Noshin Anjum Nisa
Interactive Devices: Lab 4

Part A. Revisiting Blink

1. Blinking LEDs with Arduino

a. **What line(s) of code do you need to change to make the LED blink (like, at all)?**

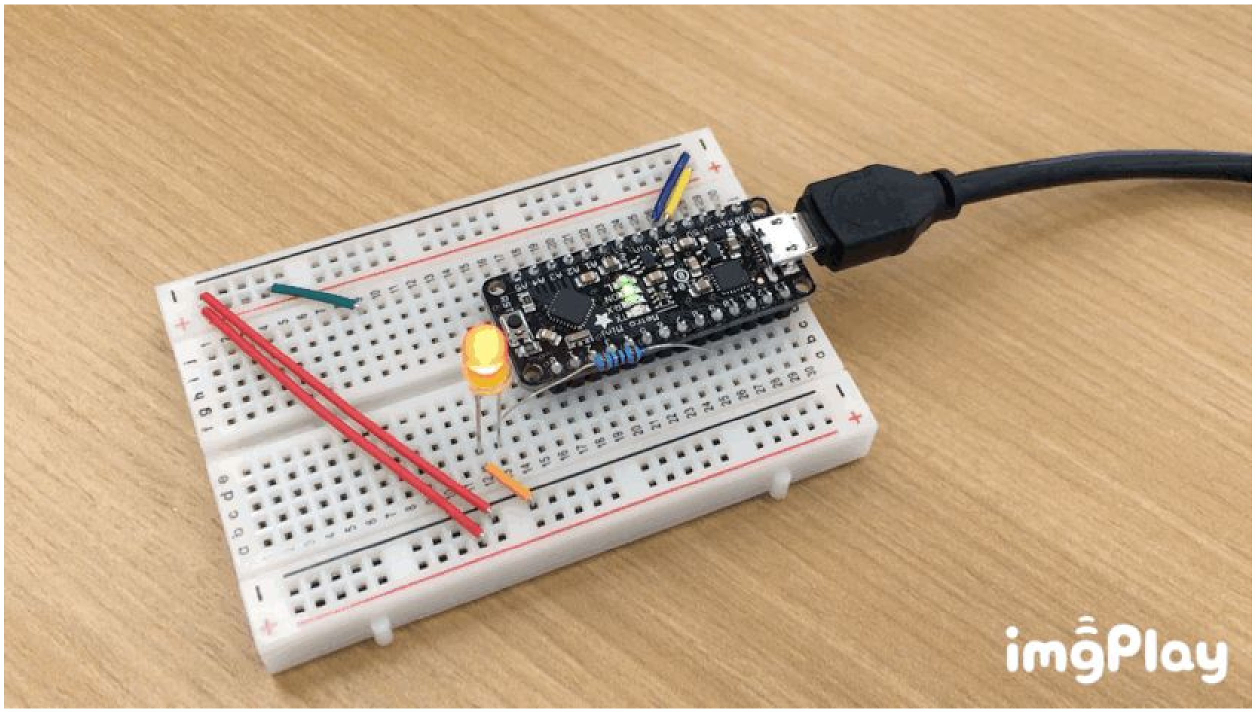
To make the light blink we don't have to change any lines of code since it is using the built in LED and delay to make it blink.

b. **What line(s) of code do you need to change to change the rate of blinking?**

To change the rate of blinking of the built in LED we only need to change the number of milliseconds we want to wait before changing the state. In other words we only need to change delay (1000) lines of code.

c. **What circuit element would you want to add to protect the board and external LED?**

We need to use a resistor while connecting our LED to make sure we control the voltage and don't burn our arduino.

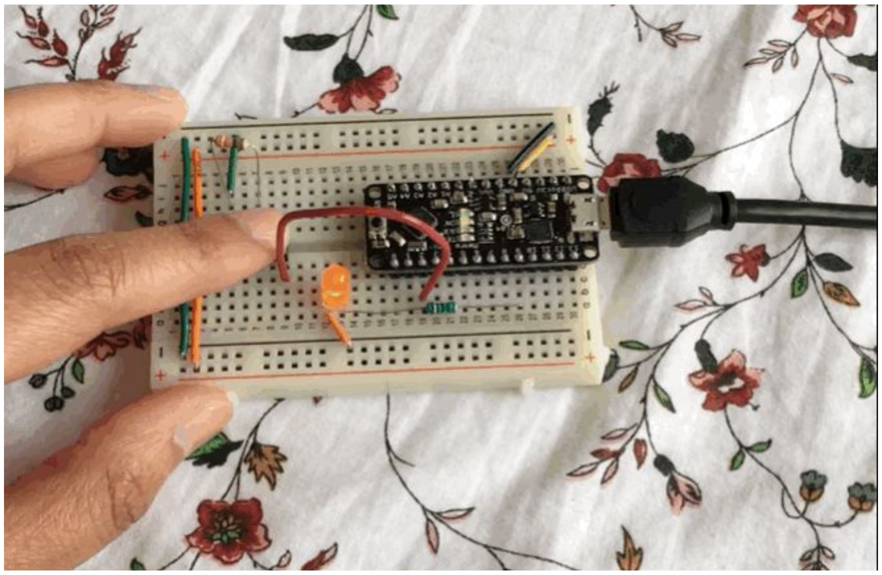


2. Digitally toggle LEDs on and off using the Arduino

a. Which lines do you need to modify to correspond with your button and LED pins?

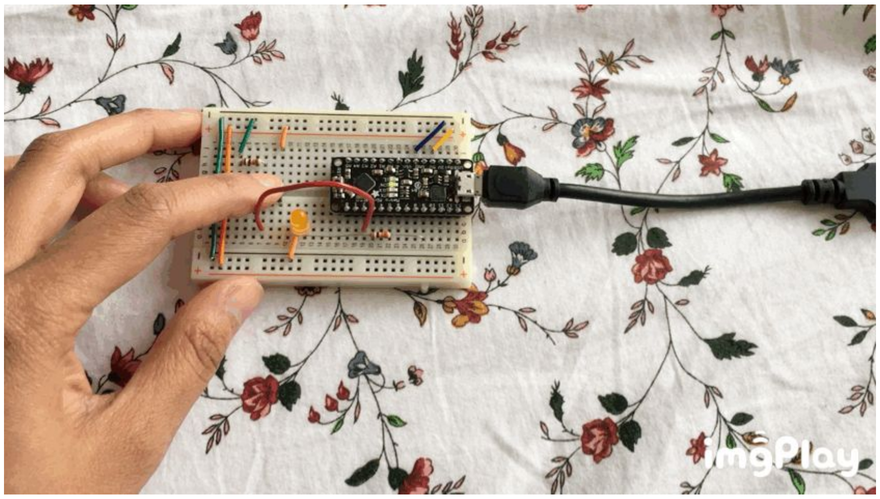
We only need to modify the button pin number from 13 to 2.

```
const int buttonPin = 2;
```



b. Modify the code or the circuit so that the LED lights only while the button is depressed. Include your code in your lab write-up.

When the pin is connected through the ground, the circuit is not complete until the button is pressed. This means that until the button is pressed the LED will not turn on.



Code:

```
// constants won't change. They're used here to set pin numbers:
```

```
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  9;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton
status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed. If it is, the buttonState is
HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
```

3. Fading LEDs on and off using Arduino

a. Which line(s) of code do you need to modify to correspond with your LED pin?

We don't have to change any line of code because the way the circuit is done, LED is already connected to one of the analog pins which is 9 given in the code itself.

b. How would you change the rate of fading?

We could change the rate of fading by changing the delay time by making it faster or slower. We could also change the incrementation value depending on how gradually we want to see the change.

c. (Extra) Since the human eye doesn't see increases in brightness linearly and the diode brightness is also nonlinear with voltage, how could you change the code to make the light appear to fade linearly?

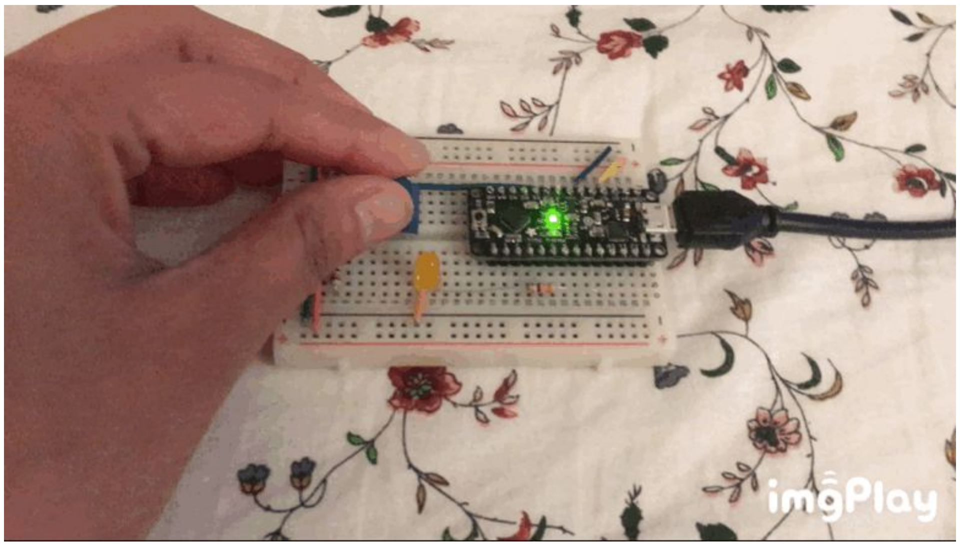
We could potentially increase the delay for voltage change so that the human eye can see it the increment values.

Part B. Advanced Inputs

1. Potentiometer

a. **Post a copy of your new code in your lab write-up.**

```
int sensorPin = A0;    // select the input pin for the potentiometer
int ledPin = 9;      // select the pin for the LED
int sensorValue = 0;  // variable to store the value coming from the
sensor
void setup() {

  // declare the ledPin as an OUTPUT:
  pinMode(ledPin, OUTPUT);

}

void loop() {

  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  //mapping the value to control the led brightness
  sensorValue = map(sensorValue, 0, 1023, 0, 255);
  analogWrite(ledPin, sensorValue);

}
```



2. Force sensitive sensor

a. **What resistance values do you see from your force sensor?**

The value ranged from 0 to 980.

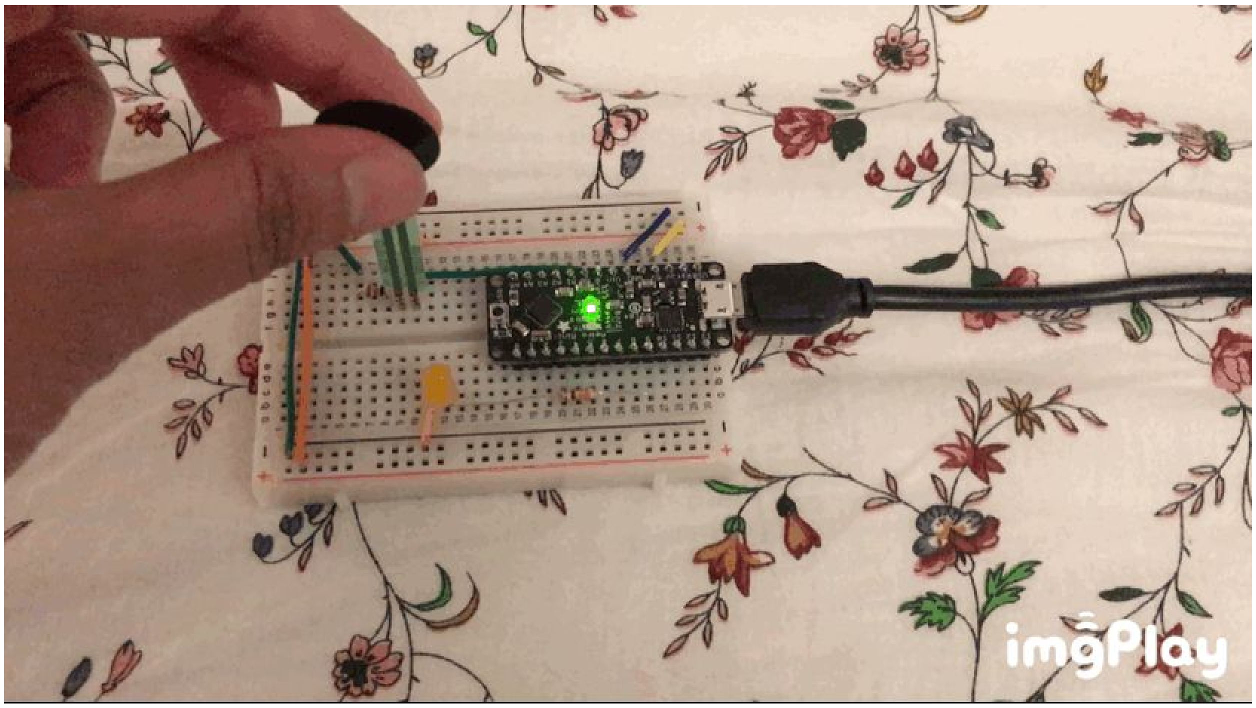
b. **What kind of relationship does the resistance have as a function of the force applied? (e.g., linear?)**

The relationship is directly proportional.

c. **Can you change the LED fading code values so that you get the full range of output voltages from the LED when using your FSR?**

We can map the FSR values using map function from 0-1023 range to 0-255 LED range.

```
int fsrAnalogPin = 0; // FSR is connected to analog 0
int LEDpin = 9;      // connect Red LED to pin 11 (PWM pin)
int fsrReading;      // the analog reading from the FSR resistor
divider
int LEDbrightness;

void setup(void) {
       Serial.begin(9600);   // We'll send debugging information via
the Serial monitor
       pinMode(LEDpin, OUTPUT);
      }
void loop(void) {
  fsrReading = analogRead(fsrAnalogPin);
  Serial.print("Analog reading = ");
  Serial.println(fsrReading);
// we'll need to change the range from the analog reading (0-1023)
down to the range
// used by analogWrite (0-255) with map!
      LEDbrightness = map(fsrReading, 0, 1023, 0, 255);
// LED gets brighter the harder you press
      analogWrite(LEDpin, LEDbrightness);
      delay(100);
}
```



Part C. Writing to the LCD

a. **What voltage level do you need to power your display? b. What voltage level do you need to power the display backlight?**

In order to power the display we need 5V and for the display backlight we need 3.3V.

b. **What was one mistake you made when wiring up the display? How did you fix it?**

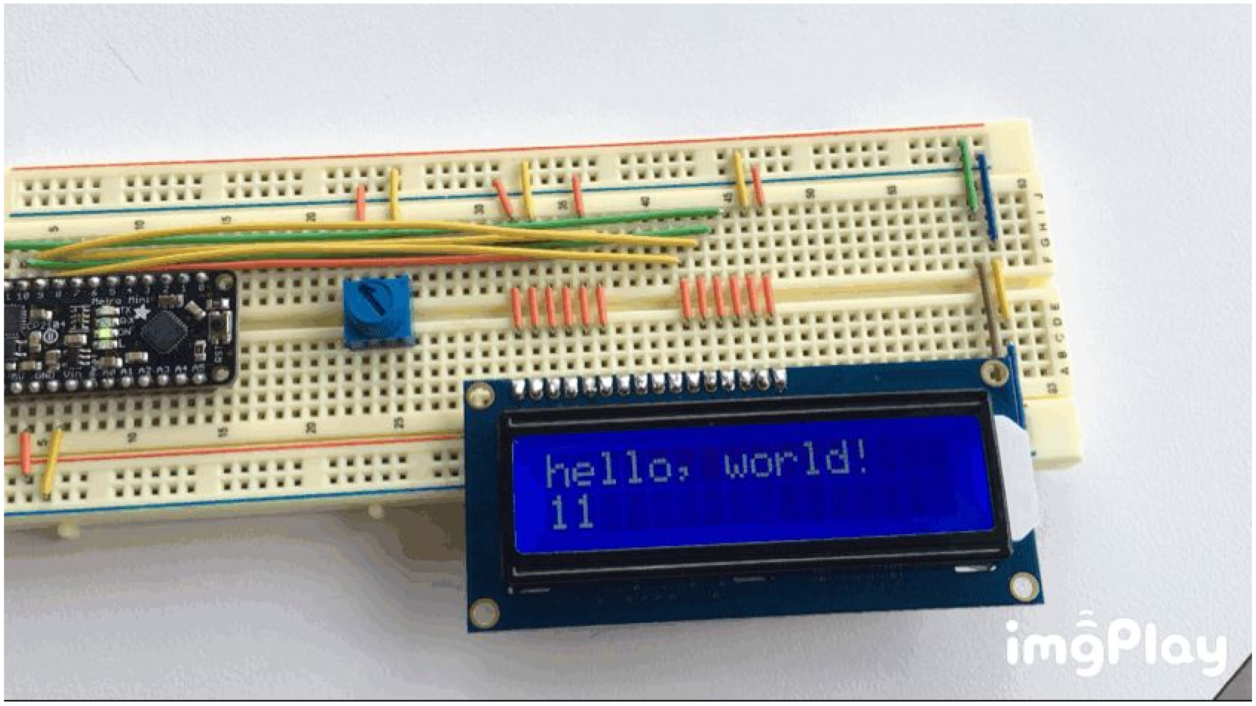
While writing the code for LCD my mapping for pins were wrong, instead of using rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2, I used 7, 8, 9, 10, 11 and 12. I changed few lines to make this work, remapped the number so that it works with the current connection. Later when I connected my FSR, instead of giving me higher value while being pressed it was giving me lower values.

c. **What line of code do you need to change to make it flash your name instead of "Hello World"?**

The "Hello World" text needs to be changed to the respective name.

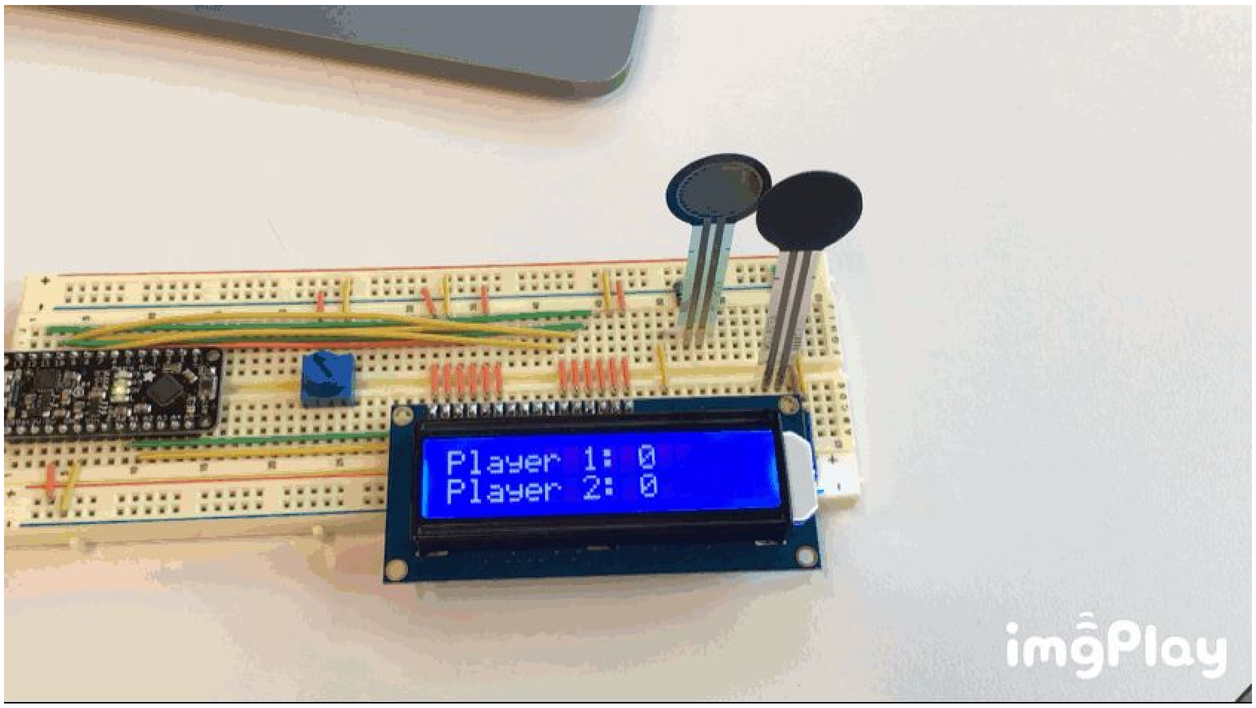
d. **Include a copy of your Lowly Multimeter code in your lab write-up.**

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
//const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
//LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

LiquidCrystal lcd(7, 8, 9, 10, 11, 12);

void setup() {

  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {

  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);

}
```



e. Include a copy of your FSR thumb wrestling code in your lab write-up.

```
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
int first_Pin = A0; // first potentiometer = pin A0
int second_Pin = A1; // second  potentiometer = pin A1

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  pinMode(first_Pin, INPUT);
  pinMode(second_Pin, INPUT);
}

void loop() {
  lcd.setCursor(0,0);
  lcd.print("Player 1: ");
  lcd.print(analogRead(first_Pin));
  lcd.print("   ");
  lcd.setCursor(0,1);
  lcd.print("Player 2: ");
  lcd.print(analogRead(second_Pin));
  lcd.print("   ");
}
```



Part D. Timer

a. **Make a short video showing how your timer works, and what happens when time is up!**
b. **Post a link to the completed lab report to the class Slack.**

```
// include the library code:
#include <LiquidCrystal.h>
#include "pitches.h"

LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
int sensor_pin = A1;

// notes in the melody:
int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
```

```
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};



void setup() {
  lcd.begin(16, 2);
  lcd.setCursor(0,0);
  lcd.print("Touch to set");
  lcd.setCursor(0,1);
  lcd.print("time!");
  pinMode(sensor_pin, INPUT);
}

void loop() {
  int sensor_value = analogRead(sensor_pin);
  if(sensor_value > 0)
  {
    for(int i = sensor_value; i>=0; i = i-1)
    {
      delay(1000);
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("You have:");
      lcd.setCursor(0,1);
      lcd.print(String(i) + " seconds left");
      if(i == 0)
      {
        music();
        lcd.setCursor(0,0);
        lcd.print("TIME'S UP");
        delay(2000);
        sensor_value = 0;
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Touch to set");
        lcd.setCursor(0,1);
        lcd.print("time!");
      }
    }
  }
  else
  {
    delay(1000);
  }
}

void music() {
  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // to calculate the note duration, take one second divided by the
note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(8, melody[thisNote], noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(8);
  }
}
```

Video of the timer:
https://www.youtube.com/watch?v=88ypVORLg0Y