Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Technology (FEM)

Bachelor Thesis

Security System using Raspberry Pi

Vishwa Vachhani

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

BACHELOR THESIS ASSIGNMENT

Bc. Vishwa Jitubhai Vachhani

Informatics

Thesis title

Security system using Raspberry Pi

Objectives of thesis

The main objective of the thesis is to devise a smart security system utilizing microcontrollers like Raspberry Pi by integrating cameras and motion sensors into a web application. The microcontroller will control the motion sensors and video cameras and detect the presence of an intruder utilizing the Computer Vision Technique (CVT). If an unauthorized entry is detected, the microcontroller will generate an alert and process it.

Methodology

The methodology of solving the theoretical part of the diploma thesis will be based on the study and analysis of professional information sources. Based on the knowledge gained in the theoretical part of the work, individual methods of initial contact will be implemented in the practical part into the experimental application.

Using Raspberry Pi, the student will create the security system from his phone and laptop with internet access. The student will connect with General Purpose Input Output Ports using the Python programming language, a basic database connection will be established. AWS integration will be utilized as the database. An open CV module will be used to do face detection and identity recognition. To represent the exhibited system we can perform the steps below. The student will utilize Open CV to see whether there is any motion.

Based on the synthesis of theoretical knowledge and the results of the practical part, the conclusions of the work will be formulated.

The proposed extent of the thesis

40-50

Keywords

security system, raspberry, motion sensor, Thermal sensor, Security Analysis, Motion Security, Thermal Security, IoT, Internet Of Things

---

Recommended information sources

al, K. e. (2017). Raspberry Pi Network .

Anitha A, K. S. (2016). A Cyber defence using artificial home automation system using IoT. International Journal of Pharmacy and Technology, 25358-64.

Anitha A, P. G. (2016). Cyber defence using Artificial Intelligence . International Journal of Pharmacy and Technology, 25352-57.

Apthorpe N, R. D. (2017). A smart home is no castle: privacy vulnerabilities of encrypted IoT traffic.

Cristian C, U. A. (2016). Energy efficiency and robustness for IoT: building a smart home security system . Faculty of Automatic Control and Computers University Politehnica of Bucharest, Bucharest, Romania , 43.

Dorri A, K. S. (2017). Blockchain for IoT security and privacy: the case study of a smart home. 2017 IEEE International Conference on Pervasive Computing and Communications Workshops. IEEE.

Sfar AR, N. E. (2018). Digit Commun Netw. A roadmap for security challenges in the internet of things, 37-118.

Expected date of thesis defence

2022/23 WS – FEM

The Bachelor Thesis Supervisor

Ing. Tomáš Vokoun

Supervising department

Department of Information Technologies

Electronic approval: 23. 8. 2021

doc. Ing. Jiří Vaněk, Ph.D.

Head of department

Electronic approval: 5. 10. 2021

Ing. Martin Pelikán, Ph.D.

Dean

Prague on 02. 03. 2023

Declaration

I declare that I have worked on my bachelor thesis titled "**Security System using Raspberry Pi**" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the bachelor thesis, I declare that the thesis does not break any copyrights.

In Prague on 2023 _____

Acknowledgment

I would like to express my gratitude towards my family and friends for supporting throughout the journey of travelling to Czech Republic and attending university over here.

I would like to thank **The Czech University of Life Sciences Prague** and my supervisor **Ing. Tomas Vokoun** for allowing me to conduct research on this topic as well guiding me and supporting me throughout the process. He was constantly encouraging me and guiding me wherever I was making a mistake.

Security system using Raspberry Pi

Abstract

IoT is the infrastructure of connected physical devices which is surging at a rapid rate as an array of devices is getting associated with the Internet. Home security is a very significant application of IoT and it is being utilized to create home security systems. The system will inform the owner about any unauthorized entry by sending a notification to the user. Once the user receives the notification, necessary actions can be taken. The main objective of the thesis is to devise a smart security system utilizing microcontrollers like Raspberry Pi by integrating cameras and motion sensors into a web application. The microcontroller will control the motion sensors and video cameras and detect the presence of an intruder utilizing the Computer Vision Technique (CVT). If an unauthorized entry is detected, the microcontroller will alert the owner through an SMS or an alarm clock.

Keywords*:* raspberry pi, opencv, security system, motion detection

Bezpečnostní systém využívající Raspberry Pi

Abstrakt

IoT je infrastruktura propojených fyzických zařízení, která se rychle rozvíjí, jak se řada zařízení spojuje s internetem. Zabezpečení domácnosti je velmi významnou aplikací IoT a využívá se k vytváření systémů zabezpečení domácností. O neoprávněném vstupu bude systém majitele informovat zasláním upozornění uživateli. Jakmile uživatel obdrží oznámení, mohou být podniknuty potřebné akce. Hlavním cílem práce je navrhnout chytrý bezpečnostní systém využívající mikrokontroléry jako Raspberry Pi integrací kamer a pohybových senzorů do webové aplikace. Mikrokontrolér bude ovládat pohybové senzory a videokamery a detekovat přítomnost narušitele pomocí techniky počítačového vidění (CVT). Pokud je detekován neoprávněný vstup, mikrokontrolér upozorní majitele pomocí SMS nebo budíku.

Klíčová slova: raspberry pi, opencv, bezpečnostní systém, detekce pohybu

Table of Contents

1. Introduction 1
2. Objective and Methodology 2
2.1 Objective 2
2.2 Methodology 2
3. Literature Review 6
3.1 Scope of IoT in homes 7
3.1.1 How the Internet of Things (IoT) is shaping the Future of Home Automation 7
3.2 Security in Smart homes 8
3.3 Existing systems 12
4. IoT Implementation - Wired vs Wireless 17
4.1 Phase 1: Raspberry Pi and Peripheral devices setup 17
4.2 Phase 2: Implementing Face Detection using Camera Module 21
4.3 Phase 3: Amazon Web Services (AWS) setup 24
4.4 Phase 4: Implementing email and phone notifications 31
4.5 Phase 5: Data Analysis and Visualization 32
4.6 Testing 34
4.7 Using Thermal Sensor 38
5. Result and Discussion 43
5.1 Result 43
5.2 Challenges 43
5.3 Discussion 44
6. Future scope and limitations 45
7. Conclusion 46
8. References 47

List of Figures

Figure 1 Working flow chart of Security System (Own Source) 2
Figure 2 Raspberry Pi ports label diagram. Reprinted from Raspberry Pi projects (Jain, A., et Al., 2019) 18
Figure 3 Raspberry PI GPIO pin model (Reprinted from Raspberry PI Documentation) ... 19
Figure 4 Raspberry Pi and peripherals final setup image (Own Source) 21
Figure 5 Schema diagram of the database (Own Source) 28
Figure 6 Sequence diagram for theft alert to user (Own Source) 31
Figure 7 Eyes open status sample stored in database table (Own Source) 32
Figure 8 Sentiment dashboard for alerts and user's emotions (Own Source) 33
Figure 9 False face detection by OpenCV library example (Own Source) 34
Figure 10 Optimization of facial detection with FaceNet library layer (Own Source) 36
Figure 11 Email alert response example when theft is detected (Own Source) 37
Figure 12 Drowsiness alert email example based on AWS API response (Own Source) ... 38
Figure 13 Face detected using Haar Cascade (Own Source) 39
Figure 14 Edge map obtained from a thermal image (Own Source) 39
Figure 15 Some examples of face detection in different conditions using contour detection. (Own Source) 40
Figure 16 Some examples of face detection in different conditions using Template Matching (Own Source) 40
Figure 17 Some examples of face detection in different conditions using Chamfer Matching (Own Source) 41
Figure 18 Template used for template matching (own source) 41
Figure 19 Distance Transform image of template (a) and query image (b) (Wang, S.P., et al., 2017) 41

List of Logs (Programs)

Log 1 Node.js Subroutine to test the camera module. 19
Log 2 Node.js subroutine to read the motion sensor signal. 20
Log 3 Scripts to install OpenCV and its dependencies. 22
Log 4 Node.js subroutine to detect human face. 22
Log 5 FaceNet implementation to detect human face. 23
Log 6 Response from AWS Recognition on adding new face to collection. 25
Log 7 Node.js subroutine for searching face from existing face collection. 26
Log 8 Node.js script for detecting facial details. 27
Log 9 Node.js subroutine for phone message service implementation using SNS. 29
Log 10 Node.js subroutine for email notification implementation using SES. 30
Log 11 Error response from AWS face search API without face in requested image. 35
Log 12 AWS face search API response when unknown image passed. 37

1. Introduction

Internet of Things (IoT) is the network of devices that communicate and exchange data among themselves without the need for human intervention. It is formally defined as “Infrastructure of Information Society”. IoT enables us to extract information from all kinds of mediums. Thus, any object or device which can be provided with an IP address for data transmission over a network by embedding hardware such as sensors, network gears, and software can be made part of an IoT system. (Kodali, R.K., Jain, V., Bose, S. and Boppana, L., 2016)

The IoT infrastructure has aided in rendering real-time information analysis and gathering utilizing accurate sensors and seamless connectivity, which helps in making efficient decisions. With the advancements of IoT, both the consumers and manufacturers have benefited. The manufacturers have gained close insights into their products' utility and performance while the consumers on the other hand can integrate and control more than one device for an improved and customized user experience.

The IoT-based smart home system allows the user to control everything connected to Internet in their homes. The consumer can collect vital information related to atmospheric conditions of the home like humidity, temperature, etc. One of the most crucial parts associated with home automation is security. In the past few decades, home security has seen drastic changes. The conventional systems consisted of alarms that would go off when somebody would break in, however, with advancements in technology and the introduction of smart homes the importance of security has amplified. (Iaz, U., Ameer, U., ul Islam, B., Ijaz, A. and Aziz, W., 2017)

Arduino-based home security systems are quite popular in the IoT system. The system utilizes temperature, smoke, LPG, and IR sensors to regulate security around the home. Data from these sensors are sent to Arduino, which has an inbuilt signal converter. Further, the data is sent to the Wi-Fi network. For theft detection, the existing systems utilize IR sensors. These sensors are installed on the doors and require the intruder to input a password. Failure to input the correct password alerts the owner of potential theft. Temperature and smoke detectors are also installed to reduce the risk of fire hazards. (Iaz, U., Ameer, U., ul Islam, B., Ijaz, A. and Aziz, W., 2017

2. Objective and Methodology

2.1 Objective

The main objective of the thesis is to devise a smart security system utilizing microcontrollers like Raspberry Pi by integrating cameras and motion sensors into a web application. The microcontroller will control the motion sensors and video cameras and detect the presence of an intruder utilizing the Computer Vision Technique (CVT). If an unauthorized entry is detected, the microcontroller will alert the owner through an SMS or an alarm clock.

2.2 Methodology

The user will be able to control the security system through his phone and laptop having an internet facility using Raspberry Pi. A program will be written to get the live streaming of the cameras and fetch data from the database.

Raspbian operating system will be installed in Raspberry Pi. I will be utilizing Python programming language to communicate with General Purpose Input Output Ports and a simple connection will be made with the database. The database used will be AWS integration. Face detection and Identity identification will be carried out using an open CV module.

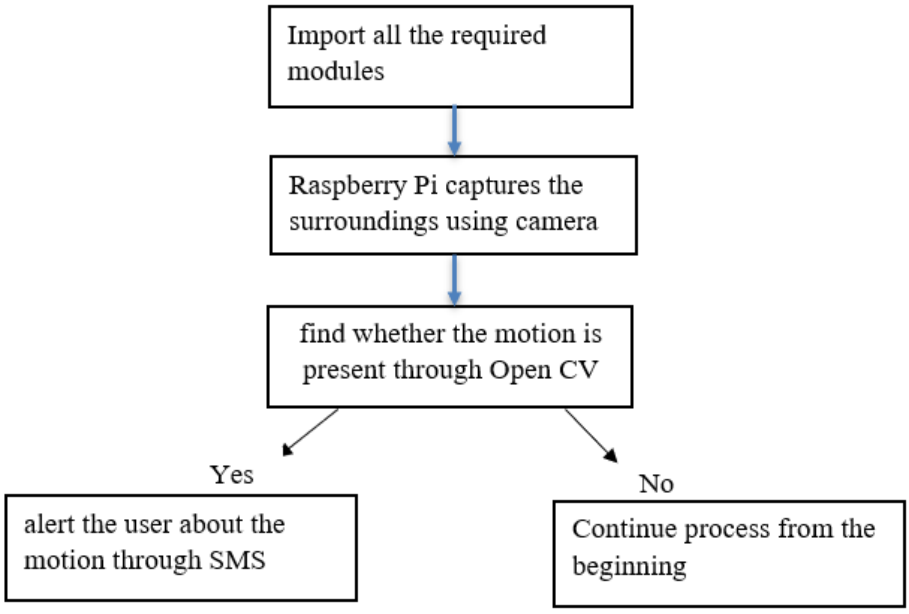


Figure 1 Working flow chart of Security System (Own Source)

The proposed system can be demonstrated with the help of the following steps.

- Import all the required modules i.e., Open CV, MYSQLDB
- Check whether the Raspberry Pi captures the surroundings using a camera.
- Using Open CV, find whether the motion is present.
- If there is a motion alert the user about the motion through SMS using GSM connected Raspberry Pi.
- If no motion is detected, continue the process from the beginning.

C++ was used for development, and the OpenCV library (a free piece of software for computer vision and machine learning) was used extensively throughout the process. These techniques and approaches will be employed on real-time systems, thus the tools and processes required to acquire thermal pictures are also discussed here. A FLIR LEPTON Long Wave Infrared (50) shutterless camera module with an 80x60 active pixel focal plane array is used to capture thermal images in real time. You won't find any radiometry in this kind of camera. Similarly, the camera has not been calibrated, making it impossible to collect pixel-level temperature readings. Changes in both the camera's infrared temperature and the ambient temperature affect the value that is sent to the outside world.

It also uses a Raspberry Pi 3 model B for networking and image processing. Raspberry Pi 3 uses Serial Peripheral Interface (SPI) communication to get output values. A software interface is also supported, with the CCI running over a Two-Wire Interface (TWI) comparable to Inter-Integrated Circuit (I2C). An image capture method created by Pure Engineering. To use OpenCV's 8-bits with one-channel picture matrix format, the resultant output values must first be received in a 14-bits data format. Figure 2 displays an example of a thermal picture on the left. In this image, darker spots represent colder temperatures. In the machine learning process known as Haar Cascades, a cascade function is taught using examples of both high- and low-quality pictures. The Viola-Jones algorithm is used in this method.

The OpenCV package includes the Haar Cascades algorithm, among others. Cascade training for this system, which demonstrates a machine learning approach for face identification, requires a large sample size of photos of the item to be recognized. The same investigation was conducted for this piece of work; however, the facial detection findings were deemed unreliable. The online dataset contains thermal pictures with faces that were

utilized for the cascade training. In the following, you will find the outcome of training a Haar Cascade.

To the best of my knowledge, computer vision researchers have not paid as much attention to facial feature identification in thermal pictures as they have in visible light images. Using various features of the OpenCV library, a few options emerge. The initial step in using a thermal picture in a process or algorithm involves segmenting and filtering the image using morphological operators to produce a binary image.

The thresholding binarization Otsus technique is used for segmentation, while morphological operators including expansion, contraction, sharpening, and smoothing are used for filtering. The following algorithms were created and put into use by me:

How to Detect a Face Using Its Contours: Acquiring and Filtering the Contours to Get the Longest Contour in a Binary Image.

Template Matching: Technique for finding areas of an image that match a template image.

Chamfer Matching: Technique to find the best alignment between two edge maps.

I. **Face Contours**: By using the Canny edge detector technique to the binary picture, an edges map can be constructed, from which the contours may be extracted. The largest of these two outlines is then selected by filtering. The highest point of the contour is located because the human body has certain features that are irrelevant for face identification, such as the neck and shoulders. This coincides with the peak of the head or the highest point of the face. Here we may begin our search for the two spots that represent the widest part of the face. These two points, together with the highest point of the face's contour, are used to identify the face.
II. **Template Matching**: Template matching is a method for locating images that most closely resemble a given template. Many distinct approaches may be used to execute the template-matching methodology. In this study, I use Normalized Cross-Correlation, a technique that is still useful in many contexts.

    Template matching for objects of varying sizes is performed by researchers using a Picture Pyramid. To boost the efficiency of template matching, this study employs the utilization of an image pyramid to recognize faces of varying sizes. As the camera's output picture is low-resolution, only a small subset of the pyramid image

is utilized. The picture is scaled down at each successive level. To improve template matching results, a high-quality template is required.

III. **Chamfer Matching**: The foundation of Chamfer Matching's distance-transform-based shape-matching method. This method generates a Distance Transform map by comparing an edges picture of the test image with the template. This pixel's value on this map represents its distance from the closest background pixel. The location of the edge orientation contours and the calculated matching cost through the distance transform maps are then included into a chamfer distance map. The face is identified by pinpointing the pixel with the smallest chamfer distance.

3. Literature Review

Internet of Things (IoT) is an emerging paradigm that allows communication across electronic devices and sensors through the internet to facilitate our lives. (Kumar, S., Tiwari, P. and Zymbler, M., 2019). IoT renders innovative solutions by utilizing smart devices to counter various challenges related to business, public/private industries across the world. (Varela-Aldás, J., Toasa, R.M. and Baldeon Egas, P.F., 2022.). IoT is progressively becoming a significant aspect of our life that puts together an extensive variety of smart systems, frameworks, and intelligent devices and sensors. It takes advantage of nanotechnology and quantum in terms of sensing, storage, and processing speed which were not conceivable beforehand.

With the surge in IoT devices and technology, great transformations have been observed in our daily routine life. One of the prominent developments of IoT is the concept of Smart Home Systems (SHS) and appliances that compromises internet-based devices, automation system for homes and reliable energy management system. (Zhou, J., Cao, Z., Dong, X. and Vasilakos, A.V., 2017). Another prominent achievement in the IoT segment is the Smart Health Sensing System (SHSS). It incorporates small intelligent equipment to support the health of human beings. These devices can be utilized to monitor and observe the variant health issues and fitness level of a being. Hence, it transformed the entire scenario of the medical domain by facilitating it with smart devices and high technology. (Sfar, A.R., Natalizio, E., Challal, Y. and Chtourou, Z., 2018)

Another important sector where IoT has proven effective is transportation. It has brought new advancements in this sector to make it more efficient, reliable, and comfortable. Intelligent sensors and drone devices are now controlling the traffic at different intersections in the city. (Behrendt, F., 2019) In addition, technically advanced vehicles are being launched in the market which could sense the upcoming heavy traffic and congestion of the map beforehand and suggest an alternative route.

Therefore, IoT sees a lot of scope in various aspects of life and technology. IoT has also shown its importance and potential in the economic and industrial growth of a developing region. The utilization of smart sensing devices in the specialized smart home commenced in the last two decades until the sensors were available commercially to be used for ubiquitous applications. The sensing devices have been utilized in everyday items and

formed a cognitive network for the individuals living in the house and rendered an independent life. Some of the applications include lighting, security camera, home appliances, and alarm systems. These smart devices are connected to the local server through a wireless medium for data analysis and collection.

Further literature review will comprehensively elaborate important findings and facts identified through a rigorous search of the literature on different domains where IoT technologies have been utilized to foster home security.

3.1 Scope of IoT in homes

The existing IoT devices come with an array of features that one could not have imagined a decade ago. New advancements in the areas such as robotics and artificial intelligence are expected to completely transform homes to make them smarter than ever before. Although the AI-based voice-enabled assistants have already been incorporated in homes, these platforms will extend their functions and serve as the brain of the entire home. A range of activities can be conducted from controlling all the smart gadgets within homes, monitoring the health and preferences of the residents, and tracking schedules. (Kumbhar, D.S., Taur, S.M. and Bhatambrekar, S.S., 2018)

3.1.1 How the Internet of Things (IoT) is shaping the Future of Home Automation

Home automation has seen the utilization of smart gadgets for some time now. However, with IoT enablement, their scope has extended tremendously to cover a range of functions.

Smart Homes

Home automation systems currently operate on voice commands, but they are expected to evolve into intelligent systems that would function even in the absence of human voice commands. Their smart technology could extend their functionality to a range of actions that offer unmatched convenience.

Home Security

While advanced home security systems of today have the capability of remote monitoring and safety alerts, with technological advancements, these systems will come with advanced capabilities such as fences with motion sensors that alert homeowners about unusual intrusions (as opposed to false alarms every time the pet jumps over the fence) and drones

that automatically activate, based on an alert, to follow the intruders and monitor their actions.

Network Infrastructure

In the smart home of the future, the increased number of intelligent devices will require improved network infrastructure. Gadgets need to be always connected so that they can work when needed, thereby increasing dependency on a reliable and stable Internet connection. Wi-Fi mesh or other self-configuring networks that allow multiple devices to connect to it and detect and resolve connectivity issues before it reaches the customer complaint stage will provide a more reliable smart home network in the future.

Personal Health & Fitness

For monitoring the health-related needs of every resident within the home, devices such as smartwatches, fitness trackers, smart scales, and wearables such as smart clothes can help to track fitness and wellness to improve quality of life by continuously monitoring health information and updating personal fitness benchmarks.

3.2 Security in Smart homes

Security is a vital factor in an IoT-based smart home to ensure the privacy of the personal information of the residents because it involves uploading data on the cloud. Regarding the analysis of security challenges and problems, another significant work (Hossain, M.M., Fotouhi, M. and Hasan, R., 2015) has been carried out to bridge the gap through a detailed study of security issues, attack surfaces, forensics, and threat models. A survey conducted by Hewlett-Packard in 2014 (Moh, M. and Raju, R., 2018) depicted that almost 80 percent of IoT-enabled devices in smart homes breach the privacy of personal information, and almost the same number of communications had security vulnerabilities and were not encrypted.

Few devices can be altered such as magnetic sensors, smart TV and the alarm system through different antennas and RF signals. The security issues associated with communication of the data can be countered by the IoT devices with proper authentication and access control of peers through which the nodes communicate. The surface attacks happen in two networks, local and public with six different communication levels: device coordinator, device-device,

coordinator-gateway, and device-controller for the local network and controller-IoT service provider and service-service for the public network.

(Santoso, F.K. and Vun, N.C., 2015) involved the employing of a strong security system by implementing an AllJoyn framework that utilized an asymmetric Elliptic Curve Cryptography to render authentication while the system operation. The system operated on a Wi-Fi network where a gateway was utilized as the central node of the system. Android-based mobile phones were utilized as end devices that provided a means for set up, access, and control the system. The authentication was carried out in two rounds to assure extra stability of the proposed system. During the first round, the user was required to load the identity and pre-shared key of the IoT connected device on the mobile device while in the second round, the identity was uploaded on the home gateway.

In the article (Santoso, F.K. and Vun, N.C., 2015), the issues related to security were addressed in terms of filtering false network traffic and avoiding unreliable home gateways. One way to circumvent the former problem is to utilize a cooperative authentication scheme to avoid insignificant or false data being distributed in the network (Jonnalagadda, A., Sujatha, B. and Krishna, V.V., 2015). The second problem can be addressed by utilizing the side-channel monitoring (SCM) technique (Li, X., Lu, R., Liang, X. and Shen, X., 2011, June) to handle the problem at the least cost. The technique is the optimized selection of subnet of neighbours for each node along with the patch of the router to determine its performance in message transmission in the forward direction. But the source has the reverse direction of message transmission as its primary communication channel. One of the networking challenges addressed by this group is in terms of the interference of the unreliable nodes during the data communication and filtering of the false data traffic in the community network.

Significant works have also been done on the network-level security of the IoT-enabled devices in smart homes to restrict the illegitimate intruding intro residing people's activities. The work was carried out to augment the device level along with the network-level security solutions to use the software-defined networking (SDN) technology to dynamically block devices with suspicious behavior. A proposal of three-part architecture was given via an open-source SDN platform to determine the efficacy to render security to the IoT-enabled devices. The solutions were based on SDN implementing dynamic security rules that were based on contexts depending on the time or occupancy of people in the house. The protocol

was developed on the assumption that the ISP has the visibility of the devices present in the house and its access was SDN enabled. Floodlight (vo.9) OpenFlow controller was utilized for operating the ISP network.

One of the most significant ideas proposed for the privacy and security of smart homes utilized blockchain technology by eliminating the Proof of Work (PoW) and concept of chains (Dorri, A., Kanhere, S.S., Jurdak, R. and Gauravaram, P., 2017). The said algorithm was executed by utilizing an online, high-resource device named 'miner', that was responsible for the communication taking place inside and outside the smart home. Although the concept of POW and bitcoins was eliminated as it went through several challenges like high-resource demand and long latency for the confirmation of the transaction, that would result in blockage and low scalability of the whole network due to broadcasting transactions.

The algorithm relied on the hierarchical structure and distributed trust to maintain the blockchain and specifying it to the IoT to ensure the security of the home. To evaluate the performance of the proposed algorithm, a further simulation was carried out utilizing three remote sensors and an IPv6 communication protocol. The simulation contained handled situations with encryption, hashing, and blockchain that were proposed in the algorithm. Although the overlay delay and processing conditions were not considered during the simulation, the transfer of the bytes between the devices, miner, and the cloud is three to seven times for the proposed algorithm compared to that of the results obtained from simulation.

Smart home security also deals with protecting the residents from cyber-attacks. This is quite beneficial from the network point of view as the resource constraint nature of few devices does not permit to render solutions for any security problems, thus elevating the vulnerability of the information of the users to be hacked or intruded on by the third party. Almost all the devices are operated using the GHz range that utilizes ZigBee, Wi-Fi, Bluetooth, and NFC as communication protocols. A few of the security issues addressed in this work are the resource and energy constraints, heterogeneous communication protocols resulting in unreliable communication and leading to decrease in the computing performance and storage capabilities, low reliability of delivery packets, limited availability of energy for communication between devices and tampering attacks respectively.

Various potential security threats in smart homes based on IoT in the five different layers in an ISO model showcased the maximum probability of threats and attack framework in the network layer. One of the most prominent attacks on the network layer is the Black Hole attack on RPL where the attack commenced with a node and progresses through all the nodes in that pathway, dropping the packets that are transmitted through the pathway. This disrupted the flow, thus leading to data trafficking in the network. The attack on the application was initiated by the XMP Plot command line exploit tool that enforces the IoT-based smart home device to remove the encryption on the transmitted data to make it easier for the attacker to modulate them during their transmission.

A few things that can be carried out to minimize the attacks on the different layers are to possess an intrusion detection system to identify unauthorized intrusions and anomalies, employing tamper-resistant devices for the physical layer, employing a device authentication to identify the authorized devices from the unauthorized ones, also have a security key management system to protect the sensing devices inside the smart homes employed with pre-installed keys.

The security attacks are also based upon property, location, access level, and strategy (Hossain, M.M., Fotouhi, M. and Hasan, R., 2015). The location attack is dependent upon the external or internal position of the attacker in terms of the IoT network. When the attacker is external, he procures unauthorized remote access of the network domain from any public network. The device property-based attack is on the low-end and high-end devices utilized to get access to the IoT network and its related devices. The low-end devices may include wearable devices utilized by the people residing inside the homes for monitoring purposes, whereas end-end devices including computers, laptops, and virtual machines. The access level attacks are dependent on active and passive attacks. Active attacks affect the functionality of the IoT system by disrupting the normal transition of data. While the passive attacks involve unauthorized access to the network domain system and perform illegitimate activities to obtain personal information of the residing people. Strategic attacks are known as the logical and physical attacks that involve the tampering and damage of the IoT devices and affect the device in the communication channel.

The vulnerability of IoT-based smart homes for security in terms of privacy was also studied (Apthorpe, N., Reisman, D. and Feamster, N., 2017) in relation to the network traffic created during the data transmission. One of the major concerns remains in revealing the personal

information of the residents through metadata and trafficking of encrypted data. Certain possibilities to deal with the trafficking problem are to separate the traffic into packet streams, labelling the streams by the definite type of device, and examining the traffic rates.

3.3 Existing systems

Smart home utilizing GSM technology for design and implementation of the security has been discussed in (Govinda, K., Prasad, S.K. and Susheel, S.R., 2014), that utilizes two methods for the implementation of IoT. One of them includes utilizing web cameras that detect motions through a camera and sounds an alarm in case of suspicious activity. This method of intrusion is quite effective, however, a little expensive due to the costs of cameras. These cameras are required to be of high quality and should have a wide range, enough to detect motion. If one decides to incorporate dome cameras, the cost will elevate.

Karri and Daniel, SMS based system using GSM has been proposed to utilize internet services to send messages or alert the owner instead of conventional SMS (Karri, V. and Lim, J.D., 2005).

A fingerprint-based authentication system has been implemented by Jayashri and Arvind (2013) to unlock doors. This system ensures high security by only allowing the entry of the authorized residents. Monitoring of all those people who have utilized the sensor to gain access to the entry is also conducted. The system reflects additional security features including gas leakage and fire accidents. However, fully relying on fingerprint sensors is not considered wise, as it is relatively easy to tamper with fingerprints. This is the reason it is advised to utilize a two-factor authentication system along with fingerprint scanners where an additional layer of security is ensured in the form of a passcode, PIN, voice recognition, etc. (Rani, R., Lavanya, S. and Poojitha, B., 2018.)

A robust security system has also been proposed by a few researchers, where a fault in one of the components in the system does not lead to the failure of the whole system. (Sowjanya, G. and Nagaraju, S., 2016). The idea is to utilize multiple devices that may or may not be compatible with each other but can be incorporated in a way that can replace the existing component of the system in case of a fault. The model can use overlap between various devices which would lead to preserving energy thus increasing the efficiency of the model. An example of the above system is a temperature sensor, Wi-Fi module, and a door sensor in place of a faulty camera. The authors are successful in demonstrating the given example.

However, systems like these are useful for people who require energy efficiency in mind and who require a high degree of robustness with their security systems.

In 2016, a system was proposed where LDR sensors and Laser rays were utilized to detect intrusion using their movement. (Chilipirea, C., Ursache, A., Popa, D.O. and Pop, F., 2016). The working of the system includes, a laser is reflected on an LDR sensor and as soon as the contact of laser to LDR sensor is disrupted, the alarm connected to the sensor goes off alerting the owners and the neighbours. This system is a solution for problems of covering the places that are out of range from the fixed cameras but poses the same difficulties which exist in systems consisting of GSM modules to send text messages, that is delivery of a message is dependent on network coverage. However, the nature of the rays is a straight beam and can be easily dodged by an intruder who knows the system, rendering the whole system pointless.

In another journal, the authors discussed a novel way to design an electronic lock utilizing IoT technology and Morse code. (Lee, C.T., Shen, T.C., Lee, W.D. and Weng, K.W., 2016). It has been claimed by the authors that this is an original idea that has not been implemented before and is the first of its kind "optical Morse code-based electronic locking system". This system utilizes LEDs" s (Light-emitting diodes) as an encrypting medium to send signals. LED in smartphones has been used, to make it more accessible to the public. There is a photosensitive resistor and a microcontroller on the receiver's side, which can decrypt the optical signal after receiving them from the LED. After decoding the signal, it can then upload the existing condition of the lock to a cloud from where the owner can monitor the system. This system has been experimented with by the authors in real-time and has proved to work under variant illumination environments along with all the functions. The authors have claimed the system to be user-friendly and convenient, as it can be used by anyone with a mobile phone. (Singandhupe, R.B. and Sethi, R.R., 2016).

Anitha, A., (2016) proposed a home automation system using artificial intelligence and proposed a model for cyber security systems (Anitha, A., Paul, G. and Kumari, S., 2016).

Rath, P.K., et Al., (2021) explains in their paper how this paper used Raspberry Pi as the network gateway. This paper uses MQTT (Message Queuing Telemetry Transport) protocol for sending and receiving the data. All the sensors used in this paper have been controlled by the web page implementing the Access Control List (ACL) for providing encryption

methods for the safe transaction of the data. This paper uses various sensors, wired and wireless, that relate to the Raspberry Pi.

A research work by (Hossain, M.M., Fotouhi, M. and Hasan, R., 2015) proposed a framework for movement discovery utilizing Raspberry-Pi and IoT approaches. To support a security framework, the movement location framework is proposed in this research. In another research work, Sunil K., et al., (2016) introduced a framework that is based on IoT for movement recognition to support insightful remote-control observations of a web server.

Likely, in another research, Patel, P.B., et al., (2016) proposed another framework that was based on android for remote server monitoring purposes. The framework deploys push notification messages on android gadgets while it detects an interruption inside the room. This system authorizes clients to make the respective observations using the framework remotely on a smartphone. Another research work by Yang, J.C., Lai, C.L., Sheu, H.T. and Chen, J.J., 2013 includes a proposal of a framework in view of the human location. The framework deals with the rule of face location and can distinguish individuals by face recognition.

Research work conducted by Hyoung, R.L. and Chi, H.L., 2016 elaborated an experiment and demonstrated a framework to identify visitors utilizing IoT through IR sensors. It assists in differentiating the human body. The research utilizes ultrasonic sensors of two types to identify the position of a visitor. In the said research a camera module was also deployed through a servo engine to determine the position of the visitor. Through a web server that is connected to a sensor, is used to facilitate the visitor's data for remote clients. Furthermore, Varisrikrishna Patchava introduced a collaborative framework on the top of Raspberry Pi modules connected to the computing vision strategy. The framework-controlled home appliances through IoT. Raspberry Pi is deployed to control movement sensors and camcorders for detection and inspection. (Jyothi, S.N. and Vardhan, K.V., 2016),

A research work initiated by Naga Jyothi included experiments with another framework for security observations utilizing the web of things. The proposed system interacts for identifying the movement location utilizing Python to determine the capacity utilization. In this research, the motion detection algorithm was performed using Pi-Camera and Raspberry-Pi2. The system also supported live video splitting and identified articles in motions and generated an alert message as soon as a movement was detected.

In another popular research, object interactions change discovery using IOT is demonstrated by (Sforzin, A., Conti, M., Marmol, F.G. and Bohli, J.M., 2016). The research demonstrated incredible achievability and sensing utilizing Raspberry-Pi during an open-source interruption recognition framework.

Menezes, V., Patchava, V. and Gupta, M.S.D., (2015) proposed another interesting framework for local observation of locations in association with business zones. The proposed system is based on Raspberry Pi and a computer vision framework to identify moving articles in the inspection territory. Based on the stimulus received from the system, turn the lights on to capture the pictures and streams using the MPJG streamer which is available to authorized users to view.

From the perspective of different research work, PIR sensors are generally used in surveillance systems (Bai, Y.W. and Ku, Y.T., 2008) and in automatic light switching systems (Rajgarhia, A., Stann, F. and Heidemann, J., 2003). It has been proved that PIR sensors have promising capabilities as low-cost camera enhancers in video surveillance systems. Besides, in the research work (Hashimoto, K., Morinaka, K., Yoshiike, N., Kawaguchi, C. and Matsueda, S., 1997) PIR sensors are used in conjunction with cameras to address privacy issues and are known to be very effective. In this research, PIR sensors are deployed in private rooms for capturing event pictures. This research work covers the tracking of a human through the correlation of information from the two systems deployed.

In another research work, different approaches were presented to trach humans utilizing PIR sensors. Due to the highly sensitive nature of the PIR sensors during changes in incident radiation the sensors were arranged in an array in concurrence with a chopper wheel. (Gopinathan, U., Brady, D.J. and Pitsianis, N.P., 2003) The paper wheel holds the same temperature as the background. As a result, the module utilized for testing generates an output subject to the body having variant temperatures compared to the background. This behaves like a thermal imager. In another similar research work (Hao, Q., Brady, D.J., Guenther, B.D., Burchett, J.B., Shankar, M. and Feller, S., 2006) has developed a pyroelectric motion tracking system based on coded apertures.

Pavithra, D. and Balakrishnan, R., (2015) have proposed IoT based monitoring and control system for home automation. This project aims at controlling home appliances via smartphones using Wi-Fi as communication protocol and raspberry pi as a server system.

The server will be interfaced with relay hardware circuits that control the appliances running at home. Communication with the server allows the user to select the appropriate device. It employs IR, PIR, and fire detection sensors that can detect light, the presence of human beings, and any fire accidents respectively. This sensor sends a signal to raspberry Pi. From the raspberry pi, using Wi-Fi configuration and IoT concept user can turn ON/OFF the light or fan or sent an alert message along with the image and video were taken in camera to the mobile phone, and an automatic phone call is made to the nearby fire station as IR, PIR, and fire detection sensor is triggered accordingly.

Vinay Sagar, K.N. and Kusuma, S.M., (2015) proposes a Home Automation system using Intel Galileo that employs the integration of cloud networking, wireless communication, to provide the user with remote control of various lights, fans, and appliances within their home and storing the data in the cloud. The Intel Galileo development board, with a built-in Wi-Fi card port to which the card is inserted, acts as a web server. The automation system can be accessed from the web browser of any local PC in the same LAN using server IP, or remotely from any PC or mobile handheld device connected to the internet with an appropriate web browser through server real IP. The model consists of different sensors like temperature, gas, motion, and LDR. The designed system not only monitors the sensor data but also actuates a process according to the requirement. It also stores the sensor parameters in the cloud (Gmail) promptly. This will help the user to analyze the condition of various parameters in the home anytime anywhere.

Pandey, A., (2016) presents a brief introduction of the Internet of Things (IoT) with its vision and motivation. It briefly describes its elements such as RFID and WSN and puts light towards the integration of cloud and IoT. The paper even highlights the technological challenges and future directions to make IoT a reality. The paper also focuses on wide-ranging applications of IoT.

As seen from different research articles, a common characteristic of security surveillance lies with movement detection and response generation as real-time as possible.

4. IoT Implementation - Wired vs Wireless

The setup and implementation of the system's many components are covered here. This section is broken down into 5 distinct stages that each address a separate topic essential to the development of the system: the selection of tools, the selection of a programming language, the implementation of the code itself, and the implementation of the system.

4.1 Phase 1: Raspberry Pi and Peripheral devices setup

The initial step in developing the system was selecting the necessary components to construct the server. I opted for Raspberry Pi and its accessories since it is an IOT-based solution that can easily be linked with a cloud-based infrastructure. I utilized it to link a variety of cameras and motion detectors. It also runs all the software I developed to make my system work. If scripting on the server were to be done, Node.js would be the language of choice. I ultimately settled on developing a Node.js server that could function in a Raspberry pi setting as a result. The most up-to-date version of Node.js (11.14.0) and all required libraries were utilized throughout development, and the Raspberry Pi 3B+ model was used, which is also the most recent product in the Raspberry Pi line. High-powered gadget with 4 USB 2.0 ports, full-sized HDMI, Camera Module connector for attaching Raspberry Pi camera, 5 Volt power input, and inbuilt wireless LAN and Bluetooth 4.2. The operating system, Raspbian, was stored on a Secure Digital (SD) flash memory card and loaded using New Out of the Box Software (NOOBS), a user-friendly OS installer that also includes Raspbian. Ports available on the Raspberry Pi to connect to other Internet of Thing's devices are shown in Figure 2.

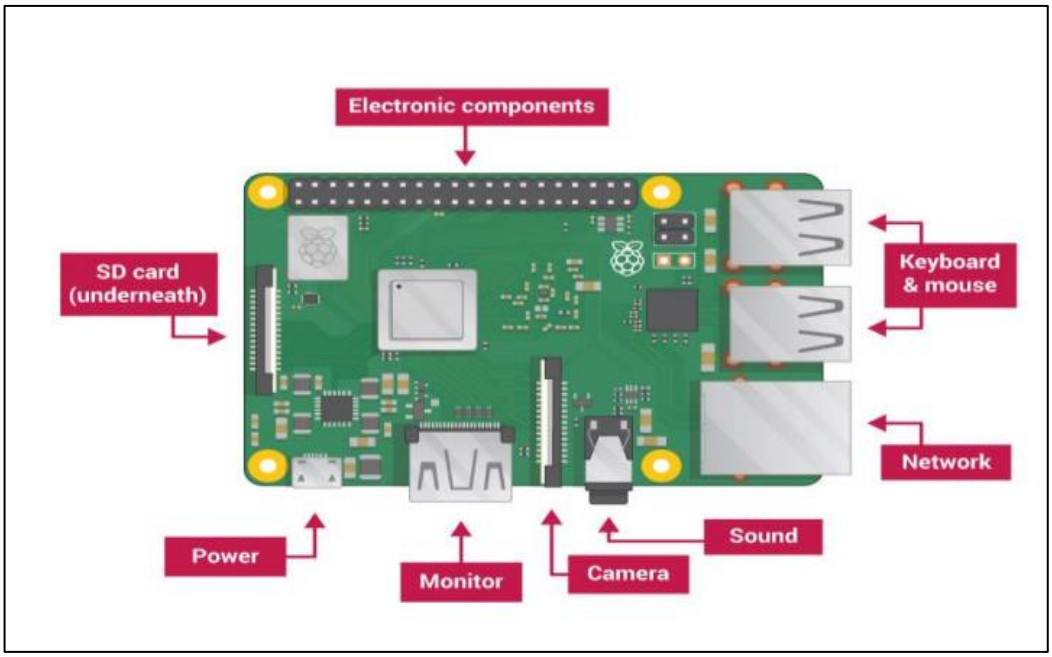


Figure 2 Raspberry Pi ports label diagram. Reprinted from Raspberry Pi projects (Jain, A., et Al., 2019)

Figure 2 shows the variety of ports available on the newest Raspberry Pi. The project did not make use of the sound port since it was not necessary. The Raspbian distribution is a customized version of the GNU/Linux system based on the Debian source code. It is the Raspberry Pi's official operating system. Raspbian comes with handy, all-purpose applications already installed. Pi Camera Module V2 was also installed and connected to the device through the camera port so that photographs could be taken. I checked the functionality of the camera after enabling the camera software in the Raspberry PI user interface. The camera components of the Raspberry Pi are used by four separate programs: raspistill, raspivideo, raspiyuv, and raspividyuv. Most of the time, the system will be using either the image encode component used by rasppstill or the video encode component used by raspivideo. Handling the Pi-Camera Node.js promised based library, a wrapper for the native raspberry Pi Camera Command Line Interface (CLI) utilities for using camera module components, I was able to test the camera signal. The most recent available version of the library, 1.2.1, was utilized for this endeavor.

The implementation of the camera test in Node.js is shown in Log 1.

```
const Picamera = require('pi-camera');
class Camera {
        constructor () {
                this.videoCamera = new Picamera ({
                mode: 'video',
                Output: ${__dirname}/video.h264, width: 1920,
                height: 1080, timeout: 3000,
                nopreview: true,
        });
        this.photoCamera = new Picamera ({
        mode: 'photo',
        output: `${__dirname}/photo.jpg`, width: 640,
        height: 480,
        nopreview: true,
        }
});
}
recordVideo () {
}
```

Log 1 Node.js Subroutine to test the camera module.

Log 1 demonstrates the development of a Camera class that implements two distinct strategies for collecting video and still images using the Raspberry Pi camera module. The procedures were carried out that put the media files in their designated directories on the device. To implement the PIR motion sensor, just three of the Raspberry Pi's GPIO pins were utilised. It's important to note that any of the pins may be designated as an output or input pin for various uses. On the board, you'll find two 5V pins, two 3.3V pins, and a few ground pins. The remaining pins are all 3.3V and may be used for whatever. It is shown in Figure 2 how to connect a PIR motion sensor to the Raspberry Pi through its GPIO pins.

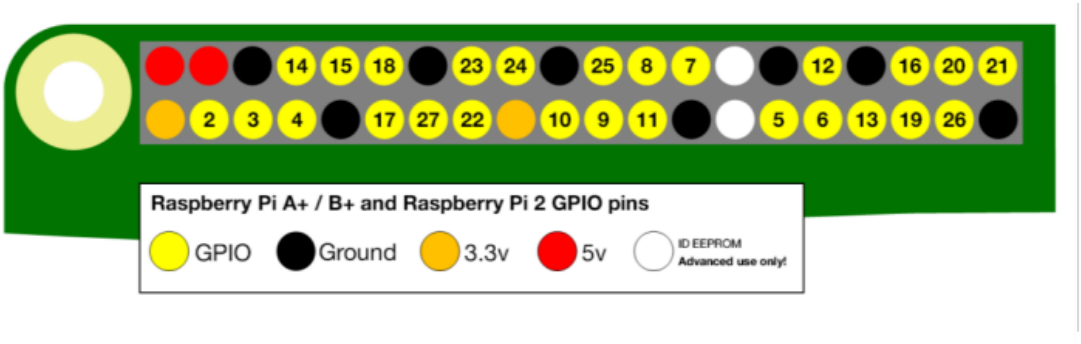


Figure 3 Raspberry PI GPIO pin model (Reprinted from Raspberry PI Documentation)

As shown in Figure 3, the two red marked pins are the input voltage, the two orange marked pins are the 3.3 Voltage outputs, the two black marked pins are the ground, and the other pins are the general-purpose pins. Many different languages and tools may be used to manipulate the GPIO pins. The project required the use of three GPIO pins for the connection of a PIR motion sensor. The PIR motion sensor's input, ground, and output pins were connected to GPIO 2, 6, and 12 correspondingly. A digital signal between 3 and 5 Volts (3-5V) is sent to the raspberry pi whenever motion is detected. The gadget accepts a 5 Voltage (5V) input signal. There was extensive testing of the setup, and the on off package for Node.js was used to read the signal. This module was developed so that Node.js may access GPIO and detect interruptions on Linux boards like the Raspberry Pi. The most current version of the library available while the system was being developed was utilized for the project, and that was version 4.1.1. The code for testing the PIR motion sensor is shown in Log 2.

```
const GPIO = require('onoff').Gpio;
const pir = new GPIO (12, 'in', 'both');

pir.watch ((err, value) => {
        if (value === 1) {
                console.log('Motion Active');
        } else {
        console.log('Motion Passive');
        }
});
```

Log 2 Node.js subroutine to read the motion sensor signal.

Using the on off library, the system was able to read the motion detection signal from the PIR motion sensor, as shown in log 2. By using 'both' as the third option in the GPIO function Object () {[native code]}, I told the system that I needed to setup for both falling and rising interrupt edges on GPIO pin 12. There were two possible binary values that resulted from the detecting process: 1 for active motion and 0 for inactive. It was expected that the equipment and its accessories would be sufficient for the task at hand. Before starting to develop the code, I made sure everything was set up correctly and tested it. Here, a picture of the Raspberry Pi hardware configuration (Figure 4) is shown.

Figure 4 Raspberry Pi and peripherals final setup image (Own Source)

Figure 4 demonstrates that the Raspberry Pi was successfully linked to a variety of add-on devices. When compared to the time and effort required for testing and verification, making the link was simple. For the sake of expediency and convenience, the real logic implementation and code writing were completed on laptop or personal computer. Git was used for both version control and shipping code to production, and a separate project repository was set up on Github. So, I used Github as a central repository to move the code from my local PC to the Raspberry Pi. When moving code from a local workstation to the Raspberry Pi environment for the first time, a script using the Node Package Manager (NPM) will be required to set up the necessary library dependencies.

4.2 Phase 2: Implementing Face Detection using Camera Module

The next step in developing a system is to put in place the necessary components to begin constructing features, after the completion of the device and peripheral configuration. This section demonstrates how to perform face detection from taken photographs before sending the data out to be analyzed by AWS Recognition. I utilized a Raspberry Pi camera module and the OpenCV and FaceNet packages for Node.js to do the detection. As can be seen in figure 4, I first installed all the prerequisite libraries needed by OpenCV, and then, after everything was done successfully, we installed OpenCV itself.

```
sudo apt-get install build-essential
sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev li-
bavformat-dev libswscale-dev
sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev
libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev
sudo apt-get install libopencv-dev
```

Log 3 Scripts to install OpenCV and its dependencies.

```
const cv= require('opencv');
const camera = new cv.VideoCapture (0);
camera.read((err, im) => {
        if (err) {
        throw (err);
}
if (im.size() [0] > 0 && im.size() [1] > 0) {
        im.detectObject (cv.FACE_CASCADE, {}, (error, faces) => {
        if (error) {
        throw (error);
        }
        /* return no data if no faces */
                if (faces.length === 0) {
                console.log('no data');
}
for (let i=0; i<faces.length; i++) {
const face = faces[i];
        if (Object.prototype.hasOwnProperty.call (face, 'x')) {
        im. rectangle ([face.x, face.y], [face.width, face.height]);
        /* save image to local machine if face available */
        im.save('../images/pic.jpg');
} else {
console.log('no data');
        }
        }
});
}
});
```

Log 4 Node.js subroutine to detect human face.

Installing OpenCV itself followed the installation of all its prerequisites, as seen in log 4. Since version 4.0 of OpenCV was the most recent available while the system was being built, it was utilized in the construction of the project. The picture was taken as often as feasible

and saved to a local disc. I utilized the library to analyze images locally on the PC and saved the discovered images in a separate folder before applying face recognition straight from the Pi Camera. To verify the reliability of the processed picture, a rectangular mark was placed on the subject's face before it was saved. Once everything was confirmed to work as intended, the implementation of the Camera Module was completed. Code to receive data from the Camera Module and analyze the picture for human faces is included in Log 4.

```
 const { Facenet | = require('facenet');
/**
*faceCheck
*@param (string) imagePath - path of image in local machine
**/
const faceCheck = async (imagePath) => {
        const facenet = new Facenet();

return new Promise (async (resolve, reject) => {
        try (
                const facelist = await facenet.align(imagePath);
                facenet.quit();
        if (facelist.length > 0) {
        resolve (imagePath) ;
        } else {
        resolve (‘no data’);
        }
        } catch (e) (
        reject (e);
        }
});
};
```

Log 5 FaceNet implementation to detect human face.

To capture a picture and then analyse the image to recognise a human face, I made use of the OpenCV library, as shown in log 4. If a face was spotted, a rectangular box appeared over it in the saved picture. The picture was saved in JPEG format, developed by the Joint Photographic Experts Group. Depending on picture quality, facial features, and other factors, the Node.js implementation of the OpenCV library was giving data with a 90% accuracy rate. After receiving the picture object from OpenCV's answer, I sent it into the TensorFlow-supported FaceNet module for further detection to enhance accuracy and optimise performance. Tensorflow's deep learning technique ensured a precise reply. When all was said and done, the picture was ready to be sent to AWS's recognition service, with a 99.6

percent accuracy rate compared to market is 98.1 percent accurate. FaceNet's Node.js implementation, which boosts its face recognition speed, is shown in Log 5.

The FaceNet Node, which was supported by TensorFlow, is shown in log 5. Using a java implementation library, I was able to identify faces in images saved to the local storage. The library then sends the picture to AWS's Recognition service after receiving the results of the deep learning algorithm's analysis of the face detection. Thus, after extensive testing, the face detection component was completed successfully. Improved performance and accuracy in face detection were achieved by using both the live camera approach and the local image method. If there was even one face missing, AWS Recognition would give an error back to the system, thus accuracy was critical.

4.3 Phase 3: Amazon Web Services (AWS) setup

Most project use cases have been implemented using AWS services. The cloud infrastructure was simple to set up because to AWS's plethora of available APIs (both CLI and SDK). The most difficult component of any installation was probably figuring out what was needed and deciding which service to use. Even though it was cheap, correct configuration was essential to minimize overhead and avoid needless API calls. The photograph was analyzed using Amazon Web Services' Recognition service so that I could determine the person's mood and level of fatigue. User interface configuration for this service is minimal. However, the initial step in using the service was to create a face collection containing photographs of recognized members. To prevent false alarms caused by unfamiliar people, a database of faces was developed that could be used to compare and determine whether a certain person was already recognized. The API would send back a response including the user's face id, the external image's id, and the image's id. When the system is initially capturing a picture, using these unique identifiers serves as a basis for comparison with subsequent images captured from the door. You can see an example of the answer you could get from AWS when adding a new face to your collection in Log 6.

```
{
        'SearchedFaceBoundingBox': {
        'width': 0.21505041420459747,
        'Height: 0.5088331699371338,
        'Left': 0.3681783080101013,
        'Top': 0.10956548154354095,
},
'SearchedFaceConfidence': 99.99998474121094,
'FaceMatches': [{
 'Similarity': 99.2770767211914,
 'Face': {
        'FaceId': 'unique-face-id',
        'BoundingBox': {
        'width': 0.1969980001449585,
        'Height': 0.2634269893169403,
        'Left': 0.45366498827934265,
        'Top': 0.1381170004606247,
},
'ImageId': 'unique-image-id
'ExternalImageId': 'unique-external-image-id',
'Confidence: 100,
},
}],
'FaceModelVersion': '4.0',
}
```


Log 6 Response from AWS Recognition on adding new face to collection.

Log 6 depicts the response payload, which includes crucial attributes such as the image's unique id, the face's id, and the external image's id. The similarity index in 'FaceMatches' is 99.27 percent, and it may be used as a metric for determining whether the picture really exists. After the presence of a human face has been confirmed locally using the OpenCV and TensorFlow libraries, the picture is sent to the AWS Rekognition service to get more face data. Properties such as "faceMatchThreshold," "Image," and "maxFaces" are included in the request body. The script to request a comparison of faces against a preexisting collection is shown in Log 7.

```
 const search Faces By Image = async (image) => {
const params = {
        CollectionId: 'collection id',
        FaceMatchThreshold: 90,
        Image: {
        Bytes: image,
        },
        MaxFaces: 5,
};
return new Promise((resolve, reject) => {
rekognition.searchFacesByImage(params, (err, data) => {
if (err) {
        reject (err.stack);
        }
resolve (data);
});
});
};
```

Log 7 Node.js subroutine for searching face from existing face collection.

Log 7 demonstrates how the parameter payload was built with the required features, such as the picture as base64 encoded bytes and the id of an existing face collection. As soon as the motion is recognized for the first time, the request is sent. This reduces the number of requests sent to the service that are unnecessary and beyond the parameters of the project. Using the AWS Recognition facial recognition API, I analyzed the persons moods in real-time. Every second a new picture was shown to gauge the person's inner state of mind. One of the system's main features is the ability to measure sleepiness, which was calculated by sending the received data to a dedicated function. A confidence percentage was used to provide a value to the characteristics of the answer payload. A higher percentage indicates more precision. The final product was saved in an S3 bucket with a unique identifier so that it could be readily located again in the future if necessary. Log 8 demonstrates how to use the AWS Face Detection API to read the mood of a person.

```
 const detect Face = async (imageBytes) => {
  const params = {
   Image: {
   Bytes: imageBytes,
},
Attributes: [
          'ALL',
          ],

return new Promise((resolve, reject) => {
rekognition.detect Faces (params, (err, data) => {
          if (err) {
          reject (err);
          }
resolve (data);
});
});
};
```

Log 8 Node.js script for detecting facial details.

Log 8 shows how to query the AWS Recognition service for face data by passing the base64 encoded byte string of the picture and an array of facial characteristics, perhaps "ALL" or "DEFAULT," as a properties of object argument. The RDS PostgreSQL database instance was used to store the data received from the API as the response payload. All face information and potential alert alerts are stored in a database organized by user image id. The attributes of responses from AWS API calls are stored in five distinct tables. The tables provided the information used to determine how tired a user was, how a user felt, and how a user felt about a potential date. Accordingly, the date each row was first created is recorded. In addition, the data is processed mathematically and integrated from other tables so that it may be properly shown in the visualization application. The database architecture for recording user feedback and warnings is shown in Figure 5.

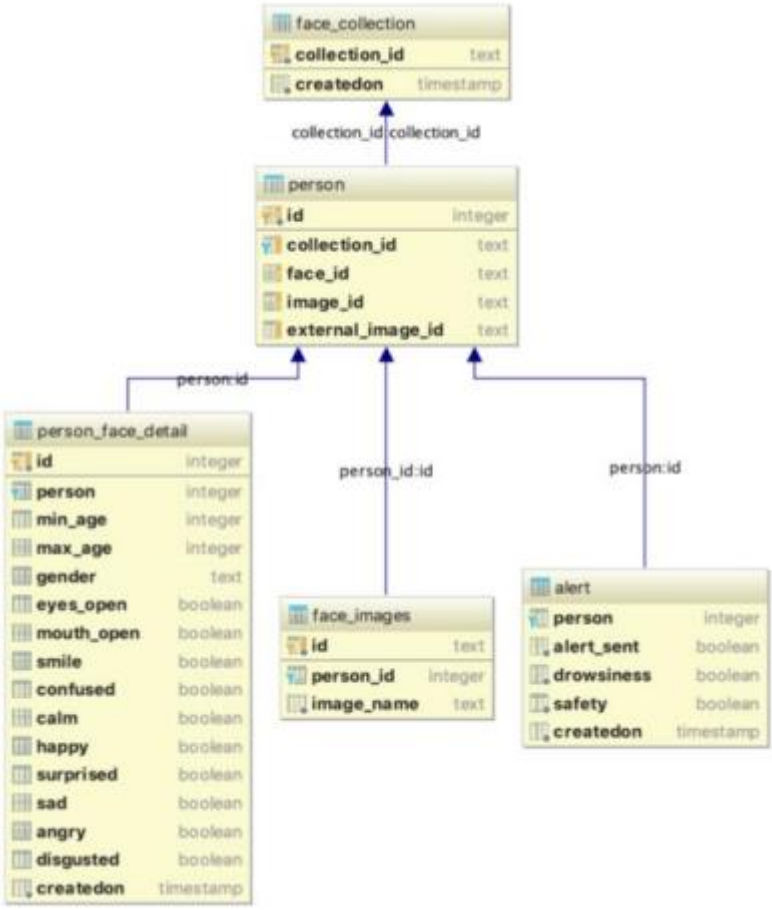


Figure 5 Schema diagram of the database (Own Source)

Figure 5 depicts the five tables that were built to hold the various API response data. Figure 5 and Figure 6 were made using lucid charts. There is a database of members' faces that may be accessed at any time. To keep tabs on the various picture IDs, face IDs, and external image IDs that come in through the AWS Recognition API, a "Person" database is created. The emotions and sentiments associated with each face captured by the AWS Face Detection API are stored in a separate database for each person's face. As pictures are uploaded to S3, a table may be used to keep tabs on them and associate each row with a unique identifier. For each person, the alert table keeps account of all the alerts that have been sent. The information utilized to generate the graphs and charts is illuminating. AWS SES and SNS were used to implement alerts or notifications by delivering emails and text messages, respectively. Member contact information including phone numbers and email addresses was saved. However, depending on the context, this data may be modified. The AWS SDK

was used to send emails and SMS messages. SNS and SES phone message alert implementations are shown in Log 9 and 10, respectively.

```
/**
        *sendAlertMessage
        *@param (string) message
        *@param {string} messageSubject
**/
const sendAlert Message = async (message, messageSubject) => {
const params = {
        Message: message,
        PhoneNumber: phoneNumber,
        Subject: messageSubject,
};
return new Promise((resolve, reject) => {
  sns.publish (params, (err, data) => {
     if (err) reject (err);
resolve (data);
});
});
```

Log 9 Node.js subroutine for phone message service implementation using SNS.

Logs 9 and 10 demonstrate how the AWS SDK was used to construct both phone and email alerts; the former uses the AWS SNS publish API to broadcast a message to a specified phone number, while the latter uses the AWS SES'sendEmail' API to notify the user through email. While the SES send email API needs a destination email address, message body, subject, and email source address, the SNS publish API request just requires the message content, topic, and phone number.

```
const sendSecurityAlertEmail = async (email, personName) => {
        const params = {
                Destination: {
                        ToAddresses: [
                        email,
                        ],
                },
Message: {
  Body: {
    Html: {
      Data: emailBodyTemplate,
      Charset: 'UTF-8',
     },
  },
Subject: {
  Data: 'email subject',
  Charset: 'UTF-8',
  },
};
Source: email,
};

try {
await new Promise((resolve, reject) => {
        ses.sendEmail (params, (err, data) => {
        if (err) {
                reject (err);
                } else {
                resolve (data);
                }
        });
        });
} catch (e)
console.error(e);
}
};
```

Log 10 Node.js subroutine for email notification implementation using SES.

4.4 Phase 4: Implementing email and phone notifications

By using AWS, setting up alerts by phone call and email took little time and effort. However, the relevant phone message and email content must be selected in the right context. The emails were written and tested on the correct use case. Theft warnings and sleepiness alerts each have their own unique information, and a jpg of the person's face is appended at the very bottom of the email. In the event of a theft alarm, the message would also include a link to add a user to the system if the person was already on the system's member list or was already recognized. Since the system now knows the user, it will not send out a theft notice if they try to use the same credentials again. The destination called upon by the connection is a service endpoint that provides access to both a user-adding database function and an AWS Recognition face collection. Here, I have a flowchart (Figure 6) of the steps involved in notifying the user of a theft.

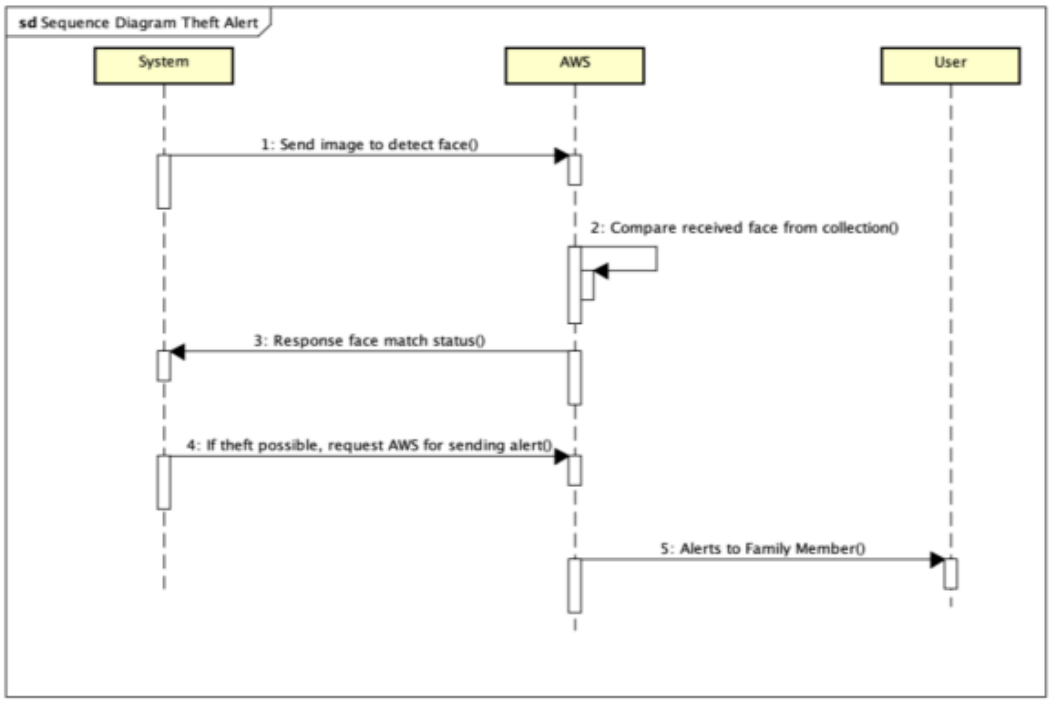


Figure 6 Sequence diagram for theft alert to user (Own Source)

As can be seen in Figure 6, the system next transmits the acquired picture to Amazon Web Services (AWS), where it is analyzed and compared to the preexisting face collection. The AWS answer included a face existence status. The system sends an API call to AWS, which then sends an email and/or SMS to the user depending on the obtained result. The system detects sleepiness based on the degree to which the eyes are closed. The picture will be sent

to the AWS Recognition API so that the eye close attributes may be checked. Amazon Web Services' face recognition API analyses images to identify people's emotions. After receiving the answer, the system analyses it to determine whether a drowsy state exists. The system will then send out alerts to the person and any current family members to head off potential disasters. Email and text message notifications are utilized, and repeated warnings are provided if necessary. Information obtained through the AWS detection API and connected to sleepiness is shown in a table, as shown in Figure 7.

	id	person	min_age	max_age	gender	eyes_open
1	24	37	19	36	Male	☑
2	26	37	26	43	Male	☐
3	27	37	26	43	Male	☐
4	28	37	26	43	Male	☑
5	29	37	26	43	Male	☑
6	34	37	26	43	Male	☑
7	35	37	26	43	Male	☑
8	36	37	26	43	Male	☐
9	37	47	35	52	Male	☐
10	38	47	26	43	Male	☐

Figure 7 Eyes open status sample stored in database table (Own Source)

Drowsiness data is recorded and assigned to a specific individual, as illustrated in figure 7, and may be used to both issue alerts and evaluate trends. If you look at the above table and see three "True" values in a row for the "eyes open" property, it implies the user was becoming sleepy at the wheel. Thus, warnings would be sent together with driving data to avert any potential danger.

4.5 Phase 5: Data Analysis and Visualization

The most important aspect in effectively communicating facts to viewers is the use of visuals. The project's dashboard, which serves as a starting point for inquiries in the event of a disaster, will offer a visual depiction of human feeling. To determine whether a certain person is fit to entre, statistical analyses also offer broad information about that person. The visual dashboard designed to see data related to emotions and alarms is shown in Figure 8.

Figure 8 Sentiment dashboard for alerts and user's emotions (Own Source)

The dashboard was developed to examine all the data collected by the face identification API, as seen in figure 8. The dashboard emphasizes prime metrics. The tables for the person's feelings and the notifications received are kept entirely separate. At the report's conclusion, a graph illustrating users' sentiments is shown. The data is live, and any changes to the filters in the header will cause the data in the dashboard's main body to refresh immediately.

4.6 Testing

The ability to recognize people and authenticate their identities based on digital media makes facial recognition systems very useful. The facts and findings are basically correct. Unfortunately, the technology isn't perfect; there are instances when it gives erroneous positive findings. The False Positive Rate is calculated by dividing the total number of compared faces by the number of incorrect results or false matches. The algorithm has also coped with outcomes like these when evaluated over many photos, as will be shown below. At home, I ran through the same scenario that I would use in a real-world evaluation of a car's functionality. Due to this, all coverages were tested at the test subject's residence. The algorithm was initially put through its paces with single-person photographs and then with group shots. On top of OpenCV, using the TensorFlow-supported FaceNet module helped reduce the number of false positives that had been seen. Figure 9 depicts False Positives caused by inaccurate facial detection.

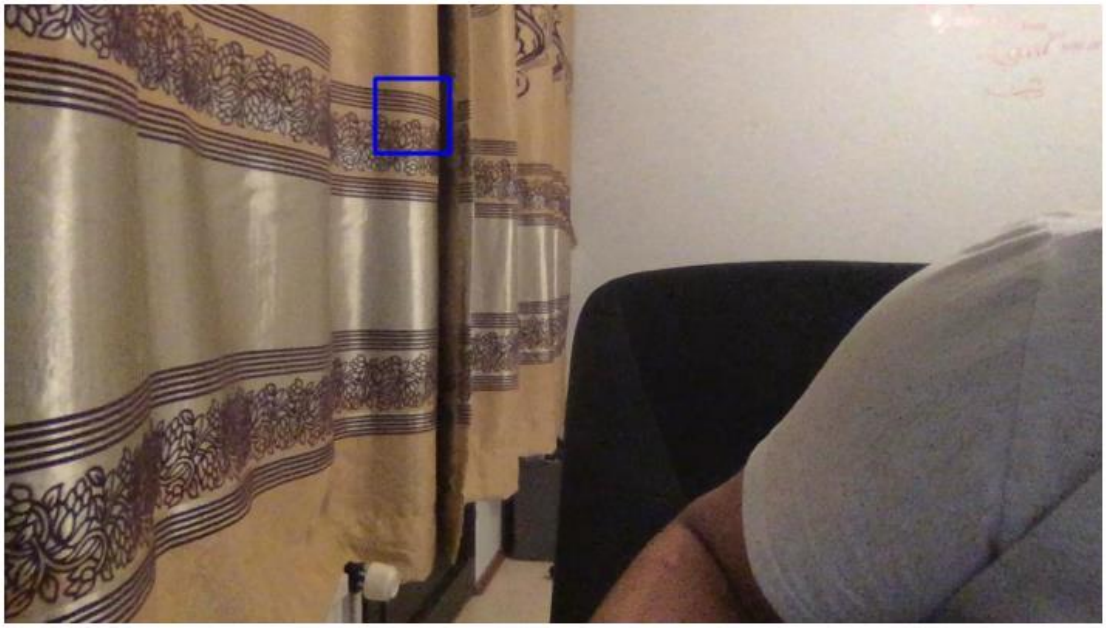


Figure 9 False face detection by OpenCV library example (Own Source)

The prevalence of false positives while utilizing face detection is seen in Figure 9. Still, it's a rare event at best. Several studies are being conducted to improve performance and reduce false positives so that accurate findings may be produced with more consistency. For this reason, a wide variety of algorithms are being utilized, and the quantity and variety of training data are both continually being expanded to account for variances. Also, the system's detection performance has been enhanced by the incorporation of a TensorFlow-supported

FaceNet layer. In this way, AWS receives more precise pictures for analysis, and errors that might otherwise halt the program's execution are avoided. For AWS to do a face comparison with an existing face collection, the picture must include at least one face. Log 11 shows the AWS error message that is returned when a picture is submitted without a face.

```
InvalidParameterException: There are no faces in the image. Should be at least
1.
at Request.extractError (/Users/macbooktester/nodejs/employee_satisfaction_in-
ternal/node_modules/aws-sdk/lib/protocol/json.js:51:27)
at Request.callListeners (/Users/macbooktester/nodejs/employee_satisfac-
tion_internal/node_modules/aws-sdk/lib/sequential_executor.js:106:20)
at Request.emit (/Users/macbooktester/nodejs/employee satisfaction_inter-
nal/node_modules/aws-sdk/lib/sequential_executor.js:78:10)
at Request.emit (/Users/macbooktester/nodejs/employee satisfaction_inter-
nal/node_modules/aws-sdk/lib/request.js:683:14)
at Request.transition (/Users/macbooktester/nodejs/employee_satisfaction_in-
ternal/node_modules/aws-sdk/lib/request.js:22:10)
at AcceptorStateMachine.runTo (/Users/macbooktester/nodejs/employee_satisfac-
tion_internal/node_modules/aws-sdk/lib/state_machine.js:14:12)
at /Users/macbooktester/nodejs/employee_satisfaction_internal/node_mod-
ules/aws-sdk/lib/state_machine.js:26:10
at Request.<anonymous> (/Users/macbook tester/nodejs/employee_satisfaction_in-
ternal/node_modules/aws-sdk/lib/request.js:38:9)
at Request.<anonymous> (/Users/macbooktester/nodejs/employee_satisfaction_in-
ternal/node_modules/aws-sdk/lib/request.js:685:12)
at Request.callListeners (/Users/macbooktester/nodejs/employee_satisfac-
tion_internal/node_modules/aws-sdk/lib/sequential_executor.js:116:18)
at Request.emit (/Users/macbooktester/nodejs/employee satisfaction_inter-
nal/node_modules/aws-sdk/lib/sequential_executor.js:78:10)
at Request.emit (/Users/macbooktester/nodejs/employee_satisfaction_inter-
nal/node_modules/aws-sdk/lib/request.js:683:14)
at Request.transition (/Users/macbooktester/nodejs/employee_satisfaction_in-
ternal/node_modules/aws-sdk/lib/request.js:22:10)
at AcceptorStateMachine.runTo (/Users/macbooktester/nodejs/employee_satisfac-
tion_internal/node_modules/aws-sdk/lib/state_machine.js:14:12)
at /Users/macbooktester/nodejs/employee satisfaction_internal/node_mod-
ules/aws-sdk/lib/state_machine.js:26:10
at Request.<anonymous> (/Users/macbooktester/nodejs/employee satisfaction_in-
ternal/node_modules/aws-sdk/lib/request.js:38:9)
Process finished with exit code 0
```

Log 11 Error response from AWS face search API without face in requested image.

Log 11 depicts an AWS error notice stating that the picture being processed must have a face to proceed. The'searchFacesByImage' API call was used to search for a picture inside a collection. To use the AWS Recognition API, a picture has to have a face detection rate of 100%. When using OpenCV to recognize an image from a camera, an extra layer of checking was included to protect against AWS API failures. Next, after receiving the result from OpenCV, I utilized FaceNet to verify the existence of faces in the picture. Errors from AWS were fully determined, preventing the system from failing, and the two combination layers helped deliver more accurate face detection results. Code implementation of an additional layer for improved face detection is shown in Figure 10.

```
const image = await new Promise((resolve, reject) => {
  camera.read((err, im) => {
    if (err) {
      reject(err);
    }
    if (im.size()[0] > 100 && im.size()[1] > 100) {
      im.detectObject(cv.FACE_CASCADE, {}, async (error, faces) => {
        if (error) {
          reject(error);
        }
        /* return no data if no faces */
        if (faces.length === 0) {
          resolve('no data');
        }

        for (let i = 0; i < faces.length; i++) {
          const face = faces[i];
          if (Object.prototype.hasOwnProperty.call(face, 'x')) {
            /* save image to local machine if face available */
            im.save(imageName);

            /* use facenet library to detect face in image again for double sure */
            const faceExists = await faceCheck(imageName);
            if (faceExists === 'no data') {
              /* delete image from memory */
              await new Promise((resolve, reject) => {
                fs.unlink(imageName, (err) => {
                  if (err) reject(err);
                  resolve('image deleted', imageName);
                });
              });

              resolve('no data');
            } else {
              resolve(im);
            }
          } else {
            resolve('no data');
          }
        }
      });
    } else {
      resolve('no data');
    }
  });
});
```

Figure 10 Optimization of facial detection with FaceNet library layer (Own Source)

Figure 10 demonstrates how the FaceNet library was used to determine whether a human face was present in a picture. Only if a human face is detected will the picture be saved and sent to the AWS API. If not, the image won't be saved to the local disc and any plans to run it will be scrapped. No faces were included in the face collection used for creating theft alert

before the image detection was performed. The system sent an email notice with a picture attached and a text message alert with a download link. In Log 12, I see the outcome of a query to the AWS Face Search API with an unknown face as the query.

```
{
        SearchedFaceBoundingBox: {
                Width: 0.21161779761314392,
                Height: 0.5229141712188721,
                Left: 0.4211474359035492,
                Top: 0.11873599141836166
        },
SearchedFaceConfidence: 99.99998474121094,
FaceMatches: [], // No Matching Face Found
FaceModelVersion: '4.0'
}
```

Log 12 AWS face search API response when unknown image passed.

Log 12 demonstrates that the AWS API response object has a key named "FaceMatches" that refers to an empty array. The system uses this value to send out alerts through email and text message. The email's content is coded in standard HTML and the user may attach JPEG files. Figure 11 depicts a sample email alert picture in the event of theft.

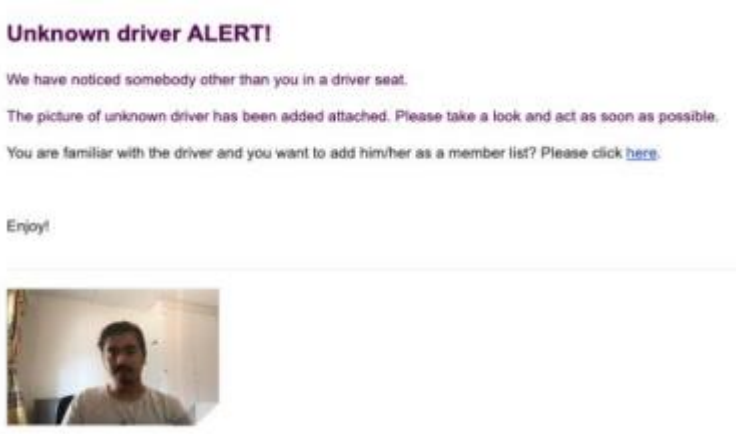
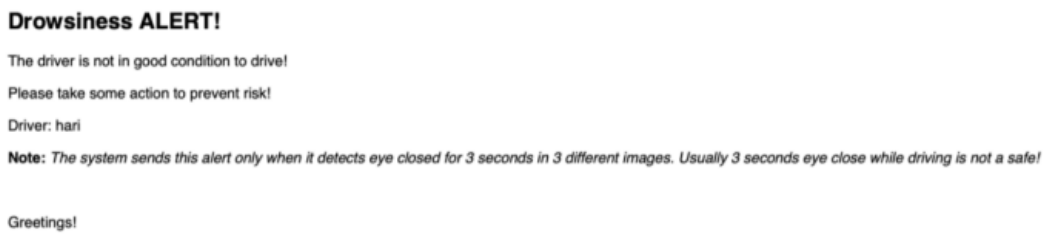


Figure 11 Email alert response example when theft is detected (Own Source)

The driving member is alerted through email as soon as the system detects a theft risk, as seen in figure 11. This guarantees the home is in good hands and allows the owner to simply take additional action based on the captured picture. In addition to the information shown on the dashboard, the database stores additional information about the person. Data from the

Amazon face detect API was also used to check for drowsiness. Within that time frame, the system sent three separate pictures of a face with closed eyes. If the API answer for eye closure status was true for three photos, an alert was sent. To maintain peak performance, this request was sent off repeatedly at the tick of a second. On my test scenarios, the result from AWS's facial recognition API was spot-on. Figure 12 is an example of sleepiness email notice.


Drowsiness ALERT!

The driver is not in good condition to drive!

Please take some action to prevent risk!

Driver: hari

Note: *The system sends this alert only when it detects eye closed for 3 seconds in 3 different images. Usually 3 seconds eye close while driving is not a safe!*

Greetings!

Figure 12 Drowsiness alert email example based on AWS API response (Own Source)

Figure 12 shows that as soon as the system identifies signs of drowsiness, an email message is delivered. Also included in the notice is the person's identity, which will allow the proper authorities to respond quickly.

4.7 Using Thermal Sensor

I provide experimental findings to validate the performance of the implemented algorithms. The samples were captured in real time using a camera coupled to a single board computer, in keeping with our focus on the immediate use of the contribution. To create an 8-bit grayscale picture from the output values that were used to construct the thermal images, special processing was required. Figure 13 is an example of this kind of pictures Haar Cascade.

Haar Cascades

The research used thermal images processed by the Viola and Jones method. According to the findings of this research, a sizable quantity of training data is required to achieve high accuracy. However, the detection time of this approach is similarly proportional to the size of the training data set, which is a major drawback.

Here, an effort was made to use cascade training for facial recognition. Figure 4 shows one possible implementation of this technique. Yet, its performance is subpar compared to that

of the cited research. Inconsistent results are shown when attempting to recognize a person's face when just their head and shoulders are visible in the picture.

Figure 13 Face detected using Haar Cascade (Own Source)

Proposed Methods for Face Detection

The segmentation acquired in these kinds of pictures and the use of morphological operators are shown. Figure 13 is an example input and output for the techniques and methods discussed.

A. Face Contours: The edge map in Figure 14 was generated using the Canny technique. See an example of contour-based face detection in Figure 15. If facial contouring suddenly stopped, this may change. Due to inconsistencies in the face's bounding box, this technique is not optimal for face detection.

Figure 14 Edge map obtained from a thermal image (Own Source)

B. **Template Matching:** If you want to use template matching, you'll need a suitable template. The utilized template is seen in Figure 18. By superimposing the template over the picture, this approach may determine how close the two images are in terms of similarity and, thus, the degree of mistake in the template. An approach using picture pyramids is also utilized to enhance the outcome of template matching by considering the size. The optimal solution is found as the worldwide maximum value. Examples of their use are shown in Figure 16.

C. **Chamfer Matching:** The Canny edge detector technique is used to the template from Figure 18 to turn it into an edge, which is then employed in the cost picture synthesis. It does this by combining the edge image with the template, which defines the distance between each query picture pixel and its closest edge pixel, to produce a distance transform map.

   The two distance transform maps are shown in Figure 19 below. After that, a map is made with the monetary value of each dot. Minimal cost occurs at the precise pixel spot where I want to focus our attention. Figure 17 displays some outcomes achieved by using Chamfer Matching.

D. **Processing Time:** The Raspberry Pi 3 model B used for all experiments allowed for near-real-time data collection.

Figure 15 Some examples of face detection in different conditions using contour detection. (Own Source)

Figure 16 Some examples of face detection in different conditions using Template Matching (Own Source)

Figure 17 Some examples of face detection in different conditions using Chamfer Matching (Own Source)

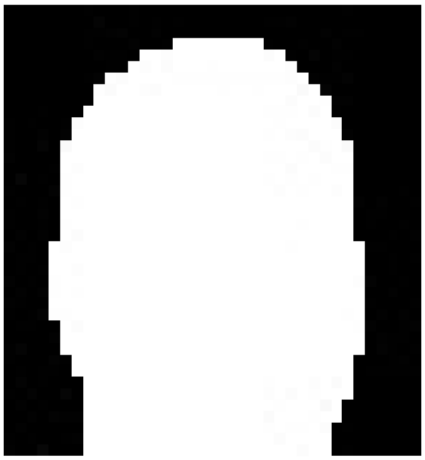


Figure 18 Template used for template matching (own source)

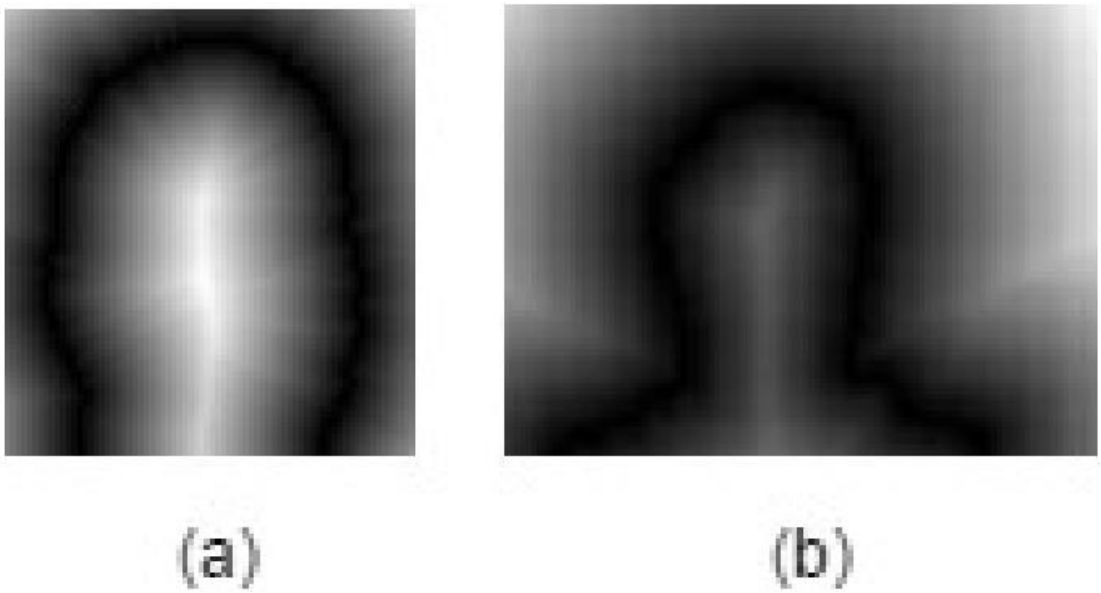


Figure 19 Distance Transform image of template (a) and query image (b) (Wang, S.P., et al., 2017)

About 72 milliseconds is the time required to process data using the Haar Cascade algorithm. An experiment lasting 70 seconds at a rate of 4 frames per second yielded the following results for the average processing time acquired using the offered methods:

i. **Face Contours -** 68ms.
ii. **Template Matching -** 69ms.
iii. **Chamfer Matching** - 257ms.

The Chamfer Matching technique takes the longest to process since it requires more processes.

5. Result and Discussion

5.1 Result

Integrating Deep Learning technologies with cloud services, a Raspberry Pi-based home safety and security system was developed. Potential and applicability of face recognition technologies were investigated in a range of settings. To show how important a face detection system in houses, I used cloud-based face recognition libraries and services. Accuracy of the facial recognition technologies which is comparatively better than ones on the market and API when dealing with photos and videos was evaluated during testing of the implementation. A Raspberry Pi-based home safety system using face recognition technology was the intended result of this work. Family safety and security were supposedly the system's top priorities, thus features like thermal sensing were made a part so that a person who might not be detected on camera could be sensed on thermal sensor. The software built on top of various face recognition libraries, like AWS Recognition, to analyse photos for face identification and sentiment analysis. However, flaws in the facial recognition libraries and machine learning stack utilised meant that the findings were not entirely reliable. Therefore, combining two libraries to reduce the errors proved useful for producing precise outcomes. This model is unique because we have adopted the most representative thermal information on the face utilizing microcontrollers like Raspberry Pi. A prototype of a Raspberry Pi-based safety system for cars of the future has been constructed.

5.2 Challenges

Several obstacles were met along the course of the senior year project. The most difficult part was choosing the right technology and technique since modern face recognition offers a vast array of algorithms for handling photos and videos. As a result of false positives discovered during OpenCV library construction, a second library, FaceNet, had to be used, lengthening implementation time, and necessitating further testing. Similar errors occurred when contacting the AWS recognising face search API with non-facial photos in the request payload, necessitating application logic to inspect the request payload.

Additionally, initial settings and preloaded libraries for programme execution varied across contexts for development and testing, namely the local environment and the Raspberry Pi environment. Because the project relied on cloud services that have a pay-as-you-go pricing

mechanism, I had to restrict the volume of requests until the work was done. Despite a few setbacks during development, the final product has been thoroughly tested and has all intended functionality and application logic.

5.3 Discussion

The manufactured gadget served as a concept's prototype and minimal viable product. Further advances in terms of device and system compatibility are needed before the gadget might be integrated into an home system as an in-built function. Prototype includes features including email and text message notifications. Adding an automated siren or speaker for notifications, as well as an immediate alarm, would increase the system's quality and reliability. Currently, historical data and statistics may only be seen on a separate dashboard. User satisfaction will be maximised by real-time dashboard views around the house. In addition, the information gleaned from identification and sentiment analysis may be used to programmatically customise the house’s audio system and playlists. Integration of cloud-based artificial intelligence technologies, such as text-to-speech or vocal bots, that do real-time sentiment analysis and provide recommendations, is possible. System performance may be enhanced with the addition of features such as automatic communication to local government and health institution in the event of exceptional incidences or casualties. To enhance the present notification system, face recognition technology may be further enhanced to identify not only the picture of a person, but also things like blood or crash. Finally, future implementation and development should consider non-functional needs such as data protection and security to prevent leaking of personal data, scalability of the device, and optimization for the response time.

6. Future scope and limitations

The senior capstone project's objective was to create a security system that makes use of cutting-edge tools including facial recognition, cloud storage, and Internet of Things. For the sake of the family and house, the system was designed to detect signs of drunk person and identify the user by name. With the help of the Camera Module, PIR motion sensors, and face recognition technologies from OpenCV and AWS Recognition, a Raspberry Pi-based home safety and security system was designed and implemented. The facial recognition libraries, Cloud API, and visualisation tool were used to develop a system that can detect drowsiness, identify faces, and analyse emotions. TensorFlow's FaceNet package, which was bound to Node.js, was utilised to decrease the number of false positives and improve the precision of the analysis.

Based on the collected data, the system can accurately identify faces, detect signs of fatigue, and provide warnings. To top it all off, the system was able to successfully gather information and display it graphically. It is anticipated that this technology will be a standard in-car feature soon. Integration of a real-time visualisation dashboard, automated alerts, and speakers, and more are all necessary for the system to be ready for use in vehicles. The senior capstone project's objective was to create a security system that makes use of cutting-edge tools including facial recognition, cloud storage, and Internet of Things. For future point of view’s sake of the person and the vehicle, the system will be designed to detect signs of sleepiness and identify the user by name. With the help of the Camera Module, PIR motion sensors, and face recognition technologies from OpenCV and AWS Recognition, a Raspberry Pi-based automotive safety and security system will update the designs and be implemented. The facial recognition libraries, Cloud API, and visualisation tool will be used to develop a system that can detect drowsiness, identify faces, and analyse emotions. TensorFlow's FaceNet package, which will bound to Node.js, will be utilised to decrease the number of false positives and improve the precision of the analysis.

Based on the collected data, the system can accurately identify faces, detect signs of fatigue, and provide warnings. To top it all off, the system was able to successfully gather information and display it graphically. It is expected that this technology will be a in-car feature soon.

7. Conclusion

In this work, an attempt is made to create a Raspberry Pi-based home security system. Raspberry Pi, camera, touch screen, and android phone are all required for the system to work. For software development, Python, AWS, and OpenCV libraries are commonly employed. As soon as a human is discovered, most intrusion detection systems will take a picture of the invader. The problem is that these systems don't have the ability to take a photo of the intruder so that they may be recognised (Face View Photo) and made visible. Illustrative methods are offered in the system to ensure that the intrusion detection is accurate, and the images obtained are clear enough to clearly identify intruders. Designed as a smart detector, the suggested device would deliver information as well as home protection to the owner through AWS SNS service. The technology is designed to work with touch and mobile devices. The owner's/mobile administrator's phone will get an alert message and a clear snapshot (Face view) of the intruder as soon as the trespass has been detected even through thermal sensor. The smart mirror's camera may record footage of the deployed area for the benefit of the owner.

Based on the Viola and Jones algorithm, Haar Cascade performs more efficiently and accurately. While effective, this algorithm needs a great deal of data and training time before it can provide satisfactory outcomes. One of the offered approaches for picture segmentation may be used to generate several face-only thermal images that can be used to train a Haar Cascade detector.

There is a high degree of accuracy in single-facial face identification when thermal images are segmented. Template Matching stands up as the best option among the suggested strategies. Face contours takes about as long to analyse as Template Matching, but the resulting bounding box is less reliable. Although Chamfer Matching's detection is comparable to that of Template Matching, its processing time is about four times slower.

The thermal camera utilised in this study is not calibrated, therefore the precise temperature in interest cannot be determined. I want to further refine our facial recognition algorithms and create calibration algorithms for this sensor so that I can monitor absolute temperatures in the future. I will also be developing a calibration method for using this and an RGB camera together.

8. References

1. Anitha, A., 2017, November. Garbage monitoring system using IoT. In *IOP Conference Series: Materials Science and Engineering* (Vol. 263, No. 4, p. 042027). IOP Publishing.
2. Anitha, A., Paul, G. and Kumari, S., 2016. A cyber defence using artificial intelligence. *International Journal of Pharmacy and Technology*, *8*(4), pp.25325-25357.
3. Apthorpe, N., Reisman, D. and Feamster, N., 2017. A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic. *arXiv preprint arXiv:1705.06805*.
4. Bai, Y.W. and Ku, Y.T., 2008. Automatic room light intensity detection and control using a microprocessor and light sensors. *IEEE Transactions on Consumer Electronics*, *54*(3), pp.1173-1176.
5. Bangali, J. and Shaligram, A., 2013. Design and Implementation of Security Systems for Smart Home based on GSM technology. *International Journal of Smart Home*, *7*(6), pp.201-208.
6. Behrendt, F., 2019. Cycling the smart and sustainable city: Analyzing EC policy documents on internet of things, mobility and transport, and smart cities. *Sustainability*, *11*(3), p.763.
7. Chilipirea, C., Ursache, A., Popa, D.O. and Pop, F., 2016, September. Energy efficiency and robustness for IoT: Building a smart home security system. In *2016 IEEE 12th international conference on intelligent computer communication and processing (ICCP)* (pp. 43-48). IEEE.
8. Dorri, A., Kanhere, S.S., Jurdak, R. and Gauravaram, P., 2017, March. Blockchain for IoT security and privacy: The case study of a smart home. In *2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)* (pp. 618-623). IEEE.
9. Gopinathan, U., Brady, D.J. and Pitsianis, N.P., 2003. Coded apertures for efficient pyroelectric motion tracking. *Optics express*, *11*(18), pp.2142-2152.
10. Govinda, K., Prasad, S.K. and Susheel, S.R., 2014. Intrusion detection system for smart home using laser rays. *International Journal for Scientific Research & Development*, *2*(3), pp.176-178.

11. Hao, Q., Brady, D.J., Guenther, B.D., Burchett, J.B., Shankar, M. and Feller, S., 2006. Human tracking with wireless distributed pyroelectric sensors. *IEEE Sensors Journal*, *6*(6), pp.1683-1696.
12. Hashimoto, K., Morinaka, K., Yoshiike, N., Kawaguchi, C. and Matsueda, S., 1997, June. People count system using multi-sensing application. In *Proceedings of International Solid State Sensors and Actuators Conference (Transducers' 97)* (Vol. 2, pp. 1291-1294). IEEE.
13. Hossain, M.M., Fotouhi, M. and Hasan, R., 2015, June. Towards an analysis of security issues, challenges, and open problems in the internet of things. In *2015 ieee world congress on services* (pp. 21-28). IEEE.
14. Hossain, M.M., Fotouhi, M. and Hasan, R., 2015, June. Towards an analysis of security issues, challenges, and open problems in the internet of things. In *2015 ieee world congress on services* (pp. 21-28). IEEE.
15. Hyoung, R.L. and Chi, H.L., 2016. Development of an IOT based visitor detection system. *IEEE Trans*.
16. Iaz, U., Ameer, U., ul Islam, B., Ijaz, A. and Aziz, W., 2017. IOT Based Home Security and Automation System. *NFC IEFR Journal of Engineering and Scientific Research*, *4*.
17. Iaz, U., Ameer, U., ul Islam, B., Ijaz, A. and Aziz, W., 2017. IOT Based Home Security and Automation System. *NFC IEFR Journal of Engineering and Scientific Research*, *4*.
18. Jonnalagadda, A., Sujatha, B. and Krishna, V.V., 2015. A Bandwidth-Efficient Cooperative Authentication Scheme for Filtering Injected False Data in Wireless Sensor Networks. *International Journal of Engineering and Management Research (IJEMR)*, *5*(4), pp.290-306.
19. Jyothi, S.N. and Vardhan, K.V., 2016, October. Design and implementation of real time security surveillance system using IoT. In *2016 international conference on communication and electronics systems (ICCES)* (pp. 1-5). IEEE.
20. Karri, V. and Lim, J.D., 2005, November. Method and Device to Communicate via SMS after a Security Intrusion. In *1st international conference on sensing technology, Palmerston North, New Zealand* (pp. 21-23).

21. Kodali, R.K., Jain, V., Bose, S. and Boppana, L., 2016, April. IoT based smart security and home automation system. In *2016 international conference on computing, communication and automation (ICCCA)* (pp. 1286-1289). IEEE.
22. Kumar, S., Tiwari, P. and Zymbler, M., 2019. Internet of Things is a revolutionary approach for future technology enhancement: a review. *Journal of Big data*, *6*(1), pp.1-21.
23. Kumbhar, D.S., Taur, S.M. and Bhatambrekar, S.S., 2018. IoT Based Home Security System Using Raspberry Pi-3. *International Journal of Innovative Research in Computer and Communication Engineering*, *6*(4), pp.1-8.
24. Lee, C.T., Shen, T.C., Lee, W.D. and Weng, K.W., 2016, November. A novel electronic lock using optical Morse code based on the Internet of Things. In *2016 International Conference on Advanced Materials for Science and Engineering (ICAMSE)* (pp. 585-588). IEEE.
25. Li, X., Lu, R., Liang, X. and Shen, X., 2011, June. Side channel monitoring: Packet drop attack detection in wireless ad hoc networks. In *2011 IEEE International Conference on Communications (ICC)* (pp. 1-5). IEEE.
26. Menezes, V., Patchava, V. and Gupta, M.S.D., 2015, October. Surveillance and monitoring system using Raspberry Pi and SimpleCV. In *2015 international conference on green computing and internet of things (ICGCIoT)* (pp. 1276-1278). IEEE.
27. Minoli, D., Sohraby, K. and Kouns, J., 2017, January. IoT security (IoTSec) considerations, requirements, and architectures. In *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)* (pp. 1006-1007). IEEE.
28. Moh, M. and Raju, R., 2018, July. Machine learning techniques for security of Internet of Things (IoT) and fog computing systems. In *2018 International Conference on High Performance Computing & Simulation (HPCS)* (pp. 709-715). IEEE.
29. Pandey, A., 2016. Paper on Internet of Things (IoT). In *National Conference on Technological Advancement and Automatization in Engineering* (Vol. 194).
30. Patel, P.B., Choksi, V.M., Jadhav, S. and Potdar, M.B., 2016. Smart motion detection system using raspberry pi. *International Journal of Applied Information Systems (IJAIS)*, *10*(5), pp.37-40.

31. Pavithra, D. and Balakrishnan, R., 2015, April. IoT based monitoring and control system for home automation. In *2015 global conference on communication technologies (GCCT)* (pp. 169-173). IEEE.
32. Rajgarhia, A., Stann, F. and Heidemann, J., 2003. Privacy-sensitive monitoring with a mix of IR sensors and cameras (Sensys demo). *USC/Inf. Sci. Inst., Tech. Rep. ISI-TR-582*.
33. Rani, R., Lavanya, S. and Poojitha, B., 2018. Iot based home security system using raspberry pi with email and voice alert. *International Journals of Advanced Research in Computer Science and Software Engineering ISSN*, pp.119-123.
34. Rath, P.K., Mahapatro, N., Sahoo, S. and Chinara, S., 2021, February. Design and Performance Analysis of an IoT Based Health Monitoring System for Hospital Management. In *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)* (pp. 721-726). IEEE.
35. Santoso, F.K. and Vun, N.C., 2015, June. Securing IoT for smart home system. In *2015 international symposium on consumer electronics (ISCE)* (pp. 1-2). IEEE.
36. Santoso, F.K. and Vun, N.C., 2015, June. Securing IoT for smart home system. In *2015 international symposium on consumer electronics (ISCE)* (pp. 1-2). IEEE.
37. Sfar, A.R., Natalizio, E., Challal, Y. and Chtourou, Z., 2018. A roadmap for security challenges in the Internet of Things. *Digital Communications and Networks*, *4*(2), pp.118-137.
38. Sforzin, A., Conti, M., Marmol, F.G. and Bohli, J.M., 2016. International IEEE conferences on Ubiquities Intelligence and Computing. In *IEEE*.
39. Singandhupe, R.B. and Sethi, R.R., 2016. Imperial Journal of Interdisciplinary Research.
40. Sowjanya, G. and Nagaraju, S., 2016, August. Design and implementation of door access control and security system based on IOT. In *2016 international conference on inventive computation technologies (ICICT)* (Vol. 2, pp. 1-4). IEEE.
41. Sunil, K., Ashutosh, K., Aditya, K.S., Naveen, B. and Anishka, K., 2016. WI-FI BASED HOME AUTOMATION SYSTEM USING CLOUD.
42. Varela-Aldás, J., Toasa, R.M. and Baldeon Egas, P.F., 2022. Support Vector Machine Binary Classifiers of Home Presence Using Active Power. *Designs*, *6*(6), p.108.

43. Vinay Sagar, K.N. and Kusuma, S.M., 2015. Home automation using internet of things. *International Research Journal of Engineering and Technology*, *2*(3), pp.1965-1970.
44. Yang, J.C., Lai, C.L., Sheu, H.T. and Chen, J.J., 2013. An intelligent automated door control system based on a smart camera. *Sensors*, *13*(5), pp.5923-5936.
45. Zhou, J., Cao, Z., Dong, X. and Vasilakos, A.V., 2017. Security and privacy for cloud-based IoT: Challenges. *IEEE Communications Magazine*, *55*(1), pp.26-33.