EyeCU - Assistive Eyewear

ECE 445 Design Document

Team 36 - Nikhil Mehta, Irfan Suzali, Abishek Venkit

ECE445 Project Proposal - Spring 2020

TA: Shuai Tang

Table of Contents

1 Introduction

1.1 Problem and Solution Overview

Many of us take full vision for granted in our everyday lives and enjoy the benefits of being able to read, identify objects, and navigate around the world without concern. However, it's not that simple for the 285 million people worldwide that live with vision impairments [1]. Although many visually impaired people live perfectly functional, independent lives, there are still some situations that are not fully accessible. For our project, we will be focusing on the specific problem of reading text. The lack of ability to read text is an extremely common problem for people with high visual impairment or blindness, and can prevent a person from being fully independent. For example, if a person with visual impairments goes to a grocery store to purchase milk, they will be unable to identify which milk carton contains whole milk, 2%, or soy without assistance from another person, or an app on your smartphone. This is just one of many examples in which the ability to read text quickly and seamlessly would be extremely beneficial, and we envision a better way to accomplish this task. There have been a few attempts to solve this problem already, with varying success. The smartphone app, "Be My Eyes", [2] solves this problem by connecting people with vision impairments to a sighted volunteer over a video call to provide assistance. This solution works in theory, but requires a network of volunteers willing to share their time out of good will. Relying on others is not true independence, and also has the potential for abuse from insincere helpers on the app. Smartphones are also difficult to navigate for the older blind population. This population is significant, with 65% of people visually impaired and 82% of all blind are over 50 years old [1]. Additionally, as found in [3], 76% of the low vision test group either did not own a cell phone at all, or owed a basic cell phone without downloadable app capabilities, and this group had a mean age of 65. The percentage of people who own smartphones had a mean age of 36. This is a glaring shortcoming of the "Be My Eyes" app. Another solution to this problem is the OrCam MyEye 2 [4], which is a wearable device that connects to a pair of glasses and provides assistive information through computer vision technology and audio. This device has generally positive reception, but costs around $4000, making it inaccessible to a large population. We are not satisfied with the existing solutions to the problem of reading assistance for people with severe vision deficiencies, so we aim to create our own product to help fill this gap.

Our solution to this problem is EyeCU, an eyewear device that has the ability to capture an image using a built-in camera, identify and read text from the image, and finally convert that text to speech and play it in your ear. This system will be wearable, always ready at the push of a button, and an affordable and user-friendly way to assist with reading text. Although the scope of our project is confined to reading text for now, the potential is much greater, and we envision that this system can be extended to run many other computer vision detection tasks in the future. However, even in its most simple state, we believe that the value will still be substantial. **Our target market for our product will be people with severe vision impairments, specifically without the ability to read from any distance. This can range from some visual ability, but without the ability to read, to complete blindness with no visual ability at all.** Our aim is to create an accessible product that is simple and intuitive enough to be used by anyone and everyone who needs it, regardless of age and knowledge of technology. Our product will be affordable, with a projected final cost of under $100. It will also be physically intuitive enough for anyone to use, and always ready at the

push of a button. Finally, it will eliminate the need to rely on anyone else, bringing true independence to the visually impaired community.

1.2 Visual Aid

Figure 1: Example usage of Eyewear Device

Figure 2: Captured Image with Text Detected

1.3 High-Level Requirements

- The speech produced by the device must have a word accuracy of above 90% on clear, unobstructed text, and above 75% for several surface distances up to 0.5 meters away, on a variety of surface textures, text styles, and text clarity.
- The glasses must weigh between 30-70 grams, and be comfortable enough for regular wear.
- The device must be reliable (powered and functional) for 30 translations (2.5 uses per hour for a 12 hour period).

2 Design

For design purposes, our block diagram has been designed with modularity in mind so that testing of different modules could be done independent of other components. We have identified 4 main modules essential for our project to function as proposed. The control unit will control the I/O of the project, sending the correct signals to modules when needed. There are two peripheral modules, the perception module and speaker module, which will handle the visual and audio components of the project. Within the perception module, the Raspberry Pi is responsible for the computer vision task. The power supply module powers all hardware.

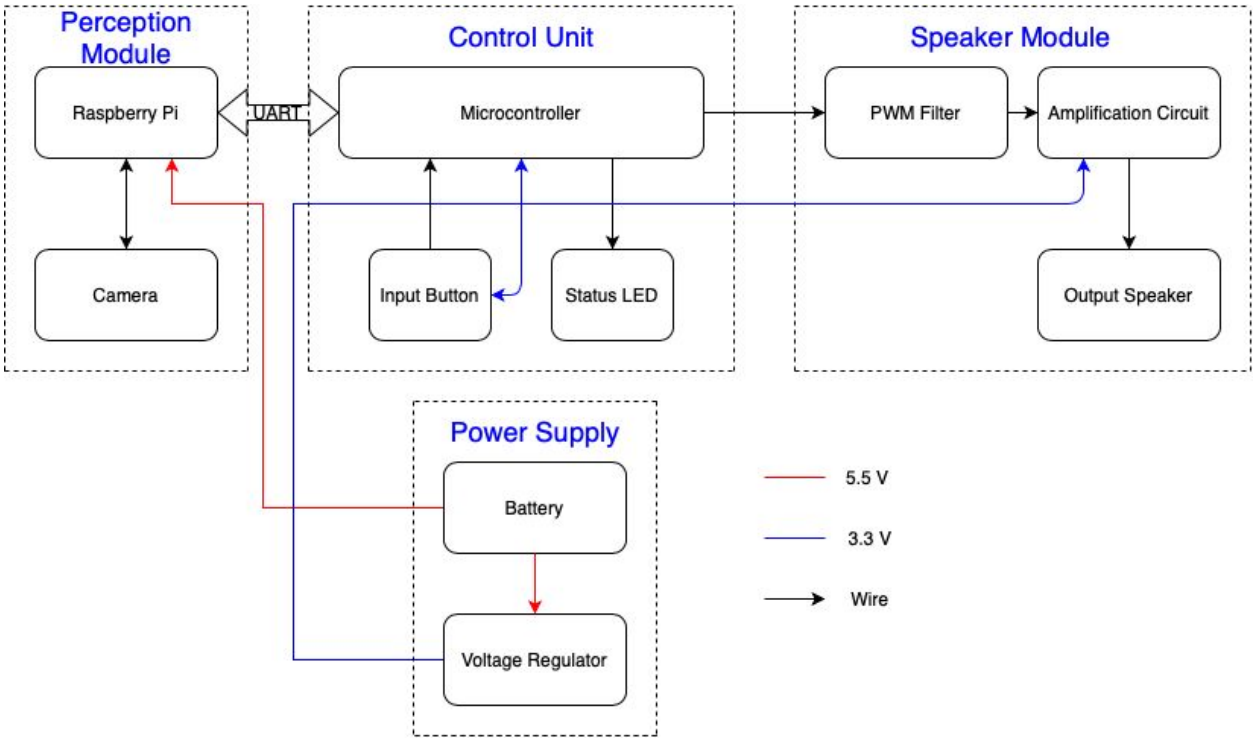


Figure 3. Block Diagram

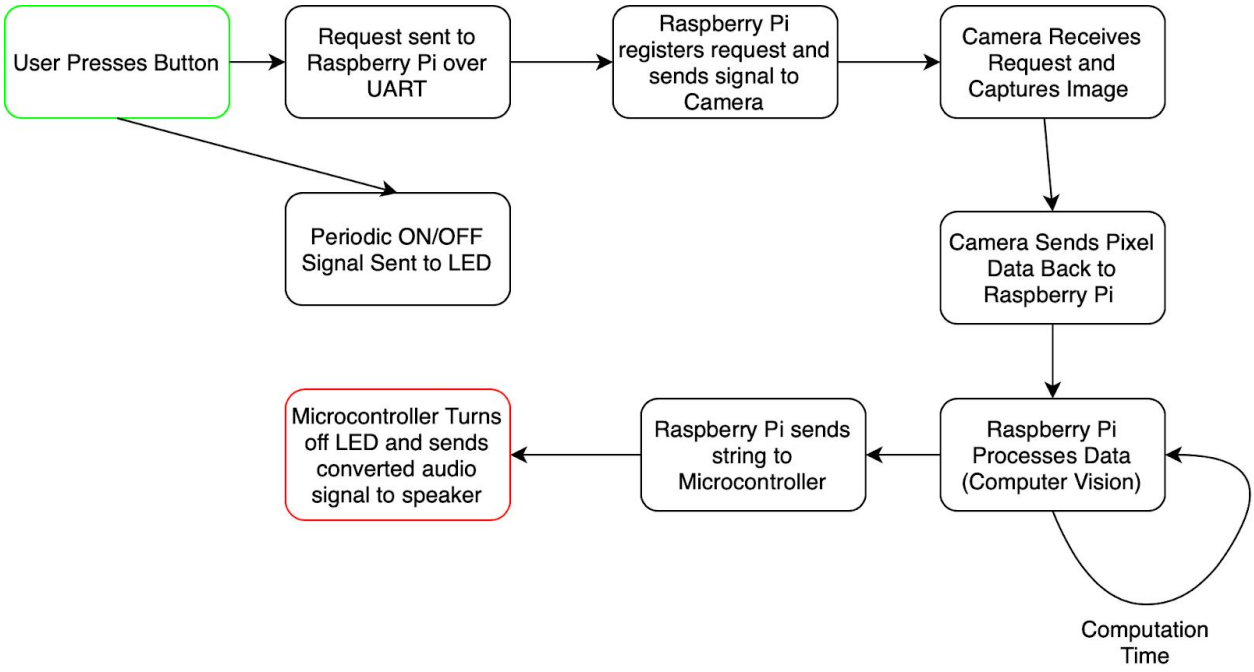


Figure 4. State/Data Flow Diagram

2.1 Physical Design

The physical design is comprised of two main components, the wearable sunglasses and pocket module. The sunglasses will have the camera and PCB board built into the sunglasses, one on each side to distribute weight for stability reasons. Two wires (which will run down the sides of the glasses) will connect the pcb and the camera to the pocket module where the power supply and image processing component is held.

(Wearable Glasses) (Pocket Module)

Figure 5. Physical Design

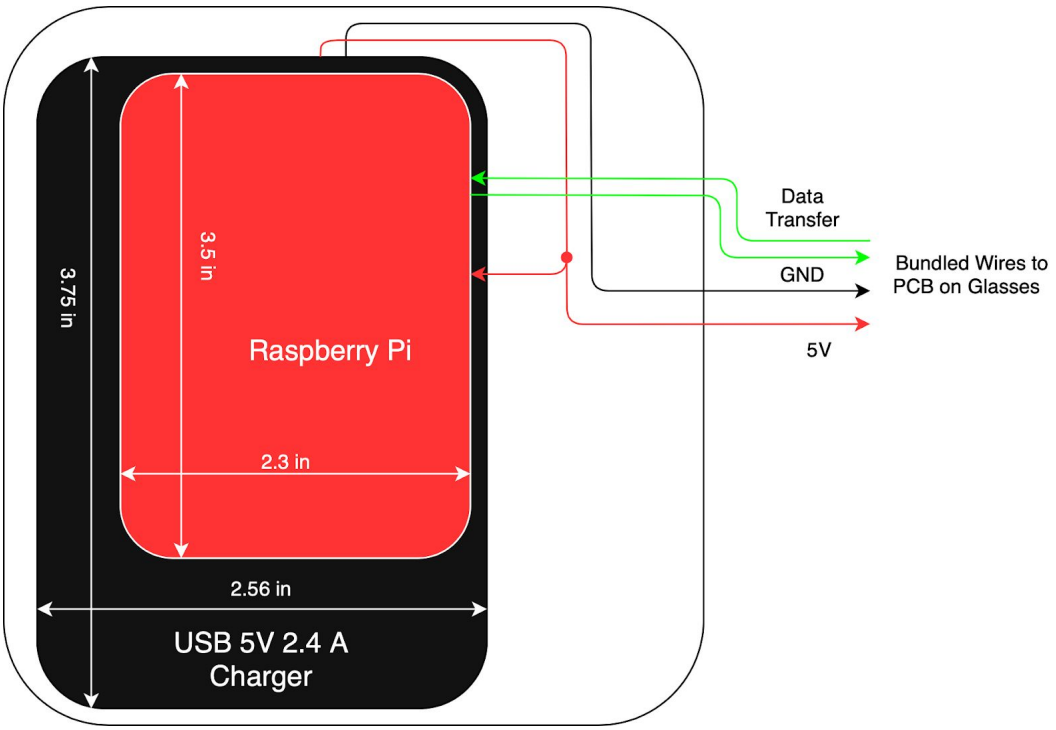


Figure 6. Internal Design of Pocket Module

2.2 Perception Module

The perception module will allow the device to take images and process them to generate text. The camera will take images and transfer this data to the Raspberry Pi. The CPU on the Raspberry Pi will do the image processing, and send a text string out to the control unit.

2.2.1 Raspberry Pi

The device will use a Raspberry Pi 3 for image processing using computer vision (openCV) [5]. The Raspberry Pi will take image input directly from the camera, process this image on the CPU, and send data out. *No other portion of the Raspberry Pi will be used.*

Hardware verification is not necessary (off the shelf), behavior is fixed.

2.2.2 Camera

The camera used will be the official Raspberry Pi camera (5 MP, 1080p resolution). The camera must respond to a button click from the control unit, and take an image of the scene in front of the user with minimal delay [6].

Hardware verification is not necessary (off the shelf), field of view, resolution, and protocols are fixed.

Perception Module *Software* Requirements and Verification	
Requirements	Verification
Software must read the image data, process it, and output string text in < 5 minutes +/- 30 seconds (for demo purposes).	1. Implement the complete Computer Vision OCR algorithm using OpenCV. 2. Start timer, and initiate a camera capture. Wait until the program finishes executing and outputs the string. Stop Timer. Repeat this process for 10+ trials on different text samples, and verify mean value time taken.
Text output must be > 90% +/- 5% accurate for clear, unobstructed text within the camera field of view (basic proof-of-concept test).	1. Implement the complete Computer Vision OCR algorithm using OpenCV. 2. Place a blank sheet of paper, with 3 words, on a white wall. 3. Position the Raspberry Pi and camera 15 cm +/- 2 cm away from the wall, using a ruler to measure. 4. Take a picture with the camera. 5. Allow the algorithm to process the image

	and output a text string. 6. Record the number of characters and words identified accurately in the lab notebook. 7. Repeat steps 2-6 with 10 more 3-word text examples of varying words. 8. Calculate the mean character accuracy and word accuracy based on the 10 trials, and verify that both data points are > 90% +/- 5%.
Text output must be > 75% +/- 5% accurate for unobstructed text, **up to 0.5 meter away** from the camera, within the camera field of view.	**Similar as above, differences in bold* 1. Implement the complete Computer Vision OCR algorithm using OpenCV. 2. Place a blank sheet of paper, with 3 words, on a white wall. 3. Position the Raspberry Pi and camera 10 cm +/- 2 cm away from the wall, using a meter stick to measure. 4. Take a picture with the camera. 5. Allow the algorithm to process the image, and output a text string. 6. Record the number of characters and words identified accurately in the lab notebook, along with the distance of the given trial. 7. Repeat steps 2-6 with 10 more 3-word text examples of varying words. **Each trial, increase the distance by 5 cm.** 8. Calculate the mean character accuracy and word accuracy based on the 10 trials, and verify that both data points are > 75% +/- 5%.
Text output must be > 75% +/- 5% accurate for visible text with **varying surface textures, styles, and clarity.**	**Similar as above, differences in bold* 1. Implement the complete Computer Vision OCR algorithm using OpenCV. **2. 3 distinct objects will be used to assess this requirement.** **a. Canned Food Item** **b. Milk Carton** **c. Cereal Box** 3. Position the Raspberry Pi and camera 15 cm +/- 2 cm away from the object, using a meter stick to measure. The center of the camera should point to the center of the text. 4. Take a picture with the camera.

	5. Allow the algorithm to process the image, and output a text string. 6. Record the number of characters and words identified accurately in the lab notebook, along with the distance of the given trial. **Because of the variety of text quantity, only large, visible text will be counted (ignore fine print).** **7. Repeat steps 2-6 for the same distance, for each object.** **8. Repeat steps 2-7 3 times, varying the polar and azimuthal angle of the object in relation to the camera by +/- 10 degrees.** 9. Calculate the mean character accuracy and word accuracy based on the **9 trials**, and verify that both data points are > 75% +/- 5%.

2.3 Power Supply

The power supply is required to allow usage of all components in the device. This includes the Raspberry Pi, the microcontroller, the camera, the LED, and the amplifier.

2.3.1 Battery

The battery will be a rechargeable, usb charger (compatible with smartphones and other devices). It must provide sustained power to all components for a 12 hour usage period. The specific device used will be **Miisso A-627KN** (10,000 mAh).

Hardware verification is not necessary, (off the shelf), output voltage and current rating is fixed.
Rated for 5V, 3A Max on USB-C output.

2.3.2 Voltage Regulator

This circuit must supply 3.3V to the microcontroller and PCB and 5V to Raspberry Pi, and must handle 5V peak input from the battery and 2A peak current draw from the Raspberry Pi. To satisfy these requirements, we have chosen a switching regulator, specifically the **TPS63070RNMT.** Specifications can be found below.

Hardware verification is not necessary, (off the shelf component). Input/Output voltage ranges are predefined.
Rated for 2V-16V input, 2.5V-9V output, 2A.

Power Supply Module Requirements and Verification	
Requirements	Verification
Power supply outputs 3.3V +/- 0.3V (PIC32 rated for 2.3-3.6V) and 5V +/- 0.25V (Raspberry Pi rated for 4.75V-5.25V) simultaneously, on separate wires.	1. Charge the Battery (rechargeable battery pack) to full capacity, using a USB wall charger. 2. Wire the USB-C power output of the battery into a breadboard. 3. Wire the voltage regulator on the breadboard, such that the input voltage comes from the USB-C, and the output voltage is connected to the PIC32 microcontroller. 4. Directly connect the USB-C power on the breadboard to the Raspberry Pi power input. 5. Use an oscilloscope to measure the voltage running from the power supply to the Raspberry Pi, and verify it is within the given range. 6. Use an oscilloscope to measure the voltage running from the voltage regulator to the PIC32, and verify it is within the given range. 7. Record the exact voltage outputs in the lab notebook.
Voltage regulator and power battery pack remain below 125°F at all times during operation	1. Connect your device setup as described above. 2. Use an IR thermometer to measure the temperature at different operations modes specified below. a. Idle mode (Raspberry Pi is doing no computation) b. Computation mode (time when Raspberry Pi is processing image) 3. Verify a safe temperature for the user and components, and record it in the lab notebook.
Power supply must be able to output reliable power for a 12 hour period, during which there will be 30 translations. The battery pack is rated for 10,000 mAh, thus for	1. Configure Computer Vision algorithm to process text. 2. Connect the power supply lines as described above, to the Raspberry Pi and PIC32 microcontroller.

12 hour usage, the mean current draw must be < 833 mA. Verify the following: (current draw during computation)*(30* average computation time) + (idle current draw * 12 hours) < 10,000 mAh	3. Use an oscilloscope and a current probe to measure the current flow coming directly out of the battery. Do this by connecting the current probe in series with the battery, and configuring the oscilloscope to read out the current. 4. First measure the idle current (no computation, all devices are powered), continuously over 1 minute. Record the mean. 5. Initiate a computation by triggering the camera capture, and during the course of the computation, record the continuous current on the oscilloscope. Determine the average current draw during this computation period. 6. Using the average current during computation and the average current during the idle state, estimate the total power consumption over 12 hours using the above equation.

2.4 Control Unit

The control unit needs to process string data coming from the Raspberry Pi, handle button presses from the user, control the status LED, and output audio to the speaker module. The button presses will be processed by the microcontroller, which needs to communicate with the Raspberry Pi to take a picture and receive string data after computation. Simultaneously, the status LED will be activated. Finally, the text is converted to speech and played by the speaker.

2.4.1 Microcontroller

The microcontroller will be a PIC32, specifically the **PIC32MX270F256D**, and will communicate with the Raspberry Pi using UART. The Raspberry Pi will use the GPIO pins 14 and 15 for data transmission and the PIC32 will use the remappable UART port . It will run the Software Automatic Mouth (SAM) Text-to-Speech (TTS) program, and interface with the status LED and speaker [7]. In order to communicate with the Speaker, it will provide a PWM output on its RB4 output port, which will be converted to an analog signal using a PWM filter.

Microcontroller Requirements and Verification	
Requirements	Verification
Communicate with Raspberry Pi over UART > 0.5	1. Connect PIC32 to Raspberry Pi GPIO

Mbps (low speed requirement because of small data packets, pins support up to 12.5 Mbps)	UART pins, specifically pins 14 and 15. 2. Write a program to generate Random 0.5 Mbit data string to send from Raspberry Pi to PIC32. 3. Start a timer, send data from Raspberry Pi to PIC, and send an echo back, stop timer. 4. Verify the time < 1000ms, and record this time in lab notebook to indicate the latency of Raspberry Pi to PIC32.
Provide valid PWM output on RB4 port, with amplitude of 0.5V +/- 0.05, and frequency 440 Hz +/- 10 Hz, and 50% duty cycle, to ensure signal accuracy and minimal audio distortion.	1. Write a program to continuously generate a 0.5V square wave PWM signal with frequency 440 Hz. 2. Output signal onto RB4 port. 3. Connect output port RB4 to oscilloscope input. 4. Verify the oscilloscope output to fall within the given ranges for amplitude and period.

2.4.2 Input Button

The input button will be a standard push button, and will signal the start of the picture → speech action. The specific button will be a **401-1586-ND**.

Input Button Requirements and Verification	
Requirements	Verification
Easy to press (no destabilization on presses)	1. Press button while wearing glasses and ensure no discomfort or destabilization. 2. Repeat test with 3 participants of different face shapes and sizes.
Reliable 95% +/- 5% of the time (requires a single press)	1. Press button and record whether LED lit up. 2. Repeat for 100 trials and verify accuracy.

2.4.3 Status LED

The status LED will indicate when the computation process is occuring by flashing on and off. This will be used for debugging purposes, and to notify bystanders of an image capture. LED model: **C512A-WNN-CZ0B0151**.

Status LED Requirements and Verification	
Requirements	Verification
Must be visible from 2-3 meters away, 95% +/- 5% of the time.	1. Stand from 3 meters away and identify LED flashes. 2. Repeat 10 times with different participants and record results.

2.5 Speaker Module

The speaker module will amplify the audio signal sent from the microcontroller, and output this signal on a single tweeter. (This implementation is ideal for demo purposes, but a refined product may replace the tweeter with an earpiece).

2.5.1 PWM Filter

The PWM filter circuit will take as input the PWM output from the PIC32 microcontroller, and output an analog signal for amplification by the amplification circuit. It will be implemented using resistors and capacitors.

PWM Filter Requirements and Verification	
Requirements	Verification
Must convert a digital audio PWM signal into a valid analog signal. A valid signal will keep the frequency intact +/- 10 Hz and ensure no amplitude clipping.	1. Output a constant frequency PWM signal from the microcontroller into the PWM filter. a. Test using a standard 440Hz tone 2. Analyze the analog output using an oscilloscope. 3. Verify that the resulting signal is a 440Hz simple sine wave with the same frequency and regularity we are expecting, +/- 10Hz. 4. Repeat test for signals ranging from 300Hz-3,000Hz (normal speech range), and record fidelity.

2.5.2 Amplification Circuit

The amplification circuit will consist of a single opAmp to amplify the signal, powered at 3.3V and 0V on its rails. Resistors will be used to tune the exact amplification factor.

2.5.3 Output Speaker

The output speaker will be a single, small, lightweight tweeter. This will allow the user to clearly hear the audio, while also allowing others to evaluate the device functionality. The specific device number is **CMS-40558N-L152**

Hardware verification is not necessary (off the shelf). This component is rated for a max output of 90dB SPL +/- 3dB SPL from 1 meter away.

Amplification Circuit and Output Speaker Requirements and Verification	
Requirements	Verification
The amplifier circuit must retain all input information (no clipping or distortion) and only increase amplitude.	1. Input a 440Hz sine wave signal into the amplifier input. 2. Analyze the amplified output using an oscilloscope 3. Verify that the sine wave maintains its shape and regularity, but increases in amplitude according to the gain factor of the amplifier.
Must amplify the audio signal enough so that the user can hear the speech signal clearly, specifically at a level of 60dB SPL.	1. Run a test speech signal through the amplifier and analyze the output signal using a reference microphone. 2. Verify that the output signal reaches a level of at least 60db SPL by recording the output and analyzing in Audacity.
Must remain below 125°F in order to maintain the safety of the user and device circuitry.	1. Connect your device setup as described above. 2. Use an IR thermometer to measure the temperature of the amplifier circuit while amplifying a test speech signal. 3. Verify a safe temperature for the amplifier, and record it in the lab notebook.

2.6 Schematics

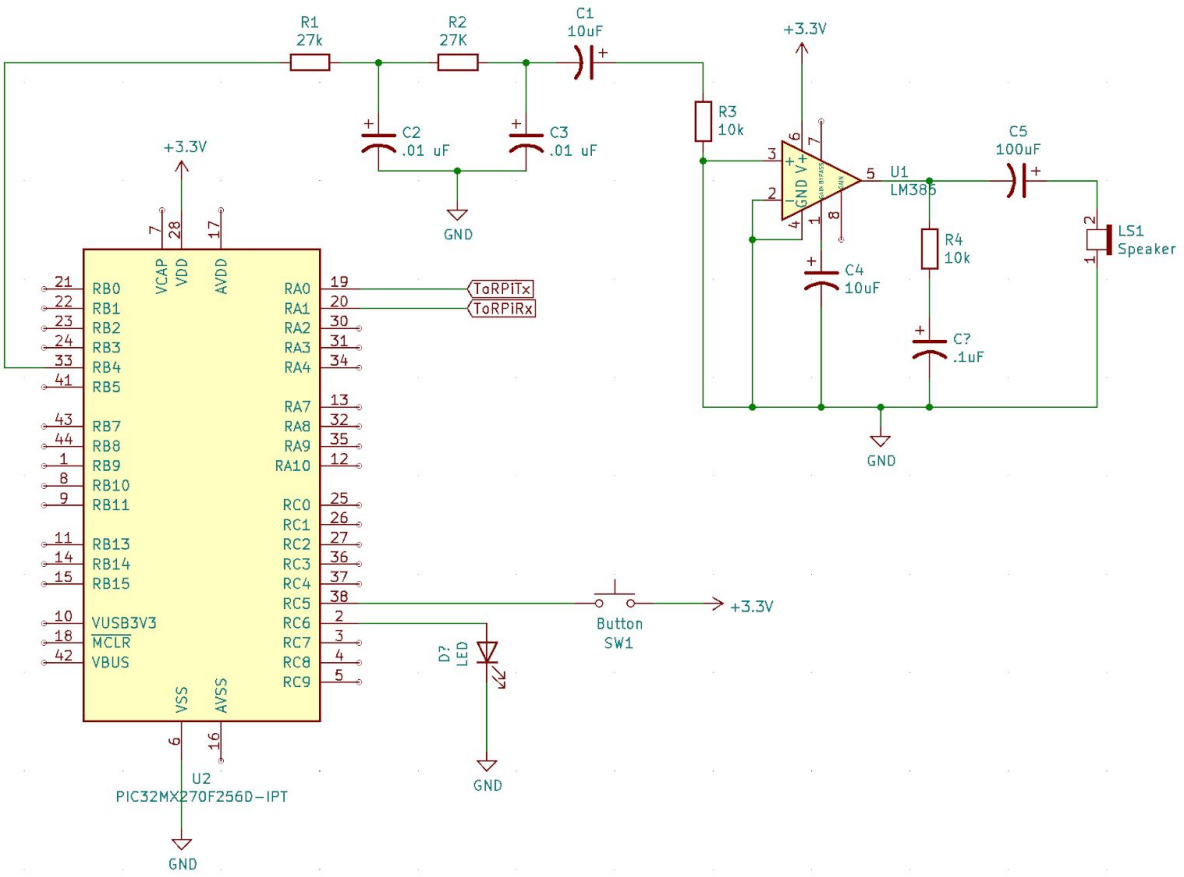


Figure 7. Circuit Schematic for Microcontroller and Speaker Module

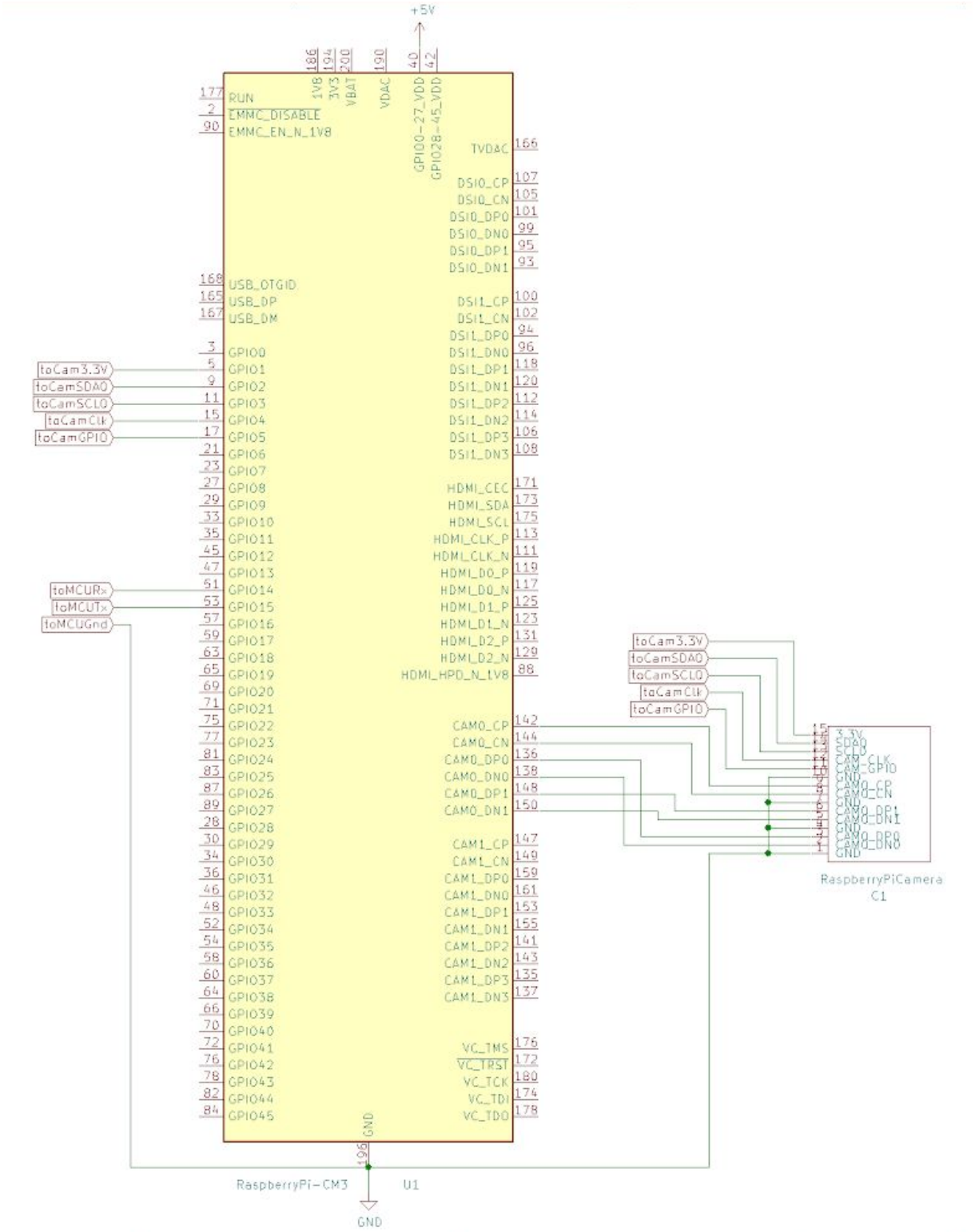


Figure 8. Circuit Schematic for Perception and Battery Module

2.7 Software

The software component of our product will be split into two main computational components. The first is the computer vision text recognition module that will run on the Raspberry Pi. The second task is converting the text recognized from the image into speech audio. This will happen on the microcontroller mounted to the glasses module.

2.7.1 Computer Vision - Optical Character Recognition

The first task is known as Optical Character Recognition, or OCR, and has a number of open source models trained to accomplish this task. We're confident in our ability to do this because this has been a huge area of development for the past several decades, and there are many open source resources that can accomplish this at a very high accuracy rate, including OpenCV Tesseract [8], Google OCR [9], and Kraken OCR. These are well-established and documented, and will help us accomplish our task effectively.

We plan to use the OpenCV open source computer vision library for the OCR task. Their model is called Tesseract [8] and performs both text detection and text recognition. Text detection is performed through the OpenCV EAST model, and the detected entities are passed onto the Tesseract model to create a complete OCR pipeline. This is a more intensive task, so we will be performing this computation on the Raspberry Pi instead of on the microcontroller. We will capture images using the Raspberry Pi camera, and extract text from them. This text is then passed onto the microcontroller as a string.

2.7.2 Text-to-Speech

The second task is text-to-speech, or TTS, and will be performed on the microcontroller. Once the microcontroller receives a string of text from the Raspberry Pi, the microcontroller will begin converting the text into an digital audio speech signal, that is then filtered, amplified, and played back through a speaker on the glasses-mounted PCB. In order to generate the digital audio speech signal, we will use a light TTS model that can be run as a standalone function on the microcontroller.

One of the lightest and most commonly embedded TTS methods used is known as "Software Automatic Mouth", or SAM [10]. This TTS is widely used, and is fast enough to run on even a low powered microcontroller, which is why we plan to use it on our glasses-mounted microcontroller.

2.8 Tolerance Analysis

The success of the project is determined mainly by our ability to convert images to text and text to speech. Thus, it is important that our camera module be able to efficiently and successfully detect text within an image with a certain degree of accuracy. Once processed, we would then need to process the string of text to an audio form that could be outputted with enough clarity for the user to understand. Since our project is centered around the user, we must ensure that the results achieved meet a certain benchmark to ensure usability.

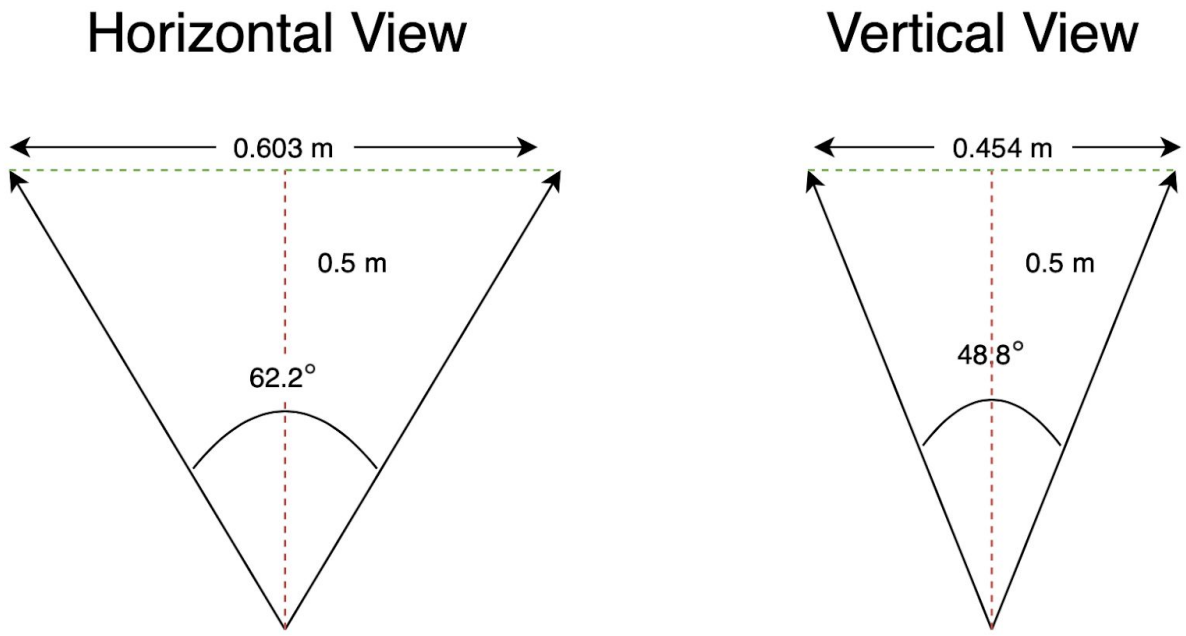
One aspect that we will need to monitor closely is the computer vision and the algorithm that converts images to text generation. In order for our images to be processed correctly, we need to ensure our pictures abide by certain tolerances in regards to the pixel density of characters, and the field of view of our image.

According to official OpenCV Tesseract documentation [8], the minimum suggested Pixels Per Inch (PPI) is 300. This correlates to an x-height of 20 pixels (the number of vertical pixels for a lower case "x" character). In order to satisfy this requirement, we can extrapolate that our software will be accurate (enough to satisfy our high level requirements) as long as images have an x-height of 30 +/- 5 pixels for various fonts and styles. This imposes certain constraints on our image capture system based on the distance from the target and the field of view.

Raspberry Pi Field of View (FOV):

- Horizontal FOV = 62.2°
- Vertical FOV = 48.8°

Below is a visual representation of the FOV, when the distance from camera to object is 0.5 meters. The calculations (1) and (2) are used to determine the field of view window.



$$tan\,(62.2/2\circ) * 0.5\,m * 2 = 0.603\,m \qquad (1)$$

$$tan\,(48.8/2\circ) * 0.5\,m * 2 = 0.454\,m \qquad (2)$$

If we choose to capture images in 720p quality (to save on computation time and power), our pixel density will be 1280x720 pixels. Calculations (3) and (4) calculate the PPI.

$$1280\ pixels\ /\ (0.603\ m * \frac{39.3701\ in}{1\ m}) = 53.917\ pixels\ per\ inch \quad (3)$$
$$720\ pixels\ /\ (0.454\ m * \frac{39.3701\ in}{1\ m}) = 40.282\ pixels\ per\ inch \quad (4)$$

If we choose to capture images in 1080p quality, our pixel density will be 1920x1080 pixels. Calculations (5) and (6) calculate the PPI.

$$1920\ pixels\ /\ (0.603\ m * \frac{39.3701\ in}{1\ m}) = 80.876\ pixels\ per\ inch \quad (5)$$
$$1080\ pixels\ /\ (0.454\ m * \frac{39.3701\ in}{1\ m}) = 60.423\ pixels\ per\ inch \quad (6)$$

For the 2 resolutions, we can calculate the necessary x-height (vertical height of font). Calculation is done in (7) and (8).

$$720p: 30\ pixels\ per\ inch\ /\ 40.282\ pixels\ per\ inch = 0.745\ inches +/- 0.124\ inches \quad (7)$$
$$1080p: 30\ pixels\ per\ inch\ /\ 60.423\ pixels\ per\ inch = 0.496\ inches +/- 0.0823\ inches \quad (8)$$

This reveals that in order to recognize text from far distances, the object used must have a relatively large font size (0.745 or 0.496 inches). These calculations confirm the tradeoff that exists between image quality and distance. For distances closer that 0.5 m, which during general use, can easily be achieved, our software should perform accurately even with smaller font sizes.

3 Cost and Schedule

3.1 Cost Analysis

In the academic year '14-'15, the average salary of a student who graduated with a BS in Computer Engineering was $84,250/year [11]. Dividing the salary by the amount of work hours in a year, 2080 hrs/year, we arrive at an hourly rate of $40.50/hour. We will assume that an employee will work 8 hours a day every week for 16 weeks. We arrive at a total of 128 hours to complete the project.

Table 1. Expected labor cost of a single employee

Employee	**Rate**	**Hours**	**Labor Factor**	**Labor Cost**
Employee X	$40.50	128	2.5	$12,960

Given that our team consists of 3 members, our total labor cost to complete the project is **$38,880**.

3.2 Parts Acquisition

Table 2. Breakdown of parts required with expenses

Parts	**Cost**	**Reason**
Battery Pack	$18	Power Supply
Raspberry Pi Camera Module V2	$25	Capture text
PCB 8 Ohm Speaker	$2.50	Output text to user
General circuit components	$10 (estimate)	Circuit design
Voltage Regulator	$2.77	Power Supply
PIC32	$4.09	Microcontroller
USB C to microUSB cable	$7	Connect battery to Raspberry Pi
Raspberry Pi 3	$35	Person Computer Vision Task
Total Cost Per Unit	**$95**	

The grand total for the completion of the project is $38,880 + $95 = **$38,975**.

3.3 Schedule

Table 3. Full project work distribution schedule

Week	**Deadlines/Objectives**	**Irfan**	**Nikhil**	**Abishek**
2/24/2020	Design Document check	-Establish cost and schedule -Finalize and order parts -Develop circuit schematics	-Refine problem statement -Develop circuit schematics	-Outline requirements and verifications -Tolerance Analysis -Develop circuit schematics
3/2/2020	Refine Design Document	-Test voltage regulator and microcontroller module	-Design PCB and submit audit	-Test amplifying circuit
3/9/2020	Submit PCB order and complete testing of hardware modules on breadboard	-Test Raspberry Pi and camera module	-Research OpenCV implementation on Raspberry Pi	-Developing text-to-speech software
3/16/2020	SPRING BREAK			
3/23/2020	Refine PCB design and	-Solder components and	-Refine PCB design	-Solder components

	test modules on PCB board	test PCB		and test PCB
3/30/2020	Testing	-Solder components and test PCB -Assist in debugging software	-Develop computer vision software	-Develop text-to-speech software
4/6/2020	Finalize PCB design	-Solder components and test PCB -Assist in debugging software	--Develop computer vision software	-Develop text-to-speech software
4/13/2020	Debug and Final Testing	-Address any issues	-Address any issues	-Address any issues
4/20/2020	Mock Demo	-Practice for mock demo	-Practice for mock demo	-Practice for mock demo
4/27/2020	Demonstration & Mock Presentation	-Practice demo -Start on final paper	-Practice demo -Start on final paper	-Practice demo -Start on final paper
5/4/2020	Presentation & Final Paper	-Review final paper	-Review final paper	-Review final paper

4 Ethics and Safety

We have an obligation to our profession to uphold the highest level of ethical and professional conduct. We stand to follow and commit ourselves to the guidelines stated by the IEEE Code of Ethics. Safety of the user is of utmost importance especially since there are significant hardware components situated on the body of the user. There is a potential danger of hardware components short circuiting and overheating after prolonged usage that could cause harm to the user. We intend to design our product with these risks in mind in accordance to the IEEE Code of Ethics #1 - "To hold public safety first and to disclose factors of our project that might endanger the public" [12]. Mitigating these risks are our main priority. To do so we will only be charging the battery pack whilst the device is not in use. In addition to this, hardware components in the pocket module, which house the battery module and Raspberry Pi 3, will be spaced out accordingly so that electrical contact is avoided risking a short circuit and ultimately device malfunction. The enclosure should uphold OSHA provision standards 1910.303(b)(7)(i) stating "Unless identified for use in the operating environment, no conductors or equipment shall be located in damp or wet locations; where exposed to gases, fumes, vapors, liquids, or other agents that have a deteriorating effect on the conductors or equipment; or where exposed to excessive temperatures."[13] to provide protection against any case of exposure to liquids that could cause a short circuit.

We acknowledge that there is a certain degree of error that can arise from recognizing text from images given different situations. The core of the project depends on users being able to trust our system to identify text with

a certain degree of accuracy. To adhere to the IEEE Code of Ethics #3 - “To be honest and realistic in stating claims or estimates based on the available data.”[12], it is our duty to be honest of the estimates provided from the available data provided to us. To uphold this, we plan on allowing only text identified with word accuracy of above 90% on clear, unobstructed text, and above 75% for several surface distances up to 0.5 meters away. This is to ensure that users can have actionable insight so that they may make decisions accordingly.

Finally, our product would not be possible without the advances in computer vision and text-to-speech algorithms developed by pioneers before us. In accordance to the IEEE Code of Ethics #7 - “To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others.” [12], we would like to formally acknowledge and give due credit to those who have contributed to the open source software, OpenCV and SAM.

References

[1] Pascolini D, Mariotti SPM. Global estimates of visual impairment: 2010. British Journal Ophthalmology Online First published December 1, 2011 as 10.1136/bjophthalmol-2011-300539

[2] "Bringing sight to blind and low-vision people," *Be My Eyes*. [Online]. Available: https://www.bemyeyes.com/. [Accessed: 13-Feb-2020].

[3] Raman Bhakhri, Robert Chun, John Coalter, Walter M. Jay; A Survey of Smartphone Usage in Low Vision Patients. Invest. Ophthalmol. Vis. Sci. 2012;53(14):4421.

[4] "OrCam MyEye 2." [Online]. Available: https://www.orcam.com/en/myeye2/. [Accessed: 26-Feb-2020].

[5] L. Hattersley, "Raspberry Pi 4 vs Raspberry Pi 3B ," The MagPi magazine, 13-Nov-2019. [Online]. Available: https://magpi.raspberrypi.org/articles/raspberry-pi-4-vs-raspberry-pi-3b-plus. [Accessed: 13-Feb-2020].

[6] "Raspberry Pi FAQs," *Raspberry Pi Documentation*. [Online]. Available: https://www.raspberrypi.org/documentation/faqs/. [Accessed: 13-Feb-2020].

[7] "English text to speech on a PIC microcontroller," *ToughDev*, 05-Jul-2019. [Online]. Available: http://www.toughdev.com/content/2014/09/english-text-to-speech-on-a-pic-microcontroller/. [Accessed: 13-Feb-2020].

[8] "OCRTesseract Class Reference," OpenCV. [Online]. Available: https://docs.opencv.org/3.4/d7/ddc/classcv_1_1text_1_1OCRTesseract.html. [Accessed: 28-Feb-2020].

[9] S. Panigrahi, "Google's Optical Character Recognition (OCR) software works for 248 languages," *Opensource.com*. [Online]. Available: https://opensource.com/life/15/9/open-source-extract-text-images. [Accessed: 13-Feb-2020].

[10] S. Macke, "SAM Software Automatic Mouth," SAM: Software Automatic Mouth. [Online]. Available: https://simulationcorner.net/index.php?page=sam. [Accessed: 27-Feb-2020].

[11] Department of Electrical and Computer Engineering, 'Salary Averages', 2018, [Online]. Available: *https://ece.illinois.edu/admissions/why-ece/salary-averages.asp* [Accessed: 3-Oct 2018]

[12] "IEEE Code of Ethics," *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 13-Feb-2020].

[13] Osha.gov. (2020). *1910.303 - General.* | *Occupational Safety and Health Administration*. [online] Available at: https://www.osha.gov/laws-regs/regulations/standardnumber/1910/1910.303 [Accessed 28 Feb. 2020].