**P.L.E.D**

**Plaque Laser Engraver Device**

**Casey Wood**

**Zachary Garrard**

**Justin Cox**

**Tate Shorthill**

**InCaseyWood@gmail.com**

**ddcbanz@gmail.com**

**Justin.N.Cox@gmail.com**

**ShorthillT@gmail.com**

**03/29/2016**

**Table of Contents**

[1. Executive Summary 3](#_Toc448785917)

[2. Introduction 4](#_Toc448785918)

[3. Requirements 5](#_Toc448785919)

[3.1 Functional Requirements 5](#_Toc448785920)

[3.2 Non-Functional Requirements 5](#_Toc448785929)

[4. Specifications 6](#_Toc448785942)

[4.1 Performance Characteristics 6](#_Toc448785943)

[4.2 Physical characteristics 6](#_Toc448785956)

[4.3 Environment 6](#_Toc448785963)

[4.3.1 Natural Environments 6](#_Toc448785964)

[4.3.2 Induced Environments 7](#_Toc448785967)

[4.4 Cleanliness 7](#_Toc448785970)

[4.4.1 Electromagnetic interference (EMI) 7](#_Toc448785971)

[4.4.2 Ventilation 7](#_Toc448785973)

[5. Methods 8](#_Toc448785975)

[6. Results 14](#_Toc448785976)

[7. Discussion 17](#_Toc448785977)

[8. Conclusion 18](#_Toc448785978)

[9. Appendices 19](#_Toc448785979)

[9.1 Appendix 1 – Budget 19](#_Toc448785980)

[9.2 Appendix 2 – Timeline 20](#_Toc448785981)

[9.3 Appendix 3 – Bill of Materials 23](#_Toc448785982)

[9.4 Appendix 4 – Schematics 24](#_Toc448785983)

[9.5 Appendix 5 – Code 25](#_Toc448785984)

[9.6 Appendix 6 – Mathematical Model 41](#_Toc448785985)

[9.7 Appendix 7 – Machining 44](#_Toc448785986)

# Executive Summary

The Plaque Laser Engraver Device (PLED) is a system which engraves high quality images into wooden plaques. Our project can create quality decorations for homes and offices, the plaques could also be used for awards and recognition.

This report contains all information pertaining to the design and construction of a (PLED) system, as well as its capabilities. The introduction discusses the motivation behind the project and its purpose. The requirements and specifications sections contain specific functional and physical requirements of the PLED. The methods section explains the design steps and tasks involved in building the PLED system. The results section displays plaques that have been engraved and the progression of our plaques. The discussion section considers the plaques and the implications of our results. The conclusion discusses how well our team followed through with our plans and potential improvements or redesigns.

Additionally there is an appendix section, which provides our timeline, budget, and mathematical model. It also contains the design details for the construction of this system including the code, drawings for machined parts, bill of materials, and schematic.

After considering this document, the reader will be capable of not only understanding the design of the PLED, but recreating it.

# Introduction

The first step in designing the Plaque Laser Engraver Device (PLED) was specifying our desired results and selecting specific design requirements to yield said results.

In specifying our desired results, the first considerations we had to make were cost and quality. We were familiar with laser engraving systems and designs that used a lot of software and materials that were cost prohibitive, especially for students and hobbyists. We wanted to design something that would be significantly less cost prohibitive, but without sacrificing quality or capability. We determined to create a very high quality, high fidelity design at a price significantly lower than that for similar systems. In order to do so, we would have to write custom software for every portion of the system, from image processing to CNC control. We would also have to find high quality components at a low cost and find a way to minimize custom machining, as many hobbyists and students cannot afford to have parts machined professionally.

Our earliest decisions were in image resolution and in machine control. As it is very simple to design a black and white laser engraver, we determined to do something more difficult that could produce much higher quality reproductions of images. Initially we wanted to design a system capable of engraving 256 shade grayscale images, however upon further investigation, it proved difficult to verify that high of resolution, and we had concerns about calibrating so many different shades. We settled upon an 8 shade solution, which would provide much more detailed reproductions than a black and white system, but had fewer challenges than a 256 shade solution. Machine control was an easier decision to make. Industry standard CNC machines use g-code, which is a numerical control programming language capable of controlling all of the basic motion and burn systems for the PLED.

A more in depth look at the details of this design will be provided in section 5 of this document.

# Requirements

The specific requirements imposed upon the Plaque Laser Engraver Device (PLED) design follow.

## 3.1 Functional Requirements

### 3.1.1 PLED shall use a laser to burn grayscale images onto wooden plaques.

### 3.1.2 PLED shall operate like a CNC cutter to create facsimile images.

### 3.1.3 PLED laser shall turn off immediately upon system error.

### 3.1.4 PLED shall not function after the xy axis alignment is disturbed.

### 3.1.5 PLED shall use commercial laser driver to control laser diode.

### 3.1.6 PLED shall utilize a PC, microcontroller, and laser module.

### 3.1.7 PLED shall utilize G-Code for positioning.

### 3.1.8 PLED shall use no more than 7 Watts to drive diode and shall utilize a single 120 VAC power input.

## 3.2 Non-Functional Requirements

### 3.2.1 PLED shall create novelty plaques with wood burnt images up to 9”x12” in size.

### 3.2.2 PLED shall not require user input beyond an image and power to the system.

### 3.2.3 PLED shall operate independently after an image is provided and a plaque is clamped in place.

### 3.2.4 PLED shall meet laser industry/government standard laser safety specifications.

### 3.2.5 PLED shall be less than 3 cubic feet in size.

### 3.2.6 PLED shall weigh less than 30 pounds.

### 3.2.7 PLED shall be capable of surviving gentle motion, but no drop tests are required.

### 3.2.8 PLED structure shall be made of aluminum.

### 3.2.9 PLED shall be capable of operating in a temperature range of 40° - 110° F.

### 3.2.10 PLED shall operate in regions with less than 90% humidity.

### 3.2.11 PLED shall use a noise-free power source.

### 3.2.12 PLED shall cost commercially no more than $1000 (PC not included).

# Specifications

Detailed specifications for the Plaque Laser Engraver Device (PLED) are laid out in this section.

## 4.1 Performance Characteristics

### 4.1.1 PLED shall use a 445 nm laser at 2 watts power to keep engraving time under 45 minutes.

### 4.1.2 PLED shall burn images with a minimum resolution of 3 bits, 8 shades of grayscale.

### 4.1.3 PLED motion on a two dimensional axis shall be accurate to a minimum of 0.5 mm.

### 4.1.4 PLED laser shall turn off immediately upon system error.

### 4.1.5 PLED shall utilize a microcontroller for processing and control.

### 4.1.6 PLED shall utilize G-Code for position tracking and movement of the laser module.

### 4.1.7 PLED shall have a minimum resolution of 85 dpi (dots per inch).

### 4.1.8 PLED shall have a maximum image size of 1600 x 1600 pixels.

### 4.1.9 PLED shall utilize 5 V logic.

### 4.1.10 PLED shall utilize 5 VDC for the microcontroller.

### 4.1.11 PLED shall utilize 18 VDC, 3.5 A max for the motor driver

### 4.1.12 PLED shall utilize 5 VDC, 1.7 A max for the laser driver.

## 4.2 Physical characteristics

### 4.2.1 PLED shall be less than 3 cubic feet in size.

### 4.2.2 PLED shall weigh less than 30 pounds.

### 4.2.3 PLED structure shall be made of aluminum.

### 4.2.4 PLED shall not function after the xy axis alignment is disturbed.

### 4.2.5 PLED shall use no more than 7.65 Watts to drive diode and 120 VAC power input.

### 4.2.6 PLED shall meet laser industry/government standard laser safety specifications.

## 4.3 Environment

### 4.3.1 Natural Environments

4.3.1.1 PLED shall be capable of operating in a temperature range of 40°-110° F.

4.3.1.2 PLED shall operate in regions with less than 90% humidity.

### 4.3.2 Induced Environments

4.3.2.1 PLED shall be capable of surviving 3 G’s, but no drop tests are required.

4.3.2.2 PLED shall use a power source with no more than 2% total harmonic distortion.

## 4.4 Cleanliness

### 4.4.1 Electromagnetic interference (EMI)

4.4.1.1 The unit shall met the EMI requirements for class IV equipment as specified in MIL-STD-461.

### 4.4.2 Ventilation

4.4.2.1 If smoke particles exceed 500 ppm concentration a fan will start to vent fresh air

# Methods

A number of tasks, divided into different phases, must be accomplished in order for the Plaque Laser Engraver Device (PLED) project to be successful. A detailed look at the timeline for those tasks can be found in Appendix 2, while a detailed overview of each tasks and its specific execution follows after the block diagram in figure 1.



Figure 1. PLED System Block Diagram

**Phase 1: Initial documentation**

*Specifications document* – Determined program requirements and compiled them into a specifications document which drove the design of the PLED.

*Proposal* – A proposal was written to outline the design and details of the PLED project and how it was to be conducted.

*Select specific components to be used* – The PLED team used the specifications document and proposal to determine which components would be used for the PLED, then the specific manufacturers and models were identified.

*Preliminary Design Review slides* – In preparation for a preliminary design review, the last details of the design were determined and a presentation was prepared outlining those details and that design.

*Preliminary Design Review* – A preliminary design review was conducted with peers and engineering professionals to assure that the design was achievable and sufficient to yield the desired results. Feedback was taken and applied as the design matured.

**Phase 2: Build prototype**

*Procure components* – Components were procured in order to begin development of the PLED. A full list of those components and their descriptions can be found in Appendix 3.

*Configure serial communications between PC and microcontroller –* A serial communications line was configured and used to pass the g-codes to the interpreter housed in the microcontroller. This was accomplished by opening a UART serial communications port on the PC and on the microcontroller. Using a special USB serial cable, data could be passed via the UART protocol between the microcontroller and host PC. This code can be found under “Serial and Parser” in Appendix 5.

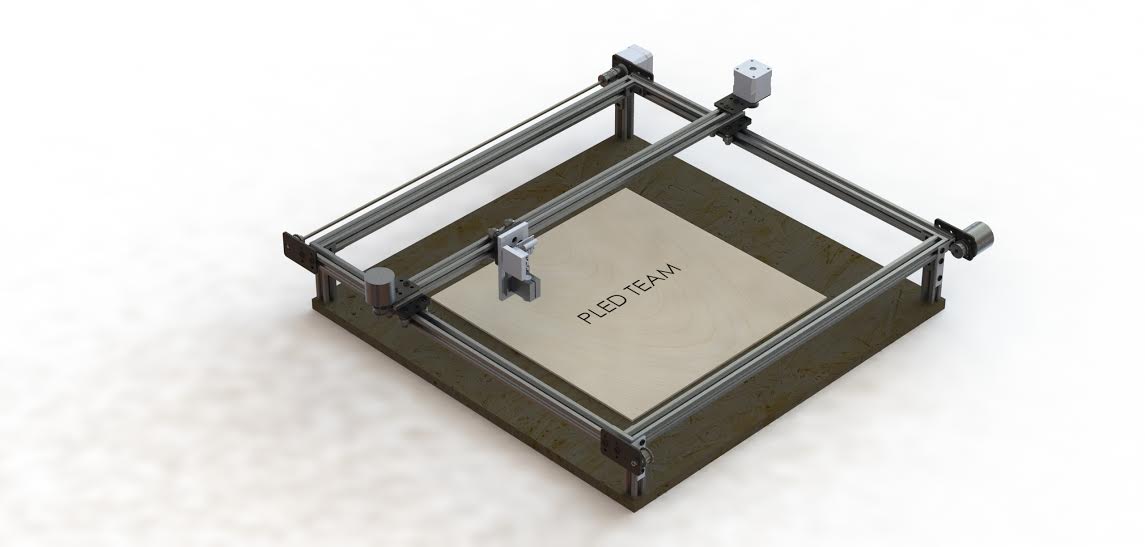
*Write software to ingest and process image and convert to g-codes* – Software was written to ingest the initial image, adjust dimensions according to board size, and convert the image to 8 shades of grayscale. At the same time, the image is turned into a series of g-codes which are later interpreted by the microcontroller to specify location and shade. Standard g-codes for position (G00 and G01) and dwell duration (G04) were used. In addition, codes specified for a milling machine were adjusted to meet our needs. The codes for spindle on and off (M04 and M05) were used to determine laser state (on or off). The code for spindle speed (S##) was used to set burn intensity (S00 is the darkest, S07 is the lightest). The milling machine code for termination (M02) was also used to fulfill the same purpose. The code for this process can be found under “Image Ingestion” in Appendix 5.

*Test components individually* – Each electrical component (motor drivers and motors, encoders, laser driver and laser) was tested individually. We characterized the functionality and performance metrics of these components, including motor resolution, speed, torque and step sizes, encoder pulses and laser threshold and intensity capabilities. Our motors are specified to 400 pulses per revolution, however they are capable of as small as 1/32 steps using microstepping. We decided to use 1/16 steps, meaning that there are 6400 microsteps per revolution. Our encoder is capable of 600 pulses per revolution, however it is a quadrature encoder, and can therefore provide 2400 pulses per revolution. The laser, once set at threshold, can have its intensity set using a PWM. We configured our laser to have a 480 pulse duty cycle, so dividing that 480 between the 7 darkened shades (plus white for the 8th shade) allows us to generate varying shades of grayscale.

*Configure communications from microcontroller to motor driver* – The motors are controlled using a pulse width modulation. By varying the microstep sizes and frequency of pulses, we set the motor resolution and speed. The code for doing this can be found under “Microcontroller” in Appendix 5.

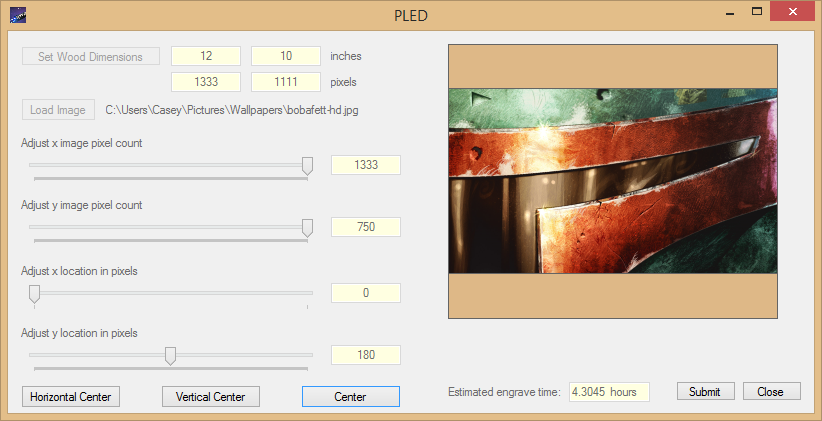
*Design mechanical system* – Our mechanical engineer designed a sturdy, reliable, and very structurally sound gantry for the PLED. The gantry can allow for images that are roughly 15x15 inches. It also prevents slippage on the motors and belts and assures adequate communication between the motors and encoders to allow for system feedback. An image of the working CAD model is included in figure 2 below.

*Implement parser for g-code files* – A g-code parser was designed on the host PC to parse and do a partial interpretation of g-codes on the host PC. This parser ingests the g-codes from the .gcode file and strips them down into a format easy to send via the UART serial protocol. These formatted codes are passed via the serial code mentioned above to the microcontroller. The code for this process can be found under “Serial and Parser” in Appendix 5.

  
Figure 2. CAD model of the PLED system.

*Write software GUI and interface* – A graphical user interface (GUI) was written in C# to accept user inputs and host all of the PC protocols in the PLED system. This code first determines plaque size, then allows the user to ingest an image. The user can then select the image size and location on the plaque. The software gives a time estimate for completion of engraving. When the user submits the information, the GUI calls the serial protocol. Code for the GUI can be found under “Graphical User Interface” in Appendix 5. An example of the GUI is displayed in figure 3 below.

Figure 2. Example of the PLED GUI.



*Procure mechanical system components* – After the mechanical structure was designed and determined adequate, the components for the system were purchased.

*Design and implement feedback system* – The feedback control system of the PLED uses rotary encoders to determine the actual location of the laser head. Comparing the actual location to the expected/desired location allows us to correct for errors of alignment. The code for this feedback can be found under “Microcontroller” in Appendix 5.

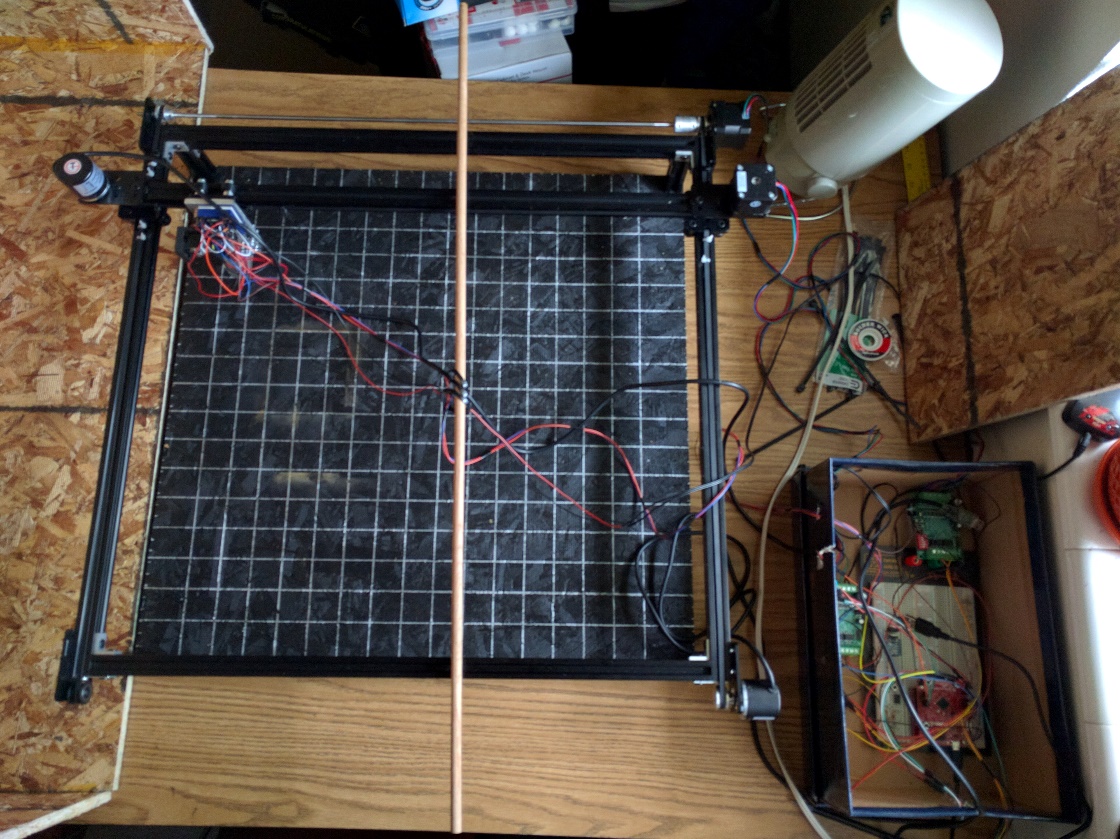
*Mathematically model system* – A mathematical model was developed for the PLED system. This model was used to calculate the maximum torque, speed, and load that our motors could bear. The full mathematical model can be found in Appendix 6.

*Implement and calibrate laser control system* – The 2 Watt laser in our system had to be calibrated and focused, and we had to specify a control system. As outlined in the section “Test components individually,” our laser system is controlled by a PWM with a 480 pulse duty cycle. This allows us to specify intensity, and therefore burn contrast. The code outlining the control of the laser can be found under “Microcontroller” in Appendix 5.

*Integrate g-code interpreter into microcontroller* – A g-code interpreter was integrated into the microcontroller. This code received the parsed codes via the UART serial protocol and interpreted them to call the functions to actually control the PLED. The code for this procedure can be found under “Microcontroller” in Appendix 5.

*Machine parts* – Despite our efforts to avoid professional machining, some of our components did have to be machined to accommodate our design. Most of this machining was to enlarge holes, counter sink / counter bore screw holes, or to cut down parts that were too large in size. Additionally, one part, the laser mount, was custom designed and machined for our application. All of the drawings for these modifications can be found in Appendix 7.

*Assemble system* – Once all components were procured and modified according to our design specifications, we assembled the system according to our design. The physical system can be seen in the photograph below in figure 4 below.

  
Figure 4. The PLED system

**Phase 3: Testing, final documentation, additional features**

*Write user’s manual –* A user’s manual has been written to outline how to configure and use the PLED. It explains how to install and launch the host software, as well as how to communicate between the host PC and the PLED system.

*Write final report* – Time was set aside to write this report, detailing our design, progress, and results.

*Testing and verification –* Once all was assembled, our system had to be tested and calibrated to assure that it met our specifications. This testing was also a period of adjustment, as we tried to assure that our images fit the criteria that we had outlined.

*Design poster –* A poster has been designed to advertise and outline the PLED, its capabilities, and how it works.

# Results

The Plaque Laser Engraver Device (PLED) was able to reproduce 8 shade grayscale images, as specified and desired. This was verified primarily by inspection. Figure 5 shows an 8 shade image used to verify the different shades of grayscale engraved by the PLED. Additionally, a series of engraved images are shown below in figures 6 - 11 to further show the successful results of the PLED.

  
Figure 5. An 8-shade image used to verify the shades. The white shade is between the others.

  
Figure 6. Hack USU logo engraved into a piece of poplar. Our first successful engraving. Contrast was lacking and shades are too dark.

  
Figure 7. Utah State logo engraved into poplar. Second engraving. Contrast is still inadequate and there are visible alignment issues.



Figure 8. An image of Boba-Fett from Star Wars engraved on poplar. This was our first engraving with improved contrast, however the alignment had issues.



Figure 9. An image of the Logan Temple engraved on poplar. Contrast and alignment are good, however we learned the ill effects of engraving against the wood grain.



Figure 10. USU Aggies logo on poplar. Good contrast and alignment. Successful engraving.

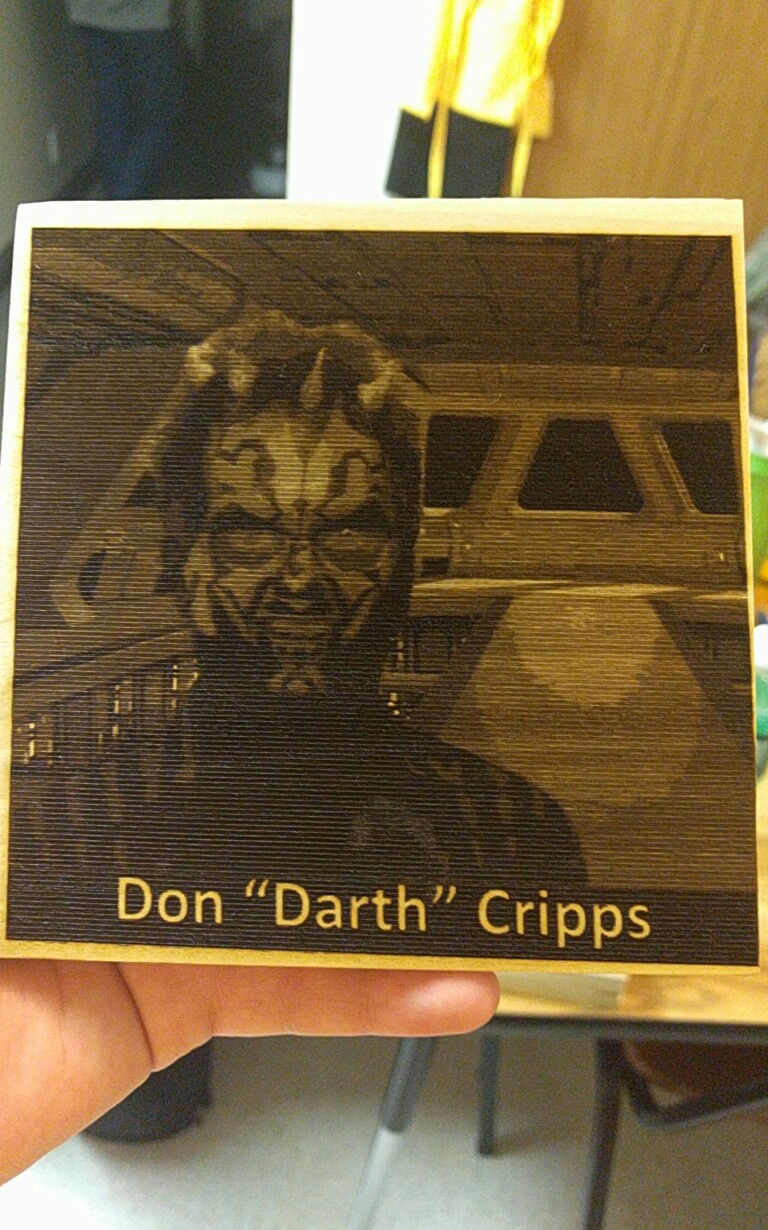


Figure 11. An engraving of Don “Darth” Cripps on the bridge of a star destroyer. Engraved on poplar, good alignment, contrast, and large in size (roughly 5.5 x 5.5 inches).

# Discussion

The Plaque Laser Engraver Device (PLED) has been successful in achieving our goals. We have written a very capable and user friendly software package that simplifies the process of turning an image into 8 shades of grayscale. We have designed an extremely high quality, efficient, and reliable gantry and motion system.

There are significant improvements that can still be made in our system. We could improve the resolution of our images. With a higher number of dots per inch (DPI) we would take longer to engrave, but our images would be clearer. We could adjust the system to produce an 8-bit, 256 shade system. Additionally, the images are well aligned. The only issue in any image is physical alignment of the board on the machine, which we are going to improve by gluing down measuring sticks.

Despite those possible improvements, our system met our initial design goals and we have laid a foundation that will make it very easy and very possible to make improvements. With sufficient time and effort, all of the weak points in our design, mentioned above, could easily be addressed. A stronger laser could be used, increased engraving time could be permitted. Pixel size definitions, and therefore number of motor steps per pixel could be decreased, allowing for a tighter image.

PLED has been a success, and as it is a system that we are passionate about, we fully intend to continue building and improving the system so that at some point, PLED will be the best laser engraver possible for the price.

# Conclusion

The Plaque Laser Engraver Device (PLED) has met our design goals. It has provided us with a design which can effectively engrave images according to our specifications.

Though we haven’t met our initial budgeting goal of $400, we have generated a design that is affordable for students and hobbyists compared to standard engravers on the market.

We have generated a software package that will either be open source, or available at a very low cost which can effectively process images and generate g-codes.

We have also gained very worthwhile experience in program management and organization, as well as in working on a multi-disciplinary team. We have learned to design a system efficiently and then build up that system to achieve our needs. We learned the importance of setting and obtaining funds for a realistic budget. Extra is always better than too little. We learned the value in giving time and resources to the design of a quality mechanical structure. We learned the importance of calibration and of writing readable code that can be easily debugged. It is also important to be willing to look at problems in different ways.

Most importantly for us, we have met our requirements to be able to fulfill our credits for the courses we are enrolled in, and required to graduate from Utah State University.

# Appendices

These appendices provide additional information in regards to the Plaque Laser Engraver Device (PLED).

## 9.1 Appendix 1 – Budget

The PLED budget is detailed below, with each subsystem budget specified in table 1.

|  |  |
| --- | --- |
| **PLED Budget** | |
| **Subsystem** | **Price** |
| Laser System | $150 |
| Motors System | $60 |
| Feedback System | $35 |
| Microcontroller | $15 |
| Mechanical Structure | $400 |
| Cabling | $35 |
| Safety | $35 |
| Testing Materials | $40 |
|  | |
| Total | $770 |

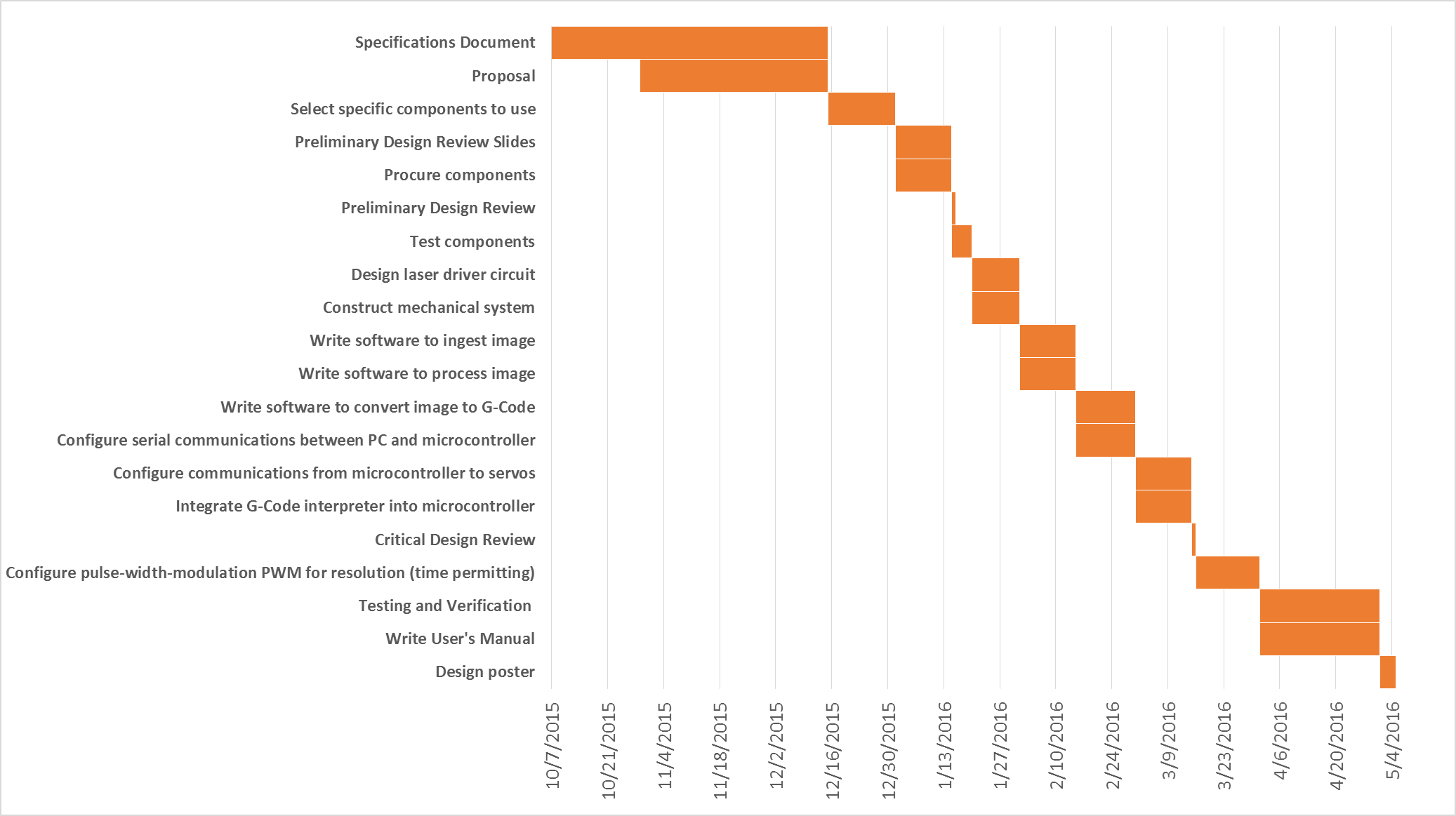
Table 1. PLED Budget

## 9.2 Appendix 2 – Timeline

For the duration of the PLED program, work was divided into three phases, described below in Table 2. Our originally proposed Gantt chart is in figure 12 and the Gantt chart as followed is in figure 13.

|  |  |  |  |
| --- | --- | --- | --- |
| **PLED Schedule** | | | |
| **Task** | **Start date** | **Due date** | **Duration** |
| **Phase 1 - Initial documentation** | | | |
| Specifications Document (Zach & Casey) | 12/1/2015 | 12/15/2015 | 14 |
| Proposal (Zach & Casey) | 12/1/2015 | 12/15/2015 | 14 |
| Select specific components to use (Team) | 12/15/2015 | 1/25/2016 | 41 |
| Preliminary Design Review Slides (Zach & Casey) | 1/1/2016 | 1/25/2016 | 24 |
| Preliminary Design Review (Team) | 1/27/2016 | 1/28/2016 | 1 |
| **Phase 2 - Build prototype** | | | |
| Procure components (Zach & Casey) | 1/1/2016 | 1/30/2016 | 29 |
| Configure serial communications PC / microcontroller (Zach & Justin) | 1/15/2016 | 2/8/2016 | 24 |
| Write software to ingest and process image and convert to g-codes (Casey) | 1/25/2016 | 2/8/2016 | 14 |
| Test components individually (Team) | 1/15/2016 | 2/10/2016 | 26 |
| Configure communications from microcontroller to motor driver (Zach) | 2/9/2016 | 2/20/2016 | 11 |
| Design mechanical system (Tate) | 1/20/2016 | 2/29/2016 | 40 |
| Impliment parser for g-code files (Justin) | 2/20/2016 | 3/15/2016 | 24 |
| Write software GUI and interface (Casey) | 2/21/2016 | 3/15/2016 | 23 |
| Procure mechanical system components (Tate & Casey) | 3/4/2016 | 3/15/2016 | 11 |
| Design and impliment feedback system (Tate, Zach, Casey) | 3/5/2016 | 3/15/2016 | 10 |
| Mathematically model system (Tate, Zach, Casey) | 2/20/2016 | 3/25/2016 | 34 |
| Impliment and calibrate laser control system (Zach) | 2/21/2016 | 3/25/2016 | 33 |
| Integrate G-Code interpreter into microcontroller (Justin) | 3/15/2016 | 3/25/2016 | 10 |
| Machine Parts (Tate) | 3/15/2016 | 3/31/2016 | 16 |
| Assemble system (Team) | 4/1/2016 | 4/8/2016 | 7 |
| **Phase 3 - Testing, final documentation, additional features** | | | |
| Write User's Manual (Zach & Casey) | 4/1/2016 | 4/29/2016 | 28 |
| Write Final Report (Zach & Casey) | 4/1/2016 | 4/29/2016 | 28 |
| Testing and Verification (Team) | 4/8/2016 | 4/29/2016 | 21 |
| Design poster (Zach & Casey) | 4/15/2016 | 4/29/2016 | 14 |

Table 2. PLED phase schedule and tasks.

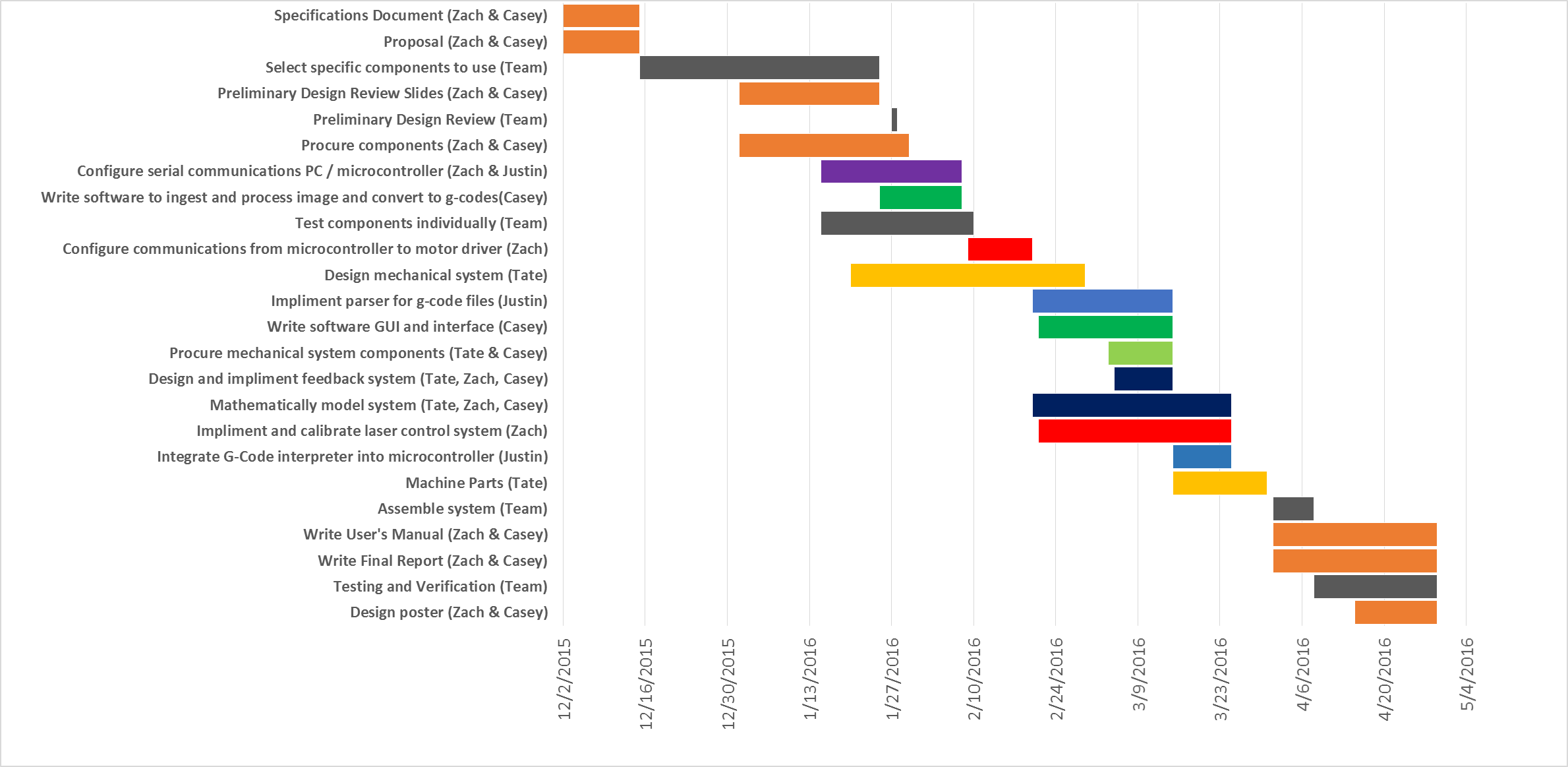


**Phase 2**

**Phase 1**

**Phase 3**

Figure 12. Proposed PLED Gantt Chart.



**Phase 3**

**Phase 2**

**Phase 1**

Figure 13. Actual PLED Gantt Chart.

## 9.3 Appendix 3 – Bill of Materials

This section details each individual component used to build a full PLED system. Table 3 contains the components, a description of them, the quantity used in the design, and its cost.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PLED Bill of Materials** | | | | |
| **P/N** | **Description** | **Vendor** | **$ / Ea** | **QTY** |
| TTL-232R-3V3 | USB UART Adapter | Amazon.com | $ 19.95 | 1 |
| DC5-24V | 600P/R Rotary Encoder | Amazon.com | $ 17.45 | 2 |
| TB6560 | Motor Driver | Amazon.com | $ 14.99 | 2 |
| Nema 17 | Stepper Motors | Amazon.com | $ 13.99 | 2 |
| O1 rod | 36", 5mm Rod | Amazon.com | $ 10.78 | 1 |
| Gdstime 3007 | Cooling Fan | Amazon.com | $ 8.59 | 1 |
| WYPH IIC I2C | Level Converter 3.3 to TTL | Amazon.com | $ 5.87 | 1 |
| SK12 | Laser Heat Sink | Amazon.com | $ 5.15 | 1 |
| Coupler | 5x5mm Coupler | Amazon.com | $ 5.09 | 1 |
| TO-220/TO-202 | Heat Sink | Amazon.com | $ 5.00 | 1 |
| VIO 1.5G | Thermal Grease | Amazon.com | $ 4.95 | 1 |
| 445 M140 Module | Laser Module | DTR's Laser Shop | $ 72.00 | 1 |
| 381531274978 | LASORB | eBay.com | $ 11.00 | 1 |
| Fasteners | Fasteners/washers/bolts | Fastenal | $ 10.67 | 1 |
| Spacers | Washers and spacers | Home Depot | $ 7.35 | 1 |
| Corner Brace | L-Beam for Frame (4 pk) | Home Depot | $ 2.84 | 1 |
| m3-wash | Motor/Encoder Spacers (6 pk) | Home Depot | $ 0.48 | 6 |
| m3-0.5x10mm | Motor Fasteners (3 pk) | Home Depot | $ 0.48 | 3 |
| FlexMod P3 | Laser Driver Heat Sink | Innolasers | $ 35.99 | 1 |
| Yard Stick | Straight edge for grid | Lowes | $ 3.18 | 1 |
| 1185 | Mini V Gantry Set | OpenBuilds.org | $ 28.95 | 3 |
| 230-LP | V-Slot Linear Rail (1500 mm) | OpenBuilds.org | $ 16.50 | 2 |
| 170-LP | V-Slot Linear Rail (1000 mm) | OpenBuilds.org | $ 11.00 | 1 |
| 200 | GT2 30-Tooth Pulley | OpenBuilds.org | $ 6.95 | 5 |
| 570 | Idler pulley Plate | OpenBuilds.org | $ 6.95 | 4 |
| 575 | Motor mount Plate | OpenBuilds.org | $ 6.95 | 2 |
| 550 | Smooth Idler Pulley | OpenBuilds.org | $ 5.45 | 1 |
| 50 | Tee Nuts (25 Pack) | OpenBuilds.org | $ 4.95 | 1 |
| 470 | GT2 Timing Belt | OpenBuilds.org | $ 2.50 | 14 |
| 545 | L Bracket | OpenBuilds.org | $ 1.00 | 8 |
| 30 | Ball Bearing | OpenBuilds.org | $ 1.00 | 1 |
| 60 | Double Tee Nut | OpenBuilds.org | $ 0.85 | 10 |
| 730 | Belt Clamp | OpenBuilds.org | $ 0.60 | 6 |
| 115 | Low Profile M5 Screws | OpenBuilds.org | $ 0.15 | 36 |

Table 3. PLED BOM

## 9.4 Appendix 4 – Schematics

Schematics for the Plaque Laser Engraver Device (PLED) appear in this appendix in figure 14.

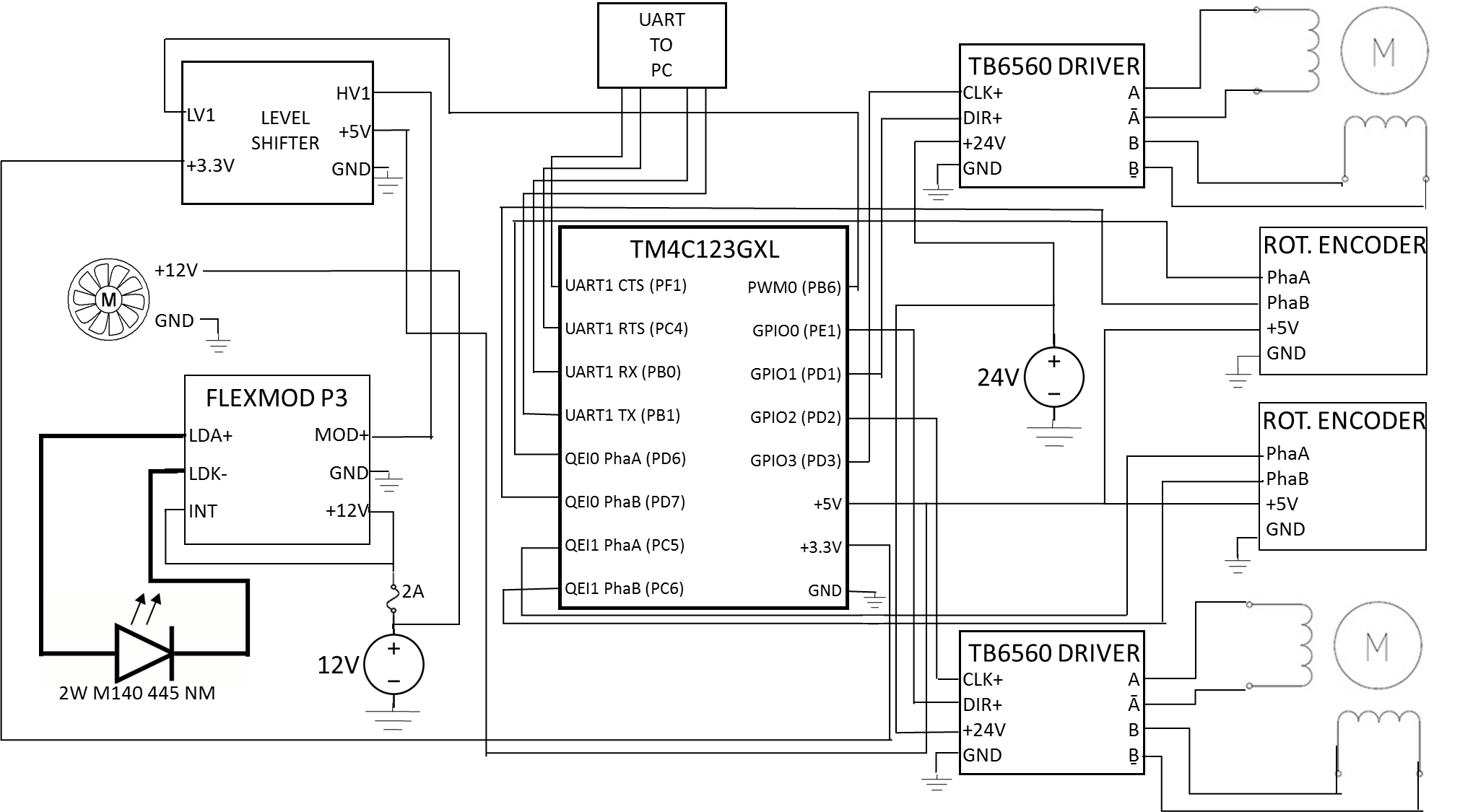


Figure 14. PLED schematic.

## 9.5 Appendix 5 – Code

***Serial and Parser***

***Microcontroller***

***Image Ingestion***

**“IngestImage.cs”**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using System.IO;

namespace IngestImage

{

class Program

{

public void Ingest(int xImageDim, int yImageDim, int xLocation, int yLocation, String inPath, String outPath)

{

//define image and file input

Image inputImage = Image.FromFile(inPath);

//c is original image in bitmap

Bitmap c = new Bitmap(inputImage);

//d is new, modified image in bitmap

Bitmap d = new Bitmap(inputImage, new Size(xImageDim, yImageDim));

/\*G-Code interpretations:

G00 X--.-- Y--.-- Z--.-- Rapid Positioning (units in mm, redefine?)

G01 X--.-- Y--.-- Z--.-- Linear Interpolation (units in mm, redefine?)

G04 P--.-- Dwell (units in seconds, redefine?)

M04 Spindle on (Laser on?)

M05 Spindle off (Laser off?)

M02 End of program

S-- Set speed of spindle (Laser intensity?)

\*/

//define which x and y pixel we are at

int row = 0;

int column = -1;

//define and create file for G codes

string path = outPath + ".gcode";

if (!File.Exists(path))

{

using (StreamWriter sw = File.CreateText(path))

{

//define image size

sw.WriteLine("size " + d.Width + "," + d.Height);

//write go to starting point of engraving and dwell G-Codes to file

sw.WriteLine("G00 X" + xLocation + " Y" + yLocation);

sw.WriteLine("G04 P0.500");

//loop pulls in pixels, converts to grayscale. i = image width (rows), x = image height (columns)

for (int i = 0; i < d.Height; i++)

{

//increase which row (y value) we are at

if (row != 0 || column != -1)

{

row++;

}

//makes sure at right pixels

if ((row % 2 == 0 && row != 0 && column != 0) || column == d.Width - 1)

{

column++;

}

if((row %2 != 0 && row != 0 && column != d.Width) || column == 0)

{

column--;

}

for (int x = 0; x < d.Width; x++)

{

//increase/decreases which column (x value) we are at

if (row % 2 != 0)

{

column--;

}

else

{

column++;

}

//select pixel

Color oc = d.GetPixel(column, row);

//define to grayscale

int grayScale = (int)((oc.R \* 0.3) + (oc.G \* 0.59) + (oc.B \* 0.11));

//truncate to 8 shades

grayScale = grayScale / 32;

//move back to being in increments of 32 - creates better variance in shades

grayScale = grayScale \* 32;

//set new pixel to grayscale

Color nc = Color.FromArgb(oc.A, grayScale, grayScale, grayScale);

//write pixel to specified position

d.SetPixel(column, row, nc);

//G-Code to move to next pixel

sw.WriteLine("G01 X" + (column + xLocation) + " Y" + (row + yLocation));

//G-Code to set intensity of laser

sw.WriteLine("S" + grayScale / 32);

//G-Code to turn on laser

sw.WriteLine("M04");

//G-Code for dwell

sw.WriteLine("G04 P0.040");

//G-Code to turn off laser

sw.WriteLine("M05");

}

}

//End G-Code File

sw.WriteLine("M02");

sw.Close();

}

}

//save new image

d.Save(outPath + ".jpg");

//dispose of bitmap variables

c.Dispose();

d.Dispose();

//end program

return;

}

}

}

***Graphical User Interface***

**“Program.cs”**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace PLED\_GUI

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

Application.Exit();

}

}

}

**“Form1.cs”**

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 IngestImage;

using System.IO;

namespace PLED\_GUI

{

public partial class Form1 : Form

{

//defines return values to be used for wood dimension window

public string ReturnValue1 { get; set; }

public string ReturnValue2 { get; set; }

//defines strings, ints, and doubles to be used

string xDimension, yDimension, imgPathString, filePath, outFilePath;

int xDimPix, yDimPix, xLocPix, yLocPix, xImgSize, yImgSize;

double xyratio, xywoodratio, xdoubletmp, ydoubletmp, tmpwoodsize, time;

public Form1()

{

//Initializes application

InitializeComponent();

//initializes file dialog settings

openFileDialog1.Title = "Load PLED Image";

openFileDialog1.InitialDirectory = @"C:";

openFileDialog1.Filter = "Image Files (\*.jpg, \*.jpeg, \*.png, \*.gif, \*.bmp)|\*.jpg; \*.jpeg; \*.png; \*.gif; \*.bmp";

openFileDialog1.FilterIndex = 1;

openFileDialog1.RestoreDirectory = true;

//initialize save dialog settings

saveFileDialog1.Title = "Select output File";

saveFileDialog1.InitialDirectory = filePath;

saveFileDialog1.Filter = "Output Filename (no extension)|";

saveFileDialog1.FilterIndex = 1;

saveFileDialog1.RestoreDirectory = true;

//start image load button disabled

imgLoad.Enabled = false;

}

//handler for clicking on SelectWoodDimensions button

private void SelectWoodDimensions\_Click(object sender, EventArgs e)

{

//creates instance of window

using (var getWoodDimensions = new woodDimensions())

{

//opens window

var result = getWoodDimensions.ShowDialog();

//set invalid or closed case

if (result == DialogResult.Abort)

{

if(getWoodDimensions.ReturnValue1 != null)

{

x\_woodDimension.Text = getWoodDimensions.ReturnValue1;

y\_woodDimension.Text = getWoodDimensions.ReturnValue2;

}

}

if (result == DialogResult.OK)

{

//saves dimensions in inches

xDimension = getWoodDimensions.ReturnValue1;

yDimension = getWoodDimensions.ReturnValue2;

//displays dimensions in inches

x\_woodDimension.Text = xDimension;

y\_woodDimension.Text = yDimension;

//converts, saves and displays dimensions in pixel count

xDimPix = Convert.ToUInt16(Convert.ToDouble(xDimension) / 0.009);

yDimPix = Convert.ToUInt16(Convert.ToDouble(yDimension) / 0.009);

xWoodDimPixels.Text = Convert.ToString(xDimPix);

yWoodDimPixels.Text = Convert.ToString(yDimPix);

//sets max slider value based on dimension in pixels

xSlider.Maximum = xDimPix - xImgSize;

ySlider.Maximum = yDimPix - yImgSize;

//set wood plaque ratio and dimensions

xywoodratio = Convert.ToDouble(xDimPix) / Convert.ToDouble(yDimPix);

if (xDimPix > yDimPix)

{

PlaqueSize.Width = 330;

tmpwoodsize = 330 \* (1 / xywoodratio);

PlaqueSize.Height = Convert.ToUInt16(tmpwoodsize);

}

else if (xDimPix < yDimPix)

{

PlaqueSize.Height = 330;

tmpwoodsize = 330 \* xywoodratio;

PlaqueSize.Width = Convert.ToUInt16(tmpwoodsize);

}

else

{

PlaqueSize.Width = PlaqueSize.Height = 330;

}

if(xImgSize != 0 && yImgSize !=0)

{

while (xImgSize > xDimPix || yImgSize > yDimPix)

{

if (xImgSize > yImgSize)

{

//adjust image size (pixel count) according to plaque size if image is larger

if (xImgSize > xDimPix)

{

//sets y image dimension to y wood dimension

xImgSize = xDimPix;

//sets x dimension according to conversion above

ydoubletmp = xImgSize \* (1 / xyratio);

yImgSize = Convert.ToUInt16(ydoubletmp);

}

}

else if (yImgSize > xImgSize)

{

//adjust image size (pixel count) according to plaque size if image is larger

if (yImgSize > yDimPix)

{

//sets x image dimension to x wood dimension

yImgSize = yDimPix;

//sets y dimension according to conversion above

xdoubletmp = yImgSize \* xyratio;

xImgSize = Convert.ToUInt16(xdoubletmp);

}

}

}

//set max slider location positions

xSlider.Maximum = xDimPix - xImgSize;

ySlider.Maximum = yDimPix - yImgSize;

//set max slider sizes

ximgslider.Maximum = xImgSize;

yimgslider.Maximum = yImgSize;

//set default position and size values

xSlider.Value = 0;

ySlider.Value = 0;

ximgslider.Value = xImgSize;

yimgslider.Value = yImgSize;

ximgsizebox.Text = Convert.ToString(xImgSize);

yimgsizebox.Text = Convert.ToString(yImgSize);

xLocBox.Text = "0";

yLocBox.Text = "0";

//set image size

xdoubletmp = (xImgSize \* PlaqueSize.Width) / xDimPix;

ydoubletmp = xdoubletmp \* (1 / xyratio);

imgBox.Width = Convert.ToUInt16(xdoubletmp);

imgBox.Height = Convert.ToUInt16(ydoubletmp);

}

//disable button

SelectWoodDimensions.Enabled = false;

imgLoad.Enabled = true;

}

}

}

private void openFileDialog1\_FileOk(object sender, CancelEventArgs e)

{

}

//handler for imgLoad button

private void imgLoad\_Click(object sender, EventArgs e)

{

//opens openFileDialog1 form

DialogResult result = openFileDialog1.ShowDialog();

if (result == DialogResult.OK)

{

//obtains file path

imgPathString = openFileDialog1.FileName;

filePath = Path.GetDirectoryName(imgPathString);

imgPath.Text = imgPathString;

//loads image

Image inputImage = Image.FromFile(imgPathString);

//c is the image in bitmap

Bitmap c = new Bitmap(inputImage);

//sets image size according to image

xImgSize = c.Width;

yImgSize = c.Height;

//sets xyratio

xyratio = Convert.ToDouble(xImgSize) / Convert.ToDouble(yImgSize);

while (xImgSize > xDimPix || yImgSize > yDimPix)

{

if (xImgSize == yImgSize)

{

//adjust image size (pixel count) to fit plaque

xImgSize = xDimPix;

yImgSize = yDimPix;

}

else if (xImgSize > yImgSize)

{

//adjust image size (pixel count) according to plaque size if image is larger

if (xImgSize > xDimPix)

{

//sets y image dimension to y wood dimension

xImgSize = xDimPix;

//sets x dimension according to conversion above

ydoubletmp = xImgSize \* (1 / xyratio);

yImgSize = Convert.ToUInt16(ydoubletmp);

}

}

else if (yImgSize > xImgSize)

{

//adjust image size (pixel count) according to plaque size if image is larger

if (yImgSize > yDimPix)

{

//sets x image dimension to x wood dimension

yImgSize = yDimPix;

//sets y dimension according to conversion above

xdoubletmp = yImgSize \* xyratio;

xImgSize = Convert.ToUInt16(xdoubletmp);

}

}

}

//set max slider location positions

xSlider.Maximum = xDimPix - xImgSize;

ySlider.Maximum = yDimPix - yImgSize;

//set max slider sizes

ximgslider.Maximum = xImgSize;

yimgslider.Maximum = yImgSize;

//set default position and size values

xSlider.Value = 0;

ySlider.Value = 0;

ximgslider.Value = xImgSize;

yimgslider.Value = yImgSize;

ximgsizebox.Text = Convert.ToString(xImgSize);

yimgsizebox.Text = Convert.ToString(yImgSize);

xLocBox.Text = "0";

yLocBox.Text = "0";

//set image size

xdoubletmp = (xImgSize \* PlaqueSize.Width) / xDimPix;

ydoubletmp = xdoubletmp \* (1 / xyratio);

imgBox.Width = Convert.ToUInt16(xdoubletmp);

imgBox.Height = Convert.ToUInt16(ydoubletmp);

}

imgBox.ImageLocation = imgPathString;

imgLoad.Enabled = false;

//Estimates engrave time for image

time = (((Convert.ToDouble(xImgSize) \* Convert.ToDouble(yImgSize) \* (0.0155)) + (0.0005 \* (Convert.ToDouble(xLocPix) + Convert.ToDouble(yLocPix)))) / 3600);

EngTime.Text = Convert.ToString(time);

}

//x slider handler

private void xSlider\_Scroll(object sender, EventArgs e)

{

//allows user to adjust location to place image on x axis

xLocPix = xSlider.Value;

//displays pixel location

xLocBox.Text = Convert.ToString(xLocPix);

//moves image block representation on x axis

xdoubletmp = ((Convert.ToDouble(xLocPix) \* (PlaqueSize.Width - Convert.ToDouble(imgBox.Width))) / Convert.ToDouble(xSlider.Maximum));

imgBox.Location = new Point((440 + Convert.ToUInt16(xdoubletmp)), imgBox.Location.Y);

//Estimates engrave time for image

time = (((Convert.ToDouble(xImgSize) \* Convert.ToDouble(yImgSize) \* (0.0155)) + (0.0005 \* (Convert.ToDouble(xLocPix) + Convert.ToDouble(yLocPix)))) / 3600);

EngTime.Text = Convert.ToString(time);

}

//y slider handler

private void ySlider\_Scroll(object sender, EventArgs e)

{

//allows user to adjust location to place image on y axis

yLocPix = ySlider.Value;

//displays pixel location

yLocBox.Text = Convert.ToString(yLocPix);

//moves image block representation on y axis

ydoubletmp = ((Convert.ToDouble(yLocPix) \* (PlaqueSize.Height - Convert.ToDouble(imgBox.Height))) / Convert.ToDouble(ySlider.Maximum));

imgBox.Location = new Point(imgBox.Location.X, (13 + Convert.ToUInt16(ydoubletmp)));

//Estimates engrave time for image

time = (((Convert.ToDouble(xImgSize) \* Convert.ToDouble(yImgSize) \* (0.0155)) + (0.0005 \* (Convert.ToDouble(xLocPix) + Convert.ToDouble(yLocPix)))) / 3600);

EngTime.Text = Convert.ToString(time);

}

//centers image

private void cent\_Click(object sender, EventArgs e)

{

//centers on actual image

xLocPix = (xDimPix - xImgSize) / 2;

yLocPix = (yDimPix - yImgSize) / 2;

xSlider.Value = xLocPix;

ySlider.Value = yLocPix;

xLocBox.Text = Convert.ToString(xLocPix);

yLocBox.Text = Convert.ToString(yLocPix);

//centers on diagram

if(xDimPix != xImgSize)

{

xdoubletmp = ((Convert.ToDouble(xLocPix) \* (PlaqueSize.Width - Convert.ToDouble(imgBox.Width))) / Convert.ToDouble(xSlider.Maximum));

imgBox.Location = new Point((440 + Convert.ToUInt16(xdoubletmp)), imgBox.Location.Y);

}

if(yDimPix != yImgSize)

{

ydoubletmp = ((Convert.ToDouble(yLocPix) \* (PlaqueSize.Height - Convert.ToDouble(imgBox.Height))) / Convert.ToDouble(ySlider.Maximum));

imgBox.Location = new Point(imgBox.Location.X, (13 + Convert.ToUInt16(ydoubletmp)));

}

//Estimates engrave time for image

time = (((Convert.ToDouble(xImgSize) \* Convert.ToDouble(yImgSize) \* (0.0155)) + (0.0005 \* (Convert.ToDouble(xLocPix) + Convert.ToDouble(yLocPix)))) / 3600);

EngTime.Text = Convert.ToString(time);

}

//vertically centers image

private void vertCent\_Click(object sender, EventArgs e)

{

//vertically centers actual image

yLocPix = (yDimPix - yImgSize) / 2;

ySlider.Value = yLocPix;

yLocBox.Text = Convert.ToString(yLocPix);

//vertically centers diagram

if (yDimPix != yImgSize)

{

ydoubletmp = ((Convert.ToDouble(yLocPix) \* (PlaqueSize.Height - Convert.ToDouble(imgBox.Height))) / Convert.ToDouble(ySlider.Maximum));

imgBox.Location = new Point(imgBox.Location.X, (13 + Convert.ToUInt16(ydoubletmp)));

}

//Estimates engrave time for image

time = (((Convert.ToDouble(xImgSize) \* Convert.ToDouble(yImgSize) \* (0.0155)) + (0.0005 \* (Convert.ToDouble(xLocPix) + Convert.ToDouble(yLocPix)))) / 3600);

EngTime.Text = Convert.ToString(time);

}

//horizontally centers image

private void horzCent\_Click(object sender, EventArgs e)

{

//horizontally centers actual image

xLocPix = (xDimPix - xImgSize) / 2;

xSlider.Value = xLocPix;

xLocBox.Text = Convert.ToString(xLocPix);

//horizontally centers diagram

if(xDimPix != xImgSize)

{

xdoubletmp = ((Convert.ToDouble(xLocPix) \* (PlaqueSize.Width - Convert.ToDouble(imgBox.Width))) / Convert.ToDouble(xSlider.Maximum));

imgBox.Location = new Point((440 + Convert.ToUInt16(xdoubletmp)), imgBox.Location.Y);

}

//Estimates engrave time for image

time = (((Convert.ToDouble(xImgSize) \* Convert.ToDouble(yImgSize) \* (0.0155)) + (0.0005 \* (Convert.ToDouble(xLocPix) + Convert.ToDouble(yLocPix)))) / 3600);

EngTime.Text = Convert.ToString(time);

}

//handler for ximgslider slider

private void ximgslider\_Scroll(object sender, EventArgs e)

{

//allows user to adjust image size on x axis

xImgSize = ximgslider.Value;

//displays pixel count

ximgsizebox.Text = Convert.ToString(xImgSize);

//sets y size according to conversion above and updates slider

ydoubletmp = xImgSize \* (1 / xyratio);

yImgSize = Convert.ToUInt16(ydoubletmp);

yimgsizebox.Text = Convert.ToString(yImgSize);

yimgslider.Value = yImgSize;

//set max slider location positions

xSlider.Maximum = xDimPix - xImgSize;

ySlider.Maximum = yDimPix - yImgSize;

//set default slider location

xSlider.Value = 0;

ySlider.Value = 0;

xLocPix = 0;

yLocPix = 0;

xLocBox.Text = "0";

yLocBox.Text = "0";

imgBox.Location = new Point(440, 13);

//set image size

xdoubletmp = (xImgSize \* PlaqueSize.Width) / xDimPix;

ydoubletmp = xdoubletmp \* (1 / xyratio);

imgBox.Width = Convert.ToUInt16(xdoubletmp);

imgBox.Height = Convert.ToUInt16(ydoubletmp);

//Estimates engrave time for image

time = (((Convert.ToDouble(xImgSize) \* Convert.ToDouble(yImgSize) \* (0.0155)) + (0.0005 \* (Convert.ToDouble(xLocPix) + Convert.ToDouble(yLocPix)))) / 3600);

EngTime.Text = Convert.ToString(time);

}

//handler for yimgslider slider

private void yimgslider\_Scroll(object sender, EventArgs e)

{

//allows user to adjust image size on x axis

yImgSize = yimgslider.Value;

//displays pixel count

yimgsizebox.Text = Convert.ToString(yImgSize);

//sets y size according to conversion above and updates slider

xdoubletmp = yImgSize \* xyratio;

xImgSize = Convert.ToUInt16(xdoubletmp);

ximgsizebox.Text = Convert.ToString(xImgSize);

ximgslider.Value = xImgSize;

//set max slider location positions

xSlider.Maximum = xDimPix - xImgSize;

ySlider.Maximum = yDimPix - yImgSize;

//set default slider location

xSlider.Value = 0;

ySlider.Value = 0;

xLocPix = 0;

yLocPix = 0;

xLocBox.Text = "0";

yLocBox.Text = "0";

imgBox.Location = new Point(440, 13);

//set image size

xdoubletmp = (xImgSize \* PlaqueSize.Width) / xDimPix;

ydoubletmp = xdoubletmp \* (1 / xyratio);

imgBox.Width = Convert.ToUInt16(xdoubletmp);

imgBox.Height = Convert.ToUInt16(ydoubletmp);

//Estimates engrave time for image

time = (((Convert.ToDouble(xImgSize) \* Convert.ToDouble(yImgSize) \* (0.0155)) + (0.0005 \* (Convert.ToDouble(xLocPix) + Convert.ToDouble(yLocPix)))) / 3600);

EngTime.Text = Convert.ToString(time);

}

private void submitPLED\_Click(object sender, EventArgs e)

{

DialogResult result = saveFileDialog1.ShowDialog();

if(result == DialogResult.OK)

{

//select save path

outFilePath = saveFileDialog1.FileName;

//pass everything to image ingestion backend

IngestImage.Program imageIngest = new IngestImage.Program();

imageIngest.Ingest(xImgSize, yImgSize, xLocPix, yLocPix, imgPathString, outFilePath);

//create outpath file

StreamWriter sw = File.CreateText(@"C:\PLED\PLEDpath.txt");

sw.Write(outFilePath + ".gcode");

sw.Close();

//tell user job ready to begin begins the engraving process

var jobComplete = new complete();

jobComplete.ShowDialog();

var eng = new engraving();

Application.Exit();

}

}

//handler for endPLED button

private void endPLED\_Click\_1(object sender, EventArgs e)

{

//closes application

Application.Exit();

}

}

}

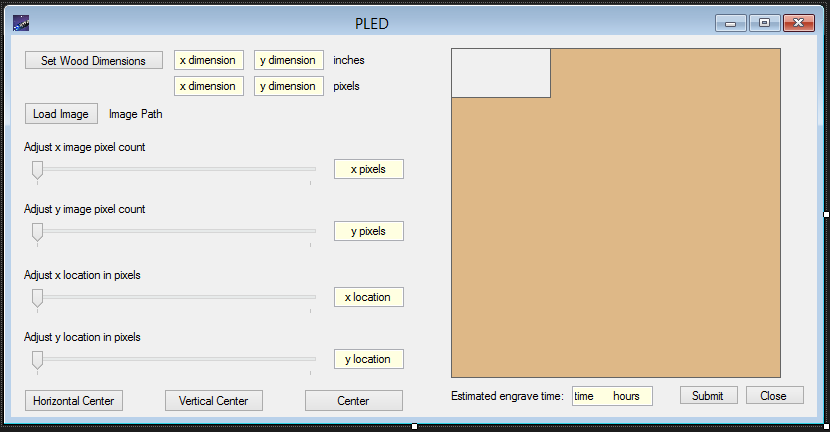


Figure 15. Form1.Designer.cs

**“complete.cs”**

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;

namespace PLED\_GUI

{

public partial class complete : Form

{

public complete()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

this.DialogResult = DialogResult.OK;

this.Close();

}

}

}

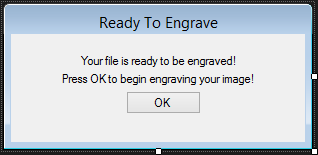


Figure 16. complete.Designer.cs

**“woodDimensions.cs”**

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;

namespace PLED\_GUI

{

public partial class woodDimensions : Form

{

public string ReturnValue1 { get; set; }

public string ReturnValue2 { get; set; }

public woodDimensions()

{

InitializeComponent();

}

private void setDimension\_Click(object sender, EventArgs e)

{

if (x\_woodDimension.Text == "x dimension (inches)" || y\_woodDimension.Text == "x dimension (inches)"

|| Convert.ToDouble(x\_woodDimension.Text) < 0 || Convert.ToDouble(x\_woodDimension.Text) > 14.4

|| Convert.ToDouble(y\_woodDimension.Text) < 0 || Convert.ToDouble(y\_woodDimension.Text) > 14.4)

{

this.ReturnValue1 = "Invalid Input";

this.ReturnValue2 = "Invalid Input";

this.DialogResult = DialogResult.Abort;

this.Close();

}

else

{

this.ReturnValue1 = x\_woodDimension.Text;

this.ReturnValue2 = y\_woodDimension.Text;

this.DialogResult = DialogResult.OK;

this.Close();

}

}

private void closeDim\_Click\_1(object sender, EventArgs e)

{

this.DialogResult = DialogResult.Abort;

this.Close();

}

}

}

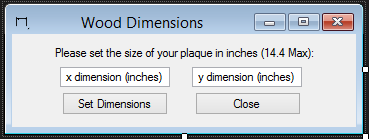


Figure 17. woodDimensions.Designer.cs

**“engraving.cs”**

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;

namespace PLED\_GUI

{

public partial class engraving : Form

{

public engraving()

{

InitializeComponent();

this.Show();

System.Diagnostics.Process.Start("C:/PLED/src/SerialToPC.exe");

}

}

}

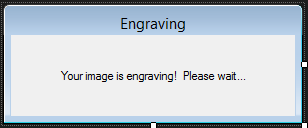


Figure 18. engraving.Designer.cs

## 9.6 Appendix 6 – Mathematical Model

The mathematical model for the Plaque Laser Engraver Device (PLED) is below. It details the torque requirements of the system, as well as its maximum load and velocity.

Using the example on page 720 (Mechatronics: An Integrated Approach), we can apply Newton’s second law to the torques in the system, resulting in the following equation of motion:

We believe that our motor is a PM (permanent magnet) stepper motor. According to page 722,

Using the torque of the motors and the radius of the sprocket to be used in our system, we can approximate the force that can therefore be produced, and the mass that can be moved.

Using specific information relative to our system, including our motor driver, microcontroller, and the specs of our components, we can determine the unloaded maximum linear velocity of our system.

v represents our maximum theoretical unloaded motor velocity of our system. Our loaded motors will move more slowly, however even if the speed is reduced by 50%, we will be moving very quickly. Additionally, by testing, we know that moving a single step takes 250 μs

Now we will consider the loaded velocity using a simplified model:

Figure 19. Diagram of mathematical model of motors

From solid works model:

Assuming sliding AKA bad fiction of mass we will use (Teflon on steel) value (0.04)

This seems fast. We will not work at this speed. We will go a lot slower.

## 9.7 Appendix 7 – Machining