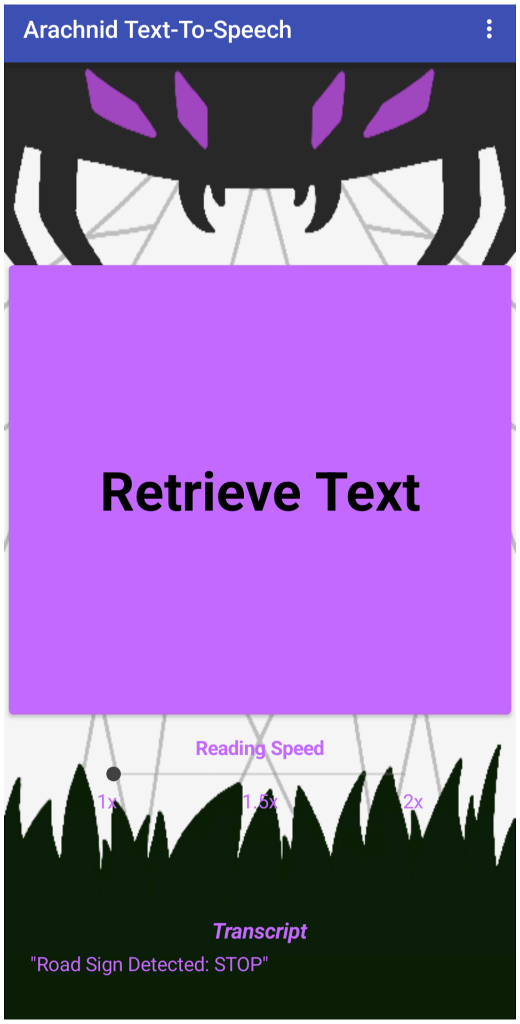
Entrepreneurship Project 2025

Project Arachnid



4/14/2025 - 5/15/2025

Names | Class & Period:
Krithi Shri | Honors Digital Electronics Per. 6
Spriha Trivedi | AP Computer Science Applications Per. 4
Chloe Barlow | AP Computer Science Principles Per. 2

Table of Contents

Problem Description

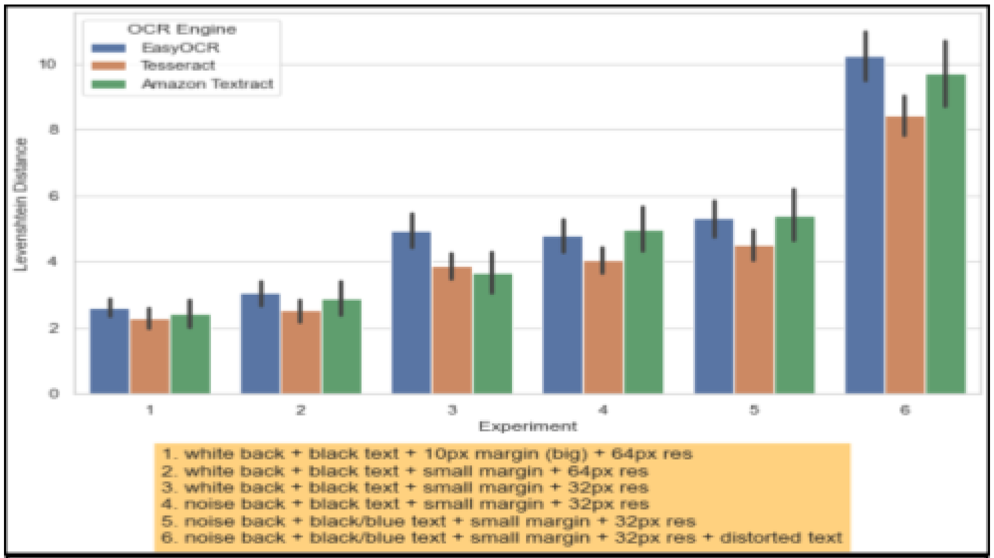
Our target consumers are blind people who struggle with navigating the world due to the fact that braille is not always guaranteed in all signs- things such as letters, street signs, et cetera are not always accessible to them. Not all caution signs can be read via braille either. This includes emergency signs- signs that tell one not to proceed due to hazardous reasons. This is more alarming when considering that the International Agency for the Prevention of Blindness (IAPB) Vision Atlas estimates 36 million people worldwide are considered legally blind, while another 217 million suffer from moderate to severe vision impairment. Our product can also cater to other user groups as well which can include—but is not limited to—those who struggle to read at far distances, people who have dyslexia, people who have difficulty reading a certain language, or even people who simply don't enjoy reading.

Our team intends to create a device that can automatically sense and read out signs and text. This device will non-obstructively notify the user whenever a sign or text is sensed, in which case the user will be able to have the passage read to them with the press of a button unless it is a safety sign, in which case it would be read out nonetheless. This kind of product will provide convenience to those who struggle to live a life in a world that is dominated with writing as a common form of communication alongside the spoken language. The product can unlock certain content that especially blind people never would have had the chance to access in their daily lives and make them feel less alienated and more normal.

Research Summary

According to the World Health Organization, over 2.2 billion people globally experience some form of visual impairment, with a significant percentage living in low- and middle-income countries. Prior solutions have come from Envision AI, Voice Dream Scanner, and Microsoft's Seeing AI. But unfortunately, all of these platforms exclude some portion of the target audience, including limitations like expensive subscriptions, $1.1K to $3.5 K in hardware expenses, and iOS-only services. So to combat this, our project addresses this gap by developing an affordable, iOS and android inclusive, and user-friendly mobile solution that converts images of text into spoken words.

All of the reading aids mentioned implement Optical Character Recognition (OCR), technology that allows computers to convert text found in images to digital text. The origins of OCR technology began from 1920s physicist Emanuel Goldberg's "Statistical Machine," which used a photoelectric cell for pattern recognition. Since then, similar inventions have emerged, initially with the purpose of helping large businesses sort mail and files, but applications later broadened after IBM released the first official OCR machine in 1959. By the 1970s, Intelligent Character Recognition (ICR), an extension of existing OCR technology modified to implement machine learning algorithms, greatly improved the scope and efficiency of digital character recognition, laying the foundation for reading aids such as those listed earlier. Further advancements catalyzed the revival of Tesseract OCR in 2005, one of the most mainstream OCR platforms today, and thus one of our first considerations when developing our product. More research showed that Tesseract OCR accuracy still varies heavily from image to image, ranging from 60%-90% accuracy. From here, our research led us to newer OCR Python machine learning libraries including EasyOCR (2019) and Keras-OCR (2023). EasyOCR was our final choice for the prototype, being the most accurate and simplest to transfer from device to device.



2021 statistics reveal that EasyOCR is on average more accurate than Tesseract OCR in most circumstances. (Amazon Textract was not considered for our project due to paywalls.

Solution Summary

Overview:

The idea that we have decided to develop will take constant footage of the surrounding area using a camera (the prototype utilises an arducam), and detect text. The device will be built using a raspberry pi and a camera. For the notification, the device will be hooked up to a vibrating device that can be clipped anywhere. The camera would be used to identify the text in question using an AI. The AI will then send the text that it has identified into an app for the product that will then interpret the text and read it out loud to the user. Though this solution has the drawbacks of being a physical device, it is optimized for convenient and easy use, and is meant to be easy to access. What distinguishes this product, though, is that it is constantly looking for text. The user does not have to activate it on every occasion and it can actively keep people from hazardous situations by reading emergency signs for them.

CAD & Sketches of the physical device

Main systems:

- Notification System:

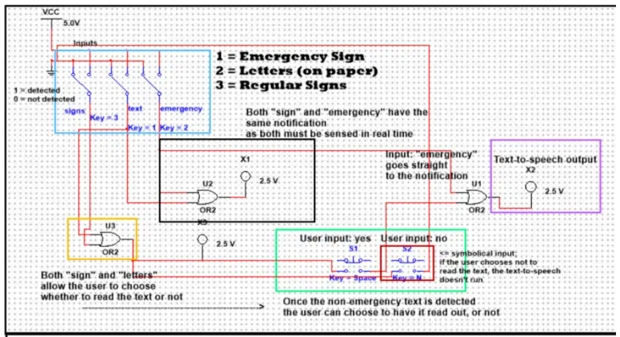
The notification system utilises AOI logic to send a notification to the user, which may look different depending on the case. There are three total cases of text: those being “emergency”; or signs for caution, “sign” for general signs or posters, and “letter” for text on paper- which would mostly come to use in the case of letters. While both “sign” and “emergency” share a notification, as they can only be detected in real time (whereas the “letter” notification is moreso for the sake of telling the user that the camera has picked up on the text that they already know is present), the system goes straight to the text-to-speech output in the emergency case, whereas it allows the user to choose to read the text or not, depending on a second input. In all cases, [0 = “Not Detected” and 1 = “Detected”].

Due to the fact that the only condition that would make it so that the text-to-speech output is not carried out is the user input, which is moreso a secondary condition, the truth table will include neither. If the notification is on, then the text-to-speech will be on unless the user presses “no” rather than “yes,” the AOI portion of the breadboard essentially ends with the notification.

Click here to see the Truth Table, here for the Multisim, here for the breadboard, and here for the PCB.

Equations:

Real-Time Notif = S+E
Letter Notif = T


VCC
5.0V
Inputs
1 = Emergency Sign
2 = Letters (on paper)
3 = Regular Signs
1 = detected
0 = not detected
signs
Key = 3
text
Key = 1
emergency
Key = 2
Both "sign" and "emergency" have the same notification as both must be sensed in real time
X1
2.5 V
U2
OR2
X3
Input: "emergency" goes straight to the notification
U1
OR2
Text-to-speech output
X2
2.5 V
U3
OR2
2.5 V
User input: yes
User input: no
S1
S2
Key = Space
Key = N
<= symbolical input; if the user chooses not to read the text, the text-to-speech doesn't run
Both "sign" and "letters" allow the user to choose whether to read the text or not
Once the non-emergency text is detected the user can choose to have it read out, or not

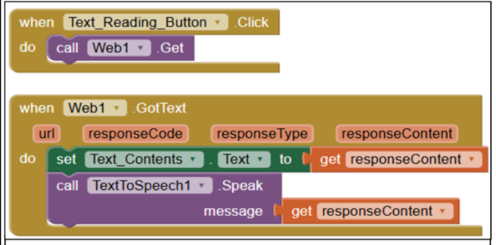
Annotated Multisim Model of Notification System

- Raspberry Pi

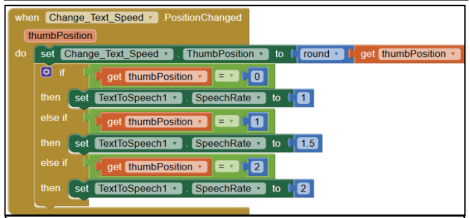
The raspberry pi was intended to be used to link the hardware with the software, and was used to code the pins. The code was not fully functional, as the buttons were simply not sensed by the GPIO (general purpose input/output) pins, which was largely attributable to the age of the microcomputer and the GPIO board used to connect to it causing malfunction. Due to time constraints, it was not possible to find a workaround. Aside from the GPIO board, however, the camera was fully functional, and is capable of providing consistent video footage to the AI at a good resolution.

- MIT AppInventor Text to Speech

The code for the app on MIT App Inventor has three blocks of code as seen below, where two are designated towards the interactable button on the screen and another is designated towards the slider that. Regarding the first two blocks, when the button is pressed, it calls the website that is linked in the 'Web' component added to the app, and that website is the AI program's output. The call takes the output—the 'responseContent'—and inserts that value as the input needed for the 'Text-to-speech' component to function. For the slider, it is set to only have 3 whole digit values and each of those digits is assigned a value that manipulates the speed of the text reading.


when Text_Reading_Button .Click
do call Web1 .Get
when Web1 .GotText
url responseCode responseType responseContent
do set Text_Contents . Text to get responseContent
call TextToSpeech1 .Speak
message get responseContent

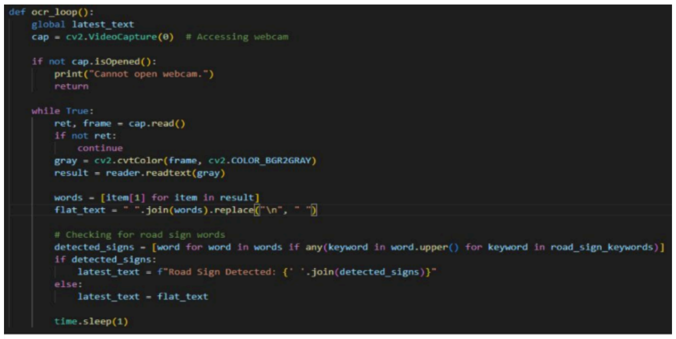
The blocks of code that manage the activation of the text-to-speech function.


when Change_Text_Speed .PositionChanged
thumbPosition
do set Change_Text_Speed . ThumbPosition to round get thumbPosition
if get thumbPosition = 0
then set TextToSpeech1 . SpeechRate to 1
else if get thumbPosition = 1
then set TextToSpeech1 . SpeechRate to 1.5
else if get thumbPosition = 2
then set TextToSpeech1 . SpeechRate to 2

The block of code that manages the speech rate of the text-to-speech (i.e. the speed that the automated voice reads at).

- AI Image Processor

Our AI software accesses a Python's machine learning library, EasyOCR, to recognize text from the raspberry pi's live camera footage and send the processed data from the hardware to the text-to-speech app. EasyOCR allows the program to support 40+ languages with competitive accuracy. The program adds special warning messages when road signs are detected, and then hosts a private website using the raspberry pi's local IP address which only the app can access. The code snippets linked here and shown below feature the program's OCR loop for detected and processing image text.



```python
def ocr_loop():
    global latest_text
    cap = cv2.VideoCapture(0)  # Accessing webcam

    if not cap.isOpened():
        print("Cannot open webcam.")
        return

    while True:
        ret, frame = cap.read()
        if not ret:
            continue
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        result = reader.readtext(gray)

        words = [item[1] for item in result]
        flat_text = " ".join(words).replace("\n", " ")

        # Checking for road sign words
        detected_signs = [word for word in words if any(keyword in word.upper() for keyword in road_sign_keywords)]
        if detected_signs:
            latest_text = f"Road Sign Detected: {' '.join(detected_signs)}"
        else:
            latest_text = flat_text

        time.sleep(1)
```

The OCR loop accesses the raspberry pi's webcam and calls the EasyOCR reader method reader.readtext(image) for frame by frame recognition. Any text containing the road signs specified earlier will be preceded by the special message "Road Sign Detected: ."

Product Evaluation

Our product is currently a very rough prototype and more like a proof of concept as we didn't have enough time to finish the integration of the hardware and software. However, the software portion of the project works as it was intended, where an AI using a camera is able to recognize signs or texts and transfer that text into the app of the product into the text-to-speech function, and the hardware was fully designed.

Regarding the product itself, the biggest issue by far was probably the wires used to connect the camera and the notifier. Though one of the goals was to make the product as non-intrusive as possible, without bluetooth or anything similar, there was no workaround to the wires.

Feedback on our current UI showed positive reactions but also suggestions for increasing text legibility. Changes we would have implemented if given the time include adding a high contrast mode, a simpler and less distracting background, and a menu to customize font size and color to make the app more user friendly for the visually impaired.

85.7% of the people we surveyed reported knowing at least one person in their life who could benefit from our product, with the majority agreeing that it can make small or complex text easier to understand for those with compromised vision.

More future developments that we would have implemented if given the time would include an improved text-to-speech function that can translate from one given language to the user's native language. Additionally, another feature that we weren't able to fully implement was the prioritization of text-to-speech that is more urgent (i.e. the automotive reading out of road hazard signs). In the future, we would also hope for Bluetooth implementation so the text-to-speech isn't spoken out loud and so that the physical wires in the hardware can be minimized, making the product less disruptive overall. With more time, we could've also explored the potential for one of the hardware components to double as a bag, making the product more convenient for use.

Key Contributors Page

Chloe Barlow

My contributions to the project are mainly considered as the web designer or UI designer for the app development. I created the app's interface and design so that it can serve the product's main and simple purpose of repeating what is read by the AI to the user while also maintaining the marketing side of the design. Specifically, I created the app on MIT App Inventor and designed the background of the app using a sketchbook app. The block coding that I did was programming the text-to-speech function where if the user presses the button in the application, the text-to-speech will read whatever it is given. I also added a function where the user can adjust the text reading speed by moving a slider. And one final aspect that I added to the app was a transcript of what was being read.

Spriha Trivedi

I developed the python software for the project, using machine learning library EasyOCR to implement image to text algorithms. I researched alternative libraries as well, listing the pros and cons of Tesseract OCR, KerasOCR, and EasyOCR, deciding on EasyOCR because of its high accuracy and easy transferability of code from computer to Raspberry Pi, our intended final hardware. After finding the optimal library, I coded my program to host a private website that our MIT AppInventor app can access by URL using the Raspberry Pi's local IP address. This ensured that our program ransecurely, safe from external interference from other networks. Using EasyOCR, this website outputs text detected by the Raspberry Pi's camera.

Krithi Shri

I created all of the hardware design. I sketched out the concept and created a rough CAD model of it on Fusion 360. Then, for the notification system, I designed a circuit and created and annotated a multisim for this design. I also designed a PCB (printed circuit board) layout of the system, as practically, this design would use one as an actual product. This required me to think outside of the context of the breadboard, as the PCB was meant to connect to the actual raspberry pi. I also coded and breadboarded the raspberry pi using the gpiozero library, though that was ultimately a lost endeavor due to both my own unfamiliarity with the microcomputer, and the fact that it was at least 5 years old at that point.

References

Envision. (n.d.). *Envision App*. Envision App - OCR that speaks out the visual world. https://www.letsenvision.com/app

Microsoft. (n.d.). *Seeing AI*. Seeing AI | Microsoft Garage. https://www.microsoft.com/en-us/garage/wall-of-fame/seeing-ai/

Research Outreach. (2018, July 30). *IAPB: Envisioning the future of universal eye care*. IAPB: Envisioning the future of universal eye care. https://researchoutreach.org/articles/iapb-envisioning-the-future-of-universal-eye-care/

Sami, T. (2021, July 15). *"Tesseract" vs "Keras-OCR" vs "EasyOCR" | by Thanga Sami | Medium*. Thanga Sami. https://thangasami.medium.com/tesseract-vs-keras-ocr-vs-easyocr-ec8500b9455b

Tripathi, P. (2025, April 11). *The Brief History of OCR Technology*. Docsumo. Retrieved May 16, 2025, from https://www.docsumo.com/blog/optical-character-recognition-history

Voice Dream. (n.d.). *Reader Feature List*. Reader Feature List - Voice Dream - Text to Speech Reader. https://www.voicedream.com/reader/reader-feature-list/

Voice Dream. (n.d.). *Scanner Feature List*. Scanner Feature List - Voice Dream - Text to Speech Reader. https://www.voicedream.com/scanner-feature-list/

Westby, S. (2022, August 1). *Control your Raspberry Pi with a BUTTON*. YouTube. Retrieved May 16, 2025, from https://www.youtube.com/watch?v=IHvtJvgM_eQ

World Health Organization. (2023, August 10). *Blindness and vision impairment*. Blindness and vision impairment. https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment

gpiozero — gpiozero 2.0.1 Documentation. (n.d.). Gpiozero.readthedocs.io. https://gpiozero.readthedocs.io/