LaSEEr

Authors: Enes Palaz, Eliana Cohen, Jake Zimmer: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—This project creates an open source laser show system capable of transforming live video feed into a visual laser output. Our creation of a Raspberry Pi HAT improves accessibility of laser show systems for hobbyists, and our software stack permits laser show enthusiasts to convert live video feed into a format they can use on their existing hardware.**

***Index Terms*- HAT, ILDA, Laser show, Raspberry Pi 4**

I. Introduction

This project improves upon existing laser-show technology by providing a hardware and software system that allows a user to convert live video feed into a laser show output. This is designed to appeal to hobbyists who wish to develop their own laser show systems. While other raspberry pi kits exist, they can require the hobbyist to design their own DAC circuitry and may lack hardware safety systems to prevent the hobbyist from eye damage while working with dangerous laser light levels [1]. Other non-raspberry pi laser show kits lack the programmability of a raspberry pi, and may not allow a hobbyist to modify the system's functionality. The software we design also provides a functionality requested by the lasershow community that does not currently exist for free on the market. By converting live video feed to ILDA format, the video feed frames can be run on most lasershow hardware systems. In order for our system to be of use, we must be able to meet the standards of other laser systems.

In order for this project to be successful, we must be able to provide a hardware system that can perform to similar standards as other hardware systems for the same price. Our system should be easily integratable onto other raspberry pi 4 boards, and our software stack should produce standardized ILDA frames for use on other laser hardware systems.

II. Design Requirements

We must ensure that the system is able to output video frames at a rate of 10/fps. This is the minimum rate for human motion recognition. We will measure this through measuring the difference in the time a frame is captured by the camera to the time a frame is output as DAC points to the laser's glavos, and by recording the visual laser frames with a slow motion camera for one second and ensuring ten distinct frames were drawn.

We must ensure our hardware system is capable of drawing ILDA frames of 12k points per second, which is equivalent to ten 1.2k point ILDA frames per second. We must also ensure our hardware can draw the standard ILDA test image. To confirm this, we will follow the standard ILDA test image calibration procedure, and our hardware must be able to properly render the test image. We must also ensure ten distinct 1.2k point ILDA frames are able to be drawn in a one second period, which we can verify through capturing the frames with a slow motion camera.

We must ensure that our safety subsystem shall never allow the lasers to persist in an unsafe state for more than 100ms. Human reaction time is approximately 1/4 of a second, and our system must be able to shut off the lasers faster than human reaction time in order to minimize exposure to dangerous amount of light. We will verify this system by attempting to run unsafe ILDA images with brightness levels and movement times outside of laser safety standards.

We must ensure that the displayed laser frame is visible in a room with light-levels of 50 lumens per square foot. This ensures the laser output is able to be used in a variety of light-level environments. The visibility of displayed laser frame is calculated from data obtained from laser module datasheet and CIE (International Commission on Illumination) color wavelength to lumens/watt table, the details of this calculation can be seen in Design Trade Studies section.

Lastly, we must verify that our color picking code can pick proper colors from images and projection laser color matches the sampled color points. Our method for verifying this is testing 3 different static images with out pipeline and observing the color projected points. Details and static images used in this verification is also given in Design Trade Studies section.

III. Architecture and/or Principle of Operation

Our system makes use of a Raspberry Pi 4, a Camera, a 2DOF Galvanometer system with Galvo PID drivers, a red, blue and green laser. We have also built a custom hardware shield with a built in DAC and safety monitoring microcontroller, and custom housing for the laser and galvos. High-level view of our system can be seen in Figure 1.

The Raspberry Pi 4 runs our system's High-level software responsible for converting camera feed into ILDA laser format. This converted image data is then piped to our low level software, which contains our DAC interface. Using SPI, the Raspberry Pi 4 communicates with the DAC to output image point values to our custom Raspberry Pi 4 HAT, the on-board DAC converts the point data into analog signals, and these signals are connected to their associated analog outputs - the X and Y galvos - and are also connected to our safety monitoring microcontroller. The safety monitoring microcontroller ensures that based on the analog outputs fed to the rest of the system, the system is still at a safe laser state. This is determined by the analog speed of the galvos computed with the brightness of each laser, and if those values are exceeded, the software on the safety subsystem will initiate a killswitch command, turning off the other lasers.

Our system has slightly changed from our previous report in that we use an off-the-shelf RGB laser board, as it provided a clearer laser stream. All other hardware was kept the same. We also added additional point-image filtering through interpolation as too many points to process were slowing down the system.

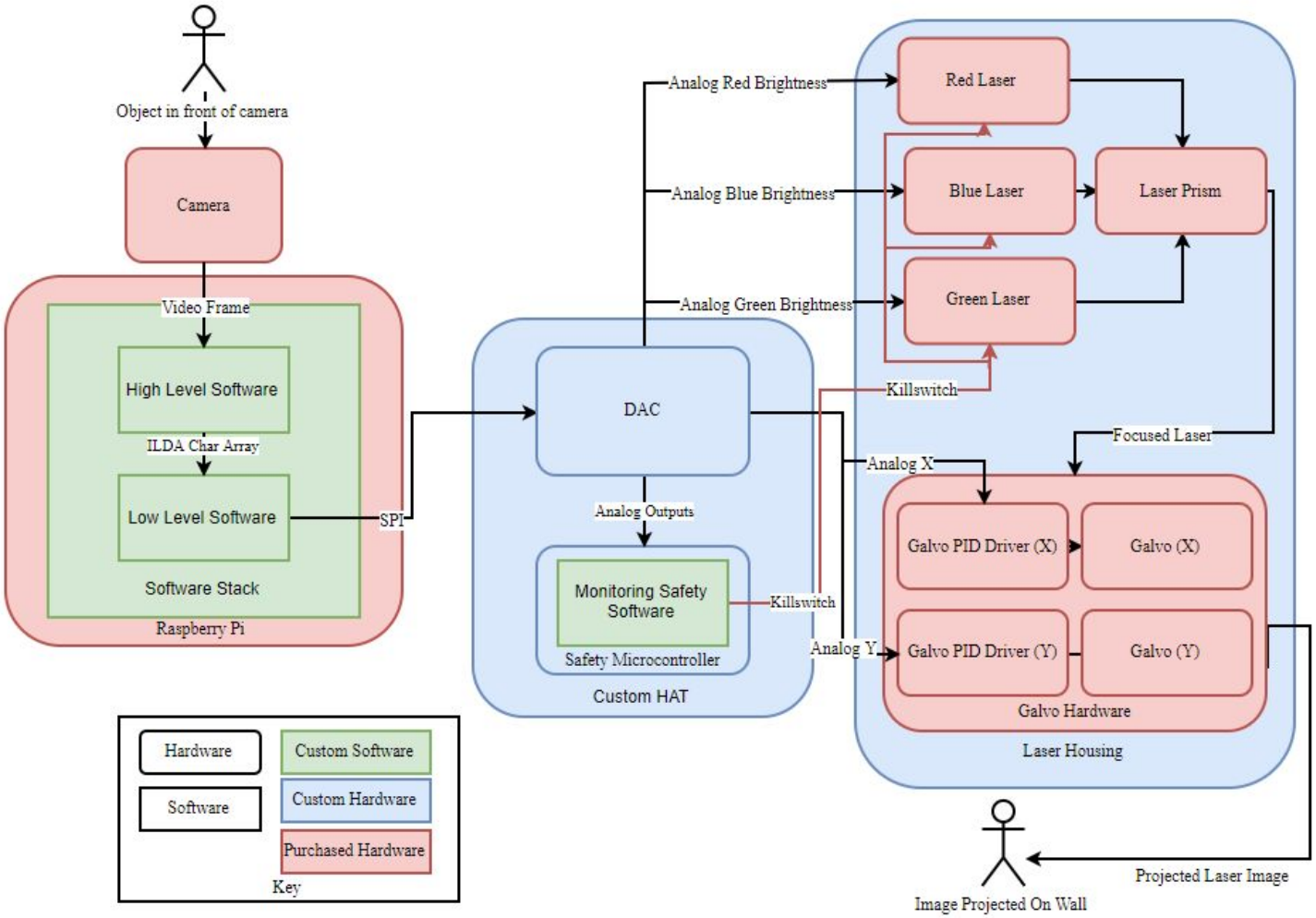

Fig. 1. System picture of the broad system

IV. Design Trade Studies

In order to satisfy our requirements, our high-level software must be able to convert camera video feed into an ILDA point data structure at a rate of no longer than 100ms.

ILDA format is defined according to the ILDA Technical Committee's ILDA Image Data Transfer Format Specification document. This is a standardized file type used by the lasershow industry. We decided to use this format in order to allow any standard laser hardware to use our software stack. While we could have designed a more efficient data structure for our specific Raspberry Pi usage, we decided to use a standardized format to allow a larger audience to benefit from our video feed to laser input software.

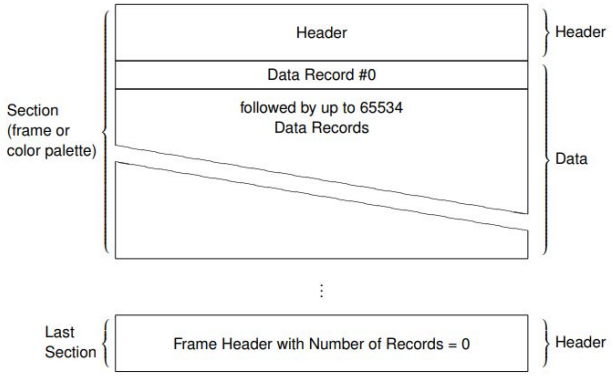

Fig. 2. High level view of ILDA format

For our system, we're using Format 5 ILDA data records. This is due to having RGB lasers, and since we're only working in a 2D coordinate space. While some systems may permit X, Y, and Z coordinate spaces, as our system doesn't read in any depth data from the Camera we're using, we decided to only mirror a 2D image in laser format. Since most users may not have access to a depth camera, this was chosen as a more affordable option.

Format 5 records have a size of 8 bytes and the following structure:

15 … 8	7 … 0
X Coordinate (1-2)	
Y Coordinate (3-4)	
Status Code (5)	Blue (6)
Green (7)	Red (8)

Fig. 3. Format 5 ILDA data record

An ILDA file consists of a series of data records, which correlate to points in an image. Using these points, it is possible to calculate the analog outputs that move the galvos so that a point is projected by the galvo-laser system that corresponds to the ILDA's 2D coordinate system.

These data points from the ILDA data records are used to control the galvanometers and laser brightness through DAC (Digital to Analog Converter). Our DAC system should update 5 different channels of outputs every

$$t = \frac{0.1s}{\#\ of\ points\ in\ ILDA\ Frame}$$

Equation 1

to successfully draw the frame. Since our requirements for points in a frame is limited by 1200 points per frame, our DAC should be able to update its values minimum every 83 microseconds. For this purpose, we had to make a decision between multiple DACs or using a single multi-channel DAC. After discussions, we decided to use a multi-channel DAC with trigger feature to update all of 5 data points simultaneously. Our designs use Texas Instruments' DAC60508 8-channel 12 bit digital to analog converters to satisfy these requirements. This DAC can update all 8 channels with single command with update time of 1 microseconds so it is a suitable choice for our design requirements.

Our design makes use of the open source PiGPIO library. We decided not to use the more common WiringPi GPIO Interface library because the WiringPi library uses the Linux SPI handler which has a maximum new transfers per second of approximately 70 thousand. With six transfers per point, achieving 12k points per second was simply at the upper limit of the library. With PiGPIO, we are able to perform hundreds of thousands of transfers per second ensuring that the system is no longer bottlenecked by SPI but rather by the physical limitations of the galvanometers and of heat dissipation. We could have also created our own library but decided against it because it may have introduced bugs into later stages of the project and would have locked the project into using only the Raspberry Pi 4. We want to potentially release the source code for this project to others and so limiting future users to a certain Raspberry Pi seemed less optimal.

The design requirements of our laser projector setup are mainly based on individually defined subsystem requirements that are chosen to ensure a smooth projection for human eye. In this part of the section, test results of requirements explained in Design Requirements section will be displayed on per specification and subsystem basis.

A. 10 Frames per Second Projection

Main performance focused design specification of our system was to be able to project at 10 frames per second with maximum points per frame is set to 1200 points. This specification was a factor in both hardware and software design aspects of the system. For all of our subsystems that has to satisfy this requirement we determined individual ways of measuring the performance. Lastly, a performance test including all these parts added to each other is also reported.

First subsystem in our projection pipeline that has to comply with frame rate requirement is the High-Level Software subsystem that is responsible with grabbing frames

from the camera, running edge detection and converting found edges in to ILDA format to be placed in a double buffer. It was key for this subsystem to work at fast as possible to not create a bottleneck in provision of frames to our low-level software subsystem. In our initial designs this system had been planned to be coded in Python but after getting stuck around 6-7 frames per second at 1200 points per frame we decided to code the whole pipeline in C++. The testing method that is used to measure FPS is using timers that keep time from the start of our subsystem receiving frame from the camera to placement of created ILDA structure into the double buffer. The results of our measurements are shown in Figure 4. As seen from the table our High-level subsystem design surpasses our initial requirement of 10 FPS at 1200 points per frame since all the samples are way below 100ms conversion time for 10FPS and doesn't create a bottleneck for following subsystems in the pipeline. Our average conversion time is 32.64 ms per frame, which makes our frame rate approximately 30 frames per second, which is three times above our spec requirements.

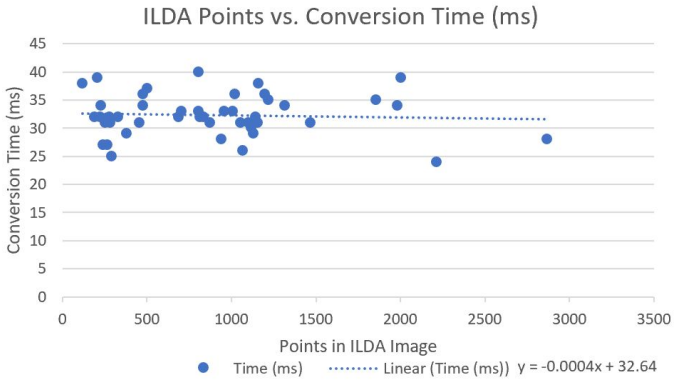

Fig. 4. Effects of Point Number of Conversion Time

Secondly, as the next part of our pipeline, low-level software subsystem receives produced ILDA frames from double buffer and process these frames to send them to DAC (Digital to Analog Converter) via SPI (Serial Peripheral Interface). This part of pipeline has potential of being a bottleneck since we know our high-level system is up-to spec and our hardware can support 20K points per second which is higher than 12K points per second requirement. For SPI, our system communicates with the DAC at 50MHz SPI clock frequency which is the maximum our DAC can support. Calculations for timing requirements were explained previously in the same section and calculations for delta between points is given in Equation 1. As explained previously, we know our DAC can support our timing requirements. In order to test this, we selected a static image (Figure X) to be projected through our pipeline with ~1200 points and recorded a slow-motion video footage of the projection. By watching and analyzing this footage, we are able to verify that projection is refreshed 10 times in a second which satisfies out specification.

Lastly, for hardware subsystem, we didn't need to do any physical testing since we had datasheet values that showed our galvanometers were able to support 20K points per second and DAC was able to update its values every 1 microseconds. In addition to these our test of whole pipeline showed us our hardware components are able to support 12K per second specification.

B. Safety System Specification

For the safety subsystem, ensuring that system causes minimal harm to nearby users. Prolonged exposure to laser light can cause blindness but temporary exposure can cause flash blindness. Flash blindness is a temporary loss of vision caused by exposure to bright objects. It is often experienced when glancing into the sun. The length and severity of flash blindness is determined by exposure time and so minimizing this is important. The safety subsystem reacts to a lack of changing inputs to the lasers and disables them after a certain amount of time which we initially specified to be around 100ms due to it being approximately human reaction time and it only being able to cause a few minutes of flash blindness at our lasers powers and wavelengths.

While the adc can determine if there has not been a change to the system at over 1MHz, it does not make sense to check this fast because the DAC is only ever sending new coordinates at approximately 10kHz. As such, the maximum reasonable time is far shorter than 1Mhz. In the below, you can see that the system has a 200Hz response time: a time found to be optimal in testing.

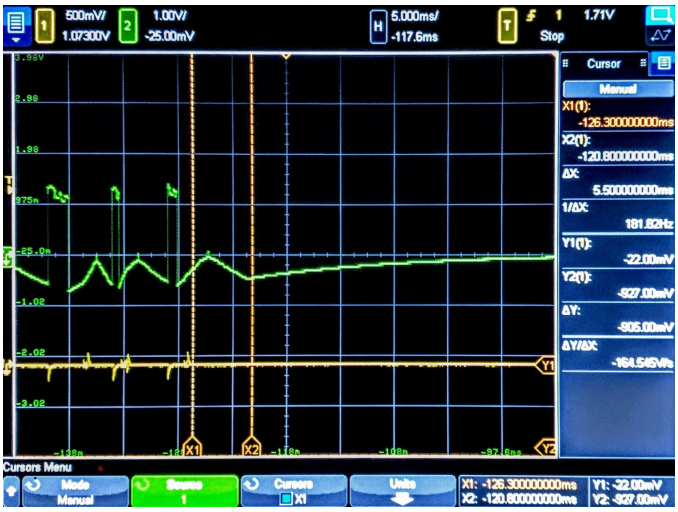

Fig. 5. Response time between DAC stall and laser shutdown.

C. Laser Brightness of 50 Lumens

For laser projection, the brightness of the laser is key for visibility. To project with proper visibility in a brightly lit room, our specification require our laser to have 50 lumen brightness. In order to calculate the brightness of our laser

module in lumens, we are using the laser power levels from our laser modules datasheet. Color, wavelength and power of the laser is shown in Table 1.

Color	Wavelength (nm)	Power (mW)	Lumen/Watt (lm/W)
Red	638	100	130
Green	515	100	320
Blue	450	100	32

Table. 1 - Laser color, power and wavelength table

The total lumen value can be calculated with equation:

$$\varphi_{total} = K_{red} * P_{red} + K_{blue} * P_{blue} + K_{green} * P_{green}$$

Equation 2

As a result of this calculation we can see total Lumen value of our alser setup is 48.2 lumens. Even though this number is below our initial specification, with visual verification of the laser brightness from our tests, we can conclude that it is bright enough to be visible in a brightly lit room.

V. System Description

A. Subsystem High Level Software

a. Our Frame to ILDA system handles taking in a frame from the Raspberry Pi 4's connected Camera and converting it into an ILDA-standard frame. We're using OpenCV to package the incoming frame as C++ Mat data structure. We then perform some image filtering which currently utilizes Canny edge detection through OpenCV's edge detection libraries, and then converting that into a contour array through OpenCV's contour libraries. We also interpolate some of the points in the contours to reduce excess points, which allows images with the same clarity but greater drawing speed.

b. Once we finish packaging the contour structure, we process that through a contour to ILDA function that takes in a vector of vectors of points, and packages that into a char array that follows the ILDA technical committee's requirements for an ILDA file. We pass the pointer of this ILDA char array to a custom double buffer library that stores the passed in ILDA frame and waits for the DAC interface to read from the double buffer.

This process will continue for each new frame read from the camera. The frame to ILDA process occurs in its own thread on the Raspberry Pi 4, so as soon as it finishes processing a frame and writing to the double buffer, it will read a new frame from the Camera and begin processing the new frame. This process occurs unders in under 100ms (average of 32.6ms) in order to meet the requirement for processing at 10/fps. Writes to the double buffer library will override 'stale' frames. Frames are considered stale if they haven't yet been read by the DAC. This is considered a fair trade-off as it is deemed more important to have up-to-date frames than to use old video feed if the frame to ILDA thread has to wait for a DAC read.

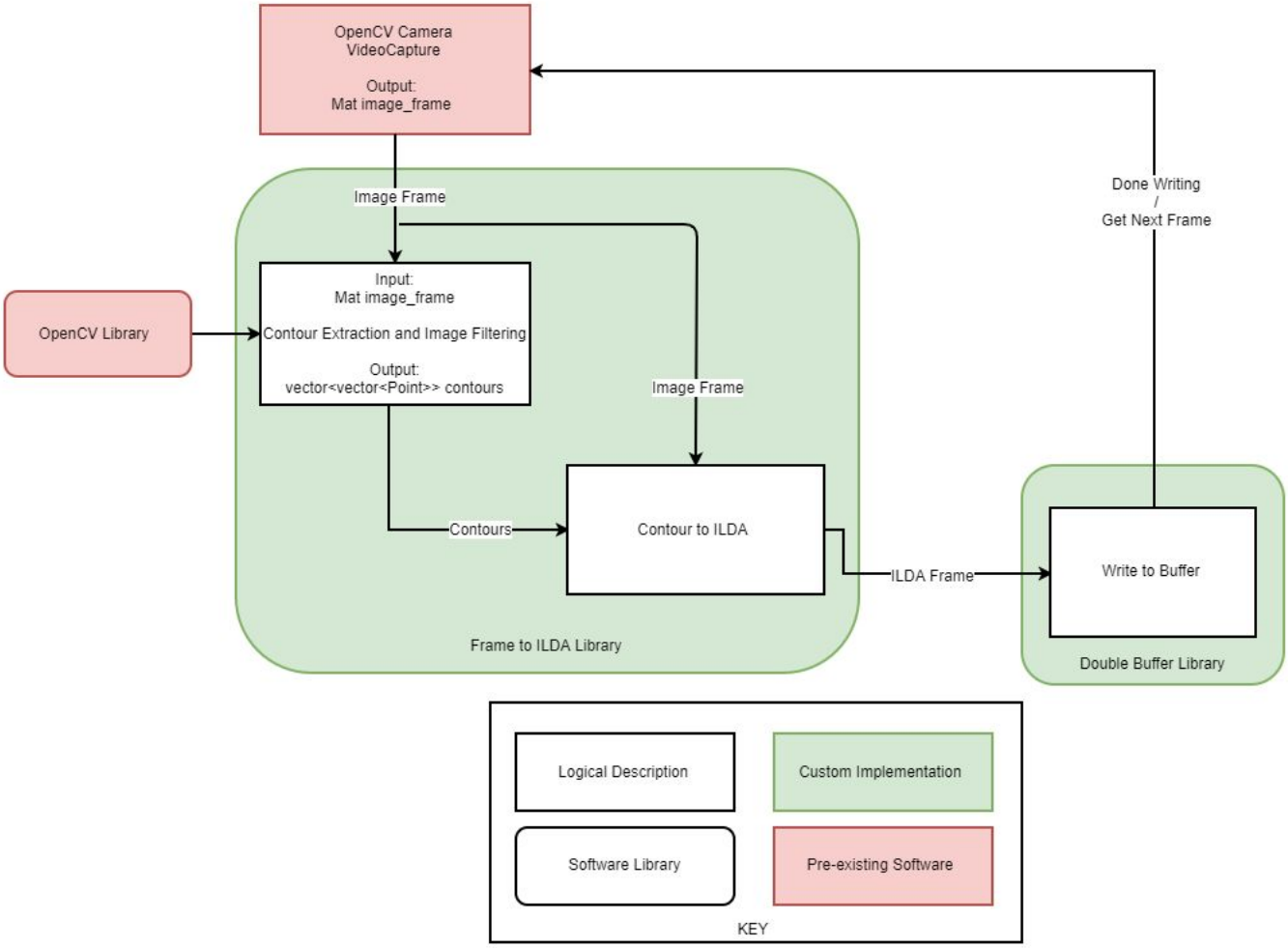

Fig. 6. High Level Software System

B. Subsystem Low Level Software

Our low level software module handles the control of galvanometers and lasers according to data buffered in the double buffer. When system is started for the first time, the low level software subsystem initializes the communication with DAC and verifies that SPI communication between the Raspberry Pi4 and DAC is functional. The verification for successful communication is conducted by reading the DEVICEID register of the DAC to verify we are talking to the right module. After this verification is done, subsystem continuously checks for a ready read ILDA frame in the double buffer. When there is a frame ready to be drawn, system starts reading the ILDA frame to find the timing constraints by checking the number of data points in the ILDA frame. This provides the system with the information on how frequently it should update. After timing information is extracted, the system starts reading points in the given order and processes these points to map to proper point in 12-bit range. Since we are using a 12-bit DAC and brightness values for lasers are in the range of $[0, 255]$, these should be mapped to proper number in $[0, 2^{12} - 1]$. Similarly, for the X and Y coordinates of the ILDA points, their values should be mapped to value from range that is in 640x480 image to values that would be transferred to DAC's 12 bit registers. In our ILDA conversion X and Y coordinates of points from 640x480 image are mapped to range $[-2^{15}, 2^{15} - 1]$. For this mapping center of the image corresponds to [0,0] points. Our DAC converts this range in values in $[0, 2^{12} - 1]$ and sets [0,0] to left top corner of the image. This full range mapping works since we are using full range of motion of our galvanometers.

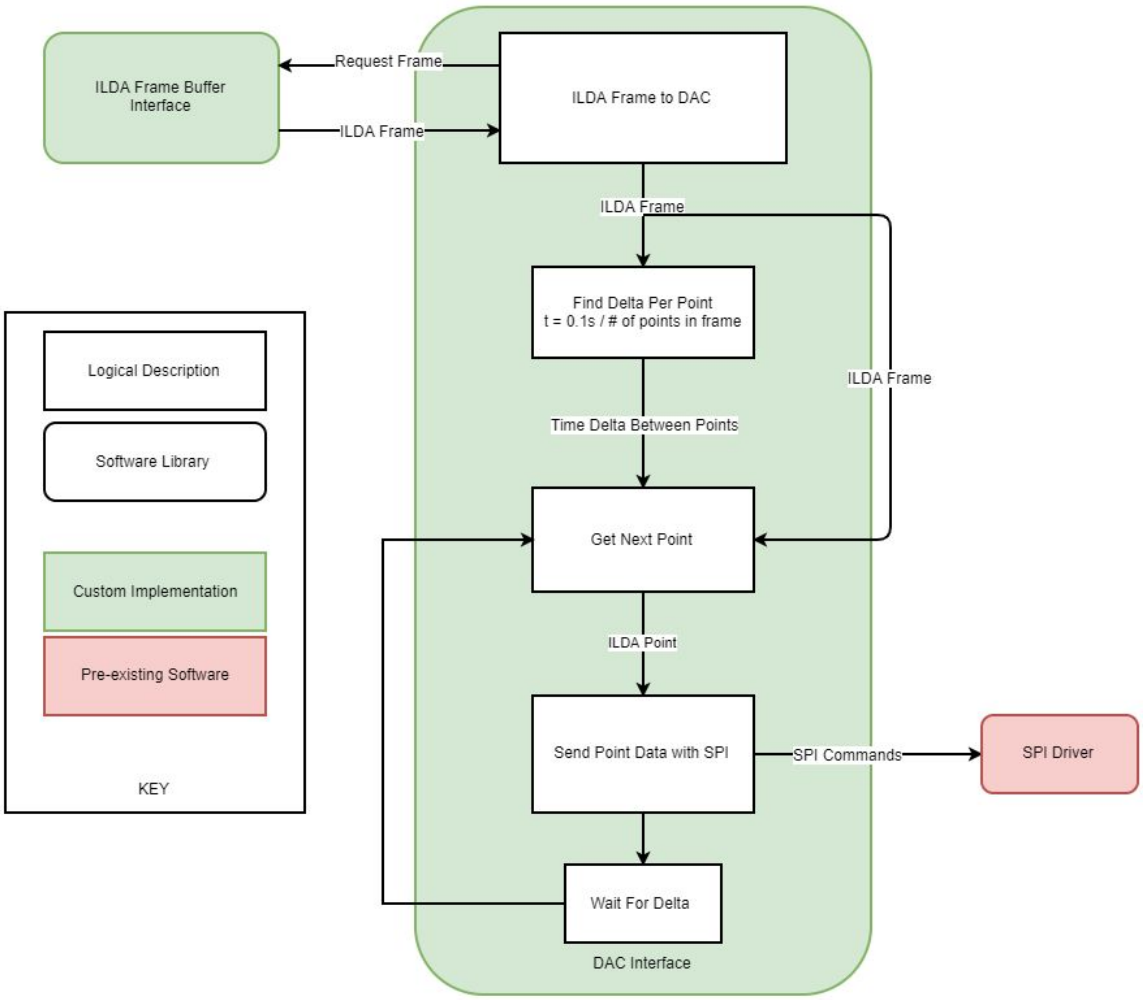

Fig. 7. Low Level Software System

Transfer of data points between low level software module and DAC is done via Serial Peripheral Interface (SPI). Both DAC and Raspberry Pi4 supports SPI clock speeds up to 50 MHz but our system design uses SPI clock set to 25MHz because of the limitations of the library we use for SPI communication. Top clock speed that our DAC support is 50 MHz. To successfully transfer an ILDA point to DAC, our system should send 5 different write requests and a single trigger request. Every request contains 3 bytes of information so a single update takes 144 clock cycles to transfer. In microseconds our single update will take 2.88 microseconds for transfer and an additional 1 microsecond for the update of galvanometer outputs. The total data write to DAC output time of 3.88 microseconds is much faster than our minimum delta between points time of 83.3 microseconds. This shows that our low level system design can easily keep up with points per second rate that we specified and will not be a bottleneck in application.

After the transfer of all ILDA point data is completed, our system goes into waiting mode for the delta time between points to pass. When the timer reaches the delta, system reads the next point and repeats the communication and trigger cycle and all the points in the ILDA frame is read. When all the points are transferred successfully, system requests another ILDA frame from Double Buffer and if it is available, it is directly used for drawing new points. If it is not available, the system waits until a new frame is written into the buffer.

While we set our design requirements in a way that there is always going to be frame available in the buffer since we will be satisfying the 10 FPS specification and low level software system times itself according to this specification, it is possible that Raspberry Pi4 can lose performance for a reason that is unknown to us. We don't our system to cease its operation in such scenarios. Therefore it is designed to wait for available frame until it is terminated.

C. Subsystem Custom Hardware

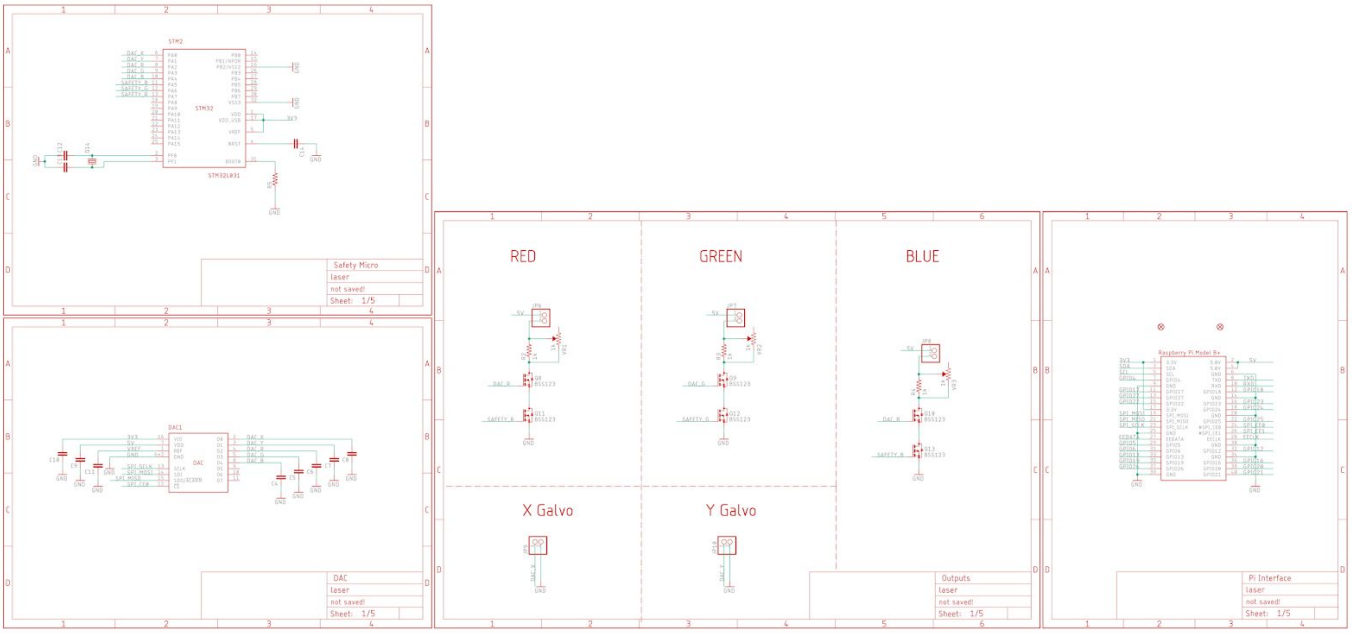

Fig. 8. Hardware Schematic

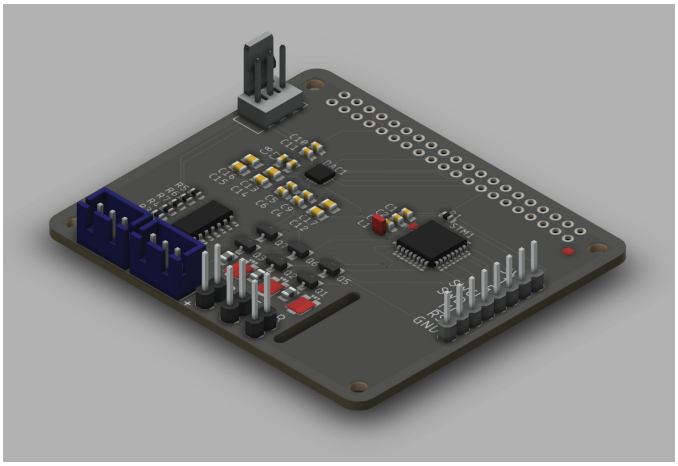

Fig. 9. Custom Raspberry Pi Hat

The custom hardware on this project contains the digital to analog converter and the safety subsystem. For ease of interfacing, these two components have been placed on a board which will interface with the Raspberry Pi like a "Hat." This is best demonstrated in (8).

D. Safety Subsystem

Due to the dangerous nature of lasers, having a safety subsystem to ensure that the total system does not enter an unsafe state is crucial. Problems are likely to arise if the software on the Raspberry Pi crashes in any state of the computation pipeline. As such, as shown in (1), we have decided to create a safety subsystem not contained on the Raspberry Pi.

There are two types of danger associated with lasers: eye damage and flash blindness. While none of the lasers we are using would be able to cause real damage to the eye, they can still cause flash blindness. Flash blindness is a condition in which exposure to a bright light source causes temporary blindness due to overwhelming the retinas. This condition, while not permanent (except in rare cases such as nuclear explosions), still causes discomfort and blindness that can last up to a few minutes. As such, it is important to protect against it.

For the safety subsystem, we have determined that at minimum, the reaction time must be less than or equal to a single frame's worth of time. Ultimately, having a faster time would be better because the faster the time, the safer the system is. As such, we are currently using a STM32L031 microcontroller due to the fact that these high speed microcontrollers have very quick analog to digital converters.

With a maximum sample speed of 1.14Msps. this microcontroller will allow for quick reaction time. The lower bound on reaction time is 4.385 microseconds as shown in equation (1). Of course, there is also some overhead time for interpreting this data and actually shutting down the lasers but the quick speed of the analog to digital conversion ensures that a major time sink is eliminated.

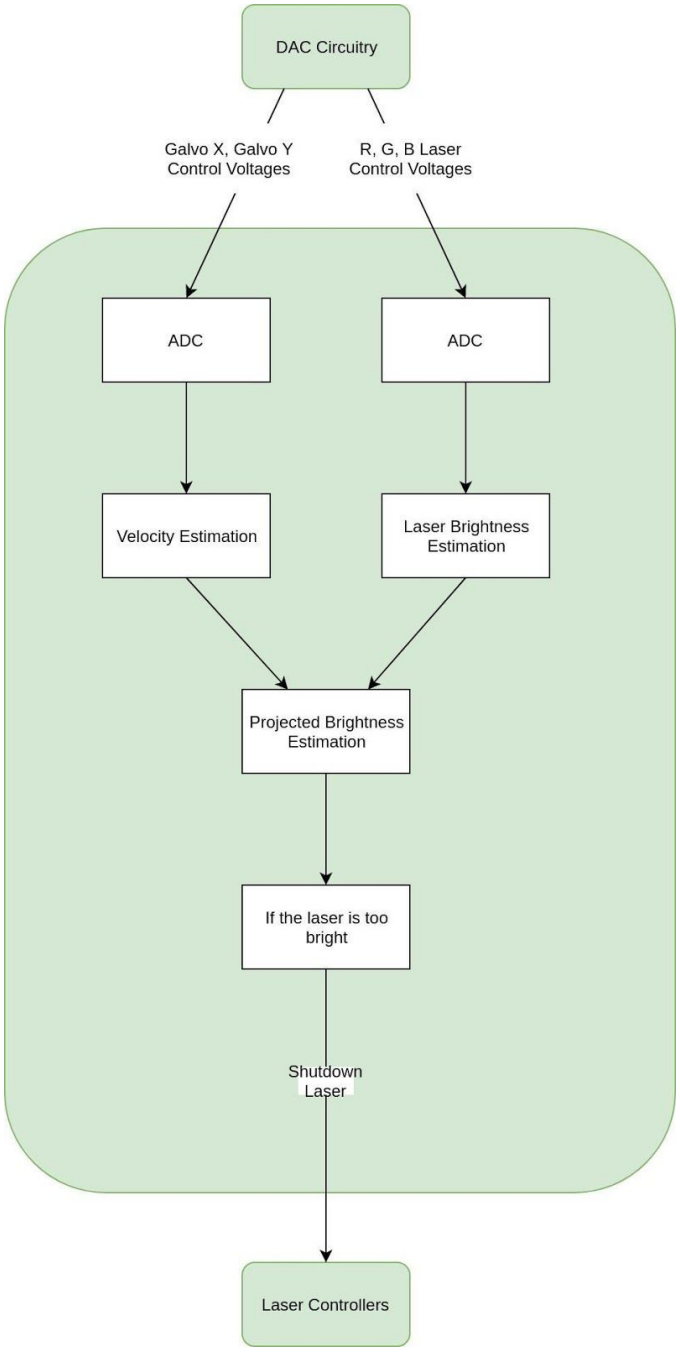

Fig. 10. Safety Subsystem Block Diagram

While this 100ms response time definitely supplies sufficient safety for lasers of our size, if this system was ever to be used with higher power lasers a faster response would be needed. If we can manage to get the performance we expect out of the safety subsystem, it will even be suitable to ensure that many higher powered lasers do not cause severe flash blindness. The speed required to eliminate mitigate the hazards can be seen in.

We tested this safety subsystem by sending an illegal frame to the DAC. An example of such a frame would be one where there is no galvanometer velocity and full laser brightness. We used an oscilloscope connected to the laser output to measure the length of time it takes to shut down the laser. Most lasers have a very quick startup and shutdown time on the order of nanoseconds and so measuring the signal going into the laser itself is adequate to determine if it is actually emitting or not.

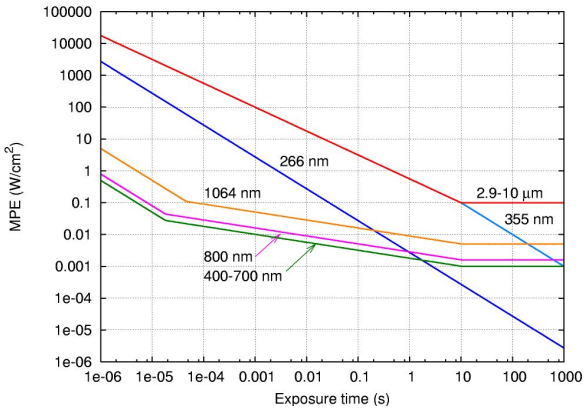

Fig. 11. Laser ExposureTime and Brightness Level [2]

VI. Project Management

A. Schedule

We've worked on the different subsystems of the project in parallel, breaking it up into the high-level software, low-level software, and hardware components. We've kept mostly on schedule from the previous report, we had finished merging our high-level software with our low-level software by October 17th, and were able to test a single red laser demo. By October 29th the hardware systems were finished, and by December 2nd we had the whole system working with RGB lasers and demo-able. By December 7th, we tweaked more GPIO and filtering code and improved laser-output frame rate that exceeds our 10fps requirement.

B. Team Member Responsibilities

Eliana Cohen: Primary responsibility is to implement a method of converting an incoming video stream to a vectorized ILDA compliant frame. Secondary responsibility is to assist with hardware design.

Enes Palaz: Primary responsibility is to implement hardware drivers to convert ILDA compliant frames to commands for the DAC. Secondary Responsibility is to help with video vectorization.

Jake Zimmer: Primary responsibility is to implement physical hardware for the DAC and safety subsystem and to write software for the safety subsystem. Secondary responsibility is to help with low level hardware drivers.

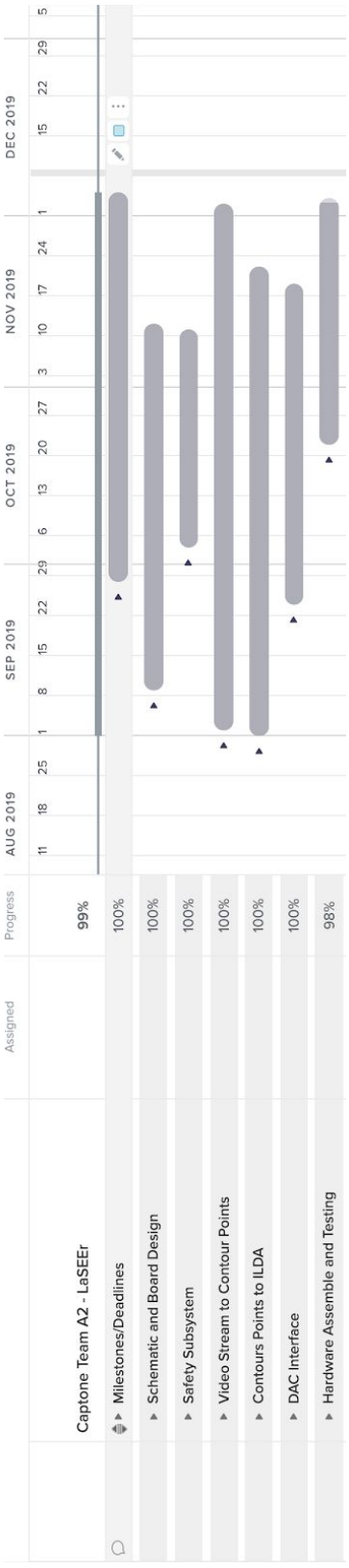

Fig. 12. Milestone Chart

C. *Bill of Materials*

#	Bill Of Materials				
	Name	**Price (USD)**	**Quantity**	**Purchased**	**Source**
1	Galvanometer Set	79	1	Yes	Ebay
2	Raspberry Pi 4 - 4GB version	0	1	Lent	Roboclub
3	Raspberry Pi 3 B	0	1	Lent	Roboclub
4	Logitech - 4K Pro Webcam	0	1	Lent	Roboclub
5	X-Cube	21.89	1	Yes	Amazon
6	RGB Laser Module	98.98	1	Yes	Amazon
7	DAC60508MRTER	6.15	2	Yes	Digikey
8	DAC60508MRTER	0	4	Samples	Texas Instruments
9	QFN16 to DIP Breakout Board	4.8	2	Yes	Digikey
10	STM32L031K6	2.68	2	Yes	Digikey
11	Misc Components (res., cap., etc.)	15	NA	Yes	Digikey

Table. 3 - Detailed list of materials

As seen in Table 2, for the design part of the project we tried to manage our budget carefully by lending parts from campus resources for testing their performance and specifications instead of directly buying them. Because of this most of our main microprocessors are both listed as to be bought and lent at the same time. We also tried to make use of requesting samples from manufacturers like Texas Instruments to reduce our costs and get access to varieties of similar products like various DACs.

D. *Risk Management*

One of the problems we encountered with our high-level software stack was speed when trying to convert images. We initially implemented that stack using Python, and when we found this implementation was too slow, we were able to switch to a lower-level language for performance improvement when we switched to C++. Speed was one of the riskiest components of our project, as we needed to ensure that we met the 10/fps requirement. Later on we found we were throttled by GPIO speed, and switching our GPIO libraries allowed us to output more points per second from the DAC. Our current bottleneck is now the galvo's speed, but we are well within framerate specs with our galvo's ability to draw 20k points per second.

VII. Summary

Our system satisfies the requirements that were set initially at a satisfactory manner. While we were able to hit our initial specification goals, our selected hardware capabilities prevent our system from scaling up properly with increasing number of points in frames. One issue that we had throughout our project is dropping frame rates with increasing number of points. To address this issue with the hardware that we currently had, we decided to create a controlled setup for our demo with a white backdrop placed a user and camera. As a future improvement, it would be a great challenge to make the system scalable with increasing amount of points. Main ways of doing this would include using faster hardware components (like ADC, a computer and galvanometers) and coding an adaptive conversion algorithm which would actively change filtering parameters based on number of points in received frames.

References

[1] https://damow.net/building-a-laser-show/
[2] Wikipedia, https://en.wikipedia.org/wiki/Laser_safety#/media/File:IEC60825_MPE_W_s.png

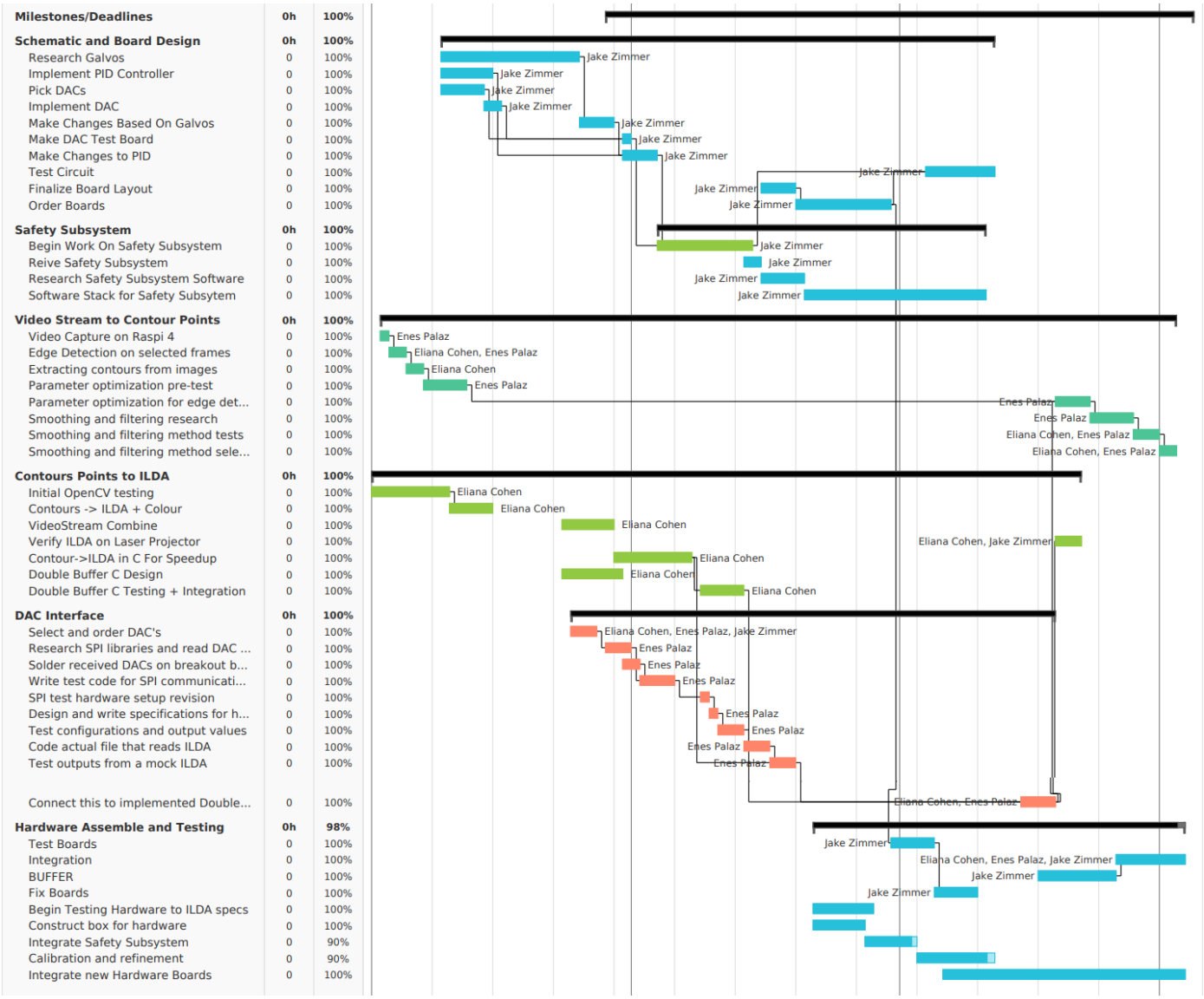

Fig. 13. In-depth Milestone Chart