Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

User's Manual

Integrated Laser and Electronic Model for Photoacoustic Neural Stimulation

Client: Professor Chen Yang
(617) 358-4837
cheyang@bu.edu

By
Team 19
Integrated Laser

Team Members:
Tian Huang tianh1@bu.edu
Ian Lee ianxlee@bu.edu
Raphael Mok rmok@bu.edu
Viet Nguyen vietng99@bu.edu
Mateus Toppin mtoppin@bu.edu

Table of Contents

Executive Summary (Tian Huang)

The integrated laser and electronic model is a function generator that will generate parameterized square waves that will be used to pulse a laser for neural stimulation purposes. The product has a hardware electronic model that utilizes Teensy 4.1 to generate the required pulse and a software application for controlling the input parameters. These input parameters include frequency in Hertz, duty cycle in millisecond, and pulse width modulation sequence. The goal of this project is to make the final product significantly smaller, less expensive, more sturdy for shipping, and with additional technical features. This is an overall improvement and a replacement to the older, bigger function generator model that is currently employed in the lab office. The main important features are the software interface for parameter inputs with an emphasis on user friendliness. The software application will be able to generate the desired pulse by connecting the Teensy hardware to a computer device. This will be easier to program the laser and help collect real-time data. In short, the finalized product will be a packaged Teensy served as a function generator along with a software application. It will be easy to use, setup, small, and cost effective.

1. **Introduction** (Tian Huang)

Neurological disorders cause a tremendous amount of problems for individuals and society. Such disorders include epilepsy, learning disabilities, neuromuscular disorders, depression, Parkinson, retina degeneration, etc. According to the World Health Organization, about 1 in 3 people are affected by neurological conditions that lead to illness and disability. Moreover, JAMA Neurology concluded a rough estimation of about 100 million Americans were affected by at least 1 neurological disorder. Especially with the more recent studies conducted by JAMA, the data suggested neurological disorders like depression are on the rise. On the other hand, the most common treatments for such disorders are typically pharmaceutical drugs, medicines, and stimulation.

Speaking of treatment, the project's client, Yang Lab, specializes and focuses on developing a unique neural stimulation technology that utilizes photoacoustic effects. In photoacoustics technology, a device driven by a pulsed laser converts the light source from the laser into a pulsed ultrasound wave. The ultrasound wave is then used to stimulate targeted neurons, brain, and retina. Such technology serves as a fundamental tool to understand how the brain functions and offers a non-drug treatment for neurological disorders. In short, the photoacoustic neural stimulation process that Yang Lab specializes in can be broken down into three steps. First, a high intensity light energy like a laser is needed to be generated. Second, the light source is used to convert into ultrasound waves. Finally, the device used the ultrasound wave to stimulate the neurons.

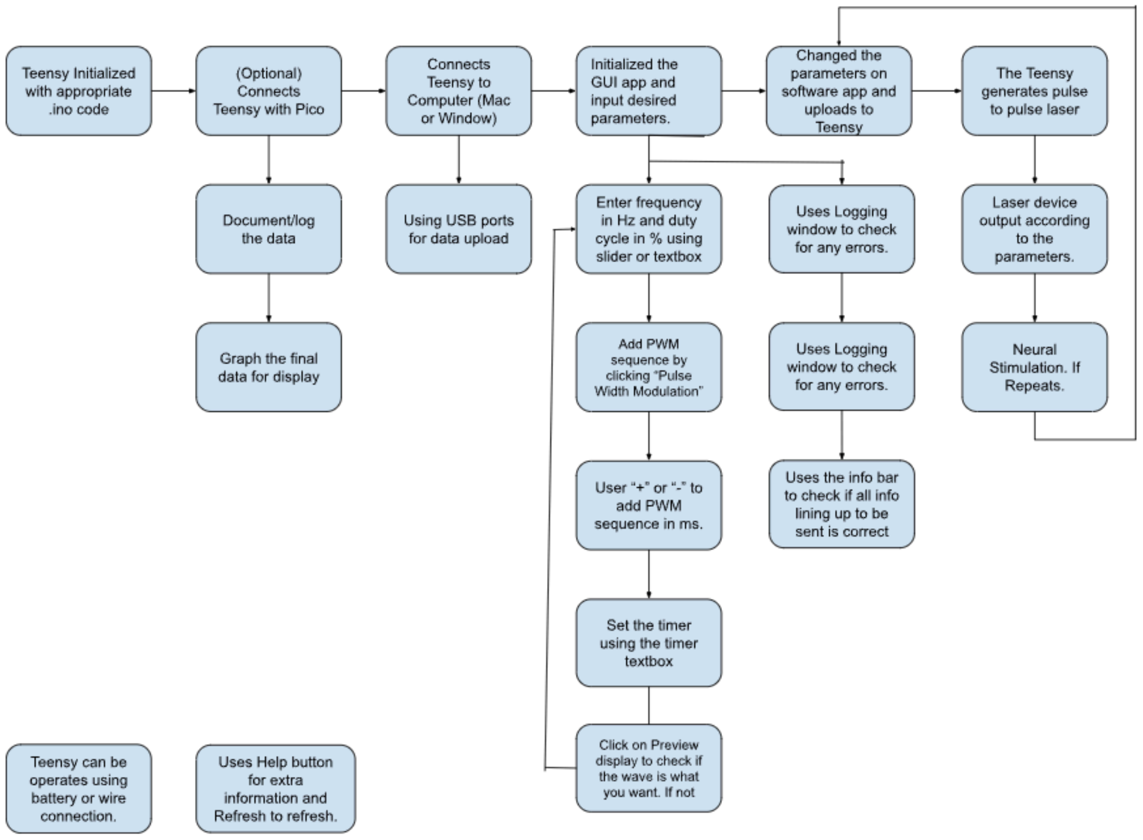
The problem Yang Lab currently faces is that their function generator devices are too big and lack software interface. They proposed a development for an electronic model that is compact, user-friendly, and cost-effective that will be compatible with the laser in the lab for outputting programmed laser to other lab devices. There are also few expected requirements such as the final product of an integrated box be no more than 0.5 feet in dimension and be compact for shipping purposes. In short, the final product is supposed to be a small box that can output square waves through the control of a software application.

In the user's perspective, this means the user will primarily be interacting with the software application for generating the pulse. In fact, the software inclusion will be the main reason for the replacement of the old lab function generator. The user can enter three main parameters, frequency, duty cycle, and pulse width modulation sequence that will generate the appropriate wave. Moreover, the software application will also include special features and various functionalities such as preview window with , auto port detection, a timer that will terminate all pulses once it is up, data log that keep track of the application history and much more. The device may even include an external trigger that can automatically send the desired wave if the incoming signal is greater than a certain voltage level.

As for issues regarding safety and security, there is none since the voltage and current level are not high enough to cause electric danger. Moreover, the enclosure is designed with safety in mind with insulators that eliminate any leakage. Overall, the objective is to design a device along with a software application using Teensy 4.1 that could generate the appropriate wave to replace the lab function generator. The remaining part of this document will first discuss system overviews like installation, setup, support, user interface, and block diagram. Right afterward, general operation of the device and any abnormal operations will be mentioned in detail. This part will also discuss some minor safety issues. Then it is followed up with technical background to discuss technical approach and relevant engineering standards. Finally, the document will end with cost breakdown and appendices.

2 System Overview and Installation

2.1 Overview block diagram (Tian Huang)

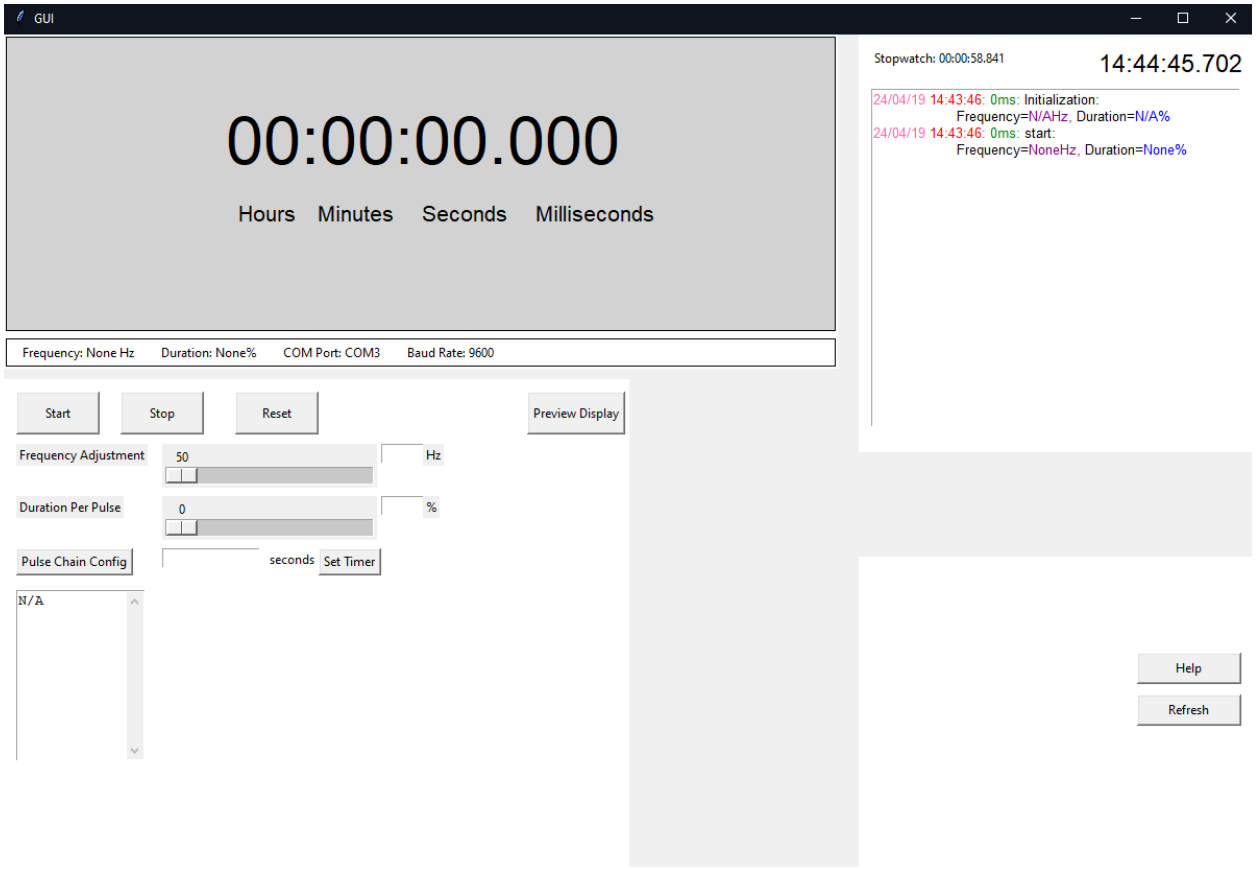

Teensy Initialized with appropriate .ino code
(Optional) Connects Teensy with Pico
Connects Teensy to Computer (Mac or Window)
Initialized the GUI app and input desired parameters.
Changed the parameters on software app and uploads to Teensy
The Teensy generates pulse to pulse laser
Document/log the data
Using USB ports for data upload
Enter frequency in Hz and duty cycle in % using slider or textbox
Uses Logging window to check for any errors.
Laser device output according to the parameters.
Graph the final data for display
Add PWM sequence by clicking "Pulse Width Modulation"
Uses Logging window to check for any errors.
Neural Stimulation. If Repeats.
User "+" or "-" to add PWM sequence in ms.
Uses the info bar to check if all info lining up to be sent is correct
Set the timer using the timer textbox
Teensy can be operates using battery or wire connection.
Uses Help button for extra information and Refresh to refresh.
Click on Preview display to check if the wave is what you want. If not

The general overview of the system starts with initialization and uploading code onto the Teensy through Arduino IDE. Then connects Teensy onto the software application by connecting Teensy to a computer. Open the software application and enter appropriate input parameters like duty cycle, frequency, and PWM sequence. Once done, check logging and info bar to make sure all data is correct. Next set the timer to desire time and finally check the preview window to see if the displayed pulse is correct. Finally, click start to generate the appropriate pulse for the laser. If it needs to change, just edit the input parameters and click send again. The Teensy will automatically take care of all pulse generating for the laser, and all the user has to do is change input on the software application. The rest of the flow chart should be straightforward.

2.2 User interface (Viet)

The Graphic User Interface is specially developed to control a Teensy 4.1-based module that generates parameterized square waves for neural stimulation purposes using pulsed lasers.

This section describes each feature of the GUI, which allows for precise control over the input parameters such as frequency, durations, and pulse width modulation sequence. These features are designed to ensure that users can easily program and monitor the laser for effective neural stimulation.



00:00:00.000

Frequency: None Hz   Duration: None%   COM Port: COM3   Baud Rate: 9600

Frequency Adjustment: 50 Hz

Duration Per Pulse: 0 %

N/A

Stopwatch: 00:00:58.841   14:44:45.702

```
24/04/19 14:43:46: 0ms: Initialization:
          Frequency=N/AHz, Duration=N/A%
24/04/19 14:43:46: 0ms: start:
          Frequency=NoneHz, Duration=None%
```

Figure 2.2.1 (Software application)

GUI Features:

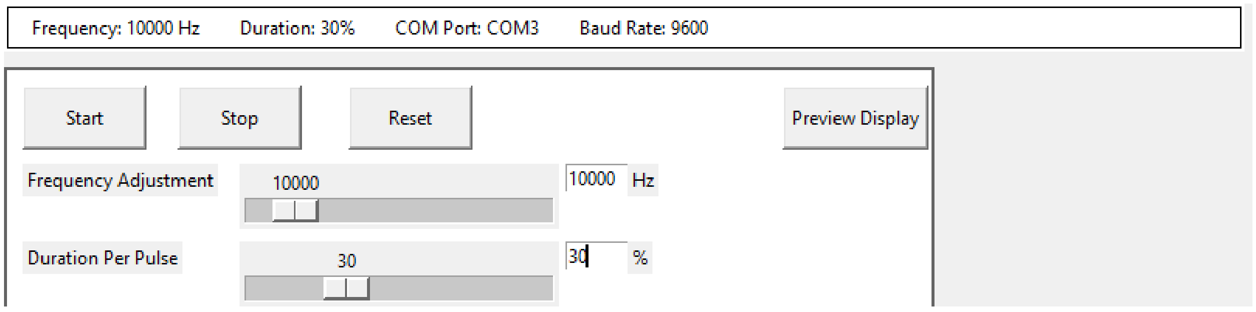
- **Control Panel (Start, Stop, Reset):** Basic operational controls for initiating, stopping, or resetting the current sequence.
- **Timer:** Sets a countdown for operations, aiding in precise control during experiments.
- **Frequency Adjustment:** Adjust frequency settings in Hertz to tailor the stimulation parameters.
- **Duration Adjustment:** Configure the duration of each pulse or sequence in milliseconds to match experimental requirements.

- **Pulse Chain Configuration:** Design and implement complex pulse chains using pulse width modulation.
- **Log Window:** Captures and displays a chronological log of operational data and changes for record-keeping and analysis.
- **Stopwatch:** Useful for tracking time precisely during experiments.
- **Real-time Clock:** Provides current time, useful for logging and coordinating experiment schedules.
- **Preview Display:** Visualizes the waveforms based on user inputs, allowing for adjustments before execution.
- **Info Bar:** Summarizes the current setup parameters, offering quick confirmation and review of settings.
- **Help Window:** Interactive guide and troubleshooting assistance to navigate the GUI effectively.
- **Refresh Button:** Manually refresh the GUI to ensure the displayed information is current.

Each feature is designed to support the user in efficiently setting up and conducting experiments, making the software interface an integral part of your research toolkit. The following sections will delve into the functionalities of each component, providing detailed instructions on how to utilize them to achieve optimal results.

2.2.1 Control Panel (Start, Stop, Reset)

The Control Panel is an essential feature of the GUI, consisting of three primary buttons: Start, Stop, and Reset. These buttons allow the user to directly interact with the function generator:


Frequency: 10000 Hz   Duration: 30%   COM Port: COM3   Baud Rate: 9600

Frequency Adjustment: 10000 — 10000 Hz

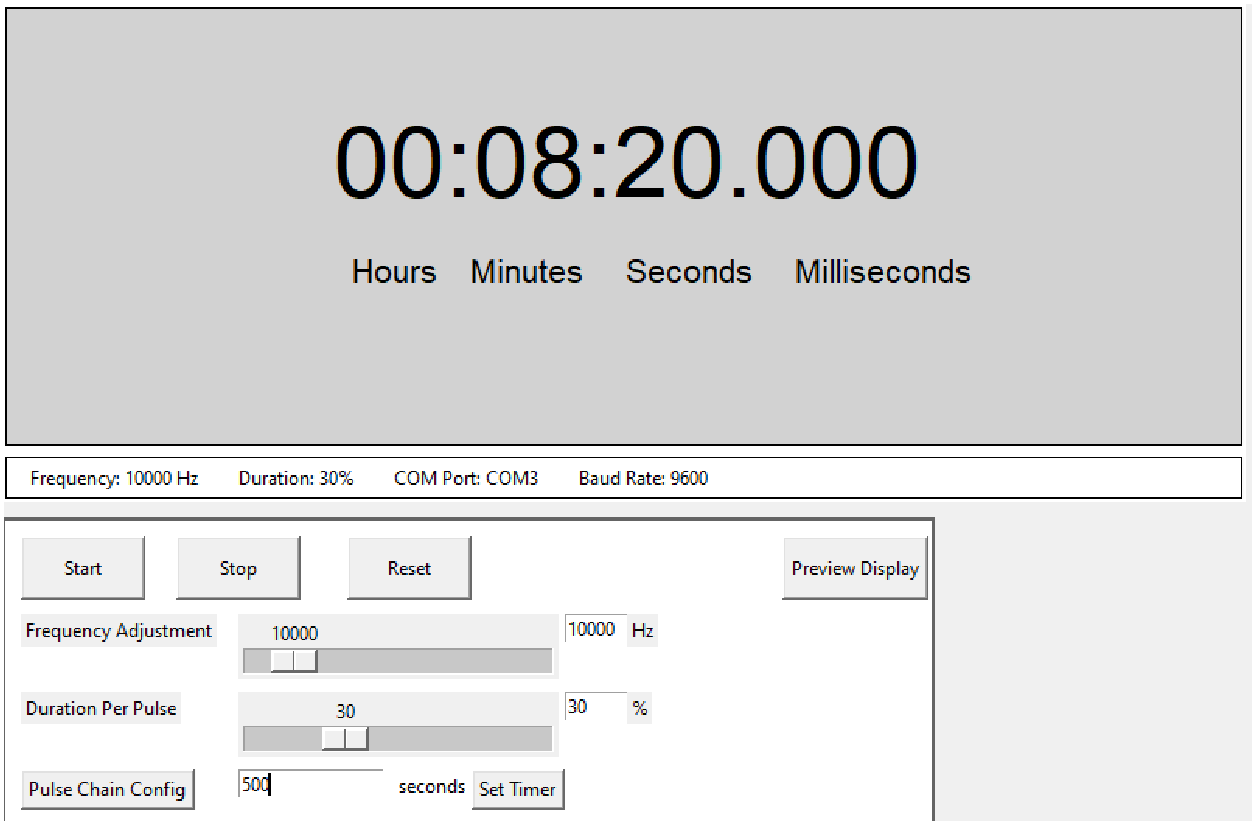
Duration Per Pulse: 30 — 30 %

Figure 2.2.1.1 (Control Panel)

- **Start:** Initiates the sequence of operations as per the configured settings, sending the command to the device to execute the user-defined input.
- **Stop:** Halts the device operation at any point, providing immediate termination of waveform generation.
- **Reset:** Reverts all settings to their default state, allowing the user to start fresh with new parameters.

2.2.2 Timer

The Timer feature enables users to input a specific time in seconds within an input box. Upon entry, this time is automatically converted and displayed in a structured format (hours, minutes, seconds) within the timer display.



Frequency: 10000 Hz Duration: 30% COM Port: COM3 Baud Rate: 9600

Frequency Adjustment 10000 10000 Hz

Duration Per Pulse 30 30 %

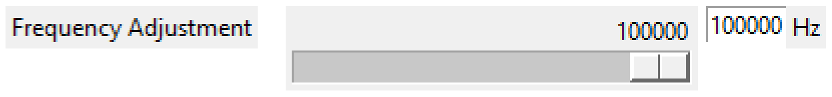
Pulse Chain Config 500 seconds

Figure 2.2.2.1 (Timer)

The timer starts when the Start button is pressed and counts down to zero. Upon reaching zero, it automatically stops the device from generating any further waves.

2.2.3 Frequency Adjustment

Frequency Adjustment allows users to set the desired frequency for waveform generation. The adjustable range is from 0 Hz to 100,000 Hz. Users can modify the frequency through



:

Figure 2.2.3.1 (Frequency Slider)

- **Input Box**: Users can enter a specific frequency value within the supported range.

- **Slider**: Allows for rapid adjustments, constrained within the range of 0 to 100,000 Hz.

Both controls are synchronized, ensuring that changes made in one are immediately reflected in the other.

2.2.4 Duration Adjustment

Duration per Pulse Adjustment lets users define the length of each pulse, with possible values ranging from 0 to 100 %. This feature provides both a slider and an input box:

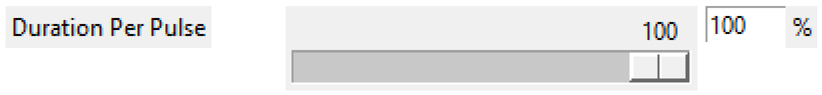
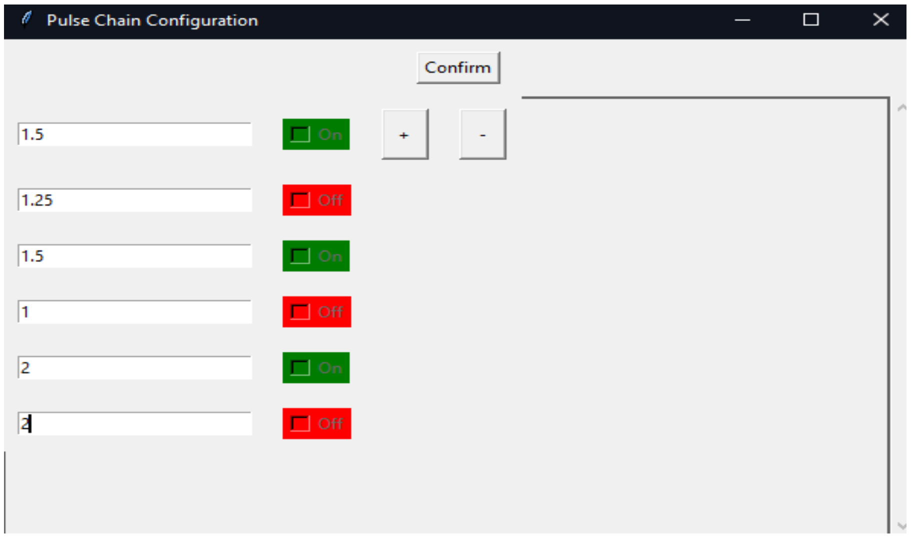


Figure 2.2.4.1 (Duration Slider)

- **Input Box**: Enables precise entry of the pulse duration within the allowed range from 0 to 100 %
- **Slider**: Offers a convenient way to adjust the duration quickly, also within the 0 to 100 % range.

2.2.5 Pulse Chain Configuration

Accessing the Pulse Chain Configuration is done by pressing the "Pulse Chain Config" button, which opens a dedicated window for setting pulse sequences. In this window, users can specify the duration in milliseconds for the "On" and "Off" states of the pulse. These states alternate, ensuring no consecutive "On" or "Off" states occur. Users can add or remove rows to adjust the sequence length:

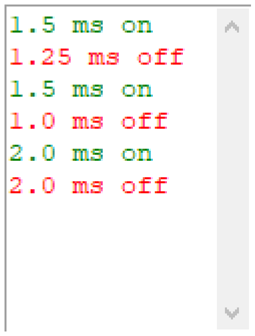


1.5	🟩 On
1.25	🟥 Off
1.5	🟩 On
1	🟥 Off
2	🟩 On
2	🟥 Off

Figure 2.2.5.1 (PWM sequence interface)

- **Add Row**: Provides additional input rows for extended configurations.
- **Remove Row**: Eliminates unwanted rows to streamline the configuration.

Upon confirming the settings, the current pulse chain configuration is displayed in a mini display below the configuration buttons. This display uses color coding—red for "Off" states and green for "On" states—for clear and quick recognition of the sequence



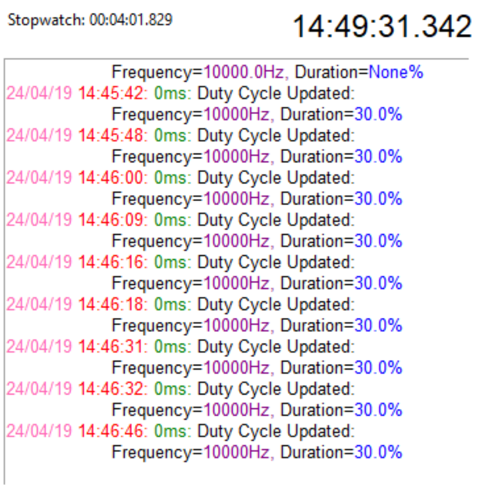
```
1.5 ms on
1.25 ms off
1.5 ms on
1.0 ms off
2.0 ms on
2.0 ms off
```

.

Figure 2.2.3.1 (Frequency Slider)

2.2.6 Log Window

The Log Window serves as an event tracker, reporting any changes made within the GUI. Each log entry includes:



```
Stopwatch: 00:04:01.829          14:49:31.342
                  Frequency=10000.0Hz, Duration=None%
24/04/19 14:45:42: 0ms: Duty Cycle Updated:
                  Frequency=10000Hz, Duration=30.0%
24/04/19 14:45:48: 0ms: Duty Cycle Updated:
                  Frequency=10000Hz, Duration=30.0%
24/04/19 14:46:00: 0ms: Duty Cycle Updated:
                  Frequency=10000Hz, Duration=30.0%
24/04/19 14:46:09: 0ms: Duty Cycle Updated:
                  Frequency=10000Hz, Duration=30.0%
24/04/19 14:46:16: 0ms: Duty Cycle Updated:
                  Frequency=10000Hz, Duration=30.0%
24/04/19 14:46:18: 0ms: Duty Cycle Updated:
                  Frequency=10000Hz, Duration=30.0%
24/04/19 14:46:31: 0ms: Duty Cycle Updated:
                  Frequency=10000Hz, Duration=30.0%
24/04/19 14:46:32: 0ms: Duty Cycle Updated:
                  Frequency=10000Hz, Duration=30.0%
24/04/19 14:46:46: 0ms: Duty Cycle Updated:
                  Frequency=10000Hz, Duration=30.0%
```

Figure 2.2.6.1 (Log Window)

- **Timestamp**: The exact time at which the modification was made.
- **Action**: Describes the nature of the change.
- **Current Setting**: Details the new setting after the change.

This feature is designed to enhance user awareness of the system's responsiveness and help prevent or identify operational errors by providing a clear, chronological overview of actions taken.

2.2.7 Stopwatch

The Stopwatch is integrated into the GUI and can be controlled using the Start, Stop, and Reset buttons. It measures time in milliseconds, allowing precise timing for activities that require exact duration measurement, such as experiments and tests.

2.2.8 Preview Display

The Preview Display enables users to visualize the waveform before activation, based on the entered duty cycle and frequency parameters. This visualization aids in ensuring that all settings are correct before commencing operations. Additional features of the Preview Display include:

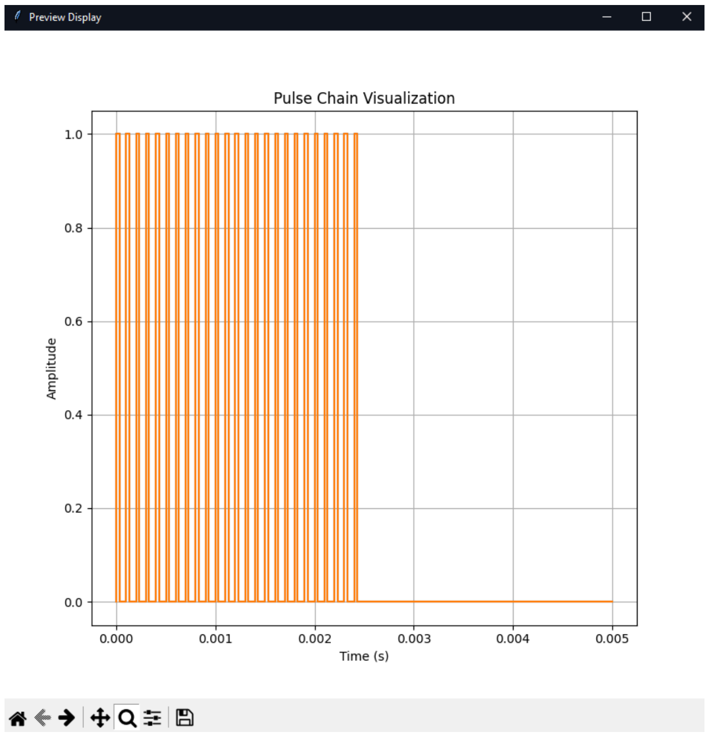


Figure 2.2.3.1 (An example Preview Display)

- **Save Feature**: Allows users to save the current waveform configuration.
- **Zoom In/Out**: Users can magnify or reduce the view for better visualization.
- **Pan**: Allows movement in all directions to examine different parts of the waveform.

2.2.9 Info Bar

The Info Bar displays vital information regarding the current operational settings, which include:

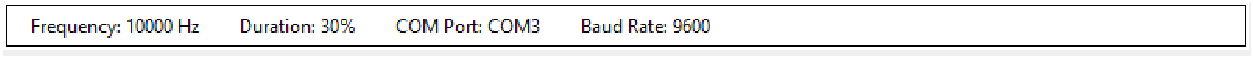

Frequency: 10000 Hz Duration: 30% COM Port: COM3 Baud Rate: 9600

Figure 2.2.9.1 (Info Bar)

- **Frequency and Duration per Pulse**: Displays the configured values.
- **COM Port and Baud Rate**: Indicates the connection status between the software and the hardware. If a connection fails, the display will read "Connection Fail." If the connection is successful, it will show the COM port in use.

This bar is crucial for ensuring that the device is properly configured and connected for operations, providing at-a-glance confirmation of system status.

2.3 **Physical description.** (Mateus Toppin)

The final device is about 2”x2”x1” in dimensions, containing a Teensy 4.1 mounted on a small breadboard. Pins 3 and GND of the Teensy have wires that are connected to alligator clips that extend out of the side port of the device. The user can connect the laser to the BNC connector of the alligator clip to utilize the device.

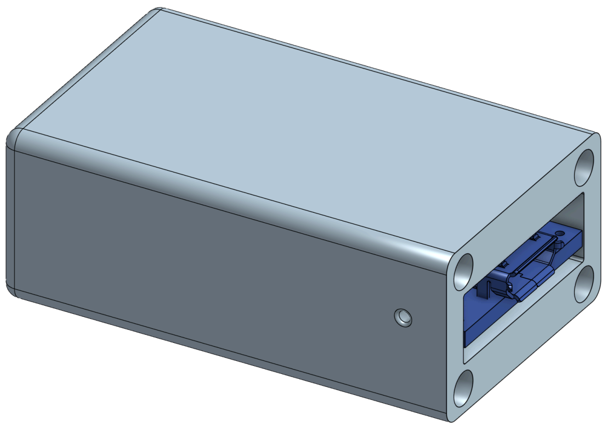


Figure 2.3.1 Visualization of Final product

2.4 **Installation, setup, and support** (Ian Lee)

Unpacking and Initial Assembly

1. Unboxing the Device:
    a. Carefully remove the device and all accompanying accessories from the packaging.

    b. Verify that all components listed on the packing slip are included.
2. System Requirements:
    a. Ensure your computer meets the minimum system requirements.
3. Initial Assembly:
    a. No assembly is required for the electronic model itself.
    b. Ensure that the device is placed on a stable, level surface away from moisture.

Installation of the Electronic Model

1. Connecting to a Computer:
    a. Use the provided USB cable to connect the electronic model to your computer.
    b. Ensure that the connection is secure and the computer recognizes the device
2. Software Installation:
    a. Download the required Python script from our GitHub repository.
    b. Navigate to the downloaded script location on your computer.
    c. Run the script by opening a command prompt or terminal and typing python **Main_finalized**.py.
    d. Follow any on-screen prompts to complete the setup. The default configuration is recommended for most users.
    e. No need to restart your computer after installation unless specifically prompted by the script.

Device Configuration

1. Connecting to the Laser:
    a. Connect the output port of the electronic model to the laser unit using the appropriate cable. Ensure that all connections are secure and correct to avoid any operational hazards.
2. Setting Preferences:
    a. Launch the python script.
    b. Navigate to the 'Settings' or 'Preferences' menu to configure the device settings.
    c. Set the desired Pulse Width Modulation (PWM) parameters according to your specific requirements. Default settings are configured for general use, but can be adjusted as needed.

Power-Up and Initial Test

1. Powering the Device:
    a. Ensure that the electronic model and the connected laser are both powered on. Follow the specific instructions for each unit regarding safe power-up procedures.
2. Conducting a Test Run:
    a. Use the software interface to initiate a test run.
    b. Verify that the laser operates according to the set PWM parameters.

    c. Adjust the settings if necessary and retest until the desired operation is achieved.

Troubleshooting

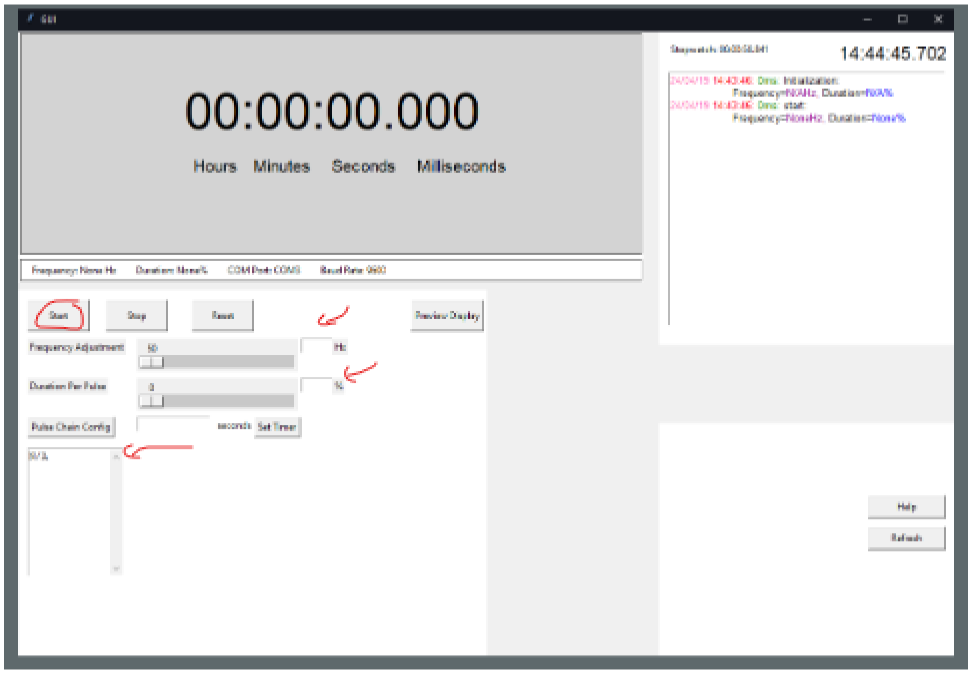
1. If the Teensy board is not functioning as expected, consult our GitHub repository for guidance.
2. Re-upload the TeensyCodeV2.ino file using the Arduino IDE to the Teensy board, ensuring to select the correct ports during the upload process.

3 Operation of the Project

3.1 Operating Mode 1: Normal Operation (Tian)

The normal operation for the device is very simple and only involves user interacting with the software side to generate pulse:

1. User open the software applications
2. User enters input on the following input box: frequency and duty cycle and enters PWM sequence in the PWM window.
3. Users can optimally set the timer. If not, click the preview display to check if the preview wave is what the user wants. If yes, hit start to send the inputs. If not, go back to step 2.
4. After data is sent, the operation is done since the rest is taken care of automatically in the backend. Users can stop the send manually by hitting stop.



```
Frequency: None Hz    Duration: None%    COM Port: COM3    Baud Rate: 9600
```

```
24/04/19 14:43:46 Dms: Initialization
          Frequency=N/AHz, Duration=N/A%
24/04/19 14:43:46 Dms: start
          Frequency=NoneHz, Duration=None%
```

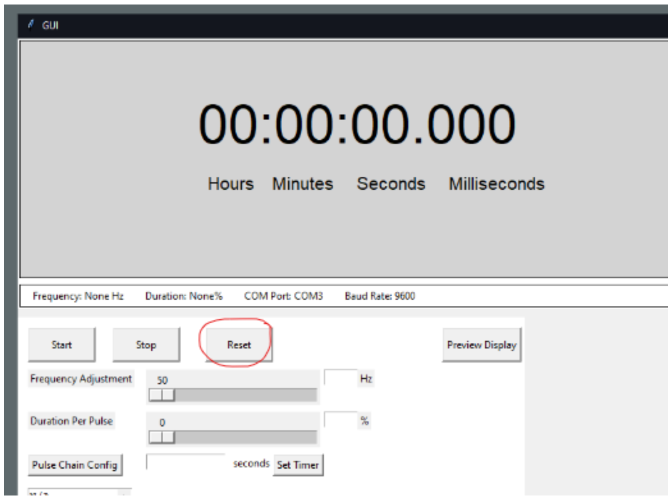
Figure 3.1.1 (All user have to do is enter the input indicated by the red arrows then hit start)

Overall, the normal operation is straightforward and easy since most of the operation is done automatically in the backend which users do not have to worry about. This is purposely done since our client emphasizes the concept of user-friendliness. In short, normal operation just involves the user to input three parameters and hit start. After start, the desired wave will be automatically outputted if all is setup correctly as discussed in section 2.

3.2 **Operating Mode 2: Abnormal Operations** (Mateus Toppin and Raphael Mok)

Software Errors

1. If the set frequency and duty cycle from the GUI does not transfer to Teensy, simply restart the GUI application.
2. If the Preview Display appears blank after a successful test, close the preview display and start the test again. If it still does not appear and the reset button does not work. Re-Launch the GUI application.



00:00:00.000

Frequency: None Hz   Duration: None%   COM Port: COM3   Baud Rate: 9600

Frequency Adjustment 50

Duration Per Pulse 0

Figure 3.2.1 (Click Reset to restart by relaunching the application)

3. If the device does not respond to the external trigger, ensure the voltage is sufficient enough to be detected. Users should not run a current of over 3.3 volts into the external trigger.

3.3 **Safety Issues** (Mateus Toppin and Raphael Mok)

1. The Teensy device does not have the capability to receive an input voltage greater than 3.3V. Users are advised not to overpower the device to ensure efficiency.
2. The interior of the device is susceptible to water damage. Users are advised to keep liquids away from the device, to ensure that liquid does not spill into the ports.
3. The Teensy board is prone to heating when extreme frequencies are run for long periods of time. Users are advised to not run the device for longer than 15 minutes at frequencies greater than 50kHz as to not overheat the Teensy or melt the plastic.
4. Low voltage can be present along the BNC cables and clips protruding from the device. Users should not directly touch any wires if they become exposed or the circuitry in the Teensy unless the device is disconnected from a power source.

4 **Technical Background** (Viet Nguyen and Ian Lee)

The Yang Lab currently faces significant limitations with their existing function generator devices, which are too large and lack a software interface for efficient operation. In response to these challenges, a new development has been proposed: a compact, user-friendly, and cost-effective electronic model designed to integrate seamlessly with the lab's laser systems. This innovative solution aims to replace the cumbersome older models with a smaller, software-driven device that enhances functionality while meeting the lab's space and budget constraints. The new design focuses on ease of use through a software application, enabling precise control over the generation of parameterized pulses and improving overall experimental workflow.

The user interface (UI) of the integrated laser and electronic module is crucial for efficiently managing the function generator in Yang Lab's setup. Developed using the Python tkinter library, the graphical user interface (GUI) serves as the main interaction point between users and the Teensy 4.1 hardware that controls the function generator. Upon launching the application, the GUI's main window, designed for intuitiveness and accessibility, allows users to adjust critical parameters like frequency, duty cycle, and pulse width modulation sequence via sliders, input boxes, and buttons. These inputs are compiled into a command string that the Teensy device executes. The GUI's layout includes a control panel with essential functions such as Start, Stop, and Reset, which manage the pulse sequence operations. Automated features in the software detect the connected COM port and set the baud rate, simplifying setup and enhancing usability. Additionally, a timer functionality counts down from a user-set time, issuing a stop command upon expiration, crucial for precise experimental timing.

Data logging in the GUI is thorough, offering real-time feedback and historical records vital for monitoring system performance and identifying issues. Every action within the GUI is time stamped and logged, detailing the performed action and adjusted settings, all visible within a specific window. Additionally, the update_data_log() function tracks every significant event, like frequency updates or start/stop actions, with timestamps, making this feature crucial for troubleshooting and reviewing session histories. Visualization tools in the GUI provide immediate feedback on pulse chain configurations; adjustments are graphically displayed in the 'Pulse Chain Visualization,' showing the pulse on/off states over time. This not only ensures the correct setup before execution but also enhances understanding of how changes affect the pulse output. The GUI also includes a help section, accessible via the open_help_window(), offering guidance on application use and troubleshooting, serving as an essential self-help resource.

The GUI features an auto-refresh mechanism that automatically updates all interface components to reflect the latest changes and status, eliminating the need for manual user intervention. This is especially useful in settings where parameters are frequently adjusted, requiring prompt feedback. For more intricate setups, like configuring a sequence of pulses with various durations and states, the GUI offers a dynamic interface where users can add or remove configuration rows as needed. Each row represents a segment of the pulse sequence, allowing users to specify durations and states (on/off). This adaptability is crucial for customizing the function generator's output to meet specific experimental requirements.

The GUI of the integrated laser and electronic module utilizes the tkinter library for its graphical interface, matplotlib for visualization, and the serial library for hardware communication. These tools are selected to create a robust, interactive, and user-friendly interface. Tkinter, Python's standard GUI toolkit, is employed to develop a responsive and straightforward interface featuring widgets like buttons, sliders, and text inputs that are crucial for configuring the function generator's parameters. These widgets are organized into frames (tk.Frame), which are sections of the GUI designated for specific tasks such as configuration controls, visualization, and logging. This structured layout facilitates intuitive navigation across different functionalities. Additionally, event-driven programming manages event handling in the GUI, where user interactions like button clicks and slider adjustments activate specific functions.

Visualization is a critical component, particularly for confirming pulse chain configurations before execution. The matplotlib library is integrated using FigureCanvasTkAgg from matplotlib.backends.backend_tkagg, allowing matplotlib figures to be embedded into Tkinter applications. In the function update_plot(), the GUI plots the current waveform using ax.plot(), where ax is a matplotlib subplot. This visualization updates in real-time as users adjust settings like frequency and duty cycle, providing immediate visual feedback.

The serial library is fundamental for the GUI to communicate with the Teensy hardware. It facilitates sending and receiving data over serial ports, crucial for a hardware-integrated project like this. The function auto_connect() scans available COM ports and automatically establishes a connection with the Teensy device, eliminating the need for manual configuration by the user. Upon pressing the Start button, the start_command() function compiles all user-defined settings into a formatted string and sends it to Teensy via ser.write(), instructing the device on how to generate the desired pulse sequence.

The GUI provides advanced configuration options such as setting the timer, pulse chain configuration, and updating the frequency and duty cycle. The timer_countdown() function, implemented in a separate thread, ensures that the GUI remains responsive, automatically sending a stop command when the timer expires, crucial for experiments requiring precise timing. Users can dynamically add or remove pulse chain configurations in the open_pulse_chain_config() function, allowing extensive customization of pulse sequences.

The Teensy 4.1 microcontroller serves as the central hardware controller in the Yang Lab setup, orchestrating the generation of pulse sequences crucial for experimental applications. It utilizes embedded C++ code to interact directly with the hardware components, providing a fine-grained control mechanism for the laser's pulse modulation. This control is achieved through a combination of direct register manipulation and utilization of Teensy's built-in PWM capabilities, facilitated by comprehensive use of the TimerOne library.

Upon receiving instructions from the GUI via serial communication, Teensy's firmware initiates the parsing and execution of these commands. The code structure employs a buffering system to capture incoming serial data, using start (<) and end (>) markers to frame command strings. This is critical for synchronizing the communication between the GUI and the microcontroller, ensuring data integrity by preventing the processing of partial commands. The recvWithStartEndMarkers() function is designed to manage this communication efficiently. It maintains a reception state, storing incoming characters until the end marker is detected, at which point it flags the complete data string for processing.

The parseData() function is where the bulk of command interpretation occurs. It tokenizes the received string using commas as delimiters to extract and convert the sequence size, frequency, and duty cycle into usable values. These values are then used to configure the PWM settings of the Teensy. The frequency and duty cycle directly influence the PWM configuration through the analogWriteFrequency() and analogWrite() functions, which adjust the timer's compare registers to match the desired output characteristics. This setup allows for dynamic adjustments to the pulse characteristics, such as changing the frequency or modifying the duty cycle in real-time based on experimental requirements.

Handling the pulse sequence involves iterating through an array of durations that dictate the timing of on and off states. The main loop checks each element of this array: if the index is even, it represents an active pulse and the PWM output is set to the computed duty cycle; if odd, the output is disabled, creating a pause or off state. Each duration is handled using a combination of delay() and delayMicroseconds(), providing the necessary precision for transitions between active and inactive states.

Moreover, Teensy's code includes feedback mechanisms via the Serial.print() function, which sends acknowledgments back to the GUI. This two-way communication protocol ensures that the GUI can monitor Teensy's status and adjust the configurations dynamically based on ongoing experimental observations or requirements.

This collaboration begins with the GUI, which allows users to intuitively manipulate and oversee critical parameters like frequency, duty cycle, and pulse width modulation sequence. The GUI's thoughtful architecture, enhanced by real-time visual feedback and robust data logging, ensures that users are not only operators but informed participants in the experimental process. Each interaction within the GUI translates into precise commands, which are then seamlessly relayed to the Teensy microcontroller through serial communication, showcasing an exceptional alignment of user inputs with machine operations.

The microcontroller's ability to parse and execute received command strings effectively ensures that the intended pulse sequences are generated accurately, thereby aligning with the user's expectations and experimental requirements. This responsive interaction is further augmented by feedback mechanisms that inform the GUI of the system's status, allowing for a dynamic, real-time dialogue between the user and the hardware.

The integration of automated detection and connection features in the GUI minimizes setup complexity, enhancing the user experience by focusing on ease of use and reliability. The auto-refresh mechanisms and the dynamic addition or removal of pulse sequence configurations allow users to adapt the system on the fly, which is critical in a fast-paced research environment. This adaptability ensures that the system not only meets the current needs of Yang Lab's experiments but is also poised to accommodate future demands.

5 **Relevant Engineering Standards** (Raphael & Tian)

This design follows IEEE 1073-1996 Standard for Common Framework of Location Services for Healthcare. This device does not save or transmit any personal patient data used in any healthcare context. The final device only takes three inputs, frequency, pulse width modulation, and duty cycle as inputted by the user. Then the input will drive the laser. Thus, no personal patient data is used.

This design follows the US Occupational Safety and Health Administration's 1910.303 standards for low voltage devices. It does not contain any specific components that would be considered hazardous if run at the voltage level recommended for this device and does not contain any exposed wiring or potentially hazardous components at this voltage level.

This design follows the ISO/IEC 15288:2015 standard for systems and software engineering processes and lifecycle stages, specifically those related to the testing of software design. Our device has gone through a series of considerable testing and has been run through a number of failure cases as well as been tested with any relevant external instruments to ensure no software bugs or issues occur,

As for software, IEEE 1012: Standard for System and Software Verification and Validation. This standard provides requirements for verification and validation processes to ensure that software meets its specified requirements and user needs throughout the development life cycle. Throughout the development of the software process, the team frequently met with the client to discuss the function needed for the software application and what needs to be implemented. These meetings are also a way to show the client the most up to date software and any feedback was given. Thus, fulfilling the practice of robust software development.

Finally, agile standards are also followed. While not formalized into a single standard, methodologies like Scrum and Kanban have guidelines and practices that are widely adopted in agile software development environments. The Agile Manifesto itself outlines principles intended to guide teams on how to develop more adaptable and responsive software. This emphases flexibility, continuous improvement, customer involvement, and a high responsiveness to change over rigid adherence to predefined processes.

6 Cost Breakdown (Tian Huang)

Project Costs for Production of Beta Version				
Item	**Quantity**	**Description**	**Unit Cost**	**Extended Cost**
1	1	Teensy 4.1	$33.08	$33.08
2	3	TI SN74AC541	$0.81	$2.43
3	3	FT260Q Buffer Chip	$1.99	$5.97
4	1	3D Printing Filament	$19.99	$19.99
5	1	SanDisk 128GB Memory Card	$20.99	$20.99
6	1	SparkFun Bi-directional level shifter	$3.50	3.50
7	1	Breadboard	$8.99	$8.99
8	2	M/M Jumper Wire	$2.10	$4.20
9	1	SparkFun Solderable Breadboard	$5.50	$5.50
10	1	USB-A to USB-C Adapter	$4.50	$4.50
11	4	Brass Thread Insert	$2.75	$11.00
			Total Cost	$120.15

Most of the items above are self-explanatory. The core hardware components are the Teensy 4.1 board. The TI and the buffer chip above are used to build power amplifier circuits. The memory card and 3D printing filament are used to print out the enclosure. Level shifter is used to increase voltage to 5 V if needed. Both breadboards are used to build the power amplifier circuits. USB-A to C is used to connect the Teensy board to the computer. Brass Thread is used for enclosure.

7. **Appendices** (Tian Huang)

7.1 **Appendix A - Specifications** (Tian Huang)

Team 19

Requirements	**Values, range, tolerance, units**
Case Dimensions Must be less than (6"x6"x6")	Current: 2"x2"x1"
Output Voltage Level (Teensy)	High 3.3 V & low 0 V
Output Current Level (Teensy)	75 mA
Shipping Requirement	The interior of the enclosure is designed with compactness for shipping purposes
Cost of the final product Must be as cheap as possible	About $100
Software Application	● The application is able to upload correct parameters: frequency, duty cycle, and PWM sequence ● The preview screen is able to display the accurately generated waves before actually generating ● The logging window displays the history of all inputs sent ● The timer function will continuously generate the entered pulse as the timer is on. But, terminate the generating process as it ends. ● Auto-detection of port and Baud rate
Board Requirement	The Teensy 4.1 is able to perform the pulse generating task to perfection with its Cortex M7 chip running at 600 MHz clock rate. The generated pulse is consistent, accurate, and with no random fuzziness.
Interface	The interface between the software application and hardware is connected through a USB port. The data uploaded from software should be read instantly by Teensy.

7.2 Appendix B – Team Information (Tian Huang)

Team 19, also known as the integrated electronic laser model team, consists of Tian Huang, Ian Lee, Raphael Mok, Viet Quoc Nguyen, and Mateus Toppin. The team consists of two computer engineers who are Tian Huang and Viet Quoc Nguyen. The team also consists of three electrical engineers Raphael Mok, Ian Lee, and Mateus Toppin. This somewhat even separation leads two sub teams for this project, the software team made up of computer engineers and the hardware team made up of electrical engineers. The reason the team chose to take on this challenge is because the team truly believes in the technology Yang Lab is working on in dealing with the neurological disorders such as anxiety, depression, Parkison, etc. Each member believes such issues are a serious matter and hope to contribute by signing up for this project. In addition, the proposal of an integrated electronic model with a laser posed very interesting engineering problems that involve many disciplines in hardware and software. The team will use their combined knowledge and skills to help design the most efficient square pulse generating device.

Tian Huang is a graduating Computer Engineer who has a passion for different technological subjects including machine learning, developing applications, circuitries, embedded programming, and robotics. Basically, a combination of hardware and software. He also spent the majority of his free time reading and learning about history and philosophy. Tian hopes to use his skills as an engineer to solve many technical problems and hopes to have a positive impact on society.

Ian Lee is a graduating Electrical Engineer who has a strong passion for machine learning, programming with python, and many other subjects with applications to the medical field. Ian hopes to use his skill as an electrical engineer to help develop something impactful and meaningful. Starting next fall, Ian will be attending University of Toronto for his Master’s degree.

Raphael Mok is a graduate of Electrical Engineering who has a strong interest in the field of electrical and computer engineering. Raphael has many interests in circuitry, embedded system, and signal processing. Raphael will continue to pursue his master at Boston University starting next fall.

Mateus Toppin is a graduate of Electrical Engineering who has a passion for programming, building circuits and semiconductor devices, and digital signal processing. He has utilized Github for many tasks and is skilled at embedding systems, signals, and systems. Mateus hopes to continue to pursue his passion in engineering.

Viet Quoc Nguyen is a graduating Computer Engineer with a deep passion for robotics, embedded systems, and programming. During his studies, he has dedicated himself to designing innovative solutions that enhance the functionality of robotic systems and embedded technologies. Viet will continue his passion as a Computer Engineer at FPT USA Corporation. There, he plans to hone his skills in developing engineering solutions and programming, before returning to Vietnam to launch his own startup. Intent on making a significant impact in his home country, Viet is poised to contribute meaningfully to the advancement of technology in Vietnam.