Software Development for Embedded ML

Michael Stanley

http://sensip.asu.edu

Mike.Stanley@ieee.org

Lecture materials can be accessed at
https://github.com/mstanley103/SenSIP_RET_2021

Arizona State University

SenSIP INDUSTRY CONSORTIUM

Lecture Mission Statement

Introduce you to embedded software development using the Arduino Nano 33 BLE Sense development board.

During the course of this lecture, we will review embedded code for streaming sensor data to your PC.

Today's Agenda

- Recap prior lecture
- Program & Debug ports
- Arduino Programming Environments
- Demos & Walthroughs
  - "Blink" code walkthrough / demo
  - Sensor data logger code / walkthrough / demos
- Data collection topics
- Python (if time allows)

Last Time…

- Vocabulary
- Options for Training & Deployment
  - Cloud
  - PC
  - Embedded
- MCU Architecture
- What you need to know about your embedded board
- Introduction to Sensors
  - Scalar sensors: Temperature, Pressure, Humidity, Microphone
  - Vector sensors: Accelerometer, Magnetometer, Gyroscope
  - Vision

Example Hardware: The Arduino Nano 33 BLE Sense

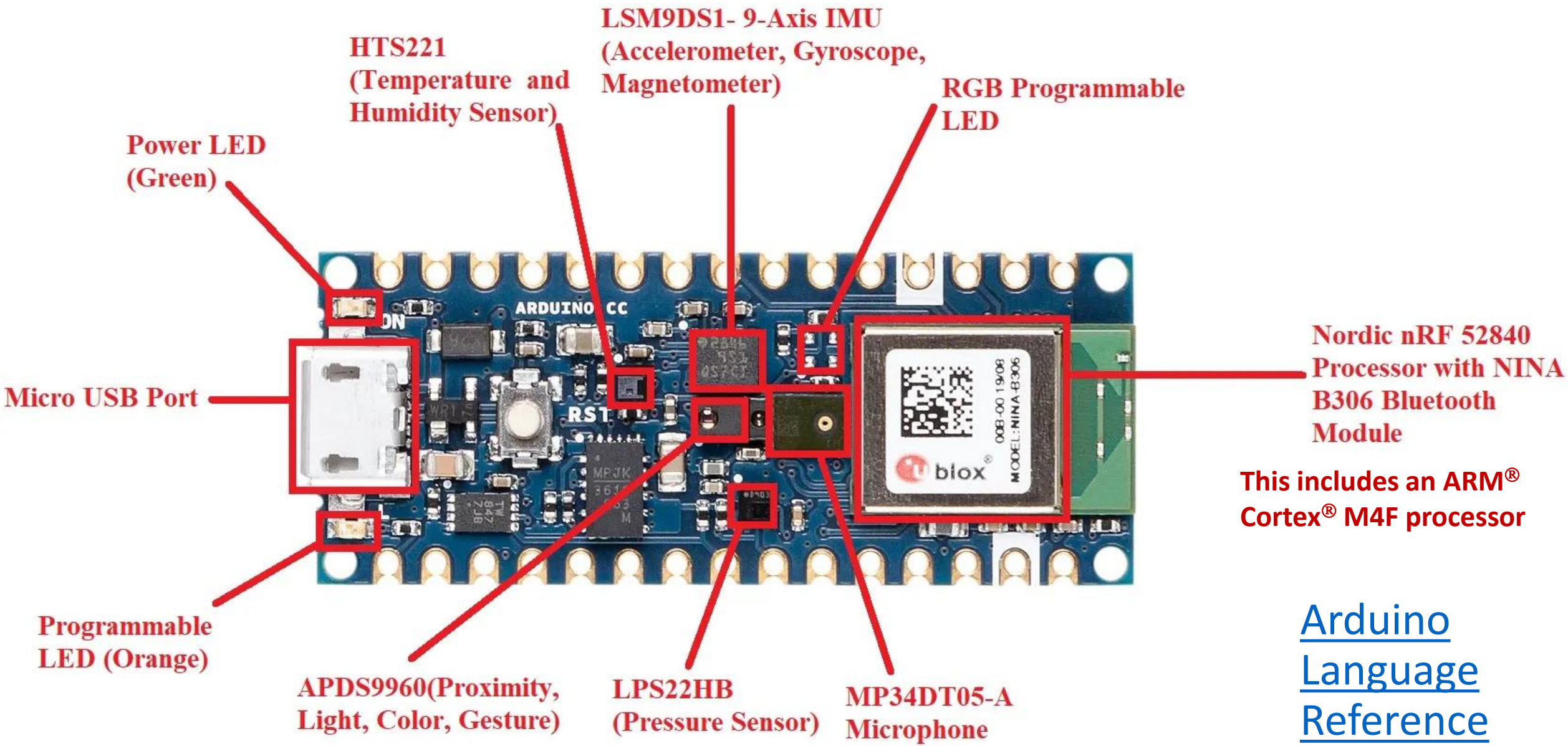


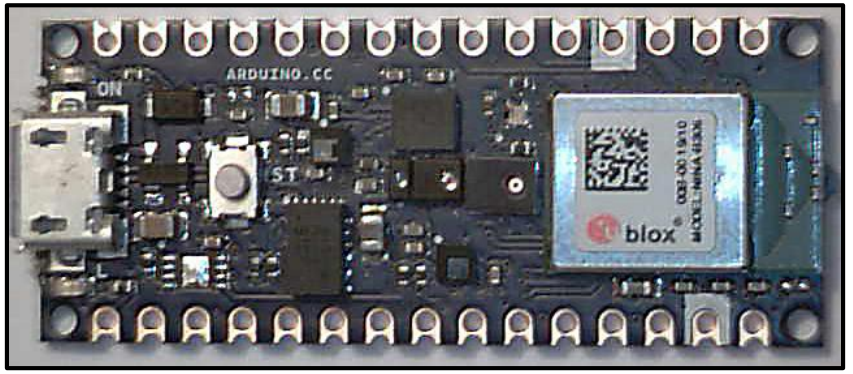
Image source: https://www.etechnophiles.com/arduino-nano-33-ble-sense-pinout-introduction-specifications/

Arduino Language Reference

Debug & Programming

Most MCUs support dedicated hardware debug ports which offer the ability:

- to start/stop the CPU
- single step through code
- examine code, variables and CPU registers
- set and clear breakpoints
- and much more
- program/erase flash memory



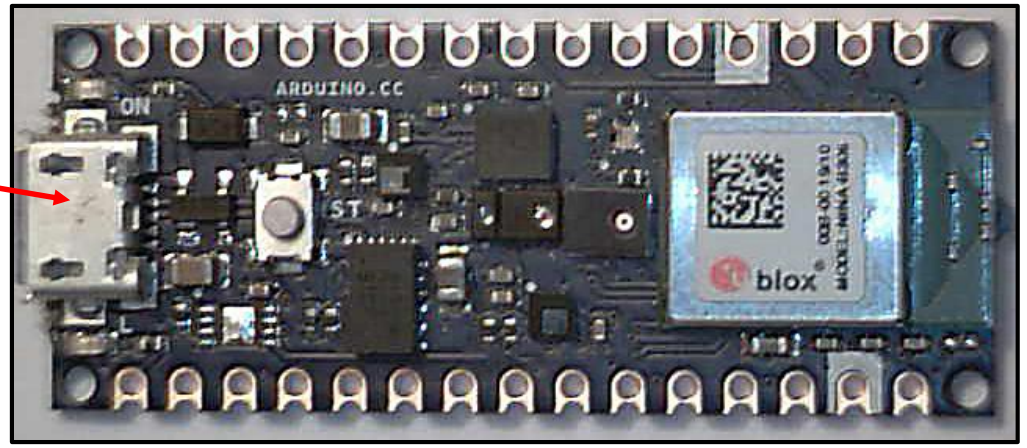
Because these consume pins (which have value and consume board space) and require an external hardware debug module, USB bootloaders have become common features for many development boards. This eliminates the need for an external, hardware-based, debug module.

The Arduino Nano 33 USB-based programming port does NOT support all of the features listed above.

WARNING: Physically, the microUSB connector is the part of the board most likely to fail. Use strain relief if you can.

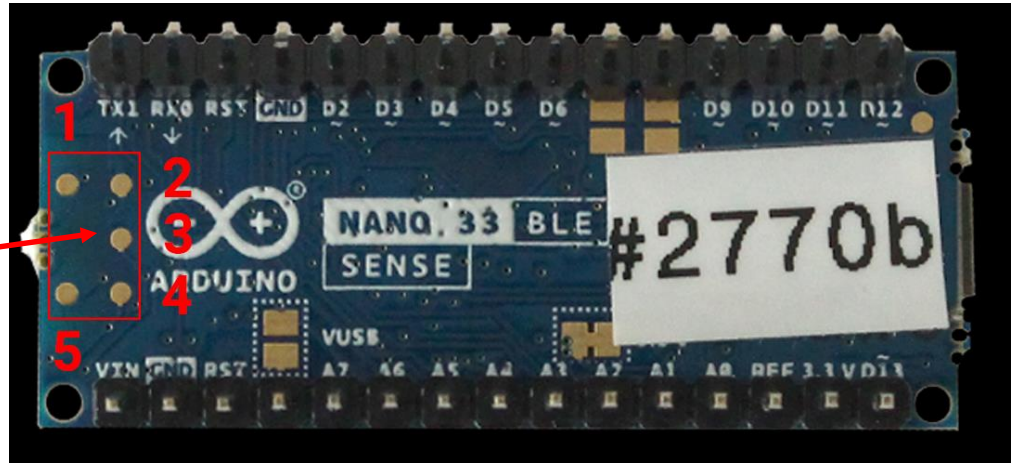
There are several ways to program/erase the Arduino Nano 33 BLE Sense

Attach a microUSB cable from board to your PC



1. A USB bootloader program can be used to download code to your board as well as implement a virtual serial port. There are two versions:
   - **ArduinoCore Mbed (default)**
   - Adafruit nRF52 (for CircuitPython)

   You need a hardware debug module attached to probe points on the bottom of the Nano to switch between them.
2. **Alternately you can use the probe points as your primary debug port.**



Hardware Debugging

To get the most from a modern software IDE, you will need to attach an external debug module such as a Segger J-link module to your board. You will need a soldering iron and a bit of time. Here are some useful links:

1. Arduino Nano 33 BLE - SEGGER Wiki
2. Arduino Nano 33 BLE/IoT Custom Debugging - Hackster.io
3. SWD Breakout and Breadboard for Nano 33 BLE Sense | Thomas J. Petz, Jr. (tjpetz.com)
4. Arduino Nano 33 IoT Debugging - Arduino Project Hub
5. J-Link EDU Mini - SEGGER - The Embedded Experts

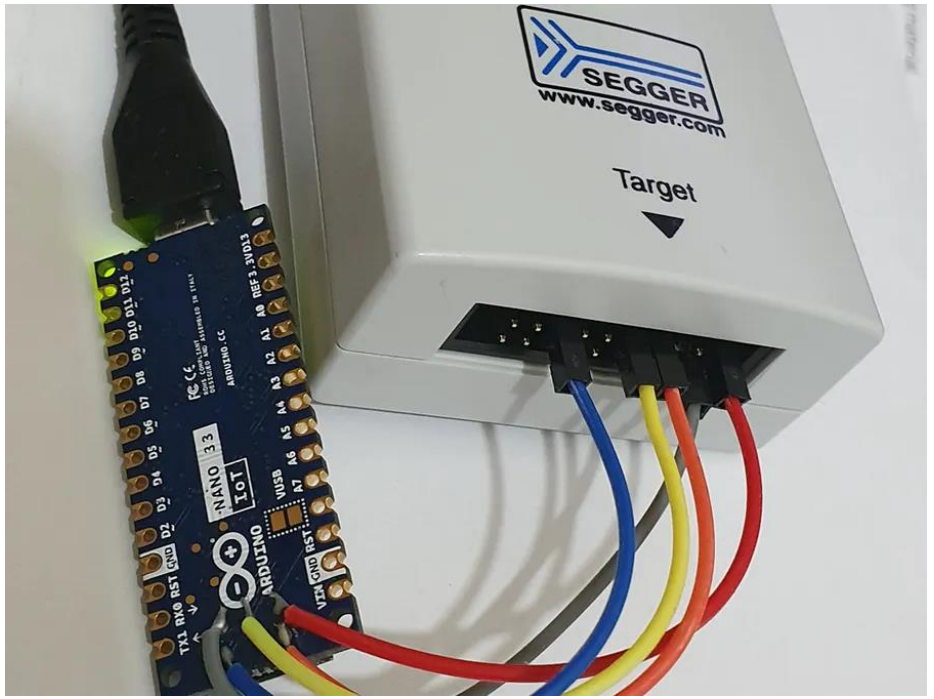


image source: [4]

Port Addresses

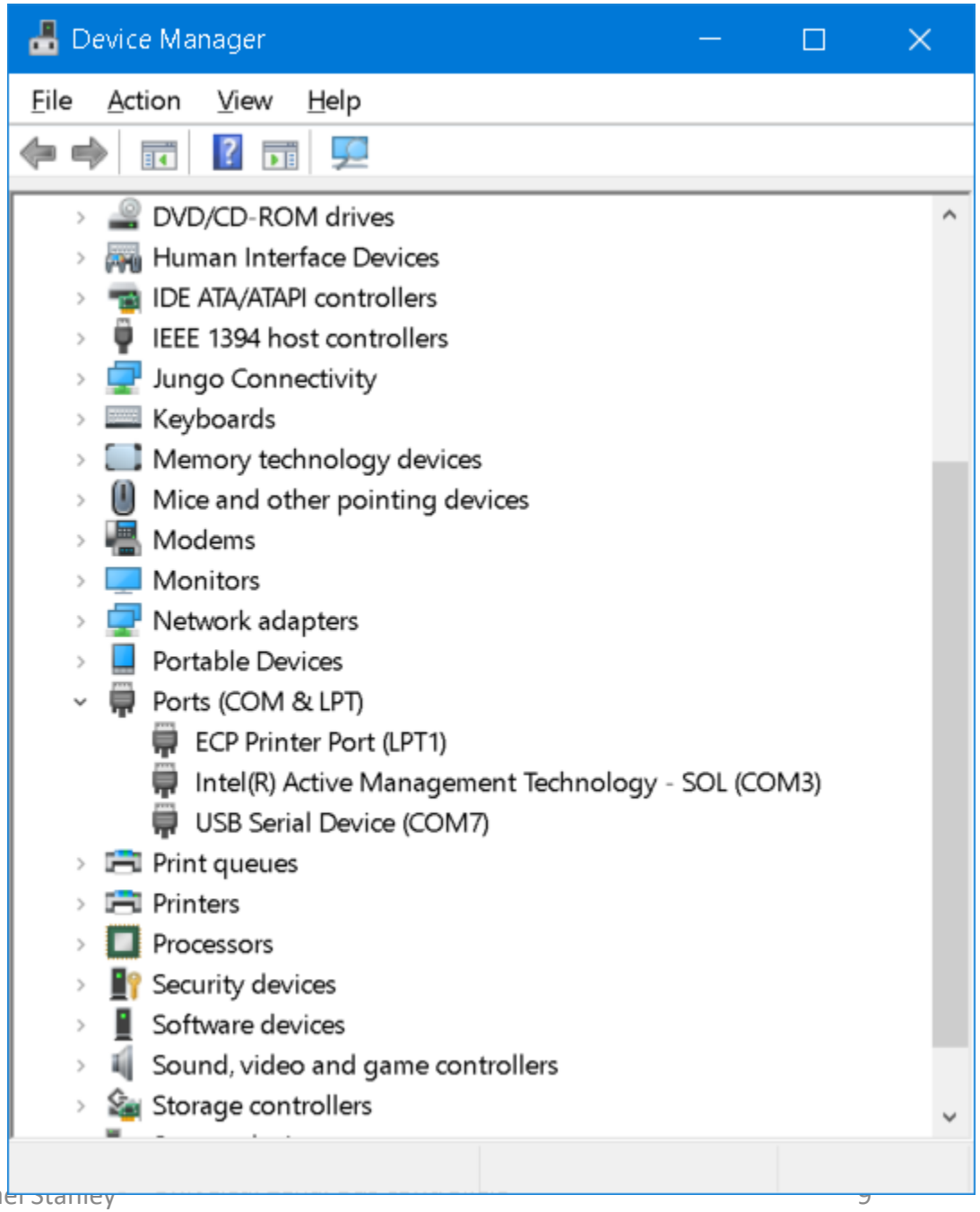
When using the default Mbed bootloader, the board will enumerate with one of two serial port addresses:

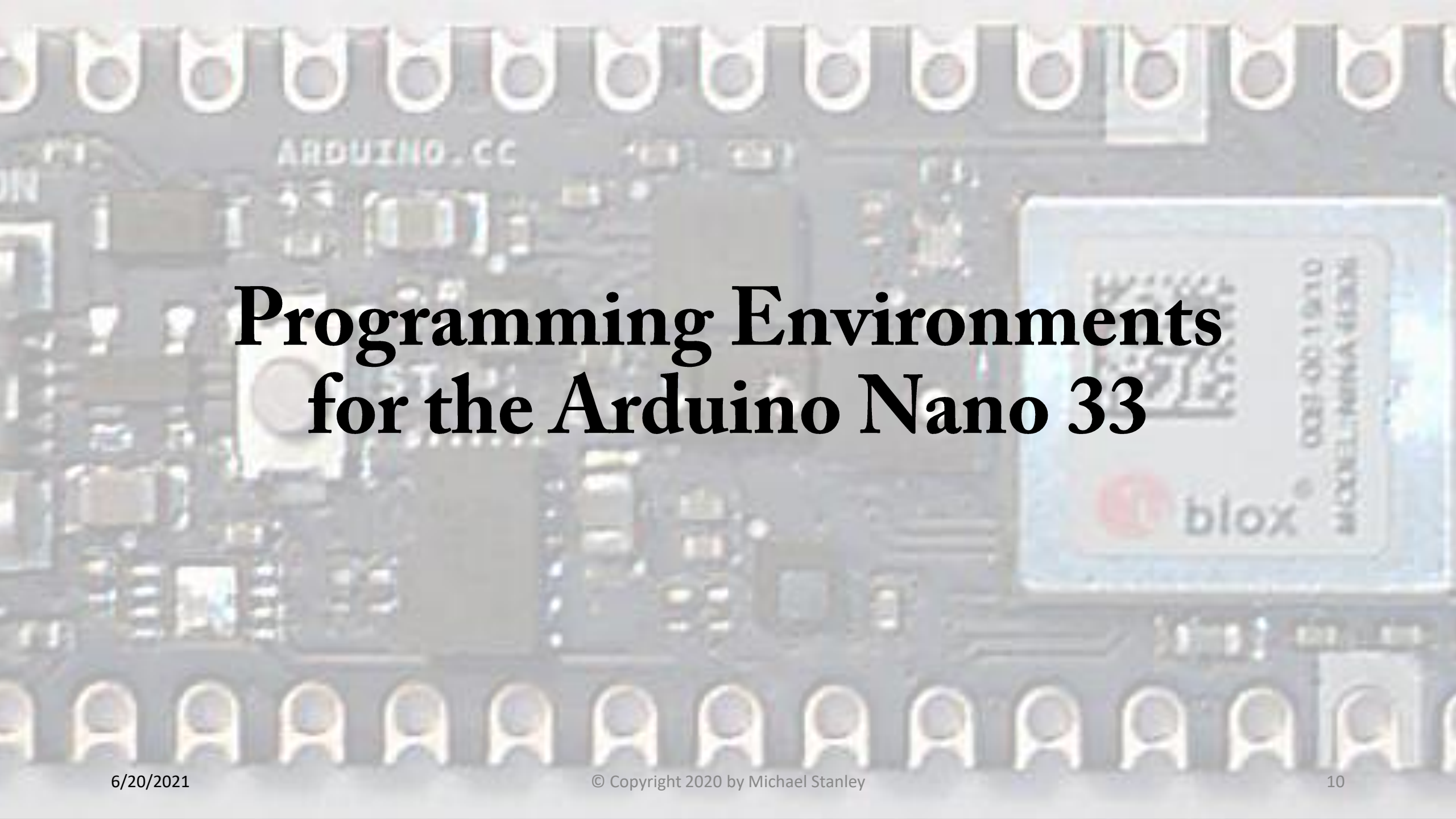
- address A = address used to program the device
- address A+1 = mission mode port ID

You will need to select the proper port address in the Arduino IDE.

This two address scheme is somewhat unique to the Arduino Nano 33 family. Other ARM devices running Mbed or OpenSDA interfaces typically only consume one address.

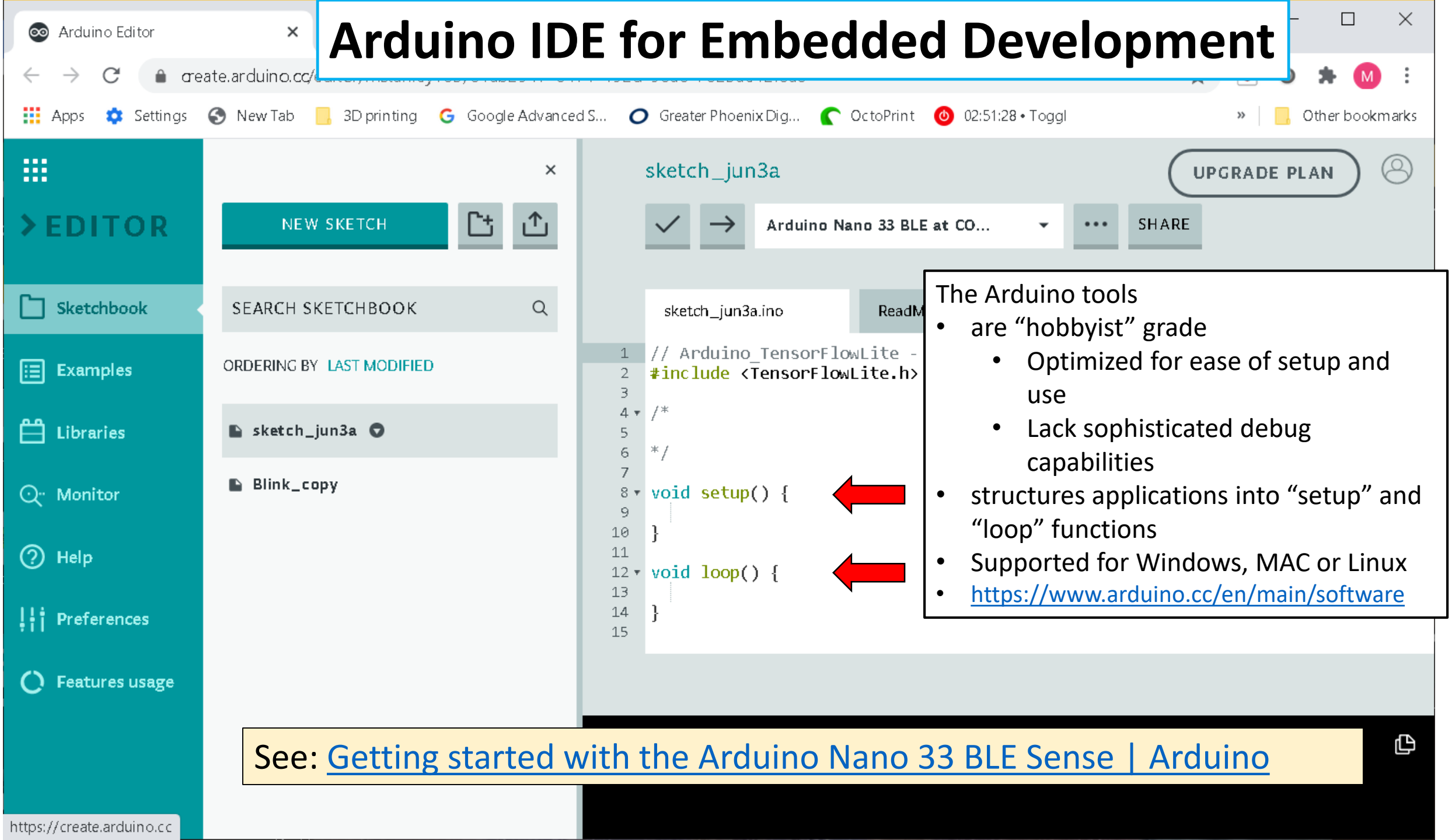
Clicking the reset button twice in fast succession will put you into bootloader mode. Once should get you mission mode.





Programming Environments for the Arduino Nano 33

Arduino IDE for Embedded Development



The Arduino tools

- are “hobbyist” grade
  - Optimized for ease of setup and use
  - Lack sophisticated debug capabilities
- structures applications into “setup” and “loop” functions
- Supported for Windows, MAC or Linux
- https://www.arduino.cc/en/main/software

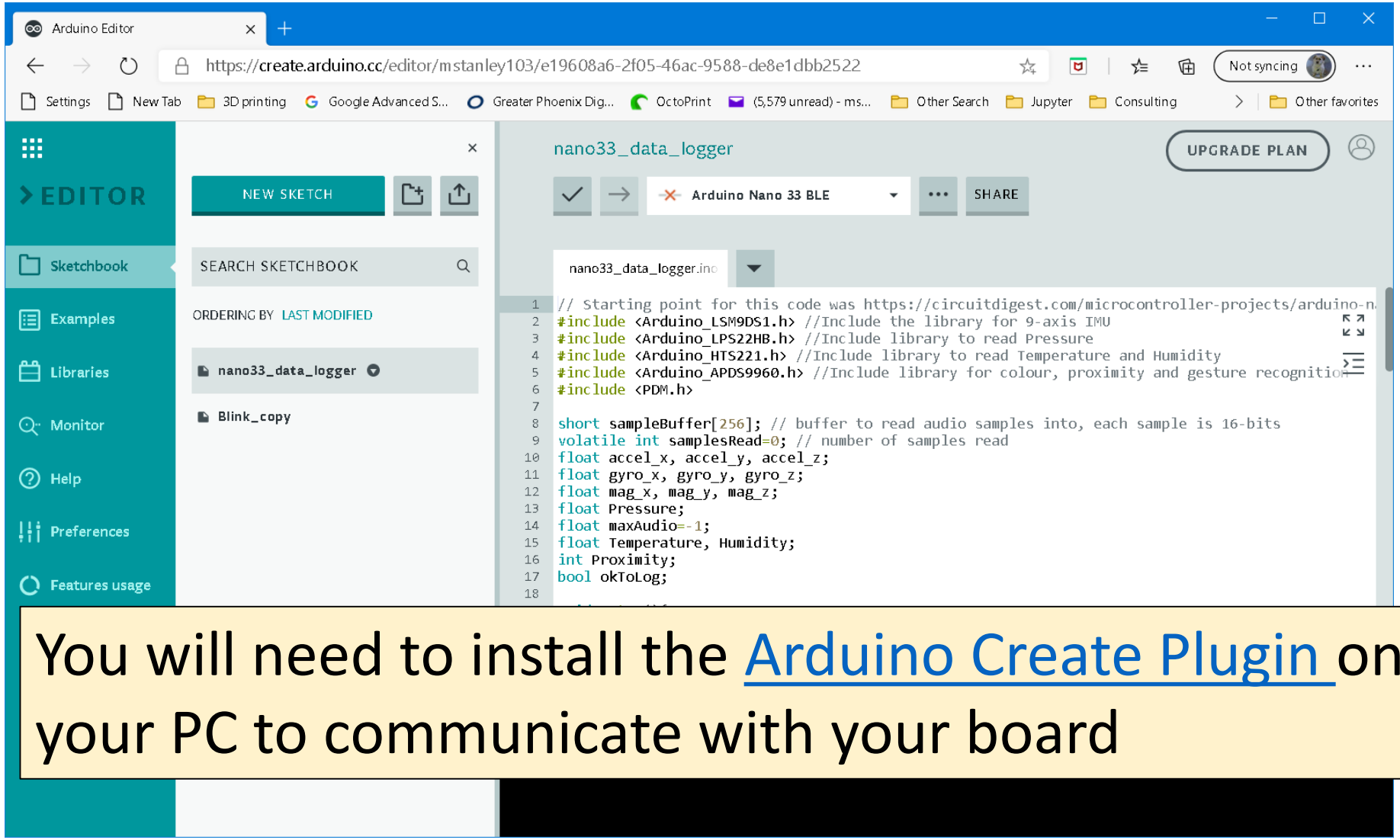
See: Getting started with the Arduino Nano 33 BLE Sense | Arduino

Libraries you may want to use with the Arduino Nano 33 BLE Sense

- https://www.arduino.cc/en/Reference/ArduinoLSM9DS1
- https://www.arduino.cc/en/Reference/PDM (appears to be preinstalled for Arduino)
- https://www.arduino.cc/en/Reference/ArduinoAPDS9960
- https://www.arduino.cc/en/Reference/ArduinoLPS22HB
- https://www.arduino.cc/en/Reference/ArduinoHTS221
- https://www.arduino.cc/reference/en/libraries/arduino_tensorflowlite/

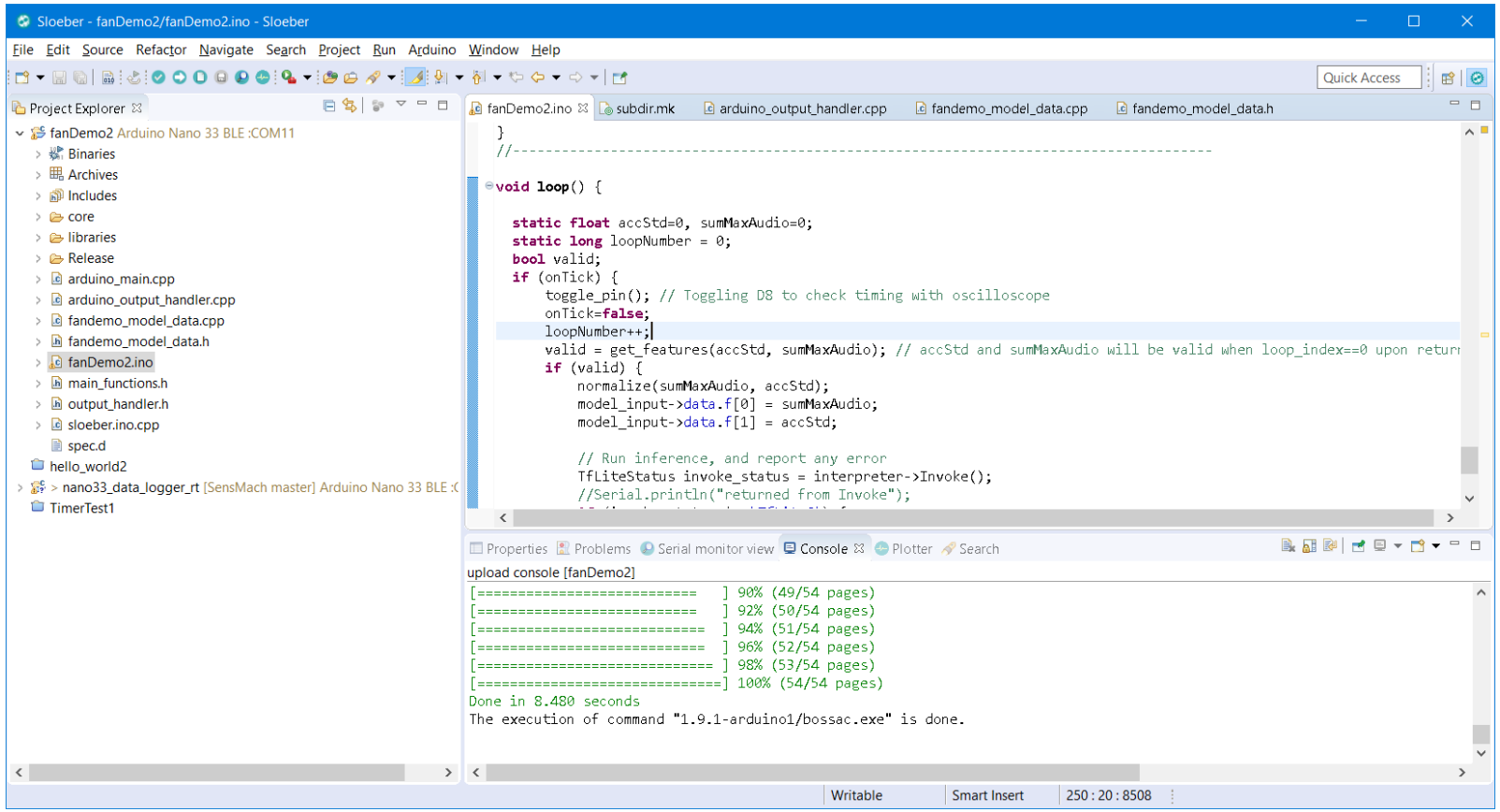
See: Getting started with the Arduino Nano 33 BLE Sense | Arduino

A Cloud-based Version is also Available



You will need to install the Arduino Create Plugin on your PC to communicate with your board

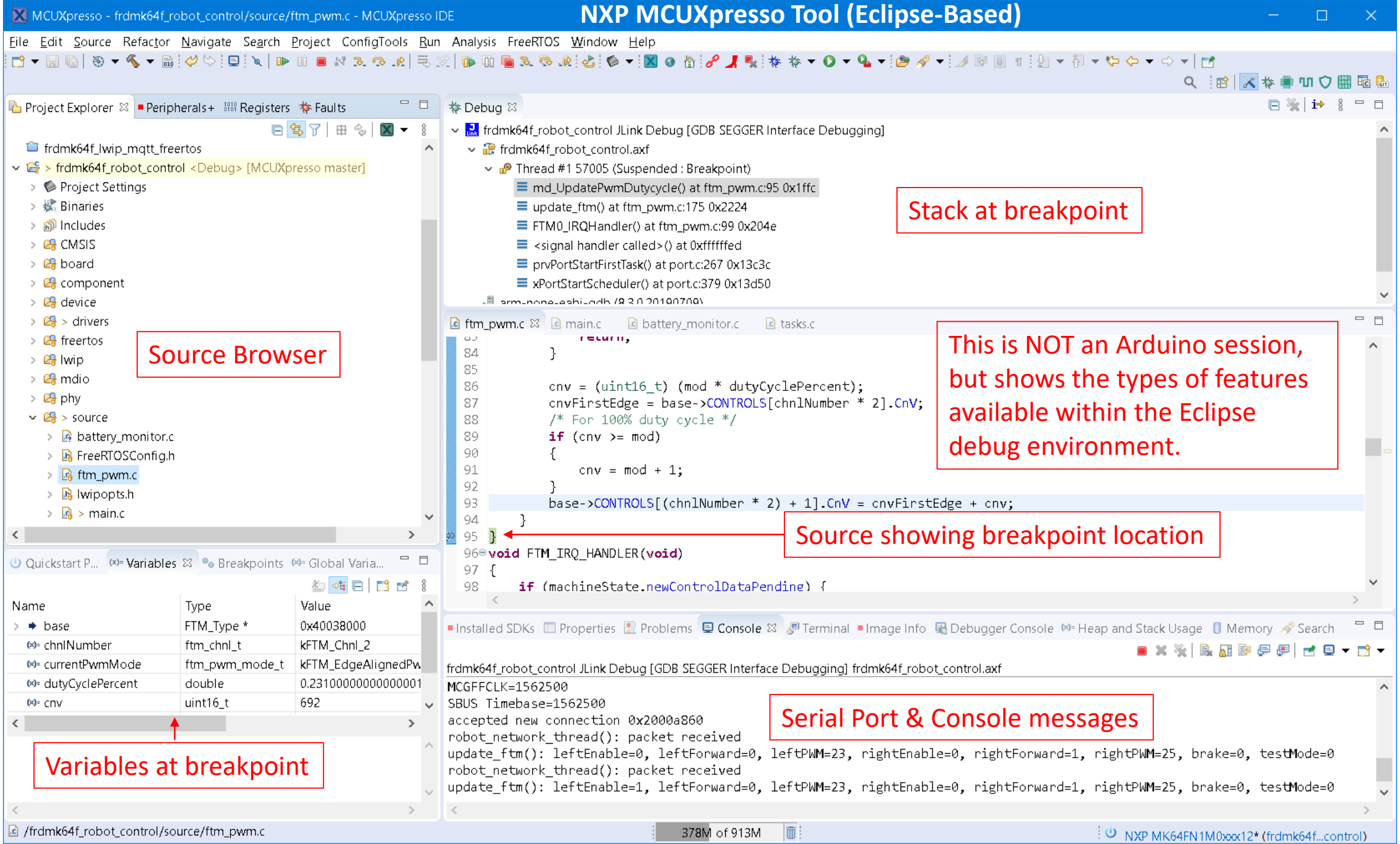
More Sophisticated Coders Will Prefer Eclipse



The Sloeber plugin for Eclipse provides:

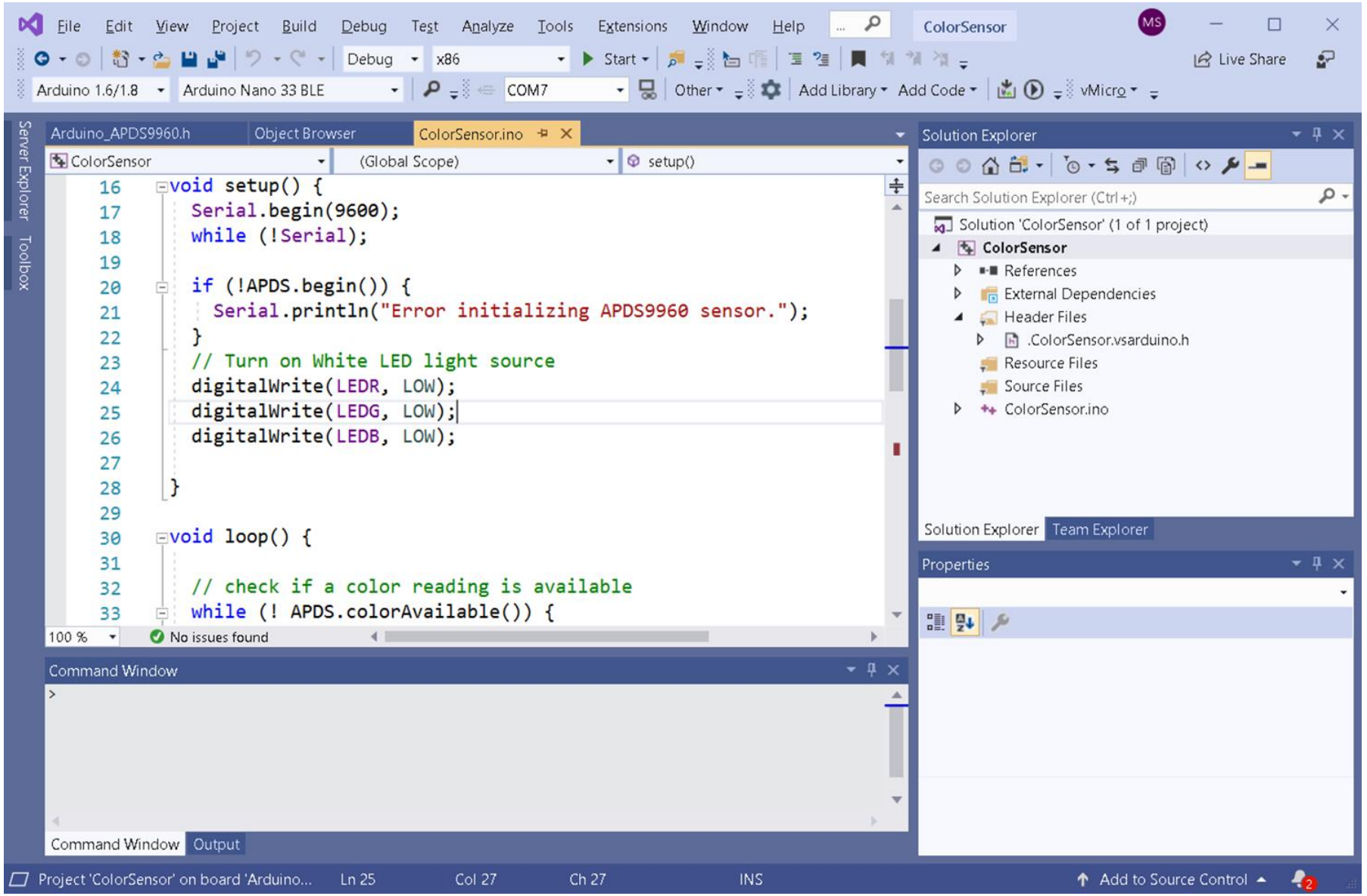
- Full C/C++ language support
- Faster build times
- Code tracing
- Compatibility with the standard Arduino libraries
- Built in serial port monitor
- Hardware debug support (may require an external debug module and board modification)

- https://eclipse.baeyens.it/

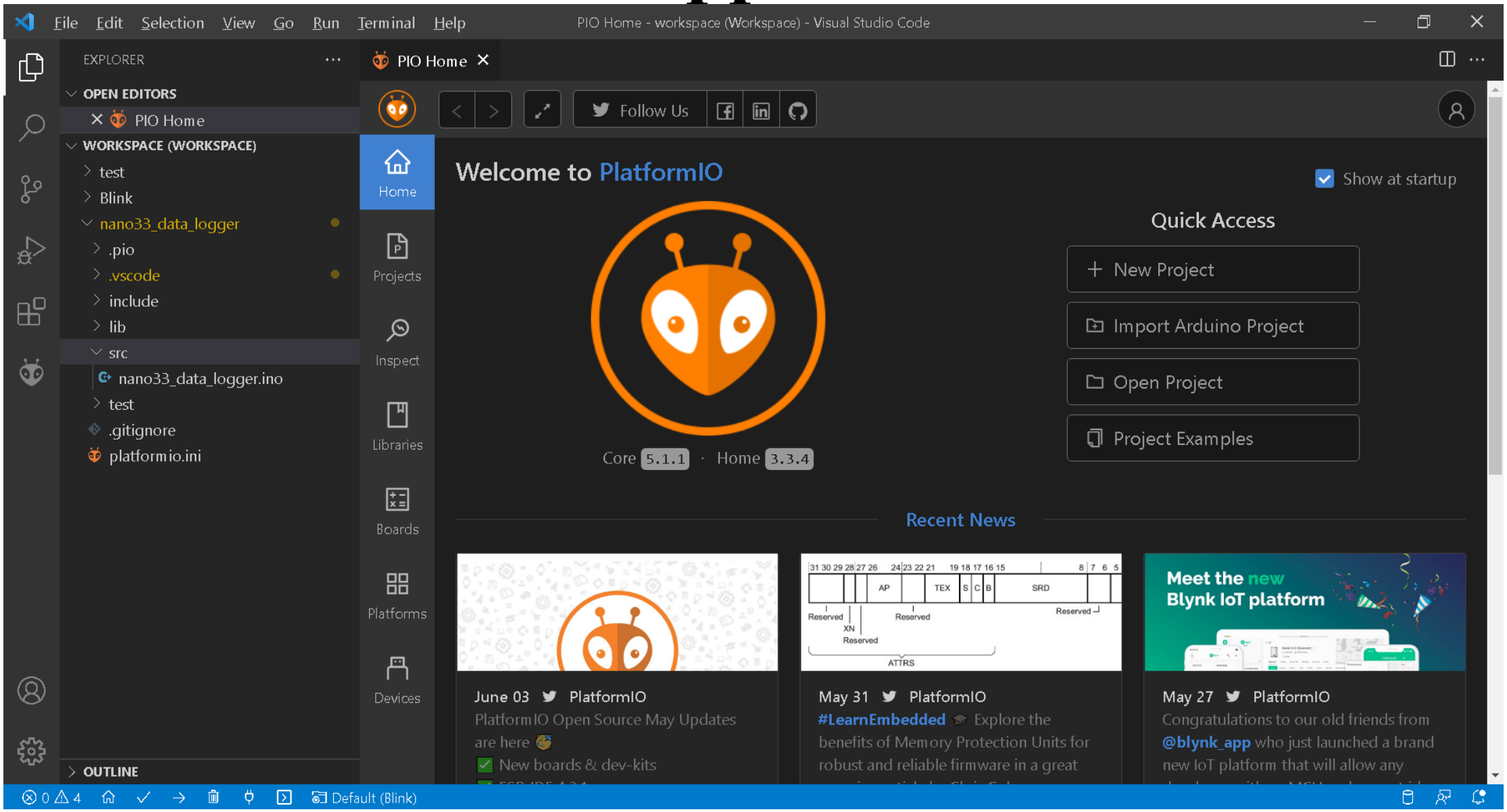

MCUXpresso - frdmk64f_robot_control/source/ftm_pwm.c - MCUXpresso IDE
NXP MCUXpresso Tool (Eclipse-Based)
File Edit Source Refactor Navigate Search Project ConfigTools Run Analysis FreeRTOS Window Help
Project Explorer
Peripherals+
Registers
Faults
frdmk64f_lwip_mqtt_freertos
frdmk64f_robot_control <Debug> [MCUXpresso master]
Project Settings
Binaries
Includes
CMSIS
board
component
device
drivers
freertos
lwip
mdio
phy
source
battery_monitor.c
FreeRTOSConfig.h
ftm_pwm.c
lwipopts.h
main.c
Source Browser
Debug
frdmk64f_robot_control JLink Debug [GDB SEGGER Interface Debugging]
frdmk64f_robot_control.axf
Thread #1 57005 (Suspended : Breakpoint)
md_UpdatePwmDutycycle() at ftm_pwm.c:95 0x1ffc
update_ftm() at ftm_pwm.c:175 0x2224
FTM0_IRQHandler() at ftm_pwm.c:99 0x204e
<signal handler called>() at 0xfffffffed
prvPortStartFirstTask() at port.c:267 0x13c3c
xPortStartScheduler() at port.c:379 0x13d50
Stack at breakpoint
ftm_pwm.c
main.c
battery_monitor.c
tasks.c
84 }
85
86 cnv = (uint16_t) (mod * dutyCyclePercent);
87 cnvFirstEdge = base->CONTROLS[chnlNumber * 2].CnV;
88 /* For 100% duty cycle */
89 if (cnv >= mod)
90 {
91 cnv = mod + 1;
92 }
93 base->CONTROLS[(chnlNumber * 2) + 1].CnV = cnvFirstEdge + cnv;
94 }
95 }
96 void FTM_IRQ_HANDLER(void)
97 {
98 if (machineState.newControlDataPending) {
This is NOT an Arduino session, but shows the types of features available within the Eclipse debug environment.
Source showing breakpoint location
Quickstart P...
Variables
Breakpoints
Global Varia...
Name Type Value
base FTM_Type * 0x40038000
chnlNumber ftm_chnl_t kFTM_Chnl_2
currentPwmMode ftm_pwm_mode_t kFTM_EdgeAlignedPw
dutyCyclePercent double 0.231000000000000001
cnv uint16_t 692
Variables at breakpoint
Installed SDKs
Properties
Problems
Console
Terminal
Image Info
Debugger Console
Heap and Stack Usage
Memory
Search
frdmk64f_robot_control JLink Debug [GDB SEGGER Interface Debugging] frdmk64f_robot_control.axf
MCGFFCLK=1562500
SBUS Timebase=1562500
accepted new connection 0x2000a860
robot_network_thread(): packet received
update_ftm(): leftEnable=0, leftForward=0, leftPWM=23, rightEnable=0, rightForward=1, rightPWM=25, brake=0, testMode=0
robot_network_thread(): packet received
update_ftm(): leftEnable=1, leftForward=0, leftPWM=23, rightEnable=0, rightForward=1, rightPWM=25, brake=0, testMode=0
Serial Port & Console messages
/frdmk64f_robot_control/source/ftm_pwm.c
378M of 913M
NXP MK64FN1M0xxx12* (frdmk64f...control)

A relatively new option is the Visual Micro extension to Visual Studio

(VisualMicro - Arduino IDE For Visual Studio)



The best of the lot appears to be PlatformIO



PlatformIO

- PlatformIO (https://platformio.org/) is available on a wide variety of hardware platforms
- As shown here, it is a plugin run within Microsoft Visual Studio Code
- PlatformIO is written in (and requires) pure Python
- It has code autocomplete, function cross-referencing, fast build and download and memory profiling features
- Look for your projects under Documents\PlatformIO\Projects

```
[env:stable]
platform = nordicnrf52
board = nano33ble
framework = arduino
monitor_speed = 115200
; change MCU frequency
board_build.f_cpu = 64000000L
; change microcontroller
board_build.mcu = nrf52840
```

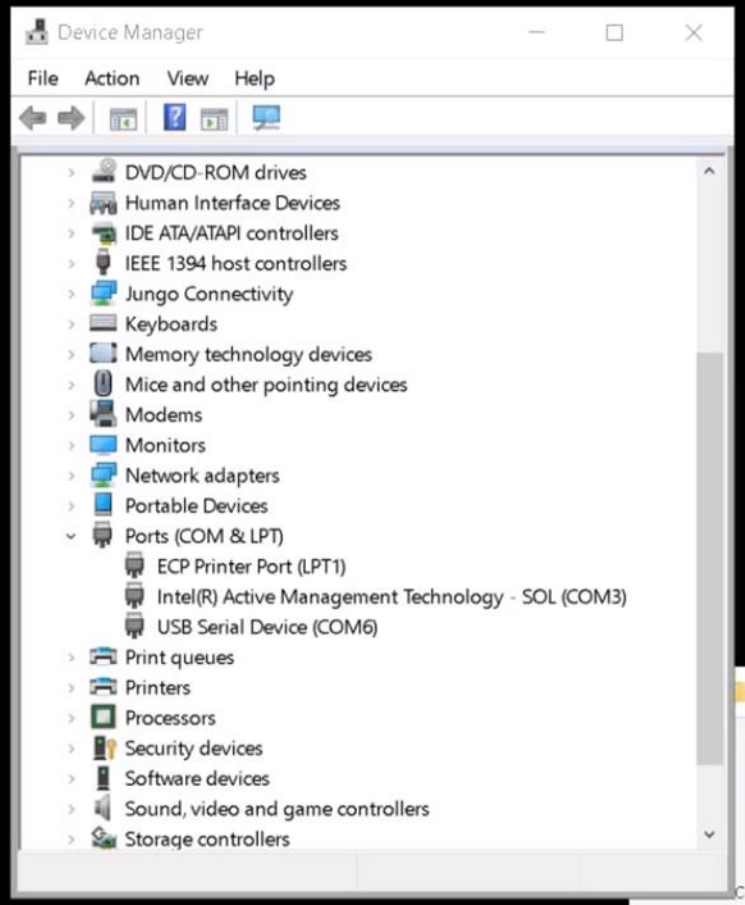
You will need the settings above included in your projects platformio.ini file for proper operation with your Arduino Nano 33. Note that the monitor_speed setting must match the baud rate you initialize your serial port to.

Adding Sensors & the SSD1306 library

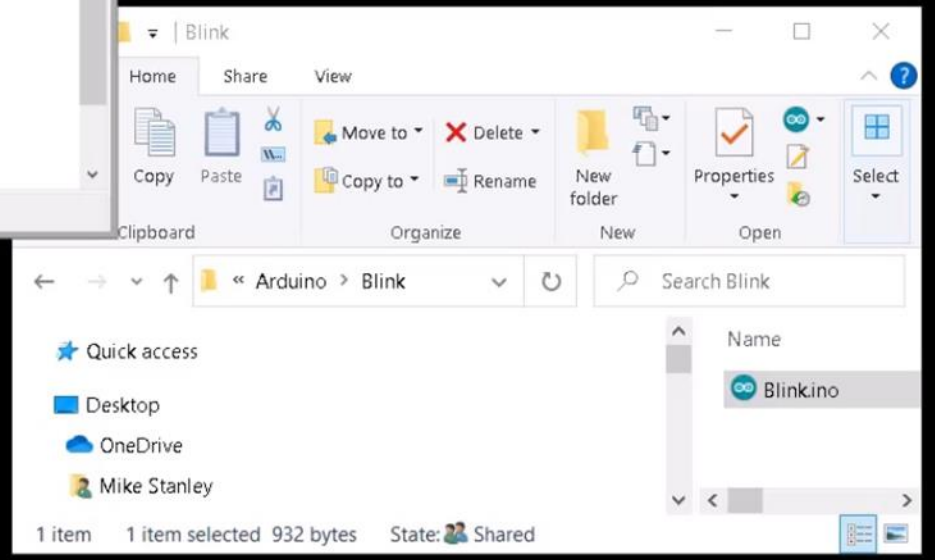
```
[env:stable]
platform = nordicnrf52
board = nano33ble
framework = arduino
monitor_speed = 115200
board_build.f_cpu = 64000000L
board_build.mcu = nrf52840
lib_deps =
    arduino-libraries/Arduino_LSM9DS1@^1.1.0
    arduino-libraries/Arduino_LPS22HB@^1.0.1
    arduino-libraries/Arduino_HTS221@^1.0.0
    arduino-libraries/Arduino_APDS9960@^1.0.3
    adafruit/Adafruit SSD1306@^2.4.6
build_flags =
    -I"C:\Users\Mike\Documents\Arduino\libraries\Adafruit_GFX_Library"
    -I"C:\Users\Mike\Documents\Arduino\libraries\Adafruit_SSD1306"
    -I"C:\Users\Mike\Documents\Arduino\libraries\Adafruit_BusIO"
```

Make sure you correct these paths to your installation before building.

IDE Demos



Break Here for "Blink" Cloud-based Arduino IDE Demo & Code Walkthrough



Arduino Editor

https://create.arduino.cc/editor/mstanley103/e19608a6-2f05-46ac-9588-de8e1dbb2522

Not syncing

Settings | Google Advanced S... | 3D printing | Greater Phoenix Dig... | (5,579 unread) - ms... | Other Search | Little Ceasars | Bookmarks | Google Calendar - 4... | OctoPrint | Settings | Other favorites

EDITOR

NEW SKETCH

SEARCH SKETCHBOOK

ORDERING BY LAST MODIFIED

- Sketchbook
- Examples
- Libraries
- Monitor
- Reference
- Help
- Preferences
- Features usage

ColorSensor (1)

nano33_data_logger

nano33_data_logger_rt

CLOUD

nano33_data_logger

UPGRADE PLAN

Arduino Nano 33 BLE

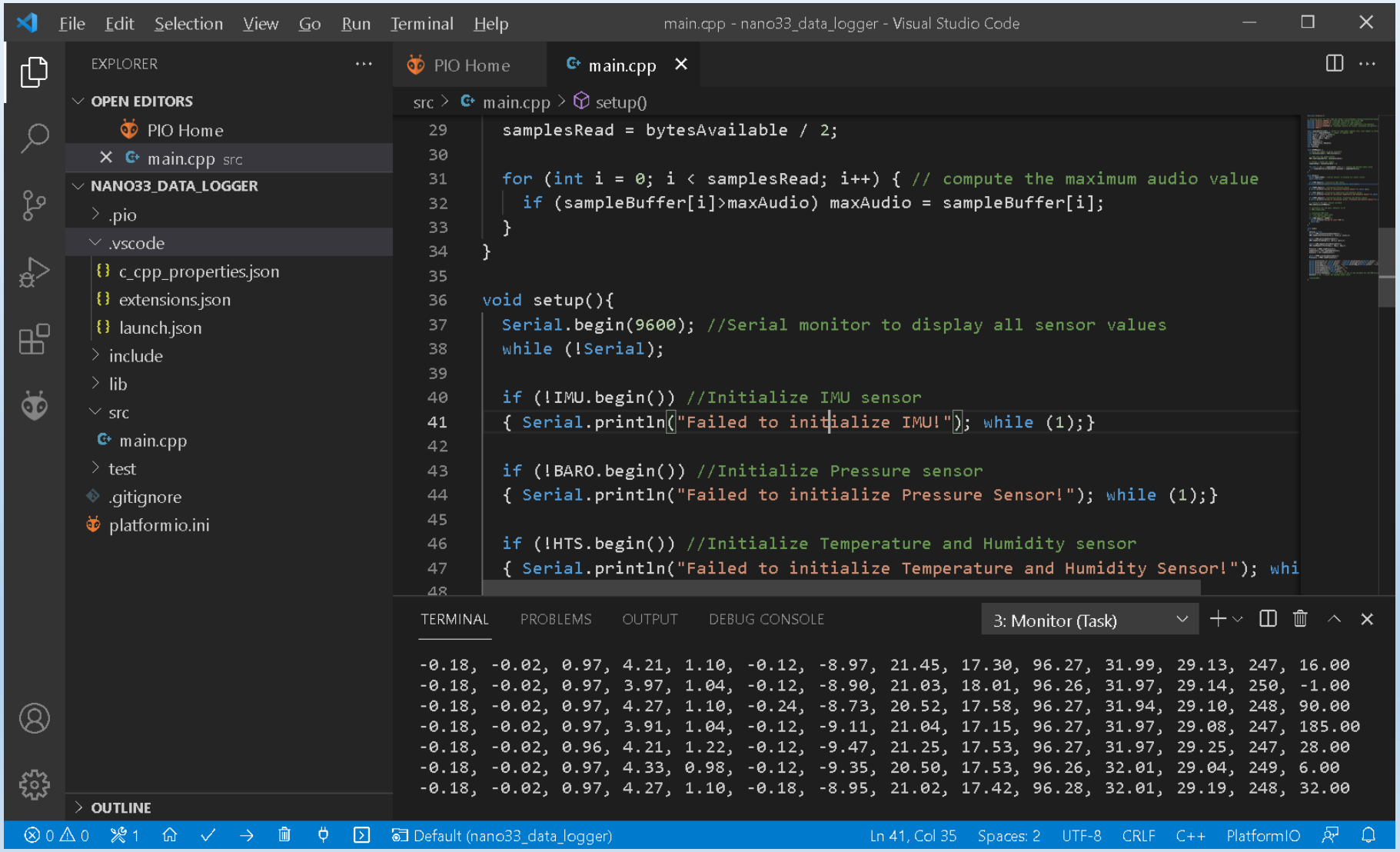
SHARE

nano33_data_logger.inc

```
// Starting point for this code was https://circuitdigest.com/microcontroller-projects/arduino-nano-33-ble-sense-board-review
#include <Arduino_LSM9DS1.h> //Include the library for 9-axis IMU
#include <Arduino_LPS22HB.h> //Include library to read Pressure
#include <Arduino_HTS221.h> //Include library to read Temperature and Humidity
#include <Arduino_APDS9960.h> //Include library for colour, proximity and gesture recognition
#include <PDM.h>

short sampleBuffer[256]; // buffer to read audio samples into, each sample is 16-bits
volatile int samplesRead=0; // number of samples read
float accel_x, accel_y, accel_z;
float gyro_x, gyro_y, gyro_z;
float mag_x, mag_y, mag_z;
float Pressure;
float maxAudio=-1;
float Temperature, Humidity;
int Proximity;
bool okToLog;

void setup(){
  Serial.begin(9600); //Serial monitor to display all sensor values
  while (!Serial);

  if (!IMU.begin()) //Initialize IMU sensor
  { Serial.println("Failed to initialize IMU!"); while (1);}
```

Double-Click Nano 33 reset pin before starting

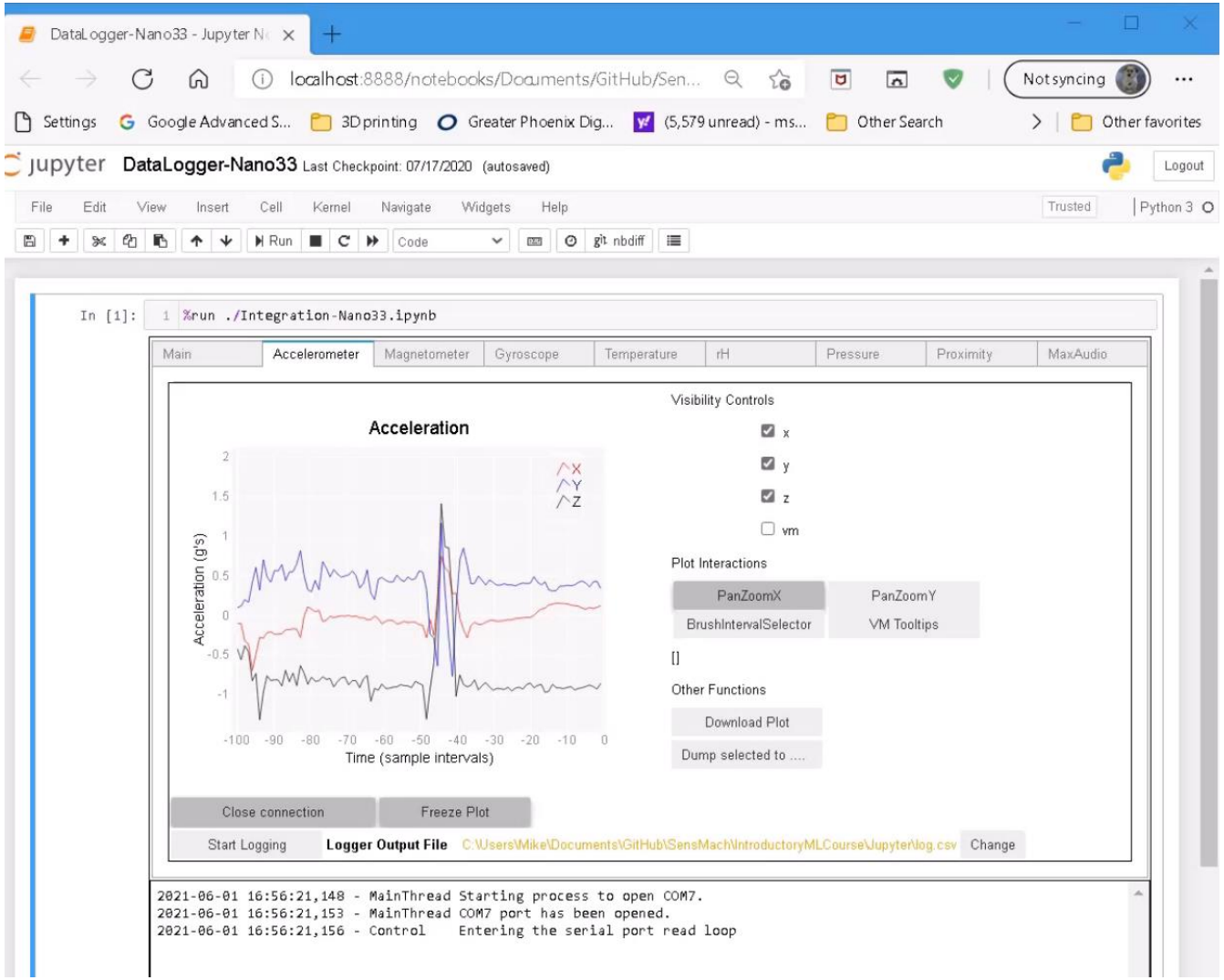
Break here for PlatformIO Demo & Walkthrough (nano33_data_logger)



Data Collection Topics

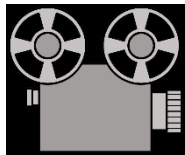
- Sensors can be noisy, and they can be impacted by other PCB components
- Ideally you would like to collect sensor samples as fast as possible
- … and at a regular (fixed) interval
- If you have multiple sensors, you would like to sample them simultaneously
- All of the above are complicated by the fact that there are huge sampling speed differences between sensor types. For example:
  - A high resolution pressure sensor may take 1 second per conversion
  - A high speed microphone might be sampling at 44KHz or higher
  - If you have more than one microphone with beamforming and filters running, things get more complicated.
- You also need to consider MCU and communications bandwidths.

Collecting Training Data



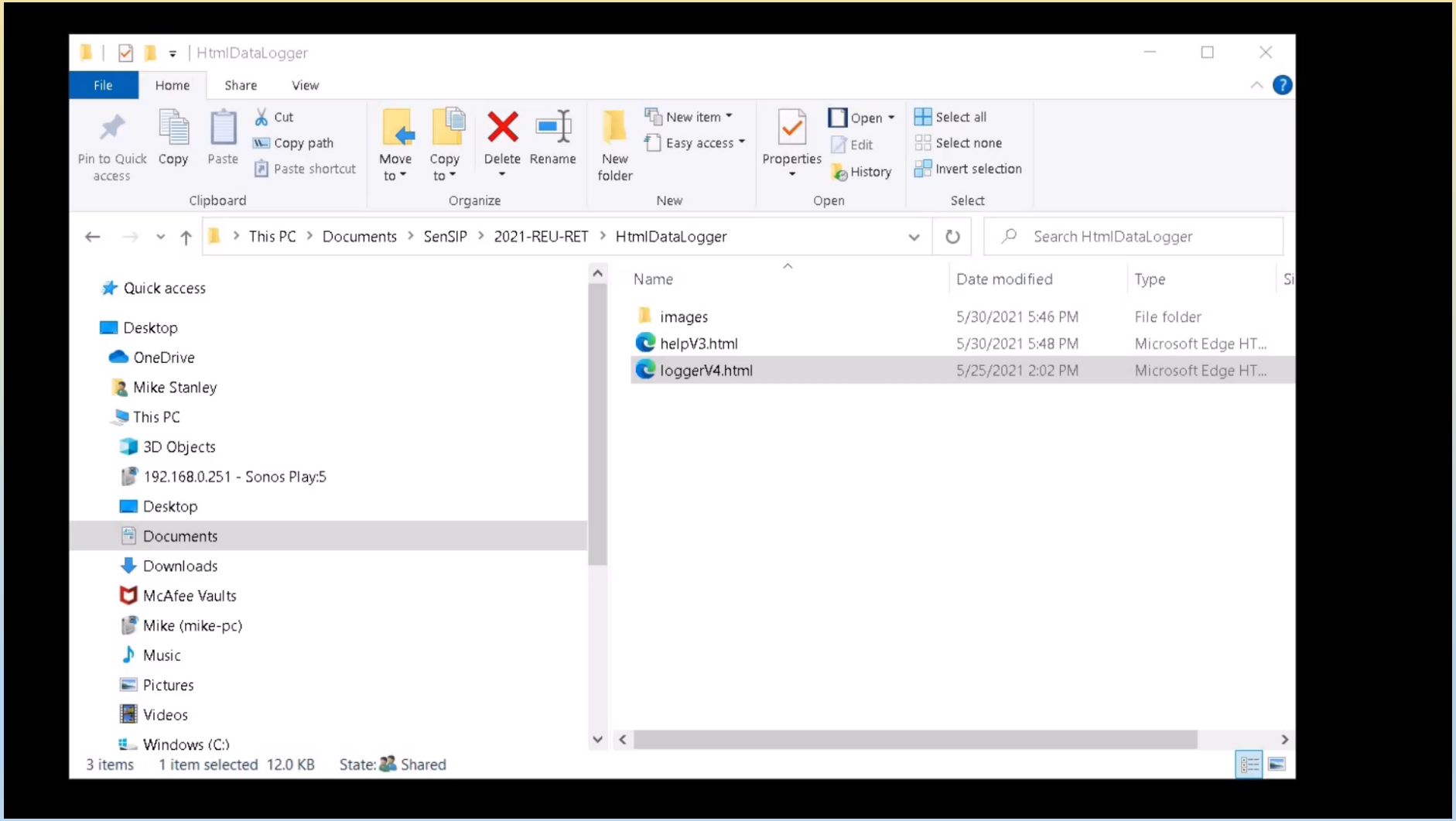
Data loggers can:

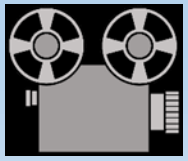
- give you the ability to visually monitor signals and look for patterns.
- run timed data logging sessions
- easily record session metadata right in the log files
- easily create graphics for reports
- upload data to cloud servers
- manage your data sets
- monitor MCU CPU & memory utilization
- manage network security and handshaking



This version is pure HTML/Javascript

- No graphics
- Depends upon experimental feature in Microsoft Edge Browser
- Requires only 2 HTML files.
- Files can be locally or server hosted





AGES 10+

ARDUINO SCIENCE JOURNAL

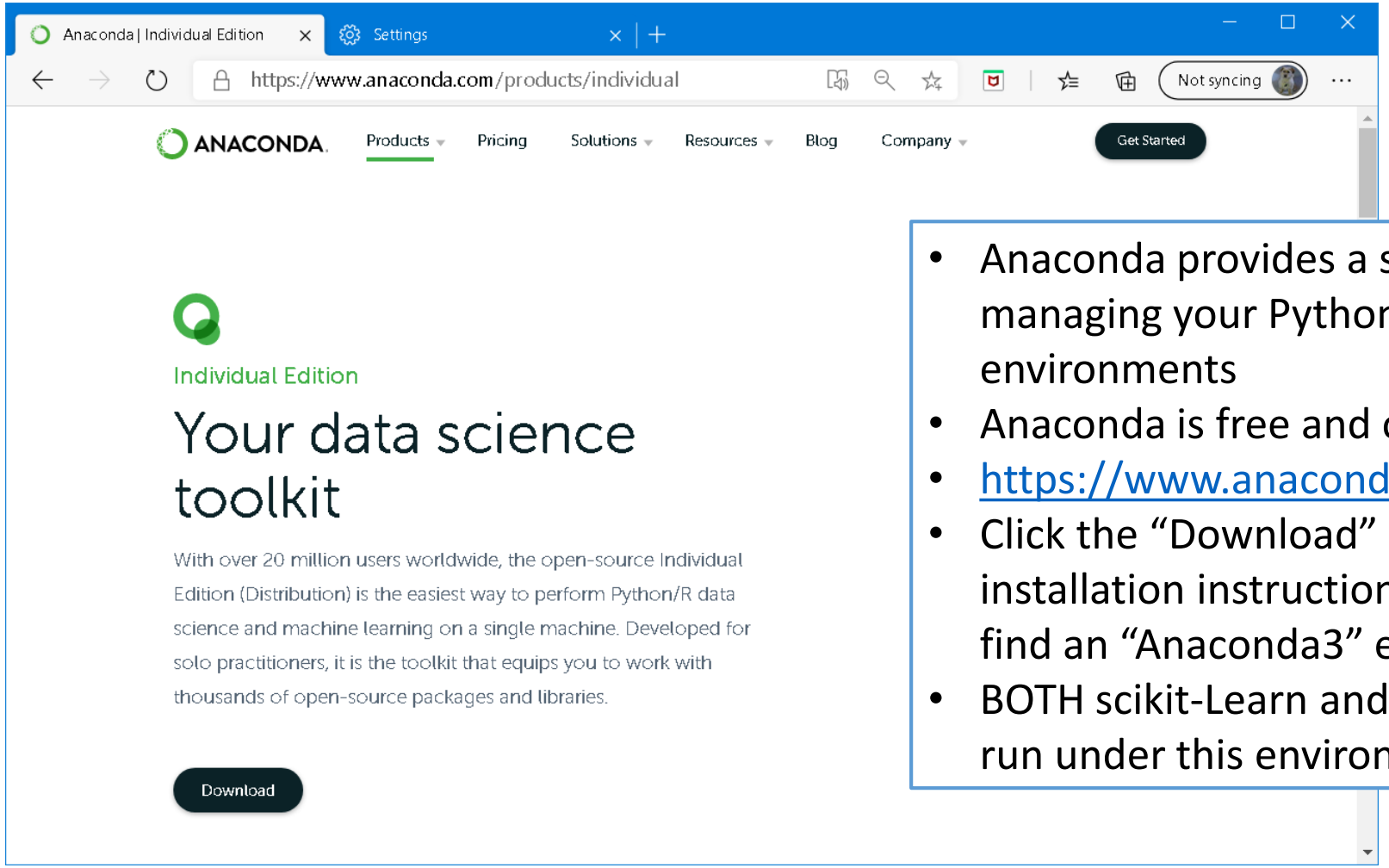
A pocket-sized science lab that encourages students to explore how the world works, record data, document observations, and experiment like a real scientist.

Download on the App Store | GET IT ON Google Play | EXPLORE IT ON AppGallery

Inspire the future generation of

If time allows…

Anaconda



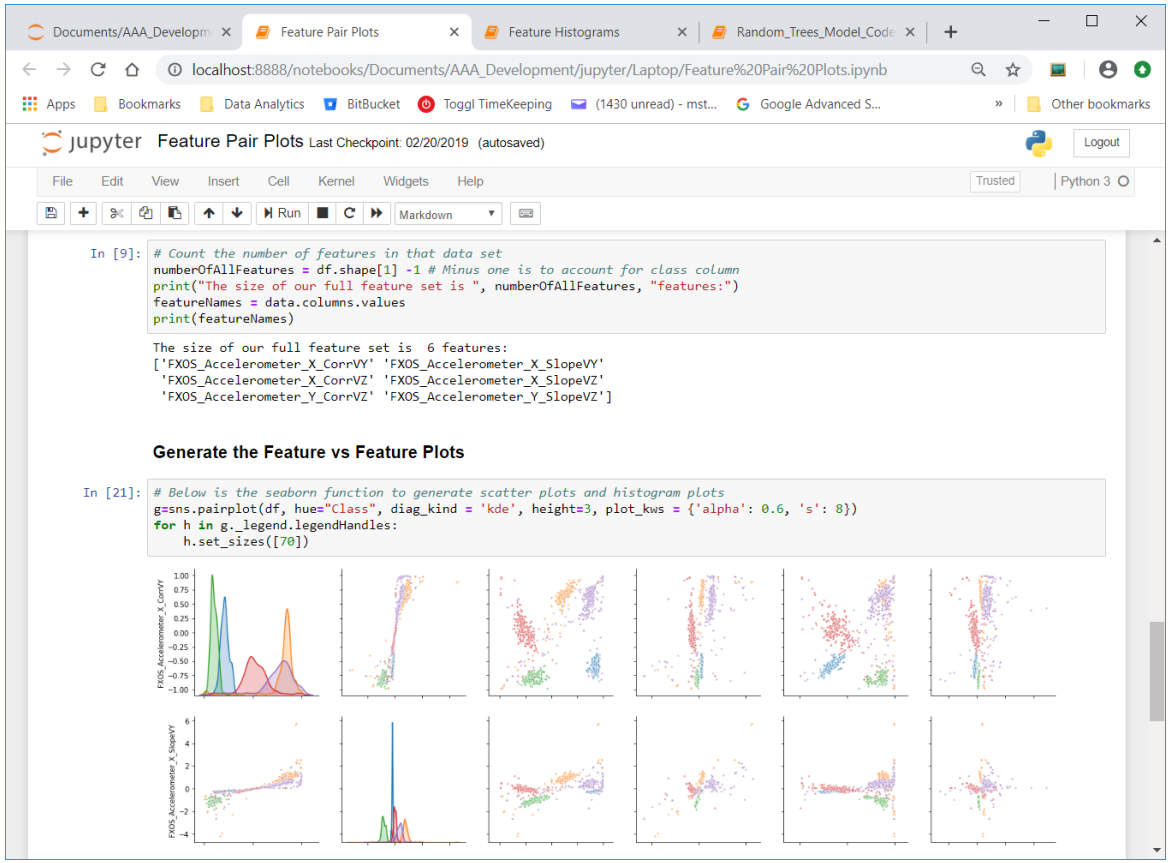
- Anaconda provides a solid framework for managing your Python, R and Jupyter environments
- Anaconda is free and open source
- https://www.anaconda.com/products/individual
- Click the “Download” button and follow installation instructions. Windows users will then find an “Anaconda3” entry in their start menu.
- BOTH scikit-Learn and Tensorflow/Keras can be run under this environment.

Anaconda

- Anaconda includes many Python packages in its default installation.
- Anaconda gives you the ability to setup multiple environments with different combinations (and versions) of libraries. You can switch between environments with a single command.
- The easiest way to tell if a 3rd party Python script or notebook will run is to try it. Python will give you clear messages indicating packages that need to be installed.
- In that case, a “pip install <package_name>” from the Python command line or a cell in a Jupyter notebook will probably do the trick. Best practice is simply to Google the package name to look up the recommended installation command.

Open Source Tools - Jupyter Notebooks

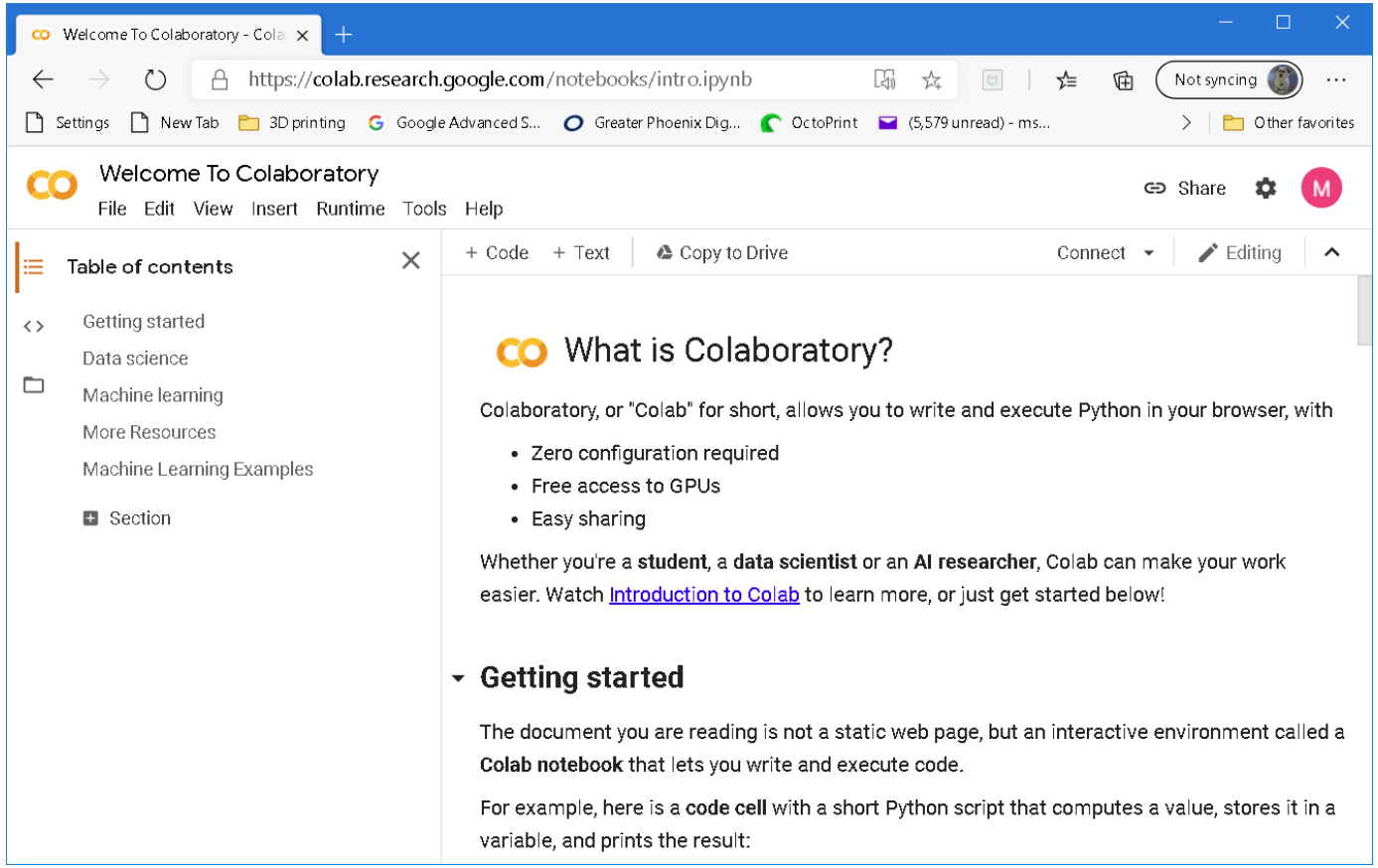
- Jupyter gives us the ability to execute R and Python scripts in a web browser environment
- Documentation, code and results are all part of a notebook
- Notebooks can easily be shared
- The Jupyter ecosystem is still being actively expanded to include additional capabilities, including an extensive set of graphical widgets.
- Python support is available for almost all non-graphical machine learning platforms



Recommendation:
https://www.anaconda.com/distribution/

Google Colab

- Execute your notebooks in the cloud.
- Google provides free access to GPUs and an easy way to share.
- Downside: No direct access to your embedded hardware



https://colab.research.google.com/

The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

https://www.python.org/

Some Python Data Types

```
In [1]: my_data = [1, 2.0, 'a', True, [1,2], (1,2)]

        print('my_data\t=', str(type(my_data)))
        for v in my_data:
            print(v, '\t=', str(type(v)))
```

```
my_data = <class 'list'>
1       = <class 'int'>
2.0     = <class 'float'>
a       = <class 'str'>
True    = <class 'bool'>
[1, 2]  = <class 'list'>
(1, 2)  = <class 'tuple'>
```

```
In [2]: for i in range(len(my_data)):
            print(my_data[i], '\t=', str(type(my_data[i])))
```

```
1       = <class 'int'>
2.0     = <class 'float'>
a       = <class 'str'>
True    = <class 'bool'>
[1, 2]  = <class 'list'>
(1, 2)  = <class 'tuple'>
```

- Lists are denoted with square brackets []
- Lists can contain any other data type
- Unlike C and C++, indentation is used to denote scope, not curly braces. White space IS significant.
- Python is designed to be easy to read.

```
In [4]: for i in range(4):
            print(i)
```

```
0
1
2
3
```

Python Dictionaries

Python dictionaries are:

- Unordered
- Changeable
- Indexed
- Written with curly brackets
- Are composed of key/value pairs

```
In [26]: thisdict = {
           'First': 'Tom',
           'Last': 'Jones',
           'DOB': 1964,
           'MaritalStatus': 'single'
         }
         thisdict['NetWorth'] = 0.05

         print(thisdict)
         print(thisdict['First'])
```

```
{'First': 'Tom', 'Last': 'Jones', 'DOB': 1964, 'MaritalStatus': 'single', 'NetWorth': 0.05}
Tom
```

Python Functions

- Because Python is loosely typed, a single function declaration can often handle multiple variable types.
- Functions can return more than one value via the use of “tuples”

In [44]:
```
# add two variables
def myAdd(x, y):
    return(x+y)

print(myAdd(2,1))
print(myAdd(2,1.0))
print(myAdd('Hello ', 'There'))
print(myAdd([2,3,4], [8,9]))
```

```
3
3.0
Hello There
[2, 3, 4, 8, 9]
```

In [35]:
```
# Function for adding 2 wide tuples
def myAddTuple(x, y):
    return x[0]+y[0], x[1]+y[1]

x, y = myAddTuple((2,3), (4,4))
print(x, y)
```

```
6 7
```

Python is Object-Oriented

```
In [28]: class object():

             def __init__(self, name):
                 self.name = name;

             def identity(self):
                 return(self.name)

         obj = object('Friend')
         obj.identity()
```

```
Out[28]: 'Friend'
```

You could derive the object class from another by specifying the parent class here

The class name is “object”

This is the constructor:

- ‘self’ is a pointer to the object
- ‘name’ is a parameter to the constructor
- Self.name is a local variable used to store the name of this object

‘identity’ is a just a function we defined.
It always has the ‘self’ pointer as an implied first parameter.

Python Libraries

- Much of the functionality in Python comes from the wealth of function libraries available.
- There is a two step process to use them:
  - Install the library in your local environment via pip or conda commands
  - Import the functions you need at the top of your program

1 Standard Module Imports

In [1]:
```
from sklearn import svm, mixture, neighbors, tree, gaussian_process, ensemble, neural_network, naive_bayes, \
    discriminant_analysis, metrics, preprocessing
import numpy as np
import pandas as pd
import bqplot as bq
import ipywidgets as widgets
from skimage import measure
import scipy.stats as stats
import math
import inspect
import matplotlib.pyplot as plt
from sklearn.gaussian_process.kernels import RBF
import traceback
```

Use the NumPy library for matrix math

```
In [4]: import numpy as np
        X = np.array([[1,2,3],[4, 5, 6],[7,8,9]])
        X

Out[4]: array([[1, 2, 3],
               [4, 5, 6],
               [7, 8, 9]])

In [5]: Y=np.arange(1,4)
        Y

Out[5]: array([1, 2, 3])
```

```
In [6]: 2*X

Out[6]: array([[ 2,  4,  6],
               [ 8, 10, 12],
               [14, 16, 18]])
```

```
In [7]: X+Y

Out[7]: array([[ 2,  4,  6],
               [ 5,  7,  9],
               [ 8, 10, 12]])
```

```
In [8]: X-Y

Out[8]: array([[0, 0, 0],
               [3, 3, 3],
               [6, 6, 6]])
```

```
In [9]: X/Y

Out[9]: array([[1. , 1. , 1. ],
               [4. , 2.5, 2. ],
               [7. , 4. , 3. ]])
```

```
In [10]: X*Y # Element-wise multiplication

Out[10]: array([[ 1,  4,  9],
                [ 4, 10, 18],
                [ 7, 16, 27]])
```

```
In [11]: X@Y # Matrix multiplication

Out[11]: array([14, 32, 50])
```

Y is treated as 3X1

```
In [12]: Y@X

Out[12]: array([30, 36, 42])
```

Y is treated as 1X3

```
In [13]: X>4

Out[13]: array([[False, False, False],
                [False,  True,  True],
                [ True,  True,  True]])
```

```
In [14]: X[X>4]

Out[14]: array([5, 6, 7, 8, 9])
```

```
In [13]: Z=X
         Z

Out[13]: array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])
```

```
In [14]: Z+=1
         Z

Out[14]: array([[ 2,  3,  4],
                [ 5,  6,  7],
                [ 8,  9, 10]])
```

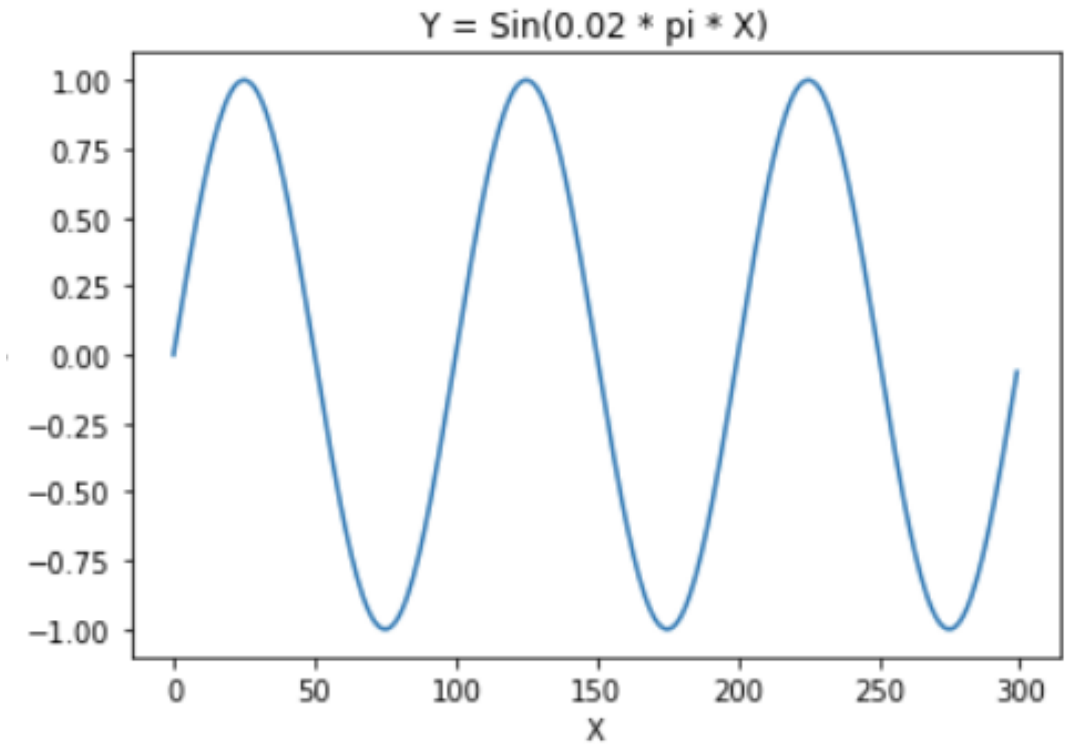
```
In [15]: X

Out[15]: array([[ 2,  3,  4],
                [ 5,  6,  7],
                [ 8,  9, 10]])
```

```
In [19]: X[0,0]+=20
         X

Out[19]: array([[22,  3,  4],
                [ 5,  6,  7],
                [ 8,  9, 10]])
```

```
In [20]: Z

Out[20]: array([[22,  3,  4],
                [ 5,  6,  7],
                [ 8,  9, 10]])
```

Matplotlib is the most popular plotting library

```
In [39]: import matplotlib.pyplot as plt
         import numpy as np
         import math
         X = np.arange(0, 300)
         Y = np.sin(0.02 * math.pi * X)
         plt.plot(X, Y)
         plt.title('Y = Sin(0.02 * pi * X)')
         plt.xlabel('X')
         plt.ylabel('Y')
```



‘bqplot’ is a great 2-D plotting library with hooks for developing interactive graphics.

Build Graphical Interfaces with the IP Widgets Library

```
In [1]: import ipywidgets as widgets
        from IPython.display import display
        w = widgets.IntSlider(description='w', min=-10, max=50, step=1)
        display(w)
```

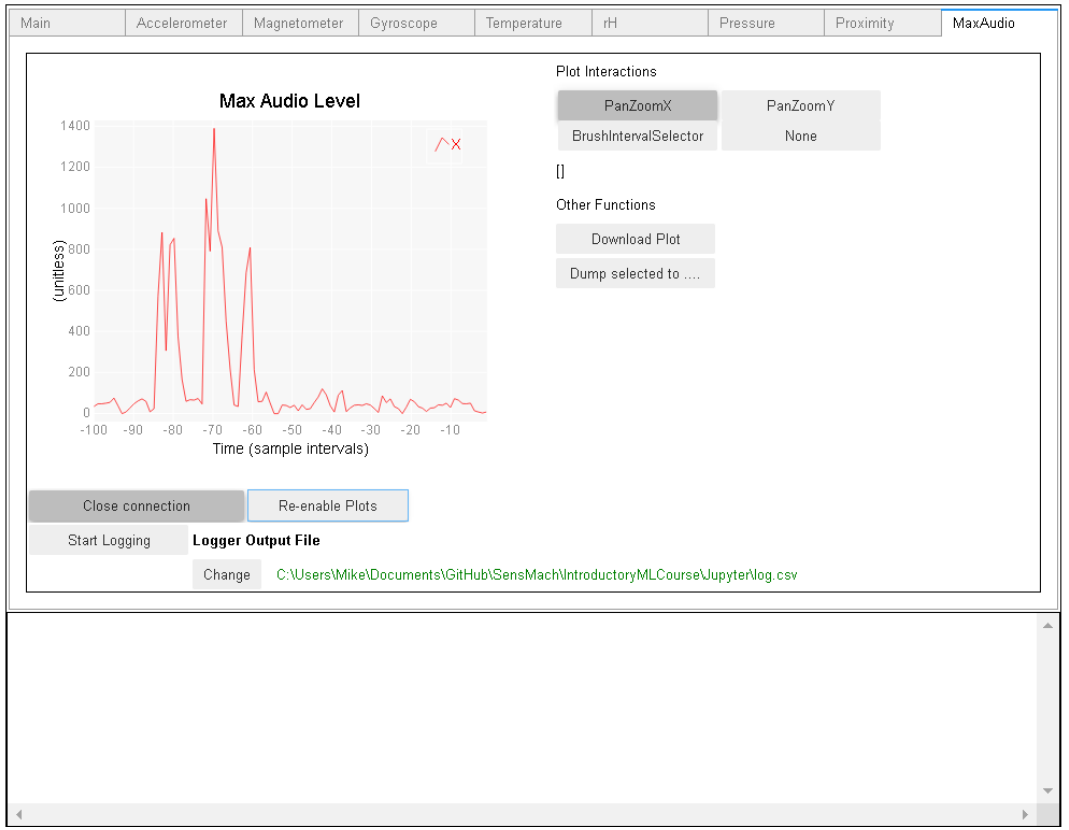
w 10

```
In [2]: w.value
```

```
Out[2]: 10
```

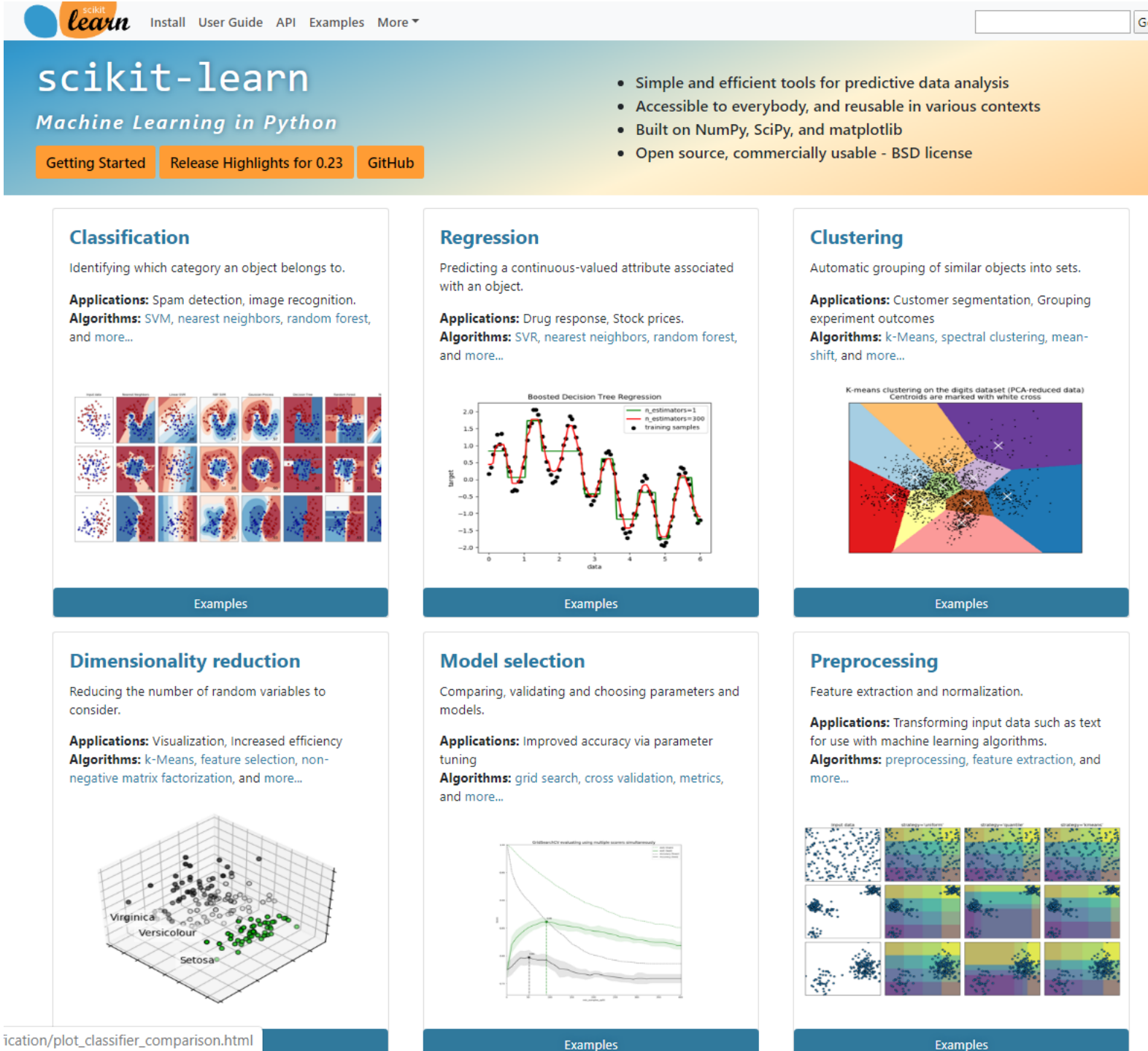
```
In [4]: widgets.ToggleButton(
            value=False,
            description='Toggle Button',
            disabled=False,
            button_style='',
            tooltip='my button',
            icon='check'
        )
```

✔ Toggle Button



Classical ML

- **scikit-learn** is easily the most accessible and well-documented toolkit available for classical machine learning.
- It can be used in a standard Jupyter environment.
- scikit-learn.org contains a huge amount of huge collection of tutorials and examples. The latter are easily downloaded and run on your laptop.



Summary/Wrapup

- We've learned how to code a simple C++ program and embedded it into an Arduino environment.
- We've looked at both Windows and Cloud-based environments
- We've seen how it's possible for Jupyter/Python code to interact with your hardware
- We've learned a bit about Python coding

Next time: Detailed design review of a fruit classification system