Lab 6: Arduino Uno

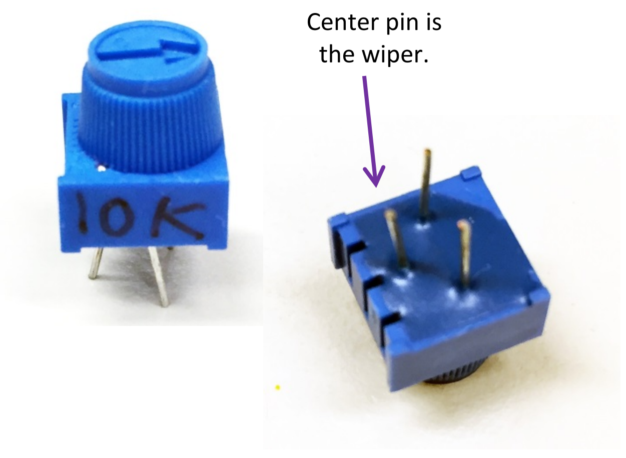
BMEn 2151 “Introductory Medical Device Prototyping”

Prof. Steven S. Saliterman

Exercise 6.1: Familiarization with Lab Box Contents

Objective: To review the items required for working with the Arduino Uno.

Arduino Uno R3 Microcontroller



Potentiometer (top and bottom)

Power Adapter for Arduino

Dupont Wire Jumpers
(Peel off what you need.)

Last Update 12/18/2023

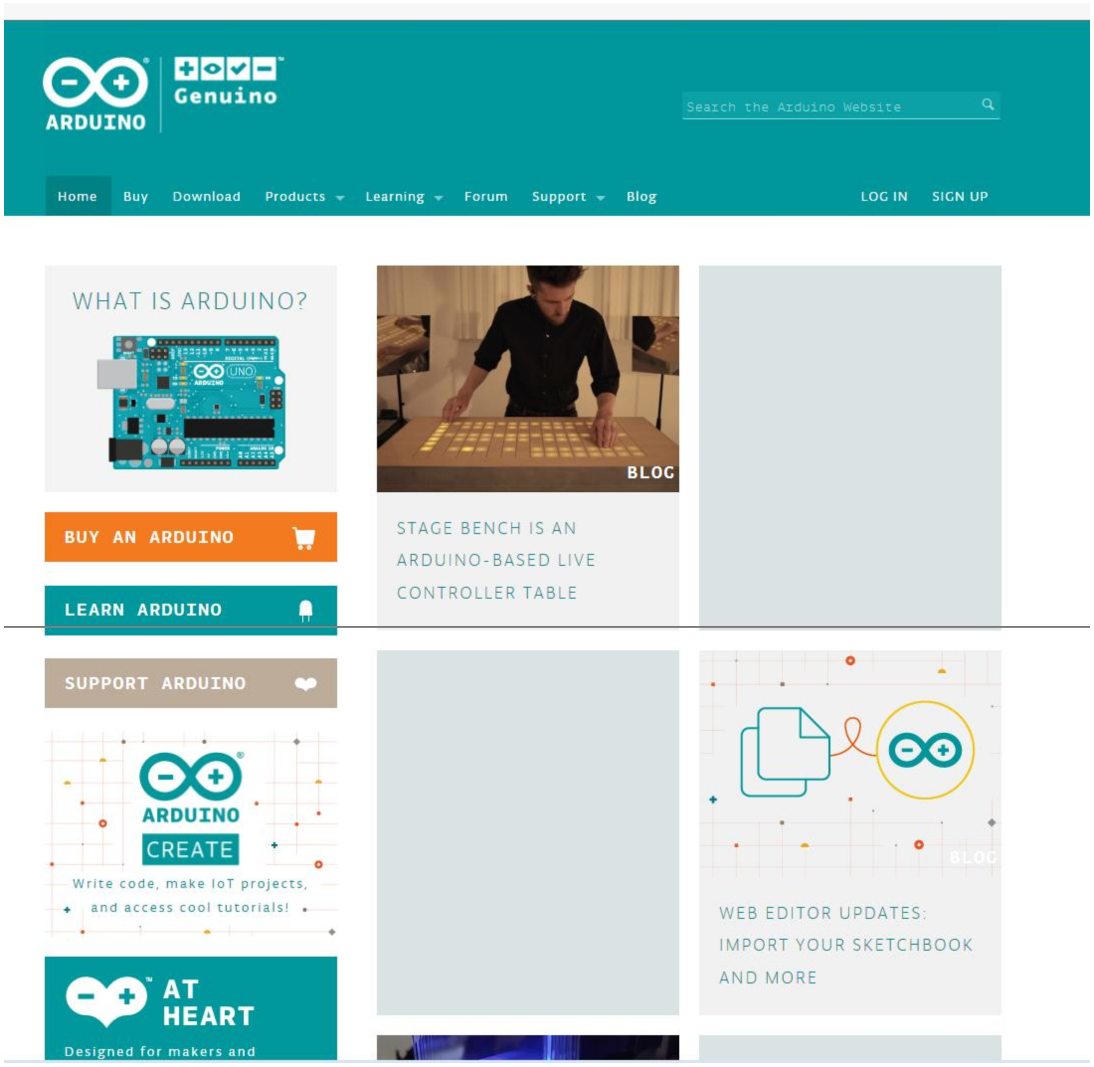
Exercise 6.2: Installing Arduino Software

Objective: To download and install the Arduino software for programming your Arduino Uno microcontroller. You will run a simple "sketch" or program to blink an LED.

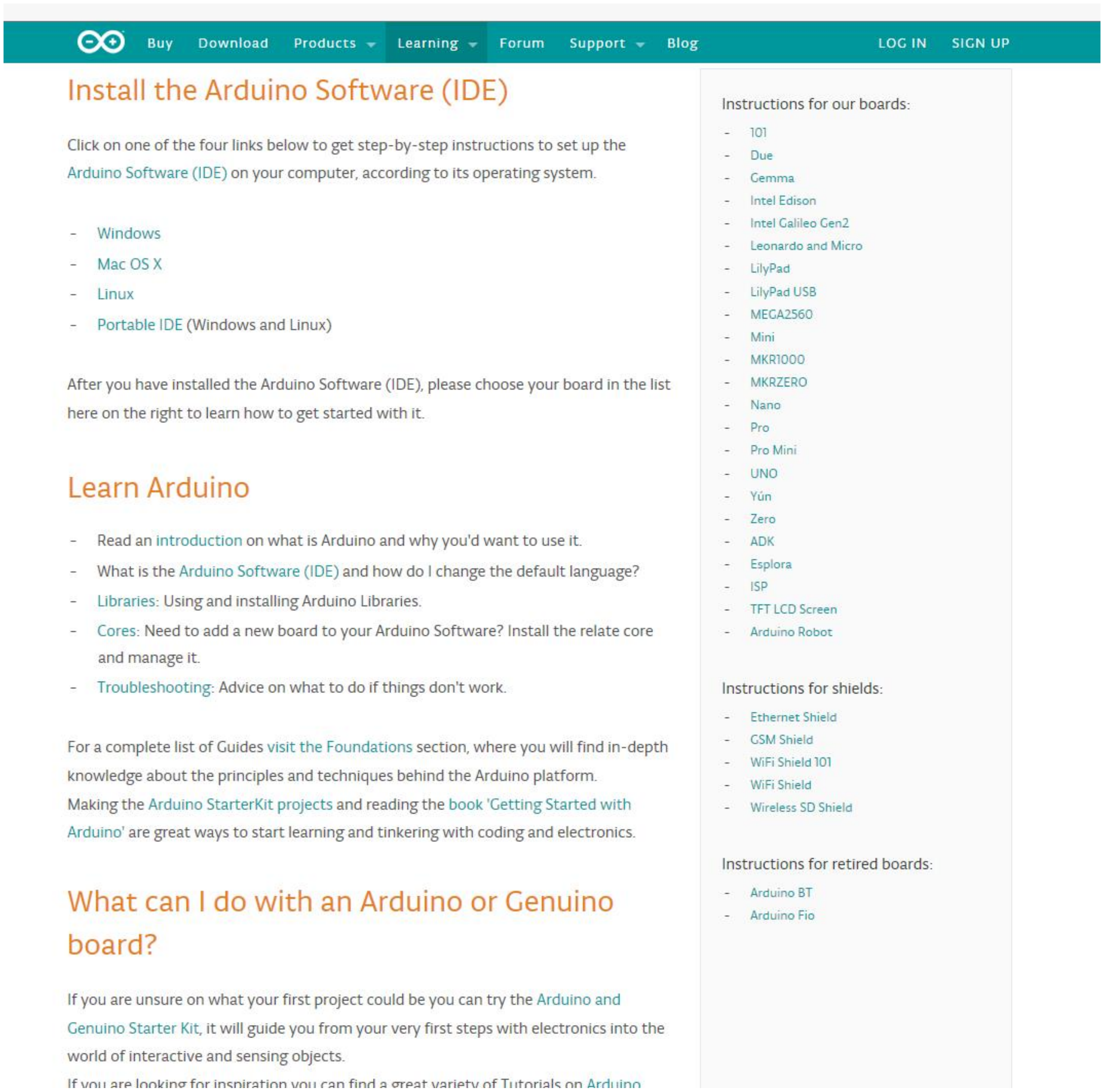
1. You will be connecting your computer to the Arduino via a USB 2.0 cable. This cable will provide power to the Arduino, and will allow you transfer programs from your computer to the Arduino. In addition, if your program is using the Arduino serial interface, then you can display the data on your computer.

Our discussion of C programming was limited primarily to its application with the Arduino IDE (Integrated Development Environment). For this reason we did not discuss input/out of data (excepting the above), files or advanced C features. However, you will be able to program and allow your Arduino to operate independently of the computer, and you may power it with a separate wall-mount power supply or even a battery.

2. Arduino home page is shown below (https://www.arduino.cc):


ARDUINO
Genuino
Search the Arduino Website
Home
Buy
Download
Products
Learning
Forum
Support
Blog
LOG IN
SIGN UP
WHAT IS ARDUINO?
BLOG
STAGE BENCH IS AN ARDUINO-BASED LIVE CONTROLLER TABLE
BUY AN ARDUINO
LEARN ARDUINO
SUPPORT ARDUINO
ARDUINO CREATE
Write code, make IoT projects, and access cool tutorials!
WEB EDITOR UPDATES: IMPORT YOUR SKETCHBOOK AND MORE
AT HEART
Designed for makers and

3. Select the “Learn Arduino” menu item, and following display appears:



Install the Arduino Software (IDE)

Click on one of the four links below to get step-by-step instructions to set up the Arduino Software (IDE) on your computer, according to its operating system.

- Windows
- Mac OS X
- Linux
- Portable IDE (Windows and Linux)

After you have installed the Arduino Software (IDE), please choose your board in the list here on the right to learn how to get started with it.

Learn Arduino

- Read an introduction on what is Arduino and why you'd want to use it.
- What is the Arduino Software (IDE) and how do I change the default language?
- Libraries: Using and installing Arduino Libraries.
- Cores: Need to add a new board to your Arduino Software? Install the relate core and manage it.
- Troubleshooting: Advice on what to do if things don't work.

For a complete list of Guides visit the Foundations section, where you will find in-depth knowledge about the principles and techniques behind the Arduino platform.
Making the Arduino StarterKit projects and reading the book 'Getting Started with Arduino' are great ways to start learning and tinkering with coding and electronics.

What can I do with an Arduino or Genuino board?

If you are unsure on what your first project could be you can try the Arduino and Genuino Starter Kit, it will guide you from your very first steps with electronics into the world of interactive and sensing objects.
If you are looking for inspiration you can find a great variety of Tutorials on Arduino

Instructions for our boards:

- 101
- Due
- Gemma
- Intel Edison
- Intel Galileo Gen2
- Leonardo and Micro
- LilyPad
- LilyPad USB
- MEGA2560
- Mini
- MKR1000
- MKRZERO
- Nano
- Pro
- Pro Mini
- UNO
- Yún
- Zero
- ADK
- Esplora
- ISP
- TFT LCD Screen
- Arduino Robot

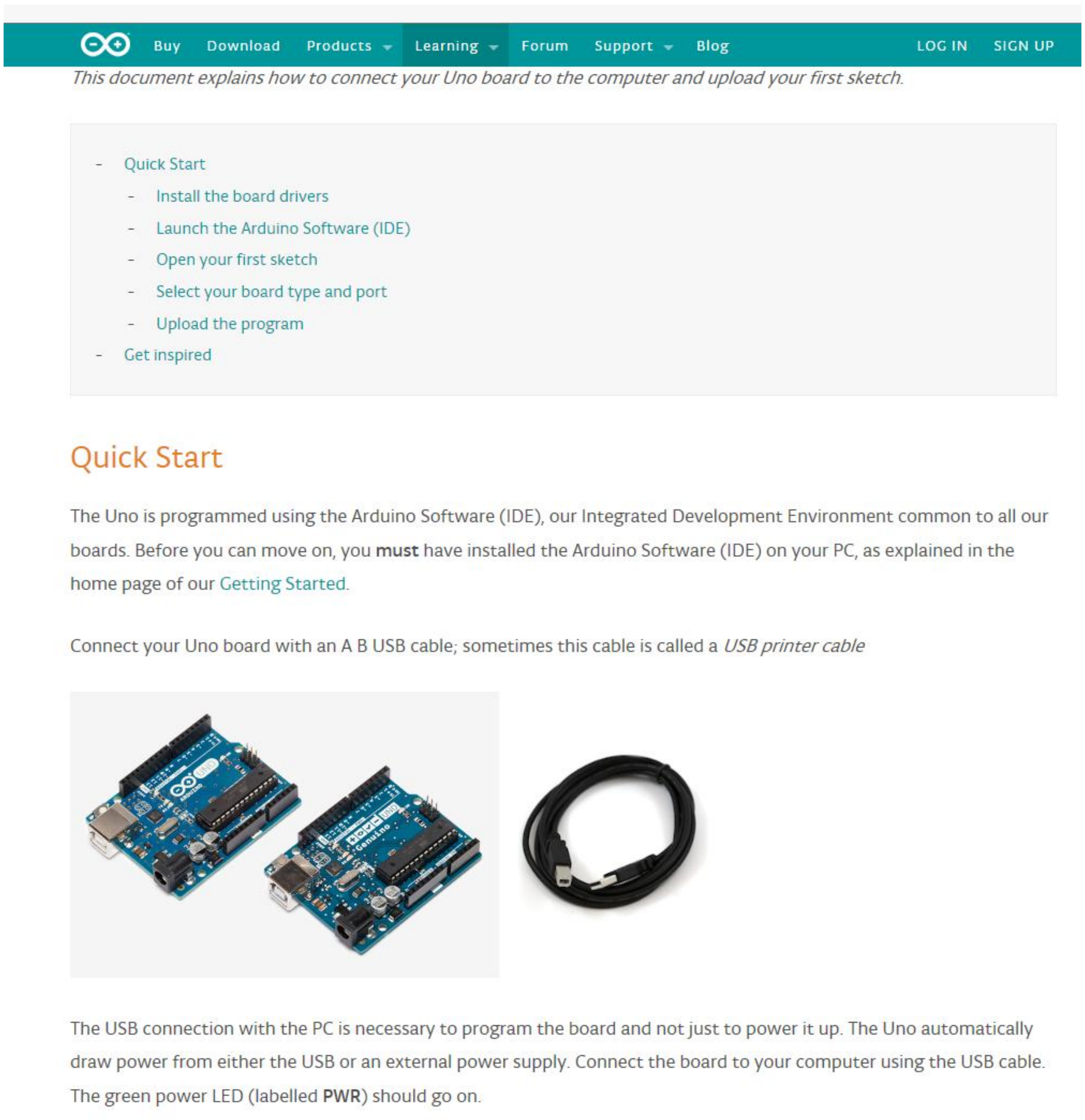
Instructions for shields:

- Ethernet Shield
- GSM Shield
- WiFi Shield 101
- WiFi Shield
- Wireless SD Shield

Instructions for retired boards:

- Arduino BT
- Arduino Fio

4. Follow the directions for your particular board and operating system. On the right hand side is additional instruction for the Uno board. Download the appropriate software. Follow the Quick Start instructions, including opening and running the LED blink “sketch” or program.



This document explains how to connect your Uno board to the computer and upload your first sketch.

Quick Start

The Uno is programmed using the Arduino Software (IDE), our Integrated Development Environment common to all our boards. Before you can move on, you **must** have installed the Arduino Software (IDE) on your PC, as explained in the home page of our Getting Started.

Connect your Uno board with an A B USB cable; sometimes this cable is called a *USB printer cable*

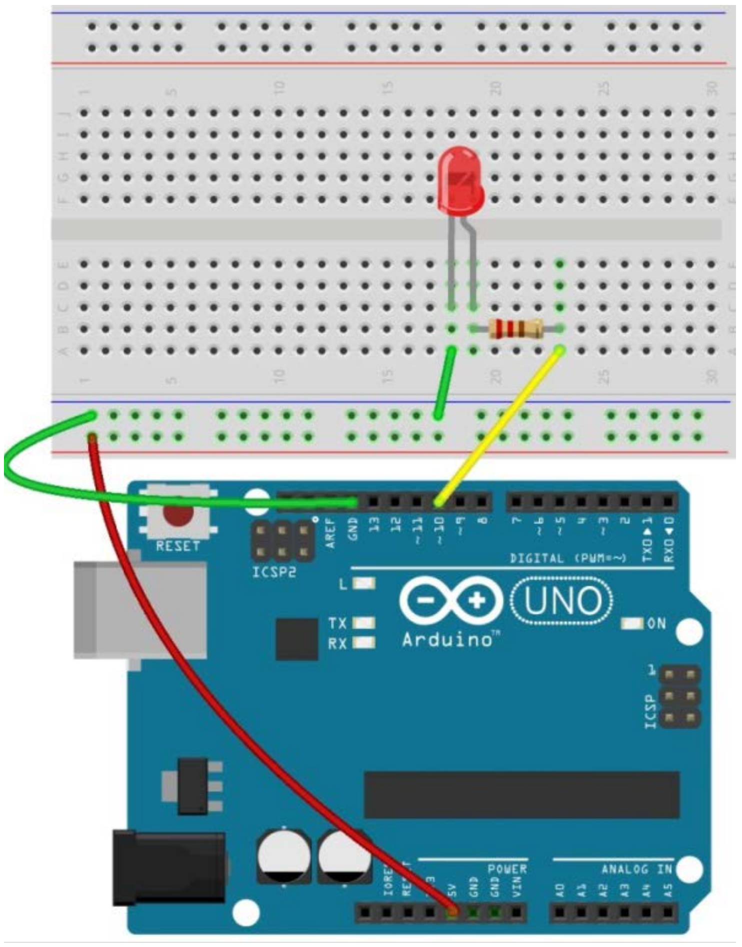
The USB connection with the PC is necessary to program the board and not just to power it up. The Uno automatically draw power from either the USB or an external power supply. Connect the board to your computer using the USB cable. The green power LED (labelled **PWR**) should go on.

Exercise 6.3: Wiring Your Own LED

Objective: Connecting an external LED to the Arduino, and then a push button to control it.

Breadboard

1. You should not have your computer connected to the Arduino when you are actively breadboarding.
2. Breadboard the circuit below with an LED and 220 ohm resistor as shown:



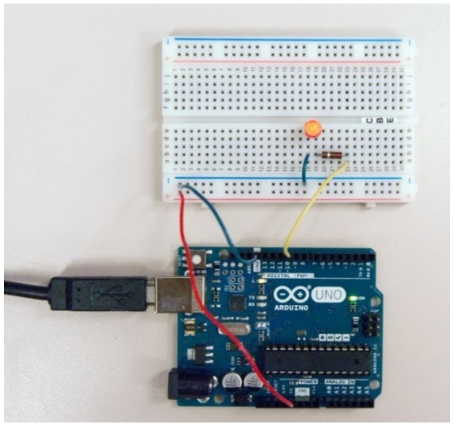
3. Prior to connecting to your computer, it helps to first connect the external power adaptor and be sure everything is ok. If ok (board LED lights) disconnect the external power adaptor, and then use the USB cable to connect to your computer. Never have both the external power supply and your computer USB cable connected at the same time.

Task

1. Download the Arduino software to your computer if you have not already done so. Copy, paste and load the following program into the Arduino (see website for code).
2. Connect your Arduino with a USB cable to your computer. Confirm powered.

```
const int LED = 10;
int blinks = 5;                    // blink 5 times;
bool done = false;
void setup()
{
  pinMode(LED, OUTPUT);            //set pin 10 as an OUTPUT
  digitalWrite(LED, LOW);          // Initialize off
}
void loop()
{
 while (done != true)
  {
   for (int i = 1; i<= blinks; i=i+1)
   {
     digitalWrite(LED, HIGH);      // Turn on LED
     delay(500);                   //Pause
     digitalWrite(LED, LOW);       // Turn off LED
     delay(500);                   //Pause
   }
   done = true;
  }
}
```

3. Run the program. How many times does the LED blink? _______________ (Yes or No)



4. Write a new program and wire the Arduino to take a pushbutton input. When the button is first pressed, the LED should turn on. When the push button is pressed again, the LED should turn off. The LED should continue to alternate states with each button push. You may need to wire a debounce circuit or *debounce in software*! You may wish to complete Exercise 5-5, and then come back to this.

Write your final tested code with comments here:

5. Take a photograph of your schematic and circuit setup.

Exercise 6.4: Reading a Potentiometer

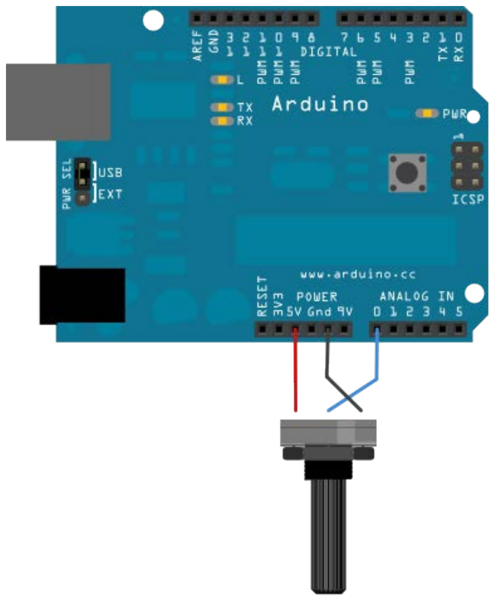
Objective: Reading analog data and turning on an LED based on a resistance threshold, then flashing the LED at a rate relative to the resistance of the potentiometer.

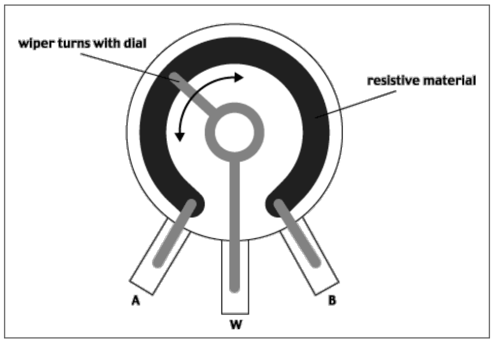
Discussion

This example demonstrates the use of the "if()" statement. It reads the state of a potentiometer (an analog input) and turns on an LED only if the potentiometer goes above a certain threshold level. It prints the analog value regardless of the level using the serial output.

Breadboard

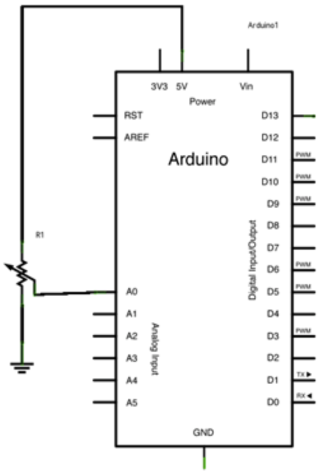
Wire the Arduino and potentiometer as shown:

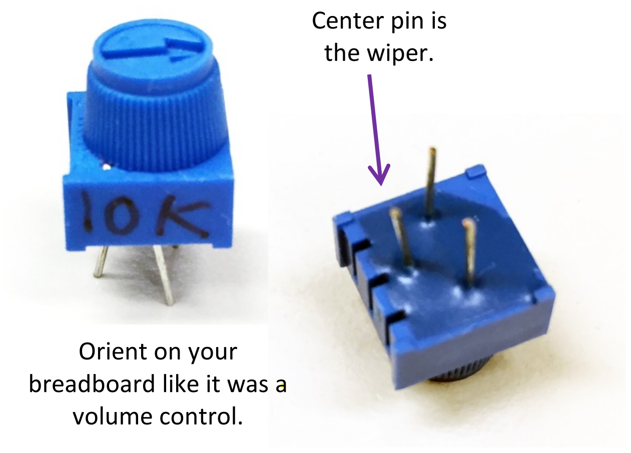




Feddersen, J. *Inside the Potentiometer*
http://fddrsn.net/pcomp/examples/potentiometers.html

The potentiometer is connected to analog pin 0. The center pin of the potentiometer goes to the analog pin. The side pins of the potentiometer go to +5V and ground. On most Arduino boards, there is already an LED on the board connected to pin 13, so you don't need any extra components for this example.





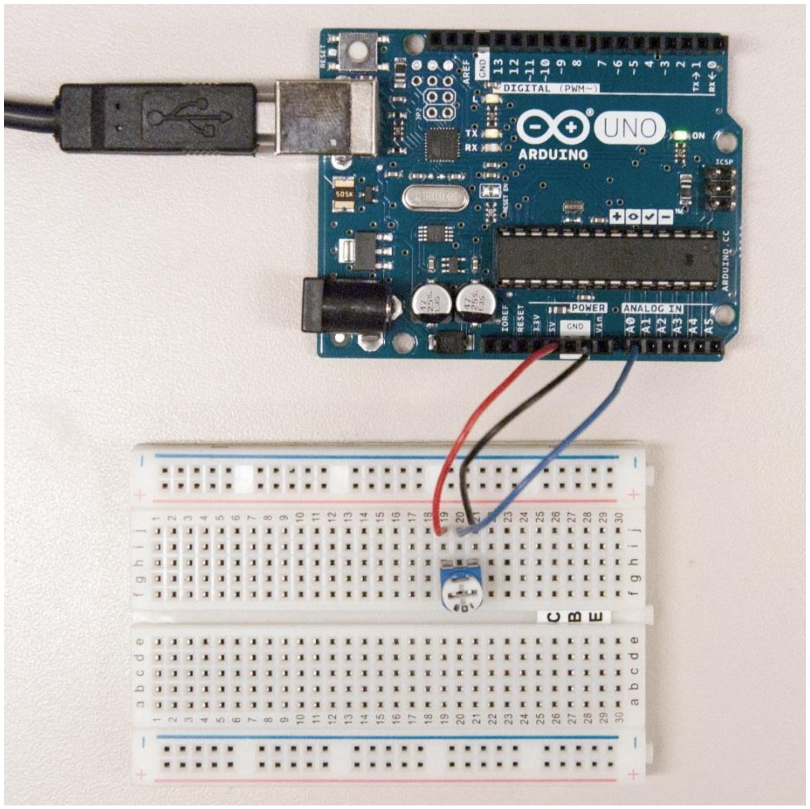
Task

1. Copy, paste and upload the following program from the course website:

```
const int analogPin = A0;          // pin that the potentiometer is attached to
const int ledPin = 13;             // pin that the LED is attached to on UNO
const int threshold = 400;        // an arbitrary threshold level that's in the range of
                                  //  the analog input

void setup()
{
  pinMode(ledPin, OUTPUT);        // initialize the LED pin as an output
  Serial.begin(9600);             // initialize serial communications
}

void loop()
{
 int analogValue = analogRead(analogPin);     // read the value of the potentiometer
 if (analogValue > threshold)     // if the analog value is high enough, turn on the LED

   {
     digitalWrite(ledPin, HIGH);
    }
else
   {
    digitalWrite(ledPin, LOW);
    }
  Serial.println(analogValue);       // print the analog value
  delay(1);                          // delay in between reads for stability

}
```

2. Run the program and display the Serial Monitor (under the "Tools" menu).

Does the LED respond to a threshold? _______________ (yes or no)
Do you have a serial output of the data? _______________ (yes or no)
At what value does the LED change state? ______________



3. Take a photograph of your setup.

4. Write a new program so that the LED flashes at a rate relative to the potentiometer position (resistance). Include a serial output of the potentiometer reading and flash rate. Write your final tested code with comments here:

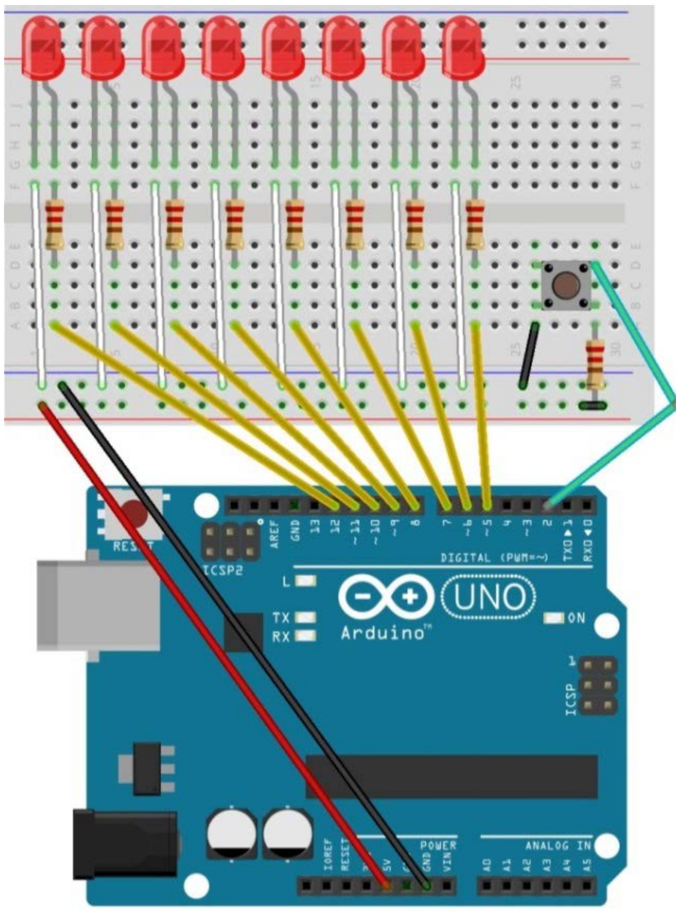
Exercise 6.5: Make an 8-Bit Binary Counter

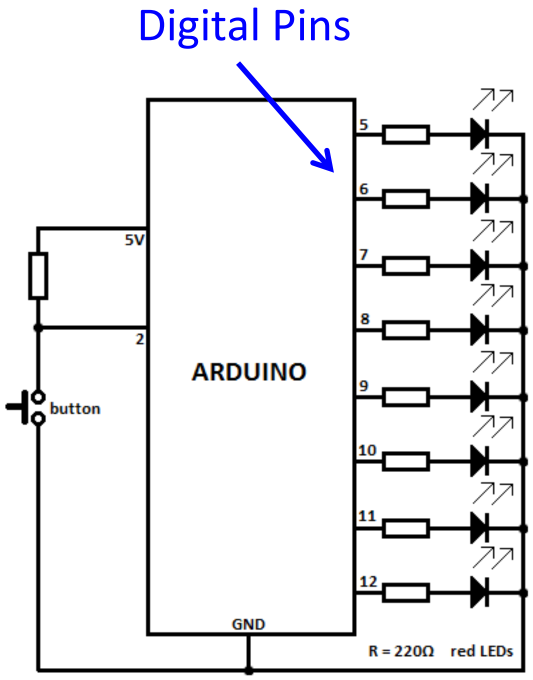
Objective: Use of a software button-debounce routine and interrupt function. This counter is the software equivalent of what you had previously done with 74HCT74 flip flops. For more information, go to the web link at the bottom of the code listing.

Breadboard

Wire the following circuit from the Fritzing diagram and schematic below:

128 64 32 16 8 4 2 1 Binary Place, on=1, off=0





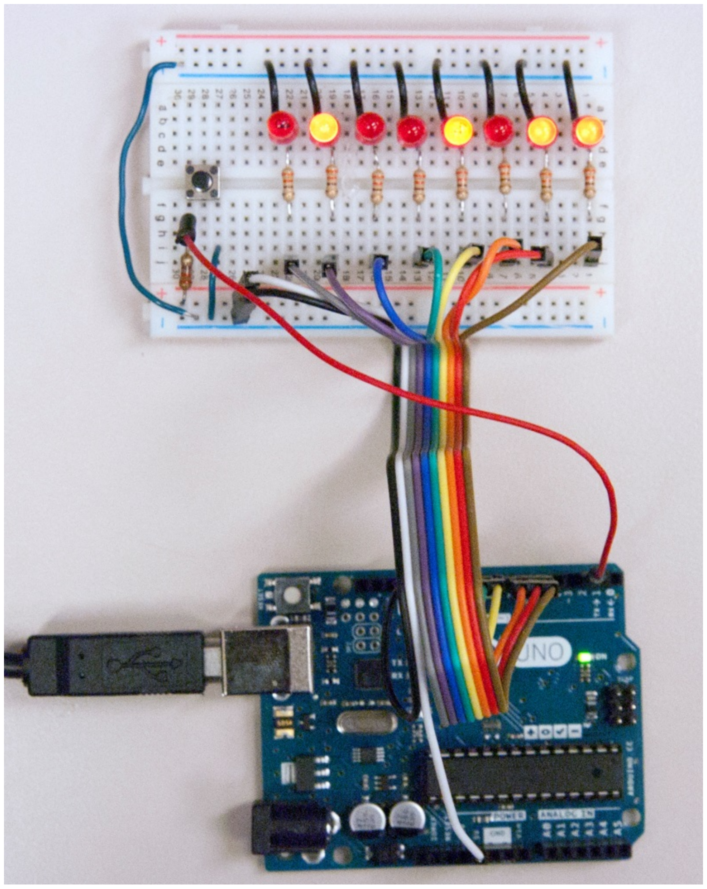
Task

1. Copy, paste and upload the following program from the course website:

```
int button = 2;                       // pin to connect the button
int presses = 0;                      // variable to store number of presses
long time = 0;                        // used for debounce
long debounce = 100;                  // how many ms to "debounce"
const byte numPins = 8;               // how many LEDs
int state;                            // used for HIGH or LOW
byte pins[] = {5, 6, 7, 8, 9, 10, 11, 12};        // LED Pins

void count()  // function count the button presses
{
if(millis() - time > debounce)  presses++; //debounce pushbutton
 time = millis();
}
void setup()
{
for(int i = 0; i < numPins; i++)         // set LED pins to outputs
 {
  pinMode(pins[i], OUTPUT);
 }
 pinMode(button, INPUT);
 attachInterrupt(0, count, LOW);   // pin 2 is interrupt 0 on UNO
}

void loop()
{
 /* convert presses to binary and store it as a string */
 String binNumber = String(presses, BIN);
int binLength = binNumber.length();  //get length of string
  if(presses <= 255)  // if we have less or equal to 255 presses
  {
   for(int i = 0, x = 1; i < binLength; i++, x+=2)
    {
      if(binNumber[i] == '0') state = LOW;
      if(binNumber[i] == '1') state = HIGH;
      digitalWrite(pins[i] + binLength - x, state);
     }
   // do something when we reach 255
  }
}
 // http://www.electroschematics.com/9809/arduino-8-bit-binary-LED
```

2. What is the maximum number that can be displayed in binary? _______________
   What is the maximum number that can be displayed in decimal? _______________

3. Take a photograph of your setup.



Use the Fritzing diagram and schematic on page 12 when you do your wiring, and not the photograph above (it is too hard to see the connections).

End of Arduino Uno Lab Exercises