Lab Assignment 1 CSE474 Spring 2023

Prof. Vikram Iyer[1] University of Washington

Getting Started with the Arduino Mega

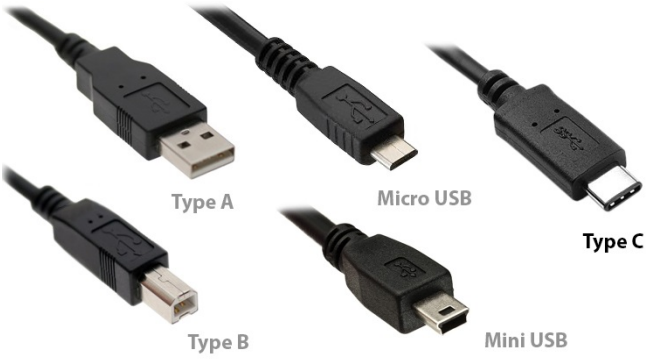
Learning Objectives

With successful completion of this lab, the student will be able to

- Install and set up Arduino IDE
- Build and run a basic sketch (program) using the Arduino Libraries
- Modify and demonstrate blinking light code and speaker output tone.
- Learn to use an oscilloscope for debugging

Equipment

1) Arduino Mega Microcontroller board
2) External Arduino power supply
3) LEDs and 250 Ohm resistors
4) Small 8-Ohm Speaker
5) USB- Type-B cable (with USB-A or USB-C for computer end) (you probably got one with Arduino)
   (image: globetek.com)
6) Solderless Breadboard
7) Wires (either with pins (see the kits), or cut and strip your own 22AWG solid wire).



Helpful Resources:

If you haven't worked with electronics before, or want to brush up, check out this list of compiled resources that will help you learn the basics. [LINK]. For a more in depth introduction as well as additional resources on getting started with Arduinos, check out this excellent series of tutorials by Jon Froehlich: makeabilitylab.github.io/physcomp/electronics/
Feel free to ask TAs any questions in office hours/Ed Discussion as well!

Technical Requirements

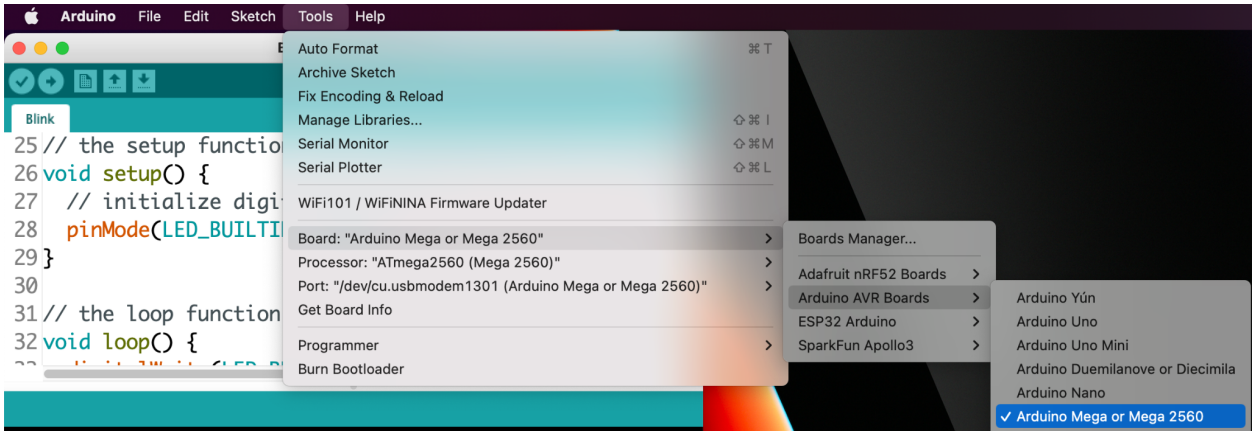
In this lab we will use some standard Arduino code packages to blink an LED and to make a tone on a speaker.

Thanks for valuable guidance from Prof. Blake Hannaford

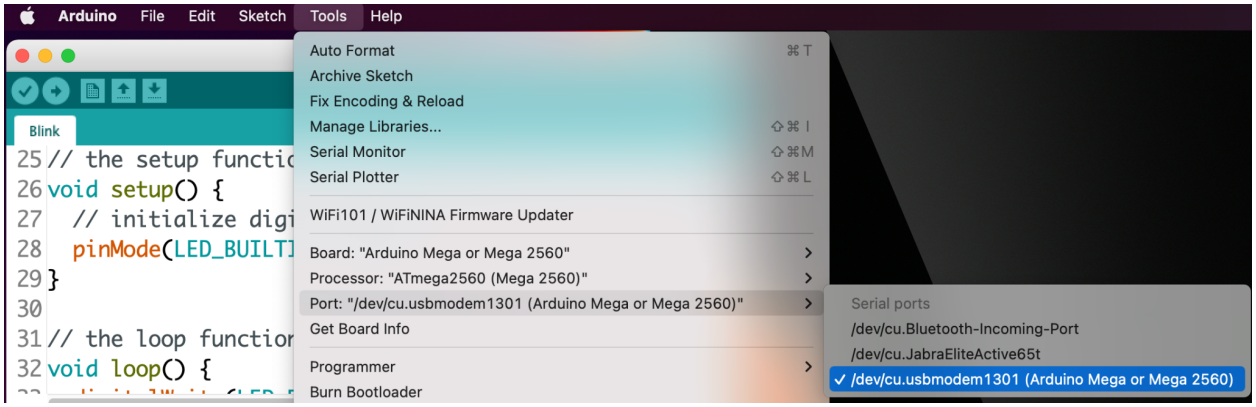
Required Procedures

(suggestions for breaking down the lab, step-by-step testing, etc.)

1. **Part I Intro to Arduino:**
   Install the open-source Arduino Interactive Development Environment (IDE) from HERE on your Windows, Mac OS, or Linux computer. In case of difficulty, try Arduino Troubleshooting Guide, or Arduino Help Center. Then if those don't solve the problem (along with searching elsewhere online) please reach out to the ECE474 TA's.
   1.1. Open the Arduino IDE by clicking its desktop icon (OS dependent).
   1.2. Open the blink example by "File->Examples->0.1Basics->Blink".
   1.3. "Break" your code by deleting the required semicolon (";") on line 28 (line number shows in the lower left). Hit the checkmark (upper left) and observe the orange error messages.
   1.4. Fix your bug and build the code again (checkmark)
   1.5. Connect your Arduino MEGA to your PC via USB (observe steady green "ON" light indicating power)
   1.6. Configure IDE for your board and connection. Remember, because C is a compiled language and different chips have varying memory and peripherals we have to select the right target platform. To do this:
      1.6.1. "Tools->Board->Arduino Mega 2560 or Mega 2650"

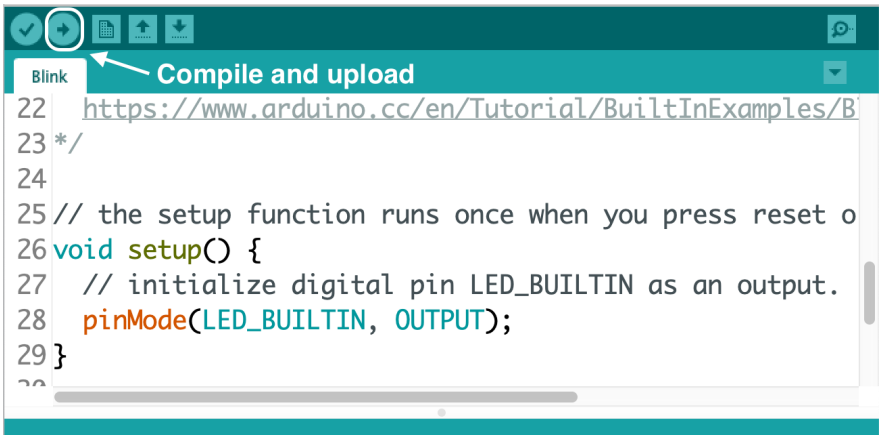


      1.6.2. "Tools->Serial Port->" Select the one that says: "(Arduino Mega or Mega 2650)" The exact port name may be different (e.g. COM3 on Windows)



Thanks for valuable guidance from Prof. Blake Hannaford

1.7. Upload “blink” to the Arduino (right arrow icon at the top). You should see a pair of LEDs on the board rapidly flashing. This indicates code downloading into your Arduino. After that your code automatically starts.



1.8. The LED labeled “L” near the USB connector should blink 1 time every two seconds.

2. **Part II: Modifying your sketch and your LED**

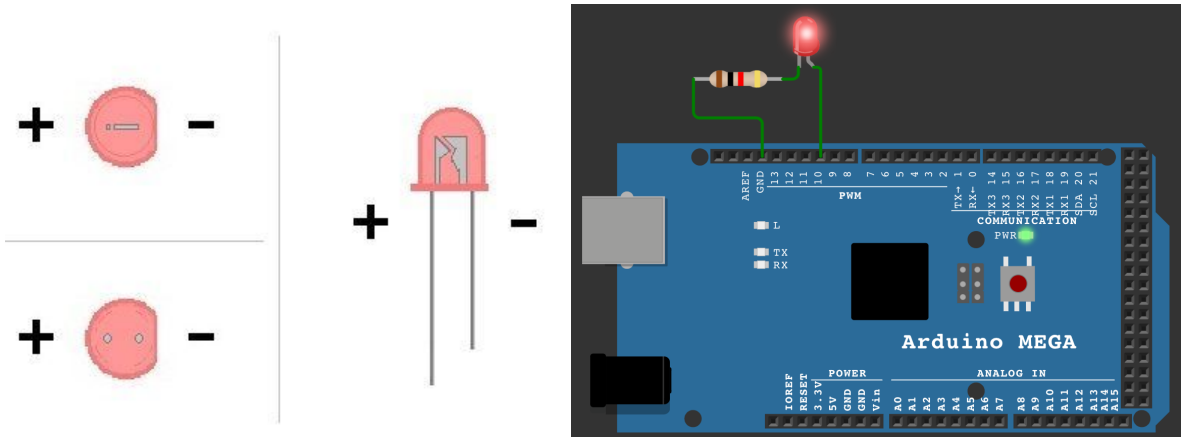
2.1. Change the delay values from 1000ms to 200ms. Recompile (checkmark) and re-upload your sketch.

2.2. The LED should go faster.

2.3. Now unplug the USB cable from your Arduino and plug in the 120VAC power adapter. The Arduino has non-volatile memory so whenever it’s plugged into power it will automatically load the last code uploaded, so the flashing light program should run again. If not, try pressing the reset button.

3. Part III External LED Hardware

3.1. Connect a 250 Ω resistor between the cathode (short wire) of an LED and a GND/0V pin on the Arduino. (See picture below for reference) Make the resistor-LED connection by either breadboard connecting, soldering, wire twisting, or alligator clip as you prefer.Connect the long LED wire with IO/PWM pin 10.



3.2. Modify the `blink.ino` code on line 10 so that the LED pin is now pin 10

3.3. Compile and upload. Verify your LED attached to pin 10 is blinking as expected.

Thanks for valuable guidance from Prof. Blake Hannaford

4. **Part IV Multiple tasks**
   - 4.1. Keeping your external LED wired up, wire the 8 Ohm speaker between pin 2 and +3.3V.
   - 4.2. Modify the blink “sketch” (code) to make clicks on the speaker.
     HINT: What happens if you do the same `digitalWrite` on pin 2?
   - 4.3. Now add new code to blink both the on-board LED (pin 13), and the external LED (pin 10) such that one is ON when the other is OFF and the speaker clicks when the LEDs change state.

5. **Part V Differing Periodicities**
   - 5.1. Modify your code so that the LEDs flash the same as 4.3, but instead of clicking, the speaker emits a continuous tone of 250Hz. **Pro tip (and lab requirement):** write your code so the sound turns off after a few seconds (it will drive you crazy!). Make sure your LEDs keep flashing in the correct pattern after audio stops (your code didn’t just crash).

     NOTE: Do not use the `tone()` function for this. This can cause your program to freeze up and not allow you to blink the LED.

6. **Part VI Measuring signals with an oscilloscope**
   - 6.1. Read the following background information to learn how an oscilloscope works

What is an oscilloscope?

An oscilloscope (scope for short) is an instrument that displays a 2D graph of voltage versus time on the screen. A scope lets you see the actual electrical signal coming from or going into your microcontroller and can be very useful for debugging things like timing errors, etc.

Video tutorial on the basic functions of an oscilloscope:

How to Use an Oscilloscope

Note: the exact placement of knobs on different scope models may vary, but they’re generally the same. The first ~8 min of this video does a nice job explaining these controls.

Thanks for valuable guidance from Prof. Blake Hannaford

For the purpose of this lab, these 3 systems/modules are most important:

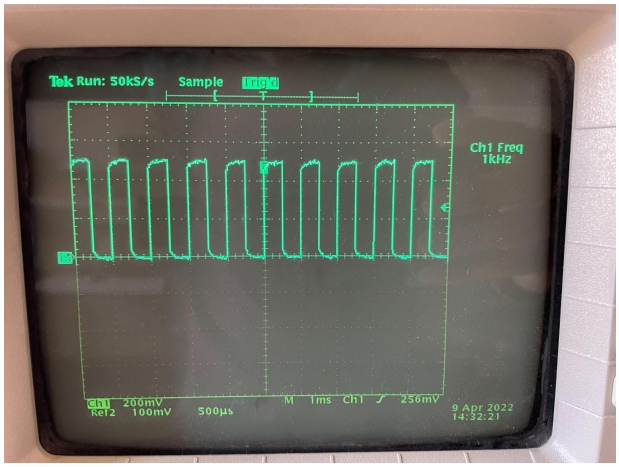
Vertical: Controls the vertical (voltage) measurement of the signal. Needs to be tuned to display the correct voltage/division (division = the vertical length of the ‘rectangles’ on the screen)

Horizontal: Controls the time/division of the screen. Horizontal scale needs to be adjusted so we can see the desired horizontal/time interval of the signal.

Trigger: For repeating waves/signals, the trigger ‘goes off’ at certain points of the signal so the oscilloscope can stabilize it on the display. Trigger levels are also used to capture pulses/single events.

For further reading if you are interested, Tektronix has a very detailed explanation of how to operate oscilloscopes in this document. It explains in excellent detail on how to tune the oscilloscope to capture the desired information. (Chapter 4, linked above is relevant to this lab)

6.2. Try measuring a waveform. Connect the oscilloscope probe to your Arduino output and ground. Try playing around with the knobs to make sure you can get a stable image of the waveform on the screen. You can also use the “Measure” button to add things like a measurement of the frequency, etc.



6.3. Use the in-lab oscilloscope to show that the waveform from your Arduino is steady and demonstrate that it’s frequency is indeed 250 Hz. Explain the controls of the oscilloscope and what you need to adjust to measure the signal..

Thanks for valuable guidance from Prof. Blake Hannaford

Turn In Requirements

Learning Objectives

With successful completion of this lab, the student will be able to

- Install and set up Arduino IDE
- Build and run a basic sketch (program) using the Arduino Libraries
- Modify and demonstrate blinking light code and speaker output tone.

Report Turn In:

Turn in a PDF report following the document template and the Lab Overview/Guidance CSE474 documents.

Demo Requirements: (please see Lab Overview/Guidance Sp23)

Demo will be in-person. All team members must be present for the demo. The following features must be demonstrated live to an instructor or TA:

1. Blink on-board LED @ 200ms
2. Blink off-board LED @ 200ms (note that one compile can demo both of these)
3. Blink on-board and off-board LEDs with speaker clicks.
4. Make a tone @ 250Hz with LEDs flashing @ 200ms without audible glitches.
5. Use the in-lab oscilloscope to show that the waveform is steady and demonstrate that it's frequency is indeed 250 Hz.

In the Demo, verbally answer the following questions:

6. What happened when you deleted the semicolon in the Blink.ino file?
7. How did you change the flashing LED from on-board to off-board?
8. How did you make the speaker emit a tone of 250Hz?
9. How did you make the speaker emit a 250 Hz tone and the LEDs flash at the right rates?
10. Answer other questions at discretion of the instructor/TA.

Code Turn In:
Turn in ALL .ino, .h and .c files in the project. Submit a zipped folder containing all code. Comment out (but do not delete) code which might be required for intermediate steps but is not required for the final code. This will help us give you partial credit!
Turned in code should include functionality for the following parts above:

- Part 1.8
- Part 2.2
- Part 3.3
- Part 4.2
- Part 4.3
- Part 5.1

Thanks for valuable guidance from Prof. Blake Hannaford