A PROJECT REPORT ON

ELECTRONIC VOTING MACHINE USING FINGERPRINT SENSOR ON ARDUINO

SUBMITTED BY

SURYANSH BANSAL (23/EC/209)

VAGISH KAUSHIK (24/A14/044)

VISHESH NANDA (24/A14/054)

DELHI TECHNOLOGICAL UNIVERSITY

(FORMERLY DELHI COLLEGE OF ENGINEERING)

BAWANA ROAD, DELHI – 110042

NOVEMBER 2024

CANDIDATE'S DECLARATION

We, **Suryansh Bansal (23/EC/209)**, **VAGISH KAUSHIK (24/A14/044) And VISHESH NANDA (24/A14/054)** students of **1st year** B. TECH ECE declare that the Project Report titled "**ELECTRONIC VOTING MACHINE USING ARDUINO**" which is submitted by me to Department of Electronics and Communication, Delhi Technological University, Delhi, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma, Fellowship or other similar title or recognition.

Place: DTU, Delhi

Date: 08/11/2024

ACKNOWLEDGMENT

We are very thankful to Mr. Ajay Kumar Dwivedi, all faculty members, and the lab staff, of the ECE Department at DTU, for their invaluable support and guidance throughout this project. Their insights and encouragement were instrumental in completing our work. We are also grateful to the university for providing the laboratories, infrastructure, test facilities, and an environment conducive to research and learning. Additionally, we would like to thank our seniors and peers for sharing their knowledge and advice, which greatly contributed to our project.

Suryansh Bansal **(23/EC/209)**

Vagish Kaushik **(24/A14/044)**

Vishesh Nanda **(24/A14/054)**

Mr. Ajay Kumar Dwivedi

(Supervisor)

INDEX

INTRODUCTION

ABSTRACT

This project report documents the development of an Arduino-based electronic voting machine (EVM) aimed at enhancing the accuracy, security, and efficiency of the voting process. By utilizing an Arduino microcontroller, the EVM facilitates a smooth voting experience through a clear interface, with an LCD display that provides voter instructions and buttons for candidate selection. Each vote is recorded in real time to ensure accuracy and prevent duplicate entries. The system operates on a simple yet robust logic that manages voter interaction, secure vote storage, and results display. Designed as a prototype for small-scale use, this EVM presents a cost-effective, user-friendly solution that can be expanded for broader applications. By demonstrating effective data handling and intuitive design, the project underscores the potential for low-cost electronic systems in transforming traditional voting practices into more modern, accessible formats.

INTRODUCTION TO ELECTRONIC VOTING MACHINES

Electronic Voting Machines (EVMs) offer a modern alternative to traditional paper-based voting. Traditional systems require significant human resources, are time-intensive, and are vulnerable to errors during vote counting and handling. In contrast, EVMs streamline the voting process by digitizing vote casting and counting, reducing the need for paper ballots and manual counting. This digital approach enhances accuracy, speeds up result processing, and minimizes risks of human error and fraud. With appropriate safeguards, EVMs ensure a secure, efficient, and accessible voting experience for all participants.

PROJECT SCOPE AND RELEVANCE

Secure voting is fundamental to a democratic society, and modern EVMs play a crucial role in achieving this goal. This project demonstrates a simplified prototype for electronic voting using an Arduino-based system, highlighting its potential for secure, real-time vote tallying without requiring extensive infrastructure. Designed as an accessible and user-friendly solution, this project underscores the relevance of digital voting solutions, particularly for small-scale

elections or educational settings. It provides a foundation for developing more sophisticated systems with enhanced security and usability in future electronic voting applications.

OBJECTIVES

1. Develop a cost-effective electronic voting system using Arduino.
2. Develop a fingerprint-enabled EVM for secure voter identification.
3. Prevent duplicate and unauthorized voting.
4. Ensure an accurate and secure vote-casting process.
5. Prevent duplicate voting and unauthorized access.
6. Provide clear, user-friendly instructions via an LCD display.
7. Enable real-time vote tallying and result display.
8. Demonstrate the feasibility of small-scale electronic voting for educational or prototype applications.
9. Highlight the potential of Arduino in creating accessible electronic voting solutions.

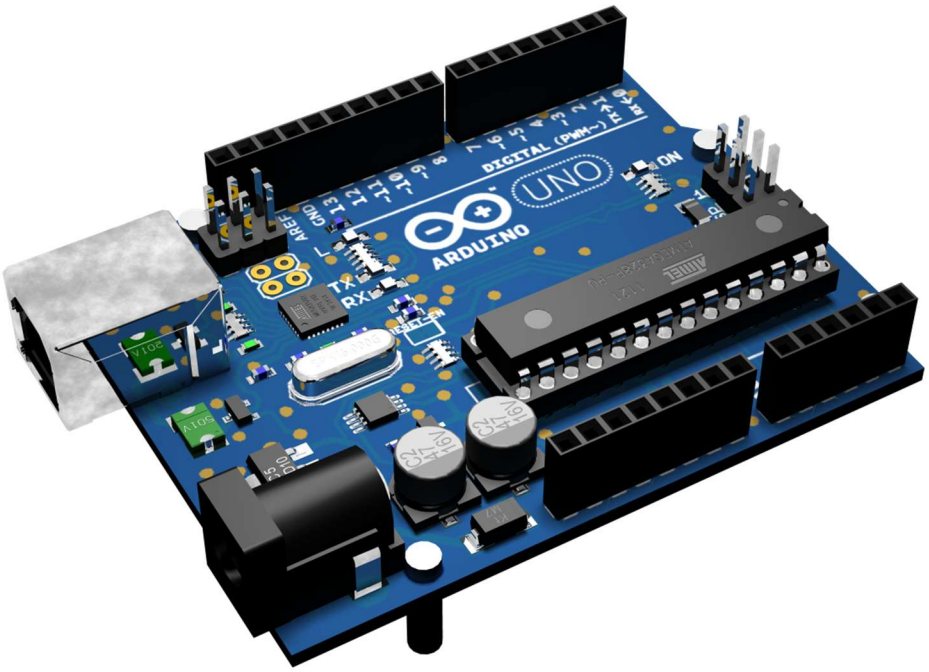
MATERIALS REQUIRED

1. **Arduino Uno**: Microcontroller used to control voting operations.
2. **I2C LCD Display**: Used to display instructions and feedback to the user.
3. **3 Push Buttons**: For selecting candidates during voting.
4. **Buzzer**: Alerts or confirms voting actions.
5. **Breadboard**: For circuit assembly and connections.
6. **Connecting Wires and Resistors**: Essential for stability and proper connections.

COMPONENT DESCRIPTIONS

1. ARDUINO UNO

- **Specifications:**
  - Microcontroller: ATmega328P
  - Operating Voltage: 5V
  - Input Voltage (recommended): 7-12V
  - Digital I/O Pins: 14 (of which 6 provide PWM output)
  - Analog Input Pins: 6
- **Purpose:**
  The Arduino Uno serves as the control unit for the voting machine. It processes input from push buttons, manages data storage, and updates the display.

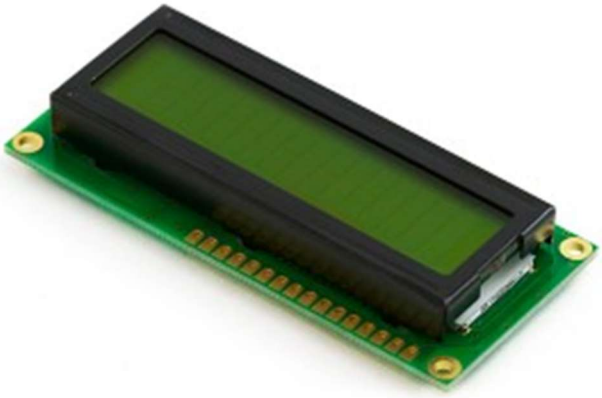


2. I2C LCD Display

- **Specifications:**
  - Type: 16x2 character LCD
  - Interface: I2C, reducing pin usage
  - Operating Voltage: 5V

- Backlight: Integrated LED backlight for clear display

- **Purpose:**
  Displays instructions for users and shows voting options. At the end of voting, it displays results or any necessary messages.



3. PUSH BUTTONS

- **Specifications:**
  - Type: Momentary push buttons
  - Operating Voltage: 5V
  - Debouncing Circuit: Required for stable signal

- **Purpose:**
  Each button represents a different candidate, allowing the user to select. Once pressed, the Arduino records the corresponding vote.

4. BUZZER

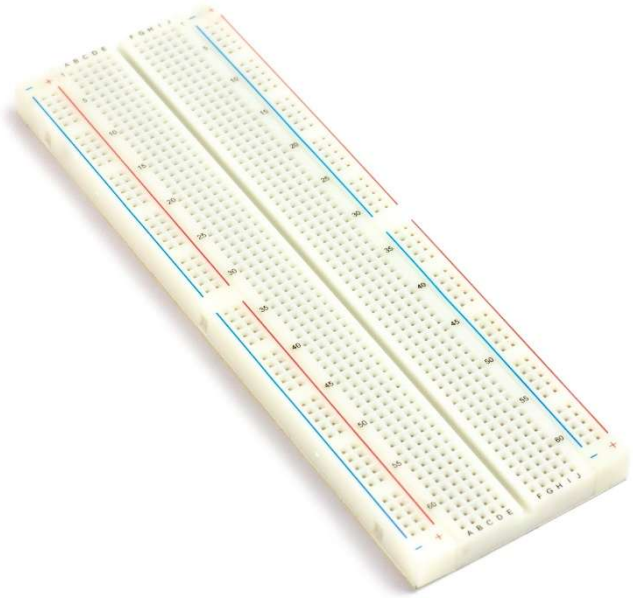
- **Specifications:**
  - Type: Piezoelectric buzzer
  - Operating Voltage: 5V
  - Sound Output: Around 85 dB

- **Purpose:**
  Provides an auditory signal to confirm actions, such as successful vote casting or system alerts.

5. BREADBOARD

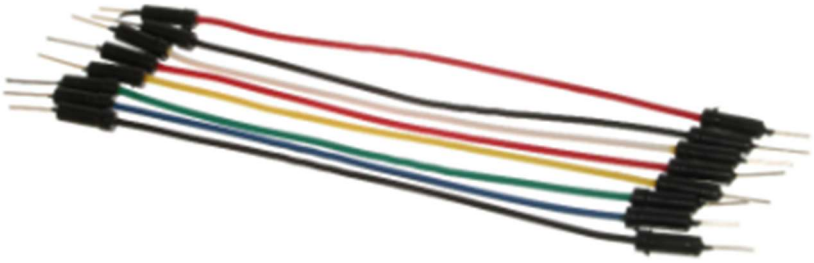
- **Specifications:**
  - Type: Solderless
  - Rows and Columns: Multiple for easy circuit connections
  - Capacity: Supports up to several components

- **Purpose:**
  Allows temporary connections between components, making circuit assembly modular and easy to modify.



6. CONNECTING WIRES AND RESISTORS

- **Specification:**
  - Wires: Insulated copper wires suitable for low-voltage circuits
  - Resistors: 220Ω and 10kΩ for stable current flow and voltage control.
- **Purpose:**
  Wires connect components, and resistors manage current flow, ensuring safe operation and preventing component damage.



THEORY AND WORKING MECHANISM

The electronic voting machine is built on an Arduino Uno board, integrating a fingerprint sensor, five push buttons, LEDs, and an LCD display. Upon powering up, the system initializes and displays "VOTING MACHINE" on the LCD. Voter authentication is managed by the fingerprint sensor, which verifies each voter's identity before granting access to the voting interface.

Four buttons are designated for candidate selection, with each press incrementing the respective candidate’s vote count and briefly illuminating LED1 to confirm the action. The polling officer uses a fifth button to conclude the voting process and display the results on the LCD, with LED0 lighting up to signal the result display mode. The Arduino continuously monitors the fingerprint sensor and button states, managing vote tallying and preventing unauthorized access or duplicate voting. After displaying results, the system resets for the next election round.

The architecture and control logic are programmed using Arduino’s setup() for initialization and loop() for real-time functions. The completed code, uploaded via Arduino IDE, ensures the system operates reliably, securely, and efficiently.

OPERATION OF EVM

The following steps outline the operation of the electronic voting machine (EVM) with fingerprint authentication:

1. **Initialization**:

   The system initializes by setting up the LCD, fingerprint sensor, and buttons. An instance of the LiquidCrystal class is created with designated pins (RS, EN, D4, D5, D6, D7 connected to pins 11, 10, 9, 8, 7, and 6 on the Arduino). The fingerprint sensor is set up, and initial values for vote counts are declared. Pin modes are defined, with A0, A1, A2, A5, and A4 for button inputs, and pins 12 and 13 for LED outputs.

2. **Fingerprint Verification**:

   Before voting, the fingerprint sensor prompts the voter to authenticate. If the fingerprint matches a stored record, the voter is granted access to proceed. Unauthorized access attempts or duplicate entries are denied.

3. **Button Press Detection**:

   Following successful authentication, the Arduino enters a loop to detect button presses for candidate selection. The buttons connected to A0, A1, A2, and A5 correspond to different candidates.

4. **Vote Counting and LED Feedback**:

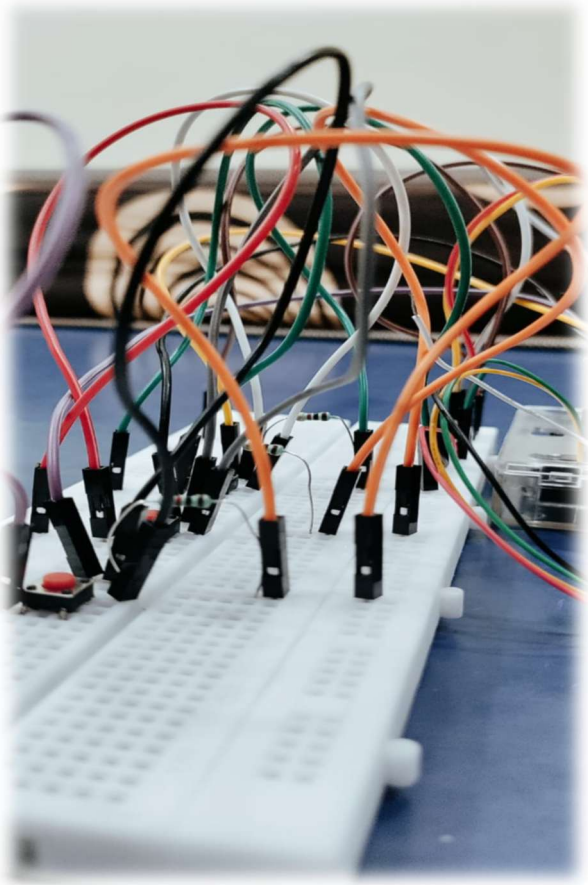
   Each button press increments the corresponding candidate's vote count, briefly illuminating LED1 to confirm the vote registration. This setup prevents unauthorized users from casting multiple votes, ensuring accuracy in vote tallying.

5. **Result Display and System Reset**:

   After voting concludes, the polling officer presses the fifth button to display results. The system identifies the candidate with the highest votes or indicates a tie on the LCD. LED0 lights up to signal results mode. After the display, vote counts are reset to zero, and the system clears the LCD, preparing for the next voting session.

RESULT

In traditional voting systems, manual counting delays result declaration until all votes are processed. By contrast, electronic voting machines (EVMs) with fingerprint authentication offer immediate result display as soon as voting concludes. The EVM setup integrates an Arduino with an LCD, push buttons, and a fingerprint sensor, allowing secure, real-time vote tallying and instant result display. This automation enhances efficiency, reduces errors, and ensures quick, accurate reporting of results, making the voting process more streamlined and secure.



CONCLUSION

The integration of a fingerprint sensor with a microcontroller-based voting system exemplifies the advancements in secure and reliable electronic polling devices. This project aimed to design and implement an affordable, secure, and straightforward electronic voting machine tailored for controlled environments like educational institutions, local communities, or small-scale elections. Unlike traditional systems, this EVM leverages biometric fingerprint authentication to ensure that only authorized individuals are permitted to vote, greatly reducing the risk of impersonation or fraudulent voting practices.

Constructed with a fingerprint sensor for user verification and a microcontroller for process control, this system provides real-time polling data, dynamically displaying each vote as it is cast and updating the results on an LCD screen. In a populous democracy like India, where secure and accurate elections are essential to governance, a biometric EVM system could potentially address key challenges of traditional methods, including issues related to booth capturing, result manipulation, and the time-intensive process of counting votes manually.

This project demonstrates the potential of a compact, secure, and efficient voting system. With minimal staff required for setup and management, as well as the simplicity of the system's design, it promises a streamlined and cost-effective approach to small-scale elections. The successful implementation of this EVM offers a model for future developments in secure electronic voting technology, which could be further scaled and adapted for larger applications. By providing a secure, transparent, and efficient means of voting, this system contributes to fostering a fair election process, thereby supporting democratic stability and economic growth.

FUTURE SCOPE

The development of electronic voting systems presents the dual challenges of maintaining security and protecting ballot secrecy, which are foundational to free and fair elections. Currently, there is no known technology capable of fully guaranteeing the secrecy, security, and verifiability of ballots cast over the internet. Online voting, though convenient, introduces a range of vulnerabilities, from unobserved vote manipulation to other security threats such as denial of service attacks, malware intrusions, and privacy breaches. Additionally, online voting systems lack a physical paper trail, which is critical for auditing and verifying election outcomes.

Blockchain-based voting, which leverages a decentralized and distributed digital ledger, also encounters significant security concerns. While promising in some areas, blockchain technology in voting remains susceptible to many of the same flaws as internet-based voting. Malware, for instance, can manipulate votes on a user's device before the ballot is cast, and the absence of a secret ballot remains a significant limitation. This means that both online and blockchain-based voting systems carry the risk of undetectable, large-scale election failures.

In contrast, our project—a microcontroller-based Electronic Voting Machine (EVM) enhanced with a fingerprint authentication system—provides a more reliable approach to securing elections. By ensuring that each vote is tied to an individual's unique fingerprint, our system addresses the critical issue of voter authentication, eliminating the risk of fake votes and impersonation. This project can also be further scaled, incorporating more sophisticated fingerprint verification techniques and adapting for broader applications.

In essence, this fingerprint-enabled EVM prototype offers a secure, auditable, and accessible solution, bringing efficiency to the voting process without compromising the principles of secrecy and security. By refining and expanding this technology, we envision a future where electronic voting not only enhances election integrity but also reinforces trust in democratic processes on a larger scale.

REFERENCES

1. Google Search. Supplementary Information for EVM Project Research, conducted via Google for background research, component datasheets, and programming guidance.

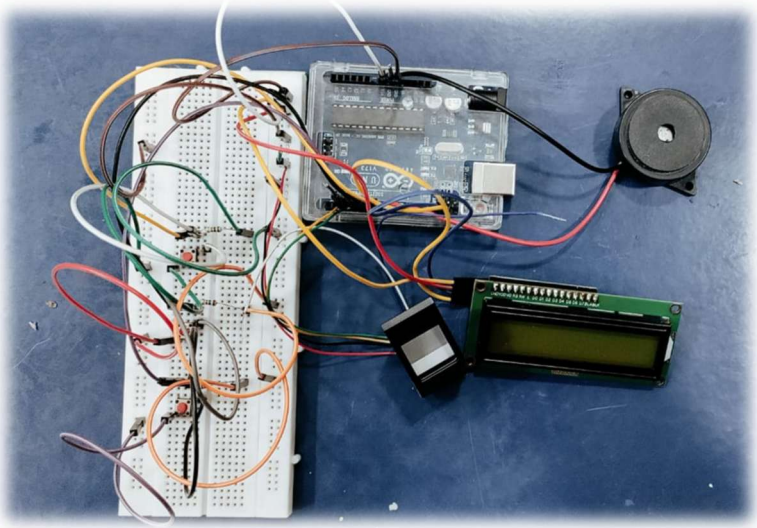
2. IJRAR. (2023). Arduino-Based Electronic Voting Machine International Journal of Research and Analytical Reviews. Retrieved from (IJRAR Research Journal)

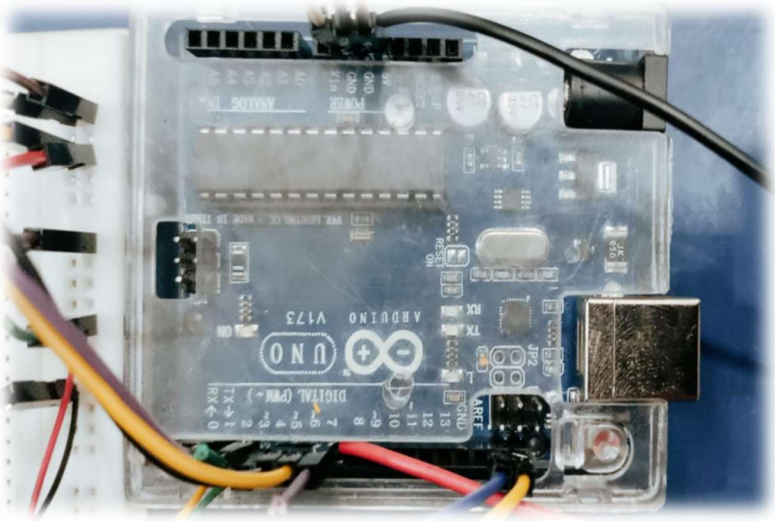
3. ChatGPT. (2024). Online Assistance for Report Writing and Technical Guidance. Accessed at: (ChatGPT)

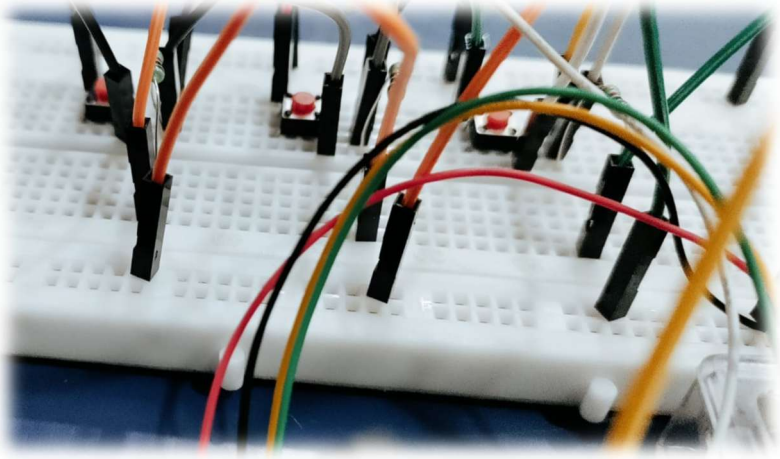
4. Gupta, A., & Sharma, P. (2023). *Arduino-Based Electronic Voting Machine. ResearchGate. Available at: ((PDF) Arduino – Based Electronic Voting Machine)

ANNEXURE

IMAGES







CODE USED

```
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);
SoftwareSerial mySerial(2, 3);
const int buttonPin1 = 4;
const int buttonPin2 = 5;
const int buttonPin3 = 6;
const int buzzer = 7;
int buttonState1 = 0;
int buttonState2 = 0;
int buttonState3 = 0;
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
int id =0,previous_voter_id = 0, vote_taken = 0;
int party_1_count=0,party_2_count=0,party_3_count=0;
String winner_name = "";
void setup()
{
  pinMode(buzzer, OUTPUT);
  pinMode(buttonPin1, INPUT);
  pinMode(buttonPin2, INPUT);
  pinMode(buttonPin3, INPUT);
    // initialize the lcd
  lcd.init();
```

```
  // Turn on the Backlight
 lcd.backlight();
 Serial.begin(9600);
 while (!Serial);  // For Yun/Leo/Micro/Zero/...
 delay(100);
 Serial.println("\n\nAdafruit finger detect test");

 // set the data rate for the sensor serial port
 finger.begin(57600);

 if (finger.verifyPassword()) {
  Serial.println("Found fingerprint sensor!");
 } else {
  Serial.println("Did not find fingerprint sensor :(");
  while (1) { delay(1); }
 }

 finger.getTemplateCount();
 Serial.print("Sensor contains "); Serial.print(finger.templateCount);
Serial.println(" templates");
 Serial.println("Waiting for valid finger...");
  lcd.clear();
 // Set cursor (Column, Row)
 lcd.setCursor(0, 0);
 lcd.print("Smart Electronic");
 lcd.setCursor(0,1);
 lcd.print("Voting Machine");
 delay(3000);
```

```
}

void loop()                    // run over and over again
{

 // Clear the display buffer
 vote_taken = 0;
 lcd.clear();
 // Set cursor (Column, Row)
 lcd.setCursor(0, 0);
 lcd.print("Please place your");
 lcd.setCursor(0,1);
 lcd.print("finger");
 delay(100);
 id = getFingerprintIDez();
 if(id > 0)
 {
  // Clear the display buffer
 lcd.clear();
 // Set cursor (Column, Row)
 lcd.setCursor(0, 0);
 lcd.print("Your Voter ID");
 lcd.setCursor(0,1);
 lcd.print(id);
 delay(2000);
 if(id == 4)
 {
```

```
    if((party_1_count > party_2_count) && ((party_1_count > party_3_count)))
    {
      winner_name = "BJP";
    }
    else if((party_2_count > party_1_count) && ((party_2_count > party_3_count)))
    {
      winner_name = "NCP";
    }
    else
    {
      winner_name = "Congress";
    }
   // Clear the display buffer
   lcd.clear();
   // Set cursor (Column, Row)
   lcd.setCursor(0, 0);
   lcd.print("winner party");
   lcd.setCursor(0,1);
   lcd.print(winner_name);
   while(1);
  }
  if(previous_voter_id != id)
  {
   do
   {
   // Clear the display buffer
```

```
lcd.clear();
// Set cursor (Column, Row)
lcd.setCursor(0, 0);
lcd.print("Give Your vote");
lcd.setCursor(0,1);
lcd.print("Press Button");
delay(500);
previous_voter_id = id;
buttonState1 = digitalRead(buttonPin1);
delay(10);
buttonState2 = digitalRead(buttonPin2);
delay(10);
buttonState3 = digitalRead(buttonPin3);
delay(10);
if (buttonState1 == HIGH)
{
  party_1_count = party_1_count +1;
  vote_taken = 1;
}
else if(buttonState2 == HIGH)
{
  party_2_count = party_2_count +1;
   vote_taken = 1;
}
else if(buttonState3 == HIGH)
{
  party_3_count = party_3_count +1;
```

```
    vote_taken = 1;
   }
   else
   {
    vote_taken = 0;
   }
   if(vote_taken == 1)
   {
    // Clear the display buffer
    lcd.clear();
    // Set cursor (Column, Row)
    lcd.setCursor(0, 0);
    lcd.print("Thanks for your");
    lcd.setCursor(0,1);
    lcd.print("vote");
    delay(200);
    digitalWrite(buzzer, HIGH);   // turn the LED on (HIGH is the voltage
level)
    delay(1000);                  // wait for a second
    digitalWrite(buzzer, LOW);    // turn the LED off by making the voltage
LOW
    delay(1000);
   }
   }while(vote_taken == 0);
  }
  else
  {
      // Clear the display buffer
```

```
  lcd.clear();
  // Set cursor (Column, Row)
  lcd.setCursor(0, 0);
  lcd.print("Duplicate Vote");
  lcd.setCursor(0,1);
  lcd.print("buzzer on");
  delay(2000);
  digitalWrite(buzzer, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                  // wait for a second
  digitalWrite(buzzer, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);
  digitalWrite(buzzer, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                  // wait for a second
  digitalWrite(buzzer, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);
  digitalWrite(buzzer, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                  // wait for a second
  digitalWrite(buzzer, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);
  }
  }
}
```

```
uint8_t getFingerprintID() {
  uint8_t p = finger.getImage();
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.println("No finger detected");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      return p;
    default:
      Serial.println("Unknown error");
      return p;
  }

  // OK success!

  p = finger.image2Tz();
  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!
p = finger.fingerFastSearch();
if (p == FINGERPRINT_OK) {
  Serial.println("Found a print match!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
  Serial.println("Communication error");
  return p;
} else if (p == FINGERPRINT_NOTFOUND) {
  Serial.println("Did not find a match");
  return p;
```

```
  } else {
    Serial.println("Unknown error");
    return p;
  }

  // found a match!
  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of "); Serial.println(finger.confidence);

  return finger.fingerID;
}

// returns -1 if failed, otherwise returns ID #
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)  return -1;

  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of "); Serial.println(finger.confidence);
  return finger.fingerID;
}
```