**CHAPTER 1: INTRODUCTION 1**

* + [Introduction to voting system 1](#_heading=h.1fob9te)

[**CHAPTER 2: LITERATURE SURVEY 2**](#_heading=h.3znysh7)

**CHAPTER 3 : DESIGN OF FINGERPRINT BASED**

**EVM USING AADHAR**

* 1. [Existing system](#_heading=h.2et92p0) 
     1. Manual balloting
     2. Drawbacks of manual balloting
  2. [Proposed system](#_heading=h.tyjcwt) 
     1. [Objective](#_heading=h.3dy6vkm)
     2. [Block diagram](#_heading=h.1t3h5sf)
     3. [Flow chart](#_heading=h.4d34og8)
     4. [Working](#_heading=h.2s8eyo1)
     5. [Tools Required](#_heading=h.17dp8vu) 
        1. [Hardware tools](#_heading=h.3rdcrjn)
        2. [Software tools](#_heading=h.26in1rg)
     6. [Applications](#_heading=h.lnxbz9)
     7. [Advantages](#_heading=h.35nkun2)

**CHAPTER 4: HARDWARE AND SOFTWARE DESCRIPTION**

* 1. Arduino Mega 2560
     1. [Overview](#_heading=h.1ksv4uv)
     2. [Power](#_heading=h.44sinio)
     3. [Memory](#_heading=h.2jxsxqh)
     4. [Input and output](#_heading=h.z337ya)
     5. [Communication](#_heading=h.3j2qqm3)
     6. [Programming](#_heading=h.1y810tw)

4.2 R307 module

* + 1. [Operation principle](#_heading=h.4i7ojhp)
    2. [Features](#_heading=h.2xcytpi)
    3. [Specifications](#_heading=h.1ci93xb)
    4. [Pin configuration](#_heading=h.3whwml4)
  1. 4\*4 Matrix Keypad
     1. [Introduction](#_heading=h.2bn6wsx)
     2. [Working](#_heading=h.qsh70q)
     3. Pin Configuration
     4. Features and Specifications
     5. Applications
  2. LCD Module( 2x16 character)
  3. GSM Module
  4. Arduino IDE
     1. [Connecting the arduino](#_heading=h.3as4poj)
     2. [Preparing the board](#_heading=h.1pxezwc)
     3. Loading code
  5. Embedded C programming
     1. [Introduction to Embedded C programming](#_heading=h.49x2ik5)
     2. [Embedded systems programming](#_heading=h.2p2csry)
     3. Difference between C programming and

Embedded C programming

* + 1. [Programming using Embedded C](#_heading=h.147n2zr)

**CHAPTER 5: RESULTS AND DISCUSSION**

[**CONCLUSION**](#_heading=h.3o7alnk)

[**FUTURE SCOPE**](#_heading=h.23ckvvd)

**REFERENCES**

**APPENDIX**

## ABSTRACT

Biometric Finger print devices are used in the Electronic Voting machine for voter verification. We have designed a finger print based voting machine where there is no need for the user to carry his ID which contains his required details. The person at the polling booth needs only to place his Finger on the device, thus allowing the acquisition of an on-spot fingerprint from the voter which serves as an identification. This Finger print reader reads the details from the tag. This data is passed onto the controlling unit for the verification. The controller fetches the data from the reader and compares this data with the already existing data stored during the registration of the voters. If the data matches with the pre-stored information of the registered fingerprint, the person is allowed to cast his vote. If not, a warning message is displayed on LCD and the person is barred from polling his vote. The vote casting mechanism is carried out manually using the push buttons. LCD is used to display the related messages, warnings and ensuing results.

## CHAPTER 1

**INTRODUCTION**

## Introduction to voting system

Modern voting system came from French philosophers **Marquis de Condorcet** and **Jeans-Charles Borda**. Voting system is directly involved with economics, political science and social science. So that if one has no proper knowledge of the voting system then he cannot serve the society in proper way and cannot expect the economic development of the society. Condorcet, Borda and even many modern politicians believe that elections are logically imperfect. Even in the 21st century the voting system in different countries are not fully perfect. Some cases political leaders apply their suitable policies in voting for of their own benefits and many cases they did not see the citizens welfare. Many organizations and some countries are applying these voting system is fully or partially. This book has discussed Condorcet voting method, Borda voting, majority judgement of voting, single transferable voting, approval voting and median voting method.

In a democratic framework the irregular residents choose an up-and-comer by law who speaks to them and works for them for their government assistance. In the event that an off-base up-and-comer is chosen, it carries a catastrophe to the country. In creating nations like India, the irregular method for political race is polling form paper based framework in days of yore, which is particularly tedious and once in a while especially risky. By this framework there is in every case some hazard to choose an inappropriate applicant. With the goal that the legislature of India presented EVM political race however this framework additionally needs security. Thinking about these issues, right now, new arrangement of casting a ballot is proposed dependent on electronic voting machine (EVM).

The extra component of this framework is biometric security alongside the aadhar number of the resident which will be acknowledged by the fingerprints of the voters. In casting a ballot frameworks, the framework ought to be anything but difficult to confirm and check, it likewise ought to have high exactness rate and unwavering quality. The framework additionally must be financially savvy and remarkable. Right now, the proposed framework will distinguish every voter by their unique finger impression and their aadhar number with the assistance of a unique mark scanner (FPS) . It can recognize on the off chance that somebody is an enrolled voter or not by contrasting and the database present. It will deny on the off chance that somebody attempts to make a subsequent option.

## CHAPTER 2 LITERATURE SURVEY

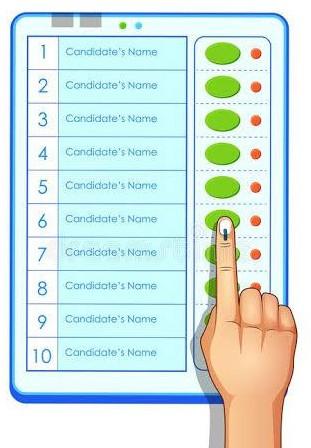
## CHAPTER 3

**DESIGN OF EVM SYSTEM**

## Existing system

EVM is the existing system. In existing system the voter can easily cast the vote without any fingerprint authorization. After the verification of the identity card issued by the government there will be no authorization. Due to that there are more chances for fake votes to be casted.

## EVM



**Fig 3.1.1 : EVM (Electronic voting Machine)**

* + 1. **Drawbacks of EVM**
* Consumes more time to declare results.
* Chance of casting fake votes.
* No authorization required.

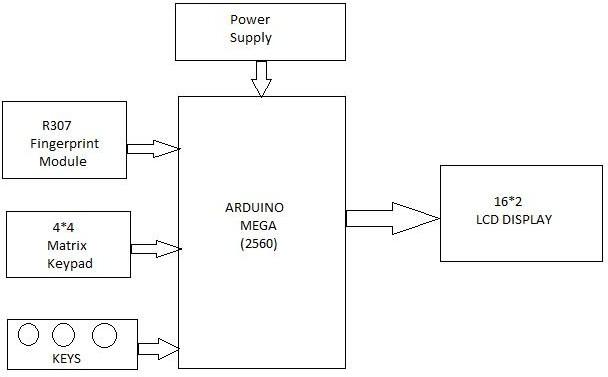
## Proposed system

## Objective

The main objective of the proposed system is to design a electronic voting machine for election process. At present the government is using ballot voting for the election process. This will consume time and inaccurate because water quality parameters may alter with time. In order to overcome this problem, technology should be brought to election process minimize the fake votes.

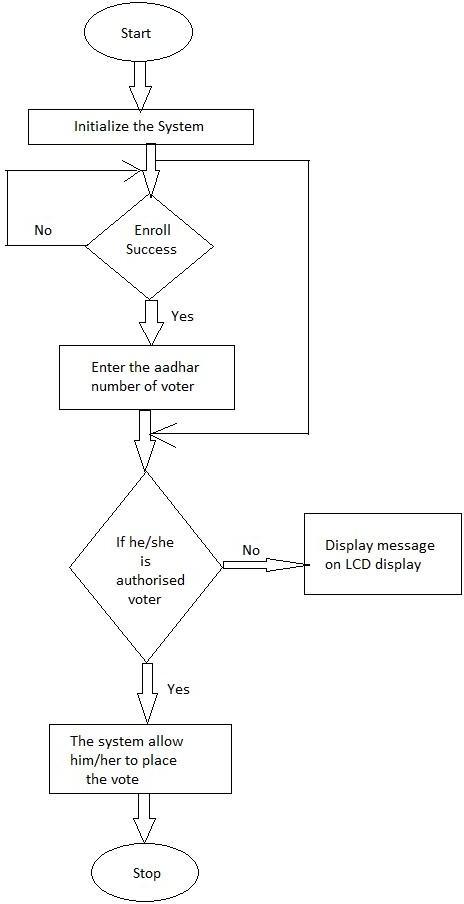
The proposed work uses an integrated on chip computer arduino mega 2560 controller. Fingerprint sensor is mounted to sense the voter finger and allow him/her to place his/her vote. This is reliable and reduces the time taken for election process.

## BlockDiagram



**Fig.3.2.2 Block diagram of proposed system**

## Flow chart



**Fig.3.2.3 Flowchart of proposed system**

## Working

First we have to enroll voters fingers into the fingerprint module. After enrolling the fingers the system allows the voter to place his/her vote. If he/she is authorised voter the system allows the voter to place his/her vote. If the voter place the vote for the second time the system does not accept the vote.

# Enrolling

While enrolling the fingerprint module scans the voters finger two times.

To enroll the fingers of the voter, First we have to press the enroll button and then select the finger print ID. After selecting the id the system shows the message(place the finger) on the LCD display. After placing the finger it scans the voters finger , After scanning the finger it again shows the message(Remove finger & place finger again) on LCD display, And it scans the finger for the 2nd time and the fingerprint of the voter is stored into the EEPROM memory of the fingerprint module.

# Deleting the fingerprints

To delete the single fingerprint, Press delete button and then select the id and then press ok. Then the fingerprint on selected id is deleted.

To delete all the fingerprints in at a time, press reset button on the arduino board and wait for some time. Then the LCD display show the message that FOUND MODULE. After showing that message on LCD display, immediately press the delete button. Then all the fingerprints stored in the fingerprint module are deleted.

----> To see the results, press results button and the results are shown on the LCD display.

---->To reset the system, power on the system and if the LCD display shows the message that FOUND MODULE, then immediately press results button.

----> To place his/her vote, press match key and then place the voters finger on the fingerprint module.

## Tools Required

## Hardware Tools

* + - * + Arduino Mega 2560
        + R307 Fingerprint module
        + 4\*4 Matrix Keypad
        + LCD display

## Software Tools

* + - * + Arduino IDE

## Applications

* The main of application of this system to conduct the elections in proper manner without any mismanagement.

## Advantages

* Fast and easy way of conducting Election.
* Voters can view background of each Candidate.
* Candidate can present themselves against voters.

## CHAPTER 4 HARDWAREAND SOFTWARE DESCRIPTION

* 1. **Arduino Mega 2560**



**Fig .4.1Arduino Mega 2560**

## Overview

The Arduino Mega 2560 is a micro controller board based on the ATmega2560 (data-sheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the micro controller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the ArduinoDuemilanove or Diecimila.

## Power

The Arduino Mega can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board

may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts. The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows

The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

## Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM

## Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode() , digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

* + - * Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
      * External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
      * PWM: 0 to 13. Provide 8-bit PWM output with the analogWrite() function.
      * SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, Duemilanove and Diecimila.
      * LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
      * I2C: 20 (SDA) and 21 (SCL). Support I2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I2C pins on the Duemilanove or Diecimila.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function. There are a couple of other pins on the board:

* AREF. Reference voltage for the analog inputs. Used with analogReference()

. ● Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button

## Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial

communication on pins 0 and 1). A SoftwareSerial library allows for serial communication on any of the Mega2560's digital pins. The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. For SPI communication, use the SPI library.

## Programming

The Arduino Mega can be programmed with the Arduino software (download). For details, see the reference and tutorials. The ATmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files). You can also bypass the bootloader and program the microcontroller through the ICSP (In Circuit Serial Programming) header; see these instructions for details

# FINGERPRINT module



## Fig.4.2 R305 module

R307 Fingerprint Module consists of optical fingerprint sensor, high-speed DSP processor, high-performance fingerprint alignment algorithm, high-capacity FLASH chips and other hardware and software composition, stable performance,

simple structure, with fingerprint entry, image processing, fingerprint matching, search and template storage and other functions.

## Operation principle

Fingerprint processing includes two parts: fingerprint enrollment and fingerprint matching (the matching can be 1:1 or 1:N).

When enrolling, user needs to enter the finger two times. The system will process the two time finger images, generate a template of the finger based on processing results and store the template. When matching, user enters the finger through optical sensor and system will generate a template of the finger and compare it with templates of the finger library. For 1:1 matching, system will compare the live finger with specific template designated in the Module; for 1:N matching, or searching, system will search the whole finger library for the matching finger. In both circumstances, system will return the matching result, success or failure.

## Features

* **Perfect function** : independent fingerprint collection, fingerprint registration, fingerprint comparison (1: 1) and fingerprint search (1: N) function.
* **Small size** : small size, no external DSP chip algorithm, has been integrated, easy to install, less fault.
* **Ultra-low power consumption** : low power consumption of the product as a whole, suitable for low-power requirements of the occasion.
* **Anti-static ability** : a strong anti-static ability, anti-static index reached 15KV above.
* **Application development is simple** : developers can provide control instructions, self-fingerprint application product development, without the need for professional knowledge of fingerprinting.
* **Adjustable security level** : suitable for different applications, security levels can be set by the user to adjust.
* Finger touch sensing signal output, low effective, sensing circuit standby current is very low, less than 5uA.

## Specifications

* Supply voltage: DC 4.2 ~ 6.0V
* Supply current: Working current: 50mA (typical) Peak current: 80mA
* Fingerprint image input time: <0.3 seconds
* Window area: 14x18 mm
* Matching method: Comparison method (1: 1)
* Search method (1: N)
* Characteristic file: 256 bytes
* Template file: 512 bytes
* Storage capacity: 1000 pieces
* Security Level: Five (from low to high: 1,2,3,4,5)
* Fake rate (FAR): <0.001%
* Refusal rate (FRR): <1.0%
* Search time: <1.0 seconds (1: 1000 hours, mean value)
* Host interface: UART \ USB1.1
* Communication baud rate (UART): (9600xN) bps Where N = 1 ~ 12 (default N = 6, ie 57600bps)
* Working environment: Temperature: -20 ℃ - +40 ℃ Relative humidity: 40% RH-85% RH (no condensation)
* Storage environment: Temperature: -40 ℃ - +85 ℃ Relative humidity: <85% H (no condensation).

## Pin configuration

* + 5V Regulated 5V DC
  + GND Common Ground
  + TXDData output - Connect to MCU RX
  + RXD Data Input - Connect to MCU TX
  + TOUCH Active Low output when there is touch on sensor by finger
  + 3.3VUse this wire to give 3.3V to sensor instead of 5V

# 4\*4 KEYPAD



## Fig.4.3 4\*4 matrix keypad

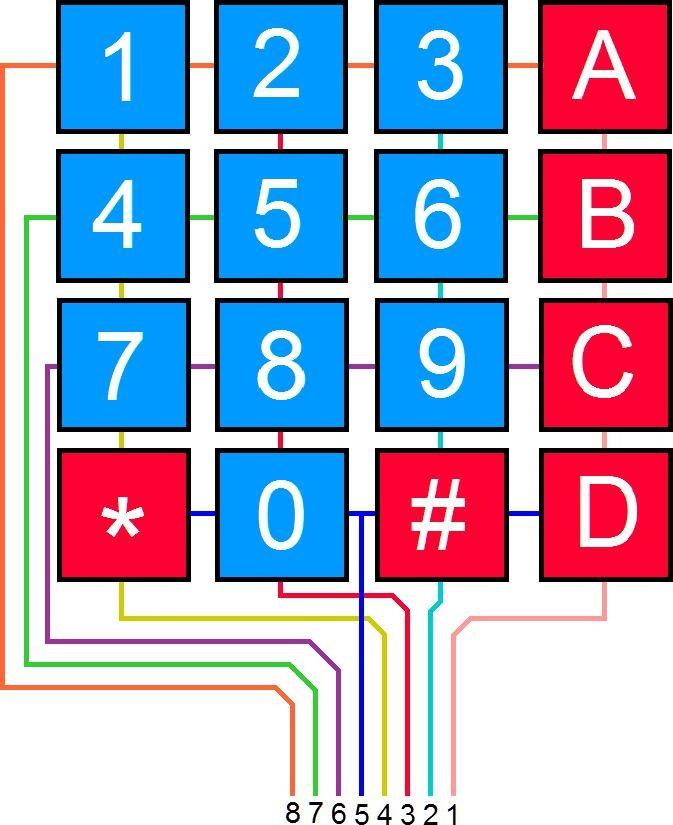
## Introduction

A keypad is a set of buttons arranged in a block or “pad” which bear digits, symbols or alphabetical letters. Pads mostly containing numbers are called a numeric keypad.

The 4 x 4 matrix keypad usually is used as input in a project. It has 16 keys in total, which means the same input values. It is ultra-thin, easy to interface with any micro controller and has an adhesive backing for easy mounting for a variety of applications..

## Working

Matrix keypads use a combination of four rows and four columns to provide button states to the host device, typically a micro controller. Underneath each key is a push button, with one end connected to one row, and the other end connected to one column. These connections are shown in Figure 1.



## Fig 4.3.2 : Matrix Keypad Connections

In order for the micro controller to determine which button is pressed, it first needs to pull each of the four columns (pins 1-4) either low or high one at a time, and then poll the states of the four rows (pins 5-8). Depending on the states of the columns, the micro controller can tell which button is pressed.

## 4\*4 Keypad pin configuration

4X4 KEYPAD MODULES are available in different sizes and shapes. But they all have same pin configuration. It is easy to make 4X4 KEYPAD by arranging 16 buttons in matrix formation by yourself.

Pin Number Description ROWS

1. PIN1 is taken out from 1st ROW
2. PIN2 is taken out from 2nd ROW
3. PIN3 is taken out from 3rd ROW
4. PIN4 is taken out from 4th ROW COLUMN
5. PIN5 is taken out from 1st COLUMN
6. PIN6 is taken out from 2nd COLUMN
7. PIN7 is taken out from 3rd COLUMN
8. PIN8 is taken out from 4th COLUMN

As given in above table a 4X4 KEYPAD will have EIGHT TERMINALS. In them four are ROWS of MATRIX and four are COLUMNS of MATRIX. These 8 PINS are driven out from 16 buttons present in the MODULE. Those 16 alphanumeric digits on the MODULE surface are the 16 buttons arranged in MATRIX formation.

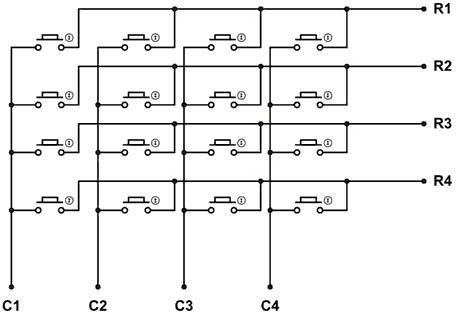


Fig 4.3.3 : 4x4 Keypad Internal Structure

## 4\*4 Keypad Module Features and Specifications

* + Maximum Voltage across EACH SEGMENT or BUTTON: 24V
  + Maximum Current through EACH SEGMENT or BUTTON: 30mA
  + Maximum operating temperature: 0°C to + 50°C
  + Ultra-thin design
  + Adhesive backing
  + Easy interface
  + Long life.

## 4\*4 Keypad Module Applications

* Security systems.
* Vending machines.
* Industrial machines.
* Engineering systems.
* Measuring instruments.
* Data entry for Embedded Systems
* Hobby projects.
* Basically any where INPUT device is needed.

# LCD Module (2X16 character)

Dot matrix LCD modules is used for display the parameters and fault condition.16 characters 2 lines display is used. It has controller which interface data’s and LCD panel. Liquid crystal displays (LCD’s) have materials, which combine the properties of both liquids and crystals. Rather than having a melting point, they have a temperature range within which the molecules are almost as mobile as they would be in a liquid, but are grouped together in an ordered form similar to a crystal. An LCD consists of two glass panels, with the liquid crystal material sandwiched in between them. The inner surface of the glass plates are coated with transparent electrodes which define the character, symbols or patterns to be displayed polymeric layers are present in between the electrodes and the liquid crystal molecules to maintain a defined orientation angle.

One each polarizer’s are pasted outside the two glass panels. These polarizer’s would rotate the light rays passing through them to a definite angle, in a particular direction When the LCD is in the off state, light rays are rotated by the two polarizes and the liquid crystal, such that the light rays come out of the LCD without any orientation, and hence the LCD appears transparent.

When sufficient voltage is applied to the electrodes, the liquid crystal molecules would be aligned on a specific direction.

## Fig.4.4 LCD Diagram

The LCD's are lightweight with only a few millimeters thickness. since the LCD's consume less power, they are compatible with low power electronic circuits, and can be powered for long durations .

Each [pixel](https://en.wikipedia.org/wiki/Pixel) of an LCD typically consists of a layer of [molecules](https://en.wikipedia.org/wiki/Molecule) aligned between two [transparent](https://en.wikipedia.org/wiki/Transparency_(optics)) [electrodes](https://en.wikipedia.org/wiki/Electrode), and two [polarizing](https://en.wikipedia.org/wiki/Polarizer) [filters](https://en.wikipedia.org/wiki/Filter_(optics)) (parallel and perpendicular), the axes of transmission of which are (in most of the cases) perpendicular to each other. Without the [liquid crystal](https://en.wikipedia.org/wiki/Liquid_crystal) between the polarizing filters, light passing through the first filter would be blocked by the second (crossed) polarizer. Before an [electric field](https://en.wikipedia.org/wiki/Electric_field) is applied, the orientation of the liquid-crystal molecules is determined by the alignment at the surfaces of electrodes. In a twisted nematic (TN) device, the surface alignment directions at the two electrodes are perpendicular to each other, and so the molecules arrange themselves in a [helical](https://en.wikipedia.org/wiki/Helix) structure, or twist. This induces the rotation of the polarization of the incident light, and the device appears gray. If the applied voltage is large enough, the liquid crystal molecules in the center of the layer are almost completely untwisted and the polarization of the [incident light](https://en.wikipedia.org/wiki/Incident_light) is not rotated as it passes through the liquid crystal layer. This light will then be mainly polarized [perpendicular](https://en.wikipedia.org/wiki/Perpendicular) to the second filter, and thus be blocked and the [pixel](https://en.wikipedia.org/wiki/Pixel) will appear black. By controlling the voltage applied across the liquid crystal layer in each pixel, light can be allowed to pass through in varying amounts thus constituting different levels of gray. Color LCD

systems use the same technique, with color filters used to generate red, green, and blue pixels

**4.5 GSM Module:**

GSM (Global System for Mobile Communications) is a standard developed by the European Telecommunications Standards Institute (ETSI) to describe the protocols for second-generation digital cellular networks used by mobile devices such as tablets, first deployed in Finland in December 1991. 2G networks developed as a replacement for first generation (1G) analog cellular networks, and the GSM standard originally described as a digital, circuit-switched network optimized for full duplex voice telephony.

SIM900 is a complete Quad-band GSM/GPRS module. Featuring an industry-standard interface, the SIM900 delivers GSM/GPRS 850/900/1800/1900MHz performance for voice, SMS, Data, and Fax in a small form factor and with low power consumption. The Module is coming with RS232 interface, which allows you connect PC as well as microcontroller with RS232 Chip(MAX232). The baud rate is configurable from 9600-115200 through AT command. It is suitable for SMS, Voice as well as DATA transfer application in M2M interface. The onboard Regulated Power supply allows you to connect wide range unregulated power supply. Using this modem, you can make audio calls, SMS, Read SMS, attend the incoming calls and internet etc. through simple AT commands.

What are AT Commands:

AT commands are commands which are used to control the modems where AT stands for Attention. These commands were derived from Hayes commands which were used by the Hayes smart modems. Every wireless as well as the dial up modems require an AT command to interact with a computer machine. These AT commands along with other extended commands also require Hayes command set as a subset.

Usage:

The AT commands can be used with GSM module and GPRS MODEMs or phone to access these services and information:

· SMS

· MMS

· Fax

Voice link and other data over mobile network

Information and configuration concerning the mobile devices or MODEM and SIM card.

Types of AT Command:

There are 4 basic types of AT commands:

Test: The test command is utilised to check the compatibility of a command by a modem. SYNTAX: AT=?

2. Read: Read command is used for extracting the mobile or modem settings required for operations.

SYNTAX: AT?

3. Set: This commands is used to make changes into mobile phone or modem settings required for the operation.

SYNTAX: AT=value1, value2, …, valueN

4. Execution: As the name suggests, this command is used to execute the said operation.

SYNTAX: AT=parameter1, parameter2, …, parameterN

Most Commonly Used AT Commands:

The below 7 are the most commonly used AT commands:

AT : Used to check the interaction between the computer and the module. This command is usually replied with an OK if the port and the module can connect correctly, else wise it comes back with a result code ERROR.

2. +CMGF : Used to setup the SMS mode. By adding 1 or 0 with the command text or PDU mode can be selected. Here the text mode is easy to operate although it only allows a few limited features of SMS. Where as the PDU mode allows a more detailed access to the SMS service, although to use this you require some basic knowledge of TDPU.

SYNTAX: AT+CMGF=<mode>

3. +CMGS : Used to send SMS to a particular phone number

SYNTAX: AT+CMGS= serial number of message to be send.

4. ATD : Used to make call to a particular number

SYNTAX: ATD;(Enter)

5. ATA : Used to answer the incoming calls. The calls are denoted by a message ‘RING’ which duplicated for every ring of the call. After the call ends a message saying ‘NO CARRIER’ is displayed.

SYNTAX: ATA(Enter)

6. +CMGW: Used to store a message in the SIM. After the execution of the command, the ‘>’

sign appears in the next line where the message can be entered.

SYNTAX: AT+CMGW=” Phone number”> Message to be stored Ctrl+z

7. ATH : this commaned is utilized to disconnect a remote user with the GSM module. SYNTAX: ATH (Enter)

List of AT Commands

Testing :

Command Description

AT Checking communication between the module and computer.

Call control :

Command Description

ATA Answer command

ATD Dial command

ATH Hang up call

ATL Monitor speaker loudness

ATM Monitor speaker mode

ATO Go on-line

ATP Set pulse dial as default

ATT Set tone dial as default

AT+CSTA Select type of address

AT+CRC Cellular result codes

Data card Control :

Command Description

ATI Identification

ATS Select an S-register

ATZ Recall stored profile

AT&F Restore factory settings

AT&V View active configuration

AT&W Store parameters in given profile

AT&Z Select Set as power up option

AT+CLCK Facility lock command

AT+COLP Connected line identification presentation

AT+GCAP Request complete capabilities list

AT+GMI Request manufacturer identification

AT+GMM Request model identification

AT+GMR Request revision identification

AT+GSN Request product serial number identification (IMEI)

Phone control :

Command Description

AT+CBC Battery charge

AT+CGMI Request manufacturer identification

AT+CGMM Request model identification

AT+CGMR Request revision identification

AT+CGSN Request product serial number identification

AT+CMEE Report mobile equipment error

AT+CPAS Phone activity status

AT+CPBF Find phone book entries

AT+CPBR Read phone book entry

AT+CPBS Select phone book memory storage

AT+CPBW Write phone book entry

AT+CSCS Select TE character set

AT+CSQ Signal quality

Computer data interface :

Command Description

ATE Command Echo

ATQ Result code suppression

ATV Define response format

ATX Response range selection

AT&C Define DCD usage

AT&D Define DTR usage

AT&K Select flow control

AT&Q Define communications mode option

AT&S Define DSR option

AT+ICF DTE-DCE character framing

AT+IFC DTE-DCE Local flow control

AT+IPR Fixed DTE rate

Service :

Command Description

AT+CLIP Calling line identification presentation

AT+CR Service reporting control

AT+DR Data compression reporting

AT+ILRR DTE-DCE local rate reporting

Network Communication parameter :

Command Description

ATB Communications standard option

AT+CBST Select bearer service type

AT+CEER Extended error report

AT+CRLP Radio link protocol

AT+DS Data compression

Miscellaneous :

Command Description

A/ Re-execute command line

AT? Command help

AT\*C Start SMS interpreter

AT\*T Enter SMS block mode protocol

AT\*V Activate V.25bis mode

AT\*NOKIATEST Test command

AT+CESP Enter SMS block mode protocol

SMS Text mode :

Command Description

AT+CSMS Select message service

AT+CPMS Preferred message storage

AT+CMGF Message format

AT+CSCA Service centre address

AT+CSMP Set text mode parameters

AT+CSDH Show text mode parameters

AT+CSCB Select cell broadcast message types

AT+CSAS Save settings

AT+CRES Restore settings

AT+CNMI New message indications to TE

AT+CMGL List messages

AT+CMGR Read message

AT+CMGS Send message

AT+CMSS Send message from storage

AT+CMGW Write message to memory

AT+CMGD Delete message

SMS PDU mode :

Command Description

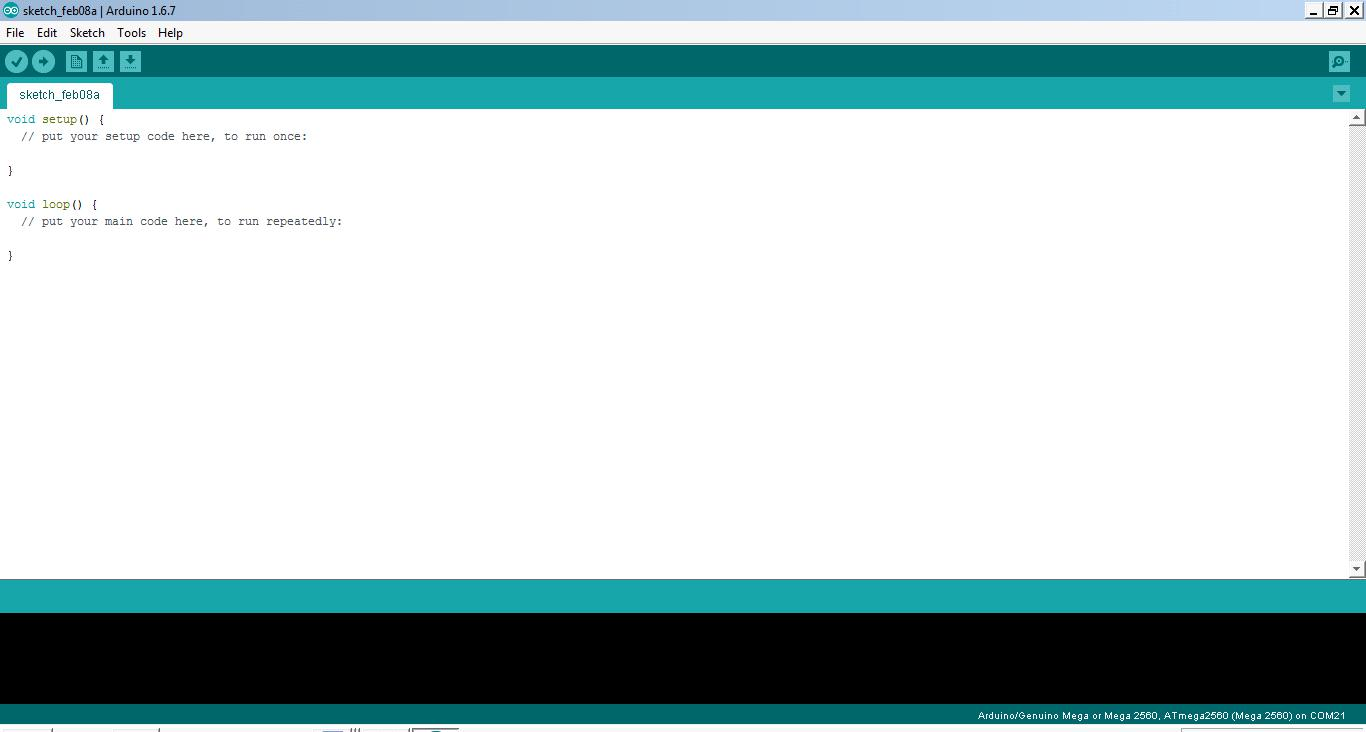
AT+CMGL List Messages

AT+CMGR Read message

AT+CMGS Send message

AT+CMGW Write message to memory

# 4.6 Arduino IDE

The Arduino IDE is incredibly minimalistic, yet it provides a near- complete environment for most Arduino-based projects. The top menu bar has the standard options, including “File” (new, load save, etc.), “Edit” (font, copy, paste, etc.), “Sketch” (for compiling and programming), “Tools” (useful options for testing projects), and “Help”. The middle section of the IDE is a simple text editor that where you can enter the program code. The bottom section of the IDE is dedicated to an output window that is used to see the status of the compilation, how much memory has been used, any errors that were found in the program, and various other useful messages.

## Fig.4.6 Arduino IDE page

Projects made using the Arduino are called sketches, and such sketches are usually written in a cut-down version of C++ (a number of C++ features are

not included). Because programming a micro controller is somewhat different from programming a computer, there are a number of device-specific libraries (e.g., changing pin modes, output data on pins, reading analog values, and timers). This sometimes confuses users who think Arduino is programmed in an “Arduino language.” However, the Arduino is, in fact, programmed in C++. It just uses unique libraries for the device.

To begin, download the Arduino IDE from [the Arduino website](http://arduino.cc/en/Main/Software). Make sure to select the right version for your Operating System (OS). For a full getting started guide for each OS, please refer to [the Arduino guide.](http://arduino.cc/en/Guide/HomePage) Once the arduino.zip file has been downloaded, extract the file to a folder somewhere on your computer. There is no install - simply open the folder and double click the .exe.

# Connecting the arduino

Connecting an Arduino board to your PC is quite simple. On Windows

* Plug in the USB cable - one end to the PC, and one end to the Arduino board.
* When prompted, select "Browse my computer for driver" and then select the folder to which you extracted your original Arduino IDE download.
* You may receive an error that the board is not a Microsoft certified device - select “Install anyway.”
* Your board should now be ready for programming.

When programming your Arduino board it is important to know what COM port the Arduino is using on your PC. On Windows, navigate to Start->Devices and Printers, and look for the Arduino. The COM port will be displayed underneath.

Alternatively, the message telling you that the Arduino has been connected successfully in the lower-left hand corner of your screen usually specifies the COM port is it using.

## Preparing the board

Before loading any code to your Arduino board, you must first open the IDE. Double click the Arduino.exe file that you downloaded earlier. A blank program, or "sketch," should open.

The Blink example is the easiest way to test any Arduino board. Within the Arduino window, it can be found under File->Examples->Basics->Blink.

Before the code can be uploaded to your board, two important steps are required.

1. Select your Arduino from the list under Tools->Board. The standard board used in RBE 1001, 2001, and 2002 is the Arduino Mega 2560, so select the "Arduino Mega 2560 or Mega ADK" option in the dropdown.
2. Select the communication port, or COM port, by going to Tools->Serial Port.

If you noted the COM port your Arduino board is using, it should be listed in the dropdown menu. If not, your board has not finished installing or needs to be reconnected.

## Loading the code:

The upper left of the Arduino window has two buttons: A checkmark to Verify your code, and a right-facing arrow to Upload it. Press the right arrow button to compile and upload the Blink example to your Arduino board.

The black bar at the bottom of the Arduino window is reserved for messages indicating the success or failure of code uploading. A "Completed Successfully" message should appear once the code is done uploading to your board. If an error

message appears instead, check that you selected the correct board and COM port in the Tools menu, and check your physical connections.

If uploaded successfully, the LED on your board should blink on/off once every second. Most Arduino boards have an LED prewired to pin 13.

It is very important that you do not use pins 0 or 1 while loading code. It is recommended that you do not use those pins ever.

Arduino code is loaded over a serial port to the controller. Older models use an [FTDI](http://www.ftdichip.com/) chip which deals with all the USB specifics. Newer models have either a small AVR that mimics the FTDI chip or a built-in USB-to-serial port on the AVR micro-controller itself.

# Embedded C programming

## Introduction to Embedded C programming

Looking around, we find ourselves to be surrounded by various types of [embedded systems](http://www.engineersgarage.com/articles/embedded-systems). Be it a digital camera or a mobile phone or a washing machine, all of them has some kind of processor functioning inside it. Associated with each processor is the embedded software. If hardware forms the body of an embedded system, embedded processor acts as the brain, and embedded software forms its soul. It is the embedded software which primarily governs the functioning of embedded systems.

During infancy years of microprocessor based systems, programs were developed using assemblers and fused into the EPROMs. There used to be no mechanism to find what the program was doing. LEDs, switches, etc. were used to check correct execution of the program. Some ‘very fortunate’ developers had In- circuit Simulators (ICEs), but they were too costly and were not quite reliable as well.

As time progressed, use of microprocessor-specific assembly-only as the programming language reduced and embedded systems moved onto C as the embedded programming language of choice. C is the most widely used

programming language for embedded processors/controllers. Assembly is also used but mainly to implement those portions of the code where very high timing accuracy, code size efficiency, etc. are prime requirements.

Initially C was developed by Kernighan and Ritchie to fit into the space of 8K and to write (portable) operating systems. Originally it was implemented on UNIX operating systems. As it was intended for operating systems development, it can manipulate memory addresses. Also, it allowed programmers to write very compact codes. This has given it the reputation as the language of choice for hackers too.

As assembly language programs are specific to a processor, assembly language didn’t offer portability across systems. To overcome this disadvantage, several high level languages, including C, came up. Some other languages like PLM, Modula-2, Pascal, etc. also came but couldn’t find wide acceptance. Amongst those, C got wide acceptance for not only embedded systems, but also for desktop applications. Even though C might have lost its sheen as mainstream language for general purpose applications, it still is having a strong-hold in embedded programming. Due to the wide acceptance of C in the embedded systems, various kinds of support tools like compilers & cross-compilers, ICE, etc. came up and all this facilitated development of embedded systems using C.

Subsequent sections will discuss what is Embedded C, features of C language, similarities and difference between C and embedded C, and features of embedded C programming.

## Embedded systems programming

Embedded systems programming is different from developing applications on a desktop computers. Key characteristics of an embedded system, when compared to PCs, are as follows:

* Embedded devices have resource constraints(limited ROM, limited RAM, limited stack space, less processing power)
* Components used in embedded system and PCs are different; embedded systems typically uses smaller, less power consuming components. Embedded systems are more tied to the hardware.

Two salient features of Embedded Programming are code speed and code size. Code speed is governed by the processing power, timing constraints, whereas code size is governed by available program memory and use of programming language. Goal of embedded system programming is to get maximum features in minimum space and minimum time.

Embedded systems are programmed using different type of languages:

1. Machine Code
2. Low level language, i.e., assembly
3. High level language like C, C++, Java, Ada, etc.
4. Application level language like Visual Basic, scripts, Access, etc.

Assembly language maps mnemonic words with the binary machine codes that the processor uses to code the instructions. Assembly language seems to be an obvious choice for programming embedded devices. However, use of assembly language is restricted to developing efficient codes in terms of size and speed. Also, assembly codes lead to higher software development costs and code portability is not there. Developing small codes are not much of a problem, but large programs/projects become increasingly difficult to manage in assembly language. Finding good assembly programmers has also become difficult nowadays. Hence high level languages are preferred for embedded systems programming.

## Difference between C programming and Embedded C programming

Though C and embedded C appear different and are used in different contexts, they have more similarities than the differences. Most of the constructs are same; the difference lies in their applications.

C is used for desktop computers, while embedded C is for microcontroller based applications. Accordingly, C has the luxury to use resources of a desktop PC like memory, OS, etc. While programming on desktop systems, we need not bother

about memory. However, embedded C has to use with the limited resources (RAM, ROM, I/Os) on an embedded processor. Thus, program code must fit into the available program memory. If code exceeds the limit, the system is likely to crash.

Compilers for C (ANSI C) typically generate OS dependant executables. Embedded C requires compilers to create files to be downloaded to the microcontrollers/microprocessors where it needs to run. Embedded compilers give access to all resources which is not provided in compilers for desktop computer applications.

Embedded systems often have the real-time constraints, which is usually not there with desktop computer applications.

Embedded systems often do not have a console, which is available in case of desktop applications.

So, what basically is different while programming with embedded C is the mindset; for embedded applications, we need to optimally use the resources, make the program code efficient, and satisfy real time constraints, if any. All this is done using the basic constructs, syntaxes, and function libraries of ‘C’.

# Programming using Embedded C

Embedded C use most of the syntax and semantics of standard C, e.g., main() function, variable definition, datatype declaration, conditional statements (if, switch. case), loops (while, for), functions, arrays and strings, structures and union, bit operations, macros, etc. In addition, there are some specifics to embedded C which are mentioned below:

Embedded programming requires access to underlying hardware, i.e., timers, memory, ports, etc. In addition, it is often needed to handle interrupts, manage job queues, etc. As C offers pointers and bit manipulation features, they are extensively used for direct hardware access.

## CHAPTER 5 RESULTS AND DISCUSSION

The final result of the project is as shown in the below figures. Each figure represents

each stage of the process.



## Fig 5.1 Completed hardware

The above fig shows the final prototype of the proposed system I.e., **AUTHORISED VOTING MACHINE.** The below figures are the images of the device at all the stages of the operation.



## Fig 5.2 when the system is ON

The above fig shows the stage of the device when it is in ON state. When the power supply is given to the device it will be as shown in the above fig. It is the initial stage of the device.



## Fig 5.3 Fingerprint scanning state

As the device works on the aadhar number and fingerprint, the above fig shows the fingerprint scanning state. In this stage fingerprint of the particular person will be scanned and will match with the data base.



## Fig 5.4 when authorized voter is found

After checking the fingerprint with the data base it will let us know whether the user is authorized or not. The above fig 5.1.4 shows that an authorized voter was found. So that, the person can cast his vote. The person can only cast his vote only if he/she is an authorized voter. If not so the person cannot cast his vote.



## Fig 5.5 unregistered voter

If the voter is not registered as a voter, then the device will display a message as fingerprint not found. The above fig 5.1.5 represents that the fingerprint of the particular person is not found. So that, the person cannot cast his vote.



## Fig 5.6 Number of votes received by each candidate

The above fig 5.6 shows the number of votes received by each candidate. By this we can know the total votes obtained for a candidate. The final winner can be displayed automatically after knowing the number of votes received by each candidate.



## Fig 5.7 The final result

The above fig 5.7 shows the final winner of the elections. There is no chance for a person to change the result as the result is shown clearly. The final winner will be displayed on the screen.

## APPENDIX

**CODE:**

#include<EEPROM.h>

#include<LiquidCrystal.h>

LiquidCrystal lcd(13,12,11,10,9,8);

#include <SoftwareSerial.h>

#include <Keypad.h>

SoftwareSerial mySerial(4, 5);

#include <Adafruit\_Fingerprint.h>

uint8\_t id;

Adafruit\_Fingerprint finger = Adafruit\_Fingerprint(&Serial);

#define enroll 26

#define del 27

#define up 28

#define down 29

#define match 31

#define indVote 6

#define sw1 32

#define sw2 33

#define sw3 35

#define resultsw 30

#define indFinger 7

#define buzzer 19

#define records 25

int vote1,vote2,vote3;

int flag;

char s[13];

int m=0,j=0,c=0,k,g=0,q;

char key;

const byte ROWS = 4; //four rows

const byte COLS = 4; //three columns

char keys[ROWS][COLS] = {

{'1','2','3','A'},

{'4','5','6','B' },

{'7','8','9','C'},

{'\*','0','#','D'}

};

byte rowPins[ROWS] = {40, 38, 36, 34}; //connect to the row pinouts of the keypad

byte colPins[COLS] = {48, 46, 44, 42}; //connect to the column pinouts of the keypad

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

String adhar[records];

void setup()

{

mySerial.begin(115200);

Serial.begin(57600);

delay(1000);

pinMode(enroll, INPUT\_PULLUP);

pinMode(up, INPUT\_PULLUP);

pinMode(down, INPUT\_PULLUP);

pinMode(del, INPUT\_PULLUP);

pinMode(match, INPUT\_PULLUP);

pinMode(sw1, INPUT\_PULLUP);

pinMode(sw2, INPUT\_PULLUP);

pinMode(sw3, INPUT\_PULLUP);

pinMode(resultsw, INPUT\_PULLUP);

pinMode(buzzer, OUTPUT);

pinMode(indVote, OUTPUT);

pinMode(indFinger, OUTPUT);

lcd.begin(16,2);

delay(1000);

lcd.clear();

lcd.print("Voting Machine");

// Serial.println("voting machine");

lcd.setCursor(0,1);

lcd.print("by Finger Print");

delay(2000);

lcd.clear();

lcd.print("using aadhar");

lcd.setCursor(0,1);

// lcd.print("Saddam Khan");

delay(2000);

if(EEPROM.read(0) == 0xff)

EEPROM.write(0,0);

if(EEPROM.read(1) == 0xff)

EEPROM.write(1,0);

if(EEPROM.read(2) == 0xff)

EEPROM.write(2,0);

//finger.begin(57600);

Serial.begin(57600);

lcd.clear();

lcd.print("Finding Module");

Serial.println("Finding Module");

lcd.setCursor(0,1);

delay(1000);

if (finger.verifyPassword())

{

//Serial.println("Found fingerprint sensor!");

lcd.clear();

lcd.print("Found Module ");

//Serial.println("Found Module ");

}

else

{

//Serial.println("Did not find fingerprint sensor :(");

lcd.clear();

lcd.print("module not Found");

Serial.println("module not Found");

lcd.setCursor(0,1);

lcd.print("Check Connections");

// Serial.println("Check Connections");

while (1);

}

if(digitalRead(resultsw) ==0)

{

for(int i=0;i<records;i++)

EEPROM.write(i+10,0xff);

EEPROM.write(0,0);

EEPROM.write(1,0);

EEPROM.write(2,0);

lcd.clear();

lcd.print("System Reset");

Serial.println("System Reset");

delay(1000);

}

if(digitalRead(del)==0)

{

for(int i=0;i<=records;i++)

{

deleteFingerprint(i) ;

delay(500);

}

}

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Cn1");

lcd.setCursor(4,0);

lcd.print("Cn2");

lcd.setCursor(9,0);

lcd.print("Cn3");

////lcd.setCursor(12,0);

// lcd.print("Cn4");

lcd.setCursor(1,1);

vote1=EEPROM.read(0);

lcd.print(vote1);

lcd.setCursor(5,1);

vote2=EEPROM.read(1);

lcd.print(vote2);

lcd.setCursor(10,1);

vote3=EEPROM.read(2);

lcd.print(vote3);

delay(3000);

}

void loop()

{

lcd.setCursor(0,0);

lcd.print("Press Match Key ");

Serial.println("press match key ");

lcd.setCursor(0,1);

lcd.print("to start system");

if(digitalRead(resultsw)==0)

{

seeres();

}

digitalWrite(indVote, LOW);

digitalWrite(indFinger, LOW);

if(digitalRead(match)==0)

{

digitalWrite(buzzer, HIGH);

delay(200);

digitalWrite(buzzer, LOW);

digitalWrite(indFinger, HIGH);

for(int i=0;i<3;i++)

{

lcd.clear();

lcd.print("Place Finger");

delay(2000);

int result=getFingerprintIDez();

if(result>=0)

{

flag=0;

for(int i=0;i<records;i++)

{

if(result == EEPROM.read(i+10))

{

lcd.clear();

lcd.print("Authorised Voter");

lcd.setCursor(0,1);

lcd.print("Please Wait....");

delay(1000);

lcd.clear();

lcd.print("Please Place");

lcd.setCursor(0,1);

lcd.print("Your Vote");

delay(2000);

lcd.clear();

lcd.print(adhar[result]);

lcd.setCursor(0,1);

lcd.print("voted for:");

delay(2000);

Vote();

for(i=0;i<records;i++)

{

if(result==EEPROM.read(i+10))

EEPROM.write(i+10, 0xff);

}

flag=1;

return;

}

}

if(flag == 0)

{

lcd.clear();

lcd.print(adhar[result]);

String g = adhar[result];

//delay(1000);

//lcd.clear();

//lcd.print(result);

lcd.setCursor(0,1);

lcd.print("Already Voted");

SendMessage(g);

//lcd.setCursor(0,1);

//lcd.print("")

digitalWrite(buzzer, HIGH);

delay(5000);

digitalWrite(buzzer, LOW);

return;

}

}

}

lcd.clear();

}

checkKeys();

delay(1000);

}

void checkKeys()

{

if(digitalRead(enroll) == 0)

{

lcd.clear();

lcd.print("Please Wait");

delay(1000);

while(digitalRead(enroll) == 0);

Enroll();

}

else if(digitalRead(del) == 0)

{

lcd.clear();

lcd.print("Please Wait");

delay(1000);

delet();

}

}

void Enroll()

{

int count=0;

lcd.clear();

lcd.print("Enter Finger ID:");

while(1)

{

lcd.setCursor(0,1);

lcd.print(count);

if(digitalRead(up) == 0)

{

count++;

if(count>25)

count=0;

delay(500);

}

else if(digitalRead(down) == 0)

{

count--;

if(count<0)

count=25;

delay(500);

}

else if(digitalRead(del) == 0)

{

id=count;

getFingerprintEnroll();

for(int i=0;i<records;i++)

{

if(EEPROM.read(i+10) == 0xff)

{

EEPROM.write(i+10, id);

break;

}

}

return;

}

else if(digitalRead(enroll) == 0)

{

return;

}

}

}

void delet()

{

int count=0;

lcd.clear();

lcd.print("Enter Finger ID");

while(1)

{

lcd.setCursor(0,1);

lcd.print(count);

if(digitalRead(up) == 0)

{

count++;

if(count>25)

count=0;

delay(500);

}

else if(digitalRead(down) == 0)

{

count--;

if(count<0)

count=25;

delay(500);

}

else if(digitalRead(del) == 0)

{

id=count;

deleteFingerprint(id);

for(int i=0;i<records;i++)

{

if(EEPROM.read(i+10) == id)

{

EEPROM.write(i+10, 0xff);

break;

}

}

return;

}

else if(digitalRead(enroll) == 0)

{

return;

}

}

}

uint8\_t getFingerprintEnroll()

{

int id1=id;

int p = -1;

lcd.clear();

lcd.print("finger ID:");

lcd.print(id);

lcd.setCursor(0,1);

lcd.print("Place Finger");

delay(2000);

while (p != FINGERPRINT\_OK)

{

p = finger.getImage();

switch (p)

{

case FINGERPRINT\_OK:

//Serial.println("Image taken");

lcd.clear();

lcd.print("Image taken");

break;

case FINGERPRINT\_NOFINGER:

//Serial.println("No Finger");

lcd.clear();

lcd.print("No Finger");

break;

case FINGERPRINT\_PACKETRECIEVEERR:

//Serial.println("Communication error");

lcd.clear();

lcd.print("Comm Error");

break;

case FINGERPRINT\_IMAGEFAIL:

//Serial.println("Imaging error");

lcd.clear();

lcd.print("Imaging Error");

break;

default:

//Serial.println("Unknown error");

lcd.clear();

lcd.print("Unknown Error");

break;

}

}

// OK success!

p = finger.image2Tz(1);

switch (p) {

case FINGERPRINT\_OK:

//Serial.println("Image converted");

lcd.clear();

lcd.print("Image converted");

break;

case FINGERPRINT\_IMAGEMESS:

//Serial.println("Image too messy");

lcd.clear();

lcd.print("Image too messy");

return p;

case FINGERPRINT\_PACKETRECIEVEERR:

//Serial.println("Communication error");

lcd.clear();

lcd.print("Comm Error");

return p;

case FINGERPRINT\_FEATUREFAIL:

//Serial.println("Could not find fingerprint features");

lcd.clear();

lcd.print("Feature Not Found");

return p;

case FINGERPRINT\_INVALIDIMAGE:

//Serial.println("Could not find fingerprint features");

lcd.clear();

lcd.print("Feature Not Found");

return p;

default:

//Serial.println("Unknown error");

lcd.clear();

lcd.print("Unknown Error");

return p;

}

//Serial.println("Remove finger");

lcd.clear();

lcd.print("Remove Finger");

delay(2000);

p = 0;

while (p != FINGERPRINT\_NOFINGER) {

p = finger.getImage();

}

//Serial.print("ID "); //Serial.println(id);

p = -1;

//Serial.println("Place same finger again");

lcd.clear();

lcd.print("Place Finger");

lcd.setCursor(0,1);

lcd.print(" Again");

while (p != FINGERPRINT\_OK) {

p = finger.getImage();

switch (p) {

case FINGERPRINT\_OK:

//Serial.println("Image taken");

break;

case FINGERPRINT\_NOFINGER:

//Serial.print(".");

break;

case FINGERPRINT\_PACKETRECIEVEERR:

//Serial.println("Communication error");

break;

case FINGERPRINT\_IMAGEFAIL:

//Serial.println("Imaging error");

break;

default:

//Serial.println("Unknown error");

return;

}

}

// OK success!

p = finger.image2Tz(2);

switch (p) {

case FINGERPRINT\_OK:

//Serial.println("Image converted");

break;

case FINGERPRINT\_IMAGEMESS:

//Serial.println("Image too messy");

return p;

case FINGERPRINT\_PACKETRECIEVEERR:

//Serial.println("Communication error");

return p;

case FINGERPRINT\_FEATUREFAIL:

//Serial.println("Could not find fingerprint features");

return p;

case FINGERPRINT\_INVALIDIMAGE:

//Serial.println("Could not find fingerprint features");

return p;

default:

//Serial.println("Unknown error");

return p;

}

// OK converted!

//Serial.print("Creating model for #"); //Serial.println(id);

p = finger.createModel();

if (p == FINGERPRINT\_OK) {

//Serial.println("Prints matched!");

} else if (p == FINGERPRINT\_PACKETRECIEVEERR) {

//Serial.println("Communication error");

return p;

} else if (p == FINGERPRINT\_ENROLLMISMATCH) {

//Serial.println("Fingerprints did not match");

return p;

} else {

//Serial.println("Unknown error");

return p;

}

//Serial.print("ID "); //Serial.println(id);

p = finger.storeModel(id);

if (p == FINGERPRINT\_OK) {

//Serial.println("Stored!");

lcd.clear();

lcd.print("Stored!");

delay(1000);

lcd.clear();

gtptadno(id);

delay(3000);

} else if (p == FINGERPRINT\_PACKETRECIEVEERR) {

//Serial.println("Communication error");

return p;

} else if (p == FINGERPRINT\_BADLOCATION) {

//Serial.println("Could not store in that location");

return p;

} else if (p == FINGERPRINT\_FLASHERR) {

//Serial.println("Error writing to flash");

return p;

}

else {

//Serial.println("Unknown error");

return p;

}

}

int getFingerprintIDez()

{

uint8\_t p = finger.getImage();

if (p != FINGERPRINT\_OK)

return -1;

p = finger.image2Tz();

if (p != FINGERPRINT\_OK)

return -1;

p = finger.fingerFastSearch();

if (p != FINGERPRINT\_OK)

{

lcd.clear();

lcd.print("Finger Not Found");

lcd.setCursor(0,1);

lcd.print("Try Later");

delay(2000);

return -1;

}

// found a match!

//Serial.print("Found ID #");

//Serial.print(finger.fingerID);

return finger.fingerID;

}

uint8\_t deleteFingerprint(uint8\_t id)

{

uint8\_t p = -1;

lcd.clear();

lcd.print("Please wait");

p = finger.deleteModel(id);

if (p == FINGERPRINT\_OK)

{

//Serial.println("Deleted!");

lcd.clear();

lcd.print("Figer Deleted");

lcd.setCursor(0,1);

lcd.print("Successfully");

delay(1000);

}

else

{

//Serial.print("Something Wrong");

lcd.clear();

lcd.print("Something Wrong");

lcd.setCursor(0,1);

lcd.print("Try Again Later");

delay(2000);

return p;

}

}

void Vote()

{

digitalWrite(indVote, HIGH);

digitalWrite(indFinger, LOW);

digitalWrite(buzzer, HIGH);

delay(500);

digitalWrite(buzzer, LOW);

delay(1000);

while(1)

{

if(digitalRead(sw1)==0)

{

vote1++;

voteSubmit(1);

EEPROM.write(0, vote1);

while(digitalRead(sw1)==0);

return;

}

if(digitalRead(sw2)==0)

{

vote2++;

voteSubmit(2);

EEPROM.write(1, vote2);

while(digitalRead(sw2)==0);

return;

}

if(digitalRead(sw3)==0)

{

vote3++;

voteSubmit(3);

EEPROM.write(2, vote3);

while(digitalRead(sw3)==0);

return;

}

}

digitalWrite(indVote, LOW);

}

void voteSubmit(int cn)

{

if(cn == 1)

{

lcd.clear();

lcd.setCursor(5,0);

lcd.print("Can1");

delay(1000);

}

else if(cn == 2)

{

lcd.clear();

lcd.setCursor(5,0);

lcd.print("Can2");

delay(1000);

}

else if(cn == 3)

{

lcd.clear();

lcd.setCursor(5,0);

lcd.print("Can3");

delay(1000);

}

lcd.clear();

//lcd.setCursor(0,1);

lcd.print("Thank You");

lcd.setCursor(0,3);

lcd.print("For Voting");

digitalWrite(buzzer , HIGH);

delay(1000);

digitalWrite(buzzer, LOW);

digitalWrite(indVote, LOW);

return;

}

void gtptadno(uint8\_t id1)

{

lcd.print("Enter adhar:");

lcd.setCursor(0,1);

lcd.print("No & press \*");

delay(2000);

lcd.clear();

lcd.print("you entered:");

lcd.setCursor(0,1);

while(m<13)

{

key = keypad.getKey();

if(key=='\*')

{

//Serial.println("you entered");

adhar[id1]=s;

//Serial.print(s);

Serial.print(adhar[id1]);

//lcd.print(adhar[id1]);

// j++;

c++;

m=0;

g=0;

return;

//Serial.print("\n");

//erial.print("enter ur no. nd pres \*");

}

else if(key != NO\_KEY and key!='\*')

{

lcd.print(key);

s[m]=key;

m++;

}

}

}

void seeres()

{

while(1)

{

lcd.clear();

lcd.print("Please Wait");

delay(1000);

results();

return;

}

}

void results()

{

lcd.clear();

lcd.setCursor(0,0);

lcd.print("Can1");

lcd.setCursor(6,0);

lcd.print("Can2");

lcd.setCursor(12,0);

lcd.print("Can3");

for(int i=0;i<3;i++)

{

lcd.setCursor(i\*6,1);

lcd.print(EEPROM.read(i));

}

delay(3000);

int vote=vote1+vote2+vote3;

if(vote)

{

if((vote1 > vote2 && vote1 > vote3))

{

lcd.clear();

lcd.print("Can1 Wins");

delay(2500);

lcd.clear();

}

else if(vote2 > vote1 && vote2 > vote3)

{

lcd.clear();

lcd.print("Can2 Wins");

delay(2500);

lcd.clear();

}

else if((vote3 > vote1 && vote3 > vote2))

{

lcd.clear();

lcd.print("Can3 Wins");

delay(2500);

lcd.clear();

}

else

{

lcd.clear();

lcd.print(" Tie Up Or ");

lcd.setCursor(0,1);

lcd.print(" No Result ");

delay(2500);

lcd.clear();

}

}

else

{

lcd.clear();

lcd.print("No Voting....");

delay(2500);

lcd.clear();

}

vote1=0;vote2=0;vote3=0;vote=0;

lcd.clear();

return ;

}

void SendMessage(String gp)

{

mySerial.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode

delay(1000); // Delay of 1000 milli seconds or 1 second

mySerial.println("AT+CMGS=\"+919949091946\"\r"); // Replace x with mobile number

delay(1000);

mySerial.println("Aadhar number: " + gp + " Trying to vote second time");// The SMS text you want to send

delay(100);

mySerial.println((char)26);// ASCII code of CTRL+Z

delay(1000);

}