RV INSTITUTE OF TECHNOLOGY AND MANAGEMENT®



**BIOLOGY FOR ENGINEERS**

ASSIGNMENT - 2

TOPIC: REPORT ON FINGERPRINT SENSOR MANAGEMENT SYSTEM WITH SERVOMOTOR INTEGRATION

NAME: VAJRAMANASA PALGUNA H V, SUMUKH SHARMA, MANAS KALOSE

USN: 1RF22EC060, 1RF22EC054, 1RF23EC400

SEM: 4TH SEM ECE

**TABLE OF CONTENTS:**

1. INTRODUCTION – OVERVIEW, OBEJCTIVES
2. SYSTEM COMPONENTS - HARWARE AND SOFTWARE
3. CIRCUIT DESIGN - SCHEMATIC DIAGRAM, PIN CONNECTIONS
4. SOFTWARE IMPLEMENTATION – INITIALIZATION AND SETUP, ENROLL FINGERPRINT FUNCTION, VERIFY FIGNERPRINT FUNCTION, DELETE FINGERPRINT FUNCTION
5. OPERATION MODES – ENROLLMENT MODES, VERIFICATION MODE, DELETION MODE
6. TESTING AND RESULTS – TEST SCENARIOS
7. OBSERVATIONS – PERFORMANCE ANALYSIS, CONCLUSION, SUMMARY, FUTURE ENHANCEMENTS, REFERENCES

**INTRODUCTION:**

**OVERVIEW**:

This report presents the design and implementation of a fingerprint sensor management system integrated with a servo motor. The system allows for three primary functions: enrolling fingerprints, verifying fingerprints, and deleting fingerprints. Upon successful verification of a fingerprint, a servo motor is activated to perform a mechanical action, demonstrating the potential application of fingerprint recognition in security systems.

**OBJECTIVES**:

The primary objectives of this project are to design a system that can enroll, verify, and delete fingerprints using the Adafruit Fingerprint Sensor. Additionally, the system will integrate a servo motor that responds to successful fingerprint verification. A user-friendly interface will be provided through serial communication, allowing users to select different operation modes easily.

**2. SYSTEM COMPONENTS**

The hardware components for this project include the Adafruit Fingerprint Sensor, which is a capacitive fingerprint sensor module used for capturing and processing fingerprint images. The Servo Motor (SG90) is a small servomotor that demonstrates an action triggered by fingerprint verification. The Arduino Microcontroller serves as the central processing unit, interfacing with both the fingerprint sensor and the servo motor to coordinate their functions.

**THE SOFTWARE COMPONENTS:**

For this project include the Arduino IDE, which is the development environment used for writing and uploading code to the Arduino microcontroller. The Adafruit Fingerprint Library simplifies the process of interfacing with the fingerprint sensor, while the Servo Library facilitates the control of servo motors. These libraries and tools work together to ensure smooth operation and interaction between the system's hardware components.

## CIRCUIT DESIGN:

**PIN CONNECTIONS**

**FINGERPRINT SENSOR**

- TX to Arduino Pin 2 (or Serial1 TX on Mega)

- RX to Arduino Pin 3 (or Serial1 RX on Mega)

- VCC to 5V

- GND to GND

**SERVO MOTOR**

- Signal to Arduino Pin 9

- VCC to 5V

**4. SOFTWARE IMPLEMENTATION:**

**INTIALIZATION AND SETUP**

The `setup()` function initializes serial communication and the fingerprint sensor, verifies the sensor's password, and sets the initial position of the servo motor.

**ENROLL FINGERPRINT FUNCTION**

The `enrollFingerprint()` function guides the user through enrolling a new fingerprint. It captures two images of the fingerprint, converts them into a model, and stores this model in the sensor’s memory.

**VERIFY FINGERPRINT FUNCTION**

The `verifyFingerprint()` function continuously checks for a fingerprint on the sensor. If a match is found, the system activates the servo motor to perform a predefined action.

**DELETE FINGERPRINT FUNCTION**

The `deleteFingerprint()` function allows the user to remove a specific fingerprint from the sensor’s memory. It matches the fingerprint with stored records and deletes it upon confirmation.

**5. OPERATION MODES**

**ENROLLMENT MODE**

In this mode, the user can enroll a new fingerprint by entering an ID. The fingerprint is stored in the sensor’s memory under the provided ID.

**VERIFICATION MODE**

This mode is used to verify a fingerprint against stored records. Upon successful verification, the servo motor is activated.

**DELETION MODE**

The user can delete a stored fingerprint by entering the corresponding ID or by placing the finger on the sensor. The fingerprint is then removed from the memory.

**6. PROGRAM FOR THE FINGERPRINT SENSOR:**

#include <Adafruit\_Fingerprint.h>

#include <Servo.h>

#if (defined(\_\_AVR\_\_) || defined(ESP8266)) && !defined(\_\_AVR\_ATmega2560\_\_)

SoftwareSerial mySerial(2, 3);

#else

#define mySerial Serial1

#endif

Adafruit\_Fingerprint finger = Adafruit\_Fingerprint(&mySerial);

Servo myServo;

const int servoPin = 9;

uint8\_t id;

void setup() {

Serial.begin(9600);

while (!Serial);

delay(100);

Serial.println("\n\nFingerprint Sensor Management");

// Initialize fingerprint sensor

finger.begin(57600);

if (finger.verifyPassword()) {

Serial.println("Found fingerprint sensor!");

} else {

Serial.println("Did not find fingerprint sensor :(");

while (1) { delay(1); }

}

myServo.attach(servoPin); // Attaches the servo on pin 9 to the servo object

myServo.write(0); // Initialize servo position

}

void loop() {

Serial.println("Select mode: (e)nroll, (v)erify, (d)elete");

while (!Serial.available());

char mode = Serial.read();

// Clear any remaining characters in the buffer (like newline)

while (Serial.available()) {

Serial.read();

}

switch (tolower(mode)) {

case 'e':

enrollFingerprint();

break;

case 'v':

verifyFingerprint();

break;

case 'd':

deleteFingerprint();

break;

default:

Serial.println("Invalid mode! Please select (e), (v), or (d).");

}

}

void enrollFingerprint() {

Serial.println("Enrolling fingerprint...");

Serial.println("Please type in the ID # (from 1 to 127) you want to save this finger as...");

id = readnumber();

if (id == 0) {

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("No finger detected");

break;

case FINGERPRINT\_PACKETRECIEVEERR:

Serial.println("Communication error");

break;

case FINGERPRINT\_IMAGEFAIL:

Serial.println("Imaging error");

break;

default:

Serial.println("Unknown error");

break;

}

}

p = finger.image2Tz(1);

if (p != FINGERPRINT\_OK) {

Serial.println("Error in converting image");

return p;

}

Serial.println("Remove finger");

delay(2000);

p = 0;

while (p != FINGERPRINT\_NOFINGER) {

p = finger.getImage();

}

Serial.print("ID "); Serial.println(id);

Serial.println("Place same finger again");

p = -1;

while (p != FINGERPRINT\_OK) {

p = finger.getImage();

switch (p) {

case FINGERPRINT\_OK:

Serial.println("Image taken");

break;

case FINGERPRINT\_NOFINGER:

Serial.println("No finger detected");

break;

case FINGERPRINT\_PACKETRECIEVEERR:

Serial.println("Communication error");

break;

case FINGERPRINT\_IMAGEFAIL:

Serial.println("Imaging error");

break;

default:

Serial.println("Unknown error");

break;

}

}

p = finger.image2Tz(2);

if (p != FINGERPRINT\_OK) {

Serial.println("Error in converting image");

return p;

}

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;

}

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;

}

void verifyFingerprint() {

Serial.println("Verifying fingerprint...");

Serial.println("Place your finger on the sensor");

while (true) { // Continuous loop

uint8\_t p = finger.getImage();

if (p != FINGERPRINT\_OK) {

switch (p) {

case FINGERPRINT\_NOFINGER:

Serial.println("Waiting for finger...");

break;

case FINGERPRINT\_PACKETRECIEVEERR:

Serial.println("Communication error");

break;

case FINGERPRINT\_IMAGEFAIL:

Serial.println("Imaging error");

break;

default:

Serial.println("Unknown error");

break;

}

delay(1000); // Wait a second before trying again

continue; // Skip the rest of the loop and start over

}

// If we get here, we've successfully captured an image

p = finger.image2Tz();

if (p != FINGERPRINT\_OK) {

Serial.println("Failed to convert image.");

continue;

}

p = finger.fingerFastSearch();

if (p == FINGERPRINT\_OK) {

Serial.print("Found ID #"); Serial.print(finger.fingerID);

Serial.print(" with confidence of "); Serial.println(finger.confidence);

// Fingerprint verified, activate servo

Serial.println("Fingerprint verified! Activating servo.");

myServo.write(90); // Turn servo to 90 degrees

delay(1000); // Keep servo activated for 1 second

myServo.write(0); // Return servo to initial position

} else if (p == FINGERPRINT\_NOTFOUND) {

Serial.println("Fingerprint not recognized.");

} else {

Serial.println("Error in fingerprint search.");

}

Serial.println("Verify another? (y/n)");

while (!Serial.available());

char response = Serial.read();

if (response != 'y' && response != 'Y') {

break; // Exit the verification loop

}

Serial.println("Place your finger on the sensor");

}

Serial.println("Exiting verification mode.");

}

void deleteFingerprint() {

Serial.println("Delete Fingerprint");

Serial.println("Place finger to delete on the sensor");

while (true) {

uint8\_t p = finger.getImage();

if (p != FINGERPRINT\_OK) {

switch (p) {

case FINGERPRINT\_NOFINGER:

Serial.println("Waiting for finger...");

break;

case FINGERPRINT\_PACKETRECIEVEERR:

Serial.println("Communication error");

break;

case FINGERPRINT\_IMAGEFAIL:

Serial.println("Imaging error");

break;

default:

Serial.println("Unknown error");

break;

}

delay(1000);

continue;

}

p = finger.image2Tz();

if (p != FINGERPRINT\_OK) {

Serial.println("Failed to convert image");

return;

}

p = finger.fingerFastSearch();

if (p != FINGERPRINT\_OK) {

Serial.println("Finger not found in database.");

return;

}

// Found a match!

Serial.print("Found ID #"); Serial.print(finger.fingerID);

Serial.print(" with confidence of "); Serial.println(finger.confidence);

Serial.print("Deleting ID #"); Serial.println(finger.fingerID);

p = finger.deleteModel(finger.fingerID);

if (p == FINGERPRINT\_OK) {

Serial.println("Deleted!");

} else if (p == FINGERPRINT\_PACKETRECIEVEERR) {

Serial.println("Communication error");

} else if (p == FINGERPRINT\_BADLOCATION) {

Serial.println("Could not delete in that location");

} else if (p == FINGERPRINT\_FLASHERR) {

Serial.println("Error writing to flash");

} else {

Serial.print("Unknown error: 0x"); Serial.println(p, HEX);

}

return;

}

}

uint8\_t readnumber(void) {

uint8\_t num = 0;

while (num == 0) {

while (!Serial.available());

num = Serial.parseInt();

}

return num;

}

**7. TESTING AND RESULTS**

**TEST SCENARIOS**

- Scenario 1: Enroll a new fingerprint and verify it successfully.

- Scenario 2: Attempt to verify a non-enrolled fingerprint.

- Scenario 3: Delete an enrolled fingerprint and verify the deletion by trying to authenticate the same finger.

**OBSERVATIONS**

The fingerprint sensor was able to enroll and verify fingerprints accurately in most cases. The servo motor responded promptly to successful verifications. The system effectively prevented unauthorized access by denying non-enrolled fingerprints.

**PERFORMANCE ANALYSIS**

- **SPEED**: The system responded within 1-2 seconds for enrollment and verification.

- **ACCURACY**: The fingerprint sensor demonstrated high accuracy with minimal false rejections.

- **RELIABILITY**: The system was reliable in various lighting conditions and for different users.

**8. CONCLUSION**

**SUMMARY**

The fingerprint sensor management system successfully demonstrated the integration of biometric authentication with mechanical action. The system effectively enrolled, verified, and deleted fingerprints, with the servo motor acting as a visual indicator of successful verification.

**FUTURE ENHANCEMENTS**

- Multi-fingerprint Enrollment: Enhancing the system to enroll multiple fingerprints for the same ID.

- Remote Management: Implementing remote fingerprint management via wireless communication.

- Additional Security Features: Adding features like PIN verification or two-factor authentication.

**9. REFERENCES**

- Adafruit Fingerprint Sensor Datasheet

- Arduino Servo Library Documentation

- Arduino IDE Official Site