EECS149/249 Project Report: VISION++

Enrique Flores Medina and Tianyi Liu
enriquefloresm@berkeley.edu, tianyi-liu-fr@berkeley.edu

ABSTRACT

According to Eye Community Health Journal, 253 million people around the world are either blind or visually impaired [1]. VISION++ empowers them, by providing a glove that can turn any readable text into speech. The glove is connected to the user's phone hotspot, and then, by simply pressing a button, it captures and uploads an image through WIFI to the cloud storage platform Google Drive. The image is then processed using Amazon Textract, a machine learning service that detects text, and the results are written in a Google Spreadsheet. Finally, a smartphone app automatically converts the text in the spreadsheet into speech, which the user can hear with headphones or the phone's speaker.

1 DESIGN AND IMPLEMENTATION

The initial step to generate the design of the project was the modeling of the desired behavior of the system through a Finite State Machine (FSM), which is shown in figure 1.

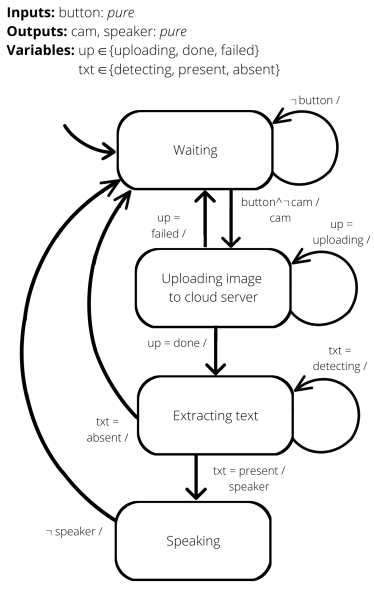

Figure 1: **FSM modeling the behavior of the system**

The initial state of the FSM is waiting. If the FSM is in this state and the button input is received, an image is captured and is uploaded to a cloud server. Once it is uploaded, the text is extracted. Finally, it is turned into speech.

With this desired behavior known, the first implementation decision was made. It was the use of the ESP32-CAM board [2], equipped with a OV2640 camera [3]. This board is low-cost, and includes a WIFI module, to upload the images to the cloud server, an ESP32 chip, to be the microcontroller, and several accessible GPIO pins, to connect the button and other external components. The rest of the design decisions were made to accommodate the use of this board.

1.1 Hardware

In terms of hardware, a PCB was designed for the circuitry between the ESP32-CAM board, the power supply, and the button to trigger image capturing. The schematic for the PCB is shown in figure 2.

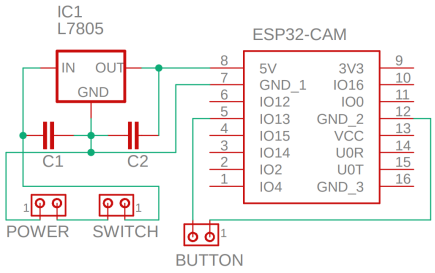

Figure 2: **PCB schematic**

The schematic includes a port to connect any DC power supply ranging from 5V to 35V, a port for a power switch, and a port for the button that takes the picture. This button drives a pin to ground when pressed. The schematic also includes a L7805 5V voltage regulator, and two capacitors for noise filtering. In figure 3, the PCB is shown.

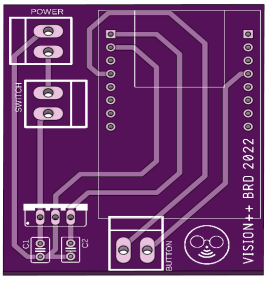

Figure 3: **PCB CAD**

Additionally, an enclosure was designed and 3D printed to hold the PCB, the power supply, and all the components. It is shown in figure 4.

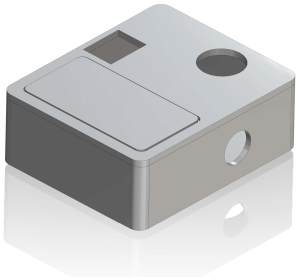

Figure 4: **Enclosure CAD**

1.2 Software

The software part of the project involves: (1) the source code for the ESP32-CAM, which allows for the capturing of images upon the trigger of a button, the formatting of them as JPG, and their uploading to Google Drive, (2) two Python scripts to download images, extract the text and publish it in a Google Spreadsheet, and (3) a Smartphone App to turn the text into speech. All these elements can be found in the GitHub repository, listed at the end of the document. The interaction between these software elements and the hardware elements is shown in the architecture drawing of figure 5.

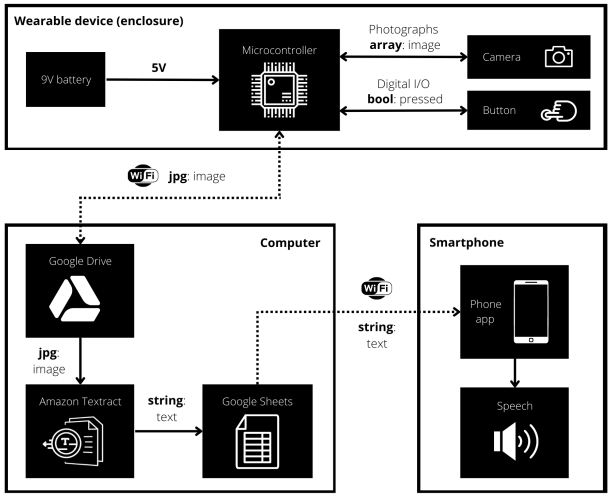

Figure 5: **Architecture Drawing**

The source code for the ESP32 is governed by the following state machine:

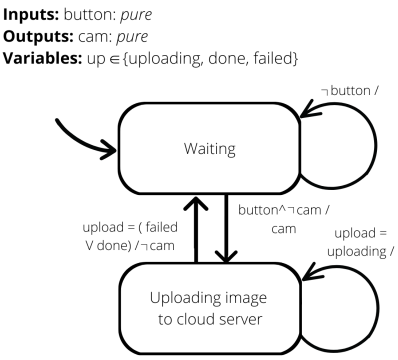

Figure 6: **FSM modeling the source code for the ESP32-CAM**

Upon the receiving of the button input, the camera is triggered and the image is captured and uploaded to Google Drive. When the upload is done, or if the upload fails, the FSM goes back to the *Waiting* state.

The first Python script is used to constantly download the images uploaded from Google Drive. Its behavior is modeled by the FSM shown in figure 7. The output *img_del* indicates that the uploaded image is deleted from Google Drive after being downloaded, minimizing the storage usage.

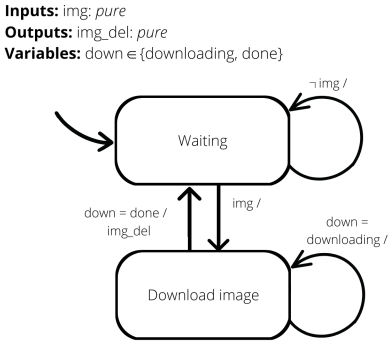

Figure 7: **FSM modeling the code to automatically download uploaded images**

The second Python script is used to automatically extract the text in the downloaded images, and write it in a Google Spreadsheet. Its behavior is governed by the FSM shown in figure 8.

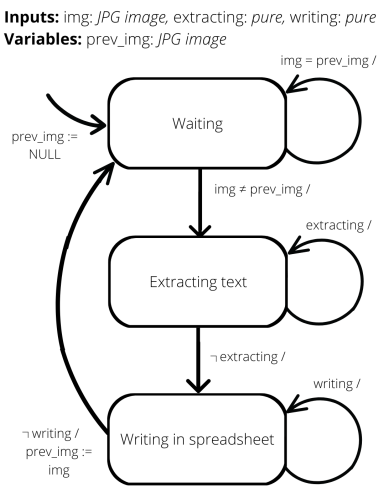

Figure 8: **FSM modeling the code to automatically extract text from images**

This FSM initializes the variable *prev_img* to null. Therefore, as long as the input image is not null too, the first iteration will always transition to the *Extracting text* state. After the text is detected in the image and written in the spreadsheet, the variable prev_img is set to be equal to the image downloaded. Because of this, whenever a new image is downloaded, the condition img ≠ prev_img will be met and the text will be extracted only once.

The Smartphone App is used to automatically turn into speech any text written in the Google Spreadsheet. Its behavior is modeled by the FSM shown in figure 9.

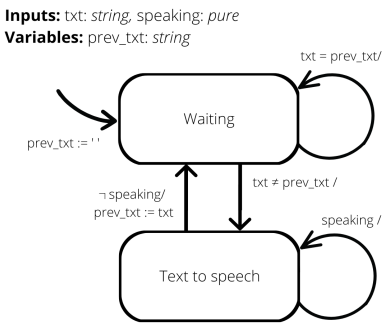

Figure 9: **FSM modeling the Smartphone App to turn text into speech**

Similarly to the FSM shown in figure 8, the variable *prev_txt* is initialized as an empty string. As long as the first input string is not empty, the first transition will always turn the text into speech. After this is done, the variable *prev_txt* will be assigned to be the same as *txt,* and the FSM will return to *Waiting.*

2 PICTURES OF THE BUILT PROJECT

The appearance design of the Smartphone App is shown in figure 10. The recognized text from the image captured is displayed for demonstration purposes only. It can be seen that the app allows the user to choose the speech rate, the pitch, and the language.

Figure 10: **Smartphone App design**

Following are images of the glove device. They show how the user would wear it, and how it looks on the inside.

Figure 11: **Glove device**

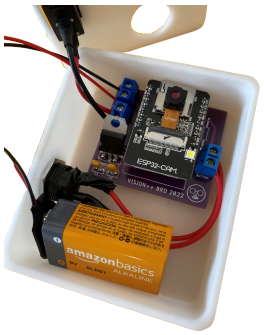

Figure 12: **Inside of the enclosure**

3 PERFORMANCE EVALUATION

High accuracy and reliability on the text detection service is the main goal of the project. To measure its performance, four images were taken with the OV2640 camera and the accuracy of the text detection was calculated for each one. The total results are shown in table 1, and each test is shown in the following figures.

1. **Small handwritten text test**

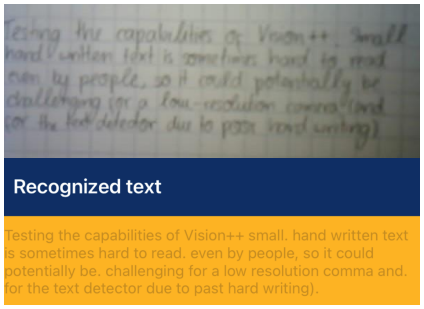

Figure 7: **Testing the project with small handwritten text**

This first test was the reading of small handwritten text. It yielded an **accuracy of 92.1% (35/38 words).**

2. **Small printed text**

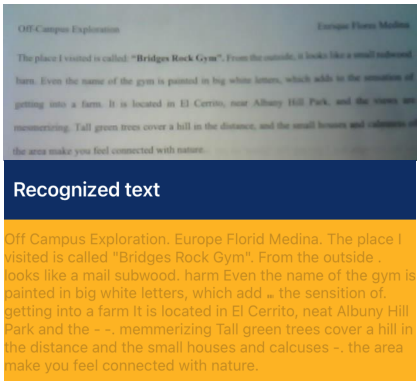

Figure 8: **Testing the project with small printed text**

This second test was the reading of small printed text. It yielded an **accuracy of 89.4% (76/85 words).**

3. **Big handwritten text**

Figure 9: **Testing the project with big handwritten text**

The third test was the reading of big handwritten text. It yielded an **accuracy of 100% (4/4 words).**

4. **Big printed text**

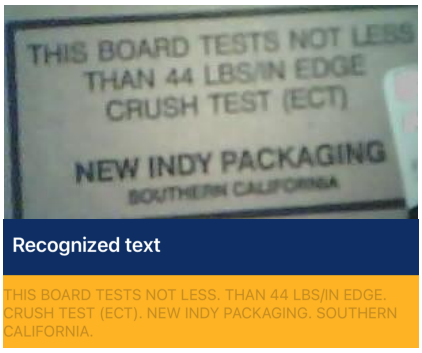

Figure 10: **Testing the project with big printed text**

The fourth and final test was the reading of big printed text. It yielded an **accuracy of 100% (17/17 words).**

Test	**Words detected**	**Accuracy**
Small handwritten text	35/38 words	92.1%
Small printed text	76/85 words	89.4%
Big handwritten text	4/4 words	100%
Big printed text	17/17 words	100%
Total	*132/144 words*	*91.66%*

Table 1. Performance evaluation results

4 KEY COURSE CONCEPTS

The behaviors of all the software aspects of the project are governed by Finite State Machines, which constitutes a key course concept. These FSMs encapsulate the actions to be taken to achieve a desired outcome, and their use is essential.

The communication between the different hardware components, such as the glove, the computer, and the smartphone, is done wirelessly with WIFI. This concept of wireless networks is also key for Embedded Systems, since it opens up endless design possibilities.

Finally, VISION++ uses sensor readings to capture text in the real world and turn it into speech. The camera communication is done using the I2C protocol, and a GPIO pin is used to read the button that triggers the camera.

5 FUTURE IMPLICATIONS

There are many improvements that can be made to the project. In terms of hardware: (1) The ESP32-CAM development board is equipped with several modules that are not used, which increase the cost and the size required for the hardware. (2) Using an led to indicate when the device is connected to the user's hotspot is not useful, since the project is designed for low vision people. A buzzer would be a better option. (3) The resolution of the OV2640 camera is insufficient to accurately detect small pieces of text.

In terms of software: (1) The latency of the image uploading to the cloud service is in the range of 6 to 17 seconds, which makes the use of the device tedious. This would be the number one priority in terms of improvements to be made. (2) The smartphone app is developed with MIT App Inventor, which means it can only be accessed to demonstrate the functionality. A proper app has to be developed for the user to be able to save their desired configuration. (3) The Amazon Textract service, which is used to detect text in the images captured, only identifies text in English. To make the project accessible for people all around the world, more languages have to be readable.

This list is not exhaustive, but it should provide a solid starting point for future work. A smaller and cheaper device would make VISION++ more accessible, and would truly empower low vision people all over the world.

6 DIFFICULTIES DUE TO LACK OF GSI INTERACTION

Without the help of GSIs, acquiring hardware was a travesty. We used a ESP32-CAM development board, which was available for pick up at the beginning of December, making the testing stage of the project extremely delayed. At one point, a 4 hour trip to Richmond was necessary to get capacitors and resistors. Furthermore, a lack of GSIs implied that troubleshooting and debugging was tougher, and the research for problem solving was longer. Not having feedback after milestone 2 and not knowing the date for any deliverables made work very stress inducing, which could have been solved with the help of GSIs.

ACKNOWLEDGMENTS

The conceptual idea for the project is credited to PhD Prabal Dutta. This project is only an abstraction of the full functionality.

REFERENCES

[1] Ackland, P.; Resnikoff, S.; Bourne, P. (2017). "World blindness and visual impairment: despite many successes, the problem is growing", on Community Eye Health Journal. p.p. 71-73. Retrieved from: https://tinyurl.com/visual-impairment

[2] Random Nerd Tutorials. (2020). "ESP32-CAM AI-Thinker Pinout Guide". Retrieved from: https://randomnerdtutorials.com/esp32-cam-ai-thinker-pinout/

[3] Arducam. (2022). "OV2640 – Specs, Datasheets, Cameras, Features, Alternatives". Retrieved from: https://www.arducam.com/ov2640/

GITHUB REPO AND DEMO VIDEO

GitHub Repo:
https://github.com/tianyi-liu-fr/project_ee149_ucBERKELEY

Demo video:
https://youtu.be/q09WU-yeQec