Free Radical Labs

Controlling the Brightness of an LED with a Potentiometer
Tutorial #2

Controlling Brightness of an LED with a Potentiometer

After learning how to upload and write a program to the Sparkfun Redboard, it's time to do something fun with it! This tutorial will have you controlling an LED with a Potentiometer similar to how a dimmer switch works with a light fixture.

Part 1. Gathering Materials

For this tutorial, you'll need a to acquire a few parts first:

- A computer with the Arduino IDE installed and setup for the Sparkfun Redboard
- 1 x Trim Potentiometer
- 1 x LED
- 1 x 330 Ω resistor
- 1 x breadboard
- 5 x 6" Jumper Wire
- 1 x Sparkfun Redboard
- 1 x USB (B) Cable

Part 2. Wiring the Components

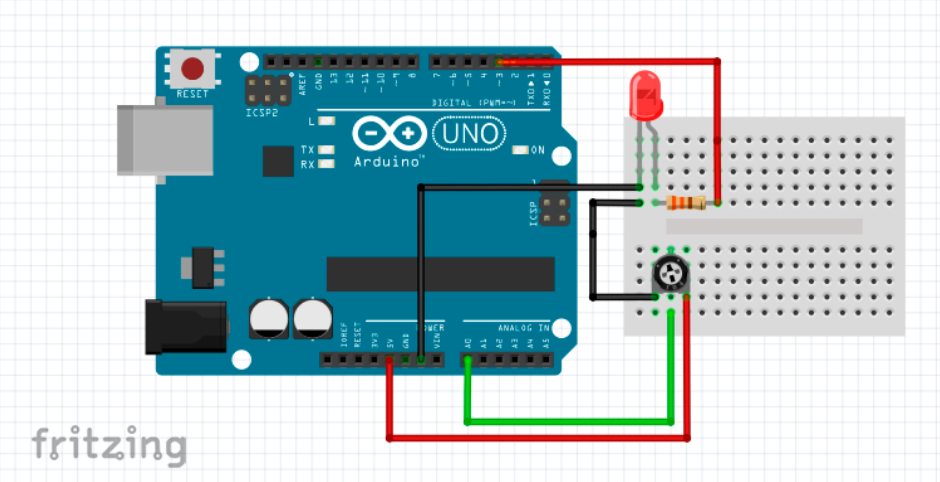

Figure 1. Wiring Diagram for a Potentiometer and LED control system

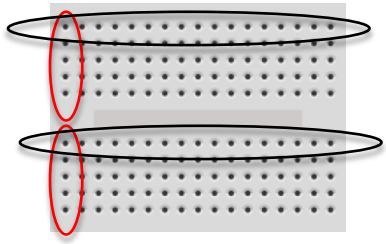

Figure 2. A breadboard similar to the ones used in this tutorial. Red circles represent electrically connected pins. While black circles represent non-connected pins

A breadboard works by having pins connected electrically in each column. With each column being isolated by a spacer in the middle. The columns are separated into two parts, allowing ICs to be connected such as a resistor network.

After gathering the materials, wire the setup to how it's shown in Figure 1. Connect a wire between Pin 3, and one end of the resistor with the other end of the resistor being connected to the non-flat side of the LED. The flat side of the LED gets connected by a wire to a GND Pin on the Arduino. The potentiometer has three pins. One pin on the ends go to the 5V Pin on the Arduino and the other potentiometer pin on the opposite side connects to the same column of the connection made between the LED and the GND pin. The middle pin connected to A0, an analog input pin.

Part 3. Programming

```
void setup() {

  //Tells the Arduino to use Pin 3 as an output pin.
  pinMode(3, OUTPUT);

}

void loop() {

  /* analogRead uses the pin specified, in this case A0
   * and reads the voltage and produced a whole number
   * (an integer) between 0 and 1023. Corresponding to
   * 0V and 5V respectively. For example, if A0 had
   * an applied voltage of 2.5V, the command
   * analogRead(A0) would return a value of 511.
```

```
  */

  int potentiometerValue = analogRead(A0);

  /*  potentiometerValue will correspond to how far
   *  the potentiometer has been turned. With a value
   *  of zero corresponding to a full left turn and a
   *  value of 1023 corresponding to a full right turn.
   *
   *  To control how bright the LED is, we need to send out
   *  a PWM (Pulse Width Modulation) signal. A PWM signal sends out 5V for
   *  a set amount time and then sends out 0V with a ratio
   *  that equals the value given to the analogWrite command.
   *
   *  analogWrite needs two values, the first being which pin
   *  will be used for the output, and the second number corresponding
   *  to the ratio of on time and off time the pin will have.
   *  With 0 being completely off and 256 being completely on.
   *
   *  The signal is turned on and off fast enough that, to our
   *  eyes is blurred and the LED will have a brightness that is related
   *  to the value in analogWrite.
   *
   *  Because analogWrite has a maximum of 255 and analogRead
   *  has a maximum of 1023, the potentiometer value needs
   *  to be divided by 4.
   */

    analogWrite(3, potentiometerValue/4);

}
```

Because the code is contained in “Void loop()”, it will run continuously until power is disconnected from the Arduino.

Part 4. Testing and Conclusion

This tutorial has shown you how to control the brightness of an LED using a potentiometer. Along with how to use the analogRead and analogWrite functions of the Arduino IDE. These are powerful commands that allow for the use of distance sensors, motors, accelerometers and other components frequently used in the microcontroller world.