Electrical Engineering and Computer Science
Colorado School of Mines

Prepared by Vibhuti Dave and Tyrone Vincent


# EENG 498: SEED Lab
# Fall 2015
# Exercise 1: Getting Started with the Arduino

- **Introduction**

  The purpose of this exercise is to get yourself familiar with an Arduino board, its hardware features and how to interact with it using your PC/laptop. These basic exercises will set up the foundation to be able build a much more complex system; a mobile self-balancing robot.

- **The Hardware**

  Recall that a micro-controller is a small computer with a processor, memory and programmable input/output peripherals. You have been provided with an Arduino Uno Board that uses an ATmega328 microcontroller. You should bookmark the pages with the technical specifications of the board as well as the datasheet of the microcontroller.

  1. What frequency does the microcontroller on the uno operate at?
  2. How many digital I/O pins do you have access to? Provide examples where you might use the PWM functionality of those pins?
  3. How many analog I/O pins do you have access to?
  4. What is the operating voltage?
  5. What is the recommended input voltage for the microcontroller?
  6. How many external interrupts are available?

- **The Software**

  To be able to do some basic projects, you will need to write simple programs (also called sketches) and upload them on the board. The Arduino Software (IDE) will help you do that. The Arduino language is based on C and C++. All C and C++ constructs work with the Arduino. This software has been installed on all machines in BB 305. You can also install it on your personal laptops if you prefer that. It runs on Windows, Mac OS X and Linux. You can find a step by step guide for Windows, Mac OS X, and Linux on Arduino's Website. After installation refer to the Arduino Development Environment guide to understand its purpose and what it helps you do.

- **Tutorials (Click on the links to get to the basic tutorials on Arduino's website)**

  1. Bare Minimum
  2. Blink an LED (The "Hello World" of microcontrollers")
  3. Digital Read Serial
  4. Analog Read Serial
  5. Button
  6. Analog Input

1. What are the two functions that will always be a part of an Arduino sketch? Explain the purpose of each of these functions.
2. What is the purpose of the delay function? What is the main drawback of using the delay function to control timing?
3. What are the alternatives to using the delay function?

- **Challenges**
  1. Hook up 4 LEDs to pins 2 – 5. Use arrays to turn on each one in order until all of them have lit up and extinguish them in order until every one of them is off.
  2. Use two pushbuttons to act as "gas" and "brake. The "gas" button should speed up the blinking rate of the LED, and the "brake" button should slow it down.
  3. Make a digital thermometer. Break the challenge down in to smaller tasks before putting it all together. Ensuring that each individual piece works makes debugging much easier.
     a. Use serial.print functions to keep track of what is happening in your code and debugging.
     b. Use the TMP36 chip as the temperature sensor and send the values to the serial monitor. More information can be found here.
     c. Interface an LCD screen with the Arduino. You will need to include the LCD library that provides functions to work with the LCD. You can make your own libraries as well if you need to, later on based on the kind of hardware you use. The LCD library is a standard library that comes pre-installed with the Arduino IDE. Hook up the LCD and follow the "Hello World" tutorial. Play around with other LCD tutorials if you need to.
     d. Bring everything together such that the temperature is displayed on the LCD in Celsius and Fahrenheit.

- **Documentation Deliverables**
  - Answers to all questions posed in this handout
  - Upload sketches for all challenge exercises. Comment your code.

- **Demonstrations**
  - You will be expected to demonstrate and answer questions pertaining to Challenges 1 and 3.

## Appendix A

### A.1 Constants

| Name | Description |
|---|---|
| HIGH/LOW | Used to set pin state |
| INPUT/OUTPUT | Used with pinMode |
| Integer constants | Used when you put an integer in your code. Can be followed by L for long and U for unsigned |
| Floating Point Constants | Used when you have floating point numbers in your code. Use e or E for scientific notation |

### A.2 Data Types

| Name | Description |
|---|---|
| void | Used in function declarations. Implies nothing is returned. |
| bool | Holds true or false |
| char | 8 bit character. Ranges from -127 to 127 |
| unsigned char | 8 bit character. Ranges from 0 – 255 |
| byte | Same as unsigned char |
| int | -32,768 to 32767 |
| unsigned int | 0 to 65,555 |
| Word | Same as unsigned int |
| long | -2,147,483,648 to 2,147,483,647 |
| unsigned long | 0 to 4,294,967,295 |
| float - | -3.4028235E+38 to 3.4028235E+38 |
| double | same as float If doing math with floats, you need to add a decimal point, otherwise it will be treated as an int |
| string | array of characters - defined as char strName[5]; |
| String | a String Object |

### A.3 Conversion

| Name | Description |
|---|---|
| char() | converts to the char type |
| byte() | converts to the byte type |
| int() | converts to the int type |
| word() | converts to the word type |
| long() | converts to the long type |
| float() | converts to the float type |

### A.4 Variable Scope & Qualifiers

| Name | Description |
|---|---|
| static | preserve the data between calls, visible only to |

| | |
|---|---|
| | that scope (file or function) |
| volatile | for when a variable can be changed by something external. Normally in arduino, this is for variables that are changed in an interrupt service routine (ISR) |
| const | the variable cannot be changed |