HANDS-ON LAB INSTRUCTION SHEET – Learning Kit MODULE 3

The Arduino IDE and the Basic LED Blink Project

NOTES:
If you did not finish Module 1 AND Module 2, be sure to finish them NOW *before starting this Module or you will fall behind the rest of the class.* ***Labs MUST be done in order.***

BILL OF MATERIALS

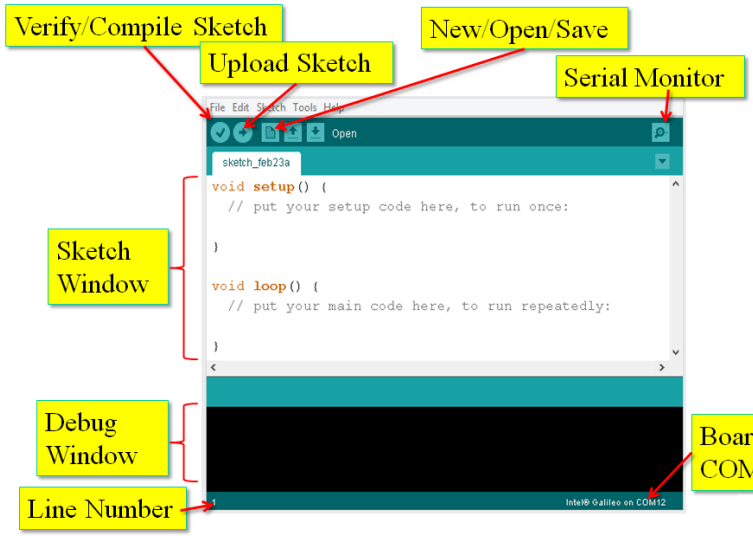
(1) **Arduino UNO R3 Microcontroller & USB Cable**
(1) **RED** Light-Emitting Diode (LED)
(1) **1KΩ (1000 Ohm)** Resistor (*actually any value **220Ω** through **1KΩ***)

3.A. Installing the Arduino IDE

1. Download the Windows or Mac files from **https://www.arduino.cc/en/main/software**
2. Install the IDE Integrated Design Environment) files on your laptop/computer accepting all default settings.
3. Plug your Arduino into the provided USB Cable and plug the cable into a USB port on your computer. There are a few lights on the board which may light up***. Its OK!***

Figure 3.1. Attaching the Arduino

4.Open the IDE on your desktop. It should look like this:



QUESTION 3.1 →

Figure 3.2. Arduino IDE

QUESTION 3.1: ***(See figure 3.2***) Do a screen capture (or photo) of your installed IDE and embed it into your results page.

Revision 211021: This document is available electronically at:
http://www.CharlesRubenstein.com/222/**03Lab_LK.pdf**

Question 3.2 →

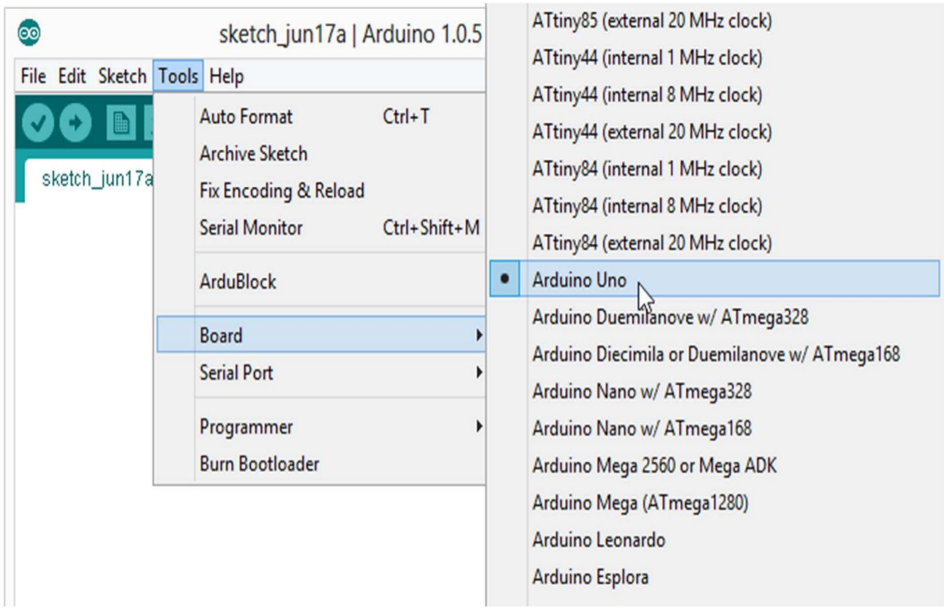


Figure 3.3. Tools: Board Menu Selections

QUESTION 3.2: ***(See figure 3.3***) What is written on the top line of the IDE which includes the IDE version?

3.2) ANS: _____

5. Select the "Arduino Uno" board from the Tools Menu as shown in figure 3 by pressing:
Tools → Board → Arduino Uno
6. Your computer communicates with the Arduino microcontroller board via a serial "**COM**" port through a **USB-Serial adapter**. The Board drivers must be properly installed of you will not be able to upload compiled programs to your microcontroller. Check to make sure that the port *SEES* your Board by opening the serial port: **Tools → Serial Port** and click on whichever USB port is appropriate…

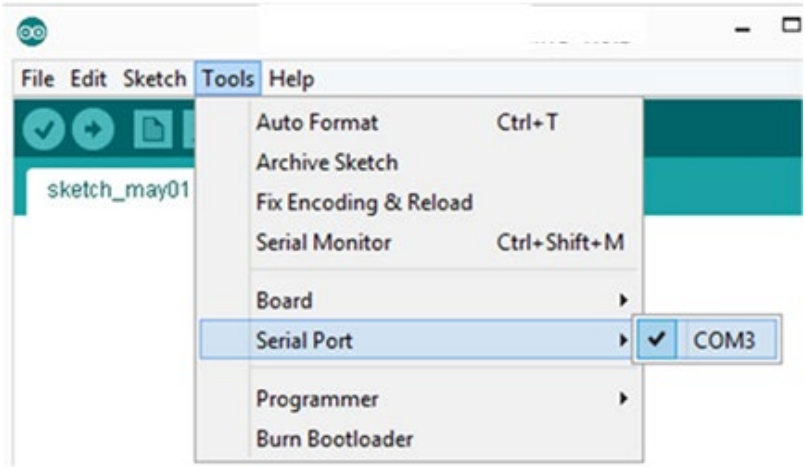


Figure 3.4. Tools: Serial Port Menu selections

QUESTION 3.3: After plugging in and setting up your Arduino Board, what does the IDE's bottom line, which includes the line number and serial COM port information (*See Figure 3.2.)* say?

3.3) ANS: _____

Revision 211021: This document is available electronically at:
http://www.CharlesRubenstein.com/222/**03Lab_LK.pdf**

3.B. Arduino Manuals

One of the Arduino kits we use in the lab is *from Elegoo and can also be downloaded from the class website* **http://www.charlesrubenstein.com/222/ElegooManual.pdf**

The other Starter Kit 'manual' we will use is the Vilros Ultimate Stater Kit. See ***vilros.pdf*** *on our class website* ***http://www.charlesrubenstein.com/222/vilros.pdf***

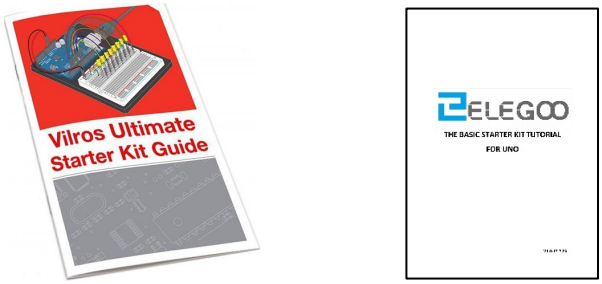


Figure 3.5. What the manuals would look like when printed…

3.C. Installing the Example Sketches:

There are many built-in coding examples included with the downloaded IDE app,
See:

https://www.arduino.cc/en/Tutorial/BuiltInExamples

We will be using the USKCODE zip file from**: http://www.vilros.com/uskcode** (or from the class website **http://www.charlesrubenstein.com/222/USKcode.zip**) to provide us with a series of example sketches that we can review, understand, and then modify as needed.

The Elegoo kit also has some example code for its exercises. These are online on the class website at **http://www.charlesrubenstein.com/222/ElegooCode.zip**

In either case, download the zip file to your computer and open the zip file – do not modify the contents as they need to be in the manner you see so you can install them on your IDE and use them in the labs.

On a PC install the example files in the IDE on your computer by going to
Start → Programs → Arduino → examples
and then copy the USK Guide Code folder (and/or Elegoo folder) into the Arduino Examples folder.

On a Mac install the example files in the IDE on your computer by opening your Applications Folder or control click on the installed '**Arduino.app**' and select 'Show Package Contents' Open the folder: **Contents → Java → Examples** and copy the "USK Guide Code" folder (and/or Elegoo folder to "Examples."

NOTE: The file type for Arduino code examples is ".ino" and all coding *MUST be saved as plain text*.

3.D. Controlling LEDs

USING THE BLINK SKETCH EXAMPLE CODE

1. Open the IDE on your desktop. Plug the Arduino Board USB cable into your computer.
2. Your computer communicates with the Arduino microcontroller board via a serial port.

Basic BLINK Code – Turning on an LED

The **BLINK** '**program**' or '**sketch**' or '**code**' is one of many built-in coding examples included with the downloaded IDE app. The file *type* for the sketch programs or code examples is "**.ino**" and must be saved as plain text.

The object of the BLINK program is to use the Arduino to turn an LED ON for a set period of time and then to turn it OFF for a set period of time.

The Arduino is commonly coded in milliseconds (= 0.001 seconds) therefore a one second ON or OFF would last for 1000 ms.

If we used 500 ms then the LED would turn ON and OFFF every ½ second:

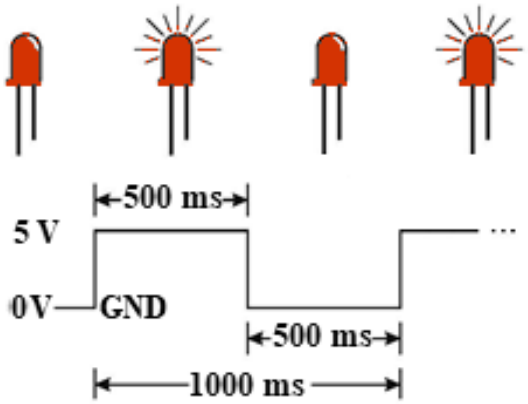


Figure 3.6. LED Duty Cycles

The duty cycle, in percent, is calculated as the time ON divided by the TOTAL time:

$$t_{duty\ cycle} = t_{on} / (t_{on} + t_{off}) \times 100\%$$

QUESTION 3.4.: What is the duty cycle of an LED turned ON for 500 ms and OFF for 500 ms?

3.4 ANS: _____

3.E. ARDUINO PROJECTS

All Arduino microcontroller projects need to have a two-part design associated with them:

1. A physical portion – *the components – and some schematic to show how they are interconnected and connections needed to and from the microcontroller, and*

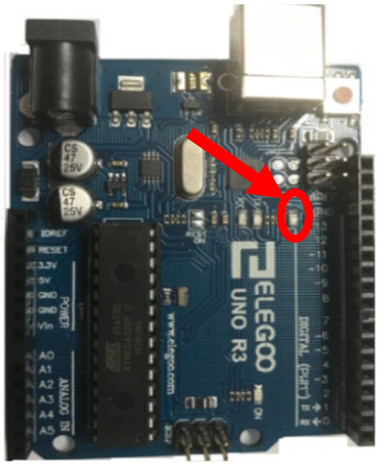
2. A programming portion – *the sketch code – which requires flow charting what we want the microcontroller to do, and when …*

Together, and operating properly, these combine with the Arduino to minimize the number of logical and control components necessary to complete your project as well as make it easy to modify the results without soldering and desoldering individual components.

Revision 211021: This document is available electronically at:
http://www.CharlesRubenstein.com/222/**03Lab_LK.pdf**

3.F. ABOUT THE ARDUINO MICROCONTROLLER

The Arduino Board can provide up to 40 milliamperes (40 mA) to the load on each of the fourteen Digital I/O pins #0 through #13, up to a maximum of 200 mA for the entire Board. Digital I/O Pin #13 is connected to an onboard LED. *Without anything connected to it, the Arduino Board consumes approximately 20 mA.*



← Digital I/O pin #13

Figure 3.7. Location of Digital I/O Pin 13 on the Arduino

If we use Digital I/O Pin #13 - the onboard LED - as our output *we need NO OTHER WIRES nor any external components to see the effects of our program code*.

The basic, functional, schematic for an Arduino Board shows its analog inputs and digital input/output ports and regulated voltage connections needed for powering external circuits.

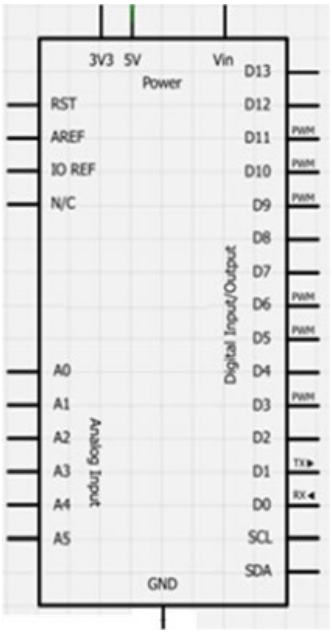


Figure 3.8. Basic Arduino Block Diagram

Revision 211021: This document is available electronically at:
http://www.CharlesRubenstein.com/222/**03Lab_LK.pdf**

The flowchart for an LED control sketch might look like:

Figure 3.9. Flow Chart for Arduino-controlled LED

3.G. THE ARDUINO IDE SKETCH OR CODE PANE

When we installed the IDE the default structure of a program was in the central sketch or code pane. Although it did NOT include the '**Header**' comments portion, it had the other two required parts of any code; the '**Setup**' and '**Loop**' sections.

3.G.1. ARDUINO CODE TYPES

There are four basic sets of Arduino coding instructions. They are:

1. Structure & Flow
- Basic Program Structure
- Control Structures

2. Variables, Arrays and Data
- Data Types
- Constants
- Arrays
- Strings
- Pointer Access
- Qualifiers

3. Operators
- General, Compound, and Bitwise Operators

4. Built-in Functions
- Input/Output Pins
- Time
- Type Conversions
- Math and Random Numbers
- Bits and Bytes

We will not need to know all of the command statements or their syntax (how to use them) but this shows the wide variety of basic commands the Arduino can respond to.

3.G.2. BASIC CODE RULES

The structure of each code statement, function, or command is defined by several rules that allow the IDE 'compiler' (a portion of the IDE that reviews and verifies each line of your code as correct and then converts the 'English' command statements into a binary code that the microcontroller understands but uses only a few bytes of space instead of one byte per letter!). Basic rules include:

1. A **semicolon** "**;**" is required at the end of each command for the IDE's complier to know to move to the next command. Without it you will see a syntax error.
2. **Braces {** and **}** define a block of code. Leave one or the other off and, again, you will confuse the compiler.
3. **Comments**. *When the code is compiled any 'comment' lines are ignored and NOT compiled*. There are two types of comments and they are noted with the "*/*" mark, for example:

Single Line Comments

// a single line of comments, etc. ***NOTE: There are no CLOSING hashes***

Multiple Line Comments

/ lots of remarks and explanations over more than one line
of comments about the code or inputs or outputs. ***MUST have a closing hash /***

4. **Case Sensitivity**. When we define a particular variable in our code, e.g., **LED1** we must use that exact 'spelling and case' everywhere in the code. **LED1** is NOT the same variable name as **Led1** (etc.). Even more challenging is the use of built-in commands since the command is **Serial.println()** which ***is not the same as*** **serial.printin()**
5. The two ***REQUIRED*** parts of a sketch (program) are the Setup and the Loop functions.

3.H. DOWNLOADING THE BLINK PROGRAM

The BLINK program code is accessed within the IDE by clicking:

File → Examples → 01.Basics → Blink

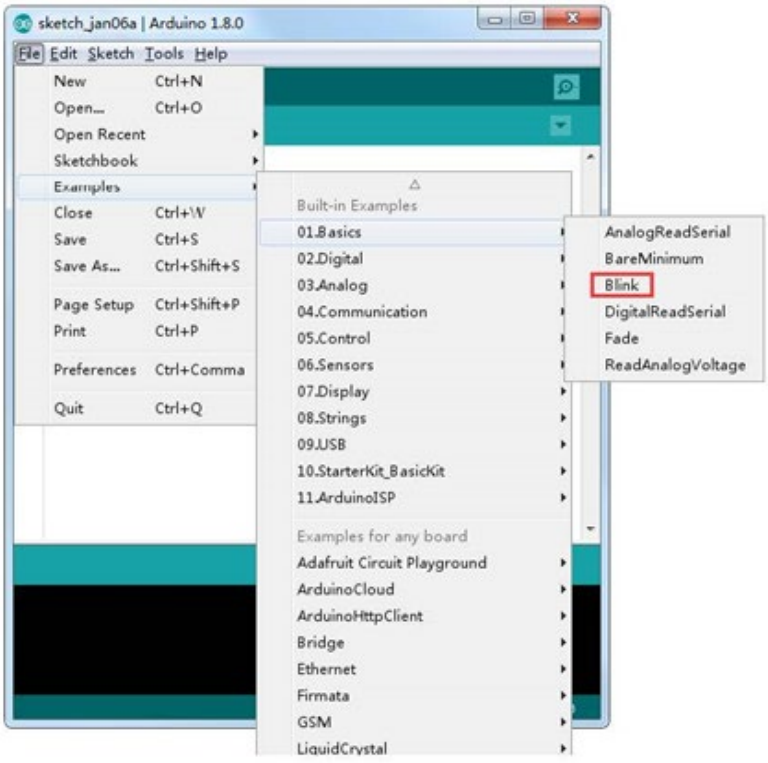


Figure 3.10. **File → Examples → 01.Basics → Blink**

Revision 211021: This document is available electronically at:
http://www.CharlesRubenstein.com/222/**03Lab_LK.pdf**

The resulting BLINK Sketch:

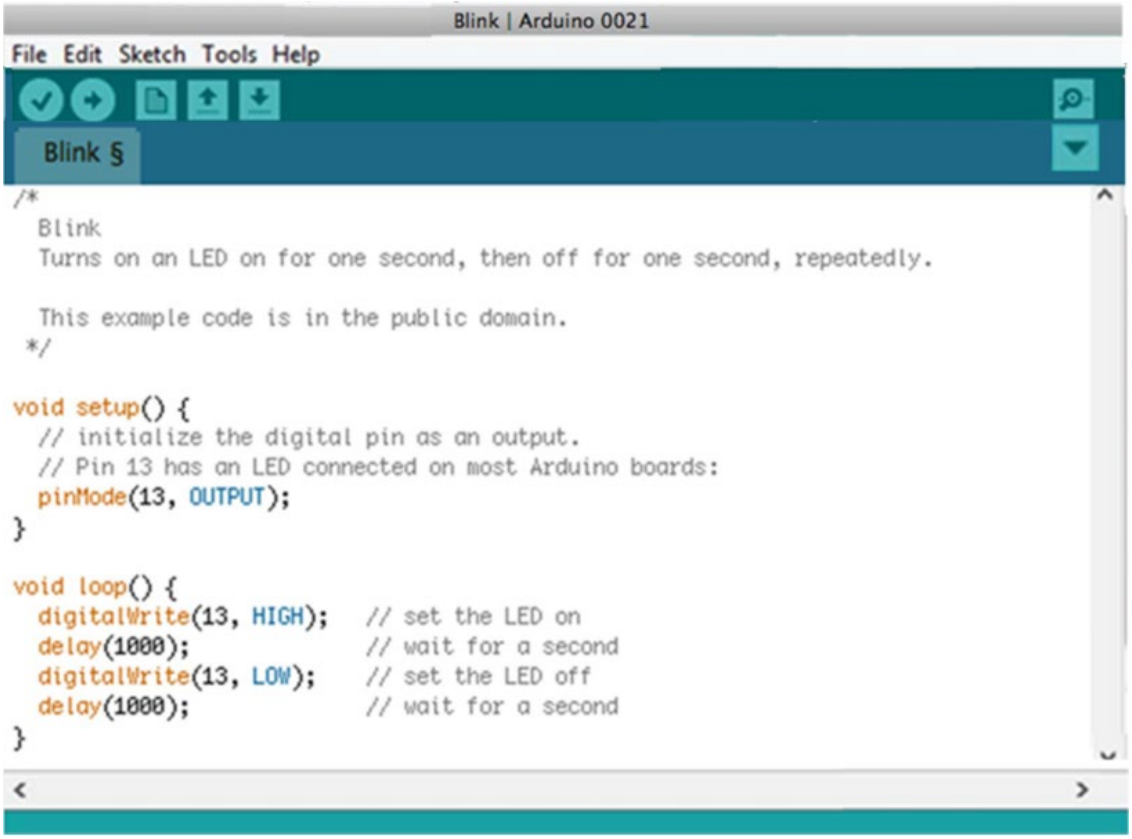


Figure 3.11. **Blink Sketch in the IDE**

ABOUT THE FUNCTIONS IN "BLINK"

3.H.1. SETUP AND LOOP FUNCTIONS

After any ***comments*** about the sketch and what inputs and outputs and programming and any parameter definitions are provided, the next part of any sketch is the setup function:

```
void setup()
      {
  // which runs only once to setup the Arduino to go into the loop function
      }
```

The only other required part of a sketch is the loop function:

```
void loop()
      {
  // which repeats over and over until the power is removed
      }
```

Note that ANY code and commands would be inserted between the braces in either function.

3.H.2. DIGITAL I/O FUNCTIONS IN "BLINK"

These I/O commands are placed inside the braces for the setup or loop functions.
In the Setup function we include one-time instructions that do not change,
For example:

pinMode(pin#,state);

where state = INPUT or OUTPUT
Configures the specified pin to behave either as an input or an output
Syntax: **pinMode(pin, mode);**
Example: pinMode(13, OUTPUT); // sets Digital I/O Pin #13 to output mode

In the Loop function we have instructions which are repeated over and over:

digitalWrite(pin#,state);

where state = HIGH (=ON) or LOW (=OFF)
Writes a HIGH or a LOW value to a digital pin
If set to OUTPUT with pinMode() then 5V for HIGH, 0V (ground) for LOW.
Syntax: **digitalWrite(pin, value);**
Example 1: digitalWrite(13, HIGH); // +5 volts to I/O pin 13
Example 2: digitalWrite(13, LOW); // I/O pin 13 connects to Ground

delay(#);

*where **#** is in milliseconds (1000ms = 1 second)*

Note:
1) ALL Arduino functions are ***CASE-SENSITIVE!***
2) ALL Arduino functions ***MUST*** end in a semicolon "**;**".

3.H.3. Optional Definition of Integer Variables

We can also define a variable just before the Setup function with a name that can then be used instead of a pin number. This technique is handy when you want to make a single change there and not have to change all the lines of code that follow as we will see in our next lab.

3.H.4. DISSECTING THE BLINK CODE

The first line of a **function** is its *definition*, and it has three parts: *return type*, *name*, and *parameter list*. In the function **void setup ( )** the *return type* is **void**, the *name* is **setup**, and the *parameter list* is empty - there is nothing inside the parentheses **( )**. An **empty parameter list** means that these functions *do not need to receive any values* when they are called to do their jobs. **Void** means 'nothing'—when another function calls **setup** or **loop**, these functions *would not return a value* to the function 'calling' them.

Please note that when you code a sketch, before it is used to program the Arduino, the compiler checks the syntax and all rules to see if the code is correct. With few exceptions, if your code is WRONG it will not give you the output that you might have expected, or might not function properly, but like writing HTML for a web page, your Arduino system will PROBABLY not 'blow up' due to the error. **It /might/ blow up if you try to provide more current to your outputs than it can safely handle…**

We will start controlling LEDs with the BLINK Sketch Code and modify it as needed in Lab 4.

Project 3.0: BLINKING THE ONBOARD LED

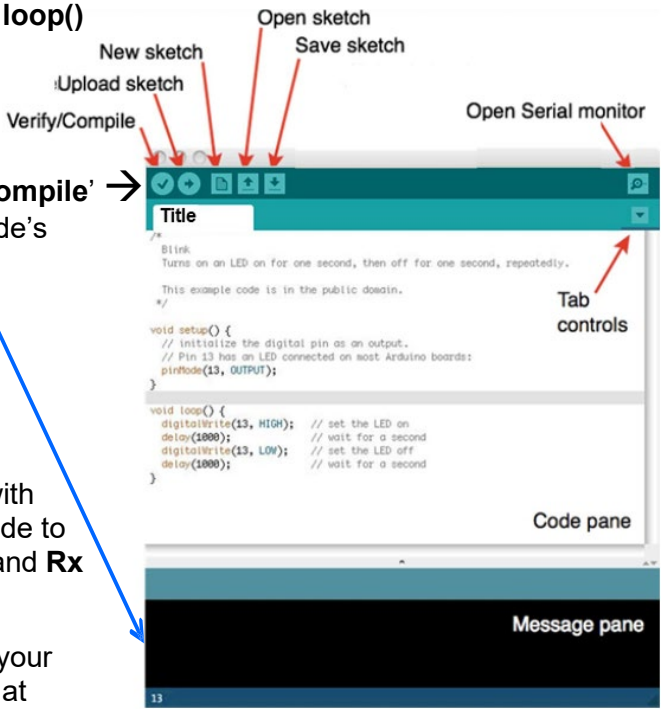
You have already selected the BLINK code example and the code pane should now look like: Let's look at each line of the BLINK code in each of the three sections of the sketch - Header, Setup, and Loop. (*Note Line Numbers and Section names have been added here but are not in the actual code file…)*

THE BLINK SKETCH EXAMPLE *(Section and line #s added)*

```
                                    /Section 1: HEADER & COMMENTS:/
Line 01:  // Blink 1.0: Turn on-Board LED (#13) ON
Line 02:  // 1 second and then OFF for 1 second, Repeat
                                    /Section 2: SETUP:/
Line 03:  void setup() // Runs only once
Line 04:  {           // open block of code
Line 05:  pinMode(13, OUTPUT); // Pin 13 = Output
Line 06:  }           // close block of code
                                    /Secion 3: LOOP Function:/
Line 07:  void loop() // function runs and then returns here
Line 08:  {         // open block of code
Line 09:  digitalWrite(13, HIGH); // Turns ON the "L" LED
Line 10:  delay(1000); // Wait for 1000 milliseconds (1 second)
Line 11:  digitalWrite(13, LOW);     // Turns OFF the "L" LED
Line 12:  delay(1000);  // Wait for 1000 milliseconds (1 second)
Line 13:  } // closes block of code and returns to top of loop()
```

To have the Arduino control the onboard **L** LED you must compile and verify the sketch code and then upload it to the Arduino Board:

Verify the code in your Sketch by clicking the '**Verify/Compile**' → button with your mouse. The IDE will then verify the code's SYNTAX, *but not any coding errors (wrong pin #, etc.). Any error or verification messages will be noted in the "Message Pane" - If the Sketch is 'correct' the IDE "compiles" it for the particular Arduino Board you setup the IDE to work with in* ***Tools****.*
NOTE: EACH ARDUINO BOARD IS DIFFERENT!



Clicking the '**RIGHT ARROW**' ***Upload Sketch*** button with your mouse 'sends' the compiled machine language code to your Board via the USB cable. You should see the **Tx** and **Rx** LEDs onboard your Arduino blinking – ***that's good!***

When the Blink sketch finishes loading the "**L**" LED on your Arduino board should begin blinking **ON** and then **OFF** at one second (*1000 ms*) intervals.

Figure 3.12. Blink Sketch in the IDE

NOTE: *By setting our digital I/O to* ***pin #13*** *we can check out the BLINK code* ***without*** *using any additional components by looking at* ***L*** *– the Arduino's on-board LED.*

Congratulations - you've now programmed a Microcontroller!

End of Lab 3

Revision 211021: This document is available electronically at:
http://www.CharlesRubenstein.com/222/**03Lab_LK.pdf**