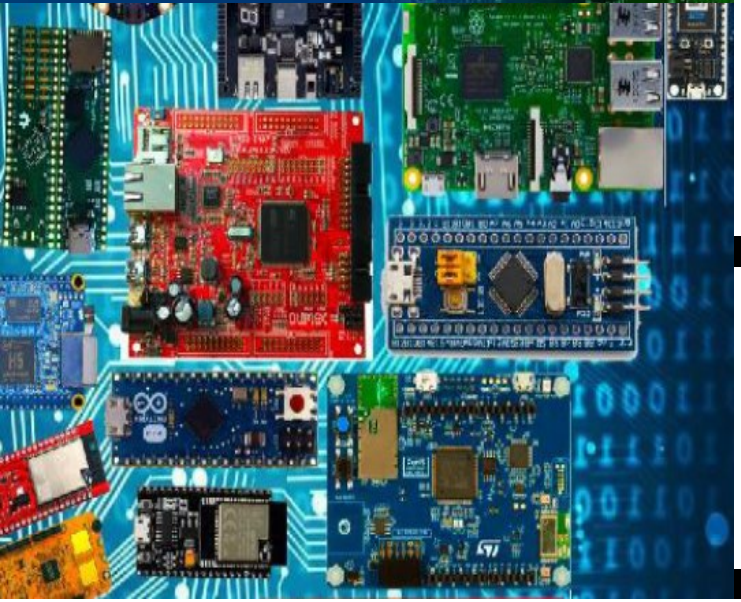


Embedded Systems

SETUP TESTING

Dennis A. N. Gookyi

CONTENTS

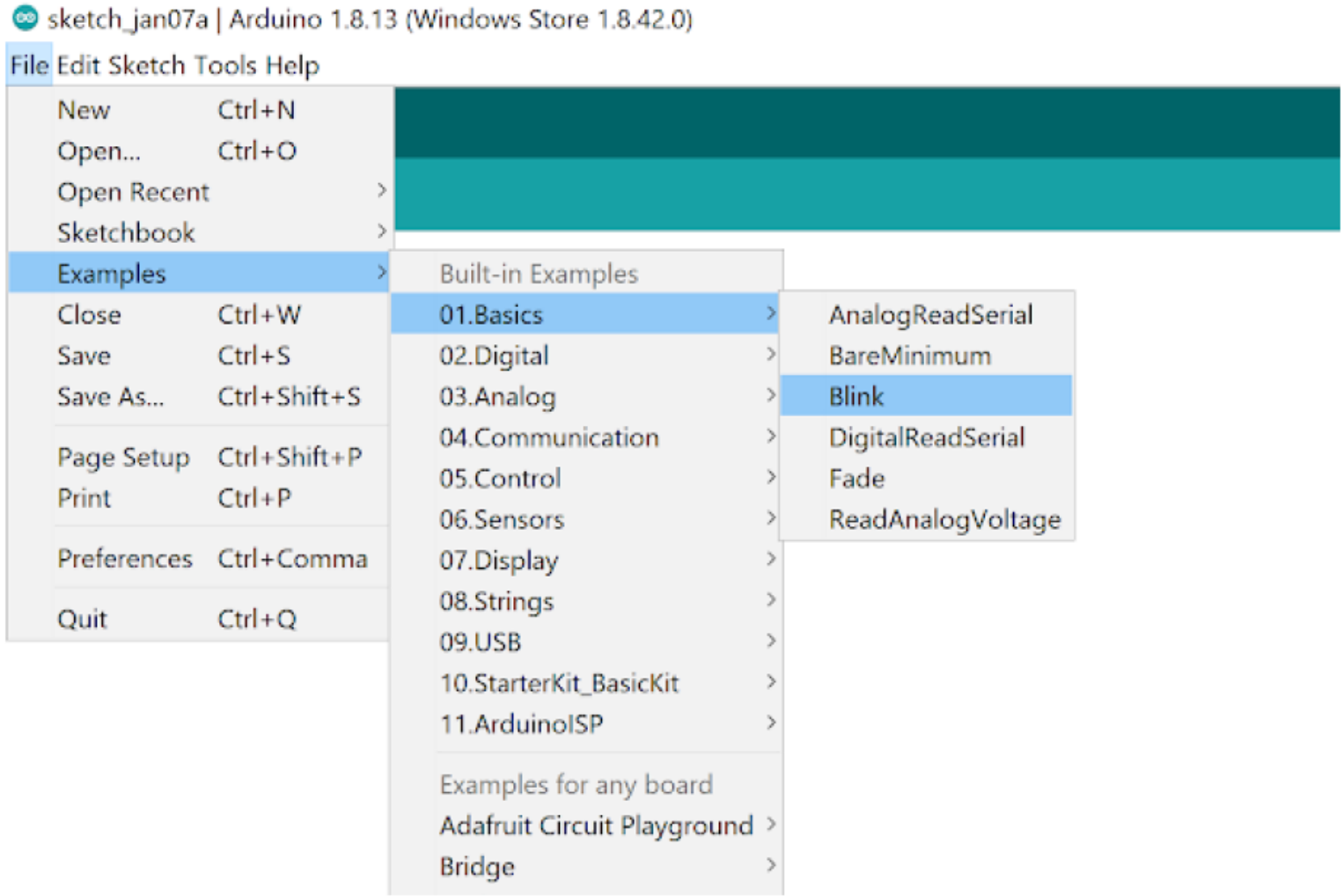
- Setup Testing

BLINK EXAMPLE

- Preparing for Deployment
  - Use a USB cable to connect the Arduino Nano 33 BLE Sense to your machine
  - You should see a green LED power indicator come on when the board first receives power
  - Open the Blink.ino sketch, which you can find via the File drop-down menu
  - Navigate as follows: File → Examples → 01.Basics → Blink
  - Arduino has provided a wealth of examples to choose from should you like to explore the board more on your own outside of the course material
  - There is great documentation about those examples on the Arduino website
  - https://docs.arduino.cc/built-in-examples/

BLINK EXAMPLE

- Preparing for Deployment

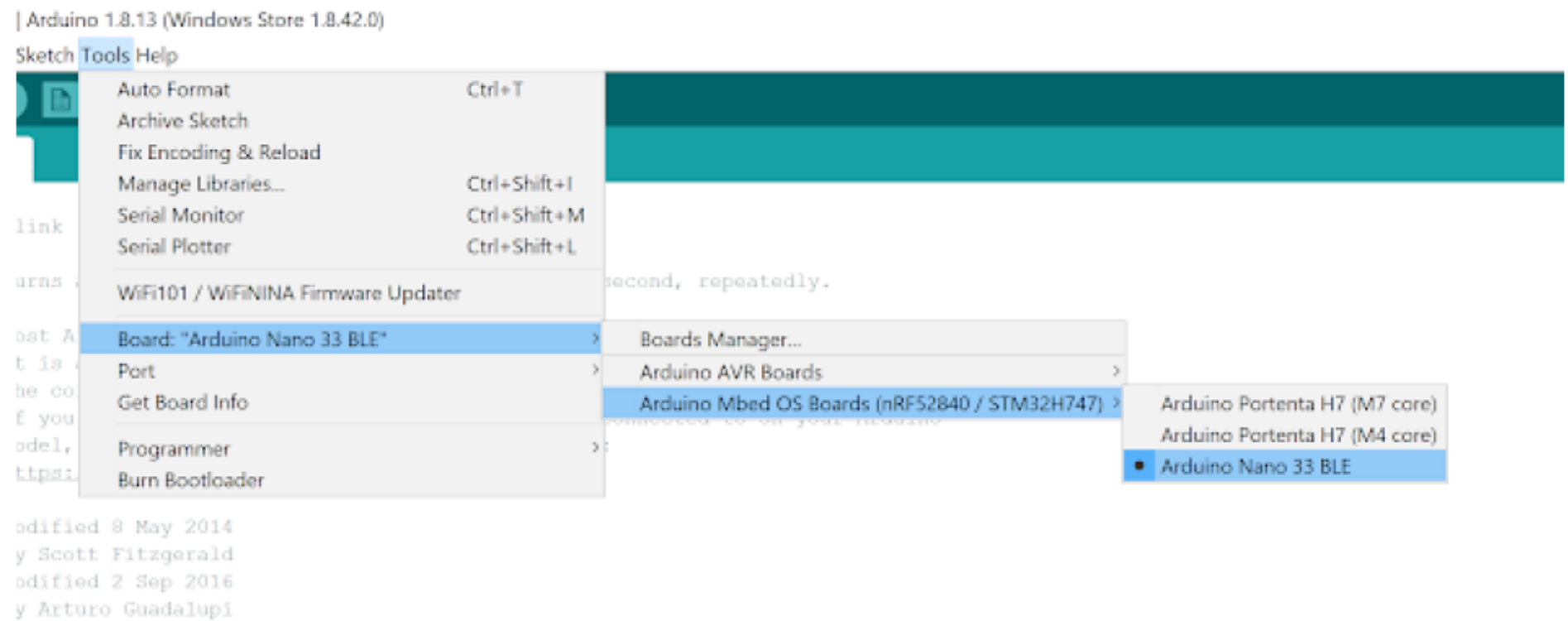


BLINK EXAMPLE

- Preparing for Deployment
  - Use the Tools drop-down menu to select the appropriate Port and Board
  - This is important as it is telling the IDE which board files to use and on which serial connection it should send the code
  - In some cases, this may happen automatically
  - 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

BLINK EXAMPLE

- Preparing for Deployment

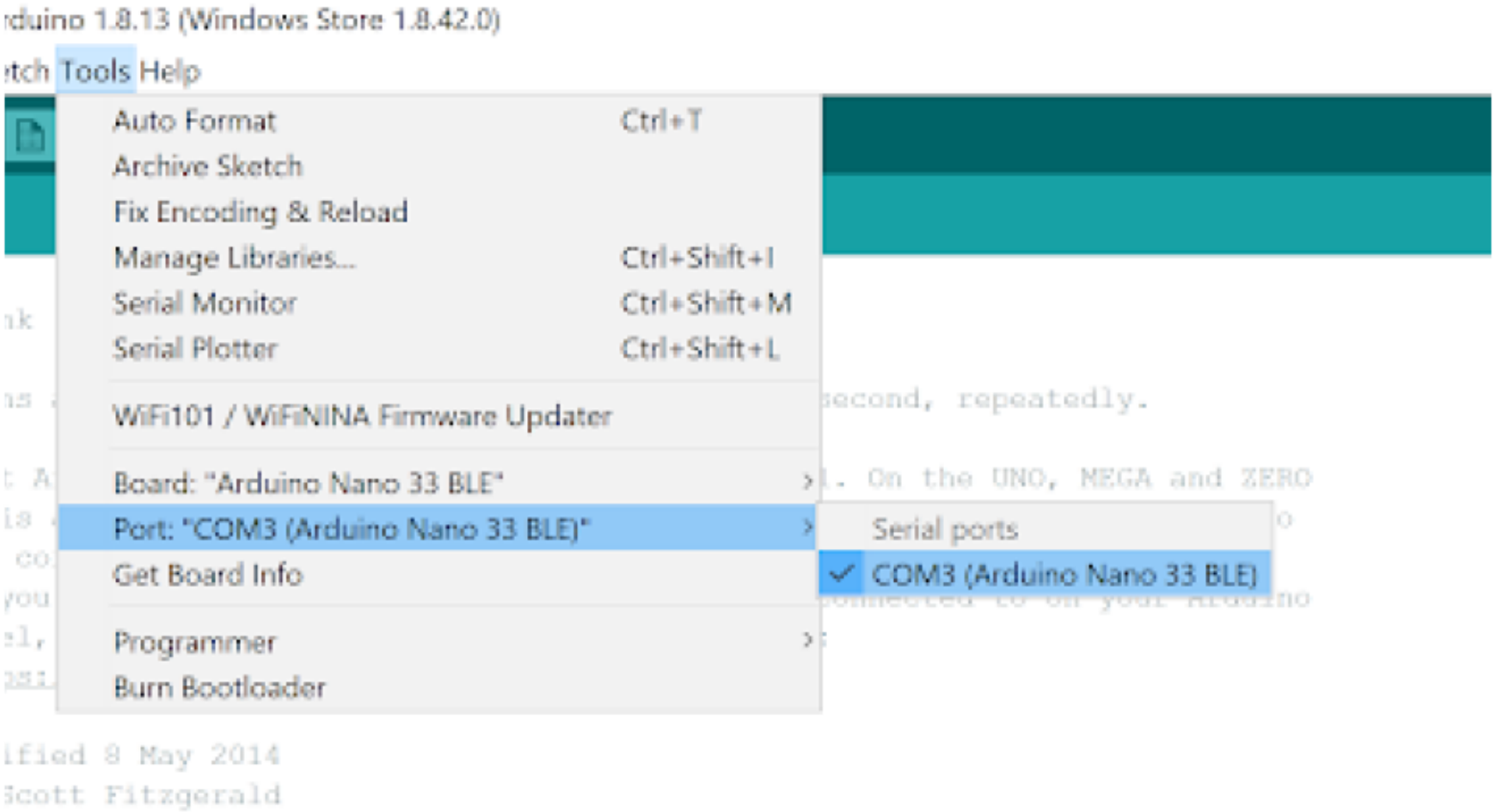


BLINK EXAMPLE

- Preparing for Deployment
  - 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<#>

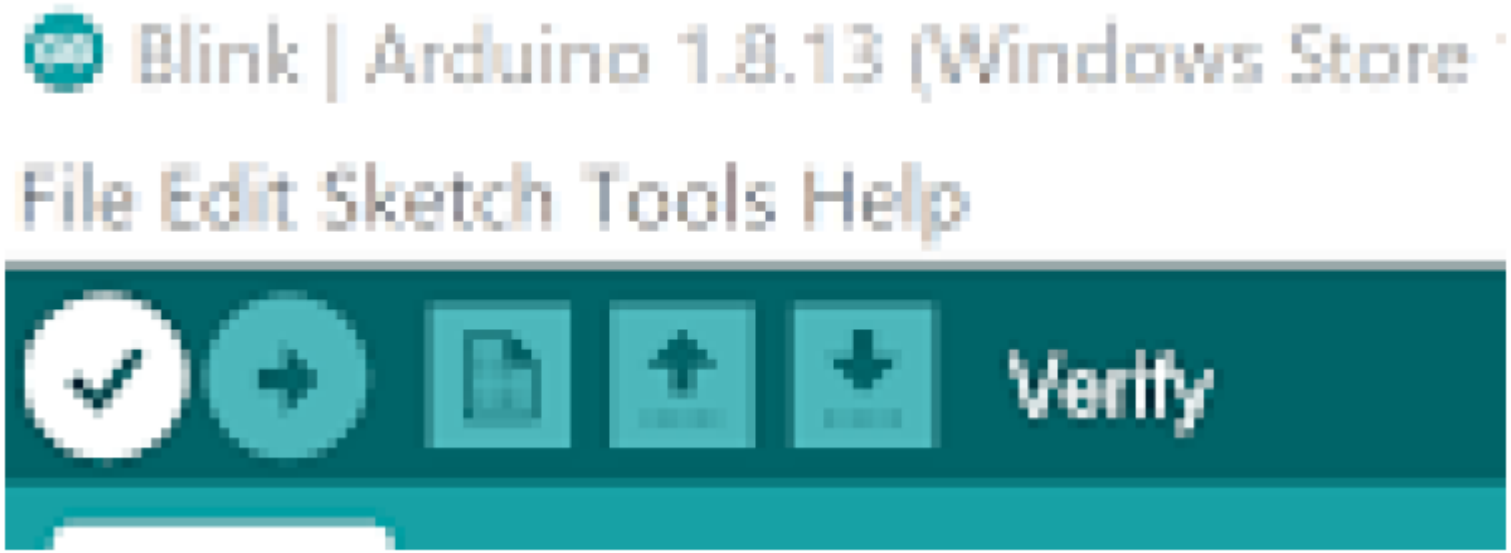
BLINK EXAMPLE

- Preparing for Deployment



BLINK EXAMPLE

- Preparing for Deployment
  - Finally, use the checkmark button at the top left of the UI to verify that the code within the example sketch is valid



BLINK EXAMPLE

- Preparing for Deployment
  - Verification will compile the code, so take note of the status and results indicated in the black console at the bottom of the IDE
  - The level of detail presented here will depend on whether or not you have enabled 'verbose output during compilation' in Preferences
  - You should most likely at the end see a final output indicating how much memory the sketch will take on the Arduino once it is uploaded
  - Something like: "Sketch uses 86568 bytes (8%) of program storage space, Maximum is 983040 bytes, Global variables use 44696 bytes (17%) of dynamic memory, leaving 217448 bytes for local variables, Maximum is
  - 262144 bytes"
  - As you can see this is a very simple example and does not take up much space
  - While, as you'll see in a moment, uploading your code will also verify your code automatically, it is often helpful to verify your code first as you can iron out any compilation errors without having any hardware on hand

BLINK EXAMPLE

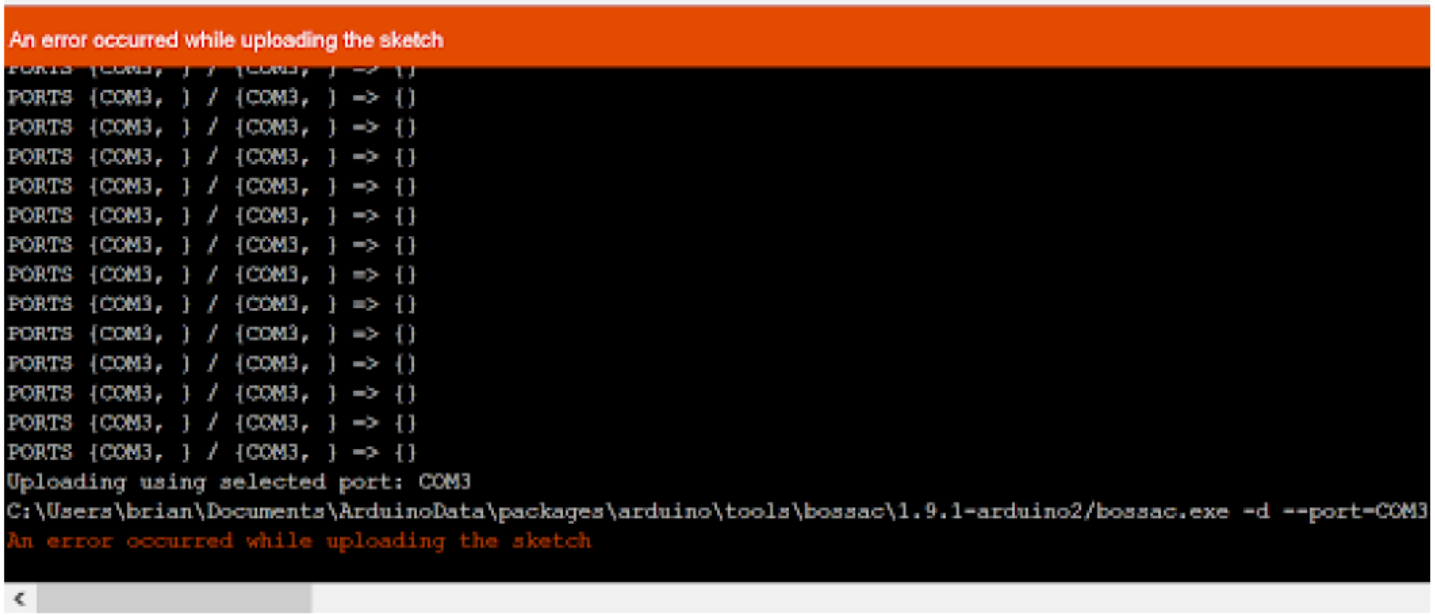
- Deploying (Uploading) the Sketch
  - Once we know that the code at hand is valid, we can 'flash' it to the MCU
  - Use the rightward arrow next to the 'compile' checkmark to upload/flash the code
  - Note that pragmatically, this step will re-compile the sketch before flashing the code, so that in the future if you intend to sequentially compile and flash a program, you need only press the 'upload' arrow

BLINK EXAMPLE

- Deploying (Uploading) the Sketch
  - As before, take note of the status and results indicated in the black console at the bottom of the IDE
  - The level of detail presented here will depend on whether or not you have enabled 'verbose output during compilation' in Preferences, accessible via the File drop-down menu in the IDE
  - You'll know the upload is complete when you see the 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 this is the first time you are uploading a sketch to an Arduino the upload may hang for a little while until you get another administrator approval popup and approve it

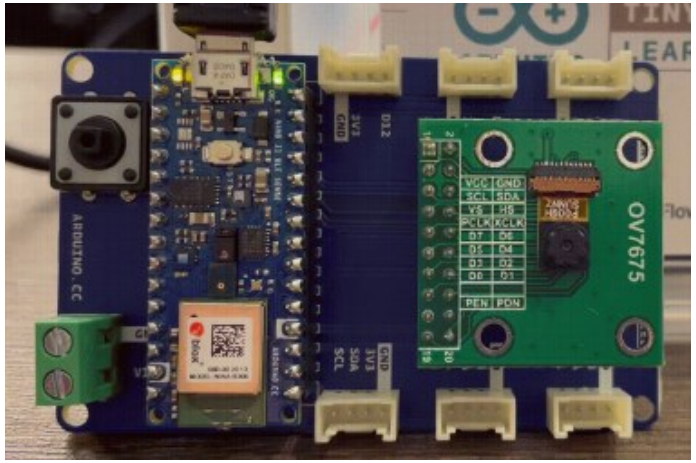
BLINK EXAMPLE

- Deploying (Uploading) the Sketch
  - 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
    - https://github.com/tinyMLx/appendix/blob/main/ArduinoFAQ.md



BLINK EXAMPLE

- Deploying (Uploading) the Sketch
  - At this point, you'll want to look to the board itself
  - The orange LED opposite the green LED power indicator about the USB port should now be blinking



  - As a final note if you'd like to learn more about how the process of flashing code to a microcontroller works please check out this appendix document
    - https://github.com/tinyMLx/appendix/blob/main/MCUFlashing.md

BLINK EXAMPLE

- Understanding the Code in the Blink Example

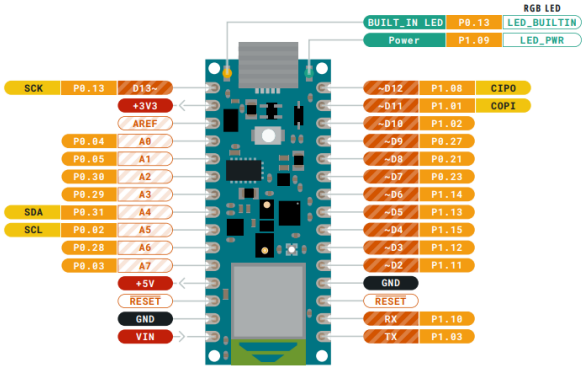
```
25  // the setup function runs once when you press reset or power the board
26  void setup() {
27    // initialize digital pin LED_BUILTIN as an output.
28    pinMode(LED_BUILTIN, OUTPUT);
29  }
30
31  // the loop function runs over and over again forever
32  void loop() {
33    digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
34    delay(1000);                      // wait for a second
35    digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
36    delay(1000);                      // wait for a second
37  }
```

BLINK EXAMPLE

- Understanding the Code in the Blink Example
  - It consists of two functions: setup and loop
  - This is the standard setup for an Arduino sketch
  - This is because when the Arduino turns on it runs the setup() function ONCE to initialize the sketch
  - Then it runs the loop() function infinitely many times to execute the sketch
  - This works well for most tinyML applications as they are designed to respond to continuous sensor input
  - You can imagine in the case of Keyword spotting that we need to initialize the neural network and the microphone and then in a loop we want to listen to audio and trigger (or not) depending upon the output of the neural network

BLINK EXAMPLE

- Understanding the Code in the Blink Example
  - You'll also notice that both functions have the void return type as they do not ever return anything but instead have side effects
  - For example, in the setup function the LED_BUILTIN (which is a shortcut name for the pin that controls the voltage to the LED) is set to be an output for the duration of the loop
  - In general you will need to set all of the pins you use as either inputs or outputs during the setup function
  - If you wired up the camera yourself you have already explored a lot of the special names reserved for the pins as shown on the pinout diagram



BLINK EXAMPLE

- Understanding the Code in the Blink Example
  - In the loop function you'll notice that we are alternating between writing a H IGH (turning on the LED) and writing a LOW (turning off the LED)
  - The delay of 1000 milliseconds (1 second) between each step is crucial as otherwise the light would turn on and off so fast that it would be imperceptible
  - In fact if you make the delay too short the light will simply seem to be dim
  - This is a trick called Pulse Width Modulation that is actually used often in industry to control motors
  - Feel free to experiment with modifying these delays and redeploying the code to see its effect

TESTING THE SENSORS

- We can turn our attention to verifying that the sensors required for this course are functioning properly

- By the end of this experiment, 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

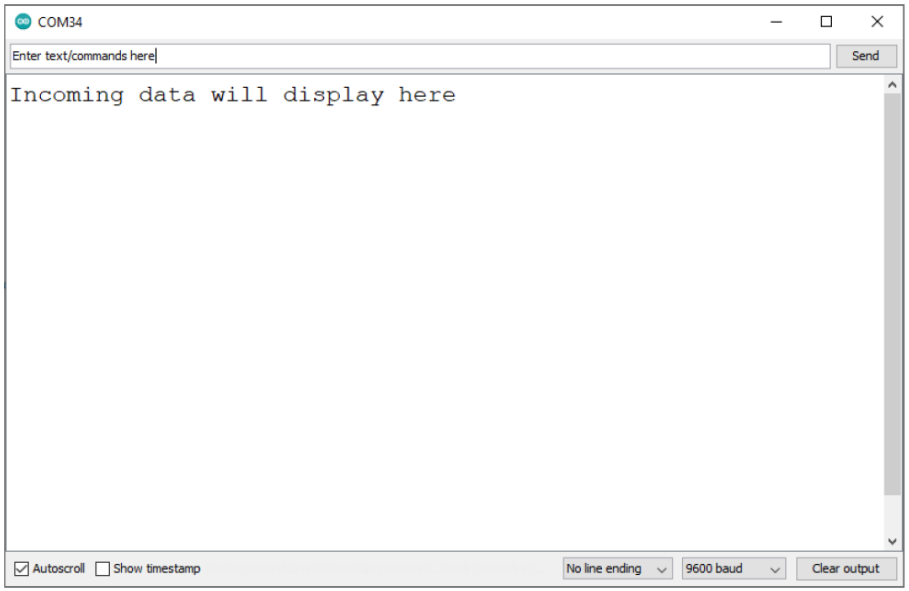
- We will also detail the optional extensions you'll need to do to obtain a live video feed from the camera as well

TESTING THE SENSORS

- 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

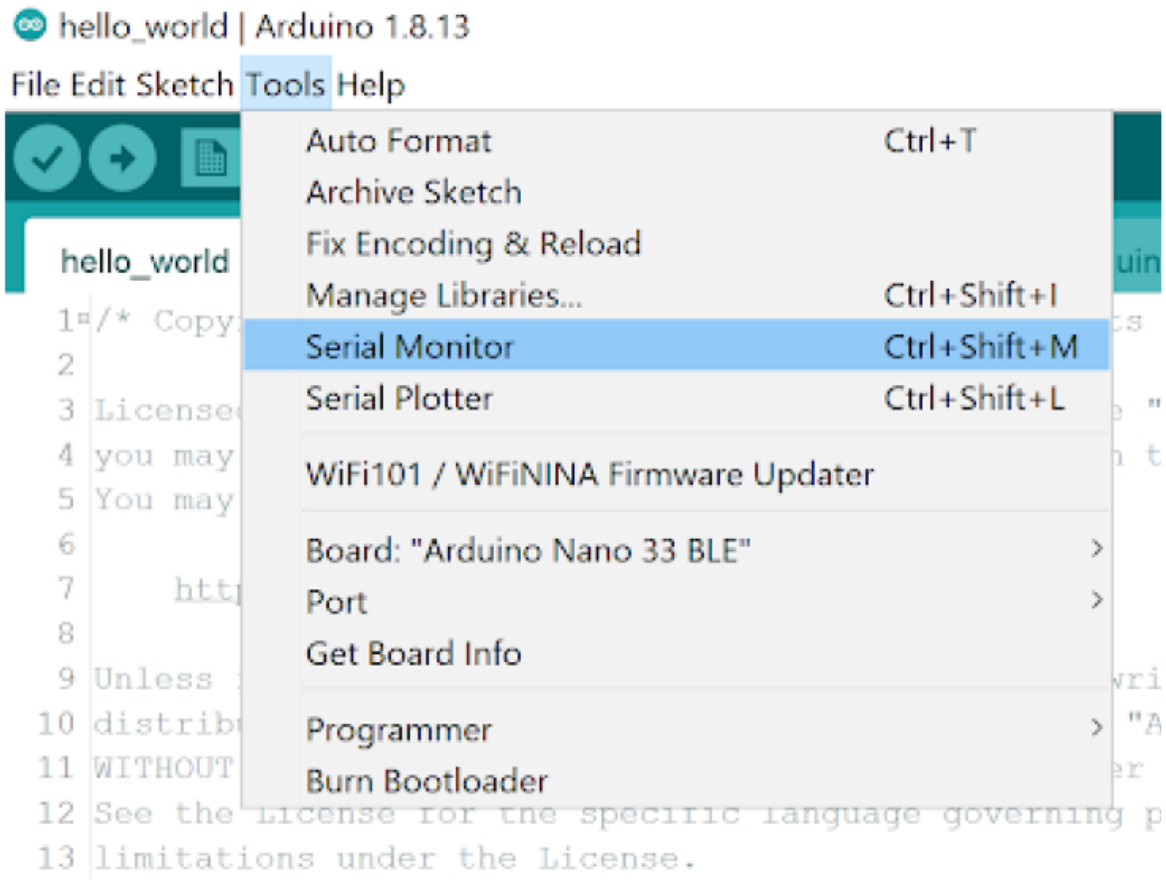
TESTING THE SENSORS

- The Serial Monitor & Plotter
  - 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 is 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



TESTING THE SENSORS

- 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 SENSORS

- Testing the Microphone
  - 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
  - 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

TESTING THE SENSORS

- Testing the Microphone
  - 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

TESTING THE SENSORS

- Testing the Microphone
  - 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 → C OM<#>
    - imacOS → /dev/cu.usbmodem<#>
    - Linux → ttyUSB<#> or ttyACM<#>

TESTING THE SENSORS

- Testing the Microphone
  - Use the rightward arrow next to the ‘compile’ checkmark to upload/flash the code
  - You’ll know the upload is complete when you see the 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”

TESTING THE SENSORS

- Testing the Microphone
  - Open the Serial Monitor
  - You can accomplish this in three different ways as shown
  - 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
    - Use the menu to select: Tools → Serial Monitor
    - Click on the magnifying glass at the top right of the Arduino Desktop IDE
    - Use the keyboard shortcut: CTRL + SHIFT + M o r CMD + SHIFT + M depending on your operating system

TESTING THE SENSORS

- Testing the Microphone
- When the Monitor opens, you should see the following text appear:

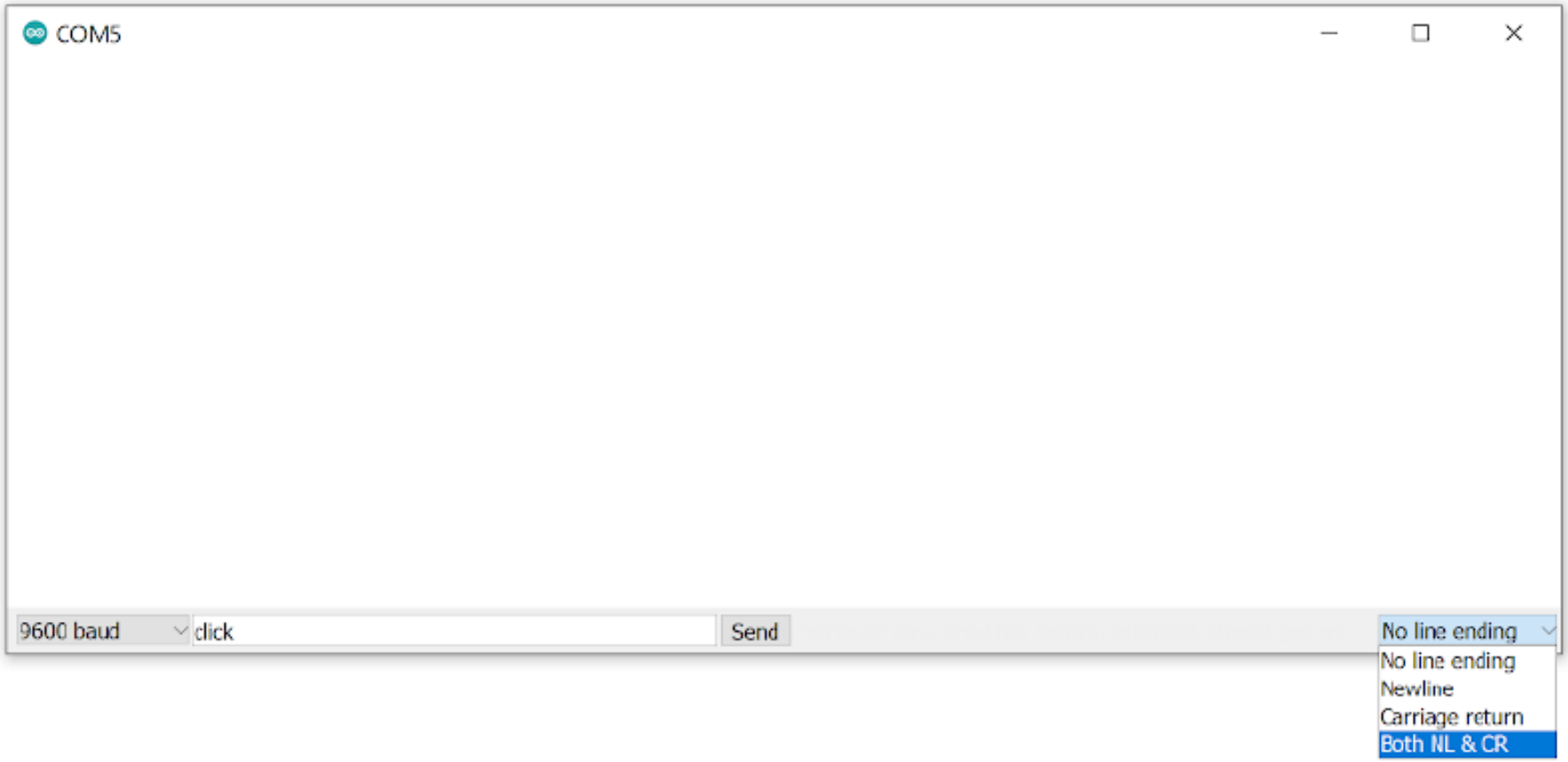
```
Welcome to the microphone test for the built-in microphone on the
Nano 33 BLE Sense

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
```


TESTING THE SENSORS

- Testing the Microphone
  - 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
  - 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

TESTING THE SENSORS

- Testing the Microphone



TESTING THE SENSORS

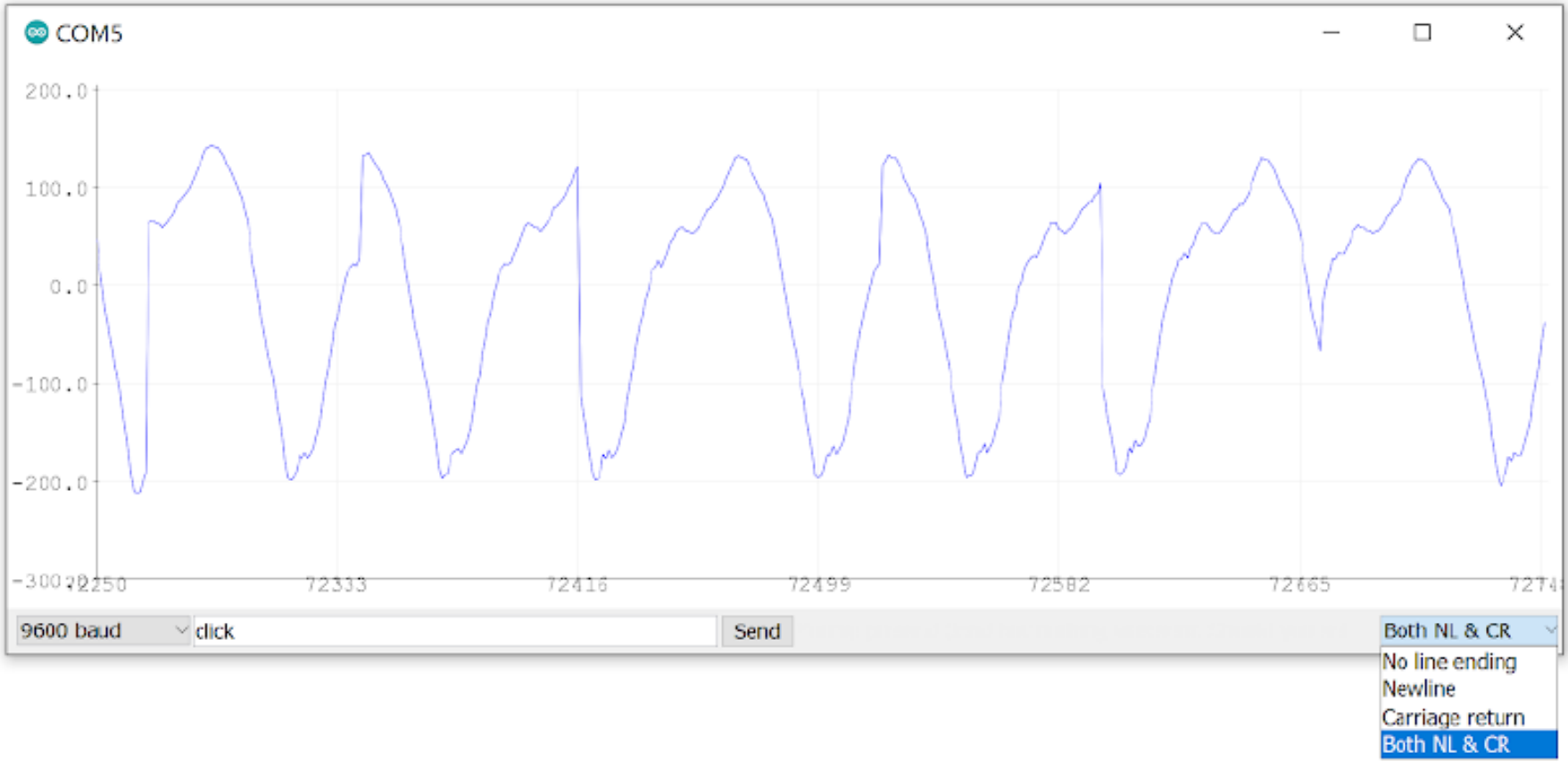
- Testing the Microphone
  - To help visualize this a little better we are going to use the Serial Plotter
  - Note that you cannot have both the Serial Monitor and Serial Plotter open at the same time as the Arduino can only communicate over a single serial port
  - Importantly, this also means that 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:
    - Use the menu to select: Tools → Serial Plotter
    - Use the keyboard shortcut: CTRL + SHIFT + L o r CMD + SHIFT + L depending on your operating system

TESTING THE SENSORS

- Testing the Microphone
  - 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

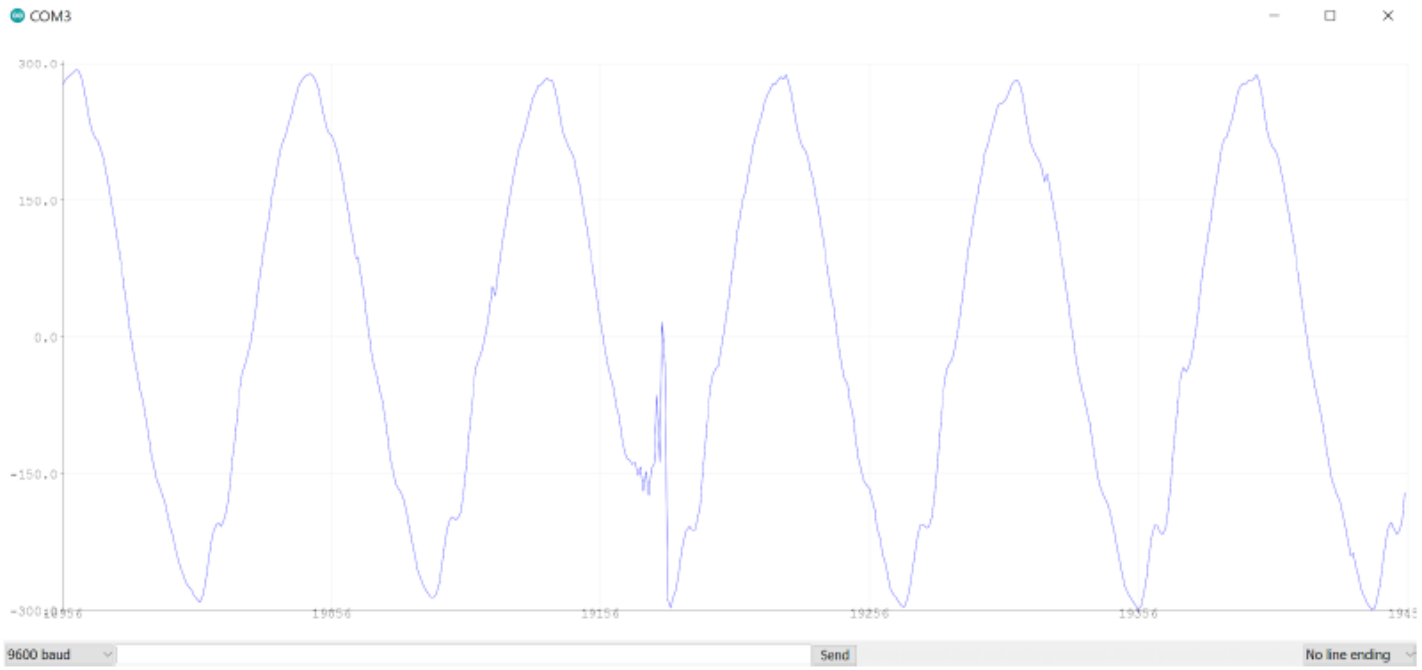
TESTING THE SENSORS

- Testing the Microphone



TESTING THE SENSORS

- Testing the Microphone
  - 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



TESTING THE INERTIAL MEASUREMENT UNIT (IMU)

- Testing the Inertial Measurement Unit
  - Now l et's open the test_IMU.ino sketch, which you can find via the File drop-down menu
  - Navigate, as follows: F ile → Examples → Harvard_TinyMLx → test_IMU
  - Use the Tools drop down menu to select appropriate Port and Board
  - 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

TESTING THE INERTIAL MEASUREMENT UNIT (IMU)

- Testing the Inertial Measurement Unit
  - Open the Serial Monitor and you should see the following

```
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
```


TESTING THE INERTIAL MEASUREMENT UNIT (IMU)

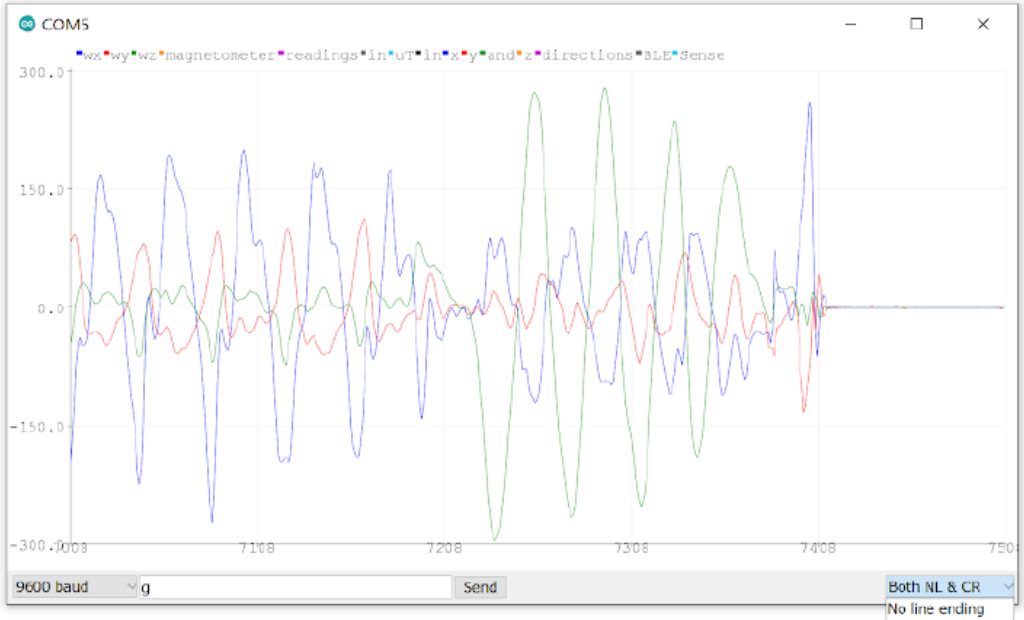
- Testing the Inertial Measurement Unit
  - Like with the microphone, in the drop down menu that reads "No line ending" by default at the bottom right of the Serial Monitor, you need to select "B oth NL & CR"
  - Now you can enter one of the possible arguments (a, g, or m) into the text entry bar across 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
    - Accelerometer: computing acceleration in the x, y, or z direction
    - Gyroscope: computing the angular velocity -- the change in the roll, pith, and yaw
    - Magnetometer: 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

TESTING THE INERTIAL MEASUREMENT UNIT (IMU)

- Testing the Inertial Measurement Unit
  - Close the Serial Monitor and open the Serial Plotter
  - As a reminder you can open the Serial Plotter in two ways:
    - Use the menu to select: Tools → Serial Plotter
    - Use the keyboard shortcut: CTRL + SHIFT + L or CMD + SHIFT + L depending on your operating system

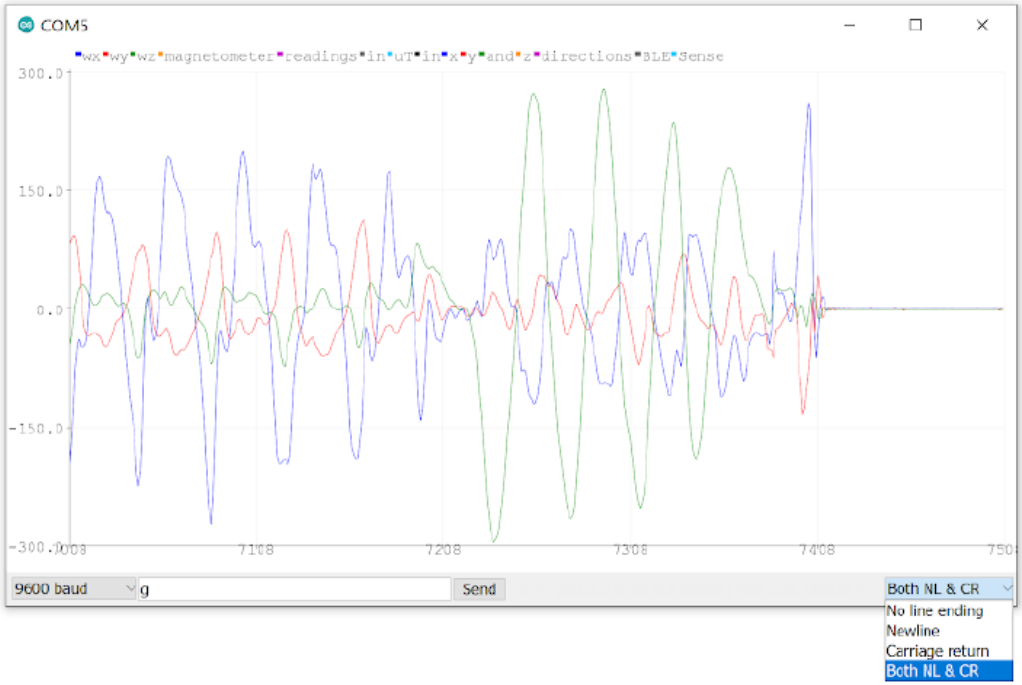
TESTING THE INERTIAL MEASUREMENT UNIT (IMU)

- Testing the Inertial Measurement Unit
  - With the Plotter open, you should see the same stream of data presented graphically
  - The most interesting one to plot is the gyrocope
  - 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"



TESTING THE INERTIAL MEASUREMENT UNIT (IMU)

- Testing the Inertial Measurement Unit
  - 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

- Testing the OV7675 Camera
  - Now l et'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
  - As always, use the Tools drop down menu to select appropriate Port and Board

TESTING THE OV7675 CAMERA

- Testing the OV7675 Camera
  - 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
  - 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

TESTING THE OV7675 CAMERA

- Testing the OV7675 Camera
  - 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
```


TESTING THE OV7675 CAMERA

- Testing the OV7675 Camera
  - 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

TESTING THE OV7675 CAMERA

- Testing the OV7675 Camera
  - 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 was created to display the image
    - https://colab.research.google.com/github/tinyMLx/colabs/blob/master/4-2-12-OV7675ImageViewer.ipynb