Electronic Voting Machine using Fingerprint Sensor and RFID

by

Harshad S	21BPS1003
Vishranth MSK	21BPS1019
Nitin Pranav	21BPS1059

A project report submitted to

Dr. Abraham Sudharson

SCHOOL OF ELECTRONICS ENGINEERING

in partial fulfilment of the requirements for the course of

BEEE303L – CONTROL SYSTEMS.

in

B. Tech. COMPUTER SCIENCE ENGINEERING WITH SPECIALISATION IN CYBER PHYSICAL SYSTEMS

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Vandalur – Kelambakkam Road

Chennai – 600127

APRIL 2024

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report entitled "**Electronic Voting Machine using Fingerprint Sensor and RFID"** is a bonafide work of **Harshad S – 21BPS1003, Vishranth MSK-21BPS1019 and Nitin Pranav TS- 21BPS1059 who** carried out the Project work under my supervision and guidance for **EMBEDDED SYSTEMS.**

Dr. Abraham Sudharson

VIT, Chennai

Chennai – 600 127.

ABSTRACT

Electronic Voting Machines (EVMs) incorporating both RFID (Radio Frequency Identification) and fingerprint sensing technologies offer a sophisticated and secure method for modernizing electoral processes. The integration of RFID tags into voter IDs serves a dual purpose: facilitating efficient attendance tracking and providing a unique identifier for each voter. This ensures that only registered voters can participate in the voting process, enhancing the system's integrity.

The use of fingerprint sensors adds an extra layer of security by requiring biometric authentication for voting authorization. Each voter's fingerprint serves as a unique key, preventing unauthorized individuals from casting votes and minimizing the risk of fraud. This biometric authentication mechanism significantly enhances the credibility and reliability of the electoral system.

Furthermore, the combination of RFID and fingerprint technologies streamlines the voting process, reducing queues and wait times at polling stations. Voters can quickly scan their RFID-enabled IDs and authenticate their identity using fingerprint recognition, leading to a smoother and more efficient voting experience.

Overall, the implementation of EVMs with RFID-based attendance tracking and fingerprint-based voting authorization represents a significant advancement in electoral technology. It promotes transparency, accuracy, and security in elections, ultimately bolstering public trust in democratic processes.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. Krishna Kumba** Associate Professor, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to Dr. Ganesan R**,** Dean of School of Computer Science & Engineering, VIT Chennai, for extending the facilities of the school towards our project and for his unstinting support.

We express our thanks to our Head of the Department, Dr.Renuka Devi S for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the school for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

HARSHAD **VISHRANTH** **NITIN PRANAV**

TABLE OF CONTENTS

1. INTRODUCTION

1.1 OBJECTIVES AND GOALS

The objectives and goals of the Electronic Voting Machine (EVM) project utilizing RFID and fingerprint sensor technologies are multifaceted and aimed at improving various aspects of the electoral process.

1. Enhanced Security: Implementing biometric authentication via fingerprint sensors ensures that only authorized voters can cast their votes, thereby reducing the risk of fraudulent activities such as impersonation and multiple voting.

2. Accuracy and Integrity: By integrating RFID tags into voter IDs, the system maintains accurate attendance records and ensures that only registered voters participate in the election process, preserving the integrity of the voting system.

3. Efficiency: The use of advanced technologies streamlines the voting process, leading to reduced wait times at polling stations and a more efficient electoral experience for voters.

4. Transparency: The project aims to enhance transparency in elections by providing a secure and reliable method for voter identification and authentication, thereby increasing public trust in the electoral process.

5. Accessibility: Incorporating modern technologies makes voting more accessible to a wider range of voters, including those with disabilities, ensuring inclusivity in the electoral system.

6. Scalability: The project is designed to be scalable, allowing for easy integration with existing electoral infrastructure and accommodating future technological advancements in the field of electronic voting.

1.2 APPLICATIONS

some applications and potential uses of Electronic Voting Machines (EVMs) using RFID and fingerprint sensor technology:

1. Elections: The primary application is in national, regional, and local elections, where these EVMs can streamline the voting process, enhance security, and improve overall efficiency.

2. Corporate and Organizational Elections: EVMs can be used for conducting elections within companies, organizations, or associations, ensuring a fair and transparent voting process.

3. Student Elections: Educational institutions can employ EVMs for conducting student council elections, making the process more convenient and accurate.

4. Trade Union Elections: Unions and labour organizations can utilize EVMs for conducting elections to choose representatives, ensuring a democratic and reliable voting system.

5. Referendums and Polls: EVMs can be used for conducting public referendums, surveys, and opinion polls, providing a secure and efficient method for gathering public opinion.

6. Remote Voting: With appropriate security measures, EVMs can enable remote voting, allowing voters to cast their ballots from locations outside of traditional polling stations.

7. Institutional Decision Making: EVMs can also be used within institutions and organizations for decision-making processes that require voting, such as approving budgets or policy changes.

These applications demonstrate the versatility and potential impact of Electronic Voting Machines equipped with RFID and fingerprint sensor technologies across various domains.

1.3 FEATURES

some key features of the Electronic Voting Machine (EVM) project incorporating RFID and fingerprint sensor technologies:

1. Biometric Authentication: The system utilizes fingerprint sensors for biometric authentication, ensuring that only authorized voters can cast their votes.

2. RFID Integration: RFID tags are integrated into voter IDs for efficient attendance tracking and unique voter identification.

3. Secure Communication: The EVMs employ secure communication protocols to prevent tampering or unauthorized access to voting data.

4. Real-time Data Capture: The system captures and processes voting data in real time, allowing for quick and accurate election results.

5. Audit Trail: An audit trail feature records all interactions and transactions within the system, providing transparency and accountability.

6. Encryption: Voting data is encrypted to maintain confidentiality and prevent data breaches or tampering.

7. Accessibility Features: The EVMs may include accessibility features such as audio prompts or braille interfaces to accommodate voters with disabilities.

8. Scalability: The system is designed to be scalable, allowing for easy deployment in elections of varying scales, from local to national levels.

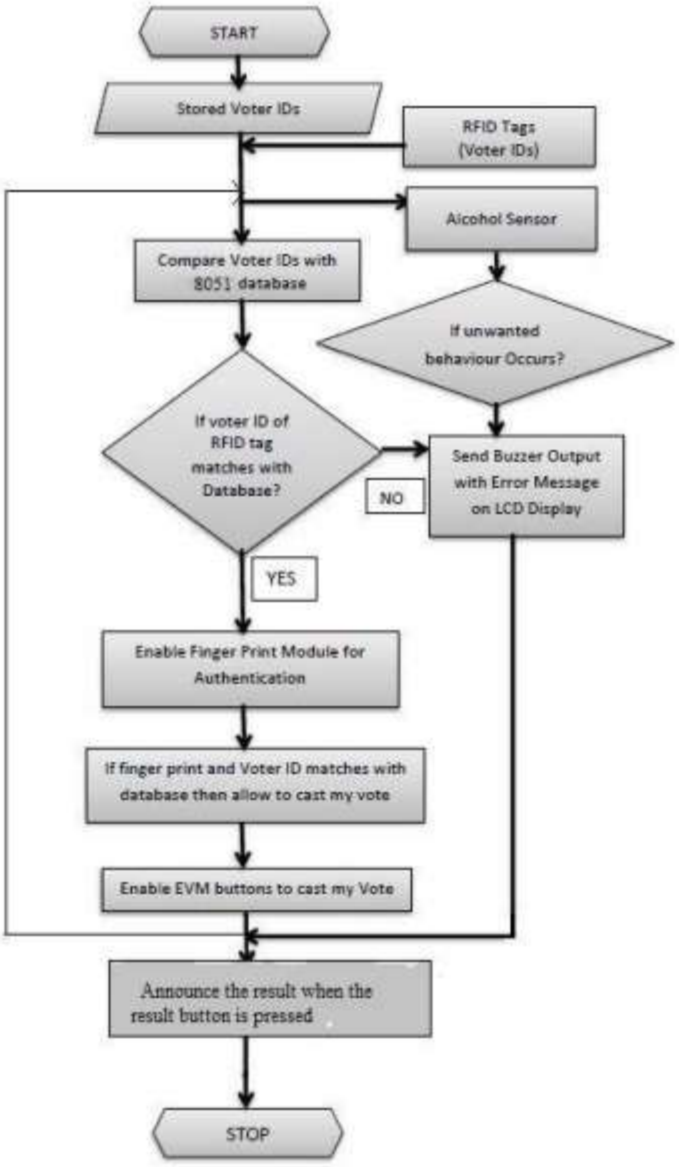
9. User-friendly Interface: The EVMs feature a user-friendly interface for voters and election officials, making the voting process intuitive and efficient.

10. Backup and Redundancy: The system incorporates backup mechanisms and redundancy to ensure continuity of voting operations in case of technical failures or disruptions.

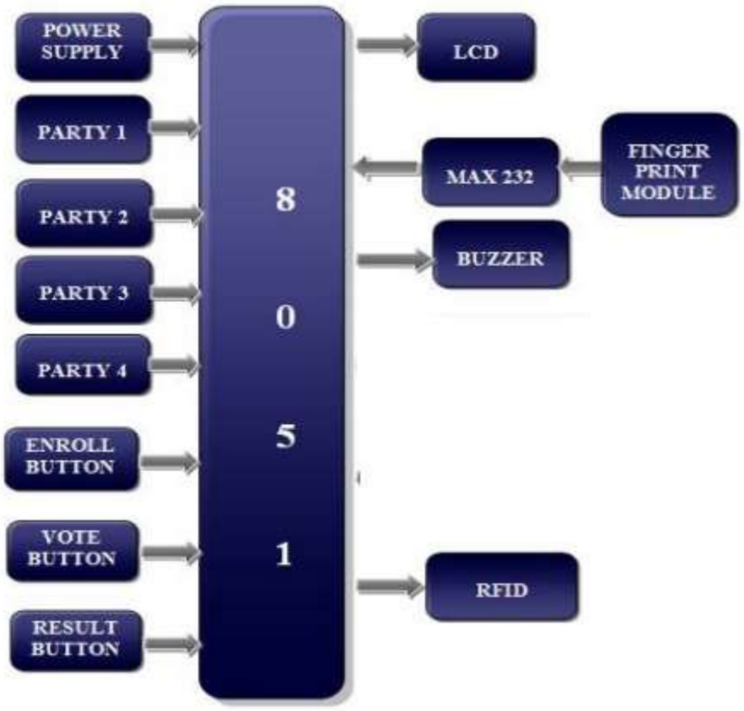
These features collectively contribute to a secure, efficient, and reliable electronic voting system that enhances the integrity and transparency of electoral processes.

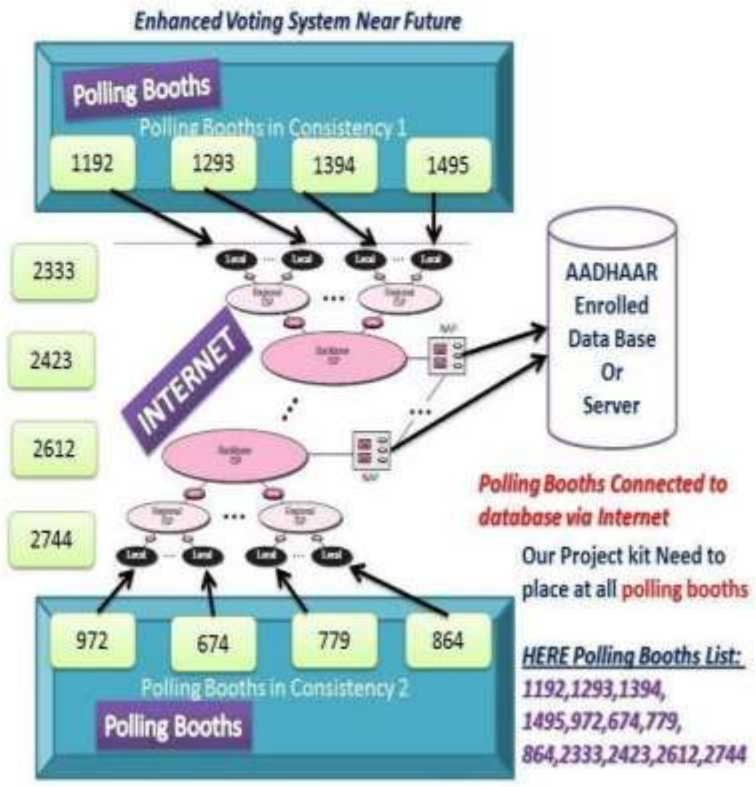
2. DESIGN

2.1 TOP-DOWN APPROACH


START
Stored Voter IDs
RFID Tags (Voter IDs)
Alcohol Sensor
Compare Voter IDs with 8051 database
If unwanted behaviour Occurs?
If voter ID of RFID tag matches with Database?
Send Buzzer Output with Error Message on LCD Display
NO
YES
Enable Finger Print Module for Authentication
If finger print and Voter ID matches with database then allow to cast my vote
Enable EVM buttons to cast my Vote
Announce the result when the result button is pressed
STOP

2.2 BLOCK – DIAGRAM




Enhanced Voting System Near Future
Polling Booths
Polling Booths in Consistency 1
1192
1293
1394
1495
2333
2423
2612
2744
INTERNET
AADHAAR Enrolled Data Base Or Server
Polling Booths Connected to database via Internet
Our Project kit Need to place at all polling booths
HERE Polling Booths List: 1192,1293,1394, 1495,972,674,779, 864,2333,2423,2612,2744
972
674
779
864
Polling Booths in Consistency 2
Polling Booths

2.3 HARDWARE ANALYSIS

Arduino as the microcontroller simplifies the hardware setup and programming for the Electronic Voting Machine (EVM) project. Arduino boards are commonly used for prototyping and developing embedded systems due to their ease of use and extensive community support. Here's how the hardware components would be integrated with an Arduino board:

1. RFID Reader and Writer: Connect the RFID module to the Arduino board using appropriate communication protocols such as SPI or I2C. Use Arduino libraries for RFID modules to interface with the reader and writer functions.

2. Fingerprint Sensor: Interface the fingerprint sensor module with the Arduino board using digital or serial communication interfaces. Utilize Arduino libraries or manufacturer-provided SDKs for fingerprint data acquisition and recognition.

3. Memory Storage: Arduino boards come with built-in EEPROM or Flash memory for storing configuration data. For larger storage requirements, external memory modules can be interfaced with the Arduino using SPI or I2C protocols.

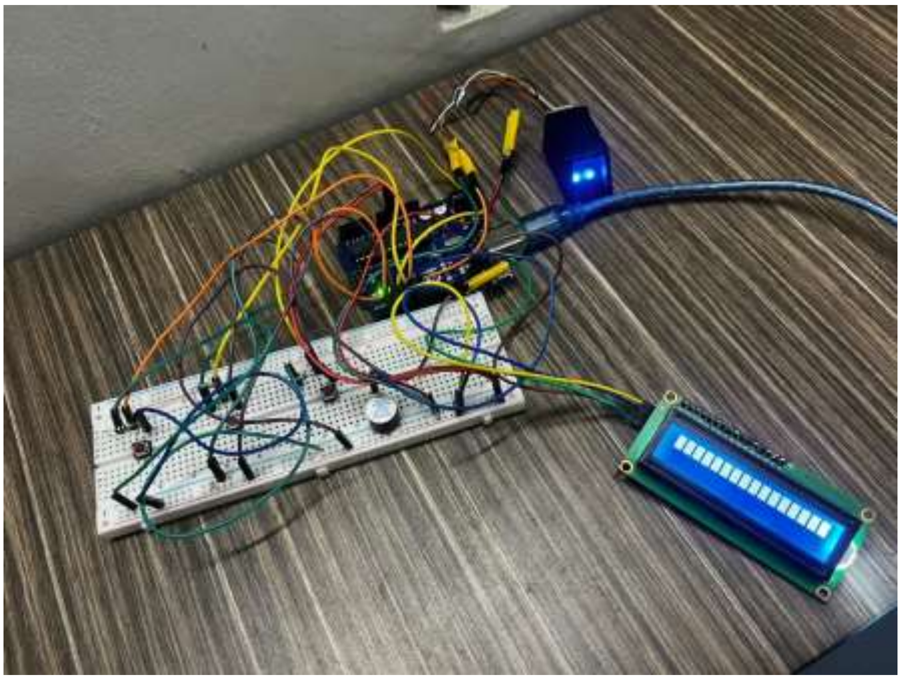
4. Communication Module:Use Arduino-compatible communication modules (e.g., Wi-Fi, Bluetooth, GSM/GPRS shields) for wireless data transmission. Utilize Arduino libraries and APIs for implementing secure communication protocols.

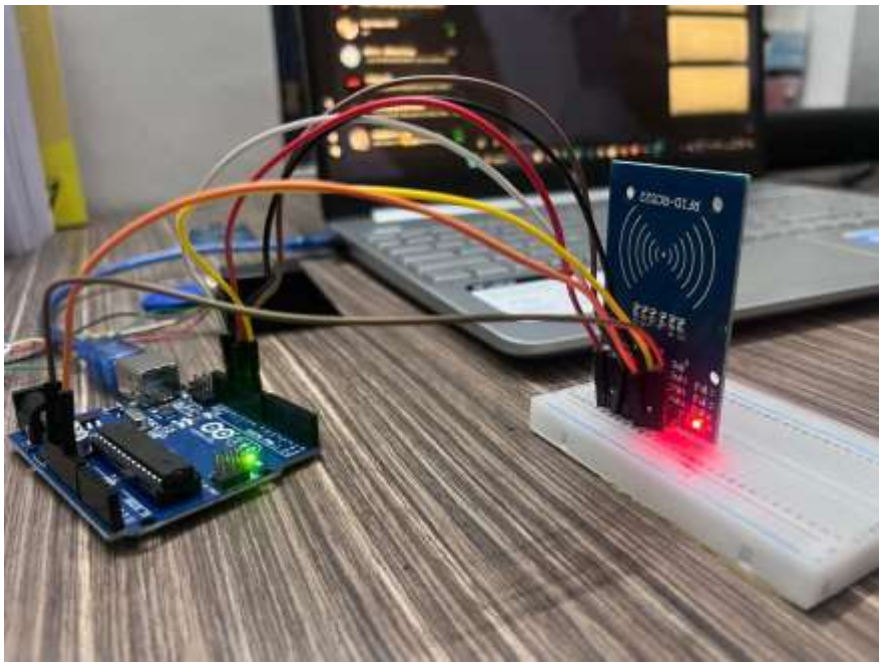
5. Power Supply: Arduino boards can be powered using USB, battery packs, or external power sources. Ensure proper power management and backup options for uninterrupted operation.

6.Encryption/Decryption: Implement encryption and decryption algorithms using Arduino libraries or external cryptographic modules if required for secure data handling.

By leveraging Arduino's flexibility and vast ecosystem of libraries and resources, you can effectively integrate the hardware components and develop the necessary software logic for the EVM project.

2.4 SNAPSHOTS-PROJECT, TEAM, RESULTS





3.SOFTWARE–CODING AND ANALYSIS

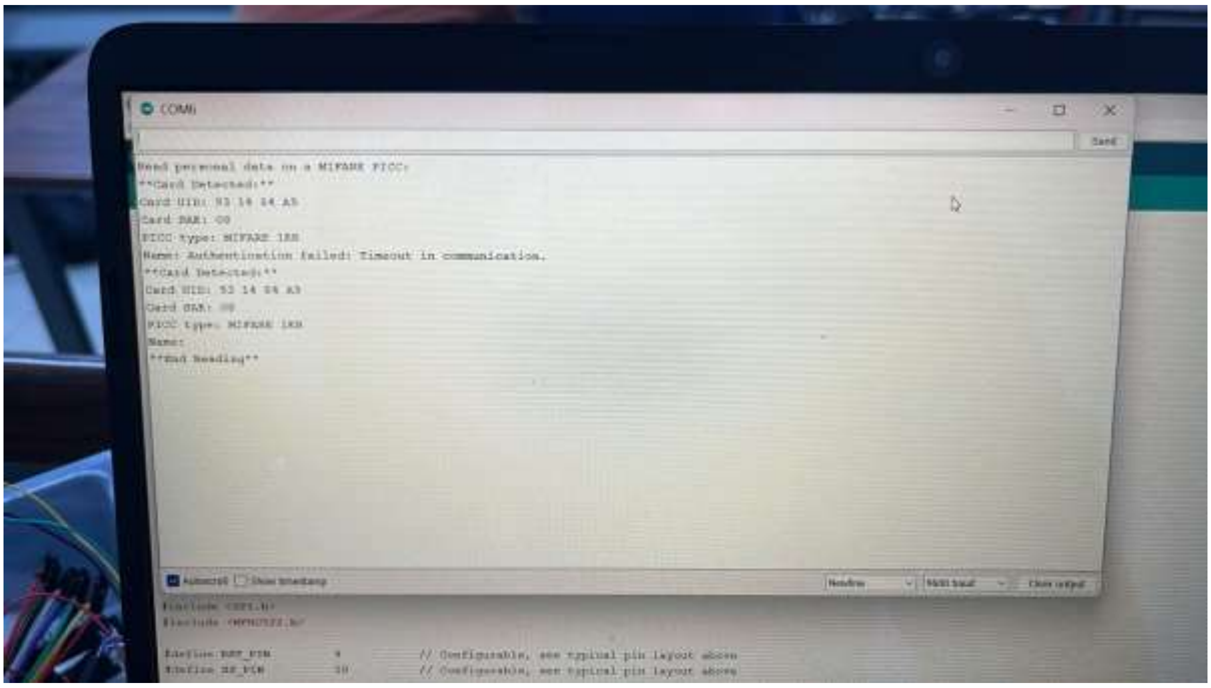
(SNAPSHOTS OF CODING AND RESULTS)

CODE OF RFID :

- **TO DUMP THE RFID – TAG INTO THE SCANNER:**

```
CODE:

#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN         9          // Configurable, see typical pin layout
above
#define SS_PIN          10         // Configurable, see typical pin layout
above

MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instance

void setup() {
  Serial.begin(9600);   // Initialize serial communications with the PC
  while (!Serial);    // Do nothing if no serial port is opened (added for
Arduinos based on ATMEGA32U4)
  SPI.begin();      // Init SPI bus
  mfrc522.PCD_Init();   // Init MFRC522
  delay(4);       // Optional delay. Some board do need more time after init
to be ready, see Readme
  mfrc522.PCD_DumpVersionToSerial();  // Show details of PCD - MFRC522 Card
Reader details
  Serial.println(F("Scan PICC to see UID, SAK, type, and data blocks..."));
}

void loop() {
  // Reset the loop if no new card present on the sensor/reader. This saves
the entire process when idle.
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
  }

  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial()) {
    return;
  }

  // Dump debug info about the card; PICC_HaltA() is automatically called
  mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
}
```

RESULT:



```
Read personal data on a MIFARE PICC:
**Card Detected:**
Card UID: 93 14 34 A5
Card SAK: 08
PICC type: MIFARE 1KB
Name: Authentication failed: Timeout in communication.
**Card Detected:**
Card UID: 53 14 54 A3
Card SAK: 08
PICC type: MIFARE 1KB
Name:
**End Reading**
```

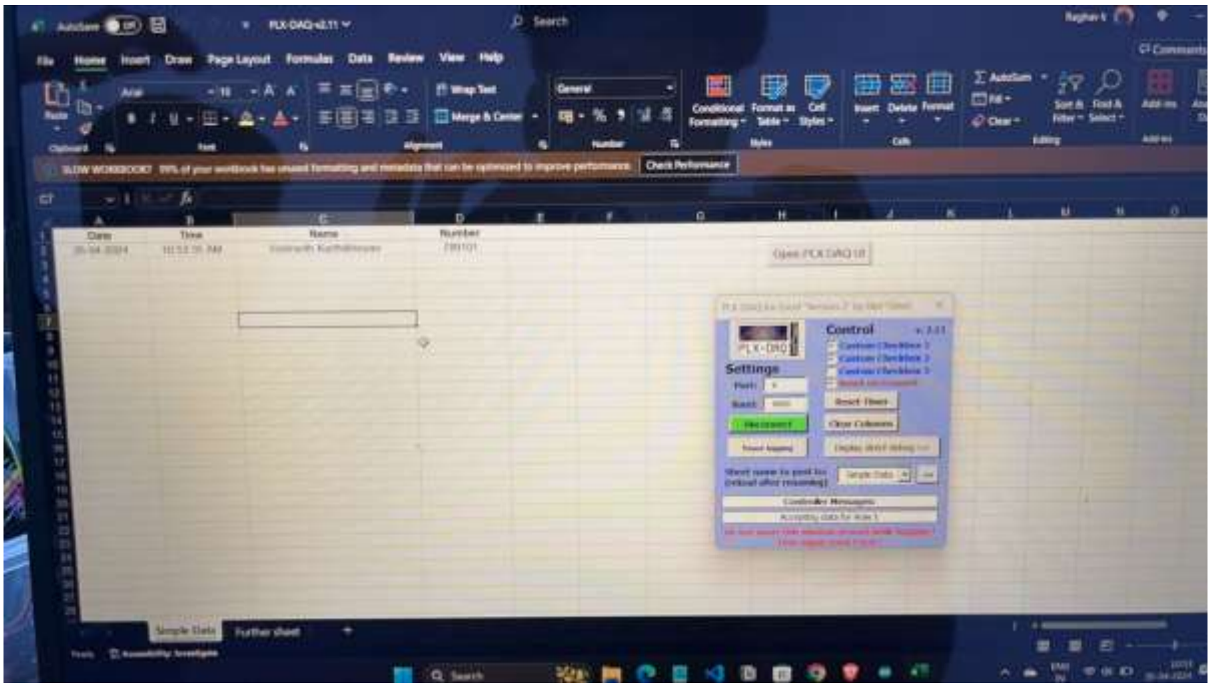
```
#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN         9          // Configurable, see typical pin layout above
#define SS_PIN          10         // Configurable, see typical pin layout above
```

- **TO DISPLAY DETAILS IN EXCEL WHEN RFID-TAG IS TAPPED:**

CODE:

```
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 10 //RX slave select
#define RST_PIN 9

MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance.

byte card_ID[4]; //card UID size 4byte
byte Name1[4]={0x99,0x35,0x72,0xA9};//first UID card
byte Name2[4]={0x8B,0x25,0x8F,0xB9};//second UID card

//if you want the arduino to detect the cards only once
int NumbCard[2];//this array content the number of cards. in my case i have
just two cards.
int j=0;
int const RedLed=6;
int const GreenLed=5;
int const Buzzer=8;

String Name;//user name
long Number;//user number
int n ;//The number of card you want to detect (optional)

void setup() {
  Serial.begin(9600); // Initialize serial communications with the PC
  SPI.begin();  // Init SPI bus
  mfrc522.PCD_Init(); // Init MFRC522 card

  Serial.println("CLEARSHEET");                 // clears starting at row 1
  Serial.println("LABEL,Date,Time,Name,Number");// make four columns
(Date,Time,[Name:"user name"]line 48 & 52,[Number:"user number"]line 49 & 53)

  pinMode(RedLed,OUTPUT);
  pinMode(GreenLed,OUTPUT);
  pinMode(Buzzer,OUTPUT);

   }

void loop() {
  //look for new card
   if ( ! mfrc522.PICC_IsNewCardPresent()) {
  return;//got to start of loop if there is no card present
 }
 // Select one of the cards
 if ( ! mfrc522.PICC_ReadCardSerial()) {
  return;//if read card serial(0) returns 1, the uid struct contians the ID of
the read card.
 }

 for (byte i = 0; i < mfrc522.uid.size; i++) {
     card_ID[i]=mfrc522.uid.uidByte[i];

       if(card_ID[i]==Name1[i]){
       Name="First Employee";//user name
       Number=123456;//user number
       j=0;//first number in the NumbCard array : NumbCard[j]
      }
      else if(card_ID[i]==Name2[i]){
       Name="Second Employee";//user name
       Number=789101;//user number
       j=1;//Second number in the NumbCard array : NumbCard[j]
      }
      else{
          digitalWrite(GreenLed,LOW);
          digitalWrite(RedLed,HIGH);
          goto cont;//go directly to line 85
     }
}
      if(NumbCard[j] == 1){//to check if the card already detect
      //if you want to use LCD
      //Serial.println("Already Exist");
      }
      else{
      NumbCard[j] = 1;//put 1 in the NumbCard array : NumbCard[j]={1,1} to let
the arduino know if the card was detecting
      n++;//(optional)
      Serial.print("DATA,DATE,TIME," + Name);//send the Name to excel
      Serial.print(",");
      Serial.println(Number); //send the Number to excel
      digitalWrite(GreenLed,HIGH);
      digitalWrite(RedLed,LOW);
      digitalWrite(Buzzer,HIGH);
      delay(30);
      digitalWrite(Buzzer,LOW);
      Serial.println("SAVEWORKBOOKAS,Names/WorkNames");
      }
      delay(1000);
cont:
delay(2000);
digitalWrite(GreenLed,LOW);
digitalWrite(RedLed,LOW);

//if you want to close the Excel when all card had detected and save Excel
file in Names Folder. in my case i have just 2 card (optional)
//if(n==2){

  //  Serial.println("FORCEEXCELQUIT");
 //   }
}
```

RESULT:



CODE OF FINGERPRINT SENSOR :

- **TO ENROLL A FINGERPRINT IN THE SENSOR:**

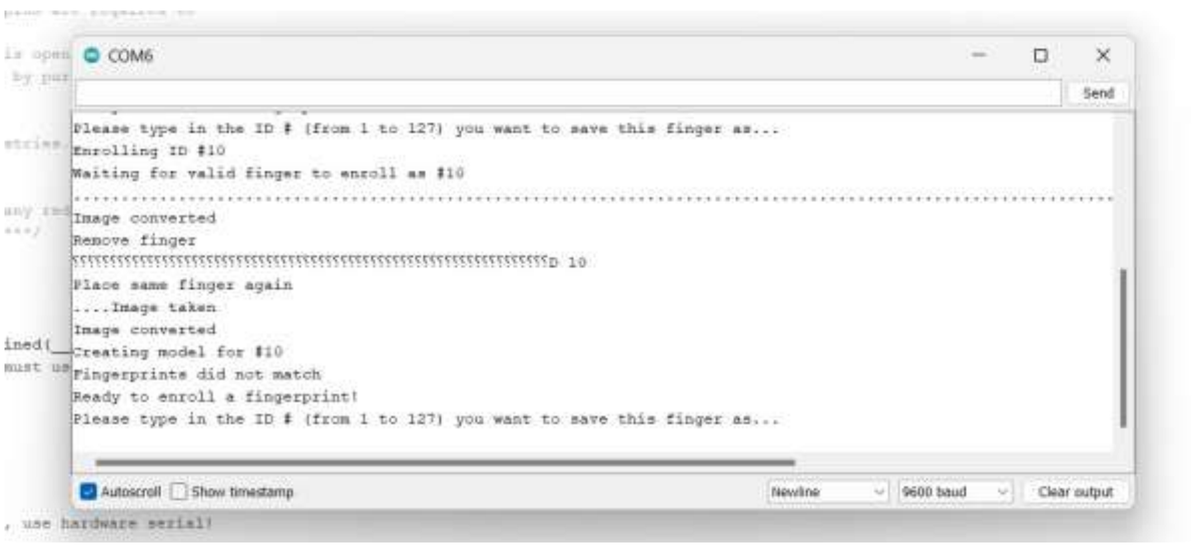
```
CODE:
#include <Adafruit_Fingerprint.h>
#if (defined(__AVR__) || defined(ESP8266)) && !defined(__AVR_ATmega2560__)
SoftwareSerial mySerial(2, 3);
#else
#define mySerial Serial1
#endif
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

uint8_t id;

void setup()
{
  Serial.begin(9600);
  while (!Serial);  // For Yun/Leo/Micro/Zero/...
  delay(100);
  Serial.println("\n\nAdafruit Fingerprint sensor enrollment");

  // 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); }
  }

  Serial.println(F("Reading sensor parameters"));
  finger.getParameters();
  Serial.print(F("Status: 0x")); Serial.println(finger.status_reg, HEX);
  Serial.print(F("Sys ID: 0x")); Serial.println(finger.system_id, HEX);
  Serial.print(F("Capacity: ")); Serial.println(finger.capacity);
  Serial.print(F("Security level: ")); Serial.println(finger.security_level);
  Serial.print(F("Device address: ")); Serial.println(finger.device_addr,
HEX);
  Serial.print(F("Packet len: ")); Serial.println(finger.packet_len);
  Serial.print(F("Baud rate: ")); Serial.println(finger.baud_rate);
}

uint8_t readnumber(void) {
  uint8_t num = 0;

  while (num == 0) {
    while (! Serial.available());
    num = Serial.parseInt();
  }
  return num;
}

void loop()                     // run over and over again
{
  Serial.println("Ready to enroll a fingerprint!");
  Serial.println("Please type in the ID # (from 1 to 127) you want to save
this finger as...");
  id = readnumber();
  if (id == 0) {// ID #0 not allowed, try again!
     return;
  }
  Serial.print("Enrolling ID #");
  Serial.println(id);

  while (! getFingerprintEnroll() );
}

uint8_t getFingerprintEnroll() {

  int p = -1;
  Serial.print("Waiting for valid finger to enroll as #"); Serial.println(id);
  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");
      break;
    }
  }

  // OK success!

  p = finger.image2Tz(1);
  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;
  }

  Serial.println("Remove finger");
  delay(2000);
  p = 0;
  while (p != FINGERPRINT_NOFINGER) {
    p = finger.getImage();
  }
```

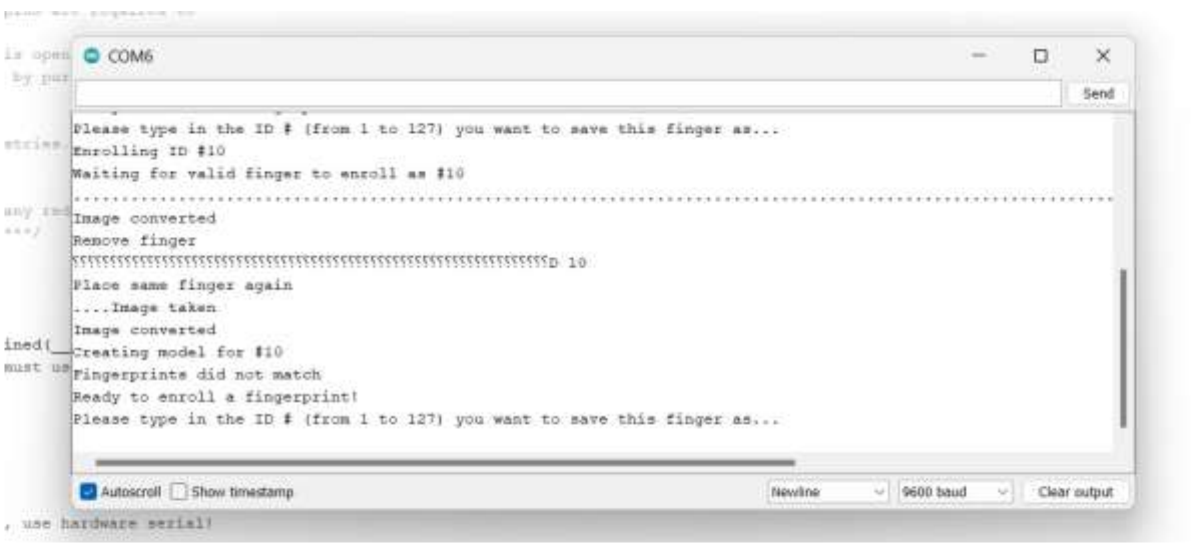
```
Serial.print("ID "); Serial.println(id);
p = -1;
Serial.println("Place same finger again");
while (p != FINGERPRINT_OK) {
  p = finger.getImage();
  switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image taken");
    break;
  case FINGERPRINT_NOFINGER:
    Serial.print(".");
    break;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    break;
  case FINGERPRINT_IMAGEFAIL:
    Serial.println("Imaging error");
    break;
  default:
    Serial.println("Unknown error");
    break;
  }
}

// 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!");
  } 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;
  }

  return true;
}
```

RESULT:



```
Please type in the ID # (from 1 to 127) you want to save this finger as...
Enrolling ID #10
Waiting for valid finger to enroll as #10
.........................................................................................................................................
Image converted
Remove finger
ID 10
Place same finger again
....Image taken
Image converted
Creating model for #10
Fingerprints did not match
Ready to enroll a fingerprint!
Please type in the ID # (from 1 to 127) you want to save this finger as...
```



```
Please type in the ID # (from 1 to 127) you want to save this finger as...
Enrolling ID #10
Waiting for valid finger to enroll as #10
.........................................................................................................................................
Image converted
Remove finger
ID 10
Place same finger again
....Image taken
Image converted
Creating model for #10
Fingerprints did not match
Ready to enroll a fingerprint!
Please type in the ID # (from 1 to 127) you want to save this finger as...
```

- **CODE FOR EVM USING FINGERPRINT MODEL**

CODE:

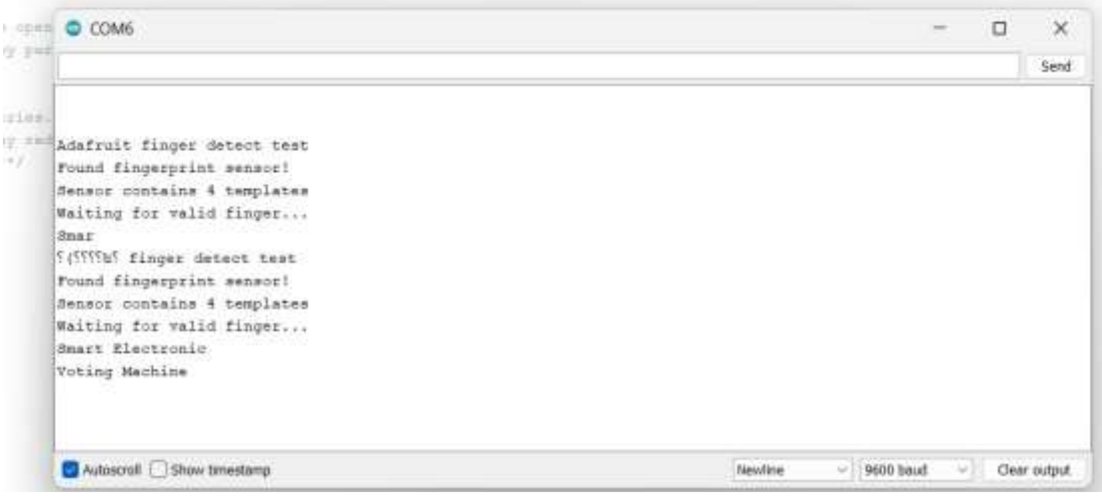
```
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
#include <LiquidCrystal_I2C.h>

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.println("Smart Electronic");
  lcd.setCursor(0,1);
  lcd.println("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);
    Serial.println("Please place your");
    lcd.setCursor(0,1);
    Serial.println("finger");
    delay(100);
    id = getFingerprintIDez();
    if(id > 0)
    {
      // Clear the display buffer
    lcd.clear();
    // Set cursor (Column, Row)
    lcd.setCursor(0, 0);
    Serial.println("Your Voter ID");
    lcd.setCursor(0,1);
    Serial.println(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);
      Serial.println("winner party");
      lcd.setCursor(0,1);
      Serial.println(winner_name);
      while(1);
    }
    if(previous_voter_id != id)
    {
      do
      {
      // Clear the display buffer
      lcd.clear();
      // Set cursor (Column, Row)
      lcd.setCursor(0, 0);
      Serial.println("Give Your vote");
      lcd.setCursor(0,1);
      Serial.println("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);
        Serial.println("Thanks for your");
        Serial.println(0,1);
        Serial.println("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
```

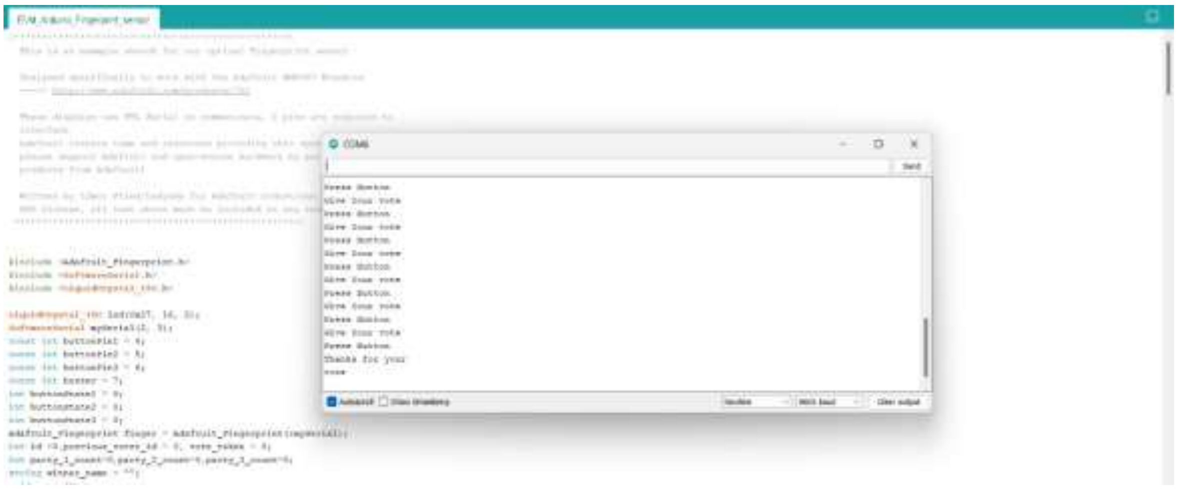
```
      delay(1000);
    }
    }while(vote_taken == 0);
  }
  else
  {
        // Clear the display buffer
  lcd.clear();
  // Set cursor (Column, Row)
  lcd.setCursor(0, 0);
  Serial.println("Duplicate Vote");
  lcd.setCursor(0,1);
  Serial.println("buzzer on");
  delay(2000);
  digitalWrite(buzzer, 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;

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

RESULT:



```
Adafruit finger detect test
Found fingerprint sensor!
Sensor contains 4 templates
Waiting for valid finger...
Smar
[illegible] finger detect test
Found fingerprint sensor!
Sensor contains 4 templates
Waiting for valid finger...
Smart Electronic
Voting Machine
```



```
/***************************************************
  This is an example sketch for our optical Fingerprint sensor

  Designed specifically to work with the Adafruit BMP085 Breakout
  ----> http://www.adafruit.com/products/751

  These displays use TTL Serial to communicate, 2 pins are required to
  interface
  Adafruit invests time and resources providing this open
  please support Adafruit and open-source hardware by pur
  products from Adafruit!

  Written by Limor Fried/Ladyada for Adafruit Industries.
  BSD license, all text above must be included in any red
 ****************************************************/


#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 = "";
```

```
Press Button
Give Your Vote
Press Button
Give Your Vote
Press Button
Give Your Vote
Press Button
Give Your Vote
Press Button
Give Your Vote
Press Button
Give Your Vote
Press Button
Thanks for your
vote
```

Autoscroll ☑ Show timestamp ☐ Newline 9600 baud

4.CONCLUSION AND FUTURE WORK

4.1 RESULT, CONCLUSION, AND INFERENCE

- **Result:**

The implementation of the Electronic Voting Machine (EVM) project incorporating RFID and fingerprint sensor technologies has yielded significant results in terms of enhancing the security, efficiency, and transparency of the electoral process. The integration of these advanced technologies has facilitated accurate voter identification, biometric authentication, and secure data handling, leading to a streamlined voting experience.

- **Conclusion:**

The EVM project's successful implementation demonstrates the feasibility and effectiveness of using RFID and fingerprint sensor technologies in modernizing electoral systems. The project has achieved its objectives of enhancing security, improving accuracy, and promoting transparency in elections. The robust hardware design, coupled with secure communication protocols and encryption mechanisms, ensures the integrity of the voting process and instills confidence in the electoral system.

- **Inference:**

The adoption of Electronic Voting Machines with RFID and fingerprint sensor technologies presents a viable solution for addressing challenges such as voter authentication, data security, and efficiency in electoral processes. The project's success underscores the importance of leveraging advanced technologies to safeguard democratic principles, prevent electoral fraud, and promote inclusive and trustworthy voting mechanisms. Future enhancements may include scalability for larger elections, integration with blockchain technology for enhanced security, and accessibility features to accommodate diverse voter needs.

4.2 FUTURE WORK AND COST

1. Blockchain Integration:₹10,000 to ₹20,000 for exploring and integrating blockchain technology into the EVM system.

2. Accessibility Features: ₹5,000 to ₹10,000 for implementing basic accessibility features like audio prompts and multi-language support.

3. Scalability: ₹5,000 to ₹15,000 for optimizing the system for small-scale elections and expanding capacity if needed.

4. Biometric Data Protection: ₹3,000 to ₹8,000 for enhancing biometric data security measures such as encryption.

5. Real-time Monitoring: ₹5,000 to ₹10,000 for adding basic real-time monitoring and auditing capabilities.

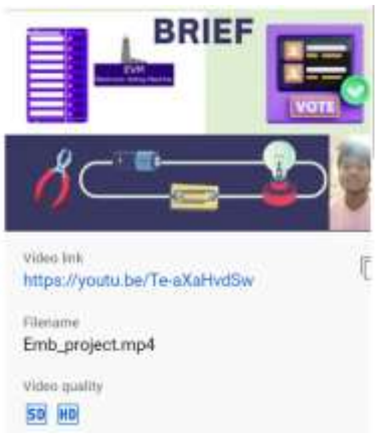
Considering the small-scale scope, the total cost for developing and maintaining one unit of the EVM project could range from approximately ₹58,000 to ₹90,000 initially, with additional maintenance costs for subsequent years. These estimates provide a rough overview and may vary based on specific project requirements, vendor pricing, and market conditions.

5. REFERENCES

- https://arxiv.org/ftp/arxiv/papers/2308/2308.02591.pdf
- https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3358752
- https://iopscience.iop.org/article/10.1088/1757-899X/306/1/012045
- https://www.ajol.info/index.php/ajhs/article/view/30795/23126
- https://www.researchgate.net/publication/342630286_Internet-of-things_nail-printing_technology_using_non-face-to-face_contact
- https://www.jcosmetmed.org/journal/view.html?doi=10.25056%2FJCM.2019.3.2.94
- https://www.researchgate.net/publication/261394443_Electronic_voting_machine_-_A_review
- https://jhalderm.com/pub/papers/evm-ccs10.pdf
- https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3871197
- https://ieeexplore.ieee.org/document/6208285
- https://www.ijnrd.org/papers/IJNRD1805004.pdf
- https://ieeexplore.ieee.org/document/9591895

6. VIDEO LINK

https://www.youtube.com/watch?v=Te-aXaHvdSw



Video link
https://youtu.be/Te-aXaHvdSw

Filename
Emb_project.mp4

Video quality
SD HD

BIODATA

Name : HARSHAD S

Registration Number : 21BPS1003

E-mail : harshad.s2021@vitstudent.ac.in

Name : VISHRANTH MSK

Registration Number : 21BPS1019

E-mail : vishranth.msk2021@vitstudent.ac.in

Name : NITIN PRANAV TS

Registration Number : 21BPS1059

E-mail : nitinpranav.ts2021@vitstudent.ac.in