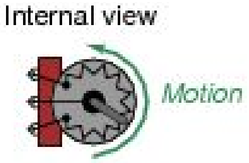
## Section 1: What is a Potentiometer, Reading a Potentiometers Signal and Writing to Serial.

Below is a link explaining what potentiometers are and how they work:
https://www.allaboutcircuits.com/textbook/experiments/chpt-3/potentiometer-voltage-divider/



Above is an internal view of a potentiometer you will see in the link above that helps show what a potentiometer is. To make sense of this diagram, it is a variable resistor that changes resistance from the maximum resistance to zero as the dial is turned.

The pictures show how to set up the hardware to get the Arduino to work properly using the code below. The white wire connected to the middle pin on the potentiometer is connected to analog pin 3 A3 on the Arduino. One side of the potentiometer is connected to 5V on the Arduino (the red wire). The black wire is connected from the last pin left on the potentiometer to the Arduino ground (GND).

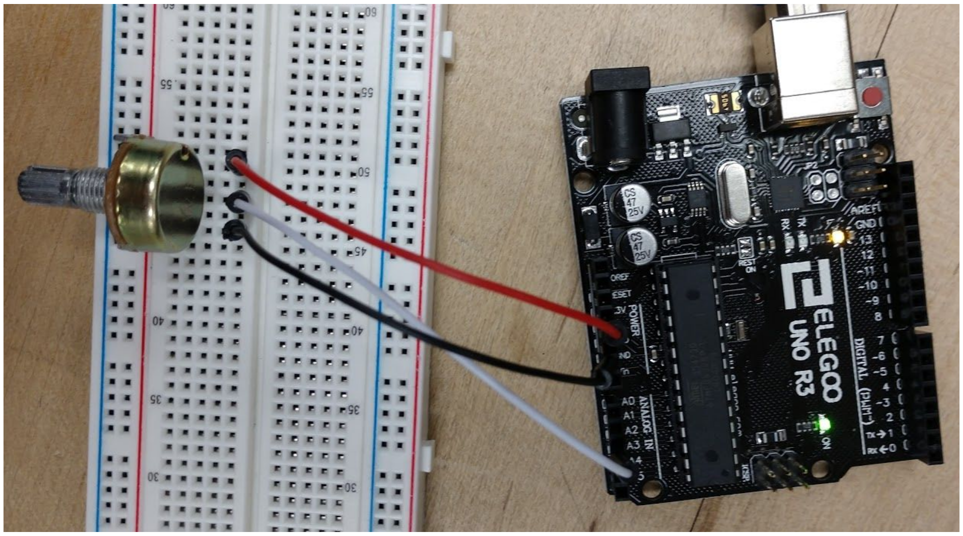


Figure 1.1: Hook up example overview

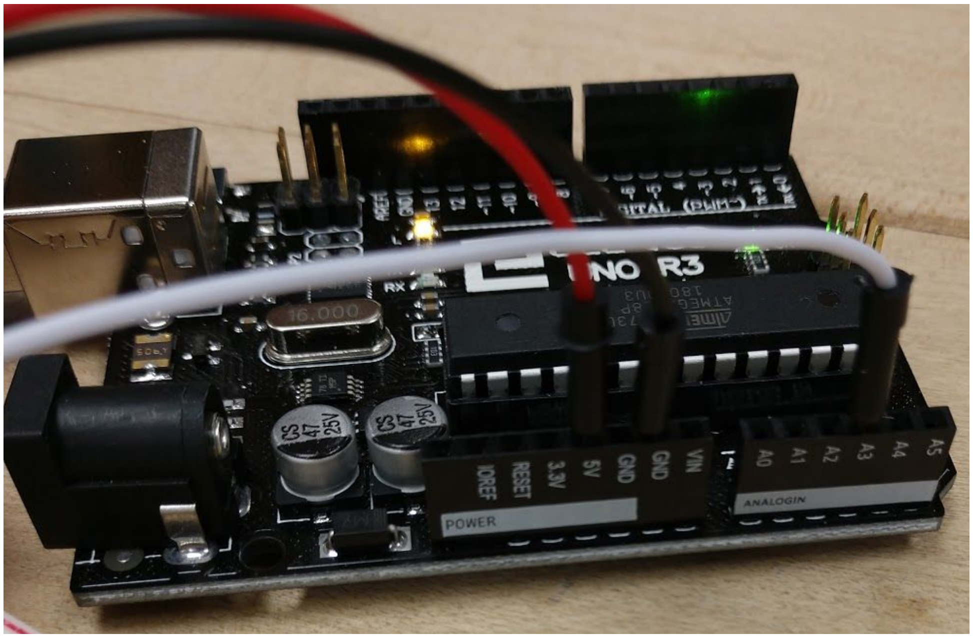


Figure 1.2: Hook up example overview

### Section 1.1: Explanation for Reading Potentiometer and outputting to serialport

In the sample of code on the next page we are creating three variables: one for an analog pin, one for the reading taken from that analog pin, and one for converting the analog reading into voltage. We will be printing the calculated voltage to the Serial Monitor using Serial Communication. The line “Serial.begin(9600);” opens this communication, allowing for the Arduino to send data printed on your computer monitor. The function analogRead() reads the input coming from our potentiometer and ranges from 0 to 1023. We can then use that knowledge to convert our reading to what corresponding voltage value that means we are getting from 0 to whatever voltage we are reading. To open the Serial Monitor go to Tools>Serial Monitor on the toolbar at the top left of the screen when you are in the Arduino software or use the shortcut Ctrl+Shift+M. This will be where the voltage values will appear while the program is running.

## Section 1.2: Code for Section 1.1

```
//code starts here
int potentiometer = A3; //makes the potentiometer variable analog pin 3
double reading = 0.0; //sets a variable for reading the analog input
double voltage = 0.0; //sets a variable to calculate the voltage

void setup() {

  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps

}

void loop() {

  reading = analogRead(potentiometer); //reads the analog input from the potentiometer
  voltage = (reading/1023)*5; //calculates the voltage
  Serial.print(voltage); //prints the value for the voltage
  Serial.print(" volts\n"); //print the word volt after the value
  delay(1000); //delays the reading every 1000 milliseconds (1 second)

}
```

# Section 2: Read Potentiometer Signal and Writing to LED(s) via PWM

## 2.1: Reading Potentiometer and Outputting to a Single LED

https://www.arduino.cc/en/Tutorial/AnalogInOutSerial

The image below shows how to hook up a potentiometer and LED to a arduino, so it can dim and brighten a LED based on the signal of the potentiometer. The LED needs to be connected to the PWM pins on the arduino shown by a ~ symbol. PWM stands for pulse width modulation and allows an arduino to digitally simulate an analog signal or value which allows the LED to be dimmed.

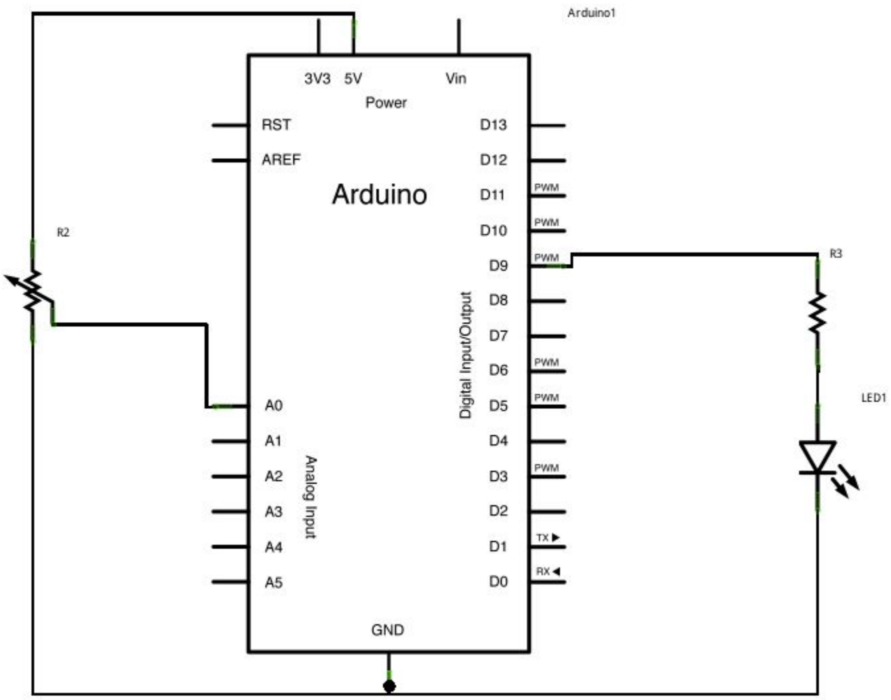


Figure 2.1: Hook up schematic for LED and potentiometer

### 2.1.1: Explanation for Reading Potentiometer and output to a single LED

In the sketch below, after declaring two pin assignments (analog 0 for our potentiometer and digital 9 for your LED) and two variables, sensorValue and outputValue, the only things that you do in the setup() function is to begin serial communication.

Next, in the main loop, sensorValue is assigned to store the raw analog value read from the potentiometer. Arduino has an analogRead range from 0 to 1023, and an analogWrite range only from 0 to 255, therefore the data from the potentiometer needs to be converted to fit into the smaller range before using it to dim the LED.

In order to convert this value, use a function called map():

```
outputValue = map(sensorValue, 0, 1023, 0, 255);
```

outputValue is assigned to equal the scaled value from the potentiometer. map() accepts five arguments: The value to be mapped, the low range and high values of the input data, and the low and high values for that data to be remapped to. In this case, the sensor data is mapped down from its original range of 0 to 1023 to 0 to 255.

The newly mapped sensor data is then output to the analogOutPin dimming or brightening the LED as the potentiometer is turned. Finally, both the raw and scaled sensor values are sent to the Arduino Software (IDE) serial monitor window, in a steady stream of data.

### 2.1.2: Code for reading potentiometer, outputting to a single LED

```
// These constants won't change. They're used to give names to the pins used:
const int analogInPin = A0;  // Analog input pin that the potentiometer is attached to
const int analogOutPin = 9; // Analog output pin that the LED is attached to

int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM (analog out)

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // change the analog out value:
  analogWrite(analogOutPin, outputValue);

  // print the results to the Serial Monitor:
  Serial.print("sensor = ");
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);
  // wait 2 milliseconds before the next loop for the analog-to-digital
  // converter to settle after the last reading:
  delay(2);
}
```

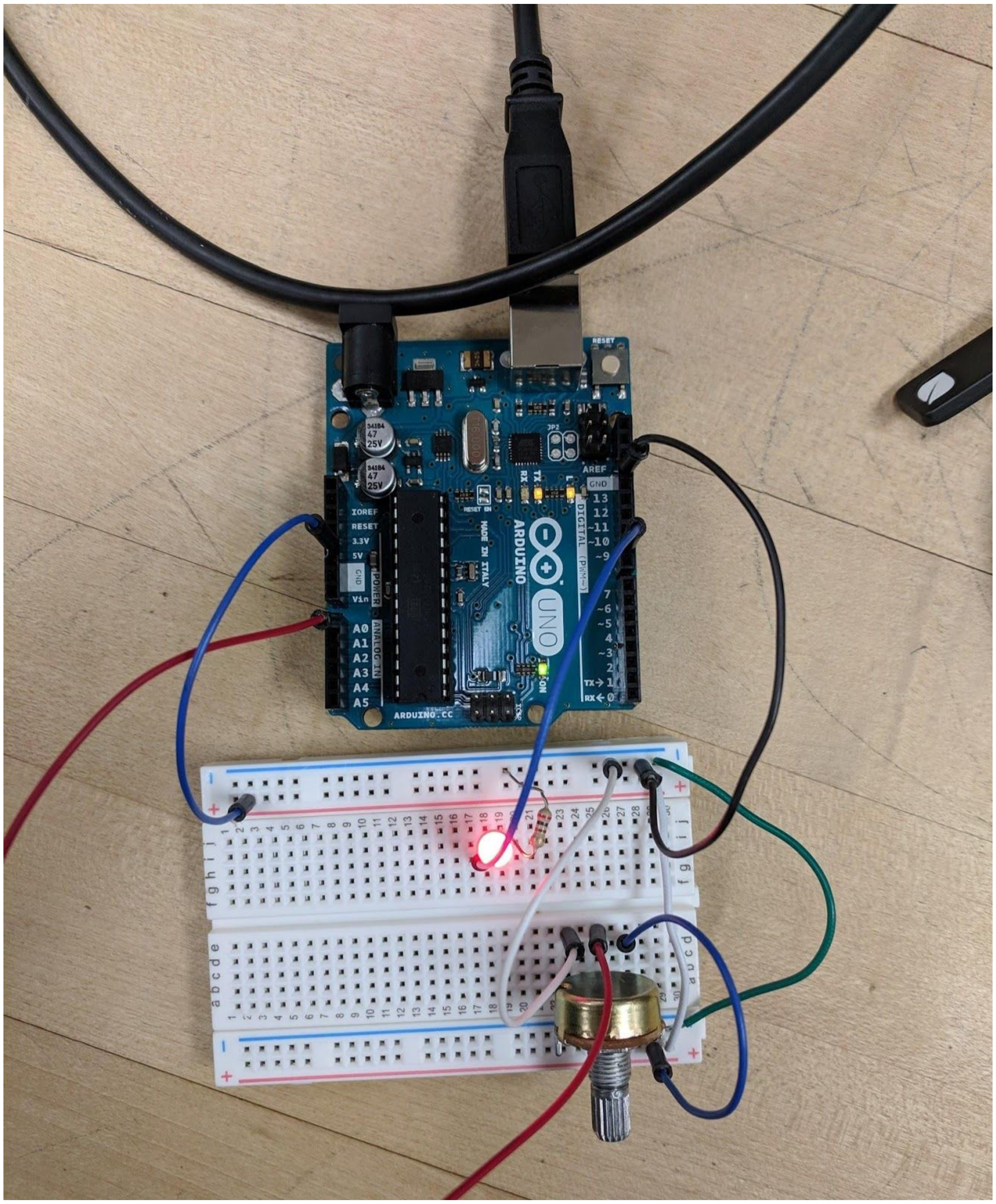


Figure 2.2: Hook up example for LED and Potentiometer

## Section 2.2: Reading potentiometer, outputting to a multiple LEDs

Now using what you have done in the first Section and section 2.1, expand on them and build something more slightly more complicated without the code provided. Connect 2 more leds to PWM pins (analog output) and program the arduino so that when the potentiometer is turned, it lights up and brightens the LEDs one after each other. They should also dimm off one after each other as the potentiometer knob is turned.

So When the value from the potentiometer is 0 no LED is lit but as that value increases the first LED light up and gets brighter until a third of the way through. At this point the 1st LED should always be lit but then the 2nd LED will light up and start to get brighter until 2/3s of the way through. Now the first 2 LEDs should be lit and the third LED will light up and start to get brighter till the max value of the potentiometer.

# Section 3: Read Potentiometer signal and set servo position based on it.

This is just using the servo library and doing the same as before
https://www.arduino.cc/en/Tutorial/Knob

## Section 3.1 Explanation for Reading Potentiometer setting to

Servo motors have three wires: power, ground, and signal. The power wire is typically red, and should be connected to the 5V pin on the Arduino or Genuino board. The ground wire is typically black or brown and should be connected to a ground pin on the board. The signal pin is typically yellow or orange and should be connected to pin 9 on the board.

The potentiometer should be wired so that its two outer pins are connected to power (+5V) and ground, and its middle pin is connected to analog input 0 on the board.

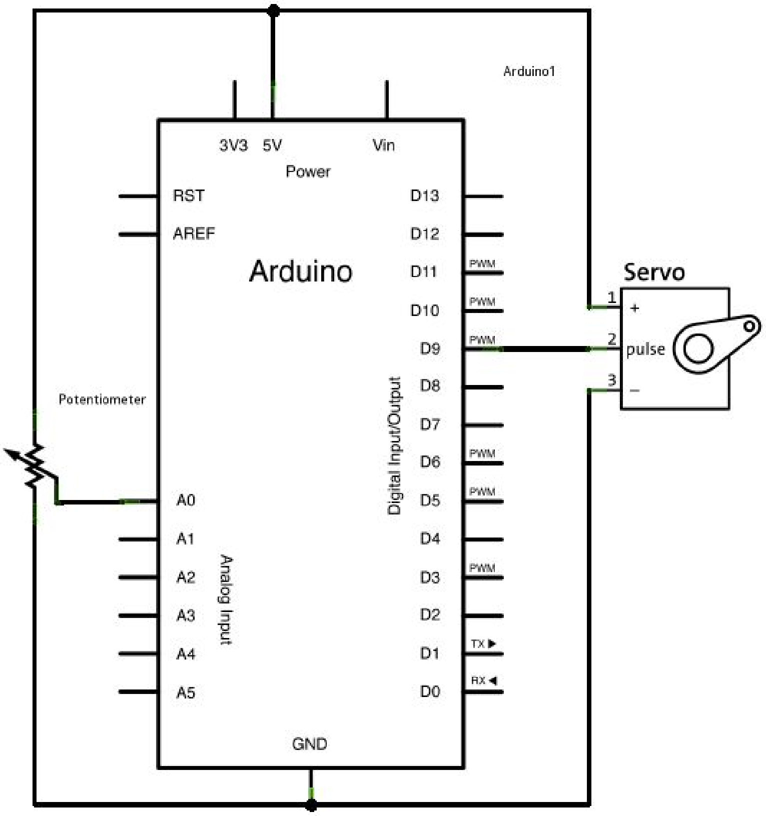


Figure 3.1: Hook up schematic for potentiometer and servo

### Section 3.2: Code for reading potentiometer and setting servo position.

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo

int potpin = 0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  val = analogRead(potpin);            // reads the value of the potentiometer (value between
0 and 1023)
  val = map(val, 0, 1023, 0, 180);     // scale it to use it with the servo (value between 0
and 180)
  myservo.write(val);                  // sets the servo position according to the scaled value
  delay(15);                           // waits for the servo to get there
}
```