Lab – Direct Current Measurement

Objective – Apply Ohm's Law to predict current flow through an LED based on changing voltage and implement Arduino code to use the onboard analog to digital converter to measure and validate the predictions.

Materials –

LED, Two 100 Ω Resistors, Arduino Uno, Breadboard, Jumper-wires, Digital multimeter (DMM)

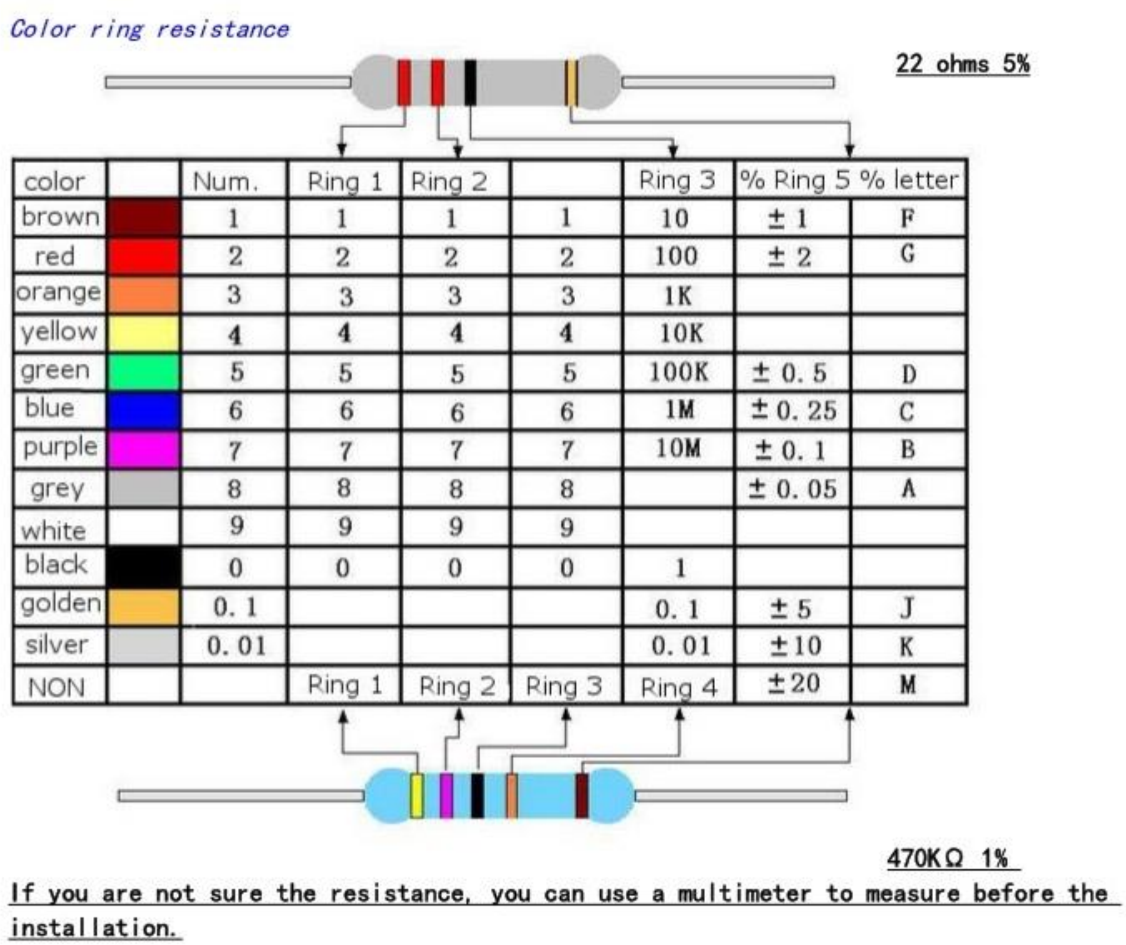
Relevant Equations -

$V = IR$ (Ohm's Law)

$0 = V_1 + V_2 + \cdots + V_n$, in a closed loop (i.e., all voltages add)

$V_{diode} = V_{LED} \approx 0.7V$

Table 1: Resistor Color Code Chart



color		Num.	Ring 1	Ring 2		Ring 3	% Ring 5	% letter
brown		1	1	1	1	10	± 1	F
red		2	2	2	2	100	± 2	G
orange		3	3	3	3	1K		
yellow		4	4	4	4	10K		
green		5	5	5	5	100K	± 0. 5	D
blue		6	6	6	6	1M	± 0. 25	C
purple		7	7	7	7	10M	± 0. 1	B
grey		8	8	8	8		± 0. 05	A
white		9	9	9	9			
black		0	0	0	0	1		
golden		0. 1				0. 1	± 5	J
silver		0. 01				0. 01	±10	K
NON			Ring 1	Ring 2	Ring 3	Ring 4	±20	M

Background –

To start this lab, you should review topics from Electronics and Sensors concerning Ohm's Law, as well as a brief overview of how an analog to digital converter (ADC) works. The following will provide a brief overview of these topics to prepare you for success in this lab.

Ohm's Law and* Kirchhoff's Voltage Law (*KVL):

There are three main values which can be used to define nearly any electrical circuits: Current, Voltage, and Resistance (impedance). In many cases, these are described with a waterflow analogy where current is the rate of flow, voltage is the pressure, and resistance is the diameter of the pipe. This description is not altogether wrong, but it is important that the terms are understood in terms of energy. The following are definitions for the three terms, with appropriate units:

Voltage – Amount of charge energy carried [Joules per Coulomb (J/C); in Volts (V)]

Current – Number of charges flowing per unit time [Coulombs per second(C/s); in Amps (A)]

Resistance – Energy loss [in Ohms (Ω)]

Coulomb – Electrical charge [in Coulombs (C)]

When a charge passes through a resistance, there is a certain amount of energy lost based on a reduction in the current flowing through the resistance. This loss can be determined by Ohm's Law,

$$\mathrm{V} = \mathrm{IR}.$$

The law says that if there is a current, I, passing through a resistance, R, there will be a drop in voltage, V, across the resistance. The equation can also be solved for R or I, given that you know the values of the missing variables (it's a linear relationship 😊).

In electronics lab, we generally work with discrete components of voltage, such as batteries and power supplies, and resistance, in the form of fixed resistors. Voltage sources are sources for potential energy which can be provided to a circuit or other system. Just like other types of energy, the property of conservation of energy applies here, in that energy cannot be created or destroyed. Thus, any energy provided to a system, for example by a battery to a circuit, must be consumed by that circuit if current is flowing (remember, must form a closed loop or 'circuit' for this to happen). This gives rise to Kirchhoff's Voltage Law. In a closed loop, the total potential energy provided (batteries) must be consumed (resistors or other components) when the circuit is closed (loop). The law is formalized as

$$0 = \mathrm{V}_1 + \mathrm{V}_2 + \cdots + \mathrm{V}_\mathrm{n}.$$

Let us look at a simple circuit as an example.

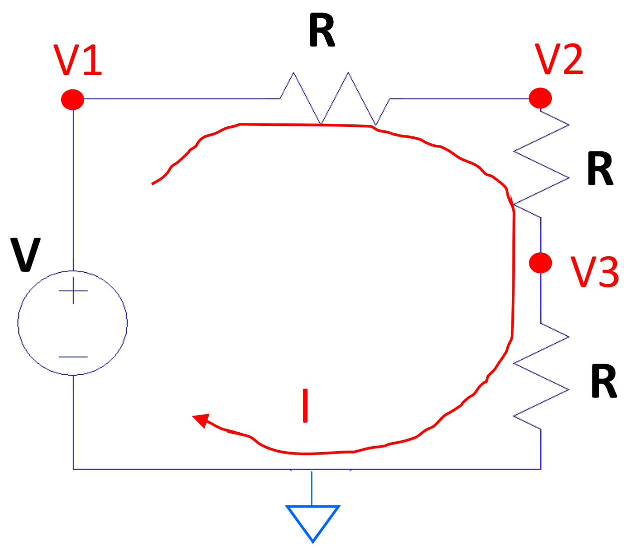


Figure 1: Simple Circuit Diagram

Assume V=9V, R1=R2=R3=100 Ω. We are looking to find the current (I in red) through the circuit, and the resulting voltages at each point along the circuit. The triangle at the bottom of the circuit represents 'ground', or 0V. It is used as a reference point in electrical circuits.

In figure 1, each R represents a 'voltage drop' meaning that energy is dissipated by the resistor. The total energy drop of all the R's in the circuit equals the total energy applied to the circuit. Thus, we start with 9 V (+ terminal of battery) and end at 0 V (at the triangle at the bottom of the circuit which is also the – terminal of the battery). So, each resistor drops some voltage and Ohm's Law tells us how to compute this. Before we do that there is a very important thing to understand – the current flowing in a closed circuit is CONSTANT (it does not change). That is to say the SAME CURRENT FLOWS THROUGH EACH R in a closed circuit. So how much voltage is 'dropped' by each resistor? The answer is 1/3 since they are equal in value. Thus 3 V is dropped in each resistor.

Now we can see what the voltage is at each point in the circuit, marked as V1, V2, and V3 (in red) on the schematic. If each R drops 1/3 of 9V, or 3V each, then V1 = **9V**, V2 = 9-3 or **6V** and V3 = 6-3 or **3V**. It's that easy. If the R's are not equal the calculation is a bit more involved, but the same basic principle applies (do what I do in that case – use a simple circuit simulator like pSPICE to compute the values for you!). One last caution – V is relative, so the values are 'across' the R terminals meaning we put our multimeter leads 'across' the resistor to measure the drop. If we, instead, put the negative lead of the multimeter at ground and the other lead where V1, V2, and V3 is shown you will get the numbers above. MEASURE BOTH WAYS and you will understand how this works – it's easy but important that you see this for yourself.

How much CURRENT is flowing in this case? Easy! Using Ohms Law V=IR or I=V/R = 3V/100Ω = 0.03A or 30 mA. So, I = **30 mA** and the same current flows through the entire loop.

Analog to Digital Conversion:

Analog to digital conversion serves to convert a continuous number into one which is readable by computers. This involves splitting the value into discrete 'steps', the size of which are determined by the size of the converter. The size is generally represented by a number of ‘bits’, or binary digits, used to represent the number. For example, the Arduino you will be using in the lab has a 10-bit ADC. A 10-bit number can be used to represent 1023 individual steps ($2^{10} - 1$, since one of the steps is 0).

To convert back to a readable number, the computer requires a reference number, which represents the highest value (1023 in this case) of the ADC. By default, the Arduino reference value is 5.0. So, a value of 1023 in binary corresponds to 5.0V at the Arduino analog pin. Likewise, a value of 223 in binary would correspond to a value of $5.0 \cdot (223/1023) = 1.09\text{V}$ at the analog pin. This method can be used for any size ADC so long as the reference and size are known.

Circuits –

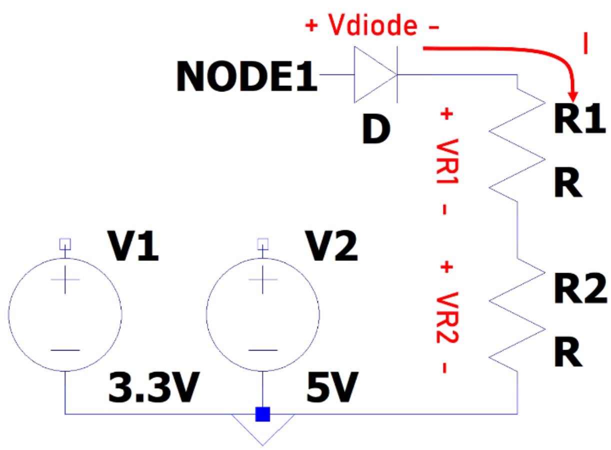


Figure 2a: Circuit diagram with two voltage sources.

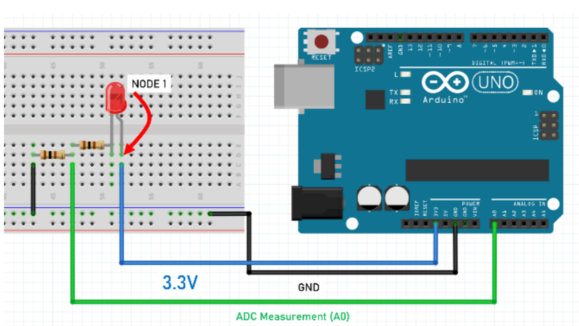


Figure 2b: 3.3V Breadboard Layout

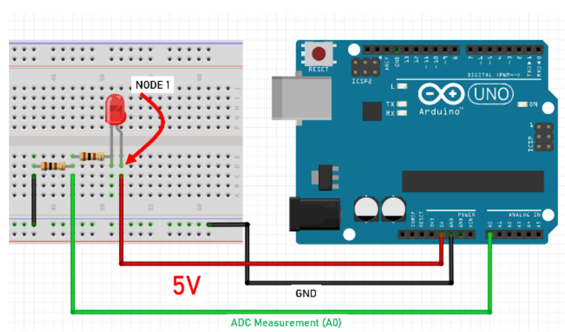


Figure 2c: 5V Breadboard Layout

Figure 2b and c: Physical layout of the circuit with two possible voltage sources. Note voltage sources is connected by placing a jumper wire between each voltage source one at a time at NODE1. Observe how the circuit (i.e., the LED, light emitting diode) responds.

Lab Exercise

1) Calculate the expected current and voltage as shown in Figure 2b circuit diagram if a 5V is applied to the circuit at NODE 1. Assume V_{diode} = 0.7V (note this is an assumption, value varies for each type of diode) and is a sink (drops voltage like a resistor). For example, if 3.3V is applied, the current and voltage at Node 1 would be calculated as follows:

   R1 and R2 are each 100Ω resistors. So

$$V_1 - V_{LED} - V_{R1} - V_{R2} = 0$$

$$3.3V - 0.7V - (100\Omega) * I - (100\Omega) * I = 0V$$

   Solving for I, $\quad 2.6V = (200\Omega)I \text{ or } I = 2.6V/200\Omega = 0.013 \text{ Amps} = 13 \text{ mA}$

   Calculating the voltage across each resistor using Ohm's Law,

$$V_{R1} = V_{R2} = (100\Omega)I = 1.3 \text{ Volts.}$$

2) What voltage would you expect to see across R_2 (i.e., V_{R2}) when 5V is applied at NODE 1? Using information from 1a, we can see that the voltage across R_2 is $V_{R2} = 1.3$ Volts when 3.3V is applied.
3) Do you expect to see any changes in brightness when different voltages are applied? Why or why not?

Build the circuit in Figure 2b using the materials listed above.

4) Ensure the Arduino is disconnected from your laptop/PC before beginning.
5) Populate the breadboard with the LED and Resistors as shown in Fig. 2b
6) Connect the breadboard and Arduino as shown in Fig. 2b
7) Connect the Arduino to your laptop/PC and upload the code (Code can be found on next page).
8) Once the code has been successfully uploaded, open the serial monitor (Tools -> Serial Monitor, or Ctrl+Shift+M)
9) Ensure that the baud rate matches what was entered in the Serial.begin() statement in your code (9600 is default)

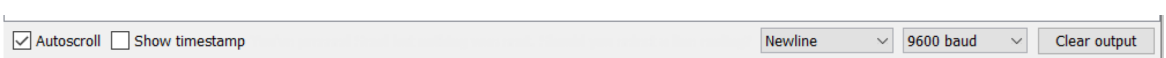


Figure 3: Serial baud rate output

10) You should start getting data into the serial monitor. Be aware that until you apply a voltage to NODE1, this data is just noise and not valid.
11) Using a spare jumper wire, connect the 3.3V line to NODE1 and collect at least 5 data points for the current, and V_{R2}. Use your multimeter to measure the voltage across R_2. How does your measurement compare to the Arduino printout? Take a screenshot of the serial monitor.
12) Make a note of the brightness of the LED for comparison.
13) Disconnect the 3.3V line from NODE1
14) Repeat steps 10-12 for the 5V line using the connections in Fig. 2c
15) Submit one photo of your breadboard and Arduino connections with the LED turned on.
16) Submit your Arduino printout from the serial monitor in step 11 for 3.3V and 5V.
17) Submit results from steps 1-3 and 15-16

Copy and Paste the following Arduino code

```
/* Circuit Lab Arduino Code - Alex Otten USF 10/18/20
 *  -- Code uses Arduino ADC in conjunction with a voltage divider
 *  to determine the current flowing through a resistor + LED circuit.
 *  Used to validate predictions made using Kirchhoff's Voltage Law and
 *  Ohm's Law. R1 and R2 can be made of any series or parallel combination
 *  of resistors.
 */

// The pin used for reading analog signals
int analogPin=A0;

// Variable to store the bit-code
int analogValue;

// Baud rate for communications
int baudRate=9600;

// Reference voltage used by the ADC
float vRef=5.0;

// Number of unique codes the ADC can produce
float numStep=pow(2,10)-1;

// Conversion rate between the ADC code and volts
float vStep=vRef/numStep;

// Variable to store the resulting voltage value
float volts;

// Variable to store the current value (assumes
// 100 Ohm resistor
float current;

void setup() {
  // Starts communication between the Arduino and
  // your PC
  Serial.begin(baudRate);
}

void loop() {
  // Gets the analog code from the ADC
  analogValue=analogRead(analogPin);

  // Translates the ADC code to volts
  volts=analogValue*vStep;

  // Ohms law to find current
  current=volts/100;
```

```
  // Serial communication with the PC
  // println prints whatever is in "" followed
  // by a new line (think pressing enter), while
  // print prints whatever is in "" without
  // starting a new line
  Serial.println("----------");
  Serial.print("Voltage: ");

  // Using "" causes the exact typed words/numbers
  // to be printed, whereas putting a variable
  // without "" causes whatever number is in that
  // variable to be printed
  Serial.print(volts);
  Serial.println(" V");

  Serial.print("Current: ");
  Serial.print(current*1000);
  Serial.println(" mA");
  Serial.println("----------");

  // Causes the Arduino to pause for 500ms
  // Good to do to avoid sending too many messages
  // too quickly to the PC
  delay(500);
}
```