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

Carlos Acosta, Darlandie Moise, Brian Umbrechet, Kim Le

Department of Engineering and Computer Science and College of Optics and Photonics, University of Central Florida, Orlando, Florida, 32816-2450, U.S.A.

Abstract* — This paper outlines the hardware and software functions using various methods to design a pair of smart glasses that enables the user to view projected text and outlines, for both features of color detection and speech detection or translation, concurrently with their surroundings. Demonstrating the results and functions of the designs produced which are sectioned into the following categories: (1) Camera Lens Design; (2) AR Projection Lens Design; (3) Speech to Text Software Design; (4) Color Detection Software Design.

***Index Terms* — Augmented reality, image sensor, imaging lenses, LCD, microprocessor, optical reflection**

I. INTRODUCTION

The project was developed based on personal motivation and genuine passion to create a technology that would significantly benefit those with hearing impairment and color blindness. Project IRAS (Intricate Real-time Assistance Spectacles) is the next step for smart glasses. Our main overall project goal is to create a functional 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 follow outlines for the user who has trouble distinguishing specific colors around their environment. 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.

Each component has been researched for its specifications and compatibility with one another for incorporating them into our design. For the optics, we designed a lens system for the camera (more in the Camera Lens System Section) and the projection of the text from the LCD (more in the Lens Projection System Section). We purchased the best camera and LCD considering our budget and availability in the market. Once we decided which lenses we would design, we purchased the lenses after calculating the focal lengths and field of view that we required for our system specification. We then tested multiple variations of AR projections on lenses and wide-angle camera systems. During our initial testing, we slightly redesigned and calculated varying focal lengths, FOV angles, and depth of field for our proposed design to line up with our project's specifications.

The computer science majors have researched different operating systems, various speech-to-text application programming interfaces, and current speech-to-text services. They also researched ways for object tracking, color detection, capturing audio, and controlling the device with a mobile app. This helped them determine which systems to utilize and how they wanted to proceed with the project and design their code. They approached programming the glasses to display live speech-to-text and color detection with object tracking. Their first objectives were to display one language and set of colors to work, as well as customize how the information is displayed. Later objectives would include multi-lingual support, a wider range of color blindness detection, improved display, and utilization of existing options for mobile control of the device.

II. CAMERA LENS SYSTEM

A. Calculations for the Required Camera FOV

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 also how the image would display on the CMOS sensor [1][2].

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

$$\text{FOV} = 2tan^{-1}(\frac{h}{2f}) \qquad (1)$$

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 µm and f = 3.6 mm. This makes FOV about 54° which is roughly the 60° 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 objects 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 use the following equation:

$$M = \frac{h}{FOV} \quad (2)$$

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 µm and FOV = 0.541, equation the magnification M to be about x0.007. 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 ½ 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:

$$F/\# = \frac{f}{D} \quad (3)$$

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 stated as follows:

$$N.\ A. = \frac{1}{2 * F/\#} \quad (4)$$

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 ⅙, which is roughly N.A. = 0.17. This tells us about 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:

$$W.D. = \frac{f}{M} \quad (5)$$

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” = x0.007, 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:

$$\text{Horizontal FOV (mm)} = \text{Working Distance (mm)} * 2tan(\frac{FOV\ (degrees)}{2}) \quad (6)$$

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°, which equates our horizontal FOV to be about 524.1 mm. Taking that value, we can calculate the sensor resolution of our CMOS sensor which is capable of capturing good images with the following equation:

$$S.R. = 2(\frac{FOV\ (mm)}{Smallest\ Feature}) \quad (7)$$

The horizontal FOV of the lens system is 524.1 mm, and the smallest feature is 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:

$$I.C.D. = \sqrt{width^2 + height^2} \quad (8)$$

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 OV5647 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.

B. Camera Choice/Specifications

The 5MP OV5647 Autofocus Camera is essentially 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 special 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 ¼”, 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, as well as about 4.5 millimeters thick, giving it more flexibility to set up on our glasses.

C. Lens Design Simulation and Modeling

Due to the combined nature of the Ov5647 CMOS sensor and its additional features of having an already established 60° FOV format, a new lens system must be constructed to adhere to the optical design requirements. In that case, the next best approach is to establish a new lens system that combines the lenses of other M12s and creates a new about 80° FOV lens system. Table 1 lists all the components that are within the custom-made lens system, both the types of optical components used and the pieces that make up the overall lens system design. 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.

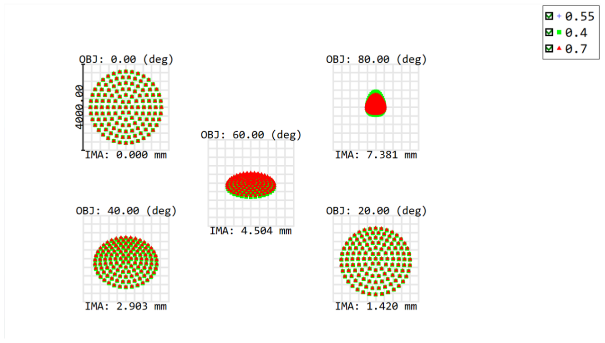

Figure 1: Spot Diagram of 80° FOV on Zemax with varying wavelengths (550 nm - 7 nm).

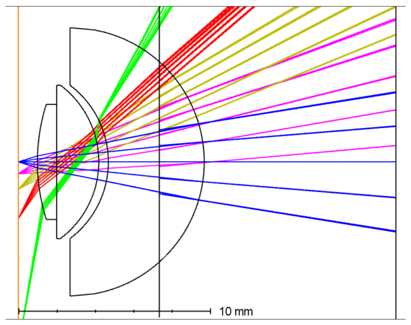

Figure 2: Cross Section of 80° FOV Lens System on Zemax with a Total Axial Length of 19.7 mm.

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 refuse to share details of the lenses within the lens system, making testing and measuring the lenses some human error. Regardless, this newly 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 6, the lens system provides a spot diagram that still keeps the colors from being very distorted. While chromatic aberration is still present, it is kept minimal towards the end of the spot diagram when viewing the 80° FOV.

Part	Specification	Purpose
Lens System (1.2)	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 with a height of about 5.8 mm	Houses the optical design of the camera
Lens Cap (1.4)	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
Plastic Spacer (1.3)	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
Metallic Spacer (2.3)	Diameter - 5 mm, Thickness - 1 mm	Keeps the smaller lens in place and separate from the larger lens, little distortion
Smaller Lens (2.2)	Diameter - 3 mm and 4 mm, Thickness - 2 mm, Focal Length - 10 mm	The entry lens for the optical design
Larger Lens (2.4)	Diameter - 10 mm, Thickness - 2 mm, Focal Length - -17 mm	The main lens providing the FOV of the lens system
Lens Mount (1.1)	Base - 18 mm x 18 mm Inner diameter - 14 mm Outer diameter - 16 mm	Holds camera and lens system together
Camera (2.1)	8.5 mm x 8.5 mm x 5.4 mm	The main source for visual input of our project

Table 1: This table lists all the separate pieces that encompass the Mutated Lens System.

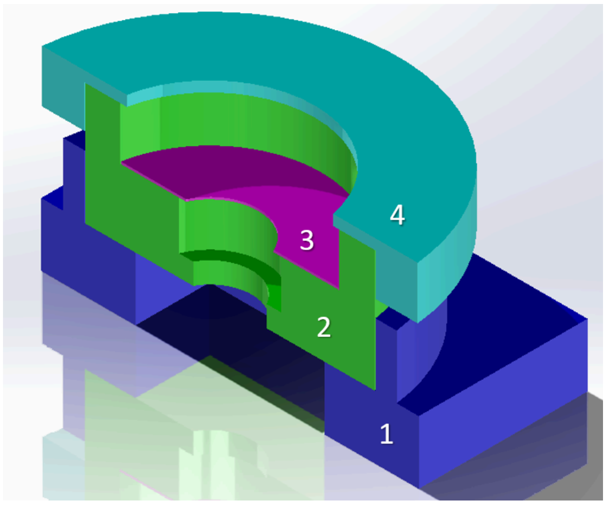

Figure 3.1: Cross-section of Mutation Lens System on SolidWorks Color Coded and Numbered: Lens Mount (1.1), Lens System (1.2), Spacer (1.3), and Lens Cap (1.4)

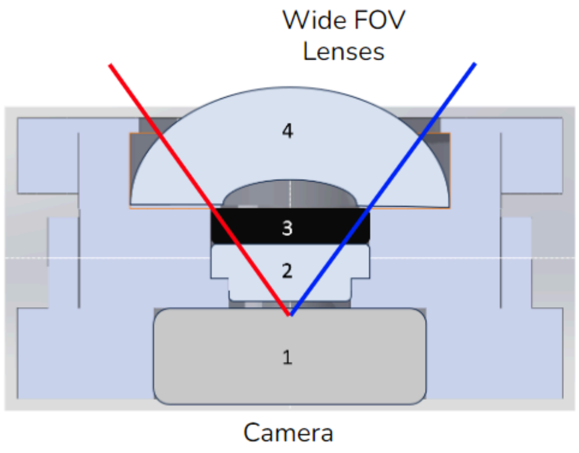

Figure. 3.2: Cross-section of the Mutation Lens System on SolidWorks with the Labeled Parts within the System. Camera (2.1), Smaller Lens (2.2), Metallic Spacer (2.3), and Larger Lens (2.4).

As mentioned before, the point of the titled "Mutation Lens System" is to widen the camera's FOV from its established 60° to about 80°, 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.

III. LENS PROJECTION SYSTEM

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

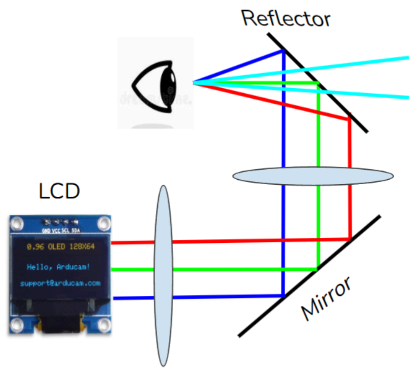

Figure 4: Lens Projection Design

The display used is the 1.69-inch color LCD TFT module with 240 x 280 resolution and brightness of 480 Cd/m² 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. 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 using the following equations:

$$\frac{1}{f} = \frac{1}{d_0} + \frac{1}{d_i} \tag{9}$$

$$M = -\frac{d_i}{d_o} \tag{10}$$

Where "f" is the focal length of the lens, and "d_0" is the distance of the object, which in this case would be considered our display being used. The distance between the lens and the image is represented by “d_i ”.

We utilized Eq. 9, the thin lens equation, and Eq. 10, the magnification equation, 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 of our lenses and mirrors 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 from 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 millimeters away at a 45° 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.

IV. HARDWARE DESIGN

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.

A. 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).

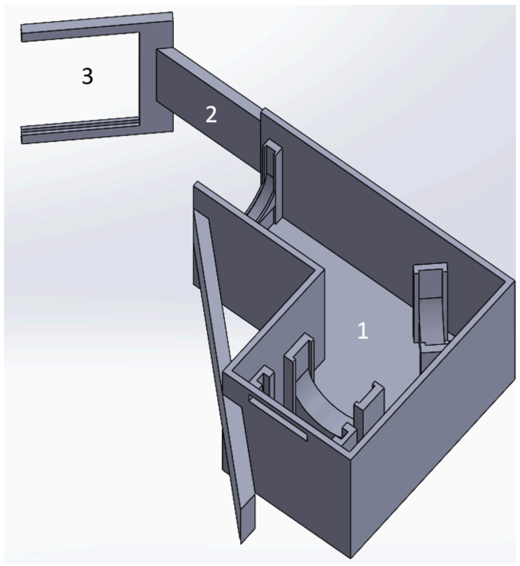

Figure 5: Solidworks LCD Housing that has 3 sub-parts, the Main Housing (1), the Support Beam for the Beamsplitter (2), and the Beamsplitter Holder (3).

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.

B. 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 6 and 7, 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.

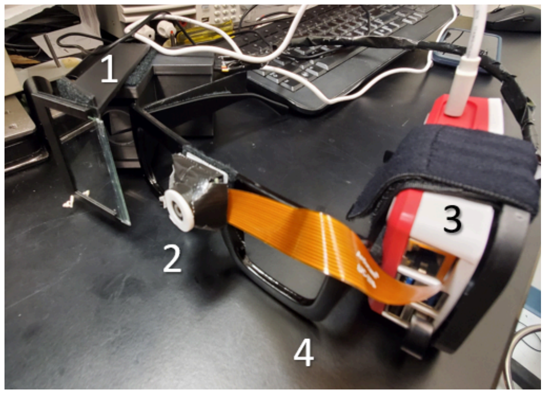

Figure 6: Project I.R.A.S. final design that can be divided into 4 parts, the LCD Projection Housing (1), the Mutated Lens System + Camera (2), the Raspberry Pi 5 (3), and the Main Glasses Frame (4).

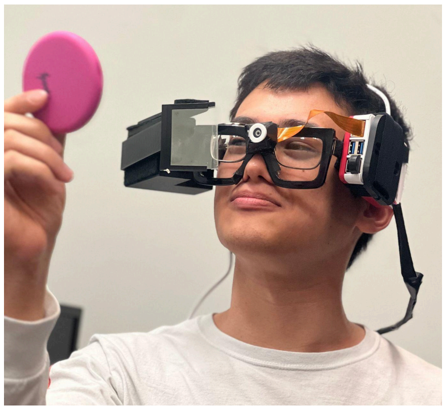

Figure 7: One of the colleagues testing the glasses' color detection values.

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.

V. SPEECH-TO-TEXT SOFTWARE

To implement the speech-to-text, two methods were considered; the utilization of a cloud-based service or an offline, and completely onboard solution. We decided accuracy was the most significant priority and settled on using Speechmatics [3]. Speechmatics is a speech AI company that develops speech recognition software. The key features that they provide which we were most interested in are real-time transcription and translation, high accuracy, a wide range of languages, and ease of implementation.

When implementing speech-to-text, we initially started with the Xiao ESP32S3 Sense. We started testing by seeing how to capture audio using the onboard microphone and streaming the input over WebSockets. WebSockets is an internet communication protocol like HTTP. It allows for constant, real-time bidirectional communication which is ideal for our goal [4]. Once we were comfortable with that, we created a NodeJS server that would receive the live WebSocket transmission. We chose to use NodeJS because Speechmatics didn't indicate how to use their service in C++ which is the language we were programming the ESP32 in, and JavaScript was easier for us to code with. Also, Speechmatics provided an example to follow in JavaScript. The server waits for a WebSocket client to connect, then upon receiving data from the client, it sends it to the Speechmatics servers, and then it'll receive a response in the form of a transcription or translation. This text will then be sent back to the ESP32 and displayed using proper formatting so that the text fits well on the screen and refreshes constantly.

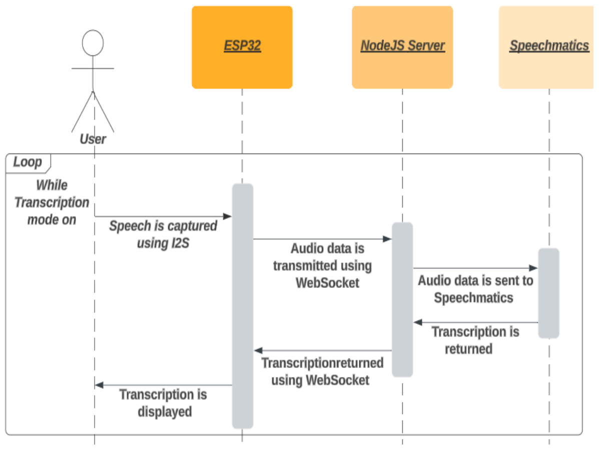
ESP32
NodeJS Server
Speechmatics
User
Loop
While Transcription mode on
Speech is captured using I2S
Audio data is transmitted using WebSocket
Audio data is sent to Speechmatics
Transcription is returned
Transcriptionreturned using WebSocket
Transcription is displayed

Figure 8: Sequence Diagram for audio data transfer and server interaction

Once we were able to utilize the server and transcribe speech, we focused on improving the system. Starting with the Wi-Fi, we had issues connecting to the campus Wi-Fi and were forced to use a cellular hotspot. This made the speech-to-text extremely slow and it skipped words frequently. We did more research and were able to connect to the UCF_WPA2 and eduroam successfully by using an ESP32 WPA2 library. The security of the campus networks was the issue. When testing, we found that connecting to a campus network was often inconsistent, but found that UCF_WPA2 worked the best. After the Wi-Fi, we implemented a ping/pong system for the WebSocket connections so that it could handle sudden disconnects from incidents like loss of power. Previously, the server had to be restarted every time we unpowered the ESP32 because the server didn't know that the WebSocket connection had ended. The final improvement we made to the server was hosting it on Amazon Web Service. We were able to run the server for a year at no price at all. We also made it so that the server ran in the background constantly so that we wouldn't need to have the terminal open.

When we implemented color detection on the ESP32, we found that it was extremely slow, so we decided to switch to the Raspberry Pi 5. This meant most of the progress we made previously had to be abandoned, but we were able to utilize our familiarity with the previous implementation to aid our future development. Initially, we weren't sure if we wanted to connect the ESP32 to the Raspberry Pi and transmit the audio data over an interface or just buy a microphone that connects directly to the Raspberry Pi. We decided that the latter was the simplest and went with that. Implementing Speechmatics on the Raspberry Pi was significantly easier than on the ESP32. This is because Speechmatics had an example of how to transcribe audio from a microphone in Python using their service, which is exactly what we needed. This also meant we did not need the NodeJS server which helps in reducing latency. Combining the speech-to-text program with the color detection program was the hard part. The speech-to-text program had to be run while in a virtual environment because it was dependent on the Speechmatics library which had to be installed using pip. This meant we couldn't just combine the two programs into one file. The main issue we had was with the GPIO pins indicating they were being used, even though we were clearing them. We were able to use print to the screen in the color detection program, but once it switched to the other, it consistently threw an error related to GPIO. We spent many hours exploring threads, imports, and subprocesses, to figure out how to connect both programs, and settled on using subprocesses. By using subprocesses, we could start the speech-to-text program using its virtual environment in the color detection program. This allowed us to have control over it using Blynk and send arguments over to indicate the translation mode. To solve the GPIO issue, instead of printing the text in the speech-to-text program, we send the text over stdin and listen to it in the color detection program. We also implemented formatting so that the text will appear cleanly. The final issue we encountered was connecting to the campus networks. Again we had inconsistencies connecting.

Some days it would work and others it wouldn't, but once it connected, the speech-to-text had no issue.

VI. COLOR DETECTION SOFTWARE

The color detection mode uses the OpenCV-Python library to recognize colors using a technique called color space conversion and thresholding to detect colors in images. Images are typically captured in RGB (Red, Green, Blue) format. However, detecting colors directly in RGB is not very efficient due to lighting variations and dependencies between color channels. OpenCV offers functions to convert the image from RGB to other color spaces like HSV (Hue, Saturation, Value). HSV is more intuitive for color detection because Hue represents the actual color itself (red, green, blue, etc.), Saturation represents the intensity of the color, and Value represents the brightness of the color. Once the image is in HSV format, we can define a range of Hue values that correspond to the color we want to detect. OpenCV provides functions like cv2.inRange to create a binary mask image where pixels within the specified range are set to white (1), and all other pixels are set to black (0). This mask image now highlights the pixels that fall within the color range we specified. For our application, we choose to overlay the bounding boxes, corresponding to the widths and heights of the masks, on the original frame to visualize the detected color region. This decision helps to reduce the overhead of drawing the contour outline for every frame while still being able to present the colors by having the color's name on its top edge. We're able to achieve an almost real-time reaction using these techniques on the latest Raspberry Pi 5, which is a huge improvement over our old MCU choice, the Xiao ESP32, which had a latency of around 2 seconds per image frame.

VII. THE MOBILE APP

To control the functionalities of the program without the need to install physical buttons, we decided to go with Blynk, a user-friendly IoT platform with mobile apps (iOS and Android) to control the Raspberry Pi. Blynk provides a cloud service that acts as a central hub for communication between the hardware and mobile app. It offers a visual programming interface where we can drag-and-drop widgets to build the UI for our app, and secure communication between the devices and the cloud using authentication tokens. These simplify the development process by a great deal and allow us to spend valuable time focusing on and refining the main program's code. We opted for the Maker plan which costs $7 a month and caters to up to 20 devices. It has enough widgets to set up our template with two main tabs, Color Detection mode and Speech Translation. In the Color Tab, there is an on/off button and 7 other buttons to choose the colors the users need to detect. Similarly, in the Speech tab, there is a button to turn on or off and several more to choose the languages they want to translate from and to. Currently, a user can join with his or her email and open the link within the Blynk app to get our project's template to control the glasses.

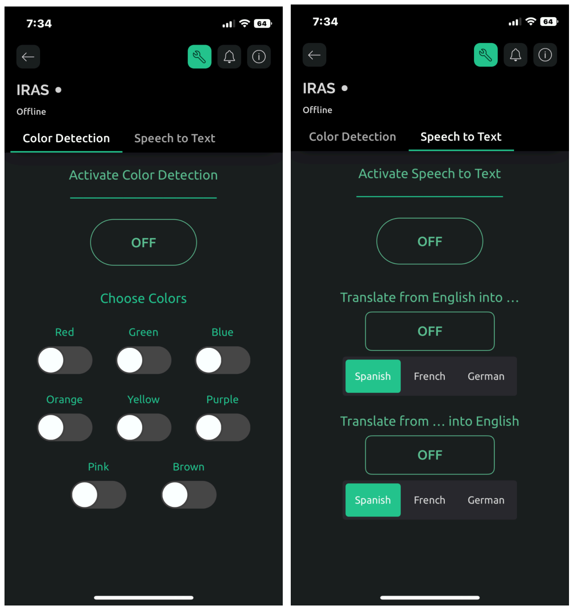

Figure 9: Our Blynk mobile app's user interface.

VIII. CONCLUSION

Project I.R.A.S has been created with all of these design implementations previously mentioned and their complexities to create a pair of wearable smart glasses that support speech dictation and translation to multiple languages, and color detection. Despite various challenges, such as electrical component complications and incompatibilities, we have been determined to explore alternative possibilities. Thus, our final prototype design results from our dedication and collaboration.

ACKNOWLEDGEMENT

The authors would like to acknowledge the assistance of Dr. LiKamWa, Dr. ChungYong Chan, and Dr. Aravinda Kar for their guidance throughout our project. We would also like to thank the College of CREOL and the College of Computer Science for giving us the necessary space, guidance, and resources to help us develop this project. We also extend our gratitude to the professors who have agreed to review our project.

BIOGRAPHY

Carlos Acosta

Graduating from CREOL with a B.S. in Optics and Photonics, Carlos Acosta has his eyes set on taking it to the next level, pursuing a PhD. at the prestigious Boston University. He plans to continue pursuing his research in Biophotonics and take his interest to better the lives of those with the advancement of medical optics.

Brian Umbrechet

Graduating from UCF with a bachelor's in Computer Science, Brian enjoys working on tough projects and learning new ideas. He plans to explore software engineering after graduation and work his up the ladder.

Darlandie Moise

Graduating from the University of Central Florida and receiving her B.S. in Photonic Science and Engineering. She currently works as an R&D Optical Engineer Intern at Everix. She plans on continuing her work at this company as an Optical Design Engineer upon graduation.

Kim Le

Graduating with a Bachelor's in Computer Science from UCF, Kim is fascinated by the intersection of technology and interactive experiences. She is fueled by caffeine, a passion for problem-solving, and a drive to deliver the best result with her capability.

REFERENCES

[1] Yobani Mejía. "Fundamentals of Optics: An Introductory Course." SPIE EBooks, 5 Jan. 2023, https://doi.org/10.1117/3.2660873.

[2] Sarkar, Mukul, and Albert Theuwissen. A Biologically Inspired CMOS Image Sensor. Studies in Computational Intelligence, Springer Berlin, Heidelberg, 1 Jan. 2013.

[3] Çakar, Debi. "Speech Recognition Accuracy Comparison Test 2023." SESTEK, SESTEK, 27 Mar. 2023, www.sestek.com/speech-recognition-accuracy-comparison-test-2023-blog.

[4] 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/.