Live Caption Badge

EECS 473
Fall 2024

Angel Cheng, Shuyu Lin, Kyle Rodriguez, Yihang Yin

I. Introduction and overview of project

In noisy environments, such as convention halls or crowded public spaces, effective communication becomes a challenge, especially for individuals who are hearing-impaired or non-native speakers. The combination of high ambient noise levels and language barriers makes it difficult for people to follow conversations, leading to frustration and exclusion.

Our goal is to ensure that everyone at a public event can understand and participate, regardless of the environment, fostering an accessible and inclusive society. We envision a device that can capture speech, transcribe it in real-time, and stream it between participants. This device would display captions in the language spoken, and optionally translate it into another language.

What we made was the Live Caption Badge. At the CoE Design Expo, we demonstrated the device by inviting visitors to wear the badge and talk with us. With the press of a button, the visitor can hear our voice, and we can hear theirs. We also showed the visitor that their speech was transcribed live on the screen.

We achieved most of the goals and milestones from our original proposal. Audio transmission, speech transcription and caption display are all functional. Pairing badges proved challenging, but was eventually implemented. Translation was a feature we made progress on but remained a stretch goal. Overall, we met most objectives with room for improvement.

II. Description of project

i. Goal, the system concept, and feasibility

The goal of our project is to improve communication in noisy environments, such as convention halls, for hearing-impaired individuals and non-native speakers. Our solution is a live caption badge that records and uploads speech to a server running a transcription API. The badge then displays the text on an e-paper screen. Badges can form a talkgroup, where users can hear each other's speech.

The system consists of a number of portable badges and a central server, connected to the same local network (LAN). Each badge is equipped with a headset for speech capture and playback, Wi-Fi and Bluetooth for communication, and an e-paper screen. The badges transmit speech audio to the central server, on which Vosk, an open-source speech-to-text engine, provides real-time transcription. The badges can also pair via Bluetooth Low Energy (BLE). The use of compact hardware makes the system feasible for use in large events, while its power efficiency ensures long-term operation. Overall, the project is both practical and scalable for improving accessibility in noisy environments.

ii. Design Constraints

The design of our Live Caption Badge is constrained by several factors, including budget, time, hardware, and software limitations. With a total budget of $800, we need to prioritize cost-effective components while ensuring functionality and reliability. The project timeline spans from September 5th to December 5th, approximately 12 weeks, requiring efficient planning and execution to meet milestone deliverables.

Hardware constraints include the integration of an audio interface, e-paper screen, microcontroller module, and battery within a compact, wearable badge form. The badge must be lightweight enough to be comfortably worn with a lanyard, and to withstand rough handling. We mostly achieved this goal with a 3D-printed enclosure, although the fragile e-paper remains exposed. The size is 184 × 132 × 55 mm (width × height × depth).

The microphone must capture speech and reject noise. We bought microphones advertised as unidirectional, but it proved difficult to fully isolate the user's voice from others. We also lack proper equipment for acoustic testing.

The e-paper screen must provide large, legible text and refresh fast enough to keep up with speech. We chose a monospace font (Fira Code, OFL license) printed at 48pt. Each glyph is ~6.5×12 mm large, which is readable from a distance of ~2 m. The e-paper can refresh at ~2 Hz. To help the user locate the text as it appears, we split the screen vertically in two halves. Once one half is filled, we clear and print text to the other half. One problem, however, is the degradation of quality and "ghost image" if we don't fully refresh the screen for a long time.

The badge must be power-efficient to ensure prolonged use, at least lasting 4 hours. Per our measurements in Table 1, it lasts much longer.

Table 1 Current draw of Badge in different modes of operation

Mode	**Current (mA)**	**% of time spent in mode over session**	**Average current over session (mA)**
Idle	121	50	60.5
Receiving speech	145	20	29.0
Transmitting speech	130	20	26.0
Both	143	10	14.3
Total			**129.8**

With our current implementation, which lacks a sleep mode, the average current draw over a typical session is around 130 mA. Our badge is powered by a 3200 mAh 18650 Li-ion battery. Assuming 2000 mAh of the battery is usable before the LDO fails to provide 3.3V, we expect 2000 / 130 ≈15 hours of battery life.

Software constraints mainly revolve around networking. We need audio transmission between badges and the server to be reliable and fast, but at the same time we want a protocol supported by official libraries. We chose HTTP, because the official example code uses HTTP. However, we had to make some compromises for ease of development, the reasons for which we will explain in Section II (iii). Under our current architecture, audio latency from one badge to another is ~1.0 second.

Given these constraints, we tried our best to balance component selection, system performance, and user experience to meet the project's goals within the set budget and timeframe.

iii. System Architecture

The Live Caption Badge is a network of badges and a server. Each badge can advertise itself and discover other badges over BLE. The badges communicate with the server via HTTP over Wi-Fi, to form talkgroups, transmit audio, and receive transcriptions. Under our current architecture, badges do not talk to each other over HTTP. Badges are primarily HTTP clients, but act as a server for several simple requests.

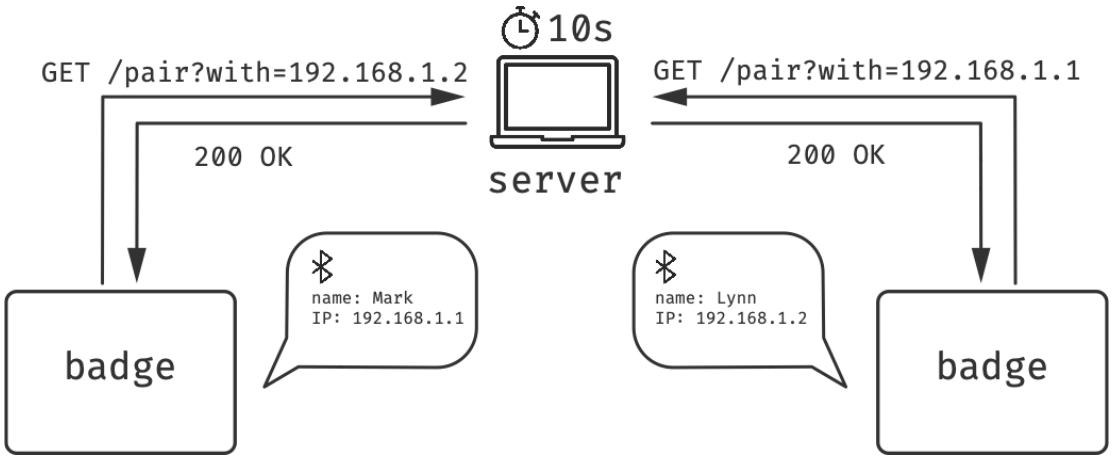


Fig. 1 Two badges requesting to pair.

Figure 1 shows the pairing of two badges. Advertisement and discovery of badges happen via BLE with the GATT profile. Each badge advertises itself to nearby badges. If Badge A wishes to pair with Badge B, it scans for nearby badges. Once it discovers Badge B, it connects with it and reads two GATT characteristics: Username and IP address. The username is shown to the user, and the IP address is used internally to uniquely identify badges.

Once the badges have discovered each other, they prompt the users to confirm or cancel the pairing. Both users must press Confirm in a window of ten seconds. The badges send an HTTP request to the server (GET /pair?with=<IP of other badge>). The server responds with status code 200 to accept the pairing, forming a talkgroup. Talkgroups are currently limited to two badges.

Figure 2 shows the audio transmission and transcription workflow. An ADC chip (ES7210) takes input from an analog (electret) microphone and encodes it into I²S. Once the user of Badge A presses the Unmute button, the MCU (ESP32-S3) begins reading the I²S data. It opens a request to the server (POST /audio) to transmit the audio in 4096-byte chunks with the "Transfer-Encoding: chunked" header. This request is kept alive until the user presses the Mute button. This process is abstracted away by ESP-ADF.

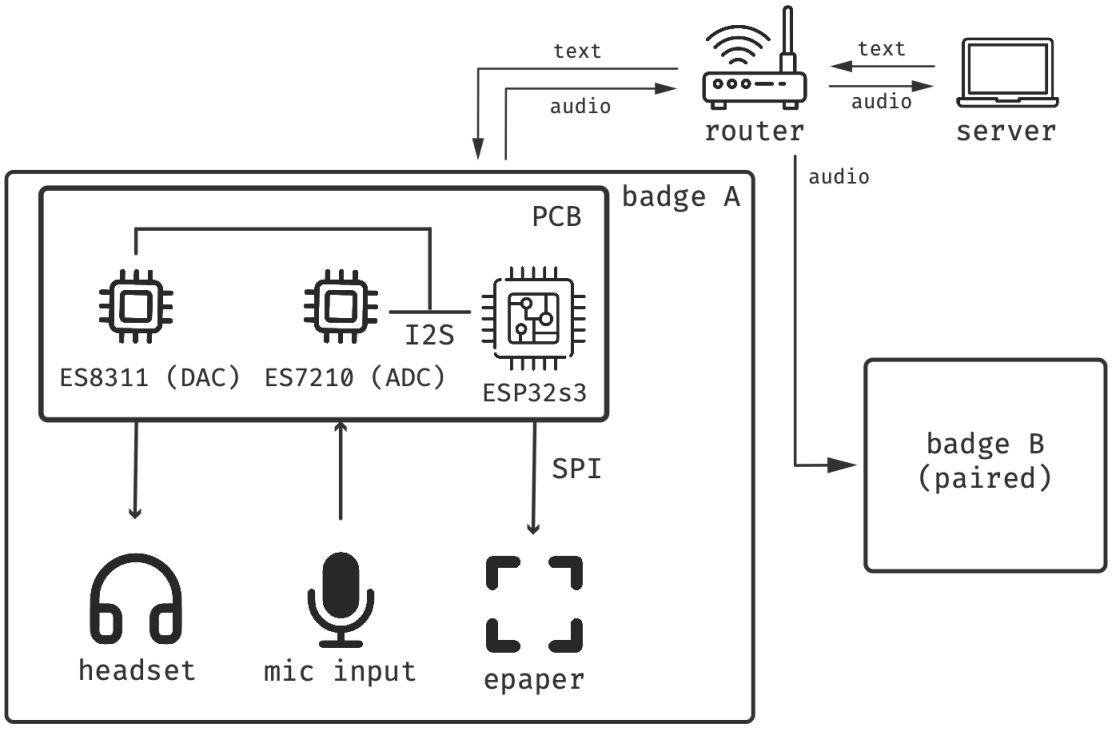


Fig. 2 Overview of the Wi-Fi network.

The server stores audio in a buffer and invokes the Vosk API to generate real-time transcription. Once a transcription is available, the server sends a request (POST /transcription) to Badge A with the transcription as payload.

Once Badge A receives the transcribed text, it is queued to be displayed on the e-paper screen. A task is scheduled to refresh the e-paper screen via SPI once the last refresh completes.

When the server receives a POST /audio request, it also “pokes” Badge B (GET /poke), as shown in Figure 3. Once Badge B receives this request, it requests audio (GET /audio) from the server. The server creates a queue, where all chunks of audio received from Badge A will be pushed. These chunks are then sent to Badge B with the “Transfer-Encoding: chunked” header. Badge B then plays the audio via the DAC chip (ES8311) in the headphones.

Once the POST /audio request is finished, which means the first user has muted themselves, the server sends a GET /unpoke request to the other badge. This request instructs Badge B to terminate the GET /audio request.

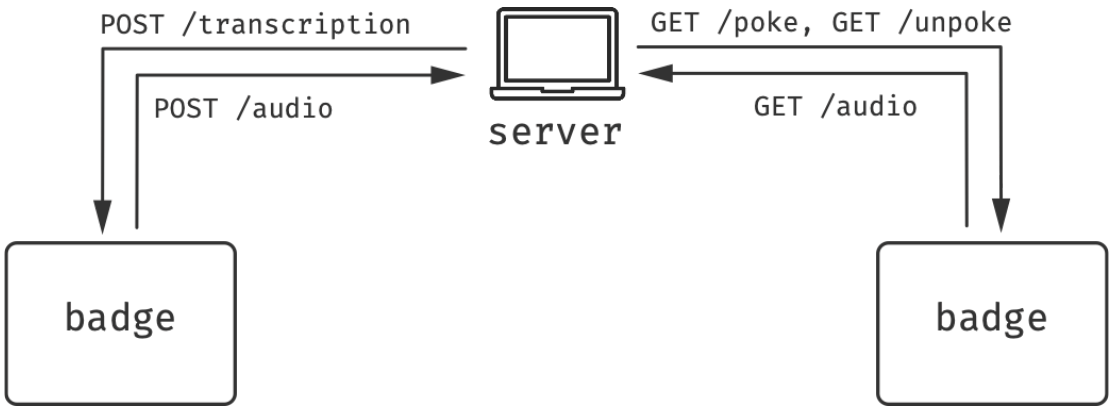


Fig. 3 HTTP endpoints defined on badge and server.

The “POST /audio — GET /poke — GET /audio — GET /unpoke” process described above is suboptimal. Ideally, the audio could be transmitted directly from Badge A to Badge B, without the latency introduced by the server. However, we had to use this workaround, because the HTTP server library in ESP-IDF does not support receiving chunked data, which is vital for audio streaming. If we patch the library, we could potentially cut the latency in half.

The final product can be controlled by the user with three general-purpose buttons located on the top. The software state diagram is outlined in Figure 4. Upon starting the device, it connects to Wi-Fi (SSID and password hardcoded in firmware). Once connected, users can initiate pairing by pressing the Pair button. By pressing the Unmute button, the system enters transcription mode, and the Unmute button

becomes the Mute button. Pressing the Mute button returns the badge to sleep mode, halting both audio streaming and transcription.

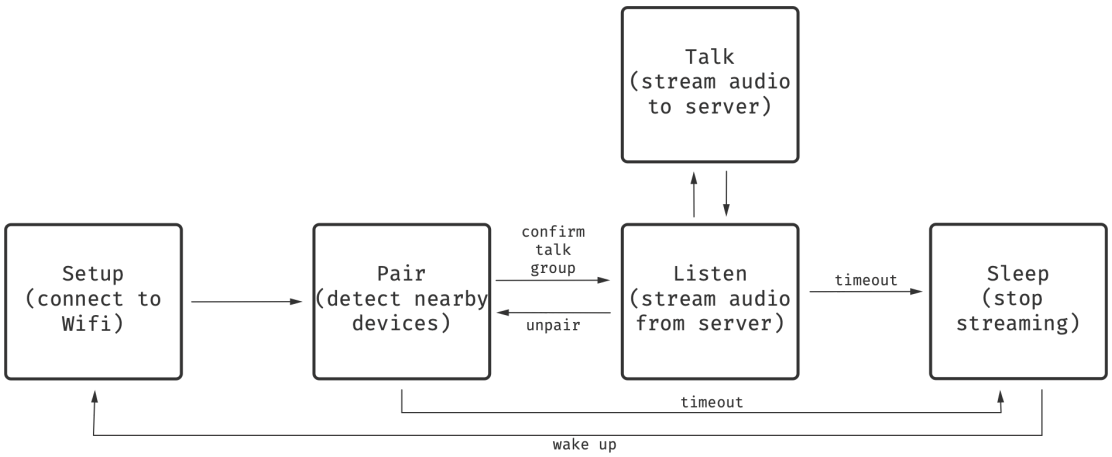


Fig. 4 State diagram of software system on badge.

III. Milestones, schedule, and budget

We encountered a few setbacks that affected both our schedule and budget, although we managed to stay mostly on track. We were a bit behind our original timeline due to some shipping delays and several design changes. Initially, we planned for a cloud-based backend for the transcription service, but after evaluating concerns around latency, we made the decision to switch to a locally run transcription service, which introduced some delays to our schedule as we adapted the system. The PCB design phase went largely according to plan, but we did face additional delays when we had to order a revision 2 of the PCB to resolve issues with the USB charging port and audio chip. On the budget side, we were initially conservative, estimating that we would need $600. However, as minor unforeseen problems arose and additional components were required, we ended up using almost the entire $800 budget. Overall, while we experienced some delays, we managed to make necessary adjustments and keep the project moving forward.

Milestone deliverables:

- Able to establish Wi-Fi or Bluetooth connection and send/receive data

We successfully implemented Wi-Fi communication capabilities. We set up a local server on a laptop that the ESP's can receive and send HTTP requests to. We were able to observe that a connection was established and that data was sent reliably. This was done via sending audio recorded from the badge to the server, which saves it to a WAV file and upon hearing it, was of decent quality.

- E-paper is able to display text

The e-paper display was integrated successfully. After some careful testing, we were able to display text, with little delay between subsequent partial refreshes (0.5 seconds). We also experimented with different styles of showing display, and after some research, settled on a teleprompter style, ensuring clear readability for users.

- Demonstrate that speech-to-text transcription works on a computer

Speech-to-text functionality was implemented using the Vosk API running locally on our server. This was able to be demonstrated successfully on a computer, with accurate transcription of audio input in real time. The system showed promising performance under different speech patterns and environments.

- Written interfaces for the above deliverables

We developed clear and usable interfaces for each of the components, including Bluetooth communication, text display, and speech-to-text transcription. These interfaces made it easy to integrate the various parts of the system and allowed for smooth development and troubleshooting.

- Able to transmit and receive decent quality audio

The audio transmission and reception system worked well, with acceptable quality audio captured by the microphones and transmitted without significant distortion or loss of data. We performed several tests to ensure clarity and consistency of the audio output.

- Demonstrate speech-to-text and display method

We successfully demonstrated the complete workflow, where speech is captured, transcribed to text, and displayed on the e-paper. The process ran smoothly, providing real-time transcription with acceptable delay, showing the core functionality of the device.

- Users can reliably pair devices

We encountered some challenges with ensuring that devices could pair. We experienced occasional issues with Bluetooth pairing, which caused delays in finalizing this feature. In the lack of a third badge, we cannot test the pairing of more than two badges.

- Translation possibly implemented (Stretch goal)

Translation was a stretch goal for the project, and we chose an API that supports translation. However, due to time constraints and the challenges with device pairing, this feature was not fully implemented by the end of the project. If we were given the opportunity to develop our badge more, it would have the ability to have users select what language their transcribed speech is displayed in.

IV. Lessons learned

Angel Cheng:
At first the project felt a bit anxious, especially with the uncertainties around latency and wireless communication, but the final result exceeded expectations. The device not only functions well but also has a polished appearance that gives it a sense of completeness. Looking back, I would tell our group to trust the process, and focus on steady progress. Personally, I've gained valuable experience in designing embedded systems, improving soldering techniques, considering layout designs, and implementing wireless communication methods. These skills have made this project an incredible learning opportunity.

Shuyu Lin:
Throughout the project, my primary focus was on developing the transcription API and ensuring seamless WiFi and BLE connections. While referencing ESP-ADF and ESP-IDF sample code, I realized the importance of designing a user-friendly interface,

as working with the provided APIs can sometimes be challenging. The most difficult aspect of this project was integrating all the functions and pipelines into a cohesive system. This required careful management, as different components could interfere with each other when running simultaneously.

Yihang Yin:
I am glad that we scheduled a weekly meeting early on. It enabled us to check in each other's progress (or lack thereof). Another correct decision is to collaborate on our own branches on GitHub.

The reason why I fell behind on the PCB schedule was that we didn't know what we wanted. We had never agreed on the specifics, e.g. where do the buttons go. Audio circuitry was also a whole new experience for me. As I sat in the Fishbowl 24 hours before deadline staring at datasheets, I felt the pressure that one wrong trace will doom our project.

As it turned out, I had not one, but five wrong traces, four of which being the headphone jack. Sometimes my own stupidity surprises me. A continuity test with an aux cord would have saved me from days of confusion.

However, I must
Learn from my mistakes and try
To let go of them

Kyle Rodriguez:
Much of my contributions came from designing the exterior of the badge. Although this seemed like a trivial task at first, it proved to be more tedious than I thought. A lot of thought had to go into adhering to the dimensions of our components, such as the e-paper screen, PCB, and the placement of its external connectors (USB-C, buttons, headphone jack). In addition, I also had to find ways to minimize filament usage, both because of my limited supply and to lessen the weight of the badge. In the end, I learned more about best design practices, and the importance of coordinating with your team, especially about PCB dimensions.

V. Contributions of each member of team

Team Member	Contribution	Effort
Angel Cheng	Interface programming, soldering, wrote report	20%
Shuyu Lin	Interface programming, transcription, wrote report	25%
Kyle Rodriguez	Prototyping, 3D design and print, wrote report	25%
Yihang Yin	E-paper interface, PCB design, soldering, HTTP, integration test, report	30%

VI. Cost of Manufacture

The cost of manufacturing our live caption badge includes several key components. JLCPCB offers a batch of 1,000 printed circuit boards (PCBs) for approximately $2,000, while the ESP32-S3 microcontroller costs $3.90 each, totaling around $3,900 for 1,000 units. The 3D printed outer casing is estimated at $8 per unit, and the e-paper screen costs $50 each.

Additional miscellaneous costs, such as factory operations, labor, and other materials, also contribute to the overall expense. Based on these estimates, the total cost to mass-produce one device is approximately $73.90, excluding potential economies of scale that could reduce costs further. For mass production, the outer casing can be done a different way than 3D printing, therefore there is room for reducing the cost of one device.

Component	$ Per Unit	$ Per 1k Units
PCB (JLCPCB)	2.00	2,000
ESP32-S3	3.90	3,900
3D printed casing	8.00	8,000
E-paper	50.00	50,000
Misc	10.00	Estimated
Total	**73.90**	**73,900**

VII. Parts and Budget (without tax or shipping)

Description	Item	Qty	Unit cost / $	Total cost / $
DevKit	ESP32-Lyra-Mini	1	22.00	22.00
DevKit	ESP32-S3-Korvo-2	1	50.00	50.00
E-Paper	Waveshare 7.5-inch e-Paper HAT	3	1 × 63.35 2 × 65.99	195.33
Headset	Logitech H111	2	1 × 14.99 1 × 9.99	24.98
Headset	JBL Quantum 100	1	19.95	19.95
PCB + Stencil	JLCPCB	2	Rev 1: 49.72 Rev 2: 44.52	94.24
MCU Module	ESP32-S3-WROOM-1-N16R8	10	3.90	39.00
ADC	ES7210	10	0.77	7.67
DAC	ES8311	10	0.46	4.58
DAC (plan B)	ES8388	5	0.96	4.80

Amplifier	IS31AP2005-SLS2-TR	10	0.51	5.07
Battery boxes	BH-18650-B1BA002	5	1.08	5.40
LDO	HE9073A33M5R	20	0.09	1.86
Pushbuttons	TL1100CF160Q	20	0.46	9.26
Slide switches	SLW-867569-4A-N-D	10	0.44	4.36
Batteries	18650	4	7.49	29.94
Battery charger		1	6.99	6.99
I²S microphone	Adafruit 3421	4	6.95	27.80
Lanyards		5	1.80	8.99
Connectors				23.22
Passives				49.94
Diodes, MOSFETs, BJTs				16.05
Total				**651.43**

VIII. References and Citations

E-paper and caption interface

```
/*
 * Initializes epaper and FreeRTOS stuff. Starts epaper_task.
 */
epaper_err_t epaper_init(void);

typedef enum {
      EPAPER_REFRESH_SLOW,
      EPAPER_REFRESH_FAST,
      EPAPER_REFRESH_PARTIAL,
      EPAPER_REFRESH_CLEAR,
      EPAPER_REFRESH_SLEEP,
} epaper_refresh_mode_t;

typedef struct {
      epaper_refresh_mode_t mode;
      // bounding box position, in pixels
      // start-inclusive, end-exclusive
      UWORD x_start, y_start, x_end, y_end;
} epaper_refresh_area_t;

/*
 * Queue of epaper_refresh_area_t objects. epaper_task pops
 * from the queue and refreshes the display.
 */
extern QueueHandle_t epaper_refresh_queue;

/*
 * Initializes data structures needed for caption.
 * Does not talk to hardware.
 */
epaper_err_t caption_init(caption_cfg_t *cfg);

/*
 * Clears caption area on screen, so that the next chunk of
 * text will be printed in the top left corner.
 */
```

```
epaper_err_t caption_clear();

/*
 * Schedules `string` to be printed in caption area on screen.
 *
 * string: null-terminated C string
 */
epaper_err_t caption_append(const char *string);

/*
 * Updates area on screen dedicated to caption.
 */
epaper_err_t caption_display();
```

BLE interface

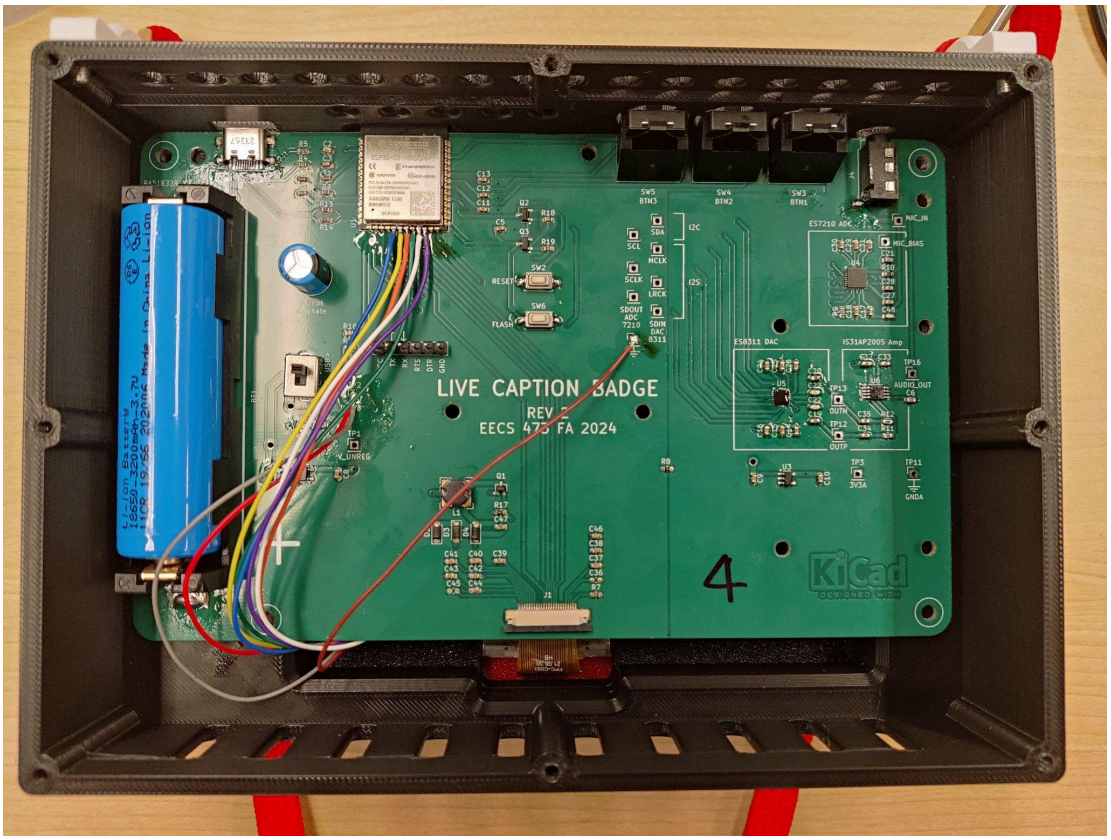
```
/*
 * `name` is shown to user, `ip` identifies badge internally
 */
typedef struct {
     char name[20];
     char ip[20];
} user_t;

/*
 * Begins to scan for nearby badges.
 */
esp_err_t gattc_start(void);

/*
 * Stops scanning.
 */
esp_err_t gattc_stop(void);

/*
 * Initialize GATT server to advertise self.
 */
void       gatts_init(void);
```

```
/*
 * Stop advertising self and deinitialize GATT server.
 */
esp_err_t  gatts_deinit(void);
```

Wi-Fi interface

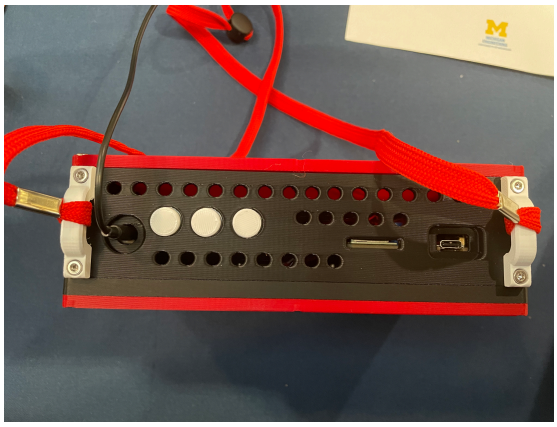
```
/*
 * IP address of self. Obtained when connected to Wi-Fi.
 */
extern char ipAddrComplete[20];

/*
 * Is true iff connected to Wi-Fi.
 */
extern bool wifiIsConnected;

/*
 * Connect to Wi-Fi.
 */
void        wifi_init(void);
```

Full-page, color, picture of soldered PCB

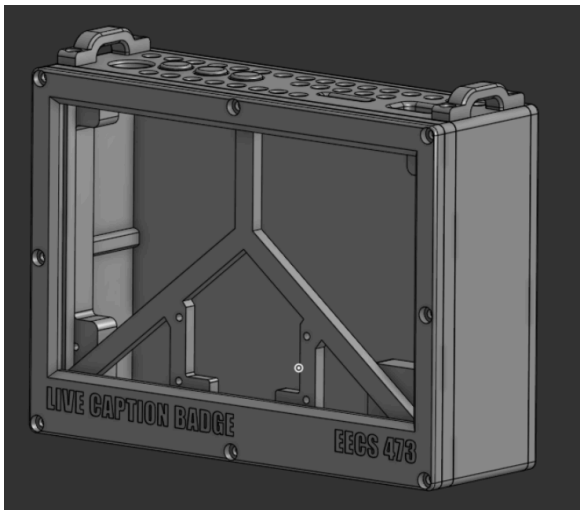


The wires connect the epaper breakout board directly to pins on the ESP32, because our driver circuit (center bottom of PCB) didn't work.

Assembled product



3D model



Credits

Here is a list of libraries and resources we used or referenced throughout the project:

- Espressif, ESP-IDF
- Espressif, ESP-ADF
- Espressif, Korvo-2 devkit schematics
- Waveshare, E-Paper datasheet
- Waveshare, E-Paper breakout board schematic
- Waveshare, E-Paper example code and driver code
- Eurofurence, ef28-badge schematics
- Various other datasheets
- Nikita Prokopov, Fira Code
- stmn, font2bitmap
- All icons used for UI are credited on GitHub

References

Alpha Cephei Inc. (n.d.). *Vosk Speech Recognition API.* Retrieved November 27, 2024, from https://github.com/alphacep/vosk-api

Espressif Systems. (n.d.). *ESP-ADF User Guide: ESP32-S3-Korvo-2 Development Board.* Retrieved November 27, 2024, from https://docs.espressif.com/projects/esp-adf/en/latest/design-guide/dev-boards/user-guide-esp32-s3-korvo-2.html