Goal:

This document was written to provide the reader with an overview for the P.O.T.A.T.O software system hereby referred as the Matrix. The communication architecture, functions and usage of the user interfaces are covered in this document.

Definitions and Names:

The UI touch screen can be referred to as morpheus but will be referred to in the code and this document as the touch screen.

The serial terminal can be referred to as the red pill but will be referred to in the code and this document as the serial terminal and UI.

The physical box the components sit in can be referred to as Nebuchadnezzar but will be referred to in the code and this document as the box.

The PCB and project boards that interface between the optical, electrical and software subsystems can and is referred to as NEO.

System architecture:

The software is set up to be integrated with the original three wavelengths of 850, 1310 and 1550. The software does not require all laser wavelengths to be functional in order to do testing with one of the wavelengths. Everytime the system is set up the touch screen must be calibrated and then it will start in touch screen mode. To get to serial mode you navigate to the serial mode button which will be covered later. For reference the overall system architecture is shown below in Figure 1.

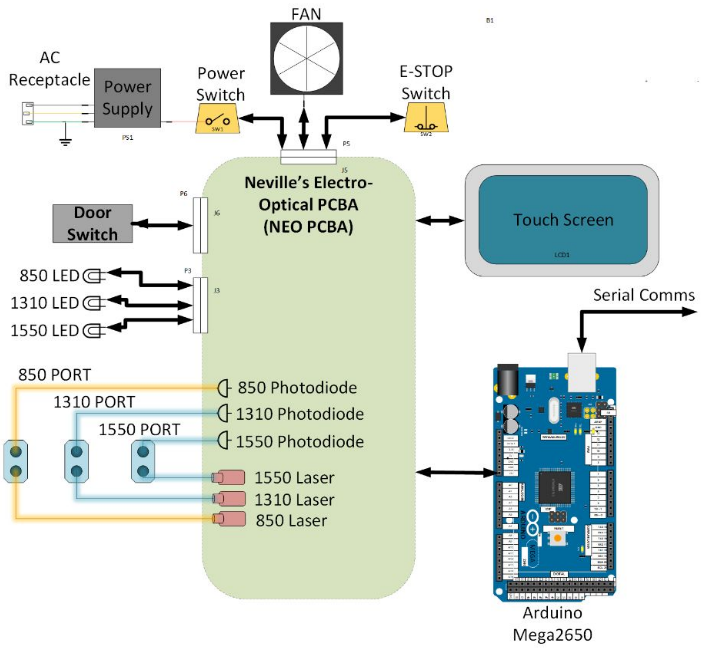
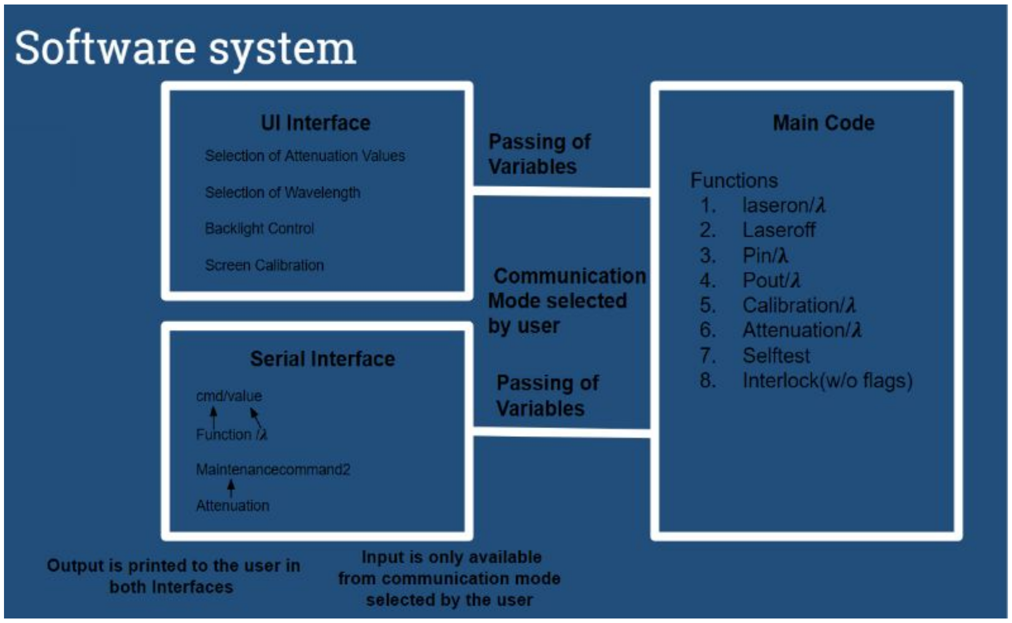


Figure 1: System Architecture

The Software Architecture, and the different communication modes is described in Figure 2 .



Things to note in this diagram is the switching of communication modes by the user and the difference between input and output regarding communication modes. Output is printed to the user in both interfaces while input is only available from the communication mode currently selected by the user. Also note the make up of commands in the serial interface.

The overall flow chart of the software is shown in Figure 3.

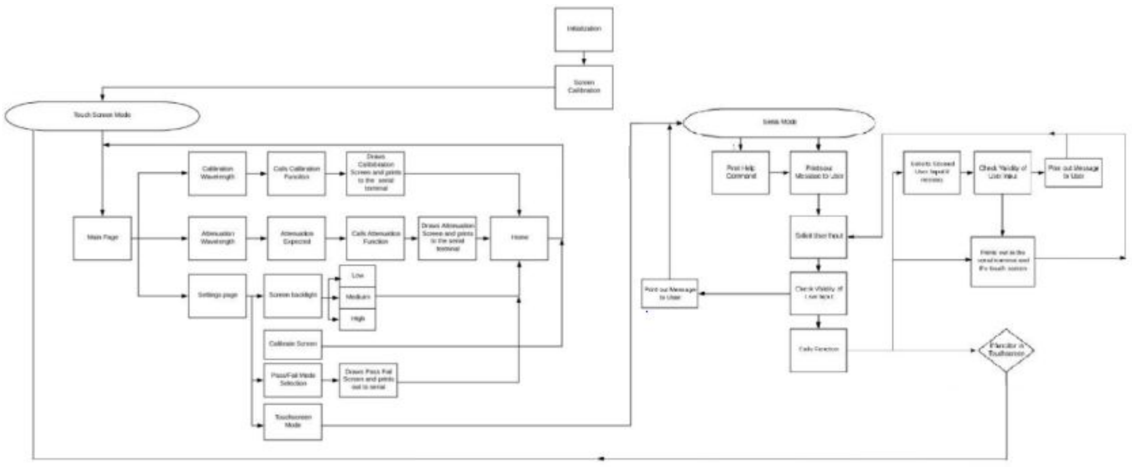
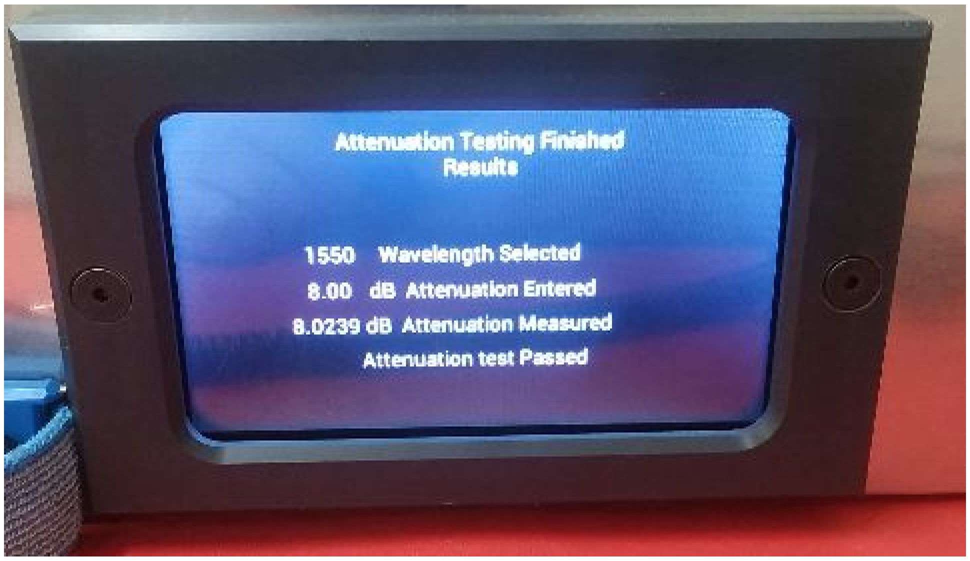

Initialization
Screen Calibration
Touch Screen Mode
Calibration Wavelength
Calls Calibration Function
Draws Calibration Screen and prints to the serial terminal
Main Page
Attenuation Wavelength
Attenuation Expected
Calls Attenuation Function
Draws Attenuation Screen and prints to the serial terminal
Home
Settings page
Screen backlight
Low
Medium
High
Calibrate Screen
Pass/Fail Mode Selection
Draws Pass Fail Screen and prints out to serial
Touchscreen Mode
Serial Mode
Print Help Command
Print out Message to User
Solicit User Input
Check Validity of User Input
Calls Function
Print out Message to User
Check Validity of User Input
Print out Message to User

Figure 3: Software Flowchart

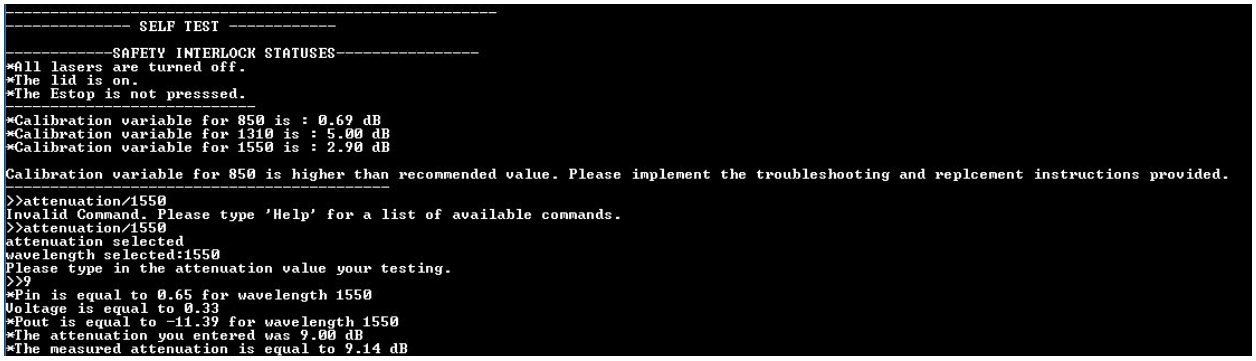
Figure 4 below shows the touchscreen interface.


Attenuation Testing Finished
Results

1550 Wavelength Selected
8.00 dB Attenuation Entered
8.0239 dB Attenuation Measured
Attenuation test Passed

Figure 4: Touchscreen

Figure 5 shows the serial Interface.


```
-------------------------------------------------------
-------------- SELF TEST ------------

------------SAFETY INTERLOCK STATUSES----------------
*All lasers are turned off.
*The lid is on.
*The Estop is not presssed.
----------------------------
*Calibration variable for 850 is : 0.69 dB
*Calibration variable for 1310 is : 5.00 dB
*Calibration variable for 1550 is : 2.90 dB

Calibration variable for 850 is higher than recommended value. Please implement the troubleshooting and replcement instructions provided.
-----------------------------------------
>>attenuation/1550
Invalid Command. Please type 'Help' for a list of available commands.
>>attenuation/1550
attenuation selected
wavelength selected:1550
Please type in the attenuation value your testing.
>>9
*Pin is equal to 0.65 for wavelength 1550
Voltage is equal to 0.33
*Pout is equal to -11.39 for wavelength 1550
*The attenuation you entered was 9.00 dB
*The measured attenuation is equal to 9.14 dB
```

Figure 5: Serial Interface

Code:

The code itself was written in the Arduino IDE and was tested in the teraterm terminal. The code functions are separated into five different types. They are the Main, FTMI, serial, UI and UI Serial. The code is organized so that all FTMI code (code that was written entirely for the screen by the manufacturer) has the prefix of FTMI_ in the tab. Serial code (code that controls the serial terminal or just base functions of the code, pout, pin, attenuation.llll) has the prefix of S_ in the tab. UI code (code that controls the drawing of the screens that functionality of user input and output for the screen) has the prefix of UI_ in the tab. The SUI codes (code that draws on the screen when functions are called in the serial terminal) have the prefix of SUI_ in the tab. These prefixs are shown below in Figure 6.

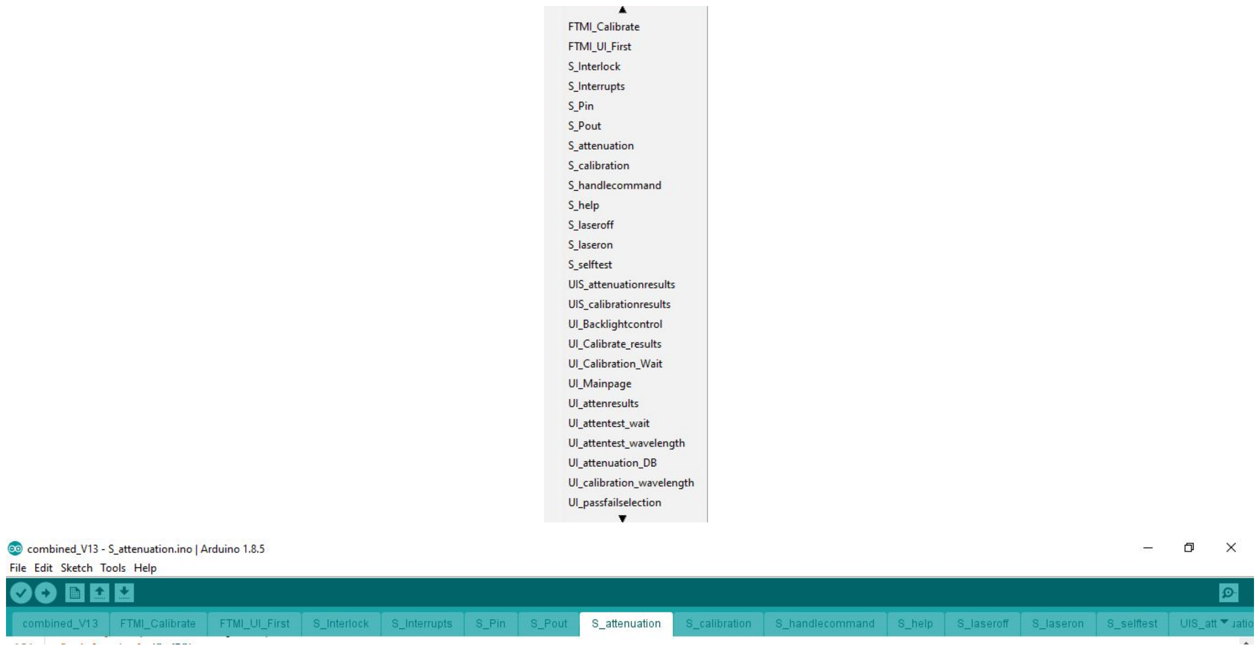


Figure 6: Examples of naming convention for tabs

<u>List of Functions:</u>

int16_t BootupConfigure()
int32_t Dec2Ascii(char *pSrc,int32_t value)
void setup()
void Calibrate(): Handles the screen calibration “please touch the dot”
void Touch(): Draws the first Main Page
void interlock(): displays the interlock statuses
void Estop_ISR():Estop ISR
void LID_ISR(): LID ISR
String Pin(String value): Serial Pin function
String Pout(String value): Serial Pout function
int attenuation(int value): Serial Attenuation function
int calibration(int value): Serial Calibration function
void handlesecondstring(): gets the second user input from the serial command line
void readCommand(void) : reads user input from the serial terminal
void handleCommand(void): list of if statements serial uses to select function
void clearBuffer(void)// clears the buff to stop overflow
void help()// prints out the help command
void laseroff()// turns the laser off
int laseron(int value)// turns the laser on
void selftest()// runs at the beginning
void SUIattenresults()// draws the attenuation results on the screen
void SUI_calibrationresults ()// calibration results

```
void backlightcontrol()// controls the backlight of the touch screen
void Calibrationresults()// displays the calibration
void Calibrationpage()// calibration wait page
void Mainpage()// home page
void PassFail()// sets the pass fail range
void Testingpage()// waiting page
void Testingoptions()// attenuation testing
void Attenuationselect()// select attenuation to test
void calibrationwavelength(void)// select calibration  wavelength
void UIpassfailselection()// has the user pick pass fail for the touch screen
void passfailsettings()// draws the pas fail screen selection
void Touchscreen(void)// goes to touchscreen mode
void serialscreen(void)// draws the serial screen
void Settingspage()// draws the setting screen
```

Button Placement:

Some of the buttons in the UI screen might look like they are in questionable place, for example the home button might leave the user asking why is the home button in the top right of the screen. The reason is that this screen does not have any code to stop the user from double pressing two buttons accidentally in a row, like hitting attenuation and accidentally hitting 850 immediately. Therefore all buttons were lined up in way to limit the user from accidentally double pressing. This problem was found and tested during software testing.

Additional code in UI screen software sections that look unnecessary:

Due to the lack of proper documentation of the screen library most of the screen coding was done through trial and error over multiple of hours. While I will try to encapsulate the knowledge I have gained hear in this document there are some instances where it was safer to leave in parts of code from other screen even if unused(such as registering touch values for screens that don't have user input) in order to minimize troubleshooting, and untraceable errors. While not good coding practice it was effective for risk mitigation concerning screen code documentation. Because of this you will notice that most of the screen code follows the same general format of establishing variable, creating a holding loop, printing text and assigning tag values, and positions to buttons and having the user change the state of the screen using the tag values from the buttons.

How to get the Arduino software and the TeraTerm software

Download the latest version of the Arduino IDE Here
https://www.arduino.cc/en/Main/Software

Download the latest version of Tera Term software Here
https://ttssh2.osdn.jp/index.html.en

Website of the screen manufacturer(NHD-4.3CTP-SHIELD-L)
https://www.mouser.com/new/newhavendisplay/newhaven-tft-arduino-shield/
-data sheet can be found here

Github of example code from the screen manufacturer
https://github.com/NewhavenDisplay/FTDI_FT801

FTMI Chip (video processing chip) data sheet available here
http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT801.pdf

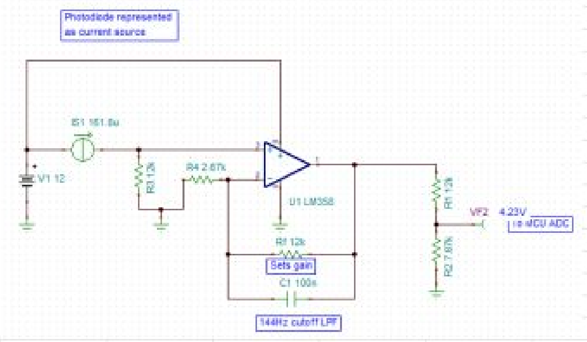
How to get the magic numbers in the equations

1550 wavelength

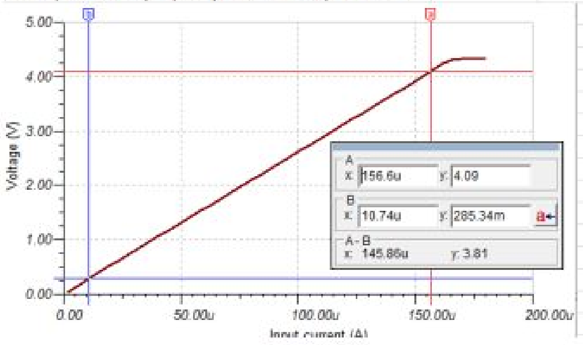
Pin

With a loopback plugged in the Voltage read from the Arduino was 4.23V
The simulation was used to find the current coresponding to this voltage (161.8 uA)
The optical power was read using the power meter (1.14 mW)
The responsivity was calculated using these two numbers

I_PD	Po	Responsivity
161.8	1.14	**0.1419298246**

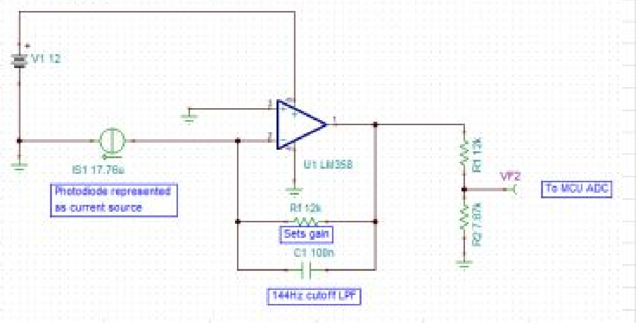


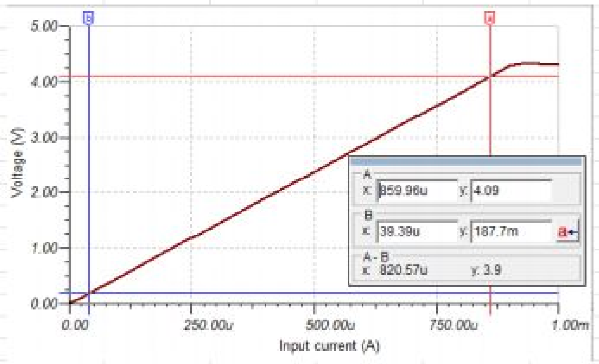
The current was swept from 1.6 uA to 180 uA to simulate the full range of optical power anticipated
The response of the OpAmp was plotted and the slope extracted



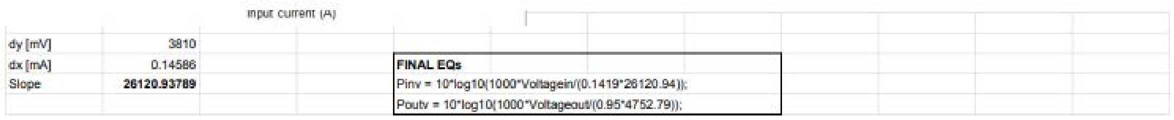
Pout

The responsivity of the Pout Photodiode was assumed to be 0.95 based on the datasheet
The slope of the OpAmp response was found by sweeping the PD current from 2.26 uA to 1 mA
Responsivity
0.95





dy [mV]	3900
dx [mA]	0.82057
Slope	**4752.79379**

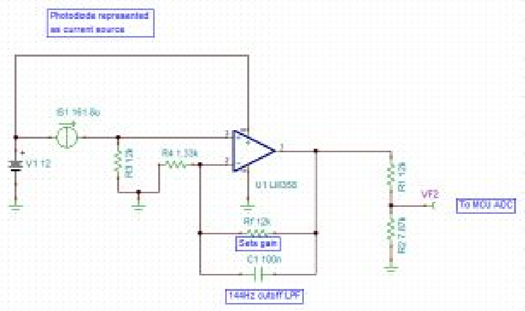


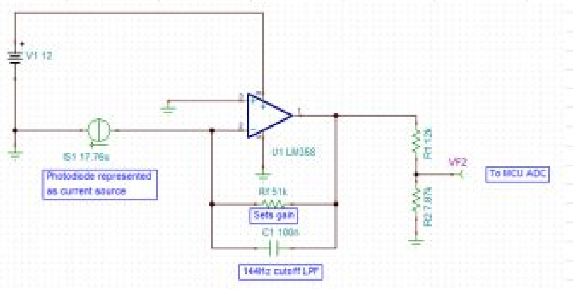
Input current (A)

dy [mV]	3810		**FINAL EQs**
dx [mA]	0.14586		Pinv = 10*log10(1000*Voltagein/(0.1419*26120.94));
Slope	**26120.93789**		Poutv = 10*log10(1000*Voltageout/(0.95*4752.79));

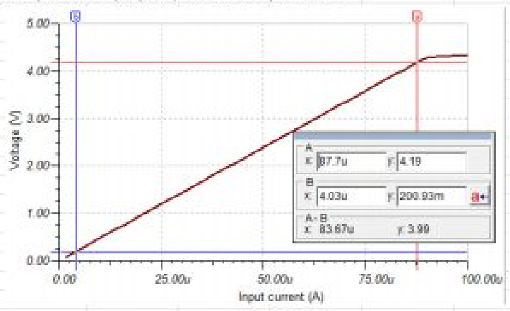
1310 wavelength

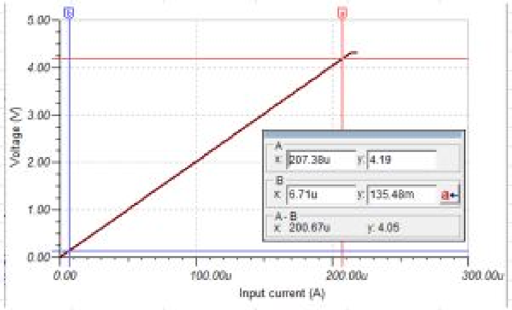
Pin	Pout
Responsivity	The responsivity of the Pout Photodiode was assumed to be 0.9 based on the datasheet
0.2116	The slope of the OpAmp response was found by sweeping the PD current from 1 uA to 220 uA
	Responsivity
	0.9





The current was swept from 1 uA to 100 uA to simulate the full range of optical power anticipated
The response of the OpAmp was plotted and the slope extracted





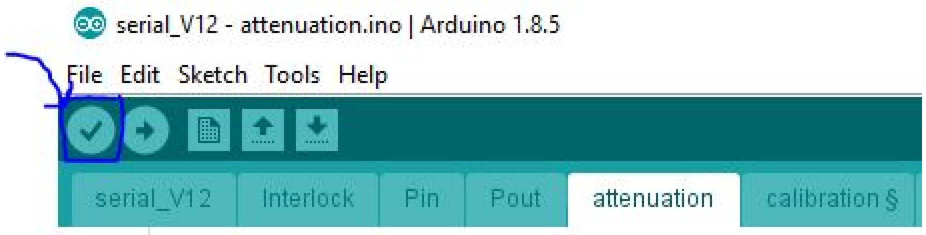
dy [mV]	3990	dy [mV]	4050
dx [mA]	0.08367	dx [mA]	0.20067
Slope	**47687.34313**	Slope	**20182.389**

FINAL EQs
Pinv = 10*log10(1000*Voltagein/(0.2116*47687.34313));
Poutv = 10*log10(1000*Voltageout/(0.9*20182.389));

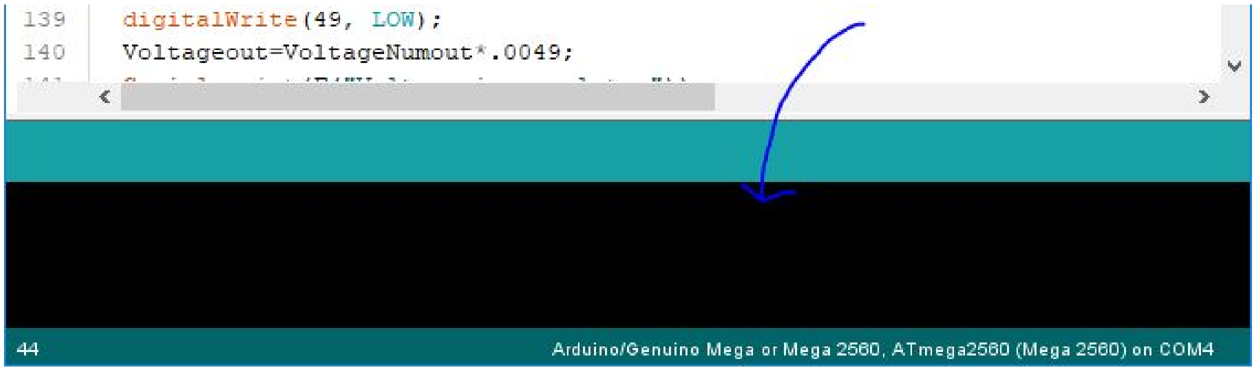
850 wavelength was never gotten too due to component failure

<u>How to compile and run the code.</u>

First open The Arduino code in the IDE, make sure all the appropriate files are there, then hit the compile button in the Top right corner


serial_V12 - attenuation.ino | Arduino 1.8.5
File Edit Sketch Tools Help
serial_V12
Interlock
Pin
Pout
attenuation
calibration §

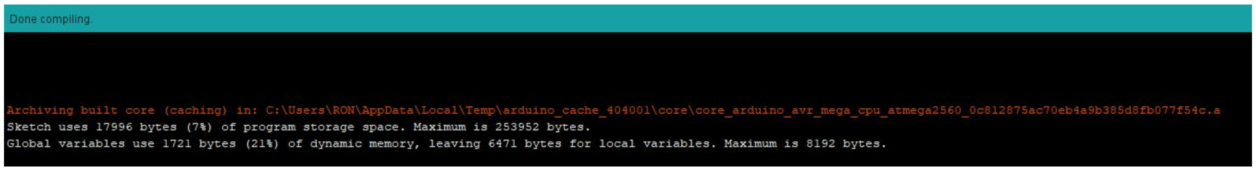
Any errors will appear in the box below.



```
digitalWrite(49, LOW);
Voltageout=VoltageNumout*.0049;
```

```
44                                    Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM4
```

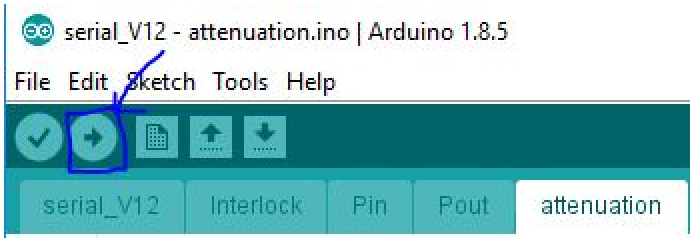
The box will say successfully compiled.



```
Done compiling.

Archiving built core (caching) in: C:\Users\RON\AppData\Local\Temp\arduino_cache_404001\core\core_arduino_avr_mega_cpu_atmega2560_0c812875ac70eb4a9b385d8fb077f54c.a
Sketch uses 17996 bytes (7%) of program storage space. Maximum is 253952 bytes.
Global variables use 1721 bytes (21%) of dynamic memory, leaving 6471 bytes for local variables. Maximum is 8192 bytes.
```

If no errors appear hit the upload button


serial_V12 - attenuation.ino | Arduino 1.8.5
File Edit Sketch Tools Help
serial_V12
Interlock
Pin
Pout
attenuation