

UNIwersytet Pedagogiczny im. KEN w KRAKOWIE
Instytut Informatyki



Projekt inżynierski: Dokumentacja techniczna

III rok Informatyka studia niestacjonarne

8 czerwca 2021

Spis treści

1	Wstęp	3
2	Zespół projektowy	3
3	Opis części praktycznej	4
3.1	Schemat podłączeń	4
3.2	Opis schematu podłączeń	5
4	Opis części programistycznej	7
4.1	Użyte biblioteki	7
4.2	Opis zmiennych	8
4.3	Opis funkcji	9
4.4	Implementacja i opis kodu	9
5	Załączniki	15
6	Bibliografia	15

1 Wstęp

Niniejszy projekt jest studenckim przedsięwzięciem, które na celu ma stworzenie zamka elektronicznego z wielopoziomową kontrolą dostępu. Nasz zamysł, jest połączeniem elektroniki, oraz programowania w programie Arduino. Zamek w swoim założeniu posiada trójstopniową kotrolę, w której skład wchodzi:

1. Pomiar temperatury użytkownika.
2. Odczyt breloka.
3. Wpisanie poprawnego kodu.

2 Zespół projektowy

Martyna Kłosek

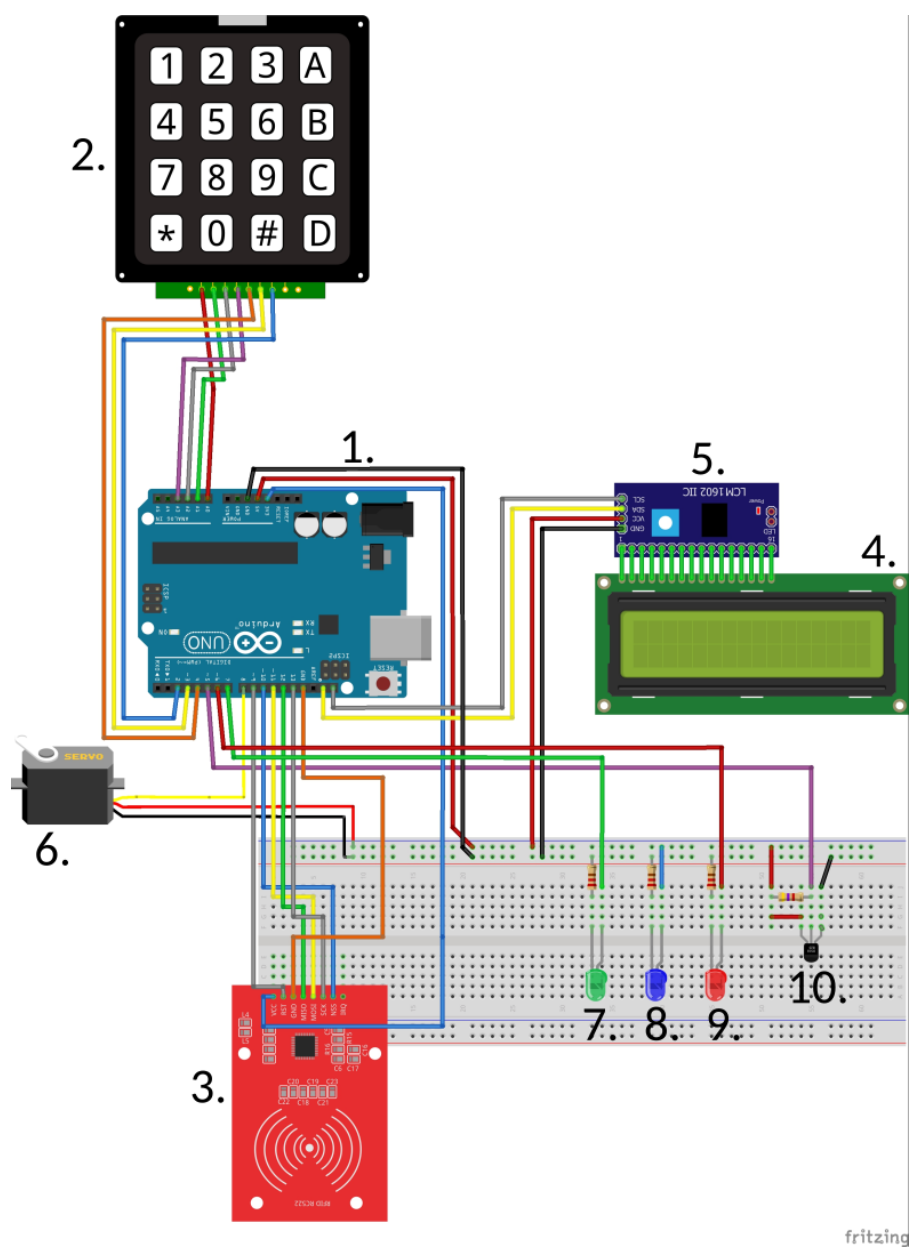
e-mail: martyna.klosek@student.up.krakow.pl

Jakub Mazurczyk

e-mail: jakub.mazurczyk@student.up.krakow.pl

3 Opis części praktycznej

3.1 Schemat połączeń



Rysunek 1: Schemat połączeń środowiska

3.2 Opis schematu połączeń

W zaprojektowanym przez nas schemacie podłączenia wszystkich potrzebnych w projekcie modułów, wykorzystaliśmy następujące części, które zostały odpowiednio oznaczone na schemacie (Rysunek 1):

1. Moduł Arduino Uno
2. Klawiatura
3. Moduł RFID MFRC522
4. Wyświetlacz LCD
5. Konwerter I2C dla wyświetlacza LCD
6. Serwomechanizm modelarski typu micro
7. Dioda LED zielona
8. Dioda LED niebieska
9. Dioda LED czerwona
10. Cyfrowy czujnik temperatury DS18B20 1-wire

Podane moduły zostały następnie połączone według zaprojektowanego schematu (Rysunek 1), w następujący sposób:

- Klawiatura została podłączona do kontrolera Arduino z wykorzystaniem pinów 1-7, które obsługują matrycę cyfr 0-9 oraz znaki specjalne "*" i "#". Ostatni, ósmy pin nie został podłączony, ponieważ symbole "A", "B", "C", "D" nie będą wykorzystywane w projekcie. Schemat podłączenia opisany został w tabeli (Tabela 1):

Klawiatura	Arduino
Pin1	A0
Pin2	A1
Pin3	A2
Pin4	A3
Pin5	4
Pin6	3
Pin7	2

Tabela 1: Schemat połączeń klawiatury

- Moduł RFID RC522 został podłączony do Arduino w sposób przedstawiony w tabeli (Tabela 2):

Moduł RFID	Arduino
RST	9
SDA	10
MOSI	11
MISO	12
SCK	13
GND	GND
3.3V	3.3V

Tabela 2: Schemat podłączeń modułu RFID RC522

- Wyświetlacz LCD został podłączony do Arduino w sposób przedstawiony w tabeli (Tabela 2):
- Konwerter I2C dla wyświetlacza LCD został podłączony do Arduino w sposób przedstawiony w tabeli (Tabela 3):

Konwerter I2C	Arduino
SCL	SCL
SDA	SDA
GND	GND
VCC	5V

Tabela 3: Schemat podłączeń konwertera I2C

- Serwomechanizm modelarski typu micro został podłączony do Arduino w sposób przedstawiony w tabeli (Tabela 4):

Serwomechanizm	Arduino
Żółty (sygnał)	D8
Ćzerwony (+)	5V
Czarny (GND)	GND

Tabela 4: Schemat podłączeń serwomechanizmu modelarskiego

- Diody LED zielona, niebieska, czerwona zostały podłączone do Arduino w sposób przedstawiony w tabeli (Tabela 5):

Diody LED	Arduino
Zielona	D7
Czerwona	D6
Niebieska	5V
Katody(przez rezystor)	GND

Tabela 5: Schemat podłączeń diody LED

- Cyfrowy czujnik temperatury DS18B20 został podłączony do Arduino w sposób przedstawiony w tabeli (Tabela 6) :

Czujnik	Arduino
VDD	5V
GND	GND
DQ	Pin 10

Tabela 6: Schemat podłączeń czujnika temperatury DS18B20

4 Opis części programistycznej

4.1 Użyte biblioteki

W celu łatwiejszego zaprogramowania zamka elektronicznego użyliśmy następujących bibliotek:

1. MFRC522.h (1) *biblioteka odpowiadająca za obsługę breloka RFID.*
2. LiquidCrystal_I2C.h (2) *biblioteka odpowiadająca za obsługę wyświetlacza LCD 2x16.*
3. Keypad.h (3) *biblioteka odpowiadająca za obsługę klawiatury - matryca 16 x tact switch.*
4. Servo.h (4) *biblioteka odpowiadająca za obsługę servomechanizmu.*
5. SPI.h (5) *biblioteka odpowiadająca za komunikację z urządzeniami SPI podpiętymi pod Arduino.*
6. OneWire.h (6) *biblioteka odpowiadająca za komunikację za pomocą protokołu 1-Wire.*
7. DS18B20.h (7) *biblioteka odpowiadająca za obsługę czujnika DS18B20.*

4.2 Opis zmiennych

W programie użyliśmy następujących zmiennych:

- Zmienna przechowująca adres czujnika temperatury:
`byte adres[8] = 0x28, 0x82, 0x30, 0x3B, 0xA, 0x0, 0x0, 0xA8;`
- Zmienna przechowująca numer PIN-u podłączenia diody zielonej:
`constexpr uint8_t diodaZielona = 7;`
- Zmienna przechowująca numer PIN-u podłączenia diody czerwonej:
`constexpr uint8_t diodaCzerwona = 6;`
- Zmienna przechowująca numer PIN-u podłączenia servomechanizmu:
`constexpr uint8_t servoPin = 8;`
- Zmienna przechowująca domyślny kod do zamka:
`char kod_domyslny[4] = '1', '2', '3', '3';`
- Zmienna przechowująca kod seryjny breloka RFID:
`String tagUID = "74 91 51 A3";`
- Zmienna przechowująca kod wprowadzony przez użytkownika:
`char kod_wprowadzony[4];`
- Zmienna przechowująca wartość, za pomocą której sprawdzamy poprawność breloka RFID:
`-boolean RFIDMode;`
- Zmienna przechowująca wartość, za pomocą której sprawdzamy temperaturę użytkownika: `boolean TempMode = true;`
- Zmienna przechowująca ilość wciśniętych klawiszy:
`char key_pressed = 0;`
- Zmienne przechowujące kolejno liczbę wierszy, oraz kolumn:
`const byte wiersze = 4;`
`const byte kolumny = 4;`
- Zmienna przechowująca przypisanie mapy klawiszy do wierszy, oraz kolumn:
`char mapa_klawiszy[wiersze][kolumny] =`
`'1', '2', '3', 'A',`
`'4', '5', '6', 'B',`
`'7', '8', '9', 'C',`
`'*', '0', '#', 'D'`
`;`
- Zmienne przechowujące numery pinów do których została przyporządkowana mapa klawiszy:
`byte piny_w[wiersze] = A0, A1, A2, A3;`
`byte piny_k[kolumny] = 4, 3, 2;`

4.3 Opis funkcji

W programie wyróżniamy dwie główne funkcje:

`void setup()` - funkcja ta, zawiera instrukcję, którą mikrokontroler wykona raz po uruchomieniu.

`void loop()` - funkcja ta, zawiera instrukcję, które będą powtarzane przez cały czas działania programu.

4.4 Implementacja i opis kodu

W celu łatwiejszego omówienia części programistycznej przeprowadziliśmy podział kodu na następujące części:

1. Biblioteki, oraz zaimplementowane zmienne.
2. Wnętrze funkcji `void setup()`.
3. Kod odpowiadający za pomiar temperatury.
4. Kod odpowiadający za odczytanie breloka RFID.
5. Kod odpowiadający za rozpoznanie poprawnie wpisanego kodu numerycznego.

Listing 1 przedstawia biblioteki opisane w dziale 4.1, zmienne opisane w dziale 4.2 oraz funkcje opisane w dziale 4.3. W celu przypomnienia sobie zasady ich działania, wróć do odpowiedniego rozdziału.

```
1 #include <LiquidCrystal_I2C.h>
2 #include <Keypad.h>
3 #include <MFRC522.h>
4 #include <Servo.h>
5 #include <SPI.h>
6 #include <OneWire.h>
7 #include <DS18B20.h>
8
9 #define ONEWIRE_PIN 5
10 byte adres[8] = {0x28, 0x82, 0x30, 0x3B, 0xA, 0x0, 0x0, 0xA8};
11
12 LiquidCrystal_I2C lcd(0x27, 16, 2);
13 MFRC522 mfrc522(10, 9);
14 Servo sg90;
15 OneWire onewire(ONEWIRE_PIN);
16 DS18B20 sensors(&onewire);
17
18 constexpr uint8_t diodaZielona = 7;
19 constexpr uint8_t diodaCzerwona = 6;
20 constexpr uint8_t servoPin = 8;
21
22 char kod_domyslne[4] = { '1', '2', '3', '3' };
23 String tagUID = "74 91 51 A3";
24 char kod_wprowadzony[4];
25 boolean RFIDMode;
26 boolean TempMode = true;
27 char key_pressed = 0;
28 uint8_t i = 0;
29
30 const byte wiersze = 4;
31 const byte kolumny = 4;
32
33 char mapa_klawiszy[wiersze][kolumny] = {
34   { '1', '2', '3', 'A' },
35   { '4', '5', '6', 'B' },
36   { '7', '8', '9', 'C' },
37   { '*', '0', '#', 'D' }
```

```

38 };
39
40 byte piny_w[wiersze] = {A0, A1, A2, A3};
41 byte piny_k[kolumny] = {4, 3, 2};

```

Listing 1: Kod przedstawiający implementację bibliotek oraz zmiennych.

Listing 2 przedstawia fragment opisujący wnętrze funkcji "void setup()":

```

45 void setup() {
46   pinMode(diodaCzerwona, OUTPUT);
47   pinMode(diodaZielona, OUTPUT);
48
49   sg90.attach(servoPin);
50   sg90.write(0);
51
52   lcd.begin();
53   lcd.backlight();
54   SPI.begin();
55   mfrc522.PCD_Init();
56   lcd.clear();
57
58   while(!Serial);
59   Serial.begin(9600);
60
61   sensors.begin();
62   sensors.request(adres);
63 }

```

Listing 2: Kod przedstawiający wnętrze funkcji void setup().

W pierwszej kolejności przeprowadzamy konfigurację PIN-ów Arduino (Listing 2, linia 46-47). Kolejno deklarujemy PIN-y dla servomechanizmu (Listing 2, linia 49), oraz ustawiamy pozycję servomechanizmu (Listing 2, linia 50). Po wykonaniu wcześniejszych kroków, inicjalizujemy wyświetlacz LCD, SPI, oraz MFRC522 (Listing 2, linia 52-56). W następnym kroku tworzymy pętlę warunkową while, która nie wykonuje żadnych czynności dopóki "serial monitor" nie zostanie otwarty (Listing 2, linia 58), gdy to nastąpi otwieramy port i ustawiamy prędkość na 9600 bodów (Listing 2, linia 59-62). Kolejno przechodzimy do następnego fragmentu kodu.

Listing 3 Fragment opisujący działanie czujnika temperatury:

```

67 void loop() {
68   if (TempMode == true)
69   {
70     if (sensors.available())
71     {
72       lcd.setCursor(0, 0);
73       lcd.print("Zmierz");
74       lcd.setCursor(0, 1);
75       lcd.print("temperature");
76       delay(4000);
77       lcd.clear();
78
79       for (float t=0 ; t<15 ; t++)
80       {
81         lcd.setCursor(0, 0);
82         lcd.print("Mierzenie ..");
83         temperature = sensors.readTemperature(adres);
84         Serial.print(temperature);
85         Serial.println(F(" °C"));
86         sensors.request(adres);
87         delay(1000);
88       }
89
90       if (temperature >= 25 && temperature <= 31)
91       {

```

```

92     lcd.clear();
93     lcd.setCursor(0, 0);
94     lcd.print("Blokada 1/3");
95     lcd.setCursor(0, 1);
96     lcd.print("Temperatura OK");
97     digitalWrite(diodaZielona, HIGH);
98     delay(5000);
99     digitalWrite(diodaZielona, LOW);
100    lcd.clear();
101    temperature = 0;
102    RFIDMode = true;
103    TempMode = false;
104  }
105  else if (temperature < 25 || temperature > 31)
106  {
107    lcd.clear();
108    lcd.setCursor(0, 0);
109    lcd.print("Blokada 1/3");
110    lcd.setCursor(0, 1);
111    lcd.print("Brak dostepu");
112    digitalWrite(diodaCzerwona, HIGH);
113    delay(3000);
114    digitalWrite(diodaCzerwona, LOW);
115    lcd.clear();
116    TempMode = true;
117  }
118  }
119 }

```

Listing 3: Kod przedstawiający implementację i działanie czujnika temperatury.

W pierwszej kolejności sprawdzamy stan zmiennej TempMode (Listing 3, linia 68) , jeśli zmienna ma wartość true, sprawdzamy czy podłączony czujnik temperatury jest dostępny (Listing 3, linia 70), jeśli tak wypisujemy na ekranie komunikat o przystąpieniu do mierzenia temperatury (Listing 3, linia 71-77).

Kolejnym krokiem jest wejście do pętli, która przez piętnaście sekund odczytuje temperaturę z opuszka palca użytkownika (Listing 3, linia 79-88).

Po odczytaniu dochodzimy do pętli warunkowych, które po przeanalizowaniu wyników użytkownika wyświetlają napis "Temperatura OK" w przypadku temperatury w granicach od 25 do 31 stopni Celsjusza (Listing 3, linia 90-104), bądź napis "Brak dostępu" w przypadku temperatury wykraczającej poza zakres (Listing 3, linia 105-117).

Fragment opisujący działanie breloka RFID:

```

122 {
123     lcd.setCursor(0, 0);
124     lcd.print("Zeskanuj brelok");
125
126     if ( ! mfr522.PICC_IsNewCardPresent() ) {
127         return;
128     }
129
130     if ( ! mfr522.PICC_ReadCardSerial() ) {
131         return;
132     }
133
134     String tag = "";
135     for (byte j = 0; j < mfr522.uid.size; j++)
136     {
137         tag.concat(String(mfr522.uid.uidByte[j] < 0x10 ? " 0" : " "));
138         tag.concat(String(mfr522.uid.uidByte[j], HEX));
139     }
140     tag.toUpperCase();
141 }

```

```

142     if (tag.substring(1) == tagUID)
143     {
144         lcd.clear();
145         lcd.setCursor(0, 0);
146         lcd.print("Blokada 2/3");
147         lcd.setCursor(0, 1);
148         lcd.print("Brelok zgodny");
149         digitalWrite(diodaZielona, HIGH);
150         delay(3000);
151         digitalWrite(diodaZielona, LOW);
152
153         lcd.clear();
154         lcd.setCursor(0, 0);
155         lcd.print("Podaj kod:");
156         lcd.setCursor(6, 1);
157         RFIDMode = false;
158     }
159
160     else
161     {
162         lcd.clear();
163         lcd.setCursor(0, 0);
164         lcd.print("Blokada 2/3");
165         lcd.setCursor(0, 1);
166         lcd.print("Brelok niezgodny");
167         //digitalWrite(buzzerPin, HIGH);
168         digitalWrite(diodaCzerwona, HIGH);
169         delay(3000);
170         digitalWrite(diodaCzerwona, LOW);
171         lcd.clear();
172         TempMode = true;
173         RFIDMode = false;
174     }
175 }

```

Listing 4: Kod przedstawiający implementację i działanie breloka RFID.

W pierwszej kolejności wyświetlamy komunikat "Zeskanuj brelok" (Listing 4, linia 123-124). Następnie przechodzimy przez dwie pętle if (Listing 4, linia 126-132), które kolejno sprawdzają dostępne karty i wybierają jedną z nich.

Program wchodzi do pętli warunkowej if, która w przypadku zeskanowania poprawnego breloka wyświetli komunikat "Brelok zgodny" (Listing 4, linia 142-158), oraz dioda zielona zaświeci się.

W przypadku zeskanowania błędnego breloka wyświetli komunikat "Brelok niezgodny" (Listing 4, linia 160-175), oraz następuje zapalenie diody czerwonej.

Fragment opisujący działanie kodu numerycznego:

```

178 if (RFIDMode == false) {
179     key_pressed = keypad_key.getKey();
180     if (key_pressed){
181         kod_wprowadzony[i++] = key_pressed;
182         lcd.print("*");
183     }
184     if (i == 4){
185         delay(200);
186         if (!(strcmp(kod_wprowadzony, kod_domyslny, 4)))
187         {
188             lcd.clear();
189             lcd.setCursor(0, 0);
190             lcd.print("Blokada 3/3");
191             lcd.setCursor(0, 1);
192             lcd.print("Kod zgodny");
193             delay(3000);
194             sg90.write(90);
195             digitalWrite(diodaZielona, HIGH);

```

```

196         lcd.clear();
197         lcd.setCursor(1, 0);
198         lcd.print("Drzwi otwarte");
199         lcd.setCursor(7, 1);
200         lcd.print("5");
201         delay(1000);
202         lcd.setCursor(7, 1);
203         lcd.print("4");
204         delay(1000);
205         lcd.setCursor(7, 1);
206         lcd.print("3");
207         delay(1000);
208         lcd.setCursor(7, 1);
209         lcd.print("2");
210         delay(1000);
211         lcd.setCursor(7, 1);
212         lcd.print("1");
213         delay(1000);
214         lcd.setCursor(7, 1);
215         lcd.print("0");
216         delay(1000);
217
218         digitalWrite(diodaZielona, LOW);
219         sg90.write(0);
220         lcd.clear();
221         i = 0;
222         TempMode = true;
223     }
224
225     else
226     {
227         lcd.clear();
228         lcd.setCursor(0, 0);
229         lcd.print("Blokada 3/3");
230         lcd.setCursor(0, 1);
231         lcd.print("Kod niezgodny");
232         digitalWrite(diodaCzerwona, HIGH);
233         delay(3000);
234         digitalWrite(diodaCzerwona, LOW);
235         lcd.clear();
236         i = 0;
237         TempMode = true;
238     }
239 }
240 }
241 }

```

Listing 5: Kod przedstawiający implementację i działanie kodu numerycznego.

W pierwszej kolejności zostaje utworzona instrukcja warunkowa "if" dla flagi RFID równej false (Listing 5, linia 178). Następnie przechwytywane są znaki wpisywane przez użytkownika na klawiaturze, a wpisywane dane zostają gwiazdkowane.

Po wpisaniu przez użytkownika 4 znaków program porównuje je z ustawionym hasłem (Listing 5, linia 184). W przypadku wpisania poprawnego hasła, użytkownik dostaje informację o jego poprawności i otwarciu zamka. Towarzyszy temu zapalona zielona dioda oraz stoper wyświetlający informację o pozostałym czasie otwarcia zamka (Listing 5, linia 198-223).

Kiedy użytkownik poda błędne hasło, program wyświetli odpowiednią informację oraz na 3 sekundy zostanie zapalona dioda LED w kolorze czerwonym (Listing 5, linia 225-241).

Odnosnik do całego kodu- [Dokumentacja](#)

5 Załączniki

6 Bibliografia

Literatura

- [1] <https://github.com/miguelbalboa/rfid>
- [2] <https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library>
- [3] <https://playground.arduino.cc/Code/Keypad/>
- [4] <https://www.arduino.cc/reference/en/libraries/servo/>
- [5] <https://www.arduino.cc/en/reference/SPI>
- [6] <https://github.com/PaulStoffregen/OneWire>
- [7] <https://github.com/nettigo/DS18B20>