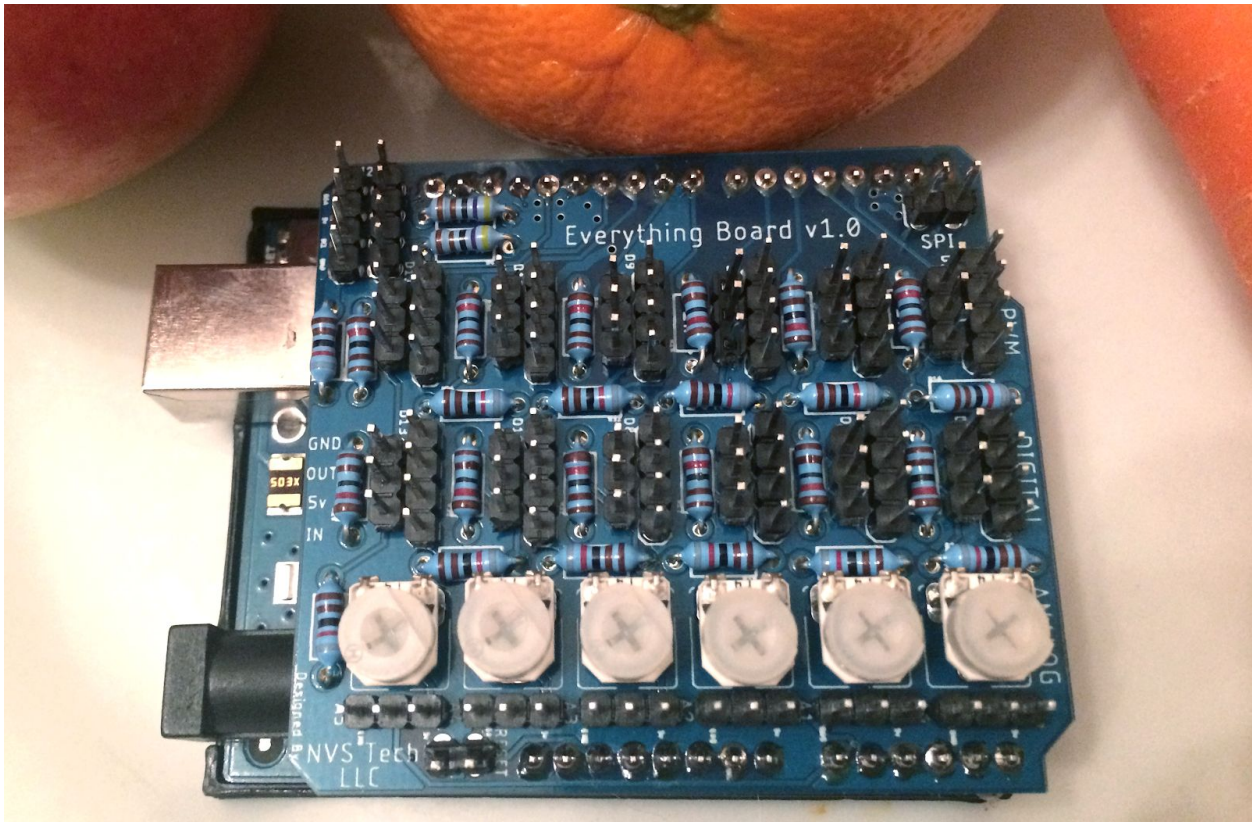
Prototyping
Fall 2018

The Arduino Everything Board



What is it?

The Everything Board is an Arduino Shield which is a type of Printed Circuit Board designed to attach to the top of an Arduino microcontroller to grant added functionality. The Arduino Everything Board is designed for easy, quick prototyping of physical computing projects. It provides common circuitry which is used for sensors and actuators such as buttons, switches, LDR, FSR, flex sensors, and more. It is designed to save you time on your projects. Instead of having to use a breadboard for prototyping, the Everything Board does most of the electronics work for you so you can focus on testing your ideas instead of mucking around with electronics. Although it is designed to be "plug and play," you still need to know what pins to plug your components into. This document serves as a cheat sheet which gives you all the information you need to get your project off the ground.

How to Use This Document

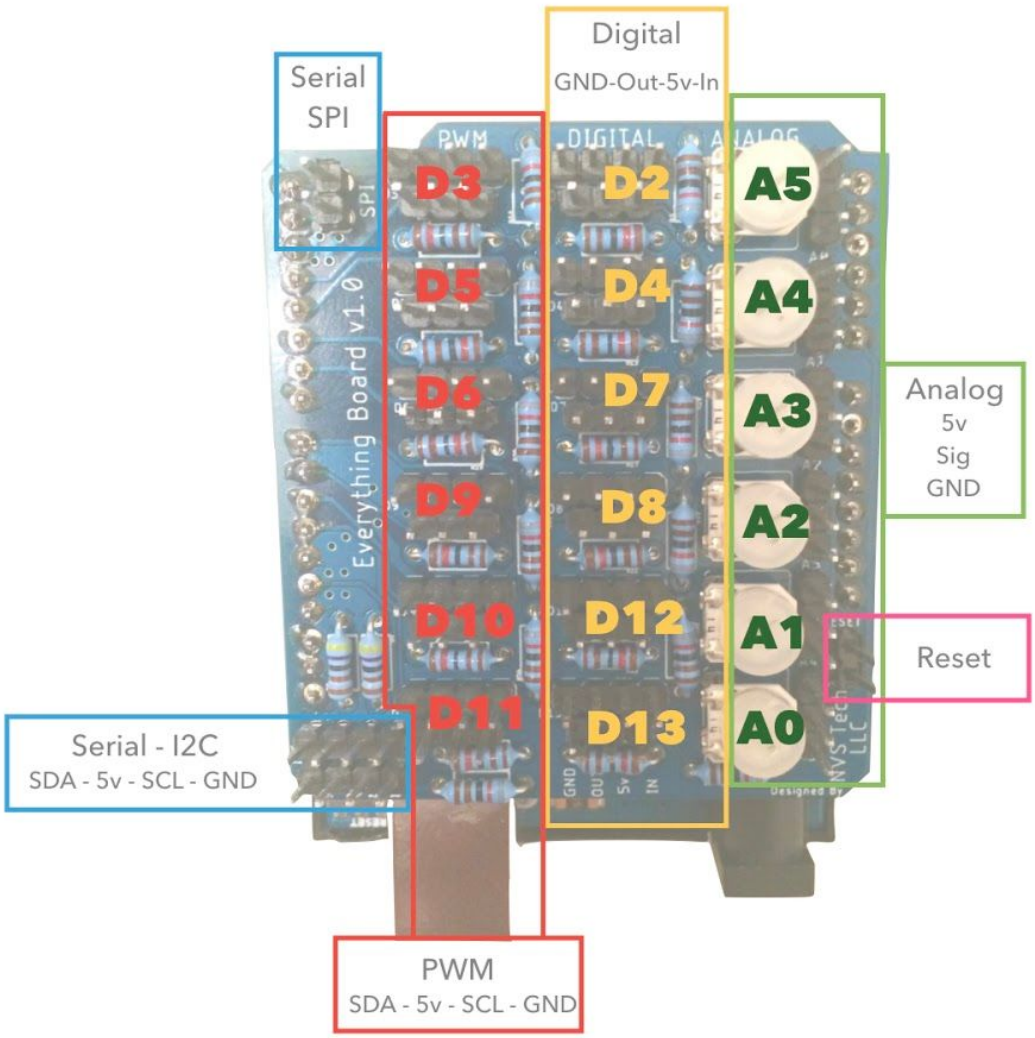
To use the Everything Board, read the board overview section to get a general idea of the functionality it provides. Then move on to the section which addresses the type of sensor you want to use. Follow the directions for setting the jumper pins and adjusting the trim pot (if applicable) and plug in your sensor or actuator. Then download the code examples from https://github.com/nathanshaw/CCA-Arduino and navigate to the folder named after your sensor. Copy the code contained in the example into a new Arduino file,, change the pin numbers (if needed) and run the example. Modify the code using filtering (from the filtering folder) and by adding the appropriate processing and mapping of your input data. Then move on to the next sensor or actuator (keeping your old one plugged into the board). After you get all your sensors and actuators working move on to combining the code from all your component specific files into a master file named after your project.

MAKE SURE NOT TO MODIFY THE EXAMPLE CODE ITSELF. Add the functionality one sensor at a time. First get the code to read and print out the data from all your inputs, then one by one map the input data to your outputs. If you get stuck, ask for help from the TA or Professor. If you are not in class search Google for an answer and if you are still unable to find a solution post your problem, along with the code, to Google Classroom and we will figure it out together.

Remember to save your work often and use some form of version control. That can be as simple as adding a v1, v2, v3, to the end of your file name or by using a tool such as github.com or bitbucket.com. It is a good idea to make a new version as soon as you add a new feature. Whenever you program does something new and is working save it as a new version. This way if the code breaks (and it will) you have a cushion to fall back on and don't lose all your work.

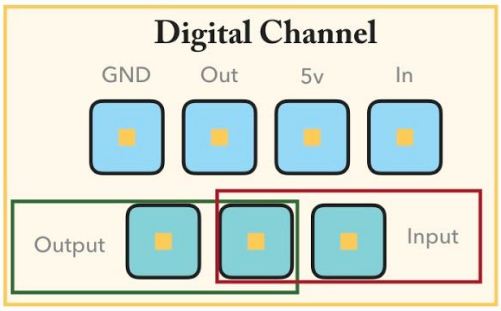
Overview

Just like the Arduino Uno itself, the board is broken up into four main sections: Digital, PWM, Serial (I2C and SPI) and Analog. The sections are labeled on the board for reference. While the shield may seem complicated and intimidating, there is a large amount of repetition. Each section contains the same repeated circuity and inputs/outputs so one you are able to get one sensor working with a section you can get it working for any of the inputs/output for that section. **Do not blindly plug things into your board or you might damage the Arduino, please instead refer to the diagrams below.**

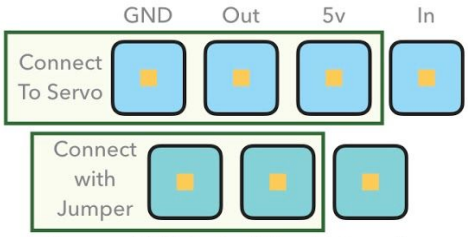


Digital Section

The digital section provides support for sensors which have a binary function. In other words, it works for sensors which have a simple on/off state and for controlling actuators with a simple on/off functionality. This includes: LEDs, Buttons, Switches, Proximity Sensors (not IR or Ultrasonic rangefinders) and others.

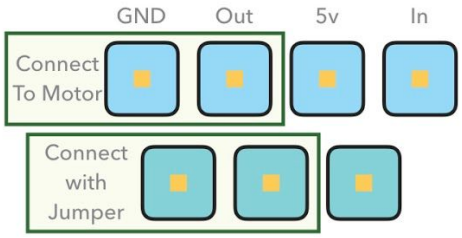


Servo



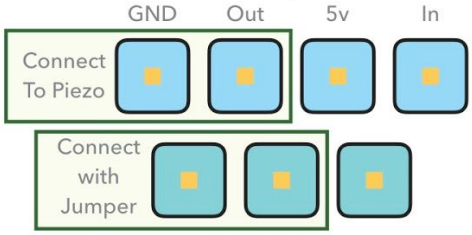
Always use PWM pins 9 + 10 first with Servos (they can also be controlled using the other PWM and Digital Pins). When you use Servos you will be unable to use pins 9 and 10 for anything other than servo control.

DC Motor

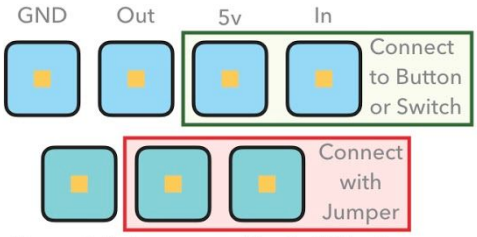


Please note that if you use the "Digital" section for controlling motors you will only be able to turn them on or of. For controlling the speed of a motor use the PWM section.

Piezo Output

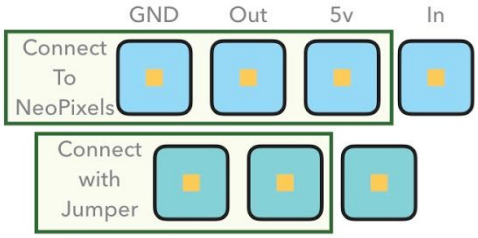


Button/Switch



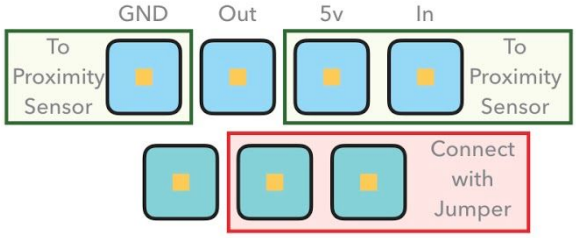
Please note that you can use either the PWM or the Digital section for buttons and switches. Even if the switch has three connectors you only need to use two.

Neo Pixels



Please note that you can use any of the Digital or PWM pins to control NeoPixels.

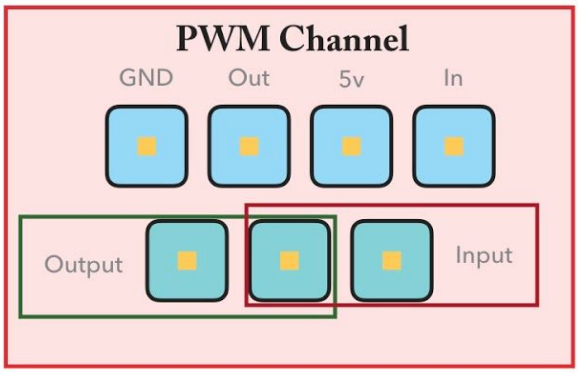
Proximity Sensor

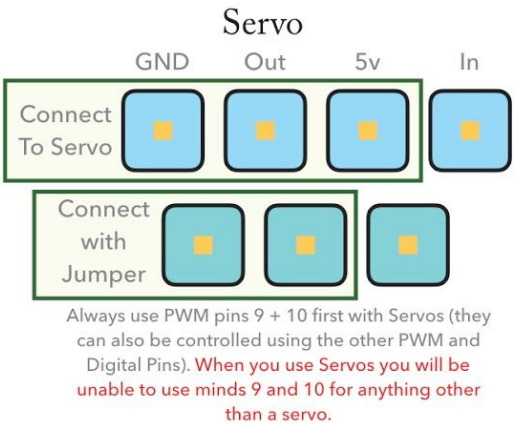


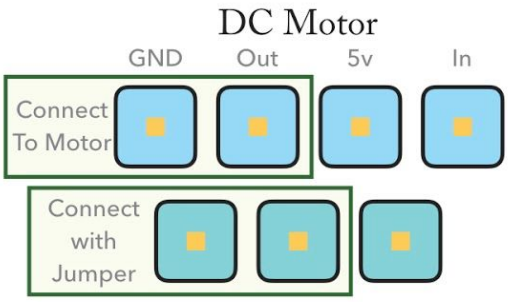
Please note that this is for the "dome" style proximity sensors, for IR Sensors or Ultrasonic rangefinders please refer to different diagrams.

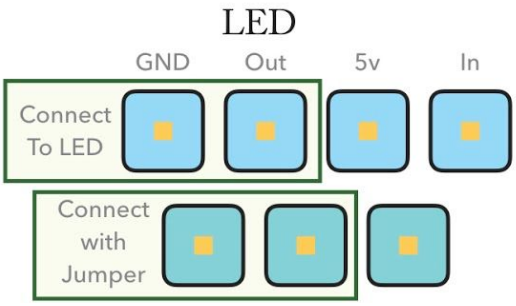
PWM Section

The PWM section functions in the same manner as the Digital section when used for input sensors but gives much more control with certain Actuators. Instead of simply turning an actuator on and off, we are able to give it a signal with a varying amount of power (ranging from 0-255). This means we are able to control the brightness of an LED or the speed of a motor instead of simply just turning them on and off.



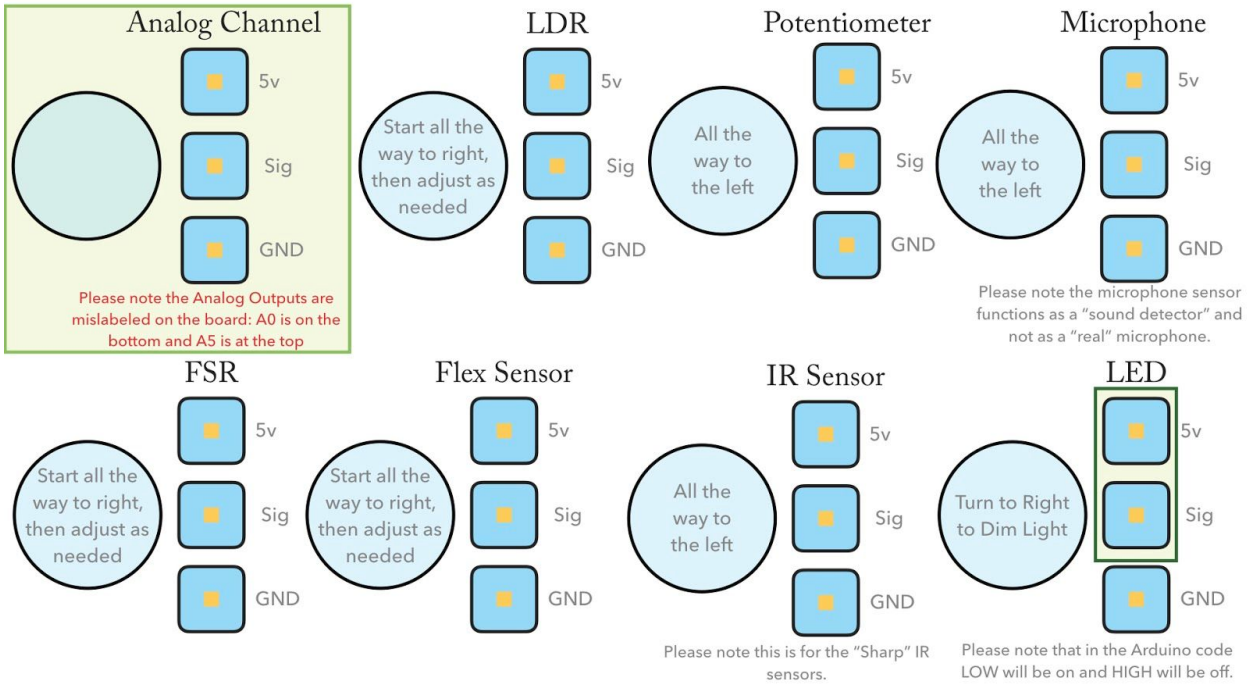






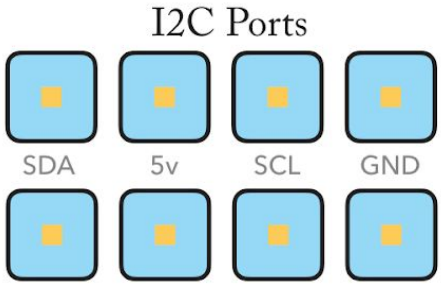
Analog Section

The Analog Section allows you to read values from various types of continuous (analog) sensors. While the digital section exhibits binary (on/off) polling of sensors, the analog section gives you in between values with a range from 0-1023. Analog inputs are able to tell us, for instance, how hard a FSR sensor was pressed, or how close someone is to an IR Sensor. Please note that the Analog section can be used in the same manner as the Digital section, but exhibits a different pinout and does not contain the same circuitry. This means that if you run out of available Digital input/outputs you are able to use the Analog section, but might have to add a resistor or two to your circuitry and do some minor code modifications. The small dials on the board are used for some sensors but not all. Please reference below for instructions on how to adjust their position for your particular application.

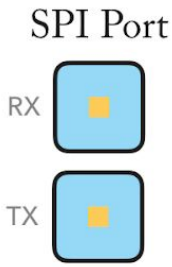


Serial Section

The Serial section is broken into two parts on the PCB: SPI and I2C. I2C and SPI are the two most common serial communication protocols which are commonly used for "complicated" sensors such as a RGB color sensor, LED display, and 9-DOF. Whenever possible use the I2C ports (there are two) instead of the SPI ports. I2C is much easier to use and is what we will focus on in this class.



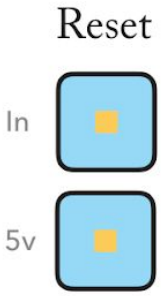
I2C Sensors include: Time of Flight, 9DOF, RGB Color Sensor, Cap Touch, and many others. Please note, the pin order on board might not directly mirror your sensor, make sure you wire everything correctly.



Please not that you should not be using SPI unless you have to. Please use I2C for serial sensor whenever possible.

Reset Section

The reset section consists of two pins which a button or switch can be connected to. When pressed, or toggled, the sensor will restart the Arduino. Most projects will not require this functionality, but it is available for use if you like.



Attach a button or switch to these two pins if you want a hardware reset built into your design.