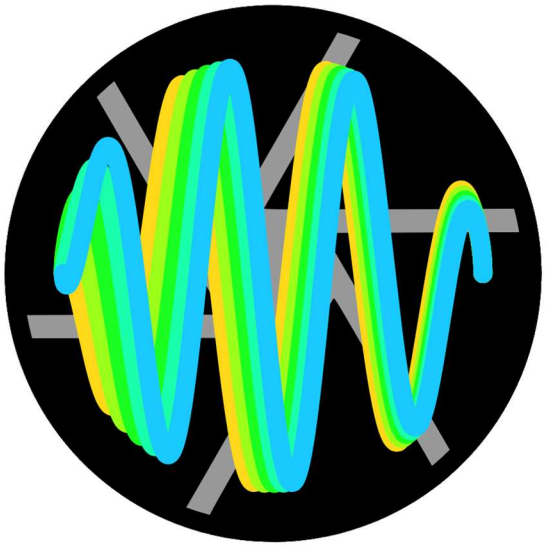


TSUNAMI

Hyperspectral Imager for Ecological Analysis

Group 9 Authors:

Emilio Armas
Electrical Engineering and Optical Engineering

Jacob Silver
Optical Engineering

Xander Kin
Electrical Engineering

Review Committee:

Mark Maddox
Lecturer, ECE

Dr. Konstantin Vodopyanov
Professor, CREOL

Advisors:

Dr. Chun Yong Chan
Senior Lecturer, ECE

Dr. Aravinda Kar
Professor, CREOL

Table of Contents

Chapter 1 – Executive Summary

TSUNAMI is a senior design project at the University of Central Florida serving as a hyperspectral imager for ecological systems. Different molecules in plants and soils have different absorption spectra, leading to different visible colors. However, the human eye as well as standard camera sensors can only resolve three color channels: red, green, and blue. This leads to various mixtures of molecules looking the same to the eye but truly containing important unresolved spectral data. A hyperspectral camera can resolve this data by diffracting an incoming slit of light and scanning across a scene, creating a set of 2D images with each at a specific wavelength, turning the three color channels of the eye and standard cameras in hundreds of channels.

TSUNAMI consists of visible-NIR optics translated via a stepper-motor-driven rail system and interfaced through a Raspberry Pi 4B. Our image target will be a flat bed of soil with different minerals and biological materials. The image data will resolve different mineral and biomolecule concentrations and could serve as a powerful tool for the agricultural and environmental sciences. This project also serves as the culmination of the electrical and optical education experience of Emilio Armas, Xander Kin, and Jacob Silver.

This report will detail the design considerations, decisions, and implementations we've made. Chapters act near-chronologically, with different considerations and choices first and decisions and implementations later. Component information is contained in appropriately labeled sections, with bills of material located in Appendix B.

Chapter 2 – Project Description

2.1 Project Background and Motivation

Many imaging systems in our daily lives are designed around the fact that human eyes are spectrally selective to three general wavelength bands. Through natural evolution, our biology developed sensitivity to red, green, and blue light within the visible range of the electromagnetic spectrum. As a result, most cameras and displays we use for general purposes are designed to both simultaneously capture and output these three wavelengths in varying strength, hence red-green-blue, or “RGB” color-space. The pixel, or the most fundamental unit used in imaging, is therefore a three-element vector that serves as the atom of the color images and videos we use on a daily basis.

Three-band RGB spectral information can tell us a lot about the world around us, allowing us to use wavelength as a factor in how we make decisions. Having only three spectral data points on a given pixel is sufficient for such things, but imaging more than just three (multi-spectral), or even a full spectrum of wavelengths per pixel (hyper-spectral) can provide researchers and professionals with images containing a vastly larger wealth of insight from the information it provides. In typical applications, RGB (and sometimes multi-spectral) imaging is accomplished with the use of a spatially multiplexed filter array, assigning a different wavelength filter per pixel in a repeating array, sacrificing the base resolution of the image sensor with the number of bands to be imaged. In RGB imaging, a square repeating pattern of two green, one red, and one blue filter is used, which is referred to as a “Bayer” filter, conceived by Bryce Bayer of Eastman Kodak [1].

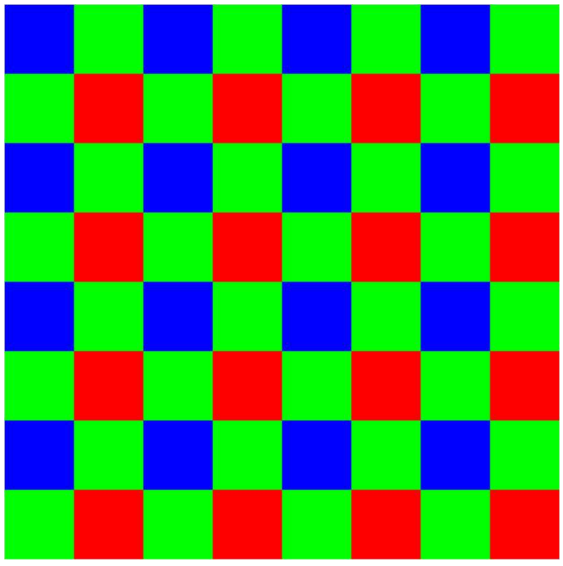


Figure 1: Bayer Filter color pattern

Hyperspectral imaging is typically accomplished by either switching one axis of an image sensor from spatial to wavelength, or done with a variable and tight controllable filter for an incident image. These are referred to as spatial scanning and wavelength scanning techniques, respectively. We intend to use spatial scanning via translation of our instrument, as a large-aperture tunable bandpass filter with very tight passband is prohibitively expensive and not possible for the spectral resolution we are interested in. In spatial scanning, the imaging field of view is left as a line,

which diffracts internally to the spatial/wavelength dimensioned image. With composite processing of all images recorded at all scan positions, a final image “cube” is created, consisting of many monochromatic 2-D images, each centered at a different wavelength. This resulting dataset is extraordinarily useful for many remote-sensing applications, in fields such as agriculture, climatology, meteorology, geology, oceanography, and other sciences [2-5].

2.2 Current Technologies and Past Projects

2.2.1 Commercial: Resonon Pika L-F

Resonon Inc. offers a hyperspectral camera geared toward the agricultural processing industry. The Pika L-F has an operating wavelength range of 420-980 nm with 224 spectral channels and a full-width-half-max (FWHM) spectral resolution of 3.1 nm. The camera has a maximum frame rate of 585 fps, which allows it to be placed above an agricultural processing belt and quickly identify different produce types and/or qualities for sorting. They can image at an F/# of 2.8 and in a small form factor of 115 mm x 104 mm x 66 mm [2].

Figure 2: Resonon Pika-LF (see [2])

While this will be covered in more detail in later chapters, this product was a major inspiration for our decision to design a translational scanning system. A linear FOV such as that seen on this commercial product clearly lends itself to the lateral translation of either the imager, or its target object to be scanned. As many of these types of systems are placed above a belt or translational scanner, we chose this option as other translation methods such as rotation or translation of the slit itself would prove too difficult to design in a reasonable timespan. From simple analysis and thought, we knew that the specifications of this commercial product designed and built by experienced engineers would likely outperform any design we could create at the collegiate level. As a result, these figures for wavelength resolution, spectral range, and spectral channels were very much taken into consideration when setting our expectations for what we could achieve.

2.2.2 Research: JPL AVIRIS Imager

The NASA Jet Propulsion Laboratory's Airborne Visible/InfraRed Imaging Spectrometer (AVIRIS) is a 380-2500 nm hyperspectral imaging system located on an aircraft. This imaging system has been used to sweep wide areas of California to look for oceanic, botanical, geological, and atmospheric markers and patterns. The camera has a FWHM spectral resolution of 10 nm and a field of view (FOV) of 34 degrees x 1 mrad. Images are taken from 4-20 km of altitude [3,4].

The AVIRIS project features impressive optical performance metrics and echoes the sentiments of the last section, as these figures give us a reference for where to set our expectations for practical limitations in our system. This imager like the last, also scans laterally, but uses the motion of an aircraft to scan the unmoving earth from the ground refence frame. From this project, we considered initially the idea of translating the imager but ultimately decided that because our instrument would likely be large and fragile, we would absolutely decide to translate our object instead. This compounded with the review of the previous project made this choice concrete.

2.2.3 Past-Collegiate: UARK Undergraduate Thesis

As an example of what has been done at the university level, we found the work of Joshua Moorhouse at the University of Arkansas Fayetteville. Moorhouse designed and 3D printed an optical housing for a hyperspectral imaging system which was able to differentiate three different laser lines. No field images were taken in this work and FWHM spectral resolution wasn't specified, but the optical design of the set-up gives us confidence that our optical design will be functional [5].

While still in the research phase of the project, we were still concerned with the practicality of creating an imaging spectrometer like the previous two cases studied. Seeing a system that functioned at a basic level, being accomplished by *one* undergraduate student, gave us the confidence to proceed with our design as something feasible. Although we had less time, we are a group of three and this aided our decision. Some other key takeaways mainly pointed towards the physical construction of the imaging optics. Knowing that optics could be supported at least to a basic level of functionality from 3D printed parts alone ensured that we wouldn't necessarily have to stretch the budget for custom machined parts. This at least not for the majority of components.

2.3 Goals and Objectives

In this section, we outline our goals and objectives for the TSUAMI imaging spectrometer instrument. **Goals** are defined through the project's overall purpose and **qualitative** motivation, while our **objectives** are written to detail **quantifiable deliverables** to guide our implementation. Our goals and objectives are detailed within three tiers: "basic", "advanced", and "stretch". Basic goals/objectives MUST be met, with advanced goals/objectives being attempted with full effort. Within these, our team also details "stretch" goals and objectives. While we may not adhere to these in our final product, we will guide our design in such a way that these are possible without *significant* extra development.

Table 1: System Performance Goals

Basic Goals	Advanced Goals	Stretch Goals
Create an imager to faithfully capture spatial and VNIR spectral (400nm-850nm) data simultaneously	Design the instrumentation optics to limit chromatic aberration along the spatial capture axis	Design the instrumentation optics to be completely diffraction-limited on the spatial capture axis
Capture and display data locally on an embedded computing platform, with a small screen	Implement the user interface as a touch screen for easy display of the scan results	Design a custom physical user interface with dedicated mechanical buttons, knobs and display lights
Locally perform basic image rendering tasks to display the captured data at a selected wavelength	Locally perform more advanced rendering to show false-color images as remapping of the RGB display channels	Be able to export finalized hyperspectral data to another device via removable storage media
Implement a laterally moving translation stage to allow the instrument to scan a small ecological target, such as soil or plant matter	Perform real-time monitoring of the translation stage's position to synchronize scanning for smooth outputs	Design an auto-calibration routine for the imager's spectral response and translational scanning accuracy
Create a line-shaped laser "cursor" to match the imager's linear target area, allowing for a user to see the scan location before capture	Develop a high-performance laser diode driver to pulse the laser cursor out-of-phase with the captures, allowing for live display of scan	Use multiple different wavelength lasers to indicate the functional state or progress of the scan along the target as colored information

Design optics to feature a fixed target focus to the target sample on translation platform	Allow for adjustment of target focal length to the sample	Incorporate secondary optics for magnification, or mount to attach 3rd party lenses

Table 2: System Performance Objectives

Basic Objectives	Advanced Objectives	Stretch Objectives
Implement embedded motor control of translation stage using a microcontroller's (MCU) timing and data-transfer peripherals	Design the MCU firmware to monitor power supply voltages and currents to report back to the embedded computer	Write custom monitoring code to automatically detect potential power faults as well as reporting these to the computer
Utilize an embedded computing platform to read-out data from a commercial CMOS image sensor and render the UI locally	Design the MCU firmware to pulse the laser cursor out-of-phase with timer control, with synchronization signals from the computer	Support variable brightness and closed-loop MCU control of laser optical output power via the laser diode's auxiliary photodiode
Ensure the MCU handles all other tasks besides UI processing and CMOS sensor readout, **being commanded by the UI software over serial interface**	Utilize a graphics programming library on the embedded computer to design a custom, interactive touch user interface for rendering and control of the instrument	Connect buttons, switches, knobs, and LEDs to extra peripherals on the MCU to create a custom interface, relaying user inputs to the embedded computer
Construct all optical and mechanical structures such that no optical or mechanical alignment needs to be done by the end user	Design the MCU firmware to support open-loop control of the motor's angular position, using micro-stepping techniques to move accurately.	Adapt the MCU firmware to support closed-loop feedback of the motor's angular (and thus lateral) position
Design power electronics to supply the computer, control, and MCU systems from a single external DC feed with their needed	Design the translation motor power supply to have digitally programmable voltage and current limiting ability, protecting against back	The translation stage may be tuned to achieve better positional accuracy per pixel than the directly image spatial axis of the imager

voltages and currents, reliably	EMF all while interfacing to the MCU	

2.3.1 Overall Application Diagram

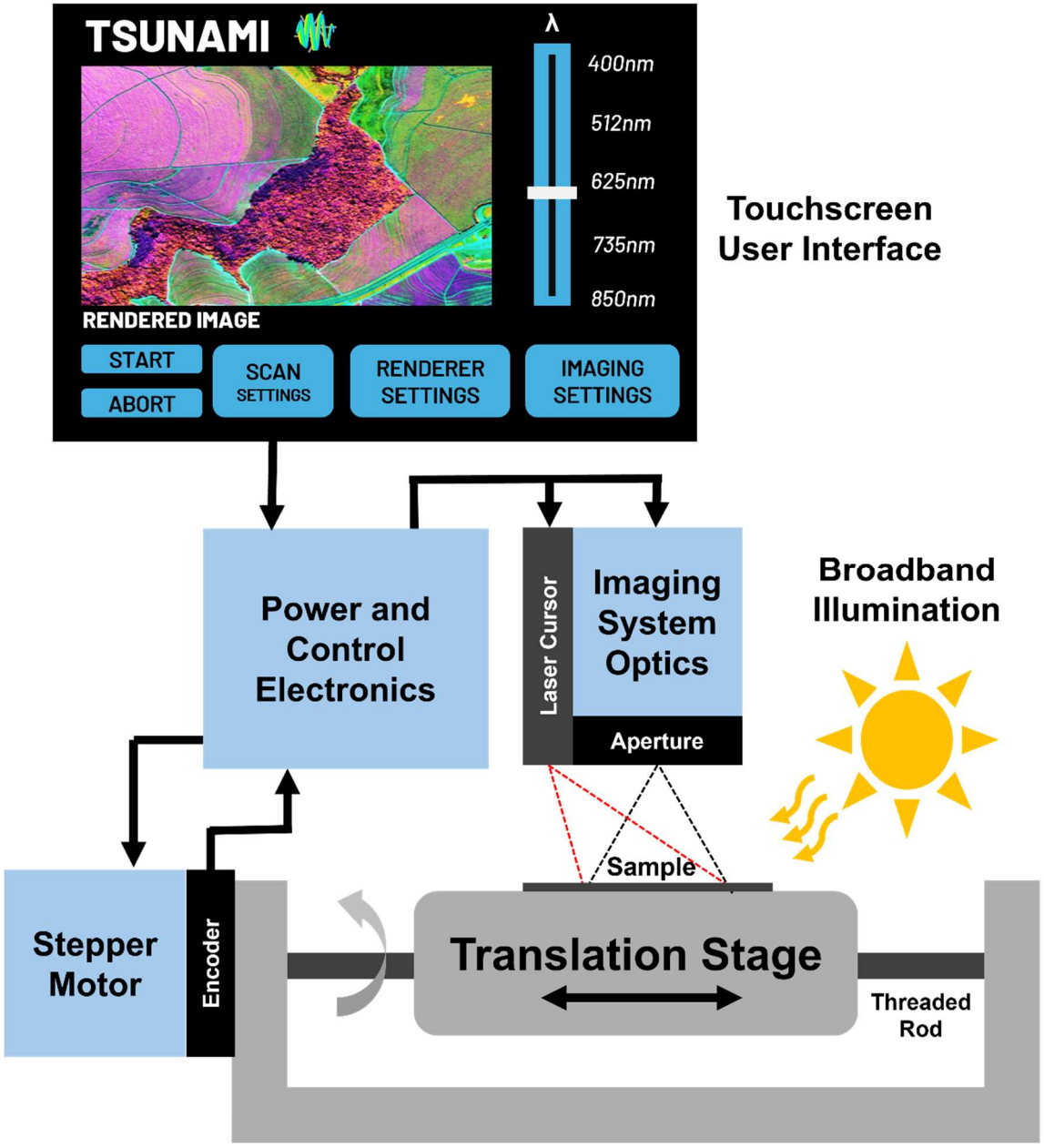


Figure 3: General application diagram

This general diagram shows the concept of the instrument as a whole. Our sample is to be translated along a CNC-stepper actuated mount. The optical systems will be built into a designated optical assembly, which is then paired with our

electronics/PCB stack to power and control all systems. The end user is then able to control the instrument and see results from the externally mounted touch screen that connects to the embedded computer.

2.3.2 Imaging System Optical Diagram

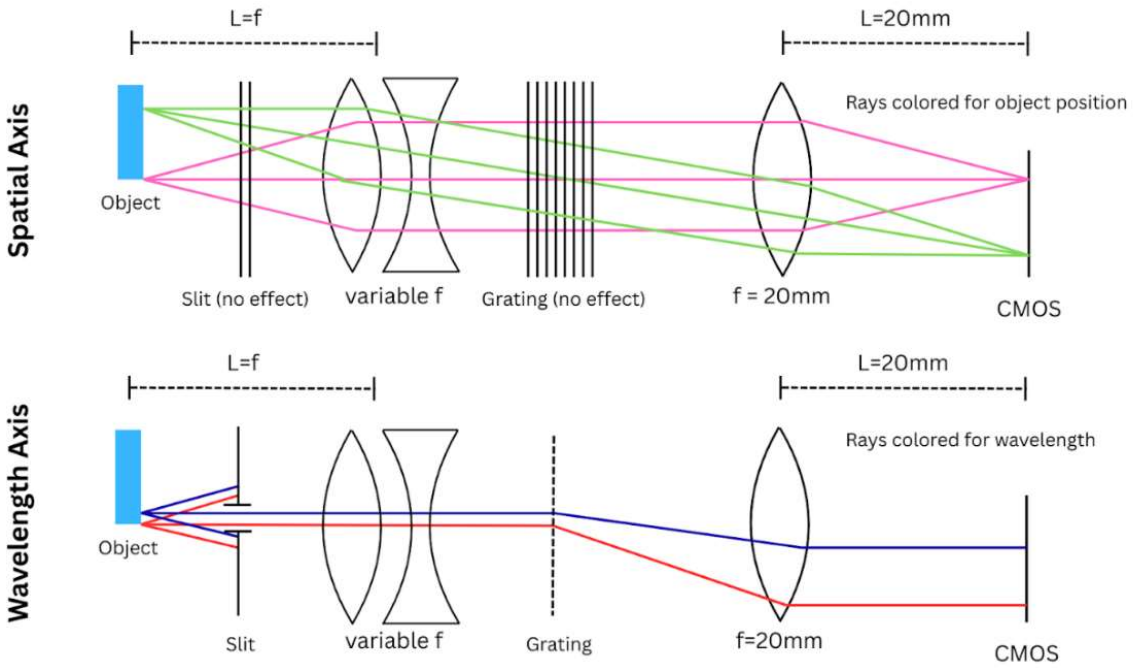


Figure 4: Optical ray diagram of imaging system

The optical ray diagram above depicts the optical paths along both the spectral and spatial axes, separately. Basic optical components are paraxially traced to show basic function of the imaging system. The rays originate from the left-hand side of the diagram and propagate to the right onto the CMOS image sensor. On the top diagram, the spectral/wavelength axis information of the object's image is projected onto **longer** axis of the image sensor. For the bottom diagram, the spatial information of same object across the linear field-of-view is projected onto the **shorter** axis of the image sensor.

2.3.3 Hardware Block Diagram

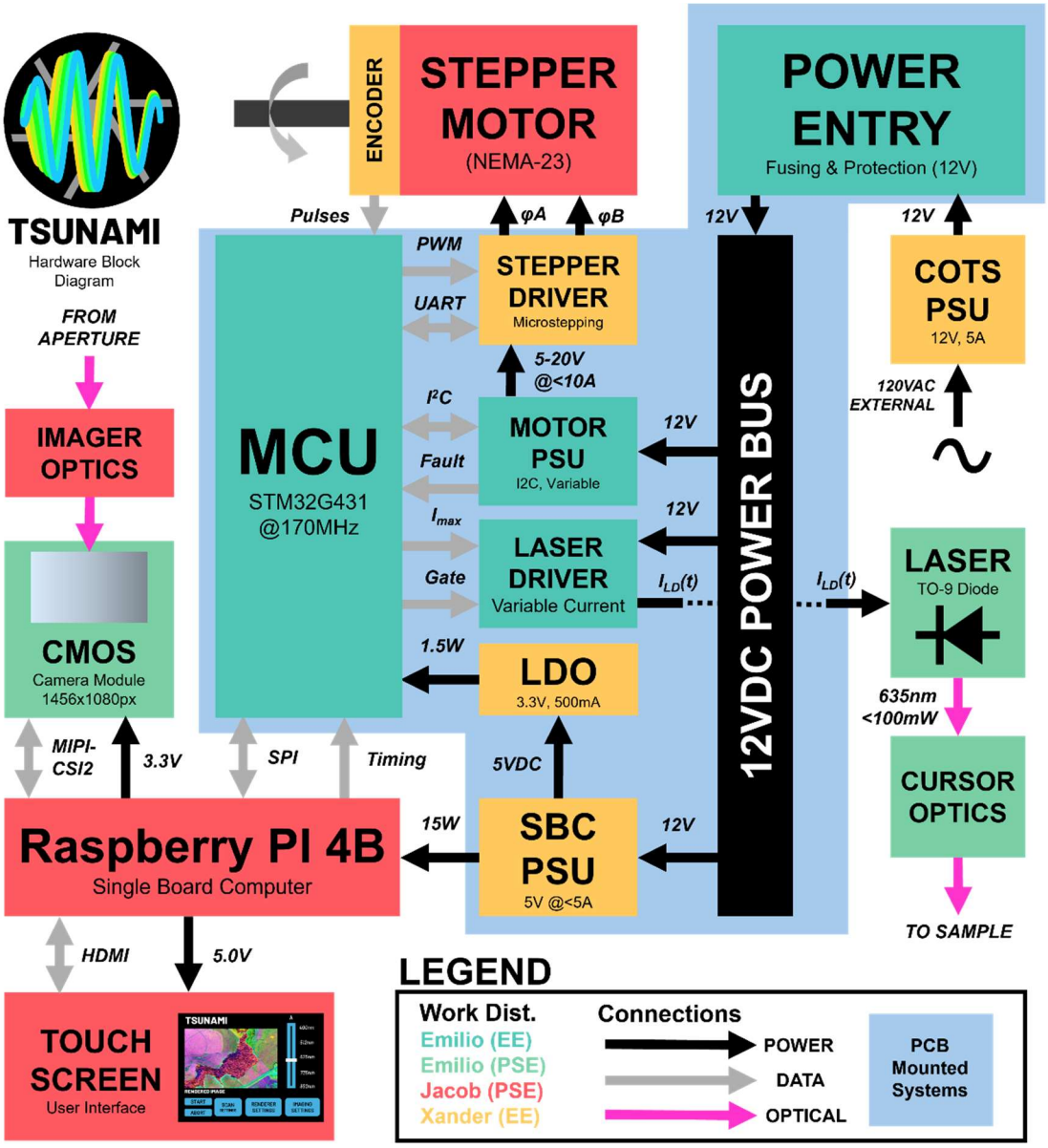


Figure 5: System hardware block diagram

From the legend in the above diagram, work distribution per person (and per major) is highlighted via the color of the hardware blocks themselves. Connections are illustrated with arrows, and the bidirectional ones serve to indicate two-way transfer. Color coding is used to denote connection type, and all parts/circuits designed to be integrated into our PCB stack are given a light blue backdrop.

2.3.4 Overall Software Block Diagram

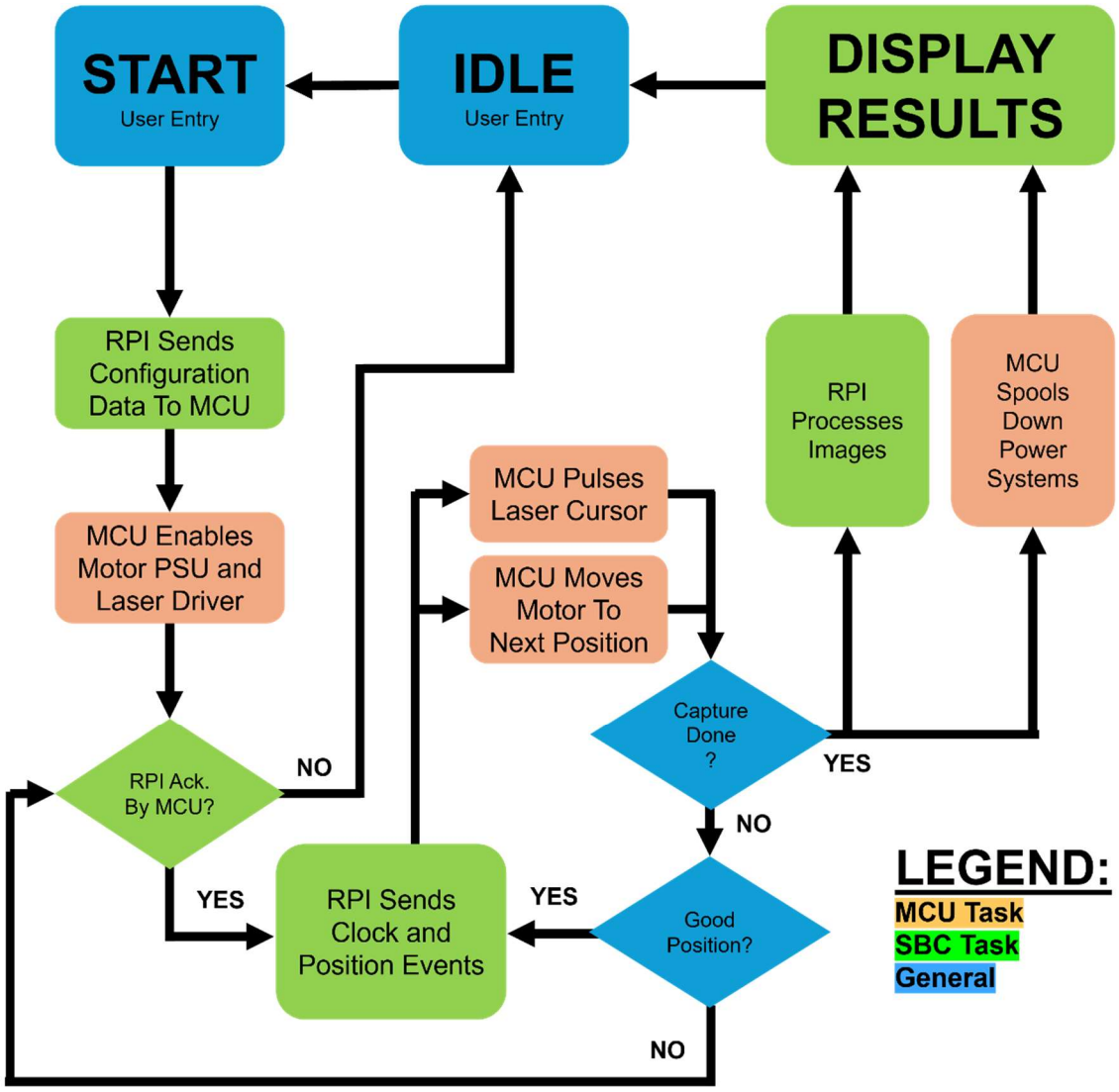


Figure 6: General software block diagram

Although this is a very general overview of the system operation, this serves as the master reference for the overall state-machine created by both the embedded computer (Raspberry PI model 4B) and the microcontroller (STMicroSTM32G431).

2.4 Project Features and Functionality

2.4.1 Overall System Functionality

Our objective is to design and fabricate a line-scanning hyperspectral imager. This instrument will be able to capture an image across a two-dimensional field of view, simultaneously recording an entire visible/NIR spectra per “pixel”. The camera will do this by limiting its field of view to a vertical line (parallel to the z-axis, which is

orthogonal to the mounting surface), and translating the sample below along the x-axis to capture a strip-image. A linear incident beam is used, as when it is passed through a diffractive grating, it will form a two-dimensional image on a given plane. This image, rather than containing two traditionally spatial dimensions, now includes only the z-axis dimension, with the other being wavelength. When illuminated onto an image sensor array, this image is now a per-pixel spectral representation of the incident beam into the imager. While the final image cannot be captured simultaneously, with a translating target, a faithful hyperspectral image can be produced, where any selected (z, x) pixel can have a full spectrum read from it. The entire device will be powered via an external AC/DC wall-adapter, feeding in DC power and connecting to a computer (laptop) via USB interface to stream captured image “cubes” for PC processing

2.4.2 Imager Optics

As we intend to use a commercially available CMOS image sensor for both cost and integration ease, we are limited to the spectral response of a typical commercial silicon pixel. For most CMOS image arrays (barring external NIR block filters), we can expect an effective wavelength bandwidth (above -10dB loss from peak) of roughly 350-1050nm. This depends on the image sensor used, but the optics described in this system will be designed to work across this visible/NIR target wavelength range. The imager subsystem will comprise of five major components, which are as follows: the imaging slit, our focusing lens pair, the diffraction grating (in a transmissive mode), our CMOS image sensor (at m = +/-1 order, off axis), and the mechanical substructure specific to aligning and translating these particular parts. The focusing lens pair will be a translatable positive and negative focal length lens. Through translation on a threaded stage, the pair will vary effective focal length from infinite far field down to 1 meter. The object distance will determine the focal length of the pair and will be manually focused. The diffraction grating will be a visible-NIR transmissive grating which will operate efficiently for our wavelength range. The CMOS sensor will collect the first order diffracted light from the grating.

2.4.3 Laser Cursor

To aid with the user experience of working with the instrument, as well as act as a functionally useful indicator, we find it necessary to design a laser “cursor” subsystem. This subsystem will be based on a commercially available red laser diode and will take the diode’s beam and expand it along the z-axis of the imager. The laser will be pulsed in such a fashion where the laser illuminates the target between successive captures along the imagers’ x-direction. With human persistence of vision, the imager would be able to display a scan-line as it images

the objective, without appearing in the image. The laser will be intensity pulse modulated 180-degrees out-of-phase with the pulse signal for the CMOS sensor's global shutter. The laser driver will be designed in such a way where it can receive a TTL pulse signal from the master controller.

2.4.4 Mechanical Structures

Our structures will be designed to be able to accurately translate the sample stage while maintaining tolerance and strength of the system under motion. A stepper motor will be used to provide torque to the translation stage's threaded rod with a tracked position from the microcontroller. If we have time in the design process, we can attempt to meet a stretch goal by achieving a measurable position via the use of a shaft encoder. Some of the structural material will be made as 3D-printed PLA components. For mounts, cases, and support components, this will give us the tolerance required while keeping cost down and iteration easy. Threaded components (i.e. lens translation) will be machined from metal stock, as 3D-printed PLA does not have the tolerance or low-friction required. The frame mechanical system will be constructed from commercial aluminum extrusion pieces in order to keep the system from being top-heavy and to keep the translation stage from moving erratically or with backlash. Aluminum extrusion is also advantageous as it is a standard part that can be easily constructed to mount all of our subsystems using corner brackets and the like.

2.4.5 Electronics Overview

While the electronics needed for this system will be working together as a whole, the electronics subsystem can be generally divided into responsibilities of both image acquisition/user interface, and power/control. Overall, both portions will be integrated using a serial communication interface between our single-board computer (SBC, Pi 4B), and a high-performance microcontroller to handle the non-imaging tasks. The MCU will simultaneously manage motor control, power monitoring, laser diode pulsing, all while staying in lockstep with directives from the SBC's UI program. The MCU chosen (STM32G431) has a rich peripheral set and therefore it is highly advantageous to use the hardware blocks in it to automate many tasks away from the limited ARM Cortex-M4 CPU. We may leverage memory transfer and computational hardware outside the CPU such as direct-memory-access (DMA) controllers and filter math accelerators (FMACs) to take advantage of these offered modern MCU peripherals and save on CPU time while communicating with the SBC.

Our electronics will be constructed as a multi-PCB stack, with the SBC on the bottom. The profile of the SBC's board will be followed, as well as utilizing the same 2x20 pin data/power header it provides. A logic/control board directly above the

SBC will host the MCU, laser diode driver, and motor driver components. External connections to the laser and motor driver will be made here, as well as connecting downwards to the SBC using the serial interfaces and GPIO lanes on the 2x20 connector mentioned. Our power electronics will deliver power and communicate with the MCU via other transfer headers located on the sides of the logic PCB. These power components are mounted above and on separate PCB(s) to aid in head dissipation and keep the layout of our logic board from being very cramped.

Our SBC will be delivered its 5V power feed via the 2x20 connector, transferred through an interface on the logic board. The SBC is a commercially bought system from the Raspberry Pi foundation, and interfaces to our image sensor and touch screen with plug/play a MIPI/CSI2 ribbon cable and HDMI (or USB) connection, respectively. The SBC's UI rendering code and program operational script will be written in a python framework and run on startup of the instrument. Most of the complexity on the side of the SBC is software led, communicating commands and requesting data from the MCU via the SBC's SPI interface. All hardware peripherals on the SBC such as the mentioned SPI and GPIO for timing signals are accessible through the Linux kernel drivers on the installed operating system distribution provided by the manufacturer.

For controls, positional accuracy of the translation stage is paramount, and directly impacts our spatial resolution along the x-axis. We want to utilize motor control electronics to "micro-step" our stepper motor to reach our target spatial (angular) resolution for the imager. This controller/driver will need to accurately deliver, and report time-varying current signals as produced by the MCU to meet this. Open-loop control of the motor position will be employed first, along with careful tuning of steps vs distance in the software. If this is not sufficient to produce resolvable images, then we will attempt to push for a closed-loop control using an external rotary encoder. For this, either hardware (FMAC) or software computation for the control law calculation will be done in real-time. As this is a stretch goal, we hope that this degree of control will not need to be pursued for the sake of software development complexity.

Our laser diode in the cursor subsystem will be driven with a constant-current circuit, with on-off keying (OOK) input from the MCU. As our MCU features an integrated 12-bit digital-to-analog converter (DAC), we can use this peripheral to output the voltage to the laser diode driver circuit that commands the peak pulse current to the diode. OOK modulation in the form of pulse-width-modulation (PWM) is to be employed using an on-board timer peripheral to pulse the cursor out-of-phase with the CMOS image sensors' frame-capture event.

2.5 System Specifications

2.5.1 Specification Value Table

Below is our system specification table, which sets the overall operational performance targets for the imager design. Three key performance specifications are highlighted as our demonstrable specifications for grading/evaluation.

Table 3: System Performance Specifications

Type	Specification	Value
Optical	Operational Wavelength Range	400nm – 850nm
	Spectral Resolution (FWHM)	≤ 10nm
	Spatial Pixel Count	1080 pix
	Spectral Channel/Pixel Count	1440 pix
	Entry Focal-Length Range	20cm – ∞cm
	Focused Spot-Size Diameter	2μm – 20μm
	Laser Cursor Profile (FWHM asp. ratio)	≥ 20:1
	Laser Cursor Operational Output Power	0mW – 20mW
	Laser Cursor Pulse Slew (10%-90%)	≤ 10μs (rise/fall)
Electrical	Time Needed for Full Scan/Render	≤ 10s
	System Power Consumption	≤ 60W (peak)
	Input Power Voltage Range	9VDC – 36VDC
	Imaging Sensor Type	CMOS
Mechanical	Instrument Weight (total)	≤ 10kg
	Instrument Size (tentative, maximum)	≤ 50x50x50cm
	Translation Stage Linear Resolution	≤ 0.2mm
	Translation Stage Speed	≥ 50mm/s

2.5.2 House of Quality Diagram

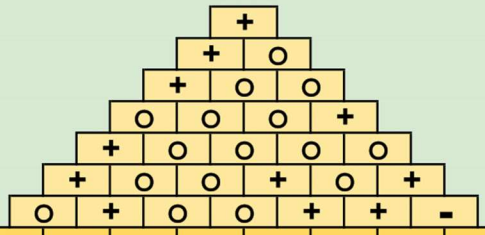
HOUSE OF QUALITY	Roof Corr.: + Positive, - Negative, O None								
TSUNAMI	**Technical Requirements:**	Instrument Size	Wavelength Range	Spectral Resolution	Spatial Resolution	Laser Power	Power Draw	Linear Speed	Linear Accuracy
Customer/Marketing Requirements:	***Goal:***	↓	↑	↑	↑	↓	↓	↓	↑
Affordability	↑	●	◒	◒	●	●	●	◒	◒
Transportability (Weight)	↑	◒	O	O	O	O	●	●	●
Overall Reliability	↑	●	●	●	●	●	●	◒	◒
Ease of Use	↑	●	●	●	●	●	●	●	◒
Ease of Maintnance	↑	●	●	●	●	●	●	●	●
Expandability of Optics	↑	●	●	◒	◒	O	O	O	O
Instrument Longevity	↑	●	●	●	●	●	●	●	●
Customization	↑	O	O	●	●	●	O	●	●
System Scan Time	↓	●	●	●	◒	O	◒	◒	◒
Requirement Matrix Correlation Key	**Target Value:**	≤ 50x50x50cm	400nm – 850nm	≤ 10nm (FWHM)	1080 pix	0mW – 50mW	≤ 60W (Peak)	≥ 50mm/s	≤ 0.2mm
Strong Correlation: ◒									
Medium Correlation: ●									
Weak Correlation: ●									
No Correlation: O									

Figure 7: House of Quality Diagram; see green sections for keys

2.6 Component Specifications

2.6.1 Imaging System Components

Table 4: Imaging System Components and Their Specifications

Component	Specification	Value
LA1509 Positive Entry Lens	Focal Length	+100 mm
	Diameter	25.4 mm
	Material	N-BK7, Uncoated
	Wavelength Range	350 – 2200 nm
LC1120 Negative Entry Lens	Focal Length	-100 mm
	Diameter	25.4 mm
	Material	N-BK7, Uncoated
	Wavelength Range	350 – 2200 nm
GT25-03 Transmission Grating	Grating Period	300 grooves/mm
	Size	25 mm x 25 mm
	Groove Angle	17.5°
	Wavelength Range	400 – 1100 nm
AC254-030-AB Imaging Achromat	Focal Length	+30 mm
	Diameter	25.4 mm
	Material(s)	S-BAH11 / S-TIH6, AR Coated
	Wavelength Range	400 – 1100 nm
Sony IMX296LQR-C Camera Sensor	Resolution	1456 x 1088 pixels
	Frame Rate	60 frames per sec
	Sensor Size	6.3 mm diagonal
	Pixel Size	3.45 µm x 3.45 µm

Housing	Material(s)	PLA / TPU, 3D-printed
	Color(s)	Black / Dark Blue

2.6.2 Laser Cursor Components

Table 5: Laser Cursor System Components and Their Specifications

Component	Specification	Value
RLD63NPC8 635nm Single Mode Laser Diode	Lasing Wavelength	635 nm
	Optical Power	20 mW
	Operating Current @P = 20 mW	65 mA
	Operating Voltage @P = 20 mW	2.25 V
	Beam Divergence	8° x 30°
	Beam Divergence Tolerance	±3° x ±4°
LA1859 Spherical Collimating Lens	Focal Length	+20 mm
	Diameter	18.0 mm
	Material	N-BK7, Uncoated
	Wavelength Range	350 – 2000 nm
LK1662L1 Cylindrical Diverging Lens	Focal Length	-50 mm
	Size	20 mm x 22 mm
	Material	N-BK7, Uncoated
	Wavelength Range	350 – 2000 nm

2.6.3 Electronic Components (integrated circuits only)

Table 6: Electronic Components and Their Specifications

Component	Specification	Value
TPS54541	Voltage – Input (Min - Max)	4.5 V – 42 V
	Voltage – Output (Min/fixed - Max)	0.8 V – 41 V
	Current – Output	5 A

	Frequency - Switching	100 kHz ~ 2.5 MHz
	Operating Temperature	~40°C ~ 85°C (TA)
TMC2209	Voltage - Supply	4.8 V ~ 5.25 V
	Voltage - Load	5.5 V ~ 29 V
	Current – Output	2 A
	Step Resolution	1 ~ 1/256
TPS55288	Voltage – Input (Min - Max)	2.7 V – 36 V
	Voltage – Output (Min/fixed - Max)	0.8 V – 22 V
	Current – Output	16 A
	Frequency - Switching	200 kHz ~ 2.2 MHz
	Operating Temperature	~40°C ~ 150°C (TJ)
AD8656ARZ	Slew Rate	11 V/µs
	Voltage – Supply Span (Min – Max)	2.7 V – 5.5 V
	Current – Supply	3.7 mA (x2 channels)
	Current – Output / Channel	220 mA
	Operating Temperature	~40°C ~ 125°C
Raspberry Pi Model 4b	Power Consumption	15W
	Voltage Input	5 V DC
	Current Input (idle, normal, max)	2.5 A, 3.0 A, 3.5 A
	Operating Temperature	0 – 50°C

Chapter 3 – Research and Investigation

3.1 Physical Optics Background

3.1.1 Paraxial Imaging

Our system must act as a standard paraxial imaging system, turning spatial information of an object at a distance into location along our detector. There is an entire scientific field's worth of solutions to this problem, but looking at standard telescope [6,8] and camera [7,8] geometries and keeping in mind that our light will have to collimated through the diffraction grating (discussed later), we will be designing an adjustable collimator and choosing and appropriate imaging lens (diagram shown in 2.4.4).

Optical formulas, geometries, and phenomena discussed can be referenced in [8]. A lens of focal length f acts to turn light coming from an object at distance d_o into an image at distance d_i by the relation

$$\frac{1}{f} = \frac{1}{d_i} + \frac{1}{d_o}$$

For positive focal length lenses, this leads to a real image formed past the focal length, and collimated light being imaged at the focal length. This means that our imaging lens (i.e. the one preceding the detector) should have the image sensor right at its focal length. For negative focal length lenses, this formula leads to a virtual image. Light entering the lens will be spread as if it came from an object located in the virtual image plane. This process is also reversible, i.e. light entering the lens that would be imaged at d_i will instead be focused at d_o. This leads to the possibility of an adjustable collimator.

Collimation means the negative lens d_{o-} is infinite, and therefore $d_{i-} = f$. To accomplish adjustable collimation one can place a positive lens in front of a negative lens with spacing $d_{i-} + f$ (keeping in mind that f is negative). This will cause the positive lens to attempt to image the light f away from the negative lens and therefore collimate it instead.

These considerations so far consider ideal lenses. Nonidealities come in three main forms: thick lens behavior, geometric aberration, and chromatic aberration. Thick lens behavior arises from ray travel within the glass of the lens. This leads to the formation of principal planes and potentially differing focal lengths from the surfaces of the front and back of the lens. The locations and specifications of these are either given directly by the manufacturer or can be calculated once given the material and lens geometry. Any lens sold without sufficient data to compensate for thick lens effects will not be utilized in this work.

The second main nonideality is geometric aberration. When using spherical lenses, which are by far the cheapest and most readily available geometry, ideal focusing can't be achieved. The paraxial approximation assumes a parabolic lens surface, so when other geometries are used the wavefront error order is r^4, where r is a normalized distance away from the lens center that the ray is passing. This aberration can therefore be minimized by keeping the apertures small and the surface radii of curvature large (i.e. longer focal lengths).

The third main nonideality is chromatic aberration. Differing indices of refraction at different wavelengths lead to different focal lengths. These aberrations are on the order of r^2. Techniques exist to remove them via matched compensating material choices. Our imaging lens, with a very short focal length, will need to be achromatic. The aberrations can also be minimized in a similar fashion to geometric aberrations, by keeping aperture small and focal lengths long. Calculations for all parameters given in this section will be given in Chapter 6.

3.1.2 Diffractive Gratings

Diffractive gratings utilize interference to produce off-axis beams of light. A grating consists of small, evenly spaced grooves in a transmissive or reflective surface. These grooves act to disturb the phase of the wavefront passing through. This phase disturbance is regular across the wavefront, and can act constructively at specific angles, producing beams of light off-axis from the original propagation direction.

The angles of the beams are related to the wavelength of the light. The relationship is given by the following:

$$m\lambda = asin(\theta_m)$$

where *m* is an integer and *a* is the distance in between grooves of the grating. As an example, for a grating of 300 grooves/mm and a center wavelength of 625 nm, one should expect first-order diffraction (*m = 1*) at *θ = 10.8°*, meaning that an image sensor will have to be 10.8° off-axis from the entrance imaging system. Typically, higher-order diffraction beams have lower powers, so most systems utilize first-order beams.

Many gratings also come blazed [9,10], where an angle is put into the grooves of the grating in order to minimize the power sent to the unused zeroth-order beam. In the Littrow configuration [8-10], this zeroth-order power can be eliminated. However, the Littrow configuration isn't necessary for operation, and power will be concentrated away from the zeroth-order beam even when light is directly impinging.

When integrating a grating into an imaging system, the output angle range must fit across the detector. As the detector is also matched to the imaging lens, the wavelength range and focal length are tied by the relationship (after linearizing sin(*θ) ~ θ):*

$$\Delta\theta = \frac{h}{f_i} = \frac{\Delta\lambda}{a} \rightarrow f_i = \frac{ha}{\Delta\lambda}$$

where h is detector size and f_i is imaging lens focal length. As an example, for a wavelength range of 400-1000 nm, a grating with 300 lines/mm, and detector size of 6 mm, the imaging lens focal length is 33 mm, which is fairly short considering aberrations. Coarser gratings allow for longer focal lengths.

3.1.3 Laser Diodes

Semiconductor laser diodes are a relatively modern technology. Laser diodes (LDs) enable the production of coherent optical light using just an electrical current as a source of energy. The laser resonator cavity is built using planar semiconductor fabrication techniques. The cavity is made within the high refractive index intrinsic semiconductor layer. Electrically, the diode is in a P-I-N structure, with p-type and n-type doped semiconductor materials sandwiching this intrinsic layer. Fresnel reflection via careful design of dielectrics around the cavity is used to reflect light from the cavity walls back along the optical axis.

As electrical current begins to flow through the diode, incoherent light begins to emit within the cavity, much the same way as an LED. Below a certain threshold current, the LD operates very similarly to an LED. Following traditional laser operation, ions in the intrinsic material are pumped into an excited state using this emitted light from the bandgap structure. After exceeding the threshold current of the LD, enough carriers are present to begin stimulated emission. At this point, any newly generated photons from the bandgap emission directly partake in the stimulated emission process. The spectral width dramatically shrinks along with optical power output increasing substantially.

Beyond threshold, the relationship between the coherent optical output power and the input current to the LD is almost entirely linear. This ratio is known as slope efficiency and is an important parameter to consider when designing the driver for this diode (covered in a later chapter).

Another very important quality to consider of most commercially available laser diodes is their exiting beam divergence. Most laser diodes due to their physical construction have a substantial divergence angle; bare diodes designed to emit to free space are nearly never collimated. In addition, divergence angles on both axes perpendicular to the propagation axis also are often not equal. As a consequence,

the exiting beam of the laser diode will often take the shape of a cone with an elliptical irradiance profile. To compensate for this, an optics train is usually put in place to collimate the beam on both axes perpendicular to the optical axis.

3.2 Existing Optical Device Characteristics

3.2.1 Visible-NIR Lenses

Existing lenses within our scope are made from common glass compositions and have spherical surfaces. Anti-reflection coated lenses exist, but these coatings have spectral responses that may interfere with the performance of our system. Aspheres exist, but limited options as well as a high cost of ~$500 mean that we are better off controlling aberration via aperture and focal length. We will need to use at least one achromat considering our wide wavelength range, but options for sale prevent us from using achromats for every lens. Uncoated N-BK7 lenses, such as the ThorLabs LA1509, have a transmission from 350nm to beyond 2μm, easily covering our system's spectral range. These lenses are also relatively affordable (~$25) and simple to evaluate. Below is a tabulation of lens options sold by ThorLabs.

Table 8: Visible-NIR Lens Selection Table

1" diameter lens options	**Uncoated N-BK7**	**Visible AR-Coated N-BK7**	**Aspheric**	**Achromat 400nm to 700nm**	**Achromat 400nm to 1100nm**
Available Focal Length Range	-100 to +2500 mm	-100 to +2500 mm	+20 **or** +50 mm	-100 to +500 mm	+30 to +200 mm
Wavelength Range	350nm to 2000nm	350nm to 700nm	350nm to 2000 nm	400nm to 700 nm	400nm to 1100 nm
Price	$25	$37	$280	$100	$120

In comparison, extremely limited options and high prices put aspheres off the table. Achromats that cover our wavelength range aren't offered with negative focal lengths, so only the final imaging lens can be achromatic. Considering collimation and a necessarily short focal length, this lens is the most important to be achromatic, and Zemax simulation (discussed further in Chapter 6) supports the theory when a single uncoated N-BK7 lens is traded for an achromat.

3.2.2 Diffractive Gratings

Existing transmissive diffractive gratings consist of either glass substrates precisely scored with grooves, holographic plates with groove-like structure embedded, or of plastic films with grooves impressed in. While plastic options are very affordable, the flexibility of such a film can lead to inconstant system behavior. Their affordability is therefore in part due to usage only within educational demonstrations rather than any commercial or research system. Holographic systems have low polarization dependence but are rather expensive and unnecessary for this project.

Reflective gratings exist with large grating periods and therefore allow for a longer focal length imaging lens and therefore less aberration. However, the sacrifice in field of view of a longer imaging system means that a balance must be struck that can be achieved with easier-to-integrate transmissive gratings.

Transmissive gratings that meet our scope, such as the ThorLabs GT25-03, are made on a rigid glass substrate (in the case of GT25-03, Schott B270). This grating has a spectral response range of 400-1100 nm, which our system's spectral range is within. The groove angle is 17.5°, so power will be concentrated away from the zeroth-order beam. Below is a tabulation of grating options sold by ThorLabs.

Table 9: Visible-NIR Lens Selection Table

Grating options	**Visible Ruled Transmission**	**Visible Holographic Transmission**	**Visible Ruled Reflective**	**Echelle Grating**
Available Lines/mm Range	300, 600, 830, 1200 lines/mm	300, 600, 830, 1200 lines/mm	300, 600, 830, 1200 lines/mm	300, 600, 830, 1200 lines/mm
Wavelength Range	400nm to 1100 nm	450nm to 750 nm	350nm to 1100 nm	UV-MIR
Price	$136	$524	$137	$278

The balance between field of view and aberration means that neither the holographic nor Echelle gratings are necessary. Ruled transmission and reflective gratings have very similar characteristics and prices, but transmissive gratings are simpler to integrate into an optics chain.

3.2.3 Laser Diodes

Existing laser diodes are typically either made to output into free-space from their package, or to feed into a pre-fitted fiber to guide the optical output to another fiber-input system. In our case, our laser diode will need to output directly into an optical train, and thus we will need a diode packaged for free-space output. Semiconductor lasers are rather sensitive devices, both to physical shock and electrostatic discharge (ESD). Their bare semiconductor dies also produce a substantial amount of excess heat that needs to be removed to maintain stable operation of the device.

These diodes are also prone to dust/moisture. Scattering and reflections from contaminants on their optical surfaces not only significantly harm optical performance, but can also result in burning of contaminants resulting in permanent damage. As a result, bare laser diodes of this type are typically packaged in metal cans to provide electrical and thermal connection. These packages also provide an optical window into the device, creating a protective sealed barrier around the diode. Some laser diodes (LDs) even come with a photodiode inside the package to provide external feedback of optical output power.

Various manufacturers produce packaged laser diodes and are typically sold where general optoelectronics are. Most optical device vendors have diodes available in various wavelengths, maximum output powers, packages, and other features. Our requirements state that we are looking for a laser diode able to output up to 100mW of peak output power and output a wavelength between 620 and 750nm for a red hue. With those considerations in mind, various types of can-packaged laser diodes were considered from Thorlabs available products:

Table 10: Laser Diode Selection Table

Laser Diode Options	**Diode-Pumped Solid-State LDs**	**Distributed Feedback LDs**	**Volume Holographic Grating LDs**	**Fabry-Perot QCL and ICL LDs**
Available Wavelength Ranges	375nm to 1653nm	1270nm to 1653nm	785nm to 976nm	3400nm to 9500nm
Max. Output Power	5mW to 2000mW	5mW to 130mW	270mW to 600mW	Up to 1000mW
Price Range	$21 to $5588	$98 to $1,016	$1,755 to $1,885	$1,718 to $4,584

Our design needs do not require our laser to have a particularly stable wavelength, single mode operation, or exact output power. Despite the great selection of laser diode technologies across a large wavelength range, we find a standard diode-pumped solid-state LD to be adequate for our design specifications.

3.2.4 CMOS Sensors

Existing CMOS sensors build on a long history of development from the inception of the digital image sensor. When sourcing commercially available CMOS detector arrays, a trade-off between pixel count and speed is typically considered. Pixel count, pixel density, maximum sensitivity, ADC resolution, shutter type, and maximum framerate are some of the major factors considered within sensor selection. CMOS detector arrays are typically packaged and sold either as the bare sensors themselves, or as pre-mounted PCB modules designed to be interfaced to the appropriate drive and read-out electronics. As bare CMOS sensors are incredibly delicate and quite challenging to design integrating electronics for, our team chose to source a CMOS sensor pre-soldered to a PCB module.

Image sensor arrays can generally be thought of as large arrays of photodiodes, able to each store a charge or voltage. In CCD arrays, only a charge is stored at each pixel, and each charge is manually shifted out and read-back using a single amplifier and analog-to-digital converter (ADC). While CCDs have better low-light performance and sensitivity, they typically are much more power hungry, read-out much slower, and cost more than CMOS sensor arrays. As CMOS arrays feature charge amplifiers per pixel, the read-out electronics can work much faster to pull pixel data off of the sensor. Unlike CCD sensors, CMOS sensors also often embed their own power conversion and control electronics, making them significantly easier to integrate in embedded applications. CMOS image sensors today are made at such a large scale for many devices, and as a result are dramatically cheaper and easier to source.

One important thing to consider with our design application is the difference between a sensor having a rolling-shutter, and a global-shutter. Much like a mechanical shutter, most CMOS sensors feature an electronically controlled shutter, which controls when a given pixel on the sensor will start and stop allowing incident light to collect charge. As with any imaging system, this timing and the pattern in which it operates influences the resulting image qualities dramatically. Rolling-shutter circuits on image sensors are most common as they are the easiest to implement and cheapest to fabricate. Rolling-shutters circuits not only read-out pixel data in a snake-pattern, but they also expose pixels in the same way. As a result, fast-moving subjects taken by a rolling-shutter sensor may appear very distorted in a twisting or rotating way. Global-shutter image sensors expose and

capture all pixel data simultaneously, reading out each frame pixel-by-pixel after each exposure. These types of sensors are less common and usually reserved for use in high-speed imaging or industrial applications. For our application, our sample will travel at a constant velocity underneath our imager. To avoid distortion in our imaging spectra, we may need to use a globally shuttered CMOS image sensor.

For our design, we are choosing to use the existing MIPI-CSI-2 camera interface hardware on our Raspberry Pi Model 4B SBC. MIPI, the Mobile Industry Processor Alliance (of various companies) standardized the electronic Camera-Serial-Interface (CSI) physical construction and protocol. This widely adopted interface is typically used to connect a vast number of CMOS sensor arrays to computing hardware.

Conveniently, the Raspberry Pi foundation manufactures 1st party CMOS image sensor modules, as well as develops and supports back-end software for their usage. As mentioned previously, these modules dramatically ease the integration of our sensor to our processing software and electronics. Without prefabricated CMOS modules and back-end software, the scope of our project would far exceed our available timeline. Below is a table detailing the performance and specifications for available CMOS image sensor modules from Raspberry Pi:

Table 11: Image Sensor Selection Table (1st party)

CMOS Module Options	**Camera Module v.3**	**AI/CV Optimized Module**	**High-Quality (HQ) Module**	**Global Shutter (GS) Module**
Image Sensor	Sony IMX708	Sony IMX500	Sony IMX477	Sony IMX296
Pixel Count	11.9Mpixels	12.3Mpixels	12.3Mpixels	1.58Mpixels
Resolution (pixels)	4608 × 2592	4056 × 3040	4056 × 3040	1456 × 1088
Sensor Size	6.45mm × 3.63mm	6.287mm × 4.712 mm	6.287mm × 4.712 mm	5.023mm × 3.753mm
Pixel Size	1.4µm × 1.4µm	1.55 µm × 1.55 µm	1.55 µm × 1.55 µm	3.45 µm × 3.45 µm
Integrated Lens(es)	Yes, but removable	Yes, but removable	No, but mount provided	No, but mount provided
Price	$25	$70	$50	$50

Despite all specifications found, we immediately observed that 1st party hardware only supported a single global shutter enabled CMOS sensor module. We did investigate the potential of using 3rd party hardware, but due to the potential software complexity needed to implement Linux-kernel support for other sensors, we chose to proceed with the IMX296 global shutter module.

As all of these sensors feature an RGB color filter above their detectors, digital summation of all spectral responses will be necessary to monochromatically capture our images. We will need to do this to observe our spectra on each of our spatial pixel columns. The Raspberry Pi foundation provides a very rough spectral characterization of the pixels on the IMX296. As the *full* datasheet for the IMX296 is restricted by Sony, **our group will need to conduct a spectral characterization of the IMX296 pixel for our target wavelength range**. As a rough estimation towards our spectral responsivity, we've included their chart for reference:

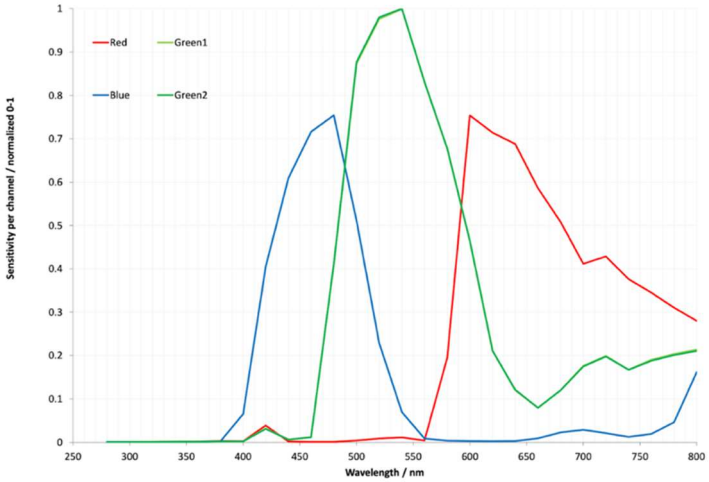


Figure 8: IMX296 Global Shutter Camera Module per-pixel spectral responsivity (provided by the Raspberry Pi foundation)

3.3 Digital Electronics and Interfacing

3.3.1 Embedded Platforms (MCU vs FPGA vs SBC)

Computation and control in the embedded world is now more diverse than ever before. The need for embedded platforms has exploded in recent decades due to the rise of many portable technologies such as the smart phone/watch, IOT enabled appliances, defense applications, and research technologies. Small, portable, and power efficient systems are in high demand for this application space, and the parts engineers choose to implement them is of considerable importance to our project as we plan to employ two (MCU and SBC). Three very important factors to consider when choosing an embedded device are power consumption, peripheral needs, and data throughput/processing power. Microcontrollers (MCUs), Field Programmable Gate Arrays (FPGAs), and single-board computers (SBCs) each address these needs with their own strengths and weaknesses.

Embedded solutions can generally be split based on their needs between processing power and control authority. For example, MCUs excel in enabling precise control over a large number of direct-to-hardware peripherals, but tend to have rather limited ability to do tasks with heavy computational burden. MCUs can and do excel with real-time processing of direct hardware, and are a great choice for interfacing to low-level components that make up the motor controllers, power supplies, function generators, timers and other integrated hardware products. MCUs would not be suitable in an entirely software dependent environment, due to their much lower clock speeds and smaller CPUs, usually featuring 32-bit architectures at *most.* For our application, **MCUs** present a great way to establish an embedded monitoring and control node for our power and control electronics subsystems.

FPGAs are a particularly interesting class of embedded devices as their architecture is usually completely user-definable (barring large modern FPGA in SOC systems). FPGA design engineers design combinational or sequential digital circuits in software using hardware-description-languages (HDLs) in an onion of layers representing a large design hierarchy. Tiny circuits group into bigger ones, and in very large-scale integration (VLSI) design, these digital circuits can be grouped into massive structures such as CPUs, FFT computation blocks, digital filters, RAM, and complex ethernet control peripherals, or otherwise. FPGAs are much more complicated to design on and implement than an MCU or SBC, but they pay dividends when used in situations where large numbers of parallel tasks are needed. FPGA design engineers, limited by the number of logic-elements on

their chosen part, can design custom digital circuits to heavily accelerate large scale parallel computation and control where MCUs and SBCs can't.

SBCs are a modern development born out of the need to implement operating system (OS) running microprocessors for embedded applications with high-computational load. In recent years, many independent companies have developed single board designs for providing power, RAM chips, RF hardware, camera/display interfaces, external standard I/O connections, and even external direct GPIO to microprocessor systems. SBCs have found a great application space as a way to integrate simple embedded hardware with the power of an OS and large CPU. For our application, the software load of hosting an interactive UI, rendering images, and reading data from a camera module is absolutely above the data throughput limit of an MCU, making an **SBC** great for this application.

Using an MCU and SBC, we now look back at our system requirements and specifications to guide us in our selection process of each. MCUs and SBCs are made by many manufacturers, and engineers will choose which particular unit better suits their application based on this, and also their team's prior experience as many of these products are very different to use based on the manufacturer. Below we tabulate the specifications relevant to our design of our final three chosen MCUs and three SBCs, selecting a part afterwards:

Table 12: MCU Selection Table

MCU Options	Espressif ESP32-S3	STMicroelectronics STM32G431RBT6	Texas Instruments MSP430FR6989IPN
CPU Core	Xtensa LX7 *x2*	ARM Cortex M4 *x1*	RISC (16bit!) *x1*
Max Clock	240MHz	170MHz	16MHz
ADC Specs	2xSAR, 12bit	2xSAR, 12bit	1xSAR, 12bit
DAC Specs	**N/A**	4x, 12bit, 1MSPs	**N/A**
Timers	4x54bit	10x16bit, 1x32bit	5x16bit, 7CCR/each
Digital Comms	2xI^2C, 4xSPI, 3xUART	3xI^2C, 4xSPI, 4xUART	1xI^2C/SPI, 1xSPI/UART
Development Complexity	Significant	Moderate	Low
DMA Channels	5 TX, 5 RX	6 TX/RX x2, 12 total	3 TX/RX
Price	$1.85/1 unit	$5.60/1 unit	$11.20/1 unit

Between the three previously tabulated MCUs, the ESP32-S3 clearly outperforms both the STM32G431 and the MSP430 across the board, besides the integrated DACs. It may be tempting to resort to the use of a higher performance and cheaper MCU, but we chose to stick to a simpler system as having more control over less peripherals would lend itself to easier development in firmware. The MSP430 series MCUs, while familiar and approachable, are rather sub-par in compared to more modern offerings such as those from STMicroelectronics or potentially newer TI parts. We ultimately chose the G431 as not only did the person responsible for firmware have extensive experience on the chip, but also due to the simple fact that its performance as a mixed signal high performance MCU perfectly aligned with our motor driving and control needs. The on-board ADC suite and especially the DAC are very well integrated in their software interface and makes firmware development simple, although more complex than an MCU with a much more limited architecture such as the FR6989. As mentioned, the relevant specifications for the SBC choice is also tabulated for our final three entries:

Table 13: SBC Selection Table

SBC Options	**Raspberry Pi Model 4B**	**Raspberry Pi Model 5**	**Pine64 RockPro64 4GB**
CPU Cores	4xARM Cortex A72 @1.8GHz	4xARM Cortex A76 @2.4GHz	2xARM Cortex A72 @1.4GHz, 4xARM Cortex A53 @1.8GHz
Power Consumption	15W	15W	36W
RAM Size	4GB (selected)	4GB (selected)	4GB, LPDDR4
Networking	Gigabit Ethernet + WiFi/BLE	Gigabit Ethernet + WiFi/BLE	Gigabit Ethernet
External I/O Connections	4K Digital Video, 2xUSB 2.0, **2xUSB 3.0**, MIPI-SCI, 2x20 GPIO header	4K Digital Video, 2xUSB 2.0, **2xUSB 3.0**, MIPI-SCI, 2x20 GPIO header	4K Digital Video, 2xUSB 2.0, **1xUSB 3.0**, MIPI-SCI, 2x20 GPIO header
Manufacturer Support	Long-term device support, 2034	Long-term device support, 2036	Long-term device support, **2027**
3rd Party Software	Significant	Moderate	Limited.
Price	$55.00/1 unit	$66.00/1 unit	$80.00/1 unit

While the Pine64 offered SBC was definitely a promising lead from its raw compute power and similar physical specifications to the other two, its power consumption and lack of support made it less compelling. This, along with price and the 1st party CMOS image sensor offerings of the Raspberry Pi foundation pushed us to compare the newer model 5, and older model 4B. The Pi 5 boasts a more powerful processing system, equal and better peripherals, and similar power consumption metrics as the 4B. However, it being relatively newer and featuring an MCU on it that we would not be using made the choice bounce back towards the 4B. As the 4B has been on the market and used by many engineers and hobbyists, its 3rd party software offerings for various hardware and software extras are much more developed and prevalent. We therefore chose the 4B.

3.3.2 Microcontroller Peripherals

Our microcontroller includes various peripherals that make the IC interact with the circuit in various ways. Some of these peripherals include UART (Universal Asynchronous Receiver and Transmitter), SPI (Standard Peripheral Interface), I2C (Inter-IC Communication), timers/counters, ADCs (analog to digital converters), GPIO (General-Purpose Input/Output) pins, Watchdog Timer, DMA (Dynamic Memory Controller), and much more. Some of these peripherals will be essential and others will be nice to have or not useful for our application.

Communication between computers and our microcontroller is essential to tell exactly what the microcontroller should do and send out. UART is one of our standard communication protocols before USB and isn't used very much. Our communications will mostly be focused on SPI and I2C. These peripherals facilitate the possibility of connecting various external ships to one communication bus. This helps offload the microcontroller from doing everything, and will let the Raspberry Pi by just asking the Pi to send the data through these ports rather than processing the data itself.

Timers and counters are typically used to track time within our microcontroller allowing us to control multiple areas with one global clock or specific clocks. We can use these timers and counters for PWM (Pulse-Width Modulation) which uses a digital signal to control the average power to our analog device using fast signal switches.

ADCs lets us read an analog voltage into a digital number to use as a value/logic for various processes.

GPIO pins are our simple on/off communication that reads a button or turns on an LED. They can provide more complexity depending on the application such as pull-up/pull-down and slew rate control.

Our Watchdog timer makes sure that our main CPU doesn't get stuck on bad code such as recursive code that is useless and hogs processing power. If, per say, the timer isn't counted, it will restart the program execution from the beginning. It typically isn't necessary; it's good overhead to make sure that any bugs that slip though doesn't make the program hang unnecessarily.

The DMA allows for high-speed data transfer between peripherals and memory without asking the microprocessor once more, freeing it up to do more computation. DMA can be set up to buffer received communication in a larger area of memory, allowing it to be processed when the microprocessor is ready. There's also a DMA request router (DMAMUX) that can route the DMA control lines between the peripherals and the DMA controllers of the product.

All of these peripherals are essential to make sure our code interacts with our hardware effectively. The crucial communications, SPI and I2C, allows us to transfer data and code our microprocessor to process the data it's taking in. GPIO pins give us extra control to interact with our Raspberry Pi or potentially physical components. Our DMA gives the high-speed data transfer while alleviating the CPU resources to focus mostly on processing and overall computation.

3.3.3 Digital Communications

Our project primarily concerns the use of three embedded digital communication interfaces; Inter-Integrated Circuit (I^2C), Serial-Peripheral-Interface (SPI), and Universal-Asynchronous-Receive-Transmit (UART). Referring to the hardware diagram presented, the following communication paths will need to be established:

- We will need to run an I^2C interface to communicate from the MCU to the motor driver PSU
- An SPI interface (along with some timing signals) is needed to send/receive commands from the SBC to the MCU
- A single-line UART interface can be used to potentially write to registers within the motor driver IC's configuration

I^2C Technical Briefing:

I^2C is a popular/widespread protocol and physical implementation used across many electronics applications. I^2C is a rigorously standardized protocol, enabling low-speed communications across a large number of devices connected to only two lines. As a synchronous protocol, I^2C features a serial-data line (SDA) and a serial-clock line (SCL). A clock signal typically between 100kHz (standard mode)

and 1MHz (fast mode plus) is sent upon the start of a transaction to enable devices on the bus to shift in and out data within their hardware peripherals. Notably, I^2C is a half-duplex protocol, meaning that master and slave devices may only communicate one-at-a-time on the bus. Failure to meet this part of the standard may result in bus crashes, a highly undesirable consequence of this physical implementation

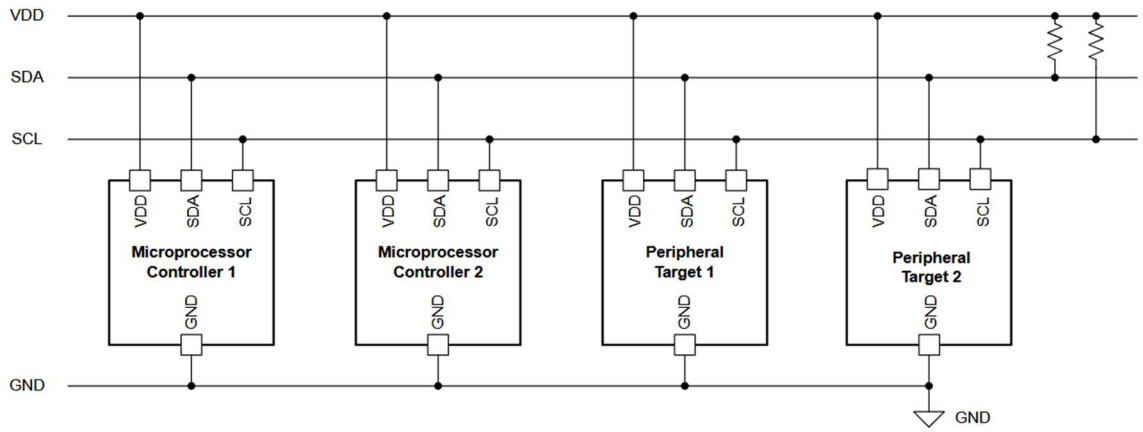


Figure 9: I2C hardware interface, taken from Texas Instruments' "A Basic Guide to I2C," by Joseph Wu

I2C unidirectionally sends its SCL signal from the active master on the bus (there can be multiple masters), and transmits/receives on the SDA line bidirectionally. The SDA line is implemented as an active-low signal for data entry. This line is pulled high by default, which enables peripherals to safely pull down these lines with internal low side switching semiconductors.

SPI Technical Briefing:

Within the Serial-Peripheral-Interface (SPI), data transfer is approached as a full-duplex transfer, where both a master and slave device are capable of transmitting data to each other simultaneously. Despite common misconception, SPI as a protocol isn't strictly defined, and varies between manufacturer implementations. As SPI is a very simple protocol, some parts of the standard such as clock polarity, phase, and exact timing are left to be interpreted by the board designers and IC designers. Most commonly seen are the Motorola SPI frame standard, and the Texas Instruments (TI) frame standard.

The STM32G431 allows for a 4-16bit data word with not only selection to frame standard, but also control over clock phase and polarity.

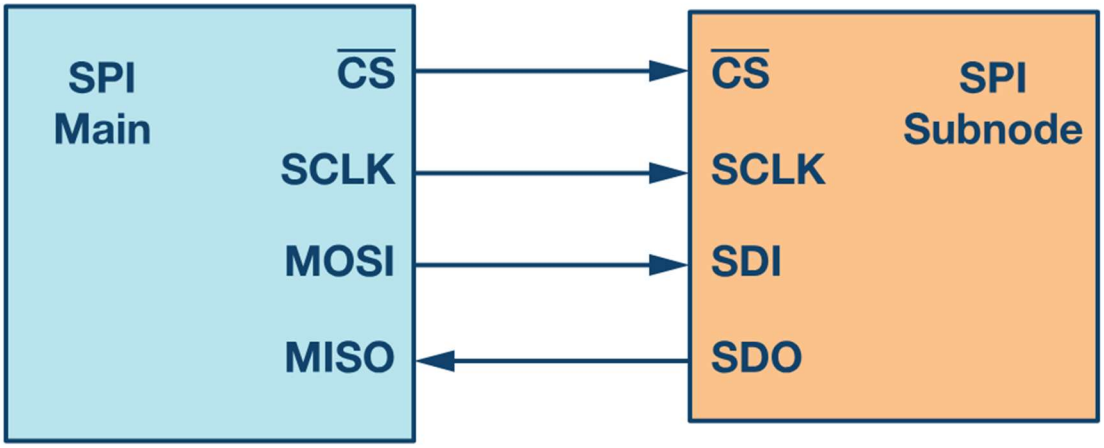


Figure 10: SPI hardware interface, taken from Analog Devices' "Introduction to SPI Interface," by Piyu Dhaker

UART Technical Briefing:

The UART standard is an older and very widespread communication standard adopted to meet the needs of serial communications in low pin-count applications. As UART is asynchronous, clock recovery and expectation of baud (bit) rate is needed from devices linked to function properly. UART is very sensitive to start/stop conditions as well as the defined baud rate of the link. In general, UART transmissions can be limited to byte messages with a start bit, data byte, parity bits (optional), and a stop signal. The protocol is very easy to implement, with either TX/RX direction only needing a single wire to transfer data.

Care must be taken to ensure the UART implementation, such as the one between our MCU and the motor driver IC, is carefully set-up and tested.

An example UART transmission frame is provided below for refence to the above hardware and timing description:

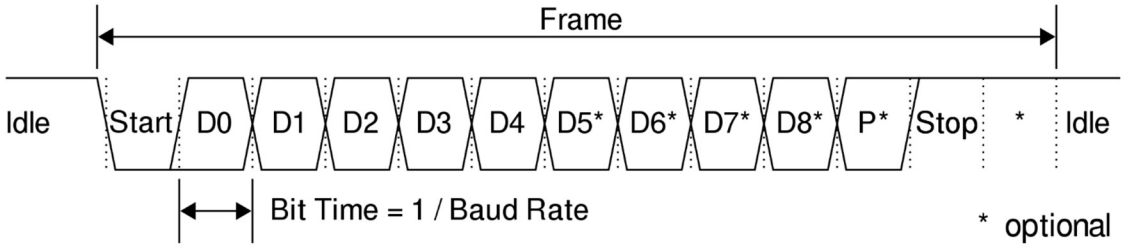


Figure 11: UART hardware timing diagram, taken from the Digilent Blog "UART Communication Explained," by Ian Etheridge

3.4 Power and Control Electronics

3.4.1 Power Delivery and Requirements

Several elements in our PCB require power delivered through specific voltage regulation circuits. The system we implement must be equipped to handle all specific voltage levels and power ranges, enabling them to run safely and stably. It should also have methods to reduce the voltage from the main power source and limit the current to components to prevent short-circuiting our parts. Typically, we will have a much higher input voltage coming from the power adapter, in which the buck converters and linear regulators will bring down the voltage to levels required, mainly 5V and 3.3V, with little ripple.

Table 14: Power Management Comparison

Component	Power Consumption	Operating Voltage
Raspberry Pi 4B	~3W(Idle) ~10W(Moderate Load) ~20W(Heavy Load)	5.00V - 5.25V
Motor Driver	10W - 58W (when on)	5V - 29V
Microcontroller	330 mW	1.8V – 3.6V (3.3V nom.)

The proper power adapter is going to be crucial and will impact our performance, stability, and reliability of the whole system. Our adapter is going to be our primary source of power that will provide all the correct voltages and currents to our main components. A lackluster adapter will cause an erratic power supply, voltage sag, and a reduction in overall lifespan for components, leading to more failures and problems. Our voltage needs to not only provide specific voltages and currents but also provide overcurrent and short-circuit protection for our differing demands in our system.

3.4.2 Energy Sources

Our selection of a DC power jack was critical in ensuring reliability, safety, and overall long-term functioning of the system, especially in demanding power consumption tasks such as encoding and decoding the photo. Our power jack is the primary interface between our external power adapter and the general power management system of the PCB. It also needs to handle the total current load and maintain a secure connection when gathering photos. A poor power jack leads to inconsistent voltage/current rating, which can potentially overvolt our system and

ruin our precise motor movements for photos. Our power jack also needs to physically fit the plug size of the power adapter, while also rating for continuous high current that will supply the power-hungry Raspberry Pi 4b, stepper motor, and other required components.

Table 15: Adapter Model Comparisons

Model	Output Voltage	Output Current	Power Rating	Plug Size (mm)	Cost
HQ-60W-12V	12V	5A	60W	5.5x2.5	$14.15
TOBWOLF DC AC Wall Outlet	12V	5A	60W	5.5x2.5	$9.99
Arybroourd AC Adapter Power Supply	12V	8A	96W	5.5x2.5	$18.23
ALITOVE Power Supply	12V	10A	120W	5.5x2.5	$20.00
Facmogu Power Supply	12V	10A	120W	5.5x2.5 5.5x2.1	$19.99

The Facmogu power supply is a great choice for our system due to the balance of price, high current throughput, stability, and overall reliability needed for supplying power to all of our parts. A constant supply of 12V at 10A allows the adapter to provide a theoretical total output of 120W, allowing a sufficient headroom to run heavy loads for multiple high-power components, such as the Raspberry Pi 4b, microcontroller, our motor driver, and our CMOS camera. Although we do not intend to draw more than 60W at peak, having extra headroom improves the safety of the system if the instrument were to fail at fault mitigation.

3.4.3 Physical Interface for Entry

Wall Outlet:

Our simplest and reliable source will be a wall outlet, supplying a steady and uninterrupted supply of power. In the system, the adapter is plugged into an outlet and will guarantee it’s powered throughout the capture sequence. It also eliminates the dramatic voltage drop or loss of power associated with small-scale energy storage. The capability to include voltage regulators to change the voltage levels also allows us to accommodate our specific needs of our Raspberry Pi and our

MCU. This can also allow continuous scanning and decoding to provide a more accurate photo.

Batteries:

Batteries as a power source would provide limitations in our system in consistently powering our power-hungry components, while demanding tasks are undergoing, such as taking and processing the pictures taken. While we can make it more mobile with batteries for more remote applications, the lack of energy storage makes consistent recharging and upkeep, which could be problematic during the photos being produced. Since we also want a compact setup, making a sort of “hat” for the Raspberry Pi, batteries make for a bulkier and heat-prone system that can interact poorly. The rechargeable, high-capacity batteries also have the limitation of a limited cycle life, which decreases our overall lifespan/reliability.

Power Banks:

A power bank, seemingly a compromise solution, also suffers from similar problems with the batteries. Even with their higher capacities than regular batteries, the constant long-term power needed for analyzing and measuring photos poses problems, such as recharging frequently, adding more heat to our system than batteries, and being extremely bulky for our setup. We could use them as backup sources, when power could be spotty at certain locations, but the consistent power source of a wall outlet is much more reliable and stable for our applications. The wall outlet as our power source allows our system to benefit from consistent voltage, power, and almost no downtime that comes with recharging/replacing the power banks and batteries, respectively. This selection allowed us to take in the power and step it down to a DC signal, providing all the power needs of our components. Power banks and batteries allow us to be more portable, but the leading downtime that comes with them, while also being bulky/heat-prone, makes the wall outlet a better source for safety and reliability purposes.

Table 16: Power Source Comparison Table

Power Source	Energy Density	Limitations	Best For	Cost
Wall Outlet	Consistent Energy	Must be consistently plugged into a wall/near a wall outlet	Consistent power throughout the circuit for prolonged photo	$19.99 for power jack

			processing/maintaining power	
Batteries	50-150 Wh/kg	Replacing/charging batteries frequently rather when prolonged power usage is needed	Portability and low power applications	$10 ~ $50 depending on how often replacing/charging is required
Power Bank	150-250 Wh/kg	Mixed use of decent energy and portability, however frequent charging and potential heating of our compact circuit	Portability with slightly higher power applications, but not prone to overheating	~$30

With these comparisons in check, the obvious choice is the wall outlet as our physical interface for powering our device. With our consistent need for power for the power-hungry Raspberry Pi 4b and our MCU, a good supply of power helps in both making sure our device stays on but has a good flow of current with no random spikes. Going forward, we take into consideration the power jack/wall outlet into our design and provide accommodations like space and keeping our device compact.

3.4.4 Power Converter Topologies

The right voltage regulator is essential to ensure that our parts are running at the particular voltage they need with enough current, supplying consistent power. Our system power is being provided by the high-voltage DC power jack and the wall outlet, meaning each device will require a regulated/stable voltage to ensure everything functions normally. Wrong or inconsistent voltage regulation components can lead to voltage fluctuations, improper processing of data, or even permanent damage by over-volting our core components.

Linear Regulator:

A linear regulator is a simple device that maintains a constant output voltage, dispersing the rest of the voltage as heat. This component varies its resistance, continuously adjusting that voltage divider network to maintain a consistent and constant output voltage. It’s well-suited for noise-sensitive applications due to its

low output noise and accurate transient response. However, due to the voltage regulators needing their voltage difference to be dissipated in the form of heat, this makes them quite inefficient for cases where the input and output voltage difference is high. Their ease of use and make allow them to be greater for low-current applications in certain scenarios, where efficiency and heat-generation aren't too much of a concern. These would be useful in connecting components, where power input won't be dramatically higher than power output, but our main distributor of power will have to be more efficient.

Switching Regulator:

A switching regulator is an active device that converts one voltage level into another with high efficiency using components such as inductors, capacitors, and, at times, transformers using high-frequency switching. Instead of dissipating the voltage as heat, it instead rapidly turns the input voltage on and off while storing that energy in the inductor or capacitor, then releasing it to maintain that stable output voltage. This allows us increased efficiency, with high-current and high-voltage drops, which is most suitable for applications in battery-powered systems. However, switching regulators tends to be more complicated, while introducing high-frequency noise and even electromagnetic interference, creating caution and precision when employing it in the circuit. Different types of these switching regulators are buck and boost regulators. Our system is powered through an adapter sourced to a wall outlet, requiring both buck and boost regulators to efficiently manage the voltage levels needed for our entire circuit.

Buck Regulator:

A buck regulator is a step-down converter, where the input voltage is lowered to a reduced stable output voltage. This circuit is essential to our system, since it will be the main device powering our Raspberry Pi, our MCU, and our motor due to specific voltages ranging from 3.3V to 5V. The efficiency required to make our system consistent is easily achieved by the buck regulator, by switching instead of dissipating the energy. This allows our system to be more reliable and optimal in our power.

Boost Regulator:

Conversely, a boost regulator, or step-up converter, has the opposite function in that it raises the input voltage to the required value for a component that operates within a higher voltage than the power supply supplies. This circuit may help any power-hungry components where the input voltage may not be sufficient for our needs without requiring more power adapters for different scenarios.

Table 17: Power Supply Topologies Comparison

Power Supply Topology	Input/Output Voltage Relationship	Efficiency	Complexity	Size
Linear Regulator	0 to 1	35% - 40%	Low	Small (SMD)
Buck Regulator	0 to 1	85% - 95%	Moderate	Medium (50mm x 100mm)
Boost Regulator	1 to infinity	80% - 90%	Moderate-High	Medium (50mm x 100mm)

3.4.5 Power Component Selection

Finding ICs That Meet Our Needs

Our design requirements specify that our main power source is nominally a 12V DC feed, with enough output current to drive our subsystems. We face the challenge of designing power supply systems to feed our SBC, our motor driver circuitry, and our microcontroller/digital logic. In addition, our motor driver circuit is specified to be designed in such a way that the maximum output current and output voltage are programmable via interface to the MCU.

As we find our circuits to be in a mildly space-constrained situation, higher switching frequencies (>100kHz) are desired as inductor sizes can shrink dramatically. With up to a 5A peak load current for our highest load regulator, low switching frequencies would already increase the size of our needed inductor quite significantly. To ensure higher probability of success and to save on space, an integrated FET switch is desired within our buck regulator IC. Most modern buck regulator chips feature efficient internal switches with low conduction losses, and very good heat dissipation from their specially designed packages.

IC size is another design consideration, and the smaller the regulator, the more applicable the IC to our uses. Obviously, certain packages such as BGA that feature difficult to verify connections aren’t desirable for student assembly and troubleshooting. Packages with exposed pads (EPs) for thermals are also rather attractive as large, multi-via connections to the internal ground planes of a board can aid significantly in the sinking of any waste heat from the IC. Considering our large current demands, this feature is paramount.

SBC Power Supply:

Our primary objective when it comes to delivering power to our SBC is to maintain tight control over its 5V supply, with large dynamics in the load current. As our SBC's power consumption will vary drastically with the operations and programs running on it, we will expect to see large load transients in regular operation. Stepping down from 12V to 5V immediately suggests the use of a buck, switch-mode regulator. As we have a voltage drop of 7V and a supply current of up to 5A at peak (3A max nominal), a linear regulator would be completely unsuitable for this application.

Reviewing our initial considerations and our specific objectives for delivering power to the SBC, we browsed TI's integrated-FET buck regulator catalogue and decided to pick between three different ICs.

Table 18: SBC PSU IC Selection Reference

Buck IC Options	**MAX20406AFOB**	**TPS54541**	**LMR51450SDRRR**
Input Voltage	3V – 36V	4.5V – 42V	4V – 36V
Out. Voltage	0.8V – 10V	0.8V – 41V	0.8V – 28V
Current Limit	6A (continuous)	5A (continuous)	5A (continuous)
Operational Efficiency	**~93%** (12V to 5V @3A load)	**91%** (12V to 5V @3A load)	**~94%** (12V to 5V @3A load)
Thermal Res. ($R_{\theta JA}$)	38.4°C/W	**35.1°C/W**	47.4°C/W
Switching Frequency	400kHz – 3.0MHz	100kHz – 2.5MHz	200kHz – 1.1MHz
Package	17-PowerWFQFN	10-WSON (4x4mm)	12-WSON (3x3mm)
Protection Features?	Overtemperature, Short-Circuit	Overvoltage, Undervoltage	Overtemperature, Short-Circuit, Undervoltage
Price	$4.36/1 unit	$5.14/1 unit	$2.03/1 unit

The TPS54541 was eventually chosen as the best overall (ignoring cost), able to regulate our 9V – 36V input voltage, and deliver the needed maximum continuous current of 3A. The TPS54541 also notably had the best thermal resistance of the three, very important for our high current application. We chose to balance our

solution size as our highest design priority, while maintaining careful consideration on thermal performance.

Motor Controller PSU: TPS55288

For supplying power to our motor driving circuit, we found the ability to digitally control the voltage and maximum current to the driver IC to be a nice feature to have. This would allow for us to directly control the maximum energy allowed to be fed to the motor, very useful for limiting power draw from the source in less demanding regimes of operation.

Buck regulator ICs that met our design requirements for our motor control circuitry (5V to 18V, up to 10A peak current) were easy to find, **but adding the ability to digitally control this made finding a part much more difficult**. We initially considered digitally controlling the gain of the feedback path of an existing regulator. After careful consideration, we decided that implementing such a technique would be far too risky and take too much time to develop as custom circuitry.

With a bit more research, we discovered that TI offers **one** digitally controllable buck regulator that suits our exact application in this voltage/current range. ICs easily source-able in the USA that meet these needs are extremely limited. The TPS55288 is uncommon in its extensive on-board register table and digital control over many parameters. A buck/boost IC, it features an I2C interface, allowing designers to digitally adjust the gain to its feedback path, enabling precise voltage control from 0.8V to 22V with 20mV steps. This buck-boost topology also allows for us to experiment with feeding higher voltages into the motor driver IC, beyond our 12V input.

Very importantly, the IC also offers an internal high-side current-sense amplifier, which can be fed into the voltage control circuitry to reduce the output voltage on overcurrent. This allows for an effective clamp on maximum current draw, which is highly desired for ensuring safe operation of the motor and actively fusing supply current to the motor driver IC. The design considerations around this power management chip and all others are followed up in chapter 6.

Digital Logic PSU:

Our digital logic circuitry, primarily concerning the microcontroller, needs a relatively low supply current (<500mA) at 3.3V. While our SBC features an internal 5V to 3.3V regulation, we chose not to use it for a few reasons. As to one in particular, we wanted to specifically allow for our custom circuitry to run independently of its connection to the SBC. This would enable us to test and debug the system without having to have the PCB plugged into the SBC.

Due to the existence of our 5V rail to power the SBC on startup, we chose to implement a simple linear regulator from the 5V rail to our 3.3V logic voltage. As a voltage difference of only 1.7V was needed to be dropped, paired with our limited supply current needs and desire to stay compact, a commonly available low-dropout linear regulator (LDO) was chosen. Below is a tabulated selection guide of four LDOs we considered in our component selection. After consideration, we proceeded with the LD1117S33CTR.

Table 19: MCU PSU LDO Selection Reference

MCU Power Supply Options	**LD1117S33CTR**	**TLV1117-33IKVURG3**	**TLV1117-33CDCYG3**	**LD1117D33CTR**
Voltage – Input (Max)	15 V	15 V	15 V	15 V
Voltage – Output (Min/Fixed)	3.3 V	3.3 V	3.3 V	3.3 V
Voltage Dropout (Max)	1.3 V @ 800 mA	1.3 V @ 800 mA	1.3 V @ 800 mA	1.3 V @ 800 mA
Current - Output	800 mA	800 mA	800 mA	800 mA
Thermal Resistance junction-case (SOT-223)	15°C/W	57.9°C/W	55.6°C/W	15°C/W
Cost	$0.32	$0.96	$0.91	$0.66

3.4.6 Motor Selection

Our motor is critical in our system to provide complete environmental observation when observing soil patterns. Moving the optical camera using the motor allows the system to get a wide field of view and also captures every element of the soil. The motor must also operate with high accuracy and efficiency because proper positioning is necessary to ensure that the camera is capturing a steady and constant feed. The motor will be part of our system, taking more of the bulk in terms of size, but also that lets us be space conscious to make sure the motor is pulling too much of the load.

Our system is going to move across a linear set to make sure we can get a wide enough field of view. A 360-degree view is nice but causes the photo to only capture a slightly smaller angle of otherwise wider soil patterns. Using this setup allows TSUNAMI to effectively capture large view of a picture, while displaying it in a sort of panoramic type of photo. We can also adjust the programming to move for a 360-degree view since our setup is functional enough for that, potentially leading to drone implementation for such causes if we were to expand on this project.

Stepper Motor

A stepper motor is the motor of choice for TSUNAMI. This motor moves in precis angular increments, typically within an open-loop control system, making it extremely useful for applications requiring accurate and repeatable positioning without the need for feedback. The stepper motor in this project allows for controlled stepwise scanning of the environment, giving a thorough examination of all segments within its field of view. The stepper motor will hold their position when powered, so that it maintains the scanning direction without drifting.

There are limitations included with the stepper motor. Higher speeds cause them to lose a great amount of torque, meaning rapid repositioning is challenging if we were to rescan an area. Stepper motors also consume power continuously, even when they are in the same location which leads to higher dissipation of heat and more power consumed. Even with these disadvantages, they are useful tools to move our camera in the linear direction we desire.

Different Types of Stepper Motors

Nema 17

The Nema 17 stepper motor is the smallest, most compact motor out of all of them at 42mm x 42mm. This motor is very efficient when it comes to wattage, assuming the operating currents of around 1.2 A to 1.7 A, meaning it also generates less heat. This makes it useful for battery applications or even more

for test applications. For the final design, it would be too bulky for the motor to move, but to get our software up and running with a test motor, it's still something we take into consideration.

Nema 23

The Nema 23 has a frame size of 57mm x 57mm, with enough power for moderate load handling for smooth and accurate scanning. It draws 2.8A per phase, thus giving better torque than the Nema 17 while still being economical. This does consume more power and generates more heat than the Nema 17, but it's the perfect size to handle the load we have while still being a good price for our project.

Nema 24

Applications that demand greater torque without jumping to the industrial standards would be the Nema 24. This has a frame size of 60mm x 60mm, providing a stronger holding power and smoother operation at higher speeds. The motor draws around 3.5A and 4.2A per phase, so not as energy efficient as the Nema 23, but accommodates for larger loads and greater fields of view. The trade-off is quite significant with it being much bigger, higher power consumption, and generating much more heat.

Nema 34

This motor is the industrial standard which is the most powerful motor of the selection list, containing a frame size of 86mm x 86mm. This motor specializes in heavy, demanding equipment/loads. The motor consumes around 5.5A to 6.3A per phase and requires a high voltage power supply, ranging around 36V to 80V. This motor is much more than what we require with also being high in cost, making it a lackluster choice for our specific specifications.

Table 20: Motor Selection Reference

Feature	Nema 17*	Nema 23	Nema 24	Nema 34
Frame Size	42 x 42mm	57 x 57mm	60 x 60mm	86 x 86mm
Holding Torque	0.14 -29.3 Nm	0.44 – 40 Nm	0.44 – 40 Nm	2 – 157 Nm
Step Angle (degrees)	0.9/1.8	0.9/1.8	0.9/1.8	1.8
Weight (kg)	0.37-0.44	0.8-1.75	0.84-1.83	3.05-4.5

Power Efficiency	High	Moderate	Low	Low
Typical Cost Range	$12-$24	$25-60	$32-$64	$80-$200+

Nema 17 will be used for testing only, with Nema 23 being in the final design

The Nema 23 was the chosen motor to be used in our final design to achieve the excellent performance-to-cost ratio while also not being too bulky for our final design. The Nema 17 was also grabbed to test our coding using development boards rather than using our final design motor, to grab an affordable and easily accessible motor. The primary factor for choosing the Nema 23 versus the other motors is mainly due to the costs when it comes to the energy demand being much greater than what our system really is supplying. The versatility, cost-effectiveness and ability to adapt to our mechanical needs is what makes the Nema 23 a much-needed choice for our technical requirements and efficiency.

3.4.7 Motor Driver

In our project, the motor is used to move our camera in a linear direction to grab a sort-of panoramic view of the soil we will be analyzing allowing a sufficient coverage of the soil patterns. To get this level of accurate and proper control, this requires a compatible motor diver for our motor to control that stepping sequence, current limiting, and voltage supply. Our motor will be a bipolar stepper driver which will need an appropriate motor driver to ensure the motor works efficiently and won’t consume too much power.

By selecting an appropriate motor driver, it allows us to have simpler wiring, reduced control complexity, and reliable operation for continuous scanning. We must consider factors like power efficiency, current handling, micro stepping capability, and control interface compatibility. This guarantees the scanning mechanism runs smoothly without the camera jittering and potentially causing blurry photos for the scanning process.

Motor Driver Technologies

Eason TB6600

This motor driver is a commonly used solution; however, it’s also not very efficient. The general consensus is that efficiency is quite poor when handling micro stepping and current regulation, affecting the smoothness and responsiveness of the motor that we desire. The chip does support up 4A of current, however the power dissipation and heat management is quite lacking when long term use would be the potential use of this camera, especially for soil tracking purposes.

DRV8825

The DRV8825 is also a very common stepper motor driver, especially in applications where moderate power ratings are required. It handles a maximum of 2.2A per coil, being suitable for Nema 23 motors where lower levels of current are employed. This driver is also quite known for current limiting and thermal protection features, being reliable and reducing power loss. However, if our motor were to demand higher amounts of current, we will not provide enough torque capability especially for longer term usages,

G203V

In terms of high performance and durability, the G203V is a popular choice especially. It drives 7A per phase and has the highest current drivers for Nema 23 stepper motors. This motor has low heat dissipation and provides smooth micro stepping control so that the stepper motor moves with low vibration. The main downside would be the cost; however, the long-term durability makes it viable for applications which could be in 1000s of hours for usage.

TMC2209

The TMC2209 is also popular, especially for two-phase stepper motors. The continuous drive current would be around 2A and peak current 2.8A. It also provides 256 microsteps while also being extremely quiet, making it efficient in terms of power consumption and heat dissipation. It's quite cost efficient with adjustable current limiting on the fly, to adjust our motor output by the use of potentiometer.

Table 21: Motor Control IC Selection Referenced

Feature	TB6600	DRV8825	G203V	TMC2209
Max Current	4 A	2.2 A	7 A	2.8 A
Voltage Range	9 V – 42 V	8.2 V – 45 V	18 V – 80 V	4.75 V – 28 V
Power Efficiency	Moderate	High	Very High	High
Microsteps	1/32	1/32	1/10	1/256
Control Interface	Step/DIR	Step/DIR	Step/DIR	Step/DIR or UART

Cost	$10.89	$12.95	$155	$8.95

The TMC2209 is the best selection due to the balance of specs while also being extremely affordable. The performance, efficiency, and control accuracy for the Nema 23 is consistently great. The micro stepping has incredible range with 1/256 stepping being a great strength of the TMC2209. This accuracy will be essential for our camera since it needs a smooth linear path to make sure each frame is not blurry and able to be rendered.

This stepper motor driver is a great component allowing peak currents to reach for the Nema 23, while also able to be used for the Nema 17 as a test motor. This makes transitioning much smoother and easier, while also pinpointing failures much clearer. A chip that is able to handle both motors makes testing and actual production of our final design much smoother while enabling greater control over problems that might arise. Our blend of high precision, good performance, on-board safety features, and this competitive pricing make the TMC2209 the most competitive stepper motor driver in our list specifically for hyperspectral soil analyzer project.

3.5 Software Background

3.5.1 C Programming Language

In the TSUNAMI project, programming the microcontroller using the C language provides us with various benefits that align with our system's requirements for efficiency and precision. C is a versatile and widely used language in embedded systems since its low-level hardware access, portability across various os's without heavy modification, high performance with lower overhead, fine control over memory allocation, and real-time capabilities make it an ideal choice for controlling our microcontroller.

One of the main benefits of using C is its low-level hardware access and memory. Embedded systems frequently interact with hardware components through software, and C provides direct access to memory and hardware registers through pointers and bitwise operators. This fine control gives us more room to be precise in photo processing, motor control, and even UI inputs to be fast and efficient. Memory management is essential for embedded systems due to the compromise in computation over size and convenience, allowing for systems to have their own specific processes confined to simple programs.

C is also highly portable, allowing minimal modifications when transitioning between different microcontroller platforms. This helps greatly especially when we

have a Linux based software in our Raspberry Pi and most of our programming will be done in windows. We can also iterate and improve future versions of the TSUNAMI project by improving hardware or more functionalities such as interacting with drone/movement-based components. The deterministic execution of C also allows predictable response times, providing itself useful for real-time applications that demand quick and reliable processing.

C also provides multiple libraries for interacting with the Pi and the microcontroller itself, making it smooth and easy to interact between both rather than making our methods. An extensive community and various libraries to help ease the software, makes the focus on hardware easier and more air-tight while also allowing debugging to be easier and efficient.

Leveraging C as our main language for microcontroller programming allows the project to have a well-established, efficient, and scalable approach in embedded system development. The strengths of the C language give us reliable performance and control makes it the simple choice to manage the peripheral controls of the microcontroller.

3.5.2 STMicroelectronics STM32 HAL API

STM has a built-in API (application programming interfaces) that improves our workflow, making the embedded hardware simple and easy to use. STM32CubeMX is a graphical software configuration tool which allows to generate C initialization code using graphical wizards. This allows us to set pins through the several GPIO pins while looking at the pinout itself making it easy to digest. The STM32Cube HAL, the abstraction layer, ensures maximum portability across the entire STM32 portfolio while also being available through all peripherals.

The HAL driver layer is a simple, generic multi-instance set of APIs to interact with the upper layer, including applications, libraries, and stacks. It’s mostly split between two categories, generic APIs which are common and generic functions for all STM32 series, and extension APIs which include specific and customized functions for a given line/part number. These APIs are more feature oriented rather than IP oriented, allowing for functions to be split between multiple categories. An example of this would be the timer APIs being split into several categories such as basic timer, capture and pulse width modulation. The HAL driver layer also implements run-time failure detection through checking the input values of all functions. The dynamic checking enhances the firmware while also being suitable for user application development and debugging. [15]

The LL drivers are hardware services that are also based on available features of the STM32 peripherals. They reflect the exact hardware capabilities and provide

atomic operations by calling on the programming model described in the product reference manual. They aren’t based on standalone processes while also not requiring any additional memory resources to save on their state, counter or data pointers [15]. The LL APIS are not provided for peripherals, unlike the HAL driver, in which optimized access is not a key feature which requires heavy software configuration.

For our project, the HAL driver is the clear choice between the two drivers mainly due to its ease of use and familiarity with the program outside of this project. The HAL provides us with a high-level and feature-oriented API, while hiding the MCU and peripheral complexity from the end-user. The LL mainly offers low-level APIs at register level, giving more optimization at the cost of complexity. This might be an option in the future if further optimization is required but for now, our code is simple enough where HAL is more than enough to be the main driver.

3.5.3 User Interface Design in Python

The user interface must be made in a graphical user interface (GUI) library compatible with both Python and the Raspberry Pi 4B. Several GUI libraries exist, but the simplest and most compatible due to its automatic inclusion within modern installations of Python is the Tkinter GUI wrapper. This library wraps the Tcl/Tk GUI program into user-friendly Python commands. Tkinter allows for quick and easy integration of buttons and image frames, as well as straightforward parameters for the visual design of the interface. Other GUI libraries (DearPyGUI, PyFLTK, etc.) suffer from poor compatibility with the Raspberry Pi 4B due to heavy instruction sets and/or a great number of required prerequisite libraries.

Chapter 4 – Standards and Constraints

4.1 Industrial Standards

4.1.1 PCB Design Standards

The IPC-2221: A comprehensive PCB design standard

The Institute of Printed Circuits (IPC), now the Global Electronics Association, released a set of guidelines for circuit board design now in use in many applications. The IPC-2221 at the time of writing is now technically obsolete, but its design standards are a great resource to follow pertaining to our application in student-led hardware design. In the subsequent figure, from the IPC's 2220 series of standards, IPC-2221 covers generic design standards for all circuit board types [12].

As substrates vary considerably between applications, different standards must be re-evaluated, thus the addition of extra addendums to the specification (2222, 2223, 2225, 2226). Many factors within circuit board design are considered within the IPC-2221, primarily concerning basic issues such as stackup material selection, thermal management, conductor sizing, and spacing of conductors/parts for optimum performance. All of these factors must be considered in any context, not just in compliance with the IPC-2221 if reliable., manufacturable, and durable boards are desired.

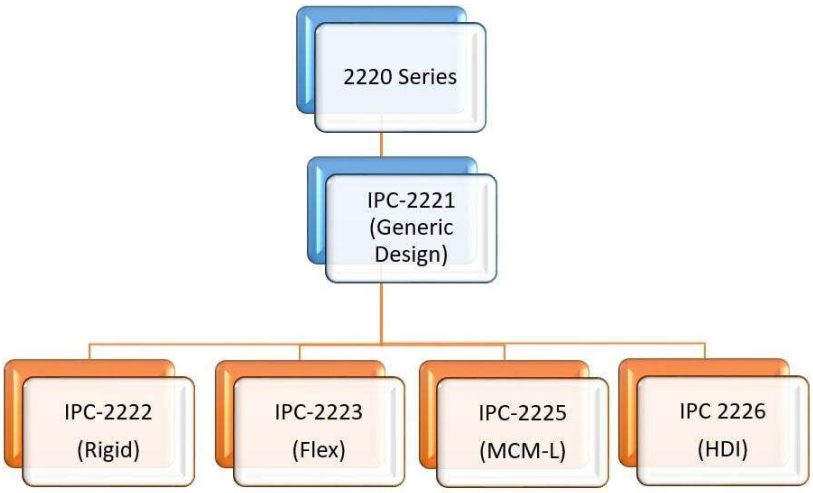


Figure 12: IPC-2220 design specifications tree, taken from Sierra Circuits' "Applying IPC Standards in Circuit Board Design," by Poulomi Ghosh

IPC-2221: PCB Material Selection

Within a circuit board's "stack-up", the choice of dielectrics, conductors, and other substrates are critical not only for the electrical design and signal integrity of the unit, but also for safety. One of the most common circuit board substrates used globally is FR-4 fiberglass, which is a specifically flame-retardant material. High-temperature withstanding substrates are paramount not only for maintaining structural integrity during high-temperature soldering/assembly, but also in fire safety. The IPC-2221 provides detailed recommendations for material uses in concerns with fire safety [12]. Other environmental factors such as thermal expansion stresses, shock, and vibration are also considered in the guidelines to ensure nominal performance beyond standard operating conditions.

IPC-2221: PCB Thermal Management

Thermal management is a critical aspect of circuit board design, performance, and manufacturing. In power applications in particular, large amounts of excess heat can be generated by board-mounted components. Strategies for managing excess heat accumulation and dissipation across the design are detailed within the IPC-2221 [12]. Such strategies include recommendations for the design of large copper pours or structures intended for heat dissipation, and thermal reliefs for component landing pads. Worthy of note is the detailing of the use of via arrays to transfer heat between layers to larger conducting thermal masses. These recommendations are of particular interest to our group as several of our subcircuits will need to dissipate some appreciable amount of heat.

IPC-2221: PCB Conductor Sizing

As there is yet no such thing as a room-temperature superconductor, all PCB in standard operation traces exhibit some degree of resistivity. With copper being the most common material used for circuit board conductors, the widths of every trace directly determine the resistance of the overall connection, and thus the temperature rise based on current. In following with the IPC-2221 recommendations, traces designed to carry high currents (such as in power supplies or motor driving applications like ours) must be given enough width and thickness such that they do not generate more heat than desired at an average current [12]. Traces left undersized to the design specification may heat up dramatically and lead to undesired performance or board failure entirely.

Beyond just thermal considerations, conductor sizing and dielectric selection are critical factors when ensuring signal integrity for controlled-impedance lines. At high frequencies, circuit traces must perform as carefully considered waveguides, adhering to microwave design principles.

IPC-2221: PCB Clearancing and Spacing

With increasing voltages on a board (and expected operating air pressure), the IPC-2221 provides clearance recommendations for conductors to prevent highly undesired effects such as arcing or interference [12]. At high voltages, the air around conductors and the dielectrics underneath them are subject to breakdown at high enough electric field magnitudes. As electric fields are measured in units of Volts per meter, shorter distances proportionally increase field magnitude and can not only destroy signal integrity via interference, but cause potentially catastrophic failure in short circuit conditions. At lower voltages, improper spacing of digital traces or digital and analog traces together can have severe adverse effects on the reliability of communication and data acquisition. The IPC-2221 also includes recommendations for spacing in these cases [12].

4.1.2 Digital Communication Standards

I2C Protocol Standards

The Inter-Integrated Circuit (I2C) protocol is a widely used serial communication standard used for data exchange between integrated circuits, especially in embedded systems [13]. I2C works in two bidirectional lines: the Serial Data Line (SDA), responsible mainly for transmitting data between devices, and the Serial Clock Line (SCL), synchronizing data transmission. These two lines are pulled up to a positive voltage, typically around +5V and +3.3V, with pull-up resistors to ensure proper signal integrity.

The I2C protocol supports multiple operational modes, with distinct maximum data transfer rates:

- Standard Mode (Sim): Up to 100 kbit/s
- Fast Mode (Fm): Up to 400 kbit/s
- Fast Mode Plus (Fm+): Up to 1 Mbit/s
- High-Speed Mode (Hs-mode): Up to 3.4 Mbit/s
- Ultra-Fast Mode (UFm): Up to 5 Mbit/s (unidirectional)

These speed classifications are essential for optimizing communication between specific requirements for our application.

I2C is a multi-master bus, meaning it allows multiple master devices to initiate communication. It prevents collisions by implementing an arbitration mechanism, where the masters support clock stretching, permitting slave devices to hold the clock line low to signal the master to pause, giving more flexible data handling.

I2C is extensively used in all domains, especially in sensor interfacing, display communication, and memory device integration between embedded devices. The

simplicity and efficiency of I2C makes it optimal for short-distance board communication.

SPI Protocol Standards

The Serial Peripheral Interface (SPI) is a synchronous serial communication protocol established by Motorola in the 1980s. It establishes high-speed, full-duplex data exchange between one master device and one or more slave devices, making it commonly used for embedded systems. Unlike I2C, SPI provides higher data rates and doesn't really require addressing, allowing efficient and direction communication between components. SPI lacks a formal standard directed by an organization, despite its widespread use.

SPI operates using a four-wire interface using a serial clock (SCLK), master out slave in (MOSI), master in slave out (MISO), and slave select. The master generates SCLK to synchronize data transmission making it much easier to use compared to asynchronous counterparts. It also requires a sperate SS line for each slave device, increasing pin usage. This can be alleviated by using a daisy-chain configuration to alleviate pin usage. The electrical characteristics are mostly defined by the hardware manufacturer, with common voltage levels being 5V, 3.3V, or even 1.8V in lower-power applications.

The primary advantage of SPI is simplicity and efficiency. The protocol requires minimal control logic, making it easy to implement in hardware and giving high data transfer rates. SPI doesn't require complex arbitration mechanisms, since the master directly controls data flow. Although SPI does contain challenges such as the lack of a built-in acknowledgement mechanism meaning no error detection, it still provides the best standard for data flow when we want high-data and low-latency.

4.1.3 Optical Hardware Standards

MIL-PRF-13830B

MIL-PRF-13830B is a US Department of Defense standard for the manufacture, assembly, and inspection of optical components. While originating as a military specification, many optical manufacturers (including ThorLabs, our main component source) follow this standard for all of their applicable sold components. The main document is 82 pages long [11], but the most important specifications of the standard that we should follow when manufacturing are the following:

3.5.2 Scratches. 3.5.2.1 Circular element. The combined length of maximum size scratches located on each surface of an optical element shall not exceed one quarter the diameter of that element.

3.5.3 Digs. 3.5.3.1 Dig designation. Dig numbers are the actual diameters of defects allowed, specified in units of 1/100mm. In the case of irregular shaped digs the diameter shall be taken as the average of the maximum length and maximum width. 3.5.3.3 Surface quality. All digs on each surface whose dig quality is number 10 or smaller shall be separated edge to edge by at least 1mm. The measurement of scattering shall not be required for surfaces where digs larger than number 10 are allowed.

3.6.3 Bonding defects (glass to metal). Bonded optical assemblies shall have a continuous bead of the cured adhesive along the edge of the bonded surface.

3.7.2 Relative humidity - temperature operation. Cemented components as a result of exposure to an ambient atmosphere of plus 130 ± 2 degrees F temperature, and at least 95 percent minimum relative humidity, and subsequent exposure to ambient air temperature of minus 80 ± 2 degrees and plus 160 ± 2 degrees F, shall not develop "feathering", show evidence of separation or softening of cement or other defects, except as specified in 3.6.

3.8.2.5.1 Vibration. After being subjected to the vibration test specified in 4.2.10.7 the optical instrument shall show no dirt (dust or lint) in excess of that allowed by the item specification. In the absence of detail requirements, dirt in any confined space shall not be in size or amounts larger than the allowable dig specification for the adjacent surface requiring the best dig quality. The instrument shall show no evidence of loose or damaged parts subsequent to this test.

3.8.2.5.3 Cleanliness. The optical surface of completed instruments shall be clean and free of condensates and volatile substances when examined by method specified in 4.2.10.9. Dust retention grease shall not be used except with specific authorization of the responsible technical activity

Appendix D Adhesive System, Epoxy-Elastomeric, For Glass To Metal D.3.7.1.2 Curing time and temperature. When subjected to a temperature not to exceed 74°C at the bond line, the adhesive shall cure in 4 hours maximum. At a temperature of 25° ± 2°C, the cure time shall not exceed 7 days.

4.2 Design Constraints

4.2.1 Power Constraints

Our system power feed specifies the use of an external DC voltage feed from a commercially available power adapter. While our instrument is not designed to draw more than 60W at absolute maximum (the same continuous power as a typical incandescent bulb), it still must respect the limitations of the power adapter. As detailed in the hardware block diagram in 2.2.3 and various choices in chapter 3, we want to feed 12V into the main bus of our system at up to 5A.

Our major constraint lies in not exceeding the current supplying capability of the input. To account for this, we will make efforts in the design to carefully monitor not only the total system power consumption, but also that of the individual subsystems. If our motor stalls from an event like a translation stage overrun, we could draw significant overcurrent and damage the entire system or PSU without fault detection.

Motors are also notorious for back-feeding inductive voltage spikes when decelerating or losing power suddenly. This is a very considerable constraint within the responsivity of our monitoring electronics, and thus fast-action transient mitigation circuitry outside of the MCU will need to be accounted for to protect the system.

4.2.2 Optical Constraints

Numerical targets and constraint relationships are given in Chapters 2 and 6. Here we'll give a practical explanation of targets and trade-offs. In an ideal imaging system, spot size is smaller than pixel size and field-of-view (FOV) is high. These two objectives alone constrain each other, as faster focusing lenses have larger FOV but worse aberration and therefore larger spot size. Rarely can both objectives be achieved simultaneously (e.g. by using aspheres), so trade-off is made by either letting the spot size be a few pixels across or limiting the FOV and panning the camera.

An additional constraint for a hyperspectral imager is the simultaneous guidance of an imaging and a diffractive axis. This connection is shown in diagrams in Chapter 2 and numerically discussed in Chapter 3, but to clarify the connection and constraints:

- The entering light must collimate in order to easily retrieve wavelength.
- The ratio between the imaged FOV and the wavelength spread angle is the same as the aspect ratio of the detector.
- The detector must be a focal length away from the imaging lens, and the grating position must be set so that the wavelength range fits on the detector

The preceding points combine to form the general constrained relationship that a greater grating groove density allows for a longer imaging focal length and a tighter spot size but less FOV.

4.3 Practical Constraints

4.3.1 Time Constraints

The hardware of this system should be completed and functional by early March. This constraint back-propagates time constraints to other parts of the system. We need to consider breaks for holidays as well, as we can't expect others to work on the project then. This means that our work and decisions have to be made at pace and uniformly throughout time until the project is completed. We cannot afford to wait just before deadlines.

Another major concern is shipping time. Many of the components required may come from overseas, and between natural delays and ongoing political trade battles, components should be ordered well in advance so that surprise shipping delays or outright cancellations don't jeopardize the completion of this project.

A detailed project timeline is given in Chapter 10.

4.3.2 Financial Constraints

Our budget for this project is $1500 ($500 per person). This project is not sponsored, and this budget cannot be exceeded. The most constraining components financially are the optical elements. Lenses and gratings manufactured within standard have significant price tags that quickly become prohibitive for more specialized components (i.e. aspheres, holographic gratings, etc.). We have to be very mindful of exactly how tolerant our optical requirements constraints are and if there are cheaper ways to fulfill them than buying specialty components (i.e. aperture vs. aberration, translation vs. large FOV).

Ordering components should be the responsibility of all team members and cost should be evenly distributed throughout the project so that no one member is placed unfairly under a greater financial burden. Purchases should be tracked completely and receipts saved so that an exact distribution amongst the three of us can occur at the end of the project.

A detailed budget is given in Chapter 10.

4.3.3 Environmental Constraints

While our project doesn't directly implicate the potential for environmental harm, there are a few examples in which we will need to stay mindful and considerate for the health of ourselves and others. One of the most notable examples of material concerns in the environment is the use of lead-based solder. Modern solder has now superseded the need for lead-based compounds, but some designs and prototypes still incorporate it. When it comes time for disposal or recycling of

electronics hardware, heavy metals such as lead pose significant environmental concern.

Beyond disposal, the sourcing of resources for various components is oftentimes a point of contention and concern. One of the prime examples of this is within the chemistry of batteries and capacitors. In our project in particular, our SBC power supply uses a tantalum-based polarized capacitor for its output filtering. Tantalum capacitors are superior in many ways outside the scope of this discussion, but importantly the sourcing of Tantalum brings with it not only environmental, but ethical concerns. In many facets of Tantalum sourcing, mining of the material is usually done in areas sensitive to environmental damage, with mining practices that typically do not concern this. Not only are environmental practices sometimes subject to being problematic, the conditions in which many workers are exposed in some supply chains for electronics can be absolutely unethical. Careful consideration must be taken to understanding an electronic component manufacturer's sourcing practices, rather than just blindly adding parts to a Digikey order.

4.3.4 Manufacturing Limitations

Circuit board fabrication houses (such as our used JLCPCB) and machine shops have soft and hard limits on what kinds of design specifications are easy, costly, or impossible to manufacture. For JLCPCB and other board houses in particular, their minimum and maximum parameters for their different production environments are typically provided as "design rules". Design rules include but are not limited to:

- Minimum trace widths and minimum/maximum copper thicknesses
- Minimum trace spacing (including spacing from a trace to board edge)
- Minimum via size and annular width dimensions
- Maximum board size allowance
- General dimensional tolerances
- Board thickness tolerance (varies with stack-up)
- Dielectric constant tolerance (necessary for ensuring controlled Z)
- Via-to-Hole minimum spacing
- Slot thickness and geometry (varies due to drill-bits needed)
- Minimum soldermask widths and soldermask expansion
- Text widths and font sizing minimums/maximums (and general silkscreen)

Most electrical design automation (EDA) software such as Altium, Fusion360, or KiCAD will support native algorithmic tools typically referred to as a "design-rules-checker" or DRC. The DRC can be fed manufacturing design rules from the board house and design processes to be used, and run a full scan/report on all facets of

the finished board geometry. This aids immensely in the circuit board design engineer's ability to catch critical or minor errors in the design before being committed to fabrication. None of the above-mentioned topics include limitations around automated assembly, which is an entirely separate process on its own full of its own considerations. Assembly varies drastically between assembly providers due to equipment capabilities.

4.3.5 Scalability of Design

Our project must adhere to the senior design timeline. Namely, we have a total of 35 weeks from project inception to design, manufacture, test, and validate performance of an entire imaging instrument. As a result, our demonstration hardware will only be a first prototype to satisfy the goals and objectives initially proposed at the project start. Our design's scalability, or its ability to be expanded in performance across various metrics, is limited in some ways more than others. For instance, our imager optics are not very easily scaled up as our optical design is fixed based on our entry aperture size and CMOS sensor array size. For the software, the capability to expand the UI from the prototype software design to a more capable and extensive user interface is perhaps the easiest way to scale the design further. As none of the three of us have proper backgrounds in computer science, this is one of our least experienced and prioritized facets of the design. Electronics can be scaled as the electronics within the design are overbuilt to slightly exceed system requirements. Significant scaling of the electronics beyond a basic level would require notable redesign.

Chapter 5 – LLMs and Other AI Platforms

We have not and will not utilize any large language models (LLMs) for the research, design, writing, or coding of any material in this project. The limited accuracy of these models provides an ethical concern for us as engineers, as hallucinations threaten our responsibility to produce a design that meets our stated objectives reliably. We recognize that LLM usage provides results much faster than traditional methods, but this time savings is for naught if the answer is wrong.

We do not see any novel capability of LLMs that would require their usage over human work. Research can be done well with existing search engines. Research via an LLM isn't reliably citable. Design can be done using our knowledge taught to us in class and personal project experience. Design via an AI assistant allows hallucinations to become physical defects. Writing can be done with our keyboards. Writing via a LLM, even without hallucination, produces a banal and encyclopedic tone that obscures the points we mean to convey. Coding can be done using our prior experience with projects and research simulation. Coding via an LLM will generate a great excess of bugs as limited examples of any part of our goal software would exist in a training data set, and hand-holding a LLM with prompt engineering could very well take just as much time as writing the code ourselves.

We take considerable pride in calling all of this work our own. If faults arise in our system, we should be accountable for them, not an LLM. When our system succeeds, we should be congratulated for it, not an LLM.

With that being said, search engines vary in quality and usability. The three main engines we currently use in our design process are the Google main search system, Google Scholar (a more research-oriented version of the main engine made for academia), and UCF Library search. Pros and cons are tabulated amongst each and are provided in the following table:

Table 22: Three case studies across search engine platforms

Engine	Pros	Cons
Google Main Search	The main bar is simple and straightforward to use. Search query results are optimized for relevant information. Topics partially related to a main query appear in search automatically to further research depth. Factuality can be discerned by comparing several search results.	There is no barrier of factuality to have information appear (i.e. sources aren't always peer reviewed or even true). Access to links isn't guaranteed (i.e. certain paywalls on news websites or journals). Academic results can get buried due to less web traffic than popular material.
Google Scholar	Links given are either peer-reviewed journals or ArXiv. Search results are optimized. Articles partially related to the main query appear in search automatically. Search occurs across all journals rather than a subset of popular or easy access titles.	ArXiv is not peer-reviewed and Scholar does not disclose this fact before displaying these results. Access isn't guaranteed (i.e. journals that UCF doesn't have institutional access to).
UCF Library Search	Information is peer-reviewed and/or professionally published. Access is (almost always) guaranteed for UCF students. Physical copies are sometimes accessible if needed.	Search engine optimization is very basic and title-centric, with related topics not always appearing.

Chapter 6 – Hardware Design

6.1 Imager Optics

6.1.1 ZEMAX Simulation

Optical design and simulation was primarily done in Ansys's ZEMAX optical simulation software. Our entry aperture, primary focusing lens, diffraction grating, secondary focusing lens, tertiary lens, and CMOS imager array target were simulated and ray-traced across our wavelength range of interest. Worthy of note, the colors of each linear beam shown superimposed is representative of its actual wavelength. ZEMAX displays the NIR as black.

Table 23: ZEMAX optical setup parameters

	Surface Type	Comment	Radius	Thickness	Material	Coating	Semi-Diameter	Chip Zone	Mech Semi-Dia	Conic	TCE x 1E-6	Lines/µm	Diffract Order	Par 3(unused)
OBJECT	Standard		Infinity	300.000			15.722	0.000	15.722	0.000	0.000			
STOP	Standard		Infinity	23.505	V		12.700	U 0.000	12.700	0.000	0.000			
(aper)	Standard	f=100 LA1509	Infinity	3.590	N-BK7		12.700	P 0.000	12.700	0.000	-			
(aper)	Standard		-51.500	46.495	P		12.700	P 0.000	12.700	0.000	0.000			
(aper)	Standard	f=-100 LC1120	-51.460	4.000	N-BK7		12.700	P 0.000	12.700	0.000	-			
(aper)	Standard		Infinity	3.000			12.700	P 0.000	12.700	0.000	0.000			
	Standard		Infinity	3.000	B270		12.700	P 0.000	12.700	0.000	-			
(aper)	Diffraction Grating	GT25-03	Infinity	0.000			12.700	P -	-	0.000	0.000	0.300	1.000	
	Coordinate Break			0.000	-		0.000	-	-		-	0.000	0.000	-10.800
	Coordinate Break			2.000	-		0.000	-	-		-	0.000	0.000	0.000
(aper)	Standard	f=30 AC254-030-AB	20.027	12.000	S-BAH11	AR	12.700	P 0.000	12.700	0.000	-			
(aper)	Standard		-17.440	3.000	S-TIH6		12.700	P 0.000	12.700	0.000	-			
(aper)	Standard		-93.082	20.536		AR	12.700	P 0.000	12.700	0.000	0.000			
IMAGE	Standard		Infinity	-			3.150	U 0.000	3.150	0.000	0.000			

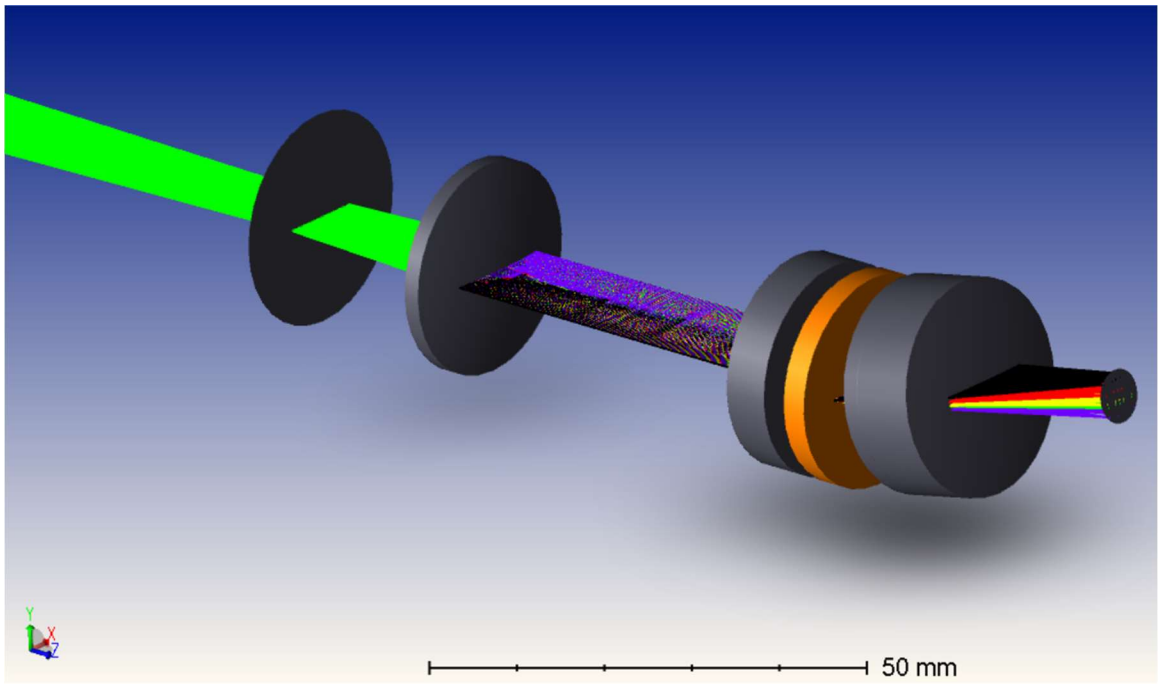


Figure 13: 3D rendered model of ZEMAX designed imager optics simulation

Our Zemax system images objects at a variable distance (d_o = *300mm* in the photos above) with an FOV of 6° and with lenses of 1" diameter. The aperture is

10 mm wide. Pickup is used so that the entry lenses are adjustable while maintaining an aperture to back lens distance of 70 mm. Our image surface has a diameter of 6.3 mm, matching the diagonal of our CMOS detector. Lens and grating specifications match those given by ThorLabs, and in the case of the lenses are the exact Zemax specifications downloadable from the ThorLabs website.

Knowing that coarser gratings lead to a longer imaging lens focal length and therefore less aberration, we chose a grating of 300 lines/mm (ThorLabs GT25-03), giving an imaging focal length of

$$f_i = \frac{ha}{\Delta\lambda} = \frac{(6.3mm)(3.33\mu m)}{(950nm - 400nm)} = 38.2mm$$

Considering options for sale from ThorLabs, we went with an imaging focal length of 30 mm as we're using the sensor diagonal for estimation with the actual detector height being slightly shorter. Manual calculation was only used for rough specification determination and rough placement. Exact dimensions are given by manufacturers, and exact placements of optical components are found by optimization. RMS spot size optimization was used to position the detector plane and adjust the entry lens system to different object distances, with the detector optimized first for an infinite object distance, and then fixed while the entry lenses were adjusted for an object distance of 300 mm.

6.1.2 SD1 Final Demonstration Optical Design

We presented a demonstration of our optical design to Dr. Kar, Dr. LiKamWa, and Dr. Hagan on 18 November 2025. This demonstration was constructed of mainly borrowed components from the lab of Dr. Vodopyanov, Jacob's principial investigator. All components were borrowed with permission and returned to their original places after the demonstration. The components that were our own and will be integrated in the final design were the Raspberry Pi 4B, the IMX296LQR-C Camera, USB keyboard, and wireless USB mouse. The parameters of the borrowed entry lenses and grating match our design specifications, but the imaging lens was a f = 50 mm BK-7 lens rather than our ideal f = 30 mm achromat. We also did not remove the IR filter from the camera sensor. Photos of the demonstration are below.

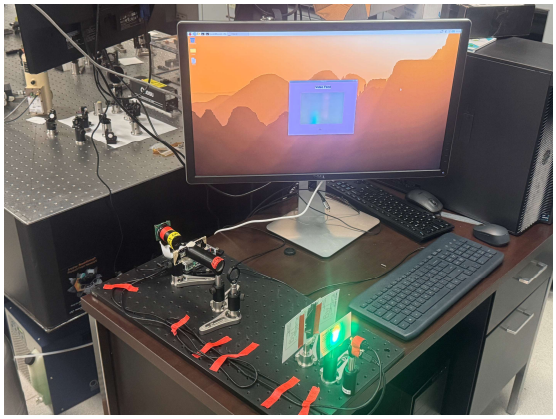


Figure 14: Setup, Uncovered

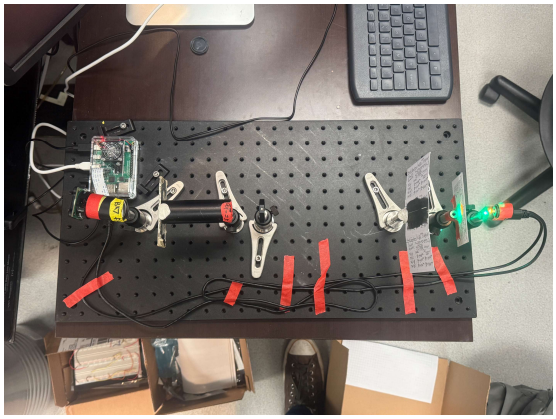


Figure 15: Setup, Top-Down

Figure 16: Setup, Operating

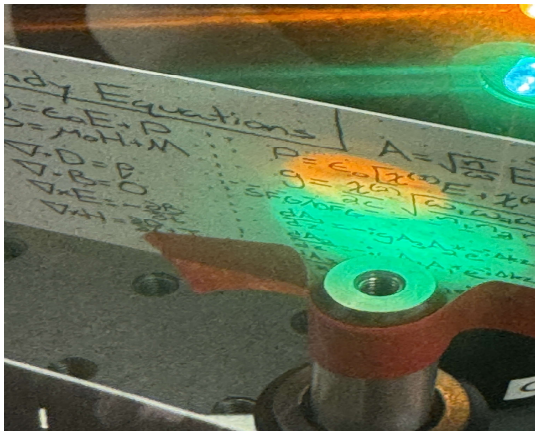


Figure 17: Test Object

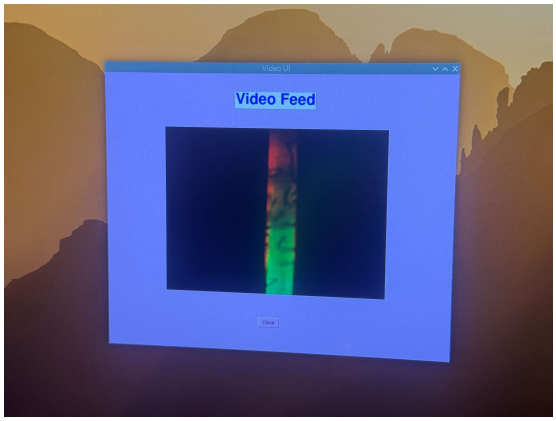


Figure 18: Standard RGB Image

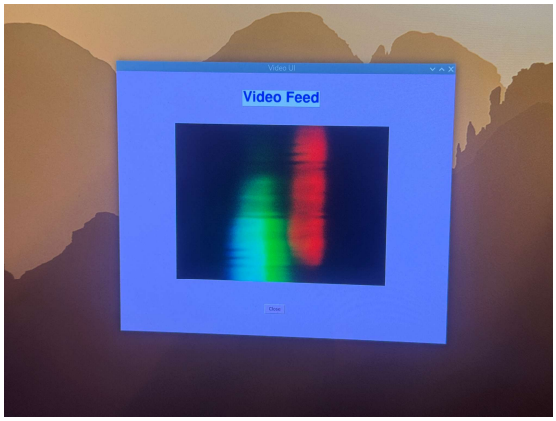


Figure 19: Hyperspectral Image

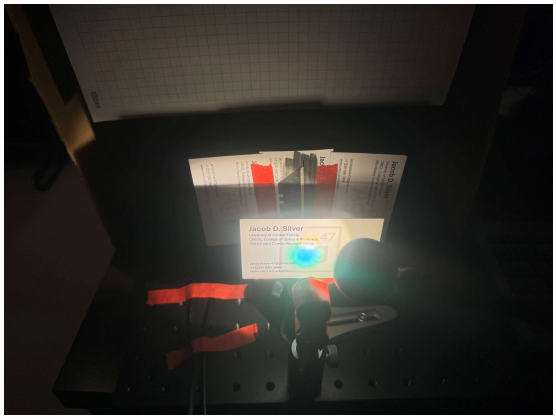


Figure 20: White-LED-lit Object

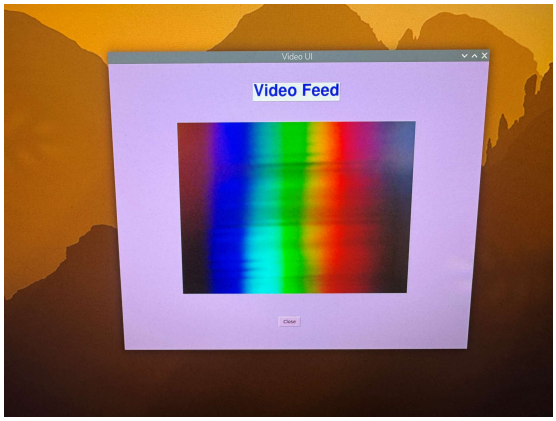


Figure 21: Hyperspectral Image of White-LED-lit Object

This demonstration was a successful test of our design and its philosophy. The hyperspectral image of the test object clearly shows the spatial distribution of the 600 nm orange LED and 520 nm green LED. What appears yellow in the standard RGB image is split into the overlap of the individual LED spectra.

Covering the setup during operation was necessary as stray light could easily wash out the image, but this concern will easily be taken care of in the final design as the imager optics will be in a 3D-printed housing. Another issue was stray diffraction patterns appearing from the other diffraction gratings included on our borrowed unit (an educational piece with 100, 300 and 600 lines/mm section), but again this concern will not appear in our final design as our intended grating will have only one single period. Adjustability on the breadboard made obvious that achieving fine spectral lines is done through not just making the slit thinner, but also by placing the entry lenses further from the slit.

6.2 Laser Cursor Optics

6.2.1 Optical Design Approach

The design of the laser cursor optics system primarily revolved around the beam shaping of our diverging elliptical beam from our laser diode. Our goal as stated in the system specifications is to deliver an elliptical beam shape, approaching a **line** with at least a 20:1 ratio of major to minor axis at the target distance. To approach this, we first began with understanding the divergence of the laser diode. Our selected diode (RLD63NPC8) featured nominal divergence angles of 30° and 8° on the fast and slow optical axis, respectively. For reference, the optical design is approached as a superposition of rays traced in both a “fast” and “slow” axis, as previously mentioned.

Many approaches to collimation, expansion, and divergence were considered in the design process. We ultimately chose to immediately collimate our laser diode's 30:8 elliptical beam shape using a standard plano-convex lens, before applying any other optics. By immediately collimating the laser, we could ensure that the numerical aperture of the lens needed wouldn't be at risk of not matching the diode's, as the 30° divergence quickly ate into the aperture size and thus implementation size needed. The size of this lens was also carefully considered as while very short focal length lenses are readily available, we really felt that implementing such a small lens would be rather difficult to align mechanically and integrate into a deliverable product. As our imager is expected to be poised at roughly 500mm away from our target in general operation, our expected linear FOV size (accounting for our 6° divergence) would result a roughly 57mm long strip. Following a 20:1 beam profile, the maximum allowable size for the lateral FWHM of the cursor would be under 2.85mm. *Following the initially collimated beam, the first lens' focal length was thus selected to match a beam-width on the slow-axis of 2.85mm or smaller.*

Going with a standard 18mm diameter lens, we picked a 20mm focal length to perform this initial collimation, working with the Thorlabs LA1859. As we had no concerns over throughput, we chose to use uncoated optics, saving on cost. To the collimated 30:8 elliptical profile, our beam diameter on each axis would be given by the following formula given our focal length *f*:

$$\mathrm{w_f} = 2\mathrm{ftan}\left(\frac{\theta_\mathrm{f}}{2}\right) = (2)(20mm)tan(\backslash frac\{30\}\{2\}) \approx 10.71mm$$

$$w_s = 2ftan\left(\frac{\theta_s}{2}\right) = (2)(20mm)tan(\backslash frac\{8\}\{2\}) \approx 2.797mm$$

With w_f and w_s being the beam widths of the fast and slow axis, respectively (along with their full-width divergence angles from the laser diode). The "trick" with this approach was to simplify the design *by allowing the slow axis to stay collimated along the propagation direction, while allowing the fast axis to expand at a slow rate*, thus trading a collimated fast axis for a much smaller exit aperture size. With a fully collimated beam on both axis on a 20:1 or better ratio, our exit aperture diameter would have to be at least 57mm wide, with very little of that lens area actually being used to refract light. Having such a wide lens and using the larger part of its diameter would also prove to be problematic for aberration at the target distance, especially when using a much more affordable, *spherical* cylindrical lenses. Non-spherical cylindrical lenses for single-axis correction are rather incredibly expensive from any source as they are so specialized.

With this in mind and the collimated beam sizes w_f and w_s, we would need an exit aperture and thus second lens diameter of at least 10.8mm. As we wanted to preserve the already collimated slow axis, we turned to cylindrical lenses to expand the beam along its propagation. As our expected cursor to target distance would be roughly 500mm, we wanted to specify a negative focal-length cylindrical lens that would reach at least a 20:1 shape ratio well before the 500mm target. 330mm was chosen as a minimum distance for the number to calculate the secondary lens focal length.

Our divergence angle was next of immediate consideration. Conveniently, matching a 6° divergence angle similar to our imager would allow us to reach the minimum 20:1 profile before the 500mm target distance. For a 6° divergence angle, a focal length of -50mm was experimentally chosen after traditional 1st order paraxial calculations proved insufficient when dealing with aberrations later in simulation. As we also didn't need this lens to be coated, we chose a Thorlabs LK1662L1 plano-concave cylindrical lens. All lenses were suitable for N-BK7 as we were simply using a single wavelength at 635nm anyways.

6.2.2 ZEMAX Simulation

As nearly all quick calculations for this system cannot factor in spherical aberration, we chose to take our design approach into simulation to account for all the other factors we may have missed. In addition, the 30° and 8° values were simply the nominal ranges in standard operation, and testing the full range of each would be important to validate the design's chance of success. In doing a simulation, we were also able to use ZEMAX's optimization tool to find the best distances between the lenses and from the lenses to the diode beyond just what our 1st order calculations found. The lens data were downloaded from the Thorlabs part pages as .ZMX files and imported into a master file. The laser diode beam pattern was created by first casting a cone of rays at the 30° divergence of the fast axis, and then creating a surface right on the first lens interface with a 30:8 elliptical aperture. This was the way we found easiest to force the program to create the 30:8 elliptical profile needed. The target was placed 330mm away from the starting surface, being the ray tracing origin (as the diode). The following images and tables show the information entered and gathered from the optical simulation:

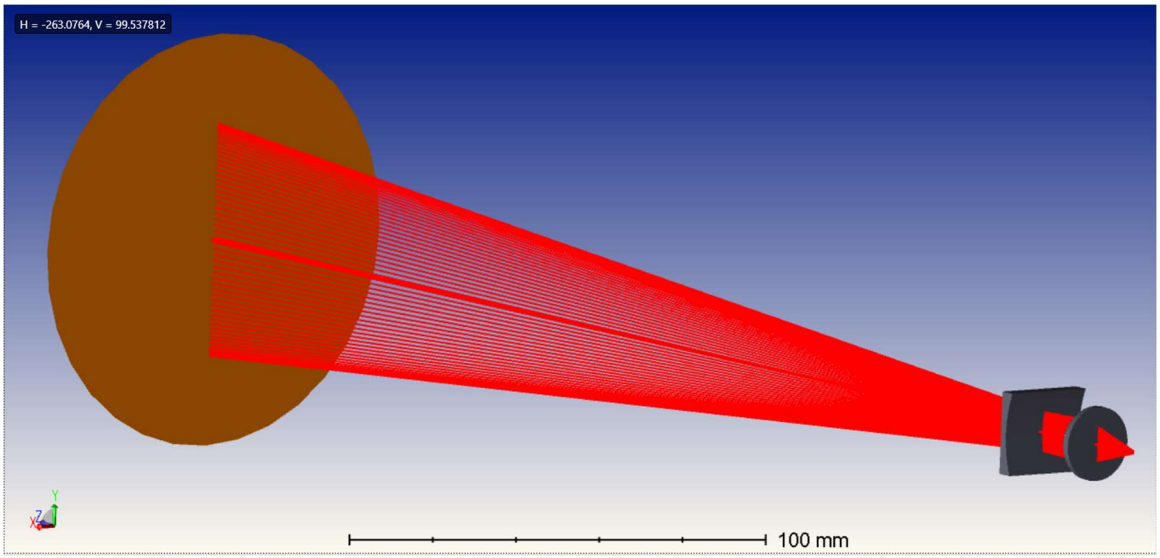
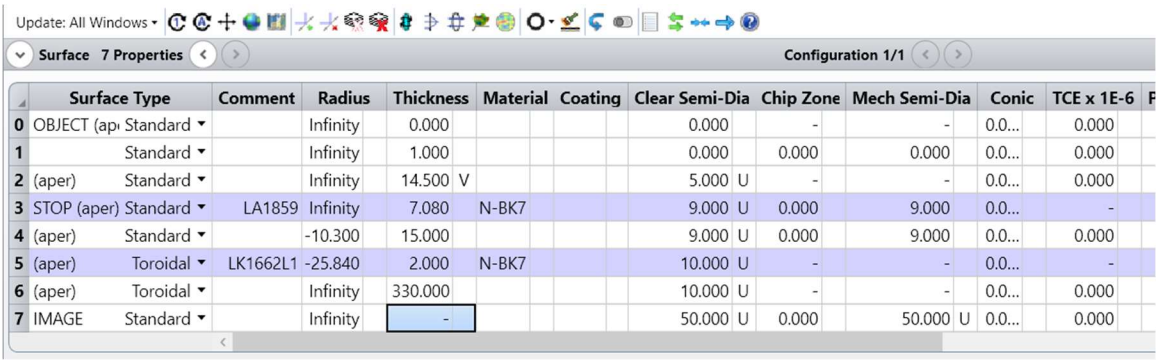


Figure 22: 3D shaded render of overall ZEMAX simulation



Update: All Windows

Surface 7 Properties

Configuration 1/1

	Surface Type		Comment	Radius	Thickness	Material	Coating	Clear Semi-Dia		Chip Zone	Mech Semi-Dia		Conic	TCE x 1E-6
0	OBJECT (ap	Standard		Infinity	0.000			0.000		-	-		0.0...	0.000
1		Standard		Infinity	1.000			0.000		0.000	0.000		0.0...	0.000
2	(aper)	Standard		Infinity	14.500	V		5.000	U	-	-		0.0...	0.000
3	STOP (aper)	Standard	LA1859	Infinity	7.080	N-BK7		9.000	U	0.000	9.000		0.0...	-
4	(aper)	Standard		-10.300	15.000			9.000	U	0.000	9.000		0.0...	0.000
5	(aper)	Toroidal	LK1662L1	-25.840	2.000	N-BK7		10.000	U	-	-		0.0...	-
6	(aper)	Toroidal		Infinity	330.000			10.000	U	-	-		0.0...	0.000
7	IMAGE	Standard		Infinity	-			50.000	U	0.000	50.000	U	0.0...	0.000

Figure 23: Lens data entry for simulation

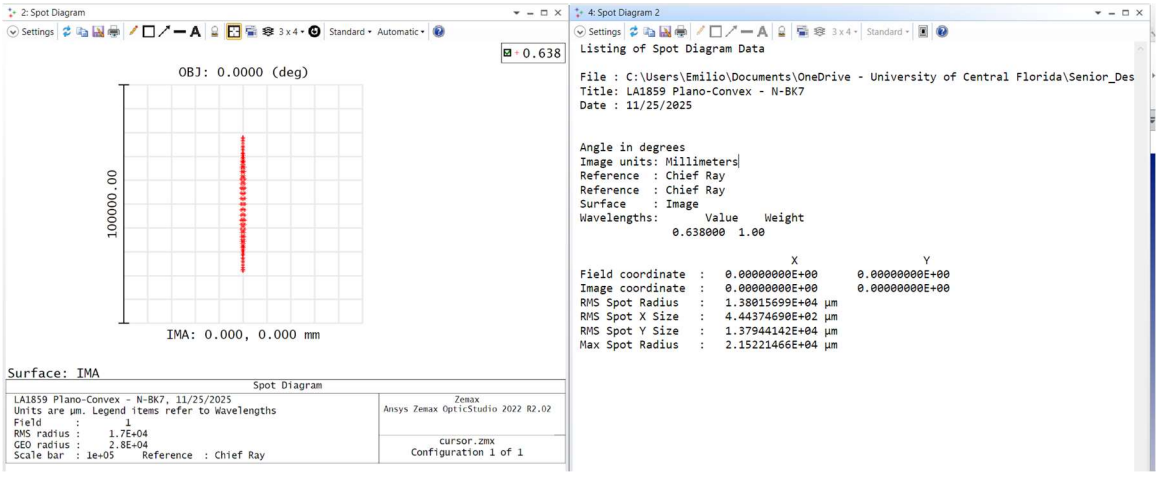


Figure 24: Spot diagram and size data

6.3 SBC Power Supply (5V)

6.3.1 Schematic Design

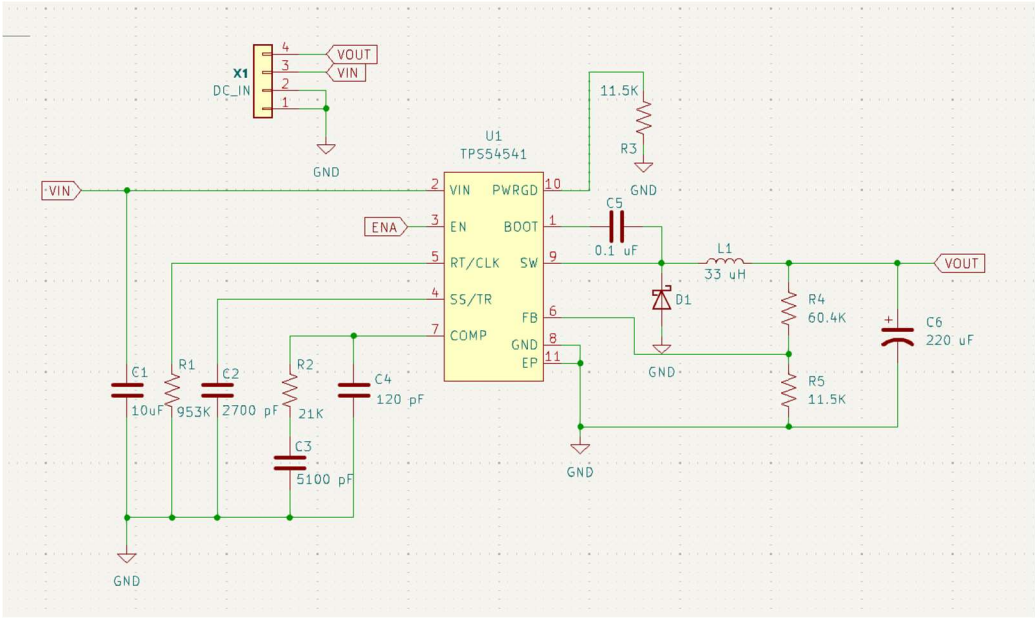


Figure 25: Schematic design for the SBC power supply circuit

The PSU intended to supply power to our Raspberry Pi Model 4B SBC is based around the Texas Instruments TPS54541 buck regulator IC. As mentioned in chapter/section 3.3.5, the TPS54541 is a buck regulator IC that meets our design specifications. This PSU will need to supply up to a 3A continuous load at 5V, with up to a 5A peak draw during transient power spikes. With a 12V input, the inductor at low load will run at a low duty cycle.

In general, the schematic design approached here very closely followed the recommended schematic detailed within the TPS54541 datasheet. Design parameters and component selection was conducted with careful analysis from both formulas within the datasheet and calculation tools provided by Texas Instruments. The design will switch above 100kHz, with the frequency adjustable using a tuning resistor on pin #5 (timing resistor input). A soft-start capacitor is attached to pin #4 to provide a gradual voltage ramp upon startup to the SBC. Following TI provided calculator outputs, the component values were chosen for the compensation network's control loop inside the IC's closed-loop feedback system. A Schottky type recovery diode was fitted on the switching node of the regulator to provide a conduction path for the negative transients produced from the inductor.

For the diode, the maximum voltage, forward voltage drop, and response time were selected following datasheet recommendations. Using the internal voltage reference within the feedback error amplifier of the chip, resistor values for the voltage divider from output to the FB pin were calculated. Values were kept in the kilo-Ohm range to avoid excess power draw while also balancing noise performance in the feedback loop. The output capacitor selection was followed closely with the TI calculator package recommendation. A tantalum capacitor was chosen to provide optimal equivalent-series-resistance (ESR) and inductance (ESL), keeping the output ripple of the circuit to a minimum. Input capacitance was followed by TI recommendation as well, using a ceramic 10uF capacitor to provide local decoupling of transient current from the main power source. In total, a full bill-of-materials (BOM) of all parts was sourced (see Appendix B)

6.4 Motor Driver Power Supply (5-20V)

6.4.1 Schematic Design

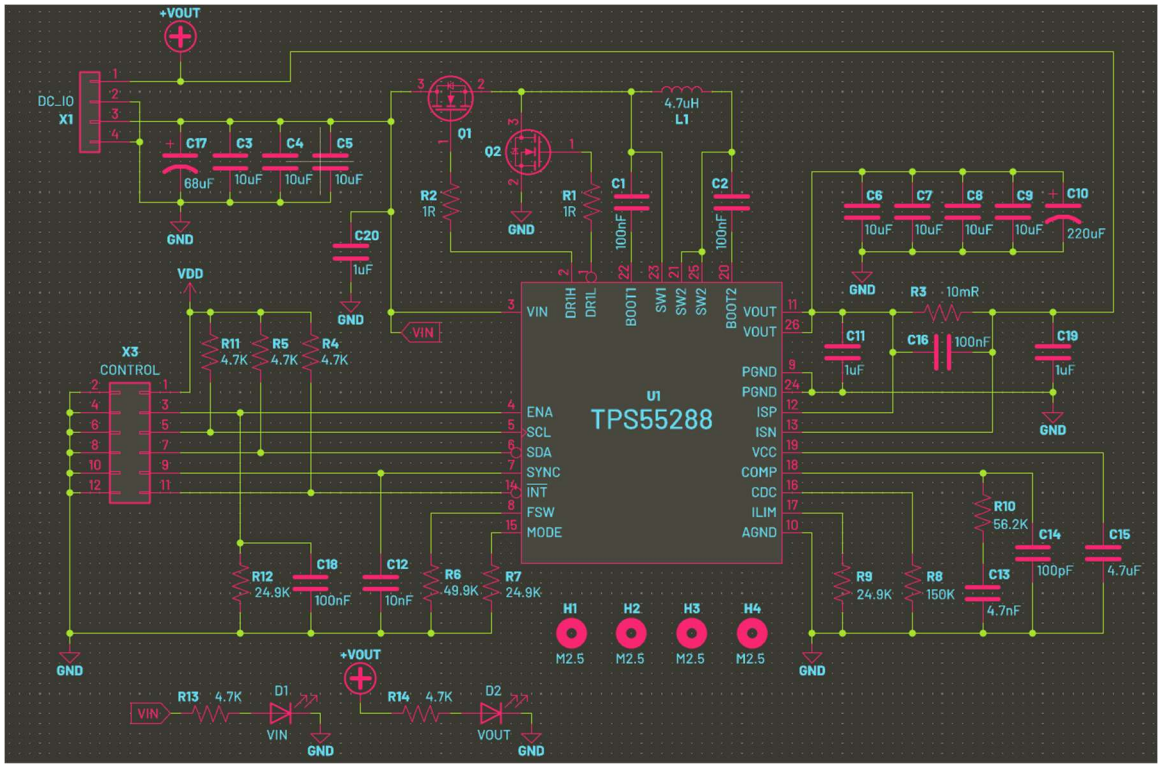


Figure 26: Schematic design for the SBC power supply circuit

As mentioned previously in section 3.4.5, our digitally controllable motor driver PSU was based around the TPS55288. Parts values were calculated with the design calculator provided by TI with cross-reference to datasheet design formulae. The component selection for each element was done through filter

search of the component vendor, following datasheet recommendations for dielectrics, voltage ratings, current ratings, tolerances, temperature coefficients, and mechanical sizing.

Notably, as this is a buck-boost converter, two of the FETs needed for boost operation are needed to be externally provided, as the internal FETs need accompaniment. External FETs were directly matched to the FETs used in both the datasheet recommendation and the evaluation kit sold for this IC. The inductor was also picked directly from the evaluation kit recommendation.

Custom-chosen connectors were implemented into the design, namely the control connector. The control connector is specifically implemented in this test design to facilitate easy interfacing between this PSU circuit and our microcontroller during development and testing of our hardware. The power connector was selected as a Wago wire-terminal connector, allowing us to directly and securely connect bare wires from our test supplies and loads to the circuit when testing.

As before, other components such as those in the compensation network were calculated and chosen from the TI provided formulae to best tune the control parameters for the loads expected on the output of the PSU. In total, a full BOM of all parts was sourced (see Appendix B).

6.5 Microcontroller Integration and Schematic Design

In developing our integrated control solution, we chose the STM32G431RBT6 as previously documented in section 3 for hardware selection. This particular -RBT6 suffix was used as a development board for this LQFP-64 variation was readily available from STMicroelectronics. Pinout selection/design would be needed for development on evaluation boards, before committing to an initial PCB design. By using the same chip from the development board provided, we would be able to copy over the pin connections when implementing the final design.

The MCU system would need a fair few auxiliary components besides just the MCU itself to be fully operational. While every microcontroller is different in its needs to be embedded, depending on the application, many external components might be needed. At the bare minimum, the MCU selected has four main VDD supply lines which need to be decoupled with a low ESL 100nF capacitor as per the datasheet recommendation. Along with this, the RESET pin on PG10 would need to be held high with decent reserve capacitance (also 100nF) to prevent the MCU from spurious resets on fast supply transients. A user button for resetting the MCU by discharging this capacitor would also prove useful to implement in the final design. On buttons, a second user button would be installed to add a user programmable test feature, useful for firmware testing and potentially for the final

build. Both of these buttons are also installed as necessary hardware on the development board.

Debugging would be achieved on the development board using the internally connected STLINK V3 debugger circuit, already integrated. For the final design, a 2x7 1.27mm connector would be used to connect to an external STLINK device through the pins PA13 and PA14 for the Cortex-M4 debug link via serial-wire debug (SWD). For the analog peripherals (DAC and ADC later for current monitoring), a stable and clean analog supply would be provided from the 3.3V rail using a matched inductor and secondary capacitor to filter any high frequency noise. The MCU features an internal 2.5V precision shunt reference to provide reference voltages to both the DAC and ADC peripherals integrated on the die.

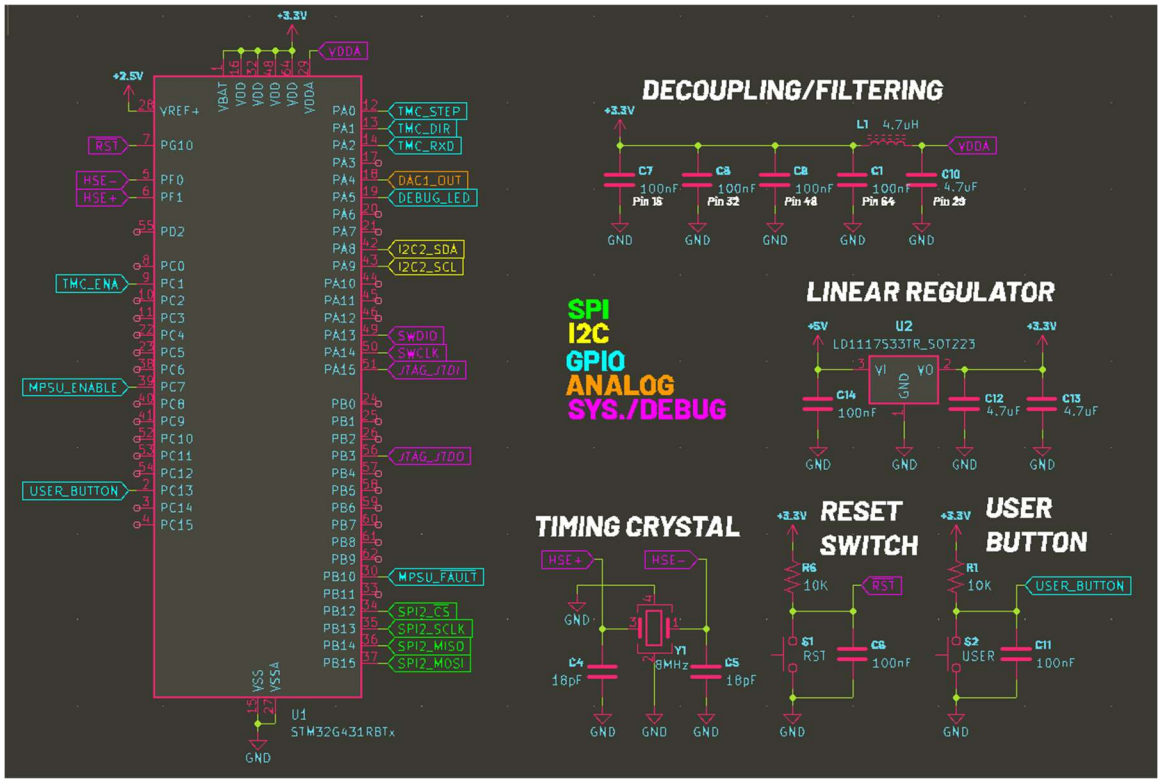


Figure 27: MCU integration schematic, featuring all necessary components

The 5V buck converter on the main board assembly would feed the 3.3V LDO previously specified in chapter 3 to easily convert over. This was done with an LDO simply for implementation ease as the 3.3V bus wouldn't realistically need to draw more than 150mA of current and only drop 1.7V from the 5V line to the 3.3V. One other thing of important note was the 8MHz external timing crystal. Loading capacitors were calculated as per the crystal datasheet recommendation. This external crystal is also present on the development board, making design transfer

slightly easier. This crystal would prove paramount in timing critical applications, concerning the SPI slave peripheral for the MCU in particular.

In the previous figure of this section, the KiCAD implemented schematic is shown featuring all parts selected to embed the MCU. One of the major motivations for using an MCU from STMicroelectronics is their CubeMX “micro-explorer” GUI tool that allows to plan and configure MCU peripherals graphically per-pin. Pin planning on this tool makes schematic design much easier and pin to peripheral mismatching errors much less likely. This CubeMX tool also very conveniently generates peripheral instantiation code for startup based on the GUI parameters entered in each peripheral. The subsequent figure (below) shows the CubeMX window view of the microcontroller pinout as it would appear on the final PCB:

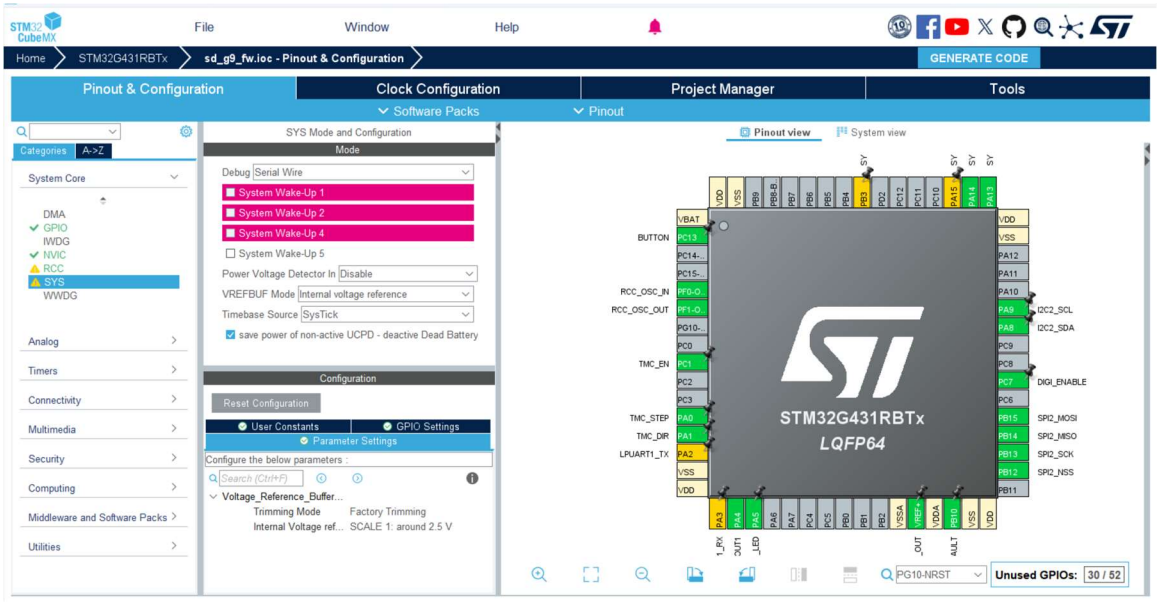


Figure 28: STMicroelectronics CubeMX GUI tool for MCU pin-planning

6.6 Stepper Motor Driver

6.6.1 Schematic Design

Our stepper motor driver design is based around the TMC2209. The parts calculated were calculated using the Analog Devices datasheet. Our component selection for each element was made through filter search of the vendor, cross reference with the datasheet, voltage ratings, current ratings, tolerances, and comparing various related designs.

As said before, we are using custom connectors in the design, but for the phase connections to the motor itself. Everything will eventually be connected through

one PCB with the MCU. This is also a Wago wire-terminal connector, giving us that direct and secure connection for bare wires of the phase cables.

Most of the global nodes inside the actual schematic will be sourced onto the MCU directly without needing connectors or cables to lower the distance needed. Our LED diodes are essential since they let visualize both the step and direction of the motor at a glance. This lets us see if anything is not working as it is supposed to. Our V_{REF} allows us to have control over internal current input, allowing the step to go fast or slow while also preventing heat buildup within the motor itself.

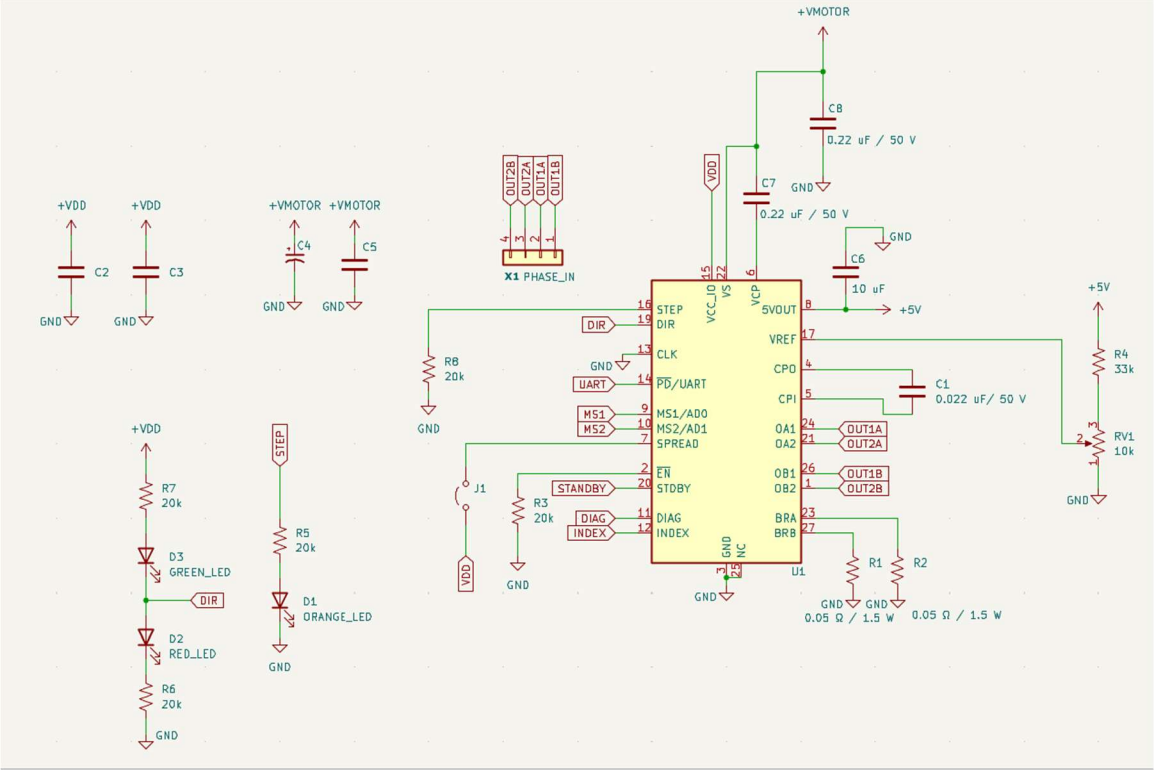


Figure 29: Schematic Design for the Stepper Motor Driver

6.7 Laser Diode Driver

6.7.1 Design Overview

The circuitry responsible for the driving of the laser diode follows a custom schematic design, detailed in this section. Our goal with this design is ultimately to create a voltage to current driving converter, to translate our voltages generated by our DAC peripheral to currents through the diode. A simple single-transistor low-side driver was developed, featuring a single current-feedback amplifier to control the transistor. The process to achieve the resulting schematic was taken through the following steps:

1. Identification of desired laser diode operation
2. Selection of current sourcing component (N-MOSFET)
3. Design of FET gate-control circuitry
4. Simulation verification
5. Schematic design and BOM

6.7.2 Specification of LD Operation

As mentioned in previous sections, a laser diode, like an LED, operates non-linearly between voltage and current. For both opto-electronic devices, a general linear trend is observed between optical output power and current. We can reference the plot of voltage and optical output vs current from the product page of our selected LD (RLD63NPC8):

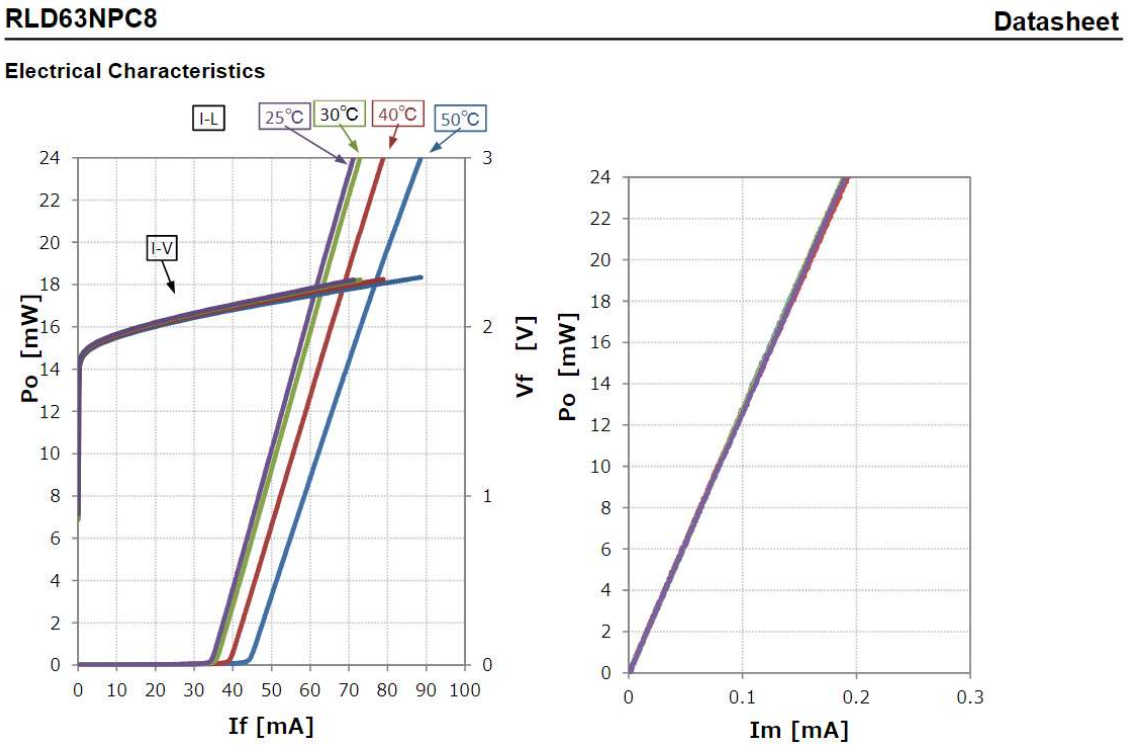


Figure 30: RLD63NPC8 Laser diode transfer curve (provided by Rohm semiconductor in the component datasheet)

To drive our diode effectively, we will need to linearly convert our input voltage from our DAC to a current along the linear region of the blue trace from the previous figure. This current will range nominally from 30mA (below the 40mA typical threshold current for the LD) up to 65mA, the LD's maximum allowable operating current. With this in mind, we move to the selection of a part capable of varying its through-current with an external input.

6.7.3 Transistor Selection

Knowing this circuit would be powered from our internal 12V bus, we chose to implement a discrete transistor to control the current from the bus through to our GND line. While the current control can be accomplished with either a bipolar junction transistor (BJT), or even an insulated gate variant (IGBT), we chose to use an N-Channel MOSFET device as it doesn't require a constant base current to operate the LD and be easily sourced. We specifically chose an N-Channel enhancement-mode FET as we wanted to control current on the low-side of our power, for ease of integration to our GND referenced microcontroller DAC signal. Our transistor would need to operate exclusively in its linear region; this heavily influenced our part choice. Our transistor needed to be selected with the following specifications:

1. Feature a continuous drain current of at least 180mA (2x for safety)
2. Feature a drain-source voltage limit of higher than 36V (from our input voltage system requirement of 9-36V)
3. Deliver low drain-source resistance across our current range to limit power dissipation by the transistor ($R_{DS,ON}$)
4. Switch fast enough to meet our 10µs current rise/fall time requirement
5. Feature a low (<4V) threshold voltage to be easily controlled by our feedback circuitry

With the above factored in, the FET was chosen as an *onsemi* **FDD86113LZ.** From this point, the appropriate SPICE model was downloaded and entered into the LTSpice simulation, covered later in this section.

6.7.4 Current Control Circuitry

The maximum current through the transistor at a given drain voltage closely follows the gate-to-source voltage V_{GS}, as an exponential trend. As not only the transistor itself presents a non-linear transfer characteristic (considering the I-V curve of the LD), open-loop control is extremely difficult and unnecessary for this application. To read the current through the transistor and thus the LD, a shunt resistor was selected for ease of implementation. The resistance of this part would directly set the current-to-voltage gain of the control circuitry. While a large voltage change from our 90mA maximum current would be ideal, we must contend with keeping the voltage drop across it minimal for the sake of the headroom for V_{GS} and the laser diode voltage In addition, a large power dissipation would be incurred through the shunt if its resistance was set too high, as power goes as I^2**R.** The value for the shunt resistor was selected such that the resistor would dissipate a maximum of 0.5W at 300mA, and more importantly, only drop 1V at 300mA. From this, a 3.3Ω resistor was selected for the shunt.

To control the current through the diode, an operational amplifier was selected and fashioned to adjust the transistor V_{GS} to match the shunt current voltage and input voltage. This amplifier would be supplied by the on-board 5V buck conversion circuitry and was experimentally selected to provide high enough slew-rate to push/pull the charge from the FET gate. To prevent the amplifier from reaching an unstable or oscillatory condition, a small resistance was added from the output of the amplifier to the gate. This added more real impedance to the amplifier's load, thus pushing the amplifier away from supplying too much reactive current in transient. **This would also help with compensating for lead inductance to the LD.** The operational amplifier selected would need to also have minimal input offset voltage to maintain accurate conversion from our input voltage and output current, with minimal temperature drift. *Rail-to-rail operation also must occur.*

With the previous considerations, the amplifier was selected as an Analog Devices **AD8656**, featuring a second channel we will likely not use and leave as a buffer connected to GND. A quick note on lead inductance, the maximum realistic lead inductance to our laser diode (22AWG, 1mm spacing, 100mm length) would be on the order of 100nH. Without the bandwidth limitation of the control loop, any significant lead inductance like this would result in the amplifier going into horrific oscillation as the pole introduced by the inductor would often lay on the positive real side. This is compensated on the control voltage input as well.

6.7.5 LTSPICE simulation

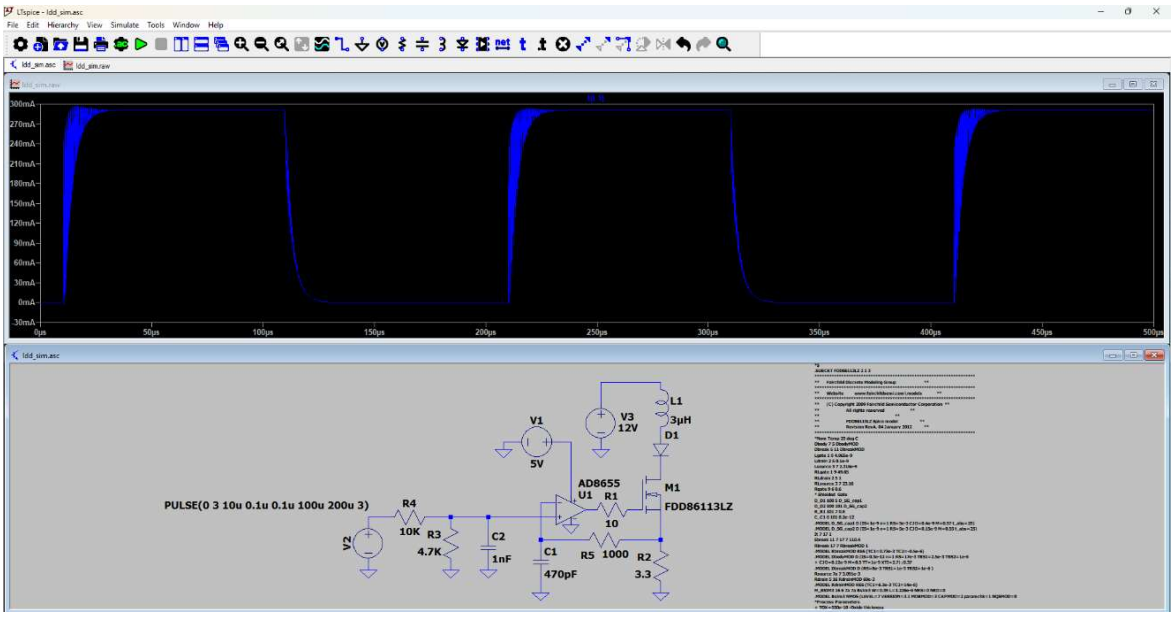


Figure 31: Laser diode driver SPICE simulation, plotting current through the LD (worst-case lead inductance)

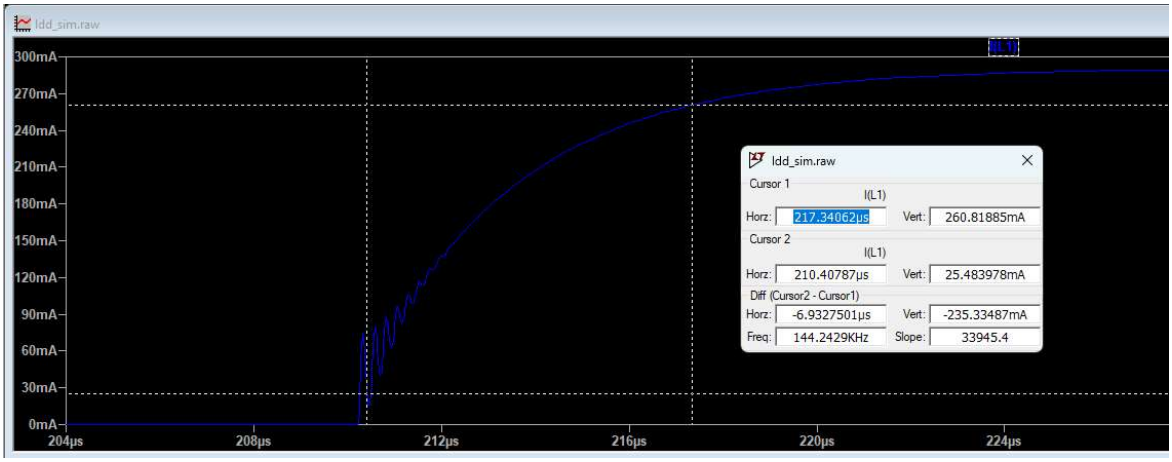


Figure 32: Laser diode current rise time measurement, hitting 7µs!

The SPICE models for the amplifier, FET, and laser diode were sourced and integrated into the simulation.

After verification in simulation, the laser diode driver circuit was able to be tuned to hit at best a 7µs 10%-90% rise/fall time. A limit on the feedback bandwidth was experimented with, using an RC filter from the shunt resistor to critically tune the transient response. With this simulation validated, the rest of the parts were chosen and fashioned into our EDA drawing and BOM table (see Appendix B).

6.7.6 Schematic Design

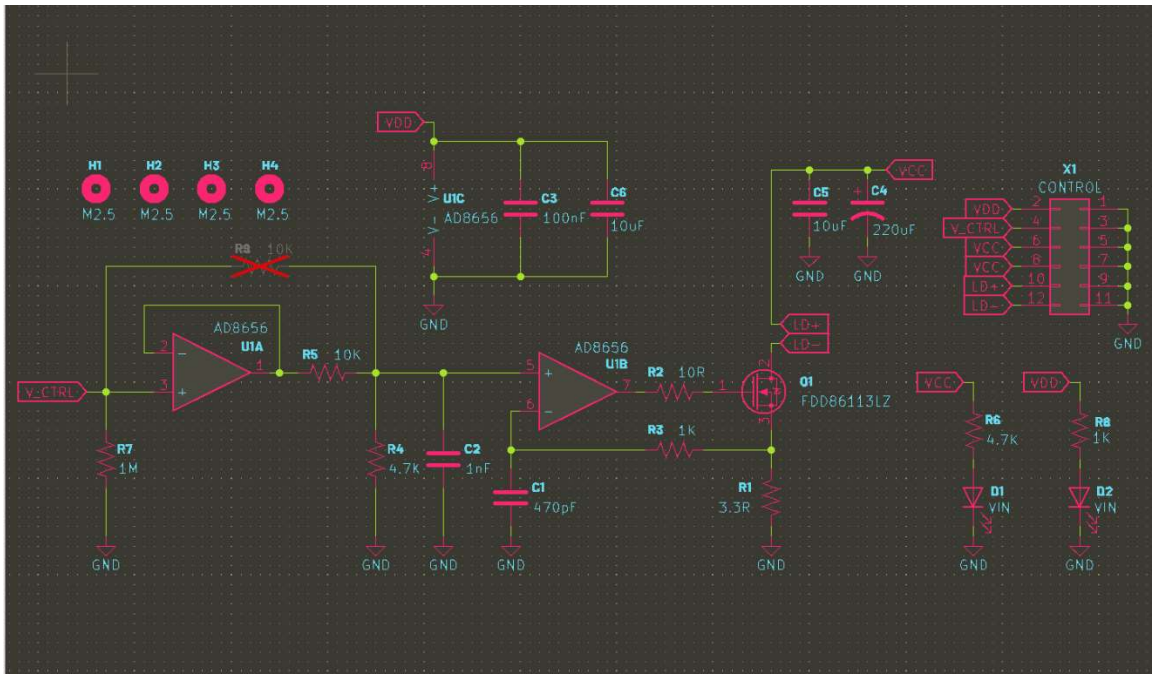


Figure 33: Implemented laser diode driver schematic

Chapter 7 - Software Design

7.1 Image Sensor Software Readout

Images are read off the sensor via the Picamera2 Python library. The camera itself is readily capable of 60 frames per second and can be stored as either .mp4 video or .png images. Files can either be saved individually or (our approach) write over each other so as to not clog memory on the Raspberry Pi.

7.2 Image Processing

Data leaving the CMOS detector will consist of a two-dimensional table of values. In a standard camera configuration, each axis will correspond to a spatial axis of the object field and each data point in the matrix is the RGB value of that pixel. In a hyperspectral camera, one of those spatial axes is traded for wavelength dependence. The RGB data of each pixel can be averaged into a monochromatic intensity as true color data is stored across the detector rather than in each point.

Two spatial axes are still needed to form a coherent image, but the system only natively records one. To capture the other axis, the camera must be scanned (linear translation) or swept (angular translation) across the scene. Many current systems are scanned [2-4] as sweeping a planar object (soil surface, conveyor belt, low-altitude landscape) requires a high depth of field as the camera traverses the scene, with the necessary depth of field given by

$$depth = d_{close}\left(sec\left(\frac{1}{2}(FOV)\right) - 1\right)$$

where d_{close} is the distance to the object in the center of the image (i.e. where the planar object is closest). As our scanned FOV is rather high, a translation approach is required as the secant taking off towards infinity makes the required depth of field unachievable with sweeping.

A standard RGB data point is 3 bytes, with one byte for each channel. When averaged to monochrome, 3 bytes can be reduced to one byte, as each channel will have the same value. The total data size of each image is therefore

$$data = 1byte \cdot p_{width} \cdot p_{height} \cdot \frac{\Delta x}{x_{step}}$$

where p_{width} and p_{height} are the pixel counts along each dimension of the detector, Δx is the range over which the camera is scanned in the object field and x_{step} is the size of each step taken in the scan. To generate a megapixel spatial image, there should be on the order of 500 to 1000 steps in the scan. Given a megapixel

detector, this generates an image data size of gigabytes, meaning that the storage capacity of any processing system must be at least several gigabytes.

Data size is not the same as resolution. True resolution can be given by

$$resolution = \frac{p_{width} \cdot d_{pix} \cdot \Delta\lambda \cdot \Delta x}{d_{spot} \cdot \lambda_{FWHM} \cdot x_{step}}$$

where d_{pix} is the pixel size, d_{spot} is the spot size, $\Delta\lambda$ is the wavelength range of the system, and λ_{FWHM} is the full-width-half-maximum spectral resolution of our diffraction grating. This resolution is determined by optical hardware rather than software but is a crucial factor and expectation to have in mind when processing the image data.

There are multiple visualization strategies for this data, all of which could be implemented when the user is given a choice between them in the interface. The first and simplest visualization strategy is to display monochrome images for a user input wavelength, similar to how H-alpha astronomical images are displayed. The images can be singly colored according to selected wavelength and allow for investigation of the distribution of specific spectral components. A second strategy would be the overlay of several images generated by the first method. The user could select spectral regions and see the image scaled for these regions specifically.

A third method would be the generation of custom false-color images. Our system will include near-infrared wavelengths that don't correspond to visible colors. These wavelengths appear on the detector as a light purple due to the spectral response of the Bayer filter, but recoloring should be possible. Recoloring of any wavelength, including visible, will allow for colors in the image to correspond to chosen compounds (similar to element-wise coloring in astronomical images and geological coloring in Earth surveys).

7.3 Instrument UI Design

7.3.1 Target UI Design

As a proof-of-concept, we have formulated a basic idea for what our user interface should look like. As this is highly subject to change, the main focus with this diagram is the basic functionality provided. Our instrument's UI should be fully capable of delivering the following:

- The user should be able to start and stop a scan with the imager
- Provide ability to modify the optical and mechanical parameters of scan
- Provide ability to modify properties of the imager (CMOS settings)
- Customize settings within the rendering software beneath the UI

- Produce false-color images and read per-pixel spectra when selected
- Allow the user to translate through the wavelength axis via a slider
- Produce greyscale images at the selected slider wavelength dynamically

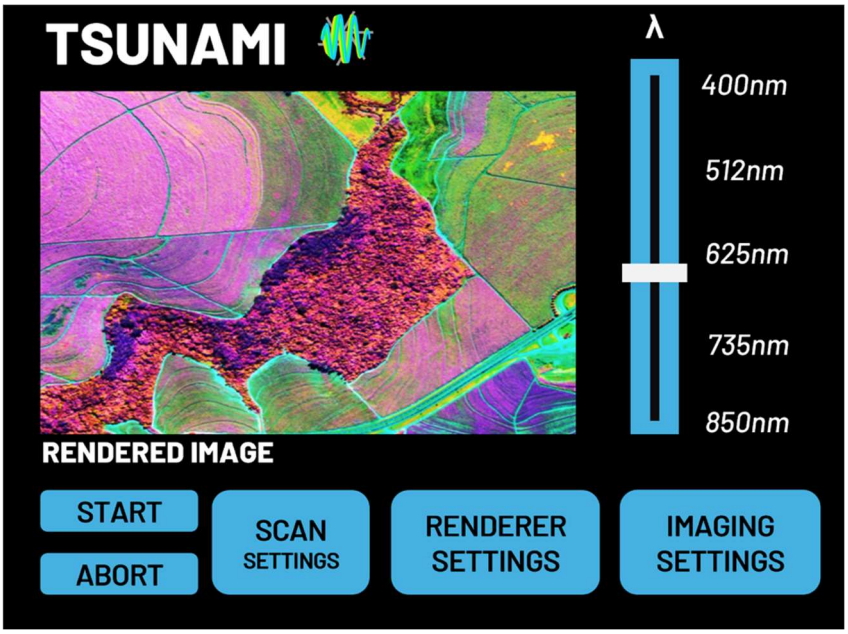


Figure 34: General concept for final user interface on SBC touchscreen

7.3.2 SD1 Final Demonstration UI Design

We presented a demonstration of our optical design to Dr. Kar, Dr. LiKamWa, and Dr. Hagan on 18 November 2025. Part of this design was a simple test user interface. While very barebones, this UI demonstrated the compatibility of the Picamera2 and Tkinter libraries into a Raspberry Pi user interface capable of data collection and display.

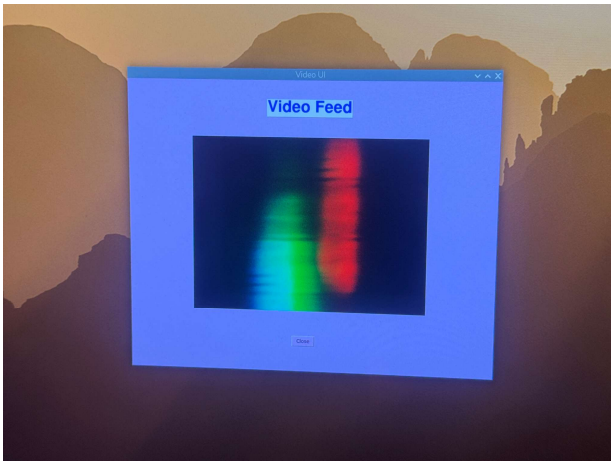


Figure 35: User Interface as of 18 November 2025

The interface displays a live feed of the camera sensor and allows for the ending of the program from within the interface. Further interactivity wasn't needed for this demo.

7.4 MCU Firmware

7.4.1 Overview

The firmware for the STM32G431 is primarily written across four distinct .c and .h file pairs. Importantly, because we are using an STMicroelectronics MCU, the CubeMX project generation tool creates the directories and base files needed to operate the MCU, such as main.c/h as well as generating all the HAL API drivers for all low level peripherals. With the project structure, low level drivers, and CMAKE build files in each directory, CubeMX does a pretty good job setting up where all the basic sections should be set. The file main.c contains within it a great deal of structural code that is necessary to be worked with to allow for the HAL API to work correctly. All of our user startup code would go within the main() function as well as our app_main loop within the looping while(1) CPU poll. Other function calls such as the interrupt service routines (ISRs) for various peripherals would either be provided by CubeMX in main.c or manually added depending on the implementation. Within the Src/ and Inc/ directories, we wrote our four custom files; app_main, TPS55288, laser, and motor.The first of these, app_main, would host the system's master structure for all state variables. App_main would also contain mathematical method functions as well as other functional methods for handling high-level control with things like SPI communication actions and motor control.

Our custom libraries were written to be independent of any dependencies from app_main, functioning solely as references for external peripheral specific things such as control register addresses, their bitfields with their corresponding values,

and the function prototypes for all functions defined in the .c file. Only the HAL API and main.h headers would need to be referenced in these, passing all parameters ultimately by reference to these libraries. In doing so, these libraries could provide a unidirectional data flow and ease implementation by not having crossing dependencies, resulting in an external variable tree mess. Provided below is a reference block diagram showing our firmware files and their dependencies based on arrow direction.

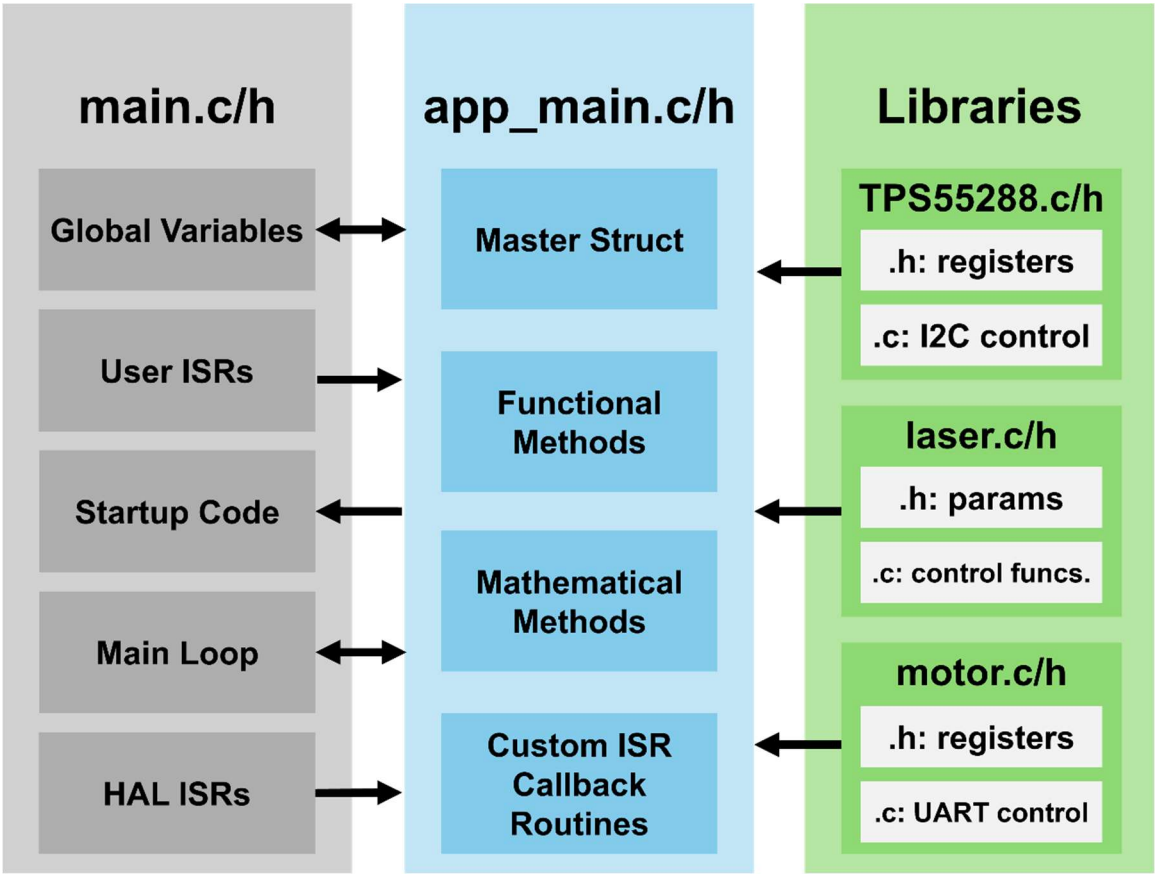


Figure 36: MCU firmware block diagram (files and their structures)

The clocks of all peripherals and other computing structures within the G431 were also very much importantly considered. On the development board we procured, this clock source was already provided as a 24MHz stable crystal reference, to which our RCC peripheral would generate the master clock from. This master clock would then be internally multiplied by a PLL near the CPU to generate the CPU clock. The CPU clock is then divided and distributed through the rest of the clocking structure to provide synchronous clocks to all hardware peripherals and timers on the master data buses such as APB1 and APB2. Some peripherals such as the UART, I2C, and ADC feature their own local clock multiplexers, able to select clocks from other locations in the system besides just their bus clock sources. All

of this is configured through the clock tree management GUI within the CubeMX software panel. Below is a figure showing the configuration just described:

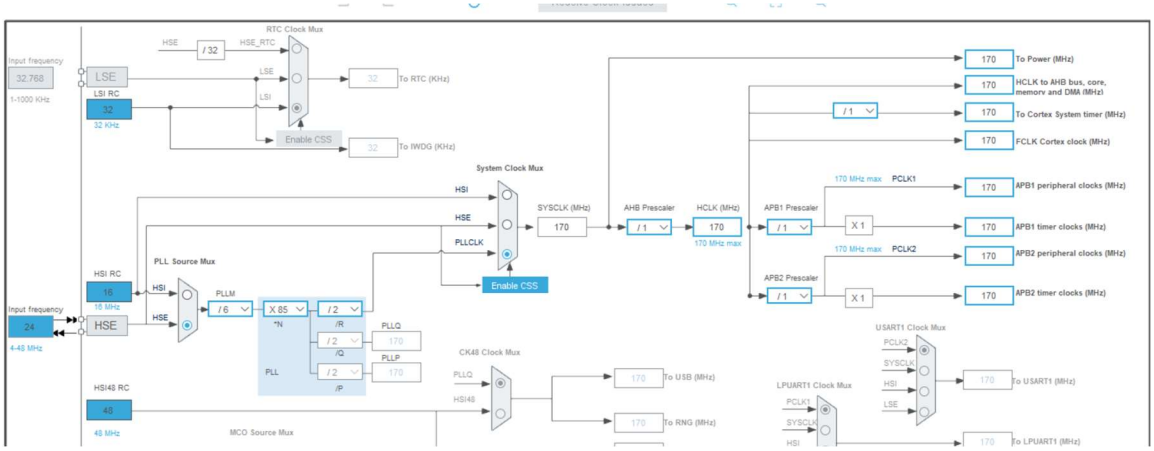


Figure 37: Clock tree of the G431 MCU, taken from the CubeMX .ioc project file

7.4.2 CubeMX Code Generation (Startup)

The CubeMX code generation system uses the STMicroelectronics HAL API as described in section 3.5.2. The CubeMX tool allows for pin-planning of the hardware chip itself, but also more importantly the configuration of peripheral settings without needing to learn all peripheral registers in excruciating detail. Modern MCU peripherals such as the ones featured in our G431 are highly feature rich, with a quite extensive set of registers needed to be configured correctly to operate properly. As a result, the CubeMX tool will automatically generate user-specific peripheral instantiation code for the MCU's start, using the handler structs of each peripheral in the HAL API. This will configure parameters for things such as timer CCR periods, SPI clock divisions, interrupt locations and priorities, among many others. We specifically chose to use this code generator to create this instantiation code for our hardware peripherals and all lower-level needed access.

Table 29: CubeMX generated function calls (user customized)

Generate Code	Rank	Function Name	Peripheral Instance Name	Do Not Generate Function Call	Visibility (Static)
☑	1	SystemClock_Config	RCC	☐	☐
☑	2	MX_GPIO_Init	GPIO	☐	☑
☑	3	MX_DMA_Init	DMA	☐	☑
☑	4	MX_DAC1_Init	DAC1	☐	☑
☑	5	MX_I2C2_Init	I2C2	☐	☑
☑	6	MX_SPI2_Init	SPI2	☐	☑
☑	7	MX_UART4_Init	UART4	☐	☑

7.4.3 Custom written drivers

As mentioned in section 7.4.1, we took liberty of writing a few custom drivers for the particular external peripherals we were interested in. For the TPS55288 in particular, no existing driver was available on the internet for the IC, let alone a

HAL compatible one. We took it upon ourselves to map out each register of the TPS55288 and learn the bitfields and operational modes of each part of the chip. Doing this not only made integrating the HAL specific I2C functions for transmitting and receiving very easy, but also gave us a much richer understanding of the chip's operational modes by reading the literature properly. While the laser diode control methods weren't particularly accustomed to a particular external chip, the linear voltage to current conversion (and optical power calculation) were done there. The TMC2209 motor driver's specific configuration could be accessed via registers over single-wire TX UART from the MCU to the IC. Much like the TPS55288, the register locations and their bitfields would be addressed, just using the HAL UART functions instead of the HAL I2C functions.

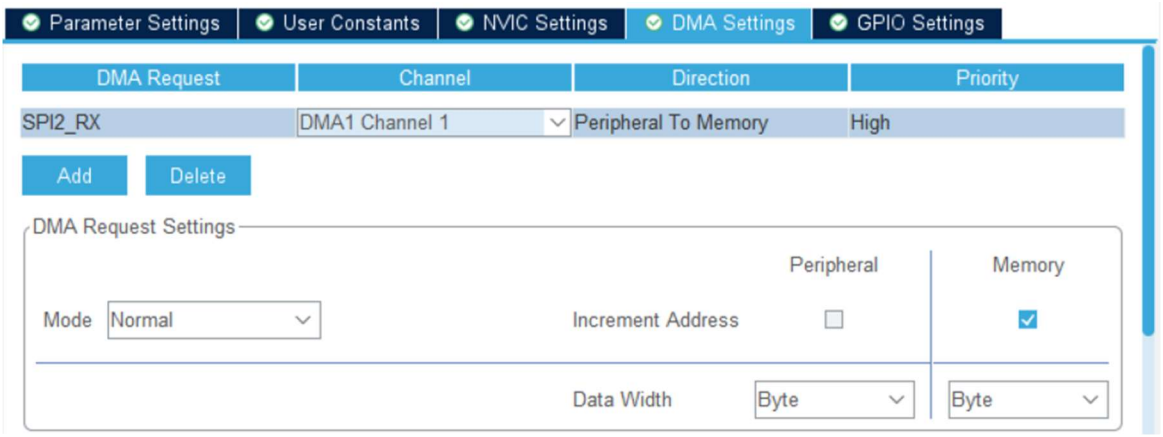
The code for the TPS55288 driver in particular is given in appendix C.

7.4.4 MCU to SBC communication

The communication from SBC to MCU would be implemented as a DMA-enabled SPI slave on the side of the MCU. The DMA or direct-memory-access is a critical peripheral used to offload duties of data transfer from memory to memory, or peripheral/memory. In our case, the SPI2 RX DMA stream would be configured to automatically transfer data bytes from each SBC communication frame into a buffer within the RAM of the MCU. Once completed after reception, an interrupt call would be triggered by the DMA to the CPU to take action regarding the incoming command bytes.

To implement this, the 8-bit SPI transaction frames would shuffle from the peripheral data register into the chosen RAM address (chosen in software configuration) by using a peripheral-to-memory specific transfer. The data pointer to the memory address requested would be incremented by the DMA automatically.

Table 30: DMA configuration for SPI2



Chapter 8 - PCB Design and Layout

8.1 Motor Driver Power Supply Board

Following the design of the motor driver PSU in chapter 6.3, a PCB layout was approached. For the TPS55288 (and nearly all switchmode PSUs), careful consideration in the layout of parts is of utmost importance when attempting to produce a design that works with all advertised and designed specifications. Failure to do so in a way that minimizes potential unwanted effects will result in a design that either does not work at all or performs poorly/erratically. In applications engineering, specialist engineers working for the companies that sell commercial regulator ICs are typically tasked with extensive testing to find the best possible layout arrangements for these parts and their satellite components.

As these engineers are typically *very* experienced, their best recommendations after much testing is to be absolutely considered and followed exactly. These layout recommendations are usually detailed within the datasheet of a particular regulator IC. With most switching regulator ICs, the trace/copper pour that connects the switching node of the IC to the inductor is typically the most sensitive to misrouting. As this node contains a lot of high-frequency energy with large current spikes through the inductor, careful consideration as to its arrangement and what parts/traces are allowed near it is very important. Aligning with this, planning of ground planes is critical to ensure that no unwanted return paths between sensitive control circuitry and large power signals align.

TI recommends that for the TPS55288, a four-layer stack-up be used in the board design. The top copper and bottom copper layer should be reserved for signaling and large copper pours for power connections. The inner layers are recommended to be a solid ground plane for the power circuitry, and a separate ground plane for the analog/control circuitry on the other. TI specifically mentioned only connecting the analog ground plane to the power ground plane at one point, right on the return area of the VCC decoupling capacitor of the IC. This will ensure all of our noisy power signals return to ground imparting minimal interference to the control circuitry.

In the design of the board outline and mechanical integration, we wanted to keep the solution size at a minimum while also maintaining plenty of space around parts and keeping to larger part sizes. Standard part size codes of 0805 and larger were used to ensure easy soldering, good thermal performance, and easy debugging of this first prototype later if needed. Four M2.5 mounting holes were added to the board corners to make mounting during testing much easier with standoffs.

Connectors were chosen to be featured at the edge of the board, as this would make connections to the other development hardware easy and less cluttered.

Extensive silkscreen markings were added to call out reference designators of each part, as well as on-board pinouts of all connectors. Some other auxiliary silkscreen references were added, such as a mounting hole diameter callouts and information about our project and team members.

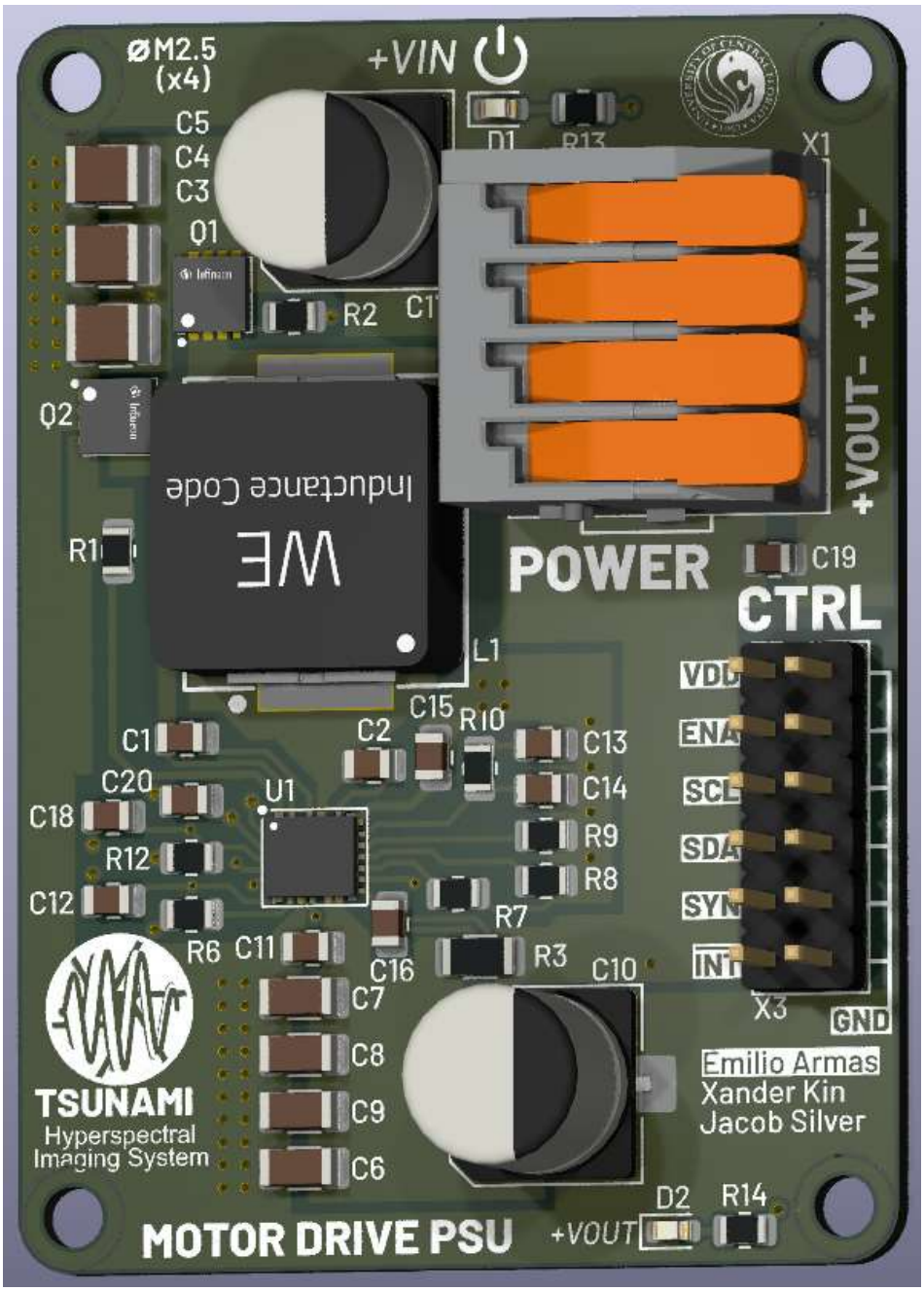


Figure 38: 3D render of finalized motor driver PSU circuit board

Motor Drive PSU PCB Per-Layer Drawings:

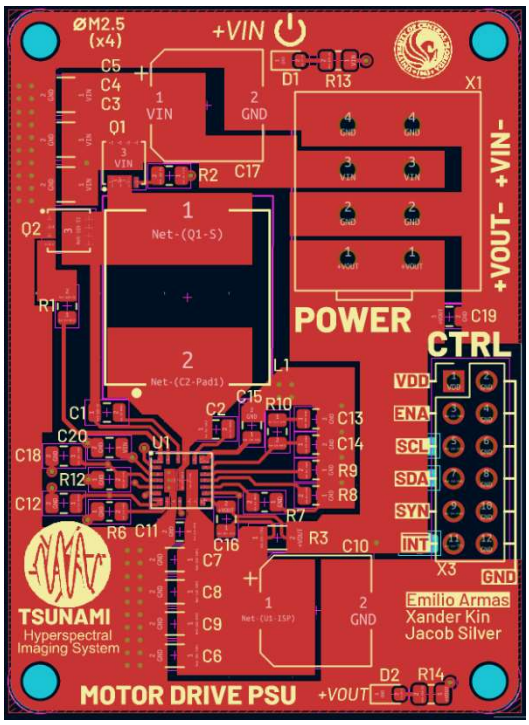


Figure 39: Top copper + silkscreen

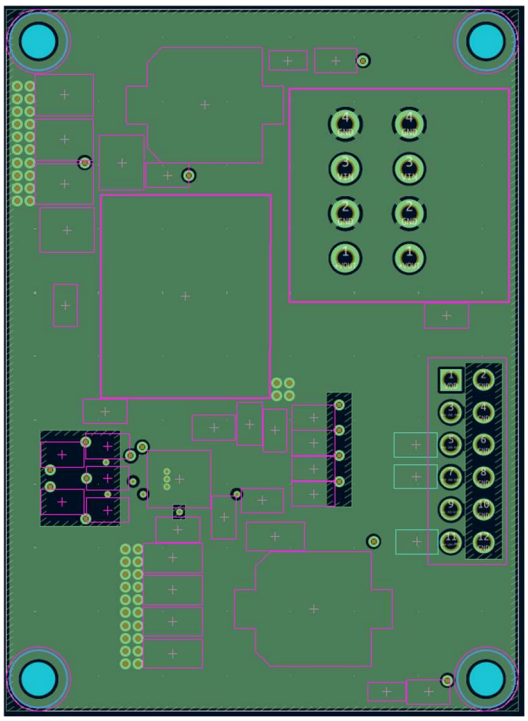


Figure 40: Inner Copper 1 (PGND)

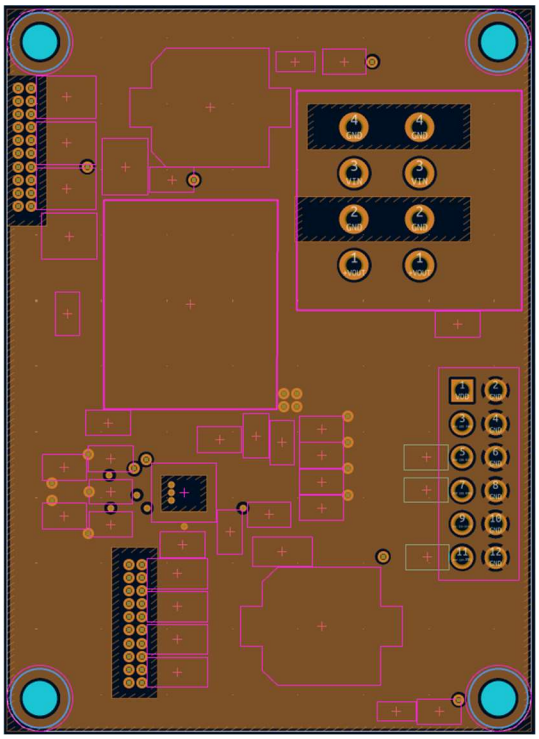


Figure 41: Inner copper 2 (AGND)

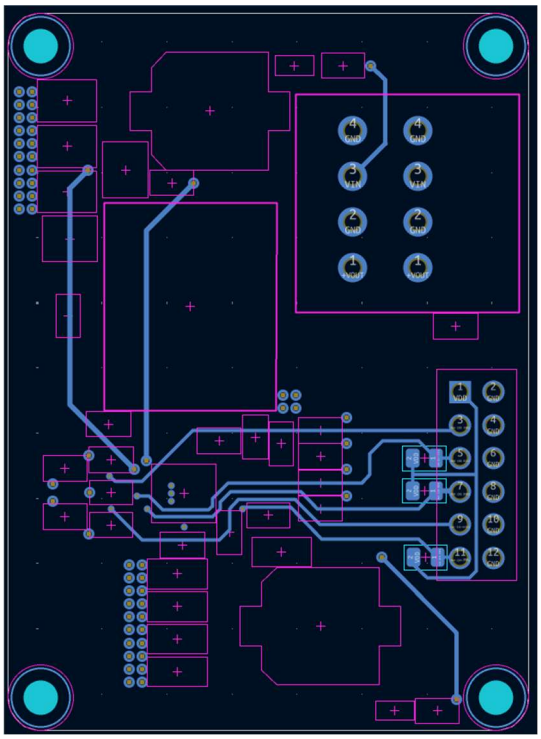


Figure 42: Bottom Copper (extras)

8.2 SBC Power Supply Board

From the schematic of the SBC power supply seen in chapter 6.3, a test-PCB for our DC/DC converter layout was designed. Again, following the design procedure detailed in the motor drive PSU board layout, many of the same recommendations were heeded. With this particular IC, the manufacturer datasheet and the provided layout of their evaluation kit did differ quite significantly. Due to this large discrepancy, we met with faculty advisor Dr. Arthur Weeks to discuss the layout, and we agreed on best practices for arrangement of the top parts and trace layout.

TI again recommends that for the TPS54541 part, a four-layer stack-up be employed in the PCB design. In this design, we used both of the inner copper layers tied together as the central ground planes, providing not only a return path for all the signals above and below, but also as a large thermal mass. Besides ease of assembly and convenience, we purposely oversized our board to provide for a larger thermal dissipation volume as this PSU would be under constant 2-3A load from the SBC's operation. Small inefficiencies will result in considerable heat dissipation and must be accounted for in the design.

One node of particular interest was the switching node of the power inductor. We kept this node's length and area minimal to reduce parasitic effects seen from other nearby traces and the PCB itself. Too much capacitive loading or too much extra inductance to the IC could result in the buck regulator IC failing to start up. Similar design practices for the mechanical arrangement were followed from the previous PCB.

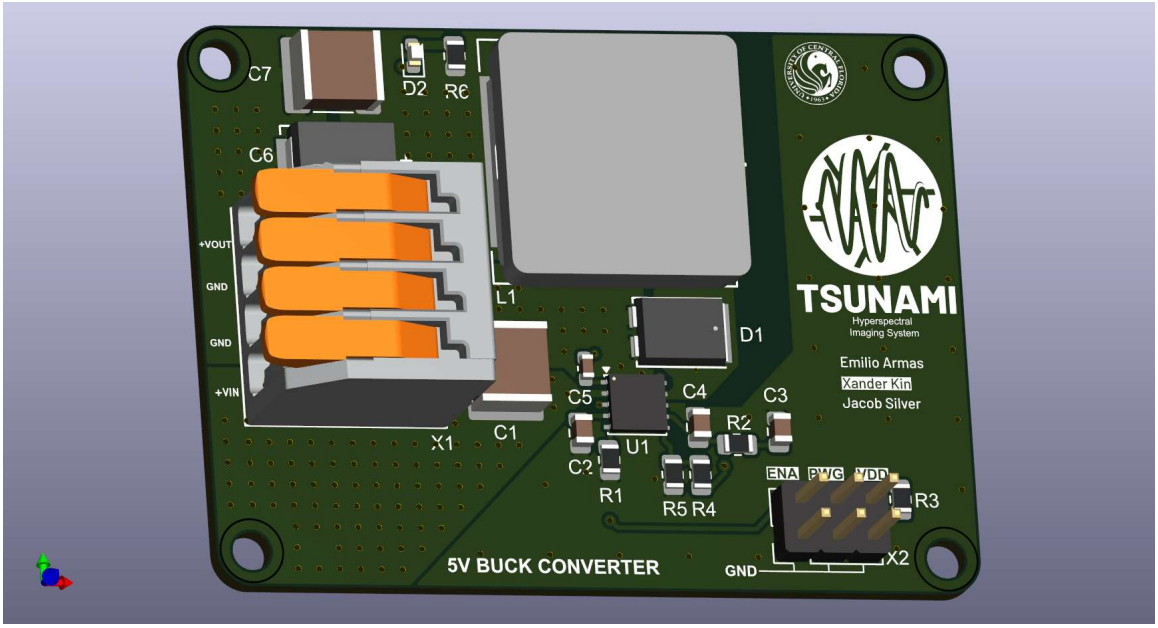


Figure 43: SBC PSU 3D render output

SBC PSU PCB Per-Layer Drawings: (inner two GND layers omitted, filled entirely)

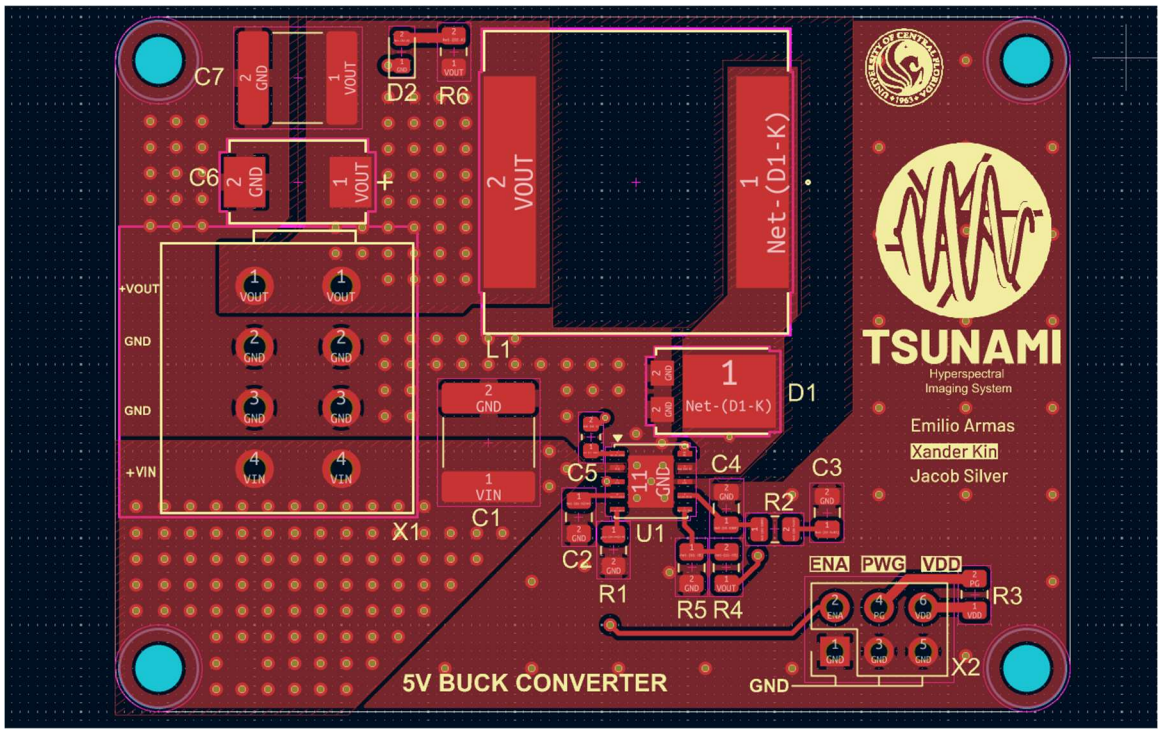


Figure 44: SBC PSU top copper + silkscreen

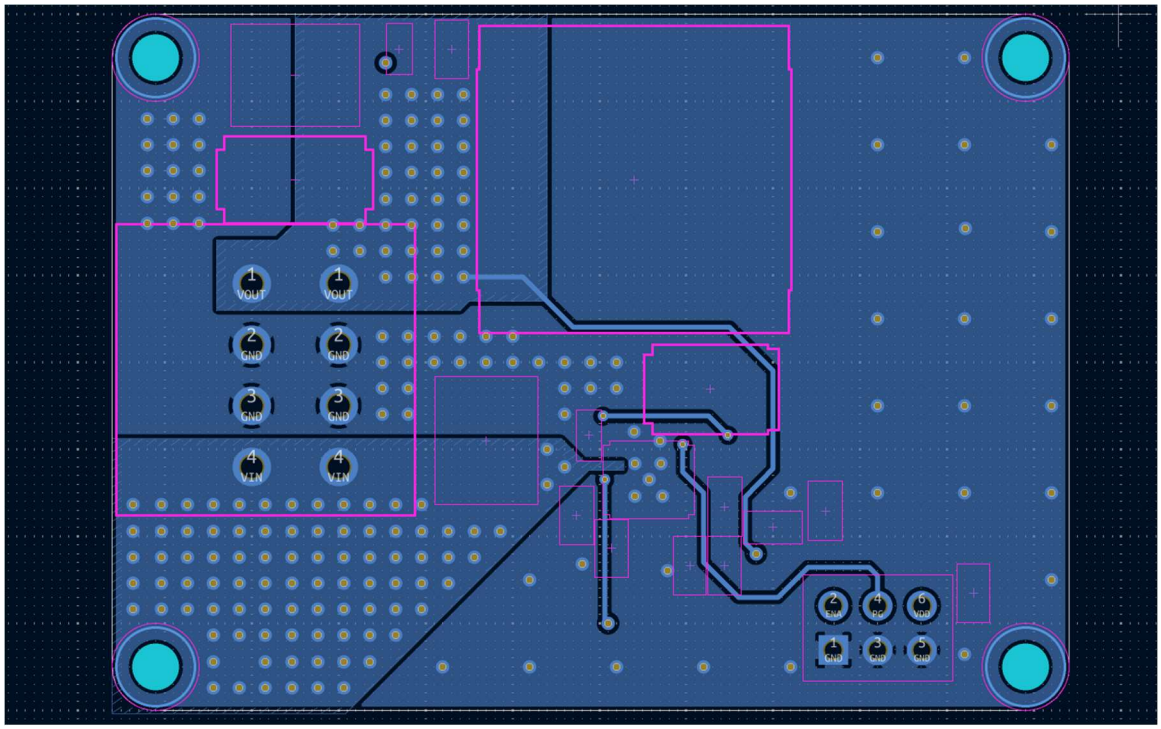


Figure 45: SBC PSU bottom copper layer

8.3 Laser Diode Driver Board

Following the design of the laser diode driver from section 6.7, a PCB layout was approached and completed to test the design. The layout centered around the driver MOSFET, as well as the large bulk decoupling capacitor designed to serve as a charge reserve for the laser diode. Only two layers were needed to effectively lay-out all parts, as not many connections needed to be crossed, and current handling capability was not to exceed 300mA at a maximum in any part of the circuit. The bottom layer served as a continuous ground plane to dissipate heat and provide a solid return path for all signals. Large copper pours were run from the FET and +VLD line back to the master connector. This connector was chosen as an interface for the power and control signals. A 2.54mm pitch connector with ground pins for each connection line would make interfacing during development easy with rectangular jumper wires in testing. Power LEDs were fashioned to provide indication for both the op-amp power rail and the diode +VLD rail.

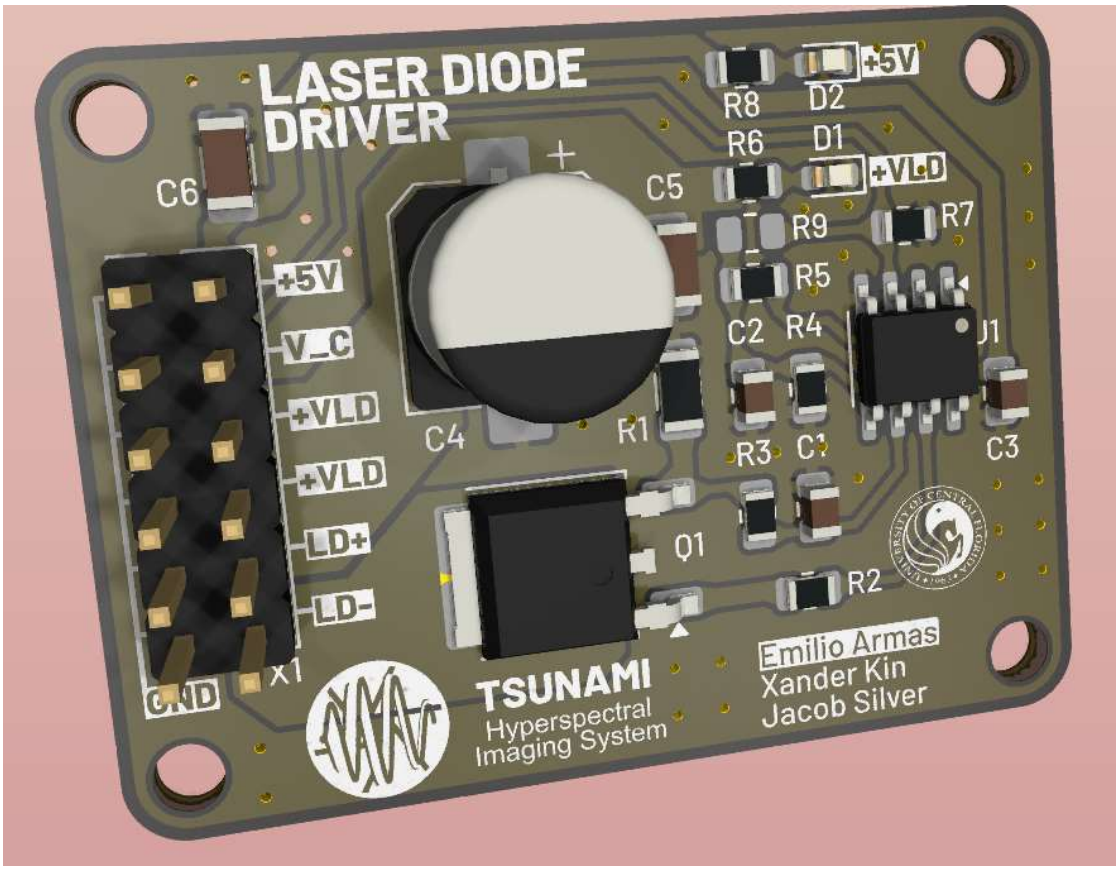


Figure 46: 3D render of laser diode driver PCB

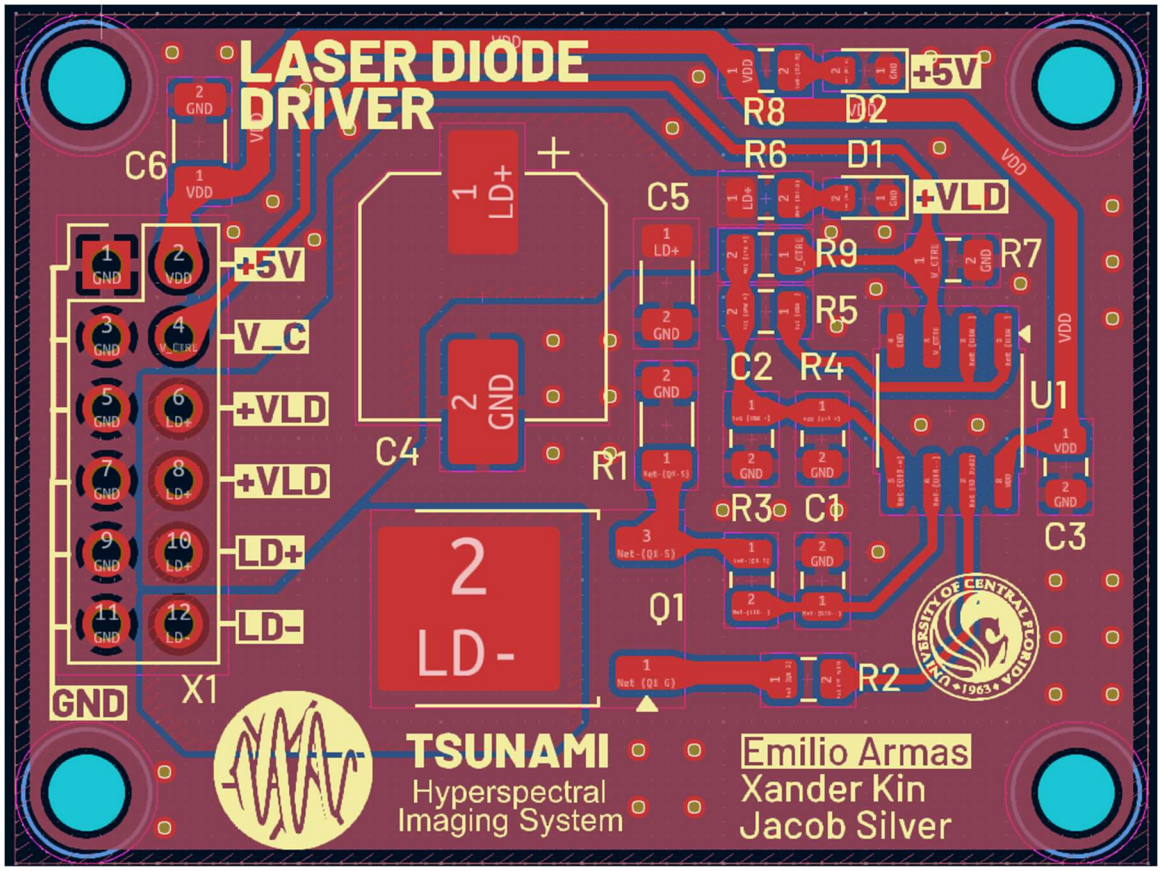


Figure 47: Laser diode driver PCB top/bottom layer in 2D (top: red, bottom: blue)

Chapter 9 – Testing and Results

9.1 Optoelectronics Feasibility Study & Testing

9.1.1 Optoelectronics Feasibility Study

We initially tested the design of our constant current laser diode driver in simulation. A test simulation in a real-time circuit simulator was conducted with rise/fall times better than 10µs. A comprehensive simulation with exactly chosen parts and their models was eventually conducted in LTSpice following these results. This circuit was built with similar parts on a breadboard for the CREOL midterm demo and showed 1.8µs and a ~6µs fall time. Despite this success, more work had to continue in simulation to harden the circuit to be more reliable and to account for the appropriate scaling for the input from the MCU DAC.

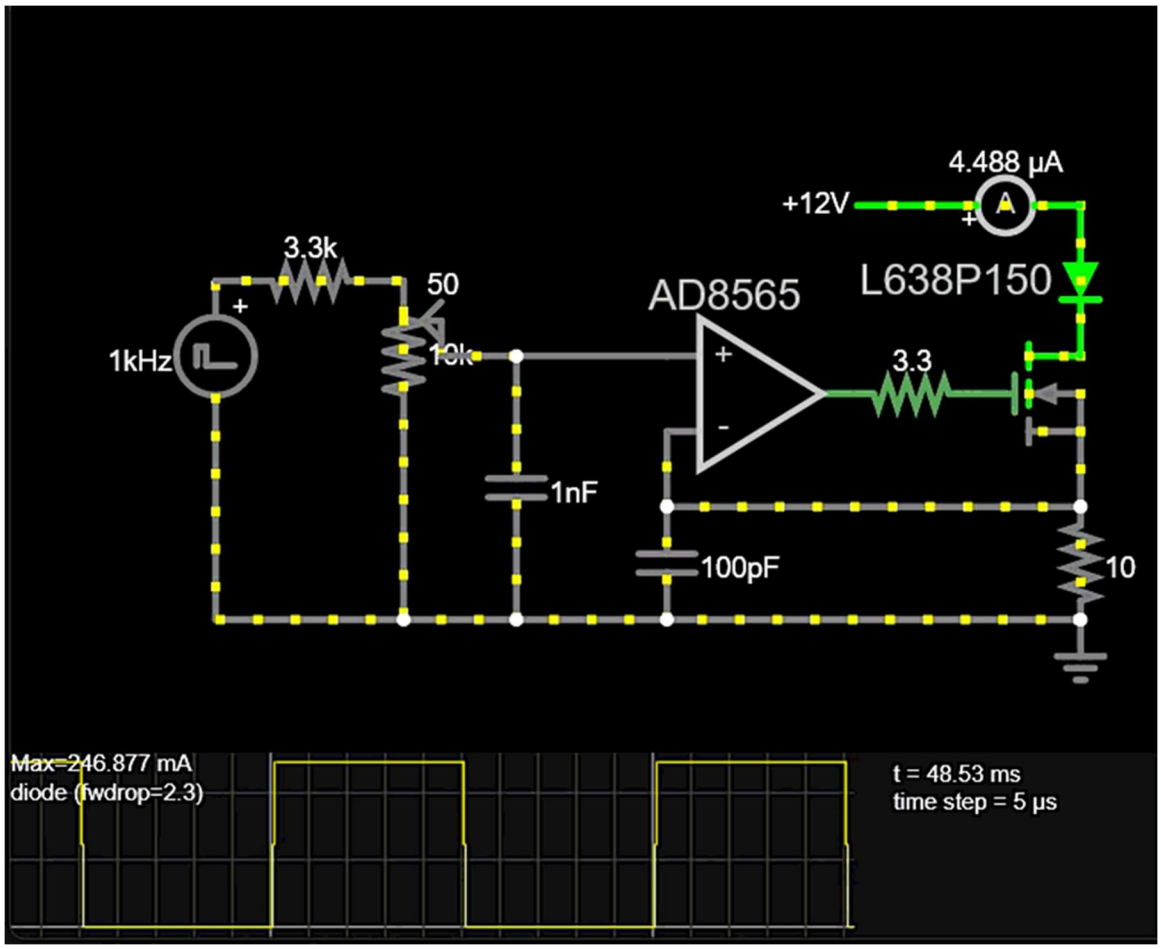


Figure 48: Laser diode driver initial simulation (before proper LTSpice design)

9.1.2 Optoelectronics Testing

Thus far, our team has succeeded in testing our CMOS camera sensor module with our SBC. We installed the operating system to the SBC's storage media and were able to successfully load the operating system on startup. We were able to configure and utilize the *rpicam* kernel driver on the SBC's operating system. In testing, we were successful in retrieving a test video feed and some test images from the unit. Our camera module hardware was attached to the on-board MIPI-CSI-2 port.

As we were testing with just the bare sensor (we didn't procure a lens as our imager optics were to obviously be used), the images received weren't amazing as nothing was focused. However, general responsivity to light and dark were observed when changing the light incident onto the sensor. This proved to be a great motivator to continue faster with finalizing the optical design.

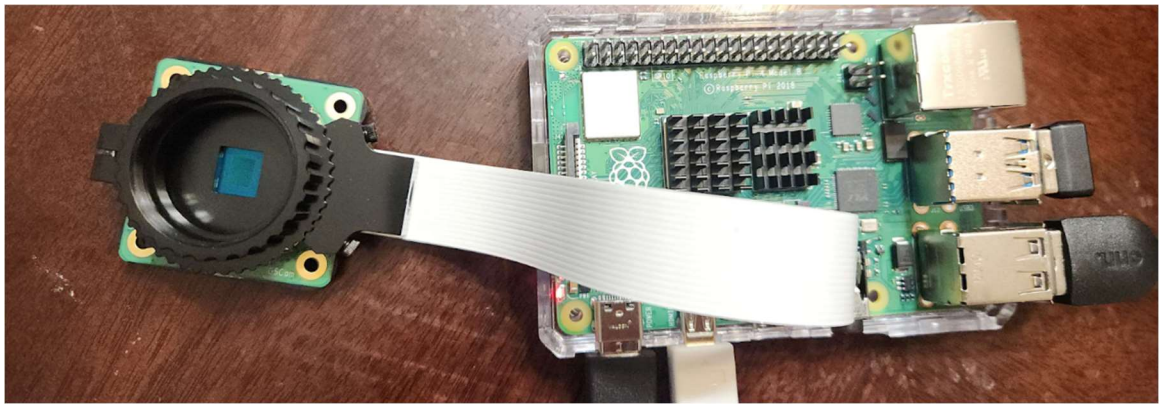


Figure 49: Global Shutter CMOS sensor module attached to SBC

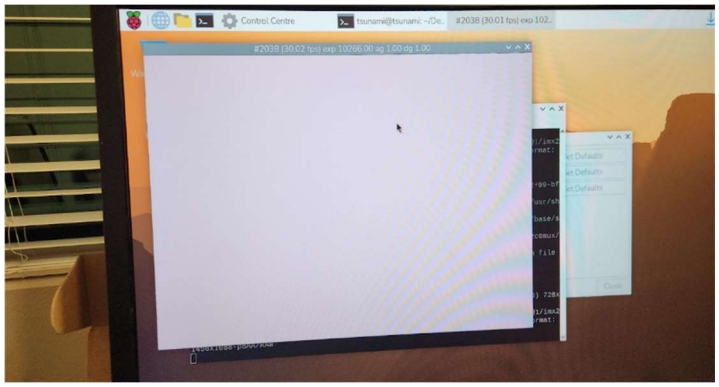


Figure 50: Resulting test image from the CMOS sensor module, no attached lens results in a completely defocused but still light-sensitive behavior.

Chapter 10 – Administrative Content

10.1 SD1 Administrative Milestones

Table 31: SD1 Administrative Milestones

#	Milestone Description	Start Date	Anticipated End Date	Duration
1	Finalize Project Idea	08/19/25	08/26/25	1 week
2	Finalize General Design Choices (Crucial Hardware)	08/19/25	09/05/25	2 weeks
3	D&C Finalization	08/28/25	09/05/25	1 week
4	D&C Meeting	09/09/25 11:30am	09/09/25	1 day
5	Committee Formation/Meeting	08/28/25	09/05/25	1 week
6	30-Page Draft	09/05/25	09/26/26	3 weeks
7	60-Page Draft	09/26/25	10/17/25	3 weeks
8	Midterm Milestone Report	10/17/25	10/31/25	1 week
9	Midterm Milestone Revision	10/31/25	11/07/25	1 week
10	90-Page Draft	10/17/25	11/07/25	3 weeks
11	Final Report	11/07/25	11/25/25	3 weeks
12	Mini Demo Video	11/07/25	11/25/25	3 weeks

10.2 SD1 Project Milestones

Table 32: SD1 Project Milestones

#	Milestone Description	Start Date	Anticipated End Date	Duration
1	Individual System Design	09/04/25	10/02/25	4 weeks
2	Individual System Testing	10/02/25	10/30/25	4 weeks
3	Breadboard Prototyping or Simulation	10/30/25	11/20/25	3 weeks
4	PCB Design/Ordering	11/20/25	12/11/25	3 weeks

10.3 SD2 Project Milestones

Table 33: SD2 Project Milestones

#	Milestone Description	Start Date	Anticipated End Date	Duration
1	PCB Testing	01/05/26	01/16/25	2 weeks
2	System Testing	01/16/25	02/13/26	4 weeks
3	Project Demo	02/13/26	02/27/26	2 weeks
4	Final Documentation	02/27/26	03/13/26	2 weeks
5	Final Presentation Practice	03/13/26	03/20/26	1 week
6	Final Presentation	TBD	TBD	TBD

10.4 Work Distribution Table

Below is a table detailing the assigned tasks to the group members. Each system is entrusted to specific members, with some blending due to how they must work together. Collaboration is set at every level allowing these systems to fit with other

system requirements. This allows all members to be aware of and involved in the design and implementation of the systems, whether they were directly or indirectly responsible for it.

Table 34: Work Distribution Table

Electrical Engineering	**Responsibilities**
Emilio Armas	*Project Lead*
	MCU Integration and Implementation
	Motor Controller Power Supply Design
	Laser Diode Driver Circuit/Optoelectronics
	Final PCB design and assembly
	Power Bus and Power Entry Circuitry
Xander Kin	MCU 3.3V LDO Implementation
	Stepper Driver Implementation
	SBC Power Supply Design
	Subcircuit pre-validation PCB design
Photonics Science and Engineering	**Responsibilities**
Jacob Silver	SBC Image Processing Software
	Imaging System Optical Design
	Motor Mechanical Implementation
Emilio Armas	Laser Cursor Optical and Optoelectronic Design

10.5 Project Budget

Our project budget will be capped at $1500, agreed upon by each member for financial feasibility. This budget encompasses many aspects of the project including the PCB, microcontroller, optics, and motors. The bulk of our budget goes towards the high-cost items in the optics section. We also account for any replacements or last-minute changes that we might go through in unexpected

circumstances. A detailed (tentative) breakdown of costs and distribution to each of us is detailed below in the bill of materials.

10.6 Bill of Materials

Table 35: Bill of Materials

Component	Subsystem	Quantity	Unit Cost	Total
PLA Filament, 1kg	Mechincal Structures	3	$22.76	$68.28
Stepper Motor	Mechanical Structures	1	$14.00	$14.00
Transmissive Grating	Hyperspectral Camera	1	$17.76	$17.76
Achromatic Lens Doublet	Hyperspectral Camera	2	$135.09	$270.18
Positive/Negative Focal Le	Hyperspectral Camera	2	$120.00	$240.00
650 nm 7mW Laser Diode	Cursor	1	$20.17	$20.17
MCU	Electronics	1	$4.87	$4.87
CMOS Camera	Hyperspectral Camera	1	$25.00	$25.00
Miscellaneous	All	1	$100.00	$100.00
			Total	$760.26

Chapter 11 – Conclusion

TSUNAMI will be able to resolve important spectra that standard RGB cameras simply can't. This design provides a framework that we will build on as we continue out into next semester and for the final product we intend to create.

We have demonstrated through the Senior Design 1 semester that this objective is possible with a budget and readily achievable with our resources and skillset. We have routinely demonstrated the optical and physical principles upon which our system relies. We have found and integrated much of the hardware necessary to realize this system. We have pulled hyperspectral image data on a Raspberry Pi4B from a camera and optical train we designed. We have communications and power transfer between seven integrated subsystems now, with progress being made on a daily basis. We are more than ready to move into Senior Design 2 and fully realize the unit we present here.

Appendix A – Bibliography

[1] “What Is a Bayer Filter? Bayer Color Filter Array Explained.” *Arrow.Com*, Arrow Electronics, Inc., 13 June 2019, www.arrow.com/en/research-and-events/articles/introduction-to-bayer-filters

[2] “Pika L-F (420-980nm).” *Resonon Pika L-F Hyperspectral Imaging Camera*, Resonon Inc., 2025, resonon.com/Pika-L-F.

[3] Thorpe, A.K., Roberts, D.A., Bradley, E.S., Funk, C.C., Dennison, P.E., Leifer I. (2013). High resolution mapping of methane emissions from marine and terrestrial sources using a Cluster-Tuned Matched Filter technique and imaging spectrometry. Remote Sensing of Environment, 134, 305-318.

[4] Thompson, D. R., J. W. Boardman, M. L. Eastwood, R. O. Green (2017), A Large airborne survey of Earth’s visible-infrared spectral dimensionality. Optics Express 25, 9186-9195 (2017).

[5] Moorhouse, J. (2020), Construction of a Hyperspectral Imager using 3D-printed and off-the-shelf components. University of Arkansas, Fayetteville

[6] Nexstar 8SE Computerized Telescope, Celestron, LLC, www.celestron.com/products/nexstar-8se-computerized-telescope?srsltid=AfmBOoornZVgJTASBlVL8NSP1j9lHvZ9FYukl6Ojrfx3zAIH0gfajhOQ

[7] Picoult, Jeff. “22 Main Parts of a Camera & How They Work.” *MeFOTO Camera Guides*, MeFOTO, 2 June 2024, www.mefoto.com/parts-of-a-camera/

[8] Hecht, Eugene, and Alfred Zając. *Optics*. pp 99–194. Addison-Wesley, 1974.

[9] “Visible Transmission Gratings.” *Thorlabs, Inc. - Your Source for Fiber Optics, Laser Diodes, Optical Instrumentation and Polarization Measurement & Control*, Thorlabs, Inc, www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=1123&pn=GT25-03

[10] “Transmission Diffraction Gratings.” *Transmission Diffraction Grating*,

Newport Corporation, www.newport.com/f/custom-plane-transmission-gratings

[11] MIL-PRF-13830B, Department of Defense, 9 Jan 1997.

[12] “IPC-2221A(L).pdf.” Accessed: Oct. 31, 2025. [Online]. Available: https://www.eng.lbl.gov/~shuman/NEXT/CURRENT_DESIGN/TP/MATERIALS/IPC-2221A(L).pdf

[13] *Ti*, www.ti.com/lit/an/sbaa565/sbaa565.pdf?ts=1741852202490

[14] “Serial Peripheral Interface (SPI).” *Serial Peripheral Interface (SPI) - SparkFun Learn*, learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all

[15] *UM1725 Description of STM32F4 Hal and Low-Layer Drivers*, www.st.com/resource/en/user_manual/um1725-description-of-stm32f4-hal-and-lowlayer-drivers-stmicroelectronics.pdf

Appendix B – Bill of Materials

Table 24: SBC PSU Bill-of-Materials (BOM)

Component Designators	Qty	Footprint	Value(s) or Part #s	Digikey Part Numbers
C1	1	2220	10uF	445-5212-1-ND
C2	1	0805	2.7nF	490-1630-1-ND
C3	1	0805	5.1nF	490-1635-1-ND
C4	1	0805	120pF	1276-1201-1-ND
C5	1	0603	100nF	399-C0603C104K5RACTUCT-ND
C6	1	Non-Standard	220uF	P123509CT-ND
L1	1	Non-Standard	33uH	541-1344-1-ND
R1	1	0805	953K Ohm	10-ERA-6AEB9533VCT-ND
R2	1	0805	21K Ohm	311-21.0KCRCT-ND
R3, R5	2	0805	11.5K Ohm	P11.5KDACT-ND
R4	1	0805	60.4K Ohm	RMCF0805FT60K4CT-ND
D1	1	D2PAK_STM	Schottky	497-10993-1-ND
U1	1	WSON-10-EP	TPS54541	296-40802-1-ND
X1	1	Non-Standard	2601-1104	2946-2601-1104-ND

Table 25: Motor Driver PSU Bill-of-Materials (BOM)

Component Designators	Qty	Footprint	Value(s) or Part #s	Digikey Part Numbers
C1, C2, C16, C18	4	0805	100nF	490-16471-1-ND
C3, C4, C5	3	1210	10uF	445-CGA6P1X7R1N106M250ACCT-ND

C6, C7, C8, C9	4	1206	10uF	445-180362-1-ND
C10	1	8mm x 10.5mm	220uF	P15437CT-ND
C11, C19, C20	3	0805	1uF	490-10743-1-ND
C12	1	0805	10nF	490-GCM2195C1K103FA16DCT-ND
C13	1	0805	4.7nF	490-1634-1-ND
C14	1	0805	100pF	399-C0805C101J5GACTUCT-ND
C15	1	0805	4.7uF	490-GRM21BR71C475KE51KCT-ND
C17	1	8mm x 10.5mm	68uF	P15445CT-ND
D1, D2	2	0603	(Orange)	475-LOQ976.01-P2R2-25-1-20-R18CT-ND
L1	1	Non-Standard	4.7uH	732-784373865047CT-ND
Q1, Q2	2	TDSON-8-32	N-CHAN	IPZ40N04S5L4R8ATMA1CT-ND
R1, R2	2	0805	1 Ohm	541-1813-1-ND
R3	1	1206	0.01 Ohm	283-MSMA1206R0100FCMCT-ND
R4, R5, R11, R13, R14	5	0805	4.7K Ohm	541-4131-1-ND
R6	1	0805	49.9K Ohm	13-RT0805FRE0749K9LCT-ND
R7, R9, R12	3	0805	24.9K Ohm	RMCF0805FT24K9CT-ND
R8	1	0805	150K Ohm	YAG3409CT-ND
R10	1	0805	56.2K Ohm	311-56.2KCRCT-ND
U1	1	RPM0062A	TPS55288	296-TPS55288RPMRCT-ND
X1	1	Non-Standard	2601-1104	2946-2601-1104-ND

X3	1	IDC 02x06	Pinheader	10129381-912001BLF-ND

Table 26: Microcontroller sub-system BOM (shared parts with 6.6, 6.7)

Table 27: Stepper Driver sub-system BOM (shared parts with 6.5, 6.7)

Table 28: Laser Driver sub-system BOM (shared parts with 6.6, 6.7)

Component Designators	Qty	Footprint	Value(s) or Part #s	Digikey Part Numbers
U1	1	SOIC-8	AD8656ARZ	AD8656ARZ-REELCT-ND
Q1	1	TO-252-3	FDD86113LZ	FDD86113LZFSCT-ND
C1	1	0805	470pF	478-KGM21BCG2A471FTCT-ND
C2	1	0805	1nF	311-3585-1-ND
C3	1	0805	100nF	478-KGM21NR71H104KTCT-ND
R1	1	1206	3.3R	541-TNPW12063R30DEEACT-ND
R2	1	0805	10R	541-10.0CCT-ND
R3	1	0805	1K	541-1.00KCCT-ND
R4	1	0805	4.7K	YAG1919CT-ND
R5	1	0805	10K	YAG1763CT-ND

Appendix C – Code

video_ui.py

```
from picamera2 import Picamera2
import tkinter as tk
from tkinter import *

from tkinter import PhotoImage


cam = Picamera2()
cam.start()


def frame():
        cam.capture_file("basic.png")



        im = PhotoImage(file="basic.png")
        imlab.configure(image = im)
        imlab.image = im

        root.after(1, frame)


root = Tk()

root.geometry('1000x800')
root.title("Video UI")
root.configure(bg='#e0c0ff')


tk.Label(root, text="Video Feed", fg="blue", bg="white", font=("Helvetica", 32,




tk.Button(root, text="Close", command=root.destroy, bg='#f0c0ff').place(x = 450,

im = PhotoImage(file = "smile.png")
imlab = tk.Label(root, image = im)
imlab.place(x=175,y=150)

root.after(1, frame)

root.mainloop()
```

Figure 51: SD1 Final Demo UI Code

TPS55288.h STM32 Custom HAL Driver

```
/*
   TPS55288 STM32 HAL Driver
   Written by: Emilio Armas
   Date: 11/22/2025
   Senior Design 2026 Group 9

   How to use:
      -Initialize the I2C bus (typically done by HAL)
      -Define a TPS55288_HandleTypeDef device struct in the application
      -Initialize the TPS55288 using the init function
         (this links the I2C handler struct, sets the I2C device, and intializes the overall device struct)
      -Read values from the device struct in application (DO NOT WRITE TO THESE VALUES)
      -Use some functions provided to set/write commands to the chip
      -Use some functions provided to read status bits from the chip
      -There is an interrupt handler function provided to be filled in by the user for FAULT detection on the INT pin
*/

#ifndef TPS55288_H
#define TPS55288_H

#include "main.h"
#include "stm32g431xx.h"

typedef struct __TPS55288_HandleTypeDef
{
   // I2C handle:
   I2C_HandleTypeDef *hi2c; // STM32 HAL i2c handler struct
   uint8_t i2c_addr; // THIS VALUE IS POST-BITSHIFTED!!!

   // Status field (read-only, use Get_Status or wait for fault interrupt to read these):
   uint8_t err_SCP; // Short-circuit protection error (bool)
   uint8_t err_OCP; // Overcurrent protection error (bool)
   uint8_t err_OVP; // Overvoltage protection error (bool)
   uint8_t power_mode; // 0: Buck, 1: Boost, 2: Buck-boost

   // User control buffer (TO BE READ ONLY, use control functions to set these)
   uint8_t shutdown;     // Device in shutdown mode (1 is shutdown, bool)
   float output_voltage; // Set in volts
   float current_limit;  // Set in amps
   uint8_t enable_SCP;   // Enable short-circuit detection (bool)
```

```
    uint8_t enable_OCP;   // Enable overcurrent detection (bool)
    uint8_t enable_OVP;   // Enable overvoltage detection (bool)
    float slew_rate;      // Output voltage slew rate, given in mV/us
    float OCP_delay;      // Overcurrent protection delay, given in ms
    uint8_t fb_source;    // 0: Internal divider, 1: external resistors (bool)
    float fb_ratio;       // Internal feedback voltage divider ratio
    float CDC_voltage;    // Cable compensation voltage for a 50mV drop across VISP-VISN

    // Mode variables!! (WRITE to these values and then use the TPS55288_SetMode function)
    uint8_t enable;       // Enable power to output (bool)
    uint8_t fsw_doubler;  // 0: No switching frequency changes, 1: fsw doubles in buck-boost mode (bool)
    uint8_t hiccup_mode;  // 0: Hiccup mode disabled during short, 1: Hiccup mode enabled during short (bool)
    uint8_t vout_disch;   // Enable 100mA current sink of vout during shutdown mode (bool)
    uint8_t vcc_source;   // 0: Internal LDO, 1: External 5V rail (bool)
    uint8_t LL_mode;      // 0: PFM on light-load, 1: FPWM on light-load (bool)

} TPS55288_HandleTypeDef;

// IMPLEMENATATION SPECIFIC VALUES:
#define DEV_ADDRESS 0x75 // Configure as 0x74 or 0x75 based on data sheet MODE pin resistor
#define ISP_ISN_SHUNT_RESISTANCE 10 // Output current detection shunt resistance (ohms)
#define TPS55288_POLLING_TIMEOUT 100 // 100ms polling-mode timeout delay

// SYSTEM REGISTERS AND THEIR FIELDS:

// Reference voltage DAC register:
#define VREF_MSB_ADDR   0x00 // Most significant byte address (only two bits)
#define VREF_LSB_ADDR   0x01 // Least significant byte address

// Current limiting register:
#define IOUT_LIMIT_ADDR   0x02 // Register address
#define CURRENT_LIMIT_ON  0x80 // Current limiting enable bit high (enabled by default)
#define CURRENT_LIMIT_OFF 0x00 // Current limiting enable bit low

// Output voltage slew rate register:
#define VOUT_SR_ADDR         0x03 // Register address
#define OCP_DELAY_128us      0x00 // 128us delay for OCP (default)
#define OCP_DELAY_3_072ms    0x10 // 3.072ms delay for OCP
#define OCP_DELAY_6_144ms    0x20 // 6.144ms delay for OCP
#define OCP_DELAY_12_288ms   0x30 // 12.288ms delay for OCP
#define SLEW_RATE_1_25_mV_us 0x00 // 1.25mV/us delay for VOUT slew
#define SLEW_RATE_2_5_mV_us  0x01 // 2.5mV/us delay for VOUT slew (default)
```

```
#define SLEW_RATE_5_mV_us    0x02 // 5mV/us delay for VOUT slew
#define SLEW_RATE_10_mV_us   0x03 // 10mV/us delay for VOUT slew

// Output voltage feedback path selection register:
#define VOUT_FS_ADDR       0x04 // Register address
#define VOUT_FB_INTERNAL   0x00 // Internal voltage feedback selection (true by default)
#define VOUT_FB_EXTERNAL   0x80 // External votlage feedback selection
#define VOUT_INT_FB_0_2256 0x00 // Set feedback ratio to 0.2256
#define VOUT_INT_FB_0_1128 0x01 // Set feedback ratio to 0.1128
#define VOUT_INT_FB_0_0752 0x02 // Set feedback ratio to 0.0752
#define VOUT_INT_FB_0_0564 0x03 // Set feedback ratio to 0.0564 (default)

// Cable voltage drop compensation register:
#define CDC_ADDR         0x05 // Register address
#define SC_MASK_ENABLE   0x80 // Enable short circuit indication (default)
#define SC_MASK_DISABLE  0x00 // Disable short circuit indication
#define OCP_MASK_ENABLE  0x40 // Enable overcurrent indication (default)
#define OCP_MASK_DISABLE 0x00 // Disable overcurrent indication
#define OVP_MASK_ENABLE  0x20 // Enable overvoltage indication (default)
#define OVP_MASK_DISABLE 0x00 // Disable overvoltage indication
#define CDC_OPTION_EXT   0x08 // Set CDC compensation via external resistor
#define CDC_OPTION_INT   0x00 // Set CDC compensation via internal register value (default)
#define CDC_OPTION_0_0V  0x00 // 0.0V VOUT rise with 50mV at (VISP - VISN) (default)
#define CDC_OPTION_0_1V  0x01 // 0.1V VOUT rise with 50mV at (VISP - VISN)
#define CDC_OPTION_0_2V  0x02 // 0.2V VOUT rise with 50mV at (VISP - VISN)
#define CDC_OPTION_0_3V  0x03 // 0.3V VOUT rise with 50mV at (VISP - VISN)
#define CDC_OPTION_0_4V  0x04 // 0.4V VOUT rise with 50mV at (VISP - VISN)
#define CDC_OPTION_0_5V  0x05 // 0.5V VOUT rise with 50mV at (VISP - VISN)
#define CDC_OPTION_0_6V  0x06 // 0.6V VOUT rise with 50mV at (VISP - VISN)
#define CDC_OPTION_0_7V  0x07 // 0.7V VOUT rise with 50mV at (VISP - VISN)

// Device operating mode register:
#define MODE_ADDR      0x06 // Register address
#define VOUT_ENABLE    0x80 // Enable the voltage output
#define VOUT_DISABLE   0x00 // Disable the voltage output (default)
#define FSWDBL_ENABLE  0x40 // Allow for switching frequency to double in buck/boost mode
#define FSWDBL_DISABLE 0x00 // No switching frequency changing during buck/boost mode (default)
#define HICCUP_ENABLE  0x20 // Enable hiccup mode during SC protection (default)
#define HICCUP_DISABLE 0x00 // Disable hiccup mode during SC protection
#define DISCHG_ENABLE  0x10 // Enable the output voltage discharge during shutdown w/ internal 100mA sink
#define DISCHG_DISABLE 0x00 // Disable the output voltage discharge during shutdown (default)
#define VCC_EXTERNAL   0x08 // Set the IC's VCC sourced to the external VCC pint (MUST BE 5V!)
```

```
#define VCC_INTERNAL   0x00 // Set the IC's VCC source to the internal LDO (default)
#define I2C_ADDR_0x75  0x04 // Set the device I2C address to 0x75
#define I2C_ADDR_0x74  0x00 // Set the device I2C address to 0x74 (default)
#define LL_MODE_FPWM   0x02 // Force PWM mode during light-load operation
#define LL_MODE_PFM    0x00 // Set PFM mode during light-load operation (default)
#define MODE_CTRL_INT  0x01 // Set VCC, I2C address, and PFM mode by internal register
#define MODE_CTRL_EXT  0x00 // Set VCC, I2C address, and PFM mode by external resistor (default)

// Device operating status register:
#define STATUS_ADDR     0x07 // Register address
#define SC_DET_BIT      0x80 // Short circuit detection bit (DOES NOT RESET TILL IT'S READ!!!)
#define OCP_DET_BIT     0x40 // Output overcurrent detection bit
#define OVP_DET_BIT     0x20 // Output overvoltage detection bit
#define BOOST_MODE      0x00 // Boost mode value (bits 1:0, to be read)
#define BUCK_MODE       0x01 // Buck mode value (bits 1:0, to be read)
#define BUCK_BOOST_MODE 0x02 // Buck-boost mode value (bits 1:0, to be read)

// FUNCTION PROTOTYPES

// Basic functions (IMPLEMENTATION SPECIFIC!!):
uint8_t TPS55288_Init(TPS55288_HandleTypeDef *hdev, I2C_HandleTypeDef *hi2c, uint8_t dev_addr); // Initialize I2C link and get status
uint8_t TPS55288_WriteReg(TPS55288_HandleTypeDef *hdev, uint8_t reg_addr, uint8_t *value);
uint8_t TPS55288_ReadReg(TPS55288_HandleTypeDef *hdev, uint8_t reg_addr, uint8_t *pdata);
void TPS55288_DeviceShutdown(TPS55288_HandleTypeDef *hdev); // Shutdown the TPS55288
void TPS55288_DeviceWakeUp(TPS55288_HandleTypeDef *hdev); // Wake-up the TPS55288

// High-level functions, all return 0 if successful
uint8_t TPS55288_SetMode(TPS55288_HandleTypeDef *hdev); // Read the mode register of the device and populate the struct accordingly
uint8_t TPS55288_GetStatus(TPS55288_HandleTypeDef *hdev); // Read the status register of the device and populate the struct accordingly
uint8_t TPS55288_SetVoltage(TPS55288_HandleTypeDef *hdev, uint8_t ena, float vout); // Enable/disable the output voltage (bool) and set its value in volts
uint8_t TPS55288_SetCurrent(TPS55288_HandleTypeDef *hdev, uint8_t ena, float ocp_val); // Enable/diable the overcurrent protection and set its value in amps
uint8_t TPS55288_SetSlewRate(TPS55288_HandleTypeDef *hdev, uint8_t slew); // Set the slew-rate of VOUT (use the defines in argument)
uint8_t TPS55288_SetOCPDelay(TPS55288_HandleTypeDef *hdev, uint8_t delay); // Set the time delay for the overcurrent protection (use the defines in argument)

#endif
```

TPS55288.c STM32 Custom HAL Driver (WIP)

```
#include "TPS55288.h"
#include "stm32g4xx_hal.h"
#include "stm32g4xx_hal_def.h"
#include "stm32g4xx_hal_i2c.h"

/*
   -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
   System functions (IMPLEMENTATION SPECIFIC!!):
*/

uint8_t TPS55288_Init(TPS55288_HandleTypeDef *hdev, I2C_HandleTypeDef *hi2c, uint8_t dev_addr)
{ // Initialize I2C link and get status
   hdev->hi2c = hi2c; // Link I2C handler to device struct
   hdev->i2c_addr = dev_addr << 1; // Set I2C device address (HAL expects a shifted address)

   // Wake-up device from shutdown state (device will NOT communicate over I2C if shut-down)
   TPS55288_DeviceWakeUp(hdev);
   HAL_Delay(100);
   // Test if device is ready:
   HAL_StatusTypeDef status = HAL_I2C_IsDeviceReady(hdev->hi2c, hdev->i2c_addr, 3, 500);
   if(status == HAL_OK)
   { // Initialize struct values to datasheet (and user) defaults
      hdev->err_SCP = 0;
      hdev->err_OCP = 0;
      hdev->err_OVP = 0;
      hdev->enable = 0;
      hdev->output_voltage = 5.00f; // default to 5V
      hdev->current_limit = 0.10f; // default to 100mA limit
      hdev->enable_OCP = 1;
      hdev->enable_OVP = 1;
      hdev->enable_SCP = 1;
      hdev->OCP_delay = 0.128f;
      hdev->CDC_voltage = 0.00f;
      hdev->fb_ratio = 0.0564f;
      hdev->fb_source = 0;
      hdev->fsw_doubler = 0;
      hdev->hiccup_mode = 1;
      hdev->vout_disch = 0;
      hdev->vcc_source = 0;
      hdev->LL_mode = 0;
```

```
    }
    return (uint8_t)(status);
}

uint8_t TPS55288_WriteReg(TPS55288_HandleTypeDef *hdev, uint8_t reg_addr, uint8_t *value)
{
     HAL_StatusTypeDef status = HAL_I2C_Mem_Write(hdev->hi2c, hdev->i2c_addr, (uint16_t)reg_addr, 1, value, 1, TPS55288_POLLING_TIMEOUT);
    return (uint8_t)status;
}

uint8_t TPS55288_ReadReg(TPS55288_HandleTypeDef *hdev, uint8_t reg_addr, uint8_t *pdata)
{
     HAL_StatusTypeDef status = HAL_I2C_Mem_Read(hdev->hi2c, hdev->i2c_addr, (uint8_t)reg_addr, 1, pdata, 1, TPS55288_POLLING_TIMEOUT);
    return (uint8_t)status;
}

void TPS55288_DeviceShutdown(TPS55288_HandleTypeDef *hdev)
{ // Shutdown the TPS55288
    GPIOC->ODR &= ~(0x0080);
    hdev->shutdown = 1;
}

void TPS55288_DeviceWakeUp(TPS55288_HandleTypeDef *hdev)
{ // Wake-up the TPS55288
    GPIOC->ODR |= 0x0080;
    hdev->shutdown = 0;
}

/*
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
    High-level functions, all return 0 if successful:
*/

uint8_t TPS55288_GetMode(TPS55288_HandleTypeDef *hdev)
{ // Read the mode register of the device and populate the struct accordingly

}

uint8_t TPS55288_GetStatus(TPS55288_HandleTypeDef *hdev)
{ // Read the status register of the device and populate the struct accordingly
    uint8_t status_data = 0x00;
```

```
    uint8_t xfer_status = TPS55288_ReadReg(hdev, MODE_ADDR, &status_data);
    if(xfer_status == 0)
    {
        if(status_data & SC_DET_BIT)
            hdev->err_SCP = 1;
        else
            hdev->err_SCP = 0;

        if(status_data & OCP_DET_BIT)
            hdev->err_OCP = 1;
        else
            hdev->err_OCP = 0;

        if(status_data & OVP_DET_BIT)
            hdev->err_OVP = 1;
        else
            hdev->err_OVP = 0;

        switch(status_data & 0x03)
        {
            case BOOST_MODE:      hdev->power_mode = 0; break;
            case BUCK_MODE:       hdev->power_mode = 1; break;
            case BUCK_BOOST_MODE: hdev->power_mode = 2; break;
        }
    }
    return xfer_status;
}

uint8_t TPS55288_SetMode(TPS55288_HandleTypeDef *hdev)
{ // Write to the device mode register based on the contents of the struct
    uint8_t tx_byte;
    if(hdev->enable)
        tx_byte |= VOUT_ENABLE;
    else
        tx_byte &= ~(VOUT_ENABLE);

    if(hdev->fsw_doubler)
        tx_byte |= FSWDBL_ENABLE;
    else
        tx_byte &= ~(FSWDBL_ENABLE);

    if(hdev->hiccup_mode)
```

```
        tx_byte |= HICCUP_ENABLE;
    else
        tx_byte &= ~(HICCUP_ENABLE);

    if(hdev->vout_disch)
        tx_byte |= DISCHG_ENABLE;
    else
        tx_byte &= ~(DISCHG_ENABLE);

    if(hdev->vcc_source)
        tx_byte |= VCC_EXTERNAL;
    else
        tx_byte &= ~(VCC_EXTERNAL);

    if(hdev->LL_mode)
        tx_byte |= LL_MODE_FPWM;
    else
        tx_byte &= ~(LL_MODE_FPWM);

    return (uint8_t)TPS55288_WriteReg(hdev, MODE_ADDR, &tx_byte);
}

uint8_t TPS55288_SetVoltage(TPS55288_HandleTypeDef *hdev, uint8_t ena, float vout)
{ // Enable/disable the output voltage (bool) and set its value in volts
    if(ena == 0)
    {
        hdev->enable = 0;
        if(TPS55288_SetMode(hdev) != 0)
            return 1;
    }
    if((vout < 0.81f) || (vout > 21.9f))
        return 1;
    float vref = (hdev->fb_ratio) * vout;
    uint16_t DAC_code = 775.194f * (vref - 0.045);
    uint8_t DAC_MSB = (DAC_code & 0x0300) >> 8; // Select bits [9:8]
    uint8_t DAC_LSB = DAC_code & 0x00FF;
    if(TPS55288_WriteReg(hdev, VREF_MSB_ADDR, &DAC_MSB) != HAL_OK)
        return 1;
    if(TPS55288_WriteReg(hdev, VREF_LSB_ADDR, &DAC_LSB) != HAL_OK)
        return 1;
    if(ena == 1)
    {
```

```
        hdev->enable = 1;
        if(TPS55288_SetMode(hdev) != 0)
            return 1;
    }
    return 0;
}

uint8_t TPS55288_SetSlewRate(TPS55288_HandleTypeDef *hdev, uint8_t slew)
{ // Set the slew-rate of VOUT (use the defines in argument)

}

uint8_t TPS55288_SetOCPDelay(TPS55288_HandleTypeDef *hdev, uint8_t delay)
{ // Set the time delay for the overcurrent protection (use the defines in argument)

}
```