Name: Vishwa B

Registration Number: 22BCT0028

Fingerprint Based Biometric Voting Machine using Arduino

Presentation of the model:

https://docs.google.com/presentation/d/16oSYhQgAA2_kou0Aik4kTfqJ9hdkRKI4u0xZGOrRVC0/edit?usp=drivesdk

Video of working model:

https://drive.google.com/file/d/1BEjhMa93VHu7J_cNE-f-HDKk0VoGFliJ/view?usp=drivesdk

Fingerprint-Based Biometric Voting Machine Using Arduino

Electronic Voting Machines (EVMs) are an essential part of modern election systems, enabling votes to be registered electronically without the need for traditional ballot paper. While these machines improve efficiency, they are still vulnerable to issues like voter impersonation, fake voting, and unauthorized multiple votes. To address these security concerns, integrating **fingerprint-based biometric authentication** into EVMs offers a robust solution.

Objectives of the Fingerprint-Based Biometric Voting Machine

The primary aim of the **Fingerprint-Based Biometric Voting Machine** is to enhance the security and reliability of the voting process using biometric authentication. Below are the detailed objectives:

1. **Ensure Secure Voter Authentication**
   - Implement fingerprint-based biometric authentication to verify voter identity.
   - Prevent unauthorized individuals from accessing the voting system.
2. **Eliminate Fake and Duplicate Voting**
   - Restrict each voter to casting a single vote by ensuring that once a fingerprint is authenticated, it cannot be reused.
   - Block attempts of proxy or fake voting by relying on unique biometric data.
3. **Improve Election Integrity**
   - Provide a tamper-proof system that guarantees only valid votes are counted.
   - Store data securely to prevent manipulation or unauthorized access to the results.
4. **Simplify the Voting Process**
   - Create a user-friendly interface that enables voters to cast their votes quickly and efficiently.
   - Reduce the need for manual oversight by election officials.

5. **Enable Real-Time Vote Tracking** ○ Record and store votes electronically in real-time.
    - ○ Provide instant or on-demand vote tallying for transparency.
6. **Promote Scalability and Adaptability**
    - ○ Design the system to handle a varying number of voters, from small-scale elections (e.g., schools and organizations) to large-scale governmental elections.
    - ○ Allow for future integration with additional biometric methods like facial recognition or iris scanning.
7. **Reduce Paper Usage and Manual Errors** ○ Transition from traditional paper-based voting to an eco-friendly electronic system.
    - ○ Minimize human errors in the voting and counting processes.

Required Components:

1. Arduino Uno
2. Finger Print Sensor Module
3. Push Buttons
4. LEDs -2
5. 1K Resistor -3
6. 2.2K resistor
7. Power
8. Connecting wires
9. Buzzer
10. 16x2 LCD
11. Bread Board

Main Components and Connections

1. **Arduino (Uno/Mega)** ○ Acts as the central microcontroller to process data and control all components.
    - ○ Connected to all peripherals, including the fingerprint sensor, LCD, pushbuttons, LEDs, and buzzer.

2. **Fingerprint Sensor**

- **Pins**: TX and RX pins of the sensor connect to Arduino's digital pins for serial communication.
- **Function**: Captures and authenticates voter fingerprints.

3. **LCD Display (16x2)** ○ **Pins**: Data pins connect to Arduino (Digital or I2C communication).

- **Function**: Displays voting instructions, candidate lists, and results.

4. **Pushbuttons**

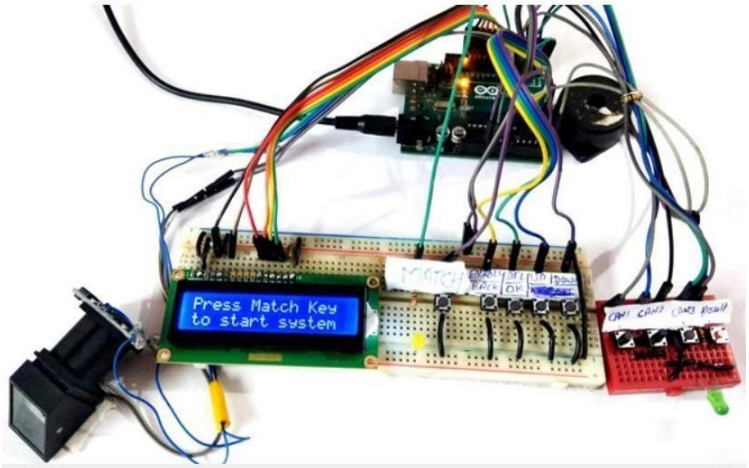
- **Pins**: Each button is connected to a digital input pin of the Arduino, with pull-down resistors to avoid noise.
- **Function**: Used to navigate and select candidates.

5. **LEDs** ○ **Pins**: Connected to digital pins via resistors. ○ **Function**: Indicate system states (e.g., authentication success, voting completed). 6. **Buzzer** ○ **Pins**: Connected to a digital pin of Arduino.

- **Function**: Provides audio feedback for successful or failed operations.

7. **Power Supply Module** ○ Provides 5V and 3.3V power to the components.

8. **Memory Storage** ○ **EEPROM/SD Card**: Stores vote data securely to prevent loss.

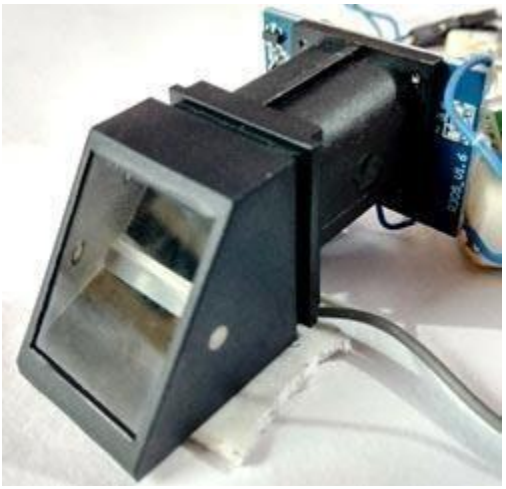


Finger Print Based Biometric Voting Machine using Arduino

Finger Print Sensor Module in Voting Machine:

Finger Print Sensor Module or Finger Print Scanner is a module which captures finger's print image and then converts it into the equivalent template and saves them into its memory on selected ID (location) by Arduino.

Here all the process is commanded by Arduino like taking an image of finger print, convert it into templates and storing location etc.



In this **FingerPrint Voting Machine Circuit**, we have used Finger Print Sensor Module to authenticate true voter by taking their finger input in the system. Here we are using 5 push buttons to Match, Enroll/back, Delete/OK, UP and Down.

Enroll and Del key have double features here. **Enroll key** is used for enrolling new finger impression into the system and back function as well.

Means when the user wants to enroll new finger then he/she needs to press enroll key then LCD asks for the ID or Location where user wants to store the finger print output.

Now if at this time user do not want to proceed further then he/she can press enroll key again to go back (this time enroll key behave as Back key).

Means enroll key has both enrollment and back function. **DEL/OK key** also has same double function like when user enrolls new finger then he/she need to select finger ID or Location by using another two key namely UP AND DOWN now user needs to press DEL/OK key (this time this key behaves like OK) to proceed with selected ID or Location.

Match key is used for whenever voter wants to vote then he/she needs to authenticate first for true voter by keeping finger on Finger Print Sensor, if he/she passed in this authentication then he/she can vote.

Basic Steps Involved in the Project

1. **System Setup and Component Integration**
    - Connect the **Arduino** to the required components, including the **fingerprint sensor**, **LCD display**, **pushbuttons**, **LEDs**, and **buzzer**.
    - Ensure proper power supply and establish circuit connections.

2. **Fingerprint Enrollment**
    - Use the fingerprint sensor to enroll fingerprints of all eligible voters before the election.
    - Store the enrolled fingerprints in the system's database with unique IDs.
3. **Authentication Process** ○ On the election day, the voter scans their fingerprint using the fingerprint sensor.
    - The system checks the scanned fingerprint against the pre-registered database.
4. **Voting Interface** ○ If the fingerprint matches, the LCD displays the list of candidates. ○ The voter selects their choice using pushbuttons.
5. **Vote Storage**
    - Once a vote is cast, the system securely stores it in non-volatile memory (e.g., EEPROM or SD card).
    - The voter's fingerprint is marked as "voted" to prevent duplicate votes.
6. **Result Display**
    - After voting concludes, the system tallies the votes and displays the results on the LCD or sends them to a computer for analysis.
7. **Testing and Verification**
    - Test the system thoroughly to ensure fingerprint authentication accuracy and proper vote recording.
    - Verify the results to confirm system reliability and data integrity.

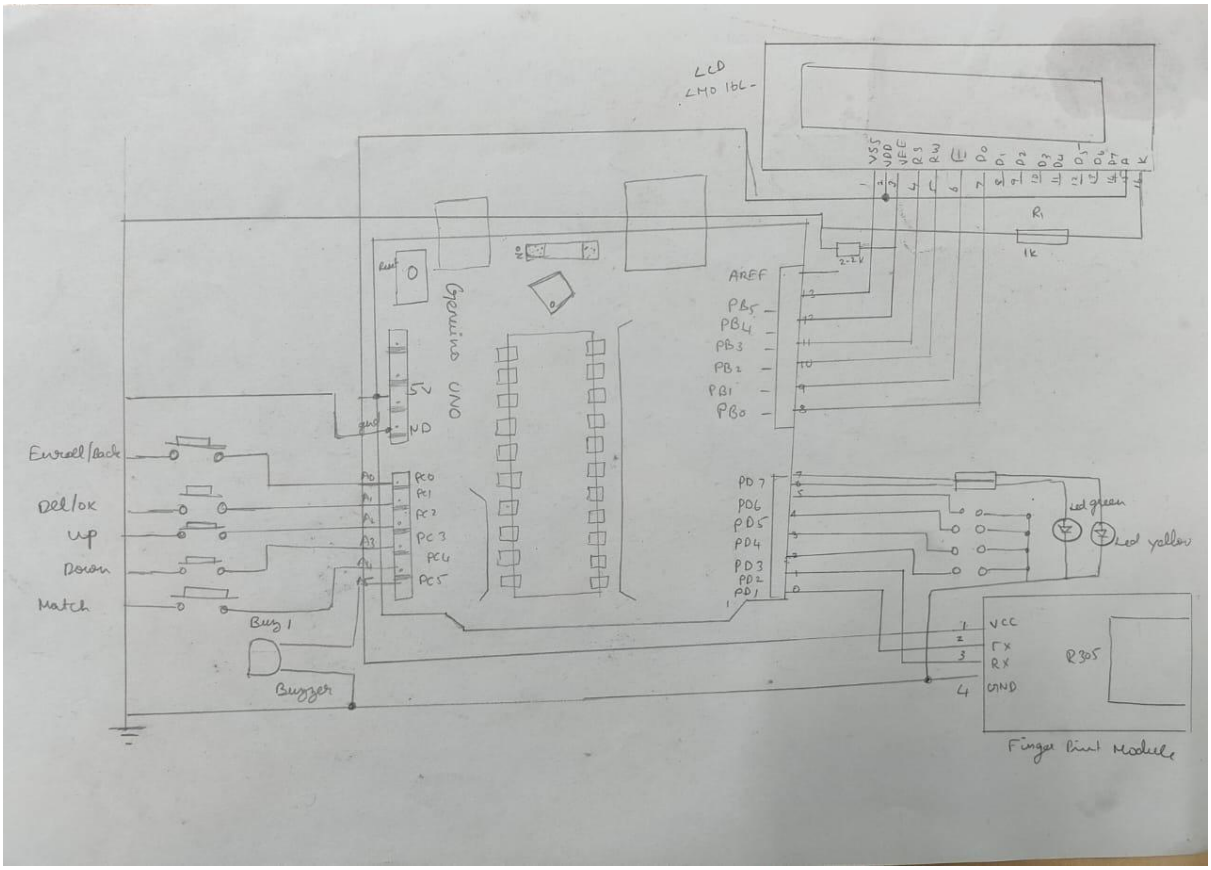
Circuit Explanation:

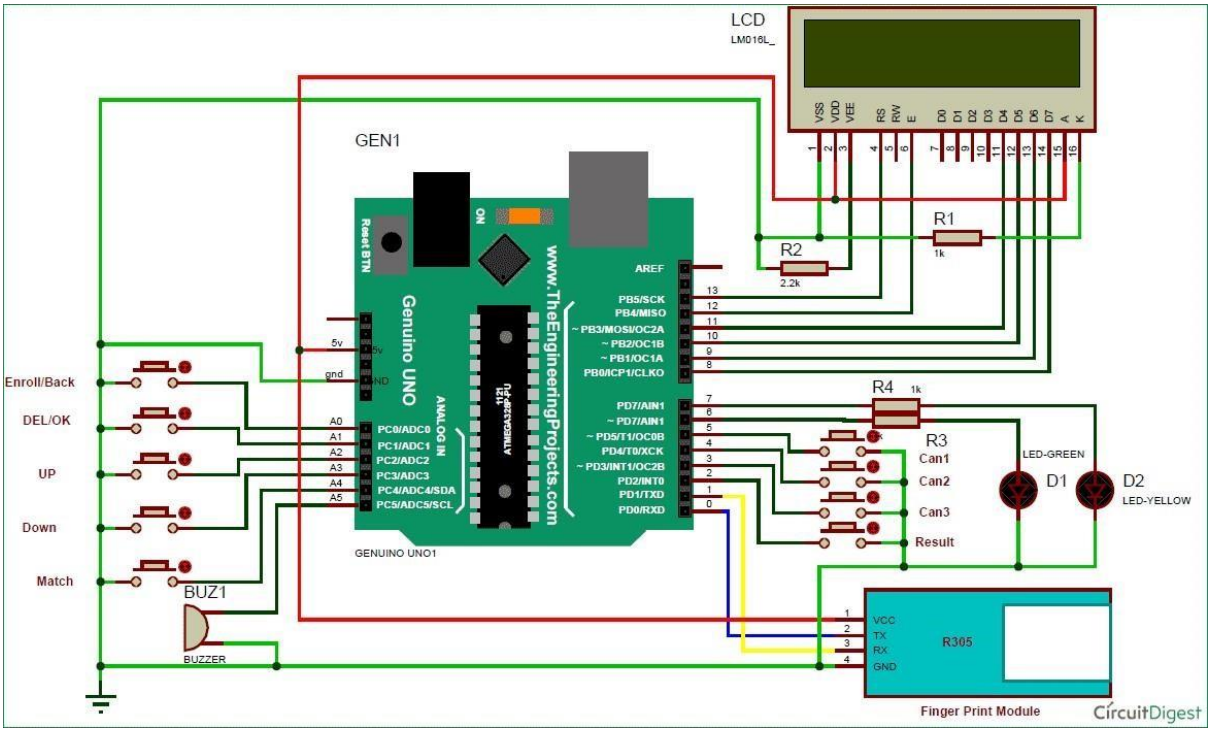
Circuit Schematic Description

- The **fingerprint sensor** is connected to Arduino's **digital pins TX/RX** for serial communication.
- The **LCD display** is connected via digital or I2C pins for displaying messages.
- **Pushbuttons** are connected to input pins with pull-down resistors to ensure stable signals.
- **LEDs** and the **buzzer** are connected to digital output pins to indicate system status and events.
- The **power supply module** provides the necessary voltage for the components.

The circuit of this **FingerPrint Based Voting Machine Project** is very simple which contains Arduino for controlling whole the process of the project, push button for enrolling, deleting, selecting IDs and voting purpose, a buzzer for alert, LEDs for indication and 16x2 LCD for instruct Voter and showing

the result as well. Yellow LED indicates that fingerprint module is ready to take an image of the finger and Green LED indicates that system is ready to take a vote or see results.





The push button is directly connected to pin A0(ENROL), A1(DEL), A2(UP), A3(DOWN) and A4(Match), D5(Can1), D4(Can2), D3(Can3), D2(Result) of Arduino with respect to ground. Yellow LED is connected at Digital pin D7 of Arduino with respect to ground through a 1k resistor and Green LED is connected to D6 of Arduino with the same method. Fingerprint module's Rx and Tx directly

connected at Serial pin Tx and Rx of Arduino. 5v supply is used for powering finger print module taken from Arduino board. A buzzer is also connected at A5. A 16x2 LCD is configured in 4-bit mode and its RS, EN, D4, D5, D6, and D7 are directly connected at Digital pin D13, D12, D11, D10, D9, and D8 of Arduino.

Functionalities of the System

1. Fingerprint Authentication

- **Description**: The system uses a fingerprint sensor to authenticate voters by comparing their fingerprints with the pre-registered database.
- **Purpose**: To ensure only authorized voters can access the system and prevent unauthorized or fake voting.

2. Candidate Selection Interface

- **Description**: After successful authentication, the system displays the list of candidates on an LCD screen.
- **Purpose**: To provide an easy and user-friendly way for voters to select their desired candidate using buttons or touch input.

3. Vote Recording

- **Description**: The system securely records the voter's choice in non-volatile memory (EEPROM or SD card).
- **Purpose**: To store votes in a tamper-proof manner for result tallying later.

4. Duplicate Voting Prevention

- **Description**: Once a fingerprint is authenticated and a vote is cast, it is flagged in the system to prevent the same individual from voting again.
- **Purpose**: To ensure that each voter can cast only one vote.

5. Result Tallying and Display

- **Description**: After voting ends, the system calculates the total votes for each candidate and displays the results on the LCD or exports them to a computer.
- **Purpose**: To provide a quick and accurate way to view the election outcome.

6. Error Feedback

- **Description**: The system uses LEDs and a buzzer to provide feedback in cases like authentication failure or invalid operations.
- **Purpose**: To improve user experience and troubleshooting during the voting process.

Code for functionality :

```
#include<EEPROM.h>
#include<LiquidCrystal.h>
LiquidCrystal lcd(13,12,11,10,9,8); #include
<Adafruit_Fingerprint.h> uint8_t
id;
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&Serial);
#define enroll 14
#define del 15
#define up 16
#define down 17
#define match 18
#define indVote 6
#define sw1 5
#define sw2 4
#define sw3 3
#define resultsw 2
#define indFinger 7
#define buzzer 19 #define
records 25 int
vote1,vote2,vote3;
int flag; void setup()
{
   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);   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");       delay(1000);
 }
   lcd.clear();
   lcd.print("Voting Machine");
lcd.setCursor(0,1);    lcd.print("by Finger
Print");    delay(2000);
   lcd.clear();
lcd.print("Circuit Digest");
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(1) == 0xff)
EEPROM.write(1,0);
   //finger.begin(57600);    Serial.begin(57600);
```

```
   lcd.clear();
lcd.print("Finding Module");
lcd.setCursor(0,1);     delay(1000);
if
(finger.verifyPassword())
   {
     //Serial.println("Found fingerprint sensor!");
     lcd.clear();
lcd.print("Found Module ");       delay(1000);
   }
   else
   {
   //Serial.println("Did not find fingerprint sensor :(");
lcd.clear();     lcd.print("module not Found");
lcd.setCursor(0,1);     lcd.print("Check Connections");
   while (1);
   }
 lcd.clear();
lcd.setCursor(0,0);
lcd.print("Cn1"); lcd.setCursor(4,0);
lcd.print("Cn2");
lcd.setCursor(8,0);
lcd.print("Cn3");
lcd.setCursor(12,0);
lcd.print("Cn4");
lcd.setCursor(0,1);
  vote1=EEPROM.read(0);
  lcd.print(vote1);   lcd.setCursor(6,1);
vote2=EEPROM.read(1);
lcd.print(vote2);   lcd.setCursor(12,1);
vote3=EEPROM.read(2);
```

```
lcd.print(vote3);   delay(2000);
}
void loop()
{
 lcd.setCursor(0,0);
lcd.print("Press  Match  Key  ");
lcd.setCursor(0,1);   lcd.print("to
start                    system");
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);
            Vote();
            EEPROM.write(i+10, 0xff);
            flag=1;
return;
         }
       }
       if(flag == 0)
       {
       lcd.clear();
       lcd.print("Already Voted");
//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 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(2000);
 } 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()  {
lcd.clear();
lcd.print("Please        Place");
lcd.setCursor(0,1);
lcd.print("Your           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;
      }
       if(digitalRead(resultsw)==0)
       {
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(2000);          int
vote=vote1+vote2+vote3;
        if(vote)
        {
         if((vote1 > vote2 && vote1 > vote3))
         {
          lcd.clear();
          lcd.print("Can1 Wins");            delay(2000);
lcd.clear();
         }
         else if(vote2 > vote1 && vote2 > vote3)
         {
          lcd.clear();
lcd.print("Can2 Wins");
delay(2000);            lcd.clear();
         }
         else if((vote3 > vote1 && vote3 > vote2))
         {
lcd.clear();
lcd.print("Can3 Wins");
delay(2000);            lcd.clear();
         }
        else
        {           lcd.clear();
lcd.print("   Tie Up Or   ");
lcd.setCursor(0,1);            lcd.print("
No Result   ");            delay(1000);
lcd.clear();
```

```
        }
       }
        else
{
          lcd.clear();
lcd.print("No Voting....");          delay(1000);
lcd.clear();
        }
        vote1=0;vote2=0;vote3=0;vote=0;          lcd.clear();
return;
      }
 }
  digitalWrite(indVote, LOW);
}
void voteSubmit(int cn)
{  lcd.clear();  if(cn == 1)
lcd.print("Can1");  else if(cn ==
2)     lcd.print("Can2");  else
if(cn == 3)
lcd.print("Can3");
lcd.setCursor(0,1);
lcd.print("Vote Submitted");
digitalWrite(buzzer , HIGH);
delay(1000);  digitalWrite(buzzer,
LOW);  digitalWrite(indVote,
LOW);
 return;
}
```

Conclusion:

The **Fingerprint-Based Biometric Voting Machine** offers a secure, efficient solution to modernize voting systems. By using fingerprint authentication, it eliminates issues like voter impersonation and duplicate voting, ensuring that only eligible individuals can cast their vote. The system is userfriendly, with a simple interface and quick authentication, making the voting process faster and more transparent. It provides real-time results and can be easily scaled for elections of any size. This system not only enhances election security but also paves the way for future improvements with additional biometric technologies.