Introduction to Arduino:
The Arduino is a relatively inexpensive, yet versatile open-source microcontroller. It is designed to facilitate interaction with the physical world via sensors while being able to perform calculations and various functions. The Arduino can be connected to a computer via a USB cable and programmed using a simplified version of the C programming language, and it has both analog and digital pins from or to which it can read or write values. The maximum voltage that it is able to supply is 5V; thus, a “HIGH” digital pin corresponds to 5V, while a “LOW” digital pin corresponds to 0V. There are many “shields” and sensors that are designed for interaction with the Arduino, or microcontrollers in general. The Arduino can read values from sensors or other inputs, and it can also write values to other components based on computations given by the program.

Arduino is fast becoming one of the most popular microcontrollers on the market. Its ease of use, extensive software library and most importantly, its low cost (~$30 for a basic set compared to $150 for other microcontrollers) have come to make it as popular as it is today. Many projects using the Arduino can be found on www.hackaday.com.

In order to start having fun with the Arduino, free software can be found at: http://arduino.cc/en/Main/Software for Macs, Windows and Linux operating systems. This website also provides tons of easy tutorials for you to start. Tutorials can be found at: http://arduino.cc/en/Tutorial/HomePage

In this introductory lab, you will become familiar with the Arduino and some of its applications. First, you will learn to use a breadboard to build simple circuits. You will also apply Ohm’s Law to analyze your circuits. You will then use the Arduino to interface between hardware (your circuits) and software (the code).
**Procedure:**

***Answer all questions on the handout to be turned in at the end of lab***

1. **Light up an LED**

   - Pick out an LED (Light-Emitting Diode) and a 1kΩ resistor.
   - Measure the resistance of the resistor using the Digital Multimeter (DMM). Find the appropriate cables from the red box at your station and connect them as shown in Figure 3 below. Turn on the HP 34401A DMM at your station. To measure resistance, press the button with the omega symbol Ω (third button from the left in the top row).

   ![Figure 3: Measuring resistance with the HP 34401A DMM](image)

   You will now build a very basic circuit on a **breadboard**, also called a **protoboard**. The breadboard shield that we are using, shown in Figure 7, features 30 rows of holes arranged in sets of five on each side of the partition. Each of the five holes in a row is electrically connected by a conducting plate running underneath. Therefore, if you were to connect a wire from 5V to a hole, anything connected to the other four holes in that row would be at a potential of 5V.

   - Build the circuit shown in Figures 5 and 6. Consult Figure 7 for clarification.
   - Use the pin labeled 5v as your voltage source (+5V).
   - Use one of the pins labeled Gnd as ground (0V).
   - **The longer leg of the LED is the positive end, so make sure that end is connected to the voltage source.**

   ![Figure 4: LED Schematic](image)

   *(Source: http://www.societyofrobots.com/electronics_led_tutorial.shtm)*
- You should *always* put a resistor in series with an LED in order to limit the current flowing through the LED (and thereby prevent it from burning out).
- **Always make sure no bare wires are touching!**

- Connect the Arduino (with the breadboard shield attached) to the computer using the USB cable. This will supply power to your circuit. Your LED should now be lit!
- How much current is flowing through the LED in your circuit (assuming the LED has no resistance)? Hint: Remember Ohm’s Law! $V = IR$
- What happens if you change the value of $R1$?

Circuits are generally designed to accomplish some sort of fixed task, such as blinking a light or amplifying a sound signal. Often, however, you will want to add user input to your circuit, so that you can change its behavior on the fly (for example, setting the blink rate or adjusting volume). You will now examine two simple ways of controlling the behavior of your circuit.
2. Adjust the brightness of the LED with a photoresistor

A photoresistor, sometimes called a photocell, is a form of variable resistor whose resistance decreases with exposure to light. In effect, a photoresistor is a light sensor.

- Obtain a photocell, and measure its resistance as you vary its exposure to light.
- Approximate the maximum and minimum resistances it can achieve. Record these values.
- Replace the 1kΩ resistor in your circuit from Figure 5 with the photoresistor.
- Use your hand to adjust the amount of light exposed to the photoresistor, and observe the effect that it has on the LED.

3. Adjust the brightness of the LED with a potentiometer

A more common way to provide user input to a circuit is by using a potentiometer, or pot for short. You have all used a potentiometer at some point, probably to adjust the volume of some device. A potentiometer looks like this:

![Potentiometer Schematic and Image](image)

A potentiometer is a three-terminal device. The resistance between the two outermost terminals, R1-3, is constant (this constant value is in fact written on the side of the potentiometer), but the center terminal (2) acts as a “wiper” that slides back and forth along the resistor as you rotate the potentiometer. Therefore, R1-2 and R2-3 change, but their sum R1-2 + R2-3 = R1-3 is constant.

- Remove the power supply from the circuit (unplug the USB cable).
- Obtain a 50kΩ potentiometer. A potentiometer (pot) is a variable resistor; you can vary the resistance of a 50kΩ pot from 0 to 50,000 ohms.
- Using the DMM, measure the resistance across the two end terminals (1 and 3) of the potentiometer. Record this measurement.
- Now measure the resistance between one end terminal (1 or 3) and the middle terminal (2). Observe what happens when you adjust the potentiometer with your screwdriver.
- If the resistance between the 1st and 2nd terminal is 10kΩ, then what is the resistance between the 2nd and 3rd terminal?
- Add the potentiometer to the circuit as shown in Figure 10/11. Note that the 1kΩ resistor is still connected. Why must this be?
Reconnect the power supply to the Arduino. Use a screwdriver to vary the resistance of the pot. What happens to the intensity of the LED?

When the pot is at 50% (rotated halfway), what is the current through the LED? First, calculate this current using Ohm’s law, assuming no LED resistance.

Then, measure this current with the DMM. Connect the red lead to the port on the DMM marked “I”, and enter the ammeter mode by pressing “Shift” and “DC V” (shown in Figure 12). Unlike measuring resistance or voltage, to measure current, you have to connect the DMM in series at the point of measurement. So, to measure the current through the LED, remove one side of the LED out of the circuit. Then, use the DMM leads to complete the circuit you just broke. This will give you the current through that part of the circuit. Since this is a simple series circuit, it doesn’t matter where in the circuit you insert the DMM. If you’re having trouble understanding this step, ask the TA.
4. **Flashing the LED using the Arduino**

Now you will upload a code to the Arduino to make an LED flash on and off.

Open the Arduino IDE (see the [Arduino Software Instructions document](#) for help on using the Arduino software and uploading code). Copy the following code into the window, or go to Files → Examples → Basic → Blink:

```java
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeatedly.

This example code is in the public domain.
*/

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);  // set the LED on
  delay(1000);            // wait for a second
  digitalWrite(13, LOW);  // set the LED off
  delay(1000);            // wait for a second
}
```

Notice that text following two slashes // or between /* */ is grayed-out and is not read by the Arduino; these are known as “comments,” and they serve to assist the reader in understanding the code. Also note that **time is specified in milliseconds**.

- Take a moment to read the code and understand what each line achieves.
- Upload the code onto your Arduino (see the Arduino Instructions document on Blackboard).
- Verify that the LED flashes correctly. **Note:** the red, on-board LED next to digital pin 13 will flash, not the LED from the circuit you’ve just built!
- Modify the code so that the LED flashes at a frequency of 1 Hz (1 cycle per second).
  - Hint: it is now flashing at .5 Hz.
- Disassemble your circuit and build the circuit from Figure 5 (part 1), except **connect the positive lead of the LED to digital pin 12**. Notice that there is a ground pin right next to pin 13, so you can connect your resistor here.
- Modify the code so that your LED flashes instead of the on-board LED.
- Modify the code so that the two LEDs flash in an alternating pattern, i.e. when one is on, the other is off. **Show a TA!**
5. Using a potentiometer to change the flashing rate

Construct the circuit shown in Figures 13 and 14. You will need to use a few jumper wires to connect the potentiometer to the pins.

Copy the following code into the window and upload to the Arduino:

```c
/*
Blink with variable rate
Turns an LED on and off repeatedly at a rate varied by a potentiometer.
*/

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

void setup() {
    pinMode(12, OUTPUT);  // initialize the digital pin as an output
}

void loop() {
    val = analogRead(analogPin);  // set val equal to the value read from analog pin 0
    digitalWrite(12, HIGH);       // set the LED on
    delay(val);                  // wait for val milliseconds
    digitalWrite(12, LOW);       // set the LED off
    delay(val);                  // wait for val milliseconds
}
```
- Take a moment to read and understand the code. Note that the variables `analogPin` and `val` can have almost any name you’d like them to have; they are just used to store values.
- The potentiometer acts as a voltage divider in this circuit. When you rotate the pot, the voltage on analog pin 0 varies from 0 to 5V. The `analogRead` function reads the voltage on the specified pin to a value between 0 and 1023, with 1023 corresponding to 5V.
- When the potentiometer is at 50% (rotated halfway), what is the voltage on analog pin 0? What is the value of the variable `val`?

6. Conditional LED lighting

Keep the circuit from Figure 13 on your breadboard and upload the following code:

```cpp
int analogPin = 0; // analog pin used to connect the potentiometer
int val; // variable used to store the value from the analog pin

void setup() {
  pinMode(12, OUTPUT); // initialize the digital pin as an output
}

void loop() {
  val = analogRead(analogPin);
  if(val >= 512 && val < 1000) {
    digitalWrite(12, HIGH);
    delay(250);
    digitalWrite(12, LOW);
    delay(250);
  } else if(val >= 1000) {
    digitalWrite(12, HIGH);
  } else {
    digitalWrite(12, LOW);
  }
}
```

- Describe what the code does. You may find these reference links helpful if you are having trouble:

- **Extra Credit**: If you have time, modify your circuit and code so that the LED turns on only when you cover a photoresistor with your finger. The LED should remain off otherwise. Hint: you will need to use another resistor in your circuit to create a voltage divider:
- Show a TA your extra credit!