Installing and Navigating the Arduino IDE
What is an IDE

• An **integrated development environment** (IDE) is software designed to increase productivity by integrating useful tools into one application.

• These tools vary between different IDEs, but commonly include a source code editor, compiler, and debugger.
Source Code Versus Machine Code

- **Source code** is designed for human readability and uses textual syntax that is translated into machine code

- **Machine code** is low-level binary data written for a computer that does not need additional translation
Source Code Editor

- A text editor designed specifically for editing code

- Features:
  - Syntax highlighting and brace matching
  - Automatic indentation
  - Auto-complete word prediction that fills in common words or phrases as the programmer is typing

```c
void loop() {
  // run repeatedly
}
```

- Brace matching: Clicking beside a brace puts a box around its counterpart (orange arrows)
- Syntax highlighting groups elements by color; `void` is a data type and `loop` is a function (red box)
Compiler

• Used to convert one language into another language

• Converts the source code into machine code for the computer to read
Debugger

• Software designed for testing the source code

• Oftentimes, the debugger will offer suggestions based on expectations to help the programmer identify and resolve issues
Common IDEs

- Arduino
- Visual Studio
- Eclipse
- Komodo
- Android Studio
- NetBeans
- Atom
- BlueJ
The Arduino IDE

- Arduino software is open source and can be downloaded for free at [https://www.arduino.cc/en/Main/Software](https://www.arduino.cc/en/Main/Software)
- Compatible with Windows, Mac OS X and Linux systems
Choosing the Correct Package

• Select download link for the appropriate operating system

Download the Arduino IDE

**ARDUINO 1.8.8**

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the **Getting Started** page for Installation instructions.
Installation

• Follow on-screen instructions to run the installation software (example is for Windows)

1. 

2. 

3. 

LSU rev20AUG2020 L07.02
Opening the IDE

- Locate the Arduino icon on your desktop and double click to open the program

Clicking this should give you this.
Navigation

• The toolbar at the top of the IDE contains functions for file, edit, sketch, tools, and help

• The next line gives shortcuts for verify, upload, new, open, and save

• The tab at the bottom shows the name of the current sketch
The Serial Monitor

• The icon of the magnifying glass in the top right-hand corner opens the **serial monitor**

• This is a pop-up window that allows the programmer to see interactions as the code runs
File

• The file folder contains features such as new, open, save, and print

• The sketchbook subfolder contains a collection of code written by the programmer

• The examples subfolder contains fully functional code written for the programmer to assist with common tasks
File: Examples

- The **examples** subfolder covers a wide range of processes
File: Preferences

- The preferences subfolder allows you to customize the IDE
- Display line numbers is a popular feature that adds or removes line numbers from the source code
**Edit**

- **Edit** provides shortcuts for useful features such as undo, redo, cut, copy, and paste.

- You can increase or decrease indentations as well as font size.

- A simple shortcut lets you comment or uncomment an entire section of highlighted code.
Sketch

- **Sketch** provides shortcuts for verifying, compiling, and uploading code
- It allows you to include libraries or incorporate files into the code as needed
Verifying and Uploading Code

- **Verifying** checks for problems
- **Uploading** verifies and sends the code to a microcontroller
- It is recommended to verify often as it is easier to locate mistakes from a short line of code as opposed to an entire script
Libraries

- Libraries are a collection of precompiled modules that use keywords to activate functions.
- The example below uses a library named pitches.h for its preset melody, duration, and notes without the need for additional code.

```c
#include "pitches.h"

int melody[] = {NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4};

int noteDurations[] = {4, 8, 8, 4, 4, 4, 4};

tone(8, melody[thisNote], noteDuration);
```
How to Add a Library

• Select **Manage Libraries** subfolder

![Manage Libraries Subfolder Image]

• Type **keyword** to locate a library and select install to download

![Library Manager Image]

• Locate and select new library in **include library** subfolder

![Contributed Libraries Image]
• **Tools** is where the board and processor are selected
  - Code will not run properly if this does not match your equipment

• Use ‘**Get Board Info**’ if you are unsure about the microcontroller being used
Tools: Port

• The serial port number is determined by the microprocessor

• The following link contains instructions for determining the correct port for Windows, Mac, and Linux:

http://www.me.umn.edu/courses/me2011/arduino/technotes/debug/arduinodebug.html
• If ever stuck, **help** offers links for getting started, frequently asked questions, and troubleshooting
Baud Rates

- The **baud rate** is the rate by which information is transferred.
- The serial monitor will display unintended caricature if the baud rate on the monitor is not set to match what is dictated in the code.

```cpp
Serial.begin(9600);
```
Debugging Code

• The debug window provides information based on normal expectations

• The example shows two conflicting data types assigned to one variable, int and word

• The debugger highlights the potential error in the text window (top) and alerts the programmer in the debug window (bottom)
Getting Started

- Arduino breaks the sketch into two parts, `void setup()` and `void loop()`
- Setup runs once whereas loop runs repeatedly
Global Declarations

- Libraries and global variables go outside of the two main functions (setup and loop) and are visible to every line in the code.
- The following example defines the **libraries** needed to initiate an SD card reader on lines 2 and 3.
- It defines **global variables** on lines 5, 6, and 7.

```cpp
#include<Wire.h>
#include<SD.h>

const int chipSelect = 10;
int error = 0;
long timeStamp;

void setup() {
  Serial.begin(9600);
}
Void Setup

- Functions that only need to run once go under void setup
- Can declare baud rate and initialize system checks
- In the example, the SD card writes via chipSelect (pin 10 defined on slide 28). If it fails to write, it prints an error message in the serial monitor (at a rate of 9600) and stops running the code

```c
void setup() {
  Serial.begin(9600);
  Serial.println("Initializing SD card...");
  pinMode(chipSelect, OUTPUT);
  if (!SD.begin(chipSelect)){
    Serial.println("Card failed, or not present");
    error = 1;
    return;
  }
}
```
Void Loop

- Location of the main code that will loop repeatedly
- In the example, the variable dataString will store the reading from a sensor (sensorVal) with 4 measurements taken and it will create a timeStamp that repeats once every millisecond (millis); this will repeat indefinitely

```java
void loop() {
  String dataString = "";
  timeStamp = millis();

  for (int analogPin = 0; analogPin < 4; analogPin++)
  {
    int sensorVal = analogRead(analogPin);
    dataString += String(sensorVal);
  }
}
```
Troubleshooting

• Check syntax
  – Missing semicolons, brackets, etc.

• Libraries
  – Proper syntax and keywords; some libraries may conflict with one another

• Correct baud rate

• Case sensitive context

• Correct data types
References

• https://www.freeiconspng.com/img/12780
• https://www.iconfinder.com/icons/37037/apple_face_finder_mac_os_x_mettalic_icon
• https://www.flaticon.com/free-icon/linux_518713