


Setting up your Software

In this reading, we will walk through setting up the software you'll need for this course, the Arduino integrated development environment (IDE). We will be using the Arduino IDE to program your microcontroller.



Screencast of Brian
walking through this
section goes here
on the edX course

Introduction

What is Arduino?

Arduino [describes itself](#) as “an open-source electronics platform based on easy-to-use hardware and software.” In large part, this description is fitting, as the company designs and sells a collection of microcontroller development boards that simplify deployment of embedded hardware alongside a software framework that abstracts away all but the most relevant considerations for your application. Perhaps what's under-represented by this description is the role that the surrounding community plays in enabling many plug-and-play experiences given the number and quality of auxiliary hardware modules, support libraries, and tutorials that have and continue to be produced within Arduino's ecosystem.

One thing we'd like to acknowledge here is that Arduino's mission of creating easy-to-use hardware and software has the necessary tradeoff of limiting the feature set of its development environment to the essentials.

What is an Integrated Development Environment (IDE)?

As is true for all forms of programming, an integrated development environment, or IDE, is an application with a feature set that facilitates software development generally or within a particular niche. The Arduino Desktop IDE is highly specific in the sense that it is intended to facilitate software development for a specific set of microcontroller boards, in C++.

Within the niche of IDEs for embedded software, there is a noticeable dichotomy between light-weight applications, (like Arduino's IDE) that minimize functionality and abstract away details in the name of simplicity and full-featured IDEs (like [Keil µVision](#) and [Visual Studio Code](#)) that exist to support industry professionals.

If you're interested in finding a happy medium to use in future embedded projects, our staff have had good experience using [VS Code](#) with the hardware specific extension [PlatformIO](#), that together enable nice-to-have features like line completion, reference tracking, et cetera, without all of the complexity that more advanced IDEs introduce. Having said this, this particular combination of VS Code + PlatformIO will not be officially supported within this course.

What is the Arduino IDE?

As you might expect given the description above, the Arduino IDE is a lightweight development environment with features that permit you to very quickly manipulate microcontroller development boards. While there is a cloud-based offering (the Arduino [Create Web Editor](#)) as well as a so-called [Arduino Pro IDE](#) that is in development (specifically in the alpha stage at the time of writing), we are going to use the standard [Arduino Desktop IDE](#) in this course.

Downloading and Installing the Arduino IDE

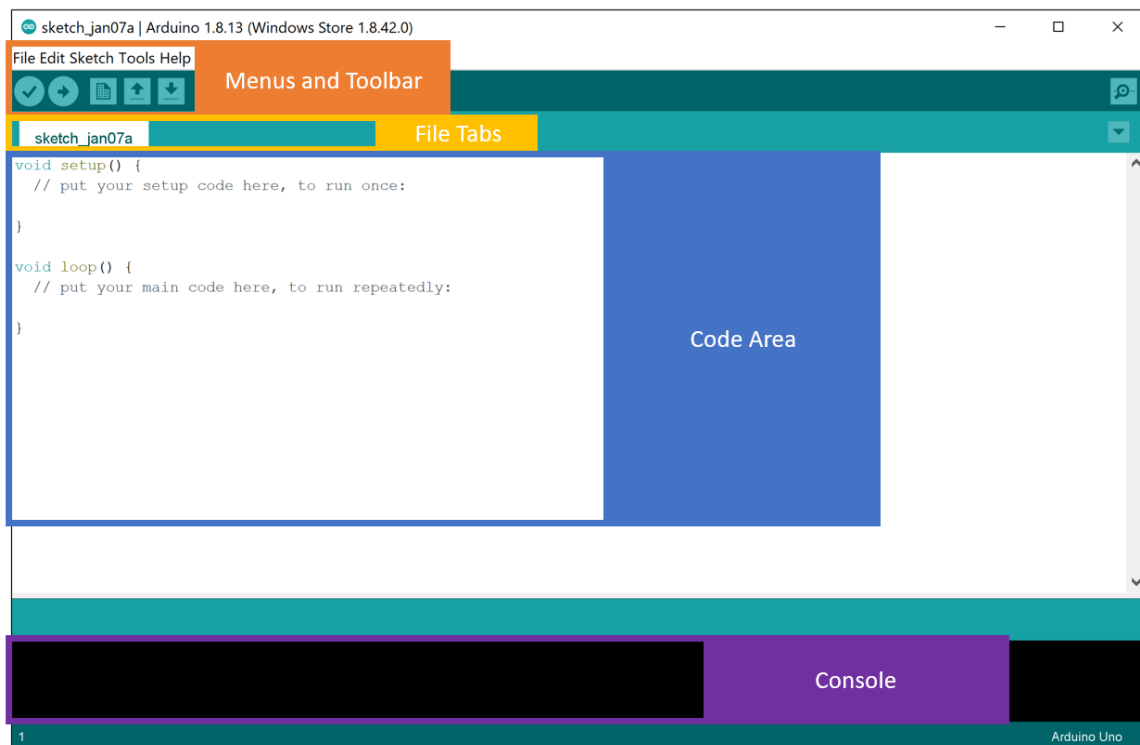
Note: Our staff will only officially support the latest release of the Arduino Desktop IDE (v1.8.13). If you have an older version of the Desktop IDE, we recommend that you update your software.

1. Navigate to arduino.cc and at the top of the page select Software and click Downloads
2. Click on the download link appropriate for your machine
3. There is no obligation to contribute, you can click 'Just Download' to proceed

Operating specific installation instructions vary, so if what follows isn't self evident, you can find guidance, by OS, at the following links:

- [Windows](#)
- [Mac OS X](#)
- [Linux](#)

A Quick Tour of the IDE



When you first open the IDE you will see a screen that looks something like the above screenshot.

Up at the top of the IDE you will find the menus and the toolbar which we will explore as we work through the rest of this document and when we run the tests later in this section.

Below that you will find the file tabs. For now there will only be one file open that is most likely named `sketch_<date>`, however later in the course when we work through more complicated examples you will see all of the various files that make up the project across the file tab area. In “Arduino speak,” a sketch is a simple project/application.

In the middle of the screen you will see the large code area. Each sketch can consist of many files and use many libraries but at its core each sketch is made of two main functions, “setup” and “loop”, which you can see pre-populated in the code area. We’ll explore what those functions are used for through the Blink example in a future video. For now, let’s continue orienting ourselves with and setting up the IDE.

Finally, at the bottom of the screen you’ll find the console. This is where you will see debug and error messages that result from compiling your C++ sketch and uploading it to your Arduino.

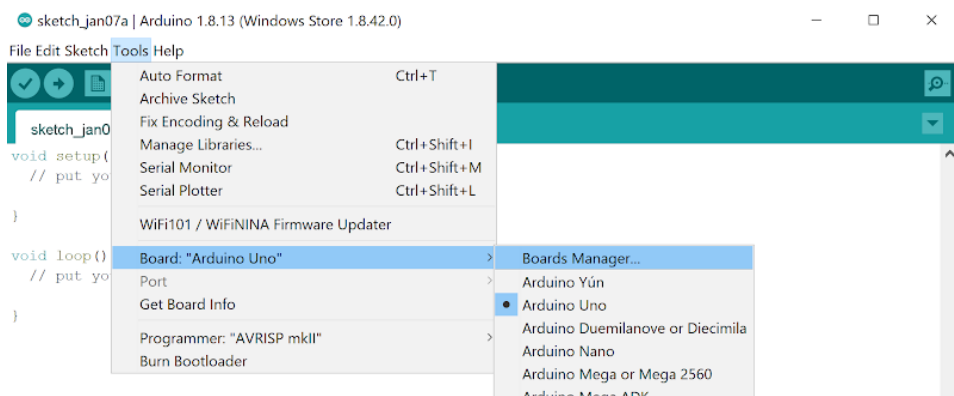
Installing the Board Files for the Nano 33 BLE Sense

Screencast of Brian walking through this section goes here on the edX course

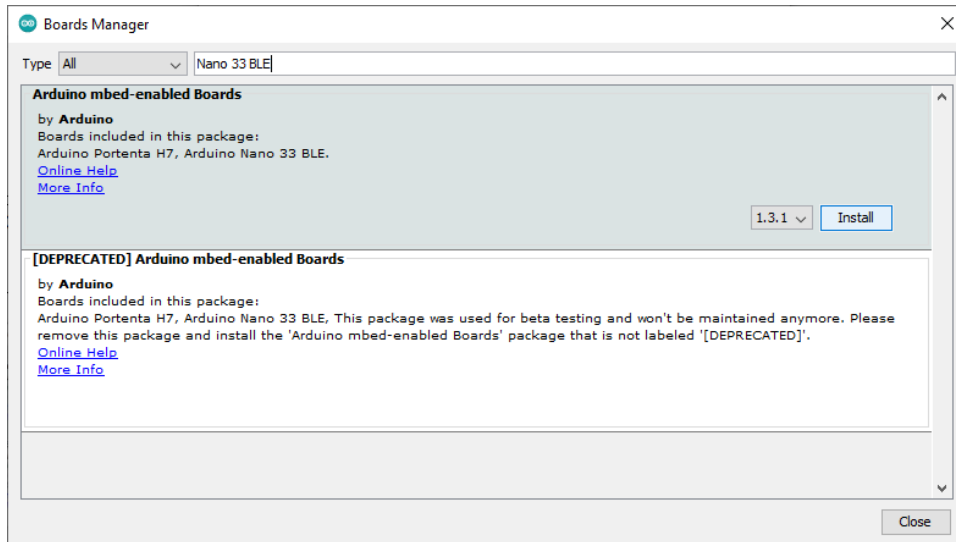
One of the primary advantages that the Arduino ecosystem affords is the portability of code you write for one or another board within their line-up or even in porting code to affiliate boards. This is made possible by the support files organized in the Boards Manager, which coordinates a download and installation of files that detail the Arduino functions (sometimes 'core') that are defined for that particular board (which is how hardware differences between boards are abstracted) as well as compiler or linker details specific to the given board.

To install the board files you will need for your Arduino Nano 33 BLE Sense please do the following:

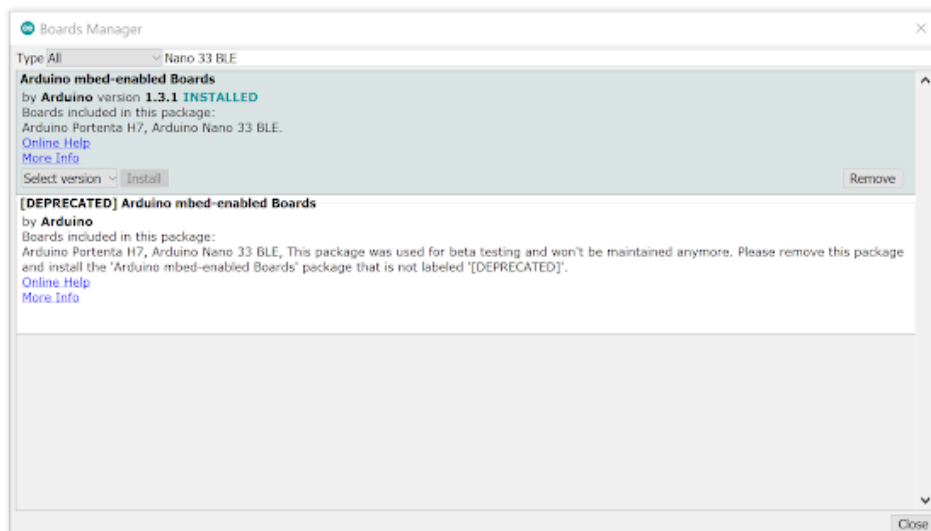
1. Open the Boards Manager, which you can find via the Tools drop-down menu. Navigate, as follows: **Tools** → **Board** → **Boards Manager**. Note that the Board may be set to "Arduino UNO" by default



2. In the Boards Manager dialog box, use the search bar at the top right to search for “Nano 33 BLE,” which should bring up two results. We’re interested in the first result (as shown), named “Arduino mbed-enabled Boards,” (without the DEPRECATED tag). Make sure **Version 1.3.1** is selected and then click “Install.” As the install process progresses you will see a blue completion bar work its way across the bottom of the Board Manager window. Be patient, you may need to install USB drivers which requires you to approve an administrator privileges popup which can take a couple minutes to appear.



After you have successfully installed the board if you exit and re-open the Board Manager and search again for “Nano 33 BLE” you will now see a green **INSTALLED** next to the library and the option to “Remove” the library or install a different version.



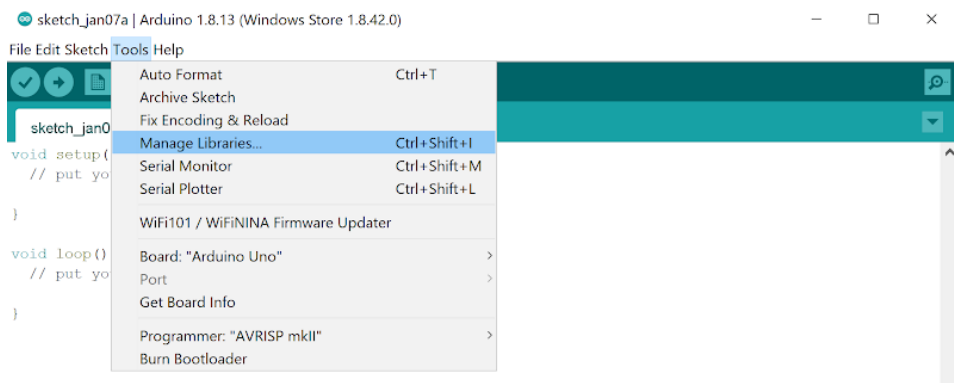
Installing the Libraries needed for this Course

Screenecast of Brian walking through this section goes here on the edX course

Another advantage of the Arduino ecosystem is the availability of a wide array of libraries for performing various tasks, such as interfacing with a sensor module or manipulating data using common algorithms. There are many libraries that can be accessed from within the Library Manager in the Arduino IDE as described below. Check [here](#) for a complete list.

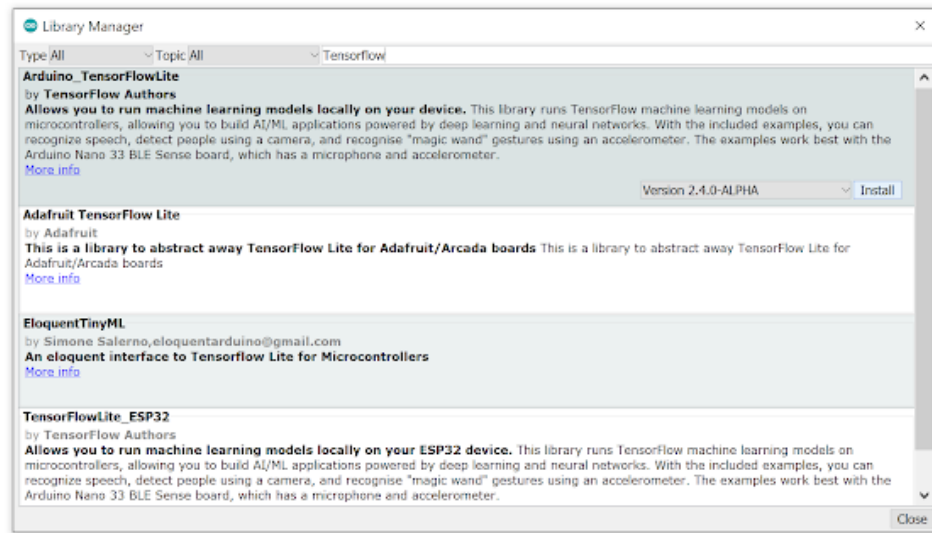
For this course we are going to need four libraries. To install the libraries please do the following and **make sure to install the version specified in the reading below or the tinyML applications will not work**:

1. Open the Library Manager, which you can find via the Tools drop-down menu. Navigate, as follows: **Tools** → **Manage Libraries**.

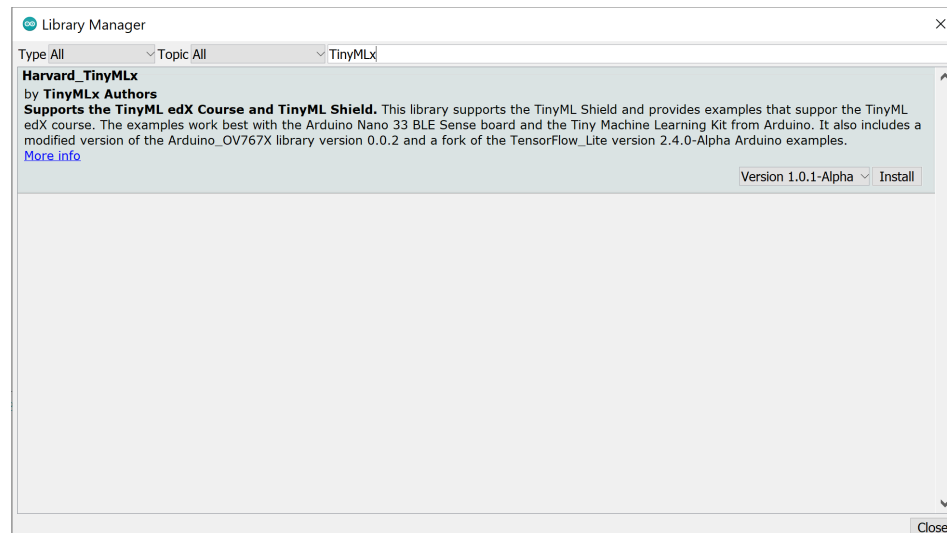


2. Then, much like for the Boards Manager, in the Library Manager dialog box, use the search bar at the top right to search for the following libraries, one at a time. Note that like with the Board manager a blue completion bar will appear across the bottom of the Library Manager window.

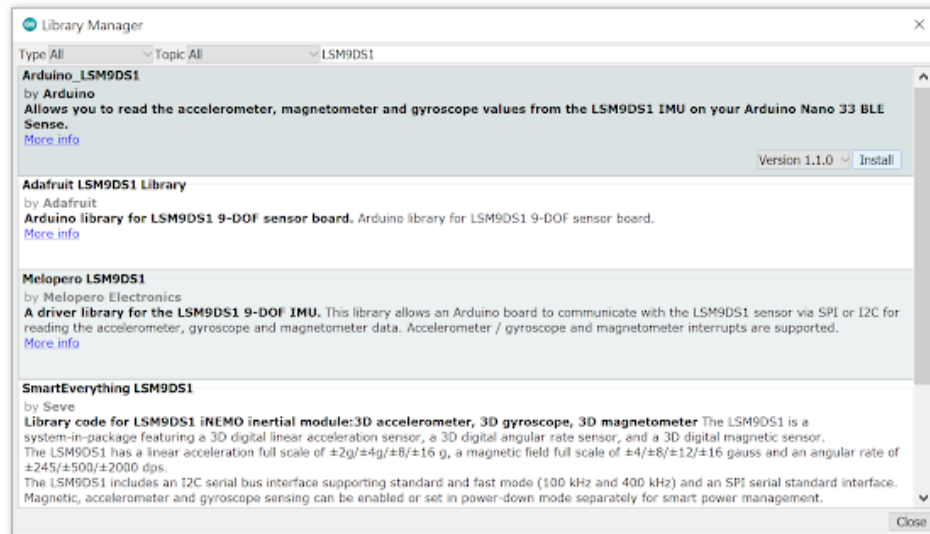
- a. The Tensorflow Lite Micro Library
 - i. Search Term: Tensorflow
 - ii. Library Name: Arduino_TensorFlowLite
 - iii. Version: 2.4.0-ALPHA



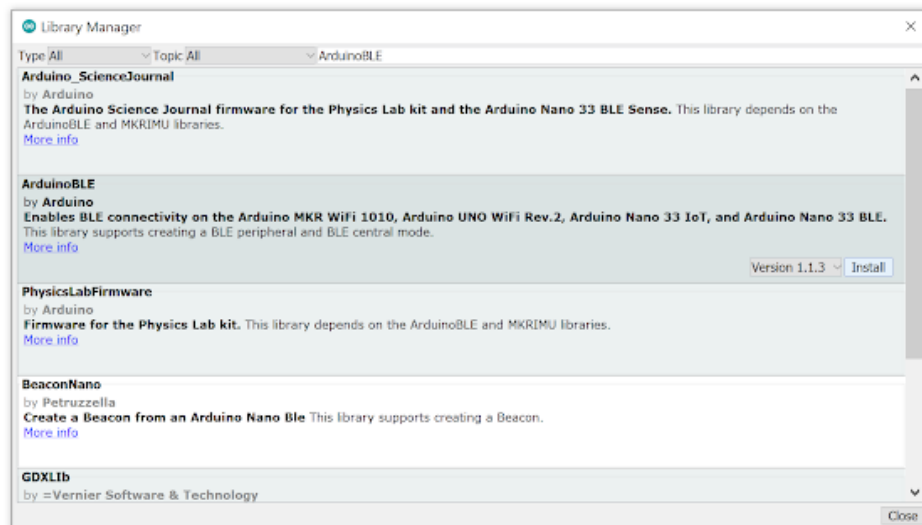
- b. The Harvard_TinyMLx Library we put together for this course!
 - i. Search Term: TinyMLx
 - ii. Library Name: Harvard_TinyMLx
 - iii. Version: 1.0.1



- c. The library that supports the accelerometer, magnetometer, and gyroscope on the Nano 33 BLE sense
- Search Term: LSM9DS1
 - Library Name: Arduino_LSM9DS1
 - Version: 1.1.0



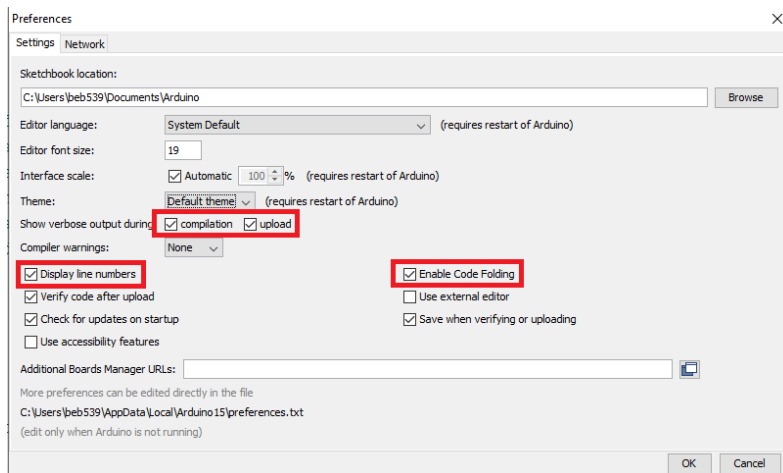
- d. ArduinoBLE
- Search Term: ArduinoBLE
 - Library Name: ArduinoBLE
 - Version: 1.1.3



Setting your Preferences

You can adjust the preferences set for the Arduino Desktop IDE via the File drop-down menu, [File → Preferences](#). There are a few preferences that we recommend enabling to make the Arduino IDE a little easier to use, namely:

1. Show verbose output during: compilation and upload
2. Enable code folding
3. Display line numbers



<Alt-text: Screenshot of the preferences window showing how to enable code folding, display line numbers, and show verbose output as described in the reading.>

Of final note if you don't like the default theme for the Arduino Desktop IDE there is a nice tutorial for a [dark theme you can find here](#). Also if you would like to learn more about the IDE, check out [Arduino's documentation](#).

And that's it! Your Arduino IDE should be all configured for this course. Now that you have all of the necessary board files and libraries installed it's time to explore more of the features of the IDE available under the "Tools" menu and start to test out your Arduino by deploying the Blink example!