Multi-well LED Irradiation System (MLIS)

Michael Au-Duong & Jonathan Lentini

Spring 2020

1 Abstract

The Multi-well LED Irradiation System (MLIS) is an array of 24 ultra-bright LEDs that can be controlled individually in terms of timing. The array is intended to be used directly under a typical 24-well plate from *ibidi*, which is used for cell microscopy in photodynamic therapy (PDT), a type of treatment that uses light, along with various photosensitizing agents, to kill cancer cells. Compared to mainstream marketed PDT systems' cost of $1500+, this project is very low-cost, at a total of less than $500. The array's full circuit consists of the LED, a basic NPN transistor, power supply, various resistors, and Arduino MEGA 2560 I/O pins; which are used to control the LEDs individually via a computer application graphical user interface (GUI). The GUI allows full manual control of the timing which will end with an automated shut off when the timer ends, along with both an audio and visual pop up to signify the end of the experiment. The system maintains a low temperature output, which is crucial in working with cell-based experiments. The case is designed to be fitted with six fans, with four circulating the air within the case and two fans venting out the air. The LEDs used in the system have an output wavelength of 630 nm. Theoretically, the LEDs will output up to 88.9 mw/cm^2 to each individual well on the *ibidi* plate. Throughout the system, the current output is relatively consistent when a different number of LEDs are turned off or on.

2 Introduction

This project aims to deliver a programmable LED array to be used in cancer research laboratories. The device would be used to assist in studying the effects of exposing cancer cell cultures to light, in hopes of enabling the researcher to conduct their research more efficiently. The array is geared toward research revolving around photodynamic therapy (PDT) as a treatment on cancer cells, specifically regarding the response of 3D cancer cell cultures; the ability to make this treatment available in resource-limited areas is also a large focus of our client's [1] . While PDT research has been shown to be applicable through clinical trials, 3D cancer cell cultures have been shown to be much more resilient

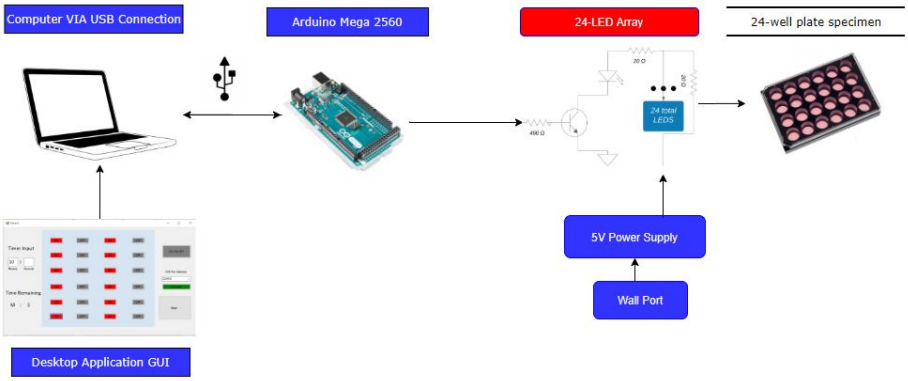

Figure 1: System block diagram for the Multi-well LED Irradiation System.

to the treatment than a two dimensional monolayer of cells [2]. Because of this, along with the global impact of the research, any increase to the speed and efficiency of PDT development would be incredibly valuable. Without our device, researchers currently use a single LED to treat the cells on the multi-well plate and must manually shift the plate whenever they wish to treat other sections. Not only does this method take away valuable time from the researchers, but it also affects the cells they are treating as the cultures are maintained at 37 C, a temperature that is not attainable during the treatment [3].

Based on this, the purpose of the Multi-well LED Irradiation System is to both increase the speed at which the cell-containing plate's treatment is conducted as well as allow the experimenters a greater degree of freedom while the experiment is ongoing. The array is easily programmable by a lab assistant or researcher and provides individual control over each LED's exposure time and intensity.

3 Background

Research is advancing in the field of photodynamic therapy as a treatment of cancer cells. An important portion of this process is exposing the cells to light for varying amounts of time. Although this step is as important as any other in determining the outcome of a PDT experiment, it requires experimenters to manually control how much light exposure each well of cancer cells is receiving and closely monitor the plate of wells. Not only is this an inefficient use of time on the part of the experimenter, but the cells must also remain incubated or they become less useful as experimental subjects.

The goal of this project is to create a device that would automatically expose the wells to a customizable level of light intensity and duration of exposure. The main requirements would be as follows:

- User-friendly interface to control the output of the light

- A notification of when the treatment is complete
- Must maintain a low heat output
- Fit a generic 24-well plate from *ibidi*
- A simple on/off switch for all 24 LEDs (with software applications)
- Easily outsourced
- Inexpensive

3.0.1 Existing Works

Although there seems to be a large amount of information on the use of high-powered LED arrays in clinical use to treat cancer cells, we are unable to use these systems in the lab due to the inability to isolate specific LEDs and specific cells. One lab group already designed a similar system for ultraviolet exposure, and their work may be adapted for the needs of a researcher's lab [4]. Most other existing methods and devices either expose the entire plate to the same dose of PDT or lack individual control over each well's exposure [5]-[6]. There is one commercial product which seems to fit the problem very well. However, this product's pricing is not publicly available and may be prohibitively expensive [7]. Additionally, it appears that this product gives individual control over regions of the LED array and not each individual LED.

In the realm of commercial LED products, there are quite a large number of LED options including LEDs emitting specific wavelengths of light across most of the visible spectrum. For example, the LED Array ceLED 96, as shown in Figure 2, can be customized to feature a set of specific wavelength LEDs.The broad array of existing LED products should allow for easy customizability of the array for any given research need. For example, the LED Array ceLED 96, as shown in Figure 2, can be customized to feature a set of specific wavelength LEDs. But, as with other LED array products designed for PDT, the ability to control the LEDs individually does not exist. By building both the array and control solution from scratch, the specific needs of experiments in the researcher's lab can be best fulfilled.

There is a substantial amount of research that has gone into thermal management of LEDs which can be drawn upon during the design process [8][9]. Depending on the final design of this well exposure array, some heat management solution may be required. Solutions may include placing the entire device within a refrigerated space, using heat fins, fans, or even Peltier cooling devices for drawing heat away from the LEDs and maintaining ideal experimental conditions for the exposed cells.

3.0.2 Global and Societal Impact

By creating this device for implementation in a PDT researcher's lab, we are not only allowing each member of the research team to work in a more efficient

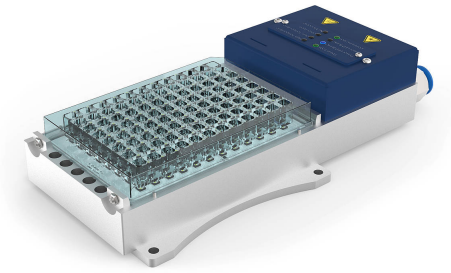

Figure 2: LED Array ceLED 96 by Cetoni.

manner, but we are also eliminating some of the concerns that may come with performing these experiments (e.g., human error and temperature of the cells). Although this device will not completely eliminate all potential for error, it allows the research team to focus on more complex matters. PDT research deals with a widespread and fatal illness, cancer, and any amount of time that is diverted from menial tasks towards solving more pressing issues within the scope of the research team can lead to an incredible social and global impact, possibly saving lives in the process.

More specifically, over the past few decades, PDT has been used for cancers in different parts of the body. Because using PDT has minimal impact on the skin and does not affect the entire body all at once, it has been advanced more and more by cancer researchers to adapt it the different fields of oncology research. PDT has also proven to be less invasive than surgeries and takes less time per session [10] .

Also, the specific use of LEDs in photodynamic therapy has been decided on due to a significant amount of other research which showed substantial results in treatments by infrared light. With LEDs, treatments can be cheaper and some dermatology risks could be avoided [11] . Consequently, it is clear that fabricating a more efficient and automated system to test LED exposure and frequencies on wells of cancer cells could prove to be extremely helpful in further cancer research.

When examining the physical components of our device, it is clear that most of the components are fairly environmentally friendly. Our casing of the device is a 3D printed model, and the filament which is used to create the shell is composed of thermoplastics. Thermoplastics are fairly environmentally friendly for multiple reasons, mainly because it can be made from recycled materials (including chicken feathers) [12] and because the plastics can be recycled. 3D printing itself has a positive effect on the environment as it cuts down on distribution emissions due to the absence of shipping materials, although the drawback is that there is an increase to the electricity consumed when creating the product [13].

The Multi-well LED Irradiation System's affect on public welfare cannot be understated as it is helping to increase the efficiency of a PDT researcher's lab

work. The researcher's work with PDT not only advances the treatment and elimination of three dimensional oral cancer tumors as well as treatment of pancreatic cancer, but allows the advancements to be utilized by underfunded and underprivileged areas of the world through his efforts to create these treatments in a way that is both inexpensive and portable.

3.0.3 Economic Factors

In terms of cost, the design will be relatively inexpensive. Much of the costs will be focused on housing and cooling the LEDs within the system. The housing will also include integrating lenses to focus the LEDs' light along with lowering the bleeding effect between the LEDs. The anticipated total cost will be approximately $300 to prototype. One of the requirements of this project would be to make the design outsourceable, which, in combination with its inexpensive cost, will make PDT research much more affordable..

3.1 Relevant Engineering Standards

Standard Number	**Standard Name**	**Description**
IEEE 82079-1-2019	IEEE/IEC International Standard for Preparation of use (instructions for use) of Products	These standards are used to ensure that all aspects of the software created to use the device are communicated in a way that ensures that the device is utilized effectively and efficiently by the client.
IEEE 12207-2008	ISO/IEC/IEEE International Standard - Systems and software engineering – Software Life Cycle Process	This standard is used to regulate the supply, development, acquisition, and maintenance of engineering software.
N/A	NPSE Code of Ethics for Engineers	This standard is used to express that an engineer should pursue their practices in a safe and ethical manner as well as outlining how to do so.

IES LM-80-08	N/A	This standard is used to create a common way for LED lumen maintenance to be measured in LED arrays as well as LED packages and modules.
IES UL8750	N/A	This standard is used to issue safety requirements for LED devices who utilize LEDs that operate between 400-700nm.
Instruction Manual	The manual should provide clear instructions on using the device as well as information on how it was built.	The document will be reviewed by the client and researchers in his lab to ensure it covers all the needs of their experiments.

4 Design

The overall system consists of 24 ultra-bright LEDs, which function within the wavelength requirement of 630 nm, organized in an array suited to an *ibidi* well plate. The system is controlled via an Arduino with a GUI designed to implement individual timers for each LED, which includes an audio and visual pop up when each timer ends. The case is designed to stabilize all the components and fit six fans to circulate and vent out any heat coming from the components. The overall system can be visually depicted as shown in Figure 1.

4.1 Specifications and Requirements

Requirements	Note	Verification
Exposure LED	Can be controlled by an Arduino to provide exposure in a wide range of wavelengths and time.	Spectroscopic tests will be done on different LEDs to determine the best LED to use for different wavelength exposure. This test will also show how long the LED can function.
Physical Case	It should hold the LEDs securely against the sample plate while drawing heat away from the samples if necessary.	LED testing will decide the need for a heat sinking solution. The Physical case design will depend on the chosen LEDs and heat sink. But due to the previously used setup by the client, a glass cover will be implemented, so that the wells can be directly positioned above the LEDs.
LED array	Multiple LEDs integrated into one case but functioning individually based on parameters set through the GUI.	Test will be run on the finished array to verify that exposure light is contained within the target well. A simple photoresistor will be used to measure the light intensity at each well to ensure that the LEDs provide consistent experimental results.
Arduino	An off the shelf Arduino that will provide control signals for the LEDs as instructed by the GUI.	The exposure system functionality will be tested for all likely scenarios.

GUI	A GUI implementation on a client computer will be used for defining LED parameters within the Arduino.	The GUI will be thoroughly tested for all likely experiment scenarios. The GUI will also be approved by the client and/or researchers in his lab.
Instruction Manual	The manual should provide clear instructions on using the device as well as information on how it was built.	The document will be reviewed by the client and researchers in his lab to ensure it covers all the needs of their experiments.

4.2 Proposed Design Goals

4.2.1 Goal 1: Purchasing Samples

Buying multiple test LEDs and arduino as well as any other needed supplies in order to identify the best materials to use for prototyping.

4.2.2 Goal 2: Building Single LED Proof of Concept

Building a single LED exposure system. This should consist of a single LED, the arduino, and a simple interface for setting the exposure time and intensity.

4.2.3 Goal 3: Design and Build Physical Case

Once the specific LED is chosen, a physical structure will be designed to hold the LEDs below the sample wells. The structure should hold as many LEDs as possible while also ensuring that light from each LED does not expose any adjacent sample wells. Ideally the case will allow for easy swapping of LEDs for maintenance or future design changes.

4.2.4 Goal 4: Build Full Exposure System

After completing a single LED proof of concept, a full array of LEDs will be integrated with the physical case and arduino. This system should provide LED isolation to prevent exposure of LEDs. The system should ideally fit within one quadrant of a pre-made 96 wells-plate, where each LED will match up perfectly with each individual well.

4.2.5 Goal 5: Design control interface

Using LabVIEW or a similar user-friendly control interface, a GUI will be designed to allow custom control of the experiment parameters. The UI should allow for individual control over the exposure time and intensity of each individual LED. The interface should clearly describe which LED's parameters are

being set and should have the ability to provide intermittent exposure within an experiment's duration.

4.2.6 Goal 6: Full Design Iteration

Once a fully operational device is created, the design will be tested and improved to best serve the needs of the lab. The GUI should be fully capable of adjusting any settings desired for any relevant experiment the lab may desire. The LED array should be precise, stable, and fully functioning. The design 6 should also be tested at the extreme ends of possible experimental settings to ensure it covers all likely scenarios.

4.2.7 Goal 7: Testing Under Lab Conditions

The finalized design will be tested in the lab under real experimental conditions to ensure its full functionality and ease of use.

4.3 Proposed Design Tasks

4.3.1 Task 1: Identifying Materials (Goal 1)

Description: Choosing LEDs, Arduino, and cooling solutions for device design.
Challenges: There may be significant research required to choose the best LED and heat management solutions that will fit the experiment parameters of the client's lab. Managing the amount of time spent on research will be vital to identifying the best products while staying on schedule.

4.3.2 Task 2a: Designing the Arduino Code

Description: This will involve making a simple code to verify the LEDs can be controlled via timing and brightness adjustments.
Challenges: Making sure that the LEDs are full controllable will be the biggest challenge.

4.3.3 Task 2b: Testing Purchased LEDs (Goal 1)

Description: If multiple LEDs are identified that could serve as light sources for the device, testing will be done to choose the LED best suited for this project.
Challenges: Designing a test setup that most closely mimics the eventual usage of the LED will be essential to establishing confidence in our test results.

4.3.4 Task 3a: Build Single LED Prototype (Goal 2)

Description: Once the materials are chosen, a single LED exposure circuit will be built and attached to the arduino. This circuit will be programmed to mimic experiment exposure conditions to ensure the LED and circuit will meet the needs of the client's lab experiments.

Challenges: The material task of creating this prototype should be straightforward, but the challenge will be in fully understanding the complete requirements and needs of the experiments being done in the lab.

4.3.5 Task 3b: Filing down LEDs

Description: Due to the size of the new star shaped LEDs, the LEDs are required to be filed down on the ends, by at least 1mm.
Challenges: The LEDs should ideally be filed flat (curves or slants will make positioning of LEDs difficult). Also, filing the LEDs will require precision due to the fact that the part of the LEDs that require filing will be very close to the soldered areas.

4.3.6 Task 4: Designing Device Casing (Goal 3)

Description: Construct a design for the device casing taking into account the different components of the case; the shell, place for arduino, fan, heat sink, etc.
Challenges: None of the engineers on this project have extensive mechanical design experience, so creating a 3D model will require learning new skills. Depending on the heat management solution, creating the smallest most efficient design may require multiple iterations and significant design time.

4.3.7 Task 5: 3D Printing Device Case (Goal 3)

Description: Upon the finalization of our device's case design, the shell will be created using a 3D printer and its connected software. This case will then be equipped to the array, ensuring that all of the portions of the device fit within comfortably.
Challenges: 3D printing, especially with the resources available at UMASS Boston often requires multiple attempts with different printing parameters. The 3D printing settings for orientation, fill, and precision can significantly impact the final product.

4.3.8 Task 6: Building Exposure Circuit (Goal 4)

Description: The full array of LEDs will be assembled, connected to the arduino, and integrated with the case.
Challenges: When assembling the completed design, ensuring that the LEDs and arduino have adequate power may be challenging. Keeping wiring neat and organized can be difficult on small electronics.

4.3.9 Task 7: Testing, verify, and debug functionality (Goal 4)

Description: Verify the functionality of the whole array by ensuring we have full control over the brightness and timing settings of the LEDs.
Challenges: As with any form of debugging, problems may arise from any mistakes made in previous tasks.

4.3.10 Task 8: Creating User Interface (Goal 5)

Description: Use a system design platform, like LabView, to develop an interface that is user friendly and contains inputs and outputs that are compatible with the physical device.
Challenges: The engineers on this project have limited experience programming user interfaces so there may be a significant learning curve in this task.

4.3.11 Task 9: Device Testing (Goals 6 and 7)

Description: The device will be brought into the lab where it will be attempted to be utilized in one of the daily experiments. This task will also include verifying and debugging the system as a whole, in conjunction with the designed GUI.
Challenges: Testing often reveals design flaws that were missed in the design process, so planning for enough time for significant design changes will be important as the project nears conclusion.

4.4 Budget

4.4.1 Proposed Budget

Item	Cost	Quantity	Total
ARDUINO MEGA 2560 REV3	$38.50	2	$77.00
Cree XLamp XP-E High Power LED Star	$4.00	30	$120.00
Wires	N/A	Black + Red	0
Potentiometer (10 pack)	$16.00	1	$16.00
Transistors: VP2106	$0.40	30	$12.00
3-D Printed prototyping material	$0.00	0	$0.00
1/8" x 24" x 48" (nominal) Clear Extruded Acrylic	$18.56	1	$18.56
1W 3W High Power LED lens 14mm 16mm 120degree with black holder For IR CCTV DIY [50pcs]	$12.80	1	$12.80
Arduino Cooling Fan	$3.50	6	$21.00
Metal Housing[Sheets]	$15	2	$30.00
PCB	$10	3	$30.00
		Total:	$337.36

Figure 3: Total budget for the components used for the MLIS prototyping

4.4.2 Total Budget Used In Project

Item	Cost	Quantity	Total
Cree XLamp XP-E High Power LED Star	$4.00	30	$120.00
ARDUINO MEGA 2560 REV3	$38.50	1	$38.50
Perfboard	$5.00	1	$5.00
OSRAM LED (KS DMLN31.23-FYHX-68-J3T3)	$1.00	50	$50.00
Wires	N/A	Black + Red	0
Transistors: VP2106	$0.40	30	$12.00
Arduino Cooling Fan	$3.50	6	$21.00
Resistors(20ohm)	$0.09	30	$2.70
Transistors: 2n3904	$0.04	30	$1.20
C10449_TINA-W Lens	$3.00	10	$30.00
RQ-125B Power Supply	$33.75	1	$33.75
Voltage Regulator MCP1825S-2502E/AB	$0.82	25	$20.50
		Spent so far:	$334.65
		Originally planned budget:	$380

Figure 4: Total budget for the components used for the MLIS prototyping

4.4.3 Final Budget for MLIS Prototype

Item	Quantity	Cost
High Powered LEDs	24	$96.00
Arduino MEGA 2560	1	$38.50
Arduino Cooling Fan	6	$21.00
NPN Transistor: 2N3904	24	$0.96
Resistors (20 Ω)	24	$2.16
Resistors (460 Ω)	24	$2.16
RQ-125B Power Supply	1	$33.75
Lenses	24	$72.00
	Total Cost:	$266.53

Figure 5: Final budget for the final components used for the MLIS prototype.

4.5 Gantt Chart (Timeline)

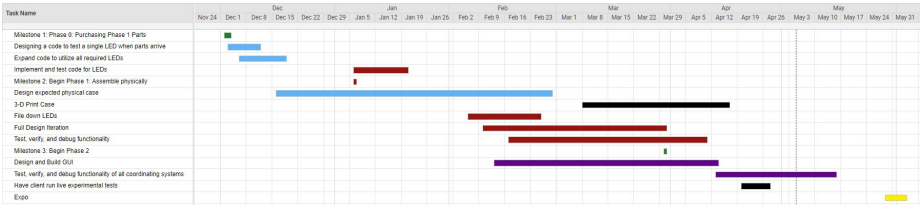

Figure 6: Final gantt chart timeline for the project.

4.6 Milestone Updates Along the Way

Milestone 1

- Confirmed design and materials the client would like for the project.
- A list of materials was sent to the Technical Operations Director for purchasing for Phase 1.

  Behind schedule by 1 week due to complications with finding an appropriate LED to utilize in the device.

- An ongoing search to narrow down a better suited lens/housing for the LEDs.

Milestone 2

- Arduino code was finished for both single and multiple LED implementations.

  But we cannot test it yet due to the wrong type of LED[Surface Mounted] and lack of Arduino

  Surface Mounted LEDs may be too tedious/difficult to experiment with for this project

- Received some parts (Transistors, LEDs, and Arduino Fan have arrived)

  Waiting on Arduino

Milestone 3

- Figured out we were trying to achieve a brightness that was unnecessary
- Overall system turns on and functions as originally planned
- Placed an order for an universal power supply
- Due to the large amount of heat coming from the Arduino, two case designs were created with the direction of the LEDs reversed (will now be aimed downward into the wells, as opposed to upward into them).

Milestone 4

- Extensive heat management no longer needed

  Fixed power management issue that caused a dangerous amount of heat

  But we will keep the same case designs, since they seem practical anyway

- GUI development is still ongoing, remotely
- Cannot get power sensor measured due to campus closure
- The actual system is basically done
- Soldering will have to wait due to lack of soldering equipment

Milestone 5

- GUI is completed, but there are some serial communication issue between the GUI and Arduino that need to be addressed.
- Power supply and voltage regulators are ordered
- Client has informed us brightness control is not worth the complexity of adding in variable parts, but we plan to work on the Arduino's code to get that working, if possible

  Also not to worry about the casing/soldering due to the school closure, for now.

Milestone 6

- GUI is completed and functional
- Power supply and voltage regulators have been received
- We have obtained a multimeter to measure the current going to the LEDs to make sure that the input is constant across any configuration of LEDs. While we me not be able to measure the output of the LEDs, by ensuring that the input is constant we can conclude that the power output from the LEDs is also constant.

Milestone 7

- Circuit was re-simulated in PSpice to ensure that the physically measured values lined up with theoretical value as well as allowed an educated estimate of what resistance should go in between the power supply and each individual LED.
- Circuit was constructed with available parts and evaluated to make sure that the current was consistent between LEDs and configurations.

4.7 Completed Design Goals And Tasks

4.7.1 Choosing a suitable LED

First, the most important component for this project, the LED, must be selected. The specifications required for the project was that the LED must output, at least, a uniform irradiance of 30 mW/cm^2 over a 1.9 cm^2 individual well of the 24-well plate, which will be approximately 60 mW of output power. Any higher output can be lessened through the implementation of a semi-transparent cover or lens. Along with the output power, the wavelength must be approximately 630 nm.

The LED that the design ultimately arrived at was the Cree XLAMP XPE-FRD-3. These LEDs fit close to the required wavelength, with an output range

of 620 nm to 630 nm, and the LED has a minimum drive current at 100 mA and a max drive current of 1 A, which allows a large range of control, in terms of brightness/output. To reach the required output and wavelength, the selected LEDs must function at approximately 500 mA.

Another crucial specification is that the LEDs must be able to fit in a way where they are positioned in the center of each well, to ensure uniform irradiance. The selected LEDs have a size of 19 mm, which is much too large to fit comfortably, due to the LEDs needing to be at least 1.5 mm smaller, or 0.75 mm on each side. The LEDs were filed down, on all four sides, by about 0.8 mm, which allows a bit extra room, compared to filing down exactly 0.75 mm each side.

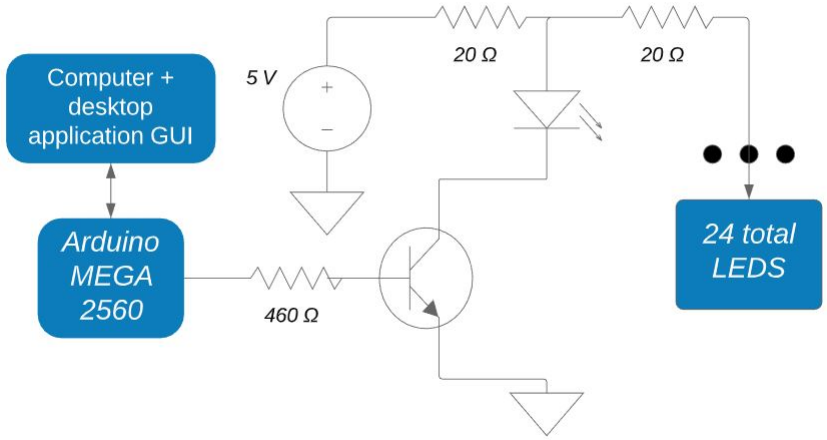

Figure 7: Circuit schematic for one of the 24 LEDs.

4.7.2 Designing the LED Array Circuitry

The schematic in Figure 7. can be referred to visually. The LEDs will be driven by an Arduino MEGA 2560 and a DC power supply. To utilize the Arduino's low current output, a 2N3904 NPN transistor is used, which also acts as an effective power control and switch. The LED Array circuit utilizes a simple design. The LED and transistor circuit is used with a R_b resistance value of 460 Ω and a R_L resistance value of 0 with the power supply at the positive end of the LED. All of the LEDs are set in parallel with the power supply and attached to the collector of the transistor, with individual I/O pins connected to the the base of the transistor, and ground to the remaining emitter of the transistor.

4.7.3 Writing the Arduino Code Program

In order to control whether or not the transistors allowed the current to travel from the LED to ground, thereby completing the circuit and lighting the LED,

the designed user interface was required to control the Arduino through serial communication. By allowing the GUI user to select the USB port they would like to attach the Arduino to, the devices are paired, and serial communication becomes possible.

For the GUI, a Boolean variable must be set for each LED to represent whether the user has decided to turn the LED on or have it remain in the default position (off). When the user selects to begin the experiment, the GUI checks to see which LED has been selected as "On" using the variables and then outputs a unique string which starts with "#", ends with "/n", and is 4 characters long excluding the aforementioned characters. When the timer reaches zero, the GUI then sends a string to the Arduino which turns all of the connected pins to a low output.

For the Arduino, first the Arduino was configured to allow all of the pins used to be outputs. There were then variables created which represented the input of the serial communication, the command within the input string, a variable which will be used to state whether or not the Arduino is connected to a port, and a variable which is used to note whether or not the string has been completed. Within a serial event function, the input of the serial communication from the GUI is added to the input string and also sets the completion variable to "true" if the end characters of the GUI string have been received. The command is then taken from the input string, and each defined port is assigned a unique string from the GUI which, when received, will turn the output of the port to high. Finally, when the Arduino receives the end command from the GUI, all ports on the Arduino used are defined as low.

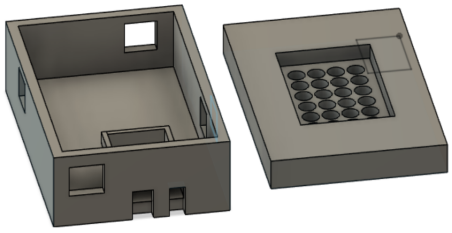

Figure 8: 3-D Printed Case design for the LED Array.

4.7.4 Designing the Physical Case via 3-D Printing

The device's case was designed in order to efficiently store all components of the device while allowing the user to easily place the 24-well plate in line with the LEDs. The easiest way to establish the latter would be through designing the case to have the plate placed on top of the device as it would not only allow for the user to visually confirm that the plate was correctly set. By allowing the plate to be placed on top of the case, it also allowed for the case to have

grooves that the plate would be able to securely fit into; this would minimize the amount of effort it would take for the user to correctly begin the experiment. While grooves were not impossible to implement for a case design that had the case over the wells, they would be much more cumbersome to utilize for the user and there would be no way to visually confirm the placement of the plate. In order to reduce the amount of internal heat generated by the components of the device, 4 Arduino fan slots were implemented into the case design, as shown in Figure 8.

While the device was not intended to be used in a mobile environment, when designing the case's internal structure there needed to be no chance of dislodging any of the components due to any jostling that may accompany placing the plate onto the device or placing the USB wire into the Arduino. With this in mind different sections of the device were created to offer additional support to the Arduino, the power supply, and the perf board. These sections had walls that were specified to the dimensions of each component; each component would be placed into their individual section being safely secured in place.

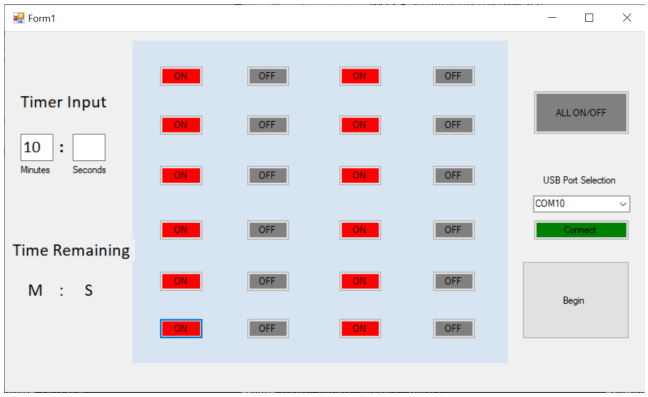

Figure 9: The computer application GUI developed to control the LED Array.

4.7.5 Designing the Graphical User Interface (GUI)

The final design of the GUI is shown in Figure 9. The device will be controlled by a program that is run on the user's chosen device and will connect to the Arduino within the device through the port in the device's case. The program allows the user to choose which LED(s) they would like activated as well as how long they would like the LEDs to be active for. An option for the experimenter to turn on all of the LEDs at once was also an important addition as turning on all of the LEDs manually could be a tedious task. The buttons used in the interface that are used to turn on each individual LED as well as the button which turns on all LEDs also function as a means to turn off one/all of the LEDs; this was done to reduce congestion in the interface, and the buttons change color and text to represent the current state of the LED(s).

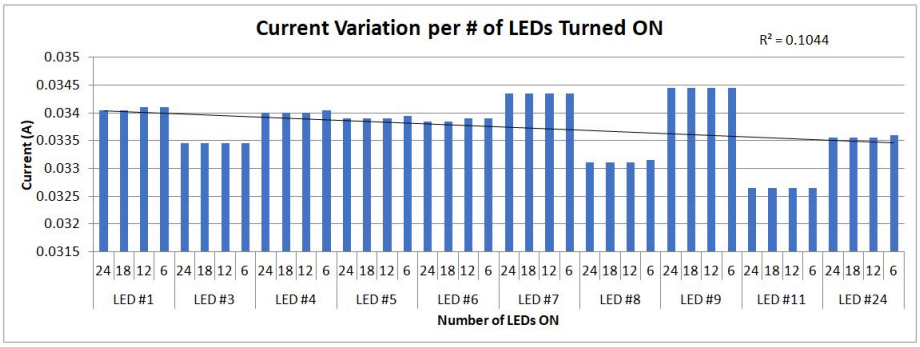

Figure 10: Current Variation when a certain number of LEDs are turned on.

With the original goal of the device being to allow the experimenters to be more efficient in their research, there is a need for the device (and in turn the GUI) to allow the user to perform other tasks while the experiment is underway. In order to allow the user to have greater freedom when conducting the experiment, the device needs to be as automatic as possible. When designing the GUI having this in mind, the program alerts the user to the end of the experiment through both an audio cue and a popup notification. Another way the interface allows for an increased user freedom is that it automatically turns off all of the LEDs when the experiment ends, ensuring that if the user is not present with the device when the experiment ends there will be no overworking of the cells.

5 Results

For the results of the MLIS, the actual optical power output could not be measured directly. We have decided to measure the consistency of the output instead. The consistency is much more important, as the output can always be increased by changing to a higher voltage and current rated power supply.

As shown in Figure 10, the current does not change more than 100 μA for each individual LED when a different number of LEDs are turned off or on, which is less than 1% difference. This is incredibly important because in other LED-Array systems, when a certain number of LEDs is turned off, the LEDs will receive an increase of current, or a decrease when more LEDs are turned on. However, the LEDs differ about, at most, 4% in current output when compared to each other, for example LED #8 and LED #9, the difference is about 0.0015 A. Overall, the individual LEDs may differ 4%, but the LED array as a whole remains relatively consistent, which will be useful in conducting multi-well experiments.

In terms of the power output itself, we were unable to confirm the output effectively. Based off of the voltage and the low current output measurements that have been made, the power per LED was about 89 mW, as shown in Figure

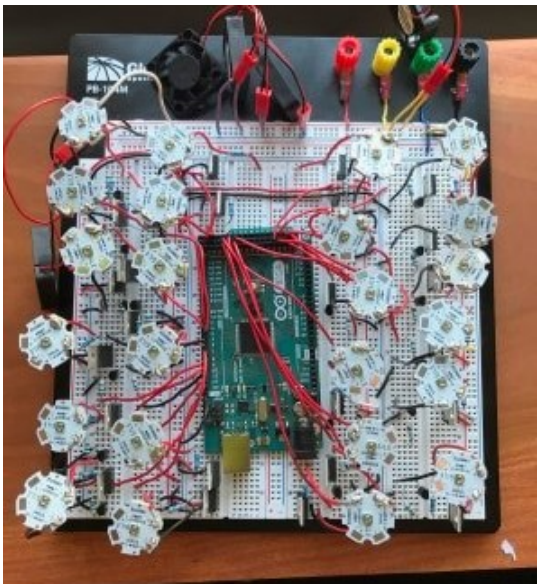

Figure 11: Final prototype of the Multi-well LED Irradiation System on a breadboard.

12. Though, these measurements do not make sense based on the LED's data sheets and our previous experience with these LEDs. The LEDs are not suppose to function below 100 mA, but the measurements come out to 35 mA, so we were unable to accept these measurements and use them to calculate the power output approximates. The power output will have to be measured via a sensor in order to retrieve any valuable measurements related to the power output.

LED	Number of LEDs (ON)	Voltage Drop (V)	Voltage	Current (A)	Current (mA)	Power (Watts)	Power(mW)	Power Per Well (mW/cm^2)
6	24	0.677	2.491	0.03385	33.85	0.0863175	86.3175	6.13486141
	18	0.677	2.496	0.03385	33.85	0.0863175	86.3175	6.13486141
	12	0.678	2.497	0.0339	33.9	0.086445	86.445	6.14392324
	6	0.678	2.492	0.0339	33.9	0.086445	86.445	6.14392324
1	24	0.681	2.493	0.03405	34.05	0.0868275	86.8275	6.17110874
	18	0.681	2.494	0.03405	34.05	0.0868275	86.8275	6.17110874
	12	0.682	2.491	0.0341	34.1	0.086955	86.955	6.18017058
	6	0.682	2.495	0.0341	34.1	0.086955	86.955	6.18017058
4	24	0.68	2.498	0.034	34	0.0867	86.7	6.16204691
	18	0.68	2.497	0.034	34	0.0867	86.7	6.16204691
	12	0.68	2.496	0.034	34	0.0867	86.7	6.16204691
	6	0.681	2.491	0.03405	34.05	0.0868275	86.8275	6.17110874
5	24	0.678	2.492	0.0339	33.9	0.086445	86.445	6.14392324
	18	0.678	2.498	0.0339	33.9	0.086445	86.445	6.14392324
	12	0.678	2.493	0.0339	33.9	0.086445	86.445	6.14392324
	6	0.679	2.497	0.03395	33.95	0.0865725	86.5725	6.15298507
3	24	0.669	2.492	0.03345	33.45	0.0852975	85.2975	6.06236674
	18	0.669	2.493	0.03345	33.45	0.0852975	85.2975	6.06236674
	12	0.669	2.494	0.03345	33.45	0.0852975	85.2975	6.06236674
	6	0.669	2.497	0.03345	33.45	0.0852975	85.2975	6.06236674
8	24	0.662	2.494	0.0331	33.1	0.084405	84.405	5.9989339
	18	0.662	2.492	0.0331	33.1	0.084405	84.405	5.9989339
	12	0.662	2.492	0.0331	33.1	0.084405	84.405	5.9989339
	6	0.663	2.493	0.03315	33.15	0.0845325	84.5325	6.00799574
7	24	0.687	2.497	0.03435	34.35	0.0875925	87.5925	6.22547974
	18	0.687	2.494	0.03435	34.35	0.0875925	87.5925	6.22547974
	12	0.687	2.491	0.03435	34.35	0.0875925	87.5925	6.22547974
	6	0.687	2.491	0.03435	34.35	0.0875925	87.5925	6.22547974
11	24	0.653	2.498	0.03265	32.65	0.0832575	83.2575	5.9173774
	18	0.653	2.497	0.03265	32.65	0.0832575	83.2575	5.9173774
	12	0.653	2.493	0.03265	32.65	0.0832575	83.2575	5.9173774
	6	0.653	2.496	0.03265	32.65	0.0832575	83.2575	5.9173774
24	24	0.671	2.495	0.03355	33.55	0.0855525	85.5525	6.08049041
	18	0.671	2.494	0.03355	33.55	0.0855525	85.5525	6.08049041
	12	0.671	2.491	0.03355	33.55	0.0855525	85.5525	6.08049041
	6	0.672	2.493	0.0336	33.6	0.08568	85.68	6.08955224
9	24	0.689	2.494	0.03445	34.45	0.0878475	87.8475	6.24360341
	18	0.689	2.496	0.03445	34.45	0.0878475	87.8475	6.24360341
	12	0.689	2.494	0.03445	34.45	0.0878475	87.8475	6.24360341
	6	0.689	2.494	0.03445	34.45	0.0878475	87.8475	6.24360341

Figure 12: Measurements done to the circuit for some randomly selected LEDs in the array.

6 Conclusion

In the end, the MLIS is a successful prototype that fulfils many of the initial requirements. The MLIS is powered by a generic wall port, controlled via a computer application GUI, encased in a 3-D printed case that allows the 24-well plate to be placed directly above, along with a cooling system, and only costs approximately $300. However, the MLIS is only 90% completed, as some

sections of the project could not be completed due to the present pandemic. Ultimately, compared to what was originally planned, the MLIS only lacks the 3-D printed case and construction of the circuit within the case.

7 Future Work

In the end, we have made meaningful and substantial progress towards the completion of the project, but unfortunately due to the COVID-19 pandemic, we were not able to complete crucial aspects of our semester's plan. Although the groundwork for a completed device is solid and we have made some design adjustments to allow greater room for error in our initial assumptions, there is still a nonminimal amount of work that would have to be taken for the device to be operational.

The most notable task that must be completed is the measurement of the power output of each LED. Once we realized that there would be no way for our group to accurately determine how much optical power the LED was producing, we made increased efforts to ensure that the LEDs would be consistent both between each other as well as between different configuration settings. If the power being emitted from the LEDs is not enough for the purposes of the experiments, the current power supply should be more than capable of handling an increased power output to the circuit; the increase in power would most easily be achieved through obtaining a set of voltage regulators which allow more voltage through the circuit (the highest most likely being about 3.5-4 volts) as well as a lower resistance for the current limiting resistors.

Another important task that was unable to be achieved was the validation of our current lenses for the LEDs. The lenses were designed to allow the energy of the LEDs to be evenly distributed across the wells, and while we were able to purchase a few lenses for testing we were unable to verify that these lenses efficiently performed the task. If these lenses prove to be inadequate, the person continuing this project would have to either find new lenses to purchase or an opaque sheet/cover which would also serve this purpose; both of these outcomes could possibly result in this person also having to slightly alter the current 3D case design.

Other tasks that we were unable to complete due to the closing of the campus were the 3D printing of the case, the soldering of the circuit to the breadboard, and the construction of the circuit within the case.

7.1 Tasks and Timeline

(i) 3-D Print Case. Estimated time: 1 week

(ii) Solder all components to a perfboard. Estimated time: 1 week

(iii) Assemble all soldered components into the Case. This will also include inserted the fans to the case. Estimated time: 1 Week.

(iv) If possible, measure and confirm LED power output via a sensor. Estimated time: 1 Week.

(v) If needed, order and implemented lenses and/or a semi-transparent cover. This will require trial and error as the power output would need to be measured each time. Estimated time: 2-3 Weeks.

(vi) Instruction manual must be written up to match the functionality of the complete MLIS. Estimated time: 1 Week.

7.2 Budget

The majority of the parts have already been ordered and are in hand. And estimated cost of $100 to $150 would be needed for the remaining lenses and/or semi-transparent cover.

7.3 Appendix

7.3.1 Arduino Code

```
String Input = "";
    boolean Complete = false;
    String Command = "";
    boolean Connection = false;
    void setup()
    Serial.begin(9600);
    while (!Serial)
    ;
    pinMode(12, OUTPUT);
    pinMode(11, OUTPUT);
    pinMode(10, OUTPUT);
    pinMode(9, OUTPUT);
    pinMode(8, OUTPUT);
    pinMode(7, OUTPUT);
    pinMode(6, OUTPUT);
    pinMode(5, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(3, OUTPUT);
    pinMode(2, OUTPUT);
    pinMode(23, OUTPUT);
    pinMode(28, OUTPUT);
    pinMode(29, OUTPUT);
    pinMode(30, OUTPUT);
    pinMode(31, OUTPUT);
    pinMode(32, OUTPUT);
    pinMode(33, OUTPUT);
    pinMode(48, OUTPUT);
    pinMode(49, OUTPUT);
    pinMode(50, OUTPUT);
    pinMode(51, OUTPUT);
    pinMode(46, OUTPUT);
    pinMode(53, OUTPUT);
    // the loop function runs over and over again forever
    void loop()
    if(Complete)
    Complete = false;
    getCommand();
    if(Command.equals("STOP"))
    digitalWrite(12, LOW);
    digitalWrite(11, LOW);
    digitalWrite(10, LOW);
    digitalWrite(9, LOW);
```

```
digitalWrite(8, LOW);
digitalWrite(7, LOW);
digitalWrite(6, LOW);
digitalWrite(5, LOW);
digitalWrite(4, LOW);
digitalWrite(3, LOW);
digitalWrite(2, LOW);
digitalWrite(53, LOW);
digitalWrite(23, LOW);
digitalWrite(28, LOW);
digitalWrite(29, LOW);
digitalWrite(30, LOW);
digitalWrite(31, LOW);
digitalWrite(32, LOW);
digitalWrite(33, LOW);
digitalWrite(48, LOW);
digitalWrite(49, LOW);
digitalWrite(50, LOW);
digitalWrite(51, LOW);
digitalWrite(46, LOW);
else if(Command.equals("ENDS"))
digitalWrite(12, LOW);
digitalWrite(11, LOW);
digitalWrite(10, LOW);
digitalWrite(9, LOW);
digitalWrite(8, LOW);
digitalWrite(7, LOW);
digitalWrite(6, LOW);
digitalWrite(5, LOW);
digitalWrite(4, LOW);
digitalWrite(3, LOW);
digitalWrite(2, LOW);
digitalWrite(53, LOW);
digitalWrite(23, LOW);
digitalWrite(28, LOW);
digitalWrite(29, LOW);
digitalWrite(30, LOW);
digitalWrite(31, LOW);
digitalWrite(32, LOW);
digitalWrite(33, LOW);
digitalWrite(48, LOW);
digitalWrite(49, LOW);
digitalWrite(50, LOW);
digitalWrite(51, LOW);
digitalWrite(46, LOW);
else if(Command.equals("01on"))
```

```
digitalWrite(12, HIGH);
else if(Command.equals("02on"))
digitalWrite(11, HIGH);
else if(Command.equals("03on"))
digitalWrite(10, HIGH);
else if(Command.equals("04on"))
digitalWrite(9, HIGH);
else if(Command.equals("05on"))
digitalWrite(8, HIGH);
else if(Command.equals("06on"))
digitalWrite(7, HIGH);
else if(Command.equals("07on"))
digitalWrite(6, HIGH);
else if(Command.equals("08on"))
digitalWrite(5, HIGH);
else if(Command.equals("09on"))
digitalWrite(4, HIGH);
else if(Command.equals("10on"))
digitalWrite(3, HIGH);
else if(Command.equals("11on"))
digitalWrite(2, HIGH);
else if(Command.equals("12on"))
digitalWrite(53, HIGH);
else if(Command.equals("13on"))
digitalWrite(23, HIGH);
else if(Command.equals("14on"))
digitalWrite(28, HIGH);
else if(Command.equals("15on"))
digitalWrite(29, HIGH);
else if(Command.equals("16on"))
digitalWrite(30, HIGH);
else if(Command.equals("17on"))
digitalWrite(31, HIGH);
else if(Command.equals("18on"))
digitalWrite(32, HIGH);
else if(Command.equals("19on"))
digitalWrite(33, HIGH);
else if(Command.equals("20on"))
digitalWrite(48, HIGH);
else if(Command.equals("21on"))
digitalWrite(49, HIGH);
else if(Command.equals("22on"))
digitalWrite(50, HIGH);
else if(Command.equals("23on"))
digitalWrite(51, HIGH);
else if(Command.equals("24on"))
```

```
digitalWrite(46, HIGH);
Input = "";
void getCommand()
if(Input.length() ¿ 0)
Command = Input.substring(1,5);
void serialEvent()
while (Serial.available())
char inChar = (char)Serial.read();
:
Input += inChar;
if (inChar == '/n')
Complete = true;
```

7.3.2 GUI

```
using System;
    using System.Collections.Generic;
    using System.ComponentModel;
    using System.Data;
    using System.Drawing;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;
    using System.Windows.Forms;
    using System.IO.Ports;
    namespace GUI
    public partial class Form1 : Form
    bool on1 = true;
    bool on2 = true;
    bool on3 = true;
    bool on4 = true;
    bool on5 = true;
    bool on6 = true;
    bool on7 = true;
    bool on8 = true;
    bool on9 = true;
    bool on10 = true;
    bool on11 = true;
    bool on12 = true;
    bool on13 = true;
    bool on14 = true;
    bool on15 = true;
    bool on16 = true;
    bool on17 = true;
    bool on18 = true;
    bool on19 = true;
```

```
bool on20 = true;
bool on21 = true;
bool on22 = true;
bool on23 = true;
bool on24 = true;
bool on25 = true;
bool Connection = false;
string[] ports;
SerialPort port;
int S;
int M;
public Form1()
InitializeComponent();
getAvailableComPorts();
foreach (string port in ports)
comboBox1.Items.Add(port);
Console.WriteLine(port);
if (ports[0] != null)
comboBox1.SelectedItem = ports[0];
void getAvailableComPorts()
ports = SerialPort.GetPortNames();
private void connectToArduino()
Connection = true;
string selectedPort = comboBox1.GetItemText(comboBox1.SelectedItem);
port = new SerialPort(selectedPort, 9600, Parity.None, 8, StopBits.One);
port.Open();
port.Write("DAWN");
button27.Text = "Disconnect";
button27.BackColor = Color.DarkRed;
private void disconnectFromArduino()
Connection = false; port.Write("STOP");
port.Close();
button27.Text = "Connect";
button27.BackColor = Color.Green;
private void Form1_Load(objectsender, EventArgse)
button1.Text = "OFF";
button1.BackColor = Color.Gray;
button2.Text = "OFF";
button2.BackColor = Color.Gray;
button3.Text = "OFF";
button3.BackColor = Color.Gray;
button4.Text = "OFF";
button4.BackColor = Color.Gray;
button5.Text = "OFF";
button5.BackColor = Color.Gray;
button6.Text = "OFF";
```

button6.BackColor = Color.Gray;
button7.Text = "OFF";
button7.BackColor = Color.Gray;
button8.Text = "OFF";
button8.BackColor = Color.Gray;
button9.Text = "OFF";
button9.BackColor = Color.Gray;
button10.Text = "OFF";
button10.BackColor = Color.Gray;
button11.Text = "OFF";
button11.BackColor = Color.Gray;
button12.Text = "OFF";
button12.BackColor = Color.Gray;
button13.Text = "OFF";
button13.BackColor = Color.Gray;
button14.Text = "OFF";
button14.BackColor = Color.Gray;
button15.Text = "OFF";
button15.BackColor = Color.Gray;
button16.Text = "OFF";
button16.BackColor = Color.Gray;
button17.Text = "OFF";
button17.BackColor = Color.Gray;
button18.Text = "OFF";
button18.BackColor = Color.Gray;
button19.Text = "OFF";
button19.BackColor = Color.Gray;
button20.Text = "OFF";
button20.BackColor = Color.Gray;
button21.Text = "OFF";
button21.BackColor = Color.Gray;
button22.Text = "OFF";
button22.BackColor = Color.Gray;
button23.Text = "OFF";
button23.BackColor = Color.Gray;
button24.Text = "OFF";
button24.BackColor = Color.Gray;
button25.Text = "ALL ON/OFF";
button25.BackColor = Color.Gray;
button27.Text = "Connect";
button27.BackColor = Color.Green;
private void button1$_{C}click_{2}(objectsender, EventArgse)$
if (on1)
button1.Text = "ON";
button1.BackColor = Color.Red;
on1 = false;

else
button1.Text = "OFF";
button1.BackColor = Color.Gray;
on1 = true;
private void button2$_C$*click*(*objectsender*, *EventArgse*)
if (on2)
button2.Text = "ON";
button2.BackColor = Color.Red;
on2 = false;
else
button2.Text = "OFF";
button2.BackColor = Color.Gray;
on2 = true;
private void button3$_C$*click*(*objectsender*, *EventArgse*)
if (on3)
button3.Text = "ON";
button3.BackColor = Color.Red;
on3 = false;
else
button3.Text = "OFF";
button3.BackColor = Color.Gray;
on3 = true;
private void button4$_C$*click*(*objectsender*, *EventArgse*)
if (on4)
button4.Text = "ON";
button4.BackColor = Color.Red;
on4 = false;
else
button4.Text = "OFF";
button4.BackColor = Color.Gray;
on4 = true;
private void button5$_C$*click*(*objectsender*, *EventArgse*)
if (on5)
button5.Text = "ON";
button5.BackColor = Color.Red;
on5 = false;
else
button5.Text = "OFF";
button5.BackColor = Color.Gray;
on5 = true;
private void button6$_C$*click*(*objectsender*, *EventArgse*)
if (on6)
button6.Text = "ON";
button6.BackColor = Color.Red;
on6 = false;
else button6.Text = "OFF";

button6.BackColor = Color.Gray;
on6 = true;
private void button7$_C$*click*(*objectsender*, *EventArgse*)
if (on7)
button7.Text = "ON";
button7.BackColor = Color.Red;
on7 = false;
else
button7.Text = "OFF";
button7.BackColor = Color.Gray;
on7 = true;
private void button8$_C$*click*(*objectsender*, *EventArgse*)
if (on8)
button8.Text = "ON";
button8.BackColor = Color.Red;
on8 = false;
else
button8.Text = "OFF";
button8.BackColor = Color.Gray;
on8 = true;
private void button9$_C$*click*(*objectsender*, *EventArgse*)
if (on9)
button9.Text = "ON";
button9.BackColor = Color.Red;
on9 = false;
else
button9.Text = "OFF";
button9.BackColor = Color.Gray;
on9 = true;
private void button10$_C$*click*(*objectsender*, *EventArgse*)
if (on10)
button10.Text = "ON";
button10.BackColor = Color.Red;
on10 = false;
else
button10.Text = "OFF";
button10.BackColor = Color.Gray;
on10 = true;
private void button11$_C$*click*(*objectsender*, *EventArgse*)
if (on11)
button11.Text = "ON";
button11.BackColor = Color.Red;
on11 = false;
else
button11.Text = "OFF";
button11.BackColor = Color.Gray;

on11 = true;
private void button12$_C$*click*(*objectsender*, *EventArgse*)
if (on12)
button12.Text = "ON";
button12.BackColor = Color.Red;
on12 = false;
else
button12.Text = "OFF";
button12.BackColor = Color.Gray;
on12 = true;
private void button13$_C$*click*(*objectsender*, *EventArgse*)
if (on13)
button13.Text = "ON";
button13.BackColor = Color.Red;
on13 = false;
else
button13.Text = "OFF";
button13.BackColor = Color.Gray;
on13 = true;
private void button14$_C$*click*(*objectsender*, *EventArgse*)
if (on14)
button14.Text = "ON";
button14.BackColor = Color.Red;
on14 = false;
else
button14.Text = "OFF";
button14.BackColor = Color.Gray;
on14 = true;
private void button15$_C$*click*(*objectsender*, *EventArgse*)
if (on15)
button15.Text = "ON";
button15.BackColor = Color.Red;
on15 = false;
else
button15.Text = "OFF";
button15.BackColor = Color.Gray;
on15 = true;
private void button16$_C$*click*(*objectsender*, *EventArgse*)
if (on16)
button16.Text = "ON";
button16.BackColor = Color.Red;
on16 = false;
else
button16.Text = "OFF";
button16.BackColor = Color.Gray;
on16 = true;

```
private void button17_Click(object sender, EventArgs e)
if (on17)
button17.Text = "ON";
button17.BackColor = Color.Red;
on17 = false;
else
button17.Text = "OFF";
button17.BackColor = Color.Gray;
on17 = true;
private void button18_Click(object sender, EventArgs e)
if (on18)
button18.Text = "ON";
button18.BackColor = Color.Red;
on18 = false;
else
button18.Text = "OFF";
button18.BackColor = Color.Gray;
on18 = true;
private void button19_Click(object sender, EventArgs e)
if (on19)
button19.Text = "ON";
button19.BackColor = Color.Red;
on19 = false;
else
button19.Text = "OFF";
button19.BackColor = Color.Gray;
on19 = true;
private void button20_Click(object sender, EventArgs e)
if (on20)
button20.Text = "ON";
button20.BackColor = Color.Red;
on20 = false;
else
button20.Text = "OFF";
button20.BackColor = Color.Gray;
on20 = true;
private void button21_Click(object sender, EventArgs e)
if (on21)
button21.Text = "ON";
button21.BackColor = Color.Red;
on21 = false;
else
button21.Text = "OFF";
button21.BackColor = Color.Gray;
on21 = true;
private void button22_Click(object sender, EventArgs e)
```

if (on22)
button22.Text = "ON";
button22.BackColor = Color.Red;
on22 = false;
else
button22.Text = "OFF";
button22.BackColor = Color.Gray;
on22 = true;
private void button23$_C$*click*(*objectsender*, *EventArgse*)
if (on23)
button23.Text = "ON";
button23.BackColor = Color.Red;
on23 = false;
else
button23.Text = "OFF";
button23.BackColor = Color.Gray;
on23 = true;
private void button24$_C$*click*(*objectsender*, *EventArgse*)
if (on24)
button24.Text = "ON";
button24.BackColor = Color.Red;
on24 = false;
else
button24.Text = "OFF";
button24.BackColor = Color.Gray;
on24 = true;
private void button25$_C$*click*(*objectsender*, *EventArgse*)
if (on25)
button1.Text = "ON";
button1.BackColor = Color.Red;
on1 = false;
button2.Text = "ON";
button2.BackColor = Color.Red;
on2 = false;
button3.Text = "ON";
button3.BackColor = Color.Red;
on3 = false;
button4.Text = "ON";
button4.BackColor = Color.Red;
on4 = false;
button5.Text = "ON";
button5.BackColor = Color.Red;
on5 = false;
button6.Text = "ON";
button6.BackColor = Color.Red;
on6 = false;

```
button7.Text = "ON";
button7.BackColor = Color.Red;
on7 = false;
button8.Text = "ON";
button8.BackColor = Color.Red;
on8 = false;
button9.Text = "ON";
button9.BackColor = Color.Red;
on9 = false;
button10.Text = "ON";
button10.BackColor = Color.Red;
on10 = false;
button11.Text = "ON";
button11.BackColor = Color.Red;
on11 = false;
button12.Text = "ON";
button12.BackColor = Color.Red;
on12 = false;
button13.Text = "ON";
button13.BackColor = Color.Red;
on13 = false;
button14.Text = "ON";
button14.BackColor = Color.Red;
on14 = false;
button15.Text = "ON";
button15.BackColor = Color.Red;
on15 = false;
button16.Text = "ON";
button16.BackColor = Color.Red;
on16 = false;
button17.Text = "ON";
button17.BackColor = Color.Red;
on17 = false;
button18.Text = "ON";
button18.BackColor = Color.Red;
on18 = false;
button19.Text = "ON";
button19.BackColor = Color.Red;
on19 = false;
button20.Text = "ON";
button20.BackColor = Color.Red;
on20 = false;
button21.Text = "ON";
button21.BackColor = Color.Red;
on21 = false;
button22.Text = "ON";
```

```
button22.BackColor = Color.Red;
on22 = false;
button23.Text = "ON";
button23.BackColor = Color.Red;
on23 = false;
button24.Text = "ON";
button24.BackColor = Color.Red;
on24 = false;
button25.Text = "ALL ON";
button25.BackColor = Color.Red;
on25 = false;
else
button1.Text = "OFF";
button1.BackColor = Color.Gray;
on1 = true;
button2.Text = "OFF";
button2.BackColor = Color.Gray;
on2 = true;
button3.Text = "OFF";
button3.BackColor = Color.Gray;
on3 = true;
button4.Text = "OFF";
button4.BackColor = Color.Gray;
on4 = true;
button5.Text = "OFF";
button5.BackColor = Color.Gray;
on5 = true;
button6.Text = "OFF";
button6.BackColor = Color.Gray;
on6 = true;
button7.Text = "OFF";
button7.BackColor = Color.Gray;
on7 = true;
button8.Text = "OFF";
button8.BackColor = Color.Gray;
on8 = true;
button9.Text = "OFF";
button9.BackColor = Color.Gray;
on9 = true;
button10.Text = "OFF";
button10.BackColor = Color.Gray;
on10 = true;
button11.Text = "OFF";
button11.BackColor = Color.Gray;
on11 = true;
button12.Text = "OFF";
```

```
button12.BackColor = Color.Gray;
on12 = true;
button13.Text = "OFF";
button13.BackColor = Color.Gray;
on13 = true;
button14.Text = "OFF";
button14.BackColor = Color.Gray;
on14 = true;
button15.Text = "OFF";
button15.BackColor = Color.Gray;
on15 = true;
button16.Text = "OFF";
button16.BackColor = Color.Gray;
on16 = true;
button17.Text = "OFF";
button17.BackColor = Color.Gray;
on17 = true;
button18.Text = "OFF";
button18.BackColor = Color.Gray;
on18 = true;
button19.Text = "OFF";
button19.BackColor = Color.Gray;
on19 = true;
button20.Text = "OFF";
button20.BackColor = Color.Gray;
on20 = true;
button21.Text = "OFF";
button21.BackColor = Color.Gray;
on21 = true;
button22.Text = "OFF";
button22.BackColor = Color.Gray;
on22 = true;
button23.Text = "OFF";
button23.BackColor = Color.Gray;
on23 = true;
button24.Text = "OFF";
button24.BackColor = Color.Gray;
on24 = true;
button25.Text = "ALL OFF";
button25.BackColor = Color.Gray;
on25 = true;
public void timer2_Tick(objectsender, EventArgse)
S = S - 1;
if (S == -1)
M = M - 1;
S = 59;
```

```
if (S == 0  M == 0)
timer2.Stop();
port.Write("ENDS");
label7.Text = "00";
label8.Text = "00";
button1.Text = "OFF";
button1.BackColor = Color.Gray;
on1 = true;
button2.Text = "OFF";
button2.BackColor = Color.Gray;
on2 = true;
button3.Text = "OFF";
button3.BackColor = Color.Gray;
on3 = true;
button4.Text = "OFF";
button4.BackColor = Color.Gray;
on4 = true;
button5.Text = "OFF";
button5.BackColor = Color.Gray;
on5 = true;
button6.Text = "OFF";
button6.BackColor = Color.Gray;
on6 = true;
button7.Text = "OFF";
button7.BackColor = Color.Gray;
on7 = true;
button8.Text = "OFF";
button8.BackColor = Color.Gray;
on8 = true;
button9.Text = "OFF";
button9.BackColor = Color.Gray;
on9 = true;
button10.Text = "OFF";
button10.BackColor = Color.Gray;
on10 = true;
button11.Text = "OFF";
button11.BackColor = Color.Gray;
on11 = true;
button12.Text = "OFF";
button12.BackColor = Color.Gray;
on12 = true;
button13.Text = "OFF";
button13.BackColor = Color.Gray;
on13 = true;
button14.Text = "OFF";
button14.BackColor = Color.Gray;
```

```
on14 = true;
button15.Text = "OFF";
button15.BackColor = Color.Gray;
on15 = true;
button16.Text = "OFF";
button16.BackColor = Color.Gray;
on16 = true;
button17.Text = "OFF";
button17.BackColor = Color.Gray;
on17 = true;
button18.Text = "OFF";
button18.BackColor = Color.Gray;
on18 = true;
button19.Text = "OFF";
button19.BackColor = Color.Gray;
on19 = true;
button20.Text = "OFF";
button20.BackColor = Color.Gray;
on20 = true;
button21.Text = "OFF";
button21.BackColor = Color.Gray;
on21 = true;
button22.Text = "OFF";
button22.BackColor = Color.Gray;
on22 = true;
button23.Text = "OFF";
button23.BackColor = Color.Gray;
on23 = true;
button24.Text = "OFF";
button24.BackColor = Color.Gray;
on24 = true;
button25.Text = "ALL OFF";
button25.BackColor = Color.Gray;
on25 = true;
string text = "Experiment Complete";
MessageBox.Show(text);
string minutes = Convert.ToString(M);
string seconds = Convert.ToString(S);
label7.Text = minutes;
label8.Text = seconds;
public void button26_Click(objectsender, EventArgse)
if (textBox1.Text == "")
textBox1.Text = "0";
if (textBox2.Text == "")
textBox2.Text = "0";
M = Convert.ToInt32(textBox1.Text);
```

```
S = Convert.ToInt32(textBox2.Text);
if (on1 == false)
port.Write("01on");
if (on2 == false)
port.Write("02on");
if (on3 == false)
port.Write("03on");
if (on4 == false)
port.Write("04on");
if (on5 == false)
port.Write("05on");
if (on6 == false)
port.Write("06on");
if (on7 == false)
port.Write("07on");
if (on8 == false)
port.Write("08on");
if (on9 == false)
port.Write("09on");
if (on10 == false)
port.Write("10on");
if (on11 == false)
port.Write("11on");
if (on12 == false)
port.Write("12on");
if (on13 == false)
port.Write("13on");
if (on14 == false)
port.Write("14on");
if (on15 == false)
port.Write("15on");
if (on16 == false)
port.Write("16on");
if (on17 == false)
port.Write("17on");
if (on18 == false)
port.Write("18on");
if (on19 == false)
port.Write("19on");
if (on20 == false)
port.Write("20on");
if (on21 == false)
port.Write("21on");
if (on22 == false)
port.Write("22on");
if (on23 == false)
```

port.Write(”23on”);
if (on24 == false)
port.Write(”24on”);
timer2.Start();
private void button27$_{C}$*lick*(*objectsender*, *EventArgse*)
if (!Connection)
connectToArduino();
else
disconnectFromArduino();

References

[1] "Quantum.umb.edu. (2019)." *CBPM Website*.

[2] "Y.-C. Chen, X. Lou, Z. Zhang, P. Ingram, and E. Yoon, "High-Throughput Cancer Cell Sphere Formation for Characterizing the Efficacy of Photo Dynamic Therapy in 3D Cell Cultures.," Scientific reports, vol. 5, no. 1, pp. 12175–12175, 2015."

[3] "J. Hempstead et al., "Low-cost photodynamic therapy devices for global health settings: Characterization of battery-powered LED performance and smartphone imaging in 3D tumor models.," Scientific reports, vol. 5, no. 1, pp. 10093–10093, 2015."

[4] "H. Kagel, H. Jacobs, F. Bier, J. Glökler, and M. Frohme, "A Novel Microtiter Plate Format High Power Open Source LED Array," Photonics, vol. 6, p. 17, Feb. 2019.."

[5] "J. Matsumoto et al., "Photodynamic therapy of human biliary cancer cell line using combination of phosphorus porphyrins and light emitting diode," Bioorganic & Medicinal Chemistry, vol. 25, Oct. 2017."

[6] "M. El-Khatib et al., "Aminolevulinic Acid-Mediated Photodynamic Therapy of Human Meningioma: An in Vitro Study on Primary Cell Lines," International Journal of Molecular Sciences, vol. 16, pp. 9936–9948, May 2015."

[7] "C. G. A. und Microsysteme, "LED Array ceLED 96 — CETONI GmbH." [Online]."

[8] "B. Roberge, R. Roberts, I. Shikh, I. Lys, B. Koerner, and T. Mollnow, "LED-based fixtures and related methods for thermal management," US7828465B2, 09-Nov-2010."

[9] "C. Biber, "Basics of Thermal Design for LEDs," in Thermal Management for LED Applications, C. J. M. Lasance and A. Poppe, Eds. New York, NY: Springer New York, 2014, pp. 53–69."

[10] "P. Agostinis et al., "PHOTODYNAMIC THERAPY OF CANCER: AN UPDATE," CA Cancer J Clin, vol. 61, no. 4, pp. 250–281, 2011."

[11] "G. Ablon, "Phototherapy with Light Emitting Diodes," J Clin Aesthet Dermatol, vol. 11, no. 2, pp. 21–27, Feb. 2018."

[12] "L. Asfa - Wossen, "Thermoplastics show their feathers.(Environment and Sustainability)(use of waste chicken feathers to make thermoplastics)," Materials World, vol. 19, no. 5, p. 6, 2011."

[13] ""How does 3D Printing impact the environment? Everything you NEED to know!", Geeetech Blog, 2019. [Online]."