Skill Demo 1: Arduinos, Ohm's Law, LEDs, and Breadboards

Fall 2022

Goals:

- Set up your Arduino
- Program your Arduino to perform a time-based action.
- Use Ohm's Law to evaluate a circuit.
- Use Ohm's Law to design a circuit.
- Learn how to use an LED in a circuit.
- Learn how to select a resistor based on the requirements of other components in a circuit.
- Learn how to use a breadboard.

Background:

Reading

There are some chapters that explain many of these concepts in detail in the Practical Electronics for Inventors book:

- Theory chapters on current, voltage, resistance, DC power, Ohm's Law
- Hand-on Electronics on schematics and multimeters

Sparkfun Tutorials

Multimeters
How to Use a Multimeter (handy 4-page pdf)
LEDs
Breadboards
Resistors
Ohm's Law

CS 3651 YouTube videos (for reference)

(You may want to watch ahead of time for this skill demo)
CS3651 - Intro to Multimeters
CS3651 - Using Multimeter in a Circuit
CS 3651 Introduction to Circuit Schematics
CS 3651 Introduction to Resistors
Intro to LEDs
Pull up and pull down resistors
Introduction to Breadboard (Protoboards) (if don't remember from ECE2031)

Install the toolchain and connect your Arduino

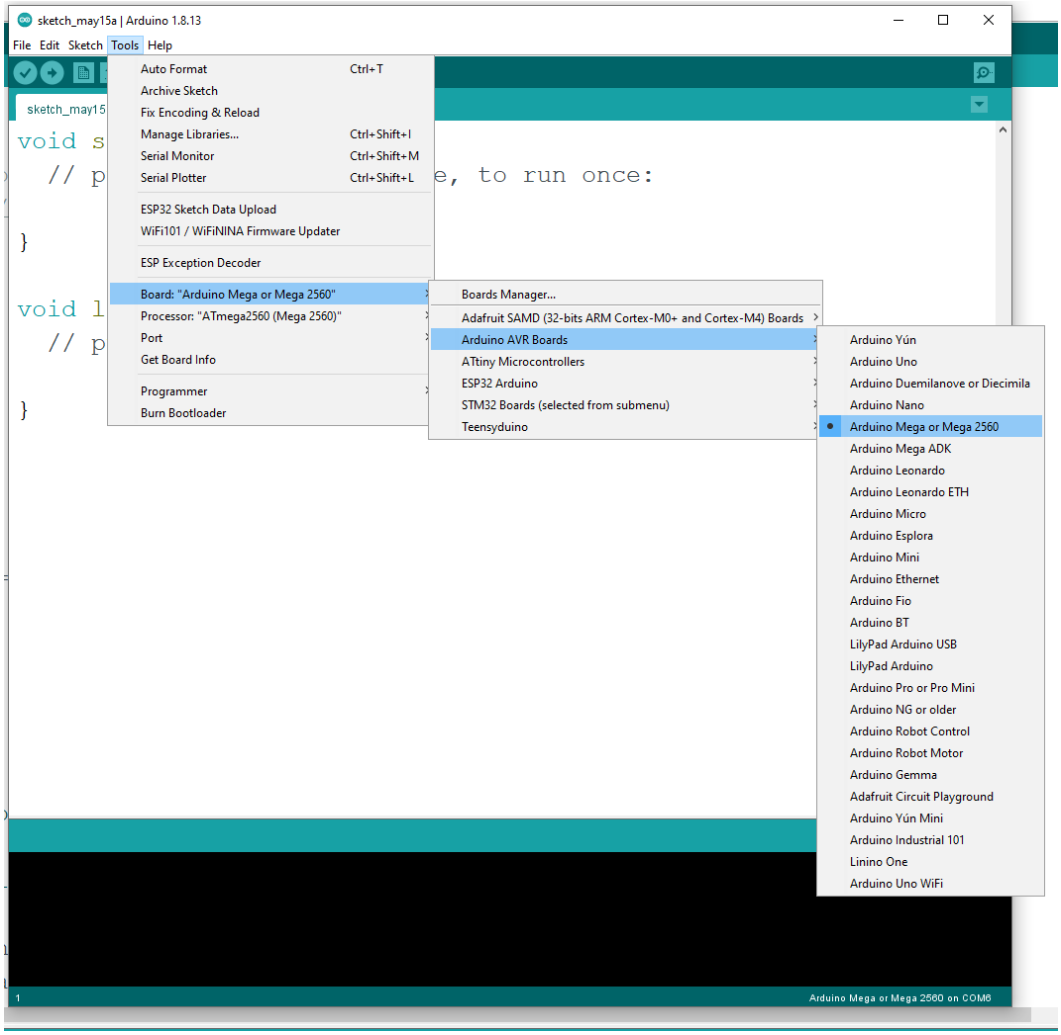
Parts

- Arduino Mega 2560
- Your USB cable

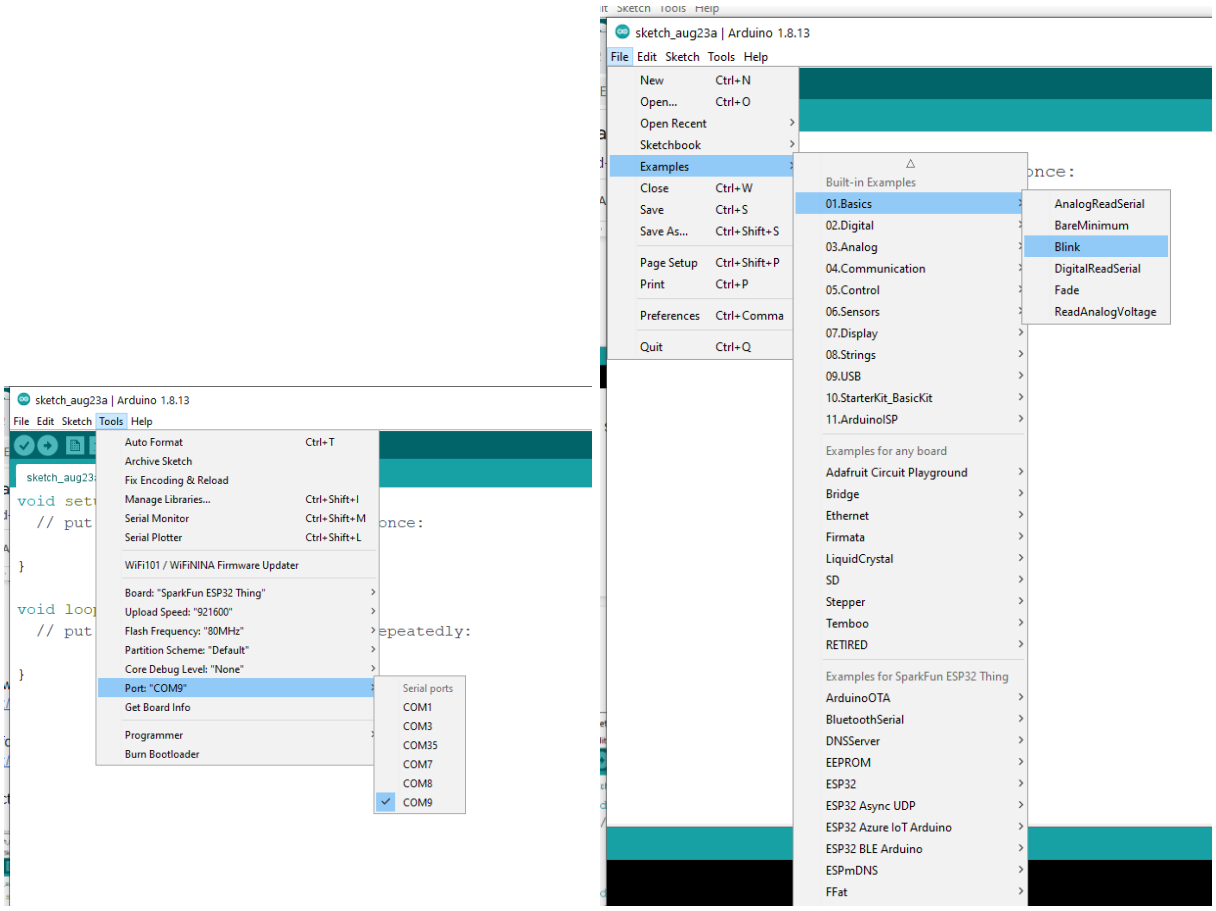
Follow these instructions here to install the drivers for your Arduino.
https://learn.sparkfun.com/tutorials/how-to-install-ftdi-drivers

And follow this to install the Arduino software.
https://www.arduino.cc/en/Main/Software

Select Arduino Mega 2560 as the target board under tools

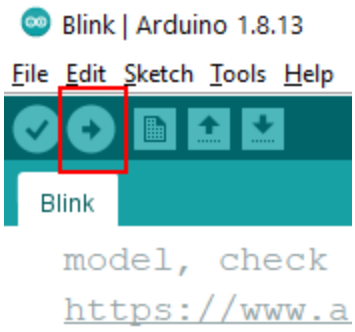


Then select the port for the Arduino under tools: (it will likely be the highest numbered port).



Now let's compile and download the blink example to the Redboard / Arduino Uno.

Select File->Examples -> 01 Basics -> Blink as indicated in the above screenshot.



Then click the upload button at the top left. You should see something that looks like this in the bottom Arduino pane:

```
:avrdude: Version 6.3-20190619
         Copyright (c) 2000-2005 Brian Dean, http://www.bdmicro.com/
         Copyright (c) 2007-2014 Joerg Wunsch

         System wide configuration file is
"C:\Users\Peter\AppData\Local\Arduino15\packages\arduino\tools\avrdude\6.3.0-arduino17/etc/avrdude.conf"

         Using Port                    : COM5
         Using Programmer              : arduino
         Overriding Baud Rate          : 115200
         AVR Part                      : ATmega328P
         Chip Erase delay              : 9000 us
         PAGEL                         : PD7
         BS2                           : PC2
         RESET disposition             : dedicated
         RETRY pulse                   : SCK
         serial program mode           : yes
         parallel program mode         : yes
         Timeout                       : 200
         StabDelay                     : 100
         CmdexeDelay                   : 25
         SyncLoops                     : 32
         ByteDelay                     : 0
         PollIndex                     : 3
         PollValue                     : 0x53
         Memory Detail                 :

                                  Block Poll               Page                       Polled
           Memory Type Mode Delay Size  Indx Paged  Size   Size #Pages MinW  MaxW   ReadBack
           ----------- ---- ----- ----- ---- ------ ------ ---- ------ ----- ----- ---------
           eeprom        65    20     4    0 no       1024    4      0  3600  3600 0xff 0xff
           flash         65     6   128    0 yes     32768  128    256  4500  4500 0xff 0xff
           lfuse          0     0     0    0 no          1    0      0  4500  4500 0x00 0x00
           hfuse          0     0     0    0 no          1    0      0  4500  4500 0x00 0x00
           efuse          0     0     0    0 no          1    0      0  4500  4500 0x00 0x00
           lock           0     0     0    0 no          1    0      0  4500  4500 0x00 0x00
           calibration    0     0     0    0 no          1    0      0     0     0 0x00 0x00
           signature      0     0     0    0 no          3    0      0     0     0 0x00 0x00

         Programmer Type : Arduino
         Description     : Arduino
         Hardware Version: 3
         Firmware Version: 4.4
         Vtarget         : 0.3 V
         Varef           : 0.3 V
         Oscillator      : 28.800 kHz
         SCK period      : 3.3 us

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e950f (probably m328p)
avrdude: reading input file "C:\Users\Peter\AppData\Local\Temp\arduino_build_803167/Blink.ino.hex"
avrdude: writing flash (924 bytes):
```

```
Writing | ################################################## | 100% 0.38s

avrdude: 924 bytes of flash written
avrdude: verifying flash memory against
C:\Users\Peter\AppData\Local\Temp\arduino_build_803167/Blink.ino.hex:
avrdude: load data flash data from input file
C:\Users\Peter\AppData\Local\Temp\arduino_build_803167/Blink.ino.hex:
avrdude: input file C:\Users\Peter\AppData\Local\Temp\arduino_build_803167/Blink.ino.hex contains 924
bytes
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 0.34s

avrdude: verifying ...
avrdude: 924 bytes of flash verified

avrdude done.  Thank you.
```

Now you should see the blue LED on the arduino blink. More later.

Introduction

This skill demo will teach you how to design a circuit, calculate the appropriate component values, and then build the circuit on your breadboard.

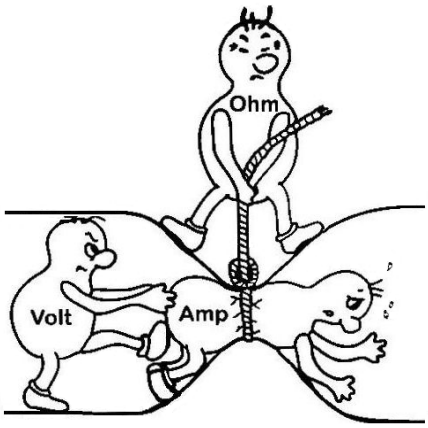
LEDs or light emitting diodes are small electronic devices that produce light when current flows in one direction. They are commonly used for indicator lights on electronic devices, backlights on LCD monitors, or even as room lights replacing incandescent or fluorescent bulbs. For more information on them, take a look at section 4.2 or 5.3 in Practical Electronics for Inventors.

Resistors resist the flow of electricity through them. Read through section Practical Electronics for Inventors 3.5 on resistors for more information. You could also watch **Introduction to Resistors**. For the theory behind voltage and current, read through the theory chapter of Practical Electronics for Inventors, specifically section 2.1 on current, 2.2 on voltage, 2.3 on resistance, 2.6 on electric circuits, 2.7 on Ohm's law.

Through-hole resistors (the ones with wires that stick in your breadboard) are identified by color codes. These colors identify the resistance in ohms of the resistor. For more details review this page on Sparkfun.

Ohm's Law

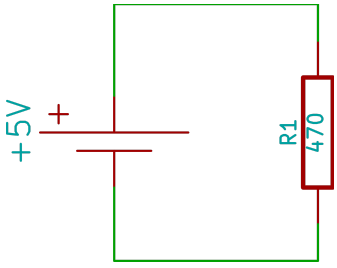
$$I = \frac{V}{R} \quad V = I * R \quad R = \frac{V}{I}$$



(courtesy http://www.sengpielaudio.com/calculator-ohmslaw.htm)

Ohm's law describes the relationship between voltage, current, and resistance in a circuit. If we know two of these parameters, Ohm's Law allows us to find the third parameter. (If physics is a distant memory Sparkfun has an excellent review of Ohm's law.) Note that Ohm's law only applies to things that behave like resistors.

Using Ohm's Law:



Consider the above schematic . We have a power source (V in Ohm's Law) and a resistor (R). Since we know these two parameters we can determine the current flow (I).

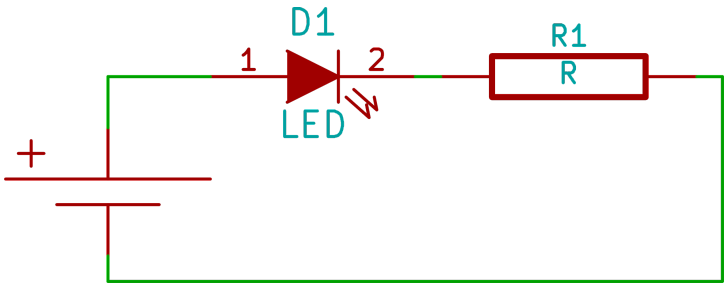
(answer these on the turn-in page)

1) What is the current flow through resistor R1 in amps?
(Remember I (amps) = V (volts) / R (ohms).)

This is a rather small number. Because of the scale, we often use milliamps to describe current in electronic circuits. An amp is a large amount of current useful for household appliances. Our battery operated devices typically usually use milliamps of current.

2) Convert this value in amps to milliamps (1 amp = 1000 milliamps)

Design a circuit to light up an LED



A very common use of a resistor is to limit the amount of current flowing through another part of a circuit. The circuit above is an example for an LED. In order to do that precisely, we need to know the voltage drop across the resistor.

LEDs, like all diodes, behave differently from resistors. They conduct very little unless they have more than their 'forward voltage' applied across them, and they try to conduct nearly infinite current if we apply much more voltage beyond that (this kills the LED). Thus LEDs are often used with current-limiting resistors in series with them. We can think of this red LED as always having the same voltage drop (the forward voltage) when current is applied.

In the lab we have 9V battery snaps, 6 AA battery holders, 4 AA battery holders, and also a number of bench power supplies based on a computer power supply. We also have Arduinos that are capable of giving 3.3V and 5V output.

For this skill demo we will use your Arduino 5V output as a power source.

3) What is the voltage of your power source? (Yes, copy it here).

LEDs have a parameter called a voltage drop. **The voltage drop for the red LEDs in the lab is 2 volts.** Therefore if 5 volts is applied to the anode or pin 1 of the LED, 3 volts will appear at the cathode or pin 2. This voltage drop remains mostly the same regardless of the amount of current flowing through the LED.

4) Since you know the voltage of your battery or other power source, and you know the voltage drop across the LED, what is the remaining voltage drop needed across resistor R1?

We can safely pass a maximum of 20 milliamps (mA) through our red LEDs. Any more, and the LED may get hot and burn out.

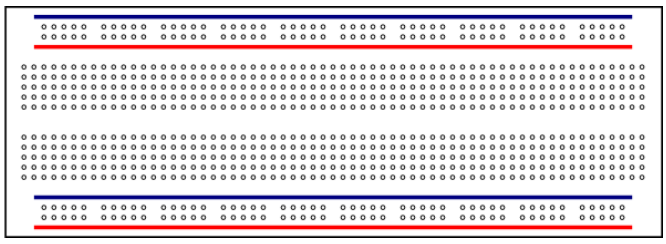
5) What resistance could we use for R1 to get exactly 10 mA of current through the LED?

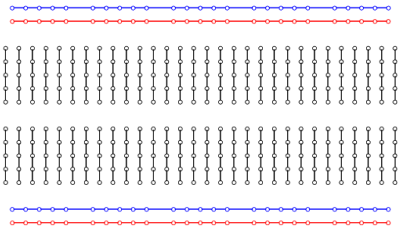
We can't get exactly the resistance that you calculated. Instead, we'll have to settle for a resistor that we have on hand in our resistor set. Select the closest resistance that won't let more than 10 mA through your LED. Make sure it will allow at least 2 mA through, otherwise the LED will be too dim to see.

6) What are a few of the closest values of the resistors in your kit? Which resistor did you end up selecting? How much current would it allow to flow through the circuit?

Now build the circuit on the breadboard

This is a breadboard:





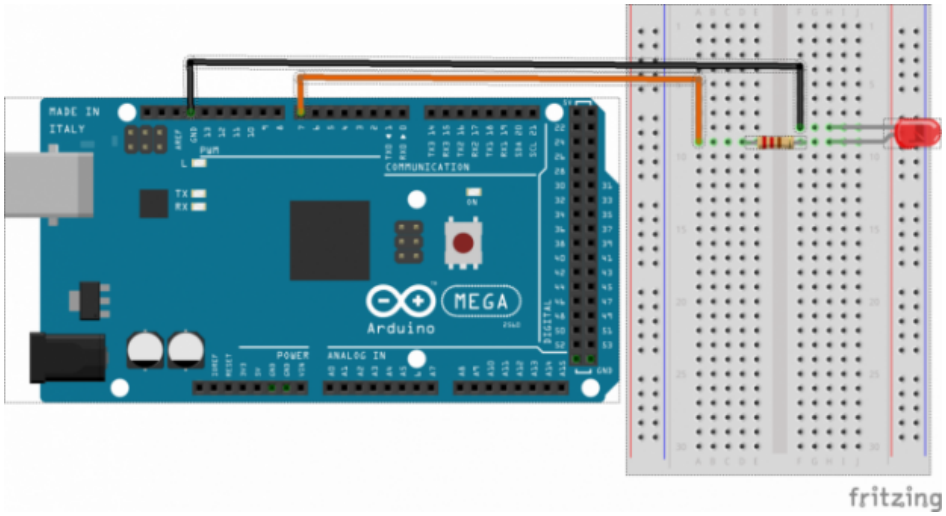
There are a series of holes in the breadboard that will accept small wires or electronic components. Groups of the holes are connected together to electrically connect the wires and components. The next image shows how the holes are connected. A solid line between holes indicates a connection.

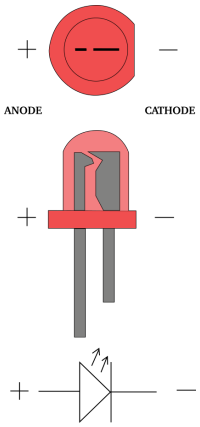
Plug the power source that you selected into the holes running down the long edge of the protoboard. DO NOT PLUG BOTH POWER SUPPLY WIRES INTO THE SAME SET OF CONNECTED HOLES! This will cause a short circuit and a very hot battery or voltage

regulator. Pick two different sets of holes, either running in parallel on the same side or on opposite sides of the protoboard.

The power source has a positive (+) and negative (-) terminal. Positive is often indicated by the red wire terminal and negative is indicated by the black wire. The two sets of connected holes that you have just plugged your battery into will become the positive and negative power rails. By convention electric current flows from positive to negative.*

* Actually the charge carrying particles, electrons, move from negative to positive, but Ben Franklin got it wrong in 1752, so “conventional current” flows from positive to negative.





Find the small red LED in your parts kit.

Next plug the resistor and LED into the protoboard. The resistor should connect from one power rail (positive or negative) to a set of 5 connected holes in the middle of the protoboard. The LED should connect from the other power rail to another hole in the set of 5 holes that the resistor is connected to. This is the circuit you have created:

Now plug the power source into the connector. Did the LED light up? If not you may have to reverse the direction of the LED.

Why might this have happened? LED or Light Emitting Diodes are directional. They will only conduct current and light up if the positive and negative legs (or anode and cathode) are hooked up to the positive and negative power rails.

Why do we have a resistor in the circuit? The purpose of the resistor is to prevent the LED from conducting too much current and destroying itself. More than that will cause the LED to heat up and eventually fail. Because of this, the LEDs that come with your Tinker Kit have resistors built-in.

Measure your LED's current

7) Now that you have your LED illuminated, measure the amount of current that is being drawn by the LED.

To do this, you'll need to break the circuit and connect the multimeter in **series**. You'll also need to move the multimeter probes to the 'COM' and 'mA' terminals on the multimeter to enable it to pass current, as well as switching it to the mA current mode.

8) Now write a program to blink your external LED with morse code "S-O-S". (You are welcome to use the Arduino blink example code as a starting point but change the output pin to something other than 13.)

DO NOT USE GPIO PINS 0 or 1!

S-O-S is a blink pattern of short, short, short, long, long, long, short, short, short. You can implement this with a series of digitalwrite() and delay() commands in the loop function.

These are the timing rules for Morse code:

- The length of a dot is 1 time unit.
- A dash is 3 time units.
- The space between symbols (dots and dashes) of the same letter is 1 time unit.
- The space between letters is 3 time units.
- The space between words is 7 time units.

Pick an appropriate unit between 100-300 milliseconds and implement a morse code flashing LED. Use the words space between complete sequences of SOS.

<u>Serial debugging</u>

Serial port output is how we debug code on limited devices like the Arduino. To implement serial port debugging do the following:

- Create a serial stream with serial.begin(115200)
- Use serial.print() and serial.println() to print out each character to the console as the LED blinks that character.
- Open the Arduino serial console and set the monitor speed to 115200 in the lower right corner. Verify that the characters are being printed as the LED blinks.