ECE/COE 1896

Senior Design

Facial Recognition Lock System with Anti-theft- Final Report

Team 2

Prepared By: Dane Krall

Rayan Hassan

Jake Smith

Table of Contents

Table of Figures

Table of Tables

1. Introduction

The market has grown more technologically advanced with cameras starting to be added to doorbells. For example, the Ring doorbell is a product that homeowners are starting to buy for their protection and safety from potential intruders. It uses facial recognition, which has become widely popular in lock devices due to its high accuracy and efficiency. The problem with modern lock devices is that most of them are not designed specifically to work under power outages. Additionally, scammers can use someone else's picture to break in, which makes them less secure. The proposed Facial Recognition Lock with Anti-Theft Lock System is designed to combat these problems and make sure the user is always protected.

The device comprises three main units: the hardware consisting of a microprocessor, camera, lock, power devices and sensors, a Computer Vision Unit responsible for verifying the user using a machine learning algorithm, and a website that helps the user to log in and receive warnings about any intruder.

The device takes a video as input and first checks whether the user is real or fake using blink detection. That is, if the input is a real, live person or an image placed in front of the camera to pass as the authorized user. A Liveness Detection Unit (LDU) is responsible for that. After the user is identified as "real", the Facial Recognition Unit (FRU) captures a live image and feeds it to the machine learning algorithm that will verify the user. If verified, the solenoid will unlock, otherwise it will stay locked and warnings will be sent to the real user through the website. Hardware components like LEDs will turn ON signaling a fraudulent or a successful attempt. An email will also be sent to the user to notify them of the entry.

The user will first be asked to login to the web app and take a video showing their face from different angles and facial expressions, which is stored in a dataset. The latter is fed to the Computer Vision Unit to be used as input to the FRU. In case of any intruder attack, warnings will be sent and stored in that dataset. The communication between the web app and the other units is based on HTTPS requests which offer lower memory usage.

The device functions under power outages to make sure users are always protected. For that, a backup energy source is used to make sure it is always powered. The backup source activates once input power is disconnected or lost through a battery bank. The battery bank remains active until its energy supply is depleted or input power is restored.

Testing was performed to determine the accuracy of the facial recognition module and the blink detection unit. Additionally, the bit transfer rate from the website to the Raspberry Pi was monitored. Lastly, proper voltages in the rectifier, longevity of the battery bank, and switching time between input sources was tested for the hardware. Overall, the main design requirements were satisfied through the results obtained. The system was able to perform well under different scenarios with facial recognition accuracy of 90% and higher. However, it had some limitations which were mainly caused by the camera used as well as other hardware components. Not to mention the limited training data available for the Machine Learning algorithm.

2. Background

Lock systems have become more reliant on facial recognition, as it is safe to use and provides accurate results. Many machine learning algorithms have been designed for that purpose. Most of them are based on Convolutional Neural Networks (CNNs), as they can process large amounts of data and produce highly accurate predicitons [1]. These models can learn complex and abstract features on their own which makes them extremely reliable. Other algorithms that are widely used are Support Vector Machine (SVM) and K-Nearest-Neighbors (KNN) [1]. The first one maps the data into an N-dimensional space after choosing the N most important features, which reduces the number of features considerably and therefore improves the computational time. That makes it easier for the data to be linearly separable. KNN on the other hand finds the distances between the data points to separate them into classes with the nearest ones being associated to the same class. All algorithms have been proven to produce accurate results in facial recognition. There are many pre-trained models out there that use them.

FaceNet is an example that uses CNN. It was developed by Google and is widely used for facial recognition. It is a 22 layers deep neural network that trains its output to be a 128-dimensional embedding [2]. The loss function used is called triplet loss that minimizes the distance between the anchor (the live image taken by the device) and positives (other stored images of the same user) on one hand, and maximizes the distance between the anchor and negatives (other training images of random individuals) on the other. The problem with pre-trained algorithms like this one is that they might not be fully compatible or transferable to any task since they are trained using a fixed dataset that may not resemble the intended one [3]. Therefore, problems like image sizing and number of classes can be encountered. Additionally, they are too complex to understand and debug.

The proposed approach uses a simple machine learning algorithm based on a SVM model. The latter can be trained using any dataset with any dimensions, since it resizes all the input and ensures consistency among all training and testing images. Also, it uses Principal Component Analysis (PCA). PCA is a dimensionality reduction method that “summarizes” the content in a large dataset into a smaller one that retains only the most important features and handles the data in a lower dimensional space. This considerably reduces the training time of the algorithm which makes it more practical and efficient.

As facial recognition is becoming more popular with the years, another problem arises. Spoofing is when a scammer disguises themselves as a known or trusted source. This can be done through holding a picture or a video of the real user in front of the camera. Many solutions have been implemented to solve that problem like the use of local binary patterns (LBPs), a method that studies the surrounding pixels of a point to analyze the textures of the image in more detail [4-5]. This method is usually combined with Difference of Gaussian (DoG) and SIFT to classify the input as live or spoof [5]. These methods enhance the edges of the noisy images and further detect and analyze features respectively. Although this approach produces a high level of accuracy, it performs poorly with specific, unconstrained scenarios because the features are sensitive to variations in lighting. Other anti-spoofing methods include CNN algorithms that fuse

different frames of the video taken to study their respective features and detect any changes in the edges of the picture to figure out if the scammer is holding a picture or not [4]. This method is too complex and might not always work if the picture held in front of the camera is still for example.

To solve that problem, blink detection will be used to make sure the user is a live person. This method consistently calculates an eye ratio that indicates how wide the eyes are opened or closed. Therefore, it can detect blinks every few seconds and consequently prevents spoofing using images of authorized users.

Additionally, there needs to be a way to interface with the locking system to make it easier for users to track fraudulent attempts and manually control the lock. Many lock systems have companion applications such as Lockin, SMONET, Veise, and ULTRALOQ. These take advantage of the ever growing ubiquity of these devices. This is further shown in Objective, a widely known tech blog. The amount of smartphone users in 2020 was expected to reach 3.8 billion [6]. Using technology users already have simplified the design and cut extra costs by removing the need to include these features on the lock itself.

The proposed device has features found in higher end models like a login system and security alerts. The lock uses HTTPS requests to transfer data which is better since it only allows connection when needed, which decreases the chance of cyber attacks [7].This also allows for the system to run with less latency by cutting out handshake procedures [7].

Finally, consistent power flow to activate a device to remain operational at all times is an issue that needs to be addressed. Power outages have become an issue due to the rise of electric grids switching over to renewable power that is not as stable compared to using fossil fuels or nuclear fission. As a result, not all utilities are able to consume electricity that is needed to keep items active at all times. For example, California has rolled out solar power extensively and has suffered through blackouts as a result of renewable energy [8]. The electrical grid will be seeing changes in the upcoming years as renewable energy starts to grow more prominent as fossil fuels continue to dwindle later this century. Renewable energy is not completely reliable yet as it is not as stable as using fossil fuels to create electricity.

A backup energy source needs to be accounted for. There are devices around the world that need to be powered up 24 hours a day and 7 days a week. In the event that input power is lost, a backup power supply needs to be incorporated into the system to keep components active [9]. The backup power supply will serve as a counter to the electrical grid as countries across the world begin to grow accustomed to alternative forms of generating electricity.

3. System Requirements

3.1 Functional Requirements

3.1.1 Liveness Detection Unit

- The system shall not start before a motion sensor is triggered
- The system shall take the live video as an input and produce a binary output of whether the user is real or fake
- The system shall detect the blinks of the user when occurred
- The system shall alert the user if spoofing is detected using LEDs and warnings sent to the authorized user
- The solenoid shall stay locked if spoofing is detected
- The whole process should not take more than a few minutes to execute and finish.

3.1.2 Facial Recognition Unit

- The system shall start right after the LDU and only if the user is identified as “real” (or “live”).
- The system shall take 3 inputs: the live images taken of the user, positives (other images of the user stored in the dataset) and negatives (images of random people).
- The output of the system shall be binary; class 0 (unverified) and class 1 (verified).
- The system shall notify the user if verified or not using LEDs and warnings sent to the authorized user
- The solenoid shall stay locked if the user is unverified and unlock if they are verified
- The system should not take more than a few minutes to verify the user.

3.1.3 Computer Vision Unit (LDU and FRU)

- The system shall take in a video of the authorized user from the web app
- The system shall capture 500 images from the video sent from the web app
- The system shall train the Machine Learning algorithm whenever a new user is the new authorized user
- The training should only happen once: whenever the device is first turned ON and only if there is a new authorized user to the system
- The system shall proceed to LDU right after the training of the SVM model is finished
- After training is finished, the system shall be continuously usable by any user who is trying to unlock the solenoid
- Training of the Machine Learning algorithm should not take more than a few minutes
- Capturing the 500 images from the web app video should not take more than two minutes

3.1.4 Data Transmission Unit

- The system shall only let authorized users access and modify the data being used in the locking system.
- The system shall transmit pictures taken from the app to the FRU to register valid users.
- The system shall transfer a signal to the solenoid lock system allowing it to be unlocked without use of the FRU in case of manual control mode through the web app.
- The system shall notify the user of any attempted entry.
- The system shall have a login system.

3.1.5 Power System Unit

- The system should take in 120VAC for input power from a wall outlet.
- The system should revert to backup power in the event that the input power source is disconnected or lost.
- The system should use the rectifier to step 120VAC down to 5VDC to power the module.
- The solenoid lock should not be unlocked for a period greater than 7 seconds.
- The motion sensor should detect whether a person approaches the recognition device.

3.2 Non-Functional Requirements

3.2.1 Liveness Detection Unit

- The algorithm should use the openCV library in Python.
- A monitor should show the video taken of the user at any time where the programmer can track the eye and head positions of the user
- The algorithm shall be robust and produce high accuracies of liveness detection

3.2.2 Facial Recognition Unit

- The algorithm should be written in Python and make use of libraries like scikit-learn and openCV.
- The system shall use a robust machine learning algorithm that produces high accuracies of 90% or more.
- All input images shall be of a consistent size (for example, 32 x 32 pixels).
- The data should be split into 80% for training and 20% for testing.

3.2.3 Computer Vision Unit (LDU and FRU)

- The system shall communicate output to a red LED, green LED and solenoid lock which should be configured in order to notify the user of the results (real, not real, verified, unverified)
- The system shall be connected to the server to communicate results to the authorized user (email warnings)
- The system shall be configured to take in input from a motion sensor to know whenever a new user wants to use it

- The system shall be entirely ran on the Raspberry Pi server and make use of a Picamera
- The system shall be entirely executable through only one command

3.2.4 Data Transmission Unit

- The web app shall be written in python using the Flask framework.
- None of the web app functions should take over 3 seconds to complete.
- The data should be sent using HTTPS requests to a central server.

3.2.5 Power System Unit

- The system should be wired and soldered in a way that prevents the user from facing any electrical hazards.
- The system should be at full power to increase longevity if input power is lost.

4. Design Constraints: Standards and Impacts

4.1 Conventional Constraints

4.1.1 Time

Time is one of the biggest limitations for this project since the team only has 12 weeks to come up with a fully functional design. Multiple measures will be taken to overcome that like planning a fixed and detailed schedule, laying out roles for each member and having good communication. This will allow the team to be in synchronization and consistent throughout the semester. Enough time must be allowed to complete the prototype and perform testing.

4.1.2 Manpower

The team will consist of only three members. Therefore dedication and communication are required from each member to ensure the project is successfully finished in time.

4.1.3 Financial

The budget for the project is 200 US dollars. Any component needed should be thoroughly discussed and researched, in order to ensure that the team stays within the budget and still gets all the necessary tools. The team should provide clear reasons and explanations in case the budget is to be surpassed.

4.2 Project Constraints

4.2.1 Computer Vision Related Constraints

These are limitations related to images and videos taken of the users.

- The system will not accommodate extremely dark or extremely light setups. Captured pictures should be clear enough for the algorithm to work accurately. The system is sensitive to lighting changes.
- The user should be standing still in front of the camera. If the input video/image is flu, the algorithm might produce wrong results.
- The algorithm will not be designed to specifically adapt to facial changes like beards, glasses, masks, etc. The training images will consist of a variety of face pictures but will not guarantee variety in facial features (of the same person).
- The system will only accommodate only one authorized user at a time
- The frame rate of the Picamera is extremely slow and therefore the blink detection might not work if the user is not blinking enough
- The user's face should be entirely placed in the center of the frame otherwise the system might not detect a face from the beginning and results might be inaccurate
- The system might lead to inaccurate results if the live video taken of the user includes other faces in the background.
- The authorized images used for training are captured from a laptop's camera and have different resolution than the live images captured using the PiCamera. This might lead to some inaccuracies in the FRU.

4.2.2 Data Transmission Constraints

- The system will not entertain the effects that a large number of users could have on transmission speeds
- While the system will be easy to use and made as simple as possible, basic technological literacy will be expected to use our product.
- The data transmission rate is determined by the Wi-Fi standard used by the router.

4.2.3 Power Constraints

- The system will not last more than two hours after being disconnected from input power. The available backup power is limited based on the maximum battery life capacity.
- The system will be wired in a way that prevents users from getting electrocuted in the event that the PCB is exposed by the user. The AC power flowing into the rectifier will need to be wired and soldered so that no stray wires are exposed.
- Government: The batteries should remain between -18 - 55 degrees Celsius (-0.4 - 131 degrees Fahrenheit) [28].

4.3 Impacts in Non-Technical Contexts

4.3.1 Environmental

The backup power supply may have environmental issues that are not limited to carbon emissions, air pollution, noise pollution, or resource consumption (“Backup Systems”). An uninterruptible power supply (UPS) will run into noise pollution as it starts up when input power is lost. In addition, the batteries inside of a UPS will need to be replaced over time.

4.3.2 Public Health

The device must be wired so that the National Electric Code (NEC) is followed for the respective location. In the United States, each state follows a different edition of the NEC.

4.3.3 Global, Cultural and Societal

Increased ubiquity and accessibility of lock systems can be a deterrent to those who would perform thefts. This could decrease the rate of spontaneous thefts that occur in society making a safer environment for individuals.

4.3.4 Diversity, Equity and Inclusion

The device can be used by anyone, from any different age, race, ethnicity, etc. The application will make it accessible and easy to use for anyone. Steps will be clear for first time users on how to login and manually control the lock. This will ensure that even people with accessibility issues can easily use and understand it.

The computer vision unit and hardware will walk the user through a bunch of interactive steps to ensure authentication (whether the user is real or fake) and verification. These will be mainly based on LEDs and a vibration sensor. This will ensure that even people with hearing problems can use it. Additionally, the system won’t require any physical movement by the user, all they have to do is position their face in front of the camera. This will prevent any issues for people with certain physical disabilities.

4.3.5 Welfare and Safety

The device will prevent spoofing; that is, assuming someone else’s identity by placing a picture or a video in front of the camera. The system will detect it and send warnings to the real user. Additionally, the lock won’t unlock unless the user is verified, which will be based on a robust machine learning algorithm for facial recognition. That way users will be safe and protected against any intruder attack.

4.3.6 Economic

If the anti-theft system was installed to serve as a lock system for a business or corporation, the business would benefit by not having to deal with robberies. The use of the system would allow businesses to save money in the event they have to deal with a robbery in a power outage.

5. Summary of Design

The device consists of an anti-theft system based on facial recognition. It comprises three main sections, a website application that helps users to log in or manual control the lock, a computer vision unit responsible for verifying the user as well as preventing spoofing and a hardware unit which holds the system together and includes sensors, power devices and a camera.

5.1 System Description

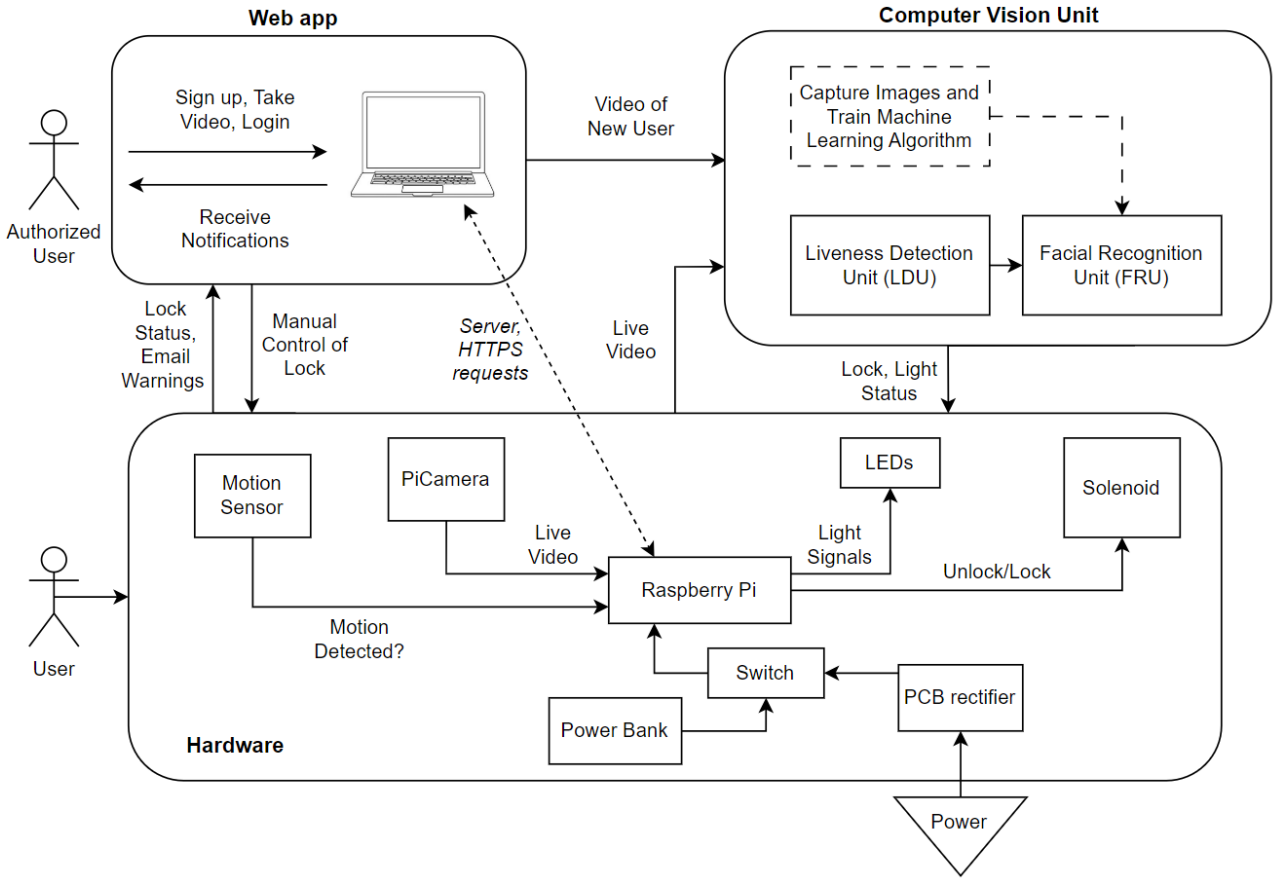

Figure 1: System Interaction Diagram

The figure above shows all the interactions between the three modules and the information communicated. A user would first need to create an account through the web app and upload a

one minute video of their face capturing different angles and facial expressions. Once the video is submitted, it will be sent to the Raspberry Pi server through HTTPS requests. Once the device is turned ON, 500 frames will be captured from the video and later used to train the Machine Learning algorithm. Once training is finished, any user can start using the device and try unlocking the solenoid. One only has to touch the motion sensor, which will be triggered and cause the PiCamera to turn ON. The user should then position their head in the center of the frame and pass the blink detection check (LDU). If blink is not detected, the red LED will turn ON and the solenoid stays locked. However, if blink is detected, the user will be identified as “real” and the FRU will proceed in capturing 20 images from the live video which will be fed to the Machine Learning model for predictions. If the majority of those images are classified as “authorized” then the user is verified, the solenoid will unlock and the green LED will turn ON. Else, the user will be identified as “unauthorized” and the solenoid will stay locked while the red LED turns ON. In both cases, emails will be sent to the authorized user notifying them about the entry. Other users can try to unlock the solenoid after that following the same procedure just by triggering the motion sensor. Additionally, the authorized user can manually unlock the solenoid through the website. The device is powered with the help of a rectifier that controls the voltage and current coming from the wall outlet, and switches to a power bank in case of a power outage.

5.2 Software Design

5.2.1 Web App

The Web App consists of a login system where an authorized user can create or login to an account in order to interact with their lock system. From the home page the user is able to unlock the button directly from the website in cases where they may want to remotely open the lock for someone. This is also a useful feature in case the system malfunctions. The Web App also allows for a user to press a button, access the webcam of their device and take a one minute video that is sent to the Computer Vision Unit for training. The figures below show the different pages of the web app and their functionalities.

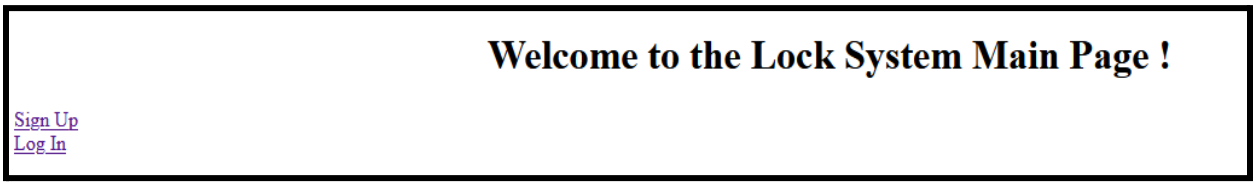

Figure 2: Website Main Page

Figure 3: Login Page

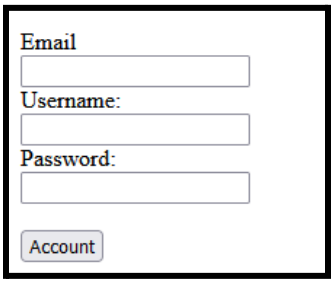

Figure 4: Sign Up Page

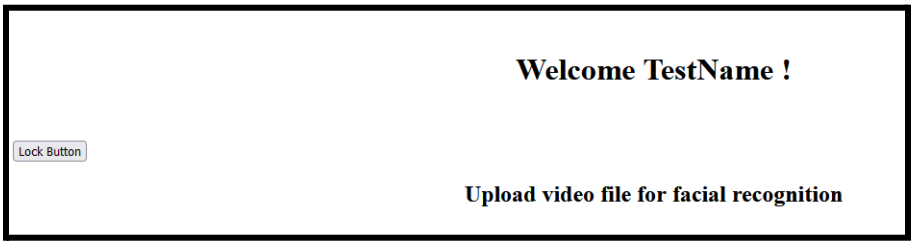

Figure 5: Homepage Lock Button

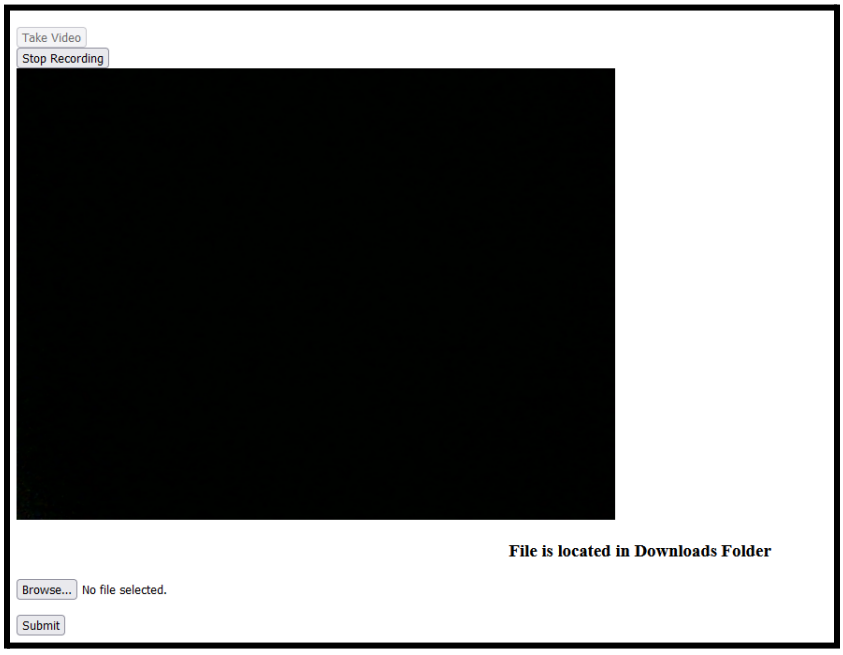

Figure 6: Homepage Video Creation/Upload

5.2.2 Data Transmission and Communication

The Transmission of data is facilitated by the use of a server. The latter stores the user's account information to determine valid users of the lock system. It also encrypts their passwords using SHA-256 encryption to keep the information secure incase of a data breach.The converted password is then converted again into a hex string to be compatible with the json format. The server allows for the video from the user to be sent to the server and saved on the Raspberry Pi. It also facilitates the unlocking feature of the web app. This manual control can only be done when the solenoid is originally locked (lock status 0) and when motion is not detected. When triggered, a value 1 is XORed with the lock status value (0 in this case) to produce a final lock status of 1, allowing the solenoid to unlock. After 10 seconds, the value is XORed again with a 1 to return the solenoid to its original state. When the server is active, the Raspberry Pi also has access to the users' Email and sends them security alerts whenever a new attempt at unlocking the solenoid is detected. The figure below shows the encryption performed on the user's password as well as the information communicated between the servers.

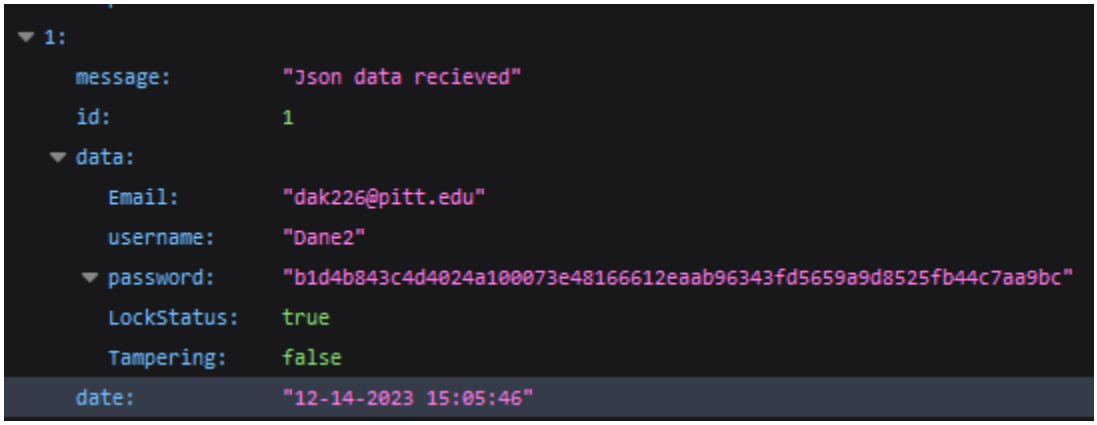

Figure 7: Password Encrypted on Server

5.2.3 Computer Vision Unit

The Computer Vision Unit consists of two main parts, a Liveness Detection Unit (LDU) and a Facial Recognition Unit (FRU) as mentioned before. When the device is first turned ON, if a new authorized user signed up through the web app, the video received will be split into 500 frames/images that will be stored in a dataset of training images. These images will correspond to class 1 (or class "authorized") for training. The dataset also contains 5000 images of random people found online [34], which correspond to class 0 (or "unauthorized"). Therefore, in total the dataset contains 5500 images which are split into 80% training and 20% testing. Note that the "unauthorized" class contains a significantly higher amount of pictures in order to generalize to any unauthorized user who tries to break in. The figure below shows the function calls every time a user tries to use the system. An explanation of each main function is provided afterwards.

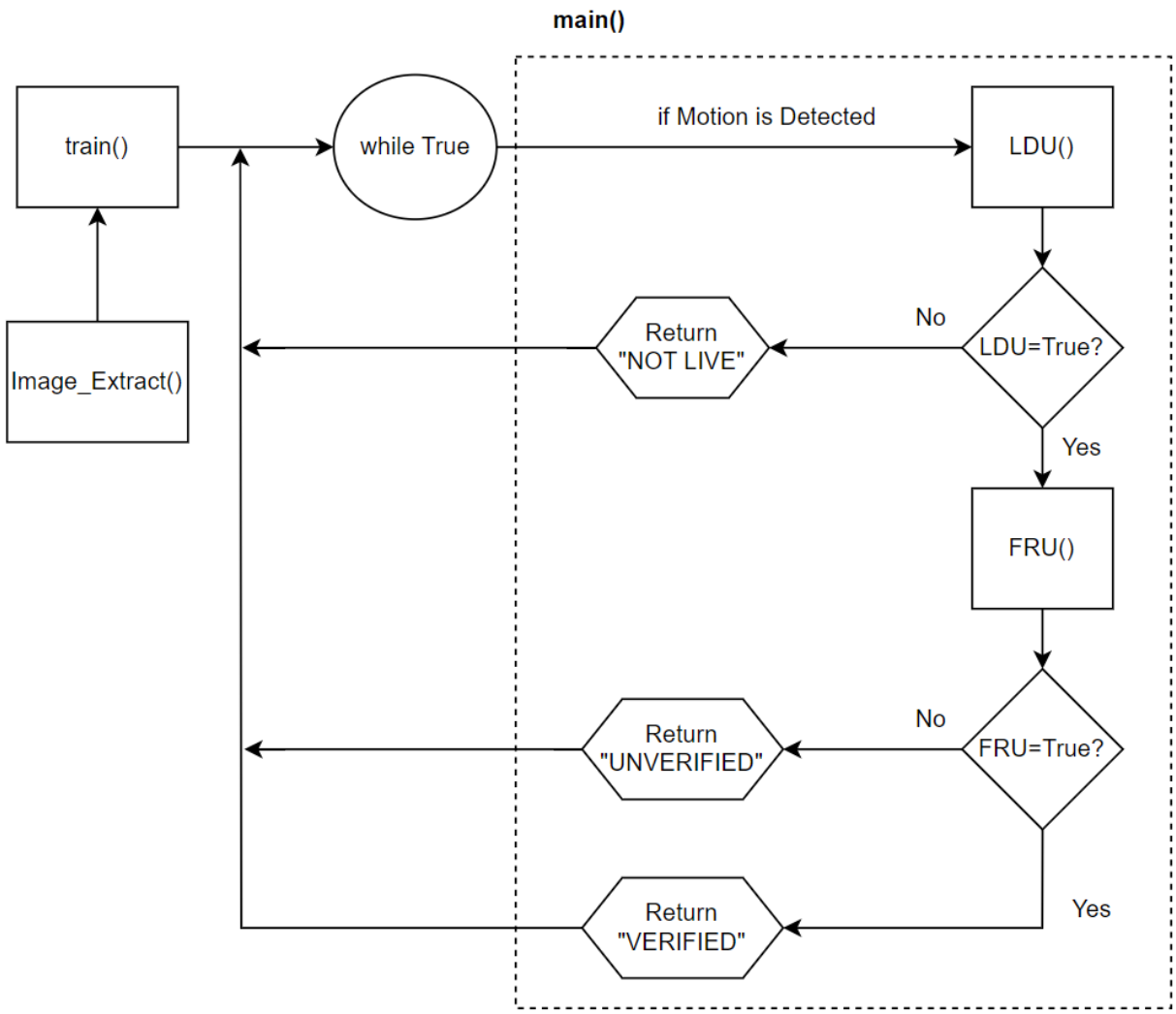

Figure 8: Main Function Calls for the Computer Vision Unit

Image_Extract()

This function reads the video received by the web app and captures 500 frames reducing them to face images (without the background). For that, Haarcascade implementation is used to detect edges, lines and directions of the picture which helps in detecting the face. For instance, it can easily detect sudden changes in lighting, which helps in knowing where the boundaries are between the background and the face. The haarcascade_frontalface_default.xml file is taken from an open source library (openCV) [12]. After that, the 500 images are saved in a “class1” folder which is itself stored in the training images’ folder.

train()

This function implements the Support Vector Machine algorithm used for facial recognition. First the images stored are read and each converted into 100x100 grayscale images. This is done to

reduce the computational training time. After that they are converted into matrices and then each flattened into one large vector. The data is therefore transformed into a 5500x10000 matrix that contains all 5500 images (each row corresponding to an image) and 10000 features (since the images are converted into 100x100 matrices like mentioned previously). As this is a large number of features, dimensionality reduction is used to keep only the relevant ones. PCA is used and only 100 features are retained, which capture the most variance. This reduces the training time considerably. After that, the SVM model can be trained. A linear kernel is used with a parameter of C=0.001 after hypertuning. Once the model is trained, it is saved as a .joblib file, which can be used later to make live predictions instead of having to train the algorithm everytime.

LDU()

This function implements the blink detection. Each eye is represented by 6 points (p1,...,p6) with (x,y) coordinates starting from the left-corner of the eye and working its way clockwise around the rest of the region. The following figure depicts that:

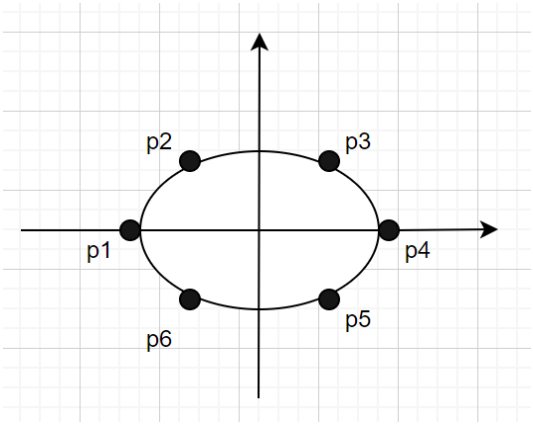

Figure 9: Detection of Points Around The Eye

The circle/oval in that figure represents the eye. After that, an Eye Aspect Ratio (EAR) is calculated [15]:

$$EAR = \frac{||p2-p6|| + ||p3-p5||}{2||p1-p4||}$$

The EAR is constant when the eye is opened but will decrease close to zero when it is closed. Another way to view it is when the eye is closed, the 6 points will all get closer to the horizontal x-axis. This ratio is constantly updating until a blink is detected. However, if none is detected after 10 seconds, a warning is sent and the user is identified as “fake”. The algorithm also uses openCV library and the live frames captured by the camera. This function returns “False” if the user is fake and “True” if they are real.

The PiCamera is also configured in this function in order to turn ON and take the live video of the user whenever the motion sensor is triggered. In order to do that, the picamera2 library is used. A variable PiCam = Picamera2() is created to capture frames using the capture_array()

function. A while loop keeps updating until the LDU check is done. Once this happens and only if the user is identified as “real”, the function automatically captures 20 frames and saves them in the working directory. These images are used later by the FRU.

FRU()

This function is used for facial recognition right after the LDU check is passed. It takes each of the 20 live images captured and passes each one of them through the SVM model for predictions (using the .joblib file). If the majority of the images are classified as 0, the user is unverified and the function returns “False”, else they are verified and it returns “True”.

Configuring the Hardware sensors

The main code that is executed contains a main() function that contains all the functions explained before. This function also configures the sensors used to interact with the user. The LEDs, the solenoid lock and the motion sensor are all connected to the GPIO pins of the Raspberry Pi. The code implemented to turn them ON and OFF uses the gpiozero library. Specifically, the functions LED(), OutputDevice() and MotionSensor() are used respectively for the LEDs, the solenoid and the motion sensor. The GPIO pins are first specified and stored in variables. For instance, “green_LED = LED(17)” means that the 17th GPIO pin on the Raspberry Pi is connected to the green LED. After that, depending on the results of FRU() and LDU(), the sensors are either turned ON or OFF. The following diagram shows how and when the sensors are triggered.

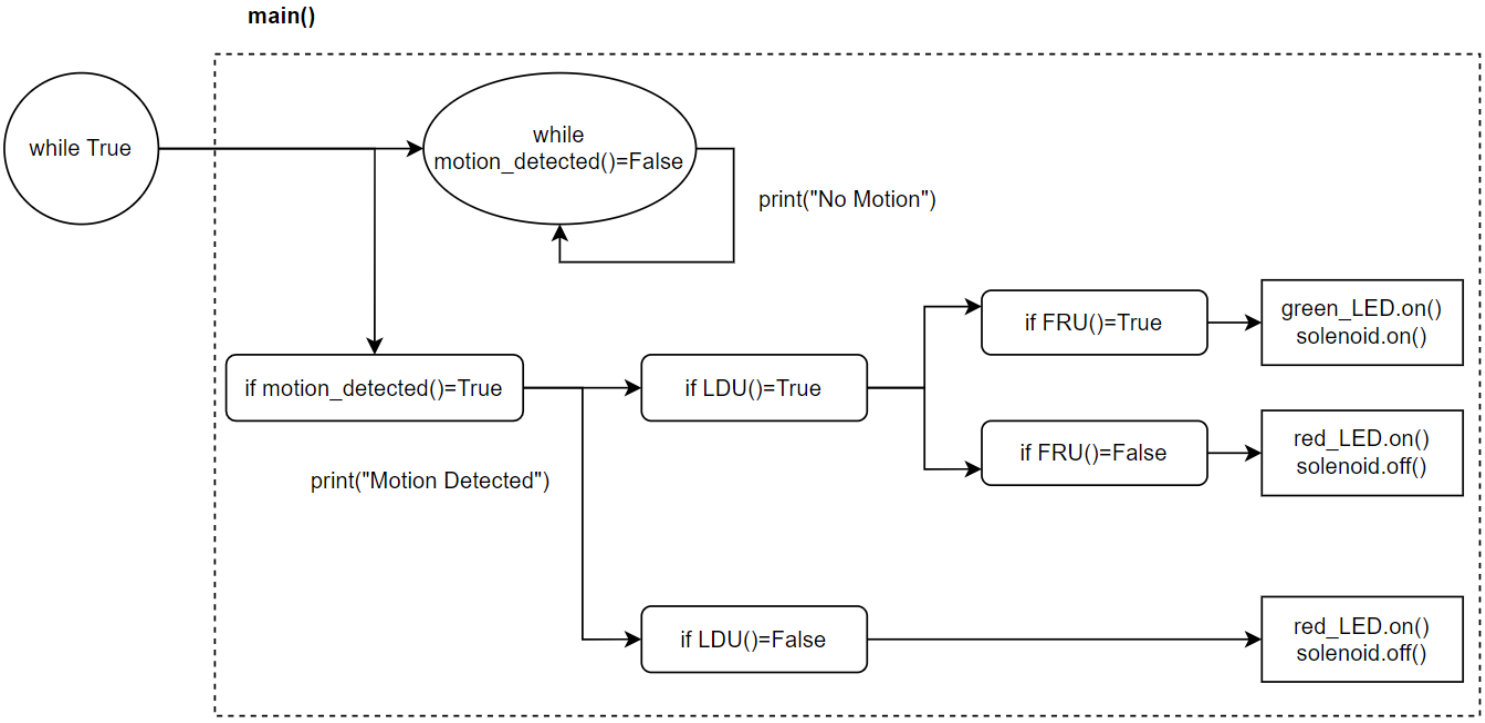

Figure 10: Software Configuration of Sensor, Solenoid, and LEDs

5.3 Hardware Design

There were different approaches that were considered for the use of hardware. A Raspberry Pi was used as the microprocessor for the project. It served as a better source to use in place of an Arduino microcontroller in terms of camera connectivity. As a result, the Raspberry Pi camera was utilized due to its compatibility with the chosen microprocessor. A standard 5V solenoid lock was used and was able to connect to the 5V power pin on the Raspberry Pi. To avoid the need of having to develop a large circuit just for the solenoid, an SPST relay was used in conjunction with the GPIO pins on the Raspberry Pi. Ideas were considered of making an external circuit to power the lock, but the relay gave an easier approach. Also, a motion sensor was installed to start up the camera rather than just using an ordinary SPST button. In terms of generating successful responses from the programming, LEDs were utilized over an LCD screen. The LEDs are an easier form of successful or unsuccessful unlock attempts and utilize less power than what an LCD screen would require.

One of the key modules that was decided upon was the development of a voltage rectifier. This was developed to avoid the need for a trivial plug that could be plugged into the Raspberry Pi off the shelf. Another module that was developed was the use of a battery bank. Other options were considered such as an uninterruptible power supply (UPS) along with the creation of a rechargeable power supply. The UPS was too trivial without any sort of new design. In addition, the rechargeable power supply was too pricey with not enough time to complete given one semester. The battery bank that was developed served as a backup power supply if input power was disconnected at any point.

As a whole, the entire system is meant to supply power at all times between input power from the rectifier and the battery bank. Also, the system is supposed to work with the GPIO pins coder from the software modules.

5.3.1 Voltage Rectifier

The voltage regulator is meant to take a 120VAC signal from wall power and step it down to 5VDC to power the Raspberry Pi used for completing the project. The design starts by taking a 10:1 transformer that steps down the 120VAC wall power down to 12VAC. The transformer output is then fed into the voltage rectifier. A bridge rectifier was developed for converting from AC to DC voltage. In addition, a current divider was added in to control the amount of heat losses that the regulator experienced. A 5V/3A voltage regulator was used to control the voltage and current going into the Raspberry Pi. Since the Raspberry Pi takes a USB-C input, a USB-C breakout connector was connected to the rectifier output to power up the Raspberry Pi.

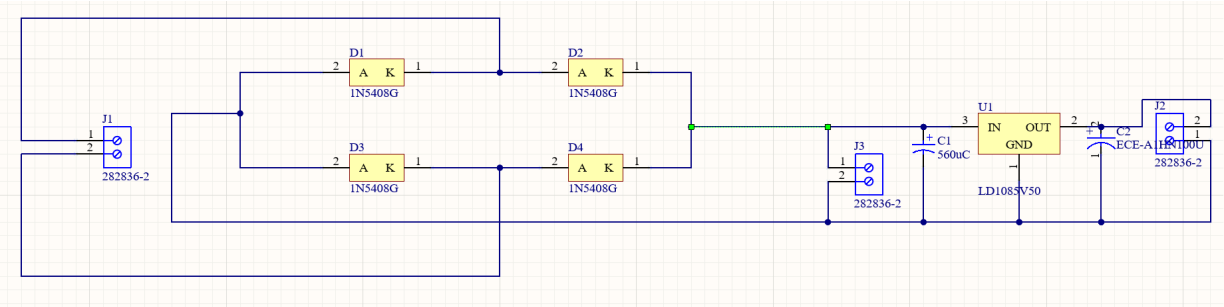

Figure 11: Voltage Rectifier Schematic

5.3.2 Battery Bank

A battery bank was designed and developed as a backup power source if power from the voltage rectifier is lost. The design provides an 18VDC power source with a capacity of about 2 Ah. An image of the battery bank is shown below. The battery bank was fed into a 5V/3A buck converter to provide consistent power into the Raspberry Pi.The buck converter steps down the voltage while providing a current boost. Below is an image of the developed battery bank.

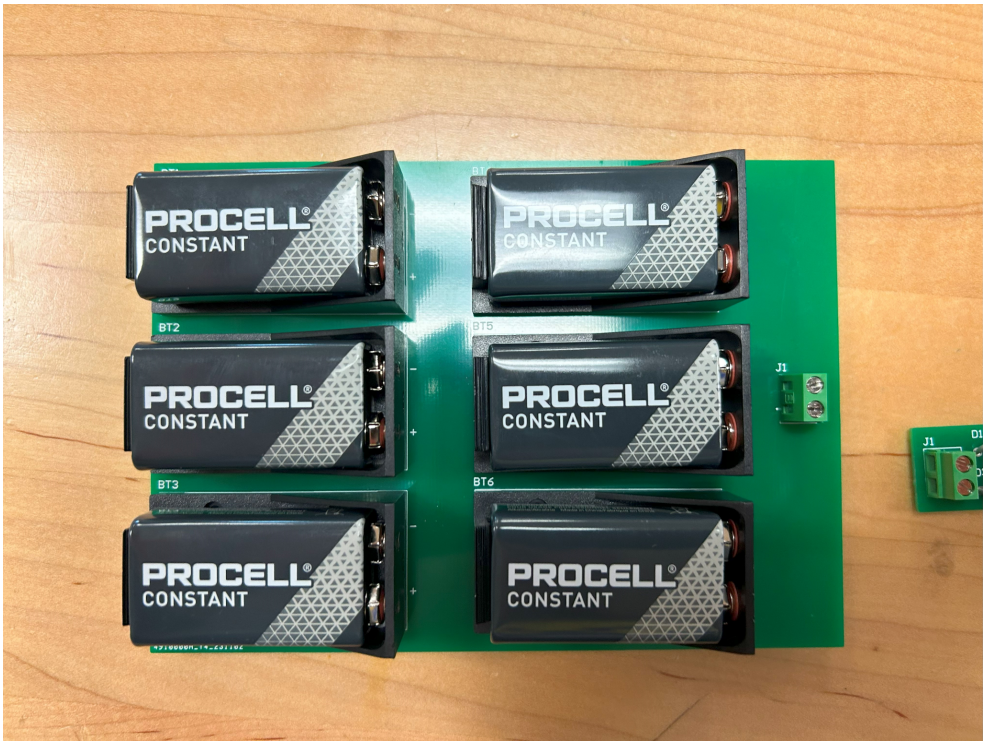

Figure 12: Image of Battery Bank

5.3.3 Input Switch

A DPDT relay was used to switch between input power that is energized through the voltage rectifier and backup power from the battery bank. The relay switch is changed once power from the rectifier is detected. The battery bank is in the normally closed condition while the voltage regulator is in the normally open condition. Regardless of if the rectifier or battery bank is supplying power, electricity will be supplied to the Raspberry Pi. Below is an image of the PCB.

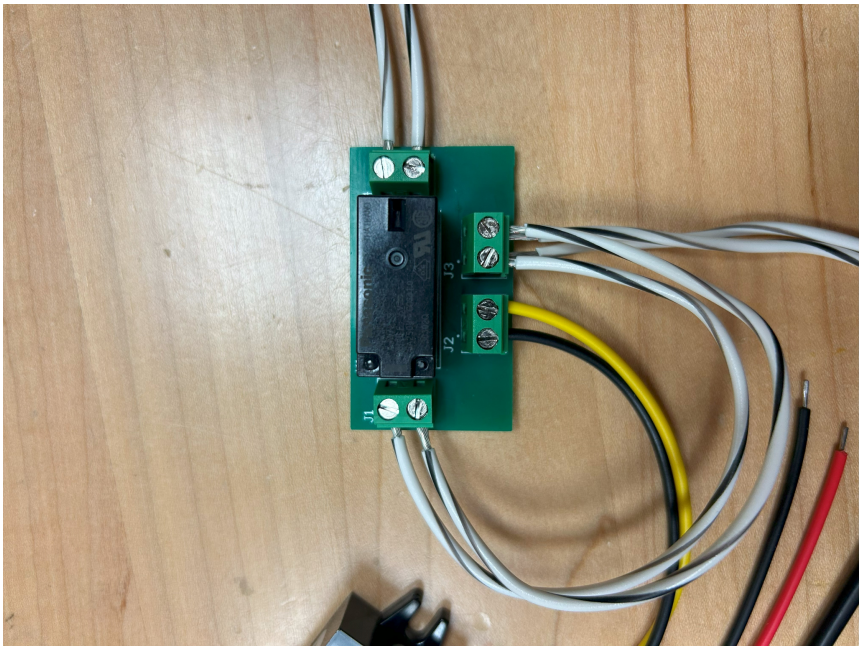

Figure 13: Input Switch between Rectifier and Battery Bank

5.3.4 LED Array

To avoid having to use a breadboard, a PCB was developed to hold the LEDs for the various configurations set up from the programming that was set up from a separate module. A red LED is connected to the GPIO pins to signify if there is an unauthorized user attempting to use the device and/or there is not a real person at the camera. A green LED is also connected to the GPIO pins on the Raspberry Pi and will light up if a real and authorized user is detected by the Raspberry Pi camera. Below is an image of what the PCB looks like.

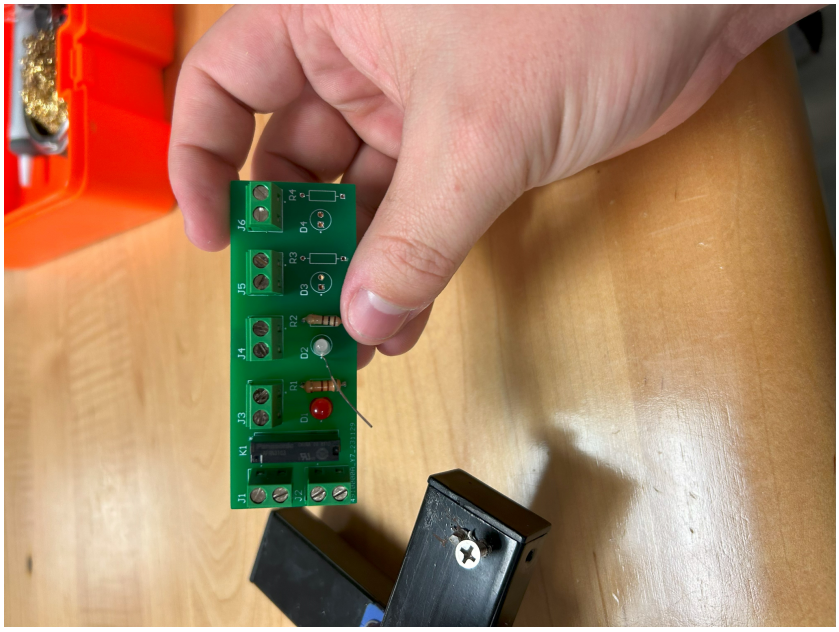

Figure 14: LEDs with SPST Relay for the Solenoid Lock

5.3.5 Motion Sensor and Solenoid Lock

The motion sensor is connected to the GPIO pins and sends a signal if motion is detected. The sensor is powered by connecting to a 5V pin on the Raspberry Pi. In addition, the solenoid lock is connected to an SPST relay. The lock is wired to the switch, but is controlled by the GPIO pin the relay's coil is connected to to open and close the switch. The image in section 5.3.4 shows the PCB that the relay is a part of since it's in the same image as the LEDs.

6. Testing, Data Analysis and Results

6.1 Web App and Data Transmission Testing and Results

The testing for the Data Transmission and Web App is twofold. A ping testing is first performed followed by the calculation of throughput of data through the web app.

6.1.1 Ping Testing

For ping testing, a python script was used to record the round-trip time used to make HTTPS requests over 100 iterations. This measures how long the server takes to respond to a request from a client. From the results shown in Figure 15, the average round-trip time was 5.62ms, which represents the latency of the HTTPS requests used in the communication of the lock system. Gabriel from IOFlood, a tech blog and server provider, states that 100ms is considered acceptable for website browsing and file downloading, 200ms being the upper bound of tolerance [20]. The ping of the Web App is small due to the server running on a Raspberry Pi being at a relatively short distance from the user. Even with the few outliers gathered in the data, the ping never increases over the target amount. The small amount of time that these requests take satisfy the requirement stating that they shouldn't take more than 3 seconds to execute.

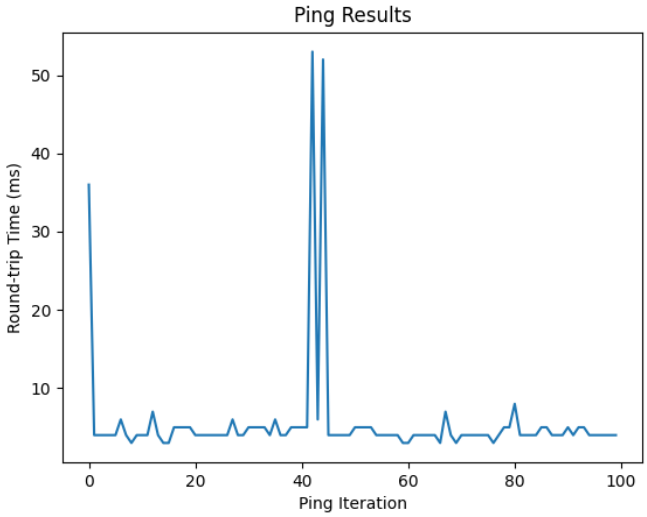

Figure 15: Ping Testing Results

6.1.2 Throughput Testing

Throughput testing was done by sending videos through the Web App of different sizes and checking how long it took for the server to receive them. Ten videos of various sizes were taken and sent through the Web App to calculate an average throughput as seen in Table 1 below.

Trial	Time(s)	Data(KB)	Throughput(kB/s)	Avg Throughput (kB/s)
1	1	5240	5240	5437.510931
2	1	5906	5906	
3	13	65106	5008.153846	
4	17	100089	5887.588235	
5	95	563005	5926.368421	
6	60	363034	6050.566667	
7	22	130379	5926.318182	
8	77	436824	5673.038961	
9	8	31511	3938.875	
10	10	48182	4818.2	

Table 1: Trials to Determine Throughput

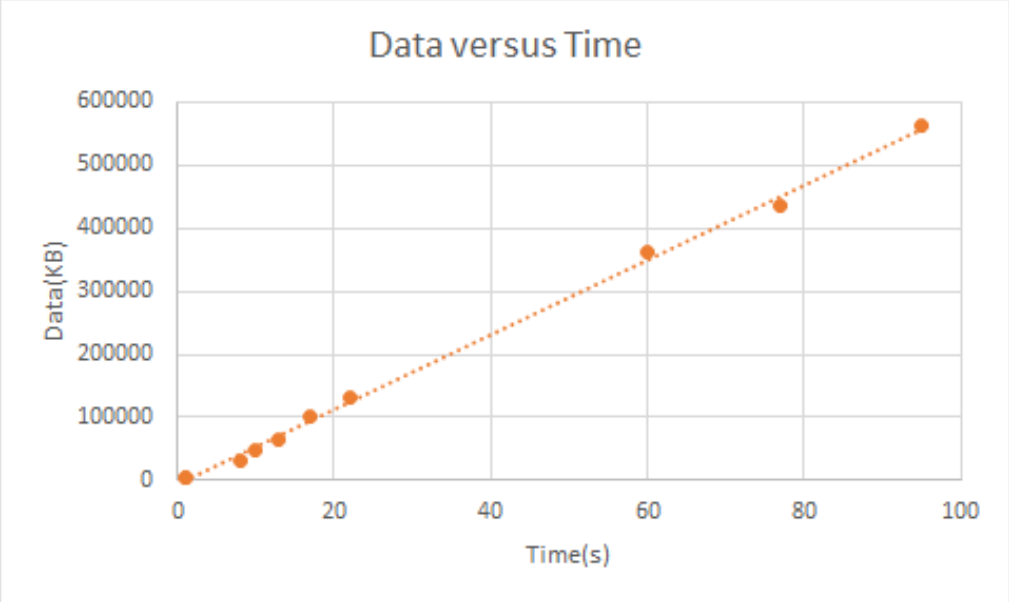

Figure 16: Graph of Consistent Throughput

From the data collected it is seen that an average throughput of 54327.5 KB/s was found with a range of 2111.69 KB/s. The typical video sent through the Web App is around the file size of 5000 KB-6000 KB which will send in a range of 0.92s-1.10s keeping in line with the three second range for a good user experience. Also when graphing the data versus time as seen in Figure 16, it can be concluded that the throughput remains relatively constant which helps remain in this time range.

6.2 Computer Vision Testing and Results

The testing for the Computer Vision Unit comprised many stages. Before adopting the SVM model for the FRU, a Convolutional Neural Network (CNN) was implemented and tested. Although this model did not end up being used in the final version of the system due to incompatibility reasons with the Raspberry Pi 4, some results of its testing have been collected and can be used for future work as it offers a more robust solution for facial recognition.

6.2.1 FRU Testing

CNN implementation and Testing

The CNN model has 3 convolution layers with 32, 64 and 128 neurons respectively and 3x3 kernels. Each convolution layer is followed by a maxPooling layer with a 2x2 kernel. After the 3rd layer the image is flattened into one large vector and passed through 3 dense layers with 128, 84 and 2 units respectively. The last 2 units correspond to the two classes “authorized” and “unauthorized”. The activation function for each layer is a ReLU function, and the one used for the last (predictive) layer is a Softmax function. Additionally, with every convolution and dense layer, an L2 regularization was added to prevent overfitting, which is when the algorithm memorizes the training data so well that it fails to generalize to new, unseen data (the testing data). The images fed to the model were also 32x32x1, which meant that they were grayscale images (with only 1 channel), in order to decrease the training time.

To test the model, the data was split into 80% training and 20% testing. 10% of the training data was used as validation data, in order to hypertune the parameters. The number of epochs chosen initially was 5.The following graphs show the accuracies obtained after 5 epochs for regularization parameter k equal to 0.1 and 0.0001.

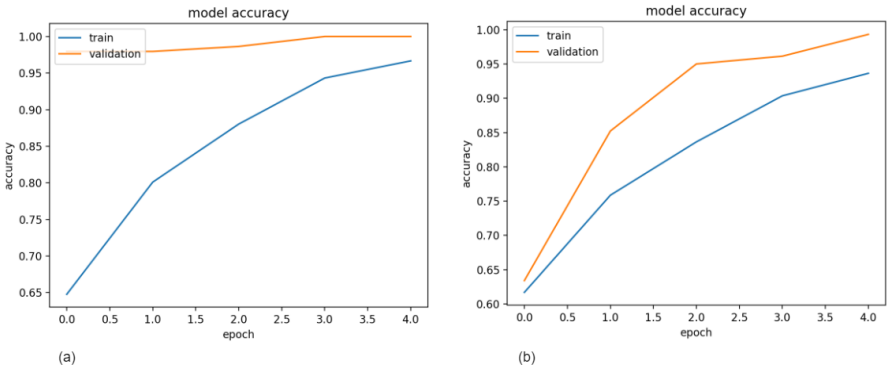

Figure 17: Accuracy Over 5 Epochs with (a) k=0.1 (b) k=0.0001

These results show that the model is overfitting since training accuracy is lower than the validation one and that the learning rate (rate at which the model learns new features) is too high. The fact that the regularization parameters had no effect on overfitting meant that the issue is in the architecture of the model itself.

Multiple methods have been added to resolve these issues. Starting with Dropout layers after each convolutional layer, which deactivate certain neurons in order to prevent the algorithm from memorizing the data and therefore overfitting. Also, Batch Normalization was used to rescale the image (matrices) after each stage and ensure consistency throughout each epoch. The number of epochs was increased as well to help better visualize the model's performance as well as give it more time to learn features.

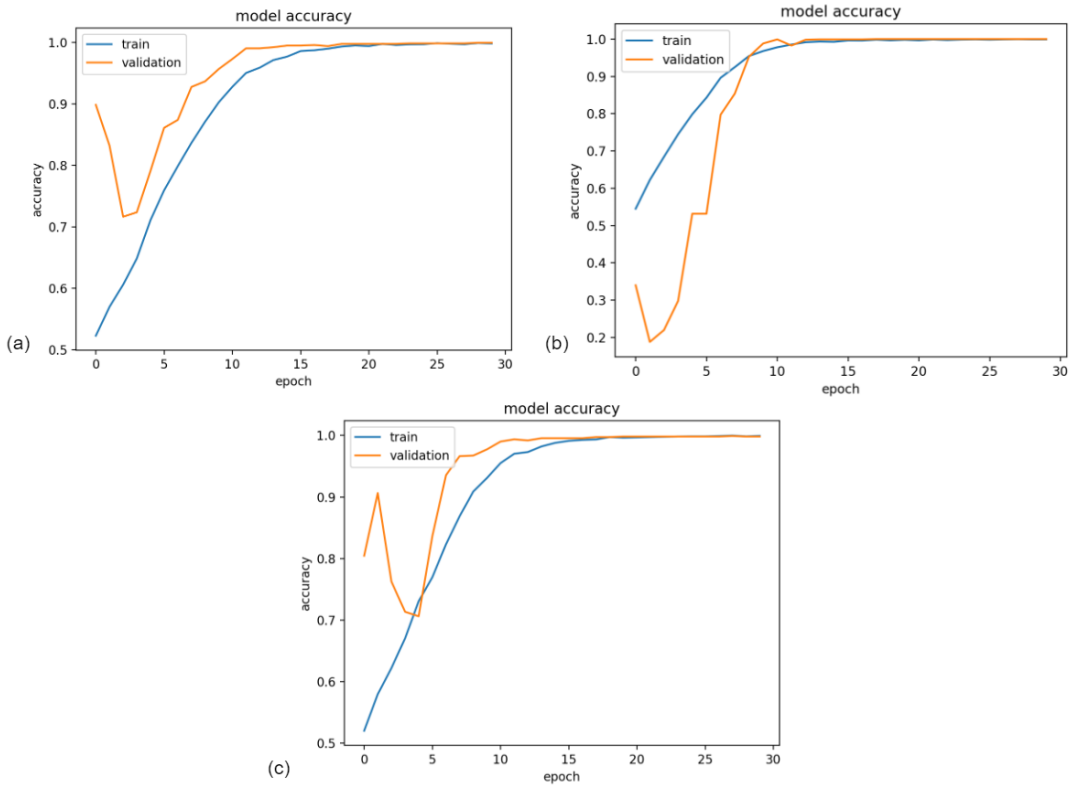

Figure 18: Accuracy over 20 epochs with (a) C=0.2 (b) C=0.3 (c) C=0.6

These adjustments seemed to have fixed the learning rate issue, however the model was still overfitting. The graphs in Figure 18 above differ with different dropout rates C: 0.2, 0.3 and 0.6 for figure (a), (b) and (c) respectively. All of them had a regularization parameter k of 0.001. This shows that the dropout parameter is not the problem since there is no change in overfitting. Multiple attempts at fixing the issue and tests with different combinations of dropout and regularization parameters were conducted.

However, it became clear that the problem was external and had to do with either the images used or the model's architecture. Therefore, instead of having 3 dense layers, only two were kept, a 84-unit layer and the 2-unit layer at the end. On another hand, the compiler used

throughout all the previous trials adopted Adaptive Moment Estimation as optimizer (Adam). This time, another optimizer was tested using Stochastic Gradient Descent (SGD). Finally, the training data was changed and 500 extra images of the authorized user (so 1000 in total) were generated using Data Augmentation techniques with different rotations, shifts, and zoom and shear ranges. This was done to create more variety in the authorized user's images since it was very homogenous compared to the other 5000 unauthorized images. The image size was also changed to (64,64,3), which meant that the images were now colored and larger, and more features/details could be detected by the algorithm. The figure below shows the first test with this new model architecture.

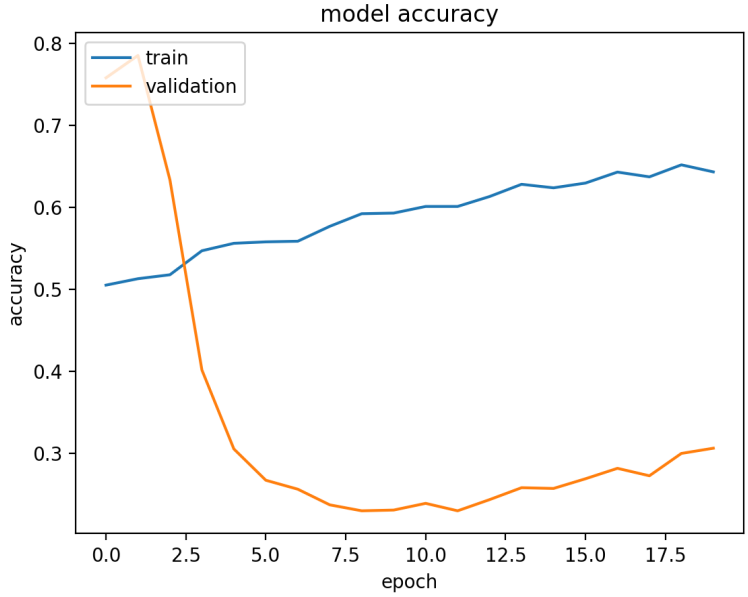

Figure 19: Accuracy Over 20 Epochs with 64x64x3 Images

As can be seen, the validation accuracy was lower than the training one, which meant that the model was learning features at a proper rate and without overfitting. However, both accuracies were extremely low throughout the 20 epochs. Therefore, a significantly higher number of epochs had to be added. This is because the images were much larger now and the model needed more time to learn features and train.

However, the time taken to train the model with 20 epochs alone was around 30 minutes, which meant it was not efficient nor practical to try with more epochs. Different image sizes were tested afterwards. The figures below show the results obtained using these changes for (64,64,1), (32,32,3) and (32,32,1) images respectively and over 20 epochs.

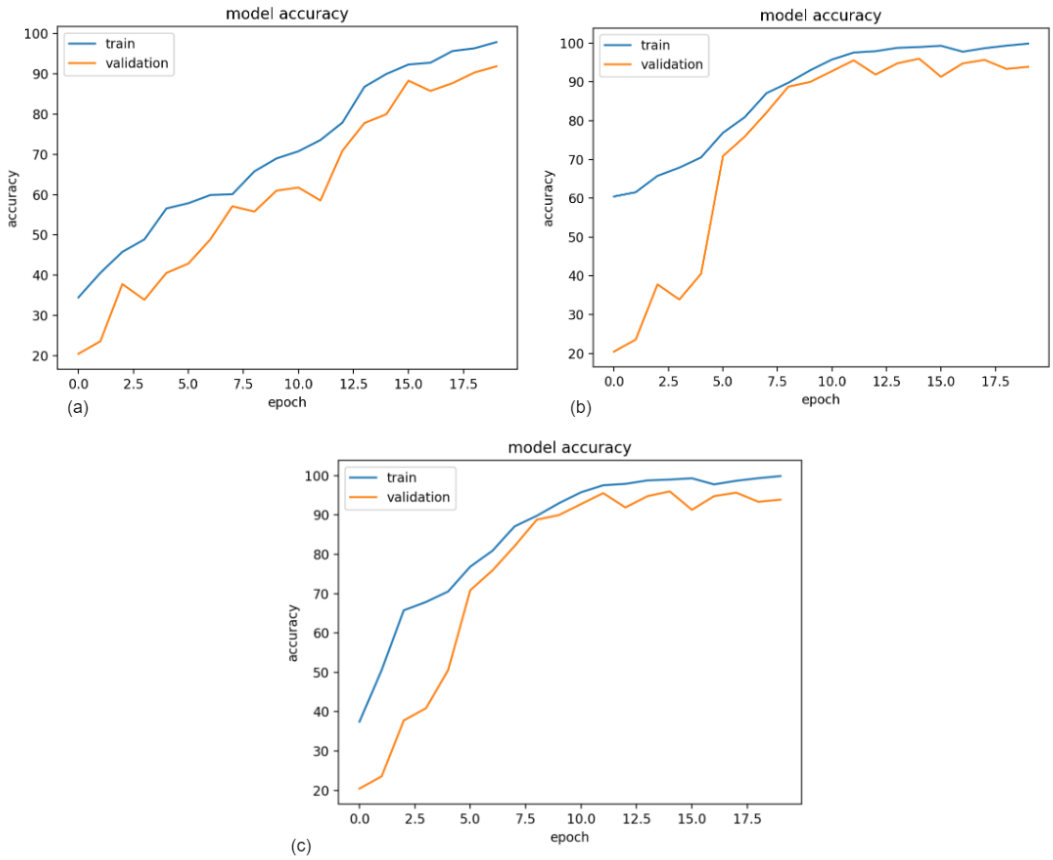

Figure 20: Accuracy Over 20 Epochs with Image Sizes (a) 64x64x1 (b) 32x32x3 (c) 32x32x1

As seen in the figures, these changes seemed to have resolved the overfitting problem and reasonably looking graphs were able to be generated. The CNN was learning at a proper rate and the validation accuracy was constantly lower than the training one. At this point, it was a matter of time. Therefore, multiple tests were conducted using different image sizes and hypertuning the parameters. The average training time and testing accuracies were collected for each in the table below. Note that a fixed amount of epochs was used (20 epochs) in order to compare the performances properly.

Image size	Average Training Time	Average Testing Accuracy
(64,64,3)	30 minutes	35%
(64,64,1)	25 minutes	88%
(32,32,3)	6 minutes	91.6%
(32,32,1)	3 minutes	90.3%

Table 2: Average Training Time and Accuracy for Different Image Sizes

The table clearly indicates a huge difference in training time for images of larger sizes. The testing accuracy using those images is lower than the other cases which might seem illogical. However, since these images are bigger in dimensions, the model would need more time (epochs) to learn their features. Given that the model was trained over 20 epochs for all cases, it would make sense for the accuracy using the larger images to be lower. It would have been higher had the algorithm run for many more epochs.

After all these tests (and even more, using various combinations of Dropout and regularization parameters to find which ones were optimal), the CNN was going to take in 32x32x1 images and use the last architecture previously mentioned.

CNN Testing on Raspberry Pi

After the CNN model was completed, it had to be transferred onto the Raspberry Pi and used for live predictions of images taken from the PiCamera. The CNN model uses Keras and Tensorflow libraries. Installing these libraries required the installation of wheels specific to the architecture and python version on the Pi. The SD card that was first purchased for the Raspberry Pi was a 64-bit card. It had an architecture armv7l with a python version 2.7.8. The latter was not suitable for the Tensorflow library and had to be upgraded. Although this was able to be done and the Python version was upgraded to 3.9.12, no wheel could work with the arm7l architecture.

Another SD card was purchased after that, a 32-bit card with a aarch64 architecture. A virtual environment was created where the python version was also upgraded to a 3.9.12 and a wheel was finally compatible with the Raspberry Pi: tensorflow-2.8.0-cp39-none-linux_aarch64.whl [29]

The problem this time came with installing the OpenCV library with Tensorflow, as it requires a different python version and Pi architecture. Numerous attempts at installing both were done by creating other virtual environments and working with different Python versions, however the two libraries could not be imported together in the same code. This was a huge problem since these libraries are used together to capture the live face images (openCV) and for facial recognition (Tensorflow). After all these trials, a decision had to be made on which library to keep. OpenCV was vital for this project, so Tensorflow was abandoned and the FRU had to be implemented using a different model. A Support Vector Machine (SVM) that uses sklearn library (which could easily be installed on the Pi) was therefore adopted, which turned out even better time-wise, since training it only took a few seconds.

SVM model Testing

The pictures used for the SVM model had 10000 features as mentioned before. This is a huge amount of features and therefore dimensionality reduction was needed. The number of features that retain the most variance (that are the most important at representing the images) had to be found. For that, 100 components were tested to figure out how many were needed.

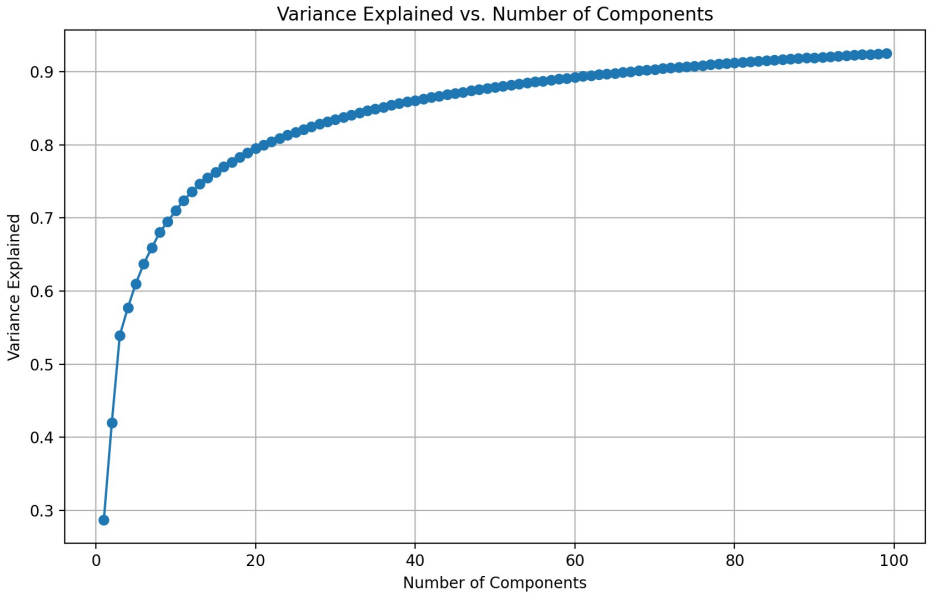

Figure 21: Variance vs. Number of Components

The figure above shows that 100 components retain around 93% variance, as opposed to only 20 components for instance which retain 80%. This meant that 100 components/features were enough to represent the images. This is a huge reduction from 10000 features which saves a considerate amount of training time (from hours to only a few seconds)

After that, it was a matter of hypertuning and testing the SVM model. Unlike the CNN, this was much easier to accomplish since less parameters are required and the training each time only took a few seconds. The regularization parameter was the only parameter to tune. The kernel chosen was a linear one, as it works well with binary classifications. Like before, the data was split into 80% training and 20% testing. The figures below show the training and testing accuracy collected over 20 trials. Each trial was made using a different shuffling of the data, and even different users (authorized images). The regularization parameters used for each of figures (a), (b), (c) and (d) are 1.0, 0.1, 0.01, 0.001 respectively.

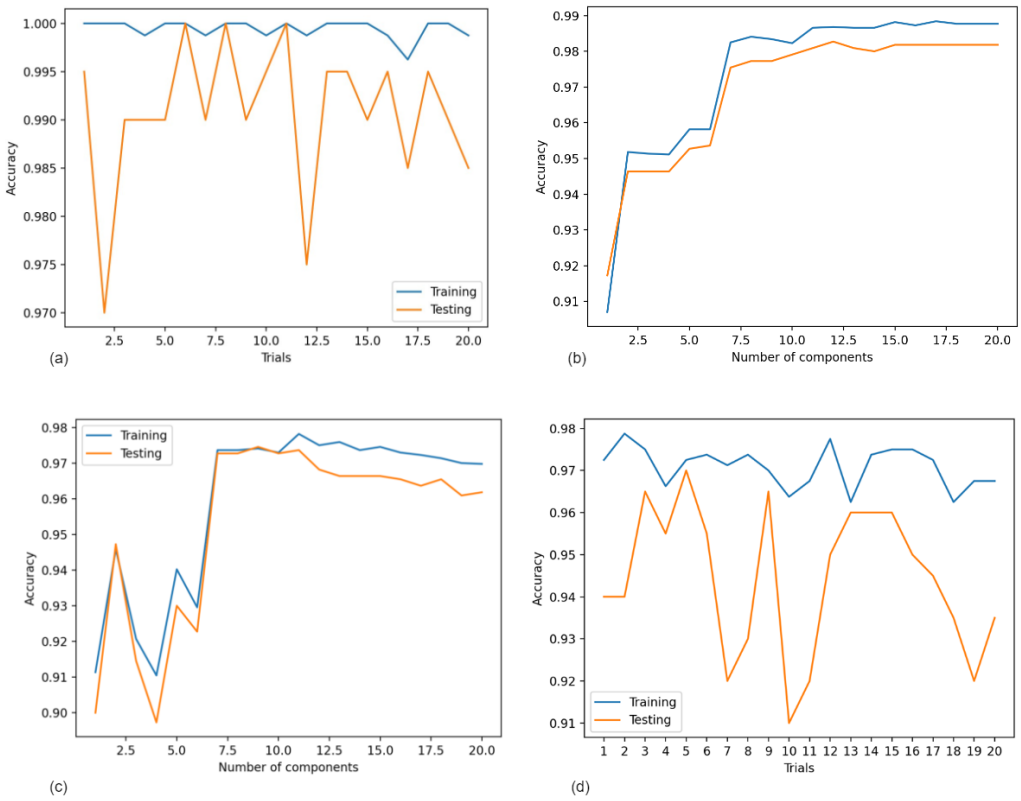

Figure 22: Accuracy of SVM model with (a) k=1.0 (b) k=0.1 (c) k=0.01 (d) k=0.001

These figures show that with a regularization parameter of 1.0, the model seems to overfit since the training and testing accuracies are too high (ranging between 97% and 100%) and sometimes equal. A parameter of 0.1 or 0.01 makes things slightly better, however the model still overfits sometimes since the testing accuracy can be higher than or equal to the training one. The last figure is the most optimal one, the accuracies are reasonable and fairly high. The testing accuracy is always lower than the training one, but high enough, ranging from 90 to 97%. The training time taken for each trial was also collected. The table below shows the training times for the first 5 trials.

Trials	Training Time
1	35 seconds
2	50 seconds
3	45 seconds
4	37 seconds
5	42 seconds

Table 3: SVM Training Time

In general, the average time taken to train the SVM model was around 40 seconds, which is a significant improvement from the CNN implementation.

6.2.2 LDU Testing

The LDU was first implemented as an interactive system that prompts the user to perform certain head movements (turn left, turn right, look up, look down). This makes it robust against spoofing using videos of the authorized user placed in front of the camera. The system was fully implemented but couldn't work properly on the Raspberry Pi since it was too heavy on the PiCamera. As no measurements were collected, it won't be considered as a testing unit in this document.

Blink Detection Testing

The main challenge was making the system work using the PiCamera. The latter's frame rate was not ideal and made it hard to detect blinks as not all frames were detected/read during the live video. The frame rate was changed multiple times but it did not make a difference.

The one parameter to test was the blink threshold. As mentioned previously, an EAR is calculated for each eye. This ratio ranges between 0 and 1, corresponding to eye closed and eye opened respectively. The average EAR of both eyes is calculated and compared to a blink threshold, if it is lower than that threshold then a blink is detected. Multiple thresholds have been tested, as indicated in the table below.

Blink Threshold	Blinks Performed	Blinks Detected	Accuracy
0.5	10	50	NA
0.4	10	13	NA
0.37	10	14	NA
0.35	10	7	70%
0.35	20	13	65%
0.33	10	5	50%
0.33	20	11	55%
0.32	10	6	60%
0.32	20	10	50%
0.3	10	4	40%

Table 4: Blink Detection Accuracy

As shown in Table 4, 0.5, 0.4 and 0.37 were too high of a threshold. The number of blinks detected was much higher than the ones actually performed since the algorithm mistakes closed eyes with open ones. When they are opened, the average EAR rages between 0.4 and 1 (approximately), which indicates that the threshold should be lowered. 0.35 was a better choice and 7 blinks were detected among 10, which gives it an accuracy of 70%. The same threshold was tried with 20 blinks performed, and it gave an accuracy of around 65%. Lower thresholds were also attempted and the algorithm gave different results. None of the accuracies were perfect since like mentioned before, the PiCamera detects frames at a very low rate. Therefore, accuracies equal or higher than 60% were considered reasonable. 0.35 was hence chosen as the final blink threshold.

6.2.3 Overall Computer Vision Testing

To test the LDU and the FRU working together, several test cases were run. The table below shows each one of them as well as the results yielded. Note that “passed” refers to the unit producing the right, expected results.

Test Case	Expected Behavior	Results
Real, Authorized User	Identify user as Real Identify user as Authorized Green LED ON Solenoid Unlocked	LDU passed FRU passed
Real, Unauthorized User	Identify user as Real Identify user as Unauthorized Red LED ON Solenoid Locked	LDU passed FRU passed
Real, Authorized User (different facial expressions)	Identify user as Real Identify user as Authorized Green LED ON Solenoid Unlocked	LDU passed FRU passed
Real, Authorized User (with/without glasses)	Identify user as Real Identify user as Authorized Green LED ON Solenoid Unlocked	LDU might fail FRU might fail
Real, Authorized User (shaved vs beard)	Identify user as Real Identify user as Authorized Green LED ON Solenoid Unlocked	LDU passed FRU might fail

Real, Authorized User (extreme lighting conditions)	Identify user as Real Identify user as Authorized Green LED ON Solenoid Unlocked	LDU might fail FRU might fail
Fake User (picture on phone, close)	Identify user as Fake Red LED ON Solenoid Locked	LDU passed
Fake User (picture on phone, far)	Identify user as Fake Red LED ON Solenoid Locked	LDU might fail
Real User (no blinking in front of camera)	Identify user as Fake Red LED ON Solenoid Locked	LDU passed

Table 5: Computer Vision Unit Test Cases and Results

This table shows that under normal settings, the LDU and FRU perform just as expected. By normal settings it is meant that the lighting is normal (not too dark, not too bright) and that the user does not have any major facial changes such as glasses, beards or makeup compared to the training images. The one limitation of the machine learning model is that there is no variation in the training images used for the authorized user. The one-minute video sent from the web app only captures a live video of the user in one specific setting, which means that despite being able to capture and train on different facial expressions, it doesn't have a lot of variety when it comes to facial changes (with or without glasses, makeup, different lighting settings, etc.). Therefore, the algorithm will, as expected, misclassify some authorized users if they differ considerably from the training images.

On the other hand, blink detection also has its limitations. Similar to the FRU, it is sensitive to lighting settings and extreme facial changes, specifically the ones that involve the eyes (like glasses). Additionally, the distance at which the phone is placed in front of the camera is an important factor in accurately detecting blinks. If the phone is too far, the eyes will appear very small and therefore the average EAR might be very close to 0 which can lead to blinks being detected. Therefore, the phone should be placed as close to the camera as possible for the LDU to work. Another limitation is the PiCamera's frame rate, the user has to blink multiple times otherwise it might not detect the frame where they are blinking.

Overall, the Computer Vision Unit performed as expected in most cases and produced accurate results.

6.3 Hardware Testing and Results

There are three sets of hardware tests that were performed to validate the results of the power system module that was developed. The first test that was completed was the validation of the rectifier used to power the Raspberry Pi. The results will be verified by using multimeter measurements. Data from the transformer input, transformer output, and the voltage output from the voltage regulator. The voltage verifies that the Raspberry Pi will operate since the device powers up as a result. Next, the run time of the battery bank backup supply is timed to determine how long the supply can last for. A video showing the timer with the battery bank running will validate the run time. Lastly, the switching time between input power and backup power will be validated using video demonstrations. The images from the rectifier testing along with the Youtube links are below.

The criteria for the voltage rectifier is to test for obtaining the correct voltage which will be used to power the Raspberry Pi. The desired voltage should be around 5VDC. In addition, the battery bank should be capable of lasting for at least 0.5 hours. The input switch should cause an instantaneous change between power sources when the rectifier power is lost or restored.The results show the measurements taken for the rectifier using a multimeter along with a video to demonstrate the time the battery bank can last for. In addition, a video to show the input switch is provided.

Figure 23: Transformer Input Voltage (VAC)

Figure 24: Transformer Output Voltage / Rectifier Input (VAC)

Figure 25: Rectifier Output (VDC) to Raspberry Pi

Below is a URL to a YouTube video to show the Raspberry Pi being powered through a no load test from the battery bank. Figure 26 shows an approximate graph of the battery bank's voltage over time to show its lifespan.

Battery Bank Video: https://youtu.be/msJmvlamVZg

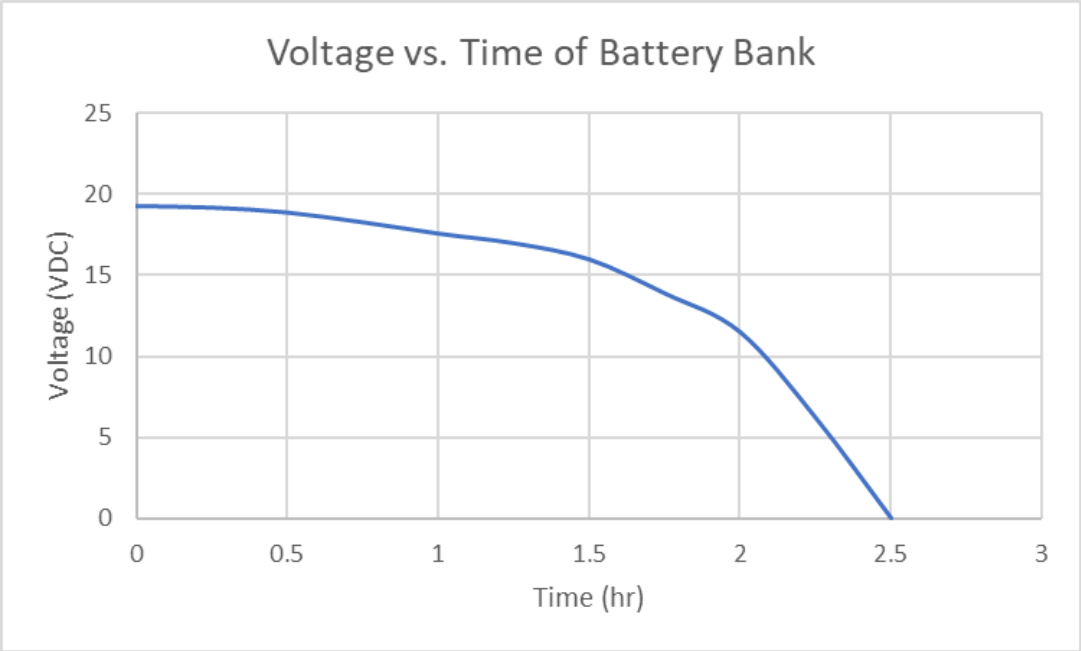

Figure 26: Battery Bank Life Depletion

Below is a URL to a YouTube Video to show the input power being switched over to the battery bank backup system and vice versa.

Power Switch Video: https://youtu.be/pdB-TQYCq2w

From testing, the Raspberry Pi voltage from the regulator reached approximately 4.9VDC at 3A to supply 15W of power. Occasionally, the Raspberry Pi would provide a warning that the input voltage is lower than desired since it requires 5V, but was still able to be powered with the voltage rectifier.

The battery bank was only tested to show up to about 45 minutes to meet the desired requirement, but is capable of lasting up to 2 hours. To meet the government constraint of the batteries not overheating, the current from the batteries were tested to see if it would remain consistent in a constant area at room temperature. The current remained the same and none of the batteries were damaged while being used in the battery bank. Also, the power switch is able to swap between sources in less than a second, but requires a system restart as a result.

7. Team and Timeline

7.1 Jake Smith

Jake will serve as the electrical engineer of the group for the project. He has design engineering experience with Mitsubishi Electric. He will be responsible for the hardware design. The hardware design includes component selection from third-party vendors (i.e. Digikey) for the microcontroller or microprocessor, the solenoid lock, the vibration sensor, the power system, and the camera configuration. For ECE design, the plan is to design the rectifier to power the Raspberry Pi module, design the vibration sensor to integrate with the Raspberry Pi module, configure the camera to work with the microcontroller, and working with the solenoid to integrate with the team to lock/unlock. Jake will also focus on how to provide power to the system in the event of a power outage at any time. As needed, Jake will design the PCB(s) to integrate shelf and external components together.

7.1.1 Skills Learned in ECE coursework

Jake has taken classes in ECE 0101 - Linear Circuits, ECE 0102 - Microelectronic Circuits, and ECE 1212 - Electronic Circuit Design Lab and will use knowledge from these classes for designing the power system for the circuit. Jake has an understanding of C++ from ECE 0301 - ECE Problem Solving with C++ and ECE 0302 - Data Structures & Algorithms if C++ is used on the Raspberry Pi module. Jake learned PCB and through-hole soldering skills in ECE 1895 - Junior Design. Jake got an understanding of uninterruptible power supplies in ECE 1775 - Power Quality to power circuits in the event of power outages.

7.1.2 Skills Learned Outside ECE coursework

Jake has gotten more proficient with surface-mount soldering through his time at Mitsubishi Electric. With PCB design and soldering, he will be able to reduce the size of circuit boards in order to save costs from the overall budget. Jake consulted with Dr. Robert Kerestes to validate the design of the 120VAC to 5VDC rectifier along with obtaining knowledge of a battery bank.

Jake has never used the Raspberry Pi module, so he will need to consult with a faculty member or online tutorials to learn how to use the device to integrate components to. In addition, the Raspberry Pi module comes preloaded with Python and Jake has not learned that programming language. If necessary, he would need to consult with ECE faculty in order to learn how to use the programming language. Jake observed Dr. Robert Kerestes using Python before and may be a good resource in the department.

7.2 Rayan Hassan

Rayan is a Computer Engineering major and will work on the computer vision unit of the project.

7.2.1 Skills Learned in ECE Coursework

Rayan took classes like Junior Design (ECE 1895) and Project and Systems Engineering (ECE 1140), where he worked in teams on a “Bop-It” inspired game and a Train Control System respectively. In the first one, he was in charge of the hardware part where he learned skills like PCB design and prototype testing. In the other one, he worked on the Wayside/Track Controller (software). A great deal of knowledge was gained in both classes, like teamwork and project organization, but also technical skills like Git, PyQt, Arduino Uno, etc. Another important class taken was Introduction to Machine Learning (ECE 1395), where he learned different algorithms and concepts for supervised and unsupervised learning (Regression, Classification, KNN, SVM, CNN, Binary Tree Decision, etc.). Throughout his academic journey, Rayan has learned many languages like Python, C, C++, Java, SQL, VHDL and R.

7.2.2 Skills Learned Outside ECE Coursework

Rayan has had some experience outside of the ECE curriculum. He had an online internship at the Chinese University of Hong Kong as part of the Summer Undergraduate Research Program 2022. He worked on Wearable Robots for Ultrasound Scanners, specifically on localization of targeted areas in US scanning using AI (CNN algorithms). Although his research was conceptual (review paper), he plans to use his knowledge about neural networks for this project.

7.3 Dane Krall

Dane will take on the role of data communication between the phone app, the Raspberry Pi and the locking system. Dane is a computer engineering major at Pitt and also majors in Physics from Slippery Rock University. Dane will be responsible for sending the viable photo data from the app to the Raspberry Pi to be compared against with the facial recognition software. This also includes interactions with a server the app and decoding the signals between all three to perform the actions required of these systems.

7.3.1 Skills Learned in ECE Coursework

Dane has taken ECE 1140 (Systems and Project Engineering) which leads to experience in software development and integration between different modules sending data between each other. This also gave experience in using Github and the importance of communication with a team. Dane has also taken ECE 1895 (Junior Design) and is currently taking ECE 1175 (Embedded Systems), giving him microcontroller experience using Arduino and Raspberry Pi respectively.

7.3.2 Skills Learned Outside ECE Coursework

During module implementation, Dane expects to do research about data communication between a Raspberry Pi, a server running on it, and a mobile app. While he has no prior experience in creating a mobile app, there are similarities between the work to be done as mentioned in ECE 1140.

7.4 Timeline

Week 1-2 CHECKOFF #1

Jake: Designed and implemented 120VAC to 5VDC voltage rectifier.
Rayan: Prepared the training data, implemented the CNN model and tested it with different parameters.
Dane: Designed simple app UI and implemented some HTTPS request functionality.

Week 3-4 MIDTERM PRESENTATION WEEK

Jake: Battery bank research & design for use with buck converter.
Rayan: Finalized the FRU and implemented the LDU (interactive system following first design concept).
Dane: Worked on implementation of different HTTPS requests in the python server and thoroughly tested their functionality.

Week 5-7 CHECKOFF #2

Jake: Developed input switch between voltage inputs & configured GPIO pins.
Rayan: Transferred everything onto the Raspberry Pi and configured the PiCamera and LEDs. Problems with Tensorflow and OpenCV libraries were encountered which resulted in the use of the 2nd design concept. SVM model was implemented instead of the CNN for the FRU, as well as blink detection instead of an interactive system for the LDU.
Dane: Further developed the Web App to have a login, signup and able to send information to the server to be stored.

Week 8-11 FINAL PRESENTATION, EXPO, REPORT

Jake: Testing and troubleshooting; all developed modules should be working and GPIO can be verified with code.
Rayan: Finalized the main code for both the FRU and the LDU, and ran several test cases for each. Configured the motion sensor and the solenoid lock as well. Tested communication with all other modules.
Dane: Implemented lock button and notifications.Finished integration with the Raspberry Pi and facial recognition and tested the full system.

8. New Skills Acquired & Learning Strategies

8.1 Dane Krall Responses

The skills acquired during this project were learning to use the httpserver library in Python to create the server and implement the HTTPS requests within it.The first resource used to learn was the documentation page so I could learn the function needed,their purposes,and how they function [21]. Next, a youtube tutorial by Neural Nine on the basics of implementing these requests was used to see the process of defining the HTTPS request and the main functions needed [22]. These functions were continuously updated throughout the project to allow for functionality with specific data types and to modify separate groups of data.

Another learning curve was how to use Flask in python to create the WebApp. Features learned were how to create pages,redirect pages,inputting data to a page,saving data submitted to a page, and how to use HTML to create pages. A number of resources were used here like the server; the documentation page was used to look at the functions being used and understand what they do [23]. A web tutorial and video tutorial by Dyouri from Digital Ocean and NeuralNine respectively were also used to learn the basics of running the Flask application and how to create an organized directory to learn good practice for applications created using the library [24] [25]. Another video used was one that had the basics of creating a login system that I refactored for my system and used as a basis for the signup system and a guide on how to submit data from a page. This knowledge was used to help implement the main functionality such as changing the lock state and submitting the video [26].

Lastly, the skill of how to send an email using python for the notification system had to be learned. To do this an online tutorial by de Langen from Real Python was read that gave a multitude of ways to send Emails using python [27]. Using this resource the method chosen was Python's built in email library due to its simplicity.

8.2 Rayan Hassan Responses

The highest learning curves for Rayan were the use of OpenCV and Tensorflow libraries. Although he had general knowledge in Machine Learning through ECE 1395, he had never used them before and had never worked on a facial recognition problem. Starting with OpenCV, detecting the faces and eyes required the use of a Haar Classifier. For that, online resources helped understand the gist of it and know exactly which .xml files to use, as there are many [30]. For the Tensorflow library, Rayan had never built a CNN before. A general tutorial about building it was followed [31]. This tutorial was not specific to facial recognition but instead explained how object recognition works. Reading it helped in getting acquainted with the different functions and parameters used in building the layers for the neural network. After that, it was a matter of

building a model that fits the requirements of this project, which was difficult to do as many limitations were imposed.

The biggest challenge however was working with the Raspberry Pi. Rayan had general knowledge about it from ECE 1175 (Embedded Systems), however working with libraries like OpenCV and Tensorflow was completely new. Virtual environments had to be created and different configurations of the Raspberry Pi's architecture were needed. Additionally, Linux was mainly used which Rayan had limited experience with. Finally, configuring the hardware sensors was also a new learning curve. A tutorial explaining how the picamera2 library is used was followed [32]. However even then, the camera had to be configured to accommodate the Pi model used, which was different. Finally, the GPIO pins had to be configured to control the LEDs, motion sensor and solenoid lock. For that, the official Raspberry Pi documentation [33] was referenced, as it showed all the pins and their functionalities. Overall, Rayan learned many skills specific to the project, but also gained experience in software and performance testing, as well as teamwork and system integration.

8.3 Jake Smith Responses

To learn skills that were not already covered in the classroom, web resources along with a professor at the University of Pittsburgh needed to be referenced. Even though rectifiers were covered in ECE0102, reference [18] provided a refresher along with how a transformer is used to step down the voltage. The information was organized by referencing text and images. The battery bank was suggested by Dr. Robert Kerestes of the University of Pittsburgh as a way to provide backup power. To better understand the topic, reference [19] provided a better understanding of a battery bank. The information showed how to add voltage or amp-hours based on series or parallel battery configurations.

Using the information that was gathered, a voltage bridge rectifier was developed with a voltage that was stepped down through a 10:1 transformer. The knowledge of transformers was successfully applied to the rectifier, but had heat loss issues with an excess amount of current flowing into a voltage regulator. As a result, the rectifier needed to be redesigned to add a current divider as a way to limit the amount of heat loss. This would prevent the need for a heat sink on such a small regulator. Once the divider was added, the regulator losses were eliminated. With the battery bank knowledge acquired, two sets of three 9VDC connected in parallel were connected in series to provide an 18 VDC source at about 2 Ah. The source was fed into a buck converter to meet the minimum voltage and current requirements to power the Raspberry Pi. The design was made on a PCB to apply the knowledge that was gained.

9. Conclusions and Future Work

The Facial Recognition Lock System created was able to meet the main requirements and produce accurate results. However, the device had multiple limitations. The Machine Learning algorithm was trained on a very uniform and homogenous training data for the authorized user. Therefore, it is sensitive to changes in lighting and noticeable facial changes. Additionally, the PiCamera used had a slow frame rate and could not perform well under certain circumstances. For hardware limitations, there was a limited period that the battery bank could be active for since non-rechargeable batteries were used. They needed to be replaced once they were depleted since there is no safe way to recharge them. In addition, the input switch transience resulted in a moment where there was no power being supplied to the Raspberry Pi. The microprocessor needed to restart as a result of switching between power supplies. On the data transmission side, one limitation is that the device currently only works for a singular lock system and not multiple, which would be needed for a commercial product. The web app also only works on a local network and would ideally have to function using any WiFi network.

Future work can be centered around producing a more robust Machine Learning algorithm, with more diverse training data with different lighting setups and facial changes. A CNN model can be adopted as it offers more flexibility and better results. The system can also be modified to prevent spoofing not only using pictures on a phone, but also videos of the authorized user. The first LDU implementation was able to do that, but it can be developed into a more robust and secure system that works on the chosen Hardware. In addition, more work can be performed on trying to optimize the backup power supply. Using the input power source, a circuit could be developed to recharge a backup power supply so that batteries don't need to be swapped out after they become discharged. Also, more work would need to be done to optimize the transience experienced from switching from the voltage regulator to the battery bank. Transience would need to be eliminated when switching between sources that currently requires a restart of the system. On another hand, to make it so that the communication system can work with multiple lock systems, each profile would be dedicated to a lock system through some kind of unique code.The profiles would then be added and identified using this code. The current system also doesn't support sending multiple videos, therefore a system that allows for multiple videos or subprofiles of users can be developed. This feature would allow for a logging feature to keep track of any user flagged as valid when they use the system. To make the WebApp work for users outside the WiFi network either some port mapping would have to be done on a personal network or a hosting service would need to be used.

References

[1] L. Li, X. Mu, S. Li and H. Peng, "A Review of Face Recognition Technology," in IEEE Access, vol. 8, pp. 139110-139120, 2020, doi: 10.1109/ACCESS.2020.3011028.

[2] F. Schroff, D. Kalenichenko and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015, pp. 815-823, doi: 10.1109/CVPR.2015.7298682.

[3] Networks, A. N. (2023, August 31). *What are the advantages and disadvantages of using pre-trained CNNS?*. Pre-trained CNNs: Benefits and Challenges for Image Tasks. https://www.linkedin.com/advice/0/what-advantages-disadvantages-using-7050897204365991936

[4] K, GOPALA KRISHNAN, Face Anti-Spoofing Using Deep Learning. Available at SSRN: https://ssrn.com/abstract=4089543 or http://dx.doi.org/10.2139/ssrn.4089543

[5] Anthony, Peter & Ay, Betul & Aydin, Galip. (2021). A Review of Face Anti-spoofing Methods for Face Recognition Systems. 1-9. 10.1109/INISTA52262.2021.9548404.

[6] “The Advantages and Disadvantages of a Mobile App.” *Objectiveit*, Objective, 2021, objectiveit.com/blog/the-advantages-and-disadvantages-of-a-mobile-app/. Accessed 28 Sept. 2023.

[7] Roomi, Mishal. “5 Advantages and Disadvantages of HTTP: Drawbacks & BENEFITS OF HTTP.” *HitechWhizz*, HitechWizz, 14 Aug. 2020, www.hitechwhizz.com/2020/08/5-advantages-and-disadvantages-drawbacks-benefits-of-http.html.

[8] Rolling, Mitch. “Why Transitioning to Renewable Energy Leads to Power Outages.” *American Experiment*, 26 Aug. 2020, www.americanexperiment.org/why-transitioning-to-renewable-energy-leads-to-power-outages/.

[9] “The Pros and Cons of Power Backup Systems.” *Energy5*, 22 Sept. 2023, energy5.com/the-pros-and-cons-of-power-backup-systems#anchor-3.

[10] DeCapua, Todd. “Front-End vs Back-End Performance Metrics for Mobile Apps.” *TechBeacon*, TechBeacon, 6 Feb. 2020, techbeacon.com/app-dev-testing/understanding-front-end-vs-back-end-performance-metrics-mobile-apps.

[11] GeeksforGeeks. (2023, January 3). *Determine the face tilt using OpenCV - Python*. GeeksforGeeks. https://www.geeksforgeeks.org/determine-the-face-tilt-using-opencv-python/

[12] GitHub. (n.d.). https://github.com/opencv/opencv/tree/master/data/haarcascades

[13] Tranter, Jeff. “Control Raspberry Pi GPIO Pins from Python.” *ICS - Integrated Computer Solutions*, ICS, 31 July 2019, www.ics.com/blog/control-raspberry-pi-gpio-pins-python. Accessed 28 Sept. 2023.

[14] Cassidy, Lance. “Arduino vs Raspberry Pi: Which Is the Best Board for You.” *Www.flux.ai*, Flux, 7 Feb. 2023, www.flux.ai/p/blog/arduino-vs-raspberry-pi-comparison. Accessed 25 Sept. 2023.

[15] A. S. Savanth, K. G. R. Manish, P. Narayan, M. L. Nikhil and V. G. Gokul, "Face Recognition System with 2D Anti-Spoofing," 2022 IEEE World Conference on Applied Intelligence and Computing (AIC), Sonbhadra, India, 2022, pp. 226-230, doi: 10.1109/AIC55036.2022.9848909.

[16] Breuss, Martin. “Python-Driven Web Applications – Real Python.” *Realpython.com*, Real Python, 1 Feb. 2021, realpython.com/python-web-applications/. Accessed 28 Sept. 2023.

[17] McKenzie, Cameron. “How to Install Apache’s Web Server on Windows 10 Quickly.” *Www.theserverside.com*, TechTarget, 15 Jan. 2022, www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Install-Apache-Web-Server-24-Windows-10-ServerRoot-Error. Accessed 28 Sept. 2023.

[18] Gupta, Sourav. “AC to DC Converter Circuit.” *Circuit Digest*, circuitdigest.com/electronic-circuits/ac-to-dc-converter-circuit-diagram. Accessed 2 Oct. 2023.

[19] GEGCalculators. “Battery Series and Parallel Connection Calculator.” *GEGCalculators*, 8 Oct. 2023, gegcalculators.com/battery-series-and-parallel-connection-calculator/.

[20] Ramuglia, Gabriel. “What Is a Good Ping? Why Ping Speed Matters for Websites and Gaming.” *Linux Dedicated Server Blog*, IOFlood, 8 Nov. 2023, ioflood.com/blog/what-is-a-good-ping-result-good-ping-time/.

[21]“Http.Server - HTTP Servers.” *Python Documentation*, Python Software Foundation, docs.python.org/3/library/http.server.html#. Accessed 13 Dec. 2023.

[22]“ Simple HTTP Server in Python .” *YouTube*, NeuralNine, 26 Dec. 2021, https://www.youtube.com/watch?v=DeFST8tvtuI. Accessed 13 Dec. 2023.

[23]“Quickstart.” *Quickstart - Flask Documentation (3.0.x)*, Pallets, flask.palletsprojects.com/en/3.0.x/quickstart/#a-minimal-application. Accessed 13 Dec. 2023.

[24]Dyouri, Abdelhadi. “How to Structure a Large Flask Application with Flask Blueprints and Flask-Sqlalchemy.” DigitalOcean, DigitalOcean, 18 Nov. 2022, www.digitalocean.com/community/tutorials/how-to-structure-a-large-flask-application-with-flask-blueprints-and-flask-sqlalchemy.

[25]“ Flask Blueprints Make Your Apps Modular & Professional .” *YouTube*, NeuralNine, 24 Apr. 2023, https://www.youtube.com/watch?v=_LMiUOYDxzE. Accessed 13 Dec. 2023.

[26]“ Create Login Page With HTML And Flask| .” *YouTube*, Nachiketa Hebbar, 14 Dec. 2019, https://www.youtube.com/watch?v=R-hkzqjRMwM. Accessed 13 Dec. 2023.

[27]de Langen, Joska. “Sending Emails with Python.” *Real Python*, Real Python, 17 Nov. 2023, realpython.com/python-send-email/.

[28] *Product Datasheet - Energizer*, data.energizer.com/pdfs/522.pdf. Accessed 8 Nov. 2023.

[29] Qengineering. (n.d.). *Qengineering/tensorflow-raspberry-pi_64-bit: Tensorflow installation wheels for Raspberry Pi 64 OS*. GitHub. https://github.com/Qengineering/TensorFlow-Raspberry-Pi_64-bit

[30] Patil, S. (2019a, October 19). *Face and eyes detection - using opencv*. Medium. https://soumyapatilblogs.medium.com/face-and-eyes-detection-using-opencv-9fcad47656a4

[31] *Convolutional Neural Network (CNN) : Tensorflow Core*. TensorFlow. (n.d.). https://www.tensorflow.org/tutorials/images/cnn

[32] YouTube. (2023, January 19). *Raspberry Pi lesson 45: Using the raspberry pi camera in Bullseye with opencv*. YouTube. https://www.youtube.com/watch?v=kuJpdAf07WQ

[33] *Raspberry pi documentation*. Raspberry Pi hardware. (n.d.). https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

[34] Subedi, S. (2018, August 16). *Utkface*. Kaggle. https://www.kaggle.com/datasets/jangedoo/utkface-new