Nile Higher Institute
For Engineering and Technology

Department of
Communication and
Electronics

ARMY AKE System (Biometric Lock)

By

Abd El-Rahman Ahmed Hamed

Eslam Hesham Badawy

Ahmed Ashraf Elgareeb

Yousef Mohamed Gomaa

Kareem Alaa Elshazli

Mohamed Diaa Abd El-Sataar

Rafat Mohamed Abo-Ramadn

Under Supervision

Dr. Rehab Mahmoud

Eng. Manar Mohamed

Contents

Acknowledgment

We would like to thank all faculty members in Communications and Electronics department for their efforts in providing information and experiences throughout the years of study.
In particular, the project
supervisors: Dr. Rehab Mahmoud
who didn't keep any effort in encouraging us to do a great job, providing our group with valuable information and advices to be better each time. Thanks for the continuous support and kind communication which had a great effect regarding to feel interesting about what we are working on. We are really thankful to her. Also, we would like to thank our friends who helped us a lot in finalizing this project within the limited time frame. Finally, we express our deep appreciation to our families and friends for their kind endless support.

Project Team
2022

Abstract

ARMY AKE system is a biometric lock system that is made to apply high security systems with continuous technology revolution that we face nowadays and to find a lock method that saves life's against different appearing viruses like corona virus to substitute the traditional mechanical key which is easily exposed to lose, stole, damage, copied or easily virus transmission through people.

This system is an intelligent security system that depends on different artificial intelligence (AI) technologies such as face recognition, fingerprint, etc….. These technologies can work separately or can be connected with each other making a secure system with high efficiency. It can be used by users as a biometric lock for different applications like: car lock, treasure lock, hospital, research labs, national and different security building that needs high security authority to pass and etc…, It is secure, fast, reliable, immune to viruses, no need for mechanical keys and can be transferred independently to any device (for example: there is no need to buy the latest car to add this technology.

Chapter 1

Introduction

1. Introduction

The growing need for high security systems that is keeping pace with the development and emergence of internet of things (IOT) and mobile networks infrastructure that is a breakthrough in connecting anything and all things with each-other, also the appearing of viruses like corona virus. Leads to the need to develop a security system that can work well with the new technologies and also safe on people health so as not to get any viruses.

This leads to and initiate the need to Biometric artificial intelligent lockers (AI lockers) to substitute the traditional keys which is easily exposed to lose, stole, damage in any way or even easily copied.

In the Following sections an explanation of the ARMY AKE system is introduced starting by an introduction to the biometric and biometric methods.

1.1 Introduction to Biometric

It is hard to prove that you are who you say you are. You have a name, and so you can tell people your name. But someone else could impersonate you by using the same name. What do you do if you have to prove that you are the person that you say you are?

ofcourse, we must prove it all the time. We sign documents, we provide passwords to log in, and we present photo IDs. Sometimes we are required to provide our social security number and date of birth, as though only we would know that information. possession of a smartphone also acts as a personal identifier. Now, we can make purchases based on possession of our personal cell phone.

Identity fraud and identity theft are increasingly serious problems costing tens of billions of dollars per year in the US alone. All interactions with government, with financial institutions, and most interactions with businesses involve authentication as proof of identity.

An unacceptable solution is to install a chip into every human upon birth. In lieu of this distasteful solution, society is increasingly turning to technology and employing biometrics to authenticate a person.

Biometrics are unique physiological and behavioral attributes that can be used to identify individuals.

These characteristics are individualized, relatively fixed, and recordable. Typically, they are also hard to forge. In what follows, we discuss the emerging possibilities for automated biometric authentication.

Ultimately, the most unique and immutable property of each individual is their DNA sequence. (Of course, identical twins have the same DNA sequence, but there are other markers to distinguish them.) By identifying an appropriate number of specific markers that vary across the population, but uniquely identify a particular individual, it should be possible to biochemically authenticate a person. With advances in biotechnology, we foresee a time when signatures can be replaced with fast and efficient biochemical tests.

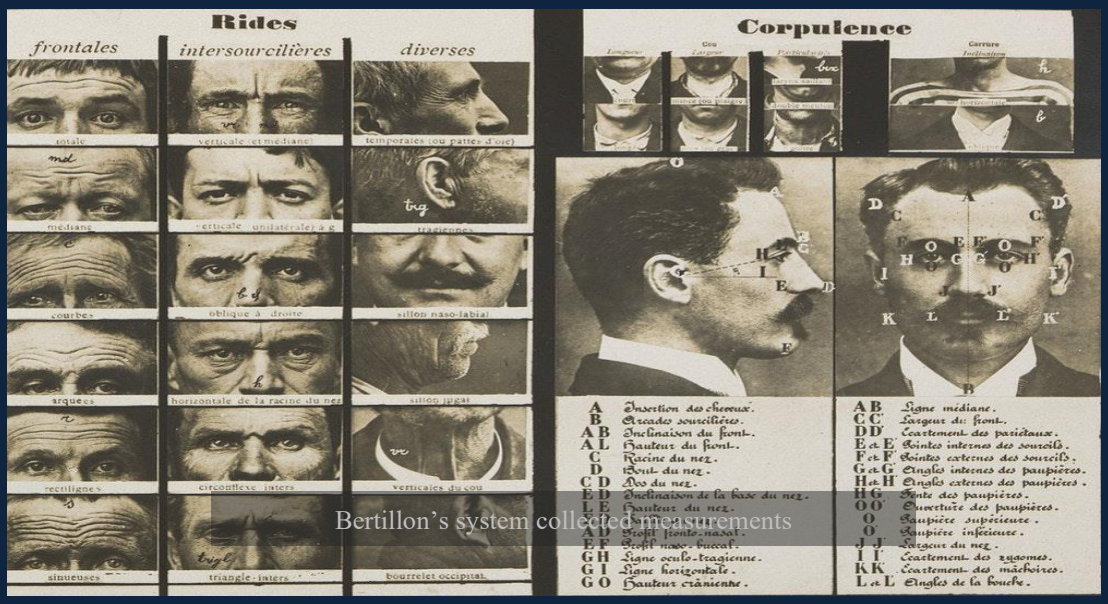
1.1.1 The Biometric and Databases

The concept of using biometric data for authentication is to demonstrate that a set of traits unique to a person match a prior recorded registration of those traits. The pre-stored database associates identity with the biometric data. The recorded data can be stored locally to the individual in an unalterable form or can be accessed remotely. Information technology allows us to access such a database quickly, and the fact that the individual claims an identity means that accessing the appropriate record is easy and does not require a search (although a search is also generally easy).

The stored database needs to be secure, or else an imposter can change the associations. Moreover, stored biometric data will often

be classified as personally identifiable information, and so must be kept protected. While compromised passwords can be changed, compromised biometric information is not easily changed. Public key encryption and homomorphic access technologies to compare sampled data with the encrypted stored data are possible solutions to maintain security. However, compromises from hacking are always possible, and privacy concerns exist across much of civil society with respect to biometric technologies.

1.1.2 History of Biometrics for Authentication



Bertillon's system collected measurements

Detailed ways of identifying people based on unique traits dates to the Bertillon System of 1879. Bertillon's system collected measurements to accompany a photograph of the subject and recorded the data on a filing card to track individuals in the criminal justice system. Bertillon's system captured five

main measurements—the head length, head breadth, the length of the middle finger, the length of the left foot, and the length of the cubit (the length of the forearm). While these measurements were not exactly unique and did lead to a few mistaken identity events, they represent an early approach to systematic identity management.

More modern types of biometrics range across multiple modalities and can be categorized as external physiology, behavioral, and internal physiological. Table 1 lists examples of biomarkers that can be used as biometrics, within categories. Most can be used to help authenticate a person, to provide entry, or to permit authorized actions. Some biometrics provide binary data: they either match or they don't. Other biometrics are analog, and match only if the value is close enough. When used for authentication and referencing encrypted stored values, the associated encryption method needs to preserve "nearness" for such analog measures. For search, filtering, and identification applications, machine learning approaches might be useful in training the recognition system. For authentication, however, machine learning will be useful for finding features in the data that might be best extracted to do the comparison for verification, and a separate decision procedure is needed to decide if the features match the pre-stored features for each authentication instance.

Category	Biometric	Description
External physiological	Voice	Identification based upon personal voice tracts
	Facial recognition	Identification by matching features from a map of facial features
	Fingerprints	Pattern recognition of the unique features of a fingerprint
	Palm prints	Similar to fingerprints, but not as widely used
	Hand geometry	Identification through the shape and dinensions of the hand
	Iris	Measuring the iris texture through visible and near-IR light to create a unique profile
	Ear shape	Similar to facial recognition but uses a detailed profile of the ear
	Scars	Considered a "soft" biometric identified along with marks and tattoos to identify an individual
	Eye veins	Using pattern-recognition on video images of the veins of the eyes
	Periocular	"Eye" recognition that includes the eyelids, eyelashes, eyebrows, tear duct, eye shape, and skin texture
	Odor	Characterizing and recognizing an individual based on their odor
	Footprint	Measuring the geometry, shape, and texture of footprints for identification
	Skin reflection	Using spectral reflectance of the skin for identification
Internal physiological	Sweat	Analyzing a profile based on amino acids and other compounds of each user from a sweat sample
	Blood and urine	Analysis of blood and/or urine samples. Blood samples can also be used to get DNA samples
	Microbiome	Using microbe data from stool, saliva, skin, and other collection sites, it has been shown that identification is possible and some samples remained stable for >80% of study participants after 1 year
	EEG/ECG	Taking EEG, ECG, or similar signals collected during a perception or mental task for identification purposes
	Tissue	Can include 2D ultrasound biometric systems. Direct tissue samples can also be used to get DNA samples
	Saliva	Saliva samples can be used to extract DNA samples
Behavioral	Typing habits	Based on monitoring how a user types, identity and mood can be identified
	Signature	Includes writing rhythm, acceleration, and habits
	Gestures	Generally captured from the face or hand; can classify and identify human motion
	Gait	Monitoring and modeling the way someone walks to identify them
	Touch screen tendencies	Integrating authentication into interaction based on personal tendencies on how a user touches a screen to ensure security
	Accelerometer data	A behavioral biometric identifier built around a user's movements
	How a device is held	Similar to touch screen tendencies and the accelerometer data, how a device is held can also build up a behavioral data set

Table 1. Examples of biomarkers that can be used as biometrics, within categories.

1.2 Types of Biometrics for Authentication

- Current and Future Directions

Certain biometrics for identity authentication have been around for ages and are mature technologies, while others are emerging and still under development. There is often little data about performance levels because the accuracy depends so heavily on the particular application, operating environment. The following is a survey of selected biometric modalities.

1.2.1 Face Recognition Systems

1.2.1.1 History of Face Recognition

Humans use face recognition as the primary method of identifying people that they meet. Automated face recognition using image processing traces its roots to 1964 and the work of Bledsoe et al., who proposed identification based on 21 measurements. Since then, face recognition has been a mainstay of computer vision research. Recognition often compares facial features based on the spacing of the eyes, the bridge of the nose, the contour of the lips, ears, and chin. Other approaches use features such as the residuals after a principal components decomposition of a cropped image of the face.4 More recent developments include the use

of machine learning for automated feature extraction
and recognition against a database of stored faces.

1.2.1.2 Development in face recognition's accuracy.

Due to years of research progress, face recognition works quite well when evaluated in laboratory settings. Typical benchmark reports show better than 99% accuracy, others with error rate of less than 1%, depending on the number of faces in the pallet of possibilities. Algorithm cists compete internationally: The Gaussian Face algorithm developed in 2014 at The Chinese University of Hong Kong achieved facial identification scores of better than 98%. In 2020, one facial recognition algorithm test
had an error rate of 0.08%.

1.2.2 Fingerprint

1.2.2.1 HISTORY OF Fingerprint

Fingerprint recognition is one of the oldest and most developed biometric recognition methods. Latent fingerprint identification has been used forensically since at least the 19th century. Today, automated fingerprint recognition for authentication is regularly used for access control. Historically, fingerprints were collected using ink impressions on cardboard cards. Now, fingerprints can be collected using optical approaches, and can even be obtained by contactless methods. Contact systems can use an image, or measure conduction from a capacitive surface. Certain

smartphones now use ultrasonic sensors to collect fingerprints. Contactless fingerprint technologies include commercially available contactless fingerprint scanning technologies, with at least four mobile (smartphone-based) apps and two stand-alone contactless devices on the market. Contactless devices generally require that the fingers are in close proximity to the reader.

1.2.2.2 Fingerprint identification system

the Fingerprint Identification System uses the Integrated Automated Fingerprint Identification System to match fingerprints against the database.
Fingerprint-based access control systems for computer access or physical portal control are available commercially. Many laptops now have built-in fingerprint readers to control login access and use of a password manager, although the software generally allows for a password-based backup in case the fingerprint identification falsely rejects the user. As another example, the airport screening company Clear uses biometrics to authenticate people, with fingerprints as one of the biometrics that can be used.

The technology for recognizing fingerprints can use a direct comparison of the stored image of the fingerprint against the scanned print, invariant to a certain amount of variation of position and angle. However, this approach can fail to align the features

accurately, and so the established method of recognizing fingerprints is by observing specific features (such as loops, whirls, and arches), categorized manually, or extracted automatically using image processing. Biometric identification research continues to include developing improvements to fingerprint recognition, especially for contactless technologies.

The accuracy of fingerprint identification is highly variable, and controversial. For authentication, most systems will establish a loose threshold, with the assumption that imposters will be rare. Crime-solving using fingerprints is well established, but often makes use of other evidence to help narrow the search and improve the apparent accuracy of the fingerprint identification.

1.2.3Voice Recognition (VR)

1.2.3.1 Meaning of the process

Voice recognition is the process of converting a voice into digital data. The technology first appeared about 50 years ago, but it has become really popular in recent years.

1.2.3.2 Meaning of VR

Voice or speaker recognition is the ability of a program to identify a person based on their unique voiceprint. It works by scanning the speech and establishing a match with the desired voice fingerprint. The development of AI opened up extensive opportunities for this subfield of computer science. It enables us to interact with machines without touching them. It is growing rapidly, and developers are finding more and more ways to apply it in various fields.

1.2.3.3 HISTORY OF SPEECH RECOGNITION

The first significant steps of this technology began at IBM's Bell Laboratory. In 1952, IBM introduced Audrey, the first documented speech recognizer. Audrey was a fully analogic system that understood single numbers with pauses in the between. Ten years later, IBM introduced Shoebox, capable of recognizing 16 English words and numbers from 0 to 9. In the early 1970s, there was a leap in the development of this technology. This was mostly

due to DARPA, the R&D agency of the U.S. Department of Defense. Five years of research gave birth to Harpy by Carnegie Mellon. A machine capable of understanding 1011 words. In addition, Harpy was significantly different from its predecessors. It could understand sentences. In the early 80s, the size of the speech recognition system's vocabulary increased to several thousand words. This was mainly achieved thanks to the Hidden Markov statistical model. Speech recognition switched from pattern-based digital signal processing to predicting words from unknown sounds using statistical models.

Moreover, machines became more accurate in recognizing words. The Speech Recognition Group at IBM introduced Tangora, an experimental transcription system, in the mid-80s. Tangora was capable of recognizing 20000 words. Starting from the 1990s, speech recognition products such as Dragon Dictate became available to consumers thanks to personal computers. In the last two decades, many tech giants have been engaged in this technology. Later in this article, you will get acquainted with their products.

Chapter 2

Project Definition

2.1 ARMY AKE Definition:

ARMY AKE is a biometric lock System; this system depends on allowing the users to lock and temporarily unlock their locks using their biometrics. This facility aims to strengthen the privacy and confidentiality using user's Biometrics Data.

It is based on using biometric data such as face recognition, finger print, voice recognition, etc…, whether using them separately or working together which helps in getting more levels of security, flexibility, high efficiently.

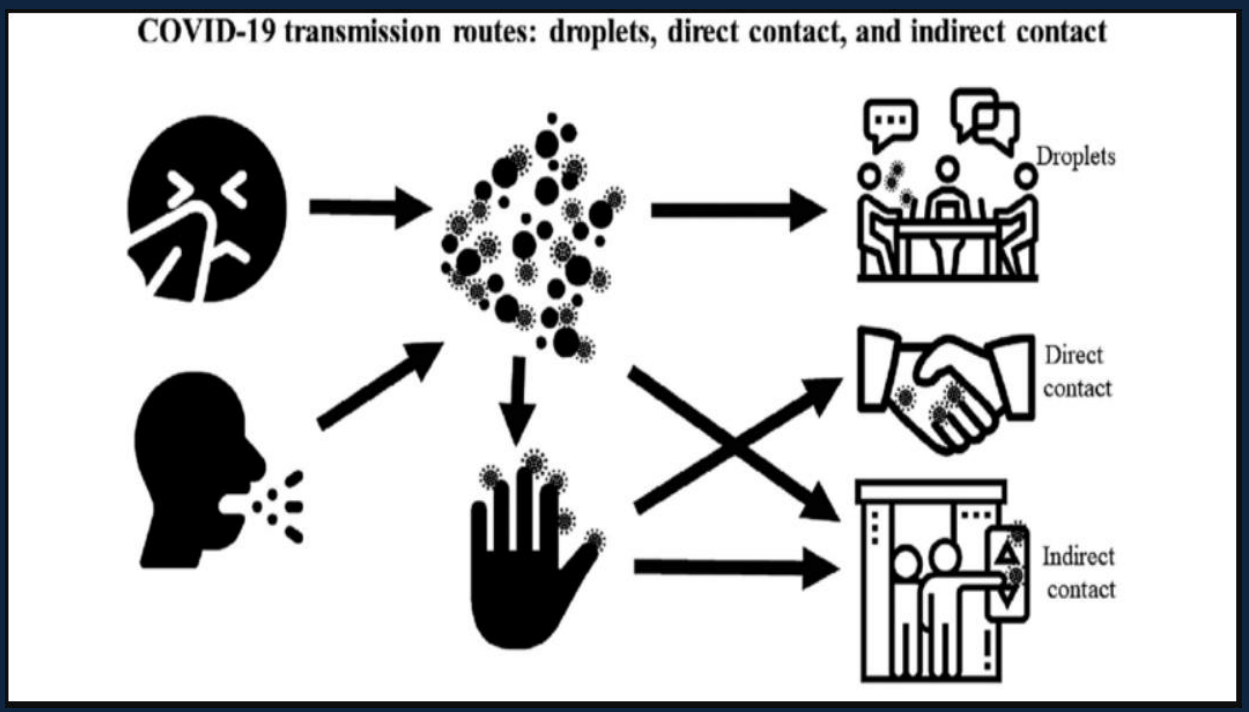
This technology is the future since it is fast, reliable and can be transferred independently to any device (there is no need to buy the latest car, new house or change your storage to add this technology), the component for these systems can be found easily at any electronics shops these components are Raspberry Pi - sensor camera – Fingerprint sensor in addition to the software that is very important for the work.

There are a lot of reasons that initiate our decision to start making Biometric Lock system, besides that these systems are more secure and flexible than the traditional mechanical keys, also traditional mechanical keys are easily exposed to lose, stole, damage in any way or even easily copied, which put you in problems that you don't need to be in. Losing a key of an important money safe, car or even the house's door or get stolen because your key copied or compromised, this will be your lose.

In addition to this reason the easiest and simplest of using the biometric lock systems and its security, making it good choice for be safe and comfortable.

One of the biometric lock feature that help you be more relaxed is that you can use it remotely by using your mobile and there are a variety of the technology you want to use it such as fingerprint, face recognition, voice recognition and etc.

The reasons that make the biometric lock is available are the developing of the technology, the continuous enhancement in the communication-infrastructure and also the developing in networks. The benefits of using a lot of technology are getting more security, flexibility, get away from the transmission of the diseases and viruses like corona virus.



The Transmission of corona virus

So If you have something worth protecting why not give it the star treatment? Biometric physical access control solutions are stronger authentication methods than keys, key cards and

PINs for a simple reason: they’re what you are, not what you have. While a key can be lost or stolen and used by an unauthorized person, your fingerprint is something unique that only you have. Fingerprint or Face processing biometric locks are perfect for keeping doors closed to all but those authorized to use them.

Face recognition process

Fingerprint process

2.2 problem statement:

- Some of the used technology like 5G is not available in all places.
- We need to buy some hardware components from abroad which was a problem with corona virus restricts.

2.3 Objectives

1. Fast and accurate identification

For only a matter of seconds, a biometric lock unlocks. You only have to let the device scan the biological measurements required. Also, the passcodes cannot be forged or engineered. The device requires the exact patterns and features before granting access.

2. Convenience to use

A biometric lock provides convenience when your hands are full of grocery bags. Instead of entering the codes or digging your keys, you only need to place your finger on the scanner or let the device scan your eyes, voice, or face.

3. Be Aware of Who's Coming and Going

With a biometric fingerprint door lock you can allocate Pins and passwords to different family members. This means you'll automatically know who has entered your home by the pin they entered. This can be a great relief for parents

who need to know their children or other family members have arrived home safely.

4. Customize to Your Needs

Using the many built-in features of your biometric fingerprint door lock you can customize the device to suit your own needs. Do you prefer to have a maximum of 3 entries? before the system locks out? That's easy!

Do you want each of your family members to have their own unique pin code? That's easy too!

Would you like to regularly change all codes every 3-months? No problem at all!

With all the above benefits it means you can keep using your biometric fingerprint door lock for years and years without having to replace it.

4. Restrict infection by viruses or any other diseases
5. Extremely User Friendly especially for disables.

2.4 Existing solutions

In this section a study will be presented for the Biometric Locks which are commonly used in the worldwide market.

1- Ardwolf A60 Biometric Door Lock System

Biometric Fingerprint Device Fingerprint scanner locks work by scanning the fingerprints of the people. The lock captures the image of the fingerprint, and it runs through the software to look for a match. If it has
found one, it opens the lock with fingerprint.

2- Biometric Device Iris Recognition Reader

Biometric Device Iris Recognition Reader Network Door Access Control System Night Vision Touch Screen Time & Attendance. Iris Recognition Device Another type of biometric lock is the iris recognition device. With the different colors of the eyes and their unique patterns, they help in identifying individuals. Aside from that, it provides contactless authentication.

3- TUXEDO TOUCH™ BY HONEYWELL

Voice Recognition System with unique tone, pitch, and frequency, the voice recognition system is now

incorporated into the biometric lock mechanism. After enrollment, the voice is converted into a digitized sample. Through voice, you can lock and unlock the device.

4- ZK Iface 302

Biometric Facial Recognition

A biometric facial recognition works by analyzing the nodal points of the face, such as the nose's width, cheekbone's shape, jawline's length, and eye sockets' depth. Whenever an individual faces the camera, the device looks for the matches saved in the database.

2.5 Development & Economic

Biometric Access Control is an industry developing daily with new and advanced technologies such as DNA matching, and most recently, advanced Biometric Facial recognition cameras, however in the end, the ultimate goal remains the same to allow access for authorized personnel in a convenient

and hassle-free manner without jeopardizing the safety of the business and its employees. 2018 has been a flagship year with numerous industries and sectors taking on the use of Biometric Access Control end masse, a few examples include Airports, Automotive industry and Gyms and Municipalities to name a few. Biometric Access Control Systems have many advantages and benefits to modern day society making it one of the most popular forms of security.

The below Graph depicts the expected exponential growth of Biometric Access Control Between 2015 - 2024

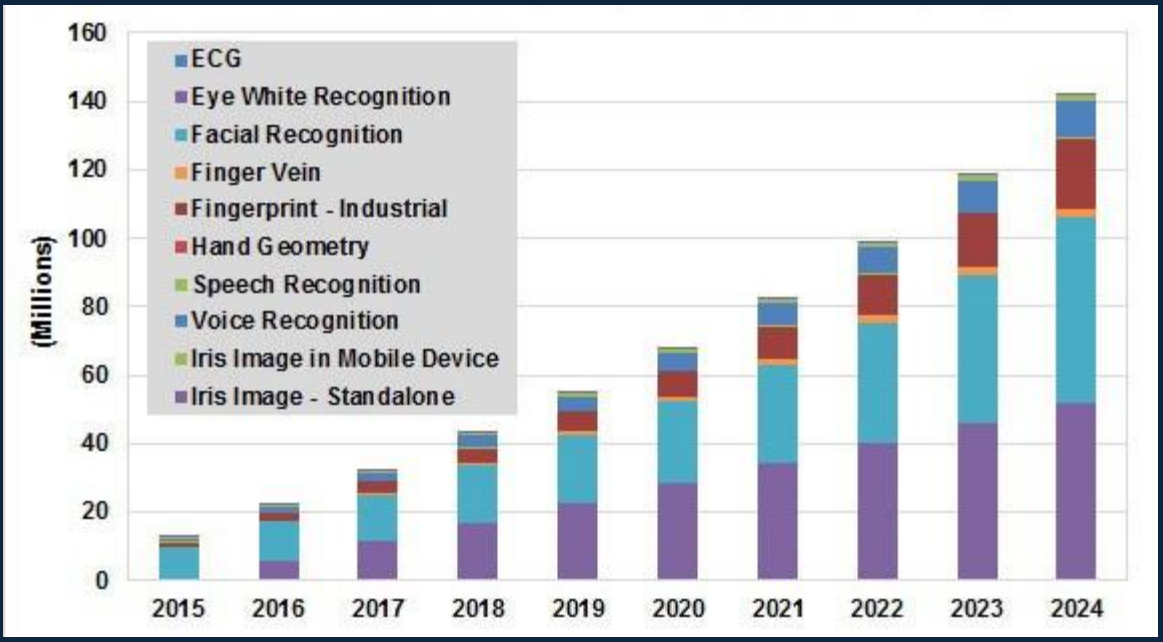


Image: Enterprise Biometrics Devices and Licenses by Modality, World Markets 2015 – 2024

Chapter 3

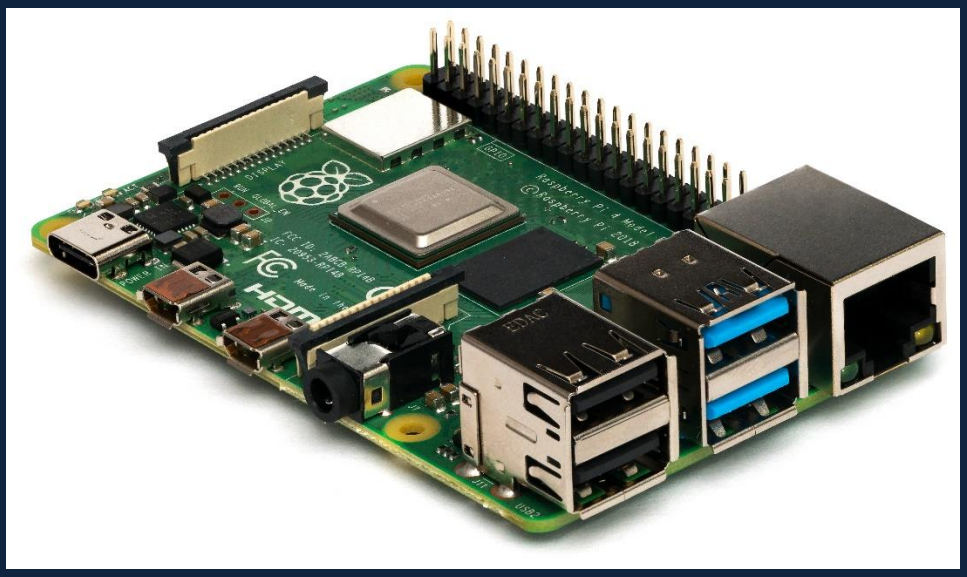
Components

Components

This chapter explains in details the project parts, hardware and software.

3.1 Hardware purposed component

1- Raspberry Pi



The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

What's more, the Raspberry Pi has the ability to interact with the outside world, and has been used in a wide array of digital maker projects, from music machines and parent

detectors to weather stations and tweeting birdhouses with infra-red cameras. We want to see the Raspberry Pi being used by kids all over the world to learn to program and understand how computers work.

2- Fingerprint sensor

The fingerprint sensor connects with raspberry pi this sensor is one kind of sensor which is used in a fingerprint detection device. These devices are mainly inbuilt in the fingerprint detection module and it is used for computer safety. The main features of this device mainly include accuracy, better performance, robustness based on exclusive fingerprint biometric technology. Both fingerprint scanner otherwise reader is an extremely safe & suitable device for safety instead of a secret word. Because the password is easy to scan and also it is hard to keep in mind.

3- Camera module

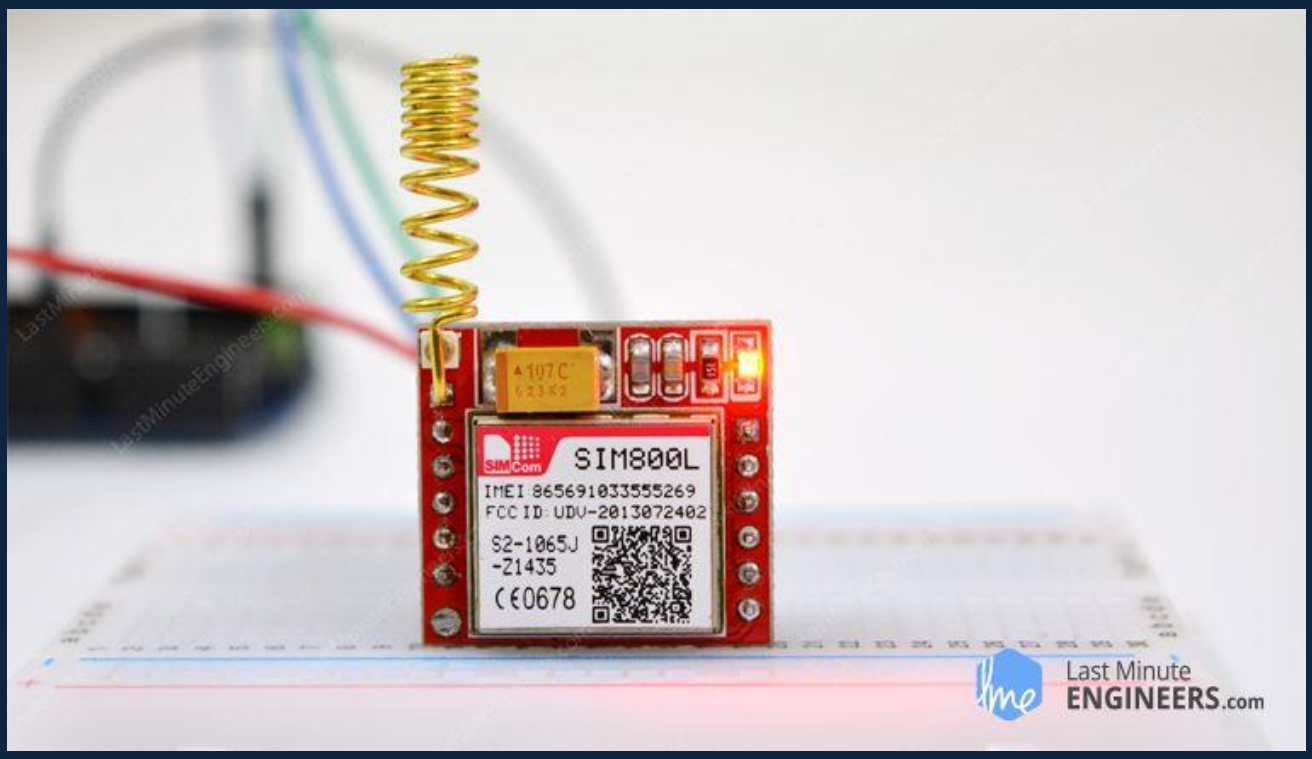
This module connects with the raspberry pi execute the facial recognition process by turning a Raspberry Pi into a camera.

4- Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light

on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing. So that it's the best choice to quickly execute orders like sensor any actions with system and reply with the order you make like making it messaging you through attach it with sim module.

5- **SIM module**

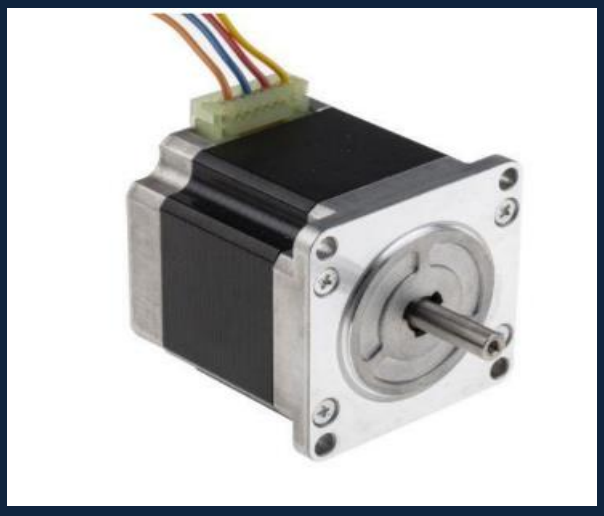


This Module is small and cheap module for GPRS/GSM communication. It is common with Arduino and microcontroller in most of embedded application. The

module offers GPRS/GSM technology for communication with the uses of a mobile sim. It uses a 900 and 1800MHz frequency band and allows users to receive/send mobile calls and SMS. The keypad and display interface allows the developers to make the customize application with it. Furthermore, it also has modes, command mode and data mode. In every country the GPRS/GSM and different protocols/frequencies to operate. Command mode helps the developers to change the default setting according to their requirements.

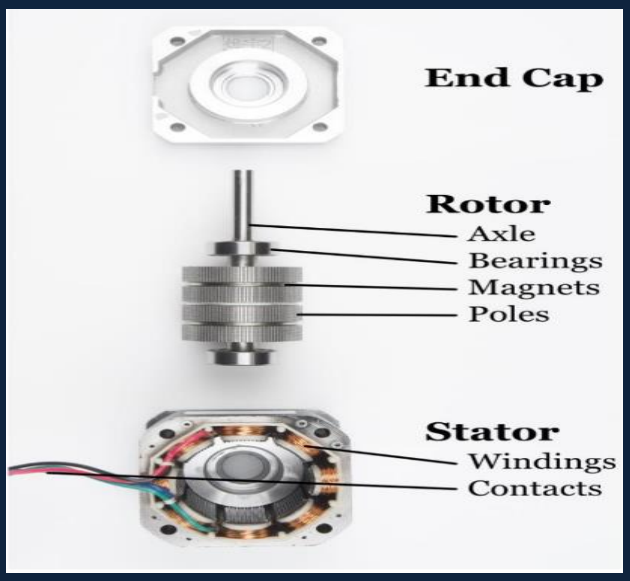
6- DC Motor

Motors used usually as actuators to drive the moving parts with the control of an external driver. A lot of motor types have been commonly used for controlling mechanical devices. In this proposed design the DC motor high torque and stepper motor will be used as they have the advantages which suitable for this design.



Components of Stepper Motors

A brushed DC motor is made up of 6 basic components:



Stepper Motors Components

1- Axle - Transfers the mechanical power of the motor to the user application.

2- Bearings - Minimizes friction for the axle.

3- Magnets - Provide a magnetic field for the windings to attract and repel.

4- Poles - Increases the resolution of the step distance by focusing the magnetic field.

5- Windings - Converts electricity to a magnetic field that drives the axle.

6- Contacts - Brings power from the controller to the windings.

3.2 Software purposed component

The software is divided to two main parts:

1- Operating System

Linux

2- Programming language

Python

1- Linux

Linux is a community of open-source Unix like operating systems that are based on the Linux Kernel. It was initially released by Linus Torvalds on September 17, 1991. It is a free and open-source operating system and the source code can be modified and distributed to anyone commercially or noncommercial under the GNU General Public License. Initially, Linux was created for personal computers and gradually it was used in other machines like servers, mainframe computers, supercomputers, etc. Nowadays, Linux is also used in embedded systems like routers, automation controls, televisions, digital video recorders, video game consoles, smartwatches, etc.

2 Why Linux?

It is the Raspberry Pi operating system used

3 Advantages of Linux

1- Security

The Linux security feature is the main reason that it is the

most favorable option for developers. It is not completely safe, but it is less vulnerable than others. Each application needs to authorize by the admin user. The virus is not executed until the administrator provides the access password. Linux systems do not require any antivirus program.

2- Free

Certainly, the biggest advantage of the Linux system is that it is free to use. We can easily download it, and there is no need to buy the license for it. It is distributed under GNU GPL (General Public License). Comparatively, we have to pay a huge amount for the license of the other operating systems.

3- Stability

Linux is more stable than other operating systems. Linux does not require to reboot the system to maintain performance levels. It rarely hangs up or slow down. It has big up-times.

4- Performance

Linux system provides high performance over different networks. It is capable of handling a large number of users simultaneously.

5- Flexibility

Linux operating system is very flexible. It can be used for desktop applications, embedded systems, and server applications too. It also provides various restriction options for specific computers. We can install only necessary components for a system.

2- Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language − Python is a great

Language for the beginner-level programmers because of its simplest.

4 Why Python With AI project?

Artificial intelligence is considered to be the trending technology of the future. Already there are a number of applications made on it. Due to this, many companies and researchers are taking interest in it. But the main question that arises here is that in which programming language can these AI applications be developed? There are various programming languages like Lisp, Prolog, C++, Java and Python, which can be used for developing applications of AI. Among them, Python programming language gains a huge popularity and the reasons are as follows:

**** Simple syntax & less coding**

Python involves very less coding and simple syntax among other programming languages which can be used for developing AI applications. Due to this feature, the testing can be easier and we can focus more on programming.

**** Inbuilt libraries for AI projects**

A major advantage for using Python for AI is that it comes with inbuilt libraries. Python has libraries for almost all kinds of AI projects. For example, NumPy, SciPy, matplotlib, nltk, SimpleAI are some the important inbuilt libraries of Python.

Open source − Python is an open source programming language. This makes it widely popular in the community.

Can be used for broad range of programming − Python can be used for a broad range of programming tasks like small shell script to enterprise web applications. This is another reason Python is suitable for AI projects.

5 Advantage of Python

1- Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

2- Easy-to-read − Python code is more clearly defined and visible to the eyes.

3- Easy-to-maintain − Python's source code is fairly easy-to maintain. A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

4- Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

5- Databases − Python provides interfaces to all major commercial databases.

6- GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Chapter 4

Face Recognition

4.1 Face Recognition Technology

As we said at the previous chapter, one of the benefits of using the biometric lock systems that may protect and get us away from the transmission of the diseases and viruses like corona virus the biometric lock systems and the technology that help us to this protection is face recognition this technology doesn’t need to be contact with any physical attachment so you can avoid any infection that may happen.

Facial recognition is a Biometric Artificial Intelligence based application that can uniquely identify a person by analyzing patterns based on the person’s facial textures and shape. A facial recognition system is a technology capable of identifying or verifying a person from a video source. Face recognition is a successful application of image analysis and understanding, which is applied in many aspects, such as business, security, crime detection and tracking any anonymous action from strangers and this feature is useful for more security.

5.2 Face recognition libraries

For using the technology, we need to learn about its libraries which it works on so for make face processing and do face recognition we need to learn about pillow, Open-CV, Pickle, Face recognition and Imutils.

So for pillow,

In today's digital world, we come across lots of digital images. In case, we are working with Python programming language, it provides lot of image processing libraries to add image processing capabilities to digital images.

Some of the most common image processing libraries are: OpenCV, Python Imaging Library (PIL), Scikit-image, Pillow. However, in this tutorial, we are only focusing on Pillow module and will try to explore various capabilities of this module.

Pillow is built on top of PIL (Python Image Library). PIL is one of the important modules for image processing in Python. However, the PIL module is not supported since 2011 and doesn't support python 3.

Pillow module gives more functionalities, runs on all major operating system and support for python 3. It supports wide variety of images such as "jpeg", "png", "bmp", "gif", "ppm", "tiff". You can do almost anything on digital images using pillow module. Apart from basic image processing functionality, including point operations, filtering images using built-in convolution kernels, and color space conversions.

OpenCV

Is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on

image processing, video capture and analysis including features like face detection and object detection.

Face Recognition

Recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library.

Pickle

Python's terminology for serialization and deserialization is pickling and unpickling respectively. The pickle module in Python library, uses very Python specific data format. Hence, non-Python applications may not be able to deserialize pickled data properly. It is also advised not to unpickle data from un-authenticated source.

So, You can get facial recognition encoding from facial recognition and store it in pickle file. You can read this encoding again and match against the faces in image and video file. If it is match then it will throw known_name which we stored in program. If it is not found then it will throw unknown.

NumPy

NumPy is a Python library used for working with arrays.

It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in

2005 by Travis Oliphant. It is an open source project and you

can use it freely. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

Chapter 5
Fingerprint

5.1 Fingerprint Technology

A fingerprint is the representation of the dermal ridges of a finger. Dermal ridges form through a combination of genetic and environmental factors; the genetic code in DNA gives general instructions on the way skin should form in a developing fetus, but the specific way it forms is a result of random events such as the exact position of the fetus in the womb at a particular moment. This is the reason why even the fingerprints of identical twins are different. Fingerprints are fully formed at about 7 months of fetus development and finger ridge configurations do not change throughout the life of an individual, except in case of accidents such as severe cuts on the fingertips. This stability makes fingerprints a very attractive biometric identifier. Several mathematical models based on the anatomy of friction ridge skin were developed over the years to quantify fingerprint individuality and to prove that finding two persons with identical fingerprints is extremely unlikely. This does not imply that fingerprint recognition is a perfect technique: in fact, various kinds of errors can affect fingerprint acquisition and processing thus requiring to introduce thresholds to decide if two fingerprint impressions are similar enough to be considered to belong to the same person. As for any biometric technique, a sound performance evaluation (see fingerprint databases and evaluation) is extremely important to estimate the accuracy of

a fingerprint-based biometric system and to understand if it is well-suited for a particular application. Recent independent evaluation campaigns such as FVC2006 proved that state-of-the-art fingerprint recognition algorithms are nowadays very accurate (i.e., EER less than 0.1% for a database collected with a large area optical scanner).

So after all of this the fingerprint recognition regard as an unique biometric lock way so that using it with our project add more security and make it more credibility.

To add this way, we needed to the make our code

Chapter 6

ARMY AKE code

As face recognition and fingerprint codes has been discussed in the previously, in this chapter a third security
merging code of both codes is presented which is the most project preferred authentication way and is called with our project name ARMY AKE code Authentication

Chapter 7
Credibility & Mechanism

7- Credibility & Mechanism

7.1 SMS alarm messages and Credibility

Any system in the world is possible to compromised to hack and to secure our system and solve this problem the system is programmed to send SMS messages to the mobile when any action happens at any time to alarm us anytime a person is trying to open the lock through the following using SIM module code presen

7.2 Mechanism

After security steps is done and the authentication is gained, the next step for the person who has the access to the system is to open the door and this is done using a DC Motor.

Chapter 8
Conclusion and future work

Conclusion:

By performing a biometric locker based on face recognition and finger print together with mobile application provides a high level of security, fast, reliable, immune to viruses, environment friendly system, with no need for mechanical keys and can be transferred independently to any device (for example: there is no need to buy the latest car to add this technology.

Future Work:

In future we aim to use pattern recognition and mobile application to increase the system security.

Relation with Environment:

It is environment friendly

Chapter 9

Appendix

9.1 Software Codes

9.1.1 Face recognition code

```
import face_recognition
import cv2
import pickle
import os
import numpy as np

import tkinter as tk
from tkinter import messagebox

from pathlib import Path
import glob


# class: Dlib Face Unlock
# Purpose: This class will update the encoded known face if the directory has changed
# as well as encoding a face from a live feed to compare the face to allow the facial recognition
# to be integrated into the system
# Methods: ID
class Dlib_Face_Unlock:
    # When the Dlib Face Unlock Class is first initialised it will check if the employee photos directory has been updated
    # if an update has occurred either someone deleting their face from the system or someone adding their face to the system
    # the face will then be encoded and saved to the encoded pickle file
    def __init__(self):
        # this is to detect if the directory is found or not
        try:
            # this will open the existing pickle file to load in the encoded faces of the users who has sign up for the service
            with open(r'C:\Users\barry\PycharmProjects\face_rec\labels.pickle', 'rb') as self.f:
                self.og_labels = pickle.load(self.f)
            print(self.og_labels)
        # error checking
        except FileNotFoundError:
            # allowing me to known that their was no file found
            print("No label.pickle file detected, will create required pickle files")
```

```
        # this will be used to for selecting the photos
        self.current_id = 0
        # creating a blank ids dictionary
        self.labels_ids = {}
        # this is the directory where all the users are stored
        self.BASE_DIR = os.path.dirname(os.path.abspath(__file__))
        self.image_dir = os.path.join(self.BASE_DIR, 'images')
        for self.root, self.dirs, self.files in os.walk(self.image_dir):
            # checking each folder in the images directory
            for self.file in self.files:
                # looking for any png or jpg files of the users
                if self.file.endswith('png') or self.file.endswith('jpg'):
                    # getting the folder name, as the name of the folder will be the user
                    self.path = os.path.join(self.root, self.file)
                    self.label = os.path.basename(os.path.dirname(self.path)).replace(' ', '-').lower()
                    if not self.label in self.labels_ids:
                        # adding the user into the labels_id dictionary
                        self.labels_ids[self.label] = self.current_id
                        self.current_id += 1
                        self.id = self.labels_ids[self.label]

        print(self.labels_ids)
        # this is compare the new label ids to the old label ids dictionary seeing if their has been any
new users or old users
        # being added to the system, if there is no change then nothing will happen
        self.og_labels = 0
        if self.labels_ids != self.og_labels:
            # if the dictionary change then the new dictionary will be dump into the pickle file
            with open('labels.pickle', 'wb') as self.file:
                pickle.dump(self.labels_ids, self.file)

            self.known_faces = []
            for self.i in self.labels_ids:
                # Get number of images of a person
                noOfImgs = len([filename for filename in os.listdir('images/' + self.i)
                                if os.path.isfile(os.path.join('images/' + self.i, filename))])
                print(noOfImgs)
                for imgNo in range(1, (noOfImgs + 1)):
                    self.directory = os.path.join(self.image_dir, self.i, str(imgNo) + '.png')
                    self.img = face_recognition.load_image_file(self.directory)
                    self.img_encoding = face_recognition.face_encodings(self.img)[0]
                    self.known_faces.append([self.i, self.img_encoding])
```

```
            print(self.known_faces)
            print("No Of Imgs" + str(len(self.known_faces)))
            with open('KnownFace.pickle', 'wb') as self.known_faces_file:
                pickle.dump(self.known_faces, self.known_faces_file)
        else:
            with open(r'CC:\Users\barry\PycharmProjects\face_rec\KnownFace.pickle', 'rb') as
self.faces_file:
                self.known_faces = pickle.load(self.faces_file)
            print(self.known_faces)

    # Method: ID
    # Purpose:This is method will be used to create a live feed .i.e turning on the devices camera
    # then the live feed will be used to get an image of the user and then encode the users face
    # once the users face has been encoded then it will be compared to in the known faces
    # therefore identifying the user
    def ID(self):
        # turning on the camera to get a photo of the user frame by frame
        self.cap = cv2.VideoCapture(0)
        # seting the running variable to be true to allow me to known if the face recog is running
        self.running = True
        self.face_names = []
        while self.running == True:
            # taking a photo of the frame from the camera
            self.ret, self.frame = self.cap.read()
            # resizing the frame so that the face recog module can read it
            self.small_frame = cv2.resize(self.frame, (0, 0), fx=0.5, fy=0.5)
            # converting the image into black and white
            self.rgb_small_frame = self.small_frame[:, :, ::-1]
            if self.running:
                # searching the black and white image for a face
                self.face_locations = face_recognition.face_locations(self.frame)

                # if self.face_locations == []:
                #     Dlib_Face_Unlock.ID(self)
                # it will then encode the face into a matrix
                self.face_encodings = face_recognition.face_encodings(self.frame, self.face_locations)
                # creating a names list to append the users identify into
                self.face_names = []
                # looping through the face_encoding that the system made
                for self.face_encoding in self.face_encodings:
                    # looping though the known_faces dictionary
                    for self.face in self.known_faces:
                        # using the compare face method in the face recognition module
```

```
                    self.matches = face_recognition.compare_faces([self.face[1]], self.face_encoding)
                    print(self.matches)
                    self.name = 'Unknown'
                    # compare the distances of the encoded faces
                    self.face_distances = face_recognition.face_distance([self.face[1]],
self.face_encoding)
                    # uses the numpy module to comare the distance to get the best match
                    self.best_match = np.argmin(self.face_distances)
                    print(self.best_match)
                    print('This is the match in best match', self.matches[self.best_match])
                    if self.matches[self.best_match] == True:
                        self.running = False
                        self.face_names.append(self.face[0])
                        break
                    next
            print("The best match(es) is" + str(self.face_names))
            self.cap.release()
            cv2.destroyAllWindows()
            break
        return self.face_names


"""
dfu = Dlib_Face_Unlock()
dfu.ID()
"""


def register():
    # Create images folder
    if not os.path.exists("images"):
        os.makedirs("images")
    # Create folder of person (IF NOT EXISTS) in the images folder
    Path("images/" + name.get()).mkdir(parents=True, exist_ok=True)
    # Obtain the number of photos already in the folder
    numberOfFile = len([filename for filename in os.listdir('images/' + name.get())
                    if os.path.isfile(os.path.join('images/' + name.get(), filename))])
    # Add 1 because we start at 1
    numberOfFile += 1
    # Take a photo code
    cam = cv2.VideoCapture(0)

    cv2.namedWindow("test")
```

```
    while True:
        ret, frame = cam.read()
        cv2.imshow("test", frame)
        if not ret:
            break
        k = cv2.waitKey(1)

        if k % 256 == 27:
            # ESC pressed
            print("Escape hit, closing...")
            cam.release()
            cv2.destroyAllWindows()
            break
        elif k % 256 == 32:
            # SPACE pressed
            img_name = str(numberOfFile) + ".png"
            cv2.imwrite(img_name, frame)
            print("{} written!".format(img_name))
            os.replace(str(numberOfFile) + ".png", "images/" + name.get().lower() + "/" +
str(numberOfFile) + ".png")
            cam.release()
            cv2.destroyAllWindows()
            break
    raiseFrame(loginFrame)


# Passing in the model
def login():
    # After someone has registered, the face scanner needs to load again with the new face
    dfu = Dlib_Face_Unlock()
    # Will return the user's name as a list, will return an empty list if no matches
    user = dfu.ID()
    if user == []:
        messagebox.showerror("Alert", "Face Not Recognised")
        return
    loggedInUser.set(user[0])
    raiseFrame(userMenuFrame)


# Tkinter
root = tk.Tk()
root.title("Face Login Example")
```

```
# Frames
loginFrame = tk.Frame(root)
regFrame = tk.Frame(root)
userMenuFrame = tk.Frame(root)

# Define Frame List
frameList = [loginFrame, regFrame, userMenuFrame]
# Configure all Frames
for frame in frameList:
    frame.grid(row=0, column=0, sticky='news')
    frame.configure(bg='white')


def raiseFrame(frame):
    frame.tkraise()


def regFrameRaiseFrame():
    raiseFrame(regFrame)


def logFrameRaiseFrame():
    raiseFrame(loginFrame)


# Tkinter Vars
# Stores user's name when registering
name = tk.StringVar()
# Stores user's name when they have logged in
loggedInUser = tk.StringVar()

tk.Label(loginFrame, text="Face Recognition", font=("Courier", 60), bg="white").grid(row=1,
column=1, columnspan=5)
loginButton = tk.Button(loginFrame, text="Login", bg="white", font=("Arial", 30),
command=login)
loginButton.grid(row=2, column=5)
regButton = tk.Button(loginFrame, text="Register", command=regFrameRaiseFrame, bg="white",
font=("Arial", 30))
regButton.grid(row=2, column=1)

tk.Label(regFrame, text="Register", font=("Courier", 60), bg="white").grid(row=1, column=1,
columnspan=5)
tk.Label(regFrame, text="Name: ", font=("Arial", 30), bg="white").grid(row=2, column=1)
```

```
nameEntry = tk.Entry(regFrame, textvariable=name, font=("Arial", 30)).grid(row=2, column=2)

registerButton = tk.Button(regFrame, text="Register", command=register, bg="white",
font=("Arial", 30))
registerButton.grid(row=3, column=2)

tk.Label(userMenuFrame, text="Hello, ", font=("Courier", 60), bg="white").grid(row=1, column=1)
tk.Label(userMenuFrame, textvariable=loggedInUser, font=("Courier", 60), bg="white",
fg="red").grid(row=1, column=2)
tk.Button(userMenuFrame, text="Back", font=("Arial", 30),
command=logFrameRaiseFrame).grid(row=2, column=1)

# Load Faces
dfu = Dlib_Face_Unlock()
raiseFrame(loginFrame)
root.mainloop()
```

the last code of the face recognition process to recognize the identification of the user:

```
# import the necessary packages
from imutils import paths
import face_recognition
#import argparse
import pickle
import cv2
import os

# our images are located in the dataset folder
print("[INFO] start processing faces…")
imagePaths = list(paths.list_images("dataset"))

# initialize the list of known encodings and known names
knownEncodings = []
knownNames = []

# loop over the image paths
for (i, imagePath) in enumerate(imagePaths):
	# extract the person name from the image path
	print("[INFO] processing image {}/{}".format(i + 1,
		len(imagePaths)))
	name = imagePath.split(os.path.sep)[-2]

	# load the input image and convert it from RGB (OpenCV ordering)
	# to dlib ordering (RGB)
	image = cv2.imread(imagePath)
	rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

	# detect the (x, y)-coordinates of the bounding boxes
	# corresponding to each face in the input image
	boxes = face_recognition.face_locations(rgb,
		model="hog")

	# compute the facial embedding for the face
	encodings = face_recognition.face_encodings(rgb, boxes)

	# loop over the encodings
	for encoding in encodings:
		# add each encoding + name to our set of known names and
		# encodings
		knownEncodings.append(encoding)
		knownNames.append(name)

# dump the facial encodings + names to disk
print("[INFO] serializing encodings…")
data = {"encodings": knownEncodings, "names": knownNames}
f = open("encodings.pickle", "wb")
f.write(pickle.dumps(data))
f.close()
```

9.1.2 Fingerprint code

```
import RPi.GPIO as GPIO
ledPin = 23
duty = 75
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(ledPin,GPIO.OUT)
GPIO.output(ledPin,0)
import time
import board
#import busio
from digitalio import DigitalInOut, Direction
import adafruit_fingerprint

led = DigitalInOut(board.D13)
led.direction = Direction.OUTPUT

#uart = busio.UART(board.TX, board.RX, baudrate=57600)
# If using with a computer such as Linux/RaspberryPi, Mac, Windows with USB/serial converter:
# import serial
# uart = serial.Serial("/dev/ttyUSB0", baudrate=57600, timeout=1)
# If using with Linux/Raspberry Pi and hardware UART:
import serial
uart = serial.Serial("/dev/ttyS0", baudrate=57600, timeout=1)

finger = adafruit_fingerprint.Adafruit_Fingerprint(uart)

def get_fingerprint():
    """Get a finger print image, template it, and see if it matches!"""
    print("Waiting for image...")
    while finger.get_image() != adafruit_fingerprint.OK:
        pass
    print("Templating...")
    if finger.image_2_tz(1) != adafruit_fingerprint.OK:
        return False
    print("Searching...")
    if finger.finger_search() != adafruit_fingerprint.OK:
        return False
    return True

# pylint: disable=too-many-branches
def get_fingerprint_detail():
    """Get a finger print image, template it, and see if it matches!
    This time, print out each error instead of just returning on failure"""
    print("Getting image...", end="", flush=True)
    i = finger.get_image()
    if i == adafruit_fingerprint.OK:
        print("Image taken")
    else:
        if i == adafruit_fingerprint.NOFINGER:
            print("No finger detected")
        elif i == adafruit_fingerprint.IMAGEFAIL:
            print("Imaging error")
        else:
            print("Other error")
```

```
print("Templating...", end="", flush=True)
    i = finger.image_2_tz(1)
    if i == adafruit_fingerprint.OK:
        print("Templated")
    else:
        if i == adafruit_fingerprint.IMAGEMESS:
            print("Image too messy")
        elif i == adafruit_fingerprint.FEATUREFAIL:
            print("Could not identify features")
        elif i == adafruit_fingerprint.INVALIDIMAGE:
            print("Image invalid")
        else:
            print("Other error")
        return False

    print("Searching...", end="", flush=True)
    i = finger.finger_fast_search()
    # pylint: disable=no-else-return
    # This block needs to be refactored when it can be tested.
    if i == adafruit_fingerprint.OK:
        print("Found fingerprint!")
        return True
    else:
        if i == adafruit_fingerprint.NOTFOUND:
            print("No match found")
        else:
            print("Other error")
        return False

# pylint: disable=too-many-statements
def enroll_finger(location):
    """Take a 2 finger images and template it, then store in 'location'"""
    for fingerimg in range(1, 3):
        if fingerimg == 1:
            print("Place finger on sensor...", end="", flush=True)
        else:
            print("Place same finger again...", end="", flush=True)

        while True:
            i = finger.get_image()
            if i == adafruit_fingerprint.OK:
                print("Image taken")
                break
            if i == adafruit_fingerprint.NOFINGER:
                print(".", end="", flush=True)
            elif i == adafruit_fingerprint.IMAGEFAIL:
                print("Imaging error")
                return False
            else:
                print("Other error")
                return False

        print("Templating...", end="", flush=True)
        i = finger.image_2_tz(fingerimg)
        if i == adafruit_fingerprint.OK:
            print("Templated")
        else:
            if i == adafruit_fingerprint.IMAGEMESS:
                print("Image too messy")
```

```
elif i == adafruit_fingerprint.FLASHERR:
            print("Flash storage error")
        else:
            print("Other error")
        return False

    return True


##################################################


def get_num():
    """Use input() to get a valid number from 1 to 127. Retry till success!"""
    i = 0
    while (i > 127) or (i < 1):
        try:
            i = int(input("Enter ID # from 1-127: "))
        except ValueError:
            pass
    return i


while True:
    print("----------------")
    if finger.read_templates() != adafruit_fingerprint.OK:
        raise RuntimeError("Failed to read templates")
    print("Fingerprint templates:", finger.templates)
    print("e) enroll print")
    print("f) find print")
    print("d) delete print")
    print("----------------")
    c = input("> ")

    if c == "e":
        enroll_finger(get_num())
    if c == "f":
        if get_fingerprint():
            print("we found #", finger.finger_id, "with acuracy", finger.confidence)
            GPIO.output(ledPin,GPIO.HIGH)
            time.sleep(1)
            GPIO.output(ledPin,GPIO.LOW)
            time.sleep(0.5)

        else:
            print("Finger not found")
    if c == "d":
        if finger.delete_model(get_num()) == adafruit_fingerprint.OK:
            print("Deleted!")
        else:
            print("Failed to delete")
```

9.1.3 ARMY AKE preferred code

```
data = pickle.loads(open(encodingsP, "rb").read())
GPIO.setwarnings(False)

# Pin Definitions
ledPin = 23
ledPinblue = 21
butPin = 26
butPin2 = 13
in1 = 25
in2 = 8
in3cooling=4
gsm_pin=12
lighter=5
# Setup GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(ledPin,GPIO.OUT)
GPIO.setup(ledPinblue,GPIO.OUT)
GPIO.setup(gsm_pin,GPIO.OUT)
GPIO.setup(lighter,GPIO.OUT)
GPIO.setup(in1,GPIO.OUT)
GPIO.setup(in2,GPIO.OUT)
GPIO.setup(in3cooling,GPIO.OUT)

GPIO.setup(butPin,GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(butPin2,GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.output(ledPin, GPIO.LOW)
x=1
t=1

while 1:
    time.sleep(0.1)
    x=GPIO.input(butPin)
    t=GPIO.input(butPin2)
    print("x=",x)
    print("t=",t)
    data = pickle.loads(open(encodingsP, "rb").read())
    # start the FPS counter
    fps = FPS().start()
    prevTime = 0
    doorUnlock = False
    GPIO.output(lighter,GPIO.HIGH)
#=============================================================
=========================================
    if  x ==1:            # Not Pressed the prijectis  close ========
        GPIO.output(ledPin, GPIO.HIGH)
        GPIO.output(in1,GPIO.LOW)
        GPIO.output(in2,GPIO.LOW)
        GPIO.output(in3cooling,GPIO.HIGH)
        GPIO.output(lighter,GPIO.LOW)
        GPIO.output(gsm_pin,GPIO.LOW)
        GPIO.output(ledPinblue,GPIO.LOW)
        print("cclosed")
```

```
if x==0:

        data = pickle.loads(open(encodingsP, "rb").read())
        # start the FPS counter
        fps = FPS().start()
        prevTime = 0
        doorUnlock = False
        GPIO.output(lighter,GPIO.HIGH)
        # loop over frames from the video file stream
        while True:
           GPIO.output(lighter,GPIO.HIGH)
           # grab the frame from the threaded video stream and resize it
           # to 500px (to speedup processing)
           frame = vs.read()
           frame = imutils.resize(frame, width=500)

            # convert the input frame from (1) BGR to grayscale (for face
            # detection) and (2) from BGR to RGB (for face recognition)
           gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
           rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

            # detect faces in the grayscale frame
           rects = detector.detectMultiScale(gray, scaleFactor=1.1,
                   minNeighbors=5, minSize=(30, 30),
                   flags=cv2.CASCADE_SCALE_IMAGE)

            # OpenCV returns bounding box coordinates in (x, y, w, h) order
            # but we need them in (top, right, bottom, left) order, so we
            # need to do a bit of reordering
           boxes = [(y, x + w, y + h, x) for (x, y, w, h) in rects]

            # compute the facial embeddings for each face bounding box
           encodings = face_recognition.face_encodings(rgb, boxes)
           names = []

            # loop over the facial embeddings
           for encoding in encodings:

                        # attempt to match each face in the input image to our known
                        # encodings
              matches = face_recognition.compare_faces(data["encodings"],encoding)

              name = "Unknown" #if face is not recognized, then print Un
                        # check to see if we have found a match
              if True in matches:

                                # find the indexes of all matched faces then initialize a
                                # dictionary to count the total number of times each face
                                # was matched
                 matchedIdxs = [i for (i, b) in enumerate(matches) if b]
                 counts = {}
```

```
# update the list of names
                    names.append(name)

        #lock the door after 5 seconds


             # loop over the recognized faces
            for ((top, right, bottom, left), name) in zip(boxes, names):


            # draw the predicted face name on the image – color is in BGR
                cv2.rectangle(frame, (left, top), (right, bottom),(0, 255, 0), 2)
                y = top - 15 if top - 15 > 15 else top + 15
                cv2.putText(frame, name, (left, y), cv2.FONT_HERSHEY_SIMPLEX,.8, (255, 0, 0), 2)

             # display the image to our screen
            cv2.imshow("Facial Recognition is Running", frame)
            key = cv2.waitKey(1) & 0xFF

             # quit when 'q' key is pressed

            t=GPIO.input(butPin2)
            if key == ord("q"):
                break
            if t == 0:

                fps.stop()
# do a bit of cleanup
                cv2.destroyAllWindows()
                vs.stop()
                break
             # update the FPS counter
            fps.update()

# stop the timer and display FPS information
        fps.stop()


# do a bit of cleanup
        cv2.destroyAllWindows()
        vs.stop()

        GPIO.output(ledPin, GPIO.HIGH)
        GPIO.output(lighter, GPIO.HIGH)
#==================================
    if t==0:
        print("door closing")
        GPIO.output(ledPin, GPIO.HIGH)
        GPIO.output(in1,GPIO.LOW)
        GPIO.output(in2,GPIO.HIGH)
        GPIO.output(in3cooling,GPIO.HIGH)
        GPIO.output(lighter,GPIO.LOW)
        GPIO.output(gsm_pin,GPIO.LOW)
        GPIO.output(ledPinblue,GPIO.LOW)
        print("door still locked")
        time.sleep(3)
        cv2.destroyAllWindows()
        vs.stop()
```

9.1.4 SIM module code

```
#include <SoftwareSerial.h>

//Create software serial object to communicate with SIM800L
SoftwareSerial mySerial(3, 2); //SIM800L Tx & Rx is connected to Arduino #3 & #2

void setup()
{
  //Begin serial communication with Arduino and Arduino IDE (Serial Monitor)
  Serial.begin(9600);

  //Begin serial communication with Arduino and SIM800L
  mySerial.begin(9600);

  Serial.println("Initializing...");
  delay(1000);

  mySerial.println("AT"); //Once the handshake test is successful, it will back to OK
  updateSerial();

  mySerial.println("AT+CMGF=1"); // Configuring TEXT mode
  updateSerial();
  mySerial.println("AT+CMGS=\"+ZZxxxxxxxxxx\"");//change ZZ with country code and xxxxxxxxxxx with phone
number to sms
  updateSerial();
  mySerial.print("Last Minute Engineers | lastminuteengineers.com"); //text content
  updateSerial();
  mySerial.write(26);
}

void loop()
{
}

void updateSerial()
{
  delay(500);
  while (Serial.available())
  {
    mySerial.write(Serial.read());//Forward what Serial received to Software Serial Port
  }
  while(mySerial.available())
  {
    Serial.write(mySerial.read());//Forward what Software Serial received to Serial Port
  }
}
```

9.1.5 DC Motor code

```
import RPi.GPIO as GPIO
import time

in1 = 25
in2 = 8
butPin = 26
butPin2 =13


GPIO.setmode(GPIO.BCM)
GPIO.setup(in1,GPIO.OUT)
GPIO.setup(18,GPIO.OUT)
GPIO.setup(in2,GPIO.OUT)
GPIO.output(in1,GPIO.LOW)
GPIO.output(in2,GPIO.LOW)
GPIO.setup(butPin,GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(butPin2,GPIO.IN, pull_up_down=GPIO.PUD_UP)
t=1
x=1
while 1:
    time.sleep(0.1)
    x=GPIO.input(butPin)
    t=GPIO.input(butPin2)
    print("t=",t)
    print("x=",x)
    if x==0:
        print("run")
        GPIO.output(18,GPIO.HIGH)
        GPIO.output(in1,GPIO.HIGH)
        GPIO.output(in2,GPIO.LOW)
        print("forward")
        time.sleep(3)
        GPIO.output(in1,GPIO.LOW)
        GPIO.output(in2,GPIO.LOW)


    if t==0:
        GPIO.output(in1,GPIO.LOW)
        GPIO.output(in2,GPIO.HIGH)
        print("backward")
        time.sleep(5)
        GPIO.output(in1,GPIO.LOW)
        GPIO.output(in2,GPIO.LOW)
    if x==1:
        GPIO.output(in1,GPIO.LOW)
        GPIO.output(in2,GPIO.LOW)
        print("cclosed")
        GPIO.output(18,GPIO.LOW)

    if t==1:
        GPIO.output(in1,GPIO.LOW)
        GPIO.output(in2,GPIO.LOW)
        print("cclosed")
```

9.2 IEEE Standards:

The IEEE Standards Association (IEEE SA) is **a globally recognized standards-setting body within IEEE**. It develops consensus standards through an open process that engages industry and brings together a broad stakeholder community.

Following are the IEEE Standards used in project:

Arduino: IEEE Std 1855

Dc Motor: IEEE 113-1985

References

References

1. William Thompson, et al., "Latent Fingerprint Examination," AAAS September 15, 2017. https://www.aaas.org/sites/default/files/reports/Latent%20Fingerprint%20Report%20FINAL%209_14.pdf?i9xGS_EyMHnIPLG6INIUyZb66L5cLdlb ; "Report on the Erroneous Fingerprint Individualization in the Madrid Train Bombing Case." FBI Forensic Science Communications 7(1) 2005. https://archives.fbi.gov/archives/about-us/lab/forensic-science-communications/fsc/jan2005/special_report/2005_special_report.htm ;Ulery, Bradford T., et al. "Accuracy and reliability of forensic latent fingerprint decisions," Proceedings of the National Academy of Sciences 108(19), May 10, 2011. https://www.pnas.org/content/108/19/7733
2. Alex Hern, "Hacker Fakes German Minister's Fingerprints using Photos of Her Hands." The Guardian December 30, 2014, https://www.theguardian.com/technology/2014/dec/30/hacker-fakes-german-ministers-fingerprints-using-photos-of-her-hands.
3. For instance, IDEMIA's MorphoWave system collects data from four fingers at once. https://www.idemia.com/contactless-fingerprint.
4. "Automated Fingerprint Identification System (AFIS) Overview - A Short History," Thales Group, April 5, 2021. https://www.thalesgroup.com/en/markets/digital-identity-and-security/government/biometrics/afis-history.
5. William Crumpler, "How Accurate Are Facial Recognition Systems – and

Why Does It Matter?" Center for Strategic & International Studies, April 14, 2020. https://www.csis.org/blogs/technology-policy-blog/how-accurate-are-facial-recognition-systems-%E2%80%93-and-why-does-it-matter.

6. **"Facial Recognition: Top 7 Trends (Tech, Vendors, Markets, Use Cases & Latest News)," Thales Group April 10, 2021. https://www.thalesgroup.com/en/markets/digital-identity-and-security/government/biometrics/facial-recognition.**

7. **Drew Harwell, "This Facial Recognition Website Can Turn Anyone Into a Cop – Or a Stalker," The Washington Post,**

   May 14, 2021,

https://www.washingtonpost.com/technology/2021/05/14/pimeyes-facial-recognition-search-secrecy/

8. **Sergio Mannino, "Council Post: How Facial Recognition Will Change Retail," Forbes May 8, 2020.**

https://www.forbes.com/sites/forbesbusinesscouncil/2020/05/08/how-facial-recognition-will-change-retail/?sh=19df789f3daa .

9. **Rajeev Ranjan, et al., "A Fast and Accurate System for Face Detection, Identification, and Verification," IEEE Transactions on Biometrics, Behavior,**

and Identity Science 1(2) 2019: 82-96. https://ieeexplore.ieee.org/document/8680708 .

10. "Fingerprints: The Convoluted Patterns of Racism." Dickinson College. April 20, 2021, http://dh.dickinson.edu/digitalmuseum/exhibit-artifact/babes-in-the-woods/fingerprints .

11. The Bertillon System," U.S. National Library of Medicine, April 20, 2021, https://www.nlm.nih.gov/exhibition/visibleproofs/galleries/technologies/bertillon.html .

12. https://armguard.com/tuxedo_touch.html

13. https://revolar.com/what-is-a-biometric-lock/#What_Are_the_Benefits_of_Using_a_Biometric_Lock

14. https://www.intervid-africa.co.za/blog/biometric-access-control-%E2%80%93-south-africa

15. https://link.springer.com/referenceworkentry/10.1007/978-0-387-73003-5_47

16. https://lastminuteengineers.com/sim800l-gsm-module-arduino-tutorial/

17. https://elocksecurity.com/6-reasons-why-biometric-fingerprint-door-locks-are-the-safest-security-solution

18. https://www.miteksystems.com/blog/advantages-and-disadvantages-of-biometrics

19. https://recfaces.com/articles/what-is-voice-recognition