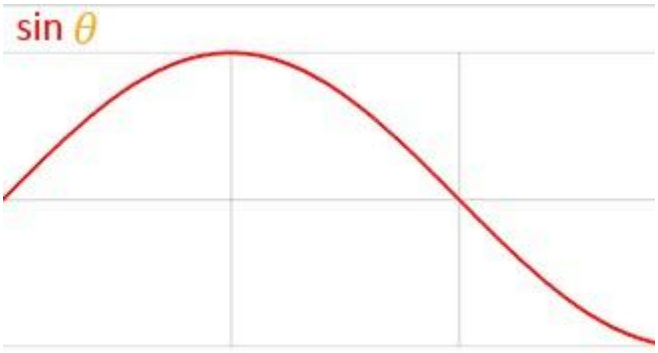
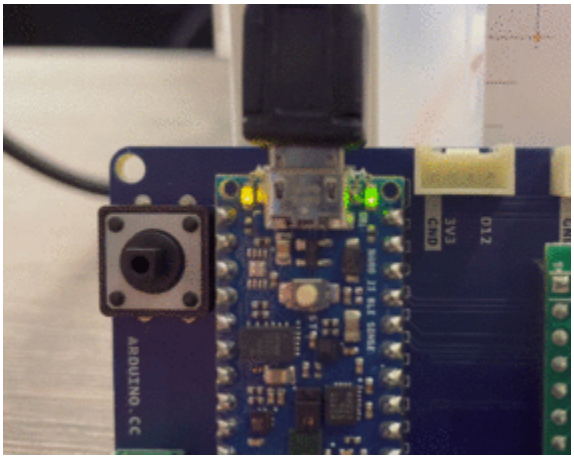
Testing the TFLM Installation

In this reading, we are going to walk through deploying the 'Hello World' example provided in the TensorFlow Lite for Microcontrollers (TFLM) library. The Hello World application runs a neural network model that can take a value, x, and predict its sine, y. Visually, the sine wave is a pleasant curve that runs smoothly from –1 to 1 and back. This makes it perfect for controlling a visually pleasing light show! We'll be using the output of our model to power a smoother blink of your Arduino LED.



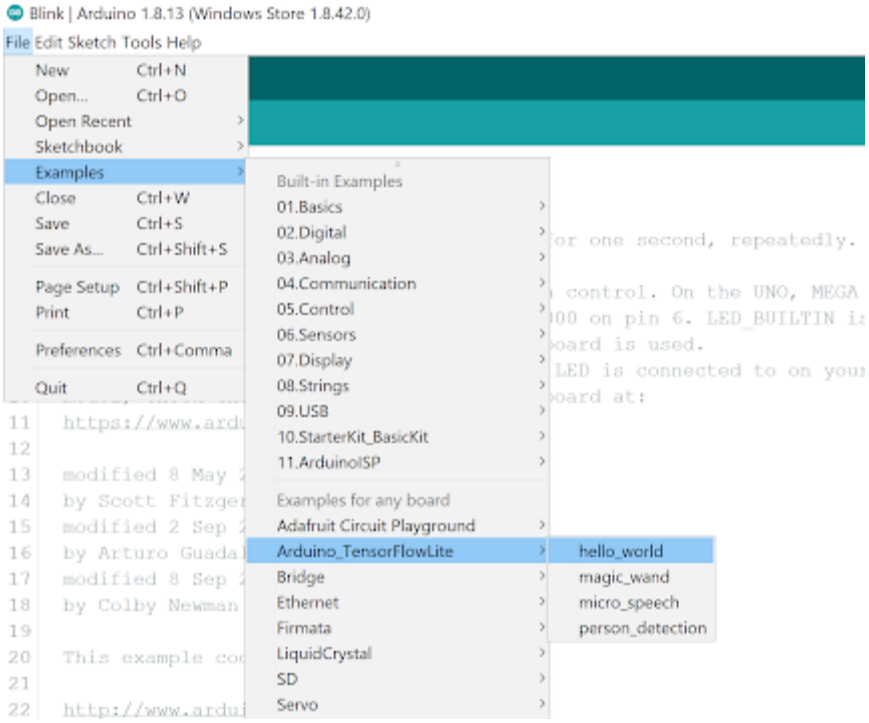


While the result won't differ greatly from what we've accomplished with the standard Arduino Blink sketch we just deployed (in either case, an LED will be turning on and off), in this instance, we are going to be deploying a model. Previously, we have learned about how neural networks can learn to model patterns they see in training data to go on to make predictions. Here, the model we deploy will map an input (x) to an output y = sin(x). While this is admittedly contrived, it will allow us to quickly evaluate if the TFLM stack is functional on the hardware in front of us by using this output y to control the intensity of light emitted by the on-board LED. As promised, below, you'll find instructions and screenshots and short videos walking you through the process!

Preparing for Deployment:

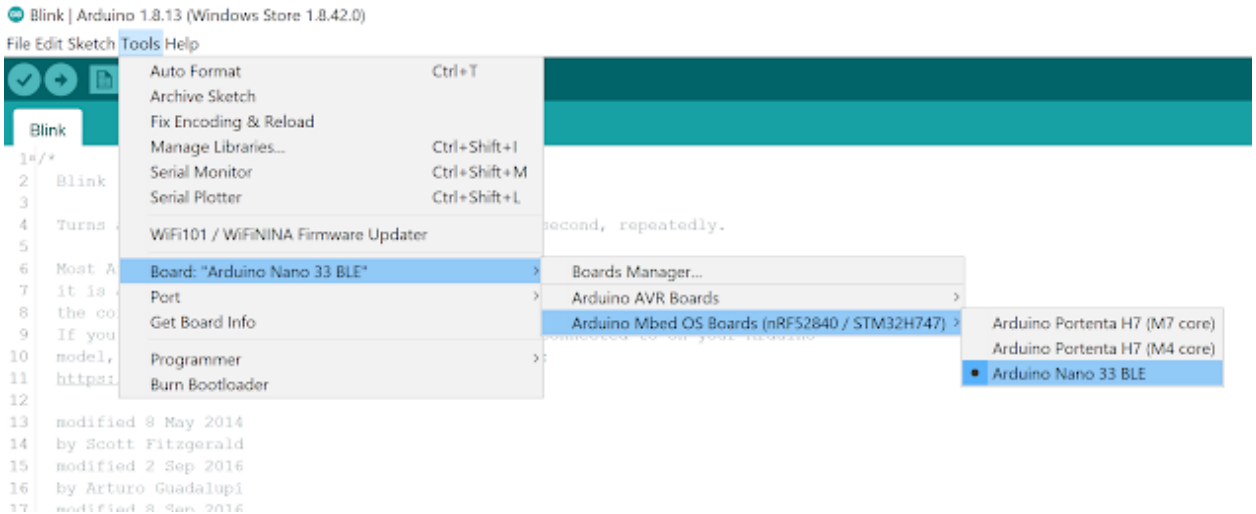
Use a USB cable to connect the Arduino Nano 33 BLE Sense to your machine.

Open the hello_world.ino sketch, which you can find via the File drop-down menu. Navigate, as follows: File → Examples → Arduino_TensorFlowLite → hello_world.



Just like with the Arduino Blink example (and as we will do for every deployment), use the Tools drop-down menu to select appropriate Port and Board.

Select the Arduino Nano 33 BLE as the board by going to Tools → Board: <Current Board Name> → Arduino Mbed OS Boards (nRF52840) → Arduino Nano 33 BLE. Note that on different operating systems the exact name of the board may vary but/and it should include the word Nano at a minimum. If you do not see that as an option, then please go back to Setting up the Software and make sure you have installed the necessary board files.

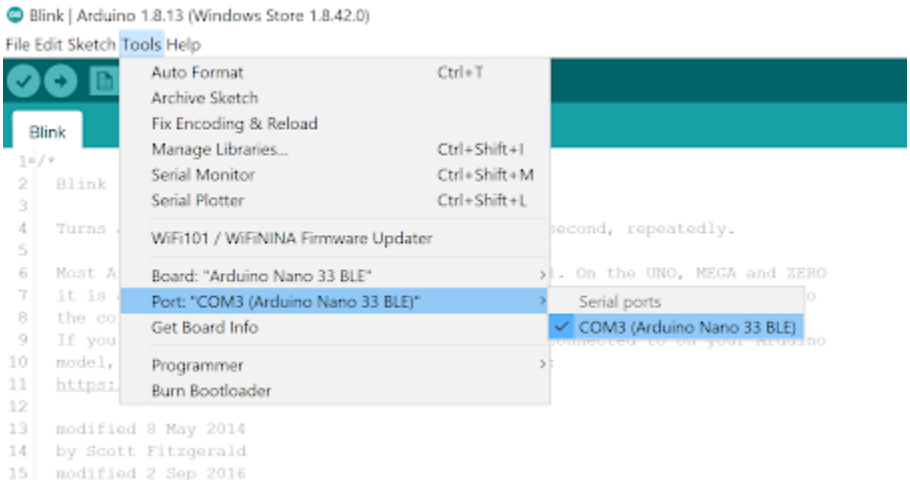


Then select the USB Port associated with your board. This will appear differently on Windows, macOS, Linux, but will likely indicate ‘Arduino Nano 33 BLE” in parenthesis. You can select this by going to Tools → Port: <Current Port (Board on Port)> → <TBD Based on OS> (Arduino Nano 33 BLE). Where <TBD Based on OS> is most likely to come from the list below where <#> indicates some integer number.

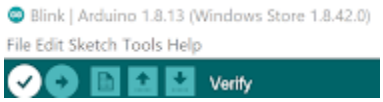
Windows → COM<#>

macOS → /dev/cu.usbmodem<#>

Linux → ttyUSB<#> or ttyACM<#>



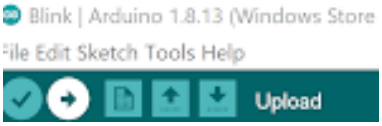
As always, it is best practice to then verify that the code is valid. Note that this may take a while the first time you run this as all of TFLM is being compiled (it will compile much faster in the future). Also, you may, or may not, see a series of compiling warnings (but not errors) as the code compiles, this is entirely normal either way and is the result of TFLM being bleeding edge software.



Deploying (Uploading) the Sketch

Once we know that the code at hand is valid, we can flash it to the MCU:

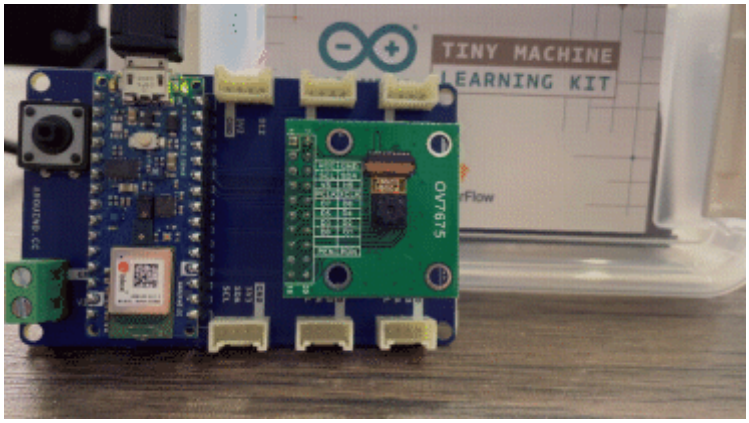
Use the rightward arrow next to the ‘compile’ checkmark to upload / flash the code.



You’ll know the upload is complete when you see red text in the console at the bottom of the IDE that shows 100% upload of the code and a statement that says something like “Done in <#.#> seconds.”

If you skipped the verification step, do note that this may take a while the first time you run this as all of TFLM is being compiled (it will compile much faster in the future). Also, you may, or may not, see a series of compiling warnings (but not errors) as the code compiles, this is entirely normal either way and is the result of TFLM being bleeding edge software.

At this point, you’ll want to look at the board itself. The orange LED opposite the green LED power indicator about the USB port should now be *fading* at the rate set by the sinusoidal model. Note that the default rate is quite fast so it should almost appear to be blinking very quickly. However, if you look closely you’ll notice that it is fading in and out vs. blinking.



Congratulations! At this point, if all has gone well, you can be sure that your embedded microcontroller is configured properly to run TensorFlow Micro.

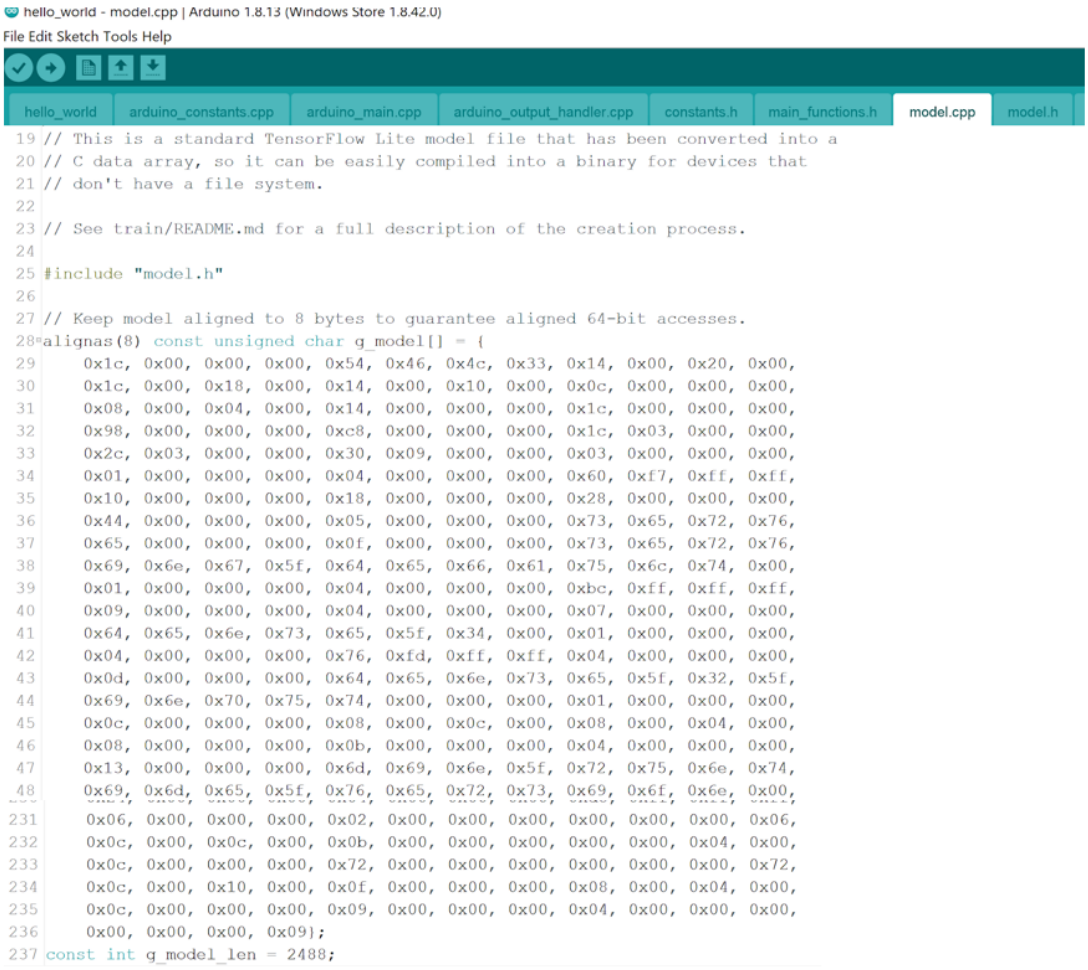
Understanding the TFLM Hello World Example

The first thing you will notice when looking at this example is that there is a lot more code in this TFLM version of blink! However, at a high level the code is structured the same. There is still a setup() function that runs once to initialize things and a loop() function that runs continuously to blink the LED.

As far as the high level differences go, in the main hello_world file, before either of those functions, there are a set of #include lines and global variable definitions. The #include lines are used to (in Python speak) import the various libraries we need for this example. In this case it is the TFLM library and some other helper functions. The global variables are defined outside of the setup()and loop() functions, so that there are references to variables that can be used in both the setup() function as well as in every loop of the loop() function. For example, the global tensor_arena memory is used to initialize and store our neural network once and then is used to execute the neural network every loop iteration. This is much more efficient than re-initializing the neural network in each loop iteration.

One other high level difference is that you'll notice that there are a series of tabs across the top of the IDE window. These tabs each represent different files that make up this hello_world project. Each holds some helper functions or definitions that our main hello_world file uses.

For example, the model.cpp file contains a large array which contains the binary data for the neural network model (in fact, it is simply the binary version of the TensorFlow Lite file) as well as an integer, which tells TFLM how long that array is. We'll explore how that binary file is generated later in the course!



Getting back to the main hello_world file, as mentioned above, the setup() function has a decent amount of code in it, but at a high level, it does exactly what you would expect, it initializes our neural network.

The loop() function also has a decent amount of code in it, but again at a high level, all it does is keep track of a counter over time for the x-value and uses the neural network to compute the approximate y-value of y = sin(x) and then set the brightness of the LED according to that approximate y value.

***