ECE 445

Spring 2025

Senior Design Final Project

Non-Intrusive Smart Unlocking Mechanism for College Dormitory Rooms

Team 7:

Raghav Pramod Murthy, Arnav Mehta,

Yuhao Cheng

TA: John Li

Abstract

We present a portable, non-intrusive smart door unlocking system that enables secure access using dual biometric authentication. The system verifies both the user's face and voice to ensure accurate identity recognition. An Android application captures a selfie and voice sample, which are sent to a Flask-based server for verification. The server uses DeepFace for facial recognition and SpeechBrain for voice authentication, comparing user inputs against stored embeddings using FAISS. Upon successful verification, the server communicates with an ESP32-S3 microcontroller over Wi-Fi to trigger a gear motor that physically unlocks the door using a flexible steel cable mechanism. Powered by a 12V battery and stabilized with an LM2596 voltage regulator, the system requires no modifications to the existing door, making it ideal for dorms and rental spaces. This approach provides an effective and user-friendly solution for secure and temporary access control.

1. Introduction

1.1 **Problem**

Many college students living in dormitories frequently forget or misplace their keys, leading to lockouts, inconvenience, and even security concerns. For many students, this is their first experience managing their own access, making it easy to lose track of keys or leave them behind. Lockouts often require assistance from residence hall staff or costly locksmith services, disrupting daily routines and compromising safety. While modern technologies like facial and voice recognition offer more secure and convenient alternatives to traditional locks, they typically require permanent modifications to the door or locking mechanism—such as installing cameras, sensors, or electronic actuators. These changes are not allowed in most dormitories or rental properties due to building regulations and lease restrictions. Furthermore, many biometric systems are designed for home environments and are neither portable nor easy to integrate into temporary or shared living spaces.

1.2 **Solution**

To address the challenges of lost keys and the impracticality of installing conventional smart locks in dorm settings, we developed a portable, non-intrusive smart door unlocking system that uses dual biometric authentication—facial recognition and voice verification—to grant access. The system is designed to attach externally to the door without any permanent modifications, making it ideal for dorms, rentals, and other temporary living arrangements.

Our solution consists of an Android app that captures the user's face and voice, sending this data to a Flask-based server for verification. Upon successful authentication using DeepFace for facial recognition and SpeechBrain for voice matching, the server sends an unlock signal over Wi-Fi to an ESP32-S3 microcontroller. The ESP32 activates a gear motor via a driver circuit, which physically turns the door handle using a mechanism. The entire system is powered by a rechargeable 12V battery and regulated by an LM2596 buck converter, ensuring stable operation in compact form.

1.3 **High-Level Requirements**

1. The facial recognition model must achieve at least **90% accuracy**, and the voice recognition system must achieve at least **90% accuracy** in correctly identifying registered users. This accuracy is both on test datasets and actual full-system trials.
2. The system needs to be portable. It should be able to be removed from one door and mounted on another within **10 minutes**.

3. The total time from when a user submits their face and voice for authentication to when the unlocking mechanism fully turns the lock must be **under 5 seconds**.

2. Design

2.1 Block Diagram

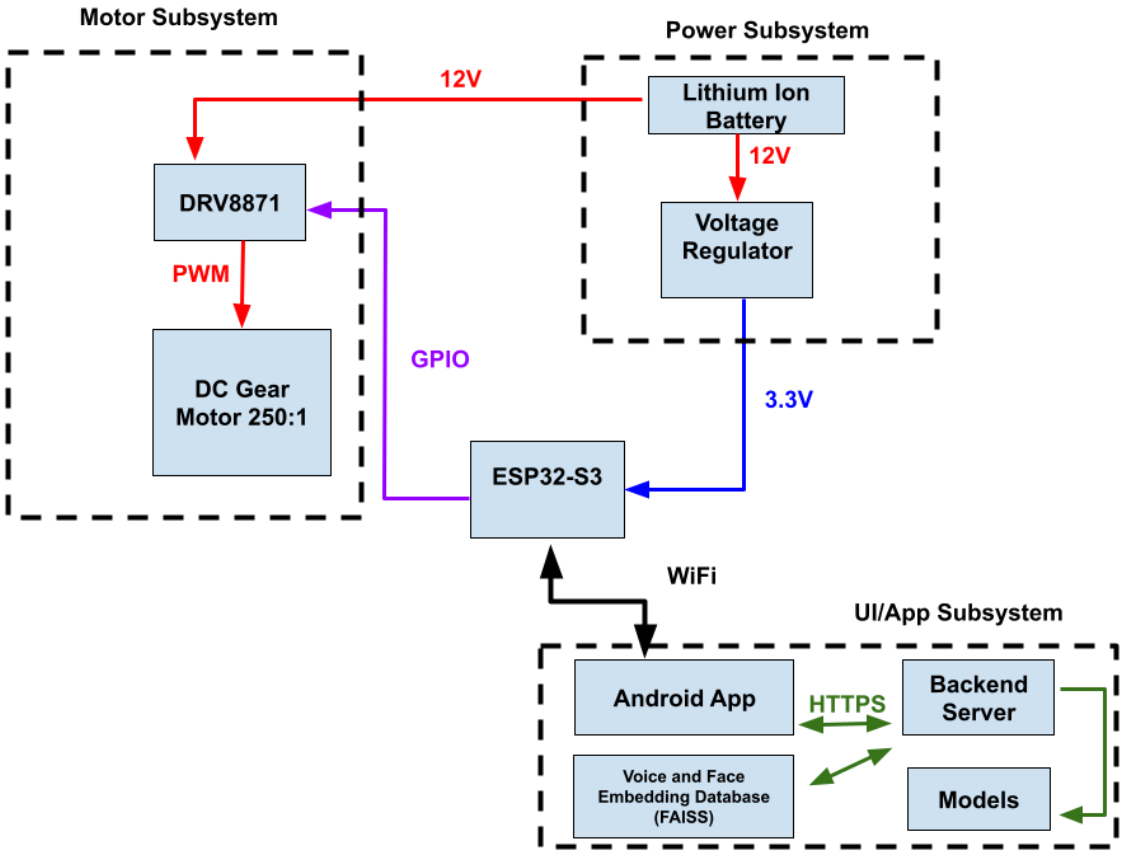


Figure 1: Block Diagram

This system diagram outlines the architecture of the smart door unlocking mechanism, which is divided into three key subsystems: Motor, Power, and UI/App. At the core is the ESP32-S3 microcontroller, which communicates over Wi-Fi with an Android app and backend Flask server. Users initiate authentication through the app, which captures face and voice data and sends it securely via HTTPS to the backend server. The server verifies the inputs using stored embeddings in a FAISS database, applying facial recognition (DeepFace) and voice matching (SpeechBrain) models. Upon successful authentication, the server sends an unlock signal to the ESP32-S3, which controls a DRV8871 motor driver. This driver sends a PWM signal to a 250:1

gear motor, mechanically actuating the lock. Power is supplied by a 12V lithium-ion battery, which also feeds a voltage regulator to provide a stable 3.3V supply to the ESP32-S3. This modular design ensures portability, low power consumption, and non-intrusive installation for temporary housing environments.

2.2 Authentication Subsystem

To build a secure and portable smart door unlocking system, we adopted a dual-biometric authentication approach using DeepFace for facial recognition and SpeechBrain for voice verification. These models were selected for their open-source availability, active maintenance, and real-world reliability, which allowed us to integrate them locally into our system without relying on any external APIs or cloud services. This decision was crucial: running models locally on our Flask backend ensures low latency, increased privacy, and offline inference capability (within the network). Both models also output embedding vectors, allowing us to use a cosine similarity-based matching system—a computationally efficient and widely adopted method for comparing high-dimensional data in real-time.

DeepFace was specifically chosen for its high-level API and support for multiple facial recognition backends like VGG-Face, Facenet, and ArcFace. It provides a complete pipeline that includes face detection, alignment (to normalize facial pose), feature extraction via CNNs, and final verification. The face image is ultimately converted into a fixed-length embedding vector, which is compared against stored embeddings using cosine similarity. This process minimizes the impact of lighting, angle, or minor variations in facial expression. For the voice pipeline, we selected SpeechBrain due to its modular design and TDNN architecture, which captures both spectral characteristics (via CNNs) and temporal dynamics (via TDNNs). Like DeepFace, SpeechBrain also outputs embedding vectors, allowing for consistent similarity comparison logic across both modalities.

To manage and search these embeddings efficiently, we employed FAISS, a high-performance similarity search library developed by Facebook AI Research. FAISS supports approximate nearest neighbor search and is optimized for large-scale vector databases, making it ideal for storing multiple users' biometric data and quickly retrieving the closest match during authentication. This was a conscious design decision to ensure our system could scale to multiple users without degrading verification speed or performance.

The Flask server acts as the central processing unit in our system. It listens for HTTP POST requests from the Android app containing a face image and audio clip. The server parses these requests, processes the image and audio using the models, performs embedding comparisons using FAISS, and returns a JSON response indicating whether both biometric verifications succeeded. All communication between the Android app and the Flask server is secured using HTTPS, fulfilling TLS encryption requirements to protect sensitive biometric data in transit from interception or tampering. We deliberately avoided cloud inference or third-party APIs to

maintain control over security and to ensure the system works in local networks (e.g., a dorm Wi-Fi).

On successful dual authentication, the Flask server sends an HTTP unlock signal to the ESP32-S3 microcontroller. This decision was made because the ESP32-S3 supports both Wi-Fi communication and GPIO control, making it ideal for this kind of IoT application. The ESP32 receives the unlock command and triggers a DRV8871 motor driver using a PWM signal, which in turn powers a 250:1 DC gear motor. Snippets of code that control interaction between the Flask server and the ESP32-S3 microcontroller can be found in the Appendix (Figure 9 under Authentication Subsystem).

D.2 in the Appendix shows how our Flask server communicates with the ESP32-S3 microcontroller to trigger the door unlocking mechanism. After successful face and voice authentication, the Flask backend sends an HTTP GET request to the ESP32's local IP address, including a password parameter in the URL. On the ESP32 side, the request is only accepted if the provided password matches the predefined unlock_password, adding an extra layer of security to prevent unauthorized access. If the password is valid, the ESP32 sets two GPIO pins high to activate the motor driver and unlock the door. This password check ensures that only verified commands from the backend can trigger the unlocking process.

Table 1. Authentication Subsystem R&V Table

Requirement	Verification
The facial recognition model must achieve at least 90% accuracy in recognizing registered users	● Conduct a test using 10 images of registered users and 10 images of unregistered users. ● Measure the True Positive Rate (TPR) and False Positive Rate (FPR) to verify 90%+ accuracy.
The voice recognition system must achieve at least 90% accuracy in identifying registered users.	● Conduct a test where each registered user provides a voice sample, and each unregistered user provides a voice sample ● Measure the accuracy based on successful and failed identifications.

The system must process authentication requests within 5 seconds.	• Conduct 5 authentication trials. • Record the time from when the user submits their face and voice input to when the authentication decision is made (visible by spinning the motor). • Verify that the time taken is within the 5-second limit.

The R&V Table outlines tests to ensure facial and voice recognition each achieve at least 90% accuracy using both registered and unregistered samples. It also verifies the system responds within 5 seconds by timing authentication trials from input to motor activation.

We failed to meet our high-level requirements for model accuracy, largely due to the limited number of facial samples stored per user and the models' sensitivity to input variation. As seen in D.1 and D.3, images taken under good lighting were consistently matched with stored embeddings, but performance degraded under less ideal conditions. D.4 and D.5 show examples where camera flashlight distortion significantly altered facial features, resulting in failed authentications. These false negatives were common because our system initially stored only one image per user, which was insufficient to generalize across varying lighting and angles. To investigate this, we ran a small experiment summarized in D.6, where increasing the number of stored face images from 1 to 8 reduced the false negative rate from 29% to just 5%. These results highlight the need for more diverse data per user to improve model robustness and ensure reliable authentication.

2.3 Power Subsystem

The power subsystem is responsible for supplying power to the robot throughout the duration of the match. It provides two separate output voltages: 12V and 3.3V. This subsystem consists of two main components.

The first component is a 12V input source that supplies power to the motors. The second component is a 3.3V buck converter, which steps down the 12V input to a stable 3.3V output to power the ESP32-S3 microcontroller and its surrounding peripherals.

Initially, a 450mAh LiHV battery was considered. However, after measuring the actual current draw of the system, a 300mAh battery was chosen instead. Calculations confirmed that the smaller battery was sufficient to meet the system's power requirements while still maintaining a safety margin.

-

Linear Regulator vs Buck Converter:
Linear regulators dissipate more power, while buck converters are more efficient.

P_linear = (Vin - Vout) * I = (12V - 3.3V) * 0.35A = 3.045W

P_buck = (Vout * I) / efficiency - (Vout * I) = (3.3V * 0.35A / 0.9) - (3.3V * 0.35A) ≈ 0.128W

An undervoltage lockout circuit was added using a voltage divider to ensure safe startup only above 12V.

Buck converter testing showed:

- Voltage error: 1.8%

Validation:

Requirement	Validation Method
Provide enough power for 2 minutes	Measured draw via power supply, calculated against battery specs
Provide stable 3.3V output ± 0.1V	Oscilloscope measurement of buck output with different input voltages, and applying varying loads from 120 mA to 350 mA

The first requirement is validated through the measurements recorded and referenced in Appendix A.2 using a power supply and multimeter.

As demonstrated in Figure 2, varying the input voltage between 12 V and 8 V had little effect on the output voltage of the regulator. It stayed within the 3.3 V ± 0.1V requirement. The point of this verification was to simulate the real world scenario of the battery's polarity decreasing over time.

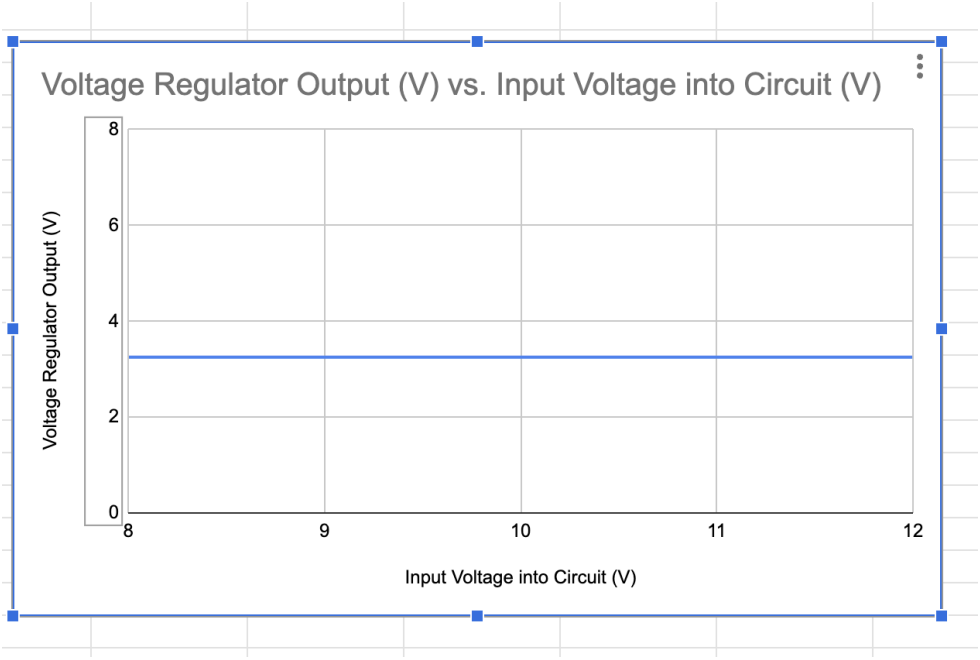


Figure 2: Buck Converter Stable Output vs Varying Voltages

2.4 Motor Subsystem

The motor subsystem is responsible for physically actuating the door lock mechanism after successful user authentication. It mimics the manual unlocking action by rotating the door handle or knob via a flexible mechanical linkage.

Components

- **DC Gear Motor**: Pololu 12V Gearmotor (250:1 ratio)
- **Motor Driver**: DRV8871 motor driver IC
- **Mechanical Coupling**: Flexible adhesive linkage to rotate the lock

Operation

1. **Authentication**: The user passes both face and voice verification via an Android app.
2. **Command Transmission**: The backend server sends an HTTPS request to the ESP32-S3.
3. **Signal Generation**: The ESP32-S3 sets GPIO5 and GPIO18 HIGH and sends a GPIO digital signal to the DRV8871 motor driver.
4. **Actuation**: The motor rotates the door handle via the attached adhesive linkage, unlocking the door.
5. **Reset**: After a few seconds, the GPIO pins are set LOW, resetting the system.

Electrical Details

- **Input Voltage**: 12V (from lithium-ion battery)
- **Control Signal**: GPIO from ESP32-S3
- **Regulation**: Current limited between **0.1A to 1.5A** to prevent overheating

Performance Requirements

Requirement	**Validation Method**
Current between 0.1A and 1.5A	Measured with current probe during operation

Requirement	**Validation Method**
Torque ≥ 0.5 N·m	Torque calculation and verification under load
Unlocking time ≤ 5 seconds	10 trial runs from auth to full unlock
Driver must not overheat	Sustained usage tests with thermal monitoring
System installation ≤ 10 minutes	Measured in dry-run installation scenario

Torque Calculation

To estimate the operating torque of the motor, we assume torque scales linearly with current:

Let:

$T_{stall} = 14\,kg \cdot cm$

$I_{stall} = 1.6\,A$

$I_{oper} = 0.68\,A$

We calculate the operating torque as:

$$T_{oper} = T_{stall} \cdot \frac{I_{oper}}{I_{stall}} = 14 \times \frac{0.68}{1.6} = 5.95\,kg \cdot cm \approx 0.5836\,N \cdot m$$

Final Result:
The motor produces approximately **0.5836 N·m** of torque at the operating current of **0.68 A**.

This **meets the system requirement** that the torque must be at least **0.5 N·m** to reliably rotate the door lock mechanism.

2.5 Microcontroller Subsystem

The microcontroller subsystem serves as the central controller that receives unlock commands from the backend server and sends control signals to the motor driver to actuate the lock mechanism. We used an ESP32-S3-WROOM-1-N16R2 module as our microcontroller. The schematic is based on the bare minimum configuration provided in the course website to allow essential functions like: Power-up control via CHIP_PU pin, UART-based serial communication for flashing firmware and GPIO output for motor control.

This simplified configuration was sufficient to: Upload and debug code over UART, power the module with 3.3V regulated input, and communicate with external peripherals,

GPIO Usage

- GPIO5 and GPIO18 are used as digital outputs to control the motor driver's inputs.
- These pins are set HIGH during an unlock command to activate the driver and then reset to LOW.

The schematic below shows the simplified ESP32-S3 connections. This streamlined schematic allowed rapid development and programming while maintaining enough flexibility for motor control. It also aligned with our space and complexity constraints by avoiding unnecessary peripherals.

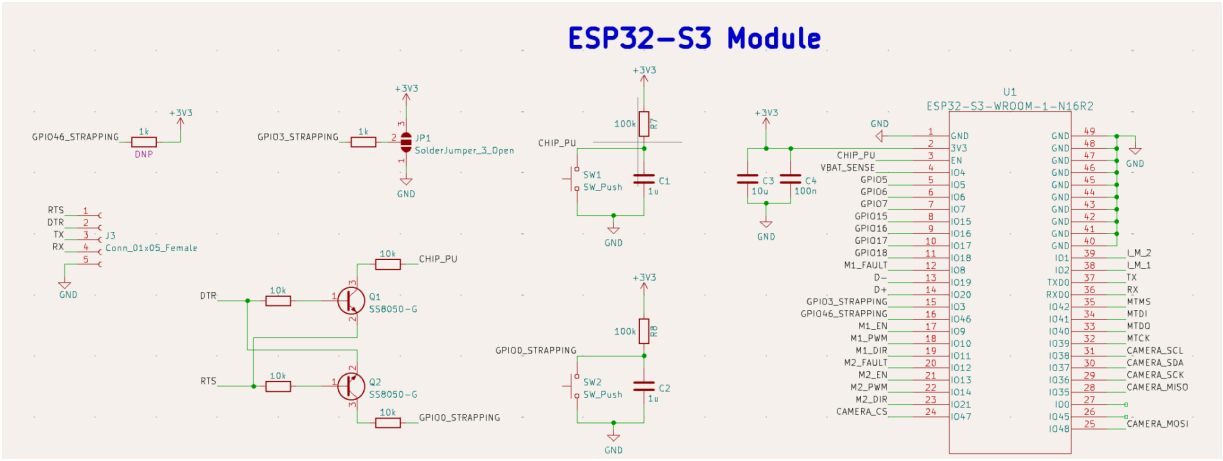


Figure 2 : Microcontroller Schematic

3. Cost and Schedule

3.1 Costs

1. Labor ($60/hr)
    a. Software (36 hours)
    b. Electrical Design (5 hours)
    c. Electrical Manufacturing (5 hours)
    d. Mechanical Design (2 hours)

2. Parts Cost: $93.80 + $0.92 + $63.12 + $12.84 + $6.24 + $66.16 = $243.08
3. Labor Cost: 48 hours x $60/hour = $2880

Description	**Manufacturer**	Quantity	Cost	Link
Pololu 12V 150:1 DC Gearmotor	Pololu	4	$93.8	Link
1N5824 Schottky Diode	STMicroelectronics	4	$0.92	Link
LM2596S-3.3 Step Down Buck Converter	Texas Instruments	6	$63.12	Link
DRV8871DDA Motor Driver	Texas Instruments	8	$12.84	Link
Flexible Steel Wire	SparkFun Electronics	2	$6.24	Link
12 V Lithium Ion Battery	SparkFun Electronics	4	$66.16	Link

3.2 Schedule

Week	Task
March 10 – March 17	● Finalize PCB design and submit design (Yuhao and Arnav) ● Create Flask server to deploy image and voice models for voice and face recognition (Raghav) ● Write code for ESP32 to receive bluetooth signal (Arnav) ● Work on Android app, make UI and connect app to server (Yuhao) ● Connect all subsystems and create mini-demonstration with motor for breadboard demo, get ready for breadboard demo and Design Review (All)
March 17 – March 24	● Make sure that chosen models are robust and have high accuracy (Raghav) ● Connect app to vector database to store voice and image embeddings to store user data (Yuhao and Arnav) ● Improve UI for app (Raghav)
March 24 – March 31	● Soldering PCB and test to make sure the PCB works as intended (All) ● Make sure that the app works with the PCB (app can connect esp32-3 and esp32 can drive motor to open door) (All)
March 31 – April 7	● Create a mounting mechanism for PCB onto the door, make sure that our contraption can successfully turn the lock on the door. Make adjustments to the PCB (change motor, etc.., to make sure that lock will change) (All)
April 7 – April 14	● Prepare for final demo, continue to integrate subsystems together and testing the project (All)

April 14 – April 21	● Prepare for final demo, continue to integrate subsystems together and testing the project (All)
April 21 – April 28	● Final Demo, and prepare for final presentation (All)
April 28 – May 5	● Final Presentation, write final report (All)

4. Conclusion

In this project, we successfully designed and built a portable, non-intrusive smart door unlocking system that supports face and voice-based authentication using machine learning models. The system is tailored for dormitory and rental settings where permanent modifications are not permitted. By integrating the DeepFace and SpeechBrain models with an ESP32-S3 microcontroller and a mechanical unlocking mechanism driven by a gear motor, we created a functional prototype that allows users to unlock their door using just their biometrics.

While we did not fully meet all high-level accuracy requirements for biometric verification, our system was able to reliably authenticate users and trigger the unlocking process end-to-end. More importantly, the project provided valuable learning experiences across both hardware and software domains. We gained hands-on exposure to KiCAD PCB design, soldering and assembling custom circuits, and working with web servers and HTTP communication protocols. These skills—along with the challenges we overcame in system integration, power regulation, and user interaction—have greatly enriched our understanding of building secure, real-world embedded systems. The result is a promising proof of concept that could be expanded upon in future iterations for improved accuracy and broader usability.

Shortcomings

To improve system performance, we would enhance the robustness of our facial and voice recognition models to better handle variations in lighting, background noise, and user expression. This would make the system more reliable at accurately identifying users across a wider range of real-world conditions. Additionally, we would solder the ESP32 directly onto the PCB for improved hardware stability and use a physical Android phone for testing to better reflect actual deployment scenarios.

Improvements for Future

In future versions, we plan to deploy the backend server on Google Cloud Platform (GCP) and implement MQTT for more efficient and reliable communication with the ESP32 microcontroller. We also aim to improve the accuracy and robustness of our facial and voice recognition models to better handle real-world variability. Additionally, we will work on making the entire system more compact and portable, allowing for easier setup and transfer between different locations.

4.1 Ethics

A key ethical concern in this project is the privacy and security of biometric data, including facial and voice information. In line with IEEE Code of Ethics Principle 1, we ensure this data is securely stored and transmitted using protocols like OAuth 2.0 to prevent unauthorized access. To further protect user privacy, we allow individuals to delete their stored data upon request, supporting compliance with laws such as the Biometric Information Privacy Act.

4.2 Safety

The system is designed with safety in mind to prevent electrical, mechanical, or user-related hazards. All electronics are enclosed to avoid contact with exposed wiring, and the motor operates at low voltage to minimize risk. Power is regulated using a buck converter to ensure stable output, and current levels are kept within safe limits for all components. Additionally, the mechanical unlocking mechanism uses low-torque motion to prevent damage to the door or injury to the user.

Appendix A: Power Subsystem

A.1 Buck Converter Schematic

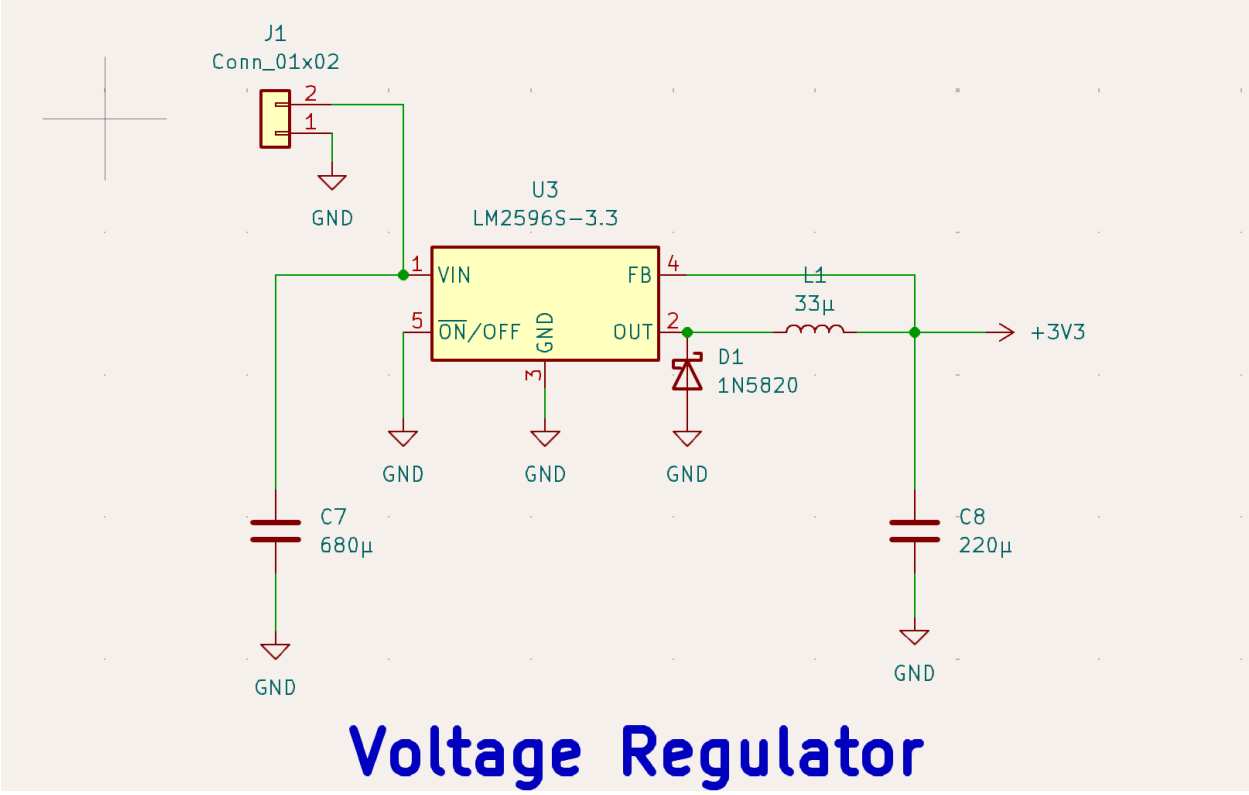


Figure 2: Schematic for 3.3V Buck Converter

A.2

Load Resistor	Output Voltage	Load Current
33 Ohm		120 mA
22 Ohm		169 mA

10 Ohm		341 mA

Figure 3: Load vs Output Voltage Stability for Power System

Appendix B: Motor Subsystem

B.1 Motor Driver Schematic

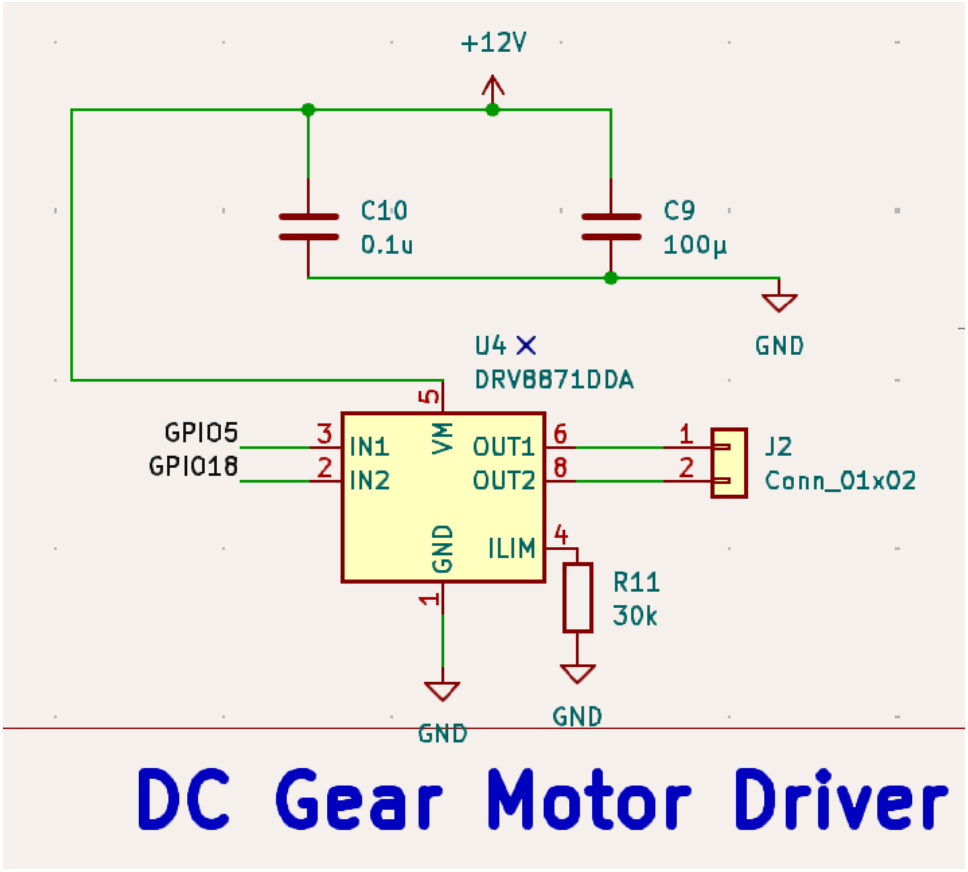


Figure 3: Motor Driver Schematic

B.2 Operating Current Comparison

Figure 5: Operating with no load

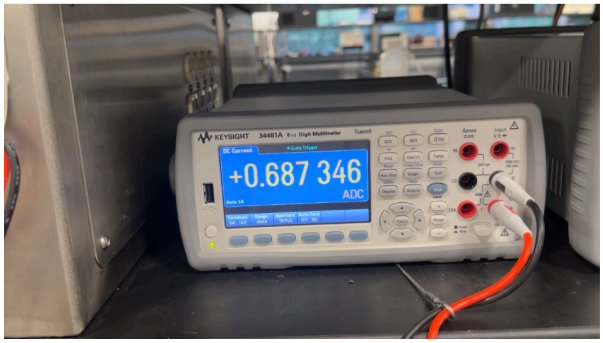
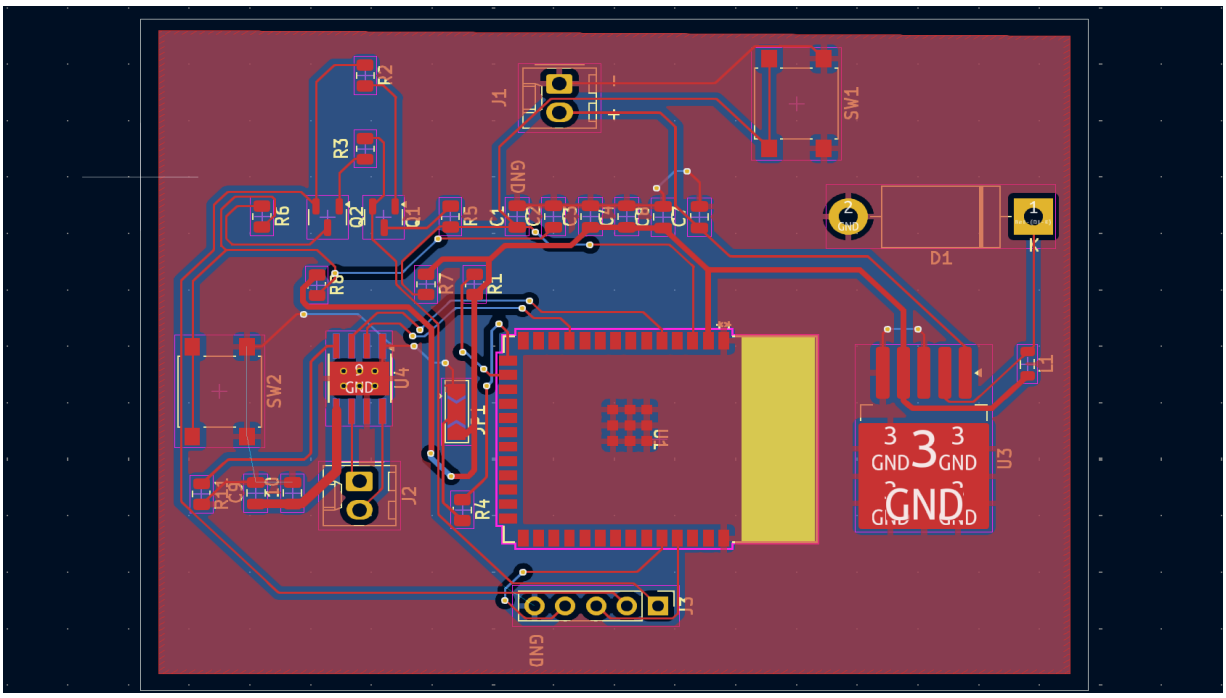


Figure 6: Operating when rotating lock

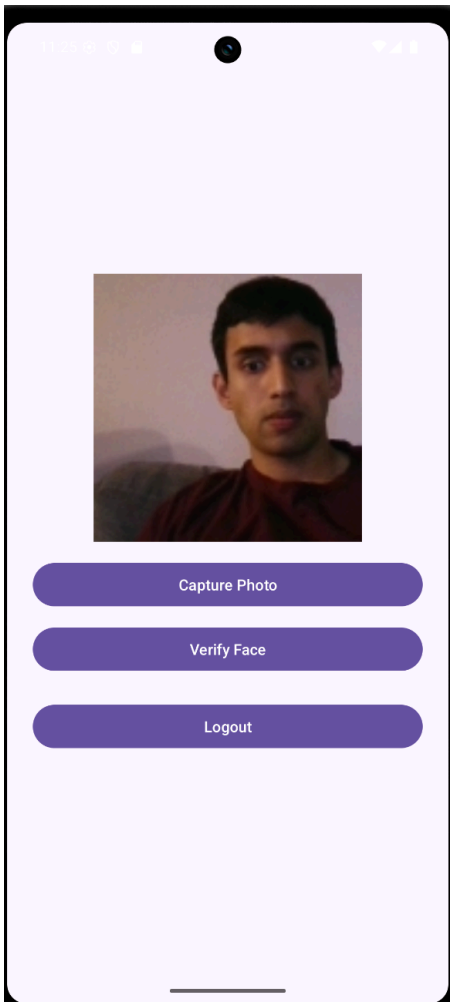
Appendix C: PCB Overview

C.1 PCB



Appendix D: Authentication Subsystem

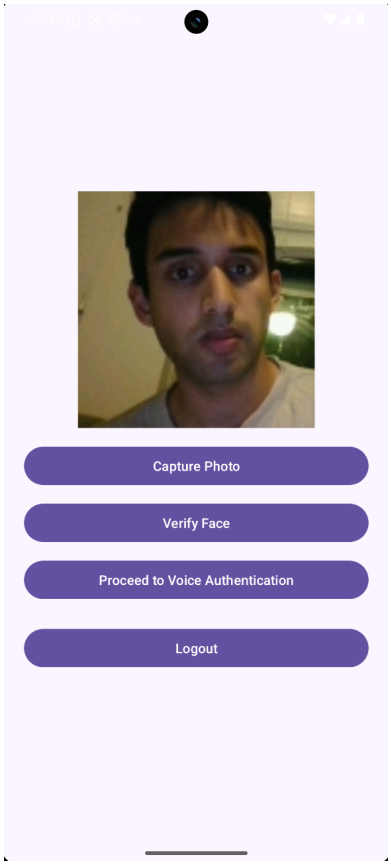
D.1 Android App Overview



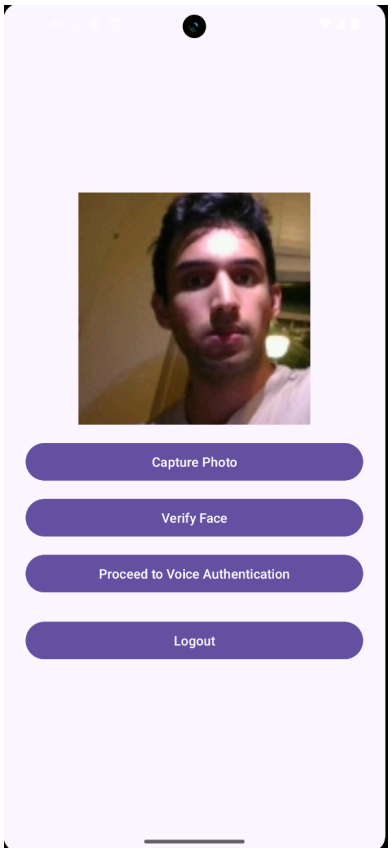
D.2 Flask - ESP32-S3 Communication

```
const char* ssid = "ESP32_Access_Point";
const char* password = "esp32pass";  // WiFi AP password

const char* unlock_password = "mysecret";
WebServer server(80);

const int pin1 = 5;   // GPIO5
const int pin2 = 18;  // GPIO18

void handleRoot() {
  if (server.hasArg("password") && server.arg("password") == unlock_password) {
    digitalWrite(pin1, HIGH);
    digitalWrite(pin2, HIGH);
    server.send(200, "text/plain", "Unlock successful!");
    Serial.println("Unlock signal received. Pins set HIGH.");

    delay(3000);
    digitalWrite(pin1, LOW);
    digitalWrite(pin2, LOW);
  } else {
    server.send(401, "text/plain", "Unauthorized");
    Serial.println("Unauthorized attempt.");
  }
}
```

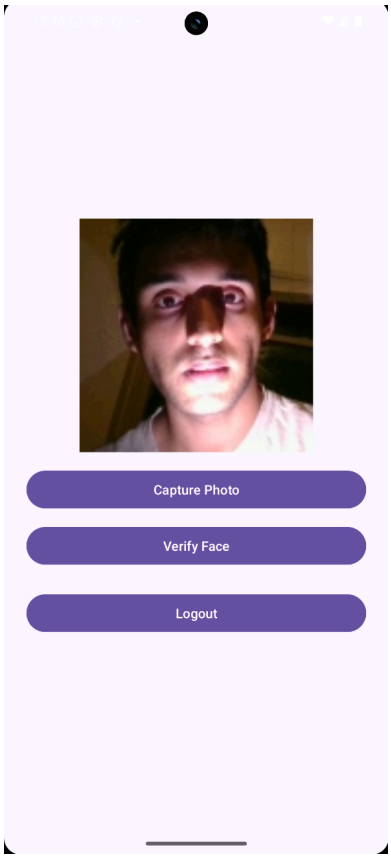
```
# # ESP32 communication
ESP32_IP = "http://192.168.4.1"
ESP32_UNLOCK_PASSWORD = "mysecret"

def send_unlock_signal():
    print("IN SEND UNLOCK SIGNAL")
    url = f"{ESP32_IP}/?password={ESP32_UNLOCK_PASSWORD}"
    try:
        response = requests.get(url)
        print(f"Unlock signal sent! ESP32 response: {response.text}")
    except requests.exceptions.RequestException as e:
        print(f" Failed to send unlock signal: {e}")
```



D.3 Face Image with Good Lighting



D.4 Face Image with Distortion



D.5 Face Image with More Distortion

D.6 False Negative Rates of DeepFace Model

# of Images per User in Database	False Negative Rate (%)
1	29%
5	10%
8	5%

References

1. Rotor Riot Battery: https://rotorriot.com/products/4s-300mah-80c-lihv-battery-with-xt30-connector
2. TI LMR50410 Datasheet: https://www.digikey.com/en/products/detail/texas-instruments/LMR50410Y3FQDBVRQ1
3. TI MCF8316A Datasheet: https://www.ti.com/lit/ds/symlink/mcf8316a.pdf
4. TI MCF8329A Overview: https://www.ti.com/product/MCF8329A
5. Repeat Robotics Hubmotor: https://repeat-robotics.com/buy/2207-battle-ready-hubmotor
6. Repeat Robotics Tangent Drive Motors: https://repeat-robotics.com/buy/repeat-tangent-drive-motors
7. CP2102N USB Bridge: https://www.silabs.com/interface/usb-bridges/usbxpress/device.cp2102n-gqfn20
8. ESP32-S3 Datasheet: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf
9. Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). **DeepFace: Closing the Gap to Human-Level Performance in Face Verification**. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* https://doi.org/10.1109/CVPR.2014.220
10. Schroff, F., Kalenichenko, D., & Philbin, J. (2015). **FaceNet: A Unified Embedding for Face Recognition and Clustering**. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* https://arxiv.org/abs/1503.03832
11. Ravanelli, M., Parcollet, T., & Bengio, Y. (2021). **SpeechBrain: A General-Purpose Speech Toolkit**. *arXiv preprint arXiv:2106.04624.* https://arxiv.org/abs/2106.04624