**Lab Exercise 3- Controlling Buzzer in Arduino UNO**

In this lab exercise, you will learn how to use a buzzer with an Arduino to create sound. You will control the buzzer to produce different tones and patterns based on the code you write.

**Objective:**

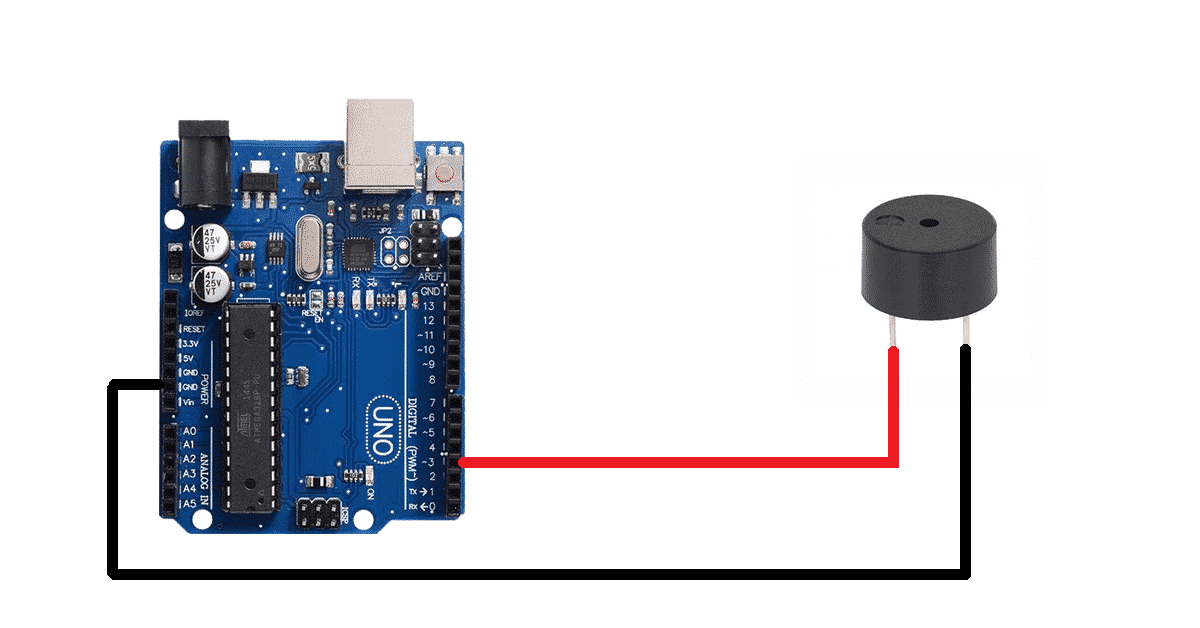
* Control a buzzer using an Arduino to produce sound.

**Materials Needed:**

* Arduino board (e.g., Arduino Uno)
* 1x Passive or Active Buzzer
* Breadboard (optional, but recommended)
* Jumper wires
* USB cable to connect the Arduino to your computer

**Circuit Diagram:**

1. Connect the components as follows:
   * Connect the **positive terminal** (longer leg) of the buzzer to **Digital Pin 9** on the Arduino.
   * Connect the **negative terminal** (shorter leg) of the buzzer to **GND** on the Arduino.



**Steps to Perform the Lab:**

**Step 1: Set Up the Arduino IDE**

1. Open the Arduino IDE.
2. Connect your Arduino board to your computer via the USB cable.
3. In the **Tools** menu, select the correct **Board** (e.g., Arduino Uno) and **Port**.

**Step 2: Write the Arduino Code**

1. **Code to Control the Buzzer:**

// Define the pin for the Buzzer

const int buzzerPin = 3; // Connect the Buzzer to Digital Pin 9

void setup() {

// Set the buzzer pin as an output

pinMode(buzzerPin, OUTPUT);

}

void loop() {

// Play a tone

tone(buzzerPin, 1000); // Play 1000 Hz for 1 second

delay(1000); // Wait for 1 second

noTone(buzzerPin); // Stop sound

delay(500); // Wait for 0.5 seconds

// Play a different tone

tone(buzzerPin, 500); // Play 500 Hz for 1 second

delay(1000); // Wait for 1 second

noTone(buzzerPin); // Stop sound

delay(500); // Wait for 0.5 seconds

}

1. **Explanation:**
   * The buzzer is connected to **Digital Pin 9**. The code uses this pin to generate sound.
   * In setup(), we set the buzzer pin as an output.
   * In loop(), we use the tone() function to play sound at specified frequencies (in Hz). The noTone() function stops the sound.

**Step 3: Upload the Code**

1. Upload the code to the Arduino by clicking the **Upload** button (right arrow icon).
2. Once uploaded, the buzzer should start producing sound based on the code.

**Step 4: Test the Buzzer**

1. After uploading the code, the buzzer will produce a tone at **1000 Hz** for one second, followed by a **500 Hz** tone.
2. You should hear alternating tones every second.

**Step 5: Experimentation**

* **Modify the Code:**
  + Change the frequencies in the tone() function to create different sounds. For example, try tone(buzzerPin, 2000) for a higher pitch.
* **Add Melodies:**
  + You can define an array of notes and durations to play a simple melody. Here’s an example:

// Melody notes

int melody[] = {262, 294, 330, 349, 392, 440, 494, 523}; // C4 to C5

int noteDuration = 500; // Duration of each note in milliseconds

void loop() {

for (int i = 0; i < 8; i++) {

tone(buzzerPin, melody[i]); // Play the note

delay(noteDuration); // Wait for the note duration

noTone(buzzerPin); // Stop sound

delay(50); // Brief pause between notes

}

delay(1000); // Wait for 1 second before repeating

}

**Summary of Key Concepts:**

* **Output Control:** Using the tone() and noTone() functions to control sound output from the buzzer.
* **Frequency:** Understanding how different frequencies produce different tones.
* **Melody Creation:** Creating simple melodies by playing a series of tones in succession.

This lab exercise provides a foundation for working with sound in Arduino projects. Enjoy experimenting with different sounds and melodies!