Introduction

Welcome to the documentation for the Shadow Footworks Project! This project aims to provide a real-time scoring system with sensors that accurately track and display player footwork speed, offering personalized feedback based on player performance. While inspired by the traditional 6 corner footwork practice, this project goes beyond the basics, providing additional features to enhance your training experience.

Features

- **Scoring System:** The project incorporates sensors that capture footwork movements and calculate the player's speed. The real-time scoring system allows you to monitor your footwork performance instantly.
- **Multi-Level Practice:** The system offers three levels of difficulty tailored to different player skill levels: Beginner, Intermediate, and Advanced. This allows you to challenge yourself and progressively improve your footwork skills.
- **Review System:** After each practice session, the project provides a review system that evaluates your performance based on the selected level. This feedback helps you understand how well you performed and identifies areas for improvement.

Why is it Required?

Traditional footwork practice in badminton often relies on another person to track time and measure speed, limiting the ability to practice independently. This project addresses this limitation by providing a self-contained system that allows players to practice footwork anytime and anywhere. By eliminating the need for a dedicated timer or assistance from others, players gain independence and flexibility in their training routine.

Hardware Requirements

To build the Badminton Footwork Practice Project, you will need the following components:

- **Arduino Uno R3** (Quantity: 1)
    - The Arduino Uno R3 board serves as the main microcontroller for the project, providing the necessary processing power and I/O capabilities. You can purchase it from the official Arduino Store or various online electronics retailers. Here is a link to buy it from the Arduino Store: Arduino Store - Arduino Uno R3
- **Red LED** (Quantity: 12)
    - The red LEDs are used to indicate various feedback signals during the practice sessions. You can find red LEDs in electronic component stores or online retailers. Here is an example link to buy red LEDs from a popular online retailer: Red LEDs on Adafruit
- **Pushbutton** (Quantity: 10)
    - Pushbuttons are utilized to interact with the system, allowing you to navigate through different levels and initiate actions. Pushbuttons can be purchased from electronic component stores or online retailers. Here is an example link to buy push buttons from a popular online retailer: Pushbuttons on SparkFun
- **PCF8574-based 32 LCD 16 x 2** (Quantity: 1)
    - The PCF8574-based LCD module with a 16x2 character display is used to show the real-time scoring and feedback. You can find this module in electronic component stores

or online retailers. Here is an example link to buy the PCF8574-based LCD module from an online retailer: PCF8574-based LCD Module on Amazon

- **8 Pin Header** (Quantity: 1)
  - The 8 pin header is used for convenient connections and mounting of the LCD module. You can find 8 pin headers in electronic component stores or online retailers. Here is an example link to buy 8 pin headers from a popular online retailer: 8 Pin Headers on Digi-Key
- **Piezo Buzzer** (Quantity: 1)
  - The piezo buzzer generates audio feedback signals to provide additional cues during the practice sessions. Piezo buzzers are available in electronic component stores or online retailers. Here is an example link to buy a piezo buzzer from a popular online retailer: Piezo Buzzers on Adafruit

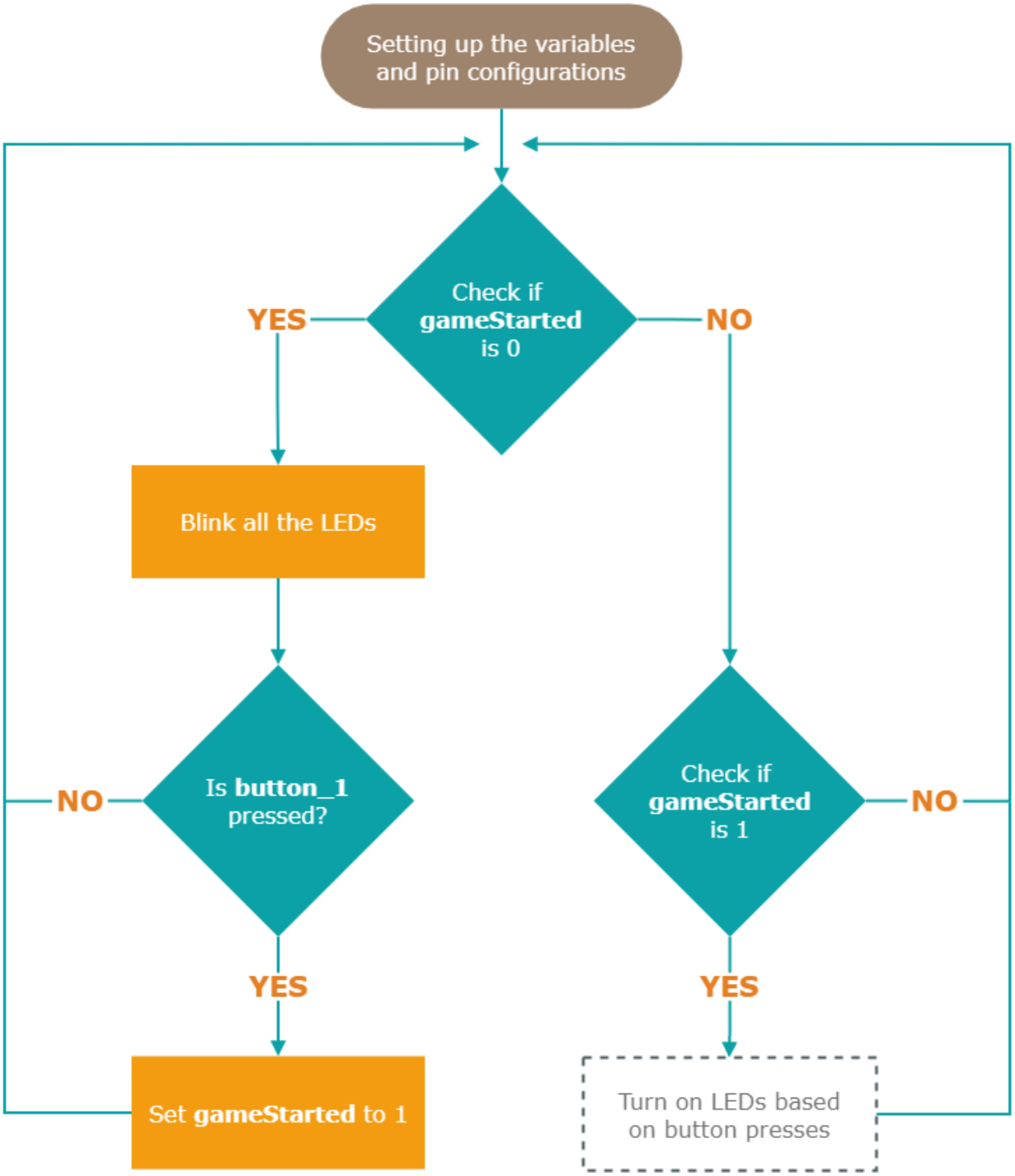
Please note that the provided links are for reference purposes, and you can explore different stores and retailers to find the components that suit your preferences and availability.

Name	Quantity	Component
U1	1	Arduino Uno R3
D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12	12	Red LED
S2, S3, S4, S5, S6, S7, S8, S9, S10, S1	10	Pushbutton
U2	1	PCF8574-based, 32 LCD 16 x 2 (I2C)
JP1	1	8 Pin Header
PIEZO2	1	Piezo

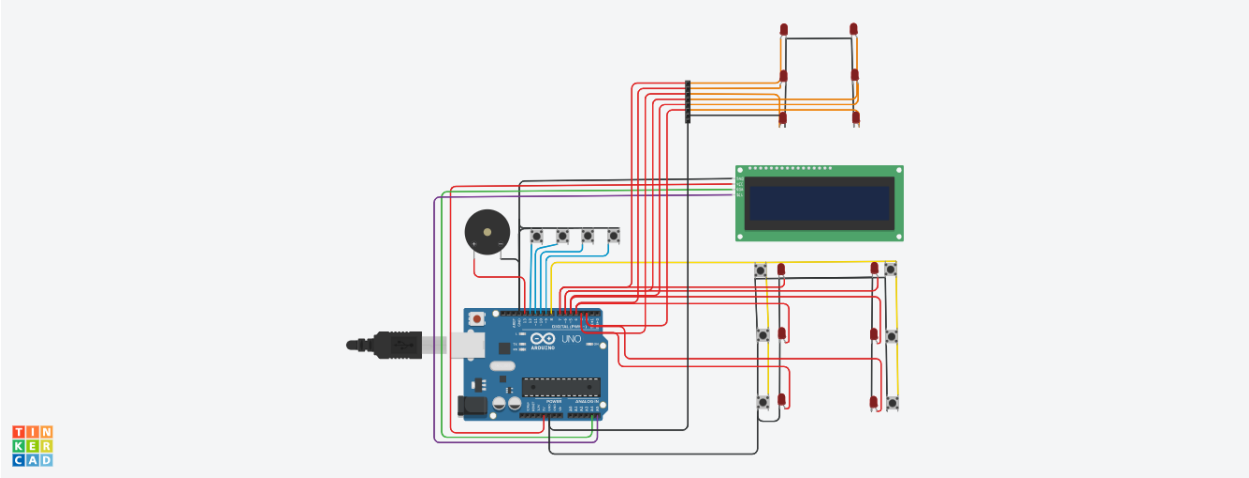
Software Requirements:

- Arduino IDE
- Wire.h Library: The code requires the Wire library to communicate with devices connected via I2C (Inter-Integrated Circuit) protocol. The library should be included and properly configured.
- LiquidCrystal_I2C Library: The code uses the LiquidCrystal_I2C library to control the I2C-based LCD display. The library should be included and properly configured.

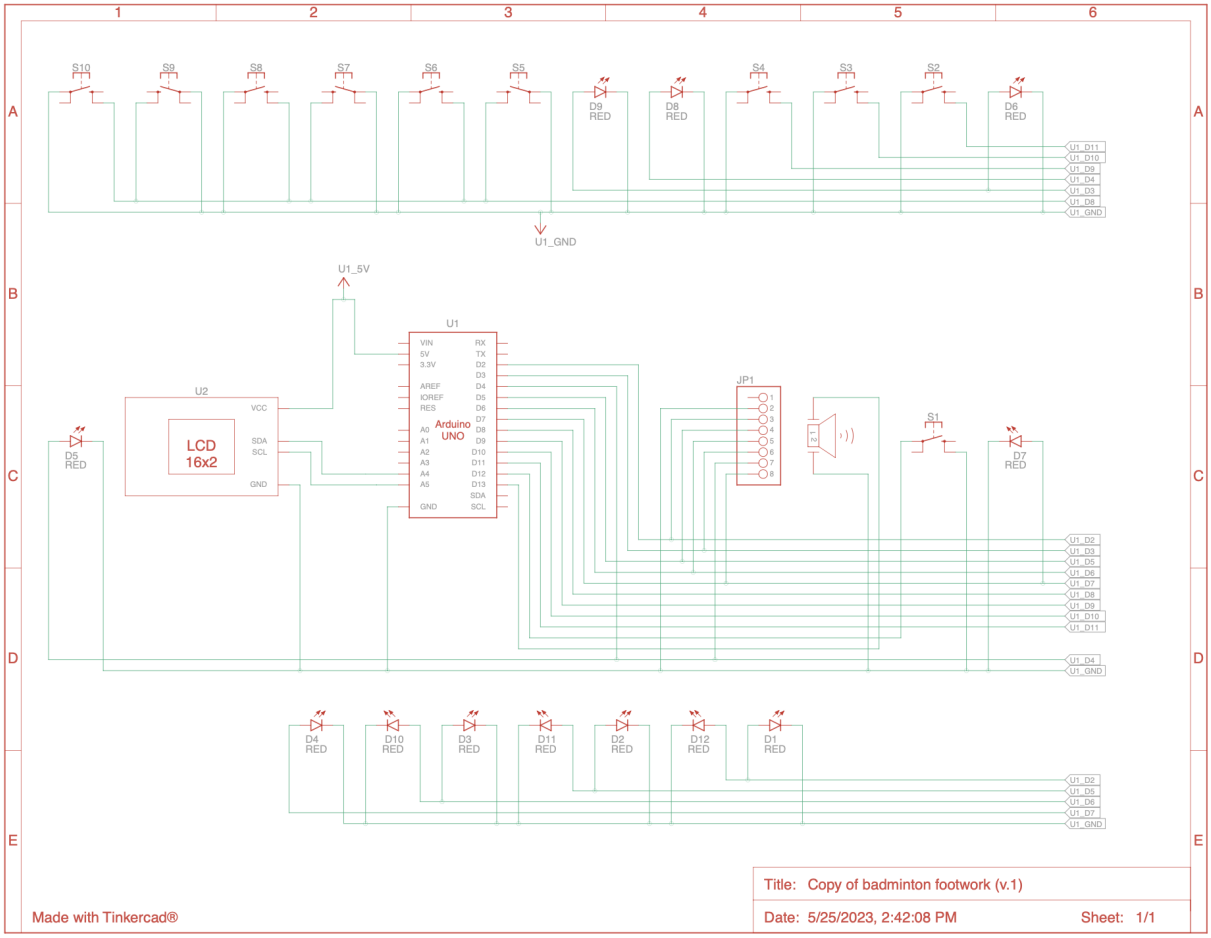
System Architecture:


Setting up the variables and pin configurations
Check if gameStarted is 0
YES
NO
Blink all the LEDs
Is button_1 pressed?
NO
YES
Set gameStarted to 1
Check if gameStarted is 1
NO
YES
Turn on LEDs based on button presses

Circuit Design :



Schematic View:



Code Implementation :

1. Open the Arduino IDE or your preferred Arduino development environment.
2. 2. Create a new sketch and copy the entire code into the sketch.
3. Make sure you have the required libraries installed. This code uses the "Wire" and "LiquidCrystal_I2C" libraries. You can install them by going to "Sketch" -> "Include Library" -> "Manage Libraries" and then searching for the libraries by name.
4. Connect your Arduino board to your computer using a USB cable.
5. Select the appropriate board and port from the "Tools" menu in the Arduino IDE.
6. Click the "Upload" button to upload the code to your Arduino board.
7. After the code is uploaded successfully, you should see the LCD display showing the initial messages.
8. Use the buttons connected to the Arduino (Button1, Button2, Button3) to select the desired level (2 mins, 5 mins, or 10 mins).
9. Once a level is selected, the program will start a countdown timer on the LCD display.
10. When the timer reaches zero, the program will display the score based on the number of times the button (BUTTON) was pressed during the timer countdown.
11. The program will then wait for another level selection or a reset button (Reset) press to start the process again.
12. Make sure you have the necessary hardware components (such as LEDs, buttons, buzzer, and LCD) connected to the Arduino board as specified in the code.

Below find the Code:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define LED1 2
#define LED2 3
#define LED3 4
#define LED4 5
#define LED5 6
#define LED6 7
#define BUTTON 8
#define Button1 11
#define Button2 10
#define Button3 9
#define Reset 12
#define Buzzer 13
// Initialize the library with the I2C address of the display
LiquidCrystal_I2C lcd(0x27, 16, 2);
//---------------- store the custom characters in arrays ----------------//
byte hrt[8] = {
```

```
  B00000,
  B01010,
  B11111,
  B11111,
  B11111,
  B01110,
  B00100,
  B00000
};
byte smil[8] = {
  B00000,
  B00000,
  B01010,
  B00000,
  B10001,
  B01110,
  B00000,
  B00000
};
byte mansmil[8] =
  {
    B00000,
    B00000,
    B01010,
    B00000,
    B10001,
    B01110,
    B00000,
    B00000
  };
byte man2hrt[8] =
  {
    B00000,
    B01010,
    B11111,
    B11111,
    B11111,
    B01110,
    B00100,
    B00000
  };
byte average[8] =
  {
    0b00000,
    0b10001,
    0b01010,
    0b10001,
    0b00100,
    0b01110,
```

```
    0b10001,
    0b00000
  };
byte Q1[8] = {
  0b00000,
  0b10101,
  0b10101,
  0b01010,
  0b01010,
  0b00101,
  0b00101,
  0b00010
};
byte Q2[8] = {
  0b00000,
  0b01010,
  0b01010,
  0b10101,
  0b10101,
  0b01010,
  0b01010,
  0b10001
};
byte Q3[8] = {
  0b00000,
  0b10101,
  0b10101,
  0b01010,
  0b01010,
  0b10100,
  0b10100,
  0b01000
};
byte Q4[8] = {
  0b00010,
  0b00001,
  0b00001,
  0b00000,
  0b00000,
  0b00000,
  0b00000,
  0b00000
};
byte Q5[8] = {
  0b01010,
  0b00100,
  0b01010,
  0b11111,
  0b10001,
```

```
  0b11111,
  0b11111,
  0b01110
};
byte Q6[8] = {
  0b01000,
  0b10000,
  0b10000,
  0b00000,
  0b00000,
  0b00000,
  0b00000,
  0b00000
};
byte smile[8] = {
  B00000,
  B00000,
  B01010,
  B00000,
  B10001,
  B01110,
  B00000,
  B00000
};
int ledPins[] = { LED1, LED2, LED3, LED4, LED5, LED6 };
int selectedLed = -1;
int count = 0;
int buttonState = 0;
int prev_buttonState = 0;
void setup() {
  lcd.begin(16, 2);
  lcd.init();
  lcd.backlight();
  //---- create custom characters ----//
  lcd.createChar(1, hrt);
  lcd.createChar(2, smil);
  lcd.createChar(3, mansmil);
  lcd.createChar(11, man2hrt);
  lcd.createChar(4, average);
  lcd.createChar(5, Q1);
  lcd.createChar(6, Q2);
  lcd.createChar(7, Q3);
  lcd.createChar(8, Q4);
  lcd.createChar(9, Q5);
  lcd.createChar(10, Q6);
  lcd.createChar(12, smile);
  lcd.setCursor(0, 0);
  lcd.print("     SHADOW");
  lcd.setCursor(3, 1);
```

```
  lcd.print("Footwork's");
  lcd.setCursor(0, 0);
  lcd.write(5);
  lcd.setCursor(1, 0);
  lcd.write(6);
  lcd.setCursor(2, 0);
  lcd.write(7);
  lcd.setCursor(0, 1);
  lcd.write(8);
  lcd.setCursor(1, 1);
  lcd.write(9);
  lcd.setCursor(2, 1);
  lcd.write(10);
  lcd.setCursor(13, 0);
  lcd.write(5);
  lcd.setCursor(14, 0);
  lcd.write(6);
  lcd.setCursor(15, 0);
  lcd.write(7);
  lcd.setCursor(13, 1);
  lcd.write(8);
  lcd.setCursor(14, 1);
  lcd.write(9);
  lcd.setCursor(15, 1);
  lcd.write(10);
  delay(5000);
  Serial.begin(9600);  // initialize serial communication
  // initialize LED pins as outputs
  for (int i = 0; i < 6; i++) {
    pinMode(ledPins[i], OUTPUT);
  }
  pinMode(BUTTON, INPUT_PULLUP);   // initialize the button pin as an input with pull-up resistor
  pinMode(Button1, INPUT_PULLUP);  //initializing buttons for 2 min mode
  pinMode(Button2, INPUT_PULLUP);  //intializing button for 5 min mode
  pinMode(Button3, INPUT_PULLUP);  //initializing button for 10 min mode
  pinMode(Reset, INPUT_PULLUP);    // initialize the reset button pin as an input with pull-up resistor
  pinMode(Buzzer, OUTPUT);         // initialize the buzzer pin as an output
}
void RandomLED() {
  buttonState = digitalRead(BUTTON);
  if (selectedLed == -1) {                  // if no LED is selected
    int randomLed = random(0, 6);           // generate a random number between 0 and 5
    digitalWrite(ledPins[randomLed], HIGH);  // turn on the selected LED
    selectedLed = randomLed;                // store the index of the selected LED
  }
  if (prev_buttonState != buttonState) {
    if (!digitalRead(BUTTON)) {  // if the button is pressed
      tone(Buzzer, 2600,200);  // generate a tone of 1000Hz on the buzzer
      count++;
```

```
      //Serial.print(count);
      digitalWrite(ledPins[selectedLed], LOW);  // turn off the selected LED
      selectedLed = -1;                         // reset the selected LED
      // print the elapsed time in milliseconds
      // Serial.print("Button pressed after ");
      // Serial.print(elapsedTime);
      //Serial.println(" milliseconds");
    } else {
      noTone(Buzzer);  // stop the tone on the buzzer
    }
  }
  prev_buttonState = buttonState;
}
void loop() {
  bool resetButtonPressed = false;  // Flag to indicate if reset button is pressed
  selectedLed = -1;  // reset the selected LED
  count = 0;
  lcd.clear();
  lcd.print("Select Level");
  lcd.setCursor(0, 1);
  lcd.print("LVL1 /LVL2 /LVL3");
  while (digitalRead(Button1) || digitalRead(Button2) || digitalRead(Button3)) {
    if (!digitalRead(Button1))  //2 mins
    {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print(" Ready ");
      tone(Buzzer,388,1000);
      delay(1500);
      lcd.print("  Steady");
      tone(Buzzer,388,1000);
      delay(1500);
      lcd.setCursor(0, 1);
      lcd.print("    !!GO!!");
      tone(Buzzer,320,1500);
      delay(400);
      lcd.clear();
      lcd.print("    Level 1");
      lcd.setCursor(0, 0);
      lcd.write(5);
      lcd.setCursor(1, 0);
      lcd.write(6);
      lcd.setCursor(2, 0);
      lcd.write(7);
      lcd.setCursor(0, 1);
      lcd.write(8);
      lcd.setCursor(1, 1);
      lcd.write(9);
      lcd.setCursor(2, 1);
```

```
lcd.write(10);
lcd.setCursor(13, 0);
lcd.write(5);
lcd.setCursor(14, 0);
lcd.write(6);
lcd.setCursor(15, 0);
lcd.write(7);
lcd.setCursor(13, 1);
lcd.write(8);
lcd.setCursor(14, 1);
lcd.write(9);
lcd.setCursor(15, 1);
lcd.write(10);
//write code for adding 2 mins timer and displaying it on lcd
unsigned long startTime = millis();
unsigned long elapsedTime = 0;              // initialize elapsedTime to 0
const unsigned long timerDuration = 120000;  // 2 minutes in milliseconds
unsigned long remainingTime = timerDuration - elapsedTime;
while (millis() - startTime < 120000) {  // currently 10 seconds
  // Update the elapsedTime variable
  elapsedTime = millis() - startTime;
  remainingTime = timerDuration - elapsedTime;  // update the remainingTime variable
  // Calculate the minutes and seconds
  int minutes = remainingTime / 60000;
  int seconds = (remainingTime % 60000) / 1000;
  // Display the remaining time on the LCD
  lcd.setCursor(6, 1);
  if (minutes < 10) {
    lcd.print("0");
  }
  lcd.print(minutes);
  lcd.print(":");
  if (seconds < 10) {
    lcd.print("0");
  }
  lcd.print(seconds);
  // Check if the timer has reached zero
  if (remainingTime <= 0) {
    lcd.setCursor(0, 1);
    lcd.print("Time's up!");
    // Add additional code here for any actions to be taken after the timer expires
  }
  RandomLED();
  //Reset Button
  if (digitalRead(Reset) == LOW) {
    resetButtonPressed = true;  // Set the resetButtonPressed flag
    // Turn off all LEDs
for (int i = 0; i < 6; i++) {
  digitalWrite(ledPins[i], LOW);
```

```
  }
     loop();                    // Break the loop and exit the program
    }
    delay(50);
  }
  // Turn off all LEDs
  for (int i = 0; i < 6; i++) {
    digitalWrite(ledPins[i], LOW);
  }
  lcd.clear();
  lcd.setCursor(0, 0);
  if (count >= 36)  // 75% of 48 i.e.
  {
    lcd.print("Awesome  ");
    lcd.write(11);  //heart
  } else if (count >= 21) {
    lcd.print("Great Going  ");
    lcd.write(12);  //smile
  } else {
    lcd.print("Try harder  ");
    lcd.write(4);  //average
  }
  lcd.setCursor(0, 1);
  lcd.print("Score = ");
  lcd.print(count);
  delay(15000);
  break;
}
else if (!digitalRead(Button2))  // 5 min
{
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(" Ready ");
  tone(Buzzer,388,1000);
  delay(1500);
  lcd.print("  Steady");
  tone(Buzzer,388,1000);
  delay(1500);
  lcd.setCursor(0, 1);
  lcd.print("    !!GO!!");
  tone(Buzzer,320,1500);
  delay(400);
  lcd.clear();
  lcd.clear();
  lcd.print("    LEVEL 2");
  lcd.setCursor(0, 0);
  lcd.write(5);
  lcd.setCursor(1, 0);
  lcd.write(6);
```

```
lcd.setCursor(2, 0);
lcd.write(7);
lcd.setCursor(0, 1);
lcd.write(8);
lcd.setCursor(1, 1);
lcd.write(9);
lcd.setCursor(2, 1);
lcd.write(10);
lcd.setCursor(13, 0);
lcd.write(5);
lcd.setCursor(14, 0);
lcd.write(6);
lcd.setCursor(15, 0);
lcd.write(7);
lcd.setCursor(13, 1);
lcd.write(8);
lcd.setCursor(14, 1);
lcd.write(9);
lcd.setCursor(15, 1);
lcd.write(10);
//write code for adding 5 mins timer and displaying it on lcd
unsigned long startTime = millis();
unsigned long elapsedTime = 0;              // initialize elapsedTime to 0
const unsigned long timerDuration = 300000;  // 5 minutes in milliseconds
unsigned long remainingTime = timerDuration - elapsedTime;
while (millis() - startTime < 300000) {
  // Update the elapsedTime variable
  elapsedTime = millis() - startTime;
  remainingTime = timerDuration - elapsedTime;  // update the remainingTime variable
  // Calculate the minutes and seconds
  int minutes = remainingTime / 60000;
  int seconds = (remainingTime % 60000) / 1000;
  // Display the remaining time on the LCD
  lcd.setCursor(6, 1);
  if (minutes < 10) {
    lcd.print("0");
  }
  lcd.print(minutes);
  lcd.print(":");
  if (seconds < 10) {
    lcd.print("0");
  }
  lcd.print(seconds);
  // Check if the timer has reached zero
  if (remainingTime <= 0) {
    lcd.setCursor(0, 1);
    lcd.print("Time's up!");
    // Add additional code here for any actions to be taken after the timer expires
  }
```

```
    RandomLED();
    //Reset Button
    if (digitalRead(Reset) == LOW) {
      resetButtonPressed = true;  // Set the resetButtonPressed flag
      // Turn off all LEDs
   for (int i = 0; i < 6; i++) {
    digitalWrite(ledPins[i], LOW);
   }
      loop();                    // Break the loop and exit the program
    }
    delay(50);
   }
   // Turn off all LEDs
   for (int i = 0; i < 6; i++) {
    digitalWrite(ledPins[i], LOW);
   }
   lcd.clear();
   lcd.setCursor(0, 0);
   if (count >= 67)  // 75% of 90 i.e.
   {
    lcd.print("Awesome  ");
    lcd.write(11);  //smile
   } else if (count >= 49) {
    lcd.print("Great Going  ");
    lcd.write(3);  //man
   } else {
    lcd.print("Try harder  ");
    lcd.write(4);  //average
   }
   lcd.setCursor(0, 1);
   lcd.print("Score = ");
   lcd.print(count);
   delay(10000);
   break;
  }
  else if (!digitalRead(Button3))  //10 min
  {
   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print(" Ready ");
   tone(Buzzer,388,1000);
   delay(1500);
   lcd.print("  Steady");
   tone(Buzzer,388,1000);
   delay(1500);
   lcd.setCursor(0, 1);
   lcd.print("    !!GO!!");
   tone(Buzzer,320,1500);
   delay(400);
```

```
lcd.clear();
lcd.clear();
lcd.print("    LEVEL 3");
lcd.setCursor(0, 0);
lcd.write(5);
lcd.setCursor(1, 0);
lcd.write(6);
lcd.setCursor(2, 0);
lcd.write(7);
lcd.setCursor(0, 1);
lcd.write(8);
lcd.setCursor(1, 1);
lcd.write(9);
lcd.setCursor(2, 1);
lcd.write(10);
lcd.setCursor(13, 0);
lcd.write(5);
lcd.setCursor(14, 0);
lcd.write(6);
lcd.setCursor(15, 0);
lcd.write(7);
lcd.setCursor(13, 1);
lcd.write(8);
lcd.setCursor(14, 1);
lcd.write(9);
lcd.setCursor(15, 1);
lcd.write(10);
//write code for adding 10 mins timer and displaying it on lcd
unsigned long startTime = millis();
unsigned long elapsedTime = 0;              // initialize elapsedTime to 0
const unsigned long timerDuration = 600000;  // 10 minutes in milliseconds
unsigned long remainingTime = timerDuration - elapsedTime;
while (millis() - startTime < 600000) {
  // Update the elapsedTime variable
  elapsedTime = millis() - startTime;
  remainingTime = timerDuration - elapsedTime;  // update the remainingTime variable
  // Calculate the minutes and seconds
  int minutes = remainingTime / 60000;
  int seconds = (remainingTime % 60000) / 1000;
  // Display the remaining time on the LCD
  lcd.setCursor(6, 1);
  if (minutes < 10) {
    lcd.print("0");
  }
  lcd.print(minutes);
  lcd.print(":");
  if (seconds < 10) {
    lcd.print("0");
  }
```

```
      lcd.print(seconds);
      // Check if the timer has reached zero
      if (remainingTime <= 0) {
        lcd.setCursor(0, 1);
        lcd.print("Time's up!");
        // Add additional code here for any actions to be taken after the timer expires
      }
      RandomLED();
      //Reset Button
      if (digitalRead(Reset) == LOW) {
        resetButtonPressed = true;  // Set the resetButtonPressed flag
        // Turn off all LEDs
    for (int i = 0; i < 6; i++) {
      digitalWrite(ledPins[i], LOW);
    }
        loop();                   // Break the loop and exit the program
      }
      delay(50);
    }
    // Turn off all LEDs
    for (int i = 0; i < 6; i++) {
      digitalWrite(ledPins[i], LOW);
    }
    lcd.clear();
    lcd.setCursor(0, 0);
    if (count >= 90)  // 75% of 120 i.e.
    {
      lcd.print("Awesome  ");
      lcd.write(11);  //smile
    } else if (count >= 67) {
      lcd.print("Great Going  ");
      lcd.write(3);  //man
    } else {
      lcd.print("Try harder  ");
      lcd.write(4);  //average
    }
    lcd.setCursor(0, 1);
    lcd.print("Score = ");
    lcd.print(count);
    delay(10000);
    break;
   }
  }
}
```

Usage Instructions:

1. Run the program:
    1. Once the code is successfully uploaded to the Arduino board, the program will start running automatically.
2. LCD Display:
    1. The LCD display will show the startup message "SHADOW Footwork's" with custom characters.
    2. After 5 seconds, the display will show the "Select Level" message with three options: LVL1, LVL2, LVL3.
3. Select the desired level:
    1. Press one of the three buttons (Button1, Button2, Button3) to choose the desired level:
    2. Button1: Selects Level 1 (2 minutes).
    3. Button2: Selects Level 2 (5 minutes).
    4. Button3: Selects Level 3 (10 minutes).
4. Timer and LED Control:
    1. Depending on the selected level, the program will start a timer for the specified duration (2, 5, or 10 minutes).
    2. During the timer countdown, the LCD display will show the remaining time in minutes and seconds.
    3. Random LEDs will turn on and off during the countdown.
    4. Press the BUTTON to turn off the currently lit LED and increment the count.
    5. If all LEDs are turned off before the timer ends, a buzzer will sound briefly.
    6. If the timer reaches zero, the LCD display will show "Time's up!"
5. Reset:
    1. If you want to start a new session or reset the game, press the Reset button (connected to digital pin 12).
    2. The program will restart, and you can select a new level.
6. Repeat Steps 3 to 5 for different levels or game sessions.
7. Enjoy the game and improve your footwork skills!

FAQs and Troubleshooting :

Question: How do I change the I2C address for the LCD display?
Ans: If your LCD display has a different I2C address, modify the following line in the code to match your LCD's address:

“LiquidCrystal_I2C lcd(0x27, 16, 2);”

Troubleshooting:

1. Verify hardware connections:

- Check that all components are connected properly, and the wiring matches the instructions.
- Ensure that the LEDs, buttons, buzzer, and LCD display are correctly connected to the designated Arduino pins.

2. Check for library installation:
    - Make sure you have installed the required libraries (Wire and LiquidCrystal_I2C) correctly.
    - Go to the "Sketch" -> "Include Library" menu and check if the libraries are listed. If not, install them.
3. Uploading issues:
    - If you encounter problems uploading the code to the Arduino board, ensure that you have selected the correct board type and port under the "Tools" menu.
    - Check if any other programs or processes are using the serial port, as it may prevent successful uploading.
4. LCD display issues:
    - If the LCD display does not show any text or shows garbled characters, double-check the I2C address and wiring connections.
    - Verify that the contrast of the LCD display is properly adjusted. If needed, adjust the potentiometer on the back of the LCD module.
5. LED or button malfunctions:
    - Inspect the connections of the LEDs and buttons. Ensure that they are correctly connected to the designated pins and that there are no loose or faulty connections.
    - Check for any issues with the resistors used for the LEDs. Make sure they are of the correct value and properly connected.
6. Timer or game logic errors:
    - If the timer does not start or the LEDs do not behave as expected, review the code and ensure that there are no syntax errors or logical mistakes.
    - Verify that the button press events are registered correctly and that the LED control and randomization are functioning properly.

Future Enhancements :

1. Wireless Functionality:
    - Integrate a wireless module, such as Wi-Fi or Bluetooth, into the project to enable communication between the Arduino and other devices.
    - Choose a suitable wireless module based on your requirements and compatibility with the Arduino board.
    - Modify the code to incorporate wireless communication protocols and commands for transmitting and receiving data wirelessly.
    - Implement a smartphone application or a web-based interface that can connect to the Arduino wirelessly and control the game, display session records, and perform other functions.
1. Smartphone Integration:
    - Develop a smartphone application that can connect to the Arduino wirelessly and retrieve session records.
    - Design a user-friendly interface in the application to display the session records, including scores, timestamps, and player information.

- Implement features for sharing session records via social media, email, or other communication channels.

References :