BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 416 (January 2022) **A1**
Microprocessor and Embedded Systems Laboratory

Final Project Report

Fingerprint based Biometric Electronic Voting Machine with IoT based real time voting monitoring

Evaluation Form:

STEP	DESCRIPTION	MAX	SCORE
1	Report (Format, Reference)	10	
2	Design Method and Complete Design (Hardware Implementation)	15	
3	Video Demonstration	10	
4	Novelty of Design	15	
5	Project Management and Cost Analysis	10	
6	Considerations to Public Health and Safety, Environment and Cultural and Societal Needs	10	
7	Assessment of Societal, Health, Safety, Legal and Cultural issues relevant to the solution	10	
8	Evaluation of the sustainability and impact of designed solution in societal and environmental contexts	10	
9	Individual Contribution (Viva)	20	
10	Team work and Diversity	10	
	TOTAL	120	

Signature of Evaluator: ______________________

Academic Honesty Statement:

IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. Type the student ID and Write your name in your own handwriting. *You will not receive credit for this project experiment unless this statement is signed in the presence of your lab instructor.*

"In signing this statement, We hereby certify that the work on this project is our own and that we have not copied the work of any other students (past or present), and cited all relevant sources while completing this project. We understand that if we fail to honor this agreement, We will each receive a score of ZERO for this project and be subject to failure of this course."

Table of Contents

1 Abstract

Electronic Voting Machine is an essential instrument for conducting a democratic election properly. In this era of digitalization, EVM is highly needed for a fair and transparent balloting. In order to add secure voting, biometric authentication can be added. Also, to avoid any suspicion in the voting results, live monitoring via Internet of Things (IoT) should be used. In our project, we have implemented a Fingerprint based Biometric Electronic Voting Machine with IoT based real life voting monitoring.

2 Introduction

Selection of monarchs is an age-old procedure. From ancient times people have been choosing their representatives using the method of democracy. Nowadays the process is getting more and more complicated and modernized. To prevent fake voting and maintain transparency, electronic voting machines are used which ensures high efficiency and safety. If we use fingerprint sensors in EVM, we can authorize registered voters. By installing a buzzer, we can raise an alarm when someone wants to try to give a second vote.
Another important feature is using IoT - Internet of Things to monitor the voting process live. There are often accusations of fixing the vote. To prevent this, ThingSpeak online platform was used to show the process using a channel. On the ThingSpeak channel, each vote is updated real time and also the result is shown as soon as it is published. A fingerprint based EVM with live monitoring will certainly boost the future in democratic elections.

3 Design

3.1 Design Method

Components used:

1. Arduino Uno
2. LCD Display
3. Push Buttons
4. ESP32 WIFI Module
5. Custom Printed Circuit Board
6. Fingerprint Sensor
7. Battery
8. Voltage Dividers

Methodology:

We have used the Arduino Uno board as a controller and it is the heart of our schematic. Fingerprint sensor is used to enroll and match the voters, ESP32 module is used to send data to the cloud. Push buttons are used to navigate the whole process where there is a buzzer for the alarm system. And the overall process is displayed and guided by an LCD display.

3.2 Circuit Diagram:

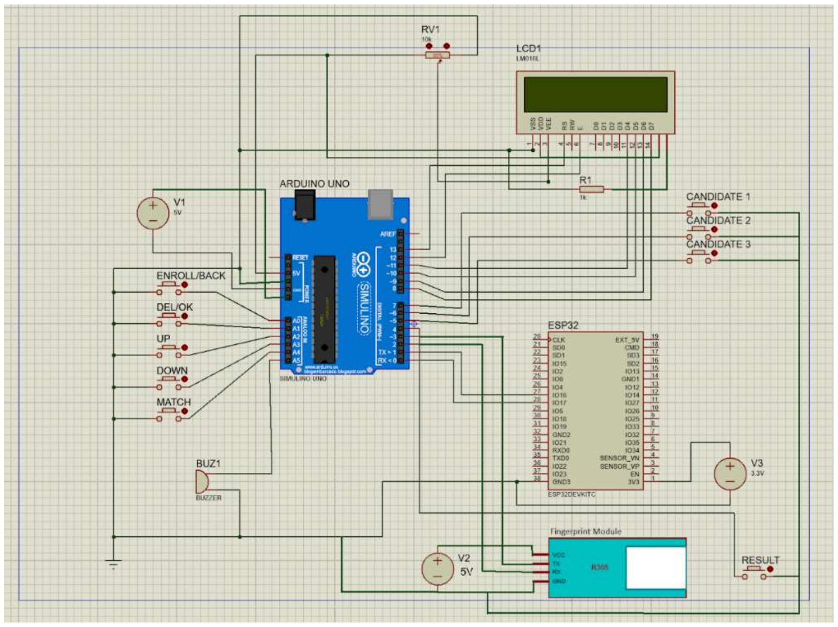


Fig: Circuit Schematic

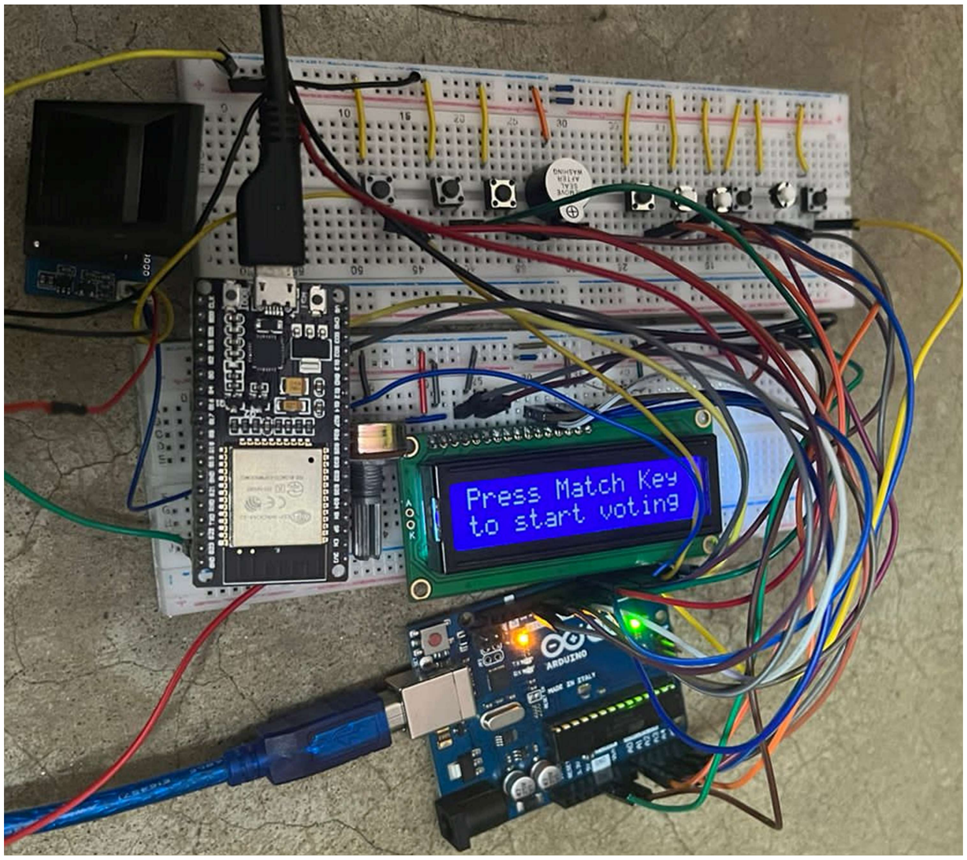


Fig: Hardware Setup

3.2 Full Source Code of Firmware

```
#include<EEPROM.h>
#include <Adafruit_Fingerprint.h>
#include<LiquidCrystal.h>
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);
uint8_t id;

#define enroll 14

#define del 15

#define up 16

#define down 17


#define match 18

//#define indVote 6

#define sw1 7
#define sw2 6

#define sw3 5

#define resultsw 4

//#define indFinger 7

#define buzzer 19

#define records 10
#define cnt_const 10

int vote1, vote2, vote3;

int voter_cnt[records];

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);
```

```
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("EVM");

  lcd.setCursor(0, 1);

  lcd.print("by Finger Print");

  delay(2000);

  lcd.clear();

  lcd.print("EEE 416");

  lcd.setCursor(0, 1);

  lcd.print("Group 2");

  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(9600);

  lcd.clear();

  lcd.print("Finding Module");
lcd.setCursor(0, 1);

  delay(1000);
```

Table: Source Code for the main program

```
  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(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()

{
  //checkKeys();

  lcd.setCursor(0, 0);

  lcd.print("Press Match Key ");

  lcd.setCursor(0, 1);

  lcd.print("to start voting");

  if (digitalRead(match) == 0)

  {
    delay(500);
for (int i = 0; i < 3; i++) //change to candidate number
    {
      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) &&
voter_cnt[result] == 0)

          {
            lcd.clear();

            lcd.print("Authorised Voter");
            delay(1000);

            lcd.setCursor(0, 1);

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

            delay(1000);

            Vote();
            flag = 1;

            voter_cnt[result] = 1;
            return;
          }

        }
if (flag == 0)

        {

          lcd.clear();

          lcd.print("Already voted");
          digitalWrite(buzzer, HIGH);

          delay(1000);

          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();
  }
  else 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();

    delay(1000);

    EEPROM.write(0, 0);

    EEPROM.write(1, 0);

    EEPROM.write(2, 0);

    lcd.clear();

    lcd.print("System Reset");

    delay(1000);

    for (int j = 0; j < records; j++)
      voter_cnt[j] = {0};

   return;
  }
}

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 > cnt_const)

        count = 0;

      delay(500);

    }
```

```
else if (digitalRead(down) == 0)
    {
      count--;
      if (count < 0)
        count = cnt_const;
      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:
```

```
else if (digitalRead(down) == 0)
    {
      count--;
      if (count < 0)
        count = cnt_const;
      delay(500);
    }
    else if (digitalRead(del) == 0) /// ok or delete button
    {
      id = count;
      getFingerprintEnroll();
      for (int i = 0; i < records; i++)
      {
        if (EEPROM.read(i + 10) == 0) ///// original condition
was EEPROM.read(i+10)== xFF, had to change it to enter into the
loop & store the id
        {
          //Serial.println("id_in loop_:");
          //Serial.println(id);
          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 > cnt_const)
        count = 0;
      delay(500);
    }
```

```
else if
    {
      co
      if

      de
    }
    else
    {
      id
      de
      fo
      {




      }
      re
    }
    else
    {
      re
    }
  }
}


uint8_t
{
  int p
  lcd.cl
  lcd.pr
  lcd.pr
  lcd.se
  lcd.pr
  delay(
  while
  {
    p =
    swit
    {
      ca
      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();
  }
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");
```

```
        //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;
    }
  }
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");
```

```
      re
    case
      //
      lc
      lc
      re
    case
      //
      lc
      lc
      re
    defa
      //
      lc
      lc
      re
  }

  //Seri
  lcd.cl
  lcd.pr
  delay(
  p = 0;
  while
    p =
  }
p = -1;
  //Seri
  lcd.cl
  lcd.pr
  lcd.se
  lcd.pr
  while
    p =
    swit
      ca


      ca


      ca
```

```
        break;

      case FINGERPRINT_IMAGEFAIL:

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

        break;

      default:

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

        return p;

    }

  }
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)
```

```

      ca


      de



    }

  }
  {

    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("Finger 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");

  delay(500);

  //digitalWrite(buzzer, LOW);

  delay(1000);

  while (1)

  {

    if (digitalRead(sw1) == 0)

    {

      vote1++;
      Serial.print(vote1);
      Serial.print(" ");


      Serial.print(vote2);
      Serial.print(" ");
      Serial.print(vote3);
      // Serial.print(" ");
      //Serial.print("can1 vote count\t");


      voteSubmit(1);

      EEPROM.write(0, vote1);

      while (digitalRead(sw1) == 0);

      return;

    }

    if (digitalRead(sw2) == 0)

    {

      vote2++;
      //          Serial.println(vote2);
      //          Serial.println("\t");

      Serial.print(vote1);
      Serial.print(" ");


      Serial.print(vote2);
      Serial.print(" ");
      Serial.print(vote3);
      //Serial.print(" ");

      voteSubmit(2);

      EEPROM.write(1, vote2);

      while (digitalRead(sw2) == 0);

      return;

    }

    if (digitalRead(sw3) == 0)

    {

      vote3++;
      //          Serial.println(vote3);
      //          Serial.println("\t");

      Serial.print(vote1);
      Serial.print(" ");


      Serial.print(vote2);
      Serial.print(" ");
      Serial.print(vote3);
      //Serial.print(" ");

      voteSubmit(3);

      EEPROM.write(2, vote3);

      while (digitalRead(sw3) == 0);

      return;

    }


  }

}


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;

}
```

4 Implementation

4.1 Description

Initially we implemented the schematic and components on breadboard - due to feasibility and low cost. Basically, we used Arduino as controller, LCD display to show the status, ESP32 to send data to cloud, fingerprint sensor for enrolling and verification. All these components eventually sum up to a system that can enroll voters, verify voters, allow them to vote if qualified, show live results and eventually send them to the cloud. When we finally ensured that the project was functioning, we decided to move to PCB as it is more compact, robust, neat, easy to use and less vulnerable to disconnections. To design PCB, we used Proteus software and for simplicity of the design, we opted for two layers design so that we did not have to worry about wire overlapping. We decided to keep the Arduino at the bottom of the PCB, so all wires went to Arduino pin ended up at top layer. All the other wirings started and ended up at the bottom layer.

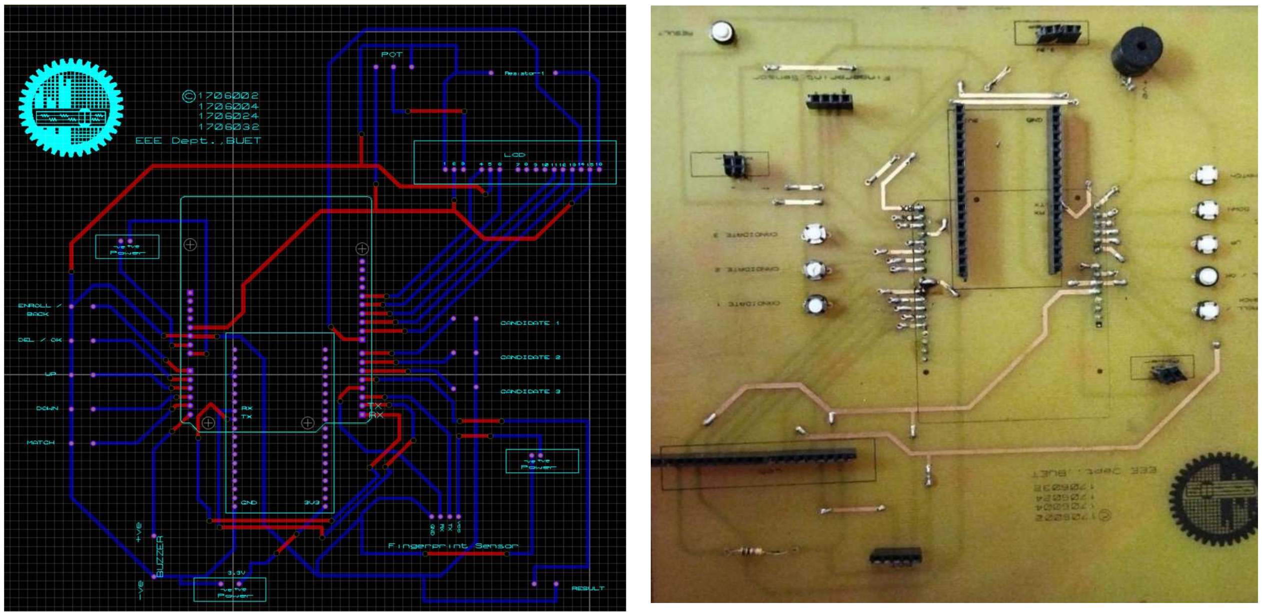


Figure 2: (Left) PCB Layout and (Right) Implementation of Design

4.2 Results

The whole process basically consists of two modes

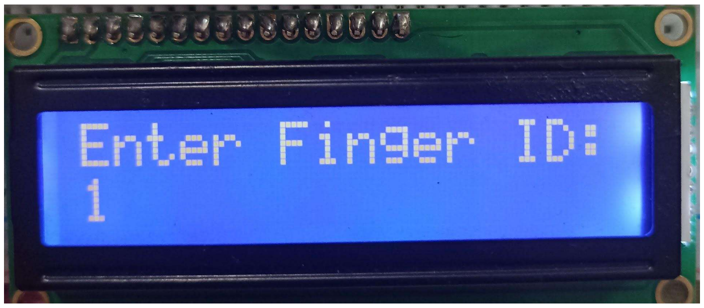
a. Voter enrollment

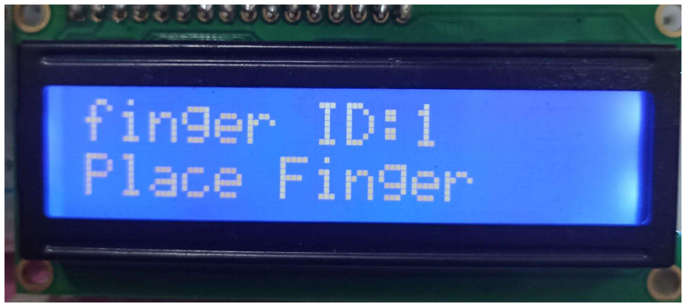
b. Vote casting

In the pre-election process, voter enrollments need to be done, i.e. voters' fingerprints need to be enrolled. During election the voters need to authenticate their identity, if they are verified then they are allowed to cast their vote. If the fingerprint matches the finger which is stored in the database then it is validated and verified.

Voter Registration process

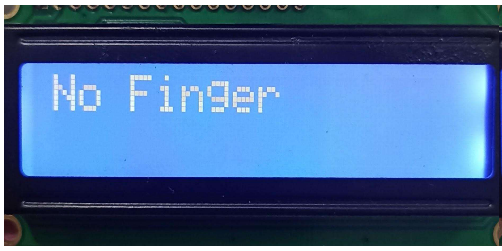
Voter's enrollment can be done by pressing the "Enroll" push button and the LCD displays a message "Enter Finger ID". This is a unique registration id for each voter. The voter needs to place the same finger on the fingerprint sensor so that the voter is enrolled. Suppose if there is any error in placing the finger then it displays "No finger" .If it is successful then the voter is enrolled and LCD displays "Stored!". This process is only accessible by the election commission or electoral officials.

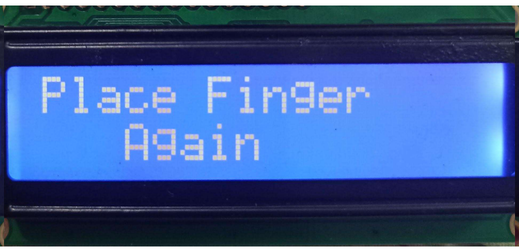




a) b)

Fig: a) During the enrollment process unique voter registration id corresponding to voter fingerprint is stored in the database, b) After entering a fingerprint ID the registrant is asked to place finger on this sensor.



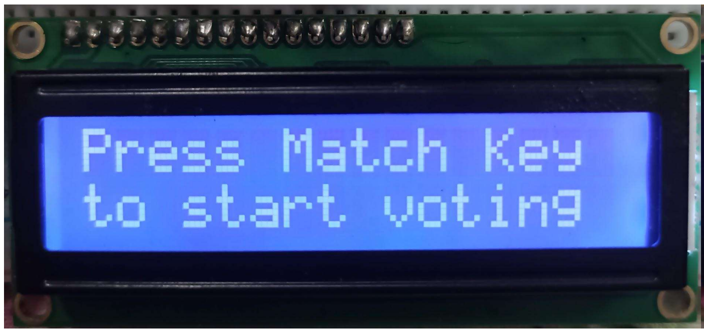


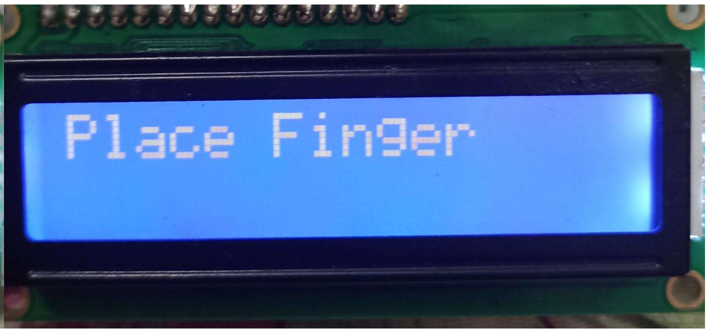
a) b) c)

Fig: a) If the sensor cannot capture the fingerprint properly, the LCD shows “No Finger”, b) If fingerprint image is taken properly, the registrant is asked to place his finger again for proper fingerprint information, c) Once the enrollment process is complete without any error, the LCD shows “Stored!”.

Vote Casting Process

In this process, the voter is allowed to prove his identity by verifying and validating his biometric. If the voter is an enrolled person, then the LCD displays a message “Authorized Voter” and he can cast his vote. If the person is not enrolled and comes to vote, biometric is mismatched with that of the database then it shows an error message on the LCD “Finger Not Found”, and is not allowed to cast his vote. Suppose a same person who has already voted comes to vote again then during the authenticating process it displays an error message “Already Voted” and is not allowed to cast his/her vote. If a person tries to vote twice, the buzzer will make a sound to alarm the law enforcers which can be mentioned as a modest security feature of our project. So, only the authenticated votes are updated for counting.

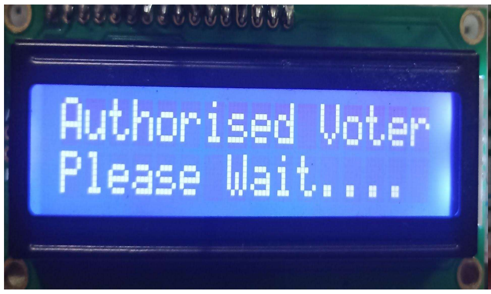


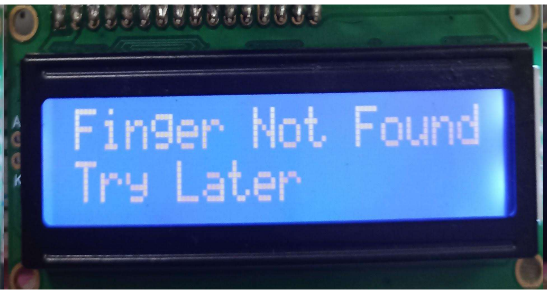


a)

b)

Fig: a) In the vote casting stage, a voter is asked to press the “Match” key/button to start voting, b) After pressing the match button, the voter is asked to place his finger on the R305 module for identity authentication.





a) b) c)

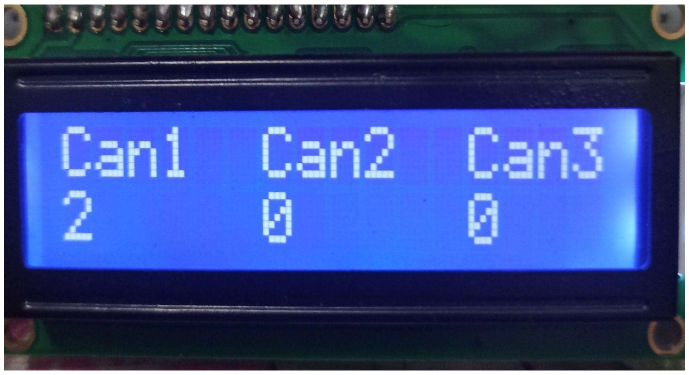
Fig: After placing the finger on the sensor, 3 different situations may arise:- a) If the fingerprint of the voter matches with the fingerprint id stored in database, LCD shows “Authorized Voter”, b) If the person is not a legitimate voter or his id is not enrolled in database or the sensor cannot recognize his fingerprint due to poor finger placing, LCD displays “Finger Not Found, Try Later”, c) If a person tries to vote twice, sensor detects it and the LCD shows “Already Voted”

a) b)

Fig: a) If the voter identity is validated, the voter is asked to place his vote to his preferred candidate,

b) When a voter submits a vote to a candidate, LCD displays that his vote is submitted and also the candidate's name who he voted for. Here, the voter has submitted his vote to candidate 1.

Voting Result:
This functionality is handled by election administration. When the vote casting process is complete, the officials can press the result button to see the total vote count of each candidate. Result is displayed in two methods- 1. Offline result, 2. Online Result
Offline results are shown on LCD Display.



a) b)
Fig: a) To see the result, officials press the "Result" key/button. After pressing the result button, LCD shows candidate name and their obtained total votes. In this case, candidate 1, candidate 2 and candidate 3 have obtained total 2, 0 and 0 votes respectively, b) Based on the vote count, the candidate with the highest vote count is declared as winner. In this case, candidate 1 has obtained the highest votes, so he is declared the winner.

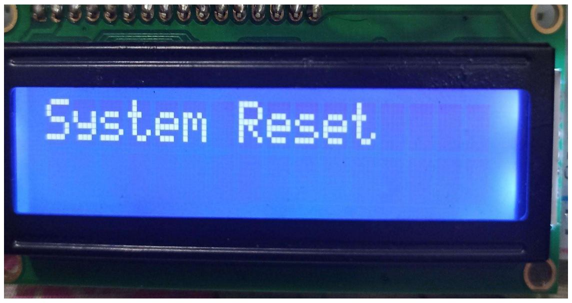
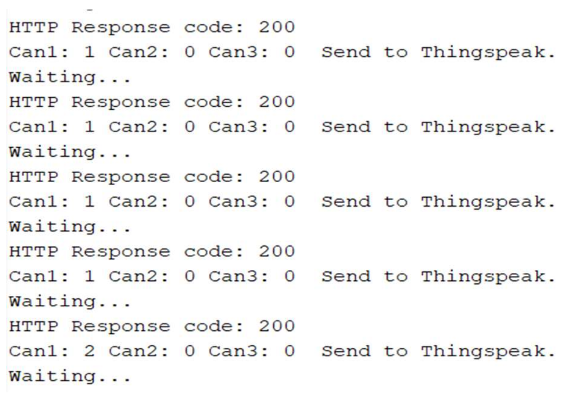


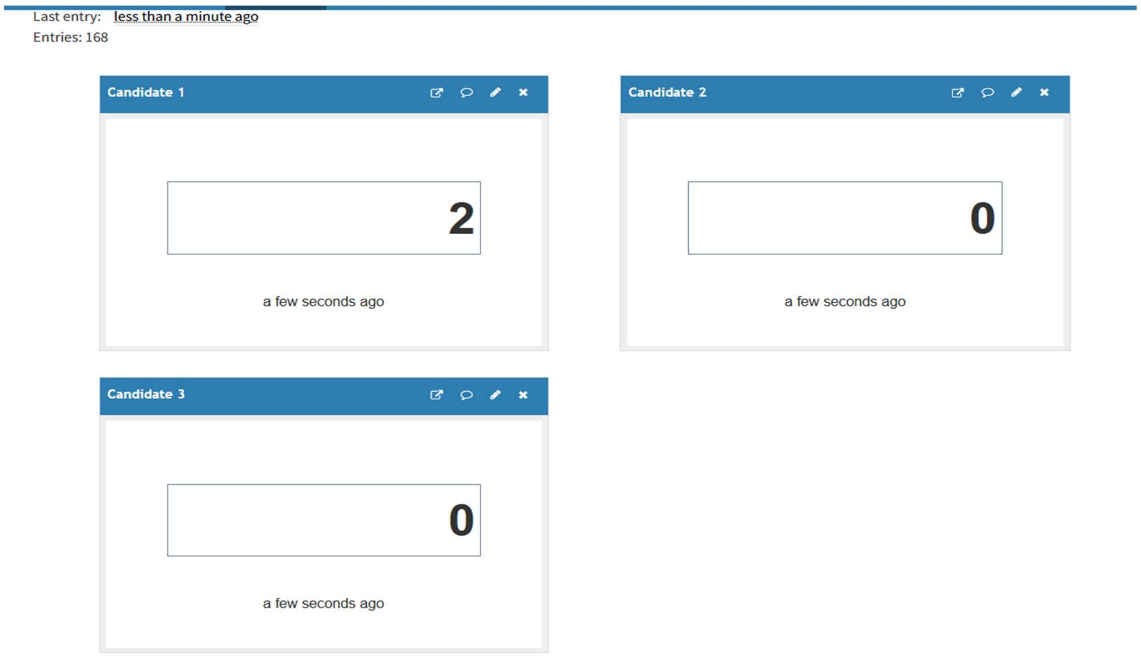
Fig: After showing the winner, the system is reset. In this way, the same EVM can be used again as its earlier memory is erased.

The online part of the result or live monitoring is executed by the wifi module ESP32. We have used "Thingspeak" as our IoT platform. ThingSpeak is an open-source Internet of Things (IoT) application and API to store and retrieve data from things using the HTTP protocol over the web or via Local Area Network. Thingspeak enables the creation of sensor logging applications, location tracking applications, and a social network of things with status updates. First, we have to create an account in ThingSpeak, Login and create a new channel. We will get a channel id and API key. For our channel, we created 3 numerical data fields for the total vote count of 3 candidates.



```
HTTP Response code: 200
Can1: 1 Can2: 0 Can3: 0  Send to Thingspeak.
Waiting...
HTTP Response code: 200
Can1: 1 Can2: 0 Can3: 0  Send to Thingspeak.
Waiting...
HTTP Response code: 200
Can1: 1 Can2: 0 Can3: 0  Send to Thingspeak.
Waiting...
HTTP Response code: 200
Can1: 1 Can2: 0 Can3: 0  Send to Thingspeak.
Waiting...
HTTP Response code: 200
Can1: 2 Can2: 0 Can3: 0  Send to Thingspeak.
Waiting...
```

Fig: Serial monitor showing that individual candidate vote counts are sent to thingspeak. When there is no new vote the same digits are sent to Thingspeak. Here, initially vote count for candidate 1,2, and 3 were 1, 0 and 0. After sometime, the vote count is changed to 2,0 and 0 which is being updated to thingspeak.



Last entry: less than a minute ago
Entries: 168

Candidate 1: 2 (a few seconds ago)
Candidate 2: 0 (a few seconds ago)
Candidate 3: 0 (a few seconds ago)

Fig: Live monitoring in thingspeak channel. Total vote count of each candidate is displayed on the numerical dashboard. The channel view is kept in private mode so that only the administration can monitor the live update. However, the view can be changed to public mode.

4.3 GitHub Link

https://github.com/Aroni-Ghosh/EVM_G2_EEE416

4.4 YouTube Link

5 Design Analysis and Evaluation

5.1 Novelty

We have generalized the Electronic Voting Machine for different kinds of elections where the number of candidates and voters may vary. Also, we have used IOT to show live results which will ensure fairness in voting count and make the system easy to use for every citizen of a country.

5.2 Project Management and Cost Analysis

5.2.1 Bill of Materials

Component	Per unit Price (Tk.)	Total Price (Tk.)
Arduino Uno -1 unit	925	925
Fingerprint Sensor R305	1700	1700
Breadboard- 3 units	85	255
ESP32- 1 unit	400	400
16x2 LCD- 1 unit	140	140
Buzzer- 1 unit	15	15
Jumper Wire	45	45
Push Button- 10 units	3	30
10k Pot- 1 unit	15	15
9V Battery -2 units	60	120
PCB Printing	600	600
	Total Materials cost=	Tk.4245

5.2.2 Calculation of Per Unit Cost of Prototype:

The cost per unit prototype will be 4245 Tk.\-

5.2.3 Calculation of Per Unit Cost of Mass-Produced Unit

If we do mass production, the cost will be reduced.

In an industry, when mass production goes online, per PCB sheet cost should be reduced by at least 30% and fall to 400. We can assume the rest of the components cost should be reduced by 20% too. Hence per prototype the cost will be around Tk. 2000.

5.2.4 Timeline of Project Implementation

Date	Task Implemented
16-07-2022	Equipment bought from Patuatuli
18-07-2022	Initial EVM circuit setup without R305 module
27-07-2022	Problem of R305 sensor was solved
28-07-2022	Second EVM circuit implemented but LCD could not detect sensor
12-08-2022	LCD and R305 interfacing error solved
17-08-2022	Default sensor data sent from ESP32 to IOT
21-08-2022	Different logical errors in code were handled successfully
26-08-2022	Vote Data from Arduino to ESP 32 and ESP32 to ThingSpeak sent successfully
27-08-2022	PCB Design completed and sent for printing
29-08-2022	PCB Board obtained but…

5.3 Practical Considerations of the Design to Address Public Health and Safety, Environment, Cultural, and Societal Needs

5.3.1 Considerations to public health and safety

The Voting system which is now used for the elections in Bangladesh is a slow process which takes a lot of time, as a result people have to stand in the line for a long time which sometimes may deteriorate the health of voters, so our goal is to build a fast Electronic Voting system.

We also tried to produce a system which will cause less environmental pollution which will ensure a less polluted environment for the citizens to live in.

To ensure public safety we try to build a live result monitoring system which will let the voters know the live result by staying at home.

5.3.2 Considerations to environment

In our EVM system we try to lessen the wastage production that is made from the existing system. So first our goal is not to use any kind of paper in our EVM system.

We also tried to make the EVM small so the transportation of the machine will become easy and flexible, causing less environmental pollution.

As we are not using any paper in our EVM there will be no need of cutting trees. This will ensure us a more clean, beautiful, greener environment.

5.3.3 Considerations to cultural and societal needs

Our goal was to make an Electronic Voting System that will be easy for everyone to use whether the person is educated or not educated or disable person. And we were able to create such a machine that is easy for everyone to use.
Our EVM machine can be used for any language which overcomes the language barrier.
Our other goal was to ensure that a person can't give a counterfeit vote. As we are using biometric system, we can ensure the civilized right of every citizen.

5.4 Assessment of the Impact of the Project on Societal, Health, Safety, Legal and Cultural Issues

5.4.1 Assessment of Societal Issues

The system is made in such a way that there will be no counterfeit vote which will ensure the voting right of everyone.

This system is very easy to use. People from all ages, occupation can use it easily. The function of the system is also very easily understandable.

The cost of preparing the machine is low, so government can save money which can be used to make the people life much better

5.4.2 Assessment of Health and Safety Issues

Our EVM will ensure a fast-voting system, as a result there will not be a long line of voters.

Our system will ensure a transparent and peaceful election which will help to ensure the safety of people.

5.4.3 Assessment of Legal Issues

As we are using a biometric system to recognize there will be no counterfeit vote or anyone cannot vote two times.

The live monitoring system will help us to ensure a transparent election.

5.4.4 Assessment of Cultural Issues

We make our EVM in a way so that elections can be done in different languages.

5.5 Evaluation of the Sustainability the and Impact of the Designed Solution in the Societal and Environmental Contexts

5.5.1 Evaluation of Sustainability

The EVM we have built is very much user friendly. People from all ages, culture can use it.

The working procedure of our EVM is also very easy. So, a simple training is needed to learn how to use it and debug it.

As it is very small in size the transportation is much easier than the voting system that is used now.

So, the comparisons show us that the sustainability of our EVM system is much more better than the existing one.

5.5.2 Evaluation of Impact of Design in Societal Context

Our EVM will ensure a transparent election and it is easy to use for everyone even for illiterate or. disabled people.

Even the cost of making the system is very less than the existing one. So, we can say our EVM is a better option than the existed one in terms of societal context

5.5.3 Evaluation of Impact of Design in Environmental Context

Our Electronic Voting system is environmentally friendly. The production of wastage is very low. There is no need for ballot papers for our project.

The transportation of the system is very easy. We know plastic is very harmful for our environment but there is no use of plastic in the project.

The declutter of the system is also not difficult and the parts can be used for another project ensuring better waste management. So, the performance of our one is much better to reduce environmental pollution.

6 Reflection on Individual and Team work

6.1 Individual Contribution of Each Member

Member ID	Individual Contribution
1706002	Hardware assembling
1706004	Software Coding for the base EVM and IoT based live monitoring
1706024	Connection setup for IoT, buying equipment
1706032	Schematic and PCB Design, buying equipment

6.2 Mode of Teamwork

Each member gathered together in central library and worked for hours. When one member did hardware setup, another member did the coding part, another member did field work like buying equipment from market and another member provided different online references and supplementary help.

6.3 Diversity Statement of Team

- Two Female members, Two male members
- Members from both Hall resident and attached
- Members are from different districts.

6.4 Log Book of Project Implementation

Date	Milestone achieved	Individual Role	Team Role	Comments
16-07-2022	Equipment bought from Patuatuli	By 1706032		
18-07-2022	Initial Prototype of an EVM was assembled	By 1706002	Team provided schematics and software code	
27-07-2022	Problem of R305 sensor was solved	By 1706002-Hardware debugging, 1706004-software debugging	Team provided online references	
28-07-2022	Second EVM circuit implemented but LCD could not detect sensor	By 1706002-Hardware debugging, 1706004-software debugging	Team provided online references	
12-08-2022	LCD and R305 interfacing error solved	By 1706004-Hardware and software debugging	No team Role	
17-08-2022	Default sensor data sent from ESP32 to IOT	By 1706004-Hardware and software setup	No team role	
21-08-2022	Different logical errors in code were handled successfully	By 1706002-hardware debug By 1706004, 1706024- logical debugging		
26-08-2022	Vote Data from Arduino to ESP 32 and ESP32 to ThingSpeak sent successfully	By 1706002-Hardware Debug, by 1706004-Programming, 1706024- ESP32 setup 1706032- provided Efficient Code		
27-08-2022	PCB Design completed and sent for printing	By 1706032- PCB Design	Cross Checking design	
28-08-2022	Sending data to IOT with generalized code	By 1706004	Team provided online references	
29-08-2022	Soldering done for PCB	By 1706002, 1706032	No Team Role	
30-08-2022	Went to Patuatuli to bought R305 module twice	By 1706024, 1706032		

7 References

- *Fingerprint Based Biometric Voting Machine Project using Arduino (circuitdigest.com)*
- *IRJET- Electronic Voting Machine (EVM) using Finger-Print Sensor (slideshare.net)*
- *https://www.researchgate.net/publication/351638332*