Getting Started with Arduino:

> "You will struggle when you first learn to write code in Arduino, those who succeed are those who persevere through the frustration and learn to use the plentiful resources available to them."
>
> -the person who wrote this

Good terms to know:

Arduino:

-The company that created the hardware and software that you will use to create the control system for your MAE106 robot.

Arduino Boards:

-The microcontroller that will serve as the control hub for your final project. They are capable of being connected to various inputs (sensors, power sources, etc.) as well as providing outputs to control various parts. Arduino boards are controlled by user-uploaded code, created in the IDE/

IDE:

-An IDE, or "Integrated Development Environment", is the platform on which you will construct the code that will be uploaded and run by your Arduino board.

Functions:

-More than a party, functions are modules of code that a programmer (you!) create to accomplish a task. These functions can be used repeatedly and can be utilized by other functions in the same code.

Pseudocode:

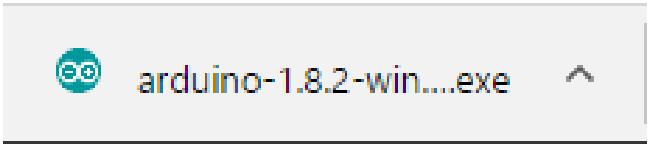
-An incredibly useful practice/tool in your coding arsenal, pseudocode is an informal description of the function (or actions) that you want to transform into code. Pseudocode is great for logically organizing your thoughts before or during code-writing.

Debugging:

-Debugging is the process of fixing problems and/or run-errors in your code. Often demonstrated when you compile your code, debugging can involve simple issues like adding that semi-colon or bracket that you forgot or can require addressing major logic errors. Most of your learning (and frustrations) will be during this vital process.

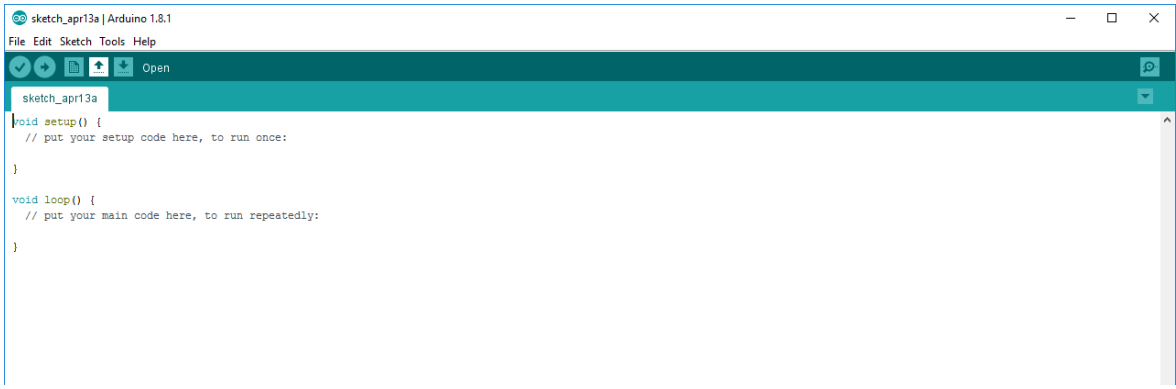
Downloading your IDE (NO NEED TO DO THIS IN THE LAB COMPUTERS):

Step 1: Go to the Arduino website [www.arduino.cc/en/main/software] and select the operating system that you are running and download the corresponding software. **NOTE: YOU ARE NOT REQUIRED TO CONTRIBUTE, THERE IS A "JUST DOWNLOAD" OPTION**



Step 2: Open the executable file after it has completed downloading and follow the instructions in the set-up wizard.

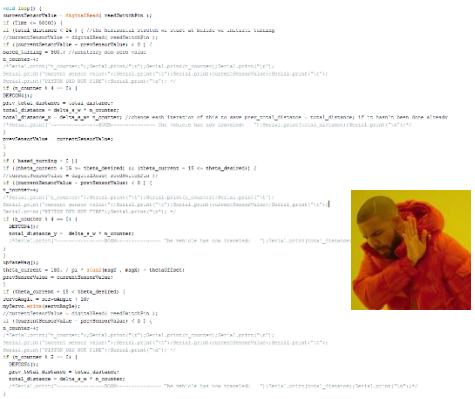
Step 3: After you are done setting up through the wizard, your IDE should open up and you are ready to code!



Things to keep in mind…

1. "ctrl" + "z" is your friend! These two buttons will allow you to undo your last action each time you press them in series. Great for when you make a small mistake and need to take a step back.

2. ALWAYS save your work, no matter how insignificant you think it is. You never know what you might want to look back to.
    a. Making multiple save files for different trains of thoughts can be very helpful! It allows you to look back and compare code or see how your train of thought has changed
3. You're never "in too deep". Don't be afraid to take a few steps back to fix or simplify previous code. Building code on a poor foundation will only make your life harder in the long run!
4. Build good practices! There are many nuances in writing code, each person develops their unique flair, but there are many common practices that are conducive to good code formulating that will improve your understanding as well as help with debugging.
    a. Camel case – manthiswasprobablyprettyhardtoread ButReadingThisIsALotEasier. When you have to use variables and can't use your spacebar to denote separate words, camelCase is your friend, save your eyes the trouble.
    b. You **almost** always have semi-colons after each statement. Double-check those (and similar nuances)
    c. Indent after each new "loop". Sure you might not know what a loop is, but look at the pictures below and tell me what you like better. Save your eyes, and your head, the pain.



5. Don't be afraid to ask for help. Utilize all your resources; the internet is your friend, but so are your TA's! They are all here to help you achieve the most you can.

Resources:

The following are just some **(BUT NOT ALL)** resources that you can turn to while you're laboring on your control system.

1. An online booklet of MANY functions and capabilities of the Arduino boards and software
2. Some helpful techniques to help your debugging process
3. Video that goes over many things about your Arduino Board as well installing your IDE. Start at 15:13
4. Video that strips your IDE down to the basics! Good place to start if you've never worked with Arduino before
5. Confused about why your function isn't working or looking for inspiration? Forums are great!
6. Last Resort
7. Talk to your peers!
8. Contact your TAs

Arduino:

1. Pull-up Resistor + LED
    a. Connect an LED circuit to the Arduino and write a sketch to output a digital signal that will power the LED. (attach link of LED-resistor diagram)
    b. Create a circuit connecting the button to the Arduino and establish it as an input_pullup pin.
    c. Use the Serial Monitor to visualize how the pull-up resistor works with the button.
    d. Use the values from the button to control the LED, create a sketch that turns the LED off when the button is pressed and turns the LED on when the button is not pressed.
2. Add switch cases
    a. Use a "switch case" that has three states with the following functions in each state. The states should be changed by pressing the button. After the third state, pressing the button should return the state to the first state.
        i. Turn LED on
        ii. Blink LED without using "delay()" at a frequency of 2Hz
        iii. Blink LED without using "delay()" at a frequency of 8Hz
    b. Save your file on your computer and open a new sketch to begin part 3. You will need to come back to this sketch at the end of part 3.
3. PWM Piecewise Function
    a. Re-create the circuit from Week 2 Lab with the potentiometer and 5 Volt power supply but this time use the 5V from the Arduino.
    b. Measure the value of the output from the potentiometer and display it in the serial monitor.
        i. Show the raw value as well as the voltage equivalent.
    c. Below is a piecewise function that you will turn into code that is dependent on your potentiometer. Your potentiometer is the input that the piecewise function is dependent on. The output is the PWM that goes into your LED. The function is displayed below:

$$PWM(pot) = \begin{cases} PWM_{\max} - pot, & pot \le PWM_{max} \\ 0, & PWM_{max} < pot \le 3 * (PWM_{max} + 1) \\ pot - 3 * (PWM_{max} + 1), & pot > 3 * (PWM_{max} + 1) \end{cases}$$

        i. Things to consider before programming:
            1. What is the PWM_max?
            2. What is the range of the values you read from the potentiometer? What are the units?
            3. Why is the PWM_max + 1 necessary instead of just PWM_max? Be able to explain what happens if the +1 was not in the piecewise function.
        ii. Upon completion, turn it into a callable function if you have not done so already and call it from your void loop.

   d. Replace the **third state** of your “switch case” with the PWM(pot) piecewise function.
4. Show a TA your final work to complete this section and move on to *general*

General:

The purpose of these problems is to challenge you to learn self-dependence when it comes to programming and struggling through problems. The TA’s will help you but only after you have demonstrated an exhaustive search of online resources and each other. You can refer to any resource you might need, you can look in the guide for a start.

1. Write a sketch that displays, in the serial monitor, the n terms of harmonic series and their final sum where n = 10.

   Harmonic Series: 1 + 1/2 + 1/3 + 1/4 + 1/5 ... 1/n terms

2. Write a sketch that displays the sum of the series [1+x+x^2/2!+x^3/3!+....] up to (x^5/5!) and one for (x^10/10!).
   a. Note: you do not need to write two separate sketches, instead you should be able to just modify a variable in your code to obtain both solutions.

3. Write a sketch to display the perfect numbers between 1 and 50 in the serial monitor.