DOOR UNLOCK SYSTEM USING FACE RECOGNITION AND BIOMETRIC DEVICE

A project report submitted in partial fulfillment for the award of degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

Submitted
By

P. Suneetha
(16MT1A0535)

K. Dhana Lakshmi
(16MT1A0520)

K. Prameela
(16MT1A0526)

CH. Sravani
(16MT1A0509)

D.Gowri Shankar
(16MT1A0511)

Under the Esteemed Guidance of

Sri R.V.L.S.N.Sastry

Associative Professor & Head of the Department

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SRI VENKATESWARA COLLEGE OF ENGINEERING & TECHNOLOGY
(Approved by AICTE, New Delhi; Affiliated to JNTUK, Kakinada; G.O.Ms.No.101,Dt. 09-07-08)
N H – 5, ETCHERLA – 532 410, SRIKAKULAM (Dist), Andhra Pradesh
Email: hodcse.svcet@gmail.com

2019-2020

SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY

(Approved by AICTE, NEW DELHI, Affiliated to JNTUK, Kakinada, AP)
Accredited by NAAC with "A'' Grade (An ISO 9001:2015 certified institution)
ETCHERLA, SRIKAKULAM-532410
(2016-2020)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project report entitled **"DOOR UNLOCK SYSTEM USING FACE RECOGNITION AND BIOMETRIC DEVICE"** being submitted by **P.SUNEETHA (16MT1A0535), CH.SRAVANI (16MT1A0509), D.GOWRI SHANKAR (16MT1A0511), K.DHANA LAXMI (16MT1A0520) and K.PRAMEELA (16MT1A0526),** in partial fulfillment for the award of the degree of bachelor of technology in **COMPUTER SCIENCE AND ENGINEERING** during ***2016-2020*** is a record of bonafied work carried out under the guidance and supervision of **Sri. R.V.L.S.N.SASTRY.** The result embodied in this project report has not been submitted to any other university or institute for the award of any degree.

Project Guide **Head of the Department**

External Examiner

ACKNOWLEDGEMENT

It gives us an immense pleasure to express sense of gratitude to our guide **Sri. R.V.L.S.N.SASTRY,** Associate Professor and Head of the Department of CSE for his whole hearted and valuable guidance throughout the project. Without his sustained and sincere effort, this project would not have taken this shape. He encouraged and helped us to overcome various difficulties that we have faced at various stages of our project. We would like to sincerely thank for providing all the necessary facilities that leads to the successful completion of our project.

We would like to take this opportunity to thank our beloved Principal **Dr. V.SURENDRA REDDY** for his great support during our project period.

We take this opportunity to express our deepest and sincere gratitude to **Dr. B.SRIRAM MURTHY, Director, VIKAS Educational Society**, for providing all facilities and support to meet our project requirements.

We would like to take this opportunity to express our sincere gratitude to **Sri. I. KISHORE, Director, ISVC**, for his support during our project period.

We would like to thank all the faculty members of CSE department for their direct and indirect support in helping us for the completion of our project. Also we would like to thank all our lab technicians of CSE department for their valuable suggestions and providing facilities in completion of our project work.

Finally we would like to thank all of our friends and my family members for their continuous help and encouragement.

P. Suneetha [16MT1A0535]

CH. Sravani [16MT1A0509]

D. Gowri Shankar [16MT1A0511]

K. Dhana laxmi [16MT1A0520]

K. Prameela [16MT1A0526]

ABSTRACT

Automation is a necessity in the current times as it makes processes more economical and affordable in the long run. Once a process is automated the only check that is to be performed is whether it is turned on or not. Automated processes are not prone to errors and even if an error is identified rectification is easy and can be applied system-wide without any delay. In this project- **Door Unlocking System** is proposed that uses facial recognition and biometric based IoT technology to automate the entire system.

Today we are facing security threats in every aspect. One of the problems we face in daily life is unauthorized persons entry which causes many security issues. So we can resolve our home or workplace entry level security issues by allow only authenticated or authorized people in without any manual checking by using **"Face Recognition and biometric based door unlock system"** that uses Raspberry pi. The Face recognition module to capture Human images and to compare with stored database, if it matches with the authorized user then the system will unlock the door by an electric door lock. Biometric device is used to capture the biometrics and to compare with stored database; if it matches with the authorized user then the system will unlock the door using an Electric Door Lock.

LIST OF FIGURES

CONTENTS

CHAPTER-1

INTRODUCTION

INTRODUCTION

1. The Door unlocking System

The most important feature of any home security system is to detect the people who enter the house. The major drawbacks in a common door lock are that anyone can open a conventional door lock by duplicating or stealing the key that's why we are using smart lock.

This door unlock system is used face recognition and biometric. The camera is used to capture the image and biometric device is used to scans biometrics and compare with the database. If the credentials were matched then provide the authentication.

The automated Door Unlocking system implemented with face recognition using image processing with combination of IoT technology will overcome the disadvantages of other proposed technologies.

This explains how students and teachers will use this attendance management system. Following points will make sure that attendance is marked correctly, without any problem:

(1) All the hardware will be inside home. So outside interference will be absent.

(2) To remove unauthorized access and unwanted attempt to corrupt the hardware by people, all the hardware except raspberry pi could be put inside small cabin.

(3) When a person enters the main lobby in front of main door, the face marking will start.

1.1 Overview

The most important feature of any home security system is to detect the people who enter the house.

The major drawbacks in a common door lock are that anyone can open a conventional door lock by duplicating or stealing the key that's why we are using smart lock.

This door unlock system is used face recognition and biometric. The camera is used to capture the image and biometric device is used to scans biometrics and compare with the database. If the credentials were matched then provide the authentication.

The proposed **Face Recognition And Biometric Door Unlock System** has been developed to prevent robbery in highly secure areas like home environment with lesser power consumption and more reliable standalone security device for door security. This system is powered by **Raspberry Pi** circuit.

ADVANTAGES:

- High reliability.
- It provides enough flexibility to suit the requirements.
- More secure due to face detection.

A Face Recognition is a computer application for automatically identifying or verifying a function from a digital image or a video frame from a video source.

One of the ways to do this is by comparing selected facial features from the image and a facial database.

Face recognition there are two types of comparisons

- 1.Verification
- 2.Identification

Image acquisition:

Facial scan technology can acquire faces from almost any static camera or video system that generates images of sufficient quality and resolution. High quality enrolment is essential to eventual verification and identification enrolment images define the face characteristics to be used in all future authentication events.

Image processing:

Images are cropped such that avoid facial image remains and color images are normally converted to black and white in order to facilitate initial comparisons based on grayscale characteristics.

First the presence of faces or face in a scene must be detected. Once the face is detected it must be localized and normalization process may be required to bring the dimensions of the live facial sample in alignment with the one on the template.

Process of Automated Door Unlock system:

The proposed model contains 5 phases to capturing the attendance:

1. Face Training: Train the system using trained dataset. The dataset contain number of pictures of different persons with different poses.

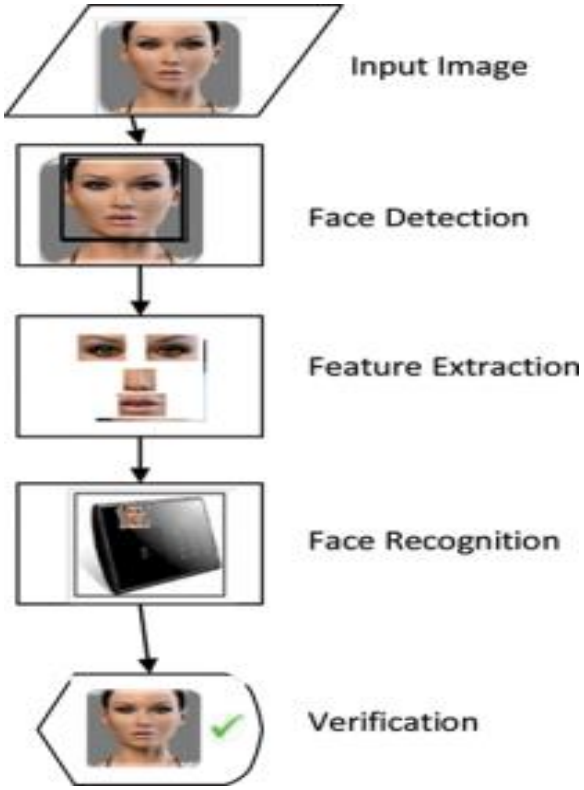


Fig 1: Face training process

2. Image Capturing: In this phase, using the camera, we run the program to start the camera that captures the images from the students.

3. Face Detection: The face detection phase identifies the face part from human bodies.

4. Face Recognition: In this phase it matches faces with the trained dataset. If any face is matched to the trained data, that displays the roll number. Otherwise it displays the unknown. Even more than once it recognizes the same faces of the persons, it considers only one time per day.

```
saikumar@saikumar-Dell:~/face-recognition-opencv$ python encode_faces.py --datas
et dataset --encodings encodings.pickle
[INFO] quantifying faces...
[INFO] processing image 1/15
[INFO] processing image 2/15
[INFO] processing image 3/15
[INFO] processing image 4/15
[INFO] processing image 5/15
[INFO] processing image 6/15
[INFO] processing image 7/15
[INFO] processing image 8/15
[INFO] processing image 9/15
[INFO] processing image 10/15
[INFO] processing image 11/15
[INFO] processing image 12/15
[INFO] processing image 13/15
[INFO] processing image 14/15
[INFO] processing image 15/15
[INFO] serializing encodings...
saikumar@saikumar-Dell:~/face-recognition-opencv$
```

Fig 2: Train the images in the dataset

5. Database Development: The recognized faces can be stored in the database.

CHAPTER -2

LITERATURE SURVEY

LITERATURE SURVEY

Literature Survey

The most important of feature of any home security system is to detect the people who enter or leave the house. Instead of monitoring that through passwords or pins unique faces can be made use of as they are one's biometric trait. These are innate and cannot be modified or stolen easily. The level of security can be raised by using face detection. The proposed face recognition and fingerprint recognition door lock security system has been developed to prevent robbery in highly secure areas like home environment with lesser power consumption and more reliable standalone security device for both Intruder detection and for door security. This system is powered by raspberry pi circuit. Raspberry Pi electronic board is operated on Battery power supply, wireless internet connectivity by using USB modem, it includes camera, PIR motion sensor and a door. Whenever the person comes in front of the door, it recognizes the face and if it is registered then it unlocks the door, if the face is not registered it will raise an alarm and clicks a picture and send it on the registered number. This is how the system works.

2.1 Stand-Alone Programs

2.1.1 Fingerprint Sensor:

Not only in Hollywood films, fingerprint readers are seen more and more frequently. Such modules are often installed in home surveillance systems and are used for the simple but secure verification of persons. With such a Raspberry Pi Fingerprint Sensor you can also implement some other projects, such as secured locks.

One of the advantages is that passwords and / or number codes can be completely omitted. Although this is still possible, but it's a lot more comfortable without. This tutorial is about the connection as well as the reading, saving and verifying of imported fingerprints.

ACCESSORIES:

A USB TTL adapter with 3.3V and 5V voltage output can be used for many serial modules.

These sensors were originally developed for the Arduino and can be read via UART. The Raspberry Pi has two pins (pin 8 / GPIO14 and pin 10 / GPIO 15), but they work with 3.3V. Since there are different fingerprint

sensors, which do not all work with 3.3V, a USB UART converter is recommended. Some models can be used with both 3.3V and 5V voltage. These are particularly suitable (also in connection with an Arduino).

The following is therefore required:

Raspberry Pi Fingerprint Sensor.Apart from the voltage the models do not differ much.

Serial USB converter with 3.3V and 5V connection

Female-Female Jumper wires , if not included with the USB converter.

Connection of the Raspberry Pi Fingerprint Sensor

The USB adapter is labeled, but the fingerprint sensor cables are not. However, the cables have a clear color, which we can identify and connect to the USB converter. We only need four of the cables (if your fingerprint sensor has more, you can ignore the remaining colors):

Red: Depending on the accepted voltage of the sensor (3.3V or 5V).

White: RXD

Green: TXD

Black: GND

If your sensor needs a higher voltage than 3.3V (and the maximum value is equal to or greater than 5V), you can connect the red cable to the 5V pin.

To check whether the cabling is correct and whether the sensor is detected, you can open your console and perform the following before and after connecting:

ls /dev/ttyUSB*

If no other serial devices are connected via USB, nothing should be displayed first and afterwards /dev/ttyUSB0. If the name differs (because, for example, other devices are connected), you have to adapt it accordingly in the following steps.

Installation of the Raspberry Pi Fingerprint Library

For some commands of the installation, root privileges are required. Therefore we start a terminal session and type the following, which executes all the following commands as root:

sudo bash

Now we add the necessary package sources (the author has also a few other exciting projects):

wget -O - http://apt.pm-codeworks.de/pm-codeworks.de.gpg | apt-key add -

wget http://apt.pm-codeworks.de/pm-codeworks.list -P /etc/apt/sources.list.d/

We then update the available packages and install the Python library:

apt-get update

apt-get install python-fingerprint --yes

If an error has occurred (in particular, that not all dependent packages have been installed), then execute the following:

apt-get -f install

To return to the normal shell (under the Pi user), type exit.

2.1.2 Python:

Python is a popular programming language. It was created in 1991 by Guido van Rossum.

It is used for:

web development (server-side),

software development,

mathematics,

system scripting.

What can Python do?

Python can be used on a server to create web applications.

Python can be used alongside software to create workflows.

Python can connect to database systems. It can also read and modify files.

Python can be used to handle big data and perform complex mathematics.

Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).

Python has a simple syntax similar to the English language.

Python has syntax that allows developers to write programs with fewer lines than some other programming languages.

Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

Python can be treated in a procedural way, an object-orientated way or a functional way.

Variables in Python:

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume). Rules for Python variables:

A variable name must start with a letter or the underscore character

A variable name cannot start with a number

A variable name can only contain alpha-numeric characters and underscores

Variable names are case-sensitive (age, Age and AGE are three different variables)

Python Operators

Operators are used to perform operations on variables and values.

Python divides the operators in the following groups:

Arithmetic operators

Assignment operators

Comparison operators

Logical operators

Identity operators

Membership operators

Bitwise operators

Functions in Python:

A function is a set of statements that take inputs, do some specific computation and produces output. The idea is to put some commonly or repeatedly done task together and make a function, so that instead of writing the same code again and again for different inputs, we can call the function. Python provides built-in functions like print(), etc. but we can also create your own functions. These functions are called user-defined functions.

Pass by reference or Pass by value:

One important thing to note is, in Python every variable name is a reference. When we pass a variable to a function, a new reference to the object is created. Parameter passing in Python is same as reference passing in Java.

Default arguments:

A default argument is a parameter that assumes a default value if a value is not provided in the function call for that argument.

Keyword arguments:

The idea is to allow caller to specify argument name with values so that caller does not need to remember order of parameters.

Variable length arguments:

We can have both normal and keyword variable number of arguments. Please see this for details.

Anonymous functions: In Python, anonymous function means that a function is without a name. As we already know that def keyword is used to define the normal functions and the lambda keyword is used to create anonymous functions.

Python – Loops

while loop

Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.

for loop

Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.

nested loops

You can use one or more loop inside any another while, for or do..while loop.

Loop Control Statements

break statement

Terminates the loop statement and transfers execution to the statement immediately following the loop.

continue statement

Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

pass statement

The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

Python - Object Oriented:

Class − A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members and methods, accessed via dot notation.

Class variable − A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.

Data member − A class variable or instance variable that holds data associated with a class and its objects.

Function overloading − The assignment of more than one behavior to a particular function. The operation performed varies by the types of objects or arguments involved.

Instance variable − A variable that is defined inside a method and belongs only to the current instance of a class.

Inheritance − The transfer of the characteristics of a class to other classes that are derived from it.

Instance − An individual object of a certain class. An object obj that belongs to a class Circle, for example, is an instance of the class Circle.

Instantiation − The creation of an instance of a class.

Method − A special kind of function that is defined in a class definition.

Object − A unique instance of a data structure that's defined by its class. An object comprises both data members and methods.

Operator overloading − The assignment of more than one function to a particular operator.

Features of Python:

Python is an object-oriented, high level language, interpreted, dynamic and multipurpose programming language. There are a lot of features provided by python programming language. Some features of python are given below

High Level Language

Easy to use

Expressive Language

Interpreted

Platform Independent

Open Source

Object-Oriented language

Huge Standard Library

GUI Programming

Integrated

Advantages of Python:

Extensive Support Libraries

Integration Feature

Improved Programmer's Productivity

Productivity

Disadvantages of Python:

Difficulty in Using Other Languages

Weak in Mobile Computing

Gets Slow in Speed

Run-time Errors

2.2 Point Based Programs

2.2.1MySQL

MySQL is an open source relational database management system based on Structured Query Language.

Its name is a combination of "My" the name of co-founder MichaelWidenius's daughter and "SQL", the abbreviation for Structured Query Language.

MySQL runs on virtually all platforms, including Linux, UNIX, and Windows. Although it can be used in a wide range of applications, MySQL is most often associated with web-based applications and online publishing and is an important component of an open source enterprise stack called LAMP.

LAMP is a Web development platform that uses Linux as the operating system, Apache as the web server, MySQL as the relational database management system and PHP as the object-oriented scripting language.

MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality.

MySQL Storage Engines:

MySQL storage engines – a storage engine (or database engine) is the underlying software component that a database management system (DBMS) uses to create, read, update and delete (CRUD) data from a database.

Most database management systems include their own application programming interface (API) that allows the user to interact with their underlying engine without going through the user interface of the DBMS.

MySQL storage engines include:

Aria –

Its storage engine for the MariaDB and MySQL relational database management systems.

Its goal is to make a crash-safe alternative to MyISAM. It is not yet transactional but plans to add proper support for database transactions at some point in the future.

The long-term goal is for Aria to be the default transactional and non-transactional storage engine for MariaDB.

Berkeley DB –

Its software library that provides a high-performance embedded database for key/value data.

Berkeley DB is written in C with API bindings for C++, C#, PHP, Java, Perl, Python, Ruby, Tcl, Smalltalk, and many other programming languages.

Prior to v5.1, MySQL included a BDB data storage backend.

BlitzDB

CassandraSE

CONNECT (storage engine)

Falcon –

It was a transactional storage engine being developed for the MySQL relational database management system.

Development was stopped after Oracle purchased MySQL.

It was based on the Netfrastructure database engine. Falcon was designed to take advantage of Sun's zfs file system.

FederatedX

InfiniDB

It is scalable, software-only columnar database management system for analytic applications.

However, on 1 October 2014 InfiniDB ceased operations and filed for bankruptcy protection in US Bankruptcy Court in the Eastern District of Texas.

Existing customers may be able to receive support from other companies, notably MariaDB.

InnoDB –

Its storage engine for MySQL. MySQL 5.5 and later use it by default. It provides the standard ACID-compliant transaction features, along with foreign key support (Declarative Referential Integrity).

mroonga

MyISAM –

It is default storage engine for the MySQL relational database management system versions prior to 5.5.

It is based on the older ISAM code but has many useful extensions.

MySQL Archive –

This analytic storage engine can be used to create a table that is “archive” only. Data cannot be deleted from this table, only added.

MySQL Cluster –

This technology providing shared-nothing clustering and auto-sharing for the MySQL database management system.

It is designed to provide high availability and high throughput with low latency, while allowing for near linear scalability.

MySQL Cluster is implemented through the NDB or NDBCLUSTER storage engine for MySQL ("NDB" stands for Network Database).

MySQL Federated –

It allows a user to create a table that is a local representation of a foreign (remote) table. It utilizes the MySQL client library API as a data transport, treating the remote data source the same way other storage engines treat local data sources whether they be MYD files (MyISAM), memory (Cluster, Heap), or tablespace (InnoDB).

NDB Cluster –

It is a storage engine for storing tables of rows. NDB Cluster can concurrently support access from many types of API processes including from a MySQL server, Memcached, JavaScript/Node.JS, Java, JPA and HTTP/REST.

All API processes can operate on the same tables and data stored in the NDB Cluster.

sequence

SphinxSE

TokuDB

It is open source, high-performance storage engine for MySQL and MariaDB.

It achieves this by using a Fractal tree index.

It is a scalable, ACID and MVCC compliant storage engine that provides indexing-based query improvements, offers online schema modifications, and reduces slave lag for both hard disk drives and flash memory.

WiredTiger

XtraDB –

It is storage engine for the MariaDB and Percona Server databases, and is intended as a drop-in replacement to InnoDB, which is one of the default engines available on the MySQL database.

Applications of using MySQL Database:

Drupal

Joomla

MODx

MyBB

phpBB

TYPO3

WordPress

Features of MySQL:

MySQL is offered under two different editions: the open source MySQL Community Server and the proprietary Enterprise Server.

MySQL Enterprise Server is differentiated by a series of proprietary extensions which install as server plugins, but otherwise shares the version numbering system and is built from the same code base.

Major features as available in MySQL:

A broad subset of ANSI SQL 99, as well as extensions

Cross-platform support

Stored procedures, using a procedural language that closely adheres to SQL/PSM

Triggers

Cursors

Updatable views

Online DDL when using the InnoDB Storage Engine.

Information schema

Performance Schema that collects and aggregates statistics about server execution and query performance for monitoring purposes.

A set of SQL Mode options to control runtime behavior, including a strict mode to better adhere to SQL standards.

X/Open XA distributed transaction processing (DTP) support; two phase commit as part of this, using the default InnoDB storage engine

Transactions with savepoints when using the default InnoDB Storage Engine. The NDB Cluster Storage Engine also supports transactions.

ACID compliance when using InnoDB and NDB Cluster Storage Engines.

SSL support

Query caching

Sub-SELECTs (i.e. nested SELECTs)

Built-in replication support (i.e., master-master replication and master-slave replication) with one master per slave, many slaves per master.

Multi-master replication is provided in MySQL Cluster, and multi-master support can be added to unclustered configurations using Galera Cluster.

Full-text indexing and searching.

Embedded database library.

Unicode support

Partitioned tables with pruning of partitions in optimizer

Shared-nothing clustering through MySQL Cluster

Multiple storage engines, allowing one to choose the one that is most effective for each table in the application.

Native storage engines InnoDB, MyISAM, Merge, Memory (heap), Federated, Archive, CSV, Blackhole, NDB Cluster.

Commit grouping, gathering multiple transactions from multiple connections together to increase the number of commits per second.

The developers release minor updates of the MySQL Server approximately every two months. The sources can be obtained from MySQL's website or from MySQL's Git Hub repository, both under the GPL license.

Advantages of MySQL:

It's Easy To Use

Support Is Readily Available Whenever Necessary

It's Open-Source

It's Incredibly Inexpensive

It's An Industry Standard (And Still Extremely Popular)

2.2.2OpenCV

OpenCV is the most popular library for computer vision. Originally written in C/C++, it now provides bindings for Python.

OpenCV uses machine learning algorithms to search for faces within a picture. Because faces are so complicated, there isn't one simple test that will tell you if it found a face or not. Instead, there are thousands of small patterns and features that must be matched. The algorithms break the task of identifying the face into thousands of smaller, bite-sized tasks, each of which is easy to solve. These tasks are also called classifiers.

For something like a face, you might have 6,000 or more classifiers, all of which must match for a face to be detected (within error limits, of course). But therein lies the problem: for face detection, the algorithm starts at the top left of a picture and moves down across small blocks of data, looking at each block, constantly asking, "Is this a face? … Is this a face? … Is this a face?" Since there are 6,000 or more tests per block, you might have millions of calculations to do, which will grind your computer to a halt.

To get around this, OpenCV uses cascades. What's a cascade? The best answer can be found in the dictionary: "a waterfall or series of waterfalls." Like a series of waterfalls, the OpenCV cascade breaks the problem of detecting faces into multiple stages. For each block, it does a very rough and quick test. If that passes, it does a slightly more detailed test, and so on. The algorithm may have 30 to 50 of these stages or cascades, and it will only detect a face if all stages pass.

The advantage is that the majority of the picture will return a negative during the first few stages, which means the algorithm won't waste time testing all 6,000 features on it. Instead of taking hours, face detection can now be done in real time.

Cascades in Practice

Though the theory may sound complicated, in practice it is quite easy. The cascades themselves are just a bunch of XML files that contain OpenCV data used to detect objects. You initialize your code with the cascade you want, and then it does the work for you.

Since face detection is such a common case, OpenCV comes with a number of built-in cascades for detecting everything from faces to eyes to hands to legs. There are even cascades for non-human things. For example, if you run a banana shop and want to track people stealing bananas, this guy has built one for that

Installing OpenCV:

To start, open up your command line and update the apt-get package manager to refresh and upgrade and pre-installed packages/libraries:

$ sudo apt-get update

$ sudo apt-get upgrade

Install some developer tools:

$ sudo apt-get install build essential cmake pkg-config

The pkg-config package is already installed on your system, but be sure to include it in the above apt-get command just in case. The cmake program is used to automatically configure our OpenCV build.

OpenCV is an image processing and computer vision library. Therefore, OpenCV needs to be able to load various image file formats from disk such as JPEG, PNG, TIFF, etc. In order to load these images from disk, OpenCV actually calls other image I/O libraries that actually facilitate the loading and decoding process. We install the necessary ones below:

$ sudo apt-get install libjpeg8-dev libtiff5-dev libjasper-dev libpng5-dev

The following commands to install packages used to process video streams and access frames from cameras:

$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv41-dev

$ sudo apt-get install libxvidcore-dev libx264-dev

OpenCV ships out-of-the-box with a very limited set of GUI tools. These GUI tools allow us to display an image to our screen, wait for/record key presses, track mouse events, and create simple GUI elements such as sliders and track bars. Again, you shouldn't expect to be building full-fledged GUI applications with OpenCV — these are just simple tools that allow you to debug your code and build very simple applications.

Internally, the name of the module that handles OpenCV GUI operations is high GUI. The high GUI module relies on the GTK library, which you should install using the following command:

$ sudo apt-get install libgtk-3-dev

Now we can install the Python development headers and libraries for both python 2.7 and python 3.6.

$ sudo apt-get install python2.7-dev python3.6-dev

Download the OpenCV source

Download the OpenCV 3.1.0, which we download a .zip of and unarchive using the following commands:

$ cd ~

$ wget –O opencv.zip http://github.com/Itseez/opencv/archive/3.1.0.zip

$unzip opencv.zip

However, we're not done downloading source code yet — we also need to opencvcontrib repository as well:

$ wget –O opencv.zip http://github.com/Itseez/opencv/archive/3.1.0.zip

$unzip opencv.zip

Setup your Python environment — Python 2.7 or Python 3

We are now ready to start configuring our Python development environment for the build. The first step is to install pip, a Python package manager:

$ cd ~

$ wget http://bootstrap.pypa.io/get-pip.py

$ sudo python get-pip.py

Both virtualenv and virtualenvwrapper. These Python packages allow you to create separate,independent Python environments for each project that you are working on. The following commands are used to create the both:

$ sudo pip install virtualenvvirtualenvwrapper

$ sudo rm -rf ~/get-pip.py ~/.cache/pip

Once we have virtualenv andvirtualenvwrapper installed, we need to update our ~/.bashrc file to include the following lines at the bottom of the file:

#set up virtualenv and virtualenvwrapper

export WORKON_HOME=$HOME/.virtualenvs

source usr/local/bin/virtualenvwrapper.sh

The ~/.bashrc file is simply a shell script that Bash runs whenever you launch a new terminal. You normally use this file to set various configurations. In this case, we are setting an environment variable called WORKON_HOME to point to the directory where our Python virtual environments live. We then load any necessary configurations from virtualenvwrapper .

To update your ~/.bashrc file simply use a standard text editor. I would recommend using nano , vim , or emacs . You can also use graphical editors as well, but if you're just getting started, nano is likely the easiest to operate.

After editing our ~/.bashrc file, we need to reload the changes:

$ source ~/.bashrc

Creating your Python virtual environment

If you decide to use Python 2.7, use the following command to create a Python 2.7 virtual environment:

$ mkvirtualenv cv –p python2.

Otherwise, use this command to create a Python 3 virtual environment:

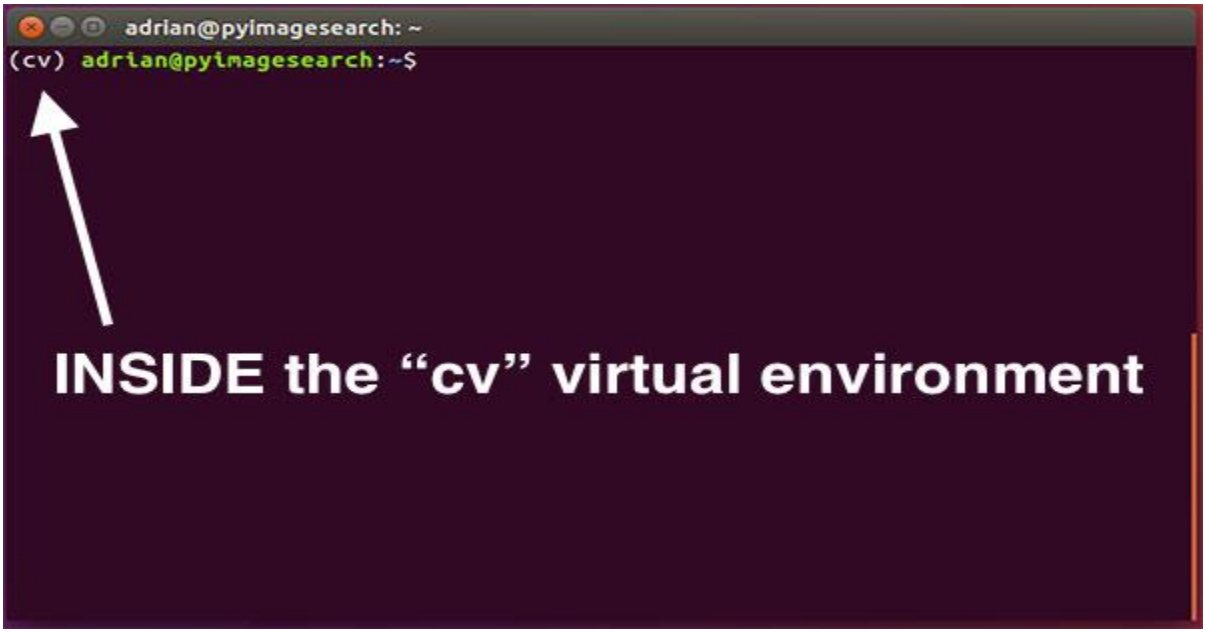
$mkvirtualenv cv –p python3

Verifying that you are in the "cv" virtual environment

If you ever reboot your ubuntu system; log out and log back in; or open up a new terminal, you'll need to use the workon command to re-access your cv virtual environment. An example of the workon command follows:

$ workon cv

That command shows the following output:



Install NumPy into your Python virtual environment

The final step before we compile OpenCV is to install NumPy, a Python package used for numerical processing. To install NumPy, ensure you are in the cv virtual environment. From there execute the following command:

$ pip install numpy

Now, we can install python for opencv:

$ sudo pip install opencv-python

Install the dlib and imutils packages:

$ sudo pip install dlib

$ sudo pip install imutils

Finally install face_recognition package

$ sudo pip install face_recognition

CHAPTER - 3

SYSTEM ANALYSIS

SYSTEM ANALYSIS

3.1 Existing System

In this existing system they are using password-based door lock system which makes the user memorize password all the time.

So to overcome this drawback we designing a Face recognition and biometric based door lock system using Raspberry pi.

3.2 Disadvantages

It provides less security.

Every time password is required for unlocking.

3.3 Proposed System

The proposed Face recognition and biometric door lock security system has been developed to prevent robbery in highly secure areas like home environment with lesser power consumption and more reliable standalone security device for both intruder detection and for door security.

This system is powered by raspberry pi circuit.

3.4 Advantages

High reliability.

It provides enough flexibility to suit the requirements.

More secure due to face detection.

CHAPTER - 4

REQUIREMENTS

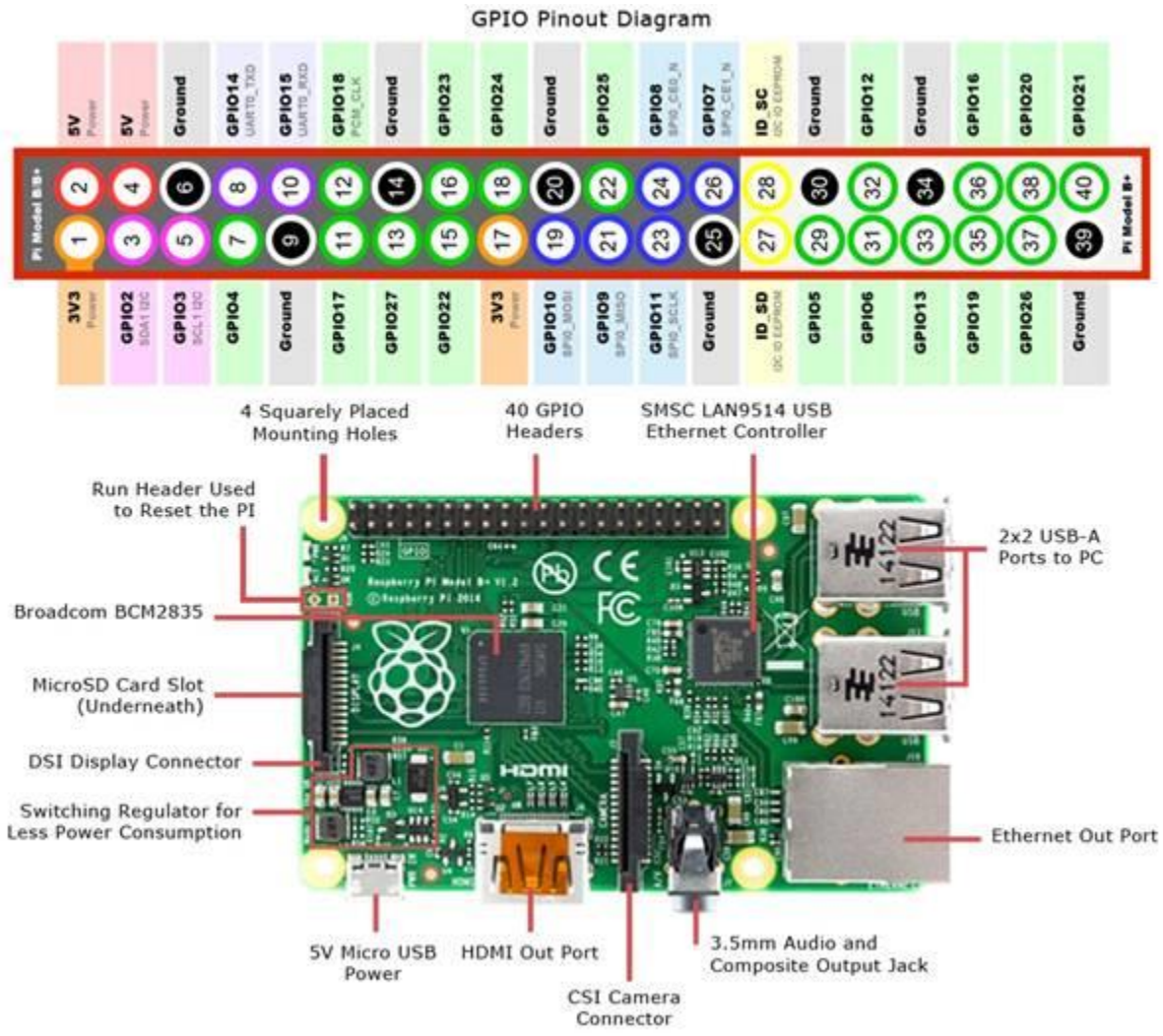
REQUIREMENTS

Raspberry Pi

Raspberry Pi Hardware Specifications

The raspberry pi board comprises a program memory (RAM), processor and graphics chip, CPU, GPU, Ethernet port, GPIO pins, X bee socket, UART, power source connector. And various interfaces for other external devices. It also requires mass storage, for that we use an SD flash memory card. So that raspberry pi board will boot from this SD card similarly as a PC boots up into windows from its hard disk.

Essential hardware specifications of raspberry pi board mainly include SD card containing Linux OS, US keyboard, monitor, power supply and video cable. Optional hardware specifications include USB mouse, powered USB hub, case, internet connection, the Model A or B: USB WiFi adaptor is used and internet connection to Model B is LAN cable.



Hardware Specifications of Raspberry pi

Memory

The raspberry pi model Aboard is designed with 256MB of SDRAM and model B is designed with 51MB.Raspberry pi is a small size PC compare with other PCs. The normal PCs RAM memory is available in gigabytes. But in raspberry pi board, the RAM memory is available more than 256MB or 512MB

CPU (Central Processing Unit)

The Central processing unit is the brain of the raspberry pi board and that is responsible for carrying out the instructions of the computer through logical and mathematical operations. The raspberry pi uses ARM11 series processor, which has joined the ranks of the Samsung galaxy phone.

GPU (Graphics Processing Unit)

The GPU is a specialized chip in the raspberry pi board and that is designed to speed up the operation of image calculations. This board designed with a Broadcom video core IV and it supports OpenGL

Ethernet Port

The Ethernet port of the raspberry pi is the main gateway for communicating with additional devices. The raspberry pi Ethernet port is used to plug your home router to access the internet.

GPIO Pins

The General purpose input & output pins are used in the raspberry pi to associate with the other electronic boards. These pins can accept input & output commands based on programming raspberry pi. The raspberry pi affords digital GPIO pins. These pins are used to connect other electronic components. For example, you can connect it to the temperature sensor to transmit digital data.

X Bee Socket

The X Bee socket is used in raspberry pi board for the wireless communication purpose.

Power Source Connector

The power source cable is a small switch, which is placed on side of the shield. The main purpose of the power source connector is to enable an external power source.

UART

The Universal Asynchronous Receiver/ Transmitter is a serial input & output port. That can be used to transfer the serial data in the form of text and it is useful for converting the debugging code.

Display

The connection options of the raspberry pi board are two types such as HDMI and Composite. Many LCD and HD TV monitors can be attached using an HDMI male cable and with a low-cost adaptor. The versions of HDMI are 1.3 and 1.4 are supported and 1.4 version cable is recommended. The O/Ps of the Raspberry Pi audio and video through HMDI, but does not support HDMI I/p. Older TVs can be connected using composite video. When using a composite video connection, audio is available from the 3.5mm jack socket and can be sent to your TV. To send audio to your TV, you need a cable which adjusts from 3.5mm to double RCA connectors.

Model A Raspberry Pi Board

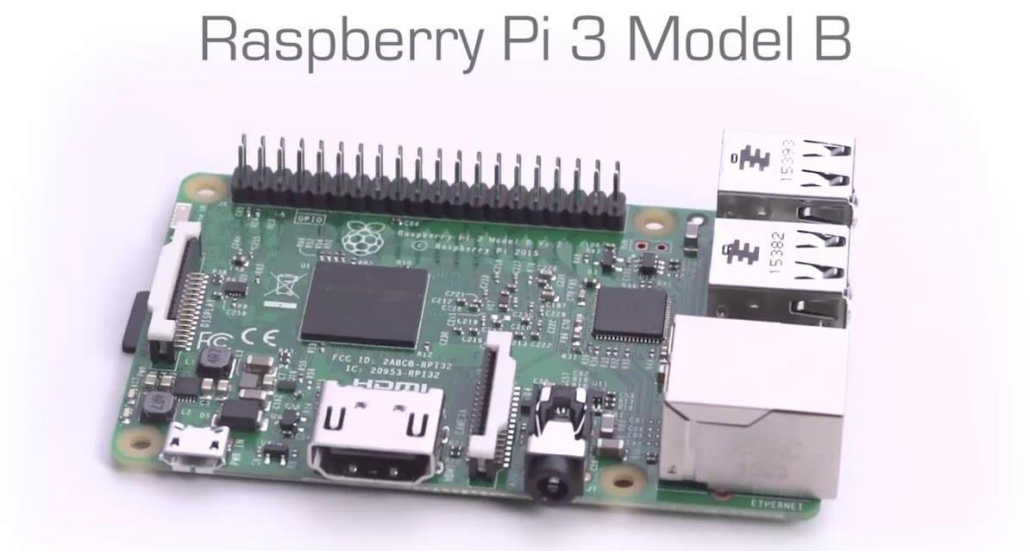
The Raspberry Pi board is a Broad com (BCM2835) SOC (system on chip) board. It comes equipped with an ARM1176JZF-S core CPU, 256 MB of SDRAM and 700 MHZ,. The raspberry pi USB 2.0 ports use only external data connectivity options. The board draws its power from a micro USB adapter, with min range of 2. Watts (500 MA). The graphics, specialized chip is designed to speed up the operation of image calculations. This is in built with Broadcom video core IV cable, that is useful if you want to run a game and video through your raspberry pi.

Features of Raspberry PI Model A

- The Model A raspberry pi features mainly includes
- 256 MB SDRAM memory
- Single 2.0 USB connector
- Dual Core Video Core IV Multimedia coprocessor
- HDMI (rev 1.3 & 1.4) Composite RCA (PAL and NTSC) Video Out
- 3.5 MM Jack, HDMI, Audio Out
- SD, MMC, SDIO Card slot on board storage
- Linux Operating system
- Broadcom BCM2835 SoC full HD multimedia processor
- 8.6cm*5.4cm*1.5cm dimensions

Model B Raspberry pi Board

The Raspberry Pi is a Broadcom BCM2835 SOC (system on chip board). It comes equipped with a 700 MHz, 512 MB of SDRAM and ARM1176JZF-S core CPU. The USB 2.0 port of the raspberry pi boars uses only external data connectivity options. The Ethernet in the raspberry pi is the main gateway to interconnect with other devices and the internet in model B. This draws its power from a micro USB adapter, with a minimum range of 2.5 watts (500 MA). The graphics, specialized chip is designed to speed up the manipulation of image calculations. This is in built with Broadcom video core IV cable, that is useful if you want to run a game and video through your raspberry pi.



Model B Raspberry pi Board

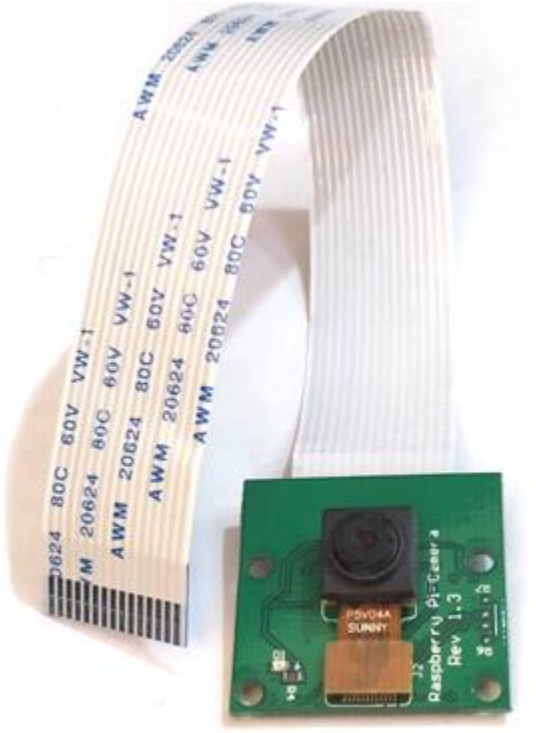
Features of Raspberry PI Model B

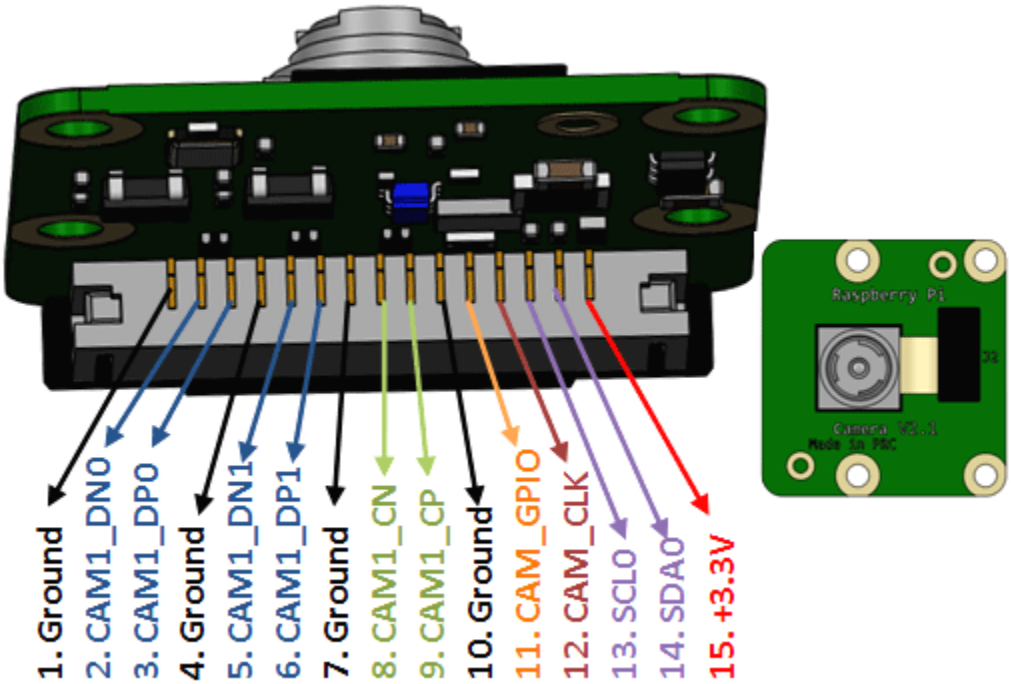
- 512 MB SDRAM memory
- Broadcom BCM2835 SoC full high definition multimedia processor
- Dual Core Video Core IV Multimedia coprocessor
- Single 2.0 USB connector
- HDMI (rev 1.3 and 1.4) Composite RCA (PAL & NTSC) Video Out
- 3.5 MM Jack, HDMI Audio Out
- MMC, SD, SDIO Card slot on board storage
- Linux Operating system
- Dimensions are 8.6cm*5.4cm*1.7cm
- On board 10/100 Ethernet RJ45 jack

Applications of Raspberry Pi

The raspberry pi boards are used in many applications like Media streamer, Arcade machine, Tablet computer, Home automation, Carputer, Internet radio, Controlling robots, Cosmic Computer, Hunting for meteorites, Coffee and also in raspberry pi based projects.

Pi Camera Module





Pi Camera Module

Pi Camera Pinout

The **Pi camera module** is a portable light weight camera that supports Raspberry Pi. It communicates with Pi using the MIPI camera serial interface protocol. It is normally used in image processing, machine learning or in surveillance projects. It is commonly used in surveillance drones since the payload of camera is very less. Apart from these modules Pi can also use normal USB webcams that are used along with computer.

Pi Cam Features

- 5MP camera module without microphone for Raspberry pi
- Supports both Raspberry Pi Model A and Model B
- MIPI Camera serial interface
- Omni vision 5647 Camera Module
- Resolution: 2592 * 1944
- Supports: 1080p, 720p and 480p
- Light weight and portable (3g only)

Alternatives Camera module for Raspberry Pi

IR Night Vision camera, USB Camera, 8MP MIPI Camera, 5MP Wide angle camera with IR.

How to use Camera module with Pi

The **Pi camera module** when purchased comes along with a ribbon cable, this cable has to be connected to the **CSI** (Camera Serial Interface) **port of the Pi**. This port can be found near the HDMI port just connect the cable to it as shown below.

After interfacing the hardware, we have to configure the Pi to enable Camera. Use the command "**sudo rasconfig**" to open the configuration window. Then under interfacing options enable camera. Finally reboot the Pi and your camera module is ready to use. Then, you can make the Pi to take photos or record videos using simple python scripts.

Applications

- Surveillance projects
- Time-lapse video recording
- Image processing
- Machine learning
- Robotics

PCB And Breadboard

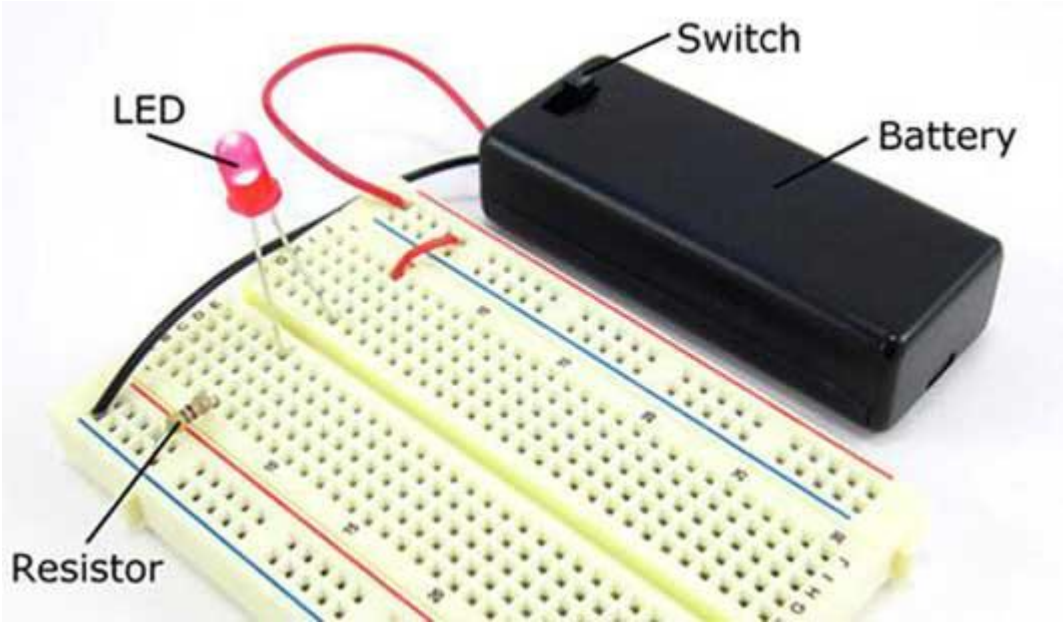
PCB prototype is an important process to your project's development, which is a trial manufacturing for the printed circuit board before mass production, electronic engineer is designing the circuit and that is mainly in doing small volume trail manufacturing for PCB manufacturer after designing the circuit and finishing PCB layout for the electronic engineer. However, there is no limit for PCB manufacturing quantity, generally speaking, PCB prototype is the best practice method to verify the quality of a design before proceeding.

PCB is the support body of electronic components and the carrier of electrical connection of electronic components, which mainly plays the role of support and interconnection. What's more, the PCB will be your finished product. Now printed circuit board (PCB) is widely used in various electronic and related products.

What is a breadboard?

Breadboard, also known as protoboard, which is the bread-and-butter of DIY electronics. A breadboard is a simple device designed to let you create circuits without the need for soldering. It is a rectangular plastic board with a bunch of tiny holes, and the holes can make it easy to insert electronic components to prototype an electronic circuit, such as this one with a battery, switch, resistor, and an LED (light-emitting diode). however, breadboard will have a less permanent compared to a printed circuit board, so you can remove and change the breadboards if needed as they have sockets that you push the components into. It can't be denied that a

breadboard will be more for Experimenting designing and testing circuit connections before making it permanent.



Where does the name "breadboard" come from?

You might be wondering what any of this has to do with bread. The term breadboard comes from the early days of electronics, when people would literally drive nails or screws into wooden boards on which they cut bread in order to connect their circuits. Luckily, since you probably do not want to ruin all your cutting boards for the sake of an electronics project, today there are better options.

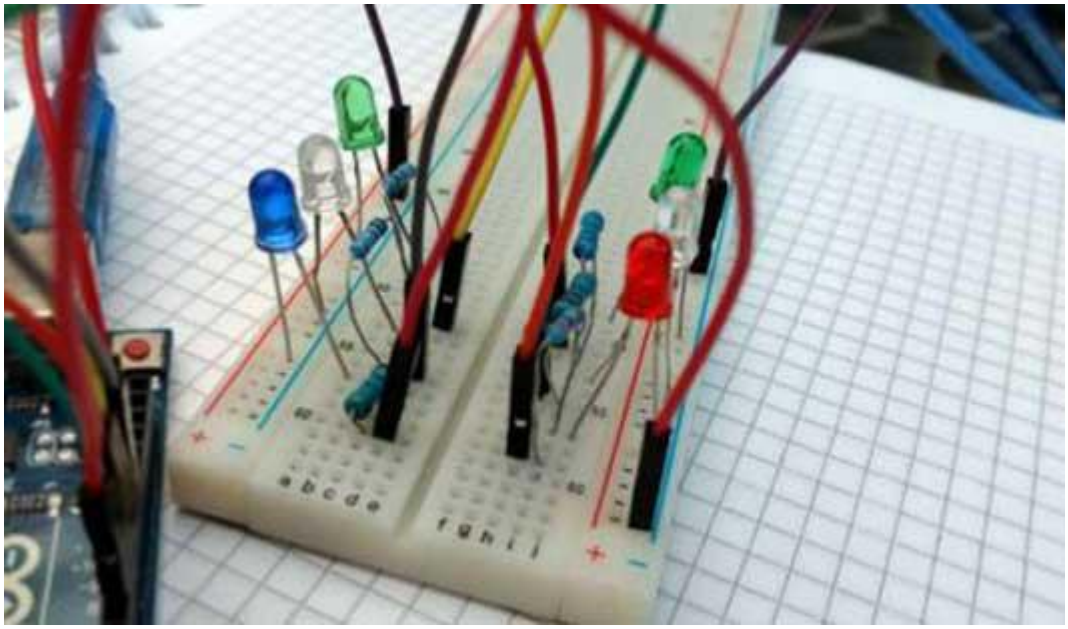
How does a breadboard work?

You need to make a breadboard before creating the permanent circuit board because it can remove and change components on a breadboard. What's more, you will draw out the schematics and connect the wires accordingly.

As you know, the centre of PCB is the prototyping region, which is made up of two rows of five holes. There is a channel between the two rows, you can place a chip with pins on either side so that prevent them from connecting together. Also, you can seek out power busses (either one or two) on the side of the breadboard for working power and grounding.
In fact, designing breadboards are used for integrated circuit (ICs). you can place an IC chip over the channel, which you can access to the pins on either side of the existing chip. What's more, connecting a resistor to the power bus into the channel, at the same time taking an LED from the ground bus to create the whole circuit.

Breadboard and Printed circuit board

On one hand, a breadboard is usually used as the first step before creating a printed circuit board. You can change and move circuits that are otherwise permanent on a PCB with a breadboard. On the other hand, breadboards are used for design and investigation, while the boards are for your finished products.



The Advantages of breadboard

- You can rapidly change connections and test various plans in a development phase.
- It's easy and fast to assemble as there are no permanent solder connections.
- You can also change various components such as the capacitor or resistor value.
- You can add an ammeter anywhere with shifting wires (breaking into) any branch of your circuit. What's more, the current measurement on PCBs require you to break tracks or add extra resistors in your design.

The advantages of a printed circuit board

- The board is permanent to have an electronic device worked.
- PCB has a better current carrying capacity comparing to a breadboard, you can make your traces wider to take more current so that work well.
- You can add terminals to your printed circuit board for external connections.
- You can mount heat-sinks to the board so that have them rigid.
- There is widely used in electronic devices.

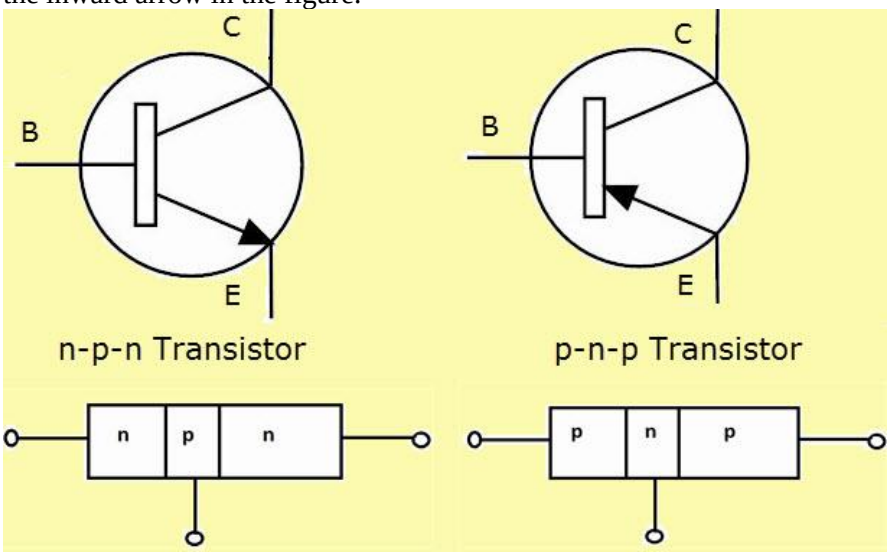
Transistors

Transistor is an active component and that is establishing in all over electronic circuits. They are used as amplifiers and switching apparatus. As the amplifiers, they are used in high and low level, frequency stages, oscillators, modulators, detectors and in any circuit need to perform a function. In digital circuits they are used as switches. There are a huge number of manufacturers approximately the world who produces semiconductors (transistors are members of this family of apparatus), so there are exactly thousands of different types. There are low, medium and high power transistors, for functioning with high and low frequencies, for functioning with very high current and or high voltages. This article gives an overview of what is a transistor, different Types of transistors and its applications.

Transistor Symbol

A diagrammatic form of n-p-n and p-n-p transistor is exposed. In circuit is a connection drawn form is used. The arrow symbol defined the emitter current. In the n-p-n connection we identify electrons flow into the emitter. This means that conservative current flows out of the emitter as an indicated by the outgoing arrow.

Equally it can be seen that for p-n-p connection, the conservative current flows into the emitter as exposed by the inward arrow in the figure.

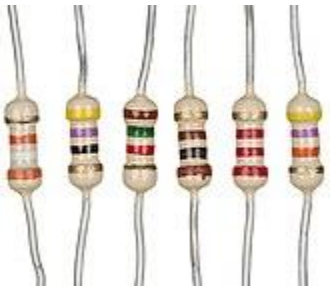


RESISTOR

A **resistor** is a passive two-terminal electrical component that implements electrical resistance as a circuit element. In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses. High-power resistors that can dissipate many watts of electrical power as heat, may be used as part of motor controls, in power distribution systems, or as test loads for generators. Fixed resistors have resistances that only change slightly with temperature, time or operating voltage. Variable resistors can be used to adjust circuit elements (such as a volume control or a lamp dimmer), or as sensing devices for heat, light, humidity, force, or chemical activity.

Resistors are common elements of electrical networks and electronic circuits and are ubiquitous in electronic equipment. Practical resistors as discrete components can be composed of various compounds and forms. Resistors are also implemented within integrated circuits.

The electrical function of a resistor is specified by its resistance: common commercial resistors are manufactured over a range of more than nine orders of magnitude. The nominal value of the resistance falls within the manufacturing tolerance, indicated on the component.



Electric door lock

Electronic door locks use something called actuators, which move the bolt of the lock into

place so that they door can open.

you simply have to trigger the actuator and your door will open.

Electric locks use magnets, solenoids, or motors to actuate the lock by either supplying or removing power.

Operating the lock can be as simple as using a switch, for example an apartment intercom door release, or as complex as a biometric based access control system.

An **electronic lock** (or **electric lock**) is a locking device which operates by means of electric current. Electric locks are sometimes stand-alone with an electronic control assembly mounted directly to the lock. Electric locks may be connected to an access control system, the advantages of which include: key control, where keys can be added and removed without re-keying the lock cylinder; fine access control, where time and place are factors; and transaction logging, where activity is recorded. Electronic locks can also be remotely monitored and controlled, both to lock and to unlock.

Biometrics

As biometrics become more and more prominent as a recognized means of positive identification, their use in security systems increases. Some electronic locks take advantage of technologies such as fingerprint scanning, retinal scanning, iris scanning and voice print identification to authenticate users.

RFID

Radio-frequency identification (RFID) is the use of an object (typically referred to as an "RFID tag") applied to or incorporated into a product, animal, or person for the purpose of identification and tracking using radio waves. Some tags can be read from several meters away and beyond the line of sight of the reader. This technology is also used in some modern electronic locks.

Display device

A **display device** is an output device for presentation of information in visual[1] or tactile form (the latter used for example in tactile electronic displays for blind people).[2] When the input information that is supplied has an electrical signal the display is called an *electronic display*.

Types of display devices

These are the technologies used to create the various displays in use today.

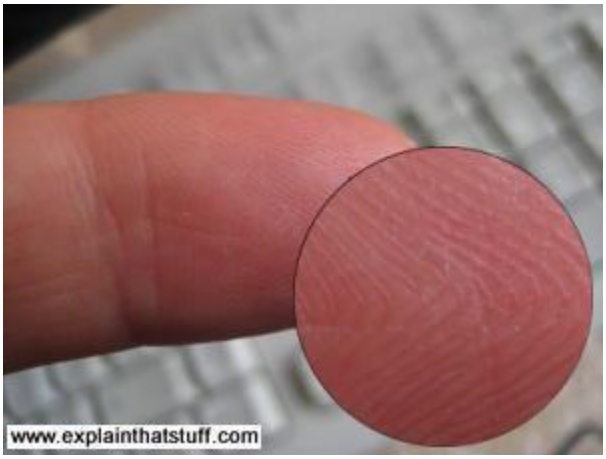
- Electroluminescent (ELD) display
- Liquid crystal display (LCD)
  - Light-emitting diode (LED) backlit LCD
  - Thin-film transistor (TFT) LCD
- Light-emitting diode (LED) display
  - OLED display
  - AMOLED display
- Plasma (PDP) display
- Quantum dot (QLED) display

Biometric fingerprint scanners

Nothing is completely secure. Locks can be picked, safes can be broken into, and online passwords can be guessed sooner or later. How, then, can we protect the things that we value? One way is to use **biometrics**—fingerprints, iris scans, retinal scans, face scans, and other personal information that is more difficult to forge.

Not so long ago, if you'd had your fingerprints taken, chances are you were being accused of a crime; now, it's innocent people who are turning to fingerprints to protect themselves. And you can find fingerprint scanners on everything from high-security buildings to ATM machines and even laptop computers. Let's take a closer look at how they work!

Why are fingerprints unique?



It's pretty obvious why we have fingerprints—the tiny **friction ridges** on the ends of our fingers and thumbs make it easier to grip things. By making our fingers rougher, these ridges increase the force of friction between our hands and the objects we hold, making it harder to drop things. You have fingerprints even before you're born. In fact, fingerprints are completely formed by the time you're seven months old in the womb. Unless you have accidents with your hands, your fingerprints remain the same throughout your life.

What makes fingerprints such a brilliant way of telling people apart is that they are virtually unique: fingerprints develop through an essentially random process according to the code in your DNA (the genetic recipe that tells your body how to develop). Because the environment in the womb also has an effect, even the prints of identical twins are slightly different. While it's *possible* that two people could be found who had identical fingerprints, the chances of this happening are so small as to be virtually negligible. In a criminal case, there are usually other pieces of forensic evidence that can be used with fingerprints to prove a person's guilt or innocence beyond reasonable doubt. Where fingerprints are being used to control access to something like a computer system, the chances of a random person having just the right fingerprint to gain entry are, generally speaking, too small to worry about—and much less the chance of someone guessing the right password or being able to break through a physical lock.

Enrollment and verification

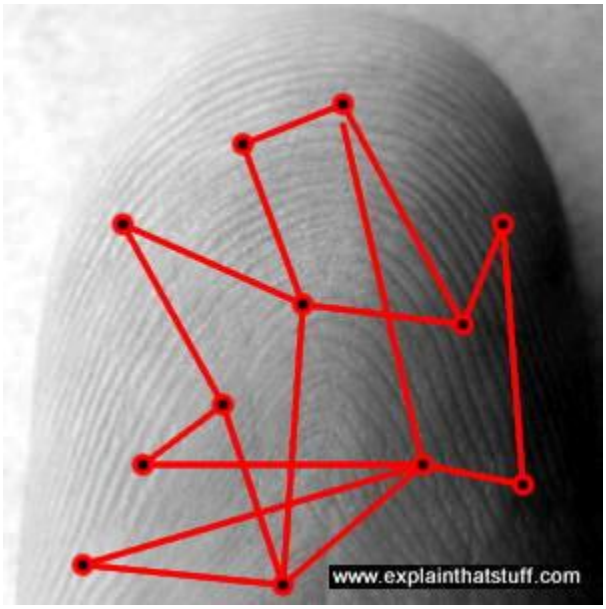
Suppose you're in charge of security for a large bank and you want to put a fingerprint scanning system on the main entry turnstile where your employees come in each morning. How exactly would it work?

There are two separate stages involved in using a system like this. First you have to go through a process called **enrollment**, where the system learns about all the people it will have to recognize each day. During enrollment, each person's fingerprints are scanned, analyzed, and then stored in a coded form on a secure database. Typically it takes less than a half second to store a person's prints and the system works for over 99 percent of typical users (the failure rate is higher for manual workers than for office workers).

Once enrolment is complete, the system is ready to use—and this is the second stage, known as **verification**. Anyone who wants to gain access has to put their finger on a scanner. The scanner takes their fingerprint, checks it against all the prints in the database stored during enrolment , and decides whether the person is entitled to gain access or not. Sophisticated fingerprint systems can verify and match up to 40,000 prints per second!

How fingerprints are stored and compared

When fingerprints were first used systematically for criminal investigation in 1900, by Sir Edward Henry of the Metropolitan Police in London, England, they were compared slowly and laboriously by hand. You took a fingerprint from a crime scene and another fingerprint from your suspect and simply compared them under a magnifying glass or microscope. Unfortunately, fingerprints taken under different conditions can often look quite different—the one from the crime scene is much more likely to be incomplete or smudged—and comparing them to prove that they are identical (or different) sometimes takes great skill. That's why forensic scientists (people who study evidence collected from crime scenes) developed a reliable system for matching fingerprints where they looked for between eight and sixteen distinct features. In the UK, two fingerprints need to match in all sixteen respects for the prints to be judged the same; in the United States, only eight features need to match.



When a computer checks your fingerprints, there obviously isn't a little person with a magnifying glass sitting inside, comparing your fingerprints with all the hundreds or thousands stored in the database! So how can a computer compare prints? During enrollment or verification, each print is analyzed for very specific features called **minutiae**, where the lines in your fingerprint terminate or split in two. The computer measures the distances and angles between these features—a bit like drawing lines between them—and then uses an algorithm (mathematical process) to turn this information into a unique numeric code. Comparing fingerprints is then simply a matter of comparing their unique codes. If the codes match, the prints match, and the person gains access.

How fingerprint scanners work

Having your fingerprints taken at a police station involves pressing your fingers onto an ink pad and then rolling your fingers onto paper to leave a clean impression on the page. Your prints are also stored on a computer database so the police can check if you've committed any known crimes or if you do so in future.

But when fingerprints are being used to control access to buildings and computer systems, more sophisticated methods have to be used: a computer has to scan the surface of your finger very quickly and then turn the scanned representation into a code it can check against its database. How does this happen?

There are two main ways of scanning fingers. An **optical scanner** works by shining a bright light over your fingerprint and taking what is effectively a digital photograph. If you've ever photocopied your hand, you'll know exactly how this works. Instead of producing a dirty black photocopy, the image feeds into a

computer scanner. The scanner uses a light-sensitive microchip (either a CCD, charge-coupled device, or a CMOS image sensor) to produce a digital image. The computer analzes the image automatically, selecting just the fingerprint, and then uses sophisticated pattern-matching software to turn it into a code.

Another type of scanner, known as a **capacitive scanner**, measures your finger electrically. When your finger rests on a surface, the ridges in your fingerprints touch the surface while the hollows between the ridges stand slightly clear of it. In other words, there are varying distances between each part of your finger and the surface below. A capacitive scanner builds up a picture of your fingerprnt by measuring these distances. Scanners like this are a bit like the touch screens on things like iPhones and iPads

Here's how the process works with a simple optical scanner:

1. A row of LEDs scans bright light onto the glass (or plastic) surface on which your finger is pressing (sometimes called the platen).
2. The quality of the image will vary according to how you're pressing, how clean or greasy your fingers are, how clean the scanning surface is, the light level in the room, and so on.
3. Reflected light bounces back from your finger, through the glass, onto a CCD or CMOS image sensor.
4. The longer this image-capture process takes, the brighter the image formed on the image sensor.
5. If the image is too bright, areas of the fingerprint (including important details) may be washed out completely—like an indoor digital photo where the flash is too close or too bright. If it's too dark, the whole image will look black and details will be invisible for the opposite reason.
6. An algorithm tests whether the image is too light or too dark; if so, an audible beep or LED indicator alerts the operator and we go back to step 1 to try again.
7. If the image is roughly acceptable, another algorithm tests the level of detail, typically by counting the number of ridges and making sure there are alternate light and dark areas (as you'd expect to find in a decent fingerprint image). If the image fails this test, we go back to step 1 and try again.
8. Providing the image passes these two tests, the scanner signals that the image is OK to the operator (again, either by beeping or with a different LED indicator). The image is stored as an acceptable scan in flash memory, ready to be transmitted (by USB cable, wireless, Bluetooth, or some similar method) to a "host" computer where it can be processed further. Typically, images captured this way are 512×512 pixels (the dimensions used by the FBI), and the standard image is 2.5cm (1 inch) square, 500 dots per inch, and 256 shades of gray.
9. The host computer can either store the image on a database (temporarily or indefinitely) or automatically compare it against one or many other fingerprints to find a match.

Fingerprint scanning is the most popular biometric technology (used in over half of all biometric security systems)—and it's easy to see why. We store more and more information on our computers and share it, online, in ever more risky ways. Much of the time, our bank information and personal details are protected by just the few hastily thought-out numbers in our passwords. Anyone can use your credit or debit card to get money from an ATM (automated teller machine or "cashpoint") if they know just four numbers!

In future, it will be much more common to have to confirm your identity with biometric information: either your fingerprint, a scan of the iris or retina in your eye, or a scan of your face. Some laptop computers and cell phones now use fingerprint scanning to make them more secure. Large banks, such as Bank of America and JPMorgan Chase, have introduced fingerprint authentication as part of the sign in process for their smartphone apps. Soon we could be seeing fingerprint scanners on ATMs, in airport security scanners, on checkouts in grocery stores, in electronic voting systems, and perhaps even replacing the keys in our (self-driving) automobiles!

Some people don't like the sound of a "Big Brother" society where you have to do everything with your fingerprints—and it's true that there are important issues of privacy. But humans have always used biometrics for personal identification: we tell one another apart chiefly by recognizing one another's faces and voices. Worry about the drawbacks, by all means, but don't forget the advantages too: your information should be much more secure from criminals—and you'll never again have the problem of losing your keys or forgetting your password!

Buzzer

A buzzer or beeper is an audio signalling device,[1] which may be mechanical, electromechanical, or piezoelectric (*piezo* for short). Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke.

Jump wire

A jump wire (also known as jumper wire, or jumper) is an electrical wire, or group of them in a cable, with a connector or pin at each end (or sometimes without them – simply "tinned"), which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.[1]

Individual jump wires are fitted by inserting their "end connectors" into the slots provided in a breadboard, the header connector of a circuit board, or a piece of test equipment.

Capacitor

A capacitor is a device that stores electrical energy in an electric field. It is a passive electronic component with two terminals.

The effect of a capacitor is known as capacitance. While some capacitance exists between any two electrical conductors in proximity in a circuit, a capacitor is a component designed to add capacitance to a circuit. The capacitor was originally known as a condenser or condensator. This name and its cognates are still widely used in many languages, but rarely in English, one notable exception being condenser microphones, also called capacitor microphones.

The physical form and construction of practical capacitors vary widely and many types of capacitor are in common use. Most capacitors contain at least two electrical conductors often in the form of metallic plates or surfaces separated by a dielectric medium. A conductor may be a foil, thin film, sintered bead of metal, or an electrolyte. The nonconducting dielectric acts to increase the capacitor's charge capacity. Materials commonly used as dielectrics include glass, ceramic, plastic film, paper, mica, air, and oxide layers. Capacitors are widely used as parts of electrical circuits in many common electrical devices. Unlike a resistor, an ideal

capacitor does not dissipate energy, although real-life capacitors do dissipate a small amount. (See Non-ideal behavior) When an electric potential, a voltage, is applied across the terminals of a capacitor, for example when a capacitor is connected across a battery, an electric field develops across the dielectric, causing a net positive charge to collect on one plate and net negative charge to collect on the other plate. No current actually flows through the dielectric. However, there is a flow of charge through the source circuit. If the condition is maintained sufficiently long, the current through the source circuit ceases. If a time-varying voltage is applied across the leads of the capacitor, the source experiences an ongoing current due to the charging and discharging cycles of the capacitor.

The earliest forms of capacitors were created in the 1740s, when European experimenters discovered that electric charge could be stored in water-filled glass jars that came to be known as Leyden jars.
In 1748, Benjamin Franklin connected a series of jars together to create what he called an "electrical battery", from their visual similarity to a battery of cannon, which became the standard English term electric battery. Today, capacitors are widely used in electronic circuits for blocking direct current while allowing alternating current to pass. In analog filter networks, they smooth the output of power supplies. In resonant circuits they tune radios to particular frequencies. In electric power transmission systems, they stabilize voltage and power flow.[2] The property of energy storage in capacitors was exploited as dynamic memory in early digital computers.

Connectors:

An electronic connector is an electro-mechanical device whose purpose is to quickly and easily disconnect or interrupt a circuit path.

Connectors come in a variety of sizes, shapes ,complexities and quality levels.

Cables:

Electrical cables are used to connect two or more devices ,enabling the transfer of electrical signals or power from one device to the other .

Cables are used for a wide range of purposes ,and each must be tailored for that purpose.

Cables are used extensively in electronic devices for power and signal circuits.

Display:

A display device is an output device for presentation of information in visual or tactile form (the latter used for example in tactile electronic displays for blind people).

When the input information that is supplied has an electrical signal the display is called an electronic display.

SOLENOID DOOR LOCK:

As a manufacturer of high-tech locking systems, KendrionKuhnke Automation has achieved a very reliable compact solenoid lock based on long-term experiences with a lot of different locking applications.

- Door locking technology and access control, such as fittings or turnstiles
- Aviation, such as overhead storage, galleys and life support
- Medical and analysis technology, such as centrifuges and disinfection equipment
- Devices and industrial equipment, such as combi steamer, steamer, ovens, autoclaves or washing machines

CHAPTER – 5

DESIGN

DESIGN

5.1 UML DIAGRAMS:

Introduction to UM:

Unified Modeling Language is one of the most exciting tools in the world of system development today. The UML enables system builders to create blue prints that capture their visions in a standard, easy to understand way and communicate them to others. The UML is the brainchild of Grady Booch, James Rambaugh and Ivar Jacobson.

Definition of UML:

"The UML is a language for visualizing, specifying, constructing and documenting of artifacts". These are the artifacts of a software-intensive system. The abbreviation for UML is Unified Modeling Language.

The Components of UML:

UML consists of number of graphical elements combine to form diagrams. The purpose of the diagrams is to present multiple views of the system and this set of multiple views is called a Model. A UML model of a system is something like a scale model of building. It is important to note that a UML model describes what a system is supposed to do. It doesn't tell how to implement the system. Some of the diagrams that help for the diagrammatic approach for the Object Oriented Software Engineering are

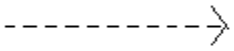
- Use Case Diagrams
- Class Diagrams
- Sequence Diagrams
- Collaboration Diagrams
- State Chart Diagrams
- Activity Diagrams
- Component
- Deployment Diagrams

Using the above mentioned diagrams we can show the entire system regarding the working of the system or the flow of control and sequence of flow the state of the system and the activities involved in the system.

5.2.2 Relationships in the UML

There are four kinds of relationships in the UML

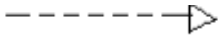
Dependency: It is a semantic relationship between two things in which a change to one thing may affect the semantic of the other thing .Graphically; decency is rendered as a dashed line, possibly directed and occasionally including a level. It is represented as follows



Association: It is a structural relationship that describes a set of links, a link being a connection among objects. Aggregation is a special kind of association, representing a structural relationship between a hole and its parts. It is represented as follows

Generalization: It is a specialization/generalization relationship in which objects of specialized element are substitutable for object of the generalized element. It is represented as follows

Realization: It is a semantic relationship between classifiers, wherein one classifier specifies a contract to carry out. It is represented as follows.



5.2 Use Case Diagram:

Use-case diagrams graphically represent system behavior (use cases). These diagrams present a high level view of how the system is used as viewed from an outsider's (actor's) perspective. A use-case diagram may contain all or some of the use cases of a system.

A use-case diagram can contain:

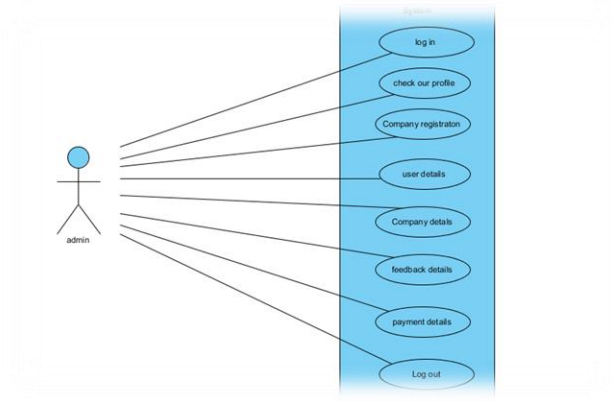
• Actors ("things" outside the system)

• Use cases (system boundaries identifying what the system should do)

• Interactions or relationships between actors and use cases in the system including the associations, dependencies, and generalizations.

Use-case diagrams can be used during analysis to capture the system requirements and to understand how the system should work. During the design phase, you can use use-case diagrams to specify the behavior of the system as implemented.

Fig no:5.1.1 use case diagram for ADMIN



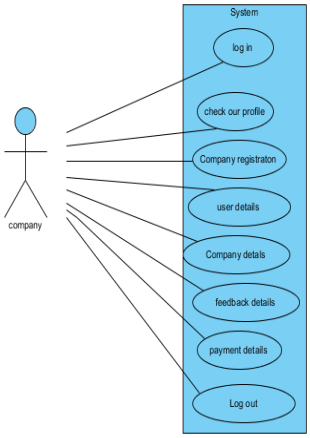
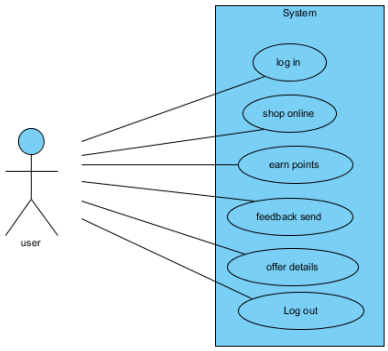


Fig no: 5.1.2 use case diagram for PROVIDER



5.3 Class Diagram:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes. The class diagram is the main building block in object-oriented modeling. They are being used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. The classes in a class diagram represent both the main objects and or interactions in the application and the

objects to be programmed. In the class diagram these classes are represented with boxes which contain three parts:

Fig no:5.1.2 class diagram for ADMIN

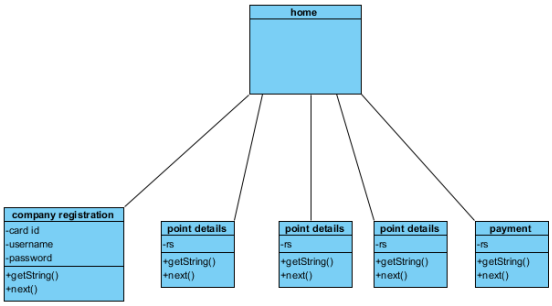


Fig no:5.1.2 class diagram for PROVIDER

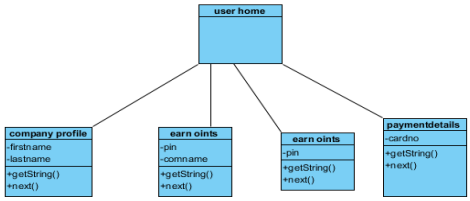


Fig no:5.1.2 class diagram for CUSTOMER

5.4 Sequence Diagram:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

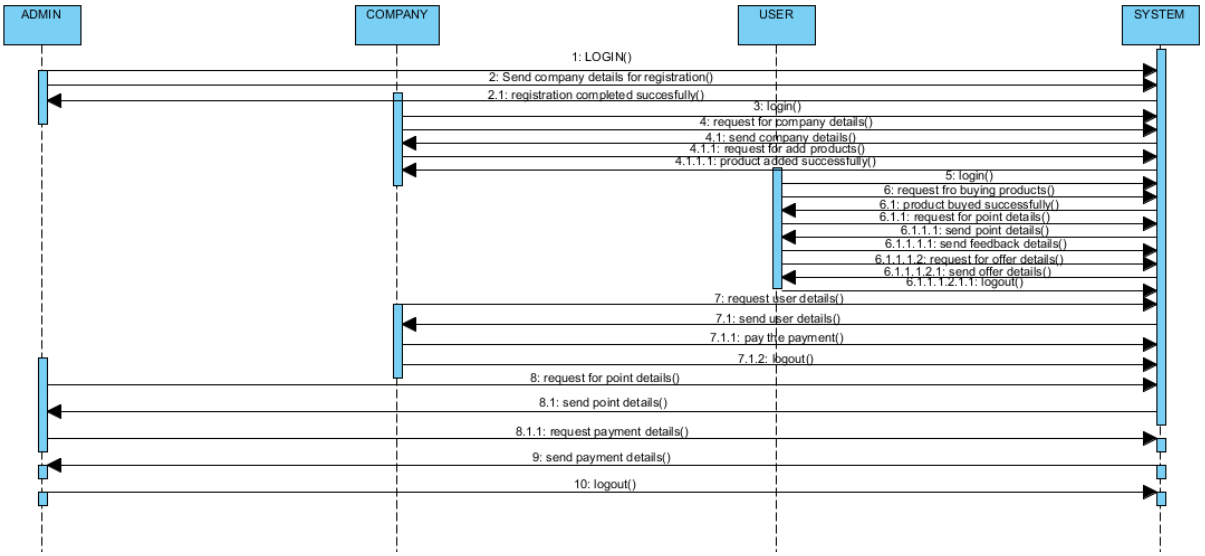


Fig no :5.1.3 Sequence Diagram

5.5 Activity diagram :

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

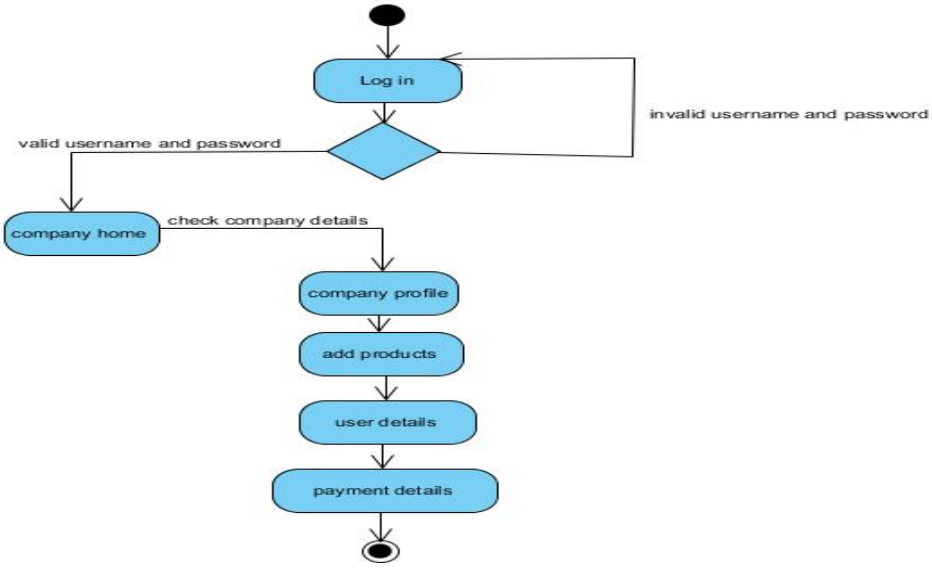


Fig no:5.1.4 Activity diagram for COMPANY

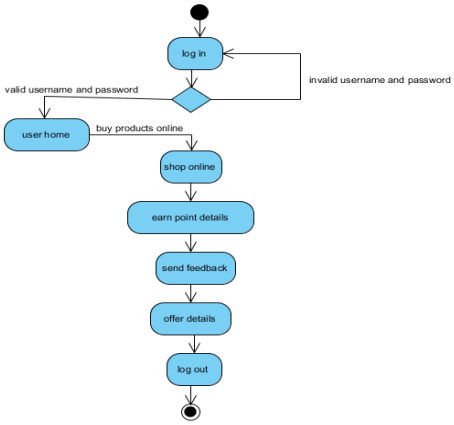


Fig no:5.1.4 Activity diagram for USER

5.6 State chart diagram:

A state chart diagram shows a state machine including simple states transactions and nested composite states. A state chart diagram describes the behavior of system

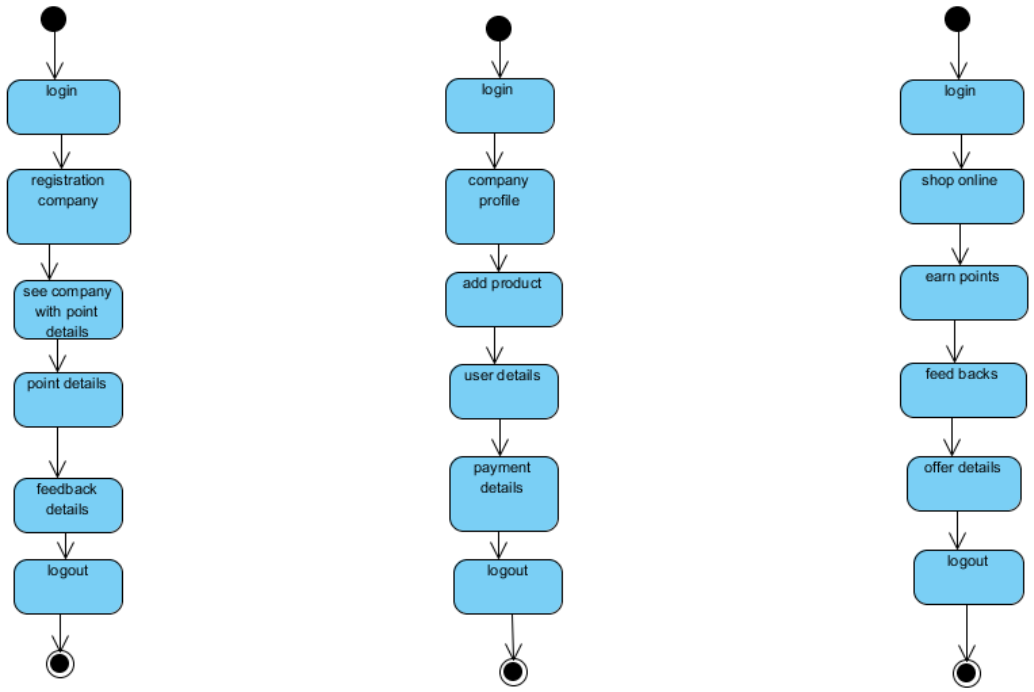


Fig no: 5.1.5.State chart diagram

CHAPTER 6

IMPLEMENTATION

IMPLEMETATION

SOURCE CODE:

```
import time
from pyfingerprint .pyfingerprint import
 PyFingerprint
import RPi.GPIO as gpio



RS =18
EN =23
D4 =24
D5 =25
D6 =8
D7 =7

enrol=5
delet=6
inc=13
dec=19
led=26

HIGH=1
LOW=0

gpio.setwarnings(False)
gpio.setmode(gpio.BCM)
gpio.setup(RS, gpio.OUT)
gpio.setup(EN, gpio.OUT)
```

```
gpio.setup(D4, gpio.OUT)
gpio.setup(D5, gpio.OUT)
gpio.setup(D6, gpio.OUT)
gpio.setup(D7, gpio.OUT)
gpio.setup(2,gpio.OUT)
gpio.setup(enrol, gpio.IN, pull_up_down=gpio.PUD_UP)
gpio.setup(delet, gpio.IN, pull_up_down=gpio.PUD_UP)
gpio.setup(inc, gpio.IN, pull_up_down=gpio.PUD_UP)
gpio.setup(dec, gpio.IN, pull_up_down=gpio.PUD_UP)
gpio.setup(led, gpio.OUT)
gpio.output(2,gpio.HIGH)
try:
    f = PyFingerprint('/dev/ttyUSB0', 57600, 0xFFFFFFFF, 0x00000000)

    if ( f.verifyPassword() == False ):
        raise ValueError('The given fingerprint sensor password is wrong!')

except Exception as e:
    print('Exception message: ' + str(e))
    exit(1)

def begin():
  lcdcmd(0x33)
  lcdcmd(0x32)
  lcdcmd(0x06)
  lcdcmd(0x0C)
  lcdcmd(0x28)
  lcdcmd(0x01)
  time.sleep(0.0005)
```

```
def lcdcmd(ch):
  gpio.output(RS, 0)
  gpio.output(D4, 0)
  gpio.output(D5, 0)
  gpio.output(D6, 0)
  gpio.output(D7, 0)
  if ch&0x10==0x10:
    gpio.output(D4, 1)
  if ch&0x20==0x20:
    gpio.output(D5, 1)
  if ch&0x40==0x40:
    gpio.output(D6, 1)
  if ch&0x80==0x80:
    gpio.output(D7, 1)
  gpio.output(EN, 1)
  time.sleep(0.005)
  gpio.output(EN, 0)
  # Low bits
  gpio.output(D4, 0)
  gpio.output(D5, 0)
  gpio.output(D6, 0)
  gpio.output(D7, 0)
  if ch&0x01==0x01:
    gpio.output(D4, 1)
  if ch&0x02==0x02:
    gpio.output(D5, 1)
  if ch&0x04==0x04:
    gpio.output(D6, 1)
```

```
  if ch&0x08==0x08:
    gpio.output(D7, 1)
  gpio.output(EN, 1)
  time.sleep(0.005)
  gpio.output(EN, 0)

def lcdwrite(ch):
  gpio.output(RS, 1)
  gpio.output(D4, 0)
  gpio.output(D5, 0)
  gpio.output(D6, 0)
  gpio.output(D7, 0)
  if ch&0x10==0x10:
    gpio.output(D4, 1)
  if ch&0x20==0x20:
    gpio.output(D5, 1)
  if ch&0x40==0x40:
    gpio.output(D6, 1)
  if ch&0x80==0x80:
    gpio.output(D7, 1)
  gpio.output(EN, 1)
  time.sleep(0.005)
  gpio.output(EN, 0)
  # Low bits
  gpio.output(D4, 0)
  gpio.output(D5, 0)
  gpio.output(D6, 0)
  gpio.output(D7, 0)
  if ch&0x01==0x01:
```

```
    gpio.output(D4, 1)
  if ch&0x02==0x02:
    gpio.output(D5, 1)
  if ch&0x04==0x04:
    gpio.output(D6, 1)
  if ch&0x08==0x08:
    gpio.output(D7, 1)
  gpio.output(EN, 1)
  time.sleep(0.005)
  gpio.output(EN, 0)
def lcdclear():
  lcdcmd(0x01)

def lcdprint(Str):
  l=0;
  l=len(Str)
  for i in range(l):
    lcdwrite(ord(Str[i]))

def setCursor(x,y):
    if y == 0:
        n=128+x
    elif y == 1:
        n=192+x
    lcdcmd(n)

def enrollFinger():
    lcdcmd(1)
    lcdprint("Enrolling Finger")
```

```
time.sleep(2)
print('Waiting for finger...')
lcdcmd(1)
lcdprint("Place Finger")
while ( f.readImage() == False ):
    pass
f.convertImage(0x01)
result = f.searchTemplate()
positionNumber = result[0]
if ( positionNumber >= 0 ):
    print('Template already exists at position #' + str(positionNumber))
    lcdcmd(1)
    lcdprint("Finger ALready")
    lcdcmd(192)
    lcdprint("   Exists    ")
    time.sleep(2)
    return
print('Remove finger...')
lcdcmd(1)
lcdprint("Remove Finger")
time.sleep(2)
print('Waiting for same finger again...')
lcdcmd(1)
lcdprint("Place Finger")
lcdcmd(192)
lcdprint("   Again    ")
while ( f.readImage() == False ):
    pass
f.convertImage(0x02)
```

```
    if ( f.compareCharacteristics() == 0 ):
        print "Fingers do not match"
        lcdcmd(1)
        lcdprint("Finger Did not")
        lcdcmd(192)
        lcdprint("   Mactched   ")
        time.sleep(2)
        return
    f.createTemplate()
    positionNumber = f.storeTemplate()
    print('Finger enrolled successfully!')
    lcdcmd(1)
    lcdprint("Stored at Pos:")
    lcdprint(str(positionNumber))
    lcdcmd(192)
    lcdprint("successfully")
    print('New template position #' + str(positionNumber))
    time.sleep(2)

def searchFinger():
    try:
        print('Waiting for finger...')
        while( f.readImage() == False ):
            #pass
            time.sleep(.5)

        f.convertImage(0x01)
        result = f.searchTemplate()
        positionNumber = result[0]
```

```
        accuracyScore = result[1]
        if positionNumber == -1 :
            print('No match found!')
            gpio.output(2,gpio.HIGH)
            lcdcmd(1)
            lcdprint("No Match Found")
            time.sleep(2)
            return
        else:

            print('Found template at position #' + str(positionNumber))
            lcdcmd(1)
            lcdprint("Found at Pos:")
            gpio.output(2,gpio.LOW)
            lcdprint(str(positionNumber))
            time.sleep(2)
            return positionNumber

    except Exception as e:
        print('Operation failed!')
        print('Exception message: ' + str(e))
        exit(1)

def deleteFinger():
    positionNumber=searchFinger()
    count=positionNumber
    lcdcmd(1)
    print("Delete finger postion")
    lcdprint("Delete Finger")
```

```
    lcdcmd(192)
    lcdprint("Position: ")
    lcdcmd(0xca)
    lcdprint(str(count))



    print(positionNumber)
    if f.deleteTemplate(positionNumber) == True :
        print('Template deleted!')
        lcdcmd(1)
        lcdprint("Finger Deleted");
        time.sleep(2)

begin()
lcdcmd(0x01)
lcdprint("FingerPrint ")
lcdcmd(0xc0)
lcdprint("Interfacing ")
time.sleep(3)
lcdcmd(0x01)
lcdprint("Circuit Digest")
lcdcmd(0xc0)
lcdprint("Welcomes You  ")
time.sleep(3)
flag=0
lcdclear()

while 1:
    gpio.output(led, HIGH)
```

```
    lcdcmd(1)
    print("1.Enroll\n2.Delete\n3.Search")
    opti = input("Enter option:")
    lcdprint("Place Finger")
    if opti == 1:
        gpio.output(led, LOW)
        enrollFinger()
    elif opti == 2:
        gpio.output(led, LOW)


        deleteFinger()
    else:
        searchFinger()
```

Encoding_faces.py

```
# python encode_faces.py --dataset dataset --encodings encodings.pickle


from imutils import paths
import face_recognition
import argparse
import pickle
import cv2
```

```
import os


ap = argparse.ArgumentParser()
ap.add_argument("-i", "--dataset", required=True,
	help="path to input directory of faces + images")
ap.add_argument("-e", "--encodings", required=True,
	help="path to serialized db of facial encodings")
ap.add_argument("-d", "--detection-method", type=str, default="hog",
	help="face detection model to use: either `hog` or `cnn`")
args = vars(ap.parse_args())


print("[INFO] quantifying faces...")
imagePaths = list(paths.list_images(args["dataset"]))


knownEncodings = []
knownNames = []
for (i, imagePath) in enumerate(imagePaths):
	print("[INFO] processing image {}/{}".format(i + 1,
		len(imagePaths)))
	name = imagePath.split(os.path.sep)[-2]


	image = cv2.imread(imagePath)
	rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)


	boxes = face_recognition.face_locations(rgb,
		model=args["detection_method"])
	encodings = face_recognition.face_encodings(rgb, boxes)
	for encoding in encodings:
```

```
                knownEncodings.append(encoding)
                knownNames.append(name)
print("[INFO] serializing encodings...")
data = {"encodings": knownEncodings, "names": knownNames}
f = open(args["encodings"], "wb")
f.write(pickle.dumps(data))
f.close()
```

Recognize_faces_video.py

```
# python recognize_faces_video.py --encodings encodings.pickle
```

```
# python recognize_faces_video.py --encodings encodings.pickle --output output/jurassic_park_trailer_output.avi --display 0

from imutils.video import VideoStream
import face_recognition
import argparse
import imutils
import pickle
import time
import cv2
import MySQLdb
import datetime

ap = argparse.ArgumentParser()
ap.add_argument("-e", "--encodings", required=True,
	help="path to serialized db of facial encodings")
ap.add_argument("-o", "--output", type=str,
	help="path to output video")
ap.add_argument("-y", "--display", type=int, default=1,
	help="whether or not to display output frame to screen")
ap.add_argument("-d", "--detection-method", type=str, default="hog",
	help="face detection model to use: either `hog` or `cnn`")
args = vars(ap.parse_args())

print("[INFO] loading encodings...")
data = pickle.loads(open(args["encodings"], "rb").read())

print("[INFO] starting video stream...")
```

```
vs = VideoStream(src=0).start()
writer = None
time.sleep(2.0)

while True:
        frame = vs.read()

    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    rgb = imutils.resize(frame, width=750)
    r = frame.shape[1] / float(rgb.shape[1])

    boxes = face_recognition.face_locations(rgb,
        model=args["detection_method"])
    encodings = face_recognition.face_encodings(rgb, boxes)
    names = []

        for encoding in encodings:
matches = face_recognition.compare_faces(data["encodings"],
            encoding)
        name = "Unknown"
if True in matches:
matchedIdxs = [i for (i, b) in enumerate(matches) if b]
counts = {}
            for i in matchedIdxs:
                name = data["names"][i]
                counts[name] = counts.get(name, 0) + 1
            name = max(counts, key=counts.get)
        names.append(name)
```

```
str=name
t=datetime.datetime.now()
                print("identified faces:",str)
                print("date:",datetime.datetime.now())
                db=MySQLdb.connect
("localhost","phpmyadmin","sai518","facerecognition")
                Mycursor=db.cursor()
                mycursor.execute
("insert IGNORE into recognizedfaces values(%s,%s,%s,%s)",(str,t,t,'present'))
db.commit()
                mycursor.execute("select rollno,status from student")
                rs=mycursor.fetchall()
                for i in rs:
                        x=i[0]
                        y=i[1]
mycursor.execute("""insert into attendance values(%s,%s,%s)""",(x,t,y))
mycursor.execute("update attendance inner join recognizedfaces on
recognizedfaces.rollno=attendance.rollno set attendance.Status='present'")
                db.commit()
                db.close()
for ((top, right, bottom, left), name) in zip(boxes, names):
                top = int(top * r)
                right = int(right * r)
                bottom = int(bottom * r)
                left = int(left * r)
                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,
0.75, (0, 255, 0), 2)
        if writer is None and args["output"] is not None:
                fourcc = cv2.VideoWriter_fourcc(*"MJPG")
                writer = cv2.VideoWriter(args["output"], fourcc, 20,
                        (frame.shape[1], frame.shape[0]), True)
        if writer is not None:
                writer.write(frame)
        if args["display"] > 0:
                cv2.imshow("Frame", frame)
                key = cv2.waitKey(1) & 0xFF
                if key == ord("q"):
                        break
cv2.destroyAllWindows()
vs.stop()
if writer is not None:
                writer.release()
```

CHAPTER - 7

TESTING

TESTING

7.1 Introduction:

software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. software testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

software testing can also be stated as the process of validating and verifying that a software program/application/product:

1. meets the business and technical requirements that guided its design and development;

2. works as expected; and

3. can be implemented with the same characteristics.

software testing, depending on the testing method employed, can be implemented at any time in the development process. however, most of the test effort occurs after the requirements have been defined and the coding process has been completed. as such, the methodology of the test is governed by the software development methodology adopted.

different software development models will focus the test effort at different points in the development process. newer development models, such as agile, often employ test driven development and place an increased portion of the testing in the

hands of the developer, before it reaches a formal team of testers. In a more traditional model, most of the test execution occurs after the requirements have been defined and the coding process has been completed.

Testing can never completely identify all the defects within software. Instead, it furnishes a criticism or comparison that compares the state and behavior of the product against oracles—principles or mechanisms by which someone might recognize a problem. These oracles may include (but are not limited to) specifications, contracts,[2] comparable products, past versions of the same product, inferences about intended or expected purpose, user or customer expectations, relevant standards, applicable laws, or other criteria.

Every software product has a target audience. For example, the audience for video game software is completely different from banking software. Therefore, when an organization develops or otherwise invests in a software product, it can assess whether the software product will be acceptable to its end users, its target audience, its purchasers, and other stakeholders. Software testing is the process of attempting to make this assessment.

A study conducted by NIST in 2002 reports that software bugs cost the U.S. economy $59.5 billion annually. More than a third of this cost could be avoided if better software testing was performed.

7.2 TYPES OF TESTS:

1.UNIT TESTING:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

2.INTEGRATION TESTING:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

3. FUNCTIONAL TESTING:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:
Valid Input : Identified classes of valid input must be accepted.
Invalid Input : Identified classes of invalid input must be rejected. Functions : Identified functions must be exercised.
Output : Identified classes of application outputs
Systems/Procedures : Interfacing systems or procedures must be invoked.

4. SYSTEM TESTING:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

5. WHITE BOX TESTING:

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6. BLACK BOX TESTING:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested.

Testing methods:

The box approach:

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

White box testing:

White box testing is when the tester has access to the internal data structures and algorithms including the code that implement these.

Types of white box testing:

The following types of white box testing exist:

* API testing (application programming interface) - testing of the application using public and private APIs * Code coverage - creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least onc

* Fault injection methods - improving the coverage of a test by introducing faults to test code paths

* Mutation testing methods

* Static testing - White box testing includes all static testing

Test coverage:

White box testing methods can also be used to evaluate the completeness of a test suite that was created with black box testing methods. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested.Two common forms of code coverage are:

* Function coverage, which reports on functions executed

* Statement coverage, which reports on the number of lines executed to complete the test.

Black box testing:

Black box testing treats the software as a "black box"—without any knowledge of internal implementation. **B**lack box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.

Specification-based testing:

Specification-based testing aims to test the functionality of software according to the applicable requirements. Thus, the tester inputs data into, and only sees the output from, the test object. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case.Specification-based testing is necessary, but it is insufficient to guard against certain risks.

Advantages and disadvantages:

The black box tester has no "bonds" with the code, and a tester's perception is very simple: a code must have bugs. Using the principle, "Ask and you shall receive," black box testers find bugs where programmers do not. But, on the other hand, black box testing has been said to be "like a walk in a dark labyrinth without a flashlight," because the tester doesn't know how the software

being tested was actually constructed. As a result, there are situations when (1) a tester writes many test cases to check something that could have been tested by only one test case, and/or (2) some parts of the back-end are not tested at all. Therefore, black box testing has the advantage of "an uaffiliated opinion," on the one hand, and the disadvantage of "blind exploring," on the other.

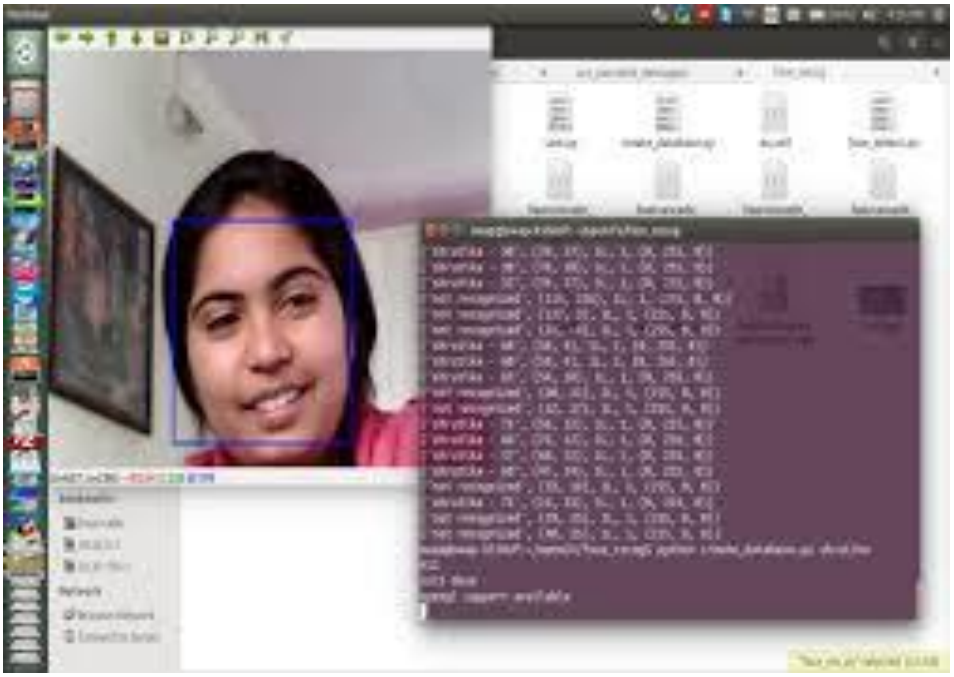
CHAPTER - 8

EXPERIMENTAL RESULTS

EXPERIMENTAL RESULTS

Output of fingerprint py

Output of face recognize.py:



CHAPTER - 9

CONCLUSION

CONCLUSION

We successfully finished a face recognition door unlock system as we planned .The biometric and face recognition works well . there is high accuracy in recognizing house owner faces and it could realize sending the matched face image to another raspberry pi in time and give a good output. And it takes a little bit time to recognize.at all, we all are satisfied to build it.

Outcome Of This Project:

- ✓ A Scientific approach was developed during project work.
- ✓ Skills and self-confidence in coding and working with software was developed.
- ✓ An applicable door unlocks using face recognition and biometric device is used in security purpose environment.
- ✓ An improved and faster facial recognition system was developed for authentication person only unlock the door.
- ✓ Various new algorithms like HOG,CNN were invented in this project.
- ✓ It was then estimated that for a database of size millions, our system will take only around 1 minute at maximum while existing one will take around half an hour for identification of an individual in worst case.
- ✓ Facial recognition is an efficient tool to detect fraud and verify identity.
- ✓ Biometric device is also efficient tool to detect fraud and verify identity.
- ✓ It provides a simple and cost effective method of door unlock using face recognition and biometric device.

Chapter -10

References

REFERENCES

BIBLIOGRAPHY:

Java Complete reference 5th edition Head First Java.

Sites Referred:

https://tutorials-raspberrypi.com/how-to-use-raspberry-pi-fingerprint-sensor-authentication/

https://www.datacamp.com/community/tutorials/face-detection-python-opencv

https://maker.pro/raspberry-pi/projects/how-to-create-a-facial-recognition-door-lock-with-raspberry-pi

https://iotdesignpro.com/projects/face-recognition-door-lock-system-using-raspberry-pi

http://java.sun.com,

https://community.createlabz.com/knowledgebase/rfid-reader-and-fingerprint-sensor-based-door-lock-system-using-raspberry-pi-zero-with-mysql-database/

http://www.analysisandsolution.com,

http://www.dbbalance.com,

http://www.java2s.com.

SCOPE FOR FUTURE WORK

Future scope:

The future scope of this work is very wide, and there are many other security tool can be added to the system to provide more security such as iris scanner for a person visual identification, fire sensors connected to the alerting alarms, and the system can be enhanced using artificial intelligence techniques to speed up and facilitate the process of identification of a person by using face recognition technique with a database of popular burglars