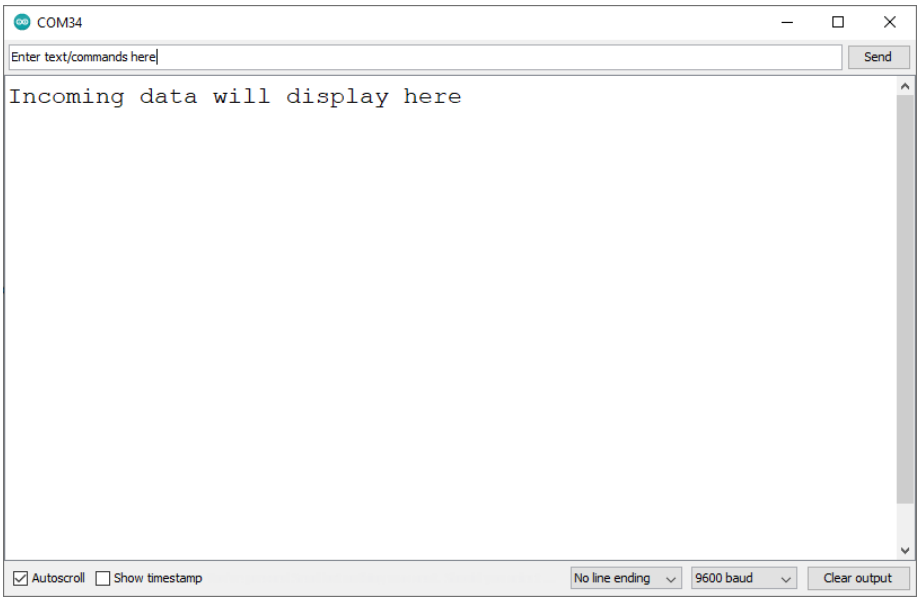
Testing the Sensors

With tests of the microprocessor itself and TFLM stack completed, we can turn our attention to verifying that the sensors required for this course are functioning properly. To do so, we'll be calling on a predefined script that the course staff have put together. By the end of this reading, you will have seen live data streams for the on-board microphone, the STMicroelectronics, MP34DT05, the on-board internal measurement unit (IMU), the STMicroelectronics, LSM9DS1, as well as at least a static image return from the off-board OV7675 camera module, the OmniVision OV7675. We will also detail the optional extensions you'll need to do to obtain a live video feed from the camera as well.

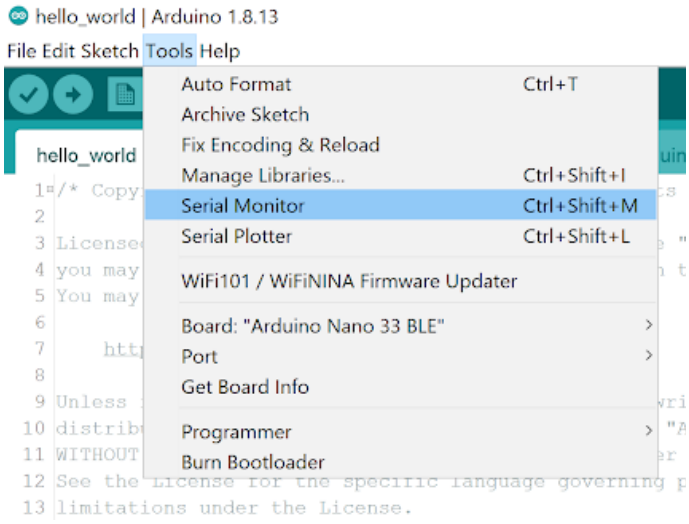
The Serial Monitor & Plotter

The Serial Monitor & Plotter are two core tools that you can use to get information from your Arduino when it is connected to your computer with a data USB cable. In fact, as you develop your own Arduino applications you'll probably find yourself opening up the Serial Monitor and using the Serial.println() command in your Arudino sketches much like how you have used the print() function in Python or printf() function in C++. Since the Arduino sketch runs in an infinite loop you may find that it may be easier to plot the graph of the data you are sending back from the Arduino instead of printing the raw values, and that is where the Serial Plotter will come in.

While the primary function of the Serial Monitor (and the Serial Plotter) is to view data incoming from the MCU, the Serial Monitor also features a text entry field that we will use to issue pre-determined commands conveyed in the console. Below we have included a screenshot of the Serial Monitor elements to differentiate where we expect to see incoming data and where we can enter commands to send to the microcontroller:



You can open up the Serial Monitor (and Serial Plotter) by navigating through the menu to `Tools → Serial Monitor` or `Tools → Serial Plotter`.



Testing the Microphone

1. Use a USB cable to connect the Arduino Nano 33 BLE Sense to your machine. You should see the green LED power indicator come on when the board first receives power.
2. Open the test_microphone.ino sketch, which you can find via the File drop-down menu. Navigate, as follows: `File → Examples → Harvard_TinyMLx → test_microphone`.
3. As always, use the Tools drop down menu to select appropriate Port and Board.

a. 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.

b. 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
   i. Windows → `COM<#>`
   ii. macOS → `/dev/cu.usbmodem<#>`
   iii. Linux → `ttyUSB<#>` or `ttyACM<#>`

4. Use the rightward arrow next to the 'compile' checkmark to upload / flash the code.

   You'll know the upload is complete when you 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."

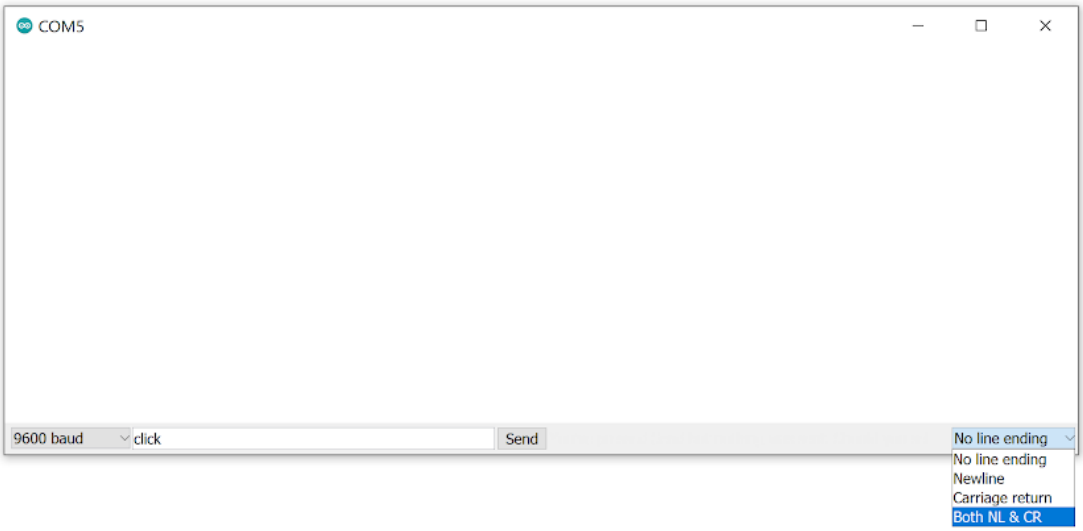
   If you receive an error you will see an orange error bar appear and a red error message in the console (as shown below). Don't worry -- there are many common reasons this may have occurred. To help you debug please check out our FAQ appendix with answers to the most common errors!

5. Open the Serial Monitor. You can accomplish this three different ways as shown below. **If the Serial Monitor fails to open check to make sure the appropriate Port is selected.** Sometimes the port is reset during the upload process.
   a. Use the menu to select: `Tools → Serial Monitor`
   b. Click on the magnifying glass at the top right of the Arduino Desktop IDE
   c. Use the keyboard shortcut: `CTRL + SHIFT + M` or `CMD + SHIFT + M` depending on your operating system.

6. When the Monitor opens, you should see the following text appear:

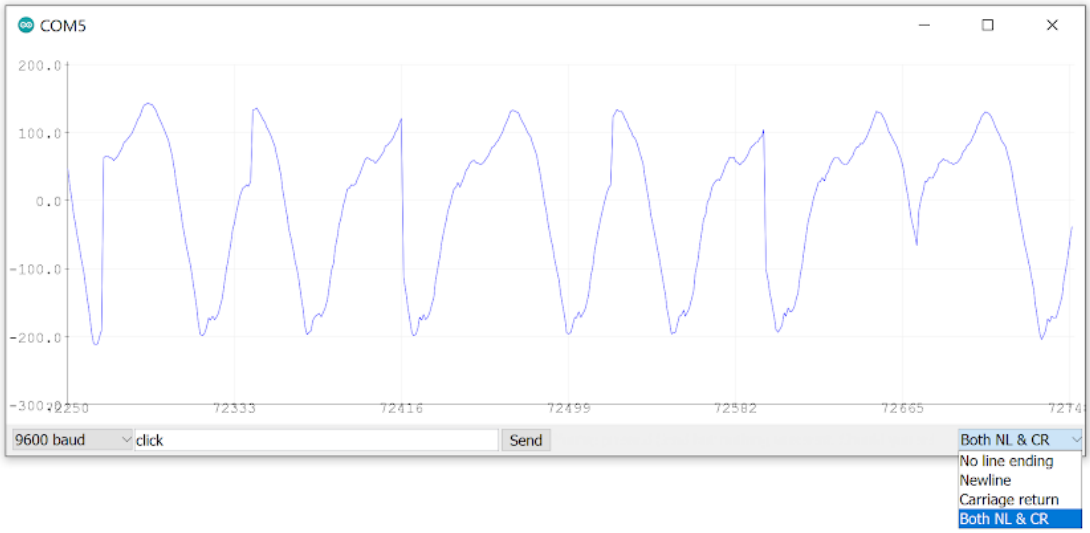
```
Welcome to the microphone test for the built-in microphone on the
Nano 33 BLE Sense

Use the on-shield button or send the command 'click' to start and
stop an audio recording
Open the Serial Plotter to view the corresponding waveform
```

7. Press the on-shield button to start an audio recording. After you do, a stream of data should appear in the Serial Monitor. Press the button again to stop the recording. If the numbers were changing then your microphone is most likely correctly recording audio! Congratulations! If you do not have the shield you can also control the starting and stopping of the waveform by sending the command `click` through the serial monitor. In the drop down menu that reads “No line ending” by default at the bottom right of the Serial Monitor, you need to select “**Both NL & CR**”. This tells the Serial Monitor to send both a NL = new line character as well as a CR = carriage return character every time you send a message. This is important for our application as our Arduino sketch is looking for that to differentiate between each input.

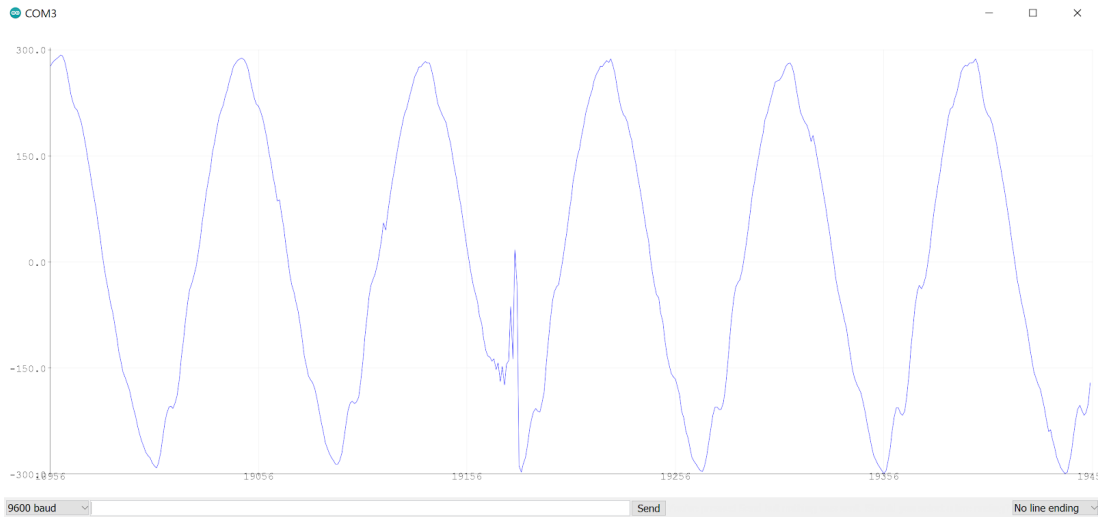


8. To help visualize this a little better we are going to use the Serial Plotter. **Note that you cannot have both the Serial Monitor and Serial Plotter open at the same time as the ARduino can only communicate over a single serial port. Importantly, this also means that you cannot upload new code to the Arduino if either the Serial Monitor or Plotter is open!** You can open the Serial Plotter in two ways:
    a. Use the menu to select: `Tools → Serial Plotter`
    b. Use the keyboard shortcut: `CTRL + SHIFT + L` or `CMD + SHIFT + L` depending on your operating system.

9. Perform the same experiment, press the on-shield button (or send the serial command `click`) to start an audio recording and again to stop it. You should see a waveform appear on the Serial Plotter. This is simply a graphical representation of the numbers you saw earlier on the Serial Monitor. Note that the vertical axis of the Serial Plotter scales dynamically so that the magnitude of the data conveyed is not fixed with respect to the enclosing window, but rather scaled to occupy most of the headroom.



10. Now record some audio again. This time try to whistle or hum a constant tone. If you can keep the tone constant you should see the waveform start to look like a sinusoid. An example the course staff made is below. Don't worry if you can't get this to work perfectly. You should in general find that a constant tone has a more constant pattern. If when you make different sounds you find that the waveform changes, you can be confident that your microphone is working.



Testing the Inertial Measurement Unit

1. Now let's open the test_IMU.ino sketch, which you can find via the File drop-down menu. Navigate, as follows: `File → Examples → Harvard_TinyMLx → test_IMU.`
2. As always, use the Tools drop down menu to select appropriate Port and Board.
3. Then use the rightward arrow next to the 'compile' checkmark to upload / flash the code. If you get the error "Arduino_LSM9DS1.h: No such file or directory" then please go back to Setting up the Software and make sure you install the accelerometer, magnetometry, and gyroscope library! To help you debug please check out our FAQ appendix with answers to the most common errors!
4. Open the Serial Monitor and you should see the following. As a reminder you can open the serial monitor in one of three ways shown below. **If the Serial Monitor fails to open check to make sure the appropriate Port is selected.** Sometimes the port is reset during the upload process.
    a. Use the menu to select: `Tools → Serial Monitor`
    b. Click on the magnifying glass at the top right of the Arduino Desktop IDE
    c. Use the keyboard shortcut: `CTRL + SHIFT + M` or `CMD + SHIFT + M` depending on your operating system.

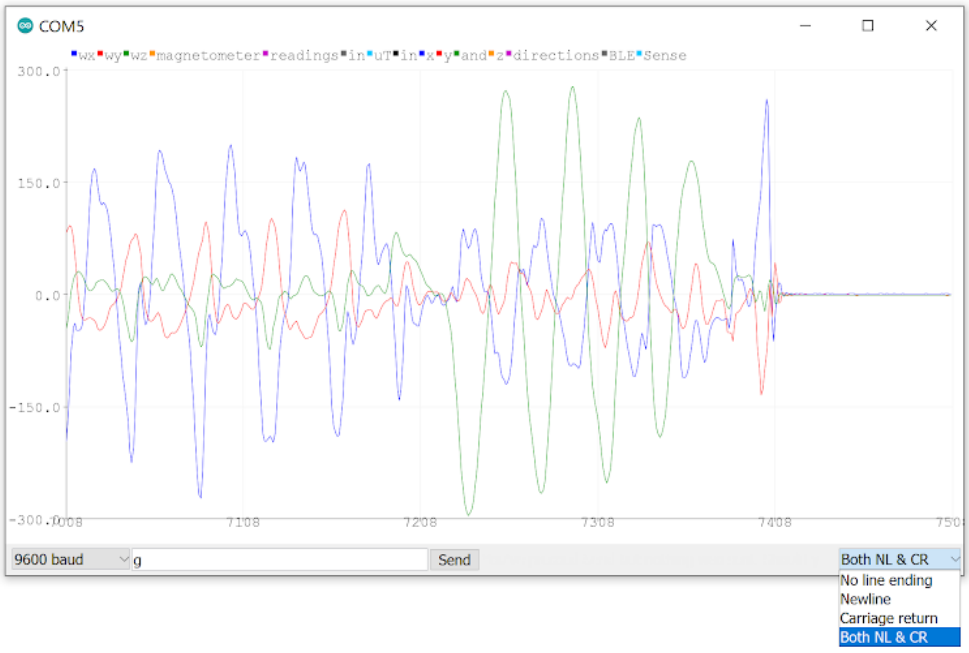
```
Welcome to the IMU test for the built-in IMU on the Nano 33 BLE Sense

Available commands:
a - display accelerometer readings in g's in x, y, and z directions
g - display gyroscope readings in deg/s in x, y, and z directions
m - display magnetometer readings in uT in x, y, and z directions
```

5. Like with the microphone, in the drop down menu that reads “No line ending” by default at the bottom right of the Serial Monitor, you need to select “**Both NL & CR**”.

6. Now you can enter one of the possible arguments (a, g, or m) into the text entry bar across the top of the Serial Monitor and click “Send”. As mentioned in the Serial Monitor this will start to print the data coming from the [a]ccelerometer (computing acceleration in the x, y, or z direction), the [g]yroscope (computing the angular velocity -- the change in the roll, pith, and yaw), or the [m]agnetometer (computing the magnetic forces present on the IMU). Depending on your selection, you should now see a stream of corresponding data. However like with the microphone this data is hard to interpret in raw form. So instead lets us the Serial Plotter.

7. Close the Serial Monitor and open the Serial Plotter. As a reminder you can open the Serial Plotter in two ways:
    a. Use the menu to select: `Tools → Serial Plotter`
    b. Use the keyboard shortcut: `CTRL + SHIFT + L` or `CMD + SHIFT + L` depending on your operating system.

8. With the Plotter open, you should see the same stream of data presented graphically. The most interesting one to plot is the [g]yrocope. Again make sure you have selected “Both NL & CR” and type “g” into the text entry box (now at the bottom of the window) and click “Send.”



9. Move the board around and observe the response. Can you figure out which axis of rotation is the x, the y, and the z (also known as roll, pitch, and yaw)? If by moving the board around in different directions you get different responses from the various lines in the Serial Plotter you can feel confident that you have a working IMU.

Testing the OV7675 Camera

1. Now let's open the test_camera.ino sketch, which you can find via the File drop-down menu. Navigate, as follows: `File → Examples → Harvard_TinyMLx → test_camera`.

2. As always, use the Tools drop down menu to select appropriate Port and Board. **Note:** if you did not purchase the Tiny Machine Learning Kit and sourced an OV7670, as compared to the OV7675 that comes with the kit, you will need to make one software changes as detailed at the end of this document.

3. Then use the rightward arrow next to the 'compile' checkmark to upload / flash the code. If you get the error "Arduino_OV767X.h: No such file or directory" then please go back to Setting up the Software and make sure you install the accelerometer, magnetometry, and gyroscope library! To help you debug please check out our FAQ appendix with answers to the most common errors! **Note:** if you get an error message that contains something like `'OV7675' was not declared in this scope` this means you already had a conflicting copy of the Arduino_OV767X library installed on your system. We have bundled a forked copy of it with our library so please uninstall the conflicting version.

4. Open up the Serial Monitor and you should see:

```
Welcome to the OV7675 test

Available commands:

single - take a single image and print out the hexadecimal for each
```

```
pixel (default)
live - the raw bytes of images will be streamed live over the serial
port
capture - when in single mode, initiates an image capture
```

5. Type “single” in the text entry field and press send or hit the enter / return key. The camera is now primed to take a picture. Now you can either text “capture” or press the on-shield button to take a selfie (or photo). To get the photo oriented correctly make sure the “OV7675” text on the camera module (or the “Tiny Machine Learning Shield” on the shield) is oriented such that it would be readable by the subject of the photo (e.g., you if you are taking a selfie). **The camera does not have a large field of view so if you are taking a selfie make sure to hold the camera far away from your face.**

6. To view your image you will need to copy the sequence of hexadecimal digits that will print to the Serial Monitor and paste them into this Google Colab that we created to display the image (link included below as well).

   https://colab.research.google.com/github/tinyMLx/colabs/blob/master/4-2-12-OV7675ImageViewer.ipynb

(Optional) Real Time Video with Processing

If you are interested in seeing a live stream capture from the OV7675, you can do that but not through the Arduino IDE. You’ll have to use an additional piece of software, Processing. You can download the application for your OS, here (we’ve tested with version 3.5.4). We would like to note that we have found this to be buggy on Windows, which is why we provide this as an optional extension and not the default suggested process.

1. Launch the `test_camera` example through the Arduino IDE, open the Serial Monitor, and then type “live” in the text entry field and press send or hit the enter / return key. Your camera will now be live streaming the image to the computer at 1 frame per second. **Make sure to then close the Serial Monitor to free up the serial port as we will be using it again later.**

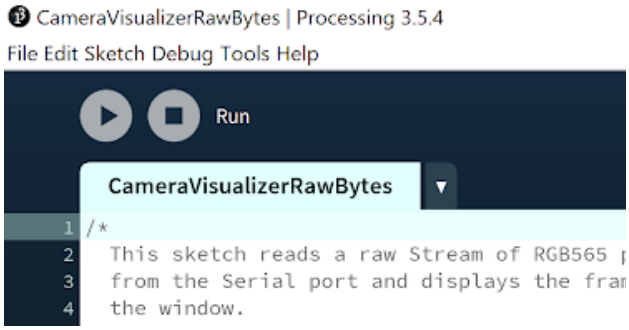
2. With Processing installed (which for some operating systems is as simple as unzipping the pre-built application folder), open the Processing Application. You’ll find that the application looks a lot like the Arduino Desktop IDE.

3. Then find your Arduino folder. On most operating systems it is located either in your `home` or `Documents` directory.

4. In Processing open the file `Arduino → Libraries → Harvard_TiynMLx → extras → CameraVisualizerRawBytes → CameraVisualizerRawBytes.pde`

5. Go to lines 34-36 and update them to be the name of the serial port on which your Arduino is currently connected.

6. Now click the “run” play button at the top left of the UI. A popup should appear streaming the camera image. Again, we have found this to be buggy on Windows so don’t be surprised if the image looks odd on Windows.



Changes for the OV7670

To use the OV7670 instead of the OV7675, simply change the fourth argument of the call to `Camera.begin()` on line 25 of the `test_camera` sketch, as highlighted below, from `OV7675` to `OV7670`. That’s it! The library will handle the rest!

```
// Initialize the OV7675 camera
void if(!Camera.begin(QCIF, RGB565, 1 OV7675)) {
   Serial.println(“Failed to initialize camera”);
   While (1);
}
```

Additional Resources

- If you’d like to learn more about debugging microcontrollers we’ve put together a short guide in this appendix document.

- If you’d like to learn more about powering your arduino using a battery we’ve put together another short guide in this appendix document.