EECS 1011 – Lab E: Arduino Intro

Dr. James Andrew Smith, PEng

Summary: Learn to use Arduino-compatible hardware with MATLAB. Make the board flash a light and buzz out some sounds.

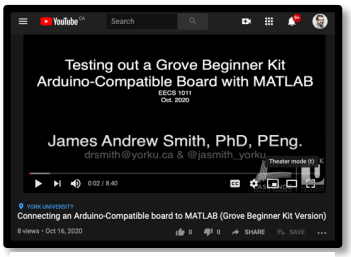


Figure 1 Watch the "setup" video for your Grove board: https://youtu.be/ni72AXdbHoY

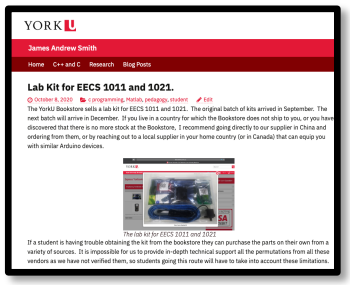


Figure 2 Making your own kit? https://bit.ly/3k6Du6m

Pre-Lab:

- **Buy the Grove board kit** (Arduino-compatible) from the YorkU Bookstore.
  - Alternatively: **create your own** lab kit using a Grove Beginner Kit or other Arduino UNO compatible kit using suggestions on the blog:
    - https://drsmith.blog.yorku.ca/2020/10/lab-kit-for-eecs-1011-and-1021/
  -
- Install MATLAB 2020a (or 2020b) on your home computer
  - Follow the instructions from the MATLAB + Arduino setup videos on eClass.

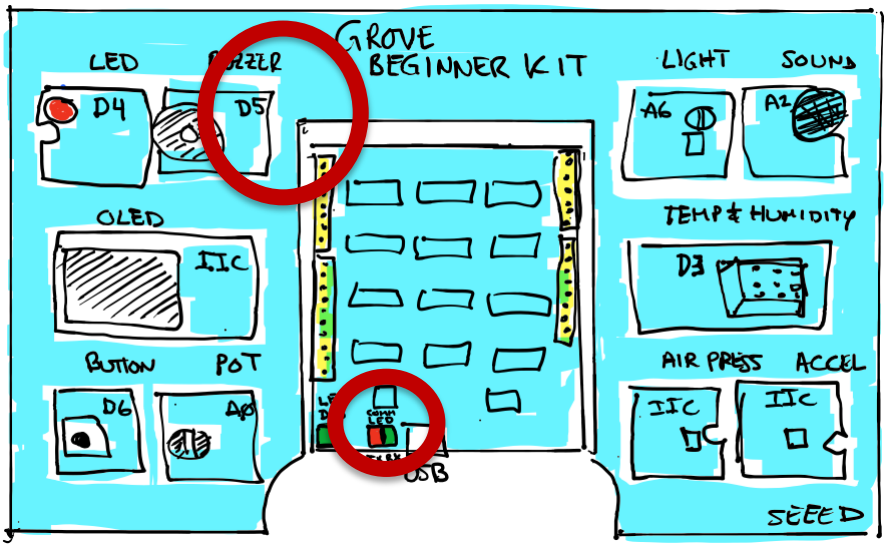


Figure 3 You'll be using the Arduino-compatible Seeed Studio Grove Beginner Kit. You'll be using the Buzzer connected to D5 and the LED connected to D13, as shown above.

Intro

The lab is divided up into two parts:

- **Part 1**: Flash an LED ('D13')
- **Part 2**: Try out the buzzer ('D5')
- ~~**Part 3:** Make your buzzer speak Morse Code.~~ (update: Part 3 removed)

Due date for Lab Report. There is **no report**. Perform the lab demonstrations, as required. Refer to the eClass section for this particular lab and conduct all (if any) interactive activities under the Lab E heading. All lab-related activities are due by the Sunday after your synchronous lab session.

Marking Guide: each component of the lab is equally weighted. 0.5 for Part 1 and 0.5 for Part 2. Don't have Grove/Arduino hardware? Demo the instructions for half points. (updated for students without Arduino Hardware)

Lab Instructions

This lab is designed to make sure that you are familiar with some elements of your Arduino-compatible board.

You will need to plug in the Grove kit and determine how your computer connects to it via the USB subsystem within the computer. As discussed in the setup video for Matlab + Arduino, each computer can have a different way of identifying the board. The most straight-forward way to figure that out is to turn on the Arduino IDE and to use Tools -> Port method to figure that out.

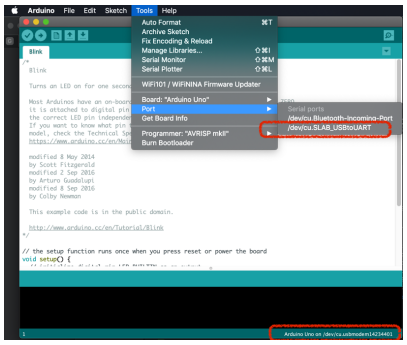


Figure 4 The Arduino IDE can be very helpful in finding out how your board is connected to your computer. Here, the board is connected to /dev/cu.SLAB_USBtoUART. Your board will likely have a different name.

If you're having trouble determining which of the strange-looking names it is, unplug your board, then look at the entries in the Tools -> Port menu. Then, plug in the board and look again at Tools -> Port. The new entry is your key to Arduino happiness.

Got back to MATLAB and connect to the board. Do this by typing in something like

```
a = arduino('/dev/tty.SLAB_USBtoUART', 'uno')
```

where `/dev/tty.SLAB_USBtoUART` will likely be different for you.

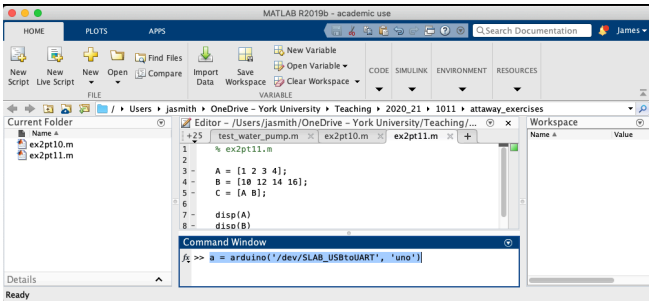


Figure 5 Connect to your board in MATLAB. See video https://youtu.be/ni72AXdbHoY for more details.

See video for details: https://youtu.be/ni72AXdbHoY

Part 1: Turn on a LED

Feel free to refer to the Matlab help page on writing values to the digital pins (https://tinyurl.com/y2b43uzc or https://bit.ly/2ES6sXS)

There are two LEDs on the board that you can access:

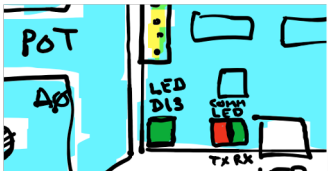
- D13: this is the traditional Arduino "blinky light" LED. If you've used an Arduino UNO before, you're probably used it before. It's found right
- D4: this is the big red LED found in the upper-right corner of the board, as shown in Fig. 1.

After having initialized the Arduino object ("a = arduin…" as shown above), from the command window in MATLAB type in

```
>> writeDigitalPin(a, 'D13', 1);          % write a "1" to pin D13 on Arduino object 'a'
```

Note that there are two other immediately to the left of the USB port. These are the communication LEDs and show when your PC is talking to the board.

They'll flash all the time. Instead, focus on flashing LED D13, as shown below.



Look at your board. Did the LED beside the USB port light up? Now turn off the LED:

```
>> writeDigitalPin(a, 'D13', 0); % write a "0" to pin D13 on Arduino object 'a'
```

Okay. Now add a pause after it turn on or turns off. Use the pause function. For instance, if I want to pause the current task for four and a half seconds I would do:

```
>> pause(4.5); % pause for 4.5 seconds.
```

Write a script in the script editor, combing the commands listed above to do the following:

1. Turn on and wait half a second.
2. Turn off and wait half a second.
3. Turn on, wait a full second, and then turn off. Stay off.

When it's ready, highlight the script with your mouse and execute using a right-click on your mouse.

Demo 1: Demo this to the TA for 0.5 points by sharing your webcam during the lab session. If you do not have the Arduino hardware, screen share the commands that you would have used. You'll get 0.25 points if they are basically correct. (updated)

Part 2: Use the Buzzer on your Grove Board

There is a passive buzzer on your board. It's located in the upper left and is connected to the D5 pin of your microcontroller.

Make the buzzer turn on for half a second and then turn off.

```
>> writePWMDutyCycle(a,'D5',0.33); pause(0.2); writeDigitalPin(a,'D5',0)
```

What this function does is send pulses to the buzzer, exciting it. The "0.33" means that the pulse is 33% of the time "on" and 67% of the time "off". Changing the percentage from 0 to 50% should modify the pitch of the buzzing.

Demo 2: Demo this to the TA for 0.5 points by sharing your webcam during the lab session. Show that you can make the buzzer make a buzzing sound to the TA. If you do not have the Arduino hardware, screen share the commands that you would have used. You'll get 0.25 points if they are basically correct. (updated)

Part 3: Communicate in Morse code with your Buzzer

Here is an example of how to make the buzzer do 'SOS' in Morse code.

It's not very efficient or elegant, but it demonstrates how to make sound on the buzzer and to separate out long and short sounds, as well as characters ('o' versus 's').

Using the character set listed on Wikipedia, (https://bit.ly/3ba52Eq or https://tinyurl.com/cj2r4r5), express the first letter of your first (given) name in Morse code. Use the same timings (0.2 and 0.4 seconds) as used in the example.

```
% SOS in Morse on buzzer (version 1)

% ---------- 's' ------------
% 'short'
writePWMDutyCycle(a,'D5',0.33); pause(0.2); writeDigitalPin(a,'D5',0);
pause(0.2);

% 'short'
writePWMDutyCycle(a,'D5',0.33); pause(0.2); writeDigitalPin(a,'D5',0);
pause(0.2);

% 'short'
writePWMDutyCycle(a,'D5',0.33); pause(0.2); writeDigitalPin(a,'D5',0);
pause(0.2);

% -------- 'long pause' ---------
% add another pause between letters
pause(0.4);

% ---------- 'o' ------------
% 'long'
writePWMDutyCycle(a,'D5',0.33); pause(0.4); writeDigitalPin(a,'D5',0);
pause(0.2);

% 'long'
writePWMDutyCycle(a,'D5',0.33); pause(0.4); writeDigitalPin(a,'D5',0);
pause(0.2);

% 'long'
writePWMDutyCycle(a,'D5',0.33); pause(0.4); writeDigitalPin(a,'D5',0);
pause(0.2);

% -------- 'long pause' ---------
% add another pause between letters
pause(0.4);
```

Demo 1

Log in to the video conference session during your scheduled lab time and **demonstrate to the TA** that you can make an audible signal come out of your buzzer and that you can make it spell out the first letter in your name in Morse Code.

Demo 2

The TA will then ask you to repeat the above, but for a different letter of the TA's choosing. You will be expected to modify your source code in less than three minutes to **demonstrate this** to the TA.

Both demonstrations are to take place in less than five minutes (total).

To achieve full marks in today's lab you need to demonstrate both tasks. If you demonstrate only one (fully or partially) then you get half the marks.

Part 3 is no longer going to be demonstrated to the TAs. Focus on Parts 1 and 2 instead.