Univerzitet u Sarajevu

Elektrotehnički fakultet
Odsjek za automatiku i elektroniku

Realizacija alarmnog sistema pomoću Arduino Uno mikrokontrolera

Studenti: Adin Heco (18 362), Faris Baždar (18 413)
Odgovorni nastavnik: Red. prof. dr. Jasmin Velagić
Predmet: Mehatronika
Akademska godina: 2020/2021

Sarajevo, 2021.

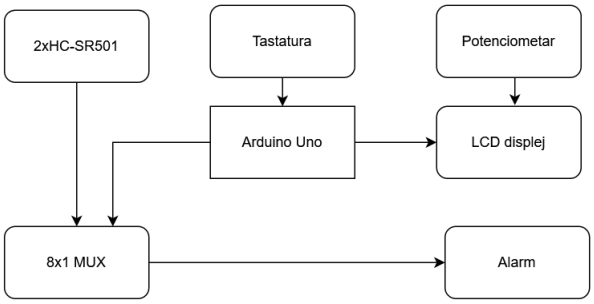
Sadržaj

Popis slika

1 Uvod

U seminarskom radu ćemo realizovati alarmni sistem baziran na Arduino Uno mikrokontroleru. Alarm treba osigurati objekat od mogućih neželjenih ulazaka kroz vrata ili prozore, a ujedno treba da spriječi neovlašteno kretanje kroz objekt. Radi minimalizacije troškova, koristit ćemo detektore pokreta.

1.1 Blok dijagram sistema

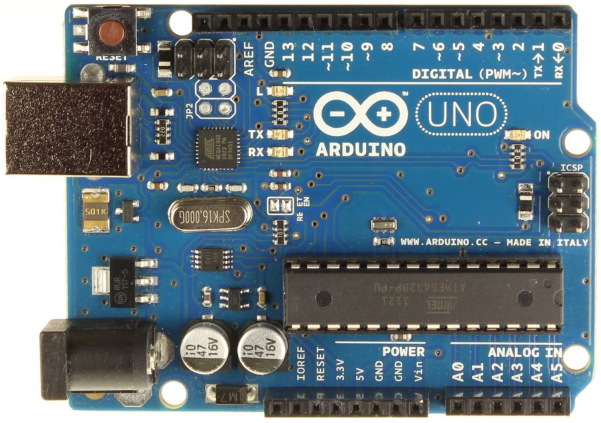


Slika 1

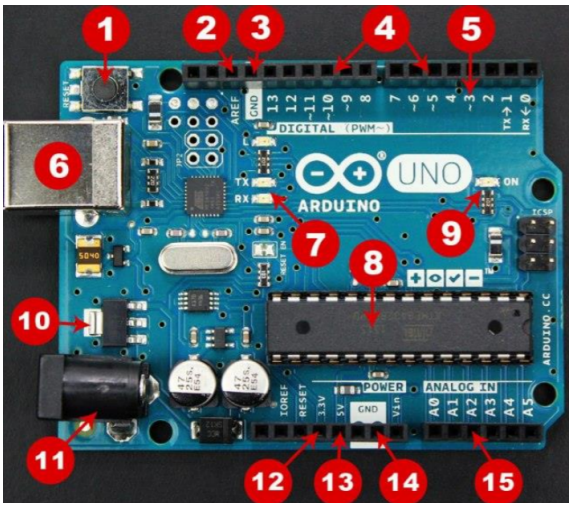
2 Tehnika

2.1 Arduino

Arduino predstavlja open-source programabilnu štampanu ploču koja se može koristiti za realizaciju mnogih projekata, od jednostavnijih do složenijih. Arduino sadrži mikrokontroler kojeg korisnik programira. Povezivanjem Arduina, senzora i ulaznih veličina, moguće je ostvariti interakciju sa izlazima, kao što su LED diode, motorima, displejima... Zbog svoje niske cijene, Arduino je danas u širokoj upotrebi (najviše se koristi Uno).



Slika 2: Arduino Uno



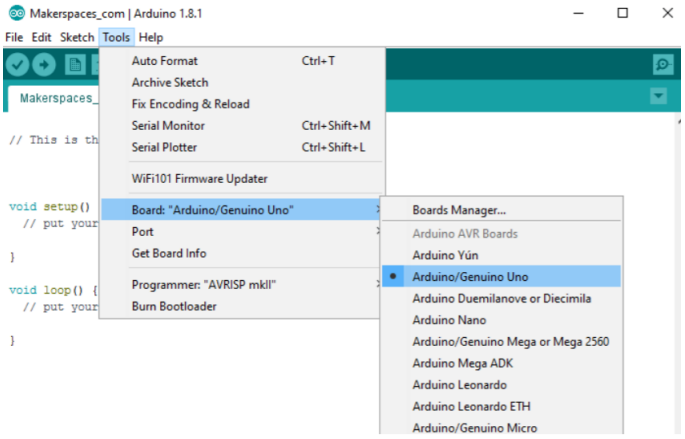
Slika 3: Komponente Arduina Una

Na slici 3 su prikazani pojedini elementi Arduina, a to su:

- (1) Reset taster - resetuje bilo koji kod koji je Arduino učitao
- (2) AREF (eng. Analog Reference) - postavlja referentni vanjski napon
- (3) Pin za masu - postoji više pinova koji predstavljaju masu i obavljaju istu funkciju
- (4) Digitalni ulaz/izlaz (input/output) - pinovi od 0 do 13 se mogu koristiti kao digitalni ulazi/izlazi
- (5) PWM (širinsko impulsna modulacija) - pinovi oznaceni sa ~ mogu simulirati analogne izlaze
- (6) USB konektor - služi za napajanje Arduina i za upload programskog koda
- (7) TX/RX - LED diode koje sluze da signaliziraju prenos i primanje podataka
- (8) ATmega mikrokontroler - predstavlja upravljačku jedinicu u koju se smješta programski kod
- (9) LED dioda koja se pali ukoliko je Arduino priključen na izvor napajanja
- (10) Naponsko regulator - reguliše nivo napona koji dolazi na štampanu ploču
- (11) Mjesto za baterijsko DC napajanje Arduina
- (12) 3.3V pin - služi za napajanje komponenti sa naponom 3.3V
- (13) 5V pin - služi za napajanje komponenti sa naponom 5V
- (14) Pin za masu
- (15) Analogni pinovi - očitavaju analogni signal sa senzora i konvertuju isti u digitalni kroz A/D konverziju

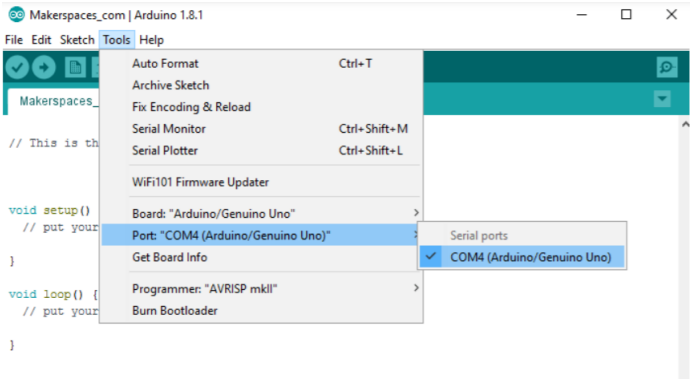
2.2 Razvoj softvera u Arduino IDE

Arduino softver je besplatan i može se skinuti na linku: https://www.arduino.cc/en/Main/Software. Prije pisanja bilo kakvog koda, potrebno je u taskbar-u otići na opciju **Tools -> Board** i odabrati koju vrstu Arduina koristimo.



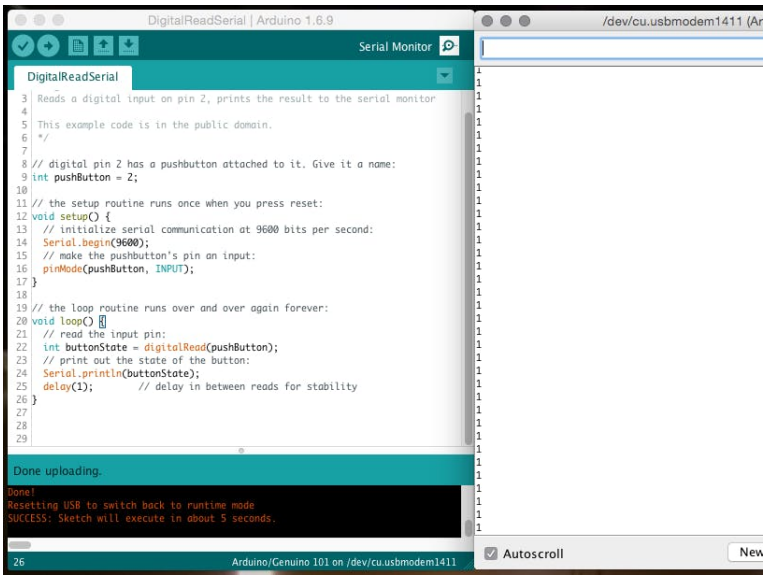
Slika 4

Nakon što smo definisali vrstu Arduina koju koristimo, potrebno je ponovo otići na opciju **Tools -> Port**, a onda odaberemo port na kojem piše Arduino.



Slika 5

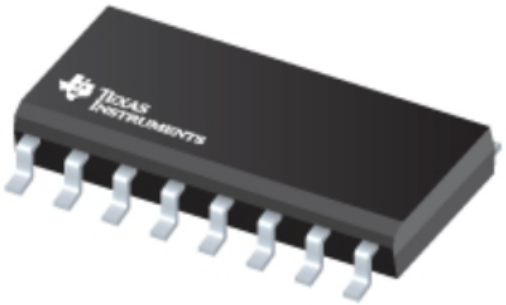
Svaki Arduino program ima dvije glavne funkcije, ***setup()*** i ***loop()***. Funkcija *setup()* se poziva samo jednom pri izvršavanju koda i u njoj se obično vrši inicijalizacija pinova. Funkcija *loop()* se poziva beskonačno puta i obavlja zadane instrukcije, sve dok se Arduino ne ugasi. Programiranje u Arduino je slično programiranju u jeziku C. Arduino također pruža mogućnost prikaza vrijednosti ili teksta na displeju, odnosno Serial monitoru, koji se nalazi u gornjem desnom uglu programskog okruženja. Da bi omogućili serijsku komunikaciju, koristimo naredbu *Serial.begin(9600)*. U ovom slučaju, brzina slanja podataka je 9600 bita po sekundi. Naredba kojom ispisujemo vrijednosti ili tekst na serijski monitor je *Serial.print() ili Serial.println()*. Razlika između ove dvije naredbe je ta što druga, nakon ispisa na ekran, automatski prelazi u novi red.



Slika 6

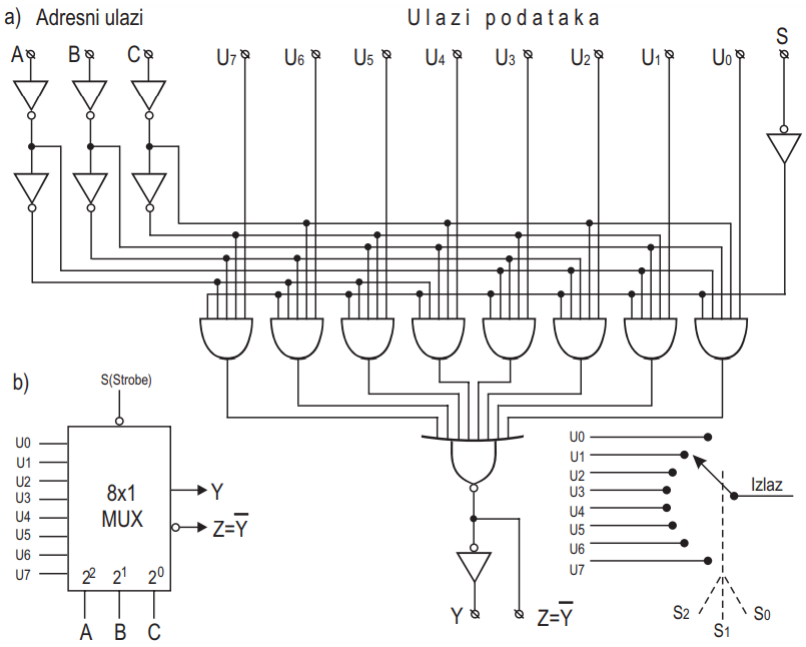
2.3 Multiplekser

Multiplekser je modul MSI tehnike koji funkcionira analogno selektorskom prekidaču. Na sljedećoj slici je prikazana elektronska shema i simbol osmoulaznog troadresnog multipleksera, sa direktnim izlazom Y, kao i njegovim komplementom. [1]



Slika 7: Izgled multipleksera 8x1

[1] Dokumentacija za relej: https://www.ti.com/lit/ds/symlink/sn74hc151-q1.pdf?ts=1615741812371&ref_url=https%253A%252F%252Fwww.google.com%252F



Slika 8: a) Logički dijagram 8x1 MUX-a
b) Simbol 8x1 MUX-a

ULAZI												IZL.	
A	B	C	S	U_0	U_1	U_2	U_3	U_4	U_5	U_6	U_7	Y	Z
x	x	x	1	x	x	x	x	x	x	x	x	0	1
0	0	0	0	0	x	x	x	x	x	x	x	0	1
0	0	0	0	1	x	x	x	x	x	x	x	1	0
0	0	1	0	x	0	x	x	x	x	x	x	0	1
0	0	1	0	x	1	x	x	x	x	x	x	1	0
0	1	0	0	x	x	0	x	x	x	x	x	0	1
0	1	0	0	x	x	1	x	x	x	x	x	1	0
0	1	1	0	x	x	x	0	x	x	x	x	0	1
0	1	1	0	x	x	x	1	x	x	x	x	1	0
1	0	0	0	x	x	x	x	0	x	x	x	0	1
1	0	0	0	x	x	x	x	1	x	x	x	1	0
1	0	1	0	x	x	x	x	x	0	x	x	0	1
1	0	1	0	x	x	x	x	x	1	x	x	1	0
1	1	0	0	x	x	x	x	x	x	0	x	0	1
1	1	0	0	x	x	x	x	x	x	1	x	1	0
1	1	1	0	x	x	x	x	x	x	x	0	0	1
1	1	1	0	x	x	x	x	x	x	x	1	1	0

Slika 9: Tabela istine za 8x1 MUX

Za realizaciju alarmnog sistema ćemo koristiti 8x1 MUX, jednu LED diodu, tastaturu za unos šifre koja omogućava aktivaciju alarma i zvučnik. [2]

[2]Dokumentacija: https://html.alldatasheet.com/html-pdf/26984/TI/74HC4051/427/16/74HC4051.html

2.4 Senzor HC SR501

Ovaj senzor ćemo koristit za detekciju pokreta. Napajanje je 5V. Senzor HC-SR501 koristi LHI778 pasivni infracrveni senzor i BISS0001 IC za način detekcije pokreta. Senzor ima mogućnost podešavanja udaljenosti do koje se mjeri pokret (od 3m do 7m). [3]

Slika 10: Izgled senzora HC-SR501

2.5 Arduino tastatura

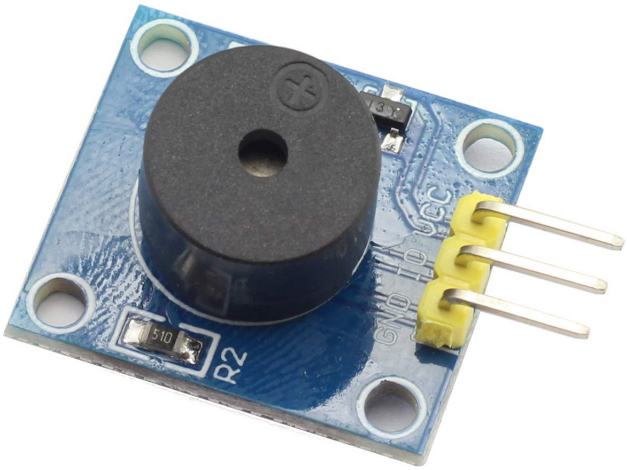
Tastaturu ćemo koristit za unos i izmjenu šifre potrebne za aktivaciju i deaktivaciju alarma. Za potrebe ovog projekta, mi zapravo trebamo 3x4 tastaturu.

Slika 11: Izgled Arduino tastature

2.6 Buzzer

Buzzer nam služi za zvučnu signalizaciju, nakon što senzori pokreta detektuju kretanje, dok je alarm aktiviran. Buzzer ima tri pina, jedan pin se konektuje na masu, drugi na napajanje, a treći pin ima ulogu I/O (input/output).

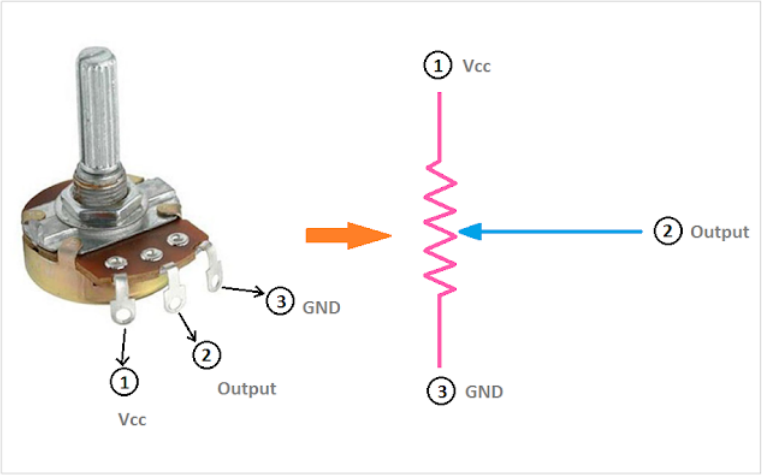
[3]Dokumentacija: https://www.epitran.it/ebayDrive/datasheet/44.pdf



Slika 12: Izgled buzzera

2.7 Potenciometar

Potenciometar ćemo koristiti za adaptaciju kontrasta, odnosno jačine svjetla na displeju.

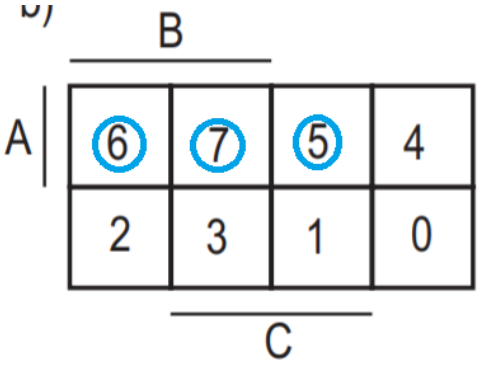


Slika 13: Izgled i shema potenciometra

3 Princip rada alarmnog sistema

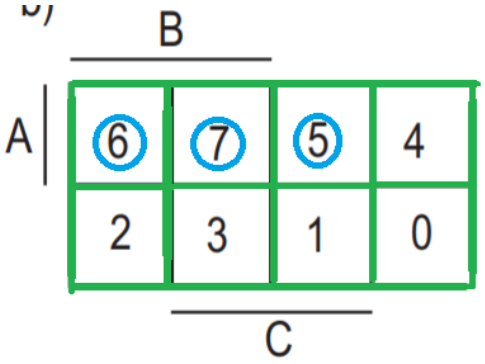
Alarmni sistem smo realizovali preko digitalnog pina 2 i 8x1 multipleksera. Multiplekser ima tri adresne varijable jer ovdje koristimo tip 0 realizacije ($2^{n-0}x1$, gdje n predstavlja broj adresni varijabli), i konstantne ulaze (logička nula ili jedinica). Sistem ima tri bita. Najznačajniji bit nam govori da li je alarm upaljen, tj. da li je korisnik aktivirao alarm (ukoliko je bit 1, znači da je alarm upaljen, ukoliko je 0, onda je ugašen). Da bi korisnik aktivirao alarm potrebnu je unijeti odgovarajuću šifru. Korisnik pri prvom porketanju alarma na LCD displeju dobiva obavijest da unese šifru (unos šifre se završava kada korisnik unese onoliko karaktera koliko ih šifra sadrži). Početna šifra u ovome slučaju je *456*, a korisnik bi je dobio prilikom kupovine alarma. Ukoliko bi korisnik želio promijeniti šifru, potrebno je pritisnuti "*". Nakon toga, na displeju će se pojaviti obavijest da korisnik treba unijeti staru šifru, a onda unijeti novu. U ovom slučaju potvrda unosa završava sa znakom "#". Na LCD displeju korisnik dobiva obavijest da je šifra uspješno promijenjena.
Sljedeća dva bita se odnose na to, da li su vrata ili prozor otvorena, tj. da li je detektovan pokret. Da bi detektovali pokret, koristimo senzor HC SR501. Znači ukoliko je senzor detektovao pokret na vratima ili na prozoru ili oboje, posljednja dva bita prelaze iz 0 na 1, te se pali buzzer. Zelena LED dioda se pali kada korisnik aktivira alarm, odnosno kada je najznačajniji bit postavljen na 1. Slučajevi koji aktiviraju alarm u slučaju provale,su sljedeći: 110, 101 ili 111. Da bi odredili način spajanja konstanti na ulaze MUX-a, koristit ćemo Veitchove dijagrame. Postupak je sljedeći:

1. Unutar Veitchovog dijagrama odredimo mjesta gdje su konstituente jedinica. U našem slučaju, to su binarno 110, 101 i 111, kojima odgovara dekadni ekvivalent 6, 5 i 7, respektivno. Na sljedećoj slici su zaokružene konstituente jedinica:



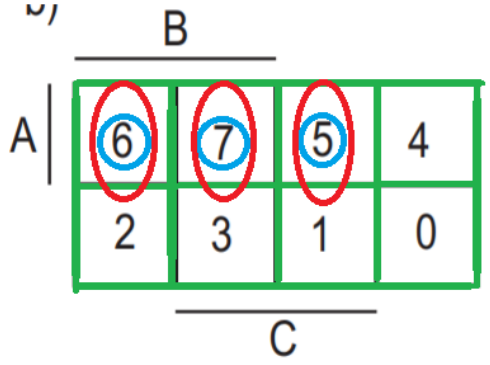
Slika 14

2. Pošto imamo 3 adresne varijable, granice između dijelova omeđenih sa A, B i C podebljamo, kao na sljedećoj slici:



Slika 15

3. Sada je potrebno konturom obuhvatiti što veći broj konstituenti jedinica. Broj obuhvaćenih konstituenti mora odgovarati stepenu broja 2. Međutim, kako su sve granice podebljane, mi ne možemo obuhvatiti više od jedne konstituente. Način zaokruživanja kontura je prikazan na sljedećoj slici:

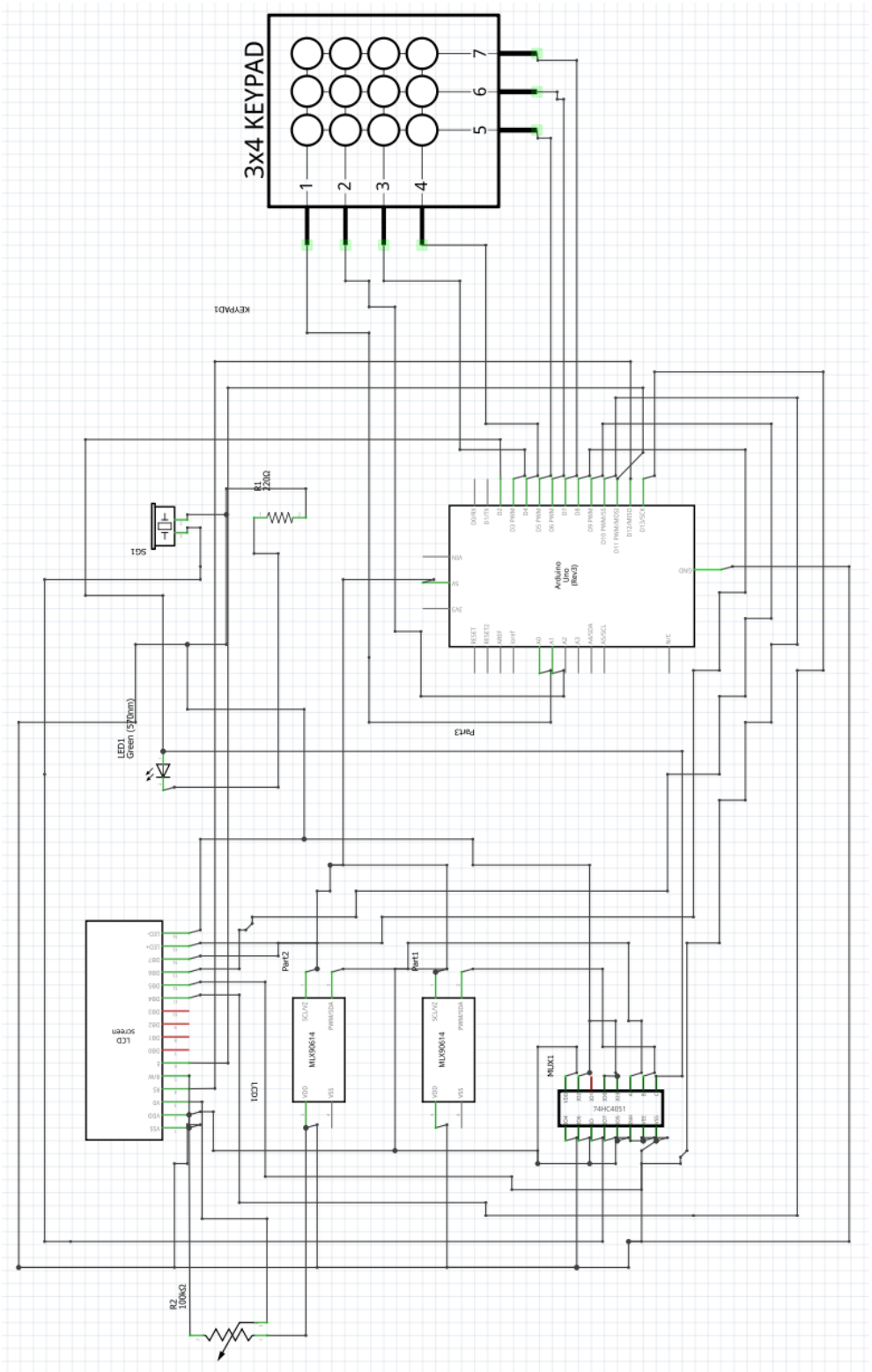


Slika 16

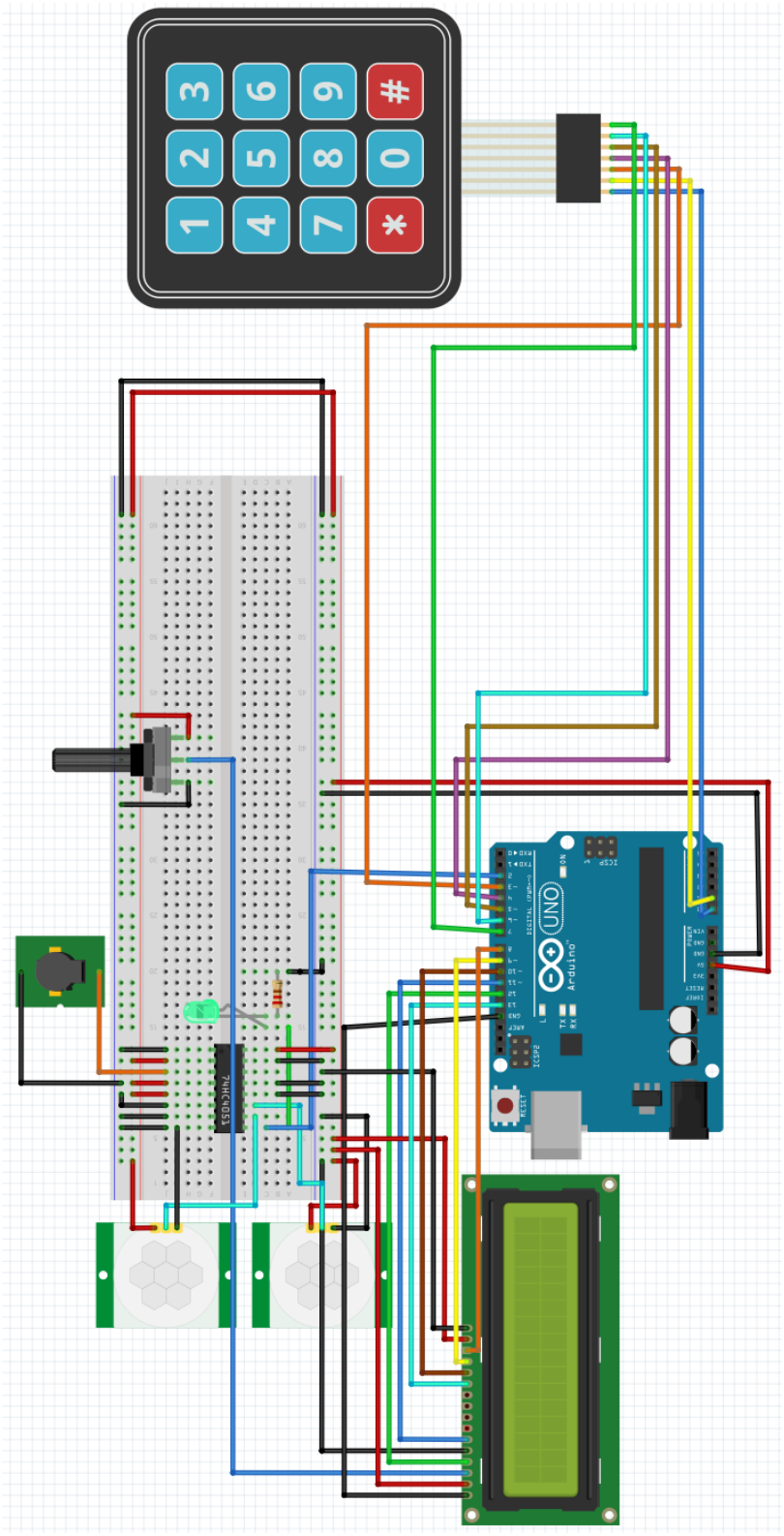
4. Obzirom da konture koje su obuhvatile konstituente jedinica nalaze na pozicijama 5, 6 i 7, znači da će prvih 5 ulaza biti logička nula, a posljednja 3 logička jedinica.

Na osnovu prethodno opisanog postupka, obavili smo realizaciju alarmnog sistema.

4 Sheme



Slika 17



Slika 18

5 Programski kod

Za pristup ".ino" datoteci, možete otići na link:
https://drive.google.com/file/d/10_z9MACbwRfYvPMTjiXaIsThyw1ACIEQ/view?usp=sharing

mehatronika_seminarski

```
#include <Wire.h>
#include <LiquidCrystal.h>
#include <Keypad.h>
#include <string.h>

#define max_duzina_sifre 17 //mjesta ima na displeju da prikazemo uneseno (u drugom redu); 17. je \0

char uneseno[17];
char sifra[17] = "456"; //pocetna sifra
byte uneseno_brojac = 0;
int duzina_sifre = 3;
char tipka;

const byte redovi = 4;
const byte kolone = 3;

bool preskok = 0;

char tipke[redovi][kolone] = {
  {'1', '2', '3'},
  {'4', '5', '6'},
  {'7', '8', '9'},
  {'*', '0', '#'}
};
```

Slika 19

```
#define pin 2

byte red_pinovi[redovi] = {A1, A0, 3, 4};
byte kolona_pinovi[kolone] = {5, 6, 7};

bool flag = false;

Keypad key = Keypad(makeKeymap(tipke), red_pinovi, kolona_pinovi, redovi, kolone);

const int RS = 12, E = 11, d4 = 13, d5 = 10, d6 = 9, d7 = 8;
LiquidCrystal lcd(RS, E, d4, d5, d6, d7);

void clear_uneseno() { //funkcija za brisanje unesenog stringa
  while (uneseno_brojac != 0) {
    uneseno[uneseno_brojac--] = 0;
  }
  flag = true;
}
```

Slika 20

```
void deaktivacija() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Unesi sifru D:"); //D kao deaktivacija
  uneseno_brojac = 0;
  while (1) {
    tipka = key.getKey();
    if (tipka == '*' && tipka) return; //napustamo ovu funkciju
    else if (tipka) {
      uneseno[uneseno_brojac] = tipka; //u string uneseno postavljamo karaktere sifre koje korisnik unosi
      lcd.setCursor(uneseno_brojac, 1);
      lcd.print(uneseno[uneseno_brojac]);
      uneseno_brojac++; //kako unosimo sifru, povecava se broj unesenih karaktera
      if (uneseno_brojac == duzina_sifre) break;
    }
  }
  if (uneseno_brojac == duzina_sifre && uneseno_brojac <= max_duzina_sifre) {
    lcd.clear();
    if (!strcmp(uneseno, sifra)) {
      lcd.setCursor(0, 0);
      lcd.print("Ok sifra");
      lcd.setCursor(0, 1);
      lcd.print("Alarm D"); //D-deaktiviran
      digitalWrite(pin, 0); //deaktivacija alarma
      delay(1000);
      //Serial.println(digitalRead(1));
```

Slika 21

```
      //Serial.println(digitalRead(1));
    }
    else {
      lcd.print("Netacna sifra");
      delay(1000);
    }

    lcd.clear(); //brisemo ekran
    clear_uneseno();  //funkcija za brisanje "uneseno" stringa
  }
}


void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);
  pinMode(pin, OUTPUT); //pali alarm i zelenu diodu
  digitalWrite(pin, 0);
  //delay(3000);
}
```

Slika 22

```
void loop() {
  if (flag) {
    lcd.clear();
    flag = false;
  }
  lcd.setCursor(0, 0);
  lcd.print("Unesi sifru A:"); //A kao aktivacija

  tipka = key.getKey(); //pritisak tipke

  //UNOS NOVE SIFRE
  if (tipka && tipka == '*') { //ako smo pritisnuli *, onda unosimo novu sifru
    lcd.clear(); //brisemo ekran
    lcd.setCursor(0, 0);
    lcd.print("Unos stare sifre");
    delay(1000);
```

Slika 23

```
  while (1) {
    tipka = key.getKey();
    if (tipka == '*') return; //restart unosa, korisnik opet mora pritisnut * da bi unio novu sifru
    else if (tipka == '#') break; //napustamo do-while petlju kad potvrdimo unos sa #
    if (tipka) {
      uneseno[uneseno_brojac] = tipka; //u string uneseno postavljamo karaktere sifre koje korisnik unosi
      lcd.setCursor(uneseno_brojac, 1);
      lcd.print(uneseno[uneseno_brojac]); //printa karaktere na displej
      uneseno_brojac++; //kako unosimo sifru, povecava se broj unesenih karaktera
      if (uneseno_brojac >= max_duzina_sifre) break; //ako smo presli max duzinu ili je dostigli, kraj unosa
    }
  }

  if (uneseno_brojac == duzina_sifre && uneseno_brojac <= max_duzina_sifre) {
    lcd.clear();
    if (!strcmp(uneseno, sifra)) {
      lcd.print("Ok sifra");
      delay(1000);
      duzina_sifre = 0; //duzina sifre je sad nula jer unosimo novu
    }
    else {
      Serial.println(uneseno_brojac);
      lcd.print("Netacna sifra");
      delay(1000);
      return; //sve ponovo
    }
```

Slika 24

```
  lcd.clear();
  lcd.setCursor(0, 0);

  lcd.print("Unesi novu sifru");
  delay(1000);
  uneseno_brojac = 0;
  while (1) {
    tipka = key.getKey(); //mora sifra samo imati brojeve i mora biti manja od 17 kolko moze stat u 1 red displeja
    if (tipka == '*') return; //sve ponovo, korisnik 2x mora pritisnu * da bi ponovo unio sifru
    else if (tipka == '#') break; //kraj unosa sa potvrdom
    if (tipka) {
      duzina_sifre++; //unosom sifre povecava se njena duzina (logicno)
      Serial.print("Vrijednost brojaca je: ");
      Serial.println(uneseno_brojac);
      if (duzina_sifre == max_duzina_sifre) return; //ne mozemo dalje unositi karaktere
      uneseno[uneseno_brojac] = tipka; //unosimo u string karaktere
      lcd.setCursor(uneseno_brojac, 1);
      lcd.print(uneseno[uneseno_brojac]); //ispis na displej
      uneseno_brojac++;
    }
  }
  strcpy(sifra, uneseno);
  clear_uneseno();
}
lcd.clear();//NOVA IZMJENA
lcd.setCursor(0, 0);
lcd.print("Nova sifra ok");
```

Slika 25

```
  delay(2000);
  lcd.clear();
  return; //preskacemo sve ostalo i ponovo se poziva loop()
}

//STANDARDNI UNOS SIFRE
//ovde uslov koji gleda jel pin 0 na keca
if (digitalRead(pin) != HIGH) {
  if (tipka && tipka != '#' && tipka != '*') {
    uneseno[uneseno_brojac] = tipka; //u string uneseno postavljamo karaktere sifre koje korisnik unosi
    lcd.setCursor(uneseno_brojac, 1);
    lcd.print(uneseno[uneseno_brojac]);
    uneseno_brojac++; //kako unosimo sifru, povecava se broj unesenih karaktera
  }

  if (uneseno_brojac == duzina_sifre && uneseno_brojac <= max_duzina_sifre) {
    lcd.clear();
    if (!strcmp(uneseno, sifra)) {
      lcd.setCursor(0, 0);
      lcd.print("Ok sifra");
      lcd.setCursor(0, 1);
      lcd.print("Alarm A"); //A-aktiviran alarm
      digitalWrite(pin, 1); //aktivacija alarma
      delay(1000);
      //Serial.println(digitalRead(1));
      //Serial.println(digitalRead(0));
```

Slika 26

```
        }
        else {
          lcd.print("Netacna sifra");
          delay(1000);
        }

        lcd.clear(); //brisemo ekran
        clear_uneseno();  //funkcija za brisanje "uneseno" stringa
      }
    }//kraj ifa za provjeru aktiviranog alarma
    //deaktivacija alarma
    else deaktivacija();
}
```

Slika 27

5.1 Komentar na kod

U programu je definirana maksimalna duzina sifre koja se moze pohraniti, te startna sifra pri paljenju alarmnog sistema. Funckija clear_uneseno() se koristi da bi se resetovao brojac i nulirao uneseni string. Funkcija deaktivacija() služi za pokretanje potrebnog interfejsa za deaktivaciju alarma. Funkcije lcd.setCursor() i lcd.print() zajedno sa funkcijom lcd.clear() služe za vizalni prikaz trenutnog stanja u kojem se nalazimo (alarm aktiviran, alarm deaktiviran, promjena šifre). Nakon što je izvršen unos šifre u funkciji deaktivacija() vrši se provjera te šifre sa prethodno pohranjenom, to se vrši putem funkcije strcmp(str1, str2) koja daje vrijednost 0 ukoliko su stringovi isti. Ukoliko je šifra ispravna deaktivira se alarm, tj. napajanje mux-a se prekida, ukoliko se šifra ne poklapa sa prethodno pohranjenom onda se signalizira da je došlo do greške pri unosu. U setup() funkciji se vrše početne postavke lcd-a i pojedinih OUTPUT pinova. Unutar loop() funkcije se izvršava glavni kod programa. Unutar ove funckije omogućen je unos nove šifre ukoliko se unese znak "*", nakon unosa stare šifre kao potvrdu da je korisnik ovlašten da promijeni šifru, omogućava se unos nove šifre koji se potvrđuje unosom znaka "#". Upis nove šifre se vrši putem funkcije strcpy(). Korištene su funkcije delay() kako bi se mogle izvršiti vremenske zadrške nekog teksta. U svakom trenutku izvršavanja programa vrši se provjera u kojem se stanju nalazimo na osnovu unosa sa tastature i trenutnog stanja našeg programa.