



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

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

FALL SEMESTER 2023-24

DIGITAL ASSIGNMENT - 5

PROGRAMME NAME: BTECH IT

CLASS ID- VL2023240100166

FACULTY NAME: PROF. ATHIRA P SHAJI

COURSE CODE: BITE403P

CLASS GROUP: UGS04

COURSE TITLE: EMBEDDED SYSTEMS AND IOT LAB

TITLE :

**FINGERPRINT RECOGNITION AND
HOME SECURITY SYSTEM**

TEAMMATES:

SHRINIDHI B (21BIT0381)

PREETHI S (21BIT0349)

JUMANA BEGUM M (21BIT0694)

SHIVANI K (21BIT0681)

PANCHAL NAYAK (21BIT0080)

Project implementation video link:

[https://drive.google.com/file/d/1xAOTJZhGLhsAchJqUUer4Q0hGq8fh-tS/view?usp=drive link](https://drive.google.com/file/d/1xAOTJZhGLhsAchJqUUer4Q0hGq8fh-tS/view?usp=drive_link)

TABLE OF CONTENT:

S.No	Topic	Page no.
1.	AIM	3
2.	ABSTRACT	3
3.	INTRODUCTION	3
4.	HARDWARE REQUIREMENTS	4
5.	SOFTWARE REQUIREMENTS	4
6.	CODING	5
7.	OUTPUT	10
8.	RESULT AND DISCUSSION	22

AIM:

The aim of a biometric home security system is to provide enhanced security and convenience through seamless authentication and personalized access control.

ABSTRACT:

From earlier times, security has been, and still is, an issue of concern in our households, offices, shops, and other places. Everyone fears the possibility of unauthorized individuals entering their homes or offices without their knowledge. While normal doors can be fitted with locks, these locks can be broken using an alternate key. Alternative systems such as password or pattern locks are also vulnerable to being exposed and opened.

To address these issues, one possible solution is to combine door locks with biometrics. Biometric verification refers to any method by which a person can be uniquely identified based on one or more distinguishing biological traits. These unique identifiers can include fingerprints, hand geometry, earlobe geometry, retina and iris patterns, voice waves, DNA, and signatures. In this case, we will use fingerprints for biometric verification, as fingerprints are unique to each individual. By using fingerprints as the key to door locks, we can significantly enhance security and minimize the risk of unauthorized individuals trespassing into our homes, shops, offices, and other places, as duplicating such a key is not possible.

Moreover, this system eliminates the problems associated with losing keys, as carrying keys becomes unnecessary when this system is used instead of traditional locks. To implement this system with enhanced security features, we will utilize Arduino technology.

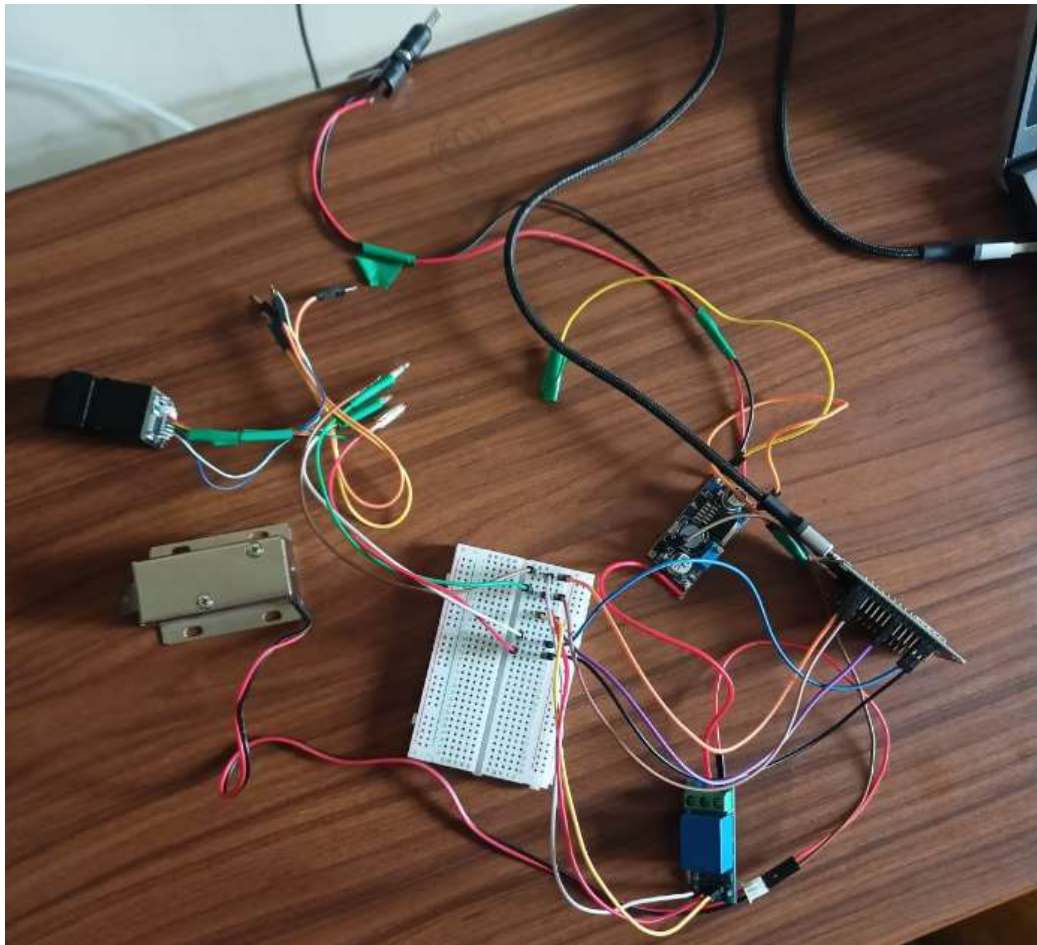
INTRODUCTION:

In this project, we aim to address security issues in homes, shops, and offices. While traditional locks can provide a solution, there is a possibility of unauthorized individuals opening the lock using duplicate keys. Furthermore, the inconvenience of carrying and potentially losing keys can cause additional problems. Although pattern-based locks can enhance security, they can still be compromised if someone manages to guess the passwords or patterns.

To overcome these limitations, we propose using fingerprints as the key for our advanced security lock system. By implementing this project with Arduino NANO, we can incorporate various devices and features to maximize security levels.

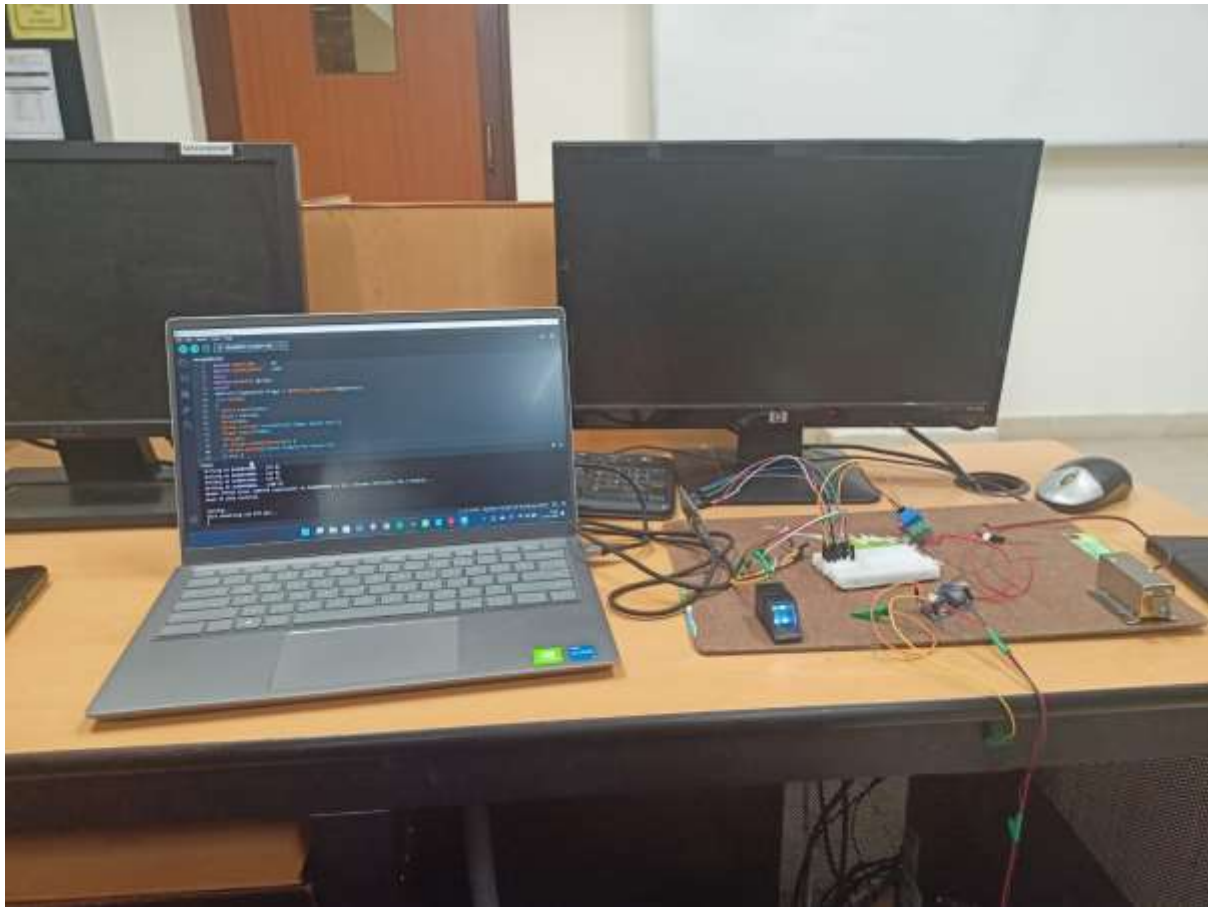
HARDWARE REQUIREMENTS:

- RR307 Fingerprint sensor
- NODE MCU
- jumper wires
- DC Voltage Regulator
- Relay Module
- Solenoid Lock
- 12V DC adapter
- Breadboard



SOFTWARE REQUIREMENTS:

- Arduino IDE
- Libraries:
 - Software serial
 - Adafruit fingerprint



CODING:

Fingerprint Registration:

Code:

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

#define RX_PIN 4
#define TX_PIN 5

SoftwareSerial mySerial(RX_PIN, TX_PIN);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

uint8_t id;

void setup()
{
  Serial.begin(9600);
  while (!Serial); // For boards like NodeMCU, Wemos, etc.
```

```

delay(100);
Serial.println("\n\nAdafruit Fingerprint sensor enrollment");

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()
{
    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.println(".");
                break;
            case FINGERPRINT_PACKETRECEIVEERR:
                Serial.println("Communication error");
                break;
            case FINGERPRINT_IMAGEFAIL:
                Serial.println("Imaging error");
                break;
            default:
                Serial.println("Unknown error");
                break;
        }
    }

    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_PACKETRECEIVEERR:
    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_PACKETRECEIVEERR:
            Serial.println("Communication error");
            break;
        case FINGERPRINT_IMAGEFAIL:
            Serial.println("Imaging error");
            break;
    }
}

```



```

        default:
            Serial.println("Unknown error");
            break;
    }
}

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;
}

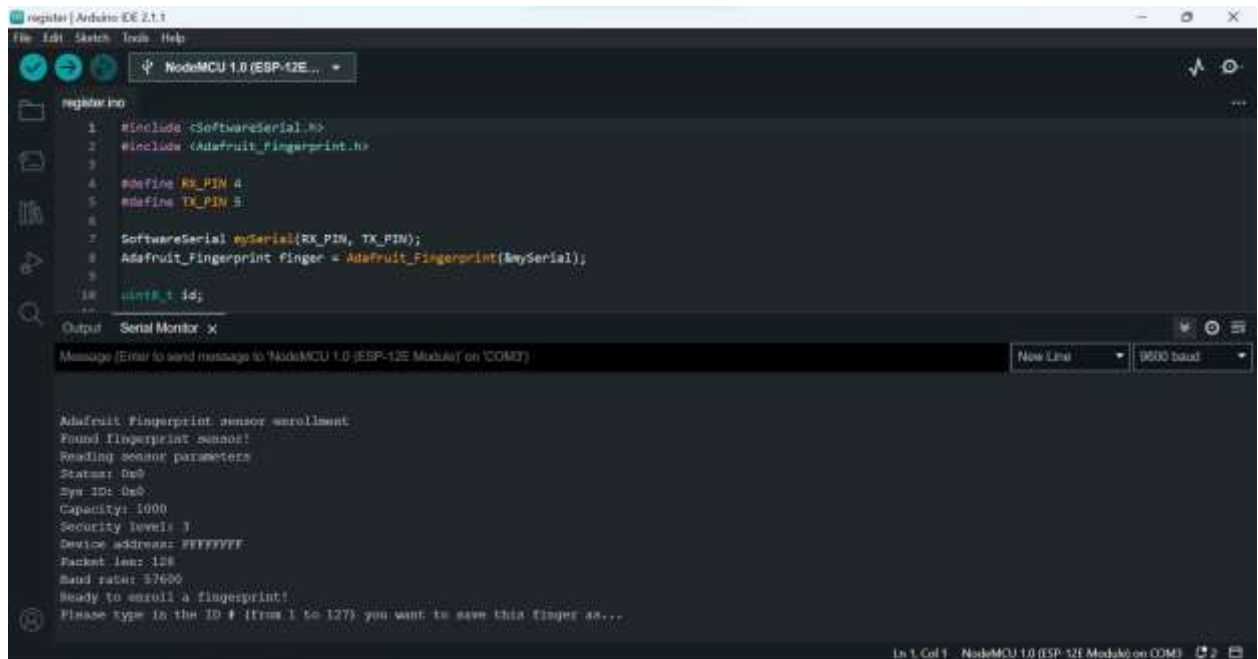
Serial.print("Creating model for #");
Serial.println(id);

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

```

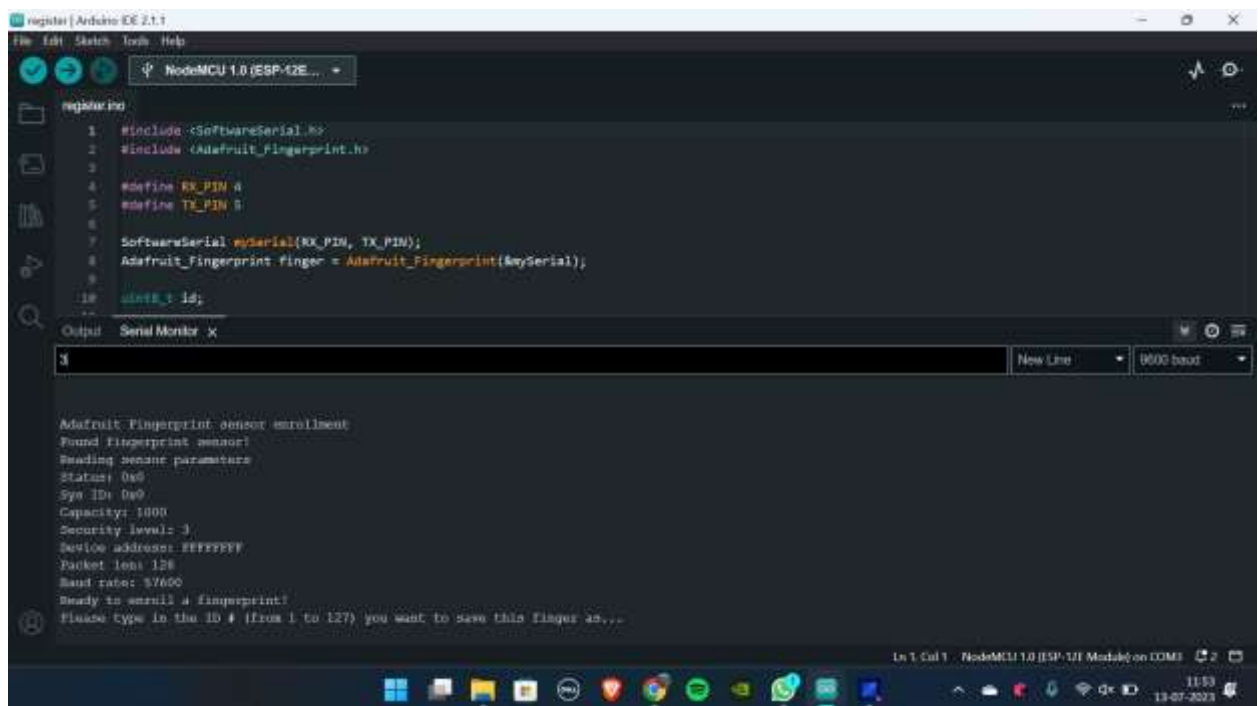
```
{  
    Serial.println("Unknown error");  
    return p;  
}  
  
Serial.print("ID ");  
Serial.println(id);  
p = finger.storeModel(id);  
if (p == FINGERPRINT_OK)  
{  
    Serial.println("Stored!");  
}  
else if (p == FINGERPRINT_PACKETRECEIVEERR)  
{  
    Serial.println("Communication error");  
    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;  
}
```

OUTPUT:

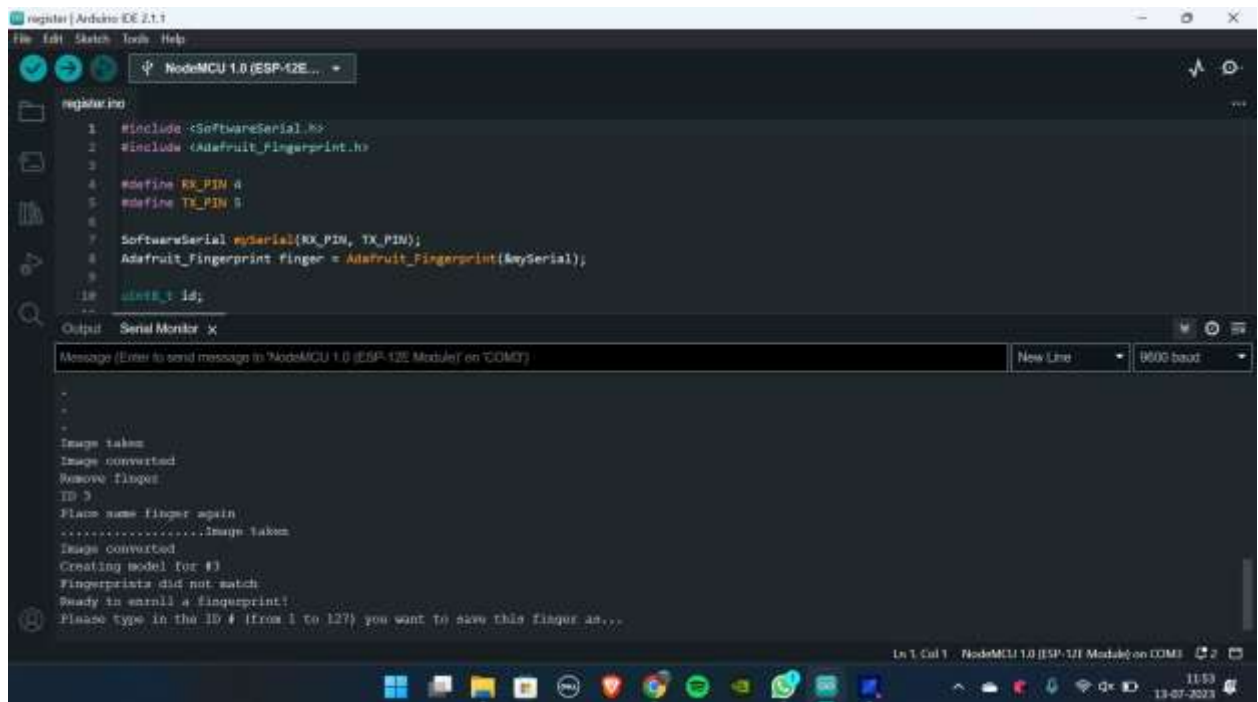
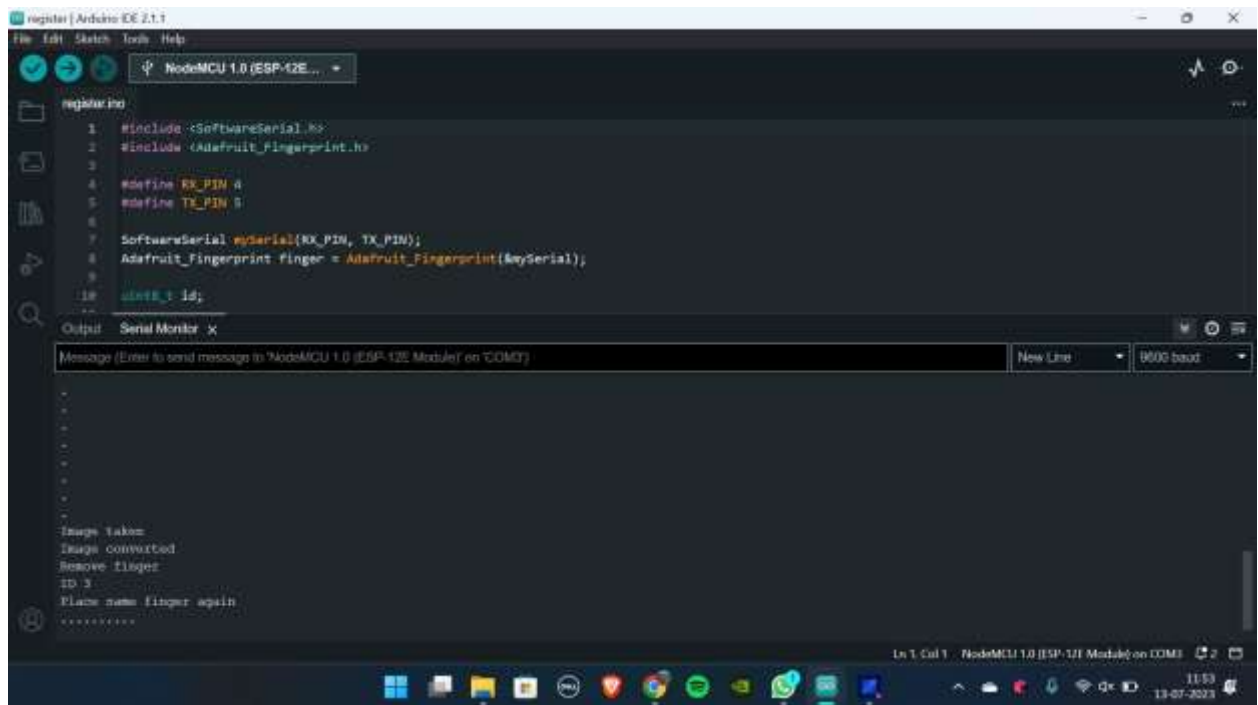


The screenshot shows the Arduino IDE interface with the 'Serial Monitor' window open. The code in the background is for initializing the Adafruit Fingerprint sensor. The output in the Serial Monitor shows the following messages:

```
Adafruit Fingerprint sensor enrollment
Found fingerprint sensor!
Reading sensor parameters
Status: 0x0
Sys ID: 0x0
Capacity: 1000
Security level: 3
Device address: 0xFFFFFFFF
Packet len: 128
Baud rate: 57600
Ready to enroll a fingerprint!
Please type in the ID # (from 1 to 127) you want to save this finger as...
```



This screenshot is similar to the one above, but the 'Serial Monitor' window now shows the number '1' entered in the input field, indicating the user has selected the ID they want to save the fingerprint as. The rest of the output text remains the same.





Recognition:

Code:

```
#include <Adafruit_Fingerprint.h>
#if (defined(__AVR__) || defined(ESP8266)) && !defined(__AVR_ATmega2560__)
SoftwareSerial mySerial(4, 5);
#define RELAY_PIN      D5
#define ACCESS_DELAY    1000
#else
#define mySerial Serial1
#endif
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
void setup()
{
  Serial.begin(9600);
  while (!Serial);
  delay(100);
  Serial.println("\n\nAdafruit finger detect test");
  finger.begin(57600);
  delay(5);
  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);

  finger.getTemplateCount();

  if (finger.templateCount == 0) {
    Serial.print("Sensor doesn't contain any fingerprint data. Please run the
'enroll' example.");
  }
  else {
    Serial.println("Waiting for valid finger...");
    Serial.print("Sensor contains "); Serial.print(finger.templateCount);
    Serial.println(" templates");
  }
}
```

```

    }
    pinMode(RELAY_PIN, OUTPUT);
    digitalWrite(RELAY_PIN, HIGH);
}

void loop()                                // run over and over again
{
    getFingerprintID();
    delay(50);                             //don't ned to run this at full speed.
}

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;
    }
    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.fingerSearch();
    if (p == FINGERPRINT_OK) {
        Serial.println("Found a print match!");
    } else if (p == FINGERPRINT_PACKETRECEIVEERR) {
        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);
    digitalWrite(RELAY_PIN, LOW);
    delay(ACCESS_DELAY);
    digitalWrite(RELAY_PIN, HIGH);

    return finger.fingerID;
}

int getFingerprintIDez() {
    uint8_t p = finger.getImage();
    if (p != FINGERPRINT_OK) return -1;

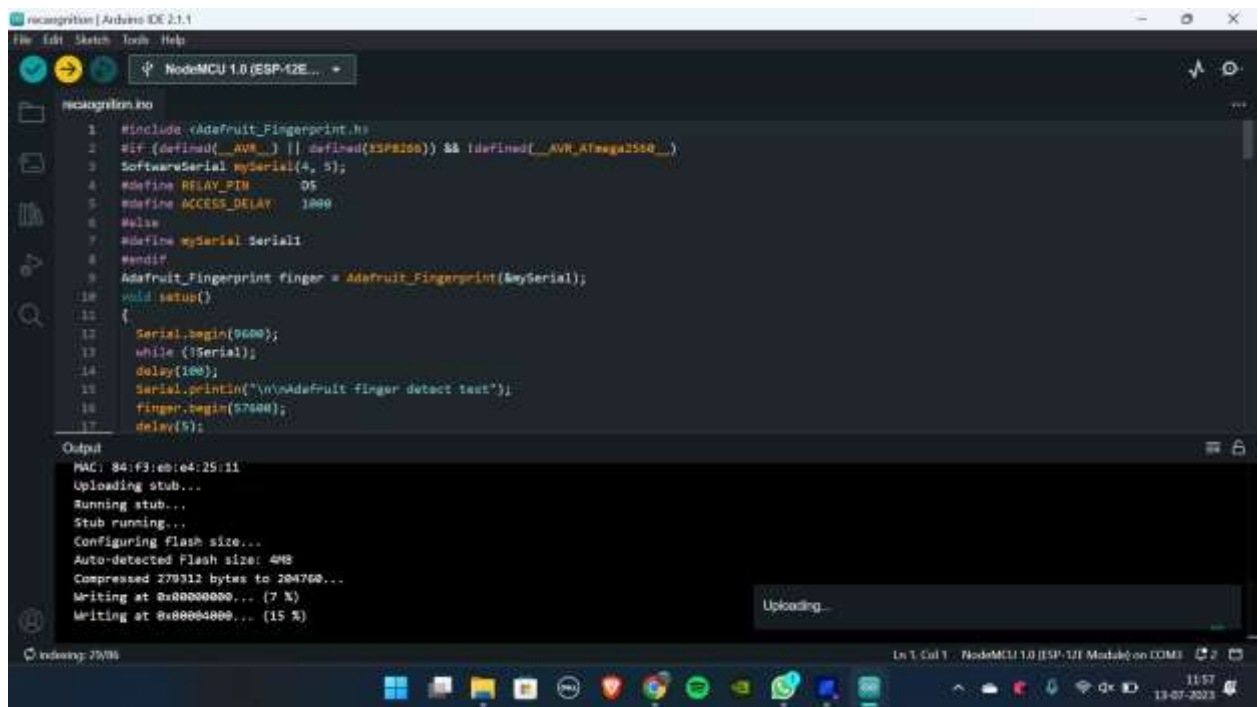
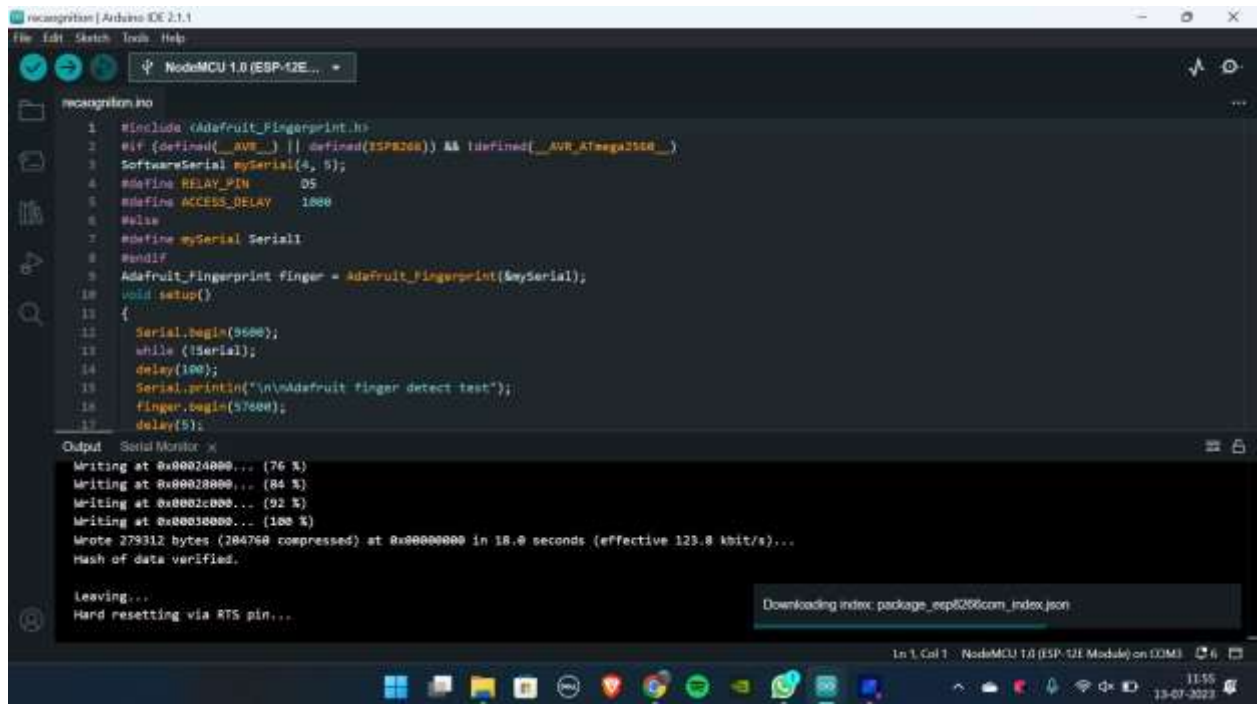
    p = finger.image2Tz();
    if (p != FINGERPRINT_OK) return -1;

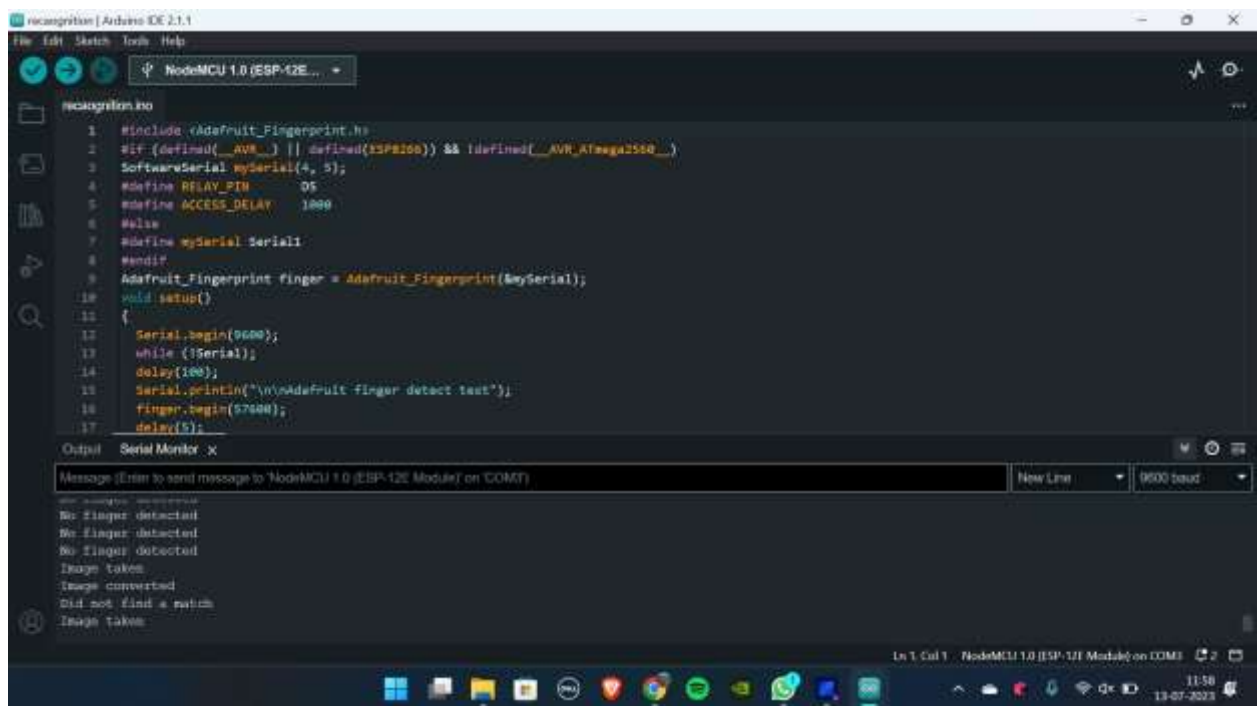
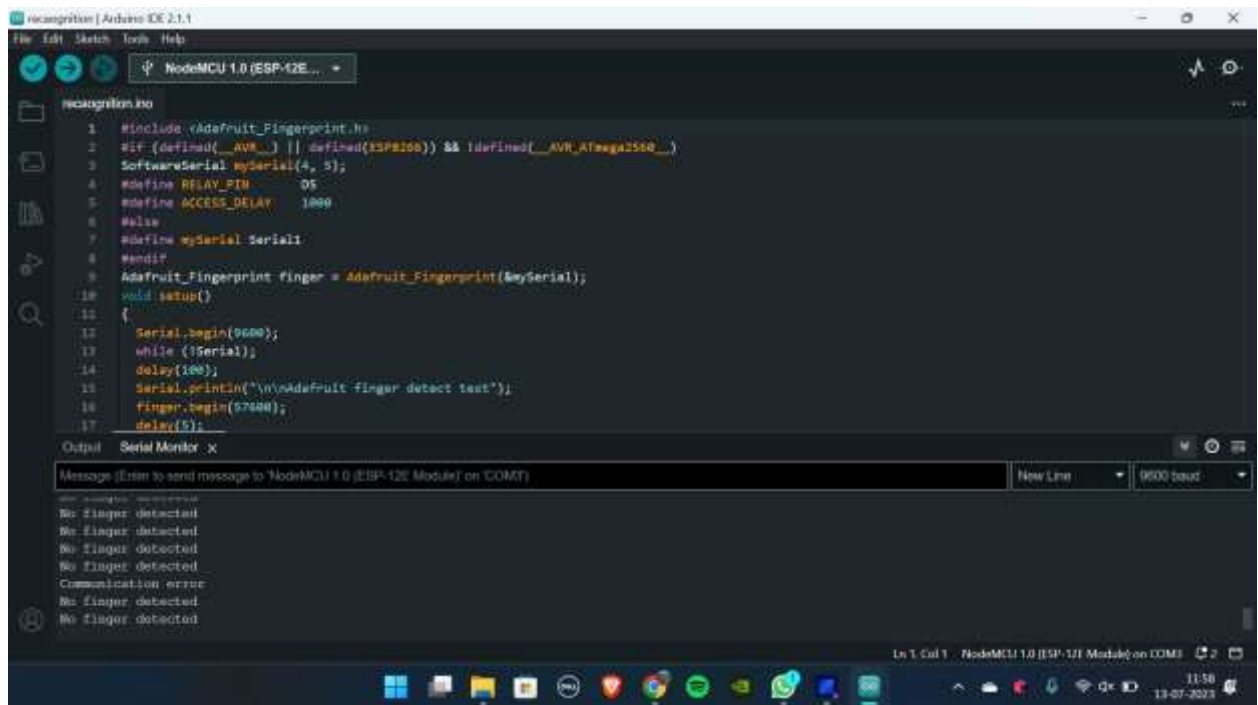
    p = finger.fingerFastSearch();
    if (p != FINGERPRINT_OK) return -1;

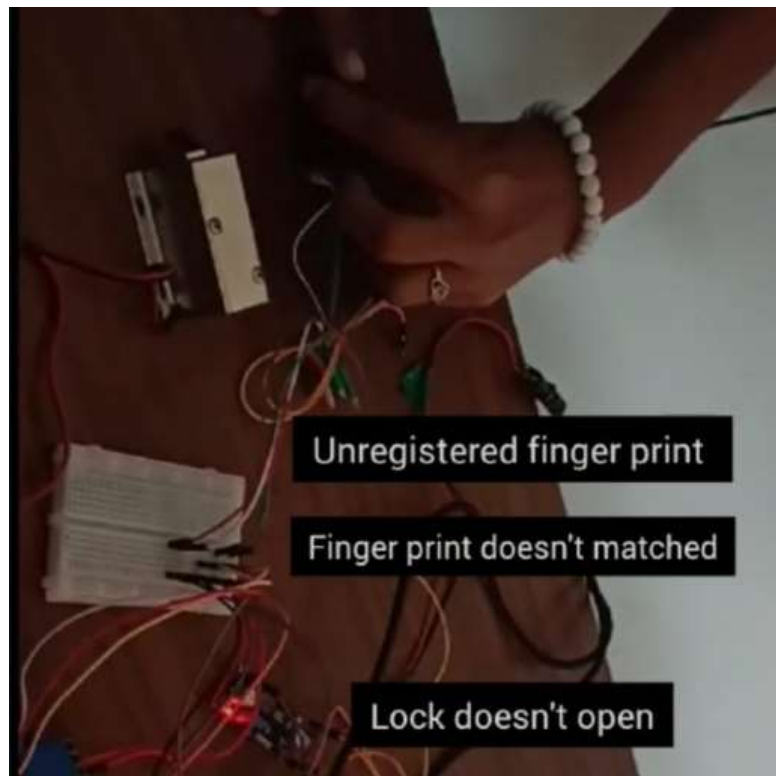
    // found a match!
    Serial.print("Found ID #"); Serial.print(finger.fingerID);
    Serial.print(" with confidence of "); Serial.println(finger.confidence);
    return finger.fingerID;
}

```


OUTPUT:







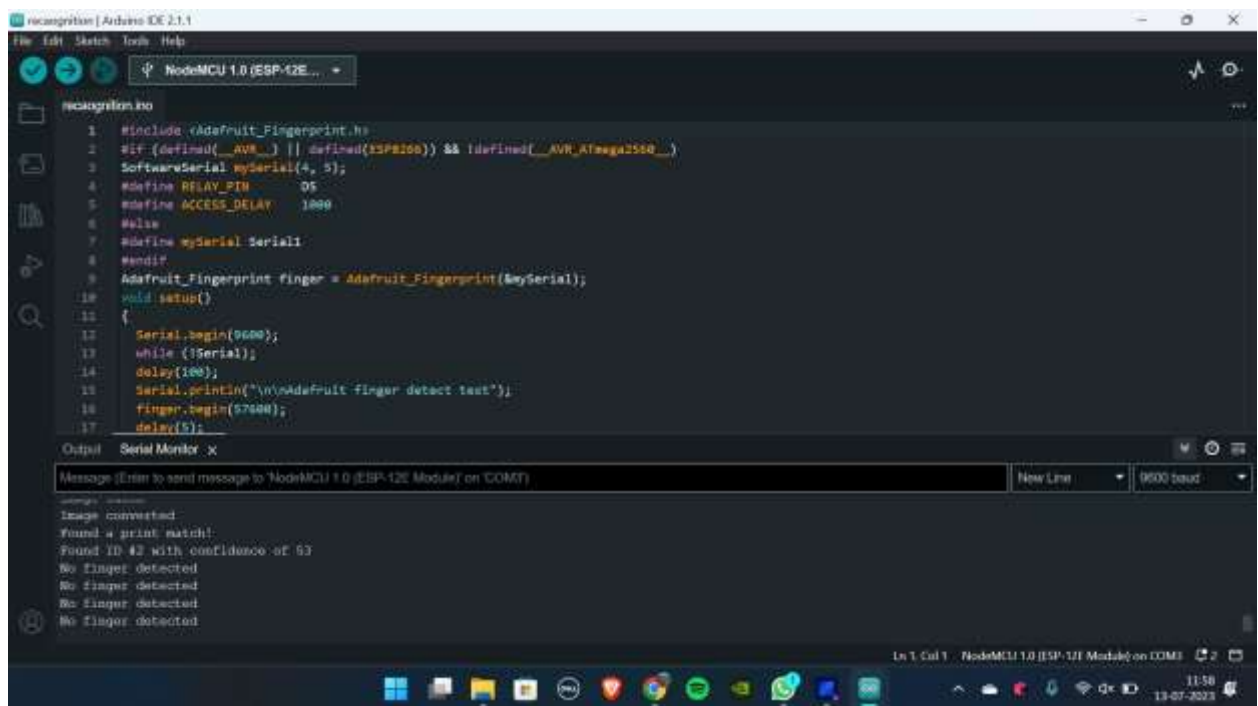
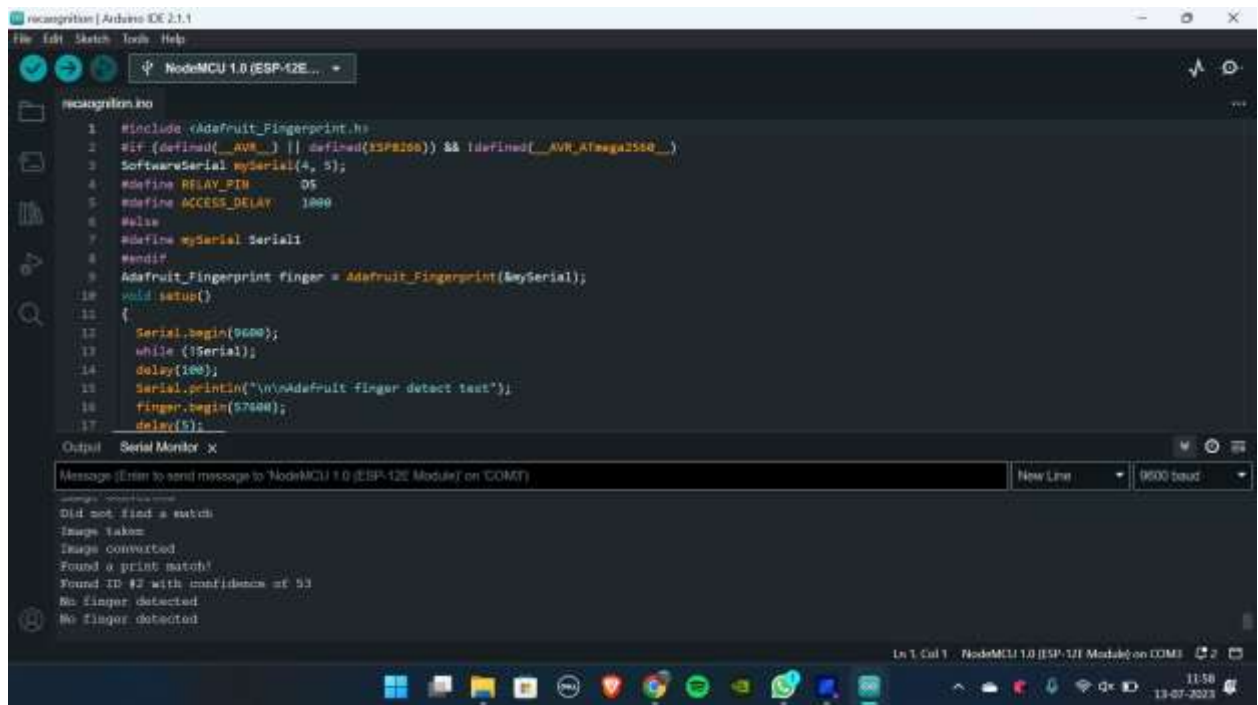
```
recognition | Arduino IDE 2.1.1
File Edit Sketch Tools Help
NodeMCU 1.0 (ESP-12E...)
recognition.ino
1 #include <Adafruit_Fingerprint.h>
2 #if defined(__AVR__) || defined(ESP8266) && !defined(__AVR_ATmega2560__)
3   SoftwareSerial mySerial(4, 5);
4   #define RELAY_PIN    05
5   #define ACCESS_DELAY 1000
6   #else
7   #define mySerial Serial1
8   #endif
9   Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
10 void setup()
11 {
12   Serial.begin(9600);
13   while (!Serial);
14   delay(100);
15   Serial.println("\n\nAdafruit finger detect test");
16   finger.begin(57600);
17   delay(5);
18 }
19
20 void loop()
21 {
22   if (finger.getImage())
23   {
24     Serial.println("Image taken");
25     if (finger.convertImage())
26     {
27       Serial.println("Image converted");
28       if (finger.match())
29       {
30         Serial.println("Did not find a match");
31       }
32       else
33       {
34         Serial.println("Image taken");
35         if (finger.convertImage())
36         {
37           Serial.println("Image converted");
38           if (finger.match())
39           {
40             Serial.println("Found a print match!");
41             Serial.println("Found ID #2 with confidence of 53");
42           }
43         }
44       }
45     }
46   }
47 }
```

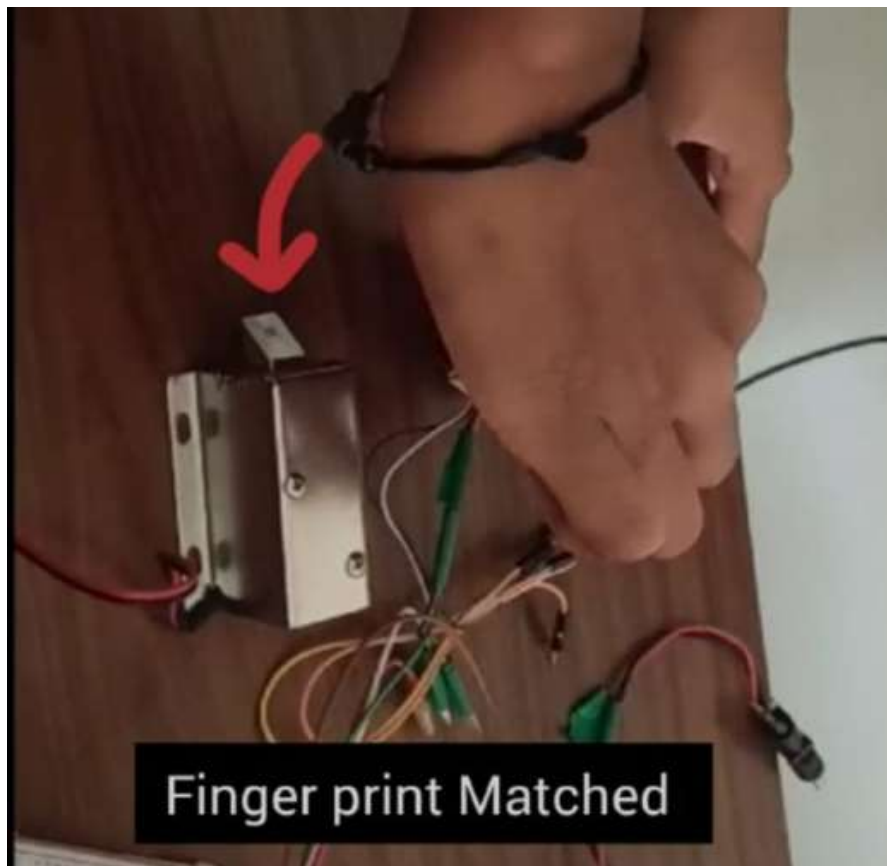
Output Serial Monitor x

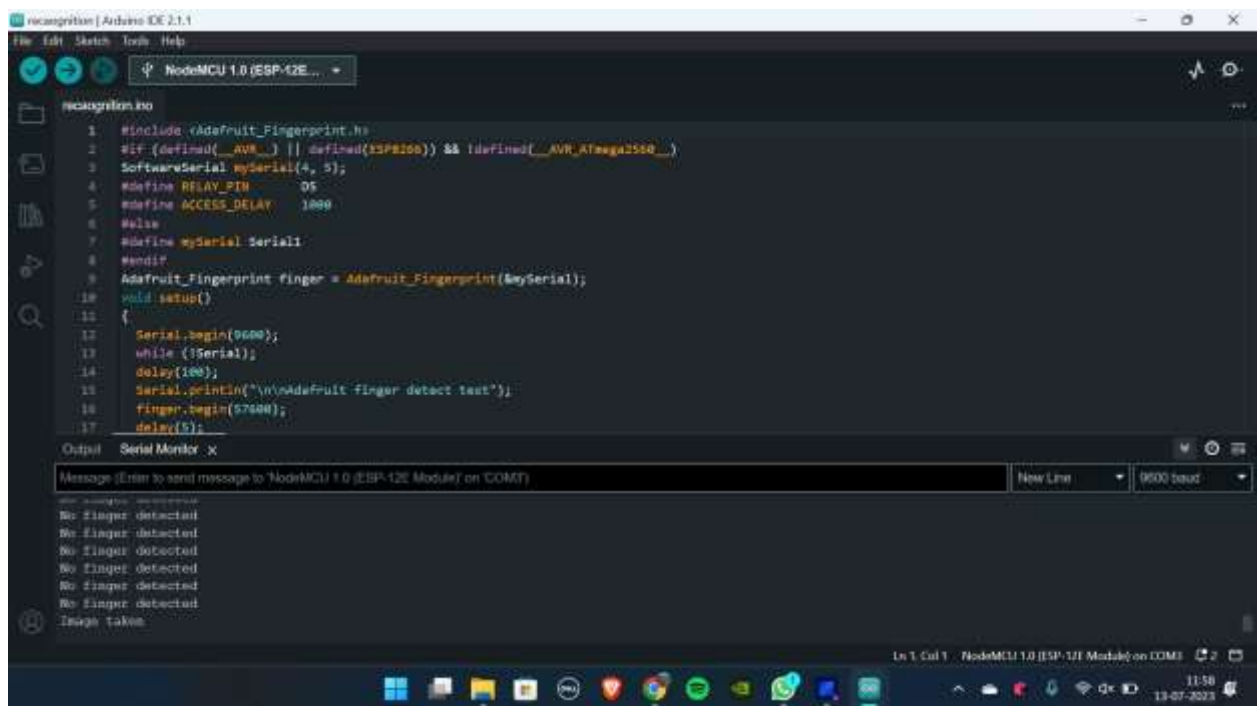
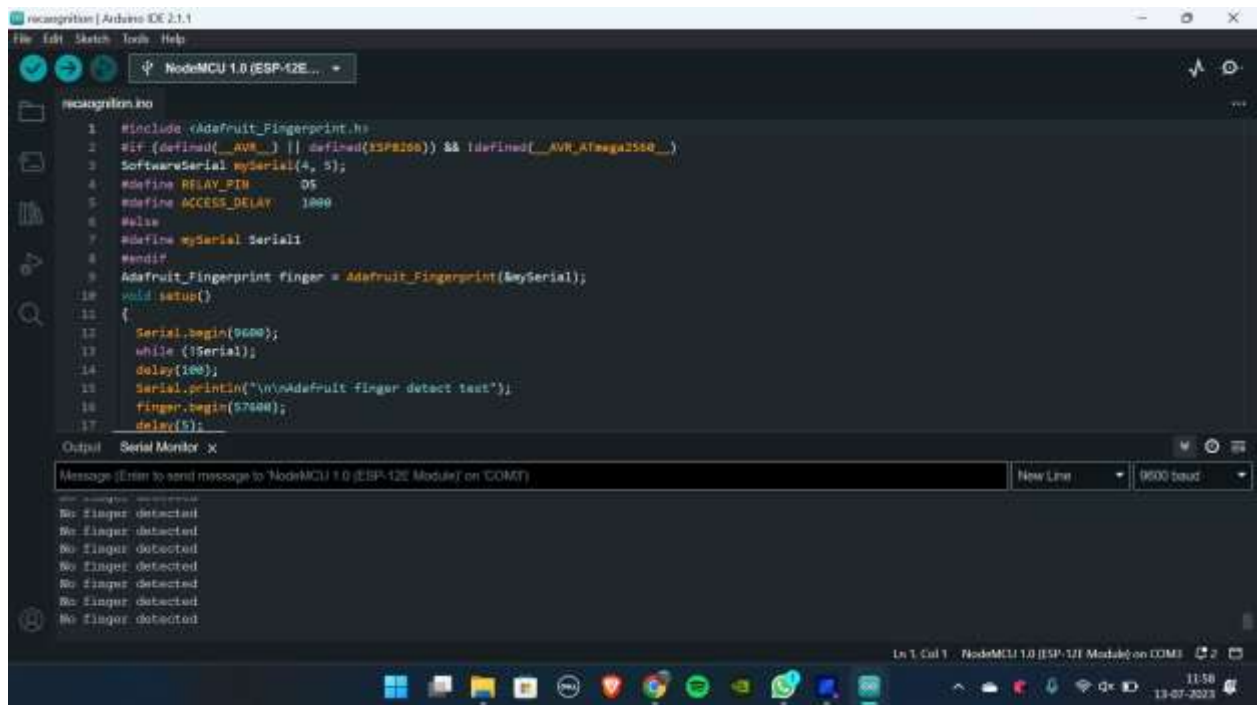
Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM3')

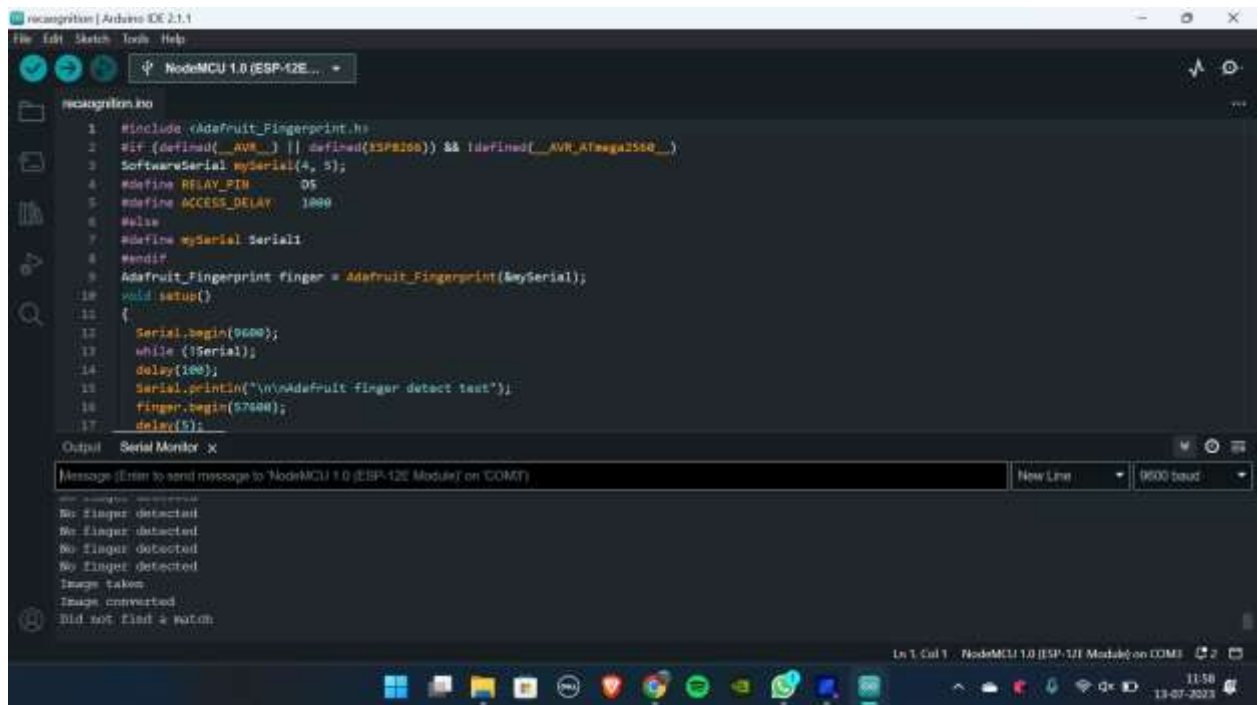
New Line 9600 baud

Ln 1, Col 1 NodeMCU 1.0 (ESP-12E Module) on COM3 11:58 13-07-2023









RESULTS AND DISCUSSION:

The design and implementation of a fingerprint-based door lock system are customizable and flexible. This door locking mechanism is comparatively cost-effective compared to the available lock systems in the traditional market. Our fingerprint-based lock system offers a high accuracy rate and quick recognition of fingerprints, enabling seamless integration with users and providing enhanced security. Many companies are interested in adopting this type of locking mechanism, but the existing systems on the market have very high installation costs.

One of the main advantages of this system is its flexibility, allowing for the implementation of several other systems. Additionally, the system is highly secure. Fingerprints are unique, and the sensor is capable of identifying all prints during testing.