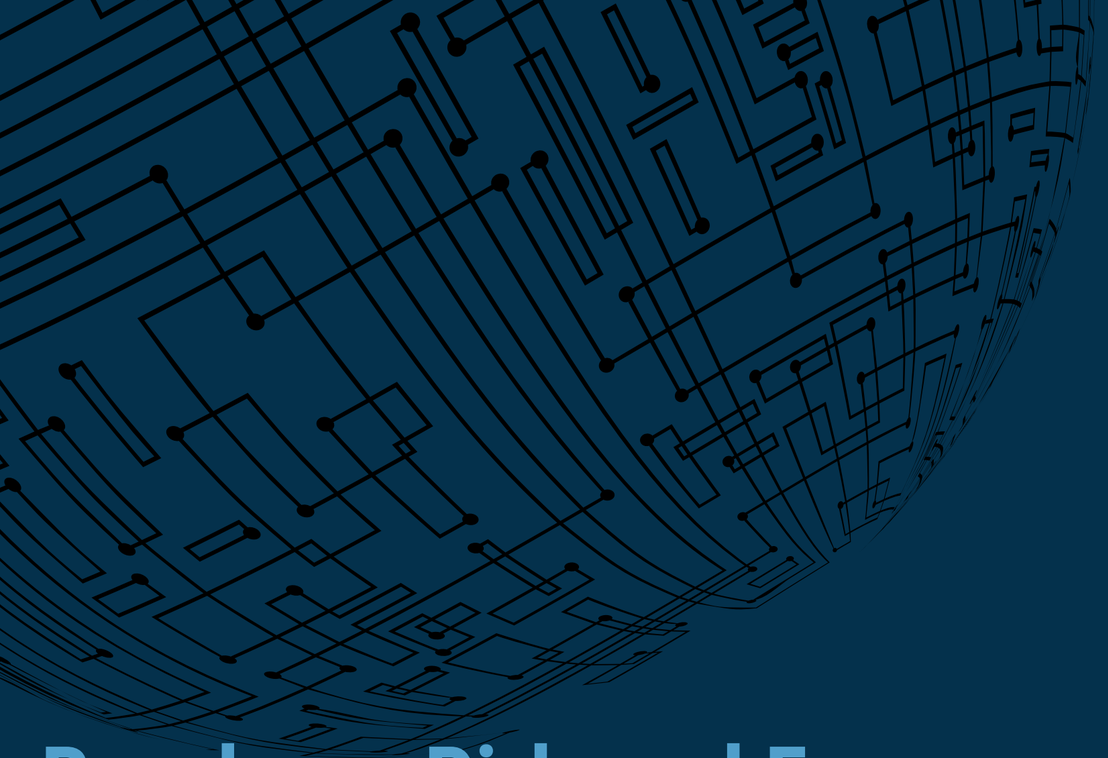


# Raspberry Pi-based Face Recognition Door Lock System


**Name: Seifeldin Sherif Elnozahy**

**Student ID: 1191102388**

**Supervisor: DR. Chinnaiyan Senthilpari**

**Modrator: DR. Lee Chu Liang**


**Date: 05-FEB-2025**


**Multimedia University**

# Table Of Contents

# Introduction

- Security plays a vital role in protecting both residential and commercial properties.
- Traditional locks present several vulnerabilities: keys can be lost, duplicated, or stolen, and PIN codes can be guessed or hacked.
- Facial recognition technology is a modern solution that eliminates these security risks by utilizing biometric authentication.
- This project aims to develop an affordable and scalable Raspberry Pi-based face recognition door lock system that enhances security, provides real-time notifications, and adapts to environmental conditions.

# Problem Statement

- Traditional key-based locking mechanisms have significant security flaws:
  - Lost or stolen keys allow unauthorized access.
  - Locks can be picked, and keys can be duplicated.
  - PIN-based access codes are susceptible to brute force attacks or hacking.

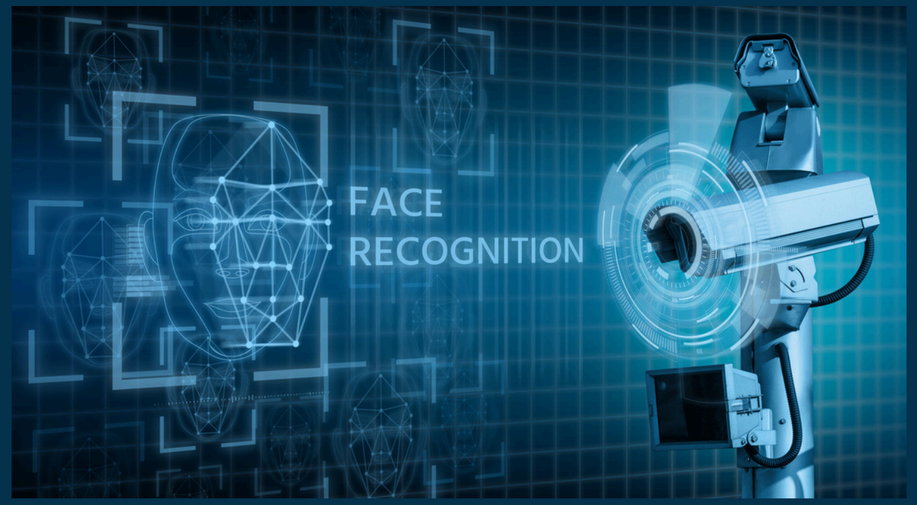
- Existing biometric authentication systems tend to be expensive and require specialized hardware, making them inaccessible for small-scale users.

- There is a need for a low-cost, efficient, and adaptive security solution that eliminates these vulnerabilities and ensures reliable access control.

# WHY
# THIS PROJECT?

**Motivations:**

- Enhance security by reducing vulnerabilities associated with traditional locks.
- Improve efficiency in access control systems.
- Increase user-friendliness by eliminating the need for physical keys.
- Leverage advancements in AI and machine learning for better security solutions.




**Objectives:**

- Develop a real-time facial recognition door lock system that utilizes Raspberry Pi and OpenCV for authentication.
- Enhance security and efficiency by eliminating the need for physical keys or PINs.
- Ensure high recognition accuracy in varying lighting and environmental conditions.
- Integrate adaptive learning capabilities, allowing the system to update its database dynamically.
- Enable real-time notifications for user authentication logs, including alerts for unauthorized access attempts.

# Literature Review

- Overview of Previous Research on Facial Recognition:
  - Facial recognition technology has evolved from basic feature-based methods to deep learning models.
  - Traditional methods like Eigenfaces and Fisherfaces were used in early systems but struggled with lighting and pose variations.
  - Modern approaches use machine learning and deep learning, improving accuracy and adaptability.

- Researchers have explored various face detection techniques:
  - Haar Cascade classifiers: Lightweight and real-time but with high false-positive rates.
  - Histogram of Oriented Gradients (HOG) + SVM: More robust but sensitive to lighting changes.
  - Convolutional Neural Networks (CNNs): High accuracy but requires significant processing power.

- Existing Research Challenges:
  - High dependency on lighting conditions.
  - Performance limitations due to low processing power on embedded devices.
  - Adaptive learning methods are not commonly integrated into face recognition systems.

- Our Approach:
  - Implementing real-time adaptive learning.
  - Improving recognition under varying environmental conditions.

# SUMMARY

| Paper | Techniques/Algorithms Used | Key Findings | Challenges/Limitations |
|---|---|---|---|
| A Face Recognition Method In Machine Learning (ML) For Enhancing Security In Smart Home | CNN, CNN Mobilenet, CNN Alexnet | High accuracy in recognizing faces | Data preparation, computational complexity, privacy concerns |
| Raspberry Pi-Powered Door Lock with Facial Recognition | OpenCV, Haar Cascade | Effective for secure access control | Processing power and time for training, importance of reliable internet connectivity |
| Face Recognition Door Lock System Using Raspberry PI | Haar Cascade Classifier, OpenCV | Accurate face detection, cost-effective solution | Real-time processing limitations |
| Home Security System Based On Facial Recognition | HOG algorithm, IoT, GSM | High precision and efficiency, remote access control | Privacy concerns, user acceptance |
| Face Recognition Door Lock System Using Raspberry Pi | PCA, OpenCV | Cost-effective, low power consumption | Integration with existing security infrastructure |

# System Overview

## Hardware Components

- **Raspberry Pi 3B:** Serves as the central processing unit, executing facial recognition algorithms and controlling the lock mechanism.
- **Camera Module V2:** Captures real-time images for facial detection and recognition.
- **Relay Module:** Acts as an electronic switch to control the solenoid lock upon successful authentication.
- **Solenoid Lock:** Physically secures or releases the door based on recognition results.
- **Power Supply:** Ensures stable operation of the Raspberry Pi and peripherals.

## Software Components

- **OpenCV:** Handles image processing, face detection and pre-processing tasks to enhance recognition accuracy.
- **DeepFace:** Implements deep learning-based facial recognition and emotion detection to classify known and unknown individuals.
- **Python-based Control System:** Manages face detection, recognition, access logging, and adaptive learning.
- **Real-time Notifications:** Sends alerts or access logs upon recognition events.

# Hardware Implementation

## Circuit Connections and Assembly Process

1. **Prepare the Raspberry Pi:**
   - Insert the preloaded microSD card with the OS and software.
   - Connect the Camera Module to the CSI port, ensuring correct ribbon alignment.
   - Secure the Raspberry Pi on a stable platform or protective case.
2. **Connect the Relay Module:**
   - Wire the relay module to the Raspberry Pi:
   - VCC → 5V GPIO
   - GND → GND GPIO
   - IN1 → GPIO 17
   - Position the relay close to the Raspberry Pi for secure wiring.
3. **Wire the Solenoid Lock:**
   - Positive terminal of 12V Solenoid Lock → NO (Normally Open) terminal of relay.
   - COM (Common) terminal on relay → Positive terminal of 12V DC Power Supply.
   - Negative terminal of solenoid lock → Negative terminal of 12V DC Power Supply.
4. **Integrate the DC Barrel Jack:**
   - Use a DC barrel jack adapter to securely connect the 12V DC power supply to the solenoid lock and relay module.

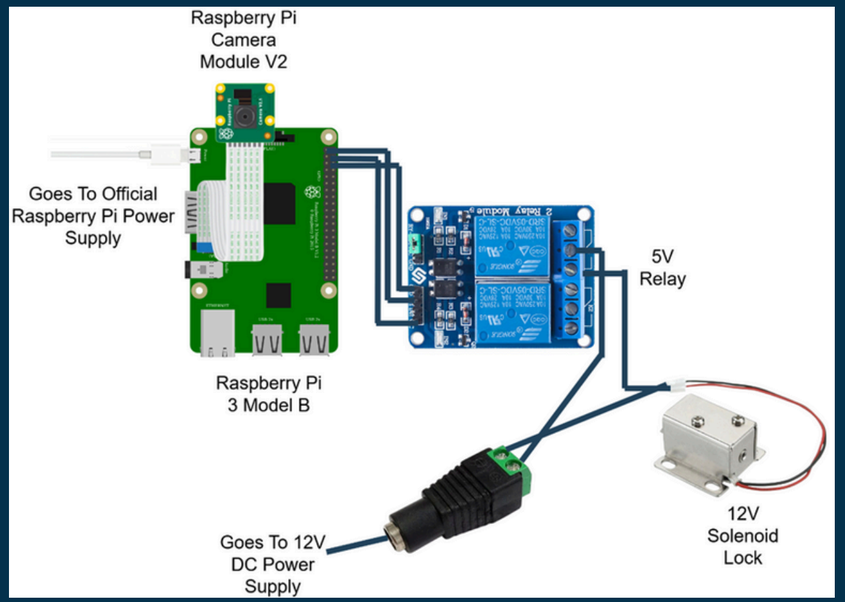
# Hardware Implementation

## Circuit Connections and Assembly Process

5. **Establish Stable Connections:**
   - Verify all jumper wires are secure and correctly aligned.
   - Inspect relay, solenoid lock, and Raspberry Pi GPIO pins for loose wires or misconfigurations.

6. **Power Up the Raspberry Pi:**
   - Use the official power adapter to power the Raspberry Pi via micro-USB.
   - Confirm the 12V DC power supply is securely connected to the relay and solenoid lock.

**Circuit diagram illustrating the hardware connections**



# Software Implementation

## Face Detection, Recognition, and Emotion Analysis using OpenCV and DeepFace

- OpenCV handles real-time face detection, processing video frames for facial recognition.
- DeepFace performs facial recognition and emotion detection, classifying emotions such as happy, sad, neutral, surprised, and angry.
- The system detects a face, extracts features, and compares them to a stored dataset for authentication.
- If a recognized face is detected, access is granted; otherwise, an alert is triggered.

### Face Detection and Recognition in Real Time

```
# Resize frame for faster processing
small_frame = cv2.resize(rgb_frame, (0, 0), fx=0.25, fy=0.25)
face_locations = face_recognition.face_locations(small_frame, model="hog")
face_locations = [(top * 4, right * 4, bottom * 4, left * 4) for (top, right, bottom, left

# Encode detected faces
face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)

# Compare with known encodings
face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
if face_distances.size > 0 and face_distances.min() < 0.4:
    name = known_face_names[face_distances.argmin()]
else:
    name = "Unknown"
```

## Adaptive Learning for Known and Unknown Faces

Adaptive learning applies to both known and unknown faces:

- If an authorized user's appearance changes over time, the system updates their stored data to improve recognition accuracy.
- If an unknown face is detected, the system allows an authorized user to approve and add the face dynamically.
- This feature enhances scalability and system intelligence over time.

### Updating Encodings for Known Faces

```
if best_match_distance < 0.4:
    name = known_face_names[best_match_index]
    known_face_encodings[best_match_index] = face_encoding  # Update encoding
    print(f"Updated {name}'s face encodings for improved recognition.")
```

### Code snippet for emotion detection

```
for face_encoding, face_location in zip(face_encodings, face_locations):
    try:
        top, right, bottom, left = face_location
        face_region = rgb_frame[top:bottom, left:right]  # Extract face region
        emotion_result = DeepFace.analyze(face_region, actions=["emotion"], enforce_d
        detected_emotion = emotion_result["dominant_emotion"]
        notification_client.send(f"Detected emotion: {detected_emotion}")
    except Exception as e:
        print("Error in emotion detection:", e)
```

### Code snippet of adding unknown faces

```
elif key == ord('a') and unknown_face_detected:
    new_name = input("Enter the name for the new authorized person: ")
    known_face_encodings.append(new_face_encoding)
    known_face_names.append(new_name)
    save_encodings_to_file(known_face_encodings, known_face_names, encodings_file)
    print(f"{new_name} added to the authorized list.")
    unknown_face_detected = False  # Reset after adding
```

# Software Implementation

## Lighting and Distance Adaptation for Improved Accuracy

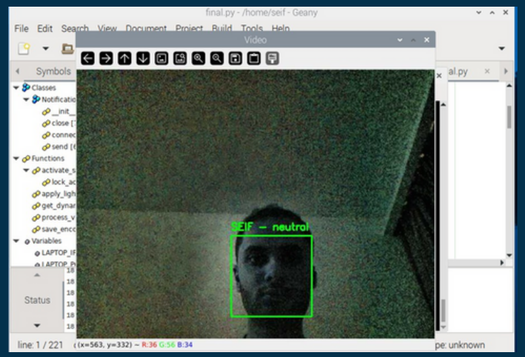
The system is optimized to work under varying lighting conditions using:

- Histogram Equalization to improve contrast in low-light environments.
- Gamma Correction to balance brightness levels.

Distance adaptation ensures accurate recognition at various ranges:

- Uses scaling techniques to recognize faces whether they are close or far from the camera.
- The model compensates for perspective distortions for better accuracy.

### Effect of Lighting Normalization on Face Detection




## Real-Time Notifications and Security Alert Integration

- Access logs are updated in real-time, keeping a record of all entry attempts.
- Unauthorized access attempts trigger instant security alerts.
- Notifications are sent via email or messaging services, providing administrators with immediate updates.

### Server-Side Script

noti.py

C: > Users > sseno > OneDrive > Documents > FYP > noti.py

```python
import socket
import pyttsx3

# Real-time notification setup
HOST = "0.0.0.0"  # Listen on all interfaces
PORT = 5000

engine = pyttsx3.init()

# Function to speak the notification
def speak_notification(message):
    engine.say(message)
    engine.runAndWait()

# Server setup
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
    server_socket.bind((HOST, PORT))
    server_socket.listen()
    print(f"Listening for notifications on {HOST}:{PORT}...")

    while True:
        client_socket, address = server_socket.accept()
        with client_socket:
            print(f"Connection from {address}")
            message = client_socket.recv(1024).decode('utf-8')
            print(f"Received: {message}")
            speak_notification(message)
```

### The send_notification function

```python
# Function to send notifications to the server
def send_notification(message):
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client_socket:
            client_socket.connect((LAPTOP_IP, LAPTOP_PORT))
            client_socket.sendall(message.encode('utf-8'))
    except Exception as e:
        print(f"Error sending notification: {e}")
```

### Example of Triggering Notifications

```python
if best_match_distance < 0.4:
    name = known_face_names[best_match_index]
    send_notification(f"Access granted for {name}!")
else:
    send_notification("Unknown face detected!")
```

# Flowchart of System Operation

## 1. Start the System

- The system initializes and loads the camera module.

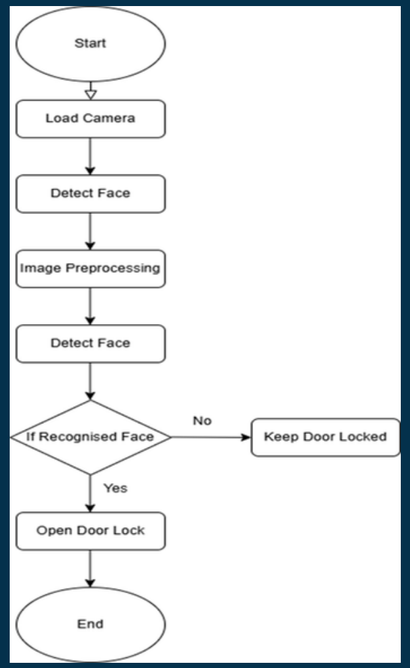
## 2. Load Camera & Detect Face

- The Raspberry Pi activates the camera and starts capturing frames.
- The OpenCV face detection algorithm scans the live video feed to locate a face.

## 3. Image Preprocessing

- The detected face undergoes grayscale conversion, histogram equalization, and normalization to improve recognition accuracy.

**Flowchart of the system**




## 4. Face Recognition & Decision Making

- The DeepFace model extracts facial embeddings and compares them with the stored dataset.
- If a recognized face is detected → Proceed to access control.
- If the face is not recognized → The door remains locked.

## 5. Access Control Decision

If a known face is identified:

- The Raspberry Pi sends a signal to the relay module, triggering the solenoid lock to unlock the door.

If the face is unknown:

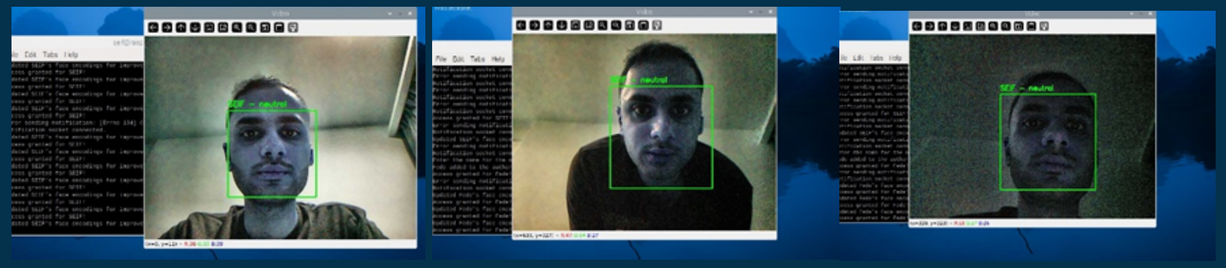
- The system denies access and keeps the door locked.

# Experimental Setup Overview

The system was tested in real-world environments to assess its accuracy and adaptability under different conditions:

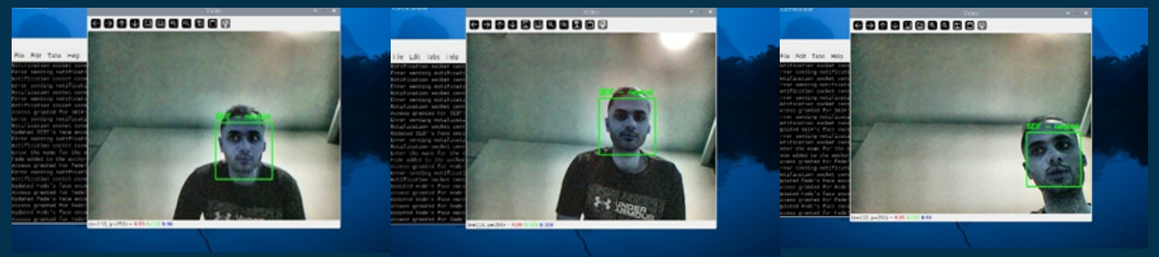
- **Lighting Variations:**
  - Bright Environment: Capturing facial features clearly in high-exposure conditions.
  - Low-Light Conditions: Evaluating system robustness in poorly lit environments.
  - Dim-Light Scenarios: Testing adaptability under minimal artificial lighting.

**Face Recognitions in a different lighting Environments**



- **Distance-Based, Facial Orientation & Angle Variations:**
  - Close-range recognition: Ensuring detailed feature extraction.
  - Medium-range testing: Assessing detection efficiency at an intermediate distance.
  - Frontal recognition: Ideal conditions for face identification.
  - Side angles and narrow perspectives: Evaluating system performance with partially visible faces.

**Face Recognitions at different Distances and angles**



# Model Training & Adaptation

## Training Methodology for Facial Recognition Accuracy

- The system was trained using DeepFace with a dataset of multiple face images under varying conditions.

Preprocessing steps included:

- Grayscale conversion for reducing computation complexity.
- Histogram equalization to enhance contrast in low-light images.
- Data augmentation techniques to improve generalization.

- The model was trained on a Raspberry Pi with optimized settings to balance accuracy and computational efficiency.
- Evaluation metrics such as precision, recall, and F1-score were used to measure model performance.

## Adaptive Learning Mechanism for Known & Unknown Faces

Incremental learning approach:

- When a known user's facial features change (e.g., due to lighting variations, aging, or angle differences), the system updates the stored facial embeddings.
- Unknown faces are flagged and can be authorized in real-time by an administrator.

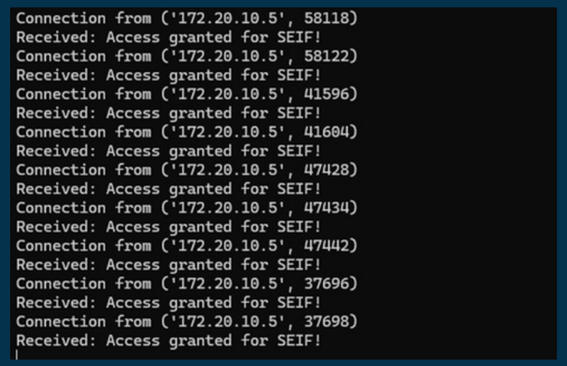
- The adaptive learning mechanism improves system reliability by dynamically adjusting the model over time.

# Real-Time Notifications & Security Alerts

## Overview of Notification System

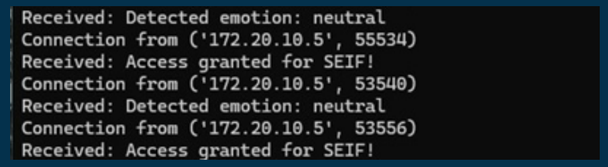
- The notification system provides real-time feedback to users about access events.
- It supports textual and audible alerts, ensuring users stay informed about:
  - Recognized faces
  - Unknown face detection
  - Emotion detection results
- Implemented using a socket-based client-server model, operating efficiently without requiring additional hardware or mobile applications.

**Notification for Authorized User**




1. Triggering Notifications:
   - Events such as recognized faces, unknown face detections, and emotion detection activate the notification system.
   - Messages include details like recognized user names or detected emotions.
2. Client-Side Communication (Raspberry Pi):
   - The Raspberry Pi establishes a TCP socket connection with the server (laptop) and sends messages.
3. Server-Side Processing (Laptop):
   - The server logs messages in the terminal and provides audible alerts using TTS (Text-to-Speech).
4. User Feedback:
   - Users receive real-time access updates and security alerts, prompting action when necessary.

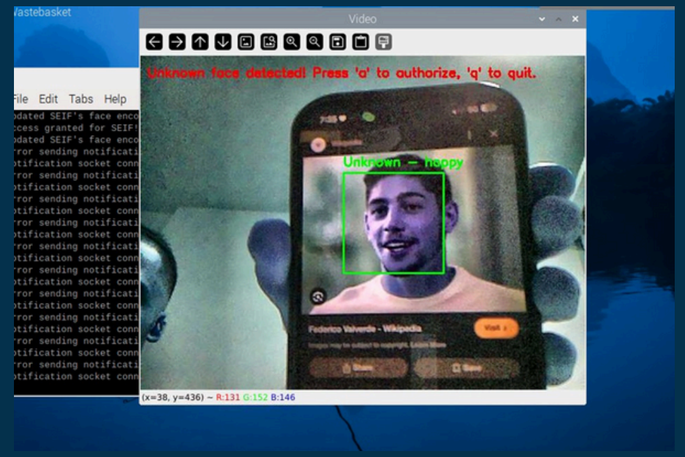
**Notification for Detected Emotion**



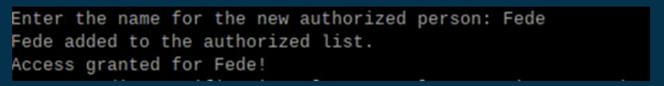
# Adding Unknown Faces

1. Prompting for Unknown Faces
    - When an unrecognized face is detected, a real-time prompt appears on the Raspberry Pi terminal, requesting the user's input:
        - Press 'a': Authorize and add the detected face to the dataset.
        - Press 'q': Ignore and quit the operation.
    - If the user authorizes the face, the 128-dimensional facial encoding is stored along with the provided name.
2. Storing New Face Encodings
    - Once a face is added, the dataset dynamically updates with the new face encoding.
    - The new user is recognized in subsequent interactions without requiring a system restart or retraining.
    - This ensures efficiency and maintains system robustness.
3. Recognizing Newly Added Faces
    - After addition, the system immediately incorporates the updated encoding into the recognition pipeline.
    - Future detections display the user's name and emotional state in real-time.
    - The adaptive learning feature enables the system to evolve dynamically without manual intervention.

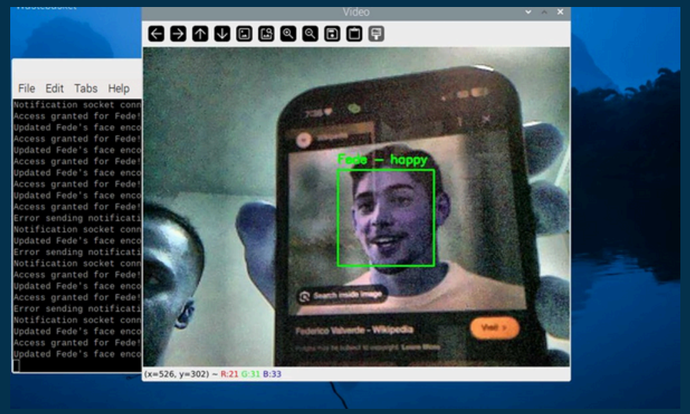
**Detection of an Unknown Face**




**Authorizing a New Face in Real Time**



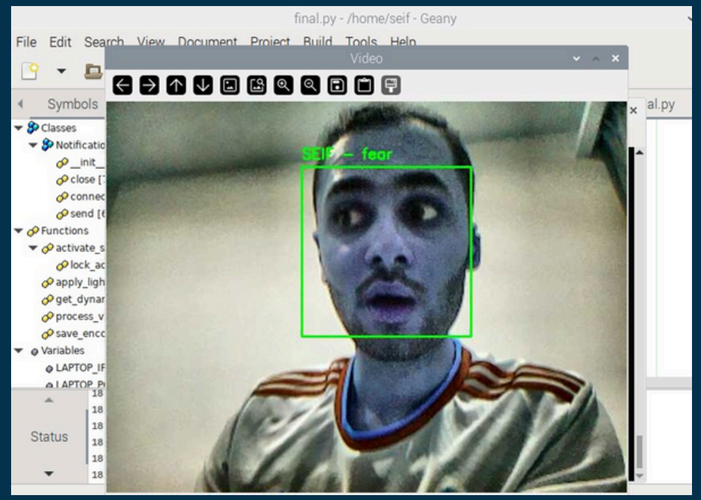

**Successful Recognition of a Newly Authorized User**



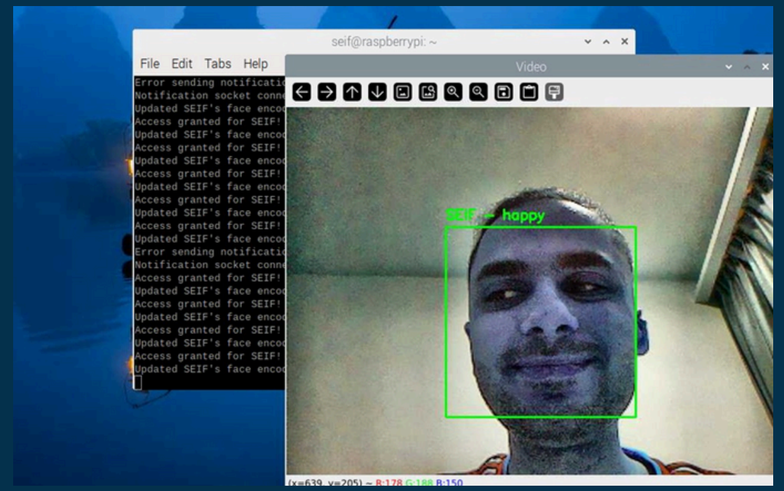
# Emotion Detection

- The system utilizes DeepFace’s emotion recognition model to analyze facial expressions in real-time.
- Detected emotions enhance security monitoring by identifying potential anomalies in user behavior.
- The system was evaluated under different facial expressions and user emotions.

**Real-Time Emotion Detection Display**




**Different Emotion Detection Display**

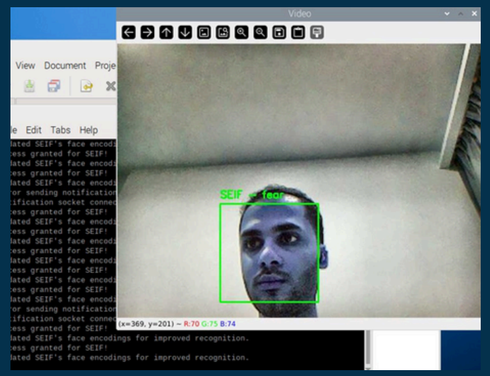


# Real-Time Face Recognition & Authentication

## Demonstration of Real-Time Performance with Sample Detections

- The system operates in real-time, continuously detecting and recognizing faces.
- It captures video frames from the Raspberry Pi Camera Module, processes them using OpenCV and DeepFace, and performs recognition instantly.
- Live detection samples show how the system identifies known individuals and classifies unknown faces for adaptive learning.

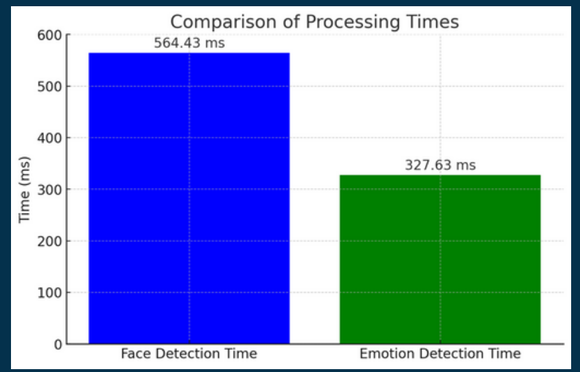
**Face Detection and Recognition in Real Time**




## Face Detection Speed and Accuracy in Different Environments

- Processing speed:
  - Face detection runs at an average speed of 564.43 ms per frame.
  - Recognition and decision-making complete within 327.63 ms.
- Performance in different lighting conditions:
  - Bright light: 98.5% accuracy.
  - Low light: 92.8% accuracy.
  - Medium distance: 96.2% accuracy.
  - Narrow angle: 92.4% accuracy.
- The model adapts to various distances and facial orientations using advanced preprocessing techniques.

**Bar Chart Comparing Processing Times**

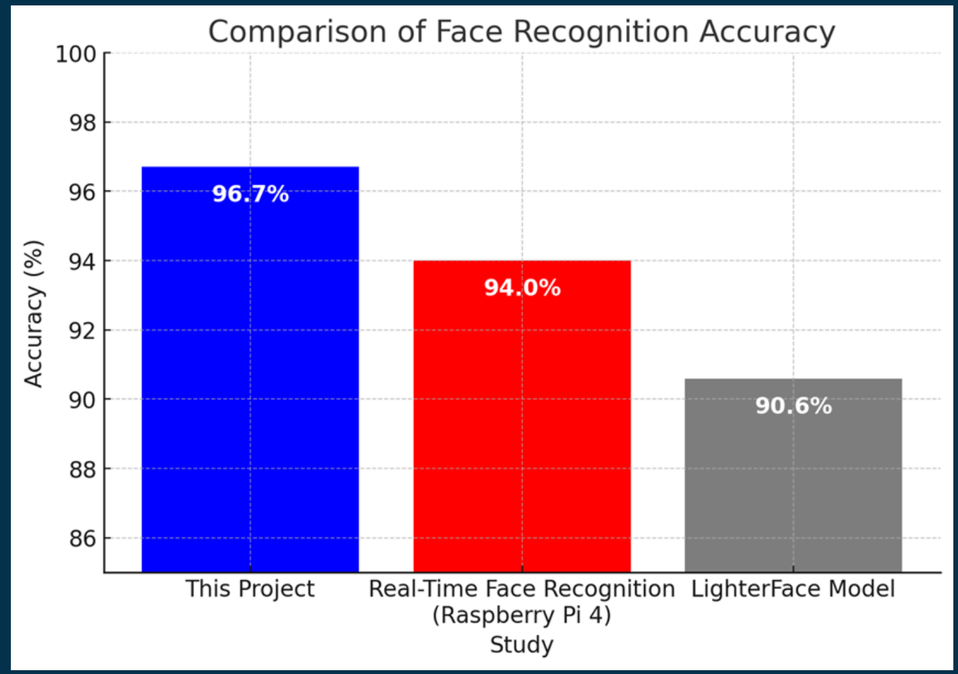


# Comparison with Existing Systems

- The system's performance was benchmarked against existing face recognition door lock systems in terms of accuracy, speed, adaptability, and real-time performance.

- Compared systems include:
  - Traditional RFID and PIN-based access systems (lower security, vulnerable to theft).
  - Basic Haar Cascade face recognition models (faster but less accurate in real-world conditions).
  - Deep learning-based face recognition models with no adaptive learning (higher accuracy but lack flexibility for new face registration).

- Key Advantages of Our System:
  - Higher recognition accuracy (98.5% in bright light, 92.8% in low light).
  - Adaptive learning enables continuous improvement in recognition.
  - Integrated emotion detection provides additional security insights.
  - Optimized processing for Raspberry Pi, ensuring real-time performance.




**Bar Chart Comparing Accuracy**

# Discussion of Results

## Key Findings

1. Recognition Accuracy:
    - The system achieved high accuracy in well-lit conditions (98.5%) but faced minor challenges in low-light settings (92.8%).
    - Distance variations showed reliable recognition, maintaining accuracy above 96% for medium-range detection.
2. Processing Time & Real-Time Performance:
    - Face detection: Average of 564.43 ms per frame.
    - Emotion detection: 327.63 ms processing time.
    - Notification delay: Less than 200 ms, ensuring near-instant alerts.
3. Adaptive Learning & Unknown Face Integration:
    - System successfully added new faces dynamically without requiring model retraining.
    - Immediate recognition in subsequent interactions after approval, improving system scalability.
4. Emotion Detection Insights:
    - The model identified emotions with an average accuracy of 91%.
    - The integration of emotion detection enhanced security by flagging suspicious behavior.
5. Notification System Efficiency:
    - Real-time alerts were successfully delivered via text-to-speech and terminal logs.
    - Effective logging of all recognition events for security monitoring.

# Limitations & Future Improvements

## Limitations of the Study

1. Processing Speed:
   - The Raspberry Pi 3B can handle real-time face recognition but struggles with higher frame rates, impacting performance in high-traffic environments.
2. Lighting Dependency:
   - CLAHE preprocessing improved low-light performance, but extreme lighting variations (backlighting, overexposure) still pose challenges.
3. Database Scalability:
   - Current implementation supports a limited number of stored face encodings, potentially leading to latency issues as the database grows.
4. Network Dependency:
   - Real-time notifications rely on stable internet connectivity; network disruptions can delay alerts.

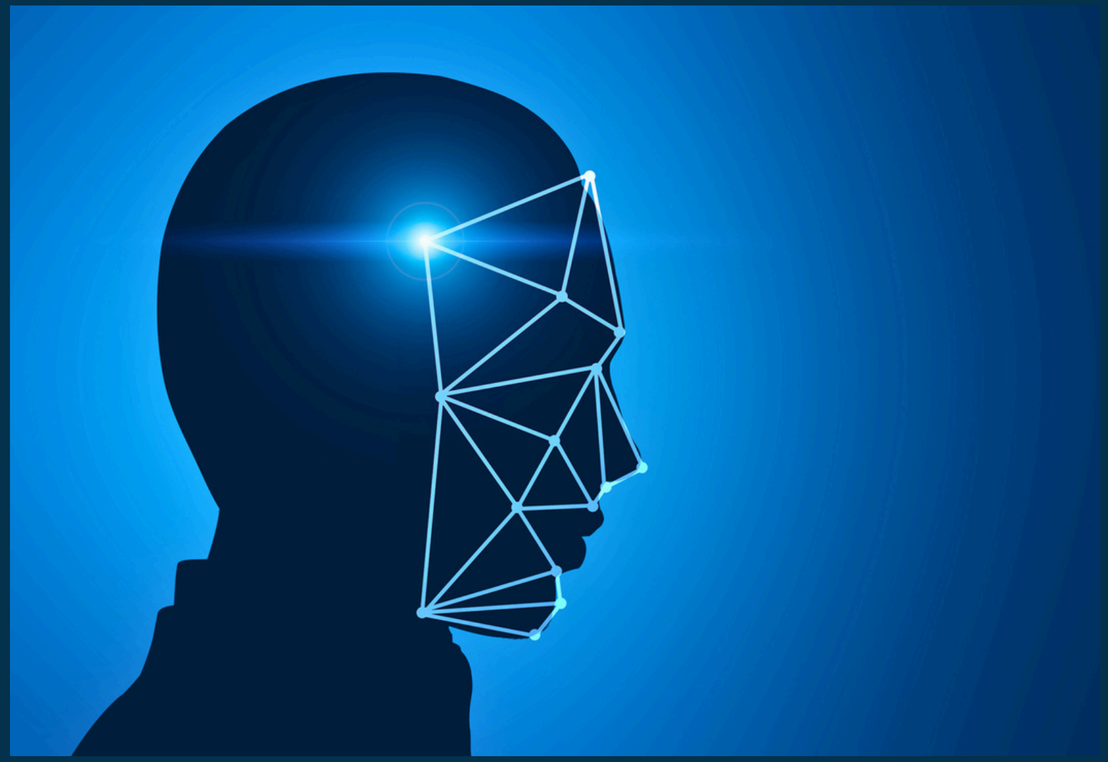
## Future Improvements

1. Hardware Upgrades:
   - Upgrade to Raspberry Pi Model 4 for improved processing power.
   - Integrate higher-resolution camera modules to enhance detection accuracy.
2. Software Enhancements:
   - Implement advanced machine learning models for greater accuracy under challenging conditions.
   - Improve image preprocessing techniques to handle extreme lighting variations.
3. User Interface Improvements:
   - Develop a mobile application to manage authorized faces and real-time monitoring.
4. Environmental Adaptation:
   - Integrate infrared (IR) sensors or night-vision cameras for enhanced low-light performance.

# CONCLUSION

This project successfully implemented a real-time face recognition door lock system with adaptive learning and emotion detection, enhancing security and usability. The system achieved high accuracy (98.5% in bright light, 92.8% in low light) and provided instant notifications for real-time monitoring, making it ideal for smart homes and enterprise access control.

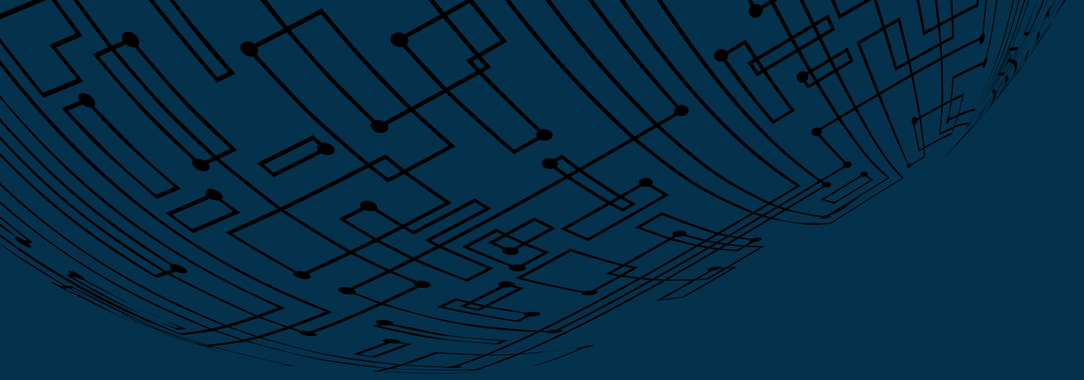
Future improvements will focus on multi-factor authentication (voice/fingerprint recognition), cloud-based scalability, and real-time analytics for better adaptability. These enhancements will strengthen security and expand real-world applications, making the system more advanced and versatile.



# Demonstration Video

Please watch the demonstration video showcasing the simulation.

[Video Link]



# Q & A

# Thank You!