ABSTRACT

This project presents the development of a face authentication system that combines AI-based face recognition running on a laptop with Arduino microcontroller communication and Telegram messaging integration. Leveraging Pre-trained face recognition models through OpenCV, the system achieves accurate and reliable face authentication capabilities. The integration of Arduino using PyFirmata facilitates seamless communication between the laptop and external hardware components, enabling real-time interaction and control. Additionally, the system utilizes the Telegram bot API to send warning messages to users via the Telegram messaging platform, ensuring timely notifications of unauthorized access attempts or security breaches. By combining these technologies, the project offers a comprehensive solution for secure access control and remote monitoring, suitable for various applications ranging from home security to workplace access management. Through this integration, users can benefit from advanced security features and proactive alerts, enhancing overall safety and peace of mind.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

AI	Artificial intelligence
CV	Computer vision
LCD	Liquid Crystal Diode
API	Application Programming Interface

CHAPTER-1

INTRODUCTION

In an era fraught with security concerns, traditional authentication methods like passwords and PINs are increasingly vulnerable to breaches, necessitating more resilient solutions. This project introduces an innovative face authentication system that merges cutting-edge technologies to fortify access control and bolster security measures.

At its core, the system harnesses artificial intelligence (AI) and face recognition, leveraging Google's advanced models accessible through OpenCV. Deep learning algorithms discern individuals based on facial characteristics, providing a secure means of authentication with precision and efficiency.

Supplementing the AI framework, Arduino microcontroller technology facilitates seamless interfacing with external hardware via PyFirmata, enabling real-time monitoring and control functionalities. This integration ensures adaptability and versatility across diverse deployment scenarios.

Moreover, Telegram messaging integration enhances user communication and notification capabilities. Utilizing the Telegram Bot API, the system dispatches instant alerts to users' devices upon detecting security breaches, empowering swift responses to potential threats.

This project demonstrates the efficacy of a comprehensive face authentication system, combining AI, Arduino, and Telegram integration to offer a robust solution for access control and security surveillance in various environments.

CHAPTER-2
LITERATURE SURVEY

Face Recognition: A Literature Review:

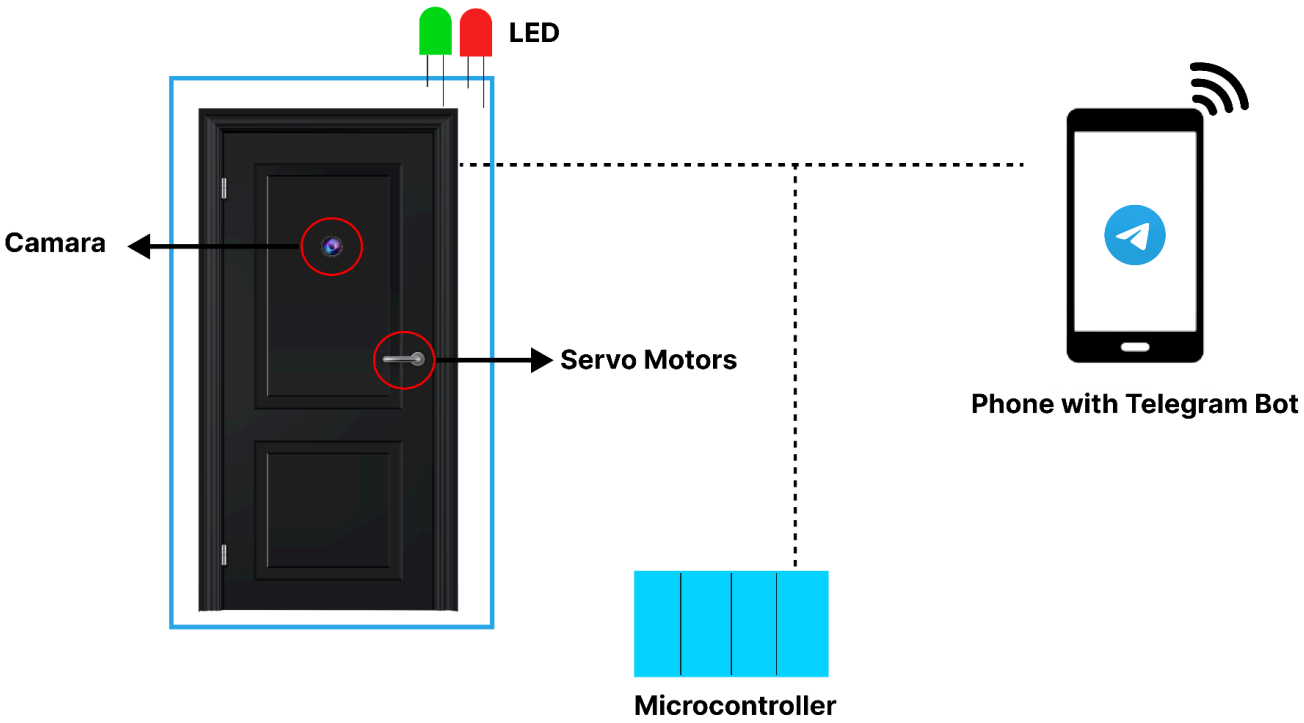
Face recognition has garnered significant attention due to its wide-ranging applications in entertainment, smart cards, security, law enforcement, and surveillance. This paper provides an overview of face recognition methods, focusing on feature extraction, distance measurement, and face recognition databases. It categorizes feature extraction methods into appearance-based and model-based approaches, highlighting techniques such as PCA, LDA, ICA, KPCA, KLDA, and EBGM. Additionally, it discusses the challenges faced in face recognition systems, including variability in facial information due to factors like lighting conditions and pose.

The paper delves into the importance of dimensionality reduction in face recognition, emphasizing the need for suitable methods to handle high-dimensional data and computational complexity, especially in large databases. It explores various applications of face recognition across security, criminal justice, surveillance, smart cards, and human-computer interactions, underscoring its versatility and significance in contemporary contexts.

Furthermore, the paper presents a generic face recognition system, detailing its three-step process of image input, identification/verification, and output. It discusses the role of linear and non-linear classifiers in feature extraction and dimensionality reduction, with examples of their application in real-world scenarios.

In conclusion, the paper provides insights into the advancements and challenges in face recognition technology, offering a comprehensive overview of methods, applications, and considerations in developing effective face recognition systems.

CHAPTER-3
SYSTEM ARCHITECTURE AND DESIGN



The system architecture integrates a Face Recognition Module with pre-trained models to detect and recognize faces. It manages a Database for storing known persons' information and utilizes Telegram Integration for sending notifications. A Decision-Making Module determines known or unknown faces, and Arduino Communication controls external hardware. Actions like door opening are executed based on recognition results, with thorough System Testing and Integration ensuring seamless functionality.

CHAPTER-4

METHODOLOGY

1. Face Recognition Model Integration:
- Integrate the pre-trained face recognition model into your system. Ensure compatibility with OpenCV for seamless integration.
- Utilize Google's pre-trained face recognition models available through OpenCV for robust face authentication.

2. Database of Known Persons:
- Create a database or a data structure to store known persons' information.
- Each entry should include the person's face embedding (a numerical representation of their face obtained from the recognition model) and their corresponding name or ID.

3. Face Recognition Logic:
- When a face is detected in a frame, compare its embedding with the embeddings of known persons in your database.
- If a match is found, retrieve the corresponding name or ID of the known person.
- If no match is found, consider the face as that of an unknown person.

4. Decision Making and Action:
- If a known person is detected, send a Telegram message indicating their name along with a predefined message.
- If an unknown person is detected, send a Telegram message indicating that an unknown person has been detected.
- Based on the detection results, trigger actions such as opening the door if a known person is detected.

5. Arduino Integration:
- Establish serial communication between the laptop/desktop and the Arduino microcontroller using the PyFirmata library.
- Configure the system to send commands to the Arduino when an unknown face is detected.
- Define commands or signals to be sent to the Arduino board based on detection results (e.g., open door signal).

3. Real-time Face Recognition:
- Develop a real-time face recognition system that captures video frames from a connected camera.
- Process each frame using the integrated pre-trained face recognition model to detect and recognize faces.

4. Telegram Integration:

- Utilize the Telegram Bot to enable communication with the Telegram messaging platform.
- Set up a Telegram bot and obtain its API token.
- Write functions to send messages to your Telegram bot.
- Define message templates for different scenarios (e.g., known person detected, unknown person detected).

5. Testing and Evaluation:

- Integrate all the components together.
- Test the system thoroughly to ensure correct detection, messaging, and action triggering.

6. Deployment:

- Deploy the integrated system in the surveillance environment, ensuring proper connectivity between components.
- Monitor the system in real-time and make any necessary adjustments to optimize its performance

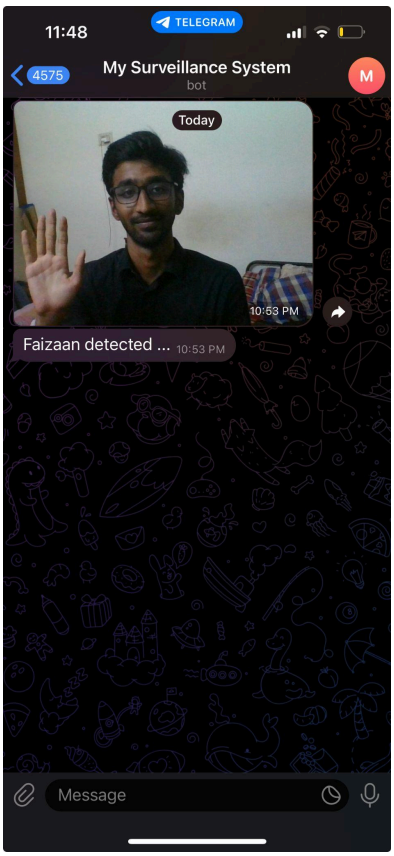
CHAPTER-5
CODING AND TESTING

```
import time
import cv2
from pyfirmata import Arduino, util
import face_recognition
import winsound
import requests

port = Arduino('COM3')
TOKEN = "7150869348:AAFbt6WepYxZd2e_NsGdU4MwYs0x6viAiW0"
warn = [False]
iter = util.Iterator(port)
iter.start()
servo = port.get_pin('d:9:s')
led = port.get_pin('d:13:o')
servo.write(0)
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
known_faces = ["./assets/1.jpg", "./assets/2.jpg", "./assets/3.jpg"]
names = ['elon Musk', 'Vishaal Sridhar', 'Faizaan']
known_faces_encodings = []

def beep():
    frequency = 2000  # Set Frequency To 2500 Hertz
    duration = 1500  # Set Duration To 1000 ms == 1 second
    winsound.Beep(frequency, duration)

def send_photo(chat_id = "5244701877"):
    file_opened = open("unkn.jpg", 'rb')
    params = {'chat_id': chat_id}
    files = {'photo': file_opened}
    url = f"https://api.telegram.org/bot{TOKEN}/sendPhoto"
    # "https://api.telegram.org/bot' + botToken + '/sendPhoto -F chat_id=' + chat_id + " -F photo=@" + imageFile"
```

```
    resp = requests.post(url, params, files=files)
    print(resp)

def Train(images = known_faces):
    for loc in images:
        img = face_recognition.load_image_file(loc)
        enc = face_recognition.face_encodings(img)[0]
        known_faces_encodings.append(enc)

def sendMessage(message):
    warn[0] = True
    chat_id = "5244701877"
                                                                    url = f"https://api.telegram.org/bot{TOKEN}/sendMessage?chat_id={chat_id}&text={message}"
    print(requests.get(url).json())

def detect_known_faces(frame):
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x, y, w, h) in faces:
        face_roi = gray[y:y+h, x:x+w]
        face_encoding = face_recognition.face_encodings(face_roi)
                matches = face_recognition.compare_faces(known_faces_encodings, face_encoding)
        if True in matches:
            return True
        else:
            return False
def detect(frame):
    face_locations = face_recognition.face_locations(frame)
    face_encodings = face_recognition.face_encodings(frame, face_locations)
    for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
        matches = face_recognition.compare_faces(known_faces_encodings,
```

```
face_encoding)
        print(True in matches)
        if True in matches:
            led.write(1)
            for i in range(len(matches)):
                if(matches[i] == True):
                    cv2.imwrite("unkn.jpg", frame)
                    send_photo()
                    sendMessage(f"{names[i]} detected ...")
            beep()
            time.sleep(1)
            servo.write(180)
            time.sleep(5)
            servo.write(0)
            led.write(0)
        else:
            sendMessage("Unkown person detected..")
            cv2.imwrite("unkn.jpg", frame)
            send_photo()
Train()
vid = cv2.VideoCapture(0)
i = 0
while(True):
    print("warn ", warn)
    if(i == 10):
        i = 0
        warn[0] = False
    if(warn[0]):
        i += 1
        time.sleep(1)
        continue
    ret, frame = vid.read()
    detect(frame)
```

CHAPTER-6
SCREENSHOTS AND RESULT



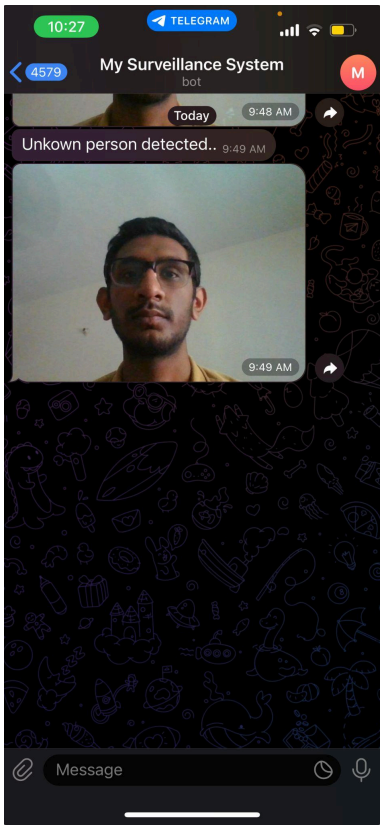
6.1 Telegram Notifications:

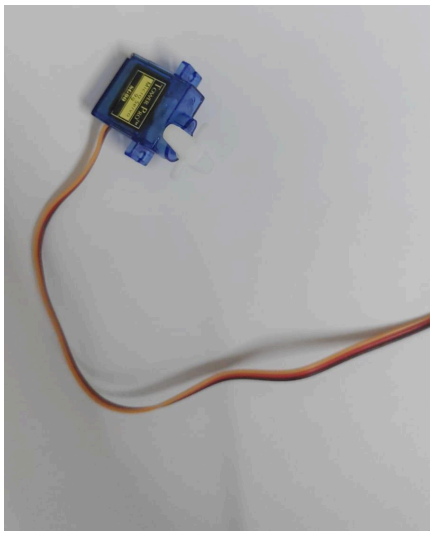
Known Person Detected Notification:

- When any known person is detected it sends the notification that <<Name>> detected.

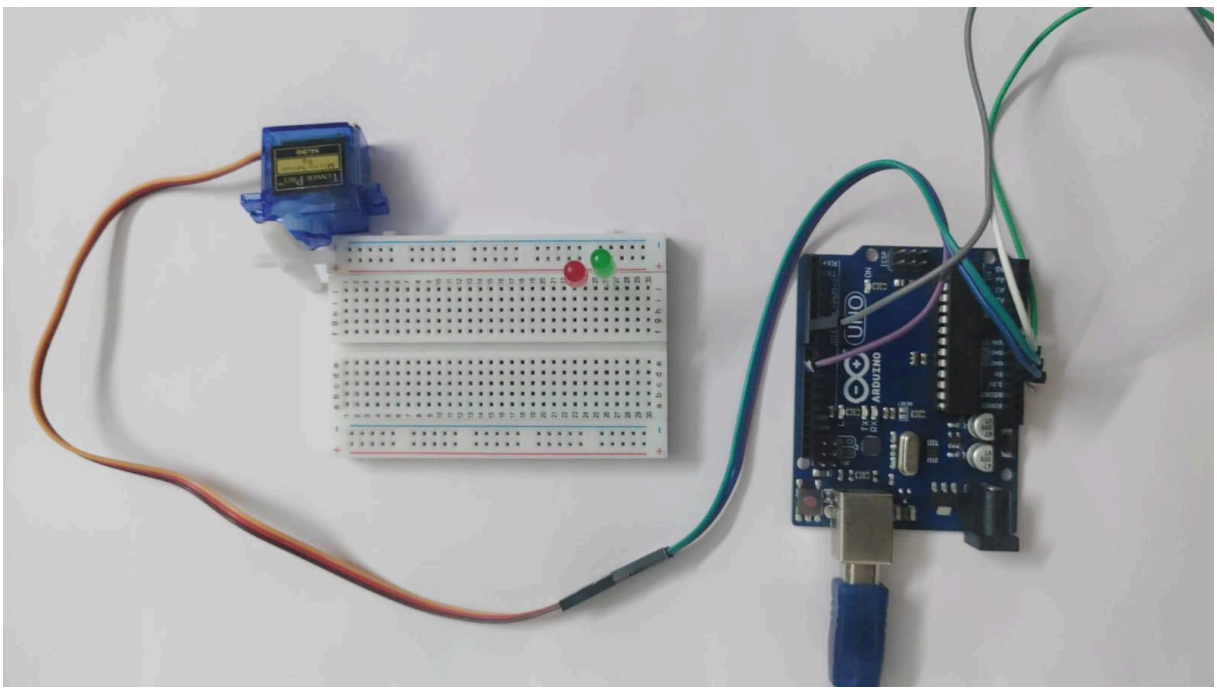
6.2 Unknown Person Detected Notification:

- When any unknown person is detected it sends the notification that the unknown is detected.





6.3 Arduino Control: When a known person is detected it will open the door



CHAPTER-7
CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion:

In conclusion, the implemented AI-based surveillance system, integrating face recognition technology with IoT devices and Telegram notifications, presents a robust solution for access control and security surveillance. By harnessing the power of AI and computer vision, the system efficiently identifies known individuals while promptly alerting users to the presence of unknown persons, enhancing security measures in various environments.

The successful integration of Google's face recognition models through OpenCV ensures reliable and accurate face authentication, bolstering the system's effectiveness in identifying authorized individuals. The communication between the AI model running on a laptop and the Arduino microcontroller via serial communication streamlines real-time monitoring and control functionalities, enabling seamless interaction with external hardware components.

Furthermore, the incorporation of Telegram notifications enhances user communication and notification capabilities, providing instant alerts to users' devices upon detecting security breaches or unauthorized access attempts. This proactive approach empowers users to respond swiftly to potential threats, thereby safeguarding assets and ensuring a heightened level of security.

7.2 Future enhancements:

1. Enhanced Facial Recognition Accuracy: Continuously refine and optimize the face recognition algorithms to improve accuracy, particularly in challenging lighting conditions or with occluded faces.

2. Multi-Camera Support: Expand the system to support multiple cameras for comprehensive surveillance coverage, allowing for more extensive monitoring of larger areas.

3. Integration with Cloud Services: Implement cloud integration to store and analyze captured data, enabling long-term storage, advanced analytics, and remote access to surveillance footage.

4. Customizable Notifications: Provide users with the ability to customize notification preferences, allowing them to specify the type of alerts they wish to receive and the frequency of notifications.

5. Mobile Application Development: Develop a dedicated mobile application to provide users with intuitive control and monitoring capabilities, facilitating remote access and management of the surveillance system.

By pursuing these future enhancements, the AI-based surveillance system can evolve into a more sophisticated and comprehensive security solution, addressing a wider range of user needs and ensuring robust protection against security threats.

CHAPTER-8
REFERENCES

1. Telegram API Documentation. (n.d.). Retrieved from https://core.telegram.org/bots/api

2. PyFirmata. (n.d.). Retrieved from https://pypi.org/project/pyFirmata/

3. OpenCV. (n.d.). Retrieved from https://opencv.org/

4. Arduino. (n.d.). Retrieved from https://www.arduino.cc/

5. Face Recognition with Python, in Under 25 Lines of Code. (n.d.). Retrieved from https://towardsdatascience.com/face-recognition-with-python-in-under-25-lines-of-code-efe074f617c

6. Telegram Bot API. (n.d.). Retrieved from https://core.telegram.org/bots/api