**CHAPTER 3**

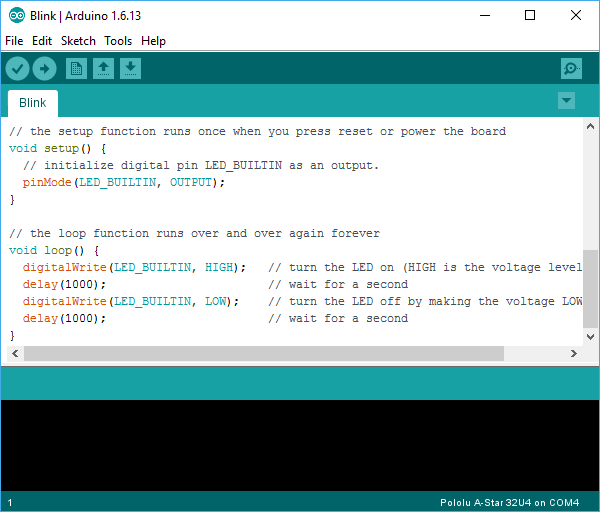
**SOFTWARE REQUIREMENTS**

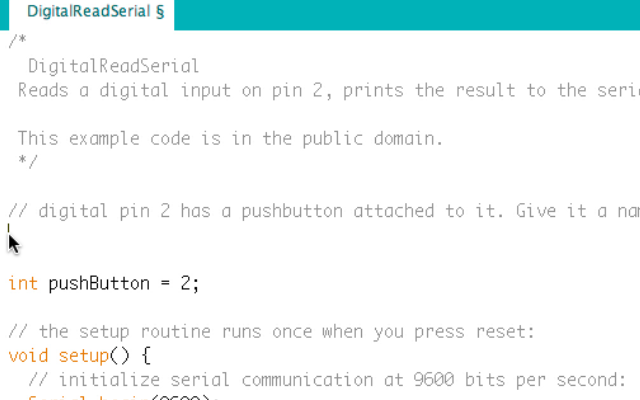
Software used in this project for uploading code onto Arduino is Arduino IDE.

**3.1 INTRODUCTION TO ARDUINO IDE**

IDE stands for Integrated Development Environment. Pretty fancy sounding, and should make you feel smart any time you use it. The IDE is a text editor-like program that allows you to write Arduino code. When you open the Arduino program, you are opening the IDE. It is intentionally streamlined to keep things as simple and straightforward as possible. When you save a file in Arduino, the file is called a sketch – a sketch is where you save the computer code you have written. The coding language that Arduino uses is very much like C++ (“see plus plus”), which is a common language in the world of computing. The code you learn to write for Arduino will be very similar to the code you write in any other computer language – all the basic concepts remain the same – it is just a matter of learning a new dialect should you pursue other programming languages.



The code you write is “human readable”, that is, it will make sense to you (sometimes), and will be organized for a human to follow. Part of the job of the IDE is to take the human readable code and translate it into machine-readable code to be executed by the Arduino. This process is called compiling. The process of compiling is seamless to the user. All you have to do is press a button. If you have errors in your computer code, the compiler will display an error message at the bottom of the IDE and highlight the line of code that seems to be the issue. The error message is meant to help you identify what you might have done wrong – sometimes the message is very explicit, like saying, “Hey – you forget a semicolon”, sometimes the error message is vague. Why be concerned with a semicolon you ask? A semicolon is part of the Arduino language syntax, the rules that govern how the code is written. It is like grammar in writing. Say for example we didn’t use periods when we wrote – everyone would have a heck of a time trying to figure out when sentences started and ended. Or if we didn’t employ the comma, how would we convey a dramatic pause to the reader?

And let me tell you, if you ever had an English teacher with an overactive red pen, the compiler is ten times worse. In fact – your programs WILL NOT compile without perfect syntax. This might drive you crazy at first because it is very natural to forget syntax. As you gain experience programming you will learn to be assiduous about coding grammar.

**3.1.1 THE SEMICOLON**

A semicolon needs to follow every statement written in the Arduino programming language. For example, …

Int LedPin=9;

In this statement, I am assigning a value to an integer variable (we will cover this later), notice the semicolon at the end. This tells the compiler that you have finished a chunk of code and are moving on to the next piece. A semicolon is to Arduino code, as a period is to a sentence. It signifies a complete statement.

### 3.1.2 THE DOUBLE BACKSLASH FOR SINGLE LINE COMMENTS //

### Comments are what you use to annotate code. Good code is commented well. Comments are meant to inform you and anyone else who might stumble across your code, what the heck you were thinking when you wrote it. A good comment would be something like this…

Now, in 3 months when I review this program, I know where to stick my LED. Comments will be ignored by the compiler – so you can write whatever you like in them. If you have a lot you need to explain, you can use a multi-line comment, shown below…

//This is an example

Comments are like the footnotes of code, except far more prevalent and not at the bottom of the page.

**3.1.3 THE CURLY BRACES**

Curly braces are used to enclose further instructions carried out by a function (we discuss functions next). There is always an opening curly bracket and a closing curly bracket. If you forget to close a curly bracket, the compiler will not like it and throw an error code.

Void loop (){

}

Remember – no curly brace may go unclosed!

**3.1.4 FUNCTION ( )**

Functions are pieces of code that are used so often that they are encapsulated in certain keywords so that you can use them more easily. For example, a function could be the following set of instructions…

This set of simple instructions could be encapsulated in a function that we call WashDog. Every time we want to carry out all those instructions we just type WashDog and voila – all the instructions are carried out. In Arduino, there are certain functions that are used so often they have been built into the IDE. When you type them, the name of the function will appear orange. The function pinMode(), for example, is a common function used to designate the mode of an Arduino pin.

What’s the deal with the parentheses following the function pinMode? Many functions require *arguments* to work. An argument is information the function uses when it runs. For our WashDog function, the arguments might be dog name and soap type, or temperature and size of a bucket.

pinMode(13, OUTPUT);

The argument 13 refers to pin 13, and OUTPUT is the mode in which you want the pin to operate. When you enter these arguments the terminology is called passing. You pass the necessary information to the functions. Not all functions require arguments, but opening and closing parentheses will stay regardless though empty.

Notice that the word OUTPUT is blue. There are certain keywords in Arduino that are used frequently and the color blue helps identify them. The IDE turns them blue automatically. Now we won’t get into it here, but you can easily make your own functions in Arduino, and you can even get the IDE to color them for you. We will, however, talk about the two functions used in nearly EVERY Arduino program.

**3.1.5 VOID SETUP ( )**

The function, setup(), as the name implies, is used to set up the Arduino board. The Arduino executes all the code that is contained between the curly braces of setup() only once. Typical things that happen in setup() are setting the modes of pins, starting You might be wondering what void means before the function setup(). Void means that the function does not return information. Some functions do return values – our DogWash function might return the number of buckets it required to clean the dog. The function analogRead() returns an integer value between 0-1023. If this seems a bit odd now, don’t worry as we will cover every common Arduino function in depth as we continue the course.

Let us review a couple things you should know about setup()…

1. setup() only runs once.

2. setup() needs to be the first function in your Arduino sketch.

3. setup() must have opening and closing curly braces.

**3.1.6 VOID LOOP ( )**

You have to love the Arduino developers because the function names are so telling. As the name implies, all the code between the curly braces in loop() is repeated over and over again – in a loop. The loop() function is where the body of your program will reside. As with setup(), the function loop() does not return any values, therefore the word void precedes it.

Does it seem odd to you that the code runs in one big loop? This apparent lack of variation is an illusion. Most of your code will have specific conditions laying in wait which will trigger new actions.

If you have a temperature sensor connected to your Arduino for example, then when the temperature gets to a predefined threshold you might have a fan kick on. The looping code is constantly checking the temperature waiting to trigger the fan. So even though the code loops over and over, not every piece of the code will be executed every iteration of the loop.

**3.2 INTRODUCTION ARDUINO LIBRARIES**

Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the built-in LiquidCrystal library makes it easy to talk to character LCD displays. There are hundreds of additional libraries available on the Internet for download. The built-in libraries and some of these additional libraries are [listed in the reference](https://www.arduino.cc/en/Reference/Libraries). To use the additional libraries, you will need to install them.

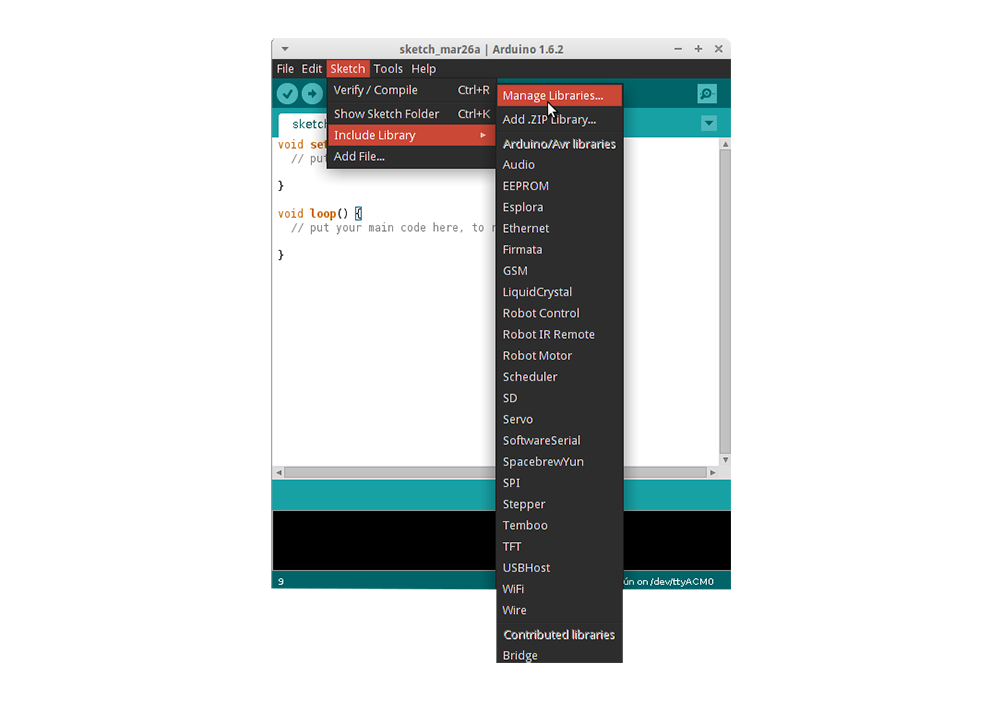
Arduino libraries are managed in three different places: inside the IDE installation folder, inside the core folder and in the libraries folder inside your sketchbook. The way libraries are chosen during compilation is designed to allow the update of libraries present in the distribution. This means that placing a library in the “libraries” folder in your sketchbook overrides the other libraries versions.

The same happens for the libraries present in additional cores installations. It is also important to note that the version of the library you put in your sketchbook may be lower than the one in the distribution or core folders, nevertheless it will be the one used during compilation. When you select a specific core for your board, the libraries present in the core’s folder are used instead of the same libraries present in the IDE distribution folder.

Last, but not least important is the way the Arduino Software (IDE) upgrades itself: all the files in Programs/Arduino (or the folder where you installed the IDE) are deleted and a new folder is created with fresh content. This is why we recommend that you only install libraries to the sketchbook folder so they are not deleted during the Arduino IDE update process.

**3.2.1 HOW TO INSTALL A LIBRARY**

To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.6.2). Open the IDE and click to the "Sketch" menu and then Include Library > Manage Libraries.



Then the Library Manager will open and you will find a list of libraries that are already installed or ready for installation. In this example we will install the Bridge library. Scroll the list to find it, click on it, then select the version of the library you want to install. Sometimes only one version of the library is available. If the version selection menu does not appear, don't worry: it is normal.



Finally click on install and wait for the IDE to install the new library. Downloading may take time depending on your connection speed. Once it has finished, an Installed tag should appear next to the Bridge library. You can close the library manager. You can now find the new library available in the Sketch > Include Library menu. If you want to add your own library to Library Manager, follow [these instructions](https://github.com/arduino/Arduino/wiki/Library-Manager-FAQ" \t "_blank).

**3.3 HOW TO CONNECT ARDUINO BOARD**

If you're using a serial board, power the board with an external power supply (6 to 25 volts DC, with the core of the connector positive). Connect the board to a serial port on your computer. On the USB boards, the power source is selected by the jumper between the USB and power plugs. To power the board from the USB port (good for controlling low power devices like LEDs), place the jumper on the two pins closest to the USB plug. To power the board from an external power supply (needed for motors and other high current devices), place the jumper on the two pins closest to the power plug. Either way, connect the board to a USB port on your computer. On Windows, the Add New Hardware wizard will open; tell it you want to specify the location to search for drivers and point to the folder containing the USB drivers you unzipped in the previous step.

The power LED should go on.

**3.4 HOW TO UPLOAD A PROGRAM**

The content of circuits and Arduino sketches can vary greatly. Before you get started, there is one simple process for uploading a sketch to an Arduino board that you can refer back to.

Follow these steps to upload your sketch:

1. Connect your Arduino using the USB cable.

The square end of the USB cable connects to your Arduino and the flat end connects to a USB port on your computer.

1. Choose Tools→Board→Arduino Uno to find your board in the Arduino menu.

You can also find all boards through this menu, such as the Arduino MEGA 2560 and Arduino Leonardo.

1. Choose the correct serial port for your board.

You find a list of all the available serial ports by choosing Tools→Serial Port→ comX or /dev/tty.usbmodemXXXXX. X marks a sequentially or randomly assigned number. In Windows, if you have just connected your Arduino, the COM port will normally be the highest number, such as com 3 or com 15.

Many devices can be listed on the COM port list, and if you plug in multiple Arduinos, each one will be assigned a new number. On Mac OS X, the /dev/tty.usbmodem number will be randomly assigned and can vary in length, such as /dev/tty.usbmodem1421 or /dev/tty.usbmodem262471. Unless you have another Arduino connected, it should be the only one visible.

1. Click the Upload button.

This is the button that points to the right in the Arduino environment. You can also use the keyboard shortcut Ctrl+U for Windows or Cmd+U for Mac OS X.