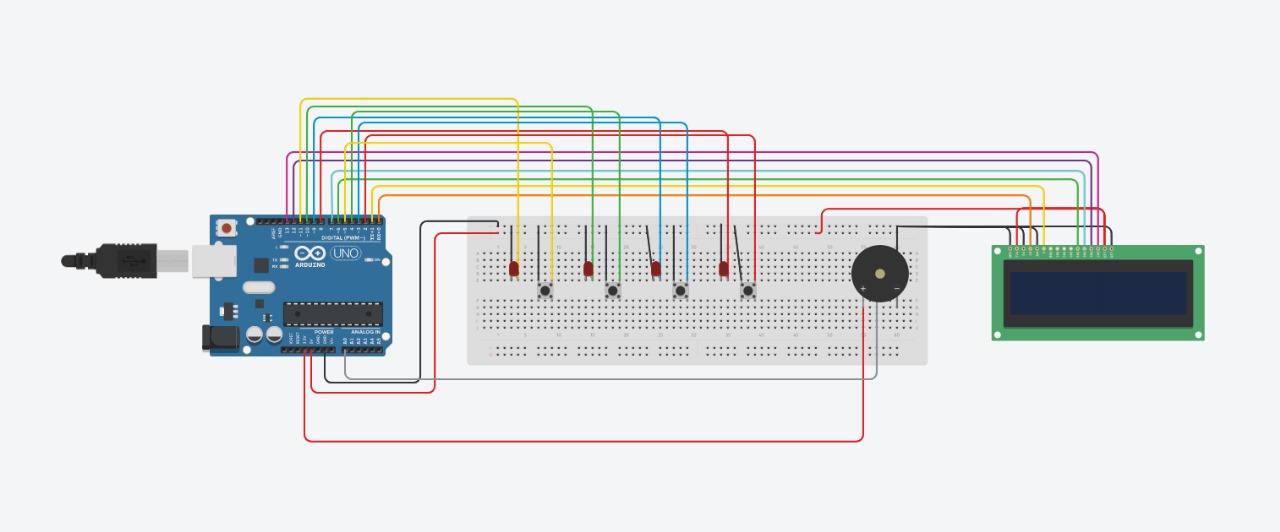
Blue LED,

Buzzer,

Hard Jumper Wire,

Male to Male Jumper Wires and Male to Female jumper Wires

**Circuit Design:**



**Software Implementation:**

Below is the code which we programmed into Arduino:

**Structure and logic of the code:**

#include <LiquidCrystal.h>

LiquidCrystal lcd(0, 1, 6, 7, 12, 13);

int buttons[4] = { 2, 3, 4, 5 }; // Define the button pins

int leds[4] = { 8, 9, 10, 11 }; // Define the LED pins

boolean button\_state[4] = { 0, 0, 0, 0 }; // Store button states

#define buzzer A0 // Define the buzzer pin

#define levelsInGame 10 // Define the number of levels in the game

int bt\_simonSaid[100]; // Store player button presses

int led\_simonSaid[100]; // Store the generated sequence

boolean lost;

int ip\_number, level, stage = 0; // Initialize game variables

void setup() {

lcd.begin(16, 2); // Initialize the LCD display

for (int i = 0; i <= 3; i++) {

pinMode(buttons[i], INPUT\_PULLUP); // Set button pins as input with pull-up resistors

pinMode(leds[i], OUTPUT); // Set LED pins as output

}

pinMode(buzzer, OUTPUT); // Set buzzer pin as output

// Display a welcome message on the LCD

lcd.setCursor(0, 0);

lcd.print(" Welcome To ");

lcd.setCursor(0, 1);

lcd.print("> Memory Game <");

delay(800);

lcd.clear(); // Clear the LCD

}

void loop() {

// The main game logic is implemented using a state machine (stage variable) and a switch statement.

// The comments below correspond to each case of the switch statement.

switch (stage) {

case 0:

// Initialization stage

// Prompt the player to press the Red button to start the game

// Once the Red button is pressed, the game transitions to the next stage (stage 1).

// Initialize game variables.

break;

case 1:

// Memorization stage

// The game generates a sequence of LED flashes and tones for the player to memorize.

// The sequence length increases with each level.

// After generating the sequence, the game transitions to the next stage (stage 2).

break;

case 2:

// Display a message indicating that the player should start playing.

// Once the player presses a button, the game transitions to the next stage (stage 3).

break;

case 3:

// Player input stage

// The player is expected to replicate the sequence by pressing the corresponding buttons.

// The player's button presses are stored in bt\_simonSaid.

// Once the player replicates the entire sequence, the game transitions to the verification stage (stage 4).

break;

case 4:

// Verification stage

// The game checks if the player's input matches the generated sequence.

// If there is a mismatch, the player loses and transitions to stage 5.

// If the input matches, the player wins and transitions to stage 6.

break;

case 5:

// Player loses

// Display a "You Lost" message and play a losing sound.

// After a brief delay, the game returns to the initialization stage (stage 0).

break;

case 6:

// Player wins

// Display a "You Win" message and play a winning sound.

// If the player completes all levels, display a "Congratulations" message.

// After a delay, the game either returns to the initialization stage or proceeds to the next level (stage 1).

break;

default:

break;

}

}

// Function to play a tone on the buzzer

void playBuzzer(int x) {

tone(buzzer, 650 + (x \* 100));

delay(200);

noTone(buzzer);

}

**Code:**

#include <LiquidCrystal.h>

LiquidCrystal lcd(0, 1, 6, 7, 12, 13);

int buttons[4] = { 2, 3, 4, 5 };

int leds[4] = { 8, 9, 10, 11 };

boolean button\_state[4] = { 0, 0, 0, 0 };

#define buzzer A0

#define levelsInGame 10

int bt\_simonSaid[100];

int led\_simonSaid[100];

boolean lost;

int ip\_number, level, stage = 0;

void setup() {

  lcd.begin(16, 2);

  for (int i = 0; i <= 3; i++) {

    pinMode(buttons[i], INPUT\_PULLUP);

    pinMode(leds[i], OUTPUT);

  }

  pinMode(buzzer, OUTPUT);

  lcd.setCursor(0, 0);

  lcd.print("   Welcome To   ");

  lcd.setCursor(0, 1);

  lcd.print("> Memory  Game <");

  delay(800);

  lcd.clear();

}

void loop() {  // the loop function runs over and over again forever

  switch (stage) {

    case 0:

      lcd.clear();

      lcd.setCursor(0, 0);

      lcd.print("Press Red Button");

      lcd.setCursor(0, 1);

      lcd.print("To Start");

      button\_state[0] = digitalRead(buttons[0]);

      while (button\_state[0] == HIGH) {

        button\_state[0] = digitalRead(buttons[0]);

      }

      level = 1, stage = 1, ip\_number = 1;

      delay(1000);

      break;

    case 1:

      lcd.clear();

      lcd.setCursor(0, 0);

      lcd.print("Level: ");

      lcd.setCursor(7, 0);

      lcd.print(level);

      lcd.setCursor(0, 1);

      lcd.print(" -- Memorize -- ");

      delay(100);

      for (int i = 1; i <= level; i++) {

        led\_simonSaid[i] = random(8,12);

        digitalWrite(led\_simonSaid[i], HIGH);

        playBuzzer(led\_simonSaid[i] - 7);

        digitalWrite(led\_simonSaid[i], LOW);

        delay(400);

      }

      delay(500);

      stage = 2;

      break;

    case 2:

      stage = 3;

      lcd.setCursor(0, 1);

      lcd.print("   -- Play --   ");

      break;

    case 3:

      for (int i = 0; i <= 3; i++) {

        button\_state[i] = digitalRead(buttons[i]);

        if (button\_state[i] == LOW) {

          bt\_simonSaid[ip\_number] = leds[i];

          digitalWrite(leds[i], HIGH);

          playBuzzer(i + 1);

          while (button\_state[i] == LOW) {

            button\_state[i] = digitalRead(buttons[i]);

          }

          delay(50);

          digitalWrite(leds[i], LOW);

          ip\_number++;

          if (ip\_number - 1 == level) {

            ip\_number = 1;

            stage = 4;

            break;

          }

        }

      }

      delay(10);

      break;

    case 4:

      lcd.clear();

      lcd.setCursor(0, 1);

      lcd.print("  Verification  ");

      delay(1000);

      for (int i = 1; i <= level; i++) {

        if (led\_simonSaid[i] != bt\_simonSaid[i]) {

          lost = 1;

          break;

        }

      }

      if (lost == 1) stage = 5;

      else stage = 6;

      break;

    case 5:

      lcd.clear();

      lcd.setCursor(0, 1);

      lcd.print(" !! You Lost !! ");

      tone(buzzer, 350);

      for (int i = 0; i <= 3; i++) {

        digitalWrite(leds[i], HIGH);

        playBuzzer(i + 1);

        delay(1);

        digitalWrite(leds[i], LOW);

      }

      //delay(1000);

      lcd.setCursor(0, 1);

      lcd.print("!! GAME  OVER !!");

      // delay(1000);

      for (int i = 0; i <= 3; i++) {

        digitalWrite(leds[i], HIGH);

      }

      playBuzzer(1);

      delay(1000);

      for (int i = 0; i <= 3; i++) {

        digitalWrite(leds[i], LOW);

      }

      level = 1, stage = 0, lost = 0;

      break;

    case 6:

      lcd.clear();

      lcd.setCursor(0, 1);

      lcd.print(" \*\* You  Win \*\* ");

      for (int i = 0; i <= 1; i++) {

        playBuzzer(i + 1);

      }

      delay(1000);

      if (level == levelsInGame) {

        lcd.setCursor(0, 0);

        lcd.print("Congratulation");

        lcd.setCursor(0, 1);

        lcd.print(" Level Complete");

        for (int i = 0;i<=5;i++){

          if(i%2==0){

            digitalWrite(leds[0],HIGH);

            digitalWrite(leds[2],HIGH);

            playBuzzer(2);

            digitalWrite(leds[0],LOW);

            digitalWrite(leds[2],LOW);

            // delay(1000);

          }else{

            digitalWrite(leds[1],HIGH);

            digitalWrite(leds[3],HIGH);

            playBuzzer(3);

            digitalWrite(leds[1],LOW);

            digitalWrite(leds[3],LOW);

            // delay(1000);

          }

        }

        delay(2500);

        lcd.clear();

        level = 1;

        stage = 0;

      } else {

        if (level < levelsInGame) level++;

        stage = 1;

      }

      break;

    default:

      break;

  }

}

void playBuzzer(int x) {

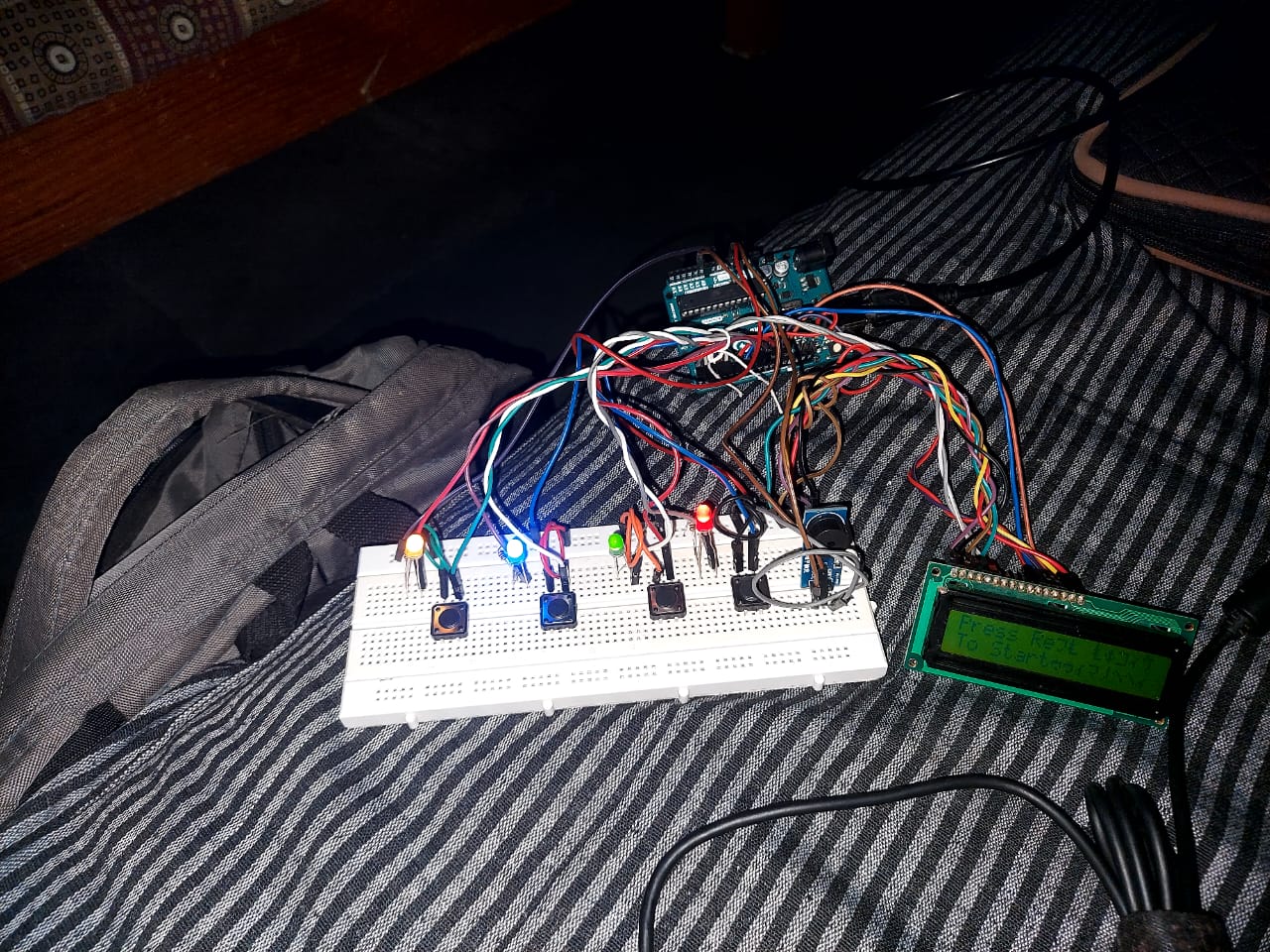
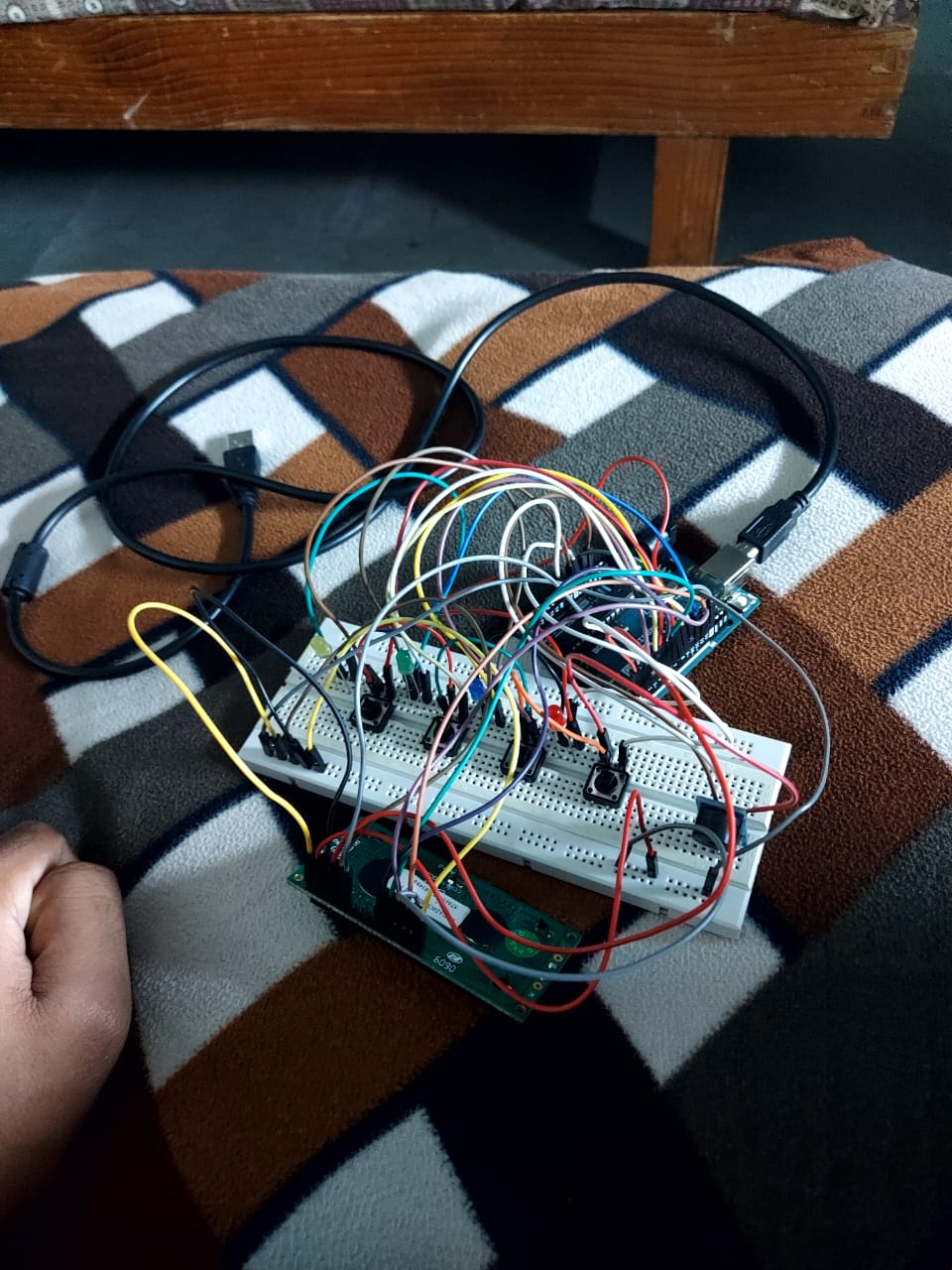
  tone(buzzer, 650 + (x \* 100));

  delay(200);

  noTone(buzzer);

}

**Related Photos:**

****