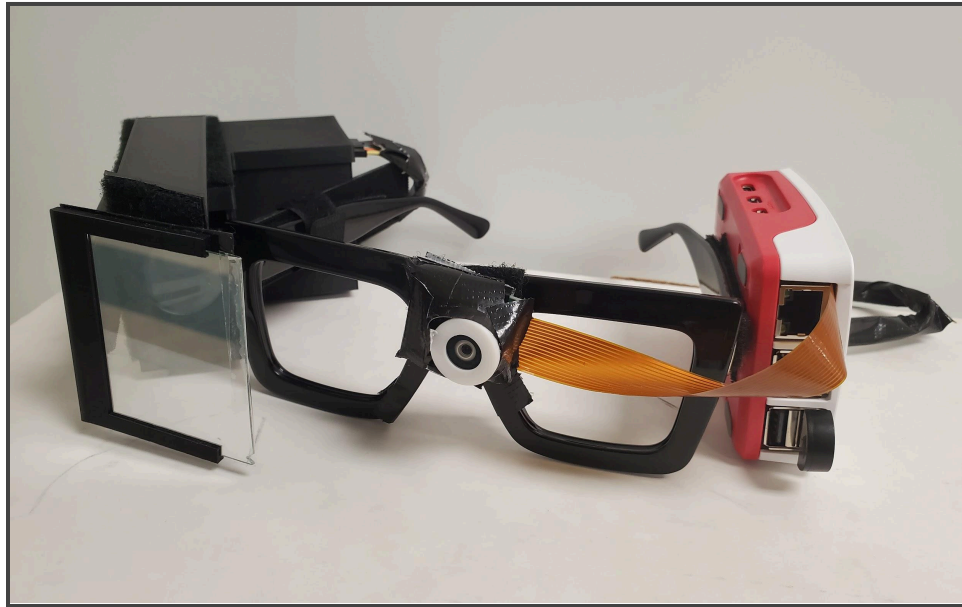


# Project I.R.A.S. (Integrated Real Time Assistance Spectacles): Speech-to-Text and Color Detection AR Glasses



*Department of Engineering and Computer Science and College of Optics and  
Photonics (CREOL)  
University of Central Florida*

*Dr. Aravinda Kar, Dr. ChungYong Chan*

*Senior Design 2 Project Paper*

Github Link: <https://github.com/BrianUmbrecht/Project-IRAS/tree/main>

Group 4 Interdisciplinary:

Carlos Acosta, PSE

Darlandie Moise, PSE

Kim Le, CS

Brian Umbrecht, CS

Reviewers:

Dr. Patrick LiKamWa

Dr. Piotr Kulik

Dr. John Aedo

# Table of Contents

- 1.0 EXECUTIVE SUMMARY..... 1**
- 2.0 PROJECT DESCRIPTION..... 1**
  - 2.1 MOTIVATION..... 1
  - 2.2 GOALS..... 3
  - 2.3 OBJECTIVES..... 3
  - 2.5 OPTICAL SYSTEM DESIGNS..... 4
  - 2.4 REQUIREMENTS AND SPECIFICATIONS..... 5
  - 2.6 HOUSE OF QUALITY..... 6
  - 2.7 SOFTWARE FLOWCHART ..... 7
  - 2.8 PROJECT HARDWARE DIAGRAM..... 8
  - 2.9 PROJECT SKETCH..... 9
- 3.0 TECHNOLOGICAL RESEARCH RELATED TO PROJECT..... 10**
  - 3.1 CAMERA SYSTEM..... 10
    - 3.1.1 Wide Angle Optical System..... 10
    - 3.1.2 Lenses for Camera System..... 10
  - 3.2 MICRO-DISPLAY TECHNOLOGIES..... 17
  - 3.3 OPTICAL SYSTEMS FOR AR PROJECTIONS..... 18
  - 3.4 OPTICAL COMBINER FOR AR DEVICES..... 19
    - 3.4.1 Geometric/ Array Waveguides..... 19
    - 3.4.2 Diffractive-Based Grating Configurations..... 20
    - 3.4.3 Thermoplastic-Based Combiners..... 24
  - 3.5 OPTICAL COMBINERS AND MAGNIFYING OPTICS COMBINATIONS..... 26
  - 3.6 EXISTING SIMILAR PRODUCTS..... 26
  - 3.7 OPTICAL TRADE-OFFS..... 28
  - 3.8 STRATEGIC COMPONENTS AND PART SELECTIONS..... 30
    - 3.8.1 Micro Displays..... 30
    - 3.8.2 Cameras..... 31
    - 3.8.3 Technological Comparison of Visual Sensors..... 35
    - 3.8.4 Microcontroller Units and Development/Flash Boards..... 44
    - 3.8.5 Microphones..... 49
  - 3.9 PROJECT SOFTWARE OVERVIEW..... 50
  - 3.10 OPERATING SYSTEMS..... 52
  - 3.11 PROGRAMMING ENVIRONMENT AND TOOLS..... 54
  - 3.12 IMPLEMENTING LIVE SPEECH-TO-TEXT OVERVIEW..... 57
    - 3.12.1 Capturing Audio..... 57
  - 3.13 SPEECH-TO-TEXT SERVICES..... 58
    - 3.13.1 On-Board Speech-to-Text..... 58

3.13.2 Cloud-based Speech-to-Text.....	60
3.13.3 UI.....	62
3.14 INTEGRATING SPEECH-TO-TEXT API.....	64
3.14.1 HTTP.....	64
3.14.2 HTTP Security and Privacy.....	65
3.14.3 HTTPS.....	65
3.14.4 Streaming.....	65
3.14.5 gRPC.....	66
3.15 OBJECT TRACKING AND COLOR DETECTION.....	66
3.15.1 You Only Look Once (YOLO).....	67
3.15.2 Region-based CNNs (R-CNNs).....	68
3.15.3 Single Shot Detector (SSD).....	68
3.15.4 OpenCV.....	69
3.15.5 Real-Time Processing.....	70
3.16 DATA COMMUNICATION PROTOCOLS.....	70
3.17 STORAGE AND DATA MANAGEMENT.....	72
<b>4.0 RELATED STANDARDS AND REALISTIC DESIGN RESTRAINTS.....</b>	<b>73</b>
4.1 ENGINEERING STANDARDS AND DESIGN CONSTRAINTS.....	73
4.1.1 Optical Standards.....	73
4.1.2 Software Standards.....	78
4.2 DESIGN RESTRAINTS.....	81
4.2.1 Project Requirement Restraints.....	81
4.2.2 Economic Restraints.....	81
4.2.3 Time Constraints.....	82
4.2.4 Health and Safety Restraints.....	82
4.2.5 Manufacturability and Sustainability Restraints.....	82
4.2.6 Social Restraints.....	83
4.2.7 Ethical Restraints.....	83
<b>5.0 COMPARISON OF CHAT GPT.....</b>	<b>84</b>
5.1 CHATGPT.....	84
5.2 ARIA.....	86
<b>6.0 HARDWARE DESIGN.....</b>	<b>87</b>
6.1 Detection Lens System.....	87
6.2 Text Projection Lens System.....	104
<b>7.0 SOFTWARE DESIGN.....</b>	<b>117</b>
7.1 Video Frames Capturing with the XIAO ESP32.....	119
7.2 OPENCV INTEGRATION.....	120
7.3 GUI DESIGN.....	121
7.4 DISPLAY PANEL CONNECTION.....	123
7.5 MOBILE APPLICATION FOR REMOTE CONTROL.....	124

7.6 ARDUINO IDE.....	127
7.7 INTERNET CONNECTION.....	128
7.8 CAPTURING AUDIO.....	128
7.9 TRANSMITTING AUDIO.....	128
7.10 USING A SPEECH-TO-TEXT SERVICE.....	129
7.11 SERVER.....	130
7.12 Switch to Raspberry Pi 5.....	131
<b>8.0 SYSTEM TESTING.....</b>	<b>134</b>
8.1 TEXT AND IMAGE DISPLAY TESTING.....	134
8.2 MCU TESTING.....	136
8.3 CAMERA SYSTEM TESTING.....	136
8.4 CONNECTING TO WI-FI.....	137
8.5 MICROPHONE TESTING.....	138
8.6 SPEECH TO TEXT TESTING.....	139
8.7 Color detection testing.....	140
<b>9.0 ADMINISTRATIVE CONTENT.....</b>	<b>141</b>
9.1 BUDGET AND FUNDING.....	141
9.2 BILL OF MATERIALS.....	141
9.3 PROJECT MILESTONES.....	142
9.4 Product Backlog.....	143
<b>10.0 CONCLUSION.....</b>	<b>143</b>
<b>11.0 APPENDICES.....</b>	<b>146</b>
11.1 REFERENCES.....	146
11.2 COPYRIGHT.....	158

## List Of Figures

Figure 1.1. Initial Two Separate Optical System

Figure 1.2. Final Two Separate Optical System

Figure 2. House of Quality Diagram

Figure 3. Software Flowchart

Figure 4. Hardware Flowchart Divided Among the Members

Figure 5. Rough Illustration of Project Design with Components

Figure 6. Positive and Negative Thin Lenses Immersed in Air ( $n = 1$ )

Figure 7. Positive and Negative Thick/Real Lenses Immersed in Air ( $n = 1$ ) with Their Estimated Principal Planes

Figure 8. Aspherical and Achromatic Lens

Figure 9. Partial Mirror-Based Optical Combiners

Figure 10. Color Dispersion Comparison of Diffraction in Diffractive Waveguide and Refraction in Prisms

Figure 11. Comparison of Color Dispersion in Single and Multi-Layer Waveguides

Figure 12. (a) Volume Bragg Grating; (b) Surface-Relief Grating; (c) Polarization Volume Grating

Figure 13. Sequence Diagram for Color Detection and Speech Transcription

Figure 14. Schematic of Initial Optical Design for CMOS Camera

Figure 15. Conventional Lenses and spacers Within the  $10^\circ$  FOV and the  $40^\circ$  FOV

Figure 16. Schematic of  $10^\circ$  FOV Optical Design for CMOS Camera

Figure 17. Schematic of  $60^\circ$  FOV Optical Design for CMOS Camera

Figure 18. Cross Section of  $60^\circ$  FOV Lens System on Zemax

Figure 19. Lens Data for  $60^\circ$  FOV Lens System on Zemax

Figure 20. Spot Diagram of  $60^\circ$  FOV on Zemax

Figure 21. Lens Data for  $80^\circ$  FOV Lens System on Zemax

Figure 22. Spot Diagram of  $80^\circ$  FOV on Zemax

Figure 23. Cross Section of  $80^\circ$  FOV Lens System on Zemax

Figure 24. Cross-section of Mutation Lens System on SolidWorks Color Coded and Numbered

Figure 25. Cross-section of the Mutation Lens System on Solidworks with the Labeled Parts within the System

Figure 26. The Overall Mutation Lens System on SolidWorks

Figure 27. Projection System Using Biconvex and Plano Concave Lens

Figure 28. Text Display Projection Light Path

Figure 29. Lens Housing Layout for Components

Figure 30. Proposed Text Lens System for 0.7-in Micro-display  
Figure 31. Final Lens Projection Design  
Figure 32: Solidworks LCD Housing  
Figure 33. Project I.R.A.S. Final Design  
Figure 34. One of the Colleagues Testing the Glasses' Color Detection...  
Figure 35. Use Cases Diagram  
Figure 36. Activity diagram for Live transcription  
Figure 27. Video Streaming Demo  
Figure 38. MicroPython and OpenCV.js Demos Operating on a XIAO ESP32S3 Sense  
Figure 39. Our GUI for Color Detection and Speech-to-Text  
Figure 40. Location of I/O pins on the XIAO ESP32S3  
Figure 41. Using the Blynk Mobile with Our Template  
Figure 42. Figma UI design  
Figure 43. Sequence Diagram for Audio Data Transfer and Server Interaction  
Figure 44: Activity Diagram for Blynk and Program operation  
Figure 45. Demonstration of Text Display Lens System  
Figure 46. Example of Successful Mobile Hotspot Connection  
Figure 47: Jira Scrum Product Backlog

## List of Tables

Table 1. Requirements and Specifications

Table 2. Comparisons of Micro-displays

Table 3. Types of Waveguides Comparison

Table 4. Thermoplastic Optical Properties Comparison

Table 5. Comparison of Existing Technologies to IRAS

Table 6. Micro-Displays Comparison

Table 7. Cameras Comparison

Table 8. MCUs and Development Boards Comparison

Table 9. On-board Speech-to-text services Comparisons

Table 10. Cloud-based Speech-to-text API Comparisons

Table 11. Optical Related Standards

Table 12. The Recorded List of Optical Components from 40° FOV Lens System

Table 13. Specifications of 3D Camera Lens System Components

Table 14. Lenses for Text Design

Table 15. Budget for Base Prototype

Table 16. Senior Design 1 Milestones

Table 17. Senior Design 2 Milestones

# 1.0 Executive Summary

The history of augmented reality can be dated back to the 60s. Since then, it has been used in the entertainment industry, our military for training, and even sports. Augmented reality technologies took a turn back in the 2010s when there was a big rise of augmented reality in the automotive industry and wearable technologies. Augmented reality has been a substantial part of our daily lives and has had many interactions. For instance, it has been used in the gaming industry for games like Pokémon GO on our mobile devices for apps like Snapchat, and in various other fields, such as healthcare and the military. As a result, people have started to gravitate more towards wearable technologies of augmented reality to meet their needs due to the rising trend of AR hardware (Poetker). As technology improves, augmented reality pushes the boundaries and offers creative solutions to integrate tech into our lives. This project will bring together the ideas of augmented reality and the need to give a hands-off experience while acquiring real-time data to assist those with difficulties hearing, color blindness, or even traveling.

Project IRAS (Intricate Real-time Assistance Spectacles) is the next step for smart glasses. It's a pair of glasses that will have two separate modes: one mode that will display real-time subtitles of who the user is talking to and the other mode that will display patches for the user who has trouble distinguishing specific colors around their environment. There are two main approaches to designing these glasses, one approach is to use an LCD to project the images and subtitles on the lenses and create a virtual image. The other approach is to integrate a see-through LCD on a lens so the user can both focus on the display and easily still see through without interfering with the user's reality. In either approach, the camera would be placed on top of the glasses between the lenses to continuously track the user's surrounding colors and provide live feedback. A microphone will also be placed on the glasses to listen to who the user is speaking with and live captioning what the person is saying. There will also be a speaker so when the camera detects a certain color it will warn the user with a buzz, giving the user extra precautions when traversing their environment.

## 2.0 Project Description

### 2.1 Motivation

Current technological trends for instruments show that people need assistance to have better face-to-face conversations. While some valuable devices help people converse with one another, more convenience would be excellent for those who have a busier lifestyle and still would like to retain a high level of professionalism.



Imagine, no matter where you are, you can have a simple one-on-one conversation and ease of navigating the world, even if you face hearing difficulties, color blindness, or even language barriers.

All the students are willing to challenge themselves to further expand these glasses in the educational and social setting. In addition, there aren't many articles that touch on the importance of making smart glasses multi-purpose, in this case, to accommodate color-blind users. We believe that no matter the disability or challenges the user may face, we strive to make the world a more accessible place.

This is an ambitious project, and all the students are willing to challenge themselves. Each member has their primary motivation for this project! The Optics students have a genuine passion for spatial augmented reality and how utilizing it can benefit the educational and social setting. The Computer Science majors have the desire to simplify the language barriers these glasses will offer and plan to integrate multiple languages to translate. They are motivated to improve their skill set by working with hardware, optics, software, APIs, and new languages. Additionally, if the popularity of AR glasses and other systems continues to increase, experience with this project could prove useful in future endeavors.

Numerous articles have shown the positives of allowing smart glasses in the industrial workspace or the concept design of applying subtitle-displaying lenses in theater for those who have a hard time listening to the characters on stage. Currently on the market, however, smart glasses like Google's HoloLens and XREAL AR glasses are adding multiple features that we believe are overcomplicating the user's experience.

The target demographic for this project would be those who are hearing impaired, color blind, or those who are constantly traveling. Even though these glasses would suit everyone, they will significantly benefit those with hearing trouble and allow better face-to-face communication. For those who struggle with multiple languages but want to travel the world, these glasses can be programmed to translate the native language the speakers are interacting with. This is a great attribute for business people, especially when they're attending conferences and don't have the time to learn foreign languages. Even those with color blindness can use the camera feature on these glasses to receive warnings if a prominent color they can't see is within their vicinity. This allows the user to better grasp their surroundings and not have to be as cautionary on the go.

We are taking the approach for these smart glasses to incorporate less complexity into the design and expand more on simple social connections. Not many glasses currently on the market solely focus on face-to-face interactions. By making these glasses multipurpose, we're only focusing on simplifying people's day-to-day tasks. AR and audio feedback are also fewer taped connections on the current market, and with our project focusing more on these

connections, we're making these glasses more accessible to everyone. In addition, there aren't many articles that touch on the importance of making smart glasses multi-purpose, in this case, to accommodate color-blind users. We believe that no matter the disability or challenges the user may face, we strive to make the world a more accessible place.

## 2.2 Goals

Our main overall project goal is to create a functional pair of glasses that will be able to display the text onto the lens, be capable of spotting distinct colors, and warn the colorblind user of their surroundings. The optical design approach would affect how the hardware would be created and this has a domino effect on how the software will be integrated with the design.

On the optics side, our goals would be to make the lens systems as simple and minimal as possible, only relying on a few key lenses to enhance the camera's wide-angle scope as well as making sure the LCD imagery can be seen on the glasses' lenses vividly. The shape of the lens system for the camera would provide little concern as the main concern is to make sure the camera lens system works. The same perspective applies to the near-eye display for the lens and LCD, as functionality would be prioritized.

After achieving proper calibrations of the lens designs for both the near-eye display and camera, the advanced goal would be to enhance the focusing of the camera and image positioning for the display. The camera would be programmed to focus with respect to the lens system so the user can focus on certain colored objects in multiple depths of fields. Once the camera detects certain colors, the MCU board processes the input of the camera and positions the outline of the color on the near-eye display with the LCD. By providing live feedback, the user can easily follow a conversation with the text display or determine the colors within their surroundings depending on the mode being used.

Once the camera and display have both achieved their primary purposes and enhanced features, then the stretch goal we have for the overall project is giving the options for remote customizations, rechargeable capabilities, and compacting the entire system just enough so the user can easily wear them. This would involve creating a 3D printed frame for each optical design and hardware component so they can latch onto each other with ease and with a less bulky nature. Additional stretch goals would be eliminating the need for an internet connection, using bluetooth to control the device, and improving the color detection under different lighting conditions.

## 2.3 Objectives

Our main objective for this project includes designing smart glasses to make them more accessible for all users. It will provide a wider lens for those with glasses, have access to more than one language for translation, and the main types of color blindness. Designing these glasses will be quite a challenge, but we will be taking steps researching current technologies on the market, and their strengths and weaknesses to determine the ways we can improve our design. Each component needed will be extensively researched for its specifications and compatibility with one another for incorporating them into our design.

For the optics, we will be designing a lens system for the camera and the projection of the text from the LCD. We will purchase the best camera and LCD considering our budget and availability in the market. Once we have decided which lenses will be used for our design, we purchase the lenses after calculating the focal lengths and field of view that we require for our system specification. We will then be testing multiple variations of AR projections on lenses and wide-angle camera systems. If necessary, we may then redesign or calculate focal length, FOV angles, and depth of field for our proposed design to line up with our project's specifications.

The computer science majors will be researching different operating systems, various speech-to-text application programming interfaces, and current speech-to-text services. They will also be researching ways for object tracking, color detection, and capturing audio. This will help them determine which systems to utilize and how they want to proceed with the project and design their code. They will program the glasses to display live speech-to-text and color detection with object tracking. Their first objectives will be getting one language and set of colors to work, as well as customizing how the information is displayed. Later objectives would include multi-lingual support, a wider range of color blindness detection, and improved display.

Each member will be responsible for ensuring that the glasses can fully perform and sustain processing power for the programs to run. Our design will be lightweight, low-cost, power-efficient, and user-friendly. Since there are quite a few added features to these spectacles, the main challenge for this project is to be able to implement all these important pieces into one small design. In addition, power usage and power consumption will be another constraint of the design of this project. Another problem we may face is how expensive this project may be. Even with proper budgeting from all members and carefully selecting inexpensive, but effective optical components, this project may come out more highly-priced than we'd like to fabricate.

## **2.5 Optical System Designs**

Each optical student on the team must create two separate optical systems within the project. One member will design a lens system either with the

approach of AR Projection or a See-Through LCD to display subtitles. The other member will design a wide FOV for the camera that will be placed on the glasses to analyze the speaker’s surrounding colors. The optical designs are illustrated in Fig. 1

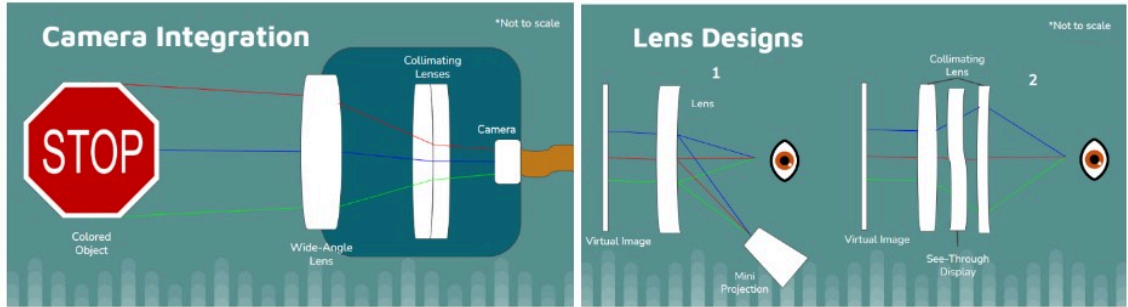


Figure 1.1. Initial Two Separate Optical System

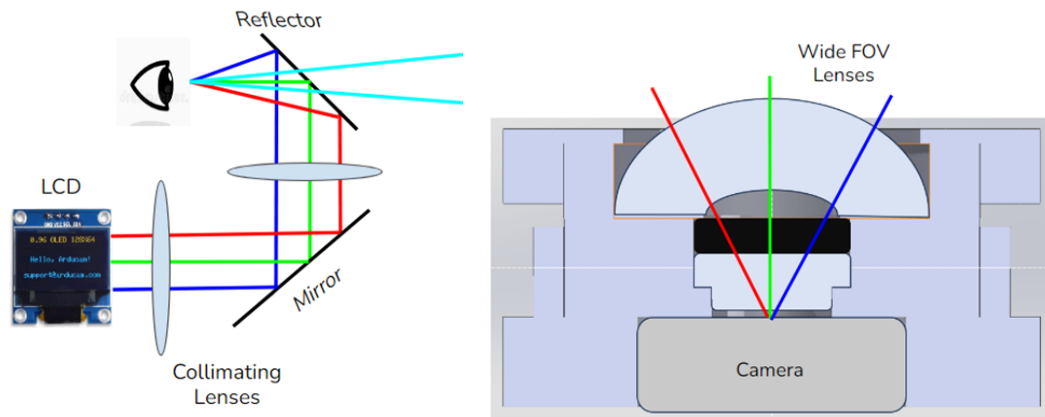


Figure 1.2. Final Two Separate Optical System

## 2.4 Requirements and Specifications

These are the base requirements we’re shooting for when making our AR glasses, to achieve our basic goals and create a functioning pair of smart glasses.

Table 1. Requirements and Specifications

Components	Parameters	Specifications
Antenna: Raspberry Pi 5	Transmission Range	50m-100m

Camera: OV5640	Capture Quality (Resolution and FOV)	Between 2-5 MP. FOV; ~60 - 80 Degrees
LCD TFT Display	Text and Color Accuracy	>80%
LCF TFT Display	Text and Color Latency	<2s
Frame of Lens Designs	Weight	<500 grams
Camera Lens System	Widen FOV	~80 Degrees Horizontally
LCD Projector	Transparency/Visibility	70% Transparent

## 2.6 House of Quality

The demanded quality and engineering requirements will be used as guidelines for building a prototype that will meet and possibly surpass the customer's demands. The house of quality is shown below to illustrate the design parameters being prioritized and define the needs and requirements of the customer.

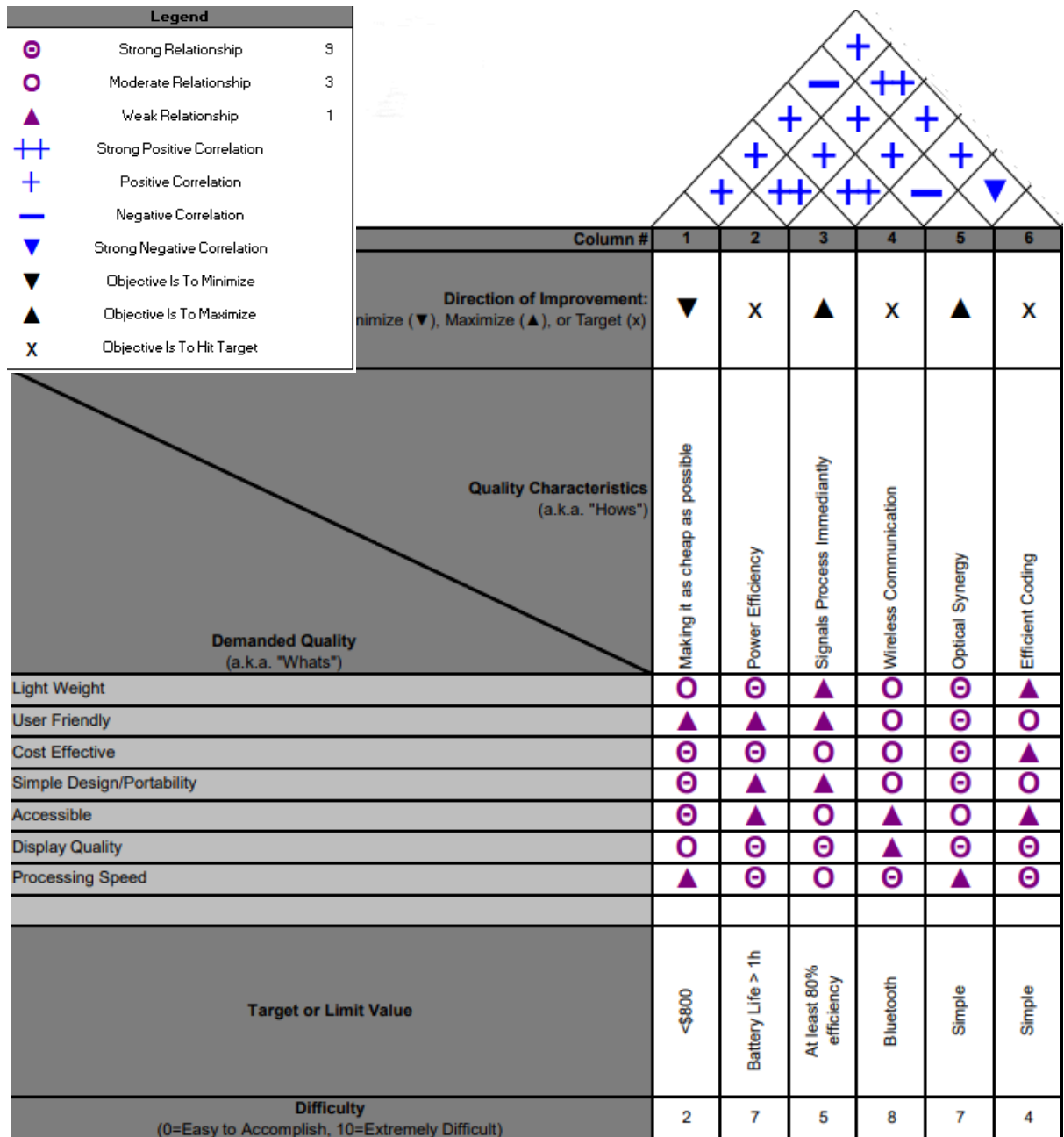


Figure 2. House of Quality Diagram

## 2.7 Software Flowchart

The following flowchart shows our system for collecting information through the sensors, processing it, and displaying the output.

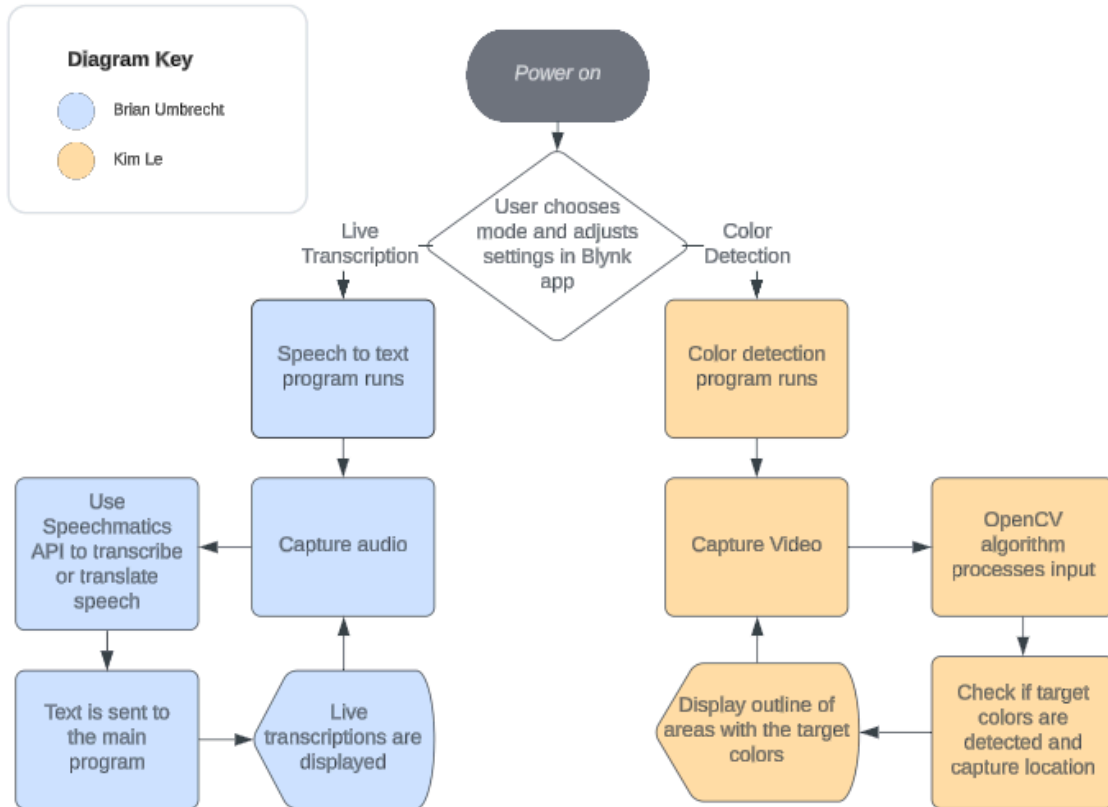
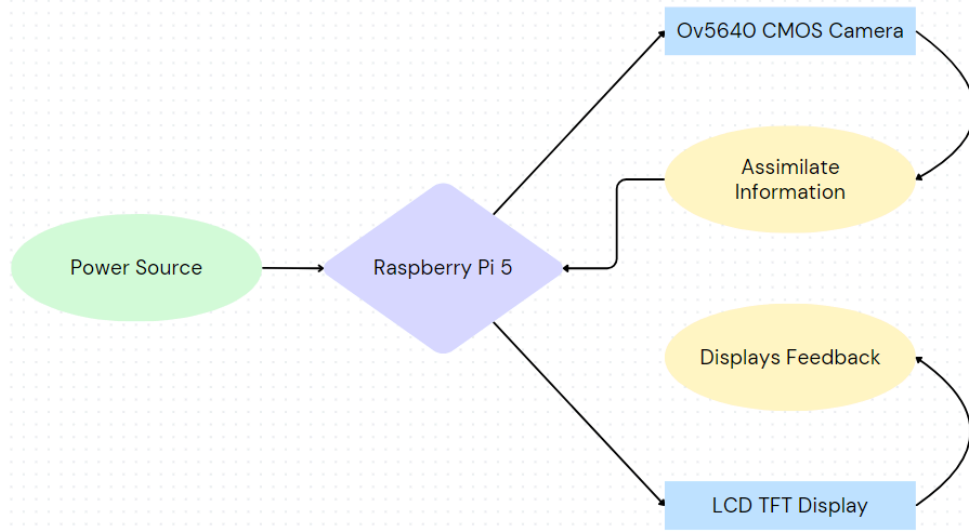


Figure 3. Software Flowchart

We may change modes, between the camera and microphone, using a switch or we may also create an app to handle this. This also goes for the language recognition and colorblind mode. Initially, we will get English and a certain set of colors to function correctly, and then work from there. We plan on using Google APIs to implement live speech to text and we'll also have to design a custom GUI to display the text. Color detection and object tracking could be implemented with software that the camera supports, computer vision libraries, and SDKs. All of this will be heavily dependent on the chosen components, especially the main processing board.

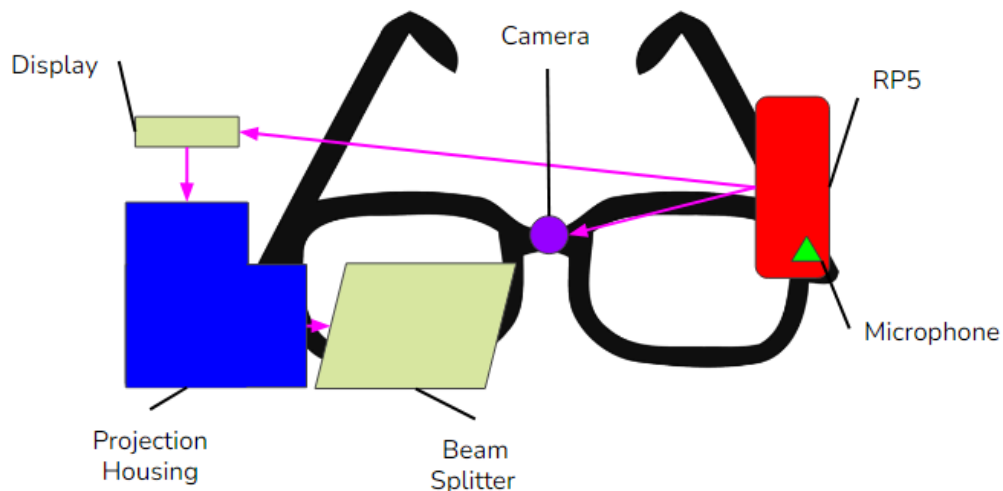
## 2.8 Project Hardware Diagram



*Figure 4. Hardware Flowchart Divided Among the Members*

The Optics students have the responsibility to configure the optical designs of 2 separate systems, one for the subtitles and patches on the lenses and the other for the camera to properly map the user's surroundings. The electrical engineering student has the critical task of making sure the whole project runs sufficiently with enough power. Computer Science students must utilize hardware and software to display live captions reliably and indicate the presence of certain colors.

## 2.9 Project Sketch



*Figure 5. Rough Illustration of Project Design with Components*

As mentioned, the current design approaches are augmented reality spatial projection or see-through liquid crystal display to display the closed captions or



notifications for the pair of smart glasses. With the addition of the camera, it collects data from the user's surroundings and gives notice of colors. All of these hardware components come together in a beautiful accumulation of both elegance and power. We will hopefully be 3-D printing the glasses base with the 3D printers in the TI engineering lab. We are very excited about this because we are enabling ourselves to be able to use a resource at the University that we have been excited to try for a very long time.

## **3.0 Technological Research Related to Project**

This section will mention all the research done related to our project and all the related parts for designing each system and selecting components.

### **3.1 Camera System**

This section will talk about the different parameters that will be used in selecting the camera and lenses to design a lens system with a wide field of view.

#### **3.1.1 Wide Angle Optical System**

Background Research and Parameters: The approach for this project will include two main optical designs mentioned before Augmented Reality projection and wide viewing camera. Not considering the electronics used for both optical designs, excluding the micro-OLED and camera, these systems will incorporate collimating lenses, mirrors, and beam splitters. The exact focal length of the lenses and size of the beam splitter will vary with the overall design of our project, but one thing is certain: we must keep the size of the project down to the most minimal state possible. With optical design, there is a crucial limit to how far you can minimize the lens system if only using conventional optics. There has been a rise in meta optics and synthetic lenses that broke this minimal limit; however, budgeting is limited for our senior design project. It is up to the PSE students to not only make sure their optical design works well above the minimal specifications, but we must also make sure to retain that threshold while minimizing our optical design to fit comfortably on a person's head. Further simulations of optical systems and parts searching will be required to smoothly transition our optical designs from the scheming phase to the testing phase, and it all starts with our background research on both optical designs.

#### **3.1.2 Lenses for Camera System**

Researching the wide-angle camera system is very general as many, if not all, camera lenses from phones to telescopes have built-in wide viewing lenses. The challenge however is creating a small enough lens design that will accommodate a wide enough Field of View (FOV) and depth. The human eye can view up to 150 degrees in a vertical arc and 210 degrees in a horizontal arc [31]. The design approach we will be making for these wide-angle viewing lenses is to make the camera mimic the viewer's binocular vision, which is the user's mid-peripheral vision that accompanies a 120-degree horizontal arc [31]. This will allow us to broaden the user's surroundings and let the user be more aware of catching colors they may miss but not too broad for the camera to catch colors that even the user cannot see from their outer peripheral vision. Another objectual challenge to consider is the Depth of Field (DoF), which allows the camera to focus on objects at certain distances. However, this consideration would be more important if we were utilizing object permanence with the camera, but we are only considering color recognition which does not require heavy usage of aperture and DoF modifications. While DoF is an important value to consider, it does not hold enough importance to the Angle of View, which is the FOV of the camera. This camera with the added wide lens system will function as a third eye to the user and give them another caution to see colors they cannot perceive in social situations.

Getting the basic design idea down for the wide-angle lens system, the next approach would be to start simulating the lens setup. We must treat the light that would be captured by the camera like rays, and with these rays, we will find the best design for our system. There are multiple approaches, however, but the realistic idea is to minimize the number of lenses within the system. From there we need to maximize the number of rays that would hit the focal point of the camera while minimizing the size of the lenses.

Looking into the field of geometrical optics, how exactly do these lenses contain the ability to bend light for sensors and even people to see? Still making light as rays to have an easier time treating the physic calculations, we apply certain equations to work with each lens specifications like its radius of curvature, focal length, the object and image's height, and more. Many of these calculations start by deriving from the *Gaussian equation* which is as follows:

$$\frac{n'}{s'} - \frac{n}{s} = \frac{(n'-n)}{R} \quad (1)$$

The Gaussian equation, created by the physicist Carl Friedrich Gauss, has been derived from all other equations for optics and sets the foundation for calculating optical properties [32]. From its equation the variables are as follows: “n” represents the glass's index of refraction (which is the medium value of the glass that will bend the light), “n” represents the environments' index of refraction, “s” and “s” represents their respective distances from each glass surfaces to either

the object or image, and “R” represents the radius of curvature, which is the curvature of the lens [33]. Deriving from the Gaussian equation (1) we can find the focal length for both sides of the lens which can be written as:

$$\frac{1}{si} - \frac{1}{so} = \frac{1}{f} \quad (2)$$

With “so” representing the object’s distance from one side of the lens, “si” representing the image’s distance on the other side of the lens, and “f” representing the focal length of the lens [33]. This equation applies to thin lenses however when the thickness of the lens is negligible to the ray’s refraction across the glass medium. Taking the other side of the previous equation (2), we can find the main equation for thin lens focal length and when the surrounding index of refraction is equal to 1, we can derive this equation for more general uses:

$$\frac{1}{f} = (nl - 1)\left(\frac{1}{R1} - \frac{1}{R2}\right) \quad (3)$$

“R1” and “R2” represent the radius of curvature for each side of the lens and “nl” represents the index of refraction for the lens itself. Depending on the radius, the lens would contain positive thickness and negative focal length, concave shape, or negative thickness and positive focal length, convex shape, better seen as follows [33]:

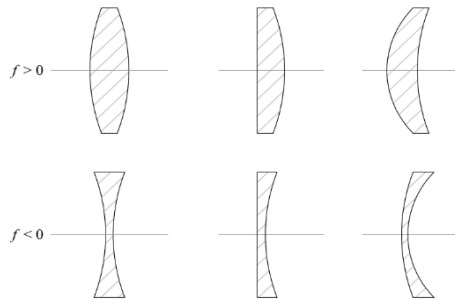


Figure 6. Positive and Negative Thin Lenses Immersed in Air ( $n = 1$ )

With thin lenses in mind, we can now add ray tracking to these thin lenses. When considering the object’s height to the image’s height, we must also factor in the magnification the rays will experience when crossing through the lens medium, given as follows:

$$m = \frac{hi}{ho} = \frac{si}{so} \quad (4)$$

Where the variables “hi” and “ho” (the heights of the image and the object) have rationing values to “si” and “so” (the distances of the image and object) and “m” is the magnitude of the magnification [33].

While these thin lenses are what is mostly calculated since they are the easiest to solve, we must consider the real world where thin lenses are not as thin as we hope. We take the previous equation (3) and expand on it to accompany real lens principles, which can be equated as follows:

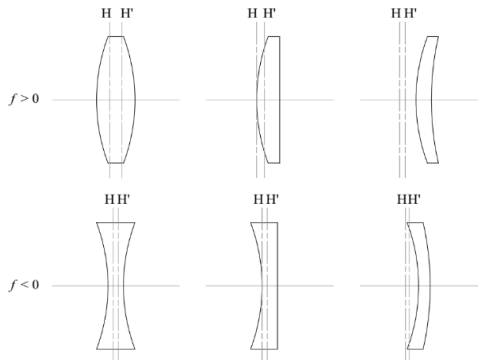
$$\frac{1}{f} = (nl - 1)\left(\frac{1}{R_1} - \frac{1}{R_2}\right) + \frac{(nl-1)^2}{R_1R_2} \left(\frac{t}{nl}\right) \tag{5}$$

Which is the thin lens equation for focal length and the addition of the lens' thickness "t." A distinct difference between thin lenses and thick (real) lenses is the accommodation of focal points and principal planes, which slightly vary the lens' focal length. The principal planes (classified as "H" for the object's principal plane and "H'" for the image's principal plane) position can be measured by finding where the secondary principal point is, which is the plane itself, and from the vertex of the refracting side's surface, equated as follows:

$$VP = - f \frac{(nl-1)t}{R_2nl} \tag{6}$$

$$V^*P^* = - f \frac{(nl-1)t}{R_1nl} \tag{7}$$

"VP" represents the main principal plane for "H" and V\*P\* represents the main principal plane for "H'" [3].



*Figure 7. Positive and Negative Thick/Real Lenses Immersed in Air (n = 1) with Their Estimated Principal Planes*

All these optical principles and equations have applied to only one defined lens, but what would occur if there were multiple lenses lined up? In comes the lens systems where the more lenses within the system, the more deviation the original principal plane's positions will be. The principal plane position corresponds well to the focal length of the lens system, whether it be for the image's position or the object's position [3]. The focal length of each lens within the lens system affects the total power that flows throughout this system, in both cases of thin lenses and real lenses. Switching gears, we have mentioned how a single lens or an array of lenses affects the flow of power, however, there is another fact that affects the flow, and it resides in all optical devices, even your eyes. To truly capture an

object image, we must regulate the flow of energy that emits from said object that allows the image to be created. Another way of saying this phenomenon is to regulate the intensity of the image from the object being captured. To regulate the intensity, you must apply a type of blocker that would mitigate the amount of light flowing through the lenses as the area of the lens affects the amount of energy reaching the image. This is what we refer to in the business as an aperture stop [3]. The aperture plays a crucial role when characterizing lenses (or even mirrors in some cases) as it affects the f-number quantity stated as follows:

$$f/\# = \frac{f}{D} \quad (8)$$

Where “D” represents the diameter of incoming light and “f” represents the focal length of the lens system. This is a crucial factor to consider before creating our optical system as we do not want to oversaturate our camera system with our proposed optical design.

As we mentioned before, the Field of View is a key factor to consider for our optical design, but there are a few optical principles that back up this specification. For instance, the marginal ray goes from the object’s base, hits the very edge of the aperture, and lands on the image’s location. Another important ray to consider is the chief ray, which goes from the object’s height right through the lens system (which includes the aperture) and lands at the tip of the image, giving the image its height [3]. These two rays determine the object’s overall shape throughout the entire lens system and can be utilized to perform quick calculations. A prime example of the usage of these two rays is by determining the field stop (where the image is only viewed in a certain region) of the lens system, which in turn helps us find the angular size of the image given as follows:

$$\tan(\alpha') = h'_{max}/l_{ex} \quad (9)$$

With “ $\alpha'$ ” representing the angle between the chief ray and the optical axis, “ $h'_{max}$ ” represents the image’s maximum height, and the “ $l_{ex}$ ” represents the distance between the exit pupil ( $P_{ex}$ ; determines effective exit base of illumination for image) and the image. The calculation for the angular size,  $2|\alpha'|$ , gives us what we have been looking for, the Field of View of the entire lens system.

Now that we have the general background of geometrical optics that correlate to our wide-angle camera lens system, we should go over how this design will act as our third eye for the user. A regular human eye, settling just with the schematic representation due to the many variations out there, contains spherical surfaces that can be easily simplified into thin lenses and an image plane. For basic understanding’s sake, we will consider the Gullstrand–Emsley schematic eye (assuming there are no errors) to understand the eye’s functions. Starting

from the eye fixing an object from a certain distance away, the eye's crystalline lens deforms and performs a technique called accommodation [3]. This allows the eye, as the object moves closer or farther from the user, to adjust the focal length to maintain the focus of the image on the retina. There is a limit to this however, especially for users who suffer from partial blindness either by age or genetics and must wear glasses, with the far point distance (FDP) and the near point distance (NDP) being the accommodation range [33].

While our camera could have the possibility to not reshape its lens to accommodate objects near and far, by adding an aperture to help focus the objects of our colors we program as images to the sensors can benefit our optical designs. Also, to better our Field of View for our device, we must design a lens system that should accompany no more than 2 lenses to get a wide enough view like the human eye, potentially no more than 120 degrees as stated before. With proper simulations with the provided background of the geometrical optics field, we are sure to create a small enough optical design capable of fitting on a pair of glasses and becoming the user's third eye. When searching for the right camera for our system there are only a few parameters we just consider (which will be stated later), but how exactly does a camera visualize and capture an image?

Two main visual sensors are widely used in all imagery applications: CCD and CMOS. A Charged Coupled Device (CCD) was first created in 1969 by George Smith and Willard Boyle at Bell Labs. This device works by taking in the optical signal of a photon incident on its matrix detector array of metal-oxide-semiconductor capacitors that charge with different shades of light [4]. These charges are converted and taken as electron-hole pairs being transported from one electrode to another until they reach the output node and the end of the array which is then converted to voltage then measured [34]. These CCD sensors were very distasteful from the manufacturers since they provided many disadvantages for any imagery usage. They were extremely expensive to make, very power insufficient, and provided poor operation for having limited processing and output nodes. It was not until the 1990s later commercialized for many imagery applications that the Complementary Metal Oxide Semiconductor (CMOS) improved the imaging processing process. Unlike CCDs, when a CMOS sensor detects a certain charge of light on its array, a single pixel captures that optical signal. Within this one pixel, or "cell," it simultaneously converts the charge into voltage then measured afterward [34]. This eliminates insufficient power consumption, reduces the expensive design with the sensors housing their circuitry, and has faster processing power for better frame rate capture. With the background of visual sensors established, it is obvious that the best sensor for our project is the CMOS sensor.

Cameras are composed of two main sections, the CMOS sensor and the lens system where the sensors will determine the kinds of lenses to be selected as well as the quantity. When picking what kind of CMOS sensor to use it is especially important, regardless of all the features, to compare the ratio of the

sensor's pixel array size and the individual pixel size itself. The greater the pixel size compared to the pixel array size ratio, the worse the quality of the captured image will become. However, if the size of the pixel is just as great but with an even greater pixel array, the image would turn out as higher quality. The vice versa can occur, where the smaller the pixel size compared to the grand area of the pixel array may produce not only a lower quality image but will consume more power [35]. This is a domino effect of errors that will lead to the CMOS sensor's poor signal processing, mismeasuring of the input values of each pixel, and the overall downfall of the camera itself eventually. The lens system varies in size depending on the purpose of the sensor.

An example is the usage of a high-powered CMOS sensor to image bacteria within a microscope compared to taking just a high-quality photo with a Canon camera. The number of lenses used to capture an image with the Canon camera is significantly lower than within a high-powered microscope [35]. With that being said, as this will be a lens system on a person's forehead, we must make sure that the number of lenses does not exceed the weight and size of our initial project idea. However, there must be a required number of integrated lenses for the CMOS camera of our choosing to compensate for it being our required optical design. Our biggest hurdle is to make sure that the best CMOS sensors we find should have a detachable lens housing, with the already established lenses not being our project's required FOV degrees. If our lens design has more than 3 lenses, then the entire system will be too clunky to function and produce too many aberrations. Too few lenses and our design would qualify our optics involvement requirement. Unfortunately, this means specially created lenses like aspherical lenses and achromatic lenses will not be considered for our final optical design. Aspherical lenses get their name by varying their lens sides with more of a free-form shape. This gives this lens the special attribute to significantly reduce spherical aberration, aka the misaligned focus of the image through the lens [6]. Achromatic lenses are a combination of 2 or even 3 lenses (face-to-face with their respective shaped sides) into one thick lens which is commonly used to fully focus the image and significantly reduce the chromatic aberration that occurs with regular lens systems that have a high focal length [36].

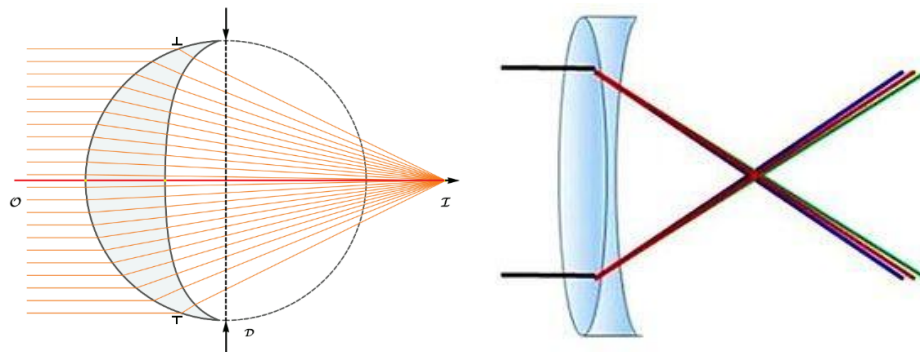


Figure 8. Aspherical and Achromatic Lens

These are the two special lenses, the Aspherical lens and the Achromatic lens (also referred to as the Achromatic doublet).

While these two lenses may significantly reduce our design's aberrations and the number of lenses for our CMOS sensor, it is the added requirement to make a proper optical lens system for the PSE students to pass.

## 3.2 Micro-Display Technologies

In this section, we will cover the numerous options of micro-displays currently in the market, and the advantages, and disadvantages of implementing these displays in our AR voice-to-text pair of glasses. Whether or not we will be using a transparent or non-transparent micro-display will be determined based on their compatibility with our system. We will mention some of the transparent micro-display options, as well as peak luminance, minimum pixel pitch, and ambient contrast ratio.

Transparent Micro-Displays: When it comes to near-eye displays, there are a lot of factors that ultimately are traded off, for example, resolution, form factor, brightness, field of view, etc. There are various types of micro-displays to be used in AR glasses, for instance, liquid-crystal-on-silicon (LCoS), organic light-emitting diodes (OLEDs), and LEDs. LCoS displays are one the main selections for micro-displays due to their high-resolution reflective display. The light engine that was considered in our first design was a transparent liquid crystal display. We believed that it would give the user access to their surroundings without the need for the use of an optical combiner or any additional magnifying optics, as it would serve both purposes. While their use would be quite beneficial, it has become more of a challenge with various issues that have arisen. Before our research, it was not apparent how inefficient the utilization of a transparent liquid crystal display (LCD) or any type of transparent display would be. They require a significant amount of power, where most of this power is converted into heat instead of light and can become quite a problem for the user's eyes. We were also unaware of the current patent for the use of transparent LCD for augmented reality and virtual reality equipment. We then turned to transparent OLED displays, but we ran into a similar concern that we would not be able to see our surroundings through the so-called see-through display. The transparent OLED is a lot more efficient than transparent LCDs with faster response. They have been primarily used in displays such as phones, watches, and even virtual reality headsets. The concern with transparent OLEDs is the fact that their lifetime diminishes substantially at an increasing level of brightness. Transparent OLED also has a current patent for augmented reality. While a transparent LCD or OLED would serve as both a micro-display and lens combiner, it would not be possible or beneficial to use as a near-eye display.



Micro Liquid Crystal on Silicon Display: LCoS is constructed using layers of liquid crystal on top of silicon. The standard layers of a micro LCoS contain a printed circuit board, a silicon chip or sensor, reflective coatings, liquid crystal, a transparent electrode, and a glass cover. The layers are used to ensure that the accurate amount of incoming light is being reflected and every component is aligned correctly. These displays are widely used in augmented reality and virtual reality displays for their brightness, high-resolution reflectivity, and contrast. They are also very low in energy usage and very small in size. One of the disadvantages of using LCoS is their requirement for backlight, which increases the thickness and the rigidity of the material.[6]

Micro Organic Light Emitting Diode Display: The use of micro-OLED displays has increased dramatically due to the thinness of the material and high brightness. The downside is that their lifetime does reduce at high brightness levels, unlike LCoS displays.[15]

*Table 2. Comparisons of Micro-displays*

Types of Devices	OLED	LCoS	LED
Backplane	Monocrystalline Silicon Wafer	Silicon	Silicon
Minimum Pixel Pitch ( $\mu\text{m}$ )	6.3	10	5
Pixel Density (PPI)	~4000	~2000	~5000
Ambient Contrast Ratio	~ $10^4$ :1	~ $10^3$ :1	~ $10^5$ :1
Peak Luminance (nits)	$10^3$ - $10^4$	$10^4$ - $10^5$	$10^5$ - $10^6$
Main Applications	EVF for Cameras; AR and VR Glasses	HUD, AR & VR, Holographic Projections	Wearable Displays

*Table 2 provides a few types of micro-display devices and some characteristic differences between one another, such as their peak brightness and ACR that are currently available in the market.*

### 3.3 Optical Systems for AR Projections

For the optical system to project the image onto the lens, we will need at least a couple of lenses that will be used to collimate and focus the image until it reaches the combiner to be visible to the user. Some of the optical combiners like diffractive waveguides that will be used to display the image can also act as a

magnifying optic. There are a variety of lenses that can be used as a collimating lens for the system.

Fresnel Lens: This type of lens is lightweight, cost-effective, and capable of focusing the light with a smaller profile than conventional lenses. They are also often used for image projection and as magnifiers. Traditional lenses can become very heavy compared to Fresnel lenses. These lenses are also made with concentric grooves, which help with the absorption of the system and keeping the same radius of curvature at every point of incidence. Higher grooves can help with maintaining high-quality images in the system, on the other hand, lower grooves have higher efficiency.

### **3.4 Optical Combiner for AR Devices**

An optical combiner in augmented reality glasses serves as a lens to overlay the information from the micro-display being used in the system onto the combiner while enabling the user to continue to interact with their surroundings. We will mention the different types of optical combiners that are currently used in the market on some augmented reality glasses and other potential materials that have great properties to be optical combiners in AR devices. We will also talk about the differences between these types of waveguides, and the challenges or advantages that they bring to this fast-growing market.

#### **3.4.1 Geometric/ Array Waveguides**

Geometric waveguides are simple and traditional combiners used such as reflective mirrors and prisms. The geometric waveguides are also known as a partial-mirror-based configuration, and they tend to have a large form factor with obstruction of the user's reality compared to the grating-based waveguides. Some potential designs using this type of waveguide can be in the figure below of (a)-(d). The images show the path of the light from the micro-display to the optical combiner, and finally to the user's eye.[16]

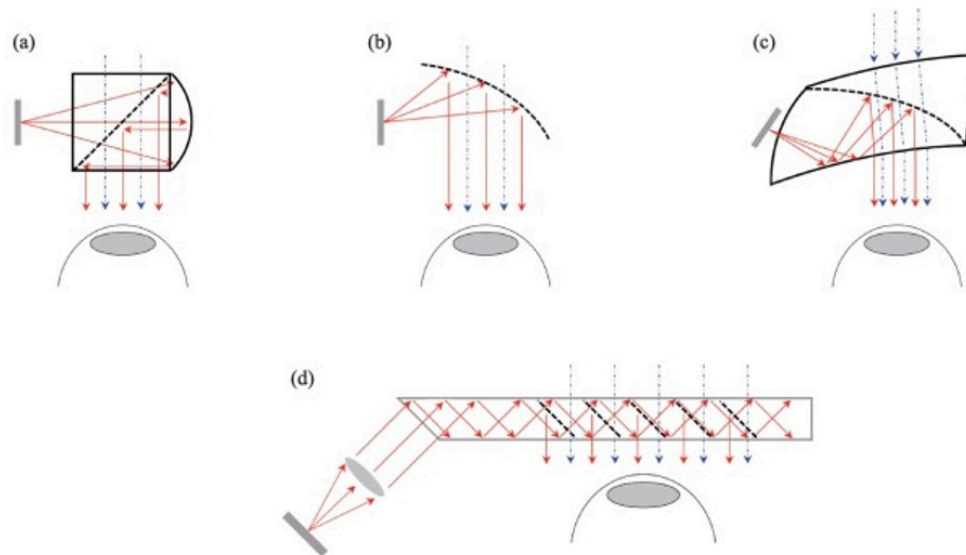


Figure 9. Partial Mirror-Based Optical Combiners

Reflective Mirror: Reflective mirror combiners are widely used because of their ability to reflect white without any loss, diffraction, or polarization states; therefore, making this system more power efficient because it is made of molded plastic. Using this type of combiner is cost-effective and can be used with reflective mirrors with any traditional coatings on the market.

Glass Reflective Mirror: It is a waveguide used which has several advantages such as great transparency with high power efficiency. The weakness of this combiner deals with the image quality having some non-uniformity. This issue happens because of the manufacturing of this material, in which tens of layers of coatings are placed on the glass with high precision and accuracy.

Free-Space/Prism Birdbath: For these types of combiners, concave mirrors are aligned on the optical system as collimators, which are then joined with the half-mirror to bring the image to the user's eyes. Using this technique has a high disadvantage of distortion in the image because of the freeform optics. This can be fixed by minimizing the image resolution and adding the eye box for a better-quality image.

### 3.4.2 Diffractive-Based Grating Configurations

Grating-based combiners have an intricate structure that makes them lightweight. The main issue with these waveguides is their manufacturability, due to the fact they are not as straightforward as some of the common optical combiners. The optical combiner in this design is traded with a planar diffraction grating, which is optimized by the nanostructures on a planar surface. They are challenging to create because this technique saves a lot of space, but it does require a lot more

precision to provide great image quality. The light for the display must propagate using total internal reflection through a very thin glass thickness between 0.3mm to 1mm. In these diffraction-based grating waveguides, the white light is being split, this is like color dispersion we've noticed before in prisms. In each grating, the different wavelengths will have a different diffraction angle. As displayed in the figure below, this effect is slightly different than what occurs in prisms because the light in the surface relief and volumetric holographic grating is being diffracted, whereas in a prism the light is being refracted.

Diffractive-based waveguides can be explained as a type of modulation of the refractive index of the material that is periodic. As displayed in the figure below, the different wavelengths will have a different diffraction angle in each grating. This effect is slightly different from what occurs in prisms because the light in the surface relief and volumetric holographic grating is diffracted, whereas in a prism the light is refracted periodically. This can be explained using this equation of

$$\sin \theta_{in} - \sin \theta_{out} = \frac{m\lambda}{n\Delta_x} \tag{10}$$

which includes the incident angle, the angle of diffraction, the order of diffraction m, n is the refractive index of the material, and is the change in x or periodicity. Equation (9) explains that the diffracted angle is directly proportional to the size of the wavelength; therefore, as the wavelength increases the diffraction angle also increases.

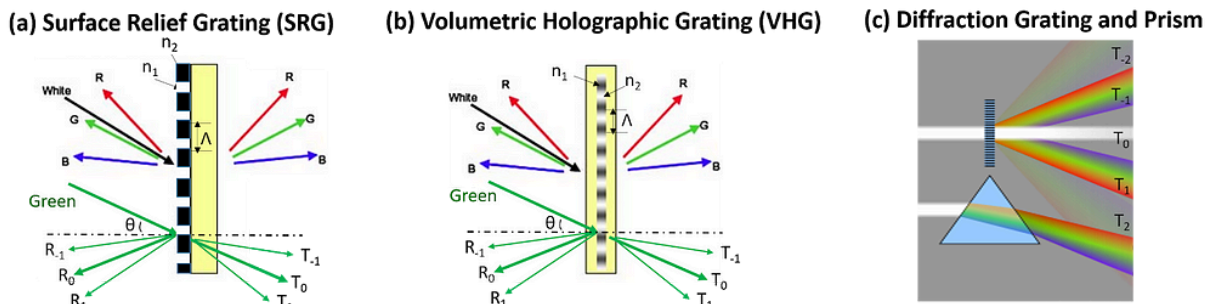
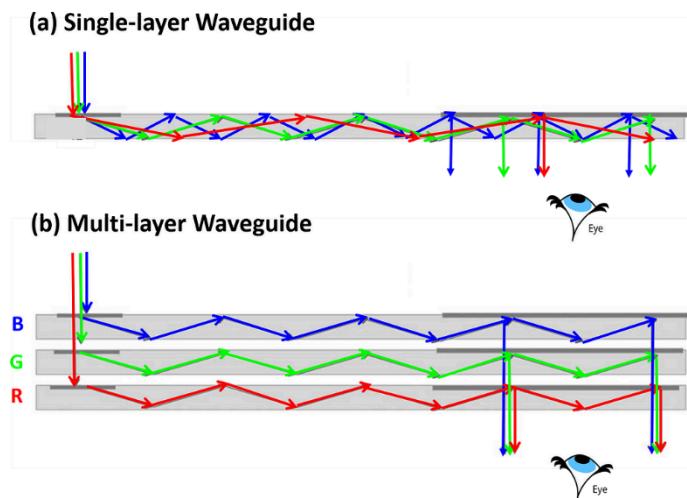


Figure 10. Color Dispersion Comparison of Diffraction in Diffractive Waveguide and Refraction in Prisms

Surface-Relief Grating Diffractive Waveguide: This type of waveguide is done by using the diffraction of the image, where the incoming collimated light travels through by total internal reflection. The light is then obtained by slanted gratings because the rectangular grating with +1 and -1 diffraction order wastes 50% of the light once it reaches the user's eye. The manufacturability of this technique is quite challenging, but it is quite an attractive form to the user because of its large field of view and the drastic reduction in the form factor compared to other combiners. This waveguide helps create a simple pair of glasses that closely

resemble normal-looking glasses, with a clear view of the user's surroundings. This technique is costly due to the need for the deep and slanted grating structure requirement for light travel, and this is not done in any type of manufacturing company today. Another issue with this type of grating is the "rainbow effect" that is caused by the coupling of light at an angle when it travels through the diffraction pattern of the material, this results in a non-uniformity of the image. This phenomenon occurs because the ratio of the reflected wave and the incident wave does not have the same amount of power or intensity once it hits the diffraction structure at an angle. Consequently, this leads to monochrome versions being used, which creates a hindrance for other applications that require full color. The waveguide does allow the user to have a larger field of view, but it is susceptible to the rainbow effect. The increase of the field of view also increases the nonuniformity of the display; it becomes obvious. Another issue that arises with this system is the "cyborg effect", which happens when the glasses are in active mode and the light from the lens completely blocks the user's eyes. This surface-relief grating is also very high in power consumption, and the extreme reduction in battery life is very troublesome in the augmented reality glasses.

One of the challenges that diffractive waveguides face as previously mentioned is color nonuniformity. The most effective way to correct this issue is to multi-layer for each wavelength to propagate through with overlapping or interfering without another as can be seen in Figure 11. This technique is also similar to the holographic grating that creates separate holograms for the wavelengths.



*Figure 11. Comparison of Color Dispersion in Single and Multi-Layer Waveguides*

Volume Phase Holographic Grating: This type of grating is similar to the previous example of surface-relief diffractive grating, but in this instance, a holographic grating is used to diffract the incident light. It is made using a laser-interfering

holographic technique, and the process is done by deposition onto a thin film and then the glass waveguide. The process itself is more controllable as it does not require the extra steps of cutting and molding necessary in geometric waveguides. It does require a high level of precision and sensitivity because of the input angles and wavelengths. Certain wavelengths of light are being reflected at an angle, which leads to the light losing its intensity. In this technique, the reflected angle is very important to make sure that we are not losing too much light and can still have a clear and excellent image. We also deal with the same rainbow effect that causes a non-uniformity in the image and the field of view is incredibly restricted. It is also an issue to achieve a full-color holographic grating because three separate holograms are needed for red, green, and blue. They also need to be “sandwiched” together for them to be able to create the image, which can also create a non-uniformity in the image. These issues of non-uniformity have posed a challenge for AR glasses, in addition to the high-power consumption and use of glass.

Polarization Volume Grating: polarization volume grading is another type of diffractive waveguide that is based on liquid crystals. There are two types of fabrication of PVGs, which include the alignment of coated liquid crystals with a chiral agent that contributes to the liquid crystal's helical structure. The other method uses a system that keeps track of the volumetric polarization field to form the PVG. This method also uses slanted gratings and three-layer panels in the same horizontal orientation with varied thickness for a full-color display. The slanted gratings help achieve a more uniform image and higher efficiency.

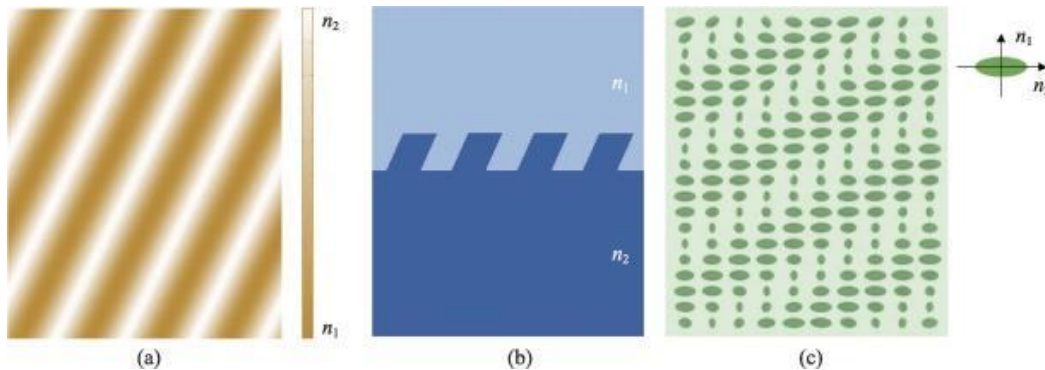


Figure 12. (a) Volume Bragg Grating; (b) Surface-Relief Grating; (c) Polarization Volume Grating

In Figure 12 (a), we can see the phase separation of the liquid crystals and the monomers that will later generate index modulation. The polarization volume grating shows the helical rotation of the liquid crystal, where the index modulation is birefringent.

Holographic Optical Element: Holographic optical elements are thin materials made from photopolymer, dichromate gelatin, and silver halide with two coherent laser beams that interfere with one another inside these materials. One of the advantages of HOEs is that the material does not diffract the image by itself compared to other glass materials. They are very transparent and can be placed on curved surfaces, which also maximizes the field of view of this combiner. Transmissive and reflective HOEs are two types of material that depend on the position of the image. These types of elements can be printed in thin films and combined with a spatial light modulator to create a more natural depth perception of the image. The thin films lessen the form factor of the glasses, but these films can only reflect one wavelength at a time. Thus, resulted in the need for three separate films to be “sandwiched” together to achieve a full-color display.

*Table 3. Types of Waveguides Comparison*

Waveguide Type	Geometric/Array Waveguide	Diffractive Waveguide	Diffractive Waveguide	Diffractive Waveguide
Optical Combiner	Reflective mirror; glass or plastic	Volumetric holographic grating; Holographic optical element	Surface-relief grating	Polarization Volume Grating
Input/Output	Reflective mirror or prism	VHG	SRG	PVG
FOV	55°	40°	56°	50°
Pros	Excellent image quality	2D pupil expansion	2D pupil expansion	Color uniformity

### 3.4.3 Thermoplastic-Based Combiners

Compared to plastic-based combiners, glass materials usually have better characteristics that help with reducing chromatic aberrations and even greater transparency. Chromatic aberrations are a common issue in imaging due to the way that the light goes through the lens and when the light refracts, each wavelength does not hit the focal plane concurrently. This phenomenon occurs because of the elements that the glass is made of, which impacts the way each wavelength travels; therefore, creating some sort of distortion in the image.[4] Plastic-based combiners have their own benefits since they are mass-produced, durable, and less susceptible to scratches. According to Team Xometry, they are “50% lighter, 90-92% clearer, more durable, and more affordable” in comparison to glass. They also make it possible to achieve a lightweight optical combiner that

will provide greater wearable comfort and resemble closer to a regular pair of glasses. [23]

Polymethyl Methacrylate: [24]Polymethyl methacrylate is a type of firm acrylic plastic that is used as an alternative to glass. It is also known as acrylic glass, for example, a common brand examples are Acrylite and Plexiglass. [13]PMMA is a transparent thermoplastic homopolymer that has been used for various applications such as lenses for glass, windows, automotive lights, home appliances, and even in many displays like LCD screens. PMMA has also been previously used to make a converging lens of 3 inches in diameter and focal length as a magnifying glass through 3-D printing. The PMMA material is an inexpensive, transparent, strong lightweight sheet of plastic in the form of glass, but it is shatterproof. The PMMA also has a few properties that can make it very helpful. Scratches on the surface can be easily removed, once polished. It can easily be cut with a laser cut for the appropriate size and it is also pliable once heated through thermoforming. The transparency also remains over time without being tinted and is generally one of the clearest plastics on the market. It is also safer to use unlike most plastics and does not contain bisphenol A, commonly known as BPA, which is a potentially harmful chemical.[2]

Polycarbonate (PC): Polycarbonate is another type of thermoplastic polymer. It is also strong, and some sheets are optically transparent, which are used in lenses. Unlike polymethyl methacrylate, polycarbonate materials are more expensive in comparison and sensitive to scratches, but it may be possible to reduce this by polishing the polycarbonate. They also contain BPA, which can greatly affect a person's health. There are three different types of polycarbonate sheets: GP, SL, and polycarbonate mirror. The clear GP sheets are known for their strengths against breakage and more for industrial work. Clear SL sheets are used in environments where the natural light can be damaging to the quality of materials and these sheets can provide protection. Polycarbonate mirror sheets are an interesting material that gives the effect of a mirror, a two-way mirror. PC materials also come in a variety of opaque forms to achieve the transparency needed. These materials are also lightweight and resistant to UV rays.[18]

Cyclic Olefin Copolymer (COC): Cyclic Olefin Copolymer is also another form of a thermoplastic copolymer, which is commonly used for its optical clarity and toughness. It is used for various optical applications because of its high level of transmission between 350 nm to 1100 nm, which is higher than polymethyl methacrylate and polycarbonate in comparison.

Table 4 below, shows the transmission percentage and refractive index of the potential thermoplastic materials that could be used as optical combiners for AR glasses. The transmission percentage is from the visible range of 350 nm to 750 nm. The spectral reflectance of the materials varies based on the wavelengths, whether the material is coated, and the type of coating used. For instance, the



reflectance of PMMA can be above 12% based on the concentration of other materials used in the sheet. [1]

*Table 4. Thermoplastic Optical Properties Comparison*

Material	Transmission %	Refractive Index	Starting Sheet Thickness (mm)
Polymethyl Methacrylate (PMMA)	89-92	1.49	0.075
Polycarbonate (PC)	86-91	1.584-1.586	0.762
Cyclic Olefin Copolymer (COC)	91	1.53	2mm

### 3.5 Optical Combiners and Magnifying Optics Combinations

Reflective Concave Mirror and Beam-Splitter Cube: One of the most straightforward approaches to creating an augmented reality pair of glasses is using a reflective concave mirror that helps project the display directly to a beam-splitter cube, which can be seen in Figure 9 (a). The reflective concave mirror behaves as a magnifying optic for the micro-display, but the use of this method does not allow for the display to show on a lens. The system is attached to the side of a pair of glasses and the beam-splitter is shown on the outside of the glasses in front of the user's eye; therefore, creating a large form factor for these types of glasses. This system can be seen in Google Glass, where the optical system is connected to the side of the glass and the beam-splitter glass is protruding from the side of the glass.

Birdbath Optics: It is like the previous optical combination because it utilizes a flat beam splitter and a spherical mirror. This method is known as birdbath optics due to the resemblance of a birdbath to a spherical mirror. The system setup includes a micro-OLED, with a lens system to collimate the light, which projects to a semi-transparent beam splitter, and then to the spherical mirror. The spherical mirror behaves as a partial mirror, which works great for a combiner and allows the user to see their surroundings, with the smallest amount of image distortion while simultaneously magnifying the image.[9]

### 3.6 Existing Similar Products

LetinAR: One of the augmented reality glasses currently in the market is LetinAR. They have been able to use a slim optical birdbath design called PinTILT with a great alternative waveguide that allows see-through quality. This is done by using the display at an angle, rather than total internal reflections that most AR glasses use. The optical combiner that they operate is produced through plastic injection molding, which requires less precision for mass manufacturing. They provide lens modules, which are lightweight and require very low power consumption. This module is quite unique because the lens and the display are combined, whereas the PinTILT uses a display that is projected from the top of the glasses. One of the most unique features is a magnetic lens frame clip that enables the user to add their own prescription lens.[12]

Pinna AR: Pinna AR is a pair of smart glasses that are still in the prototype phase and have been pitched to investors. It will have a voice activation AI assistant, with unique features such as audio cues and alerts when someone addresses the user.

ORA-2: These smart glasses are a power technology that give the user a lot of accessibility to numerous applications, such as maps, camera, calendar, Gmail, and more third-party applications. It is a powerful pair of glasses that can run its own application and connect through Bluetooth. They have a full-color panel, with a display brightness greater than 3000 nits, and a trackpad. The glasses utilized a monolithic plastic reflective mirror array as a combiner. This gives them the ability to have a thinner light guide while maintaining a large field of view and eye box for the user. With this method, they were still able to have great image quality without any type of uniformity and retain high power efficiency.

Mira: Mira is a type of headset that brings the user applications by flipping the lens down to activate AR and can receive tasks from remote supervisors without using their hands. It is not built for day-to-day use and does not resemble a regular pair of glasses. It is mostly used within the manufacturing industry, where it can be attached to a helmet or a hard hat. The technology uses a prism glass that is mostly made of plastic to display the virtual image, which continues to allow the users access to their surroundings. It also allows another user to join as a spectator through an iOS app to collaborate.[19]

Colorblind Glasses: Smart colorblind glasses do not currently exist in the market. These types of glasses work by blocking some wavelengths, for the user to be able to distinguish between other colors. They also do not work completely with everyone and are not proven to have long-term effects. They may affect or worse night vision due to the fact that they block some light from getting to your eyes.

Transcribe Glass: Transcribe glasses are not designed as a complete pair of glasses. It is a design that can be attached to a pair of glasses, which keeps the user comfortable with total flexibility. It is completely hands-free transcribes

conversations and saves the transcripts. It also gives the user control to format the captions and stream them from his/her phone.

XRAI: XRAI glasses are smart glasses that are designed for the hearing impaired. It is capable of following conversations and displaying subtitles. These glasses have a unique feature that allows the user to have a speaker ID function that allows them to see who they are talking to.

*Table 5. Comparison of Existing Technologies to IRAS*

	Project IRAS	Colorblind Glasses	Letin AR	Pinna AR	ORA-2	Mira	Transcribe Glass	XRAI Glasses
Color Detection	✓	☐	☐	☐	☐	☐	☐	☐
Voice-to-Text	✓	☐	✓	✓	✓	✓	✓	✓
Translation	✓	☐	✓	☐	✓	☐	✓	✓
AI Assistance	-	☐	☐	✓	☐	☐	☐	✓
Mobile App	-	☐	☐	✓	☐	✓	✓	✓

### 3.7 Optical Trade-Offs

Creating a pair of augmented reality glasses requires quite a few components for the optical design in order to project an excellent quality image for the use. We need to consider a few optical limitations that affect the user’s view of their surroundings and the sharpness of the image, which include the brightness and color of our display, resolution, modulation transfer function, eye box, form factor, and field of view.

Brightness and Contrast: It is very important to have the right amount of brightness for the display, as it will be projected onto mirrors and lenses. It will also be important to consider which display will be used, the brightness requirement will vary based on the optical combiner that will be used. For instance, a partial-mirror-based optical combiner can have up to 50% efficiency if a waveguide that is grating-based is used then it can be as low as 10%. The brightness of the displays is measured using nits, which is the same as candela over square meters (cd/m<sup>2</sup>). Nits’ measurement is different from lumens because it measures the total light output, and nits measure the total light at an angle. This is an issue with light from OLEDs for near-eye displays because they

have a large viewing angle of  $180^\circ$ . It makes it challenging to collimate the light for the user's eye. Once the brightness is considered, then we look into the ambient contrast ratio. This will ensure that the image being projected from the system remains clear and detectable. The minimum requirement for the contrast ratio is 3:1, for an even more sufficient system we need 5:1, and a ratio over 10:1 must be used for the absolute best contrast and finest projection. To receive the most optimal image for our display, we need to understand the difference between the optical combiners we use and the brightness of the micro display. We need to be able to consider the overall efficiency of the waveguide light we chose or blockage of ambient light and the transmittance percentage to determine if the brightness will be sufficient for the system. It might be beneficial in some designs to use a darker screen that can block some of the ambient light, this is why some current designs have a darker screen or lens that is similar to sunglasses. We must remember the light from the display will be passed through the collimating lens before reaching the combiner, to consider the amount of light that will be lost. Understanding how the brightness of our display and combiners can easily affect our system is critical to creating a high-quality image.

Modulation Transfer Function (MTF): Modulation transfer function is a metric used in order to measure the way a lens reproduces contrast as a function of spatial frequency/ resolution. It is a curve that helps us understand how optical aberrations may impact the performance of our set parameters, such as our image quality from object to image. The function can help measure the contrast, also known as the optical transfer function, based on a few parameters of the lens. This includes the working distance of the lens and the F-number, which is the ratio of the focal length of the system to the aperture diameter of the lens. The F-number ( $f/\#$ ) is a factor that controls the amount of light that will pass through the lens. The higher the F-number, the smaller the aperture for maximum light collection and the higher the depth of field. [7] The modulation transfer function enables us to measure some of the parameters such as the diffraction limit and the various heights that correspond to different positions. This is a way to define the modulation transfer function using a line test, where the contrast is measured from the object and then measured from the image after the lens system. Using these curves will help us compare them with one another to determine how well our image will be displayed. The contrast from each line is compared to see the percentage of contrast from the object and image. For instance, curves where each height comes closer together in one direction tell us that the image will have different contrasts even at the same frequency, this is caused by aberrations in the system. The function is a way to measure the quality of the image being produced from the performance of the optical system regarding the resolution and contrast. [8] A poor lens MTF performance may be as low as 20% contrast was reproduced in the image compared to as high as 90% contrast for an outstanding performance system. Another way to measure MTF is by using a line spread function. This method requires great resolution in order to measure the contrast between the pixels of the dark and light regions.

After using the Fourier transform of the image, we are able to determine the contrast of the section. This method has its own challenges due to pixelation noise that can affect the data, but it is another effective way to assess the system's MTF. [5]

Eye box: The eye box or exit pupil section of the glasses is the area on the glasses where the image is displayed with the right focus. Designing glasses to have a large eye can become challenging as it affects the user's field of view. A large eye box will also require a powerful light source with lots of brightness to overcome loss in the system and incoming light from the user's real world. It is highly desirable to be at least 4-8 mm in consideration of the human eye pupil size and support eye movements.

Field of View: The human eye's field of view is 130° vertically and 160° horizontally. Due to this fact, we decided to project the screen to be close to 25.4 mm which would have a smaller effect on the humans' eyes field of view since the overlapping region is around 120°. The small projection would provide a great display for the user without obstructing the rest of their view of the real world. When the light is propagating inside a waveguide with total internal reflection, the field of view can be widened by the position of the critical angle. This relates to the index of refraction of the material being used, for example, increasing the FOV of the system, then a high-index material should be used, more specifically greater than or equal to 1.9. This means that the critical angle will also decrease to support the larger FOV. The field of view is calculated using Equation 10:

$$FOV_{diagonal} = \text{sqrt}(FOV_{horizontal}^2 + FOV_{vertical}^2) \quad (11)$$

Form Factor: The form factor of the glass will be determined by the set of optics and display system used. For example, geometric waveguides are usually larger in form factor due to the prism forms and tilted mirrors. Reducing the form factor can be quite challenging, especially in our cases where the diffractive grating waveguides that have significantly smaller form factors are not accessible to us. It is also very difficult to make the display small, in order to have a large field of view as well as an eye box. This leads to a bulkier system that less resembles practical glasses due to the optics and added lenses.

## 3.8 Strategic Components and Part Selections

### 3.8.1 Micro Displays

High brightness in the micro-displays is necessary for quality image display and high contrast. A micro-display with a maximum brightness of 1,000 nits is not suitable for AR applications, as it is for VR applications. It may be more adequate

to be used for an AR system at 1,500 nits, but the display should have a minimal brightness of at least 3,000 nits to make sure the image remains intact with high contrast and nominal losses in transmission during projection. This leads us to a difficult decision regarding the cost and availability of displays to consumers. The micro-display we will be selecting is the Sony micro-OLED with 3000 nits because we must consider the loss in the system, especially with the use of lenses and mirrors that will be used to project the image. As displayed in Table 5, these were some of the options available on the market. We will be using BOE-FHD-039 as our display because of its size, brightness, and price. The ideal choice for the display would be the SNY023-640400 Sony display. Its max luminance would give us more range for the optical lens system.

*Table 6. Micro-Displays Comparison*

	SEY049-19 201080	Sony SNY023-64 0400	BOE-FHD-039	SID-039-XGA
Price	\$78	\$300	\$120	\$90
Size (in)	0.49	0.23	0.39	0.39
Resolution	1920 x 1080 Monochrome (Green)	640 x 400 Full Color	1920 x1080 Full Color	1024 x 768 Full Color
Max Luminance (cd/m <sup>2</sup> )	1800-3000	3000	1500	1000
Pixels per Inch (ppi)	4522	3183	5560	3314
Pixel Pitch ( $\mu$ m)	5	7	4	7
Thickness (mm)	1.48	-	4.75	1.2
Connection	MIPI DSI	I <sup>2</sup> C; MIPI DSI	I <sup>2</sup> C; MIPI DSI	I <sup>2</sup> C

### 3.8.2 Cameras

Here is the list of cameras that was researched and considered for the project:

*Table 7. Cameras Comparison*

Camera	Sensor Model	Resolution	Reasons	Price
Pixy2	MT9M114	1296 x 976	Modifiable	\$64.50

			Color Recognition Algorithm	
Mega 5MP SPI	OV05642	2592 x 1944	Open-Source Library for Multiple MCUs	\$34.99
5MP OV5647 Autofocus Camera	OV5647	2592 x 1944	High-Quality Cost-Efficient Camera	\$19.99
OV2640 SPI Camera	OV2640	1600 x 1200	Flexible and Small	\$25.99
IMX462 STARVIS Camera	Sony STARVIS IMX462	1920 x 1080	Ultra Low Color Detection	\$39.99
2MP MT9D111 Camera	MT9D111	1600 x 1200	Programmable Autofocus and Cost Efficient	\$9.99
8MP IMX219 Camera	IMX219	3296 x 2512	Autofocusing and Color Detection Modification	\$22.99
IMX519 PDAF&C DAF Camera	IMX519	4656 x 3496	High Quality Images and Autofocusing	\$24.99
OV5647 Fixed Focus Camera	OV5640	2592×1944	M12 Lens Housing	\$15.99
OV5647 Mini	OV5640	2592×1944	Versatile Dimensions	\$16.99

Camera				
MIPI Camera for RZBoard V2L	OV5640	2592×1944	High Processing Performance	\$49.99
OV5640 Camera Board C	OV5640	2592×1944	Flexible Lens Specifications	\$25.99
Grove	OV2640	1600 x 1200	Modifiable Color Recognition Algorithm and added MCU	\$25.99
IMX219-AF Programmable/Auto Focus Camera Module	IMX219	3296 x 2512	Wide FOV and Flexible Board	\$22.99
477P Camera	IMX477	4072 x 3176	High Video Resolution and Ultra HD Image	\$79.99
Wide Angle Camera Module	IMX708	4608 x 2592	High Resolution	\$ 36.99
Color Camera Module	IMX298	4656 x 3496	Hight Resolution and Compact	\$59.99
Ultra High-Resolution Autofocu	N/A	9152 x 6944	Ultra-High Resolution and Autofocusing	\$59.99



s Camera Module				
AMB82- Mini Camera	JXF37	1920 x 1080	IoT and Compact	\$24.99
MU Vision Sensor 3	MU Vision Sensor	640 x 480	Modifiable IDE Source	\$59.00
XIAO ESP32S 3 Sense	OV2640	1600 x 1200	IoT Capabilities	\$13.99

The main purpose of the optical design is to track certain colors at a set distance for the user to spot. All these cameras have a resolution that works to capture color, but the quality of that image varies from each camera model. For instance, the image quality from the Ultra High-Resolution Autofocus Camera Module will be significantly higher than the 477P Camera, regardless of the resolution of the sensors, each device is capable of capturing a wide range of colors which is the main focus of the optical design. This narrows down the list to the top 4 cameras: the Pixy2, the Grove, the XIAO ESP32S3 Sense, and the AMB82-Mini. Each of these cameras has good sensors attached to each board, providing at least 2MP of image quality. Each camera stands out from the rest, however, as each of them holds unique features. The Pixy2, as mentioned before, has a built-in color and object identification algorithm called the Color Connected Components (CCC) that is quite easy to modify through the API [59]. The Pixy2 can also be trained to detect certain objects with a distinct color, and each color segment can be adjusted to the user's liking. This will make adjusting for assorted color blindness a lot easier to implement and test. The Grove Camera also has an algorithm that can be trained to detect certain objects called YOLO. The difference between Pixy2's algorithm and Grove's algorithm is that the YOLO algorithm is not premade within the device and must be accessed from the GitHub site source. It is also not as color-focused as the CCC algorithm, being more focused on the recognition of objects and shapes, not exactly color. The Grove device, however, does have more built-in features that will come in handy for our project like the built-in microphone we can use for the hearing-impaired mode of our project. The XIAO ESP32S3 is like a minicomputer as it has wireless capabilities, high processing performance, and a built-in microphone, and can also use image processing but not to the scale of the project. It has many great functionalities, but not the main function we are looking for as it runs with a lesser version of the YOLO algorithm. The same can be said for the AMB82-Mini, with

lesser YOLO algorithm object detection, IoT capabilities, high processing, and can house its charge.

The final camera we used for our initial testing for our optical design was the XIAO ESP32S3 Camera. The XIAO ESP32S3 has not only adequate resolution but it is one of the few camera devices on the list that have a multi-disciplinary function that ties easily into our project, the built-in microphone. It also has an interchangeable camera that can switch from 2MP to 5MP compatibility leaving the project with a more flexible visual sensor. The camera also comes with Wi-Fi and Bluetooth compatibility and is one of the most high-performing boards on the market at the cheapest price too, with it just being \$14. Later on we then utilized the raspberry pi compatible camera, the OV5647, which we decided was the final camera to use for our project. The overall project must be compact to pass as a wearable device for any user, so this camera with its multipurpose board will fit perfectly for our design constraints.

### **3.8.3 Technological Comparison of Visual Sensors**

The main two components to obtain are the lenses and the camera for our optical design. The lens can be obtained by reaching out to potential sponsors for our project, either one of the available optical industries or professors around CREOL. As mentioned before our project could have the potential sponsorship to acquire optical lenses from companies like Newport MKS or resources from professors willing to pitch whether it be money or knowledge. For now, however, the lens system should be exemplified by MKS Instruments as that company is one of the largest industries for lens manufacture and would hold the standard lens specifications. Switching to the other main optical components, the CMOS sensor would be the preferred imagery sensor to consider for our project. However, different versions of CMOS cameras are integrated within a processing chip and are compatible with many MCUs that the EE of our team must choose from, which in turn affects how we approach our lens system and software development. The CMOS camera would have to work with the PCB and microcontroller provided and could be the defining fact for our entire optical system. The following is a list of CMOS cameras with their processing chips from their respective manufacturers, specifications, and the main benefits they may provide to our project.

Most of, if not all, the CMOS cameras listed are from the ArduCam site, the main site to go for visionary sensors. The site provides diverse types of sensors, main CMOS parts numbers, and an overall summary of each brand of camera's main application usage. Many of the provided sensors on the site come prepackaged with useful features like an Infrared filter, night vision, or auto/motorized focusing. While we will be considering some of these features, like the autofocus, and their prices, the main CMOS specifications we are looking for are the required resolution the camera can capture, the size of the pixels, and saturation sensitivity to capture colors. On top of that, there are specific parameters we are

looking at from these visual sensors like one of the main ones is the total pixel count of the visual sensor's area, the Megapixel count. The sensor's area to capture our live image will depend on the total amount of pixels, which is referred to as the Megapixel count. This factor also varies the device's price, with the higher pixel count meaning the device will be sold at a higher price. However, our camera does not need an extremely high pixel count, no more than 5 Megapixels (which means about 5,000,000 total pixels) for our project's application. Therefore, when selecting our final camera, we should factor out the visual sensors that contain 64 Megapixels. These factors also tie into the camera's dimensions, where the more pixels the sensors have the more housing the device requires, and the camera's processing, where the more pixels means the more processing power the processor will require (with some exceptions).

The **Pixy2** [37] is the top choice for our camera design. While this CMOS sensor MT9M114 has similar RGB color filters to the other cameras listed, it stands out from the list for coming with a pre-packaged color-based filtering algorithm that perfectly fits our project description. This device is capable of detecting color regardless of the object's size and distance, which will be great for modifying the camera's focal length to detect colors either near or far away. The sensor is also well-established in the field of visual applications for color recognition and entertainment. The CMOS sensor itself has a 1.26 Megapixel (MP) count with the image sensor dimensions being 1296 pixels horizontally and 976 pixels vertically, with the pixels being the square size of 1.9 micrometers by 1.9 micrometers. This means the CMOS sensor is 2.46 millimeters wide and 1.85 millimeters tall, which is a good area to work with as the image needs to be produced to hit the entire sensor and does not have to be so large to get captured by the sensors. This is important to note as that means there would be little to no aberrations produced when the lens system captures the selected colors.

The Pixy2 processing unit is also quite versatile, being that it is small, has quick processing, and with all its features it is not too expensive and quite easy to use. The Pixy2 sensor unit is compatible with a lot of commonly used MCUs like Arduino, Raspberry Pi, and BeagleBone just to name a few well-known units. The visual sensor as mentioned before is pretty small with the given dimensions being 42 millimeters tall, 38 millimeters wide, and 15 millimeters long, weighing only 10 grams. The visual sensor, regardless of its amazing software library and processing specifications, also has a FOV of 60 degrees horizontally and 40 degrees vertically. This kind of FOV is important to note as it will affect how I approach my wide-angle lens system. The visual sensor is also lightweight and small enough to sit on top of our makeshift glasses, so integrating this sensor into our lens system would be a welcome choice. From all the specifications of this visual sensor, the main takeaway is that this sensor comes pre-installed with libraries that are exactly what the project is supposed to be, and that's color identification. The CMOS sensor contains an RGB Bayer color array filter which allows the sensor to easily detect a wide range of color signatures. The

processing unit of the visual sensor contains malleable lines of color code we can modify and easily customize for whichever user decides to use the camera. All the positives outweigh the very few negatives this sensor has, which is the cost being \$64.50, but not all good sensors are cheap, making the Pixy2 visual sensor our number 1 choice.

The following sensors have their benefits but not as many as the Pixy2, for example, the 2nd best camera to consider is the **Mega 5MP SPI** [38]. This visual sensor, which currently does not have a datasheet attached but contains the OV05642 CMOS sensor, has an adequate resolution size being 2592 pixels wide and 1944 high with each pixel being 1.4 micrometers by 1.4 micrometers. This makes the CMOS sensor horizontally 3.64 millimeters long and vertically 2.74 millimeters tall, in other words, a 5.18 Megapixel count, which fits our area of imaging capture from our lens system. The visual sensors also come with image quality controls to better filter our color choices and auto-focusing functions so we can easily program our visual sensor to better pick up colors. These features and more are thanks to the camera's open-source application programming interface (API), which means this device is quite easy to modify within its source code and program due to its widely accessible library, something the CS members of the team will appreciate. This visual sensor, OV05642, is another widely used sensor for general imagery applications in the forms of entertainment, medical, and phones, making this a reliable sensor to incorporate within our project. There are more interesting features this specific sensor provides, like high performance, support for anti-shake, embedded memory for part identifications, and more.

The Mega 5MP SPI camera is versatile. Its dimensions are 33 millimeters wide, 33 millimeters tall, and 17 millimeters long in-depth. While this camera can easily fit on top of our glasses, its bulky appearance might be a slight problem for keeping the spectacles up. The camera also comes with a FOV of 68.75 degrees diagonally. On top of that, the camera has a focal length of about 3.3 millimeters and a focus distance from 8 centimeters to infinity. This means the lens system could have about 80 millimeters of focal length and more to properly capture images, which is another hurdle our optical design would have to work around. On the bright side, this camera being open source means that it is compatible with multiple MCUs like Arduino, Raspberry Pi, Nordic, Renesas, and plenty more. This device comes to a total of \$ 34.99, which is a decent price with all these camera features, CMOS sensor specifications, and overall device accessibility.

The next sensor on the list is the **5MP OV5647 Autofocus Camera** [39], the Mega 5MP SPI minus the bulky nature. The camera has the same sensor as the Mega 5MP SPI, Omnivision OV5647 sensor has the same 5 Megapixel 2592 pixels wide and 1944 pixels high as before. The sensor has the same specific features as visible light capture being a similar FOV of 54 degrees horizontally

and 44 degrees vertically, a similar autofocus that is programmable, similar wide usage of applications, and open source for multiple MCUs. The CMOS sensor is also just as big as before being  $\frac{1}{4}$ ," however, it has a focal length of 2 inches to infinity, making it more flexible for the lens system. The camera is also more compact, being only 25 millimeters wide and 24 millimeters tall, giving it more flexibility to set up on our glasses. On top of that, the camera kit comes with a 3.28ft extension cable, which is not much of a distinctive feature but gives us even more flexibility to set up our device's MCU to our smart glasses. With a price of \$19.99, it may seem like a cheaper product but works just as well as similar 5MP camera products.

Going back to the 2MP CMOS Sensors, the **Mini 2MP Plus – OV2640 SPI Camera Module** [40] is one of the most common CMOS sensors used for Arduino projects. This camera is equipped with the OV2640, a 2-megapixel count CMOS sensor that has an array size of 1600 pixels wide and 1200 pixels tall with each pixel being 2.2 micrometers by 2.2 micrometers. This makes the area of the visual sensor 3.52 millimeters wide and 2.64 millimeters, which is not the best pixel size to sensor array size, being it would not give us the highest quality images. This device is also small, just 20 grams 24 millimeters wide, and 34 millimeters tall. The camera, not only being very compatible with many MCUs and low power consumption, has the main selling point of having a special lens mount. This means we can attach certain lenses to this device, like a makeshift M12 lens system, with its established FOV of 60 degrees and effective focal length of about 5 mm, making this another flexible camera to use. The camera also comes with a serial peripheral interface (SPI), another open source of commands that ties in with camera control. Being that this camera is the mini version of its other 2MP predecessors, the price of \$25.99 makes more sense.

The **2MP IMX462 Color Ultra Low Light STARVIS Camera Module** [41] is one step above the Mini 2MP due to its wider range of features. The CMOS sensor within this device, the Sony STARVIS IMX462, has a 2 Megapixel count with 1920 pixels wide and 1080 tall with each pixel being 2.9 micrometers by 2.9 micrometers. This leaves the sensor size being 5.57 millimeters wide and 3.13 millimeters, the largest sensor size from the list for sensors so far, which can negatively impact how the image is captured by the sensor unless we heavily modify the lens system to produce a large enough image to get captured with good enough quality. This, of course, means a large enough lens system to encompass the entire sensor just to produce a good enough image. On that topic, this sensor has chromaticity for visible light, in other words, color, and a color filter array for Quad-Bayer RGB, making this one of the best sensors to use for color detection. The sensor is especially equipped for ultra-low light support, meaning this works even during the darkest of environments. This gives it a broad application usage of surveillance cameras and general cameras, fitting the list of what kind of camera we are looking for.

The sensor's processing unit board size is also very narrow, just 25 millimeters tall and 24 millimeters wide. The device is capable of providing high-quality videos, and 30 FPS at max resolution usage, which is important for our optical design's feedback response. The sensor also has an open-source library for multiple MCUs like Raspberry Pi and a lens housing for an M12 lens system. This allows the camera to have a horizontal FOV of 141.4 degrees (about 160 degrees diagonal FOV), which is significantly more than what our optical design calls for. The focal length, according to the datasheet, for this device is about 2.1 millimeters  $\pm$  5%, making this visual sensor flexible to make a lens system. Overall, with this device being \$ 39.99, it has a decent price for the tradeoff of some features like there being a lower-than-preferred resolution but added quad-Bayer RGB color filter and ultra-low light detection.

The **2 Megapixels MT9D111** [42] camera with autofocus and lens flex module features is a remarkable sight for CMOS cameras. The CMOS sensor MT9D111 has a compact resolution with 2 Megapixels, being only 1,600 pixels wide and 1,200 pixels tall and each pixel being 2.8 micrometers by 2.8 micrometers. The active pixel area size is then 4.73 millimeters wide and 3.52 millimeters tall, falling just below our preferred target for pixel size to the sensor's area size. This CMOS sensor is also what is known as a System-On-A-Chip (SOC) sensor, which means it has extra features built within like on-chip autofocus with optical zoom, an easy-to-modify GPIO interface, real-time anti-aliasing, and an integrated microcontroller just to name a few features correlated to our project. All those features are integrated within the CMOS sensor, this does not account for the adaptable model the sensor is housed on, which is fascinating to note as the board only gives the sensor the ability to be flexible with any MCU available commercially (like Raspberry PI or Arduino). All this high performance and low power is within a 40-millimeter wide, 20-millimeter tall, and 55-millimeter in-depth device and just 10 grams. There is only one parameter, however, this device does not have that all other cameras listed have which is a FOV, which is where our lens system would have to come into play. The price being \$9.99 makes this sensor top-listed on our optical components list.

The **8MP IMX219 Auto Focus Camera Module** [43] is a motorized focusing visual sensor commonly used for mobile medical devices, drones, and security surveillance. The CMOS sensor, the IMX219, has about 8 Megapixels with a resolution of 3296 pixels wide and 2512 pixels tall with each pixel being 1.12 micrometers by 1.23 micrometers. The active sensor area is about 3.68 millimeters wide and 2.76 millimeters tall, making this the most high-resolution sensor for taking live video feedback thanks to its established high performance. Another important feature this sensor has is modifiable color sensitivity for R, G, and B filter values, which is important for specifying the colors we would like this camera to capture. Other features this sensor has are flexibility with multiple MCUs (especially with Raspberry PI) and programmable autofocus. The board for the camera module has a few notable features as well, like the focal length

being 2.8 millimeters, the focal distance starting from 80 millimeters, the FOV being 78 degrees diagonal, and the board itself being 25 millimeters wide and 24 millimeters tall. All these features for this camera come together for \$22.99, which is a great deal for our optical project.

Moving up in pixel count, the **IMX519 PDAF&CDAF Autofocus Camera Module** [44] has 16 Megapixels and greater processing power than its predecessors. The CMOS sensor Sony IMX519 has a resolution of 4656 pixels wide and 3496 pixels tall with each pixel being 1.22 micrometers by 1.22 micrometers, leaving the sensor's area about 5.57 millimeters wide and 4.27 millimeters tall. This sensor has a high-definition resolution which is great for more high-demanding image processing applications, and well exceeds our main project's demand for color identification. The sensor also comes attached with auto-focusing lenses with a focal length starting at 100 millimeters, which leaves extraordinarily little flexibility for our lens system since we need it to be compact. The camera board is also 25 millimeters wide and 24 millimeters tall, so it fits the small criteria needed for our optical design. The main marketing feature of this device has been having high definition for taking pictures and high frame rates for high-resolution videos. It is also compatible with most MCUs, but it focuses more on Raspberry PI and NVIDIA Jetson MCU models. The device comes with a price tag of \$24.99, so the pricing for the high-definition image-capturing features can be something to consider for our project.

The next few camera models on the list contain the same type of sensor, the 5MP OV5640, but each design is tweaked with differing sizes and certain features. The OV5640, as mentioned before, has a 2592×1944 resolution, a 1.4  $\mu\text{m}$  × 1.4  $\mu\text{m}$  pixel size, and an active sensor area of 3.64 mm x 2.74 mm. As the sensor's specifications and processing power remains the same, it is the camera boards each sensor is attached to that vary each device's features and prices. For instance, the **5MP OV5647 Fixed Focus Camera Module with M12 Lens** [45] has a housing for replaceable M12 lens, compatibility with the latest models of Raspberry PI, and has dimensions 36 mm x 36 mm. The price for this device is \$15.99. The **5MP OV5647 Mini Camera Module** [46] is like the previous model except it has the dimensions 25 mm × 24 mm (hence its mini camera title), has a focal length of 3.60mm, and the focus distance from 1 meter to infinity. The price for this device is \$16.99. The **5MP MIPI Camera for RZBoard V2L with Renesas RZ/V2L Processor** [47] is also like the previous device from it dimensions but unlike its previously listed model, this camera has an interface that connects well with RZBoard V2L with Renesas RZ/V2L Processors. The price for this device is \$19.99. The **Pcam 5C** [48] camera also has a fixed lens with an already installed M12 mount lens, a PCB board that has dimensions 40 mm x 25 mm, and a dual lane MIPI CSI-2 image sensor interface for more customizable color detection. The price for this device is \$49.99. The last 5MP listed is the **OV5640 Camera Board C** [49] which also has auto-focusing, adjustable focal length starting from 3.37mm, a horizontal FOV of about 65

degrees, and a dimension of 35.70mm × 23.90mm. The price for this device is \$25.99. Each of these 5MP visual sensors has its preferred features and designs for a multitude of image-sensing applications. Choosing the best 5MP type camera that fits our optical design, it will have to be the choice between the Mega 5MP SPI and Pcam 5C, both slightly pricier cameras that have the distinctive feature of having easy-to-code color detection programs and sensor designs.

A new sensor enters the ring as the **Vision AI Module Grove** [50] is a unique model that uses AI to enhance the imaging sensing of this camera. The AI-powered visual sensor is the OV2640, which is a 2MP CMOS sensor with a resolution of 1600 x 1200 and a pixel size of 2.2 micrometers x 2.2 micrometers. The module itself comes with a few extra features like having a Himax HX6537-A processor which has easy connections to IoT, supports Edge Impulse data flow for easy modifications of code, and is open source to many MCUs like Arduino. On top of that, the model also comes with a built-in microphone as well as a 6-axis Inertial Measurement Unit, furthering the AI flexibility this single-camera model has. This camera is extra packed within 40 x 20 x 13 millimeters and provides extra features to further enhance the Smarter Glasses. Being that the camera model comes packed with a price of only \$25.99, this model is now top on the list of cameras to use in the optical design.

The **IMX219-AF Programmable/Auto Focus Camera Module** [51] is an 8MP visual sensor with programmable focusing abilities. The CMOS sensor IMX219 is an 8 Megapixel count visual sensor with a resolution of 3296 pixels horizontally and 2512 pixels vertically with each pixel being 1.12 micrometers by 1.12 micrometers. The active area of the sensor is 3.69 millimeters wide and 2.81 millimeters tall, which gives the sensor a higher definition for capturing live images and has a frame rate of 30 fps when setting the resolution to 1920 x 1080. The sensor has a horizontal FOV of 65 degrees, a focal length of 2.8 millimeters, and a flexible focal distance from 80 millimeters to infinite. The visual sensor also comes equipped with programmable focusing language, which can be easily modified with any MCU with its open-source library (primarily with the NVIDIA Jetson Orin series). Being that this versatile sensor is used in many visual applications, has flexible board size dimensions of 25 x 24mm, and is priced at \$22.99, this is a good camera that is worth mentioning for our optical design.

The **477P High-Quality Camera** [52] is another high-definition camera that has a 12-megapixel count. The visual sensor is the IMX477, a 12MP CMOS sensor that has a resolution of 4072 pixels wide and 3176 pixels tall with each pixel being 1.55 micrometers by 1.55 micrometers. This leaves the sensor's active area to be 6.31 millimeters wide and 4.92 millimeters tall, giving a high-definition resolution to the image being captured as well as near 50 fps with a resolution of 2028 × 1080. While this camera has a fixed focus, it houses a CS-Mount which is capable of giving the sensor a horizontal FOV of 65 degrees, a front focal length



of 6 millimeters, and a back focal length of about 7.53 millimeters. Being that this camera is considered Ultra HD, has an open source of compatibility with many MCUs, a board dimension of 38 millimeters by 38 millimeters, and is priced at \$79.99, it is a flexible performing camera that (due to its price) is on the bottom of our sensor's list.

The **12MP IMX708 Wide Angle Camera Module** [53] is another high-definition camera that has not only ultra-high definition like its previous version but also has a wide FOV. The visual sensor is the IMX708 CMOS sensor which has a resolution of 4608 pixels wide and 2592 pixels tall with each pixel being 1.4 micrometers by 1.4 micrometers. The active sensor area is 6.45 millimeters wide and 3.63 millimeters tall, with an average of 55 fps when the resolution is set to 2304 x 1296, well over the minimum requirement for our optical design. Some unique features of this camera module include an M12 lens housing which can give the visual sensor a wide FOV of 120 degrees, a manual focal length of 2.87 millimeters, an open-source module to is compatible with many MCUs, and board dimensions of 25 millimeters by 24 millimeters. The downside is both the manual focusing and too wide of a lens the camera has, but the price of \$ 36.99 balances the versatility this camera carries.

Going up the megapixel count, the **16MP IMX298 Color Camera Module** [54] is a higher quality camera that comes with the cost of high processing power. The IMX298 CMOS sensor has a resolution of 4656 wide and 3496 pixels tall with each pixel being 1.12 micrometers by 1.12 micrometers. This means that the visual sensor has an active array area of 5.21 millimeters wide and 3.92 millimeters tall, having an average of 50 fps at a resolution of 1280x720. This sensor has auto-focusing features with a focal length of 100 millimeters to near infinite, has a wide FOV of 60 degrees horizontally, a large compatibility library for many MCUs, and ISP support of auto exposure which leaves the captured/recorded images clear and very saturated. It also is compact, being just 40 millimeters by 40 millimeters but not as compact as previously listed devices. The camera comes attractively packaged with a price tag of \$59.99, so it is a nice camera for our project within a decent price range, making this a good contender for the final lineup of our optical design.

With the highest pixel count, the **64MP Ultra High-Resolution Autofocus Camera Module** [55] is the newest high-definition camera on the market. The 64MP Autofocus Camera has a resolution of 9152 pixels wide and 6944 pixels tall with a pixel size of 0.8 micrometers by 0.8 micrometers. The area of this sensor is 7.32 millimeters wide and 5.56 millimeters tall, having an average of 60 fps at a resolution of only 1920 x 1080. This visual sensor has about 64 million pixels, which makes this the most high-quality camera for MCU usage to date, which has its benefits and drawbacks. The main benefits are its quad bayer coding color filter for even more color detection modifications, a wide MCO compatibility library for even each model's 1st and 2nd versions, and even

auto-focusing capabilities with a FOV of 84 degrees diagonal and focal length of 5.1 millimeters. The drawbacks this device has however is due to its performance requires a lot of processing power that older MCUs cannot provide, so it is recommended to use Raspberry PI for this device. Another drawback is the focusing distance starting from 80 millimeters, which is an obstacle to our optical design's compact nature. However, this high-performing device is only 25 millimeters tall by 24 millimeters wide and has a price of \$ 59.99. While this is a powerful camera that can take ultra-high-definition videos, this may be too much for what our optical design is supposed to achieve.

The **AMB82-Mini IoT AI Camera** [56] is another unique camera model that utilizes AI-powered visual sensing and IoT capabilities for commonly used visual sensing applications. The CMOS sensor is a JXF37, a 5MP visible light sensor that has a resolution of 1920 pixels wide and 1080 pixels tall. Unfortunately, this CMOS sensor's datasheet is very secretive. The only available feature that's being displayed is the camera's resolution and the FOV is about 130 degrees horizontal. The rest of the specifications for this device come down to its IoT capabilities (Wi-Fi/Bluetooth) and AI modules, built-in high MCU and NPU processing performance while still retaining low power consumption, and complex IoT cryptographic engine security with a flexible board interface. It is an excellent quality camera with a high-performing MCU board for the price of \$24.99.

The **MU Vision Sensor 3** [57] is designed for immediate integration within robotic systems and multiple visual recognition application features. This device has the MU Vision Sensor CMOS sensor which has a resolution of 640 pixels wide and 480 pixels tall, a horizontal FOV of 84 degrees, and 60 fps using the max resolution of the sensor. The camera also has a focal length of 2.5 millimeters, an aperture size of 2.2 millimeters, and a lens diameter of 5.5 millimeters, important specifications to note which will determine the optics design itself. While the image quality is not the device's most important quality compared to other cameras listed, this device does contain a complex IDE source that multiple MCUs can access (like Arduino). This open-source library of the device contains functions premade for color recognition, human gestures, wi-fi transmission, and more premade detection algorithms. The device is quite compact, having its dimensions just 33 x 32 x 11.5 millimeters and weighing 6.8 grams, and the price tag is only \$59.00.

The **XIAO ESP32S3 Sense** [58] camera is a multipurpose device that does not just capture images but also has a built-in microphone, IoT capabilities, and an interactive interface. The sensor is an OV2640 CMOS sensor, with the specifications mentioned before having a 1600 x 1200 resolution, focal length of 3.29 millimeters, and horizontal FOV being about 69 degrees. This device can easily detach its camera however and use better CMOS cameras like the OV5640, another CMOS sensor that has a 5-megapixel count. This board is very

flexible with its visible sensor as well as a powerful MCU, great memory processing, and high performance while maintaining efficient power consumption, and it is all compacted within a dimension of 21 millimeters by 17.5 millimeters. This device also has IoT capabilities (Wi-Fi and Bluetooth), a built-in microphone, and battery charging support. In terms of making this camera wireless for our optical design, there is an option for that plus the price tag of \$13.99 makes this an interesting camera for us to use for the optical design.

### 3.8.4 Microcontroller Units and Development/Flash Boards

Some microcontroller units and their respective development boards that we feel are the most related to our project are introduced below.

*Table 8. MCUs and Development Boards Comparison*

Feature	Espressif ESP32-S3 Series	STM32H747/757 Series	Jetson Nano Developer Kit	Raspberry Pi Pico	Raspberry Pi Zero 2 W	Raspberry Pi 5	Arduino Nano 33 BLE Sense	Seeed Studio XIAO ESP32S3 Sense
Processor	Xtensa 32-bit LX7 dual-core at up to 240 MHz	Cortex-M7 at 480 MHz and Cortex-M4 at 240 MHz	Quad-core ARM A57 CPU at 1.43 GHz	Dual-core ARM Cortex-M0+ at 133 MHz	1GHz quad-core 64-bit Arm Cortex-A53 CPU	2.4GHz quad-core 64-bit Arm Cortex-A76 CPU	32-bit ARM Cortex-M4 at 64MHz	Xtensa 32-bit LX7 dual-core at up to 240 MHz
Memory	512 KB SRAM, 384 KB ROM	Up to 2 MB Flash, 1 MB SRAM with TCM RAM architecture	4 GB LPDDR4 RAM	264 KB on-chip SRAM, 2 MB QSPI flash	512 MB SDRAM	4GB and 8GB LPDDR4 X-4267 SDRAM	1 MB flash, 256 KB SRAM	8 MB PSRAM, 8 MB FLASH
Connectivity	Wi-Fi, Bluetooth Low Energy (BLE)	No wireless connectivity	Gigabit Ethernet, M.2 Key E, HDMI, display port, USB	No wireless connectivity	2.4 GHz wireless LAN, Bluetooth 4.2	Dual-band 802.11ac Wi-Fi, Bluetooth 5.0 / BLE	Bluetooth Low Energy, Bluetooth client and host device	2.4GHz Wi-Fi, BLE 5.0 dual wireless communication
Peripherals	45 GPIOs,	Rich set of	GPIO, I2C, I2S,	26 GPIO pins,	Micro-USB,	4 USB ports,	14 digital I/O pins,	1x UART, 1x

	SPI, I2S, I2C, PWM, RMT, ADC, DAC, UART, others	peripherals, extensive memory options	SPI, UART, and more	including 3 analog inputs	mini-HDMI, CSI-2 camera connector, Raspberry Pi 40 pin header	Ethernet, 2 MIPI cameras, PCIe 2.0, Raspberry Pi 40 pin header	PWM support, UART, SPI, I2C	IIC, 1x IIS, 1x SPI, 11x GPIOs (PWM), 9x ADC, 1x User LED, 1x Charge LED, 1x B2B Connector (with 2 additional GPIOs), 1x Reset button, 1x Boot button
Special Features	Vector instructions for accelerating neural networks	Graphics capabilities, TFT-LCD controller, MIPI-DSI	128-core Maxwell GPU, CUDA programming	Programmable I/O subsystem (PIO)	Compact form factor, H.264 and MPEG-4 decoding/encoding	Fastest in generation, M.2 SSD connectivity, power button	Sensors for gesture, light, proximity, and more	Integrated camera sensor, digital microphone, SD card slot (up to 32GB FAT)
Development Board (Price)	ESP32-S3-DevKit C-1 at \$15.00	Not specified	Jetson Nano Developer Kit at \$149.00	Raspberry Pi Pico at \$4.00	Raspberry Pi Zero 2 W at \$15.00	\$80	Arduino Nano 33 BLE Sense at \$40.50	Seeed Studio XIAO ESP32S3 Sense at \$13.99
Board Dimensions	25.40mm x 62.74mm	Not specified	69mm x 45mm	21mm x 51mm	65mm x 30mm	56mm x 85mm	45mm x 18mm	21mm x 17.5mm

ESP32-S3 Series: [111] is a collection of powerful, versatile Wi-Fi and Bluetooth Low Energy (LE) MCU modules designed for a spectrum of applications, particularly in the realms of AI and the Internet of Things (IoT). Anchored by the Xtensa® 32-bit LX7 dual-core processor with a clock speed of up to 240 MHz, these modules boast 512 KB of SRAM, and 384 KB of ROM, and support various

flash and external RAM interfaces. Noteworthy features include vector instructions for accelerating neural network computing and signal processing workloads, along with a rich array of peripherals encompassing 45 GPIOs, SPI, I2S, I2C, PWM, RMT, ADC, DAC, and UART, among others. Security is robustly ensured through RSA-based secure boot, AES-XTS-based flash encryption, digital signatures, and the HMAC peripheral. The series offers fully certified modules with integrated antennas, with variations in flash and PSRAM configurations. The ESP32-S3-WROOM-1, ESP32-S3-WROOM-1U, and ESP32-S3-WROOM-2 modules present distinct dimensions, pin counts, flash, and PSRAM options, making them suitable for diverse applications, from smart home devices to AI-driven solutions. Development boards like the ESP32-S3-DevKitC-1 complement these modules, providing a comprehensive ecosystem for AIoT development.

The ESP32-S3-DevKitC-1 v1.1 [112] serves as an accessible development board, retails at \$15.00, tailored for the ESP32-S3-WROOM-1, ESP32-S3-WROOM-1U, or ESP32-S3-WROOM-2 modules, combining Wi-Fi and Bluetooth Low Energy functionalities. Engineered for simplicity, it offers convenient interfacing through broken-out I/O pins, supporting peripheral connections via jumper wires, or placement on a breadboard. Key components encompass the potent ESP32-S3-WROOM modules, a 5V to 3.3V LDO power regulator, pin headers, USB-to-UART port, boot and reset buttons, USB port, USB-to-UART bridge, RGB LED, and a 3.3V power-on LED. Its dimensions are 25.40mm x 62.74 mm. Noteworthy is the availability of various storage options, including integrated flash memory capacities such as 8 MB, 16 MB, and 32 MB, with some variants featuring additional Pseudo-Static RAM (PSRAM) configurations. This diversity caters to a spectrum of development needs, ensuring ample storage space for firmware, applications, and data, making the ESP32-S3-DevKitC-1 v1.1 an invaluable resource for both novice and seasoned developers, as well as for our project. Espressif provides on their website accompanying documentation furnishing comprehensive hardware details, power supply options, pin layouts, and a step-by-step guide for seamless setup and application development.

STM32H747/757 series: [113] stands out for its exceptional performance, combining a Cortex-M7 running at 480 MHz and a Cortex-M4 at 240 MHz, delivering impressive benchmark scores. With security features like crypto/hash hardware acceleration and secure firmware services, it ensures robust protection for software IPs. Power efficiency is enhanced through a multi-power domain architecture, an embedded SMPS, and low-power modes, making it versatile for various applications. Graphics capabilities, including a TFT-LCD controller, MIPI-DSI support, and accelerators for content creation, cater to applications demanding sophisticated visuals. The rich set of peripherals, extensive memory options, and support for communication interfaces make it suitable for complex projects. However, its comprehensive feature set and potential cost may be

considered overkill for simpler applications, making it most beneficial for advanced projects requiring a balance of high performance, security, and graphical capabilities. Furthermore, the lack of wireless connectivity is a major drawback for us, as it requires some experience in circuit design and microelectronics to integrate, which is not our area of expertise.

Jetson Nano Developer Kit: [114] represents a compact and potent computing solution tailored for makers, learners, and embedded developers, bringing the prowess of modern AI to a diverse array of applications. It retails at \$149.00. At its core, the kit is equipped with a 128-core Maxwell GPU and a quad-core ARM A57 CPU running at 1.43 GHz. With 4 GB of 64-bit LPDDR4 memory boasting a bandwidth of 25.6 GB/s, the Jetson Nano Developer Kit provides a formidable platform for running multiple neural networks in parallel, enabling tasks such as image classification, object detection, segmentation, and speech processing. Its connectivity features include Gigabit Ethernet, M.2 Key E, HDMI, display port, and various USB ports, accompanied by interfaces like GPIO, I2C, I2S, SPI, and UART. The mechanical design is compact for a minicomputer (69 mm x 45 mm), and its compatibility with the NVIDIA JetPack SDK ensures ease of use, making it operable with as little as 5 watts. For those familiar with NVIDIA GPUs and CUDA programming, the Jetson Nano offers a robust solution.

For a project's feature being real-time color detection, utilizing the Jetson Nano Developer Kit could be considered overkill due to its advanced computational capabilities tailored for complex AI and deep learning applications. Real-time color detection is a relatively straightforward task that may not necessitate the sophisticated features offered by the Jetson Nano. While the platform provides flexibility for future expansion and serves as a valuable learning opportunity for advanced AI development, its higher cost, increased power consumption, and inherent complexity might outweigh its benefits for a project with more straightforward requirements. Instead, exploring alternatives like microcontrollers such as ESP32, Arduino, or Raspberry Pi, along with libraries like OpenCV, could offer a more cost-effective and power-efficient solution for our specific task.

Raspberry Pi Pico: [115] Unlike other Raspberry Pi models, Pico is a microcontroller board based on the RP2040 MCU developed by Raspberry Pi featuring a dual-core processor complex, a rich peripheral set, and a unique Programmable I/O (PIO) subsystem; it is available from \$4. The RP2040 offers some on-chip memory (264 KB on-chip SRAM; 2MB on-board QSPI flash) and supports input voltages from 1.8V to 5.5V, providing flexibility in power options. Pico W, an extension of the Pico product line, supports 2.4GHz 802.11 b/g/n wireless LAN and Bluetooth 5.2, operating in both station and access-point modes. The Raspberry Pi Pico is a compact (21 mm × 51 mm) and cost-effective board suitable for smaller projects. For projects with an external OLED display and other I/O requirements, the Raspberry Pi Pico can be easily interfaced with

through its 26 GPIO pins, including three analog inputs. It is programmed using MicroPython or C/C++ using the Raspberry Pi Pico C/C++ SDK.

Raspberry Pi Zero 2 W: [116] is a compact computer featuring its RP3A0 system-in-package, a 1GHz quad-core processor, 512MB of RAM, 2.4GHz wireless LAN, and Bluetooth 4.2 with BLE. With a mini-HDMI port for video output, micro-USB ports, a microSD card slot, and a CSI-2 camera connector, it offers versatile connectivity options and opens up possibilities for diverse projects while only retailing for \$15.00. The device supports H.264 and MPEG-4 decoding/encoding (1080p30), OpenGL ES 1.1 and 2.0 graphics, and includes a micro-USB power supply for reliable performance. It takes an input power of 5V DC 2.5A. Its small form factor (65mm x 30mm) makes it suitable for various projects, including smart home applications and IoT projects. As with all Raspberry Pi products, it comes with extensive documentation and support from the company and the community. The Raspberry Pi Zero 2 W is an impressive and affordable computing solution; however, some of its drawbacks are the power consumption is not as efficient and the form factor is not as small as we would have liked.

Raspberry Pi 5: [129] is the fastest Raspberry Pi board yet with speeds of up to 2-3x the previous generation. It has a Broadcom BCM2712 quad-core Arm Cortex A76 processor clocked at 2.4GHz, and supports RAM up to 8GB. Some of its key features are 2x USB 3.0 ports, 2x USB 2.0 ports, Dual 4Kp60 HDMI outputs with HDR support for high-quality video output, dual-band 802.11ac Wi-Fi, Bluetooth 5.0, Gigabit Ethernet, 1x PCIe 2.0 interface, and a UART debug port. It also has the standard Raspberry Pi 40 pin header that includes GPIO, PWM, I2C, I2S, SPI, and serial. It requires a 27W power supply to operate at full capability. This large power consumption has the drawback of producing more heat, so a cooling fan is also available for purchase. Another drawback is the large form factor, especially when used with the case and fan. It's expensive, coming in at \$80. Despite this, it would be a more than capable board for our project.

Arduino Nano 33 BLE Sense: [117] is a compact and versatile development board powered by the nRF52840 microcontroller, that retails at \$40.50. Operating at 3.3V, it features a 64MHz 32-bit ARM Cortex-M4 CPU, 1MB of CPU flash memory, and 256KB of SRAM. Equipped with a variety of sensors, including a 9-axis inertial sensor, humidity and temperature sensor, barometric pressure sensor, microphone, and sensors for gesture, light, and proximity, the board is well-suited for AI and IoT applications. It supports Edge Computing applications through TinyML, allowing the execution of machine learning models created with TensorFlow Lite. The Nano 33 BLE Sense operates as both a Bluetooth Low Energy and Bluetooth client and host device. With 14 digital I/O pins, PWM support on all digital pins, and interfaces such as UART, SPI, and I2C, it offers flexibility for a wide range of projects. The compact form factor (45mm x 18mm)

and low weight (5g with headers) enhance its suitability for wearables and various embedded applications. Additionally, it has a micro-USB connector, making it easy to interface with other devices. The open-source hardware design allows users to explore and modify the board for their specific needs. While it does offer some integrated storage, the amount offered is not much and its MCU is not as powerful compared to the other boards. Furthermore, it also lacks Wi-Fi connectivity and thus it is not a versatile and capable option.

Seeed Studio XIAO ESP32S3 Sense: [118] is a compact development board designed for intelligent voice and vision AI applications and retails at \$13.99. It features the powerful MCU ESP32S3 32-bit dual-core Xtensa processor, operating at up to 240 MHz, with Arduino/MicroPython support. The board includes advanced features such as a detachable OV2640 camera sensor for 1600\*1200 resolution, a digital microphone, 8MB PSRAM, and 8MB FLASH memory, along with an SD card slot supporting up to 32GB FAT memory. With exceptional RF performance supporting 2.4GHz Wi-Fi and BLE dual wireless communication, it is suitable for various applications, including image processing, speech recognition, video monitoring, wearables, smart homes, health monitoring, education, low-power networking, and rapid prototyping. The thumb-sized design, measuring 21mm x 17.5mm, makes it ideal for space-limited projects. Additionally, it is compatible with the SeeedStudio Grove ecosystem, offering extensive possibilities for expansion and exploration. Compared to the other options, this board is the most suitable for our project, not only does it have both integrated camera and microphone sensors, but it also has one of the largest onboard storage systems and one of the most powerful and cheapest boards. Its tiny size is a big strength, however, the downside to that is it lacks the ports and pins that we might need for the other I/O components which would require some time to learn, acquire necessary parts, and tinker to get the right connection.

### **3.8.5 Microphones**

The following are some microphones that we could use for our project.

The Adafruit I2S MEMS Microphone Breakout - SPH0645LM4H [119] is a tiny I2S MEMS (Micro-Electro-Mechanical Systems) Microphone with a range of 50Hz to 15 KHz. It costs \$6.95, weighs 0.4g, and is 16.7mm x 12.7mm x 1.8mm. It's 1.6-3.6V compatible. It doesn't have an analog output, it's purely digital with three digital pins: clock, data, and word select (left and right). The sound quality is good for the size and price. This microphone is best used with Cortex M-series chips like the Arduino Zero, Feather M0, or SBCs like the Raspberry Pi.



The Electret Microphone Amplifier - MAX4466 with Adjustable Gain [120] has a range of 20Hz to 20KHz. It's best used for projects such as voice changers, audio recording/sampling, and audio reaction. It uses the Maxim MAX4466 amplifier which is less noisy and scratchy than other amp breakouts. It includes an adjustable gain which can be set from 25x to 125x. It costs \$6.95 and is 7.8mm by 0.307in and supports 2.4-5V.

The ReSpeaker 2-Mics Pi HAT [121] has 2 analog microphones and WM8960 Audio Codec which provides HD voice capturing. It's specifically designed for the Raspberry Pi series. The NLU software algorithm, has features like voice activity detection, the direction of arrival, and keyword spotting. It costs \$12.90, weighs 33g, and is 65mm by 30mm by 15mm. This microphone array provides great quality but is very large.

The SunFounder USB 2.0 Mini Microphone for Raspberry Pi [129] is compatible with Raspberry Pi Models using a USB port. It costs \$7.99, weighs 10g, and is 22mm x 18mm x 7mm. Some key features it has are plug and play, 1db sensitivity, and omnidirectional pickup. Customer reviews on Amazon suggest it has mediocre audio quality.

Onboard microphones: The computer we have chosen could also come with a built-in microphone. For example, the XIAO ESP32S3 Sense has an integrated digital microphone. Some advantages to this are no need to ensure compatibility, no need to manually connect the microphone to the computer, and a more compact design as there's no need for extra hardware. Onboard microphones can also simplify the setup and development process because they're integrated into the system's hardware, and software, and usually have documentation on how to use them.

### **3.9 Project Software Overview**

The overall software architecture of our project, including high-level components and their interactions.

User Interface (UI) Layer: This layer is responsible for the user interaction and presentation of information. It includes the graphical user interface (GUI) components. Components include:

- **Heads-up Display (HUD):** Displays translated text, real-time object tracking information, and other visuals.
- **User Controls:** Accept user inputs, such as voice commands and button presses.

- **Speech Output:** Provides audio feedback or translation playback.

Real-Time Speech Recognition Layer: This layer focuses on capturing spoken language and converting it into text. Components include:

- **Microphone Interface:** Captures audio input from the environment.
- **Speech Recognition Engine:** Transcribes spoken words into text.
- **Language Detection:** Identifies the source language for translation.

Real-Time Translation Layer: This layer is responsible for translating text from one language to another. Components include:

- **Language Database:** Stores language translations and dictionaries.
- **Translation Engine:** Performs real-time translation from the source language to the target language.
- **Display Integration:** Delivers translated text to the HUD for presentation.

Communication Layer: Facilitates data exchange between components and external services. Components include:

- **Wi-Fi/Bluetooth:** Provides connectivity for cloud-based services and remote control.
- **API Integrations:** Communicates with cloud-based services for speech recognition, and translation.
- **Data Exchange:** Handles the exchange of data between components.

Real-Time Operating System (RTOS): An RTOS manages task scheduling, ensuring that real-time components like speech recognition and color detection have priority. RTOS manages system resources and provides thread synchronization.

APIs and Libraries: Utilize APIs and libraries for speech recognition, translation, and computer vision tasks. Third-party APIs and open-source libraries may be integrated for added functionality.

Data Storage: This project requires local data storage to store some necessary dictionaries, user preferences, and object profiles.

Error Handling and Logging: This layer implements error-handling mechanisms to manage unexpected scenarios, log errors, and system events for debugging and performance monitoring.

The software will be organized into modular components, each responsible for a specific function or feature. Each module will have a clear role and responsibilities. The speech recognition module is responsible for capturing audio input, processing it for speech recognition, and converting it to text. The object tracking module handles image or video input from the camera, identifies and tracks objects, and provides their positions and data. The user interface

module manages user interactions, displaying information on the smart glasses' display, and receiving user commands.

This project has 2 main systems: live speech-to-text and color detection. The AR glasses will include multiple main components, those being the microphone, mini display, camera, and computer, in this context, computer refers to microcontrollers (MCUs), single board computers (SBCs), etc. This project requires the use of different types of software and application programming interfaces (APIs) to interact with all these components. These can be easier or harder to implement depending on what components we choose. For example, a camera we're considering, the Pixy2, is easily capable of color detection with object tracking, as it comes with its processor, allowing it to offload the resource drain, and it has supported software that's easy to install and use. The main downside is its size, which isn't ideal for wearability.

The following sequence diagram shows how the user and objects will interact with each other over time. It shows when an object is actively being used and how long it's active. It is useful when building our glasses because it visualizes when a process should be active.

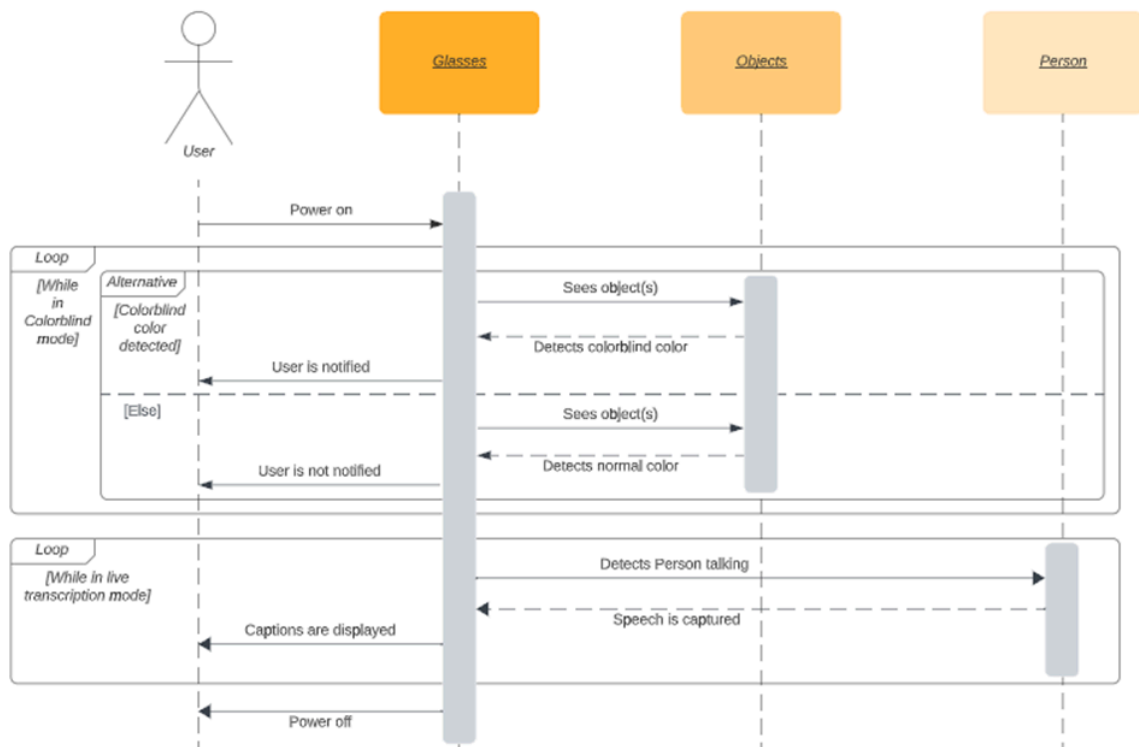


Figure 13. Sequence Diagram for Color Detection and Speech Transcription

### 3.10 Operating Systems

Since our group prioritizes low power consumption, there are a few notable operating systems to consider based on our list of potential SBCs or MCUs for our project.

Raspbian: Raspberry Pi is a single-board computer (SBC) that is supported by its operating system called the “Raspberry Pi OS” formerly known as Raspbian. It is a Debian-based variant of Linux OS distribution, specified and optimized for the Raspberry Pi hardware. It is lightweight and has a user-friendly desktop environment that is similar to traditional desktop operating systems, acclimating for users of varying levels of Linux experience. It comes preinstalled with programming tools among various other software, such as a web browser, office suite, and more, suitable to provide a complete computing experience out of the box. Advanced developers can utilize the capable command-line interface if they prefer to work on the terminal. It is also highly customizable and allows users to freely install different desktop environments and software packages, and configure the system however they like. It is an excellent choice for various Raspberry Pi projects, from simple educational tasks to more advanced applications like IoT devices, media centers, and servers, for many beginners and hobbyists. Python is the most popular and widely used language on Raspberry Pi due to its simplicity and the availability of GPIO libraries for hardware interfacing. However, developers have the flexibility to choose any language that best suits their project's requirements like C/C++, JavaScript, Java, etc. Developers can choose any environment to write their sketches, a text editor, or an Integrated Development Environment (IDE) like Thonny (Python), Visual Studio Code, Geany, or Vim [78].

Apart from the official Raspberry Pi OS, some alternative operating systems and distributions can be installed on Raspberry Pi boards such as:

- **Ubuntu**: Ubuntu has an official version optimized for Raspberry Pi, known as "Ubuntu Server for Raspberry Pi," which is suitable for server and IoT applications.
- **Fedora**: Fedora offers a version called "Fedora for ARM" that can be used on Raspberry Pi devices.
- **Arch Linux**: Arch Linux enthusiasts can use the Arch Linux ARM distribution, which provides a minimalistic and customizable Linux experience.
- **Other Linux Distributions**: Many other Linux distributions, including Debian, CentOS, and more, can be adapted for Raspberry Pi with some effort.
- **Windows IoT Core**: Microsoft provides Windows 10 IoT Core for Raspberry Pi, which is designed for Internet of Things (IoT) applications.
- **Specialized Operating Systems**: There are specialized operating systems like RetroPie for retro gaming or OSMC for media center applications [79].

FreeRTOS is an open-source real-time operating system (RTOS) that is widely used in embedded systems and microcontroller-based platforms like the ESP32,

Arduino, STM32, and others. It has real-time capabilities, ensuring that tasks and interruptions are executed within specified timeframes. FreeRTOS allows developers to create and manage multiple tasks or threads, each with its own priority level and execution context which then gives its scheduler an order to preempt higher-priority tasks over lower ones. FreeRTOS provides a mechanism for managing system resources and supports efficient interrupt handling. It is designed to have minimal overhead, both in terms of memory usage and processing time. Thanks to its portability and configurability, FreeRTOS provides a consistent API across different platforms, making it easier to migrate projects between hardware. This makes it suitable for resource-constrained devices and thus supports a wide range of microcontrollers and architectures. C is the most used language for FreeRTOS development, given its efficiency and suitability for low-level programming on microcontrollers and other embedded platforms. C++ can also be used when necessary. Additionally, FreeRTOS has extensive documentation, including user manuals and examples, to help developers get started, as well as a vibrant online community to gather knowledge and support. [80]

Mongoose OS is a versatile platform compatible with various microcontroller units (MCUs) and tailored for IoT applications. Its pros include cross-platform support, suitability for IoT-focused smart glasses, over-the-air updates, cloud integration, robust security features, an active community, and support for multiple programming languages. However, its resource usage may need evaluation for resource-constrained smart glasses, and there might be a learning curve for newcomers. Additionally, while the open-source version is free, the commercial version (Mongoose OS Pro) comes with associated costs for advanced features, support, and compatibility with specific MCUs should be confirmed. [81]

### **3.11 Programming Environment and Tools**

The Espressif IoT Development Framework, referred to as ESP-IDF, is the official development framework for ESP32 and ESP8266 MCUs. It's built upon the FreeRTOS environment and tailored for the Espressif Systems which are widely used in IoT and embedded applications due to their low power consumption, built-in Wi-Fi and Bluetooth capabilities, and cost-effectiveness. It includes a comprehensive software development kit (SDK) that includes everything needed to develop, build, flash, and debug applications. It provides extensive hardware support for peripherals like GPIO pins, SPI, I2C, UART, and more. There is support for over-the-air (OTA) firmware updates, allowing developers to update their firmware remotely. While C is the primary language, using C++ for ESP-IDF development if needed. However, most of the core components and examples are written in C [82].

The Arduino Integrated Development Environment (IDE) is an open-source software application that serves as the primary programming environment for Arduino boards and a wide range of compatible microcontrollers including the Teensy, STM32 series, and more. It is cross-platform (available for Windows, macOS, and Linux) and provides a beginner-friendly interface for writing, compiling, and uploading code to Arduino hardware. It includes a text editor with features like syntax highlighting, auto-indentation, and code completion to help users write and edit Arduino sketches (programs). Arduino libraries provide pre-written code to simplify tasks like working with sensors, displays, and communication protocols. The IDE includes a Library Manager for easy library installation and management. Arduino supports a variety of hardware platforms, including official Arduino boards and third-party platforms. The Board Manager allows users to select the target board and install board-specific support packages. Developers can compile and upload Arduino sketches to the target board directly from the IDE, either via USB or other supported communication methods (e.g., Bluetooth or Wi-Fi). Arduino IDE can be extended with additional plugins and tools to enhance its functionality. Arduino boards primarily use the Arduino programming language, which is a simplified version of C/C++. Arduino language and the Arduino Integrated Development Environment (IDE) make it easy for beginners to write code for various Arduino boards. The language is C/C++-based and provides many built-in functions for working with sensors, actuators, and other peripherals. Additionally, users can program Arduino boards using other languages and platforms, such as Python (MicroPython), JavaScript, Scratch, and C/C++. There is a large and supporting community available to help users learn and troubleshoot their projects. Just like the Raspberry Pi OS, it is a versatile platform that makes it an excellent choice for beginners and hobbyists [83].

The Arduino Web Editor, also known as Arduino Create Web Editor, is a cloud-based integrated development environment designed for programming Arduino microcontrollers. The Arduino Web Editor offers an innovative approach to Arduino microcontroller development by removing the need for local software installations and enabling online access to Arduino programming. Some key features and advantages of the Arduino Web Editor include cross-platform support, easy accessibility, and cloud-based storage for project files. Since it's a web-based platform, we can use it on various operating systems, including Windows, macOS, and Linux, as well as on Chromebooks. The Arduino Web Editor, as well as its cloud storage, is easily accessible from anywhere with an internet connection, making it an extremely convenient option for on-the-go programming. The absence of local software installations simplifies the onboarding process for new users. The web-based IDE provides real-time code compilation, integration with Arduino libraries, and collaborative features, allowing users to share and work on projects together. It supports multiple Arduino boards and third-party hardware, making it versatile for various applications. This platform has gained attention for its convenience,

cross-platform compatibility, and the ability to streamline the development process [84].

STM32CubeIDE: STM32 microcontrollers are typically programmed using STM32CubeIDE, an integrated development environment provided by STMicroelectronics. STM32CubeIDE is based on the Eclipse platform and is tailored for STM32 microcontroller development. It offers features like code editing, debugging, project management, and support for STM32 hardware configurations. STM32CubeMX is a graphical configuration tool that helps generate initialization code for STM32 microcontrollers. It can be used alongside STM32CubeIDE to configure peripherals, pin assignments, and middleware settings [85].

JetPack SDK: The NVIDIA Jetson Nano runs on the JetPack SDK, which includes various software packages and libraries to support AI and deep learning applications. JetPack includes the following key components:

- **CUDA Toolkit**: NVIDIA's parallel computing framework for GPU acceleration.
- **TensorRT**: A deep learning inference optimizer and runtime.
- **cuDNN**: NVIDIA's deep neural network library.
- **OpenCV**: A popular computer vision library.
- **Linux Kernel and Drivers**: Jetson Nano runs on a custom Linux distribution.

There is also NVIDIA Nsight Systems which is a profiling and tracing tool for analyzing the performance of GPU applications on the Jetson Nano [86].

Geany is a powerful and lightweight IDE. It is built to be small and fast, yet still provides useful features like syntax highlighting. It supports many common languages, including C, Java, PHP, and Python. It can be customized with plugins and even supports theme changes like dark mode. Geany should come preinstalled on the Raspberry Pi OS, making it an easy IDE to use.

Thonny Python IDE is another programming environment that should come preinstalled on the Raspberry Pi OS. Thonny is mainly focused on Python and has a more minimalistic design with fewer features than Geany. It has an interactive debugger with the ability to run programs step by step. Function calls open a new window that shows the local variables making issues with recursion easier to diagnose. Similarly, to Geany it highlights syntax errors. Thonny is a great choice, especially for beginners.

Visual Studio Code is another IDE we could use. It can be run directly on the Raspberry Pi or run on a desktop using a remote development plugin. VSCode is lightweight yet powerful and supports a wide range of languages. It doesn't come

preinstalled with the Raspberry Pi OS but is easy to install. It has numerous features like debugging tools, Git integration, and syntax highlighting.

## 3.12 Implementing Live Speech-to-Text Overview

No matter what computer and other components we choose, there's a common overview of the steps we need to take to implement live speech-to-text. First, we need to capture the audio using a microphone. Then we need to transform our audio data into text using one of two methods, onboard speech recognition and cloud-based recognition. After our speech has been transcribed, we then need to send the text to our display which may require us to create a graphical user interface (GUI), so the text is displayed nicely.

We may change modes, between the camera and microphone, using a switch or we may also create an app to handle this. This also goes for the language recognition and colorblind mode. Initially, we will get English and a certain set of colors to function correctly, and then work from there.

Optimizing speech recognition software for low-latency performance is crucial for real-time transcription and translation applications. We are choosing between a few high-quality ASR (Automatic Speech Recognition) services, such as Google's Speech-to-Text, OpenAI's Whisper, and Speechmatics, which offer robust and accurate recognition capabilities. If the chosen service allows, we will deploy our application in a data center geographically close to the target audience to reduce the time it takes to send audio data and receive results. Before sending audio data for recognition, there are a few operations that can help improve the signal quality and accuracy of recognition like noise reduction, echo cancellation, audio normalization, and applying audio compression using an audio processing library. We will also segment the audio into small chunks or streams, ideally of around 200-500 ms, and maybe utilize multi-threading to parallelize the recognition process (sending multiple audio segments simultaneously can reduce latency). If our chosen ASR service supports it, we can use WebSocket connections for streaming audio data. It provides a low-latency, bidirectional communication channel for sending and receiving audio. There will also be lots of experiments to adjust the buffering and chunk size to find the optimal trade-off between latency and accuracy (smaller chunks might reduce latency but can affect recognition quality).

### 3.12.1 Capturing Audio

The microphone we choose must be compatible with the computer we're using. This means checking for things like compatible voltage levels, how they interface such as with I2S, and their supported drivers. I2S is a serial bus interface used for connecting digital audio devices. It communicates pulse code-modulated (PCM) audio data between a controller and a target. Pulse code modulation is a



way to represent analog signals in a digital format. I2S microphones have 3 main pins; the clock, word select, and data. The clock synchronizes the data sent between the controller and the target. The word select indicates which channel, left or right, the audio is for. The data pin transmits the actual PCM audio data [67]. Computers such as the ESP32 and Raspberry Pi Zero 2 W support I2S, yet the code implementation for the same microphone will differ. It's important to check if the microphone we select has available drivers for our computer. Drivers dictate how the microphone hardware communicates with the computer.

When we capture audio, we may need to clean it up so that it sounds good when we try to transform it into text, aka preprocessing. For example, something we might have to account for is noise reduction or filtering out background noise. Another factor we may need to adjust, which is simpler to implement, is the gain which affects the audio signal's amplitude [68]. Audio preprocessing can be done in many ways which can be dependent on the computer. Computers like the Raspberry Pis are more powerful than an ESP32 and can perform more complex preprocessing. It's unlikely we'll need to perform a complex preprocessing method, however. If our microphone is good enough, preprocessing won't be necessary or will be minimal.

Microphones continuously stream audio, which may be difficult to manage, especially for a microcontroller. This is why we'll need to implement a buffer. In a buffer, audio is captured in chunks of predetermined size. This allows us to process one chunk of data while capturing more data, meaning that we don't have to wait for the speech-to-text conversion to finish. Buffers also ensure that audio data isn't lost if the system is too busy to process the audio. Something to consider is that larger buffers will take more time to process [69].

## **3.13 Speech-to-Text Services**

### **3.13.1 On-Board Speech-to-Text**

There are 2 main methods for converting our captured audio into text, onboard and cloud-based speech recognition. Cloud-based speech recognition would be done entirely on the computer with machine learning services such as TensorFlow Lite, and Edge Impulse, and speech recognition systems such as CMU Sphinx. The problem with doing speech recognition on computers like the ESP32 is these systems are limited in memory, processing power, and on-board speech to text will likely be too resource expensive. We'd need something more powerful like a Raspberry Pi. Additionally, onboard speech-to-text solutions are limited in vocabulary and accuracy. They can't translate speech with multiple languages in it into one language. That said, they have some significant advantages, those being no need for an internet connection or transmission latency, although processing latency would likely be significant. For our purposes, high text accuracy, which on-board speech recognition services may

perform worse at, is critical, so cloud-based speech recognition will be our preferred way to implement speech recognition. The need for Wi-Fi and the potential for large latency is a very limiting factor so we may experiment with on-board services or maybe a hybrid of both options. The following are some on-board speech-to-text services we could use.

CMUSphinx (or PocketSphinx): PocketSphinx is an open-source speech recognition system. It can be used offline and is free to use. It's very lightweight, making it ideal for computers with limited memory and processing power. It supports 16 languages; however, the quality may vary. We can add more languages, but we'd have to collect audio data with transcriptions and build our own models for each language. It has a wide range of tools like keyword spotting. CMUSphinx can be programmed with Python, C, and Java.

Vosk: Vosk is another open-source speech recognition system. It's free, supports over 20 languages, and can be used offline. It's more accurate than PocketSphinx. Again, to add more languages we'd have to train our models. It supports many programming languages including Python, C, and C++. It has features like speaker identification and vocabulary reconfiguration.

Kaldi: Another free, offline, and open-source speech recognition toolkit, Kaldi has a large online community. It supports C++ and Python. It has English and Mandarin language models, and we can add more by training our own. It's more accurate than Vosk at the cost of increased resource consumption.

Mozilla DeepSpeech: Mozilla DeepSpeech is free, can be used offline, and is open source. It's based on machine learning. It has English and Mandarin language models, and we can add more by training our own. It can be used with Python, NodeJS, C, and more. It's similar in resource cost and accuracy to Kaldi.

*Table 9. On-board Speech-to-text services Comparisons*

Feature	PocketSphinx	Vosk	Kaldi	Mozilla DeepSpeech
Can work Offline	Yes	Yes	Yes	Yes
Price	Free	Free	Free	Free
Relative resource cost	Low	Moderate	High	High
Programming languages	Python, C, and Java	Python, Java, C++, and more	C++ and Python	Python, NodeJS C, and more
Languages (no training needed)	16	20+	English and Mandarin	English and Mandarin

Accuracy	Fair	Good	Great	Great
----------	------	------	-------	-------

If we're going to implement just an offline speech-to-text engine, Vosk or Kaldi are likely our best options. If we wanted to implement a hybrid of offline and online speech-to-text, PocketSphinx or Vosk would be our ideal choices. Kaldi and Mozilla DeepSpeech are likely to be too resource-costly to implement on our device, despite their superior performance. They also have less pre-trained language models. PocketSphinx and Vosk provide a good balance between resource cost and accuracy. On-board solutions would be a good place to start, but to significantly improve the accuracy and available languages of transcriptions, we must use a cloud service.

### 3.13.2 Cloud-based Speech-to-Text

Cloud-based speech recognition is implemented using APIs. An API is part of software that provides a way for two applications to communicate with each other. This can be a function that communicates with hardware to light an LED or, more typically, some program/software communicating with an online service to obtain and transmit information. In our project, API refers to the methods of communication that an online speech-to-text service provides. We would use APIs to send audio data from our computer to cloud-based speech recognition services. In this method all the processing is done on a remote server, freeing up resources on our MCU. Cloud-based services offer very accurate speech recognition and support many different languages. There are some significant cons to consider when using cloud-based speech-to-text. One con is that a stable internet connection is required and transmission latency as well as potentially increased processing latency if a service is bad are introduced. Another is that online speech-to-text services will cost money the longer we use them, although services like Google Cloud speech-to-text offer a free amount of time. Despite these cons, cloud-based speech recognition services provide far more accurate transcription, their ability to transcribe text in real time can be superior to on-board solutions, (depending on the computer's processing power), and a wider range of languages are available to use. Something to consider when trying to transcribe speech that contains multiple languages into one language is we'll likely have to make 2 API requests. One to transcribe the speech in whatever language they're in, and the other to translate all the transcription into English, or any other language. This will introduce additional latency and may also require different cloud-based speech-to-text services which may require different APIs.

There are many Cloud-based speech recognition services. Some of the most prominent ones are Google Cloud speech-to-text, Microsoft Azure speech-to-text, IBM Watson speech-to-text, and Amazon Transcribe. The following is an overview of each service.

Google Cloud Speech to Text: Google's Cloud speech to text is a very popular and solid service that offers a wide range of features. It supports over 125 languages and can recognize many different dialects. Some of its features are streaming speech recognition, speech adaptation which is custom vocabulary and phrases, and noise robustness meaning it can handle noisy audio. It has good pricing at \$0.016/minute with 60 minutes free. It can be integrated easily through REST and gRPC APIs, making it compatible with most development platforms. Google's speech-to-text is widely used and has many references making integration and diagnosing issues easier.

Microsoft Azure Speech to Text: Microsoft Azure's speech-to-text service provides high-quality transcription with customizable vocabulary. It has a 300-minute free trial and good pricing at \$0.016/minute. The service supports real-time continuous recognition and can process long-duration audio files. Its software development kit (SDK) has support for many programming languages and platforms, making it a good choice for diverse development platforms. It supports transcription of over 100 languages. It can be implemented using the Speech SDK provided by Microsoft or REST APIs.

IBM Watson Speech to Text: IBM's Watson Speech to Text provides fast and accurate speech transcription. It supports 14 languages. Its pricing starts at \$0.02/minute with 500 free minutes. Some of its features are model training, speaker diarization which is the identification of different speakers, and word filtering for keyword spotting or profanity. Integration options include WebSocket and standard HTTP REST methods.

Amazon Transcribe: Amazon Transcribe, part of Amazon's web service, provides automatic speech-to-text. It has 60 minutes free and \$0.024/minute afterward. It can transcribe in 39 different languages and dialects and has SDKs for numerous programming languages including Python and C++. Some of its features are its ability to detect toxic audio, multi-language identification, and speaker diarization. It can also redact sensitive information like an address or email in real-time.

Speechmatics: Speechmatics is a speech-to-text service that emphasizes transcription and translation. It supports transcription in 49 languages and translation in over 30 languages. It provides 480 free minutes and then starts at \$0.0173 per minute. Some of its features are language identification, custom vocabulary, and speaker diarization. It can be integrated using WebSocket API. A very useful feature it has is the ability to transcribe and translate in one API call. This would likely provide the least latency when translating a transcription as compared to other services because they would need to make two calls.

*Table 10. Cloud-based Speech-to-text API Comparisons*

Feature	Google Cloud	Microsoft Azure	IBM Watson	Amazon Transcribe	Speechmatics
Free trial	60 minutes	300 minutes	500 minutes	60 minutes	480 minutes
Pricing	\$0.016 / min	\$0.0167 / min	\$0.02 / min	\$0.024 / min	\$0.0173 / min
Accuracy	High	Highest	Lowest	Highest	High
Languages	125+	100+	14	39	49

Something to note regarding accuracy is very difficult to accurately determine the accuracy of these services in comparison to each other. A lot of data is a few years old or potentially biased. The best comparison we could find showed that all of these services were above 90% accuracy as of March 2023, with their Word error rates, (WER), within a few percentage points of each [77]. This comparison was obtained by tests done by another speech recognition service that's not listed, and thus there are no corroborating sources. These accuracies aren't strict because one "less accurate" speech recognition engine could be more accurate than another based on the type of speech.

All these speech-to-text services are very capable, but we have to focus on using one and we can always choose a different one later. We will most likely use Speechmatics because it can transcribe and translate in one step. This will most likely provide the least latency. It may not be the most accurate and it may not support the greatest number of languages, but the reduced latency is critical, and the accuracy is good enough. Another disadvantage would be less documentation and examples of implementation online as compared to services like Google.

If we didn't want to use Speechmatics and wanted to transcribe spoken speech in a different language, we could use a service like Google's translation API for this step, which is separate from their speech-to-text service. It has similar API integrations to their speech-to-text, using REST and gRPC. It supports over 100 languages. The first 500,000 characters are free per month and then it's \$20 per million after that. This translation step would add extra latency.

### 3.13.3 UI

Displaying the result in a user interface (UI) can be accomplished using various UI frameworks and libraries, depending on the application requirements and

programming language. Here's an overview for displaying tracking results for speech translation and color detection:

The UI design should be minimalist and provide real-time visual feedback for both speech translation and color detection. We are considering using unobtrusive text overlays or visual cues. Translated speech can be displayed as text on the UI component. The text should be easy to read but not interfere with the user's view. Implement text resizing and positioning based on user preferences. When the glasses identify a color, the detected color information can be presented as color swatches, icons, or textual descriptions in the UI display. We need to ensure that the UI component updates in real-time as new translations or color detections occur and the users should see immediate feedback. We need to ensure that our selected display has supported libraries. These allow us to use commands to interact with the display [76]. Libraries can be obtained from sites like GitHub or from the manufacturer. They can be installed using a package manager or from the command line [75]. Library support will vary depending on the model of computer we use. Specific functions will vary from display to display.

When choosing a UI framework or library for smart glasses, we want to consider lightweight, efficient, and augmented reality (AR) development tools. Some options include:

- ARKit (for iOS): [87] If the glasses are iOS-based, ARKit can be used to create AR experiences.
- ARCore (for Android): [88] For Android-based glasses, ARCore provides similar AR capabilities.
- Unity 3D: [89] Unity is a versatile platform for developing AR applications that can run on various smart glasses.
- Vuforia: [90] Vuforia is an AR development platform that supports multiple platforms.
- OpenCV for UI Integration: [91] OpenCV can be used in combination with other AR libraries for UI integration.
- LVGL (Light and Versatile Graphics Library): [121] An open-source graphics library with a focus on lightweight and high-performance GUI development for embedded systems.

Regarding the user experience, we considered incorporating voice commands, hand gestures, or head movements as methods for users to interact with the UI component and access additional information or settings as our stretch goals. We could also develop a mobile app to further expand the serviceability of our glasses, it is currently one of our stretched goals. Extensive testing is required to

ensure the UI components function correctly and provide a positive user experience. User feedback will be collected and used to refine the design further.

## **3.14 Integrating Speech-to-Text API**

The ways we can implement a speech to text service depends on the service and computer we choose. One way we could start is with Google Cloud Speech to Text. From here we could send the audio in batches, which is where we record a few seconds of audio, send it to Google's servers, and repeat the process. An advantage of implementing speech-to-text this way is it's simpler in comparison to streaming. A disadvantage would be increased latency, which depends on the batch size. If we record 5 seconds of audio and then send it to Google's servers, the minimum latency for a spoken word to be transcribed is 5 seconds. Larger batches could also take longer to send and process. Additionally, storing large chunks of audio is a concern on memory-constrained devices. To send data using the batch method, we'll need to use HTTP. The way we integrate the same service will differ depending on the computer and software we use [70].

### **3.14.1 HTTP**

HTTP is a request/response protocol that provides a way for users to interact with the web using HTTP requests. In an HTTP request, a user interacts with a client, in our case an ESP32, which builds and sends an HTTP request to a server using the address for Google Speech to text with our API key. Then the server reads the request and knows how to because it's formatted as an HTTP request. The server builds and sends an HTTP response. The client receives the response and can understand the information in it because of its HTTP format. The client then does whatever it needs to do with the information in the response, such as displaying it on a screen.

An HTTP request has a verb, aka method, and a path. The path tells the server what resource/information is being requested. The verb tells the server what it should do with that resource/information. There are several verbs, but the main ones used for CRUD (create, read, update, and delete) operations are POST, GET, PUT, and DELETE. The POST method creates a new resource with the information in the request. The GET method retrieves a resource. The PUT method updates a resource, although it can also create a resource. A key difference is that PUT requests are idempotent, meaning that sending the same PUT request multiple times will have the same effect as sending one. In contrast, sending the same POST request multiple times will create multiple of the same resource. Lastly, the DELETE method deletes a resource [71].

Our project will not interact with a database, so most of the CRUD methods aren't useful to us. The method we'll use when using HTTP for an online speech-to-text service is POST. We'll only be sending audio data and the

information contained in the responses object will be our transcribed text, so there won't be a need for another method such as GET.

### **3.14.2 HTTP Security and Privacy**

There are many security and privacy issues with HTTP. HTTP requests are sent in plaintext, which means that anyone monitoring the connection can read the information that is sent. Attackers could modify the unencrypted audio data that we send or the transcribed text we receive. Another issue, among many more, would be the inability of the client to verify it's communicating with Google's servers and not an impostor [72].

There are also specific issues to consider such as regulatory violations like the GDPR from the security perspective, sensitive information such as login credentials that are spoken and transmitted to Google could be intercepted when using HTTP. This is unlikely because sensitive information usually isn't spoken aloud, but it must still be considered. Another security issue would be the ability of an attacker to read and use our API key. This would allow them to communicate with Google as if they were us and incur charges on our account depending on the amount of audio data they send.

From the privacy side, any spoken data that is transmitted using HTTP could be eavesdropped on by third parties. This would allow an attacker to build a profile about the user(s), and further target them through methods such as harassment on social media. Additionally, response transcriptions sent by an attacker would be displayed to the user. This means they can alter the meaning of a speaker's words and cause confusion as well as produce inappropriate transcriptions [72].

### **3.14.3 HTTPS**

HTTPS has the same functionality as HTTP, but it encrypts the requests and responses. HTTPS also ensures clients are interacting with the intended server by verifying servers using SSL/TLS certificates. This makes HTTPS far superior to HTTP in terms of privacy and security. The problem is implementing HTTPS is more complicated than HTTP. This is because we have to set up the certificate for the server we want to connect to. We have to obtain a valid certificate, use that in the code, and also be aware of the certificate's expiration date [73]. These extra steps aren't a huge difference to implementing HTTP and are worth it for the extra security.

### **3.14.4 Streaming**

Instead of batch processing, we could stream the audio instead. This is similar to batch processing in the sense that we still implement a buffer to temporarily hold the audio, but this buffer is much smaller. The main advantage of streaming is it



provides lower latency. If we want to stream audio, we'll have to use gRPC. This is hard to implement on resource-constrained computers like the ESP32 and there isn't much information online on how to do it. Something we could do is send the audio data to a server that handles the gRPC interaction, but this would introduce extra latency. Streaming on the ESP32 using gRPC likely isn't feasible. We'd have to use a more powerful platform like the Raspberry Pi.

### **3.14.5 gRPC**

gRPC, (Remote Procedure Calls), is an open-source Remote Procedure Call Framework. Similarly, to HTTP, gRPC enables online communication but goes about it differently. gRPC is based on the idea of defining services with methods that can be called remotely. gRPC uses protocol buffers as its interface. These serialize structured data with more efficiency than formats like JSON, reducing latency. They use structures called messages that hold information using name-value pairs. There are four types of services. The most useful one to us is bidirectional streaming RPCs where the client and the server send a sequence of messages. In this service, the client and server operate independently meaning they can read and write messages in whatever order. Continuous bidirectional streaming means no need for multiple requests and responses, reducing latency [74]. gRPC supports many languages including Java, C++, and Python. gRPC is ideal for systems that require low latency, making it a great option for us.

## **3.15 Object Tracking and Color Detection**

Object detection is the task of identifying and classifying objects within an image or video by using bounding boxes. It combines image classification (assigning a label to an object) and object localization (identifying an object's position and size). Object detection methods can be neural network-based (e.g., YOLO, R-CNN) or non-neural approaches (e.g., Haar features, SIFT, HOG). Neural network methods are more accurate and are often built on convolutional neural networks (CNNs). Some of the technical and practical challenges for object detection include the use of rectangular bounding boxes, non-neural methods' accuracy limitations, and the need for large, annotated datasets. Since object detection using bounding boxes provides limited information about object shape. Objects with complex shapes, such as animals or irregular objects, are challenging to represent accurately. Some non-neural methods may not achieve high accuracy or may produce numerous false-positive detections, which can be problematic in applications where precision is crucial. Neural network-based methods, including deep learning models, require large amounts of annotated data for training. Annotated datasets are expensive to create and can be time-consuming.

For our project, we are more focused on real-time color detection. Real-time color detection and object detection are related but distinct computer vision tasks. While they share some common algorithms and concepts, they serve different purposes and involve different techniques. Real-time color detection identifies and tracks specific colors or color patterns within an image or video stream. Its primary goal is to recognize predefined colors or color ranges and respond when they are detected. Color detection is generally simpler than object detection and can be based on traditional computer vision techniques or color thresholding. Techniques include color-based segmentation, color filtering, and color-tracking algorithms. While there are differences between these tasks, real-time color detection can be considered a subset of object detection in certain contexts. In our context, the goal is to detect and track objects based on their color attributes, hence it combines the characteristics of both tasks. We will dive deeper into some of the algorithms mentioned below and learn more about their mechanics and use cases to achieve our objective and to take into consideration stretch goals and future enhancement.

### **3.15.1 You Only Look Once (YOLO)**

YOLO is one of the most popular real-time object detection algorithms known for processing images quickly and accurately. Introduced by Joseph Redmon in 2016, it offers end-to-end object detection in a single forward pass. YOLO is influenced by the GoogLeNet model and comprises 24 convolutional layers preceded by two fully connected layers. It predicts object classes and bounding box locations. Each bounding box is defined by its center coordinates, width, height, class, and probability of the prediction. How YOLO works is it divides an image into a grid and assigns a vector to each grid cell, representing its content. The object is identified if the middle point of a bounding box falls within a cell. YOLO handles overlapping bounding boxes by comparing their overlap scores with a threshold in the NMS unit. This enables one object detection per grid cell, reducing the need for sliding windows. Thus, YOLO can predict multiple objects per grid cell by using a vector with components for each bounding box and class. On the other hand, YOLO might predict multiple bounding boxes for the same object. Non-max suppression is used to select the box with the highest probability while removing redundant boxes based on intersection over union (IoU). YOLO v2 introduced anchor boxes for predefined bounding box dimensions, improving object localization. Subsequent versions (v3, v4, v5) incorporated advanced CNN architectures and features like CloU loss and more. Overall, YOLO is a fast object detection algorithm with applications in diverse fields, such as autonomous driving, security, manufacturing, sports, and various computer vision applications, but it sacrifices some localization precision for real-time performance. [92]

### 3.15.2 Region-based CNNs (R-CNNs)

Region-based Convolutional Neural Networks (R-CNNs) are pioneering approaches in object detection, including R-CNN, Fast R-CNN, Faster R-CNN, and Mask R-CNN:

R-CNN extracts region proposals (e.g., 2000) from an input image, label their classes, and determines bounding boxes. Each region's proposal undergoes forward propagation through a CNN to extract its features. Features and labels of each proposal are used to classify objects using support vector machines and predict bounding boxes using linear regression. R-CNN is accurate but slow and computationally intensive, making it unsuitable for real-time applications.

Fast R-CNN improves upon R-CNN by performing CNN forward propagation on the entire image instead of individual proposals, reducing repeated computation. It introduces the region of interest (RoI) pooling layer, enabling efficient feature extraction from region proposals of different shapes. Moreover, it transforms concatenated features using fully connected layers for class and bounding box prediction.

Faster R-CNN replaces selective search with a region proposal network (RPN), reducing the number of proposals without compromising accuracy. Mask R-CNN further enhances detection by introducing pixel-level predictions based on detailed labels.

Regarding color detection, Fast R-CNN might be the most suitable choice among these models as it offers a balance between accuracy and speed. Using R-CNNs for color detection offers the advantage of precise object localization and classification, making it suitable for fine-grained color recognition tasks. R-CNNs excel at providing detailed information about object regions within an image. However, trading speed for accuracy, they are slower than YOLO for real-time detection. [93]

### 3.15.3 Single Shot Detector (SSD)

Single Shot Detectors (SSDs) are a fast and efficient object detection method that utilizes a single convolutional neural network (CNN) to predict bounding boxes and class labels for objects in an image. Unlike other approaches, such as the R-CNN family, which involve multiple networks and stages, SSDs are known for their real-time capabilities, making them well-suited for applications like self-driving cars and surveillance systems. SSDs leverage a pre-trained base network, such as VGG or ResNet, for image classification and then add extra layers responsible for detecting objects at different scales. In summary, the SSD flow involves preprocessing an input image, passing it through a convolutional

neural network to extract features, applying anchor boxes to generate potential bounding boxes, classifying the objects within these boxes, using Non-Maximum Suppression (NMS) to refine the results, and finally outputting the bounding boxes and their associated class probabilities. The process allows SSDs to efficiently detect objects in real-time with a single pass through the network. This architecture allows them to achieve both speed and accuracy, especially when trained on smaller datasets.

In comparison, SSDs may not be as accurate as methods like R-CNN, and they can be sensitive to object scale variations in images. Between SSD and YOLO, the primary distinction lies in their approach to handling multiple bounding boxes of the same object. SSD uses fixed-size anchor boxes and considers the IoU metric with a threshold above 0.5. SSD employs different convolutional models for each feature layer. YOLO is limited to predicting only two boxes per cell and a single class per box. While SSD is more accurate, YOLO is faster, making it suitable for real-time applications. In summary, SSDs are a valuable choice for real-time object detection tasks that prioritize efficiency while achieving respectable accuracy. [94]

### **3.15.4 OpenCV**

OpenCV, a popular computer vision library, offers a range of functions and tools to process and analyze images and video streams. These algorithms can be quite efficient, making them suitable for real-time applications. Based on a numerous tutorials online, we have conducted an outline of a suitable approach for real-time color detection in OpenCV:

Our color detection process begins by translating the input image or video frame into the HSV color space. This allows us to define a precise range or set of color values, acting like a targeted net, to capture the specific hues we seek. Applying a color threshold then creates a binary mask, a stark black and white image where only pixels within our desired color range shine brightly. To refine this mask and eliminate noise or enhance color regions, we may employ morphological operations, which are like digital sculpting tools. Finally, we utilize contour detection to identify connected components within the mask, revealing the exact outlines and locations of the detected color regions. This information becomes the foundation for real-time tracking, allowing us to monitor and estimate their changes over time.

In conclusion, OpenCV provides functions, and many instructions as well as documents, for each of these steps, making it a great choice for real-time color detection. It allows the power of a popular computer vision library without the computational intensity of full object detection algorithms. [95]

### 3.15.5 Real-Time Processing

When it comes to real-time color detection, especially in a continuous stream, it involves some techniques to enhance the efficiency and accuracy of the detection process. We can simply utilize the simple color detection functions in OpenCV instead of the complex machinery of full object detection algorithms like YOLO or Faster R-CNN. A key strategy that we will consider is to focus detection on a defined region of interest to reduce the area to be processed. Since high resolution is not necessary for our application, input frames can be resized to a lower resolution. Another technique is implementing a ring buffer to capture and process a stream of frames. This can help to keep processing only the latest frames and discard older ones if needed. Training a machine learning model on the MCU such as a TinyML using Edge Impulse, is also a good choice as it can be optimized for speed and accuracy. Parallel processing, hardware acceleration, and optimized camera settings can be harnessed to maximize the efficiency of the color detection process. Asynchronous frame processing is something we keep in mind as it can help to reduce latency. Since we will display a GUI of the color detection, we can consider implementing a Real-Time Feedback Loop to update the UI accordingly based on network conditions and a Background Thread for Display to offload tasks such as image loading, network requests, file I/O, rendering UI elements, etc., to an independent background thread.

### 3.16 Data Communication Protocols

WebSocket: is a communication protocol that establishes an initial connection through a standard HTTP handshake. When a client intends to upgrade the connection to WebSocket, it sends an HTTP request to the server, which, if agreed upon, responds with an HTTP 101 status code, signaling the switch to WebSocket protocol. This upgraded connection enables full-duplex communication, where both the client and server can send data independently at any time, making it ideal for real-time applications. WebSocket data is transmitted in small units called frames, which can carry text, binary, or control data. This framed approach ensures efficient data exchange. WebSocket connections are long-lived, minimizing latency and enabling instantaneous data transmission, making it suitable for real-time chat, online gaming, live data feeds, collaborative tools, and more. Both client and server support for WebSocket is required, but with widespread browser and server support, WebSocket is a practical choice for a variety of real-time applications. [96]

MQTT (Message Queuing Telemetry Transport): is a lightweight and efficient messaging protocol designed for low-bandwidth, high-latency, or unreliable networks, making it an ideal choice for the Internet of Things (IoT) and real-time communication applications. In MQTT, devices communicate by publishing and subscribing to topics, acting as either clients or brokers. Clients publish

messages on specific topics, and other clients interested in those topics receive these messages. MQTT's publish/subscribe model minimizes the need for continuous connections, conserving both bandwidth and power, which is crucial in IoT scenarios where resources may be limited. Its simplicity, small footprint, and support for Quality of Service (QoS) levels make MQTT a popular choice for various IoT and real-time projects, ensuring reliable and efficient data exchange in constrained environments. [97]

REST API (Representational State Transfer Application Programming Interface): is a set of rules and conventions for building and interacting with web services. It follows the principles of REST, which is a software architectural style that defines a set of constraints for creating scalable, stateless, and easily maintainable web services. REST APIs use standard HTTP methods, such as GET, POST, PUT, and DELETE, to perform various actions on resources, which are typically represented as URLs. These APIs enable clients, such as web applications or mobile devices, to access and manipulate resources on a server over the internet. REST APIs are widely used for their simplicity, scalability, and compatibility with various programming languages and platforms, making them a popular choice for building web services and enabling communication between different software systems. [98]

Real-Time Transport Protocol (RTP): is a network protocol specifically designed for transmitting real-time data, such as audio and video, over IP (Internet Protocol) networks. It plays a crucial role in applications that require low-latency and synchronized multimedia streaming, like VoIP (Voice over IP) calls, video conferencing, and live video broadcasts. RTP breaks multimedia data into small, manageable packets and adds timestamp and sequence number information to maintain synchronization and order. Additionally, it pairs with the Real-Time Control Protocol (RTCP) to provide feedback on data quality and network performance. Together, RTP and RTCP ensure that real-time multimedia content can be delivered seamlessly over networks, prioritizing timely arrival and minimal delays. [99]

Here is how we can implement these protocols into our project:

- WebSocket: WebSocket is a full-duplex, bidirectional communication protocol that allows real-time data transfer and is suitable for both speech-to-text and color detection updates.
- RTP (Real-Time Transport Protocol): RTP is designed for real-time multimedia streaming and is ideal for transmitting audio data in real-time. You can use it for speech-to-text translation.
- MQTT (Message Queuing Telemetry Transport): MQTT is a lightweight publish-subscribe protocol that can be used to transmit control commands or

configuration data between your smart glasses and the cloud service, reducing the need for continuous polling.

- REST API: For occasional or non-real-time data transfer, RESTful APIs can be used to configure your smart glasses, send commands, or retrieve information like translations and color analysis results.

### 3.17 Storage and Data Management

The XIAO ESP32S3 Sense has Flash memory, a non-volatile storage medium that retains data even when the power is turned off, (8MB) and PSRAM (8MB). Pseudo-static RAM (PSRAM) is a type of dynamic random-access memory (DRAM) that acts like static RAM (SRAM) but with a higher density. When working with an MCU module like the XIAO ESP32S3, we can utilize the Flash memory to store non-volatile, essential data and program code, while PSRAM enhances the device's capabilities by offering additional dynamic memory for more demanding computational tasks such as real-time processing action. Here are some details of how audio data, text translations, color tracking data, and user preferences can be stored on the board:

Audio Data: The audio data recorded for speech recognition will be temporarily stored in RAM during the recognition process. Once converted to text, it can be discarded to save RAM space since the text is what's needed for translation. If necessary, a circular buffer can be used to manage limited RAM.

Translated Text: The translated text will be stored in RAM for real-time display or audio output. However, to prevent excessive RAM usage, a FIFO (First-In, First-Out) buffer can be implemented to manage the most recent translations while older translations can be pushed to non-volatile storage.

Color Tracking Data: PSRAM can be utilized for efficient image processing. When performing real-time color tracking, the captured image can be loaded into PSRAM for rapid access and processing. Algorithms for color tracking can analyze the image data directly from the PSRAM. This is beneficial when quick analysis and decision-making are required, as RAM allows faster read and write speeds compared to Flash memory.

User Preferences: Individual user profiles and configuration files can be stored in the Flash memory, providing quick access to user preferences. SRAM can be used for efficient handling of some user-specific data.

## 4.0 Related Standards and Realistic Design Restraints

### 4.1 Engineering Standards and Design Constraints

#### 4.1.1 Optical Standards

Standards are used in many different products and applications to maximize the quality, safety, and reliability of the products and their processes. The following standards covered in Table 9 are from published documents that determined the conditions and requirements of a process to promote and increase the efficiency of processes, products, and materials.

*Table 11. Optical Related Standards*

Standard Name	Description
IEC 62341-6-1	Organic Light Emitting Diode (OLED) Display-Measuring Methods of Optical and Electro-Optical Parameters
ANSI/CAN/UL 8400:2023	Virtual Reality, Augmented Reality and Mixed Reality Technology Equipment
DS/IEC 63145-1-2:2022	Eyewear Display – Part1-2: Generic-Terminology
DS/IEC 63145-20-20:2019	Eyewear Display- Part 20-20 Fundamental Measurement Methods – Image Quality
IEC 63145-22-10:2020	Eyewear Display- Part 22-10: Specific Measurement Methods for AR Type – Optical Properties
ASTM D7195 – 21	Standard Guide for Setting Object Color Specifications
ASTM E2544 – 11a	Standard Terminology for Three-Dimensional (3D) Imaging Systems
ISO/TS 17321-1:2012	Graphic technology and photography — Colour characterisation of digital still cameras (DSCs) — Part 1: Stimuli, metrology and test procedures
ISO/TS 17321-2:2022	Photography — Electronic still picture imaging terminology — Part 2: Other defined terms
ISO/TS 17321-3:2017	Graphic technology and photography — Colour characterization of digital still cameras (DSCs) — Part 3: User controls and readouts for scene referred imaging applications



ISO/TS 17321-4:2022	Graphic technology and photography — Colour characterization of digital still cameras (DSCs) — Part 4: Programmable light emission system
ISO 12233:2023	Photography — Electronic still picture imaging — Resolution and spatial frequency responses
ISO 19264	Standardization of Image Quality Analysis

IEC 62341-6-1: This is a 2022 revised international standard of OLED displays prepared by IEC technical committee that replaces the previous 2017 version. It identifies the criteria for assessing the techniques and specific measuring requirements used to determine several limitations of the display unit. This includes measuring parameters for power consumption, response time, luminance, contrast ratio, resolution, color, and viewing angle of the display module. The standard provides manufacturers with the requirements of how testing needs to be conducted, including the equipment necessary and the correct test patterns to follow.

Using an OLED display has a great impact on us and is heavily dependent on selecting the right product that will be compatible with other components and have the essential specifications needed to project a great quality image that is clear and legible to the user. Brightness is an important factor especially when considering loss due to collimating lenses that will be used. It also involves the correct power consumption to make sure that all components run smoothly and that our circuit does not have unnecessary stress that may affect other elements. We also have to consider the response time as this project deals with live audio and displaying the text promptly.

ANSI/CAN/UL 8400:2023: This is a safety standard that focuses on electrical and electronic equipment used for virtual reality, augmented reality, and mixed reality devices. It directly states that the rated voltage must not be above 600 V in any AR/VR/MR technologies, relating to smart glasses, holographic displays, and head-mounted displays. The standard intends to diminish health issues that arise with the use of these types of technologies. These problems include motion sickness that is generated by visual effects, skin sensitization, flicker, visual opacity, biomechanical stress, optical occlusion, and heat exposure to the eye. It also focuses on the concerns such as thermal burn, electrical shock, and fire. Designing a pair of smart glasses can be difficult to create in a compact size, especially when implementing additional components that require a lot of power. This standard is very important because we must consider all the parts that will be used and their power supply requirements. We will need to make sure that all components are powered correctly to operate safely and reliably without any failure in our system. We must also understand the impact our product will have on users, as it may cause vision problems, fatigue, eyestrains, etc. It can even

lead to epileptic seizures from visible flickers, where this short-term exposure has a long-lasting effect on a person's life.

CAN/CSA-C22.2: This standard addresses the correct terminologies that are used for eyewear display in AR/VR/MR and the images or videos shown. This contains terms for different types of electronic displays, optical display systems, performances, visual ergonomics, etc.

The standard does not affect the design of our project because this only refers to terminologies, which do not alter our physical prototype. Distinguishing between different terms is important for research purposes and understanding the characteristics and parameters. It will make sure that we are selecting the right components that maximize efficiency.

DS/IEC 63145-20-20:2019: The standard concentrates on the measurements of image quality, such as distortion, focal distance, color, Michelson contrast, and the field of view and eye box based on the contrast. In addition, it also involves optical characteristic measurements for luminance, uniformity, and pixel angular density. It includes images and test patterns to consider examples, as well as systems and procedures for calculations.

These standards are very important to the designer because each parameter affects the quality of the image provided to the user. The selected design and display will impact each specification and determine whether the result falls within the limitations.

IEC 63145-22-10:2020: This standard specifies the measurement systems for determining the optical performance of the display with contrast modulation test using ambient lighting conditions. It is also for measuring color difference and spectral directional transmittance, which is the efficiency of a surface to transmit radiant energy at a specific wavelength.

ASTM D7195 – 21: This standard, provided by the formerly known American Society for Testing and Materials (ASTM), is a guide that provides users a list of sub-standards that have ties to the overall process of color specifications and the decisions needed for said color specifications whether it's for visual (like a camera) or instrumental methods). This guide also includes in-depth steps for each of the corresponding standards, the exceptions, intended warnings for the users, and how to finalize reports. The main significance of this guide is to provide a constant color specification standard to manufacturers to avoid common goods being rejected due to product color not meeting expectations. When meeting specific color standards for the user, the guide references two other sub-standards that give a customary process on the selection of observers (Guide E1499) and guidance on how to perform critical visual observations (Guide E1808). The guide also mentions the varying advances in color accuracy and measurements for visual applications and instruments, which should be

clarified by its calculations within the industry (Practice D2244). It is also important to not only evaluate the color by company standards but by address the customer specifications as well, allowing both the producer and customer to agree on a target color and adjust accordingly. With many of these tests there also arises some uncertainty, in which the process to estimate the uncertainty within the results is established within substandard Practice E2867 and Practice E1345 to evaluate multiple colors on a specimen or multiple specimens. Every color specification goes through the process of setting the tolerance, which includes the importance of this process (Practice D3964 and Guide D5531), the primary steps of using the original material, and laying out a wide range of sampled visually evaluated to compare and established standard tolerance (Practice D4086). Testing the tolerance with instrumental usage is also provided within the sub-standards (Practice E1164 or Test Method E1347) which goes more into detail on the utilization of spectrophotometry and filter colorimetry. The guide mentions that these tolerance practices are commonly presented in the LCh space, which are measured values from the spectral reflectance factor converted to CIE colorimetric values within the CIELAB space (Practice E308). A standard color spacing technique is the box tolerancing, more commonly known as rectangular tolerancing or elliptical tolerancing, to distinguish the product's color position. The guide finalizes the report procedure either by any instrument techniques used to finally establish the material's color specifications (Practice E805) or if color evaluation is exchanged between the producer and user electronically (Practice E1708) [60].

This standard guide is the main source of reference for our project as one of our main objectives is to establish a color blindness mode for our glasses that will satisfy any user's needs. This guide references multiple sub-standards that go into more detail on some procedures the glasses must conduct to satisfy the users, with standards that involve how to approach willing users, how to properly receive feedback, and how to properly adjust the color specifications to multiple users. The glasses, but more specifically the camera, should also be evaluated for how much color tolerance is built within the CMOS sensor and how much color it can truly capture. The wider the color range, the greater the satisfaction the camera will receive for a wider audience.

ASTM E2544 – 11a: This is another ASTM standard that lists out the common terminology used to define terms and acronyms associated with three-dimensional (3D) imaging systems. Terms that are defined into greater details here that correlated to the project include pixel, accuracy of measurement, calibration, error (of measurement), limiting conditions, backward compatibility, camera image, field of view (FOV), and range resolution [61].

ISO 17321-1:2012: This standard was established by the International Organization for Standardization (ISO) and established the standardization of how color channels analyze colors within digital still cameras. Due to the

differences that human observers have in identifying colors compared to how colors are analyzed with instrumentation of the CIE colorimetric standard, it is important to establish the spectral sensitivity, illumination, and color encoding space for digital still cameras. This standard goes into detail on the general flow of how a digital still camera should process an image (from its workflow to the exact numerical values for color positioning) and makes important notes on how certain operations must be noted for each digital still camera model (like the color pixel reconstruction and white balancing). The overall purpose of this standard is to allow an easier flow of characterization and testing of color analysis within digital still cameras for manufacturers to smoothly replicate and modify while retaining graphical technological application standard measurements [62].

ISO 17321-2:2022: This standard, established by ISO, is to reduce the confusion of standard video recording and camera photography terminology when measurements apply electronically and not through the process of “film”. Many of the terms established within the document are also references from other standards, making this standard a listed source for terms that should be defined with products that have any relevance to electronic still picture imaging [63].

ISO 17321-3:2017: This standard, established by the ISO, defines the main types of photography digital cameras use since pictorial photography is such a broad field for producers to home in on. The two main types of images produced are output-referred representations which are meant to deliver a “complex artistic vision” conveyed by the photographer and scene-referred (SR) images which are meant to accurately capture colorimetric images. The main purpose of this document is to establish a scene-referred (SR) capture-processing mode standard to allow users to properly use digital still cameras to capture colorimetrically accurate images [64].

ISO 17321-4:2022: This standard, established by the ISO, describes the standards of a programmable light emission system and devices like LEDs. It details the background information of how the system's spectra are made and the combinations of light being used to create a spectrum that matches the desired reference spectrum with its preferred evaluations and descriptions [65].

ISO 12233:2023: This document, revised and established by ISO, helps standardize the terms and measurements of resolution and spatial frequency response (SFR) for digital still cameras. It goes into detail on the proper monochrome and color camera measurements for it to output digital or analog data [66].

ISO 19264: This standard, established by ISO, details the process of standardizing the intake from a variety of image qualities received from all sources of visual and instrumental sources and evaluating each different sources and methods to analyze each image source. While previous standards have

been established before, this standard updates the technical specifications to analyze image quality testing and how to properly discuss the meaning behind each kind of analysis and methodology [100].

A few of these previously stated standards relate to some aspects of how the camera will detect color, translate the input data from the CMOS sensor to the MCU, and how the MCU will process the given input data to execute the required program for the micro-OLED.

The optical properties standard will impact our project because parts of this standard are important for the display as well as the optical combiner used. It is critical to have the right technologies, especially our optical combiner, which will affect how much light in our display is transmitted and reflected.

#### **4.1.2 Software Standards**

ISO/IEC 25010:2011: part of the Software Product Quality Requirements and Evaluation (SQuaRE) series, is a comprehensive international standard that provides a framework for evaluating the quality of software products and systems. It defines a set of quality characteristics and sub-characteristics that can be used to assess and measure the quality attributes of software, including functionality, reliability, usability, performance efficiency, compatibility, security, maintainability, and portability. ISO/IEC 25010 is widely recognized in the software industry and serves as a valuable reference for organizations and software professionals looking to define, evaluate, and improve the quality of their software products. It offers a structured approach to understanding and assessing software quality, making it an essential resource for ensuring that software applications meet the desired standards and user expectations [101].

ISO/IEC 25062:2006 (Common Industry Format for Usability Test Reports): is a standard developed to streamline and standardize the reporting of usability testing results. Usability testing is a critical component of user-centered design processes, allowing organizations to evaluate the effectiveness and efficiency of their products from a user's perspective. This standard defines a common structure and content for usability test reports, making it easier to share findings, compare results, and improve the usability of products and systems. By adhering to ISO/IEC 25062, usability professionals and organizations can ensure that usability test reports are more consistent, informative, and actionable, ultimately developing more user-friendly and effective products [102].

ISO/IEC/IEEE 12207:2017 (Systems and Software Engineering - Software Life Cycle Processes): is an international standard that outlines the processes involved in the software life cycle. It provides a comprehensive framework for software engineering and system engineering processes. ISO/IEC 12207 covers

the entire software development life cycle, from concept and initiation through development, operation, and maintenance to retirement. This standard defines processes and activities, their outcomes, and their relationships, making it a valuable resource for organizations and professionals in the software and systems engineering fields. ISO/IEC 12207 helps ensure that software projects are well-structured, efficiently managed, and result in high-quality software products. It provides guidance on the processes required for the effective development, delivery, and maintenance of software, which is essential for achieving successful software projects in various industries [103].

IEEE 730 (Software Quality Assurance Processes): is a standard that addresses Software Quality Assurance (SQA) processes. It outlines the requirements and recommendations for establishing and managing SQA activities within a software development project. The purpose of IEEE 730 is to ensure that software products meet the specified quality standards and conform to the established processes and procedures. This standard defines the objectives and activities of the SQA process, emphasizing the need for planning, documentation, and verification throughout the software development life cycle. IEEE 730 provides guidance on creating a quality management system, conducting quality audits, and managing non-compliance issues. By adhering to IEEE 730, organizations can enhance the overall quality of their software products and maintain consistency in their software development processes [104].

ISO 9241-210:2019 (Human-Centered Design for Interactive Systems): is an essential standard that focuses on human-centered design principles for interactive systems. It provides guidance on ensuring that interactive systems, which include software and hardware interfaces, are designed to be user-friendly and meet user needs effectively. The standard emphasizes user-centered design processes, including the involvement of users throughout the design and development phases. ISO 9241-210 helps organizations create products and systems that are intuitive, efficient, and satisfying for users, ultimately leading to improved user experiences and increased user acceptance. It's a critical standard for designing technology and software that aligns with human capabilities and preferences [105].

IEEE 802.11 (Wi-Fi Standards): refers to a set of wireless communication standards developed by the Institute of Electrical and Electronics Engineers (IEEE). These standards govern the technologies used for wireless local area networking (WLAN) and enable wireless connectivity in various devices, such as laptops, smartphones, routers, and IoT devices. The IEEE 802.11 family includes a series of specifications that define different aspects of wireless communication, including data transfer rates, frequency bands, security protocols, and more. These standards have evolved over the years, with versions like 802.11b, 802.11g, 802.11n, 802.11ac, and 802.11ax, each offering improvements in terms

of speed and performance. Adherence to IEEE 802.11 standards for Wi-Fi can ensure compatibility and performance [106].

Agile/Scrum: is a popular and flexible approach to software development that emphasizes iterative, incremental, and customer-focused processes. It was developed as a response to traditional, heavyweight project management methodologies. Scrum is one of the most widely used frameworks within the Agile approach, and it organizes work into small, time-boxed iterations called "sprints." During each sprint, a cross-functional team works on a prioritized set of features or user stories, producing a potentially shippable product increment by the sprint's end. The Scrum framework includes roles like the Product Owner, Scrum Master, and Development Team, as well as various ceremonies like Sprint Planning, Daily Standup, Sprint Review, and Sprint Retrospective. Agile/Scrum promotes adaptability, collaboration, and continuous improvement throughout the software development process. It is widely used in various industries for its effectiveness in delivering value to customers and managing change in complex projects. They can be beneficial for managing software development in dynamic projects [107].

ISO/IEC 27001:2022 (Information Security Management System): is an international standard that outlines the requirements for establishing, implementing, maintaining, and continually improving an Information Security Management System (ISMS). An ISMS is a systematic approach to managing sensitive company information, ensuring its confidentiality, integrity, and availability. ISO 27001 provides a structured framework for organizations to identify and manage information security risks effectively. It includes processes for risk assessment, security policies, procedures, and controls, as well as ongoing monitoring and improvement. ISO 27001 is widely recognized and adopted by organizations to protect their data and ensure compliance with various data protection regulations. It is a crucial standard for those seeking to safeguard sensitive information and maintain the trust of stakeholders and customers [108].

The National Institute of Standards and Technology (NIST) Cybersecurity Framework: is a comprehensive set of guidelines and best practices aimed at improving the cybersecurity posture of organizations. It was developed by NIST, a federal agency within the U.S. Department of Commerce, and provides a structured approach to managing and reducing cybersecurity risk. The framework is built around core functions: Identify, Protect, Detect, Respond, and Recover. These functions help organizations categorize and prioritize their cybersecurity efforts, emphasizing risk management and continuous improvement. The NIST Cybersecurity Framework is widely recognized and used by both government and private sector entities to enhance their cybersecurity resilience and protect against evolving cyber threats. It serves as a valuable resource for organizations seeking to establish, develop, or optimize their cybersecurity programs [109].

PEP 8 – Style Guide for Python Code: is a widely used guide that provides coding conventions for Python. It goes into great detail on the ways to stylize Python code, which includes line indentation, bracket alignment, white space, comments, naming conventions, and much more. Following this standard provides improved readability and consistency to a programmer's code. Having a standard that everyone follows makes working with code easier, and faster, and is especially important when working in a team [110].

## **4.2 Design Restraints**

### **4.2.1 Project Requirement Restraints**

When designing our AR pair of glasses, image quality and the compactness of the device have been important. With this in mind, we face some limitations such as weight, size, image clarity, efficiency, and eye box. As previously mentioned, some of these technologies that enable us to create a slimmer and more compact pair of glasses are not available to be purchased. This leads to weight and size being a significant concern. When designing with the components that are accessible to us, we must consider how it will affect our image and the efficiency of our display without chromatic aberrations or dissipating heat onto our user. Another factor we must consider is the latency of the information displayed. As close to real-time as possible is our goal because information that's displayed a long time after someone talked or a color was detected isn't useful.

### **4.2.2 Economic Restraints**

The economic restraints we will be facing deal with limitations due to external economic influence over our project. The entire project will be funded by the team, and it is crucial to communicate each component that will be purchased to ensure that performance and quality are essential for the project. We must share alternatives and ways that can help amplify the efficiency of the technology. The amount of money needed for this project can add up very quickly due to a lot of expensive parts needed for this project. The speech-to-text services cost a little over a dollar per hour of speech transcribed once the free trial has ended. This means we should use efficient testing methods. The display modules needed for our project are quite complicated to manufacture, due to their high requirement for quality and precision. They also require a lot of materials to make these displays, resulting in their prices increasing rapidly. They are in high demand for other applications, which makes them even harder to receive and not as affordable for the average consumer. This results in AR glasses on the market costing thousands of dollars. This type of technology is not quite ready to be mass-produced since it is still in its early stage of development. The design that we choose to select will not only affect the optical tradeoffs for the text display but also the cost required for the set design. For example, one of the main



components of text display is the optical combiner. These types of waveguides are produced by companies that have also produced smart glasses and are not selling the waveguides to the average consumer. This leaves us in a difficult position to find alternatives that may give us close to the result that we need.

### **4.2.3 Time Constraints**

Time restraint is another hindrance that plays a major role in our project. We have been able to tackle this by meeting regularly to keep everyone motivated and on track. We have also been breaking the assignments into smaller parts to make sure it is easier to follow without feeling overwhelmed. We make sure to give ourselves an earlier deadline and then come together as a team to put everything together and help one another as needed. Every member needs to meet deadlines because that may affect everyone else's work. We need to communicate with each other, especially about components that may affect another member's work. Another main concern is receiving our components in a timely manner due to the high demand for the parts needed for our design. We also don't have our electrical engineer, which leaves the rest of the team with additional issues we may not be able to easily resolve. We must utilize and familiarize ourselves with the resources available on campus. In addition, we must be willing to dedicate more time to learning any new methods or technologies that will need to be implemented into our design.

### **4.2.4 Health and Safety Restraints**

Near-eye displays such as smart glasses can have a few potential impacts on our health with extended day-to-day use. Their use of smart glasses causes eye strain; this occurs by trying to focus on the display. Visually induced motion sickness is usually found in VR, but it has been linked to some AR glasses, like Microsoft HoloLens. This leads to discomfort, dizziness, and nausea due to a limited field of view. It's important to make sure we power and wire our components properly. This would prevent short-circuiting and overheating. The batteries we use must be properly handled and protected. For example, improper handling of LiPo batteries can result in fire, explosions, and toxic smoke.

### **4.2.5 Manufacturability and Sustainability Restraints**

The manufacturability of our project is mainly due to affordable and accessible parts. If we were to manufacture most of the parts like most companies do, it would cost us greatly to produce. This explains why companies charge thousands of dollars for the products that are currently on the market. There's a high demand for micro-displays for other wearable device applications. The parts used in these products are sometimes custom-made to fit specific requirements. Our device should be able to be used indoors and outdoors, which means that

we need to use a framework out of 3-D printed material that will be able to house all the parts. We have to make certain that all components run smoothly and can run for a long period.

#### **4.2.6 Social Restraints**

In the social world, AR glasses can become a distraction for day-to-day life. Although smart glasses bring necessary apps to the user's eye. It can still be difficult to focus on reality. Some of the current technologies also have a large form factor, which cannot always be worn. In the right setting, it can help greatly in some industries for its visual analysis capabilities. It can help guide the user in completing a task and provide the information needed. Inconspicuousness is important when considering that these glasses will have a microphone and camera. Some people wouldn't like to feel like they're being recorded.

#### **4.2.7 Ethical Restraints**

One of the main problems that AR glasses currently face is privacy concerns, especially with glasses with cameras, such as Google Glass. It allows the user to record with a simple voice command, which can result in others being recorded without their knowledge. To record someone without having their consent can lead to legal actions or consequences which may vary from place to place. These types of glasses collect data such as images and videos from the user's surroundings, which are then transmitted to servers for processing. In our case, we will also be recording the user's surroundings for color detection purposes. Audio will also be recorded and processed to provide live-captioning and translation to the user. An important note is that we won't be permanently storing this data as if it were to be accessed later, but it will be temporarily stored for processing and display. We must employ secure data transfer protocols because sensitive information is still susceptible to third-party attacks. As an example, HTTP would not be ideal for this project because login credentials and user-identifying information could be transmitted. Another area where the security and privacy of user data applies is in the legal aspect. While we wouldn't be subject to laws such as FIPA, and the Florida Information Protection Act, it's still good to follow legal standards.

Another ethical restraint we must consider is how we will work with each other. We must respect each other's values, opinions, and character if we're to work well together. We expect everybody to pull their weight and respect the deadlines and meetings we set. If someone is stuck on an issue or temporarily can't work on something, we expect other teammates to be understanding and help them out. Practicing good ethics in a team is critical to creating a project on time that we can be proud of.

## 5.0 Comparison of Chat GPT

### 5.1 ChatGPT

ChatGPT, is a new AI tool model that has been widely used and certainly changed the way of learning. It can be of great use when used correctly; however, it has been abused severely by everyone. It pushes us to be lazy and allows the AI to do all the work without any additional research and critical thinking. Chat GPT can help in some ways, it can help in enhancing learning with its 24/7 availability to quickly clarify a topic or concept. It can create separate conversations and use context from previous inputs and outputs to change how it responds.

We have asked ChatGPT to provide a list of senior design project topics for two CSs and two PSEs. It did not understand the abbreviation for photonic students. It is believed that it was referring to power system engineers. It was able to populate a list of different project ideas that were quite interesting such as a LiDAR system for autonomous vehicles, a character recognition system for ancient texts, and an augmented reality headset. Although Chat GPT can be repetitive at times, it was able to provide a new list of project ideas that even when prompted again, and even with examples of exact conditions that the team can use to build their device and specification.

It is evident that Chat GPT is not able to coherently put together a research paper, but it can be used to provide an outline that can make it easier to organize our thoughts and research together. It frequently recapitulates the information given in the prompt when it is not able to process the information. One of the downsides is that users are not willing to double-check the work of Chat GPT and comprehend that it is not a perfect system. It is not always able to provide accurate information because it cannot fact-check the information it produces. It also doesn't have access to the latest information.

It cannot create a visual representation of any diagram for the user. It is also not able to process information through other software such as Microsoft PowerPoint. It is not possible to create the diagram using Microsoft PowerPoint for the user, but it is interesting to note that it does its best to prepare some sort of description of each diagram in text form. It creates a table that can be used for the house of quality to easily add it to an actual house of quality representation including all the ratings. It warns the user that the examples given are a lot simpler and should be used to clarify and further our understanding by adding more details that will be more tailored to the project. It can separate the customer requirements and engineering specifications using the correct legends given in the prompt, but it does not understand how to compare the positive and negative correlations to the requirements. It treats each specification as separate and generates a correlation for each without relating them to one another.

We also noticed that it is very sensitive to the order of how a question is phrased, which can make it difficult and frustrating for users. This can lead to inconsistencies in the answers it provides because it is interpreting the question differently each time. Once it was prompted to regenerate a house of quality and correlate each specification to one another, it was able to make a better table with the specifications in relation to one another. It is important to always double-check the work of Chat GPT as it is prone to make mistakes.

When prompted for code examples, ChatGPT can provide detailed code. One problem with using code provided by ChatGPT is the programmer may not fully understand it. For smaller programs and functions, it can provide fully functional code, which makes it easy to copy, paste, and then move on. The problem comes when trying to fix bugs and implement the code in other areas. The programmer's lack of understanding of ChatGPT's code makes it significantly harder and takes longer to work with. Thankfully ChatGPT is also great at walking through what code does which can improve the programmer's understanding and skills. When asked how to implement gRPC on a Raspberry Pi, it first started with some prerequisites to consider like an internet connection and a properly set up API key. Then it provided some commands to input into the Raspberry terminal which was for setting up gRPC and Google speech-to-text. When asked to elaborate on how it got these commands, it provided official documentation and detailed what each part of the commands does. Then it gave a simplified Python script that uses gRPC for transcribing speech. It's difficult to verify the functionality of the code with a lack of experience coding in Python. When asked to walk through what the code does, it provided a detailed, section-by-section analysis. When asked if there were any errors in the code, it noted some of the details that were missing because it was a simplified snippet, and also a few incorrect implementations such as with Google's SpeechRecognitionConfig. Something it didn't provide but hinted at how to start was capturing audio. These prompts demonstrate how ChatGPT is useful in getting started on a program. It can elaborate on the information it provides to help the programmer's understanding. It also shows how it will give errors which is important to consider when using the code it gives.

ChatGPT can fail when asking about very specific details. When using it to debug, it can catch more obvious errors like syntax errors and small logical errors, but it will struggle with more complex code. It's very dependent on the context and code we provide it. It can also struggle if an issue is specific to the hardware or software that we use. ChatGPT is great at providing a pseudocode skeleton which is good for recognizing what functions need to be made.

As mentioned before, Chat GPT is an amazing AI Large Language Model (LLM) to utilize to help kickstart project ideas. Like trying to master the tool of a calculator, Chat GPT can be used as a great tool unless misused. For instance, it should be well noted that much of Chat GPT's outputs are from multiple sources

it pulls from its limited database. This is why giving Chat GPT the prompt "...with the links or sources" many of the times will leave the LLM to pull dead links that no longer work or sources that are long outdated. Chat GPT also suffers from being limited with what kinds of sources it is allowed to generate outputs from, with the example of it being incapable of reviewing direct articles and must rely on the input summary the reader gives which of course cannot generate the most accurate information. It's important to understand the input limitations so users can generate their desired outcome.

While Chat GPT has its limitations in providing information to users, in many cases it can give a broad knowledge of topics in a variety of applications. It can be used at any time of the day and with different accessibility for users. It is also a great way to enhance and assist with learning different materials. With the consistent improvement of Chat GPT, it may be able to get to the point where we will be able to fully depend on AI. For now, we need to continue to push for critical thinking and problem-solving in doing our own additional research. We cannot blindly follow every piece of information we are given by Chat GPT as factual because it is essentially compiling information found on the web; therefore, we must investigate further to differentiate between correct and inaccurate data.

## 5.2 Aria

Aria is another AI chatbot. It's based on the Generative Pre-trained Transformer (GPT) architecture just like ChatGPT. It's free and built into the Opera browser making it very easy to use. It has context awareness and separate conversations. When compared to ChatGPT, Aria seems less capable, and its responses seem to understand certain prompts less. When giving both models the same prompt of "provide the most recent speech recognition accuracy comparisons", Aria said the equivalent of "I don't understand the question". ChatGPT stated that it didn't have the most up-to-date information, yet it gave some of the most prominent engines, gave some examples of where to look for the latest comparisons, and talked about word error rate and other factors to consider for accuracy.

When asked how to implement gRPC on a Raspberry Pi, Aria gave a descriptive overview of what to do, mentioning the installation of necessary libraries, setting up a Google Cloud account, and implementing gRPC. It didn't say how to do these steps but that they were steps that needed to be done. When asked to give code examples in the same conversation, it simply restated what it said in the previous prompt and provided zero code. This strongly demonstrates how Aria has inferior prompt comprehension and response generation.

Aria can do many of the things ChatGPT can do. It can provide code examples and walk through how it works. It can create a diagram out of text and explain

what it represents. It's able to answer questions on a wide range of topics. It also suffers many of the pitfalls that ChatGPT does. When asked for a source on a topic, it will sometimes provide links that no longer work. It's limited when it comes to details and performs better at giving broader information. It can provide blatantly wrong information, for example, when it was asked what it can do that ChatGPT cannot do, everything it said was possible using ChatGPT. Although we didn't use Aria nearly as much as ChatGPT, our brief testing showed that ChatGPT is superior in most cases.

## 6.0 Hardware Design

In this section, we will be demonstrating how our project will be coming together with each section's designs. We will further cover the reasonings behind the components chosen for each section and how it will complement our overall system to create a functional pair of glasses. We will include any software simulations used and calculations done throughout the design process.

### 6.1 Detection Lens System

For our initial optical design, the schematic included either 2 or 3 lenses, with a combination of bi-concave/convex lenses, plano-concave/convex lenses, or even implemented achromatic or aspheric lenses. Early renditions and simulations have demonstrated that concave lenses allow the beam to expand with greater angular size. However, this resulted in distortion and chromatic aberration which is a consequence of widening the FOV of the lens system.

#### Aberrations and Distortion Compensation:

As mentioned before, chromatic aberration occurs when visible light passes through a single lens and the refractive index shifts each wavelength of color within the visible light to a different path from the lens itself. This means that the wavelength of blue will defocus significantly more compared to the wavelength of red, leading to a much worse blurred image trying to be captured by the visual sensor. There are two main types of chromatic aberration, axial chromatic aberration and lateral chromatic aberration.

Axial chromatic aberration occurs when the pathing of wavelengths shifts from the same focal plane and can not focus, which leaves the image to capture layers of colors with the longer wavelengths lighting the outline of the captured image at these set focuses. This commonly occurs in every lens system but is more noticeable within simple lens systems. One way to combat this is by using lenses called doublet, which are two lenses that each contain strong low dispersion and weak high dispersion properties right next to each other. Another way is by using an image blocker like an aperture and decreasing its diameter

which increases the f-number and increases the depth of field which allows different wavelengths to focus at different distances [123].

Lateral chromatic aberration occurs at the end of lenses where each frequency of color is shaped differently across the image, in other words, chromatic difference of magnification occurs. This can cause a curve to appear for the image, with each wavelength distorting the greater the angle from the principal plane. Unlike axial chromatic aberrations being fixed by using aperture or doublet lenses, lateral chromatic aberrations do not occur at the center of images and can be fixed by using digital sensors to line the varying layers of colors appropriately [123].

Some commercially available lens systems have these kinds of lens achromatic designs that help mitigate the lateral and axial chromatic aberrations; however, every lens suffers from lens distortion. Distortion is a type of aberration that affects how the magnification of the images is captured by the camera, not inherently reducing the image quality but more of a mix of multiple aberrations into a single view. It occurs significantly with lens designs that have a wide field of view and short focal length due to a greater cubic field of dependence [124]. For images that require a wide range of wavelengths, the distortion will vary across the visible spectrum, and environments that have high distortion can greatly impact the image resolution of the camera. The best way to combat the issue is by creating a more complex optical lens design, which shifts the narrative of this project's initial lens system design.

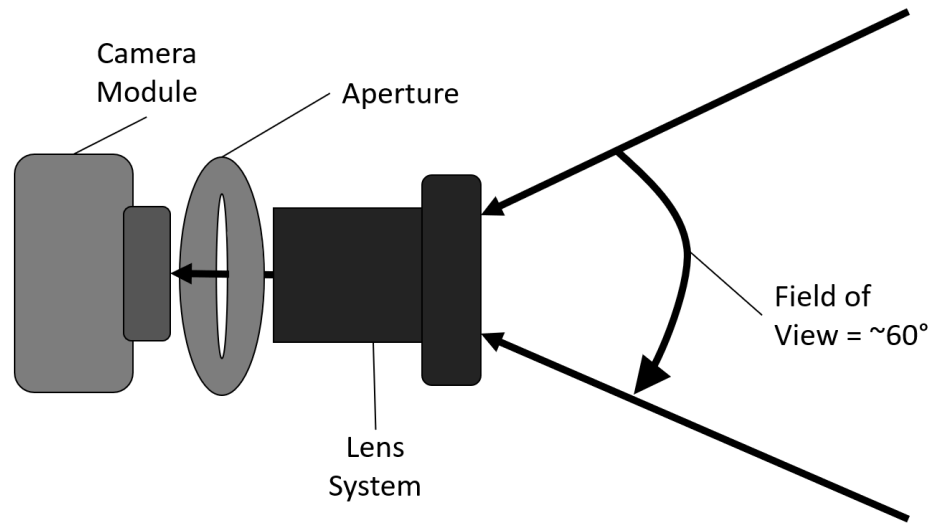
Distortion must be considered for our lens system design because of its varying percentage errors and lens alignment issues. Geometrical distortion involves either negative distortion, where the FOV appears too close to the center, and positive distortion where the FOV appears too far away. Imaging of these distortions is better depicted within the reference as well as equations used to calculate actual distortion and predicted distortion for geometric distortion and TV distortion [124]. Keystone distortion is also important to consider when designing the lens since it is the distortion created based on lens alignment. So while geometrical distortion forms from lenses and can be solved by creating more complex lens designs, keystone distortion forms from combined lens alignments, regardless of the lens system's complexity.

### Lens Mounts:

Depending on the size of the camera sensor, there are certain types of industry-standard lens mounts made for each kind of sensor. The more common commercially available lens mounts are S-Mount (M12), C-Mounts, F-Mounts, and TFL & TFL-II Mounts. The main lens mount for this lens system will utilize the M12 lens mount, which is a type of lens system commonly used for CMOS sensors and lenses that have at most a 12-millimeter diameter (M12).

### Lens Design:

As mentioned before, the initial design used 2 bi-convex lenses to allow beam expansion to occur and the simulation to achieve a wide enough FOV for our design. However, heavy distortion and aberrations occurred because of how simplistic the design was, so the best way to fix this optical issue was to make the system more complex, which involves a combination of singlet and doublet achromatic lenses.



*Figure 14. Schematic of Initial Optical Design for CMOS Camera*

Figure 14 displays the main 3 optical components for the design: The camera module, aperture, and the lens system. As stated before, the camera module is the OV5640, a color CMOS QSXGA 5MP image sensor that will be connected to the ESP32 Sensor board. The aperture which will go between the sensor and the lens system, will be 3D printed and allow the camera to have an F/# of 1/1.4, which will help reduce the color aberration the camera would receive. The lens system that will be attached is an M25360H06 Auducam Lens, a commercially available wide-angle 60° M12 lens with a 1/2.5" optical format.

The reason for each part is as follows, the OV5640 image sensor is not only good at capturing images of color with adequate resolution but it is also compatible with the previously selected Xiao ESP32 Sense which comes preattached with an OV2640. The OV2640 can be switched out with a better quality camera with a higher pixel count, autofocus capabilities, ¼ inch optical size, and an IC2 interface which allows the OV5640 image sensor to process video imagery and customizable software. The aperture allows the lens to produce minimal distortion and color aberration to occur on the camera sensor. While the current F/# is set to 1/2 for the lens system, this can be changed towards the later stages of the project development to accommodate any



modifications needed. Finally, the M25360H06 lens system from Auducam was chosen for many reasons. The manufacturing for this optical system is compatible with any 5 MP count sensor (exactly the kind of camera we have). The M25360H06 is not only commercially available but also a common lens system structure that is compatible with the M12 Lens Mount, which can be 3D printed for the base of the mount to accommodate the square area of the module camera. The mechanical and reference drawing for this lens system is referenced in [125]. On top of that, the M25360H06 not only has a 1/2.5" optical format and F/# equal to 2, but it also has an effective focal length of 3.6 mm, back focal length of 5 mm and because our image sensor has a ¼ inch optical format the lens allows the camera to have an effective focal length of 38.9 mm and horizontal FOV to be about 67°, the kinds of numbers we are looking for to enhance our camera's imaging capabilities.

To further explore this lens system's capabilities, we ordered a lens kit that came packed with 10 lenses that had varying FOVs from 10° to 200° that are each compact and have an infrared cutoff filter to any visible light sensitive sensors. While the kit comes with specifications of the focal length, FOV, dimensions, and optical formatting for each of the lens systems, there are no details on what lenses exactly match within the lens systems themselves and their details, which makes constructing the simulation for the optical design on Zemax hard to accomplish. The best course of action then is to reverse engineer into some of the lens systems and figure out exactly the type of lenses that are aligned within some of these lens systems, from figuring out each lens's focal length to their dimensions, radius of curvature, and to stretch it each of their materials too. Each lens system comes with adhesives with special glue, so the safest approach to open up the lens system is to submerge the frontal lens cap into isopropyl alcohol, which has been proven before to dissolve the glue on our lens system while minimizing the damage on the conventional lenses if they were ever coated (which in a commercially available lens kit is very unlikely). After leaving the lens system within the solution for about 30 minutes, we were able to uncap the top piece holding the lenses aligned and allowing each spacer and lens to fall out one by one. The following image shows the different optical components of each lens system from the 10° (top setup) and the 40° (bottom setup) FOV.

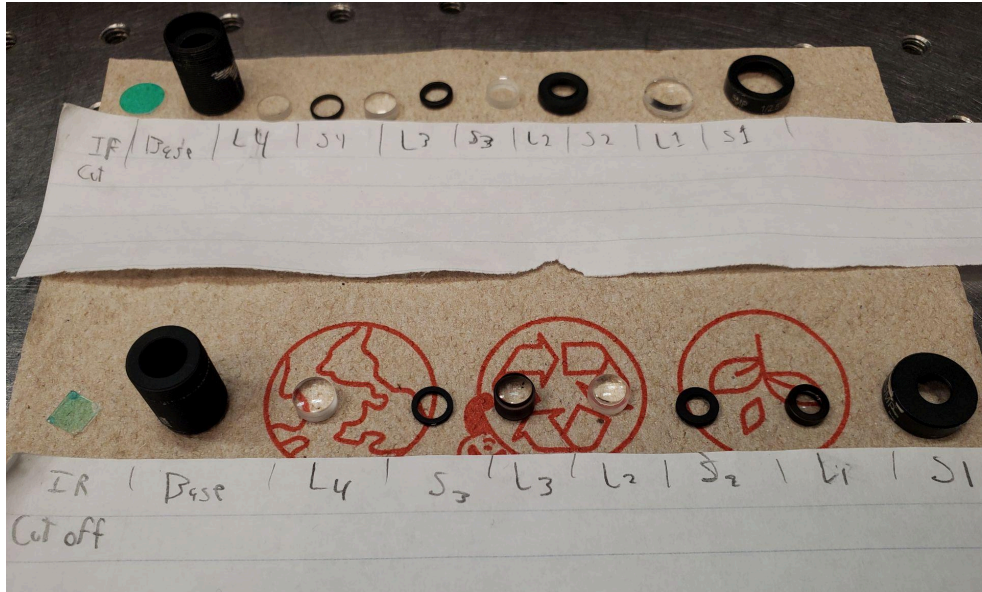


Figure 15. Conventional Lenses and Spacers Within the  $10^\circ$  FOV and the  $40^\circ$  FOV (Reading from top to bottom)

The interesting thing to note after dismantling the  $10^\circ$ ,  $40^\circ$ , and  $100^\circ$  (which isn't displayed) a  $10^\circ$  conventional lens setup is different from the lens setup of  $40^\circ$  to  $100^\circ$ , which can be shown by the two images Figure 16 and Figure 17. It is also important to note that these schematics are just to display the different lens types and not the spacing between each lens since in actuality there is no spacing between each conventional lens.

Also, for clarification, these lenses make a certain thickness length which is not equated with the theoretical equations and is partially accounted for in the Zemax simulations. This is just to display the types of conventional lenses that line the lens system and how, like within a puzzle set, line up with each other. Also, these lenses were not intended to be used, so very little care was taken to keep these lenses in their original high-quality state. For future testing with the  $60^\circ$  FOV lens system, proper care will be brought to that lens system when testing.

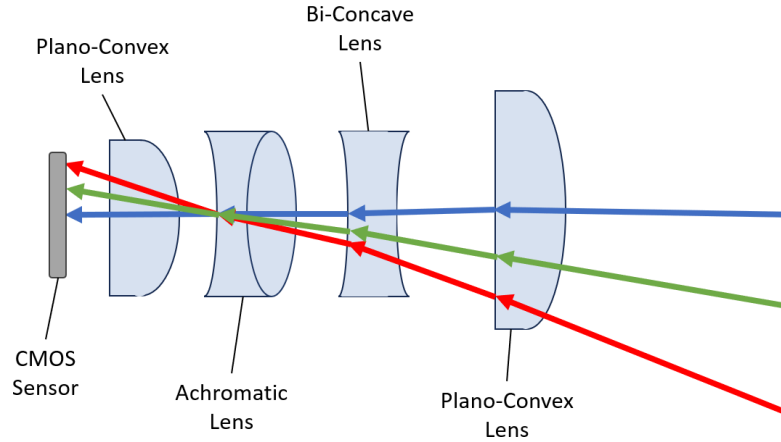


Figure 16. Schematic of 10° FOV Optical Design for CMOS Camera

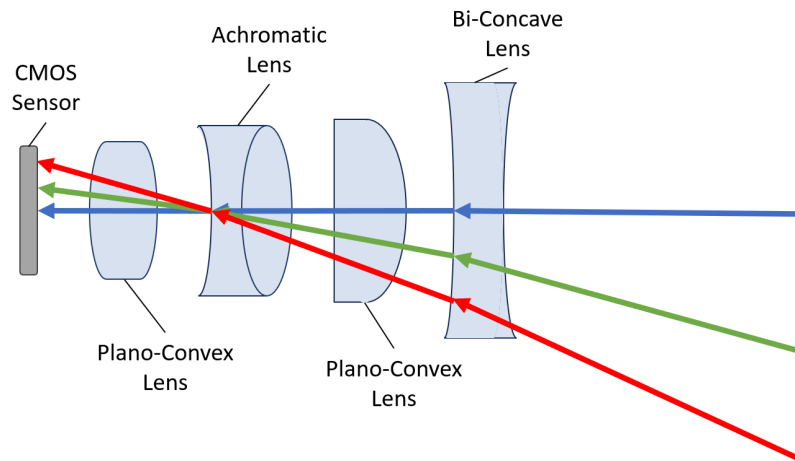


Figure 17. Schematic of 60° FOV Optical Design for CMOS Camera

Because dismantling the lens system would ruin the lens alignment, we tested on other lenses within the kit to get a rough picture of how the 60° looks. Focusing now on the 40° FOV lens system, the specifications for each of the lenses are as follows (which are not 100% accurate but this information was used to help create the Zemax simulation and we assume the material used was Uncoated Silica with a refractive index of 1.5):

Table 12. The Recorded List of Optical Components from 40° FOV Lens System

Optical Component	Dimensions	Thickness	(For Lenses) Focal Length	(For Lenses) Lens Type	Related Part #
IR Cut Off*	6 mm x 6	1 mm	NA	NA	NA

	mm				
Base	~12 mm Diameter	16.5 mm	NA	NA	NA
L4	~7 mm Diameter	~4 mm	8 mm	Bi-Convex	Thorlabs: LA4280
S3	7 mm Diameter	1 mm	NA	NA	NA
L3	OD** = 7 mm ID*** = 6 mm	~5 mm	NA	Doublet Convex-Concave	NA
L2	OD = 7 mm ID = 6 mm	3 mm	~5 mm	Aspheric Plastic	Thorlabs: CAY046
S2	6 mm Diameter	1 mm	NA	NA	NA
L1	7 mm Diameter	~2 mm	6 mm	Bi-Concave	Thorlabs: LD4797
S1	OD = 14 mm ID = 5 mm	4 mm	NA	NA	NA

Most of the specifications for each lens were compared to commercially available lenses from well-established optics manufacturing companies, with the most compatible lens from Thorlabs. The focal length and dimensions were both measured simply by using metric ruler measurements and the basic eyeing the focal length test. These trials allowed us to handpick the most compatible lenses from Thorlabs and implement these lens specifications to the Zemax simulation later discussed.

From this collected data, we were able to apply a Zemax simulation of the proposed lens system in Figure 18 with the following cross-section when introduced to visible light (400nm to 700nm) and have already about 60° of light going into and stopping at the first surface:

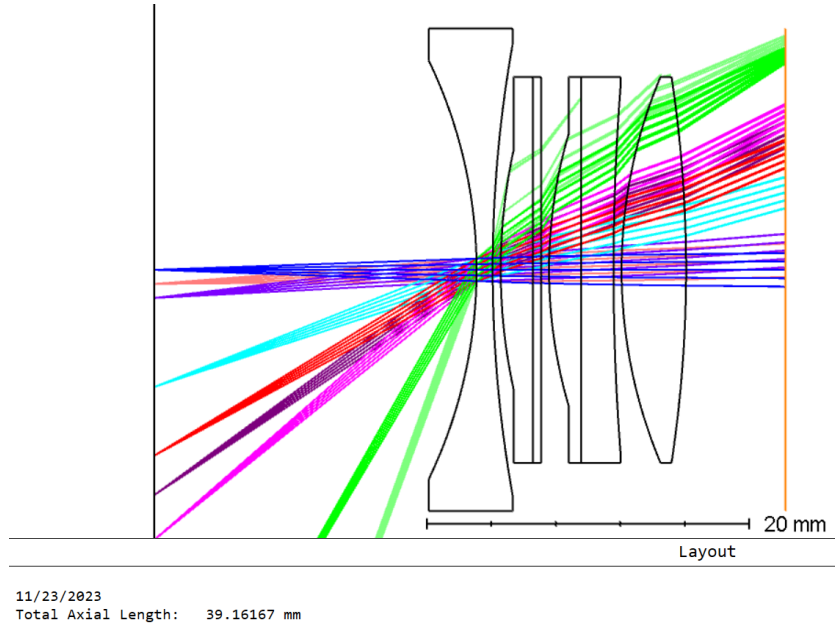


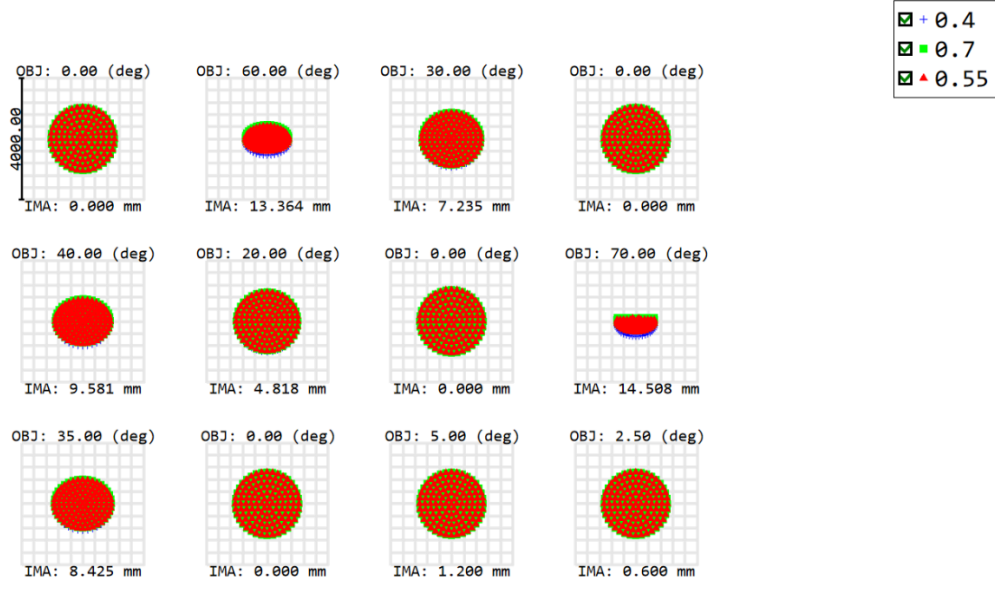
Figure 18. Cross Section of 60° FOV Lens System on Zemax

The setup for this cross-section is also displayed in the following lens data table:

	Surface Type	Comment	Radius	Thickness	Material	Coating	Clear Semi-Dia	Chip Zone	Mech Semi-Dia	Conic	TCE x 1E-6
0	OBJECT	Standard	Infinity	20.000			54.950	0.000	54.950	0.0...	0.000
1	STOP (aper)	Standard	-30.000	1.000	SILICA		13.000 U	0.000	15.000 U	0.0...	-
2	(aper)	Standard	80.000	0.500			14.000 U	0.000	15.000 U	0.0...	0.000
3	(aper)	Standard	35.000	2.000	SILICA		7.500 U	0.000	12.000 U	0.0...	-
4	(aper)	Standard	Infinity	0.500	QUA...		7.500 U	0.000	12.000 U	0.0...	-
5	(aper)	Standard	Infinity	0.500			8.000 U	0.000	8.000 U	0.0...	0.000
6	(aper)	Standard	30.000	2.000	SILICA		8.500 U	0.000	12.000 U	0.0...	-
7		Standard	Infinity	2.000	SILICA		10.000 U	0.000	12.000 U	0.0...	-
8	(aper)	Standard	150.000	0.500			11.500 U	0.000	12.000 U	0.0...	0.000
9	(aper)	Standard	31.000	4.000	SILICA		12.000 U	0.000	12.000 U	0.0...	-
10	(aper)	Standard	-80.000	2.000			12.000 U	0.000	12.000 U	0.0...	0.000
11	IMAGE	Standard	Infinity	-			15.000 U	0.000	15.000 U	0.0...	0.000

Figure 19. Lens Data for 60° FOV Lens System on Zemax

And the recorded spot diagram for the varying degrees of rays hitting the image surface with each wavelength (400nm - 700nm):



Surface: IMA

Spot Diagram												Zemax Zemax OpticStudio 22.2.1	
11/23/2023 Units are $\mu\text{m}$ . Legend items refer to Wavelengths Field : 1 2 3 4 5 6 7 8 9 10 11 12 RMS radius : 791.002 450.085 698.911 791.002 622.925 751.477 791.002 400.211 663.656 791.002 788.611 790.405 GEO radius : 1043.33 724.841 971.142 1043.33 906.847 1013.16 1043.33 640.061 941.608 1043.33 1041.45 1042.88 Scale bar : 4000 Reference : Chief Ray												60_LS.ZOS Configuration 1 of 1	

Figure 20. Spot Diagram of 60° FOV on Zemax

As it shows, the table produces a complex lens system with about a 39 mm focal length which is similar to the required focal length of the commercially available lens system we plan to use for our camera. The table was optimized to make sure the cross-section within Figure 19 can properly display rays with minimal chromatic aberration across all surfaces of each lens as shown in Figure 20. As shown in Figure 20, the spacing of each wavelength is kept to a minimum going from the object's perspective and input degrees from 0° to 60°. Unfortunately, at 70° the rays are outside the scope of view for the image which clips the top rays too short. Something important to notice too is that going from 0° to 60°, the image surface's circle decreases as the input angle of the object increases, which can be described as the distortion of the object increases the greater the FOV of the lens system becomes as mentioned before. Fortunately, while distortion may occur, it doesn't affect the overall visible color detection for the image surface, which is what the optical design is aiming for.

The 60° FOV Lens System on Zemax was designed with the intent to mimic the physical lens system design and was mainly shown to display the reasoning for using this specific lens system for our project's camera. Unfortunately, the type of material, radius of curvature, and even the exact focal length are not the same as

the physical lens system, but it is the closest it can be since the manufacturer will not disclose patent material.

### Mathematics:

While the exact numbers for the Zemax simulation have some error percentage, the exact specifications for the lens system, camera sensor, and compatibility can be proven with some basic mathematical equations that are covered by geometrical optics and visual optics. These equations range from finding the actual Field of View (FOV) for the lens system to the potential image spot size not just mentioned within the lens system specifications but of how the image would display on the CMOS sensor.

To start, the FOV can be determined by first finding the horizontal FOV as follows:

$$FOV = 2 \tan^{-1} \left( \frac{h}{2f} \right) \quad (12)$$

Where “h” is the width of the sensor and “f” is the effective focal length of the lens system, which at the time was  $h = 3673.6 \mu\text{m}$  and  $f = 3.6 \text{ mm}$ . This makes FOV about  $54^\circ$  which is roughly the  $60^\circ$  FOV we’re looking for in our system. The camera module also comes with its own FOV but it is not wide enough to accommodate the parameter we are looking for, which is why this kind of lens will not only widen the FOV but enlarge the images for the camera to see a wider area of object for our project.

Magnification plays a huge part in figuring out exactly how wide and tall the CMOS sensor should be initially when compared to scanning its surrounding environment. To find the rough magnification of our system we revisit equation (4) which is reworded to be:

$$M = \frac{h}{FOV} \quad (13)$$

With “h” being the width of the sensor area and FOV the horizontal FOV of what the sensor is trying to capture, which in this case is “h” =  $3673.6 \mu\text{m}$  and  $FOV = 0.541$ , equation the magnification M to be about  $\times 0.007$ . It makes sense in this case since we are trying to capture a wide object in such a small image area, so reduced magnification must occur for this to happen.

With every lens system, there comes the description of the F-Number, which currently is  $\frac{1}{2}$  for the lens system. While fine, we need to reduce this number to allow better capture images for the camera sense, which can be calculated as follows:

$$F/\# = \frac{f}{D} \quad (14)$$

Where “f” is the focal length of the lens system and “D” is the diameter of the aperture allowing light through. In our current setup the focal length “f” is 3.6 mm and the aperture diameter “D” is roughly 1.2 mm, making  $F/\# = 3$ . This number can be changed depending on the diameter of the aperture and will help us in a later equation.

As mentioned before, to reduce the chromatic aberration the lens system must come with equipment with a low enough Numerical Aperture value which can be seen as

$$N. A. = \frac{1}{2 * F/\#} \quad (15)$$

Where  $F/\#$  is the F Number of the lens system ( which was previously calculated). Currently, the desired N. A. the design is aiming for is  $\frac{1}{6}$ , which is roughly  $N.A. = 0.17$ . This tells us through the resolution power the camera will be experiencing as well as increasing the clarity of the images captured as well.

The working distance of our lens system would work in our favor as we want to capture objects no farther than 1 meter. so, to calculate the average working distance that would benefit our system we must use the following equation:

$$W.D. = \frac{f}{M} \quad (16)$$

Where “f” is the effective focal length of our design and “M” is the magnification of our lens system. In this case from our previously calculated values if “f” = 3.6 mm and “M” =  $\times 0.007$ , then our working distance “W.D.” = 0.514 m, which is roughly the ballpark of where we want the lens system to focus.

Following through with the previously stated FOV, the horizontal FOV in our lens system can be equated as follows:

$$\text{Horizontal FOV (mm)} = \text{Working Distance (mm)} * 2 \tan\left(\frac{\text{FOV (degrees)}}{2}\right) \quad (17)$$

Here, we have already collected the Working Distance which is 514 mm and the original FOV of the lens system in degrees which is  $54^\circ$ , which equates our horizontal FOV to be about 524.1 mm. Taking that value, we can calculate the sensor resolution (S.R.) of our CMOS sensor which is capable of capturing good images with the following equation:

$$S.R. = 2\left(\frac{\text{FOV (mm)}}{\text{Smallest Feature}}\right) \quad (18)$$

With the FOV being the horizontal FOV of the lens system in mm which is 524.1 mm and the smallest feature being the smallest width of our camera that can operate, which in this case for our CMOS camera is about 0.896 mm. Equating this formula we come up with about 1170 pixels needed minimum for our camera



to operate at a good quality resolution, which is perfect since the camera can be formatted to have a 1280x960 resolution.

The image circle diameter, another important equation to not look over, is what allows the user to see what exact kind of lens system to look for when trying to compare the sensor's dimensions to what can be captured. The image circle diameter can be equated as follows:

$$\text{I.C.D.} = \sqrt{\text{width}^2 + \text{height}^2} \quad (19)$$

Where the width is the measured width of the sensor and the height is the measured height of our sensor. Taking the already established specifications of our OV5640 CMOS sensor, we get our Image Circle for good quality imagery cooperation to be about 4.58 mm. This fits well within our sensor's areas as we want the diameter of the image circle to perfectly encapture our sensor, but not either stretch too far or come up short of our sensor's dimensions. Current simulations from Zemax show the circle comes up to about 3.52 mm, which is 1 millimeter short however that can be adjusted later within the physical project.

These are the general equations that were used to give reasons as to why we chose this exact lens system with the camera we originally picked for our project. Subject to change the following semester as we grow our project, but for our initial design it is indeed a step in the right direction for how we want to approach our optical design for our project.

#### A Lens System Mutation:

Due to the combined nature of the Ov5640 CMOS sensor and its extra features of having an already established 60° FOV format, a new lens system must be constructed to adhere to the optical design requirements. This that case the next best approach is to establish a new lens system that combines the lenses of other M12s and create a new about 80° FOV lens system. The next few sections will go into detail on the approach of creating the new lens system, the specifications for each part created and used, and how overall this enhances the camera's FOV.

To start, we strip the lenses contained within the 80° and 100° FOV M12 lens systems. By taking the spacers and lenses with the lens systems, we were able to establish a lens design that visually widens the FOV of the camera and provides little chromatic distortion. A Zemax simulation is provided that shows how much the lens system alters the viewing experience with it's specifications (Figure 21).

	Surface Type	Comment	Radius	Thickness	Material	Coating	Clear Semi-Dia	Chip Zone	Mech Semi-Dia	Conic	TCE x 1E-6
0	OBJECT	Standard	Infinity	0.000			9.682	0.000	9.682	0.000	0.000
1		Standard	Infinity	1.000			9.682	0.000	9.682	0.000	0.000
2	(aper)	Standard	10.000	1.000	SILICA		3.000 U	0.000	3.000 U	0.000	-
3	STOP (aper)	Standard	Infinity	2.200	SILICA		3.000 U	0.000	3.000 U	0.000	-
4	(aper)	Standard	-5.000	0.500			4.000 U	0.000	4.000	0.000	0.000
5	(aper)	Standard	-5.000	5.000	SILICA		4.000 U	0.000	7.000	0.000	-
6	(aper)	Standard	-7.000	10.000			7.000 U	0.000	7.000	0.000	0.000
7		Standard	Infinity	-12.345			100.000 U	0.000	100.000	0.000	0.000
8	IMAGE	Standard	Infinity	-			8.114	0.000	8.114	0.000	0.000

Figure 21. Lens Data for 80° FOV Lens System on Zemax

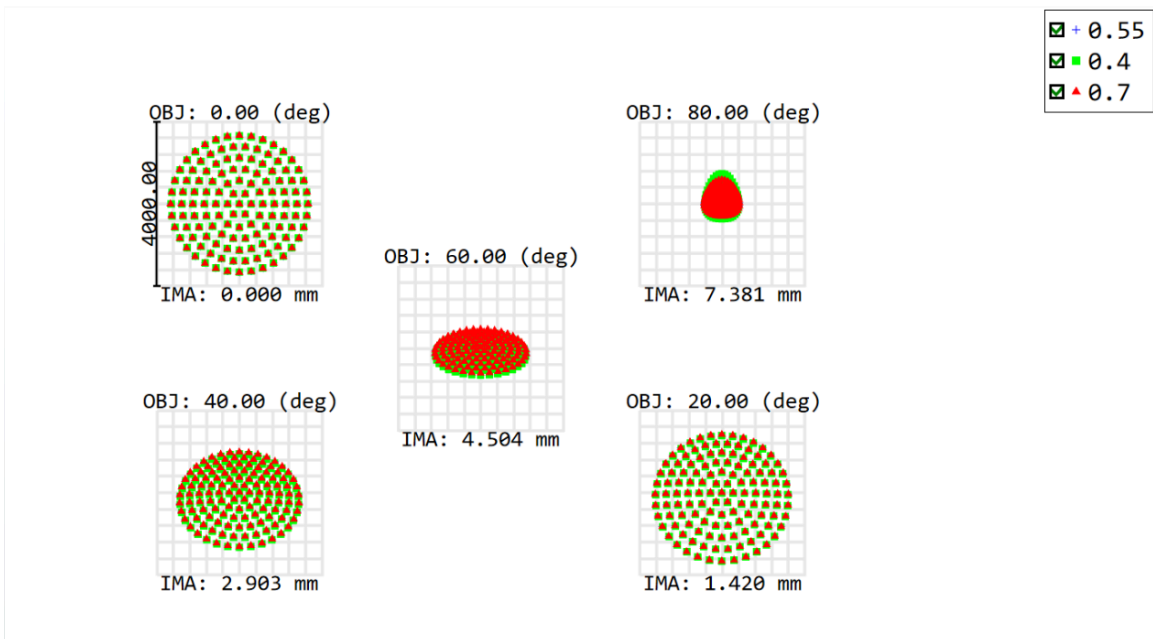
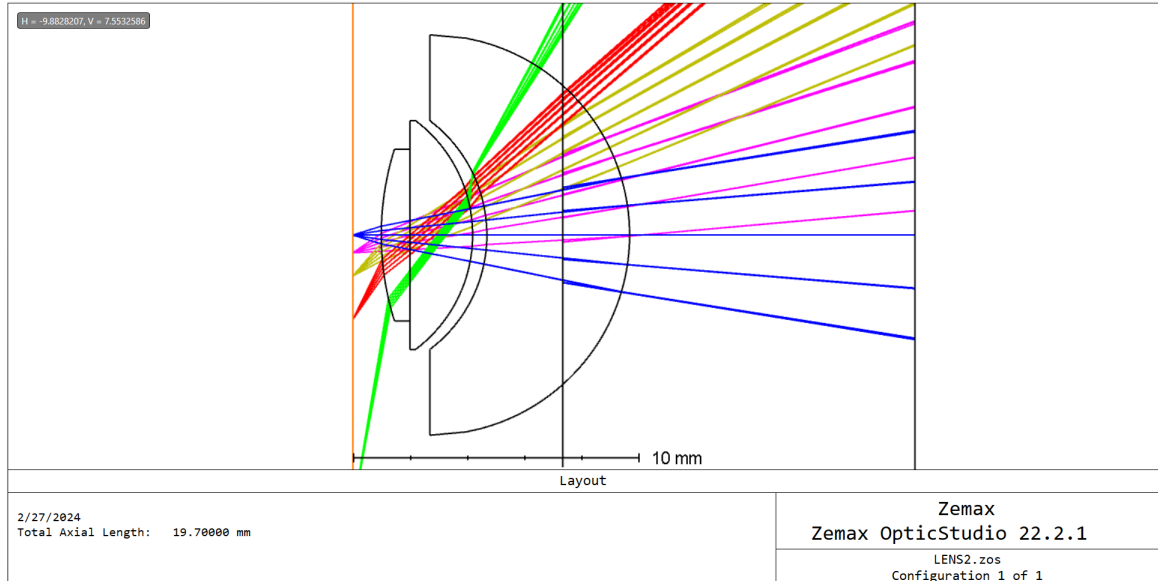


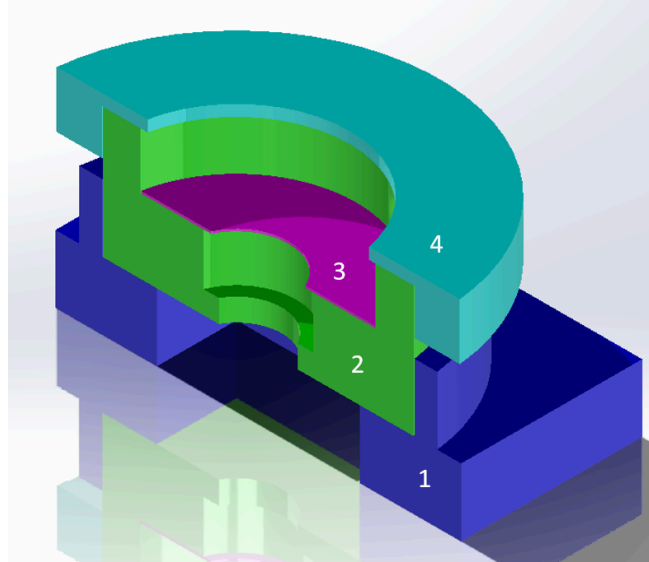
Figure 22. Spot Diagram of 80° FOV on Zemax



*Figure 23. Cross Section of 80° FOV Lens System on Zemax*

While the cross section of the lens system isn't exact to the official design, it still provides almost the same information with the total effective focal length of the system being about 10.5 mm and the length of the lens system a rough estimate. Unfortunately, as mentioned before, the providers of the M12 lenses refused to share details of the lenses within the lens system, making testing and measuring the lenses having some human error. Regardless, this new established system provides a clear but wider FOV for our camera to utilize, which does not compromise our camera's viewing quality. As shown in Figure 22, the lens system provides a spot diagram that still keeps the colors from being very distorted and chromatic aberration while present is kept minimal being towards the end of the spot diagram when viewing the 80° FOV.

After testing which lenses worked with the camera and creating a Zemax simulation for them, it was time to create the casing of the overall lens system. There are 4 parts that are shown all combined with the following images, consisting of the lens mount (1), the lens system (2), the spacer (3), and the lens cap (4). Each of these parts have their specifications, but a percentage of an error still occurs due to the 3D printing's filaments.

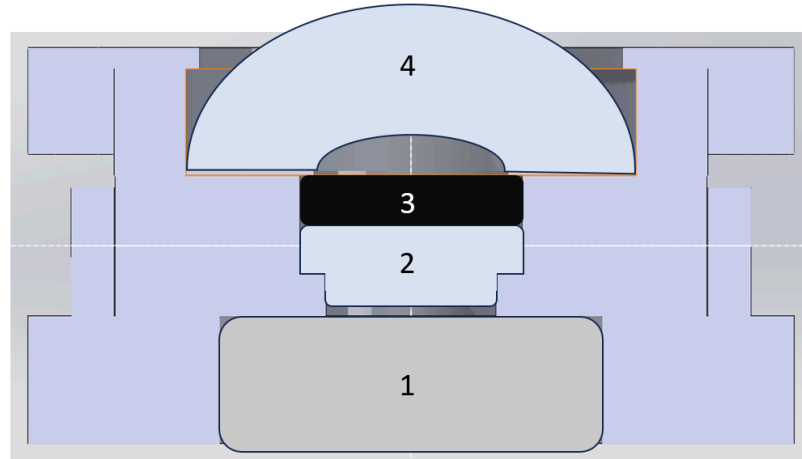


*Figure 24. Cross-section of Mutation Lens System on SolidWorks Color Coded and Numbered: Lens Mount (1), Lens System (2), Spacer (3), and Lens Cap (4)*

Each part serves their purpose, starting with the lens mount which is labeled at #1. The lens mount sits right on the Ov5640 Camera which has a dimensions of about 8.5 mm by 8.5 mm and about 5.4 mm tall. The Camera can easily fit right in with no wiggle room so the images can be captured with little to no distortion. The base of the lens mount is about 18 mm by 18 mm and has an inner diameter of about 14 mm to fit the lens system and an outer diameter of about 16 mm. The lens system, labeled at #2, is the main housing of the lenses used for the optical design. It houses 2 lenses and a metallic spacer to keep the lenses from touching. As it shows in the diagram, the smallest diameter hole is about 4 mm with a height of about 1 mm. This acts as the pinhole for the camera to the lenses within the lens system. For the next size up, the diameter is about 5.2 mm and the depth is about 2.3 mm. It's set with these specifications so it can house both the first lens and the metallic spacer respectively in that order from bottom to top. The following space has a diameter of about 10.6 mm and a depth of about 2.5 mm. This large space houses the larger 2nd lens for the lens design as well as the spacer (which will be talked about in a bit). The overall diameter of the lens system is about 14 mm with a height of about 5.8 mm. This lens system fits perfectly within the lens mount. The spacer, labeled as #3, has an inner diameter of about 4.6 mm and outer diameter of 10.6 mm, with a thickness of about .1 mm. The purpose of this spacer is to keep the metallic spacer and the smaller lens within the lens system as well as keep the larger lens separated, allowing all the pieces to stay put and fit well together. Finally the lens cover, labeled as #4, has an outer diameter of about 18mm, with the next inner diameter matching the lens system outer diameter of about 16mm, and the smaller diameter to be about 10 mm. The thickness of the lens cap is about 2.5 mm with the inner thickness being about 1 mm. The specifications are set so that the lens not only stays in

place but so that the lens can capture as much information as designed without being accidentally covered by the lens cover.

As there are 4 main parts of the 3D printed lens system, there are 4 main components that play a part within this system which include the



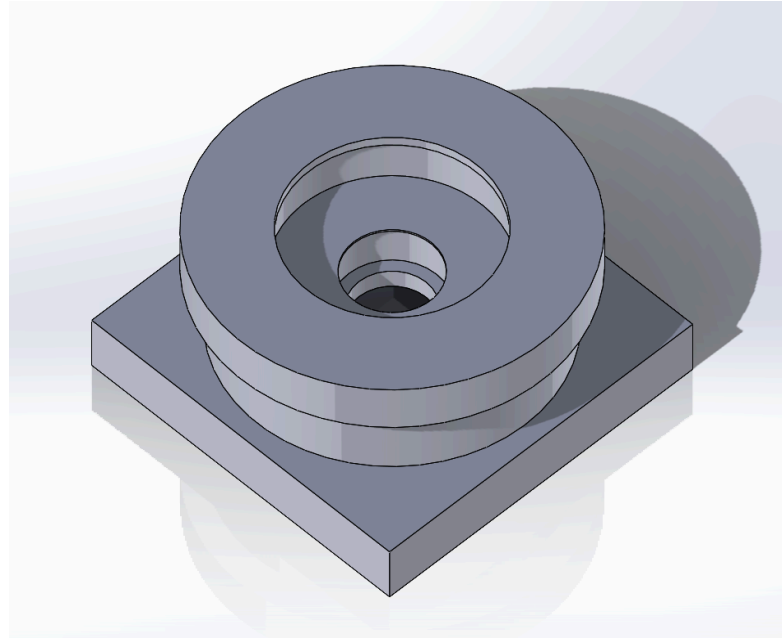
*Figure 25. Cross-section of the Mutation Lens System on SolidWorks with the Labeled Parts within the System. Camera (1), Smaller Lens (2), Metallic Spacer (3), and Larger Lens (4).*

Shown below is a more formal table laying out each of the 3D printed parts and components within the system and details as well as an image showing the overall mutated lens system.

Table 13. Specifications of 3D Camera Lens System Components

Part	Specification	Purpose
Lens Mount	Base - 18 mm x 18 mm Inner diameter - 14 mm Outer diameter - 16 mm	Holds camera and lens system together
Lens System	Pinhole diameter - 4 mm with a thickness of about 1 mm. Small housing diameter - 5.2 mm and the depth is about 2.3 mm. Larger housing diameter - 10.6 mm and a depth of about 2.5 mm. Outer diameter - 14 mm	Houses the optical design of the camera

	with a height of about 5.8 mm	
Plastic Spacer	Inner diameter - 4.6 mm Outer diameter of 10.6 mm, with a thickness of about .1 mm	Separates and holds the lenses in place
Lens Cap	Outer diameter - 18mm, Larger Inner diameter - 16mm, Smaller Inner diameter - 10 mm. Length of Lens Cap - 2.5 mm with the Inner thickness being about 1 mm	Caps the lenses within the lens system and keeps them from falling out
Camera	8.5 mm x 8.5 mm x 5.4 mm	The main source for visual input of our project
Smaller Lens	Diameter - 3 mm and 4 mm, Thickness - 2 mm, Focal Length - 10 mm	The entry lens for the optical design
Metallic Spacer	Diameter - 5 mm, Thickness - 1 mm	Keeps the smaller lens in place and separate from the larger lens, little distortion
Larger Lens	Diameter - 10 mm, Thickness - 2 mm, Focal Length - -17 mm	The main lens providing the FOV of the lens system



*Figure 26. The Overall Mutation Lens System on SolidWorks*

As mentioned before, the point of the titled “Mutation Lens System” is to widen the camera’s FOV from its established  $60^\circ$  to about  $80^\circ$ , allowing the camera to track the user’s surroundings with a wider scope. There is slight distortion when adding the extra lenses on top of another lens system, however this does not reduce the camera’s image quality, and thus does not hinder the visuals.

## **6.2 Text Projection Lens System**

In this section, we will break down the components that will be used for the LCD projection system. We will include the designs, and calculations, as well as the challenges for designing the system and how it will be incorporated into the overall project.

Before designing the text projection lens system, there are a few factors that must be considered when choosing all the elements needed for a clear and quality image. This includes light source, magnification, lens diameter, focal length, projection distance, etc. These factors will greatly affect the quality and clarity of the image that will be projected for the user. We also must remember size constraints as the lens system will be attached to the side of the frame of the glasses. This became a bit challenging to include in the design as the focal length of the lenses is a huge factor and the need to magnify the projection from such a small display.

The LCD will be our light source for this system and is a critical part as it will be used to display the live captioning for the user, as well as the warning once the color is detected. As discussed previously, the brightness of the LCD is important

in this project to be able to fully display the text and symbols, while taking into account the amount of loss that will occur throughout the system as well as the combiner being used. It is also important to note the number of pixels in the display must be sufficient to continuously follow a conversation without the text lagging. The size of the display can be seen as a positive as well as a hindrance. The size is a fantastic way for us to keep all the other components small as it will help us achieve a smaller form factor of the design. The size of the display can also become an issue since we will need to be able to magnify the texts and symbols as needed to project them onto a larger screen in front of the user. We will try to optimize our system.

We also must consider other factors, such as the lens thickness, desired image size, weight, and cost. The previous micro-OLED that was chosen has an active area size of 6.48 mm x 4.86 mm, with a diagonal size of around 8 mm. This is important to make sure that the lens diameters will encompass all the images and text from the micro-OLED, as well as the desired size of the image we want to project to the user. We want it to be easy to read and without the user having to strain his or her eyes to read the text so close in front of their eyes.

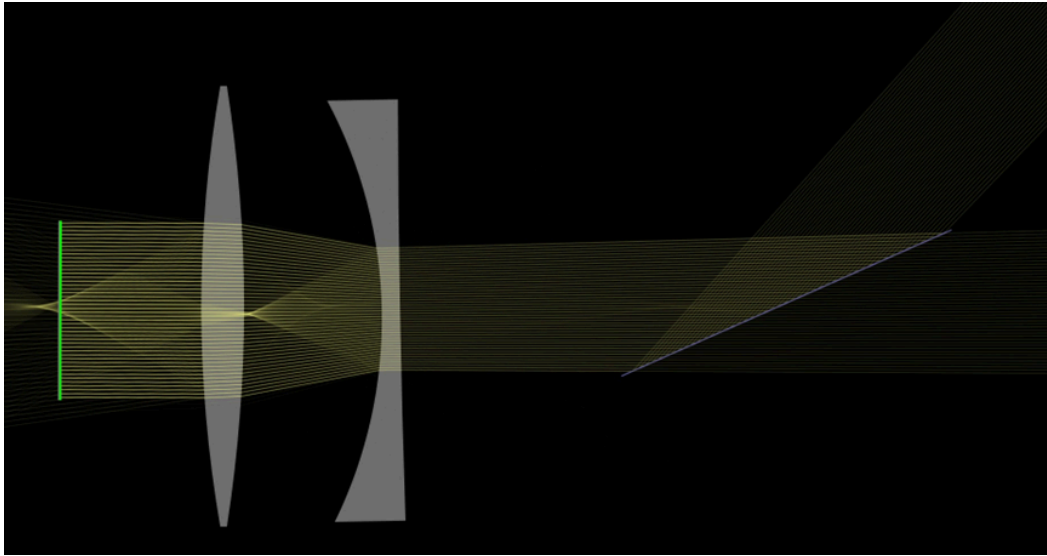
With these in mind, we produced three separate designs that would be possible. The simplest design includes the micro-OLED and cube-form beam splitter. This was the most straightforward design, but it does mean a large form factor of the large cube glass in front of the glasses, obstructing the user's eyes in some way. This also reduces the eye box of the user as it will take up a large amount using a cube beam splitter.

There are many lens combinations and countless relay optics that can be used for this projection system. There are a few types of lenses that were carefully considered for the text projection lens system, such as bi-convex, plano-convex, achromatic doublets, and aspheric achromats. Achromatic lenses can be used in similar systems for imaging as they diminish aberrations, since this type of lens uses a positive and negative index, Lens systems with this type of lens significantly reduce spherical aberrations and chromatic aberrations are negligible. Aspheric lenses can also substantially lessen spherical as well as chromatic aberrations. They are also exceedingly small in size, which can be used for design similar to our project, where size is a great factor. The challenge with these two lenses is the cost of a higher-quality image. Utilizing a double convex lens is great for a one-on-one imaging system due to its shape and it also allows lesser aberrations in the system. In any optical system, aberrations are quite inevitable, and the use of a double convex at a low f-number will still have some sort of aberrations caused by the shape factor of the lens.

$$S = \frac{R_2 + R_1}{R_2 - R_1} \quad (20)$$

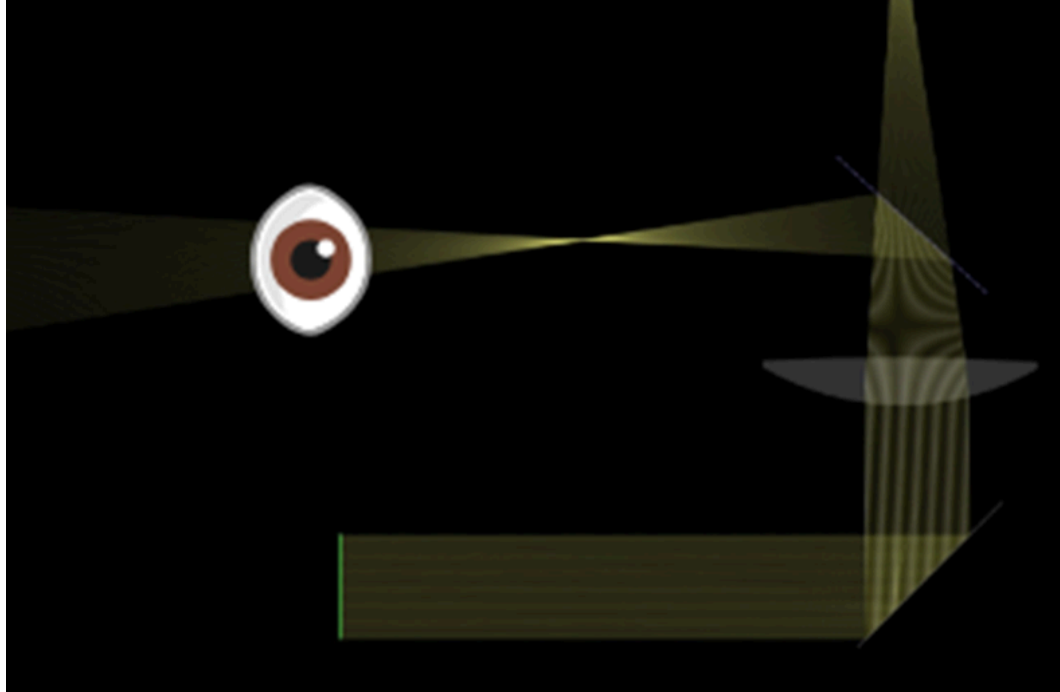


In the shape factor equation,  $R1$  and  $R2$  are both the radii of each surface of the biconvex lens. While the use of relay optics makes it easier to correct aberrations, we can still calculate the best way to decrease aberrations in a system with just one lens. It was found that the glasses of 1.5 indexes have a shape factor of around 0.8 to reduce basic aberrations, such as coma and spherical aberrations.[126] For the following design, we decided to pair the biconvex lens with a plano-concave lens to help reduce aberrations, this is done with the combination of the convex and concave lens, which can be seen in Figure 21. This imaging was done using the 3DOptix Software to show the incoming light rays and how they will pass through the system.



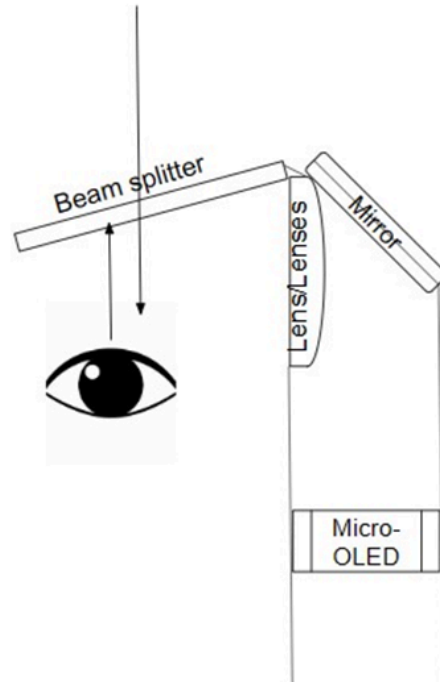
*Figure 27. Projection System Using Biconvex and Plano Concave Lens*

For the next subset, we decided to pursue this design with a plano-convex lens, mirror, and beam splitter. Plano-convex are great lenses to use for this system, as they are commonly used for various applications such as collimating and magnifying objects. Figure 22 demonstrates the design that will be used for the text display. The rays represent the micro-OLED display and travel of the light. The light path first interacts with a mirror, followed by a plano-convex lens, then a beam splitter. The lens and the mirror will be stored in a 3-D printed housing as shown in Figure 23. This will enable us to put away most of the other components attached. It will also provide an optical axis for the travel of the light.



*Figure 28. Text Display Projection Light Path*

In this figure, we are using a simple plano-convex lens for the display. This design is much simpler and the most cost-effective. It allows us to still be able to see the image without the need for a relay of optics, which can quickly add up. The ray tracing helps us visualize the light path, but we will then mention the different equations that help us understand what is actually happening.



*Figure 29. Lens Housing Layout for Components*

The figure above shows the layout for the housing of the lens system and other components. This will also be used to hold other components such as the MCU and the battery pack to power the micro-OLED as well as the camera. This housing will be attached to the side of a pair of glasses to try to maintain the wearability of the device with all the additional components.

Aberrations will be lowered drastically in this system as it will be using a lens with a remarkably high f-number of 7.8. The relation of these aberrations in a system can be seen in the following equations for coma, spherical, and astigmatism. With our calculations, we can see that astigmatism will be roughly 0.12, whereas the remaining two are notably less than 0.01.

$$\text{Coma} \propto \frac{1}{(f/\#)^2} \quad (21)$$

$$\text{Spherical} \propto \frac{1}{(f/\#)^3} \quad (22)$$

$$\text{Astigmatism} \propto \frac{1}{(f/\#)} \quad (23)$$

Comatic aberration is a type of aberration that is caused by the magnification of the system, which causes rays off the axis to create a comet-like blur, this is very similar to spherical aberrations. They are mostly affected if the wavefront is not centered or tilted. Spherical aberrations occur when the light rays on the edges focus at different points and not the center point. This effect can significantly reduce the image quality and occurs when lenses have spherical surfaces. Astigmatism is when the light rays are not focused on the same plane, which creates a distorted image. This type of aberration can be fixed using cylindrical lenses to focus the image in the same plane.

For our system, we also have to know our desired magnification for our imaging from the micro-OLED to the beam splitter. We can determine this magnification using the following equation which uses the focal length or distances between the image and object.[126]

$$M = \frac{F_2}{F_1} \quad (24)$$

$$M = \frac{l}{o} \quad (25)$$

$$M = \frac{h_i}{h_o} = \frac{-d_i}{d_o} \quad (26)$$

For this design, we have decided that it would be necessary to have a 3-4x magnification of the image for the size of our micro-OLED display. We can determine the focal length needed for the lens or lenses that will be used to magnify the image. We can use the equation above which shows us the relation between the distance of the object and image with the height of the object and image to the magnification. The distances are measured from the center of the lenses. We can use the thin lens equation, which will also help us determine how far away the object should be placed from the lens.

$$\frac{1}{d_o} + \frac{1}{d_i} = \frac{1}{f} \quad (27)$$

In order to determine if the image is formed we can follow certain cases depending on the type of lens being used. For instance, if the object is placed closer to the lens than the focal length of the converging lens, the image will be magnified. In this example, the image will be virtual, which means that we will not be able to project it. With converging lenses, the object must be placed farther away from the lens' focal length. This will give us a real image, but it will be inverted, whereas a virtual image cannot be projected but it will be upright. This

is a challenge in itself to ensure that we can have a magnified, real, and upright image.[127]

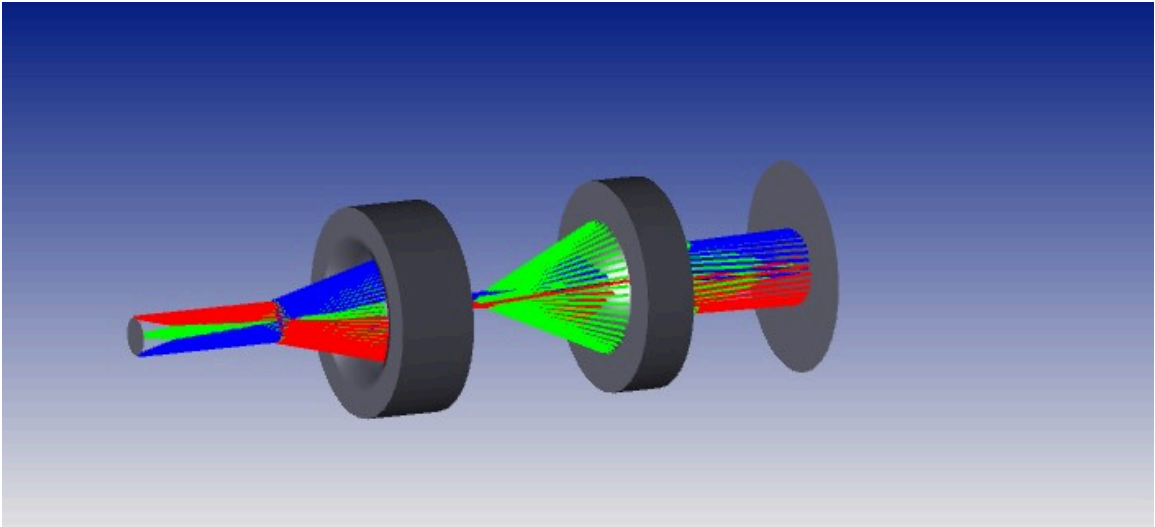
When determining the previous two designs, we based them on the micro-OLED display that was 0.32 inches. We wanted to have a very small and compact design that would allow the user more flexibility with a lightweight design. In designing with such a smaller display, it became apparent that the magnification needed to fully display all the content of the micro-OLED to the screen in front of the user was very large. This means that the number of lenses needed for the lens system will need to increase to achieve this magnitude of magnification in the system to increase the eye box. This also means increased aberrations with the addition of more lenses to acquire the desired projection size, and our system would also be more prone to misalignment. It would affect the sharpness of our image and cause a high level of distortion that can cause the user discomfort. With all these factors in mind, we have decided to change our selection to a larger micro-OLED of 0.70 inches. With this, we created a simple design that can project our image on a plate beam splitter of 10/90 ratio of the reflection and transmission ratio. For the other design with the 0.70-inch micro-display, we have designed a biconvex lens with a focal length of 19.6 millimeters and a plano convex lens with a focal length of 38.1 millimeters. With this design, it is difficult to determine the quality of the image, and quite difficult to quantify these types of challenges. It will also require us to do a lot more testing to find the best approach with the best result.

*Table 14: Lenses for Text Design*

Model	Manufacturer	Diameter	Effective Focal Length (EFL)	Price
KBX043AR.14 Biconvex Lens	Newport	25.4	19	\$55
KPX079AR.14 Plano-convex	Newport	25.5	38.1	\$35

For this new design, we were able to run a Zemax simulation to demonstrate the proposed lens system as shown in the figure below. This shows the exact lenses from the table above with diameters at 50 millimeters. We were not able to calculate some of the other factors, such as the modulation transfer function. This type of curve can help us understand ways optical aberrations can alter our

image quality. Zemax was not able to properly calculate and graph the curve as the sampling data was too low. We were also not able to graph the point spread function for the same reason. It was not possible to gather as much data from this simulation as we would have wanted, but it does show us the propagation of our system. The one challenge that did arise was the total length of this system, which was 172 millimeters. This does pose a challenge for us as it is the length of this system will be slightly larger for a standard pair of glasses.



*Figure 30: Proposed Text Lens System for 0.7-in Micro-display*

We have previously mentioned that loss is a big factor and we need to be aware of it in our system. A combiner such as a beam splitter can usually reflect around 5-50% of the light without taking into account the loss. For instance, a beam splitter with a 10/90 reflection and transmission ratio, assuming that there's roughly 4% loss in the system, will result in only 7% of the light throughput. Whereas in a 50/50 ratio, the light throughput in the system is around 23% assuming still 4% loss in the system. This can become a problem for our system and even make it difficult to be used in broad daylight. It also makes it challenging to choose the beamsplitter to ensure that the right amount of light is reflected on the beam splitter for the user to see the display. It also makes it hard to understand and track all the loss occurring in the system with the additional lenses. With this in mind, we have to consider that in darker environments, the display can be too bright for the user. This means in very dark settings the system should display at a brightness of around 50-100 nits to ensure that the user is not blinded by such bright light. The user should be able to use it as well in broad daylight and still see a clear image. While it's hard to design a

well-defined balance, the positive side of our system is the fact that we do not have an additional combiner that will be used. Our plated beam splitter will be our combiner, compared to a spherical combiner that is more reflective than transmissive, which would not allow our user to see through the design. We will be using either a plated beam splitter or dark shades of sunglasses in our design to have an easier display. It will also make sure we are not blocking a significant amount of light from the real world. With our systems, our micro-OLED display's brightness will be adjustable to help reduce the effects on the user's eyes and still have a see-through display.

Several designs were considered and tested for the lens projection, as previously anticipated. We ultimately decided to use an LCD TFT, two lenses, a mirror, and a beamsplitter, as shown in Figure 31 below.

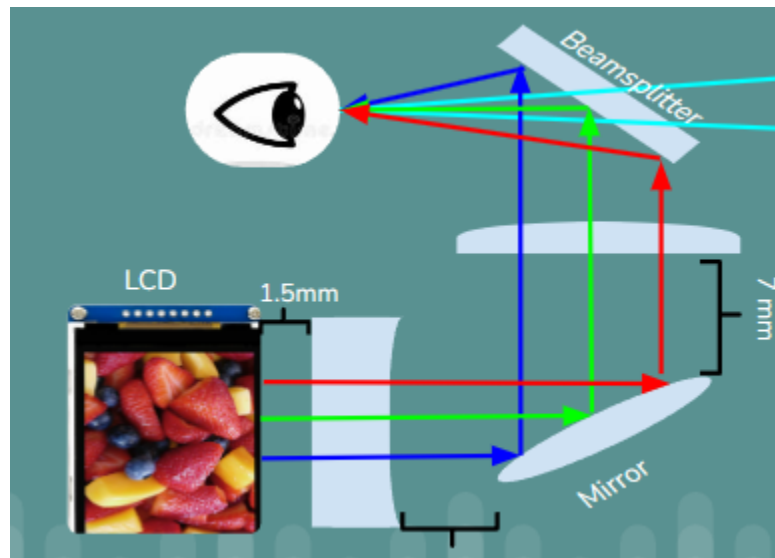


Figure 31. Final Lens Projection Design

For the final design of the lens projecting design, we had decided to use a 1.69-in LCD TFT module of 240 x 280 resolution and brightness of 480 Cd/m<sup>2</sup> or nits. This display was much bigger than originally designed, but it was much more compatible with our components needed for the software design. One of our concerns for this design was choosing the right display that's not only compatible with our other components but also makes sure that the total output brightness of the display will be sufficient. With this in mind, we were able to collimate the light and keep all our optical components in a small dark black box to ensure minimal light loss as seen in Figure 32. This helped create the most optimal and finest projection for the user to see a well-focused projected image with a clear surrounding.

First, the texts and images will travel from the LCD to the plano concave lens, with a 1.5-millimeter distance between the two. This helps to maintain a clear and legible text throughout by minimizing any sort of distortion. Keeping the lens closer to the display, it ensures that the magnification is not decreased, which would occur as the light rays diverge more since a plano concave lens is being used. The lens has a -50 mm focal length and 25.4 millimeters diameter to ensure that the text can be properly centered to capture the entirety of the LCD and roughly 5.5 mm edge thickness, which was taken into account for the design of the hardware. With this lens we were able to design the most effective distance to project the best image without compromising the magnification, by utilizing Eq. 26 and 27 to ensure the correct distances between each component. With these equations, we were able to keep the magnification at 0.952 before utilizing our other lens. This was very important in this design, due to the display size being significantly larger than we had previously designed and the size of all our lenses and mirror being 25.4 mm in diameter. Any major magnifications done to our design would cause the text to be cut off on the sides. It would also result in us needing to minimize the text in the software and would become substantially too small for the user to view.

The lens is then followed by a silver-coated mirror, placed 3.5 mm away at a  $45^\circ$  angle. This enables us to form a real image in the mirror, as well as, mirror the text to be read correctly at the final image. The silver coating helps us maintain a clear and bright image due to this type of coating's high reflectivity greater than 97% compared to regular mirrors. The manufacturing of this type of mirror removes any defects in comparison to regular mirror construction and contributes to its high reflective surface quality. It also removes any distortion or double image, as we have previously tested, that would have been created due to its reflective surface quality. Its reflectance enables us to produce an image with brighter reflectivity while minimizing scattering.

A plano convex lens is placed 7 mm away from the mirror. This also serves to slightly magnify the text and image. In this part of the design, we also utilized equations 9 and 10, which led to a 1.38 magnification of the texts and images without causing any parts to be unseen. This lens helps with the projection of the texts that can easily overlay onto the beam splitter and the user's environment. This final lens of 25.4 mm focal length enables us to maximize the user experience by being able to properly adjust the beam splitter without compromising the user's field of view.

Lastly, we use a beam splitter that has the dimensions of 40 mm x 30 mm x 1 mm and 70T/30R transparency and reflectivity, making it clear enough to allow the user to see both what's in front of them and the text and outlines that are projected from the LCD. The optical design for the text projection was designed for the user to be able to see the text on the reflector/beamsplitter, while still being capable of seeing their environment, which is very much possible with 70%

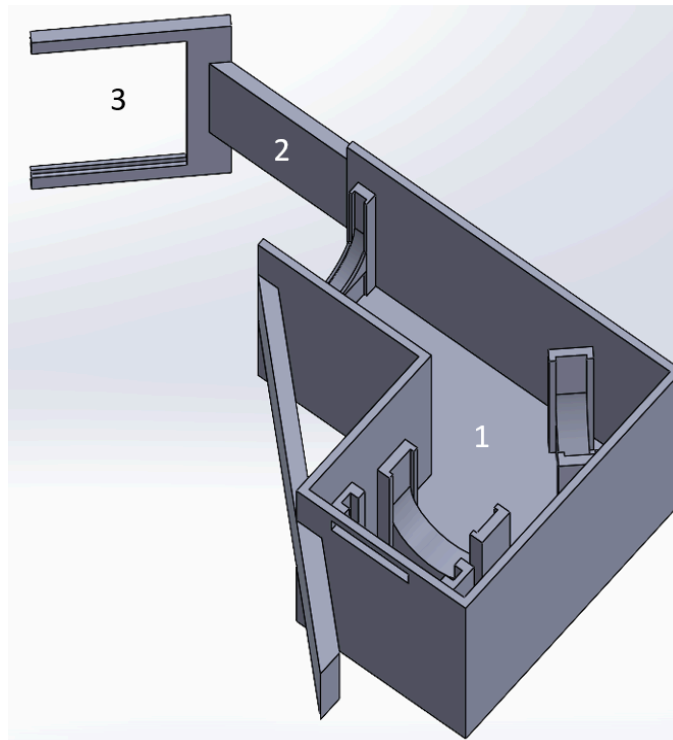


transparency. The beam splitter is coated with an anti-reflection coating to make sure the text is clear and is at the best optimal distance with the help of the final lens. We have previously wanted to use clear acrylic sheets as our reflector for the image, but we quickly realized the poorness of the image quality. Utilizing a beam splitter with anti-reflection coating, helps us to reduce reflections on the surface that may cause ghosting to occur. The use of the beam splitter greatly helps our design to guarantee transparency and enhance clarity while reducing any glare that may affect the images or the user's experience.

### Hardware Design of LCD Projection Housing

The overall design of the glasses can be broken down into two main pieces: the thick glasses with the Raspberry Pi 5 and camera/microphone attachment and the main optical housing that holds the lens system for LCD AR projection to occur.

LCD Projection Housing: The following picture (Figure 4) shows the Solidworks model of the LCD Housing which has 3 sub-parts, the main housing (1), the support beam for the beamsplitter (2), and the beamsplitter holder (3).



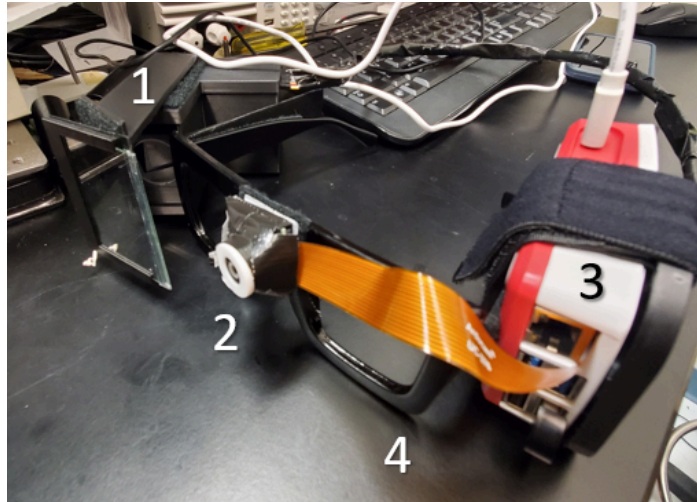
*Figure 32: Solidworks LCD Housing*  
*This housing has 3 sub-parts, the Main Housing (1), the Support Beam for the Beamsplitter (2), and the Beamsplitter Holder (3).*

The beamsplitter holder (3) is designed to hold our glasses' beam splitter which has the dimensions that stay respective to the beamsplitter, as previously mentioned, also has a flat wide base that stretches to 10 mm which provides additional flexibility to the support beam. The support piece for the beamsplitter holder (2) is designed to hold the beamsplitter and LCD housing close to each other. With the optical design, we had to consider the eye box of the glasses. This is the area where the text/image is displayed at the right focus. This also includes the range of eye movements, while maintaining a clear image. While a larger eye box is much more beneficial for the user without compromising the clarity and visibility of the image, it can be a great disadvantage that affects the field of view of the user. With this in mind, the end of the beam splitter is angled directly toward the LCD housing, giving the user the ability to see the text and outlines. The support piece is also lightened with industrial-strength velcro for the outside of the LCD housing, giving the user the option to easily adjust the beamsplitter to their preferred viewing distance. This takes into consideration the variety of users that will be able to use these glasses and gives them the ability to easily adjust the beam splitter based on individuals.

The main piece, the LCD projection housing (1), is designed to fit each of the required optical components (lenses, LCD, and mirror) at the proper distance between each other with little to no wiggle room. The shape of the outer shell of the housing is thin enough to keep the structure together with a thickness of 2 mm. The long wide piece that sticks out off the side of the housing opposite to the side that holds the beamsplitter is a surface aligned with velcro that allows the LCD housing to stick to the glasses frame easily, also allowing the user to adjust the projection to his/her fitting. There is also a roof for the housing that helps close the LCD projection shut and keep all the optical components in their fixed place. There is also another support beam that will be aligned with the top of the LCD housing, aka, the roof of it.

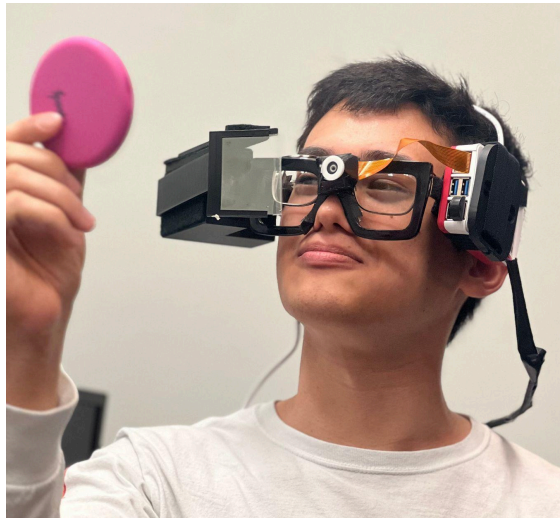
Glasses: Our initial approach was to procure a pair of thick glasses that any individual, regardless of already wearing spectacles, could still utilize. While we still plan to try reaching that goal, our current prototype still allows the user to easily wear the glasses with little trouble in maintaining them and keep the housing of the LCD projection system on their side as well. The thick pair of glasses are commercially available and gave us the convenience to apply velcro to the sides and top of the glasses to allow easy implementation of the LCD housing with the beamsplitter attached on the side. The other side also has a counterweight to balance the glasses. The Raspberry Pi 5 and the cooling fan are mounted on the side of the user's head with velcro applied to keep it centered, with wired running off the side to the LCD housing to the input components of the camera and microphone attached to the top front of the glasses. Again, thanks to the huge frame of the glasses, the components can be easily velcroed/taped on.

The final overall design, shown in Figures 33 and 34, has led to the project being about 10 ounces, just under 1 pound. This combined with the bulky nature of the current version of the glasses has led to users having to try to hold the glasses up while looking around. However, the design has little interference with the glasses' main functionality.



*Figure 33. Project I.R.A.S. Final Design*

*This design can be divided into 4 parts, the LCD Projection Housing (1), the Mutated Lens System + Camera (2), the Raspberry Pi 5 (3), and the Main Glasses Frame (4).*



*Figure 34. One of the Colleagues Testing the Glasses' Color Detection Values*

Further testing is required but so far minimal shaky traversal shows positive results and people who have been diagnosed with color blindness have given helpful feedback to help improve our project. Valued feedback was encouraged to better suit these glasses for everyone.

## 7.0 Software Design

This section will go over all the software-related aspects of this project. It will show the methods we will most likely take to develop each system. This is important so that we have an idea of the steps we can take. This includes capturing audio and video, transmitting that data over WiFi, setting up a server to handle that data, implementing the appropriate algorithms, and processing the response.

The Use Cases diagram below depicts the various ways in which users can interact with the system to achieve their desired goals. The primary actors are the user and the system. The user can interact with the system to perform a variety of tasks, such as:

- **Select Language:** The user can select the language they would like to be translated in the Speech Translation mode.
- **Select Colors:** The user can select the colors they would like to be recognized in the Color Detection mode.
- **Connect Wi-Fi:** The user can connect the glasses to Wi-Fi for wireless communication.
- **Speech-to-Text:** The system sends the audio data to a transcription service in Speech Translation mode.
- **Start Speech Translation:** The user can turn on the Speech Translation mode. This will start a series of work for the system to execute and deliver the result back to the user in real time.
- **End Speech Translation:** The user can turn off the Speech Translation mode.
- **Start Color Detection:** The user can turn on the Color Detection mode. This will start a series of work for the system to execute and deliver the result back to the user in real time.
- **End Color Detection:** The user can turn off the Color Detection mode.

The system also provides several features that support the user's use cases and experience, such as:

- **Capture Audio:** The system captures audio in Speech Translation mode.
- **Speech-to-Text:** The system sends the audio data to a transcription and translation service in Speech Translation mode.

- Translate: The translation service translates the text and sends it back to the system in Speech Translation mode.
- Display Translation: The system sends the received translation to the connected display panel.
- Capture Video: The system captures audio in Color Detection mode.
- Analyze Color: The system analyzes the input data through an onboard program that detects colors in Speech Translation mode.
- Display Translation: The system sends the result color indicators to the connected display panel.

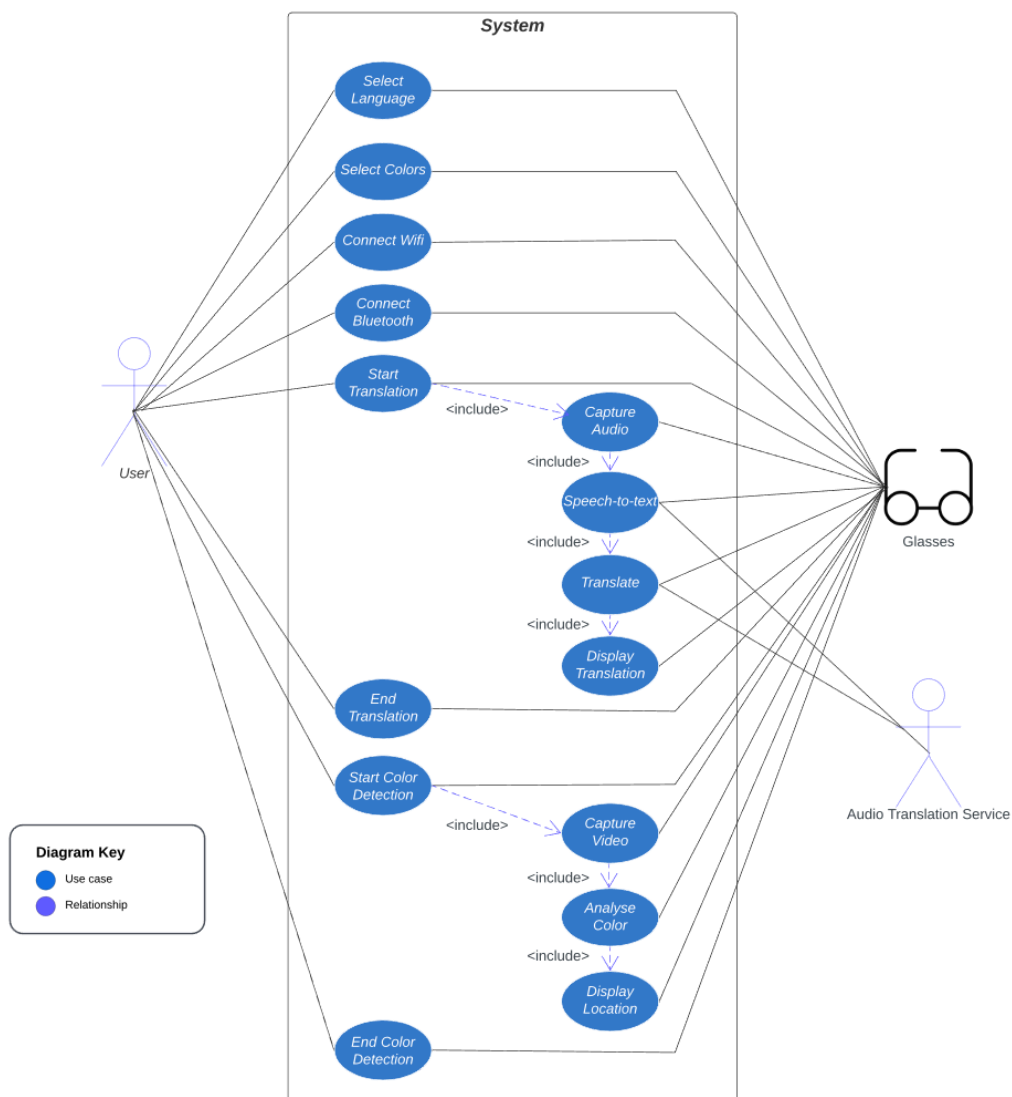


Figure 35. Use Cases Diagram

The following figure shows the series of actions that will occur in our live transcription system. It has two sections for the user and the glasses which indicates which entity is taking those actions. This diagram is useful for understanding at a high level what actions are taken when using live transcription.

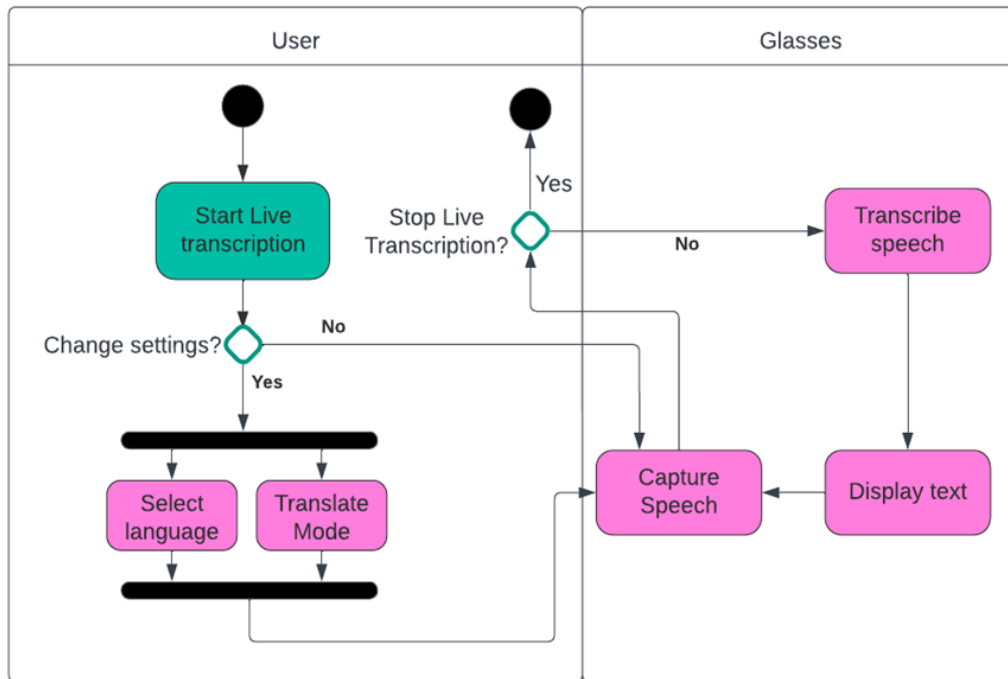


Figure 36. Activity Diagram for Live transcription

## 7.1 Video Frames Capturing with the XIAO ESP32

In our first attempt at using and setting up a XIAO ESP32S3 as our one main board in the Arduino IDE, we installed the ESP32 Camera library and initialized the camera sensor. The most popular choice was the ESP32 CameraWebServer sketch, which is an example provided by Espressif for controlling the ESP32 camera functionality, and it was what we modified and used for this early prototype. Within the example code, we could configure camera settings such as resolution and frame rate and adjust them according to our requirements. This step required some testing to find the right setup that gives the best performance to our needs. To facilitate video streaming for testing purposes, we needed to establish a Wi-Fi connection on the XIAO ESP32S3 by adding the Wi-Fi library and providing the necessary credentials to host a web server. This server handles the transmission of video frames from the camera. We used libraries like ESPAsyncWebServer for asynchronous communication, which can in turn enhance the responsiveness of the server. The Arduino sketch needed to

capture video frames from the camera and transmit them to the connected clients. This involves regularly capturing frames, compressing them if needed, and sending the data over the established Wi-Fi connection. The streamed video is then received and can be displayed on a web browser by running the IP address. We found that the dependence on wifi connection for camera feed streaming was not good on the XIAO ESP32 as it was slow and unreliable for the majority of time due to the hardware's constraints.

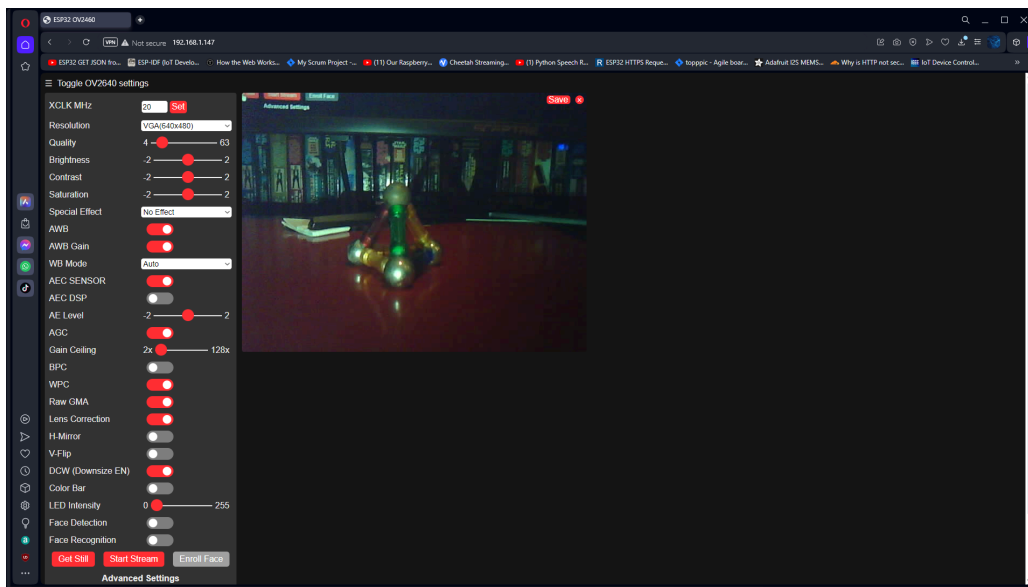


Figure 37. Video Streaming Demo

## 7.2 OpenCV Integration

Our goal was to seamlessly integrate hardware and software components, enabling the XIAO ESP32S3 to continuously capture and analyze color information from its surroundings. We had two options, OpenCV integration with MicroPython or OpenCV.js.

MicroPython is a lightweight and streamlined version of the Python 3 programming language specifically designed for microcontrollers and resource-constrained environments. It incorporates a carefully selected subset of the Python standard library, ensuring optimal performance and efficiency. Since its inception in 2014, MicroPython has expanded its support to numerous microcontrollers, including the ESP32S3. We hoped for it to provide a balance between simplicity and functionality, allowing us to focus on the core color detection algorithms while benefiting from a user-friendly and efficient programming environment. Implementing a full-fledged OpenCV library on the Xiao ESP32 with MicroPython is challenging due to the microcontroller's limited resources. OpenCV's large memory footprint and computationally demanding

algorithms can easily overwhelm the ESP32's RAM and processing capabilities. Furthermore, MicroPython's nature as an interpreted language, along with the potential lack of mature OpenCV ports for MicroPython, can lead to performance bottlenecks. We also found it difficult to find actual resources for implementing MicroPython to OpenCV. While OpenCV ports for MicroPython might exist, they are less mature or less optimized than the main OpenCV library. This could limit available functionality and the performance that we want for real-time color detection.

We then tried OpenCV.js which is a JavaScript library designed to bring the capabilities of the OpenCV library to web browsers. It lets you perform real-time image and video processing tasks like feature detection, object recognition, and image manipulation directly within your web applications. While optimized, the browser performance still limits the complexity of computer vision tasks compared to native desktop applications. Therefore, we didn't want to pursue these routes any further due to the lack of time and human resources and switched over to a surer approach, which is using OpenCV Python on a Raspberry Pi 5 later on to deliver the best possible performance despite the trade-off in form factor.

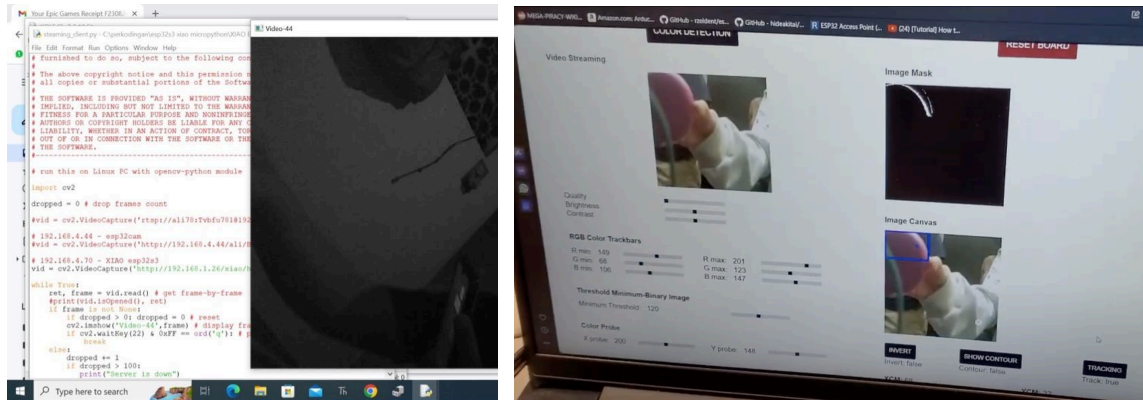


Figure 38. MicroPython (left) and OpenCV.js (right) Demos Operating on a XIAO ESP32S3 Sense

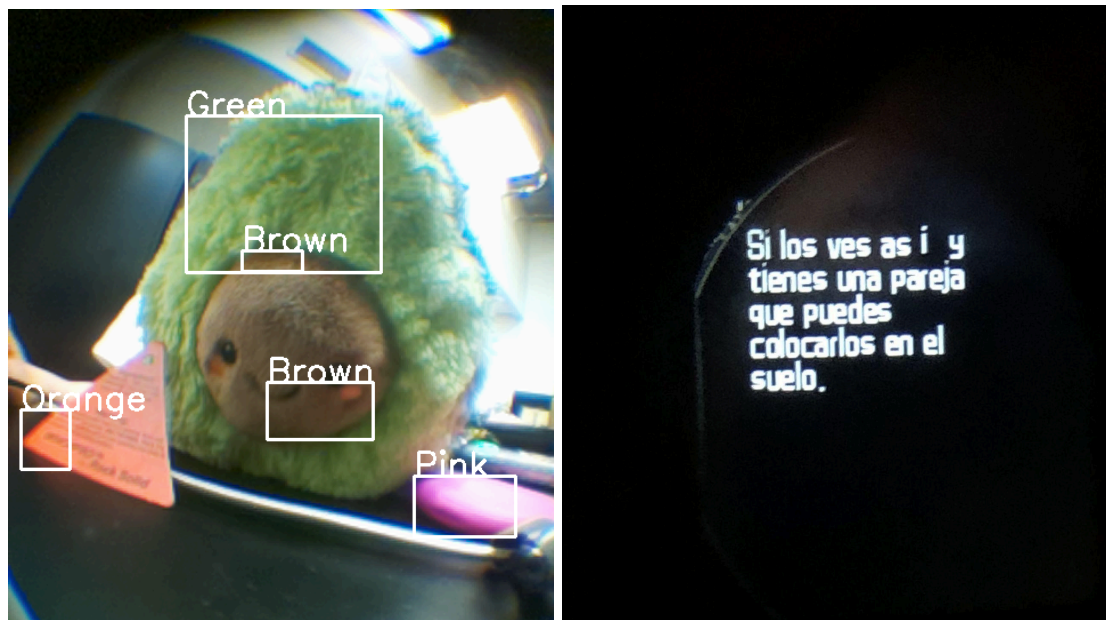
## 7.3 GUI Design

For our smart glasses GUI design, our group follows a user-centric approach that prioritizes clarity, simplicity, and intuitive navigation. We keep in mind the unique context of smart glasses usage and optimizing the interface for hands-free operation. We started with a minimalist design, focusing on displaying essential information and functions without overwhelming users. We wanted to use clear icons and labels to guide users through the app effortlessly. Optimal text sizes are crucial for smart glasses usage hence we'll carefully select font sizes that are easy to read on the display, considering variations in lighting conditions and individual user preferences. To enhance readability and reduce eye strain, we



employed high contrast between text and background colors. This design choice ensures clear visibility under various lighting conditions.

Since clear feedback is essential for a smooth user experience, we provide clear visual cues to respond to user actions and state changes on the app, ensuring users are always informed about the app's status. For one of our stretch goals, we would like to incorporate personalization options to cater to individual user preferences. Users would be able to customize certain aspects of the GUI, such as color schemes or font sizes, to enhance their comfort and experience. To provide users with a smooth onboarding experience, we wanted to include user assistance features and tutorials that guide them through the initial setup and usage of the device. This should help users quickly familiarize themselves with the interface and its capabilities. Another key aspect is the importance of battery life for wearable devices. We'd like to incorporate a prominent and easy-to-understand battery indicator into the GUI. Users would be informed about the remaining battery life, allowing them to manage their device usage accordingly. Unfortunately, time and hardware constraints like battery power prevented full implementation of our design and stretch goals. We were unable to connect a portable battery to the Raspberry Pi 5 due to the complication of finding a 5A 5V battery pack and the lack of an electrical engineer student to figure these hardware problems. By following these design principles, we aim to create a user-friendly and intuitive GUI that complements the functionality of our smart glasses application, providing a simple and engaging user experience.



*Figure 39. Our GUI for Color Detection and Speech-to-Text.*

## 7.4 Display Panel Connection

The Xiao ESP32S3 supports both SPI and I2C interfaces for display communication. SPI offers high-speed data transfer while I2C is ideal for short-range connections to nearby sensors. To connect a display, we need to install a suitable driver library for our specific display and configure its parameters. We then need to prepare data according to the chosen interface (SPI: bytes, I2C: packets) and use library functions to transmit it to the display, updating the screen as well as handle ongoing data updates to keep the display current. A summary of the specific pinouts for the Xiao ESP32S3:

SPI Interface:

- CLK (Serial Clock): GPIO15
- MOSI (Master Out, Slave In): GPIO13
- MISO (Master In, Slave Out): GPIO12
- CS (Chip Select): GPIO5 (can be changed in software)

I2C Interface:

- SCL (Serial Clock): GPIO21
- SDA (Serial Data): GPIO22

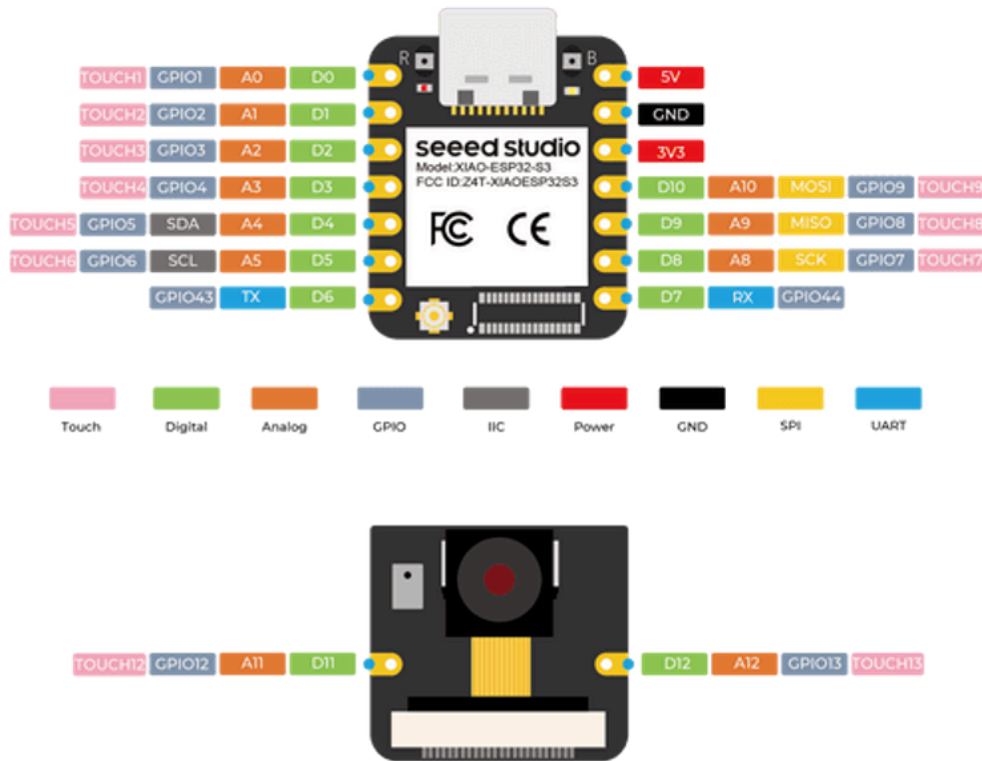


Figure 40. Location of I/O Pins on the XIAO ESP32S3

The Raspberry Pi 5 offers several ways to connect displays: HDMI for standard monitors, MIPI DSI for specialized embedded displays, or GPIO pins configurable for SPI or I2C communication. We chose the HDMI interface for our testing purpose, SPI for the LCD display, and then installed a compatible Python library and initialized it according to our display's specifications. We also prepared image data and used the library functions to send it to the display, then implemented code to handle ongoing display updates for a dynamic user interface.

## 7.5 Mobile Application for Remote Control

A mobile app provides a user-friendly remote control interface for the smart glasses, eliminating the need for physical buttons or conspicuous gestures. It expands the glasses' functionality by offering personalized settings, seamless interactions with other devices, and easy over-the-air (OTA) updates. This creates a more versatile and convenient user experience. This is crucial for our smart glasses as we don't have a dedicated engineer to install extra hardware like buttons as well as providing a discrete experience for our users. There are a few IoT (Internet of Things) mobile apps available currently that offer user-friendly interfaces for creating mobile apps to control and monitor IoT devices:

Blynk is a powerful and popular IoT platform offering a drag-and-drop interface that makes it easy for users with no coding experience to create mobile apps. It supports Bluetooth, Wi-Fi, Ethernet, and other communication protocols, making it versatile for connecting to various IoT devices. Blynk provides various widgets for visualizing real-time data from IoT devices, such as graphs, charts, and gauges. However, Blynk's user interface customization options are somewhat limited, and advanced functionality will require writing code using Blynk's built-in Arduino sketch editor. The following figure shows what the Blynk app looks like.

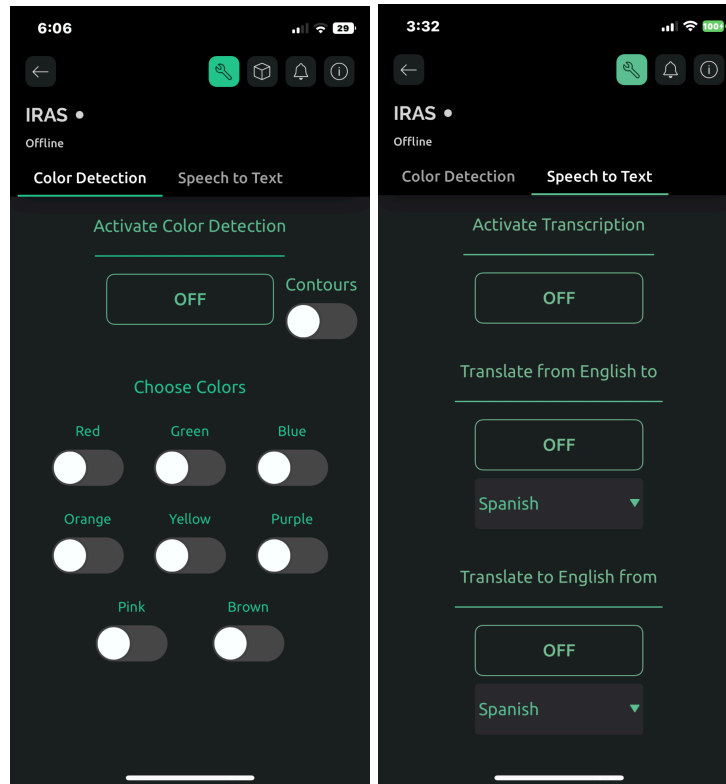


Figure 41. Using the Blynk Mobile with Our Template

Ubidots offers a simple and intuitive interface that is easy to navigate. It provides powerful data visualization tools for analyzing and understanding sensor data. Ubidots offers cloud storage for data collected from IoT devices, allowing users to access and analyze it from anywhere. The disadvantages are that they support a limited range of IoT devices compared to other platforms and require a paid subscription for advanced features, such as custom data processing and alerts.

ThingSpeak is an open-source platform that allows users to extend its functionality and customize it for their specific needs. It offers flexible data storage options, including cloud storage, CSV files, and MATLAB files. ThingSpeak supports various programming languages, including Arduino, Python, and MATLAB, for device communication and data processing. However, it is only a web app and requires more work to turn it into a mobile app. There is an easy way of exporting its data to the MIT App Inventor where it works like the other IoT control apps but both of them are still limited in UI customization options.

Cayenne offers a drag-and-drop interface for creating mobile apps, making it easy to use for beginners. It provides built-in device management tools for managing and controlling multiple IoT devices and supports a wide range of IoT platforms, including Arduino, Raspberry Pi, and Espressif. The disadvantages

are that they require a paid subscription for advanced features like Ubidots, and their real-time data visualization options are somewhat limited.

MQTT Dash is a platform suitable for resource-constrained devices. As the name suggests, it utilizes the MQTT protocol, which is a lightweight and efficient protocol for IoT device communication. MQTT Dash is an open-source platform that allows users to extend its functionality and customize it for their specific needs. Like other apps, its UI customization options are also limited, and setting up and configuring MQTT Dash requires some technical knowledge of IoT protocols and device communication.

The common disadvantages of these IoT mobile apps are that, while they get the jobs done quickly and easily, they are limited in UI customization options, and require an initial technical setup for the IoT devices since they are made to be compatible with a lot of platforms, and often are not free to use advanced features. Thus they can be less user-friendly and don't portray well the concept of our product as a personalized app does. Although it is a better idea to create our own mobile app, there is a big time constraint on our end as well since we only have two programmers on the team. Therefore, we would like to make this a stretched goal and facilitate it in the future if the circumstances allow.

To create a mobile app to wirelessly instruct the glasses and provide a more in-depth customization experience, we can first select a mobile app development platform that suits the skill level and requirements. Popular options are Flutter, React Native, and Kotlin Native.

- Flutter is a cross-platform development framework with a single codebase for both iOS and Android apps. Flutter is considered the easiest to learn among the three due to its declarative syntax and hot reload feature, which allows users to see changes instantly without having to restart the app.
- React Native is a JavaScript-based framework for building native mobile apps using a declarative coding style. It is also relatively easy to learn if developers have experience with JavaScript or React. Its component-based approach makes it easy to create modular and reusable UI elements.
- Kotlin Native is a multiplatform framework for developing native mobile apps using Kotlin, a programming language similar to Java. It has a steeper learning curve compared to Flutter and React Native due to its more traditional programming.

The following figure shows our Figma design for what the user interface might look like. The home screen will have two buttons to activate the speech-to-text mode and color detection mode. When one button is active, the other can't be activated. There is a settings button that will bring you to the settings page. In the settings page, the user can choose which language they want to transcribe, and what language they want the transcription to be in. They can also choose which

type of color blindness to detect. The user can return to the home screen using the home button. If the user doesn't adjust the settings, a default configuration will be used.

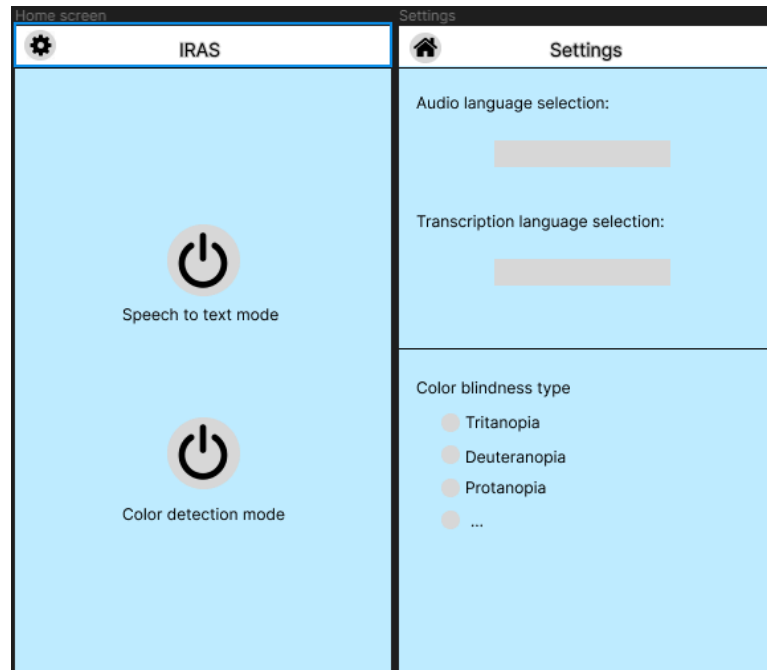


Figure 42: Figma UI design

Our final design looked similar to our Figma mockup. We had two main buttons for color detection and transcription. The two different modes were separated on different pages. We ended up using two more buttons for activating translations and contours. Additionally, instead of letting the user choose a type of color blindness, we listed individual colors which improves flexibility. Because we used Blynk, we were limited in the amount of customization we could do.

## 7.6 Arduino IDE

Our initial MCU was a Xiao ESP32-S3-Sense but it can be programmed using the Arduino IDE. The Arduino IDE is a great programming environment for beginners in embedded systems. It has a huge online community and plenty of documentation example programs. This makes it easy to start with. It provides less control over our MCU as compared to using just the ESP-IDF, but it's able to use tools and libraries from the ESP-IDF so it's a good compromise. To use the Arduino IDE with our MCU, there's some setup required. First, we should download the latest version of the Arduino IDE. Then we'll have to add a custom Arduino-esp32 board manager in the settings. After that, we'll have to install the latest version of the ESP32 board manager. Finally, to upload programs to our MCU we'll have to select the XIAO\_ESP32S3 board, a COM port, and simply

click upload. We can test it out by using the example sketches provided in the software. In the Arduino IDE, we'll be coding in C and C++.

## 7.7 Internet Connection

We'll need an internet connection because we'll be sending our captured speech to a server using an online speech-to-text service. This way, we don't need to have a speech recognizer on board. This provides increased accuracy and offloads some of the processing. To connect to the internet, we'll use the Arduino Wi-Fi library. This library provides the tools we need and many guides online showing how to use them and troubleshoot issues. Typically, a connection will require an SSID, the Wi-Fi name, and the password. If we're trying to connect to a public Wi-Fi, we can omit the password. Then it's just as easy as typing one command. We'll want to use a strong Wi-Fi connection to minimize latency.

## 7.8 Capturing Audio

Our MCU has an embedded digital I2S microphone, so we'll be using I2S to capture audio. I2S is an electric serial interface that provides a way to communicate digital audio data. Arduino and the ESP-IDF both have an I2S library. For setting up the I2S microphone, example programs for our MCU used the Arduino library. This simplifies the process and requires much less configuration. If we wanted to use the I2S tools from ESP-IDF to configure the microphone, we must know the PINS that support I2S on our MCU. We were unable to find documentation on the I2S pins in our device. This may be because our MCU is designed by a third party that is separate from Espressif which makes the ESP32 chips, or there simply may be no I2S pins. Regardless, the Arduino I2S serves our needs well.

To capture audio data, we'll again use tools from the I2S library but this time we'll use the ESP-IDF library. All the documentation online including our MCU's manufacturer uses this method. Its parameters require the I2S port number, a pointer to the buffer where the data will be stored, the size of the buffer, and the number of bytes read. This data can be stored on a microSD card or transmitted over Wi-Fi. To store data on an SD card we can use tools provided by the Arduino SD library and examples online.

## 7.9 Transmitting Audio

To transmit audio data to a server we'll first capture the audio using the method described in the previous section. From here we could constantly stream audio using a while loop, or we could transmit audio in batches by recording for a few seconds onto the SD card. There are example programs on how to do these, but our optimal approach would be to stream data in order to minimize latency. We'll

likely use WebSocket for real-time audio data transfer because it's a communication protocol that allows live bi-directional communication between a server and our MCU. To use WebSocket, we'll have to use the `ArduinoWebSockets` library. We'll have to know the IP address of the server that we're sending data to, as well as the port number. When the server is hosted locally, the IP address would be obtained from the Wi-Fi we're using. If we're hosting the server online, the IP address would be the domain name of the server or the public IP address provided by the service we use. The ports we'll use will likely be 8888 and 8000 because these ports are commonly used for HTTP and web server software. We could use many other ports but it's best to conform to the standard. When using these ports, we'll have to make sure it isn't used elsewhere to avoid conflicts and ensure these ports aren't blocked by a firewall or network policy. We'll need to know the IP address and the port number so that the MCU knows where it's sending data.

Many programs that use an embedded system offload complex functionality to a server. This saves processing power, and energy use, offers better network stability, and allows more complex tasks. For these reasons, we'll use a server and transmit our audio there.

## 7.10 Using a Speech-to-Text Service

Setting up an online speech-to-text service will vary depending on the one we choose. We'll have to reference the service's documentation. For example, we've used Google's Speech-to-Text API, and as part of the setup to obtain an API key we had to create a project with a connected billing account, select the Cloud Speech-to-Text API, create a service account with an appropriate role, and create a JSON key which is then downloaded onto the computer. Most speech-to-text services will require something similar where we'll have to set up an account with billing and create an API key but the specific way we'll do it will differ. How we implement the speech-to-text will also differ for each service. For example, with Google, we had to set an environment variable that accesses the file that contains our API key. Then we had to set up a Node.js server that follows the implementation for Node.js provided on Google's website. This will vary for each service.

The speech-to-text service we want to use is Speechmatics. It follows a similar setup and implementation as Google. We must create an account on the Speechmatics website. Then we have to create an API key and store it somewhere secure. Finally, we can implement the server code using the example provided on their website. For real-time transcription, they support CLI, Python, and NodeJS. We'll probably use NodeJS as this is what many example programs involving a server use.



## 7.11 Server

The server is where we'll be sending all our audio data. It will handle all the speech-to-text processes and transmit the transcripts back to our MCU. Using a server to handle the speech-to-text functionality allows the ESP32 to dedicate memory and processing power elsewhere. We will most likely use a NodeJS server to perform this logic. NodeJS is a very popular run-time environment used for server-side programming and networking. To use NodeJS, we'll have to install it on the computer we're using. NodeJS uses npm or the node package manager. Packages are pieces of software that are available for developers to use. We'll have to install packages for the speech-to-text service we use. For example, Google's speech-to-text has a speech library that's different from Speechmatics'. The NodeJS server can be hosted locally or online. Right now we're hosting it locally for testing purposes, but we'll want to shift to an online server. To do this we could use free services like Amazon Web Service or Microsoft Azure. We'll have to reference documentation and example implementations to deploy an online NodeJS server. The programming language we'll use for NodeJS is JavaScript.

We may also create an HTTP server using NodeJS's tools to handle HTTP requests such as for receiving an audio file or responding with transcriptions. We could also use Express which is a back-end framework for creating RESTful APIs. It could be used for serving the user an HTML file to create an interactable application either for testing or consumer use.

We are trying to stream audio data so the server will likely use WebSocket. NodeJS provides the tools to use WebSocket. As mentioned previously, we'll need to know the IP address and port we're streaming the data to. Then we can use examples and resources online to use WebSocket in a NodeJS server.

The following sequence diagram depicts how our system for transcribing speech may look like regarding the server and data transfer. It shows when each part of the system is active and what they'll do.

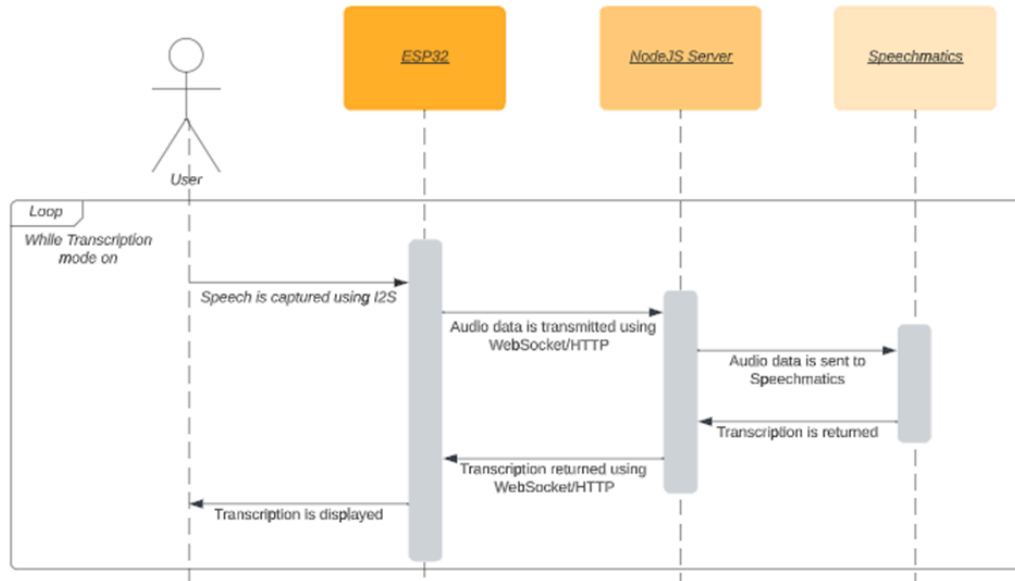


Figure 43: Sequence Diagram for Audio Data Transfer and Server Interaction

## 7.12 Switch to Raspberry Pi 5

While we initially used the XIAO ESP32S3 Sense, when we started to develop the color detection, we found that it was too slow in terms of fps, and real time delay. This is when we switched to using one of the Raspberry Pi models, and ended up using the Raspberry Pi 5. This came with a much larger form factor, power consumption, and price, but with the benefit of massively increased capability and speed.

To use the camera, we first had to find a library that supports our camera. The Raspberry Pi 5 is a very new release so we couldn't find much relevant information on using cameras with it. With a lot of research we found the Picamera2 library which allowed us to successfully control our camera. Then we connected the ST7789V2 SPI display to our Pi and followed a tutorial on how to use it from the manufacturer, Waveshare. This was relatively simple although we did run into a few troubles with outdated instructions and conflicting libraries and modules installed using pip and apt. Apt is a package management tool used for Debian distributions, while pip is a package management tool specific to Python. We solved this issue by using a virtual environment. A virtual environment is an isolated environment with its own packages and libraries separate from the global environment. This allows us to install libraries that can only be installed using pip and avoid conflict issues. This was critical because one of the main libraries that we need, Speechmatics, is only installable using pip.

After we set up the display and camera and verified they were working, we started developing the color detection, as this was the main reason we switched

boards. We installed OpenCV and looked at their documentation and examples that used OpenCV. Implementing Color detection using OpenCV was fairly straightforward and easy in Python. When using the ESP32S3 Sense we were only able to show the bounding boxes of detected colors on the display. These bounding boxes weren't aligned with the real view of the user, so it would be hard to tell what object is being detected. The frame rate was at most 1 fps, and it relied on an internet connection. Using the Pi 5, we were able to show the live camera feed with the bounding boxes on the display. The latency is almost imperceptible, the frame rate is decent as well at 10-20 fps, and no internet connection is necessary for the OpenCV. This switch allowed us to massively improve our color detection.

Once we had the color detection working, we started working on the speech to text. First we ordered a microphone that was compatible with the Pi 5 and verified it was working and adequate for speech recognition. When we implemented speech to text on the ESP32S3, we used a NodeJS server because Speechmatics wasn't able to be used on board. Some caveats to this implementation are increased latency, and more complexity. For example, one issue we ran into and fixed was the server thinking it was still connected to the ESP32S3 even though it was unpowered. This resulted in an error the next time the ESP32S3 was powered and tried to send audio to the server. We fixed this by implementing a ping pong system that checks if the connection is still alive every 5 seconds. Using the Pi 5, we could completely eliminate the server and directly send audio to Speechmatics' server. This decreased latency and reduced complexity. This was pretty simple because Speechmatics provided examples in Python on how to use their API. Because we used Speechmatics, translating speech was very easy to do and only required updating the config with whatever language is supported by Speechmatics, and which mode, either translating from English or to English.

The next step was to combine both systems so that the user can easily switch between modes. The main issue we ran into was that the speech to text program had to be run in its own virtual environment. This meant we couldn't use threads, so we used subprocesses. This was straightforward but we ran into the issue of GPIO pins being in use. This was because both programs used the SPI display, even though we cleaned up the resources used for the display using the display library's functions before calling the subprocess. We couldn't find a solution to this issue, so we combined the speech to text program's display printing with the color detection's. This led to the issue of obtaining the text from the speech to text program in the main, color detection program. The various methods we tried to use blocked the execution of the program, so the main program would wait for forever to read text and wouldn't be able to switch back to color detection. The final solution we used was to read the stdout from the speech to text program using the select module, which is able to be used in a non-blocking manner. This allowed us to read text as it's printed from the subprocess without affecting the

main program's execution. For translation, we passed arguments to the subprocess which indicate the mode of translation and the target translation language. One thing we were able to reuse from the ESP32S3 was the text printing functionality. Previously we had formatted the text so that it wasn't too small or large, had consistent spacing, centered margins, and would calculate the length of the current word and print it on the next line or refresh the screen if it cut into the margins. It was written in C++, but was easy to translate to Python. We used Noto fonts from Google for specific languages where some characters were omitted or didn't look right when printed.

After we were able to successfully combine both programs, we started using Blynk to control our device. We used virtual pins which would listen for a change in the value of a datastream using event decorators. This was setup using the online Blynk dashboard and mobile app. The value of a datastream is changeable using the Blynk app through widgets such as a button. We used virtual pins for every element the user would interact with, those being the mode buttons, color select buttons, and language selector dropdowns. The color detection program serves as the main program, so this is where we combined the Blynk functionality. The user can stop and start the color detection at the press of a button and can change the colors detected in real time. Whenever the user activates transcription or translation, a subprocess is started with the corresponding arguments. This subprocess is stopped whenever the user switches modes. With Blynk, integrating an app with our program was easy to do.

The following activity diagram displays how the Blynk app interacts with the main program and how it affects its operation. The main program is always active so that it can listen for changes in the value of the Blynk virtual pins. The speech to text subprocess only gets activated when the user chooses transcription/translation. The color detection program kills the subprocess if it's active. The user changing speech to text modes, i.e. from transcription to translation into English, or changing languages kills the current subprocess and starts a new one. Another thing the diagram doesn't depict is what happens if no mode is active. In this case, the main program is still active and simply listening to the Blynk virtual pins.

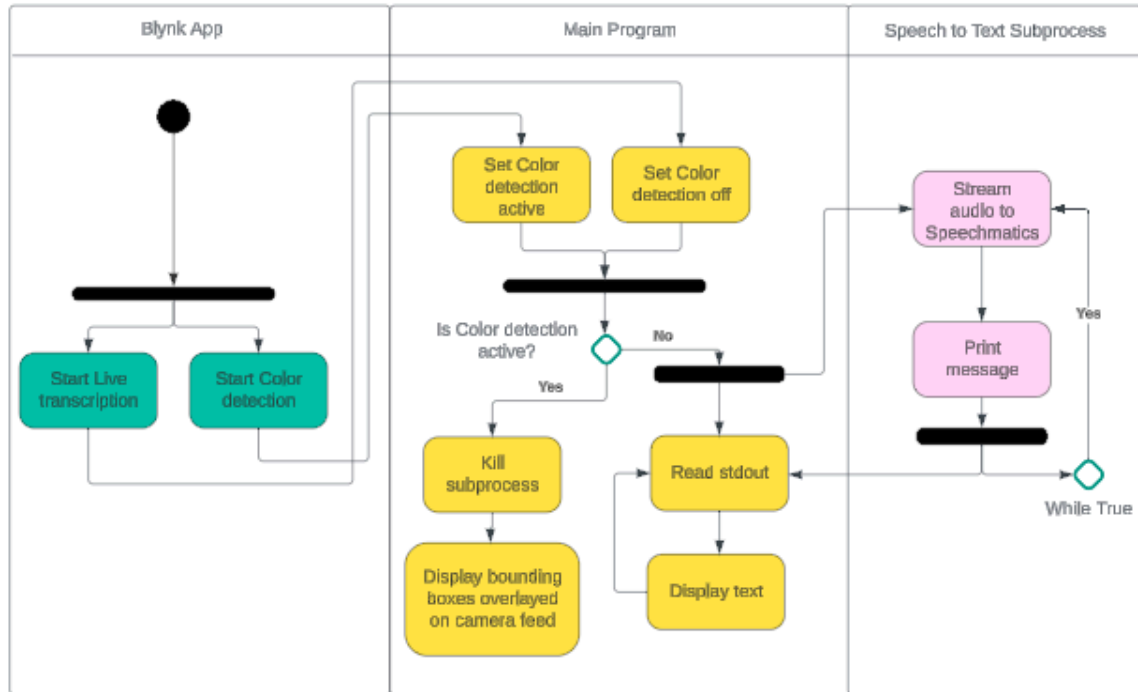


Figure 44: Activity Diagram for Blynk and Program operation

## 8.0 System Testing

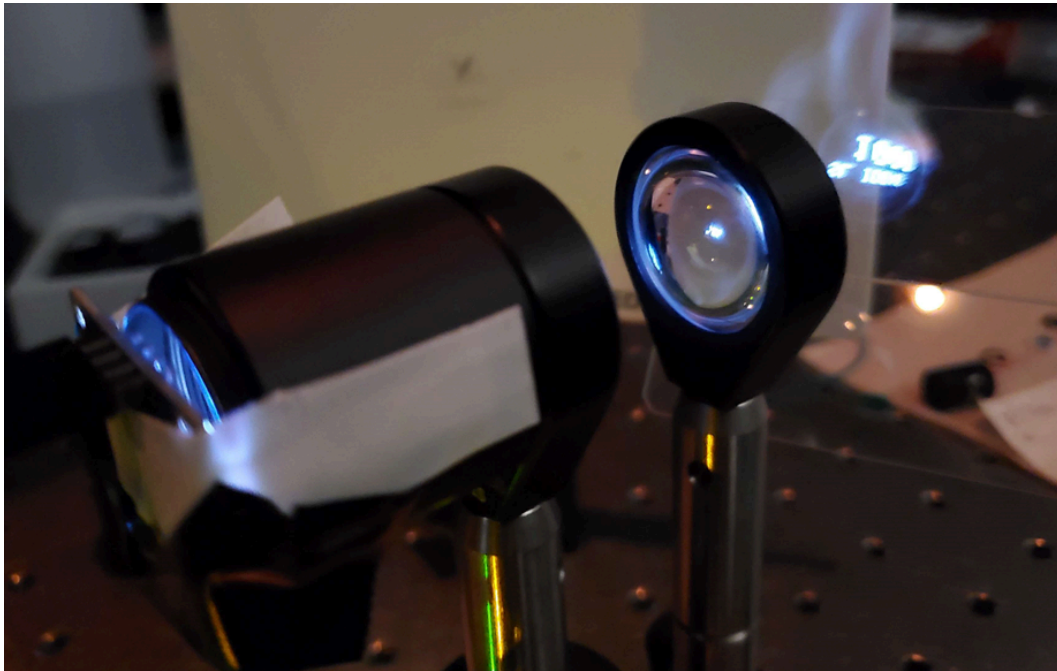
In this section, we will cover the parts that we have tested for our prototype. We will mention the challenges that have arisen and how we plan on fixing these issues moving forward with our project. Testing is important to know that each part of our system works. We'll be testing each part of our design such as the MCU, camera, microphone, and software. Testing will allow us to spot problems earlier and address them in a timely and ordered manner. It also assures that if a tested part works, we can look elsewhere to solve the problem.

### 8.1 Text and Image Display Testing

For this subset of the design, we will need to assess the LCD and how the image is reflected off the beam splitter. This also means following the standards that we previously mentioned to make sure that our design is safe for use. We need to evaluate whether or not there are any flickers in the system, and if it causes eye fatigue the user may strain his or her eye, or any other issues that may cause vision problems. We will need to assess the right focal length for the lens and the dimensions for the 3-D printed housing that will allow an easy pathway for the display. We will need to have the lens with the correct diameter, and the positioning of each component, such as the LCD, lens, beam splitter, and mirror to ensure that the light is reflected correctly. We will also need to know how the image is displayed in brighter environments, as well as darker. This will be an

issue if the image cannot be seen clearly during the daytime, since this prototype is ideal for day-to-day activities. We have thought of this being an issue, and to incorporate a dark shade of sunglasses, which will help create a darker surrounding for the image to be easily seen.

We purchased a cheaper version of our LCD from Amazon for testing. We were able to test a LCD display, which was significantly dimmer and had less resolution. At this time, we had not purchased any of the actual lenses that we wanted for our design. We utilized materials that were provided in the CREOL senior design lab, which mimicked the lenses we wanted as closely as possible. We used a Bi-Convex lens of 40-millimeter focal length, 25.4 millimeters in diameter, and N-BK7 glass material. The second lens used was a plano-convex lens of 25.4-millimeter focal length, with the same diameter and glass material. The image from the LCD could be easily seen on a microscope slide that we used as our reflector.



*Figure 45. Demonstration of Text Display Lens System*

As shown in Figure 44, this demonstration helped us understand the system we would be working with a lot more, in terms of having tangible parts that we could test instead of just basing every aspect on calculations we have done. We also realized how easily we could lose the image even if the reflector was slightly out of focus. Therefore, our calculations for designing the housing and pairing these components must be as accurate as possible to ensure that the image is projected, reflected, and can be seen by the user. In the testing, we also noticed

that the image did not appear right-side up. This was not an issue here as we were just testing the path of the light and the display, but it will be a problem for our design since the user needs to be able to read the text shown. This is something we will need to keep in mind as well for our designs.

## 8.2 MCU Testing

The easiest way to test if our MCU is functioning correctly is to first, set up the Arduino IDE for the Xiao ESP32-S3, and then use the Arduino blink program. We found it worked perfectly. Some problems we had when testing programs were random failure to upload and random failure to work correctly. For example, when connecting to Wi-Fi, it would try to connect forever even though there's nothing wrong with the code. Luckily the MCU comes with a reset button which usually causes the program to function correctly. Disconnecting and reconnecting the MCU to power also seems to work. If the program fails to upload or the COM port doesn't appear, we can use the boot button to put the MCU in BootLoader mode by holding it down, connecting it to the computer, and releasing it. These two buttons should fix most problems.

Again, we ended up using the Raspberry Pi 5. We verified it worked correctly by flashing a microSD card with the latest Raspbian OS and powering it. We tested all our peripherals and found everything to work correctly. We had some inconsistencies with connecting to the campus network. Sometimes it just wouldn't connect and other times it would connect after a while. Eventually it connected consistently, although we weren't sure what the problem was. Something we found out by destroying 3 microSD cards was that the Pi should only be unpowered using the onboard button or the shutdown option in the desktop or command line. It should never be unpowered by unplugging it from an outlet or else you risk the microSD card becoming unusable.

## 8.3 Camera System Testing

To test the camera, we can use the program provided by SeeedStudio in their online wiki. Our MCU is designed specifically for the OV2640 and OV5640, so SeeedStudio gives an example of how to use them. Some of the examples they give are for taking photos, recording videos, and a live video stream. We tested the OV2640 camera using the video stream example and found that the program worked well, but the video quality wasn't great. Additionally, the stream would occasionally freeze for a few seconds. Despite this, the quality was good enough to discern differently colored objects. We tested with the OV5640 camera and found similar performance. It supports larger resolutions, but the video stream drops to less than one frame per second.

One concern we had when we started testing was the heat produced by the MCU when it was running. We started with a video stream test and the MCU would become too hot to hold and start to smell after it was running for a little while. We talked with an advisor, and he reassured us that this was normal. We can minimize the heat produced by only using the camera when necessary and reducing the resolution used.

The camera we used in the end was the OV5647 camera. We had to use a special cable that would fit the Pi 5's smaller port and the camera's larger port. The main issue we had with using the camera was finding a library. Once we found the Picamera2 library, we were able to test the camera and found it had much better color range, resolution, and frames per second than the XIAO ESP32S3 Sense's OV2640. It maintained image quality even with prolonged use and wasn't too hot.

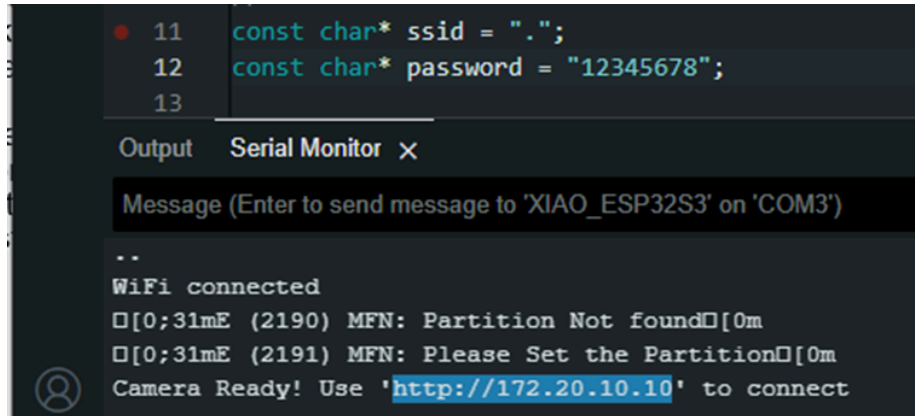
## **8.4 Connecting to Wi-Fi**

Connecting to Wi-Fi is easy using the tools provided by Arduino. When testing the video stream on a private network at home, the connection worked flawlessly and connected every time. When we tested on a public network and a private network at UCF, it had trouble connecting. This may be due to the way networks at UCF are configured or it may be the computer we were testing on. We tried connecting using a mobile hotspot on the same computer and still had trouble connecting, however, when testing the mobile hotspot on the computer that worked originally, the connection worked.

Additional testing showed that the USB port we connected to on the laptop was the issue for connecting to a mobile hotspot. Research showed that the UCF guest Wi-Fi has an authentication page, and the private Wi-Fi is WPA2. To connect to these networks, we'd have to do additional configuration. Further research and testing is required.

The following figure shows what a successful Wi-Fi connection looks like. When we had trouble connecting, the Wi-Fi would try to connect infinitely.





```
11 const char* ssid = ".";
12 const char* password = "12345678";
13
```

Output Serial Monitor X

Message (Enter to send message to 'XIAO\_ESP32S3' on 'COM3')

..

WiFi connected

[0;31mE (2190) MFN: Partition Not found[0m

[0;31mE (2191) MFN: Please Set the Partition[0m

Camera Ready! Use '<http://172.20.10.10>' to connect

*Figure 46: Example of Successful Mobile Hotspot Connection*

The ESP32-S3-Sense comes with an antenna. We tried connecting to a Wi-Fi network without an antenna and found that it wouldn't connect. Therefore, when our testing involves using a Wi-Fi network, we should always have the antenna connected.

We were able to connect to campus networks on the ESP32S3 Sense by using an ESP32 WPA2 library. We ran into issues when testing such as not being able to connect to a locally hosted server on UCF\_WPA2, but being able to do so on a server hosted on Amazon Web Service.

With the Raspberry Pi 5, the wifi settings support WPA-2 Enterprise which is what the UCF\_WPA2 and eduroam networks are. Despite this, connecting to these networks was inconsistent despite us configuring the settings correctly. One day it automatically connected and we never had an issue with it again, so we weren't sure what the problem was. The connection was strong and reliable.

## 8.5 Microphone Testing

We first tested if the microphone worked at all using a program that detects sound loudness. We configured the I2S settings in the setup function. Then we took a reading and printed the value to the serial monitor in the loop function. Arduino has a serial plotter tool that displays the readings in a graph. Using this program, we found that our microphone works and reacts to different levels of noise.

When we obtained a microSD card we then tested if we could record an audio sample to it, again using a program from SeeedStudio. First, we had to check that the card was formatted to FAT32 using an SD card reader. In the program, we first configure the I2S settings and initialize the SD card. Then we record for a preset amount of time and write to a file opened in the SD card. When we played

back the audio files, we found the quality was very good for what the microphone is.

Then we tested streaming audio from the ESP32 to a server and playing the audio back in the browser. We modified an ESP32 streaming program from ThatProject to use different I2S settings. The original program used the I2S functions from the ESP-IDF which requires that we know the I2S pins. Because we couldn't find information on the I2S pins for our mic, we used the Arduino I2S tools for initializing the microphone. Then we changed the WebSocket address to the Wi-Fi IP. Then the audio is streamed to a NodeJS server using WebSocket. An HTTP server is also set up using Express, where we can display an HTML page in a browser. To use NodeJS, we first had to install it. Then, in a terminal, we had to navigate to the folder containing server.js. Then we had to run npm install to install the required dependencies. Then we ran node server.js to start the server. From here, we put the WebSocket address, port, and audio.html path in a web browser. Then we were able to listen to what the microphone was hearing in real-time as well as visualize it in a plot graph. We found that the audio quality was very good. There was a constant clicking noise, but it was quiet enough to not affect the clarity. The audio playback was delayed by a few seconds.

We were able to listen to the playback of the Sunfounder microphone on the Pi 5 by using ALSA. We found that the audio quality was quite poor with a constant clicking noise, but it was clear enough to distinctly hear voices and sensitive enough to pick up speech from across the room. Once we verified the microphone worked, we started using it to send audio to Speechmatics.

## 8.6 Speech to Text Testing

We did some testing using Google's speech to text API. We used an example provided by SeeedStudio that uses this API. It required some setup such as creating an account, making a Google environment variable with our API key, and installing Google's cloud speech package using npm. The program captures a preset time of audio. Then we used a NodeJS server and HTTP to send the audio data to the server. Then we used the code implementation provided by Google in an asynchronous function to transcribe the speech. It's then returned in the HTML response object and the text is shown in the serial monitor in the Arduino IDE.

When the program worked, it only took a few seconds for the speech to be transcribed after it was sent to the server. The problem is that this program requires a recording, and we want speech to be transcribed live. Another problem is that the NodeJS server is hosted locally, so we'd have to be connected to the same network. An issue we ran into was the HTML request

would consistently return with a “-1” error response, and using console logs, the server didn’t appear to be contacted. We’re not sure what caused this, but after resetting the MCU and server multiple times it started working. We found that with quieter recordings, the transcription wasn’t very accurate. Our next steps would be to transcribe speech from live audio input, instead of a recording, and to host the server online. This will require more testing and research to accomplish. Additionally, we want to use Speechmatics’ API instead of Google, so we’ll also perform tests that use Speechmatics.

We set up Speechmatics on a NodeJS server and the Raspberry Pi 5. Because we ultimately used the Pi 5 and both implementations are similar, we’ll only mention the Pi 5. Speechmatics’ API uses WebSocket where we stream audio to Speechmatics and receive transcriptions or translations back using WebSocket event handlers. We start a session using Speechmatics’ configuration settings. Here we were able to customize options such as the maximum delay for a transcription to be formed, the language used, and partial transcriptions. We found that partials weren’t very useful to use because they sacrificed accuracy and printed previous transcriptions repeatedly. The minimum value we were able to use for the max delay option was 2 seconds. With regular English transcription we found that the delay from speech to text being displayed was consistently less than 2 seconds with continuous speech. If the speech suddenly stopped, the last transcription could take longer than 2 seconds. With translations we found that it often took longer to start and for some reason would sometimes keep a larger buffer of translations so instead of printing a sentence or two at a time, we’d receive numerous sentences. This had the negative effect of causing the translation to be printed into the margins of the display and being too long to read before the screen refreshed. We couldn’t find a solution for this.

## **8.7 Color detection testing**

As mentioned previously in 7.2, the OpenCV implementation of color detection on the ESP32S3 Sense was less than subpar. OpenCV on the Pi 5 was much better. The most significant improvement was the frame rate of the camera. This allowed for real-time color tracking, significantly improving the user experience over the previous 3 second delay. When testing with users, some things we found and fixed were the font size of the bounding box captions. We made them larger and bolder. Another was the color of the bounding boxes. One colorblind tester couldn’t easily tell the difference between two blue colored boxes. We discussed heavily on what color would be most visible to all users and if there were colors that would be distinguishable to all color blind people. We settled on white bounding boxes and captions because this would provide the most contrast in most settings. A potential improvement we could’ve made is a user adjustable setting that would change the color of the bounding boxes around certain colors. We adjusted the HSV ranges for the colors we chose to make them pickup colors in different lighting conditions as best as possible. One issue we found was that

the camera didn't detect colors as they were in reality. It would see purple as more of an indigo hue. This led to difficulty with detecting pink and purple. Another issue we encountered was with defining the red ranges. There are two different reds, one at the beginning of the spectrum as it transitions to orange and one at the end where it transitions from violet. This caused us to detect red objects as colors other than red. We fixed this through more testing and adjusting ranges.

## 9.0 Administrative Content

### 9.1 Budget and Funding

The majority of the expenses for this project will primarily be funded by the team. Our current plan to cover some of the costs for this project is to try to reach out to professors who are willing to fund this project as well as companies willing to donate some supplies. Some optical industries we're thinking of reaching out to for spare parts could be Newport or professors who are willing to support us could be those from CREOL. Dr. Wu, in particular, has been a great help in guiding our project in designing our lens system and his many papers on AR/VR systems.

### 9.2 Bill of Materials

The table will be updated throughout the process for the total expenses of the final product. The current estimated cost for parts is shown below.

*Table 15. Budget for Base Prototype*

Item Description	Estimated Cost per Unit
Raspberry Pi 5 Kit	\$120
OV5640 CMOS Cam	\$25
1.69" LCD TFT	\$6
Glasses Frame	\$10
Optical Components (Plano-Convex, Plano-Concave, Doublet Convex, and Convex-Concave, Plate Beamsplitter)	\$150
Blynk App/Speechmatics	\$47

Velcro/Tape	\$25
Battery Bank	\$50
<b>Total Cost</b>	<b>~\$433</b>

### 9.3 Project Milestones

Time is a limiting factor to consider for this project. The project milestones will be used to demonstrate the team's project plan and help monitor progress and important deadlines.

*Table 16. Senior Design 1 Milestones*

Task	Duration	Date	Role	Status
Brainstorming	2 weeks	8/31/23	Everyone	Complete
Divide and conquer	2 weeks	9/15/23	Everyone	Complete
Research Project	1 week	9/15/23	Everyone	Complete
Update Divide and Conquer to Website	1 week	10/6/23	CS	Complete
60 Page Draft Documentation	6 weeks	9/18/23	Everyone	Complete
Update 60 Page to Website	1 week	11/17/23	CS	Complete
120 Page Final Document	4 weeks	12/5/23	Everyone	Complete

*Table 17. Senior Design 2 Milestones*

Task	Duration	Date	Role	Status
Purchase	5 Weeks	12/5/23	Everyone	Complete
Build Prototype	7 weeks	1/12/24	Optics	Complete
Testing & Redesign	4 weeks	3/8/24	Optics	Complete
Finalize Prototype	2 weeks	3/28/24	Everyone	Complete

Peer Presentation	1 week	4/19/24	Everyone	Complete
Final Report	1 week	4/23/24	Everyone	Complete
Final Presentation	1 week	4/19/24	Everyone	Complete

## 9.4 Product Backlog

The following figure shows our product backlog using scrum on Jira. Scrum is a project management framework that focuses on sprints; a designated amount of time for an increment in the development of the project. The product backlog shows a list of the work the Computer Science students will do. These tasks are derived from the objectives to create each part of the software system. The higher up a task is on the list, the more important it is. This gives us a clear idea of what we can work on over the break and in SD2. We didn't use Jira as much as we should've in SD1, but we will use it more in SD2 now that our research is done and we have most of our parts

▼ Backlog (11 issues)		0	0	0	Create sprint
<input checked="" type="checkbox"/>	SCRUM-11 Use Speechmatics to transcribe speech	AUDIO SYSTEM	TO DO	-	👤
<input checked="" type="checkbox"/>	SCRUM-16 Transcribe speech that's streamed from the microphone	AUDIO SYSTEM	TO DO	-	👤
<input checked="" type="checkbox"/>	SCRUM-21 Create a live color detection algorithm	VISUAL SYSTEM	TO DO	-	👤
<input checked="" type="checkbox"/>	SCRUM-20 Install MicroPython and OpenCV onto the ESP32	VISUAL SYSTEM	TO DO	-	👤
<input type="checkbox"/>	<input checked="" type="checkbox"/> SCRUM-22 Display the live feed of color detection on the display	VISUAL SYSTEM	TO DO	-	👤 ...
<input checked="" type="checkbox"/>	SCRUM-15 Host the speech to text server online, and ensure the functional...	AUDIO SYSTEM	TO DO	-	👤
<input checked="" type="checkbox"/>	SCRUM-14 Display text onto the micro-display	AUDIO SYSTEM	TO DO	-	👤
<input type="checkbox"/>	SCRUM-19 As a product user, I want a user interface to be able to change t...		TO DO	-	👤
<input checked="" type="checkbox"/>	SCRUM-17 Transcribe into multiple languages	AUDIO SYSTEM	TO DO	-	👤
<input checked="" type="checkbox"/>	SCRUM-18 Transcribe from multiple languages	AUDIO SYSTEM	TO DO	-	👤
<input checked="" type="checkbox"/>	SCRUM-23 Add different color blindness modes	VISUAL SYSTEM	TO DO	-	👤

Figure 47: Jira Scrum Product Backlog

## 10.0 Conclusion

Project I.R.A.S. aims to make the lives of everyone better through the power of optical principles and coding ingenuity. We as a collective of four undergraduates want to bring more recognition to those who have the challenge of hearing impairment and color blindness in their everyday lives. This device hopefully will help bring a new perspective for those with and without hearing and sight

handicaps. It started from a “what if” scenario, daring to see how far we initially wanted to go for a senior design project to see how far we could achieve a project like this, since we initially had trouble holding our group together, let alone finalize our overall project idea.

I.R.A.S. is what we hope future products will look to help accommodate everyone’s everyday needs. The Integrated Real-time Assistance Spectacle (I.R.A.S.) demonstrates two main modes of disabled assistance which are easily switched between color blindness and hearing impaired. In the color blindness mode, the wide 60° low distortion CMOS camera will pick up the selected distorted colors in the view of the user and display their exact position by utilizing a LCD to project the outlines on the user’s lens. This will help the user track what colors are surrounding their environment and stay more alert. In the hearing impaired mode, the same board that holds the camera will have a digital microphone that will track the speech the user is talking to and receive close captioning displayed on the lens by the LCD projection. We aim to make both modes accommodate extra customization for multiple color blindness treatments as well as detection and translation for multiple languages, allowing the user to travel anywhere around the world and communicate with whoever they meet.

We must acknowledge that our journey was set back with many hurdles, however, our team has fought through with resilience, adaptability, and a strong determination to reach the final goals. Originally comprising five members, with representation from the Optics, Computer Science, and Electrical Engineering departments, we encountered an unforeseen setback when our Electrical Engineering member became unresponsive and ultimately withdrew from the project. This left us without the crucial support and guidance needed for handling electrical components, complete PCB design, and other hardware configurations. We, the remaining members, rallied together to navigate the complexities of the electrical aspects. Despite the absence of dedicated expertise, we invested significant time and effort in exploring alternative solutions independently. While these solutions may not be fully optimized, and there are sacrifices in form factor and potential performance, our team's determination and collaborative spirit have propelled us forward, demonstrating the strength of our interdisciplinary collaboration and problem-solving capabilities.

During the research and testing of this project, we have learned a lot about our components, the process, and what it took to design each part of this project this semester to create a smart pair of glasses. We understand our strengths and the challenging parts, which help us figure out how to tackle each section. We learned to settle and prioritize goals when we switched from the ESP32S3 Sense. Although we sacrificed weight, price, and size, the switch helped us achieve one of our main goals. If we were to do this again, we’d look at MCUs/SBCs more capable than the ESP32S3 Sense but less powerful than the Raspberry Pi 5, as it was overkill. A custom PCB would be the optimal solution

as it could be designed to fit the form factor of glasses. A couple of stretch goals we didn't achieve were making the Speech to text completely offline and making a Bluetooth app for completely offline functionality. These are some features a future group could work on.

I.R.A.S. is designed to be the simplest AR wearable device currently on the market since the current products stated earlier have many complicated features attached. This prototype aims to focus on enhancing the user's social interactions regardless of their surrounding environment. With the initial concept approved by the advisors of CREOL and the design finalized with the PSE and CS students, the next phase was the creation of our project. We planned to approach the first phase of Senior Design 2 by gathering our listed parts from our initial design and testing multiple iterations of the optics designs, like the different types of optical combiners for the near-eye display. As the optical designs were being tested, the CS members worked on the software of the ESP32 and Raspberry Pi boards to unlock the full library of customizable capabilities. Once the optical designs and software was finalized, the next phase involved having to combine the hardware of both the optics and the MCUs with each other and creating an overall framework of the components.

Extensive testing across diverse platforms ensures compatibility, while user testing, including sessions with individuals with color blindness and hearing impairment or need of language translation, refine usability for all. Accessibility features were expanded to support multiple color blindness treatments, and language detection and translation were further developed for inclusivity. The finalization phase encompassed optimization checks, comprehensive documentation, and preparation for Senior Design 2's showcase event, emphasizing key features and the impactful outcomes of Project I.R.A.S.

As we conclude this project, our team feels like it's essential to reflect on the envisioned impact of I.R.A.S. and its alignment with the vision of creating inclusive technology. The project aligns with the vision of creating inclusive technology by addressing specific challenges faced by individuals with sensory impairments. By focusing on simplicity, practicality, and user-centric design. I.R.A.S. contributes to the ongoing effort to make technology more accessible to diverse populations. The commitment to simplicity distinguishes I.R.A.S. from existing complex AR wearable devices, ensuring that the technology is approachable and user-friendly. As technology continues to evolve, we hope that the success of I.R.A.S. will set a precedent for any future endeavors, inspiring the integration of inclusive features in technological innovations whether they are ours or of any watchers of this project. We want to underscore the belief that technology should be a tool for empowerment and social integration, breaking barriers and fostering a more inclusive and connected world.



## 11.0 Appendices

### 11.1 References

- [1] Abdulrahman Agha, Waqas Waheed, Nahla Alamoodi, Bobby Mathew, Fadi Alnaimat, Eiyad Abu-Nada, Aissa Abderrahmane, Anas Alazzam. "A Review of Cyclic Olefin Copolymer Applications in Microfluidics and Microdevices." *Macromolecular Materials and Engineering* 307.8 (2022). <<https://onlinelibrary.wiley.com/doi/10.1002/mame.202200053>>.
- [2] Areen A. Bani-Salameh, A. M. Alsaad, A. A. Ahmad, I. A. Qattan, and Ibsan A. Aljarrah. "Synthesis, Optical, Chemical and Thermal Characterizations of PMMA-PS/CeO<sub>2</sub> Nanoparticles Thin Film." *MDPI* 13.7 (2021). 2023. <<https://www.mdpi.com/2073-4360/13/7/1158#:~:text=The%20transmittance%20T%25%20%28%CE%BB%29%20and%20reflectance%20R%25%20%28%CE%BB%29,room%20temperature%20in%20the%20spectral%20range%20%28250%20%28%93700%29%20nm.>>>.
- [3] Azurdy, Jorge. *Virtual and Augmented Reality: Pros and Cons*. Ed. Flavia Negrao. 05 06 2020. 20 10 2023. <<https://www.encora.com/insights/virtual-and-augmented-reality-pros-and-cons#:~:text=Augmented%20Reality%201%20Pros%3A%20AR%2C%20when%20used%20properly%2C,effectively%20for%20exhibitions%20and%20events.%20...%20More%20items>>>.
- [4] Cromie, Greg. *What is Chromatic Aberration (and How to Correct It)?* 15 08 2020. 22 10 2023. <<https://shotkit.com/chromatic-aberration/>>.
- [5] Eisenberg, Eric. *A More Accurate Measure of MTF to Ensure Quality of AR/VR/MR Devices*. 07 06 2021. 10 23 2023. <<https://www.radiantvisionsystems.com/blog/more-accurate-measure-mtf-ensure-quality-ar/vr/mr-devices#:~:text=MTF%20is%20the%20primary%20method%20used%20to%20measure,the%20intersection%20of%20bright%20and%20dark%20image%20areas.>>>.
- [6] *Exploring the Potential of LCoS Microdisplays*. 14 10 2021. Radiant Vision Systems. 29 9 2023. <<https://www.azom.com/article.aspx?ArticleID=20844#:~:text=LCoS%20microdisplays%2C%20such%20as%20those%20present%20in%20AR%2FVR,%28pixelated%29%20surface%20and%20one%20transparent%20thin-film%20transistor%20%28TFT%29.>>>.
- [7] Gregory Hollows, Nicholas James. *System Throughput, f/#, and Numerical Aperture*. n.d. 23 10 2023.

<<https://www.edmundoptics.com/knowledge-center/application-notes/imaging/lens-iris-aperture-setting/>>.

[8] —. The Modulation Transfer Function (MTF). n.d. 23 10 2023.

<<https://www.edmundoptics.com/knowledge-center/application-notes/imaging/modulation-transfer-function-mtf-and-mtf-curves/>>.

[9] Gutttag, Karl. Near-Eye Bird Bath Optics Pros and Cons - And IMMY's Different Approach. 03 03 2017. 2023.

<<https://kgutttag.com/2017/03/03/near-eye-bird-bath-optics-pros-and-cons-and-immys-different-approach/>>.

[10] Jianghao Xiong, En-Lin Hsiang, Zigian He, Tao Zhan, & Shin-Tson Wu. "Augmented reality and virtual reality displays: emerging technologies and future perspectives." Nature Light Science Applications 10.216 (2021). <all

[11] Kallenbach, David. Near-to-eye applications with OLED microdisplays. 22 02 2022. 2023.

<<https://www.framos.com/en/articles/near-to-eye-applications-with-oled-microdisplays#:~:text=With%20a%20minimal%20pixel%20pitch%20of%206.3%20%C2%B5m%2C,has%20a%20high%20pixel%20density%20of%204%2C032%20ppi.>>.

[12] Kim, Jaehyeok. LetinAR. 04 10 2016. 23 10 2023.

<<https://letinAR.com/en/pintilt>>.

[13] Making a Magnifying Glass from Start to Finish With Acrylic Plastic. 03 03 2016. 18 10 2023.

<<https://www.creativemechanisms.com/blog/cnc-3d-print-acrylic-plastic-magnifying-glass>>.

[14] Marieke Burghoorn, Dorrit Roosen-Melsen, Joris de Riet, Sami Sabik, Zeger Vroon, Iryna Yakimets, and Pascal Buskens. "Single Layer Broadband Anti-Reflective Coatings for Plastic Substrates Produced by Full Wafer and Roll-to-Roll Step-and-Flash Nano-Imprint Lithography." National Library of Medicine 6.9 (2013): 3710-3726.

<<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5452668/>>.

[15] McMillan, Malcolm. What is Micro OLED? This might be the future of display technology. 16 06 2023. 2023.

<<https://www.tomsguide.com/features/micro-oled#:~:text=Each%20Micro%20OLED%20silicon%20wafer%20needs%20to%20be,30%2C000%20to%20be%20used%20for%20Micro%20OLED%20displays.>>.

[16] Optivent. Waveguide Combiners for AR Glasses. n.d. 2023.

<<http://www.optivent.com/ar-waveguide-combiners-reflective-diffractive-holographic-glass-and-polymer/#:~:text=Diffraction%20of%20the%20image%20rays%20>

is%20performed%20by,the%20eye%20with%20another%20set%20of%20slanted%20gratings.>.

[17] Park, Soon-gi. "Augmented and Mixed Reality Optical See-Through Combiners Based on Plastic Optics." *Society for Information Display* 37.4 (2021): 6-11. <<https://sid.onlinelibrary.wiley.com/doi/full/10.1002/msid.1226>>.

[18] SpecialChem. *Comprehensive List of Transparent Polymers*. 25 10 2017. 2023. <<https://omnexus.specialchem.com/tech-library/article/comprehensive-list-of-transparent-polymers>>.

[19] Takahashi, Dean. *Mira unveils lightweight augmented reality glasses for \$100*. 18 07 2017. 2023. <<https://venturebeat.com/business/mira-unveils-lightweight-augmented-reality-glasses-for-100/>>.

[20] Truly, Alan. *Their new smart glasses put a 500-nit, 330-inch screen on your eyes*. 24 10 2023. 25 10 2023. <<https://www.digitaltrends.com/computing/xreal-air-2-and-air-2-pro-are-better-lighter/>>.

[21] Wu, Tao Zhan and Kun Yin and Jianghao Xiong and Ziqian He and Shin-Tson. "Augmented Reality and Virtual Reality Displays: Perspectives and Challenges." *iScience* 23.8 (2020): 101397. 08 10 2023. <<https://www.sciencedirect.com/science/article/pii/S258900422030585X>>.

[22] Xia Xinxing, Guan Frank Yunqing, Cai Yiyu, Magnenat Thalmann Nadia. "Challenges and Advancements for AR Optical See-Through Near-Eye Displays: A Review." *Frontiers in Virtual Reality* 3 (2022). 10 10 2022. <<https://www.frontiersin.org/articles/10.3389/frvir.2022.838237>>.

[23] Xometry, Team. *All About Polycarbonate (PC)*. 07 05 2022. 19 10 2023. <<https://www.xometry.com/resources/materials/polycarbonate/>>.

[24] *Everything You Need to Know About Acrylic PMMA*. 04 05 2022. 10 19 2023. <<https://www.xometry.com/resources/materials/acrylic-pmma/>>.

[25] Yuge Huang, En-Lin Hsiang, Ming-Yang Deng & Shin-Tson Wu. "Mini-LED, Micro-LED and OLED displays: present status and future perspectives." *Light: Science & Applications* 9.105 (2020). <<https://www.nature.com/articles/s41377-020-0341-9>>.

[26] Yun-Han LEE, Tao ZHAN, Shin-Tson WU. "Prospects and challenges in augmented reality displays." *Virtual Reality & Intelligent Hardware* 1.1 (2019): 10-20. <<https://www.sciencedirect.com/science/article/pii/S2096579619300038>>.

- [27] Zhibin, Weng. "Transparent backlight panel and process, transparent LCD screen, electronic equipment, and VR/AR equipment." (n.d.).
- [28] Eric Eisenberg, Jens Jensen, "Measuring and qualifying optical performance of AR/VR/MR device displays and addressing the unique visual requirements of transparent AR/MR displays," Proc. SPIE 11310, Optical Architectures for Displays and Sensing in Augmented, Virtual, and Mixed Reality (AR, VR, MR), 113100S (19 February 2020). < <https://doi.org/10.1117/12.2546613>>.
- [29] Valter Drazic, Oksana Shramkova, Bobin Varghese, Laurent Blondé, Vincent Brac De La Perrière, Jesse Schiffler, Patrice Twardowski, Sylvain Lecler, Benjamin Walter, Estelle Mairiaux, Fuanki Bavedila, Marc Faucher, and Valérie Allié, "Over-wavelength pitch sized diffraction gratings for augmented reality applications," Opt. Express 30, 1293-1303 (2022)
- [30] Poetker, Bridget. "A Brief History of Augmented Reality (+ Future Trends & Impact)." G2, 9 August 2023, <https://www.g2.com/articles/history-of-augmented-reality>. Accessed 13 September 2023
- [31] Spector RH. Visual Fields. In: Walker HK, Hall WD, Hurst JW, editors. Clinical Methods: The History, Physical, and Laboratory Examinations. 3rd ed. Boston: Butterworths; 1990. Chapter 116. PMID: 21250064.
- [32] Dunnington, Waldo. "The Sesquicentennial of the Birth of Gauss." WayBackMachine, 12 June 2000, [web.archive.org/web/20080226020629/http://www.mathsong.com/cfgauss/Dunnington/1927/](http://www.mathsong.com/cfgauss/Dunnington/1927/).
- [33] Yobani Mejía. "Fundamentals of Optics: An Introductory Course." SPIE EBooks, 5 Jan. 2023, <https://doi.org/10.1117/3.2660873>.
- [34] Sarkar, Mukul, and Albert Theuwissen. A Biologically Inspired CMOS Image Sensor. Studies in Computational Intelligence, Springer Berlin, Heidelberg, 1 Jan. 2013.
- [35] Aissati, Sara, et al. Discovering Light: Fun Experiments with Optics. Bellingham, Washington (1000 20th St. Bellingham WA 98225-6705 USA), SPIE, 2021.
- [36] González-Acuña, Rafael G, and Héctor A Chaparro-Romo. Stigmatic Optics. IOP Science, 2020.
- [37] Pixy2 - [https://media.digikey.com/pdf/Data%20Sheets/Seed%20Technology/102991074\\_Web.pdf](https://media.digikey.com/pdf/Data%20Sheets/Seed%20Technology/102991074_Web.pdf)

- [38] Mega 5MP SPI Camera Module with Autofocus Lens for Any Microcontroller -  
<https://www.arducam.com/product/presale-mega-5mp-color-rolling-shutter-camera-module-with-autofocus-lens-for-any-microcontroller/>
- [39] Arducam 5MP OV5647 Autofocus Camera -  
<https://www.arducam.com/product/arducam-pi-camera-for-octoprint-octopi-webcam-with-motorized-lens-adjustable-focal-length-for-monitoring-3d-printer-3-28ft-100cm-long-extension-flex-ribbon-cable-for-raspberry-pi/>
- [40] Mini 2MP Plus – OV2640 SPI Camera Module -  
<https://www.arducam.com/product/arducam-2mp-spi-camera-b0067-arduino/>
- [41] 2MP IMX462 Color Ultra Low Light STARVIS Camera Module -  
<https://www.arducam.com/product/2mp-imx462-color-ultra-low-light-starvis-camera-module-with-141h-wide-angle-m12-lens-for-raspberry-pi/>
- [42] 2 Megapixels MT9D111 Auto Focus Lens Camera -  
<https://www.uctronics.com/2megapixel-mt9d111-auto-focus-lens-camera-flex-module-with-adapter-board-p-2168l.html>
- [43] 8MP IMX219 Auto Focus Camera Module -  
<https://www.arducam.com/product/arducam-for-raspberry-pi-camera-8mp-imx219-auto-focus-camera-module-b0393/>
- [44] IMX519 PDAF&CDAF Autofocus Camera Module -  
<https://www.arducam.com/product/imx519-autofocus-camera-module-for-raspberry-pi-arducam-b0371/>
- [45] 5MP OV5647 Camera Module with M12 Lens -  
<https://www.arducam.com/product/arducam-5mp-m12-picam/>
- [46] 5MP OV5647 Mini Camera Module -  
<https://www.arducam.com/product/b0033b0087-arducam-5mp-ov5647-mini-camera-video-module-with-15-pin-1-0mm-pitch-to-22-pin-0-5mm-and-15pin-to-15pin-1-0mm-ribbon-cable-for-raspberry-pi-model-a-b-b-pi-2-pi-3-pi-4-and-pi-zero/>
- [47] 5MP MIPI Camera for RZBoard V2L with Renesas RZ/V2L Processor -  
<https://www.arducam.com/product/arducam-5mp-mipi-camera-for-rzboard-v2l-with-renesas-rz-v2l-processor/>
- [48] Pcam 5C -  
[https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/2326/410-358\\_Web.pdf](https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/2326/410-358_Web.pdf)

- [49] OV5640 Camera Board C -  
<https://www.waveshare.com/ov5640-camera-board-c.htm>
- [50] Vision AI Module Grove -  
<https://www.seeedstudio.com/Grove-Vision-AI-Module-p-5457.html>
- [51] IMX219-AF Programmable/Auto Focus Camera Module -  
<https://www.arducam.com/product/b0181-arducam-imx219-af-programmable-auto-focus-camera-module-nvidia-jetson-nano/>
- [52] 477P High Quality Camera -  
<https://www.arducam.com/product/b0240-arducam-imx477-hq-quality-camera/>
- [53] 12MP IMX708 Wide Angle Camera Module -  
<https://www.arducam.com/product/arducam-12mp-imx708-wide-angle-camera-module-for-nvidia-jetson-nano-xavier-nx-orin-nx-orin-nano-agx-orin/>
- [54] 16MP IMX298 Color Camera Module -  
<https://www.arducam.com/product/arducam-pivariety-16mp-imx298-color-camera-module-for-rpi-4b-3b-2b-3a-pi-zero-cm3-cm4/>
- [55] 64MP Ultra High-Resolution Autofocus Camera Module -  
<https://www.arducam.com/product/64mp-af-for-raspberry-pi/>
- [56] Realtek AMB82-Mini IoT AI Camera -  
[https://www.seeedstudio.com/AMB82-MINI-RTL8735B-IoT-AI-Camera-Dev-Board-p-5584.html?queryID=8f05587237edfad01810ff569ff0a455&objectID=5584&indexName=bazaar\\_retailer\\_products\\_sales\\_qty\\_desc](https://www.seeedstudio.com/AMB82-MINI-RTL8735B-IoT-AI-Camera-Dev-Board-p-5584.html?queryID=8f05587237edfad01810ff569ff0a455&objectID=5584&indexName=bazaar_retailer_products_sales_qty_desc)
- [57] MU VISION SENSOR 3 -  
[https://www.seeedstudio.com/MU-VISION-SENSOR-3-AI-Robot-Vision-Camera-Supported-by-Arduino-Micro-Bit-p-4449.html?queryID=68510e18d6cfc8fd0a683fc8fca45b4c&objectID=4449&indexName=bazaar\\_retailer\\_products\\_sales\\_qty\\_desc](https://www.seeedstudio.com/MU-VISION-SENSOR-3-AI-Robot-Vision-Camera-Supported-by-Arduino-Micro-Bit-p-4449.html?queryID=68510e18d6cfc8fd0a683fc8fca45b4c&objectID=4449&indexName=bazaar_retailer_products_sales_qty_desc)
- [58] Seeed Studio XIAO ESP32S3 Sense -  
[https://www.seeedstudio.com/XIAO-ESP32S3-Sense-p-5639.html?queryID=31b62cafe596fafb1c0b66088248feab&objectID=5639&indexName=bazaar\\_retailer\\_products\\_sales\\_qty\\_desc](https://www.seeedstudio.com/XIAO-ESP32S3-Sense-p-5639.html?queryID=31b62cafe596fafb1c0b66088248feab&objectID=5639&indexName=bazaar_retailer_products_sales_qty_desc)
- [59] Pixycam. "Color Connected Components." Pixy Documentation, Pixy, 2018, docs.pixycam.com/wiki/doku.php?id=wiki:v2:color\_connected\_components.
- [60] ASTM International. "Guide for Setting Object Color Specifications." D7195 – 21, 23 June 2021, <https://doi.org/10.1520/d7195-21>.

- [61] International, ASTM. "Terminology for Three-Dimensional (3D) Imaging Systems." E2544 – 11a (Reapproved 2019), 3 Mar. 2022, <https://doi.org/10.1520/e2544-11ar19e01>.
- [62] International Organization for Standardization. "- Graphic technology and photography - Colour characterization of digital still cameras (DSCs) - Part 1: Stimuli, metrology, and test procedures." ISO/TS 17321-1:2012, Nov. 2022.
- [63] International Organization for Standardization. "Photography — Electronic still picture imaging terminology — Part 2: Other defined terms." ISO/TS 17321-2:2012, Oct. 2012.
- [64] International Organization for Standardization. "Graphic technology and photography — Colour characterization of digital still cameras (DSCs) — Part 3: User controls and readouts for scene-referred imaging applications." ISO/TS 17321-3:2017, Apr. 2017.
- [65] International Organization for Standardization. "Graphic technology and photography — Colour characterization of digital still cameras (DSCs) — Part 4: Programmable light emission system." ISO/TS 17321-4:2022, Nov. 2022.
- [66] International Organization for Standardization. "Photography — Electronic still picture imaging — Resolution and spatial frequency responses." ISO 12233:2023, Feb. 2023.
- [67] "ESP32 Sound - Working with I2S." *YouTube*, DroneBot Workshop, 22 May 2022, <https://youtu.be/m-MPBjScNRk?feature=shared>.
- [68] "#419 ESP32 Audio Tutorial with Lots of Examples." *YouTube*, Andreas Spiess, 6 Mar. 2022, <https://www.youtube.com/watch?v=a936wNgtcRA>.
- [69] Tatum, Malcolm. "What Is a Data Buffer?" *Easy Tech Junkie*, Easy Tech Junkie, 23 Sept. 2023, [www.easytechjunkie.com/what-is-a-data-buffer.htm](http://www.easytechjunkie.com/what-is-a-data-buffer.htm).
- [70] Jena, Satyabrata. "Difference between Batch Processing and Stream Processing." *GeeksforGeeks*, GeeksforGeeks, 5 May 2023, [www.geeksforgeeks.org/difference-between-batch-processing-and-stream-processing/](http://www.geeksforgeeks.org/difference-between-batch-processing-and-stream-processing/).
- [71] "How the Web Works, HTTP Request/Response Cycle." *Turing*, Turing, [backend.turing.edu/module2/lessons/how\\_the\\_web\\_works\\_http](http://backend.turing.edu/module2/lessons/how_the_web_works_http).
- [72] "Why Is HTTP Not Secure? | HTTP vs. HTTPS." *CloudFlare*, CloudFlare, [www.cloudflare.com/learning/ssl/why-is-http-not-secure/](http://www.cloudflare.com/learning/ssl/why-is-http-not-secure/).

- [73] Santos, Sara. "ESP32 Https Requests (Arduino IDE)." *Random Nerd Tutorials*, Random Nerd Tutorials, 23 Mar. 2023, [randomnerdtutorials.com/esp32-https-requests/](https://randomnerdtutorials.com/esp32-https-requests/).
- [74] "Introduction to Grpc." *gRPC*, gRPC, 15 Feb. 2023, [grpc.io/docs/what-is-grpc/introduction/](https://grpc.io/docs/what-is-grpc/introduction/).
- [75] "Raspberry Pi Pico - SSD1306 OLED Micro Python Library and Setup." *YouTube*, Novaspirit Tech, 23 Feb. 2021, <https://www.youtube.com/watch?v=YR9v04qzJ5E>.
- [76] "Add an OLED Stats Display to Raspberry Pi OS Bullseye." *YouTube*, Michael Klements, 15 Nov. 2021, <https://www.youtube.com/watch?v=IRTQ0NsXMuw>.
- [77] Çakar, Debi. "Speech Recognition Accuracy Comparison Test 2023." *SESTEK*, SESTEK, 27 Mar. 2023, [www.sestek.com/speech-recognition-accuracy-comparison-test-2023-blog](http://www.sestek.com/speech-recognition-accuracy-comparison-test-2023-blog).
- [78] Raspberrypi.com, 2023, [downloads.raspberrypi.com/raspios\\_armhf/release\\_notes.txt](https://downloads.raspberrypi.com/raspios_armhf/release_notes.txt).
- [79] "Raspberry Pi: 10 Operating Systems for the Minicomputer." *IONOS Digital Guide*, 1 Mar. 2023, [www.ionos.com/digitalguide/server/know-how/the-best-raspberry-pi-operating-systems-a-brief-portrait/](http://www.ionos.com/digitalguide/server/know-how/the-best-raspberry-pi-operating-systems-a-brief-portrait/).
- [80] FreeRTOS. "FreeRTOS - Market Leading RTOS (Real Time Operating System) for Embedded Systems with Internet of Things Extensions." FreeRTOS, [www.freertos.org/index.html](http://www.freertos.org/index.html).
- [81] "Mongoose OS - Features." *Mongoose-Os.com*, [mongoose-os.com/features.html](http://mongoose-os.com/features.html). Accessed 3 Nov. 2023.
- [82] "Get Started - ESP32 - — ESP-IDF Programming Guide Latest Documentation." *Docs.espressif.com*, [docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html).
- [83] Söderby , Karl , and Jacob Hylén. "Getting Started with Arduino IDE 2.0 | Arduino Documentation." *Docs.arduino.cc*, [docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2](https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2).
- [84] Söderby, Karl. "Getting Started with the Web Editor | Arduino Documentation." *Docs.arduino.cc*, [docs.arduino.cc/arduino-cloud/getting-started/getting-started-web-editor](https://docs.arduino.cc/arduino-cloud/getting-started/getting-started-web-editor).



[85] “STM32CubeIDE.” STMicroelectronics, [www.st.com/en/development-tools/stm32cubeide.html](http://www.st.com/en/development-tools/stm32cubeide.html).

[86] Nvidia. “JetPack SDK.” NVIDIA Developer, [developer.nvidia.com/embedded/jetpack](http://developer.nvidia.com/embedded/jetpack).

[87] Apple. “Augmented Reality - Apple Developer.” Apple Developer, 2019, [developer.apple.com/augmented-reality/](http://developer.apple.com/augmented-reality/).

[88] ARCore. “Choose Your Development Environment | ARCore.” Google Developers, [developers.google.com/ar/develop](http://developers.google.com/ar/develop).

[89] Unity Technologies. “Augmented Reality | Unity.” Unity, 2019, [unity.com/unity/features/ar](http://unity.com/unity/features/ar).

[90] Vuforia. “Getting Started | VuforiaLibrary.” Library.vuforia.com, [library.vuforia.com/](http://library.vuforia.com/).

[91] Lakshmanamoorthy, Rajkumar. “Real-Time GUI Interactions with OpenCV in Python.” Analytics India Magazine, 17 Apr. 2021, [analyticsindiamag.com/real-time-gui-interactions-with-opencv-in-python/](http://analyticsindiamag.com/real-time-gui-interactions-with-opencv-in-python/).

[92] Zvornicanin, Enes. “What Is YOLO Algorithm? | Baeldung on Computer Science.” Www.baeldung.com, 11 June 2022, [www.baeldung.com/cs/yolo-algorithm#:~:text=YOLO%20algorithm%20aims%20to%20predict](http://www.baeldung.com/cs/yolo-algorithm#:~:text=YOLO%20algorithm%20aims%20to%20predict).

[93] Nanos, Georgios. “Fast R-CNN: What Is the Purpose of the ROI Layers? | Baeldung on Computer Science.” Www.baeldung.com, 28 Oct. 2022, [www.baeldung.com/cs/fast-r-cnn-roi-layers](http://www.baeldung.com/cs/fast-r-cnn-roi-layers).

[94] Nanos, Georgios. “Object Detection: SSD vs. YOLO | Baeldung on Computer Science.” Www.baeldung.com, 29 Sept. 2022, [www.baeldung.com/cs/object-detection-ssd-yolo](http://www.baeldung.com/cs/object-detection-ssd-yolo).

[95] OpenCV. “OpenCV: Image Processing (Imgproc Module).” Docs.opencv.org, [docs.opencv.org/3.4/d7/da8/tutorial\\_table\\_of\\_content\\_imgproc.html](http://docs.opencv.org/3.4/d7/da8/tutorial_table_of_content_imgproc.html).

[96] ArpitAsati. “What Is Web Socket and How It Is Different from the HTTP?” GeeksforGeeks, 4 Dec. 2019, [www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/](http://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/)

[97] MQTT. “MQTT - the Standard for IoT Messaging.” Mqtt.org, 2022, [mqtt.org/](http://mqtt.org/).

[98] IBM. “What Is a REST API? | IBM.” Www.ibm.com, [www.ibm.com/topics/rest-apis#:~:text=the%20next%20step-](http://www.ibm.com/topics/rest-apis#:~:text=the%20next%20step-)

- [99] SUDIPTADANDAPAT. “Real Time Transport Protocol (RTP).” GeeksforGeeks, 18 May 2020, [www.geeksforgeeks.org/real-time-transport-protocol-rtp/](http://www.geeksforgeeks.org/real-time-transport-protocol-rtp/).
- [100] International Organization for Standardization. “Standardization of Image Quality Analysis.” ISO 19264, Jun. 2021.
- [101] “ISO/IEC 25010:2011.” *International Organization for Standardization*, 29 Mar. 2019, [www.iso.org/standard/35733.html](http://www.iso.org/standard/35733.html).
- [102] “ISO/IEC 25062:2006.” ISO, International Organization for Standardization, 1 Oct. 2006, [www.iso.org/standard/43046.html](http://www.iso.org/standard/43046.html).
- [103] “ISO/IEC/IEEE 12207:2017.” ISO, International Organization for Standardization, 11 Oct. 2023, [www.iso.org/standard/63712.html](http://www.iso.org/standard/63712.html).
- [104] IEEE Standard for Software Quality Assurance Processes - IEEE XPLORE, [ieeexplore.ieee.org/document/6835311/definitions](http://ieeexplore.ieee.org/document/6835311/definitions). Accessed 3 Nov. 2023.
- [105] “ISO 9241-210:2019.” ISO, International Organization for Standardization, 3 Jan. 2019, [www.iso.org/standard/77520.html](http://www.iso.org/standard/77520.html).
- [106] Ruth, Corey. “The Evolution of Wi-Fi Technology and Standards.” *IEEE Standards Association*, 24 Aug. 2023, [standards.ieee.org/beyond-standards/the-evolution-of-wi-fi-technology-and-standards/#:~:text=IEEE%20802.11%E2%84%A2%20is%20the,for%20Wi%2DFi%20wireless%20networks.&text=Wi%2DFi%20TECHNOLOGY-,Wi%2DFi%20technology%20is%20based%20on%20the%20IEEE%20802.11%E2%84%A2,we%20communicate%20and%20access%20information](https://standards.ieee.org/beyond-standards/the-evolution-of-wi-fi-technology-and-standards/#:~:text=IEEE%20802.11%E2%84%A2%20is%20the,for%20Wi%2DFi%20wireless%20networks.&text=Wi%2DFi%20TECHNOLOGY-,Wi%2DFi%20technology%20is%20based%20on%20the%20IEEE%20802.11%E2%84%A2,we%20communicate%20and%20access%20information).
- [107] “What Is Agile and What Is Scrum?” Cprime, 17 Apr. 2023, [www.cprime.com/resources/what-is-agile-what-is-scrum/#:~:text=Scrum%20is%20a%20subset%20of,be%20consistent%20with%20the%20framework](http://www.cprime.com/resources/what-is-agile-what-is-scrum/#:~:text=Scrum%20is%20a%20subset%20of,be%20consistent%20with%20the%20framework).
- [108] “ISO/IEC 27001:2022.” ISO, International Organization for Standardization, 25 Oct. 2022, [www.iso.org/standard/27001](http://www.iso.org/standard/27001).
- [109] “Getting Started.” NIST, National Institute of Standards and Technology, 21 Apr. 2023, [www.nist.gov/cyberframework/getting-started](http://www.nist.gov/cyberframework/getting-started).
- [110] van Rossum, Guido. PEP 8 – Style Guide for Python Code, Python, 1 Aug. 2013, [peps.python.org/pep-0008/](http://peps.python.org/pep-0008/).
- [111] “ESP Modules: Espressif Systems.” *ESP Modules | Espressif Systems*, [www.espressif.com/en/products/modules](http://www.espressif.com/en/products/modules). Accessed 12 Nov. 2023.

- [112] “ESP32-S3-DEVKITC-1 v1.1.” *ESPRESSIF*, docs.espressif.com/projects/esp-idf/en/latest/esp32s3/hw-reference/esp32s3/use-r-guide-devkitc-1.html. Accessed 12 Nov. 2023.
- [113] “Jetson Nano Developer Kit.” *NVIDIA Developer*, developer.nvidia.com/embedded/jetson-nano-developer-kit. Accessed 12 Nov. 2023.
- [114] “Jetson Nano Developer Kit.” *NVIDIA Developer*, developer.nvidia.com/embedded/jetson-nano-developer-kit. Accessed 12 Nov. 2023.
- [115] Raspberry Pi. “Buy A Raspberry Pi Pico.” *Raspberry Pi*, www.raspberrypi.com/products/raspberry-pi-pico/. Accessed 12 Nov. 2023.
- [116] Raspberry Pi. “Buy A Raspberry Pi Zero 2 W.” *Raspberry Pi*, www.raspberrypi.com/products/raspberry-pi-zero-2-w/. Accessed 12 Nov. 2023.
- [117] “Arduino Nano 33 Ble Sense.” *Arduino Online Shop*, store-usa.arduino.cc/products/arduino-nano-33-ble-sense. Accessed 12 Nov. 2023.
- [118] Spierenburg, Peter-Frank, et al. “Seeed Studio Xiao ESP32S3 Sense - 2.4ghz Wi-Fi, Ble 5.0, OV2640 Camera Sensor, Digital Microphone, 8MB PSRAM, 8MB Flash, Battery Charge Supported, Rich Interface, IOT, Embedded ML.” *Seeed Studio XIAO ESP32S3 Sense - Seeed Studio*, 10 Oct. 2023, www.seeedstudio.com/XIAO-ESP32S3-Sense-p-5639.html?queryID=be6f53c7ae317f6cbbc28f0668fb03df&objectID=5639&indexName=bazaar\_retailer\_products.
- [119] Industries, Adafruit. “Adafruit I2S MEMS Microphone Breakout - SPH0645LM4H.” *Adafruit*, Adafruit, www.adafruit.com/product/3421.
- [120] Industries, Adafruit. “Electret Microphone Amplifier - Max4466 with Adjustable Gain.” *Adafruit*, Adafruit, www.adafruit.com/product/1063.
- [121] “Respeaker 2-Mics Pi Hat for Raspberry Pi - WM8960 Audio Codec, 2 Analog Microphones, 3 APA102 RGB Leds, 3.5mm Audio Jack, User Button, Attached with NLU Software Algorithms, Vad, DOA, KWS.” *ReSpeaker 2-Mics Pi HAT - Seeed Studio*, seeedstudio, www.seeedstudio.com/ReSpeaker-2-Mics-Pi-HAT.html.
- [122] Lvgl. (n.d.). *LVGL (Light and Versatile Graphics Library)*. GitHub. <https://github.com/lvgl/lvgl>
- [123] Elert, G. (n.d.). *Aberration. The Physics Hypertextbook*. <https://physics.info/aberration/>

- [124] Distortion | Edmund Optics. (n.d.). [Www.edmundoptics.com. https://www.edmundoptics.com/knowledge-center/application-notes/imaging/distortion/](https://www.edmundoptics.com/knowledge-center/application-notes/imaging/distortion/)
- [125] Arducam 1/2.5" M12 Mount Camera Lens M25360H06. (n.d.). Arducam. Retrieved November 28, 2023, from <https://www.arducam.com/product/m25360h06-2/>
- [126] *Optics application examples*. Edmund Optics. (n.d.). <https://www.edmundoptics.com/knowledge-center/application-notes/optics/optics-application-examples/>
- [127] Wolfe, G., Gasper, E., Stoke, J., Kretchman, J., Anderson, D., Czuba, N., Oberoi, S., Pujji, L., Lyublinskaya, I., & Ingram, D. (n.d.). *25.6 image formation by lenses - college physics for AP® courses*. OpenStax. <https://openstax.org/books/college-physics-ap-courses/pages/25-6-image-formation-by-lenses>
- [128] The Ultimate Guide to Lens Design Forms: The types of optical systems in a lens designer's toolbox. Ed. Kats Ikeda. n.d. 2023. <https://www.pencilofrays.com/lens-design-forms/>.
- [129] "Mini USB 2.0 Microphone MIC Audio Adapter Plug and Play for Raspberry Pi 5/4, Voice Recognition Software" Sunfounder, *Sunfounder*, <https://www.sunfounder.com/products/mini-usb-microphone>.

## **11.2 Copyright**