Automated Laser Beam Characterizer

Group 3 Authors:

Zachary Andreasen	Thomas Cunningham	Lynzie Smith
Optical Engineering and Electrical Engineering	*Electrical Engineering*	*Computer Engineering*

Review Committee:

Dr. Sean Pang	CREOL	Associate Professor
Dr. George Atia	ECE	Associate Processor
Dr. Peter Delfyett	CREOL/ECE	UCF Trustee Chair/ECE Joint Faculty

Advisors:

Dr. Saleem Sahawneh	ECE	Lecturer
Dr. Aravinda Kar	CREOL	Professor

Table of Contents

List of Tables

List of Figures

Chapter 1: Executive Summary

Lasers are fundamental tools in modern technology, used in applications ranging from barcode scanners and telecommunications to lidar and medical imaging. Accurate characterization of a laser beam – its polarization, wavelength, beam waist, divergence, and beam-quality factor (M^2) - is essential to ensuring optical performance in these applications. Commercial instruments capable of performing such measurements exist but are often costly and limited in scope. The Automated Laser Beam Characterizer project sponsored by Lockheed Martin was conceived to provide a unified, automated, and cost-effective solution capable of performing multiple key laser diagnostics within a single compact system

The Laser Beam Characterizer integrates four major measurement subsystems: a polarimeter, a Michelson interferometer, a beam profiler, and a M^2 factor analyzer. Together, these modules measure the polarization state, wavelength, beam profile, and quality of a laser beam. The system is designed around an ESP32 microcontroller, chosen for its real-time motor control precision and high-speed data acquisition capabilities. Supporting electronics include stepper motors for precise translation and rotation, servo motors for beam steering, and photodiode-based detectors for high-accuracy optical power measurements.

The optical subsystem employs high-precision components – such as a quarter-wave plate (QWP), linear polarizer, beam splitter, focusing lens, and power meter – to ensure measurement fidelity across a wavelength range of approximately 2 μm. The optical configuration is arranged sequentially so that each measurement will build upon the previous one, beginning with polarization analysis. Mirrors and motorized mounts are used to direct and align the laser beam between subsystems without manual intervention, ensuring stability and repeatability of results.

The software architecture is divided between embedded firmware on the ESP32 and a PC-based graphical user interface (GUI). The MCU executes a deterministic loop of motion, sampling, and communication, while the GUI provides a real-time visualization of measurements, live plotting of intensity versus angle or position, and data export functions. Communication between the two systems occurs via USB with data logged automatically for later analysis. The control software enables the user to select measurement modes, set scanning parameters, and view calculated results such as Stokes parameters, beam width, and M^2 factor.

This document will serve as a record of the Automated Laser Beam Characterizer's design process. It starts off with the context and history behind the project and goes into the research of what technologies exist for this type of instrument and what we selected for the design. Next will be standards and design constraints the project will follow and have implemented. Following that will be the schematics, screenshots, and flowcharts for hardware and software of the instrument and finally the prototype/fabrication process along with the testing and integration of the systems. The final conclusion will wrap up the document at the end with appendixes and references included afterwards.

Chapter 2: Project Description

2.1: Motivation and Background

Since its invention by Theodore Maiman in 1960, the laser has become integral in many applications today. It is used in everyday objects such as bar code scanners and laser pointers, as well as more sophisticated applications such as lidar and imaging systems. When constructing these systems, it is often critical to know precisely the state of the laser beam being emitted. Basic characterizations of the laser beam include its divergence and beam waist, its M^2 factor, as well as its polarization and wavelength. While there are other aspects of the laser that are useful to know, many are either not as significant, or require interferometric techniques. This project entails designing a single system that measures all these factors. This can be useful in many applications which make use of laser technology. For instance, most laser applications require the laser beam to be as close to diffraction limited as possible, which requires an M^2 factor as close to one as possible. Additionally, many applications require knowledge of the polarization state of the laser, such as coherent lidar, imaging, as well as many laser welding and machining technologies. Similarly, many nonlinear effects in optics are affected by polarization, such as orders of harmonic generation, Pockels Effect, the Kerr effect, and many more. Thus, the state of any laser beams polarization is also critical to knowing how it will interact with other optical systems, and the accurate characterization of these things becomes essential to knowing how it will behave in a system.

2.2: Current Technologies and Existing Products

Presently, several technologies exist which measure some of these factors, however, none of them coexist in the same system. They can be broken down into four general systems. These are profilometers, polarimeters, spectrometers, and M^2 factor measurement devices. These will all be implemented into the system and are each discussed individually in the sections that follow.

2.2.1: Beam Profilers

Laser beam profilers come in two different types, based on their method of taking measurements. These two methods are the scanning slit method and the image sensor method. Both methods are used to determine the defined beam width of the laser beam at their aperture.

2.2.1.1: Scanning Slit Method

The scanning slit method consists of conducting several knife edge tests using a thin rotating fan blade in front of a power meter. The knife edge test works by recording the power on the detector as a thin blade crosses in front of the path of the beam. The derivative of the error function curve that results from this process is the profile of the beam. Then, the $1/e^2$ point on both sides of the profile is calculated and the distance between the two points is recorded. Assuming the beam is propagating in the fundamental mode, this distance is the diameter of the beam.

Below are some scanning slit profilometer products that can be found currently on the market today. The table below compares the products specifications.

Table 1: Scanning Slit Products			
Company	Ophir Optronics [1]	Thor Labs [2]	DataRay [3]
Part Number	PH00470	BP209-IR2	BeamMap2
Spectral Range	190 nm – 100 um	900 nm – 2700 nm	650 nm – 1800 nm
Power Range	5 mW – 100 W	1 mW – 10 W	< 1 W
Max Input Beam Diameter	6 mm	9 mm	3 mm

2.2.1.2: Image Sensor Method

The image sensor method simply uses an array of optoelectronic devices such as a photodiode array, CCD, or a CMOS sensor to visually record the spatial distribution of power of the laser beam on the sensor. Assuming the beam does not saturate the sensor, the peak value can be easily found, and a beam size can be calculated from the image using the $1/e^2$ point as discussed in the previous section. Below are a couple of examples of products that can be used to do this.

Table 2: Image Sensor Profilometer Products		
Company	Ophir Optronics [4]	Allied Vision [5]
Part Number	SP90404	CL-034 XSWIR
Sensor Type	InGaAs CMOS	InGaAs
Spectral Range	1.06 um – 3000 um	900 nm – 1700 nm
Dynamic Range	60 dB	69 dB
Sensor Size	320 x 320	636 x 508
Pixel Pitch	80 um	15 um

2.2.2: Polarimeters

When measuring the polarization of light, the Stokes Parameters are generally used to completely characterize the wave. While the Jones Vector can also be used, it is limited to beams that are fully polarized and therefore cannot be used in cases where the light is unpolarized or partially polarized. Therefore, the Stokes Parameters provide a more complete characterization of the polarization of light.

There are two methods that can be used to measure the Stokes Parameters. One is the Classical Method, and the other is the Rotating Quarter Waveplate Method. Both require a quarter waveplate (QWP), a linear polarizer (LP) and a power meter (PM).

Table 3: Polarimeter Products		
Company	Thor Labs [6]	OZ Optics [7]
Part Number	PAX1000IR2(/M)	ER-100-IR
Spectral Range	900nm – 1700 nm	850 nm – 1650 nm
Sampling Rate	30 – 400 Samples/s	Unknown
Poincare Sphere Accuracy	±0.25°	Unknown
Aperture	3 mm	Fiber coupled

2.2.3: M^2 Factor Measurements

The M^2 factor of a Gaussian beam is a number that represents the quality of the beam. In an ideal beam, the M^2 factor is one. However, if the beam is aberrated, the beam quality degrades and the M^2 factor increases. Mathematically, this is described by the equation below, where θ is the far field divergence of the beam and w_0 is smallest spot size of the beam, known as the beam waist. The M^2 factor is measured using a profilometer and measuring the size of the beam at different points along its axis of propagation. Since the Rayleigh range of most lasers is much too long to practically measure enough points to characterize the beam quality, a lens is placed in the path of the beam to shorten the Rayleigh range to a measurable value. This lens must not induce any aberrations to the beam or else the M^2 measurement will not be accurate. With beam divergence and waist much easier to measure, the profilometer can be used to measure the M^2 factor [8]. More will be said on this measurement procedure in Chapter 4 in the Final Report which will discuss the ISO Standards for this measurement.

$$\theta = M^2 \frac{\lambda}{\pi w_0}$$

Below are a couple of products that can be found in the market today that measure M^2 this very way.

Table 4: M2 factor Measurement Products		
Company	Thor Labs [9]	Gentec-EO [10]
Part Number	M2MS-BP209IR2	BEAMAGE-M2
Spectral Range	900 – 2700 nm	350 – 1100 nm
Lens Focal Length	250 mm	200 mm – 500 mm
Effective Travel Range	200 mm	400 mm
Aperture Size	9 mm	48 mm
Max Power Density	< 1W*	CW: 10W/cm^2 @1064
Measurement Time	15 – 30 s	45 s
Measurement Accuracy	±5%	±5%

*See Figure 1 below [2]

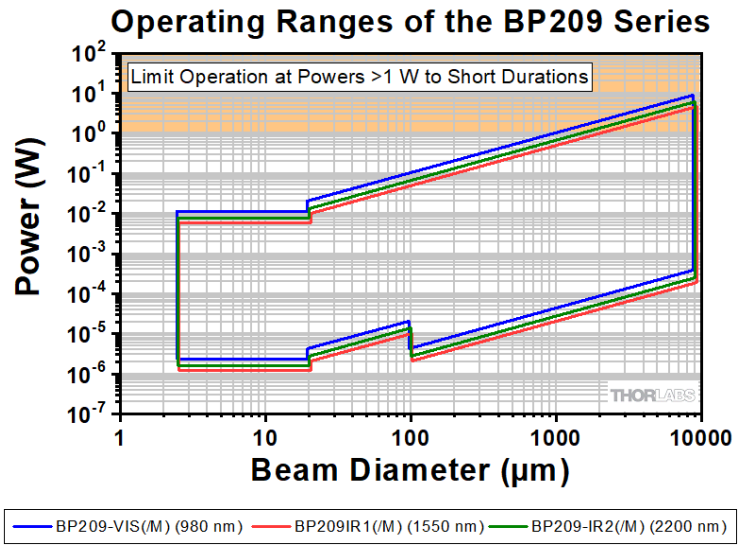


Figure 1: Thor Labs BP209 Power Density Safe Region

2.2.4: Spectrometers

Spectroscopy has its roots much deeper in history than most of the rest of the topics in this project. Starting with Isaac Newton first showing how white light disperses through a prism, to Thomas Young demonstrating the wave nature of light in his famous double slit experiment, culminating in William Hyde Wollaston and Joseph von Fraunhofer independently building the first spectrometers, the wavelength of light became essential to its characterization. Most spectrometers today use a diffraction grating to disperse light into its spectral lines. Below are some commonly used spectrometers in industry today.

Table 5: Spectrometer Products			
Company	Ocean Optics [11]	StellarNet [12]	B&W Tek [13]
Part Number	NIRquest-512-2.5	DS-NIR-512	BWS005A-20
Spectral Range	900 – 2500 nm	900 – 1700 nm	1100 – 2200 nm
Resolution	6.3 nm	1.25 nm	13 nm
Slit Size	10 – 200 um	25 um	N/A

2.2.5: Summer/Fall 2022 UCF Senior Design Project

As it happens, a previous UCF senior design project has done something similar, though for a different purpose. In the summer and fall of 2022, a group sponsored by CMS Laser built a system that was designed to measure the beam profile and M^2 factor of a laser beam. The group was comprised of one Optical Engineer, two Electrical Engineers, and one Computer Engineer. Their application, however, focused on high power lasers, and thus their application included several reflective wedges that were used to cut the power down enough to make the required measurements. The system built for this project, however, will have a different focus, since the task consists in integrating several measurements into one system.

2.3: Goals and Objectives

In this section, the basic, advanced, and stretch goals and objectives are outlined. The goals describe some of the broad overarching things this project hopes to achieve, whereas the objectives outline the specific ways in which this project will implement technologies to achieve its goals. The basic goals and objectives should be relatively easy to achieve, whereas the advanced goals and objectives represent the more complex and difficult tasks, and the stretch goals and objectives represent those that will only be implemented if there is found to be extra time and resources to do so.

Table 6: Goals and Objectives

	Goals	Objectives
Basic	Measure Polarization	Motorized Quarter Waveplate Fixed Linear Polarizer Power Meter
	Measure Wavelength	Michelson Interferometer Photodiode Circuit
	Measure M^2 factor	Long Focal Length Lens Power Meter with scanning slits
	Measure Divergence	Homing location for Power meter at focal point of Lens
	Light weight	Ensure total system is under 30 lbs.
	Under Budget	Follow Bill of Materials as close as possible
	Easy User Experience	Implement reference mirror and/or irises for ease of alignment User friendly GUI
Advanced	Allow for large input power	Design all components to function at necessary power density level Implement removable ND filter
	Allow for large input beam size	Design components to have at least 40 mm clear aperture diameter
	Detect potential measurement errors in M^2	Implement checks into data processing of parameters
Stretch	Measure Input Waist Size and Location	Use M^2 measurements to calculate waist parameters
	Allow for broad range of input wavelengths	Use achromatic components Use wavelength measurement to adjust for different wavelengths
	Include M^2 measurement error correction procedures	Implement algorithms to fix errors in measurements that do not follow ISO standards

	Report Beam Wavelength and Polarization measurements live	Implement necessary communication protocol with PC to report new measurements continuously
	Allow for pulsed lasers	Make scanning slits fast enough to be able to profile moderately pulsed lasers

2.4: Optical System Design

2.4.1: Full Optical System

For the optical system design, the challenge becomes integrating each of the four measurements into one system. The divergence measurement can be easily integrated into the M^2 factor measurement, since both measurements require taking beam profiles after a focusing optic. Therefore, there are three main subsystems to the project: the Polarimeter Subsystem, the Interferometer Subsystem, and the Beam Profiling Subsystem. These are all integrated into one system as shown in the picture below.

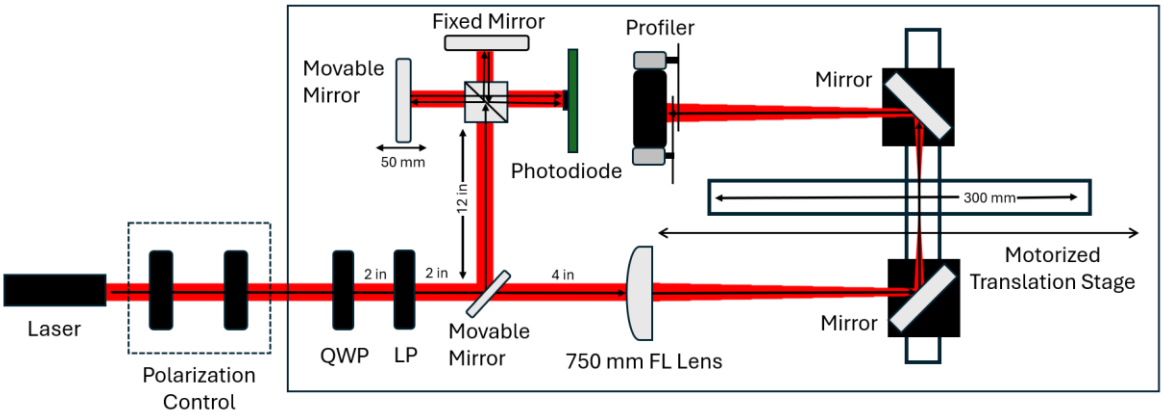


Figure 2: Optical Layout with three subsystems

As shown above, the three subsystems are integrated in a specific order. The first system that the laser beam will encounter is the Polarimeter subsystem. This has many advantages to being the first subsystem. To begin, it removes the variable of the polarization changes that can occur in the other subsystems. Many optical components, even simple mirrors, can reshape the polarization of a laser beam as the beam encounters it. Thus, if the laser beam were to encounter the polarimeter after going through another optic, there would be another variable that affects the polarization of the beam. This makes the measurement accuracy much more uncertain and is not ideal for the system. Additionally, though the polarimeter requires the use of a power meter, this detector can easily be shared by the M^2 factor profilometer which will be built from a power meter with scanning knife edges attached to the edges of it. Finally, this allows for control over the power of the beam in the other subsystems. For example, since the photodiode current is proportional to the amount of optical power on the device, if there is too much power, the voltage signal produced by the transimpedance amplifier could saturate and potentially damage some of the components in the system. Therefore, it is critical to have some control over how much power the

photodiode sees. This can easily be done if the polarimeter is first in the system, since the rotation of the quarter waveplate will change the power output from the linear polarizer. These offer great advantages to the polarimeter being first in the system.

After the polarimeter, there are two other subsystems the laser beam must interact with, those being the interferometer and the beam profiling subsystems. In both subsystems, the beam must terminate in a sensor. Therefore, it will be necessary to design the system to be able to direct the beam to both subsystems separately. There are two ways to do this. The first is to use a beam splitter. This optical component will split the beam into a transmitted and a reflected beam, which could be placed after the linear polarizer in the polarimeter to direct the beam to each of the two other subsystems. One of the main advantages of this method is that it does not require any moving parts, which can be prone to misalignments. The main issue with this method, though, is the same issue mentioned above concerning the amount of optical power on the photodiode. Though there may be less power than if the beam were directly on the photodiode, there is still the possibility of saturating the voltage gain and damaging components. Therefore, the second method will be used. This involves attaching a mirror to a servo motor and controlling the mirror placement with the microcontroller. This will give the flexibility for the quarter waveplate to rotate to a position where a reasonable amount of optical power can be incident on the photodiode and avoid any of the issues mentioned.

When looking for components, there were many manufacturers with many different products to consider. The table below lists the major components of each optical system that are particularly critical for the success of the project. The selection for each of these is then discussed in the following sections.

Table 7: Required Optical Component Considerations

Subsystem	Component
Beam Profiling	Focusing Optic
	Power Meter
Michelson Interferometer	Beam Splitter
	Photodiode
Polarimeter	Quarter Waveplate
	Linear Polarizer
	Power Meter (Same as Beam Profiling)

2.4.2: Focusing Optic

When it comes to focusing optics, there are three main factors that need to be considered. The first is simply the clear aperture of the optic. This is a straightforward requirement as

if the aperture is not large enough, the beam will be clipped. This is very undesirable as it could affect the M^2 factor and the polarization measurements. However, even though the maximum input beam requirement is 35 mm, the clear aperture of the optics used must be at least 42 mm in diameter to meet the desired power specification. For more information on the calculation of this value, see Chapter 6.

The second requirement is the focal length. This requirement is because ISO Standard 11146 requires that enough points be taken both close to the waist and far away from it in order to get an accurate M^2 factor measurement. Therefore, to have enough space to measure the required points, the focal length of the optic must be at least 600 mm. See Chapter 6 for an explanation of this calculated value.

The last requirement is wavefront error. All optics have surface imperfections on them. If these are substantial enough, they can degrade the beam and increase the M^2 factor. This would cause a major issue in the system as the measured M^2 factor would no longer reflect the actual M^2 factor of the input beam. There are several metrics for expressing surface quality, but in total the optic should not have more than $\lambda/8$ wavefront error.

With these factors in mind, the focusing optic can be selected. There are two types of focusing optics that can be used in this application. The first is a simple lens, and the second is an off-axis parabolic mirror. An off-axis parabolic mirror (OAP) is a section of a parabolic mirror that focuses reflected light at an angle. There are many advantages to this kind of mirror. To begin, there are no chromatic shifts in the focal point of the optic. This would make measuring the divergence of the beam rather easy, since the profiler would always measure the beam at the same place for any beam. However, there are many disadvantages. Many OAP mirrors do not have very good surface quality and therefore can reasonably be expected to degrade the quality of the beam. While there are some places where good surface quality can be achieved, these options are generally very expensive and have very long lead times, something not as feasible for this project. Therefore, a long focal length lens was chosen instead. The comparison of some examples of these products is tabulated below.

Table 8: Focusing Optic Comparison

Parameter	Lens	OAP 1	OAP 2
Manufacturer	Thor Labs	Edmund Optics	SORL
Part Number	LA4745-ML	#35-597	OAP40-075-02Q
Coating	Uncoated	Aluminum	AlSiO
Focal Length (mm)	750	646	1016
Clear Aperture (mm)	45.72	68.58	45.72
P-V Wavefront Error @ 2 um	$\lambda/12$	$3\lambda/2$	$\lambda/25$
Cost	$250	$810	$7000
Lead Time (weeks)	~ 2	~ 2	10
Chosen	Yes	No	No

2.4.3: Power Meter

When it comes to optical power meters, there are many different methods used. Two of the most common methods for measuring optical power in the NIR region are photodiodes and thermal sensors. With photodiode power meters, the light signal is converted into an electrical current which can be easily processed digitally. These types of detectors offer very fast response times as well as high sensitivity, making them ideal for applications requiring very accurate power readings and fast communication speeds. On the other hand, there are other sensors that use a thermal detection system to measure the heat generated by the light incident on the detector and use that to calculate the power of the beam. There are two main advantages of this detector. First, they have very high-power intake, often upwards of several hundred watts. Also, they can usually have a very large aperture size, something that most photodiode sensors lack, since the photodiode must be very small to function properly. The main downside to this though is that the response time is usually much slower than most photodiode sensors. Since the priority is to scan across the path of the beam very fast with a scanning slit, a fast response time is essential for getting an accurate beam profile measurement. Additionally, the thermal detectors tend to be more susceptible to noise, making it more difficult to get accurate measurements at lower beam powers, which is necessary for this application. Therefore, since the speed and low power accuracy are much better in the photodiode sensor, it was chosen as the detector to use for this project. The advantages and disadvantages of each of these detectors are tabulated below.

Table 9: Power Meter Comparison

Parameter	Thor Labs	Gentec-EO
Part Number	S148C	UP12E-20H-H5-D0
Power Range	1 µW – 1 W	20 W
Spectral Range	1.2 – 2.5 um	0.193 – 20 um
Response Time	< 1 µs	0.3 s
Aperture Size	Ø 5 mm	Ø 12 mm
Cost	$1000	$1166
Chosen	Yes	No

2.4.4: Beam Splitter

When using a Michelson Interferometer, it is necessary to split the beam into two paths in order to create the interference pattern. This puts restrictions on how the optic splits the laser beam. To begin, it must not be polarization dependent. Some beam splitters are designed to reflect s-polarized light but transmit p-polarized light. This is undesirable for a Michelson Interferometer as the beams in the two arms will not merge in the desired interference path but rather merge back along the transmit path. This is not only potentially reflecting light back into the laser source, which is potentially damaging to it, but is also counterproductive to the purpose of the interferometer. Additionally, some beam splitters have a small split ratio, where for instance 10% of the light is transmitted and 90% of the light is reflected, or vice versa. This is also undesirable for the Michelson interferometer because it will reduce the contrast of the interference fringes. To get the best contrast, the

two beams must have roughly equal powers, meaning the beam splitter must split half of the beam power in each direction. Many beam splitters do this, so when deciding between many options, the splitter with the most even split over the broadest bandwidth should be selected. Comparison of some of these is shown below.

Table 10: Beam Splitter Comparison			
Parameter	**BS1**	**BS2**	**BS3**
Part Number	UFBS50502	BSW23	BSW511
Spectral Range	1 – 2 um	0.9 – 2.6 um	1 – 6 um
Polarization Dependence	Little	High	Moderate
Split Ratio	50:50	50:50	50:50
Ghost ray	No	No	Yes
Cost	$380	$350	$360
Chosen	Yes	No	No

2.4.5: Photodiode

The photodiode selected will be used in the Michelson Interferometer to detect the change in the interference pattern as the mirror in one arm is moved. Because of the relatively long wavelength of this project's desired bandwidth, it will be necessary to use an extended InGaAs photodiode. Ordinarily, InGaAs has a spectral range up to about 1.7 um, however, through different manufacturing techniques beyond the discussion of this project, some can extend this wavelength range to 2.5 um. These are some of the only photodiodes that will be feasible for this project. Most of these have very similar specifications, and so there was not much deliberation concerning which photodiode to use, but rather one was found with reasonable specifications, shown below. [14]

Table 11: MTPD2601T-100 Photodiode Specifications		
Parameter	**Value**	**Units**
Spectral Range	800 - 2600	nm
Responsivity at 2 um	1.1	A/W
Dark Current	300	uA
Max Reverse Bias Voltage	1	V
Junction Capacitance	85	pF
Cost	19.45	$

2.4.6: Quarter Waveplate

When it comes to the quarter waveplate, the main requirements are that it can take a reasonable amount of optical power density to satisfy the input beam power requirements and that is has a large enough clear aperture that it does not clip the beam. Beyond this the only other specification is that the waveplate retardance be reasonably close to quarter wave. Because the refractive index of a material is dependent on wavelength, it is to be expected that across the wavelengths which will be able to use this device, the retardance of the waveplate will vary. Therefore, to get an accurate polarization measurement, it is necessary to know the wavelength of the beam in order to factor in the change in retardance. If this retardance value is known, the Stokes parameters can be scaled appropriately. For this reason, when selecting the waveplate, the design wavelength is rather flexible. What is more important, though, is the aperture size of the waveplate. This is actually a very difficult task as most waveplates are not manufactured bigger than 1 inch in diameter. The component selected was found from Thor Labs, and inquiry was made into the possibility of custom parts. The advantages and disadvantages of these options are tabulated below, but the custom option was rejected as its cost and lead time make it not feasible for this project.

Table 12: Quarter Waveplate Comparison

Parameter	Option 1	Option 2
Part Number	WPQ20ME-1650	Custom
Design Wavelength	1650 nm	2000 nm
Clear Aperture	Ø 48.5 mm	Ø 50.8 mm
Cost	$550	$7000
Lead Time	< 2 weeks	8 weeks
Chosen	Yes	No

2.4.7: Linear Polarizer

For the linear polarizer, it is necessary that the extinction ratio be sufficiently high enough to get an accurate polarization measurement. As stated in the system specifications below, it is sought that the Stokes parameters be calculated to within 5% for all parameters. This implies that the Polarization Extinction Ratio (PER) of the linear polarizer must be at least 11.5 dB. See Chapter 6 for a more in-depth explanation of this calculation. While this might be sufficient, a larger extinction ratio is still be desirable to allow for precise measurements. Additionally, the clear aperture of the polarizer must not clip the beam but must be at least 42 mm in diameter as discussed above. A comparison of a couple of products currently on the market is shown below.

Table 13: Linear Polarizer Comparison

Parameter	Option 1	Option 2
Manufacturer	Thor Labs	Edmund Optics
Part Number	WP50L-UB1	#26-996
Clear Aperture	48 mm x 48 mm	42 mm
PER at 2 um	29 dB	37 dB

Cost	$2250	$1900
Chosen	Yes	No

2.4.8: Optical Component List

Table 14: Optical Components List		
Component	**Manufacturer**	**Part Number**
Quarter Waveplate	Thor Labs	WPQ20ME-1650
Linear Polarizer	Thor Labs	WP50L-UB1
Focusing Optic	Thor Labs	LA4745-ML
Photodiode	Digikey	MTPD2601T-100
Beam Splitter	Thor Labs	UFBS50502
ND Filter	Thor Labs	NENIR20A
Power Meter	Thor Labs	S148C

2.5 System Specifications

Table 15: System Specifications		
Input Beam Specifications		
Parameter	**Value**	**Unit**
Wavelength	At least 2	µm
Max Power	1 - 10	W
Minimum Power	1 – 50	mW
Beam Diameter	4 – 35	mm
Measurement Specifications		
Measurement	**Accuracy**	**Unit**
Normalized Stokes Parameters	±0.05 for all parameters	N/A
Wavelength	±10	nm
M^2 factor	±0.1 up to M^2 =3	N/A
Beam Divergence	±10	µrad
Total Power	< 5	%

Table 16: Component Specifications			
Optical Engineering Components			
Component	**Parameter**	**Specification**	**Units**
QWP	Phase Delay	~π/2 @ 2 µm	waves
	Clear Aperture	> 42	mm
Linear Polarizer	Extinction Ratio	< -20	dB
	Clear Aperture	> 42	mm

Power Meter	Aperture Size	4 – 10	mm
	Power Intake*	1 – 1000	mW
ND Filter	OD Level	2	N/A
Lens	Spectral Range	At least 2	µm
	EFFL	600 - 1000	mm
	Clear Aperture	> 42	mm
Beam Splitter	Split Ratio	50/50	N/A
	Polarization effects	Low	N/A
Electrical Engineering Components			
Photodiode	Responsivity	>0.5 @ 2 µm	A/W
Scanning Slits	Scan Rate	10 - 20	Hz
Translation Stage	Step Resolution	< 50	µm
QWP Rotation Stage	Angle Resolution	< 2	deg
Computer Engineering Components			
MCU	ADC Sampling rate	>10	ksps

*Power level of Power meter can be less than system input power when ND filter is applied

2.6: Hardware Design

2.6.1: Hardware Block Diagram

Below is a general diagram of the flow of the system's hardware. The system's hardware involves not only aligning optical components, but more importantly integrating them all to work precisely and harmoniously. While the block diagram shows a general work distribution, many of the components required communication across the team to make sure the designs would fit the project. For example, when choosing a focal length, it was necessary to know the resolution that could be achieved by the stepper motor used to translate the power meter across the path of the focused beam. Since ISO 11146 Standard requires at least five points within the Rayleigh Range to obtain an accurate measurement, this means that the smallest step size of the chosen stepper motor must be at least five times smaller than the Rayleigh Range of the focused beam.

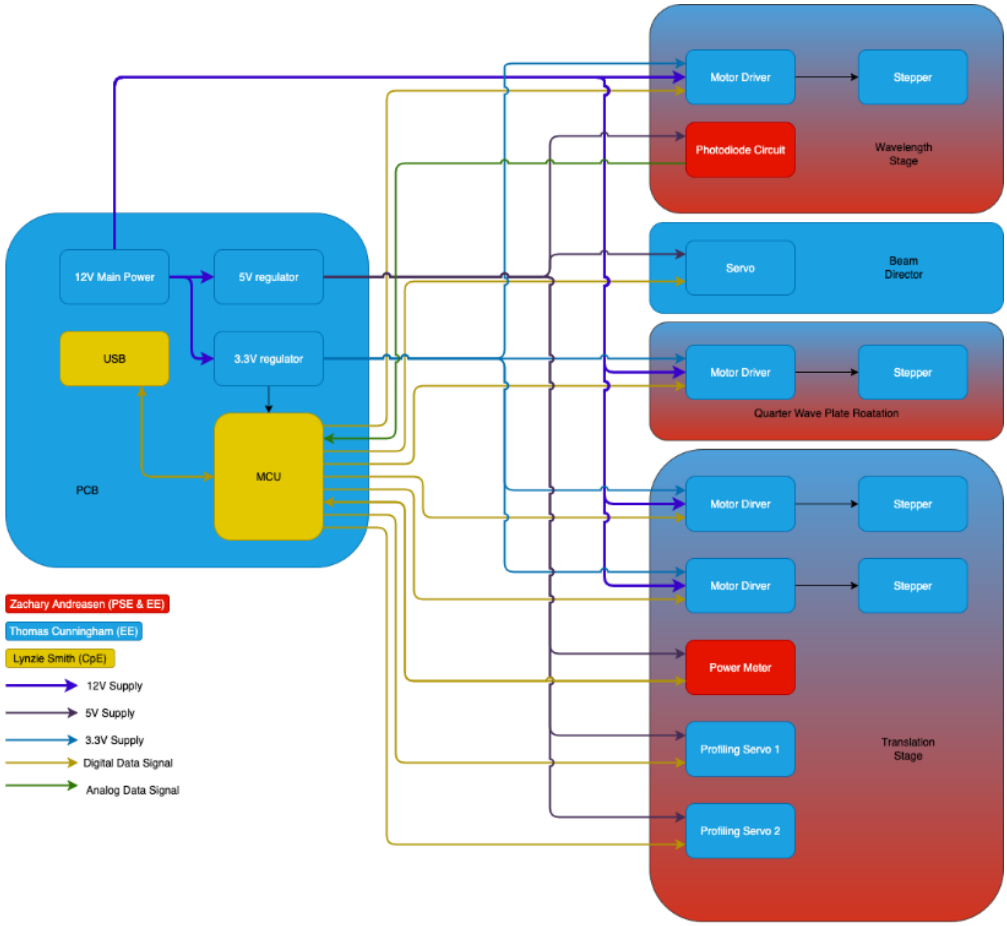


Figure 3: Laser Beam Characterizer Hardware Block Diagram

2.6.2: Motors

We are using a stepper motor for the translational stage combined with a lead screw for its dimensional accuracy. Stepper motors provide precise angle control, and when combined with a lead screw provide accurate dimensional control as well. A common application of this setup can be found on CNC machines.

We are also using a stepper motor system to rotate the QWP for angular accuracy. We need to be able to precisely rotate the QWP to sync with the power measurements.

On the power meter we will use a Servo motor with position control to operate the scanning slits. We need accurate position control to perform the optical calculations. A Servo motor is a good choice because it is very simple and effective.

2.6.3 Microcontroller

We're choosing an ESP32as the main MCU because it gives us the best mix of precise motion control for the motor systems and stable data capture. Its advanced timers make stepper moves (QWP, polarizer, slit) smooth and repeatable which is important for data collection and measurements, and its SPI + DMA handles a 12-bit ADC without jitter while

we sample power and profiles of the laser beam. It also gives us USB/Ethernet for a quiet, reliable link to the PC/UI.

We had first thought of using an STM32 MCU, but we had concerns about getting it to work on the final PCB based on previous senior design groups. After realizing that the ESP32 had enough processing capabilities and it was just as suitable for motor control we ultimately decided to go with the ESP32 instead of STM32.

2.7: System Software Design

On the MCU side of the software process, it will be running a simple loop: move, settle, sample, send. It uses the hardware timers built into the microcontroller to make the motion repeatable, reads the 12-bit ADC, averages a handful of samples, and timestamps the result. For polarization and M^2,it does light math computations on the fly and returns raw data point or computed numbers.

For the PC/UI side, we will need to be able to pick what measurement (polarization, M^2, wavelength) and set the parameters for those measurements (angles, z-positions, samples per point, wavelength). The PC/UI will send the plan to the MCU, which runs its loop, then shows live plots as the data arrives – intensity vs angle with a fitted curve and a polarization ellipse. It should display clean numerical values, logs all setting and results, and can export a CSV/PDF report at the end. Below is a block diagram that shows how this software process will work.

Figure 4: Laser Beam Characterizer Software Block Diagram

2.8 House of Quality

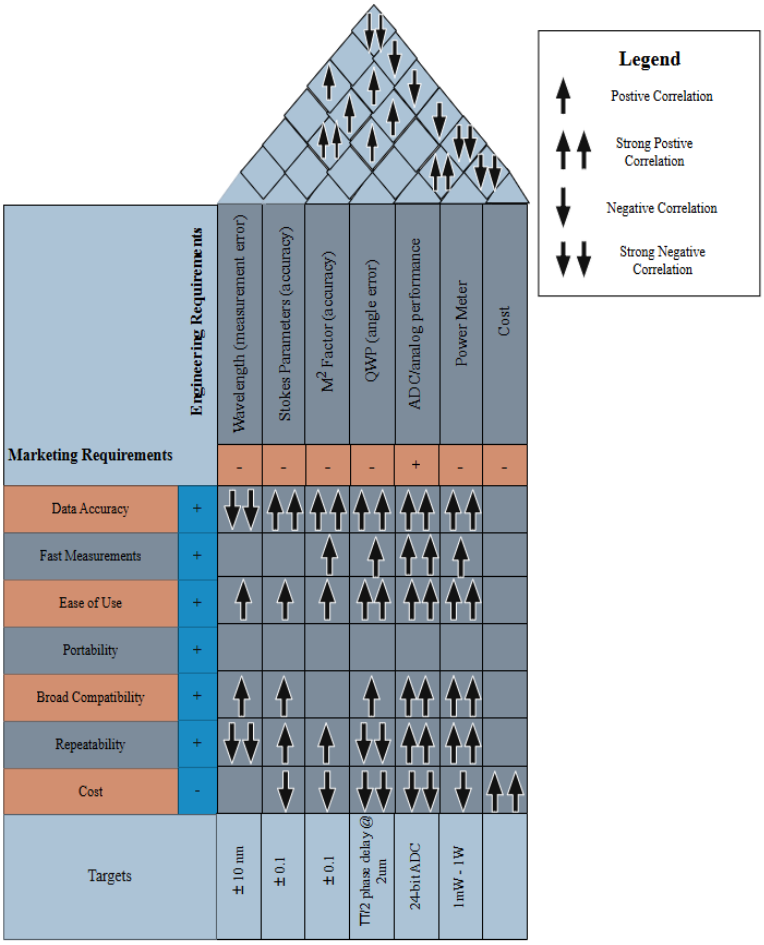


Figure 5: Laser Beam Characterizer House of Quality

Chapter 3: Research and Investigation

3.1: Polarimeter

For the polarimeter design of the project, there are two methods that were researched, known as the Classical Method and the Rotating Quarter Waveplate Method, each explained in detail below. Both methods are based on determining the Stokes Parameters, an ordered set of four values that completely characterize the polarization state of light. They are as follows:

$$S = \begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{pmatrix} = \begin{pmatrix} E_{0x}^2 + E_{0y}^2 \\ E_{0x}^2 - E_{0y}^2 \\ 2E_{0x}E_{0y}\cos\delta \\ 2E_{0x}E_{0y}\sin\delta \end{pmatrix}$$

As described in an article published in the American Journal of Physics, "the parameter S0 describes the total intensity of the optical field, S1 describes the preponderance of linearly horizontally polarized light (LHP) over linearly vertically polarized light (LVP), S2 describes the preponderance of linear +45° polarized light (L +45P) over linear −45° polarized light (L−45P), and the fourth parameter S3 describes the preponderance of right circularly polarized light (RCP) over left circularly polarized light (LCP)." [15] Since these parameters represent the intensity of the light, they can easily be measured by standard power meters. The following two sections describe methods for how these parameters can be measured. Both make use of exactly three components: a Quarter Waveplate (QWP), a Linear Polarizer (LP), and a Power Meter (PM).

3.1.1: Classical Method

In the Classical Method, the QWP is fixed with its fast axis aligned to the horizontal axis of the system and the LP is allowed to rotate. In this configuration the intensity on the power meter can be described by the following equation:

$$I(\theta,\varphi) = \frac{1}{2}(S_0 + S_1\cos 2\theta + S_2\sin 2\theta\cos\varphi - S_3\sin 2\theta\sin\varphi)$$

where I(θ,φ) is the intensity measurement at LP angle θ and waveplate retardance of φ [15], [16]. Four power measurements are taken, the first three with just the LP in the path of the beam and the last one with both the QWP and the LP. The first three measurements are with the LP at 0°, 45°, and 90°. The final measurement is taken with the LP at 45° and the QWP inserted into the path of the beam before the LP. The Stokes Parameters can then be calculated from the following equations.

$$S_0 = I(0°,0°) + I(90°,0°)$$

$$S_1 = I(0°,0°) - I(90°,0°)$$

$$S_2 = 2I(45°,0°) - S_0$$

$$S_3 = S_0 - 2I(45°,90°)$$

There are a few downsides to this method. First, the QWP will absorb some optical power, which will change how the equations depending on how much is absorbed. Additionally, the QWP will have to be realigned any time a new measurement needs to be taken. For these reasons, it is beneficial to make use of the second method, the Rotating Quarter Waveplate Method.

3.1.2: Rotating QWP Method

In the Rotating Quarter Waveplate Method, an LP is fixed in the horizontal axis and a QWP is allowed to rotate in the path of the beam before the LP. According to Mueller Calculus, the QWP and LP will change the Stokes Parameters by the following transfer matrices[17]:

$$QWP(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos^2(2\alpha) & \cos(2\alpha)\sin(2\alpha) & \sin(2\alpha) \\ 0 & \cos(2\alpha)\sin(2\alpha) & sin^2(2\alpha) & -\cos(2\alpha) \\ 0 & -\sin(2\alpha) & \cos(2\alpha) & 0 \end{bmatrix}$$

$$LP(\beta) = \frac{1}{2}\begin{bmatrix} 1 & \cos(2\beta) & \sin(2\beta) & 0 \\ \cos(2\beta) & cos^2(2\beta) & \cos(2\beta)\sin(2\beta) & 0 \\ \sin(2\beta) & \cos(2\beta)\sin(2\beta) & sin^2(2\beta) & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

where α and β are the angles of the QWP fast axis and the transmission axis of the LP with respect to the horizontal axis, respectively. Thus, assuming the LP is fixed at 0 degrees and the QWP fast axis is at some arbitrary angle θ, the output intensity corresponding to the S_0 parameter at the output of the system can be written as a linear combination of the input Stokes Parameters shown in the equation below:

$$I(\theta) = \frac{1}{2}(S_0 + S_1 cos^2(2\theta) + S_2 \cos(2\theta)\sin(2\theta) + S_3 \sin(2\theta)$$

where S_0, S_1, S_2, and S_3 are the Stokes Parameters of the input laser beam. Thus, by measuring the power transmitted through the LP at different QWP angles, it is possible to completely characterize the polarization state of the laser beam. This equation can be rewritten using common trigonometric identities to the following: [15], [16]

$$I(\theta) = \frac{1}{2}(A + B\sin 2\theta + C\cos 4\theta + D\sin 4\theta)$$

where

$$A = S_0 + \frac{S_1}{2} \qquad B = S_3 \qquad C = \frac{S_1}{2} \qquad D = \frac{S_2}{2}$$

Using Fourier Analysis, the coefficients A, B, C, and D can be calculated in the following way:

$$A = \frac{2}{N}\sum_{n=1}^{N} I_n \qquad B = \frac{4}{N}\sum_{n=1}^{N} I_n \sin 2\theta_n$$

$$C = \frac{4}{N}\sum_{n=1}^{N} I_n \cos 4\theta_n \qquad D = \frac{4}{N}\sum_{n=1}^{N} I_n \sin 4\theta_n$$

Using the equations above, the Stokes parameters can be calculated from these values as: [15]

$$S_0 = A - C \qquad S_1 = 2C \qquad S_2 = 2D \qquad S_3 = B$$

3.1.3: Polarimeter Method Comparison

When comparing the two methods, the Rotating QWP method clearly shows the most promise. Not only are there the advantages explained in the previous sections, but also there is the advantage of automation. Since this project aims to automate this measurement, it is much simpler to only have to control one parameter, namely the rotation of the QWP. In the Classical Method, automation would require not only rotating the LP but also moving the QWP in and out of the beam path. This makes the Classical Method design much more complex. Additionally, there is the issue of resolution and measurement error. In the Classical Method, there is a set number of positions that are taken, and the accuracy of the measurement is restricted to the specifications of the optical components and the accuracy of the angles used in the LP and the QWP. In the Rotating QWP method, however, accuracy

can easily be improved by increasing the number of points taken in a single measurement. Though the minimum required number of power measurements is eight points as discussed above, there is always the possibility of taking more than this to improve resolution and decrease the impact of measurement error. These reasons and others are tabulated below.

Table 17: Polarimeter Method Comparison		
Parameter	**Classical**	**Rotating QWP**
Automation Ease	Difficult	Simple
Math Complexity	Simple	Complex
Uncertainty from QWP power absorption	Present	Absent
Number of measurements required per collect	4	8
Method Chosen	No	Yes

3.1.4: Polarization Plotting

With these parameters calculated, the polarization ellipse and Poincare sphere can be easily plotted. First, for the polarization ellipse, it is necessary to know the angle of the major axis with respect to the horizontal as well as the angle formed by the major and minor axes of the ellipse, as shown in the figure below. These angles are referred to as ψ and χ, respectively. They can be calculated using the below equations, and an image of the shape is shown below.[15], [16], [18]

$$\psi = \frac{1}{2}\tan^{-1}\frac{S_2}{S_1} \qquad \chi = \sin^{-1}\frac{S_3}{S_0}$$

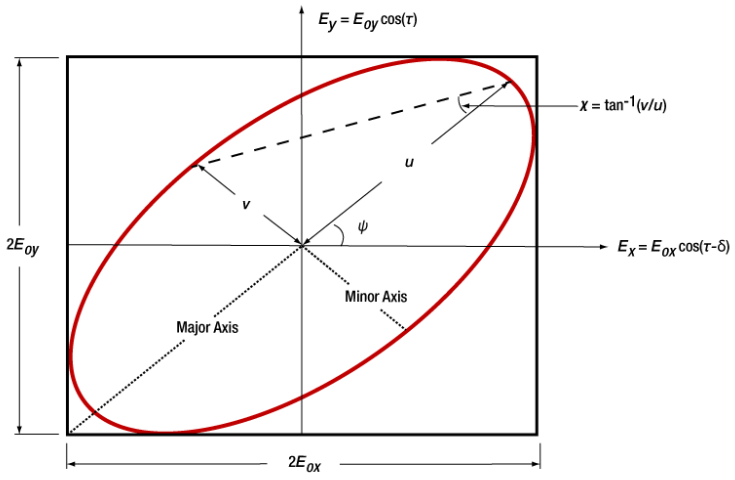


Figure 6: Polarization Ellipse

When it comes to the Poincare sphere, the plotting is relatively easy. The plot consists simply of the parameters S_1, S_2, and S_3 in spherical coordinates, whose angles are related to the angles of the polarization ellipse as shown in the figure below: [19]

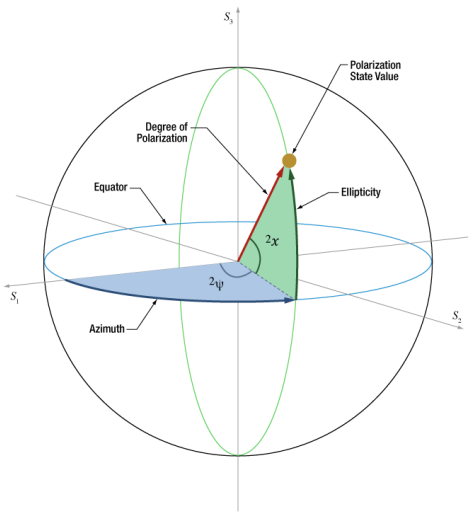


Figure 7: Poincare Sphere

3.2: M^2 Factor Measurement

When it comes to the M^2 factor, it is necessary to have some understanding of Gaussian beam propagation. When a laser propagates in space, its intensity distribution follows a Gaussian curve. Due to diffraction, this distribution does not maintain a constant width as the beam propagates in space but rather expands. Thus, the beam can never be truly collimated but rather has some divergence angle. Additionally, the beam cannot be focused down to an infinitely small spot, but rather, focuses to a minimum spot size, known as the beam waist. The diagram below gives a visual representation of this propagation. [20]

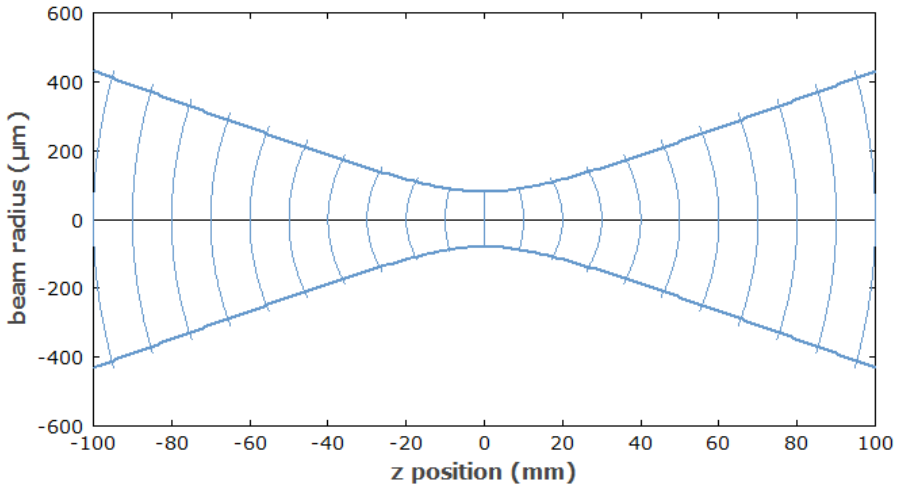


Figure 8: Gaussian Beam Propagation Diagram; note how as the beam converges, it does not focus down infinitely small, but has a minimum spot size until it diverges again at some angle

The beam waist and divergence angle are not arbitrary but rather are related to each other by the following equation.

$$\theta = M^2 \frac{\lambda}{\pi w_0}$$

where,

- θ is the divergence angle of the beam
- λ is the wavelength of the beam
- w_0 is the beam waist
- M^2 is the beam quality factor

In an ideal beam, the M^2 is equal to 1, and the equation simplifies. However, in the presence of wavefront distortions, this factor can get larger causing the beam to diverge faster. To measure this factor, one must measure the divergence of the beam and its smallest spot size and then calculate the M^2 factor from this. There is a standard for this measurement that will be discussed in Chapter 4.

3.2.1: Image Sensor

To make this measurement, it is necessary to obtain an accurate profile of the beam. As discussed in Chapter 2, there are two main ways to do this. The first is to use an image sensor with a certain pixel pitch and count. This will typically consist of a CMOS or CCD sensor that is implemented into a circuit where a digital image processor can read the image and extract the approximate 1/e^2 beam width. One of the advantages of this method of measuring the beam profile is that it allows for the beam width to be measured in any direction. Often laser beams are not entirely symmetrical but rather have an elliptical shape. Thus, if an image sensor is used to measure the beam profile, there is no limitation to which axis can be measured. Additionally, if the beam intensity distribution is not Gaussian, due to aberrations or the presence of higher order transverse modes, the image sensor will have the ability to accurately calculate the D4sigma value (discussed in Chapter 4). While there are these advantages, one of their disadvantages is that many image sensors of this type are very sensitive to optical power and therefore can only be used if the optical power density of the beam is very low. Additionally, their resolution is limited to the pixel size of the sensor, often tens of microns in pitch. This can become problematic if the beam gets small in comparison to this size.

3.2.2: Knife Edge Test

The second method is the knife edge test. In this method a single photodiode is used to detect the amount of optical power in the beam, and a knife edge blade is translated across the path of the beam, obtaining a power curve whose derivative corresponds to the intensity distribution profile. Depending on how fast the power readings can be made, this can be done very fast by attaching the blade to a motor and repeatedly scanning across the path of the beam. One of the main downsides of this method is that only one axis can be scanned at a time. So, if the laser is elliptical, only one axis could be scanned at a time. This means that it is much harder to get a complete beam profile. However, due to the control of the scan rate of the blade, the resolution of the measurement can be very high giving it a distinct advantage over other methods.

3.2.3: Profilometer Comparison

When comparing the two methods, each has its strengths and weaknesses. First, assuming the beam is in the fundamental mode, the knife edge scanning method can have much better resolution, often on the order of a single micron, whereas the image sensor is limited by the pixel size of the sensor. Additionally, the scanning slit profilometer can often accept much higher power levels and have much broader wavelength ranges than a CCD or CMOS sensor. On the other hand, when the beam is not propagating in the fundamental mode, there is often an uneven spatial distribution in the power of the laser beam, which would be much easier to see on an image sensor. This is because the scanning slit can only take a profile in one or two directions, limiting the device's ability to profile asymmetric beams. In this project, the laser will be assumed to be propagating close to the fundamental mode, where the scanning slit method will be implemented instead of an image sensor. This will be done using a simple power meter and two fan blades rotated in the path of the aperture to take the profile in the X and Y dimensions. This will offer a much cheaper option than most scanning slit profilometers on the market today while maintaining good resolution for accurate beam profiling measurements. The advantages and disadvantages of each of these are shown in the table below:

Table 18: Profilometer Method Comparison

Parameter	Image Sensor	Knife Edge Test
Max laser power	Low	High
Resolution	Moderate/Low	High
Full Beam Profile	Yes	No
Cost	Moderate/High	Moderate
Chosen	No	Yes

3.3: Wavelength Measurement

When it comes to measuring the wavelength of light, there were three different methods that were considered. These are Spectrometry, Interferometry, and Beat Frequency Analysis. Of these, the second was chosen as the best means for measuring the wavelength of the laser beam input into the system as it best compromises resolution, bandwidth, and cost. These are each described in detail below.

3.3.1: Spectrometry

When considering how to measure the wavelength of the input laser beam, the first method that was considered was a spectrometer. This would consist of using a diffraction grating to split the input beam, and detecting the beam on a photodetector array, such that the wavelength of the beam could be inferred from the position of the beam on the detector. As shown in the equation below, assuming normal incidence, the angle of each of the beams that transmits through (or reflects if using a reflective grating) the grating can be expressed as a function of wavelength and grating spacing.

$$\sin\theta_m = \frac{m\lambda}{d}$$

Since the grating spacing (d) would be a known constant, and the angle of the beam (θ_m) could be inferred from the position of the beam on the photodetector array, the wavelength (λ) for the first order mode (m=1) can be easily calculated. This is how many spectrometers work and is well proven as an effective means of measuring the spectrum of most sources.

There are several advantages to using a spectrometer to measure the wavelength of the laser beam. One of the main ones is that the linewidth of the laser can be characterized, since the beam will have a spread across the photodetector array, from which the spread of the wavelengths can be calculated in the same way as stated above. This can be very useful in many applications. Another advantage of this method is that there is a zero-order mode that will also be transmitted through the grating that can be used for some of the other subsystems of the design, such as the polarimeter or the M^2 factor measurement. This makes it easier to implement into the system and makes the system less complex. Below is an initial drawing of the proposed layout.

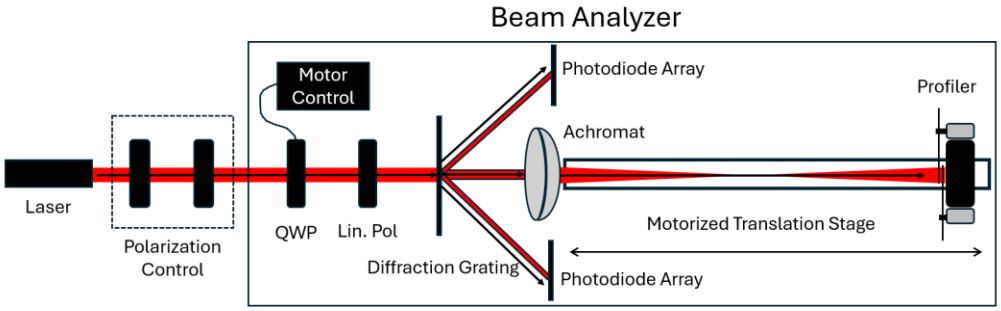


Figure 9: Initial design with spectrometer implementation integrated into the system with the polarimeter and beam quality measurement subsystems; the photodetector could be a photodiode array (as shown in the figure) or a CMOS or CCD sensor

Despite these advantages, there are several disadvantages to this design. The first one is that there is some uncertainty about whether the grating will affect the M^2 factor. In theory, the zero-order beam will simply follow the laws of refraction, thus not changing the M^2 factor. However, if there are imperfections in the grating, there is a chance the beam might induce small aberrations to the beam and reduce beam quality. This would cause issues when measuring the M^2 factor. The second major disadvantage is the compromise between the resolution of the measurement and the range of wavelengths able to be measured. This happens because when the array is close to the grating, there is a wide range of angles that will hit the array, but the resolution is limited by the spacing of the array. On the other hand, if the array were placed far away from the grating, finer angle resolution can be achieved, but the range of angles is greatly reduced. This is an inherent compromise that is difficult to get around. Finally, the last disadvantage is the cost of the detectors. Since this project aims to take inputs in the NIR range, the detector array must work at that range, which is often very expensive and not feasible for this project. For these reasons, other methods were explored.

3.3.2: Interferometry

The second method that was considered to measure the wavelength of the laser beam was an interferometer. There are many different types of interferometers, but what they all have in common is constructive or deconstructive interference of light. This occurs when two beams of light are coincident in the same position, yet their phases are not necessarily matched. Where the phases are matched, the intensity of the light will increase, but where they are out of phase, they will de-constructively interfere with each other until they are completely out of phase with each other where there will be no light. There are two different types of interferometers that were considered to achieve this goal. The first is a Fabry Perot Interferometer and the second is a Michelson Interferometer.

3.3.2.1: Fabry Perot Interferometer

The first interferometer design considered was a Fabry Perot Interferometer. This consists of two highly reflective mirrors that are positioned such that their reflective facets face each other. This forms a resonant cavity, where only certain cavity modes will resonate. These cavity modes correspond to different frequencies of light. This is because as the light reflects between the two mirrors certain frequencies will constructively interfere with each other, whereas others will de-constructively interfere with each other. This is a function of the spacing between the mirrors as seen in the equation and the figure below:[21]

$$\Delta\nu_{FSR} = \frac{c}{2l} \qquad \Delta\lambda_{FSR} = -\frac{\lambda^2}{c}\Delta\nu_{FSR} = -\frac{\lambda^2}{2l}$$

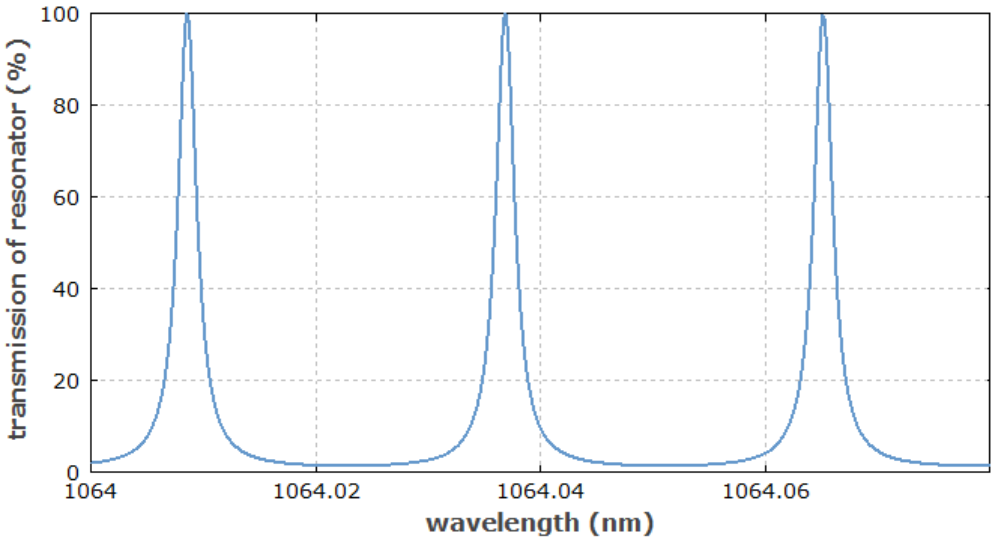


Figure 10: Transmission vs. Wavelength plot for a Fabry Perot Cavity with mirror reflectivity's of 90%

Given the above equations, the laser beam will only resonate in the cavity at certain cavity lengths. If the mirrors were to be translated to find two resonant cavity lengths, the following equations can be used to derive the formula used to calculate the laser beam wavelength.

$$2nd = m\lambda \qquad 2nd' = (m+1)\lambda$$

$$2n(d' - d) = \lambda$$

$$\lambda = 2n\Delta d$$

Where,

- λ is the wavelength of light
- n is the refractive index of the material in the cavity (usually equal to 1)
- m is the mode number
- d and d' are the cavity lengths corresponding to mode m and (m+1) respectively

There are many advantages to using a Fabry Perot interferometer. One of the main ones is the accuracy of the measurement that can be obtained. When the mirrors are highly reflective, it allows for a very narrow window of transmission as the cavity length is changed. This means that the wavelength can be very precisely calculated. Additionally, it is a very small and compact system, which means it could easily be fit into the system without taking up a lot of space. Finally, the bandwidth of the device is very broad, since it would only really be limited by the bandwidth of the photodetector used to measure the transmission out of the cavity.

Despite these advantages, there are some disadvantages. The first is that the device can be very difficult to align. In most cases, the cavity mirrors are not flat mirrors, as any small misalignment would cause the light to escape the cavity instead of resonating with it. Instead, curved mirrors are used to contain the light in the cavity. These can be much more difficult to get aligned properly. Additionally, the cavity mirrors will need to be moved very small amounts and to very precise locations if the true displacement of the cavity is to be determined and the most accurate wavelength measurement obtained. This can be done with very precise linear actuators or piezoelectric devices, but these generally will greatly increase the cost and/or complexity of the device. Thus, the Michelson interferometer was considered as a viable alternative.

3.3.2.2: Michelson Interferometer

The second interferometer that was considered was a Michelson Interferometer. This interferometer consists of a beam splitter and two retro mirrors on the two legs of the split beam. These mirrors reflect the beams back to the splitter allowing them to recombine and interfere with each other. This interference pattern is caused by the relative phase difference between the two arms of the interferometer created by the beam splitter. Thus, by keeping one mirror fixed and translating the other mirror, the relative phase difference between the two arms can be changed. By analyzing how much the interference pattern changes for a certain distance of mirror displacement, the wavelength of the laser can be calculated using the following equation.

$$\lambda = \frac{2\Delta d}{N}$$

Where,

- λ is the wavelength of the beam
- Δd is the distance traversed by the mirror
- N is the number of peaks passed for the specified mirror displacement

There are several advantages to the Michelson interferometer. The first is that it can be very accurate compared to other methods of measuring wavelength. This is because the small changes in mirror displacement can drastically affect the interference pattern, which can easily be read by a photodetector. Another advantage is that, like the Fabry Perot, it can often have a broad spectral range. Additionally, some advantages it has over the Fabry Perot are that it is both less complex, easier to implement and less costly.

One of the disadvantages of the Michelson interferometer is that it is much bigger than the Fabry Perot interferometer and will take up a lot more space in the system. Additionally, it is much more difficult to design the interferometer to function at the higher wavelengths this project seeks to function at. This is because the spectral range not just of the photodiode, as is the case also with the Fabry Perot interferometer, but also of the beam splitter must match the necessary range of input wavelengths. Additionally, as it is an interferometer, the alignment process can be rather challenging. These challenges will need to be overcome if the measurement will be successful.

3.3.3: Beat Frequency Analysis

Another method that was considered to measure the wavelength of the laser beam was to mix the input beam with another laser, called a Local Oscillator, and measure the beat frequency between the two beams. Since the wavelength of the local oscillator would be known, the wavelength of the input beam can be measured knowing the difference between the two beams using the following equation.

$$\Delta\nu = \frac{c}{\lambda^2}\Delta\lambda$$

where,

- $\Delta\nu$ is the beat frequency measured by the photodiode
- c is the speed of light
- λ is the wavelength of the local oscillator
- $\Delta\lambda$ is the wavelength difference between the input laser beam and the local oscillator

This method is useful because there are no moving parts to the system, since the beat frequency can be measured directly by a photodiode, so there are no challenges with the small incremental movements present in the interferometer designs. Additionally, it provides very accurate measurements, since even a small beat frequency between the two beams can be measured by the photodiode. There are two major challenges to this method though. First is the cost. Since the measurement depends on the wavelength of the local oscillator, this means that this laser must be very stable with no deviations in wavelength. These lasers are generally very expensive and outside the scope of this project. Additionally, while the method allows for very precise measurement, it does not allow for a very broad band of wavelengths to be measured, as the beat frequency quickly gets much too high to be read by any digital electronics. To give an example of this, suppose a 1-micron wavelength local oscillator had a wavelength separation of 1 nm from the input laser beam. Calculating the expected beat frequency yields a result of 3 THz, a frequency

much too high to be measured even on advanced oscilloscopes. Thus, this method is not very feasible for this project.

3.3.4: Wavelength Measurement Method Comparison

When comparing the three methods proposed above, each one has its advantages and disadvantages. One of the main tradeoffs that occurs in these is between wavelength resolution and bandwidth. This is especially true with the spectrometer and with the coherent wave mixing methods. Therefore, to achieve the best solution that does not overly compromise the resolution and bandwidth the system, the Michelson interferometer will be chosen. In addition to this, it has the benefit of being much cheaper than the other two methods, which require either a photodiode array or a very coherently stable laser. The full comparison of the advantages and disadvantages of these methods is tabulated below.

Table 19: Wavelength Measurement Comparison

Parameter	Spectrometer	Interferometer	Beat Frequency
Resolution	Low	Moderate/High	Very High
Spectral Range	Broad	Broad	Very Low
Complexity	Low	Moderate/High	High
Cost	High	Low	Very High
Chosen	No	Yes	No

3.4: Divergence Measurement

To measure the far field divergence of the beam, it is usually impossible to profile the beam at different points far from the waist to observe the linear expansion of the beam. This is because the Rayleigh range is usually very long in a laser beam, which means that to get an accurate measurement, one must spread the points out very far to have enough spatial resolution to see the expansion. Therefore, to integrate this measurement into the system, it will be necessary to use an alternative method. Following the well-known standard ray tracing equations allows one to follow the path of a diverging ray as it gets deviated by a focusing optic. The standard paraxial ABCD matrices which map the ray heights and divergences through the system will suffice for this derivation. They are as follows.

$$M_{Lens} = \begin{bmatrix} 1 & 0 \\ -K & 1 \end{bmatrix} \quad M_{Propagation} = \begin{bmatrix} 1 & d \\ 0 & 1 \end{bmatrix}$$

where,

- K is the power of the lens
- d is the distance the ray travels

By cascading these matrices any column vector containing a ray's height and divergence angle can be mapped to its resulting column vector after any series of optical components spaced apart by arbitrary distances. A diverging ray at some arbitrary height can therefore be traced to its position at one focal length away from the lens in the following manner.

$$\begin{bmatrix} h' \\ u' \end{bmatrix} = \begin{bmatrix} 1 & f \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{1}{f} & 1 \end{bmatrix} \begin{bmatrix} h \\ u \end{bmatrix}$$

Where,

- h, h' are the heights of the object and image space rays respectively
- u, u' are the divergence angles of the object and image space rays respectively
- f is the focal length of the surface
- n is the refractive index of the medium

Multiplying these matrices together yields two equations. The first of these equations reduces to an equation that directly related the input rays divergence angle to the height of the ray at one focal length away from the lens. It follows then that if the beam profile is measured at the focal point of the lens, that the divergence of the beam can be calculated from the size of the beam. The image below gives a more graphical representation of this method. [22]

$$\begin{bmatrix} h' \\ u' \end{bmatrix} = \begin{bmatrix} h - f(u - \frac{h}{f}) \\ u - \frac{h}{f} \end{bmatrix} = \begin{bmatrix} fu \\ u - \frac{h}{f} \end{bmatrix}$$

$$h' = fu$$

$$u = \frac{h'}{f}$$

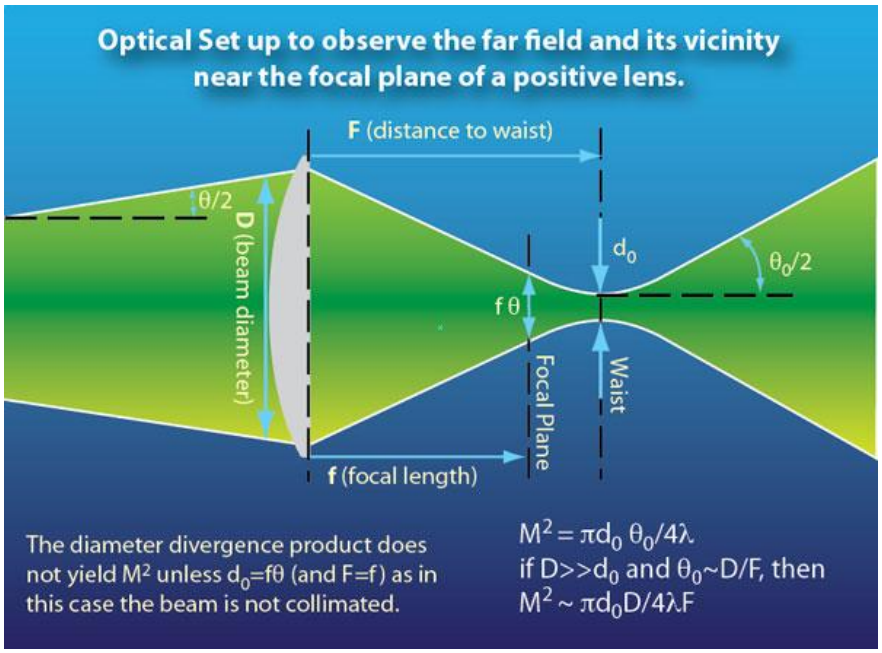


Figure 11: Visual diagram of divergence calculation method; the beam width at the focal plane of the lens can be used to calculate the divergence of the beam before the lens given the known focal length of the beam

3.5: Photodiode Circuit

3.5.1: Photodiode

For the Michelson Interferometer, it is necessary to have a sensor that can detect the presence of light in a certain area. The obvious device to use for this is a photodiode. As with all diodes, photodiodes are constructed from oppositely doped semiconductors merged in a PN junction. In some of these devices, an intrinsic region is added in between the two regions in the junction to increase the width of the depletion region, creating what is called a PIN diode. This is done so that light can easily access the region, exciting electron-hole pairs that can be manipulated electronically. There are two main configurations to do this. These are called photoconductive mode and photovoltaic mode, which are both discussed below.

3.5.1.1: Photoconductive Mode

In photoconductive mode, the photodiode is reverse biased above the breakdown voltage. This allows for an external electric field to be applied that will increase the speed of the charge carriers generated by the incident light, thereby increasing the current in the device. This is helpful since the amplifier will easily be able to see this current and amplify it. The downside of this though is that there is generally a much higher dark current, the current present when none of the light in question is incident on the device. As seen in the equation below, the current through the diode can be described by the exponential Shockley Diode equation.

$$I_D = I_S(e^{\frac{V_D}{nV_T}} - 1)$$

where,

- I_D is the current through the diode
- I_S is the reverse bias saturation current
- V_D is the voltage across the diode
- V_T is the thermal voltage (25.85 mV at room temperature)
- n is the ideality factor

In the presence of light the equation becomes,

$$I_D = I_S\left(e^{\frac{V_D}{nV_T}} - 1\right) - I_L$$

where I_L is the current generated by the light incident on it. This current is linearly proportional to the amount of optical power on the diode by the factor known as the responsivity. This value is wavelength dependent but directly relates the amount of current the diode will generate due to the light on it. In photoconductive mode, the exponential term approaches zero for even a small amount of reverse bias. The equation then simplifies to the following.

$$I_D = -I_0 - KP_i$$

where

- I_D is the current in the diode
- I_0 is the dark current
- K is the responsivity of the diode
- P_i is the incident optical power on the device

In this equation, it becomes evident that the current through the diode is linearly proportional to the amount of optical power incident on it. This is very beneficial since this current can easily be transferred to a voltage by a resistor in the feedback loop of a Transimpedance Amplifier (TIA), which can transfer the current signal to a voltage signal significant enough for a microcontroller to detect. In this way the photodiode can transfer the optical signal to a voltage signal which can be processed to extract the wavelength information.

3.5.1.2: Photovoltaic Mode

In photovoltaic mode, the photodiode is left unbiased. Thus, when light is incident on it and generates electron-hole pairs, a small voltage is created across the device. Since this voltage is positive, a large current will be created following the Shockley Diode equation shown in the previous section. This will generally be much larger than the photocurrent term, so to detect the effects of the incident light, a high impedance load must be applied to the device, so that the current is limited. Under this configuration, the current through the diode is approximately zero, so the voltage across the diode can be solved for and related to the optical power on the device by the following equation.

$$V_j = nV_T \ln\left(\frac{KP_i}{I_0} + 1\right) \approx nV_T \ln\left(\frac{KP_i}{I_0}\right)$$

The approximation made here is because the photocurrent (KP_i) is generally much higher than the dark current (I_0). Thus, it is shown that the voltage across the photodiode in this configuration is logarithmically related to the optical power incident on the diode. This voltage signal can then be amplified if needed to a level for the intended application.

3.5.1.3: Photodiode Mode Comparison

There are many advantages and disadvantages to each of the two modes discussed above. When it comes to the form of the response to optical power, linearity is generally preferred, and so often photoconductive mode is the operation of choice over photovoltaic mode. On the other hand, the lack of external bias in photovoltaic mode means that the dark current is generally very low, and the device is much more sensitive to changes in optical power. This also means the device is less power consumptive in photovoltaic mode, since there is not an external voltage used. On the other hand, though, the reverse bias present in photoconductive mode increases the depletion region in the diode which decreases the junction capacitance. This allows the device to have a much broader bandwidth than when in photovoltaic mode. Since this application requires accurate reading of the quickly changing interference signal due to the motion of the movable mirror in the Michelson Interferometer, it was chosen to operate the device in photoconductive mode. The comparison of these two modes is tabulated below.

Table 20: Photodiode Mode Comparison		
Parameter	**Photoconductive**	**Photovoltaic**
Response shape	Linear	Logarithmic
Dark Current	High	Low
Power Use	High	Low
Response Time	Fast	Slow
Chosen	Yes	No

3.5.2: Transimpedance Amplifier

Since the photodiode will be in photoconductive mode, it is necessary to convert the current signal into a significant voltage value to be read by the microcontroller. The obvious device to accomplish this is a Transimpedance Amplifier (TIA). This device consists of a single op amp with a resistor in the feedback loop connecting the output of the op amp to its inverting input. The inverting input takes in a current, while the non-inverting input is connected to ground. Thus, the output voltage of the op amp will be equal to the product of the input current and the resistance of the feedback resistor, following Ohm's Law. The equation for the transfer function of this circuit is shown below, as well as an image of the general schematic of the device. [23]

$$V_{out} = -I_{in}R_f$$

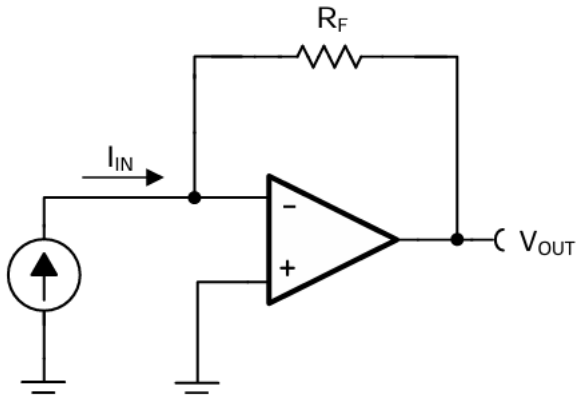


Figure 12: Basic Transimpedance Amplifier Design

To make sure the bandwidth of the amplifier is sufficient for this application, a capacitor is placed in the feedback loop in parallel with the resistor. The transfer function of the amplifier can then be calculated using Kirchoff's Current Law, as shown below.

$$I_{IN} = I_{Cf} + I_{Rf} = -\frac{V_{Out}}{R_f} - sC_f * V_{Out}$$

$$\frac{V_{Out}}{I_{IN}} = \frac{-R_f}{1 + sC_fR_f}$$

This creates a pole frequency, beyond which the gain of the circuit will decrease. The value of this pole frequency can be calculated using the equation below:

$$f_p = \frac{1}{2\pi C_f R_f}$$

Besides this capacitor though, there are several other capacitances that appear in the circuit which can cause problems with the amplification. To begin, there is a junction capacitance in the photodiode that limits the response time. In addition to this though, there are also several parasitic capacitances between the nodes of the op amp that add to the input capacitance. The figure below shows a new circuit diagram with all the new capacitances. [23]

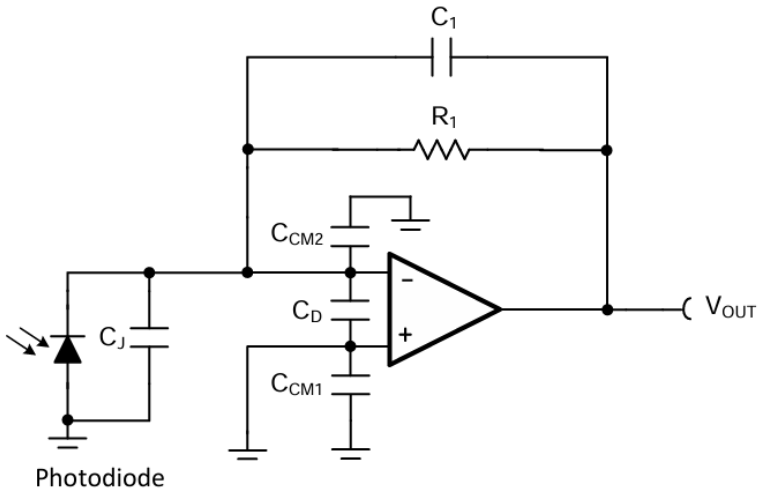


Figure 13: Updated TIA schematic with junction and parasitic capacitances shown

The parasitic capacitances at the inputs of the op amp as well as the junction capacitance can be combined into a single "Input Capacitance" term by the following equation. Most Op Amp datasheets will provide the value of C_D and C_{CM2} in one value C_{in} for the op amp, so they are combined into one term as seen in the equation below. Note that C_{CM1} is connected to ground on both terminals and so it will not contribute to the input capacitance. [23]

$$C_{in} = C_J + C_D + C_{CM2} = C_J + C_{in(Op\ Amp)}$$

where,

- C_{in} is the input circuit capacitance
- C_J is the junction capacitance of the photodiode
- C_D is the parasitic capacitance between the two inputs of the op amp
- C_{CM2} is the parasitic capacitance between the inverting input and ground
- $C_{in(Op\ Amp)}$ is the input capacitance of the Op Amp

To maintain the stability of the amplification, the gain-bandwidth product of the op amp must be sufficiently high. The required product can be calculated from the following equation. [23]

$$f_{GBW} > \frac{C_{in} + C_1}{2\pi R_1 C_1^2}$$

where,

- f_{GBW} is the gain-bandwidth product of the op amp
- C_{in} is the input capacitance (including the photodiode junction capacitance)
- C_1 is the feedback capacitance
- R_1 is the feedback resistance

Additionally, as discussed above, the photodiode will need to be reverse biased to be in photoconductive mode. This can be done by biasing the non-inverting input of the op amp. The op amp will therefore drive the inverting input to be the same voltage, creating a bias on the photodiode. This also allows the negative power supply to be connected to ground, as the minimum output voltage will be equal to this bias voltage. This bias can be achieved by connecting the op amp power supply to a voltage divider to achieve the desired bias. As power supplies can also be a source of noise, a capacitor should be placed between the non-inverting input and ground to filter out this noise. The final schematic is shown below.

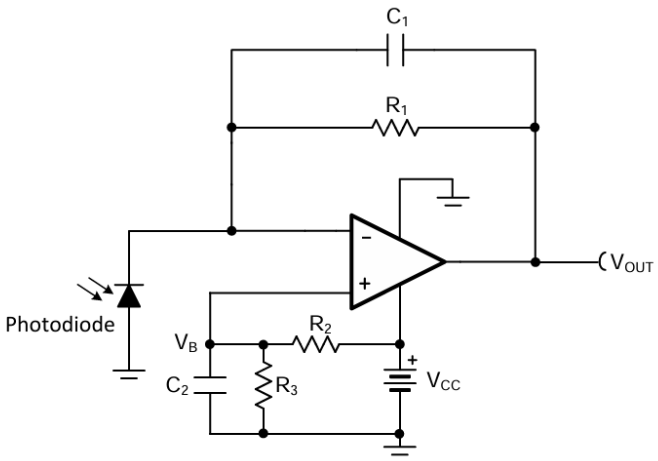


Figure 14: Complete Schematic of Photodiode amplified by Transimpedance Amplifier

3.6: Motors

3.6.1: Motor Technologies

Electromechanical actuators are a crucial element in making a project affect and observe the environment it is in. Being able to precisely and reliably move is the main function of these actuators. There are few notable technologies:

- Rotary motors
  - DC brushed
  - Stepper
  - AC induction
  - Servo

- Piezoelectric Actuators

The next few sections will describe these technologies and their pros and cons.

3.6.1.1: DC Brushed Motors

Brushed DC motors are some of the most common and simplest electric motors. At the core of this technology is an electromagnet with each end connected to a commutator. The power is supplied to the electromagnet by pushing carbon brushes against the commutators. Permanent magnets are attached around the electromagnet, so when the electromagnet is energized, a rotary force causes the motor to spin. The brushes ensure that the electromagnet is always opposing the magnetic field of the permanent magnets.

The brushed DC motor has a few notable pros and cons:

- Pros
    - Size: brushed DC motors can get really small. The size limiting factors are the permanent magnet and the electromagnetic coils.
    - Easy to operate: these motors only require a DC current to pass through to rotate, requiring no complicated hardware to achieve continuous rotary motion.
    - Inexpensive: because they are so simple in design, they can be found at very low cost compared to other motor technologies.
- Cons
    - Accuracy: the design of the motor does not have a way to report position. A rotary encoder can be added to give position feedback
    - Reliability: brushed DC motors are known for failing after some time due to the brushes and commutators being fused together. As the brushes pass across the space between commutator plates, little arcs can form and overtime weld the two together.
    - Performance: the motors do not have as much torque as a similar motor of different technology. This is due to the energy dissipated by the brushes as heat.

3.6.1.2: Stepper Motor

The stepper motor is one of the most commonly used motors for precision and accuracy. In a standard 1.8˚ stepper motor, there is a permanent magnet on the rotary shaft. On the magnet are 50 metal teeth on the north and south poles. There are then 48 teeth on the outside, arranged in 8 groups of 6 teeth. These outer teeth are connected to electromagnets, and the arrangement makes it so that at any moment, two sets of 6 are align with the center, two are unaligned, and the rest are half aligned. This makes it so that when you deenergize the first set of coils and energize the second set, the motor moves 360˚/10 = 1.8˚. Further angle resolution can be achieved by energizing the two coils at different levels, causing the teeth to go proportionally between the two. This is called microstepping, and many modern controllers offer 16x microstepping, which increases the angle resolution to 0.1125˚. Here is a list of the stepper motor's pros and cons:

- Pros

    - Accuracy: stepper motors provide amazing angular position control, making them one of the most common motor types in precision applications.
    - Reliability: steppers only allow small increment of motion at a time, making them very reliable in their position.
- Cons
    - Expensive: due to their precise design, stepper motors are generally more expensive than other motors.
    - Operation hardware: because motion is achieved through alternating energizing coils, more sophisticated hardware is required to operate. Even more so to do microstepping.

3.6.1.3: AC Induction Motor

The AC induction motor is the most used motor in the world due to its simplicity and ease of operation. The motor consists of 3 electromagnetic coils arranged around the stator. When three phase power is applied to the coils, a rotating magnetic field is generated. The stator starts to rotate because of mutual inductance; this is why it is called an induction motor. The speed of the motor can be accurately controlled by changing the frequency of the three phase power applied. Here is a list of some of the pros and cons:

- Pros
    - Speed Control: the AC induction motor has reliable speed control over a large range of speeds.
    - Reliability: these motors have stood the test of time as reliable. Since there is little to no mechanical interaction between the motor body and rotation shaft, the mechanical wear is less than its DC brushed cousin.
- Cons
    - Precise position control: the AC induction motor shines when it comes to precise velocity, but does not offer position control without an encoder.
    - Operation hardware: to get the most out of an AC motor, you need a Variable Frequency Drive to accurately control speed.

3.6.1.4: Servo Motor

The servo motor is a combination of a DC motor, reduction gears, and a control loop. They come in two flavors: closed loop which only rotate 180˚, and open loop, which have freedom of rotation. Only the closed loop will be considered. A closed loop servo motor consists of a DC motor with low torque and high rpms connected to a series of reduction gears. These gears increase torque and decrease rpms. The output shaft is connected to a potentiometer with feeds back to the motor control. Normally a servo is controlled by a Pulse Width Modulated (PWM) signal, which is turned into a voltage inside the motor. This closed loop feedback design allows for the servo motor to accurately move to any position in its range.

- Pros
    - Position Control: the servo motor is designed to give precise angular control over 180˚.

- Cons
    - Cost and availability: the servo motor is one of the most commonly used hobby motors and can be bought at a very low price.
- Cons
    - Limited range of motion: to get precise position control, a closed loop system must be used with at most 180˚ of freedom.

3.6.1.5: Piezoelectric Actuator

The piezoelectric actuator turns electrical potential into linear mechanical expansion using specific piezoelectric materials. These materials expand or contract along a certain axis when a voltage is applied. These actuators are very small and are widely used in microelectronics.

- Pros
    - High precision: piezoelectric actuators can move comfortably in the micrometer and even nanometer scale, making them popular in optics and microelectronics.
    - Simple hardware control: piezoelectric can be controlled by very simple circuitry.
- Cons
    - Very limited range: piezoelectric actuators are limited by the expansion of the material, which usually is in millimeter or micrometer scale.

3.6.2: Motor Technology Selection

The translation stage serves to move the power meter across the length of the beam. To accurately take measurements, the power meter needs to be moved across a wide range of the beam (300mm to 500mm) and be able to maintain a fine increment resolution (<50µm). There are a few technologies that provide linear translation; different metrics such as range, precision, and cost will be highlighted.

Table 21: Linear Translation Technology Comparison

Technology	Range	Precision	Motor Tech.	Relative Cost
Ball Screw	As long as screw, in the range of cm-m	<50µm (smaller can be achieved using an encoder or stepper microstepping)	Stepper or servo	Open loop stepper systems are within a couple hundred dollars, adding linear encoders to increase the price significantly.
Belt Drive	As long as the belt, in the range of meters	~100µm	Stepper	Very cost effective
Direct Drive Linear Motor Stages	Common models go to at least 300mm	<10µm	Direct Drive	Can go for around $1k

Piezoelectric Actuator Stage	In the range of µm-mm	Sub micron	Piezo	Couple hundred dollars

The translation stage must be able to traverse a meter of distance with less than 50µm precision the whole time. Because of space constraints, the optical design has two mirrors to reflect the beam back. A translation stage will move the two mirrors, effectively doubling the range of the stage. The downside is that precision doubles. Only a ball screw or belt drive system will provide enough range to operate on, but there will not be enough precision to scan the beam.

To get around the issue of precision, a second stage will be added to control the fine movement of the power meter. It is important this second stage has sub 25µm resolution. Either a ball screw or piezoelectric actuator gives an accurate resolution.

3.7: MCU Selection

The MCU is one of the most important parts of this project as it provides the main function. It is key that the microcontroller is capable of controlling the motors, reading sensors, and sending the results to the PC. The choice of MCU will dictate much of the system design if it is not capable enough.

There are a couple of commonly available MCUs, Expressive ESP32 series, ST STM32 series, and Microchip PIC32 series. The ESP32 offers a wide range of communication protocols, is easy to use, and has a lot of support. The STM32 offers some of the best hardware features, such as motor driving, ADC, and computation, but it is not as easy to setup and apply. The PIC32 is similar to the STM32, but more established. It is also not easy to impliment.

The ESP32 is a good choice for this project due to it being powerful and easy to use. There are two lines to consider, the S2 and S3

Table 22: ESP32 Comparison

Metric	ESP32-S3	ESP32-S2
Core	1 Core - 240MHz	2 Cores - 240MHz
Analog Signal Handling	2x SAR ADC	2x SAR ADC
USB Communication	Full Speed USB	Advanced Full Speed USB

The ESP32-S3 offers higher performance and more features than the S2, making it a good choice for the MCU.

3.8: Power Supply

3.8.1: Overview

The power supply is the most important part of the system, without it nothing could function, from logic to actuators. The system requires multiple different voltages to supply all the various parts. The design for the power system will need to include regulators of different voltages, currents, and distribution. The following sections will explore each in greater detail.

3.8.2: Component Power Requirements

Below is a table of all the components of the power distribution of the system and the relevant metrics for design:

Table 23: System Power Consumption

Component	Supply Voltage (V)		Supply Current (mA)		Supply Method
	Min	Max	Min	Max	
STM32G484	1.71	3.5	29.50	31.0	3.3V regulator
Photodiode Circuit	5	--	-29	--	5V regulator
Servo Motor	4.8	6.0	7.0	400.0	5V regulator
Stepper Motor	3.3	--	--	600.0	Stepper Motor Driver
Stepper Motor Driver (Logic)	3	5.5	--	10	3.3V regulator
Stepper Motor Driver (Motor Supply)	8	35	--	10	12V direct power

3.8.3: System Power

The input power supply for the system depends on the highest power demanded by the system. The system entails lower power components such as the MCU and photodiode, and the higher power components such as the servo motors and the stepper motors. The system only requires that one stepper be driven at a time. The stepper motor drivers can drive the stepper using any potential from 8-35V.

Of the standard wall adapters available, the most common and cheap option is a 12V 5A 60W charger. These are commonly used as laptop chargers and are widely available on the market. There are also a wide variety of DIN rail mountable power supplies with 10A current rating, as well as 24V options. But since the motors will not pull more than 1A, the 12V 5A 60W wall adapter will be sufficient for the system.

3.8.4: 3.3V Regulator

The 3.3V regulator will supply power to the MCU and logic power supply for the stepper motor driver. The reason both are supplied with 3.3V is that the stepper driver will need to recognize the pulse from the MCU when it rises to '1'. The 3.3V regulator will need to supply the following power requirements:

Table 24: 3.3V Regulator System Constraints		
Component	Max Current	*Total Power*
ESP32-S3-WROOM-U1	31mA	*0.1W*
A4988 Stepper Driver	10mA	*0.33W*
Total	*41mA*	*0.43W*

Here are some of the options for 12V to 3.3V regulator chips:

Table 25: 3.3V Regulator Product Comparison				
Part Num.	Max Vin	Max Iout	Expected Efficiency	Notes
TLV1117-3.3	16V	0.8A	~30%	This is a very common linear regulator on the market. It provides a low noise output, but at the cost of low efficiency.
LM3480-3.3	30V	1A	~30%	Has a very wide input voltage, provides clean output voltage at 100mA. But since it is a linear regulator it has low efficiency.
LT3045	22V	500mA	~30%	High precision 3.3V output voltage, not very good for efficiency.
LM2579-3.3	40V	3A	~85%	A cheap simple switcher with a wide Vin range and 3A current output. It has noisier output since it is a switching regulator.
LM2696-3.3	40V	3A	~90%	Very similar to LM2576 but operates at a higher frequency and thus requires smaller passive components. It comes with a higher price tag than the LM2579
TPS62125	17V	0.3A	~95%	This regulator provides high efficiency at low currents.

				The output voltage can be adjusted by external resistors.

Based on the above comparison, the LM2676-3.3V provides the best balance between efficiency, output range, and cost. There is a lot of documentation and availability that makes this part easy to use.

3.8.5: 5V Regulator

The three servo motors and the photodiode array will be powered by the 5V regulator output. At most only one servo motor will be used at a time, which will provide the current upper limit for the regulator.

Table 26: 5V Switching Regulator ICs

Part Num.	Max Vin	Max Iout	Expected Efficiency	Notes
LM2696-5.0	35V	3A	~90%	Very commonly used 5V regulator.
TPS62125	17V	0.3A	~95%	This regulator provides high efficiency at low currents. The output voltage can be adjusted by external resistors.
LM2596-5.0	40V	3A	~85%	Similar to the LM2676-5

To keep it consistent with the 3.3V regulator, the LM2679-5.0V switch regulator makes the most sense. It has good efficiency and is easy to acquire. Using the same regulator family between the 3.3V and 5V regulators will simplify the design of the system.

3.9 Communication Protocol

Reliable communication between sensors, controllers, and the user interface is essential to any embedded or measurement system. The Automated Laser Beam Characterizer requires efficient transfer of sensor data, configuration commands, and processed results between its subsystems. To achieve this, several established communication protocols are going to be used for their data rate capabilities, interoperability, and scalability. The protocols included are I2C, SPI, UART, and USB 2.0/CDC.

3.9.1: I2C (Inter-Integrated Circuit)

I2C is a synchronous, two-wire serial bus that supports communication between a master controller and multiple slave devices through unique addressing. It uses only two lines – SCL (clock) and SDA (data) - which makes it extremely efficient for connecting several low-speed peripherals with minimal wiring.

In this project, the I2C protocol is used primarily for low-bandwidth control and configuration tasks. So, the I2C bus links the STM32 microcontroller to several low-speed digital peripherals on the measurement board. These include the temperature-sensing IC used for thermal compensation of the photodiode array, a small EEPROM that stores calibration constants, and any auxiliary optical sensors mounted on the rotating-polarization head. Each device resides on the shared SDA/SCL line and is addressed individually by its 7-bit I2C address. This protocol was selected for these devices because it minimizes pin usage while allowing multiple slaves on the same two-wire bus. The sensors exchange small packets of configuration or status data, so the modest throughput of I2C (100 kHz – 1 MHz) is more than sufficient. In hardware terms, 4.7 kΩ pull-ups connect the bus to 3.3 V, and the STM's internal I2C peripheral, configured in Fast-Mode Plus, drives the communication. This architecture keeps wiring compact on the mixed-signal PCB and allows additional sensors to be added later with minimal redesign.

3.9.2: SPI (Serial Peripheral Interface)

SPI is a high-speed, synchronous serial protocol that supports full-duplex data transfer using four signals – MOSI, MISO, SCLK, and CS. It provides deterministic, low-latency communication that is widely used for ADCs, DACs, and high-speed sensors.

SPI will be the primary data-acquisition interface between the STM32 microcontroller and high-speed analog-to-digital converters or photodiode sensor modules. It connects the STM32 to the 12-bit ADC responsible for converting analog signals from the main photodiode detector and the optional polarization sensor. This choice is driven by the need for deterministic timing during real-time optical-power sampling. The ADC transmits digitalized samples at several hundred kS/s, which are captured by the microcontroller's DMA engine over SPI without processor intervention. Compared with I2C, SPI's full-duplex, clocked nature eliminates start/stop overhead and ensure phase-aligned sampling with the system timers that control the motorized translation and rotation stages. The result is low-latency, jitter-free transfer of raw intensity data, essential for accurate beam-profile reconstruction.

3.9.3: UART (Universal Asynchronous Receiver/Transmitter)

UART is an asynchronous serial protocol that transmits data without a shared clock using simple TX (transmit) and RX (receive) lines. It is widely used for low-speed communication and debugging interfaces due to its simplicity and universal compatibility.

The UART will serve as a diagnostic and development interface between the STM32 and an external computer during firmware development and calibration. The TX/RX pins are exposed through a programming header that can be accessed with a USB-to-Serial adapter. Through this link, engineers can issue manual text commands, monitor ADC readings, or verify FreeRTOS task execution using terminal software such as PuTTY. Because UART requires no clock synchronization and only two signal lines, it is ideal for early-stage debugging before USB connectivity is enabled. It also acts as a fallback communication channel if USB enumeration fails, ensuring that firmware updates and troubleshooting remain possible even when higher-level interfaces are unavailable.

3.9.4: USB 2.0/CDC (Communication Device Class)

USB 2.0 is a universal serial interface standard that supports data rates up to 480 Mbps. The Communication Device Class (CDC) - specifically the CDC-ACM (Abstract Control Model) subclass – allows USB devices to emulate a virtual COM port, enabling serial communication with a host computer through standard drivers.

USB 2.0 with CDC-ACM emulation is going to be the primary link between the embedded hardware and the host computer GUI. The STM32's on chip USB Full-Speed device controller enumerates as a virtual COM port, allowing the Python / PySide6 GUI to exchange JSON-encoded command and data packets via standard serial libraries. This interface carries the system's processed results – such as total optical power, beam waist, divergence, or Stokes parameters – from the microcontroller to the PC while simultaneously receiving configuration commands (motor step size, averaging count, measurement mode). It also supplies 5 V power for low-current operation, reducing the need for a separate supply during lab testing. The firmware module usb_comms.c manages packet framing and checksum verification, while the GUI's SerialLink class handles asynchronous reads and writes, ensuring continuous data steaming without blocking the user interface.

Table 27: Communication Protocol Comparisons

	I2C	**SPI**	**UART**	**USB 2.0/CDC**
Speed Range	100 kbps-1 Mbps	Up to 10-20 Mbps	9600 bps- 1 Mbps	Up to 12 Mbps (Full-Speed)
Wiring Complexity	2 wires (SDA,SCL)	4 wires (MOSI, MISO, SCLK, CS)	2 wires (TX,RX)	4 wires (VBUS, GND, D+, D-)
Typical Use in Project	Communication with temperature sensor, EEPROM,	High-speed data transfer from ADC and photodiode sensors	Debugging, firmware testing, and serial console	Main interface between MCU and PC GUI

	and auxiliary sensors			
Advantages	Simple, low pin count, supports multiple devices on one bus	Very fast and reliable, deterministic timing	Easy to implement, universally supported	High bandwidth, provides both data and power, plug-and-play
Limitations	Slower data rate, limited distance	Requires separate CS line per device	Not suitable for large data transfers	More complex firmware and protocol stack

By aligning each protocol with a specific subsystem – control, acquisition, debugging, and host interface – the communication design achieves both performance and modularity. I2C manages low-speed configurations; SPI ensures precise, high-rate data capture; UART supports development and fault analysis; and USB CDC provides a robust, user-facing channel integrated with the graphical interface. This layered approach mirrors professional embedded instrumentation practice, ensuring scalability and dependable operation throughout all phases of the Automated Laser Beam Characterizer's development and deployment.

3.10 Software

3.10.1: Embedded Software Technologies

3.10.1.1: Firmware Architecture Types

Bare-Metal Architecture is the simplest form of embedded software, where the program runs in a continuous loop – often called a super-loop – without an operating system. The firmware sequentially polls sensors, updates outputs, and repeats endlessly. This approach is efficient and lightweight, consuming minimal memory and processing overhead, which makes it suitable for simple, single-function devices such as digital thermostats or basic controllers. However, bare-metal systems struggle when multiple events need to occur concurrently, since all timing and scheduling must be manually handled in code. As system complexity increases, maintaining precise timing and responsiveness becomes difficult.

Interrupt-Driven Architecture improves upon the bare-metal approach by allowing hardware events to interrupt normal execution. When a sensor triggers or data arrives via a communication interface, an interrupt service routine (ISR) executes immediately to handle that event. This method enables faster responses to asynchronous inputs and better CPU utilization, since the processor can remain idle between events. Nevertheless, as the number of interrupts grows, managing priority levels and avoiding conflicts becomes challenging, and the overall firmware can become difficult to maintain.

To overcome these limitations, many modern embedded designs employ a **Real-Time Operating System (RTOS)**. An RTOS introduces a lightweight scheduler that divides the application into independent tasks or threads, each assigned a specific priority. It manages CPU time deterministically, ensuring that critical tasks – like sensor sampling or motor control – are always executed on schedule. In addition, RTOS frameworks such as FreeRTOS provide inter-task communication mechanisms, semaphores, and timers, which simplify development and enhance system reliability. This structure allows complex

embedded instruments, like the Automated Laser Beam Characterizer, to perform multiple real-time functions concurrently – data acquisition, communication, and control – without time interference.

FreeRTOS is an open-source kernel optimized for microcontrollers. It manages multiple threads – data acquisition, USB transfer, motor control, and system monitoring – under prioritized scheduling. Inter-task communication uses queues and semaphores, reducing latency and jitter. Alternatives such as Zephyr OS, Azure RTOS, or CMSIS-RTX provide similar features but demand more flash and configuration time. FreeRTOS's STM32CubeMX integration and proven reliability makes it ideal to be chosen for the Automated Laser Beam Characterizer.

3.10.1.2: Programming Languages

C Language

The most widely adopted language is C, which has been the industry standard for over four decades. C allows direct access to microcontroller registers and memory while producing compact, deterministic machine code. It is ideal for time-critical applications such as sensor sampling, motor control, and interrupt handling. In the Automated Laser Beam Characterizer, C was chosen because it provides full control over peripherals like SPI, I2C, and UART while maintaining predictable performance and small memory usage. Its low-level nature also allows developers to manipulate bits and ports directly, which is essential for precision hardware coordination.

C++ Language

C++ builds on C by adding object-oriented programming (OOP) concepts such as classes, encapsulation, and inheritance. These features improve code organization and reusability, making it easier to scale larger firmware projects. In embedded systems, C++ is often used selectively – encapsulating components such as motor driver or communication interfaces – while timing-sensitive routines remain in pure C. Modern embedded compliers now support a safe subset of C++ suitable for real-time applications, In the characterizer, small C++ modules might be considered for hardware abstraction layers and reusable components, while the core acquisition and control code remain in C to guarantee deterministic timing.

Python Language

Another high-level language sometimes used in microcontrollers is Python, primarily through interpreters like MicroPython or CircuitPython. These environments allow rapid prototyping and simplified syntax, making them popular in educational and low-power IoT applications. However, interpreted execution introduces significant latency and consumes more memory, making it unsuitable for tasks that demand precise timing, such as high-speed SPI sampling or real-time feedback loops. Consequently, while Python-based firmware can simplify testing, it cannot meet the deterministic performance needed in laboratory-grade instrumentation like the Automated Laser Beam Characterizer.

Table 28: Comparison of Embedded Programming Languages

Language	Performance & Efficiency	Hardware Control	Ease of Development	Real-Time Capability	Suitability for Characterizer
C	Very high efficiency, minimal overhead	Full low-level register access via HAL and direct memory	Moderate; requires hardware knowledge	Excellent with FreeRTOS and interrupts	Selected – ideal balance of speed and control
C++	High performance; slightly more overhead than C	Strong with abstraction layers (classes, objects)	Easier maintenance and modular design	Very good; works well with RTOS tasks	Used for modular drivers and reusable components
Python	Slow; interpreted execution	Limited GPIO/serial access only	Very easy for testing	Strong potential for concurrency	Future option for safety-critical upgrades

3.10.1.3: Embedded Development Environments

STM32CubeIDE (STMicroelectronics)

A complete development suite combining peripheral configuration (via CubeMX), C/C++ compilation, and debugging through ST-link. It integrates seamlessly with FreeRTOS and HAL drivers, making it ideal for STM32-based designs like the Automated Laser Beam Characterizer. The graphical pin-mapping and code-generation tools significantly reduce setup time and ensure consistency between hardware schematics and firmware initialization.

Xilinix Vivado Design Suite (AMD Xilinix)

A widely used environemtn for FPGA and SoC development. Vivado supports both hardware description (VHDL, Verilog, SystemVerilog) and software development through the Vitis framework for embedded processors (e.g., ARM Cortex-A9 cores). It includes block-design tools, IP catalogs, and on-chip debugging. Although Vivado is primarily hardware-oriented, it illustrates the crossover between software and digital design – useful if the characterizer was FPGA-based.

MPLAB X IDE (Microchip Technology)

An open-source, cross-platform IDE used for PIC and dsPIC microcontrollers. It supports both C and Assembly with compliers like XC8/XC16/XC32. MPLAB's debugger and simulator make it a popular choice for teaching embedded concepts and for low-power sensor-based applications.

Code Composer Studio (CCS) (Texas Instruments)

Designed for TI MCUs and DSPs, CCS integrates the TI C/C++ compiler, FreeRTOS configuration tools, and advanced real-time analysis. It is comparable in capability to STM32CubeIDE but optimized for MSP430, Tiva-C, and Sitara processors. Its strength

lies in low-power designs and precision analog control, both relevant to optical measurement systems.

Keil µVision and IAR Embedded Workbench

These are premium, industry-grade IDEs providing highly optimized compilers and in-depth debugging features such as code-coverage analysis and real-time logic tracing. They are widely used in medical, automotive, and industrial control devices where certification and efficiency are critical. However, both require expensive commercial licenses, making them less practical for student or open-source projects.

PlatformIO with Visual Studio Code

A modern, open-source development ecosystem that supports hundreds of microcontrollers and toolchains. It integrates with Git version control and continuous integration workflows, appealing to multidisciplinary teams. While externally flexible, it requires manual configuration of peripheral libraries and does not provide graphical hardware setup like CubeMX.

Each environment provides similar core functionality – editing, compiling, flashing, and debugging – but differs in ecosystem integration, cost, and hardware support. For this project, PlatformIO was selected because it aligns directly with the chosen ESP32 hardware, supports FreeRTOS, and USB CDC natively, and provides a rapid graphical configuration interface that simplifies peripheral mapping.

Table 29: Comparison of Embedded Development Environments

Development Environment	Supported Platforms / Devices	Ease of Use	Key Features	Limitations	Suitability for Characterizer
STM32CubeIDE	STM32 MCUs	High – integrates CubeMX and debugging tools	HAL/LL library generation, FreeRTOS config, ST-Link debugger, graphical pin mapping	Limited to STM32 family	Selected – ideal for STM32 development and FreeRTOS integration
Keil µVision	ARM Cortex-M/DSP MCUs	Moderate	Optimized compiler, RTOS aware debugging, trace features	Commercial license, cost-prohibitive	Excellent alternative for commercial optimization
Code Composer Studio (CCS)	TI MCUs, MSP430, C2000	Moderate	RTOS tools, analog/motor-control libraries	Tied to TI hardware only	Not suitable – not compatible with STM32
MPLAB X IDE	PIC, dsPIC, AVR MCUs	Easy for beginners	Graphical debugger, simulator, code configurator	Limited STM32 support	Not applicable to STM32 platform

Vivado / Vitis	Xilinx FPGAs and SoCs	Low – hardware design expertise required	Hardware/software co-design, IP integration	Complex setup, not MCU-oriented	Useful only for future FPGA expansion
PlatformIO (VS Code)	Multi-vendor MCUs (ESP32, STM32, AVR, etc.)	High for experienced users	Open-source build system, Git integration, CI/CD support	Manual HAL setup, no graphical pin config	Good for cross-platform open development

3.10.2: Host-Side Software Technologies

3.10.2.1: Programming Languages

Python

Python has become the de facto standard for modern scientific and engineering applications. It offers a vast ecosystem of open-source libraries – NumPy, SciPy, Pandas and Matplotlib – that streamlines numerical computation, data analysis, and visualization. Its simplicity and cross-platform compatibility make it ideal for rapid development. For the Automated Laser Beam Characterizer, Python was selected because it enables fast prototyping of data-processing algorithms and smooth integration with serial communication through the PySerial library. Python's PySide6 (Qt for Python) framework provides a professional-grade GUI environment capable of handling real-time updates and complex widget layouts while maintaining platform independence.

C++

C++ is also a strong candidate for host-side instrument control. It offers exceptional performance and native support for frameworks such as Qt 5/6 and wxWidgets. A C++ implementation can achieve lower latency and higher throughput than interpreted languages, which is advantageous in high-frequency measurement systems. However, the increased development time and lack of built-in scientific libraries make Python more attractive for research-oriented instrumentation like the characterizer, where development agility and flexibility outweigh the need for compiled-language speed.

C#

C# with the .NET Framework and Windows Presentation Foundation (WPF) remains popular in industrial and commerical measurement software. It provides a visually rich interface builder, strong integration with Windows APIs, and excellent USB/serial communication libraries. Nonetheless, it is platform-specific and not easily deployed on Linux or macOS systems. Given that the Automated Laser Beam Characterizer software aims to remain cross-platform for research labs using diverse operating systems, C# was not chosen.

MATLAB and LabVIEW

MATLAB and LabVIEW represent domain-specific, graphical programming environments historically used in laboratories and test facilities. MATLAB excels at matrix

computation, while LabView provides drag-and-drop instrument control via National Instruments' VISA drivers. Both enable rapid experimental development but are expensive and closed-source. For an academic, open-hardware system like ours, open-source alternatives provide greater flexibility and collaboration potential.

Rust and Go

Emerging languages such as Rust and Go were also reviewed for their concurrency and safety features. Rust, while efficient, lacks mature GUI toolkits and scientific plotting libraries, and Go focuses primarily on network and backend applications rather than instrumentation interfaces. For these reasons, Python remains the most balanced choice between accessibility, functionality, and computational capability.

Table 30: Comparison of Host-Side Programming Languages

Language	Performance	Ease of Development	Cross-Platform Support	Scientific Libraries	Suitability for Characterizer
Python	Moderate (speed adequate for GUI and data rates < 1 MB/s	Very high – simple syntax, rapid prototyping	Excellent (Windows / Linux / macOS)	Extensive – NumPy, SciPy, Pandas, Matplotlib, PyQtGraph	Selected – fast development and rich scientific ecosystem
C++	Very High (native machine code speed)	Moderate – steeper learning curve	Excellent (Qt is portable across OSs)	Strong GUI support (Qt), moderate scientific libraries	High performance option but longer development time
C#	High – JIT-compiled, optimized for Windows	High – powerful IDE and UI design tools	Limited (mainly Windows; partial cross-support via Mono / MAUI)	Good GUI tools, few scientific packages	Great UI but lacks cross-platform and scientific support
MATLAB	Moderate – optimized matrix math, slower I/O	High – visual App Designer interface	Moderate (Windows, macOS, Linux)	Excellent numerical toolboxes, limited custom GUI	Good for algorithm testing, not for distribution or budget
LabVIEW	High, real-time execution for hardware control	Very High – graphical block programming	Limited (Windows focus)	Built-in instrument control and DAQ libraries	Great for rapid instrument prototyping but non-portable and costly

3.10.2.2: Development Environments and Frameworks

Visual Studio Code (VS Code)

A lightweight, cross-platform IDE that supports Python, C++, and JavaScript with extensions for linting, debugging, and Git integration. VS Code's flexability allows direct integration of virtual environments and version control, making it suitable for research teams collaborating on open-source software. For the Automated Laser Beam Characterizer, VS Code was selected as the primary workspace for the Python GUI project because it provides rapid code iteration and built-in terminal support for testing serial interfaces.

PyCharm IDE (JetBrains)

A professional Python environment offering intelligent code completion, debugging, and refactoring tools. It streamlines dependency management but requires more system resources and licensing costs in its professional edition. It is useful for large-scale Python projects but offers fewer customization options compared wth VS Code's open ecosystem.

Qt Creator

A native C++/Qt development environment that provides advanced UI design capabilities, drag-and-drop widget editing, and efficient signal/slot debugging. Qt Creator would be the IDE of choice if the characterizer GUI were implemented in C++. Its tight integration with the Qt framework allows highly optimized, polished desktop applications but at the expense of longer development cycles.

Visual Studio (Community or Professional)

The primary environment of C#/.NET and C++ applications on Windows. It features powerful debugging, code analysis, and Windows-specific deployment tools. It is highly suited to professional instrument interfaces that will run exclusively on Windows workstations. However, its limited cross-platform capability makes it less desirable to open research instruments.

MATLAB App Designer and LabVIEW IDE

Both provide graphical user-interface builders connected to internal computation engines. They are excellent for quickly building instrument dashboards but lack open-source portability and deep version-control integration. These tools were evaluated but not chosen due to licensing costs and reduced integration flexibility compared with Python and Qt.

Table 31: Comparison of Host-Side Environments/Frameworks

Environment / Framework	Language	Advantages	Limitations	Suitability for Characterizer
VS Code	Python	Cross-platform, fast prototyping, scientific libraries	Moderate performance overhead	Chosen
Qt Creator	C++	High performance, Professional GUI	Longer development, steeper learning	Optional future upgrade
Visual Studio (.NET)	C#/C++	Polished UI, Strong debugging tools	Windows-only	Not selected

MATLAB / LabVIEW	Proprietary	Rapid GUI building, built-in analysis	Costly licenses, closed-source	Not selected

Chapter 4: Standards and Design Constraints

4.1: Standards

4.1.1: ISO 11146

When it comes to measuring the M^2 factor, it is important to follow the ISO standards to ensure the measurement is accurate. This requires taking beam profile measurements at several different points along the propagation of the beam. To understand this, it is necessary to know some terminology concerning the propagation of Gaussian beams. As mentioned in Chapter 3, a Gaussian beam, due to diffraction, will always have some divergence to it, which when propagating, corresponds to a smallest spot size, known as the beam waist. In an ideal diffraction-limited beam, these two variables, divergence and waist, are related by the following equation:

$$\theta = \frac{\lambda}{\pi w_0}$$

where,

- λ is the wavelength of the beam
- θ is the far field beam divergence
- w_0 is the beam waist

In the presence of wavefront error, however, this equation does not fit, but rather the divergence scales by a certain value, known as the M^2 factor. The equation then becomes:

$$\theta = M^2 \frac{\lambda}{\pi w_0}$$

Additionally, another parameter, known as the Rayleigh range (Z_R), is defined based on the value of the waist. This range is the distance it takes for the beam to double its initial beam area. This corresponds to a beam width scaling factor of $\sqrt{2}$. It is defined as shown in the equation below. Additionally, the figure below shows the general form of Gaussian beam propagation including the waist, divergence, and Rayleigh rang properties. [20]

$$Z_R = \frac{\pi w_0^2}{\lambda}$$

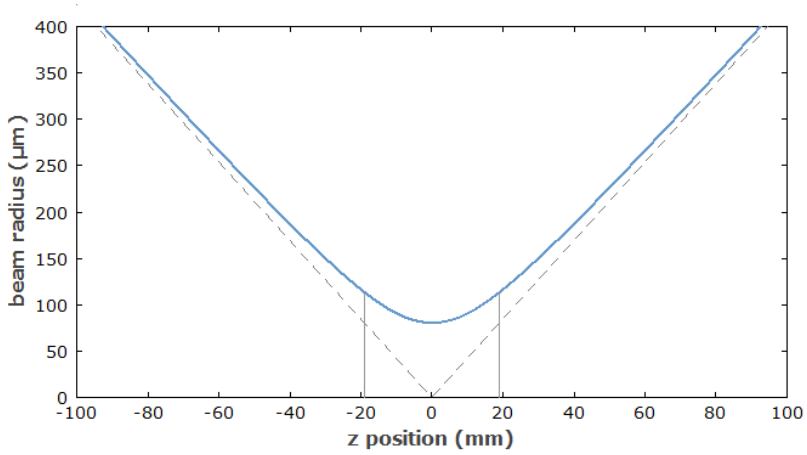


Figure 15: Graph of Gaussian Beam Propagation; the dashed lines show the divergence of the beam, which the actual beam matches asymptotically with distance from waist; the solid vertical lines indicate the Rayleigh range

These parameters are essential to know if an accurate measurement of the M^2 factor is to be obtained that adheres to the ISO standards. As stated earlier, to make an accurate measurement, the ISO 11146 standard requires taking multiple beam profiles across the beams path of propagation. Specifically, it requires that profiles be taken at five different points within one Rayleigh range, as well as an additional five points beyond two Rayleigh ranges. This is so that the data accurately reflects the true divergence and waist values.

When measuring the width of the beam, there are several different methods that can be used, but the ISO standard requires use of the D4sigma width, also called the second moments of the beam. This means that the diameter of the beam is defined as the distance corresponding to 4 standard deviations of the intensity distribution. This is the most flexible of all the definitions because it allows for consistent measurements even in the presence of higher order multimode beams that can often be oddly shaped, whereas other definitions, such as the Full Width Half Maximum (FWHM) or the 1/e^2 points are not as well defined in those situations. For a Gaussian beam, which this system will be measuring, this is equivalent to the 1/e^2 point of the intensity distribution.

4.1.2: Communication Protocol Standards

I2C (Inter-Integrated Circuit) Protocol

The I2C bus was developed by NXP (used to be Philips Semiconductors) in 1982 and has since become an industry-standard serial communication protocol for short-distance, intra-board communication. It operates on a two-wire bus – a serial data line (SDA) and a serial clock line (SCL) - that allows multiple integrated circuits to communicate through unique 7-bit or 10-bit addresses. Because all devices connect to the same bus through those unique 7-bit or 10-bit addresses, the protocol minimizes wiring complexity while maintaining reliable data transfer across components on a printed-circuit board (PCB). I2C can have a couple of different roles: a single master device that initiates and controls communication, and one or more slave devices that respond to commands. There are multiple operational speed modes available:

- Standard Mode (up to 100 kHz)
- Fast Mode (up to 400 kHz)
- Fast-Mode Plus (up to 1 MHz)
- High-Speed Mode (up to 3.4 MHz)
- Ultra-Fast Mode (up to 5 MHz)

Each mode is backward-compatible, providing designers with scalability for applications ranging from low-power sensor networks to faster digital converters. The protocol operates synchronously, with the clock line (SCL) driven by the controller to time all data transmissions, ensuring consistent sampling and setup-hold requirements across devices.

Communication is structured into frames consisting of an address byte, one or more data bytes, and acknowledgement but (ACK/NACK) to confirm successful transmission. Each transaction begins with a START condition and ends with a STOP condition, clearly defining bus ownership and preventing interference. Because of these standardized signaling conventions, I2C remains universally supported in modern microcontrollers, ADCs, DACs, EEPROMs, and sensor ICs across all areas of embedded and electronic engineering.

In the engineering field, I2C's popularity stems from its simplicity, scalability, and interoperability. It provides a uniform digital communication framework that minimizes PCB routing effort while maintaining compatibility across thousands of compliant components. As a result, the I2C standard has become a cornerstone of embedded-system design, enabling reliable, low-cost communication between processors and peripheral devices in applications ranging from consumer electronics to industrial automation.

SPI (Serial Peripheral Interface) Protocol

SPI is a high-speed, synchronous serial communication standard originally developed by Motorola in the late 1970s. It was introduced to provide a simple yet efficient way for digital devices to exchange data at a higher rate than asynchronous protocols like UART or multi-drop buses like I2C. SPI has become an industry standard for transferring large amounts of data quickly and deterministically between microcontrollers and peripheral devices.

The SPI standard defines a full-duplex, master-slave communication model based on four primary signals: Serial Clock (SCLK), Master Out Slave In (MOSI), Master In Slave Out (MISO), and Slave Select (SS). The master device generates the clock signal to synchronize data transfers, while the slave device shifts data in or out on opposite clock edges. This synchronous mechanism enables SPI to achieve clock speeds from a few kilohertz up to several tens of megahertz, depending on the device and interconnect length. Because it does not require addressing or bus arbitration, SPI achieves extremely low latency, making it the preferred standard for applications requiring high data throughput such as ADC sampling, flash memory access, and display control.

SPI's simplicity and flexibility have made it an enduring communication standard in engineering practice. It supports point-to-point as well as multi-slave topologies, where individual chip-select lines manage device selection on a shared bus. The lack of a formal addressing scheme reduces protocol overhead and contributes to its high efficiency.

Furthermore, its deterministic clocking and low power consumption make it suitable for real-time embedded systems, control networks, and mixed-signal devices where precise timing is essential. In engineering environments, SPI continues to be regarded as a robust, high-performance communication standard that bridges the gap between raw parallel data buses and slower serial interfaces.

UART (Universal Asynchronous Receiver/Transmitter)

The UART is one of the oldest and most enduring standards for serial digital communication. It defines the hardware and signaling conventions required to transmit and receive digital data in asynchronous serial form, meaning no shared clock signal is used between the transmitting and receiving devices. Instead, synchronization is achieved through predefined baud rates (bits per second) and precisely timed data framing. A typical UART frame consists of a start bit, a defined number of data bits (usually 7 or 8), an optional parity bit for simple error detection, and one or more stop bits to mark the end of transmission. These conventions ensure compatibility between devices using different microcontrollers, sensors, or computing platforms. UART communication requires only two main signal lines – TX (Transmit) and RX (Receive) - which makes it straightforward to implement on virtually all microcontrollers and peripheral devices.

The absence of a shared clock simplifies the interface hardware, making UART particularly popular for device configuration, firmware debugging, and data logging. Despite its simplicity, it remains an industry standard due to its universality, low cost, and interoperability across almost every embedded platform. In engineering applications, UART continues to serve as a fundamental communication interface for serial consoles, diagnostic ports, and peripheral configuration, forming the backbone of many system-control and testing environments.

USB (Universal Serial Bus) 2.0 / CDC (Communication Device Class)

The USB standard is a globally recognized communication interface developed and maintained by the USB Implementers Forum (USB-IF). It was introduced in the mid-1990s, USB was designed to standardize the connection, communication, and power supply between computers and external devices. The USB 2.0 specification, released in 2000, expanded on earlier versions by supporting higher data rates and improved device management. It defines three transfer modes – Low Speed (1.5 Mbps), Full Speed (12 Mbps), and High Speed (480 Mbps) - as well as detailed rules for bus topology, data framing, packet structure, and power delivery. These specifications ensure that USB-compliant devices from different manufacturers can operate together seamlessly without the need for custom drivers or wiring conventions.

Data transmission follows a packet-based protocol, where every communication transaction includes synchronization, addressing, error checking, and handshaking. The standard also integrates power delivery capabilities, supplying 5 V to 500 mA to connected peripherals, which enables devices to operate and communicate through a single interface. Within the USB specification, the Communication Device Class (CDC) defines a framework for devices that provide communication interfaces such as virtual serial ports, modems, and network adapters. The CDC-ACM (Abstract Control Model) subclass specifically outlines how a USB device can emulate a traditional COM port, ensuring

compatibility with existing serial communication software. This class standard is particularly valuable in engineering contexts because it provides a universal, driver-independent mechanism for serial data exchange between computers and embedded devices.

4.1.3 IPC-2221

When it comes to PCB design, there are standards that have to do with many of the different aspects of the design of a PCB, such as clearance and trace width standards. Often it is desirable to design PCBs as small as possible to save on cost and space in a project. However, there is a limit to how small the PCB can get before issues can arise such as parasitic devices and skin effects. The standards for clearance and trace widths are explained below.

4.1.3.1: Clearance Standards

When it comes to clearance, the main concern is that a significant electrical connection will arise between the two components and damage the circuit. This can be either a parasitic device or simply an undesired short between two parts of the circuit. Therefore, standards are made concerning how close things can be to each other. These are shown below: [24]

Feature	Clearance
Component leads	0.13 mm (up to a voltage of 50V)
Uncoated conducting areas (washers or similar mechanical hardware)	0.75 mm
Test probe sites	80% of the component height (0.6 mm minimum and 5 mm maximum)
Mounting hardware	Should not protrude more than 6.4 mm below PCB surface
PTH relief in the heat sink	2.5 mm larger than the hole (includes electrical clearance and misregistration tolerance)

Figure 16: List of Clearance Standards for PCB Design

4.1.3.2: Trace Width Standards

When it comes to trace widths, the concern is that the trace will not be able to handle the amount of current flowing through it and will burn. Thus, it is necessary to know how wide and thick the PCB traces are in order to be sure that there are no issues when the PCB is used. The image below shows the dimensions of a microstrip line on a PCB.

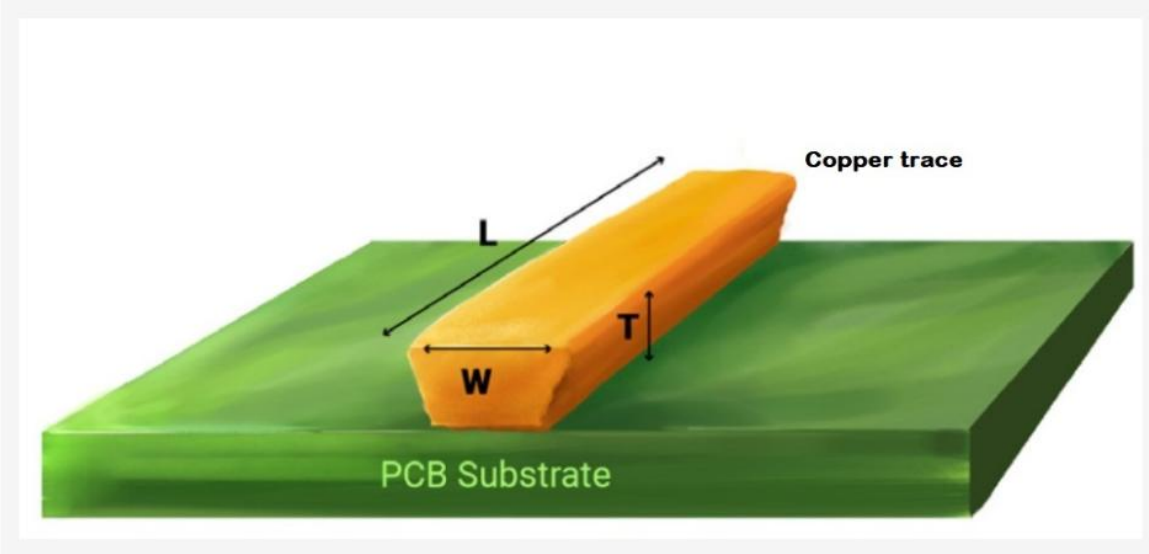


Figure 17: Diagram of a typical microstrip line PCB trace; Note that W corresponds to the width of the trace and T corresponds to its thickness. Both are necessary to know in order meet standards

When calculating these values, what is most relevant is the area of the trace and the density of the copper used. Since the trace thickness and copper density are determined by the PCB manufacturer, the minimum trace width can be calculated in the following manner.

$$Width\ [mil] = \frac{Area[mil^2]}{Thickness\ [oz] \times 1.378\ [mil/oz]}$$

Where the cross-sectional area is

$$A = \left(\frac{I}{[k \times (\Delta T^{0.44})]}\right]^{(1/0.725)}$$

Where

- I is the maximum current in Amps that will pass through the trace
- k is a constant (0.024 for internal layers; 0.048 for external layers)
- ΔT is the temperature rise above the ambient temperature in degrees C [24]

4.2: Design Constraints

4.2.1 Real-Time Processing and System Performance Constraints

A major technical constraint of the Automated Laser Beam Characterizer project is the requirement for real-time processing and deterministic system performance. The system must acquire, process, and transmit optical sensor data continuously and without interruption to ensure accurate measurement of beam power, divergence, and polarization. Any delay, jitter, or missed data sample would degrade measurement precision and introduce unacceptable errors in scientific results.

To meet this constraint, the microcontroller must manage multiple concurrent tasks – high-speed ADC sampling from the photodiode, motorized platform control, temperature sensing, and USB data communication – while maintaining synchronization between all subsystems. This combination of functions demands a high-performance microcontroller with strong real-time capabilities, a deterministic interrupt system, and support for a real-time operating system (RTOS).

The ESP32-S3-WROOM-1-N8 module was selected as the embedded processing unit due to its dual-core processing architecture, integrated wireless and USB functionality, and support for real-time task scheduling through FreeRTOS.

The ESP32-S3 features a dual-core 32-bit Xtensa® LX7 processor operating at up to 240 MHz, allowing one core to handle data acquisition and processing tasks while the other manages communication and peripheral control. This parallelism ensures that high-frequency ADC sampling and motor control routines can run concurrently without interfering with USB data transmission or sensor updates.

To ensure consistent scheduling, the firmware implements FreeRTOS, a lightweight kernal that divides execution into multiple prioritized tasks. Each subsystem – such as data acquisition, USB communication, and motor control – operates as an independent thread. FreeRTOS guarantees that time-critical tasks like SPI sampling always pre-empt lower-priority ones, maintaining predictable timing. The RTOS also simplifies system scalability by isolating functions into modular tasks, allowing future firmware updates (for example, additional sensors or a camera module) without redesigning the core schedular.

Additionally, the ESP32'S multiple high-resolution timers, nested vector interrupt, controller (NVIC), and hardware synchronization features support accurate timing down to the microsecond level. By combining these features with DMA and task priorization, the firmware achieves deterministic latency during continuous measurement cycles.

The constraint of real-time processing this directly influences hardware and software design choices across the system:

- Hardware: requires a high-speed MCU, sufficient DMA channels, and efficient interrupt architecture.
- Software: requires RTOS-based task management, non-blocking drivers, and interrupt-driven data flow.
- System Integration: requires precise clock configuration, interrupt prioritization, and efficient communication buffering.

In summary, the real-time performance constraint ensures that the Automated Laser Beam Characterizer maintains accurate, repeatable optical measurements while supporting concurrent processes. The ESP32's processing power, hardware acceleration, and RTOS compatibility make it uniquely capable of meeting these timing and performance demands.

4.2.2 Power and Thermal Management Constraint

The power and thermal design of the Automated Laser Beam Characterizer represents one of the most critical engineering constraints influencing both hardware architecture and microcontroller selection. Because the system must simultaneously power digital control

electronics, analog photodiode sensors, and motorized alignment mechanisms, its power subsystem must provide multiple regulated voltage rails with high efficiency, electrical stability, and minimal heat generation. In addition, thermal buildup directly affects optical and electrical precision; even small temperature shifts can cause photodiode sensitivity drift or alter ADC reference stability, leading to measurement errors.

The entire instrument operates from a 12 V DC wall adapter rated at 5 A (60 W), chosen for safety, cost, and adequate current headroom. From this primary source, two switching regulator stages generate the required logic and motor voltages. The LM25763 – 3.3V regulator powers the ESP32-S3-WROOM-1-N8 microcontroller and stepper-driver logic, while the LM2596 – 5.0 V regulator supplies the analog front-end and servo motors. These regulators were selected for their high conversion efficiency and ability to maintain tight output tolerances under varying loads. A two-stage regulation approach isolates digital and analog domains, reducing noise coupling between motor drivers and sensitive optical-sensing circuitry.

Measured current consumption illustrates how the design distributes power among subsystems. The MCU and low-power peripherals draw roughly 3-40 mA, the photodiode amplifier and ADC about 30 mA, and each servo or stepper motor as much as 400-600 mA when active. Consequently, most of the system's dynamic load arises from mechanical motion, while the control electronics remain relatively constant. The power architecture therefore had to provide sufficient surge current capacity for transient motor demands without allowing voltage sag that could reset the ESP32 or corrupt USB communication. Oversizing the 12 V supply and using local decoupling capacitors near each regulator mitigates this risk.

Because the entire system runs from external power rather than a large battery, thermal efficiency rather than absolute energy conservation becomes the primary concern. Power lost in the regulators and motor drivers appears as heat within a confined enclosure that also houses temperature-sensitive optics. To maintain reliable operation, component surface temperatures must remain below 80°C and the internal ambient ideally under 50°C. The design employs several passive-cooling strategies; large copper pour areas under the LM2576/LM2596 packages, thermal-via arrays connecting to internal ground planes, and vent openings in the enclosure to promote natural convection.

The ESP32 itself contributes to meeting these constraints through its low-power design. The MCU operates efficiently at 170 MHz while consuming less than 100 mW under normal conditions. It supports multiple power-saving modules – Sleep, Stop, and Standby – that allow dynamic scaling of clock speed and peripheral activity. During steady measurement periods, when motors are idle and data sampling proceeds at moderate rates, the firmware can invoke low-power modes or reduce clock frequency to limit total system current below 500 mA – the standard limit for USB-powered devices. These features ensure that the characterizer remains compliant with typical laboratory and USB safety standards.

Thermal management also intersects with system accuracy. Optical measurements rely on stable analog-to-digital conversion and photodiode linearity, both of which can drift with temperature. The ESP32'S integrated temperature sensor enables the firmware to monitor internal die temperature continuously. If thresholds are exceeded, the system can

temporarily throttle data acquisition or disable motor actuation until thermal equilibrium is restored. This closed-loop strategy prevents cumulative heat buildup that could distort readings or shorten component lifespan.

Safety and efficiency considerations further shaped the power network. Reverse-polarity protection diodes, fuses, and transient-voltage suppressors are going to be incorporated on the 12 V input to guard against adapter faults or user connection errors. Using switching regulators rather than linear ones minimizes wasted energy and ensures that the PCB does not become a significant heat source during extended operation. Proper grounding and trace width calculations keep voltage drops negligible even during simultaneous multi-motor operation.

In summary, the power and thermal constraint demanded a system capable of balancing energy efficiency, voltage stability, and thermal reliability withing a compact, UBS-powered design envelope. This requirement directly justified the choice of the ESP32-S3-WROOM-1-N8 microcontroller, whose low-power architecture, integrated thermal monitoring, and compatibility with efficient 3.3 V operation align perfectly with the characterizer's electrical and environmental goals. Through careful regulator selection, optimized current distribution, and passive thermal design, the characterizer achieves consistent performance, maintains temperature stability, and ensures long-term operational safety without exceeding its limited power budget.

4.2.3 Additional Constraints

4.2.3.1 Environmental Constraint

Environmental factors such as temperature, humidity, and ambient light can have a significant impact on the accuracy of optical measurements. Because the Automated Laser Beam Characterizer relies on sensitive photodiodes and precision optics, it must operate within a stable environment to minimize noise and drift. Thus, environmental stability forms a key design constraint in both hardware and software considerations.

Temperature fluctuations can cause mechanical expansion and contraction of the optical frame, leading to beam misalignment. Similarly, the photodiode's dark current and amplifier offset voltages are temperature-dependent, which can distort measurements if uncorrected. To mitigate these effects, materials with matched thermal coefficients (e.g., aluminum alloy) were selected for the optical baseplate, and temperature readings are continuously monitored during operation. The system can log temperature data to correct for gain variation in post-processing.

Ambient light interference poses another environmental challenge, particularly during low-power beam characterization. Optical housings and shrouds were designed to minimize stray light entry, and optical filters are used to ensure that only the laser wavelength of interest reaches the detectors.

By enforcing strict environmental control – through both mechanical design and software compensation – the characterizer ensures repeatable and accurate results even when operating in non-laboratory settings.

4.2.3.2 Safety Constraint

The use of laser sources introduces inherent safety risks, making user protection a mandatory design constraint for the project. The Automated Laser Beam Characterizer must comply with recognized laser safety standards (ANSI Z136.1 and IEC 60825-1) and integrate both hardware and procedural measures to ensure safe operation.

All exposed laser paths are enclosed within protective housings, preventing direct or reflected beam exposure. Any removable cover or inspection panel includes a magnetic interlock switch that automatically disables laser emission when opened. The system features visual – such as status LEDs or on-screen warnings – to alert users when the laser is active, or the safety interlock is bypassed.

Electrical safety was also a consideration, as the characterizer integrates motor drivers, voltage regulators, and USB-powered electronics. Overcurrent protection, fuses, and proper grounding will be implemented to prevent electrical hazards. All external surfaces remain below touch-safe voltage levels, ensuring compliance with general electrical safety requirements (UL/IEC 61010).

Through the combination of optical shielding, electrical protection, and user guidance, the Automated Laser Beam Characterizer achieves a high level of operational safety while maintaining flexibility for experimental use.

4.2.3.3 Economic Constraint

Economic feasibility was a primary consideration throughout the design of the Laser Beam Analyzer (LBA). The goal was to achieve laboratory-grade measurement performance—beam divergence, waist size, and polarization analysis—while staying well below the cost of commercial beam analyzers, which typically range from $5,000 **to** $15,000. Based on the Bill of Materials (BOM), the total project cost was $6,484, comfortably within the allocated $7,000 budget.

The majority of expenses arose from high-precision optical components, including the linear polarizer ($1,900), power meter ($1,060), and quarter-wave plate ($550). These items were essential for maintaining measurement accuracy and repeatability. In contrast, the electronic and control subsystems – including the ESP32 microcontroller, power regulators, and motor assemblies – were implemented using low-cost, open-source, or off-the-shelf components, accounting for less than 10% of total cost. This cost balance prioritized optical performance while minimizing non-critical expenses.

By leveraging open-source software, modular PCB design, and standard mechanical parts, the project achieved a high level of performance at a fraction of the commercial cost. The result is a financially sustainable and replicable optical instrument that demonstrates how careful cost allocation can meet technical objectives without exceeding project constraints.

4.2.3.4 Political Constraint

Although the Laser Beam Characterizer is a research and educational device, political and regulatory factors still influence its design and dissemination. Equipment involving lasers and optical sensors may be subject to export control regulations, particularly under ITAR (International Traffic in Arms Regulations) or EAR (Export Administration Regulations),

depending on intended use and wavelength range. To avoid such restrictions, the characterizer uses commercial off-the-shelf (COTS) components with no military or dual-use classification.

Additionally, international compatibility standards were considered during design. Compliance with **U**SB 2.0 (Full-Speed CDC Class) ensures global interoperability with host systems. The characterizer also follows FCC Part 15 Class B and CISPR 22 emission standards for unintentional radiators, reducing the risk of regulatory barriers during future commercialization.

By proactively designing regulatory neutrality, the characterizer remains freely distributable in academic settings and open to global collaboration without political limitations or certification issues. This foresight helps ensure that the project can be used internationally in classrooms, research labs, and open-source communities.

4.2.3.5 Manufacturability Constraint

The manufacturability of the characterizer affects its replicability and scalability for future iterations. The system was designed for ease of fabrication using standard machining and PCB manufacturing techniques. All electronic boards will be implemented as two-layer PCBs, keeping production costs low and simplifying assembly.

Mechanical components were designed around standardized metric hardware (M3 fasteners, 12 V barrel connectors, USB Type-B ports) and can be fabricated using CNC milling or laser cutting. Optical mounts and brackets follow standard optical breadboard hole spacing, allowing easy replacement or upgrade

From a production standpoint, the modular structure enables independent fabrication of the analog front-end, motor driver board, and power supply module, improving serviceability and reducing waste. By avoiding specialized components or tools, the characterizer can be manufactured in university labs, small workshops, or educational institutions without specialized equipment, making it suitable for teaching and research environments alike.

4.2.3.6 Sustainability Constraint

Sustainability is embedded throughout the Automated Laser Beam Characterizer's design, emphasizing low energy consumption, material recyclability, and long-term maintainability. The system consumes less than 10 W under normal operation due to its 90% efficient switching regulators, reducing both energy cost and waste heat.

Materials such as aluminum and stainless steel were selected for their durability and recyclability, while RoHS-compliant soldering prevents hazardous material use. Components that are most prone to wear – optical mounts, cables, and connectors – are designed to be easily replaceable, extending product lifespan and minimizing electronic waste.

Software sustainability was also addressed through modular architecture. The GUI and firmware can be updated or expanded without hardware redesign, ensuring technological longevity. Together, these design choices create a system that not only functions efficiently but also aligns with the principles of environmental stewardship and responsible engineering.

Chapter 5: Application of ChatGPT and Other Similar Platforms

The use of artificial intelligence (AI) in schools and industry workflows has changed how students and professionals research, design, and document complex systems. Large language models (LLMs) such as ChatGPT (OpenAI), DeepSeek, Microsoft Copilot, or Google Gemini can generate human-like text, assist in problem solving, and summarize technical material from a wide range of topics. In the context of Senior Design, these AI tools function as assistants that support rapid information gathering, report structuring, idea development, and sometime code developers.

Overall, our team has made use of ChatGPT for help with research on multiple topics, helping with selecting specific components suitable for the project, etc. In the following case studies, we will compare the differences of chat prompts from multiple AI models. Each case study will have the same prompt used for each AI model with results given and how accurate and useful they were to the project.

5.1 Case Study 1

Please refer to Appendix C [1a], [1b], and [1c] for case study 1 given prompt and outcomes from ChatGPT and Deepseek.

ChatGPT [1b] did a good job of breaking down criteria to consider when trying to select a MCU; it gave a detailed bullet point list with examples of what to look for when researching MCUs. It gave links and pictures to highly recommended MCU/dev boards with prices included along with a breakdown of the different specs of each listed board. Then ChatGPT gave its recommendation based on project type which is useful for helping narrow down which MCU to select. Overall, I would say ChatGPT did a good job of giving insight into how to select a MCU for a certain kind of project along with giving good sources to check out.

DeepSeek [1c], gave pretty similar results about MCU selection as ChatGPT did. One thing it did differently was give you a decision flow chart based on needs for MCU which is nice to help someone quickly decide what kind of MCU they are looking for and narrow down the choice. DeepSeek also gave detailed bullet points for types of MCUs along with their functionalities and why you would pick it. But it did not give links to sources to look at the MCU and price ranges were not given like in ChatGPT.

5.2 Case Study 2

Please refer to Appendix C [2a], [2b], [2c] for case study 2 given prompt and results from ChatGPT and Google Gemini.

ChatGPT [2b] gave a good walkthrough on how to build a Python-based GUI. It first walked through how to choose a framework to use as a GUI library source and then gave example code snippets on building the GUI using those different libraries. This would be helpful for people trying to figure out a good starting point in building a GUI if they had never built one before and needed some ideas.

Google Gemini [2c] on the other hand gave a simple rundown of the steps needed to build a GUI along with small snippets of code to help get you started. It walked you through picking the correct Python GUI framework first which is helpful when trying to decide how to build the GUI and if it will be easy to use. It then gives general steps to build the GUI from scratch after picking the framework. At the end of the conversation, it gives a YouTube video of an overview of different Python GUI options and what they look like.

Overall, you could say that ChatGPT gave more precise answers with prompts to help the conversation along. ChatGPT, DeepSeek, and Google Gemini all have their own strengths and weaknesses, but I feel like ChatGPT is more powerful in regard to information research and computing time. DeepSeek was a close second in the vein of being helpful to this kind of project research while Google Gemini I believe is more for common knowledge lookup and information confirmation.

Chapter 6: Hardware Design

6.1: Optical Design Calculations

6.1.1: Optical Lens Design

One of the main constraints comes in the optical design of the M^2 factor setup. As mentioned above, to meet ISO Standard 11146, enough beam profiles must be taken at different points in the path of beam propagation. Specifically, at least five points must be taken within the Rayleigh range, and another five outside two Rayleigh ranges. This puts a limitation on what focal lengths are feasible to meet the specifications of the project. If the focal length is too short, the Rayleigh range might be too short for common motorized stages to have enough step resolution to take the required number of points. On the other hand, if the focal length is too long, a small input beam will have a very long focal length, making it difficult to have a stage long enough to profile beyond the two Rayleigh range mark. Therefore, proper calculation and simulation should be conducted to ensure the right focal length lens is chosen. This is done by considering how the Rayleigh range will be affected by different focal lengths. As most stepper motors have a set step resolution, it is evident that the more difficult constraint is the large input beam constraint, corresponding to a short Rayleigh range. As this is the case, this will establish the constraint of the focal length of the lens, while the second constraint corresponding to a small input beam and a

long Rayleigh range will be left to the translation distance of the profiler. The Rayleigh range of the focused beam can be described by the following equation.

$$Z_R = \frac{4\lambda f^2}{\pi M^2 D^2}$$

where,

- Z_R is the Rayleigh range of the focused beam
- λ is the wavelength of the beam
- f is the focal length of the lens
- D is the diameter of the beam
- M^2 is the beam quality factor of the beam

Given that most stepper CNC motors can have a resolution up to 50 um, it is necessary that the Rayleigh range of the focused beam be at least 250 um. Assuming a maximum M^2 factor of 2.5, wavelength of 2 um, and a maximum beam diameter of 35 mm, this leads to the following calculation for the constraint of the focal length.

$$\lambda = 2\ \mu m \qquad M^2 = 3 \qquad D = 35\ mm \quad Z_R = 250\ \mu m$$

$$250\ \mu m < \frac{4 * (2\ \mu m) * f^2}{\pi(3) * (35\ mm)^2}$$

$$f^2 > 361\ mm$$

$$f > 600\ mm$$

This puts a large constraint on the lens which can be used in the system. On the other hand, for small beam sizes, the Rayleigh range gets very large, and therefore it is necessary to calculate how long the Raleigh range will be at a small input beam size. This can only be known once the focal length of the lens in the system is known. As discussed in Chapter 2, the focal length chosen for the system is 750 mm. Therefore, the same equation used above can be used to calculate the longest expected Rayleigh range at the smallest beam diameter of 4 mm.

$$\lambda = 2\ \mu m \qquad M^2 = 1 \qquad D = 4\ mm \qquad f = 750\ mm$$

$$Z_R = \frac{4 * (2\ \mu m) * (750\ mm)^2}{\pi * (1) * (4\ mm)^2}$$

$$Z_R = 90\ mm \approx 100\ mm$$

Therefore, since the Rayleigh range at the smallest input beam size is approximately 100 mm, it follows that to scan sufficiently outside two Rayleigh ranges as ISO standards require, the translation stage must be able to scan more than 200 mm on both sides of the waist, or 400 mm for the total beam width. This is a rather long distance to travel, and therefore it will be necessary to modify the path of the focused beam so that the system does not become too large. The figure below shows the designed method for compensating for this large translation range.

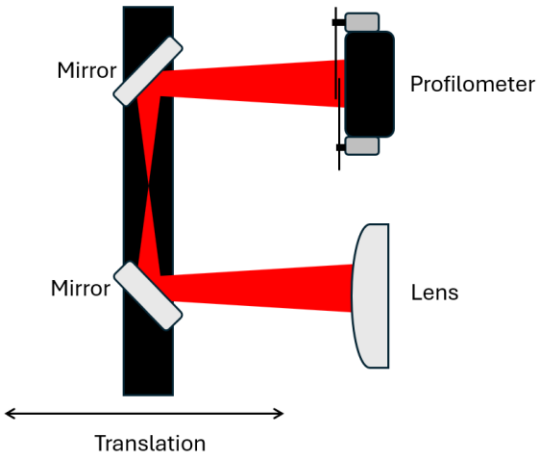


Figure 18: Modified M2 factor translation stage design to compensate for the long Rayleigh ranges at small beam sizes

As can be seen in the figure above, two mirrors aligned together on a stage can be simultaneously moved across the translation stage, while the profilometer remains stationary. This permits the whole beam path to be profiled while only translating half the necessary distance. So, in the smallest beam size case where the Rayleigh range is very long, instead of needing to translate a minimum of 400 mm, only 200 mm of translation is necessary. While this is still a significant distance, it is a great improvement to the spatial layout of the optical system. The downside to this, of course, is that the effective step size between the beam profiles has doubled, potentially making it harder to measure the points within the Rayleigh range. To verify that the new layout does not affect the large beam size measurements, the Rayleigh range calculations can be done again but for the largest required beam size with the 750 mm focal length. These are shown below.

$$\lambda = 2\ \mu m \qquad M^2 = 3 \qquad D = 35\ mm \quad f = 750\ mm$$

$$Z_R = \frac{4 * (2\ \mu m) * (750\ mm)^2}{\pi * (3) * (35\ mm)^2}$$

$$Z_R = 390\ \mu m$$

Since the effective step resolution of the stage has been increased to 100 µm, this means that the Rayleigh range is too small to meet ISO standards. Therefore, in order to get around this, the profiler will be placed on a small translation stage with the same 50 µm step resolution as the longer translation stage. Thus reduces the effective step resolution of the system to 50 µm, which is sufficient to meet ISO standards. In this way the system will be designed to satisfy all the requirements.

6.1.2: Michelson Mirror Speed Calculations

When it comes to the Michelson Interferometer, it is necessary that the light intensity signal be properly interpreted to get an accurate wavelength measurement. This constraint is best described by the Nyquist-Shannon Sampling Theorem, which states that when sampling a

signal, the sampling rate must be at least twice the bandwidth of the signal to avoid aliasing. This is an essential theorem to consider for this application, since the interference pattern signal that arises due to the translation of the mirror can change very fast. According to the datasheet for the selected MCU, the maximum sampling rate of the ADC is 100 ksps. This means that to not alias the intensity signal, the signal must not be faster than 50 kHz. This value can be directly correlated to the speed of the mirror as it is moved in the interferometer. As explained in Chapter 3, the following equation expresses the relationship between the distance traveled, the number of periods passed in the intensity signal, and the wavelength of the light being measured.

$$m\lambda = 2\Delta x$$

where,

- m is the number of intensity cycles passed
- λ is the wavelength of the laser beam
- Δx is the distance traversed by the mirror

Taking the derivative of this equation with respect to time yields an equation that relates the frequency of the intensity signal with respect to time to the speed of the mirror movement. The equation becomes:

$$\frac{dm}{dt} = \frac{2}{\lambda}\frac{dx}{dt} = \frac{2}{\lambda}v$$

Therefore, if the rate of the intensity signal must not exceed 50 kHz, the maximum speed of the mirror can be calculated as the following.

$$50\ kHz \geq \frac{2}{2\ \mu m}v$$

$$v \leq 50\ mm/s$$

Therefore, the maximum speed the mirror can translate while still satisfying Nyquist's criteria is 50 mm/s. This is a very reasonable speed to stay under, and so the main issue with getting an accurate signal will not be under sampling, but rather that of noise in the measurement and uncertainty in the mirror displacement value.

6.1.3: Gaussian Beam Clip Ratio Calculations

As mentioned in Chapter 2, the clear apertures of the optics cannot be the same size as the largest specified beam width. This is because the beam width as defined by ISO standards is the D4sigma point, which for a Gaussian beam is equivalent to the 1/e^2 point. Thus, approximately 73% of the beam's energy lies within what is called the 'beam width' with the other 27% remaining outside it. Therefore, in order to get accurate measurements, it is necessary that the apertures be larger than this. To know exactly how large, the standard Gaussian beam intensity equations can be used to find out how much of the power in the beam is clipped by an aperture of radius r. The equation below describes the spread of the energy distribution of an ideal untruncated Gaussian beam. [25]

$$I(r) = I_0 e^{-2\frac{r^2}{w^2}}$$

Now the total power of this beam extends from a radius of zero to infinity, so there will always be some power that is clipped from any aperture. The equation that expresses the loss in power is then the total power minus the power in the aperture, which can be expressed in the following way.

$$\frac{P(a)}{P_{total}} = 1 - e^{-2a^2/w^2}$$

Where,

- P(a) is the power in an aperture of radius a
- P_{total} is the total power from radius zero to infinity
- a is the radius of the aperture
- w is the width of the beam in the aperture

The fraction on the left-hand side of the equation represents the power transmitted through the aperture due to clipping. Therefore, solving for a/w yields to the following result.

$$P_{clip} = 1 - e^{-2a^2/w^2}$$

$$e^{-2a^2/w^2} = 1 - P_{clip}$$

$$\frac{a}{w} = \sqrt{-\frac{1}{2}\ln\left(1 - P_{clip}\right)}$$

Where,

- a is the radius of the aperture
- w is the radius of the beam in the aperture
- P_{clip} is the ratio of power transmitted though the aperture to the total power

So, if the power measured is to be within 5% accuracy of the total power of the beam, the minimum aperture radius corresponding to this power loss can be calculated as follows.

$$P_{clip} = 0.95 \quad w = 35\ mm$$

$$a = 35\ mm \times \sqrt{-\frac{1}{2}\ln\left(1 - 0.95\right)}$$

$$a \approx 42\ mm$$

Therefore, in order to get the required power measurement, the clear aperture of the necessary optics in the system should be greater than this.

6.1.4: Polarization Extinction Ratio Calculations

To estimate the required PER of the linear polarizer, the equations for calculating the Stokes Parameters can be used to calculate the uncertainties in the intensity measurements corresponding to their uncertainties in the normalized Stokes Parameter measurements. Assuming the angle of the quarter waveplate has no uncertainty, the uncertainty of the

Stokes Parameter values will correspond only to the uncertainty in the intensity measurements. This can be shown in the following way.

$$S_0 = A + \epsilon_A - (C + \epsilon_C)$$

$$S_1 + \epsilon_{S1} = 2 * (C + \epsilon_C)$$

$$S_2 + \epsilon_{S2} = 2 * (D + \epsilon_D)$$

$$S_3 + \epsilon_{S3} = B + \epsilon_B$$

Where,

- S_0, S_1, S_2, and S_3 are the four Stokes Parameters
- A, B, C, and D are the coefficients in the angle vs. intensity equation explained in Chapter 3
- ε_{S1}, ε_{S2}, and ε_{S3} are the errors in the calculated Stokes parameter values
- ε_A, ε_B, ε_C, and ε_D are the errors due to power miscalculations in the A, B, C, and D coefficients respectively

Note that S_0 does not have an uncertainty because these are the normalized Stokes parameters, where the value of S_0 is always equal to one. Now the errors in the B, C, and D coefficients are due to errors in intensity measurements, as stated above. These intensity measurements are averaged, so the uncertainty will decrease by the square root of the number of data points taken. Assuming this value is 8, the minimum number of points taken to meet Nyquist's criteria, the uncertainties will then translate to the following.

$$\epsilon_{S1} = 2 * \frac{\epsilon_I}{S_0\sqrt{N}}$$

$$\epsilon_{S2} = 2 * \frac{\epsilon_I}{S_0\sqrt{N}}$$

$$\epsilon_{S3} = \frac{\epsilon_I}{S_0\sqrt{N}}$$

Where,

- ε_{S1}, ε_{S2}, and ε_{S3} are the errors in the calculated Stokes parameter values
- N is the number of points taken
- ε_I is the intensity error
- S_0 is the power being measured

Since S_0 is normalized to one, the intensity errors can be calculated from the required Stokes parameter errors. The value of N will be set equal to 8 as it is the minimum number of points needed to be taken. As S_1 and S_2 require less intensity error for the same error in their calculated parameter, the required intensity error reduces to their equation shown below.

$$\epsilon_I = \frac{\sqrt{N}}{2}\epsilon_S = \frac{\sqrt{8}}{2} * 0.05 = 0.07$$

Thus, the error of the power measurement at each point must be within 7% if the calculations are to be accurate. Converting this to a Polarization Extinction Ratio (PER) in dB yields a necessary PER of about 11.5 dB.

$$\epsilon_I = \frac{1}{PER}$$

$$PER(dB) = 10 * log_{10}\left(\frac{1}{0.07}\right) = 11.5\ dB$$

6.2: Photodiode Circuit

When designing the photodiode circuit, it is first necessary to know how much current will be delivered to the amplifier. Thankfully, this is a relatively easy value to control, as the optical power on the photodiode can be controlled by rotating the quarter waveplate in the polarimeter. To use a safe and easy value, it will be assumed that the photodiode will have approximately 1 mW of optical power incident on it. When looking for photodiodes, a High Speed InGaAs photodiode was found with the following specifications. Note that VR denotes the reverse bias voltage on the diode. [14]

Table 32: Photodiode Specifications

Parameter	Condition	Value	Units
Spectral Range	VR = 0 V	800 - 2600	nm
Breakdown Voltage	IR = 100 uA	1	V
Dark Current	VR = 1 V	300	uA
Junction Capacitance	VR = 0 V	1000	pF
	VR = 1 V	85	pF
Shunt Resistance	VR = 10 mV	3.5	kOhms
Responsivity	λ = 2400 nm	1.24	A/W

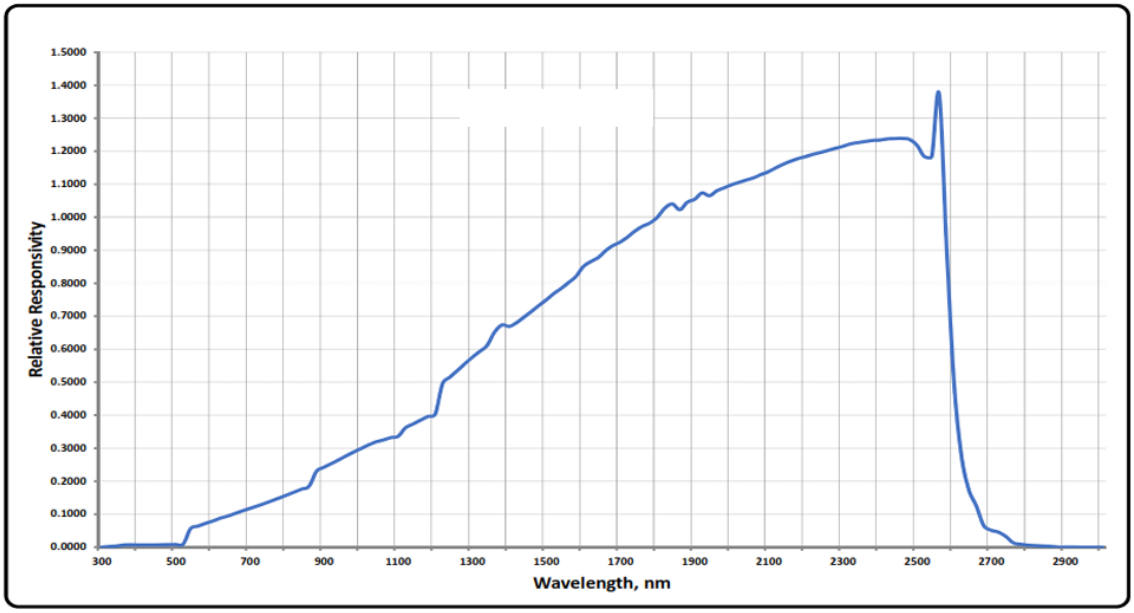


Figure 19: Photodiode Responsivity Curve

With these specifications, the circuit can begin to be designed. Since the responsivity peaks around 1.24 A/W, as seen in the figure above [14], it can be assumed that if 1 mW of optical power is on the photodiode, the maximum current it will generate is 1.24 mA. This information allows for the value of the feedback resistor to be calculated. To allow for reasonable dynamic range of the amplifier, the system will be designed to amplify between 0.1-3.2 V. The value of the feedback resistor can therefore be calculated using Ohms Law, seen in the equation below. Note that the dark current must be added to the value of $I_{IN(MAX)}$ in order to not scale the voltage higher than expected

$$R_f = \frac{V_{OUT(MAX)} - V_{OUT(MIN)}}{I_{IN(MAX)} + I_{Dark}} = \frac{3.2\,V - 0.1\,V}{1.24\,mA + 0.3mA} = 2.01\,k\Omega \approx 2\,\text{k}\Omega$$

With this value, the current from the photodiode can now be transformed into an amplified voltage signal. As discussed in Chapter 3, the bandwidth of the signal can be set with a feedback capacitor, which creates a pole in the transfer function. As shown in Chapter 4, the sampling frequency of the ESP32 ADC is around 100 kHz. To make sure the bandwidth of the amplifier covers this value, the pole frequency will be set to this value. From this the constraint of the feedback capacitor can be calculated, shown below.

$$C_f \leq \frac{1}{2\pi f_p R_f} = \frac{1}{2\pi * (100\,kHz) * (2\,k\Omega)} = 796\,pF$$

To keep the pole frequency well above 100 kHz, the value of the feedback capacitor was chosen to be 560 pF. This should allow for the required gain to remain constant across the necessary bandwidth.

Though this sets the necessary gain and bandwidth, the photodiode still needs to be reverse biased so it can operate in photoconductive mode. As seen in the photodiode specifications tabulated above, the junction capacitance decreases drastically when it is reverse biased.

The effect of this is that the necessary gain-bandwidth product needed by the op amp will be much lower than if the photodiode were left in photovoltaic mode. To reverse bias the diode, a simple voltage divider will be connected from the positive power supply to the non-inverting input of the op amp. To make sure the photodiode junction capacitance is small, the reverse bias will be set to 0.5 V. Therefore, to set the correct divide ratio, values of 13.7 kΩ and 2.5 kΩ were chosen for R_2 and R_3, respectively. This will result in approximately a 0.509 V bias on the photodiode, as calculated below.

$$V_{bias} = V_{CC}\frac{R_3}{R_2 + R_3} = 3.3 \times \frac{2.5\ k\Omega}{2.5\ k\Omega + 13.7\ k\Omega} = 0.5\ V$$

The last thing to do is select an appropriate op amp. This op amp must meet several different criteria. First, its gain-bandwidth product in combination with its input capacitance must satisfy the following formula. [23] Here C_{in} is assumed to be dominated by the junction capacitance and so an approximation is made using the photodiode junction capacitance of 85 pF.

$$f_{GBW} > \frac{C_{in} + C_1}{2\pi R_1 C_1^2} = \frac{85\ pF + 560\ pF}{2\pi * (2\ k\Omega) * (560\ pF)^2} \approx 164\ kHz$$

Additionally, the slew rate of the op amp should exceed or be close to the maximum frequency discussed above. This is so that the op amp does not distort the data as it amplifies the current generated in the photodiode into a voltage signal. Among this, it must be able to accept the designed supply voltage of 3.3 volts. The specifications of the op amp selected are shown below.

Table 33: Op Amp Specifications (3.3 V Supply)

Parameter	Value	Units
Unity Gain-Bandwidth Product	1	MHz
Slew Rate	1.9	V/µs
Phase Margin	70	degrees

For the final piece of the circuit, an LED was added to have easy verification that power is getting to the circuit. A red LED was found that has a maximum forward voltage of 2.3 V and a maximum current rating of 20 mA [26]. Therefore, since the supply voltage is 5 V a resistor will be placed in series with the LED to limit the current it can draw. Using Ohm's Law, the value of this resistor can easily be calculated.

$$V_{Supply} - V_{LED} = I_{Max}R$$

$$3.3\ V - 2.3\ V = (20\ mA) * R$$

$$R = 50\ \Omega \rightarrow R = 100\ \Omega$$

With all the necessary calculations complete, the schematic can be designed. For this circuit, Altium Designer was used. For the photodiode and Op Amp components, custom schematic symbols were used to keep the schematic more readable. A single 3-pin header is needed for communication with the rest of the system, with the three pins corresponding to the 3.3 V supply, the output voltage signal, and ground. Below is the finished schematic.

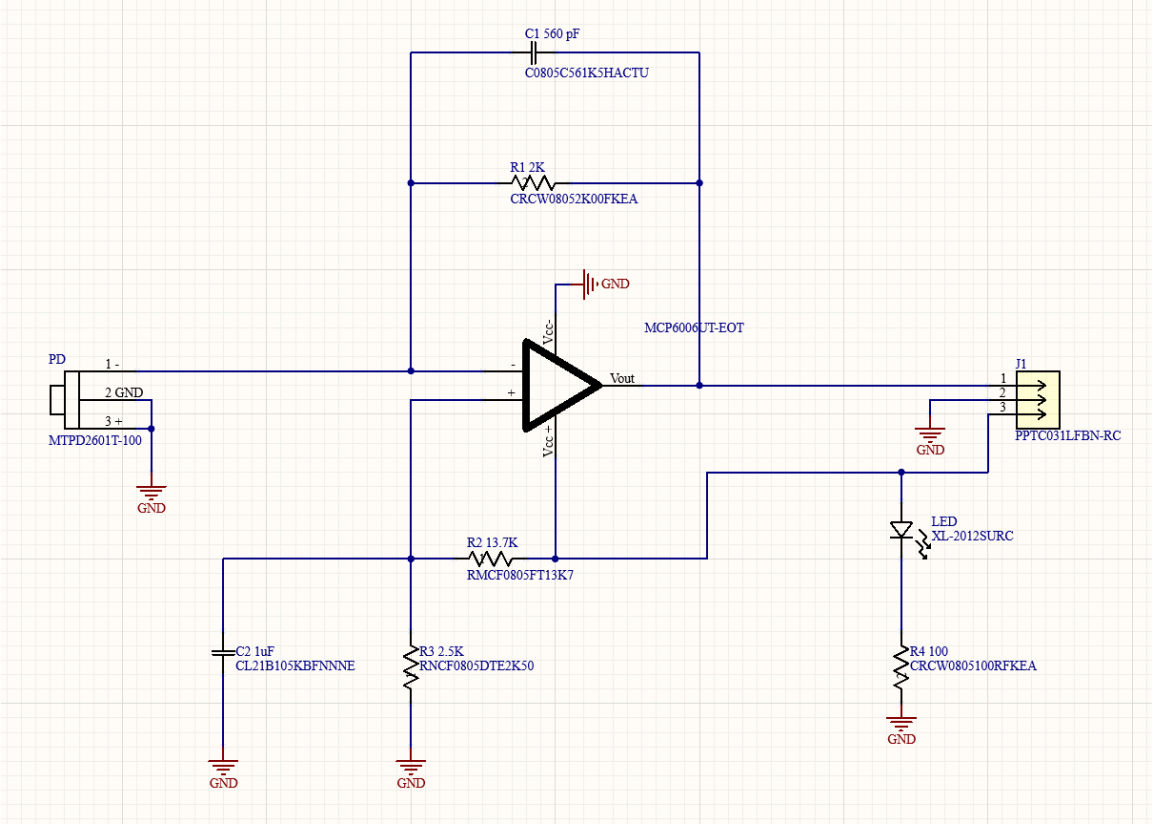


Figure 20: Completed Photodiode Circuit Schematic

To verify the gain and bandwidth of the design, the schematic was put into LTspice simulation software. The only change was that the photodiode was simulated using three components: a current source, a shunt capacitor, and a shunt resistor. The current source simulated the current that is generated in the photodiode when light is incident on it, while the shunt capacitor and resistor simulate the junction capacitance and shunt resistance of the photodiode, respectively. Two different simulations were done. The first simulation was a simple current to voltage transfer function analysis. The current source was increased linearly, and the output voltage was plotted against it. The results of the first simulation are shown below.

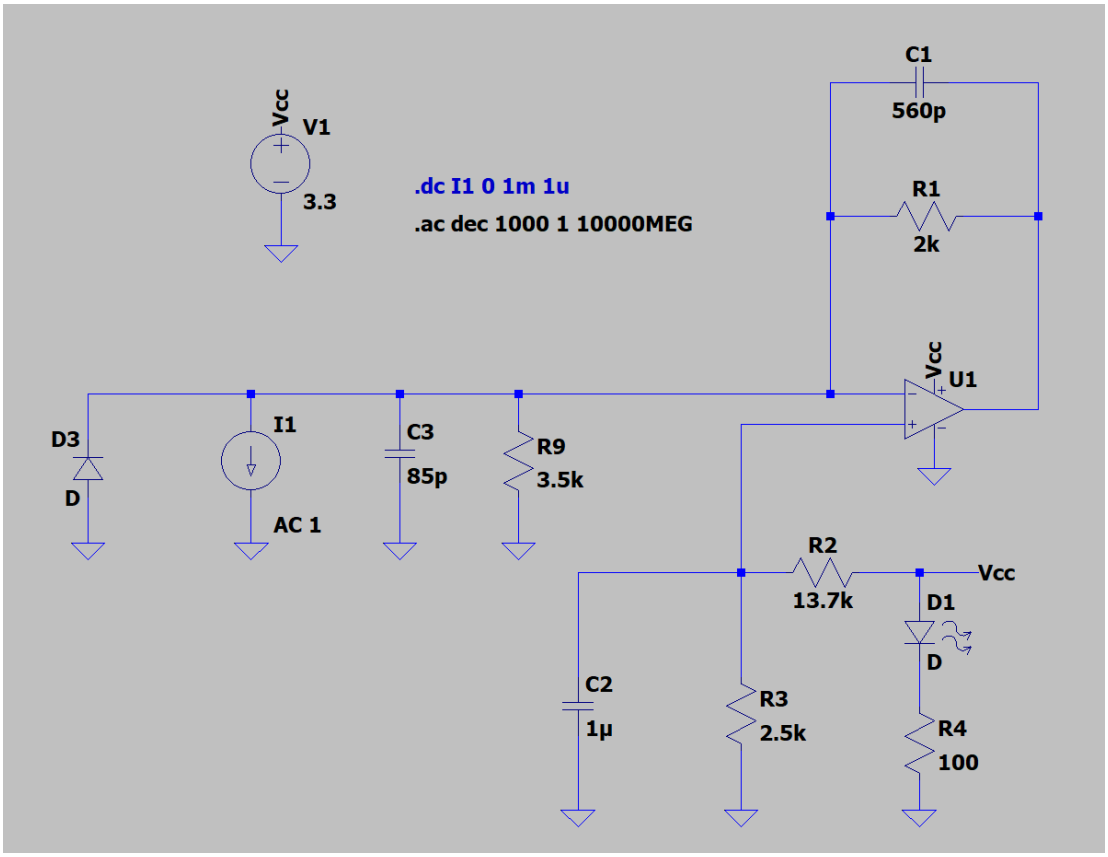


Figure 21: LTspice Circuit Simulation Schematic

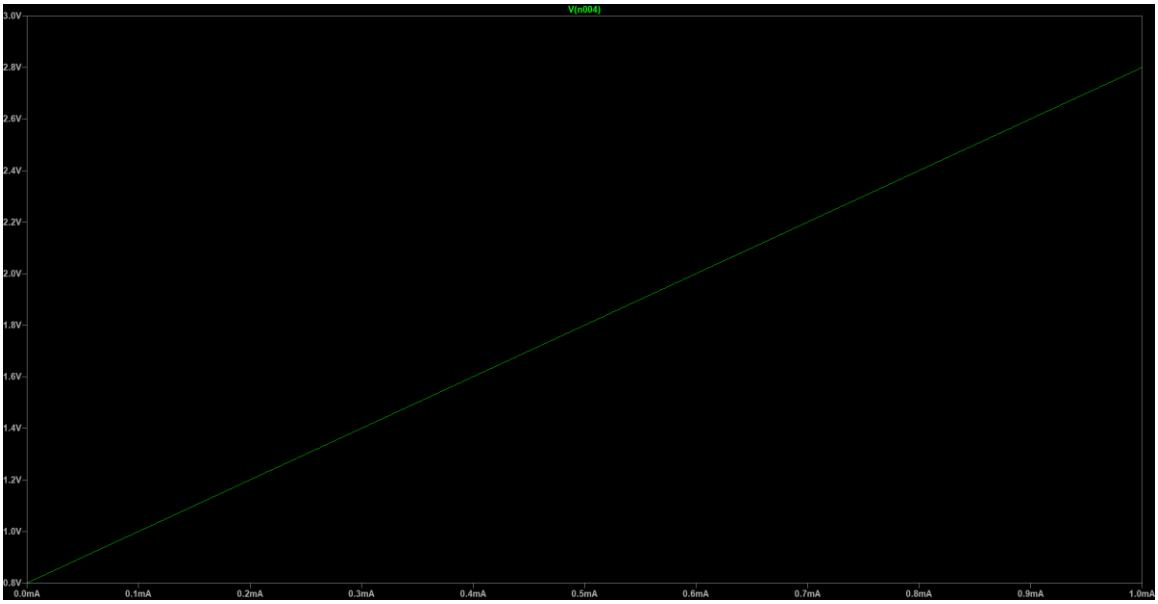


Figure 22: Current to Voltage transfer function analysis

As can be seen in the image above, the output voltage follows linearly with the input current. From 0 – 1 mA, it corresponds to a voltage range of about 0.8 – 2.8 V. This is within the source supply and is a reasonable voltage range to be read by the ESP32 microcontroller.

The next simulation was a frequency domain simulation to know how fast the photodiode can change its current and still get the required gain. The results of this are shown below.

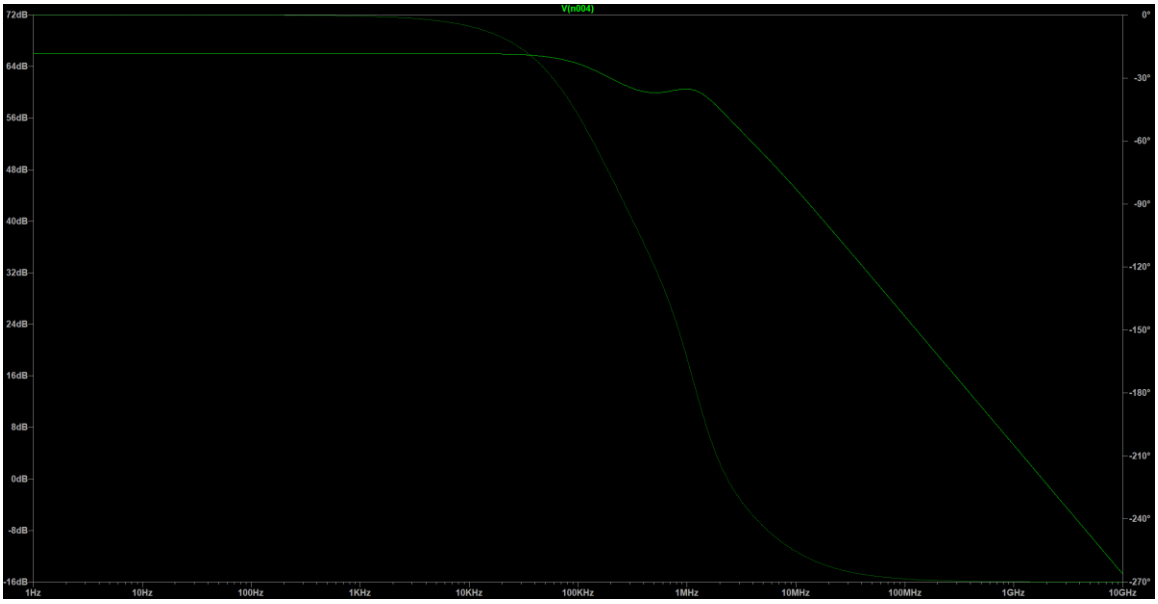


Figure 23: Frequency Response of the Photodiode-TIA circuit

As can be seen in the image above, the circuit sees 66 dB of voltage gain across a broad band of frequencies. The measured 3 dB point on this plot was found to be at approximately 157 kHz, as calculated. This is well above the frequency this circuit expects to function at, so the design will work for this application.

6.3: ESP32

The main PCB is responsible for housing thee ESP32, as well as interfacing with motors and the photodiode

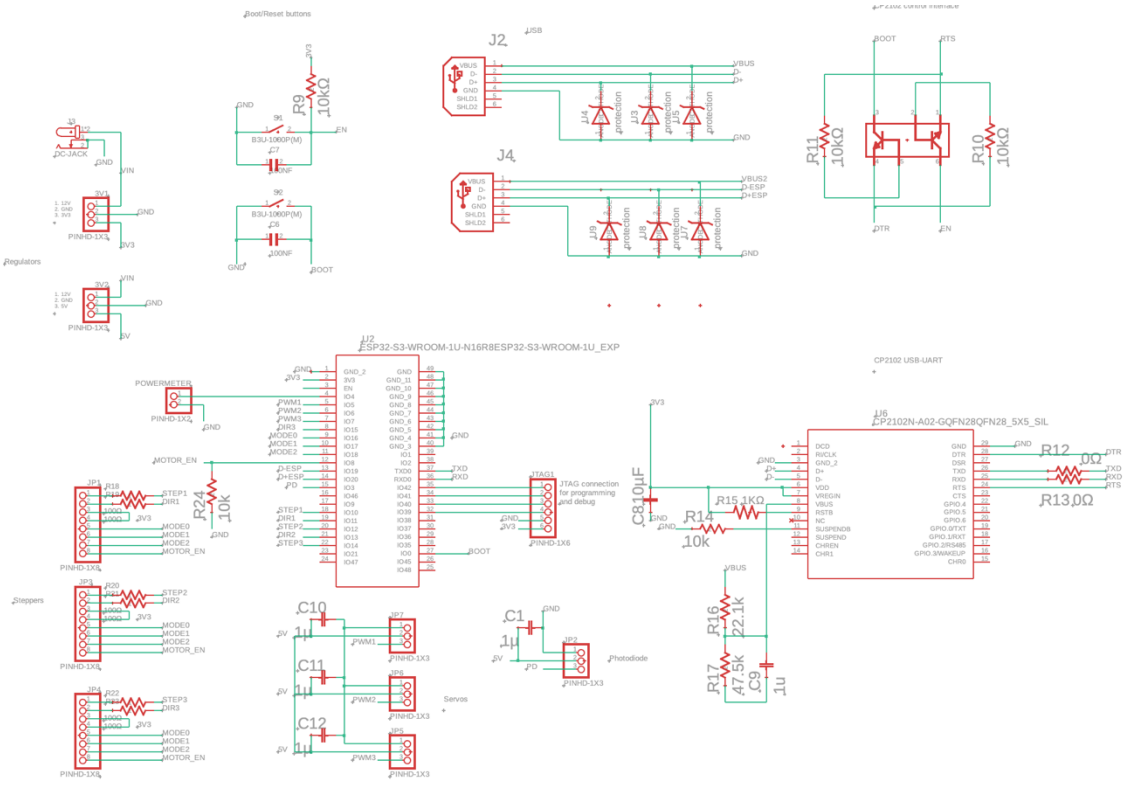
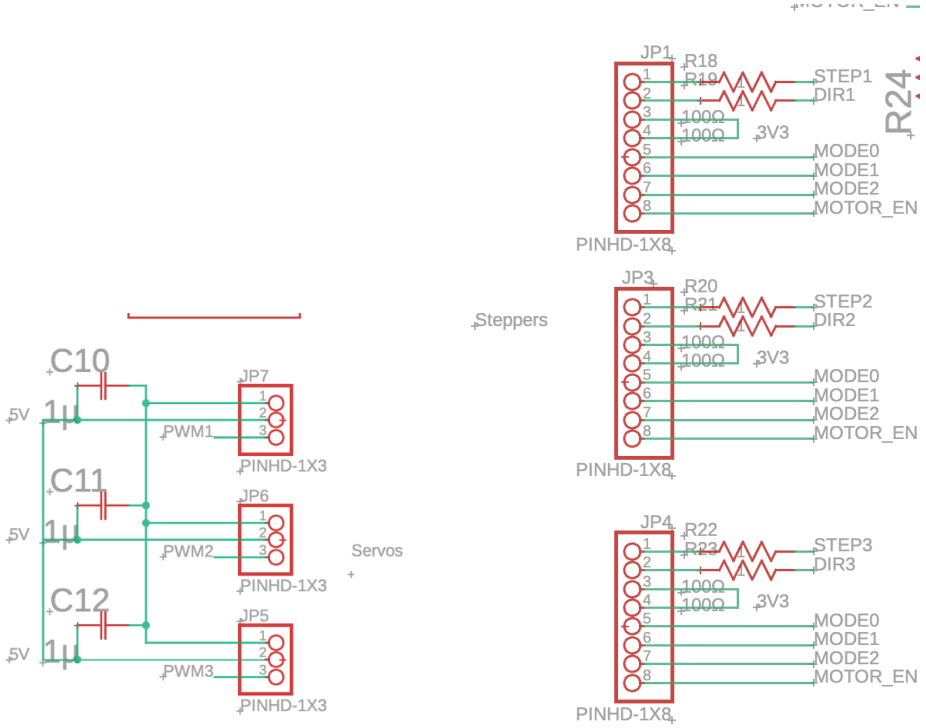
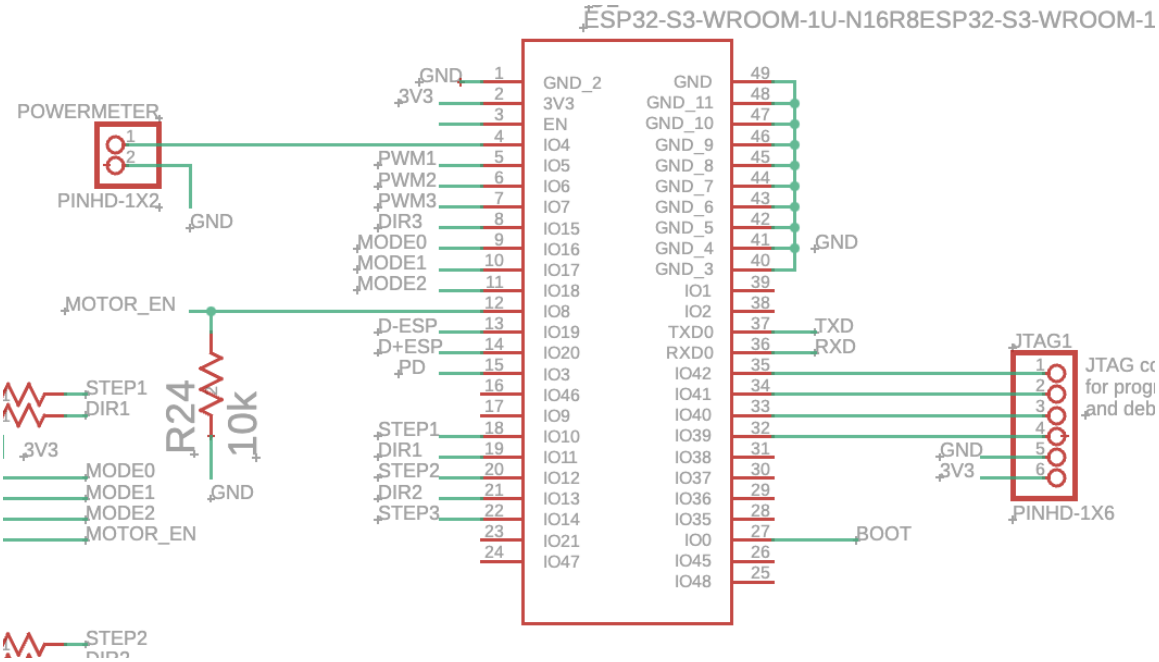


Figure 24: ESP32 and Main PCB Schematic

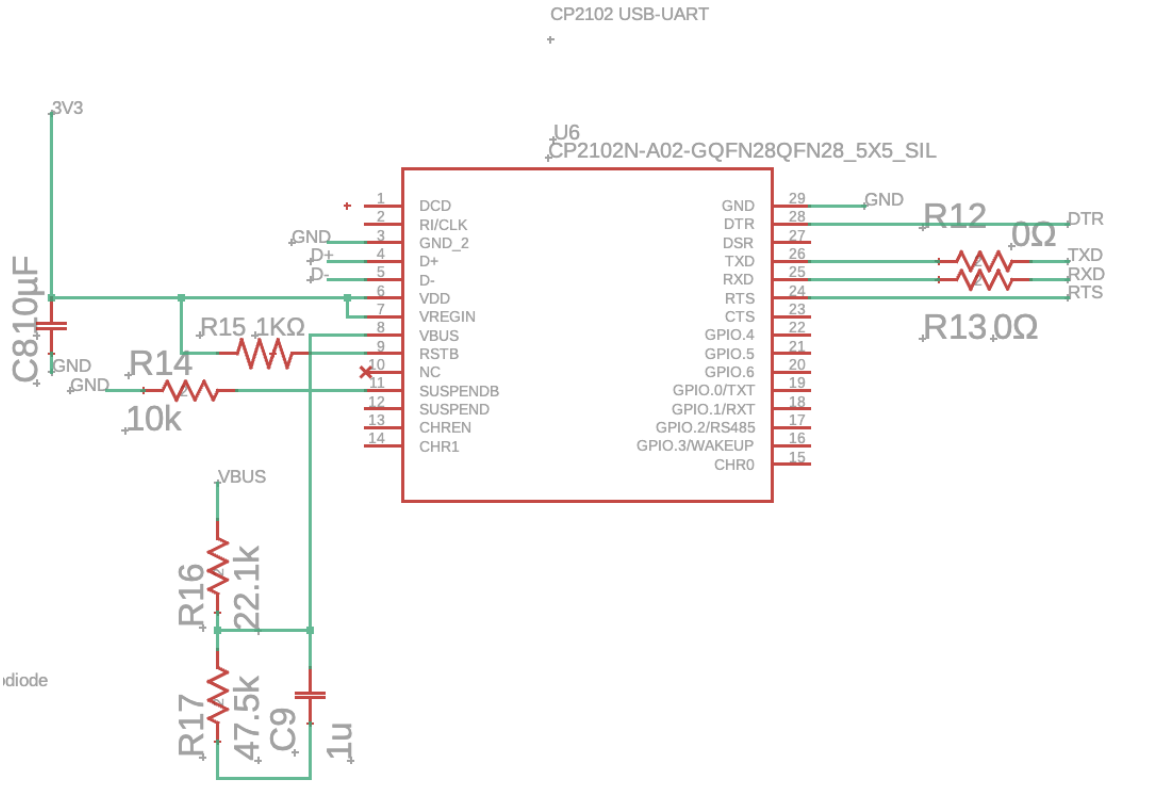
6.3.1: Motor Subsystems

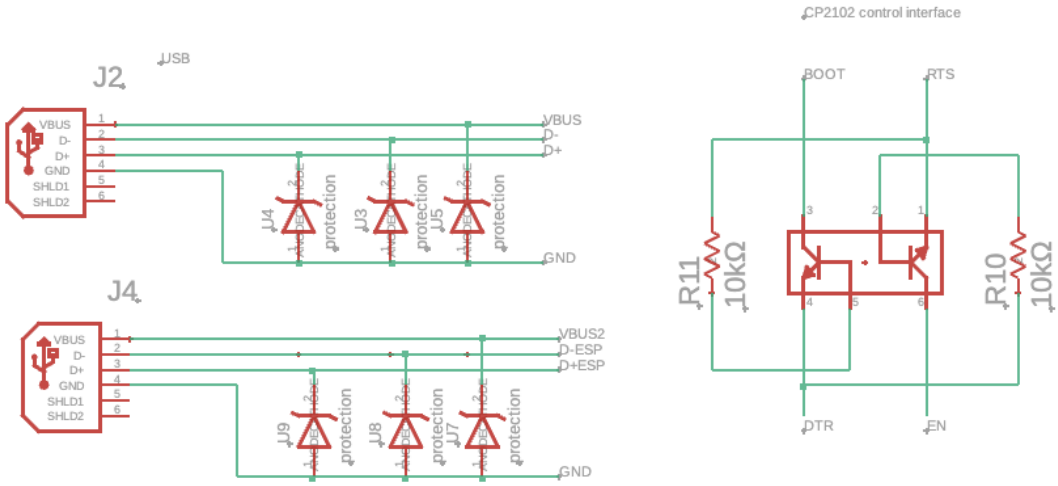


6.3.2: MCU

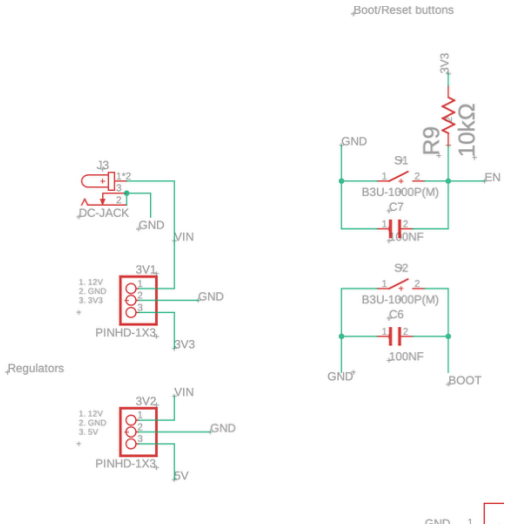


6.3.3: USB





6.3.4: Power and Control



6.4: Power Supply

Due to the importance of the power supply, a modular design is chosen. The regulator board will interface with the main PCB through header pins. The design of the 3.3V and 5V regulators are the exact same, and the LM2676-3.3V and LM2676-5.0V chips have the same footprint. This means that only one board design is needed for both regulators.

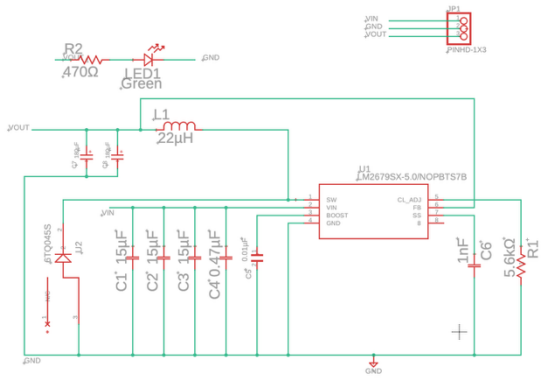
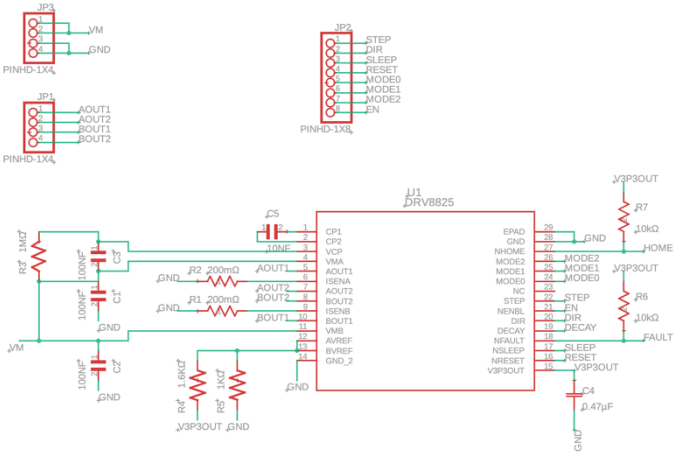


Figure 25: Modular Switching Regulator Board

These regulator boards will be connected to the main PCB to supply power to the MCU and all the peripherals. The modular design ensures changes can be made without affecting the main PCB. This same design is used for 3.3V and 5V since the chips are pin compatible.

6.3: Motor Driver Board

The motor driver board is responsible for interfacing between the motors and the MCU. The DRV8825 logic is powered by an internal 3.3V regulator, requiring only the motor power supply.



Chapter 7: Software Design

The software design of the project is designed to manage measurement control, data acquisition, and visualization for multiple laser parameters, including polarization, wavelength, divergence, and M^2. The system uses a dual-layer architecture:

1. **Embedded firmware** on the ESP32 microcontroller for real-time data collection, sensor management, and motor management.

2. **Graphical User Interface (GUI)** application for visualization of data, numerical computation, and user interaction.

The software design emphasizes modularity, real-time performance, and clear communication between the microcontroller and the host computer via a USB CDC serial interface.

7.1: Software Architecture Overview

The overall software system is designed as a client-server model, where the ESP32 acts as a deterministic data server, and the desktop GUI functions as a flexible analytical client. Communication occurs through a USB virtual COM port (CDC) that exchanges structured packets containing timestamps, sensor readings, and control instructions.

7.1.1 Embedded Firmware Layer

The embedded firmware handles all time-critical operations such as ADC sampling, PWM control of motors, and synchronization of the rotating polarizer or translation stages. The firmware is organized into independent tasks under FreeRTOS, each with defined priorities to avoid resource contention. Data is stored temporarily in circular buffers to prevent overflow before transmission to the PC.

7.1.2 Host Application Layer

The Python GUI serves as the user's entry point, providing an intuitive interface for configuring experiments and visualizing results in real-time. The GUI leverages object-oriented design, allowing measurement classes like PolarizationPlan or M^2Plan to share base functionality for polling, buffering, and timing. This structure promotes code reuse and makes the system easier to maintain.

The GUI and firmware communicate asynchronously; the GUI can queue user commands while the MCU streams measurement data concurrently, ensuring smooth user experience even during long data acquisitions.

In the figure below, we see this software design structure shown graphically

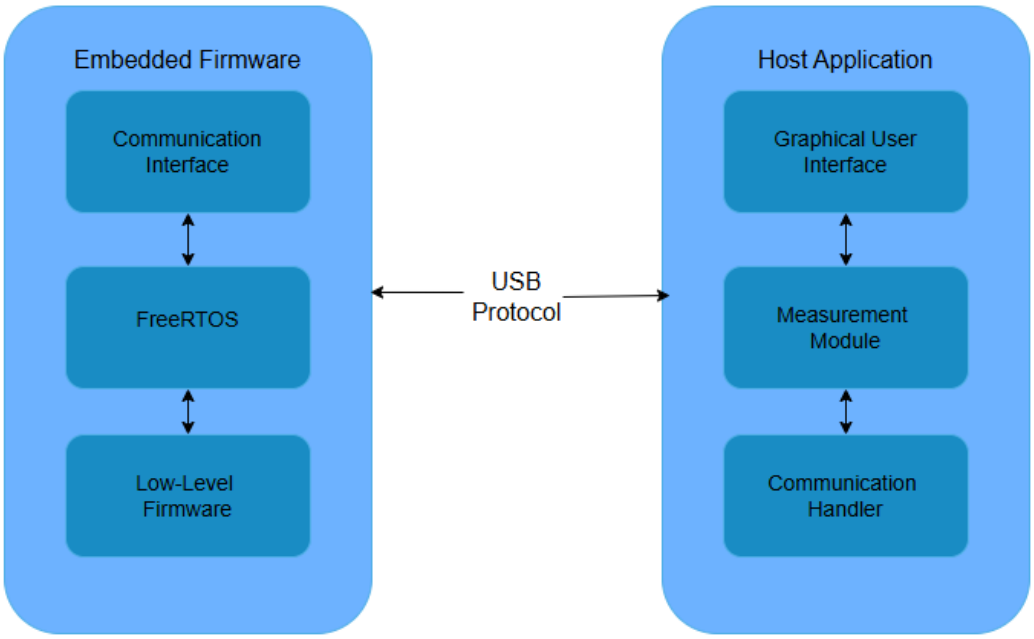


Figure 26: Software Architecture Flowchart

7.1.3 System Operation

Upon application launch, the software initializes the GUI components, communication handlers, and data structures. The system attempts to establish a USB-CDC link with the ESP32 microcontroller. If the connection succeeds, the GUI enters Hardware Mode; otherwise, it defaults to Simulator Mode for offline testing. After initialization, the user selects one of the measurement operations (Polarization, Divergence, M^2, or Wavelength), each of which triggers a separate computation and visualization pipeline. The figure below illustrates the initialization process and mode selection:

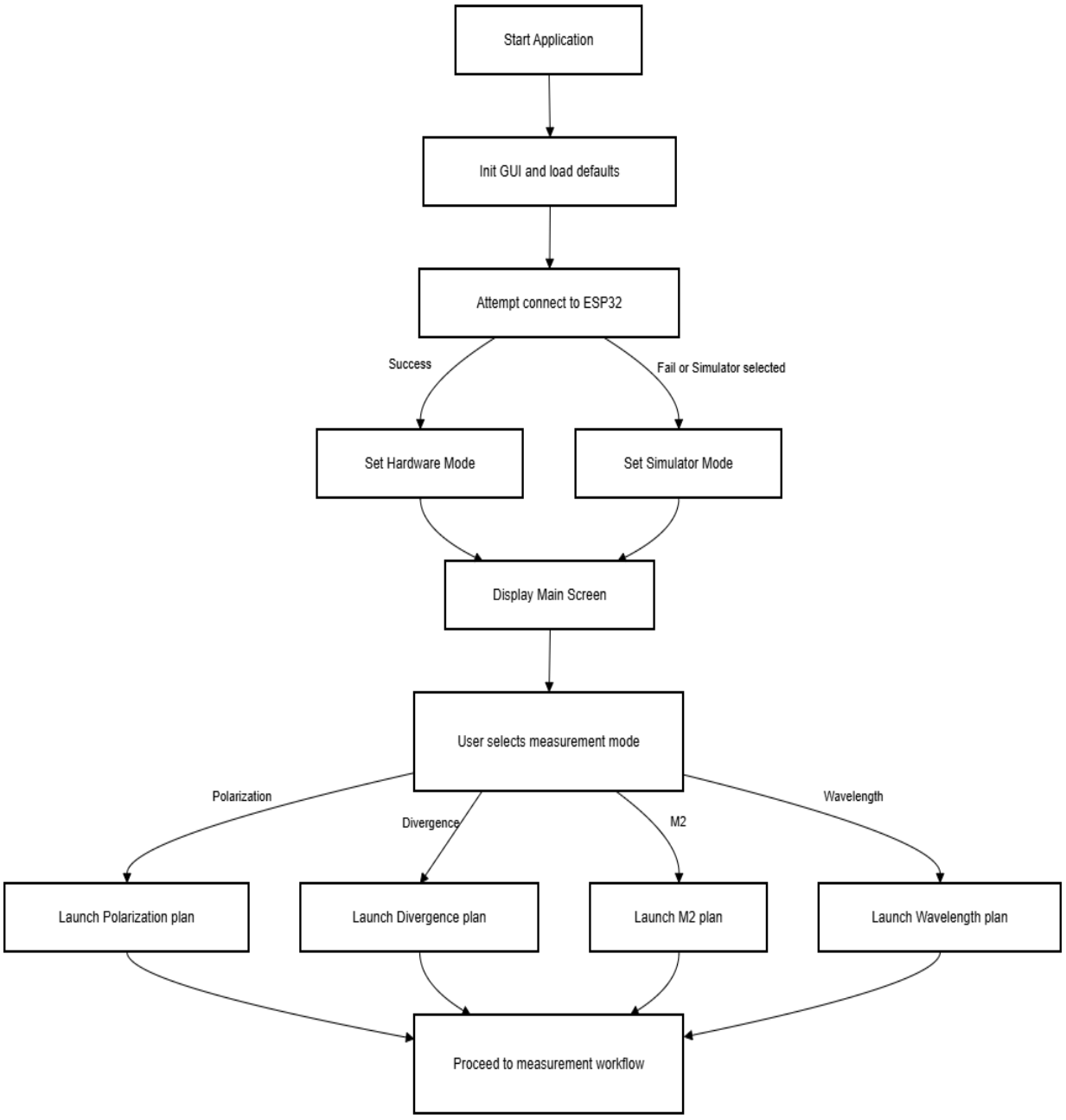


Figure 27: Initialization Process and Mode Selection

7.2 Subsystem Design

7.2.1 Embedded Firmware Subsystem

The embedded firmware is implemented on the ESP32 microcontroller using PlatformIO and written in the C programming language. The software architecture follows a modular design paradigm, in which each functional block-sensor acquisition, motor control, and communication handling- is encapsulated into individual source files (sensor.c, motor.c,

usb_comms.c). This modularization promotes code readability, ease of debugging, and isolated testing of subsystems without affecting the rest of the system.

Data acquisition relies on the 12-bit ADC peripheral of the ESP32 for analog photodiode signals, while I2C and SPI interfaces are utilized for digital sensors such as the polarization detector. The firmware ensures timing determinism through hardware timer interrupts, with all tasks scheduled under the FreeRTOS real-time operating system. Each task executes periodically according to its assigned priority, guaranteeing synchronized measurement cycles across sensors.

Interrupts are employed selectively to handle high-priority events such ADC conversion completion, motor encoder ticks, or UART reception. Less time-critical task-including data filtering and status logging- are executed within FreeRTOS threads, which simplifies timing management and prevents jitter. A watchdog timer supervises firmware operation, automatically resetting the microcontroller in the event of task starvation or deadlock. This enhances the system's fault tolerance, a critical requirement for long-duration measurement sessions.

The embedded software exposes a command-response interface over USB. Commands are serialized in a lightweight JSON format and transmitted to the host computer. This design allows flexible message parsing while maintaining human readability, making debugging possible through terminal utilities such as PuTTY or RealTerm.

7.2.2 GUI Application Subsystem

The Graphical User Interface (GUI) seen in Fig. 28 is implemented using Python and PySide6 (Qt for Python), adopting the Model-View-Controller (MVC) architecture to ensure software scalability and maintainability.

- The Model component manages data structures, buffers, and mathematical computations, including Fourier transforms, Gaussian beam fittingm and Stokes parameter estimation.
- The View is composed of Qt widgets and Matplotlib plots that render real-time visualization of sensor data, measurement progressm and computed optical parameters.
- The Controller mediates between user inputs and hardware actions by translating button presses, slider adjustments, or simulation toggles into system-level commands.

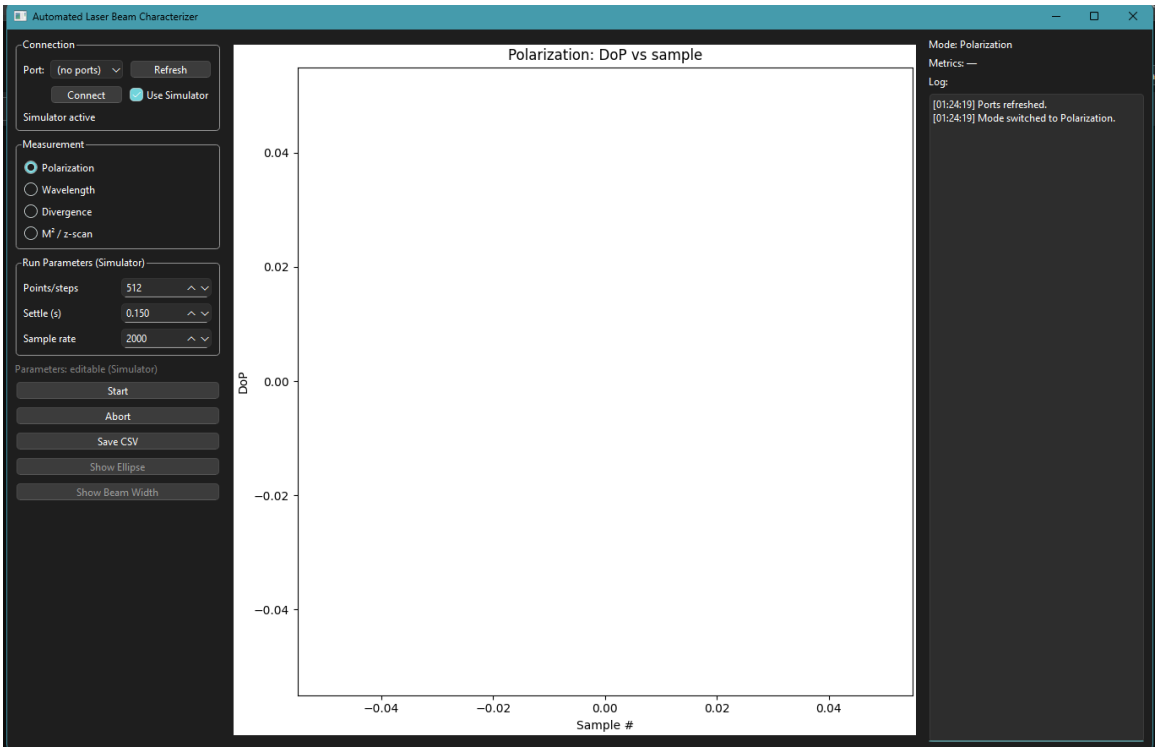


Figure 28: Main GUI window

This separation ensures that any modification in the GUI appearance does not affect the computational or communication back-end, aligning with robust software engineering principles. The GUI supports two operational modes:

1. **Simulator Mode**: Synthetic data generated by simulator.py emulate realistic sensor responses. This feature enables algorithm verification and interface testing prior to hardware availability, significantly accelerating early-stage development.
2. **Hardware Mode**: In this mode, real-time data streams from the ESP32 microcontroller via the USB link are parsed and processed identically to simulated data, preserving consistent visualization and computation workflows.

The GUI further integrates asynchronous QTimer loops to maintain live data plotting without blocking the main event loop. Data updates are buffered and displayed at a user-selectable refresh rate, ensuring smooth visualization via a SerialLink class, encapsulating USB read/write functions and error handling to maintain consistent frame integrity.

7.2.3 Interfacing Between Subsystems

The integration of the embedded firmware and host GUI is accomplished through a USB CDC (Communication Device Class) interface, which emulates a serial communication port. Data packets conform to a custom protocol, where each message begins with a header byte, followed by a payload encoded in JSON, and ends with a checksum for transmission integrity.

7.3 Algorithm and Computation Design

This section explains, in prose, how the host **GUI** consumes data produced by the ESP32 firmware and turns those streams into the metrics and plots used throughout the application. The emphasis is on the *software behaviors*—buffering, validation, transformations, and user-facing feedback—rather than on the physics already detailed in Chapter 3. Across all modes, the GUI adheres to a consistent pattern: the Controller receives framed USB messages, the Model maintains streaming state and computes metrics incrementally, and the View renders plots and updates the Metrics panel. The same logic runs against the **Simulator** so that algorithms can be exercised without hardware.

7.3.1 Polarization (Rotating-QWP)

For polarization runs, the MCU sends a sequence of angle–intensity samples as the quarter-wave plate rotates. The GUI treats this stream as an ordered experiment: each new sample is appended to a growing buffer, re-sequenced if packets arrive out of order, and immediately fed to a lightweight estimator. Rather than wait for the full sweep, the Model maintains running aggregates that correspond to the sinusoidal components implied by the rotating-QWP formulation (§3.1.2). With each packet, those aggregates are refreshed, and the current best estimate of the Stokes vector becomes available.

Two design choices make this interaction responsive and robust. First, the estimator is **streaming**: it only updates small accumulators, so the UI can refresh multiple times per second without recomputing from scratch. Second, the Model applies **guardrails** based on data sufficiency and quality. The *Show Ellipse* button and the Poincaré sphere view are disabled until a minimum number of uniformly spaced angles has been observed; obvious outliers (e.g., momentary photodiode glitches) are down-weighted using median-based heuristics. Once the Stokes vector stabilizes, the GUI presents (i) total and partial degrees of polarization, (ii) ellipse orientation and ellipticity, and (iii) a continuously updated ellipse plot overlaid on the incoming intensity trace. This immediate feedback helps users recognize misalignment or depolarization while the sweep is still underway. In Simulator Mode the exact same packet format is generated, ensuring the computation path is identical to hardware. In Figure 29 below, you can see the polarization plan in action:

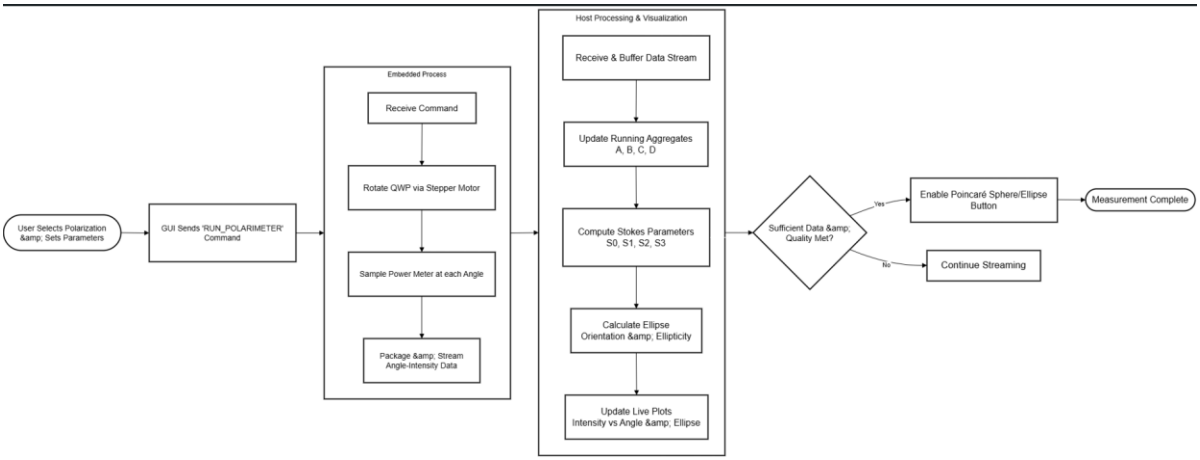


Figure 29: Polarization Measurement Software Design

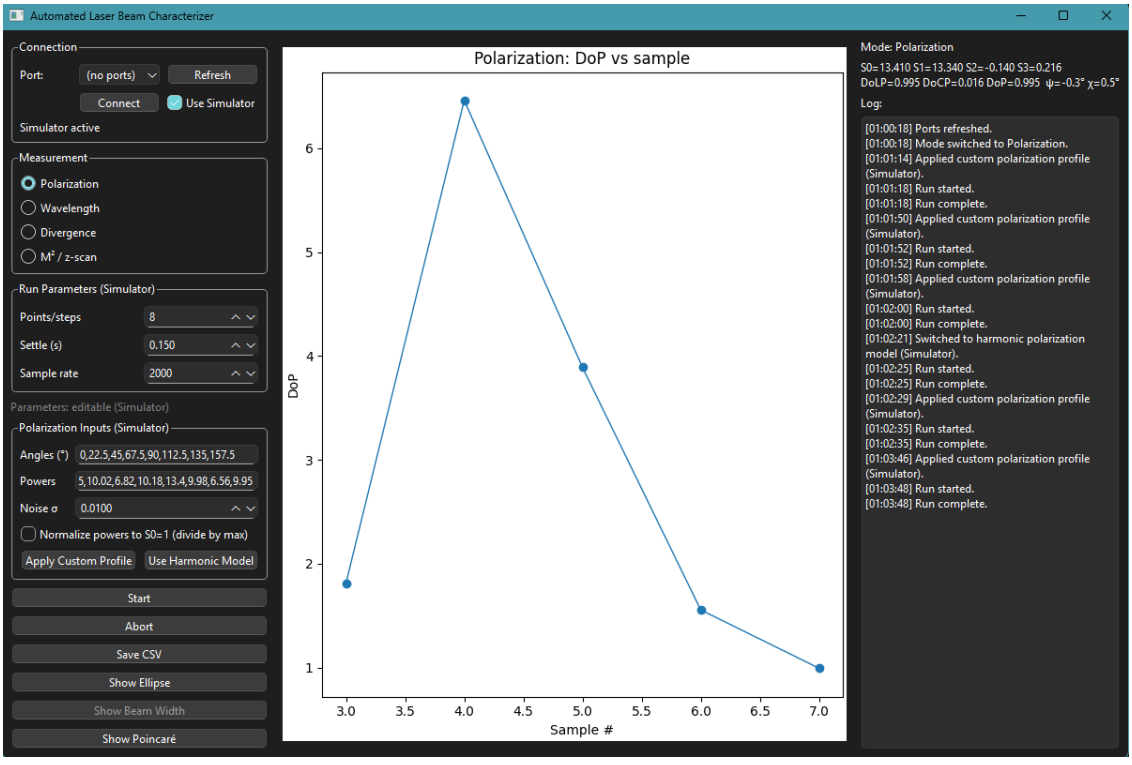


Figure 30: Screenshot of GUI running Polarization Plan in Simulator Mode

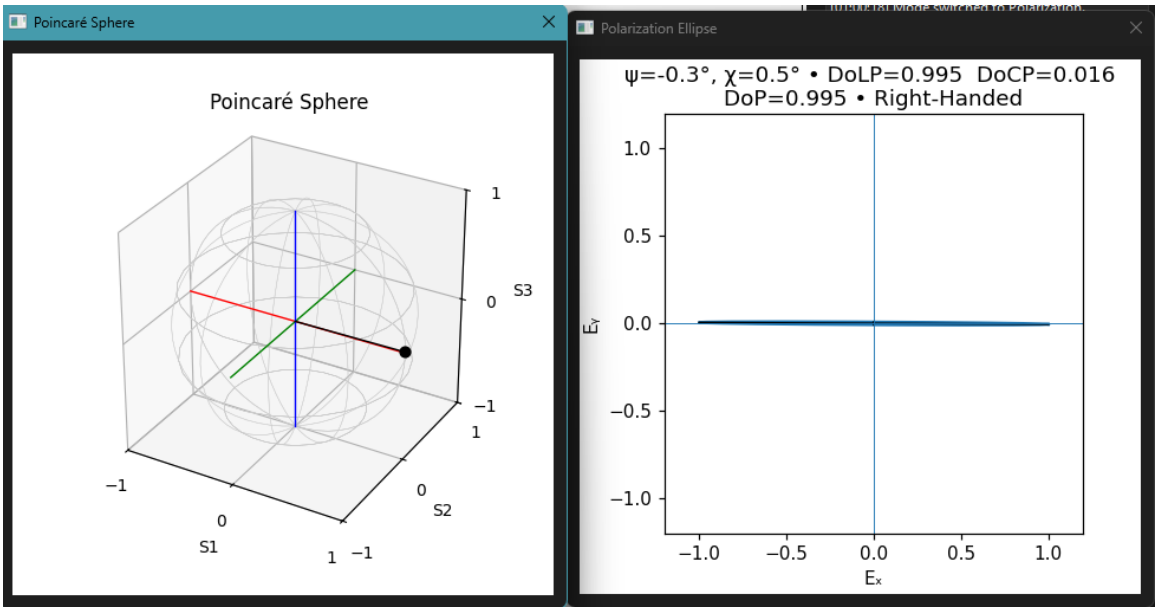


Figure 31: Screenshot of Show Ellipse/Show Poincare Sphere

7.3.2 Divergence (Knife-Edge/Scan)

In divergence mode, the firmware reports a beam-width estimate at each stage position. The GUI converts the incoming index or timestamp into physical distance using the provided step scale and maintains a live scatter of width versus position. Because divergence is most stable away from focus, the software permits users to select a fitting window and then performs a simple trend analysis over that region. The emphasis here is on **explainability** rather than curve-fitting sophistication: the View overlays a straight-line trend with confidence bands and reports the corresponding angular divergence in milliradians.

Several small design features make the output trustworthy. The Model delays publishing a final divergence until the stage has traversed a *meaningful* span; otherwise the slope is flagged as unreliable. If the device or environment produces occasional bad width readings—common with knife-edge transitions—the GUI's estimator falls back to a robust slope (Theil–Sen) that is tolerant to outliers. When available, contextual metadata such as the operating wavelength and any assumed M^2 are displayed alongside the angle so users can immediately see the implied diffraction-limited waist for the present configuration (with clear labeling to distinguish measured from inferred values). Figure 32 shows the divergence plan process while Figure 33 illustrates the live scatter, the trend overlay, and the associated numerical callouts.

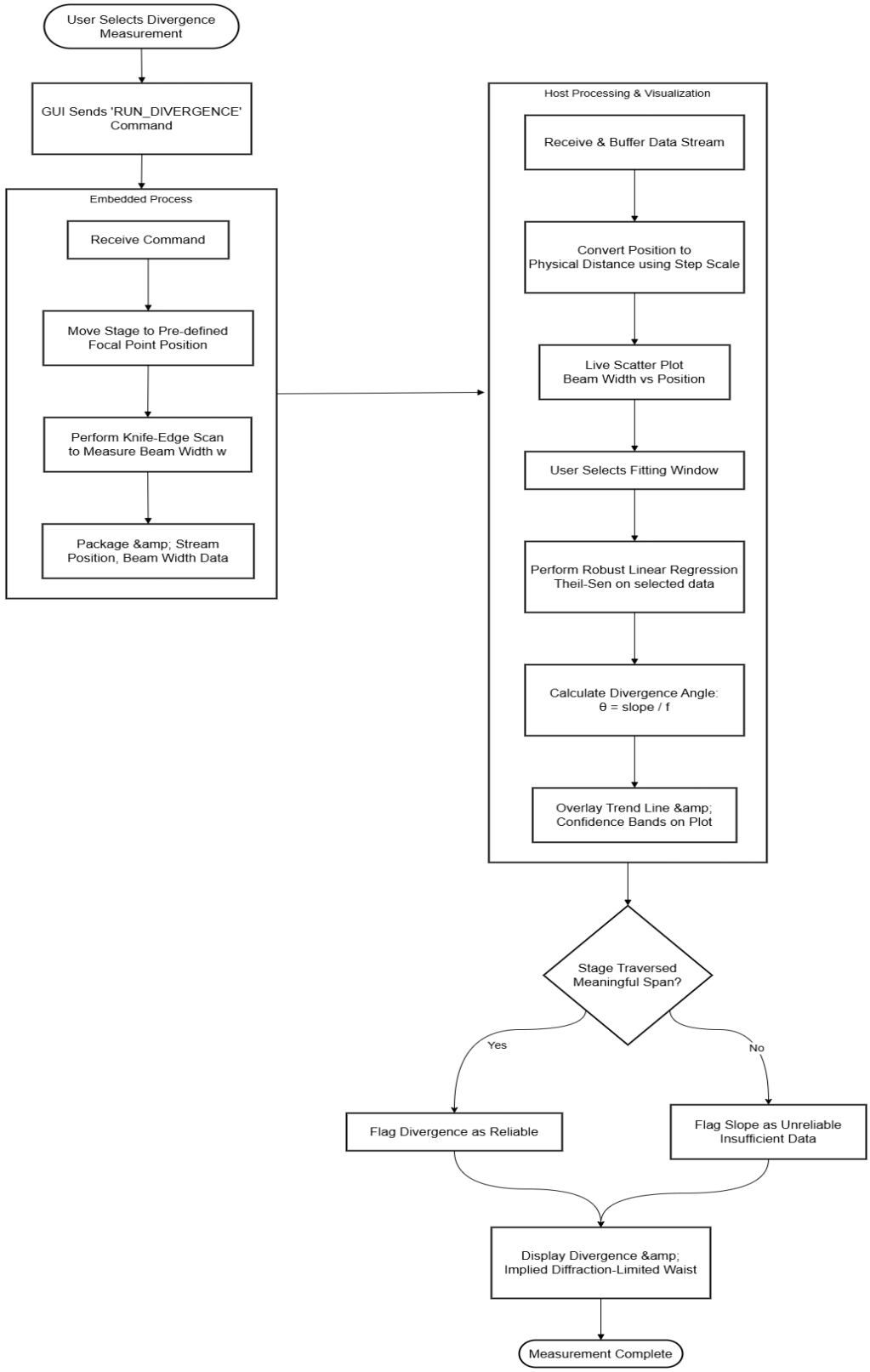


Figure 32: Divergence Measurement Software Design

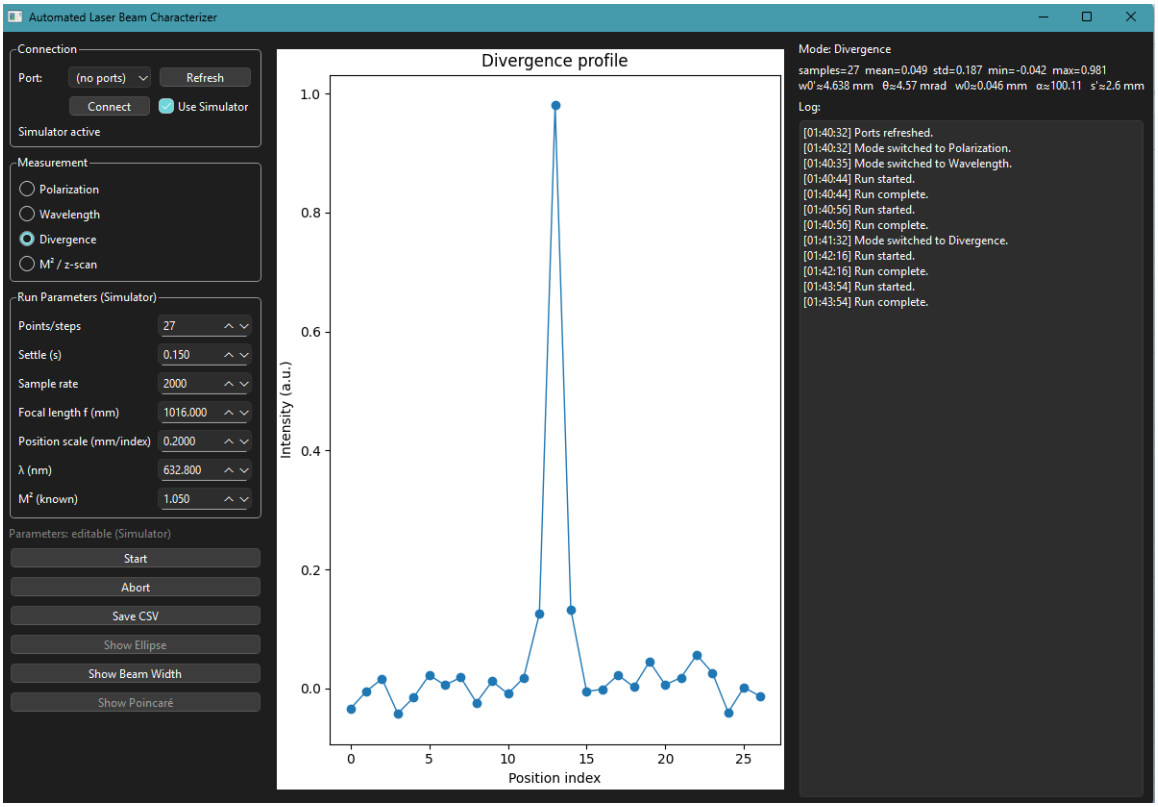


Figure 33: Screenshot of GUI running Divergence Plan in Simulator Mode

7.3.3 Beam Quality M^2 (ISO 11146 Method 2)

The M^2 workflow mirrors the structure of the standard: the firmware or simulator delivers a sequence of beam radii at known axial positions, intentionally spanning points near and far from the waist. On the software side, the Model squares the radii and performs a quadratic regression against position. The purpose is less to produce a perfect global fit and more to extract the *three interpretable quantities* the user cares about: waist size, waist location, and overall propagation quality.

Because these quantities are derived from a fit, the GUI spends as much real estate on **fit diagnostics** as on the headline numbers. The main plot shows the measured $w^2(z)$ samples and the current parabola; a secondary residual panel can be toggled to reveal systematic errors (e.g., astigmatism causing different waists for X and Y). The Metrics pane prominently displays a goodness-of-fit score, and the application warns when the sampling range appears insufficient to satisfy the "inside/outside Rayleigh range" guidance (see §3.2). Users can therefore judge result quality at a glance, and—as important—know *why* an estimate has been labeled unstable. Figure 35 shows the M^2 measurement process while Figure 36 presents a typical sequence as data arrives, and the estimates converge.

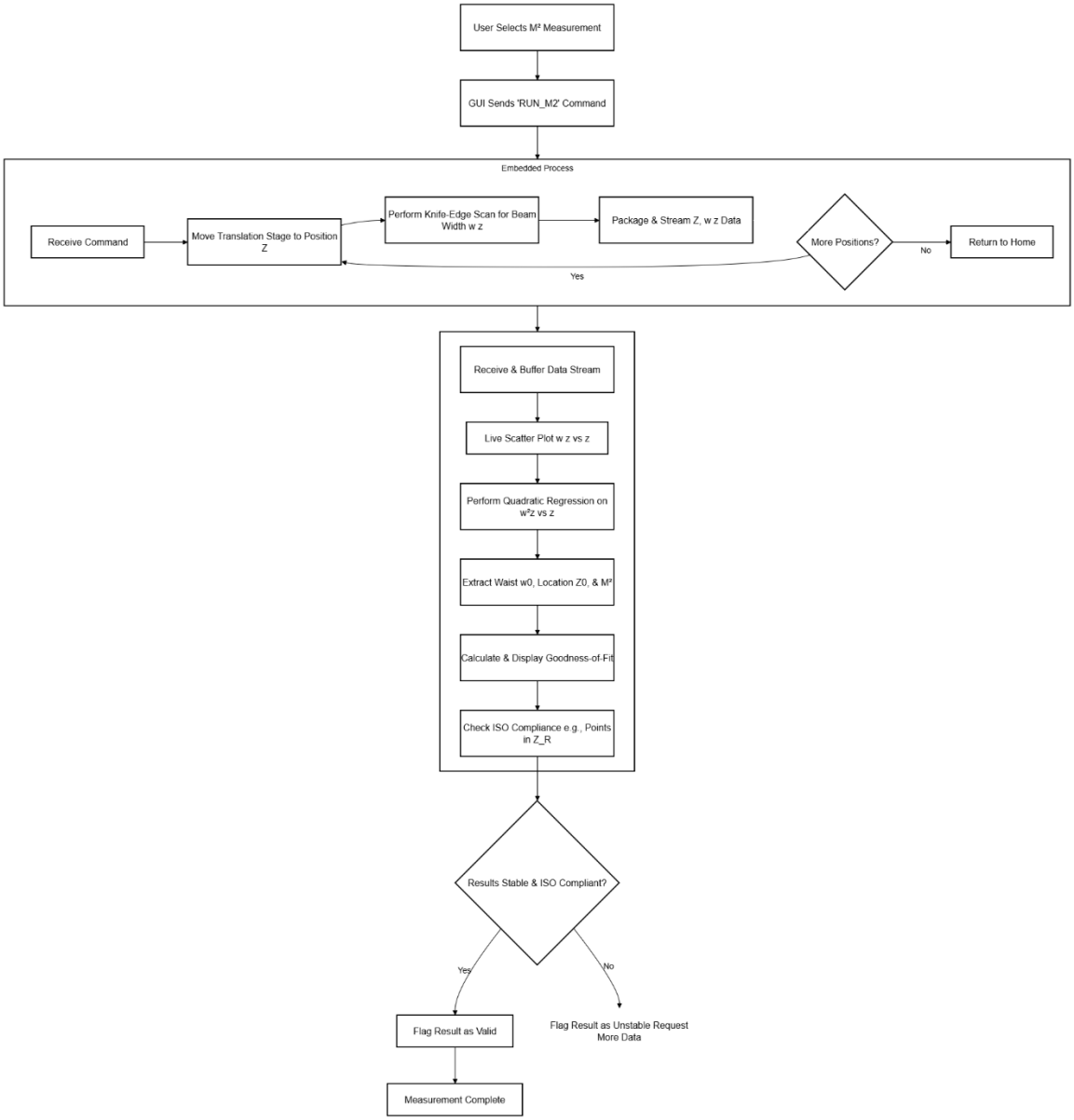


Figure 34: M2 Measurement Software Design

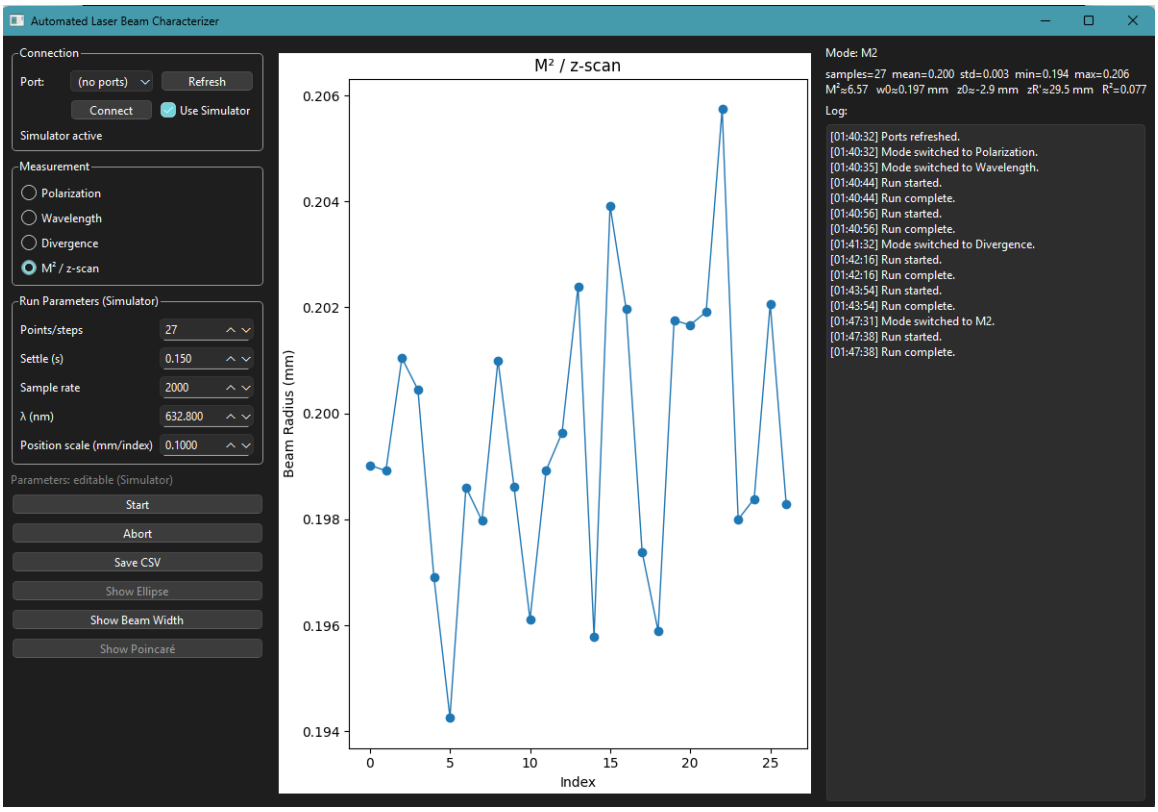


Figure 35: Screenshot of GUI running M2 Plan in Simulator Mode

7.3.4 Wavelength (Michelson, FFT-Assisted)

For wavelength, the ESP32 streams a photodiode time series while translating the interferometer mirror at a known velocity. The GUI displays the live fringe signal and, in parallel, analyzes short overlapping windows of the record using an FFT. The dominant spectral line is tracked with sub-bin interpolation and converted to a wavelength using the commanded mirror speed. This windowed approach makes the estimate remarkably **responsive**: the number stabilizes after only a few hundred samples and updates smoothly as the mirror moves.

Because interferograms in real hardware can drift or include harmonics (e.g., stage microsteps), the software includes **two safety valves**. First, the FFT search can be constrained to an expected frequency band derived from the stage velocity, so spurious peaks are ignored. Second, a **fringe-counting fallback** remains available: when the signal-to-noise ratio is high and the motion is slow, simple zero-crossing counts over a measured displacement corroborate the FFT result. The GUI shows both values side-by-side; when they agree within tolerance, the number is labeled "validated," otherwise it is marked as "estimate—check alignment." Figure 36 shows the wavelength measurement process while Figure 37 shows the time trace, the spectrum with a peak cursor, and the corresponding wavelength readout.

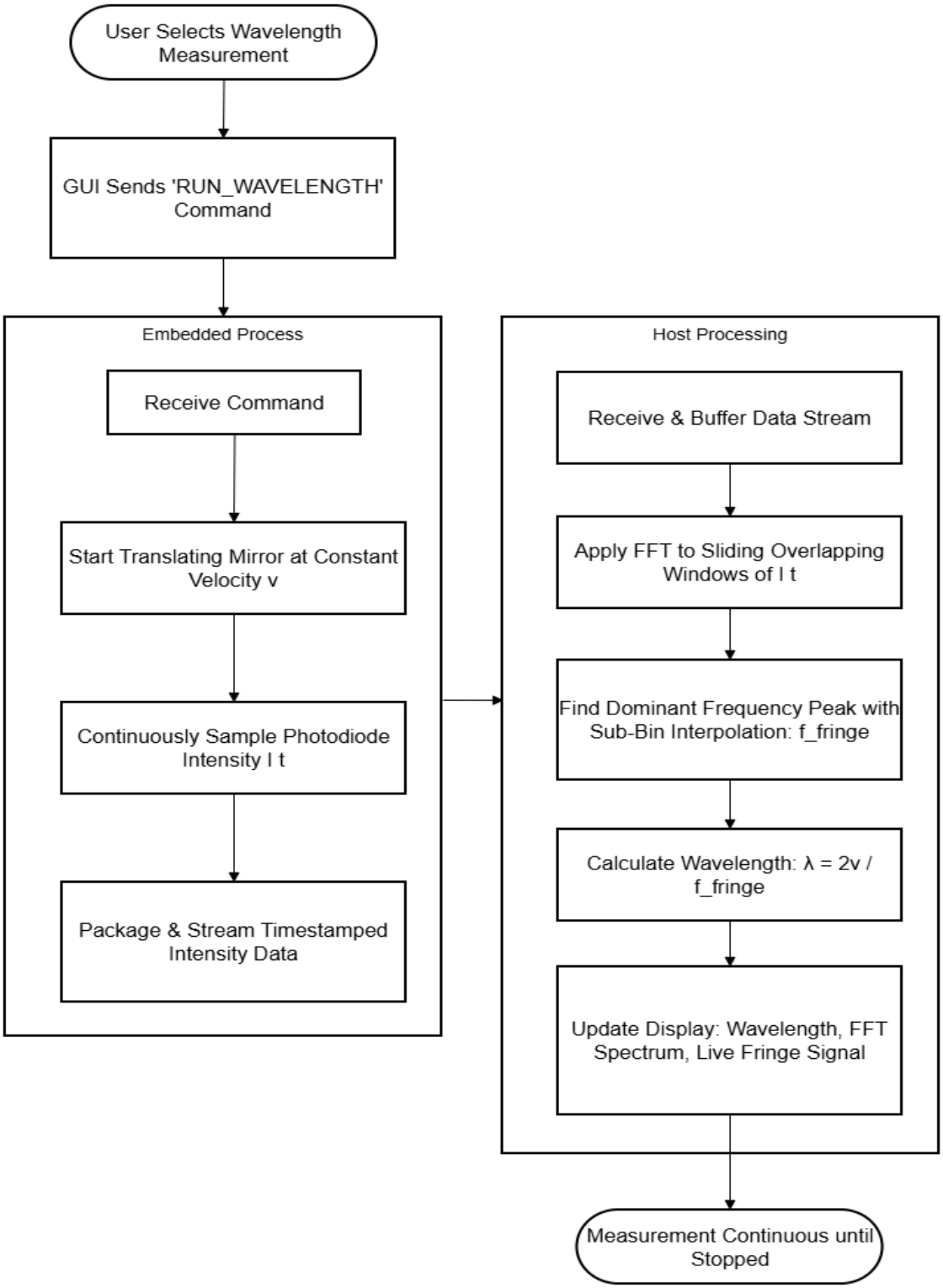


Figure 36: Divergence Measurement Software Design

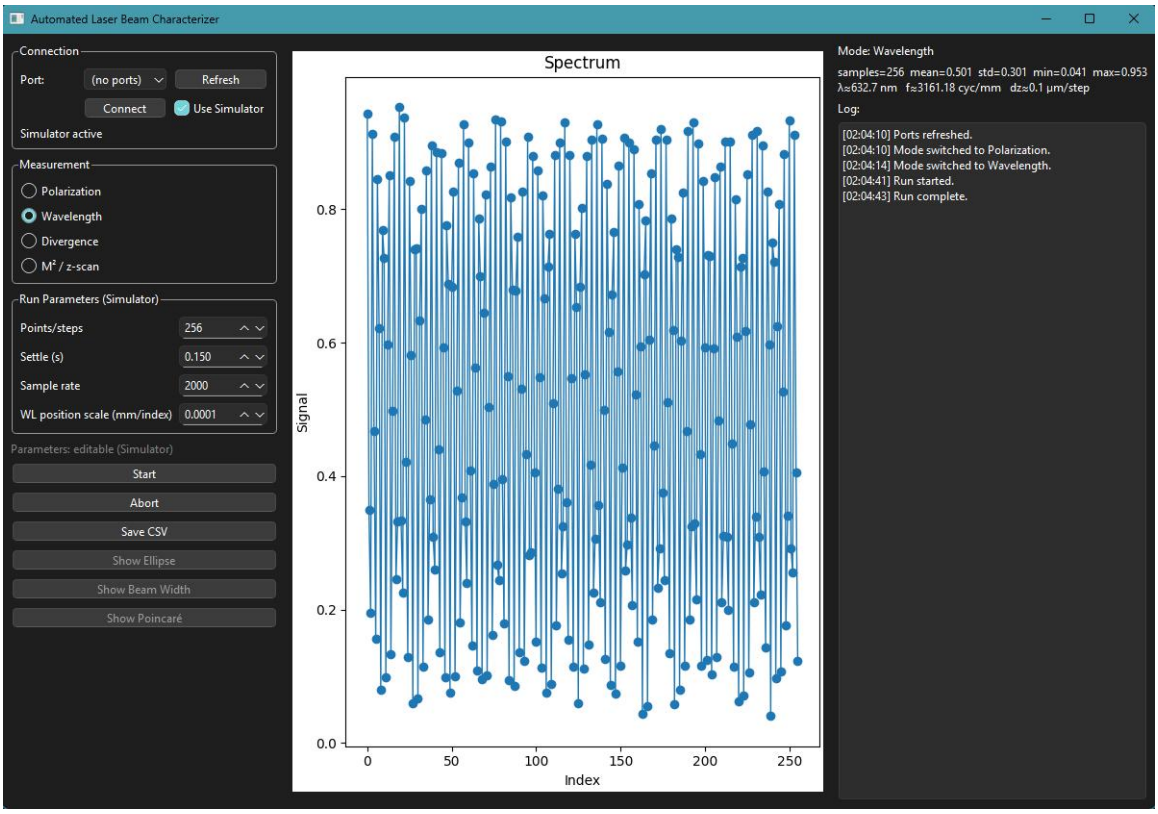


Figure 37: Screenshot of GUI running Wavelength Plan in Simulator Mode

7.3.5 Cross-Cutting Software Practices

Across all modes the GUI implements a handful of practices that keep the experience predictable:

- **Streaming numerics and low latency.** All estimators update constant-size accumulators so plots and metrics remain fluid even on modest laptops.
- **Strict unit handling.** The Model stores canonical units (mm, rad, seconds) and converts only at the View. Meta packets are validated before any scale changes are applied, preventing "unit jumps" mid-run.
- **Quality gates and progressive disclosure.** Buttons such as *Show Ellipse* and *Show Beam Width* enable only after their prerequisite metrics are stable. When they are not, the Metrics panel explains what is missing (e.g., "need ≥8 angles" or "increase z-span for stable M^2").
- **Simulator parity.** The simulator emits the same packet schema as hardware, which lets the exact computation paths, plots, and guardrails be exercised well before the first bench test.

7.5 Integration and Testing

The integration and testing phase serves as the final validation of the complete system, ensuring that the embedded firmware, communication protocols, and graphical user interface (GUI) operate cohesively under both simulated and hardware conditions. This phase verifies that the laser measurement subsystems—polarization, divergence, M^2, and wavelength—produce consistent and accurate results when controlled through the GUI and measured by the ESP32 microcontroller.

7.5.1 Simulation vs. Hardware Mode

Integration testing began with the **Simulator Mode** enabled in the GUI. In this configuration, the backend modules (e.g., polarization.py, divergence.py, m2.py, wavelength.py) generate synthetic datasets that mimic realistic signal and noise characteristics of each subsystem. This allowed the development team to validate mathematical models, plotting accuracy, and metric convergence without the need for physical hardware. The simulator ensured that core computational modules—such as FFT-based wavelength extraction and quadratic fitting for beam quality—were functioning correctly and producing expected values under controlled conditions.

Once verified in simulation, the system can be transitioned to **Hardware Mode**, where the GUI communicates with the ESP32 microcontroller via USB using the same packet schema. The identical command and response interface ensured that switching between modes required no code modifications, thereby isolating integration challenges to physical data acquisition rather than logic discrepancies. This dual-mode design greatly accelerated debugging and confirmed that computation algorithms were consistent across both operational environments.

7.5.2 USB Protocol Verification

The **communication interface** between the GUI and the ESP32 was implemented over the USB CDC (Communications Device Class) protocol. During integration testing, a logic analyzer and serial debugging tools were used to verify the timing, framing, and reliability of packet exchange.

Each message follows a JSON-based structure containing event identifiers (e.g., "ev":"SAMPLE", "ev":"STOKES", "ev":"META") and associated data fields. The GUI's backend continuously listens for these messages using non-blocking I/O to ensure smooth graphical updates while maintaining deterministic data logging. Protocol verification focused on:

- **Packet Integrity:** Ensuring all messages received were valid JSON and corresponded to expected schema definitions.
- **Command–Response Synchronization:** Confirming that each GUI command, such as "RUN_POLARIMETER" or "RUN_M2", triggered the correct MCU state transitions.
- **Error Handling:** Simulating malformed packets and unexpected disconnects to confirm that the GUI properly reported communication errors without crashing.

The STM32 firmware includes a lightweight CRC check and acknowledgment flag, which were verified to prevent false triggers or data loss during sustained acquisition runs. Figure 38 will illustrate this communication handshake, showing command initiation from the GUI, acknowledgment from the ESP32, and the continuous stream of measurement packets.

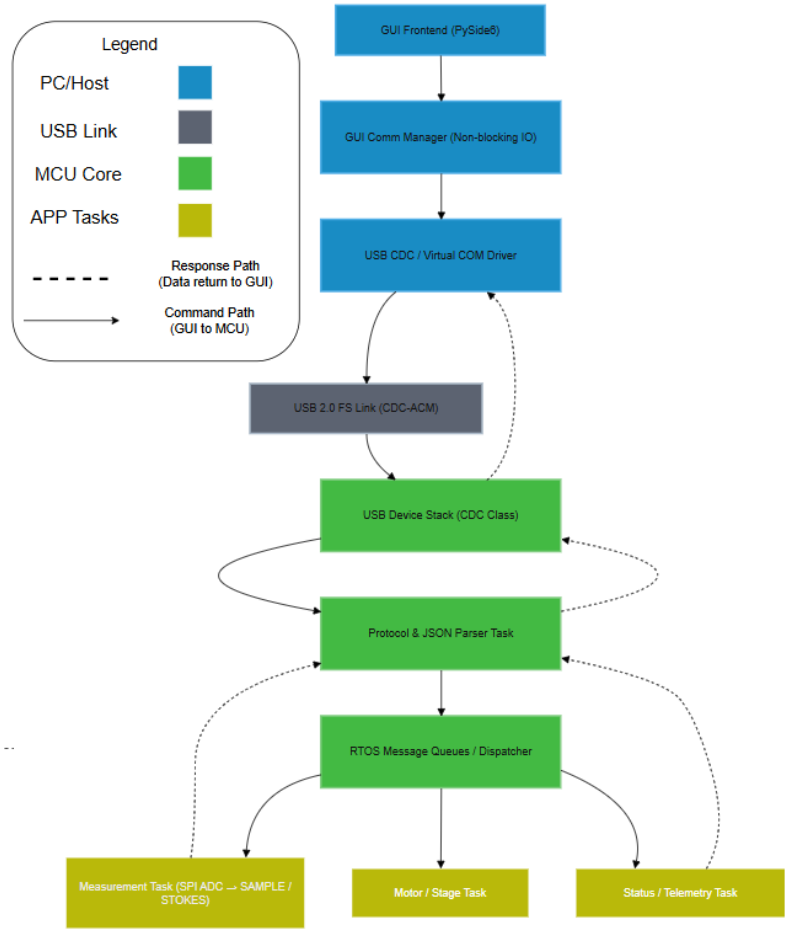


Figure 38: USB CDC Communication Software Architecture

7.5.3 Timing and Performance Validation

Since timing accuracy is critical to all optical measurements, especially for interferometric and scanning operations, extensive timing validation was performed on both software and hardware layers.

In **Simulation Mode**, timestamps generated by the GUI were compared to expected iteration rates to verify that plotting and computation threads maintained synchronization with data generation intervals. In **Hardware Mode**, timing validation included measuring the end-to-end latency between MCU data capture and GUI visualization. For the ESP32 microcontroller, the data acquisition will need to be verified to operate at sample rates exceeding 10 kHz for photodiode signals without packet loss. Performance profiling demonstrated that metric updates (e.g., Stokes parameters, M^2 fitting results) appeared on the GUI within an average latency of less than 40 ms from the time of acquisition—well within the bounds necessary for real-time operation. These benchmarks were recorded using timestamped logs and visual oscilloscope verification of synchronization pulses sent from the MCU. Below you will find Figure 39 showing the timing diagram for end-to-end data flow.

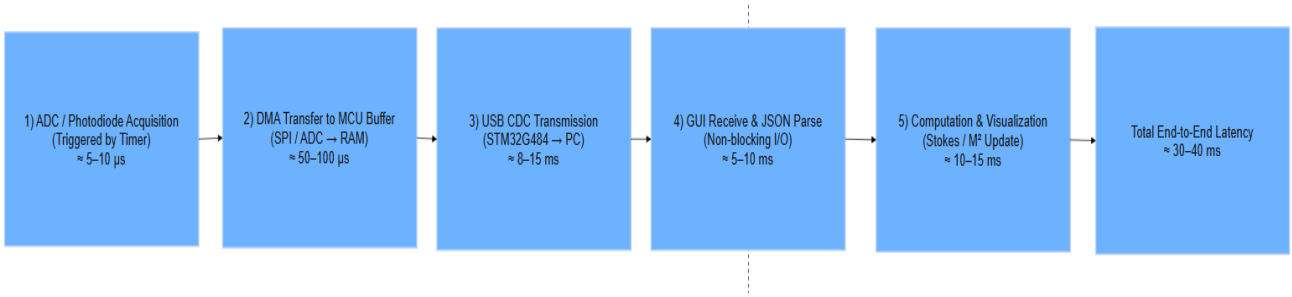


Figure 39: Timing Diagram for End-to-End Data Flow

7.5.4 Watchdog and Error Handling

To ensure long-term reliability, **watchdog timers** and **error-handling mechanisms** are going to be implemented at both the embedded and host software levels.

On the ESP32 firmware side, an independent watchdog timer (IWDG) resets the microcontroller if the control loop stalls or fails to respond within a specified timeout window. This prevents deadlocks that might occur due to unexpected serial communication errors or hardware faults. Additionally, periodic “heartbeat” packets are transmitted to the GUI to confirm MCU health; if the GUI does not receive these within a threshold, it automatically pauses ongoing acquisition and alerts the user. Below is Figure 40 which illustrates the error-handling software design that will be implemented.

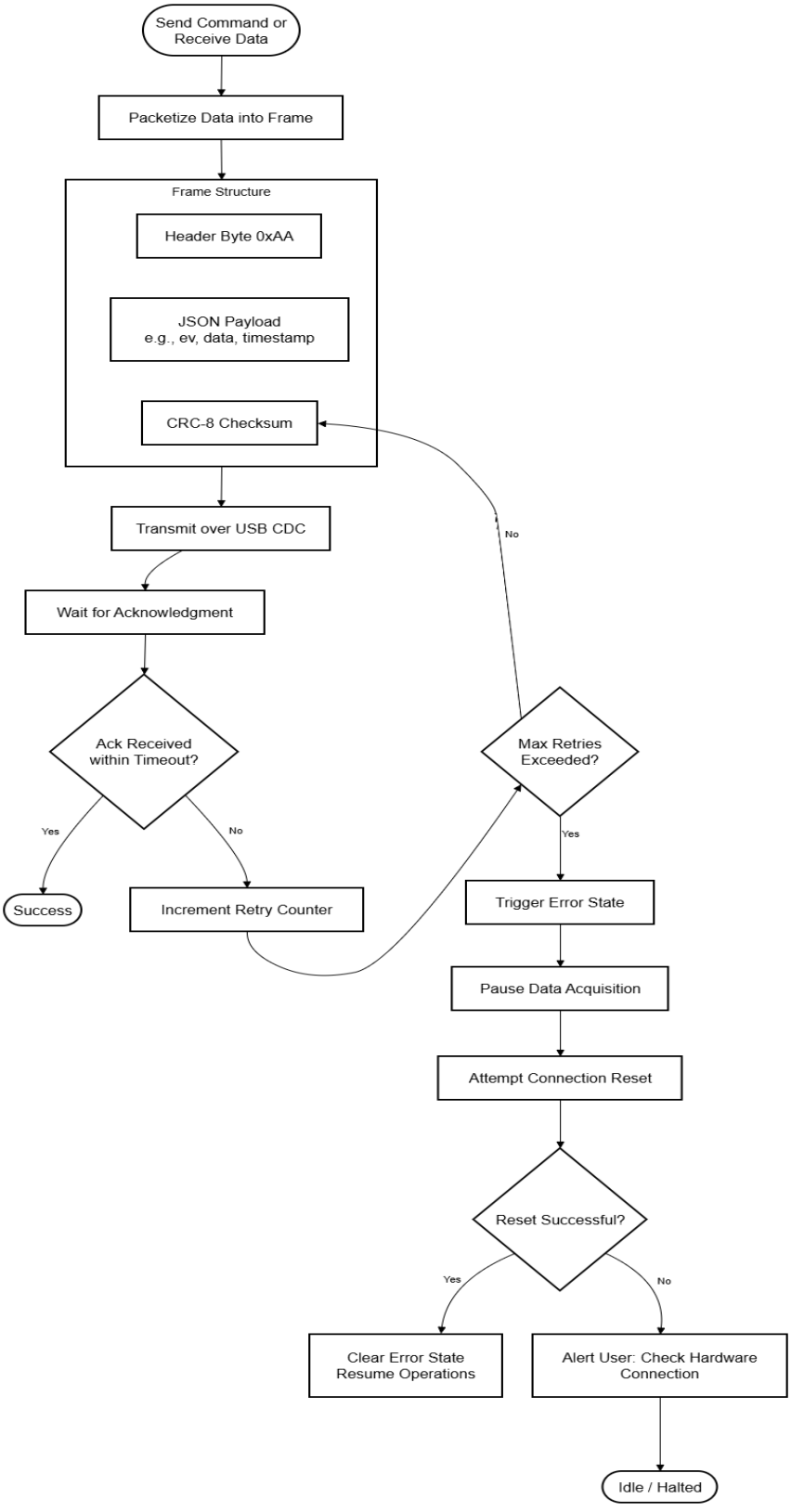


Figure 40: Error-Handling Software Design

Within the GUI, runtime exceptions—such as failed USB reads, invalid data formats, or out-of-range numerical results—are caught and logged to a diagnostics panel. The system automatically reinitializes the communication link if recovery is possible or prompts the user to restart the measurement. During integration testing, fault injection techniques (e.g., simulated disconnects, corrupted data bursts) were used to confirm graceful recovery under all tested conditions.

These safeguards collectively ensure that both hardware and software components can operate autonomously for extended experimental sessions without manual supervision or data corruption.

Chapter 8: System Fabrication/Prototype Construction

8.1: MCU PCB Design

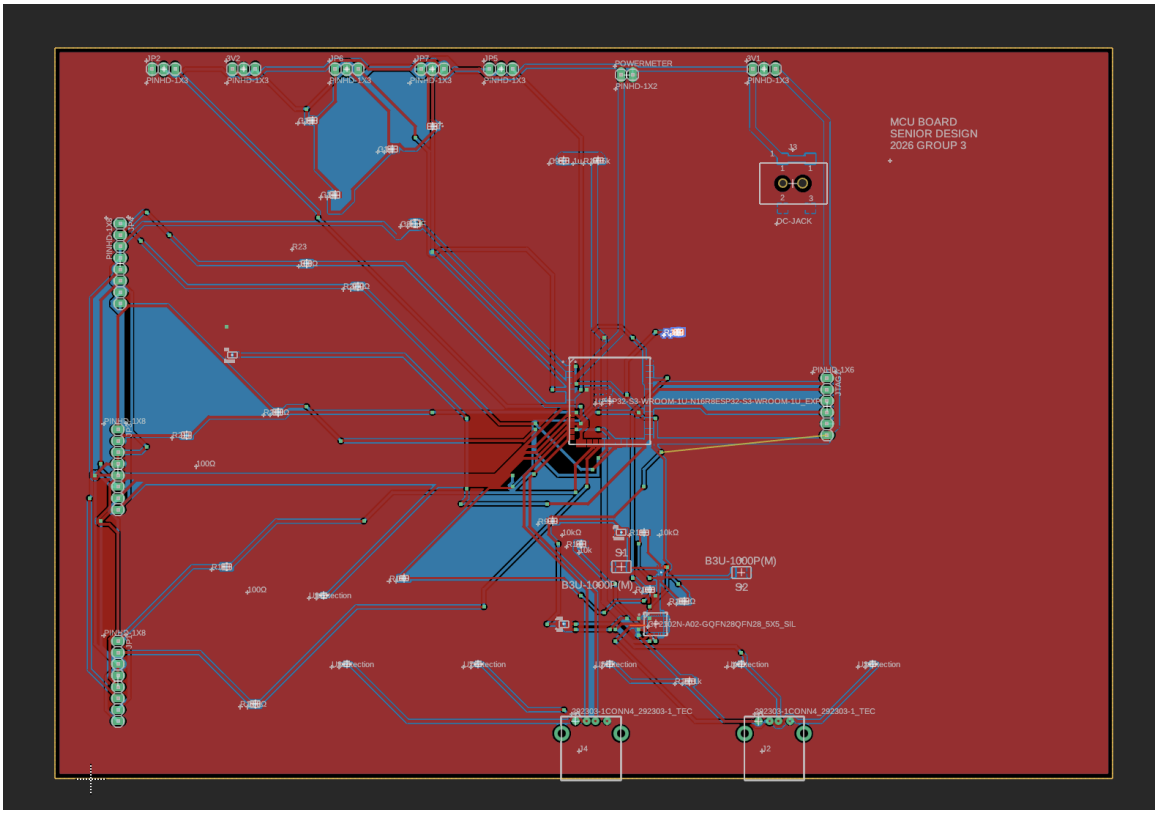


Figure 41: MCU Board PCB Design

8.2: Stepper Driver PCB Design

The stepper motor driver PCB design was taken directly off the DRV8825 datasheet recommended layout. It is laid out in such a way that it is flexible to be used with all three stepper motors.

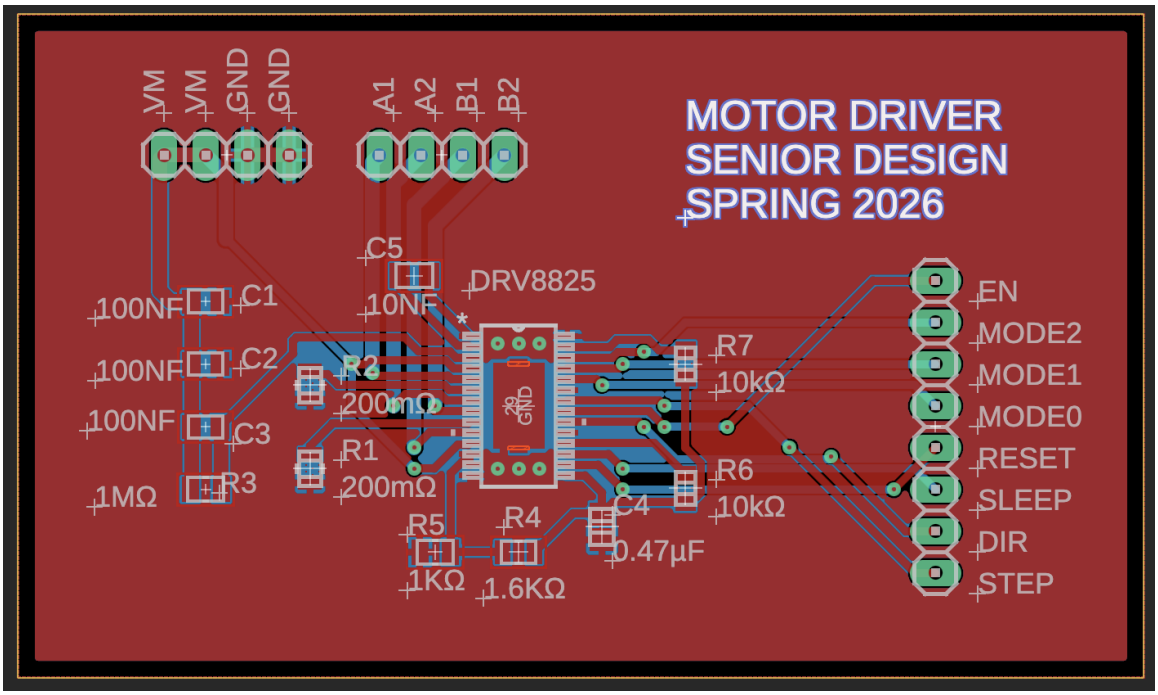


Figure 42: DRV8825 Motor Driver PCB Layout

8.3: Photodiode Circuit PCB Design

For the photodiode circuit, effort was made to make the photodiode amplifier configuration as flexible as possible and to assist with troubleshooting. Three things were done to accomplish this. The first was the addition of an LED to the circuit to tell if there is power being delivered to the circuit. This will assist in troubleshooting if for some reason there is an issue with the circuit. The second is a switch at the anode of the photodiode that will switch between ground and the VCC pin. In the original design, the photodiode was set to operate in photoconductive mode by setting the non-inverting input of the op amp to the reverse bias voltage, which will drive the inverting pin to the same voltage. The switch will allow for the reverse bias voltage to be applied directly to the photodiode instead. The third assists with this shift as well and is a zero Ohm resistor shorting the non-inverting resistor to ground which will not be installed in the initial setup but will be available should it be necessary to change the configuration. Additionally, to protect the components from the laser beam interference pattern that will be on the photodiode, the photodiode was configured to go on the bottom of the board. This will shield the components from the

optical energy on the board and prevent any concerns about the components being damaged by the laser beam. Below is the PCB layout.

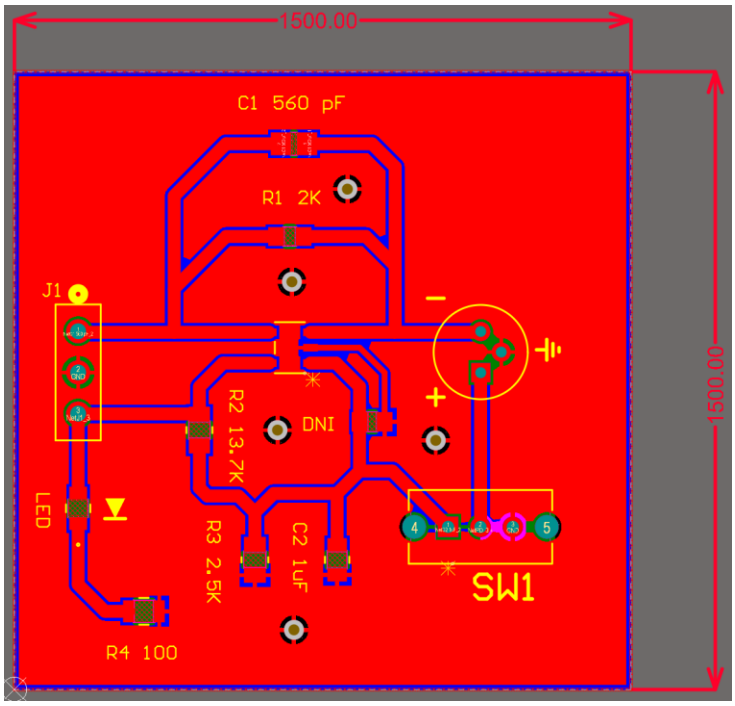


Figure 43: Photodiode Circuit PCB Layout

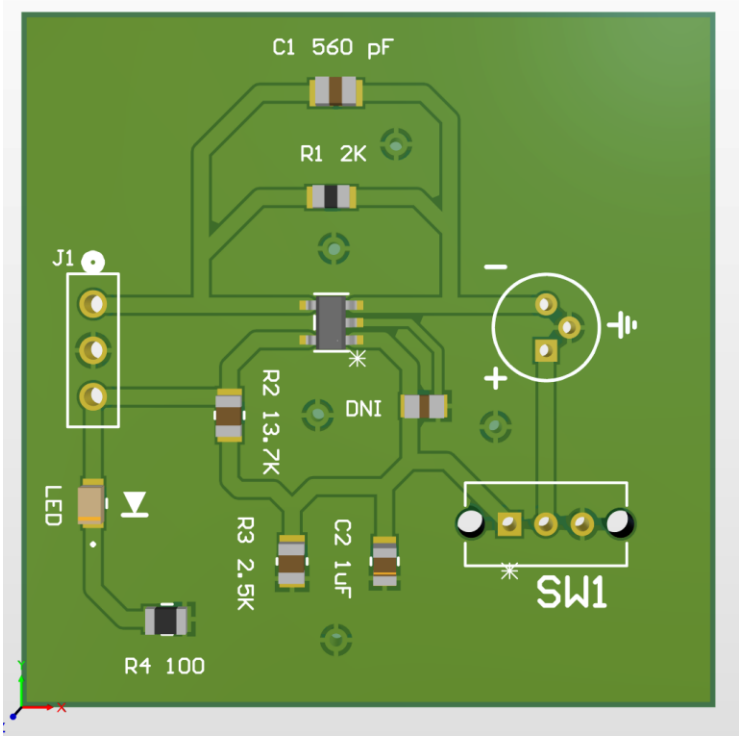


Figure 44: Photodiode Circuit PCB Layout Top Layer 3D View

The trace width recommended by the instructors for this course was 33 mils, so that is what was used. The only exception was for the ground trace for the VSS pin of the op amp, which could not fit a 33-mil trace in between the other two pins, so a 16 mil trace was used instead. Ground planes were added to the top and bottom layers, and a clearance of 6 mils was used. This was chosen because the depth of a 2-layer PCB board from JLCPCB was found to be 59 mils, so to set the trace impedance profile to be around 50 Ohms, an impedance calculator was used, and the closest trace clearance was found to be 6 mils. The calculation is shown below.

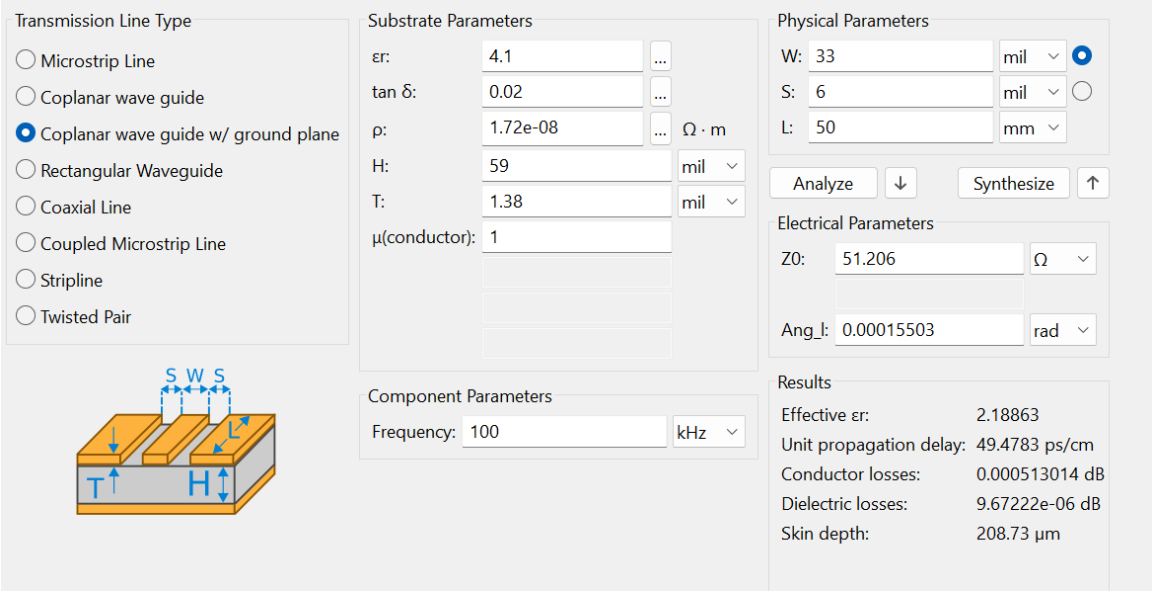


Figure 45: Photodiode Impedance Matching Calculations

8.4: Regulator PCB Design

Due to the importance of the voltage regulators, as well as the common failure, the 3.3V and 5V regulators are designed to connect to the main PCB through header pins. The modular design will make testing, troubleshooting, and revising much simpler.

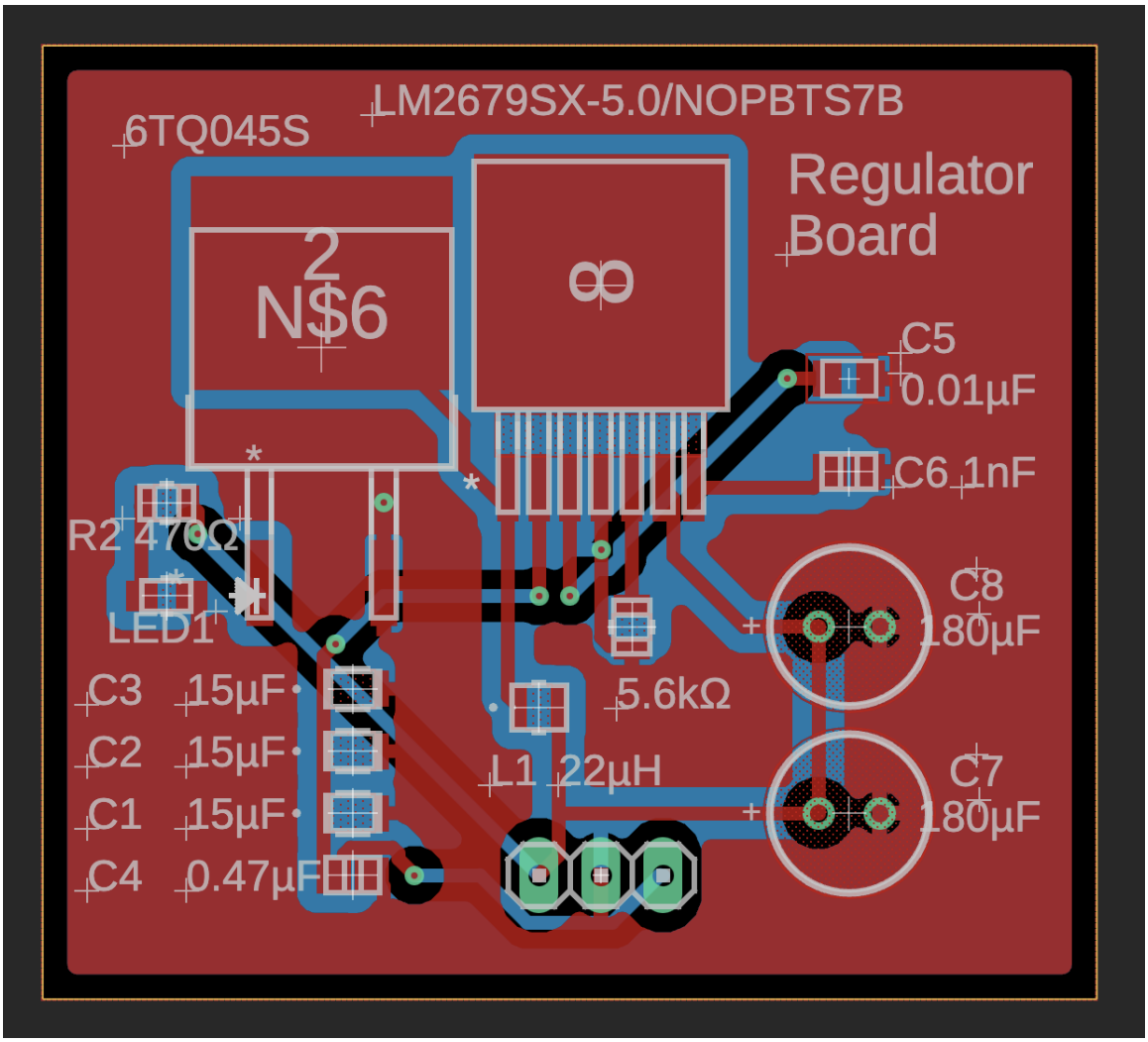


Figure 46: Regulator PCB Layout

8.5: 3D Printed Mounts

Many of the hardware parts need to fit together with custom 3D printed mounts. Since many of these parts are highly specific to our project, they were all custom modeled.

8.5.1: Quarter Wave Plate Rotation

The difficulty with designing a mount to rotate the quarter wave plate is to keep the optical area un-obstructed. We opted for a belt driven design with a stepper motor to drive the belt. This prototype was design to be easy to print and assemble. It uses some 4mm metal rods to provide less friction on the contact areas.

Figure 47: Quarter Waveplate with Stepper Motor Housing

8.5.2: Mirror Translation Mount

On the translation stage, we need a way to mount the two mirrors onto the moving block. The mount is designed to hold the two optical mounts, using a hex nut on the bottom to secure the mount. The design is slightly asymmetrical to perfectly align the mirrors on the optical bread board mounts. The mount is printed on its side to ensure maximum strength under the load of the mirrors.

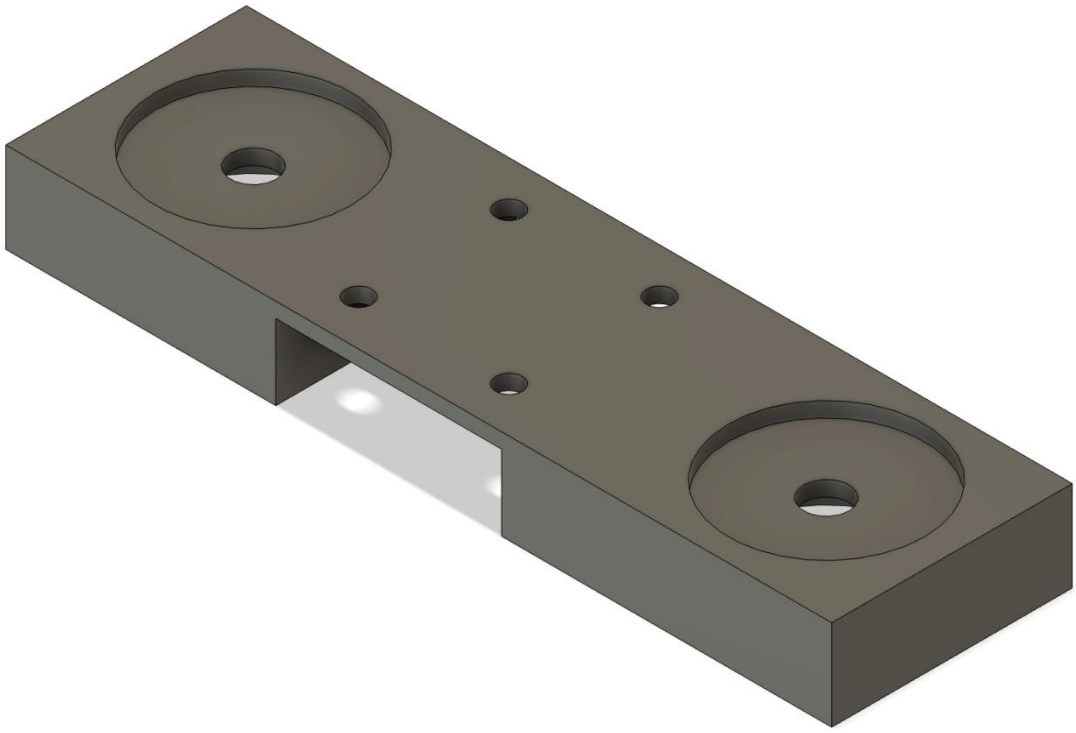


Figure 48: Beam Quality Mirror Translation Stage

8.5.3: Power Meter Servo Adapter

During beam profiling, we are using servo motors to scan the beam. These servo motors need to be mounted on the power meter using a custom mount. The mount has two holes to screw in the servos from the back, as well as a hole in the bottom to go through the power meter mounter screw.

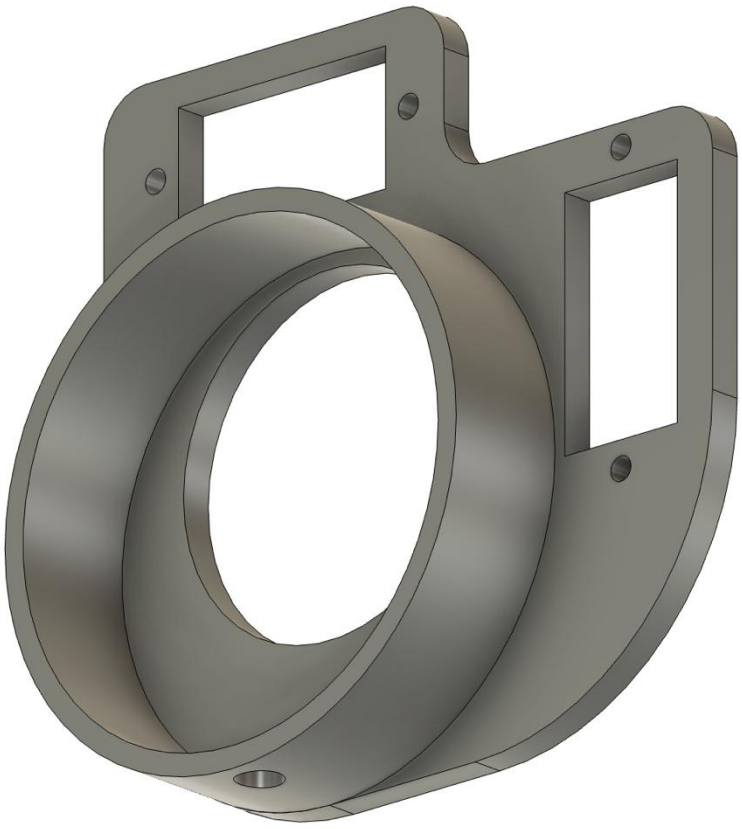


Figure 49: Power Meter Servo Adapter Housing

Chapter 9: System Testing and Evaluation

9.1: Michelson Interferometer Testing

Due to complications with sponsorship, some of the exact parts desired for this project could not be acquired. Therefore, similar parts were used to demonstrate the systems feasibility and act as proof of concept. Because of the lack of availability of a beam splitter in the IR region, a Helium Neon (HeNe) laser with a beam splitter that works in the visible region was used instead. One of the mirrors was put on the stepper motor stage (the same one shown in the Bill of Materials in Chapter 10), and the alignment was tweaked until the correct interference pattern was observed. Below is a picture of the setup.

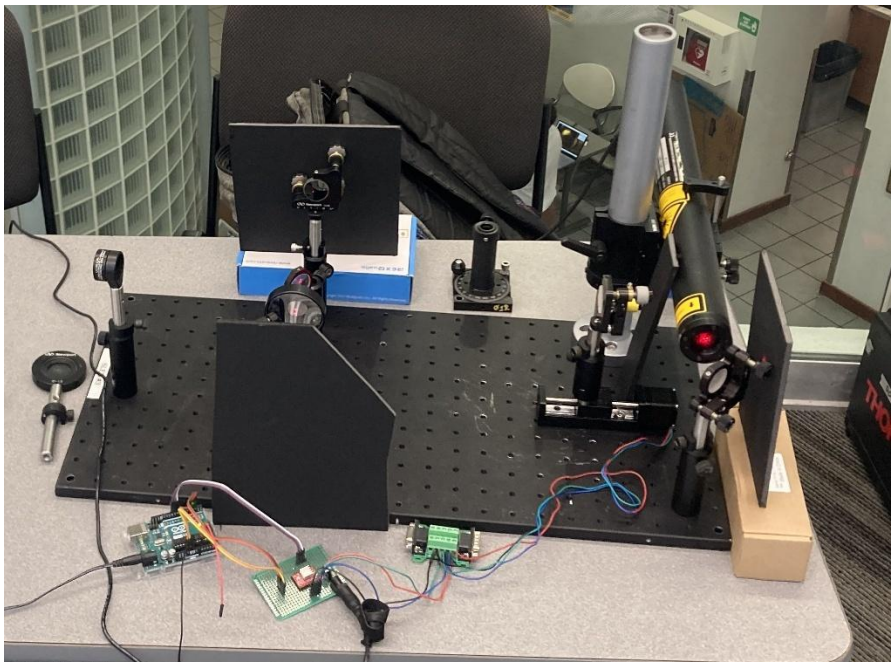


Figure 50: Michelson Interferometer Setup

As the motor next to the HeNe laser in the figure above was moved, the interference pattern would change. Because the wavelength of the laser is 633 nm, this would change too fast to visibly see, but when measuring the intensity change on an oscilloscope with a photodiode, the intensity pattern could clearly be seen. Taking the Fast Fourier Transform of this signal yields a plot showing the dominant frequencies in the signal. Given that the motor was translating around 2.5 mm/s, measuring the frequency of the highest peak in the frequency domain allows for an easy calculation of the wavelength of the laser. This was done and the highest peak was found to be around 8 kHz. As shown in the image below.

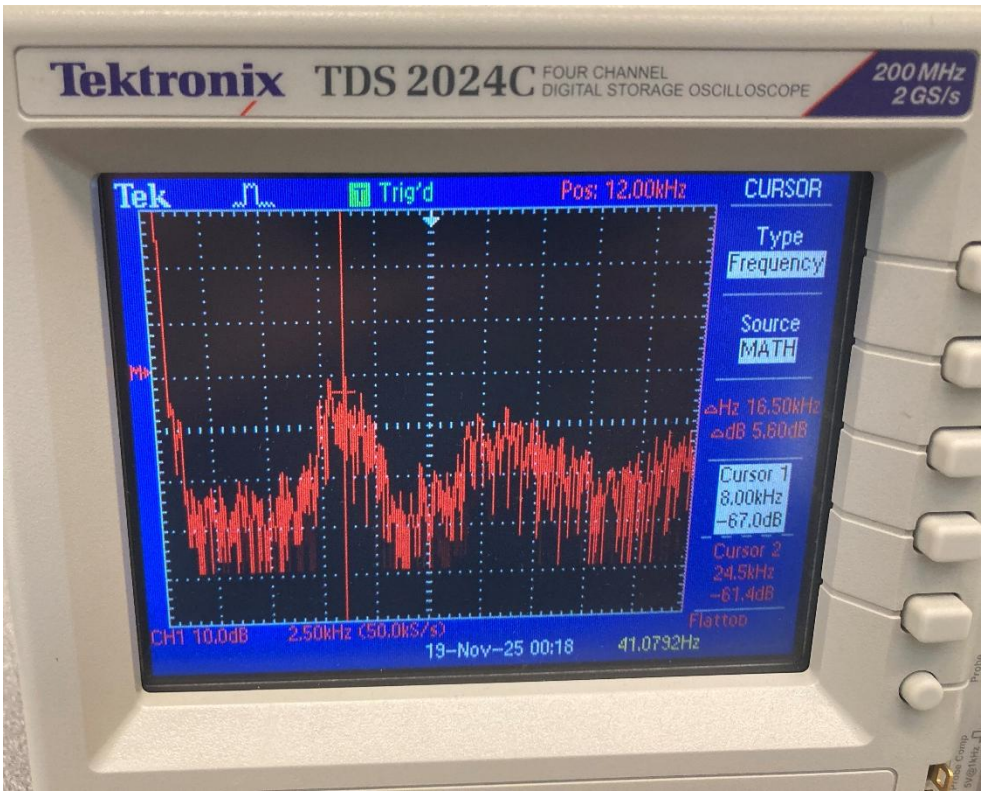


Figure 51: Oscilloscope Reading of Intensity Signal

Using these values, the equation below was used to measure wavelength, where v is the speed of the mirror and f is the frequency corresponding to the highest peak found in the frequency domain.

$$\lambda = \frac{2v}{f} = \frac{2 \times 2.5\ mm/s}{8\ kHz} = 625\ nm$$

This is an error of only 1.2% and shows that this method is definitely feasible for measuring the wavelength of the laser. With more extensive analysis, this error can likely be reduced even further to reduce the effects of noise in the signal.

9.2: Polarimeter and Beam Profiling Testing

Better parts were found for the polarimeter and beam profiling systems, that are more consistent with the desired parts for this project. These were set up in the designed layout, and the motors were implemented to demonstrate the functionality of the design. A 3D printed housing for the quarter waveplate with its respective motor was designed and implemented. The housing showed some resistance and will require some better mechanical design for smoother rotation but successfully rotated and the expected power changed through the following linear polarizer was observed as expected. The rotation was too inconsistent to get consistent angles, so an exact polarization measurement was not

observed, but previous data with this setup has shown that the method works exactly as expected. Below is some of the data that was collected for various input polarizations, all of which shows results well within the desired specifications of the project.

Table 34: Vertical Polarization Test								
QWP Angle (deg)	0	22.5	45	67.5	90	112.5	135	157.5
Power (mW)	0.0017	2.15	4.23	2.07	0.0018	2.26	4.46	2.20
Normalized Stokes Parameters				**Ellipse Parameters**				
S0	S1	S2	S3	Psi	Chi	P		
1	-0.999	0.016	-0.02	89.53	-0.65	1		

Table 35: Horizontal Polarization Test								
QWP Angle (deg)	0	22.5	45	67.5	90	112.5	135	157.5
Power (mW)	13.35	10.02	6.82	10.18	13.4	9.98	6.56	9.95
Normalized Stokes Parameters				**Ellipse Parameters**				
S0	S1	S2	S3	Psi	Chi	P		
1	0.999	-0.0097	0.017	-0.27	0.48	0.999		

Table 36: Right Hand Circular Polarization Test								
QWP Angle (deg)	0	22.5	45	67.5	90	112.5	135	157.5
Power (mW)	4.56	8.3	9.65	7.88	4.48	1.45	0.052	1.11
Normalized Stokes Parameters				**Ellipse Parameters**				
S0	S1	S2	S3	Psi	Chi	P		
1	-0.068	0.078	0.99	65.5	41.15	0.996		

For the beam profiling system, the biggest issue was figuring out how to read the data from the power meter we wish to use. The power meter is designed to communicate with a specific console from Thor Labs, the manufacturer, which is equipped with a 0-2 V analog output port with which the data could be read by the ESP32 microcontroller. When testing this, though, it was discovered that the 0-2 V output range is dependent on the power range set by the console. This makes sense in terms of hardware design, but it does cause an issue, since simply reading the analog output port of the console will leave an ambiguity

concerning which range the power meter is in. This will make reading the power on the detector impossible. Though it is possible to set the range to a certain value, this will cause resolution issues at lower power levels, which can disrupt several of the measurements this system seeks to take. Therefore, it will be necessary to design a variable amplifier that can amplify the current signal generated by the photodiode in the power meter head so that the microcontroller can detect it and know the power exactly. Until such is designed, the analog output port was used to read the signal on the microcontroller. Below is a picture of the test setup with the polarimeter and beam profiler setups integrated together. When translating the mirror stage, it was observed that the power on the detector did not change, implying that the alignment was successful and the beam would travel into the power meter without clipping even when the mirrors moved.

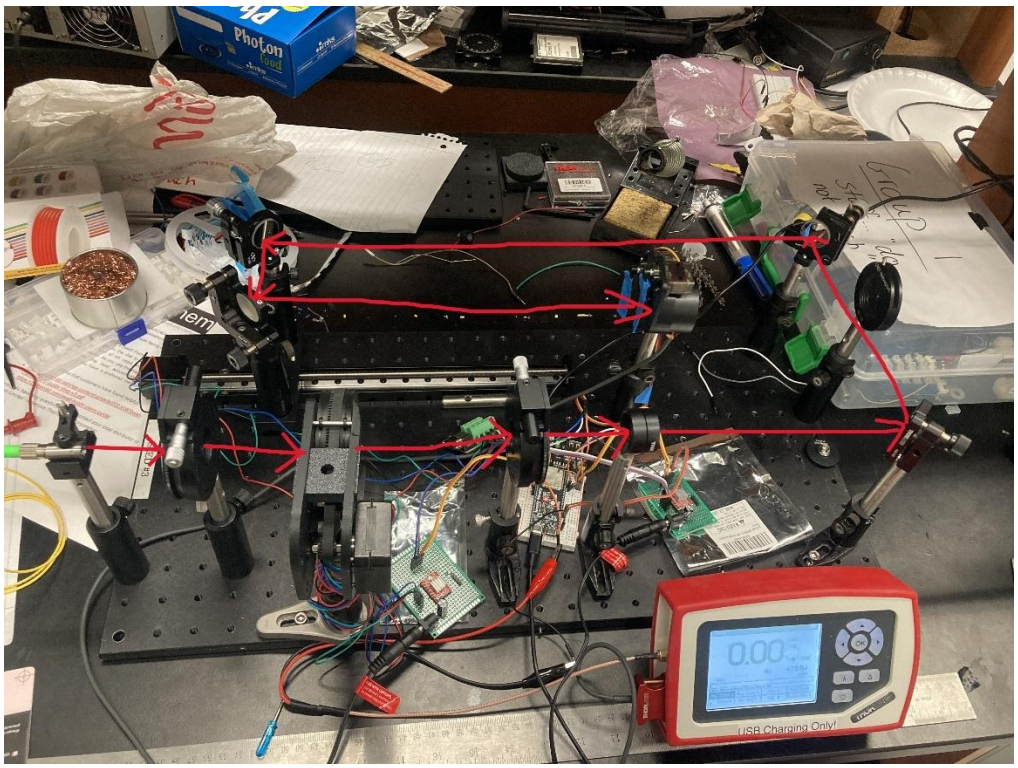


Figure 52: Polarimeter and Beam Profiling Setup

9.3 Motor Testing

The motors were all driven by an ESP32 microcontroller development board. There were 5 motors that needed to be driven, three of which were stepper motors and two of which were servo motors. For the servo motors, no driver was required, since the PWM signal that controls the position of the servo could be generated by the ESP32. The servo motors were supplied with a 5V supply, set by a purchased regulator board.

For the stepper motor drivers, the A4988 stepper motor driver was used. Though this is not the exact driver we plan to use, it functions the exact same as the DRV8825 driver and was

more readily available at the time of testing. In fact, the constraints for this driver are tighter than those of the DRV8825 driver, so it is reasonable that the DRV8825 driver will work the same if not better than this one. When testing, the motor was supplied with a 12V source and was controlled by a microcontroller. Since Arduino IDE was used, the initial testing was done with an Arduino. This was done successfully and so the test was then conducted again with the ESP32, the microcontroller we plan to use in this project. Under the same conditions, the motors were driven as expected. With these drivers, there is an adjustable current limit that was adjusted to determine the best current that will allow the motor to run while keeping it low enough to keep the motor from being damaged. This was tested with the Michelson, Beam Quality, and Quarter Waveplate Rotating Stepper motors. The Michelson and Beam Quality stepper motors work as expected, though they were a bit noisier than desired. The Quarter Waveplate motor worked, but the mount seemed to have more friction than would work for a good measurement. Better mounting designs will be explored to address this issue.

9.4: Optoelectronics Feasibility

There are two photodiodes in the system. The first one is contained in the power meter head and is used for reading power for the polarimeter subsystem as well as reporting the power curve when taking a beam profile. This photodiode is contained in an integrating sphere, such that the power into the detector is not directly incident on the photodiode but rather is spread out over the whole sphere. This allows the detector to be able to measure much higher powers than if the laser were directly incident on the detector. Additionally, the responsivity curve lies in the mA/W range, as seen in the figure below, which means that the detector can take a lot of optical power before the current becomes too high to maintain power meter functionality. This is ideal for this application to be able to profile at higher power levels.

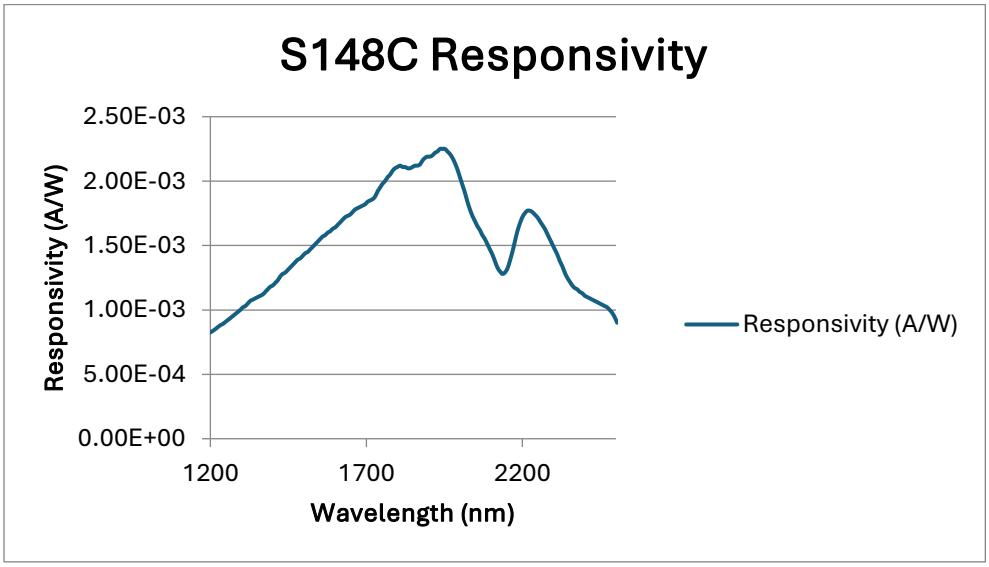


Figure 53: S148C Power Meter Responsivity Curve

Since the Michelson interferometer will be measuring the wavelength of the laser beam, the exact responsivity of the photodiode will be known. This value is critical to measuring the power of the laser beam and from that all the other quantities we seek to measure in this project. There is also a photodiode in the Michelson interferometer. This is used to read the intensity signal that changes as the mirror translates in the interferometer. The responsivity of this photodiode was shown in Figure 19 above, and it shows that the photodiode selected has high responsivity in our desired spectral range. This is critical as it means the interferometer will work at low optical powers, which is achievable since the polarimeter will be able to control the amount of power transmitted into the interferometer. Additionally, it is necessary to know the bandwidth of this photodiode as this will determine the gain at the frequency of the intensity signal. This is determined by the capacitance of the photodiode, the feedback resistance of the transimpedance amplifier, as well as the gain-bandwidth product of the op amp used. Using these values the 3 dB point of the frequency response can be calculated as the following. [27]

$$f(-3dB) = \sqrt{\frac{GBP}{4 * \pi * R_f C_J}}$$

Where

- GBP is the gain-bandwidth product of the op amp
- R_f is the value of the feedback resistor
- C_J is the value of the photodiode junction capacitance

Using this equation, the 3 dB point of the amplifier comes out to be about 684 kHz. This is clearly greater than the frequencies the interferometer will be working at, so the amplifier will work despite the junction capacitance of the photodiode. Thus this photodiode is feasible for this application.

9.4: Software Integration

For the current phase of development, the entire embedded software stack is being implemented using the Arduino IDE to program the ESP32-S3. Although the ESP32 platform ultimately supports more advanced development environments such as ESP-IDF, PlatformIO, or FreeRTOS-based systems, the Arduino environment provides a fast, straightforward, and flexible way to bring up the hardware, test critical subsystems, and iterate quickly on the firmware responsible for controlling the motion stages and reading the analog power-meter signals. Using Arduino as the initial firmware platform allows the team to validate all electrical and mechanical components before transitioning to a more complex environment later in the project.

Within the Arduino environment, the ESP32-S3 is programmed using the standard setup() and loop() structure. All pin assignments, ADC configuration, and motion-control timing parameters are defined using familiar Arduino functions like pinMode(), digitalWrite(), and analogRead(). This greatly simplifies the early development cycle and eliminates the need to configure ESP-IDF components, CMake toolchains, or hardware abstraction layers during prototyping. Because the ESP32-S3 is fully supported by the Arduino core, features such as high-resolution timers, ADC sampling, and PWM generation can be accessed with minimal configuration effort.

The firmware integrates three main control tasks: stepper motor actuation, servo positioning, and ADC power-meter sampling. The stepper motors are driven through A4988-style drivers, and the ESP32-S3 generates precise step pulses by toggling the STEP and DIR pins with microsecond-scale timing. Using Arduino's delay() function makes the timing deterministic and easy to tune. The firmware sets the direction, outputs a clean pulse on the STEP pin, waits for the pulse width to complete, and then inserts a small delay before proceeding. This ensures that each step is mechanically consistent and reduces noise coupling into the ADC. Even though the ESP32-S3 has powerful hardware timers, the basic Arduino delay functions provide sufficient precision for the current microstepping and scan-speed requirements.

Servo control is implemented using Arduino's built–in PWM functionality. The ESP32-S3 generates accurate pulse-width modulation signals that correspond to servo positions, typically between 0° and 180°. A library call such as servo.write(angle) or writing a custom pulse width allows the firmware to move rotational elements smoothly and predictably. With Arduino handling the PWM timers internally, configuration is minimal, and testing servo motion becomes a simple matter of sending a few commands and observing physical movement.

One of the most important roles of the ESP32-S3 firmware is reading the analog output of the power meter using the built-in ADC. The Arduino IDE exposes this using the familiar analogRead() function. The code captures a raw ADC value (0-4095 for the ESP32 12-bit ADC), converts it to a voltage using the known reference voltage, and then maps that voltage to an estimated optical power based on the meter's calibration range. This allows the firmware to associate each motor step with precise power reading. To improve measurement stability, a brief settling delay is inserted between step pulses and ADC reads, giving the mechanical system time to stabilize before sampling the analog signal. This timing is easy to adjust in Arduino, making it ideal for iterative testing.

Using the Arduino IDE at this stage also simplifies debugging. Serial prints can be added instantly without needing to configure USB descriptors, logging subsystems, or custom drivers. This allows the development team to view step counts, power values, servo angles, and diagnostic information directly from the Serial Monitor while tuning parameters like step speeds, ADC timing, and motion profiles.

Overall, using the Arduino IDE to program the ESP32-S3 provides a practical and efficient way to integrate the stepper drivers, servos, and analog power-meter inputs during the early phases of system development. It enables rapid prototyping, straightforward debugging, and accessible control over timing-critical hardware interactions. Once the hardware and motion sequences are fully validated, this foundation can be migrated to a more advanced

firmware environment, but for now the Arduino approach delivers everything needed to reliably control the system and gather accurate optical data.

Chapter 10: Administrative Content

10.1 Bill of Materials

Table 37: Bill of Materials				
	Component	**Quantity**	**Estimated Unit Price**	**Cost**
Optics	QWP	1	$550	$550
	Linear Pol.	1	$2200	$2200
	1” mirrors	4	$60	$240
	2” Mirrors	2	$120	$240
	Beam Splitter	1	$380	$380
	Lens	1	$250	$250
	Power Meter	1	$1060	$1060
	ND Filter	3	$75	$225
Motors	Beam Quality Stepper	1	$80	$80
	Michelson Stepper	1	$55	$55
	QWP Rotation Steppers	1	$18	$18
	Stepper Drivers	3	$10	$30
	Movable Mirror Servo	1	$15	$15
	Scanning Slit Servo	2	$5	$10
	Motor Belt	1	$10	$10
Circuits	ESP32	1	$100	$100
	Photodiode	1	$100	$100
	Power	3	$30	$90
Mounts	Posts	10	$6.20	$62
	Post Holders	10	$17.5	$175
	1” Optic Mounts	4	$19.7	$78.80
	2” Optic Mounts	4	$26.3	$105.20
Misc.	Knife Edge Blades	2	$5	$10
	Baseplate	1	$700	$700
			Total	$6784
			Budget	$7000

10.2 Milestones

10.2.1 Senior Design 1 Milestone Table

Table 38: Senior Design 1 Milestone Table			
Milestone Number	**Milestone Description**	**Start Date**	**Anticipated End Date**
1	Group Forming	8/18/2025	8/26/2025
2	Project Idea Finalized	8/18/2025	9/5/2025
3	D&C Document and Committee Formation	8/25/2025	9/5/2025
4	D&C Meeting	9/8/2025	9/10/2025
5	Midterm Milestone Report	9/30/2025	10/31/2025
6	Midterm Milestone Report Revision	11/3/2025	11/7/2025
7	Final Report	10/31/2025	11/25/2025
8	Mini Demo Video	11/17/2025	11/25/2025

10.2.2 Senior Design 2 Milestone Table

Table 39: Senior Design 2 Milestone Table			
Milestone Number	**Milestone Description**	**Start Date**	**Anticipated End Date**
1	PCB Testing	1/12/2026	1/30/2026
2	System Integration/Testing	2/2/2026	3/2/2026
3	Documentation Finalized	2/2/2026	4/13/2026
4	Final Presentation Practice	4/14/2026	4/20/2026
5	Final Presentation	TBD	TBD

10.3 Work Distribution Table

Table 40: Work Distribution Table	
Photonics Engineering	**Responsibilities**
Zachary Andreasen	• Project Lead • Polarimeter, Michelson system design • Optical alignment procedures • Photodiode system design
Electrical Engineering	Responsibilities
Primary: Tee Cunningham **Secondary: Zachary Andreasen**	• MCU selection • Motor drivers' selection and wiring • Mechanical and motion system control (Stepper/Servo) • Power design for overall system • PCB/breadboard design
Computer Engineering	Responsibilities
Lynzie Smith	• GUI software architecture and implementation • Serial communication integration • Website design • ESP32-S3 firmware (ADC sampling, motor control, etc.)

Chapter 11: Conclusion:

In conclusion, this project aims to measure several quantities of a laser beam which can benefit many different applications. Though there are many different methods to accomplish the tasks to be done, this project's design aims to choose the most effective cost-efficient ones to achieve quality measurements in a single system. The subsystems are summarized in detail below.

11.1: Polarimeter

For the polarimeter design, the rotating quarter waveplate method was selected as the best means to calculate the Stokes Parameters of the beam. Thus, this system will include a quarter waveplate, a linear polarizer, and a power meter. The quarter waveplate will be rotated by means of a stepper motor controlled by the STM32 microcontroller. To get accurate measurements, the quarter waveplate retardance will be known from the wavelength measurement and the linear polarizer has a significant polarization extinction ratio. Additionally, to save on space and money, the power meter used for this system will

be shared with the beam profiling system. The measurements can then easily be taken and processed by the software.

11.2: Wavelength Measurement

For the measuring of the wavelength, though there are several methods, the selected method was a Michelson Interferometer. This method is both broadband and high resolution and therefore compromises the advantages of the spectrometer and beat frequency analysis methods. The photodiode circuit will be able to read the intensity of the light as one of the mirrors is translated to extract the wavelength of the beam. In order to maintain an appropriate power on the device, the rotation of the quarter waveplate can be positioned such that there is the optimal amount of power on the device. A simple stepper motor was selected as the best means by which the mirror can be translated and the wavelength calculated. Additionally, the calculations were made such that the limitations on the speed of the motor without aliasing the signal are known.

11.3: M^2factor measurement

For the M^2 factor measurement, the calculations for the focal length of the lens were made to compensate for the long Rayleigh ranges in the case of a small beam, and the short Rayleigh ranges in the case of a large beam. The design was then altered to optimize space and step resolution constraints. For the beam profiler, servo motors were selected as the best motor to profile the beam as they have excellent position control and can easily be controlled by the STM32.

11.4: Divergence Measurement

The divergence measurement can easily be integrated into the M^2 factor measurement. This comes in the form of measuring the beam width at the focus of the lens chosen. Though this is dependent on wavelength, since the wavelength is measured, the focal length can easily be determined and the beam width measured at that location. This allows for the calculation of the beam divergence.

Appendices

Appendix A - References

[1] Ophir Photonics, “PH00470 Scanning-Slit Beam Profiler.” Accessed: Sept. 05, 2025. [Online]. Available: https://www.ophiropt.com/en/p/PH00470

[2] Thorlabs, “BP209-IR2 Scanning-Slit Beam Profiler.” Accessed: Sept. 05, 2025. [Online]. Available: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=804&pn=BP209-IR2

[3] DataRay Inc., “BeamMap2 Scanning-Slit Beam Profiler.” Accessed: Sept. 05, 2025. [Online]. Available: https://store.dataray.com/all-products/scanning-slit-beam-profilers/beammap2/

[4] Ophir Photonics, “SP90404 Pyrocam IV Laser Beam Profiler.” Accessed: Sept. 05, 2025. [Online]. Available: https://www.ophiropt.com/en/p/SP90404

[5] Allied Vision, “Goldeye CL-034 XSWIR 1.9 TEC2.” Accessed: Sept. 05, 2025. [Online]. Available: https://www.alliedvision.com/en/camera-selector/detail/goldeye/cl-034-xswir-19-tec2/

[6] Thorlabs, “Polarimeter Systems with High Dynamic Range (PAX1000 Series).” Accessed: Sept. 05, 2025. [Online]. Available: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=1564

[7] OZ Optics, “ER-100-IR Polarization Extinction Ratio Meter & Polarized Sources.” Accessed: Sept. 05, 2025. [Online]. Available: https://shop.ozoptics.com/er-100-ir

[8] R. Paschotta, “M^2 factor – M squared, laser beam, quality factor, beam divergence, caustic, ISO Standard 11146.” Accessed: Sept. 05, 2025. [Online]. Available: https://www.rp-photonics.com/m2_factor.html

[9] Thorlabs, “M^2 Measurement System.” Accessed: Sept. 05, 2025. [Online]. Available: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=7728

[10] Gentec Photonics, “Gentec BeamGaGe M^2 Measurement System Datasheet.” Accessed: Sept. 05, 2025. [Online]. Available: https://www.lasercomponents.com/fileadmin/user_upload/home/Datasheets/gentec/beamage-m2-measurement-system.pdf

[11] Ocean Optics, “NIRQuest OEM Spectrometer Datasheet.” Accessed: Sept. 05, 2025. [Online]. Available: https://www.oceanoptics.jp/technical/oem_datasheet_nirquest.pdf

[12] Inc. StellarNet, “Dwarf Star NIR Spectrometers.” Accessed: Sept. 05, 2025. [Online]. Available: https://www.shopstellarnet.com/dwarf-star-nir-spectrometers/

[13] B&WTek, “i-Spec.0 Spectrometer Datasheet.” Accessed: Sept. 05, 2025. [Online]. Available: https://www.snowhouse.ca/pdf/BWTek_i-Spec.0.pdf

[14] Marktech Optoelectronics, “MTPD2601T-100 High Speed InGaAs PIN Photodiode Datasheet.” Marktech Optoelectronics, Feb. 15, 2024. [Online]. Available: https://specs.marktechopto.com/pdf/products/datasheet/MTPD2601T-100.pdf

[15] B. Schaefer, E. Collett, R. Smyth, D. Barrett, and B. Fraher, “Measuring the Stokes polarization parameters,” *Am. J. Phys.*, vol. 75, no. 2, pp. 163–168, Feb. 2007, doi: 10.1119/1.2386162.

[16] Thorlabs, *Build a Polarimeter to Find Stokes Values, Polarization State (Viewer Inspired) | Thorlabs Insights*, (Oct. 20, 2021). [Online Video]. Available: https://youtu.be/pR4r7gMyN5U
[17] S. N. Savenkov, "Jones and Mueller matrices: structure, symmetry relations and information content," in *Light Scattering Reviews 4: Single Light Scattering and Radiative Transfer*, A. A. Kokhanovsky, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 71–119. doi: 10.1007/978-3-540-74276-0_3.
[18] "The Polarization Ellipse Representation of the Polarization State." [Online]. Available: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=14199
[19] "Using the Poincaré Sphere to Represent the Polarization State." [Online]. Available: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=14200
[20] R. Paschotta, "Gaussian Beams," 2005, doi: https://doi.org/10.61835/mla.
[21] R. Paschotta, "Fabry–Pérot Interferometers," 2007, doi: https://doi.org/10.61835/qrw.
[22] Litron Lasers, "Comparison of Beam Divergence, Beam Diameter and M2," 2025, [Online]. Available: https://www.litronlasers.com/technical-notes/comparison-of-beam-divergence-beam-diameter-m2/
[23] J. Caldwell, "1 MHz, Single-Supply, Photodiode Amplifier Reference Design," Nov. 2014, [Online]. Available: https://www.ti.com/lit/ug/tidu535/tidu535.pdf?ts=1759929205864
[24] P. Ghosh, "Applying IPC-2221 Standards in Circuit Board Design | ProtoExpress Blog." [Online]. Available: https://www.protoexpress.com/blog/ipc-2221-circuit-board-design/
[25] Edmund Optics, "Gaussian Beam Propagation | Edmund Optics Knowledge Center." [Online]. Available: https://www.edmundoptics.com/knowledge-center/application-notes/lasers/gaussian-beam-propagation/
[26] XL-Semicon, "XL-2012SURC Technical Data Sheet." XL-Semicon / DigiKey, ?? [Online]. Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/6472/5962_XL-2012SURC%200805%E7%BA%A2.pdf
[27] Inc. Thorlabs, "Photodiodes and Photoconductors Tutorials." [Online]. Available: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=9020

Appendix B – Copyright Permission

Appendix C – ChatGPT Prompts and Outcomes

[1a] **Case Study 1 Prompt**: "What are some of the best MCUs for an engineering project?"

[1b] **Case Study 1 ChatGPT Result**: "If you're working on an engineering project and trying to pick a microcontroller (MCU), there are many solid choices — the "best" one depends a lot on your requirements (e.g., connectivity, performance, power, peripherals). Below is a quick summary of **what to look for**, followed by **some of the best MCU / development-board options** you might consider. What to consider when choosing an MCU

Here are some of the key criteria to compare, so you pick something that fits your project rather than over- or under-spec'ing:

- **Core architecture & performance**: e.g., 8-bit vs 16-bit vs 32-bit, clock speed, floating point unit or not. For more advanced tasks you'll want a 32-bit Cortex-M or equivalent. electricaleasy.com+2MoldStud+2
- **Memory (Flash + RAM)**: Enough program space + runtime memory for your application (especially if you do buffers, real-time processing, DSP, etc.). MoldStud+1
- **Peripherals / I/O**: Do you need lots of GPIO pins, analog inputs (ADC), communication interfaces (SPI, I²C, UART, CAN, USB, WiFi, BLE)?
- **Connectivity / wireless**: If you're doing IoT, remote sensors, etc., built-in WiFi/BLE or support for external modules may matter. OneSDR - Technology+1
- **Power consumption**: For battery-powered or low-energy devices, look for MCUs with low-power modes, ultra-low sleep consumption. Nabto+1
- **Ecosystem / community / development tools**: Good documentation, libraries, sample code, development boards, ease of programming/debugging — this can save tons of time. As one Redditor put it:

"Pay attention to the community built up around the microcontroller and the software offered … Being able to go to a forum and finding a post covering the exact issue you're having makes a world of difference." Reddit

- **Cost / availability**: Often a practical constraint, especially for student projects or prototypes.
- **Scalability / future-proofing**: If you might reuse the board in other projects, consider how flexible it is.

If you match your project needs (sensor count, speed, connectivity, power) to these criteria, you'll pick something more appropriate rather than just "what looks trendy". Some of the best MCU / development board options Here are some highly recommended development boards / MCU platforms. You can pick by budget, complexity, connectivity, etc. Here's a breakdown:

- Raspberry Pi Pico (RP2040): A very low-cost board (around US$4-5) based on the RP2040 dual-core ARM Cortex-M0+ @133 MHz. Good I/O, supports MicroPython and C/C++. Great for sensor hubs, prototyping. livetoplant.com+1
- Raspberry Pi Pico (alternate listing): Same board (RP2040) via a different listing/vendor.
- STM32 NUCLEO-F302R8 (STM32): A board in the STM32 ecosystem (ARM Cortex-M). Good step up when you need more power / flexibility than very basic boards.
- SparkFun MicroMod STM32 Processor Board: A higher-end STM32 board, useful when you need extra performance, floating point, more advanced peripherals.
- Arduino Mega 2560 (ATmega2560): While older / lower-power compared to modern 32-bit MCUs, the Arduino ecosystem is extremely beginner-friendly and

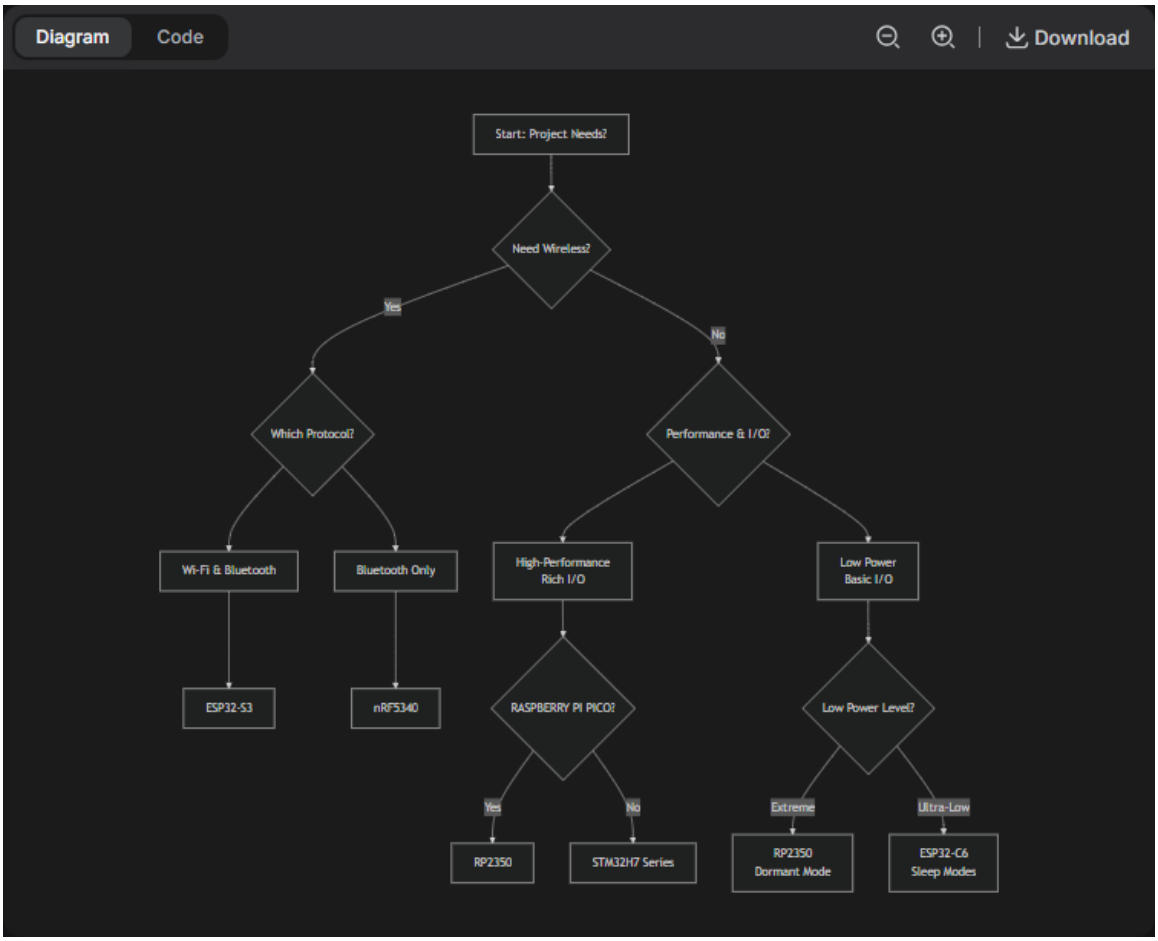
has tons of shields/libraries. Good I/O count for projects needing many sensors/actuators.
- Mikroe Ready for PIC Board: For those wanting to explore PIC ecosystem (Microchip) – another popular MCU family, especially in industrial/embedded.
- Pimoroni Tiny Dev Board (RP2350A): A very small form-factor board with modern MCU (dual-core ARM Cortex-M33). Good for space-constrained or advanced microcontroller projects.
- Microchip Development Board: A general dev board from Microchip, useful if you want to go deep into that ecosystem (PIC, dsPIC, etc).

Here are suggestions depending on what your project is:

- If you're just starting out and want something simple: go with the Raspberry Pi Pico (RP2040) or Arduino Mega.
- If you need many I/Os and connectivity (e.g., sensors + actuators + communications): pick the STM32 NUCLEO or SparkFun STM32 board.
- If you're doing IoT / wireless (WiFi, BLE) or need ultra-low power: consider MCU families that support those (for example boards built around the ESP32, nRF52, or STM32WB). Note: the above list doesn't include an ESP32 board but that is also a strong contender often discussed. OneSDR - Technology
- If footprint/size is key (wearables, compact devices): small boards like the Pimoroni Tiny Dev Board with RP2350A might be best.
- If budget is super tight or you just need something for simple monitoring/data-logging: the Pico or Arduino will suffice."

[1c] **Case Study 1 DeepSeek Results**: "Of course! This is a fantastic question, as the "best" MCU is entirely dependent on the specific requirements of your engineering project. I'll break down the top contenders by their strengths and typical use cases.

Here's a quick decision flowchart to start:



Now, let's get into the detailed breakdown of the top MCU families.

1. For All-Rounders, Hobbyists, and Rapid Prototyping

These are easy to use, have massive communities, and are incredibly versatile.

Raspberry Pi Pico (RP2040)

- **Core:** Dual-Core ARM Cortex-M0+
- **Key Features:** Low cost, versatile programmable I/O (PIO), easy-to-use C/C++ SDK and MicroPython.
- **Best For:** Beginners, projects needing precise timing (with PIO), general-purpose control, and when you want a great balance of performance and ease of use.
- **Why it's a top pick:** The **RP2040** is a modern masterpiece. Its PIO (Programmable I/O) blocks allow you to implement custom communication protocols (like DVI video or precise stepper motor control) in software, offloading the main CPUs. It's cheap, powerful enough for most tasks, and has a fantastic ecosystem.

- **Example Project:** A custom keyboard, a data logger, a motor controller, an LED panel driver.

ESP32-S3

- **Core:** Dual-Core Xtensa LX7 or single/dual-core RISC-V (depending on variant)
- **Key Features:** Wi-Fi 802.11b/g/n, Bluetooth 5 (LE + Mesh), plenty of I/O, USB-OTG, AI acceleration instructions.
- **Best For:** IoT projects that need both Wi-Fi and Bluetooth, devices with USB host capabilities, and projects experimenting with tinyML.
- **Why it's a top pick:** It's the go-to for feature-packed wireless projects. The community support is enormous (Arduino, ESP-IDF), and it offers incredible performance for its price. The USB-OTG is a huge win for acting as a host for keyboards, drives, etc.
- **Example Project:** A smart home device, a web server, a Bluetooth game controller, a device that reads data from a USB sensor.

2. For Wireless & IoT Specific Projects

If your project's primary requirement is connectivity, start here.

ESP32-C6

- **Core:** Single-core 32-bit RISC-V
- **Key Features: Wi-Fi 6 (802.11ax)**, Bluetooth 5 (LE + Mesh), and **IEEE 802.15.4 (Thread/Zigbee)**.
- **Best For:** The next generation of IoT devices, especially where you need better Wi-Fi performance, lower latency, or want to experiment with Matter/Thread.
- **Why it's a top pick:** It's future-proof. Being one of the first MCUs with Wi-Fi 6, it's ideal for dense wireless environments. The inclusion of 802.15.4 radio makes it a powerhouse for smart home standards.

nRF5340 (Nordic Semiconductor)

- **Core:** Dual-Core ARM Cortex-M33 (Application + Network core)
- **Key Features:** Bluetooth 5.4/5.5 (with Direction Finding, Audio), ultra-low power, and a dedicated network core for protocol handling.
- **Best For:** Professional, power-sensitive Bluetooth Audio (LE Audio) and complex Bluetooth Mesh applications.
- **Why it's a top pick:** Nordic is the gold standard for Bluetooth. The dual-core architecture is highly efficient, and their software support (nRF Connect SDK, based on Zephyr RTOS) is excellent for commercial products.

- **Example Project:** A high-end wireless headset, a medical wearable, a large-scale Bluetooth mesh sensor network.

3. For High Performance & Complex Applications

When you need serious computational power, DSP capabilities, or a rich set of peripherals.

STM32H7 Series (STMicroelectronics)

- **Core:** ARM Cortex-M7 (up to 600+ MHz), often with a Cortex-M4 core.
- **Key Features:** Tons of RAM/Flash, extensive peripherals (Ethernet, CAN FD, multiple timers, crypto), and DSP capabilities.
- **Best For:** Industrial control, advanced motor control, digital power supplies, graphics displays, and complex algorithms.
- **Why it's a top pick:** The Cortex-M7 core with its cache and FPU brings near-Microprocessor performance while retaining the determinism of an MCU. ST's STM32CubeIDE and HAL libraries make development relatively accessible.
- **Example Project:** A CNC controller, an oscilloscope, a flight controller for a drone, a user interface with a touchscreen.

Teensy 4.1 (with NXP i.MX RT1062)

- **Core:** ARM Cortex-M7 @ 600 MHz
- **Key Features: Extremely high performance-to-cost ratio**, native Ethernet, USB-Host, and an extensive audio library.
- **Best For:** High-performance data acquisition, real-time audio processing (synthesizers, effects), and projects that need a lot of "number crunching" in a small form factor.
- **Why it's a top pick:** It's a performance beast. The community-driven libraries (like the Audio library) are phenomenal and can save you months of development time.
- **Example Project:** A polyphonic synthesizer, a high-speed data logger, a vision system, a complex lighting controller.

4. For Ultra-Low Power & Battery-Powered Devices

When your device must run for months or years on a small battery.

STM32L4/L5 Series (STMicroelectronics)

- **Core:** ARM Cortex-M4/M33
- **Key Features:** Excellent ultra-low-power performance with a rich set of peripherals. The L5 adds ARM TrustZone for security.

- **Best For:** Wearables, sensors, medical devices, and any battery-powered application that still needs a good amount of processing power and I/O.
- **Why it's a top pick:** ST provides best-in-class power management modes and tools to accurately measure and optimize power consumption. The combination of performance and low power is hard to beat.

nRF52/nRF53 Series (Nordic Semiconductor)

- **Core:** ARM Cortex-M4/M33
- **Key Features:** Exceptional power efficiency for Bluetooth applications.
- **Best For:** The definitive choice for low-power Bluetooth devices. If you're using BLE, you should strongly consider a Nordic chip.
- **Example Project:** A fitness tracker, a smart tag, a wireless sensor that reports data once per hour.

5. For Simplicity, Legacy, and Massive Ecosystem

ATmega328P (The Classic Arduino MCU)

- **Core:** 8-bit AVR
- **Key Features:** Simple, robust, immense number of tutorials and libraries.
- **Best For:** Beginners learning electronics, very simple control tasks, and when you just need a basic, reliable digital I/O controller.
- **Why it's a top pick:** It's not "the best" in terms of performance, but its ecosystem is unmatched for simplicity. For a quick proof-of-concept or a simple project, it's often the fastest path to a working result.

How to Choose: Key Decision Factors

1. **Connectivity:** Do you need Wi-Fi, Bluetooth, Ethernet, LoRa, or Cellular? This often narrows the field the most.
2. **Performance:** How many calculations per second do you need? Do you need a DSP or FPU? 8-bit, 32-bit, or a multi-core monster?
3. **Power Consumption:** Is it wall-powered or battery-powered? If battery-powered, how long does it need to last? This dictates if you need an ultra-low-power MCU.
4. **I/O Requirements:** How many pins, ADCs, DACs, PWM channels, UARTs, SPIs, I2Cs do you need?
5. **Cost:** For a one-off project, cost is rarely an issue. For mass production, saving $0.50 matters a lot.
6. **Development Ecosystem:** How good are the manufacturer's tools? Is there an active community (Arduino, PlatformIO)? Good documentation and libraries drastically reduce development time.

7. **Security:** Do you need secure boot, hardware encryption, or TrustZone? This is critical for commercial products.

Final Recommendation Summary

- **Just getting started / General Purpose: Raspberry Pi Pico (RP2040)**
- **Wi-Fi & Bluetooth IoT Device: ESP32-S3**
- **High-Performance Computing / Graphics: STM32H7** or **Teensy 4.1**
- **Ultra-Low-Power Bluetooth: nRF5340** or **nRF52840**
- **Simple Sensor/Actuator Control (no wireless): STM32G0** (modern) or **ATmega328P** (classic)

Start by defining your project's *must-have* features (especially connectivity and power), and the list of excellent choices will become very clear."

[2a] **Case Study 2 Prompt**: "How would you go about making a GUI coded in Python language?"

[2b] **Case Study 2 ChatGPT Results**: "Excellent question — building a **Graphical User Interface (GUI)** in Python can be surprisingly approachable, thanks to several powerful frameworks available. Let's go over the main concepts, best tools, and step-by-step guidance so you can get started effectively.

🧩 1. Choose a Python GUI Framework

There are several GUI libraries to choose from, each with different trade-offs depending on your goals (simplicity, performance, looks, or cross-platform support).

Here's a quick overview of the most common ones:

Framework	Description	Pros	Best for
Tkinter (built-in)	Comes with Python, simple and lightweight	Easy to learn, no install needed	Beginners, small tools
PyQt5 / PyQt6	Python bindings for Qt (a C++ GUI framework)	Professional look, cross-platform, feature-rich	Complex / commercial apps
PySide6 (Qt for Python)	Official Qt binding (same as PyQt but with LGPL license)	Modern, open-source-friendly	Production apps
Kivy	For touch-enabled or mobile-friendly interfaces	Works on Android/iOS, stylish	Apps, dashboards
Dear PyGui	GPU-accelerated, immediate-mode GUI	Very fast, modern look	Engineering tools, visualization
wxPython	Native look and feel	Mature, cross-platform	Desktop utilities
CustomTkinter	A modern wrapper for Tkinter	Sleek modern look, simple syntax	Student projects / quick GUIs

2. Basic GUI Example (Tkinter)

Here's a minimal working GUI to demonstrate how it's structured:

```
import tkinter as tk
from tkinter import messagebox

# Create main window
root = tk.Tk()
root.title("My First GUI")
root.geometry("300x200")

# Create a label
label = tk.Label(root, text="Enter your name:")
label.pack(pady=10)

# Create an entry box
entry = tk.Entry(root)
entry.pack(pady=5)

# Create a function for button click
def greet():
    name = entry.get()
    messagebox.showinfo("Greeting", f"Hello, {name}!")

# Create a button
button = tk.Button(root, text="Greet Me", command=greet)
```

```
button.pack(pady=20)

# Start the GUI event loop
root.mainloop()
```

🔹 **How it works:**

- You create a `Tk()` root window (the main app window).
- Add widgets (Label, Entry, Button, etc.).
- Define a callback function (`greet()`) triggered by a button.
- Start the event loop with `root.mainloop()`.

This structure applies to almost any Python GUI toolkit — you create widgets, define layout, and connect events to actions.

⚙️ 3. Example with PyQt6 (Professional Look)

If you want something more advanced and modern:

```
from PyQt6.QtWidgets import QApplication, QWidget, QLabel, QLineEdit, QPushButton, QMessageBox, QVBoxLayout
import sys

class MyApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("PyQt6 GUI Example")
        self.setGeometry(200, 200, 300, 150)

        layout = QVBoxLayout()

        self.label = QLabel("Enter your name:")
        layout.addWidget(self.label)

        self.entry = QLineEdit()
        layout.addWidget(self.entry)

        self.button = QPushButton("Greet Me")
        self.button.clicked.connect(self.greet)
        layout.addWidget(self.button)

        self.setLayout(layout)
```

```
    def                                                    greet(self):
        name                    =                  self.entry.text()
        QMessageBox.information(self,     "Greeting",     f"Hello,
{name}!")

app                     =                    QApplication(sys.argv)
window                              =                              MyApp()
window.show()
sys.exit(app.exec())
```

✅ **Pros**: You can design GUI visually using **Qt Designer**, then load `.ui` files into Python. This saves time for complex UIs.

🪄 4. Designing Efficiently

You can build a GUI:

- **By code** (as above), or
- **Visually**, using tools like:
  - **Qt Designer** (for PyQt/PySide)
  - **PAGE** (for Tkinter)
  - **Kivy Designer** (for Kivy apps)

The visual route is great for complex UIs with multiple windows, forms, and layouts.

🚀 5. Advanced Options

If you want to go beyond desktop:

- **Dash or Streamlit** → Build GUI-like *web apps* in Python for data visualization.
- **Flet** → Lets you create Flutter-style apps with Python syntax.
- **BeeWare / Kivy** → Create cross-platform mobile GUIs.

🧠 Tips for Your Project

- Start simple (Tkinter or CustomTkinter).
- Structure your GUI in **classes** for scalability.
- Use **separate files** for logic vs. interface (MVC or similar).
- Plan the user flow first (sketch the interface before coding).

- Test on multiple OSes if deployment matters.

“

[2c] **Case Study 2 Google Gemini Results**: “ Building a **Graphical User Interface (GUI)** in Python involves selecting an appropriate framework and then using its tools and syntax to define the layout and functionality of your application.

Here are the general steps and some popular options for frameworks:

1. Choose a Python GUI Framework

The most critical first step is selecting a library, as it dictates your coding style, the available widgets, and the final look and feel of your application. The choice often depends on your project's needs:

- **Tkinter:** This is the **standard Python GUI library**. It's built into most Python installations, making it the easiest to start with, especially for **beginners** or simple tools. Its main drawback is a somewhat dated appearance compared to modern interfaces.
- **PyQt / PySide:** These are Python bindings for the powerful, cross-platform **Qt framework**. They are excellent for creating **professional, complex, and modern-looking** desktop applications. They offer a rich set of widgets and tools, including a visual designer (**Qt Designer**) for drag-and-drop layout creation.
    - *Note: PyQt uses the GPL license for its free version, which has restrictions for commercial applications, while PySide uses the more permissive LGPL license.*
- **Kivy:** This is ideal for developing **cross-platform** applications that support **multi-touch**, often targeting mobile devices (Android/iOS) and desktop platforms. It uses its own drawing toolkit, so the interface looks the same everywhere.
- **wxPython:** This framework wraps the native GUI components of the operating system, creating applications that have a truly **native look and feel** on Windows, macOS, and Linux.
- **PySimpleGUI:** This simplifies GUI development significantly by wrapping other libraries (like Tkinter, Qt, wxPython) to let you create interfaces with minimal, highly readable Python code. It's great for **rapid prototyping** and simple utility tools.

2. Basic Steps for GUI Development

Once you choose a framework (let's use **Tkinter** as a common example), the process typically follows these general steps:

A. Importing the Framework

You first need to import the necessary modules.

Python
```
import tkinter as tk
```

B. Creating the Main Window

This is the root container for all other elements of your GUI.

Python
```
root = tk.Tk()
root.title("My Python GUI")
```

C. Adding Widgets

Widgets are the building blocks of the GUI, such as buttons, labels, text entry fields, and frames. You create an instance of the widget class, specifying its parent container (usually the main window).

Python
```
# A Label widget to display text
label = tk.Label(root, text="Hello, GUI!")
# A Button widget
button = tk.Button(root, text="Click Me")
```

D. Defining Layout

Widgets need to be placed within the window using a **layout manager**. Common managers are:

- `pack()`: Arranges widgets in blocks.
- `grid()`: Arranges widgets in a table-like structure of rows and columns (very popular).
- `place()`: Allows absolute positioning (less flexible).

Python
```
label.pack()
button.pack()
```

E. Adding Functionality (Event Handling)

You connect user actions (like clicking a button) to specific Python functions, known as **event handlers**.

Python
```
def on_button_click():
    label.config(text="Button Clicked!") # Changes the label text
```

```
button = tk.Button(root, text="Click Me", command=on_button_click)
```

F. Running the Event Loop

The final step is to start the main loop, which listens for events (like mouse clicks or key presses) and updates the GUI until the user closes the window.

Python
```
root.mainloop()
```

To see an overview of different Python GUI options and what they look like, you might find this video helpful: Choosing a GUI Framework / Tool Kit / Libraries - Python programming. “