Home Security System Based On Facial Recognition

Dola Gobinda Padhan
Professor
Department of EEE, GRIET
Hyderabad, India
dgobinda@gmail.com

Sai Nikhila Varma
UG Student
Department of EEE, GRIET
Hyderabad, India
nikhilavarma11@gmail.com

Venkateshwari C
UG Student
Department of EEE, GRIET
Hyderabad, India
cvenkateshwari2002@gmail.com

Marakala Divya
UG Student
Department of EEE, GRIET
Hyderabad, India
divyamarakala26@gmail.com

Suguna Manasa
UG Student
Department of EEE, GRIET
Hyderabad, India
manasasharma1731@gmail.com

B Pakkiraiah
Associate *Professor*
Department of EEE, GRIET
Hyderabad, India
pakki1988@gmail.com

2023 IEEE 3rd International Conference on Sustainable Energy and Future Electric Transportation (SEFET) | 979-8-3503-1997-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/SEFET57834.2023.10244798

***Abstract*—Security systems serve the major functions of protection and safety. Home security is a major predicament in today's world. It is proven that facial recognition, which is a relatively new technology, is the most efficient way of providing security. The high-end security systems used today are very expensive and use biometrics like fingerprint or iris scanners. These systems require additional equipment for recognition which can be replaced by a simple camera while using facial recognition. Our project is aimed at creating an efficient security system incorporating the concepts of Artificial Intelligence (AI) and Machine Learning (ML) to implement facial recognition. Histograms of Oriented Gradients is the algorithm used for facial recognition. The usage of doorbells is also uprooted to attain a more automated approach to detecting the presence of someone at the door by using facial detection. The above goals are carried out with the use of OpenCV, Imutils and Dlib libraries in Python, with a Raspberry Pi as the processing unit of the security system. The door lock, controlled by a stepper motor, operates immediately once the person detected is recognized. Application of the Internet of Things (IoT) and the Global System for Mobile Communications (GSM) is also incorporated in the situation when the detected person is not recognized.**

Keywords—Facial Recognition, OpenCV, Raspberry Pi, Flask Stepper motor

I. INTRODUCTION

In today's world, home security has become a serious concern. Traditional home security systems are easily broken and obsolete. This leads to robbery, which necessitates the installation of a pricey security system. A study shows that, In the United States, a house burglary occurs every 13 seconds, 4 times per minute, 240 times per hour, and approximately 6,000 times each day. 88 percent of all burglaries are domestic, 77 percent of all crimes are property crimes, and 38 percent of all robberies are done with weapons, and identity theft is the fastest rising crime in the United States, Canada, and the United Kingdom, according to data. Within the next 20 years, three out of every four homes in the United States will be broken into. Nobody's house is safe, and the security solutions that have been devised so far are easily exploitable by burglars. Traditional methods of home security are easy to breach and lead to theft. To secure our homes, we must install a very advanced security system available, which may be rather costly. Moreover, for such expensive security systems that are available in the market, they are not scalable and are very restrained. As technology advances, a slew of new security concerns emerges. Existing security techniques contain vulnerabilities and are vulnerable to hacking. Even intruders have found their way to take over these gadgets. The presence of someone in front of the door isn't detected unless they ring the doorbell or give a knock on the door, making it impossible to know even if someone is physically overriding the security system. So, to avoid such situations, we have to develop the system in such a way that no one should get an intrusion into the system.

We aim to use Computer Vision technology to bring high-level security to our homes. Computer Vision is an up-and-coming technology that has had a significant influence on the modern world. Computer vision is defined by IBM as "A field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos, and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand." Facial recognition is an application of Computer Vision. Training the computer to recognize faces is the goal of implementing this method. Facial recognition to access gadgets is everywhere, for passwords, phones, or laptops, hence this project is to build on that application and use the facial recognition technology to access the locks of our houses. There are in numerous techniques to train a computer for facial recognition such as 'Histograms of Oriented Gradients', 'Linear Binary Histograms Patterns', 'Convolution Neural Networks', 'Eigenfaces', 'FaceNet', etc. The best algorithm for the application of the project is Histograms of Oriented Gradients.

A trained computer capable of recognizing faces will make the decision to unlock the door or keep it locked. The adoption of IoT is an override when facial recognition is not applicable. IoT will improve some security levels as well as allow for remote access and control of the system. It is simple to automate the system. As a result, by extending this security system, we may continue to construct smart houses. For system development, we will utilize a Raspberry Pi microcontroller board, a Pi camera module for facial recognition, and a stepper motor to control a door lock. The Pi camera module is employed as a means of surveillance at the front door. By eliminating doorbells, we can automatically detect the presence of someone hovering or snooping around at the door using object detection. Upon detecting, an alert is sent to the homeowner, in the form of a phone call using GSM, for them to communicate with the person at the door as well

979-8-3503-1997-2/23/$31.00 ©2023 IEEE
Authorized licensed use limited to: MULTIMEDIA UNIVERSITY. Downloaded on April 24,2024 at 03:18:26 UTC from IEEE Xplore. Restrictions apply.

as check the activity through the live feed and recognize the person.

The objective of this project is to develop a home security system based on facial recognition using the concepts of Computer Vision and the Internet of Things. Facial Recognition is implemented in three steps – Facial detection, Training, and Recognition. The model of faces is recognized by the technique Histograms of Oriented Gradients (HOG). The trained 3 recognizer will identify the faces in front of the camera and operate to unlock the door. An IoT-based automated door lock system is developed which can be controlled remotely when the person is not identified by the recognizer. The security system, placed at the front door, is meant to be monitored and operated from afar. The development of the system using Computer Vision and IoT will make significant changes in modern security technologies. The use of a micro-controller board will establish simplicity and flexibility in the system, easing scalability to incorporate any additional features when required. The challenges faced by the existing security system to detect any potential threats are also addressed by this system. This feature is aimed to eliminate any physical vulnerabilities of the security system.

II. Block Diagram

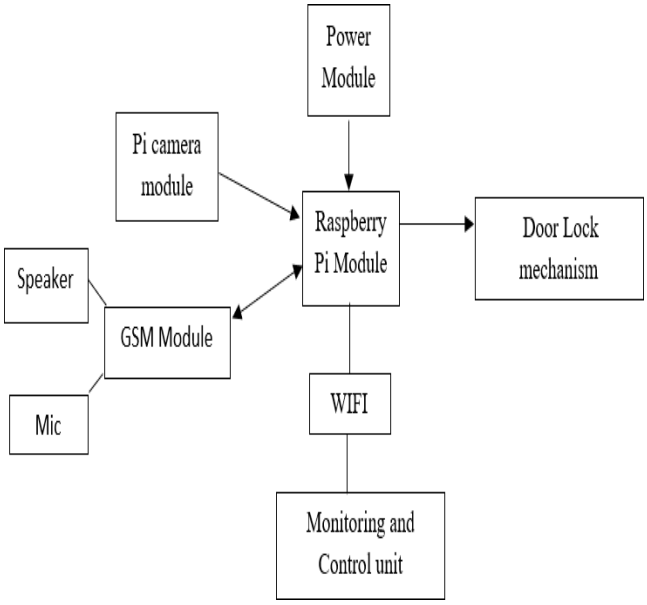

Fig. 1. Block diagram

The block diagram (Fig.1.) depicts the various interfaces and components used in the project. The Raspberry Pi 4 Model B is the brain of the project to which various interfaces are connected. The inputs for the Raspberry Pi come from its DC power source of 5V and 3A, and the Pi camera. The Pi sends output signals to stepper motor driver, A4899 and the stepper motor which constitutes the door lock mechanism. Operating stepper motor requires another DC power source, supplying 12V and 1A. A Global System for Mobile Communications (GSM) module is connected to the Pi which has a serial communication interface. The speaker and mic connected to the GSM are used for the communication system of the security system. The Pi is remotely controlled over WIFI using applications such as Putty and VNC Viewer.

III. Operation

The goal of the project is to create a security system that can be controlled and monitored from afar. The ability to run a Raspberry Pi wirelessly is the first condition for achieving this objective. Putty and VNC Viewer are two applications that may be used to operate the Raspberry Pi wirelessly. It is critical that both the Raspberry Pi and the device that controls it be connected to the same network. Facial Recognition requires some dedicated libraries in Python which facilitate Computer Vision. These libraries are:

- OpenCV – the original source for computer vision containing various machine learning algorithms.
- Face_recognition – acknowledging faces and training the system to identify these faces is done using this library. It is responsible for computing the bounding box around each face, compute facial embeddings, and compare faces in the encoding dataset.
- Imutils – it consists of a variety of functions which make basic image processing operations like skeletonization, resizing, translation, rotation, detecting edges, sorting contours, displaying Matplotlib images, very convenient and much easier for use with OpenCV.
- Pickle – it is responsible for converting python objects into byte stream and serves as file or database for the byte stream obtained.

The next step is to implement facial recognition which consists of three phases:

1. Data Gathering and Face Detection using Haar Cascade Classifiers.

2. Training Recognizer using Histograms of Oriented Patterns.

3. Facial Recognition implemented with the trained recognizer.

In the first phase of Date Gathering, we simply collect images of people who are to be recognized. Face detection using Haar Cascade Classifiers is a method where specific features of an image are detected. Haar Cascade is defined as "An object detection algorithm used to identify faces in an image or a real time video". Classifiers in this cascade refer to the specific feature to be detected, such as eyes, face, upper body, lower body, etc. For face detection, we use the "Haar Cascade Frontal Face" XML file which is a trained 18 model available on the OpenCV repository on GitHub. Once the faces are detected, the model is trained further to recognize them by naming and identifying.

Amongst the various algorithms available for training a facial recognition model, we choose the Histograms or Oriented Gradients (HOG) method due to its high precision, accuracy in all lighting, and faster action. The HOG algorithm focuses on structure and shape of the object by extracting the magnitude and direction (gradient and orientation) of pixels of localized portions of an image. It emphasizes on feature extraction, while discarding facial expressions, background, color and intensity of lighting. All the features of the image are collected in a vector form, usually of the size 36x1, giving a total of 3780 features for an image of size 64x128. Steps involved in HOG algorithm are as follows:

Authorized licensed use limited to: MULTIMEDIA UNIVERSITY. Downloaded on April 24,2024 at 03:18:26 UTC from IEEE Xplore. Restrictions apply.

1. Pre-process the image and bring down the width to height ratio to 1:2.

2. Calculate the gradient for every pixel in the image. Gradients are the small change in the x and y directions.

3. Using the gradients calculated in the last step, the magnitude and direction for each pixel value will be determined. The Pythagoras theorem will be used in this step.

This feature set, once trained to recognize, is encoded into a byte stream for storage using pickle. A file called 'encodings.pickle' is created for each face. These encodings act as the reference to recognize faces.

In the final phase of facial recognition, the program code involves actions to be taken when a face is recognized and when a face is not recognized. When a face is recognized, the stepper motor rotates in the direction to unlock the door. It is held in the unlock position for 10 seconds, and the rotates in the direction to lock the door. When a face is not recognized, a phone call is made to the home owner using the GSM module. The GSM module, connected with a speaker and a mic acts as communication system between the home owner and the unrecognized person. The home owner can check the person at the front door using his remote access to the Raspberry Pi and decide if the door should be unlocked. The incorporation of the Internet of Things involved creating a webpage using a micro web framework called Flask, which used to control the stepper motor. The webpage allows the home owner to rotated the motor in the direction to unlock the door. The developed webpage uses POST requests that establishes a data communication to the servers in the HTML form.

IV. Algorithms

A. Data Gathering

The following steps are involved for the programming to gather the data set to train the recogniser:

1. Import the necessary libraries, including cv2 for OpenCV, PiCamera for interfacing with the Raspberry Pi camera, and PiRGBArray for working with the camera's raw RGB image array.
2. Set the desired camera resolution and framerate.
3. Create an instance of the PiCamera class and a PiRGBArray object to capture and store the camera frames.
4. Initialize a counter for the image filenames and enter an infinite loop to continuously capture frames from the camera.
5. For each captured frame, display it using OpenCV's cv2.imshow() function with a window title "Press Space to take a photo".
6. Wait for a key press using cv2.waitKey() and store the pressed key's ASCII value in a variable k.
7. If the pressed key is the ESC key (ASCII value 27), break out of the loop and proceed to step 12.
8. If the pressed key is the SPACE key (ASCII value 32), construct the image filename using the name variable and the image counter, and save the image to the file using cv2.imwrite().
9. Print a message indicating that the image has been written with the filename.
10. Increment the image counter by 1 to prepare for the next image capture.
11. Truncate the rawCapture object to clear the buffer for the next frame.
12. Check if the ESC key was pressed to exit the loop. If so, print a message indicating that the program is closing and break out of the loop.
13. Close all OpenCV windows using cv2.destroyAllWindows().

B. Training the Recognizer using the Data Gathered

The recognizer is trained based on Histograms of Oriented Gradients is programmed using the following steps:

1. Import the necessary libraries, including imutils for working with image paths, face_recognition for face detection and recognition, pickle for serialization, and cv2 for OpenCV.
2. Get the list of image paths from the "dataset" directory using paths.list_images() function from imutils.
3. Initialize empty lists to store the known encodings and names.
4. Loop over the image paths using a for loop, and for each image path:
   - Extract the person's name from the image path by splitting the path using os.path.sep as a separator and taking the second-to-last element.
   - Load the image using cv2.imread() function from OpenCV.
   - Convert the image from RGB (OpenCV ordering) to dlib ordering (RGB) using cv2.cvtColor() function from OpenCV.
   - Detect the (x, y)-coordinates of the bounding boxes corresponding to each face in the image using face_recognition.face_locations() function from face_recognition.
   - Compute the facial embeddings for the detected faces using face_recognition.face_encodings() function from face_recognition.
   - Loop over the encodings using another for loop, and for each encoding:

     Append the encoding and the corresponding name to the lists of known encodings and names, respectively.
6. Serialize the known encodings and names to disk using pickle.dumps() function from pickle, and write the serialized data to a binary file using open() function with write binary mode.
7. Close the file using close() method

C. Facial Recognition using the trained recogniser

Facial Recognition is finally programmed with the help of the trained recogniser using the data gathered. Here, instructions for the actions to be taken when a face is recognised, a face is not recognised, and to save the images of all the faces detected, are given in the following steps:

1. Import necessary libraries: imutils, face_recognition, pickle, time, serial, cv2, os, and RPi.GPIO.
2. Define constants for GPIO pins (DIR and STEP) and other variables (CW, CCW, SPR, and delay).
3. Set up GPIO mode and pin configurations using GPIO.setmode() and GPIO.setup() functions.
4. Create a directory "detected_faces" if it does not exist using os.makedirs() function.

Authorized licensed use limited to: MULTIMEDIA UNIVERSITY. Downloaded on April 24,2024 at 03:18:26 UTC from IEEE Xplore. Restrictions apply.

5. Set up serial communication with the specified port and baud rate using serial.Serial() function and flush the buffer.
6. Define a function make_call(number) to make a call using the AT commands.
7. Initialize variables for face recognition (currentname) and door status (doorUnlock).
8. Load the pre-trained face encodings and face detector using pickle.loads() and cv2.CascadeClassifier() functions.
9. Start the video stream using VideoStream() class and wait for 2 seconds for the camera to warm up.
10. Start the FPS counter using FPS().start() function.
11. Enter a loop to continuously read frames from the video stream.
12. Resize the frame, convert it to grayscale and RGB format.
13. Detect faces in the frame using cv2.CascadeClassifier.detectMultiScale() function and store the bounding box coordinates in rects.
14. Convert the bounding box coordinates to the format required by face_recognition.face_encodings() function and encode the detected faces using face_recognition.face_encodings() function.
15. Loop through the encodings and compare them with the pre-trained encodings using face_recognition.compare_faces() function.
16. If a match is found, update the currentname variable and unlock the door by rotating the stepper motor in one direction using GPIO pins.
17. Keep the door unlocked for 10 seconds (prevTime and doorUnlock variables) and then lock the door by rotating the stepper motor in the opposite direction.
18. Draw rectangles and labels around the detected faces in the frame using cv2.rectangle() and cv2.putText() functions.
19. Save the detected faces as separate images in the "detected_faces" directory with the timestamp and name using cv2.imwrite() function.
20. Display the processed frame with rectangles and labels using cv2.imshow() function.
21. Exit the loop if 'q' key is pressed or make a call if "Unknown" is detected using cv2.waitKey() function.
22. Update the FPS counter using fps.update() function.
23. Stop the FPS counter and print the elapsed time and approximate FPS using fps.stop() function.
24. Close all open windows, stop the video stream, and clean up GPIO using cv2.destroyAllWindows(), vs.stop(), and GPIO.cleanup() functions respectively.

D. Webpage for stepper motor control

The webpage fundamentally creates an IoT server with Flask providing the framework for the application. The programming of the webpage requires the following steps to be carried out:

1. Import the necessary modules: Flask for web framework and rendering HTML templates, and RpiMotorLib for controlling the stepper motor.
2. Create a Flask app object.
3. Define the stepper motor parameters such as direction, step, and GPIO pins.
4. Create a stepper motor object using the RpiMotorLib library.
5. Define the HTML template for the home page which displays a form with buttons to control the direction of the stepper motor (lock or unlock).
6. Define a Flask route for the home page ('/') which renders the home page template.
7. Define a Flask route for the '/move' endpoint with the 'POST' method to handle form submission.
8. Inside the '/move' route function, retrieve the direction value from the form data using Flask's 'request' object.
9. Set the direction of the stepper motor based on the retrieved value, using the motor object's 'motor_go' method with the appropriate parameters for direction, step mode, speed, and delay.
10. Stop the motor using the motor object's 'motor_stop' method.
11. Redirect back to the home page using Flask's 'render_template_string' method with the home page template.
12. Start the Flask app in debug mode if the script is run directly

V. HARDWARE RESULTS

A folder named ‘facial_recognition’ is created in the Raspberry Pi home folder. The programs for creating the dataset, training the model, and running facial recognition to operate a relay are all added in this folder. The XML file for Haar Cascade Frontal Face Classifier, haarcascade_frontalface_default.xml, is also included in this folder.

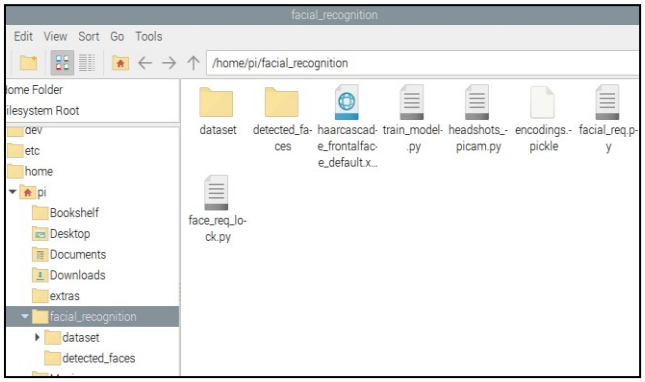

Fig. 2. ‘facial_recogntion’ Folder on Raspberry Pi

The folder called ‘dataset’ is created in the ‘facial_recognition’ folder (Fig.2.), where the photographs of the people who are to be recognized by the algorithm are included. Each person requires a separate folder with their name in the dataset as shown in Fig.3. After the dataset is created with folders with people’s names, the program for data gathering is run, capturing the images, and creating an image directory as shown in Fig.4. The created dataset is then used to train the model for facial recognition. We run the program to train the recognizer and obtain the encoded feature set. The output to this program is seen as the file ‘encodings.pickle’ in Fig. 2.

Authorized licensed use limited to: MULTIMEDIA UNIVERSITY. Downloaded on April 24,2024 at 03:18:26 UTC from IEEE Xplore. Restrictions apply.

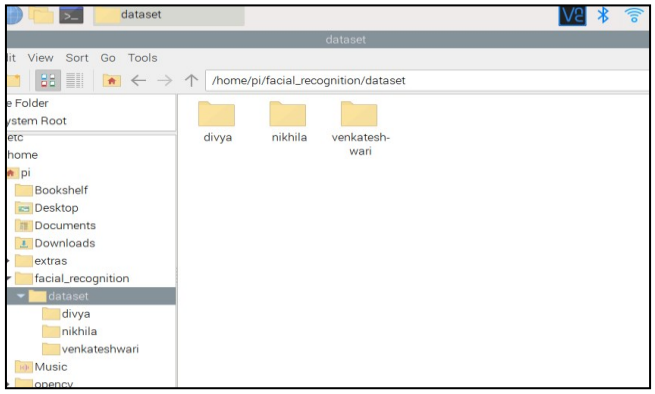

Fig. 3. 'dataset' Folder on Raspberry Pi

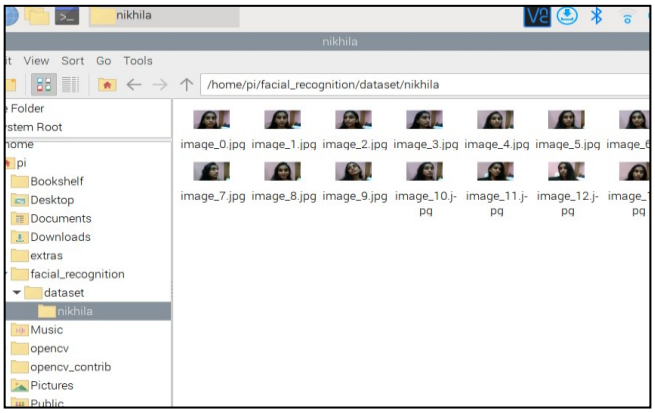

Fig. 4. Data Gathering Output

The webpage designed for this security system using the Flask framework has two buttons, one to unlock the door and the other to lock the door.

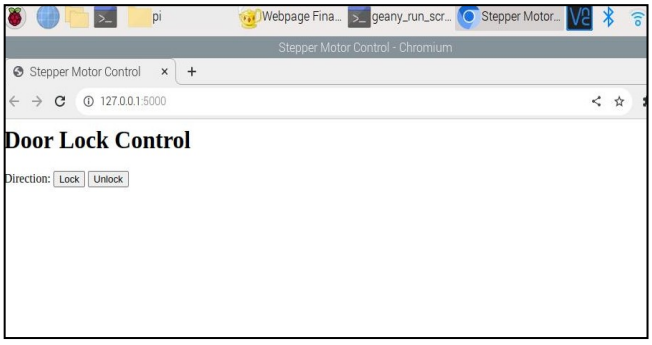

Fig. 5. Designed Webpage

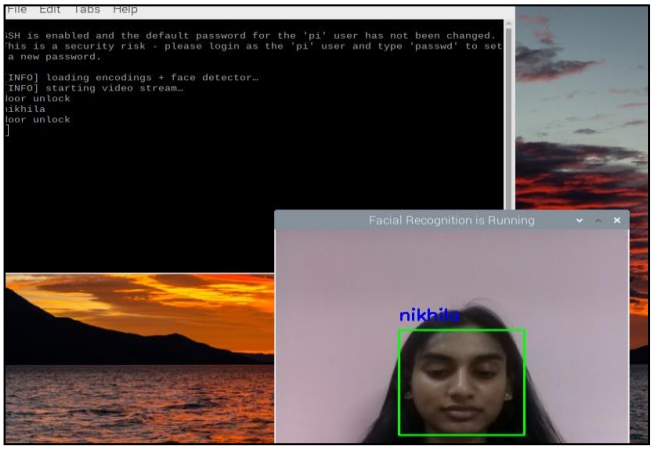

Fig. 6. Facial Recognition Implementation

Facial recognition implementation prompts a window on the raspberry pi (Fig.6.) that continuously displays the view outside the door, along with providing the status of the door lock.

The hardware model for the proposed security system is a miniature replica of the typical front door of a house. The front view of the model visualizes the components which serve as the eyes, ears, and mouth of the security system, all facing towards the outside of the house. These components include the Pi Camera, a speaker, and a mic. The back view of the model, which is on the side of the door facing inside the house, contains a compact box which houses the remaining components such as the Raspberry Pi, GSM, stepper motor driver, and their respective wiring. The stepper motor, which controls the door lock, is fixed to the key of the lock present on the inside of the house.

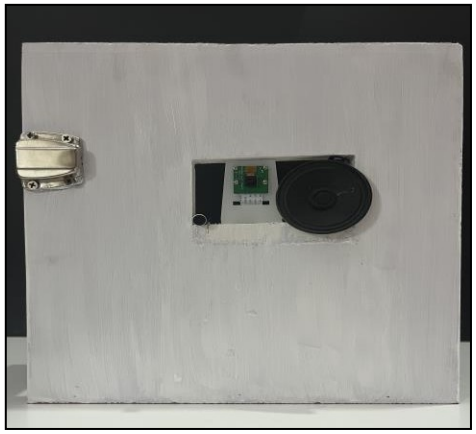

Fig. 7. Front View of Hardware Model

Fig. 8. Back View of Hardware Model

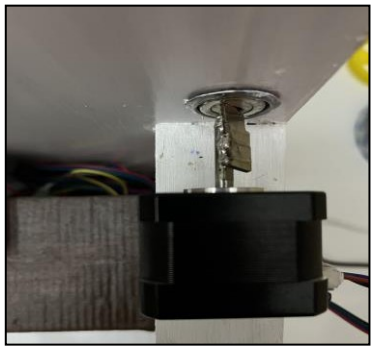

Fig. 9. Stepper Motor Connected to Door Lock Key

Authorized licensed use limited to: MULTIMEDIA UNIVERSITY. Downloaded on April 24,2024 at 03:18:26 UTC from IEEE Xplore. Restrictions apply.

VI. Conclusion

In this paper, the development of a home security system is discussed. The envisioned automated security system is energy-efficient, and compact compared to the ones available in the market. The system is more efficient and capable of real-time applications due to the use of Facial recognition and IoT. In our proposed methodology, the output achieved is close to almost maximum accuracy. The structure is implemented on WAN, which allows the homeowner to monitor and operate his home security system from a distance. The application of the Internet of Things IoT in this system allows remote control and monitoring. The system is flexible to alterations and the layout can be readily changed without disrupting the scheme's supporting parts. Real-time facial recognition using Histograms of Oriented Gradients ensures that the recognition rate is high and undisruptive.

References

[1] Sandesh Kulkarni, Minakshee Bagul, Akansha Dukare, Prof. Archana Gaikwad "Face Recognition System Using IoT", International Journal of Scientific Technology Research, November 2017.

[2] Manoj R. Dhobale, Rekha Y. Biradar, Raju R. Parwar, Sharad A. Awatade "Smart Home Security System using IoT, Face Recognition and Raspberry Pi", International Journal of Computer Applications, April 2020.

[3] A. R. Syafeeza, M. K. Mohd Fitri Alif, Y. Nursyifaa Athirah, A. S. Jaafar, A. H. Norihan, M. S. Saleha "IoT based facial recognition door access control home security system using raspberry pi", International Journal of Power Electronics and Drive System (IJPEDS), March 2020.

[4] Mansour H. Assaf, Ronald Mootoo, Sunil R. Das, Emil M. Petriu, Voicu Groza, and Satyendra Biswas "Sensor-Based Home Automation and Security System", IEEE Instrumentation and Measurement Technology Conference, May 2012.

[5] Hteik Htar Lwin, Aung Soe Khaing, Hla Myo Tun, "Automatic Door Access System Using Face Recognition", International Journal of Scientific and Technology Research Volume 4, Issue 06, June 2015.

Authorized licensed use limited to: MULTIMEDIA UNIVERSITY. Downloaded on April 24,2024 at 03:18:26 UTC from IEEE Xplore. Restrictions apply.