Security Access System

Introducere

Prezentare proiect:

- Proiectul ales este un sistem minimalist, dar bun pentru securizarea unei incuietori.
- Am cautat ceva aplicat care sa aiba de-a face cu securitatea, si asa, ideea cu Security Access System.
- Sistemul prezinta un senzor de scanare a amprentei, unul de miscare, un LED al carui comportament este dictat de senzorul de miscare, un LCD 16×2.
- Sunt prezente si 4 butoane interactive pentru comenzi destiante controlului amprentei si un servomotor ce pune in actiune sistemul de inchidere(zavorul)

Descriere generală

Prin intermediul display-ul informatiile vor fi afisate astfel incat utilizatorul sa stie pasul curent de inregistrare sau stergere amprenta. Prin butoane acesta poate seta, sterge amprenta sau naviga pentru ID-ul amprentei. Dupa ce amprenta a fost setata se poate trece la pasul urmator.

Se testeaza amprenta prin amplasarea degetului pe senzor, iar daca se confirma autentificarea amprentei, servomotorul actioneaza sistemul de inchidere/deschidere.

Hardware Design

Aici se afla tot ce ţine de hardware design:

- Arduino Uno
- Servo motor MG90S
- SENZOR ULTRASUNETE HC-SR04
- Senzor Scanare Amprenta
- 4 Butoane

- LED si BredBoard
- Ecran LCD 16×2

Software Design

Descrierea codului aplicaţiei (firmware):

- Aplicatia a fost dezvoltata in ArduinoIDE.
- Un rezumat, lcd-ul afiseaza meniul principal cu functiile: scan(stocare amprenta noua), delete(stergere) si match(verificare amprenta)
- Daca amprenta a fost validata, se actioneaza bariera(poate fi un alt sistem de securitate, pe acesta l-am ales eu) de catre servomotor si se permite deschiderea/trecerea. Senzorul ultrasunet detecteaza prezenta unei persoane/obiect in fata senzorului de amprenta si aprinde un LED(pulsatie prin PWM).

```
#include <Adafruit_Fingerprint.h>
#include <LiquidCrystal.h>
#include <SoftwareSerial.h>
#include <Servo.h>
#include <NewPing.h>


LiquidCrystal lcd(8, 9, 10, 11, 12, 13);

// Initialize SoftwareSerial for fingerprint  RX (3) and TX (2)
SoftwareSerial mySerial(2, 3);

Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

Servo myServo;

const int button1 = A0;
const int button2 = A1;
const int button3 = A2;
const int button4 = A3;

#define TRIGGER_PIN 5
#define ECHO_PIN 4
#define MAX_DISTANCE 180// Maximum distance we want to measure (in cm)
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

const int ledPin = 6; // LED pin
const int distanceThreshold = 50; // Distance threshold in cm for the LED to
pulse
```

```
int fingerprintID = 1; // Start fingerprint ID 1
int menuOption = 0; // Current menu option

unsigned long previousMillis = 0;
int brightness = 0;
int fadeAmount = 5;

//Timer part:
unsigned long matchStartTime = 0;
bool timerRunning = false;
const unsigned long timeout = 10000; // 10 seconds



void setup() {
  Serial.begin(9600);
  while (!Serial);

  // Initialize the fingerprint sensor
  finger.begin(57600);
  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1);
  }

  // Initialize the LCD
  lcd.begin(16, 2);
  showMenu();

  // Set up the button pins as inputs
  pinMode(button1, INPUT_PULLUP);
  pinMode(button2, INPUT_PULLUP);
  pinMode(button3, INPUT_PULLUP);
  pinMode(button4, INPUT_PULLUP);

  // Initialize the servo
  myServo.attach(7); // Attach the servo on pin 7
  myServo.write(0);

  pinMode(ledPin, OUTPUT);
}

void loop() {

  // Read buttons for menu navigation and selection
  if (digitalRead(button1) == LOW) {
    if (menuOption == 0) { // "Scan" selected
      scanFingerprint();
    } else if (menuOption == 1) { // "Del" selected
```

```
      deleteFingerprint();
    } else if (menuOption == 2) { // "Match" selected
      matchFingerprint();
    }
    delay(300); // Debounce delay
  }
  if (digitalRead(button2) == LOW) {
    menuOption = (menuOption + 1) % 3;
    showMenu();
    delay(300);
  }

  //TIMER !!!!!!!!!!!!

  if (timerRunning) {
    unsigned long currentMillis = millis();

    // Check if button 3 is pressed
    if (digitalRead(button3) == LOW) {
      unsigned long matchEndTime = currentMillis;
      timerRunning = false;

      // Calculate and display the elapsed time
      unsigned long elapsedTime = (matchEndTime - matchStartTime) / 1000; //
Convert to seconds
      myServo.write(0); // Return the servo to 0 degrees

      lcd.clear();
      lcd.print("Elapsed Time:");
      lcd.setCursor(0, 1);
      lcd.print(elapsedTime);
      lcd.print(" sec");
      delay(3000);
      showMenu();
    }

    // Check if the timeout has been reached
    if (currentMillis - matchStartTime >= timeout) {
      timerRunning = false;
      myServo.write(0);

      lcd.clear();
      lcd.print("Auto Closed");
      delay(3000);
      showMenu();
    }
  }
```

```
  unsigned int distance = sonar.ping_cm();
  if (distance > 0 && distance < distanceThreshold) {
    pulseLED();
  }




}

void showMenu() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Scan");

  lcd.setCursor(8, 0);
  lcd.print("Del");

  lcd.setCursor(0, 1);
  lcd.print("Match");

  if (menuOption == 0) {
    lcd.setCursor(4, 0);
    lcd.print(">");
  } else if (menuOption == 1) {
    lcd.setCursor(12, 0);
    lcd.print(">");
  } else if (menuOption == 2) {
    lcd.setCursor(5, 1);
    lcd.print(">");
  }
}

void scanFingerprint() {
  lcd.clear();
  lcd.print("Place finger");

  while (finger.getImage() != FINGERPRINT_OK);
  finger.image2Tz(1);
  lcd.clear();
  lcd.print("Remove finger");
  delay(2000);

  while (finger.getImage() == FINGERPRINT_NOFINGER);
  while (finger.getImage() != FINGERPRINT_OK);
  finger.image2Tz(2);

  if (finger.createModel() == FINGERPRINT_OK) {
    finger.storeModel(fingerprintID);
    lcd.clear();
    lcd.print("Print ID:");
```

```
    lcd.setCursor(0, 1);
    lcd.print(fingerprintID);
    fingerprintID++;
  } else {
    lcd.clear();
    lcd.print("Enroll failed");
  }
  delay(2000);
  showMenu();
}

void deleteFingerprint() {
  int idToDelete = 1;
  while (true) {
    lcd.clear();
    lcd.print("PrintID:");
    lcd.setCursor(8, 0);
    lcd.print(idToDelete);
    delay(300);

    if (digitalRead(button4) == LOW) {
      // Wait for button release
      while (digitalRead(button4) == LOW) {
        delay(10);
      }
      idToDelete++;
    } else if (digitalRead(button1) == LOW) {
      // Wait for button release
      while (digitalRead(button1) == LOW) {
        delay(10);
      }
      if (finger.deleteModel(idToDelete) == FINGERPRINT_OK) {
        lcd.clear();
        lcd.print("Deleted ID:");
        lcd.setCursor(0, 1);
        lcd.print(idToDelete);
      } else {
        lcd.clear();
        lcd.print("Del failed");
      }
      delay(2000);
      break;
    }

    // Check for menu exit button press (optional)
    if (digitalRead(button3) == LOW) {

      while (digitalRead(button3) == LOW) {
        delay(10); // Small delay to debounce
      }
      showMenu();
```

```
      return;
    }
  }

  showMenu();
}

void matchFingerprint() {
  lcd.clear();
  lcd.print("Place finger");

  while (true) {
    if (finger.getImage() == FINGERPRINT_OK) {
      if (finger.image2Tz() == FINGERPRINT_OK) {
        if (finger.fingerFastSearch() == FINGERPRINT_OK) {
          lcd.clear();
          lcd.print("Print OK");
          myServo.write(90); // Move the servo to 90 degrees
          matchStartTime = millis(); // Start the timer
          timerRunning = true; // Set the timer flag
          return;;
        } else {
          lcd.clear();
          lcd.print("Not found");
        }
      }
      delay(2000);
      break;
    }
  }
  showMenu();
}

//PWM pulse LED

void pulseLED() {
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= 30) {
    previousMillis = currentMillis;

    // Change the brightness for next time through the loop:
    brightness = brightness + fadeAmount;

    // Reverse the direction of the fading at the ends of the fade:
    if (brightness <= 0 || brightness >= 255) {
      fadeAmount = -fadeAmount;
    }


    analogWrite(ledPin, brightness);
```

```
  }
}
```

Rezultate Obţinute

Care au fost rezultatele obţinute în urma realizării proiectului vostru.

Concluzii

Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună 😉.

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

Export to PDF