Lacăt cu aprentă și accesare wireless

Introducere

Proiectul meu constă în dezvoltarea unui lacăt cu amprentă și acces wireless, folosind un microcontroller Arduino Uno și alte componente electronice. Scopul acestui proiect este de a oferi o soluție inovatoare și practică pentru securizarea obiectelor valoroase, combinând tehnologia avansată cu necesitatea de securitate și oferind control total utilizatorului asupra accesului la bunurile sale. Ideea a pornit din dorința mea de a crea un sistem de securitate eficient și ușor de utilizat, oferind o alternativă modernă la lacătele tradiționale. Consider că acest proiect este util pentru mine și pentru alții deoarece oferă o soluție sigură și convenabilă pentru protejarea bunurilor valoroase. Pentru mine, acest proiect reprezintă o oportunitate de a aplica cunoștințele mele în domeniul ingineriei și de a crea ceva cu adevărat util și inovator.

Descriere generală

Sistemul folosește un Arduino Uno Rev4 ca unitate centrală de control. Un senzor optic de amprentă este conectat la Arduino prin UART (Universal Asynchronous Receiver-Transmitter) pentru a colecta și procesa amprentele digitale.

Arduino comunică cu un display LCD 1602 prin I2C (Inter-Integrated Circuit) pentru a afișa informații despre statusul sistemului și instrucțiuni pentru utilizator. De asemenea, există trei butoane conectate la pinii GPIO (General Purpose Input/Output) ai Arduino-ului: un buton pentru atribuirea amprentei digitale (Button Assign FP), un buton pentru ștergerea amprentei (Button Delete FP) și un buton pentru asocierea cu telefonul mobil (Button Pair Phone).

Comunicarea între sistem și telefonul mobil se realizează prin WiFi, permitând utilizatorului să gestioneze accesul și configurările sistemului direct de pe dispozitivul său mobil. Aceasta include posibilitatea de a adăuga sau elimina amprente, precum și de a verifica statusul lăcatului prin intermediul unei aplicații mobile.

Hardware Design

Arduino UNO rev4	https://docs.arduino.cc/hardware/uno-r4-wifi/
Senzor optic de amprenta	https://learn.adafruit.com/adafruit-optical-fingerprint-sensor/overview
LCD 1602	n-am gasit link bun
Servo Motor	https://www.amazon.in/Robodo-Electronics-Tower-Micro-Servo/dp/B00MTFFAE0

Software Design

Mediu de dezvoltare: **Arduino IDE**

Librarii folosite: **Adafruit fingerprint**, **LCD I2C**, **Arduino IoT Cloud**

Algoritmi si structuri folosite:

```
1) Generator cheie privata pentru conectare telefon
2) Busy waiting pt asteptarea unei aprente pt assign/remove
3) Intreruperi la schimbarea variabelelor unLock si secCod (variable
```

comune cu aplicatia)

```
#include <Adafruit_Fingerprint.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Servo.h>

#include "thingProperties.h"
#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>


#if (defined(__AVR__) || defined(ESP8266)) && !defined(__AVR_ATmega2560__)
// For UNO and others without hardware serial, we must use software
serial...
// pin #2 is IN from sensor (GREEN wire)
// pin #3 is OUT from arduino  (WHITE wire)
// Set up the serial port to use softwareserial..
SoftwareSerial mySerial(2, 3);

#else
// On Leonardo/M0/etc, others with hardware serial, use hardware serial!
// #0 is green wire, #1 is white
#define mySerial Serial1

#endif


Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
LiquidCrystal_I2C lcd(0x27, 16, 2);

uint8_t id;

int state = 0;

const int assignBtn = 13;
const int removeBtn = 12;
const int pairBtn = 11;

int btn1 = 0;
int btn2 = 0;
int btn3 = 0;

Servo myServo;

int fgrp = -1;
int toRm = -1;

const char characters[] =
"ABCDEFGHJKLMNOPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz0123456789";
```

```
const int codeLength = 7;

const int numberOfCodes = 5; // Number of codes to store

char storedCodes[numberOfCodes][codeLength + 1]; // Array to store the codes
int currentCodeIndex = 0; // Index to keep track of the current code

char randomCode[codeLength + 1];

bool paired = false;

void setup()
{

  // Connect to Arduino Cloud
  initProperties();
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  while (!ArduinoCloud.connected()) {
    ArduinoCloud.update();
    delay(500);
  }

  lcd.init();         // initialize the lcd
  lcd.backlight();

  pinMode(assignBtn, INPUT);
  pinMode(removeBtn, INPUT);
  pinMode(pairBtn, INPUT);

  myServo.attach(9);

  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);

  id = 1;

  randomSeed(analogRead(0));
}

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
{

  ArduinoCloud.update();

  btn1 = digitalRead(assignBtn);
  btn2 = digitalRead(removeBtn);
  btn3 = digitalRead(pairBtn);

  switch(state) {
    case 0:
      unLock = true;
      myServo.write(0);
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Open");
      delay(10);

      if(btn1 == HIGH) {
        state = 1;
      } else if(btn2 == HIGH) {
        state = 2;
      } else if(btn3 == HIGH) {
        state = 3;
      }

      fgrp = getFingerprintIDez();
```

```
      delay(50);
      if(fgrp != -1) {
        state = 4;
      }

      break;

    case 1:
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Assign Fingerprint");

      while (! getFingerprintEnroll() );
      state = 0;
      break;

    case 2:
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Place Finger");
      lcd.setCursor(0, 1);
      lcd.print("to remove");
      toRm = -1;
      while(toRm == -1) {
        toRm = getFingerprintIDez();
        delay(50);
      }
      if(toRm != -1){
        deleteFingerprint(toRm);
        delay(50);
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Finger Removed");
        delay(500);
      }else{
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Finger not Removed");
        delay(500);
      }
      state = 0;
      break;
    case 3:
      generateRandomCode(randomCode, codeLength);
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print(randomCode);
      delay(10000);
      state = 0;
      break;
    case 4:
```

```
      unLock = false;
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Locked");
      delay(10);

      myServo.write(90);

      fgrp = getFingerprintIDez();
      delay(50);
      if(fgrp != -1) {
        state = 0;
      }
      break;
  }


  // 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 deleteFingerprint(uint8_t id) {
  uint8_t p = -1;

  p = finger.deleteModel(id);

  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 p;
}
```

```
// 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;
}

uint8_t getFingerprintEnroll() {

  int p = -1;
  Serial.print("Waiting for valid finger to enroll as #");
Serial.println(id);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Place Finger");
  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");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Place Same");
  lcd.setCursor(0, 1);
  lcd.print("Finger");
  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;
  }

  id++;
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Fingerprint OK");
  delay(2000);
  return true;
}

void generateRandomCode(char *code, int length) {
  for (int i = 0; i < length; i++) {
    int randomIndex = random(0, sizeof(characters) - 1);
    code[i] = characters[randomIndex];
  }
  code[length] = '\0'; // Null terminator for the string
}

/*
  Since UnLock is READ_WRITE variable, onUnLockChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onUnLockChange()  {
  for (int i = 0; i < currentCodeIndex; i++){
    if(storedCodes[i] == secCode){
      if(unLock)
        state = 0;
      else
        state = 4;
    }
  }
}
/*
  Since SecCode is READ_WRITE variable, onSecCodeChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onSecCodeChange()  {
  String tmp = String(randomCode);
  Serial.println(tmp);
  Serial.println(secCode);
```

```
  if (tmp == secCode) {
    secCode.toCharArray(storedCodes[currentCodeIndex], codeLength + 1);
    currentCodeIndex = (currentCodeIndex + 1) % numberOfCodes;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Phone paired");
    delay(3000);
    paired = true;
  }
}
```

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