ECED 4901: Senior Year Project II

Dalhousie University, Department of Electrical & Computer Engineering &

The Quinn Laboratory, Department of Physiology & Biophysics

FINAL REPORT:
LED Control System for Fluorescence-Based Measurements of the Heart's Electrical Activity

Written by Group #9:

Matthew Watson (B00894843)

Devon Harvey (B00890298)

Ha Vu (B00896065)

Prepared for:

Dr. Jose Gonzalez-Cueto & Zachary Long

Group Member Signatures:

Matthew Watson	Signature: _____	Date : 4/12/2025
Ha Vu	Signature : _____	Date : 4/12/2025
Devon Harvey	Signature: _____	Date : 4/12/2025

Supervisor Signatures:

Zachary Long	Signature : _____	Date : 4/12/2025
Dr. Jose Gonzalez-Cueto	Signature : _____	Date : 4/12/2025

4 December 2025

Table of Contents

List of Figures

List of Tables

List of Acronyms

- **LED** – Light Emitting Diode
- **GUI** – Graphical User Interface
- **TTL** – Transistor-Transistor Logic
- **PWM** – Pulse Width Modulation
- **MOSFET** – Metal-Oxide-Semiconductor Field-Effect Transistor
- **BJT** – Bipolar Junction Transistor
- **ISR** – Interrupt Service Routine

Abstract

This report contains an overview of the development of an LED Control System for fluorescence-based measurements of cardiac electrical activity, completed by Project Group #9. The team has successfully developed a system that is precise, repeatable and user-configurable for optical stimulation in cardiac tissue experiments by coordinating LED illumination with an external imaging system. A desktop graphical user interface (GUI) was developed that allows users to configure experiment parameters, consisting of the systems operating mode, flash frequency, pulse duration, LED illumination pattern and LED brightness. These settings are transmitted to a microcontroller in a structured configuration packet over a serial port from the user's desktop. In the final design, brightness control is achieved through a configurable pulse-width modulation (PWM) and an adjustable LED pulse on-duty cycle, which enables the control of LED brightness intensity without modifying hardware. Oscilloscope testing to ensure the system produces stable pulse trains was completed, where timing expectations were confirmed across relevant system frequency ranges. The team opted to build the physical circuit on a perforated circuit board after initially prototyping on a simple breadboard. Soldering the circuit components onto the perforated board ensures firm, reliable connections that remain secure during testing and usage. This eliminates the risk of components coming loose as well. This report outlines the group objectives, project plan, system architecture, a critical analysis of the hardware and software design decisions, verification procedures and limitations, and concludes with recommendations for future enhancements.

Note:

Appendix A of this report indicates which parts of this report were written by which members of the group, separated by section.

1. Introduction

Fluorescence-based imaging has become an important tool in studying cardiac electrophysiology, as it provides important mechanistic insights into how cardiac arrhythmias occur through the excitation and contraction of cardiac cells (Herron et al., 2012). Experiments using such imaging techniques rely on the precisely timed illumination of fluorescent dyes using high intensity LEDs that are synchronized with a high-speed camera (Lee et al., 2011). To obtain consistent and reproducible dye excitation measurements, the timing, duration and intensity of the illumination of the LEDs used to excite these dyes must be controlled (Baines et al., 2024). This project focuses on the design and implementation of a user-configurable LED control system that allows users to define the LED illumination parameters for cardiac fluorescence-based imaging experiments for researchers at the Quinn Laboratory at Dalhousie University. The system combines a desktop graphical user interface (GUI) with a microcontroller-based LED driver and a custom LED power circuit, which enables researchers to configure flashing frequency, pulse duration, illumination pattern and brightness to meet experiment requirements. By centralizing these controls in a single integrated platform, the LED control system simplifies experimental setup and improves repeatability between experiment trials.

1.1. Project Background

The Quinn Laboratory at Dalhousie University conducts fluorescence-based cardiac imaging experiments to study how arrhythmias and cardiac diseases develop in cardiac tissue. In these experiments, cardiac tissue samples are stained with fluorescent dyes then illuminated by high-intensity LEDs, where a high-speed camera records the resulting fluorescence signals emitted from the tissue (Lee et al., 2011). Prior to this project, the laboratory relied on an LED control system that was limited in configuring the LED illumination duration and brightness levels, where adjusting the pattern required manual changes to hardware and lab equipment. This made reproducing experimental conditions more difficult and resulted in a lack of standardization amongst users.

1.2. Project Rationale

The LED Control System is designed to enable precise, synchronized illumination of fluorescent dyes and proteins used in cardiac electrical activity experiments. Current commercial systems lack flexibility in brightness control, pulse synchronization, and cost-effectiveness (Lee et al., 2011). In general, these existing systems are often very complex, expensive and do not allow researchers to configure system parameters rapidly, leading to potential problems when researchers are required to adjust illumination protocols or want to replicate previous experimental conditions. The proposed system aims to address these shortcomings by providing a customizable and user-friendly interface for researchers to control LED switching patterns and brightness dynamically.

1.3. Investigation of Literature

In order to grasp the application of which the control system was to be used in; the team consulted the sponsor as well as existing literature regarding fluorescence spectroscopy. It was found that fluorescence spectroscopy is a technique that identifies, and measures molecules based on the light they emit after being excited by a specific wavelength (Romani et al., 2010). When a molecule absorbs light, its electrons move to a higher energy state and then release light of a longer wavelength as they return to a lower state (Romani et al., 2010). The technique measures either an emission spectrum such as fixed excitation, scanning emitted wavelengths or an excitation spectrum such as fixed emission, scanning excitation (Romani et al., 2010). These spectra are highly sensitive and offer lower detection limits than absorbance-based methods (Romani et al., 2010).

2. Objectives and Deliverables

This section of the report details the objectives and deliverables for the project.

2.1. Objectives

The overall objective of this capstone project is to design, build and deploy a user-configurable LED control system for precision optical experiments conducted in the Quinn Laboratory. This system is an enhancement of the current LED control system used at the Quinn Laboratory for fluorescence-based cardiac imaging. The initial objectives of the project are listed below, from the perspective of a researcher at the lab:

- Enable both externally initiated (by the camera) and user-initiated (by the control system) triggering capabilities to synchronize LED pulses with camera exposure/frames.
- Develop a user-friendly graphical user interface (GUI) which allows researchers to configure system parameters such as the control system's triggering mode, LED frequency, experiment duration, LED flashing pattern and LED brightness.
- Ensure that the system is safe, robust and easily operable in a laboratory environment where researchers have minimal training on the control system itself.
- Provide researchers at the lab with a support manual to allow for future system alteration.

These objectives were initially defined in the project progress report and guided project development during the first semester of development. At the end of first the term, the group met all these core objectives by producing a functional prototype, which consisted of a basic hardware prototype, a basic GUI and basic microcontroller firmware that was able to generate user-defined LED pulse patterns. At the

beginning of the second term of project development, the objectives of the project were refined to focus on optimizing system performance and usability, where these objectives are listed below:

- Implement MOSFET's to improve LED switching efficiency and reliability.
- Enhance the GUI and microcontroller software to allow users to define LED brightness and have more customizable LED switching pattern settings.
- Maintain precise synchronization between LED pulses and the camera system.
- Design and develop the circuit on a perforated circuit board.
- Ensure regulatory compliance and usability in experimental settings.

The objectives above were successfully met during the second term of the project's development.

2.2. Deliverables

Throughout the two-term duration of this project, the following major deliverables were produced and delivered to the Quinn Laboratory:

- Functional Hardware Driver Circuit (Completed):
    - A fully functional LED driver circuit was designed and built by the team, where a microcontroller was used to control the LED system. The final hardware using MOSFETs for high-power LED switching, which improves efficiency and reliability compared to the initial BJT-based prototype. The circuit supports high-frequency LED switching, which is suitable for the requirements of imaging experiments, and interfaces with the custom GUI over serial.
    - The circuit was soldered and assembled on a perforated board, which is to be housed within the chassis and equipped with banana plug connections for LEDs, power supplies, and camera inputs.
- Custom GUI Application (Completed):
    - A GUI written using the PyQt6 framework in Python was completed, which allows users to select a COM/ USB port, define a flash rate and experiment duration, choose between LED switching patterns and switch between manual and trigger mode settings. In the final product, the GUI was extended to include PWM-based brightness control and on-time duty cycle control through configurable parameters. The GUI displays a terminal output of all serial communication between the user's computer and the microcontroller, which provides a live view of the systems behaviour.
- Microcontroller Software Module (Completed):

    - A firmware module written in Arduino IDE was developed for the circuit. This software reads and interprets ‘configuration packets’ sent from the GUI over a serial port, allowing the microcontroller to control the overall LED system according to the system configuration parameters defined by the user on the GUI. The final firmware supports all operating modes (external and internal triggering of the lab camera), custom illumination patterns and PWM/ duty-cycle control for LED brightness control. This code was developed in a way to be maintainable and easily modifiable by future students or lab users.
- LED/ Camera Timing Synchronization Report (In Progress):
    - A dedicated report is being prepared, which summarizes the measurements of the LED relative to control and camera trigger signals. These tests verified that the system is able to synchronise LED illumination with camera exposure across the intended operating frequency range. This report details oscilloscope measurements on the team's prototype LED control system and the lab’s camera, ensuring that LED-camera synchronization requirements are met under high-frequency operation.
- Final Documentation (In Progress):
    - The completed system was delivered to the lab, where a detailed documentation package containing every aspect of the system is being written. This document contains all circuit schematics and summaries of code, a user manual and notes/ instructions to allow future system changes (if required). This document also contains a maintenance guideline and troubleshooting report, which details possible fixes to common errors.

3. Technical Requirements

Based on the project objectives that were previously outlined in section 2 of this report (above), a list of measurable technical requirements was defined to guide the design and verification of the LED control system. The requirements are written so they can be verified through oscilloscope measurements, inspection of the GUI or user testing. The intention of these requirements is to ensure that the system meets the needs of the researchers using the control system to conduct the experiments in the Quinn Laboratory, where the need for synchronized and repeatable LED illumination during cardiac fluorescence imaging experiments is required. Appendix B of this report contains a table that summarizes the main technical requirements, along with the intended verification method and a brief reasoning of the requirement. The tests described in Appendix B were completed, and the results are outlined in section 8.2 of this report.

4. Methods

This section outlines the different design approaches taken for the LED control system and summarizes the technical background of each approach to give context on the chosen final solution. The goal was to select an approach that satisfies the requirements for accurate system timing, variable brightness control and the ease of use in a laboratory environment, while keeping the system maintainable, robust and low-cost.

4.1. Possible Approaches

The different general approaches taken to complete this project are separated into two different general categories; hardware and software (firmware). These were the two general areas of the project that required unique design approaches where multiple alternative approaches were evaluated before selecting the final system architecture.

4.1.1. Hardware Approaches

This section identifies the different hardware approaches taken and details how the final hardware components were determined. In developing the LED control system, several hardware configurations were considered before finalizing the current design. The selection process emphasized durability, reliability, and performance under repeated and prolonged use. The hardware design evolved from an initial breadboard prototype to a soldered assembly on a perforated circuit board. Although a printed circuit board was the original request of the team's sponsor, it was later determined that a printed circuit board would require significant lead time to design, manufacture, and ship to the destination. Additionally, printed circuit boards are more expensive than perforated boards, and should there be an error with the printed board it would be much harder to swap out components and test. The breadboard offered convenience for early testing and rapid reconfiguration, but its inherent limitations such as loose connections, higher contact resistance, and mechanical instability made it unsuitable for long-term use. To ensure durability and consistent electrical performance, the final design was soldered onto a perforated board, providing secure connections and improved mechanical robustness.

Table 1 below details the different criterion taken into consideration when determining the final hardware circuit.

Criterion	Weight	Breadboard (1-5)	Perforated Board (1-5)	Printed Circuit Board (1-5)
Cost	0.10	5	5	2
Lead Time	0.10	5	5	1
Ease of Component Rework	0.10	5	4	1

Electrical Performance	0.10	1	4	5
Assembly Time	0.20	5	4	1
Professional Finish	0.20	1	4	5
Durability	0.20	1	4	5

Table 1: Hardware Approach Criterion

- Breadboard: **5**
- Perforated Board: **4.6**
- Printed Circuit Board: **3.1**

In parallel, the choice of switching devices was carefully evaluated. While bipolar junction transistors (BJTs) were considered, their slower switching speeds and higher power dissipation posed constraints for LED control. Metal–Oxide–Semiconductor Field-Effect Transistors (MOSFETs) were ultimately selected due to their superior durability, high current-handling capacity, and fast switching capabilities. These characteristics not only enhance efficiency but also extend system longevity, making MOSFETs the more reliable solution for driving LEDs in demanding applications. Together, these hardware decisions reflect a deliberate balance between prototyping flexibility, operational stability, and performance scalability.

4.1.2. Software & Firmware Approaches

This section contains the different approaches considered when implementing the firmware & software used to control the logic of the LED control system. When approaching the control logic, several methods were evaluated for implementing the requirements of the firmware; where the GUI software was required to seamlessly interact and give commands to the systems firmware used to drive the LEDs. These methods were divided into different core requirements; brightness control, triggering and synchronization and user interaction/ configuration. These core requirements are discussed in sections 4.1.2.1. to 4.1.2.3. below, respectively:

4.1.2.1. Brightness Control Approaches

Two main approaches were considered for LED brightness control. The trade-offs and considerations are described in detail below:

1. *Analog Current Control (Omitted Approach):*

The first method considered was an analog current control, where the average LED current was to be varied using a digital potentiometer within the LED driver circuit (Texas Instruments, 2016). This approach allows the firmware to send a numerical value/ signal that sets the LED current through an

analog interface, where the user is able to control this value. This method can provide smooth brightness control (which is ideal), but this method also requires additional analog circuitry and a very careful and complex design approach to ensure that the system is stable. This approach also requires a complex system calibration setup phase to relate the current levels being sent to the LEDs by the digital potentiometer to the brightness of the LEDs. This is a very complex and often unreliable approach, as formally correlating the perceived brightness of the LEDs to the induced current can only be completed at the end of the hardware design phase, as it requires a completed driver circuit. If changes are required after the hardware is developed, it is a tedious, ground-up process to reassess the hardware to ensure that current control is possible, and the provided current is sufficient for the LED brightness requirements.

2. *Pulse-Width Modulation (PWM) (Chosen Approach):*

The chosen approach for LED brightness control was a pulse-width modulation (PWM) scheme, where no changes in hardware were required to vary the current. The result of using PWM was that the peak LED current remains constant, and the brightness of the LEDs are controlled by varying the fraction of time that the LED is on within each LED-on cycle (Texas Instruments, 2016). Hardware timers in the microcontroller were used to generate a high-frequency PWM signal (20 kHz), where the perceived brightness of the LED from the PWM signal could be set by a numerical parameter in the GUI. This approach was chosen as it is the simplest approach when designing and implementing the firmware, as this approach only uses digital peripherals in the microcontroller itself. This allows brightness of the LEDs to be presented as a simple variable percentage in the GUI, which integrates very naturally with the configuration packet designed to be sent by the GUI to the microcontroller, while avoiding the need for extra analog circuit components.

4.1.2.2. User Interface Approaches

Several approaches were considered when implementing software logic for user interaction and configuration for the system. Three main approaches were considered when implementing a user configuration interface: fixed presets, user-defined text serial commands and a GUI-based configuration. These different approaches are described below.

1. *Static Firmware Presets (Omitted Approach):*

The first considered approach was to use fixed firmware presets, where a small number of LED illumination configurations were to be hard coded into the firmware. This approach would minimize the complexity of the software used for the user interface but would in-hand severely limit the complexity of the system. This approach would make it very difficult to adapt to new system experiments

configurations, where the firmware built onto the microcontroller itself would have to be modified and rebuilt to the board.

2. *Serial Interface (Omitted Approach):*

The second considered approach was a text-based serial command interface, where commands for the microcontroller are to be typed into a terminal, then sent to the board. This approach does allow a bit more flexibility in terms of variability of system configuration parameters, although this approach would not be user-friendly for researchers who are not familiar with command line syntax.

3. *Graphical User Interface (Chosen Approach):*

A GUI-based approach was chosen as the primary approach. This approach allows users to have an easy-to-use interface to control and adjust parameters through widgets (sliders, text input boxes, buttons, etc.), where the GUI assembles a structured configuration packet based on the user's system configuration input. The GUI was implemented using the PyQt6 graphical user interface framework in Python. This approach is optimal due to its ease of use and robustness, where the configuration packet containing the experiment configuration data that sent to the microcontroller is structured in a way that is repeatable and reliable. The user can enter the selected LED flashing mode, frequency, brightness, duty cycle, flash duration and pattern through easy-to-use widgets, where this data is packetized in this structured configuration packet that is read by the microcontroller. The microcontroller then reads and interprets this configuration packet to configure its timers and outputs for an experiment. This approach is much more intuitive for new users and makes it easy to standardize the communication protocol between the laptop used for user input and the microcontroller used to control the system.

4.1.2.3. Triggering and Synchronization Mode Approaches

For system control, two main triggering strategies were considered to drive the system. The first approach was to only use internal triggering, and the second (and chosen approach) was to use internal and external triggering of the system, where these approaches are discussed below:

1. *Internal Triggering Only (Omitted Approach):*

The first approach considered when triggering the LED control system was to use only the microcontroller to drive pulses to start the entire system. This approach would allow the user to decide when to generate camera triggering pulses, based on the state of the LEDs. This approach is valid, although ignores a core deliverable of the system which requires the system to also be triggered by the camera itself. Only using internal triggering would not allow the system to interact with the camera in a two-way communication protocol, which is a possible and completely valid experiment configuration.

2. *Internal and External Triggering (Chosen Approach):*

The chosen approach for the final system design was to write firmware that supports both internal and external trigger modes. An internal trigger mode allows the microcontroller to trigger the system to start capturing fluorescence-based imaging data, where the camera responds to incoming trigger edges to align camera captures with the LED pulses. In external triggering mode, the firmware responds to rising-edge pulses from the camera to trigger the LEDs to begin flashing. This two-way communication protocol is ideal for maximum controllability in experiments and meets all core system logic requirements that are required to be implemented in the systems firmware. The final system allows both modes, where the user can select the mode from the GUI.

4.2. Technical Background

This section describes the fundamental concepts that the LED control system relies on, where concepts in electronics, embedded systems and software design were considered during the development of the control system. This section summarizes the background required to understand the behaviour of the system and the logic behind design decisions made during the projects development.

4.2.1. Timing and Frequency Control

The final system uses low-voltage control signals sent by the microcontroller to high-powered LED driving circuit, which turn the high-voltage LEDs on and off at a user-defined rate. The control signal pulses used to control the LEDs are characterized by both their frequency (*f,* measured in hertz [Hz]) and their period (*T,* measured in seconds [s]), which are related by the simple relationship of:

$$T = \frac{1}{f}$$

For any train of LED control signal pulses, the period *T* represents the time between rising edges of consecutive pulses, and the frequency *f* represents the number of pulses per second. In fluorescence-based cardiac imaging experiments, the required frequency can range from the Hz to kHz range (depending on the system configuration). To drive the high-powered LEDs at the user-defined frequency, firmware written on the microcontroller uses hardware timers on the microcontroller rather than general software delay loops. These hardware timers on the microcontroller are configured to create interrupts periodically at the user-defined operating frequency. When a timer interrupt occurs, the firmware updates the state of the high-powered LED by sending a control signal LED driver circuit, either powering the LED on or off. Using hardware timers provided improved timing accuracy in high frequency switching configurations, which has significant importance when aligning LED pulse signals with the lab camera's external trigger.

4.2.2. Pulse Width Modulation (PWM) and Duty Cycle

To allow the LED control system to have variable brightness (user-defined), a pulse width modulation scheme for brightness control paired with a system-level duty cycle was implemented in the final solution. These are two distinct time scales used in the final implementation, described in Table 2 below:

System Component	Time Scale	Description
Duty Cycle	**Experiment Level**	Controls how long the LEDs are on within each flash cycle of the overall pulse train signal.
Pulse-Width Modulation	**Within Pulse (LED On)**	Controls the brightness of the LEDs while they are on.

Table 2: PWM and Duty Cycle Timing Descriptions

As seen in Table 2 above, the experiment-level duty cycle controls how long the LEDs are on within each flash cycle of the LED signal. The high-frequency LED pulse-level PWM duty cycle controls the brightness of the LEDs while they are on. Both parameters are configurable through the system's GUI through separate percentage slider controls labelled '*Duty Cycle*' (experiment-level control) and '*Brightness'* (pulse-level control). Technical details on these parameters are described below:

1. *Experiment-Level Flash Duty Cycle*

At the experiment level, the LED control signal is a lower frequency train of pulses that are described by its flash frequency (f_{flash}) and flash period (T_{flash}), where the following relationship holds true:

$$T_{flash} = \frac{1}{f_{flash}}$$

Within each flash period, the LEDs are configured to be on for a certain amount of time and off for the rest of the flash period, where the time that the LED is to be on is described by the variable t_{pulse} (LED pulse on time). The flash duty cycle is described by the following relationship:

$$D_{flash} = \frac{t_{pulse}}{T_{flash}} \times 100\,\%$$

This flash duty cycle is a user-defined percentage describing the 'ON duty cycle' of the LED control signal, as it determines how long the LEDs are illuminated (ON) during each cycle of the experiment-level pulse train waveform. For example, at a flash frequency of 20 Hz (50ms flash period), a duty cycle

of 50% means that the LEDs are ON for half of the time in the 50ms flash period, or ON for 25ms and OFF for 25ms.

The microcontroller implements this logic using a hardware timer to generate interrupts at the user-defined flash frequency. Within each LED flash cycle, the firmware turns the LEDs on for t_{pulse} seconds, and off for the remainder of the cycle, where the value of t_{pulse} (in seconds) is calculated with the following formula, where the duty cycle percentage (D_{flash} and flash frequency (f_{flash}) (user-defined parameters in the systems GUI) are inputs:

$$t_{pulse} = \frac{D_{flash}}{100} \times \frac{1}{f_{flash}}$$

Researchers can therefore define how often the LEDs flash, as well as how long each flash lasts, which is important for custom experiment configurations.

2. LED Pulse-Level PWM Duty Cycle

The flash duty cycle described above controls when the LEDs are on at the experiment level, although the brightness of the LEDs during the 'ON' portion (during the time t_{pulse}) is determined using a different form of a duty cycle in a high-frequency pulse-width modulation (PWM) scheme. In the system's PWM scheme, the LED is switched on and off at a very high frequency during the 'ON' time (defined by t_{pulse} above), giving a dimming effect to the LEDs (due to a lower average LED current). In PWM, the LED is turned on and off at a fixed frequency called the 'carrier frequency' ($f_{PWM} = 20\ kHz$ in the delivered control system), having a period of $T_{PWM} = 50\ \mu s$. Within each PWM cycle, the LED is on for a time t_{ON} and off for the rest of the cycle's time. The PWM duty cycle is defined below as:

$$D_{PWM} = \frac{t_{ON}}{T_{PWM}} \times 100\ \%$$

This PWM duty cycle percentage is proportional to the perceived 'brightness' of the LEDs, where a slider parameter is included in the system's GUI as a percentage slider, which changes the value of D_{PWM}. D_{PWM} is strongly proportional to the average LED current (and in-hand, the perceived LED brightness) (Texas Instruments, 2016). In the implemented system, the microcontroller configures a hardware timer (in PWM mode) to generate this high-frequency signal.

This high-frequency signal is the carrier signal of the control signal driving the MOSFET-based switching circuit. When the experiment-level logic (described above) determines that the system should be driving the LEDs 'ON' (based on f_{flash} for t_{pulse} seconds), the firmware enables the PWM output to generate this very high-frequency signal ($f_{PWM} \gg f_{flash}$), where the LEDs are driven with the selected

'brightness' percentage (D_{PWM} percentage). When the LED 'ON' time (t_{pulse}) is over, the PWM output is disabled. This approach using hierarchical levels of user-defined duty cycles allows the system to control when the LEDs flash and how bright they appear during each flash.

4.2.3. Hardware Driving & Switching of LEDs

The LED control system makes use of an IRLZ44n MOSFET to take a control signal from the microcontroller and accurately scale it up to a useable high voltage and current for the LEDs. This is done by applying a DC bias voltage of 24V via a power supply connected to the LEDs in series with the drain of the MOSFET. With this bias voltage applied, the MOSFET immediately goes from the cutoff region (zero current flowing through the LEDs) to the saturation region (high near constant current through LEDs) when the control signal from the microcontroller entering the gate of the MOSFET goes from LOW to HIGH. This allows the control system to have precise switching times while outputting a higher current than the microcontroller can supply on its own, driving the LEDs. A simple illustration of this circuitry's operation is seen in figure 4 of section 7.1.

4.2.4. Graphical User Interface and Serial Communication

The LED control system uses a graphical user interface (GUI) running on a laptop computer located in proximity to the experiment hardware setup, which allows researchers to configure experiments without needing to interact with any software to configure the setup or firmware on the microcontroller. The solution of the GUI was implemented in Python using the PyQt6 framework, which provides a clean and user-friendly approach for configuring experiments. The main window of the GUI contains the following controls, whose values are entered by the user of the system:

- Triggering Mode
- Flashing Frequency (Hz)
- Flashing Duration (s)
- Flashing Pattern
- PWM Brightness (%)
- Duty Cycle (% ON)

The GUI also contains a COM port selection dropdown, a configuration preview pane, a serial monitor display and a 'Start' button to deploy the configuration determined by the user to the microcontroller.

All communication protocols sent from the GUI running on a computer to the microcontroller is performed over a USB COM port, using a text-based 'configuration packet'. When the user presses 'Start' on the GUI after entering their complete configuration, the GUI packetizes the configured parameters into an ASCII string to send over a COM serial line to the microcontroller. The microcontroller reads the

incoming bytes, where it checks that the first three characters in the transmitted ASCII byte string are ‘SET’ (a general header), where the firmware on the microcontroller then parses the received string into the system control parameters (seen above) internally. See section 7.4.1 of this report for a detailed example of this process, step by step.

To ensure that the GUI is responsive to user inputs while continuously monitoring communication through the COM serial port, the GUI software module implements the concept of threading. Using threading separates the processing of the main GUI elements and serial I/O processing from each other and instead processes these two tasks in parallel (simultaneously). A background thread is configured to poll serial traffic coming from the microcontroller to read any messages sent (status updates, error messages, etc.). When a new message is received from the microcontroller, this thread emits a signal to the main GUI thread which updates the serial monitor widget on the running GUI display. The main GUI thread is responsible for handling and parsing any user interactions, where this parallel task execution is required to ensure that the GUI does not freeze due to the blocking of serial read operations when taking user input, thus preventing race conditions when the user is interacting with the GUI.

5. Final Gantt Chart and Workplan

This section will present the final Gantt chart and discuss any deviations from the plan.

5.1. Final Gantt Chart

The final Gantt chart for the first term is seen in figure 1 below, followed by the original workplan for this term (figure 2 below):

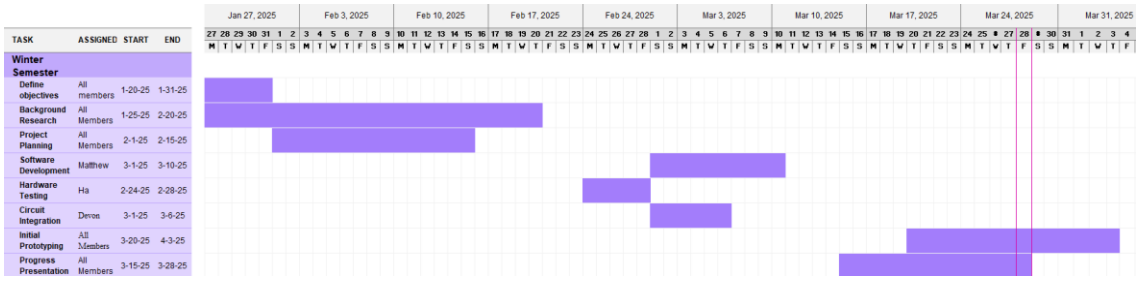


Figure 1: Work Completed in the First Term

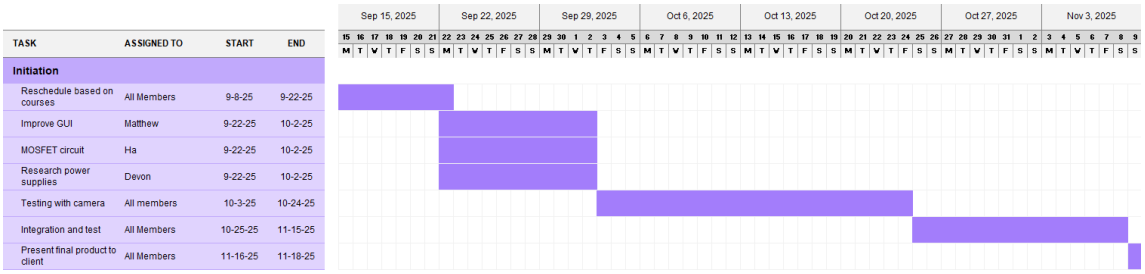


Figure 2: Workplan for Second Term

Figure 3 below shows the actual work completed this term:

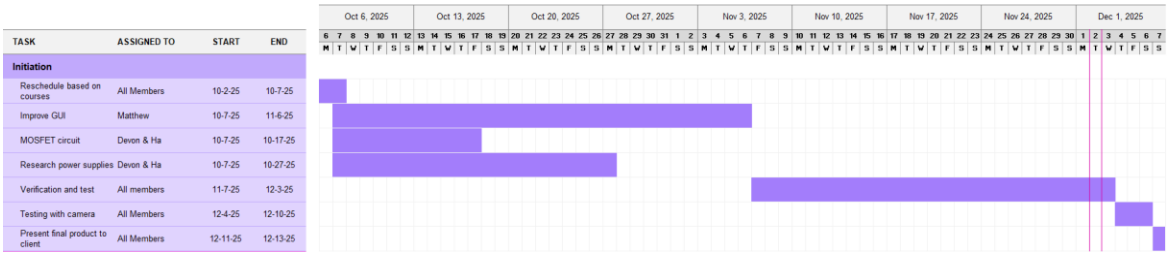


Figure 3: Gantt Chart Showing the Actual Workplan in Second Term

The following subsection (section 5.2) will go into the rationale behind these deviations in workplans.

5.2. Workplan Deviations from Technical Memo

Perhaps the most significant factor in why the workplan has changed for this term is due to the lockout/strike situation at Dalhousie, delaying the start of the term by nearly a month. This is why the new Gantt chart does not begin until the first week of October, in contrast with the planned start on September 8th. Another thing worth noting is that one group member conducted work this term remotely. Although this was straightforward to work through with the bulk of this member's tasks being software-based, it is worth mentioning as it may have influenced scheduling decisions. For one, plans were made to do most of the testing when all three members were physically present.

5.3. Final Workplan Deliverables and Completion Summary

The following table **(**Table 3 below) comprises the deliverables from the workplan and their level of completion at the time of writing this report. It is important to note that this report was due ahead of the project completion date. All deliverables are expected to be 100% complete by the project end date.

Deliverable	Percent Completion
Reschedule workplan	100%
Improve GUI	100%
MOSFET Circuit	100%
Research power supplies	100%
Verification and test	85%

Testing with camera	0%
Present final project to client	0%

Table 3: Deliverable Completion

6. Budget and Task Distribution

This section outlines the planned budget for this project, the cost of the delivered system and the task distribution for different milestones of the project. The budget for the project was allocated to cover hardware components, prototyping materials, and ancillary equipment required for testing and verification. It should be noted that the majority of components incorporated into the final design were sourced from existing inventory of the team or provided directly by the supervisor. Section 6.1 details the budget breakdown, Section 6.2 justifies the component purchases and Section 6.3 outlines the milestones of this report and the task distribution amongst the development of the different subsystems that the control system is comprised of.

6.1. Budget Summary

Table 4 below is a breakdown of all components used in the prototyping and final result of the project. Please note that costs in brackets are theoretical as the team already had those items for use.

Part	Quantity	Cost
IRLZ44NPBF MOSFETs	8	$17.04
Wall Power Supply	1	($15.99)
Barrel Jack Adapter	1	($3.04)
Arduino Nano Atmega328P	1	($11.51)
Breadboard	2	($19.28)
LEDs	4	($91.04)
Perforated Circuit Board	1	($16.25)
Solder Wick	1	$3.05
22 AWG Wire Kit	1	$17.99
Red Banana Plugs	10	$31.30
Black Banana Plugs	10	$31.30
Red Banana Tips	10	($62.60)
Black Banana Tips	10	($62.60)
Chassis	1	($11.81)
Total	-	**$394.80**
Actual Total	-	**$100.68**

Table 4: Project Cost Breakdown

6.2. Cost Savings and Constraints

The decision to use MOSFETs increased component costs compared to BJTs. This purchase was justified by the enhanced durability, faster switching speeds, and reduced thermal stress provided by MOSFETs, which directly improved system performance and longevity. Significant savings were achieved by opting for a perforated prototyping board instead of commissioning a custom PCB. While a PCB would have offered a more polished finish, the perforated board provided reliable soldered connections at a fraction of the cost and avoided long manufacturing lead times. Where possible, existing laboratory power supplies, oscilloscopes, and test LEDs were reused, minimizing the need for new purchases. Twenty-two-gauge wires were chosen for interconnections due to their durability and ability to handle higher voltage and current over extended use. Their thicker cross-section reduces heating and mechanical wear, ensuring stable, long-lasting connections for high intensity LED operation.

6.3. Milestones, Task Distribution & Leaders

Table 5 below demonstrates the group work allocation, where major deliverables are described along with the group member(s) responsible for their delivery in the final system:

Project Area	Task	Responsible Group Member(s)
Circuit Design	Designing all hardware components and determining the overall circuit 'look' on the perforated circuit board. Ensuring that microcontroller pins, LEDs, power supplies, camera inputs/outputs, and MOSFETs were aligned correctly.	Ha Vu and Devon Harvey
Software Design	Development of software and firmware for the GUI, ensuring ease of use for users.	Matthew Watson
Hardware Implementation:	Soldering all hardware components onto perforated circuit board and installing into chassis.	Ha Vu
Testing and System Integration:	Implementing the software with hardware components, ensuring overall control system functions as desired by the client.	Devon Harvey
Final Report:	Preparing and writing final report.	All Members
Final Presentation:	Preparing and presenting final presentation to be presented to clients and faculty advisors.	All Members

Table 5: Group Task Allocation

The task distribution amongst the team can be seen in the Gantt charts, pictured in figures 1-3 of this report (above). Tasks were then divided amongst team members based on their respective backgrounds and skills. Matthew Watson was designated the software lead, being in computer engineering, while Devon Harvey and Ha Vu were designated as system integration and hardware leads respectively as they are in electrical engineering. Matthew worked on developing the firmware and software behind the GUI, while Devon and Ha worked on the high-power LED driver circuit. The team then integrated these two main subsystems and developed the final product.

7. Proposed Solution

This section describes the final LED control system that was delivered to the Quinn Laboratory for use in cardiac imaging experiments. This section provides a general overview of the delivered control system, explains how the hardware and software subsystems interact with each other, and provides a summary of the design choices that allowed the system to meet the project objectives and technical requirements described in sections 2 and 3 (above). Detailed schematics, firmware components and GUI layout diagrams are provided in the appendices of this report and referenced throughout this section accordingly.

7.1. System Overview

The final LED control system was delivered has hardware-software based design which contains a laboratory desktop computer, a microcontroller-based logic control system, a high-power LED driver circuit and a laboratory camera. At a high level, this system enables a user (researcher) to configure all experiment parameters on a graphical user interface, then transmit these parameters to a microcontroller over a connected serial port. These parameters are delivered to the microcontroller in the standardized form of a designed ‘configuration packet’, which is read by the microcontroller for LED driver logic information specific to an experiment. This packet then generates precisely time LED pulse patterns that are synchronized with the camera's exposure signals. The main subsystems of the control system are described in detail in sections 7.3. (hardware) and 7.4. (software & firmware) of this report, but an overview is described at a high level below:

Frontend Subsystem:

A GUI written in Python using the PyQt6 library is ran on a laptop or desktop in proximity to the hardware. This GUI allows the user to define experiment parameters: triggering mode (manual or externally triggered), LED flash frequency, experiment duration, a user-defined LED illumination pattern (L1, L2, L1:L2, etc.), experiment-level duty cycle (% of LED ‘ON’ time per flash period) and LED brightness (PWM duty cycle). The GUI also allows the user to choose a COM port (USB port) to communicate with the microcontroller, a configuration preview and a serial monitor for viewing status messages sent from the microcontroller. The frontend GUI is compiled into a binary executable (.exe) file to ensure all Windows-based users can use the GUI.

Communication Subsystem:

When a researcher begins an experiment, the GUI encodes the selected system configuration parameters into ASCII commands, sent as a sequence ‘configuration packets’, which are text-based binary data packets encoded in ASCII. The two primary commands are:

1. *PATTERN Command (Packet)*

This command informs the firmware on the user-defined LED pattern. This command has the following syntax:

```
PATTERN <LED_Cycle1> <LED_Cycle2> <LED_Cycle2> ...
```

For example, this packet could take on the form “PATTERN L1 L2”, which configures the system to flash LED bank 1, then LED bank 2, then repeat.

2. *SET Command (Packet)*

The ‘SET’ command configures all general operational parameters for the trial, and takes on the following syntax:

```
SET <mode> <frequency> <duration> <brightness> <duty cycle>
```

An example ‘SET’ command packet could take on the form (in ASCII) of “SET 2 500 60 55 30”. These configuration packets are sent over USB to the microcontroller from the user's computer. The reason of separating the two command packets is to create a more flexible communication protocol, where a user can define complex pattern once (by sending a ‘PATTERN’ packet), then run multiple experiments by only changing ‘SET’ commands to change other system parameters like frequency or brightness. Also, the ‘SET’ command is a fixed length packet, whereas a ‘PATTERN’ packet can be very long. By separating these packets into unique protocols, routine adjustments to the systems parameters (like adjusting the brightness from 50% to 51%) would only result in sending the small ‘SET’ command over serial, avoiding having to send a long sequence of bytes (the data within the ‘PATTERN’ packet) over serial for a minor change. The use of a custom communication protocols in the final solution (configuration packets) is ideal as it ensures a simple and repeatable configuration between the user's computer and the microcontroller, while also ensuring that reconfiguration of the system is achievable without changes to the firmware itself.

Embedded Controller Subsystem:

When the embedded controller (microcontroller) receives a configuration packet, the firmware identifies the command by its header (either ‘SET’, indicating that a new system configuration has been received by the microcontroller, or ‘PATTERN’, indicating a new pattern configuration has been received), where the system the parses the ASCII space-separated binary string received over a serial line. The parsed system parameters are then stored in internal variables. Using these received variables, the firmware generates a low-frequency, configurable flash waveform using hardware timers and an

experiment-level (user defined) duty cycle. Hardware timers within the microcontroller are then used to generate a high-frequency PWM waveform for LED brightness control. The firmware supports both an internal trigger mode, where the microcontroller initiates pulse signals to initiate the system to begin, or an external trigger mode, where the microcontroller drives the LEDs based on a received rising-edge signal from the lab camera.

LED Driver / Hardware Subsystem:

The LED driver is a simple circuit that, in essence, takes a control signal from the microcontroller and steps up the current to power the high-power LEDs by using an n-MOSFET with a fast-switching capacity. Figure 4 below shows a simple drawing of how one of these operate. There is one of these for each LED bank.

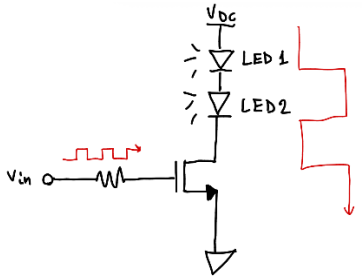


Figure 4: LED Driver Circuit Schematic Drawing

Camera Interface Subsystem:

The control system uses digital I/O lines for camera integration. In internal triggering mode, the microcontroller can output a trigger signal (rising-edge pulse) to the camera while simultaneously driving the LEDs (according to the user-defined system parameters). In external trigger mode, a digital-logic signal from the camera is sent to an input pin on the microcontroller, where the microcontroller schedules LED trigger pulses to align with camera exposure windows.

7.2. System Architecture

The LED control system can be conceptualized as two primary components that are integrated together: the software subsystem and hardware subsystem. This relationship is visualized through the green and red boxes in the system architecture diagram seen in figure 5 below:

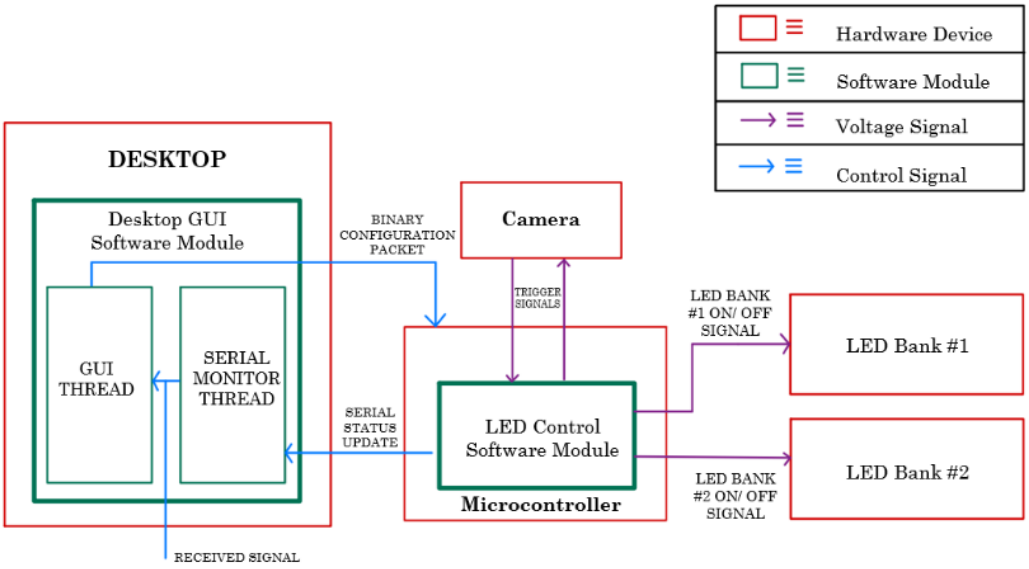


Figure 5: System Architecture Diagram

The high-level system architecture seen in figure 5 above demonstrates the key components, communication pathways and voltage signals that are used to integrate the hardware and software subsystems harmoniously. The operational flow begins when the user interacts with the GUI on the lab desktop, where the user defines all experimental parameters. When an experiment begins, the GUI translates the user's settings into the two-command 'PATTERN' and 'SET' communication protocols, which are sent sequentially over the user-defined COM (USB) port to the microcontroller. This process forms the basis of the communication subsystem, which provides simple data protocols to the embedded controller subsystem. It can be seen from figure 5 above that the GUI performs concurrent tasks using threading, where it polls the COM port for serial communication to update its serial monitor, while taking user input for configuration packetization.

The microcontroller acts as the "brain" for the hardware and is responsible for receiving these commands from the serial connection, as well as interpreting these commands accordingly. Hardware timers are used in the microcontrollers to convert the users' configurations to reliable flashing patterns, which have both high-frequency signal characteristics (a static 20 kHz f_{PWM}) and user-configured frequency characteristics (for the experiment-level flash duty cycle, which is a variable set by the user). The approach of using hardware timers and interrupts in the microcontroller ensures that LED timing accuracy does not depend on software loops. These logic-level signals (0V-5V signals) are sent to the LED driver circuit subsystem, where the microcontroller acts as a logic controller for the high-power MOSFET switch. The LED driver hardware uses a MOSFET to control the flow of high-current from the

main power supply to its respective LED bank. These LED driver circuits effectively amplify the 5V high-frequency logic-level pulse trains sent by the microcontroller.

As seen in figure 5 above, trigger signals are sent to and from the camera to the microcontroller. In ‘Manual Mode’, the microcontroller sends a digital trigger pulse to the camera to signal the start of an exposure, where the microcontroller ensures that the LEDs are active when the camera is ready. In ‘Trigger Mode’, the camera sends a logic signal to the microcontroller which immediately starts executing the user-configured LED flash pattern. The system architecture takes a layered approach, where software running on the user's PC handles configuration of the system, while the microcontroller handles the high precision timing for the experiment. This layered architecture ensures the system is user-friendly, responsive and meets all timing requirements for cardiac imaging experiments.

7.3. Hardware Architecture Overview

The hardware solution was very straight forward as components chosen to balance performance, durability, and suitability for prolonged laboratory use. The IRLZ44N MOSFET was specifically chosen over other MOSFETS and BJTs as it combines logic level gate operation with high current capacity, making it directly compatible with microcontroller outputs while still capable of driving high intensity LEDs. Unlike standard MOSFETs that require higher gate voltages, the IRLZ44N can be fully switched on with 5 V logic, eliminating the need for additional driver circuitry. Its low resistance minimizes power loss and heat generation, while its fast-switching speed ensures precise synchronization with highspeed imaging systems. Packaged in a durable TO220 form factor, the IRLZ44N also provides excellent thermal performance, allowing the circuit to operate reliably under prolonged experimental conditions. These characteristics made it the most practical and cost-effective choice compared to alternatives, ensuring both efficiency and durability in the final design.

Refer to Appendix C of this report for the circuit schematic and Appendix D of this report for the board layout. The circuit schematic was built using LT Spice. The schematic illustrates a microcontroller-driven LED control circuit using two IRLZ44N MOSFETs. An Arduino Nano serves as the control unit, sending digital signals from pins D10 and D9 to the gates of the MOSFETs through 330-ohm resistors, which limit gate current and protect the microcontroller. The MOSFETs act as low-side switches, controlling the ground path for two pairs of high-intensity LEDs powered by separate 24V supplies. When activated, the MOSFETs allow current to flow through the LEDs, illuminating them in sync with the Arduino’s output. Ground connections are shared between the Arduino and the power circuitry to ensure proper gate referencing. This configuration enables precise, high-current LED control using low-voltage logic, making it suitable for applications like fluorescence-based imaging where timing and intensity must be tightly regulated.

It should be noted that the chassis carrying the circuit has future improvements, where the body of the chassis will include drilled banana plug holes for easy voltage connection of the LED driving circuit (see Appendix E of this report for a visualization photo of these future chassis changes).

7.4. Software Architecture Overview

This section describes a detailed description of the development of the software used for user input and system configuration.

7.4.1. Overall Software Architecture

The approach to developing the overall software system was to have a software module running on a desktop located in proximity to the experiment setup, then have another software module running on the microcontroller which controls the LED system during the experiment. The purpose of software module running on the desktop is to provide a graphical user interface (GUI) that takes experiment configuration parameters from a researcher running an experiment. This software then transmits the entered experiment configuration data to the microcontroller, which runs the experiment according to the transmitted system parameters.

To do this, a set of data packets were designed and used, which the team titled ‘configuration packets’. These packets of data are composed of a header followed by user-defined system parameters. The parameters within these packets are provided by the user on the desktop, where the six system parameters are entered by the user into the GUI. These parameters are packetized by the software running on the desktop into binary ‘configuration packets’, which are then transmitted via a specified serial port (chosen by the user in the GUI) to the microcontroller. The microcontroller then decodes the received configuration packet’s sequence of bytes into the system parameters, where the experiment is then run on the microcontroller according to this decoded configuration data. A block diagram description of an example set of configuration packets is seen in figure 6 below:

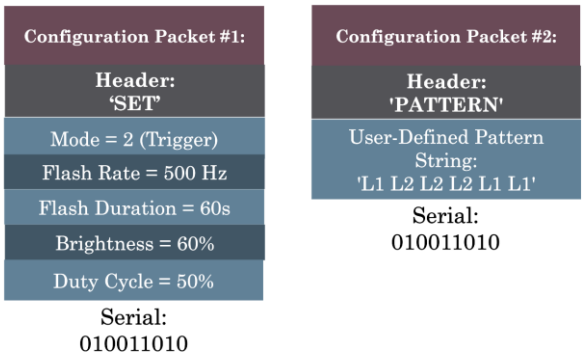


Figure 6: Block Diagram Representation of an Example Set of Configuration Packets

It can be seen from figure 6 above that two configuration packets were used in the final design, where the headers of these packets are ‘SET’ and ‘PATTERN’. The ‘SET’ command is a fixed-length packet containing six integer parameters: mode, flash rate, flash duration, flash pattern, brightness and duty cycle. These six parameters are selected by the user and are required to run an experiment. The ‘PATTERN’ command is a variable-length packet that defines the potentially complex sequence of LED activations. These packets are separated to ensure the system only re-transmits the potentially long, variable-length ‘PATTERN’ command when the user selects a new LED pattern. The short and fixed length ‘SET’ command is used to start the experiment.

In the example above (figure 6 above**),** the user would have defined a new flash pattern of ‘L1 L2 L2 L2 L2 L1 L1’, where this is a custom, user-defined pattern of the two LED banks (L1 and L2) that is repeated throughout the duration of the experiment. This packet would be transmitted as the ASCII equivalent string of ‘PATTERN L1 L2 L2 L2 L2 L1 L1\n’, where the newline terminator at the end of the packet indicates the end of the packet. After this initial packet is sent, a second packet is sent (the ‘SET’ command), where the user would have selected the experiment mode to be ‘Trigger Mode’ (the LED system is triggered by an external signal outputted by the camera), where the LEDs are to flash at 500 Hz for 60s with a 60% brightness (60% ON time for PWM carrier frequency signal) and a 50% on duty cycle (the LEDs are ON for 50% of a flash period).

The parameter description and user input method along with how the microcontroller differentiates/ interprets a parameter is seen in Appendix F of this report for the system parameters, where the packet a parameter is applicable to is seen in the ‘Command Packet’ column in the table seen in Appendix F.

A block diagram representation of the data entry/ encoding/ decoding process completed by the two software modules on the laptop and microcontroller is seen in figure 7 below:

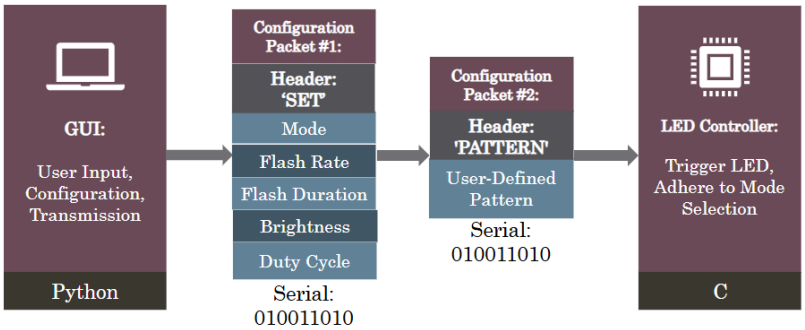


Figure 7: Block Diagram of Configuration Packet Generation, Transmission and Interpretation

Figure **7** above displays the case where both a new LED flashing pattern and system configuration are being sent to the microcontroller, where the ‘PATTERN’ packet is sent before the ‘SET’ packet. It is observed from above that the GUI (running on the lab’s desktop) takes user input of the system parameters, then packetizes this data into two ASCII-encoded strings (for the ‘PATTERN’ and ‘SET’ commands), which is are then transmitted as a sequential series of bytes (serial transmission) to the microcontroller over a serial port (USB/ COM).

When the microcontroller receives the first series of bytes (corresponding to a ‘PATTERN’ packet’s ASCII-encoded string), it decodes the string as a series of LED banks, where it decodes bytes of serial data of the string ‘L1’ or ‘L2’ separated by spaces until it reaches a newline character (‘\n’) indicating the complete command has been received. The microcontroller then stores the decoded LED configuration tokens sequentially in a dedicated array in the microcontroller's memory. This array in memory is used as a flash sequence that the system will execute whenever the microcontroller receives a subsequent ‘SET’ command that instructs the microcontroller to begin a flashing trial. An example can be used from figure **7** above, where the microcontroller would receive the string ‘PATTERN L1 L2 L2 L2 L2 L1 L1\n’, parse this received ASCII string, then store the character array ‘1222211\0’ in the microcontrollers flash memory, indicating the active flash pattern. The ‘L’ in front of the LED pattern are stripped in the stored array for efficiency when reading the array sequentially in the microcontrollers flash memory.

After this, the microcontroller receives the first series of bytes (corresponding to a ‘SET’ packet’s ASCII-encoded string), it decodes the first 3 bytes (the first 3 ASCII characters) and checks if they are equal to the ASCII equivalent of “SET” (the header string of the parameter input packet). Only if this is true, the microcontroller begins decoding the other system parameters are read in as an ASCII string, where parameters are separated by spaces. This is best described by using the example in figure 7 above, which would transmit the ASCII string “SET 2 500 60 60 50\n” over serial to the microcontroller. The microcontroller would interpret this as a new system parameter configuration (due to the “SET” header) that runs with trigger mode (due to the “2” following the first space after “SET”) at 500 Hz (due to the “500” following the second space in the received string) for 60 seconds (due to the “60” following the third space in the received string) with a PWM duty cycle (brightness) of 60% (due to the “60” following the fourth space in the received string) and an LED ON-duty cycle of 50% (LED is ON for 50% of a flash cycle) due to the “50” following the fifth space in the received string.

7.4.2. Graphical User Interface Overview

The software solution used for system configuration is a graphical user interface developed in Python using the PyQt6 framework, which is the primary method of system configuration. The delivered

GUI with block descriptions is seen in Appendix G of this report. From Appendix G, it is seen that the GUI prototype has 10 different components, with 6 of these components being reserved for system configuration parameter input (mode, pattern, rate, duration, brightness and duty cycle). These components are explained below:

- COM Port Connection:
    - A dropdown that the user can use to select the USB port that interfaces with the microcontroller
- Triggering Mode:
    - Button that allows the user to choose between manual and trigger mode.
- Flash Pattern:
    - Button that allows the user to choose their desired flashing pattern.
- Flash Rate:
    - Text box that allows the user to enter the flash rate (integer) in Hz.
- Flash Duration:
    - Text box that allows the user to enter the flash duration (integer) in seconds.
- LED Intensity/ Brightness:
    - Slider that allows the user to control the brightness of the LEDs from 0% to 100%.
- Flash Duty Cycle:
    - A horizontal slider (1% to 99%) that controls the on-time for an individual flash of an LED within the user-defined pattern.
- Configuration Preview:
    - Shows a summary of the six user-selected configuration parameters before packet deployment to the microcontroller.
- Packet Deployment:
    - This allows the user to deploy their system configurations to the microcontroller to run an experiment.
- Serial Monitor:
    - Displays messages received by the microcontroller and system status messages sent by the GUI to the microcontroller (monitors COM port traffic and displays this information to the user).

7.4.3. GUI Software Overview

To implement a functional GUI which takes user input specific to a certain intended configuration of an experiment, the PyQt6 framework was implemented. The Qt6 graphical user interface framework is a set of C++ libraries used for developing user interfaces for desktop platforms, where PyQt6 is the Python binding of the framework (The Qt Company, 2025). The GUI was developed purely in Python using the Visual Studio Code development environment.

The GUI software module itself uses two main classes which define the system architecture: ‘serialclass’ and ‘systemGUI’ (see Appendix H for the GUI software layout, where the class definitions are visible bit hidden). As seen in Appendix H at the end of this report, the Python class ‘serialclass’ was designed to handle serial communication with the microcontroller, running in a background thread (more about threading in this software module is seen below). This class (‘serialclass’) continuously polls the user-specified COM port for incoming serial data from the microcontroller, using Qt signals to communicate messages from the polling thread to the main GUI thread. The second class, the ‘systemGUI’ class performs all the GUI logic (allows users to interact with all the 10 components seen in figure **7** above).

It should be noted that threading is used in this software module (using Python’s ‘threading’ library), where two separate threads are running parallel processes. The first thread polls the COM port for incoming serial data from the microcontroller, and the second thread is responsible for the general performance of the GUI. These threads communicate with each other using PyQt6 signals, where if a new message arrives over \the serial port, the serial polling thread sends a signal to the GUI thread that a new message has arrived, where the GUI thread can then read this message and display it to the serial monitor.

Threading in this software module is required as without this architecture, the ‘arduino.readline()’ command in our GUI software module (used to read serial data from the microcontroller) would cause the entire GUI application to freeze as this command waits indefinitely for input. Using threading ensures that communication with the Arduino runs in parallel with the GUI software, which ensures a fluid and interactive user interface.

It is also noted that the GUI only sends a ‘PATTERN’ packet to the microcontroller when the user defines a new LED flashing pattern configuration. This is to avoid consistently sending potentially long flashing patterns over the serial port, when the user may only want to change one system parameter (such as changing the flashing frequency from 50 Hz to 51 Hz). This design approach minimizes unnecessary serial traffic and makes the interpretation of the users' parameters much easier from a firmware perspective. Two packet transmission protocols are implemented in the GUI software, and two packet

reception protocols are implemented in the microcontroller's firmware (separated for 'SET' and 'PATTERN' protocols). This separation is because data parsing logic for a 'PATTERN' packet is much different than a 'SET' packet, as a 'PATTERN' packet has variable length, and a 'SET' packet has a fixed length. To separate the parsing logic efficiently, these data structures into these two respective transmission protocols.

7.4.4. Firmware Overview

The firmware for the ATMega328P microcontroller was developed using Arduino Integrated Development Environment in Arduino's native language (which resembles C++). The objective of the firmware was to receive the user-defined experiment parameters from the GUI and interpret these parameters to execute an LED flashing sequence with absolute timing precision. To obtain this high level of timing precision, hardware timers in the microcontroller were used. The firmware was designed to operate independently of the PC receiving the user command inputs, which ensured that timing of the LED flashing sequence was not compromised by serial communication latency. It should be noted that the firmware avoids high-level blocking functions like the 'delay()' function, and instead directly manipulates the microcontrollers hardware registers. Using software timing functions (like the 'delay()' or 'millis()' function) results in a very inaccurate LED pulse train waveform in the output, which in testing only supported a maximum frequency output of 137Hz. This is the reason behind using the ATMEGA328P's hardware timers for LED timing.

The 16-bit Timer/Counter1 peripheral is used for system timing, which generates a high-frequency PWM signal (20 kHz) and provides a time base for the LEDs flashing pattern (Atmel, 2015.). The timer is set to 'Fast PWM Mode' by configuring the Waveform Generation Mode (WGM) bits in the peripheral's control registers (TCCR1A and TCCR1B). In 'Fast PWM Mode', the PWM signal has a static frequency of 20 kHz, which is completed by loading a calculated value into the ICR1 register (Input Capture Register 1). The ICR1 register defines the timer's maximum count value, which sets the period of the PWM wave. The LED intensity is controlled by setting the OCR1A and OCR1B (Output Compare Registers), which define the PWM duty cycle. These registers get a value that is compared against the timer's count, where when the timer matches the value in a OCR1X register, the output pin is toggled (resulting in a configured PWM duty cycle). The PWM duty cycle defines the brightness of the LEDs, where the user-defined brightness percentage parameter in the GUI is scaled and used to set the OCR1A and OCR1B registers.

The Timer/Counter1 peripheral is also used to provide a time base for the LED pattern sequence, where it uses the Timer1 Overflow Interrupt. To enable this interrupt, the firmware sets the TOIE1 bit (overflow interrupt enable bit) in the TIMSK1 (Timer/ Counter1 Interrupt Mask Register). This causes the

‘TIMER1_OVF_vect’ interrupt service routine (ISR – a function) to be called every time the timer overflows, which is 20,000 times per second (due to our defined PWM signal of 20 kHz) (see Appendix I at the end of this report the commented code for this ISR). Inside this ISR, a software counter tracks the count of interrupts that have occurred. When the counter reaches a calculated threshold corresponding to the user-defined LED pattern duration, the firmware will advance to the next stage in the user-defined pattern sequence. This ISR also manipulates the TCCR1A control register by setting or clearing the COM1A1 and COM1B1 (compare output mode bits) bits to connect or disconnect the PWM signal from the LED output pins (PIN 9 & 10 on the ATMEGA328P microcontroller, where COM1A1 is used to set/clear LED bank 1 (PIN 9) and COM1B1 is used to set/clear LED2 (PIN 10)). This method of toggling the LED’s is very fast (unlike using slow software functions like ‘digitalwrite()’).

To receive an external signal from the camera, the firmware includes a trigger that uses an external hardware interrupt. The ‘attatchInterrupt’ function configures PIN 2 on the microcontroller to trigger on a falling edge (camera trigger signal). When this happens, the microcontroller pauses whatever it’s currently doing to execute the ‘handle_trigger’ function (ISR). This approach avoids needing to continuously poll the trigger pin, which takes CPU clock cycles to do (resulting in potentially inaccurate system timing), and rather allows the system to instantaneously respond to the external camera trigger signal.

The main loop of the firmware is a non-blocking element in the code, where its only function is to monitor for configuration packets incoming from the GUI and check the status of the system’s trigger (‘trigger_fired’ flag in the ‘handle_trigger’ function/ ISR). This design allows the microcontroller to dedicate its CPU resources to hardware timers, ensuring that the LED flashing sequence is executed accurately. This is critical as it ensures that the lab camera is synced with the LEDs (according to the user inputs in the GUI).

8. Results and Verification

This section of the report describes the methodology and outcomes of system validation testing. The objective of this phase was to provide evidence that the system successfully meets all technical requirements outlined in section 3 of this report. The following subsections detail the specific test procedures executed for each technical requirement and a summary of the results which verify the performance of the system.

8.1. Verification Against Technical Requirements

This section describes the results of the tests against the technical requirements, outlined in section 3 of this report. This is detailed in Table 6 below:

<table>
<tr><th>Technical Requirement ID</th><th>Test</th><th>Verification Results</th></tr>
<tr><td>TR-1</td><td>Configurable LED flash frequency</td><td>The system was able to configure LED flash frequencies at 1 Hz, 10Hz, 100Hz, 1000Hz, and 2000Hz. Oscilloscope results of LED output period from microcontroller were captured, where all frequencies were captured as expected. Please refer to appendix X for the captured pulse trains over this series of frequencies.</td></tr>
<tr><td>TR-2</td><td>Configurable experiment duration</td><td>The timing of various pulse trains were measured using an oscilloscope. It was observed that the timing aligned with duration requested from the GUI.</td></tr>
<tr><td>TR-3</td><td>Variable Illumination Patterns</td><td>Confirmed by oscilloscope probing at the two LED banks for each pattern. Results showed the correct patterns were visible on oscilloscope, and visual inspection of pattern was confirmed over 10 consecutive cycles.</td></tr>
<tr><td>TR-4</td><td>Brightness Control</td><td>The oscilloscope was used to measure the PWM frequency and found that with 25%, 50%, and 75% brightness, the PWM carrier frequency was approximately equal to 20 kHz.</td></tr>
<tr><td>TR-5</td><td>External Trigger Synchronization</td><td>An oscilloscope was used to observe the external trigger signal. The delay was measured to be 1.94ms.</td></tr>
<tr><td>TR-6</td><td>Low Latency GUI Start in Manual Mode</td><td>The time between the button on the GUI being pressed and the first observed flashes was timed using an external timer, where the average latency time was much less than 2s.</td></tr>
<tr><td>TR-7</td><td>Safe Current Operating Limits</td><td>With maximum duty cycle (99%) and highest possible frequency (20KHz), the current was found to be 208.097mA, which is well within safe operating range.</td></tr>
<tr><td>TR-8</td><td>Continuous Operation</td><td>A 2-minute test was ran whilst monitoring signals and current. During and after completion it was observed that there were no abnormalities.</td></tr>
<tr><td>TR-9</td><td>Usability for New Users</td><td>5 users were asked to configure a simple experiment where they could select COM port, set frequency, duration of experiment, pattern, brightness and start run. Below is a summary of users, their task completion times, and any notes.
<table>
<tr><th>Users</th><th>Task Completion Time</th><th>Notes</th></tr>
<tr><td>1</td><td>3 minutes</td><td>Wrong COM Port Selected on first try.</td></tr>
<tr><td>2</td><td>2 minutes</td><td>100% Success</td></tr>
<tr><td>3</td><td>2:37 minutes</td><td>100% Success</td></tr>
<tr><td>4</td><td>4:03 Minutes</td><td>100% Success</td></tr>
<tr><td>5</td><td>3:27 minutes</td><td>100% Success</td></tr>
</table></td></tr>
</table>

Table 6: Verification Against Technical Requirements

8.2. Test Procedures, Measurements and Results

To validate that the delivered LED control system met the technical requirements outlined in Appendix B of this report, a series of quantitative and qualitative tests were completed. Table 6 above summarizes these tests. The primary test equipment included a Rohde & Schwarz RTC1002 Oscilloscope, a function generator and a simple digital multimeter for current measurements. The following subsections describe the test acceptability criteria, procedure, measured results, and final pass/fail conclusion for each technical requirement.

8.2.1. Frequency Accuracy Test (TR-1)

Test Acceptability Criteria:

This test quantified the precision of the firmware's hardware timer, where the test was a 'pass' if the period of all pulses were $\pm 5\%$ of the ideal period, T (1/f) for frequencies f = 10, 100, 1000 & 2000 Hz.

Procedure:

1. The output for LED bank 1 and 2 are connected to channel 1 and 2 on the oscilloscope.
2. Using the GUI, the system was configured to send test LED signal flash frequencies at 10 Hz, 100 Hz, 1000 Hz and 2000 Hz (the upper limit frequency of the system).
3. The oscilloscope was then paused during execution to capture the pulse train, and the frequency of the pulses were recorded & compared with the user-entered operating frequency.

Test Results:

Figure J.1 in Appendix J of this report shows the captured waveforms for frequencies of f = 10, 100, 1000 and 2000 Hz, labelled respectively. The summarized test results for these waveforms are seen in table 7 below:

LED Bank	Configured Frequency [Hz]	Configured Period (s)	Measured Frequency	Measured Period	% Difference of Measured vs. Actual Period
LED Bank 1	10	0.1	10	0.1	0.0000%
LED Bank 1	100	0.01	99.94	0.010006004	0.0600%
LED Bank 1	1000	0.001	999.6	0.0010004	0.0400%
LED Bank 1	2000	0.0005	2000	0.0005	0.0000%
LED Bank 2	10	0.1	10	0.1	0.0000%
LED Bank 2	100	0.01	99.97	0.010003001	0.0300%
LED Bank 2	1000	0.001	999.5	0.0010005	0.0500%
LED Bank 2	2000	0.0005	2000	0.0005	0.0000%

Table 7: Summary of Test Results for Frequency Accuracy Test (Test #1)

Result Conclusions (Pass/ Fail):

Because the period of all the pulses had a negligible deviation less than 5 percent from the ideal period (as seen in table 7 above), the system passed this test and therefore meets the technical requirement TR-1.

8.2.2. Timing Accuracy Test (TR-2)

Test Acceptability Criteria:

This test quantified the precision of the firmware control on the configured experiment time, where the test passed if the measured experiment time is than ± 2 % of the configured time.

Procedure:

1. The output for LED bank 1 was probed with the oscilloscope.
2. A 10 Hz, 1s long pulse train was sent over the LED bank 1 trigger line.

3. The width of the entire pulse train was measured from the rising edge of the first pulse to the falling edge of the last pulse.
4. This measured duration was compared to the configured 1s duration sent by the user on the GUI, where the difference should be less than ± 2 % of the configured time (1s).

Test Results:

Figure J.2 in Appendix J of this report shows the output of probing a 1s long waveform (where the user entered 1s as the experiment time). It can be observed from this figure that the output waveform is exactly 1s long from the cursor measure feature of the oscilloscope.

Result Conclusions (Pass/ Fail):

Because the measured experiment time is less than $\pm 2\%$ of the configured time (a 0% deviation was observed), the system has passed this test, meaning that the system adheres to technical requirement TR-02.

8.2.3. Illumination Pattern Test (TR-3)

Test Acceptability Criteria:

This test ensures the correct logical operation of four different user-defined illumination patterns, where it is a pass if this is achieved.

Procedure:

1. Connect microcontroller outputs for LED bank 1 and LED bank 2 to two separate channels on the oscilloscope.
2. Select a user-entered pattern from the GUI (ex – L1 L2), as well as a simple LED configuration (ex – 100 Hz, 50% duty cycle).
3. For each pattern, observe the waveforms on the oscilloscope and verify that the LED flashing sequence and timing of the pulses on both channels match the patterns logical user definition.
4. Confirm that the pattern is stable and repeats correctly over 10 LED flashing cycles.
5. Repeat steps 2-4 with three new LED flashing patterns.

Test Results:

The results of this test are seen in figure J.3 of Appendix J, where four tests of pulse trains were sent to the microcontroller. These four LED flashing sequences were ‘L1’, ‘L1 L2’, ‘L1 L1 L2’ and ‘L1 L1 L1 L2’, where ‘L1’ and ‘L2’ refer to LED banks 1 and 2, respectively. It can be seen from figure J.3 in Appendix J that the oscilloscope traces prove that the four expected flashing sequences were observed for

a 100% brightness level, 99% duty cycle, and 100 Hz LED pulse train configuration sent by the microcontroller.

Result Conclusions (Pass/ Fail):

Because the four expected LED pulse sequences were observed in the systems output, the system passed this test, meaning that the delivered LED control system meets TR-3.

8.2.4. PWM Brightness Control Test (TR-4)

Test Acceptability Criteria:

This test ensures that the PWM carrier frequency for brightness control was equal to 20 kHz, with a $\pm 5\ Hz$ margin of error, where it is a pass if it meets this criteria.

Procedure:

1. Connect an oscilloscope probe to the output of an LED.
2. Configure the system output to a signal with a 25% brightness level.
3. Zoom in on a pulse and measure the PWM carrier frequency displayed.
4. Repeat steps 2-3 for brightness levels of 50% and 75%.
5. Confirm that all carrier frequencies observed are stable and appropriately equal to 20 kHz.

Test Results:

The above procedure was completed on the output of LED bank 1, where figure J.4 in Appendix J shows the PWM-level oscilloscope output traces for brightness levels of 25%, 50% and 100%. The strong relation between the observed brightness and the PWM duty cycle is observed in these oscilloscope traces, where all three traces h traces, where all three traces have an observed carrier frequency of within 5 Hz of 20 kHz (all three traces were 19.99 kHz, which is within this margin of error).

Result Conclusions (Pass/ Fail):

Because all three traces observed in the results have a PWM carrier frequency within 5 kHz of 20 kHz, the system passed this test, meaning it meets TR-4.

8.2.5. External Trigger Synchronization Test (TR-5)

Test Acceptability Criteria:

This test ensures the delay between the microcontroller receiving an external trigger signal and the start of the LED pulse sequence is less than 5 ms, where the test was a pass if this condition is true.

Procedure:

1. Connect a function generator to the systems external trigger input as the trigger source.
2. Connect channel 1 of the oscilloscope to the trigger input line.
3. Connect channel 2 of the oscilloscope to the systems LED 1 output channel.
4. Configure the system (in the GUI) to be in ‘Trigger Mode’, and a pulse sequence starting with ‘L1’.
5. Set the oscilloscope trigger on falling edge of channel 2.
6. Measure the time difference between the falling edge of the trigger signal on channel 1 and the rising edge of the first LED pulse on channel 1 using the cursors on the oscilloscope.

Test Results:

Figure J.5 in Appendix J of this report demonstrates the above procedure in action, where a measurement between the falling edge of the trigger signal on channel 2 of the oscilloscope and the rising edge of the first LED pulse on channel 1 of the oscilloscope was measured to be 1.94 ms.

Result Conclusions (Pass/ Fail):

Because the measured time between the microcontroller receiving an external trigger signal and the start of the LED pulse sequence is less than 5 ms (measured 1.94 ms), the system has passed this test, meaning the system meets the technical requirement TR-5.

8.2.6. Low Latency System Start in Manual Mode Test (TR-6)

Test Acceptability Criteria:

This test ensures that the system starts the illumination of the LEDs in less than 2 seconds after receiving a 'Start' command from the GUI in manual mode, where it is a pass if this condition is true for 5 consecutive trials.

Procedure:

1. Configure the system to be in ‘Manual Mode’ on the GUI and send a LED configuration to the microcontroller.
2. Simultaneously press the ‘Start’ button on the GUI and start a timer, stopping the timer when the first LED flash is observed, recording the time.
3. Repeat test 5 times the recorded time, ensuring all test results are less than 2s.

Test Results:

The procedure outlined above was executed on the system, where Table 8 below demonstrates the measured latency time for the 5 test trials.

Trial	Measured Latency	Less than 2 second latency time?
1	0.19 ms	Yes
2	0.34 ms	Yes
3	0.63 ms	Yes
4	0.22 ms	Yes
5	0.43 ms	Yes

Table 8: Measured Latency Times for Test # 6

Result Conclusions (Pass/ Fail):

Because the system started the illumination of the LEDs in less than 2 seconds after receiving a 'Start' command from the GUI in manual mode over 5 trials, the system passed this test, meaning that the system adheres to the technical requirement TR-06.

8.2.7. Operating Current Limit Test (TR-7)

Test Acceptability Criteria:

This test measured the systems current consumption under high load conditions and verified it remains within safe operating limits, where the test is a pass if the recorded DC current is less than the rating in the LED datasheet (for the OSRAM LED ENGIN LuxiGen™, LZ4-40CW08 LED used in the lab, this max current rating is 1000 mA) (DigiKey Electronics, n.d.-d).

Procedure:

1. Insert a multimeter in series with the main power supply line for the LED bank 1 driver circuit and configure the multimeter to measure DC current.
2. Configure the system using the GUI to run at a duty cycle of 99% with a brightness level of 100%, ensuring that LED1 is activated in the experiment configuration.
3. Activate the LED output and record the current measurement on the multimeter.
4. Confirm that the recorded value is less than the current rating in the LED datasheets.

Test Results:

The system was configured to have a maximum duty cycle (99%) and brightness level (100%), where the main power supply was probed with a multimeter. Figure 8 below shows the DC amperage read from the multimeter:

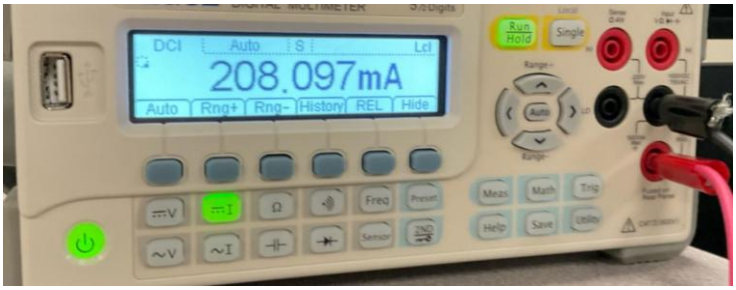


Figure 8: DC Amperage Read from Multimeter for Test #7

It can be seen from figure 8 above that the LEDs operate under 208.09 mA under the 'worst' possible conditions.

Result Conclusions (Pass/ Fail):

Because the current drawn by the LEDs under the 'worst' possible system conditions was found to be 208.097 mA, which is far below the maximum rated current of the lab LEDs of 1000 mA, the system passed this test, meaning that it met the technical requirement TR-7.

8.2.8. Continuous System Operation Test Procedure (TR-8)

Test Acceptability Criteria:

This test ensures the system remains stable over a long period of time, where this test is considered a 'pass' if the system operates normally over a 2-minte time duration.

Procedure:

1. Configure the system to run at a typical state (10 kHz, 50% duty cycle) for 2 minutes (120 seconds).
2. Start a timer simultaneously when the system begins running.
3. Ensure that system operates as expected for the entire 2-minute duration as expected, noting any signs of instability.

Test Results:

Upon configuring the system according to the above procedure, there were no issues found after a 2-minute period.

Result Conclusions (Pass/ Fail):

Because the system was able to perform its intended function over a 120-second (2 minute) period, the system passed this test, meaning that it adheres to the technical requirement TR-8.

8.2.9. New User Usability Test Procedure (TR-9)

Test Acceptability Criteria:

This test ensures that the system is intuitive to use for new users, where it is a 'pass' if 4/5 volunteers can fully configure and run the system without prior system knowledge.

Procedure:

1. Get 5 test participants with no experience using the control system.
2. Give them a simple task like running a 30 second experiment at 50 Hz.
3. Observe the time the user takes to do the task and their performance.
4. Record completion time, note any challenges they encountered and confirm they successfully completed the given task.

Test Results:

The test results for this test are seen in Table 9 below:

User	Task Completion Time	Notes
1	3 minutes	Wrong COM Port Selected on first try.
2	2 minutes	100% Success
3	2:37 minutes	100% Success
4	4:03 Minutes	100% Success
5	3:27 minutes	100% Success

Table 9: Summarized Results for System Usability Test

It can be seen from Table 9 above that 4/5 participants completed the test procedure outlined above with no issues.

Result Conclusions (Pass/ Fail):

Because 4/5 participants were able to fully configure and run the system without prior system knowledge, the system passed this test, meaning the LED control system adheres to the technical requirement TR-9.

8.3. System Validation & Demonstration

After all technical requirements were verified, a comprehensive system validation stage was completed. This demonstration validated the systems applicability to its intended final application by

simulating a complete and realistic user workflow. This general test confirms that the subsystems of the LED control system work harmoniously, where a reliable and intuitive solution for controlling LED illumination was to be provided to the lab.

For the demonstration, a typical experimental scenario was executed, which included configuring a 2-minute (120 second) LED illumination sequence using a user-defined LED flashing pattern (‘L1 L2 L1’) at an operating frequency of 75 Hz, with a duty cycle of 50% and brightness level of 80%. These parameters were entered into the GUI, where the experiment was initiated in ‘Manual Mode’ by pressing ‘Start’. The system responded immediately to the initiation (pressing ‘Start’), where both LED banks flashed according to the user input. The 120-second run was executed flawlessly, where a stable output was observed throughout the entire experiment, and the simulated experiment terminated after the coordinated 120 second stop time.

The successful execution of this validation scenario is a result of the system/ subsystem components meeting all design requirements. Because the system started immediately after pressing ‘Start’, the low-latency performance requirement (TR-6) was proven to be met. The stable 75 Hz signal was guaranteed by the absolute accuracy of the frequency outputted by firmware on the microcontroller (TR-1), while the custom ‘L1 L2 L1’ sequence was observed, meeting the pattern logic requirements (TR-3). The 80% brightness level was observed, where this brightness was accurately maintained by the PWM scheme provided by control systems firmware (TR-4) without obvious power consumption discrepancies (TR-7). The ability to run for the full 120-second duration without discrepancies was observed (meeting TR-8), where timing accuracy (TR-2) was also observed. The entire task, from user configuration to the execution of the control system was intuitive, which validated the positive results from the user usability study (TR-9).

A second run of this configuration in ‘Trigger Mode’ was completed with using a pushbutton on a simple breadboard to simulate the falling edge camera trigger, where no delay was observed during the experiment (TR-5). This secondary experiment had the same outcome as above, where all system technical requirements were observed over the 120 second test.

This detailed demonstration proved that the LED control system performs as expected, meeting all technical requirements. This successful validation scenario proves that the delivered system can function for its intended purpose, where all high-level user requirements were translated into a simple graphical user interface and reliable LED control system output. Therefore, it can be concluded that the system is validated for its intended purpose of being a robust, dependable and user-friendly control system for cardiac imaging experiments.

8.4. Test Coverage Discussion

The strategy taken for verification and validation of the LED control system provided comprehensive testing, where each technical requirement outlined in Appendix B of this report was addressed and tested. Each of the nine distinct tests were defined with acceptability criteria that were derived from the technical requirements. A step-by-step procedure was included in each test to ensure that these system tests are transparent and repeatable. The individual tests in sections 8.2.1-8.2.9 of this report verified the core performance of the overall system, including usability, timing and power requirements, where section 8.3 of this report demonstrates that the systems defined technical requirements are observable in a simulated environment. The results above highlight strengths of the overall system, where timing and frequency accuracy are met (TR-1, TR-2), as well as low latency system starts in both 'Manual' and 'Trigger' modes (TR-5 & TR-6) and a significant margin of safety in current/ power consumption under the systems 'worst' case conditions (TR-7).

The design of the system was proven to be successful through the detailed tests, although these tests show some potential areas of improvement. For example, the 2-minute continuous operation test that proved that the system can actively function for a prolonged period could be extended to a 12-hour test. This would be beneficial to test long-term thermal stability of the system, where the temperature of the active components (microcontroller, LED driver circuit) would be monitored to ensure that the system does not ever exceed thermal limits. This idea also introduces a potential design improvement, where a thermal sensor could be integrated with the LED driver circuit system that would automatically shut down the system if high temperatures are met. This improvement would ensure the long-term operation of the system by avoiding potential thermal damage.

9. Discussion

This section gives an analysis of the outcomes of the overall LED control system, where the delivered system was compared against the initial objectives, deliverables and technical requirements of the system. This section also addresses the shortcomings and limitations that are observed in the current LED control system.

9.1. Comparison with Project Objectives

This project successfully met all objectives outlined in section 2.1 of this report, where this system is a significant enhancement over the previous system that was setup in the Quinn Laboratory. The core objectives of creating a user-friendly, robust and precisely timed LED control system were directly met throughout the key deliverables of the project. The deliver of the functional hardware circuit which uses MOSFETs on a perforated board satisfied the objectives outlined in section 2.1 of this report relating to improved system efficiency and reliability. The custom graphical user interface (GUI) application

successfully addressed all user requirements and goals by providing a user-friendly interface for system configuration. The delivered GUI met the goals set after the first semester of development by adding configurable parameters for custom LED illumination patterns, duty cycle and brightness control.

9.2. Design Pros and Cons

This section of the report gives an overview of system's design advantages and limitations.

9.2.1. Pros

The primary strength of the final design is in its ability to be both user-friendly while having a complex and robust electrical system. The custom GUI abstracts away the complex underlying hardware and timing protocols, allowing researchers to control the overall system in a simple and intuitive way. This allows users with minimal training to configure a wide range of experimental configurations, from different system triggering modes to complex and custom LED illumination patterns (addressing core project objectives). The use of MOSDETs for LED switching on the hardware layer of the design also is a significant advantage, as the use of MOSFETs ensures high-efficiency and reliable high-frequency operation. The hardware design also ensures that power consumption limits are never met, which ensures the long-term hardware component health is achieved, prolonging the life of the system.

9.2.2. Cons

Although the delivered system was successful in meeting all objectives and technical requirements, the design has limitations that present opportunities for future improvement. For example, the perforated board used for the delivered system is robust for a prototype or proof of concept, a printed circuit board (PCB) would offer much more durability and a more compact hardware form of the delivered system. Using a PCB would also allow for a simplified assembly of future reproductions of the delivered system. It should be noted that generally, this trade-off was acceptable for the scope of the project but is important to consider for the evolution of the control system in the long term.

9.3. Comparison with Technical Requirements

This section evaluates the delivered systems performance against the initial project objectives and technical requirements. This results from the comprehensive testing described in section 8.2 above confirm that the system meets every desired criterion. Table 10 below provides a mapping of each technical requirement to its validation test, which also summarizes evidence that confirms it has met the requirements:

Technical Requirement ID	Test	Status	Justification & Reference to Test Results
TR-1	Configurable LED flash frequency	Met	Frequency deviation was well within the ±5% tolerance. See Section 8.2.1, Table 7 and waveforms in Figure J.1 (Appendix J).
TR-2	Configurable experiment duration	Met	Measured experiment time had 0% deviation from the user-defined value, meeting the ±2% requirement. See Section 8.2.2 and Figure J.2 (Appendix J).
TR-3	Variable Illumination Patterns	Met	Oscilloscope traces confirmed all tested patterns executed with the correct logic. See Section 8.2.3 and Figure J.3 (Appendix J).
TR-4	Brightness Control	Met	The measured 19.99 kHz PWM carrier frequency was within the ±5 Hz tolerance. See Section 8.2.4 and Figure J.4 (Appendix J).
TR-5	External Trigger Synchronization	Met	Measured trigger latency of 1.94 ms was must less than the 5 ms maximum. See Section 8.2.5 and Figure J.5 (Appendix J).
TR-6	Low Latency GUI Start in Manual Mode	Met	All 5 trials showed a start time under the 2-second limit, as detailed in Table 8 of Section 8.2.6.
TR-7	Safe Current Operating Limits	Met	Max current draw of 208.09 mA is safely below the 1000 mA LED rating. See Figure 8 in Section 8.2.7.
TR-8	Continous Operation	Met	The system operated without failure for the full 2-minute test duration, as documented in Section 8.2.8.
TR-9	Usability for New Users	Met	The 4/5 user success rate met resulted in a system pass, with results summarized in Table 9 of Section 8.2.9.

Table 10: Technical Requirement-Test Validation Map

It is seen from Table 10 above that every technical requirement was successfully validated, where proof that the systems design being successful are observed. The results from these highly granular tests align with the success of the simulated experiment discussed in section 8.3 above, further demonstrating that the system is ready for its intended use of an LED control system for cardiac imaging experiments.

9.4. Limitations and Shortcomings

At the time of this report, the circuit has not yet been mounted within the chassis. Prior to the presentation and demonstration, banana plug connectors will be installed through the chassis wall to accommodate the four LEDs, dual power supplies, and camera input, with all connections secured to the circuit by soldering.

It should also be noted that at the time of this report, the chassis carrying the hardware has future planned improvements, where the body of the chassis will include drilled banana plug holes for easy voltage connection of the LED driving circuit (see Appendix E of this report for a visualization photo of these future chassis changes).

It can be seen also from Appendix E of this report that there are plans to drill holes in the chassis of the device to allow banana-plug connections to power the two LED driver circuits in the device. Drilling these holes will be completed before the final product is delivered to the lab for sustained use.

It is also noted that two key supporting documents outlined in section 2.2 of this report are still in-progress at the time of this reports' submission. The first document is the systems the *Testing Overview,*

which defines a detailed report that summarizes the timing synchronization characteristics between the LED control system and the lab's camera. The second document that is in development is the systems *Lab Manual*, which serves as the complete documentation package for the lab personnel, which contains circuit schematics, a detailed user guide, a troubleshooting manual and maintenance instructions to ensure the systems long-term usability.

10. Conclusions

The LED control system developed for the Quinn Laboratory was successfully designed, implemented, validated and delivered to the lab for cardiac imaging experiments. This delivered system met all technical requirements, objectives and major deliverables while providing a cost-effective, user-friendly system. These system features were proven through a series of rigorous tests, which proved that the system is fit to control precisely timed LED illumination in cardiac imaging experiments. The final delivered product proved to be a significant improvement to the previous system used in the laboratory. The custom-designed graphical user interface abstracts away the complexity of the hardware and firmware implementations, which ensures that researchers can intuitively configure and execute experiments with high-frequency timing accuracy and ensures they have control over newly integrated key system parameters such as custom flashing patterns, LED brightness and pulse duty cycle. The user has complete control over the entire system through the easy-to-use user interface (developed using the PyQt6 framework). The GUI software is packetized into a binary executable file (.exe) that is ran by the user to start the system, further proving the robustness of the system. The system architecture allows researchers to execute experiments intuitively (TR-9) with hardware-timer level frequency accuracy for a LED flash period, pattern and brightness of their choosing (TR-1, TR-2, TR-3, TR-4, TR-8), while ensuring a low latency system was achieved (TR-5, TR-6) where power consumption limits of the hardware were not exceeded (TR-7). The status of the completed deliverables (outlined in section 2.2 of this report) with percentages of completion are shown in Table 11 below:

Deliverable	Status	% Complete
Functional Hardware Driver Circuit	Completed	100%
Software Design	Completed	100%
Hardware Implementation:	Completed	100%
LED/ Camera Timing Synchronization Report	In Progress	60%
Final Documentation (User Manual)	In Progress	80%

Table 11: Project Deliverables Status

Ultimately, the successful validation of all nine key technical requirements (outlined in section 8.2 of this report) and deliverables (outlined in section 2.2 of this report) proved that the system is fully

prepared for its intended application, proving that the solution is a reliable and powerful tool for cardiac imaging experiments at the Quinn Laboratory at Dalhousie University.

11. Future Recommendations

Based on the usability test results (test #9 for TR-9 in section 8.2.9 of this report above), it was noted that one of the users used the incorrect COM port (see Table 10 above). A design improvement for this technical requirement could be adding functionality to the GUI that saves experiment configurations to a file, which can enhance the workflow efficiency of the researchers and ensure that experiments are exactly repeatable.

Another addition that could be introduced to the system is the concept of reuseable and savable experiment configuration files. This feature would allow users to save and load entire experiment configurations from the GUI, which would enhance the overall efficiency of the user's workflow and ensure that experiments are exactly repeatable. The ability to restore complex timing and frequency settings from a file would reduce setup time of experiments and eliminate the potential risk of manual entry errors.

The group also made a recommendation to transition from a perforated board to a printed circuit board (PCB), which would be more suitable for long term use of the system. The current implementation of the circuit on a perforated board is suitable for development and system validation, but a custom-manufactured PCB would improve the overall durability and electrical reliability/performance of the system while creating a more compact and professional system.

Another suggestion would be to provide an enhanced thermal management and monitoring subsystem that integrates a temperature sensor that triggers an automatic system shutdown if thermal limits are exceeded. While the system did successfully pass the 2-minute continuous operation test (as described in section 8.2.8 above that tests the technical requirement TR-8), longer experiments (hours or days) could lead to heat losses being observed in the LED driver circuits. Integrating a temperature sensor that provides real-time feedback to the GUI, where an automatic shutdown circuit would prevent hardware damage and ensure long term system durability.

References

Adafruit Industries LLC. (n.d.). *Prototyping, fabrication products | DigiKey*. DigiKey. https://www.digikey.com/en/supplier-centers/adafruit-industries-llc

Atmel. (2015). *ATmega328P 8-bit AVR Microcontroller Datasheet*. Atmel Corporation. https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

Baines, O., Sha, R., Kalla, M., Holmes, A. P., Efimov, I. R., Pavlovic, D., & O'Shea, C. (2024). *Optical mapping and optogenetics in cardiac electrophysiology research and therapy: A state-of-the-art review*. Europace, 26(2), euae017. https://doi.org/10.1093/europace/euae017

DigiKey Electronics. (n.d.-a). *[Product listing]*. DigiKey.

DigiKey Electronics. (n.d.-b). *HSE-B20350-NP*. DigiKey.

DigiKey Electronics. (n.d.-c). *IRLZ44NPBF*. DigiKey.

DigiKey Electronics. (n.d.-d). *LZ4-40CW08-0055*. DigiKey. https://www.digikey.com/en/products/detail/ams-osram-usa-inc/LZ4-40CW08-0055/4970098?msockid=127fa9a9abc96b9f1f55bfc8aa796a5e

Facmogu. (n.d.). *Facmogu 3V-12V 5A 60W adjustable power supply with LED display, variable DC 3V 5V 6V 9V 12V universal AC/DC adapter, 100V-240V AC to DC 3-12V switching power converter multi-voltage regulated adaptor: Electronics*. Amazon. https://www.amazon.com/dp/B08H8QKX5C

Herron, T. J., Lee, P., & Jalife, J. (2012). *Optical imaging of voltage and calcium in cardiac cells & t issues. Circulation Research.* American Heart Association. https://www.ahajournals.org/doi/10.1161/CIRCRESAHA.111.247494

Lee, P., Bollensdorff, C., Quinn, T. A., Wuskell, J. P., Loew, L. M., & Kohl, P. (2011). *Single-sensor system for spatially resolved, continuous, and multiparametric optical mapping of cardiac tissue*. Heart Rhythm, *8*(9), 1482–1491. https://doi.org/10.1016/j.hrthm.2011.03.061

Texas Instruments. (2016). *Dimming Techniques for Switched-Mode LED Drivers*. Texas Instruments. https://www.ti.com/lit/an/snva605/snva605.pdf

The Qt Company. (2025). *Qt 6.9 documentation*. https://doc.qt.io/qt-6/

Oxford Instruments. (2012). *iXon3 hardware guide*. Andor Technology Ltd. https://andor.oxinst.com/downloads/uploads/iXon3_Hardware_Guide.pdf

Unknown. (n.d.). *Nano v3.0 3.0 ATMEGA168 CH340G CH340 mini USB UART interface board micro controller module for Arduino 3.3V 5V microcontroller*. Amazon.ca: Electronics. https://www.amazon.ca/dp/B07TSVRGFW

Appendix A: Authors by Report Section

Authors by Report Section				
Section	**Sub-section**	**2nd Sub-Section**	**3nd Sub-Section**	**Author**
1. Introduction	1.1 Project Background	-		Matthew Watson
	1.1 Project Rationale			Matthew Watson
	1.3 Investigation of Literature			Ha Vu
2. Objectives and Deliverables	2.1 Objectives	-		Devon Harvey
	2.2 Deliverables			Ha Vu
3. Technical Requirements	-			Matthew Watson
4. Methods	4.1 Possible Approaches	4.1.1 Hardware Approaches	N/A	Devon Harvey & Ha Vu
		4.1.2 Software Approaches	4.1.2.1 Brighness Control Approaches	Matthew Watson
			4.1.2.2 User Interface Approaches	Matthew Watson
			4.1.2.3 Triggering and Synchronization Mode Approaches	Matthew Watson
	4.2 Technical Background	4.2.1 Timing and Frequency Control	-	Matthew Watson
		4.2.2 Pulse Width Modulation (PWM) and Duty Cycle		Matthew Watson
		4.2.3 Hardware Driving & Switching of LEDs		Devon Harvey & Ha Vu
		4.2.4 Graphical User Interface and Serial Communication		Matthew Watson
5. Final Gannt Chart and Workplan	5.1 Final Gantt Chart	-		Devon Harvey
	5.2 Workplan Deviations from Technical Memo			Devon Harvey
	5.3 Final Workplan Deliverables and Completion Summary			Ha Vu
6. Budget and Task Distirbution	6.1 Budget Summary	-		Ha Vu
	6.2 Cost Savings and Constraints			Ha Vu
	6.3 Milestones, Task Distribution and Leaders			Ha Vu
7. Proposed Solution	7.1 System Overview	-		Matthew Watson
	7.2 System Architecture			Matthew Watson
	7.3 Hardware Architecture Overview			Devon Harvey & Ha Vu
	7.4 Software Architecture Overview	7.4.1 Overall Software Architecture	-	Matthew Watson
		7.4.2 Graphical User Interface Overview		Matthew Watson
		7.4.3 GUI Software Overview		Matthew Watson
		7.4.4 Firmware Overview		Matthew Watson
8. Results and Verification	8.1 Verification Against Technical Requirements	-		Devon Harvey & Ha Vu
	8.2 Test Procedures, Measurements and Results	8.2.1 Frequency Accuracy Test (TR-1)	-	Matthew Watson, Devon Harvey & Ha Vu
		8.2.2 Timing Accuracy Test (TR-2)		Matthew Watson, Devon Harvey & Ha Vu
		8.2.3 Illumination Pattern Test (TR-3)		Matthew Watson, Devon Harvey & Ha Vu
		8.2.4 PWM Brightness Control Test (TR-4)		Matthew Watson, Devon Harvey & Ha Vu
		8.2.5 External Trigger Synchronization Test (TR-5)		Matthew Watson, Devon Harvey & Ha Vu
		8.2.6 Low Latency System Start in Manual Mode Test (TR-6)		Matthew Watson, Devon Harvey & Ha Vu
		8.2.7 Operating Current Limit Test (TR-7)		Matthew Watson, Devon Harvey & Ha Vu
		8.2.8 Continuous System Operation Test Procedure (TR-8)		Matthew Watson, Devon Harvey & Ha Vu
		8.2.9 New User Usability Test Procedure (TR-9)		Matthew Watson, Devon Harvey & Ha Vu
	8.3 System Validation & Demonstration	-		Matthew Watson
	8.4 Test Coverage Discussion	-		Matthew Watson & Ha Vu
9. Discussion	9.1 Comparision with Project Objectives	-		Matthew Watson
	9.2 Design Pros and Cons	9.2.1 Pros	-	Ha Vu
		9.2.2 Cons		Devon Harvey
	9.3 Comparision with Technical Requirements	-		Matthew Watson
	9.4 Limitations and Shortcomings			Matthew Watson
10. Conclusions	-			Ha Vu
11. Future Recommendations	-			Devon Harvey

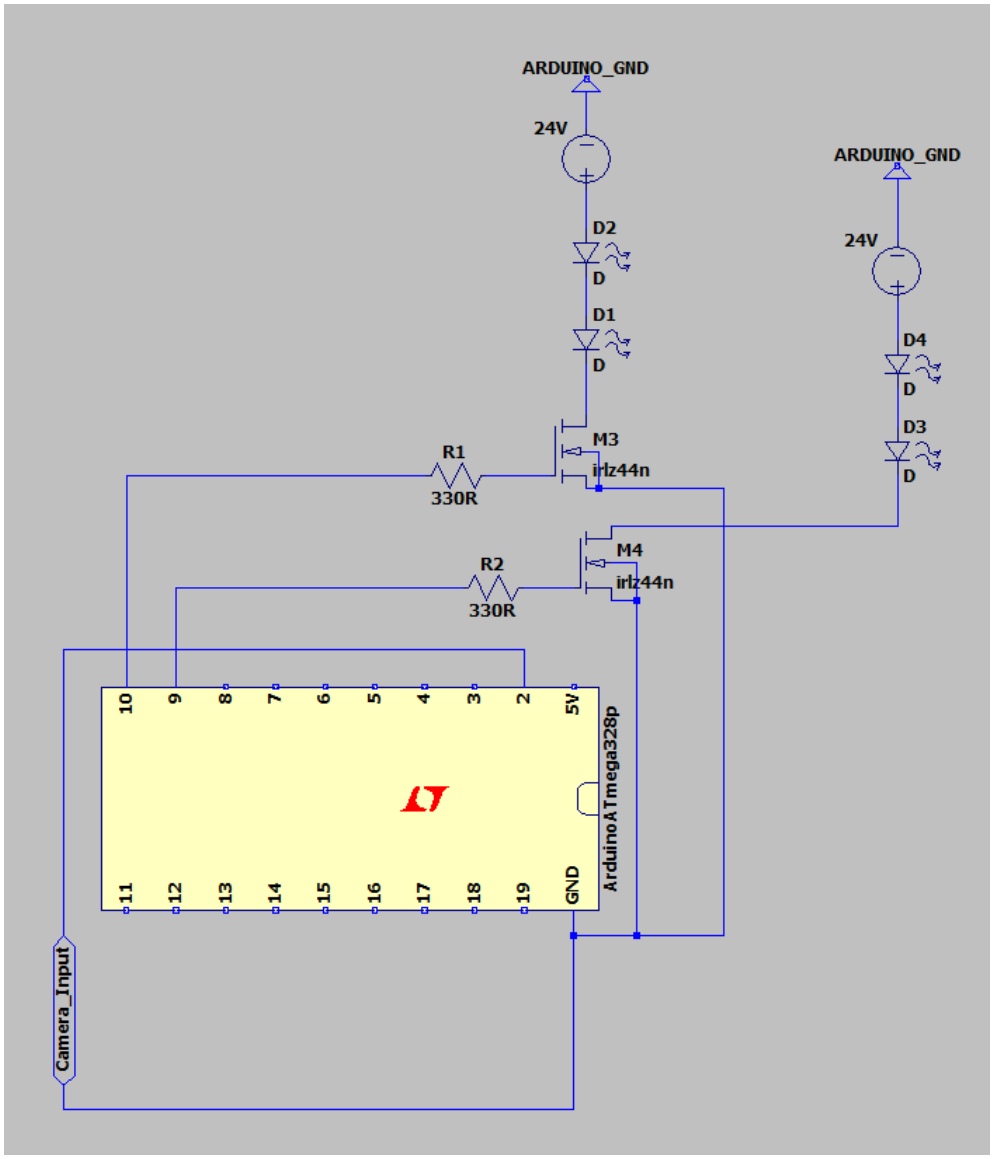
Appendix B: Summary of Technical Requirements

ID	Requirement	Requirement Overview	Verification Method	Reasoning
TR-1	**Configurable LED flash frequency**	The system will allow the user to configure the LED flash frequency between 1 Hz and 1000 Hz in integer steps of 1 Hz. For any frequency, the measured period of the LED pulses shall be within ±5% of the ideal period (T = 1/ f).	Oscilloscope measurement of LED output period from the microcontroller at a set of representative frequencies (1 Hz, 10 Hz, 100 Hz, 1000 Hz, and 2000Hz), and comparison with the expected periods.	This ensures that the system can cover typical frequencies used in fluorescence-based cardiac imaging while maintaining timing accuracy for reproducible experiments.
TR-2	**Configurable experiment duration**	The system will allow the user to configure the experiment duration between 0.1s and 60s in steps of 0.1s. The duration of the pulsing should be within ±2% of the configured value.	Analyze the timing of different pulse trains using an oscilloscope and compare this time with the requested duration in the GUI.	This ensures that LED illumination time is controlled, which ensures that the excitation of the dyes in the cardiac tissue occur only during specified times.
TR-3	**Variable Illumination Patterns**	The control system will support four illumination patterns that are selectable by the GUI, where for each selected pattern, the observed LED behaviour should match the pattern definition over 10 consecutive cycles.	Visual inspection and oscilloscope probing for the two LED banks for each pattern, confirming the correct sequence and timing.	Allows researchers to use different experimental protocols and multi-LED configurations without manually changing the hardware of the system.
TR-4	**Brightness Control**	The system will provide user-configurable brightness control implemented through PWM. The PWM carrier frequency used for brightness control shall be equal to 20 kHz to avoid perceptible flicker and to ensure that the LED intensity appears constant within a typical camera exposure window.	Measure the PWM carrier frequency with an oscilloscope and confirm it is 20 kHz.	A high PWM frequency prevents unintended flickers in the camera and reduces interaction with camera frame timing.

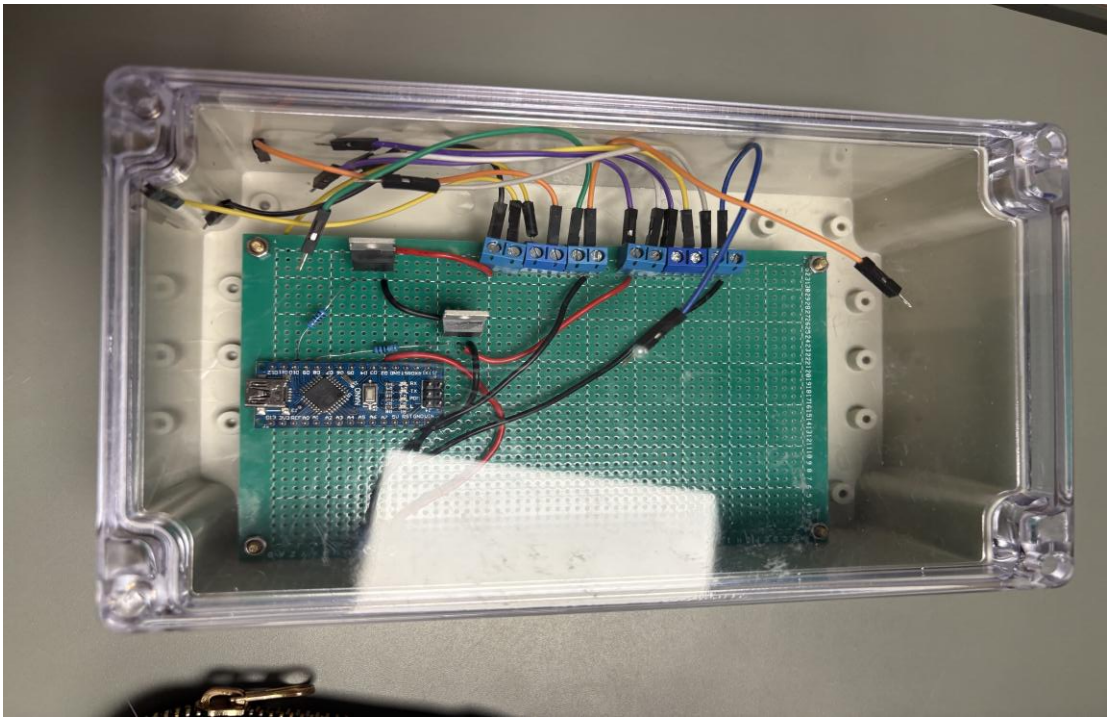
TR-5	**External Trigger Synchronization**	When the LED control system is in trigger mode, the delay between a rising edge on the trigger input and the start of the corresponding LED pulse will be less than 200 µs.	An oscilloscope is used to monitor the external trigger signal and LED output, where the delay is measured across multiple triggers.	Ensures that LED illumination can be aligned with the start of camera exposures for synchronized imaging.
TR-6	**Low Latency GUI Start in Manual Mode**	After the user presses "Start" in the GUI when the system is in manual mode, the system shall begin generating pulses with the new configuration in less than 2 s. The configuration summary displayed in the GUI will also match the parameters sent to the microcontroller.	Measure the time from GUI command to first observed pulse on the oscilloscope and compare GUI configuration text with the serialized configuration packet captured on the serial line.	Limits camera wait time and ensures alignment between user-selected settings and system behaviour.
TR-7	**Safe Current Operating Limits**	The LED driver circuit will limit the average LED current to a value within the manufacturer's specified current limit under all operating configurations (frequency, duty cycle, and pattern).	Measure high-power LED current using a current probe for worst-case settings (maximum duty cycle and highest frequency) and verify it remains below the datasheet limit.	Protects the LEDs from overcurrent damage and ensures long-term use of the system.
TR-8	**Continuous Operation**	The system will operate continuously at a "worst-case" configuration (maximum frequency and high duty cycle) for 2 minutes without unexpected resets, communication failures, or abnormal heating of major hardware components.	Run a 2-minute test while monitoring system temperature, GUI responsiveness, and LED output, recording any failures.	Demonstrates that the system is robust enough for typical experiment sessions in the laboratory.

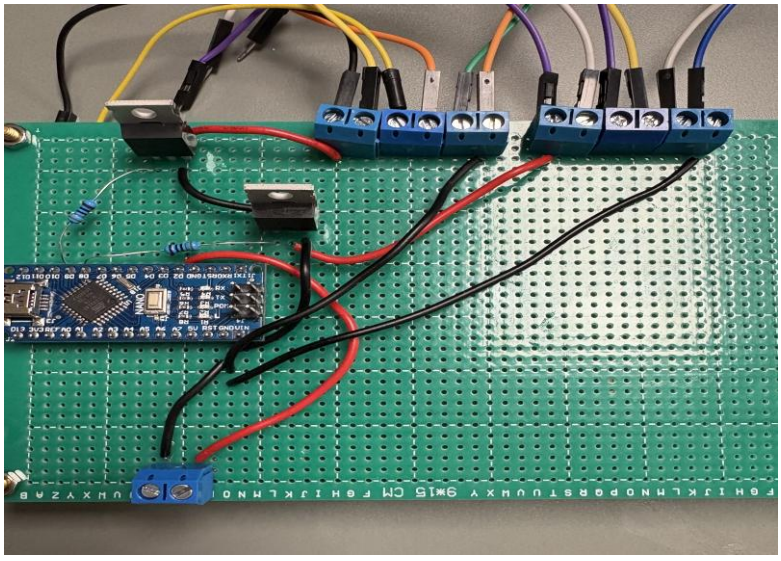
TR-9	**Usability for New Users**	A new user with no prior exposure to the system shall be able to configure a basic experiment (select COM port, set frequency, duration, pattern, and brightness, and start a run) in less than 5 minutes. At least 80% of test users shall complete the task successfully	Conduct a small user study with 5 volunteer users (students), time the completion of the task, and record success rate and any usability issues.	Ensures that the system is practical for use by students and researchers who are not involved in its development.

Appendix C: Circuit Schematic Diagram

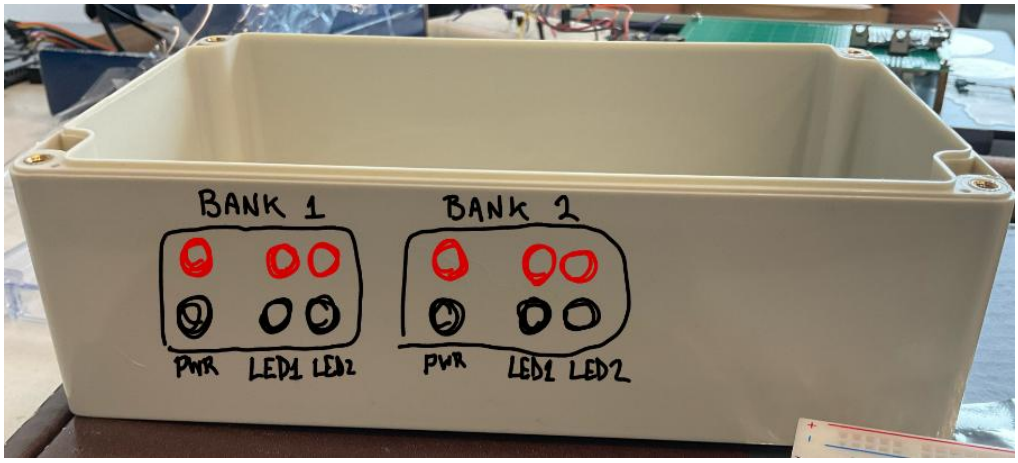


Appendix D: Final Hardware Reference Photos





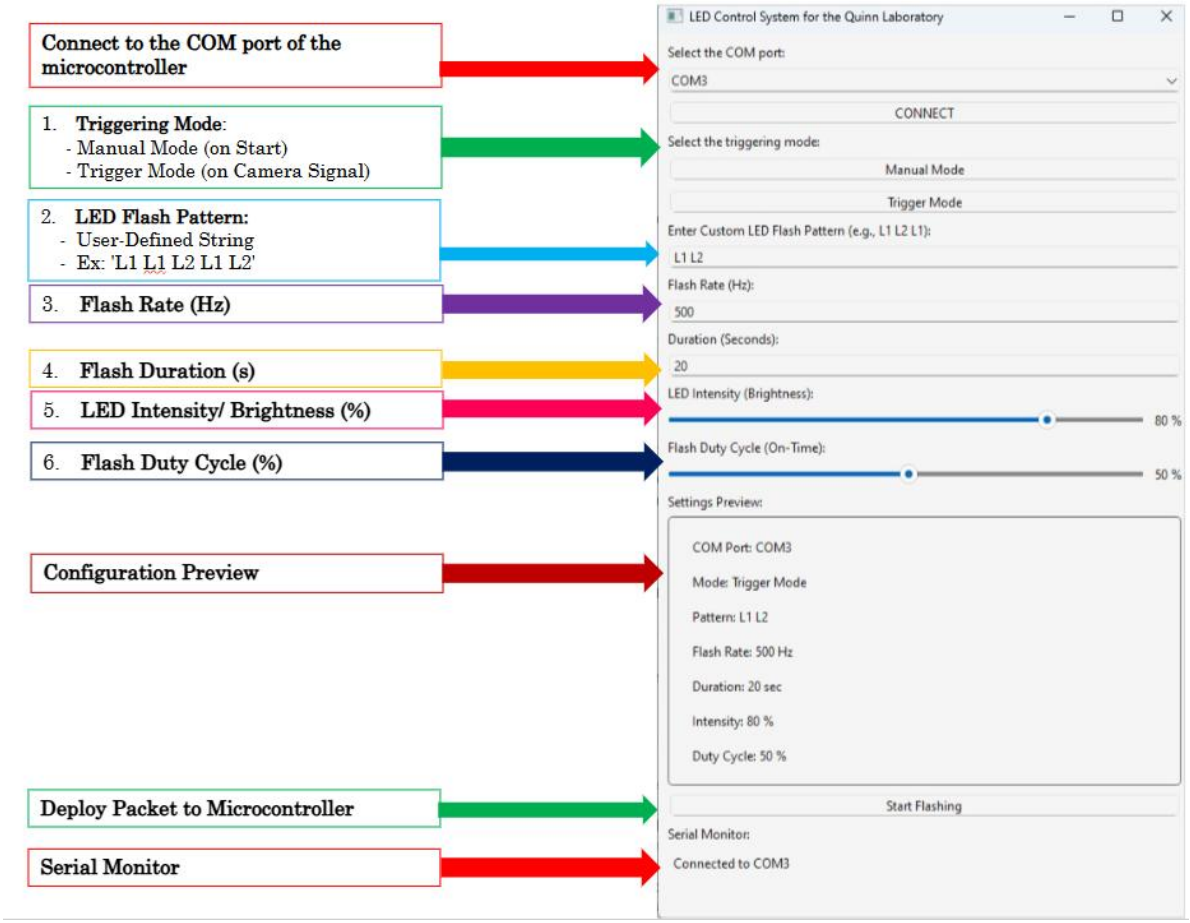
Appendix E: Future LED Bank Chassis Holes



Appendix F: Configuration Packet Parameters and Descriptions

CONFIGURATION PACKET PARAMETERS					
	Relevant Parameter Information				
Parameter	**Description**	**GUI User Input**	**GUI Input Method**	**Command Packet**	**Microcontroller Interpretation**
1) Header	Fixed header that indicates the start of a new packet to the microcontroller	N/A	N/A	Both 'SET' and 'PATTERN' commands	ASCII string "SET" or "PATTERN" followed by a space character ' ' and relavent data, and terminated by a newline character '\n'
2) Mode	Triggerring Mode of the camera (method of which an experiment is initiated)	2 Options: Option 1: Manual mode: Experiment begins by user indication Option 2: Trigger mode: Experiment begins by camera indication	Push Button	'SET' command	First integer seen after space character following 'SET' string, 2 options: Option 1: Integer '1' is manual mode Option 2: Integer '2' is trigger mode
3) Flash Pattern	Pattern at which LED banks alternate flashing	Custom user input: User-defined string of 'L1' or 'L2' seperated by spaces, where 'L1' indicates LED bank #1 and 'L2' indicated LED bank #2.	Text Box	'PATTERN' command	First two charachters after ASCII string "PATTERN " (either 'L1' or 'L2'), where each concurrent LED bank is space-seperated until a newline character termination in conifguration packet ('\n')
4) Flash Rate	Frequency at which the LEDs in the system are flashing	Integer [Hz]	Text Box	'SET' command	Integer following space character after first integer in 'SET' command
5) Flash Duration	Length of time that the experiment runs for	Integer [s]	Text Box	'SET' command	Integer following space character after second integer in 'SET' command
6) LED Brightness	Brightness percentage controlling the PWM duty cycle.	Integer [%]	Slider	'SET' command	Integer following space character after third integer in 'SET' command
7) Flash Duty Cycle	Percentage controlling the ON time of an LED during one flash cycle	Integer [%]	Slider	'SET' command	Integer following space character after fourth integer in 'SET' command before newline termination of command string ('\n')

Appendix G: Final Graphical User Interface with Component Descriptions



Appendix H: GUI Software Architecture with Classes Hidden

```
#****************CLASSES/ SETUP****************
import sys
import time
import serial
import serial.tools.list_ports
import threading

from PyQt6.QtWidgets import (
    QApplication, QMainWindow, QWidget, QVBoxLayout, QHBoxLayout,
    QLabel, QComboBox, QPushButton, QLineEdit, QTextEdit, QFrame, QSlider
)
from PyQt6.QtCore import pyqtSignal, QObject, Qt

#globals
arduino = None
selected_mode = "Manual Mode"
selected_pattern = "L1"
selected_flash_rate = 1
selected_flash_duration = 1
selected_intensity = 100
selected_duty_cycle = 50
selected_COMport = "None"

#*****************CLASS DEFINITIONS***************
class serialclass(QObject):···

class systemGUI(QMainWindow):···

#***************MAIN***************
if __name__ == "__main__":
    pyQtapp = QApplication(sys.argv)
    mainwindow = systemGUI()
    mainwindow.show()
    sys.exit(pyQtapp.exec())
```

Appendix I: Timer1 Overflow Interrupt Service Routine Firmware Excerpt

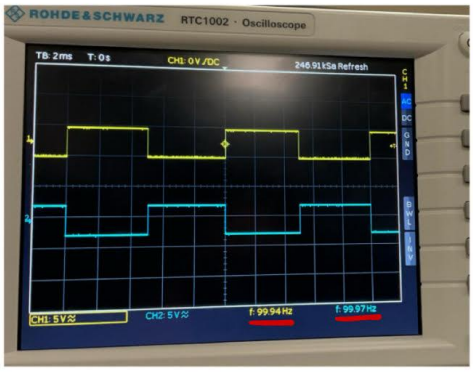
```
92  // ISR that is triggered when timer1 overflow occurs
93  // this occurs at a rate of 20 kHz
94  ISR(TIMER1_OVF_vect) {
95
96    //if the flashing sequence is not active, exit this ISR
97    if (!is_flashing) return;
98
99    // increment the couter that tracks interrupts since last step change
100   isr_step_counter++;
101
102   // have enough interrupts passed to advance the LED pattern to the next stage?
103   if (isr_step_counter >= interrupts_per_pattern_step) {
104
105     //reset counter for next step in sequence & incrememnt the sequence progress
106     isr_step_counter = 0;
107     sequence_step_counter++;
108
109     // check if flash sequence is over
110     if (sequence_step_counter >= total_steps_for_sequence) {
111       //stop all timers and turn off LEDs
112       stop_all_activity();
113       if (mode == 2) {
114         //arm the trigger again if in trigger mode
115         trigger_armed = true;
116       }
117       return;
118     }
119     // go to next pattern (wrap around)
120     pattern_step = (pattern_step + 1) % custom_pattern_length;
121   }
122
123 //ON portion of current LED step duty cycle
124   if (isr_step_counter < interrupts_for_on_time) {
125     // read the charachter corresponding to the current LED from the current LED pattern array
126     char current_led_char = custom_pattern_sequence[pattern_step];
127     //current LED is LED1
128     if (current_led_char == '1') {
129       // connect PWM to LED1 pin
130       TCCR1A |= (1 << COM1A1);
131       // disconnect PWM from LED2 pin
132       TCCR1A &= ~(1 << COM1B1);
133     // LED is LED2
134     } else if (current_led_char == '2') {
135       // connect PWM to LED1 pin
136       TCCR1A |= (1 << COM1B1);
137       // disconnect PWM from LED2 pin
138       TCCR1A &= ~(1 << COM1A1);
139     }
140   } else {
141     // disconnect PWM from both LED's if in 'OFF' part of overall duty cycle
142     TCCR1A &= ~((1 << COM1A1) | (1 << COM1B1));
143   }
144 }
145
```

Appendix J: Test Results by Requirement

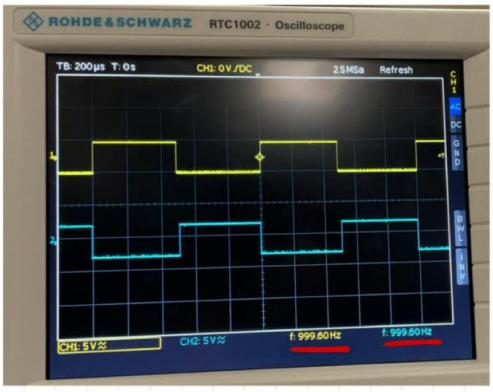
Figure J.1: Test Results for Test 1 (TR-1)

f = 10 Hz:

f = 100 Hz:



f = 1000 Hz:



f = 2000 Hz:

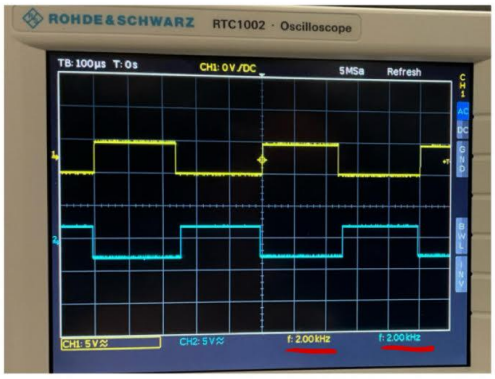
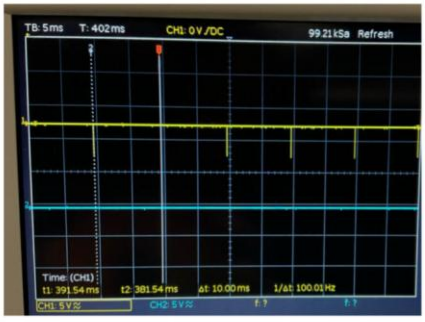


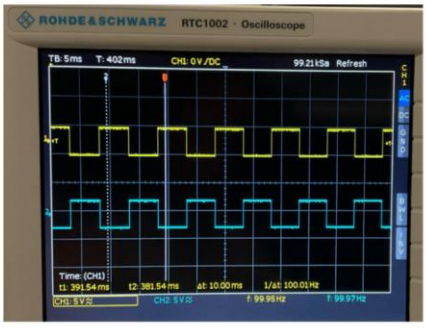
Figure J.2: Test Results for Test 2 (TR-2):

Figure J.3: Test Results for Test 3 (TR-3):

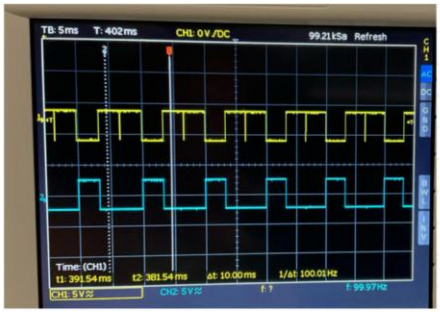
Pattern: 'L1'



Pattern: 'L1 L2'



Pattern: 'L1 L1 L2'



Pattern: 'L1 L1 L1 L2'

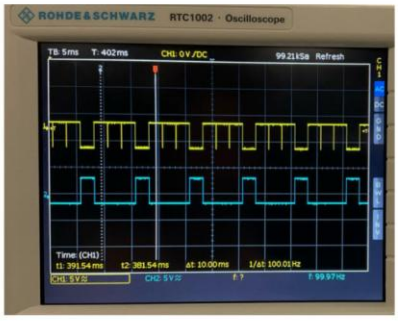


Figure J.4: Test Results for Test 4 (TR-4):

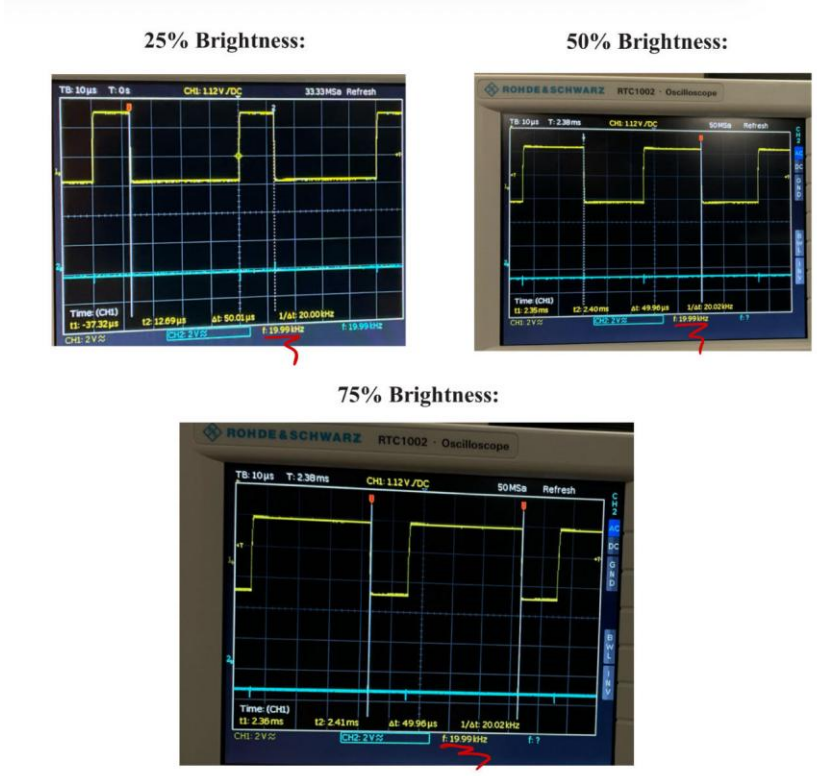


Figure J.5: Test Results for Test 4 (TR-5):