Sistem alarma - Cosor Mihai 332CAb

Introducere

Ideea mi-a venit gandindu-ma la usile pe care scrie “Acces interzis” dar de fapt nu sunt securizate in vreun fel. Proiectul consta intr-un senzor ultrasonic care detecteaza trecerea unui om prin fata lui care implica un buzzer sa scoata un sunet. Sunetul se poate opri daca persoana in cauza are o cartela autorizata si o foloseste in apropierea unui modul rfid sau daca introduce o parola corecta la tastatura. De asemenea, cartela / codul se poate folosi si inainte de trecerea prin fata senzorului pentru a-l deactiva o perioada scurta de timp. Tastatura suporta multe actiuni precum: adaugare / stergere parole noi (doar daca introduci codul de admin), stergere caractere in timpul introducerii parolei, iesire din functia selectata. Aceasta vine insotita cu un LCD care ofera mesaje sugestive la fiecare pas. Proiectul poate fi util pentru scenariul descris mai sus, dar si ca element de securitate al propriei case.

Descriere generală

Ghid de utilizare tastatura:

- A - adaugare parola (necesita parola admin)
- D - stergere parola (necesita parola admin)
- C - trimitere parola (autentificare)
- B - stergere caracter
- * - confirmare parola admin
- # - iesire din functia curenta

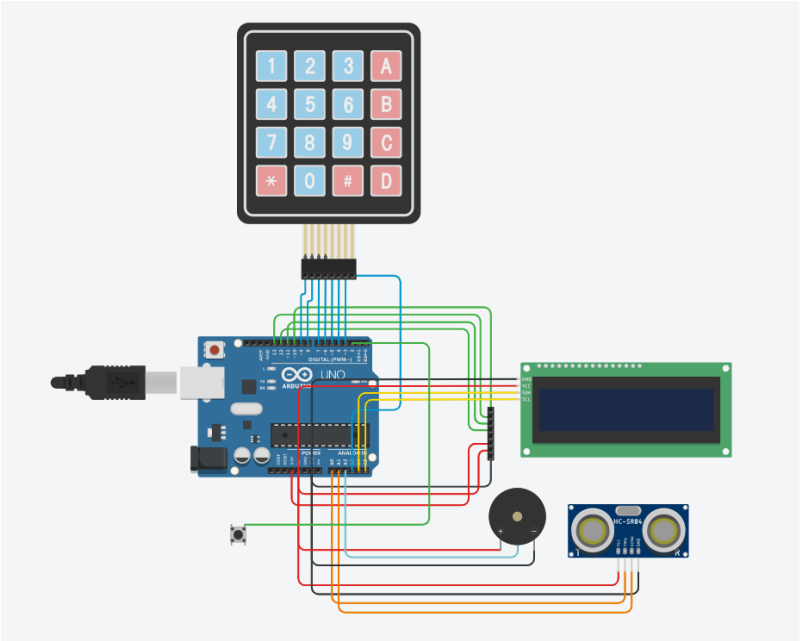
Sistemul poate stoca maxim 10 parole, fiecare de 4 sau 5 cifre.

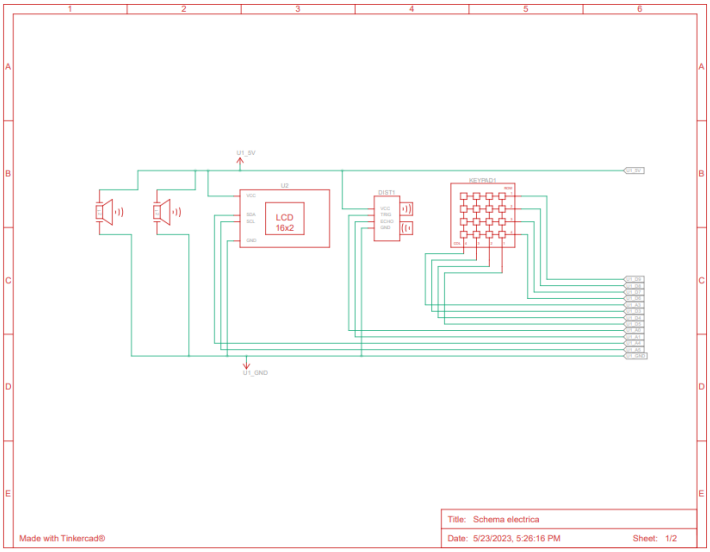
Hardware Design

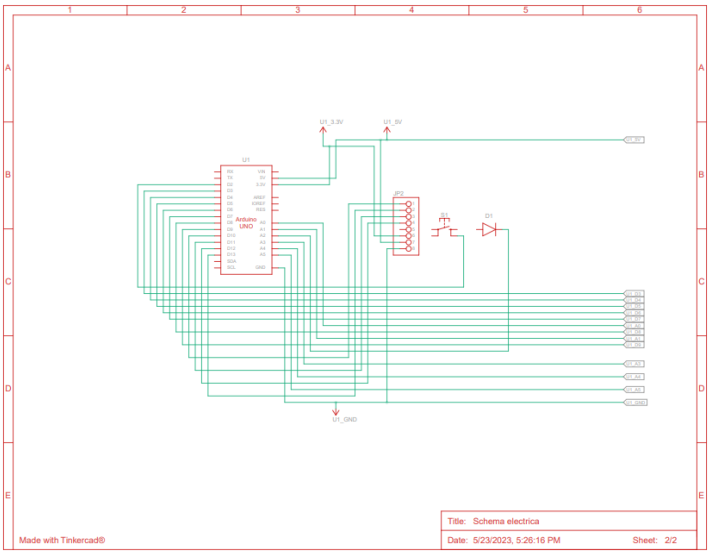
Lista de piese:

- placuta Arduino Uno
- modul RFID

- translator de nivel (necesar pentru modulul RFID)
- senzor ultrasonic
- 2x placute de test
- tastatura matriceala 4×4
- LCD cu modul I2C
- buzzer
- fire







Software Design

Am folosit atat Arduino IDE pentru a incarca codul pe placuta, cat si Visual Studio Code pentru a scrie codul mai usor.

Pentru a determina ce librarii sa folosesc, am citit descrierea pieselor pe OpimusDigital. Astfel am avut nevoie de <Keypad.h> pentru a detecta inputul de la tastatura, <SPI.h> si <MFRC522.h> pentru a comunica prin SPI cu modulul RFID si LiquidCrystal_I2C.h> cu <Wire.h> pentru a comunica prin I2C cu ecranul LCD.

De asemenea, am folosit https://github.com/robsoncouto/arduino-songs pentru melodia de alarma.

Principiul este simplu, senzorul ultrasonic este verificat constant pentru a detecta orice miscare, iar in momentul in care aceasta este detectata, se reda melodia de alarma pe buzzer. Odata introdus un cod corect la tastatura, sau o cartela acceptata apropiata de modulul RFID, alarma va inceta. Parolele pot fi adaugate / sterse folosind tastatura si introducand un cod de admin. Toate aceste actiuni sunt insotite de mesaje sugestive pe ecranul LCD. Nu in ultimul rand exista si un buton de panica, care odata apasat activeaza o intrerupere ce porneste semnalul de alarma. De data aceasta, melodia se poate opri doar folosind codul de admin.

Biblioteci utilizate

```
#include <Keypad.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include <MFRC522.h>
```

Setup module / senzori / butoane:

```
void setup() {
  // LCD setup
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(1, 0);
  lcd.print("Welcome home!");

  // RFID setup
  SPI.begin();
  mfrc522.PCD_Init();

  // Ultrasonic setup
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // Buton de panica setup
  pinMode(INTERRUPT_PIN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), panic, CHANGE);

  // Parole reset
  initPasswords();
}
```

Functia de loop verifica alarma si accepta input de la tastatura pentru unul din 3 scenarii:

```
void loop() {
  checkAlarm();
  char key = keypad.getKey();

  if (key){
    switch (key) {
      case ADD_PASSWORD:
        add_password();
        break;
      case DELETE_PASSWORD:
        delete_password();
        break;
      case SEND_PASSWORD:
        send_password();
        break;
      default:
        break;
    }
  }
}
```

Functia de verificare per total a alarmei:

```
    bool checkAlarm() {
      if (isPanic) { // verificare buton de panica
        lcd.clear();
        lcd.setCursor(3, 0);
        lcd.print("EMERGENCY!");
        startAlarm();
        return true;
      }

      if (isAuthorized) { // verificare daca s-a introdus cartela/cod
inainte
        countdown(); // countdown pana se reporneste senzorul
        isAuthorized = false;
        return false;
      }

      if (getID()) { // verificare existenta cartela
        if (tagID == MasterTag) {
          countdown();
          return false;
        } else {
          lcd.clear();
          lcd.print("Not a valid card");
        }
      }

      if (isObstacleInFront()) { // verificare obstacol in fata senzorului
        lcd.clear();
        lcd.print("INTRUDER ALERT!");
        startAlarm();
        return true;
      }

      return false;
    }
```

Functiile de citire senzor ultrasonic si calculare distanta si decidere daca exista obstacol in fata la 30 de cm sau nu.

```
    int getDistance() {
      digitalWrite(trigPin, LOW);
      delayMicroseconds(2);

      digitalWrite(trigPin, HIGH);
      delayMicroseconds(10);
      digitalWrite(trigPin, LOW);

      duration = pulseIn(echoPin, HIGH);
      distance = duration * 0.034 / 2; // formula distanta conform vitezei
sunetului
```

```
    return distance;
  }

  /*
  * Verificare senzor ultrasonic
  */
  bool isObstacleInFront() {
    int distance = getDistance();
    if (distance < 30) {
      return true;
    }

    return false;
  }
```

Functia de verificare a existentei cartelei in apropierea modulului RFID si de citire a acesteia.

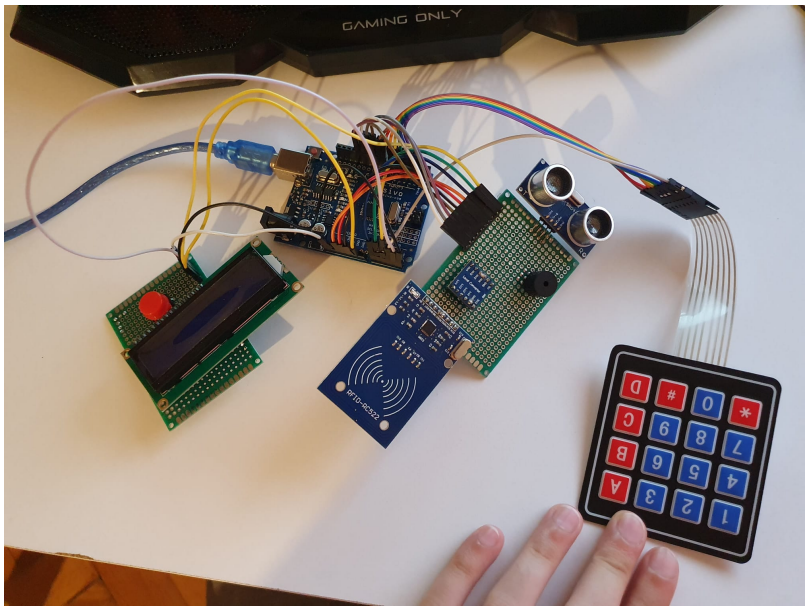
```
  bool getID()  {
    // Getting ready for Reading PICCs
    if (!mfrc522.PICC_IsNewCardPresent()) { //If a new PICC placed to RFID reader continue
      return false;
    }
    if (!mfrc522.PICC_ReadCardSerial()) { //Since a PICC placed get Serial and continue
      return false;
    }

    tagID = "";
    for (uint8_t i = 0; i < 4; i++) { // The MIFARE PICCs that we use have 4 byte UID
      //readCard[i] = mfrc522.uid.uidByte[i];
      tagID.concat(String(mfrc522.uid.uidByte[i], HEX)); // Adds the 4 bytes in a single String variable
    }

    tagID.toUpperCase();
    mfrc522.PICC_HaltA(); // Stop reading

    return true;
  }
```

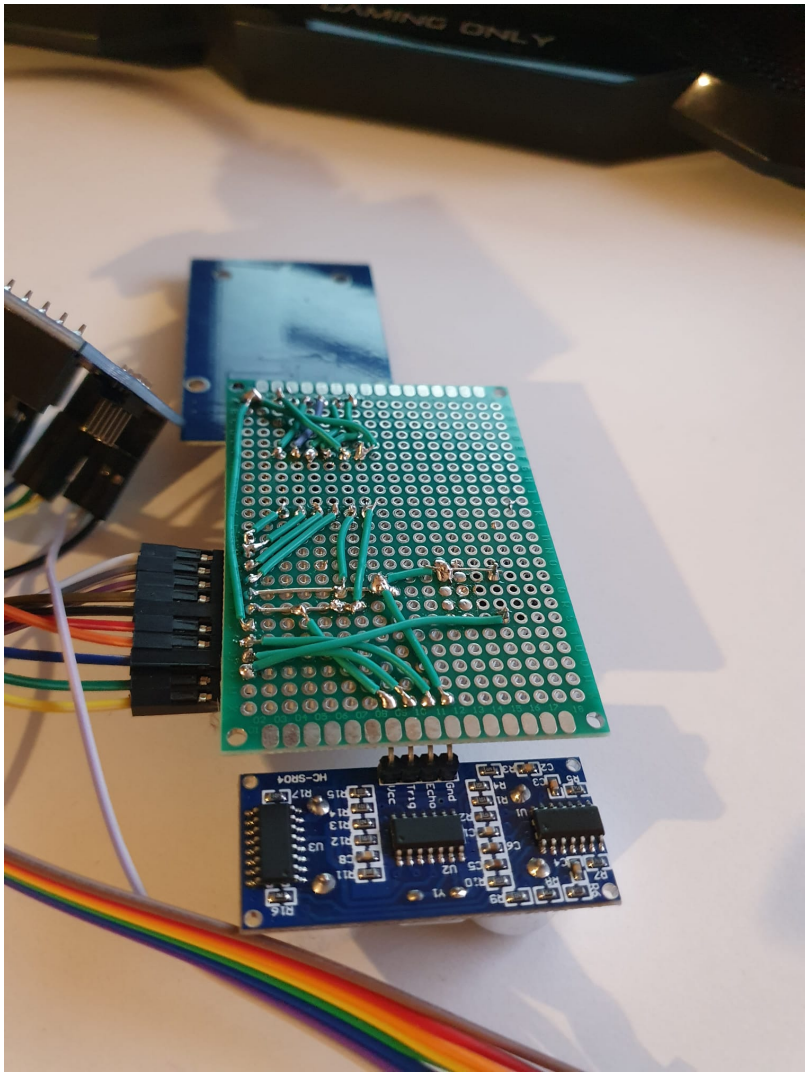
Restul functiilor folosite, doar semnaturile deoarece cele de add / send / delete sunt asemanatoare in primirea si handle-uirea inputului de la tastatura.

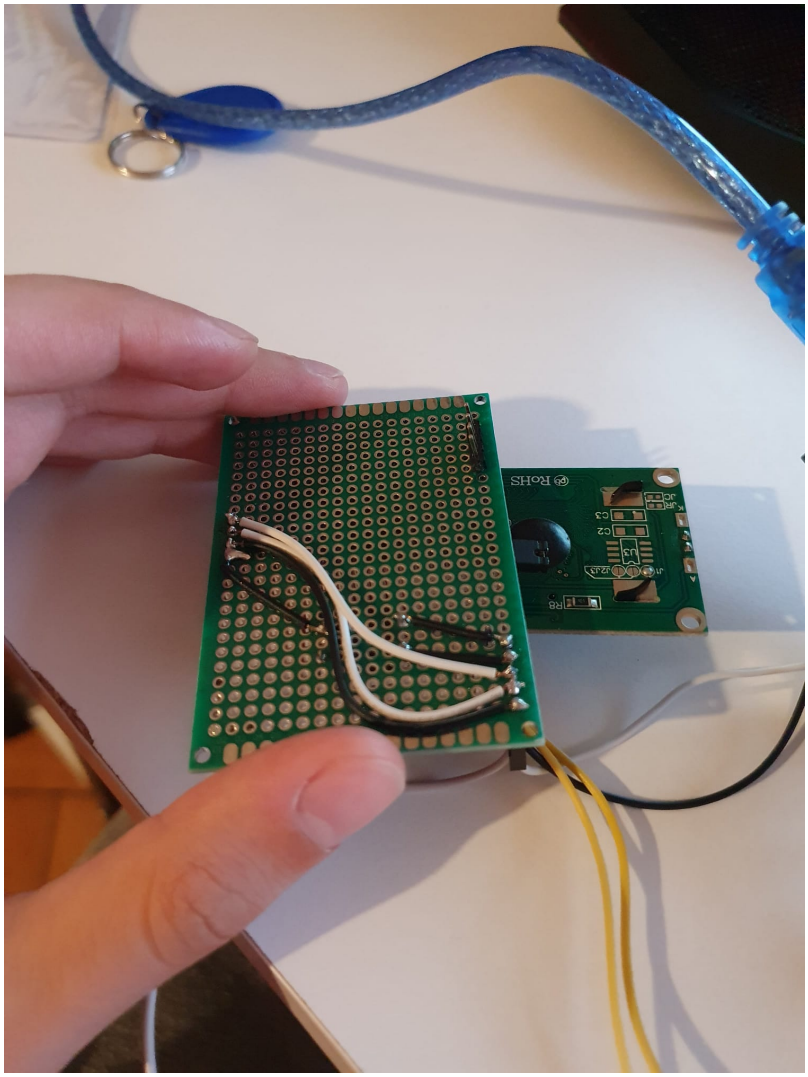
```
/*
 * Constante si initializari necesare programului in sine
 */
```

```
enum state {ACCEPTED, DENIED, WAITING};
const byte NUMBER_OF_PASSWORDS = 10;
byte FREE_SPACE_PASSWORDS = 10;
long passwords[NUMBER_OF_PASSWORDS];
const long MASTER_KEY = 2001;
bool isAuthorized = false;

/*
 * Initializare vector de parole
 */
void initPasswords() {...}

/*
 * Verificare parola
 */
state checkPassword(long password) {...}

/*
 * Verificare parola admin
 */
bool checkMasterKey() {...}

/*
 * Adaugare parola noua
 */
void add_password() {...}

/*
 * Stergere parola
 */
void delete_password() {...}

/*
 * Introducere parola pentru verificare
 */
void send_password() {...}

/*
 * Start alarma
 */
void startAlarm() {...}
```

Rezultate Obţinute


GAMING ONLY
D # 0 *
C 9 8 7
B 6 5 4
A 3 2 1


GAMING ONLY
HC-SR04
Vcc
Trig
Echo
Gnd



Concluzii

Proiectul a fost foarte interesant deoarece ideea dezvoltarii mi-a venit pe parcurs, nefiind formata complet si suficient de complex de la inceput. Pe langa asta, partea hardware a fost mai provocatoare, fiind compusa din mai multe etape de lipit / dezlipit / rearanjat piese pe placute. Odata terminata asta, software-ul a fost scris de placere, ideile de functionalitati venind pe parcurs.

Intentionez sa incorporez device-ul intr-o carcasa / cutie pentru a il utiliza frecvent.

Download

Sursa, diagrame, scheme, poze

Bibliografie/Resurse

Melodie alarma: https://github.com/robsoncouto/arduino-songs/tree/master/furelise

Export to PDF

From:
http://ocw.cs.pub.ro/courses/ - **CS Open CourseWare**

Permanent link:
http://ocw.cs.pub.ro/courses/pm/prj2023/vstoica/sistem-alarma-cosor-mihai

Last update: **2023/05/29 18:49**