1 Objectives

By the end of this lab, you will:

- Connect your Arduino to your computer
- Connect and control an LED

Materials: We will provide the parts necessary for this lab, except for the Arduino.

2 Pre-lab: Connecting your Arduino to your computer

1. Do not connect your Arduino to your computer yet. First, download and install the Arduino software and IDE for your OS from https://arduino.cc/en/Main/Software
   - **Linux**: If your distribution has a relatively recent version of Arduino, you can install it directly from your package manager. Otherwise:
     - Extract the compressed tar archive into a suitable directory.
     - Enter the directory into which you extracted the archive and run: `sudo ./install.sh`
   - **OS X**: Open the .dmg file that you downloaded and drag the Arduino application into your Applications folder.
   - **Windows** (using the .exe installer): Follow the installation prompts, making sure to leave “Install USB driver” checked. For more detailed installation instructions, see https://arduino.cc/en/Guide/Windows
   - Arduino software and IDE is also available on Brown CS department machines.

2. Launch the Arduino application, which loads the Arduino IDE.

3. Connect your Arduino using its USB cable.

4. Configure the IDE for your board type by selecting the appropriate part from the **Tools > Board** menu.

5. Verify that your board is connected properly by checking the **Tools > Serial Port** menu:
• On OS X and Linux, you should see a port with the name /dev/ttyUSB0, /dev/tty.usbmodem, /dev/ttyACM0 or similar

• On Windows, you should see a port named COMx, where x is a number.

6. Test the board by loading and running your first program. The Arduino IDE provides a series of example programs for demonstration and testing. Load the simplest of these programs, which just blinks an LED, by navigating to: File > Examples > 1.Basics > Blink.

7. You should see the source code for the program, called a “sketch”, in the main window. Take a moment to look over this code before proceeding, though you need not understand it fully yet.

8. Click the Upload button in the toolbar. In the lower window, you can see the IDE’s output as it runs the compiler and loads the compiled program onto the board. After a few seconds, the IDE should say that it is “Done uploading” and you should see an LED blinking on your board!

If you receive an error, make sure your board is connected and try again. If you continue to encounter issues, ask the course staff for help.

3 In Lab: Blinking an External LED

Next we will connect an external LED to the board and make it blink. To do this, you will need:

• A resistor between 200Ω and 1.5kΩ

• An LED (single-color or RGB)

Connect your circuit to match the schematic shown in Figure 1. Using your Arduino and breadboard, it should look similar to Figure 2.

![Figure 1: Basic LED circuit.](image)

Note that LEDs are polar components: the shorter wire usually corresponds to the cathode, which should be connected to ground (0V). LED has very small resistance when power on. Therefore, we need to add a resistor in the circuit to prevent short circuit.

If you have an RGB LED, your LED will have four wires instead of two: one wire one for each color and one wire that serves as a common cathode or anode (which is common to all colors). In an RGB LED, the longest wire is usually the common pin. If your LED has a common cathode (which is more common, so try this first), connect the cathode to ground and then connect any one of the
colors to the Arduino as in Figure 1. If this does not work, try connecting the common pin to 5V instead.

Using the blink sketch as an example, create your own sketch to blink the external LED. Make sure you understand why the sketch is making the LED blink—feel free to ask the course staff for any clarifying questions.

**Task:** Modify your sketch to blink the LED in any non-symmetric on/off pattern. For example, configure the timing such that the LED is off for one second and then on for two seconds.

### 4 In Lab: Connecting a push-button switch

Next, we will connect an input to the Arduino using a push-button switch. To do this, you will need:

- A push-button switch
- Another resistor between 1kΩ and 10kΩ

Connect the switch to the Arduino according as shown in Figures 3 and 4.

The wiring of the switch is not obvious based on its four pins. For this design, the two pins on either side of the switch (the left and right sides in Figure 4) are connected internally. Pushing the button closes the circuit and bridges the left and right sides.

This circuit implements a *pull-up resistor*. When the button is unpressed (open), the input pin is connected to 5V via the resistor—thus, the input pin would read as HIGH. When the button is
pressed (closed), the input pin is connected to ground and thus reads LOW. Adding the resistor and connection to 5V gives the pin a “default state” when the switch is open. If we did not include these components, the input would be floating, meaning that its state cannot be determined since the slightest bit of electrical noise could cause it to be read as HIGH or LOW.

In order to read the button in your program, add the following to your sketch:

- In the setup() function, configure the pin as an input using pinMode():
  

- In the main loop, read the state of the button using digitalWrite() and write to serial port with message ”Switch on” using Serial.println() when the switch is on:
  

**Task:** Combine your circuit from part 3 and part 4 and write a simple Arduino program to turn on LED light when the switch on is detected on digital pin.
## 5 Grading Rubric

<table>
<thead>
<tr>
<th>Task</th>
<th>Total Points</th>
<th>Points Earned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic LED circuit</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Switch controlling LED</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>20</strong></td>
<td></td>
</tr>
</tbody>
</table>