

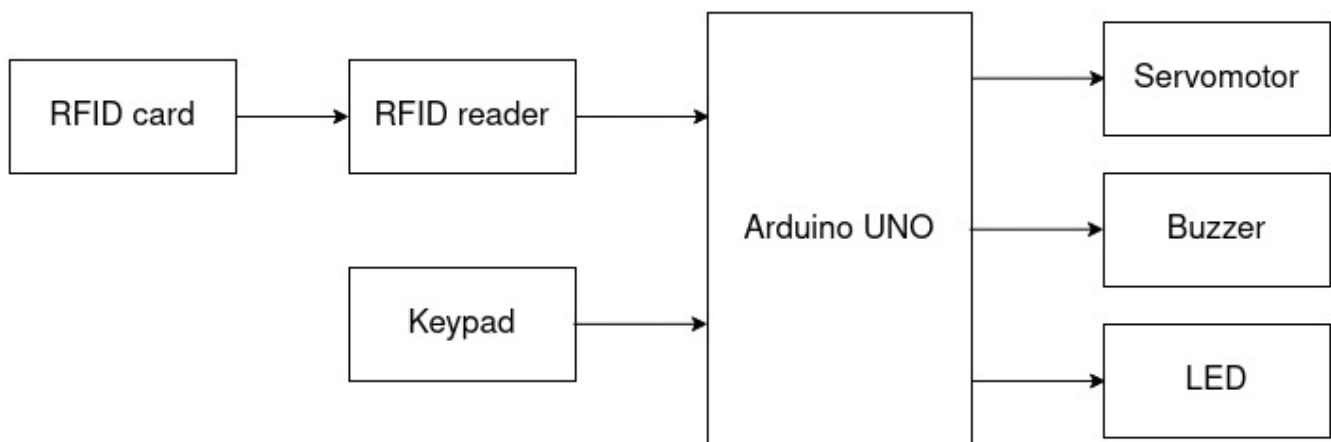
Safebox

Autor: [Mihai Ionescu](#)

Introducere

- Proiectul constă în crearea unui seif cu încuietoare. Pentru a-l deschide, trebuie scanat un card RFID.
- Ca măsură alternativă de securitate, dacă identificatorul cardului nu este recunoscut, în maxim 10 secunde trebuie introdusă o parolă pe keypad. Se va aprinde intermitent un LED.
- Dacă trece timpul, un buzzer va emite zgomot, alertând astfel o tentativă de acces neautorizat. În cazul autentificării corecte, încuietoarea seifului se deschide, prin acționarea unui servomotor.
- Dacă seiful este deschis, scanarea unui card RFID corect va determina închiderea acestuia.
- Scopul proiectului este posibilitatea de depozitare a unor bunuri în mod sigur.

Descriere generală

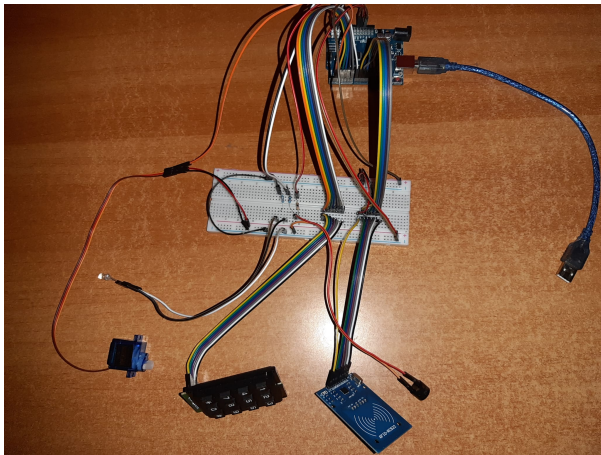


Hardware Design

Listă de piese:

- Arduino UNO
- breadboard
- keypad
- modul RFID (RC522)
- card, tag de proximitate RFID
- servomotor

- buzzer
- LED multicolor
- rezistențe (1 x 100Ω, 2 x 220Ω)
- jumpere
- headere



Schema electrică:



Software Design

Codul sursă:

```
// C++ code
//
#include <SPI.h>
#include <Servo.h>
#include <Keypad.h>
#include <MFRC522.h>
#define SERVO_PIN 2
#define LED_GREEN_PIN A3
#define LED_RED_PIN A4
#define BUZZER_PIN A5
#define SDA_PIN 10
#define RST_PIN 9
```

```
#define WAIT 1000
#define WARNING_BUZZ_INTERVAL 700
#define WARNING_TIMEOUT 10000

#define MAX_PASSWD_LENGTH 64

Servo servo;

const byte ROW_NUM = 4; //four rows
const byte COLUMN_NUM = 3; //3 columns

char keys[ROW_NUM][COLUMN_NUM] = {
  {'1', '2', '3'},
  {'4', '5', '6'},
  {'7', '8', '9'},
  {'*', '0', '#'}
};

//
//https://electropeak.com/learn/interfacing-4x3-membrane-matrix-keypad-with-arduino/
//byte pin_rows[ROW_NUM] = {5, 10, 9, 7}; //connect to the row pinouts of
//the keypad
//byte pin_column[COLUMN_NUM] = {6, 4, 8}; //connect to the column pinouts
//of the keypad
byte pin_rows[ROW_NUM] = {3, 8, 7, 5}; //connect to the row pinouts of the
keypad
byte pin_column[COLUMN_NUM] = {4, A2, 6}; //connect to the column pinouts of
the keypad

// Init keypad
Keypad keypad = Keypad( makeKeymap(keys), pin_rows, pin_column, ROW_NUM,
COLUMN_NUM );

// Init RC522
MFRC522 rc522(SDA_PIN, RST_PIN);

String enteredPasswd;

const String correctPasswd = "5378";
const byte correctCodeSize = 4;
const byte correctCode[4] = {99, 88, 50, 25};

volatile int safeState; // 0 = open, 1 = closed

void setup()
{
  Serial.begin(9600);
  SPI.begin();
  rc522.PCD_Init();
  Serial.println("Aproprie cardul..");
}
```

```
enteredPasswd = "";
pinMode(BUZZER_PIN, OUTPUT);
pinMode(LED_GREEN_PIN, OUTPUT);
pinMode(LED_RED_PIN, OUTPUT);
servo.attach(SERVO_PIN);
servo.write(0);
delay(500);
}

volatile int servoPos = 0; // var to store the servo pos
volatile bool wrongTag = false;
volatile bool proceed = false;
volatile bool state = 0; // 0 = closed, 1 = open
volatile unsigned long lastReadTime = 0;
volatile unsigned long previousMillisBuzz = 0;
volatile unsigned long warningStateStart = 0;

bool checkPasswd(String enteredPasswd) {
    int length = enteredPasswd.length();
    if (length != correctPasswd.length()) {
        return false;
    }

    for (int i = 0; i < length; i++) {
        if (correctPasswd[i] != enteredPasswd[i]) {
            return false;
        }
    }

    return true;
}

void actLED() {
    if (!state) {
        analogWrite(LED_GREEN_PIN, 0);
        analogWrite(LED_RED_PIN, 255);
    } else {
        analogWrite(LED_RED_PIN, 0);
        analogWrite(LED_GREEN_PIN, 255);
    }
}

void actBuzzer() {
    if (!state) {
        tone(BUZZER_PIN, 100, 300);
    } else {
        tone(BUZZER_PIN, 100, 100);
        delay(200);
        tone(BUZZER_PIN, 300, 100);
    }
}
```

```
void warningBuzz() {
    unsigned long currentMillis = millis();

    if (currentMillis - previousMillisBuzz >= WARNING_BUZZ_INTERVAL) {
        previousMillisBuzz = currentMillis;
        tone(BUZZER_PIN, 350, 200);
    }
}

void wrongPasswordBuzz() {
    tone(BUZZER_PIN, 450);
    delay(1000);
    noTone(BUZZER_PIN);
}

void angryBuzz() {
    Serial.println("INTRUDER");
    for (int i = 0; i < 80; i++) {
        if (i % 2 == 0) {
            analogWrite(LED_GREEN_PIN, 0);
            analogWrite(LED_RED_PIN, 255);
        } else {
            analogWrite(LED_GREEN_PIN, 255);
            analogWrite(LED_RED_PIN, 0);
        }
        tone(BUZZER_PIN, 50);
        delay(50);
    }

    noTone(BUZZER_PIN);
}

void lock_unlock() {
    servo.write(90 - servoPos);
    servoPos = 90 - servoPos;
    state = !state;

    actLED();
    actBuzzer();
}

void loop()
{
    // LED CODE
    actLED();
    // RFID READER CODE

    if (!wrongTag) {
        // Look for new cards
        if (rc522.PICC_IsNewCardPresent()) {
```

```
// Select one of the cards
unsigned long time = millis();
if (time - lastReadTime >= WAIT) {
    lastReadTime = time;
    if (rc522.PICC_ReadCardSerial()) {
        //Show UID on serial monitor
        byte letter;

        for (byte i = 0; i < correctCodeSize; i++) {
            if (rc522.uid.uidByte[i] != correctCode[i]) {
                Serial.println("ERROR! Invalid tag! Enter keys!");
                // TODO: intermittent buzz
                warningStateStart = millis();
                enteredPasswd = "";
                wrongTag = true;
            } else {
                Serial.println("OK");
                proceed = true;
            }
        }
    }
}
} else {
    warningBuzz();

    unsigned long currTime = millis();
    if (currTime - warningStateStart >= WARNING_TIMEOUT) {
        angryBuzz();
        warningStateStart = millis(); // reset
    }

    // Wrong tag => have to read keypad
    char key = keypad.getKey();
    if (key) {
        if (key == '#') {
            bool validPasswd = checkPasswd(enteredPasswd);
            if (validPasswd) {
                Serial.println("OK!");
                proceed = true;
                wrongTag = false;
            } else {
                wrongPasswordBuzz();
            }

            enteredPasswd="";
        }
        else {
            enteredPasswd += key;
        }
    }
}
```

```
    }  
  }  
  
  if (proceed) {  
    lock_unlock();  
  }  
  proceed = false;  
}
```

Rezultate Obținute

Proiectul funcționează conform descrierii. RFID Readerul și Keypadul acționează închiderea/deschiderea ușii.



Concluzii

Proiectul a fost util pentru deprinderea atât cu programarea în Arduino, cât și implementarea hardware.

Download

Arhiva cu codul sursă: [safebox_pm_mihai_ionescu.zip](#)

Jurnal

10.05.2022 - Creare pagină wiki (introducere + schemă bloc + listă piese)

31.05.2022 - Finalizare proiect

02.01.2022 - Completare documentație

Bibliografie/Resurse

<https://www.arduino.cc/reference/en/libraries/keypad/>

<https://www.arduino.cc/reference/en/libraries/mfrc522/>

<https://www.arduino.cc/reference/en/libraries/easy-mfrc522/>

[Export to PDF](#)

From:

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

Permanent link:

<http://ocw.cs.pub.ro/courses/pm/prj2022/arosca/safebox>



Last update: **2022/06/02 06:07**