

Automated Lens Parameter Metrology

Lens In, Datasheet Out.

Kiva McCracken¹, Ollie Mueller¹, Zachary Kassner², and Daniel Gomez³

CREOL Group 4 – University of Central Florida, Florida, FL 32826 USA

¹PSE, ²EE, ³CPE

Abstract

The Automated Lens Metrology system uses a fleet of sensors to measure lens properties such as diameter, center thickness, front radius of curvature, back radius of curvature, effective focal length (at different wavelengths), reflective spectra, and transmissive spectra. These properties are used to derive additional parameters and build a datasheet that provides the user with information about the inputted lens.

Advisors

Dr. Aravinda Kar & Dr. Sreeram Sundaresh

Reviewers

Dr. Stephen Eikenberry, Dr. Justin Phelps, & Dr. Nazanin Rahnavard

Contents

1	Chapter 1 Introduction to Automated Lens Metrology (ALM)	14
2	Chapter 2 Project Description	14
2.1	Motivation & Background	14
2.2	Existing Products/Projects	15
2.21	Automatic Lens Analyzer	15
2.22	Optical lens thickness gauge	15
2.23	Micro Lens Process Metrology Inspection Equipment . .	16
2.24	Software Components	16
2.25	Electronic Hardware	17
2.3	Scope: Goals & Objectives	18
2.31	Goals vs. Objectives	18
2.32	Basic Goals	18
2.33	Basic Objectives	18
2.34	Advanced Goals	18
2.35	Advanced Objectives	18
2.36	Stretch Goals	18
2.4	Features/Functionality	19
2.5	Table of Specifications	19
2.6	Hardware Block Diagrams	20
2.7	Optical Block Diagrams	22
2.8	Software Block Diagrams	25
2.9	Prototype Illustrations	26
2.10	Technology Selection	27

3	Chapter 3 Research and Investigation	27
	3.01 Selection of Raspberry Pi 5	27
3.1	Microcontroller Selection	29
	3.11 ATmega	29
	3.12 ESP32	29
	3.13 STM32	30
	3.14 Component Selection	30
3.2	STM32 Research	30
	3.21 Digital Power Supply Decoupling Research	30
	3.22 Analog Power Supply Decoupling Research	30
	3.23 USB Protocol Research	31
	3.24 USB ESD Protection Research	31
	3.25 SWDIO Protocol Research	31
3.3	Stepper Motor Driver Research	32
	3.31 What is a Stepper Motor Driver?	32
	3.32 Component Selection	32
3.4	Power Supply	32
3.5	Photosensor Research	35
	3.51 Photoresistors vs. Photodiodes vs. Phototransistors . . .	37
	3.52 Photodiode Component Selection	38
	3.53 Photodiodes	38
	3.54 Sensor Design Research	38
	3.55 Op-Amp Component Selection	40
3.6	White Light Source	42
	3.61 Source Type	42

3.62	Focusing	43
3.7	Monochromator	47
3.71	Diffraction Grating	47
3.72	Prototype 1	52
3.73	Prototype 2	54
3.74	Optical Hardware	56
3.8	Image Acquisition System	58
3.81	Sensor and Mounting	58
3.82	Linear Gantry	60
3.83	Lens Assembly	61
3.9	Optical Equations	62
4	Chapter 4 Standards and Design Constraints	67
4.1	Communication Protocols	67
4.11	Introduction	67
4.12	I ² C	68
4.13	UART	68
4.14	SPI	69
5	Chapter 5 Application of ChatGPT and Other Similar Platforms	70
5.1	Overview of Compared Platforms	70
5.11	ChatGPT (OpenAI)	70
5.12	Claude (Anthropic)	70
5.13	Gemini (Google DeepMind)	71
5.14	Copilot (Microsoft)	71
5.15	Grok (xAI)	71

5.16	DeepSeek (DeepSeek AI)	71
5.17	Strengths and Weaknesses of Each Platform	72
5.18	Discussion of Comparative Trends	72
5.19	Summary and Outlook	72
5.2	Pros and Cons of AI in Senior Design	72
5.3	ChatGPT Case Studies	72
5.31	Case Study 1	72
5.32	Case Study 2	74
5.33	Case Study 3	75
5.34	Case Study 4	77
6	Chapter 6 Hardware Design	79
6.1	Optics	79
6.11	White Light Source (WLS)	79
6.12	Diffraction Grating	79
6.13	Photodiodes	79
6.14	Monochromator	80
6.15	Image Acquisition System (IAS)	80
6.2	Switching Power Supply PCB System	80
6.3	Microcontroller PCB System	81
6.31	Microcontroller Implementation	81
6.32	3V3 Regulator	81
6.33	Digital Power Supply Decoupling Implementation . . .	84
6.34	Analog Power Supply Decoupling Implementation . . .	84
6.35	USB C USB 2.0 Implementation	85
6.36	USB ESD Protection Implementation	87

6.37	SWDIO Protocol Implementation	88
6.38	Communication Protocol Implementation	89
6.39	External Clock	89
6.310	Reset and Boot	90
6.4	Stepper Motor Driver PCB System	92
6.41	IC Implementation	92
7	Chapter 7 Software Design	95
7.1	Investigation and Testing	95
7.11	STM32 Development Environment	95
7.12	UART Communication	101
7.13	I ² C Communication	101
7.2	Integration	104
7.21	Raspberry Pi System Setup	104
7.22	Software Integration and System Coordination	109
7.23	CSV File Generation and Data Management	113
7.3	Software Block Diagram	115
7.4	Datasheet Generation	116
8	Chapter 8 System Fabrication	116
8.1	Optics	116
8.11	CAD Considerations	116
8.12	White Light Source	119
8.13	Monochromator	120
8.14	Photodetectors	120
8.15	IAS	121

8.2	Electronics	121
8.21	Solder Choice	121
8.22	Soldering Station Details	122
8.23	STM32 Manufacturing	125
8.3	Software	125
8.31	Raspberry Pi Data Access and Output Pathways	125
9	Chapter 9 System Testing and Evaluation	129
9.1	Optics	129
9.11	White Light Source	129
9.12	Monochromator	130
9.2	Electronics	130
10	Chapter 10 Administrative Content	133
10.1	House of Quality	133
10.2	Budget	134
10.3	Bill of Materials	134
10.4	Milestones	134
10.5	Task Distribution	134
11	Chapter 11 Conclusion	134
	Appendix A: ChatGPT Prompts and Outcomes	147
A.1	Case Study 1	147
A.11	Case Study 1 Prompt	147
A.12	Case Study 1 ChatGPT Outcome	147
A.13	Case Study 1 Copilot Outcome	150

A.14	Case Study 1 Claude Outcome	151
A.2	Case Study 2	152
A.21	Case Study 2 Prompt	152
A.22	Case Study 2 ChatGPT Outcome	152
A.23	Case Study 2 Gemini Outcome	156
A.24	Case Study 2 Grok Outcome	158
A.3	Case Study 3	159
A.31	Case Study 3 Prompt	159
A.32	Case Study 3 ChatGPT Outcome	159
A.33	Case Study 3 DeepSeek Outcome	162
A.34	Case Study 3 Gemini Outcome	164
A.4	Case Study 4	166
A.41	Case Study 4 Prompt	166
A.42	Case Study 4 Gemini Outcome	166
A.43	Case Study 4 DeepSeek Outcome	167

References	170
-------------------	------------

List of Tables

I	Numerical Specs.	20
II	Categorical Specs.	21
III	Comparison of Single-Board Computers Considered for System Controller Selection	137
IV	Estimated current draw for one motion control PCB	138
V	Photosensor Comparison	138
VI	Grating period comparison	138

VII	Monochromator Prototype List of Materials	139
VIII	Linear Gantry BOM	140
IX	Comparative Summary of Major AI Platforms	141
X	WLS source choice comparison.	141
XI	Diffraction grating part comparison	142
XII	Photodiode grating part comparison	142
XIII	Raspberry Pi camera comparison.	142
XIV	Bill of Materials	143
XV	STM32 Bill of Materials (BOM)	144
XVI	Bill of Materials for A4988 Stepper Driver Breakout	144
XVII	Project Initialization (SD-I)	145
XVIII	Project Fabrication (SD-II)	145
XIX	Task Distribution	146

List of Figures

1	Optical Schematic	20
2	Stepper Driver Overview	23
3	Stepper Driver Detail	23
4	Optical Block Diagram	24
5	Self-Centering Mount Rendering	24
6	Collimation Diagram	25
7	Concept Rendering	27
8	Power Supply Block Diagram	33
9	LM2596 Board	34
10	LM2596 Desoldered	35

11	A schematic of the photosensor region of the project aligned for collection of (a) the transmissive spectra and (b) the reflective spectra	36
12	A simulation of spherical lenses in a beam of light. (a) shows the beam behavior for the same lens at 0° , 30° , and 45° rotations. (b) shows the beam behavior for the same lens at 50mm, 75mm, and 25mm center diameters. (c) shows the beam behavior for the same lens at $\pm 300\text{mm}$, $\pm 450\text{mm}$, and $\pm 150\text{mm}$ focal lengths	37
13	Photodiode Equivalent Circuit[33]	39
14	Transimpedance Op-Amp[34]	39
15	Transimpedance Op-Amp Design from TI[32]	40
16	TI Op-Amp Comparison[32]	41
17	Chosen Op-Amp[32]	41
18	Spectra of different light sources, [35]	42
19	Parabolic reflector + retro-reflector system simulation.	44
20	CAD model of first reflector, featuring a cross sectional view.	45
21	Reflectivity of some common metals versus wavelength at normal incidence [18]	45
22	Beam compressor/expander optical diagram & equations.	47
23	Second parabolic reflector system test.	48
24	WLS compact case exploded view.	49
25	Photo of in development WLS.	50
26	Diffraction grating diagram (reflective) [29]	51
27	(a) blazed diffraction grating vs (b) holographic diffraction grating [30] . . .	51
28	Thorlabs GR13-1205 spectral efficiency curve [66]	52
29	Monochromator Schematic Ver. 1	53
30	Monochromator Schematic Ver. 2	55
31	Monochromator Prototype Ver. 2	55
32	Resulting colors at (a) $\alpha = -9^\circ$ (b) $\alpha = -5^\circ$ and (c) $\alpha = 2^\circ$	56

33	Exploded-view CAD files for (a) an optical post, (b) an optical post holder, (c) lens mounts, and (d) pencil connectors and rail mounts.	57
34	Normal-view CAD files for (a) the diffraction grating mount, (b) the mount with a transparent cover overlay, and (c) the printed mount in use	57
35	Laser-cut copper slits and circular apertures. From left to right: 0.5mm slit, 1.0mm slit, 1.5mm slit, 10mm circular aperture.	58
36	IAS lens assembly mechanical overview	59
37	Image assembly model, features a sheet of paper sandwiched between laser cut wooden panels. Working image area is 200 x 200mm.	60
38	Linear gantry CAD assembly.	61
39	The 7 Siedel aberrations.	62
40	Lens geometric diagram.	64
41	Experimental verification of focal length extraction method.	65
42	Threshold based radius extraction of generated gaussian profile.	66
43	Optical calculation relationship map.	67
44	I2C Implementation[1]	69
45	UART Implementation[48]	69
46	SPI Implementation[14]	70
47	Switching Regulator Schematic	80
48	STM32 PCB Rendering	82
49	STM32 Overview	83
50	AMS1117-3.3 Implementation	83
51	Digital Power Supply Decoupling Implementation	84
52	Analog Power Supply Decoupling Implementation	84
53	USB-C Implementation	85
54	USB 1.5 k Ω Pull Up Resistor	85
55	JLCPCB Impedance Calculations Used in Design Implementation	86

56	USB Differential Traces Rendering	86
57	ESD Protection Implementation	87
58	SWDIO Schematic (Left) and SWDIO PCB Rendering (Right)	88
59	External Clock Implementation [58]	89
60	X49S (Left) and TX3225 (Right)	90
61	STM32F103C8Tx BOOT TABLE [25]	91
62	Design Recommendation [25]	91
63	Reset Button and Boot Switch Implementation	91
64	a4988 Breakout	92
65	a4988 Main Schematic	92
66	Thermal GND Vias	94
67	Pull-Down Resistors	95
68	LED Indicators	96
69	Pull-Up Resistors	97
70	STM32CubeIDE Pinout and Configuration Menu	98
71	STM32CubeIDE Clock Configuration Menu	99
72	STM32 Cube Programmer Erasing and Programming Menu	100
73	An I2C Frame	102
74	Configuration of I2C in STM32CubeIDE	104
75	Raspberry Pi Configuration Tool Menu	106
76	Software Block Diagram.	115
77	Solder Paste and Solder	121
78	Soldering Tool	123
79	Hotplate Station	124
80	Holding Station and Fume Fan	124

81	STM32 PCB Layout[33]	126
82	STM32 PCB Manufactured and Working[33]	127
83	An example of a terminal connecting to a Pi via ssh	128
84	White light source testing.	129
85	Monochromator testing during demo.	131
86	Diffraction grating testing.	132
87	House of Quality	133
88	STM32 Part Placement	135

1 Chapter 1 Introduction to Automated Lens Metrology (ALM)

The Automated Lens Metrology (ALM) project is a system our group is developing to identify and characterize common optical lenses with minimal user intervention. In practice, the device accepts a lens and generates a comprehensive datasheet describing its key optical properties. The system is organized into several functional blocks: an optical block consisting of a white-light source, monochromator, and image-acquisition module; an electronic hardware block composed of custom control PCBs and stepper-motor assemblies; and a software block implemented on a Raspberry Pi and microcontroller for data processing and system coordination. These components are mechanically integrated along a linear gantry aligned with the optical axis, enabling precise and repeatable positioning during measurements. The mechanical aspects of the system, such as its gantries, linear rails, and motion control will function similarly to those of a 3D printer.

2 Chapter 2 Project Description

2.1 Motivation & Background

Optical laboratories suffer from a chronic component organization issue. Specifically, it is hard to label and keep track of lenses that perform vastly different tasks because they tend to look very similar. As a result, there are tens of thousands of dollars of unsorted optics in our department that are not utilized effectively.

As optical engineers, we have experienced this problem first-hand. A typical sight in our labs is a teaching assistant standing over a set of lenses to compare their focal lengths by eye alone. This is an imprecise and time-consuming sorting method that is not suitable for research. Our design project aims to remedy this issue by autonomously creating a comprehensive datasheet for a given test lens. This design's intended users are students, researchers, and professors in a lab setting.

Current organization solutions on the market focus on the storage portion of inventory. Companies like ThorLabs [65] sell pouches, cassettes, and large storage drawers for lenses. However, these products do not help in identifying an unknown or misplaced lens. The lens metrology systems currently available are generally geared towards production and high-accuracy single data point measurement. Our system serves as a middle ground, providing researchers with data they otherwise might not have about their lenses in a timely and convenient manner. It is designed to work on a macroscopic scale, listing lens characteristics rather than data on quality control.

Typical datasheets in the optics industry take the form of tables detailing the geometric and optical data of a particular lens. While some datasheets are more thorough than others, most include specifications for diameter, focal length, index of refraction, radii of

curvature, lens material, coating, and Abbe number [43] [44] [64]. Many also provide a transmission or reflection graph over the visible spectrum [43] [44].

The design does feature one major compromise. It doesn't measure surface or curvature quality. This is because the easiest way to do so is a white light interferometer, and based on the time constraints for the project, that isn't a realistic goal. Furthermore, most lab optics are bought already having undergone a quality control process. So our device just focuses on identifying and producing a datasheet rather than quality assurance.

2.2 Existing Products/Projects

There are several existing products that automate parts of the lens metrology process. While their scopes vary due to their intended applications, these systems usually function in the UV and visible range and measure foci, dimensions, and transmission spectra. Their physical appearance tends to be some kind of box or stage that the user inputs a lens to. They also vary in size, from a small desktop setup to a large floor-mounted installation. Consistently, the devices we found in our research are only intended for specific-use settings like medicine or optometry. They do not have the scope and affordability needed for more general-use research settings for which we are seeking support. Online catalogs showcasing these products very rarely list their price, but due to their size, scope, and intended use settings, we can estimate their price to be on the range of dozens of thousands of dollars. The goal for our product is to build it with under \$1000 of equipment and for it to be reasonably large to fit the subsystems, but ultimately fit on top of a table. The examples covered in our analysis are:

2.21 Automatic Lens Analyzer

The SOLOS Automatic Lens Analyzer by Tropcon Healthcare is a fully automated lensmeter with a spectrometer measuring UV-A and visible light from 315 nm to 800 nm. It is intended for use in optometry and automatically detects, measures, and marks the centers of single vision, progressive, and multifocal eyeglass lenses. It also measures UV-A, blue light, and visible light transmittance. The focal lengths it can detect are in the range of prescription eyeglasses, from -20mm to +20mm. Its data can be either printed on an internal printer or transmitted wirelessly over LAN or Wi-Fi. This machine is relatively small, fitting easily on the table of a lab or office. For ease of use, it includes a touch screen interface [26]. While this system provides information relevant to our use cases like transmission spectra and focal length, it fails to provide data concerning reflective spectra, index of refraction, and physical dimensions. The SOLOS Automatic Lens Analyzer is sufficient for optometry settings, but for a general optics research setting such a device would benefit from being more thorough.

2.22 Optical lens thickness gauge

The OptiSurf Optical lens thickness gauge by Trioptics uses an interferometer to take non-contact center thickness measurements of single lenses and flat optics. The description

listed on company's website notes the product's ease of use, even for "inexperienced operators" [68]. It includes custom software that automatically aligns the lens in the machine and conducts a statistical analysis of the measured data. This device can even import and read Zemax files and analyze the data like it does physical lenses. It has two levels of analysis: one intensive research & development level, and one shorter production overview level. The gauge is also relatively small and can fit on the table of a lab or office [68]. Our system will likely be closer to the shorter production overview level of analysis, since we are limited in time, resources, and scope by the nature of this course. Since the intended use of our system is to identify pre-existing lenses that belong to catalogs, this broader level of investigation is sufficient for matching and providing general knowledge about lenses.

2.23 Micro Lens Process Metrology Inspection Equipment

The Micro Lens Process Metrology Inspection Equipment by Zygo measures lens data, analyses it, and visualizes it as graphs. Although the exact lens features measured by this machine are unavailable on its catalog page, it is described as supporting "a wide range of parts and metrology parameters" [74]. It is a relatively large machine, and would likely be stored on the floor of a lab. We intend for the system we develop in this paper to be much smaller than this machine, whose size we can only guess is due to the types of data it measures and its accuracy.

2.24 Software Components

For the software portion of this project, we will be utilizing available open-source libraries as well as existing micro-controllers such as the Raspberry Pi and STM32. These libraries differ in content and scope depending on the technologies they relate to and the purposes those technologies serve.

- 1) **Raspberry Pi** The existing libraries that will be used on the Raspberry Pi will be split among four tasks:
 - a) **Hardware Communication**
For hardware communication between the Raspberry Pi and the STM32, *pyserial* [38] is an existing library that enables UART communication between the two. This library enables Python to interact with devices that communicate via UART.
 - b) **Data Handling**
CSV [21] is a standard Python library used for making CSV formatted files from data. This will be crucial in transferring raw test data into CSV format. *Pandas* [31] provides higher-level data handling, enabling structured row/column storage for CSV files. *Openpyxl* [24] enables the generation of Excel files in Python, extending compatibility for XLSX files if required.
 - c) **Image Processing**
Opencv-python [61] captures and stores images from the imaging system at defined time intervals. *Picamera2* [22] enables the camera module in a Raspberry Pi by providing camera control, exposure settings, and frame capturing.

Pillow [5] provides additional image processing capabilities such as saving, converting, and manipulating images in Python.

- d) **System Utilities** *Asyncio* [19] supports concurrent execution of serial communication, data logging, and image acquisition tasks. *logging* [20] assists with structured runtime logging for debugging and test traceability.

2) **STM32**

The STM32 micro-controller will be handling real-time control of hardware sub-systems such as the tray motion, lens clamping, lens flipping, etc. The STM32 will be using some already existent Arduino Core libraries. *AccelStepper* [40] is useful to use along side high-level stepper motors, which will be used in actions such as the tray movement and caliper positioning, *Servo.h* [50] provides simple control over servo motors for some of the motions used in this procedure.

2.25 Electronic Hardware

- 1) **STM32 Blue Pill** The STM32 Blue Pill is a compact and cost-effective development board based on the STM32F103C8T6 ARM Cortex-M3 microcontroller. In this project, it will serve as a practical platform for prototyping and validating software, ensuring that control logic and firmware can be developed and tested early before being transferred to the finalized custom STM32 board.
- 2) **A4988 Pololu** The A4988 Pololu development board is a widely used stepper motor driver that offers microstepping capabilities and reliable current limiting. Its breakout board format simplifies prototyping and software validation, allowing control logic and firmware to be developed and tested in advance before transitioning to the finalized custom A4988 board.
- 3) **LM2596 DC-DC Converter** The LM2596 is a widely used DC-DC buck converter module that provides efficient step-down voltage regulation, making it ideal for powering lower-voltage circuits from higher-voltage sources during prototyping and development. At the same time, its larger SMD package makes handling and manufacturing simpler compared to smaller regulators. Many inexpensive PCBs and breakout modules built around the LM2596 are widely available through vendors such as Amazon and can be used as reference designs to accelerate development. Studying these readily obtainable boards provides practical insight into layout, component choices, and thermal considerations. The boards are easily desolderable making them a ready source of parts.
- 4) **Closed Loop Stepper Motors** Closed-loop stepper motors integrate an internal encoder to provide real-time position feedback, allowing the driver to detect and correct any missed steps. This feedback system merges the precision of stepper control with servo-like reliability, ensuring accurate motion even under varying loads. Such functionality is especially valuable for our metrology implementation, where both precise angular control and accurate measurement are critical.

2.3 Scope: Goals & Objectives

2.31 Goals vs. Objectives

In this project, we define goals as more of abstract requirements for our design. These are qualitative benchmarks rather than quantitative ones that help us rate the performance of the final product. We define objectives as a counterpart to goals; concrete and quantitative benchmarks that can be measured and compared directly. Our objectives are what we use to determine if we meet our goals.

Our top priority for ALM is to achieve our basic goals by achieving our basic objectives. Once these requirements are met, we can reasonably move on to our advanced goals and objectives. If time and resources permit, we will work towards completing our stretch goals.

2.32 Basic Goals

- The system will measure the focal length of the test lens.
- The system will measure the transmissive and reflective spectra of the test lens.
- The system will manually measure the test lens' geometric features.

2.33 Basic Objectives

- The white light source will be built.
- The white light source will be collimated.
- The wavelength selection will be done by a single monochromator.
- The image acquisition system will record an image.

2.34 Advanced Goals

- The data produced by the system will be output in a plain text format.
- The data produced by the system will be recoverable with a flash drive.

2.35 Advanced Objectives

- The wavelength selection will be done by a double monochromator.
- The system will derive optical values from the measured geometry.

2.36 Stretch Goals

- The system will have a graphical touch interface.
- The system will generate a datasheet as a PDF.
- The datasheet output by the system will be accessible by QR code or printed document.
- The code will estimate the test lens' product number by comparing the specs to a known product database.

2.4 Features/Functionality

For the advanced scope the following inputs will be recorded or calculated:

- Geometric Properties
 - Diameter
 - Center Thickness
 - Front Radius of Curvature
 - Back Radius of Curvature
- Optical Properties
 - Effective Focal Length
 - Normalized Transmission
 - Normalized Reflection
- Controlled Variables
 - Light Wavelength
 - Lens Presence

The above will be measured using a mechanical measuring system, monochromator assembly, plane imaging system, and photodiodes.

From these values & related calculations the following will be derived:

- Geometric Properties
 - Edge Thickness
 - Lens Shape
- Optical Properties
 - Index of Refraction
 - Front Focal Length
 - Back Focal Length
 - Power
 - Spectra
 - Substrate
 - Abbe Number
 - Coatings

The system will then generate either a plain text document or datasheet from this derived data and output it to the user.

2.5 Table of Specifications

Tables I & II show the required minimum scope expectations for our device. In the future, the device may exceed the tables' current scope and accuracy.

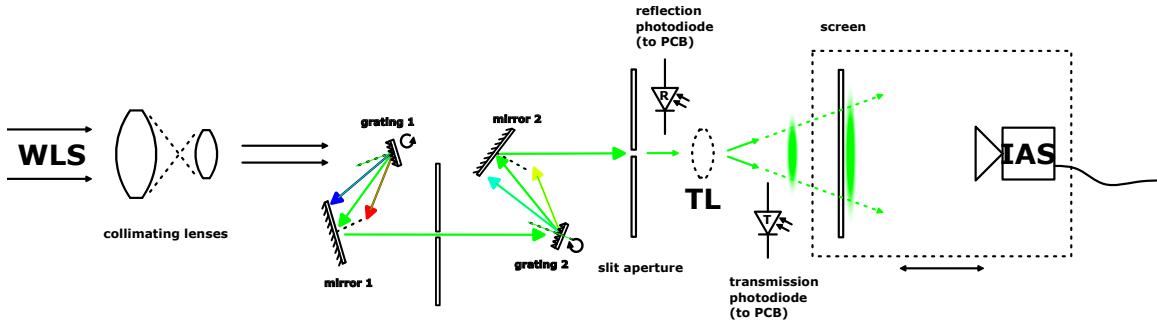


Fig. 1: Optical Schematic

Spec	Minimum	Maximum	Accuracy
Diameter (mm)	20	30	± 2
Center Thickness (mm)	5	20	± 4
Front Radius of Curvature (mm)	± 10	$\pm \infty$	± 2
Back Radius of Curvature (mm)	± 10	$\pm \infty$	± 2
Eval Wavelength (nm)	400	700	± 5
Effective Focal Length (mm)	± 10	$\pm \infty$	± 10
Index of Refraction	1.5	2	$\pm .1$
Front Focal Length (mm)	± 15	$\pm \infty$	± 10
Back Focal Length (mm)	± 15	$\pm \infty$	± 10
Power (1/m)	.1	3	$\pm .3$
Abbe Number	30	60	± 8
Coatings (nm)	400	700	± 10

TABLE I: Numerical Specs.

2.6 Hardware Block Diagrams

The Automated Lens Parameter Metrology (ALPM) project requires the precise actuation of mechanical components. This is a task that is best achieved through the use of stepper and servo motors. These motors have been chosen due to their ability to provide fine control over angular displacement and torque, which is critical in optical measurement systems where accuracy and repeatability directly influence experimental validity. For this task, a dedicated Printed Circuit Board (PCB) will be designed and fabricated. A custom PCB not only consolidates the necessary electronics into a robust form, but also provides a level of noise immunity and ease of integration with the rest of the ALPM system. At

Spec	Allowed	Prohibited
Lens Types	Plano Concave, Plano Convex, Double Concave, Double Convex, Negative Meniscus, Positive Meniscus	Aspheric, Achromatic, Compound,
Substrates	Glass, Plastic	Metal, Ceramic, PTFE, etc...

TABLE II: Categorical Specs.

a high level, the PCB can be broken down into four elements, shown in Figures 2 & 3:

- 1) Main Control Unit (MCU)
- 2) Stepper Motor Driver PCB
- 3) 3.3V Digital Voltage Supply
- 4) 12V-24V Motor Voltage Supply

The MCU serves as the central element that generates the command signals required to operate the motors. Its primary role is to issue step and direction pulses to the motor driver. For this project the STM32F103C8Tx series will be used for its high processing power and low cost (<\$2) when manufactured through JLC PCB, a China-based PCB manufacturer. The MCU will manage higher-level control of the driver through pins specific to the a4988 stepper motor driver breakout PCB such as sleep, enable, MS1, MS2, and MS3. Sleep controls whether the stepper driver creates a continuous resistive torque or not. This is useful for limiting power output, since it later leads to heat output. Enable controls whether the stepper driver is on or off, a feature typically used in systems that implement polling for power control. It likely will not find use in our design. However, the PCB design will still feature traces from the enable pin to the MCU in case the function becomes useful later on. The pins MS1, MS2, and MS3 control the micro-stepping capabilities. The specifics of which will be detailed in the stepper motor driver section.

The A4988 stepper motor driver serves as the hardware interface that converts the lowpower logic signals issued by the MCU into the high-current control necessary to operate the motor coils. It achieves this through its integrated dual H-bridge circuitry, which allows it to rapidly switch current between windings in the precise sequence required for stepper motion. The driver also provides current limiting, which is configured through an external reference voltage and sense resistors, ensuring that the motor receives adequate torque without exceeding safe thermal or electrical limits. Additionally, the A4988 supports multiple micro-stepping modes that are selectable through the MS1, MS2, and MS3 pins. These modes enable a stepper normally capable of 200 step revolution to reach up to 3200 step revolution. This does come at the cost of increased power consumption, so the MCU should be capable of toggling these pins. With the a4988

maintaining smooth motor operation, the MCU will be able to manage higher-level control logic with little issue.

The stepper motor supply will be provided through a 2.5 mm ID / 5.5 mm OD barrel jack input. This form factor is capable of 12-24 V and 2.5-5 A supply. These levels are sufficient to drive the A4988 stepper driver and energize the motor coils. The MCU and driver logic will be powered by a regulated, low-voltage rail. By isolating the motor supply in this way, the system can benefit from the flexibility of the motor voltage rail for high-power delivery without compromising the stability of the control electronics.

Both the A4988 and the STM32F103C8Tx require a clean and stable 3.3 V supply to operate reliably. Because the 2.5 mm ID / 5.5 mm OD barrel jack input will deliver a voltage between 12-24 V depending on system configuration, dedicated regulation circuitry will be necessary to step the input down to logic-safe levels. To accomplish this, two stages of regulation will be employed. A switching voltage regulator, likely the LM2596, will be used to produce a relatively stable intermittent rail from the 2.5 mm ID / 5.5 mm OD barrel jack input. The LM2596 was chosen for its ease of implementation relative to other switching regulators. From this intermittent rail, the AMS1117-3.3 low dropout regulator will be used to generate the final 3.3 V supply for the MCU and stepper driver logic. The AMS1117 is capable of excellent line regulation at a steady voltage level, even under fluctuating load conditions. 22 μ F decoupling capacitors will be placed at both regulator stages on the inputs and outputs to minimize ripple and suppress transient noise. This dual-regulator design guarantees that the control electronics will remain stable, efficient, and isolated from the motor supply, protecting the sensitive components.

The ALPM process requires the accurate collection of lens dimensions, which in turn necessitates that each lens be securely lifted and stabilized during measurement. The mechanical design, shown in Figure 5, accomplishes this using three arms that will extend outward to clasp around the lens. It was designed by the team and will be 3-D printed to ensure its dimensions and tolerances meet the needs of the project. The user will position the lens at the center of the mechanism and adjust a potentiometer embedded in a voltage divider to open or close the arms. The resulting voltage output will be proportional to the arm displacement. It will then be read by the MCU through an Analog to Digital Converter (ADC). From this measurement, the system will calculate the lens diameter while simultaneously maintaining a firm grip on the lens for further analysis.

2.7 Optical Block Diagrams

Figures 1 & 4 depict the different optical systems/components:

White Light Source: A high power Xenon gas discharge tube source adapted from a car headlight. The light source itself is a previous project by Kiva McCracken. The system will be adapted & integrated into the optical assembly.

Source Collimating: A Xenon bulb that is effectively a point source. In order for the source to be optically worthwhile, it needs to be collimated so that all the rays are parallel.

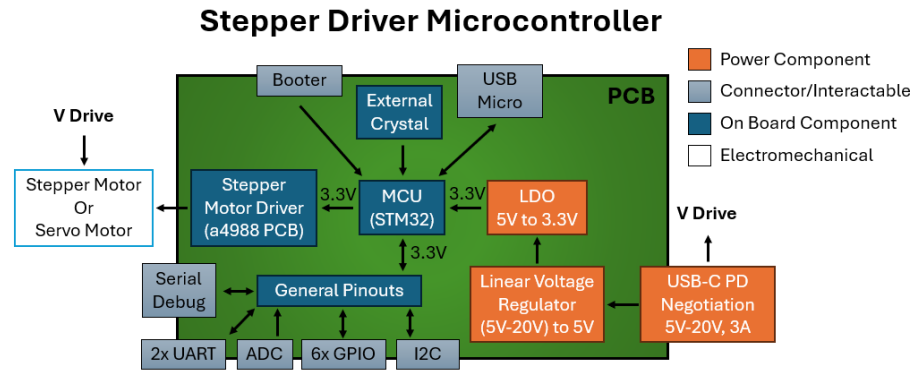


Fig. 2: Stepper Driver Overview

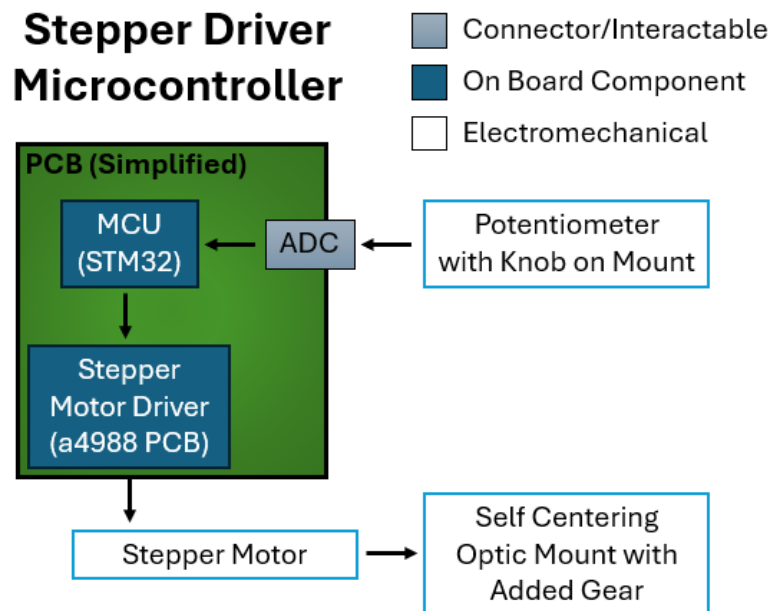


Fig. 3: Stepper Driver Detail

The least expensive & simplest way to achieve a high-quality beam is with a parabolic reflector, a converging lens, an aperture, and a collimating lens. A diagram can be found in fig. 6.

Beam Compressor: The beam will be fed into two lenses, reducing the beam to a 10mm diameter.

Diffraction Grating: A diffraction grating will split light into its spectrum through interference, creating a rainbow pattern.

Angled Mirrors: Two mirrors are mechanically coupled so that when they are rotated,

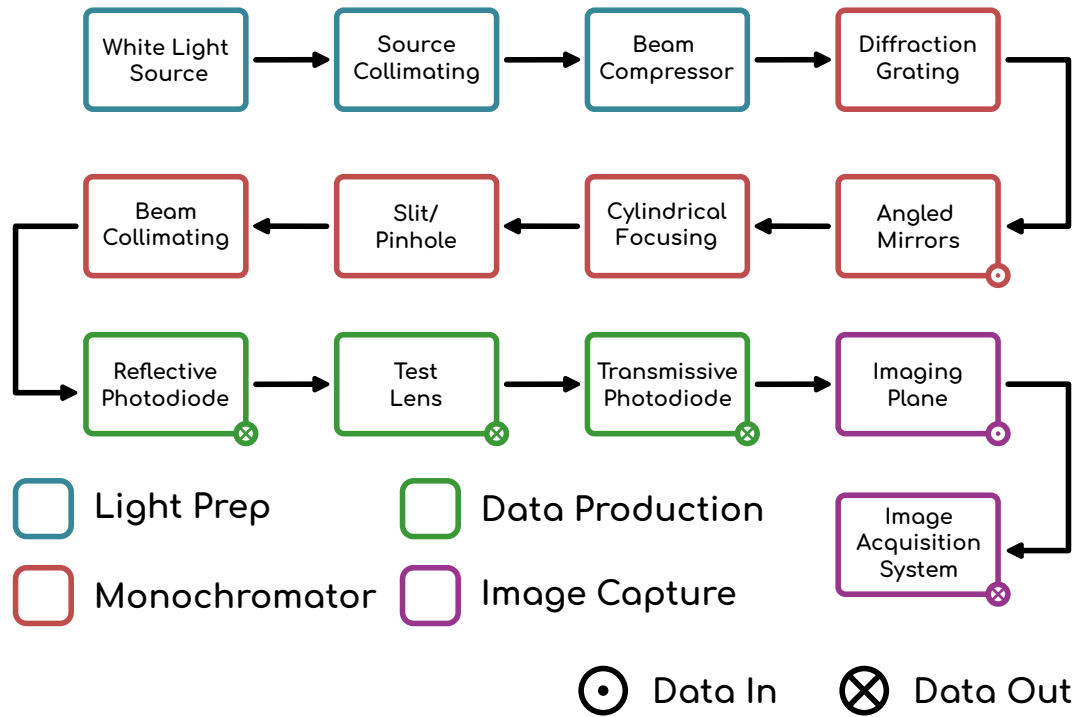


Fig. 4: Optical Block Diagram

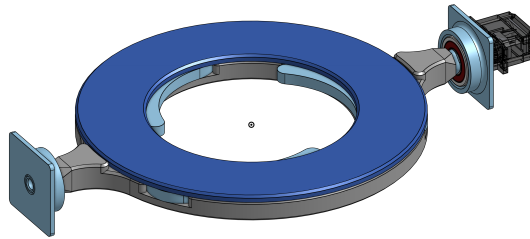


Fig. 5: Self-Centering Mount Rendering

the maxima of a given wavelength are directed towards the slit/pinhole.

Cylindrical Focusing: A cylindrical lens is used to compress the vertical spread produced by the diffraction grating and focus as much light as possible into the slit/pinhole.

Slit/Pinhole: The slit/pinhole blocks the unwanted wavelengths and acts as a monochromatic point source.

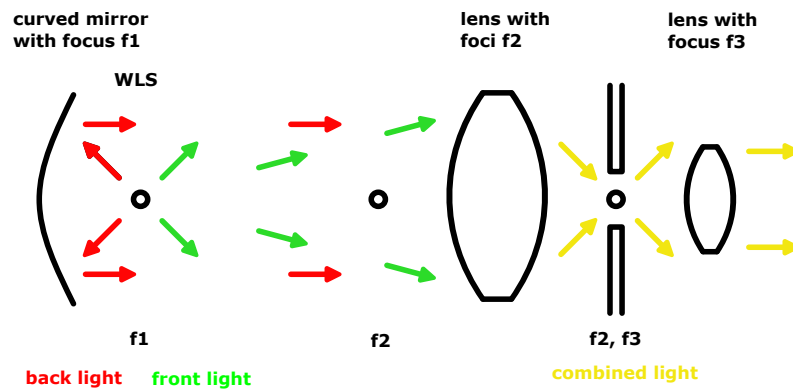


Fig. 6: Collimation Diagram

Beam Collimating: A lens placed one focal length away from the slit/pinhole collimates the light into a tightly collimated monochromatic beam.

Reflective Photodiode: A PCB consisting of multiple photodiodes (with different wavelength ranges) will allow broadband evaluation of intensity. The sensor will be positioned off-axis with regard to the lens so that any light reflected by the test lens can be evaluated.

Test Lens: This is the test optic. The supported lens types can be found in Table II. It will be held by a self-centering holder which will also measure the diameter of the optic.

Transmissive Photodiode: This is similar to the reflective photodiode, but positioned after the test lens. It measures the intensity of transmitted light through the lens.

Imaging Plane: The imaging plane is effectively a projector screen that can be viewed from the opposite side. It sets up the macro-scale evaluation of the lens spot size by using a relatively long focal length and simple image acquisition assembly. The imaging plane will jog toward or away from the test lens to evaluate the spot size over distance.

Image Acquisition System Also known as IAS. The system consists of a series of imaging lenses and a CMOS/CCD sensor to the main board. The IAS is fixed in relation to the imaging plane.

2.8 Software Block Diagrams

The software flow for the Automated Lens Parameter Metrology system has been designed to provide clear sequencing of all measurement and data-handling tasks, as illustrated in Fig. 76. The process will begin with an initialization step in which the Raspberry Pi 3 will orchestrate system startup and the STM32 will send the first command to extend the tray. Once the lens is placed and confirmed, the tray will retract and secure the lens with the self-centering mount. Displacement data from the mount will then be read by

the STM32 subsystem and transmitted over UART to the Raspberry Pi, where the lens diameter and radius will be calculated.

Following lens placement, the caliper system will measure the center thickness and then the radius of curvature on the first surface of the lens. The self-centering lens mount will subsequently be rotated 180 degrees to allow for measurement of the second surface curvature. These results will be transmitted to the Raspberry Pi for aggregation.

The next phase will involve optical characterization using a light source and imaging system. The imaging plane and camera will be positioned at the starting point of the test track. At each position, a servo mechanism will adjust the light source to transmit different wavelengths sequentially through the lens. For each wavelength, the system will record the spot measurement, photo diode readings, and capture an image using the image acquisition system (IAS). After data capture, the imaging system will advance 10 mm along the track and repeat the loop until the full lens area has been tested.

Captured images will be processed in Python on the Raspberry Pi to extract spot size and other relevant optical properties. All measurement results, including lens geometry, curvature data, light transmission characteristics, and image-derived features, will then be consolidated and formatted into a CSV file. This file will represent the complete dataset for the test run. In the current implementation, the CSV will be stored locally on the Raspberry Pi for further analysis or manual transfer.

The software architecture is divided into functional domains: kinematics/servo control, sensor acquisition, data handling, and communication. The STM32 microcontrollers will manage real-time kinematics and sensor data collection, while the Raspberry Pi will aggregate the data, process the image, and store the output. Communication between devices will be done over UART, which will ensure reliable data exchange.

This structured workflow provides a clear sequence of operations and a modular software design. By separating low-level hardware control from high-level data handling, the system will achieve both precise measurement control and robust data management, creating the foundation for future expansions such as automated networking and integration with external applications.

2.9 Prototype Illustrations

The illustration visible in Fig. 7 shows a concept of how the final design may look. It features an aluminum extrusion enclosure to keep out light, the lens holder, the beam & screen, and the IAS. In the back of the enclosure lies the WLS & monochromator.

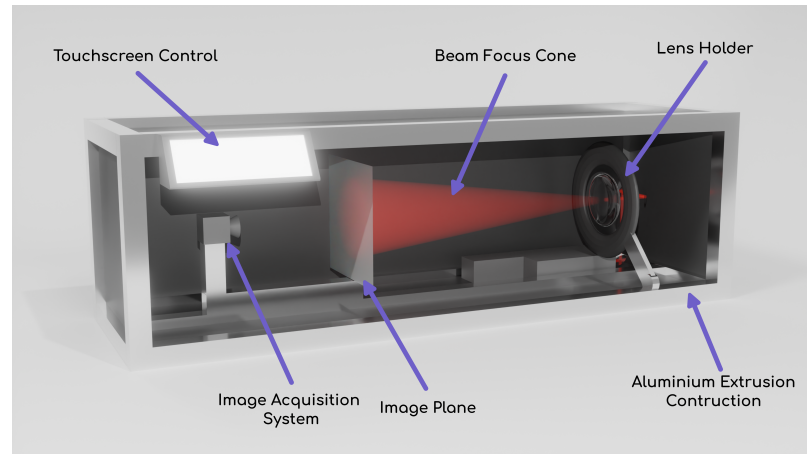


Fig. 7: Concept Rendering

2.10 Technology Selection

3 Chapter 3 Research and Investigation

3.01 Selection of Raspberry Pi 5

The system controller is a critical component of the Automated Lens Parameter Metrology system, responsible for coordinating communication between STM32 microcontrollers, handling image acquisition, processing measurement data, and generating the final CSV data files. Several single-board computers (SBCs) were evaluated for this role, including the Raspberry Pi 3, Raspberry Pi 4, and NVIDIA Jetson Nano. After comparing hardware performance, peripheral support, and system compatibility, the **Raspberry Pi 5** was selected as the most suitable platform.

a) Processing Power and Performance

The Raspberry Pi 5 features a **2.4 GHz quad-core ARM Cortex-A76 processor**, representing a significant performance improvement over the Pi 4's 1.5 GHz Cortex-A72 and the Pi 3's 1.2 GHz Cortex-A53. This enhanced CPU architecture provides substantially higher single-thread performance, enabling faster execution of Python scripts responsible for UART communication, data parsing, and image processing tasks.

In comparison, the Jetson Nano's 1.43 GHz Cortex-A57 CPU performs well for embedded AI workloads but demonstrates slower general-purpose performance and higher latency for single-threaded operations. The Pi 5's processor therefore provides a better balance between computational speed and energy efficiency for this project's mixed I/O and data-processing demands.

b) Memory and I/O Bandwidth

The Raspberry Pi 5 supports **LPDDR4X memory up to 8 GB**, with substantially higher throughput and lower latency than the LPDDR4 (Pi 4) and LPDDR2 (Pi 3) used in earlier models. The increased memory bandwidth improves image buffering and real-time data handling during optical scans.

In addition, the Pi 5 introduces a new **PCIe 2.0×1 interface**, dual full-speed **MIPI CSI/DSI connectors**, and an updated **USB 3.0 controller with dedicated lanes**—making it ideal for high-speed camera interfacing and rapid data transfers from sensors. These hardware interfaces ensure smooth communication with the imaging system and enable future expandability for higher-resolution cameras or faster photodiode data acquisition.

c) Power Efficiency and Thermal Management

Despite its performance increase, the Pi 5 maintains an efficient thermal design, consuming less power per unit of computation than the Jetson Nano, which typically requires an active cooling solution and a dedicated 5 V 4 A power supply. The Pi 5 can operate on a standard 5 V 3 A USB-C supply, simplifying integration within the machine's enclosure. Its onboard fan connector and dynamic frequency scaling allow efficient cooling management without external control circuitry.

d) Software Ecosystem and Compatibility

The Raspberry Pi 5 maintains full compatibility with the existing Raspberry Pi OS and Python ecosystem. This continuity allows direct use of the project's chosen libraries—*pyserial*, *pandas*, *openpyxl*, *opencv-python*, and *picamera2*—without additional configuration.

While the Jetson Nano offers strong GPU acceleration under NVIDIA's JetPack SDK, its Linux environment requires specialized setup and is optimized for AI inference rather than general data orchestration. The Pi 5's streamlined operating system and documentation reduce development time and complexity, making it more practical for a multidisciplinary student team.

e) Cost and Availability

From a budget standpoint, the Raspberry Pi 5 provides the best price-to-performance ratio among the evaluated options. It is significantly more powerful than the Pi 4 and Pi 3 while remaining more cost-effective and accessible than the Jetson Nano, which has higher component costs and power requirements. The widespread community support and educational resources surrounding the Pi 5 further enhance maintainability and troubleshooting.

f) Summary

Overall, the **Raspberry Pi 5** offers a combination of modern architecture, high-speed interfaces, energy efficiency, and robust software support that make it the optimal con-

troller for this system. It provides sufficient computational power for image analysis, real-time coordination with STM32 subsystems, and local data storage, all while maintaining affordability and ease of integration. These advantages ensure reliable performance throughout the system's development and testing phases and position the platform for future scalability and potential AI-based analysis extensions.

3.1 Microcontroller Selection

The selection of the microcontroller Unit (MCU) played a crucial role in the project, with several factors carefully evaluated before finalizing a decision. The primary MCU families considered were the ATmega, ESP32, and STM32 lines. Key decision criteria included cost, documentation quality, ease of programmability, logic voltage level, and the availability of existing development boards to allow our computer engineer to begin prototyping before an integrated PCB is developed.

3.11 ATmega

The ATmega line is known for its simplicity, affordability, and widespread support, making it an attractive option for low-cost or educational projects. In terms of cost, ATmega microcontrollers such as the ATmega328P are inexpensive and easy to source, though their limited performance can become a bottleneck in complex applications. Their documentation is excellent due to the extensive Arduino ecosystem, and programmability through the Arduino IDE is intuitive and beginner friendly although there are significant limitations when compared to STM32 IDE in terms of debug capabilities. While it is possible to run the ATmega chips at any voltage from 1.8-5.5V the clock faces issues when running it at voltages other than 5V. There is not much documentation regarding solving this issue. This 5V logic allows ATmega devices to interface easily with older hardware that also operates at 5V logic, but this also makes the chip much less power-efficient than 3.3 V systems. In terms of availability, ATmega-based development boards are common, but when designing custom PCBs through vendors like JLCPCB, the cost can increase slightly due to limited modern package options and fewer assembly stock parts compared to newer MCU lines. Ultimately, this line was decided against due to the 5V operating voltage and low debug availability of debugging options within its IDE.

3.12 ESP32

The ESP32 family, developed by Espressif, delivers high performance and integrated Wi-Fi and Bluetooth at a very competitive cost, making it one of the best performance-per-dollar options for IoT devices but the nature of our project would not make much use of its documentation and open-source resources are strong, and programmability is flexible with support for both the Arduino IDE and the native ESP-IDF environment. Many of the chips are also multicore, allowing multithreading applications. Running at 3.3 V logic, ESP32s are ideal for modern sensor interfacing, especially in IoT applications but the chip is incredibly power hungry amongst its 3.3V logic counterparts. The ESP32 series tends to be pricier compared to the other reviewed series.

3.13 STM32

The STM32 series from STMicroelectronics provides a balance between performance, cost, and scalability albeit at a steep learning curve. While cost varies by model, the STM32F103 line is affordable and readily available through JLCPCB's component library, which helps reduce assembly cost compared to sourcing uncommon parts. The documentation is comprehensive and industry-grade, complemented by tools such as STM32CubeIDE for reliable programmability and debugging. It cannot be stated enough the support STM32CubeIDE provides. The ability to generate verified chip pinouts for necessary features, resolve clock issues through software, and generate code with pinouts as labeled variables makes up for the steep learning curve. Operating at 3.3 V or 5V logic, STM32 devices are efficient and fully compatible with modern and older electronics standards. In terms of availability, STM32 "Blue Pill" and Nucleo boards are widely distributed, and their common package footprints (like LQFP-48) make them easy and economical to integrate in custom boards through manufacturers like JLCPCB.

3.14 Component Selection

Ultimately, the STM32 was chosen because STM32CubeIDE provided a level of support that outweighed the steep learning curve, offering features such as automatic pinout generation, clock configuration, labeled variable code output, and advanced debugging tools that made both firmware development and custom PCB design significantly more efficient.

3.2 STM32 Research

3.21 Digital Power Supply Decoupling Research

Voltage decoupling is the practice of filtering out unwanted AC variations or noise from a DC power rail so that only the stable DC supply reaches the device. This is achieved by placing capacitors close to the power pins of the integrated circuit, which act as local energy reservoirs: at high frequencies, the capacitors provide a low-impedance path to ground, effectively shunting transient noise away from the microcontroller. For an STM32 power supply, decoupling is essential because the processor's internal switching activity can introduce rapid current spikes that, if not filtered, may cause voltage dips, logic errors, or instability. By carefully selecting and positioning decoupling capacitors, the STM32 can operate from a clean and stable supply, ensuring reliable performance and reducing susceptibility to external noise.

3.22 Analog Power Supply Decoupling Research

A ferrite bead in combination with decoupling capacitors forms a low-pass filter for the analog power rail. The bead can be modeled as a frequency-dependent impedance $Z_{FB}(f)$, which at low frequencies is barely a resistor (on the order of a few hundred m Ω)

but rises sharply with frequency due to its inductive nature. Together with a capacitor C , the cutoff frequency of the LC filter can be approximated as

$$f_c = \frac{1}{2\pi\sqrt{L \cdot C}} [16] \quad (1)$$

where L is the effective inductance of the ferrite bead and C is the local decoupling capacitance (e.g., $C_6 = 10 \text{ nF}$, $C_7 = 1 \mu\text{F}$, $C_8 = 1 \mu\text{F}$). This configuration attenuates high-frequency digital noise on the 3.3 V rail before it reaches the sensitive analog domain of the STM32. Since our aim is to cutoff all AC signals, the formula is more conceptual in nature.

3.23 USB Protocol Research

The Universal Serial Bus (USB) has evolved through successive generations to increase both data transfer capability and power delivery while maintaining backward compatibility. USB 1.0 introduced “Low-Speed” (1.5 Mb/s) and “Full-Speed” (12 Mb/s) operation, primarily intended for human interface devices, with a standardized 5 V supply and power allocation defined in unit loads of 100 mA. According to Bearfield and Texas Instruments, system-level requirements such as voltage drop limits, minimum bypass capacitance at downstream ports, and over-current protection were established early in the specification to ensure reliable hot-plug operation and robust power distribution [2]. USB 2.0, released in 2000, increased the signaling rate to 480 Mb/s (“Hi-Speed”), enabling widespread use in external storage and multimedia applications. USB 3.0, introduced in 2008, further advanced the standard with “SuperSpeed” signaling at 5 Gb/s using additional high-speed differential pairs, while also raising the maximum port current to 900 mA, thereby supporting more power-intensive devices. Together, these revisions demonstrate a steady progression in bandwidth and power management, extending USB’s role as the dominant peripheral interface.

3.24 USB ESD Protection Research

A likely source of Electrostatic Discharge (ESD) events in this design is the USB-C port, which is frequently connected and disconnected during programming and testing. The differential D+ and D- data lines are particularly sensitive and can fail to transmit data properly after a single ESD strike. The VBUS line, which carries the primary 5 V supply, also requires protection due to its direct exposure to external devices. While it is generally recommended to protect the CC1 and CC2 configuration channel pins as well, the existing $5.1 \text{ k}\Omega$ 0805 pull-down resistors already provide an effective discharge path to ground, making additional protection on these lines likely unnecessary.[32]

3.25 SWDIO Protocol Research

An important consideration when designing our microcontroller was a robust method for programming the MCU. It was decided that USB alone would not be sufficient, since the

receptacle commonly detaches after repeated use. For this reason we investigated SWDIO (Serial Wire Data I/O) functionalities. The Serial Wire Debug (SWD) interface uses two dedicated signals, SWDIO and SWCLK, to provide low pin count debug access to ARM Cortex-M microcontrollers. SWDIO is a bidirectional line that carries commands, data, and status information from the debug host to the target, while also transmitting responses from the target back to the host. The protocol is clock synchronous, with data sampled relative to transitions on SWCLK and changes on SWDIO timed accordingly. For reliable operation, the target board typically includes a defined pull up resistor on the SWDIO line so the interface remains in a known idle state when the probe is not connected. This compact two wire scheme still provides full access to the ARM Debug Access Port (DAP), which allows breakpoint control, memory access, and register visibility without the complexity of a full pin JTAG interface.

3.3 Stepper Motor Driver Research

3.31 What is a Stepper Motor Driver?

A stepper motor driver is an electronic circuit that provides the correct sequence of voltage and current pulses needed to control a stepper motor. Since stepper motors require precisely timed phase energization to move in discrete steps, the driver handles the switching of the motor windings in the proper order. It also regulates current to prevent overheating and allows features such as microstepping for smoother motion and finer resolution. In practice, the driver serves as the interface between a low-power control signal from a microcontroller (such as an STM32) and the higher current demands of the stepper motor, ensuring accurate, efficient, and reliable motion control. [27]

3.32 Component Selection

The use of integrated circuits (IC) for this task is crucial, thus the correct IC must be chosen. The A4988 stepper motor driver is a strong choice because it is widely supported, cost-effective, and easy to integrate into microcontroller projects. It provides features such as adjustable current limiting, over-temperature protection, and up to 1/16 microstepping, which allows for smoother and more precise motor control. A major advantage is the availability of ready-made breakout boards, which provides a template to replicate and allows immediate prototyping. In addition, the A4988 has already been used successfully in previous personal projects, adding familiarity and confidence in its performance and reliability. [42]

3.4 Power Supply

a) Power Budget Estimates

Table IV summarizes the estimated current draw for one motion control PCB, and the values are based directly on the functional blocks shown in the system diagram.

The dominant load is the NEMA 17 stepper motor driven through the A4988, which typically draws around 1 A during normal operation and up to 1.6 A in a worst case stall or high torque condition, making this the primary driver of the 12 V rail sizing. The STM32 microcontroller has a very small current requirement, with a typical draw of approximately 50 mA and a conservative worst case assumption of 150 mA to account for higher clock speeds and peripheral activity. The logic supply for the A4988 is similarly light, requiring only tens of milliamps for its digital interface and internal control circuitry. Finally, the LED indicators and small peripheral devices contribute a minor additional load, estimated at roughly 30 mA in normal use and up to 100 mA when all indicators are active. Summed together, these assumptions produce a realistic total current requirement of about 1.2 A typical and up to 2.0 A worst case from the 12 V rail, which aligns with the expected behavior of the blocks shown in the power distribution diagram and ensures sufficient margin for reliable operation.

b) Power System Block Diagram

This power system uses a star distribution layout to prevent current from one module from flowing through another. The 12 V barrel jack feeds a central copper pad that acts as the common source node, and each sub-assembly containing a NEMA 17 stepper motor, A4988 driver, and STM32 microcontroller is connected directly to that point through its own 12 V line. This ensures that each module draws current independently from the main supply instead of passing it through neighboring boards. The result is a cleaner and more predictable current path, avoiding shared trace heating or uneven loading that could occur in a daisy-chained setup while keeping the system modular and easy to expand. An estimated current draw for each sub-system is shown below.

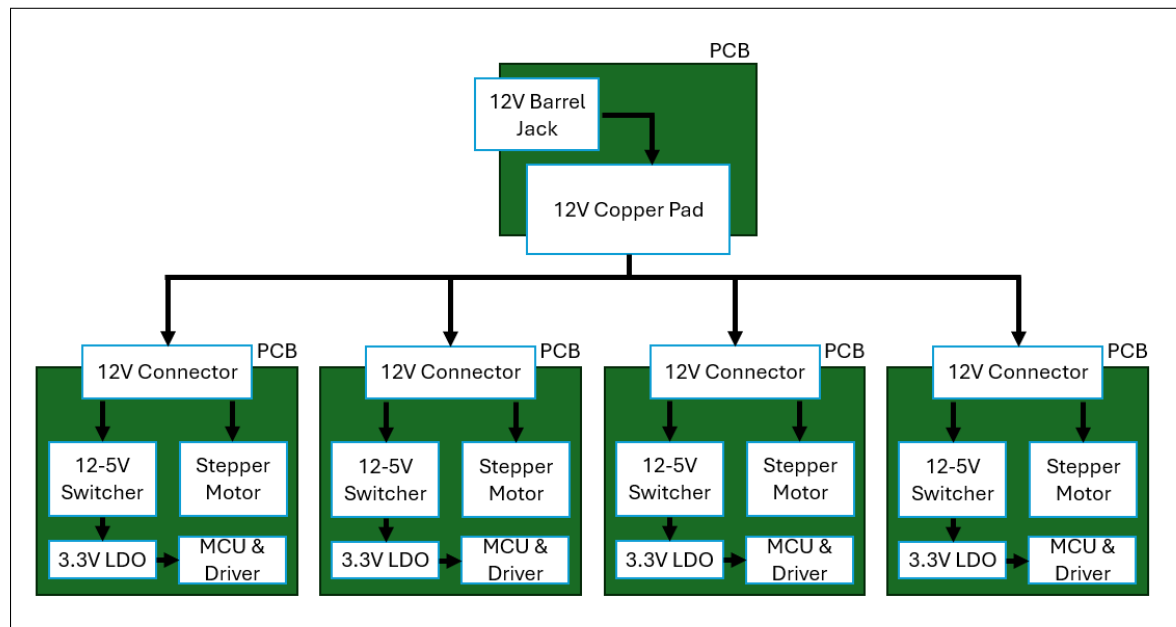


Fig. 8: Power Supply Block Diagram

c) Switching Regulator Selection

The TPS562201 buck regulator was chosen because it provides a compact, efficient, and modern solution for generating the 5 V rail from a 12 V input. Its high switching frequency allows the use of much smaller inductors and capacitors compared to older regulators, which reduces board area and improves transient performance. The device uses synchronous rectification, eliminating the need for an external Schottky diode and significantly increasing efficiency under both light and heavy loads. In addition, the low quiescent current and integrated protection features make it well suited for digital systems that require stable, low ripple power. Overall, the TPS562201 offers higher efficiency, smaller component count, and better electrical performance than legacy options while maintaining reliable and predictable operation in a compact footprint.

d) Switching Regulator Backup

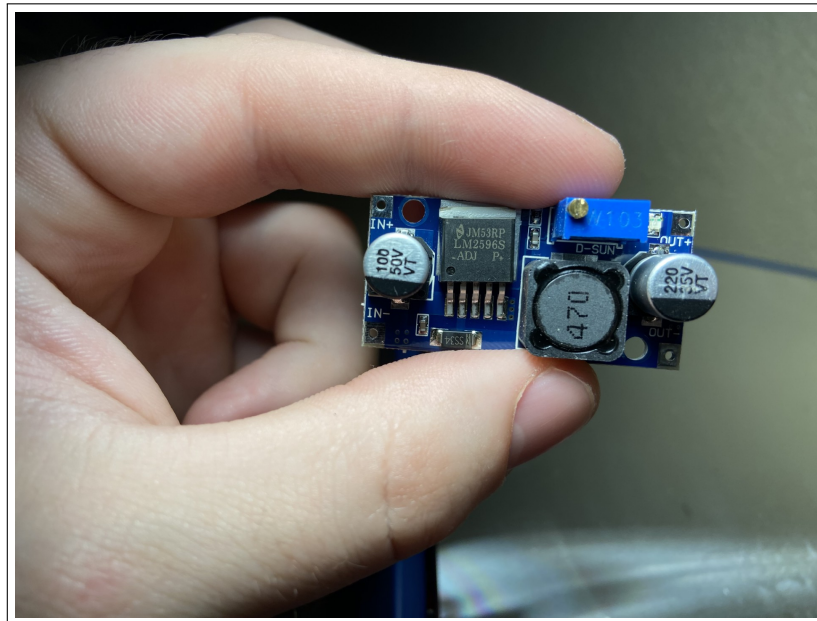


Fig. 9: LM2596 Board

To ensure that the power system remained robust even in the event that the primary TPS562201 design failed or could not be sourced, I purchased several off-the-shelf LM2596 buck converter modules and desoldered them on the hot plate to expose and study the underlying layout. Reverse engineering these boards allowed me to verify the inductor selection, diode rating, feedback network values, and grounding strategy used in a commercially functional design. Although the LM2596 is an older and less efficient regulator, its widespread availability and tolerance for larger passive components make it a reliable fallback option. By documenting the topology and component choices of the module, I created a validated backup design path that could be quickly implemented if the primary high-frequency buck converter required revision or if component shortages occurred. Although the TPS562201 remains the primary regulator in my design, the reverse-engineered LM2596 module serves as a fully validated backup option in case the main converter ever needs to be substituted.

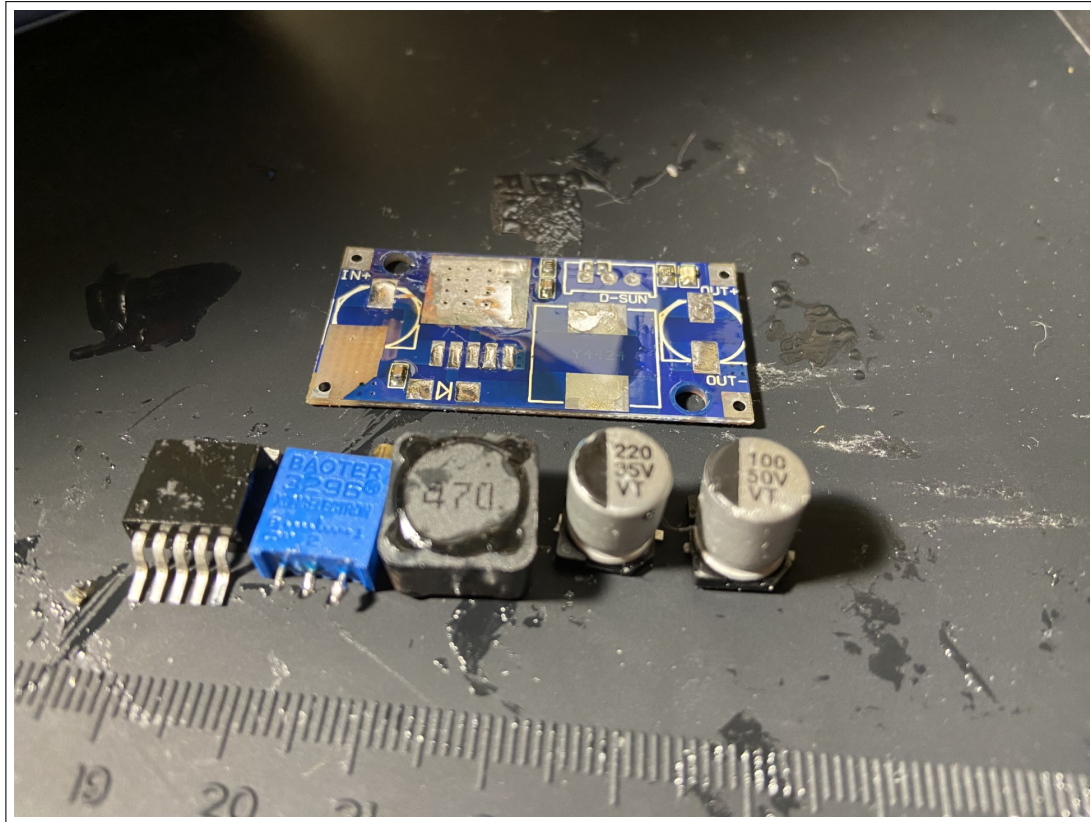


Fig. 10: LM2596 Desoldered

3.5 Photosensor Research

The transmissive and reflective spectra from the test lens will be measured using some

The photosensors will measure the intensity of the single-color light output by the monochromator. One will be for the light that is transmitted through the lens, and one will be for the light that is reflected from the lens. Once we build these two spectra, we can compare them with a catalog of part numbers or lens materials to find the best match for our test lens.

At normal incidence of the light on the test lens, the transmissive photosensor can go directly in front of the test lens along the optical axis, though it would block the IAS and thus need to be moved out of the way later on in the process. However, the reflective photosensor cannot be put along the optical axis because to be in front of the lens on the axis would block the light from reaching the lens at all.

In order to remedy this, we have decided to rotate the lens mount using a stepper motor. The reflective photosensor will be fixed in place and the motor will rotate until the reflected light meets the diode (Fig.11). After the spectral data has been taken, the motor will return the test lens to its original position, normal to the optical axis.

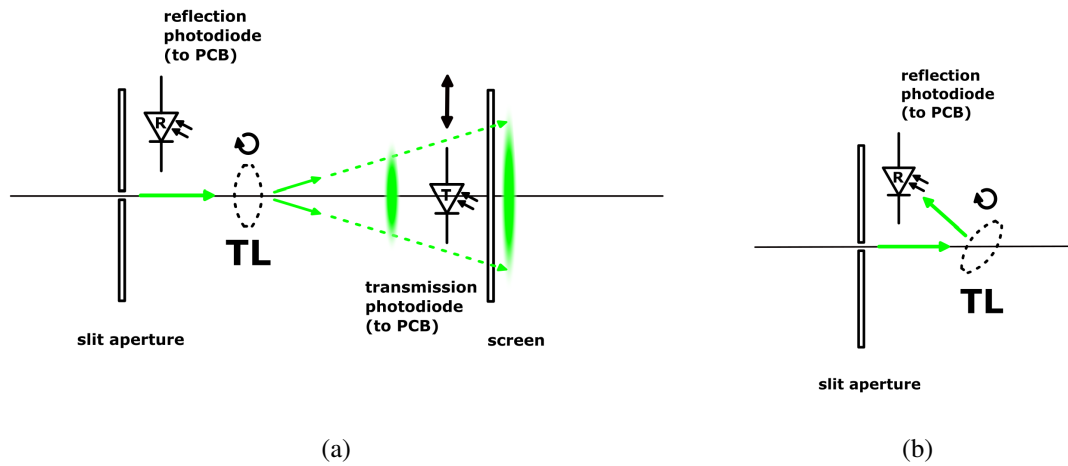


Fig. 11: A schematic of the photosensor region of the project aligned for collection of (a) the transmissive spectra and (b) the reflective spectra

We tested the concept of rotating the lens by simulating several cases in the Phydemo Ray-Optics Simulator [69]. This is an open-source optics simulator that works in a web browser. The “default” lens created for this experiment was a spheric lens with radii of curvature of $\pm 300\text{mm}$ and a center thickness of 50mm in front of a collimated light source and a ruler for scale.

First, we compared the reflected light at different rotations of the lens (Fig.12a). Staying centered, we tested rotations of 0° , 30° , and 45° . As predicted, the reflected light at 0° overlaps with the incident light, making it impossible to measure accurately. At 30° , the reflected light is pointed out of the incident beam with relatively good efficiency. At 45° , the critical angle for total internal reflection is reached for the lens and some of the light that is reflected by the lens never makes it out. From this data, we continued testing at our optimal wavelength of 30° .

Once we found a good sample rotation angle, we investigated how the thickness of a lens impacts its reflected beam (Fig. 12b). We compared thicknesses of our default lens (50mm) to a thickness of 75mm and 15mm respectively. The thickness of the test lens does not seem to have a significant impact on the reflected light from the source.

Still at our sample rotation angle of 30° , we tested the effects of radius of curvature on the reflected beam (Fig. 12c). We compared our default lens ($\pm 300\text{mm}$) to a $\pm 450\text{mm}$ and a $\pm 150\text{mm}$ lens respectively. Changing the radii of curvature (and thus the focal length) of the lenses had a visible effect on the location of the reflected beam. The $\pm 450\text{mm}$ lens had the shortest focal length of the three, and similarly to a high angle of rotation, became susceptible to total internal reflection. This issue may be solved by changing the angle of rotation and/or the placement of the target photosensor.

Regarding the transmissive photosensor, it will still have to be along the optical axis. In every case of the simulation, the transmitted light from the test lens was still generally

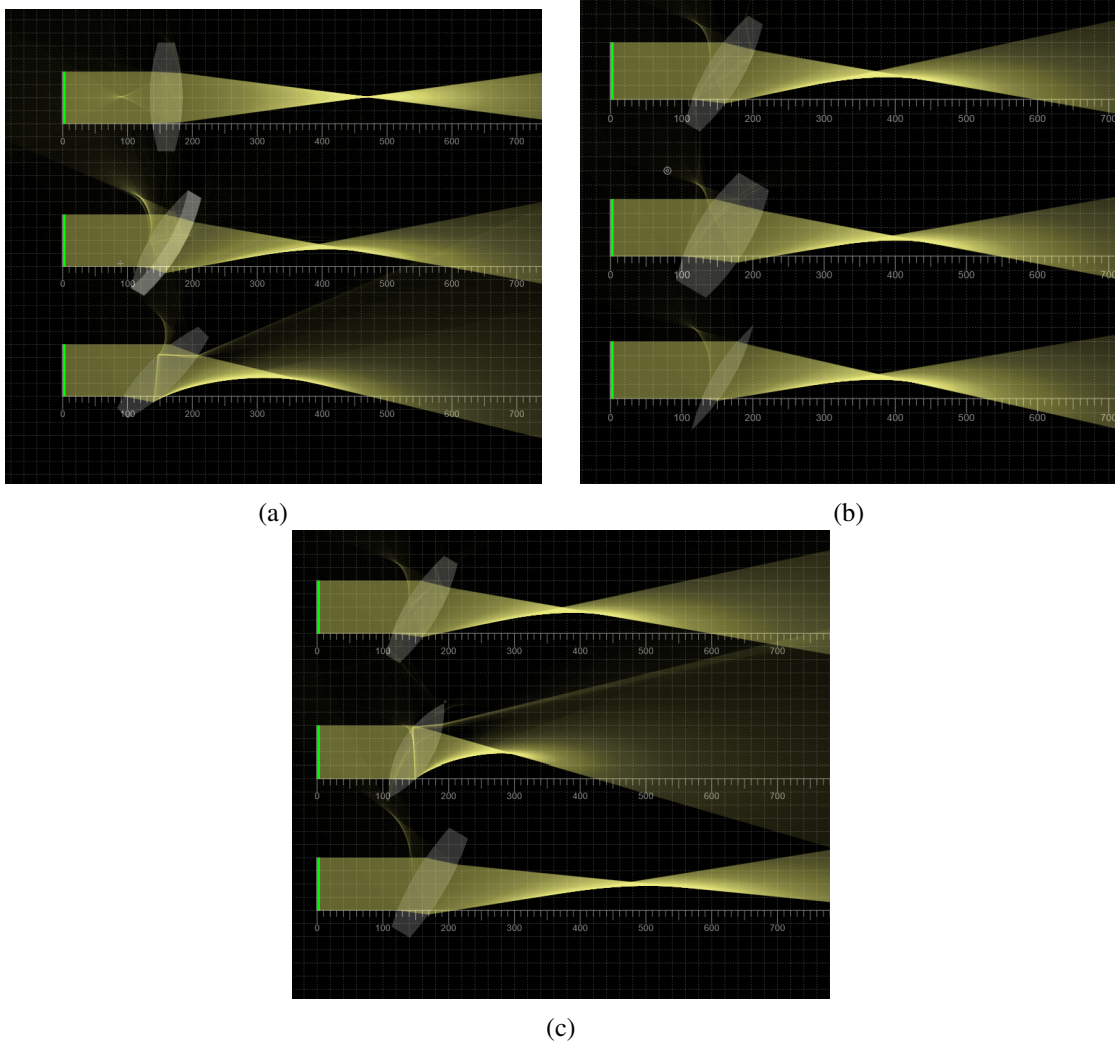


Fig. 12: A simulation of spherical lenses in a beam of light. (a) shows the beam behavior for the same lens at 0° , 30° , and 45° rotations. (b) shows the beam behavior for the same lens at 50mm, 75mm, and 25mm center diameters. (c) shows the beam behavior for the same lens at $\pm 300\text{mm}$, $\pm 450\text{mm}$, and $\pm 150\text{mm}$ focal lengths

along the axis. One way to measure this light is to have a photosensor slide in laterally into the lens' transmitted beam at normal incidence. After the data is taken, the photosensor will return to its original position, out of the way so that the light can fully intersect with the screen for the IAS. This concept can be seen in Fig. 11a.

3.51 Photoresistors vs. Photodiodes vs. Phototransistors

There are three main options for converting an optical signal to an electrical one are photoresistors, photodiodes, and phototransistors.

A photoresistor acts as a normal resistor under dark conditions. When illuminated, its resistance decreases as the amount of incident light increases.

A photodiode is based on a PN junction diode with its junctions encased in a transparent material so that they can be exposed to light. When illuminated, a current flows through the junctions in what is known as the photoelectric effect. This effectively makes the photodiode a switch; under dark conditions current flows one way, and when there is light it flows the other way.

A phototransistor is a combination of two photodiodes that act together as a bipolar transistor. Instead of a base voltage to allow the flow of current, a phototransistor requires incident light for current to flow.

3.52 Photodiode Component Selection

The choice of photodiodes over other photosensors was made for higher performance at a lower cost. After the monochromator, two photodiodes will be utilized in photosensors placed in front and behind the test lens respectively. The photodiode in front of the test lens will measure its reflected intensity over the visible spectrum, and the photodiode behind the test lens will measure its transmitted intensity over the visible spectrum. By gathering this data, we will be able to compare the test lens against the spectra of known materials to estimate its composition.

Table V compares the three options. Photodiodes have a clear advantage to us over photoresistors and phototransistors due to their sensitivity and response time.

3.53 Photodiodes

When it comes to choosing which photodiodes to use, the optical criteria to consider are spectral range, peak wavelength, and response time. Since we are working in the visible portion of the EM spectrum, the photodiodes should be at least functional from 400 nm to 700 nm wavelengths. Similarly, the peak wavelength should be somewhere in this range, preferably near the middle for maximum efficiency.

3.54 Sensor Design Research

Photodiodes by themselves make poor sensors. This is due to tiny output current, noise and drift, and slow unloaded response time.

A photodiode generates current linearly proportional to its sensed optical power. This is modeled by the following equation.

$$I_{\text{photo}} = R_{\lambda} P_{\text{opt}}$$

where R_{λ} is the responsivity (A/W), and P_{opt} is the optical power. R_{λ} has a typical value of around 0.2-0.6 A/W, meaning an entire Watt of optical power will only create a current response in the range of few hundred milliamperes.

The photodiode can be modeled as a current source I_{PD} in parallel with a junction capacitance C_J , a shunt resistance R_{SH} , and a series resistance R_S . This equivalent circuit captures both the optical-to-electrical conversion and the primary parasitic effects that

influence the photodiode's response.

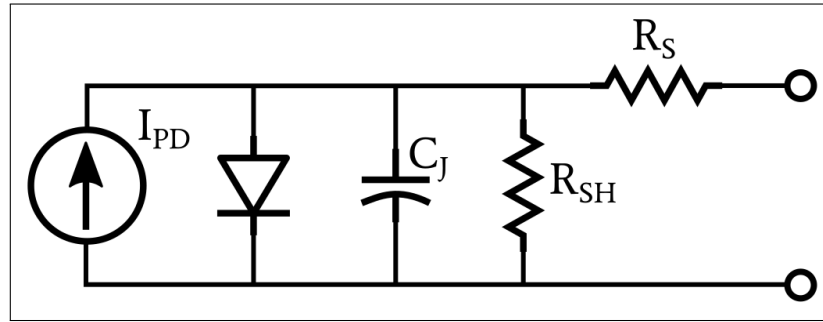


Fig. 13: Photodiode Equivalent Circuit[33]

This means that signals are easily distorted by any stray capacitances or input impedances of the measurement equipment, as well as causing a slow unloaded response time. In order to work around these inherent problems, a transimpedance op-amp topology can be employed.

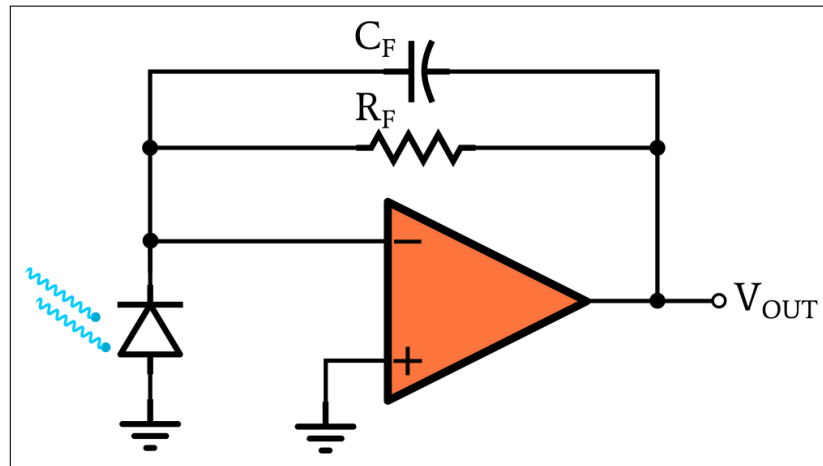


Fig. 14: Transimpedance Op-Amp[34]

In a photodiode amplifier circuit, the feedback resistor R_F converts the photocurrent from the diode into a voltage, setting the overall gain of the circuit. This is calculated as

$$V_{\text{out}} = -I_{\text{PD}}R_F$$

The feedback capacitor C_F is added in parallel with R_F to stabilize the amplifier and control its bandwidth. Without C_F , the circuit could become unstable due to the photodiode's junction capacitance. This creates a frequency dependent gain of

$$V_{\text{out}}(s) = -I_{\text{PD}}(s) \frac{R_F}{1 + sR_FC_F}$$

There is a problem with this configuration in that the output is a negative voltage. To work around this a clever design from Texas Instrument's can be implemented.

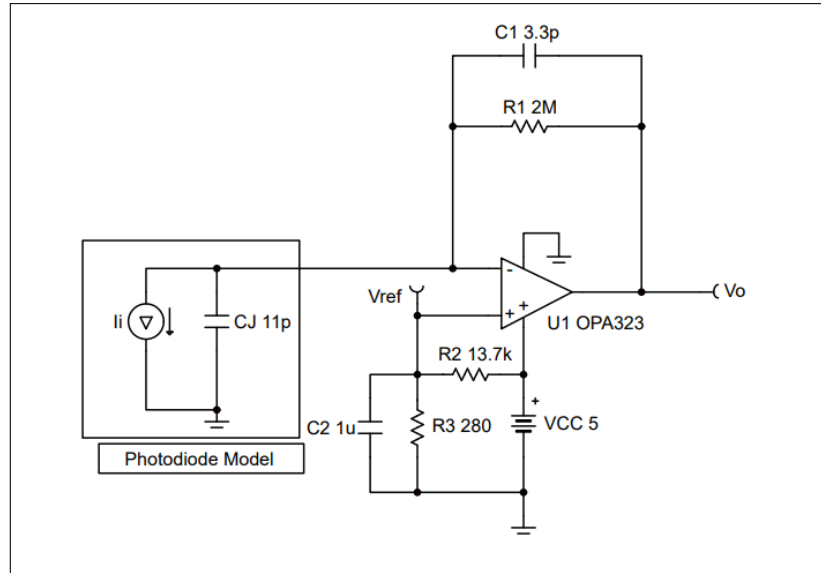


Fig. 15: Transimpedance Op-Amp Design from TI[32]

By utilizing a reference voltage (V_{ref}) to the non-inverting input terminal the output becomes

$$V_o = V_{ref} - I_i R_f$$

The output remains inverted, meaning a higher output voltage corresponds to lower light intensity; however, it is now limited to the positive range between ground and V_{cc} . This inversion in logic can be easily corrected in software.

3.55 Op-Amp Component Selection

The table in Fig. 12 compares several precision operational amplifiers from Texas Instruments (OPA328, OPA392, and OPA322). Each device features rail-to-rail input and output capability, low input offset voltage, and wide supply voltage compatibility. The main differentiators are quiescent current consumption, unity-gain bandwidth (UGBW), and slew rate (SR).

The OPA328 offers the highest performance, with a 40 MHz UGBW and 30 V/ μ s slew rate, making it well suited for high-speed or wide-bandwidth sensor interfaces. However, this comes at the cost of higher quiescent current (3.8 mA per channel) and a correspondingly higher unit price on Digi-Key of \$2.99 a unit in low order quantities. [15]

The OPA392 provides the lowest power consumption (1.22 mA per channel) and exceptional input bias characteristics (10 fA), optimized for low-frequency and high-impedance measurements. Its limited bandwidth of 13 MHz and slew rate of 4.5 V/ μ s make it less

	OPA328	OPA392	OPA322
V_{cc}	2.2V to 5.5V	1.7V to 5.5V	1.8V to 5.5V
V_{inCM}	Rail-to-rail	Rail-to-rail	Rail-to-rail
V_{out}	Rail-to-rail	Rail-to-rail	Rail-to-rail
V_{os}	3 μ V	1 μ V	0.5mV
I_q	3.8mA/Ch	1.22mA/Ch	1.6mA/Ch
I_b	0.2pA	10fA	0.2pA
UGBW	40MHz	13MHz	20MHz
SR	30V/ μ s	4.5V/ μ s	10V/ μ s
#Channels	1 and 2	1, 2, and 4	1, 2, and 4
Link	OPA328	OPA392	OPA322

Fig. 16: TI Op-Amp Comparison[32]

ideal for fast-changing signals but very stable for DC coming in at a price of \$2.31 a unit in low order quantities. [15]

The OPA322 occupies a middle ground, offering moderate bandwidth (20 MHz) and a reasonable 10 V/ μ s slew rate while maintaining low power consumption. It comes in a price of \$1.36 a unit at low order quantities and \$0.98 per unit in an order of ten. [15]

OPA323	
V_{cc}	1.7V to 5.5V
V_{inCM}	Rail-to-rail
V_{out}	Rail-to-rail
V_{os}	0.15mV
I_q	1.6mA/Ch
I_b	0.5pA
UGBW	20MHz
SR	33V/ μ s
#Channels	1, 2, and 4
Link	OPA323

Fig. 17: Chosen Op-Amp[32]

The OPA323 was selected for its low offset, low bias current, and rail-to-rail performance. Its 20 MHz bandwidth and 33 V/ μ s slew rate provide fast, accurate response while maintaining low power use and cost. The OPA323 came in at a stunningly low \$5 per batch of 10, allowing easy prototyping without fear of breakage. [15]

3.6 White Light Source

The white light source (WLS) is a broadband source that provides illumination for the rest of the system.

It has three major components to consider. First, we considered the type of source and resulting intensity vs. wavelength plot. Then the collimation and the output of the source. Finally the compression and filtering processes that produces a thin, high-power beam with **only** collimated light. Whose light is passed to the monochromator.

3.61 Source Type

There are three main types of sources used in laboratory illuminators: LED, Halogen, & Xenon. The "quality" of the light proceeds in that order from lowest to highest. Fig. 18 shows an overview of the different light source options and their spectral density.

Halogen sources are typically used in laboratory environments due to their low cost, however Xenon sources feature an almost uniform spectral density from 200 to 800 nanometers. Notably, Xenon includes significant illumination in the UV and IR ranges, which comes with safety and material considerations.

From our research, it is abundantly clear that Xenon provides both the most consistent and wide range of illumination over the visible spectrum. The only concern being cost.

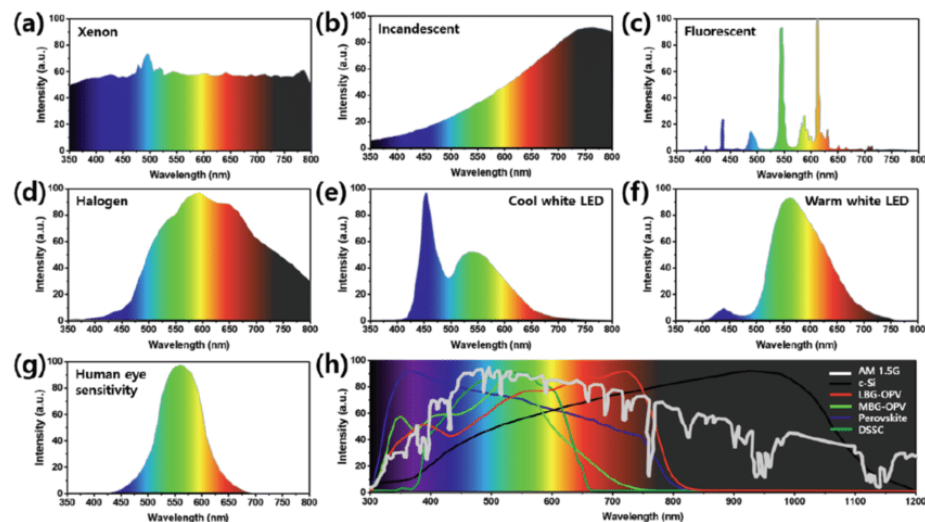


Fig. 18: Spectra of different light sources, [35]

Normally, Xenon light sources are prohibitively expensive, with the average Newport system around \$10k. However, there is another common option for Xenon bulbs – HID headlights.

This is a mature market in which competition has driven the price of Xenon gas discharge tubes down to the 10s of dollars. We were able to purchase a system including the ballast

(12V to HV system) for \$30. This brings otherwise research-level light sources to the hobbyist level.

This part of the system is an initiative led by author Kiva McCracken and the design will be released in the future at kivamccr.xyz.

Heat is a very important consideration for these types of light sources. Since the source produces so much IR radiation, any dark materials instantly get hot. Even when placing your hand in front of the beam heat is felt radiating from it. We may eventually consider removing the IR via a dichroic mirror.

Furthermore, there's the conduction and convection heat to worry about. As a result the illumination system requires fans to keep the bulb and surrounding material cool.

The system used is rated at 80W, using a luminous efficacy of 32lm/W from a similar device [36] we can estimate a brightness 2560 lumens. Correspondingly the device is approximately 203 candelas.

3.62 Focusing

Once the light is generated, it must be focused into a tight beam. This starts with getting the light out of the source as a collimated beam. There are a few different options for this:

- Conical reflector. This is lossy, but cheap.
- Parabolic reflector + retro-reflector. This allows the alignment of both outputs into a single cone, but is complex and requires substantially more fabrication.
- Parabolic reflector + aspheric lens. This setup is typically used in professional lighting systems. It allows the collimation of both the forward aperture light and the reflected light.
- Micro parabolic reflector. This produces a smaller beam but less consistent collimation.

Initially the conical reflector was tested, simply a rolled up cone of aluminum. This got the light out, however with effectively no directionality making it near impossible to collimate efficiently.

Via simulation we tested a parabolic reflector + retro-reflector system, which has near 100% efficiency.

The retro-reflector is effectively a cylinder that causes any forward facing light to be returned towards the source and thus the small backing reflector.

The backwards facing light is instantly collimated, whereas the forward facing light first does a round trip with the retro-reflector. Then the light can be collimated using a standard lens based beam compressor.

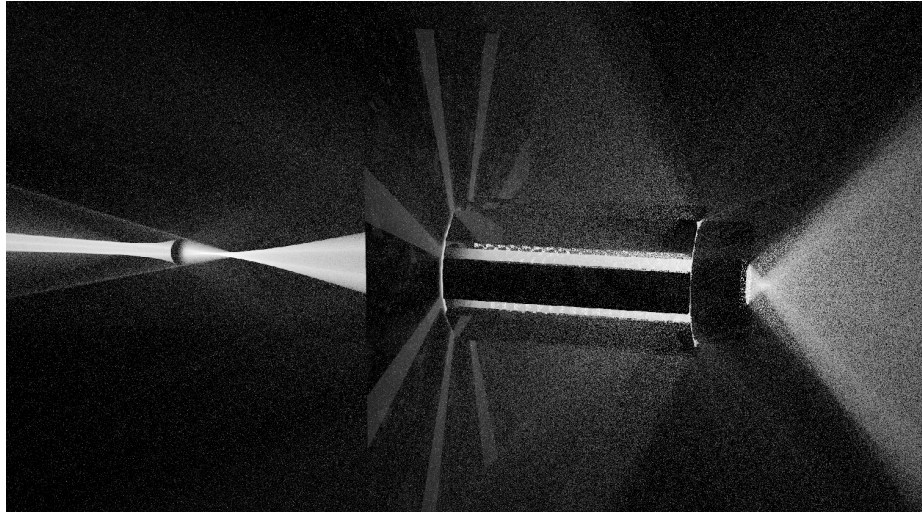


Fig. 19: Parabolic reflector + retro-reflector system simulation.

Initially we considered producing this custom however due to size limitations of the engineering clean room sputtering chamber we abandoned the idea.

Then two larger parabolic reflectors were tried, one from a stage light and the other custom.

The stage light reflector had a focal length of approximately 21mm and required the bulb to be inserted along its axis, this ruled out any use of an asphere to collimate the forward light however demonstrated that a large parabolic reflector paired with collection lens could focus to about a 50mm spot before diverging. Check out the model in fig. 20.

In order to solve this we moved onto a reflector with a focal length matching such that a bulb could be placed diagonally allowing vertical room for the aspheric mirror. However in order to do this we would need to build our own reflector.

That left many choices with regard to materials. First, we will cover reflector materials in general.

The system is looking to support between 400 & 700nm wavelengths. Different materials have different reflectivity vs. wavelength curves. Fig. 21 shows aluminum (Al) as the clear winner maintaining around 90% reflectivity from .2 to 2 μ m.

Clearly aluminum is best option, and that matches up with industry. Most parabolic reflectors are vacuum coated with aluminum either by sputtering or chemical vapor deposition. Optical quality parabolic reflectors are also quite expensive.

To manufacture a reflector we could choose custom CNC lathed surface + polishing, 3D printed with SLA + electroplating, or FDM + aluminum tape. After weighing all the costs we decided to try latter and only spend the money if it didn't meet the desired performance.

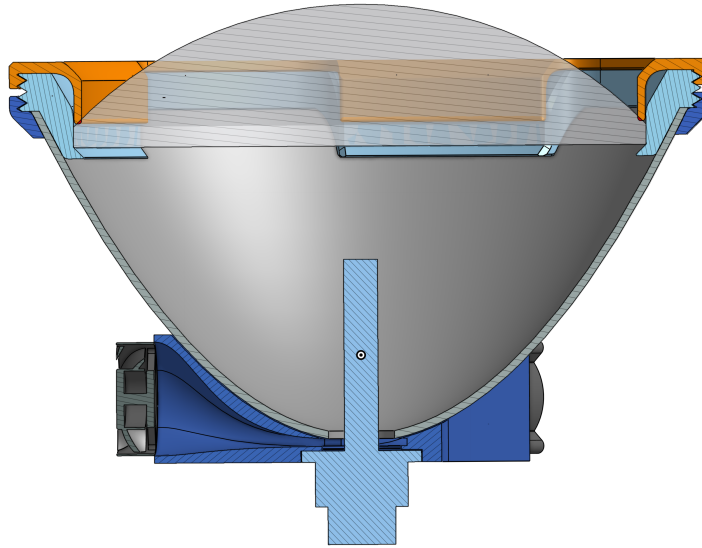


Fig. 20: CAD model of first reflector, featuring a cross sectional view.

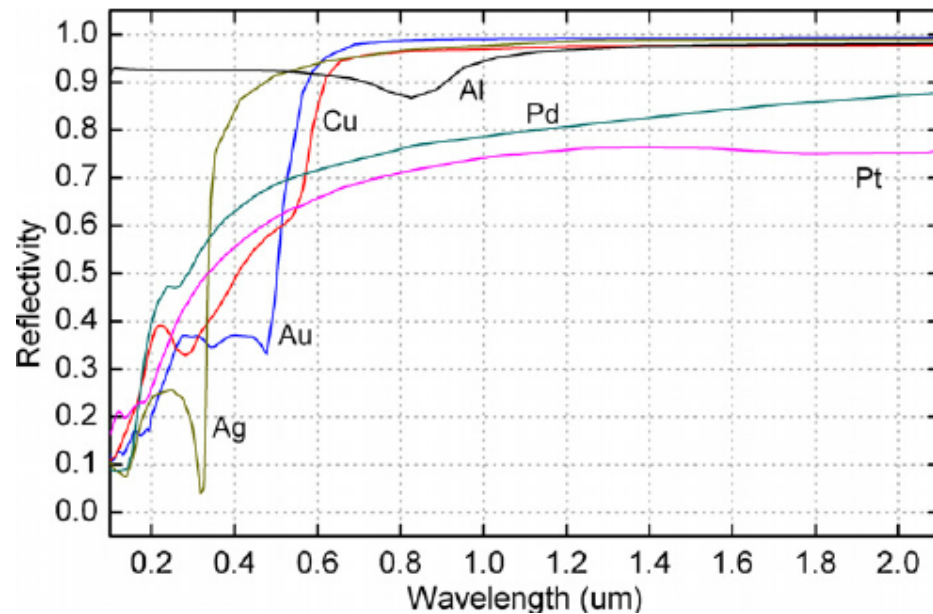


Fig. 21: Reflectivity of some common metals versus wavelength at normal incidence [18]

To handle the heat we chose HT-PLA-GF by Polymaker [73], this filament withstands heat without deflecting up to 150°C after annealing while being printable on conventional 3D printers. These properties make HT-PLA-GF the perfect material to withstand heat of the bulb.

Furthermore to reduce heat twin ducts were placed feeding into the focal point to actively cool the bulb.

Initially we tried scaling the reflector to achieve a smaller form factor, however parabolic reflectors only work in the exact shape defined by:

$$y = \frac{x^2}{4f} - f, \text{ where the focus is at the origin} \quad (2)$$

First an intricate 3D model was made of the bulb to help define any interferences and the position of the point source. Around that a parabolic curve was generated, scaled and placed so that the point source was at the focal point and the bulb was as horizontally oriented as possible in relation to the parabolas axis. This allows the aspheric lens to be placed essentially right in front of the point source without intersecting the glass of the bulb as a result occupying the full forward numerical aperture while affecting as little of the reflected light as possible.

To maintain a smaller/realistic size the focal length of the reflector needed to be between 10 & 40mm. The light source was then inserted such that the xenon gas cavity is centered on the focal point. However in order to achieve a numerical aperture that matches the parabolic mirror the lens must be positioned as closely to the focal point as possible.

Initially we considered using a fresnel lens to collimate the forward light. Unfortunately most fresnel lenses are manufactured out of plastic and the light intensity of incident IR light could risk melting the lens. So instead we opted for a glass aspheric lens. While we did go forward with ordering the lens, we did not integrate it into this setup.

The structure was then carefully covered with aluminum tape. Providing a reflective surface at low cost and with the benefits of aluminum's spectral reflectivity, fig. 21.

From there the large collecting lens WLS-BC-1 is mounted, note that the lens is not flush with the parabolic mirror so that air is able to flow. This represents the first element of beam compression. A

beam compressor/expander consists of at least two lenses with positioning and magnification as depicted in Fig. 22. Typically there is a large lens and a small lens, with a stop positioned at the shared focal point to "purify" the light.

The final build of this test system is visible in fig. 23.

Far earlier in the project on a whim a small 40mm diameter flashlight parabolic reflector was bought, clearly CNCd out of aluminum and polished to a perfect finish. When it arrived it was modified to fit the HID bulb.

Upon testing this single component held just in hand created a compact highly collimated beam (This was both delightful and incredibly frustrating for the authors at the time).

Resolving on this we then chose to integrate the small parabolic mirror into a small

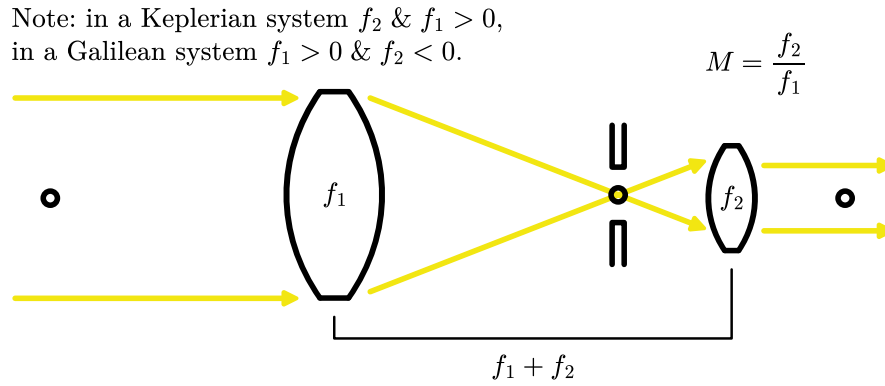


Fig. 22: Beam compressor/expander optical diagram & equations.

compact box design, with threading allowing future beam compressor components.

This system also integrates an internal power supply, fuse, and relay allowing for easy iteration to the final project but also as a stand alone system for future experimental use. An exploded view of the compact system is visible in fig. 24 & 25. Note however that it does not feature a completed optical axis, only the parabolic mirror.

3.7 Monochromator

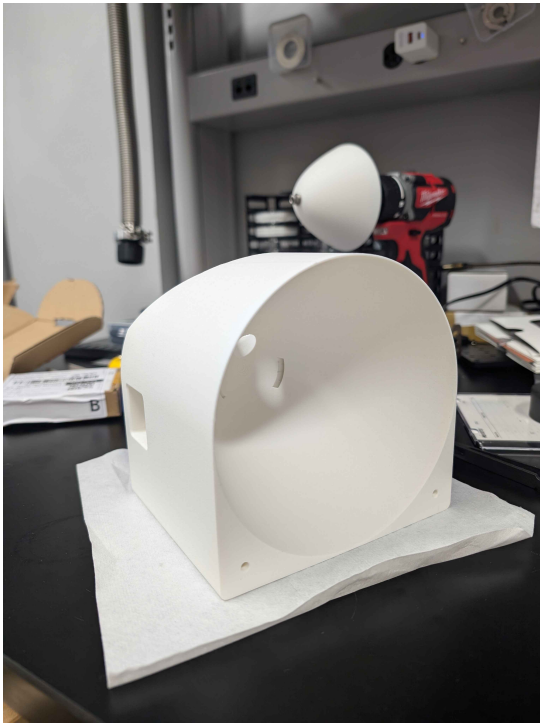
The monochromator will select specific wavelengths from the WLS to interact with the test lens and generate data. By measuring the angles at which light is refracted as a function of wavelength, we will be able to calculate the Abbe number of the test lens. By measuring the intensity of the light transmitted and reflected by the lens as a function of wavelength, we will be able to estimate the lens' material. To build a monochromator, we must use a diffraction grating to split the white light into its constituent wavelengths. Then, we must direct a beam of single-wavelength light into the test lens at normal incidence. This will be done by mirrors and iris apertures.

3.71 Diffraction Grating

The core component of a monochromator is the diffraction grating. A diffraction grating is a surface with a periodic surface structure that causes different wavelengths of light to propagate at different angles [53]. These propagation angles can be derived from the following equation (visualized in Fig. 26):

$$m\lambda = d(\sin \alpha + \sin \beta) \quad (3)$$

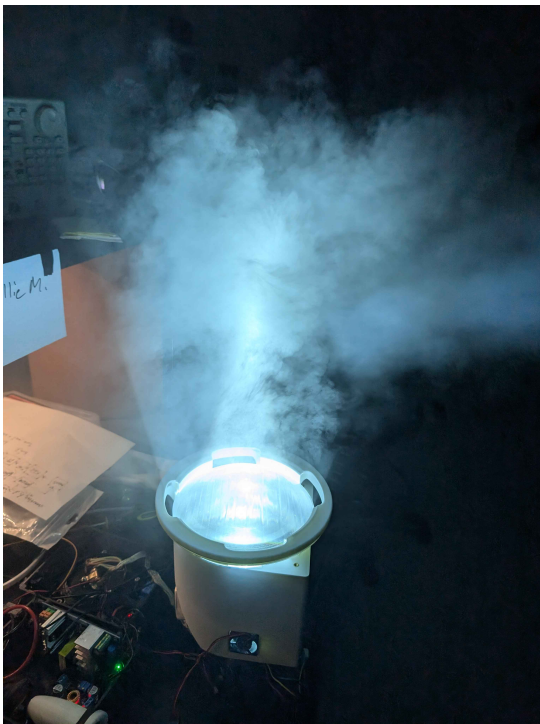
Where β is the diffraction angle relative to the grating's normal vector, α is the incident angle of light, m is the diffraction order, or mode, λ is the diffracted wavelength, and d is the distance between grooves [29]. Note that the 0 order mode ($m = 0$) acts as simple reflection, representing the case $\alpha = \beta$.



(a) 3D printed HT-PLA-GF reflector body.



(b) HID bulb is mounted at an angle in the reflector.



(c) Light cone formed by single large collection lens and reflector.



(d) Semi-collimated beam created via a beam compressor configuration

Fig. 23: Second parabolic reflector system test.

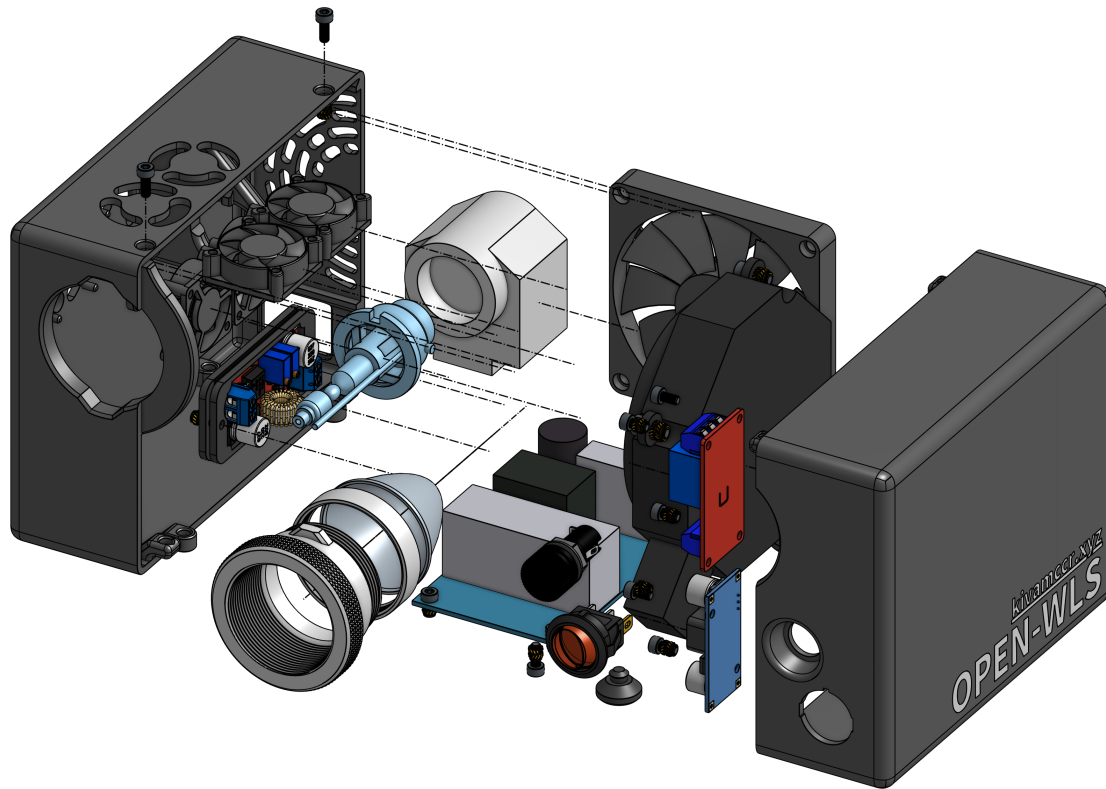


Fig. 24: WLS compact case exploded view.

The first mode ($m = 1$) is the greatest in terms of output power at a single wavelength [53]. Since the power will be significantly cut by the aperture, we will be using the brightest mode to compensate. Additionally, diffraction gratings can either be reflective and bounce light off of them, or transmissive and allow light through them. Transmissive gratings work very similarly to their reflective counterparts, using a periodic array of slits instead of grooves. However, transmissive gratings introduce more loss than simple reflection [53], so for the same reason of maximizing power at a single wavelength our final choice for the part was a reflective grating.

We can infer from the above equation that the distance between grooves d is inversely related to the total angle ($\sin \alpha + \sin \beta$). If we are to properly compare different diffraction gratings, we would assume the incident angle α to stay constant for now. Most manufacturers specify the distance between grooves on a diffraction gratings in terms of lines/mm or $\frac{1}{d}$. With this information in mind, we have inferred one step further to say that there is a positive relationship between the diffraction angle β and lines/mm $\frac{1}{d}$. A wider diffraction angle means a greater spread in the resulting spectrum, which in turn means more accurate wavelength selection. Therefore, our design should have grating with a high number of lines/mm for a high level of wavelength purity.

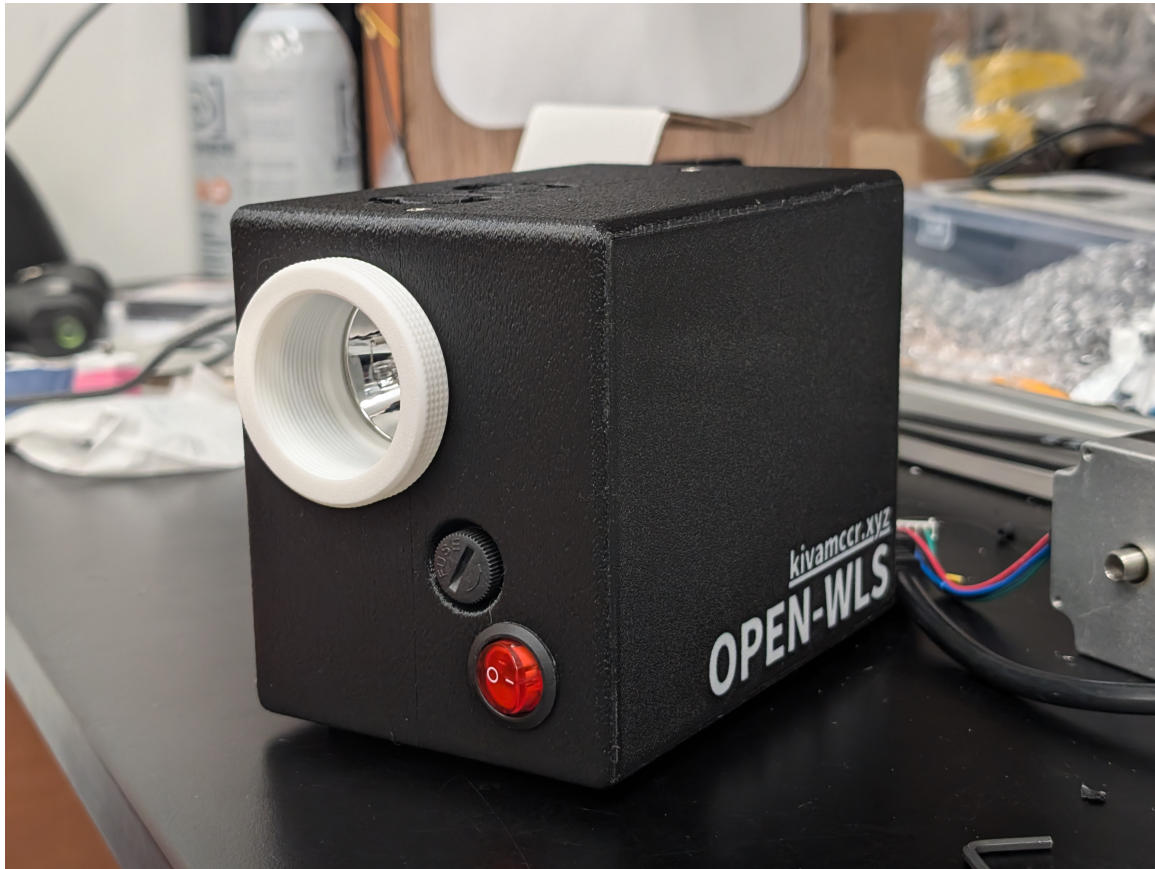


Fig. 25: Photo of in development WLS.

The three grating periods we found to be most commonly available for use with visible light were 300 lines/mm, 600 lines/mm, and 1200 lines/mm. A quantitative comparison of their spectral width is shown in Table VI.

Within reflective diffraction gratings, there is also the question of whether the grating should be ruled or holographic. The shape of the groove profile, broadly speaking, is triangular or trapezoidal for ruled gratings and sinusoidal for holographic (Fig. 27)[30]. Notice that ruled gratings (a) are usually not symmetrical, while holographic gratings (b) are symmetrical. This means that the grating is most effective when light is coming from a certain direction (left, in the case of Fig. 27) for ruled gratings, but the equally effective when light is coming from multiple directions (left or right) for holographic gratings [29]. For our final design of the monochromator, only a single diffraction mode is used so this difference is ultimately of little consequence.

All diffraction gratings have a varying efficiency over their operating wavelengths due to their groove geometry. For the purposes of this project, a grating with high efficiency over the entire visible spectrum is desirable. This way we will only need to make minimal corrections for the monochromator's output power. A grating's nominal wavelength is a specification given as the peak of its efficiency curve. Generally speaking, holographic gratings tend to have broader efficiency curves than ruled gratings. In theory, this makes

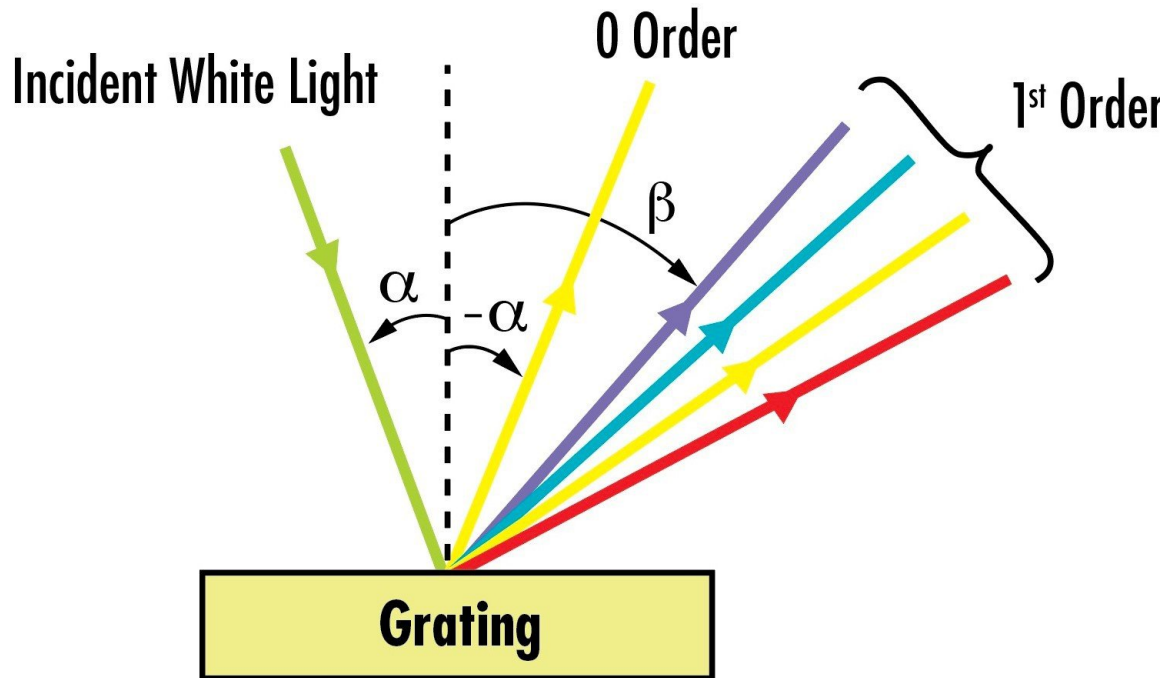


Fig. 26: Diffraction grating diagram (reflective) [29]

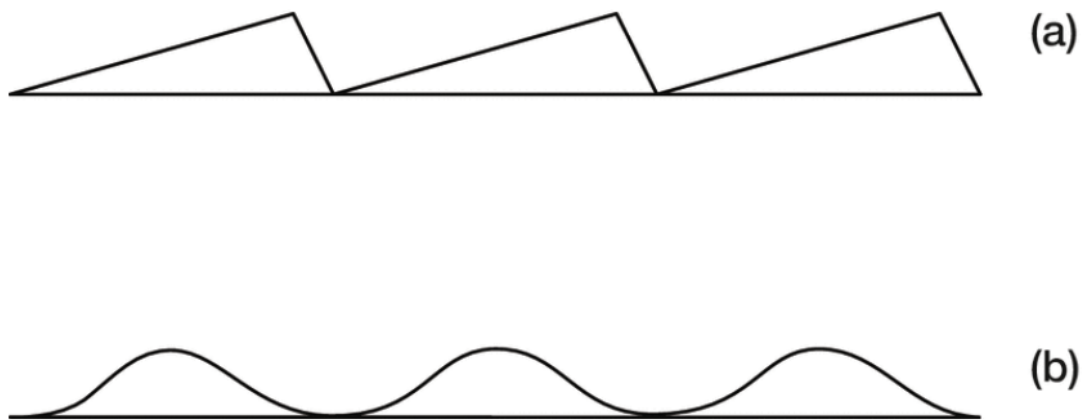


Fig. 27: (a) blazed diffraction grating vs (b) holographic diffraction grating [30]

holographic gratings more desirable because they would require less correction during data analysis. However, ruled gratings tend to have higher peak efficiencies [53], and for the sample wavelengths we will be using defined later in this paper, the Thorlabs GR13-1205 reflective diffraction grating [66] had the highest efficiency in the region of interest. Its perpendicular spectral efficiency, as shown in Fig. 28, is between 50% and 85% for the rough operating range of 400 nm to 800 nm. Since the white light will be close to normal incidence to the grating, this is the primary curve on the graph we need to consider. A more detailed breakdown of the grating choice is given in section VIIA.

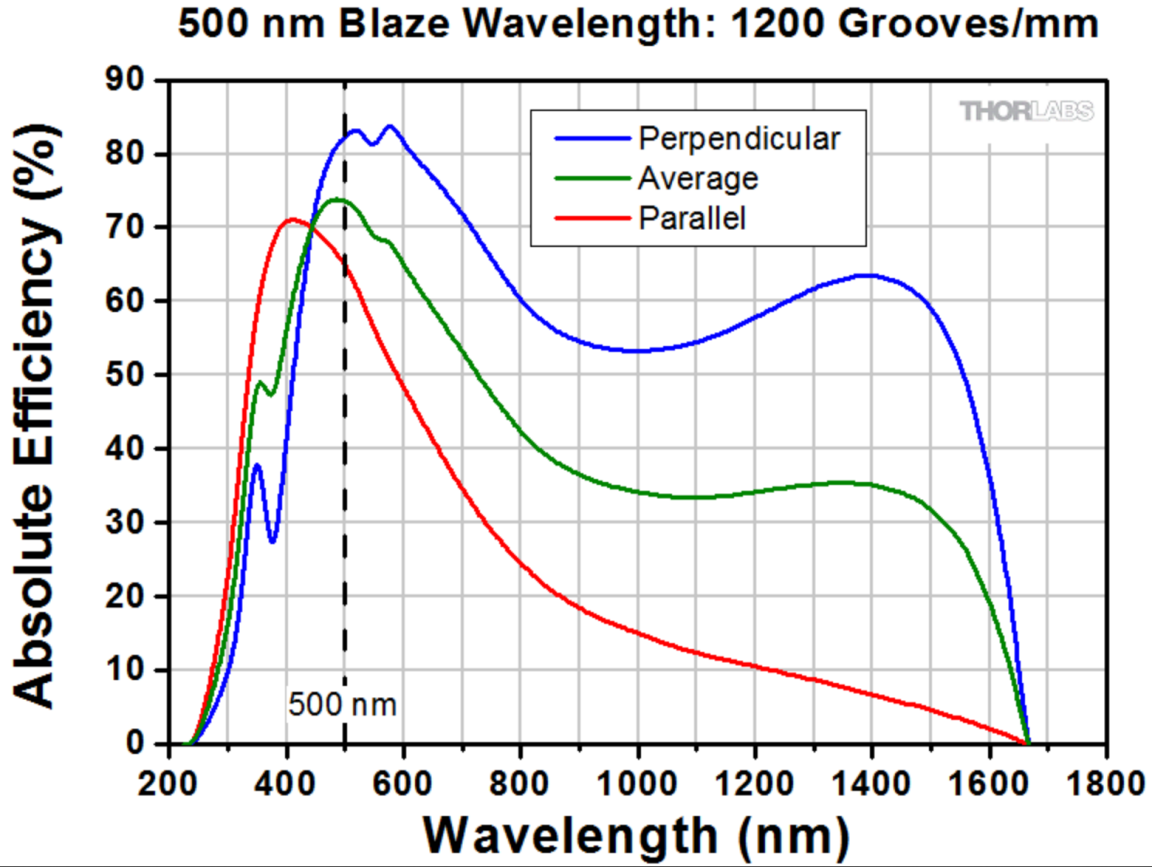


Fig. 28: Thorlabs GR13-1205 spectral efficiency curve [66]

3.72 Prototype 1

The original design of the monochromator prototype consisted of a static 1200 lines/mm diffraction grating normal to the WLS beam. Two mirrors, for $m = 1$ and $m = -1$ respectively, would be posted equal distances away from both the grating and the optical axis set by the WLS beam. The mirrors would be controlled by a single MGS90S servo [50], connected via a gear system so that they rotate in equal and opposite directions to reflect the corresponding spectra of the two modes.

This design was tested according to the schematic shown in Fig. 29, where θ is the specified angle of the mirrors normal to the optical axis needed to select a certain wavelength. θ is equal across both mirrors since the design is symmetrical across the optical axis. The value of θ can be derived by constructing a right triangle with legs tracing the optical axis and its normal. From here we see that θ is supplementary to β . Since the incident angle α is fixed at 0° from normal, it can be removed from equation 3. Lastly, we are only considering modes $m = 1$ and $m = -1$. This modified equation can then be rearranged to solve output wavelength λ as a function of mirror angle θ :

$$\lambda = \pm d \sin(90^\circ - \theta) \quad (4)$$

A physical model was then built with the materials available from the CREOL undergraduate laboratory. A green laser was used for alignment since it is a single wavelength in the middle of the visible spectrum, which enables precise tuning of the geometry for the two mirror spots to overlap. After alignment, the laser was replaced with a broad white light source for testing.

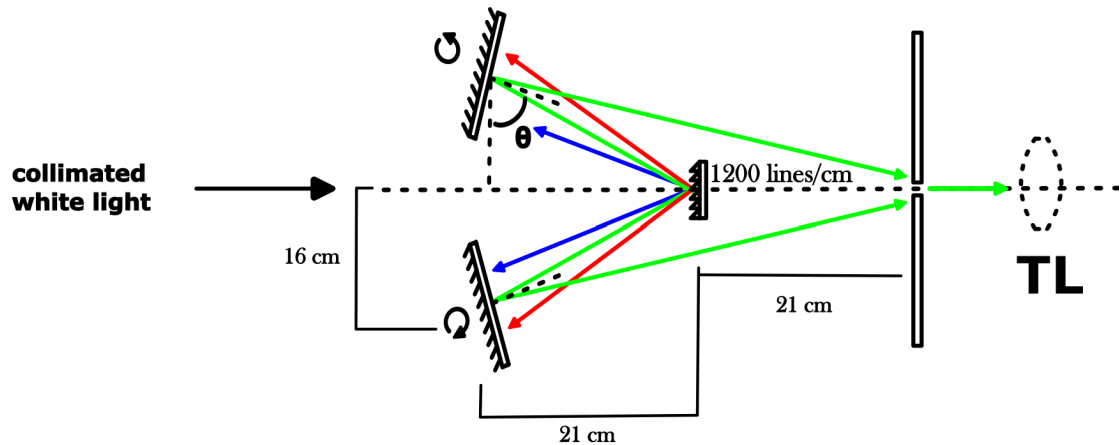


Fig. 29: Monochromator Schematic Ver. 1

Upon the testing of this prototype, we discovered a design problem. With the grating being fixed in place with only the mirrors rotating, the full diffracted spectrum could not be utilized effectively. The width of the spectrum spreading to approximately 4 inches and our mirrors only measuring 1 inch in width, we could not fit our three test wavelengths defined in equation (6).

There are several ways to remedy this problem. Possible solutions include:

- 1) The mirrors could be moved closer to the grating so that they encompass the entire diffracted spectrum. However, this would vastly increase the angle sensitivity required by the servo system controlling the mirrors. The angular resolution needed from the servo would be too fine, and compensating with gear ratios would cause the design to exceed its volume constraints.
- 2) The mirrors could be scaled up in size to encompass the entire diffracted spectrum. This would also push the volume limits of the design. Large mirrors could potentially pose as a point of failure, since they could hit other components or block light as they are rotated.
- 3) The design could be changed so that the spectrum moves rather than the mirror. This would be done by rotating the diffraction grating to change the incident angle of the white light. In this case, the mirrors could stay stationary as long as they are properly aimed at the exit aperture. This design change would simplify the monochromator by requiring fewer moving parts, but it would also mean that only one mirror could be feasibly used. If we aimed our diffraction grating at the mirror for $m = 1$, the wrong wavelength could be reflected by the mirror for $m = -1$ if

any light reaches it at all, and vice versa.

For the long-term flexibility and accuracy of the project, we pursued the third solution. This reduced our number of modes used to one ($m = 1$), which will lead to a slightly lower output power of the monochromator. However, maximizing the monochromator's output power may be re-prioritized due to the sheer magnitude of intensity entering the module from the WLS.

3.73 Prototype 2

The core layout of the monochromator remains the same. The grating will be situated along the optical axis set by the white light beam, and like the previous mirror setup, be rotated by an MG90S servo with an optional gear ratio. By rotating the grating, the diffracted spectrum can fully scan across the mirror even if the spectrum is wider. When only the mirror rotated, the system was limited by its size relative to the width of the spectrum.

Furthermore, by having the mirror stationary, the possibility opens up for a more intensive wavelength selection process in the form of a double monochromator. A double monochromator follows the logic of a regular (single) monochromator, but repeats the wavelength selection process for improved overall accuracy and performance [72]. The spectral bandwidth defined by the first grating and aperture is itself spread further by the second grating, leading to an even narrower bandwidth passing through the second aperture.

With the needed kinematic adjustments and the newly achievable goal of higher-accuracy wavelength selection, our system's monochromator was updated accordingly. Fig. 30 reflects these changes as well as showcases the function of a double monochromator. Note that the direction of the blaze of grating 2 is opposite the direction of grating 1. This is done intentionally to maximize the throughput of the system. The strongest mode $m = 1$ propagates normal to the blazed edge of the grating. By flipping one of our gratings "upside down," we can make full use of the brightest spectra available.

Similar to the first version, the wavelength selection of the module is controlled by a single angle variable. However, by moving the diffraction gratings directly, the incident angle α is under our direct control, so fewer calculations are needed and equation 3 can be used as-is. Since we will be selecting for the same value of λ for both gratings, α (and thus β) is equal across the two. This means that like version 1, this version's moving parts can be synchronized to rotate in equal and opposite directions. Due to the distance between the gratings, they will most likely be controlled by separate servos.

We will be measuring three sample wavelengths in our final product: Hydrogen Red at $\lambda_C = 656.3\text{nm}$, Sodium Yellow at $\lambda_D = 589.3\text{nm}$, and Hydrogen Blue at $\lambda_F = 486.1\text{nm}$. More detail about these wavelength choices can be found in Equation 6. Using an alignment wavelength of 520 nm at normal incidence for $\beta = 38.6^\circ$, the expected angles to achieve our sample wavelengths are:

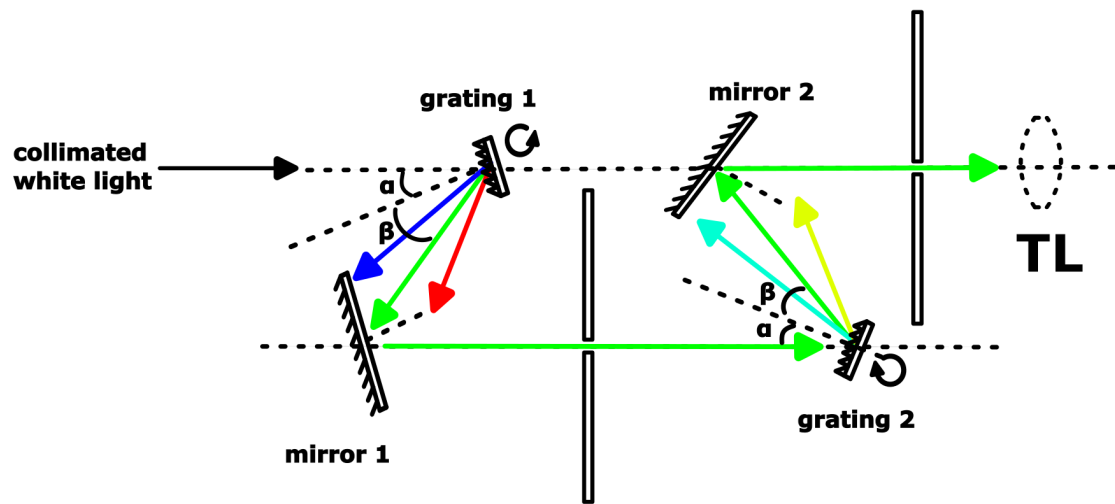


Fig. 30: Monochromator Schematic Ver. 2

This version of the design was tested experimentally according to its schematic, and it has been thus far successful. Like before, a green laser was used for alignment and a broad white light source replaced it for testing. A full list of materials used in this test build can be found in table VII. Since the light source used in the experiment was uncollimated, a simple two-lens system was included along with an aperture to simulate the 10 mm beam that will be used in the final project. A beam block was also added along the main optical axis after grating 1 to prevent light leakage. This will likely not be needed when the proper WLS is integrated since it will have minimal light leakage. Manual rotation stages under the diffraction gratings stood in place of the servo-gear systems for the purposes of this trial. Figs. 31 and 32 show an overview of this prototype and the rotation angles for our chosen wavelengths, respectively.

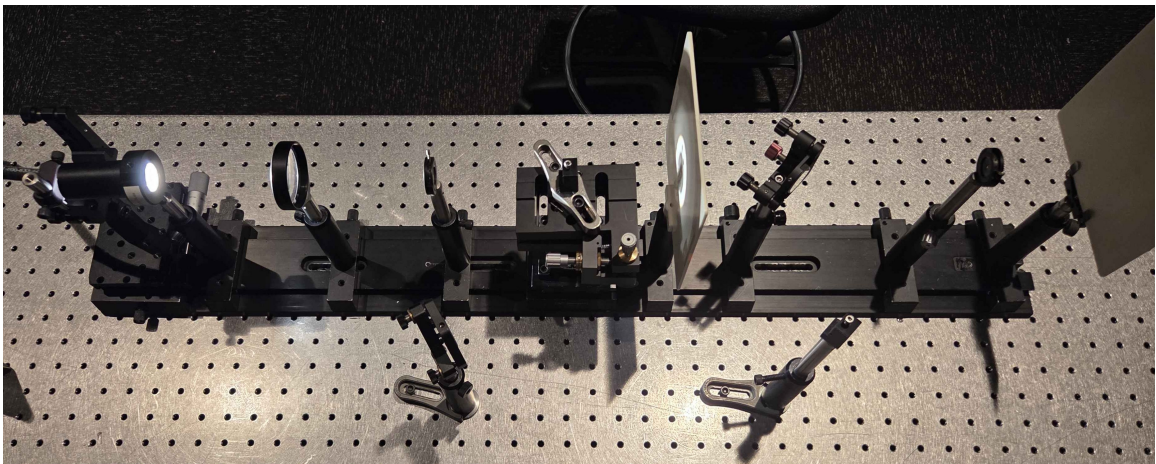


Fig. 31: Monochromator Prototype Ver. 2

The colors seen at the sample values of α roughly match what we would expect the sample

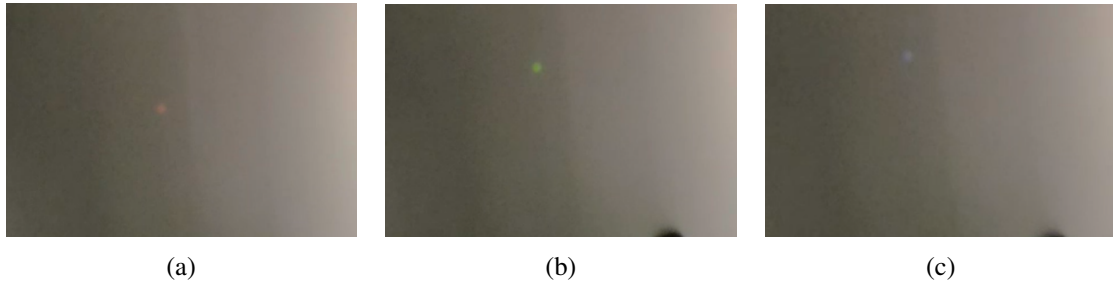


Fig. 32: Resulting colors at (a) $\alpha = -9^\circ$ (b) $\alpha = -5^\circ$ and (c) $\alpha = 2^\circ$

wavelengths to look like. Sources of error in this experiment may include misalignment, light leakage, and low-light camera quality.

3.74 Optical Hardware

While CREOL-owned optics are permitted for use in testing, all components must be returned to the department at the end of this class. In order to avoid disassembling our project and to ensure its future use, we have chosen to buy or make all of our parts ourselves. For optical posts and post holders, we will be 3-D printing open-source components available on Kiva McCracken's website [41]. These models are parametric and can be edited to suit our needs. Exploded views of this hardware are shown in Fig. 33. These parts use a print-in-place technique to encase standard 8/32 and 1/4-20 size nuts in the plastic. By doing this, the components will effectively function like their commercial counterparts. For the arrangement of our optics, we are using standard wooden pencils slotted in the hexagonal holes in the post holders. These pencils will also be supported by custom-printed mounts and right-angle connectors.

The diffraction gratings are housed in a custom-printed casing created in CAD software by Ollie Mueller. This part also features print-in-place standard-size nuts for attachment to the posts mentioned above. To ensure proper alignment, arrows were included on the back of the housing that match with the direction of the blaze. Due to the fragile and expensive nature of the gratings, covers were made alongside the housing to protect them. The covers are able to snap into place using small magnets pressed into craters on both the covers and the grating holder. This ensures that they stay securely attached when we need them to up until the final parts of project development when the entire monochromator is encased in a box. These parts can be seen simulated and printed in Fig 34.

The mirrors in the final monochromator will be simple 1 inch square stickers attached directly to the printed optical posts [28]. These mirrors are abundant and cheap, which is advantageous for experimental testing. Their adhesive back negates the need for mounting. However, slight angle adjustments may be made manually for proper alignment.

The finished monochromator will have three apertures: at the entrance, between mirror 1 and grating 2, and at the exit. The apertures used in the final product will also be fabricated by our group. The apertures will have a fixed diameter of 10 mm to maintain

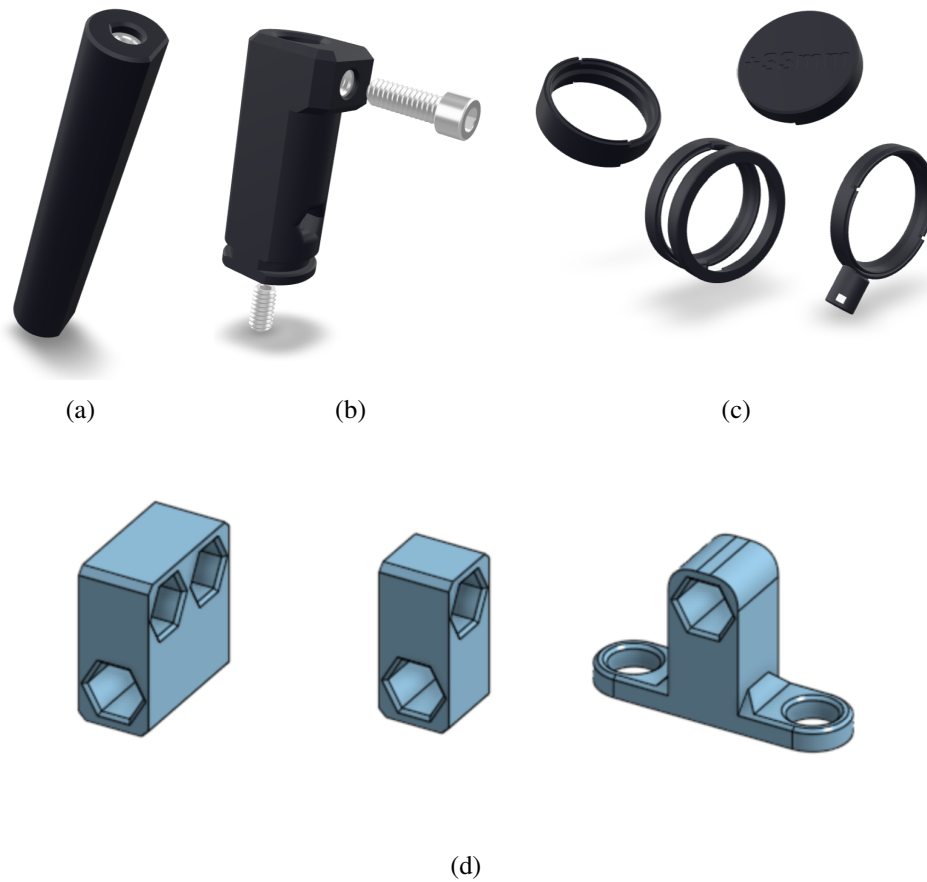


Fig. 33: Exploded-view CAD files for (a) an optical post, (b) an optical post holder, (c) lens mounts, and (d) pencil connectors and rail mounts.

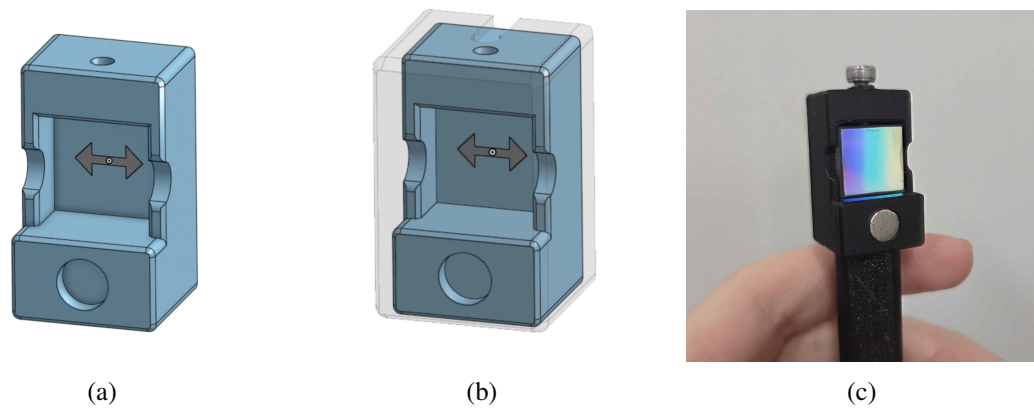


Fig. 34: Normal-view CAD files for (a) the diffraction grating mount, (b) the mount with a transparent cover overlay, and (c) the printed mount in use

the original beam size. They will be cut out of a 1/2 mm copper sheet using a fiber laser cutter. Then, using a similar print-in-place technique as before, the irises will be encased in a plastic ring that will snap into place with the lens mount system shown in Fig. 34c. Various sizes of slit apertures and 10mm circular apertures cut from copper in this way are shown in Fig. 35.

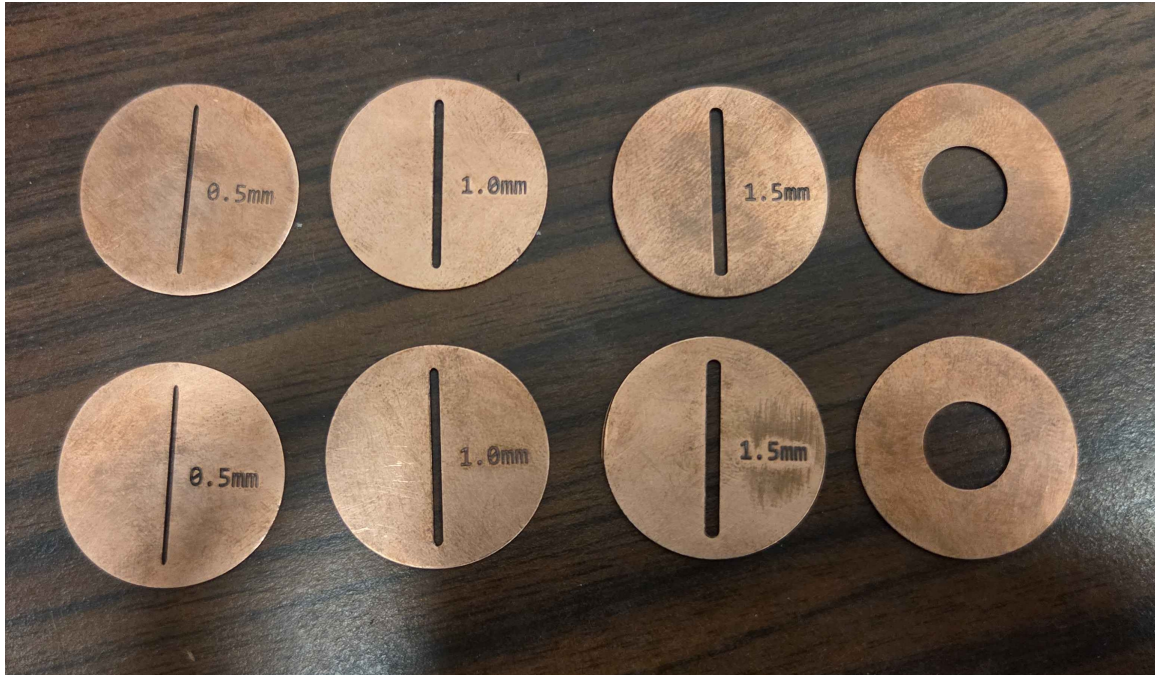


Fig. 35: Laser-cut copper slits and circular apertures. From left to right: 0.5mm slit, 1.0mm slit, 1.5mm slit, 10mm circular aperture.

3.8 Image Acquisition System

3.81 Sensor and Mounting

The image acquisition system aka IAS is the end of the optical axis. All the light has been processed to a wavelength and is now diverging or converging based on the test lens. The IAS consists of two major components the image plane, and the imaging system. The first consists of a mounted sheet of paper which forms a illuminated spot from the back projection of the light cone on the translucent plane. The paper scatters the light and allows the macroscale measurement of spot size via a smaller more conventional imaging system. The image plane and the imaging system are around 250mm in separation.

For the imaging system we selected a Sony IMX477 CMOS sensor. CMOS sensors operate with each pixel having its own amplification and bit output allowing fast data speed but causes higher noise. CCD sensors effectively store the energy and then are read out in sequence allowing tighter packing and higher sensitivity [62]. Our system is relatively high power, and isn't particularly limited by exposure time. Since cost and

integration difficulty is the highest barrier to the project, our team decided to go with CMOS as it has more support for consumers. Specifically we chose the Raspberry Pi 12.3MP High Quality Camera with a 7.9mm diagonal sensor size. This camera sensor is perfect for our project. It is relatively low cost (\$50), has an established user base, and is compatible with custom lens mounting system. Furthermore since we are using a Raspberry Pi 5 for processing it works natively.

The imaging plane and imaging system are kept at a fixed distance of approximately 250mm as they are translated along the gantry. This significantly de-complicates the imaging system since it can use a fixed focal length and thus doesn't require any lens dynamics.

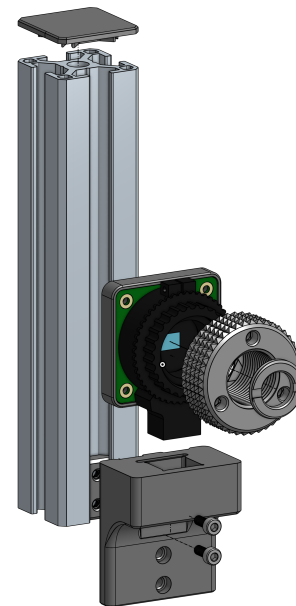
In order to allow futureproof lens assembly design and easy testing a modular screw based system was designed. It integrates with the existing metal components of the HQ camera sensor, allowing the continued use of the focusing mechanism. View this subsystem in fig. 36.

Because each lens is effectively its own screw washer, they can be adjusted by removing their predecessors and spinning until the desired output is achieved.

While this system is not adaptable to large changes in focal distance, our system is set at a fixed focal length and thus shouldn't be affected.



(a) 3D printed and tested system.



(b) Exploded View of lens assembly

Fig. 36: IAS lens assembly mechanical overview

Its within this system that stops will be mounted to prevent stray light and back reflections from muddling the image.

Since our imaging plane (visible in fig. 37) is 200 x 200mm and our image sensor is a height of 4.712mm the system must have a magnification of approximately $\times \frac{1}{50}$ for the full image plane to be in focus.



Fig. 37: Image assembly model, features a sheet of paper sandwiched between laser cut wooden panels. Working image area is 200 x 200mm.

Most lens systems struggle with spherical aberration around the edges of the sensor. While that is certainly a concern, since our image is simple and stationary any aberration can be corrected in software. As well our area of the focus/measurement is small and positioned directly in the center of the image plane and therefore sensor.

Based on back of the napkin math, the lens assembly must have at least 3 lenses and 2 stops.

3.82 Linear Gantry

Our system evaluates the spot size over distance, to do so we must translate the imaging system back and forth. To do so we chose a stepper, belt, and linear extrusion assembly.

Stepper motors are common place and rotate in discrete steps that allow precision movement. We chose these instead of servo motors because we need a large range of motion, approximately 400mm. To couple the systems together we use a belt which is fixed on one side to the imaging carriage, as the stepper rotates the carriage translates back and forth.

One disadvantage of stepper motors are their lack of effective encoding, so the system must be homed upon startup. To do this a limit switch was installed on the far end of the aluminum extrusion from the motor. Upon bootup the system translates the carriage slowly towards the endstop checking every 1mm for a change.

When the limit switch is pressed that is set as the home position. From there a serial input will allow the carriage to translate to the entered distance.

During testing this system was found to have a sub mm accuracy. Which surprising considering that configuration lacked horizontal stabilization. See in order to have the aluminum extrusion carriage slide upon the rail a PTFE sliding component was inserted, however these have loose tolerance and allow back and forth motion. This will be solve in the final product.

We chose aluminum extrusion because it is cheap, strong, and has alot of supporting hardware. The aluminum extrusion is 30 x 30mm T-slot, and comes out to around 18 cents a cm.

This setup is a robust but cheap way to move the carriage and maintain a fixed optical axis. View the system in fig. 38.

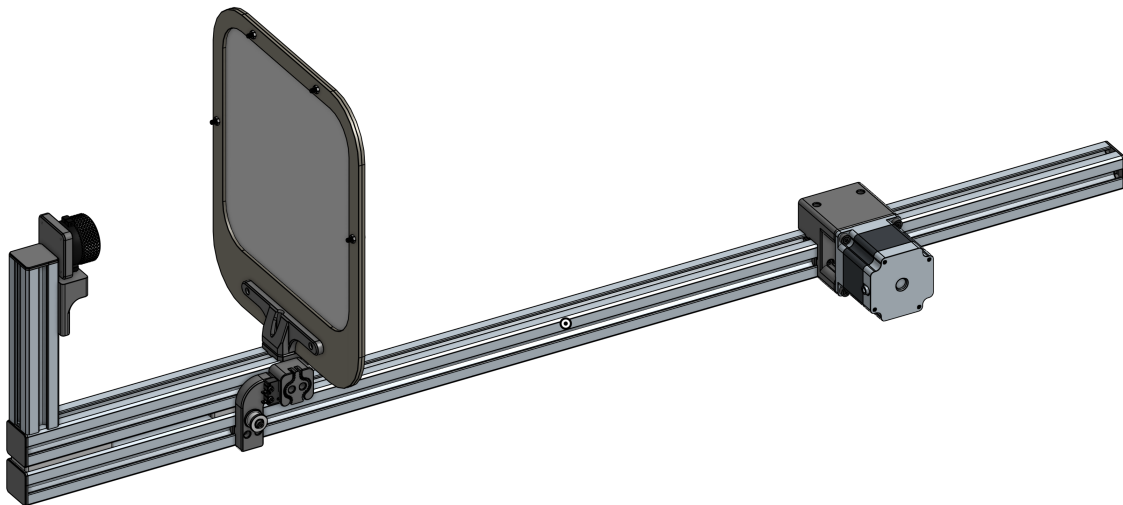


Fig. 38: Linear gantry CAD assembly.

A bill of materials is visible in table VIII.

3.83 Lens Assembly

For the lens assembly and imaging structure we have chosen to use a Cooke Triplet. This is a three element system which is famous for fixing the 7 Seidel aberrations with the fewest components [4].

The removal of these aberrations was achieved through three principles: lens powers sum

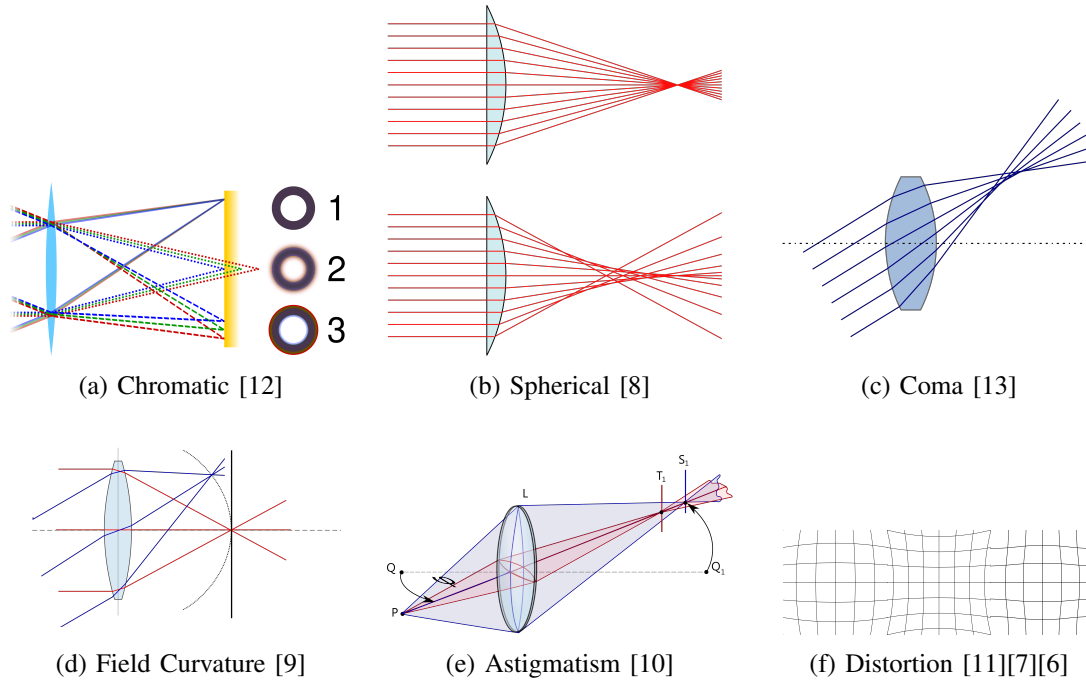


Fig. 39: The 7 Siedel aberrations.

up to zero, positive and negative lenses spaced apart, and symmetrical design. Typical designs also use multiple glass types. The left and right lenses are crown glass, and the middle flint [3].

Crown glass describes materials with an abbe number greater than 55 whereas flint glasses possess an abbe number below 50 [47].

One particularly useful number is the index difference between the crown and flint glass, denoted as Δn_{c-f} . Increasing this difference helps reduce field curvature, distortion, astigmatism, coma, and chromatic aberration.

From there the surfaces and spacings are optimized to reduce spherical aberration and balance the aberration contribution so they negate each other.

3.9 Optical Equations

Our system aims to provide a broad range of lens specifications from as little data as possible. Using spot size, distance, wavelength, intensity, and a series of mechanically captured measurements we can recreate almost the entirety of standard optical datasheets. There are some key exceptions particularly the modulation transfer function (MTF), surface roughness, curvature centering, clear aperture, and other surface abnormalities. However in terms of our system those are relatively irrelevant (with the exception of clear aperture), the user uses the machine to identify lenses, not perform quality control. Any

of the lenses going in the machine are most likely research grade lenses that have been manufactured and verified by a company with higher standards than our scope could hope to verify.

The magic of the system is the monochromator, being able to vary the wavelength allows the extraction of far more data, and particularly allows material identification. Then there is the spot size vs distance measurement that is extracted from the linear gantry and IAS, this gives us our focal length. A set of photodiodes allows the measurement of intensity. Finally an automated probing system measures the lens dimensions, combining these values with the focal length we can derive the index of refraction and front/back focal length. Finally by evaluating index of refraction vs wavelength we can get the abbe number which helps in identifying materials. Lets go ahead and break down each of these processes.

The monochromator outputs a tight range of wavelengths centered about our "measuring wavelength" $\lambda_{\text{measuring}}$. For our purposes we consider this beam fully monochromatic and only regard the measuring wavelength. This is noted down for each focal measurement and/or intensity sub-process.

The measuring wavelength is set by selecting the wavelength and referencing a set of experimentally derived equations relating mirror angle to spectrometer calibrated output peak.

Then there is the mechanical measuring side of things. Initially we considered using white light interferometry to create a 3D model of each surface via a Zygo interferometer setup which compares a spherical reference surface to that of the lens. This was quickly abandoned as the complexity and cost is the reason why Zygo interferometers are valued starting in the 10s of thousands of dollars.

So instead we use a robotized form of the cheapest, electronic, and most plentiful measuring device in any scientific laboratory – calipers.

To define our lens geometrically we need 5 main measurements.

- Front Radius of Curvature (mm)
- Back Radius of Curvature (mm)
- Front Thickness (mm)
- Back Thickness (mm)
- Diameter (mm)

Its important to note that these measurements require the lens to be at least somewhat centered in the holder. So the mounting system we adopt needs to enforce that somehow or communicate the requirement to the user.

A geneva gauge measures the radius of curvature of a object by comparing two fixed pins to a variable length coupled pin. The variable pin measures the sagitta. By jogging a caliper towards a lens and rotating the lens to present the surface we want to measure we

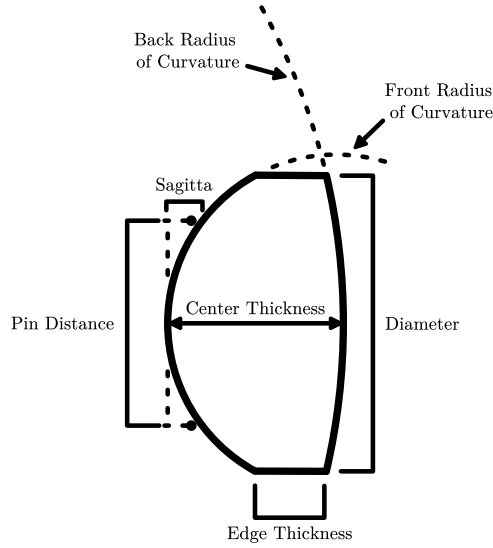


Fig. 40: Lens geometric diagram.

can extract the radii of curvature and the diameter from the relatively stopping length and the center thickness by using the caliper as a virtual limit switch on the linear gantries axis. Check equation 5 to see how radius is calculated from the sagitta and pin distance.

$$\text{Radius of Curvature} = \frac{\text{Sagitta}^2 + \frac{1}{4} (\text{Pin Separation})^2}{2 \cdot \text{Sagitta}} \quad (5)$$

Abbe number is a variable describing the chromatic aberration of a material. Chromatic aberration is the effect that wavelength has on effective index of refraction. This causes a spreading of the light and that classic blue-green-red edge effect. The number is different for every material and is especially important when designing optical assemblies. Thus when we choose our wavelengths to test focal length and therefore index of refraction we adhere to the abbe number formula (6) [70].

$$\nu = \frac{n_D - 1}{n_F - n_C} \begin{cases} n_D & \text{Sodium Yellow} & 589.3\text{nm} \\ n_F & \text{Hydrogen Blue} & 486.1\text{nm} \\ n_C & \text{Hydrogen Red} & 656.3\text{nm} \end{cases} \quad (6)$$

Focal length is tested at minimum at these three wavelengths. To do so we feed a collimated monochromatic beam into our test lens. Under these conditions we can expect a simple diverging or converging cone to form, whose radius can be described by the formula (7).

$$y = r_{\text{beam}} \cdot \left| \frac{x - f}{f} \right| \quad (7)$$

There are of course more complex optical phenomena at play, a single lens can only

focus to a spot as small as the source. In our case this means we want to optimize the monochromators outgoing beam to have as small a radius as possible while still being able to make measurements with our camera. However we can effectively ignore this for the sake of our math. By applying a fit of equation 7 to our data, we can extract the x-intercept which represents the focal length from the test lenses principle plane since the fit essentially ignores the "beam waist" since its symmetric about the focal point. This was tested in some of the worst conditions: poorly collimated beam, off axis camera, lit room, manual distance measurement, etc... Yet for lens of 150mm focal length we achieved a measured focal length of 156mm, thats 4% error. Our spec calls for within 10mm, that was already exceeded. Check out fig. 41 to see the data and fit.

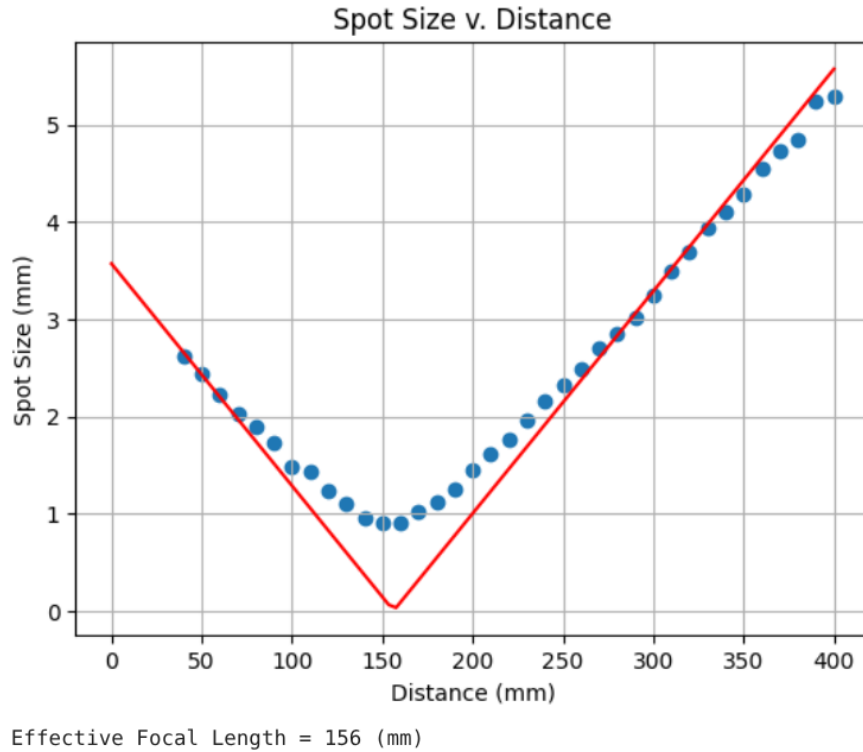


Fig. 41: Experimental verification of focal length extraction method.

In order to get this radius we use python's opencv library and apply a contour pass and then fit circles to the largest contour, which block all other light is the beam spot. Initially we tried use full-width-half-max which is a standard descriptor for optical beams. This however yielded unruly results at smaller spot sizes, so instead we tested using a threshold of 0.2, this worked far better but the value may change in the future. A representation of this process using a mathematically generated profile is visible in fig. 42.

From our geometric values and focal length we can work backwards to derive index of refraction. For that we use the thick lens equation (8) [37].

$$P = \frac{1}{f} = (n - 1) \left[\frac{1}{R_1} - \frac{1}{R_2} + \frac{d(n - 1)}{nR_1R_2} \right] \quad (8)$$

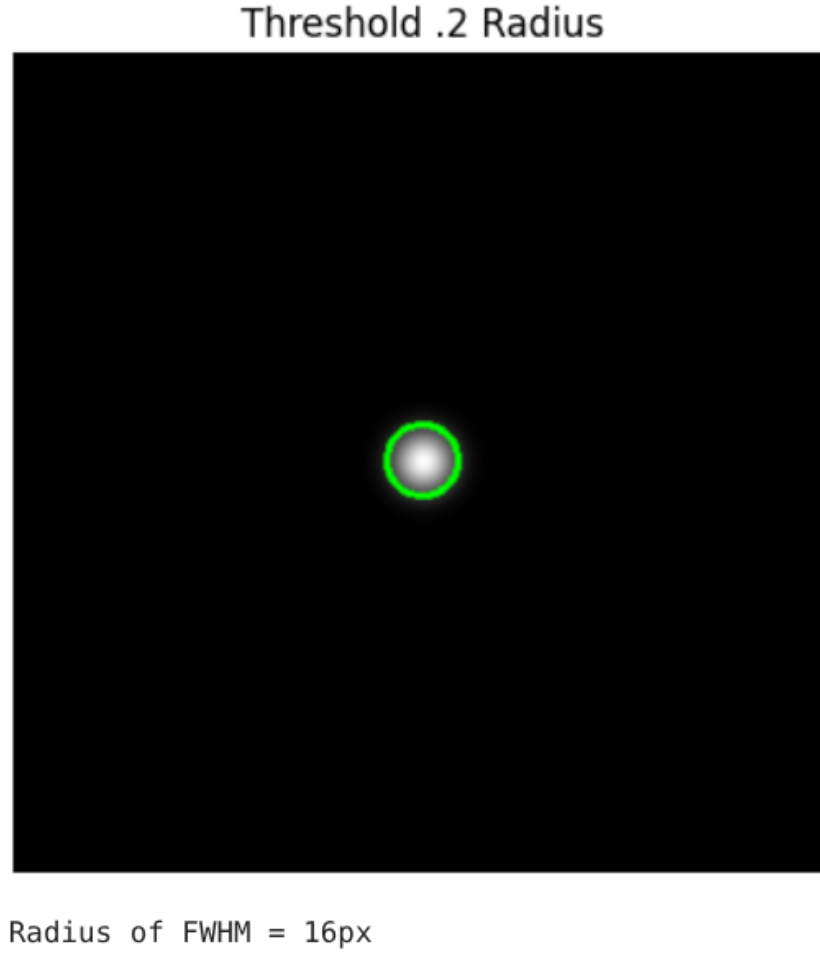


Fig. 42: Threshold based radius extraction of generated gaussian profile.

There is a key issue, index of refraction n in this case is squared when evaluating the equation. As a result solving it creates a monster of a equation with a negative and positive solution. So the fix? Absolute value both solutions and take the bigger one. It is patchwork but works for all the lens combinations we have tested. Check out the monstrosity in equation (9).

$$n = \frac{fR_2 - fR_1 + 2fd + R_1R_2 \pm \sqrt{f^2R_1^2 + f^2R_2^2 - 2f^2R_1R_2 + 2fR_1R_2^2 - 2fR_1^2R_2 + 4fR_1R_2d + R_1^2R_2^2}}{2(fR_2 - fR_1 + fd)}, f \neq 0 \quad (9)$$

Finally the reflection, transmission, and absorption spectra are calculated. In order to do so we place photodiodes within the chamber and have the monochromator do a sweep through all the working wavelengths. We normalize each of the reflection and transmission graphs and then add them on top of each other and renormalize. Finally we fill in those blanks with absorbance. This allows us to extract all the data without worrying about incident power as that will differ based on the lens geometry. Check out 43 to see the

relationships of different equations.

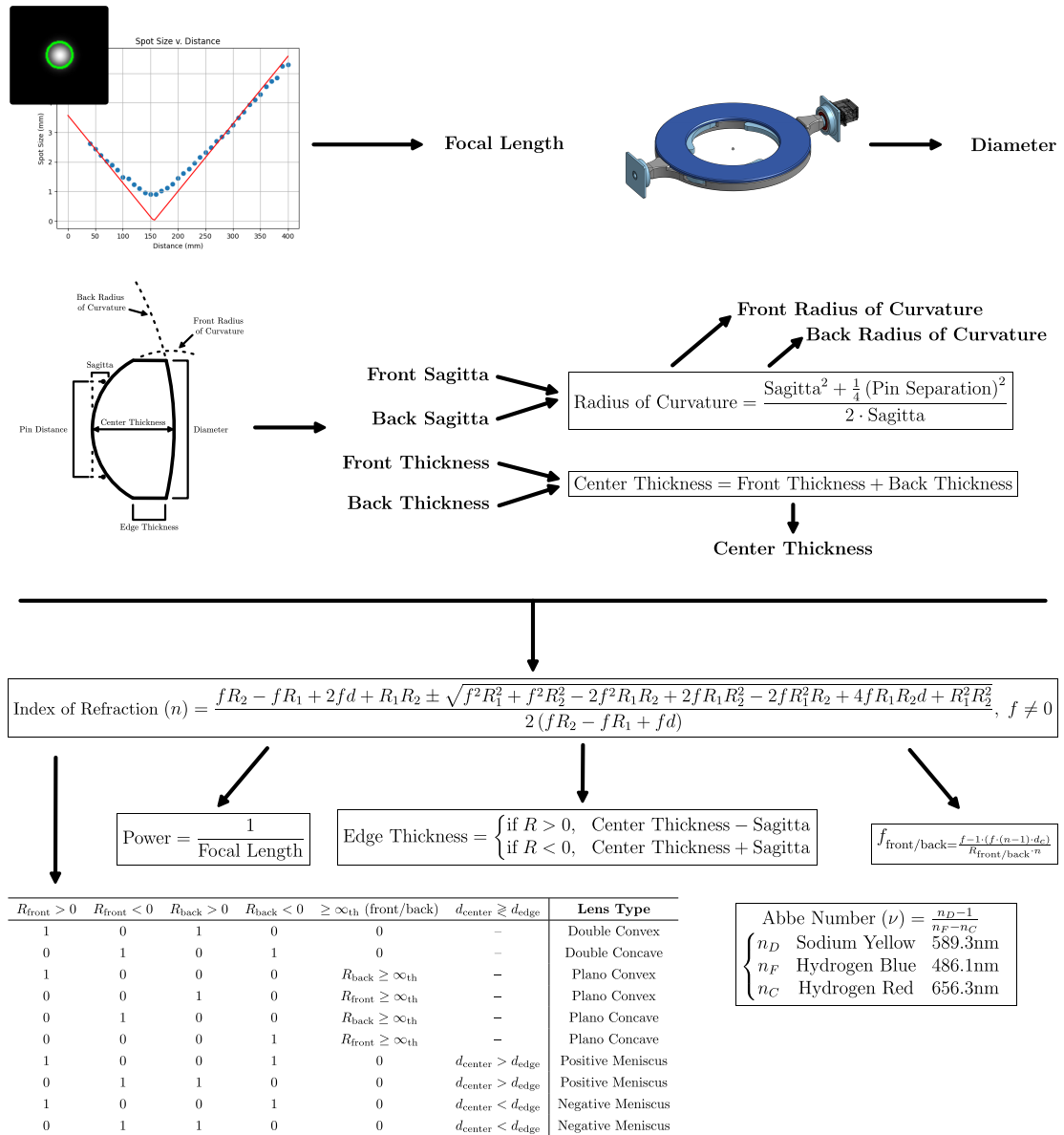


Fig. 43: Optical calculation relationship map.

4 Chapter 4 Standards and Design Constraints

4.1 Communication Protocols

4.11 Introduction

The proposed ALM implementation is designed around a distributed architecture in which multiple low-complexity microcontrollers are managed and coordinated by a central Raspberry Pi unit. In order to establish reliable communication and data exchange between the Raspberry Pi and the subordinate microcontrollers, appropriate serial communication protocols must be employed. For this implementation, three widely adopted standards, I²C (Inter-Integrated Circuit), SPI (Serial Peripheral Interface), and UART (Universal Asynchronous Receiver/Transmitter), will be investigated. These protocols provide a robust and efficient means of handling control signals, configuration commands, and data transmission, thereby ensuring proper synchronization and integration of all system components.

4.12 I²C

I²C, shown in Fig. 9, is a synchronous, multi-master, multi-slave, serial communication protocol originally developed by Philips. It operates using only two bidirectional lines: the Serial Data Line (SDA) and the Serial Clock Line (SCL). Communication is initiated by a master device, which generates the clock signal and addresses individual slave devices via unique 7-bit or 10-bit addressing. Data transfer is organized in frames, each consisting of an address phase, read/write control bit, acknowledgment bit, and one or more data bytes. Pull-up resistors are required on both SDA and SCL lines to maintain signal integrity, and bus speeds commonly range from 100 kHz (Standard Mode) up to 3.4 MHz (High-Speed Mode). I²C is particularly well-suited for systems requiring multiple peripherals on a shared bus with minimal pin usage.

a) Rationale for Using I²C

In the Automated Lens Parameter Metrology system, multiple STM32 boards are expected to coordinate the operation of various actuators and sensors, including stepper motors, light sources, and position sensors. The Raspberry Pi must act as a central controller that periodically polls each STM32 for status updates or sends operational commands such as tray positioning or lens rotation. I²C provides an efficient solution to this coordination problem by supporting multi-slave communication using unique device addresses on a single shared bus. The protocol's hardware-level acknowledgment mechanism ensures reliable data integrity, which is essential for time-sensitive measurements and mechanical synchronization.

4.13 UART

UART, shown in Fig. 11, is a widely used asynchronous serial communication protocol that enables full-duplex, point-to-point data transfer between two devices. Unlike synchronous protocols, UART does not rely on a shared clock line; instead, both devices must be pre-configured to operate at the same baud rate. Data is framed with a start bit, a programmable number of data bits (typically 8), an optional parity bit for error detection, and one or more stop bits to signal the end of transmission. Communication occurs over two unidirectional lines—TX (transmit) and RX (receive)—and can be extended with hardware flow control lines (RTS/CTS) when required. UART is valued for its simplicity, low overhead, and ability to establish reliable communication over short to

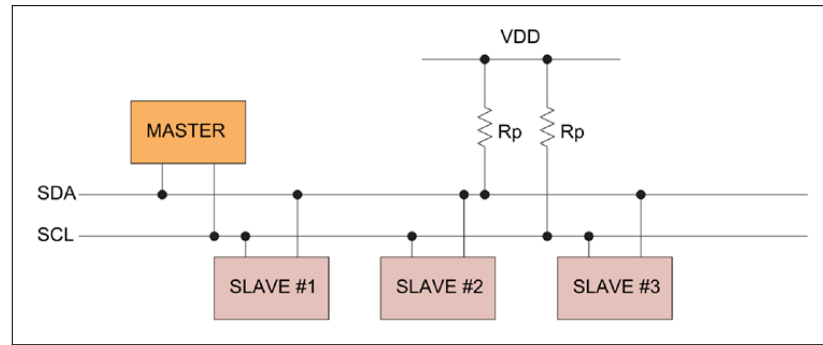


Fig. 44: I2C Implementation[1]

moderate distances, making it ideal for direct device-to-device links.

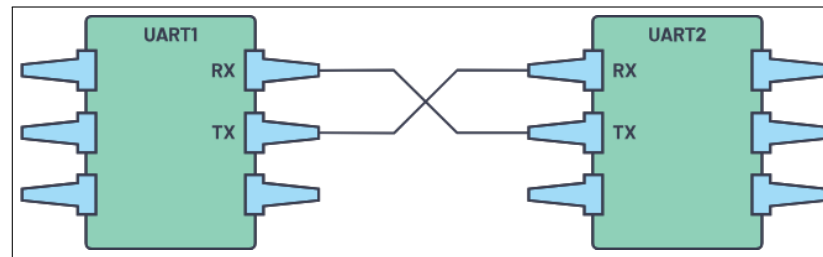


Fig. 45: UART Implementation[48]

4.14 SPI

The Serial Peripheral Interface (SPI) is a synchronous, full-duplex bus that connects a single “main” (master) device to one or more “subnodes” (slaves) and transmits and receives data simultaneously, synchronized by a clock line generated by the main. In the widely used four-wire configuration, the signals are: SCLK (clock from main), CS (chip-select from main), MOSI (“main out, subnode in”), and MISO (“main in, subnode out”). The main selects the subnode by asserting CS (typically active low), then sends the clock while data is shifted out on MOSI and simultaneously shifted in on MISO; the sampling and shifting edges are determined by the selected clock polarity and phase (CPOL/CPHA) which define the four SPI modes. When multiple subnodes are used, each typically needs its own CS line (“regular mode”) unless daisy-chain configuration is supported, in which the subnodes pass data sequentially through a chain—with the trade-off that the number of clock cycles increases with subnode position.

There are two primary optical standards that must be met for this project, a 10mm collimated beam exiting the WLS assembly, and a 10nm FWHM collimated beam exiting the monochromator system.

These are the two optical specifications that our group has chosen for this project.

In order to evaluate spec compliance a lab calibrated spectrometer will be used to verify the output full width half max of the monochromator. In order to lower the FWHM the

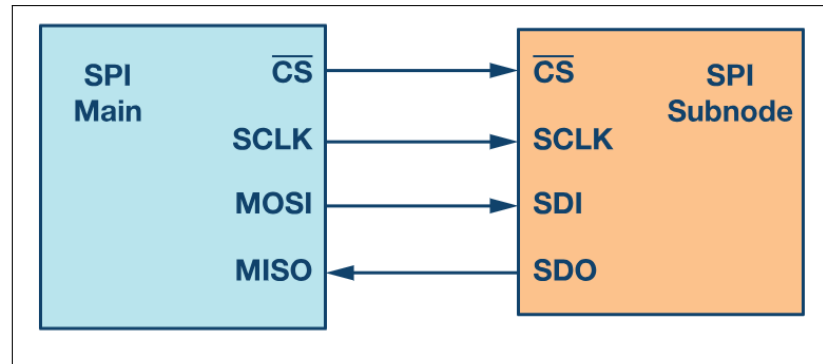


Fig. 46: SPI Implementation[14]

entrance slit can be decreased to try and gate out wavelengths.

As for the white light source the beam can be verified by aligning twin 10mm irises over a length of 2 meter. If the beam stays within both irises then it should function within the systems total optical path length.

5 Chapter 5 Application of ChatGPT and Other Similar Platforms

5.1 Overview of Compared Platforms

5.11 ChatGPT (OpenAI)

ChatGPT, developed by OpenAI, represents one of the most mature conversational AI systems, widely deployed through both web and API interfaces. It is powered by OpenAI's GPT model family, with the GPT-4 and GPT-5 architectures offering multimodal capabilities, strong reasoning, and broad instruction-following performance. OpenAI's alignment techniques, such as reinforcement learning from human feedback (RLHF) and iterative model refinement, have positioned ChatGPT as a general-purpose assistant adaptable to a wide range of domains. However, its closed-source nature limits research transparency and reproducibility [45].

5.12 Claude (Anthropic)

Anthropic's Claude models emphasize constitutional AI, a methodology that encodes alignment principles directly into the model's behavior rather than relying solely on human feedback. This approach enhances consistency and ethical adherence in dialogue generation [55]. Comparative evaluations suggest that Claude exhibits strengths in interpretability and safety but may underperform slightly in mathematical reasoning and complex code synthesis tasks relative to ChatGPT and Gemini [17]. Its extended

context length, reaching up to hundreds of thousands of tokens, enables Claude to handle document analysis and long-form reasoning more effectively than some peers.

5.13 Gemini (Google DeepMind)

Gemini, developed by Google DeepMind, integrates text, image, and code reasoning into a unified multimodal framework. It leverages Google's extensive search infrastructure and proprietary datasets for retrieval-augmented generation (RAG), giving it an advantage in factual recall and web-based information synthesis [45]. Gemini's ecosystem integration with Google Workspace and Android provides practical deployment benefits. However, similar to OpenAI's models, Gemini remains closed-source, and detailed architectural information has not been fully disclosed, restricting independent validation of benchmark claims.

5.14 Copilot (Microsoft)

GitHub Copilot, powered by models co-developed with OpenAI, is specialized for code generation and software development. Its tight integration with Visual Studio Code and GitHub repositories makes it a dominant tool for assisted programming. Copilot's performance is strongest in pattern-based code completion and documentation generation, though it remains limited by context window constraints and domain-specific reasoning outside programming tasks [55]. Moreover, its reliance on OpenAI's backend ties it closely to the ChatGPT model family, making its differentiation more functional than architectural.

5.15 Grok (xAI)

Grok, designed by xAI under Elon Musk's leadership, is positioned as a socially integrated, humor-tolerant assistant connected to the X (formerly Twitter) platform. While benchmark data for Grok remains limited, its distinguishing feature lies in its conversational tone and integration with real-time social data streams. Grok's ability to access current information contrasts with models that operate on static datasets. However, its technical transparency and reproducibility are minimal, and early user reports suggest performance variance in factual reasoning and professional applications [45].

5.16 DeepSeek (DeepSeek AI)

DeepSeek represents one of the most capable open-source entrants to the LLM ecosystem, emphasizing efficiency, transparency, and multilingual proficiency. Its architecture and training data have been partially published, allowing academic evaluation and fine-tuning for research. Despite not matching the peak benchmark scores of proprietary models like GPT-5 or Gemini Ultra, DeepSeek offers strong reasoning performance relative to its computational footprint [17]. This efficiency-oriented design demonstrates that high-quality AI assistance can be achieved without the massive compute resources typical of frontier models.

5.17 Strengths and Weaknesses of Each Platform

Table IX summarizes the key comparative attributes of each AI platform based on reported data, observed usage, and ecosystem characteristics.

5.18 Discussion of Comparative Trends

When comparing these platforms, three trends emerge. First, **performance parity** among leading proprietary models (ChatGPT, Gemini, Claude) indicates a convergence in large-scale LLM capability. Benchmark differences are increasingly narrow, suggesting that innovation now centers more on **alignment, transparency, and integration** rather than raw accuracy [46].

Second, **ecosystem integration** plays a growing role in model adoption. ChatGPT's API ecosystem and plug-in framework mirror Google's integration of Gemini within Workspace, while Microsoft's Copilot suite extends LLM access directly into professional workflows. By contrast, open-source models such as DeepSeek appeal to academic and research communities prioritizing control and customizability over convenience.

Third, **data governance and transparency** remain significant differentiators. OpenAI and Google continue to restrict training corpus disclosures, citing intellectual property and safety considerations, while DeepSeek's partial openness promotes replicability but at the expense of peak performance optimization. Claude's constitutional AI method stands out as a distinctive philosophical and technical approach to model alignment, offering an alternative to reinforcement-based feedback loops [55].

5.19 Summary and Outlook

In summary, while ChatGPT remains a benchmark for general reasoning and usability, competing models have introduced complementary advantages. Gemini's factual grounding and Claude's alignment-driven reasoning reflect diversification in model priorities, while DeepSeek illustrates how open approaches can compete effectively under constrained compute budgets. Copilot and Grok represent domain-specific and socially embedded deployments, respectively, showing that the next frontier in AI performance may depend less on model size and more on **contextual integration and user-centered design**.

5.2 Pros and Cons of AI in Senior Design

5.3 ChatGPT Case Studies

5.31 Case Study 1

Please refer to Appendix A, as well as Sections A.11, A.12, A.13, A.14 for the Case Study 1 prompt and outcomes generated by ChatGPT, Copilot, and Claude.

In this case study, the performance of three AI platforms—ChatGPT, Claude, and Copilot—was evaluated based on their responses to a prompt comparing three microcontrollers: the ATmega32, ESP32, and STM32. Although each model produced a response that was usable to some degree, their depth, clarity, and usefulness varied significantly, providing an interesting look into how differently these systems handle technical material.

ChatGPT offered the most comprehensive and thoughtful analysis of the three. Its response demonstrated a clear understanding not only of the microcontrollers themselves, but also of the broader engineering context in which these devices are typically used. Instead of merely listing characteristics, it explained why certain features matter and how they affect practical decision-making. ChatGPT broke down the strengths and limitations of each MCU in detail and framed them within realistic use cases. It did so in a structured way that showed clear organization and intent, even providing a table to summarize key attributes across the devices. The level of nuance in ChatGPT's writing gave the impression that it understood the trade-offs an engineer would consider when selecting a microcontroller for a project. This made the response both informative and highly applicable, as if it were written with an awareness of real embedded systems work rather than only surface-level specifications.

Copilot produced a noticeably shorter and more streamlined answer. It still managed to identify the general advantages and disadvantages of each microcontroller and, like ChatGPT, attempted to match each device to an appropriate type of project. However, the level of detail was more limited, and the analysis was not as deeply developed. Copilot's response tended to emphasize quick summaries rather than comprehensive explanations. While the information it provided was correct, it did not explore the nuances that would affect engineering decisions in practice. The writing style felt more like a concise technical note than a full comparison. Even so, Copilot was still helpful; the clarity and brevity of the response made it easy to understand, and it could be valuable in cases where a user wants a broad overview without extensive context.

Claude's response, although readable and organized, was the least informative for this task. It provided a basic pros and cons list for each microcontroller, but the analysis was noticeably shallower than the others. Claude's descriptions were more generic and did not establish meaningful distinctions between the devices beyond their most obvious characteristics. Unlike ChatGPT and Copilot, which both attempted to connect each MCU's traits to realistic project types, Claude remained closer to surface-level summarization. As a result, its answer felt more like a restatement of general knowledge rather than a targeted comparison aimed at helping a user make a decision. The absence of deeper commentary, numerical context, or comparative reasoning limited the usefulness of the response, especially for someone seeking technical guidance.

Taken together, these results highlight clear differences in how each AI platform approaches technical analysis. ChatGPT demonstrated the strongest ability to synthesize detailed engineering information into a coherent and practical explanation. Copilot provided a functional but more constrained overview, suitable for quick reference rather than in-depth study. Claude, while clear in its writing, did not offer the depth or comparative insight needed for a meaningful technical evaluation. This case study illustrates that even

when the same prompt is given, AI systems vary widely in analytical depth, technical fluency, and their ability to support engineering decision-making.

5.32 Case Study 2

Please refer to Appendix A, as well as Sections A.21, A.22, A.23, A.24 for the Case Study 2 prompt and outcomes generated by ChatGPT, Gemini, and Grok.

In this case study, the same prompt was given to three different AI systems—ChatGPT, Gemini, and Grok—to assess how well each model could explain the process of designing a monochromator. Although the prompt is relatively specialized, involving both optical engineering considerations and practical instrument design, each system approached the problem in noticeably different ways. The resulting responses varied not only in detail but also in structure, accuracy, and the degree to which they demonstrated an understanding of how monochromators are actually designed in laboratory or industrial contexts.

ChatGPT produced the most organized and technically thorough response of the three. Its explanation demonstrated a clear awareness of the multiple stages involved in building a monochromator, covering both the theoretical and practical aspects of the design. ChatGPT outlined the optical principles underlying monochromator function, including wavelength selection, slit geometry, diffraction grating properties, focal lengths, and stray-light reduction. What made its answer especially useful was the way it connected foundational concepts to the step-by-step workflow that an engineer would realistically follow: determining the spectral range of interest, selecting the appropriate grating line density, calculating dispersion and resolution requirements, choosing the geometry of the optical bench, and then considering the mechanical elements that ensure alignment and stability. The response made it clear that ChatGPT understood that designing a monochromator is a balanced process involving optical physics, mechanical precision, and performance trade-offs. While the explanation remained accessible, it did not sacrifice technical accuracy, and the writing felt cohesive, purposeful, and directly relevant to someone preparing to design or evaluate such an instrument.

Gemini's response, while still useful, was more general and less developed than ChatGPT's. Its explanation covered many of the same broad themes—such as the importance of slit sizes, diffraction gratings, and wavelength selection—but the treatment was lighter and did not demonstrate the same level of engineering reasoning. Gemini tended to frame the design process as a sequence of conceptual considerations rather than a concrete workflow. For example, it mentioned selecting appropriate optical components but did not explain how one would choose grating densities based on spectral resolution requirements or how focal length affects linear dispersion. It also touched on alignment and calibration but did not elaborate on the tolerance levels or mechanical challenges involved. This gave the impression that Gemini's response was more of an introductory overview rather than a guide that someone could use to begin an actual monochromator design. The writing was clear and logically structured, but the absence of deeper detail made the explanation feel more like a summary than a technical analysis. In academic or engineering settings, such an answer may be helpful to readers new to the subject but would likely require supplementary information to be truly actionable.

Grok provided the least technically rigorous response of the three systems. Its explanation was more conversational and at times felt less concerned with engineering principles than with presenting an accessible, high-level description. While Grok did mention elements like gratings and optical pathways, its treatment was notably shallow. Important details that would matter in real monochromator design—such as spectral bandwidth, mechanical tolerances, aberration control, grating blaze angles, or stray-light suppression—were either glossed over or omitted entirely. In several cases, Grok seemed to focus more on describing what a monochromator generally does rather than how one would intentionally design the system from scratch. As a result, the response lacked the type of depth that would be expected in a technical or academic context. The writing style was friendly and readable, but the content did not reflect the complexity or precision involved in designing optical instrumentation, making Grok’s answer the least suited for detailed engineering work.

Across the three AI platforms, these differences reveal varying levels of technical fluency and practical reasoning. ChatGPT demonstrated the strongest grasp of monochromator design by articulating the process in a way that aligns with established optical engineering workflows. Gemini offered a clear but simplified overview that captured foundational ideas without exploring them deeply. Grok, while approachable, did not provide enough detail for someone seeking meaningful guidance on how monochromators are designed in practice. Together, these responses highlight how the same prompt can elicit different degrees of analytical depth depending on the model, with ChatGPT showing the greatest ability to handle specialized engineering topics in a technically complete and structured manner.

5.33 Case Study 3

Please refer to Appendix A, as well as Sections A.31, A.32, A.34, A.33 for the Case Study 3 prompt and outcomes generated by ChatGPT, Gemini, and DeepSeek.

In this case study, three AI platforms—ChatGPT, Gemini, and DeepSeek—were evaluated on their ability to answer a technical prompt concerning embedded communication architectures. The prompt asked for the most efficient method of allowing multiple STM32 microcontrollers to communicate with a Raspberry Pi 5, with the relevant data coming from sensors and stepper motors. While all three systems produced useful responses, the depth, reasoning, and practical engineering insight they demonstrated varied considerably, revealing meaningful differences in how each model handles complex hardware-integration questions.

ChatGPT provided the most detailed and engineering-minded response. Instead of simply listing communication protocols, it approached the problem by analyzing electrical noise concerns, bus topology, cable length limitations, and the challenges that stepper-motor environments introduce. ChatGPT supported its argument with comparisons of RS-485, I²C, SPI, UART, CAN, and Ethernet, explaining how each protocol behaves in multi-drop or electrically noisy systems. What stood out most was how ChatGPT aligned its recommendations with real embedded-systems constraints: it emphasized that I²C is fragile in longer-distance or noisy contexts, that SPI becomes unwieldy when extended to

more than one slave, and that plain UART lacks multi-node capability without adaptation. Its recommendation of RS-485 as the best overall solution was supported by practical engineering reasons such as noise immunity, cable-length scalability, low CPU overhead on both sides, and the ease of implementing Modbus RTU or custom UART-framed protocols. Additionally, ChatGPT highlighted CAN and Ethernet as higher-level alternatives depending on bandwidth or robustness needs, and it even supplied architecture diagrams and hardware suggestions, demonstrating an understanding that communication design depends on both protocol theory and real wiring constraints. The response felt comprehensive, precise, and aligned with the type of reasoning a systems engineer would apply when designing multi-node embedded networks.

Gemini also offered a technically grounded answer, but its treatment was more general and less deeply analytical. It ranked RS-485, CAN, and Ethernet as the primary communication options and provided reasonable explanations for each choice. Gemini did acknowledge factors such as noise immunity, physical distance, implementation complexity, and multi-drop support, and it described typical hardware configurations like UART-to-RS-485 transceivers or CAN transceiver chips. However, compared to ChatGPT, Gemini's discussion stayed closer to a conceptual summary than an engineering-oriented analysis. For instance, it did not explore failure modes, trade-offs between half-duplex and full-duplex communication, or the specific timing issues introduced by stepper-motor noise. While it mentioned that Modbus RTU is often used in RS-485 networks and that CAN offers deterministic arbitration, the explanation lacked the deeper context that would help a designer choose between these based on real-world constraints such as polling latency, protocol overhead, or cable routing in motor-heavy environments. The style was clear and well organized, but the response read more like an overview suitable for someone beginning to explore communication protocols rather than guidance tailored to complex embedded system design. Even so, the information was accurate and approachable, and Gemini's prioritization of RS-485 as the most balanced solution aligned with conventional best practices.

DeepSeek produced a response that was noticeably more forceful in its claims but less balanced in its engineering reasoning. Unlike ChatGPT and Gemini, which both recommended RS-485 as the best overall option for this type of system, DeepSeek insisted that CAN Bus was the top choice. While CAN is unquestionably robust and widely used in automotive and industrial environments, DeepSeek's explanation placed heavy emphasis on its determinism, arbitration, noise immunity, and ability to support large numbers of nodes, but it did not fully consider the practical challenges of implementing CAN on a Raspberry Pi 5, which lacks a native CAN controller. As a result, the response somewhat oversimplified the hardware requirements by focusing on CAN's advantages without exploring the costs or trade-offs of adding external controllers, adapting firmware, or managing the complexity of CAN protocols. DeepSeek did mention RS-485 as a strong alternative and described its strengths fairly, yet the comparison felt narrower in scope. The system tended to present each protocol with a list-like set of advantages rather than integrating the information into a cohesive analysis that connected back to the actual use case. As a result, the recommendation of CAN Bus, while defensible in certain contexts, felt less grounded in the constraints of the original problem, especially regarding implementation simplicity and common architecture choices for STM32–Raspberry Pi

networks.

The stylistic differences among the three models were notable as well. ChatGPT's writing conveyed a structured, engineering-oriented mindset, incorporating comparisons, tabular reasoning, and architecture sketches. Gemini's response, by contrast, favored clarity and accessibility, choosing to explain concepts at a higher level without diving deeply into implementation trade-offs. DeepSeek presented its ideas in a way that highlighted strengths energetically but at times lacked the balance necessary to judge which protocol best fits the specified application. Across the three, ChatGPT provided the most nuanced understanding of the interplay between electrical noise, bus topology, bandwidth needs, and hardware complexity. Gemini offered a solid but less detailed overview, while DeepSeek delivered a confident but somewhat oversimplified account of the communication options involved.

This case study illustrates the diversity of approaches that modern AI systems take when confronted with a moderately complex embedded-systems question. Although all three models addressed the major protocols involved, their responses differed in technical depth, practical experience, and their ability to reason through engineering constraints in a realistic way. The differences are useful for understanding how each system supports technical design work and where each one may excel or fall short depending on the user's needs.

5.34 Case Study 4

Please refer to Appendix A, as well as Sections A.41, A.42, and A.43 for the Case Study 3 prompt and outcomes generated by Gemini and DeepSeek.

In this final case study, the task was to compare the performance of Gemini and DeepSeek in responding to a prompt about selecting the most appropriate device for converting an optical signal into an electrical one. The prompt asked whether a photoresistor, photodiode, or phototransistor would be the best choice. Although the topic is not extremely complex, it does require the models to understand the relationships between speed, sensitivity, linearity, and output-signal characteristics across different types of optical sensors. Both systems produced technically accurate descriptions, but the clarity, structure, and depth of their reasoning differed in noteworthy ways.

DeepSeek produced a response that was strikingly thorough, structured, and confident. It delivered a complete comparison of the three components, beginning with a concise summary and then transitioning into detailed explanations of how each device works. The response demonstrated a clear understanding of the underlying physics, mentioning concepts such as reverse biasing in photodiodes, transistor gain in phototransistors, and the slow resistance-driven behavior of photoresistors. One of the defining features of DeepSeek's answer was its emphasis on practical design considerations, such as the need for a transimpedance amplifier when using a photodiode, or the limitations introduced by the junction capacitance of phototransistors. It also addressed topics like spectral accuracy, saturation behavior, noise susceptibility, and even regulatory concerns such as cadmium use in light-dependent resistors. In doing so, DeepSeek displayed an almost tutorial-like

approach, effectively guiding the reader from basic concepts to real engineering decisions.

The style of DeepSeek's response was methodical, and while highly structured, it still felt accessible because it always tied technical characteristics back to real applications. For instance, it explained why photodiodes dominate high-speed systems such as fiber-optic receivers, why phototransistors perform best in object-detection and encoder-based applications, and why LDRs remain common in simple ambient-light circuits. It framed performance trade-offs in a way that a student or engineer could use to make an informed choice, and even offered a decision table summarizing key parameters such as speed, sensitivity, linearity, and circuit complexity. The result was a response that showed both breadth and depth, demonstrating an understanding not just of the components themselves but also of the practical realities of circuit design.

Gemini, by comparison, produced a response that was accurate and neatly organized but noticeably more condensed. It correctly identified the fundamental trade-offs between speed, sensitivity, and simplicity and similarly concluded that photodiodes were the best choice for high-speed optical-signal conversion. However, the treatment of the topic did not go as deep into the fine-grained technical behavior of each device. Gemini tended to summarize rather than elaborate, stating that photodiodes offer linear output and extremely fast response times without exploring the implications of these characteristics on circuit design. Likewise, it explained that phototransistors offer higher sensitivity due to internal gain, but it did not expend much effort discussing the rate limitations, saturation characteristics, or recovery times that differentiate various transistor configurations.

Although Gemini did provide a table comparing the same core parameters as DeepSeek, the explanation surrounding the table was more lightweight. Some factors mentioned by DeepSeek—such as wavelength-dependent response curves, dark current considerations, or the mechanical and chemical properties of LDRs—were left out entirely. Instead, Gemini focused primarily on the functional distinctions and on guiding the user toward selecting the correct device. This made its response feel clear and user-friendly, but also more introductory in tone. The writing was easy to read and suitable for an audience that needed a quick explanation, but it lacked some of the additional nuance that would help someone designing or analyzing optical sensing circuits at a deeper level.

Another difference between the models emerged in how they addressed the user's needs. DeepSeek framed its entire answer around the trade-offs that determine which device is best for a specific application. It repeatedly emphasized constraints such as bandwidth, gain, saturation, and the need for external amplification. Its tone was instructional, resembling that of a knowledgeable engineer walking a novice through the decision process. Gemini, on the other hand, adopted a more conversational tone, frequently reminding the user that the "best" device depends on the signal's characteristics. While this framing was helpful, the response sometimes repeated the same three distinctions—speed, sensitivity, and simplicity—rather than diving deeper into the specific reasons those parameters matter.

Even though both models delivered correct and well-structured explanations, the difference in level of detail was clear. DeepSeek's response felt like a comprehensive technical

reference, enriched with examples and real-world use cases. Gemini's answer was more concise and high-level, focusing on the essential ideas without expanding into circuit-level or device-physics considerations. In practice, a response like Gemini's works well for readers who want a quick conceptual overview, while DeepSeek's is more suitable for users who need a deeper engineering analysis or who intend to design an actual optical-detection circuit.

Overall, the difference in how the two models handled the same prompt illustrates the broader contrast between them. DeepSeek tends to produce denser, more technical, and more thorough explanations, whereas Gemini prioritizes clarity and accessibility, even if it sometimes sacrifices depth. The two responses together show how different AI systems balance detail, precision, and readability when tackling engineering-related topics.

6 Chapter 6 Hardware Design

6.1 Optics

6.11 White Light Source (WLS)

Previously the spectral benefits of different light sources were broken down, the following breaks down the cost and performance comparison in Table X.

6.12 Diffraction Grating

As mentioned in Research & Investigation, the core component of a monochromator is the diffraction grating. When selecting a part to use in our project, we considered reflective gratings with a groove period of 1200 lines/mm and blaze angles optimized for visible light of wavelengths ranging from approximately 480 nm to 660 nm. Since we know the incoming beam from the WLS to be 10 mm in diameter, the dimensions of our grating would realistically only have to be slightly wider than 10 mm to fully encompass the spot. Smaller gratings tend to be cheaper than larger ones, so we added the criterion of having the grating be as small as possible while still encompassing the WLS spot. The two major criteria left to consider when buying a grating were a high quantum efficiency over our desired spectrum and a low price. In table XI we compare our narrowed-down list of diffraction gratings. We ultimately selected the second option shown because it best meets these final criteria.

6.13 Photodiodes

Photodiodes will collect the data necessary to calculate the test lens' Abbe number and estimate its material makeup. As stated in Research Investigation, the main optical concerns for these parts are spectral range and peak operating wavelength. For data analysis, a shorter response time for the diodes will be advantageous because we will be taking many data points per lens. Within the parameters of peak and range wavelengths

in the visible portion of the EM spectrum, commercially available photodiodes have been compared in table XII. We ultimately selected the third option shown because it fulfills the optical criteria at a low response time and low cost.

6.14 Monochromator

Given the approximate size of our project, the final monochromator setup should be contained within roughly a 40 cm square in area. Most of this distance will likely be between the diffraction grating and the mirror, as a longer distance means more spectral spread for higher wavelength accuracy.

6.15 Image Acquisition System (IAS)

For the IAS we chose to use the raspberry pi camera platform as it offers an abundance of documentation and inherent support, the following shows the specs considered in the final camera choice (HQ Camera).

The selection of lenses for our Cooke triplet lens assembly starts with our mounting system. The holding diameter of the system is a max of 17mm, depth is 22.5mm. Opening up Zemax we need to create a starter system to then change and optimize. Luckily Zemax has an included project file for the 1896 Cooke triplet patent [60]

6.2 Switching Power Supply PCB System

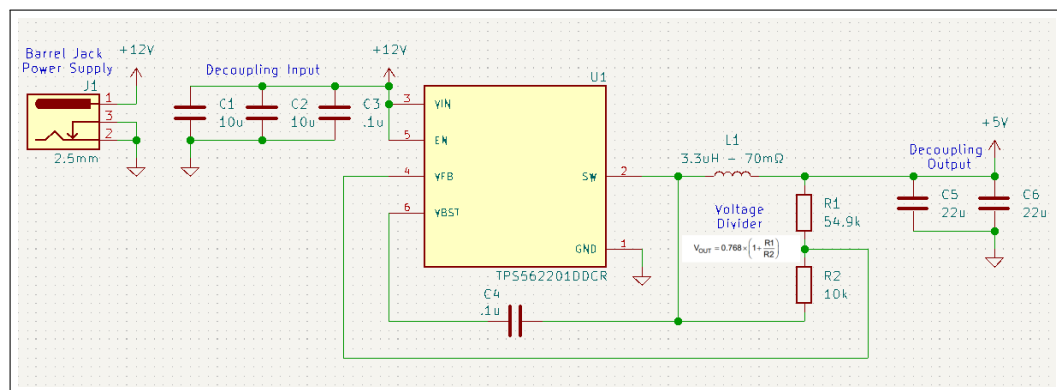


Fig. 47: Switching Regulator Schematic

a) Buck Converter Design Using the TPS562201

The power stage uses the TPS562201 synchronous buck regulator to step the 12 V input down to a regulated 5 V rail. Input decoupling capacitors are placed directly at the VIN pin to reduce high frequency transients, and the output network consists of a 3.3 uH inductor

and low ESR 22 μF capacitors to ensure stable transient response. The feedback divider is sized to set the output to 5 V based on the device reference voltage, while the SW and BST nodes follow the typical layout and filtering recommendations for minimizing switching noise. This device was chosen instead of the commonly used LM2596 because it offers much higher switching frequency, significantly smaller required passives, and higher overall efficiency due to its synchronous rectification. The TPS562201 also provides lower output ripple, faster transient response, and a much smaller footprint compared to the older, non synchronous LM2596 architecture. As a result, the final design achieves better power density, reduced heat dissipation, and improved electrical performance while maintaining stable 5 V output for downstream digital circuitry.

6.3 Microcontroller PCB System

6.31 Microcontroller Implementation

The STM32 microcontroller was implemented as a development board with all of the pins broken out allowing use of all of the STM32's many features. This layout allows for the use of all of the chips features. Specific attention was paid to peripheral assignments, power distribution, and ease of integration into the larger system. The reasoning behind these design choices that guided the final configuration is discussed in detail throughout the remainder of the microcontroller section.

6.32 3V3 Regulator

The AMS1117-3.3 linear regulator is a widely used component in microcontroller-based systems due to its simplicity and availability. Its ability to provide a stable 3.3 V rail from a common 5 V input makes it particularly useful for USB powered applications. In this application the 5V input can be sourced from either the USB-C bus or from the V-Bus pin seen in the breakout. If sourced from the V-BUS pin the voltage may be as low as 4.5 Volts assuming low voltage ripple. The following is a generalized equation to determine this value. Note that this is idealized and assumes an insignificant voltage ripple.

$$V_{\text{IN,DC,min}} = V_{\text{OUT}} + V_{\text{DO}}$$

The regulator requires minimal external components (only two decoupling capacitors, C12 and C13), offers sufficient current capacity (up to 800 mA with proper heat dissipation), and has built-in protection features such as thermal shutdown and short-circuit protection. These characteristics make it a reliable and cost-effective choice for powering 3.3 V microcontrollers and peripherals in embedded designs. One important design consideration is heat dissipation: because the AMS1117 is a linear regulator, the power it must dissipate is proportional to the voltage drop across it multiplied by the output current

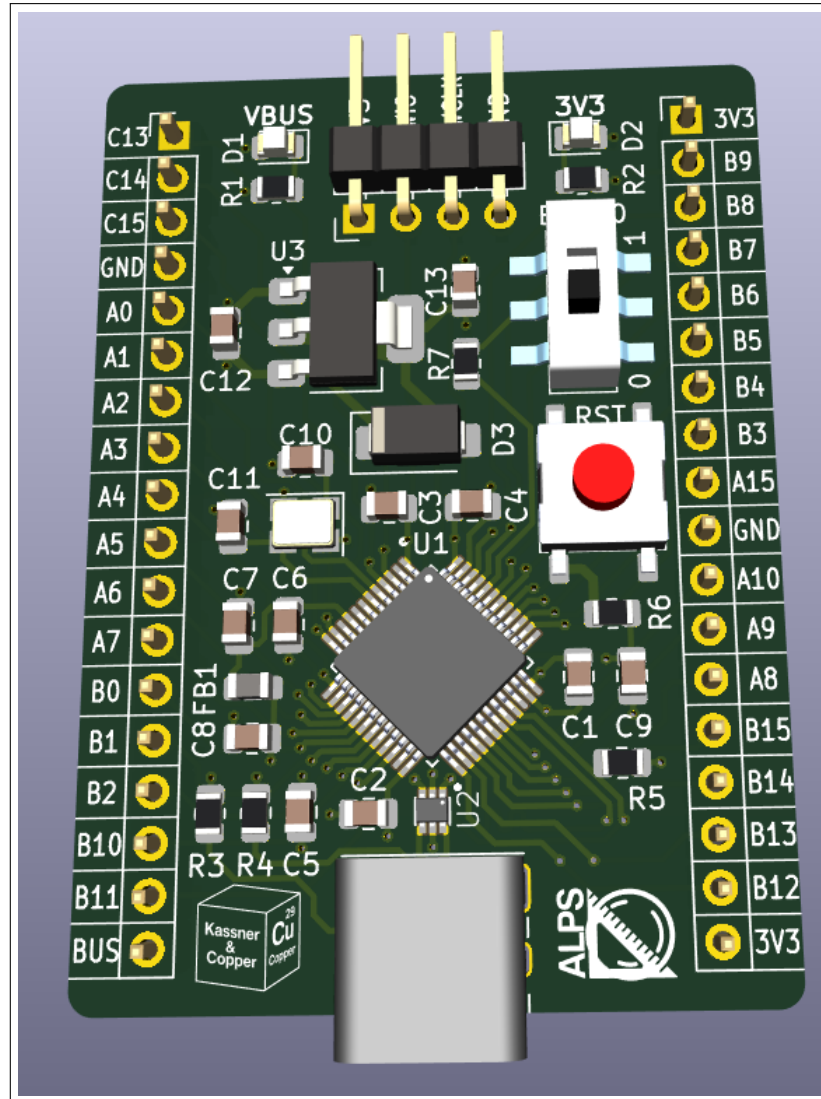


Fig. 48: STM32 PCB Rendering

$$P = (V_{in} - V_{out}) I_{out}$$

At higher loads, this can lead to significant heat generation, so thermal management and efficiency tradeoffs must be evaluated during system design. For usage in an STM32 system with an estimated load of 150mA. This is an absolute limit as the STM32 chip itself does not allow for more than 150mA. [58]

A Schottky diode (D3) is included to prevent reverse current flow, which can occur if the regulator's output capacitor discharges faster than the input capacitor during power-down events. Without this diode, the charge stored on the output side could flow backward through the regulator, potentially damaging it or causing instability. Schottky diodes are preferred in this application because of their low forward voltage drop and fast switching

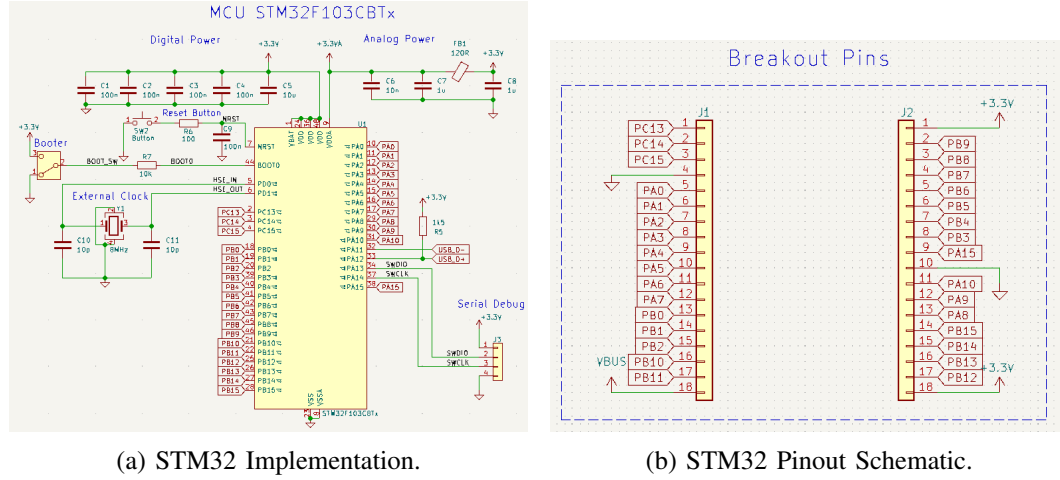


Fig. 49: STM32 Overview

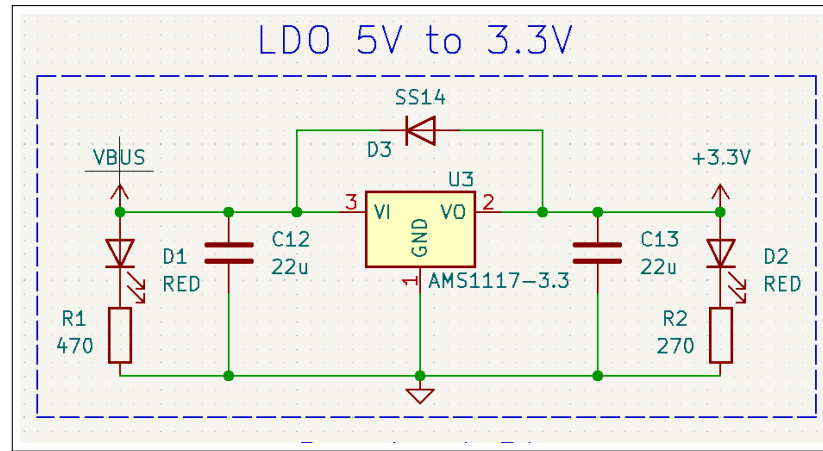


Fig. 50: AMS1117-3.3 Implementation

characteristics, which provide effective protection without significant power loss.[54]

Finally, the LED indicators (D1 and D2) serve as visual cues to confirm that the system is powered and the 3.3 V rail is active. Current-limiting resistors (R1 and R2) were calculated based on the LED's forward voltage drop

$$V_f \approx 2.0 \text{ V}$$

and desired operating current using the following equations.

$$I_{LED} = \frac{V_{SUPPLY} - 2.0 \text{ V}}{R_{LED}}$$

This ensures reliable illumination while avoiding excessive current draw that could damage the LEDs or waste power. This provides the user with a simple and intuitive way to verify proper system operation.

6.33 Digital Power Supply Decoupling Implementation

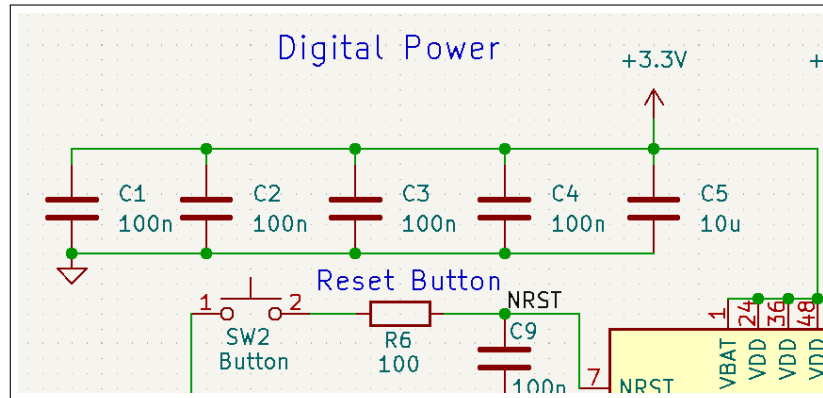


Fig. 51: Digital Power Supply Decoupling Implementation

To ensure a stable and noise-free supply for the STM32, each power rail is decoupled using a 100 nF ceramic capacitor placed close to the corresponding power supply pins: VBAT and the three VDD pins. These capacitors provide high-frequency filtering, shunting fast switching transients directly to ground before they can create noise in the STM32's power supply. In addition to these local capacitors, a single 10 μ F bulk capacitor is included on the rail to supply charge during slower load changes and to smooth out lower-frequency voltage dips. This combination of distributed high-frequency decoupling and bulk energy storage maintains a clean 3.3 V supply, reducing the risk of logic errors, resets, or EMI issues.

6.34 Analog Power Supply Decoupling Implementation

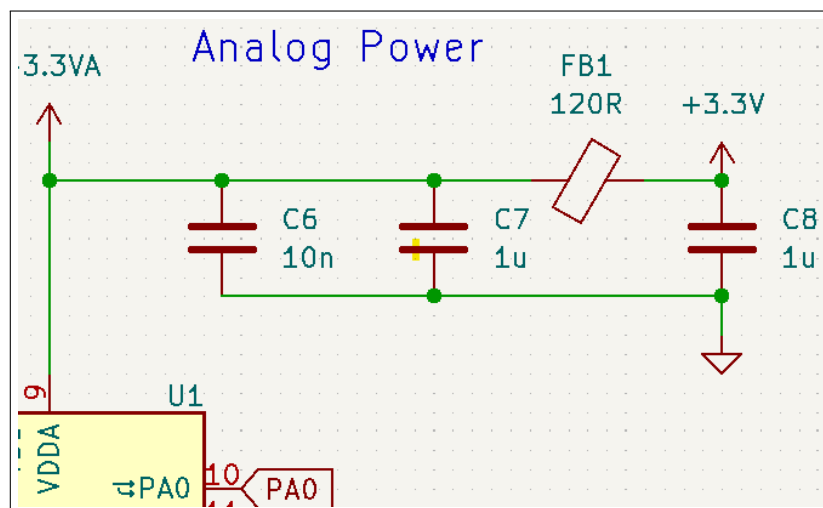


Fig. 52: Analog Power Supply Decoupling Implementation

Using the concepts from the previously provided formula the analog supply (VDDA) is isolated from the main 3.3 V rail using a ferrite bead (FB1), which suppresses high-frequency noise originating from the digital circuitry.[16] On the VDDA side, a bank

of capacitors (10 nF, 1 μ F, and 1 μ F) provides broadband decoupling, ensuring a stable and quiet analog reference. The 10nF capacitor targets high frequencies while the 10 μ F capacitors act as local power banks. The combination of the ferrite bead's impedance and the shunt capacitors creates a low-pass filter, effectively delivering a clean 3.3 V analog supply for the ADC and other sensitive analog functions of the STM32. [16]

6.35 USB C USB 2.0 Implementation

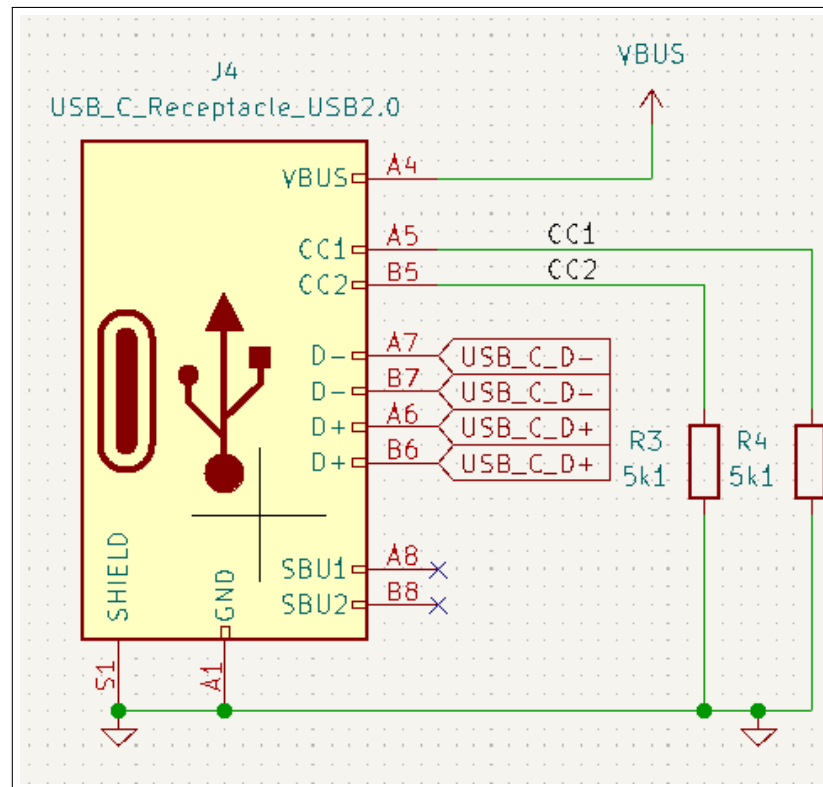


Fig. 53: USB-C Implementation

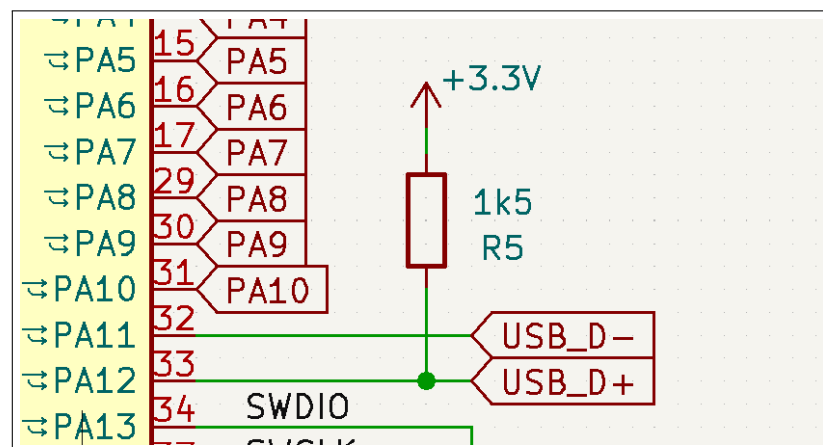


Fig. 54: USB 1.5 k Ω Pull Up Resistor

After careful consideration, a decision was made to implement USB 2.0 (Full-Speed, 12 Mbps) connectivity using the GCT USB4085 USB-C receptacle, configured for USB-only operation. To activate this protocol, a 1.5 k Ω pull-up resistor must be placed on the D+ USB line. Because the system does not require USB Power Delivery or high-speed signaling, only the VBUS, GND, and differential D+/D- lines are routed, along with the CC1 and CC2 configuration channel pins required for USB-C detection. Each CC line is terminated to ground with a 5.1 k Ω 0805 pull-down resistor, identifying the board as a UFP (Upstream Facing Port) that receives power from the host. The D+ and D- lines are routed as a matched differential pair with controlled impedance (approximately 90 Ω differential) and carefully maintained equal lengths to minimize skew and ensure signal integrity at the full-speed data rate. Trace widths and spacing are tuned to the board's dielectric stack-up, and both traces are kept on the same layer with minimal via transitions to preserve symmetry. The calculations for this impedance matching as well as the differential routing are shown below. [67]

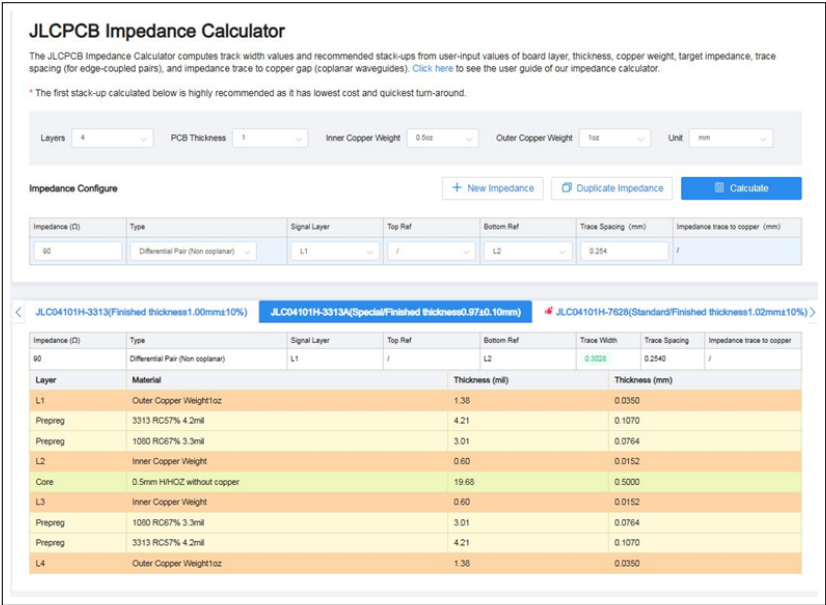


Fig. 55: JLCPCB Impedance Calculations Used in Design Implementation

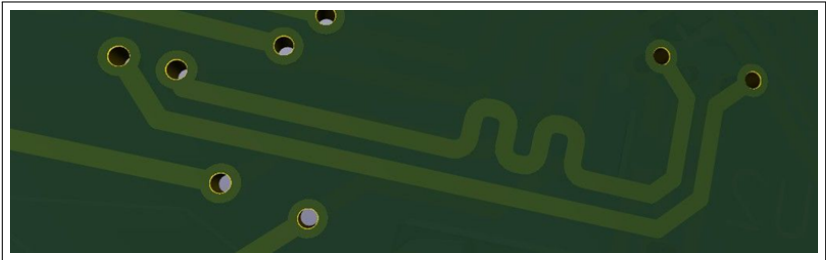


Fig. 56: USB Differential Traces Rendering

The VBUS line is connected through a current-limiting fuse and local decoupling capacitors to provide a stable 5 V input while minimizing conducted noise on the power rail. The connector's shield and ground tabs are tied directly to the system ground plane

to ensure signal reference stability and mechanical robustness. A 3D printed encasement will be added later to increase the structural stability of the port. Overall, this layout provides a standards-compliant USB 2.0 interface that ensures reliable system.

6.36 USB ESD Protection Implementation

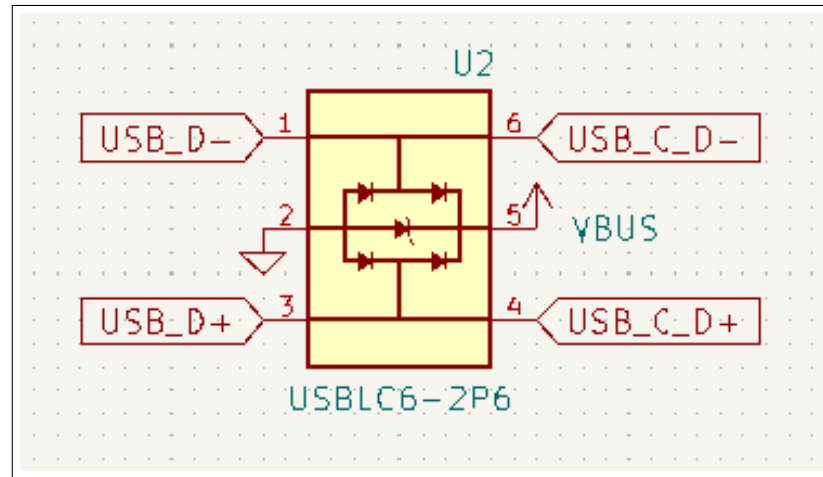


Fig. 57: ESD Protection Implementation

In embedded hardware design, Transient Voltage Suppression (TVS) diodes are the standard method for mitigating ESD events by clamping high-voltage transients and safely diverting the excess energy to ground. However, the current design imposes a minimum component size constraint of 0805, making it impractical to use individual TVS diodes on every line without introducing routing congestion. To address this, the implementation of a TVS diode array IC is recommended, as it provides multi-line protection in a compact package. A suitable choice for this application is the USBLC6-2P6, which is specifically designed for high-speed USB data line protection and integrates matched diodes to maintain signal integrity while providing robust ESD suppression across the D+, D-, and VBUS lines.

An alternative originally considered for ESD protection was the Texas Instruments TPD8S300, a highly integrated USB Type-C port protector that combines IEC-rated ESD suppression with overvoltage protection and support for USB Power Delivery. This device is designed to safeguard multiple high-speed differential lines such as SBU1, SBU2, and USB SuperSpeed pairs, making it well-suited for Type-C implementations where short-to-VBUS conditions and higher voltage negotiation are expected. While its feature set provides robust protection for USB-C designs, the additional functionality was unnecessary for the present application, which only required protection for USB 2.0 signaling and power lines. To maintain simplicity and avoid unnecessary routing complexity, the design instead employed the USBLC6-2P6, which offers a compact and cost-effective solution optimized for ESD protection on the D+, D-, and VBUS lines without the overhead of full USB-C support [32].

The USBLC6-2P6 is a low-capacitance TVS diode array IC designed by STMicroelec-

tronics specifically for USB and high-speed data line protection. It integrates multiple transient suppression diodes into a single compact SOT-666 package, providing efficient protection for both the D+ and D- differential signal lines, as well as the VBUS power line. Each internal diode is engineered to clamp voltage surges quickly during ESD or transient events, redirecting the excess energy safely to ground while maintaining normal signal operation under typical voltage conditions. One of the key advantages of the USBLC6-2P6 is its very low input/output capacitance, typically around 0.6pF, which ensures minimal distortion and signal loss on high-speed USB-2.0 lines operating at 480 Mbps; far higher speeds than our full-speed USB-2.0 12Mbps application. This makes it far superior to discrete TVS diodes when protecting differential data lines, as it preserves signal integrity while offering consistent and symmetric protection. The IC also boasts low leakage current and a fast response time (in the nanosecond range), ensuring reliable operation even in noise-sensitive or power-constrained designs.

From a manufacturing standpoint, the USBLC6-2P6's integrated multi-line configuration greatly simplifies PCB layout and reduces routing complexity compared to placing separate TVS diodes for each line, an important benefit given the project's minimum component size constraint of 0805. Its compact footprint helps conserve board space while ensuring a single, well-grounded return path for transient energy, improving overall ESD robustness. For these reasons, the USBLC6-2P6 represents a practical and efficient solution for safeguarding the project's USB interface without compromising performance, manufacturability, or signal integrity.

6.37 SWDIO Protocol Implementation

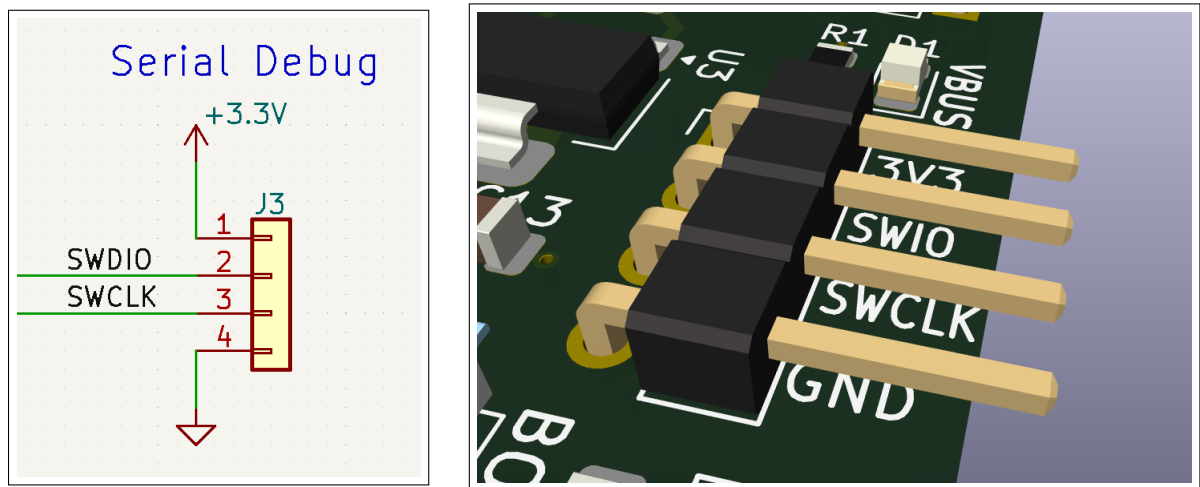


Fig. 58: SWDIO Schematic (Left) and SWDIO PCB Rendering (Right)

The schematic shown illustrates the Serial Wire Debug (SWD) header implementation used for programming and debugging the microcontroller. It provides four essential connections: SWDIO for bidirectional data transfer, SWCLK for clock synchronization, a 3.3 V reference supply, and ground. Careful placement of this header on the PCB ensures accessibility during development and testing without interfering with the main system layout.

6.38 Communication Protocol Implementation

As demonstrated, both I²C, SPI, and UART provide effective means of serial communication, each suited to different roles. For our design only I²C and UART will be utilized. SPI will be avoided due to the extreme difficulties incurred when debugging. UART will serve as the primary protocol for communication between the Raspberry Pi and the subordinate microcontrollers, providing a straightforward and reliable point-to-point link. In contrast, I²C will be employed by the microcontrollers to interface with peripheral electronic devices, leveraging its multi-slave bus structure and efficient two-wire design to minimize pin usage while supporting multiple peripheral connections.

Since all of the STM32 pins are broken out in our microcontroller PCB, we can choose to utilize these communications as we go, allowing flexibility in implementation.

6.39 External Clock

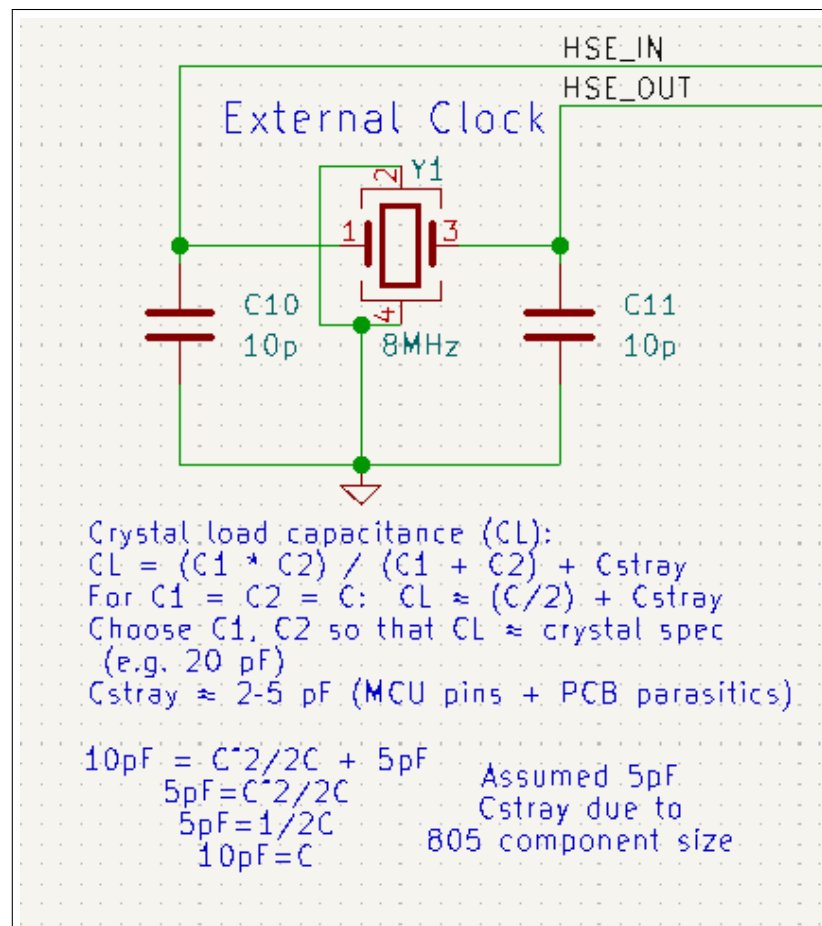


Fig. 59: External Clock Implementation [58]

The use of external crystal oscillators in microcontroller designs requires the inclusion of load capacitors to ensure stable and accurate oscillation at the desired frequency. A quartz crystal by itself behaves as a resonant element, but to operate in its fundamental mode

it must be paired with capacitors that create the proper load capacitance (C_L) specified in the crystal's datasheet. The effective load capacitance is determined by the series combination of the two external capacitors (C_1 and C_2), along with the stray capacitance introduced by MCU pins and PCB parasitics. The formula shown in Fig. 15 is used to select capacitor values such that the oscillator sees the correct load and resonates at the specified frequency (e.g., 8 MHz in this design). Without properly chosen capacitors, the oscillator may shift in frequency, exhibit poor startup behavior, or fail to oscillate entirely. In this case, 10 pF capacitors were selected, accounting for an estimated 5 pF of stray capacitance, to satisfy the crystal's load requirement while ensuring reliable startup and long-term frequency stability. The unusually high 5 pF stray capacitance assumption arises from the constraint of using 0805 component sizes including the load capacitors for the oscillator.

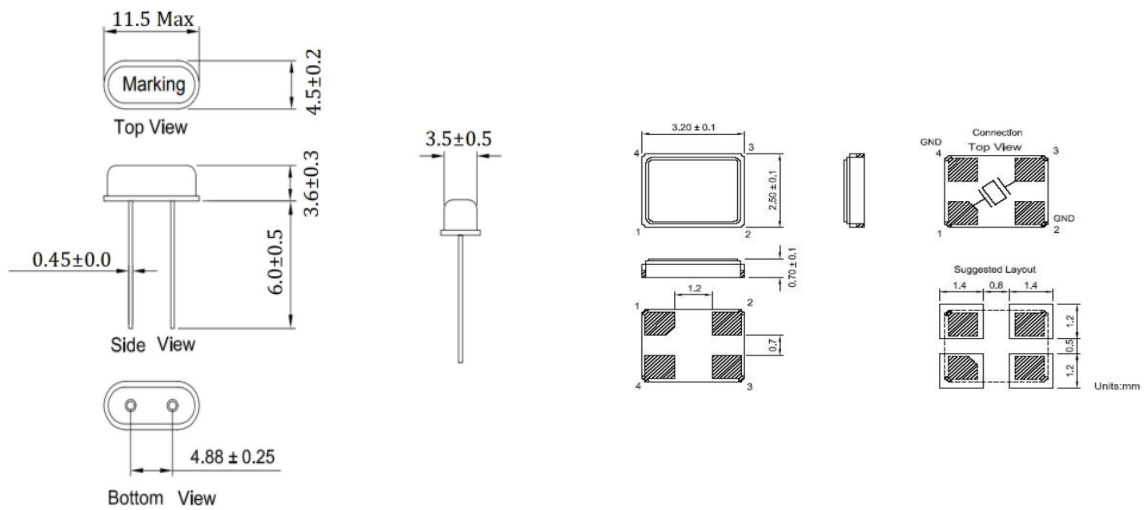


Fig. 60: X49S (Left) and TX3225 (Right)

When considering oscillators for an STM32 board, the X49S series stands out mainly because it is the standard choice on the popular STM32 “Blue Pill” board. Its proven track record and widespread community support make it a safe and well-understood option, especially if you want compatibility with existing examples and tutorials. However, it comes in a through-hole package, which can complicate routing since it blocks the back copper plane and forces traces to route around the pins. By contrast, the TX3225 (3.2 × 2.5 mm SMD crystal) costs only about one cent more on JLCPCB but offers a much smaller footprint and the advantages of being surface-mount: it doesn't interfere with backside routing and helps keep the PCB more compact. The trade-off is that it lacks the same legacy presence as the X49S in STM32 reference designs, but from a manufacturing and layout perspective, it's cleaner and more efficient.

6.310 Reset and Boot

The STM32 boot process is determined by the configuration of the `BOOT0` pin, the `nBOOT0` and `nBOOT1` option bits, and the `BOOT_SEL` bit. As shown in Fig. 17, different combinations of these signals select whether the device boots from main Flash, system memory (bootloader), or embedded SRAM. By default, holding `BOOT0` low causes the

Boot mode configuration				Mode
nBOOT1 bit	BOOT0 pin	BOOT_SEL bit	nBOOT0 bit	
x	0	1	x	Main Flash memory is selected as boot area ⁽²⁾
1	1	1	x	System memory is selected as boot area
0	1	1	x	Embedded SRAM is selected as boot area
x	x	0	1	Main Flash memory is selected as boot area
1	x	0	0	System memory is selected as boot area
0	x	0	0	Embedded SRAM is selected as boot area

Fig. 61: STM32F103C8Tx BOOT TABLE [25]

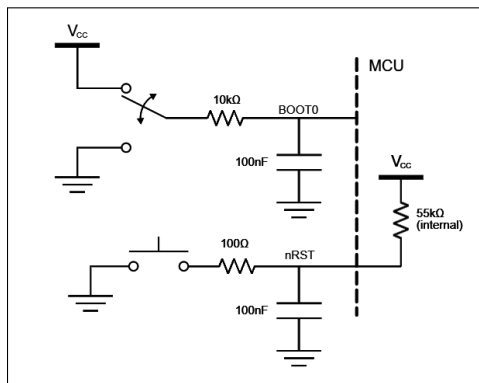


Fig. 62: Design Recommendation [25]

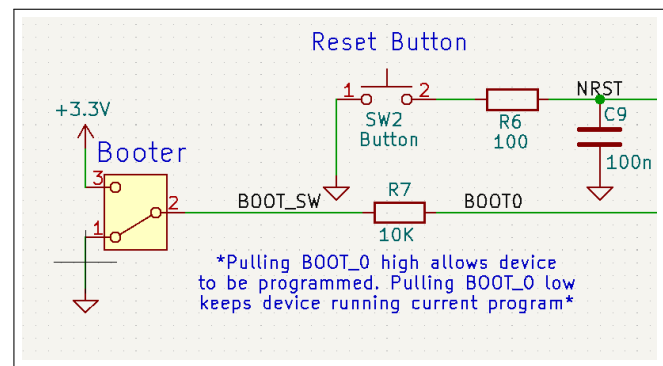


Fig. 63: Reset Button and Boot Switch Implementation

MCU to run the program stored in Flash, while pulling it high enables the system memory bootloader for programming via serial interfaces.

To support this functionality, a blog post on Acheron Docs recommends the hardware circuit in Fig. 18. The `BOOT0` pin is pulled high or low with a 10 kΩ pull-down resistor leading to a toggle-able switch. A pushbutton, pulled to V_{CC} normally by an internal 55 kΩ resistor, allows `BOOT0` to be driven high normally and driven low with a button press. A 100 nF capacitor provides filtering against noise.

Using these references, the circuit shown in Fig. 19 was implemented. The design closely follows the recommended circuit in Fig. 18.

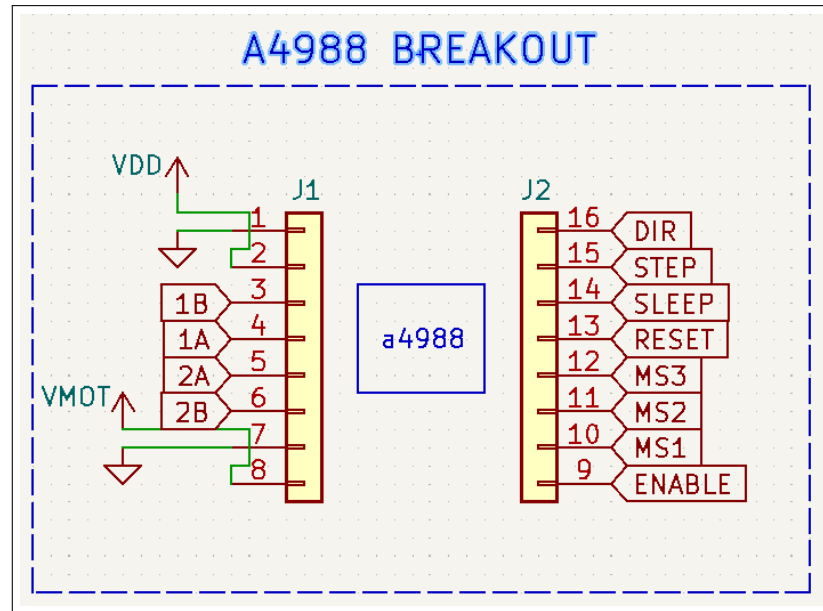


Fig. 64: a4988 Breakout

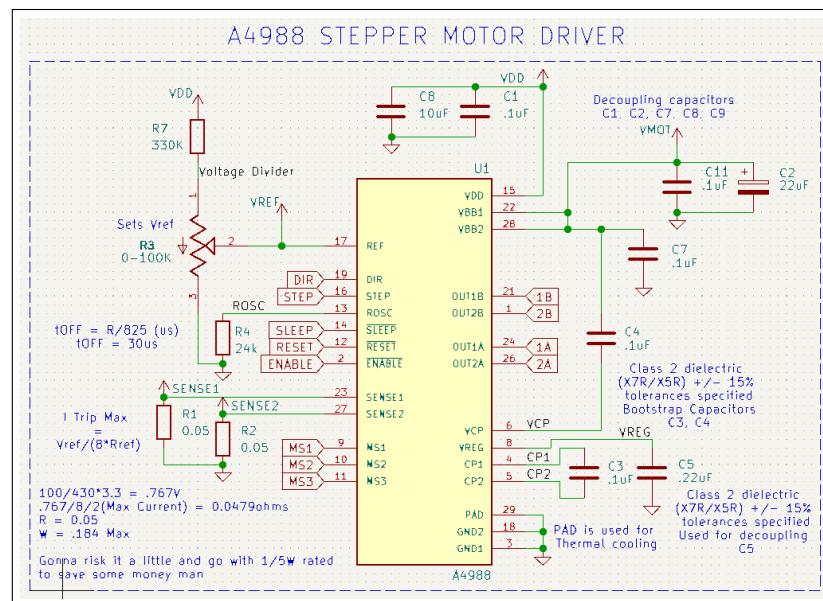


Fig. 65: a4988 Main Schematic

6.4 Stepper Motor Driver PCB System

6.41 IC Implementation

a) Decoupling and Power Conditioning

Our design closely follows the reference application circuit provided in the Allegro A4988 datasheet [42], with particular attention to decoupling strategy for both the logic supply

(VDD) and the motor supply (VBB). Local ceramic capacitors (C1, C7, C8, C9) are placed as close as possible to their respective pins to suppress high-frequency switching transients, while the 22 μF bulk capacitor C2 on VMOT provides the surge energy required for coil energization. All ceramics are Class II (X5R/X7R) with $\pm 15\%$ tolerance, ensuring stable capacitance under temperature swings and DC bias.

b) Current Sensing and Regulation

Current regulation is implemented using the low-value sense resistors R1 and R2, which allow the A4988's fixed off-time PWM current control to accurately regulate winding current. The reference voltage V_{REF} is set by a voltage divider consisting of R7 and the adjustable potentiometer R3, producing the current limit

$$I_{\text{TripMAX}} = \frac{V_{\text{REF}}}{8R_{\text{sense}}}.$$

The chosen resistor values yield a usable V_{REF} range while keeping power dissipation in the sense resistors within acceptable limits. Although the calculated worst-case power approaches the resistor rating, the design intentionally accepts this margin for cost efficiency while remaining safe in typical operating conditions.

c) Charge Pump and Bootstrap Network

The bootstrap capacitors C3 and C4 support the internal charge pump that drives the high-side MOSFETs. Their placement and dielectric class follow the datasheet's guidance, ensuring sufficient gate drive voltage even at higher step frequencies. Undervalued or distant bootstrap capacitors can cause missed steps or unstable current regulation, so these components are selected and positioned with care.

d) Timing and Mixed Decay Control

The ROSC timing resistor R4 sets the fixed off-time for the current control loop according to

$$t_{\text{OFF}} \approx \frac{R_{\text{OSC}}}{825}.$$

The resulting off-time of approximately 30 μs influences the mixed-decay behavior of the current waveform, reducing audible noise and improving microstepping linearity. The MS1, MS2, and MS3 pins provide selectable microstepping resolution, offering flexibility between precision and speed according to application requirements.

e) Thermal Design and PCB Layout

The exposed thermal pad (PAD) is tied directly into the ground plane to create a low-thermal-resistance path for heat dissipation. Because the A4988 dissipates most of its heat through this pad, stitching it to a large GND copper region significantly improves thermal reliability under high current loads. This approach aligns with Allegro's PCB layout recommendations and ensures stable operation even during extended motor activity.

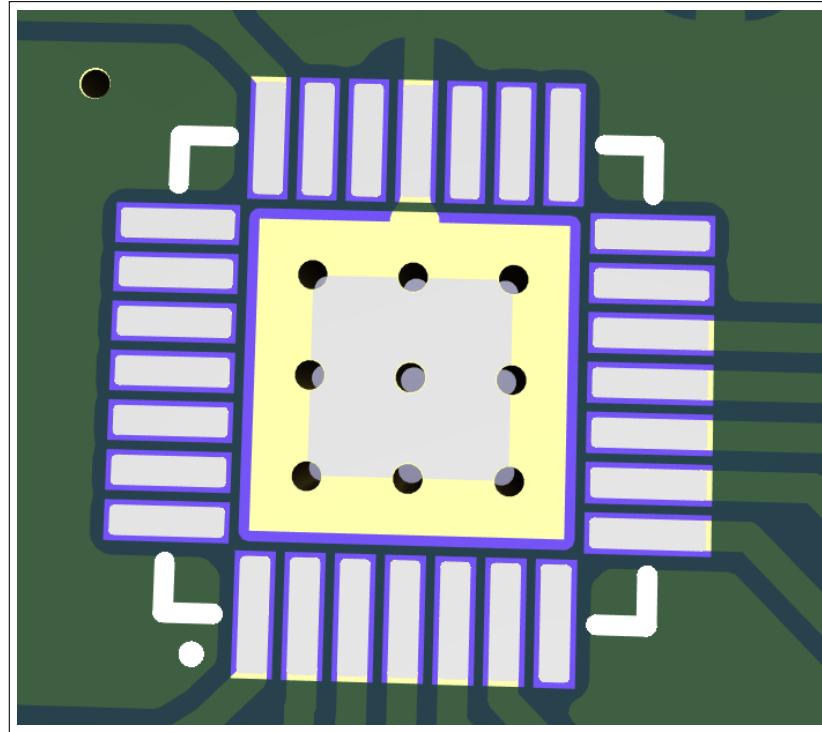


Fig. 66: Thermal GND Vias

f) Pull-Down Resistors on ENABLE, DIR, and STEP

The ENABLE, DIR, and STEP inputs are all digital control lines that become undefined when left floating, which can lead to unpredictable motor behavior. To prevent accidental motion or mis-stepping, each of these pins is equipped with a $24\text{ k}\Omega$ pull-down resistor that forces the signal to a known logic-low state whenever the microcontroller is inactive, held in reset, or disconnected. This ensures that the driver remains disabled by default (ENABLE low), the direction defaults to a consistent state (DIR low), and no unintended pulses are presented to the STEP input. Pull-down biasing is therefore essential for safety, deterministic startup behavior, and noise immunity on long or externally routed control traces.

g) LED Indicators for VDD, DIR, and STEP

The LED indicators provide immediate visual confirmation of the logic state on the VDD, DIR, and STEP pins, offering a valuable debugging tool during development and testing. Each indicator consists of a red LED in series with a 200Ω resistor to limit current, producing a clear, low-intensity signal without significantly loading the control line. The VDD LED confirms proper logic power supply operation, while the DIR and STEP LEDs allow real-time observation of direction changes and stepping pulses—useful when verifying firmware timing, checking for noise or double-pulsing, and diagnosing wiring issues. Because stepper drivers often fail silently when timing signals are incorrect, these LEDs provide essential at-a-glance insight into system behavior.

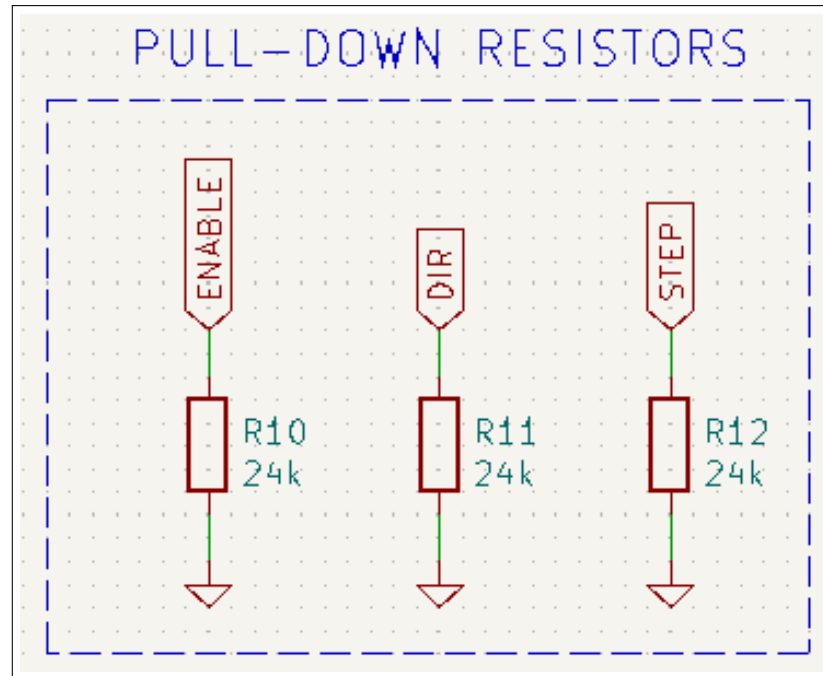


Fig. 67: Pull-Down Resistors

h) Pull-Up Resistors for SLEEP, RESET, and Microstepping Pins

The SLEEP, RESET, and microstepping configuration pins (MS1–MS3) are pulled up to VDD to ensure reliable default operation of the A4988 when no microcontroller is actively driving these signals. The four-resistor network (R6) provides a compact and consistent pull-up path that guarantees the driver powers up in a fully enabled state, with a known microstepping mode rather than floating inputs that could cause erratic stepping. The additional pull-up on SLEEP (R13) ensures the device remains awake unless intentionally placed in low-power mode. Using pull-ups here aligns with Allegro's recommended default logic levels and ensures a predictable startup configuration while still allowing firmware to override each pin as needed.

7 Chapter 7 Software Design

7.1 Investigation and Testing

7.11 STM32 Development Environment

During the initial phase of the project, significant effort was dedicated to learning and mastering the STM32 development environment. Since the STM32 Blue Pill serves as the core embedded controller for the Automated Lens Parameter Metrology system, understanding the software tools used for firmware development, debugging, and pro-

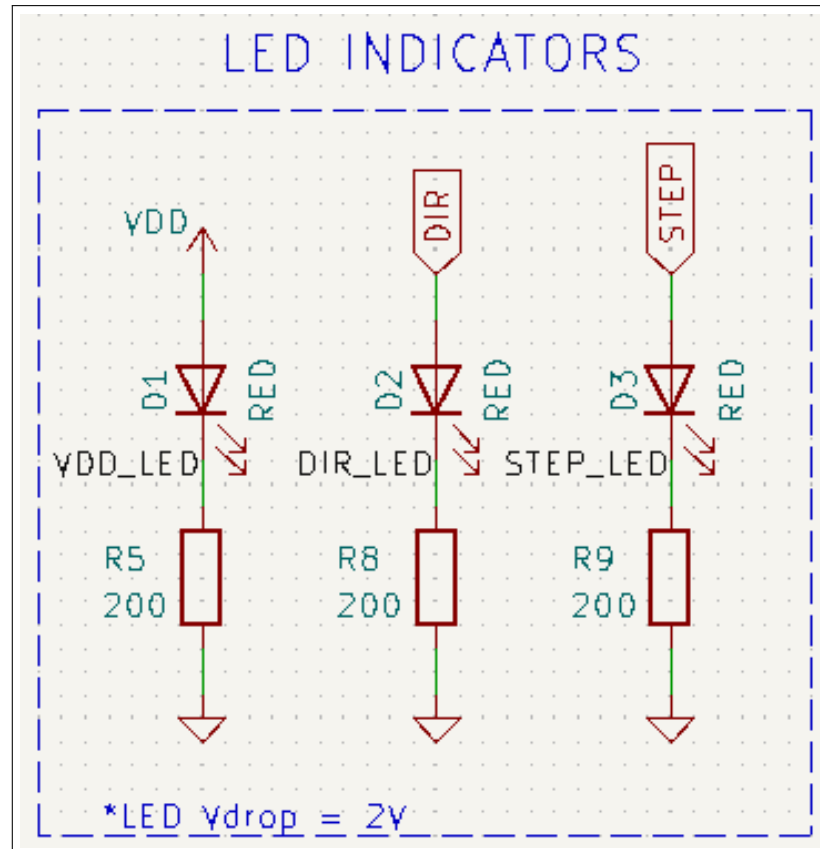


Fig. 68: LED Indicators

programming was essential for reliable progress. Two primary tools were utilized throughout the research and testing stages: **STM32CubeIDE** and **STM32CubeProgrammer**.

Both tools are part of STMicroelectronics' integrated development ecosystem designed to support the entire workflow—from code generation to device programming and debugging. Gaining proficiency with these tools was a necessary milestone in building the firmware components used for UART, I²C, and ADC communication experiments.

a) Overview of STM32CubeIDE

STM32CubeIDE is an integrated development environment (IDE) built on the Eclipse framework and supported by STMicroelectronics. It combines code editing, configuration, compilation, and debugging capabilities in a single workspace. It integrates closely with STM32CubeMX, a graphical configuration utility that simplifies peripheral initialization through an interactive pinout and clock configuration interface.

At the beginning of the project, the STM32CubeIDE was installed and configured to recognize the STM32F103C8T6 microcontroller used on the Blue Pill board. This required setting up the proper toolchain, verifying USB drivers, and connecting the ST-Link V2 debugger to the development board. Once configured, CubeIDE automatically detected

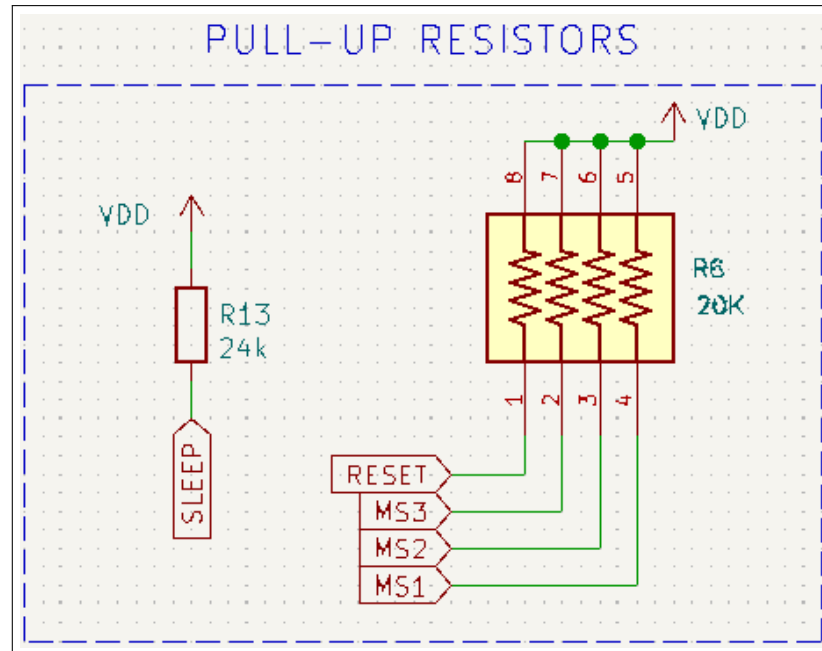


Fig. 69: Pull-Up Resistors

the connected MCU, populated its peripheral map, and generated a default initialization project.

Within CubeIDE, code development primarily revolved around the following tasks:

- **Peripheral Configuration:** Using the graphical configuration tool, we enabled and configured peripherals such as UART, ADC, GPIO, and I²C. For instance, UART1 (TX = PA9, RX = PA10) and ADC1 Channel 0 (PA0) were set up with precise pin assignments and sampling settings.
- **Clock Tree Management:** The system clock was configured for 72 MHz operation using the internal high-speed oscillator (HSI) or an external 8 MHz crystal, multiplied through the PLL. CubeIDE provides an intuitive clock tree diagram that visualizes frequency propagation, allowing us to ensure accurate peripheral timing.
- **Code Generation:** Once configuration was complete, CubeIDE automatically generated initialization code through the HAL (Hardware Abstraction Layer) libraries. This process created a `main.c` file and an `MX_XXXX_Init()` function for each peripheral, significantly reducing setup complexity.
- **Compilation and Build:** Using the built-in GCC ARM toolchain, firmware was compiled into `.elf` and `.bin` files. The console logs were reviewed to confirm successful builds and to identify potential errors in peripheral initialization or syntax.
- **Debugging and Live Variable Inspection:** STM32CubeIDE includes a full-featured debugger that allows live variable tracking, breakpoint setting, and register inspection. These capabilities were instrumental in confirming the real-time behavior of UART and ADC readings and in diagnosing communication synchronization issues.

Learning to navigate STM32CubeIDE required understanding the relationship between

generated code and user-defined logic. The auto-generated sections, guarded by comment markers, separate code that can safely be modified from code that should remain intact to preserve regeneration compatibility. This workflow taught best practices for structured embedded development, enabling efficient firmware iteration without reinitializing configurations manually.

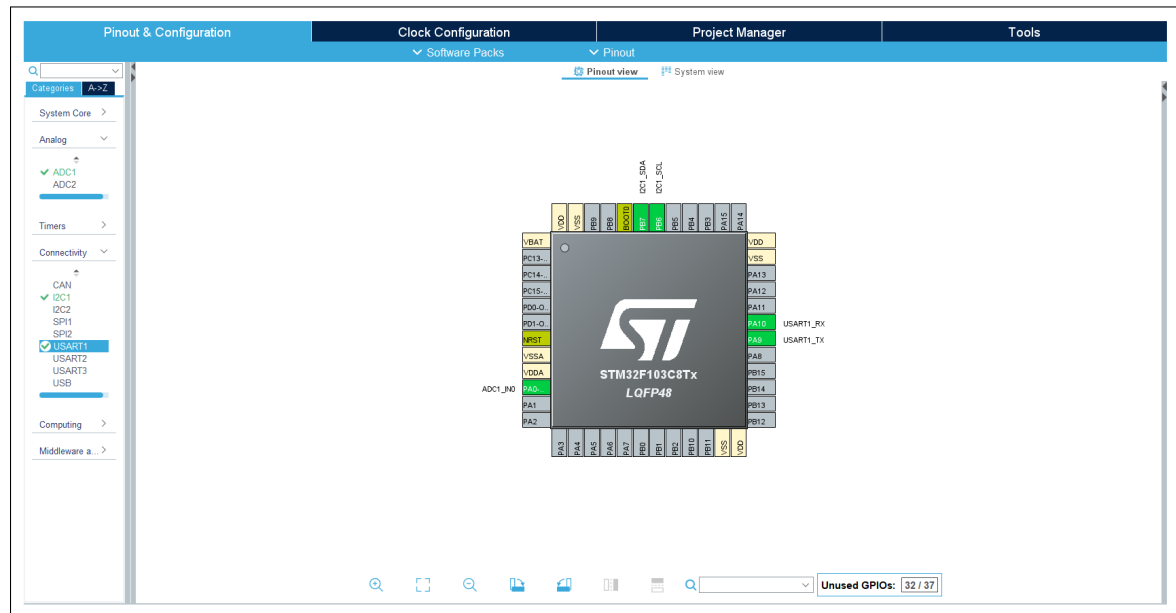


Fig. 70: STM32CubeIDE Pinout and Configuration Menu

b) Project Creation and Workflow

Creating an STM32 project in CubeIDE follows a defined workflow that we repeatedly applied during testing phases. The steps are summarized below:

- 1) **Start a New STM32 Project:** From the launch menu, a new STM32 project was created by selecting the target MCU—STM32F103C8Tx—from the device database.
- 2) **Configure Pinout:** The Pinout and Configuration tab was used to visually assign pins. For example, PA9 and PA10 were assigned to USART1 for serial communication, while PA0 was reserved for ADC input.
- 3) **Adjust Clock Configuration:** The clock settings were modified to achieve stable system timing. The internal PLL was configured to reach a 72 MHz system clock for consistent ADC sampling and UART baud rate generation.
- 4) **Enable Peripheral Settings:** Each peripheral was opened in the Configuration tab to fine-tune parameters such as baud rate (e.g., 115200 bps for UART), ADC resolution, sampling cycles, and I²C addressing mode.
- 5) **Generate Initialization Code:** Clicking “Generate Code” produced the `main.c`, `stm32f1xx_hal_msp.c`, and `stm32f1xx_it.c` files with function templates for initialization and interrupt handling.
- 6) **Implement Application Logic:** Within the user code sections, custom functions were added to perform continuous ADC sampling, UART data transmission, or I²C

communication routines.

- 7) **Build and Flash Firmware:** The code was built, and the resulting .bin file was loaded onto the STM32 using either the integrated ST-Link debugger or STM32CubeProgrammer.

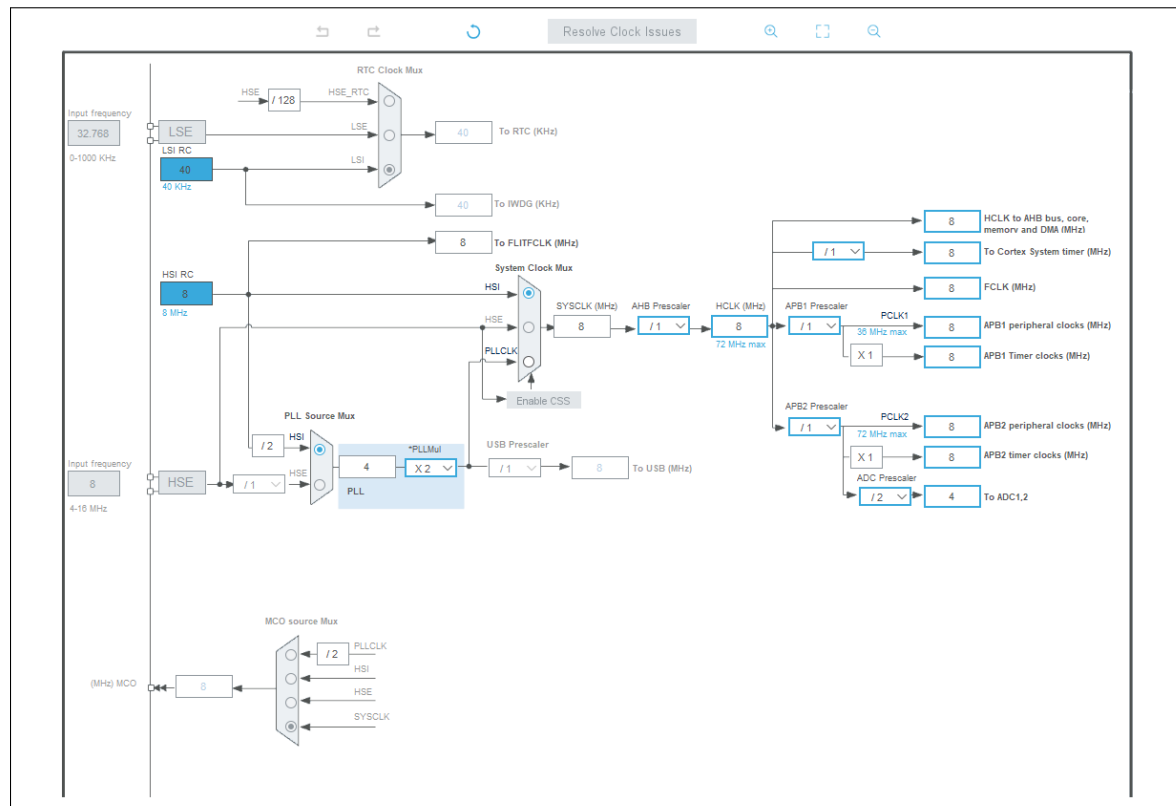


Fig. 71: STM32CubeIDE Clock Configuration Menu

Each iteration of this workflow allowed us to refine peripheral configurations and experiment with new firmware modules. Over time, this process became the backbone of our embedded development routine.

c) Overview of STM32CubeProgrammer

STM32CubeProgrammer is a standalone utility that allows users to flash firmware onto STM32 microcontrollers through various interfaces such as ST-Link, UART, or USB DFU. It is also capable of performing memory inspection, option byte modification, and debugging connectivity issues.

We initially used STM32CubeProgrammer to verify board connectivity and to upload compiled binaries to the Blue Pill via the ST-Link interface. The tool provided valuable feedback, including detected device ID, flash memory size, and bootloader configuration. These features helped confirm correct wiring and MCU detection early in the project.

The workflow for programming the STM32 through CubeProgrammer typically included:

- 1) **Connecting the ST-Link:** The ST-Link programmer was connected to the STM32

via SWDIO, SWCLK, 3.3V, and GND pins.

- 2) **Detecting the Device:** The “Connect” button in CubeProgrammer automatically recognized the STM32F103C8T6 and displayed detailed chip information.
- 3) **Loading the Firmware:** The compiled .bin file was selected through the interface.
- 4) **Flashing the Device:** Clicking “Start Programming” wrote the firmware into flash memory and verified the checksum.
- 5) **Running the Program:** Upon successful flashing, the MCU was reset to execute the uploaded firmware.

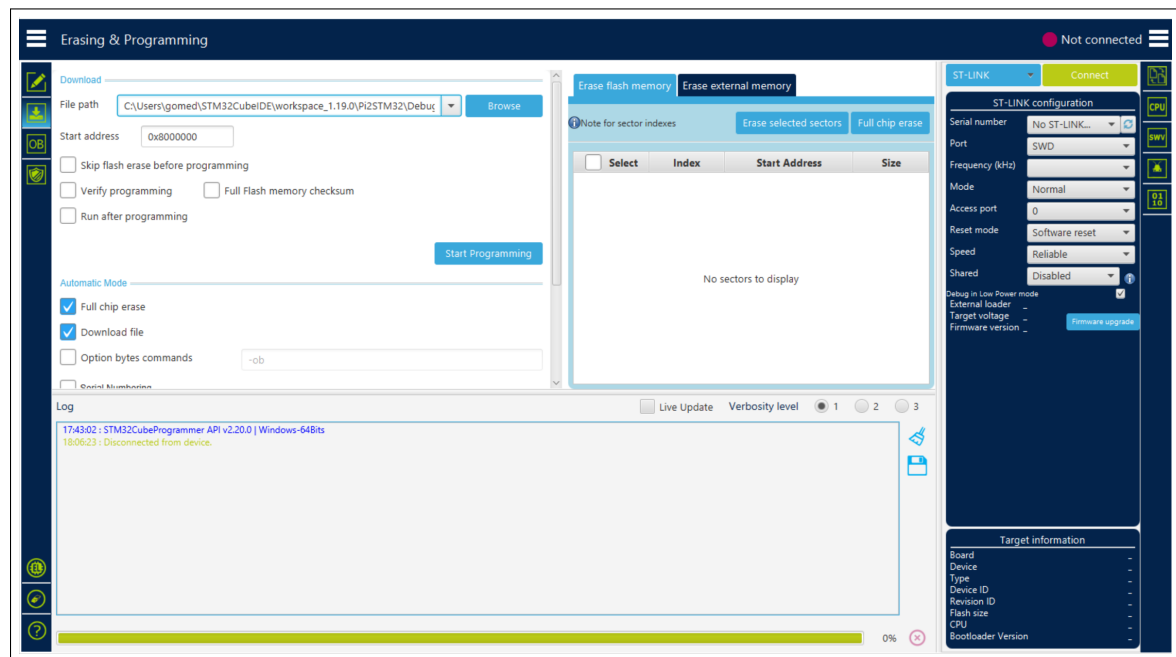


Fig. 72: STM32 Cube Programmer Erasing and Programming Menu

d) Alternative Upload via UART Bootloader

In addition to using the ST-Link interface, we explored programming the STM32 through a USB-to-UART adapter. By setting the BOOT0 jumper to logic high (1) and BOOT1 to low (0), the STM32 entered system memory mode, activating its built-in UART bootloader. This allowed firmware uploading using CubeProgrammer’s “UART” interface. Once programming was complete, BOOT0 was set back to 0 to boot from flash memory normally.

This method provided a reliable backup for firmware uploading, especially useful when debugging ST-Link connectivity issues or when field-programming devices without dedicated ST-Link hardware.

e) Practical Outcomes and Skills Developed

Through extensive hands-on testing and iteration, the following technical proficiencies were achieved:

- Understanding the structure of CubeMX-generated projects and their integration with HAL libraries.
- Configuring, compiling, and flashing firmware for multiple peripherals.
- Diagnosing ST-Link connectivity and bootloader configuration issues.
- Using the UART bootloader as an alternative programming method.
- Debugging using watch variables, breakpoints, and live register inspection.
- Adjusting peripheral timing and clock configurations to achieve consistent system performance.

These competencies not only established the foundation for firmware development within this project but also contributed to a broader understanding of professional embedded systems workflows. Mastering STM32CubeIDE and STM32CubeProgrammer allowed seamless iteration between hardware testing and software refinement, ultimately enabling reliable communication and sensor acquisition systems.

f) Conclusion

Mastering the STM32 development environment proved essential for the success of the project's embedded component. STM32CubeIDE streamlined peripheral configuration and firmware creation, while STM32CubeProgrammer ensured straightforward and verifiable device flashing. Together, they formed a cohesive toolchain that supported the rapid testing and debugging of UART, I²C, and ADC systems. The familiarity gained through this process now enables faster development cycles and more confident integration of future sensors, actuators, and communication modules.

7.12 UART Communication

UART was implemented to provide a simple, unidirectional serial communication channel for transmitting sensor and ADC data from STM32 microcontrollers to the Raspberry Pi. Wiring consisted of STM32 TX (PA9) to Pi RX (GPIO15) and STM32 RX (PA10) to Pi TX (GPIO14), with a shared ground connection. Communication was verified using Python's `pyserial` library and confirmed via serial monitor tools.

During testing, data transmission was consistently successful, demonstrating the STM32's ability to reliably output measurement information and the Raspberry Pi's capacity to read incoming data streams without packet loss.

7.13 I²C Communication

The Inter-Integrated Circuit (I²C) protocol will serve as the primary method of communication between the STM32 Blue Pill microcontrollers and the Raspberry Pi system controller. This serial communication interface was selected due to its simplicity, efficient use of wiring, and suitability for master-slave communication in embedded applications. In this project, the Raspberry Pi will function as the I²C master, while the STM32 microcontroller will act as the slave device responsible for transmitting sensor data and receiving control instructions. The programming, configuration, and debugging of this interface will be carried out within the STM32CubeIDE environment, using the hardware

abstraction layer (HAL) generated by STM32CubeMX.

a) STM32CubeIDE and STM32CubeMX Configuration

The configuration process will begin in STM32CubeMX, which is integrated into STM32CubeIDE. This graphical interface allows the developer to visually assign I²C functionality to the appropriate pins—typically PB6 (SCL) and PB7 (SDA) for I²C1 on the STM32 Blue Pill. Once the peripheral is enabled, STM32CubeMX automatically generates the corresponding HAL initialization code, which is responsible for configuring the clock source, enabling GPIO alternate functions, and setting timing parameters based on the selected bus frequency. The initialization code typically resides in the `main.c` and `i2c.c` files.

In addition to hardware configuration, STM32CubeMX allows users to select I²C addressing modes (7-bit or 10-bit), enable interrupts, and define the microcontroller's own slave address. The generated code leverages functions such as `HAL_I2C_Init()`, `HAL_I2C_Slave_Transmit()`, and `HAL_I2C_Slave_Receive()` from the STM32 HAL library [59]. These functions abstract away the low-level register manipulation, providing a clean and maintainable interface for application-level programming.

b) Programming Methodology and Planned Implementation

Following configuration, the STM32 firmware will be written in C using STM32CubeIDE's built-in editor and compiler toolchain (based on GNU Arm Embedded). The primary loop in the program will continuously listen for I²C commands from the master Raspberry Pi. When a command is received, the STM32 will interpret the message, perform the requested operation (such as reading an ADC value or activating a motor driver), and respond with an acknowledgment or data packet.

The I²C interrupt and event callbacks provided by the HAL—such as

`HAL_I2C_SlaveRxCpltCallback()` and `HAL_I2C_SlaveTxCpltCallback()`—will be used to handle asynchronous communication events without blocking the main execution flow. This non-blocking approach ensures that time-critical peripherals, such as timers and ADCs, remain operational during ongoing communication.

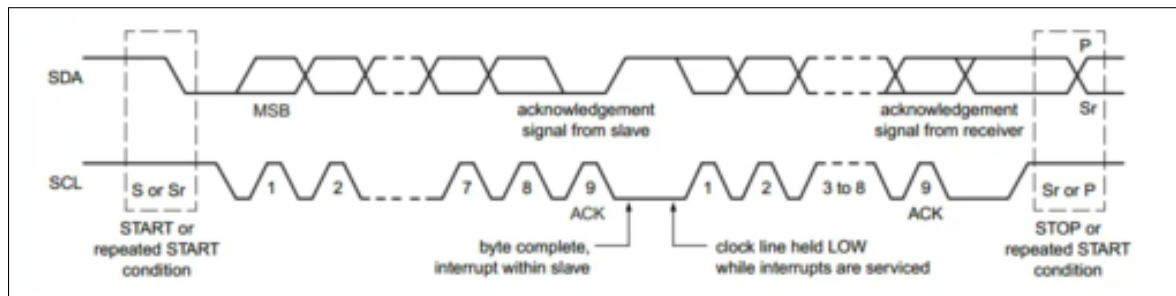


Fig. 73: An I2C Frame

The expected communication workflow is as follows:

- 1) The Raspberry Pi (master) issues a start condition and sends the 7-bit address of the target STM32.
- 2) The STM32 (slave) detects its address on the bus and acknowledges the master.
- 3) The master sends a command byte indicating the requested operation.
- 4) The STM32 interprets this command and either performs a read or write operation accordingly.
- 5) The STM32 responds with data (for example, a sensor reading) using `HAL_I2C_Slave_Transmit()`.
- 6) The master terminates communication by generating a stop condition.

c) Verification and Debugging Process

Once the firmware is compiled, the binary will be flashed onto the STM32 using STM32Cube Programmer. During development, the `HAL_I2C_GetError()` function will be used to diagnose transmission errors such as `HAL_I2C_ERROR_AF` (acknowledge failure) or `HAL_I2C_ERROR_TIMEOUT`. These return codes are essential for ensuring reliable bus operation. On the Raspberry Pi side, Linux utilities such as `i2cdetect`, `i2cdump`, and `i2cget` will be employed to confirm address detection and data consistency [23].

Additionally, logic analyzers or oscilloscopes will be used to verify bus timing, signal integrity, and voltage levels, ensuring compliance with I²C electrical standards. Each transaction will be logged and analyzed to confirm correct start/stop conditions and acknowledgments. This will help isolate potential issues such as bus contention, noise, or incorrect addressing.

d) Future Integration and Data Flow

Once the I²C layer is validated, it will be integrated into the broader system framework where each STM32 microcontroller reports measurement data to the Raspberry Pi. The Pi will then aggregate this information, process it through Python scripts, and generate corresponding CSV files for analysis. This hierarchical communication model ensures modularity—each STM32 operates semi-independently while the Pi maintains central coordination and data logging.

e) Summary

In summary, the STM32CubeIDE and STM32CubeMX environments will facilitate a structured workflow for implementing I²C communication. The HAL-based approach abstracts complex register-level programming, improving code reliability and portability. This integration will establish a robust communication link between the STM32 Blue Pill modules and the Raspberry Pi, enabling synchronized operation across all mechanical and sensor subsystems in the machine.

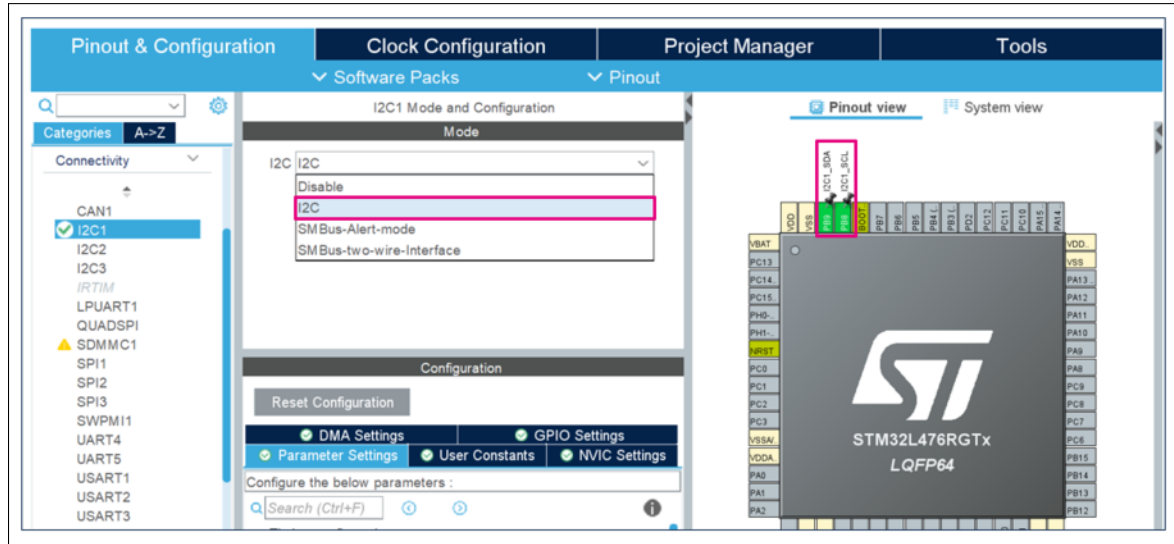


Fig. 74: Configuration of I2C in STM32CubeIDE

7.2 Integration

7.2.1 Raspberry Pi System Setup

The Raspberry Pi will serve as the high-level system controller responsible for orchestrating communication, data collection, and task coordination in the optical testing machine. In this architecture, the Pi will manage communication with multiple STM32 Blue Pill microcontrollers through UART and I²C interfaces, execute data processing tasks using Python scripts, and generate CSV files summarizing test results. This subsection outlines the planned setup, software environment configuration, and operational flow of the Raspberry Pi in the context of this project.

a) Overview of the Raspberry Pi's Role

The Raspberry Pi acts as a supervisory control node within the distributed embedded system. Its main responsibilities include:

- Initiating and managing communication sessions with STM32 subsystems.
- Collecting and aggregating sensor and actuator data for analysis.
- Performing light processing and data validation tasks.
- Storing results locally and exporting them as CSV files.
- Optionally transmitting the results over a network for documentation or further analysis.

This role requires a combination of hardware interfacing, real-time data handling, and operating system-level task management, all achievable through the Linux-based environment that underpins the Raspberry Pi platform.

b) Operating System Installation and Configuration

The setup process begins with installing *Raspberry Pi OS (64-bit)* on a microSD card using the Raspberry Pi Imager utility [52]. Raspberry Pi OS was selected for its compatibility, wide community support, and integrated toolchain for Python-based development. During the imaging process, the hostname, Wi-Fi credentials, and SSH access are pre-configured to facilitate headless operation. This allows the Pi to be accessed remotely over a local network using Secure Shell (SSH) commands, enabling developers to work from another system without requiring a dedicated monitor or keyboard.

Once booted, system updates will be performed using:

```
sudo apt update && sudo apt upgrade -y
```

This ensures that all installed packages, firmware, and kernel modules are current. Peripheral interfaces, such as UART and I²C, are then enabled via the `raspi-config` utility under *Interface Options*. The proper configuration of these interfaces is critical to ensure communication compatibility with STM32 devices.

c) Software Environment and Dependencies

To execute its coordination and data-processing tasks, the Raspberry Pi will utilize a Python-based software stack. Python 3.11 (or later) will serve as the primary scripting language due to its extensive library support and ease of integration with hardware-level communication. The following packages will be installed:

- `pyserial` — to handle UART communication with STM32 devices.
- `smbus2` — for I²C master communication.
- `pandas` — for structured data manipulation and CSV file generation.
- `openpyxl` — for potential Excel file export.
- `picamera2` — for camera-based imaging tasks.
- `opencv-python` — for image capture and analysis in future testing stages.

These libraries will be installed via the Python package manager:

```
sudo apt install python3-pip
pip install pyserial smbus2 pandas openpyxl
opencv-python picamera2
```

The inclusion of these modules allows the Pi to function as both a communication hub and a lightweight data-processing workstation [51], [63].

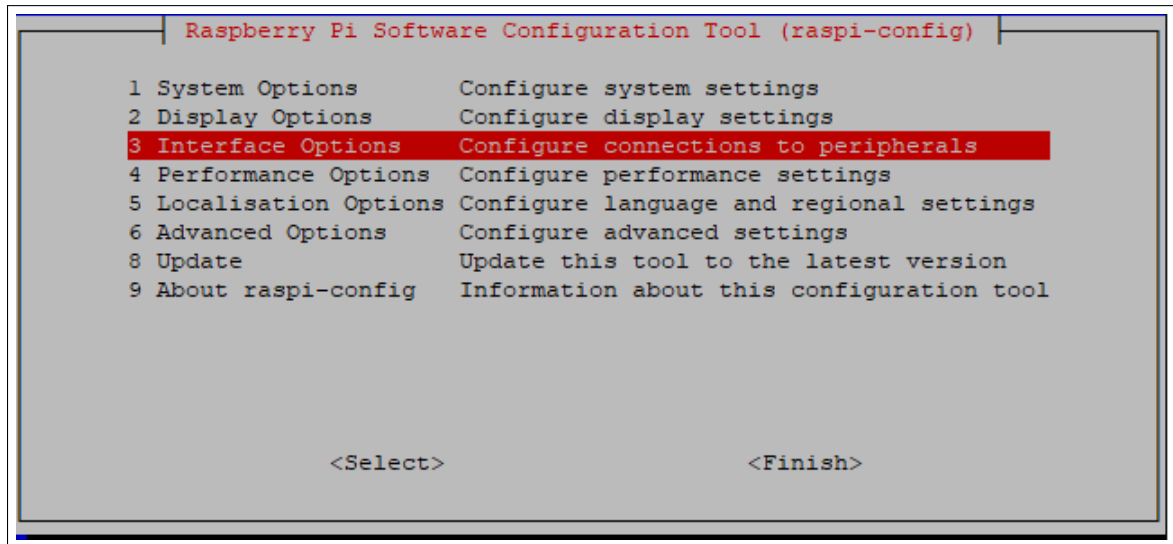


Fig. 75: Raspberry Pi Configuration Tool Menu

d) *Peripheral Interface Configuration: I²C, UART, and Serial Communication*

In order for the Raspberry Pi to communicate effectively with the STM32 microcontrollers, several onboard interfaces must be enabled and configured. These include the I²C bus, the UART serial interface, and the general serial console. This configuration is performed through the command-line utility `sudo raspi-config`, which provides access to various system-level settings.

To begin, the user enters:

```
sudo raspi-config
```

From the menu, the path *Interface Options* is selected, revealing several communication-related toggles. The following settings are adjusted:

- **I²C:** Enabled to allow the Raspberry Pi to function as an I²C master device. This activates the corresponding SDA (GPIO 2) and SCL (GPIO 3) pins, making them available for communication with slave devices such as the STM32. Once enabled, the user must verify the activation by checking for the appropriate device file:

```
ls /dev/i2c-1
```

If the output lists `/dev/i2c-1`, the interface has been successfully enabled.

- **Serial Port (UART):** The Pi's primary UART, mapped to GPIO 14 (TXD) and GPIO 15 (RXD), must be enabled for serial communication. However, the default configuration uses this UART as a Linux login console. To repurpose it for communication with external microcontrollers, the serial console must be disabled while keeping the serial port hardware enabled. In `raspi-config`, this is achieved by

selecting *Interface Options* → *Serial Port* → *No* when asked if the login shell should be accessible over serial, and *Yes* when asked to enable the serial port hardware.

After making these changes, the configuration utility prompts for a reboot to apply the modifications. Once the system restarts, the user can verify each interface using the following commands:

```
ls /dev/serial*  
ls /dev/i2c*
```

If the expected devices (`/dev/serial0`, `/dev/ttyAMA0`, `/dev/i2c-1`) appear, the interfaces are properly configured. It is also recommended to confirm that the necessary kernel modules (`i2c-dev`, `i2c-bcm2835`) are loaded automatically at startup. This can be verified using:

```
lsmod | grep i2c
```

To simplify I²C device detection and debugging, the `i2c-tools` package can be installed via:

```
sudo apt install i2c-tools
```

Once installed, the following command scans for connected I²C devices:

```
sudo i2cdetect -y 1
```

A grid-like output will appear, displaying the hexadecimal addresses of any devices detected on the bus. This is particularly useful for verifying the STM32's I²C slave address and ensuring reliable electrical connections. Proper configuration of these interfaces is a prerequisite for the communication routines described in later sections [52], [39].

e) Directory Structure and File Management

The project directory structure on the Raspberry Pi will follow a modular organization to maintain clarity and ease of debugging. The proposed structure is as follows:

```
/home/pi/ALPM/  
    main_controller.py  
    data/
```

```
raw/  
processed/  
csv_output/  
logs/  
stm32/  
    uart_handler.py  
    i2c_handler.py  
config/  
    system_settings.json  
    device_addresses.txt
```

This directory hierarchy ensures separation of concerns, where communication scripts, configuration files, and output data are stored independently. The system will also generate log files during runtime to track communication status, time stamps, and error messages, aiding in long-term debugging and reproducibility.

f) Task Execution Framework

Once the environment and dependencies are configured, the Raspberry Pi will execute its tasks through a layered control flow. The main control script, `main_controller.py`, will act as the entry point. This script will initialize all communication channels, perform system diagnostics, and begin periodic data acquisition from the STM32 microcontrollers.

Each communication interface will run as a dedicated Python process or thread:

- The UART thread will listen for asynchronous messages, such as error codes or ADC readings, from STM32 boards.
- The I²C thread will periodically poll devices for status updates or sensor measurements.

To prevent timing conflicts, Python's built-in `threading` or `asyncio` modules will be used to manage concurrent operations. Each thread will write its results to a shared buffer, which will later be processed into structured CSV format. This multi-threaded approach ensures that no communication channel blocks another, maintaining responsiveness and real-time data integrity.

g) Data Handling and Storage

The data received from the STM32 devices will initially be stored in temporary memory buffers. Periodically, these buffers will be converted into Pandas DataFrames for easier manipulation and formatting. Each DataFrame will represent a specific testing cycle, containing key variables such as sensor ID, measurement time, raw ADC value, and converted physical units.

Processed data will then be exported as CSV files and saved under the `/data/csv_output/` directory. The naming convention for each file will follow the pattern:

measurement_YYYYMMDD_HHMMSS.csv

This timestamp-based organization facilitates traceability and version control, ensuring that all measurement sessions can be easily referenced.

h) System Monitoring and Automation

For system reliability and long-term testing, the Raspberry Pi will employ automation scripts and background processes. The `cron` scheduling service available in Linux will be used to execute specific tasks at predefined intervals, such as automatic startup of the control script upon boot or periodic backups of the CSV files to an external storage device.

Additionally, diagnostic scripts will monitor CPU temperature, memory usage, and peripheral connectivity. If communication errors occur, the system will attempt automatic recovery by reinitializing the relevant interface. All such events will be logged in real-time under the `/logs/` directory.

i) Summary

The Raspberry Pi setup process will establish a robust and flexible control platform for the optical testing machine. By combining Linux-based multitasking, Python scripting, and standardized communication protocols, the Pi will coordinate real-time data acquisition and system management. The structured workflow described in this section ensures modularity, scalability, and reliability—key factors in the success of this embedded system integration.

7.22 Software Integration and System Coordination

The integration of software components across the Raspberry Pi and STM32 Blue Pill microcontrollers will form the backbone of the Automatic Lens Parameter Metrology system. In the final design, the STM32 boards will handle low-level sensing, actuation, and timing-sensitive operations, while the Raspberry Pi will act as the supervisory controller responsible for high-level coordination, image capture, and data management. This distributed architecture is intended to ensure that each device operates within its strengths: deterministic real-time control on the STM32 and complex computation, communication, and data logging on the Raspberry Pi.

a) System-Level Overview

The planned integration will follow a modular hierarchy consisting of three primary layers:

- 1) **Low-Level Control Layer (STM32):** This layer will handle direct hardware interaction—reading ADC values, driving motors, controlling servos, and collecting sensor data. Firmware will be developed using STM32CubeIDE with HAL libraries

for UART, I²C, and GPIO management.

- 2) **Communication Layer (UART / I²C):** This layer will provide bidirectional data exchange between the STM32 and Raspberry Pi. UART will be used for high-throughput streaming (e.g., sensor logs or debugging), while I²C will be used for command-driven interactions where the Pi acts as master and STM32 as slave.
- 3) **Supervisory Layer (Raspberry Pi):** The Raspberry Pi will execute Python-based orchestration scripts to coordinate tasks such as tray extension, optical scanning, lens centering, and data storage. Libraries such as `pyserial`, `smbus2`, `pandas`, and `picamera2` will be used for communication, I/O, and image processing.

Each layer will communicate through defined APIs and serial protocols, making the system extensible and adaptable to future hardware changes. The following subsections describe the planned design methodology, data flow, and synchronization strategies for integration.

b) Design of the Communication Protocol

Communication between the STM32 microcontrollers and the Raspberry Pi will be established through simple, structured message protocols. UART will be selected for continuous data streams, such as ADC readings or status updates, while I²C will provide command-response style transactions for precise control.

UART Communication:

The STM32 will transmit ASCII-formatted data strings using the HAL function

`HAL_UART_Transmit()`, while the Raspberry Pi will receive and decode messages via the `pyserial` library [?]. Each message will follow a standardized frame:

```
<START><COMMAND>:<DATA><END>
Example: <SENSOR>:1023<\n>
```

This format will allow the Pi to parse messages using regular expressions or line-based processing, ensuring synchronization even when multiple sensors are polled rapidly. The Pi will also be able to transmit control commands back to the STM32 through the same UART channel, such as “SET_MOTOR_SPEED:100” or “START_SCAN”.

I²C Communication:

The I²C interface will supplement UART by providing deterministic, byte-level transactions. The Pi, configured as master, will use the `smbus2` library [?] to send command codes to the STM32, which will run in slave mode using `HAL_I2C_Slave_Transmit()` and `HAL_I2C_Slave_Receive()` routines. This interface will be particularly useful

for triggering time-critical actions or retrieving structured status packets. For example, a “READ_STATUS” command could be issued, and the STM32 would respond with a data buffer representing system temperature, voltage, or actuator state.

The dual-protocol approach (UART + I²C) is expected to ensure fault tolerance—if UART experiences interference, I²C can continue exchanging critical control messages.

c) Software Architecture on the Raspberry Pi

The Raspberry Pi will serve as the central controller, integrating high-level functions written primarily in Python. The software architecture will consist of multiple modules communicating through asynchronous event loops:

- **Communication Manager:** Will handle UART and I²C data exchange, continuously monitoring serial ports and dispatching incoming messages to corresponding handlers.
- **Task Scheduler:** Will implement timed control loops to coordinate optical tests, ensuring that operations such as lens centering, scanning, and imaging occur in sequence.
- **Image Acquisition Module:** Will use the `PiCamera2` library [?] to capture high-resolution images during testing. Image metadata (such as exposure time and lens ID) will be appended to each frame.
- **Data Processing and Logging:** Will employ `pandas` and `openpyxl` [?] for storing and exporting results in CSV or Excel format. Each measurement cycle will log motor position, captured intensity, and timestamped results.
- **User Interface Layer:** A planned module that will provide a graphical or web-based interface to visualize test progress and review results remotely via Wi-Fi.

The Raspberry Pi software will be designed with scalability in mind, supporting concurrent control of multiple STM32 nodes. This architecture will be modeled after a simplified ROS (Robot Operating System)-like pattern, where each STM32 will act as a “node” publishing and subscribing to specific data topics.

d) Firmware Integration on the STM32

On the STM32 side, firmware integration will center on the coordination of multiple peripherals: UART, I²C, ADC, and GPIO. Each peripheral will be initialized via STM32CubeMX-generated code and managed within the `while(1)` main loop using interrupt-driven or polling mechanisms.

To facilitate communication and integration:

- The UART interrupt handler will be used to receive commands asynchronously.
- The ADC sampling function will continuously convert analog sensor values (such as potentiometer readings) to 12-bit digital results.
- The I²C callbacks will handle incoming master requests, preparing buffers with current system status or sensor data.

This modular firmware structure will ensure clean separation of responsibilities, making it easier to integrate new hardware (such as additional sensors or actuators) with minimal modification to existing code.

e) Synchronization and Data Flow

A critical part of the integration will involve synchronizing data exchange between asynchronous systems. The Raspberry Pi's Linux-based environment operates on a multitasking scheduler, whereas the STM32 executes firmware in a single-threaded loop or interrupt context. To manage this difference, a deterministic handshake mechanism will be implemented:

- 1) The Pi will issue a command (e.g., "START_MEASURE") over I²C.
- 2) The STM32 will acknowledge receipt by setting a status flag and executing the corresponding routine.
- 3) Once the routine completes (for example, after an ADC measurement), the STM32 will transmit a completion message via UART.
- 4) The Pi will then log the returned data and trigger the next phase (such as image capture).

This synchronized loop is expected to ensure reliable coordination between hardware actions and software operations, minimizing timing conflicts or data loss. The mechanism will later be validated through UART and I²C communication tests.

f) Error Handling and Debugging Framework

During the integration phase, several error-handling strategies will be implemented:

- **Checksum Verification:** Each UART message will include a simple checksum to detect data corruption.
- **Timeout Recovery:** The Pi will monitor for lost responses and will resend commands after a defined interval.
- **Debug UART Channel:** A secondary UART output will provide debug messages (e.g., "ADC Ready" or "I²C Event Triggered") for development visibility.
- **Logging System:** All communication transactions will be logged locally on the Pi for later review.

These methods are expected to ensure robust integration between hardware and software, enabling consistent operation even under intermittent noise or hardware resets.

g) Future Expansion and Network Integration

The modular structure of this integration plan will allow for future upgrades to include network connectivity and cloud data synchronization. The Raspberry Pi's built-in Wi-Fi capabilities will enable remote access, live data streaming, and integration with cloud-based dashboards. Python libraries such as `requests` or `paho-mqtt` could be used to push measurement data to online servers or IoT frameworks [?].

Additionally, the architecture will support scaling the system to control multiple STM32 nodes simultaneously via I²C addressing or UART multiplexing. This will enable multiple measurement stations to operate in parallel under centralized Pi supervision.

h) Conclusion

The planned integration of software components across the STM32 and Raspberry Pi platforms represents a crucial step in this research. By combining low-level firmware control with high-level orchestration, the system is expected to achieve both precision and flexibility. The modular communication framework, structured data handling, and robust synchronization methods will ensure that the machine operates autonomously, performs repeatable measurements, and logs results reliably. These integration principles will not only complete the current phase of development but also establish a foundation for scalable, network-enabled future versions of the system.

7.23 CSV File Generation and Data Management

A key functional component of the Automated Lens Parameter Metrology system software architecture will involve the generation, organization, and storage of measurement data in Comma-Separated Value (CSV) format. CSV files will be selected as the primary data output medium because of their portability, simplicity, and compatibility with a wide range of scientific and engineering tools such as Microsoft Excel, MATLAB, and Python-based data analytics frameworks. This section describes the planned methodology for data collection, structuring, and CSV file generation within the integrated Raspberry Pi and STM32 control environment.

a) Purpose and Design Considerations

The CSV generation system will be designed to ensure that every test cycle performed by the machine produces a well-structured dataset containing sensor readings, positional information, and optical measurements. Each row within the CSV file will represent a single data acquisition event, and each column will correspond to a relevant parameter such as timestamp, motor position, light intensity, and image reference identifier.

The rationale for adopting CSV files instead of more complex database systems stems from the need for immediate data readability and compatibility with both local and remote analysis workflows. CSV files are human-readable, require no specialized database server, and are supported natively by most spreadsheet and data visualization programs. For a project that emphasizes experimental testing and iterative analysis, CSV output will enable team members to rapidly visualize and compare measurement results during system validation.

b) Data Acquisition Workflow

The process of generating a CSV file will begin with the Raspberry Pi orchestrating a complete measurement cycle through UART and I²C communication with the STM32 boards. The Pi will act as the master controller, initiating data collection routines such

as lens alignment, light scanning, or ADC sampling on the STM32 nodes. Once these operations are completed, the STM32 will transmit sensor readings back to the Pi, where they will be organized into structured data entries.

To ensure consistency, the Raspberry Pi software will utilize an internal buffering mechanism that temporarily stores all data within memory before committing it to a CSV file. This strategy will help prevent incomplete file writes and avoid corruption in the event of communication interruptions. Each buffered entry will contain a timestamp generated by the Pi's system clock, ensuring that every data point can later be synchronized with corresponding images or hardware events. The timestamping method will rely on the Linux `datetime` utilities integrated within Python's runtime environment [?].

c) Implementation Strategy

CSV file creation and management will be implemented entirely on the Raspberry Pi using high-level Python data libraries. The `pandas` library will serve as the primary data management tool, offering a flexible and efficient means of handling tabular datasets [?]. During operation, the system will continuously append new rows to a `pandas DataFrame` object, which will represent the in-memory collection of all ongoing measurements.

Upon completion of each testing session, the accumulated `DataFrame` will be written to disk as a CSV file using the `to_csv()` method provided by `pandas`. This operation will automatically handle column naming, text encoding, and file formatting according to standard UTF-8 specifications. If the system is required to generate Excel-compatible outputs for advanced documentation or report inclusion, the `openpyxl` library will be used in conjunction with `pandas` to export data in the XLSX format [?]. However, CSV files will remain the default due to their smaller file size and easier version control integration.

To prevent data overwriting, each CSV file will be named dynamically using a timestamp-based naming convention, such as:

```
TestSession_2025-05-23_15-42-10.csv
```

This method will ensure that each session's data remains distinct and traceable, while also enabling automated archival or cloud upload procedures in the future.

d) Error Handling and Data Integrity

Maintaining data integrity during CSV generation will be an essential consideration. To achieve this, several mechanisms will be implemented:

- **Transaction Logging:** Every write operation will be preceded by a log entry noting the number of data points and the file path. This will allow quick validation of whether the data was successfully saved.
- **Temporary File Buffers:** Data will first be written to a temporary file before being renamed to its final CSV filename. This will reduce the risk of partial file corruption

during unexpected shutdowns.

- **Checksum Verification:** The system will compute a basic checksum or file size comparison after each write operation to verify integrity.
- **Redundant Backups:** CSV files will be mirrored to a backup directory on the Raspberry Pi's SD card, providing protection against file loss or corruption.

In addition, periodic validation tests will be performed during integration to ensure that CSV outputs conform to the expected schema and that no data fields are misaligned. These validation routines will assist in identifying anomalies resulting from communication delays, data truncation, or software timing mismatches between the STM32 and Raspberry Pi.

e) Integration with Image and Measurement Data

As the Automated Lens Parameter Metrology system will also capture images of each lens or optical component, the CSV file generation module will include functionality to link each measurement entry with a corresponding image file. The `Picamera2` library will be used for image acquisition, and its metadata interface will allow image filenames and timestamps to be directly recorded in the same dataset [?]. This integrated approach will ensure that both numerical and visual data can later be correlated during post-processing or report generation.

7.3 Software Block Diagram

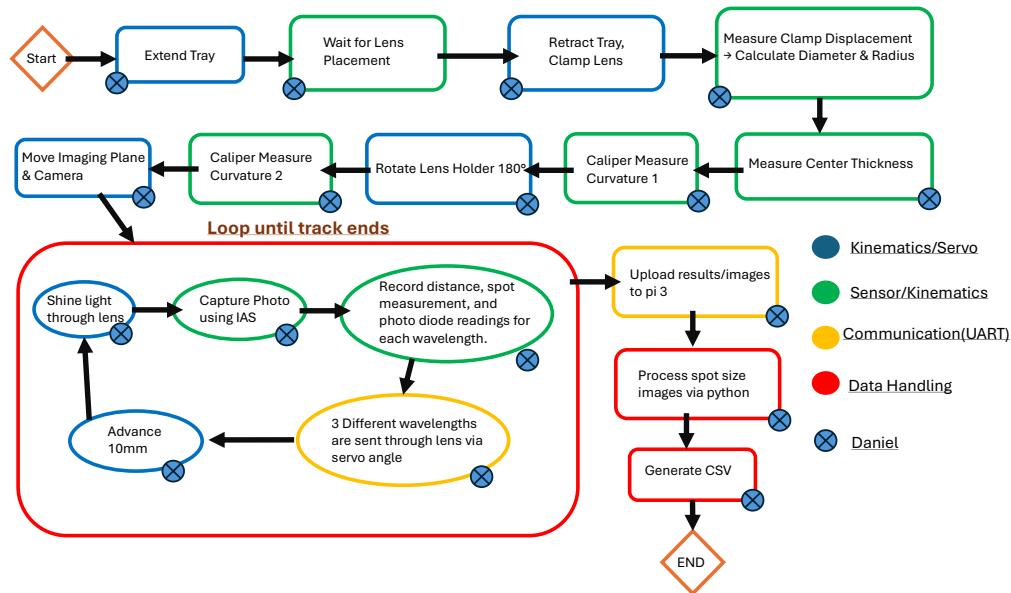


Fig. 76: Software Block Diagram.

7.4 Datasheet Generation

The CSV file plays an important role in saving data however is difficult to discern info at a glance. One of our advanced goals for this project is to present the data in a industry standard styled datasheet.

To do so a work flow was implemented to turn CSV data into formatted PDF. First each individual figure must be created, for this we use raw SVG code that is edited via regex and then rendered out as a pdf. This makes it incredibly easy to iterate, a template can be created in standard vector editing software and then altered to specification after the fact.

Those figures and other data are then passed by regex to a LaTeX source file. LaTeX is a coding language/ecosystem which allows the generation of complex PDFs in a coding framework. The altered source file can then be generated and displayed.

This workflow allows the use of tools that our team are familiar with, multiple stages to catch bugs, and puts everything together at the end to present the user with a beautiful and legible datasheet.

8 Chapter 8 System Fabrication

8.1 Optics

8.11 CAD Considerations

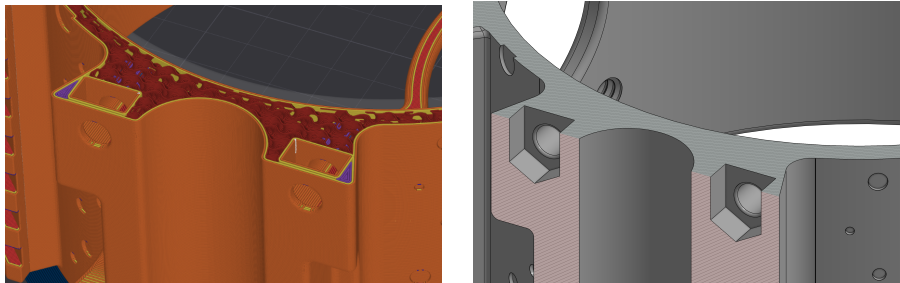
This projects has leveraged the recent advancements in prototyping and manufacturing technology. For the most part fabrication has involved laser-cutters and 3D printers. Specifically this project has used both Fiber and CO2 laser-cutters, and FDM printers such as BambuLab's X1C/P1P or Positrons V3.2 system. This work flow has allowed rapid iteration of parts however comes with important considerations based on the technology used.

When it comes to laser-cutting the restrictions of the medium and tolerances are both really important considerations. Fitting together pieces in a finger-joint style is really powerful however nothing will fit properly if you don't take into consideration the thickness of the material being cut and the kerf of the laser (the effective cut width). To compensate for this we used a combination of parametric design and offsets, that way the thickness can be a set variable and face offsets can be used to take into account outer and inner kerf compensation. From there we leveraged an Onshape feature script known as AutoLayout as well as Kerf Compensation. Laying out and tiling parts for efficient material use is a large time sink in industry, but tools like AutoLayout or DeepNest can automate the process. Lasercutters can only manufacture parts that are either flat or bend

in only one axis (living hinges), as a result we employed lasercut parts for any large paneling or flat features.

3D printing is a powerful technology however the layer by layer nature requires some unusual design constraints. The following is a list of each consideration and reasoning.

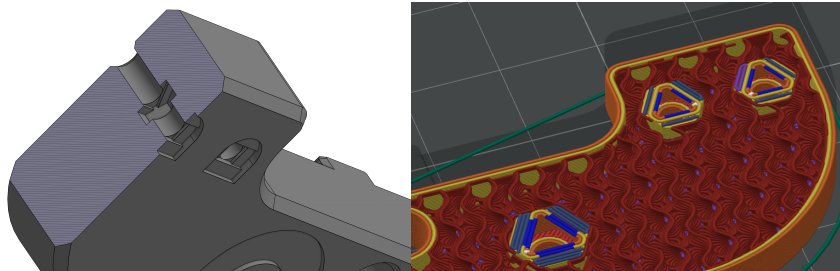
- Tolerances are specific to each 3D printer however the following are safe defaults for most pieces. 0.0mm interference, 0.15mm press, 0.3mm slip fit. These apply on the xy print plane, z plane tolerances should be considered based on layer height, for example a layer height of 0.2mm typically requires a slip fit tolerance of 0.1mm. Note that tolerance in this case defines the offset between faces in this case.
- Print orientation must be considered from the outset, 3D printers excel in the xy plane however any features in the z plane must be around 45° otherwise as the layers stack they may fail. One notable exception is bridges. If bridges are setup such that they form straight lines from point to point then they have a high chance of success.
- Due to print orientation constraints fillets are best used oriented on the xy plane where as chamfers are best on the xz or yz plane. Vertically oriented fillets due to slicing form vertical steps that don't really match the geometry.
- Printing in place is an extremely powerful tool for decreasing build complexity and increasing quality. Print in place is a term used for the process of pausing a print mid way and adding objects into internal cavities. This is particularly useful when adding metal threading to prints. Metal threading improves quality and a print in place nut while larger than a heat set is an excellent and cheaper option in cases where space isn't a concern.



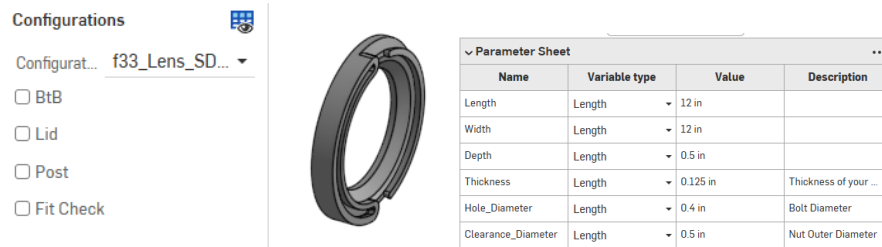
Slicer preview to the left shows how the print must be paused before the enclosing layer, CAD preview to the right shows how clearance must be allowed for object placement.

- Heatsets are small cylindrical objects with "teeth" on the outside and a desired thread on the inside. To place heatsets use a soldering iron and slowly press them into place. Heat sets are good options for when space is a constraint or printing in place an orientation would be impossible. As well printing in place requires "babysitting" of the print, while heatsets can be added after the fact. When designing a hole for a heatset: make the hole 0.2mm smaller than the OD of the heatset, about a layer height deeper than the heatset height, and finally add a 0.5mm chamfer to the edge of the hole to aid in positioning.
- Don't be afraid to do multiple versions, filament is cheap and time is not. Just keep in mind that dimensions can usually be verified without printing the entire device. Use slicer or CAD tools to crop the model to the region of interest. Just remember to maintain the same print orientation.

- Sometimes in cases of models having little to no connection to the bed will they encounter poor bed adhesion. In cases like these CAD custom supports or alternatively rely on slicer features such as rafts or brims.
- Countersunk holes are great addition to make a print more functional and polished, however keep in mind that bridges only work in straight lines. If counter sunk hole is aligned head side down try cutting a polygon such as a triangle or hexagon into the bottom surface to steps. By splitting the circular feature into multiple steps high quality undercuts/hangs can be printed without support.

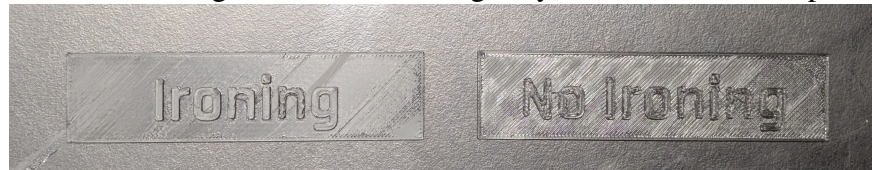


- Square corners aren't real, that the first thing any machinist will tell you, and the same applies to 3D printers. In cases where perfect corners matter try adding what is known as a "dog-bone" these are small cylindrical cut outs that provide clearance from the limitations of print line size.
- Parameters are king, going all the way down to the sketch level to fix repeated dimensions is a waste of time. Instead build parameter tables, variables, or even configuration setups. These will allow the rapid iteration of tolerances and give more flexibility in design. A excellent example is the optical mounts at kivamccr.xyz.

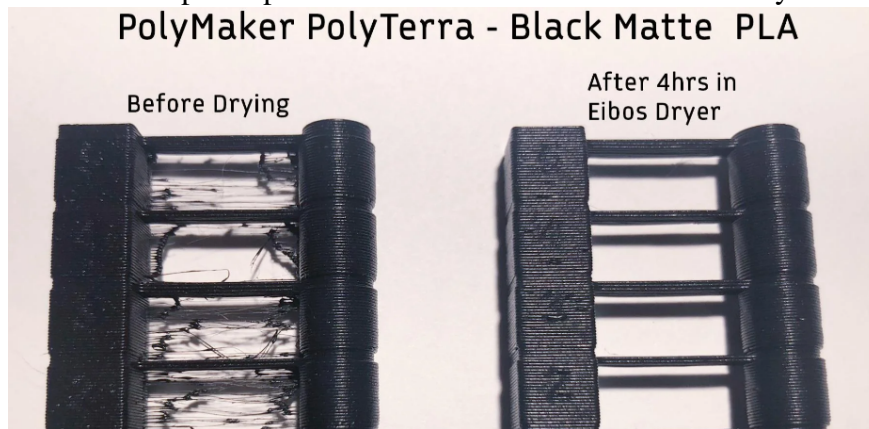


- If something is meant to be gripped/twisted add knurling for a high quality feel. To make knurling easily simply create a square sketch and extrude up to the surface on one side and 60 degree draft on the other. This will create a pyramidal structure, then pattern them across the given surface. If the dimensions may change in the future then use variables to extract values such as circumference and link to patterning calculations.
- Creating a hole oriented on the xz or yz plane? They might end up squished due to the layer by layer slicing. To fix this create ellipses instead, preserve the width but change the height to be about one layer greater in height.
- Less is more, less features and especially for printing– less contours is the name of the game. Creating parts with low feature counts saves time and prevents print failures. What can go wrong will go wrong.
- The extra 5% is the difference between a prototype and a product. Take the 10 minutes to polish the design, its always worth it.

- Doing overhanging chamfers starting on the first layer? Don't! Elephants foot is the term for the first layer extra squishing and dimensional inaccuracy due to the heat of the bed reducing cooling potential. Instead have a layer height or two of flat before the chamfer and it'll look way better. Though this may not be a concern on a properly calibrated printer.
- Ironing is an excellent feature and should be used as much as possible. Doing so will cause prints to be way higher quality and with smoother surfaces. Do keep in mind that for certain angles or fillets ironing may hurt more than help.



- Infill is lighter than isogrids. 3D printing slicing software does a remarkably good job in preserving strength, adding isogrids will increase wall count and ultimately weight.
- Print doesn't seem quite right? Seeing stringing or hearing pops? Your filament is wet! Dry the filament in a enclosed printer or in an oven. Be mindful of temperature. Different materials pick up moisture faster than others so check your datasheets.



[71]

- When choosing filaments the following is a good rule of thumb: PLA for regular printing, ABS for springing, TPU for rubbery, PETG for chemicals and durability, HT-PLA for temperature, and ASA for weatherproofing.
- Be wary of glass or carbon fiber filled filaments, they are highly abrasive and will destroy nozzles that aren't compatible. As well there is very little research on their handling safety and people have found fibers sticking in their hands after handling. As well make sure to wear a mask when sanding these filaments.

8.12 White Light Source

The WLS will be physically integrated using the magnetic feet and relay control. The magnetic feet allow extra indexing into a holder so the WLS can be removed and reintegrated as needed. The WLS also features an internal relay and external spring connectors, this makes it incredibly easy to turn on and off the WLS via an electrical input. This is really valuable because Xenon bulbs produce a lot of heat, so turning off the bulb between runs saves on energy and bulb lifespan. The light going to the monochromator

will be a 10mm beam however this must be further compressed on a single axis in order to achieve a lower full width half max, to do so we may integrate a cylindrical lens beam compressor.

8.13 Monochromator

The monochromator will begin with an aperture made out of sheet copper. This will make sure that any stray light that might be coming from the WLS or outside is not included. We will be arranging the mirrors using the custom breadboard-like function of the optical post holders shown in Fig 33. This is done by slotting standard wooden pencils through the hexagonal holes near the bottom of the posts. The pencils will act as rods to keep the mirrors fixed at the distances and angles we choose. The post holders will then be attached to a base and fixed to the rail along the optical axis. By only permanently attaching the post holders and not the posts themselves, we will be able to easily adjust or replace parts without compromising the alignment of the monochromator.

The other two apertures will also be made out of sheet copper and be mounted on their own optical posts and post holders to maintain the flexibility of the design.

The diffraction gratings cannot be fixed in place by the wooden pencils in the same way because they will need to rotate. Instead, they will be mounted on a pair of stepper motors that move in sync with each other. These motors only need to be coded once for the both of them because they will move the same angular distances, just in opposite directions to account for the geometry of the monochromator. These motors are anticipated to only have to rotate up to 20° total, as this is the size of the angle of refraction (β in Equation 3) for a 1200 lines/mm diffraction grating. These motors will be connected to the optical post holders that carry the diffraction gratings via custom-modeled and printed connectors.

8.14 Photodetectors

The photodetectors placed after the monochromator will be mounted differently for each type of spectra.

The reflective photodiode will be fixed in place about 3cm off of the optical axis so that it doesn't block the incoming light from the exit slit of the monochromator. In order to meet it, the test lens will be rotated by its mount by a stepper motor in a similar fashion to the diffraction gratings in the monochromator. The lens will rotate until the reflective photodiode detects a sufficient amount of light power incident upon it, at which point it will begin to send power data at each of our test wavelengths.

The transmissive photodiode will be mounted on a small rail normal to the optical axis so that it can slide into and out of place from the side. This movement will also be provided by a stepper motor. Once in place, the photodiode will send power data at each of our test wavelengths. Once this is completed, the motor will move the photodiode along the rail so it is no longer on the optical axis and no longer blocks the screen for the IAS.

Both of these photodiodes must be wired to the PCB, so their mounts will custom-modeled

and printed to accommodate for electronics adjustments and maintenance.

8.15 IAS

In order to allow the movement of the IAS on the linear gantry, we chose to use a 2 meter long raspberry pi cable. That way video can be transmitted from the mounted IAS during operation. The IAS will move approximately half a meter during operation. This large range enables the focal length extraction process.

8.2 Electronics

8.21 Solder Choice



Fig. 77: Solder Paste and Solder

a) SAC305 Lead-Free Solder Wire

For hand soldering, I used a SAC305 lead-free alloy composed of 96.5 percent tin, 3.0 percent silver, and 0.5 percent copper. This alloy is widely used in RoHS-compliant electronics because it provides strong mechanical joints, good thermal fatigue resistance, and stable electrical performance. The addition of silver improves joint strength by refining the microstructure, while copper helps stabilize the intermetallic layer that forms

on copper PCB pads. The 0.81 mm wire diameter provides controlled solder feed for through-hole and medium-pitch components, and the integrated no-clean flux core ensures reliable wetting without requiring post-cleaning. Overall, SAC305 produces high-integrity joints with predictable behavior during both prototyping and production-level assembly.

b) Low-Temperature No-Clean Solder Paste

For surface-mount assembly, I used a low-temperature bismuth-based solder paste with a melting point of 138 degrees Celsius. The reduced reflow temperature greatly decreases thermal stress on sensitive components such as LEDs, MEMS devices, and thin-film passives, and it also minimizes PCB warping and pad lifting during heating. The no-clean flux system maintains activity at lower temperatures while providing stable viscosity and oxidation resistance, resulting in consistent wetting on both ENIG and copper-OSP finishes. Because it allows reflow at much lower thermal budget than conventional SAC alloys, this solder paste is well-suited for fine-pitch SMD components and mixed-technology assemblies where temperature limits are critical.

8.22 Soldering Station Details

a) Use of a Combined Soldering Iron and Hot-Air Rework Station

A dual-function soldering station that includes both a temperature-controlled soldering iron and a hot-air rework tool provides a versatile and controlled platform for assembling and repairing modern electronic hardware. The soldering iron enables precise point-contact heating for through-hole components, connectors, and single-pin work, ensuring strong joints with minimal thermal spread. The hot-air tool, by contrast, delivers uniform convective heating across a wider area, which is essential for surface-mount reflow, removing multi-pin ICs, shrinking heat-sensitive tubing, and reworking fine-pitch packages without mechanical stress. The ability to adjust both airflow and temperature allows the operator to match the thermal profile to the specific solder alloy and component density on the board. Using both tools together significantly improves reliability during assembly, reduces risk of overheating components, and supports effective rework of dense SMD layouts that cannot be serviced with an iron alone.

b) Hot Plate Reflow Setup

A small laboratory hot plate was used as the primary heating source for reflowing the low temperature solder paste. The large flat aluminum surface provides uniform thermal distribution, allowing the entire PCB to heat gradually and evenly without localized hot spots that could stress components. By controlling the plate temperature and monitoring the solder behavior directly, it becomes easy to achieve a repeatable reflow process without the complexity of a dedicated reflow oven. A compact fume extraction fan was positioned adjacent to the hot plate to remove flux vapors produced during heating, improving both visibility of the joints and operator safety. This setup provides a simple, effective, and reliable method for assembling small SMD boards using low temperature solder paste.



Fig. 78: Soldering Tool

c) 3D Printed PCB Holding Station

I designed and 3D printed a custom PCB holding station to provide a stable and ergonomic platform for soldering and inspection. The printed arms and rotating clamps allow the board to be held securely without mechanical stress on any components or connectors, which is especially important when working with small SMD parts. The adjustable rotation of the gripped wheels makes it easy to access both sides of the PCB without repeatedly handling or flexing it, reducing the risk of accidental damage during assembly. The wide base increases stability and keeps the working area rigid during fine soldering operations. Using a printed fixture also allowed full customization of geometry, clearances, and gripping pressure, providing a more component-safe and task-specific solution compared to generic commercial holders.

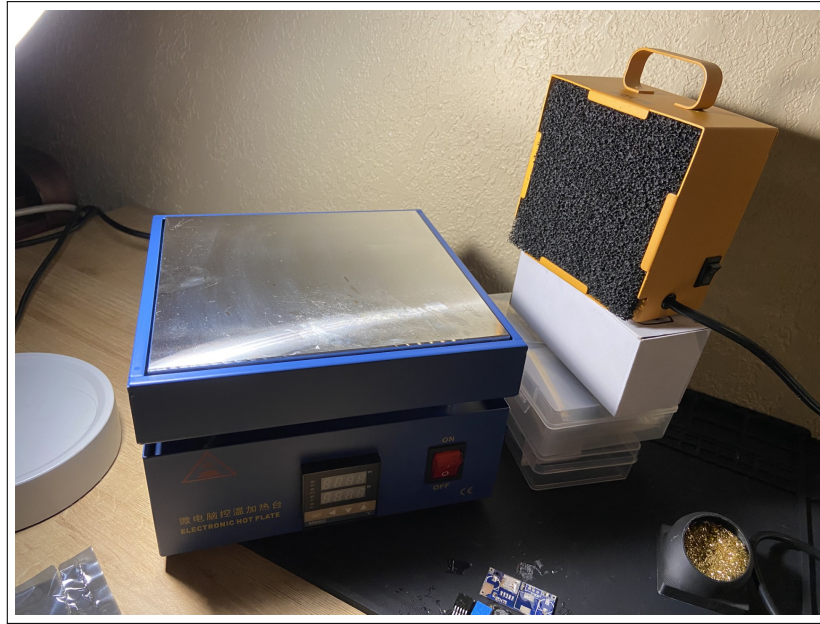


Fig. 79: Hotplate Station

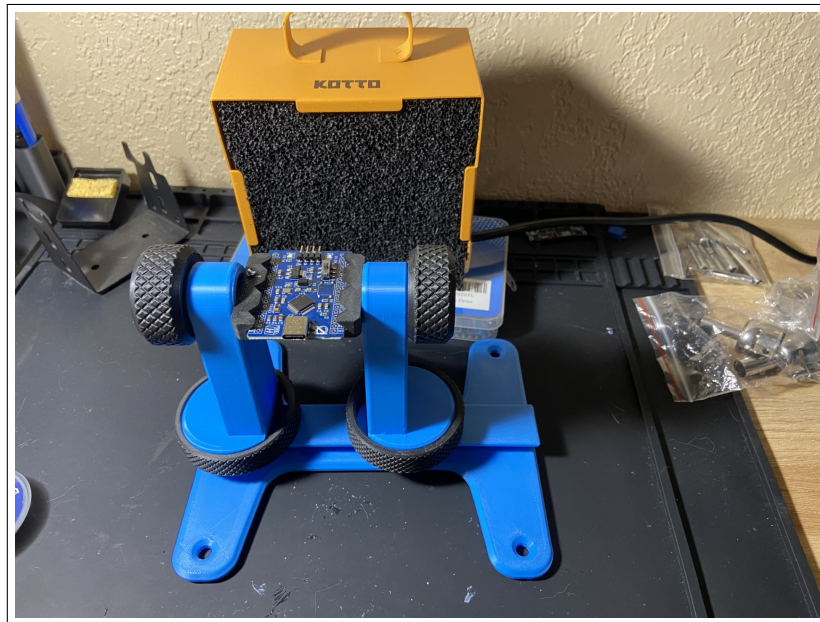


Fig. 80: Holding Station and Fume Fan

d) Use of a Fume Fan and Filter

A compact fume extraction fan with a replaceable filter was used to capture airborne particles and flux vapors generated during soldering. Direct airflow across the work area removes particulate matter and combustion byproducts that would otherwise accumulate near the operator, improving both visibility and safety. The activated filter material traps fumes and reduces inhalation exposure, which is important when working for extended

periods or when using no-clean fluxes that still release irritating vapors under heat.

8.23 STM32 Manufacturing

For STM32 board fabrication, I used JLCPCB's integrated manufacturing workflow, which connects directly to KiCad through the JLCPCB fabrication and assembly add-ons. After completing my schematic and PCB layout in KiCad, I generated the Gerber, BOM, and CPL (pick-and-place) files using the JLCPCB plug-in, which automatically formats the outputs to match their assembly requirements. This integration made it straightforward to preview component footprints, verify polarity markings, and check for part availability in their SMT assembly service. Once uploaded to JLCPCB's web interface, I selected surface-mount assembly for the top side and confirmed all part placements and orientations using their interactive viewer. The manufacturing process itself included two-layer PCB fabrication, solder mask and silkscreen printing, component soldering, and optical inspection before shipment. I chose to have JLCPCB handle the full assembly, meaning the component placement, soldering, and reflow were all completed in-house according to my CPL file, leaving me with a fully populated board that was ready for testing immediately upon arrival.

Figures 81 and 82 illustrate the design and verification of the STM32 control board. Figure 81 shows the final PCB layout in KiCad, featuring the completed routing and component placement optimized for short signal paths and clean power distribution. Figure 82 presents the assembled and powered board, where the illuminated indicator LEDs confirm that the power supply stage and microcontroller are operating correctly, validating the design and successful manufacturing process.

8.3 Software

8.31 Raspberry Pi Data Access and Output Pathways

As the Raspberry Pi becomes the central controller for the Automatic Lens Placement Metrology System, one of the practical questions that naturally arises is: how will users actually see the data that the system produces? The Pi is responsible for coordinating the entire measurement process, communicating with the STM32 microcontrollers, and generating the test results as CSV files. Because of this, it is important for anyone operating the machine to have a clear and straightforward way to view, retrieve, and transfer those files once a test cycle is completed. At this stage, two main options have emerged for how the Pi will present this information: using a small display directly attached to the machine, or allowing users to remotely access the Pi from another computer through SSH.

The first option—and the one currently favored for overall usability—is to mount a compact HDMI display directly onto the machine. Modern Raspberry Pi boards make this quite simple, as they already support HDMI or micro-HDMI output. Using a screen allows the entire system to behave more like a self-contained piece of laboratory equipment.

network (either via Ethernet or Wi-Fi), the operator can log in from anywhere with the correct credentials.

This remote-access model offers several advantages. For example, if the machine is used in an environment where data is regularly analyzed on a particular workstation, then it might be easier for users to retrieve CSV files directly from that workstation using tools like `scp`. The workflow becomes as simple as powering on the machine, letting it run, and then downloading the CSV file when the test is finished. Remote access also reduces the physical hardware attached to the machine, decreasing clutter and minimizing the chance that a screen or cable becomes damaged.

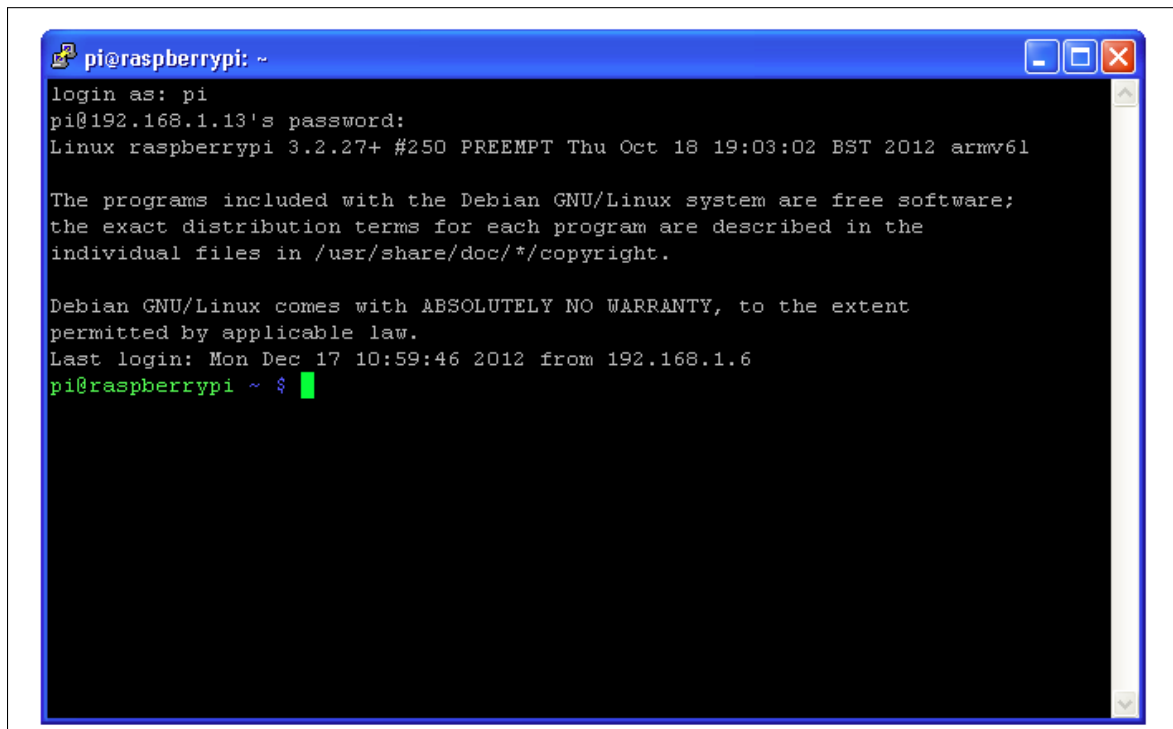


Fig. 83: An example of a terminal connecting to a Pi via ssh

SSH also naturally integrates with more advanced setups. If the project eventually incorporates data archiving, automated uploading to a shared server, or even a lightweight web interface, running the Pi headlessly makes those features straightforward to implement. In this sense, the SSH method opens the door to future expandability. Meanwhile, the operators still retain full visibility into the machine's behavior through terminal access; they can read logs, check directories, or even launch a graphical dashboard on demand if the Pi is configured to support it.

One of the benefits of both approaches is that they are not mutually exclusive. Nothing prevents the system from supporting a small HDMI screen for hands-on use while still allowing remote connections for data transfer. In fact, this combination may provide the most flexibility. A technician working beside the machine can see live status updates on the display, while a researcher at a nearby workstation can download the CSV file without

leaving their desk. This hybrid model also ensures that the system remains usable even if network access is temporarily unavailable.

Ultimately, the goal is to make the operator's experience as smooth as possible. Whether the data is viewed directly on a mounted display or retrieved through a remote session, the system will be designed so that accessing the CSV output is intuitive and reliable. The testing machine will generate the necessary measurement files in a consistent format, and users will be able to access them in whichever way best fits their workflow. Both approaches serve the broader purpose of ensuring that the data is easy to interpret, easy to retrieve, and easy to incorporate into the larger research process.

9 Chapter 9 System Testing and Evaluation

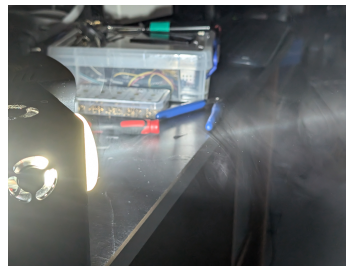
9.1 Optics

9.11 White Light Source

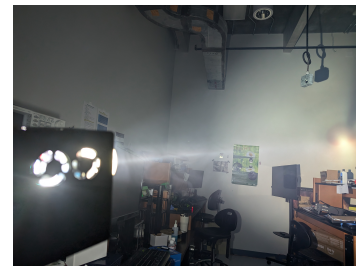
View WLS testing in fig. 84. During testing a power of $1\text{W}/\text{m}^2$ was transmitted over a length of 9 meters. This demonstrates a high degree of collimation. Collimation will only improve as better optics are incorporated.



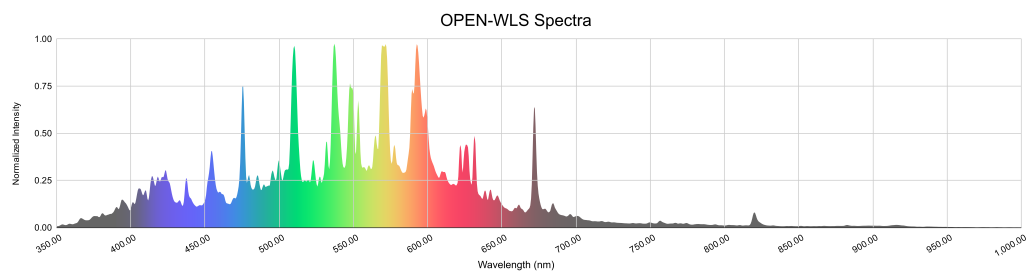
(a) WLS testing during demo.



(b) WLS beam close up.



(c) WLS wall illuminated spot.



(d) WLS spectral distribution.

Fig. 84: White light source testing.

9.12 Monochromator

The monochromator prototype 2 used in our Senior Design I Showcase is visible in Fig. 85, successfully coupled with our current WLS prototype. It features our custom-modeled and printed optics hardware as well as our laser-cut apertures and stick-on mirrors. A cardboard sheet was added behind the entrance slit to prevent light leakage from the WLS. To select different colors from the white light, these diffraction gratings must be moved manually.

Testing of our diffraction gratings at three visible wavelengths (red, green, and blue) and one ultraviolet wavelength is shown in Fig. 86. This was performed with a GREEN-Wave Low Cost Spectrometer from StellarNet [56] connected to a computer running the SpectraWiz Spectroscopy Suite [57]. The spectrometer measures wavelengths in the UV and visible ranges, and some NIR (350nm to 1150nm). It is made to be compatible with the company's software SpectraWiz to quickly and accurately take spectral measurements of incident light on the spectrometer's input cable.

As predicted, the full-width half-max (FWHM) of the diffracted spectrum increases drastically at wavelengths outside of its recommended range (400nm to 700nm in this case). FWHM is a measure of spectral purity, or how well the diffraction grating is able to single out a specific wavelength of light and direct it to one spot. Both gratings yielded nearly identical results, with an average FWHM for visible light of 3.58nm. This is satisfactory for our evaluation wavelength tolerance of $\pm 5\text{nm}$.

It is worth noting that while the diffraction gratings have a peak efficiency of 500nm, the "green" spectrum shown in Fig. 86b has a lower incident power on the spectrometer than the "red" spectrum in Fig. 86a. This may be due to differences in position and distance of the spectrometer's input to the incident light. If a light source is further away from a sensor, the sensor may read an artificially low power due to the light's dispersion in the air.

9.2 Electronics

a) STM32 Board Testing

Testing of the STM32 control board will begin by verifying that the 3.3 V and 5 V power rails start up correctly and remain within tolerance under light load. Once the power system is validated, diagnostic firmware will be uploaded to toggle GPIO pins at known frequencies, allowing confirmation of clock configuration, pin assignments, and signal integrity using an oscilloscope or logic analyzer. The board's USB or UART interface will also be tested to ensure reliable programming and serial communication. After these initial checks, additional firmware will be written to exercise timers, PWM modules, and interrupt routines, ensuring that all peripherals required for driving the stepper system will behave correctly before integrating the A4988 driver.

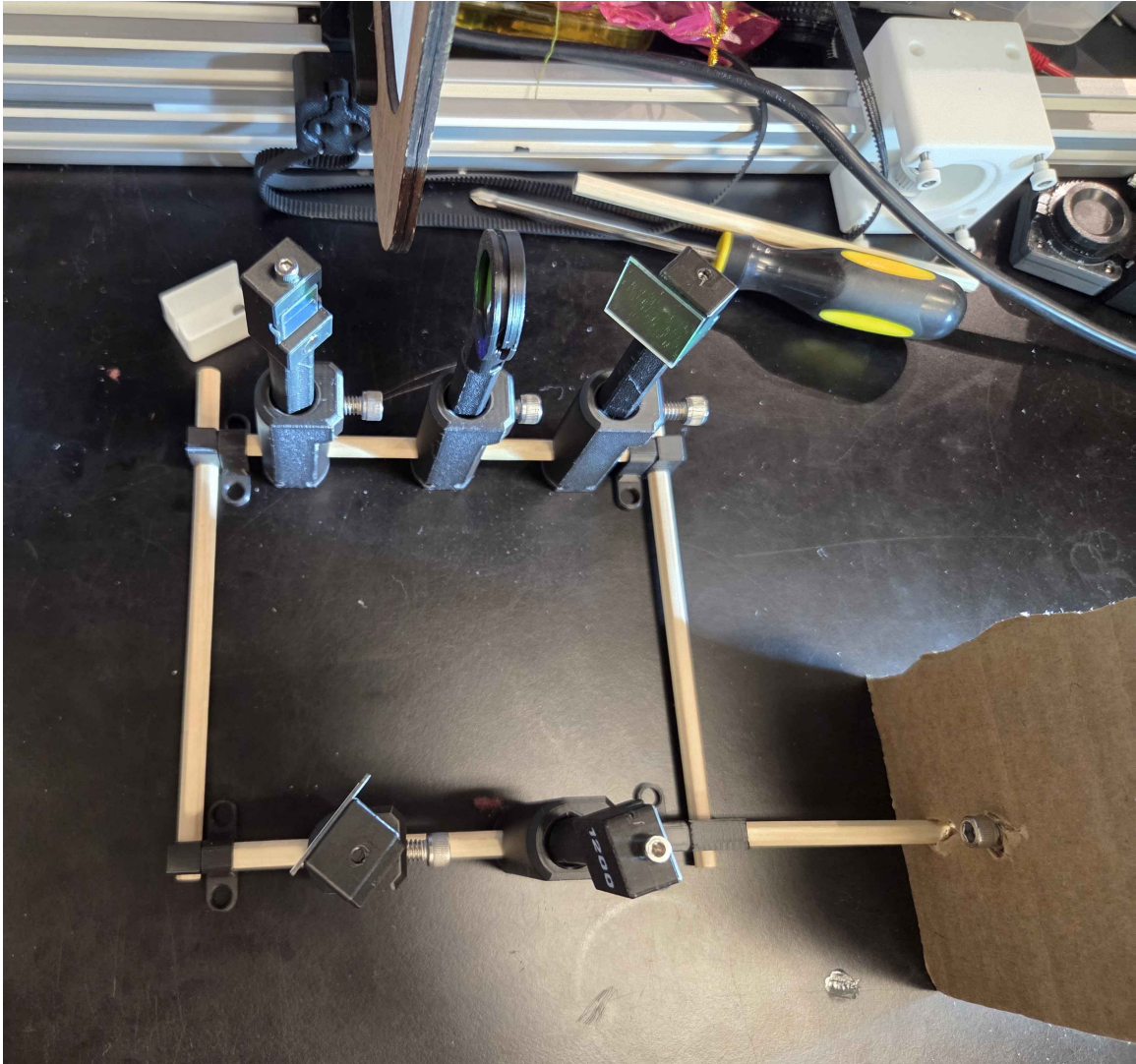
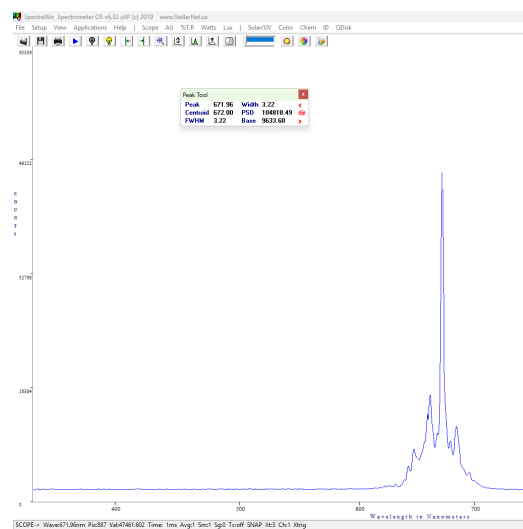


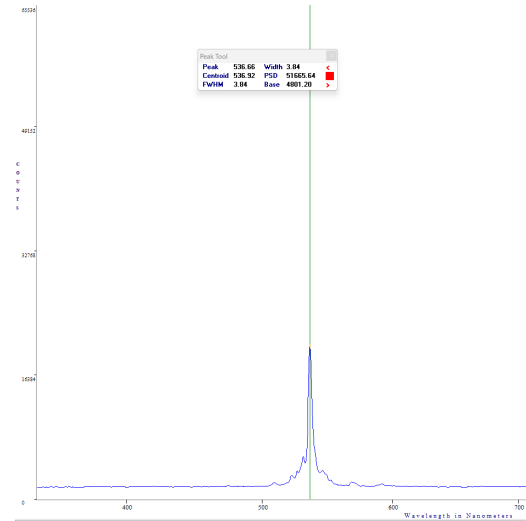
Fig. 85: Monochromator testing during demo.

b) A4988 Stepper Driver Testing

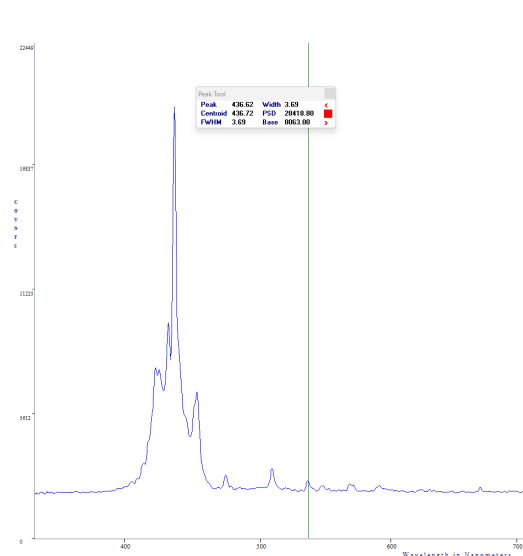
The A4988 stepper driver will be verified by first checking that its VMOT, VDD, and VREF voltages fall within the expected ranges, beginning with adjustment of the VREF potentiometer to set the desired current limit. The ENABLE, DIR, and STEP pins will then be driven from the STM32 to confirm correct digital logic behavior and to ensure that the pull-up and pull-down networks bias the inputs properly when inactive. Once these logic tests pass, controlled STEP pulse trains will be generated to confirm that the driver responds predictably, initially without a motor attached and later with a connected stepper motor running at low currents and reduced step rates. Thermal performance, microstepping accuracy, and current-limiting behavior will be evaluated to ensure stable operation under expected load conditions.



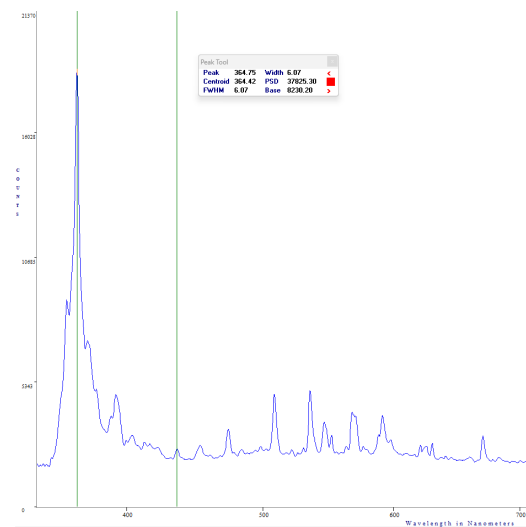
(a) Diffraction grating spectrum tested at 672nm.



(b) Diffraction grating spectrum tested at 536nm.



(c) Diffraction grating spectrum tested at 436nm.



(d) Diffraction grating spectrum tested at 365nm.

Fig. 86: Diffraction grating testing.

c) TPS562201 Power Supply Testing

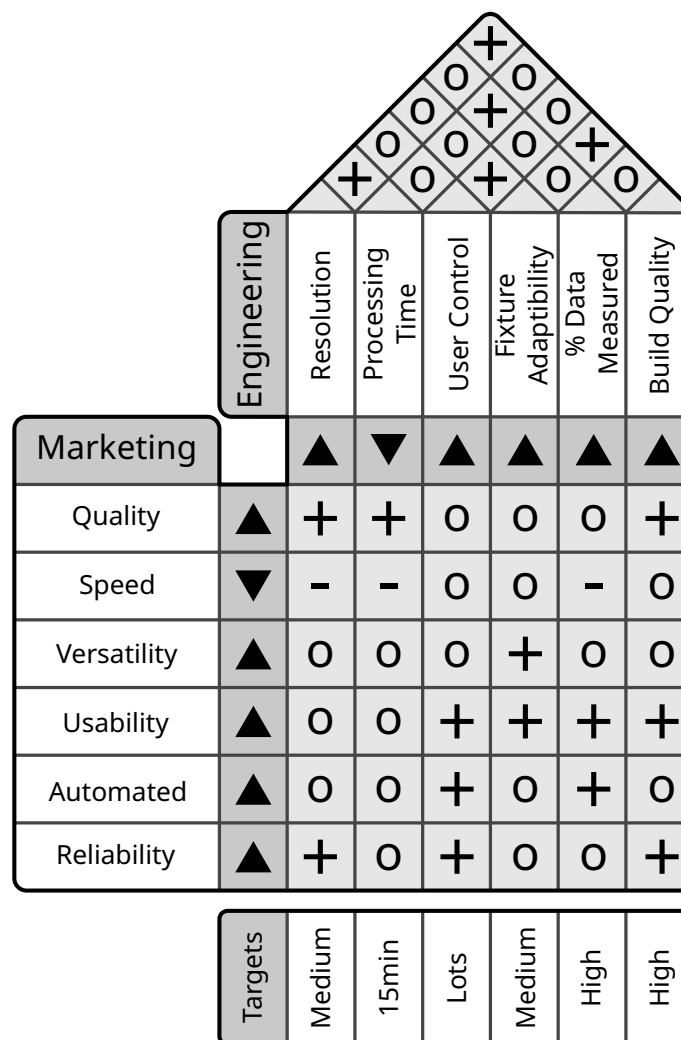
Testing of the TPS562201 buck regulator will begin by confirming that the regulator produces the correct output voltage from the battery input under no-load conditions. The output will then be evaluated under incrementally increasing loads to ensure that the converter maintains regulation and does not enter current limit or thermal shutdown. Output ripple and transient response will be characterized using an oscilloscope to verify that the input and output capacitors provide sufficient filtering and that the converter remains stable during sudden load changes, such as motor startup or rapid stepping transitions. Once validated independently, the TPS562201 will be tested alongside the

STM32 and A4988 driver to confirm that it can reliably supply system current demands without excessive heat generation or noise coupling into sensitive logic lines.

10 Chapter 10 Administrative Content

10.1 House of Quality

A house of quality (visible in Fig. 87) is a planning system that allows the comparison of customer needs to engineering requirements, and helps in balancing goals to fit.



+/- indicates correlation, ▲/▼ indicates desire

Fig. 87: House of Quality

10.2 Budget

We are aiming for a sub \$1000 system. These numbers do not include already owned materials, or those provided by sponsors. The breakdown by category is below:

- Electronics \$400
- Mechanics \$200
- Optics \$300
- Replacement parts and other misc. \$100

10.3 Bill of Materials

Table XIV is an overview/estimate of the materials needed. Note that the Linear Gantry expenses are detailed in Table VIII. The STM32 bill of materials is expanded upon in Table XV with a supplementary figure (Fig. 88) showing the placement of each part listed.

10.4 Milestones

The milestones are visible in Tables XVII & XVIII.

10.5 Task Distribution

Our team consists of two Photonic Science and Engineering (PSE) majors, an Electrical Engineering (EE) major, and a Computer Engineering (CPE) major. The tasks are divided by specialization in Table XIX.

11 Chapter 11 Conclusion

Within the constraints of Senior Design I, the Automated Lens Parameter Metrology system is well on its way to realization. By the end of Senior Design II, we anticipate having a functional system with integrated optics, electronic hardware, software, and mechanics. The finished device will be able to take in common spheric and meniscus lenses and directly measure their diameter, center thickness, and radii of curvature with mechanical tools. The front and back focal lengths of the lenses as well as the power of the lenses will be measured by the IAS. The index of refraction and Abbe number will be calculated using data obtained by the monochromator and IAS working in conjunction with one another. The lenses' coating will be estimated using the transmissive and reflective spectral data generated by the monochromator and photodiode setup.

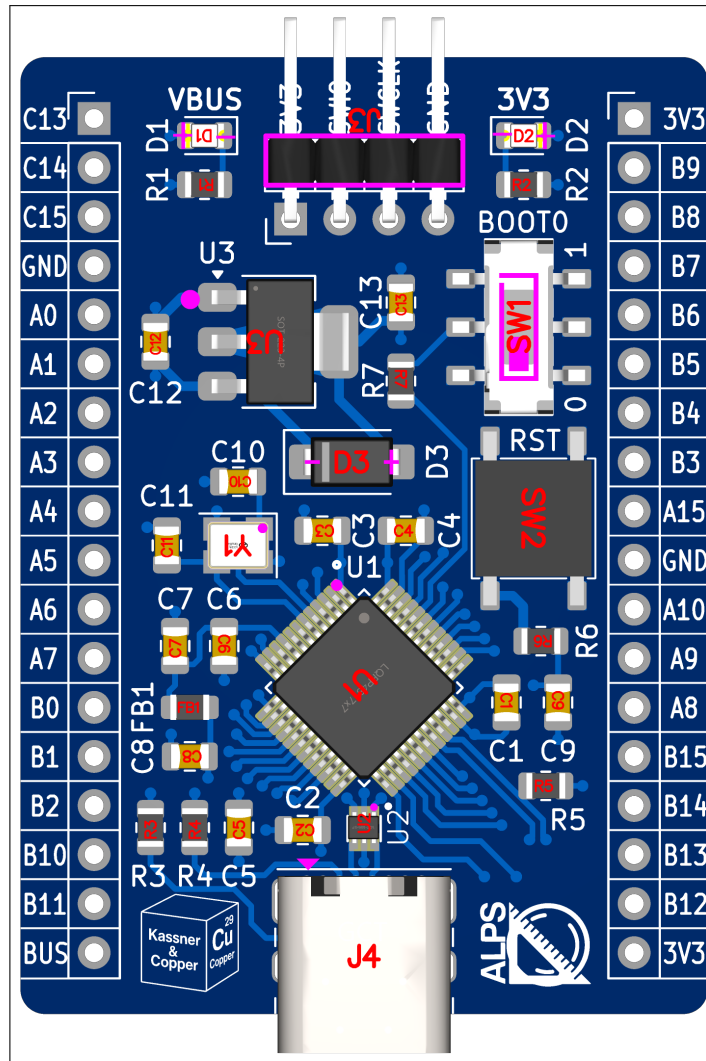


Fig. 88: STM32 Part Placement

The optical and mechanical controls of the system will be provided by stepper motors and custom PCBs including an STM32 with a stepper motor driver and power supply. The photodiodes of the optical system will be directly connected to the PCB by an Op-Amp system to support their performance as sensors.

Our lens data processing and analysis will be performed by a software block that is integrated with our electronic hardware choices. The lead controller is a Raspberry Pi unit that oversees smaller microcontrollers. They are coordinated together using primarily the I²C protocol in which the Raspberry Pi periodically checks each independent subsystem for data.

The WLS has been fully researched, planned, and constructed, acting as the first step in the method by which we will process a lens. It is nearly collimated down to our objective of a 10mm spot size. The WLS feeds directly into the double monochromator, which

outputs a re-collimated single-wavelength beam in the visible range. This monochromator has been successfully prototyped and will be integrated with the electronic and mechanical systems in Senior Design II. The monochromatic light shines through the test lens and at this point data collection begins. A transmissive photodiode sends power data for each color to the Raspberry Pi through the STM32. The test lens is then rotated and the process is repeated for the reflective photodiode. Our STM32 PCB has a fully-planned layout and the coding of it as well as the Pi and microcontrollers is well under way. Once returned to its original position, the test lens begins interacting with our IAS for focal length, power, and spot size measurements. The IAS imaging plane, camera, and linear gantry have all been acquired and are currently undergoing integration which will continue with the electronic hardware and software blocks in Senior Design II. After data collection is completed, the software block will perform the necessary calculations to complete our list of lens characteristics. It will then work with the electronic hardware to output the data to the user via a file, screen, or printer. This final step will be developed further in Senior Design II.

Overall, our group is on track to meet all of our basic goals and objectives by the end of Senior Design II.

TABLE III: Comparison of Single-Board Computers Considered for System Controller Selection

Specification	Raspberry Pi 3 Model B	Raspberry Pi 4 Model B (4 GB)	Raspberry Pi 5 (4 GB)	NVIDIA Jetson Nano
Processor	1.2 GHz Quad-Core ARM Cortex-A53	1.5 GHz Quad-Core ARM Cortex-A72	2.4 GHz Quad-Core ARM Cortex-A76	1.43 GHz Quad-Core ARM Cortex-A57
GPU	Broadcom VideoCore IV	Broadcom VideoCore VI	Broadcom VideoCore VII	128-core Maxwell GPU
Memory	1 GB LPDDR2	4 GB LPDDR4	4 GB LPDDR4X	4 GB LPDDR4
Storage	microSD	microSD	microSD (and PCIe NVMe via adapter)	microSD
USB Ports	4× USB 2.0	2× USB 3.0, 2× USB 2.0	2× USB 3.0 (dedicated lanes), 2× USB 2.0	4× USB 3.0
Camera Interface(s)	1× CSI (shared)	2× CSI/DSI (shared lanes)	2× full-speed CSI/DSI connectors	1× MIPI CSI
Networking	10/100 Ethernet, 2.4 GHz Wi-Fi, Bluetooth 4.1	Gigabit Ethernet, Dual-Band Wi-Fi, Bluetooth 5.0	Gigabit Ethernet, Dual-Band Wi-Fi 802.11ac, Bluetooth 5.0	Gigabit Ethernet (no Wi-Fi onboard)
I/O Interfaces	40-pin GPIO, UART, I ² C, SPI	40-pin GPIO, UART, I ² C, SPI	40-pin GPIO, UART, I ² C, SPI, PCIe 2.0 ×1	40-pin GPIO, UART, I ² C, SPI
Power Requirements	5 V @ 2.5 A (micro-USB)	5 V @ 3.0 A (USB-C)	5 V @ 5.0 A (USB-C with PD)	5 V @ 4.0 A (barrel jack)
Operating System	Raspberry Pi OS (32-bit)	Raspberry Pi OS (32/64-bit)	Raspberry Pi OS (64-bit)	NVIDIA JetPack (Ubuntu-based)
Approx. Cost (USD)	\$35	\$55	\$70	\$130
Notable Advantages	Low cost, broad community support	Improved CPU/GPU, dual 4K output, good value	Highest CPU performance, PCIe, dual camera ports, best future scalability	Strong GPU acceleration for AI workloads
Limitations	Slower CPU, USB 2.0 only	Shared I/O lanes reduce peak throughput	Slightly higher power draw, limited early software support	High power use, limited non-AI software ecosystem

TABLE IV: Estimated current draw for one motion control PCB

Subsystem	Supply rail	Typical current [A]	Worst case [A]
NEMA 17 stepper (via A4988)	12 V	1.0	1.6
STM32 microcontroller	3.3 V	0.05	0.15
A4988 logic supply	3.3 V	0.02	0.05
LEDs and small peripherals	3.3 / 5 V	0.03	0.10
Total equivalent from 12 V	12 V	≈ 1.2	≈ 2.0

	Photoresistor	Photodiode	Phototransistor
Sensitivity	low	very high	high
Can it generate current?	no	yes	yes
Strong temperature dependence?	yes	no	no
Dark current?	no	yes	no
Cost	low	low	high

TABLE V: Photosensor Comparison

lines/mm	β (656 nm)	β (489 nm)	spectral angular width
300	11.36°	8.44°	2.92°
600	23.19°	17.07°	6.12°
1200	51.95°	35.95°	16.00°

TABLE VI: Grating period comparison

Sample	Wavelength	α
λ_C	656.3nm	-9°
λ_D	589.3nm	-5°
λ_F	486.1nm	2°

Part Name	Vendor	Part #	Quantity
White Light Generator	Dolan-Jenner Industries	Fiber-Lite High-Intensity Illuminator Series 180	1
Fiber Optic Cable 2 ft	Dolan-Jenner Industries	BG2820M	1
Optical Rail	Newport	PRL-36	1
Optical Rail Carrier 3 in	Newport	PRC-3	2
Optical Rail Carrier 1 in	Newport	PRC-1	7
Ø1/2" Post Holder	ThorLabs	PH1 - PH6	9
Ø1/2" Pedestal Post Holder	ThorLabs	PH4E	2
Clamping Fork	ThorLabs	CF125	2
Linear Stage 1	ThorLabs	XR25 Series	1
Linear Stage 2	Newport	426 Series	1
Rotation Stage	ThorLabs	RP01	2
Tilt Stage	Newport	F-91TS	1
1 in Lens Tube	ThorLabs	SM1S20	1
90 deg Post Mount	ThorLabs	Ra90	1
Filter Mount	ThorLabs	KM200V/M	1
Ø1/2" Optical Post	ThorLabs	TR1 - TR6	12
f = -30 mm Lens	ThorLabs	LC2679	1
f = +150 mm Lens	ThorLabs	LA1417	1
2 in Lens Housing	ThorLabs	LMR2	1
Iris	ThorLabs	ID25	3
Mirror	ThorLabs	BB1-E01	2
Kinematic Mirror Mount	ThorLabs	KM100	2
1200 lines/mm Grating	ThorLabs	GR13-1205	2
Double Slotted Base	Newport	B-1A	1
Screen	ThorLabs	EDU-VS1	2
Filter Mount	Newport	M-PPF50	1
8/32 Screw Set	Newport	SK-08A	1
1/4 - 20 Screw Set	Newport	SK-25A	4
Green Laser	ThorLabs	PL201	1

TABLE VII: Monochromator Prototype List of Materials

Name	Quantity	Description	Vendor
30-3030C	1	1000mm Aluminium Extrusion	Automation Direct
162968	2	FATH Profile Slider	Automation Direct
30-3030C	1	250mm Aluminium Extrusion	Automation Direct
6627T53_Stepper	1	Nema 23 4pin 23HS22-2804S	Previously Owned
3693N214 Idler	1	Belt Idler	McMaster-Carr
3764N106_High-Strength Gt Timing Belt Pulley	1	Belt Pulley	Previously Owned
MotorMount	1	Mounts Stepper to Extrusion	3D Printed
ZM1	1	NO Endstop	Skycraft
3030 Endcap	3	3030 Endcap	3D Printed
Slider Spacer	1	Keeps sliders apart	3D Printed
Idler-Endstop Mount	1	Mounts Idler & Endstop	3D Printed
M3toM5	4	Shim for M3 Bolts	3D Printed
91292A114_18-8 Stainless Steel Socket Head Screw	5	M3-12mm Bolt	McMaster-Carr
91828A211_18-8 Stainless Steel Hex Nut	13	M3 Nut	McMaster-Carr
91292A112_18-8 Stainless Steel Socket Head Screw	8	M3-8mm Bolt	McMaster-Carr
Idler-Endstop Fixture	2	T-Nut	3D Printed
MotorMount Fix	2	T-Nut	3D Printed
Belt Block	1	Fixes Belt to Extrusion	3D Printed
90591A265_Zinc-Plated Steel Hex Nut	2	M2-Nut	McMaster-Carr
91292A834_18-8 Stainless Steel Socket Head Screw	2	M2-12mm Bolt	McMaster-Carr

TABLE VIII: Linear Gantry BOM

TABLE IX: Comparative Summary of Major AI Platforms

Platform	Strengths	Limitations
ChatGPT (OpenAI)	State-of-the-art reasoning, strong instruction following, robust multimodal support	Closed-source; limited transparency in training data; subscription-based access model
Claude (Anthropic)	Ethically aligned responses, very long context window, coherent long-form analysis	Slightly lower coding/mathematical performance; limited integration ecosystem
Gemini (Google DeepMind)	Multimodal reasoning, integration with Google Search and Workspace, factual reliability	Limited transparency; dependency on Google ecosystem; constrained availability
Copilot (Microsoft)	Strong coding assistance, integration with developer tools, productivity enhancement	Narrow domain focus; relies on OpenAI backend; limited reasoning beyond software tasks
Grok (xAI)	Access to live data from X platform, conversational and humorous tone	Inconsistent factual reasoning; lack of academic transparency; early-stage development
DeepSeek (DeepSeek AI)	Open-source, efficient architecture, transparent benchmarking, multilingual proficiency	Slightly lower raw performance on large benchmarks; smaller ecosystem support

Source Type	Effective Spectra	Cost	Collimation Difficulty
LED	Low Quality, three peaks	Medium	Super Easy
Incandescent	Medium Quality, black body curve	Low	Medium
Halogen	Medium Quality, brighter black body curve	Medium	Medium
Fluorescent	Low Quality, five peaks	Low	Incredibly difficult
Xenon	High Quality, constant spectral density	High	Easy/Medium

TABLE X: WLS source choice comparison.

	Vendor	Part number	Average quantum efficiency from 480 nm - 660 nm, approximately	Price
1	ThorLabs	GR25-1204	50%	\$137.27
2	ThorLabs	GR13-1205	75%	\$83.58
3	Newport	33205FL01-280R	72%	\$94.00
4	Edmund Optics	43-004	70%	\$113.42
5	Edmund Optics	41-048	50%	\$113.42

TABLE XI: Diffraction grating part comparison

	Vendor	Part number	Spectral range	Peak wavelength	Response time	Unit price
1	Advanced Photonix	PDB-C171SM	400nm ~1050nm	660 nm	100ns	\$3.32
2	ams-OSRAM USA INC.	SFH 2240	400nm ~690nm	620 nm	90ns	\$1.42
3	Advanced Photonix	PDB-C156	400nm ~1100nm	660 nm	15ns	\$1.13
4	Advanced Photonix	PDB-C154SM	400nm ~1100nm	660nm	10ns	\$1.36
5	Mitsubishi Electric Europe B.V.	T1678P-SD-F	440nm ~700nm	570 nm	100ns, 415ns	\$14.00

TABLE XII: Photodiode grating part comparison

	Camera Module 3	Camera Module 3 Wide	HQ Camera	AI Camera
Net price	\$25	\$35	\$50	\$70
Sensor resolution	4608 × 2592 pixels	4608 × 2592 pixels	4056 × 3040 pixels	4056 × 3040 pixels
Pixel size	1.4 μm × 1.4 μm	1.4 μm × 1.4 μm	1.55 μm × 1.55 μm	1.55 μm × 1.55 μm
Sensor image area	6.45 × 3.63mm	6.45 × 3.63mm	6.287mm × 4.712 mm	6.287mm × 4.712 mm
Focal length	4.74 mm	2.75 mmm	Depends on lens	4.74 mm

TABLE XIII: Raspberry Pi camera comparison.

This table is selectively pulled from [49].

Item Name	Vendor	Total cost
Aluminium Extrusion 1m	Automation Direct	\$13.00
Aluminium Extrusion .25m	Automation Direct	\$3.25
FATH Profile Slider	Automation Direct	\$28.00
Timing Belt Pulley	McMaster-Carr	\$11.17
Idler Pulley	McMaster-Carr	\$6.04
1.1m 6mm Belt	McMaster-Carr	\$14.48
M3x12mm (x100)	McMaster-Carr	\$6.00
M3 Nut (x100)	McMaster-Carr	\$4.73
M3x8mm (x100)	McMaster-Carr	\$20.30
HQ Camera	Adafruit	\$55.00
2m Flex Cable	Adafruit	\$5.95
1M Linear Pot	Adafruit	\$21.03
Raspberry Pi 5 4GB	Amazon	\$71.04
4K Micro HDMI to HDMI Male Cable Adapter	Amazon	\$13.16
Raspberry Pi 5 Active Cooler	Amazon	\$11.39
ST-Link V2	Amazon	\$6.41
Aluminium Tape	Amazon	\$9.48
Mirror Tape	Amazon	\$0.97
3010 Fans x2	Amazon	\$11.99
1200l/mm Diffraction Grating	Thorlabs	\$167.16
Aspheric Lens	Edmund Optics	\$49.82

TABLE XIV: Bill of Materials

Comment	Designator	Footprint	JLPCB Part #
10n	C6	0805	C1710
1k5	R5	0805	C4310
Button	SW2	K2-1157SP-I4SW-01	C49234152
USB_C_Receptacle_USB2.0	J4	USB_C_Receptacle_GCT_USB4085	C7095263
SS14	D3	D_SMA	C2480
1u	C7,C8	0805	C28323
Conn_01x04	J3	PinHeader_1x04_P2.54mm_Horizontal	C91552
22u	C12,C13	0805	C45783
100n	C1,C2,C3,C4,C9	0805	C49678
8MHz	Y1	Crystal_SMD_3225-4Pin_3.2x2.5mm	C5308002
USBLC6-2P6	U2	USBLC6-2P6	C15999
10u	C5	0805	C440198
10p	C10,C11	0805	C157608
RED	D1,D2	0805	C965812
270	R2	0805	C17590
SW_SPDT	SW1	HXMST22D18G2	C5149822
120R	FB1	0805	C5671
470	R1	0805	C17710
AMS1117-3.3	U3	SOT-223-3_TabPin2	C6186
10K	R7	0805	C17414
STM32F103C8Tx	U1	QFP50P900X900X160-48N	C8734
100	R6	0805	C17408
5k1	R3,R4	0805	C27834

TABLE XV: STM32 Bill of Materials (BOM)

Reference	Value	Digi-Key Part Number
C1	0.1 uF	—
C2	22 uF	—
C3, C4, C7, C11	0.1 uF	399-C0603C104K5RACTUCT-ND
C5	0.22 uF	490-5402-1-ND
C8	10 uF	1276-1096-1-ND
D1, D2, D3	Red LED	LSM0603412VCT-ND
H1, H2, H3, H4	Mounting Hole	—
J1, J2	Conn_01x08	—
R1, R2	0.05 Ω	273-KDV06FR050ETCT-ND
R3	0–100k Potentiometer	TC33X-104ECT-ND
R4, R10, R11, R12, R13	24k	541-5285-1-ND
R5, R8, R9	200 Ω	RHM200ADCT-ND
R6	20k (Resistor Network)	13-YC324-JK-0720KLCT-ND
R7	330k	311-330KGRCT-ND
U1	A4988 Driver IC	620-1343-1-ND

TABLE XVI: Bill of Materials for A4988 Stepper Driver Breakout

Milestone	Start Date	Completion Date
Group Formation	8/19/25	8/28/25
Project Idea	8/19/25	8/28/25
Committee Formation	8/28/25	9/5/25
D&C Completion	8/28/25	9/5/25
D&C Meeting	9/10/25 5pm	9/10/25 5pm
D&C Document Revision	9/10/25	9/12/25
Research Components	9/12/25	TBD
Midterm Milestone Report	9/13/25	10/31/25
Midterm Report Revision	11/3/25	11/7/25
Final Report		11/25/25
Demo Video		11/25/25

TABLE XVII: Project Initialization (SD-I)

Milestone	Start Date	Completion Date
Order Initial PCB	1/1/25	2/1/25
Fabrication of Tray and Lens Mount Assembly	1/1/25	2/1/25
STM32 Firmware Development and Raspberry Pi Software Development	1/1/25	4/1/25
PCB Assembly	3/1/25	4/1/25
Optical Light Source and Photodiode Assembly	1/1/25	3/1/25
Imaging System Integration	TBD	TBD
System Integration and Debugging	TBD	TBD
Final Prototype Build	TBD	TBD
Final Testing	TBD	TBD
Final Presentation	TBD	TBD

TABLE XVIII: Project Fabrication (SD-II)

Major	PSE		EE	CPE
Member	Kiva	Ollie	Zachary	Daniel
Tasks	WLS, IAS, Optical Data Processing & Mechanical Systems	Diffraction Grating & Monochromator	System Supporting Electronics & PCB	Embedded Develop & Data Collectio
Alternates	Ollie	Daniel	Kiva	Zach

TABLE XIX: Task Distribution

Appendix A

ChatGPT Prompts and Outcomes

A.1 Case Study 1

A.11 Case Study 1 Prompt

"Can you explain the pros and cons of using an ATMEGA32, ESP32 and STM32 MCU for a project?"

A.12 Case Study 1 ChatGPT Outcome

"Here's a clear, practical comparison of **ATmega32**, **ESP32**, and **STM32** microcontrollers, focusing on the real pros/cons that matter when choosing one for a project.

—

ATmega32 vs ESP32 vs STM32 — Pros Cons

Below is a structured comparison that covers capability, ease of use, peripherals, cost, and ideal use-cases.

—

ATmega32 (8-bit AVR)

Pros

Simple architecture — very easy to learn for beginners. **Extremely stable** predictable timing (8-bit, no complex pipeline). **Wide support** in low-level embedded courses, Arduino-like workflow possible. **Rich set of basic peripherals**: ADC, UART, SPI, I2C, timers. **Low power consumption**, good for battery devices with sleep modes. **Tons of documentation** and community examples.

Cons

Very limited performance (8-bit, typically 16–20 MHz max). **Small memory** (usually ; 32 KB flash, 2 KB SRAM). **No built-in connectivity** (no WiFi/BLE). **Not great for multitasking** or complex control loops. **Limited ADC resolution** no advanced peripherals like DMA, high-speed ADC, etc.

Best for

Simple control systems * Hobby robotics * Basic sensor interfaces * Projects needing deterministic timing without complexity

****ESP32 (Xtensa / RISC-V Dual-core SoC)****

****Pros****

* ****WiFi + Bluetooth built-in****, best for IoT. * ****Very cheap**** for how powerful it is (3–8). * ****Dual-core CPU**** at 240 MHz → excellent processing power. * ****Large memory**** (hundreds of KB RAM + external flash). * ****Rich peripherals****: ADC, DAC, I2C, SPI, CAN, touch sensors. * ****Easy Arduino IDE or MicroPython programming****. * ****Great for web servers, data logging, wireless control****.

****Cons****

* ****Not real-time deterministic**** (WiFi/BLE subsystems cause jitter). * ****ADC noise is notoriously bad**** (poor linearity and accuracy). * ****Higher power consumption**** unless deep sleep is used. * ****Can be unstable for mission-critical systems**** without RTOS tuning. * ****Boot process quirks****, especially with auto-programming circuits.

****Best for****

* Wireless/IoT projects * Web APIs on a microcontroller * Over-the-air update systems
* Smart home or mobile robotics with cloud integration

****STM32 (ARM Cortex-M0/M3/M4/M7 family)****

****Pros****

* ****Huge range of performance tiers****:

* ****M0****: low power, very cheap * ****M3/M4****: mid-performance * ****M7****: extremely fast, DSP capable * ****Real-time performance**** with deterministic interrupt latency. * ****Excellent peripherals****:

* High-speed ADC/DAC * DMA * Multiple UART/I2C/SPI * CAN / USB / I²S / PWM
* ****STM32CubeIDE + HAL/LL libraries**** give professional development environment.
* ****Low power modes**** across the lineup. * ****Very reliable for industrial and robotics applications****.

****Cons****

* ****Steeper learning curve**** than ATmega or ESP32. * ****More boilerplate code**** needed unless using CubeMX/HAL. * ****Different families can behave differently**** (F-series vs L-series etc.). * ****More expensive than ATmega32****, usually 2–6 but can go higher.

****Best for****

* Robotics and control systems * High-speed sensors (IMUs, LIDAR, advanced ADC) * Real-time applications (motor control, drones) * USB or CAN-based devices * Industrial embedded designs

—

****Quick Comparison Table****

Feature	ATmega32	ESP32	STM32
CPU	8-bit AVR	Dual-core 32-bit ARM (M0-M7)	32-bit ARM (M0-M7)
Speed	16 MHz	240 MHz	32-480 MHz
Connectivity	None	WiFi, Bluetooth	Some models (Ethernet, CAN, USB)
Memory	kB	Hundreds of kB	Varies widely
Real-time?	Very good	Poor (unless carefully tuned)	Excellent
Ease of use	Simple	Easy	Moderate
Cost	Low	Low	Low-Medium
Best for	Basic electronics	IoT/wireless	Real-time/industrial/robotics

—

****How to Choose for Your Project****

Here's a rule-of-thumb:

Choose ****ATmega32**** if:

* You want something simple and predictable * No wireless needed * Low power / low cost * You're doing coursework or basic embedded systems

Choose ****ESP32**** if:

* WiFi/Bluetooth is important * You want very high processing power for cheap * You don't need precise timing or high-accuracy ADC * You want fast prototyping

Choose ****STM32**** if:

* You need real-time performance * You need high-quality ADCs, DMA, timers * You're building something professional/industrial * You need USB, CAN, or advanced peripherals * You care about reliability and deterministic timing

—

If you want, tell me ****your specific project****, and I can recommend the ideal MCU and justify the choice."

A.13 Case Study 1 Copilot Outcome

”Great question! These three MCUs—**ATMEGA32**, **ESP32**, and **STM32**—are popular choices, but they serve different purposes. Here’s a detailed comparison:

****ATMEGA32 (8-bit AVR MCU)****

****Pros:****

* ****Simple architecture****: Easy to learn for beginners, widely used in academic projects.
 * ****Low power consumption****: Good for battery-operated devices. * ****Stable and mature ecosystem****: Tons of tutorials, libraries, and community support. * ****Cost-effective****: Very cheap compared to ESP32 and STM32.

****Cons:****

* ****Limited performance****: 8-bit, typically runs at 16 MHz, not suitable for heavy computations. * ****Small memory****: Limited Flash (32 KB) and RAM (2 KB). * ****No built-in connectivity****: Requires external modules for Wi-Fi/Bluetooth. * ****Basic peripherals****: No advanced features like DMA or high-speed ADC.

****ESP32 (32-bit Xtensa MCU with Wi-Fi/Bluetooth)****

****Pros:****

* ****Integrated Wi-Fi and Bluetooth****: Ideal for IoT projects. * ****High performance****: Dual-core, up to 240 MHz, much faster than ATMEGA32. * ****Rich peripherals****: ADC, DAC, PWM, touch sensors, CAN, etc. * ****Large memory****: Up to 520 KB SRAM, plus external flash. * ****Low cost for features****: Great value for IoT and wireless applications.

****Cons:****

* ****Higher power consumption****: Not ideal for ultra-low-power applications. * ****Complexity****: More challenging for beginners compared to ATMEGA32. * ****Limited real-time performance****: Not as deterministic as STM32 for critical timing tasks.

****STM32 (ARM Cortex-M MCU family)****

****Pros:****

* ****Wide range of options****: From low-power Cortex-M0 to high-performance Cortex-M7. * ****Excellent performance****: Up to hundreds of MHz, floating-point support. * ****Advanced peripherals****: DMA, high-speed ADC/DAC, CAN, USB, Ethernet. *

****Real-time capability****: Great for industrial and automotive applications. ****Strong ecosystem****: STM32CubeMX, HAL libraries, RTOS support.

****Cons:****

****No built-in Wi-Fi/Bluetooth****: Requires external modules. ****Higher cost****: More expensive than ATMEGA32 and sometimes ESP32. ****Steeper learning curve****: More complex development environment.

****Which one should you choose?****

****ATMEGA32**** → Best for ****basic, low-cost, low-power projects**** (e.g., simple sensors, LED control). ****ESP32**** → Best for ****IoT and wireless applications**** (e.g., smart home, Wi-Fi-enabled devices). ****STM32**** → Best for ****high-performance, real-time, industrial or complex projects**** (e.g., motor control, robotics).

Do you want me to ****make a comparison table**** for these three MCUs with specs like clock speed, memory, connectivity, and cost? Or should I also include ****use-case examples**** for each?"

A.14 Case Study 1 Claude Outcome

"I'll break down the pros and cons of each microcontroller to help you choose the right one for your project.

ATMEGA32 (AVR Family)

****Pros:**** - Very beginner-friendly with extensive Arduino support and documentation - Low power consumption in sleep modes - Simple, straightforward architecture that's easy to learn - Excellent community support and countless tutorials - Affordable and widely available - Stable and reliable for simple applications - Good selection of development tools and libraries

****Cons:**** - Limited processing power (16 MHz, 8-bit architecture) - Only 32KB of flash memory and 2KB RAM - No built-in connectivity (WiFi, Bluetooth) - Fewer advanced peripherals compared to modern MCUs - Limited to 5V or 3.3V logic levels with no built-in USB - Becoming somewhat dated for complex modern applications

ESP32

****Pros:**** - Built-in WiFi and Bluetooth (Classic and BLE), making IoT projects simple - Powerful dual-core processor (up to 240 MHz) - Large memory (up to 520KB SRAM, 4MB+ flash typically) - Very cost-effective given its capabilities - Rich set of peripherals (ADC, DAC, touch sensors, I2S) - Active development community and good

documentation - Arduino IDE support plus native ESP-IDF framework

****Cons:**** - Higher power consumption, especially when WiFi is active - More complex to program than simpler MCUs if using advanced features - Can be overkill for simple projects - Some peripheral libraries are less mature than Arduino ecosystem - Documentation can be inconsistent between different ESP32 variants - Requires external components for programming (USB-UART bridge)

STM32 (ARM Cortex-M Family)

****Pros:**** - Extremely wide range of models from basic to high-performance - Excellent performance-to-cost ratio with 32-bit ARM architecture - Rich peripheral set (CAN, USB, DMA, advanced timers) - Professional-grade with good industrial support - Low power consumption with multiple sleep modes - Extensive HAL libraries from ST - Good debugging support with ST-Link - Scalable across product lines

****Cons:**** - Steeper learning curve, especially for beginners - Less community support compared to Arduino ecosystem - Development environment setup can be complex - More expensive development boards (though chips themselves are affordable) - Documentation can be overwhelming due to the variety of options - Requires more understanding of registers and hardware configuration - Less plug-and-play than Arduino-compatible boards

Quick Selection Guide

- ****Choose ATMEGA32**** for simple projects, learning electronics, or when you need proven reliability with minimal complexity - ****Choose ESP32**** for IoT projects, when you need wireless connectivity, or when you need more processing power on a budget - ****Choose STM32**** for professional applications, when you need specific peripherals, low power with high performance, or plan to scale to production

What type of project are you working on? That would help me give more specific recommendations.”

A.2 Case Study 2

A.21 Case Study 2 Prompt

”What are the proper steps to take when designing a monochromator?”

A.22 Case Study 2 ChatGPT Outcome

” Designing a monochromator — step-by-step

Nice question — designing a monochromator is a mix of optics, mechanics and practical engineering tradeoffs. Below is a practical, engineering-focused roadmap you can follow, followed by the key equations and a short worked example.

1) Specify the system requirements (first and most important)

Write down hard specs and soft targets:

* **Wavelength range** (e.g., 200–800 nm). * **Spectral resolution** () or **resolving power** ($R = \lambda / \Delta\lambda$). * **Throughput/etendue** requirement (light-collection area acceptance solid angle). * **Stray-light/out-of-band rejection** (e.g., < 0.01). * **Scanspeed/stepsize** (for scanning monochromators). * **Detector type/sensitivity** (PMT, CCD/CMOS, photodiode). * **Physical constraints** : size, weight, cost, environmental (temperature, vibration). * **Calibration requirements** and how often you must re-calibrate.

2) Choose the dispersive element / optical configuration

Main options:

* **Diffraction grating** (most common): choose groove density, blaze, and substrate. Good for wide range and high resolution. * **Prism**: simpler, good throughput in some ranges, nonlinear dispersion (not ideal if linear dispersion desired). * **Filter-based** (interference filters): for fixed narrow bands only. * **Optical layouts**:

* **Czerny–Turner** (collimating mirror → grating → focusing mirror): flexible and common. * **Ebert–Fastie** (single mirror + grating): compact, fewer optics. * **Monochromator with double stage** (two monochromators in series) for very low stray light.

3) Optical design key geometry

Decide entrance slit → collimator → disperser → camera → exit slit / detector geometry. Important choices:

* Collimated beam diameter (sets number of grating grooves illuminated → resolving power). * Incident (α) and diffraction (β) angles (use grating equation below). * Focal length of focusing optics (affects linear dispersion at detector). * Slit widths | entrance and exit slits.

4) Calculate dispersion and resolution (essential formulas)

Use these standard relations for a grating:

* **Grating equation** [$m\lambda = d(\sin \alpha + \sin \beta)$] where (m) is diffraction order, (d) grooves spacing (m), (α) incident angle, (β) diffraction angle.

* **Angular dispersion** [$d\beta/d\lambda = \frac{m}{d \cos \beta}$]

* **Linear dispersion at image plane** [$D = f_{\text{cam}} \frac{d\beta}{d\lambda} = f_{\text{cam}} \frac{m}{d \cos \beta}$] where (f_{cam}) is camera focal length. Units: mm per nm (if f in mm and d in mm).

* **Wavelength resolution due to slit width** [$\Delta\lambda \approx \frac{w}{D}$] where (w) is the projected image width of the slit.

* **Resolving power (grating limit)** [$R = \lambda / \Delta\lambda \leq mN$] where (N) is total number of grooves illuminated (grooves across illuminated width).

* **Important practical relationship:** if you require $\Delta\lambda$, you can find needed slit width or illuminated width (and thus grating size/density).

—

5) Choose grating/prism properties

For a grating:

* Groove density (lines/mm) — higher density $\rightarrow$ higher dispersion but can reduce efficiency and usable wavelength range. * Blaze wavelength/angle — maximizes efficiency near design λ . * Order selection: higher order gives higher resolution (m) but overlaps; may need order-sorting filters. * Coating (Al, Au, dielectric) depending on UV/VIS/IR.

—

6) Slits, apertures, and stray light control

* **Slit design:** precision slits with knife-edges; consider motorized adjustable slit if you need variable resolution/throughput. * **Baffles & stray-light traps:** blackened interiors, knife-edge baffles, light traps at unused orders. * **Internal coatings:** low-reflectance, matte black finishes; minimize scattering surfaces.

—

7) Detectors and electronics

* Pick detector based on wavelength and required sensitivity: PMTs (deep UV / low light), CCD/CMOS (parallel detection), photodiodes (simple). * For scanning monochromators, consider synchronous detection (lock-in) for low SNR tasks. * Consider ADC resolution, readout noise, cooling (for CCDs), and linearity.

—

8) Mechanical, thermal, and alignment design

* Rigid, thermally stable base to avoid wavelength drift. Use low-CTE materials if needed. * Precise rotation stages for grating (stepper/encoder) and optionally for entrance/exit slits. * Provision for fine alignment (tilts, focus). * Services: cable routing, light-tight housing, access ports.

9) Calibration, testing characterization

* Wavelength calibration: use known spectral lines (Hg, Ne, Ar, etc.) or lamps/laser lines.
 * Measure instrumental linewidth and compare to design . * Measure throughput vs. wavelength (using calibrated lamp and photodiode). * Measure stray-light / out-of-band rejection (e.g., by using a narrow-line source and measuring background). * Repeatability and stability tests (temperature cycles, time drift).

10) Software, control, and user interface

* Motor control (grating rotation, slit adjustment). * Data acquisition and processing (wavelength calibration, baseline subtraction). * Automation for scans, saving configurations, logging environmental parameters (T).

11) Safety, manufacturability, and cost tradeoffs

* UV safety (if UV wavelengths) — enclosures and interlocks. * MOQ and availability for custom gratings or custom optics. * Balance cost vs. performance: double monochromators reduce stray light but increase cost/size.

Quick worked example (illustrative)

Requirement: 200–800 nm, $\Delta\lambda = 0.1 \text{ nm}$ at $\lambda = 500 \text{ nm}$.

* Resolve power needed: ($R = \lambda / \Delta\lambda = 500 / 0.1 = 5000$). * *If using first order ($m = 1$), need ($N = R / m = 5000$) grooves illuminated.* * *If you pick a 1200 lines/mm grating (i.e., ($g = 1200$) grooves/mm), illuminated width (W) required :*

* Compute ($W = N / g = 5000 / 1200$). * Do the arithmetic: divide numerator and denominator by 100 $\rightarrow (50/12)$. Reduce $\rightarrow (25/6)$. ($25 \div 6 = 4.1666\dots$) mm. * So you must illuminate about $\approx 4.17 \text{ mm}$ of grating width. That's modest; higher resolution demands larger illuminated width or higher order.

Then choose camera focal length to match slit width and linear dispersion so that slit image width (w) gives $\Delta\lambda \approx w / D$.

Practical tips common pitfalls

* **Don't ignore stray light.** It often dominates performance in real systems. Add

baffles and consider a double stage for demanding specs. * **Thermal drift.** Even small temperature changes shift wavelengths; either stabilize temperature or frequently recalibrate. * **Slit/optics mis-match.** A too-wide slit will kill resolution; too narrow will kill throughput and SNR. * **Order overlap.** Use order-sorting filters if you operate at high orders or over broad ranges. * **Manufacturing tolerances.** Surface figure of mirrors and grating flatness affect resolution; specify these to vendors.

Design checklist (short)

* ☐ Specs: range, , throughput, stray-light, scan speed * ☐ Choose disperser (grating vs prism) and groove density/blaze * ☐ Select optical layout (Czerny–Turner, Ebert, double) * ☐ Compute dispersion and slit geometry (use formulas above) * ☐ Choose detector and electronics * ☐ Mechanical design and thermal control * ☐ Baffling and stray light control * ☐ Calibration plan and test protocols * ☐ Software and control interfaces

If you want, give me **your target wavelength range, required (or R), and whether you need scanning or an imaging spectrometer**, and I'll run the numbers and propose a concrete optical layout (grating density, focal length, slit widths, expected throughput and a short parts list)."

A.23 Case Study 2 Gemini Outcome

"Designing a monochromator is an engineering process that balances **resolution**, **throughput** (light collection), and **spectral range**. The process typically follows a "V-model" approach: start with high-level requirements, drill down into component selection, and then verify the design through modeling.

Here are the proper steps to take when designing a monochromator.

1. Define the Key Performance Specifications Before selecting any components, you must quantify the instrument's goals. These parameters act as the constraints for your optical math. * **Spectral Range:** What is the full band of wavelengths you need to scan (e.g., 400 nm – 1100 nm)? * **Spectral Resolution (Bandpass):** What is the minimum wavelength difference ($\Delta\lambda$) you need to distinguish? (e.g., 0.1 nm). * **Throughput (f):** How much light do you need? (Lower f-numbers gather more light but increase aberrations). * **Stray Light Rejection:** How critical is signal purity? (Determines if you need a single or double monochromator).
2. Select the Optical Configuration Choose the geometry that best fits your constraints. The **Czerny-Turner** configuration is the industry standard for general-purpose monochromators due to its coma-correcting geometry and flexibility.

* **Czerny-Turner:** Two concave mirrors (collimating and focusing) + one planar grating. Good aberration correction. * **Littrow:** Uses a single mirror for both collimation and focusing. More compact but suffers from higher stray light. * **Concave Holographic:** Uses a single curved grating. Fewer reflections (higher efficiency) and robust, but harder to scan over wide ranges.

3. Select the Diffraction Grating The grating is the heart of the system. Its selection dictates the resolution and efficiency limits. * **Groove Density (d):** Calculate the required lines/mm. Higher density (e.g., 1200 lines/mm) gives better resolution but a narrower usable spectral range. * **Blaze Wavelength:** Choose a blaze angle that peaks efficiency in your primary region of interest (e.g., 500 nm). * **Dispersion Calculation:** Use the grating equation to ensure your grating can physically disperse the light across your exit slit width.

$$n\lambda = d(\sin \alpha + \sin \beta)$$

(Where n is order, d is groove spacing, α is incidence angle, β is diffraction angle).

4. Calculate Focal Length and Slit Widths These two parameters are chemically linked to your resolution. * **Focal Length (f):** Longer focal lengths improve resolution (linear dispersion increases) but result in a physically larger instrument and often a slower f-number. * **Linear Dispersion:** Calculate the linear dispersion (D) at the exit plane:

$$D = \frac{f \cdot n}{d \cdot \cos \beta}$$

* **Slit Width (W):** define the entrance and exit slit widths. Your bandpass (resolution) is roughly:

$$\text{Bandpass} \approx \frac{W}{D}$$

* **Trade-off:** Narrower slits improve resolution but drastically cut signal intensity.

5. Match the Optics (F-number) Ensure the "speed" of your monochromator matches your light source or input fiber. * **f-number Matching:** If your input light is f/4 (diverging cone), your monochromator mirrors should ideally be f/4 or faster (e.g., f/3.5) to prevent overfilling, which causes stray light. * **Optic Size:** Calculate the required diameter of your collimating and focusing mirrors to ensure they capture the full beam without vignetting.

6. Aberration Analysis Optimization Theoretical designs often fail in practice due to optical errors. * **Coma Correction:** In a Czerny-Turner design, you can eliminate coma (comet-shaped blur) at one wavelength by calculating the specific angles of the collimating and focusing mirrors relative to the grating. * **Ray Tracing:** Use optical simulation software (like Zemax or calibration simulations) to trace rays through the system. Check for astigmatism and spherical aberration at the slit extremities.

7. Mechanical Design Drive Mechanism The final step is designing the physical mechanism that moves the optics. * **Sine Drive vs. Direct Drive:** Design the mechanism to rotate the grating. A **Sine Bar** drive is traditional because it linearizes the relationship

between the motor steps and wavelength change. * **Housing:** Design a light-tight box with internal baffling (ridges or varying textures) to trap stray light reflections.

Summary Checklist — Step — Action — Key Formula/Constraint — — : — — : — — : —
 — — ****1**** — ****Specs**** — Define λ range Resolution — — ****2**** — ****Config****
 — Select Czerny-Turner (Standard) — — ****3**** — ****Grating**** — Select lines/mm
 Blaze λ — — ****4**** — ****Optics**** — Calculate Focal Length (f) for Resolution —
 — ****5**** — ****Throughput**** — Match f/λ to source — — ****6**** — ****Simulate**** —
 Ray-trace to minimize aberrations —

Would you like me to walk through a calculation example for a specific resolution (e.g., getting 0.5 nm resolution with a 1200 line/mm grating)?”

A.24 Case Study 2 Grok Outcome

” Steps to Design a Monochromator

Designing a monochromator involves balancing optical, mechanical, and performance requirements to isolate specific wavelengths from a broader spectrum. The process typically follows these key steps, drawing from established optical engineering principles. While specific designs (e.g., prism-based or grating-based) may vary, the overall approach emphasizes defining needs, selecting components, and optimizing for efficiency.

1. ****Define Design Requirements and Specifications**** Start by outlining the key performance parameters based on the intended application, such as spectroscopy or light filtering. This includes the desired wavelength range (e.g., UV to IR), spectral resolution (e.g., 0.01–0.1 nm), bandwidth (FWHM, often 1 nm), throughput (transmittance in the pass band), stray light suppression (e.g., ≤ 0.1

2. ****Choose the Type and Optical Configuration**** Select between prism or diffraction grating as the dispersive element, and decide on the layout (e.g., Czerny-Turner, Fastie-Ebert, or Littrow). Czerny-Turner is common, using separate collimating and focusing mirrors to minimize aberrations. For prisms, opt for materials like F2 glass with a 60° apex angle for refraction-based dispersion. Gratings (blazed or holographic) are preferred for high efficiency and low scattering but require handling parasitic orders. Double monochromators can be used for ultra-low stray light (e.g., one millionth of peak).

3. ****Design the Optical Path and Select Components**** Map out the light flow: entrance slit (e.g., 50 μm for resolution), collimating optics (curved mirror or achromat lens to make rays parallel), dispersive element, focusing optics (another mirror or lens), and exit slit or fiber. Use reflective optics to avoid chromatic aberrations. For gratings, calculate groove density and blaze angle for efficiency. Incorporate apertures or black shields to reduce stray light. If scanning is needed, plan for motorized grating or prism rotation.

4. ****Incorporate Mechanical and Control Systems**** Design mounts for stability, using precision rotation stages (e.g., with 0.04° steps) for wavelength tuning. Add stepper motors for automation, aligned via micropositioners, and compensate for nonlinear

wavelength-angle relationships in software. For high resolution, use toroidal mirrors or curved slits to correct spherical aberrations and allow taller slits for better light gathering.

5. ****Optimize for Performance and Simulate**** Use modeling tools (e.g., MATLAB scripts) to simulate dispersion, bandwidth, and sensitivity based on slit size, focal length (e.g., 50 mm lenses), and angles. Adjust for constant bandwidth in prism designs by varying slit widths. Minimize bandwidth (e.g., 1–2 nm) by using smaller slits or longer focal lengths, but balance with signal-to-noise ratio. Test for independence of center wavelength from input alignment.

6. ****Assemble, Align, and Calibrate**** Assemble components, align using monochromatic sources (e.g., fluorescent lamps) to ensure vertical slit imaging. Calibrate deviation angles and wavelength selection against theoretical models (e.g., Snell's law for prisms). Verify with known light sources like argon-mercury lamps, and integrate detectors (e.g., photomultiplier tubes) for intensity measurement.

7. ****Test and Iterate**** Evaluate the prototype for resolution, stray light, and efficiency using spectrographs or known spectra. Iterate on optics (e.g., switch to off-axis parabolic mirrors for better achromaticity) or mechanics for improvements. For advanced designs, consider variants like echelle gratings for enhanced resolution.

This process ensures a functional, application-specific monochromator. For hands-on builds, start with simple prism setups before advancing to automated grating systems.”

A.3 Case Study 3

A.31 Case Study 3 Prompt

”What would be the most efficient way to have multiple STM32 boards communicate to a Raspberry Pi 5? The data being communicated would come from sensors and stepper motors.”

A.32 Case Study 3 ChatGPT Outcome

”Below is a clear comparison of all the practical communication methods between ****multiple STM32 boards**** and a ****Raspberry Pi 5****, along with which one is most efficient depending on your sensor/stepper-control use case.

—

****Short Answer (Best Overall Solution)****

For ****multiple STM32 boards****, the most efficient architecture is:

****Use RS-485 (Modbus-RTU or custom UART protocol) over a differential bus.****

* One Raspberry Pi 5 many STM32 nodes on one shared twisted-pair cable * Long distance, electrically robust, low noise * Multi-drop capability (unlike I²C/SPI) * Very stable even with motor EMI * Easy to integrate with stepper drivers and sensors

If you need *higher bandwidth* or *complex data*, then use **Ethernet**.

—

****Best Choice: RS-485 Multi-Drop Bus****

Why RS-485?

RS-485 solves everything that I²C/SPI struggle with:

Feature	RS-485	I ² C	SPI	UART
Multi-device bus	Excellent	—	—	—
Unreliable	1–2m	Poor	Point-to-point only	—
Noise immunity	Very high	Low	Medium	Low
Cable length	1–1000 m	1 m	reliable	—
Speed	Up to 10 Mbps	400 kHz–1 MHz	10–20 MHz	115 kbps–1 Mbps
Hardware needed	Cheap MAX485 chip	none	none	none

Architecture

“ RS-485 BUS (half-duplex) + ———— + — RPi5 — USB → RS485 adapter — STM32 1 — STM32 2 — STM32 3 ... ”

Each STM32 has:

* 1 UART peripheral * 1 MAX485/ADM485 transceiver * Optional protocol layer: **Modbus RTU** or a simple custom frame format

Advantages

* **Robust** even near stepper motors (high EMI) * **Only 2 wires** + ground * **Scales easily** to 20+ STM32 nodes * Very low CPU overhead * Raspberry Pi supports Modbus RTU easily ('pymodbus', 'minimalmodbus')

—

Alternative 2: ****CAN Bus****

If your STM32s have CAN (many F1/F4/F7 do):

Why CAN?

* Built-in multi-master arbitration * Very robust, automotive-grade * Handles noise better than RS-485 * Automatic packet retries and CRC * Excellent for sensor networks

Architecture

“ RPi5 (via MCP2515 + SPI) ↓ CAN BUS ↓ STM32 nodes ”

Pros

* Extremely reliable * True multi-master * Works well for motor applications

Cons

* Raspberry Pi has *no native CAN* * Must use an MCP2515 hat (limited to 1 Mbps due to SPI)

RS-485 tends to be simpler and more flexible for non-real-time systems.

—

Alternative 3: **Ethernet (Fastest most scalable)**

If you need:

* high-rate data streaming * many nodes * 100+ meter cables

Use STM32s with Ethernet (F407, F746, H7, etc.)

Pros

* 100 Mbps * Full networking (TCP/UDP, MQTT, REST, ROS2) * Multiple devices on a switch * Very long distances

Cons

* Higher cost * Configuration overhead * MCU must have Ethernet or use ENC28J60 (slow)

Most overkill unless you're building a distributed robotics system.

—

Methods You *Should Not* Use for Multi-Node STM32 RPi Networks

I²C

* Multi-drop electrical spec is weak * RPi pull-ups + STM32 pull-ups often break bus
* Very noisy when stepper motors are involved * Distance limited to **30 cm**

SPI

* Not designed for more than one slave without complexity * Very sensitive to cable

length * Requires chip-select lines for each STM32 * Noisy with motors

Plain UART

Good for *one* STM32 RPi, but not for multiple devices unless you add RS-485.

—

FINAL RECOMMENDATION (Best combination)

If you want:

****Reliability****

****Distance up to tens of meters****

****Multiple STM32s****

****Noise immunity near stepper motors****

****Simple wiring****

****RS-485 bus (Modbus RTU or custom protocol)****

If you want even higher robustness:

****CAN Bus****

If you need huge bandwidth or network scalability:

****Ethernet****

—

If you tell me:

* How many STM32 boards? * How many sensors per board? * Distance between boards and the Pi? * Data rate? (e.g., values per second) * Are the stepper motors near the wiring?

I'll design the exact recommended wiring diagram, protocol, and RPi code."

A.33 Case Study 3 DeepSeek Outcome

"For multiple STM32 boards communicating with a Raspberry Pi 5 in a sensor/motor control system, here are the most efficient approaches ranked by performance and practicality:

****Top Recommendation: CAN Bus****

****Why it's best:**** - ****Deterministic**** communication with arbitration - ****High noise immunity**** (critical for motor environments) - ****Multi-drop**** topology - up to 110 nodes - ****Hardware-level**** message prioritization - ****2-wire**** connection simplifies wiring

****Implementation:**** - STM32: Built-in CAN peripherals + CAN transceiver (e.g., MCP2551/TJA1050) - RPi 5: USB-CAN adapter (e.g., PEAK PCAN-USB) or MCP2515 SPI-CAN module - ****Throughput:**** Up to 1 Mbps

****Excellent Alternative: RS-485****

****Good for:**** - ****Long distances**** (up to 1200m) - ****Multi-drop**** (32+ devices) - ****Noise-resistant**** differential signaling

****Implementation:**** - STM32: UART + MAX485 transceiver - RPi 5: USB-RS485 adapter or GPIO UART - Use Modbus RTU protocol for standardization

****High-Speed Option: Ethernet****

****When to use:**** - ****High bandwidth**** requirements - ****Existing network infrastructure**** - ****Long distances**** needed

****Implementation:**** - STM32: Ethernet PHY (some STM32 have built-in) or W5500 SPI-Ethernet - RPi 5: Built-in Gigabit Ethernet - Protocols: Modbus TCP, MQTT, or custom UDP/TCP

****Other Options (Situational):****

****SPI (for short distances, high speed):**** - ****Pros:**** Very high speed (10+ Mbps), full duplex - ****Cons:**** Short distance only, complex wiring for multiple devices

****I2C (for simple sensor networks):**** - ****Pros:**** Simple 2-wire bus, built-in addressing - ****Cons:**** Limited distance, speed, and device count

****Recommended Architecture:****

“ Stepper Motors STM32 CAN Bus RPi 5 Sensors ”

****Protocol Suggestions:**** - ****CAN:**** Custom binary protocol or CANopen - ****RS-485:**** Modbus RTU - ****Ethernet:**** Modbus TCP or MQTT for IoT applications

****Why CAN Bus Wins for Your Application:**** 1. ****Motor noise immunity**** - critical with steppers 2. ****Deterministic timing**** - no random delays 3. ****Multi-master**** capability 4. ****Error detection**** and fault confinement 5. ****Scalable**** - easy to add more STM32 nodes

****Implementation Tip:**** Use STM32's built-in CAN filters to reduce CPU load on both ends.

For most sensor+stepper motor applications, CAN Bus provides the best balance of reliability, speed, and noise immunity.”

A.34 Case Study 3 Gemini Outcome

”The most efficient way to have **multiple STM32 boards communicate to a Raspberry Pi 5** depends heavily on the **required data rate**, the **physical distance** between the boards, and the need for **robustness** (especially in electrically noisy motor environments).

Given the typical application of sensor data and stepper motor control, where you need a multi-drop bus (one Raspberry Pi master, multiple STM32 slaves), here are the most efficient and recommended options, from most to least recommended:

1RS-485 with Modbus Protocol (Recommended Balance)

This is often the most practical and robust solution for industrial or robotics applications where you have multiple devices over potentially longer cables (up to several thousand feet).

*** Communication Protocol:** **Modbus RTU** over RS-485. Modbus is a simple, well-established master-slave protocol that works well for reading/writing sensor data (registers) and motor commands (coils). *** Physical Layer:** **RS-485** uses differential signaling, which provides excellent **noise immunity**—crucial when dealing with stepper motors that can generate electrical noise. It is inherently a multi-drop bus, allowing you to connect many STM32 boards to a single set of twisted-pair wires (using a single RS-485 transceiver/adaptor on the Raspberry Pi).

[Image of RS-485 network topology]

*** STM32 Interface:** The STM32’s **UART** peripheral is connected to an **RS-485 transceiver IC** (e.g., MAX485, SP3485). *** Raspberry Pi Interface:** The RPi 5’s **UART** (or a USB-to-RS485 adapter) is connected to an RS-485 transceiver. *** Efficiency:** It’s highly efficient for moderate data rates (easily 9600 to 115200 baud, which is more than enough for most sensor readings and motor step commands). The master (RPi) polls each slave (STM32) sequentially, keeping the bus clear.

2CAN Bus (Controller Area Network)

CAN is highly robust, specifically designed for automotive and industrial control systems where reliability and message prioritization are paramount.

*** Communication Protocol:** CAN. This is a message-based protocol where all nodes can transmit, and messages are prioritized based on their ID, ensuring critical motor com-

A.4 Case Study 4

A.41 Case Study 4 Prompt

”What would be the best option for converting an optical signal to an electrical one? Photoresistors, photodiodes, or phototransistors?”

A.42 Case Study 4 Gemini Outcome

”The ”best” option depends entirely on whether you need **speed** (data transmission), **sensitivity** (low light), or **simplicity** (just turning something on/off).

If you are looking for a general rule of thumb for **signals** (information carrying light, like fiber optics or IR remotes): **The Photodiode is generally the best engineering choice**, but it requires more complex circuitry to work well.

Here is the breakdown of why you would choose one over the others.

Quick Comparison

— Feature —	Photodiode	Phototransistor	Photoresistor (LDR)
— : —	: —	: —	: —
— Primary Use —	High-speed data, precision measurement	Object detection, remote controls	Day/night detection
— Response Time —	Fastest (Nanoseconds)	Medium (Microseconds)	Slowest (Milliseconds)
— Sensitivity —	Low (Needs amplification)	High (Internal gain)	High (But very slow)
— Linearity —	Excellent (Linear output)	Poor (Non-linear)	Very Poor
— Output —	Current (very small)	Current (amplified)	Resistance Change

1. Photodiodes: The ”Speed Precision” Choice This is the standard for converting optical **signals** (data) to electrical ones. If you are building a fiber optic receiver, a laser rangefinder, or a high-speed counter, this is your only real option.

Why it’s best: It is incredibly fast and has a linear response (2x light = 2x current). It doesn’t ”saturate” easily. **The Catch:** The electrical signal (current) it produces is tiny. You almost always need an Op-Amp circuit (Transimpedance Amplifier) to convert that tiny current into a usable voltage. **Use for:** Fiber optics, high-speed data links, precise light meters.

2. Phototransistors: The ”High Gain” Choice Think of this as a photodiode with a built-in amplifier. It is a transistor where the ”Base” pin is replaced by a lens that collects light.

Why it’s best: It outputs a much stronger signal than a photodiode, often strong enough to connect directly to a microcontroller pin without an Op-Amp. **The Catch:** It is significantly slower than a photodiode. It takes time for the transistor to ”turn off”

after the light stops, which limits its bandwidth. It works for TV remotes (38 kHz) but fails at high-speed data (100 MHz). * **Use for:** Object detection (beam break), line-following robots, TV remotes.

3. Photoresistors (LDRs): The "Day/Night" Choice These are chemically coated ceramic discs that change resistance based on light.

* **Why it's best:** It is the simplest to use. You treat it just like a resistor. It handles high voltages easily and is "omnidirectional" (sees light from all angles). * **The Catch:** It is "painfully slow". It has a "memory effect" where it takes milliseconds or even seconds to settle back to darkness. It cannot track a signal; it can only track ambient conditions. * **Note:** Many LDRs contain Cadmium, making them banned in some regions (RoHS non-compliant). * **Use for:** Streetlights, night lights, solar garden lamps.

Summary Recommendation

* **Go with a Photodiode** if you need to read data, measure light intensity accurately, or detect pulses faster than 100 kHz. * **Go with a Phototransistor** if you want to detect a beam break or a simple IR signal and want to avoid complex Op-Amp circuitry. * **Go with a Photoresistor** only if you are measuring ambient light levels (like a dusk-to-dawn sensor) and don't care about speed.

What is the frequency or data rate of the optical signal you are trying to capture? I can help you design the amplification circuit if you choose the photodiode."

A.43 Case Study 4 DeepSeek Outcome

"Of course. This is an excellent question, as the choice depends heavily on the specific requirements of your application, particularly regarding **speed** and **sensitivity**.

Here's a quick summary, followed by a detailed breakdown.

Quick Answer:

* **Photodiodes** are the best for **high-speed** applications (like data communication, fiber optics). * **Phototransistors** are the best for **high sensitivity** in medium-speed applications (like object detection, encoders, twilight sensors). * **Photoresistors (LDRs)** are the best for **simple, low-cost** light-on/light-off applications where speed is irrelevant (like a night light).

—

Detailed Comparison

Let's break down the characteristics of each device.

1. Photoresistor (LDR - Light Dependent Resistor)

Principle: A semiconductor whose electrical resistance decreases with increasing light intensity. **Key Advantage:** **Extremely Simple to Use:** Often only needs a series resistor to create a voltage divider. No bias voltage required. **High "Dark" Resistance:** Can be in the M range, making it very sensitive to the presence or absence of light. **Key Disadvantages:** **Extremely Slow:** Response time can be tens to hundreds of milliseconds. Completely unsuitable for any application where the light changes quickly. **Not Spectrally Accurate:** Their response varies significantly with the color (wavelength) of light. **Best For:** Daylight sensors for street lights * Automatic night lights * Simple "is the light on or off?" projects **Avoid For:** Any application involving data transmission, remote controls, or fast-moving objects.

2. Photodiode

Principle: A PN junction that generates a small current (photocurrent) when illuminated. It is typically used in **reverse-bias** mode for best performance. **Key Advantage:** **Very High Speed:** Response times can be in the **nanosecond** or even **picosecond** range. This is the fastest option of the three. **Linear Response:** The output photocurrent is very linear with light intensity, making it good for precision light measurement. **Key Disadvantages:** **Low Output Current:** Generates a very small signal (microamps or less), often requiring a transimpedance amplifier (TIA) circuit to convert the current to a usable voltage. **Less Sensitive** than a phototransistor for simple detection circuits. **Best For:** Fiber optic communication * High-speed data links (e.g., IR remote control receivers) * Laser rangefinders (LIDAR) * Precision scientific instruments (spectrophotometers) **Avoid For:** Simple projects where you want a large output signal without additional circuitry.

3. Phototransistor

Principle: Essentially a bipolar transistor where light acts on the base region. The light-generated base current is amplified by the transistor's gain (or hFE), resulting in a much larger collector current. **Key Advantage:** **High Sensitivity / High Output:** It provides significant current gain, often producing a milliamp-level output that can directly drive LEDs, relays, or logic inputs without needing an amplifier. **Key Disadvantages:** **Slower than Photodiodes:** Response times are in the **microsecond** range due to the larger junction capacitance and the nature of the transistor action. Fast enough for many applications but not for very high-speed data. **Non-linear Response:** The gain of the transistor introduces some non-linearity between light intensity and output current. **Best For:** Object detection (e.g., slot-type interrupters, limit switches) * Rotary encoders * IR receiver modules (for slower remote protocols) * Twilight sensors for relays **Avoid For:** Gigabit fiber optics or other very high-frequency applications.

—

Decision Table

— Feature — Photoresistor (LDR) — Photodiode — Phototransistor — — : — — : —

— :— — :— — — ****Speed**** — Very Slow (ms) — ****Very Fast**** (ns/ps) — Medium (μ s) — — ****Sensitivity**** — Good (Resistance Change) — Low (needs amp) — ****Very High**** (has gain) — — ****Output Signal**** — Resistance Change — Low Current — ****High Current**** — — ****Linearity**** — Poor — ****Excellent**** — Fair — — ****Circuit Complexity**** — ****Simplest**** — Most Complex (needs TIA) — Simple — — ****Cost**** — ****Very Low**** — Low to Medium — Low —

Final Recommendation

* Choose a ****Photodiode**** if your primary concern is ****speed**** (e.g., you're measuring a rapidly changing light signal or transmitting data). * Choose a ****Phototransistor**** if your primary concern is ****sensitivity and simplicity**** (e.g., you need a strong signal to trigger a microcontroller or switch a relay, and speed is not critical). * Choose a ****Photoresistor (LDR)**** if you are building a simple, low-cost light-activated switch where speed doesn't matter at all."

References

- [1] Sal Afzal and Analog Devices. I2c primer: What is i2c? (part 1), Sep 2016.
- [2] Jonathan Bearfield and Texas Instruments. Msp power supply products usb universal serial bus.
- [3] Wonjoon Choi, Ryan Quny Ya Ong, and Matthew Smith. *Design of a Cooke Triplet*. MIT, 2009.
- [4] Roger Cicala. The seven deadly aberrations, Oct 2010.
- [5] Alex Clark and contributors. Pillow: Python imaging library. <https://pillow.readthedocs.io/en/stable/>, 2024. Accessed: 2025-09-05.
- [6] Wikimedia Commons. File:mustache distortion.svg — wikimedia commons, the free media repository, 2020. [Online; accessed 18-November-2025].
- [7] Wikimedia Commons. File:pincushion distortion.svg — wikimedia commons, the free media repository, 2021. [Online; accessed 18-November-2025].
- [8] Wikimedia Commons. File:spherical aberration 2.svg — wikimedia commons, the free media repository, 2021. [Online; accessed 18-November-2025].
- [9] Wikimedia Commons. File:field curvature.svg — wikimedia commons, the free media repository, 2022. [Online; accessed 18-November-2025].
- [10] Wikimedia Commons. File:astigmatism.svg — wikimedia commons, the free media repository, 2023. [Online; accessed 18-November-2025].
- [11] Wikimedia Commons. File:barrel distortion.svg — wikimedia commons, the free media repository, 2023. [Online; accessed 18-November-2025].
- [12] Wikimedia Commons. File:comparison axial lateral chromatic aberration.svg — wikimedia commons, the free media repository, 2024. [Online; accessed 18-November-2025].
- [13] Wikimedia Commons. File:lens coma.svg — wikimedia commons, the free media repository, 2025. [Online; accessed 18-November-2025].
- [14] Piyu Dhaker and Analog Devices. Introduction to spi interface, 2018.
- [15] Digikey. Opa328dbvr — digikey electronics, 2018.
- [16] Jefferson Eco, Aldrick Limjoco, and Analog Devices. An-1368: Ferrite bead demystified, 2025.
- [17] Epoch AI. Notable ai models dataset, 2024. Accessed: 2025-10-29.
- [18] M. Fabian, Elfed Lewis, Thomas Newe, and Steffen Lochmann. Optical fibre cavity for ring-down experiments with low coupling losses. *Measurement Science and Technology*, 21:094034, 07 2010.
- [19] Python Software Foundation. asyncio — asynchronous i/o. <https://python-pillow.org/>, 2024. Accessed: 2025-09-05.
- [20] Python Software Foundation. logging — logging facility for python. <https://docs.python.org/3/library/logging.html>, 2024. Accessed: 2025-09-05.
- [21] Python Software Foundation. Csv file reading and writing. <https://docs.python.org/3/library/csv.html>, 2025.
- [22] Raspberry Pi Foundation. Picamera2 python library. <https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf>, 2024. Accessed: 2025-09-05.
- [23] Raspberry Pi Foundation. *Raspberry Pi I²C Interface Documentation*, 2024.
- [24] Eric Gazoni and contributors. openpyxl: A python library to read/write excel 2010 xlsx/xlsm files. <https://openpyxl.readthedocs.io/>, 2024. Accessed: 2025-09-05.
- [25] Gondolindrim. A single-push reset circuit for stm32 - acheron docs, 2019.
- [26] Topcon Healthcare. Solos automatic lens analyzer, May 2024.
- [27] Thomas Hopkins and STmicroelectronics. An235 application note stepper motor driving, 2012.
- [28] Amazon Inc. 200 pcs mini square mirror tiles, 2025.
- [29] Edmund Optics Inc. All about diffraction gratings, 2025.
- [30] Newport Inc. Differences between ruled and holographic gratings, 2025.
- [31] NumFOCUS Inc. Python data analysis library. <https://pandas.pydata.org/>, 2024. Accessed: 2025-09-05.
- [32] Texas Instruments. Usb type-ctm: Will your esd solution protect the port?
- [33] Robert Keim. Understanding the photodiode equivalent circuit, Dec 2020.
- [34] Robert Keim. Design tips for photodiode amplifiers, Jan 2021.
- [35] Soyeon Kim, Muhammad Jahandar, Jae Hoon Jeong, and Dong Chan Lim. Recent progress in solar cell technology for low-light indoor applications. *Current Alternative Energy*, 3(1):3–17, Jan 2019.
- [36] Lamptech. Osram-stud short arc xenon xbo1001, 2025.
- [37] libretxts. 24.3: Lenses, Sep 2018.
- [38] Chris Liechti. Python serial port extension. <https://pyserial.readthedocs.io/>, 2024. Accessed: 2025-09-05.
- [39] Linux Kernel and Raspberry Pi Documentation Team. *Linux Serial and I2C Interface Configuration Guide*, 2024. Accessed: 2025-10-23.
- [40] Mike McCauley. Logging facility for python. <https://www.airspayce.com/mikem/arduino/AccelStepper/>, 2024. Accessed: 2025-09-05.
- [41] Kiva McCracken. Accessible optics, 2025.

- [42] Allegro MicroSystems. Features and benefits low r ds(on) outputs automatic current decay mode detection/selection mixed and slow current decay modes synchronous rectification for low power dissipation internal uvlo crossover-current protection 3.3 and 5 v compatible logic supply thermal shutdown circuitry short-to-ground protection shorted load protection five selectable step modes: full.
- [43] Newport. Kbx022ar.14 bi-convex lens, 2025.
- [44] Edmund Optics. Eco-550 glass datasheet.
- [45] Organisation for Economic Co-operation and Development (OECD). Developments in artificial intelligence: Markets and model indicators, 2025. Accessed: 2025-10-29.
- [46] M. Ott et al. Mapping global dynamics of benchmark creation and saturation in ai. *arXiv preprint arXiv:2203.04592*, 2022.
- [47] R. Paschotta. Optical glasses. *RP Photonics*, Jan 2019.
- [48] Eric Peña, Mary Grace Legaspi, and Analog Devices. Uart: A hardware communication protocol understanding universal asynchronous receiver/transmitter, 2020.
- [49] Raspberry Pi. Camera - raspberry pi documentation, 2016.
- [50] Arduino Project. Servo library. <https://www.arduino.cc/reference/en/libraries/servo/>, 2024. Accessed: 2025-09-05.
- [51] Python Software Foundation. *Python 3 Standard Library Documentation*, 2025. Accessed: 2025-10-23.
- [52] Raspberry Pi Foundation. *Raspberry Pi Documentation*, 2024. Accessed: 2025-10-23.
- [53] Horiba Scientific and Analytical Instruments. Diffraction gratings ruled and holographic, 2012.
- [54] Mark Sellers and Texas Instruments. Ldo basics: Preventing reverse current.
- [55] Stanford Institute for Human-Centered Artificial Intelligence. Ai index report 2022, 2022. Accessed: 2025-10-29.
- [56] Inc StellarNet. Green-wave low cost spectrometer, 2025.
- [57] Inc StellarNet. Spectrawiz spectroscopy suite, 2025.
- [58] STmicroelectronics. An5042 application note how to calibrate the hse clock for rf applications on stm32 wireless mcus.
- [59] STMicroelectronics. *STM32Cube MCU Package User Manual*, 2024. UM1718, Rev 44.
- [60] H.D Taylor. Cooke triplet, US Patent 568052A, 1896.
- [61] OpenCV team. Opencv-python: Open source computer vision library. <https://opencv.org/>, 2024. Accessed: 2025-09-05.
- [62] TELEDYNE. Ccd vs cmos, 2025.
- [63] The Pandas Development Team. *Pandas Library Documentation*, 2025. Accessed: 2025-10-23.
- [64] ThorLabs. N-bk7 plano-convex lenses, 2023.
- [65] ThorLabs. Storage, 2023.
- [66] Inc. Thorlabs. Visible ruled reflective diffraction gratings, 2025.
- [67] Contributors to. 24-pin reversible-plug serial digital connector system designed to replace all other usb connectors by carrying all usb data protocols (and some others) and power levels (in either direction, up to 240 w), Dec 2013.
- [68] Trioptics. Stationary thickness gauge, 2015.
- [69] Yi-Ting Tu. Phydemo ray-optics simulator, 2025.
- [70] Georgia State University. Dispersion, 2025.
- [71] u/WithGreatRespect, 2025.
- [72] Herminso Villarraga Gómez. Magneto-optical studies of cobalt-doped nickel oxide thin films. *Graduate School of the University of Puerto Rico*, page 56, 08 2010.
- [73] Polymaker Wiki. Ht-pla-gf, Aug 2025.
- [74] Zygo. Micro lens process metrology inspection equipment, 2025.