**Music Controller**

**Using Arduino**

**ABSTRACT**

This project demonstrates a user-friendly music controller circuit using Arduino. The system features a 16x2 LCD display for real-time song information and three control buttons for basic playback functions: Play/Pause, Next, and Previous. The design aims to deliver a simple yet functional music interface for small-scale applications, demonstrating the integration of microcontrollers and user input devices.

**OBJECTIVE**

Music controllers are essential components in audio systems, offering users control over playback. This project explores the application of Arduino as the core controller to develop a cost-efficient music control system. The system handles basic operations—playing, pausing, and switching tracks—and displays the status on an LCD screen. The project emphasizes simplicity, usability, and educational value.

**COMPONENTS AND SOFTWARE USED**

Hardware Components:

1. Arduino Uno Microcontroller
2. 16x2 LCD Display
3. Push Buttons (Play/Pause, Next, and Previous)
4. Resistors (10k Ohm for pull-up configuration)
5. Breadboard and Jumper Wires

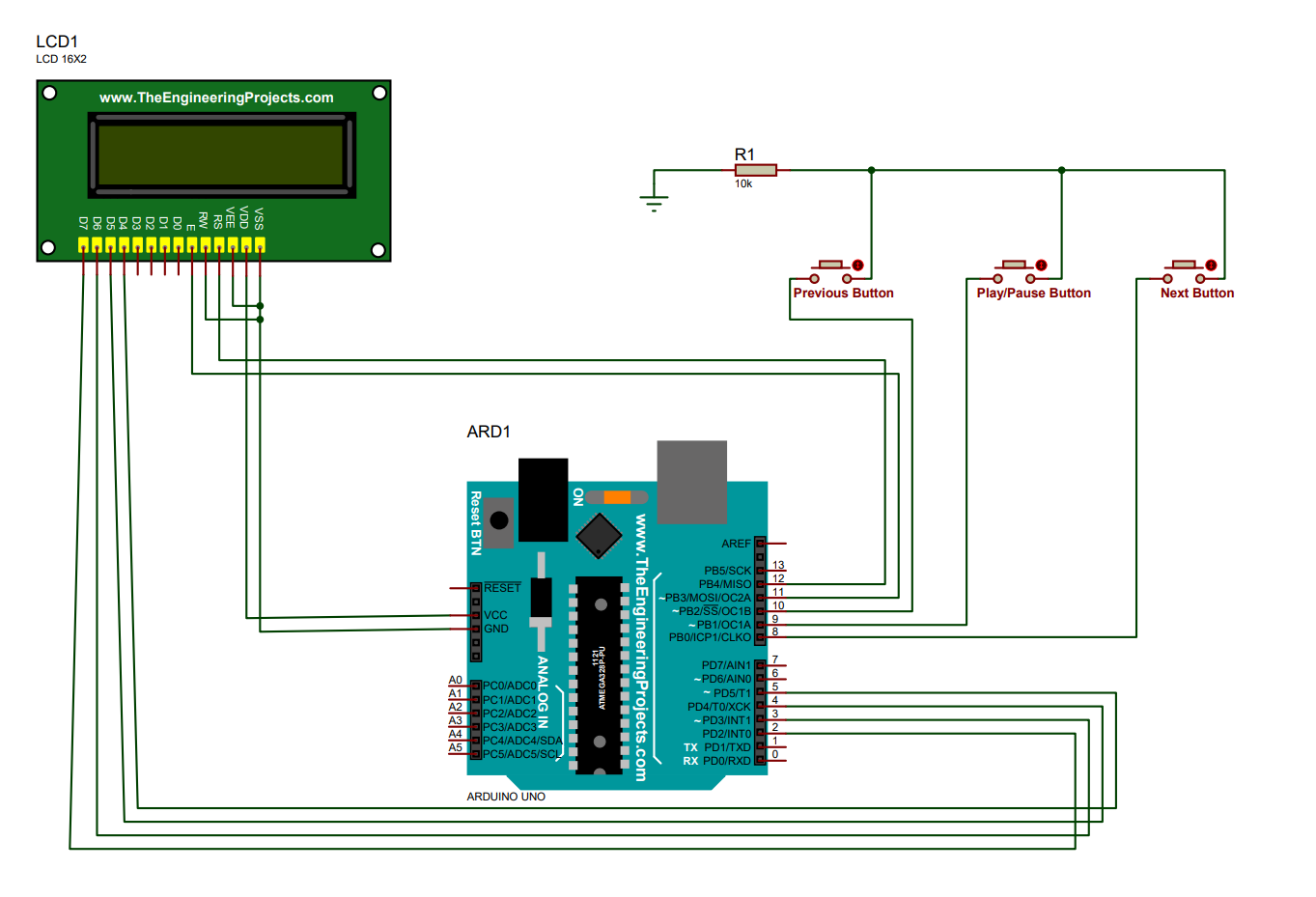
Software Used:

1. Arduino IDE: For programming the microcontroller.
2. Standard Arduino libraries (LiquidCrystal library) for LCD control.

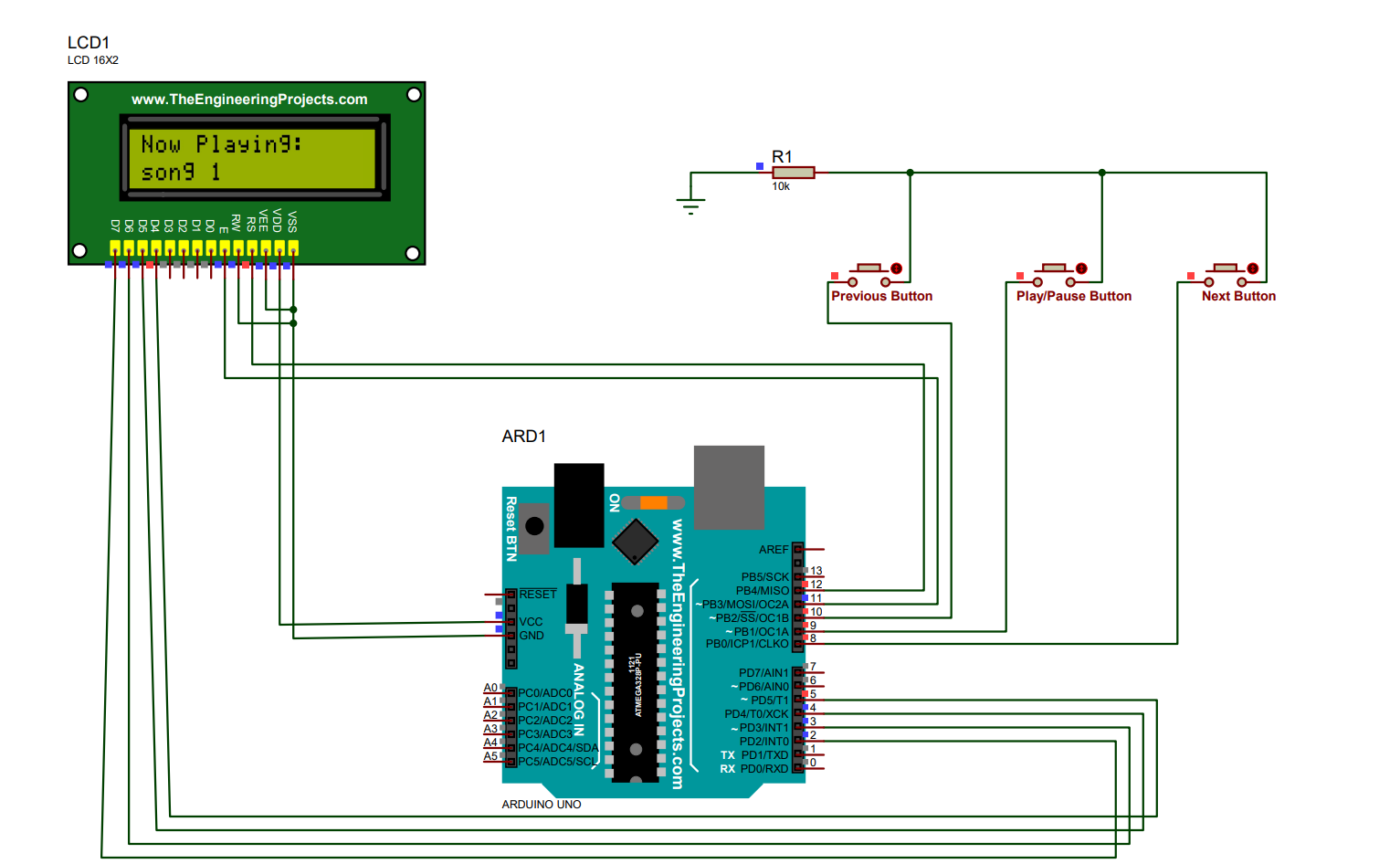
**CIRCUIT DIAGRAM**

The circuit connects the LCD display and buttons to the Arduino:

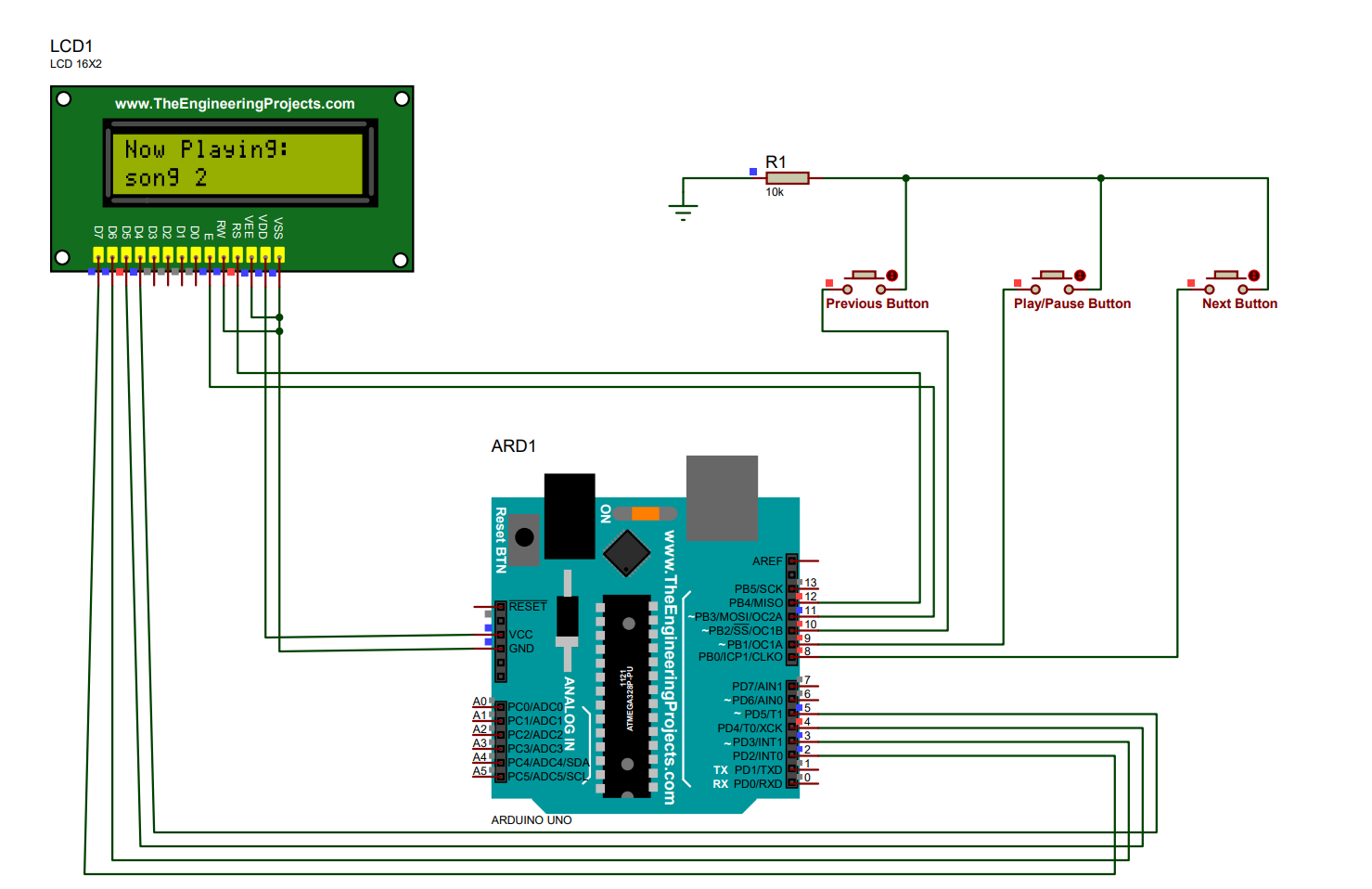
* LCD Connections: RS (12), E (11), D4 (5), D5 (4), D6 (3), D7 (2).
* Button Connections: Play/Pause (9), Next (8), Previous (10)



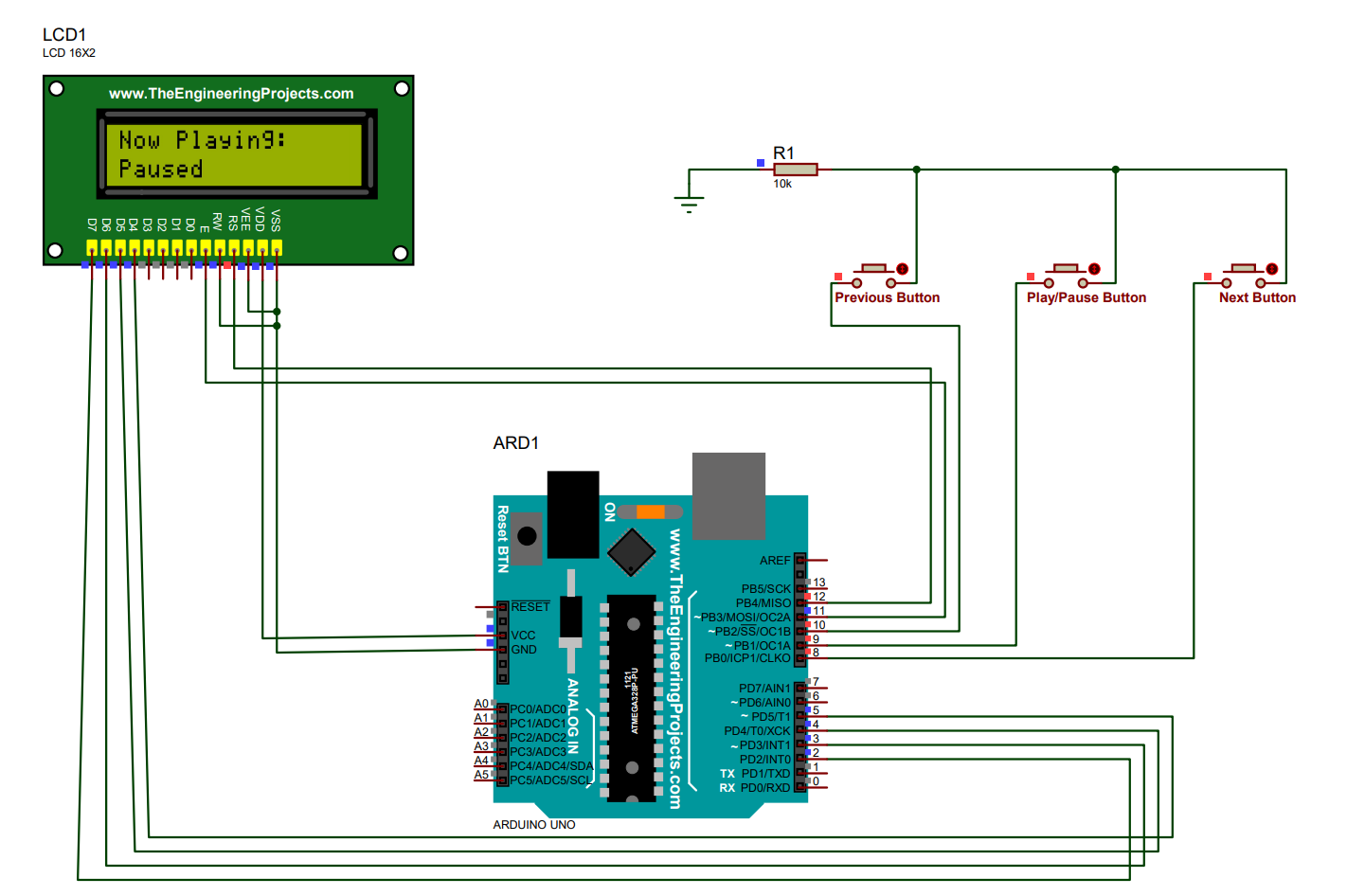
After Switch ON



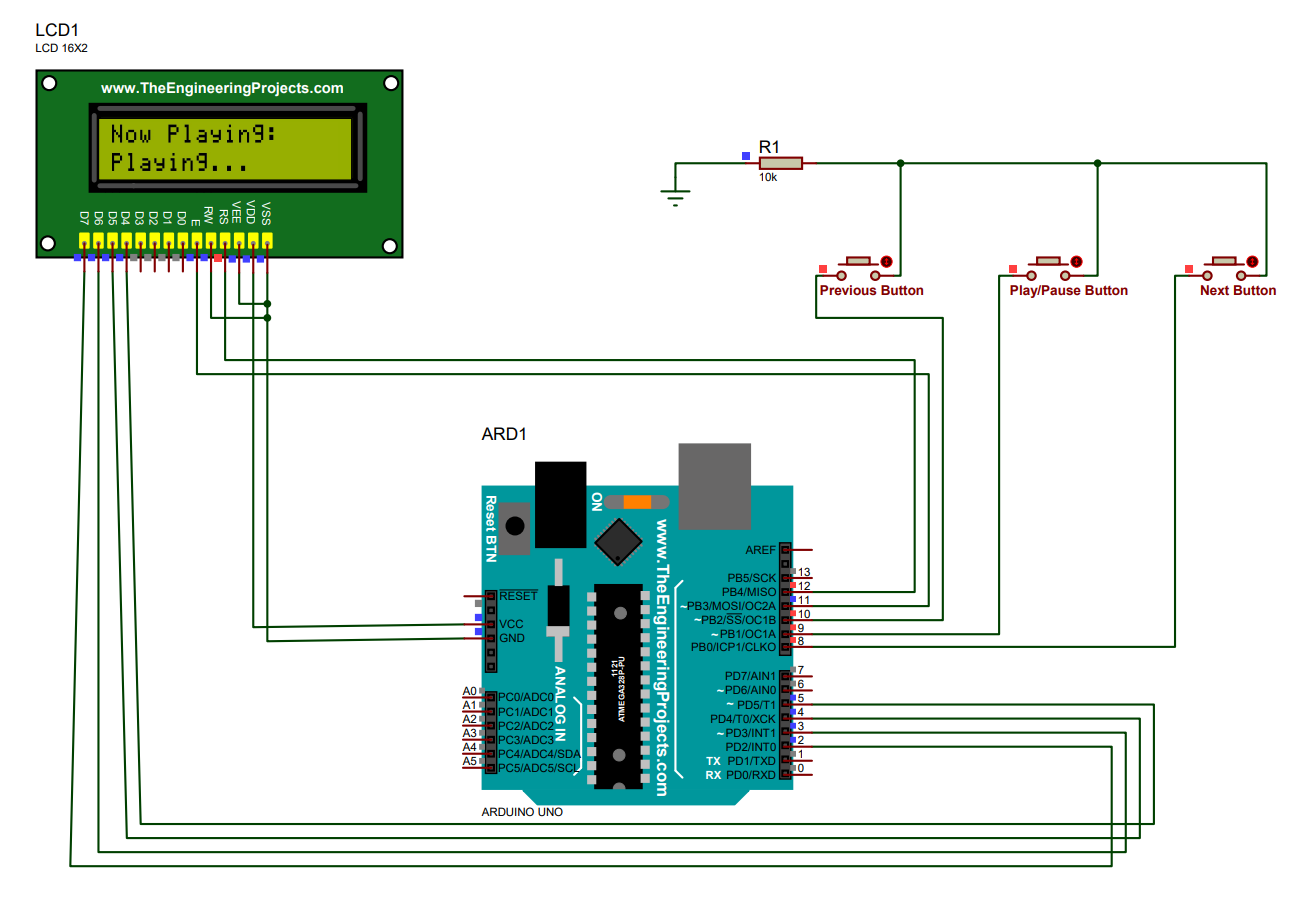
After clicking Next Button



After Clicking Pause Button



After Clicking Play Button



**CODE EXPLANATION**

The system is driven by the Arduino code, which manages button inputs, updates the LCD, and handles the state of music playback.

**Key Features of the Code:**

1. **LCD Initialization**: The LCD displays the system status and the currently playing song.
2. **Button Functionality**:
   * **Play/Pause Button**: Toggles between "Playing" and "Paused" states.
   * **Next Button**: Cycles through the predefined list of songs.
   * **Previous Button**: Cycles backward through the song list.
3. **Debouncing**: Button inputs are debounced with short delays to ensure stability.
4. **Dynamic LCD Updates**: The updateLCD() function ensures real-time updates on the LCD as per the user’s actions.

**Code Snippet Overview:**

**Initialization**:

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

const int playPauseButton = 9, nextButton = 8, prevButton = 10;

String songs[] = {"song 1", "song 2", "song 3"};

**Play/Pause Toggle**:

if (digitalRead(playPauseButton) == LOW) {

isPlaying = !isPlaying;

lcd.setCursor(0, 1);

lcd.print(isPlaying ? "Playing..." : "Paused");

}

**Song Navigation**:

currentSong = (currentSong + 1) % 3; // For next song

currentSong = (currentSong - 1 + 3) % 3; // For previous song

**COMPLETE CODE**

#include <LiquidCrystal.h>

// LCD pins: (RS, E, D4, D5, D6, D7)

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// Button pins

const int playPauseButton = 9;

const int nextButton = 8;

const int prevButton = 10;

// Song names array

String songs[] = {"song 1", "song 2", "song 3"};

int currentSong = 0;

bool isPlaying = true; // Initial state is 'Playing'

void setup() {

  // Set button pins as input with pull-up resistors

  pinMode(playPauseButton, INPUT\_PULLUP);

  pinMode(nextButton, INPUT\_PULLUP);

  pinMode(prevButton, INPUT\_PULLUP);

  // Initialize the LCD

  lcd.begin(16, 2);

  lcd.print("Music Controller");

  delay(2000);

  updateLCD();

}

void loop() {

  // Play/Pause Button

  if (digitalRead(playPauseButton) == LOW) {

    isPlaying = !isPlaying; // Toggle between play and pause

    if (isPlaying) {

      Serial.println("Play");

      lcd.setCursor(0, 1);

      lcd.print("Playing...      ");

    } else {

      Serial.println("Pause");

      lcd.setCursor(0, 1);

      lcd.print("Paused          ");

    }

    delay(300); // Debounce delay

  }

  // Next Button

  if (digitalRead(nextButton) == LOW) {

    currentSong = (currentSong + 1) % 3; // Loop through songs

    Serial.println("Next Track");

    updateLCD();

    delay(10); // Debounce delay

  }

  // Previous Button

  if (digitalRead(prevButton) == LOW) {

    currentSong = (currentSong - 1 + 3) % 3; // Loop backward through songs

    Serial.println("Previous Track");

    updateLCD();

    delay(300); // Debounce delay

  }

}

// Function to update LCD with current song name

void updateLCD() {

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("Now Playing:");

  lcd.setCursor(0, 1);

  lcd.print(songs[currentSong]);

}

**Results**

The system successfully performs the following:

* Displays the current song on the LCD.
* Allows users to switch between songs seamlessly using the "Next" and "Previous" buttons.
* Toggles between playing and paused states with the "Play/Pause" button.

**Applications**

1. Low-cost DIY music systems.
2. Educational demonstrations of Arduino's capabilities.
3. Integration into small-scale embedded systems requiring playback control.

**Conclusion**

This project highlights the utility of Arduino in creating interactive systems. The design meets its objectives of providing basic music control with real-time feedback. It demonstrates how simple hardware and software integration can lead to functional systems.

**Future Scope**

1. Adding features like volume control and track seek.
2. Wireless control using Bluetooth modules.
3. Expanding song storage options using SD cards or USB integration.